UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN

IMPLEMENTACIÓN DE UN SISTEMA CRM/ERP EN SALESFORCE

Abril - 2026

AUTOR: Fabio Barcelona García
DIRECTOR: Jesús Javier Rodríguez Sala

RESUMEN

Este trabajo describe el diseño, desarrollo e implantación de un sistema CRM/ERP sobre Salesforce para una empresa ficticia de venta de productos tecnológicos llamada IT Solutions. El supuesto parte de una situación bastante común en empresas medianas: gestiona sus clientes con hojas de Excel, el inventario con un software básico que no esta conectado con nada, la facturación con un programa independiente que obligaba a introducir los datos dos veces, y la atención al cliente se realiza únicamente por email sin ningún sistema de seguimiento centralizado. Todo esto generaba ineficiencias, duplicidades y una falta de visibilidad global del negocio que dificulta crecer de forma ordenada.

Para resolver esta situación se ha elegido Salesforce tras compararlo con otras cinco alternativas del mercado (Microsoft Dynamics 365, Odoo, SAP, Zoho y HubSpot). La elección se justifica principalmente por la experiencia profesional previa del autor con la plataforma, aunque también influyen otros factores como la posibilidad de migrar a la nube sin un desembolso inicial muy elevado, la capacidad de personalización tanto con herramientas declarativas como con código, y el hecho de que Salesforce lleva varios años consecutivos siendo el líder del mercado CRM.

El proyecto se ha desarrollado siguiendo una metodología iterativa e incremental en cuatro fases a lo largo de 16 semanas. En la primera fase se configuró Sales Cloud para gestionar todo el ciclo comercial, desde la captación del Lead hasta la generación del pedido. En la segunda se implantó Service Cloud para la atención al cliente, con colas de soporte, Email-to-Case y una base de conocimiento. La tercera fase se dedicó a Experience Cloud, donde se creó un portal web para que los clientes puedan consultar sus pedidos y el estado de los envíos. En la cuarta fase se desarrollaron las automatizaciones, la integración con una API externa de seguimiento logístico y el proceso batch de inventario.

Desde el punto de vista técnico, la solución sigue una arquitectura de tres capas (presentación, lógica de negocio y datos) y utiliza tres productos de Salesforce de forma simultánea. Para cubrir funcionalidades propias de un ERP que Salesforce no ofrece de forma nativa se crearon dos objetos personalizados para la gestión del stock y para el seguimiento de envíos. Las pruebas unitarias alcanzan una cobertura superior al 75% exigido por la plataforma para despliegues a producción.

Como conclusión general, el proyecto demuestra que Salesforce es una opción viable y competitiva para implementar un sistema CRM/ERP en una empresa mediana, permitiendo centralizar todos los procesos de negocio en una sola plataforma cloud sin necesidad de gestionar infraestructura propia.

AGRADECIMIENTOS

Quiero agradecer a mi familia todo el apoyo que me ha dado a lo largo de estos años, no solo durante el grado sino en general, en especial a Jesica, Ramón y sobre todo a mi madre. Sin ella no sería quien soy hoy.

También quiero dar las gracias a mi tutor Jesús Javier por la orientación durante el desarrollo del trabajo.

Por último, quiero agradecer a mis amigos, incluso a los que ya no forman parte de mi vida, que de una forma u otra han formado parte de este camino.

Para mi padre y mi abuela, que ojalá pudieran verlo.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Capítulo 1
Introducción

1.1.- INTRODUCCIÓN A LOS SISTEMAS CRM y ERP

Conocer a los clientes siempre ha sido una de las bases más importantes para garantizar el éxito en cualquier actividad empresarial, independientemente del sector o del tamaño de la compañía. Aunque en la actualidad el término CRM, que proviene del inglés Customer Relationship Management (en español, Gestión de Relaciones con los Clientes), pueda sonar novedoso y esté estrechamente vinculado al uso de herramientas tecnológicas avanzadas, la esencia de esta práctica es tan antigua como el comercio mismo. Desde los primeros intercambios comerciales, entender quién poseía un determinado producto o servicio, así como conocer la ubicación de personas o empresas potencialmente interesadas en adquirirlo, ha sido un factor clave para llevar a cabo cualquier negociación. Tradicionalmente, una de las maneras más sencillas y efectivas de conservar y organizar esta información ha sido mediante la creación de listas físicas, donde se registran los datos

de clientes y/o proveedores. Estas listas no solo ayudan a estructurar la información, sino que también facilitan el acceso rápido a contactos de interés, permitiendo a los comerciantes establecer relaciones más sólidas y personalizadas con sus clientes, proveedores o socios estratégicos.

Esta técnica de crear fichas con información relevante de clientes ha evolucionado y perfeccionado con el tiempo. Incluso algunas empresas llegaron a utilizar superordenadores en los años 70 y 80 para recopilar y digitalizar información. Sin embargo, no fue hasta los años 90 cuando la digitalización se popularizó y aparecieron los primeros softwares dedicados a la gestión de relaciones con los clientes, con la fundación de la empresa **Siebel Systems** en 1993, liderada por el empresario pionero Tom Siebel [1][2].

En general, un CRM se puede definir como un software para gestionar y analizar las interacciones con clientes actuales y potenciales, teniendo en cuenta el proceso de negocio que puede variar según el sector, empresa, canales de venta, entre otras variables. El objetivo principal de este software es lograr una mayor satisfacción del cliente, lo que se traduce en una mayor fidelización hacia una marca o línea de productos, una mejor relación de la empresa con el cliente y, en última instancia, un incremento en los beneficios empresariales.

Por otro lado, contamos con los sistemas ERP (Enterprise Resource Planning, Planificación de Recursos Empresariales en español). Un ERP es un tipo de software (comúnmente dividido en módulos o aplicaciones de negocio que coexisten dentro de un mismo sistema) que provee funcionalidades como automatización de procesos multidisciplinares dentro de una empresa.

Cada módulo de un ERP suele estar ligado a un área específica dentro de una empresa, por ejemplo finanzas (contabilidad, presupuestos), recursos humanos (imputación de horas de los trabajadores, nóminas, concesión de dietas), logística (seguimiento de pedidos desde el origen hasta el destino) o gestión de inventario son ejemplos clásicos donde se podría integrar un software de planificación de recursos empresariales, lo que permite a las empresas gestionar todas sus operaciones de manera eficiente y centralizada desde una misma plataforma.

Otra característica común de los sistemas ERP es la integración con otros sistemas especializados en tareas más específicas, como por ejemplo un CRM. De esta forma, las empresas pueden obtener una visión global de todos sus procesos internos y la información de sus clientes, simplificando el entramado tecnológico de la propia empresa al centralizar la información, facilitando la toma de decisiones estratégicas y mejorando la experiencia de los clientes.

La evolución de estas herramientas ha estado marcada por avances tecnológicos y la necesidad de gestionar procesos de manera más eficiente. Para entender cómo se ha llegado a los CRMs y ERPs actuales, es necesario analizar el proceso de digitalización a lo largo del tiempo.

1.1.1.- Proceso de digitalización

La evolución de los sistemas ERP fue similar a la de los CRMs. Se comenzó a digitalizar en los años 70; en concreto en el sector industrial, bajo el nombre de MRP (Manufacturing Requirements Planning, Planificación de Requisitos de Fabricación en español), haciendo uso de superordenadores y sistemas distribuidos, una infraestructura que aunque era más rápida que la gestión manual, era muy cara y requería de una gran superficie donde colocar los ordenadores [3].

En los años 80 y 90, el sector de la fabricación era cada vez más competitivo, lo que creó nuevas necesidades y urgencia en desarrollar nuevas soluciones. Es así como nació MRP II y más tarde los ERP, una versión mejorada que integraría procesos de negocio multidepartamentales, como por ejemplo Recursos Humanos, Contabilidad, etc, y que no solo se enfocaría en el sector de la fabricación, sino que sería aplicable para un rango mayor de industrias [4].

1.1.2.- Situación actual de los CRMs y ERPs

A día de hoy, tanto los sistemas CRM como los ERP intentan destacar entre la competencia implementando soluciones con tecnología de vanguardia, tales como Inteligencia Artificial, IoT, Procesamiento del Lenguaje Natural (NLP, del inglés Natural Language Processing), Cloud Computing, mayor funcionalidad Out-of-the-Box, herramientas low-code/no-code, entre otros [5]. Dependiendo de la empresa que se elija, se podrá implementar de distintas maneras (Cloud, On-Premise, Hybrid).

Por último, cabe mencionar que estas herramientas no están pensadas únicamente para grandes empresas, sino que pequeñas y medianas empresas también pueden aprovecharse de las ventajas que supone la implementación de un sistema CRM o ERP.

En el caso de las PyMEs, la informatización ha supuesto una transformación clave, permitiendo automatizar procesos que antes requerían una gestión manual intensiva. Gracias a la adopción de soluciones en la nube y modelos de pago por suscripción, incluso las pequeñas empresas pueden acceder a herramientas avanzadas sin grandes inversiones iniciales.

El precio de la solución dependerá de las necesidades de cada uno, las personalizaciones que se requiera desarrollar para adaptar el software a los procesos de negocio de cada empresa o el tipo y número de licencias que se requiera.

En un ecosistema donde la filosofía Customer Centric es recompensada y valorada [6], disponer de herramientas que ayuden a gestionar una empresa y mejorar la relación con los clientes puede llegar a ser clave para destacar sobre otros competidores, reducir costes y tomar mejores decisiones.

Sin embargo, la implementación de estos sistemas no siempre es sencilla. Los sistemas existentes con poca resistencia al cambio, la integración de estas nuevas herramientas con las que ya están en uso y el desembolso inicial de personalizar y ajustar la solución a nuestras necesidad (si las funcionalidades Out-of-the-Box no cubren por completo nuestro caso de uso) siguen siendo retos para muchas empresas, especialmente las pequeñas.

Tabla 1.1: Comparativa entre CRM y ERP.

	CRM	ERP
Propósito	Gestionar las relaciones con clientes actuales y potenciales.	Optimizar y automatizar los procesos internos de la empresa.
Áreas de aplicación	Ventas, marketing, servicio al cliente.	Finanzas, logística, producción, recursos humanos, inventario.
Beneficio principal	Mejora la captación y retención de clientes, optimizando la experiencia del cliente.	Mejora la operativa interna, reduciendo costes y errores administrativos.
Usuarios principales	Equipos de ventas, marketing y atención al cliente.	Gerentes, contables, recursos humanos, producción y logística.
Funcionalidades clave	Gestión de contactos, automatización de ventas, marketing, seguimiento de clientes, informes y análisis.	Gestión financiera, control de inventario, planificación de producción, recursos humanos, gestión de compras y ventas.
Integración	Puede integrarse con ERP para sincronizar datos de clientes con productos y pedidos.	Puede integrarse con CRM para alinear operaciones con estrategias de ventas y clientes.
Ejemplo de software	Salesforce, HubSpot, Microsoft Dynamics 365 CRM.	SAP ERP, Oracle ERP, Microsoft Dynamics 365 ERP.

1.2.- JUSTIFICACIÓN DEL PROYECTO

Como se ha mostrado en anteriores apartados, contar con un CRM y/o ERP adaptado a las necesidades de un negocio puede representar una gran ventaja competitiva y generar o fortalecer en los clientes un sentimiento de pertenencia y lealtad con la marca.

Incluso una simple automatización como puede ser el envío de un correo electrónico felicitando el cumpleaños, puede hacer que un cliente se sienta valorado. Además, esta acción le da a una empresa la oportunidad ideal para fidelizar y aumentar la retención de dicho cliente, por ejemplo, con un descuento u oferta temporal.

Un aspecto claro es que, para implementar estrategias como la mencionada en el párrafo anterior, es fundamental contar con un lugar donde se vaya almacenando de una forma segura y estructurada la información de los clientes de la empresa. Aunque existen infinidades de softwares dedicados al almacenamiento de datos, lo óptimo es optar por uno centrado en el mundo empresarial y los clientes, es decir, un CRM.

1.2.1.- Salesforce como plataforma elegida

En el panorama actual de los sistemas CRM, disponemos de numerosas opciones, pero hay un líder indiscutible entre ellos, y este es Salesforce [7]. Esta empresa se ha consolidado como una de las plataformas CRM más potentes y versátiles del mercado, ofreciendo una amplia gama de herramientas para la gestión de clientes, automatización de procesos y análisis de datos.

Su capacidad de integración con otros sistemas, su esfuerzo por ser más que un simple CRM; ofreciendo una amplísima variedad de productos para diferentes tipos de industrias [8]. y su enfoque en ofrecer una potente solución basada en la nube mediante distintos tipos de suscripción han permitido que empresas de todos los tamaños optimicen su operativa y mejoren su relación con los clientes [9].

1.2.2.- Motivación personal del autor

La primera y principal motivación de este trabajo radica en el interés y experiencia profesional del autor con Salesforce. Durante varios años, ha trabajado en entornos internacionales y profesionales con esta tecnología y le ha resultado especialmente interesante por su flexibilidad, escalabilidad y rapidez con la que se puede conseguir una solución funcional.

Al ser una plataforma en la nube, Salesforce abstrae muchos aspectos técnicos, permitiendo a los desarrolladores, consultores funcionales y administradores centrarse en la creación de valor para el negocio sin preocuparse por infraestructura, la seguridad de la plataforma o el complejo mantenimiento que supone disponer de una plataforma de estas características.

El autor, a lo largo de su trayectoria profesional, ha tenido la oportunidad tanto de implementar proyectos desde cero como de realizar tareas de mantenimiento en proyectos de larga duración, utilizando muchas de las herramientas que ofrece la plataforma. Esta experiencia le ha permitido comprender y experimentar de primera mano sus ventajas y puntos fuertes, así como sus limitaciones y desafíos en entornos empresariales reales y de gran escala.

Por último, cabe destacar que, a fecha de redacción de este trabajo, el autor cuenta con ocho certificaciones oficiales que avalan su conocimiento y experiencia en Salesforce.

1.2.3.- Supuesto ficticio

A pesar de que el autor cuenta con amplia experiencia en el uso de esta tecnología, se ha decidido idear un supuesto ficticio para la realización de este trabajo, en lugar de basarse por completo en un proyecto real del que se haya sido partícipe. Esta decisión se debe fundamentalmente a que se ha preferido dar un enfoque prioritario a las funcionalidades estándar y Out-of-the-Box que proporciona Salesforce.

Aunque es cierto que se puede crear uno o varios flujos de trabajo sólidos y funcionales solo con funcionalidad estándar, la mayoría de proyectos y escenarios reales requieren de ciertas personalizaciones y/o de un desarrollo completo a medida, haciendo uso de tecnologías web estándar (HTML, CSS, JavaScript), de lenguajes propietarios de la plataforma (Apex) o incluso de herramientas nativas low-code/no-code (Flows).

Por lo tanto y en resumen, este proyecto buscará aprovechar al máximo las funcionalidades estándar y herramientas low-code/no-code nativas de Salesforce, combinando distintos productos de Salesforce que se detallarán en apartado posteriores.

Aunque el trabajo no quedará exento de personalizaciones de código, estas se describirán de forma breve y solo se implementarán en los casos en los que las funcionalidades estándar no ofrezcan una solución escalable y mantenible al problema, o directamente no sean suficientes para cubrir las necesidades del negocio.

1.3.- OBJETIVOS

Con este trabajo no solo se pretende realizar un análisis sobre las funcionalidades básicas de Salesforce, sino que también se desea plasmar la mayor cantidad de conocimiento y experiencia que el autor ha adquirido durante su trayectoria profesional. Además, representa una oportunidad para consolidar sus habilidades, además de investigar con más

detalle y profundidad las herramientas que ofrece la plataforma y que por características de los proyectos en los que ha participado, no ha podido hacer uso. El desarrollo de una aplicación ficticia permitirá demostrar conceptos teóricos con ejemplos prácticos.

1.3.1.- Objetivos principales

Como objetivos principales del presente proyecto se plantean los siguientes:

- Dar una explicación clara sobre qué es Salesforce, introducir su plataforma de aprendizaje, su tienda de aplicaciones y hacer una comparativa frente a otras alternativas del mercado.

- Describir brevemente las diferentes industrias en las que se puede aplicar Salesforce de forma estándar.

- Desarrollar una aplicación funcional adaptada a las necesidades de una empresa ficticia, con problemáticas y casos de uso comunes en empresas reales, en la que se necesita de múltiples productos de Salesforce (conocidos como *nubes*).

- Hacer uso de al menos tres nubes distintas, conocidas como **Sales Cloud, Service Cloud y Experience Cloud**. Para cada una de ellas se explicará su propósito, las características que comparten (si aplica), y las áreas del negocio en las que pueden aplicarse.

- Aunque Salesforce se trata por definición de un CRM, también se pretende incluir en la aplicación características de sistemas ERP (Productos, inventarios, finanzas).

1.3.2.- Objetivos secundarios

Adicionalmente, como objetivos secundarios se destacan los siguientes:

- Desarrollar una o varias funcionalidades que requieran código, como una integración con una REST API externa.

- Utilizar herramientas y extensiones no oficiales para mejorar la experiencia de desarrollo en Salesforce.

- Explorar y aplicar algún producto de **Inteligencia Artificial** desarrollado por Salesforce.

1.3.3.- Objetivos personales

Por último, los objetivos personales desde el punto de vista del autor son:

- Demostrar y reforzar la maestría y conocimientos del autor sobre Salesforce y las funcionalidades con las que ha trabajado previamente.

- Hacer uso de características y tecnologías que, por requisitos y alcance de los proyectos en los que el autor ha participado, no ha tenido la oportunidad de usar.

1.4.- LÍMITES DEL PROYECTO

Dada la gran cantidad de funcionalidad que ofrece Salesforce, se van a dejar fuera de alcance y a acotar muchos de los aspectos que requieran una preparación previa muy grande o compleja, requieran el uso avanzado de tecnología de terceros para que sean mínimamente funcionales, o el pago de suscripciones o licencias de usuario.

Por nombrar algún ejemplo, no se hará uso de **External Objects**, una característica que permite representar de forma virtual datos de otra instancia de Salesforce o de una base de datos externa accesible desde Internet, haciendo un mapeo de las columnas de una tabla de una base de datos o de un objeto (tabla) de Salesforce que resida en otra instancia contra un objeto de Salesforce desde donde se crea dicho External Object.

Otro ejemplo podría ser el **aprovisionamiento de usuarios** mediante un sistema externo, como puede ser **Microsoft Entra ID** (antes conocido como Azure Active Directory). Desarrollar esta funcionalidad requeriría de un *setup* complejo, y no está contemplado dentro del alcance del trabajo.

No se hará uso de todas las tecnologías propietarias de Salesforce. Aunque se hará una breve explicación, no se usará ni **Visualforce** ni **Aura Components**. Estas tecnologías eran muy usadas (aunque siguen teniendo su funcionalidad en casos muy concretos) para crear componentes e interfaces antes de que se introdujera a finales de 2018 la capacidad de usar lenguajes web más modernos y sencillos de usar (HTML, CSS y JavaScript basado en el estándar de Web Components [10]), haciendo uso de un framework llamado **Lightning Web Components** [11].

Tampoco se hará uso de tecnologías que, siendo obsoletas, no han sido eliminadas de la plataforma por motivos de retrocompatibilidad y para no introducir *breaking changes* u obligar a las organizaciones existentes a migrar de tecnología de forma brusca. Ejemplos

de esto pueden ser **Process Builder** o **Workflow Rules**, habiendo sido sustituidos por **Flows** [12].

Por último, queda totalmente fuera de alcance cualquier tecnología que requiera de una licencia de pago y no ofrezcan entornos de prueba. Esto aplica a ciertas nubes de Salesforce (Marketing Cloud, Commerce Cloud, Net-zero Cloud, Non-profit Cloud…) o a otros productos que han sido adquiridos previamente por Salesforce y/o requieren de una licencia o suscripción (Tableau Software, Slack, Heroku o MuleSoft).

La aplicación a desarrollar no tiene como propósito final ser comercializada o publicada, ni se pretende cumplir con todos los estándares de calidad y seguridad que Salesforce exige para hacer pública una aplicación dentro de su tienda (AppExchange).

El fin que se busca es crear una demostración funcional de distintos productos de Salesforce a través de un entorno de prueba gratuito tomando como base los requerimientos de una empresa ficticia.

Capítulo 2 Antecedentes y estado de la cuestión

En este capítulo se expondrá la situación actual de una empresa ficticia en relación a lo que se quiere abordar con este trabajo. También se incluirá una comparativa de las principales opciones para elegir un Software as a Service (SaaS) enfocado al ámbito de CRM y ERP.

2.1.- SITUACIÓN ACTUAL DE LA EMPRESA

Se supondrá una empresa mediana ficticia llamada **IT Solutions,** cuya actividad principal es la venta de productos tecnológicos (componentes informáticos, periféricos y soluciones a medida para otras empresas). Fundada en 2016, la empresa ha experimentado un crecimiento sostenido, ampliando su cartera de clientes y su catálogo de productos. Sin embargo, este crecimiento ha evidenciado diversas carencias en sus procesos internos y en la gestión de la relación con sus clientes, lo que impacta negativamente a su eficiencia operativa y su capacidad para ofrecer un servicio de calidad.

Actualmente, IT Solutions cuenta con una colección independiente de herramientas para gestionar distintas áreas de cualquier negocio:

- **Gestión de clientes**: Se realiza a través de hojas de cálculo de Excel, donde se registran manualmente los datos de los clientes, las oportunidades de venta y el historial de interacciones (llamadas, reuniones, notas) con otras empresas y/o particulares. Este sistema es propenso a errores humanos, duplicación de información y pérdida de datos.

- **Inventario**: Utilizan un software básico de gestión de inventario que no está integrado con el resto de los sistemas. Esto requiere una sincronización manual del stock actual con respecto a las ventas, además de dificultar el seguimiento de qué productos ha comprado un cliente.

- **Facturación y contabilidad**: La contabilidad se maneja mediante un programa específico que no se sincroniza con la gestión de ventas, lo que requiere una doble entrada de datos y aumenta el riesgo de errores humanos.

- **Atención al cliente**: Las solicitudes de soporte se gestionan a través del correo electrónico de la empresa, sin un sistema centralizado de seguimiento, lo que dificulta la resolución eficiente de incidencias y la creación de un historial detallado de cada cliente.

2.1.1.- Problemática principal y necesidades de mejora

Las principales dificultades que enfrenta la empresa debido a la falta de integración de sus sistemas son las siguientes:

- **Falta de visibilidad global**: La información se encuentra dispersa en múltiples plataformas, servidores y/o departamentos, lo que impide tener una visión global del estado del negocio.

- **Ineficiencia operativa**: La duplicación de tareas, como la introducción manual de datos en distintos sistemas, ralentiza los procesos y aumenta la posibilidad de errores humanos.

- **Atención al cliente deficiente**: La ausencia de un sistema de gestión de casos impide ofrecer un soporte personalizado y eficiente, a la vez que resulta engorroso trabajar con varios hilos de correos electrónicos.

- **Toma de decisiones lenta**: La falta de datos actualizados y centralizados dificulta el análisis en tiempo real y la capacidad de tomar decisiones informadas.

- **Escalabilidad limitada**: La empresa trabaja con diversos sistemas básicos e incomunicados entre sí, lo que hace que trabajar de esta forma sea un incordio y no sean capaces de tener flujos de venta, atención al cliente y gestión de inventario claros y sólidos.

El objetivo principal de la empresa es migrar a un sistema *Cloud*, cuya implementación no necesite de un elevadísimo desembolso inicial.

También desean simplificar su infraestructura actual, reduciendo el número de servidores locales y diferentes licencias.

Como requisito obligatorio, se deberá poder transicionar de un sistema a otro sin tener que interrumpir su actividad económica. Esto quiere decir que el tiempo que se dedique a idear y desarrollar la solución no deberá impactar en el funcionamiento actual de la empresa.

2.1.2.- Recursos técnicos disponibles actualmente

IT Solutions cuenta con la siguiente infraestructura tecnológica:

- Servidores locales para el almacenamiento de datos internos, ya que las plataformas que usan actualmente son *On-Premise*.

- Conexión a Internet de alta velocidad.

- Equipos informáticos de prestaciones suficientes.

- Licencias básicas de software que utilizan actualmente (Excel, Outlook, software gestión de inventario y de facturación).

2.1.3.- Por qué un CRM/ERP ayudaría a solucionar la problemática de la empresa

La implementación de un sistema CRM/ERP, ya sea como plataforma centralizada o diversas plataformas bien integradas y sincronizadas, podrían abordar los problemas mencionados anteriormente:

- **Centralización de la información**: Unificar los datos de clientes, ventas, inventario y finanzas en una plataforma única, mejorando la coherencia y reduciendo errores.

- **Automatización de procesos**: Eliminar la introducción manual de datos redundantes, optimizando tareas como la facturación, la gestión de inventario y el seguimiento de oportunidades de venta.

- **Mejora en la atención al cliente**: Registrar y rastrear todas las interacciones con los clientes, permitiendo ofrecer un servicio más ágil, personalizado y eficiente.

- **Análisis y reportes**: Acceso a paneles de control con métricas actualizadas para una mejor toma de decisiones.

- **Escalabilidad y adaptabilidad**: Facilitar la incorporación de nuevas funcionalidades a medida que la empresa crezca o diversifique sus operaciones.

Además de todas estas ventajas, la implementación de una solución CRM/ERP *Cloud* garantiza accesibilidad remota con un *uptime* de prácticamente 24/7/365, sin requerir de un gran desembolso inicial, ya que se optará por un modelo de suscripción y adquisición de licencias.

Además, podremos delegar en el proveedor del SaaS las tareas de actualización, mantenimiento y seguridad del sistema, lo que se traducirá directamente en una reducción de infraestructura local de la empresa, junto a su costo.

Todos estos puntos se alinean perfectamente con la estrategia de digitalización y modernización que se desea implantar.

2.2.- HERRAMIENTAS DISPONIBLES EN EL MERCADO

A continuación evaluaremos las herramientas SaaS CRM/ERP más destacadas y reconocidas en el mercado y que podrían satisfacer las necesidades de IT Solutions [13].

No se hará una separación entre herramientas CRM y ERP, porque aunque cada una tiene funciones específicas, muchas plataformas incluyen ambos tipos de herramientas o permiten personalizarlas y crear módulos a medida según las necesidades de cada empresa, lo que las hace útiles tanto para CRM como para ERP.

2.2.1.- Salesforce

Figura 2.1: Logo Salesforce

Salesforce [14] es actualmente una de las plataformas más reconocidas y utilizadas a nivel mundial en el ámbito de la gestión empresarial. Su popularidad se debe, en gran parte, a su alta flexibilidad, escalabilidad y capacidad de personalización, lo que le permite adaptarse con facilidad a empresas de distintos tamaños y sectores. Esta adaptabilidad convierte a Salesforce en una herramienta especialmente útil para organizaciones que necesitan una solución eficiente y configurable.

La plataforma permite gestionar de forma centralizada distintas áreas del negocio como las ventas, el servicio de atención al cliente, las campañas de marketing, el análisis de datos y la automatización de procesos internos. Esta integración de funcionalidades la convierte en una opción muy completa y atractiva para mejorar la eficiencia operativa de cualquier organización.

Entre sus características principales destacan:

- **Personalización avanzada**, posible tanto mediante el desarrollo con código como con herramientas de drag and drop, conocidas como no-code, low-code o “clicks”.

- **Salesforce AppExchange**, una tienda que permite instalar aplicaciones de terceros, gratuitas o con suscripción, para ampliar aún más las capacidades del sistema.

- **Gestión integral** de procesos empresariales clave, incluyendo la gestión de ventas, control de inventario y soporte técnico, todo desde una única plataforma Cloud.

- **Informes personalizados y análisis de datos**, gracias a sus Informes y Paneles (Dashboards) que permiten visualizar métricas en gráficas o tablas configurables.

- **Arquitectura basada en la nube**, lo que facilita el acceso remoto, la escalabilidad y la disponibilidad del sistema, además de permitir actualizaciones automáticas sin necesidad de intervención del usuario.

- **Amplia gama de productos** dentro del propio ecosistema Salesforce, que permiten cubrir diferentes necesidades empresariales bajo una misma licencia, lo que simplifica la gestión tecnológica y reduce los costes de integración.

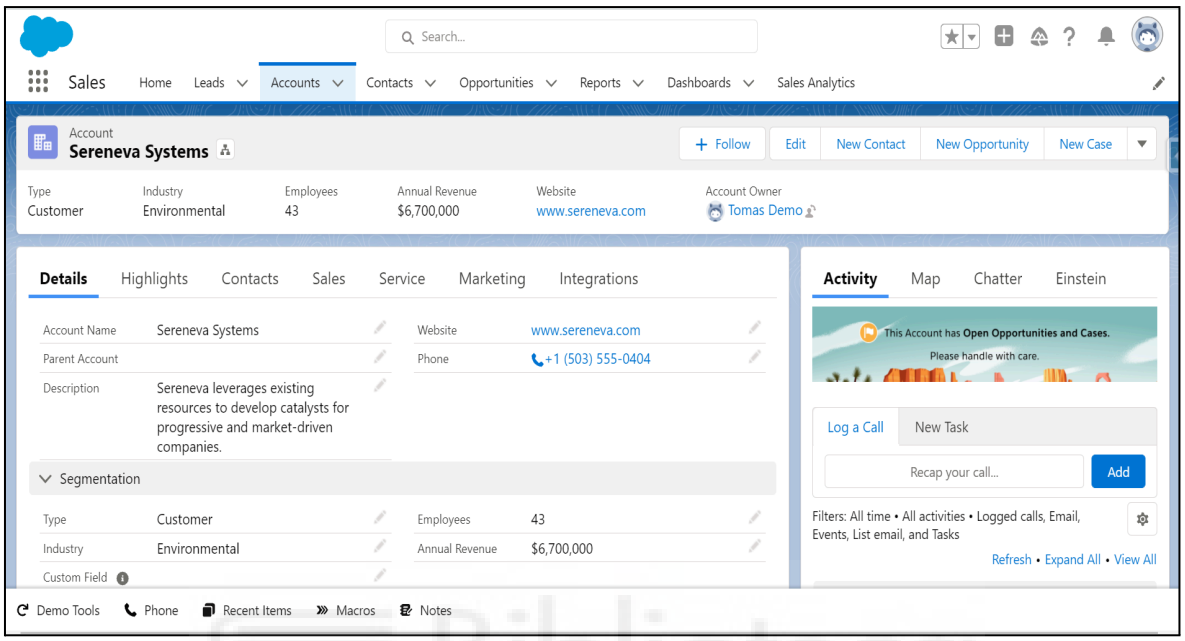

Figura 2.2: Interfaz de un registro de Salesforce

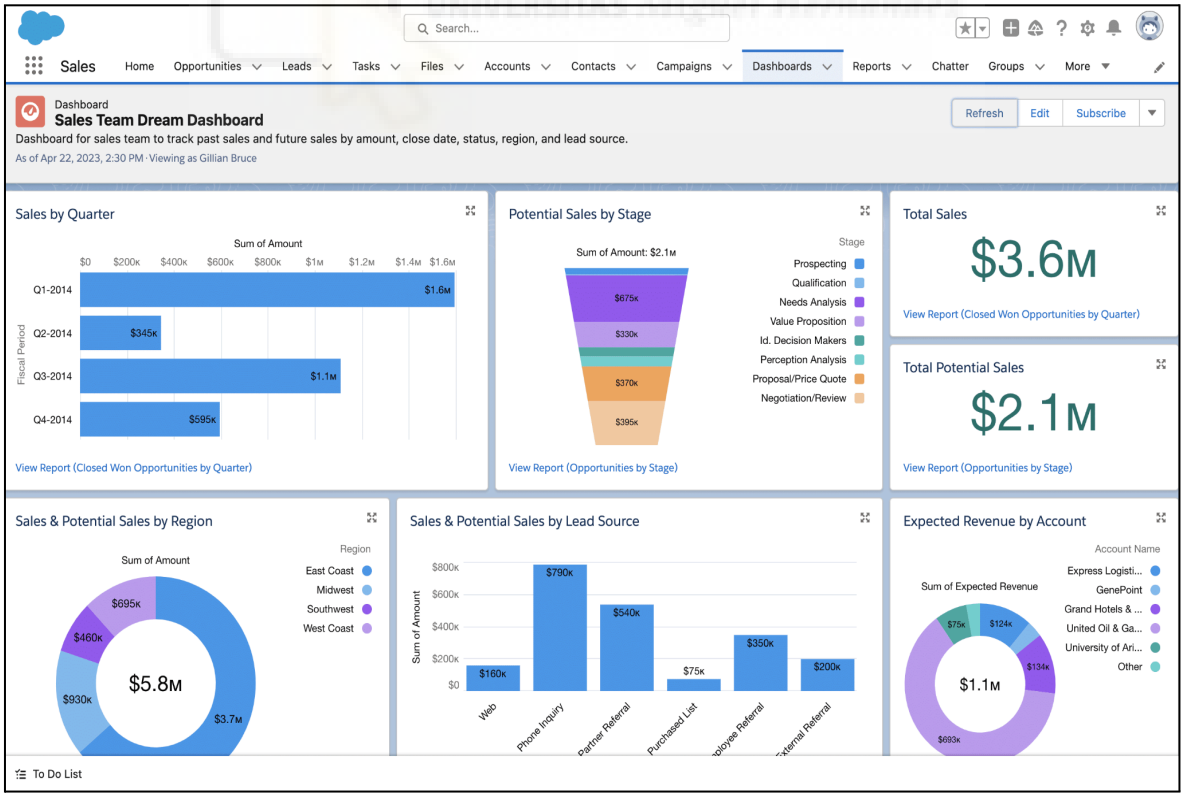

Figura 2.3: Ejemplo de Dashboard con múltiples Reportes en Salesforce

2.2.2.- Microsoft Dynamics 365

Figura 2.4: Logo Microsoft Dynamics 365

Microsoft Dynamics 365 [15][16] es una plataforma empresarial que combina funcionalidades de CRM y ERP, ofreciendo una solución integral para gestionar distintos aspectos del negocio desde un único sistema. Su principal ventaja es la integración nativa con el ecosistema Microsoft, lo que facilita su adopción en empresas que ya trabajan con herramientas como Microsoft 365, Teams, Azure o SharePoint, simplificando la conectividad y el intercambio de datos entre plataformas.

Esta solución está diseñada en torno a módulos personalizables, que permiten activar únicamente las funciones necesarias para cada organización: ventas, marketing, atención al cliente, contabilidad, gestión de inventario, entre otros. Esta modularidad aporta flexibilidad y control sobre la implementación.

Además, Dynamics 365 incorpora tecnologías como la inteligencia artificial, machine learning y análisis predictivo, que mejoran la eficiencia operativa y permiten anticiparse a las necesidades del mercado mediante datos en tiempo real.

Gracias a su infraestructura basada en la nube, garantiza acceso seguro desde cualquier dispositivo, actualizaciones automáticas y alta disponibilidad, lo que contribuye a la continuidad del negocio y a una mayor movilidad de los equipos.

Características principales:

- **Integración nativa** con Microsoft 365, Azure, Teams, SharePoint, lo que facilita la interoperabilidad con otras herramientas ampliamente adoptadas.

- **Módulos personalizables** adaptados a diferentes áreas de negocio: ventas, marketing, atención al cliente, contabilidad, finanzas o gestión de inventario.

- **Funcionalidades avanzadas de inteligencia artificial y análisis predictivo** que permiten automatizar procesos y tomar decisiones basadas en datos.

- **Acceso desde cualquier dispositivo** gracias a su infraestructura *Cloud*.

- **Interfaz familiar y coherente** con el resto de productos de Microsoft, lo que reduce la curva de aprendizaje para los usuarios.

- **Entorno de desarrollo ampliado** con herramientas como Power Apps, Power BI o Power Automate para crear soluciones a medida sin partir de cero.

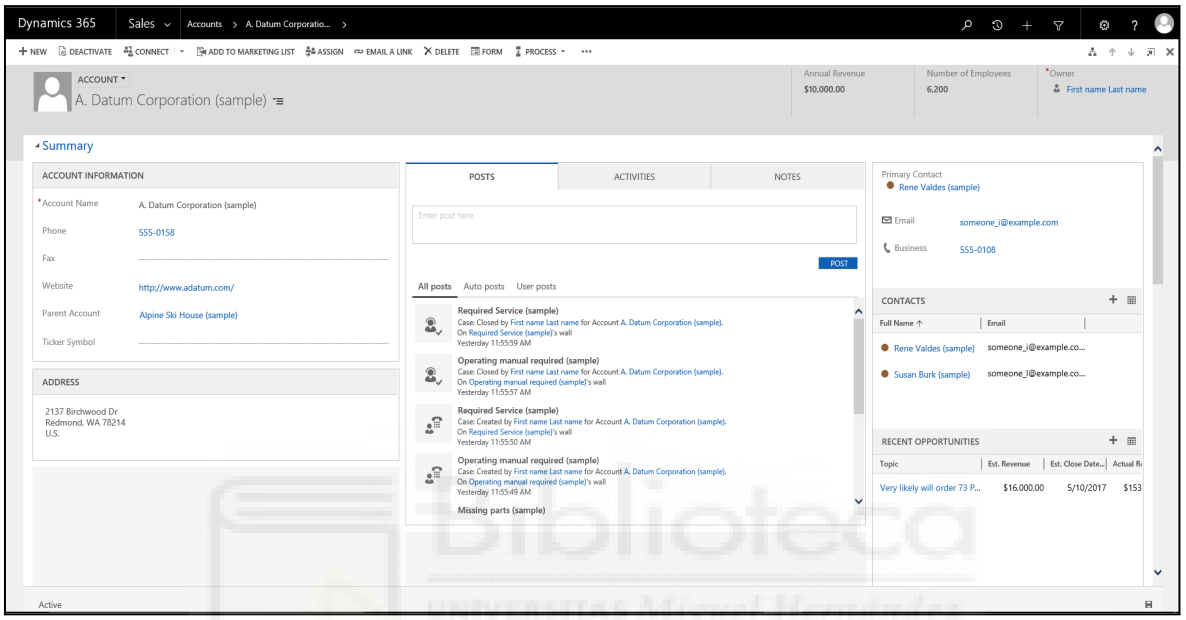

Figura 2.5: Interfaz de un registro de Microsoft Dynamics 365

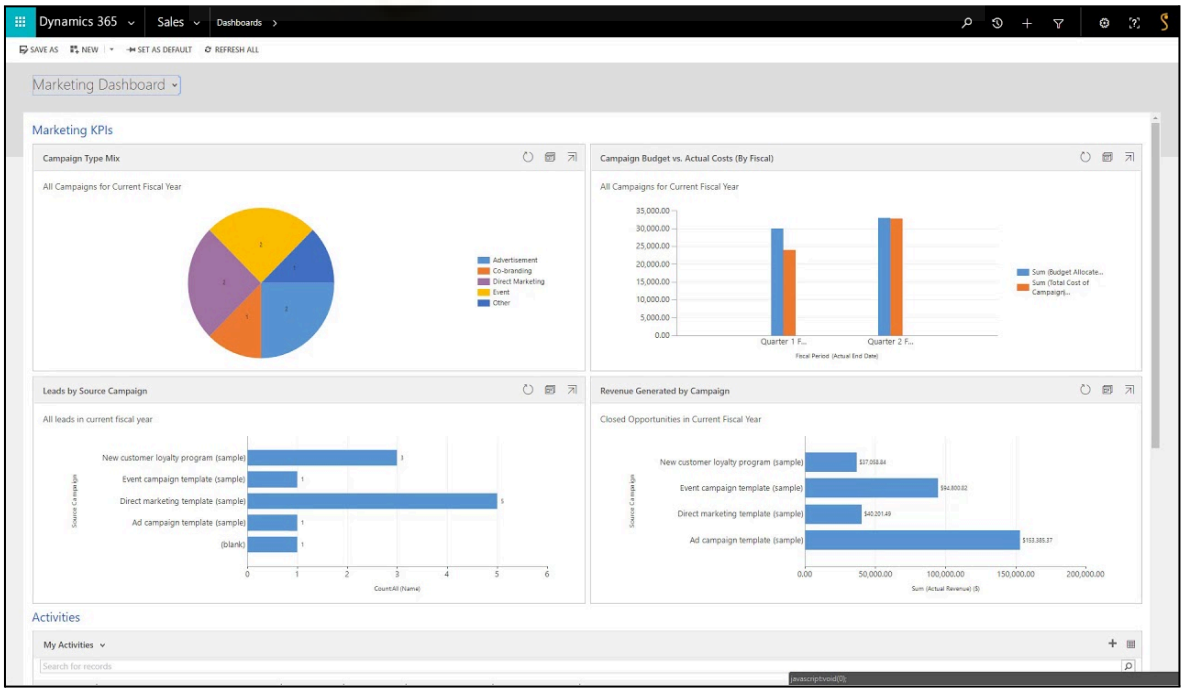

Figura 2.6: Ejemplo de Dashboard en Microsoft Dynamics 365

2.2.3.- Odoo

Figura 2.7: Logo Odoo

Odoo [17] es una plataforma de gestión empresarial de código abierto [18] que destaca por su enfoque modular y su amplia comunidad de desarrollo. Está diseñada para cubrir múltiples áreas de una empresa, desde CRM y ERP, hasta contabilidad, recursos humanos, fabricación, gestión de inventario, comercio electrónico y mucho más. Su arquitectura flexible permite a empresas de cualquier tamaño adaptar y escalar la solución en función de sus necesidades, lo que la convierte en una alternativa muy competitiva frente a plataformas propietarias más costosas.

Al estar desarrollada principalmente en Python [19], y con una interfaz web moderna y limpia, Odoo permite una rápida personalización tanto en términos funcionales como visuales. Esto lo convierte en una herramienta ideal para empresas que desean tener control total sobre su software, además de una gran comunidad que contribuye con módulos, actualizaciones y soporte.

Características principales:

- **Modularidad completa**, permitiendo añadir o quitar funcionalidades fácilmente según las necesidades del negocio.
- **Modelo de código abierto**, lo que reduce costes de licencias y favorece la personalización.
- **Precio más accesible** en comparación con plataformas propietarias como Salesforce o SAP.
- **Acceso desde navegador web y dispositivos móviles**, favoreciendo el trabajo fuera de una oficina.
- **Gran comunidad de usuarios y desarrolladores**, que proveen soporte, módulos adicionales y documentación.
- **Interfaz de usuario intuitiva**, con posibilidad de personalizar dashboards, vistas y formularios.

- **Backend en Python y base de datos PostgreSQL**, tecnologías ampliamente utilizadas y mantenidas.

- **Versión gratuita (Community) y versión empresarial (Enterprise)**, adaptándose a distintos niveles de necesidad y presupuesto.

- **Actualizaciones periódicas** y evolución continua del producto gracias a su fuerte comunidad open source y a Odoo S.A.

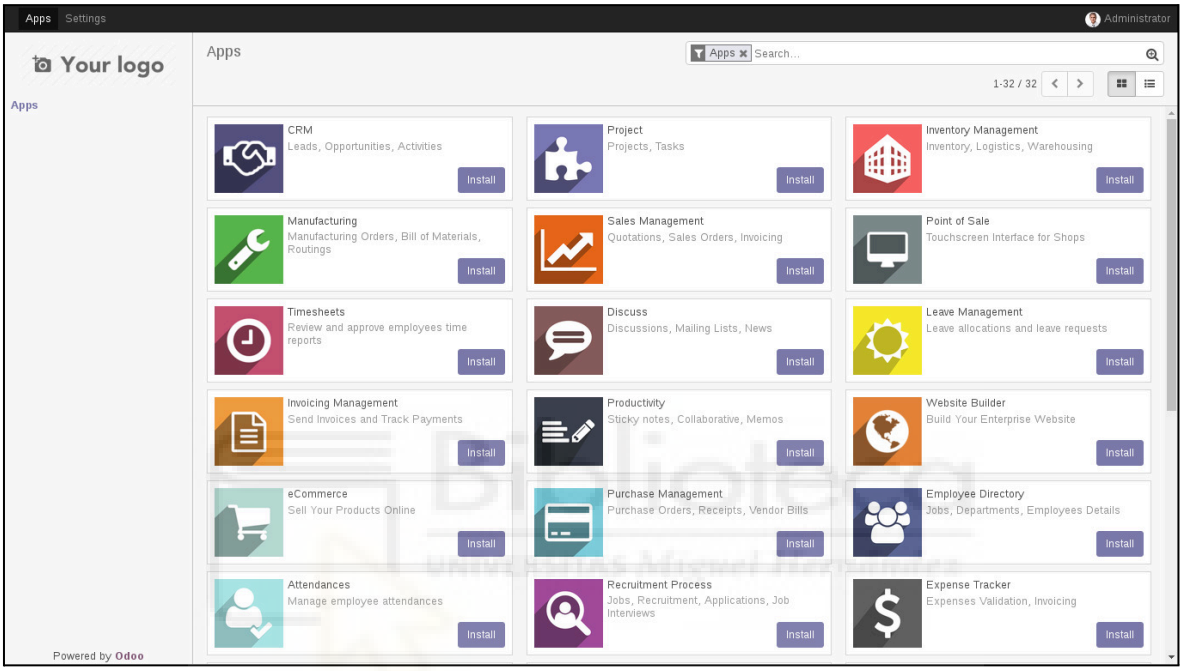

Figura 2.8: Menú de aplicaciones de Odoo

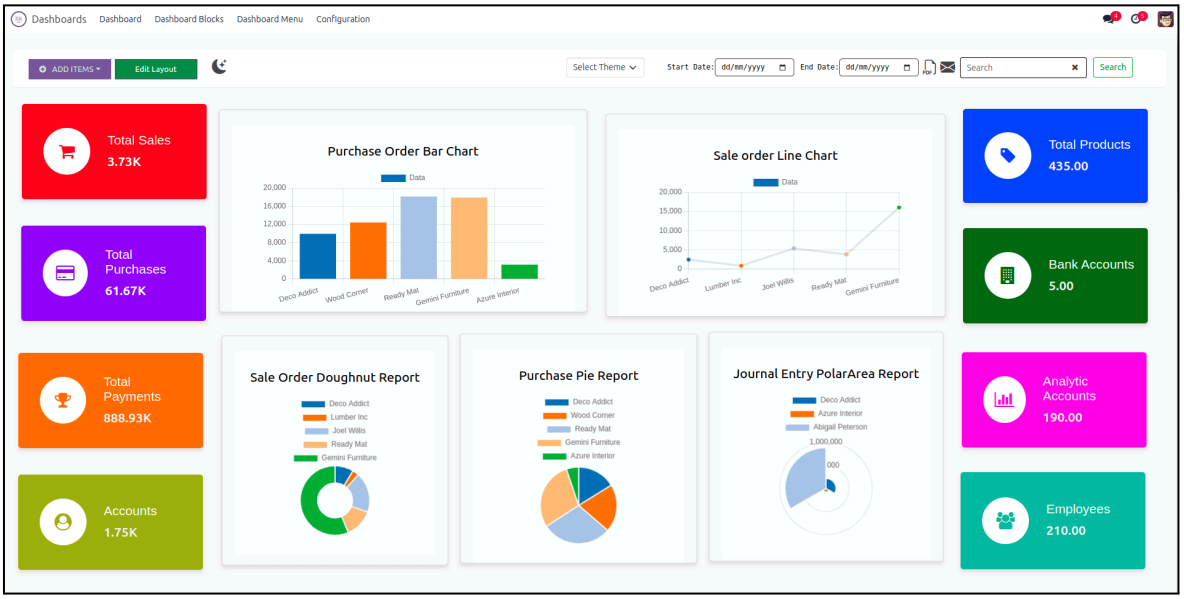

Figura 2.9: Ejemplo de Dashboard en Odoo

2.2.4.- SAP Business One y SAP S/4HANA

Figura 2.10: Logo SAP

SAP es una de las compañías líderes en software de gestión empresarial, ofreciendo una amplia gama de soluciones ERP adaptadas a las necesidades de empresas de todos los tamaños. Entre sus productos más destacados se encuentran SAP Business One, orientado a pequeñas y medianas empresas, y SAP S/4HANA Cloud, una solución de nueva generación diseñada para organizaciones más grandes o en rápido crecimiento, con procesos más exigentes y complejos.

SAP Business One [20] proporciona una herramienta completa e integrada para la gestión de áreas clave como finanzas, compras, ventas, CRM, inventario y operaciones. Su enfoque es ofrecer una solución sencilla de implementar, con una curva de aprendizaje más asequible, ideal para compañías que buscan digitalizar y automatizar su operativa sin la complejidad de sistemas empresariales más grandes.

Por otro lado, SAP S/4HANA [21] está basado en la potente tecnología in-memory de HANA, lo que permite procesar grandes volúmenes de datos en tiempo real. Esta solución está orientada a empresas que requieren una infraestructura escalable, con capacidad analítica avanzada y procesos optimizados de principio a fin, todo en entorno en la nube.

Características principales:

- **Automatización de procesos empresariales** en áreas clave como finanzas, logística, compras y ventas.
- **Integración con herramientas analíticas avanzadas**, como SAP Analytics Cloud.
- **Interfaz moderna y unificada** (SAP Fiori), con experiencia de usuario optimizada.
- **Capacidad de personalización y extensión** mediante SAP Business Technology Platform.
- **Acceso multi-dispositivo**, permitiendo a los usuarios trabajar desde cualquier lugar.

- **Amplia red de soporte y partners**, que facilita la implantación y mantenimiento del sistema.

- **Migración progresiva desde sistemas anteriores de SAP**, como SAP ECC, hacia S/4HANA.

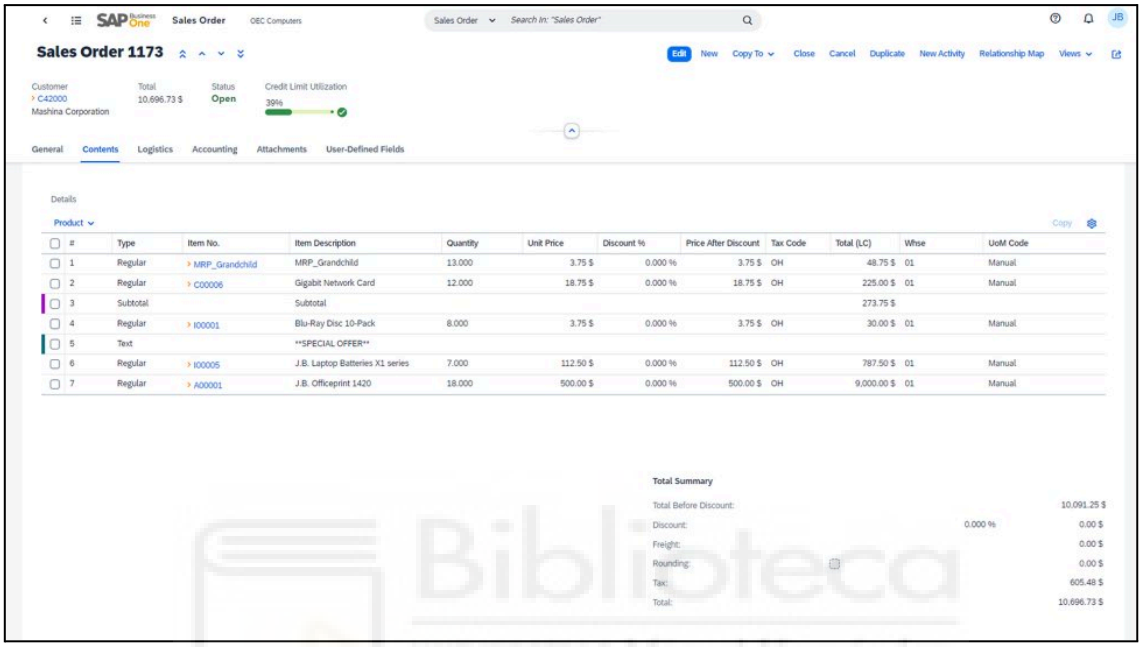

Figura 2.11: Interfaz de un registro de SAP Business One

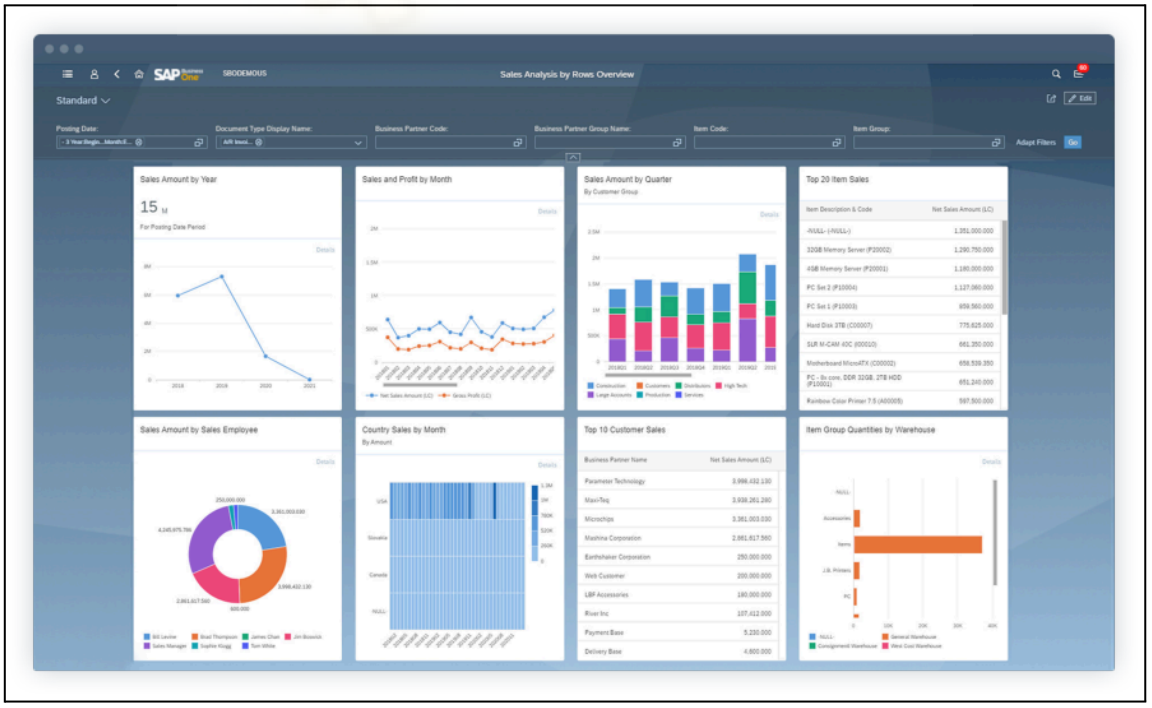

Figura 2.12: Ejemplo de Dashboard en SAP Business One

2.2.5.- Zoho

Figura 2.13: Logo Zoho

Zoho [22] es una suite de aplicaciones empresariales en la nube que ofrece soluciones integradas para la gestión de diversos procesos como ventas, marketing, atención al cliente, finanzas, recursos humanos, y más. Su propuesta destaca por ser altamente accesible para pequeñas y medianas empresas, combinando una buena relación calidad-precio con un conjunto de herramientas funcionales y personalizables.

Una de las principales ventajas de Zoho es su enfoque todo-en-uno, permitiendo a las empresas centralizar sus operaciones en una única plataforma sin necesidad de adquirir múltiples productos o realizar integraciones complejas. Además, gracias a su arquitectura cloud y su enfoque low-code [23], facilita una rápida implantación y adaptación a los procesos específicos de cada empresa.

Zoho CRM es uno de sus productos más populares, proporcionando funcionalidades completas para la gestión de relaciones con clientes, automatización de ventas y seguimiento de oportunidades. Todo ello dentro de una interfaz sencilla y fácil de usar.

Características principales:

- **Amplia suite integrada** con más de 45 aplicaciones empresariales: CRM, Books, Projects, Recruit, People, etc.
- **Modelo SaaS con precios asequibles**, ideal para pequeñas empresas o negocios que quieran comenzar una digitalización sin invertir demasiado de primeras.
- **Automatización de procesos** mediante flujos de trabajo personalizables.
- **Interfaz intuitiva con posibilidad de personalización** mediante Zoho Creator (plataforma low-code propietaria).
- **Zoho Analytics** para informes y dashboards avanzados.

- **Integración nativa entre las apps del ecosistema Zoho** y con herramientas externas (Google Workspace, Microsoft 365, etc.).

- **Soporte multilingüe y multiempresa.**

- **Énfasis en la privacidad** y el cumplimiento de normativas como el RGPD.

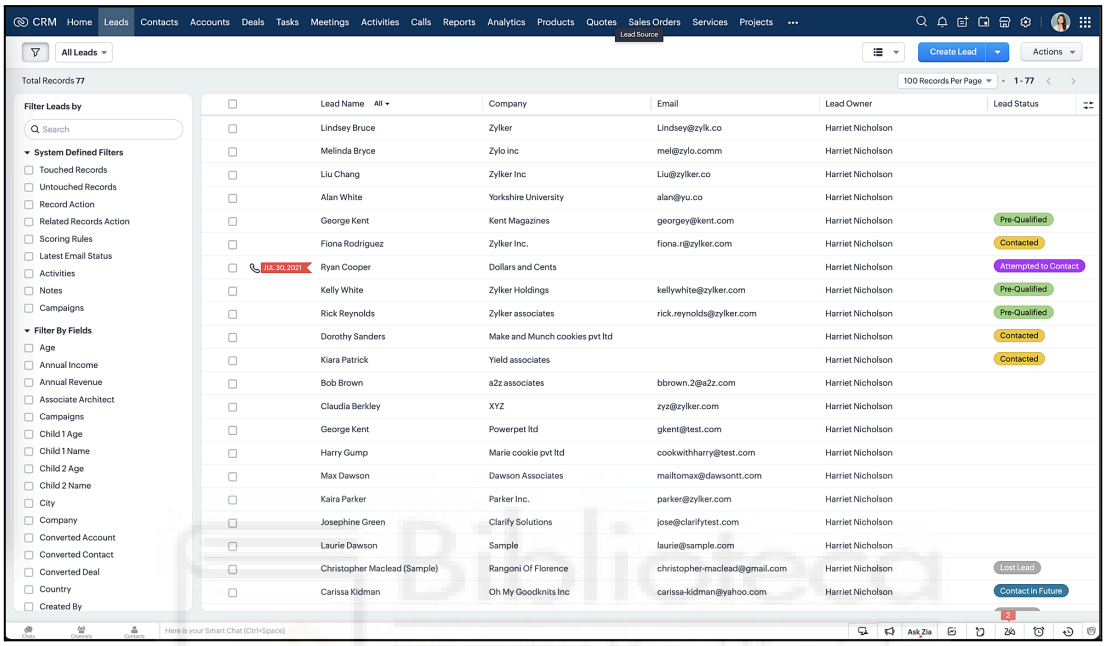

Figura 2.14: Lista de registros en Zoho

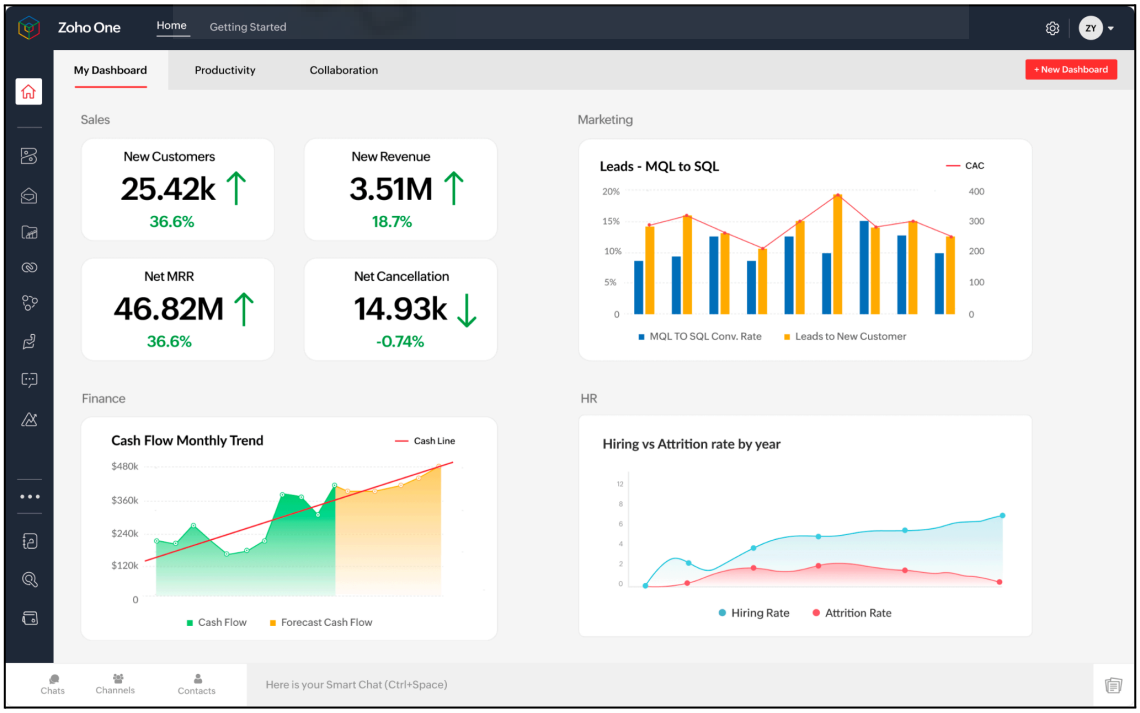

Figura 2.15: Ejemplo de Dashboard en Zoho

2.2.6.- HubSpot

Figura 2.16: Logo HubSpot

HubSpot es una plataforma de gestión empresarial enfocada principalmente en la gestión de relaciones con los clientes (CRM) [24], marketing digital, ventas y atención al cliente. Es especialmente reconocida por su enfoque en áreas concretas y por ofrecer una interfaz amigable, intuitiva y accesible, lo que la convierte en una solución ideal para pequeñas y medianas empresas que buscan profesionalizar su gestión comercial sin una elevada curva de aprendizaje.

Una de las características que diferencia a HubSpot es su modelo freemium: el acceso básico a su CRM es gratuito, lo que permite a las empresas comenzar a utilizarlo sin inversión inicial. A medida que las necesidades crecen, se pueden añadir módulos de pago que amplían las capacidades del sistema en áreas como automatización de marketing, gestión avanzada de ventas, analítica o soporte técnico.

Además, HubSpot pone un fuerte énfasis en la integración de herramientas de marketing, permitiendo la creación de campañas automatizadas, seguimiento de leads, gestión de contenidos y análisis de resultados, todo desde una única plataforma.

Características principales:

- **CRM gratuito** con funcionalidades básicas para ventas, marketing y atención al cliente.
- **Automatización de marketing** con workflows personalizables (en planes de pago).
- **Gestión de contactos y oportunidades**, con historiales de interacción sobre los registros.
- **Plantillas de emails, landing pages y formularios.**
- **Análisis y reporting de campañas** con paneles visuales y datos en tiempo real.

- **Integración con múltiples herramientas externas** como Gmail, Outlook, Shopify, Slack, entre muchas otras.

- **Marketplace de extensiones** para ampliar funcionalidades.

- **Formación continua y certificaciones** a través de HubSpot Academy.

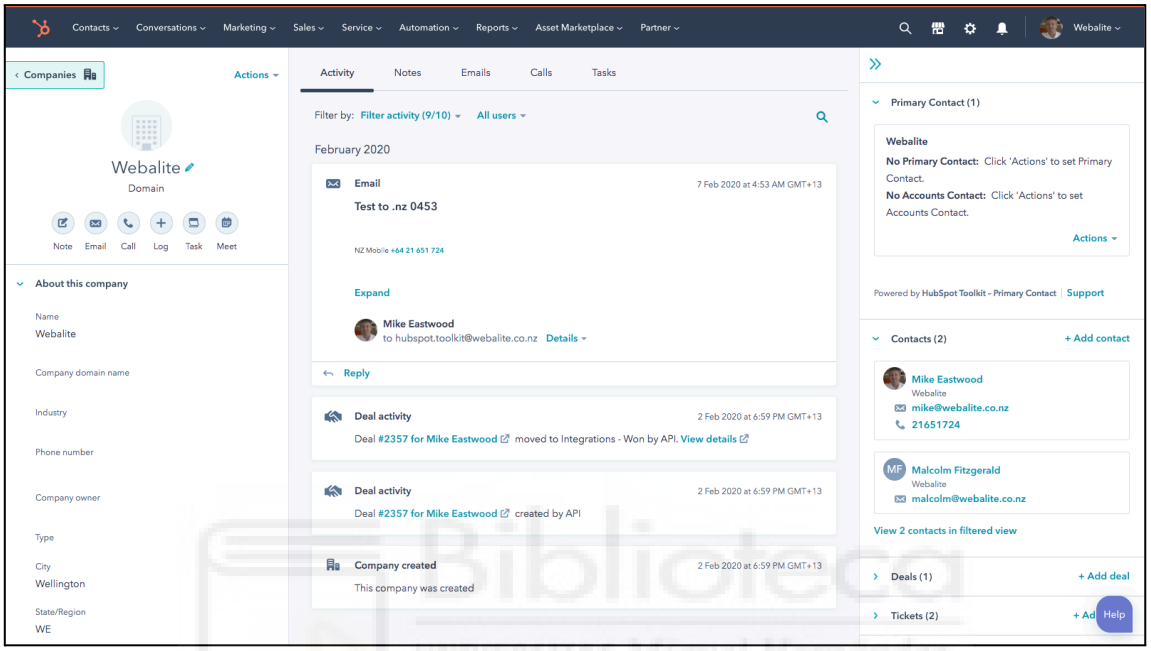

Figura 2.17: Interfaz de un registro de HubSpot

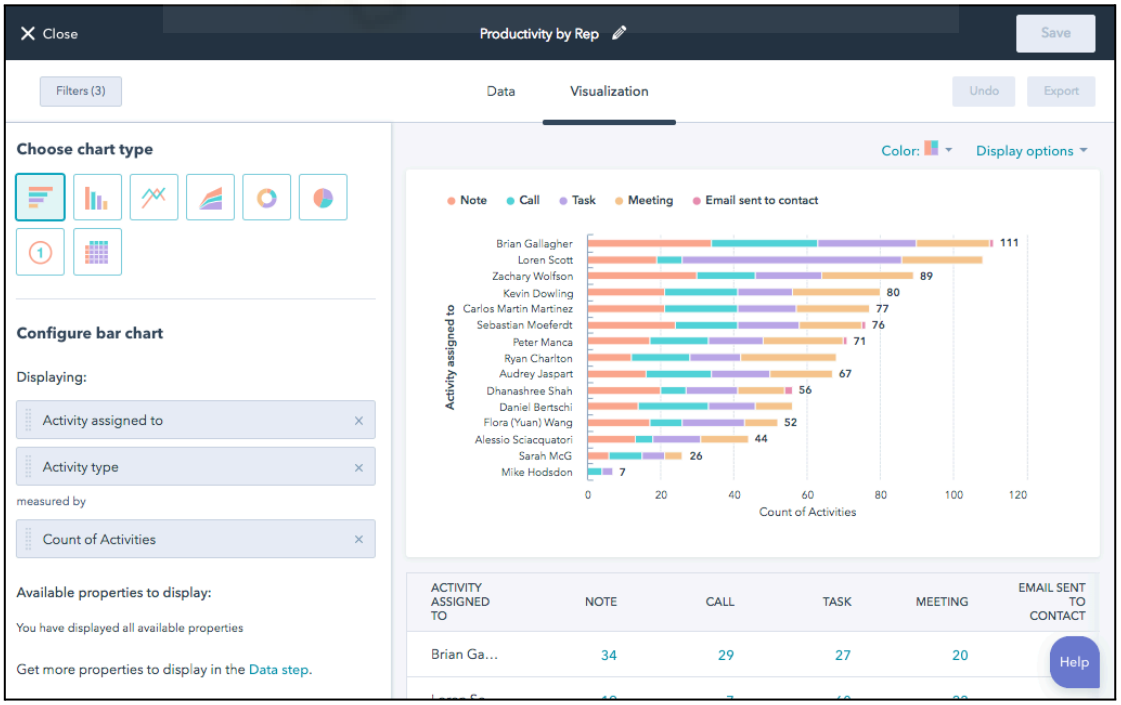

Figura 2.18: Ejemplo de analíticas en HubSpot

2.3.- VALORACIÓN

La tabla 2.1 muestra de forma resumida las principales ventajas y desventajas que se aprecian en las herramientas descritas en el apartado anterior.

Tabla 2.1: Ventajas e Inconvenientes de distintas plataformas CRM/ERP

Plataforma	Ventajas	Inconvenientes	Observaciones
Salesforce	Alta personalización, gran ecosistema, AppExchange, automatización potente	Curva de aprendizaje elevada, coste alto	Ideal para empresas con necesidades complejas y equipos técnicos especializados
Microsoft Dynamics 365	Integración nativa con Microsoft, IA avanzada, flexible por módulos	Coste elevado, puede requerir personalización compleja	Muy recomendable si ya se usa el ecosistema Microsoft
Odoo	Código abierto, modular, económico, personalizable con Python	Requiere conocimientos técnicos para personalización	Buena opción para startups o PyMEs con equipo técnico interno
SAP Business One o S/4HANA	Procesos en tiempo real, muy escalable, solución robusta	Implementación compleja, licencias y mantenimiento costosos	Potente en entornos industriales y logísticos
Zoho	Precio competitivo, interfaz intuitiva, buen soporte de apps propias	Algunas limitaciones en funcionalidades avanzadas	Ideal para PYMEs que buscan agilidad y simplicidad
HubSpot	CRM gratuito, fácil de usar, muy orientado a marketing	Funcionalidades limitadas en versión gratuita, menos flexible	Excelente para equipos comerciales y de marketing en fase de crecimiento

Tras haber evaluado diversas soluciones CRM y ERP, se ha decidido optar por **Salesforce** debido a su versatilidad, robustez y capacidad para adaptarse de manera óptima a las necesidades de IT Solutions. A continuación, se detallan los beneficios clave que proporcionará una implementación de Salesforce:

1. **Migración a un entorno Cloud.** Esta migración permite a IT Solutions reducir los costes de infraestructura local, ya que elimina la necesidad de servidores físicos y su mantenimiento. Además, esta transición garantiza un acceso remoto continuo y seguro a la plataforma desde cualquier dispositivo. Al estar alojada en la nube, Salesforce se actualiza automáticamente, asegurando que la empresa siempre esté utilizando la versión más reciente sin tener que preocuparse por el mantenimiento o la seguridad del sistema.

2. **Centralización de programas utilizados.** Salesforce permite centralizar todos los procesos clave de IT Solutions en una única plataforma, integrando gestión de clientes, inventario, ventas y facturación. Esto reduce la dependencia de múltiples programas desconectados y facilita un flujo de trabajo más ágil y eficiente.

3. **Personalizaciones escalables y robustas mediante low-code/no-code y código.** Salesforce ofrece amplias posibilidades de personalización, desde configuraciones low-code/no-code accesibles para usuarios sin perfil técnico, hasta desarrollos complejos mediante código usando tecnologías web estándar y backend propietario para cubrir necesidades más específicas y casos de uso complejos

4. **Recursos de aprendizaje y documentación del producto.** Salesforce pone a disposición de los usuarios la plataforma Trailhead, una herramienta gratuita e interactiva de gamificación para aprender a usar y desarrollar sobre la plataforma. Junto con una documentación oficial extensa y actualizada, esto facilita la capacitación continua del equipo y acelera la adopción del sistema en IT Solutions.

5. **Reconocimiento en el mercado.** Salesforce es un claro líder del mercado CRM, con una trayectoria consolidada y un fuerte enfoque en innovación (IA, automatización de procesos, análisis predictivo…). Esto aporta confianza a la hora de elegir una solución a largo plazo que será mantenida y evolucionada continuamente.

Capítulo 3 Hipótesis de trabajo

Este capítulo recoge todos los elementos técnicos, metodológicos y normativos que se tomarán en cuenta durante el desarrollo del presente Trabajo de Fin de Grado. En él se detallan las tecnologías empleadas, tanto a nivel de lenguajes de programación como de frameworks, arquitecturas y herramientas de desarrollo. Asimismo, se describen las herramientas auxiliares necesarias para la implementación, el control de versiones, la integración continua y la gestión del ciclo de vida del software.

Dado que Salesforce es una plataforma compleja y especializada, se abordarán con mayor profundidad aquellos componentes menos conocidos o que son específicos del entorno Salesforce.

Finalmente, se explicarán brevemente aquellas tecnologías de uso más común o ampliamente adoptadas, así como una serie de estándares de desarrollo y buenas prácticas en el entorno de Salesforce.

3.1.- TECNOLOGÍAS Y ARQUITECTURAS

En este apartado se hablará de las tecnologías mínimas necesarias para poder empezar un proyecto de Salesforce, así como una explicación de la arquitectura multiusuario o multitenant.

3.1.1.- Salesforce Platform Cloud

Salesforce Platform es una plataforma de desarrollo de software empresarial basada en la nube (SaaS/PaaS) que proporciona un entorno de ejecución y desarrollo pensado para la construcción de aplicaciones de negocio.

A diferencia de un framework web de propósito general, Salesforce ofrece de forma nativa un modelo de datos relacional, herramientas low-code/no-code de automatización de procesos, un framework de componentes frontend y un runtime para código backend en su lenguaje propietario, todo ello sobre una infraestructura cloud compartida que el proveedor opera y mantiene.

Para el presente proyecto, la elección de Salesforce como plataforma de implementación responde a varios factores técnicos concretos:

- Modelo de datos extensible sin gestión de esquema: los objetos estándar cubren directamente la mayoría de las entidades del dominio del problema (clientes, oportunidades, pedidos, casos de soporte), reduciendo la cantidad de código y personalizaciones necesarias frente a un desarrollo desde cero

- Entorno de ejecución gestionado: el despliegue, la disponibilidad, las actualizaciones, el coste y la seguridad de la infraestructura son responsabilidad del proveedor, lo que permite preocuparnos únicamente del desarrollo

- Integración nativa de herramientas: la plataforma integra en un único entorno el modelado de datos, la automatización declarativa (Flow Builder), el desarrollo en código (Apex, LWC), la gestión de autenticación de usuario y sus permisos y el acceso a APIs externas, ahorrándonos el esfuerzo de tener que integrar múltiples tecnologías

En los siguientes apartados se describen los componentes técnicos de la plataforma que son relevantes para la implementación realizada.

3.1.2.- Arquitectura multitenant o multiusuario

Una de las características definitorias de Salesforce es su arquitectura multitenant [25] [26]. A diferencia de una arquitectura tradicional en la que cada cliente dispone de su propia instancia de software y su propia base de datos, en un modelo multitenant todos los clientes (denominados organizaciones) comparten la misma infraestructura física y las mismas instancias del software, aunque sus datos permanecen completamente aislados y protegidos entre sí. Sus ventajas frente al modelo tradicional son:

- **Actualizaciones automáticas y transparentes**: Salesforce publica tres versiones al año (Spring, Summer y Winter). Al compartir la infraestructura, todos los clientes reciben las actualizaciones de forma simultánea y sin interrupciones en el servicio, sin coste adicional ni intervención técnica por parte del cliente.

- **Escalabilidad**: La plataforma puede crecer sin que el usuario tenga que preocuparse por gestionar servidores adicionales ni ampliar la capacidad de almacenamiento de forma manual.

- **Eficiencia de costes**: Al compartir recursos entre múltiples clientes, el coste por usuario se reduce significativamente en comparación con soluciones instaladas en los servidores propios del cliente (On-Premise).

- **Disponibilidad y fiabilidad**: Salesforce garantiza un elevado nivel de uptime, reflejado en su portal público de estado llamado Salesforce Trust [27], donde se puede consultar en tiempo real el estado de todos sus servicios a nivel mundial, asi como un calendario con los mantenimientos planeados.

Internamente, Salesforce gestiona los denominados "**Governor Limits**" [28], que son restricciones impuestas a las operaciones que puede realizar un cliente dentro de la plataforma compartida, con el objetivo de garantizar que ninguna organización pueda monopolizar los recursos del sistema en detrimento de otros.

Estos límites afectan, entre otros aspectos, al número de registros que se pueden procesar en una sola transacción, al número de peticiones a APIs externas permitidas por día, o al almacenamiento de datos disponible. Podemos consultar en detalle los límites en la documentación, de los que se pueden destacar:

- Número máximo de queries SOQL en una transacción: 100
- Número máximo de registros obtenidos por queries SOQL: 50.000
- Número máximo de operaciones DML por transacción: 150
- Tiempo máximo de CPU por transacción: 10.000 ms (10 s)

Además, existen otro tipo de límites (aunque siempre se puede pagar más para aumentarlos), tales como el número de campos que puede tener una tabla, o número de caracteres de nuestras clases Apex (lenguaje backend propietario de la plataforma)

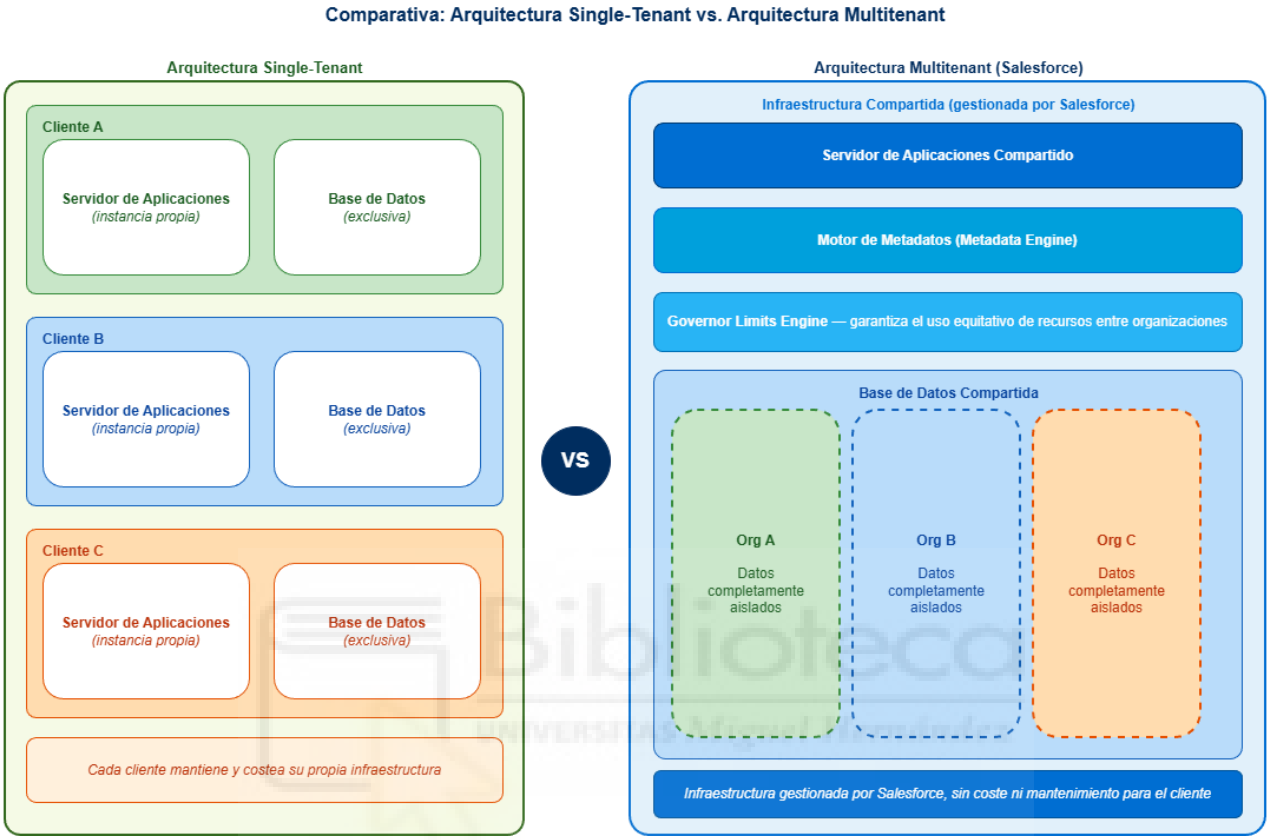

Figura 3.1: Comparativa entre arquitectura Single-Tenant y Multitenant de Salesforce

3.1.3.- Ediciones y tipos de entorno

Salesforce organiza su oferta comercial en diferentes ediciones principales (Essentials, Professional, Enterprise, Unlimited y Developer), cada una con distintas funcionalidades disponibles, límites de almacenamiento y número máximo de usuarios [29].

Para el presente proyecto se ha utilizado una Developer Edition, que es gratuita y dispone de la mayoría de las funcionalidades del plan Enterprise, siendo el entorno de referencia para el desarrollo y la exploración de funcionalidades sin coste económico. Además de las ediciones, Salesforce distingue entre diferentes tipos de entornos:

- **Production org**: La organización de producción, donde reside la actividad real de los usuarios y los datos de los clientes.

- **Sandbox**: Entornos de prueba que son copias (parciales o completas) de la organización de producción, utilizados para desarrollar y probar nuevas funcionalidades antes de desplegarlas a producción.

- **Scratch org**: Entornos de muy corta duración (1 a 30 días) y configurables desde código, utilizados en pipelines de integración y despliegue continuos (CI/CD) basados en Salesforce DX.

- **Developer Edition**: Organización independiente y gratuita, no vinculada a ninguna organización de producción, diseñada para la exploración y el aprendizaje.

3.1.3.1.- Por qué Developer Edition y no Scratch Org

Para el presente proyecto se ha optado por una Developer Edition como entorno de trabajo principal, en lugar de una Scratch Org. Aunque las Scratch Orgs son el entorno recomendado para el desarrollo profesional en equipo dentro de un pipeline de CI/CD, ya que disponen de source tracking nativo y están diseñadas para ser creadas y destruidas frecuentemente y de forma automatizada [30], presentan una limitación relevante para este contexto: su duración máxima es de 30 días, lo que resulta inadecuado para un proyecto académico desarrollado a lo largo de varios meses que requiere un entorno persistente y accesible en todo momento.

La Developer Edition, por el contrario, es permanente, gratuita y proporciona acceso a la gran mayoría de las funcionalidades del plan Enterprise, lo que la convierte en el entorno más adecuado para el desarrollo y la demostración de la solución sin restricciones temporales ni coste económico.

3.1.4.- Modelo de datos: Objetos, Campos y Relaciones

El modelo de datos de Salesforce se basa en el concepto de Objetos [31], que son análogos a las tablas de una base de datos relacional. Cada Objeto está compuesto por Campos (equivalentes a las columnas de una tabla) y cada registro es una instancia de dicho Objeto (equivalente a una fila).

Salesforce distingue dos tipos principales de Objetos:

- **Objetos estándar**: Predefinidos por Salesforce, cubren las necesidades más comunes de cualquier negocio. Entre los más relevantes para este proyecto se encuentran: Account (empresas o cuentas), Contact (personas de contacto), Lead

(clientes potenciales), Opportunity (oportunidades de venta), Case (casos de soporte), Product2 (productos del catálogo), Quote (presupuestos) y Order (pedidos).

- **Objetos personalizados (Custom Objects)**: Definidos por el administrador o desarrollador para cubrir necesidades específicas del negocio que no están contempladas en los objetos estándar. Su nombre técnico siempre termina en el sufijo **__c**. Si por ejemplo necesitáramos contemplar en nuestro modelo a alumnos de una universidad, crearemos un objeto custom llamado Alumnos, al que automáticamente se le añadiría el sufijo __c, por lo que el nombre de nuestra tabla sería: Alumnos__c.

Los campos también pueden ser estándar o personalizados, y admiten múltiples tipos de datos: texto, número, fecha, lista de selección (picklist), casilla de verificación (checkbox), moneda, fórmula, URL, entre otros.

Las Relaciones entre objetos [32] permiten vincular registros de distintos tipos. Los dos tipos de relaciones más habituales son:

- **Lookup Relationship**: Relación flexible de tipo 1:N en la que el registro hijo puede existir independientemente del padre. Es el equivalente a una clave foránea en una base de datos relacional.

- **Master-Detail Relationship**: Relación más estricta de tipo 1:N en la que el registro hijo no puede existir sin el padre. Además, habilita la funcionalidad de campos de tipo agregado (Roll-up Summary Fields), que permiten agregar valores numéricos de los registros hijos en el registro padre mediante operaciones de suma, recuento, mínimo o máximo.

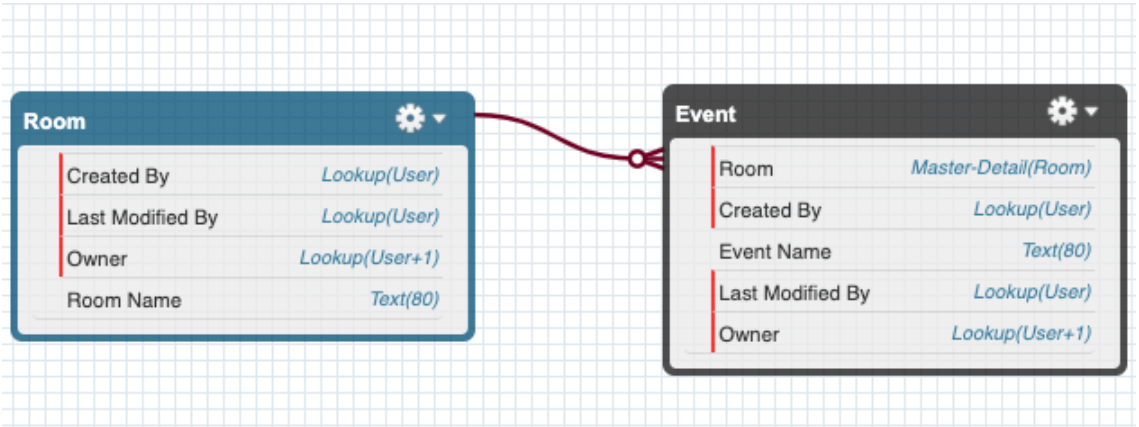

Figura 3.2: Diagrama de una relación Master-Detail

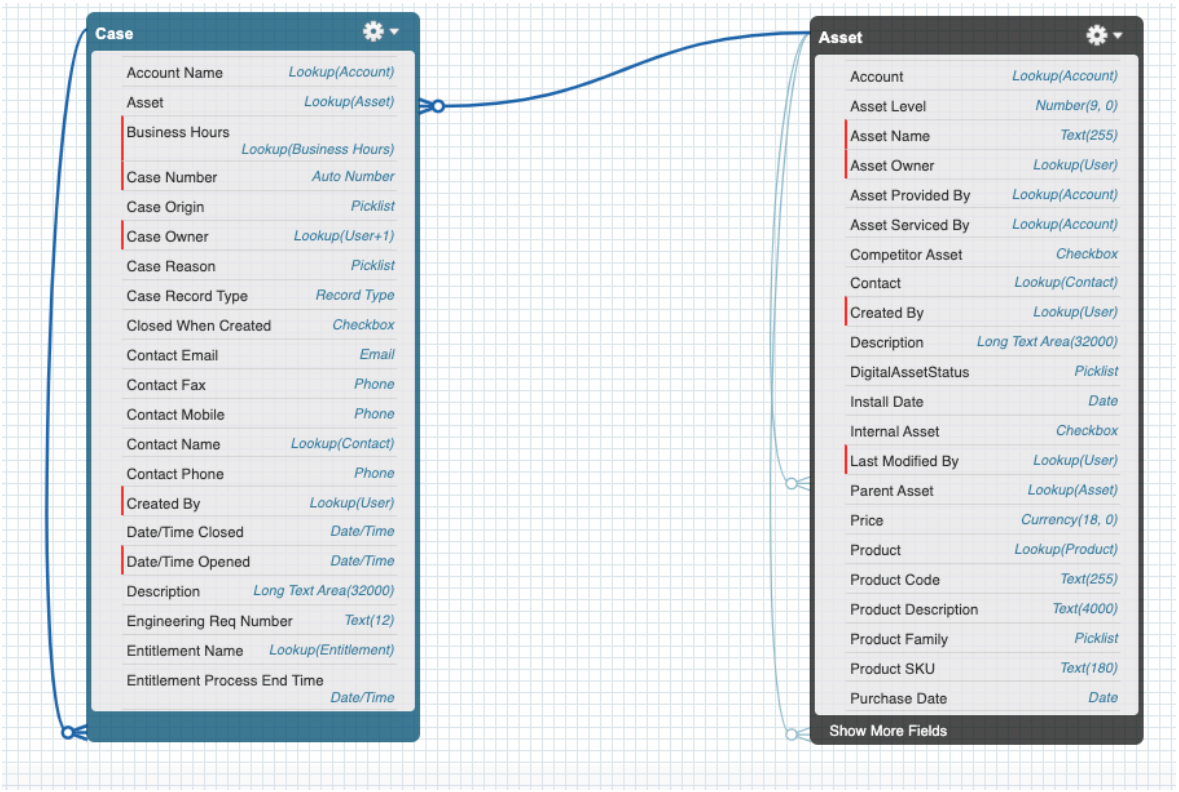

Figura 3.3: Diagrama de una relación Lookup

3.1.5.- Seguridad y control de acceso

Salesforce implementa un modelo de seguridad en capas [33] que ofrece un control granular sobre quién puede ver, crear, modificar o eliminar cada tipo de dato, dando lugar a los siguientes tipos de permisos:

- **Perfiles (Profiles)**: Definen los permisos base de un usuario: a qué objetos puede acceder, qué operaciones puede realizar (crear, leer, editar, borrar) sobre cada objeto y qué funcionalidades de la aplicación tiene disponibles. Cada usuario debe tener exactamente un Perfil asignado.

- **Conjuntos de permisos (Permission Sets)**: Permiten otorgar permisos adicionales a usuarios específicos sin necesidad de modificar su Perfil base. Son más flexibles y reutilizables que los Perfiles para casos de uso concretos. Un usuario puede tener tantos Permission Sets asignados como necesite, y puede haber overlap de permisos entre ellos sin que resulte en ningún conflicto.

- **Seguridad a nivel de objeto (Object-Level Security)**: Controla los permisos CRUD de un usuario para cierto objeto.

- **Seguridad a nivel de campo (Field-Level Security)**: Controla si un usuario puede ver o editar campos específicos dentro de un objeto, independientemente de que tenga acceso al objeto.

- **Seguridad a nivel de registro (Record-Level Security)**: Controla qué registros concretos puede ver o editar un usuario, gestionada a través de los valores predeterminados de toda la organización (Organization-Wide Defaults u OWD), jerarquías de roles y Sharing Rules.

Dentro de los Profiles y Permissions Sets, existen dos permisos que sobreescriben la Record-Level Security, estos son: **View All Data**, permiso que te habilita ver todos los registros (únicamente lectura), independientemente de su configuración de visibilidad y **Modify All Data**, que permite cualquier operación CRUD.

Los diferentes niveles de acceso se pueden representar en forma de pirámide. Un nivel concreto de la pirámide no puede restringir el acceso que proporcionan niveles anteriores. En otras palabras, cada nivel "abre" nuevos permisos, no quita.

También existe la posibilidad de configurar la visibilidad de registros por territorio. Muy útil en casuísticas donde una empresa opera en múltiples regiones y se necesita que cada responsable de región tenga visibilidad únicamente de los registros de su región.

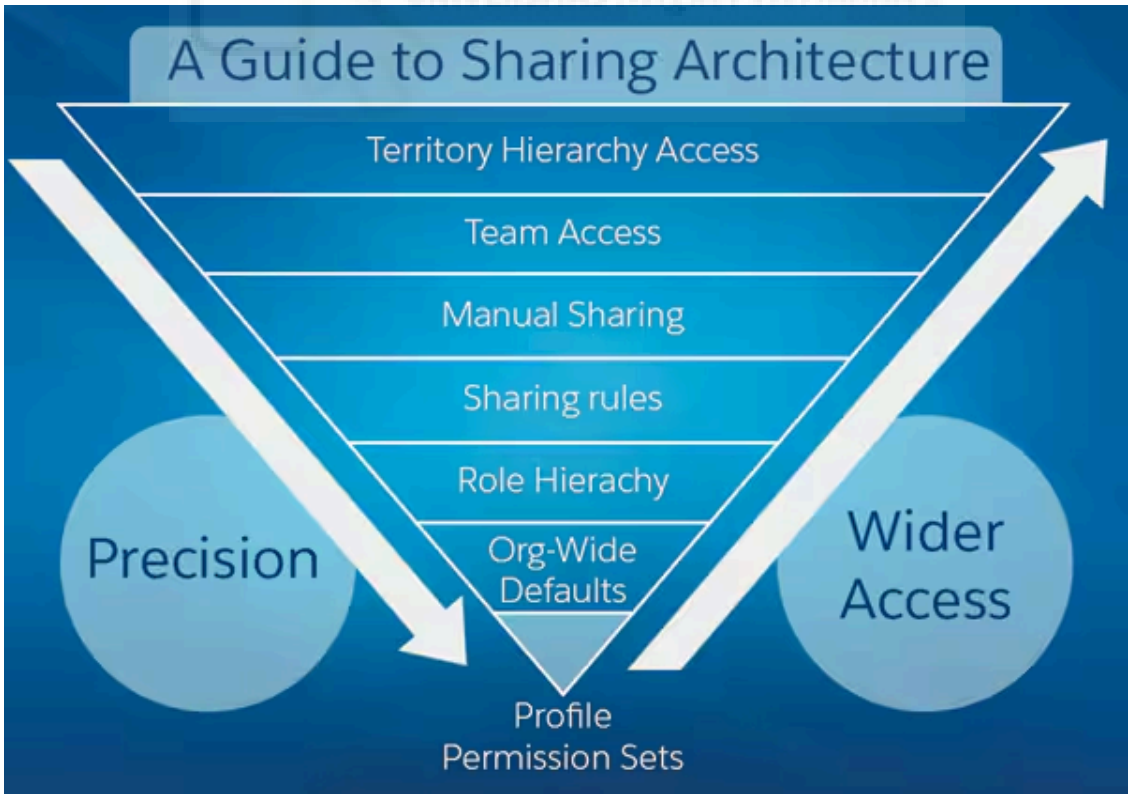

Figura 3.4: Componentes de la arquitectura de visibilidad y permisos de registros

3.2.- PRODUCTOS DE SALESFORCE UTILIZADOS

Para dar respuesta a las necesidades identificadas de IT Solutions, se han utilizado tres productos principales del ecosistema Salesforce: Sales Cloud, Service Cloud y Experience Cloud. A continuación se describe cada uno de ellos.

3.2.1.- Sales Cloud (renombrado a Agentforce Sales)

Sales Cloud [34] es el producto más conocido y utilizado de Salesforce. Está diseñado para optimizar los procesos de ventas de una organización, desde la captación de clientes potenciales hasta el cierre de un acuerdo comercial y la generación del pedido correspondiente. Sus funcionalidades principales incluyen:

- **Gestión de Cuentas y Contactos**: Permite centralizar toda la información relacionada con los clientes (empresas y personas), incluyendo el historial de interacciones, documentos adjuntos y actividades registradas.

- **Gestión de Leads**: Los Leads representan empresas o personas que han mostrado interés en los productos o servicios de la empresa pero que aún no han sido calificados como oportunidades reales. Salesforce permite gestionar su proceso de conversión en Cuentas, Contactos y Oportunidades.

- **Gestión de Oportunidades**: Una Oportunidad representa un potencial acuerdo comercial en curso. Salesforce permite definir las etapas (stages) del avance del proceso de venta para su seguimiento y la previsión de conversión a una venta real.

- **Catálogo de Productos y Listas de Precios**: Sales Cloud permite definir un catálogo de productos (objeto estándar Product2) con sus correspondientes listas de precios (PriceBook), y asociarlos a oportunidades para generar presupuestos detallados (Quotes) y, finalmente, pedidos (Orders).

- **Actividades**: Correos electrónicos, llamadas telefónicas, reuniones y tareas pueden registrarse directamente sobre los registros, manteniendo así un historial completo de la interacción con cada cliente.

- **Previsión de ventas (Forecasting)**: Permite a los responsables de ventas estimar los ingresos futuros basándose en el estado y el valor de las oportunidades abiertas en el pipeline.

3.2.2.- Service Cloud (renombrado a Agentforce Service)

Service Cloud [35] es el producto de Salesforce orientado a la gestión de la atención al cliente. Su elemento central es el Caso (Case), que representa una incidencia, consulta o solicitud de soporte abierta por un cliente.

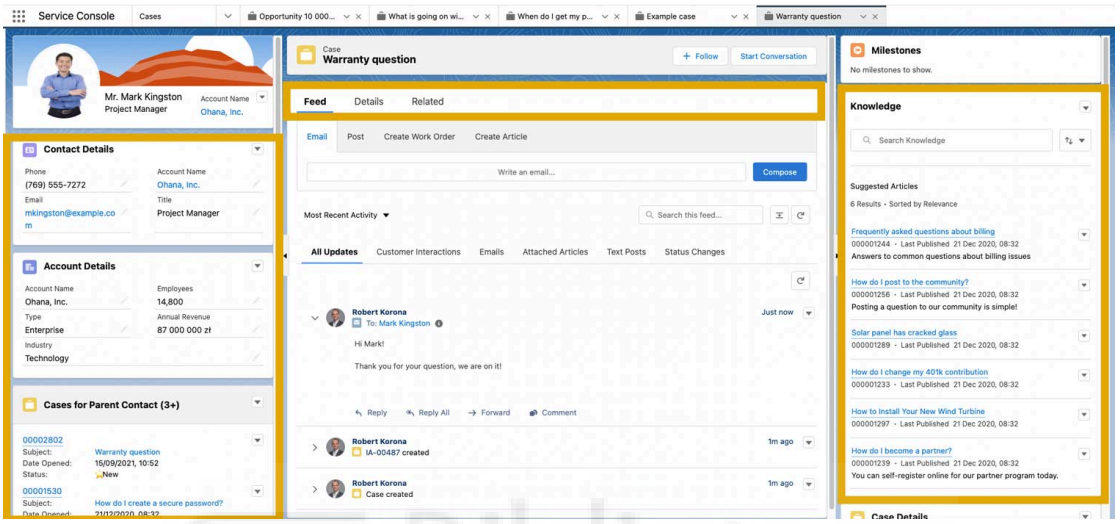

Figura 3.5: Interfaz de la consola de Service Cloud con un Caso de soporte abierto, el panel de registros relacionados (izquierda) y la búsqueda de artículos de Knowledge (derecha)

Entre sus funcionalidades principales destacan:

- **Gestión de Casos:** Permite registrar, asignar y hacer seguimiento de todas las solicitudes de soporte, manteniendo un historial completo de la comunicación con el cliente sobre cada caso.

- **Email-to-Case:** Funcionalidad que convierte automáticamente los correos electrónicos recibidos en una dirección de soporte en Casos de Salesforce [36], eliminando la necesidad de gestionar el soporte exclusivamente mediante hilos de correo electrónico.

- **Colas (Queues):** Permiten asignar casos a grupos de agentes en lugar de a un agente individual, facilitando la distribución equitativa de la carga de trabajo entre el equipo de soporte.

- **Salesforce Knowledge:** Permite crear y gestionar artículos de conocimiento [37] que los agentes pueden consultar para resolver incidencias de forma más eficiente y que, opcionalmente, pueden publicarse para que los propios clientes los consulten de forma autónoma.

- **SLA y Entitlements:** Los Service Level Agreement (SLAs) permiten definir los tiempos de respuesta y resolución comprometidos con los clientes, con alertas automáticas cuando se aproxima el incumplimiento del plazo acordado.

- **Consola de Service Cloud:** Entorno de trabajo optimizado para los agentes de soporte, que permite gestionar múltiples casos simultáneamente desde una única pantalla con paneles configurables.

Tabla 3.1: Comparativa entre Sales Cloud y Service Cloud

	Sales Cloud	**Service Cloud**
Funciones principales	- Gestión del proceso o pipeline de venta - Generación y cualificación de Leads - Seguimiento de Oportunidades de venta - Gestión de presupuestos (Quotes) y pedidos (Orders) - Previsión de ventas (Forecasting)	- Gestión de casos de soporte (Cases) - Comunicación multicanal (Email, Chat web, WhatsApp, teléfono) - Base de conocimiento (Salesforce Knowledge) - Gestión de SLAs y Entitlements - Asignación de casos mediante colas (Queues)
Orientado a	- Agentes comerciales - Jefes de ventas - Directores comerciales	- Agentes de atención al cliente - Managers de atención al cliente

3.2.3.- Experience Cloud

Experience Cloud [38] (anteriormente denominado Community Cloud) permite crear portales web externos conectados directamente a la organización de Salesforce, sin necesidad de desarrollar una aplicación web independiente. Estos portales pueden estar destinados a clientes, socios comerciales o empleados internos, y permiten exponer datos de Salesforce de forma controlada y segura a usuarios externos. En el contexto de IT Solutions, Experience Cloud se ha utilizado para crear un portal de cliente que permite a las empresas clientes:

- Consultar el estado de sus pedidos activos y el historial de compras.

- Abrir y hacer seguimiento de sus propios casos de soporte, sin necesidad de enviar un correo electrónico.

- Acceder a la base de conocimiento para resolver dudas de forma autónoma.

Experience Cloud utiliza el concepto de Sitio (Site) y su personalización se realiza mediante el Experience Builder, una herramienta visual de drag-and-drop que permite

diseñar las páginas del portal sin necesidad de escribir código. Los componentes disponibles en el Experience Builder incluyen tanto componentes estándar de Salesforce como componentes personalizados desarrollados con Flow o LWC (descritos en el apartado 3.3.1 y 3.3.2).

Los usuarios del portal son denominados usuarios de comunidad (Community Users) y disponen de licencias y perfiles específicos que limitan su acceso, de forma predeterminada, a únicamente sus propios registros, garantizando así la privacidad entre distintos clientes.

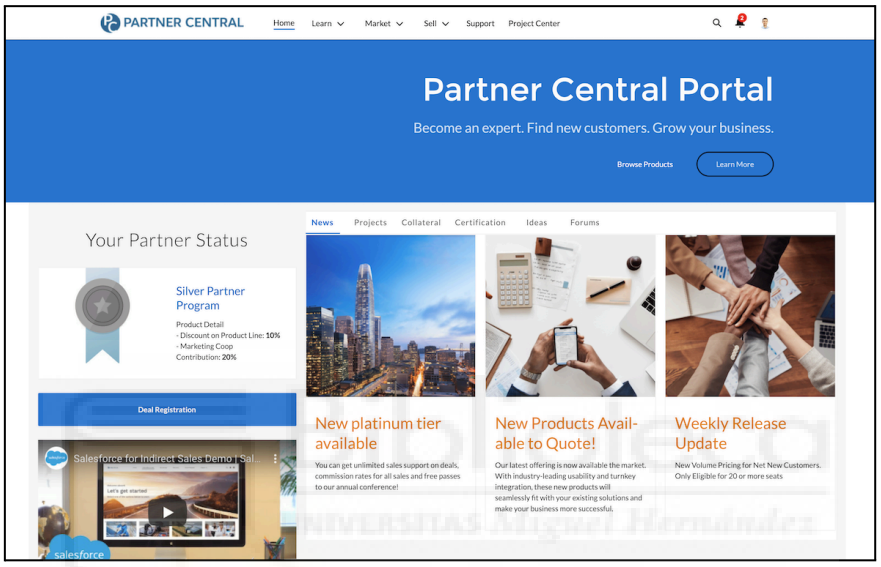

Figura 3.6: Ejemplo de portal de cliente creado con Experience Cloud Builder

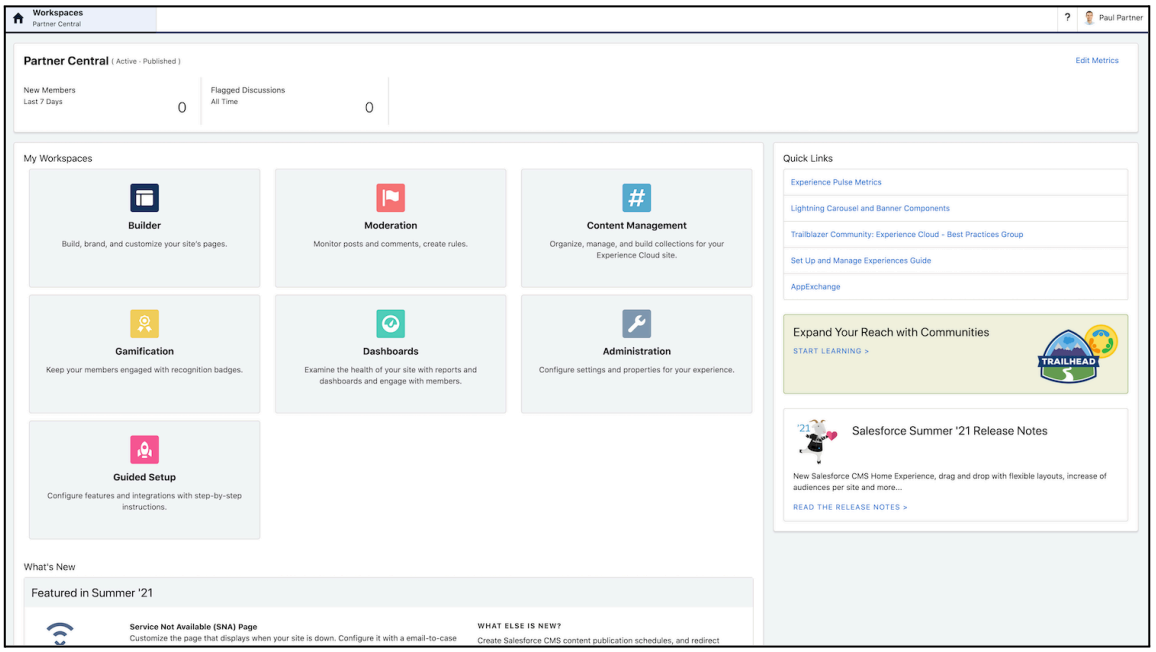

Figura 3.7: Panel de administración de Experience Cloud

3.3.- HERRAMIENTAS DE DESARROLLO Y AUTOMATIZACIÓN

3.3.1.- Flow Builder

Flow Builder [39] es la herramienta visual de automatización de Salesforce, y actualmente la recomendada por el fabricante para cubrir la totalidad de los casos de uso de automatización. Ha reemplazado a Process Builder y Workflow Rules, que han sido oficialmente retirados [40].

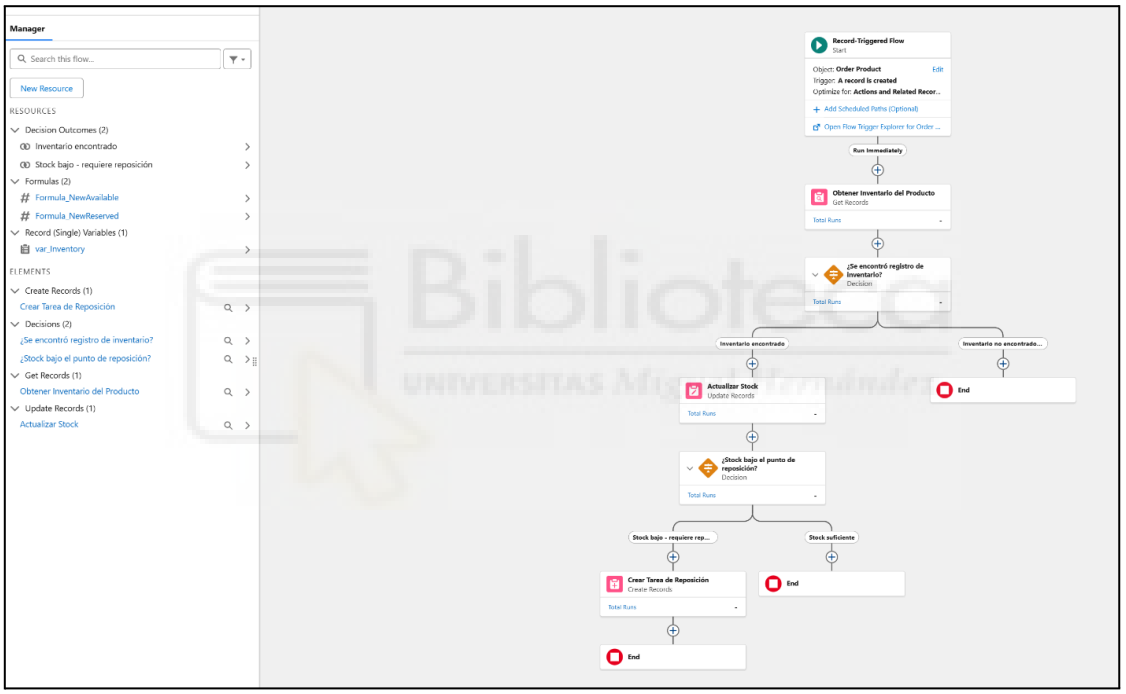

Figura 3.8: Interfaz de Flow Builder

Con Flow Builder se pueden crear distintos tipos de flujos según el caso de uso, entre los que destacan:

- **Record-Triggered Flows**: Se ejecutan automáticamente cuando se crea, actualiza o elimina un registro de Salesforce. Son la evolución de las antiguas Workflow Rules, pero con una capacidad de lógica muy superior.

- **Screen Flows**: Flujos que presentan pantallas interactivas al usuario, guiándole por un proceso paso a paso. Son especialmente útiles para implementar wizards o formularios guiados.

- **Scheduled Flows**: Se ejecutan de forma programada (por ejemplo, cada día a una hora determinada).

- **Autolaunched Flows**: Flujos invocados desde otro flujo, desde una clase Apex, desde el botón de una página, o por API. Se ejecutan en background y no presentan ningún tipo de UI al usuario.

Flow Builder cuenta con un extenso conjunto de elementos lógicos (decisiones, bucles, asignaciones, fórmulas, transformación de datos…) y acciones nativas que permiten interactuar con los datos de Salesforce (crear, leer, actualizar y eliminar registros), enviar correos electrónicos, invocar Apex y llamar a APIs externas.

3.3.2.- Lightning Web Components (LWCs)

Lightning Web Components [41] es el framework de componentes web moderno de Salesforce, introducido a finales de 2018 [42].

Está basado en los estándares web de Web Components [43] y utiliza HTML, CSS y JavaScript mediante un framework propio, LWC, lo que reduce considerablemente la curva de aprendizaje para desarrolladores web con experiencia previa.

Un componente LWC se compone de los siguientes archivos:

- **Archivo HTML (.html, opcional)**: Define la estructura y la plantilla del componente.

- **Archivo JavaScript (.js, obligatorio)**: Contiene la parte de lógica del componente que se ejecuta en el front.

- **Archivo de metadatos (.js-meta.xml, obligatorio)**: Define las propiedades del componente, como nombre, descripción, o en qué contextos puede utilizarse (página de record, página de aplicación, Experience Cloud, etc.).

- **Archivo CSS (.css, opcional)**: Para los estilos específicos del componente.

LWC implementa el concepto de Shadow DOM [44] para encapsular los estilos y el DOM de cada componente, evitando conflictos con otros componentes o con el entorno de la página en la que se inserta.

Los componentes pueden comunicarse entre sí mediante el paso de propiedades (comunicación padre-hijo), eventos custom del DOM (comunicación hijo-padre) o

mediante el servicio de mensajería de Lightning (Lightning Message Service), que sigue un modelo Pub/Sub, donde un componente publica eventos en un canal y los componentes suscritos lo escuchan (comunicación entre componentes en distinto DOM) [45].

Para acceder a los datos de Salesforce desde un componente LWC, se utilizan principalmente dos mecanismos [46]:

- **Wire Service**: Permite leer datos de Salesforce de forma reactiva, actualizando automáticamente el componente cuando los datos de la organización cambian. La lógica para obtener estos datos reside en el .js del componente. Los servicios Wire se ejecutan cuando carga el componente, no permite controlar el orden de ejecución. Se re-ejecuta automáticamente cuando cambian las propiedades reactivas que actúan como parámetros del wire adapter.

- **Apex**: Permite invocar métodos de clases Apex desde el JavaScript del componente para operaciones de lectura o escritura más complejas que no pueden cubrirse con el Wire Service, o si se necesita controlar el orden y el momento de ejecución de los métodos. Por ejemplo, si queremos realizar una acción en base de datos con el click de un botón en la interfaz, o si queremos llamar a un método en un momento concreto del flujo del componente.

3.3.2.1.- Directivas de plantillas HTML en LWC

Las directivas de plantilla o template directives en Lightning Web Components [47] permiten introducir lógica declarativa dentro de la plantilla HTML del componente. Estas directivas forman parte de la sintaxis propia de LWC y facilitan la implementación de renderizado condicional, iteraciones sobre colecciones de datos que residen en nuestro JavaScript y la asignación de propiedades de forma dinámica sin tener que manipular el DOM directamente.

A continuación se describen las directivas más relevantes y utilizadas para el desarrollo de componentes:

3.3.2.1.1.- Renderizado condicional

- lwc:if={expression}
- lwc:elseif={expression}
- lwc:else

Permiten renderizar u ocultar bloques enteros de la plantilla en función del valor booleano de una propiedad reactiva del componente. Estas directivas pueden eliminar o insertar el

nodo en el DOM según el estado evaluado, en lugar de limitarse a ocultarlo visualmente. Es decir, el HTML resultante incluirá o no el bloque de la plantilla, dependiendo del resultado.

Este mecanismo es una de las formas más habituales de implementar lógica condicional en LWC cuando no se requiere cambiar completamente la plantilla del componente, sino que solo se quiere cambiar una pequeña parte o sección de la plantilla entera ya cargada de antemano.

Algoritmo: 3.1: Ejemplo de uso de directivas HTML de renderizado condicional

```
// myComponent.js
import { LightningElement } from 'lwc';

export default class OpportunityTimeline extends LightningElement {
    renderBlockOne = false;
    renderBlockTwo = false;
}

// myComponent.html
<template>
    <template lwc:if={renderBlockOne}>
        <p>Block One Render</p>
    </template>

    <template lwc:elseif={renderBlockTwo}>
        <p>Block Two Render</p>
    </template>

    <template lwc:else>
        <!-- Se renderizará este bloque —->
        <p>Fallback Block Render</p>
    </template>
</template>
```

3.3.2.1.2.- Iteración sobre colecciones

- for:each={array}
- for:item=”currentItem”
- for:index=”index”
- key={identifier}

Estas directivas permiten renderizar listas de elementos a partir de una colección de objetos JavaScript.

La propiedad key es obligatoria y garantiza que el motor de renderizado pueda identificar de forma eficiente cada elemento durante los ciclos de actualización del DOM, optimizando el rendimiento.

Algoritmo: 3.2: Ejemplo de uso de directivas HTML de iteración sobre colecciones

```
// myComponent.js
import { LightningElement } from 'lwc';

export default class OpportunityTimeline extends LightningElement {
    contacts = [

        {
            Id: ‘1’,
            Name: ‘Fabio Barcelona’,
            Title: ‘Alumno’
        },

        {
            Id: ‘2’,
            Name: ‘Jesuja Rodríguez’,
            Title: ‘Tutor’
        }

    ];
}


// myComponent.html
<template>
    <ul>

        <template for:each={contacts} for:item=”contactItem”>

            <li key={contactItem.Id}>
                {contactItem.Name}, {contactItem.Title}
            </li>

        </template>

    </ul>
</template>
```

3.3.2.1.3.- Asociación dinámica de atributos

LWC permite enlazar dinámicamente atributos HTML con propiedades del componente mediante la sintaxis {propiedad}. Esta característica habilita un modelo reactivo en el que cualquier cambio en el estado del componente provoca automáticamente su actualización visual (volver a renderizar el componente).

Algoritmo: 3.3: Ejemplo de uso de directivas HTML de propiedades dinámicas

```
// myComponent.js
import { LightningElement } from 'lwc';

export default class OpportunityTimeline extends LightningElement {
    buttonLabel = ‘Confirmar’;
    isDisabled = false;
}


// myComponent.html
<template>

    <lightning-button label={buttonLabel} disabled={isDisabled}>
    </lightning-button>

</template>
```

3.3.2.2.- Lifecycle Hooks de un componente LWC

Tal y como se ha expuesto en apartados anteriores, Lightning Web Components (LWC) se fundamenta en el estándar de Web Components, por lo que adopta y adapta el modelo de ciclo de vida propio de esta especificación, en contraposición a otros frameworks como React o Angular, donde el ciclo de vida se gestiona mediante métodos propios del framework. En consecuencia, los componentes LWC disponen de una serie de lifecycle hooks que permiten ejecutar lógica en momentos concretos de la existencia del componente dentro del Document Object Model (DOM).

A continuación, se describen los distintos hooks disponibles en LWC, siguiendo su orden de ejecución, así como los usos habituales asociados a cada uno de ellos.

Es importante señalar que ninguno de estos métodos es de implementación obligatoria. No obstante, su conocimiento resulta esencial para desarrollar componentes robustos y bien estructurados, ya que permiten reaccionar de forma controlada a eventos relevantes como la inserción, actualización o eliminación del componente en el DOM.

Uno de los casos de uso más habituales consiste en invocar un controlador Apex para recuperar datos del servidor en el momento en que el componente ha sido insertado en el DOM, garantizando así que la información se encuentre disponible para su renderización y posterior interacción con el usuario.

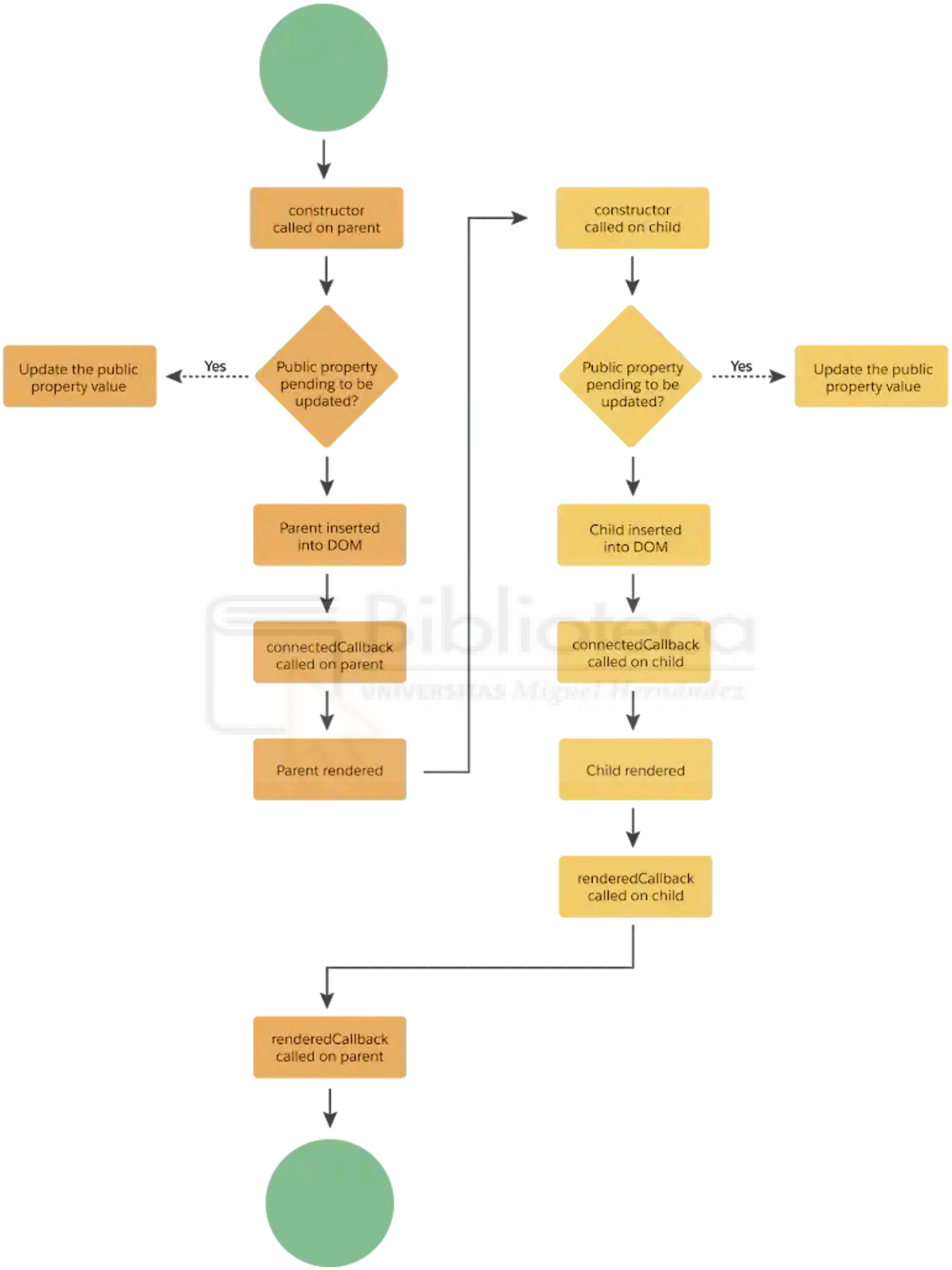

Figura 3.9: Flujo del ciclo de vida de un componente, desde la creación hasta el render

Tabla 3.2: Lifecycle Hooks en LWC y su uso más común

Lifecycle Hook	Uso común
constructor()	Se ejecuta en el momento de creación de la instancia del componente. En esta fase el componente aún no ha sido insertado en el DOM y, por tanto, no se tiene acceso a los elementos renderizados. Se emplea principalmente para inicializar propiedades internas del componente. No es recomendable realizar peticiones al servidor ni interactuar con el DOM en este punto del ciclo de vida.
connectedCallback()	Se invoca cuando el componente es insertado en el DOM. Puede ejecutarse más de una vez si el componente se elimina y vuelve a insertarse. Es adecuado para realizar tareas de inicialización que dependan de su presencia en el DOM, como la creación de event listeners o la ejecución de peticiones asíncronas a métodos Apex para recuperar datos del servidor.
render()	Método opcional que permite sobrescribir la plantilla HTML que el componente debe utilizar en cada renderizado. Devuelve la referencia a la plantilla correspondiente. Su uso es apropiado cuando se requiere seleccionar dinámicamente entre múltiples templates en función del estado interno del componente. En la mayoría de los casos no es necesario implementarlo, ya que LWC asigna automáticamente la plantilla principal asociada al componente.
renderedCallback()	Se ejecuta después de que el componente haya sido renderizado en el DOM. Este método puede ejecutarse múltiples veces a lo largo del ciclo de vida del componente, especialmente cuando cambian propiedades reactivas, por lo que debe utilizarse con precaución para evitar ejecuciones innecesarias o bucles de renderizado. Es útil para realizar manipulaciones del DOM que requieran que los elementos ya estén presentes. Se debe evitar modificar propiedades reactivas dentro de este método, ya que podría provocar bucles de renderizado.
disconnectedCallback()	Se ejecuta cuando el componente es eliminado del DOM. Es el lugar adecuado para liberar recursos, eliminar listeners previamente registrados, cancelar temporizadores (setTimeout, setInterval) o finalizar suscripciones activas a canales de Lightning Message Service.
errorCallback()	Se invoca cuando se produce un error en alguno de los componentes hijos. Permite implementar un mecanismo de manejo de errores centralizado dentro del conjunto de componentes hijo, facilitando estrategias de control de excepciones en la interfaz.

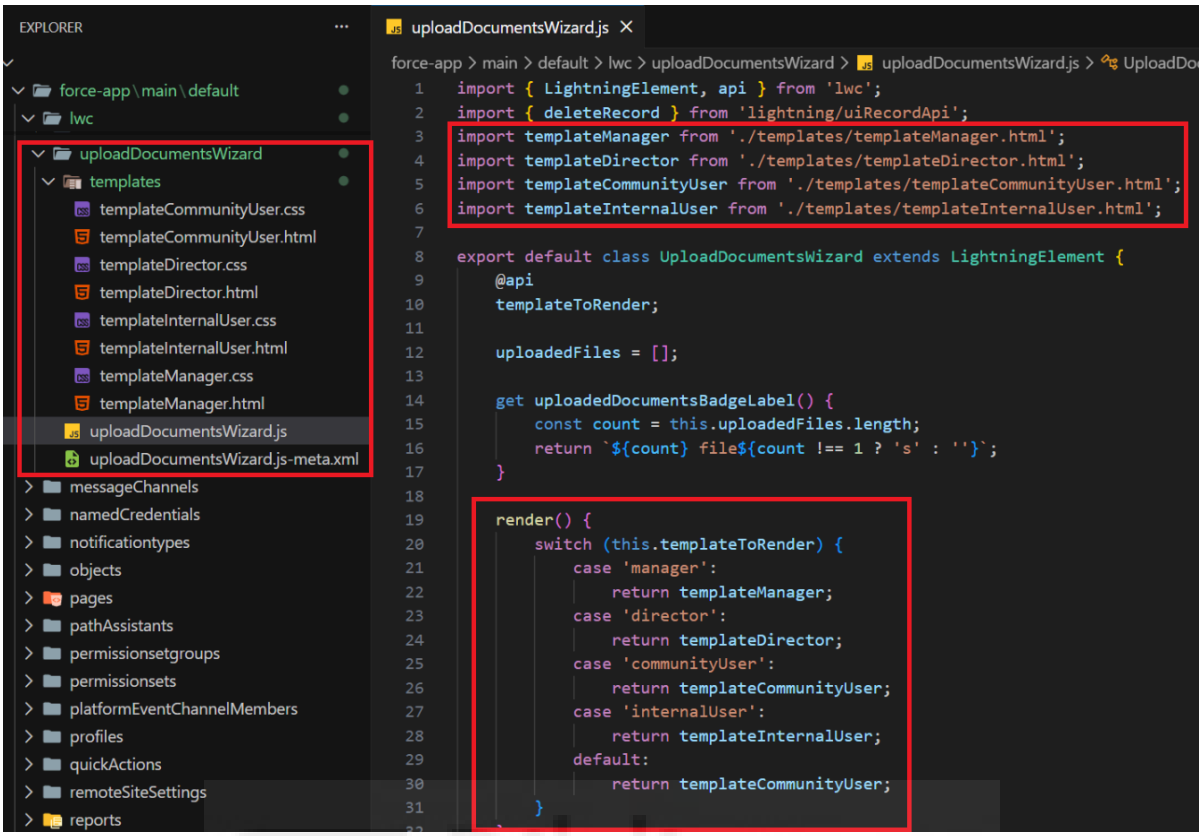

Figura 3.10: Ejemplo de uso de render() cargando un HTML según cierta lógica.

En la Figura 3.10 se observa un ejemplo práctico de sobreescritura del método **render()**, mediante el cual se selecciona dinámicamente la plantilla HTML a utilizar en función del valor de una propiedad reactiva del componente.

Algoritmo: 3.4: Ejemplo de uso de connectedCallback()

```
1  import { LightningElement, api } from 'lwc';
2  import getOpportunityHistory from
3  '@salesforce/apex/OpportunityTimelineController.getOpportunityHistory';
4
5  export default class OpportunityTimeline extends LightningElement {
6      @api recordId; // ID de la Opportunity
7
8      async connectedCallback() {
9          try {
10             const result = await getOpportunityHistory({ opportunityId:
11 this.recordId });
12             // Procesar el resultado de Apex
13         } catch (error) {
14             throw new Error('Error fetching opportunity history: ' +
15 error.message);
16         }
17     }
18 }
```

3.3.2.2.1.- Comparativa render() y directivas HTML condicionales

Mientras que las directivas de plantilla permiten introducir lógica condicional y repetitiva dentro de una misma estructura HTML, el método render() ofrece un nivel adicional de control, al permitir seleccionar completamente una plantilla diferente en función del estado del componente. Por tanto, el uso de directivas es adecuado para variaciones internas de contenido, mientras que render() resulta más apropiado cuando la estructura del componente difiere sustancialmente según el contexto.

3.3.2.3.- Consideraciones sobre el modelo reactivo en LWC

El modelo de desarrollo de Lightning Web Components se fundamenta en un enfoque reactivo, en el cual la interfaz de usuario se actualiza automáticamente cuando cambian las propiedades públicas de la clase JavaScript y que están presentes en la plantilla HTML o se usan en getters. Este paradigma elimina la necesidad de manipulación manual del DOM, reduciendo la complejidad del código y favoreciendo una mayor mantenibilidad y previsibilidad del comportamiento del componente.

Cuando una de estas propiedades públicas se modifica en JavaScript, el motor de renderizado evalúa qué partes del DOM deben actualizarse. Este mecanismo permite que el desarrollador se centre en el desarrollo de la lógica de negocio, delegando la sincronización vista-estado al framework.

No obstante, este modelo también implica ciertas consideraciones de diseño. Por ejemplo, el uso inadecuado de métodos como renderedCallback() puede provocar ejecuciones repetidas innecesarias si no se controla la lógica interna. Del mismo modo, en estructuras iterativas (for:each), la correcta asignación de claves únicas (key) resulta esencial para garantizar un re-renderizado eficiente y evitar comportamientos inesperados o errores.

En consecuencia, el desarrollo de componentes LWC no debe limitarse al conocimiento sintáctico de sus APIs, sino que requiere comprender las implicaciones arquitectónicas del modelo reactivo, especialmente en términos de rendimiento, escalabilidad y mantenibilidad a largo plazo.

3.3.2.4.- Salesforce Lightning Design System (SLDS)

Salesforce Lightning Design System (SLDS) [48] es el sistema de diseño oficial de Salesforce, compuesto por un conjunto de directrices visuales, componentes CSS y tokens

de diseño (design tokens) que garantizan la coherencia visual entre los componentes desarrollados a medida y la interfaz nativa de Lightning Experience.

Su versión actual es la SLDS 2, y está basado en estándares web modernos y permite el uso de styling hooks, variables CSS que permiten personalizar la apariencia de los componentes sin sobrescribir estilos directamente, simplemente dándole un valor nuevo dentro de nuestro componente a una variable CSS ya existente, heredada por el CSS precargado de Salesforce.

En el desarrollo de componentes LWC, SLDS se aplica mediante clases CSS predefinidas (por ejemplo, slds-button, slds-card, slds-timeline__item) que se referencian directamente en las plantillas HTML.

Salesforce recomienda su uso en todo desarrollo custom para asegurar que los componentes sean visualmente consistentes con la plataforma en todas sus versiones y dispositivos. El hecho de que esta biblioteca de clases CSS sea gestionada por Salesforce mismo, permite despreocuparse de actualizar el estilo visual de nuestros componentes si introducen cambios.

Mencionar que, aunque podemos sobreescribir el comportamiento de estas clases SLDS en nuestro componente, esto se trata de un antipatrón y deberemos evitarlo por completo. Si necesitamos un estilo custom, deberemos crear una clase CSS custom.

3.3.2.5.- Lightning Component Library

La Lightning Component Library [49] es la colección de componentes de interfaz de usuario desarrollados y mantenidos por Salesforce que sirven como bloques de construcción de Salesforce Platform en versión desktop, la app móvil de Salesforce y los sitios de Experience Cloud. Son, en esencia, los mismos componentes con los que Salesforce construye su propia interfaz nativa, y están disponibles para ser reutilizados directamente dentro de cualquier componente LWC personalizado que creemos.

Cada componente base implementa SLDS, incluyendo estilos, accesibilidad y comportamiento adaptativo según el dispositivo, lo que significa que su uso en un componente custom garantiza la coherencia visual con la plataforma sin necesidad de escribir CSS adicional o incluir clases de SLDS en nuestro HTML. Asimismo, los componentes base detectan automáticamente si la organización utiliza SLDS 1 o SLDS 2 y adaptan su apariencia en consecuencia.

La librería incluye componentes para los patrones de UI más habituales en el desarrollo de aplicaciones de negocio: **lightning-button** para acciones, **lightning-card** como

contenedor, **lightning-input** para formularios, **lightning-spinner** para indicar estados de carga y bloquear la interacción con el componente de forma total o parcial, **lightning-datatable** para la representación de tablas de registros, o **lightning-record-form** para crear formularios de creación y edición de registros sin apenas código.

En el presente proyecto, algunos componentes de esta librería son utilizados en los tres LWC desarrollados para construir la estructura visual, los formularios de búsqueda y los elementos de interacción del usuario.

La documentación oficial de cada componente detalla sus propiedades, los eventos del DOM que emite y los métodos públicos que expone, lo que facilita su integración en componentes compuestos siguiendo el modelo de comunicación padre-hijo de LWC.

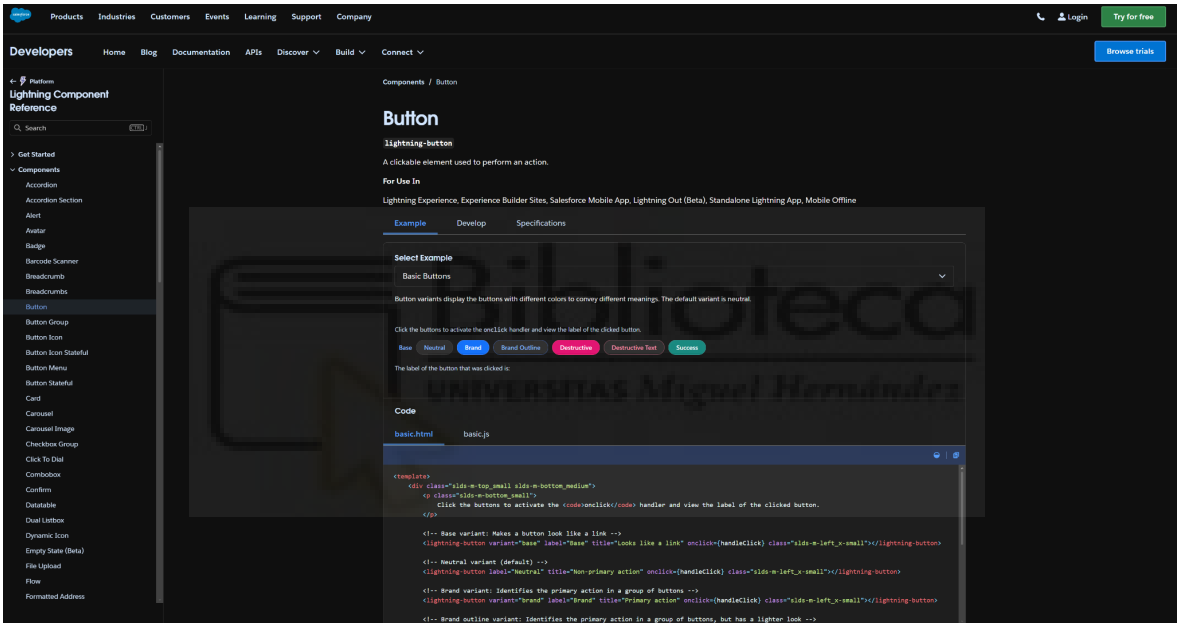

Figura 3.11: Documentación del componente lightning-button en la Lightning Component Library con ejemplos renderizados y su código

3.3.3.- Apex

Apex [50] es el lenguaje de programación propietario de Salesforce. Su sintaxis es similar a Java, con tipado estático y orientación a objetos, y permite ejecutar lógica de negocio compleja directamente en los servidores de Salesforce. Se recurre a Apex cuando las herramientas declarativas (como Flow Builder o las fórmulas) no son suficientes para cubrir un caso de uso concreto. Apex puede ejecutarse en distintos contextos:

- **Triggers**: Se ejecutan automáticamente antes o después de operaciones de manipulación de datos (insert, update, delete, undelete) sobre registros de

Salesforce. Se debe seguir el patrón "un trigger por objeto" recomendado por las buenas prácticas de la plataforma, ya que en caso de haber múltiples trigger para el mismo evento, no se garantiza el orden.

- **Clases y métodos**: Código reutilizable que puede ser invocado desde Flows, componentes LWC, otras clases Apex o la API REST de Salesforce.

- **Clases de Test**: Salesforce exige que al menos el 75% del código Apex esté cubierto por tests unitarios antes de poder desplegarlo en una organización de producción, garantizando así un mínimo de calidad del código.

- **HTTP Callouts**: Permiten realizar peticiones a APIs REST o SOAP externas desde Salesforce, posibilitando la integración con sistemas de terceros.

- **Batch Apex**: Permite procesar grandes volúmenes de registros de forma asíncrona, con Governor Limits muy superiores a los que aplicarían a un procesamiento síncrono.

- **Queueable Apex**: Permite ejecutar operaciones asíncronas de forma más flexible que el Batch Apex, con soporte para el encadenamiento de trabajos (job chaining) y el uso de tipos de datos complejos como sObjects. Es la opción recomendada cuando se necesita procesamiento asíncrono puntual sin la sobrecarga del framework de Batch.

- **Scheduled Apex**: Permite programar la ejecución de una clase Apex a una hora y frecuencia determinadas, mediante la implementación de la interfaz Schedulable. Resulta especialmente útil para tareas de mantenimiento periódico, como la sincronización nocturna de datos entre sistemas, o la regularización de registros existentes en Salesforce.

3.3.3.1.- SOQL y SOSL

Para la consulta y recuperación de datos dentro de la plataforma, Apex dispone de dos lenguajes de consulta propietarios [51].

El primero es SOQL (Salesforce Object Query Language), cuya sintaxis es intencionalmente similar a SQL pero adaptada al modelo de objetos de Salesforce: permite filtrar, ordenar y paginar registros de un único objeto o de objetos relacionados mediante subconsultas, y está sujeto a los Governor Limits de la plataforma, en particular al límite de 50.000 registros recuperables por transacción y un máximo de 100 consultas por transacción síncrona, y 200 en operaciones asíncronas.

El segundo es SOSL (Salesforce Object Search Language), orientado a la búsqueda en campos de texto en múltiples objetos simultáneamente en una sola consulta; resulta especialmente útil cuando se desconoce el objeto exacto en el que reside la información buscada, aunque su uso es mucho menos frecuente que SOQL en el desarrollo de lógica de negocio estructurada.

En el presente proyecto, SOQL se utiliza en prácticamente todas las clases Apex para la recuperación de datos de objetos como **Opportunity**, **Product2**, **Inventory__c** o **Shipment_Tracking__c**.

3.4.- HERRAMIENTAS AUXILIARES

3.4.1.- Salesforce CLI y Visual Studio Code

La Salesforce CLI [52] es una herramienta de línea de comandos oficial que permite interactuar con organizaciones de Salesforce directamente desde el terminal: autenticarse, desplegar o recuperar metadatos, ejecutar queries SOQL, ejecutar tests Apex, entre otras operaciones. Muy útil para trabajar aplicar cambios a la organización desde el propio entorno de desarrollo/terminal y obligatoria para procesos de CI/CD.

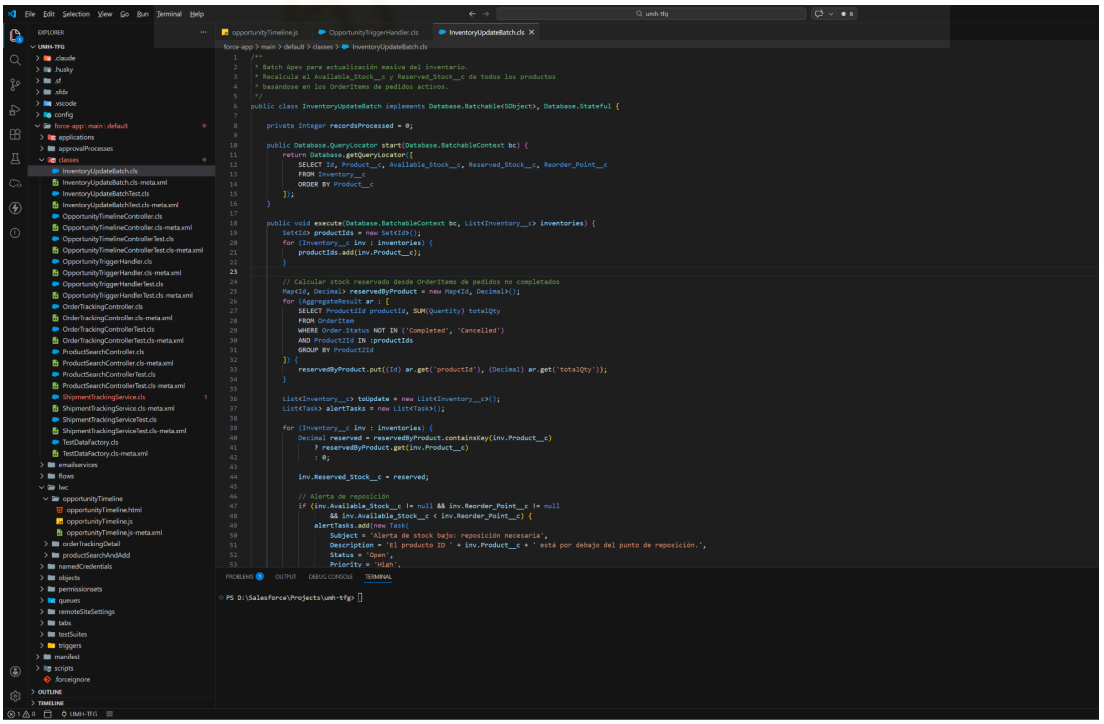

Figura 3.12: Entorno de desarrollo con Visual Studio Code, mostrando una clase Apex

Para el desarrollo de componentes LWC y clases Apex se ha utilizado Visual Studio Code (VS Code) junto con la extensión oficial Salesforce Extension Pack [53][54]. Esta extensión proporciona las siguientes características:

- Resaltado de sintaxis y autocompletado inteligente para Apex, SOQL y las plantillas HTML de LWC.

- Integración con la Salesforce CLI para desplegar y recuperar metadatos directamente desde el IDE.

- Ejecución y visualización de los resultados de tests Apex desde el propio editor.

3.4.2.- Trailhead y Developer Edition

Trailhead es la plataforma gratuita de aprendizaje interactivo de Salesforce, estructurada en módulos, proyectos y superbadges con un sistema de puntos y gamificación.

Ha sido utilizada tanto como fuente de referencia técnica junto a la documentación oficial, como herramienta de aprendizaje de funcionalidades específicas de la plataforma que no han sido previamente utilizadas por el autor.

La Developer Edition, como bien se ha justificado en los apartados 3.1.3 y 3.1.3.1, es el entorno idóneo para el desarrollo y la demostración de la solución sin coste económico.

3.4.3.- Git, GitHub y CI/CD orientado a Salesforce

El modelo de proyecto Salesforce Developer Experience (SFDX) establece que el sistema de control de versiones debe actuar como fuente de verdad (source of truth) de todos los metadatos del proyecto [55].

En este contexto, Git es la herramienta de control de versiones utilizada, y GitHub el servicio de alojamiento del repositorio remoto. La estructura de un proyecto SFDX es directamente compatible con Git: el directorio **force-app/** contiene los metadatos en formato .CLS (Apex Class), .JS/.HTML/.XML/.CSS (LWC) o .XML (Flows, Objetos, Campos, etc), lo que permite hacer seguimiento de cambios a nivel de fichero individual y revertirlos cuando sea necesario.

El fichero **.forceignore**, similar al .gitignore, define qué elementos del proyecto se excluyen de las operaciones de deploy/retrieve de metadatos entre el proyecto SFDX local y la propia organización de Salesforce a la que estamos conectados a través del CLI [56].

En entornos de desarrollo profesional, Git y SFDX se combinan con herramientas de integración y despliegue continuos (CI/CD). Una de las prácticas más extendidas en el ecosistema Salesforce consiste en utilizar GitHub Actions [57], que permite definir flujos de trabajo automatizados en ficheros YAML que se disparan ante eventos del repositorio (por ejemplo, al hacer push a una rama o al abrir un pull request).

Un pipeline típico incluye pasos como la ejecución de análisis estático del código Apex mediante Salesforce Code Analyzer [58], el despliegue de metadatos contra una Scratch Org temporal, la ejecución de los tests unitarios Apex y, si todos pasan, el despliegue automático al entorno de destino.

Alternativas a GitHub Actions en este contexto son Jenkins [59], un servidor de automatización CI/CD de código abierto ampliamente utilizado en entornos empresariales complejos. Como Jenkins, encontramos muchas más opciones, como TravisCI, CircleCI, Azure DevOps, Bitbucket Pipelines, por nombrar unos pocos.

Por último, también existen herramientas nativas de Salesforce como DevOps Center [60], la solución low-code (aunque limitada en comparación con las alternativas anteriormente propuestas) de la propia plataforma para la gestión de pipelines de despliegue entre entornos relacionados.

Para el presente proyecto, el sistema de control de versiones empleado es Git, con GitHub como repositorio remoto. La puesta en marcha de un pipeline de CI/CD completamente automatizada, si bien queda fuera del alcance de este trabajo; dado que el entorno de desarrollo es una Developer Edition individual y no un flujo de despliegue multi-entorno en producción, sí se considera una práctica fundamental en cualquier proyecto Salesforce profesional serio y real.

3.4.4.- Salesforce Inspector Reloaded

Salesforce Inspector Reloaded [61] es una extensión para los navegadores Chrome y Firefox desarrollada y mantenida por la comunidad Salesforce, con origen en el proyecto original Chrome Salesforce Inspector de Soren Krabbe, el cual fue descontinuado por el propio autor. La extensión superpone una interfaz de herramientas de desarrollo directamente sobre la UI de Salesforce, sin necesidad de acceder a entornos o herramientas externas ni de utilizar la línea de comandos, lo que la convierte en una herramienta de productividad prácticamente obligatoria que simplifica mucho las tareas cotidianas para los desarrolladores o administradores de entornos Salesforce.

Sus funcionalidades principales incluyen la ejecución de queries SOQL directamente contra la organización, con posibilidad de exportación de resultados en formato CSV, JSON o Excel, la importación masiva de datos en formato CSV o Excel, la inspección de la estructura de metadatos, también conocido como Schema, de cualquier objeto (campos, tipos, relaciones y permisos), el acceso directo a los registros por Id; lo que nos permite ver todos los valores de los campos de un registro, incluso si no están presentes en el layout que vería un usuario normal por UI, un explorador de la API REST de Salesforce y el acceso al registro debug logs sin salir de la pantalla actual.

En el contexto del presente proyecto, se ha utilizado principalmente para validar consultas SOQL durante el desarrollo de las clases Apex, inspeccionar la estructura de los objetos personalizados creados.

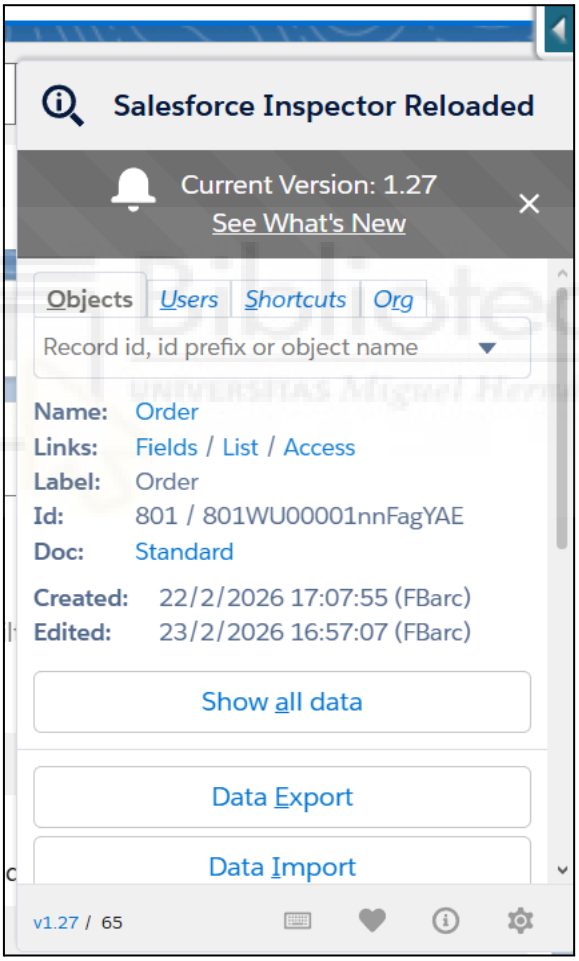

Figura 3.13: Interfaz principal de Salesforce Inspector Reloaded

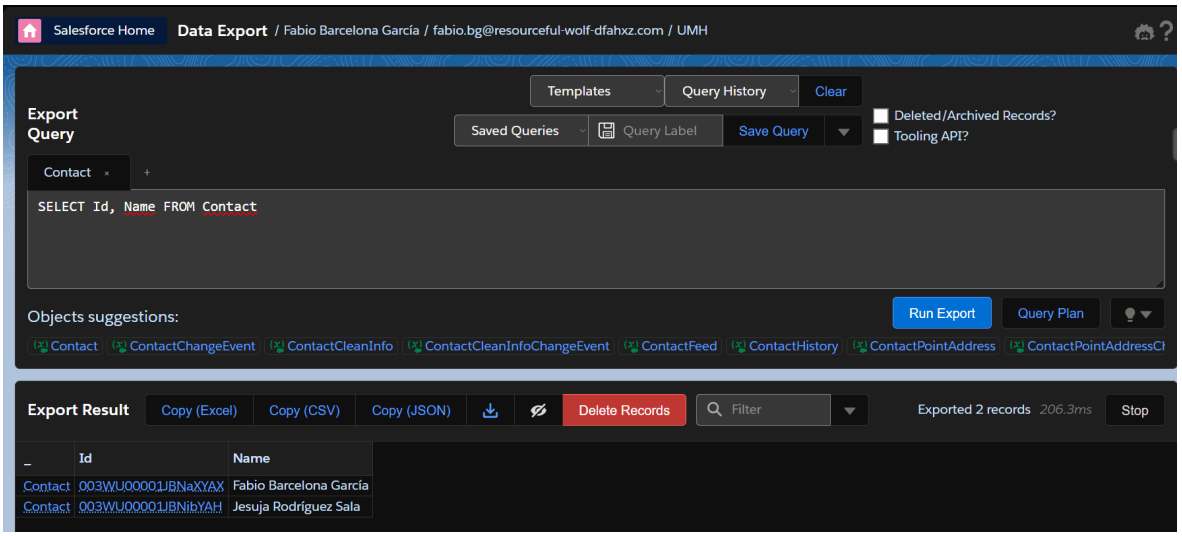

Figura 3.14: Interfaz de queries y panel de resultados con botones de acción de Salesforce Inspector Reloaded

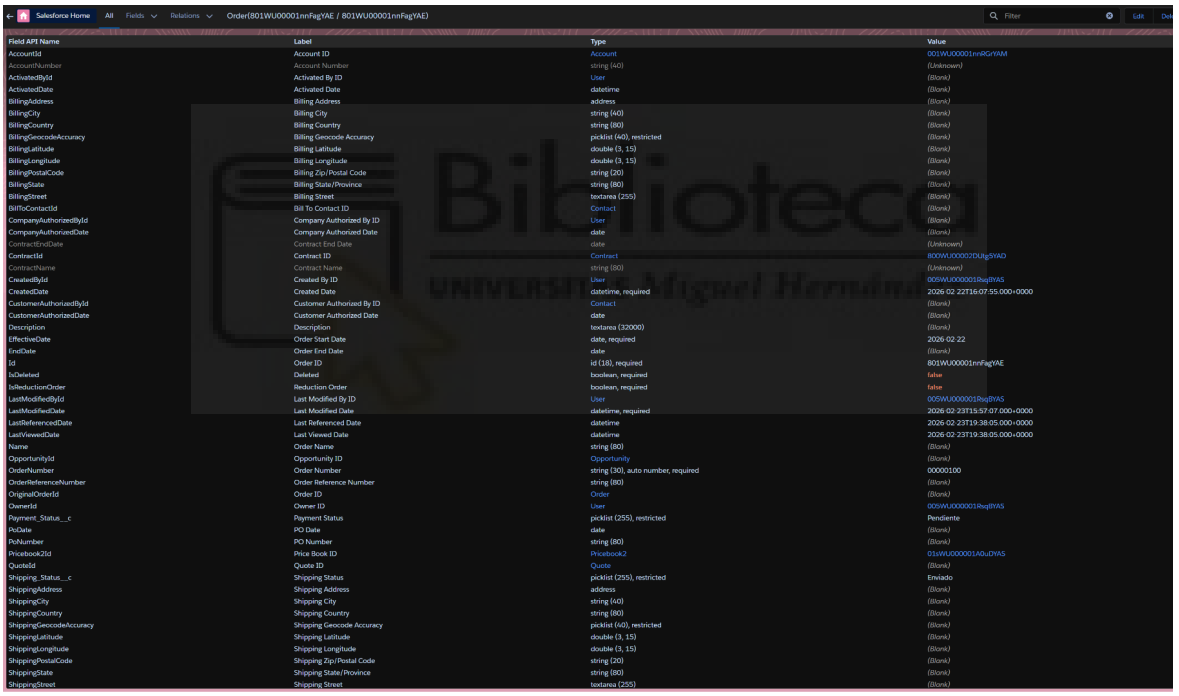

Figura 3.15: Visualización de todos los valores de un registro en Salesforce Inspector Reloaded

3.5.- RESUMEN DE TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS

La siguiente tabla recoge todas las tecnologías, lenguajes y herramientas empleadas en el desarrollo del proyecto, junto con su versión a fecha de confección del trabajo y el propósito que cumplen dentro del mismo.

Tabla 3.3: Resumen de las tecnologías a usar, su versión y propósito dentro del proyecto

Tecnología / Herramienta	Versión	Propósito en el proyecto
Salesforce Platform	66.0 (Release Spring '26, Patch 8.2 / 260.8.2)	Plataforma de ejecución de la solución
Apex	66.0	Lógica de negocio backend y triggers
SOQL / SOSL	66.0	Consulta y búsqueda de datos en la plataforma
Lightning Web Components (LWC)	66.0	Componentes de interfaz de usuario custom
Salesforce Lightning Design System (SLDS 2)	2.3.0	Sistema de estilos y coherencia visual
Flow Builder	66.0	Automatizaciones declarativas de procesos
Sales Cloud	66.0	Gestión del proceso de ventas
Service Cloud	66.0	Gestión de soporte y atención al cliente
Experience Cloud	66.0	Portal web de clientes externos
Salesforce CLI	2.123.1	Despliegue de metadatos y gestión de la org
Visual Studio Code	1.109.5	Entorno de desarrollo integrado (IDE)
Salesforce Extension Pack (VS Code)	65.18.0	Integración IDE y Salesforce CLI. Soporte Apex, LWC y SOQL en el IDE
Git	2.50.1.windows.1	Control de versiones del proyecto
GitHub	-	Repositorio remoto y colaboración
Salesforce Inspector Reloaded	1.27	Inspección de metadatos y validación de SOQL

3.6.- ESTÁNDARES Y BUENAS PRÁCTICAS DE DESARROLLO

El desarrollo de aplicaciones sobre Salesforce implica trabajar bajo un conjunto de restricciones técnicas y arquitectónicas, los Governor Limits descritos en la sección 3.1.2 que hacen que las buenas prácticas no sean simples recomendaciones, sino requisitos funcionales. El incumplimiento de ciertos patrones no solo produce código difícil de entender y mantener, también código que falla en entornos productivos cuando el volumen de datos supera el umbral de los límites impuestos por transacción. A continuación se describen los estándares y patrones que han guiado el desarrollo del proyecto.

3.6.1.- Bulkificación del código Apex

En Salesforce, cualquier operación que modifique registros (una importación masiva, una integración externa, un proceso de automatización que trabaje sobre N registros) puede invocar un trigger Apex con hasta 200 registros simultáneos en una sola transacción [62]. Por ello, el principio fundamental del desarrollo Apex es la **bulkificación**: todo el código debe diseñarse para procesar colecciones de registros y nunca asumir que la transacción contiene un único registro.

El antipatrón más clásico es la ejecución de consultas SOQL o de operaciones DML dentro de un bucle, lo que provoca que cada iteración del bucle consuma una unidad de los Governor Limits, pudiendo agotar el límite de 100 consultas o 150 operaciones DML permitidas por transacción.

La solución es acumular los Ids necesarios en colecciones (List, Set, Map), ejecutar la consulta o la operación DML usando dicha colección de Ids como filtro una única vez fuera del bucle, y procesar los resultados dentro de él.

Algoritmo: 3.5: Ejemplo de antipatrón bulk y su corrección en Apex

```
1  // Antipatrón: SOQL dentro de un bucle
2  for (Order__c o : Trigger.new) {
3      // En cada iteración del bucle, consumimos una query de 100
4      Account acc = [SELECT Id FROM Account WHERE Id = :o.AccountId];
5  }
6
7  // Patrón correcto: consulta única fuera del bucle, acceso mediante Map
8  Set<Id> accountIds = new Set<Id>();
9  for (Order__c o : Trigger.new) {
10     accountIds.add(o.AccountId);
11 }
12
13 Map<Id, Account> accountsById = new Map<Id, Account>(
14     [SELECT Id, Name FROM Account WHERE Id IN :accountIds]
15 );
```

3.6.2.- Patrón de un trigger por objeto

Salesforce no garantiza el orden de ejecución cuando existen múltiples triggers definidos sobre el mismo objeto y el mismo evento [63]. Por ello, la práctica universalmente aceptada es definir un único trigger por objeto que no contenga ningún tipo de lógica, sino que se limite a invocar a una clase handler: una clase Apex que contiene toda la lógica organizada por evento (before insert, after update, etc.) [64].

Este patrón garantiza un único punto de entrada predecible para toda la lógica de negocio del objeto, simplifica el mantenimiento y facilita la escritura de tests unitarios sobre el handler en lugar de sobre el propio trigger.

Algoritmo: 3.6: Ejemplo de patrón de un solo trigger por objeto

```
// OpportunityTrigger.trigger
trigger OpportunityTrigger on Opportunity (before update) {
  if (Trigger.isBefore && Trigger.isUpdate) {
    OpportunityTriggerHandler.beforeUpdate(Trigger.new, Trigger.oldMap);
  }
}

//OpportunityTriggerHandler.cls
public class OpportunityTriggerHandler {
  public static void beforeUpdate(List<Opportunity> newOpps, Map<Id,
                                  Opportunity>   oldOppMap) {
      // Entry point de la lógica del trigger
    }
  }
}
```

En el presente proyecto, este patrón se ha aplicado correctamente: cada objeto con lógica Apex dispone de un único trigger que delega inmediatamente en su clase handler, sin contener lógica propia.

3.6.3.- Separación de responsabilidades

Para proyectos de mayor envergadura, el ecosistema Salesforce ha desarrollado los Apex Enterprise Patterns (AEP) [65][66], un conjunto de patrones arquitectónicos introducidos por Andrew Fawcett [67] e implementados en la librería open source fflib [68], que propone una separación clara del código en tres capas:

- **Capa de Servicio (Service Layer)**: contiene la lógica de negocio transaccional, agnóstica del origen de la invocación, puede ser invocada desde un trigger, un controlador LWC, una API externa o un Flow.

- **Capa de Dominio (Domain Layer)**: encapsula la lógica orientada al objeto, especialmente la que se ejecuta en contexto de trigger.

- **Capa de Selector (Selector Layer)**: centraliza todas las consultas SOQL en clases especializadas, garantizando su reutilización y la correcta aplicación de Field-Level Security (FLS).

En el presente proyecto, dado su carácter académico y alcance acotado, se ha aplicado una versión simplificada de esta arquitectura: los controladores Apex de los componentes LWC están separados de la lógica de negocio, que reside en clases de servicio independientes, y las consultas SOQL se concentran en métodos de acceso a datos reutilizables.

Esta estructura, aunque no implementa la librería fflib en su totalidad, sigue el principio de separación de responsabilidades que los patrones fomentan.

3.6.4.- Estándares de testing y calidad del código

Como se mencionó en la sección 3.3.3, Salesforce exige un mínimo del 75% de cobertura de código Apex para poder desplegar en producción. Sin embargo, alcanzar ese umbral sin garantizar la calidad de los tests es un antipatrón bien documentado [69]: una clase puede estar cubierta sin que existan aserciones que validen el comportamiento real del código.

Los estándares seguidos en el desarrollo de los tests de este proyecto son los siguientes [70]:

- **Test Data Factory**: toda la creación de datos de prueba está centralizada en una clase **TestDataFactory** anotada con **@IsTest**, que proporciona métodos estáticos para generar registros configurables. Esto evita la duplicación de código entre tests y garantiza que los datos de prueba no dependen del estado de la organización en el momento de la ejecución, sino que se crean datos nuevos para cada test al vuelo.

- **Escenarios cubiertos**: para cada clase, los tests cubren al menos tres escenarios: el caso positivo (comportamiento esperado con datos correctos), el caso negativo (comportamiento ante datos inválidos o ausentes) y el caso masivo (bulk), donde se prueban operaciones sobre varios registros para verificar que no se superan los Governor Limits en condiciones reales de carga.

- **Aserciones explícitas**: cada test incluye el uso de **Assert.areEqual(), Assert.isTrue(), Assert.isFalse()**, entre otros métodos, para verificar que el resultado del código es exactamente el esperado, y no únicamente que el código no lanza una excepción no controlada.

Capítulo 4 Metodología y resultados

El propósito de este capítulo es proporcionar una descripción detallada del proceso de desarrollo e implementación de la solución CRM diseñada específicamente para IT Solutions, utilizando la plataforma Salesforce como base. En las siguientes secciones, se desglosará cada etapa del ciclo de vida del proyecto, comenzando por la definición de requisitos, continuando con el diseño arquitectónico del sistema, la implementación práctica y, finalmente, se detallarán las diversas pruebas y validaciones realizadas.

4.1.- PLANIFICACIÓN DEL PROYECTO

Para el desarrollo del proyecto se ha adoptado un modelo iterativo incremental, una metodología que se adapta de forma natural a las características de la plataforma Salesforce y a las necesidades de IT Solutions.

4.1.1.- Modelo de desarrollo iterativo incremental

El modelo iterativo incremental [71] estructura el desarrollo en ciclos sucesivos, donde cada iteración produce un incremento funcional que se añade al sistema existente. Este modelo resulta adecuado para el presente proyecto por las siguientes razones:

- **Entrega gradual de valor:** IT Solutions necesita disponer del módulo de ventas lo antes posible para comenzar a registrar oportunidades comerciales mientras se desarrollan los módulos restantes. El modelo incremental permite que cada iteración produzca un entregable funcional desplegable de forma independiente.

- **Adaptabilidad:** Cada iteración se cierra con una validación funcional que permite ajustar los requisitos de las iteraciones posteriores en función de lo observado. Esto resulta especialmente relevante en un proyecto con múltiples módulos interdependientes (ventas, soporte, portal, automatizaciones), donde las decisiones de diseño de una iteración condicionan las siguientes.

- **Gestión de riesgos:** Al aislar cada módulo en su propia iteración, los riesgos técnicos, como la integración con APIs externas de seguimiento de envíos, se abordan de forma localizada sin comprometer la funcionalidad ya entregada.

- **Alineación con la plataforma:** Salesforce distingue entre configuración declarativa (campos, flujos, reglas de validación) y desarrollo programático (Apex, LWC). El modelo incremental permite implementar primero la configuración declarativa de cada módulo y añadir el código personalizado cuando se dispone de una base estable, sacando partido a las capacidades de despliegue continuo que ofrece la plataforma.

El proyecto se ha estructurado en cuatro iteraciones principales:

1. **Gestión de Ventas (Sales Cloud)**: Implementación del proceso comercial completo, incluyendo la gestión de Leads con asignación automática por producto de interés, Accounts y Contacts con campos personalizados para IT Solutions, el ciclo Opportunity → Quote → Order, el proceso de aprobación de descuentos superiores al 15% y el catálogo de 25 productos distribuidos en cuatro familias (Componentes, Periféricos, Software, Servicios a medida).

2. **Gestión de Soporte (Service Cloud)**: Configuración del sistema de atención al cliente, incluyendo la gestión de Cases con tres colas de soporte especializadas (hardware, software, facturación), Email-to-Case para la creación automática de

casos desde correo electrónico, la base de conocimiento con artículos de tipo FAQ y el campo fórmula **Resolution_Time__c** para medir tiempos de resolución.

3. **Portal de Clientes (Experience Cloud)**: Desarrollo del portal web autenticado donde los clientes pueden consultar sus pedidos y el estado de envío en tiempo real mediante los componentes LWC **orderTrackingList** y **orderTrackingDetail**, abrir y hacer seguimiento de casos de soporte y acceder a la base de conocimiento.

4. **Integraciones y Automatizaciones**: Implementación de Flows automatizados (gestión de inventario, notificaciones de envío, descuentos), el Screen Flow de creación de pedidos desde presupuesto, la integración con la API del proveedor de logística mediante Named Credentials y callouts Apex, el proceso batch de actualización de inventario (InventoryUpdateBatch) y el componente LWC productSearchAndAdd para la búsqueda y adición de productos a oportunidades.

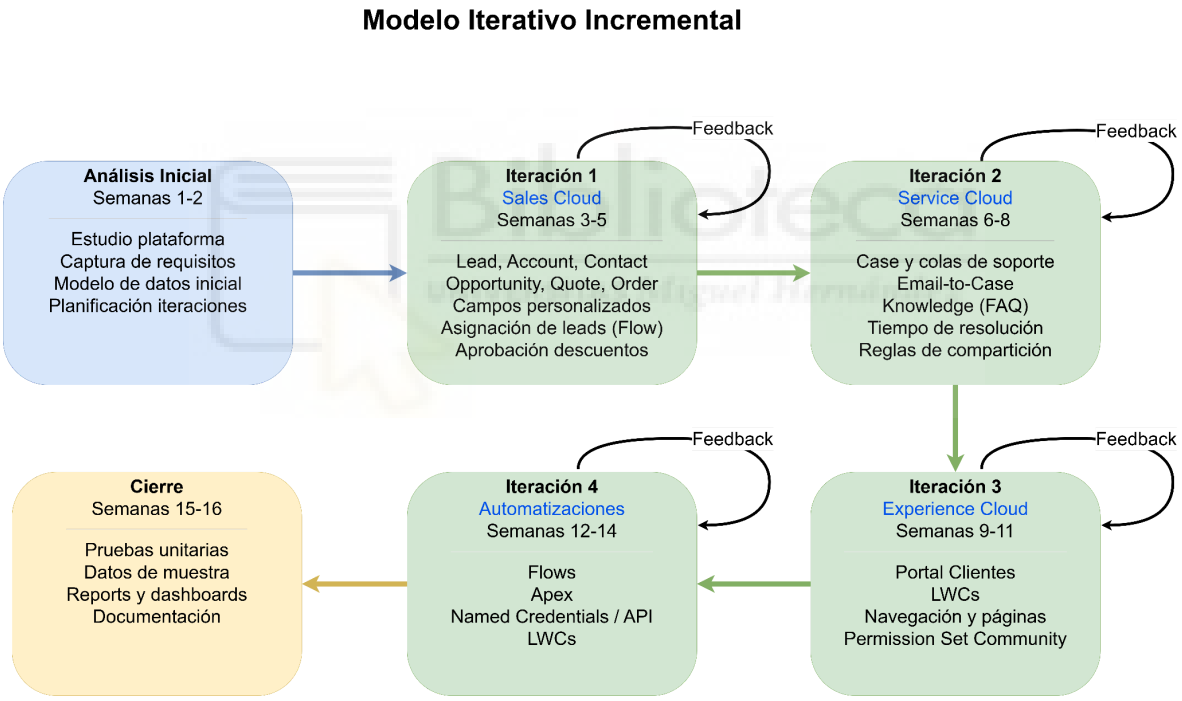

Figura 4.1: Diagrama del modelo iterativo incremental aplicado al proyecto

4.1.2.- Planificación temporal

El proyecto se ha desarrollado durante un período de 16 semanas. La siguiente tabla resume la distribución temporal por fases. Cada iteración incluye sus propias fases de análisis, diseño, implementación y pruebas, siguiendo el modelo iterativo descrito.

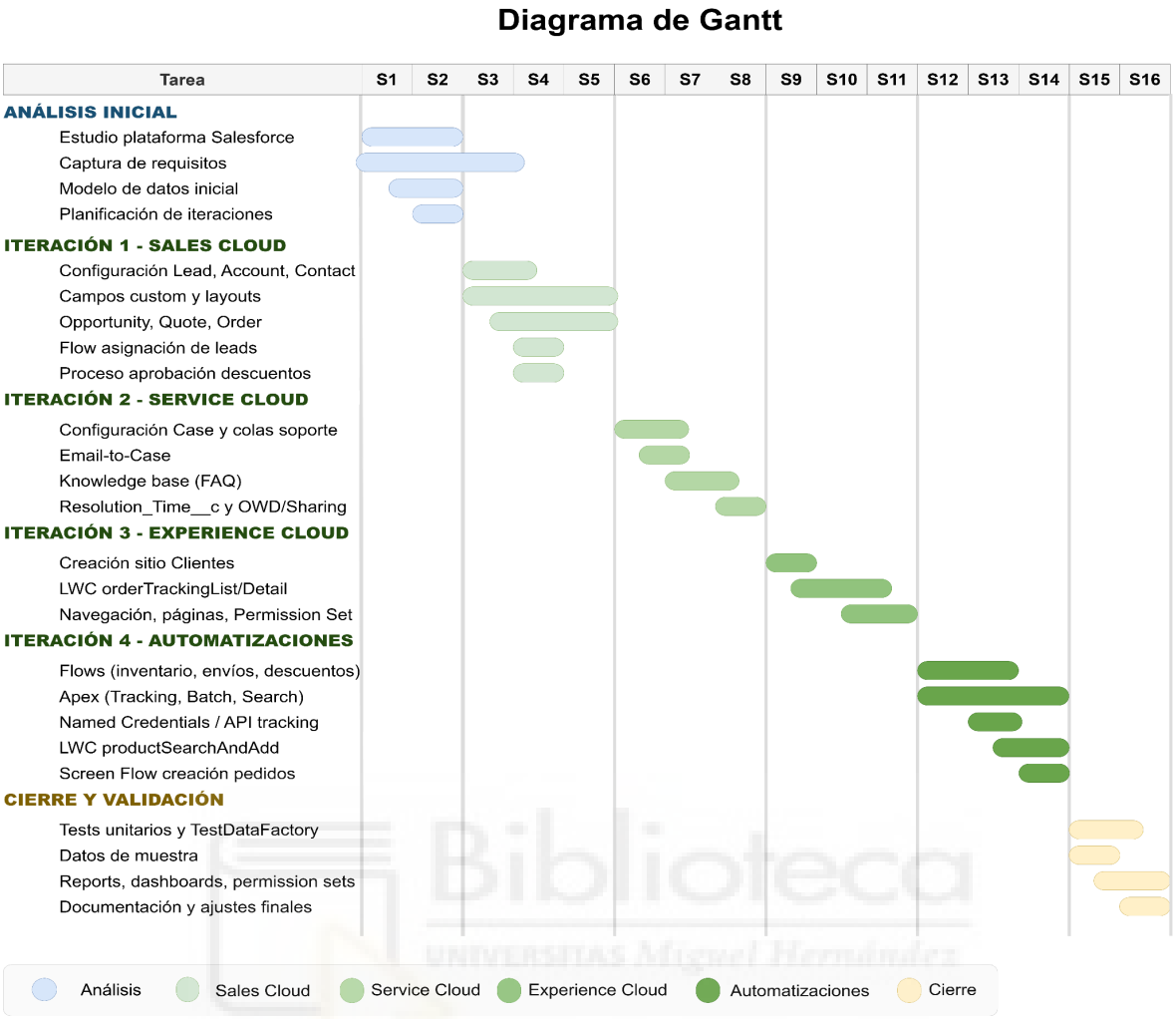

Figura 4.2: Diagrama de Gantt. Actividades del proyecto agrupado por fases

Tabla 4.1: Planificación del proyecto por fases e iteraciones

Semanas	Fase	Actividades principales
1-2	Análisis inicial	Estudio de la plataforma Salesforce, captura de requisitos, definición del modelo de datos inicial y planificación de iteraciones
3-5	Iteración 1 Sales Cloud	Objetos Lead, Account, Contact, Opportunity, Quote, Order; campos custom; Flow Asignación de Leads; proceso aprobación descuentos; catálogo de productos; validaciones
6-8	Iteración 2 Service Cloud	Objeto Case, colas de soporte, Email-to-Case, Knowledge base (FAQ)
9-11	Iteración 3 Experience Cloud	Portal de Clientes; LWC orderTrackingList y orderTrackingDetail; permission set Community; navegación y páginas del portal
12-14	Iteración 4 Automatizaciones	Record-Triggered Flows (inventario, envíos, descuentos); Screen Flow creación pedidos; Apex (ShipmentTrackingService, InventoryUpdateBatch, OrderTrackingController, ProductSearchController); Named Credentials; LWC productSearchAndAdd
15-16	Cierre y validación	Tests unitarios y TestDataFactory, datos de muestra, reports y dashboards, permission sets, documentación

4.2.- CAPTURA DE REQUISITOS

En este apartado se detallan los requisitos funcionales y no funcionales identificados para la solución, así como los roles de usuario y casos de uso que guían la implementación.

4.2.1.- Requisitos funcionales (RF) y no funcionales (RNF)

A continuación se detallan los requisitos funcionales principales, organizados por área funcional:

- **RF-01 - Gestión de Leads**. El sistema permite registrar clientes potenciales con sus datos de contacto, empresa e interés de producto. Los comerciales pueden crear Leads manualmente, y al guardarlos el sistema los asigna automáticamente a la cola de soporte correspondiente según el producto de interés. Los Leads cualificados se pueden convertir en Cuenta, Contacto y Oportunidad mediante la funcionalidad estándar de Salesforce.

- **RF-02 - Gestión de Oportunidades**. El sistema gestiona el ciclo completo de una oportunidad de venta, desde la prospección hasta el cierre. Para cerrar una oportunidad como ganada, debe existir un presupuesto sincronizado con estado ***Accepted***.

- **RF-03 - Catálogo de Productos**. El sistema mantiene un catálogo de productos organizado en cuatro familias: Componentes, Periféricos, Software y Servicios a medida. Los comerciales pueden buscar y añadir productos a oportunidades indicando cantidad y descuento por línea mediante un componente de la página.

- **RF-04 - Generación de presupuestos y pedidos**. El sistema permite generar presupuestos (Quotes) desde oportunidades, incluyendo productos, cantidades, precios y descuentos. Un presupuesto con estado ***Accepted*** se convierte en pedido (Order) mediante la Quick Action "New Order", que lanza un Screenflow con pantalla de confirmación y genera automáticamente el Order con sus OrderItems. Tras la creación exitosa, el sistema envía un email de confirmación con el número de pedido.

- **RF-05 - Gestión de Casos de Soporte**. El sistema registra y gestiona las incidencias y consultas de los clientes, que pueden crearse manualmente o de forma automática mediante Email-to-Case. Los casos se distribuyen entre tres colas especializadas: Hardware, Software y Billing.

- **RF-06 - Base de Conocimiento**. El sistema mantiene una base de conocimiento con artículos que los técnicos pueden crear y publicar. Los artículos son accesibles para los clientes desde el portal de Experience Cloud

- **RF-07 - Portal de Clientes**. El sistema proporciona un portal web para clientes donde pueden consultar su lista de pedidos con estado, fecha e importe, ver el detalle de un pedido con el estado de envío en tiempo real, abrir y hacer seguimiento de sus casos de soporte, y consultar artículos de la base de conocimiento.

- **RF-08 - Automatización de Procesos**. El sistema automatiza tareas repetitivas mediante Record-Triggered Flows y Apex: la asignación de Leads a colas según producto de interés, la aprobación de descuentos, la actualización automática de inventario, y la consulta a la API de tracking.

- **RF-09 - Reportes y Dashboards**. El sistema proporciona paneles visuales con métricas. El Dashboard de Ventas y El Dashboard de Soporte muestra gráficas y tablas con información relevante sobre oportunidades cerradas, número de casos por agente o tiempo medio de resolución en horas.

- **RF-10 - Integración con API de Seguimiento de Envíos**. El sistema se integra con una API externa para consultar el estado de los envíos. Al crear un registro de tracking, una automatización realiza una petición para actualizar el estado y la fecha estimada de entrega. Los clientes pueden consultar este estado en tiempo real desde el portal, con refresco automático si la última actualización supera una hora.

- **RF-11 - Gestión de Inventario**. El sistema gestiona el stock de productos mediante el objeto personalizado **Inventory__c**, vinculado al estándar Product2, que almacena el stock disponible, el reservado y el punto de reposición. Al crear un pedido, un Flow descuenta automáticamente unidades del stock disponible. Una validación impide que el stock disponible sea negativo y se generan alertas (Tasks) cuando el stock cae por debajo del punto de reposición.

- **RF-12 - Historial Visual de Oportunidades**. El sistema muestra una línea temporal visual con los cambios de etapa de cada oportunidad mediante un componente LWC en la página de detalle de la Oportunidad. Los cambios se presentan ordenados cronológicamente con fecha formateada, icono específico por etapa e indicadores visuales de progreso.

Además de los requisitos funcionales (lo que la aplicación debe hacer), para una completa descripción de "*qué se espera*" de la implementación final, es necesario describir los

requisitos no funcionales (cómo lo debe hacer). A continuación se describen dichos requisitos no funcionales:

- **RNF-01 - Rendimiento**. Todas las consultas SOQL y operaciones DML respetan los governor limits de Salesforce. Los procesos batch procesan los registros en lotes de 200 unidades para optimizar el uso de recursos.

- **RNF-02 - Seguridad**. Los usuarios externos solo acceden a sus propios datos. Las credenciales de la API de tracking se gestionan mediante Named Credentials, evitando exponer tokens en el código, y todas las clases Apex que acceden a datos sensibles se ejecutan bajo los permisos del usuario que las invoca.

- **RNF-03 - Disponibilidad**. La plataforma Salesforce garantiza un uptime mínimo del 99,9% mediante su infraestructura cloud multi-tenant, sin requerir gestión de infraestructura propia.

- **RNF-04 - Mantenibilidad**. El código Apex sigue el patrón de separación entre lógica de negocio y clases de test, con una cobertura superior al 75% (requisito de Salesforce para producción). Se utiliza una clase **TestDataFactory** para la creación de datos de prueba reutilizables.

- **RNF-05 - Integridad de datos**. El sistema garantiza la coherencia de los datos mediante Validation Rules que impiden valores inválidos en campos críticos (como stock negativo) y mediante validaciones en triggers, que refuerza a nivel de código la lógica de negocio de IT Solutions.

4.2.2.- Roles de usuario

Se han identificado los siguientes perfiles de usuario dentro de IT Solutions, cada uno con necesidades y permisos diferenciados dentro del sistema:

Tabla 4.2: Perfil: Comercial

Rol	Comercial
Descripción	Empleado encargado de la captación de clientes y gestión de oportunidades de venta.
Responsabilidades	- Registro y cualificación de Leads - Gestión de Oportunidades de venta - Generación de Presupuestos (Quotes) - Consulta del catálogo de productos
Permisos	- Lectura y edición de Leads, Oportunidades, Presupuestos, Cuentas y Contactos - Lectura del catálogo de productos

Tabla 4.3: Perfil: Responsable de Ventas

Rol	Responsable de Ventas
Descripción	Supervisor del equipo comercial, encargado de analizar el rendimiento y aprobar descuentos
Responsabilidades	- Supervisión de la pipeline de ventas - Aprobación de descuentos superiores al 15% - Análisis de reportes y métricas de ventas
Permisos	- Los mismos que el Comercial - Aprobación de descuentos - Acceso a todos los registros del equipo comercial

Tabla 4.4: Perfil: Técnico de Soporte

Rol	Técnico de Soporte
Descripción	Empleado encargado de resolver incidencias y consultas de clientes
Responsabilidades	- Gestión de Casos de soporte - Comunicación con clientes para resolución de incidencias - Consulta y creación de artículos de conocimiento
Permisos	- Lectura y edición de Casos, Cuentas y Contactos - Lectura y edición de artículos de conocimiento (Knowledge)

Tabla 4.5: Perfil: Administrador del Sistema

Rol	Administrador del Sistema
Descripción	Usuario con conocimientos técnicos encargado de la configuración y mantenimiento de la plataforma
Responsabilidades	- Gestión de usuarios y permisos - Configuración de automatizaciones (Flow, Apex) - Creación de reportes y dashboards - Mantenimiento general del sistema
Permisos	- Acceso completo a todos los objetos y funcionalidades de la plataforma

Tabla 4.6: Perfil: Cliente Portal (Usuario externo)

Rol	Cliente Portal (Usuario externo)
Descripción	Cliente de IT Solutions que accede al portal web (Experience Cloud) para consultar información y gestionar incidencias
Responsabilidades	- Consulta de pedidos propios y estado de envío - Apertura y seguimiento de casos de soporte - Consulta de artículos de la base de conocimiento
Permisos	- Acceso limitado a sus propios registros (Pedidos, Casos) - Lectura de artículos públicos de conocimiento

4.2.3.- Casos de uso

A continuación se presenta el diagrama de los casos de uso del sistema, que guían el diseño y la implementación. La descripción completa de cada uno de ellos se encuentra detallada en las tablas del Anexo I.

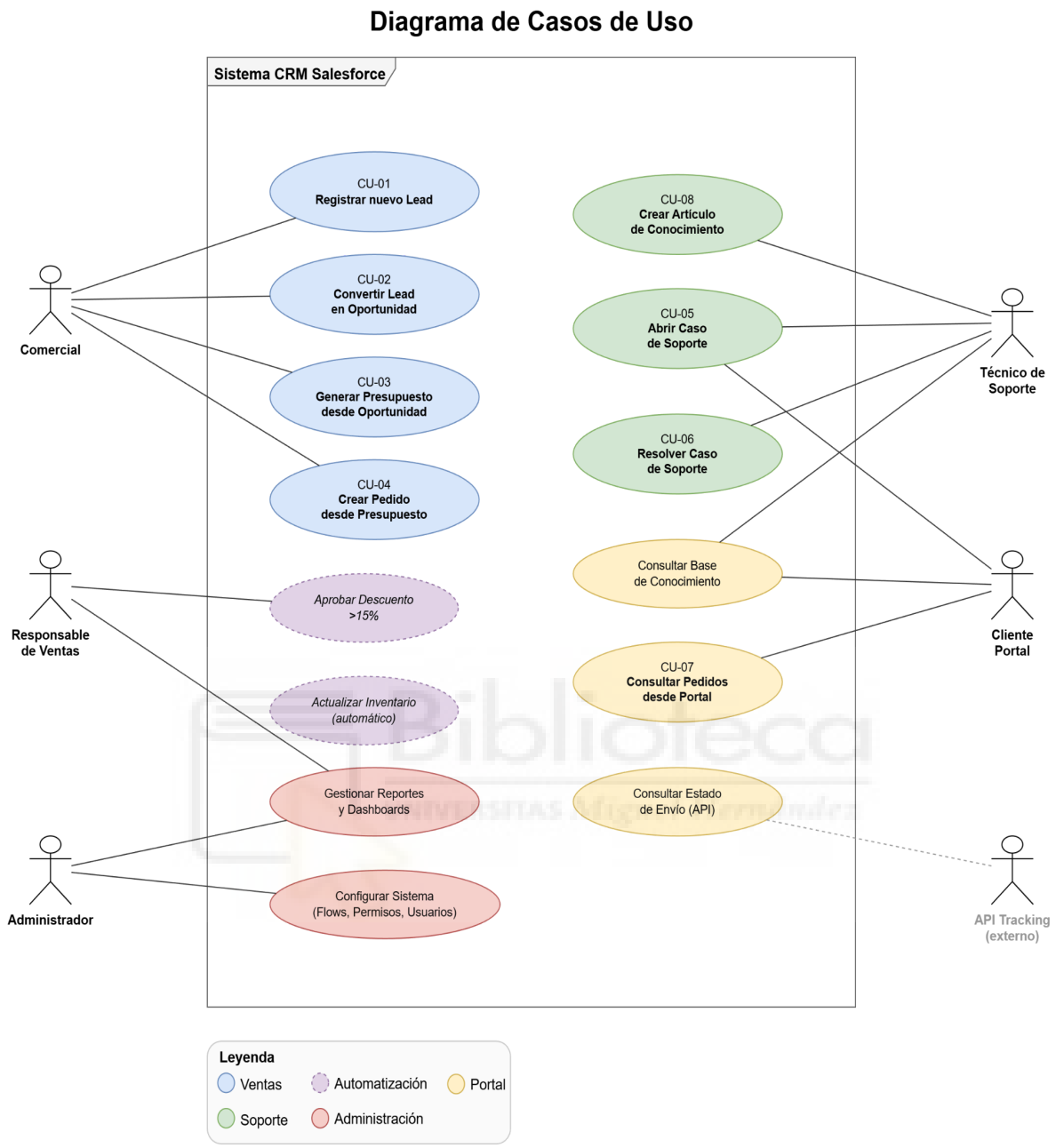

Figura 4.3: Diagrama de casos de uso mostrando las relaciones entre los cinco actores del sistema y los casos de uso principales

4.3.- DISEÑO

En este apartado se detalla el diseño de la solución implementada para IT Solutions, incluyendo el modelo de datos, la arquitectura de componentes, el diagrama de clases, los diagramas de secuencia de los procesos más relevantes y los diagramas de estado de los objetos con ciclo de vida propio.

4.3.1.- Modelo de datos

El modelo de datos de la solución se basa en los objetos estándar de Salesforce, complementados con dos objetos personalizados y campos adicionales en objetos estándar para cubrir las necesidades específicas del negocio de IT Solutions.

Comenzando con los objetos estándar, se han implementado los siguientes junto a sus campos personalizados:

- **Lead:** Clientes potenciales aún no cualificados:
  - **ProductInterest__c:** Ayuda a las automatizaciones a enrutar el Lead a la cola correspondiente
  - **Lead_Source_Detail__c:** Aportar más sobre el origen del Lead. El estándar de Salesforce es un desplegable con opciones.
- **Account:** Empresas o particulares clientes de IT Solutions:
  - **Customer_Type__c:** Clasificar el tipo de cuenta para filtrar informes.
  - **Preferred_Payment_Method__c:** Registrar el método de pago habitual del cliente para agilizar la gestión de pedidos y facturación.
- **Contact:** Personas de contacto asociadas a una cuenta.
- **Opportunity:** Oportunidades de venta en curso:
  - **Discount_Percentage__c:** Capturar el descuento negociado.
  - **Requires_Approval__c:** Marcar automáticamente las oportunidades cuyo descuento supera el 15%.
  - **Approval_Status__c:** Registrar el estado del proceso de aprobación para filtrar informes.
- **Product2:** Catálogo de productos de IT Solutions.
- **PricebookEntry:** Precio de un producto en una lista de precios.
- **OpportunityLineItem:** Productos asociados a una oportunidad.
- **Quote:** Presupuestos generados desde oportunidades.

- **QuoteLineItem:** Líneas de productos en un presupuesto.

- **Order:** Pedidos confirmados generados desde presupuestos:
    - **Payment_Status__c:** Controlar si el pedido ha sido pagado, pendiente o rechazado sin depender de integraciones externas de facturación.

    - **Shipping_Status__c:** Exponer el estado del envío directamente en el pedido para que el portal de clientes y los agentes lo consulten.

- **OrderItem:** Líneas de producto de un pedido.

- **Case:** Casos de soporte técnico:
    - **Related_Order__c:** Vincular un caso de soporte a un pedido específico para que los agentes tengan contexto completo del problema sin búsquedas manuales.

    - **Resolution_Time__c:** Calcular automáticamente las horas entre apertura y cierre del caso para la métrica de tiempo medio de resolución en el Dashboard de Soporte.

- **Knowledge__kav:** Artículos de la base de conocimiento.

4.3.1.1.- Objetos personalizados creados

A pesar de que la plataforma ofrece una generosa cantidad de funcionalidad *out-of-the-box***,** carece de funcionalidades características de un sistema ERP, tales como la gestión de inventario o el seguimiento de pedidos. Para solventar esto, se han creado dos objetos personalizados.

Para la gestión de inventario, se ha optado por un objeto **Inventory__c** que gestiona la cantidad de stock, el umbral bajo el que el sistema debe alertar por bajo stock y a qué producto está relacionado este inventario.

Sobre el seguimiento de envío, se ha implementado un objeto **Shipment_Tracking__c** junto a una integración con una API Externa. Los registros de **Shipment_Tracking__c** se relacionan con un pedido y guarda información como el número de seguimiento, la empresa transportista y el estado.

El estado del pedido se actualiza automáticamente invocando a la API cuando un usuario accede a un pedido cuyo seguimiento se actualizó hace más de una hora.

*Tabla 4.7: Campos personalizados del objeto **Inventory__c***

Campo	Tipo de dato	Descripción
Product__c	Lookup(Product2)	Producto asociado
Available_Stock__c	Number	Unidades disponibles en almacén
Reserved_Stock__c	Number	Unidades reservadas en pedidos activos
Reorder_Point__c	Number	Umbral mínimo que activa alerta de reposición
Last_Updated__c	DateTime	Fecha y hora de la última actualización

*Tabla 4.8: Campos personalizados del objeto **Shipment_Tracking__c***

Campo	Tipo de dato	Descripción
Order__c	Master-Detail(Order)	Pedido al que pertenece el envío
Tracking_Number__c	Text	Número de seguimiento del transportista
Carrier__c	Picklist	Transportista: SEUR, MRW, Correos, DHL, UPS
Status__c	Picklist	Estado del envío (En tránsito, Entregado, Incidencia, Pendiente de recogida)
Estimated_Delivery__c	Date	Fecha estimada de entrega
Last_API_Update__c	DateTime	Última consulta a la API de tracking

Tabla 4.9: Relaciones principales del modelo de datos

Relación	Tipo de relación	Cardinalidad
Lead → Account / Contact / Opportunity	Conversión	1:1 (proceso de conversión estándar de Salesforce)
Account → Contact	Lookup	1:N
Account → Opportunity	Lookup	1:N
Account → Order	Lookup	1:N
Account → Case	Lookup	1:N
Opportunity → Quote	Lookup	1:N
Quote → QuoteLineItem	Master-Detail	1:N
Quote → Order	Lookup	1:1 (un presupuesto aceptado genera un pedido)
Order → OrderItem	Master-Detail	1:N
Order → Shipment_Tracking__c	Master-Detail	1:N
Order → Case (via Related_Order__c)	Lookup	1:N
Product2 → PricebookEntry	Lookup	1:N
Product2 → Inventory__c	Lookup	1:1
Contact → Case	Lookup	1:N

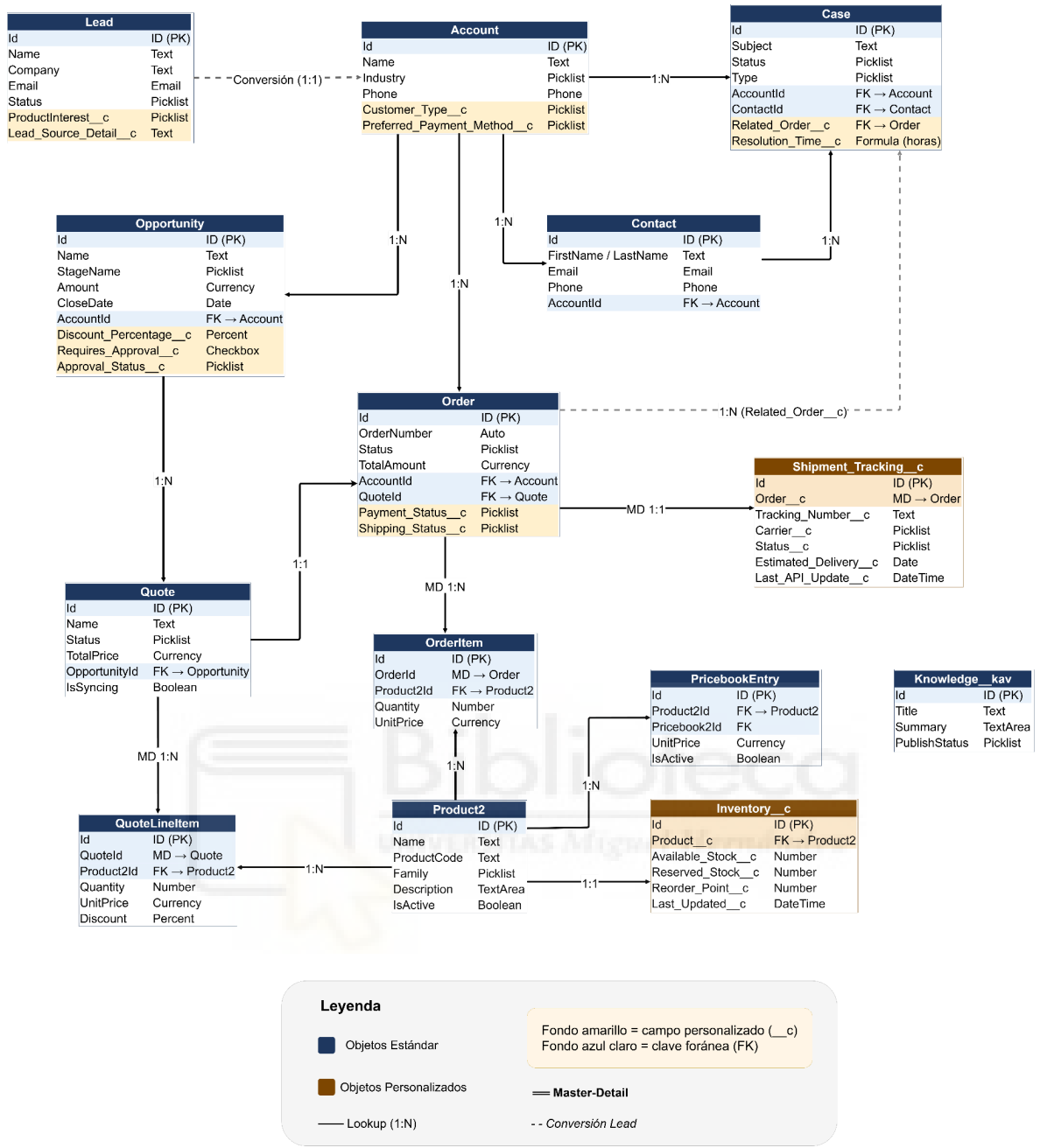

Figura 4.4: Diagrama Entidad-Relación del modelo de datos, mostrando objetos estándar (azul), objetos personalizados (naranja) y sus relaciones mediante Lookup y Master-Detail

4.3.2.- Diagrama de clases

La lógica de negocio en forma de código de la solución se implementa en Apex siguiendo el paradigma orientado a objetos. El proyecto cuenta con seis clases principales, un trigger y una clase auxiliar de pruebas, cada una con una responsabilidad concreta dentro del sistema.

1. **OpportunityTriggerHandler**: contiene la lógica del ciclo de vida de las oportunidades. Es invocado por OpportunityTrigger en el evento before update y valida que, antes de marcar una oportunidad como "Cerrada Ganada", exista un presupuesto sincronizado con estado "Aceptado".

2. **OpportunityTimelineController**: expone datos de historial de etapas (OpportunityHistory) al componente LWC opportunityTimeline, permitiendo visualizar la progresión temporal de una oportunidad.

3. **ProductSearchController**: permite buscar productos del catálogo y añadirlos directamente a una oportunidad como OpportunityLineItem, siendo usado por el LWC productSearchAndAdd.

4. **OrderTrackingController**: sirve al portal de clientes (Experience Cloud). Devuelve los pedidos del cliente autenticado y la información de tracking asociada, delegando en ShipmentTrackingService cuando los datos de envío están desactualizados (más de una hora sin actualizar).

5. **ShipmentTrackingService**: encapsula la integración con la API externa de seguimiento de envíos. Utiliza una Named Credential (TrackingProvider) para realizar llamadas HTTP asíncronas al endpoint de la integración **/v1/shipments/{trackingNumber}**, actualizando el estado y la fecha estimada de entrega en el objeto Shipment_Tracking__c.

6. **InventoryUpdateBatch**: es un proceso batch programado que recalcula nocturnamente el stock reservado de cada producto a partir de los pedidos activos, y genera tareas de reposición para los productos que caen por debajo del punto de pedido (Reorder_Point__c).

7. **TestDataFactory**: proporciona métodos de creación de datos de prueba reutilizables para todos los test classes del proyecto, garantizando consistencia y eliminando duplicación de código de test

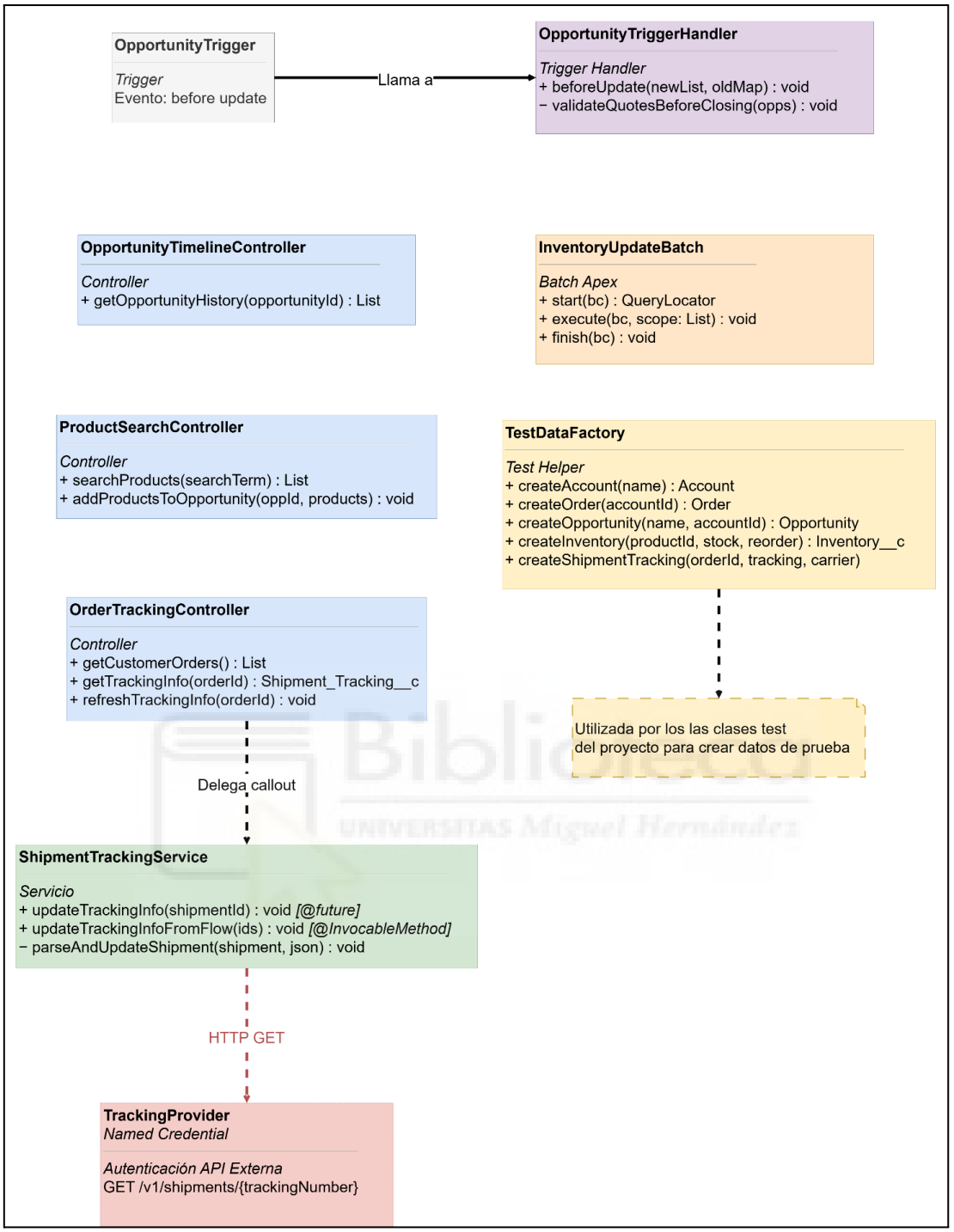

Figura 4.5: Diagrama de clases mostrando las seis clases Apex, el trigger, sus métodos principales, relaciones de dependencia y la integración con la API externa

4.3.3.- Arquitectura de la solución

La solución sigue una arquitectura de tres capas que aprovecha el modelo de plataforma de Salesforce, separando claramente la presentación, la lógica de negocio y los datos.

4.3.3.1.- Capa de presentación

Agrupa las interfaces de usuario tanto para usuarios internos como externos.

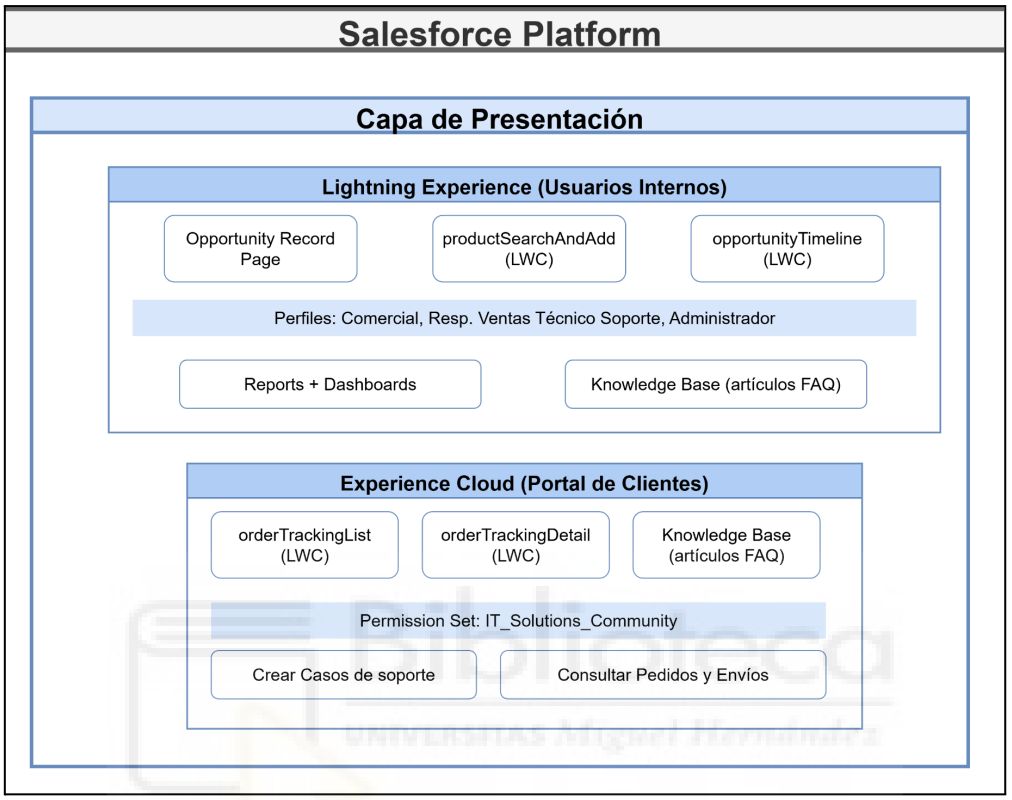

Figura 4.6: Diagrama de arquitectura de tres capas. Capa de Presentación.

Los usuarios internos (comerciales, responsables de ventas, técnicos de soporte y administradores) acceden a través de **Lightning Experience**, donde se ha configurado una aplicación con las tablas/objetos a los que los usuarios accederán (Lead, Opportunity, Inventory, etc), páginas de registros personalizadas mediante App Builder y componentes LWC a medida, así como informes y dashboards con métricas relevantes para los distintos roles de la aplicación.

Los clientes acceden mediante un site de Experience Cloud que expone los componentes **orderTrackingList** y **orderTrackingDetail** para la consulta de pedidos y envíos, junto con la base de conocimiento.

4.3.3.2.- Capa de lógica de negocio

Esta capa combina automatizaciones declarativas construidas con Flow Builder y código Apex para los casos que requieren mayor control de la lógica.

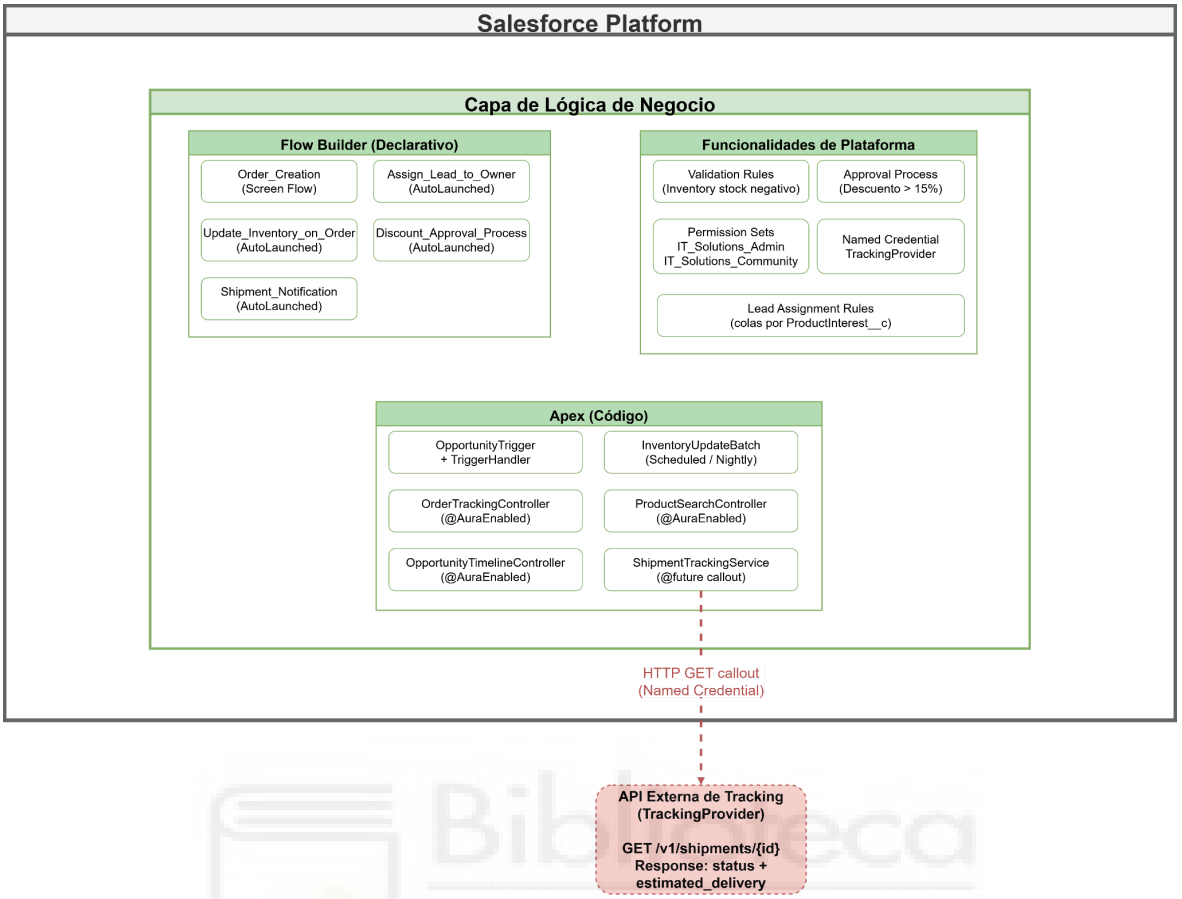

Figura 4.7: Diagrama de arquitectura de tres capas. Capa de Lógica de Negocio.

Los flows cubren la creación de pedidos (**Order_Creation**), la asignación de leads a colas por categoría de producto (**Assign_Lead_to_Owner**), la actualización de inventario al crear pedidos (Update_Inventory_on_Order), el proceso de aprobación de descuentos (**Discount_Approval_Process**) y las notificaciones de tracking (**Shipment_Notification**). El código Apex se reserva para lógica que requiere programación: validaciones en trigger, callouts HTTP asíncronos y procesos batch.

4.3.3.3.- Capa de datos

Almacena todos los registros en la base de datos multitenant de Salesforce.

Los controles de seguridad se aplican a nivel de objetos, campos, de perfil y de permission set. La integración con el sistema externo de tracking se realiza mediante callout HTTP desde **ShipmentTrackingService** usando la Named Credential **TrackingProvider**, lo que mantiene la URL del endpoint y las credenciales de autenticación fuera del código fuente.

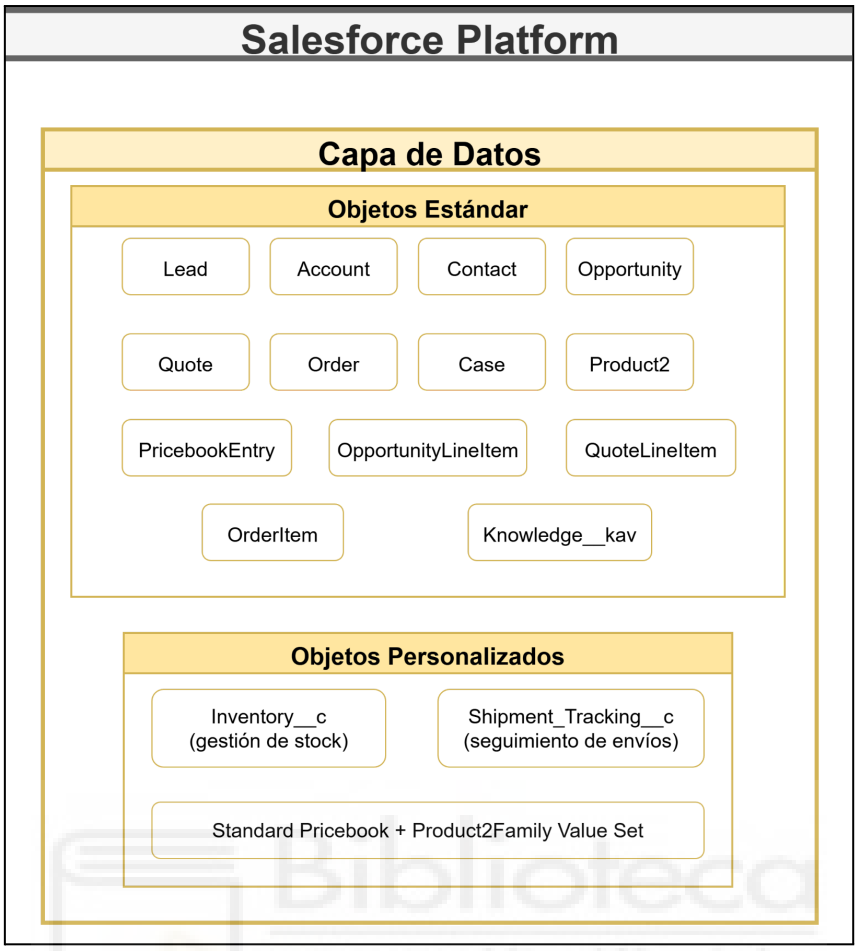

Figura 4.8: Diagrama de arquitectura de tres capas. Capa de Datos.

4.3.4.- Diagramas de secuencia

A continuación se presentan los diagramas de secuencia de los dos procesos más complejos del proyecto.

4.3.4.1.- Generación de Pedido desde Oportunidad

Este proceso se desencadena cuando un comercial intenta marcar una oportunidad como "Cerrada Ganada". El flujo implica varios actores del sistema y garantiza la consistencia entre oportunidad, presupuesto, pedido e inventario.

En primer lugar, el trigger **OpportunityTrigger** intercepta el cambio de etapa e invoca **OpportunityTriggerHandler**, que verifica que exista un presupuesto sincronizado con estado "***Aceptado***". Si la validación falla, se muestra un error al usuario y el proceso se detiene. Si pasa, la oportunidad se cierra y el comercial lanza manualmente el flow **Order_Creation** desde la Quick Action "***New Order***" disponible en la página de registro del presupuesto.

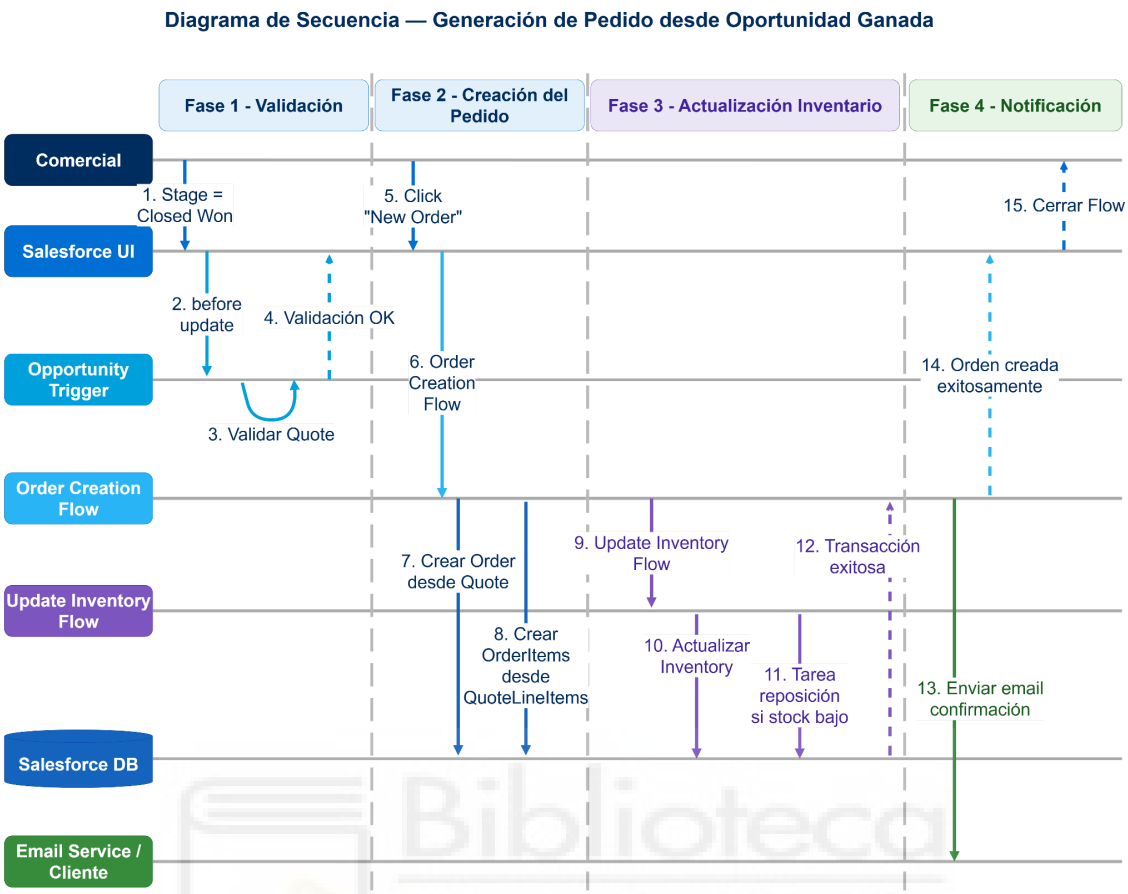

Figura 4.9: Diagrama de secuencia del proceso de generación de pedido desde una oportunidad ganada, mostrando las interacciones entre el comercial y el sistema

El flow recupera las líneas del presupuesto (**QuoteLineItem**), muestra una pantalla de confirmación al usuario con los productos, y crea el registro de Order junto con los OrderItem correspondientes. A continuación, el flow **Update_Inventory_on_Order** decrementa el stock disponible (**Available_Stock__c**) en **Inventory__c** y crea una tarea de reposición si algún producto cae por debajo de su punto de pedido. Finalmente, se envía un email de confirmación al usuario que generó el pedido.

4.3.4.2.- Consulta de Estado de Envío desde el portal de cliente

Este proceso se produce cuando un cliente autenticado en el portal de Experience Cloud consulta el estado de uno de sus pedidos.

El componente LWC **orderTrackingList** carga la lista de pedidos del cliente mediante **getCustomerOrders()**, que filtra por el AccountId del contacto autenticado. Al seleccionar un pedido, **orderTrackingDetail** invoca **getTrackingInfo()** para recuperar el registro **Shipment_Tracking__c** asociado.

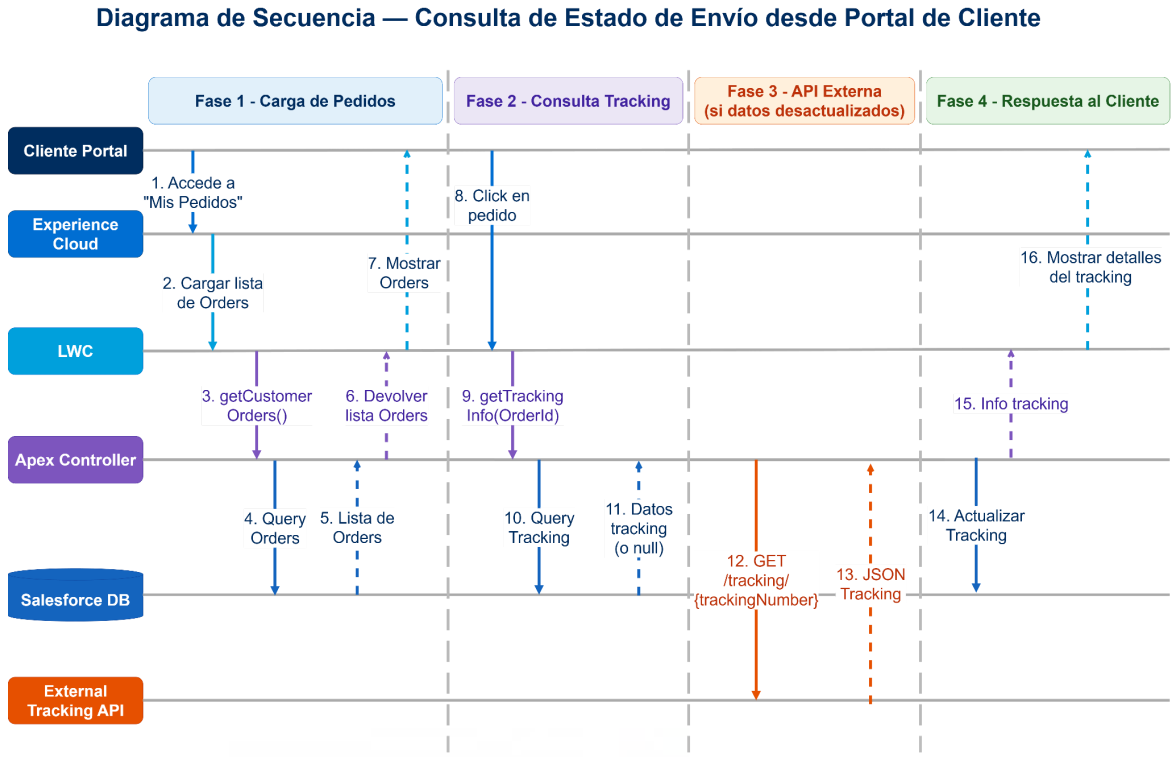

Figura 4.10: Diagrama de secuencia de la consulta de tracking desde el portal de cliente, mostrando las interacciones entre el cliente y el sistema

Si el campo **Last_API_Update__c** está vacío o han pasado más de 60 minutos desde la última consulta, el controlador llama a **refreshTrackingInfo()**, que delega en **ShipmentTrackingService.updateTrackingInfo()** para realizar un callout HTTP asíncrono al endpoint de la API externa.

La respuesta JSON actualiza los campos **Status__c** y **Estimated_Delivery__c** en el registro de tracking, y el componente refresca la vista con la información actualizada.

4.3.5.- Diagramas de estado

Los siguientes diagramas de estado representan el ciclo de vida de dos de los principales flujos de IT Solutions: Oportunidades y Envíos.

4.3.5.1- Ciclo de vida de la Oportunidad

Las oportunidades siguen un ciclo de vida predefinido por las etapas de ventas configuradas en Salesforce.

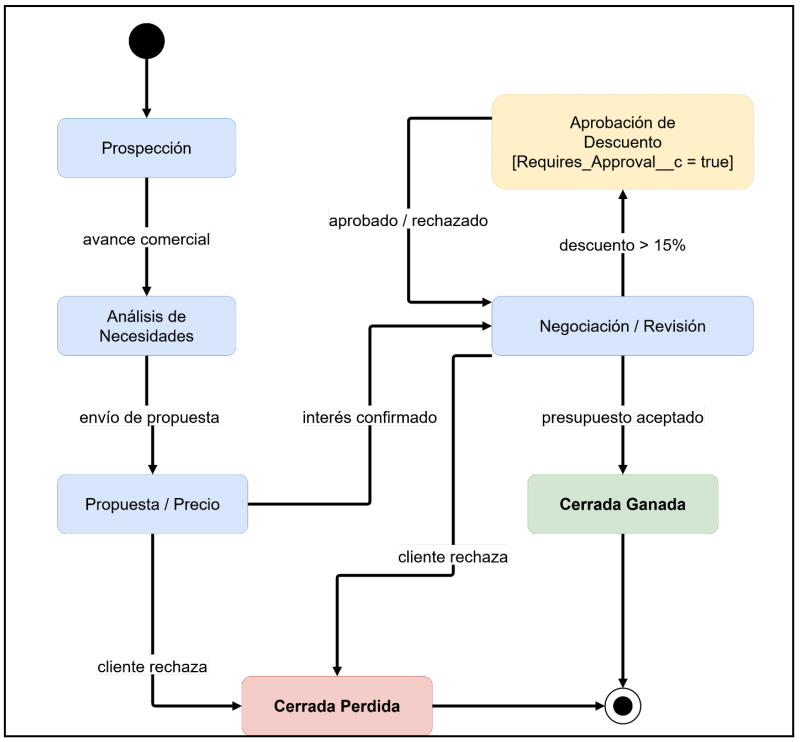

Figura 4.11: Diagrama de estado del ciclo de vida de una Oportunidad

La transición entre etapas refleja el avance del proceso comercial y tiene implicaciones en automatizaciones y validaciones:

- **Prospección:** Etapa inicial. El lead ha sido convertido y se ha identificado una posible oportunidad de venta.

- **Análisis de Necesidades:** Se evalúan los requerimientos técnicos y comerciales del cliente.

- **Propuesta/Precio:** Se elabora y envía el presupuesto al cliente mediante un Quote.

- **Negociación/Revisión:** El cliente revisa la propuesta. Si el descuento supera el 15%, el Flow **Discount_Approval_Process** activa automáticamente el campo **Requires_Approval__c** y establece **Approval_Status__c** = ***Pendiente***, bloqueando el avance hasta recibir aprobación interna.

- **Cerrada Ganada:** Estado final positivo. La transición a este estado está validada por **OpportunityTriggerHandler**: es obligatorio que exista un presupuesto sincronizado (**IsSyncing** = ***true***) con **Status** = '***Accepted***'. De lo contrario, Salesforce bloquea el cambio con un error de validación.

- **Cerrada Perdida:** Estado final negativo. No tiene restricciones adicionales.

4.3.5.2- Ciclo de vida del Envío

El objeto **Shipment_Tracking__c** registra el estado del envío físico de un pedido.

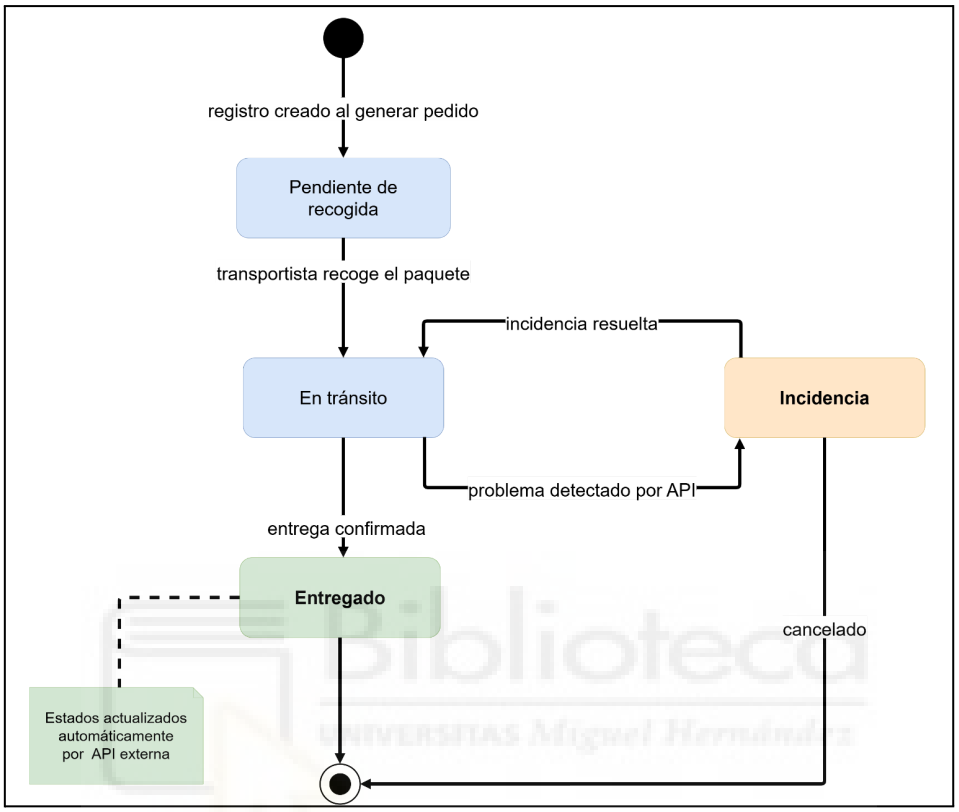

Figura 4.12: Diagrama de estado del ciclo de vida de un Envío

Sus estados reflejan la información recibida desde la API externa de tracking:

- **Pendiente de recogida:** El registro se crea al generar el pedido, antes de que el transportista recoja el paquete.
- **En tránsito:** Estado por defecto una vez el transportista confirma la recogida.
- **Entregado**: Estado final positivo. La API confirma la entrega al destinatario.
- **Incidencia:** La API reporta un problema (dirección incorrecta, intento fallido, daño). Desde aquí puede reactivarse como "En tránsito" si la incidencia se resuelve.

Las transiciones entre estados son gestionadas por **ShipmentTrackingService**, que interpreta la respuesta JSON de la API y actualiza el campo **Status__c** junto con **Estimated_Delivery__c** y **Last_API_Update__c**.

4.4.- IMPLEMENTACIÓN

Este apartado presenta las pantallas principales de la aplicación implementada y detalla los fragmentos de código más relevantes del proyecto, explicando las decisiones técnicas tomadas y los problemas resueltos durante el desarrollo.

4.4.1.- Página de Oportunidad

La figura 4.13 muestra la página de registro de una Oportunidad tal como la visualiza un comercial en Lightning Experience. La maquetación se configuró mediante App Builder, la herramienta estándar de Salesforce para construir páginas en forma de drag and drop, sustituyendo la página estándar por una personalizada para esta aplicación que distribuye los componentes en tres zonas diferenciadas.

En la parte superior, el panel de aspectos destacados (Highlights Panel) expone de un vistazo los cuatro campos más relevantes de la negociación: nombre de la cuenta vinculada, fecha de cierre prevista, importe y propietario del registro.

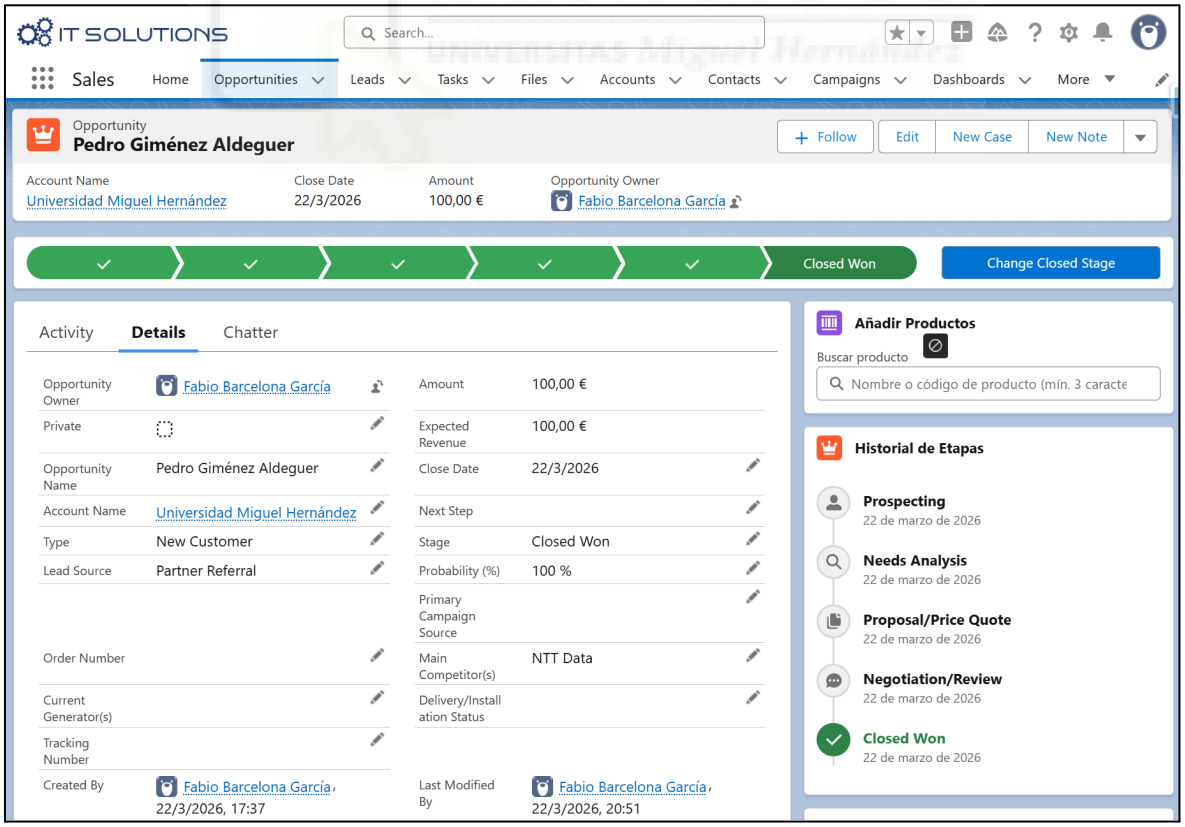

Figura 4.13: Página de detalle de Oportunidad en Lightning Experience

Inmediatamente debajo, la barra de ruta (Path Assistant) refleja el ciclo de vida completo de la Oportunidad, desde Prospecting hasta Closed Won; y permite al comercial avanzar de etapa con un clic.

El área principal recoge las pestañas Actividad, Detalles y Chatter. La pestaña Detalles agrupa todos los campos del objeto, mientras que Actividad centraliza las tareas, llamadas y correos asociados al registro. Por último, Chatter, un chat para usuarios internos a nivel de registros que ofrece la plataforma de forma nativa.

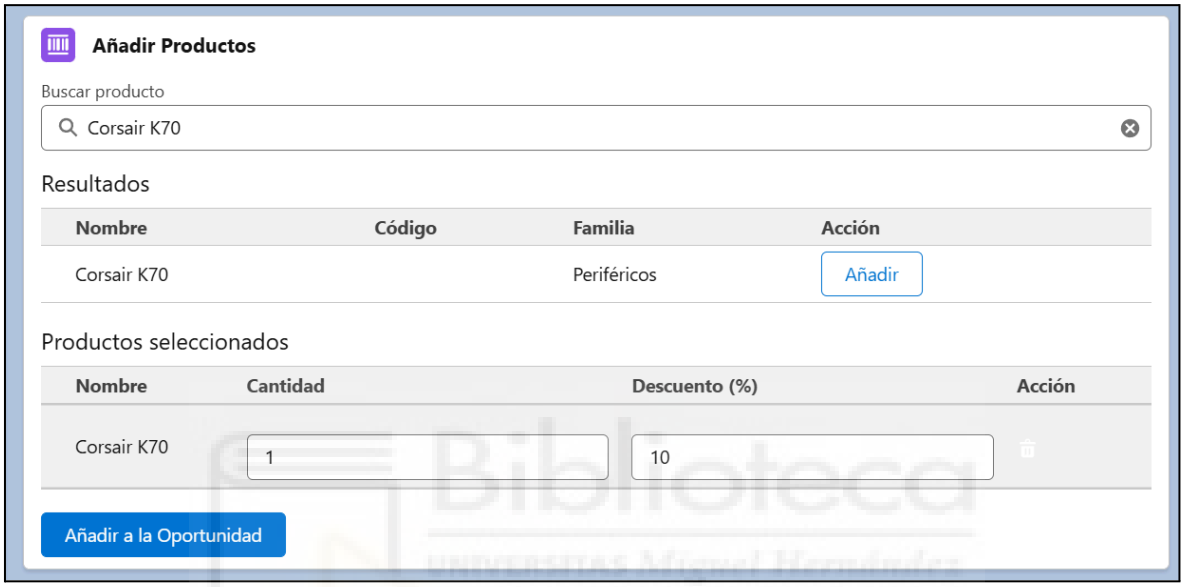

Figura 4.14: Vista detallada del componente Añadir Productos

La barra lateral derecha contiene los dos componentes LWC desarrollados a medida, visibles simultáneamente sin necesidad de desplazamiento:

- **Añadir Productos (productSearchAndAdd):** permite al comercial buscar artículos del catálogo por nombre o código, y añadirlos como Opportunity Products directamente desde la página de la Oportunidad, sin salir del registro ni navegar al objeto Product2. El componente realiza una llamada al controlador Apex **ProductSearchController**, que devuelve resultados paginados filtrados por disponibilidad en inventario.

- **Historial de Etapas (opportunityTimeline):** muestra una línea de tiempo vertical con cada cambio de etapa que ha sufrido la Oportunidad, ordenado cronológicamente. Cada entrada indica el nombre de la etapa, un icono identificativo y la fecha exacta del cambio. Los estados Closed Won y Closed Lost se resaltan visualmente en verde y rojo respectivamente mediante clases CSS condicionales aplicadas desde el controlador JavaScript. El componente consume el

controlador Apex **OpportunityTimelineController**, que consulta el historial de campo del objeto para reconstruir el historial completo de cambios de etapa.

En la parte inferior del registro, el panel Relacionados agrupa los objetos vinculados: pedidos (Orders), productos de la oportunidad (Products) y presupuestos (Quotes), permitiendo navegar a cualquiera de ellos sin abandonar el contexto de la negociación.

4.4.2.- Gestión de Casos y Colas de Soporte

El módulo de soporte de IT Solutions se articula en torno a tres colas de trabajo que segmentan las incidencias según su naturaleza:

1. **Hardware Support Queue**: recibe casos relacionados con componentes físicos y periféricos, con dirección de notificación hardware-support@itsolutions.com.

2. **Software Support Queue:** atiende incidencias de software y licencias, con notificación a software-support@itsolutions.com.

3. **Billing Queue:** gestiona consultas de facturación y pagos, con notificación a facturacion@itsolutions.com

Las tres colas están configuradas con Send Email To Members = true, de modo que cualquier caso recién asignado genera un aviso automático a los agentes de la cola correspondiente.

El objeto Case se amplió con dos campos personalizados: Related_Order__c, un campo de relación al objeto Order que vincula la incidencia con el pedido afectado, útil en casos de envío incorrecto o devolución, y Resolution_Time__c, un campo numérico que registra el tiempo de resolución en horas para el seguimiento del SLA.

Los casos llegan a las colas por dos vías diferentes: manualmente por los agentes, o de forma automática mediante el servicio Email-to-Case, una funcionalidad estándar de la plataforma que convierte los correos recibidos en las tres direcciones de soporte en registros Case sin intervención humana. La asignación a la cola correcta queda determinada por la dirección de destino del correo entrante.

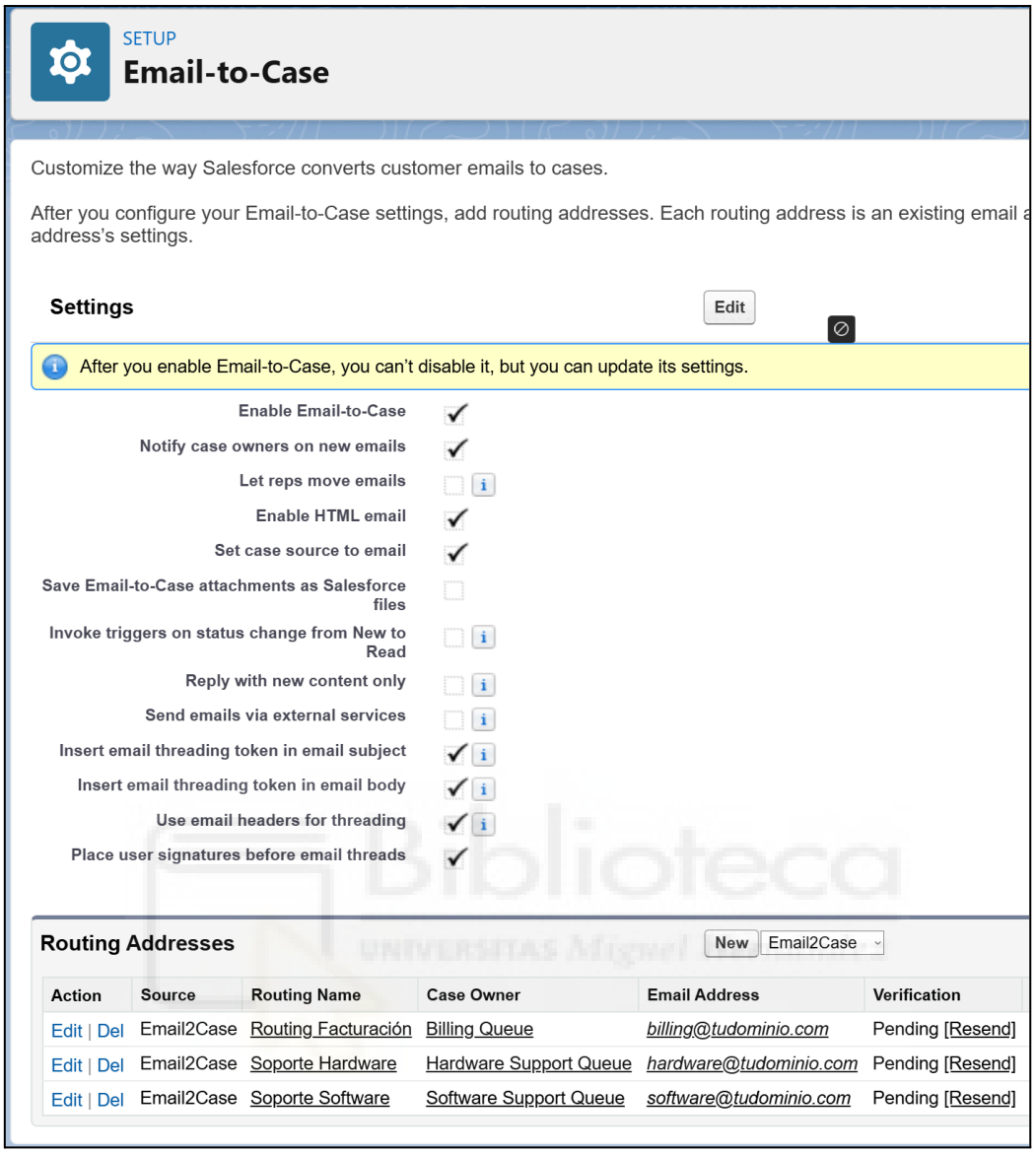

Figura 4.15: Pantalla de configuración de Email-to-Case

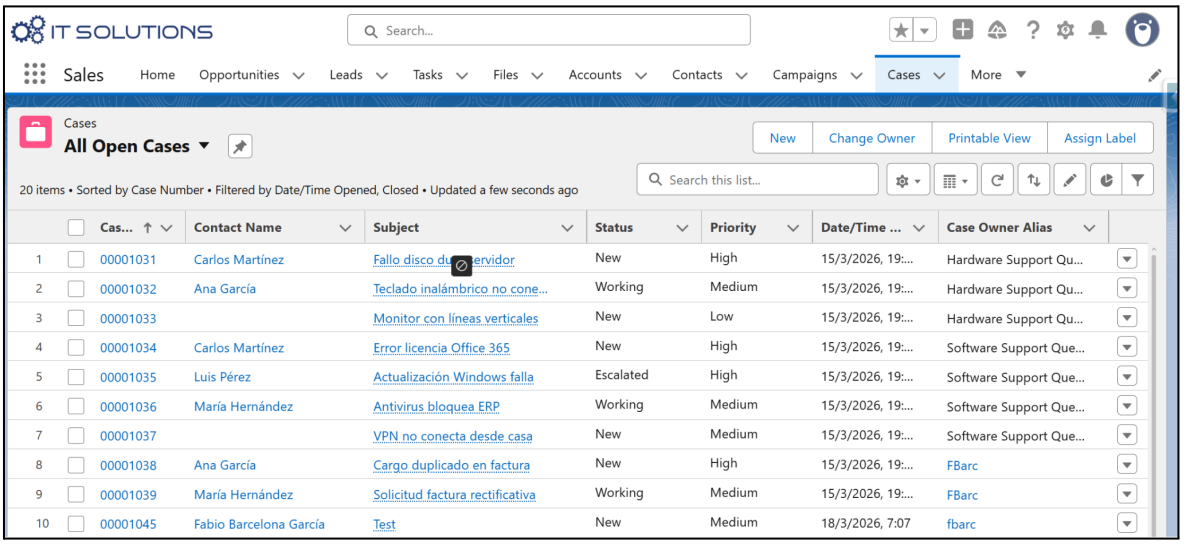

Figura 4.16: Lista de todos los Casos abiertos

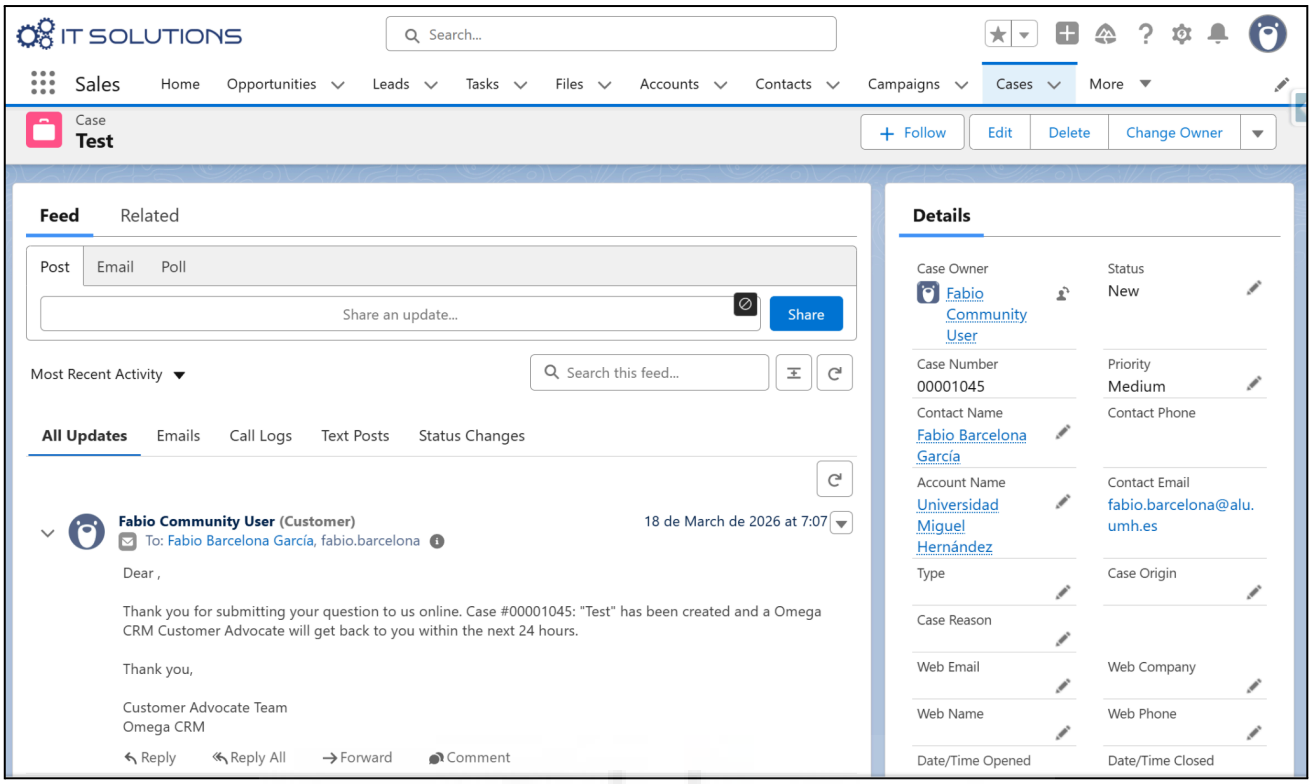

Figura 4.17: Vista detallada de un caso

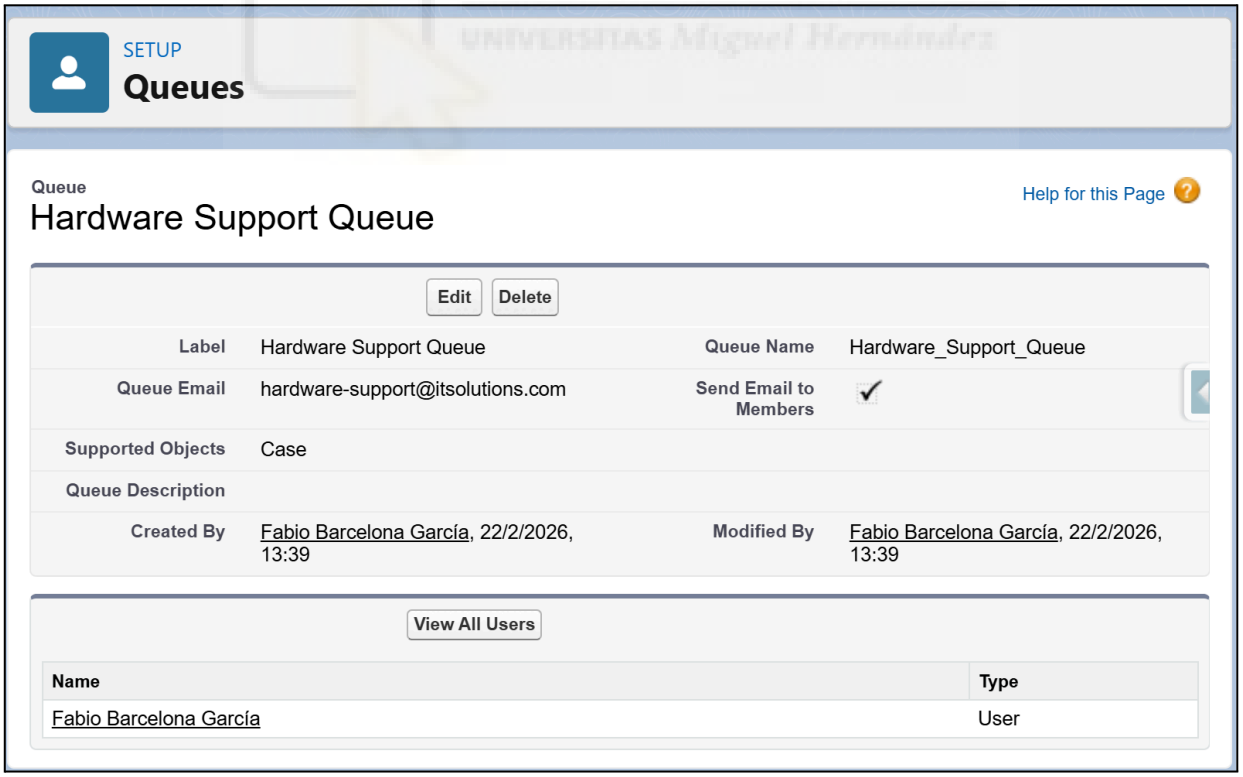

Figura 4.18: Pantalla de configuración de una de las colas, donde también podremos asignar a los integrantes

4.4.3.- Asignación Automática de Leads

La figura 4.19 muestra el canvas del Flow Builder para el Flow **Assign_Lead_to_Owner**, un Flow de tipo Record-Triggered Before Save que se ejecuta en el momento de creación de un Lead, antes de que el registro se inserte en la base de datos.

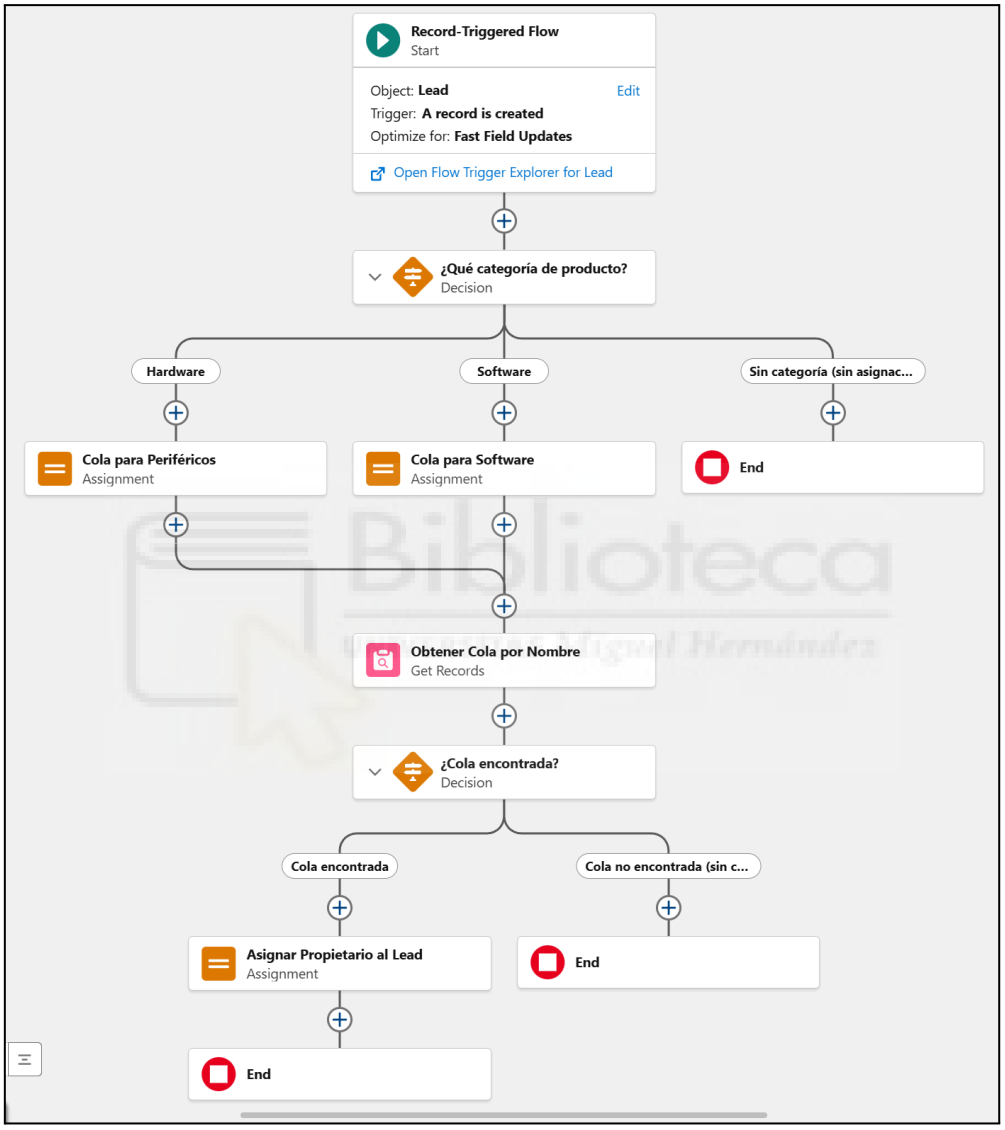

Figura 4.19: Canvas del Flow Builder mostrando el Flow Assign_Lead_to_Owner

El Flow implementa un enrutamiento declarativo basado en el campo personalizado **ProductInterest__c** del Lead, que recoge el área de interés del potencial cliente: Componentes, Periféricos, Software o Soluciones a medida. La lógica sigue la siguiente secuencia:

1. **Decisión "¿Qué categoría de producto?":** evalúa el valor de ProductInterest__c mediante cuatro ramas. Los valores Componentes, Periféricos y Soluciones a

medida asignan el Lead a la cola de Hardware. El valor Software asigna el Lead a la cola de Software

2. **Elemento de obtención de registros "Obtener Cola por Nombre":** consulta el objeto interno Group de Salesforce (donde residen las colas) filtrando por DeveloperName y Type = Queue

3. **Decisión "¿Cola encontrada?":** verifica que la cola obtenida no sea nula antes de proceder. Este paso evita errores en caso de que una cola haya sido renombrada o eliminada en el entorno.

4. **Asignación Asignar Propietario al Lead:** escribe el Id de la cola en el campo OwnerId del registro que se está procesando en el Flow. Al tratarse de un Flow Before Save, la asignación se aplica directamente sobre el registro en memoria sin necesidad de una operación DML, ya que en este punto de la transacción no existe el registro en base de datos.

4.4.4.- Flow de Creación de Pedidos

La figura 4.20 muestra la pantalla del Screen Flow **Order_Creation**, accesible desde el botón "New Order" en las acciones de una Quote.

En primer lugar, el Flow muestra una pantalla de confirmación con los productos del presupuesto (nombre, descripción, familia, cantidad y precio unitario) antes de proceder a la creación. Esta pantalla de confirmación es el único paso interactivo del flow; el resto de la lógica (creación del Order, inserción de OrderItem, actualización de inventario y envío de email) se ejecuta de forma automática en segundo plano.

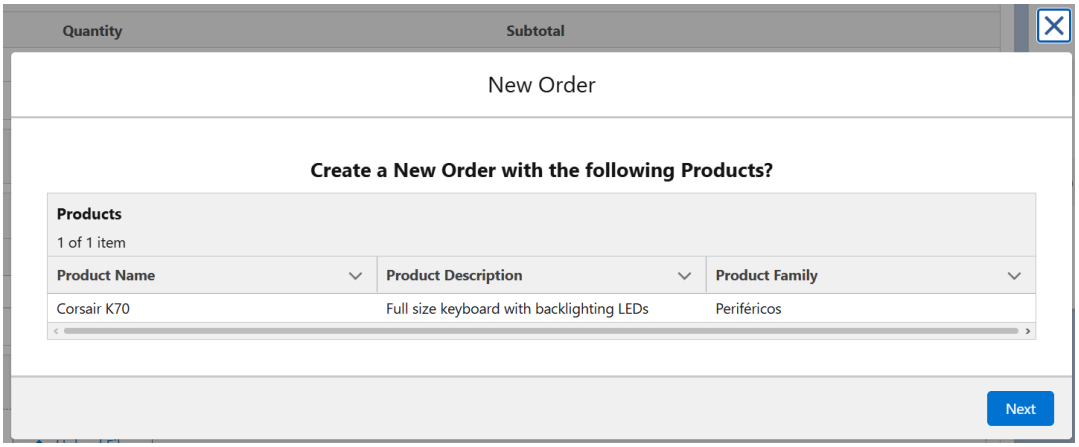

Figura 4.20: Pantalla resumen del Flow Order_Creation

En caso de que el Flow termine satisfactoriamente, se muestra una pantalla con un mensaje indicando que se ha creado el pedido correctamente. En caso contrario, se muestra una pantalla con un mensaje de error y, haciendo uso de la operación de rollback, se vuelve al estado de la base de datos anterior a comenzar el Flow. Esto se hace para evitar crear registros huérfanos.

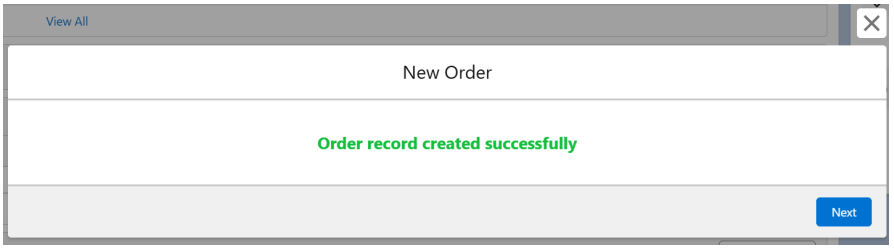

Figura 4.21: Pantalla de creación satisfactoria del Flow Order_Creation

4.4.5.- Automatizaciones de Inventario y Envíos

Esta sección muestra los dos Flows puramente automáticos que gestionan la trazabilidad del stock y del estado de los envíos. A diferencia del Flow Order_Creation, ninguno de los dos presenta interfaz de usuario: se ejecutan en segundo plano en forma de Record-triggered Flow.

4.4.5.1.- Update Inventory on Order Item

La figura 4.22 muestra el canvas del Flow **Update_Inventory_on_Order**, un Flow Record-Triggered After Save que se dispara al crear un **OrderItem**, es decir, cada vez que se añade una línea de producto a un pedido activo. El flujo de ejecución es el siguiente:

1. **Obtener Inventario del Producto:** consulta el registro Inventory__c cuyo campo Product__c coincide con el Product2Id del OrderItem recién creado, recuperando Available_Stock__c, Reserved_Stock__c y Reorder_Point__c.

2. **¿Se encontró registro de inventario?** bifurcación de seguridad: si no existe registro de inventario para ese producto, el Flow termina sin error, evitando excepciones en productos sin seguimiento de stock.

3. **Actualizar Stock:** aplica dos fórmulas calculadas en el propio flow:
    a. Formula_NewAvailable = Available_Stock__c - Quantity (reduce el stock disponible)
    b. Formula_NewReserved = Reserved_Stock__c + Quantity (incrementa el stock reservado)

4. **¿Stock bajo el punto de reposición?** compara Formula_NewAvailable con Reorder_Point__c. Si el nuevo stock disponible cae por debajo del umbral, el Flow crea una Task de prioridad alta con el asunto "Alerta: Stock bajo - requiere reposición" vinculada al registro Inventory__c afectado, que aparecerá en la lista de tareas del administrador.

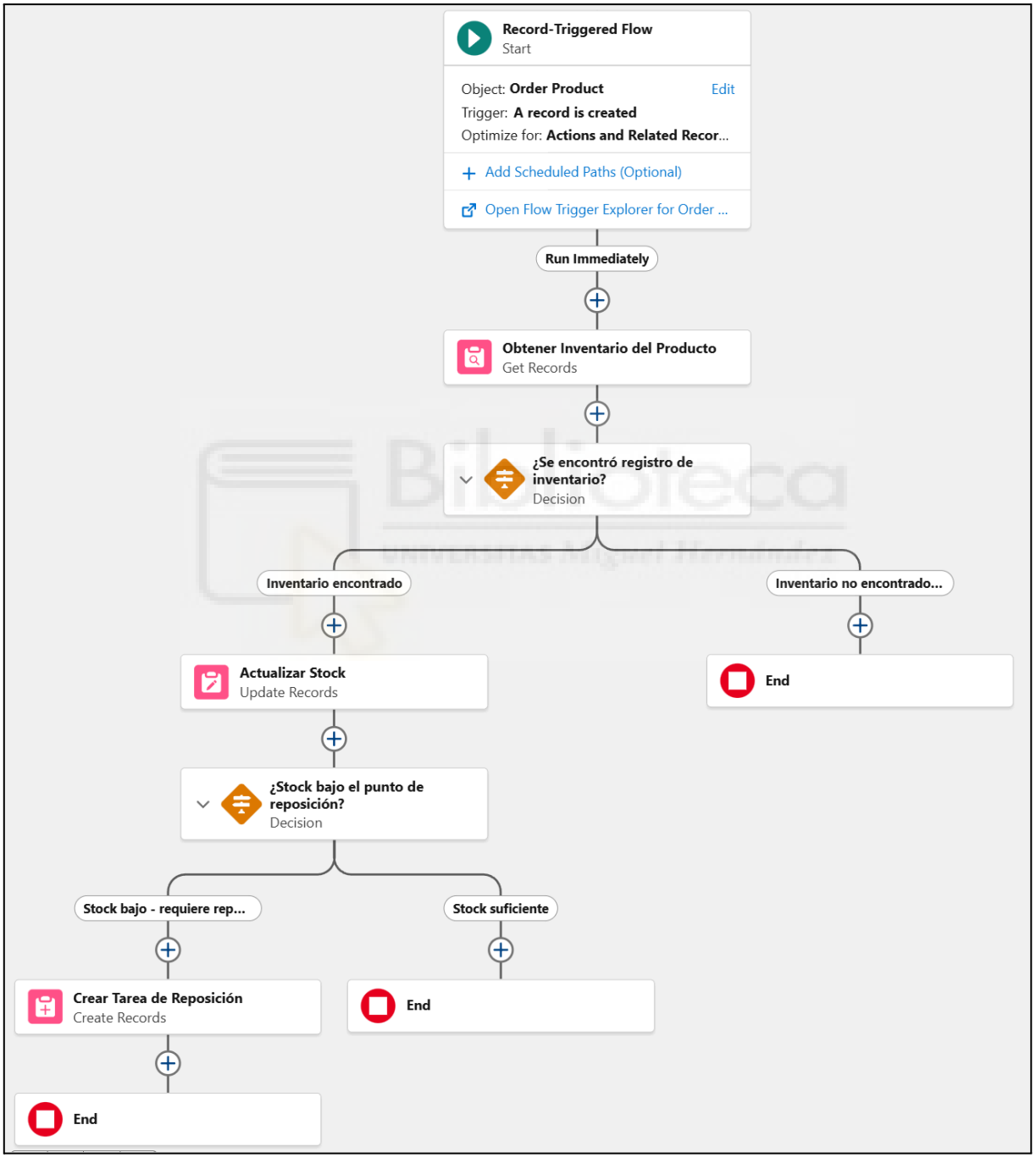

Figura 4.22: Canvas del Flow Builder del Flow Update_Inventory_on_Order

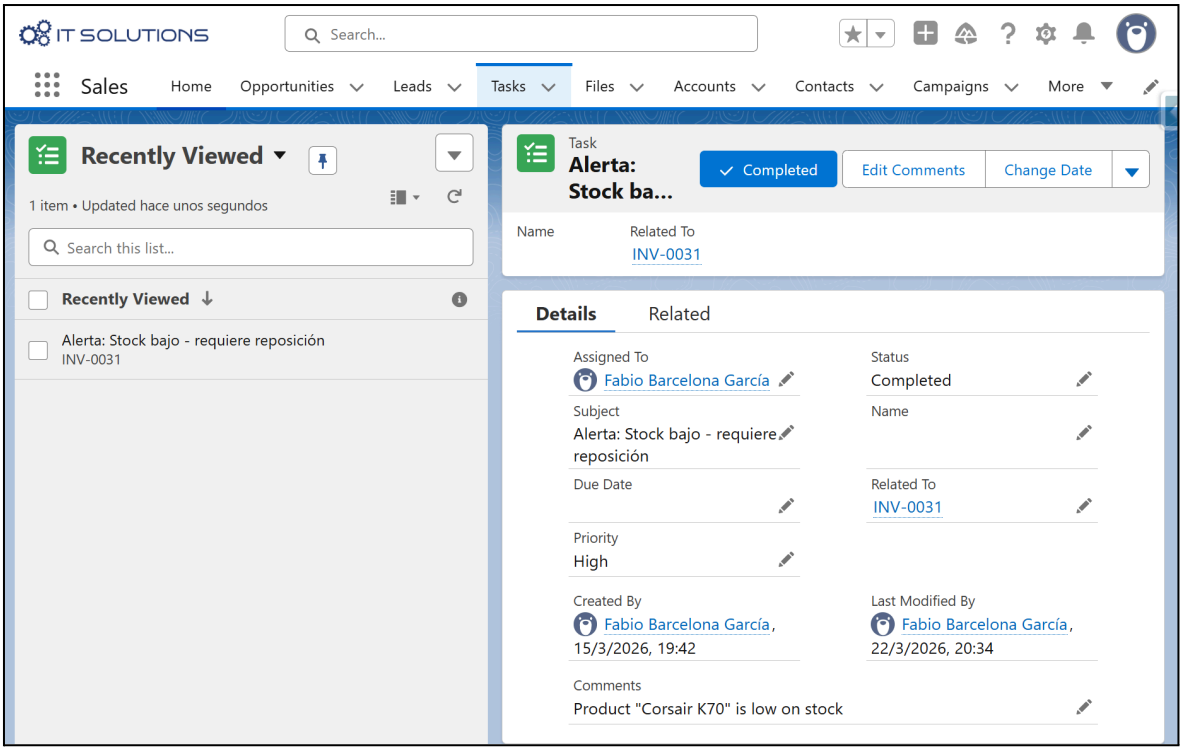

Figura 4.23: Vista detalle de la Tarea de reposición de stock bajo.

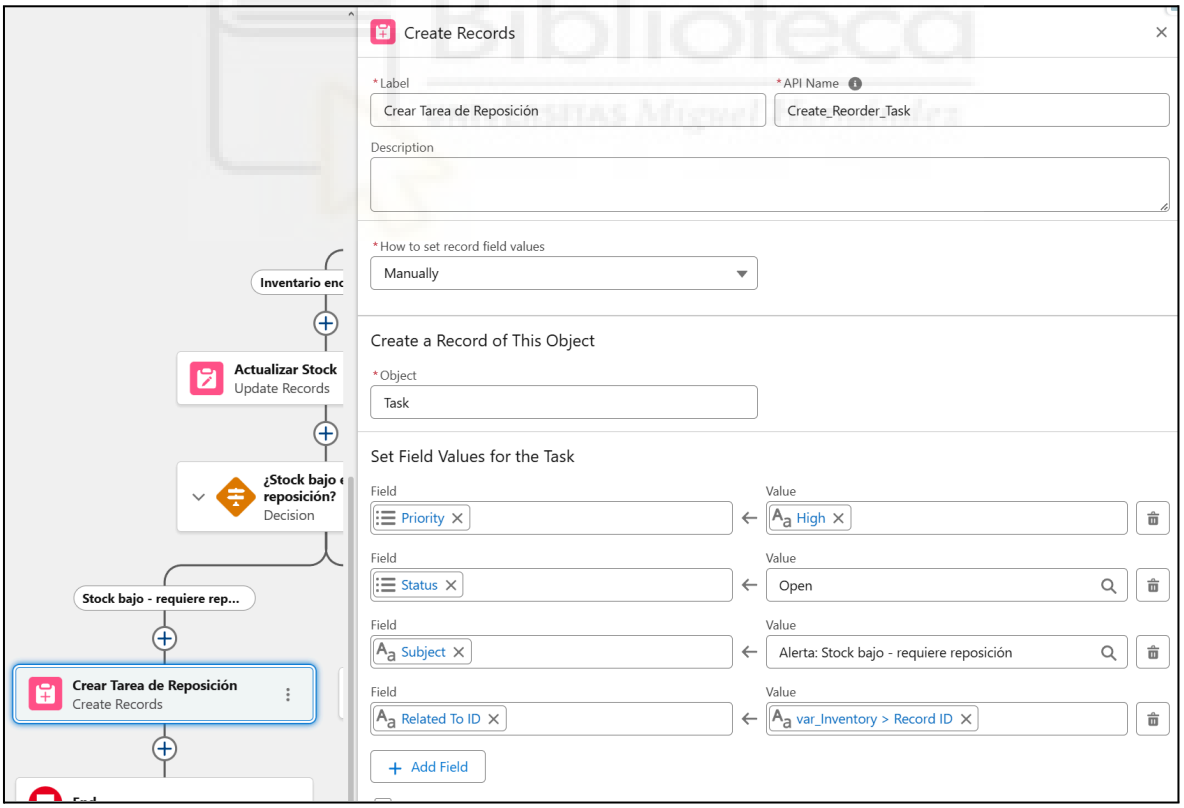

Figura 4.24: Vista detallada del nodo “Crear Tarea de Reposición”

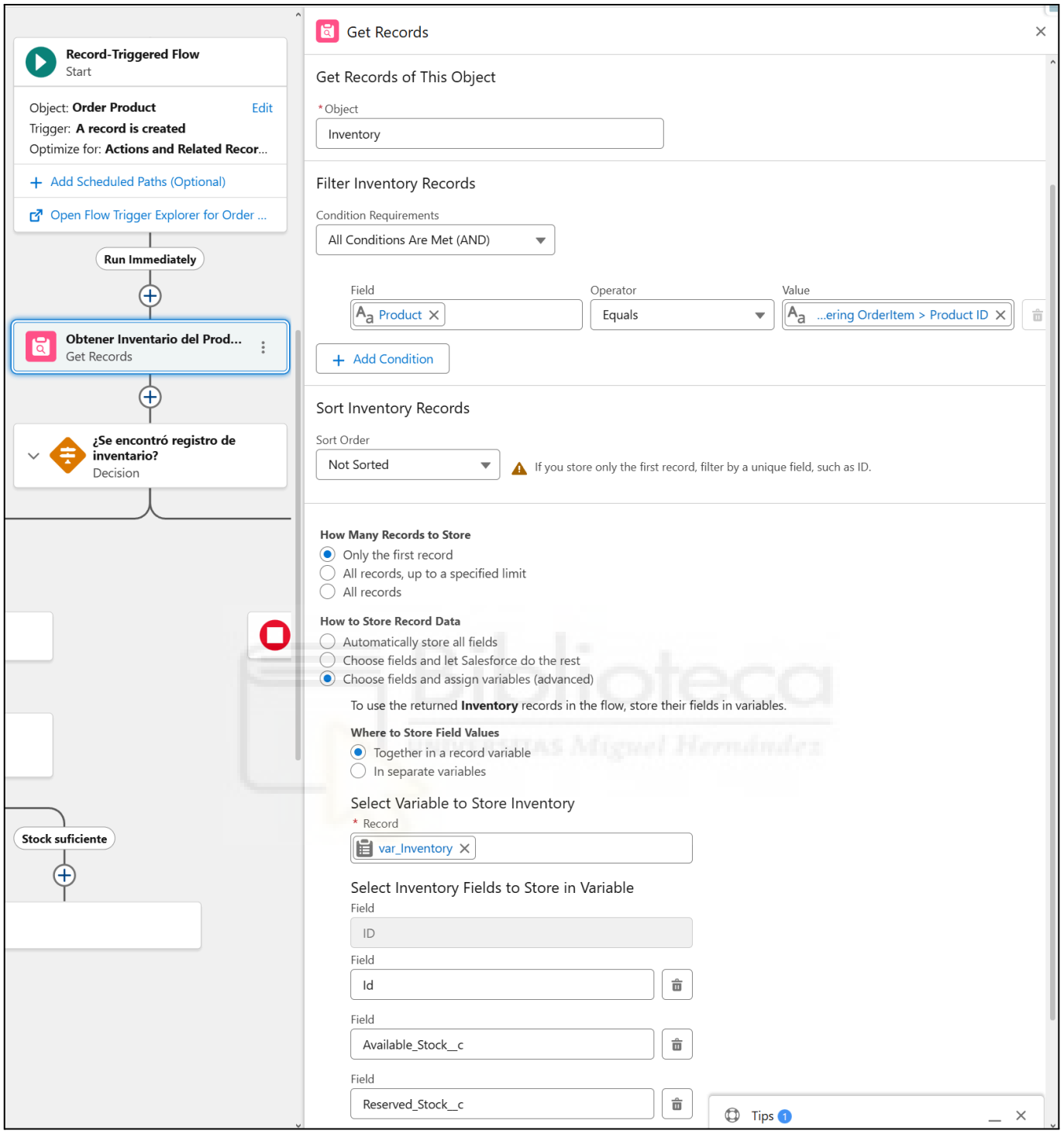

Figura 4.25: Vista detallada del nodo “Obtener Inventario del Producto”

4.4.5.2.- Notificación Envío

La figura 4.26 muestra el canvas del flow **Shipment_Notification**, un Flow Record-Triggered After Save que se ejecuta tanto en la creación como en la actualización de un registro **Shipment_Tracking__c**.

El flow distingue dos escenarios mediante una decisión inicial basada en la fórmula ISNEW():

- **Registro nuevo (envío creado):** actualiza el campo Shipping_Status__c del Order padre a "Enviado" e invoca inmediatamente la acción de Apex ShipmentTrackingService mediante una llamada a su @InvocableMethod, que encola la llamada HTTP al proveedor externo de tracking para obtener el número de seguimiento y la fecha estimada de entrega.

- **Actualización con estado "Entregado":** se activa cuando Status__c cambia a "Entregado" y el valor anterior era distinto, condición evaluada mediante **$Record__Prior.Status__c**. En ese caso, actualiza el Order padre a "Entregado", cerrando el ciclo de vida del envío

La condición de comparación con **$Record__Prior** es relevante: sin ella, el Flow se ejecutaría en cualquier actualización del registro y actualizará el pedido en cada ejecución del trigger.

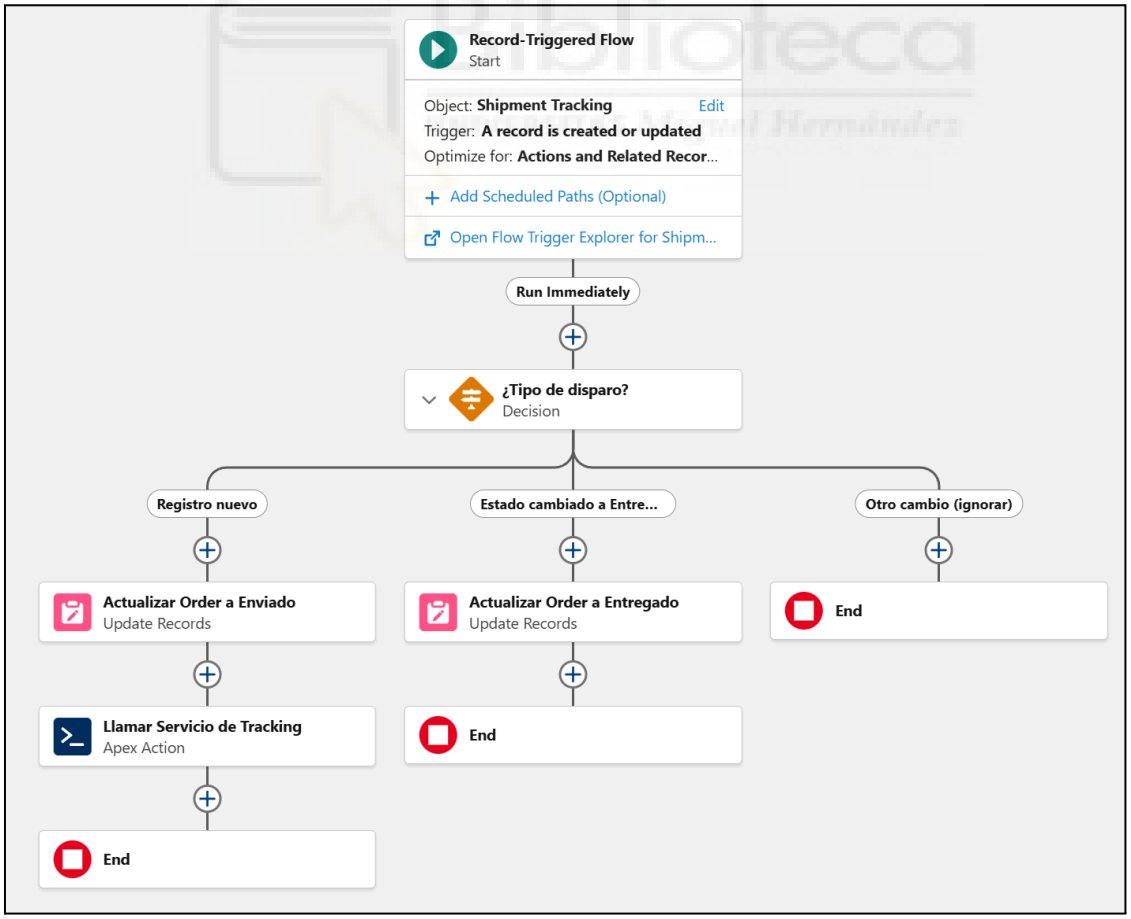

Figura 4.26: Canvas del Flow Builder del Flow Shipment_Notification

4.4.6.- Portal de Seguimiento de Pedidos (Experience Cloud)

El portal de clientes expone el componente **orderTrackingList** en la página principal del Site de Experience Cloud.

Al acceder, el cliente ve sus pedidos en formato de tarjetas, cada una con el número de pedido, la fecha, el importe total, el estado del pedido con un badge de color y el estado del envío asociado.

Al hacer clic en una tarjeta, la vista cambia al componente **orderTrackingDetail**, mostrando el detalle del envío sin necesidad de navegar a otra página.

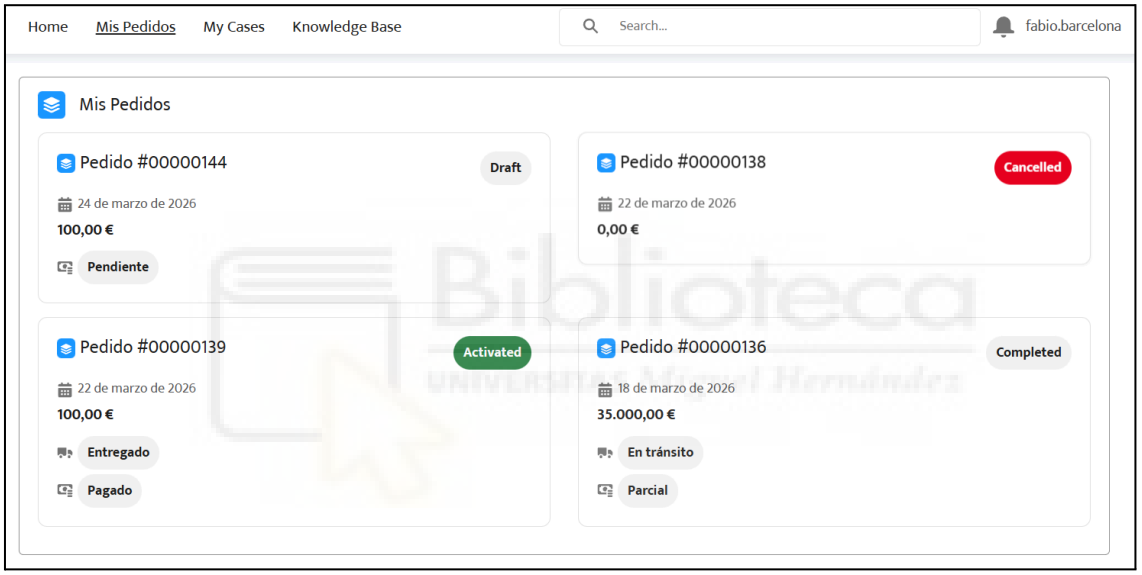

Figura 4.27: Lista de pedidos en el portal Experience Cloud del cliente.

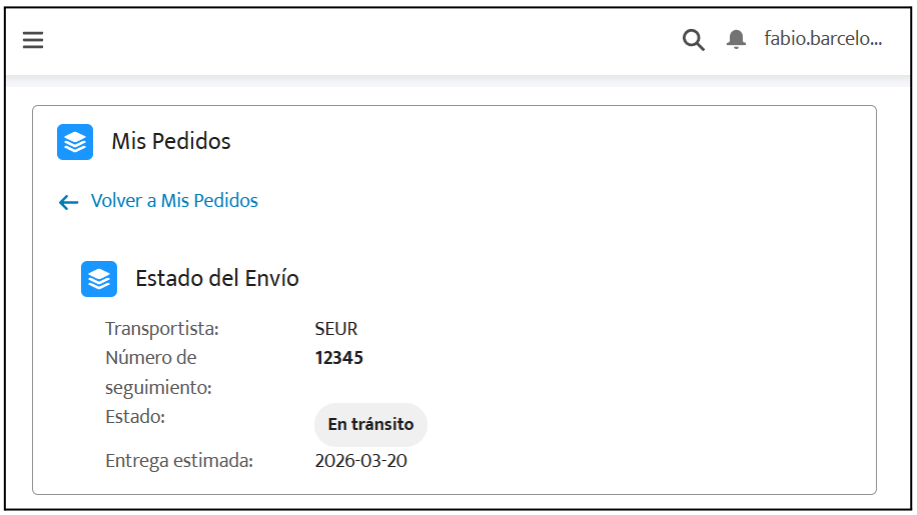

Figura 4.28: Detalle de seguimiento de envío en el portal Experience Cloud

4.4.7.- Dashboards de Ventas y Soporte

Las figuras 4.26 y 4.27 muestran los dos dashboards implementados, accesibles desde la pestaña Dashboards de la aplicación.

4.4.7.1.- Dashboards de Ventas

Presenta tres componentes:

1. **Pipeline por Etapa:** visualiza el número de oportunidades activas agrupadas por etapa de venta, excluyendo Closed Won y Closed Lost. El eje Y representa el total de oportunidades, y cada barra corresponde a una etapa de la oportunidad.

2. **Oportunidades Ganadas Este Mes**: lista las oportunidades cerradas como ganadas en el mes actual, con nombre de oportunidad, cuenta asociada a la oportunidad, fecha de cierre, importe y propietario.

3. **Oportunidades Sin Actividad 30 Días:** muestra oportunidades abiertas cuya última actividad registrada supera los 30 días, con nombre, cuenta, etapa, importe y fecha de última actividad.

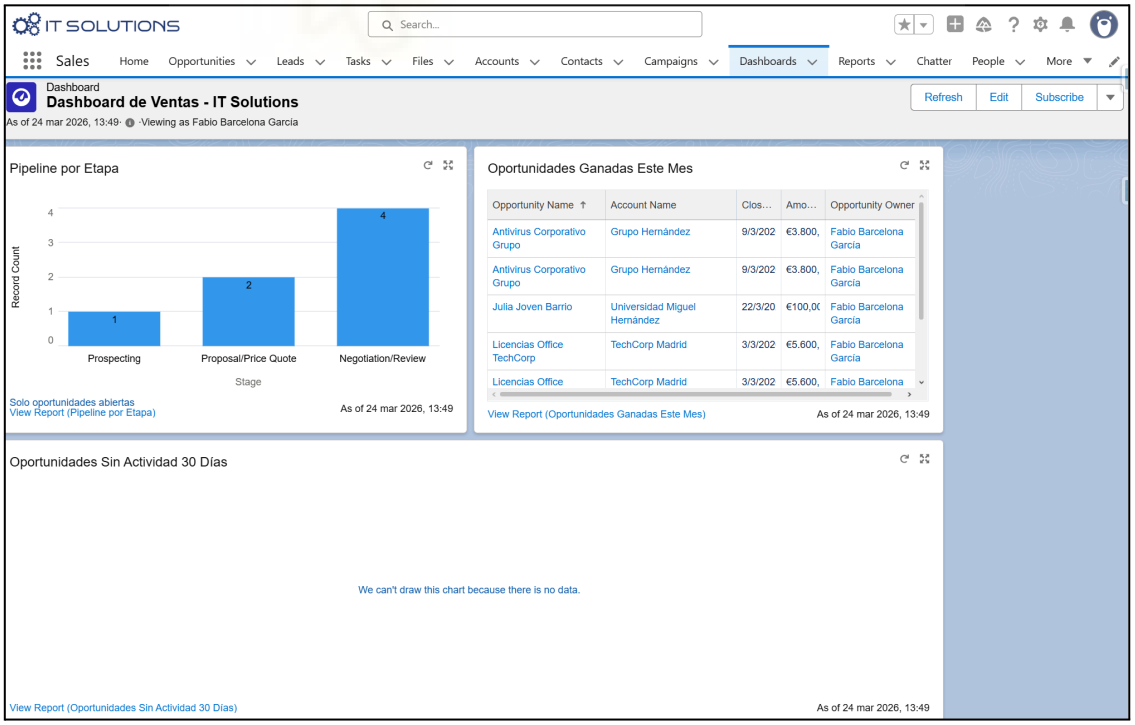

Figura 4.29: Dashboard de Ventas

4.4.7.2.- Dashboards de Soporte

Presenta tres componentes:

1. **Casos Abiertos por Cola:** muestra el volumen de casos abiertos agrupado por cola (Hardware Support, Software Support, Billing), con valores numéricos enteros sobre cada barra que representan el total de casos.

2. **Casos por Tipo y Estado**: detalla los casos abiertos con propietario (cola o agente), rol y fecha de creación.

3. **Tiempo Medio de Resolución:** muestra el promedio histórico de horas de resolución como un valor numérico único. Este componente está basado en el campo **Resolution_Time__c**.

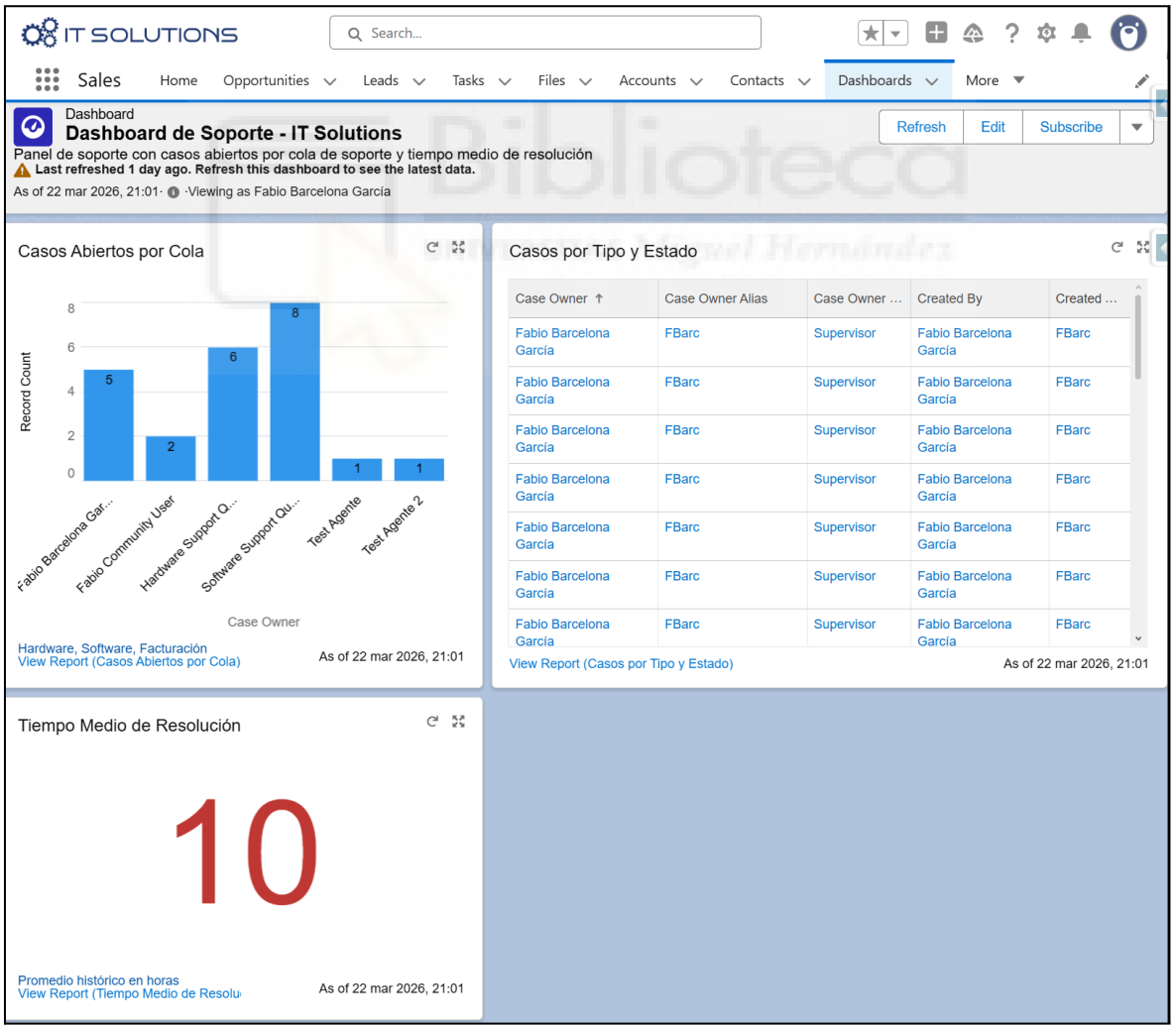

Figura 4.30: Dashboard de Soporte

4.4.8.- Fragmentos de Código: Problemas Técnicos Resueltos

Esta sección documenta tres patrones de implementación no triviales que resolvieron limitaciones específicas de la plataforma Salesforce durante el desarrollo.

4.4.8.1.- Consulta SOQL Dinámica para Quote

El trigger **OpportunityTriggerHandler** necesita acceder al objeto **Quote** para sincronizar el importe sincronizado con la Oportunidad. Sin embargo, el objeto Quote no está disponible en todas las organizaciones Salesforce, ya que su activación es opcional, y una referencia estática en SOQL lanzaría un error de compilación en orgs donde no está habilitado, por lo que la clase no se podría desplegar al entorno.

La solución adoptada fue construir la consulta mediante SOQL dinámico con **Database.query()**, de modo que el string se evalúa en tiempo de ejecución y no en compilación:

Algoritmo: 4.1: SOQL Dinámico para Quote

```
1  public class OpportunityTriggerHandler {
2    public static void beforeUpdate(List<Opportunity> newOpps,
3                                    Map<Id, Opportunity> oldOppMap) {
4      List<Opportunity> oppsClosingAsWon = new List<Opportunity>();
5      Set<Id> oppIds = new Set<Id>();
6      for (Opportunity opp : newOpps) {
7        Opportunity oldOpp = oldOppMap.get(opp.Id);
8        if (opp.StageName == 'Closed Won' &&
9            oldOpp.StageName != 'Closed Won') {
10             oppsClosingAsWon.add(opp);
11             oppIds.add(opp.Id);
12       }
13     }
14     if (oppsClosingAsWon.isEmpty()) {
15       return;
16     }
17     validateQuotesBeforeClosing(oppsClosingAsWon, oppIds);
18   }
19
20   private static void validateQuotesBeforeClosing(List<Opportunity>
21                                                   oppsClosingAsWon,
22                                                   Set<Id> oppIds) {
23     try {
24       Map<Id, SObject> syncedQuoteByOppId = new Map<Id, SObject>();
25
26       List<SObject> syncedQuotes = Database.query(
27           'SELECT Id, Status, OpportunityId FROM Quote ' +
28           'WHERE OpportunityId IN :oppIds AND IsSyncing = true'
29       );
```

```
30
31        for (SObject q : syncedQuotes) {
32          Id oppId = (Id) q.get('OpportunityId');
33          syncedQuoteByOppId.put(oppId, q);
34        }
35
36        for (Opportunity opp : oppsClosingAsWon) {
37          if (!syncedQuoteByOppId.containsKey(opp.Id)) {
38            opp.addError('No se puede cerrar oportunidad ...');
39            continue;
40          }
41
42          String status = (String) syncedQuoteByOppId
43                                     .get(opp.Id).get('Status');
44
45          if (status != 'Accepted') {
46            opp.addError('El presupuesto debe estar "Aceptado"');
47          }
48
49        }
50      }
51      catch (QueryException e) {
52        System.debug('Quote validation skipped: ' + e.getMessage());
53      }
54    }
55  }
```

Este patrón permite distribuir el mismo paquete de metadatos entre organizaciones con distinta configuración sin modificar el código fuente, lo que es especialmente relevante en entornos de desarrollo compartido o si se planea distribuir la solución.

4.4.8.2.- Separación entre llamada HTTP asíncrona y método invocable

ShipmentTrackingService expone dos puntos de entrada con propósitos distintos: un método **@InvocableMethod** para ser invocado desde Flows, y un método **@future(callout=true)** para realizar la llamada HTTP al proveedor de tracking.

El problema es que Salesforce no permite que un @InvocableMethod realice callouts directamente cuando es invocado desde un contexto de trigger o batch, ya que en ese caso el callout debe ser asíncrono.

La solución fue separar ambas responsabilidades en métodos independientes: el método invocable encola la llamada asíncrona, y el método @future ejecuta la HTTP real:

Algoritmo: 4.2: Lógica de @InvocableMethod y @future separadas

```
public class ShipmentTrackingService {

  @InvocableMethod(label='Update Shipment Tracking'
                   description='Solicita actualización al proveedor')
  public static void updateTrackingFromFlow(
          List<String> shipmentIds) {

      for (String id : shipmentIds) {
          callTrackingProviderAsync(id);
      }
  }

  @future(callout=true)
  private static void callTrackingProviderAsync(String shipmentId) {

    Shipment_Tracking__c st = [
      SELECT Id, Tracking_Number__c, Carrier__c
      FROM Shipment_Tracking__c
      WHERE Id = :shipmentId
    ];

    HttpRequest req = new HttpRequest();
    req.setEndpoint(
      'callout:TrackingProvider/api/track/' +
      st.Tracking_Number__c
    );
    req.setMethod('GET');

    HttpResponse res = new Http().send(req);

    if (res.getStatusCode() == 200) {
      TrackingResponse tr =
          (TrackingResponse) JSON.deserialize(res.getBody(),
                                              TrackingResponse.class);

      st.Status__c = tr.status;
      st.Estimated_Delivery__c = tr.estimatedDelivery;
      st.Last_API_Update__c = Datetime.now();
      update st;
    }
  }

  private class TrackingResponse {
    public String status;
    public Date estimatedDelivery;
  }

}
```

El uso de Named Credentials (callout:TrackingProvider) externaliza la URL base y las credenciales de autenticación fuera del código, evitando credenciales en texto plano y simplificando el cambio de entorno (sandbox, producción) sin modificar el código fuente.

4.4.8.3.- Batch con Estado para Actualización de Inventario

InventoryUpdateBatch actualiza el stock disponible de todos los productos vendidos en pedidos activados durante el día.

Algoritmo: 4.3: Uso de Database.Stateful para serializar el estado entre batches

```
public class InventoryUpdateBatch
             implements Database.Batchable<SObject>,
             Database.Stateful {

  // Variables de instancia, persiste entre chunks
  private Integer recordsProcessed = 0;
  private List<String> errors = new List<String>();

  public Database.QueryLocator start(Database.BatchableContext bc) {
    return Database.getQueryLocator([
      SELECT Product2Id, SUM(Quantity) totalSold
      FROM OrderItem
      WHERE Order.Status = 'Activated' AND Order.EffectiveDate = TODAY
      GROUP BY Product2Id
    ]);
  }

  public void execute(Database.BatchableContext bc,
                      List<AggregateResult> scope) {

    List<Inventory__c> toUpdate = new List<Inventory__c>();

    for (AggregateResult ar : scope) {
      Id prodId = (Id) ar.get('Product2Id');
      Decimal sold = (Decimal) ar.get('totalSold');

      List<Inventory__c> inv = [
        SELECT Id, Available_Stock__c FROM Inventory__c
        WHERE Product__c = :prodId LIMIT 1
      ];

      if (!inv.isEmpty()) {
        inv[0].Available_Stock__c -= sold;
        toUpdate.add(inv[0]);
        recordsProcessed++;
      } else {
        errors.add('Sin inventario para producto: ' + prodId);
      }
    }
    update toUpdate;
  }

  public void finish(Database.BatchableContext bc) {
    System.debug('Registros procesados: ' + recordsProcessed);
  }
}
```

El reto principal es que los pedidos se procesan en lotes de hasta 200 registros, límite impuesto por la plataforma, y es necesario acumular un contador global de unidades procesadas para generar el informe final de ejecución.

La interfaz **Database.Stateful** resuelve esto: a diferencia de un batch sin estado donde cada chunk ejecuta una instancia independiente, Database.Stateful preserva los valores de las variables de instancia entre chunks:

El uso de SUM() con GROUP BY en el método start() es especialmente relevante: en lugar de recuperar todos los OrderItem individuales y procesar el agregado en Apex, la operación de agregado se delega al motor SOQL para que nos devuelva los resultados ya listos para consumir por el Batch.

4.5.- PRUEBAS/IMPLANTACIÓN

4.5.1.- Pruebas de la aplicación

Salesforce impone como requisito de plataforma que al menos el 75% de cada clase Apex esté cubierto por tests antes de permitir cualquier despliegue a producción.

Para cumplir este requisito y validar el comportamiento de la lógica de negocio, se desarrollaron seis clases de test independientes, una por cada clase, que hacen uso de la clase utilitaria **TestDataFactory** para la creación de datos de prueba reutilizables.

La tabla siguiente resume las clases cubiertas:

Tabla 4.10: Clases de producción junto a su clase test y escenarios que cubre

Clase de producción	Clase de test	Escenarios cubiertos
ProductSearch Controller	ProductSearch ControllerTest	Búsqueda con término válido, término corto (sin resultados), adición de productos a oportunidad
ShipmentTracking Service	ShipmentTracking ServiceTest	Callout HTTP con respuesta correcta, tracking number nulo, respuesta de error del proveedor
OrderTracking Controller	OrderTracking ControllerTest	Obtención de tracking, refresco manual, consulta de pedidos del usuario de comunidad
OpportunityTimeline Controller	OpportunityTimeline ControllerTest	Historial con múltiples cambios de etapa, oportunidad sin historial
InventoryUpdate Batch	InventoryUpdate BatchTest	Ejecución completa del batch, detección de stock bajo, producto sin inventario asociado
OpportunityTrigger Handler	OpportunityTrigger HandlerTest	Cierre como Closed Won con Quote sincronizado, intento de cierre sin Quote aprobado

Las clases de test utilizan Test.startTest() / Test.stopTest() para aislar los límites de gobernador, y Test.setMock() con implementaciones de HttpCalloutMock en los tests que verifican llamadas HTTP, ya que Salesforce no permite callouts reales en contexto de test

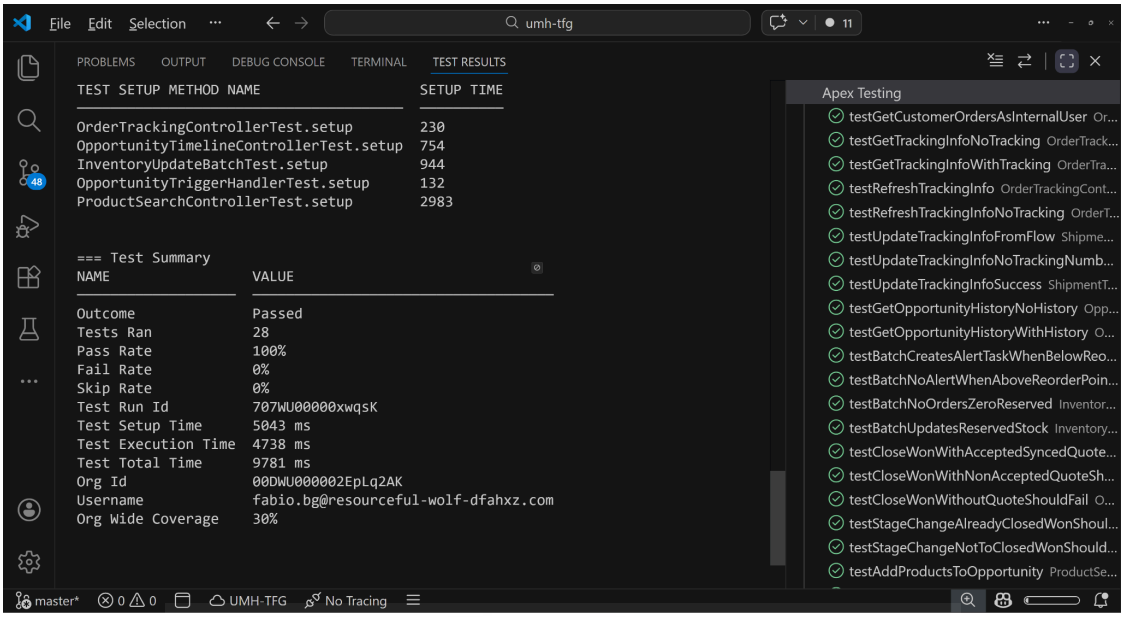

Figura 4.31: Resultados de ejecución de las clases test en Visual Studio Code

Algoritmo: 4.4: Ejemplo de clase test que implementa HttpCalloutMock

```
@IsTest
private class OrderTrackingControllerTest {
  @TestSetup
  static void setup() {
    Account acc = TestDataFactory.createAccount('Test Account Tracking');
    TestDataFactory.createOrder(acc.Id);
  }

  @IsTest
  static void testRefreshTrackingInfo() {
    Order ord = [SELECT Id FROM Order LIMIT 1];

    Test.setMock(HttpCalloutMock.class, new TrackingMock());

    Test.startTest();
    TestDataFactory.createShipmentTrackingWithStatus(ord.Id,
                      'TEST-TRACK-002', 'DHL', 'Pendiente de recogida');
    OrderTrackingController.refreshTrackingInfo(ord.Id);
    Test.stopTest();

    Shipment_Tracking__c updated = [
      SELECT Status__c, Last_API_Update__c FROM Shipment_Tracking__c
      WHERE Order__c = :ord.Id LIMIT 1
    ];
    Assert.isNotNull(updated.Last_API_Update__c,
         'Last_API_Update__c debería estar actualizado tras el refresh');
  }

```

```
29    private class TrackingMock implements HttpCalloutMock {
30      public HTTPResponse respond(HTTPRequest req) {
31        HttpResponse res = new HttpResponse();
32        res.setHeader('Content-Type', 'application/json');
33        res.setBody('{"status":"En tránsito",
34                     "estimated_delivery":"2026-03-15"}');
35        res.setStatusCode(200);
36        return res;
37      }
38    }
39  }
```

4.5.2.- Guía de implantación

El despliegue de la aplicación desde el repositorio a una nueva organización Salesforce requiere completar los siguientes pasos en orden, ya que algunos metadatos no siempre se pueden desplegar y dependen de configuración manual previa en el entorno.

4.5.2.1.- Prerrequisitos en la org destino

Antes del despliegue, la organización debe tener activadas las siguientes funcionalidades:

1. **Quotes**: Necesario para que **OpportunityTriggerHandler** funcione correctamente. Como se ha detallado anteriormente, el código está diseñado para no fallar si Quotes no está habilitado, aunque la validación de presupuestos quedará inoperativa.

2. **Experience Cloud:** el site **IT_Solutions_Customer_Portal** se despliega como metadato pero requiere que la funcionalidad esté habilitada en la org antes de intentar desplegar la comunidad, de lo contrario la plataforma rechazará el despliegue con un error.

3. **Email-to-Case**: Al igual que la comunidad de Experience Cloud, toda la configuración de direcciones de enrutamiento de Email-to-Case se despliega como metadato pero debe activarse previamente en la org destino.

4.5.2.2.- Despliegue de metadatos

Con Salesforce CLI instalado y la org autenticada:

sf project deploy start --source-dir force-app --target-org <alias>

```
PS D:\Salesforce\Projects\umh-tfg> sf project deploy start --metadata ApexClass:InventoryUpdateBatch --target-org UMH-TFG

──────────── Deploying Metadata ────────────

Deploying v66.0 metadata to fabio.bg@resourceful-wolf-dfahxz.com using the v66.0 REST API.

✓ Preparing 168ms
○ Waiting for the org to respond - Skipped
✓ Deploying Metadata 1.85s
  ▸ Components: 1/1 (100%)
○ Running Tests - Skipped
○ Updating Source Tracking - Skipped
✓ Done 0ms

Status: Succeeded
Deploy ID: 0AfWU00000WyOY50AN
Target Org: fabio.bg@resourceful-wolf-dfahxz.com
Elapsed Time: 2.02s

Deployed Source
```

State	Name	Type	Path
Unchanged	InventoryUpdateBatch	ApexClass	force-app\main\default\classes\InventoryUpdateBatch.cls
Unchanged	InventoryUpdateBatch	ApexClass	force-app\main\default\classes\InventoryUpdateBatch.cls-meta.xml

Figura 4.32: Resultado de la ejecución del comando de despliegue, mostrando los pasos realizados y una tabla con todos los componentes desplegados

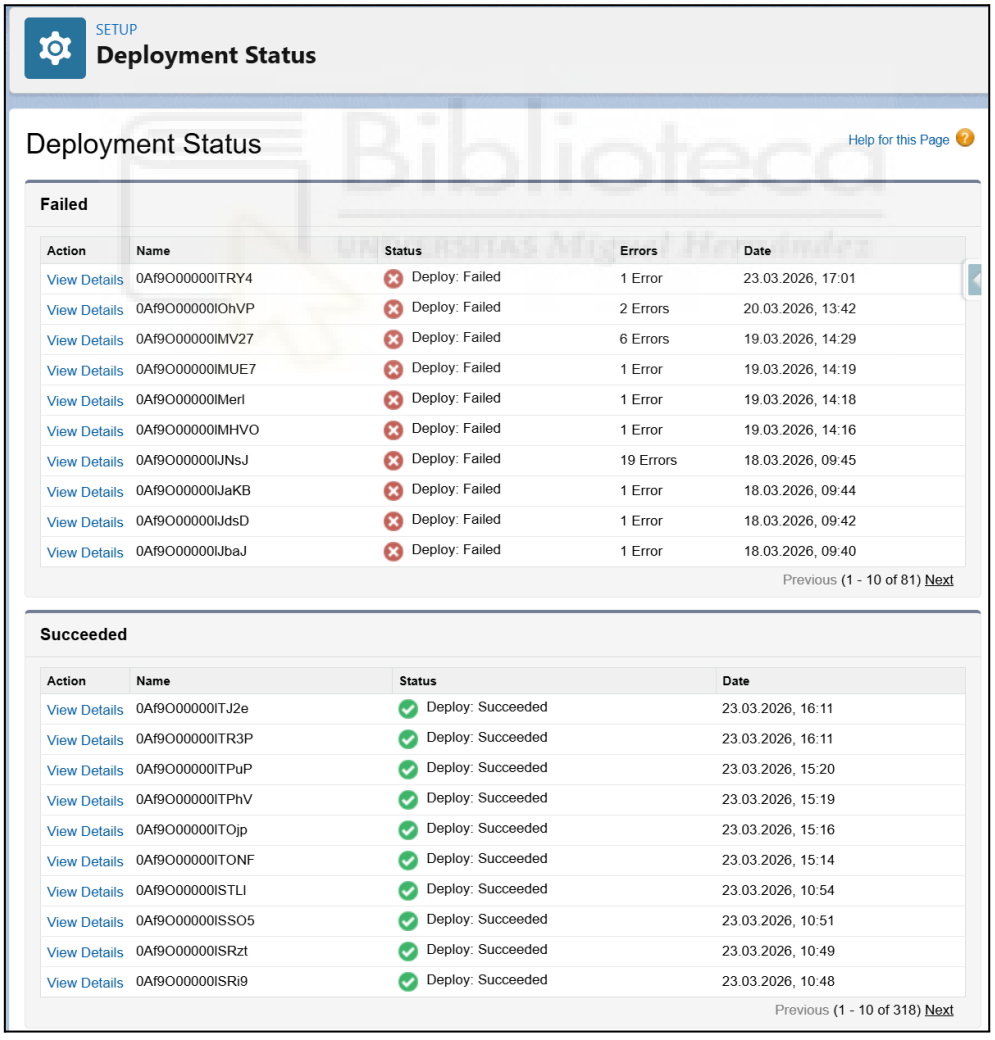

Figura 4.33: Pantalla de despliegues recientes en la interfaz de Salesforce

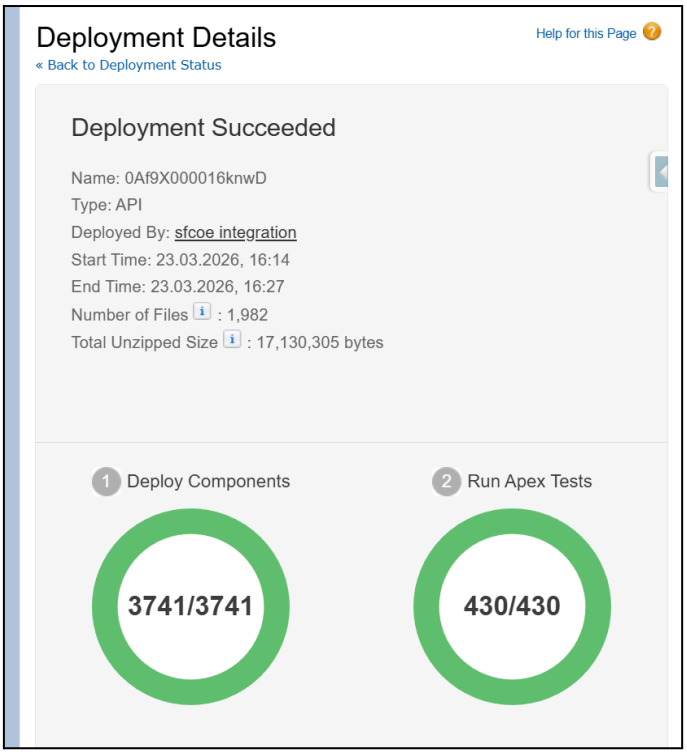

Figura 4.34: Pantalla detalle de un despliegue satisfactorio

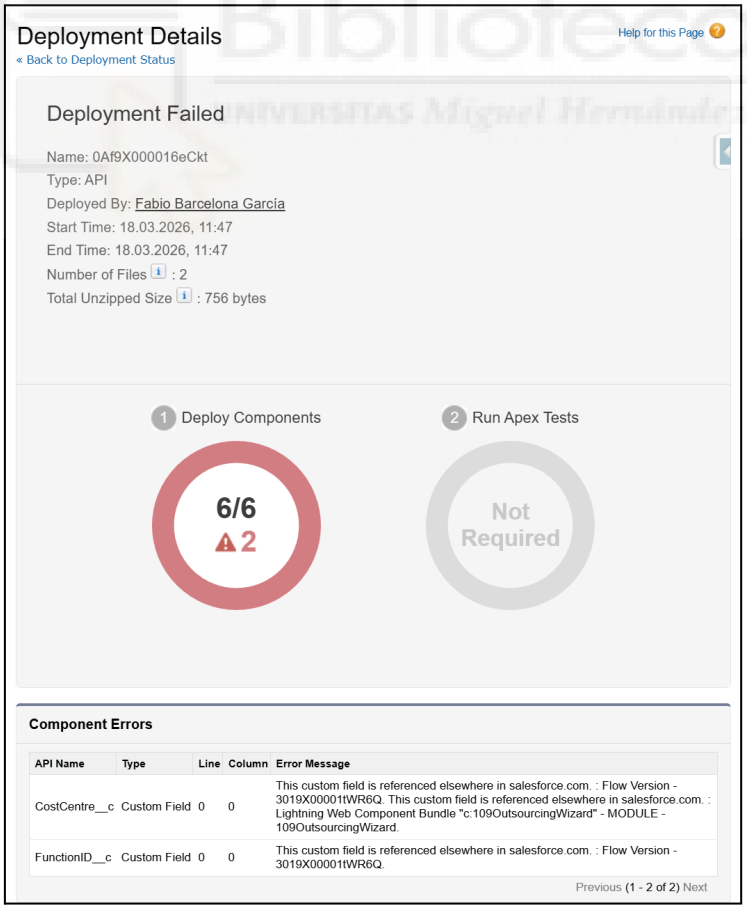

Figura 4.35: Pantalla detalle de un despliegue fallido, mostrando los errores

4.5.2.3.- Configuración post despliegue

Tras el despliegue, es necesario completar manualmente:

- **Named Credential "TrackingProvider"**: la URL base se despliega, pero las credenciales de autenticación (usuario/contraseña, token OAuth, JWT, etc) son privadas y quedan cifradas en la org por motivos de seguridad, no se pueden traer a Visual Studio Code ni desplegar a una org, deben ser configuradas manualmente en el Setup.
    - Primero deberemos configurar un External Credential, una configuración que define de forma segura cómo una organización se autentica contra servicios externos usando métodos como OAuth, API keys, tokens JWT, etc, evitando escribir credenciales sensibles en el código y permitiendo asociar identidades a usuarios o perfiles. Los External Credentials son reutilizables entre Named Credentials

    - Segundo, crearemos un Named Credential, una configuración que define el endpoint del servicio externo, autenticado mediante la External Credential, permitiendo realizar callouts desde Apex sin gestionar manualmente las URLs o headers incluidos en la petición.

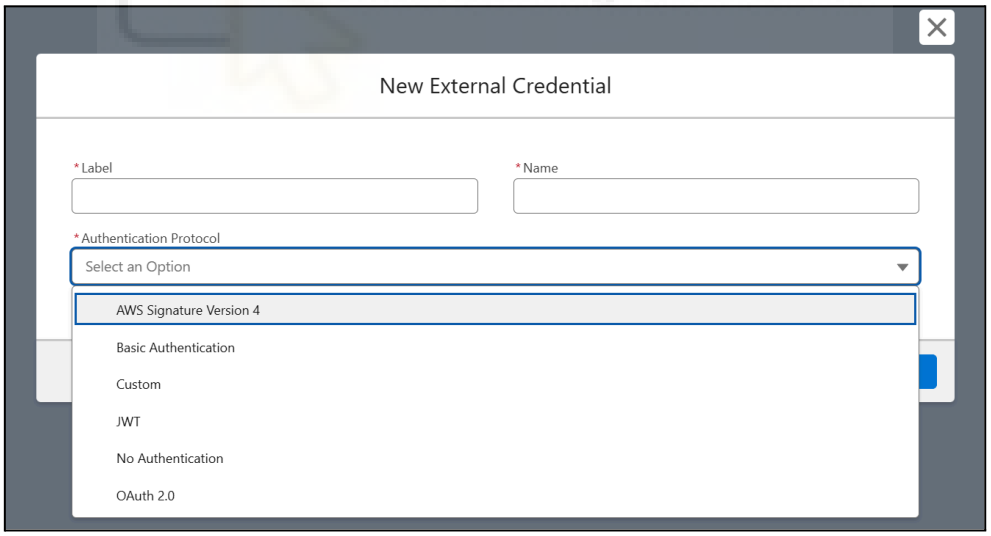

Figura 4.36: Configuración de credenciales de un sistema externo

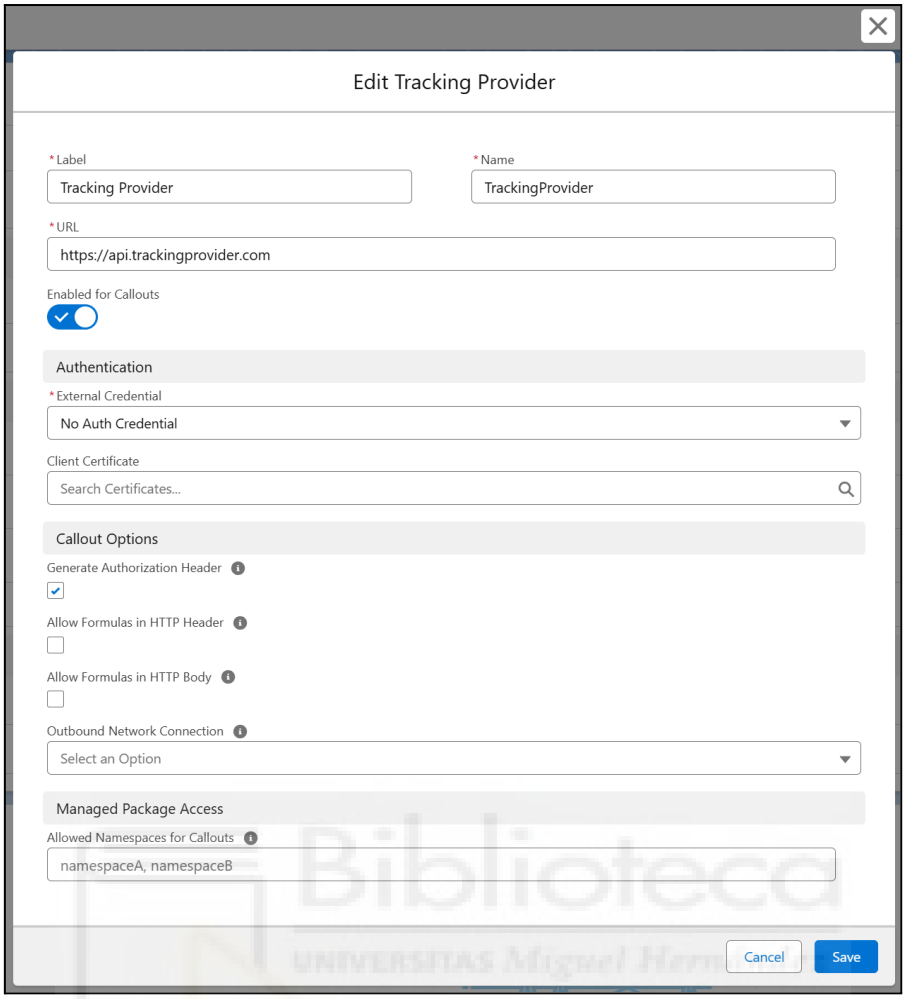

Figura 4.37: Configuración de Named Credential. Definición de URL, credenciales de autenticación, certificados, etc.

- **Activación de la comunidad de Experience Cloud:** Por defecto, las comunidades se despliegan desactivadas. Se debe configurar la comunidad para cambiar el estado de **UnderConstruction** a **Active**

- **Asignación de Permission Sets:** Las asignaciones de los permission sets a los usuarios es una tarea manual, tanto usuarios internos como externos. Se asignan en el perfil de cada usuario, o masivamente desde la página del permission set; donde podemos buscar tantos usuarios como queramos y asignar con un clic

- **Programación del batch:** La clase **InventoryUpdateBatch** debe programarse por interfaz o por código. Ejemplo del código a ejecutar:

Algoritmo: 4.5: Programación de un batch mediante código

```
1  System.schedule('Inventory Daily Update', '0 0 2 * * ?',
2                  new InventoryUpdateBatch());
```

La programación del batch por interfaz se realiza desde el Setup de Salesforce. Deberemos elegir un nombre identificable para el proceso programado, la clase a ejecutar y seleccionar, ya sea con el builder que nos proporciona la interfaz o mediante una expresión CRON, la frecuencia con la que se deberá ejecutar.

La hora de preferencia, tanto configurándose por código como por interfaz, es orientativa ya que, aunque se suele cumplir con bastante exactitud, depende de la disponibilidad de los recursos del sistema.

Una diferencia entre configurar el batch por interfaz en vez de por código es que nos obliga a seleccionar una fecha fin del proceso, mientras que por código es un parámetro que se puede omitir, quedando programado sin fecha fin.

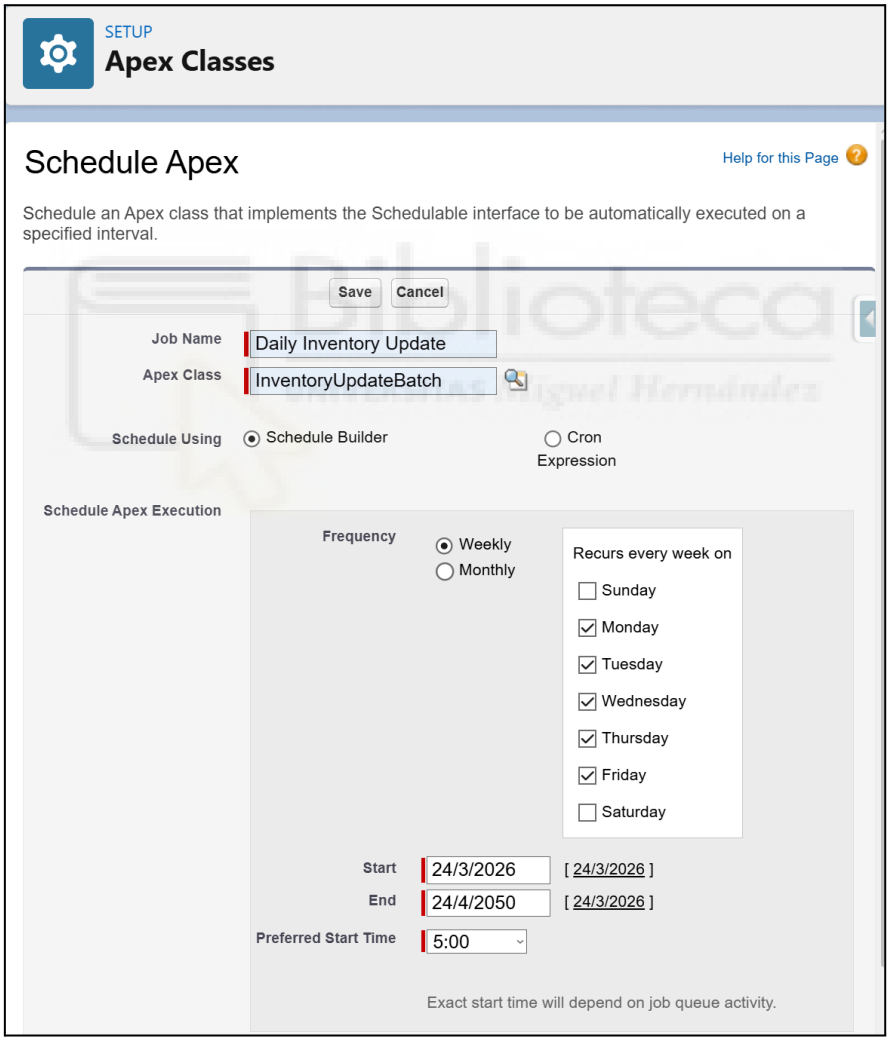

Figura 4.38: Pantalla de configuración para programar una clase Apex

Capítulo 5
Conclusiones y trabajo futuro

5.1.- CONCLUSIONES

El presente Trabajo de Fin de Grado ha permitido diseñar, desarrollar e implantar un sistema CRM/ERP funcional sobre la plataforma Salesforce, adaptado a las necesidades de una empresa ficticia de venta de productos tecnológicos. A lo largo del proyecto se han abordado todas las fases del ciclo de vida del software, desde la captura de requisitos hasta la implantación y pruebas, pasando por el diseño arquitectónico y la implementación técnica.

5.1.1.- Cumplimiento de los objetivos principales

El proyecto ha alcanzado satisfactoriamente los cinco objetivos principales planteados en el apartado 1.3.1. En primer lugar, se ha proporcionado una explicación sobre qué es Salesforce, incluyendo su plataforma de aprendizaje (Trailhead), su tienda de aplicaciones

(AppExchange) y una comparativa frente a cinco alternativas del mercado (Microsoft Dynamics 365, Odoo, SAP, Zoho y HubSpot), lo cual ha quedado documentado en el Capítulo 2.

En segundo lugar, se han descrito las diferentes industrias y áreas de negocio en las que Salesforce puede aplicarse de forma estándar, haciendo especial hincapié en las tres nubes utilizadas: Sales Cloud, Service Cloud y Experience Cloud, todas ellas detalladas en el Capítulo 3.

En tercer lugar, se ha desarrollado una aplicación funcional completa, implementada sobre una Developer Edition de Salesforce, que cubre los procesos de venta (gestión de Leads, Oportunidades, Presupuestos y Pedidos), soporte técnico (gestión de Casos y colas especializadas) y la exposición de datos a clientes externos mediante un portal web. La implementación ha sido detallada en el Capítulo 4.

En cuarto lugar, se han utilizado efectivamente las tres nubes comprometidas. Sales Cloud gestiona el ciclo comercial completo, desde la captación del Lead hasta la generación del pedido. Service Cloud proporciona la infraestructura de atención al cliente con Cases, Queues, Email-to-Case y Knowledge Base. Experience Cloud expone un portal de cliente autenticado donde los usuarios externos pueden consultar pedidos, seguimiento de envíos y artículos de conocimiento.

En quinto y último lugar, se han incorporado características propias de un sistema ERP que Salesforce no ofrece de forma nativa, concretamente la gestión de inventario y el seguimiento de envíos mediante objetos personalizados, integrado con una API externa de seguimiento logístico.

5.1.2.- Cumplimiento de los objetivos secundarios

Respecto a los objetivos secundarios definidos en el apartado 1.3.2, el cumplimiento ha sido parcial. Se ha cumplido satisfactoriamente el objetivo de desarrollar funcionalidades que requieren código. Se han implementado seis clases Apex y un trigger, cubriendo lógica de validación, integración con API externa, controladores para componentes LWC y un proceso batch programado. La integración con la API REST externa de seguimiento de envíos mediante Named Credentials y callouts HTTP asíncronos demuestra la capacidad de conectar Salesforce con sistemas de terceros de forma segura y escalable.

También se ha cumplido el objetivo de utilizar herramientas y extensiones no oficiales para mejorar la experiencia de desarrollo. Salesforce Inspector Reloaded se ha empleado durante todo el proceso de desarrollo para validar consultas SOQL, inspeccionar la

estructura de objetos y acceder a registros de depuración, tal como se describe en el apartado 3.4.4.

Sin embargo, el objetivo de explorar y aplicar algún producto de Inteligencia Artificial de Salesforce no ha sido cubierto. Aunque en el planteamiento inicial se contempló la posibilidad de integrar dichos productos, la complejidad dado el alcance técnico y temporal del proyecto junto a las limitaciones de los entornos de desarrollo, donde no se cuenta con todas las funcionalidades de Inteligencia Artificial, hicieron inviable su inclusión. No obstante, este punto queda recogido como una línea de trabajo futuro en el apartado 5.2.

5.1.3.- Cumplimiento de los objetivos personales

En cuanto a los objetivos personales (apartado 1.3.3), el proyecto ha servido al autor para consolidar y demostrar los conocimientos adquiridos a lo largo de su carrera profesional sobre la plataforma Salesforce. Además, el proyecto ha dado pie a explorar funcionalidades que el autor no había utilizado previamente en entornos profesionales, como Experience Cloud Builder para la creación del portal de cliente o el componente Knowledge para la gestión de artículos públicos.

Estas experiencias han ampliado su dominio de la plataforma más allá de los roles que había desempeñado en proyectos anteriores.

5.1.4.- Sobre las herramientas de CI/CD

En el apartado 3.4.3 se describieron las herramientas de integración y despliegue continuos (CI/CD) habituales en el ecosistema Salesforce. Aunque el desarrollo del proyecto se realizó mediante despliegues manuales con Salesforce CLI desde Visual Studio Code, las herramientas de CI/CD cobran su verdadero valor en equipos de desarrollo con múltiples entornos y flujos de integración paralelos, un escenario que excede el alcance de este trabajo.

No obstante, la descripción teórica incluida en el Capítulo 3 refleja el conocimiento del autor sobre estas prácticas y su importancia en proyectos profesionales reales.

5.1.5.- Valoración global

A nivel global, el proyecto demuestra que Salesforce es una plataforma viable y competitiva para la implementación de sistemas CRM/ERP en empresas medianas como

IT Solutions. La combinación de funcionalidad estándar con desarrollo a medida permite construir soluciones funcionales, escalables y mantenibles sin necesidad de gestionar infraestructura propia.

El modelo multitenant, las actualizaciones automáticas trimestrales, la extensibilidad mediante AppExchange y la amplia documentación y recursos de aprendizaje hacen de Salesforce una elección sólida para empresas que buscan centralizar sus procesos de negocio en una plataforma cloud.

El código desarrollado sigue las buenas prácticas del ecosistema: bulkificación, patrón de un trigger por objeto, separación de responsabilidades y tests positivos y negativos, alcanzando una cobertura superior al 75% exigido por la plataforma para despliegues en producción.

5.2.- POSIBLES DESARROLLOS FUTUROS

A partir del estado actual de la aplicación, se pueden identificar múltiples líneas de evolución que aportarían al sistema una mayor funcionalidad, robustez y valor.

5.2.1.- Integración de Inteligencia Artificial con Agentforce

La línea de trabajo futuro más relevante es la incorporación de productos de Inteligencia Artificial de Salesforce. A fecha de redacción de este trabajo, Salesforce ha renombrado sus productos principales, Sales Cloud y Service Cloud, como Agentforce Sales y Agentforce Service, lo que evidencia la apuesta de la compañía por la IA conversacional y generativa. En concreto, se podrían explorar las siguientes funcionalidades:

- **Einstein Lead Scoring** permite asignar una puntuación predictiva a cada Lead basándose en datos históricos de conversión, lo que ayudaría a los comerciales a priorizar aquellos con mayor probabilidad de convertirse en oportunidades reales

- **Agentforce**, conectado a la base de conocimiento, podría utilizarse para la generación automática de respuestas sugeridas en casos de soporte, reduciendo el tiempo de primera respuesta y aumentando la eficiencia del equipo de atención al cliente

- **Einstein Prediction Builder** permitiría crear modelos predictivos personalizados directamente desde el Setup de Salesforce, sin necesidad de conocimientos avanzados de ciencia de datos. Un caso de uso sería predecir la probabilidad de

cierre de una oportunidad en función de su historial de etapas, el tiempo medio de permanencia en cada fase y el comportamiento de oportunidades similares anteriores.

5.2.2.- Pipeline de CI/CD automatizado

Aunque en el presente trabajo los despliegues se han realizado de forma manual, la implementación de un pipeline de CI/CD completo sería el siguiente paso natural para un entorno de desarrollo profesional. Este pipeline podría incluir la ejecución de análisis estático de código Apex, la ejecución de todos los tests unitarios en un entorno limpio, y el despliegue automático al entorno de destino (sandbox o producción)

La implementación se podría realizar mediante GitHub Actions con ficheros YAML que definan los pasos del workflow, aprovechando la capacidad del Salesforce CLI y las Scratch Orgs como entornos de validación.

5.2.3.- Módulo de facturación

IT Solutions actualmente gestiona el estado de pago de los pedidos mediante un campo personalizado que se actualiza manualmente. Una evolución natural sería implementar un módulo de facturación que genere automáticamente facturas en formato PDF al activar un pedido, se integre con una pasarela de pago para el cobro automatizado y registre los pagos parciales y totales vinculados a cada factura

Esta funcionalidad podría implementarse mediante objetos personalizados (Invoice__c, Payment__c, etc) y Flows o código Apex que automaticen el proceso.

5.2.4.- Mejoras en el portal de cliente (Experience Cloud)

El portal actual cubre las funcionalidades básicas de consulta de pedidos, seguimiento de envíos y gestión de casos. Las mejoras futuras podrían incluir un catálogo de productos consultable desde el portal con la posibilidad de solicitar presupuestos directamente, un formulario de devoluciones que genere automáticamente un caso de dicho tipo vinculado al pedido original, un sistema de valoraciones y comentarios sobre productos adquiridos y la posibilidad de descargar facturas en PDF directamente desde el portal.

Bibliografía

[1] The Complete History of CRM
Salesforce / https://www.salesforce.com/ap/hub/crm/the-complete-crm-history/
Fecha de consulta: 09-Mar-2024

[2] The History of CRM From the 1950s to Today
Bianca Caballero, Jess Pingrey / https://fitsmallbusiness.com/history-of-crm/
Fecha de consulta: 09-Mar-2024

[3] La evolución de los sistemas ERP: Una perspectiva histórica
AppMaster / https://appmaster.io/es/blog/evolucion-de-los-sistemas-erp
Fecha de consulta: 14-Abr-2025

[4] Historia del ERP: origen, línea de tiempo y futuro
Javier Beleni /
https://www.calipso.com/articulos/historia-del-erp-origen-linea-de-tiempo-y-futuro/
Fecha de consulta: 14-Abr-2025

[5] What is ERP?
SAP / https://www.sap.com/products/erp/what-is-erp.html
Fecha de consulta: 09-Mar-2024

[6] Customer Centricity World Series
CCWS / https://customercentricityworldseries.com/
Fecha de consulta: 09-Mar-2024

[7] Salesforce Ranked #1 CRM Provider for 11th Consecutive Year
Salesforce /
https://www.salesforce.com/news/stories/idc-crm-market-share-ranking-2024/
Fecha de consulta: 14-Feb-2025

[8] All Salesforce products.
Salesforce /
https://www.salesforce.com/eu/products/
https://www.salesforce.com/eu/campaign/sem/salesforce-products/
Fecha de consulta: 14-Feb-2025

[9] Salesforce delivers Customer Success.
Salesforce / https://www.salesforce.com/customer-stories/
Fecha de consulta: 14-Feb-2025

[10] Web Components
Mozilla Developer Network /
https://developer.mozilla.org/en-US/docs/Web/API/Web_components
Fecha de consulta: 15-Feb-2025

[11] Introducing Lightning Web Components
Christophe Coenraets /
https://developer.salesforce.com/blogs/2018/12/introducing-lightning-web-components
Fecha de consulta: 15-Feb-2025

[12] Workflow Rules & Process Builder End of Support
Salesforce / https://help.salesforce.com/s/articleView?id=001096524&type=1
Fecha de consulta: 15-Feb-2025

[13] CRM Comparison: 10 Best CRM Software in 2025
Garima Behal / https://clickup.com/blog/crm-comparison/
Fecha de consulta: 14-Abr-2025

[14] Build for the future on the Salesforce Platform
Salesforce / https://www.salesforce.com/platform/
Fecha de consulta: 14-Abr-2025

[15] Microsoft Dynamics 365 - Información general
Microsoft / https://www.microsoft.com/es-es/dynamics-365/
Fecha de consulta: 14-Abr-2025

[16] What is Dynamics 365?
Microsoft /
https://learn.microsoft.com/en-us/dynamics365/get-started/intro-crossapp-index
Fecha de consulta: 14-Abr-2025

[17] Odoo - All Apps
Odoo / https://www.odoo.com/es_ES/page/all-apps
Fecha de consulta: 14-Abr-2025

[18] Github Odoo
Odoo / https://github.com/odoo/odoo
Fecha de consulta: 14-Abr-2025

[19] Technical Architecture of Odoo and Deployment Structure at Server
Shivam Sahu /
https://erpsolutions.oodles.io/blog/odoo-architecture-technical-deployment/
Fecha de consulta: 14-Abr-2025

[20] SAP Business One
SAP / https://www.sap.com/products/erp/business-one.html
Fecha de consulta: 14-Abr-2025

[21] SAP S/4HANA Cloud Public Edition
SAP / https://www.sap.com/spain/products/erp/s4hana.html
Fecha de consulta: 14-Abr-2025

[22] Zoho CRM
Zoho Corporation/ https://www.zoho.com/crm/
Fecha de consulta: 14-Abr-2025

[23] Zoho Creator
Zoho Corporation / https://www.zoho.com/creator/low-code-platform.html
Fecha de consulta: 14-Abr-2025

[24] HubSpot CRM
HubSpot / https://www.hubspot.es/products/crm
Fecha de consulta: 14-Abr-2025

[25] Platform Multitenant Architecture
Steve Bobrowski /
https://architect.salesforce.com/fundamentals/platform-multitenant-architecture
Fecha de consulta: 03-Jun-2025

[26] Understand the Salesforce Architecture
Trailhead /
https://trailhead.salesforce.com/content/learn/modules/starting_force_com/starting_understanding_arch
Fecha de consulta: 03-Jun-2025

[27] Salesforce Trust
Salesforce / https://trust.salesforce.com
Fecha de consulta: 03-Jun-2025

[28] Salesforce Developer Limits and Allocations Quick Reference
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.salesforce_app_limits_cheatsheet.meta/salesforce_app_limits_cheatsheet/salesforce_app_limits_overview.htm
Fecha de consulta: 03-Jun-2025

[29] Salesforce Editions
Salesforce /
https://help.salesforce.com/s/articleView?id=xcloud.overview_edition.htm&type=5
Fecha de consulta: 21-Feb-2026

[30] Scratch Orgs
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_scratch_orgs.htm
Fecha de consulta: 21-Feb-2026

[31] Modelado de Datos
Trailhead / https://trailhead.salesforce.com/es/content/learn/modules/data_modeling
Fecha de consulta: 21-Feb-2026

[32] Object Relationships Overview
Salesforce /
https://help.salesforce.com/s/articleView?id=platform.overview_of_custom_object_relationships.htm&type=5
Fecha de consulta: 21-Feb-2026

[33] Platform Sharing Architecture
Salesforce /
https://architect.salesforce.com/docs/architect/fundamentals/guide/platform-sharing-architecture
Fecha de consulta: 21-Feb-2026

[34] Agentforce Sales (formerly Sales Cloud)
Salesforce / https://www.salesforce.com/eu/sales/
Fecha de consulta: 21-Feb-2026

[35] Agentforce Service (formerly Service Cloud)
Salesforce / https://www.salesforce.com/eu/service/cloud/
Fecha de consulta: 21-Feb-2026

[36] Set Up Email-to-Case
Salesforce /
https://help.salesforce.com/s/articleView?id=service.setting_up_email-to-case.htm&type=5
Fecha de consulta: 21-Feb-2026

[37] Create a Knowledge Base with Salesforce Knowledge
Salesforce /
https://help.salesforce.com/s/articleView?id=service.knowledge_whatis.htm&type=5
Fecha de consulta: 21-Feb-2026

[38] Experience Cloud
Salesforce /
https://help.salesforce.com/s/articleView?id=experience.networks_overview.htm&type=5
Fecha de consulta: 21-Feb-2026

[39] Automate Tasks with Flows
Salesforce /
https://help.salesforce.com/s/articleView?id=platform.flow.htm&type=5
Fecha de consulta: 21-Feb-2026

[40] Salesforce Workflow Rules & Process Builder End of Support
Salesforce / https://help.salesforce.com/s/articleView?id=001096524&type=1
Fecha de consulta: 21-Feb-2026

[41] Lightning Web Components
Salesforce / https://developer.salesforce.com/docs/platform/lwc/overview
Fecha de consulta: 21-Feb-2026

[42] Introducing Lightning Web Components
Christophe Coenraets /
https://developer.salesforce.com/blogs/2018/12/introducing-lightning-web-components
Fecha de consulta: 21-Feb-2026

[43] Web Components
Mozilla / https://developer.mozilla.org/en-US/docs/Web/API/Web_components
Fecha de consulta: 21-Feb-2026

[44] Shadow DOM
Salesforce /
https://developer.salesforce.com/docs/platform/lwc/guide/create-dom.html
Fecha de consulta: 21-Feb-2026

[45] Communicate Between Lightning Web Components
Trailhead /
https://trailhead.salesforce.com/content/learn/projects/communicate-between-lightning-web-components
Fecha de consulta: 21-Feb-2026

[46] Work with Salesforce Data
Salesforce / https://developer.salesforce.com/docs/platform/lwc/guide/data.html
Fecha de consulta: 21-Feb-2026

[47] HTML Template Directives
Salesforce /
https://developer.salesforce.com/docs/platform/lwc/guide/reference-directives.html
Fecha de consulta: 24-Feb-2026

[48] Salesforce Lightning Design System
Salesforce /
https://www.lightningdesignsystem.com/2e1ef8501/p/85bd85-lightning-design-system-2
Fecha de consulta: 22-Feb-2026

[49] Get Started with Lightning Base Components
Salesforce /
https://developer.salesforce.com/docs/platform/lightning-component-reference/guide/get-started.html
Fecha de consulta: 23-Feb-2026

[50] What is Apex?
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm
Fecha de consulta: 21-Feb-2026

[51] Introduction to SOQL and SOSL
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_sosl_intro.htm
Fecha de consulta: 22-Feb-2026

[52] Salesforce CLI
Salesforce / https://developer.salesforce.com/tools/salesforcecli
Fecha de consulta: 21-Feb-2026

[53] Salesforce Extensions for Visual Studio Code
Salesforce /
https://developer.salesforce.com/docs/platform/sfvscode-extensions/guide/install
Fecha de consulta: 21-Feb-2026

[54] Salesforce Extension Pack
Microsoft /
https://marketplace.visualstudio.com/items?itemName=salesforce.salesforcedx-vscode
Fecha de consulta: 21-Feb-2026

[55] How Salesforce Developer Experience (DX) Tooling Changes the Way You Work
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_intro.htm
Fecha de consulta: 22-Feb-2026

[56] How to Exclude Source When Syncing
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_exclude_source.htm
Fecha de consulta: 22-Feb-2026

[57] GitHub Actions
GitHub / https://github.com/features/actions
Fecha de consulta: 22-Feb-2026

[58] Salesforce Code Analyzer
Salesforce /
https://developer.salesforce.com/docs/platform/salesforce-code-analyzer/guide/ci-cd.html
Fecha de consulta: 22-Feb-2026

[59] Jenkins - Build great things at any scale
Jenkins / https://www.jenkins.io
Fecha de consulta: 22-Feb-2026

[60] DevOps Center Developer Guide
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.devops_center_dev.meta/devops_center_dev/devops_center_dev_overview.htm
Fecha de consulta: 22-Feb-2026

[61] Salesforce Inspector Reloaded
Thomas Prouvot / https://tprouvot.github.io/Salesforce-Inspector-reloaded/
Fecha de consulta: 22-Feb-2026

[62] Apex Code Best Practices
Salesforce / https://developer.salesforce.com/ja/wiki/apex_code_best_practices
Fecha de consulta: 23-Feb-2026

[63] Trigger and Bulk Request Best Practices
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers_bestpract.htm
Fecha de consulta: 23-Feb-2026

[64] Apex Best Practices: The 15 Apex Commandments
Salesforce /
https://developer.salesforce.com/blogs/developer-relations/2015/01/apex-best-practices-15-apex-commandments
Fecha de consulta: 23-Feb-2026

[65] Apex Enterprise Patterns: Service Layer
Trailhead / https://trailhead.salesforce.com/content/learn/modules/apex_patterns_sl
Fecha de consulta: 23-Feb-2026

[66] Apex Enterprise Patterns: Domain & Selector Layers
Trailhead / https://trailhead.salesforce.com/content/learn/modules/apex_patterns_dsl
Fecha de consulta: 23-Feb-2026

[67] Apex Enterprise Patterns – Separation of Concerns
Andew Fawcett /
https://andyinthecloud.com/2012/11/16/apex-enterprise-patterns-separation-of-concerns/
Fecha de consulta: 23-Feb-2026

[68] fflib-apex-common
apex-enterprise-patterns / https://github.com/apex-enterprise-patterns/fflib-apex-common
Fecha de consulta: 23-Feb-2026

[69] Code Coverage Best Practices
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_code_coverage_best_pract.htm
Fecha de consulta: 23-Feb-2026

[70] Testing Best Practices
Salesforce /
https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_testing_best_practices.htm
Fecha de consulta: 23-Feb-2026

[71] Desarrollo iterativo e incremental
Proyectos Agiles / https://proyectosagiles.org/desarrollo-iterativo-incremental/
Fecha de consulta: 19-Mar-2026

Anexo I
Casos de Uso

En el presente anexo se incluye la descripción detallada de los casos de uso de la aplicación:

Tabla AI.1.- Caso de uso 1: Registrar nuevo Lead

CU-01	Registrar nuevo Lead
Actor principal	Comercial
Precondiciones	El comercial tiene acceso al sistema
Flujo principal	1. El comercial accede a la pestaña "Leads" 2. Hace clic en "Nuevo" 3. Introduce los datos del Lead: nombre, empresa, email, teléfono y producto de interés (Componentes, Periféricos, Software o Soluciones a medida) 4. Guarda el Lead 5. El sistema asigna automáticamente el Lead a la cola correspondiente según el producto de interés seleccionado
Postcondiciones	El Lead queda registrado y asignado a la cola correspondiente
Flujos alternativos	3a. Si falta información obligatoria, el sistema muestra un error y no permite guardar

Tabla AI.2.- Caso de uso 2: Convertir Lead en Oportunidad

CU-02	Convertir Lead en Oportunidad
Actor principal	Comercial
Precondiciones	Existe un Lead cualificado en el sistema
Flujo principal	1. El comercial accede al Lead 2. Hace clic en "Convertir" 3. El sistema crea automáticamente una Cuenta (si no existe), un Contacto y una Oportunidad 4. El comercial confirma la conversión 5. El Lead queda marcado como "Converted"
Postcondiciones	Se crea una Oportunidad vinculada a la Cuenta y Contacto correspondientes.
Flujos alternativos	3a. Si la empresa ya existe como Cuenta, el sistema permite asociar el Contacto a esa Cuenta existente.

Tabla AI.3.- Caso de uso 3: Generar Presupuesto desde Oportunidad

CU-03	Generar Presupuesto desde Oportunidad
Actor principal	Comercial
Precondiciones	Existe una Oportunidad con productos asociados
Flujo principal	1. El comercial accede a la Oportunidad 2. Añade productos mediante el componente LWC productSearchAndAdd, indicando cantidad y descuento para cada línea 3. Hace clic en "Crear Presupuesto" (New Quote) 4. El sistema genera un Quote con los productos de la Oportunidad 5. El comercial revisa precios y descuentos 6. Si el descuento de la oportunidad supera el 15%, el flow Discount_Approval_Process marca los campos Requires_Approval__c y Approval_Status__c, y el Approval Process solicita la aprobación del Responsable de Ventas 7. Una vez aprobado, el comercial actualiza el estado del Quote a "Accepted"
Postcondiciones	Existe un Quote asociado a la Oportunidad, con estado Accepted.
Flujos alternativos	6a. Si el Responsable rechaza el descuento, el comercial debe ajustar las condiciones antes de reenviar la solicitud.

Tabla AI.4.- Caso de uso 4: Crear Pedido desde Presupuesto

CU-04	Crear Pedido desde Presupuesto
Actor principal	Comercial
Precondiciones	Existe un Quote con estado Accepted y sincronizado con la Oportunidad.
Flujo principal	1. El comercial cambia el estado de la Oportunidad a "Closed Won" 2. El sistema valida que existe un Quote sincronizado con estado "Accepted" (OpportunityTriggerHandler) 3. El comercial accede al Quote y hace clic en la Quick Action "New Order" 4. El Screen Flow Order_Creation muestra una pantalla de confirmación con los productos del presupuesto 5. El comercial confirma la creación 6. El sistema crea el Order con sus OrderItems, mapeando productos, cantidades y precios desde el Quote 7. El flow Update_Inventory_on_Order actualiza automáticamente el inventario (Available_Stock__c y Reserved_Stock__c) 8. El sistema envía un email de confirmación al comercial con el número del pedido creado 9. Si se produce un error, el flow ejecuta un rollback automático
Postcondiciones	Se crea un Order con sus líneas y se actualiza el inventario.
Flujos alternativos	2a. Si no hay Quote sincronizado con estado Accepted, el sistema muestra un error y no permite cerrar la Oportunidad. 9a. Si el rollback se ejecuta, el sistema muestra una pantalla de error y no se crea ningún registro.

Tabla AI.5.- Caso de uso 5: Abrir Caso de Soporte

CU-05	Abrir Caso de Soporte
Actor principal	Técnico de Soporte o Cliente Portal
Precondiciones	El actor tiene acceso al sistema
Flujo principal	1. El actor accede a la opción "Nuevo Caso" 2. Introduce el asunto, descripción, prioridad y tipo de caso (Hardware, Software o Billing) 3. Opcionalmente vincula el caso a un pedido existente (Related_Order__c) 4. Si es un cliente portal, el sistema asocia automáticamente el caso a su Cuenta 5. Guarda el caso
Postcondiciones	El caso queda registrado en el sistema. Un técnico de soporte lo toma de la cola correspondiente para su gestión.
Flujos alternativos	1a. El caso se crea automáticamente si el cliente envía un email a la dirección configurada en Email-to-Case.

Tabla AI.6.- Caso de uso 6: Resolver Caso de Soporte

CU-06	Resolver Caso de Soporte
Actor principal	Técnico de Soporte
Precondiciones	Existe un caso asignado al técnico
Flujo principal	1. El técnico accede al caso 2. Consulta artículos de conocimiento relacionados 3. Comunica la solución al cliente 4. Registra las acciones realizadas en el caso 5. Cambia el estado del caso a "Closed" 6. El sistema calcula automáticamente Resolution_Time__c (horas entre creación y cierre)
Postcondiciones	El caso queda cerrado con su tiempo de resolución registrado.
Flujos alternativos	5a. Si el cliente no queda conforme con la solución, el técnico reabre el caso cambiando el estado a Working.

Tabla AI.7.- Caso de uso 7: Consultar Pedidos desde Portal de Cliente

CU-07	Consultar Pedidos desde Portal de Cliente
Actor principal	Cliente Portal
Precondiciones	El cliente está autenticado en el portal de Experience Cloud
Flujo principal	1. El cliente accede a la sección "Mis Pedidos" 2. El componente orderTrackingList muestra sus pedidos con número, estado, fecha, importe y estado de pago 3. El cliente hace clic en un pedido 4. El componente orderTrackingDetail muestra el detalle del envío: transportista, número de seguimiento, estado y fecha estimada de entrega 5. Si los datos de tracking tienen más de una hora de antigüedad, el componente refresca automáticamente la información consultando la API del proveedor de logística
Postcondiciones	El cliente ha consultado el estado actualizado de sus pedidos
Flujos alternativos	4a. Si el pedido no tiene registro de Shipment_Tracking__c asociado, no se muestra información de envío.

Tabla AI.8.- Caso de uso 8: Crear Artículo de Conocimiento

CU-08	Crear Artículo de Conocimiento
Actor principal	Técnico de Soporte
Precondiciones	El técnico tiene permisos para crear artículos
Flujo principal	1. El técnico accede a "Knowledge" 2. Crea un nuevo artículo de tipo FAQ 3. Introduce la pregunta y la respuesta 4. Publica el artículo 5. El artículo queda disponible para consulta desde el portal de clientes
Postcondiciones	El artículo queda publicado y accesible.
Flujos alternativos	-

Tabla AI.9.- Caso de uso 9: Consultar Historial de Oportunidad

CU-09	Consultar Historial de Oportunidad
Actor principal	Comercial
Precondiciones	Existe una Oportunidad con cambios de etapa registrados
Flujo principal	1. El comercial accede al detalle de una Oportunidad 2. El componente opportunityTimeline muestra la línea temporal con todos los cambios de etapa: fecha, etapa, importe y probabilidad en cada punto 3. El comercial identifica visualmente el progreso de la negociación
Postcondiciones	El comercial ha consultado el historial de la oportunidad.
Flujos alternativos	-