Trabajo Fin de Grado

Ingeniería de las Tecnologías de las Comunicaciones

Salesforce como CRM en la Administración de Fincas

Autor: Sara Cabeza Muñoz

Tutor: Antonio Jesús Sierra Collado

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado
Ingeniería de las Tecnologías de las Telecomunicaciones

Salesforce como CRM en la Administración de Fincas

Autor:
Sara Cabeza Muñoz

Tutor:
Antonio Jesús Sierra Collado

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Trabajo Fin de Grado: Salesforce como CRM en la Administración de Fincas

Autora: Sara Cabeza Muñoz

Tutor: Antonio Jesús Sierra Collado

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

Agradecimientos

En primer lugar, me gustaría expresar mi gratitud a mi tutor, Antonio Jesús, por su paciencia y su tiempo.

A mi familia, sobre todo a mis padres, por su apoyo incondicional. A mi hermana, por creer en mí en todo momento. A Pablo, por estar siempre a mi lado y animarme cuando lo necesitaba.

Este trabajo refleja todo lo que he aprendido tanto en clase como en el mundo laboral, agradecer también a mis compañeros por sus consejos y enseñanzas a lo largo del camino.

Resumen

Este Trabajo de Fin de Grado estudia el desarrollo e implementación de un sistema en Salesforce para su uso como herramienta de gestión de relaciones con clientes en el ámbito de la administración de fincas. Este sector en auge requiere soluciones eficientes para la gestión de relaciones con los clientes y la optimización de procesos internos y, dada la complejidad de la gestión inmobiliaria y las nuevas demandas del mercado, la utilización de sistemas Customer Relationship Management (CRM) como modelo de negocio mejora no sólo la atención al cliente y la gestión comercial, sino también el proceso de venta y la productividad.

En primer lugar, se presenta una introducción a los conceptos relacionados con CRM, a la que acompañan la descripción de los objetivos y la estructura del proyecto. Después, la sección dedicada al estado del arte ofrecerá una visión general acerca de CRM, centrándose en las soluciones basadas en la nube y analizando a su vez las principales plataformas del mercado actual: Salesforce, SAP y Oracle. Aquí, nos detendremos especialmente en Salesforce por ser la elegida para el caso de estudio que se llevará a cabo posteriormente.

Habiendo seleccionado Salesforce como la herramienta a emplear en nuestro proyecto, se considera oportuno profundizar en su ecosistema de forma que se cubran aspectos clave como la configuración de datos, la automatización de procesos y el desarrollo de funcionalidades a través de herramientas como Apex o componentes Lightning, que permiten adaptar la plataforma a las necesidades específicas de la empresa.

La siguiente sección gira en torno al elemento principal del proyecto: el caso de estudio. Por tanto, se detallará la creación e implementación de una solución Salesforce para la empresa “AdminFincas”, compañía ficticia dedicada a la gestión de propiedades. El contexto, los objetivos y la metodología de trabajo empleada se definirán primeramente para dar paso al análisis funcional y técnico donde se explican la arquitectura del sistema y las funcionalidades específicas que necesita nuestra empresa. Después, se presentará el desarrollo de la solución señalando a su vez las herramientas utilizadas y las decisiones técnicas que guiaron el proceso de ejecución. También se muestran las pruebas que se realizaron para su validación. Finalmente, se revelan las conclusiones extraídas de los datos que obtuvimos y se plantean algunas líneas de trabajo de cara al futuro.

A lo largo del estudio, se presta especial atención a cómo las funciones de Salesforce (configuración de datos, automatización de procesos, capacidades de personalización, etc.) pueden aprovecharse para satisfacer las necesidades específicas de las empresas de gestión de propiedades. Este proyecto demuestra cómo una solución CRM bien implementada puede mejorar el servicio al cliente, agilizar los procesos de ventas y mejorar la productividad general en el sector de la gestión de propiedades.

Palabras clave: CRM, Salesforce, administración de fincas, automatización de procesos

Abstract

This Final Degree Project analyses the development and implementation of a Salesforce system to be used as a customer relationship management tool in the field of property management. This growing sector requires efficient solutions for customer relationship management and the optimisation of internal processes and given the complexity of real estate management and the new demands of the market, the use of Customer Relationship Management (CRM) systems as a business model improves not only customer service and commercial management, but also the sales process and productivity.

First, an introduction to CRM-related concepts is presented, followed by a description of the objectives and structure of the project. Then, the section dedicated to the state of the art will provide an overview of CRM, focusing on cloud-based solutions and analysing the main platforms on the market today: Salesforce, SAP and Oracle. Here, we will focus in particular on Salesforce as it is the one chosen for the case study that will follow.

Having selected Salesforce as the tool to be used in our project, it is considered appropriate to delve deeper into its ecosystem in order to cover key aspects such as data configuration, process automation and the development of functionalities through tools such as Apex or Lightning components, which allow the platform to be adapted to the specific needs of the company.

The next section revolves around the main element of the project: the case study. Therefore, it will detail the creation and implementation of a Salesforce solution for the company 'AdminFincas', a fictitious company dedicated to property management. The context, the objectives and the working methodology used will be defined first to give way to the functional and technical analysis where the architecture of the system and the specific functionalities needed by our company will be explained. Then, the development of the solution will be presented, pointing out the tools used and the technical decisions that guided the implementation process. The tests that were carried out for validation are also shown. Finally, the conclusions drawn from the data we obtained are revealed and some lines of work for the future are proposed.

Throughout the study, particular attention is paid to how Salesforce features (data configuration, process automation, personalization capabilities, etc.) can be exploited to meet the specific needs of property management companies. This project demonstrates how a well-implemented CRM solution can improve customer service, streamline sales processes, and improve overall productivity in the property management industry.

Keywords: CRM, Salesforce, property management, process automation

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

En este primer apartado se expondrán los temas a tratar en este trabajo de forma introductoria, se analizará el concepto de CRM (*Customer Relationship Management*, o Gestión de Relaciones con los Clientes), señalando la importancia e impacto que actualmente tienen las diferentes soluciones CRM para las empresas y los problemas que estas solventan, atendiendo particularmente a mejorar la gestión de una administradora de fincas.

Además, con el fin de mostrar un alcance global del proyecto, se plantearán una serie de objetivos que se esperan lograr una vez haya finalizado el estudio planteado en este trabajo. Así como también, se detallará la estructura que seguirá la memoria para una mejor comprensión.

1.1 *Customer Relationship Management*

La Gestión de las Relaciones con Clientes (CRM), se refiere al conjunto de prácticas, estrategias comerciales y tecnologías orientadas a la relación con el cliente. Se trata de administrar y analizar las relaciones de una empresa con sus clientes (y futuros clientes potenciales) para entablar una buena relación con ellos, con objetivo de impulsar el crecimiento del negocio.

La llegada de los CRM ha revolucionado la forma en la que las empresas interactúan con sus clientes, manejan sus datos y optimizan sus procesos de ventas. Los sistemas CRM, como Salesforce, proporcionan una plataforma centralizada para la gestión, el seguimiento y la organización de las relaciones con los clientes, mejorando así el servicio al cliente y el rendimiento de la empresa.

Salesforce, uno de los principales proveedores de software CRM basado en la nube, proporciona una experiencia única, ya que su uso permite crear relaciones más significativas con los clientes. Ofrece un completo conjunto de funciones diseñadas para el servicio, el marketing y las ventas, garantizando una gestión de contactos sin complicaciones, pues simplifica la gestión de presupuestos y proporciona inteligencia integrada para mejorar la productividad de la empresa.

Uno de los tipos de empresa que utilizan software CRM son las administradoras de fincas, pues el aumento de comunidades de vecinos y la alta competencia en el sector, han llevado a los administradores de fincas a buscar herramientas que mejoren su rentabilidad y eficacia.

En la actualidad, uno de los principales problemas a los que se enfrentan los administradores de fincas es la cantidad de gestiones que tienen que realizar y, el uso de este software les proporciona una experiencia centrada en el cliente, la cual es una parte fundamental en el modelo de estos tipos de negocio.

Con este proyecto, se pretende abordar e implementar una solución CRM basada en Salesforce que se adapte a las necesidades específicas de una empresa de gestión de viviendas para comunidades de vecinos. La flexibilidad que presenta Salesforce permite crear un sistema CRM utilizando tanto funciones por defecto como personalizadas, ofreciendo así una solución a medida que agiliza los procesos, mejora el servicio al cliente y aumenta los ingresos. Por lo tanto, el sistema se diseñará para gestionar eficazmente los datos de los usuarios, optimizar los procesos de ventas y marketing y desarrollar una mejor relación con el cliente en toda la organización.

Como resultado, este trabajo de fin de grado proporcionará una guía sobre el desarrollo de un sistema CRM con Salesforce, demostrando su aplicación en un escenario real, en nuestro caso el contexto de una empresa que gestiona viviendas para comunidades de vecinos. El objetivo es mostrar cómo un sistema CRM bien implementado puede mejorar las operaciones comerciales, contribuir a una mejor relación con los clientes y, en última instancia, impulsar el crecimiento de la empresa.

1.2 Objetivos

El objetivo principal de este trabajo es desarrollar un sistema CRM basado en Salesforce para una empresa administradora de fincas que le permita gestionar su modelo de negocio desde un único entorno de trabajo. Para ello, en el desarrollo de este caso de estudio se abordarán diferentes objetivos:

- Definir el concepto de CRM, sus diferentes tipos y detallar sus características principales.
- Exponer las ventajas y desventajas asociadas al uso de un CRM.
- Realizar un análisis de los CRMs más destacados en el mercado actual.
- Analizar en profundidad el CRM de Salesforce y justificar el uso de esta tecnología en la solución propuesta.
- Desarrollar el caso de estudio previamente definido.

El propósito principal de esta implementación del sistema en Salesforce es centralizar la información, lo cual facilita su seguimiento y mejora la eficiencia. Además, se espera que tenga un impacto significativo en las ventas y expansión de clientes en la empresa administradora de fincas; así como también un impacto positivo en la experiencia del cliente, pues ofrece una experiencia personalizada y transparente.

1.3 Estructura de la memoria

Tal como se mencionaba en la introducción, una vez que los objetivos que se pretenden alcanzar han sido establecidos, es necesario organizar el trabajo en diferentes capítulos, manteniendo un hilo conductor coherente para su desarrollo.

Primero, los usuarios necesitan comprender los conceptos y tecnologías involucrados, así como sus respectivas ventajas e inconvenientes. Sobre esta base, el siguiente paso implica diseñar un caso de estudio para abordar los objetivos Determinados. Posteriormente, la solución elegida debe ser implementada, configurada y puesta a prueba. Una vez completadas estas etapas, podremos extraer conclusiones y exponer posibles mejoras del proyecto.

Para aclarar, abordaremos todos estos temas siguiendo los siguientes puntos, que abarcarán todo el alcance del proyecto:

- Estudio del estado del arte: se definirá qué es un CRM y se analizarán los diferentes proveedores de CRM que existen actualmente. Entre estos, se explicará detalladamente qué es Salesforce, ya que será la tecnología que se implementará en el caso de estudio.
- Análisis del caso de estudio: se expondrán las necesidades de la empresa administradora de fincas y se realizará el análisis de casos de uso. Además, se expondrá detalladamente cómo se va a desarrollar la solución propuesta.
- Implementación de la solución: se detallarán las configuraciones y procesos aplicados, para que el usuario se familiarice con cada concepto de forma práctica.
- Puesta en marcha, pruebas de funcionamiento y demostración de los resultados: se expondrá un plan de pruebas para lograr nuestro propósito, que la solución funcione como lo esbozado en el apartado de diseño.
- Toma de conclusiones: se realizará un análisis global del proyecto, estableciendo ventajas e inconvenientes, así como proyectar posibles líneas futuras.

2 ESTADO DEL ARTE

En este apartado, se describirá en primer lugar qué es un CRM, junto con sus diferentes tipos y características generales, así como los CRM basados en la nube. Después, se entrará en detalle en la plataforma escogida para la solución propuesta en este trabajo, Salesforce. Explorando aspectos específicos como la configuración de datos en dicha plataforma CRM la automatización de procesos y el desarrollo de soluciones personalizadas.

2.1 CRM

Existen diversas definiciones sobre *Customer Relationship Management*, de entre todas ellas podemos llegar a la conclusión de que se trata de una combinación de tecnología y estrategia con el objetivo de optimizar la interacción con los clientes y mejorar la satisfacción de estos. Desde el enfoque tecnológico, el CRM consiste en una plataforma informática diseñada para registrar y analizar las interacciones entre la empresa y los usuarios, y generar informes sobre ellas, centrándose en el marketing, la gestión comercial y la atención al cliente. A nivel estratégico, se refiere al conjunto de tácticas que abarcan el análisis de las interacciones con los clientes, la anticipación de necesidades, la optimización de la rentabilidad, el aumento de las ventas y la personalización de campañas para atraer nuevos clientes y retener los actuales.

Al integrar el CRM en sus operaciones, las empresas pueden obtener una ventaja competitiva en el mercado actual al ofrecer experiencias excepcionales a los clientes que impulsan la fidelidad, la promoción y la rentabilidad a largo plazo. Su objetivo es mejorar las relaciones comerciales, ayudando a conservar a los clientes e impulsar el crecimiento de las ventas. Los sistemas CRM ayudan en la gestión de contactos, gestión de ventas productividad y más, permitiendo una visión general de cada cliente a lo largo de todo su ciclo de vida.

La idea de CRM existe desde hace mucho tiempo, en las primeras etapas del comercio, cuando era crucial mantener relaciones con compradores y vendedores. El concepto de CRM ha evolucionado desde entonces, el enfoque básico de mantenimiento de registros y contabilidad ha dado paso a la era digital de la gestión automatizada de los datos de los clientes. Los hechos clave incluyen la aparición del marketing de bases de datos en la década de 1980, el desarrollo de productos de automatización de la fuerza de ventas en la década de 1990 y la introducción de soluciones de CRM basadas en la nube a finales de la década de 1990. El CRM ha seguido evolucionando con los avances tecnológicos, lo que ha llevado a la integración de la información de redes sociales, el CRM en dispositivos móviles y el auge de la IA, además de *big data* en los sistemas de CRM.

La característica principal de un CRM en cualquier contexto es su capacidad para recopilar y organizar la información de clientes potenciales a lo largo de todas las fases del negocio, al mismo tiempo que gestiona las relaciones con los clientes ya establecidos. No obstante, al centrarnos en el CRM como una plataforma, es conveniente señalar más detalladamente sus principales características a continuación:

- Almacenamiento de información de clientes: permite almacenar datos de clientes actuales y potenciales, como nombres, direcciones, números de teléfono, interacciones con la empresa, correos electrónicos, entre otros. Esto se logra a través de bases de datos integradas en la plataforma CRM.
- Centralización de herramientas de gestión: agrupa diversas herramientas de gestión dentro de un mismo entorno de trabajo, facilitando el acceso y la operatividad dentro de la plataforma.
- Automatización de tareas repetitivas: incluye herramientas que permiten desarrollar automatizaciones para ejecutar tareas de manera programada, simplificando procesos y aumentando la eficiencia.
- Análisis y *reporting*: proporciona herramientas para generar informes y así medir el rendimiento de los servicios de la empresa, evaluar la satisfacción del cliente, identificar tendencias y oportunidades, y tomar decisiones informadas.
- *Feedback* entre empresa y cliente: dispone de mecanismos para generar comunicaciones, como correos o avisos, hacia los clientes registrados, asegurando que toda interacción entre la empresa y el cliente

quede registrada y sea accesible dentro del entorno de trabajo.

Estas características señalan la importancia de los CRM como plataformas completas que no solo se enfocan en la gestión de contactos, sino que también ofrecen un amplio espectro de funcionalidades diseñadas para optimizar la gestión de relaciones con los clientes, automatizar procesos y proporcionar información estratégica que contribuye al crecimiento y éxito de la empresa.

En función del objetivo de la empresa, el modelo CRM puede clasificarse en tres tipos principales:

- CRM operativo: Se centra en la automatización de los procesos de cara al cliente, como ventas, marketing y servicio, para mejorar la eficiencia y la experiencia del cliente.
- CRM analítico: Hace hincapié en el análisis de los datos de los clientes para fundamentar las decisiones estratégicas y fortalecer las relaciones con los clientes.
- CRM colaborativo: Tiene como objetivo mejorar la comunicación y la colaboración con los clientes a través de diversos canales, garantizando una experiencia de cliente coherente.

Estos sistemas ofrecen una variedad de ventajas y desventajas, que podemos resumir en la siguiente tabla:

Ventajas	Desventajas
Mejora de las relaciones con los clientes (mayor fidelización y retención de clientes)	Elevados costes de implantación
Mejora del rendimiento de las ventas	Requisitos de formación (Seguridad y protección de datos)
Aumento de la productividad	No es adecuado para todas las empresas

Tabla 2-1. Principales ventajas y desventajas de un CRM.

2.1.1 CRM en la Nube

El término ‘nube’ se refiere en este contexto al espacio de almacenamiento virtual de información o sistemas informáticos que los usuarios pueden utilizar sin necesidad de instalar dispositivos de hardware. El almacenamiento en la nube elimina las preocupaciones sobre las limitaciones de capacidad de almacenamiento, a diferencia de los dispositivos físicos, ya que ofrece un espacio prácticamente ilimitado para el almacenamiento de datos (las empresas pueden comprar o alquilar el espacio de almacenamiento necesario, con la flexibilidad de ampliarlo según sus necesidades, sin tener que instalar hardware).

En el apartado anterior se ha mencionado que podemos encontrar 3 tipos diferentes de CRM, sin embargo, la clasificación de los tipos de CRM no se limita únicamente a su usabilidad dentro de la empresa, sino que también se determina según su ubicación física, dando lugar a la siguiente clasificación:

- CRM Local

El CRM Local, también llamado *CRM On-Premise*, se refiere a los sistemas CRM que están alojados en los servidores físicos de una empresa (es necesario que el software se instale directamente en estos). La particularidad de este tipo de CRM es que depende del equipo informático interno de la empresa para su mantenimiento, resolución de problemas y actualizaciones. Ofrece a las empresas un control total sobre su propio entorno CRM, lo que incluye la seguridad de sus datos y opciones de personalización. Sin embargo, también supone importantes costes iniciales elevados en hardware, licencias de software y posibles ampliaciones, así como la necesidad de un equipo informático cualificado.

- CRM en la Nube

El CRM en la Nube, o *CRM Cloud*, se basa en la tecnología de *cloud computing*[1] para alojar el software en servidores remotos gestionados por el proveedor de CRM. Este modelo elimina la necesidad de que las empresas instalen el sistema CRM en ordenadores locales o cuenten con un equipo de TI dedicado al mantenimiento del software.

Estas soluciones en línea pueden denominarse software como servicio (SaaS), ya que ofrecen la flexibilidad de acceder al sistema CRM desde cualquier lugar y en cualquier momento, mediante un navegador web en cualquier dispositivo con conexión a Internet. Este tipo de CRM reduce de forma significativa los costes iniciales, facilita la escalabilidad y garantiza que las empresas siempre tengan acceso a las últimas funciones y actualizaciones de seguridad sin intervención manual.

CRM	Ventajas	Desventajas
On-Premise	Mayor control del servidor por parte de los equipos de TI.	Un corte de energía puede ocasionar retrasos en el uso del sistema. Los costos iniciales, incluyendo configuración e instalación, son elevados. Se requiere contratar a un equipo de TI para administrar el servidor y realizar mantenimiento. Menos flexible y las actualizaciones pueden ser costosas.
Cloud	Disponible las 24 horas del día, los 7 días de la semana. Accesible desde cualquier ubicación y dispositivo móvil. Requiere una inversión inicial mínima. No necesita mantenimiento del servidor y las actualizaciones son automáticas. Acompaña de manera segura el crecimiento de la empresa.	Depende de la conexión a Internet, aunque puede sincronizarse con datos offline.

Tabla 2-2. Diferencias entre CRM On-Premise y Cloud.

2.1.2 CRMs destacados en el mercado actual

Dada la gran variedad de CRMs basados en la nube disponibles para las empresas, se va a realizar un breve análisis de los más importantes del mercado actualmente para poder tomar una decisión sobre cuál es el más indicado para implantar en el proyecto. Antes de continuar, se van a definir algunos conceptos que clarificarán la comprensión del análisis:

[1] Se refiere a la disponibilidad bajo demanda de recursos de computación como servicios a través de Internet.

- Usuario: Se refiere a cualquier persona física a la que se concede acceso a la plataforma a través de una licencia determinada. Los usuarios pueden ser clientes, administradores de la plataforma o empleados de la empresa, dependiendo de la clase de licencia adquirida. Los roles y permisos asignados a cada usuario pueden variar según su función específica dentro del ecosistema CRM.
- Licencia: Una licencia representa un contrato formal entre un proveedor y un acreedor, que otorga el derecho legal a utilizar un servicio o producto específico. En el contexto de los sistemas CRM, las licencias desempeñan un papel fundamental a la hora de delimitar los niveles de acceso y las funcionalidades dentro de la plataforma. Los distintos tipos de licencias se adaptan a los diferentes requisitos de los usuarios y a las necesidades de la organización.

2.1.2.1 Salesforce

Fundada por Marc Benioff en 1999, Salesforce introdujo una plataforma basada en la nube diseñada para centralizar la información imprescindible de la empresa. Esta plataforma no sólo permite almacenar los datos de los clientes, sino que también incluye sus oportunidades de negocio, clientes potenciales e incidencias relacionadas con el servicio. El objetivo principal de esta herramienta es mejorar las relaciones con los clientes a través de una mejor comunicación entre cliente y empresa, y un acceso transparente a la información completa de los usuarios desde cualquier lugar.

El ecosistema de Salesforce se compone varias nubes, cada una proporciona funcionalidades y servicios estandarizados adaptados a necesidades empresariales específicas. Este diseño modular permite a los clientes obtener un sistema CRM enfocado exclusivamente en la nube que requiera dentro de la plataforma Salesforce.

Las principales nubes que ofrece Salesforce son:

- *Sales Cloud* (Nube de Ventas)

  Se centra en las operaciones de ventas, la generación de clientes potenciales y la gestión de contratos. Mejora la eficiencia de las ventas al permitir al personal gestionar sus carteras de clientes y ventas, así como automatizar las renovaciones de contratos.

- *Service Cloud* (Nube de Servicios)

  Dedicada al servicio al cliente, facilita la gestión de incidencias y mejora la satisfacción del cliente a través de diferentes canales de comunicación personalizables entre los clientes y la empresa, incluyendo correo electrónico, formularios web, etc.

- *Marketing Cloud* (Nube de Marketing)

  Tiene como objetivo aumentar el compromiso del cliente, mejorando el vínculo con los mismos de manera personalizada mediante una variedad de canales de comunicación, como campañas publicitarias y plataformas de redes sociales. Estas estrategias fortalecen las relaciones con los clientes, generando mayor confianza y lealtad.

- *Community Cloud* (Nube de Comunidades)

  Agiliza la comunicación entre clientes y empresa, proporcionando acceso en tiempo real a información sobre problemas, oportunidades de negocio e interacciones con posibles clientes a través de varios tipos de asociaciones.

Para utilizar esta plataforma, sólo se necesita una conexión a Internet y una licencia para cada usuario que acceda a ella. Cada una de las nubes mencionadas anteriormente se adquiere a través de su respectiva licencia, por tanto, si una empresa opta por no utilizar funciones adicionales más allá de las capacidades básicas del sistema en cualquiera de estas nubes, solo podrá acceder a las funcionalidades estándar que estas ofrecen. Este enfoque ayuda a reducir costes, especialmente para las empresas cuyas necesidades de CRM se cubren utilizando una única nube.

La gama de funcionalidades estándar disponibles para una empresa depende del tipo de acuerdo de licencia establecido con Salesforce. En la tabla siguiente se describen las distintas licencias disponibles para las dos principales nubes de Salesforce, junto con los detalles de las funcionalidades estándar asociadas a cada una:

SALES	***Starter Suite***	***Professional***	***Enterprise***	***Unlimited***	***Einstein 1 Sales***
	$25 *user*/mes	$80 *user*/mes	$165 *user*/mes	$330 *user*/mes	$500 *user*/mes
Gestión de cuentas, contactos, clientes potenciales y oportunidades	✓	✓	✓	✓	✓
Gestión avanzada de previsiones y oportunidades	✗	✗	✓	✓	✓
Compromiso de ventas e inteligencia de conversación	✗	✗	Compra disponible	✓	✓
Plan de éxito Premier	✗	Compra disponible	Compra disponible	✓	✓
IA Generativa	✗	✗	Compra disponible	Compra disponible	✓
Datos unificados	✗	✗	Compra disponible	Compra disponible	✓
Planificación de ventas, programas y colaboración	✗	✗	Compra disponible	Compra disponible	✓

Tabla 2-3. Licencias de la nube de ventas de Salesforce (web oficial, última visita: marzo 2024).

SERVICE	***Starter Suite***	***Professional***	***Enterprise***	***Unlimited***	***Einstein 1 Sales***
	$25 *user*/mes	$80 *user*/mes	$165 *user*/mes	$330 *user*/mes	$500 *user*/mes
Gestión de casos	✓	✓	✓	✓	✓
Enrutamiento omnicanal	✗	✓	✓	✓	✓
Gestión del conocimiento	✓	✗	✗	✓	✓
Contratos y derechos de servicio	✗	✓	✓	✓	✓
Plan de éxito Premier	✗	✗	Compra disponible	✓	✓
Inteligencia de servicio	✗	✗	✗	✗	✓
Voz y mensajería unificadas	✗	✗	✗	✗	✓

Tabla 2-4. Licencias de la nube de servicios de Salesforce (web oficial, última visita: marzo 2024).

2.1.2.2 SAP

Fundada en 1972 por cinco ingenieros de IBM, SAP se centró inicialmente en el desarrollo de soluciones de software, lo que permitió su expansión en diversos sectores. La base del éxito de SAP es su software ERP (planificación de recursos empresariales), diseñado para optimizar las gestiones de los distintos departamentos de la empresa y aumentar la eficacia general.

SAP, que inicialmente ofrecía SAP CRM como solución local, se ha adaptado a la evolución del sector tecnológico trasladando su oferta de CRM a la nube, con el objetivo de competir con líderes del sector como Salesforce y, al mismo tiempo, satisfacer las necesidades de su base de clientes tradicional manteniendo el soporte para sus productos locales.

El paquete CRM de SAP incluye las nubes de ventas, servicios y marketing, cada una de ellas adaptada para optimizar aspectos específicos de la gestión de las relaciones con los clientes:

- *Sales Cloud* se centra en mejorar las actividades empresariales relacionadas con las ventas y la captación de clientes, proporcionando a los representantes de ventas herramientas para mejorar la productividad e identificar oportunidades de venta.
- *Service Cloud* se dedica a la atención al cliente, ofreciendo soluciones para la resolución de incidencias y la medición de los niveles de satisfacción de los clientes. Proporciona una visión completa de las incidencias y una perspectiva de 360 grados sobre los problemas comunicados por los clientes, agilizando la gestión de incidencias y mejorando la satisfacción del cliente.
- *Marketing Cloud* facilita la comunicación entre empresa y clientes, garantizando una interacción fluida y continua. Permite proporcionar información detallada sobre los intereses de los clientes, lo que favorece la estrategia de ventas y el desarrollo de campañas para mantener un diálogo continuo.

El modelo de licencias de SAP está diseñado específicamente para empresas, ofreciendo acceso exclusivo a su gama de productos. Esto confirma que SAP puede ofrecer una experiencia adaptada a cada cliente, aunque restringe la divulgación de detalles sobre precios y funciones al público en general.

2.1.2.3 Oracle

Fundada en 1977 por Larry Ellison, Oracle se ha consolidado como un referente en el desarrollo de productos que facilitan la gestión empresarial para compañías de todos los tamaños. Oracle se diferencia de Salesforce por ofrecer una gama más amplia de productos, incluidos sistemas de bases de datos, soluciones de software empresarial personalizadas y servicios de desarrollo de software basados en la nube, lo que la convierte en una opción versátil para las distintas necesidades de los clientes.

El CRM de Oracle, al igual que el de Salesforce, funciona en una plataforma basada en la nube diseñada para mejorar la eficacia y eficiencia del negocio. Lo consigue simplificando las tareas de los empleados, automatizando los procesos y permitiendo la integración de datos de sistemas externos, con el objetivo de ofrecer beneficios a sus clientes.

La arquitectura en la nube de Oracle abarca varias áreas clave, en particular las nubes de ventas, servicios y marketing:

- Nube de ventas: se centra en automatizar las actividades diarias de ventas, mejorar el rendimiento de las ventas, mejorar las relaciones con los clientes y proporcionar análisis avanzados. También ayuda a crear previsiones de ventas estandarizadas.
- Nube de servicios: satisface las necesidades de atención al cliente ofreciendo varios canales de comunicación para consultas, quejas o problemas de los clientes, con soporte para integraciones con sistemas CTI (*Computer Telephony Integration*), y correo electrónico. Incluye herramientas para medir la satisfacción del cliente y optimizar las tareas de asistencia.
- Nube de marketing: mejora el compromiso con el cliente a través de campañas publicitarias dirigidas y una interacción continua, lo que permite un conocimiento más profundo de los intereses del cliente y posibilita estrategias de marketing personalizadas. Esta nube ayuda a los equipos de ventas a dirigirse a clientes potenciales con verdadero interés, mejorando las estrategias y el rendimiento de las ventas.

El modelo de licencias de Oracle es similar al de Salesforce al dividir sus servicios por nubes, haciendo que los clientes sólo paguen por las funcionalidades que utilizan. Los usuarios también necesitan una licencia específica para acceder a las funciones de los servicios en la nube de Oracle.

A continuación, se detallan los precios y las funcionalidades de las principales nubes de Oracle. Las licencias de Oracle Marketing Cloud tienen en cuenta factores como el volumen de contactos, la frecuencia de contacto con el cliente y las preferencias de comunicación.

SALES	***Professional Edition***	***Standard Edition***	***Enterprise Edition***	***Premium Edition***
	$65 *user*/mes	$100 *user*/mes	$200 *user*/mes	$300 *user*/ mes
Core de Oracle	✓	✓	✓	✓
Aplicación móvil	✓	✓	✓	✓
Analíticas de venta	✓	✓	✓	✓
Pronóstico de ventas	✓	✓	✓	✓
Campañas	✓	✓	✓	✓
Herramientas de configuración	✓	✓	✓	✓
Manejo de múltiples negocios	✓	✓	✓	✓
Integración con Outlook	✗	✓	✓	✓
Gestión de territorios	✗	✓	✓	✓
Aplicación de iPad para ventas	✗	✗	✓	✓
Compensación de incentivos	✗	✗	✓	✓
Gestión de presupuestos	✗	✗	✓	✓
Integración con Gmail	✗	✗	✓	✓
Predicción de ventas	✗	✗	✓	✓
Oracle Voice	✗	✗	✗	✓
Enterprise Contracts	✗	✗	✗	✓

Tabla 2-5. Licencias de la nube de ventas de Oracle (artículo de Reveal Compliance "*Oracle Sales Cloud Licensing And Pricing Guide*", última visita: marzo 2024)

SERVICE	*Standalone Chat*	*Standard Dynamic*	*Enterprise Dynamic*	*Enterprise Contact Center*
	$90/mes	$110/mes	$140/mes	$250/mes
Funcionalidad de analítica	✓	✓	✓	✓
Portal de clientes	✓	✓	✓	✓
Chat	✓	✗	✗	✓
Gestión de contratos de servicio	✓	✓	✓	✓
Plataforma en la nube	✓	✓	✓	✓
Encuestas por chat	✓	✗	✗	✗
Gestión del conocimiento	✓	✓	✓	✓
Gestión de emails	✗	✓	✓	✓
Consola del usuario dinámica	✗	✓	✓	✓
Multicanal de comunicación de incidencias	✗	✓	✓	✓
Asistente guiado en la plataforma	✗	✗	✓	✓
Gestión del *feedback* del cliente	✗	✗	✓	✓
Personalización de comunicaciones de email	✗	✗	✓	✓
Registro de productos	✗	✗	✗	✓

Tabla 2-6. Licencias de la nube de servicios de Oracle (artículo de Reveal Compliance "*Oracle Service Cloud Licensing And Pricing Guide*", última visita: marzo 2024).

2.2 Ecosistema Salesforce

Para la realización de este proyecto, se ha elegido Salesforce como herramienta con la que desarrollar el CRM en el caso de estudio planteado de una empresa administradora de fincas. Esta decisión se debe a que es uno de los principales sistemas CRM a nivel global, contando con más de 150.000 empresas totalmente gestionadas por su plataforma. Además, su versatilidad y flexibilidad permiten adaptar fácilmente el sistema a las necesidades de la empresa, desde la gestión de las diferentes propiedades hasta la gestión de clientes y contactos.

Como se ha mencionado antes, la plataforma CRM de Salesforce es un sistema informático equipado con las tecnologías más avanzadas para satisfacer todas las necesidades que un CRM pretende abordar: ventas, marketing, servicio al cliente y postventa. Esto lo hace ofreciendo diferentes productos o servicios adaptados a los requisitos de cualquier empresa.

Figura 2-1. Productos de Salesforce.

Salesforce se ha desarrollado con una estructura de capas, una encima de la otra, que permite la utilización de objetos de una capa en otra. Esto se aplica también a las nubes de Salesforce, por ejemplo, si se crea un objeto en la nube de ventas, este mismo objeto puede ser utilizado por cualquier otra nube. Esto significa que los datos se pueden compartir y ser reutilizados por diferentes nubes, facilitando la integración y colaboración entre departamentos.

Esta estructura en capas se complementa con varias características clave de Salesforce:

- *Trusted Multitenant Cloud*: la confianza es la máxima prioridad para Salesforce, donde la nube multiinquilino es la base de todo lo que se crea en Salesforce.
- *Scalable, Metadata Platform*: el secreto de la plataforma reside en su arquitectura orientada a metadatos. Todas las personalizaciones en Salesforce, incluyendo código, configuraciones, aplicaciones y demás, se define mediante metadatos. La capa de metadatos está separada de la capa de servicios, lo que permite actualizaciones sencillas y continuas.
- *Complete CRM*: Las aplicaciones de Salesforce, como *Sales Cloud*, se basan en la plataforma Salesforce. Estas aplicaciones, junto con las aplicaciones personalizadas, ofrecen una funcionalidad coherente y potente. La plataforma garantiza una integración perfecta de aplicaciones prediseñadas y personalizadas para una solución CRM completa.

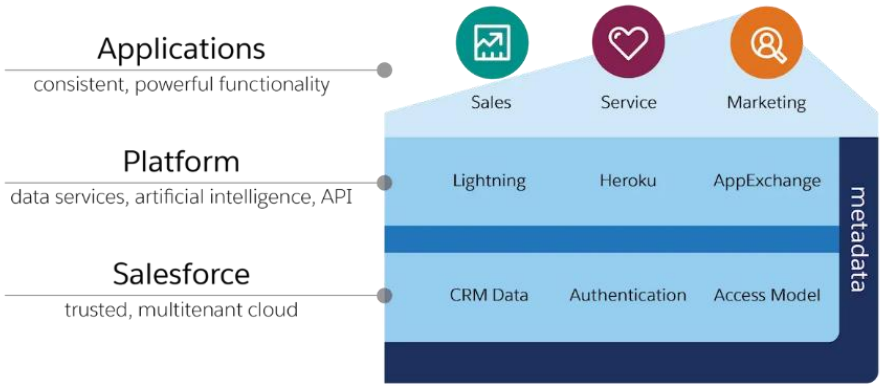


Figura 2-2. Arquitectura de Salesforce.

A continuación, se ofrecerá una visión general del modelo de datos y las herramientas claves del entorno Salesforce, mostrando solo las secciones más relevantes de las funciones principales de la plataforma CRM.

2.2.1 Configuración de datos en Salesforce

Un modelo de datos define la estructura de una base de datos organizando las tablas de forma lógica. En Salesforce, el modelo de datos está basado en objetos, donde los objetos representan tablas en la base de datos. Cada objeto contiene campos (columnas) y registros (filas). Un registro es una instancia de un objeto, lo que significa que un objeto puede tener varios registros, cada uno de los cuales almacena valores en sus campos.

Salesforce incluye objetos estándar como Cuenta, Oportunidad, Caso y Cliente potencial, que son componentes del CRM listos para usar que pueden modificarse dentro de ciertos límites. Cada objeto estándar dispone de campos que permiten al programador definir el objeto según los requisitos del sistema que se está creando. Además, los usuarios pueden crear objetos personalizados para satisfacer sus necesidades específicas, ya que los objetos estándar sólo tienen 5 campos estándar y todos los demás campos deben ser creados por el desarrollador. Estos dos tipos de objetos permiten crear un modelo de datos alineado con el modelo de negocio de la empresa.

Los componentes clave de un objeto son:

- Campos: Almacenan información específica dentro de un registro. Disponibles en varios tipos como texto, numérico, fecha, lista, fórmula y casilla de verificación.
- Relaciones: Campos que vinculan datos entre objetos. Existen varios tipos, como las relaciones de búsqueda (*lookup*), que conectan un objeto con otro de manera opcional, y las relaciones maestro-detalle (*master-detail*), que crean una relación dependiente entre dos objetos, donde el objeto detalle hereda propiedades del objeto maestro.
- Formatos de página: Determinan la disposición visual de campos, botones, enlaces y componentes en las páginas de registro de los objetos, ofreciendo diferentes plantillas de visualización en función del tipo de registro.
- Páginas de registro: Presentan la información de los registros individuales.
- Tipos de registro: Diferencian procesos de negocio, diseños de página y valores en listas de selección entre los usuarios. Permiten personalizar la presentación o comportamiento de los registros según diferentes necesidades.
- Botones: Permiten ejecutar procesos que ayudan a los usuarios en la gestión de los datos.
- Enlaces y acciones: Permiten una navegación fluida entre las páginas de registro, proporcionando un mayor control sobre los procesos.
- Reglas de validación: Garantizan la integridad y calidad de los datos, evaluando condiciones específicas y mostrando mensajes de error cuando los datos no cumplen con criterios establecidos.

2.2.2 Automatización en Salesforce

La automatización en Salesforce se refiere al uso de herramientas y funciones dentro de la plataforma para automatizar tareas repetitivas y procesos complejos, mejorando así la eficacia de las operaciones. Salesforce ofrece varias herramientas "*Point-and-Click*" que permiten a los usuarios configurar estas automatizaciones sin necesidad de amplios conocimientos de programación a través de interfaces intuitivas.

Entre las herramientas más destacadas de Salesforce se incluyen:

- *Flow Builder*: es una herramienta que permite a los usuarios diseñar procesos empresariales mediante una interfaz de arrastrar y soltar. Existen varios tipos de flujos: Flujos de pantalla (*Screen Flows*), se utilizan para guiar a los usuarios a través de pasos o formularios, recopilar datos y mostrar información de forma dinámica. Flujos iniciados automáticamente (*Auto-launched Flows*), se ejecutan automáticamente sin necesitar interacción del usuario y pueden ser lanzados por diferentes eventos (comúnmente utilizados para tareas en segundo plano). Flujos desencadenados por programación (*Scheduled Flows*), se ejecutan a horas específicas, lo que permite a las empresas automatizar tareas recurrentes según un calendario establecido.

- *Process Builder*: es otra herramienta que permite crear procesos automatizados y es útil para automatizaciones más básicas. Sin embargo, es importante tener en cuenta que Salesforce está eliminando *Process Builder* en sustitución de *Flow Builder*, ya que ofrece una funcionalidad más avanzada y una mayor flexibilidad. Es por esto por lo que se recomienda a los usuarios que pasen sus automatizaciones a *Flow Builder* para el futuro.
- *Workflow Rules*: son reglas de automatización simples que permiten definir acciones automatizadas según ciertos criterios. Son fáciles de configurar y utilizar, y pueden activar diversas acciones, como la actualización de campos, la creación de tareas y la actualización de registros relacionados. Sin embargo, a medida que Salesforce continúa evolucionando, se recomienda a los usuarios a utilizar *Flow Builder*.

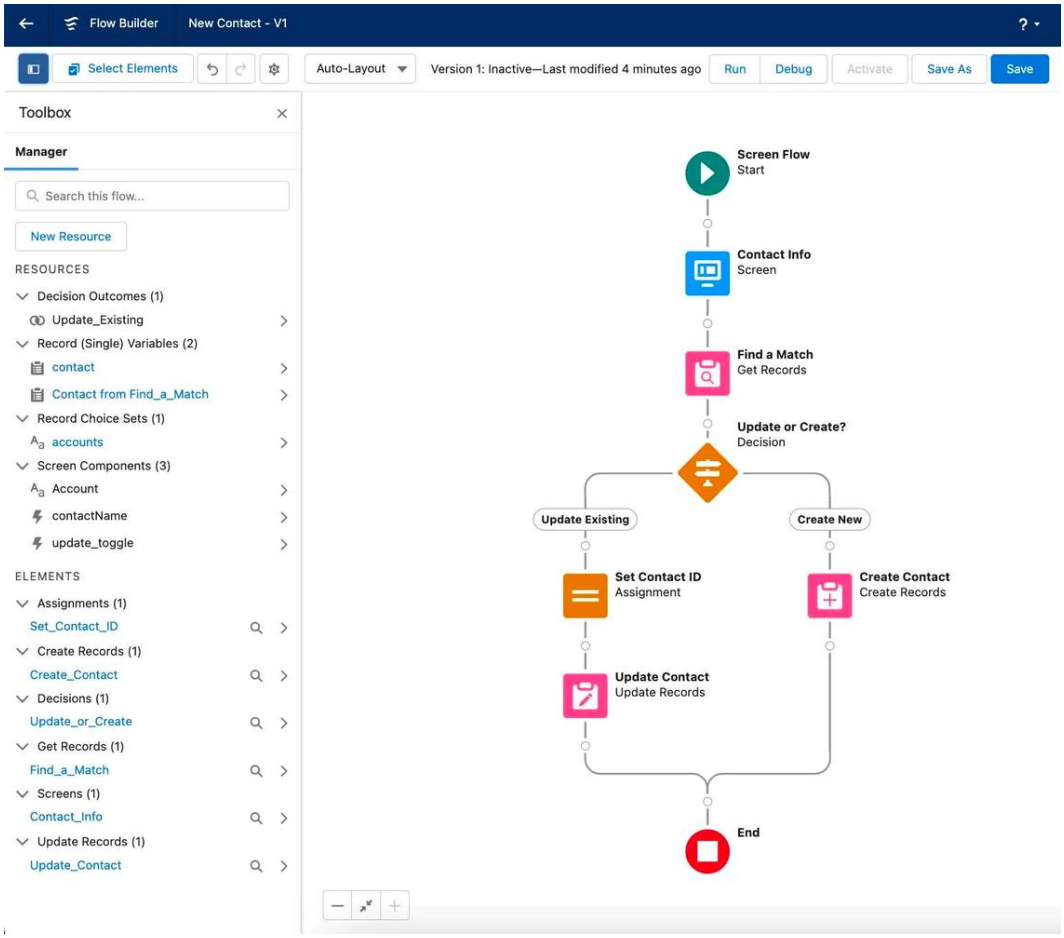


Figura 2-3. Ejemplo de flujo de pantalla en *Flow Builder*.

2.2.3 Desarrollo en Salesforce

La implementación de un CRM en Salesforce no solo requiere de la configuración de datos y la automatización de flujos de trabajo, sino también del desarrollo de funcionalidades e interfaces personalizadas para satisfacer las necesidades de la empresa.

Para ello, los desarrolladores necesitan conocimientos en lenguajes de programación, integración de sistemas y creación de aplicaciones personalizadas. Además, estos pueden aprovechar las herramientas y extensiones que ofrece Salesforce para todas las etapas del desarrollo de aplicaciones, como la consola de desarrollo, un IDE capaz de crear, depurar y probar aplicaciones dentro de un entorno de Salesforce.

2.2.3.1 Apex

Apex es un lenguaje de programación (propiedad de Salesforce) orientado a objetos que permite a los desarrolladores ejecutar instrucciones de control y transacción de flujo. Utiliza una sintaxis similar a la del lenguaje Java y se integra con procedimientos de base de datos. Apex permite a los desarrolladores agregar lógica empresarial para la mayoría de los eventos del sistema.

Criterio/Lenguaje	**Java**	**Apex**
Espacio de trabajo	En el respectivo software	En la nube
Licencia	Necesario	Opcional
Coste	Alto	Medio
Crear informes y *dashboards*	Complejo	Simple
Gestión de datos	Manual	Automatizado

Tabla 2-7. Diferencias entre Java y Apex.

El lenguaje Apex se caracteriza por:

- Facilidad de uso: está basado en Java y C#, lo que incluye variables, expresiones, sintaxis de bucles, objetos y *arrays*. Cuando Apex introduce nuevos elementos, utiliza una sintaxis y semántica fácil de interpretar.
- Centrado en los datos: está diseñado para procesar múltiples consultas y declaraciones DML en una sola unidad de trabajo simple en la plataforma.
- Rigurosidad: lenguaje de programación fuertemente tipado que utiliza referencias directas a objetos y nombres de campos. Los errores en tiempo de compilación ocurren si las referencias son inválidas.
- Alojamiento: el código Apex se guarda, compila y ejecuta en el servidor de Salesforce por lo que no necesita ninguna herramienta adicional para programar.
- Adaptación al entorno *Multitenant*: está diseñado para funcionar eficientemente en el entorno *multitenant* de la plataforma Salesforce, donde múltiples organizaciones comparten los mismos recursos. Cualquier código que viole los límites falla y envía mensajes de error fáciles de entender.
- Actualización automática: el lenguaje Apex no necesita ser reescrito cuando se actualiza, ya que el código se almacena como metadatos en la plataforma. Apex se actualiza automáticamente cuando Salesforce se actualiza.
- Facilidad de pruebas: proporciona soporte incorporado para la creación y ejecución de pruebas unitarias. Incluye resultados de pruebas que indican qué partes del código pueden ser más eficientes.
- Versionado: el código Apex se puede guardar en distintas versiones de la API.

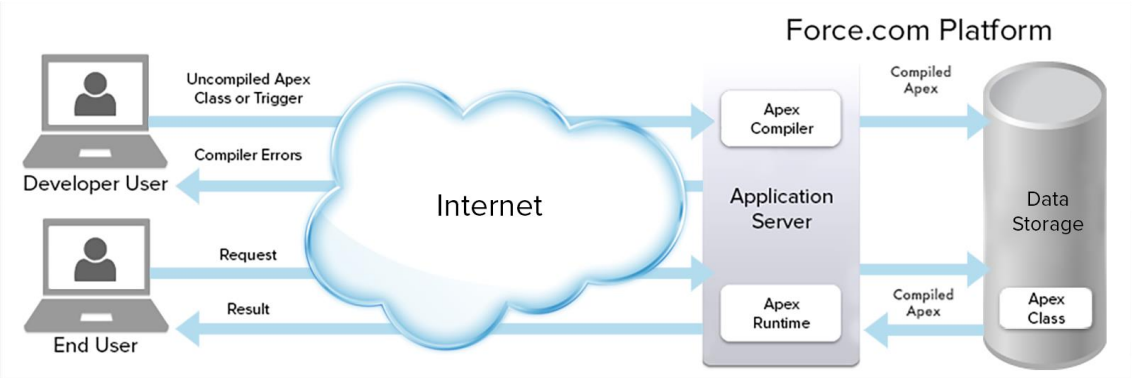


Figura 2-4. Cómo funciona Apex en Salesforce.

Además, el código Apex es particularmente útil para añadir funcionalidad a los disparadores, que se tratarán más adelante.

- Clases Apex

Las clases Apex son un componente fundamental en el lenguaje de programación Apex, utilizado en la plataforma Salesforce. representan modelos o estructuras lógicas que facilitan la creación y manipulación de objetos Apex. Una clase Apex puede contener otras clases, métodos, variables y código estático, proporcionando así un modo sencillo de implementar y ampliar la funcionalidad de los objetos en Salesforce.

Las clases Apex soportan diversos mecanismos de Salesforce, incluyendo automatizaciones, llamadas a la base de datos, componentes Lightning y disparadores (*triggers*). Esto permite a los desarrolladores crear soluciones integrales que interactúan eficazmente con múltiples aspectos de la plataforma, ofreciendo una amplia gama de funcionalidades.

Para asegurar la calidad del código desarrollado, es obligatorio que cada clase Apex esté acompañada de una clase de prueba. Estas clases test tienen como objetivo verificar el correcto funcionamiento del código y deben cubrir al menos el 75% del mismo. Actúan como bancos de prueba que ejecutan el código en situaciones simuladas, activando diferentes partes de la clase Apex para determinar si la prueba es exitosa o no. Esto es crucial para asegurar que el código desplegado en entornos de producción sea de alta calidad y libre de errores.

```
public class EmailManager {
    // Public method
    public void sendMail(String address, String subject, String body) {
        // Create an email message object
        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
        String[] toAddresses = new String[] {address};
        mail.setToAddresses(toAddresses);
        mail.setSubject(subject);
        mail.setPlainTextBody(body);
        // Pass this email message to the built-in sendEmail method
        // of the Messaging class
        Messaging.SendEmailResult[] results = Messaging.sendEmail(
                                new Messaging.SingleEmailMessage[] { mail });
        // Call a helper method to inspect the returned results
        inspectResults(results);
    }
    // Helper method
    private static Boolean inspectResults(Messaging.SendEmailResult[] results) {
        Boolean sendResult = true;
        // sendEmail returns a list of result objects.
        // Iterate through the list to inspect results.
        // In this class, the methods send only one email,
        // so we should have only one result.
        for (Messaging.SendEmailResult res : results) {
            if (res.isSuccess()) {
                System.debug('Email sent successfully');
            }
            else {
                sendResult = false;
                System.debug('The following errors occurred: ' + res.getErrors());
            }
        }
        return sendResult;
    }
}
```

Figura 2-5. Ejemplo de clase Apex.

➢ *Triggers*

En Salesforce, los disparadores son componentes clave que permiten la ejecución de acciones personalizadas antes o después de que ocurran eventos específicos en los registros de la base de datos. Estos eventos pueden incluir la inserción, actualización o eliminación de registros.

Existen dos tipos principales de disparadores:

- *Before Triggers*: Se utilizan para realizar acciones antes de que los registros se guarden en la base de datos. Estas acciones se aplican exclusivamente al propio registro que ha activado el evento. Son útiles para la validación de datos o para modificar valores del registro antes de que se confirme su almacenamiento.
- *After Triggers*: Se ejecutan después de que los registros se han guardado en la base de datos. A diferencia de los *Before Triggers*, las acciones en *After Triggers* pueden afectar tanto al registro que activó el evento como a otros registros independientes. Esto es especialmente útil para la actualización de registros relacionados o para la creación de registros adicionales basados en los cambios realizados.

```
1  trigger ContextExampleTrigger on Account (before insert, after insert, after delete) {
2      if (Trigger.isInsert) {
3          if (Trigger.isBefore) {
4              // Process before insert
5          } else if (Trigger.isAfter) {
6              // Process after insert
7          }
8      }
9      else if (Trigger.isDelete) {
10         // Process after delete
11     }
12 }
```

Figura 2-6. Ejemplo de *trigger*.

Dentro de los disparadores, existen variables de contexto que proporcionan información crucial sobre los valores de un registro en un momento dado. Estas variables permiten acceder a los valores anteriores y nuevos de los registros afectados, facilitando así la implementación de lógica compleja basada en las diferencias de estado. En la Figura 2-6. Ejemplo de *trigger*. se pueden observar las siguientes variables:

Variable	Uso
isInsert	Devuelve *true* si este desencadenador se activó debido a una operación de inserción en la interfaz de usuario de Salesforce, Apex o la API.
isDelete	Devuelve *true* si este desencadenador se activó debido a una operación de eliminación en la interfaz de usuario de Salesforce, Apex o la API.
isBefore	Devuelve *true* si este desencadenador se ha activado antes de que se haya guardado cualquier registro.
isAfter	Devuelve *true* si este desencadenador se ha activado después de que se hayan guardado todos los registros.

Tabla 2-8. Variables de contexto para desencadenadores.

Los *triggers* son esenciales para la implementación de reglas de negocio y seguridad en Salesforce. Se utilizan para asegurar que los registros cumplen con ciertos criterios o reglas antes de que se confirmen los cambios. Por ejemplo, pueden verificar que un registro cumple con políticas de seguridad específicas o que los datos introducidos son válidos.

Además, los *triggers* pueden gestionar la creación y eliminación de jerarquías dentro del sistema. Por ejemplo, al eliminar una cuenta matriz, un disparador puede configurarse para eliminar automáticamente todas las cuentas hijas y otros registros asociados, asegurando así la integridad y coherencia de los datos.

2.2.3.2 Base de datos

Las bases de datos de Salesforce utilizan una variante del estándar SQL denominada SOQL (*Salesforce Object Query Language*), desarrollada por Salesforce. Además, las operaciones DML (*Data Manipulation Language*) permiten la inserción, actualización o eliminación de registros dentro del entorno de trabajo. La combinación de SOQL con estas operaciones da como resultado un modelo de gestión de datos para la manipulación integral de bases de datos. Tanto las operaciones de consulta como las de modificación de registros pueden incorporarse en el código Apex.

- SOQL (lenguaje de consulta de objetos de Salesforce)

Una consulta SOQL tiene la siguiente sintaxis:

- *SELECT*: especifica los campos deseados en la consulta, separados por comas.
- *FROM*: indica el objeto del que obtener los campos.
- *WHERE*: opcional, añade condiciones en los campos del objeto para filtrar los registros en función de criterios específicos.

```
SELECT one or more fields
FROM an object
WHERE filter statements and, optionally, results are ordered
```

```
SELECT Id, Name
FROM Account
WHERE Name = 'Sandy'
```

Figura 2-7. Ejemplo de consulta SOQL.

Las consultas permiten recuperar registros de la base de datos a nivel de tabla, exportar datos en varios formatos (CSV, JSON, XML), modificar campos y eliminar registros.

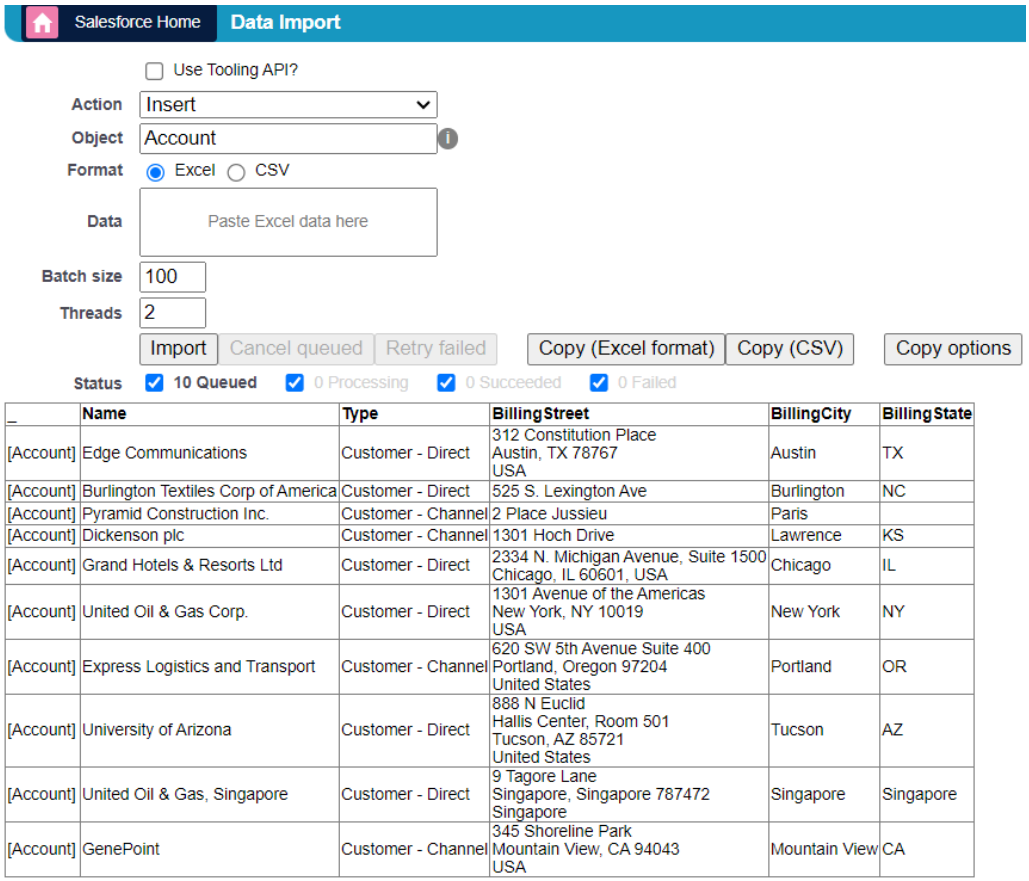
- DML (lenguaje de manipulación de datos)

Salesforce admite seis operaciones DML:

- *INSERT*: añade un registro a la base de datos.
- *UPDATE*: modifica un registro en la base de datos.
- *UPSERT*: si el registro existe, lo actualiza; si no, inserta un nuevo registro.
- *DELETE*: elimina un registro de la base de datos.
- *UNDELETE*: recupera un registro eliminado de la base de datos.
- *MERGE*: fusiona hasta tres registros en uno, eliminando los demás y vuelve a crear los registros relacionados.

Cada registro de Salesforce tiene un ID de entorno único, lo que garantiza la integridad de los datos durante las operaciones DML. Estas operaciones pueden realizarse en un único registro o en bloque en una lista de registros, recomendándose las operaciones en bloque para evitar alcanzar los límites de llamadas de Apex (150 por transacción).

Salesforce Inspector es una extensión de Google Chrome que facilita la interacción con la base de datos de Salesforce mediante SOQL (funcionalidad *Data Export*) y DML (funcionalidad *Data Import*), un ejemplo de uso es la inserción masiva de registros de un objeto, como se observa en la siguiente figura.



_	Name	Type	BillingStreet	BillingCity	BillingState
[Account]	Edge Communications	Customer - Direct	312 Constitution Place Austin, TX 78767 USA	Austin	TX
[Account]	Burlington Textiles Corp of America	Customer - Direct	525 S. Lexington Ave	Burlington	NC
[Account]	Pyramid Construction Inc.	Customer - Channel	2 Place Jussieu	Paris	
[Account]	Dickenson plc	Customer - Channel	1301 Hoch Drive	Lawrence	KS
[Account]	Grand Hotels & Resorts Ltd	Customer - Direct	2334 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA	Chicago	IL
[Account]	United Oil & Gas Corp.	Customer - Direct	1301 Avenue of the Americas New York, NY 10019 USA	New York	NY
[Account]	Express Logistics and Transport	Customer - Channel	620 SW 5th Avenue Suite 400 Portland, Oregon 97204 United States	Portland	OR
[Account]	University of Arizona	Customer - Direct	888 N Euclid Hallis Center, Room 501 Tucson, AZ 85721 United States	Tucson	AZ
[Account]	United Oil & Gas, Singapore	Customer - Direct	9 Tagore Lane Singapore, Singapore 787472 Singapore	Singapore	Singapore
[Account]	GenePoint	Customer - Channel	345 Shoreline Park Mountain View, CA 94043 USA	Mountain View	CA

Figura 2-8. Data Import: inserción masiva de registros.

2.2.3.3 Visualforce

Visualforce es un *framework* de Salesforce diseñado para el desarrollo de interfaces web personalizadas dentro del entorno de Salesforce, reemplazando las interfaces estándar. Las páginas Visualforce se construyen utilizando tres lenguajes básicos del desarrollo web: HTML, JavaScript y CSS. Aunque su sintaxis difiere del HTML estándar, los elementos funcionan de manera similar.

A continuación, se muestra un ejemplo de la parte HTML de una página Visualforce:

```
1  <apex:page standardController="Contact" >
2      <apex:form >
3          <apex:pageBlock title="Edit Contact">
4              <apex:pageBlockSection columns="1">
5                  <apex:inputField value="{!Contact.FirstName}"/>
6                  <apex:inputField value="{!Contact.LastName}"/>
7                  <apex:inputField value="{!Contact.Email}"/>
8                  <apex:inputField value="{!Contact.Birthdate}"/>
9              </apex:pageBlockSection>
10             <apex:pageBlockButtons >
11                 <apex:commandButton action="{!save}" value="Save"/>
12             </apex:pageBlockButtons>
13         </apex:pageBlock>
14     </apex:form>
15 </apex:page>
```

Figura 2-9. Ejemplo de código Visualforce.

Esta página se vería de la siguiente forma:

Figura 2-10. Ejemplo de interfaz Visualforce dentro de Salesforce.

Las páginas Visualforce permiten crear interfaces de usuario personalizadas, desde páginas completas hasta listas relacionadas, ofreciendo funcionalidades adicionales no disponibles en las opciones estándar de Salesforce. Las clases Apex actúan como controladores o constructores de estas páginas, gestionando los datos necesarios y evitando sobrecargar los *triggers*, que se limitan a realizar comprobaciones esenciales y mostrar mensajes de error o éxito.

A pesar de sus ventajas, Visualforce tiene una limitación importante: no está optimizado para dispositivos móviles. Aunque esto no fue un problema en sus inicios, hoy en día es necesario considerar soluciones que ofrezcan mejor compatibilidad y rendimiento en estos dispositivos. Por esta razón, se utilizan otros componentes, los cuales se explicarán en el siguiente apartado.

2.2.3.4 Componentes Lightning

Los componentes Lightning son aplicaciones web dinámicas diseñadas para dispositivos móviles y de escritorio dentro del entorno de Salesforce. Desarrollados para sustituir a las páginas Visualforce, ofrecen mayor eficiencia y funcionalidad.

Al igual que Visualforce, los componentes Lightning se basan en tres lenguajes básicos del desarrollo web (HTML, JS y CSS). Cada componente Lightning comienza y termina con la notación <aura:component>, equivalente a la etiqueta <html> en una página web. Esta notación define la parte visual de la aplicación web. La funcionalidad se implementa con JavaScript, que actúa como el controlador de la aplicación.

Además de HTML, JavaScript y CSS, los componentes Lightning pueden utilizar clases Apex para aumentar sus funcionalidades. Las clases Apex pueden manejar automatismos y acceder a la base de datos de Salesforce, integrando la lógica de negocio.

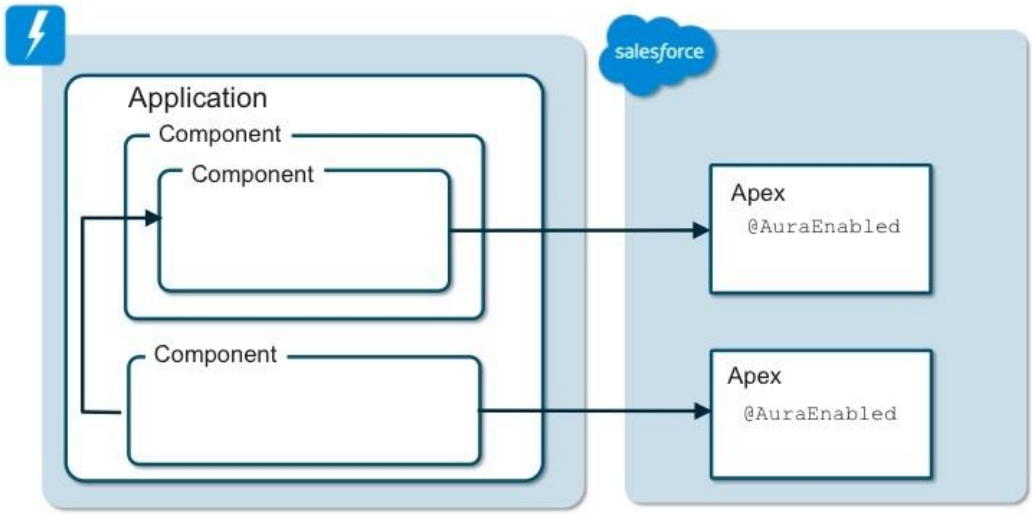


Figura 2-11. Arquitectura de componentes en Salesforce

2.2.3.5 Informes y paneles

Los informes permiten a los usuarios del sistema supervisar el progreso de los objetos deseados con la frecuencia que deseen. Para realizar el seguimiento de dichos objetos, o bien se utilizarán los informes que Salesforce proporciona para objetos estándar o bien se crearán tipos de informes personalizados; ya que es posible crear informes sobre objetos simples o sobre objetos relacionados.

Tras determinar el objeto en cuestión, el usuario tiene la posibilidad de escoger los campos del objeto que más le convengan en caso de que no le interesen todos los campos. Además, Salesforce permite exportar los datos del informe a un fichero Excel, lo cual resulta muy útil a la hora de controlar la evolución con la frecuencia seleccionada. También se podrán diseñar gráficos a partir de los datos recibidos, que a su vez se podrán utilizar como cuadros de mando.

Asimismo, el usuario dispone de diferentes intervalos que ofrece Salesforce para poder programar la ejecución de los informes. De este modo, ya no tiene que actualizar manualmente los informes para comprobar su seguimiento. Una vez ejecutado el informe programado, Salesforce enviará una notificación al usuario vía email para que se pueda realizar la exportación de datos.

3 CASO DE ESTUDIO

Tras haber presentado una visión global tanto de los sistemas CRM, como de las funcionalidades y capacidades de Salesforce, en este capítulo se expondrá el caso de estudio que se va a desarrollar con el fin de alcanzar los objetivos de este trabajo.

3.1 Diseño de la solución

Dado que la administración de fincas implica tanto la gestión de relaciones con propietarios y la captación de nuevas comunidades de vecinos (aspectos cubiertos por *Sales Cloud*) como la atención de incidencias, mantenimiento y servicio al cliente (aspectos cubiertos por *Service Cloud*), se integrarán ambas nubes para crear un sistema personalizado que combine lo mejor de ambas plataformas y se ajuste a los requerimientos de la empresa que se definirán más adelante.

3.1.1 Contexto

El caso de estudio a tratar proviene de una empresa administradora de fincas ficticia cuyos requisitos funcionales se basan en requisitos reales. Partiendo de esta premisa, se plantea el siguiente escenario:

La empresa AdminFincas es una administradora de fincas establecida en Sevilla cuyos clientes actuales son a nivel provincial. Como se acaba de mencionar, esta empresa se especializa en la gestión de fincas, es decir, su actividad principal se centra en la administración y mantenimiento de propiedades y comunidades de propietarios.

Esta empresa comenzó siendo una compañía de servicios variados, ofreciendo servicios integrales que incluían el mantenimiento y reparación de inmuebles, gestión de arrendamientos, asesoramiento en ventas de propiedad, y la administración de comunidades de propietarios. Desde hace un par de años, como parte de su estrategia de crecimiento y mejora de sus servicios, la empresa decidió especializarse exclusivamente en la administración de fincas para convertirse en un referente en este sector.

Actualmente, la empresa depende de herramientas básicas de Microsoft Office, lo que limita la productividad de los empleados y dificulta la capacidad de la empresa para alcanzar sus objetivos. Al implementar un CRM de Salesforce, AdminFincas pretende unificar los datos de la empresa y mejorar el control de los servicios que realizan a sus clientes, lo que implica un mejor rendimiento de las ventas.

A continuación, se van a describir los objetivos que se pretenden alcanzar tras la implementación del proyecto.

3.1.2 Objetivos

AdminFincas busca superar las limitaciones de sus herramientas actuales de Microsoft Office y alcanzar nuevos niveles de eficiencia y productividad, por ello, los objetivos específicos de la empresa para este proyecto se detallarán a continuación:

1. **Centralización de datos y creación de un sistema único de gestión**: unificar toda la información relacionada con propiedades, comunidades de vecinos y proveedores en una única plataforma centralizada. Esto mejorará el funcionamiento de la empresa, eliminando bases de datos múltiples y reduciendo duplicaciones y errores. También se va a abarcar la gestión de clientes, oportunidades, facturas y contratos.
2. **Automatización de tareas y ahorro de recursos**: automatizar procesos administrativos repetitivos como el envío de notificaciones, para que el personal se enfoque en actividades de mayor valor estratégico, optimizando los recursos tecnológicos y reduciendo costes.
3. **Mejora en la comunicación y consolidación de relaciones**: fortalecer las relaciones con los clientes, manteniendo una comunicación eficiente con estos, para poder proporcionar un servicio más personalizado y atento.

4. **Gestión y digitalización de documentación**: digitalizar y organizar documentos actas de juntas de vecinos y registros de mantenimiento. Esto agilizará el acceso a la documentación, garantizando una gestión más segura y ordenada de los archivos y reduciendo el riesgo de pérdida de documentos importantes.
5. **Análisis e informes**: utilizar herramientas para crear cuadros de mando o informes personalizados que analicen la rentabilidad de las propiedades, la eficiencia de los servicios ofrecidos y la satisfacción del cliente.

3.1.3 Metodología de trabajo

La metodología de trabajo empleada para el desarrollo del caso de estudio ha sido la metodología AGILE. Este tipo de metodología se ajusta perfectamente a desarrollos basados en SaaS como es el caso del CRM de Salesforce. La efectividad de esta metodología en proyectos informáticos se basa en los siguientes principios:

- Satisfacción del cliente: Ofrecer un producto final de gran calidad se traduce en una mayor satisfacción por parte del cliente.
- Adaptabilidad a los cambios durante el desarrollo: Permitir la revisión de los requisitos desde el principio hasta el final del proyecto genera un clima de reflexión continua sobre el producto final deseado.
- Comunicación directa con el cliente: Convocar reuniones periódicas con el cliente fomenta una mayor confianza entre las distintas partes del proyecto.
- Entrega de un producto funcional a corto plazo: Proporcionar un prototipo al inicio del desarrollo permite obtener una visión general del proyecto y facilita la detección temprana de posibles mejoras.
- Simplicidad: Identificar los puntos clave y los casos de uso a lograr en el proyecto permite enfocarse en los requerimientos importantes, asegurando que los esfuerzos se centren en las áreas que aportan mayor valor.
- Desarrollo sostenible: Estimar el coste productivo del proyecto proporciona una visión sobre el impacto ambiental del desarrollo, promoviendo prácticas más sostenibles.
- Trabajo conjunto entre todos los participantes del proyecto: Realizar revisiones periódicas del progreso y establecer fechas para las diferentes etapas del desarrollo incrementa la implicación y colaboración de los equipos dentro del proyecto.

Figura 3-1. Metodología AGILE.

3.1.4 Análisis

En este apartado se van a abordar los diferentes requerimientos de la empresa desde distintos niveles. Primero, se realizará un análisis a nivel funcional, donde se detallarán los procesos de negocio que se deben lograr con la implementación del proyecto, en base a los objetivos de la empresa. Luego, a partir de este análisis, se explicará a nivel técnico el desarrollo realizado para conseguir estos procesos de negocio, profundizando en los automatismos implementados. Para facilitar el análisis técnico, previamente se describirá el modelo de datos del entorno de trabajo, lo que permitirá entender mejor cómo operan los diferentes procesos del sistema.

3.1.4.1 Análisis funcional

AdminFincas propone implantar un proyecto de digitalización del modelo de negocio de la compañía con una solución CRM, más concretamente CRM basado en Salesforce, para alcanzar los objetivos de negocio mencionados en el apartado anterior.

La finalidad de este proyecto es garantizar que los empleados de la administradora de fincas gestionen el proceso de venta y de postventa o atención a los clientes de la empresa, con la intención de alcanzar los objetivos definidos.

Proceso	Caso de Uso
Ventas	Gestión de candidatos
	Gestión de clientes
	Gestión de oportunidades
Postventa	Gestión de activos
	Gestión de casos

Tabla 3-1. Casos de uso.

Para abordar la visibilidad de la información almacenada en Salesforce y la gestión de casos de uso dentro del proceso de ventas en el sistema, se implementará una jerarquía de funciones que permitirá organizar los diferentes a los trabajadores de la compañía en función de sus procesos empresariales. AdminFincas se estructura de tal forma que garantiza el máximo rendimiento de su plantilla para llevar a cabo una gestión comercial eficiente.

Tal y como se muestra en la siguiente figura, los roles pueden contemplarse jerárquicamente ordenados en forma de árbol desde un nivel superior a un nivel inferior en cada rama. Esta estructura permite una clara cadena de mando y facilita la toma de decisiones estratégicas por parte del CEO (*Chief Executive Officer*, que se traduce como director ejecutivo), que desde la cima se encarga de supervisar todas las operaciones puesto que cuenta con total acceso a los registros del sistema. A un nivel inferior, se encuentran los dos departamentos principales de la empresa: el Departamento de Operaciones y el Departamento de Activos, coordinados por el Gerente de Operaciones y el Responsable de Activos respectivamente.

El Departamento de Operaciones se encarga del análisis de datos y las relaciones comerciales. Aquí, el Consultor de Ventas está a cargo de los Agentes de Candidato y de Cuenta, que se encargan de la captación y gestión de los clientes. A su vez, el Departamento de Activos gestiona tanto la atención al cliente como la administración de los activos de la mano del Agente de Atención al Cliente y el Agente de Gestor de Activos.

Sin embargo, el Administrador del sistema CRM es el rol protagonista de nuestro proyecto ya que comprende el desarrollo e implementación de Salesforce como CRM. Pese a no tener a un equipo directamente a su cargo, su función juega un papel elemental dado que es su responsabilidad asegurar el correcto funcionamiento del sistema que soporta todas las operaciones de la empresa.

Figura 3-2. Jerarquía de roles dentro del sistema.

3.1.4.1.1 Proceso de ventas

El modelo de proceso de ventas establecido constará de diferentes casos de uso, que se describirán a continuación. El flujo estándar del proceso sería el siguiente:

En primer lugar, el proceso comienza con la gestión de un candidato. El gestor recopilará la información del interesado a través de distintas vías y la almacenará en la base de datos. A continuación, el cliente potencial se convertirá en cliente tras cumplir una serie de requisitos que confirmen su interés. Una vez realizada la conversión, este cliente será vinculado a una oportunidad comercial con la empresa. Estas oportunidades pasarán por varias etapas, concluyendo en un estado de ganada o perdida. En caso de que la oportunidad sea ganada, el gestor comercial generará un contrato utilizando la información del cliente, completando de este modo el proceso de venta.

- Caso de uso 1: Gestión de candidatos

El objetivo de este caso de uso es gestionar clientes potenciales para los servicios ofrecidos por la administradora de fincas. Los datos de los candidatos pueden ser obtenidos a través de diversas vías, como el sitio web, teléfono, visitas comerciales, recomendaciones, entre otros. En esta etapa, es crucial identificar el tipo de cliente y la propiedad para asegurar una gestión eficiente y personalizada en fases posteriores.

Una vez captados los candidatos y si muestran interés en adquirir los servicios de la empresa, pasarán por un proceso de cualificación. Durante este proceso, el gestor evaluará sus necesidades y determinará los beneficios que la empresa puede ofrecerles.

Si el candidato es apto para los intereses de la empresa, se procederá a la conversión del candidato a cliente, este estará conformado por cuenta y contacto/s. Se registrarán todos los datos obtenidos durante la fase de captación en una ficha de cliente para formalizar la relación comercial y mantener un seguimiento adecuado y personalizado a lo largo del tiempo.

- Caso de uso 2: Gestión de clientes

La finalidad de este caso de uso es obtener un registro detallado que recoja toda la información de los clientes (particulares, empresas y comunidades de propietarios). Los responsables comerciales se encargarán de actualizar y complementar la información de cada uno de ellos. Esto es necesario para poder identificar nuevas oportunidades de negocio, siendo el cliente el punto de partida para generar dichas oportunidades. Todos estos datos deben tratarse conforme al actual Reglamento General de Protección de Datos (RGPD).

- Caso de uso 3: Gestión de oportunidades

En este caso de uso se tratan las oportunidades comerciales de AdminFincas con sus clientes, reflejando los progresos que se vayan realizando con el objetivo del cierre exitoso de futuros contratos. Este flujo conlleva diferentes etapas que se mencionan a continuación:

1. Pendiente de Gestión
2. En Negociación
3. Pendiente Aprobación
4. Cerrada (Ganada/Perdida)

Si se cierra la oportunidad como "Ganada", se procederá a la creación de un contrato con los datos del cliente y servicios contratados. Si por el contrario el estado final es "Perdida", se cerrará la oportunidad comercial sin generarse el contrato.

3.1.4.1.2 Proceso de postventa

Una vez generado el contrato, da comienzo el proceso de postventa, donde los clientes pueden ponerse en contacto con el servicio de atención al cliente mediante casos para tratar consultas, quejas y reclamaciones con relación a las oportunidades comerciales con la empresa. Además, en este proceso se incluye también la gestión de los contratos. A continuación, se detallarán los casos de uso en orden de ejecución dentro de este proceso:

- Caso de uso 4: Gestión de activos

La finalidad de este caso de uso es la gestión y almacenamiento de los datos de los activos existentes en la empresa. Los activos representan los productos o servicios contratados a raíz de las oportunidades que se fueron cerradas con éxito. Dentro de una empresa dedicada a la administración de fincas, como es nuestro caso, estos activos corresponden a los contratos generados. Cada contrato estará vinculado a una propiedad y a su vez a una persona de contacto.

- Caso de uso 5: Gestión de casos

El objetivo de este caso de uso es dar soporte al cliente sobre los activos contratados con la empresa. Los casos representan consultas, quejas, solicitudes, etc., que pueden realizan los clientes, estos posibilitan la gestión del servicio de atención al cliente y permiten realizar un seguimiento de los trámites o tareas a realizar por parte de la empresa.

3.1.4.2 Modelo de datos

En esta sección, se define el modelo de datos del sistema para aclarar las relaciones entre los datos y las funciones implementadas en el sistema. Esto permitirá comprender mejor el análisis técnico posterior, ya que los requisitos funcionales del proyecto se basarán en el modelo de datos de Salesforce adaptado para una administradora de fincas.

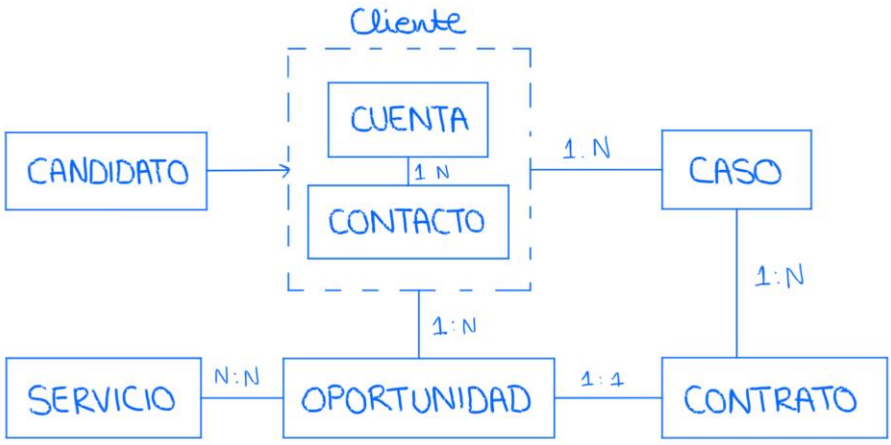


Figura 3-3. Modelo de datos.

A continuación, se van a describir los objetos más importantes necesarios para la implementación del proyecto, explicando su finalidad y detallando sus campos y relaciones más importantes mediante una tabla.

3.1.4.2.1 Candidato

El objeto estándar de Salesforce "Candidato" o "Lead" permite gestionar la información de los posibles clientes de la empresa. En este proyecto, se utiliza para recopilar los datos de estos potenciales clientes de tal forma que, si son considerados como Aptos, se consolidarán como clientes y su información se traspasará a los registros de los objetos "Cuenta" y "Contacto", donde se formalizará el expediente del cliente.

Nombre del campo	Tipo de dato	Descripción
Nombre Completo	*Texto (121)*	Se indican el nombre y los apellidos de la persona de contacto.
Teléfono	*Phone*	Se muestra el número de teléfono de la persona de contacto.
Móvil	*Phone*	Se indica el número de móvil de la persona de contacto.
Correo electrónico	*Email*	Se muestra la dirección de e-mail de la persona de contacto.
Tipo de documento	*Picklist*	Se indica el tipo de documento de identificación de la persona de contacto: DNI o NIF.
Número de documento	*Texto*	Se muestra el número del documento de identidad de la persona de contacto.
Estado de candidato	*Picklist*	Mediante una lista, se indica el estado del candidato: pendiente de cualificación, apto o no apto.
Origen del candidato	*Picklist*	A través de una lista, se podrá seleccionar el origen del candidato: web, consulta telefónica, publicidad, recomendación, lista de contactos u otro.
Propietario del candidato	*Lookup (User)*	Se muestra el nombre del gestor de "AdminFincas" que sea propietario del candidato, será Sara Agente Candidato.
Nombre de la propiedad	*Texto (1300)**	Se combina la información sobre el tipo de cliente, la dirección y código postal para identificar la propiedad.
Dirección	*Address*	Se muestra la dirección en la que se encuentra la propiedad (calle, nº, código postal, ciudad, provincia y país).
Ubicación	*Picklist*	A través de una lista se indica la zona donde se ubica la vivienda: urbana, suburbana o rural.
Solvencia	*Currency (17, 2)*	Solvencia financiera del cliente potencial.
Tipo de cliente	*Picklist*	Mediante una lista se especifica si se trata de un cliente individual, una comunidad o una empresa.
Tipo de propiedad	*Picklist*	A través de una lista, se muestra el tipo de propiedad: residencial, comercial o mixta.
Número de propiedades	*Número*	Se indica el número de propiedades del cliente.
Referencia catastral	*Texto(20)*	Muestra la referencia catastral de la propiedad.
Descripción	*Área de texto*	Espacio para más detalles o anotaciones si fuera necesario.

Tabla 3-2. Objeto Candidato (campos).

3.1.4.2.2 Contacto

El objeto de Contacto en Salesforce se encarga de almacenar información detallada sobre las personas asociadas a cuentas o propiedades.

Nombre del campo	Tipo de dato	Descripción
Propietario del contacto	*Lookup (User)*	Se muestra el nombre del gestor de "AdminFincas" que sea propietario del contrato, será Sara Agente Candidato
Nombre Completo	*Texto (121)*	Se indican el nombre y los apellidos de la persona que representa.
Nombre de la cuenta	*Texto (255)*	Se combina la información sobre el tipo de cliente, la dirección y código postal para identificar la propiedad.
Origen del candidato	*Texto (121)*	Se indica la procedencia del candidato: web, consulta telefónica, publicidad, recomendación, lista de contactos u otro.
Número de documento	*Texto*	Muestra el número del documento de identidad de la persona de contacto.
Tipo de documento	*Texto*	Se indica si documento de identidad es un DNI o un NIF.
Dirección de correo	*Address*	Se muestra la dirección de la propiedad (calle, nº, código postal, ciudad, provincia y país).
Teléfono	*Phone*	Se indica el número de teléfono principal de la persona de contacto.
Teléfono particular	*Phone*	Muestra el número de teléfono fijo residencial de la persona de contacto.
Móvil	*Phone*	Se indica el número de móvil de la persona de contacto.
Otro teléfono	*Phone*	Proporciona un número de teléfono adicional de la persona de contacto.
Correo electrónico	*Email*	Se muestra la dirección de e-mail de la persona de contacto.
Descripción	*Área de Texto*	Espacio para más detalles o anotaciones si fuera necesario.

Tabla 3-3. Objeto Contacto (campos).

3.1.4.2.3 Cuenta

Para gestionar la información de los clientes de la empresa Salesforce ofrece el objeto estándar "Cuenta" o "Account", que en nuestro proyecto será utilizado para reflejar no solo la información del cliente sino también todas operaciones comerciales que lo vinculen con la empresa. Por defecto, el objeto "Cuenta" se relaciona directamente con el objeto "Contacto" para que se puedan añadir varios contactos a una misma cuenta.

Nombre del campo	Tipo de dato	Descripción
Código de la cuenta	*Texto (30)*	Muestra el código personalizado asignado a cada cuenta a modo de identificador.
Nombre de la cuenta	*Texto (255)*	Se combina la información sobre el tipo de cliente, la dirección y código postal para identificar la propiedad.
Tipo de cliente	*Picklist*	Mediante una lista se especifica si se trata de un cliente individual, una comunidad o una empresa.
Zona	*Texto (20)*	Indica el área donde se ubica la propiedad: urbana, suburbana o rural.
Propietario de la cuenta	*Lookup (User)*	Se muestra el nombre del gestor de "AdminFincas" que sea propietario de la cuenta, será Sara Agente Cuenta
Contacto Principal	*Lookup (Contact)*	Se indica quién es la persona de contacto.
Dirección de facturación	*Address*	Se muestra la dirección de la propiedad (calle, nº, código postal, ciudad, provincia y país).
Tipo de propiedad	*Texto (20)*	Se especifica el tipo de propiedad: residencial, comercial o mixta.

Nombre del campo	Tipo de dato	Descripción
Número de propiedades	*Número*	Indica el número de propiedades relacionadas con la cuenta.
Número de propietarios	*Número*	Muestra el número de residentes de la propiedad.
Referencia Catastral	*Texto (20)*	Se indica la referencia catastral de la propiedad.
Detalles de la propiedad	*Picklist*	Se muestra una lista con características de la propiedad (Área de Juegos Infantiles, Áreas Verdes, Ascensor, Conserjería, Garaje, Gimnasio, Piscina, etc.)
Descripción	*Texto (121)*	Espacio para más detalles o anotaciones si fuera necesario.
Solvencia	*Currency (16,2)*	Solvencia financiera del cliente.
¿Morosidad?	*Picklist*	Mediante una lista se seleccionará si hay morosidad o no.
Riesgo de abandono	*Picklist*	A través de una lista se determinará si el grado de riesgo de abandono: alto, medio o bajo
Valor de la propiedad	*Currency (16,2)*	Valor económico de la propiedad
Historial de inversiones	*Área de texto*	Espacio en el que se aportan datos relevantes sobre anteriores inversiones de la propiedad
Estado de infraestructura	*Picklist*	Lista que muestra los diferentes estados de la infraestructura: excelente, buena, regular, pobre y crítica.
Eficiencia Energética	*Picklist*	Mediante una lista se podrá seleccionar la letra correspondiente a la eficiencia energética de la propiedad (A-G)
¿Datos validados?	*Casilla*	Si los datos se dan por validados se marcará la casilla.

Tabla 3-4. Objeto Cuenta (campos).

3.1.4.2.4 Servicio

Se ha creado el objeto personalizado “Servicio” para gestionar los servicios ofrecidos por la empresa, que se negociarán en la oportunidad comercial con el cliente. Se van a diferenciar según el tipo de cliente.

Nombre del campo	Tipo de dato	Descripción
Código de servicio	*Numeración automática*	Número asignado automáticamente como identificador del servicio.
Nombre del servicio	*Texto (80)*	Breve descripción del servicio prestado a modo de identificación.
Tipo de servicio	*Picklist*	Mediante una lista se escoge el tipo de servicio: Administración, Consultoría, Gestión, Mantenimiento o Seguridad.
Descripción del Servicio	*Texto (20)*	Muestra brevemente las características del servicio
Precio del Servicio	*Currency (16,2)*	Se indica el coste anual del servicio.
Propietario de la cuenta	*Lookup (User)*	Se muestra el nombre del Administrador del sistema CRM (Sara Cabeza Muñoz).
Fecha Inicio	*Fecha*	Indica la fecha del comienzo del servicio.
Fecha Fin	*Fecha*	Muestra la fecha en la que finaliza el servicio.
Servicio Vigente	*Casilla*	Si el servicio está vigente se marca la casilla.

Tabla 3-5. Objeto Servicio (campos).

3.1.4.2.5 Oportunidad

El objeto estándar de Salesforce "Oportunidad" u "Opportunity" se utiliza para relacionar el objeto "Cuenta" con otros objetos en base a unos estados. Dentro de nuestro proyecto se implementará para gestionar las oportunidades comerciales de los clientes.

Nombre del campo	Tipo de dato	Descripción
Nombre de la oportunidad	*Texto (120)*	Identificación de la oportunidad
Código Oportunidad Comercial	*Numeración automática*	Número asignado automáticamente como identificador del servicio.
Tipo de Oportunidad	*Picklist*	Cliente existente - Actualización ; Cliente existente - Sustitución; Nuevo cliente.
Nombre de la cuenta	*Lookup (Account)*	Se combina la información sobre el tipo de cliente, la dirección y código postal para identificar la propiedad.
Código de la cuenta	*Texto (30)*	Muestra el código personalizado asignado a cada cuenta a modo de identificador.
Etapa	*Picklist*	Listado de los estados de la oportunidad: Pendiente Gestión, En Negociación, Pendiente Aprobación; Ganada ; Perdida
Probabilidad	*Percent (3,0)*	Porcentaje de probabilidad de cerrar con éxito la oportunidad.
Presidente de la comunidad	*Lookup (Contact)*	Muestra a la persona de contacto.
Detalles de la propiedad	*Picklist*	En una lista aparecen las características de la propiedad (Área de Juegos Infantiles, Áreas Verdes, Ascensor, Conserjería, Garaje, Gimnasio, Piscina, etc.)
Propietario de la oportunidad	*Lookup (User)*	Se muestra el nombre del gestor de "AdminFincas", será Sara Consultor Ventas.
Fecha Cierre	*Fecha*	Indica la fecha en la que se cierra la oportunidad.
Descripción	*Área de texto*	Espacio para más detalles o anotaciones si fuera necesario.

Tabla 3-6. Objeto Oportunidad (campos).

3.1.4.2.6 Contrato

El objeto estándar "Contrato" o "Contract" se utiliza para lidiar con los activos de la empresa dentro del sistema. En este proyecto se usará para gestionar la información de los contratos en vigor o que hayan sido anulados por los clientes.

Nombre del campo	Tipo de dato	Descripción
Número del contrato	*Número*	Número identificador del contrato.
Estado	*Picklist*	A través de una lista se determina el estado del contrato: creado, en vigor o anulado.
Oportunidad origen	*Lookup (Opportunity)*	Oportunidad que ha originado el contrato.
Nombre de la cuenta	*Lookup (Account)*	Cuenta asociada al contrato.
Fecha Inicio	*Fecha*	Indica la fecha del comienzo del contrato.
Duración del contrato	*Número(4, 0)*	Duración del contrato en meses.

Nombre del campo	Tipo de dato	Descripción
Fecha Fin	*Fecha*	Muestra la fecha en la que vence el contrato.
Firmado por el cliente	*Lookup(Contact)*	Identifica el cliente que firma el contrato.
Fecha de firma del cliente	*Fecha*	Fecha en la que el cliente firma el contrato.
Firmado por la compañía	*Lookup(User)*	Identifica quién de la empresa firma el contrato.
Fecha de firma compañía	*Fecha*	Fecha en la que el representante de la empresa firma el contrato.
Propietario del contrato	*Lookup (User)*	Se muestra el nombre del gestor de "AdminFincas" que sea propietario del contrato, será Sara Gestor Activos.
Aviso de vencimiento del propietario	*Lista de selección*	Indica un aviso de vencimiento a 15, 30, 45, 60, 90 o 120 días del fin del contrato.
Lista de servicios	*Texto (40)*	Lista de los servicios contratados.
Importe total servicios	*Currency (16,2)*	Se indica el coste anual de los servicios contratados.
Medio de pago	*Picklist*	Mediante una lista se seleccionará el medio de pago: Domiciliación bancaria, Efectivo, Transferencia, Bizum o Impagado
Periodicidad	*Picklist*	A través de una lista se muestran los periodos de pago: Mensual, Trimestral, Semestral o Anual.
Condiciones especiales	*Texto (40)*	Campo de texto para detallar condiciones particulares o excepciones del contrato.

Tabla 3-7. Objeto Oportunidad (campos).

3.1.4.2.7 Caso

El objeto "Caso" o "Case" es un objeto estándar de Salesforce utilizado para gestionar la atención al cliente de la empresa. Dentro del proyecto se usará para gestionar las consultas, quejas y reclamaciones de los clientes de la empresa.

Nombre del campo	Tipo de dato	Descripción
Número del caso	*Numeración automática*	Número asignado automáticamente como identificador del caso.
Asunto	*Texto (255)*	Breve descripción de la situación.
Nombre del contacto	*Lookup (Contact)*	Muestra a la persona de contacto.
Nombre de la cuenta	*Lookup (Account)*	Se indica la cuenta vinculada al caso.
Tipo	*Picklist*	Mediante una lista se determina el tipo de caso: Reclamación, Petición u Otro.
Teléfono del contacto	*Phone*	Muestra el número de teléfono de la persona de contacto.
Correo electrónico del contacto	*Email*	Indica la dirección de correo electrónico de la persona de contacto.
Origen del caso	*Picklist*	Una lista muestra el medio de donde proviene el caso: Teléfono, Email o WhatsApp.
Contrato	*Lookup (Contract)*	Muestra el contrato vinculado al caso.
Motivo del caso	*Picklist*	Se indica a través de una lista Administración, Consultoría, Gestión, Mantenimiento, Seguridad u Otro.
Descripción	*Área de texto*	Espacio para más detalles o anotaciones si fuera necesario.
Comentarios internos	*Área de texto*	Espacio para comentarios internos de los usuarios.

Nombre del campo	Tipo de dato	Descripción
Estado	*Picklist*	A través de una lista se seleccionará el estado del caso (Pendiente/Cerrado).
Prioridad	*Picklist*	Determinará la importancia del caso: Alta, Media o Baja.
Propietario del caso	*Lookup (User)*	Se muestra el nombre del gestor de "AdminFincas", será Sara Atención Cliente.
Fecha apertura	*Fecha*	Indica la fecha del comienzo del servicio.
Fecha cierre	*Fecha*	Muestra la fecha en la que finaliza el servicio.
Servicio relacionado	*Lookup (Servicio)*	Se indica el servicio al que se refiere el caso.

Tabla 3-8. Objeto Caso (campos).

3.1.4.3 Análisis técnico

En este apartado se describen las funcionalidades más destacables de los procesos de ventas y postventa desde una perspectiva técnica basada en Salesforce, explicando qué automatismos y herramientas deben implementarse para cubrir los requisitos necesarios en cada caso de uso.

Se ha optado por crear una nueva aplicación desde cero, basada en la nube de Servicios de Salesforce:

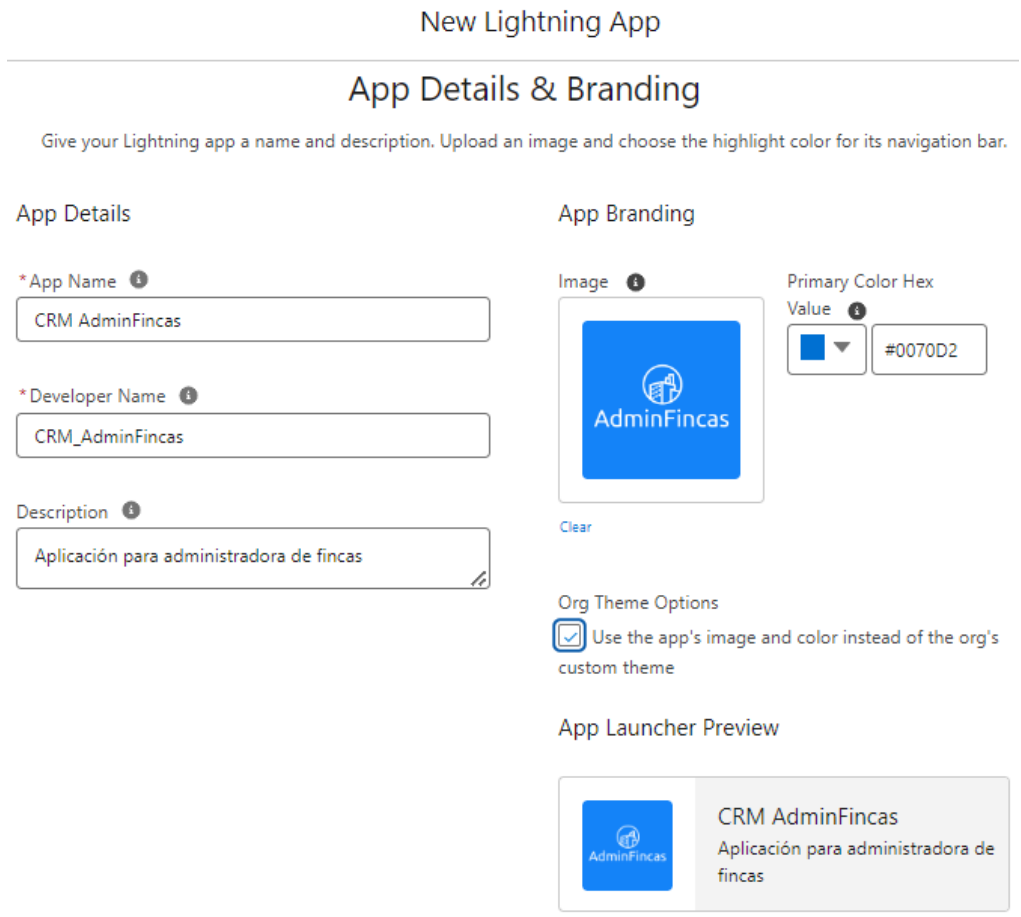


Figura 3-4. Configuración de la aplicación CRM.

A continuación, se van a exponer los casos de uso a nivel técnico:

- Caso de uso 1: Gestión de candidatos

Tal y como se define en el análisis funcional, este caso de uso es el primer paso en el proceso de ventas y consiste en la gestión de los datos de los clientes potenciales y la posibilidad de convertirlos en clientes una vez que hayan superado ciertos procesos internos de cualificación.

El registro del objeto estándar "Lead" debe crearse con estado "Pendiente Cualificación" y tras su creación, debe generarse una tarea asociada al candidato con asunto "Candidato pendiente de cualificación". Esta tarea irá destinada a un analista de la empresa para que determine la aptitud del candidato, vencerá a los 7 días y se enviará una notificación por correo como aparece en la siguiente imagen:

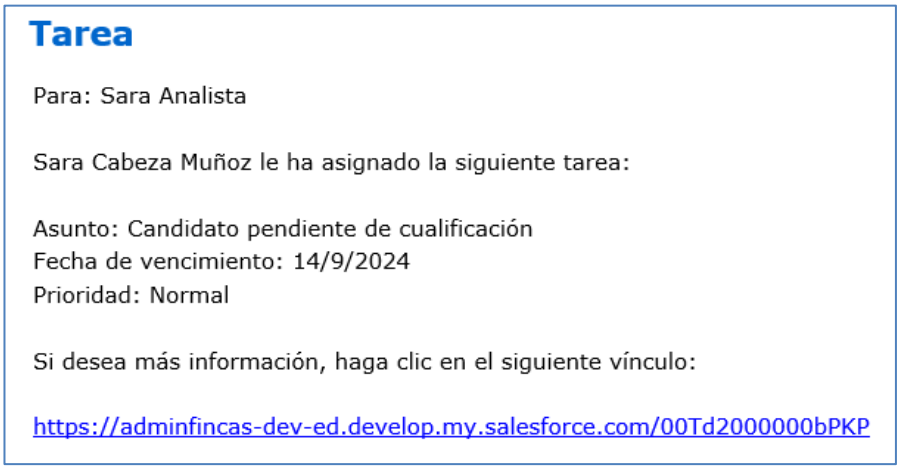


Figura 3-5. Notificación tarea "Candidato pendiente de cualificación"

Cuando se decida el futuro del candidato, el analista debe poder actualizar su estado a "Apto" o "No Apto". En caso de no ser apto, el candidato se queda en la base de datos del sistema, y en caso de ser apto, se convierte automáticamente a cliente, generándose un contacto y una cuenta relacionados entre sí con la información insertada en el candidato.

Todo este caso de uso se va a manejar con el flujo "Lead: Gestión candidatos", creado con la herramienta Flow Builder, que invoca a la clase Apex "ConvertirCandidatoApex":

```
public with sharing class ConvertirCandidatoApex {

    // método invocable desde el flujo
    @InvocableMethod(label='ConvertirCandidatoApex' description='Convierte un Candidato en Cuenta y Contacto')
    public static List<ResultadoConversionCandidato> ConvertirCandidato(List<CandidatoCuentaContacto> listaCandidatoCuenta) {
        List<ResultadoConversionCandidato> resultado = new List<ResultadoConversionCandidato>();

        // obtenemos el estado de conversión del candidato
        LeadStatus estadoConversion = [SELECT Id, MasterLabel, ApiName FROM LeadStatus WHERE IsConverted = true LIMIT 1];

        // almacenamos las conversiones de candidatos en una lsita
        List<Database.LeadConvert> listaConversiones = new List<Database.LeadConvert>();

        // se itera sobre cada candidato a convertir
        for (CandidatoCuentaContacto cc : listaCandidatoCuenta) {
            System.debug('Procesando Candidato: ' + cc);

            // configuración de la conversión de candidato
            Database.LeadConvert conversionCandidato = new Database.LeadConvert();
            conversionCandidato.setLeadId(cc.idCandidato);
            conversionCandidato.setAccountId(cc.idCuenta);      // cuenta ya se ha creado en el flujo
            conversionCandidato.setContactId(cc.idContacto);    // contacto ya se ha creado en el flujo
            conversionCandidato.setConvertedStatus(estadoConversion.MasterLabel);
            conversionCandidato.setDoNotCreateOpportunity(true); // no queremos que se creen oportunidades en la conversión

            listaConversiones.add(conversionCandidato);
        }

        // si hay candidatos para convertir se realiza la conversión
        if (!listaConversiones.isEmpty()) {
            List<Database.LeadConvertResult> resultadosConversion = Database.convertLead(listaConversiones);

            // procesamos los resultados de la conversión
            for (Database.LeadConvertResult resultConversion : resultadosConversion) {
                ResultadoConversionCandidato resultConvCandidato = new ResultadoConversionCandidato();
                if (resultConversion.isSuccess()) {
                    resultConvCandidato.ResultadoConversion = 'Éxito: Candidato convertido correctamente';
                } else {
                    resultConvCandidato.ResultadoConversion = 'Error: ' + resultConversion.getErrors();
                }
                resultado.add(resultConvCandidato);
            }
        }
        return resultado;

    // clase interna para recibir los datos del candidato, cuenta y contacto desde el flujo
    public class CandidatoCuentaContacto {
        @InvocableVariable(label='ID del Candidato' description='ID del Candidato a convertir')
        public Id idCandidato;

        @InvocableVariable(label='ID de la Cuenta' description='ID de la Cuenta asociada')
        public Id idCuenta;

        @InvocableVariable(label='ID del Contacto' description='ID del Contacto asociado')
        public Id idContacto;
    }

    // clase interna para devolver el resultado de la conversión al flujo
    public class ResultadoConversionCandidato {
        @InvocableVariable(label='Resultado de la Conversión' description='Resultado del proceso de conversión')
        public String ResultadoConversion;
    }

}
```

Figura 3-6. Clase Apex “ConvertirCandidatoApex”.

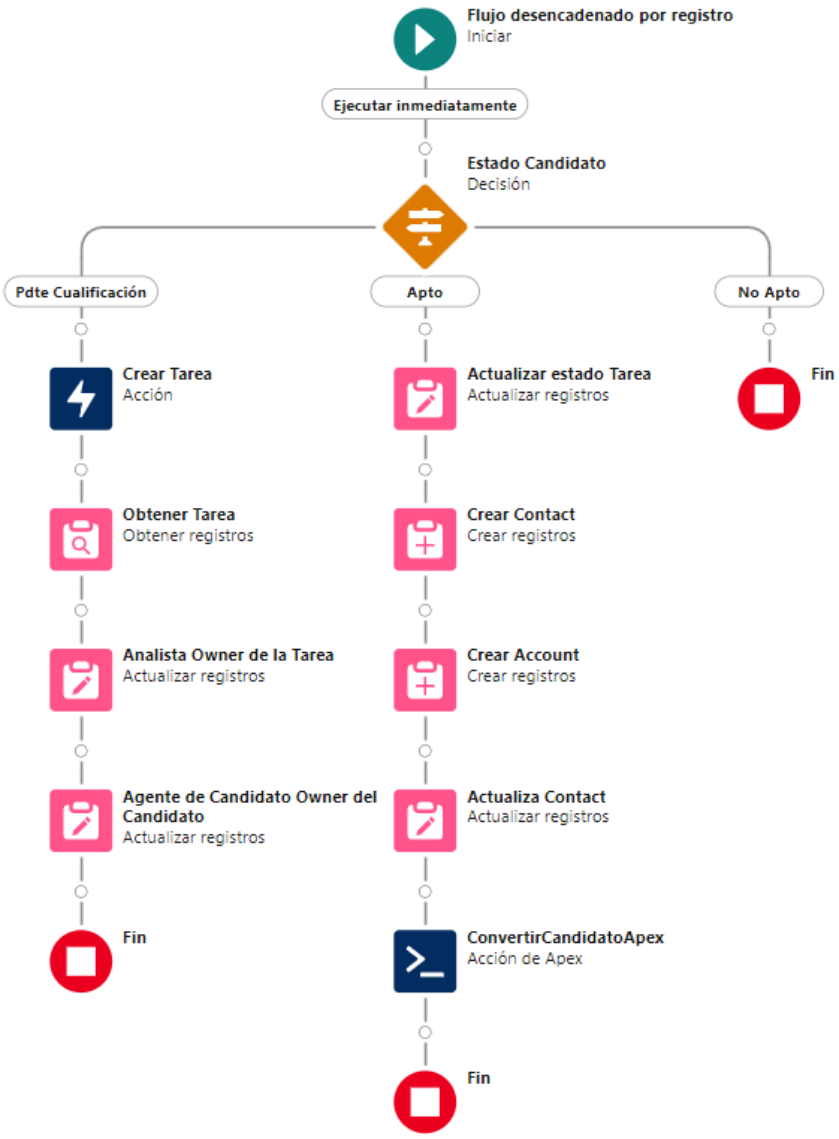


Figura 3-7. Flujo “Lead: Gestión Candidato”.

Además, para el objeto Lead, deben crearse las siguientes reglas de validación:

- Dirección obligatoria: Debe informar todos los campos de la dirección para que esta sea válida.
- Formato documentación: El número de documento debe tener 8 dígitos seguidos de una letra mayúscula.
- Formato referencia catastral: La referencia catastral debe seguir uno de los formatos válidos.
- Formato teléfono y móvil: Los números de teléfono y móvil deben tener 9 dígitos y comenzar con los prefijos correctos para España.

Por último, dada la importancia de la coherencia de datos en el registro de Candidato, se ha implementado un desencadenador que se ejecuta antes de la inserción de un nuevo candidato. Este *trigger* realiza las siguientes verifica que el estado del candidato sea "Pendiente cualificación" tras la creación del mismo, y valida el código postal y la provincia del candidato:

```
trigger TriggerCandidato on Lead (before insert) {

    // mapa estatico para almacenar los prefijos de codostal y sus provincias
    Map<String, String> mapaProvPrefijo= new Map<String, String>{
        '04' => 'Almería',
        '11' => 'Cádiz',
        '14' => 'Córdoba',
        '18' => 'Granada',
        '21' => 'Huelva',
        '23' => 'Jaén',
        '29' => 'Málaga',
        '41' => 'Sevilla'
    };

    // iteramos sobre los leads nuevos para validar
    for (Lead lead : Trigger.new) {

        // verificar que el estado sea "Pendiente cualificación" cuando se inserta el candidato
        if (lead.Status != 'Pendiente cualificación') {
            lead.addError('Cuando se inserta un nuevo candidato, su estado debe ser "Pendiente cualificación".');
        }

        // validar el cp con la provincia correspondiente
        if (lead.PostalCode != null && lead.City != null) {

            // comprobar longitud del cp
            if (lead.PostalCode.length() != 5 || !Pattern.matches('^[0-9]{5}$', lead.PostalCode)) {
                lead.addError('El código postal debe tener 5 dígitos.');

            } else {
                // cogemos dos primeros digitos del cp
                String prefijoCodPostal = lead.PostalCode.substring(0, 2);
                // buscamos la provincia correspondiente en el mapa
                String provinciaCorrecta = mapaProvPrefijo.get(prefijoCodPostal);

                if (provinciaCorrecta != null && lead.City != provinciaCorrecta) {
                    lead.addError('El código postal no coincide con la provincia ingresada. Debe ser: ' + provinciaCorrecta);
                } else if (provinciaCorrecta == null) {
                    lead.addError('El código postal ingresado no es válido en Andalucía.');
                }
            }
        }
    }
}
```

Figura 3-8. Desencadenador “TriggerCandidato”.

- Caso de uso 2: Gestión de clientes

El siguiente paso en el proceso de ventas es gestionar los clientes de la empresa. Para ello se utiliza el objeto estándar de Salesforce “Account”, cuyos campos definidos en el modelo de datos almacenan la información del expediente del cliente.

El registro de cuenta creado tras la conversión del cliente llevará asociado un contacto principal o presidente de la comunidad, dependiendo del tipo de cliente que se trate. Este contacto podrá modificarse desde el botón “Modificar Contacto”, que insertará un nuevo registro del objeto estándar “Contact” en la base de datos del sistema y lo asociará al registro de cuenta. Para la creación del contacto, se utiliza una nueva acción en el objeto Account y el flujo “Account: Actualizar Contacto Principal” relaciona este nuevo contacto con la cuenta:

Cuenta Acción
Modificar Contacto

Valores de campo predefinidos [6]

Detalle de Acción Modificar | Eliminar | Modificar formato

Etiqueta	Modificar Contacto	Nombre de objeto	Cuenta
Tipo de etiqueta estándar		Tipo de acción	Crear un registro
Nombre	Modificar_Contacto	Campo de relación	Nombre de la cuenta
Descripción	Modificar Contacto Principal de Account		
Objeto de destino	Contacto	Icono	
Crear elemento de noticias en tiempo real	✓		
Mensaje de operación correcta	Se ha modificado el Contacto		
Creado por	Sara Cabeza Muñoz, 23/8/2024, 22:48	Modificado por	Sara Cabeza Muñoz, 23/8/2024, 22:48

Modificar | Eliminar | Modificar formato

Valores de campo predefinidos Nuevo

Acción	Nombre de campo	Nombre de la API	Tipo de campo	Valor
Modificar | Eliminar	Ciudad de correo	MailingCity	Texto	Account.BillingCity
Modificar | Eliminar	País de correo	MailingCountry	Texto	Account.BillingCountry
Modificar | Eliminar	Código postal de correo	MailingPostalCode	Texto	Account.BillingPostalCode
Modificar | Eliminar	Estado o provincia de correo	MailingState	Texto	Account.BillingState
Modificar | Eliminar	Calle de correo	MailingStreet	Área de texto	Account.BillingStreet
Modificar | Eliminar	Cargo	Title	Texto	'Nuevo Contacto'

Figura 3-9. Acción (Account) “Modificar Contacto”.

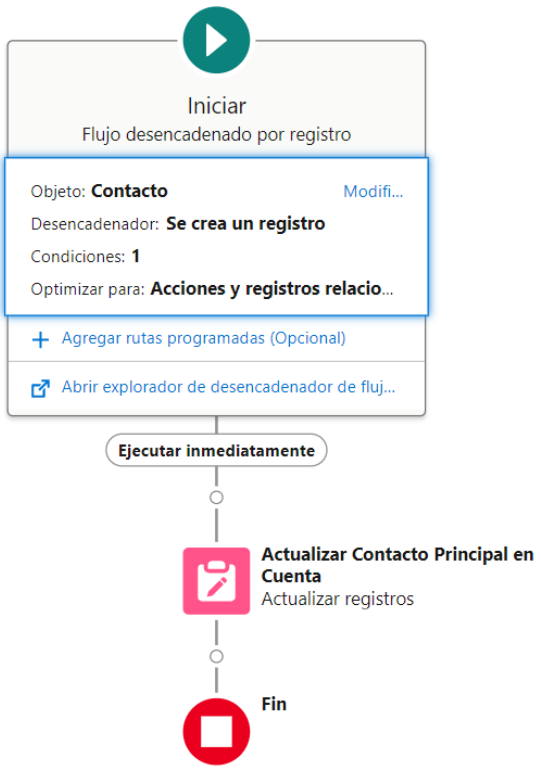


Figura 3-10. Flujo “Account: Actualizar Contacto Principal”.

Además, dado que el objeto Cuenta es el elemento central del modelo de datos del sistema, es vital garantizar que se informen todos los datos necesarios para poder formalizar la relación comercial con oportunidades. Para ello, debe implementarse un flujo que verifique la validez de la información ingresada y, en caso de que todos los campos estén informados (se evaluarán diferentes campos según el tipo de cuenta), se informará un check. En función de esta casilla de datos validados, un nuevo desencadenador permitirá la creación de oportunidades si los datos están validados.

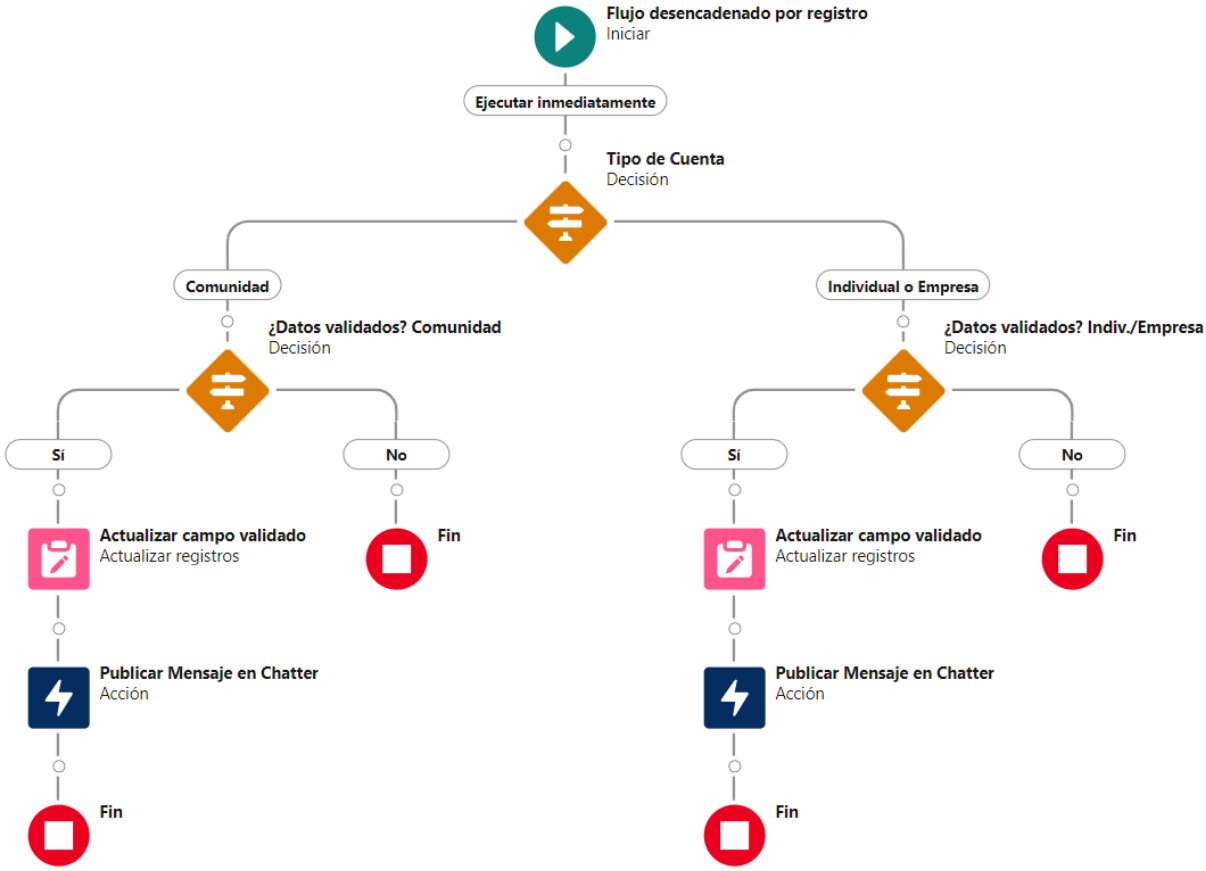


Figura 3-11. Flujo “Account: Validar Datos”

```
trigger ComprobarDatosValidados on Opportunity (before insert) {

    // almacenamos los id de las cuentas
    Set<Id> accountIds = new Set<Id>();

    // iteramos sobre las oportunidades que se quieren crear
    for (Opportunity opp : Trigger.new) {
        // añadimos el id de la cuenta relacionada
        accountIds.add(opp.AccountId);
    }

    // mapa para consultar las cuentas relacionadas
    Map<Id, Account> accountsMap = new Map<Id, Account>([SELECT Id, Datos_validados__c FROM Account WHERE Id IN :accountIds]);

    // iteramos sobre las oportunidades para ver el campo "Datos Validados"
    for (Opportunity opp : Trigger.new) {
        // comprobamos si la cuenta relacionada tiene el check marcado
        if (accountsMap.containsKey(opp.AccountId) && !accountsMap.get(opp.AccountId).Datos_validados__c) {
            // si no está marcado, añadir un mensaje de error cuando se intente crear la oportunidad
            opp.addError('No se puede crear la oportunidad porque los datos de la cuenta no están validados.');
        }
    }
}
```

Figura 3-12. Desencadenador “ComprobarDatosValidados”.

Cuando los datos de un cliente se validen, este debe reflejarse en el mapa de cuentas de la empresa:

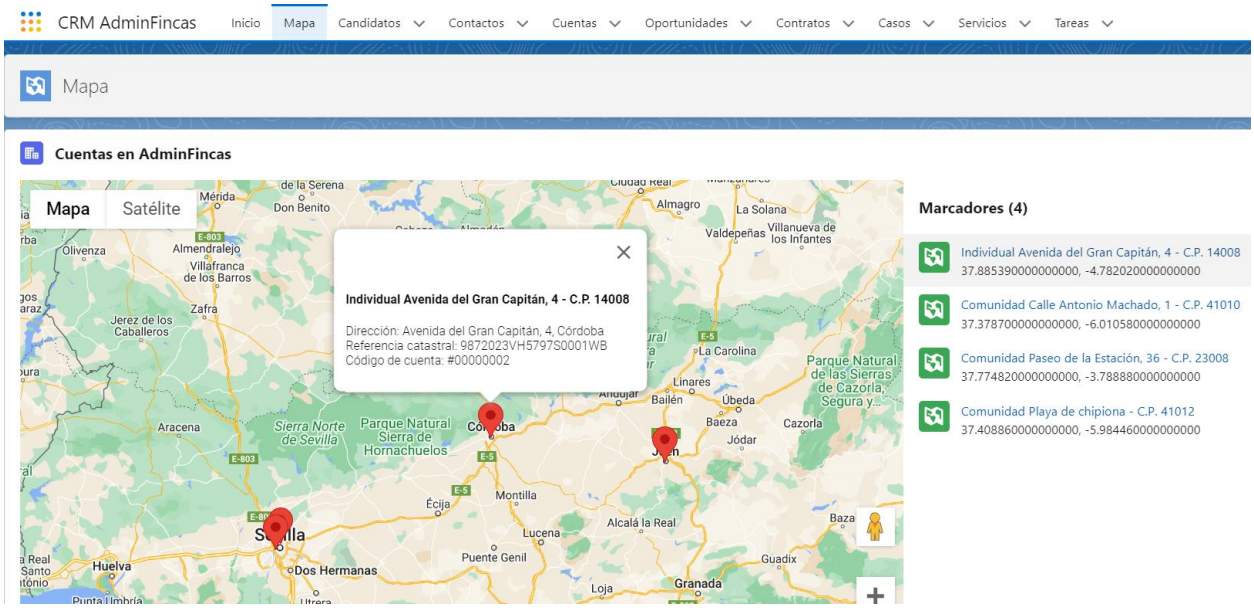


Figura 3-13. Mapa de cuentas validadas de AdminFincas.

Para la creación de este mapa, debe implementarse un componente que localice las cuentas que tengan el check informado y las muestre por pantalla.

```
<aura:component controller="MapaCuentasController" implements="flexipage:availableForAllPageTypes">

    <aura:attribute name="mapMarkers" type="Object[]" /> <!-- atributo para almacenar los marcadores del mapa -->
    <aura:attribute name="zoomLevel" type="Integer" default="7" /> <!-- atributo para el zoom del mapa -->
    <aura:attribute name="center" type="Object"
                    default="{'location': {'Latitude': '37.5', 'Longitude': '-4.7'}}" /> <!-- atributo para definir el centro del mapa Andalucía -->

    <aura:handler name="init" value="{!this}" action="{!c.doInit}" /> <!-- handler que se ejecuta al inicializar el componente -->

    <lightning:card title="Cuentas en AdminFincas" iconName="standard:account"> <!-- tarjeta lightning que va a contener el mapa -->
        <div class="slds-m-around_medium">
            <lightning:map
                mapMarkers="{!v.mapMarkers}"
                zoomLevel="{!v.zoomLevel}"
                center="{!v.center}" /> <!-- componente de mapa de lightning que muestra los marcadores -->
        </div>
    </lightning:card>
</aura:component>
```

Figura 3-14. Estructura del componente Lightning mapaCuentas.

```
({
    doInit : function(component, event, helper) {
        helper.loadAccounts(component); // llama al helper para cargar las cuentas
    }
})
```

Figura 3-15. Controlador JavaScript del componente mapaCuentas.

```
({
    // cargar las cuentas
    loadAccounts : function(component) {
        var action = component.get("c.getAccounts"); //llamada al método apex para obtener las cuentas
        action.setCallback(this, function(response) {
            var state = response.getState();
            if (state === "SUCCESS") {
                var accounts = response.getReturnValue();
                var mapMarkers = accounts.map(function(account) {
                    return {
                        location: {
                            Latitude: account.BillingLatitude,
                            Longitude: account.BillingLongitude
                        },
                        title: account.Name,
                        description: 'Dirección: ' + account.BillingStreet + ', ' + account.BillingCity +
                                     '<br/>Referencia catastral: ' + (account.Referencia_catastral__c || 'N/A') +
                                     '<br/>Código de cuenta: ' + (account.C_digo_de_Cuenta__c || 'N/A')
                    };
                });
                component.set("v.mapMarkers", mapMarkers); // se establecem los marcadores en el componente
            } else {
                console.error('Error al cargar las cuentas', response.getError());
            }
        });
        $A.enqueueAction(action);
    }
})
```

Figura 3-16. Helper JavaScript del componente mapaCuentas.

```
public with sharing class MapaCuentasController {
    @AuraEnabled(cacheable=true)
    public static List<Account> getAccounts() {
        try {
            // consulta para obtener las cuentas
            List<Account> cuentas = [SELECT Id, Name, BillingStreet, BillingCity, BillingLatitude, BillingLongitude, Referencia_catastral__c, C_digo_de_Cuenta__c
                                     FROM Account
                                     WHERE BillingLatitude != null AND BillingLongitude != null AND Datos_validados__c = true];

            if (cuentas.isEmpty()) {
                System.debug('No se encontraron cuentas con coordenadas geográficas.');
            } else {
                System.debug('Se han encontrado: ' + cuentas.size() + ' cuentas.');
            }
            return cuentas;

        } catch (Exception e) {
            // errores
            System.debug('Error al obtener las cuentas: ' + e.getMessage());
            throw new AuraHandledException('Error al obtener las cuentas: ' + e.getMessage());
        }
    }
}
```

Figura 3-17. Clase Apex MapaCuentasController para recuperar cuentas validadas.

- Caso de uso 3: Gestión de oportunidades

Una vez formalizados los datos del cliente en sus registros, se procederá a la creación de oportunidades de negocio. Para ello, se va a utilizar el objeto estándar "Opportunity" de Salesforce, que permite realizar un seguimiento de la información actualizada de la relación comercial indicando su estado actual, servicios a contratar, fecha de cierre, etc.

El proceso para cerrar con éxito la oportunidad pasa por diferentes etapas antes de que la oportunidad sea definitiva. Estas etapas son:

- **Pendiente Gestión**: Etapa inicial con la que se crea una oportunidad. El gestor asignado debe poder asegurarse de que los datos iniciales son correctos antes de avanzar la oportunidad.
- **En Negociación**: En esta etapa inicia el proceso de negociación con el cliente sobre los servicios que desea contratar. El gestor debe seleccionar los servicios del catálogo y modificar el precio base de estos según los acuerdos con los clientes. Es vital que los servicios que se muestren por pantalla estén vigentes y estén orientados al tipo de cuenta del cliente relacionado.

Para ello es necesario un nuevo componente:

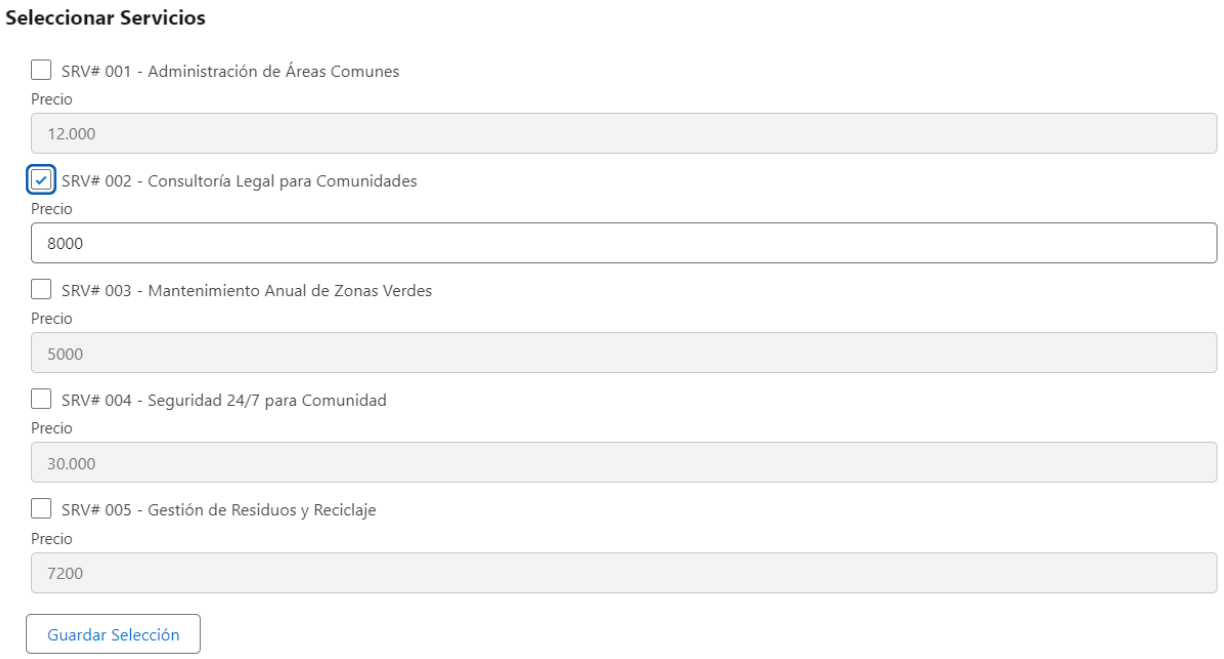


Figura 3-18. Selector de servicios en la oportunidad.

```
<aura:component controller="ServicioSelectorController" implements="flexipage:availableForRecordHome,force:hasRecordId">
    <aura:attribute name="recordId" type="Id" />
    <aura:attribute name="servicios" type="Object[]" />
    <aura:attribute name="selectedServicios" type="Object[]" />

    <aura:handler name="init" value="{!this}" action="{!c.doInit}" />

    <lightning:card title="Seleccionar Servicios">
        <div class="slds-m-around_medium">
            <aura:iteration items="{!v.servicios}" var="servicio">
                <div class="slds-p-around_xx-small">
                    <lightning:input type="checkbox"
                                     label="{!servicio.C_digo_de_Servicio__c + ' - ' + servicio.Name}"
                                     name="{!servicio.Id}"
                                     checked="{!servicio.selected}"
                                     onchange="{!c.handleSelection}" />
                    <lightning:input type="number"
                                     label="Precio"
                                     value="{!servicio.customPrice}"
                                     name="{!servicio.Id}"
                                     onchange="{!c.handlePriceChange}"
                                     disabled="{!!servicio.selected}" />
                </div>
            </aura:iteration>
            <lightning:button label="Guardar Selección"
                              onclick="{!c.saveSelection}"
                              class="slds-m-top_small" />
        </div>
    </lightning:card>
</aura:component>
```

Figura 3-19. Estructura del componente Lightning servicioSelector.

```
({
    doInit : function(component, event, helper) {
        helper.getServicios(component);
    },

    handleSelection : function(component, event, helper) {
        var servicios = component.get("v.servicios");
        var servicioId = event.getSource().get("v.name");
        var isSelected = event.getSource().get("v.checked");

        servicios.forEach(function(servicio) {
            if (servicio.Id === servicioId) {
                servicio.selected = isSelected;
            }
        });

        component.set("v.servicios", servicios);
    },

    handlePriceChange : function(component, event, helper) {
        var servicios = component.get("v.servicios");
        var servicioId = event.getSource().get("v.name");
        var newPrice = event.getSource().get("v.value");

        servicios.forEach(function(servicio) {
            if (servicio.Id === servicioId) {
                servicio.customPrice = newPrice;
            }
        });

        component.set("v.servicios", servicios);
    },

    saveSelection : function(component, event, helper) {
        helper.saveServiciosToOpportunity(component);
    }
})
```

Figura 3-20. Controlador JavaScript del componente servicioSelector.

```
({
    getServicios : function(component) {
        var action = component.get("c.getServicios");
        action.setParams({
            opportunityId: component.get("v.recordId")
        });
        action.setCallback(this, function(response) {
            var state = response.getState();
            if (state === "SUCCESS") {
                var servicios = response.getReturnValue();
                servicios.forEach(function(servicio) {
                    servicio.selected = false;
                    servicio.customPrice = servicio.Precio__c;
                });
                component.set("v.servicios", servicios);
            } else {
                this.showToast("Error", "Error al obtener servicios", "error");
            }
        });
        $A.enqueueAction(action);
    },

    saveServiciosToOpportunity : function(component) {
        var servicios = component.get("v.servicios");
        var selectedServicios = servicios.filter(function(servicio) {
            return servicio.selected;
        }).map(function(servicio) {
            return {
                id: servicio.Id,
                precio: servicio.customPrice
            };
        });

        var action = component.get("c.updateOpportunityServices");
        action.setParams({
            opportunityId: component.get("v.recordId"),
            serviciosJSON: JSON.stringify(selectedServicios)
        });
        action.setCallback(this, function(response) {
            var state = response.getState();
            if (state === "SUCCESS") {
                this.showToast("Éxito", "Servicios actualizados correctamente", "success");
                this.refreshView(component);
            } else {
                this.showToast("Error", "Error al actualizar servicios", "error");
            }
        });
        $A.enqueueAction(action);
    },

    showToast : function(title, message, type) {
        var toastEvent = $A.get("e.force:showToast");
        toastEvent.setParams({
            title: title,
            message: message,
            type: type
        });
        toastEvent.fire();
    },
    refreshView : function(component) {
        $A.get('e.force:refreshView').fire();
        // volvemos a cargar los servicios
        this.getServicios(component);
    }
})
```

Figura 3-21. Helper JavaScript del componente servicioSelector.

```
public with sharing class ServicioSelectorController {

    @AuraEnabled
    public static List<Servicio__c> getServicios(Id opportunityId) {
        Opportunity opp = [SELECT Id, Account.Type, AccountId FROM Opportunity WHERE Id = :opportunityId];
        String accountType = opp.Account.Type;

        List<Servicio__c> servicios = new List<Servicio__c>();
        if (accountType == 'Individual' || accountType == 'Comunidad' || accountType == 'Empresa') {
            servicios = [SELECT Id, Name, Precio__c, C_digo_de_Servicio__c
                         FROM Servicio__c
                         WHERE RecordType.Name =: accountType];
        }
        System.debug('servicios: ' + servicios);
        return servicios;
    }

    @AuraEnabled
    public static void updateOpportunityServices(Id opportunityId, String serviciosJSON) {
        System.debug('Entrada - opportunityId: ' + opportunityId);
        System.debug('Entrada - serviciosJSON: ' + serviciosJSON);

        try {
            Object deserializedObj = JSON.deserializeUntyped(serviciosJSON);
            List<Map<String, Object>> serviciosSeleccionados = new List<Map<String, Object>>();

            if (deserializedObj instanceof List<Object>) {
                for (Object item : (List<Object>)deserializedObj) {
                    if (item instanceof Map<String, Object>) {
                        serviciosSeleccionados.add((Map<String, Object>)item);
                    }
                }
            } else if (deserializedObj instanceof Map<String, Object>) {
                serviciosSeleccionados.add((Map<String, Object>)deserializedObj);
            }

            System.debug('Servicios seleccionados: ' + serviciosSeleccionados);

            Opportunity opp = [SELECT Id, Servicios_Seleccionados__c, Total_Servicios__c, StageName FROM Opportunity WHERE Id = :opportunityId];

            Set<Id> servicioIds = new Set<Id>();
            for (Map<String, Object> servicio : serviciosSeleccionados) {
                servicioIds.add((Id)servicio.get('id'));
            }
            Map<Id, Servicio__c> serviciosMap = new Map<Id, Servicio__c>([
                SELECT Id, C_digo_de_Servicio__c, Name, Precio__c
                FROM Servicio__c
                WHERE Id IN :servicioIds
            ]);

            List<String> serviciosFormateados = new List<String>();
            Decimal totalServicios = 0;
            for (Map<String, Object> servicio : serviciosSeleccionados) {
                Id servicioId = (Id)servicio.get('id');
                Servicio__c servicioCompleto = serviciosMap.get(servicioId);
                Decimal precioPersonalizado = Decimal.valueOf(String.valueOf(servicio.get('precio')));

                String servicioStr = 'Código: ' + servicioCompleto.C_digo_de_Servicio__c + '\n' +
                                     'Nombre: ' + servicioCompleto.Name + '\n' +
                                     'Precio Base: ' + servicioCompleto.Precio__c + '\n' +
                                     'Precio Personalizado: ' + precioPersonalizado;
                serviciosFormateados.add(servicioStr);

                totalServicios += precioPersonalizado;
            }

            opp.Servicios_Seleccionados__c = String.join(serviciosFormateados, '\n\n');
            opp.Total_Servicios__c = totalServicios;
            opp.StageName = 'Negociación'; // aseguramos que la etapa se mantenga en "Negociación"

            System.debug('Servicios formateados: ' + opp.Servicios_Seleccionados__c);
            System.debug('Total de servicios: ' + opp.Total_Servicios__c);

            update opp;
            System.debug('Oportunidad actualizada con éxito');
        } catch (Exception e) {
            System.debug('Error al actualizar la oportunidad: ' + e.getMessage());
            System.debug('Línea del error: ' + e.getLineNumber());
            System.debug('Traza de la pila: ' + e.getStackTraceString());
            throw new AuraHandledException('Error al actualizar la oportunidad: ' + e.getMessage());
        }
    }
}
```

Figura 3-22. Clase Apex “ServicioSelectorController” para recuperar los servicios

- **Pendiente Aprobación**: Una vez que se ha llegado a un acuerdo preliminar con el cliente y se tienen todos los servicios seleccionados, debe avanzar a esta etapa. Aquí se lanza el proceso de aprobación interno que determinará si una oportunidad será ganada o perdida, con aprobación unánime. Los aprobadores deben recibir una notificación por correo:

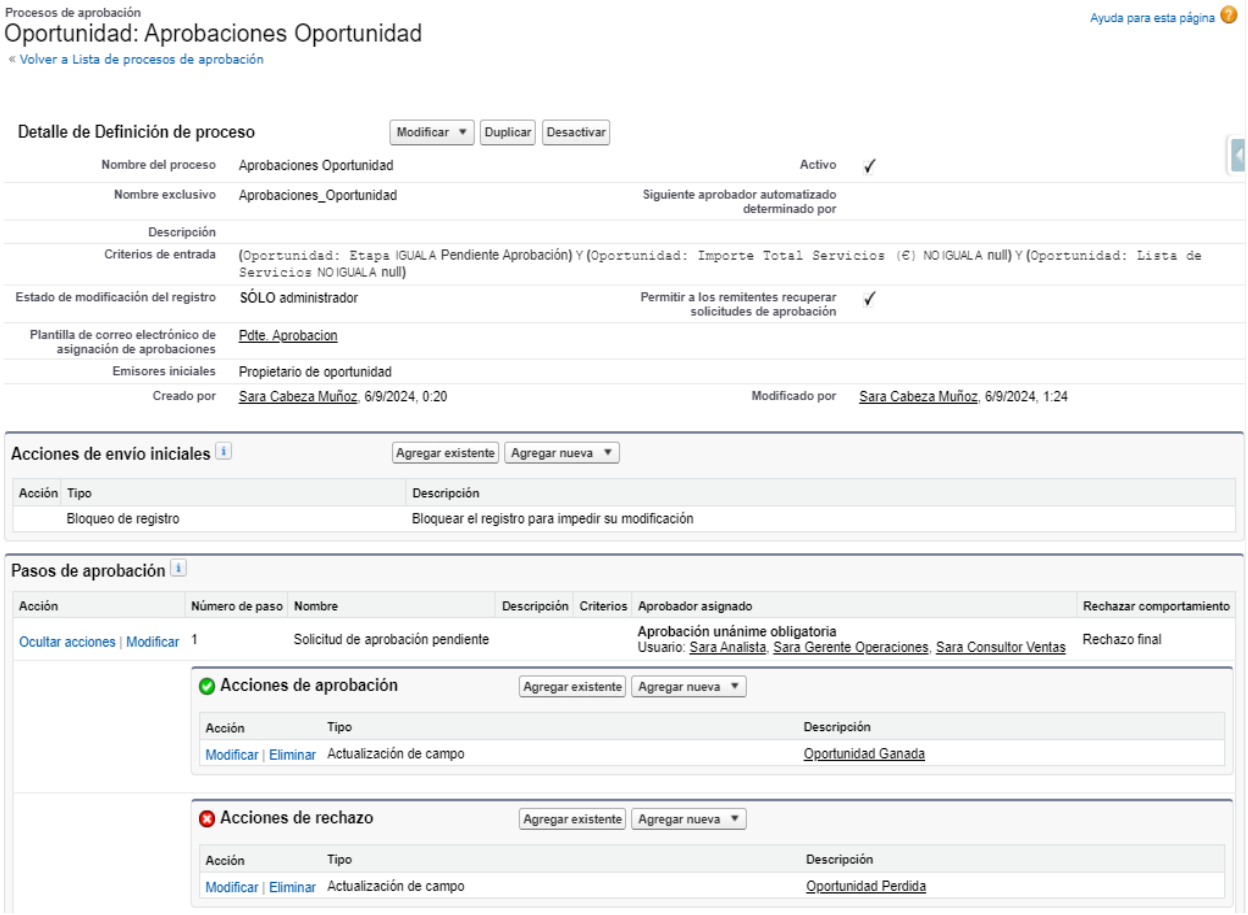


Figura 3-23. Proceso de aprobación “Oportunidad: Aprobaciones Oportunidad”.

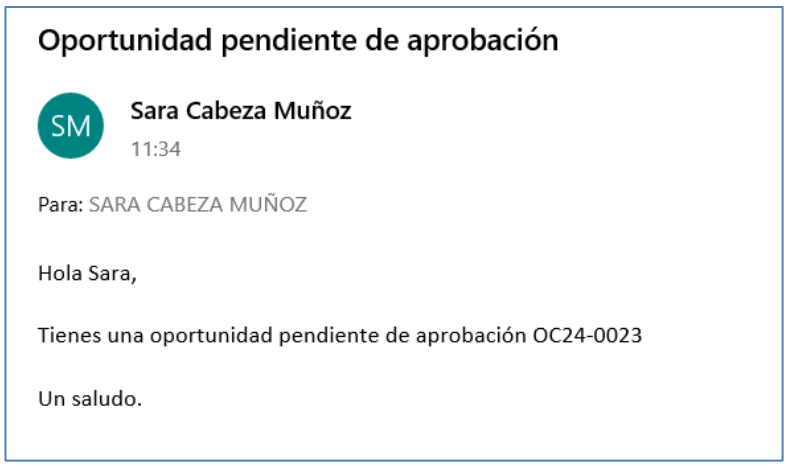


Figura 3-24. Notificación tarea de aprobación “Oportunidad pendiente de aprobación”.

- **Cerrada/Ganada:** Si las aprobaciones son favorables y se decide proceder, la oportunidad pasa a esta etapa. Aquí se finaliza el proceso de venta con éxito y se procede a la generación del contrato con el cliente.
- **Cerrada /Perdida:** Si se determina que la oportunidad no es beneficiosa para la empresa y se rechaza una de las tareas de aprobación pendientes, la oportunidad se marca como perdida.

Para la gestión global de este caso de uso debe utilizarse un nuevo flujo que permita las acciones descritas en las etapas, así como un desencadenador que no permita movimientos indeseados entre etapas:

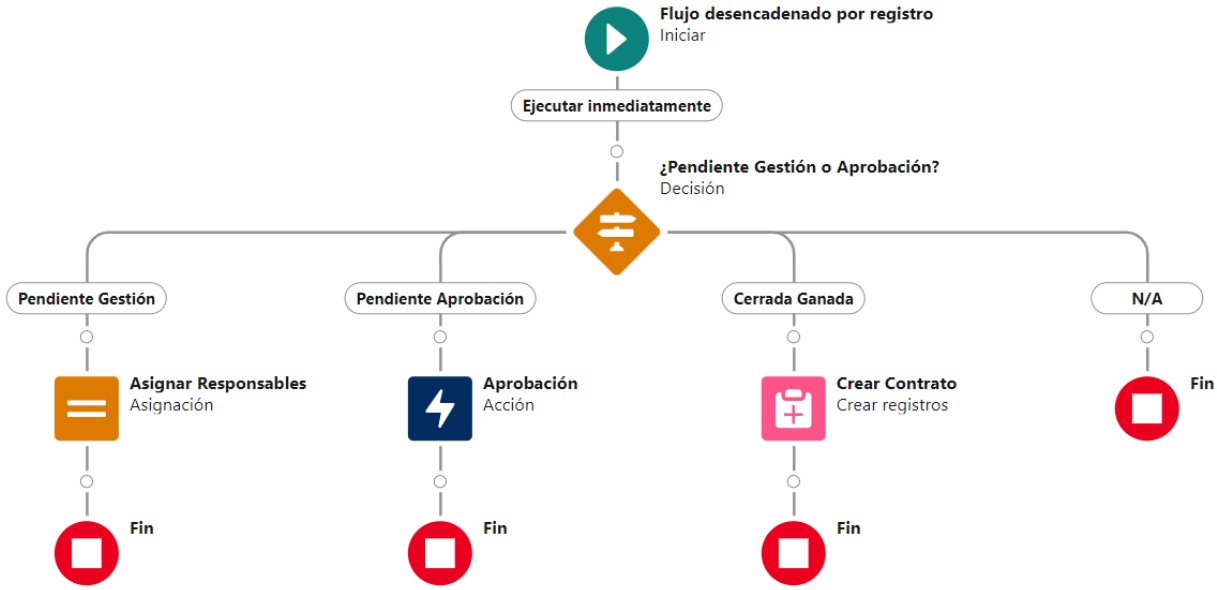


Figura 3-25. Flujo “Opportunity: Gestión Oportunidad”.

```
trigger EtapaOportunidad on Opportunity (before insert, before update) {

    for (Opportunity opp : Trigger.new) {

        // si es una oportunidad nueva
        if (Trigger.isInsert) {
            // si la etapa es distinto de "Pendiente Gestión" lanzamos error
            if (opp.StageName != 'Pendiente Gestión') {
                opp.addError('El campo Etapa solo puede ser "Pendiente Gestión" en la creación de una oportunidad.');
            }
        }

        // si se está actualizando
        if (Trigger.isUpdate) {
            // recogemos los valores antiguos de la oportunidad
            Opportunity oldOpp = Trigger.oldMap.get(opp.Id);

            // verificamos si la etapa anterior era "Perdida" o "Ganada"
            if (oldOpp.StageName == 'Perdida' || oldOpp.StageName == 'Ganada') {
                opp.addError('No se puede modificar una oportunidad que esté en "Perdida" o "Ganada".');
            }

            // flujo de estados permitido

            if (oldOpp.StageName == 'Pendiente Gestión' && opp.StageName != 'En Negociación') {
                opp.addError('La oportunidad solo puede avanzar a "En Negociación" desde "Pendiente Gestión".');
            }
            else if (oldOpp.StageName == 'En Negociación' && opp.StageName == 'Pendiente Gestión') {
                opp.addError('No se puede retroceder a "Pendiente Gestión" desde "En Negociación".');
            }
            else if (oldOpp.StageName == 'Pendiente Aprobación' && opp.StageName != 'Ganada' && opp.StageName != 'Perdida') {
                opp.addError('La oportunidad solo puede avanzar a "Ganada" o "Perdida" desde "Pendiente Aprobación".');
            }
        }
    }
}
```

Figura 3-26. Desencadenador “EtapaOportunidad”.

- Caso de uso 4: Gestión de activos

Los activos de la empresa corresponden a los contratos con sus clientes. El contrato debe generarse cuando se gana la oportunidad con estado "Nuevo" y al gestor de activos se le asignará una nueva tarea para indicarle que el contrato está pendiente de firma.

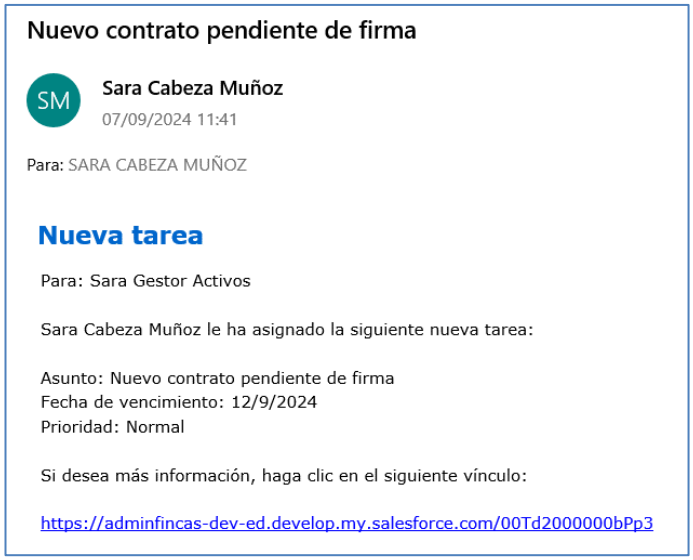


Figura 3-27. Notificación tarea "Nuevo contrato pendiente de firma".

Cuando el gestor reciba esta notificación, deben informarse todos los datos necesarios, como fecha de inicio del contrato y la duración, así como gestionar la firma del contrato (tanto del cliente como la de la compañía). Al tratarse de unos campos sensibles, se deben incluir reglas de validación:

- Fecha de firma del cliente: Debe ingresar la fecha de firma del cliente cuando el contrato ha sido firmado por el cliente.
- Fecha de firma de la compañía: Debe ingresar la fecha de firma de la compañía cuando el contrato ha sido firmado por un representante de la compañía.
- Fecha de inicio del contrato: La fecha de inicio del contrato no puede ser anterior a la fecha actual.
- Información Comercial: Debe informar los campos Periodicidad y Medio de pago para poder activar el contrato.

Una vez informados los campos, el gestor puede actualizar el estado del contrato a "En Vigor", bloqueando el registro para que no pueda modificarse ningún campo salvo el estado, ya que el contrato debe poder anularse en cualquier momento.

Para la gestión íntegra de este caso de uso debe crearse un nuevo desencadenador, que realice lo indicado:

```
trigger TriggerContrato on Contract (after insert, before update) {
    // lista para almacenar las tareas a crear
    List<Task> tareasACrear = new List<Task>();

    // si se está insertando
    if (Trigger.isInsert && Trigger.isAfter) {
        // iteramos sobre cada contrato
        for (Contract c : Trigger.new) {
            // crea una nueva tarea
            Task t = new Task(
                WhatId = c.Id,                                  // asocia la tarea al contrato
                Subject = 'Nuevo contrato pendiente de firma',  // asunto de la tarea
                ActivityDate = Date.today().addDays(5),         // fecha vencimiento de la tarea (hoy + 5 días)
                OwnerId = '005d2000001TfnBAAS'                  // ssignamos al gestor
            );
            tareasACrear.add(t);
        }

        // inserta las tareas creadas
        if (!tareasACrear.isEmpty()) {
            insert tareasACrear;
        }
    }

    // si se está actualizando
    if (Trigger.isUpdate && Trigger.isBefore) {
        // iteramos sobre cada contrato
        for (Contract newContract : Trigger.new) {
            Contract oldContract = Trigger.oldMap.get(newContract.Id);

            // comprobamos si el contrato está siendo activado
            if (newContract.Status == 'En Vigor' && oldContract.Status != 'En Vigor') {
                // comprobamos fecha inicio y duración del contrato y si no están informadas, error
                if (newContract.StartDate == null || newContract.ContractTerm == null) {
                    newContract.addError('No se puede poner en vigor el contrato sin fecha de inicio y duración.');
                } else if (newContract.CompanySignedId == null || newContract.CustomerSignedId== null) {
                    // si no están las firmas, error
                    newContract.addError('No se puede poner en vigor el contrato sin las firmas de ambas partes.');
                }
            }

            // comprobamos si el contrato está en vigor o anulado
            if (oldContract.Status == 'En Vigor' || oldContract.Status == 'Anulado') {
                // verificamos si se está intentando modificar algún campo que no sea Status
                for (String field : Schema.SObjectType.Contract.fields.getMap().keySet()) {
                    if (field != 'Status' && newContract.get(field) != oldContract.get(field)) {
                        // si se intenta modificar un campo que no sea Status, error
                        newContract.addError('No se puede modificar el contrato cuando está en vigor o anulado, salvo para actualizar su estado.');
                        break;
                    }
                }
            }
        }
    }
}
```

Figura 3-28. Desencadenador “TriggerContrato”.

- Caso de uso 5: Gestión de casos

En este caso de uso, nos ocupamos del servicio de atención al cliente. Para ello, utilizamos el objeto estándar de Salesforce “Case”, que recoge la información registrada por el cliente de la petición o reclamación a tratar. Este objeto nos permite vincular al cliente con la consulta del servicio de atención al cliente.

Los casos se diferencian por los campos tipo y motivo, que permiten clasificarlos atendiendo a su prioridad: si el tipo es “Petición” y el motivo es “Mantenimiento”, el caso se creará con prioridad “Alta”; si se tratase de una “Reclamación” y el motivo fuese “Mantenimiento”, el caso se crearía con prioridad “Media”. El resto de motivos se tratarán con prioridad “Baja”.

Cuando se cree un caso, además de clasificarlo, el sistema debe asignarle una tarea al agente de atención al cliente para que este, cuando revise y solucione el caso, pueda cerrarlo. Este proceso debe realizarse a través de un nuevo flujo:

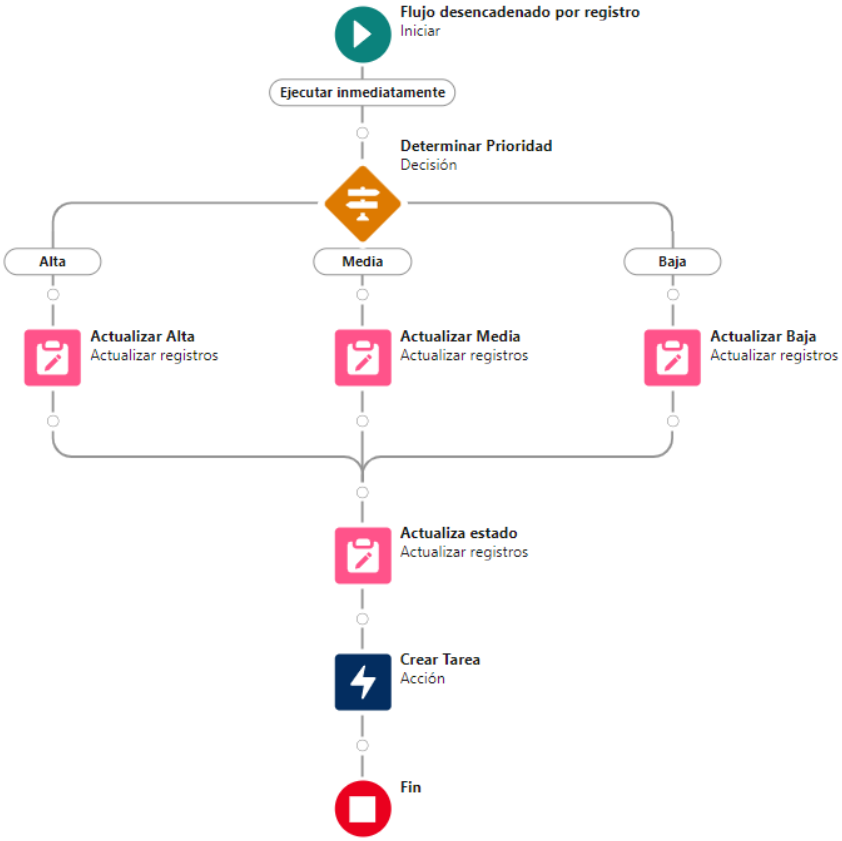


Figura 3-29. Flujo “Case: Gestión Caso”.

4 IMPLEMENTACIÓN DEL ESCENARIO

La implementación realizada en el caso de estudio se ha descrito en los apartados anteriores indicando los fragmentos de código y automatismos más relevantes del entorno de trabajo. En este apartado se realizarán comentarios y se expondrán las herramientas utilizadas, así como los pasos que se han llevado a cabo durante el desarrollo e implementación del escenario.

4.1 Consideraciones previas

Debido a que se trata de una demo con la versión gratuita de Salesforce, únicamente se disponen de dos licencias para usuarios, es decir, solo pueden estar activos dos usuarios, uno el administrador del sistema y a través de la otra licencia se ha ido activando y desactivando usuarios para comprobar el funcionamiento de este CRM. Además, el almacenamiento es muy limitado, al igual que la cantidad de objetos y registros que se pueden gestionar. Tampoco se tienen disponibles algunas funcionalidades avanzadas, lo que restringe las capacidades del sistema para simular un entorno empresarial más completo. Debido a estas limitaciones, ha sido necesario priorizar las funcionalidades esenciales del CRM, enfocándose en los procesos básicos de gestión y automatización, y adaptando el desarrollo a los recursos disponibles en esta versión.

4.2 Herramientas utilizadas en el desarrollo

Las herramientas que se han usado para el desarrollo han sido las siguientes:

- **Salesforce Inspector** es una extensión de navegador diseñada para facilitar la gestión de datos en Salesforce. Permite explorar y editar registros, exportar datos a formatos como CSV, y ejecutar consultas SOQL directamente desde el navegador. También ofrece una visión clara de la estructura y metadatos de los objetos en Salesforce, ayudando a los usuarios a entender mejor la organización y configuración de los datos. En esencia, Salesforce Inspector proporciona una manera más eficiente y directa de interactuar con los datos en Salesforce, lo que es especialmente útil para administradores y desarrolladores.
- **VSCode (Visual Studio Code) con Salesforce CLI** es una combinación efectiva para el desarrollo y gestión de aplicaciones en Salesforce. VSCode, un editor de código se complementa con Salesforce CLI, una herramienta de línea de comandos que facilita la administración y automatización de tareas en Salesforce. Juntas, permiten a los desarrolladores escribir y depurar código de Apex, Visualforce, y Lightning Web Components de manera eficiente, gestionar metadatos, ejecutar pruebas, y automatizar procesos, todo desde un entorno integrado y accesible. Esta combinación optimiza el flujo de trabajo y mejora la productividad en el desarrollo de aplicaciones Salesforce.
- **Developer Console para Apex** es una herramienta basada en la web en Salesforce que permite a los desarrolladores escribir, ejecutar y depurar código Apex de manera interactiva. Ofrece funcionalidades clave como la ejecución de bloques de código Apex, la depuración con puntos de interrupción y rastreo en tiempo real, y la visualización de logs de ejecución detallados. También facilita la ejecución de consultas SOQL y SOSL para recuperar datos, así como la gestión y revisión de pruebas unitarias. En resumen, proporciona un entorno integral para desarrollar, probar y solucionar problemas en el código Apex dentro de la plataforma Salesforce.
- **Salesforce Trailhead** es una plataforma de aprendizaje en línea que ofrece cursos interactivos y módulos para enseñar a los usuarios sobre Salesforce, desde fundamentos básicos hasta habilidades avanzadas. Proporciona rutas de aprendizaje estructuradas, preparación para certificaciones oficiales, y desafíos prácticos para aplicar conocimientos en entornos reales. Los usuarios ganan puntos y medallas por completar lecciones, lo que gamifica el aprendizaje y motiva el progreso. Además, Trailhead incluye una comunidad activa y recursos adicionales para apoyar el desarrollo profesional continuo. En

resumen, Trailhead es una herramienta integral para adquirir y mejorar habilidades en Salesforce de manera interactiva y accesible.

4.3 Desarrollo

Una vez presentadas las herramientas que se han utilizado para desarrollar el proyecto, en este apartado se explicarán los pasos realizados para el desarrollo.

El primer paso consiste en crear una organización en Salesforce, lo que supone un espacio en la nube dedicado a nuestra empresa, donde se almacenan todos los datos de los clientes, como cuentas, contactos, oportunidades, casos, y otros datos personalizados relevantes para el negocio.

Se trata de una Developer Org, es decir, una organización de desarrollo. A diferencia de otras, estas organizaciones son gratuitas y están diseñadas para el desarrollo de aplicaciones y funcionalidades personalizadas, lo que las convierte en una opción ideal para llevar a cabo la solución propuesta.

A continuación, se presenta una imagen de la página donde se creó la organización:

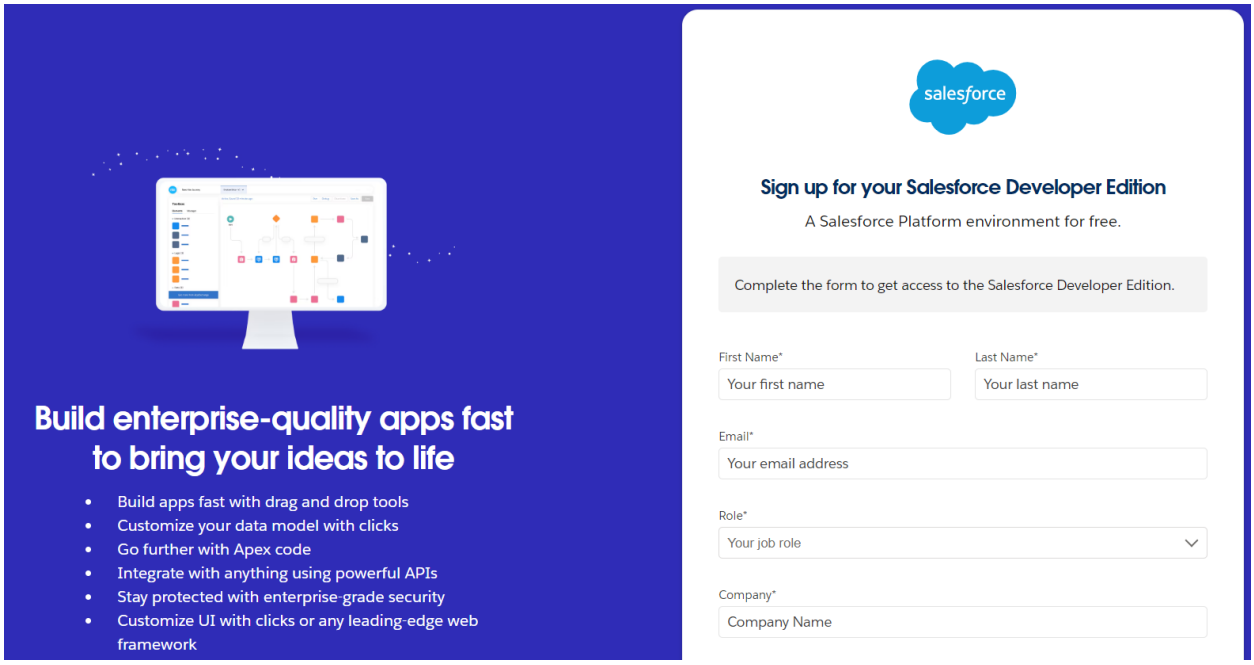


Figura 4-1. Salesforce Developer Org.

Llevar a cabo el diseño de un proyecto en Salesforce supone un proceso estructurado que abarca varias etapas esenciales para garantizar un despliegue exitoso. A continuación, se van a exponer los pasos que se han seguido para la implementación de nuestro proyecto en Salesforce de la mano de la empresa AdminFincas.

1. Preparación del entorno de trabajo

Tras crear la organización en Salesforce como se ha mencionado antes, se configuró un entorno apto para nuestro trabajo con los datos y ajustes necesarios, como cuentas, contactos y oportunidades, para asegurarnos que las configuraciones básicas estuvieran listas antes de avanzar. Además, se definieron los requisitos funcionales y se alineó al equipo de desarrollo con los *stakeholders* con el fin de garantizar una capacitación adecuada que preparase el terreno para una ejecución fluida y coordinada del proyecto.

2. Creación del modelo de datos

Salesforce facilita la creación de modelos de datos complejos sin necesidad de programación, lo que nos permite estructurar la información de una manera más eficiente. Por tanto, podemos crear objetos personalizados directamente en la plataforma, definiendo campos, configurando relaciones entre ellos y añadiendo reglas de

validación para asegurar la coherencia y la integridad de los datos. Gracias a las herramientas que Salesforce nos proporciona, este procedimiento resulta rápido y sencillo, ya que se puede llevar a cabo solamente con unos clics.

3. Desarrollo de la lógica de negocio con Apex

Ahora comienza la parte más técnica del desarrollo. Para implementar reglas de negocio complejas, utilizando Apex, el lenguaje de programación de Salesforce, se han creado *Triggers* para responder a eventos clave, como la creación o actualización de registros. También se ha usado Apex para el manejo de componentes. Para cumplir con los requisitos de Salesforce, se han desarrollado clases test que aseguraran al menos un 75% de cobertura del código, garantizando así la fiabilidad y correcto funcionamiento del sistema bajo diversas condiciones.

4. Creación de interfaces de usuario

Para diseñar interfaces de usuario intuitivas y adaptadas a las necesidades de los usuarios, se ha utilizado el Lightning App Builder para desarrollar páginas Lightning personalizadas, creando vistas específicas para cada perfil de usuario y asegurando un acceso rápido a la información relevante. Además, se han creado componentes Lightning reutilizables que mejoran la experiencia del usuario y optimizan la disposición de la información.
Para una personalización más avanzada, también se ha utilizado Visualforce, consiguiendo una interfaz amigable que satisficiera las necesidades del proyecto.

5. Pruebas y validación

Este apartado se expondrá con más detalle en la siguiente sección del documento, pero esencialmente se trata de realizar una serie de pruebas para asegurar que todo funciona correctamente. Una vez validado el sistema, se puede proceder al despliegue de la solución CRM.

5 PUESTA EN MARCHA Y PRUEBAS

Una vez se ha finalizado el desarrollo de la solución, resulta indispensable atender a su implementación. La puesta en marcha del proyecto y las diferentes pruebas que se deben llevar a cabo son esenciales para asegurar que las funcionalidades aplicadas funcionan según los requisitos establecidos, confirmando que el sistema está listo para su uso.

5.1 Configuración

En situaciones para empresas reales, cuando se administra una organización, es común contar con múltiples entornos *Sandbox*, cada uno dedicado a diferentes fases del proceso de desarrollo, como pruebas y validación. Para estos entornos, uno de los usos más comunes es el de desarrollo y pruebas, donde se realiza la mayor parte de ajustes antes de implementar los cambios en un entorno de producción.

No obstante, en este caso particular, al tratarse de un trabajo de fin de grado realizado en una *Developer Edition*, no ha sido necesario configurar entornos adicionales. Dado que esta edición de Salesforce proporciona un entorno suficientemente flexible para manejar tanto el desarrollo como las pruebas, todas las actividades relacionadas con el proyecto se han llevado a cabo dentro de un único entorno operativo, lo que ha simplificado significativamente el proceso. Esto ha permitido concentrar todos los esfuerzos en el desarrollo del proyecto sin la necesidad de gestionar múltiples entornos.

5.2 Pruebas de funcionamiento

En este capítulo se presentan las pruebas realizadas para el sistema, organizadas en una tabla para facilitar su comprensión. Para cada prueba diseñada, se indica el resultado esperado, junto con el resultado real obtenido al ejecutar la prueba, lo que permite verificar si la prueba ha sido exitosa o no. Se pueden encontrar capturas de pantalla que justifican el resultado real en el Anexo del documento.

Prueba	Resultado esperado	Resultado
Candidato		
Crear candidato	El candidato se crea correctamente	✅
Validar que la dirección sea obligatoria	Error al no ingresar dirección	✅
Validar que los apellidos sean obligatorios	Error al no ingresar apellidos	✅
Validar que el Nº de documento sea obligatorio	Error al no ingresar número de documento	✅
Validar formato correcto de Ref catastral	Error al ingresar formato incorrecto de Ref catastral	✅
Validar formato correcto de teléfono y móvil	Error al ingresar formato incorrecto de teléfono/móvil	✅
Verificar que los campos del candidato estén rellenos	Error si hay campos obligatorios vacíos	✅
Comprobar que el candidato no apto no crea cuenta ni contacto	No se crean cuenta ni contacto si el candidato es no apto	✅
Validar que al pasar a apto se cree cuenta y contacto	Se crean correctamente cuenta y contacto al pasar a apto	✅

Prueba	Resultado esperado	Resultado
Verificar la creación de tarea (correo) al cambiar estado	Se crea una tarea de correo automáticamente	✅
Cuenta		
Validar que Ref catastral sea obligatoria en cuenta	Error al no ingresar Ref catastral	✅
Modificar una cuenta existente	La cuenta se modifica correctamente	✅
Crear nueva oportunidad sin validación de datos	Se crea la oportunidad sin verificar datos	✅
Crear nueva oportunidad con validación de datos	Se crea la oportunidad tras la validación exitosa	✅
Verificar localización en mapa para cuenta	Se visualiza correctamente la localización en el mapa	✅
Oportunidad		
Crear oportunidad en estado "Pendiente de gestión"	Se crea la oportunidad con estado "Pendiente de gestión"	✅
Cambiar fase a "Negociación"	La oportunidad pasa a fase de "Negociación"	✅
Incluir un servicio a la oportunidad	El servicio se añade correctamente a la oportunidad	✅
Modificar importe de un servicio e incluir otro	Se actualiza el importe e incluye el nuevo servicio correctamente	✅
Cambiar fase a "Pendiente de aprobación"	La fase cambia a "Pendiente de aprobación" correctamente	✅
Aprobaciones (general)	Las aprobaciones se procesan correctamente	✅
Aprobación de oportunidades	La oportunidad se aprueba correctamente	✅
Contrato		
Generación de contrato	El contrato se genera correctamente	✅
Verificar que todos los datos obligatorios estén completos en el contrato	Error al faltar datos obligatorios en el contrato	✅
Comprobar que el contrato esté en vigor	El contrato entra en vigor correctamente	✅
Verificar que no se pueden modificar datos en contrato vigente	No se permite modificar los datos del contrato una vez en vigor	✅
Caso		
Crear un caso	El caso se crea correctamente	✅
Verificar que los datos obligatorios estén presentes en el caso	Error al faltar datos obligatorios en el caso	✅

Prueba	Resultado esperado	Resultado
Asignar prioridad alta y estado pendiente en un caso	El caso se asigna con prioridad alta y estado "Pendiente"	✅
Asignar caso a Atención al cliente y verificar creación de tarea	El caso se asigna a Atención al cliente y se crea la tarea	✅
Actualizar caso a estado "Cerrado"	El caso se actualiza correctamente a "Cerrado"	✅

Tabla 5-1. Pruebas funcionales.

5.3 Análisis en base a los datos

Tras la ejecución exitosa de todas las pruebas de validación, los resultados demuestran que el sistema desarrollado cumple con los requisitos establecidos y funciona correctamente, concluyendo que las validaciones de datos, flujos y automatizaciones están operando según lo esperado.

A continuación, se plantean varias ideas orientadas hacia la mejora del sistema actual y la optimización de procesos:

- Integración de Salesforce en sistemas de gestión de propiedades: Consiste en analizar cómo puede integrarse la plataforma Salesforce en sistemas específicos de gestión de propiedades para sincronizar la información de los datos.
- Conexión de Salesforce y sistemas de contabilidad para la administración de fincas: Explorar cómo puede interactuar Salesforce con un software de contabilidad para gestionar las finanzas y la comprobación de pagos en la administración de fincas.
- Integración de Salesforce con Plataformas de Gestión de Documentos: Comprende el estudio sobre cómo se puede conectar Salesforce con herramientas de gestión de documentos (como DocuSign o Dropbox) para almacenar, compartir y gestionar documentos relacionados con la gestión inmobiliaria.

6 CONCLUSIONES

El estudio que se ha llevado a cabo refleja la gran importancia de los sistemas CRM a la hora de gestionar las relaciones con los clientes hoy en día; especialmente cuando se utilizan soluciones como Salesforce. Su uso facilita la gestión de datos y la automatización de procesos, optimizando las relaciones con los clientes y aumentando el grado de satisfacción de los mismos. A medida que las plataformas CRM han ido evolucionando desde sus primeras aplicaciones locales hasta los sistemas actuales basados en la nube, con sus ventajas de actualización automática y menores costes de mantenimiento, se han visto favorecidas por empresas de todos los calibres, ya que aumentan la accesibilidad y flexibilidad de la gestión empresarial.

Salesforce, en concreto, ofrece una plataforma escalable permitiendo desarrollar soluciones tecnológicas personalizadas. Su capacidad para integrar múltiples áreas de negocio, como ventas, marketing y servicios, en un entorno colaborativo y accesible subraya su importancia en el panorama empresarial actual, en el que la personalización y la eficacia operativa son claves para el éxito.

Por tanto, Salesforce se posiciona como la mejor opción para la implantación de un CRM en cualquier compañía. La plataforma ofrece una amplia gama de herramientas que permiten la automatización de procesos, el análisis de datos y la creación de soluciones personalizadas, adaptándose a las necesidades específicas de cada empresa. Además, su sistema modular permite a las organizaciones seleccionar las funciones que realmente necesitan, evitando costes innecesarios y garantizando una implantación escalable a medida que crece el negocio. Su arquitectura en la nube permite un acceso seguro y remoto a la información, facilitando la colaboración en equipo y la gestión de las relaciones con los clientes desde cualquier lugar.

Salesforce es especialmente útil en el sector de la gestión de inmuebles. La posibilidad de gestionar los contactos y las relaciones con los propietarios en un único lugar simplifica la comunicación y mejora el servicio al cliente. Las herramientas de automatización optimizan la eficiencia operativa del administrador de propiedades al encargarse de tareas recurrentes como la gestión de oportunidades, la supervisión del mantenimiento y la resolución de problemas.

El objetivo principal de este proyecto consistía en desarrollar un entorno CRM basado en Salesforce que otorgase a una empresa dedicada a la administración de fincas la capacidad de gestionar su modelo de negocios en un mismo entorno laboral. Dicho objetivo ha sido alcanzado puesto que el entorno de trabajo se ha desarrollado al completo según se ha plasmado en el diseño y es totalmente funcional para la compañía ficticia AdminFincas.

Por todo esto, quedo satisfecha con el sistema desarrollado. Aunque es cierto que se pueden plantear futuras líneas de desarrollo, la aplicación es totalmente funcional y está lista para su implantación real.

REFERENCIAS

Adminstark. (2024, 22 mayo). Métodos Ágiles en el Desarrollo de Software | Blog | StarkCloud. *StarkCloud*. https://www.starkcloud.com/starkcloud-blog/metodos-agiles-en-el-desarrollo-de-software

Callejo, D., López, F., Pérez, J., & Vidal, J. M. V. (s. f.). *E-REdING. Biblioteca de la Escuela Superior de Ingenieros de Sevilla.* https://biblus.us.es/bibing/proyectos/abreproy/93319

Daniel, M. E. F., De València Departamento de Sistemas Informáticos y Computación - Departament de Sistemes Informàtics I Computació, U. P., & De València Escola Tècnica Superior D'Enginyeria Informàtica, U. P. (2022, 16 septiembre). *Solución CRM para compañía de seguros en Salesforce*. https://riunet.upv.es/handle/10251/186058

Editions & Pricing - Service Cloud Lightning. (s. f.). Salesforce. https://www.salesforce.com/eu/editions-pricing/service-cloud/?bc=HA

Elisa, R. S., & De Alcalá Escuela Politécnica Superior, U. (2022). *Desarrollo de sistema de gestión de incidencias con la herramienta de CRM Salesforce, a través de varios canales*. E_BUAH. Biblioteca Digital Universidad de Alcalá. https://ebuah.uah.es/dspace/handle/10017/52850

Filipsson, F., & Filipsson, F. (2024, 26 mayo). Oracle Sales Cloud Licensing and Pricing Guide. *Reveal Compliance - Oracle Licensing Experts - Independent Oracle licensing and cloud experts*. https://revealcompliance.com/oracle-sales-cloud-pricing-revealed/

Help and training community. (s. f.). Salesforce. https://help.salesforce.com/s/?language=es

Manuel, G. M. J. (s. f.). *Holderfy, un marketplace para administradores de fincas y proveedores de suministros para comunidades de propietarios | Archivo Digital UPM*. https://oa.upm.es/75011/

Oracle CRM On Demand current release. (s. f.). https://www.oracle.com/cx/what-is-crm/crmondemand-current-release/

¿Qué es la CRM? — Gestión de las relaciones con los clientes. (s. f.). Salesforce. https://www.salesforce.com/es/learning-centre/crm/what-is-crm/?bc=HA

¿Qué es un CRM y cómo funciona? (s. f.). Salesforce. https://www.salesforce.com/mx/crm/?bc=HA

Salesforce. (s. f.). *Resources for Salesforce Developers*. Salesforce Developers. https://developer.salesforce.com/

Salesforce Sales Pricing. (s. f.). Salesforce. https://www.salesforce.com/eu/sales/pricing/

SAP Service Cloud solution | CRM service and support software. (s. f.-a). SAP. https://www.sap.com/products/crm/service-cloud.html

SAP Service Cloud solution | CRM service and support software. (s. f.-b). SAP. https://www.sap.com/products/crm/service-cloud.html

Staff, S. (2024, 10 julio). *The History of Salesforce*. Salesforce. https://www.salesforce.com/news/stories/the-history-of-salesforce/?bc=HA

Trailhead. (s. f.). *Trailhead | La manera divertida de aprender*. https://trailhead.salesforce.com/es/

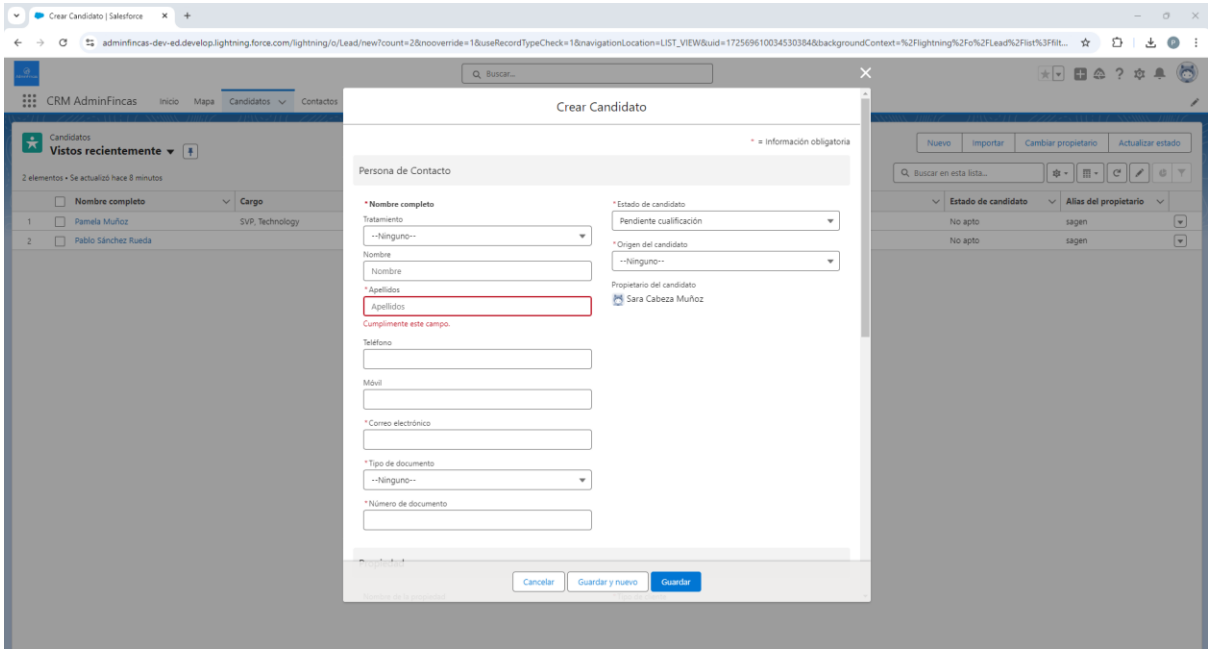
Vieira, S. (2024, 5 julio). *Administrador de fincas: qué es y qué papel tiene en la comunidad de vecinos*. Metrovacesa. https://metrovacesa.com/blog/administrador-de-fincas-que-es-y-funciones

ANEXO

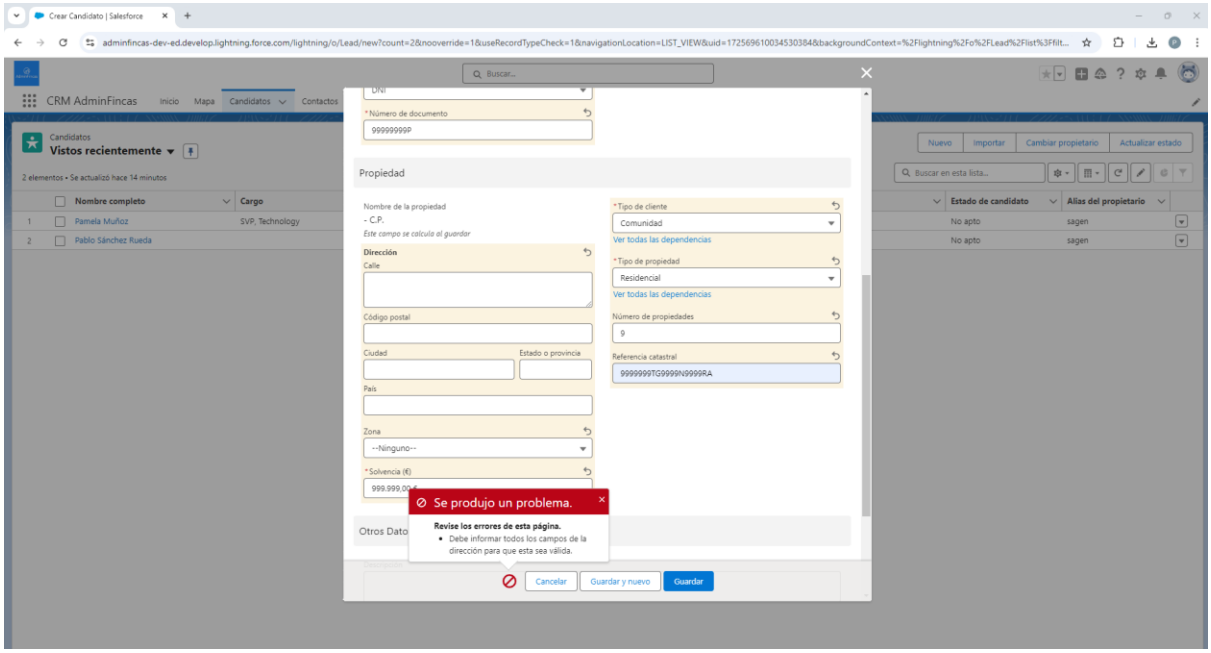
En este apartado se van a adjuntar capturas de las pruebas funcionales realizadas que validan el funcionamiento correcto del sistema:

1. CANDIDATO

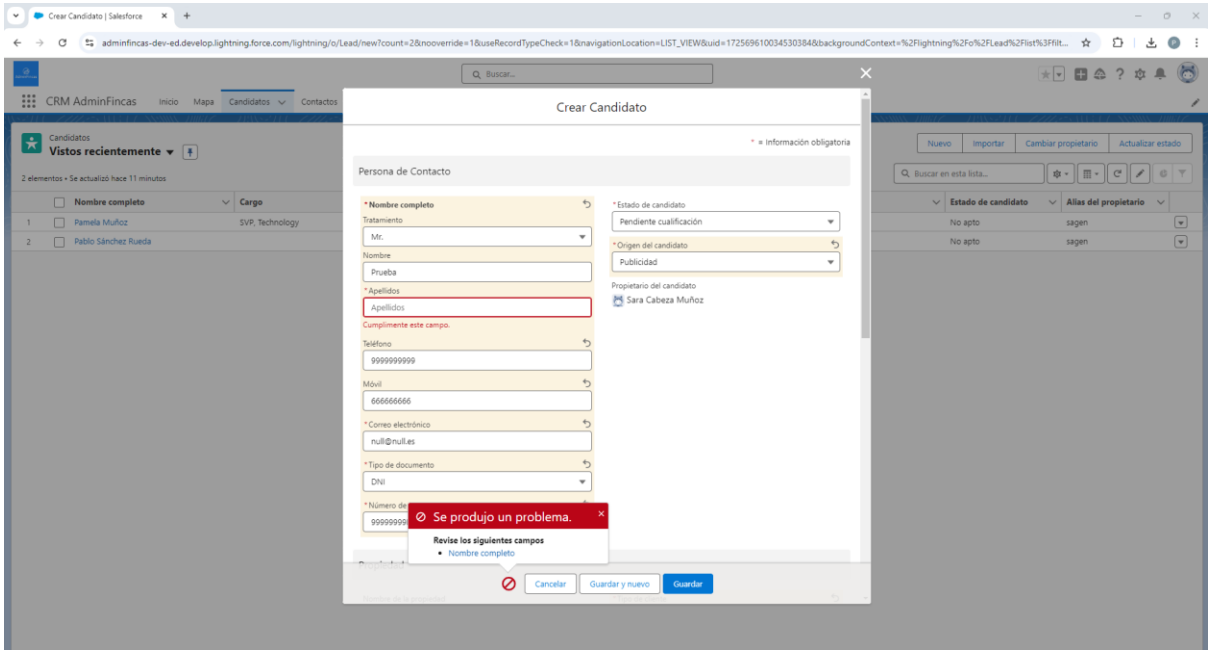
- Crear candidato



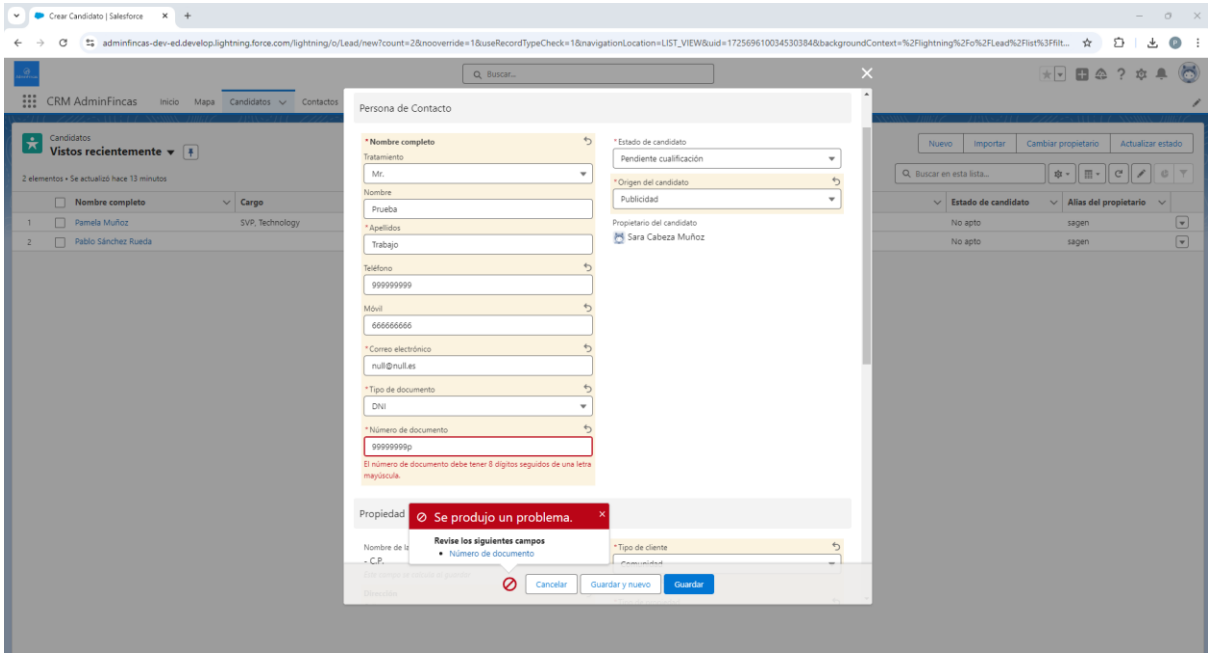
- Dirección obligatoria



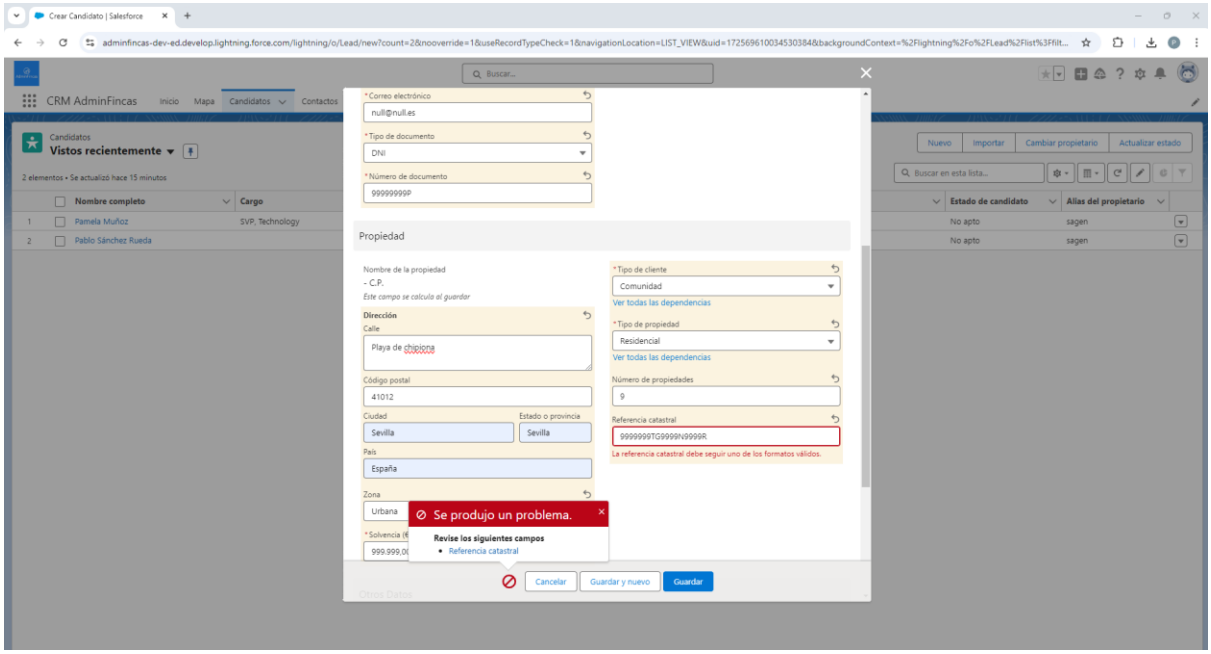
- Apellidos obligatorio



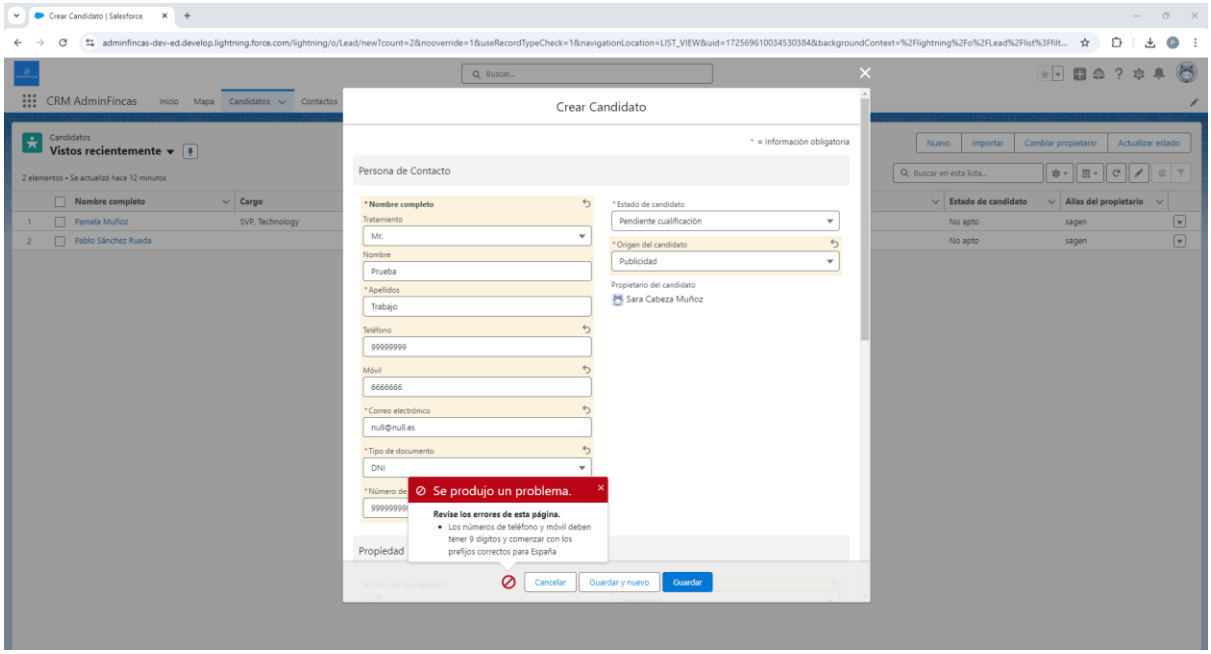
- Nº de documento obligatorio



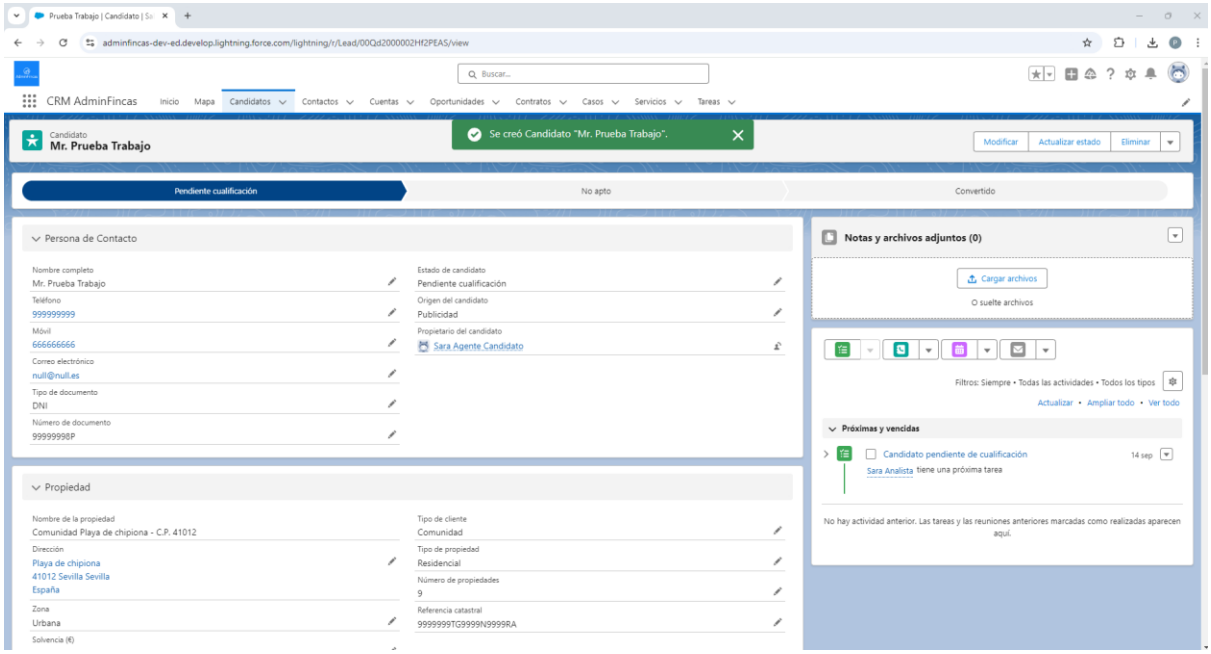
- Ref catastral con formato



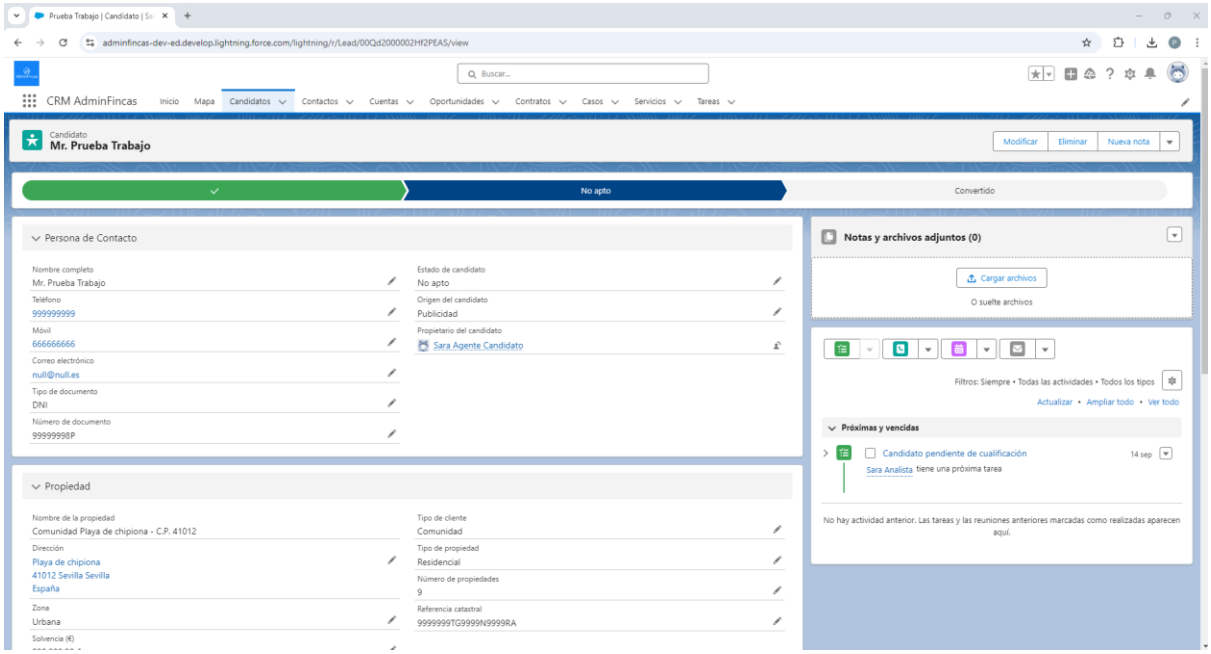
- Teléfono y móvil con formato

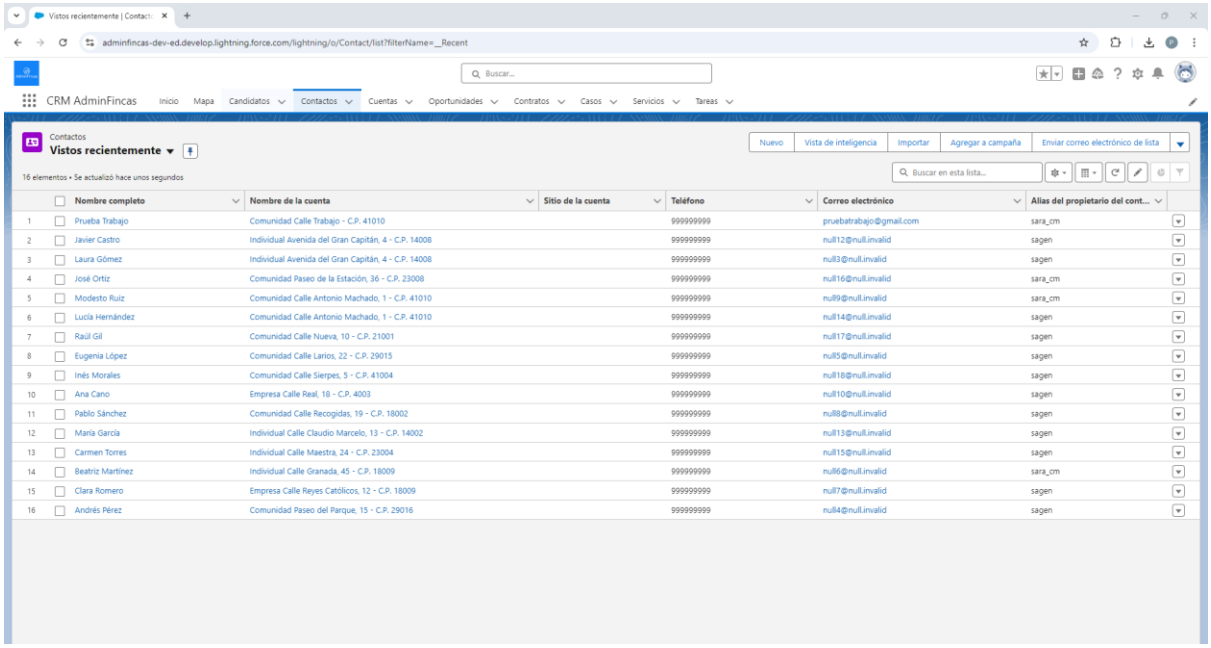


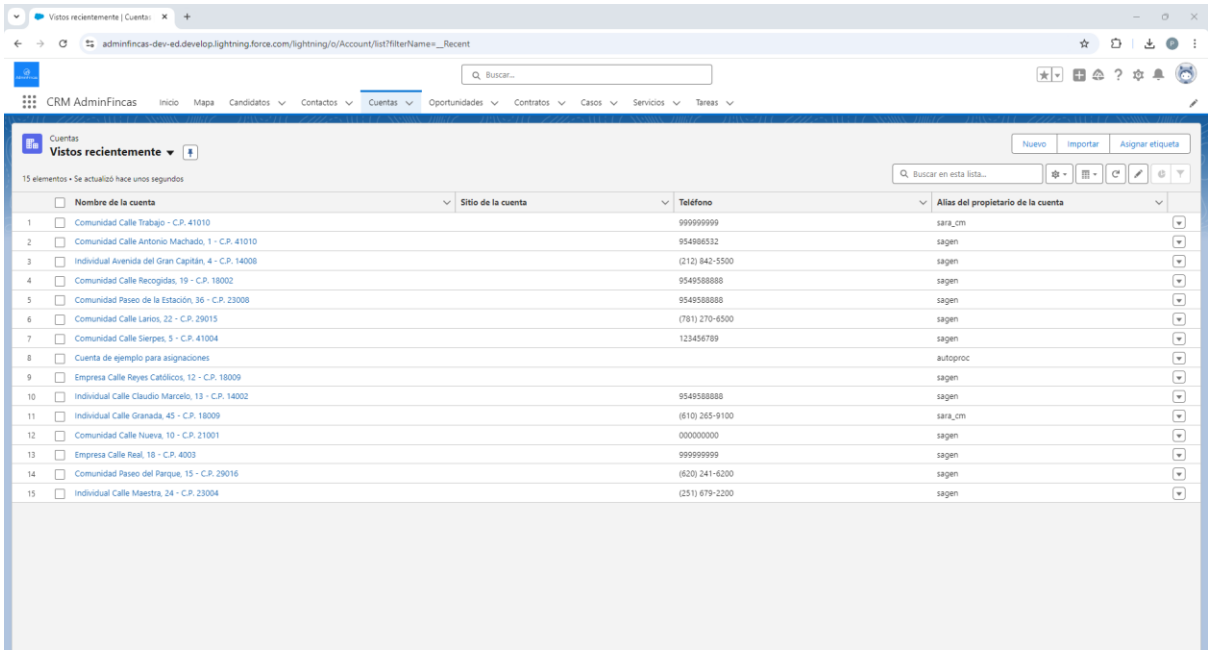
- Campos de candidatos rellenos



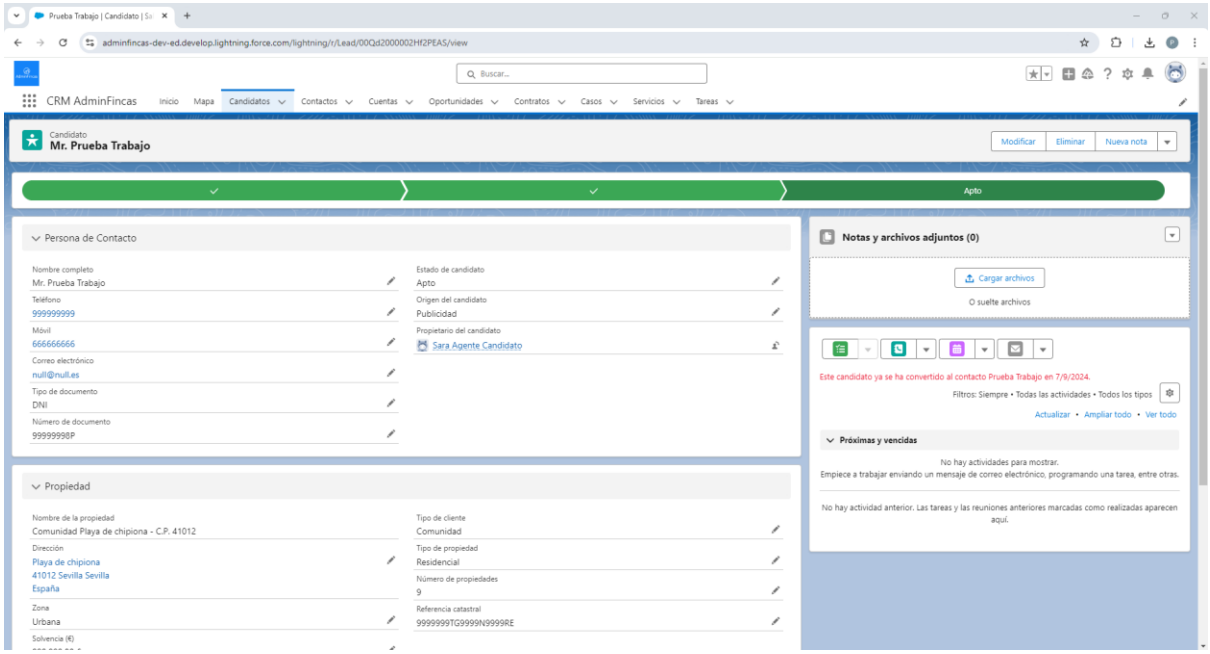
- Candidato no apto (no hay cuenta ni contacto)

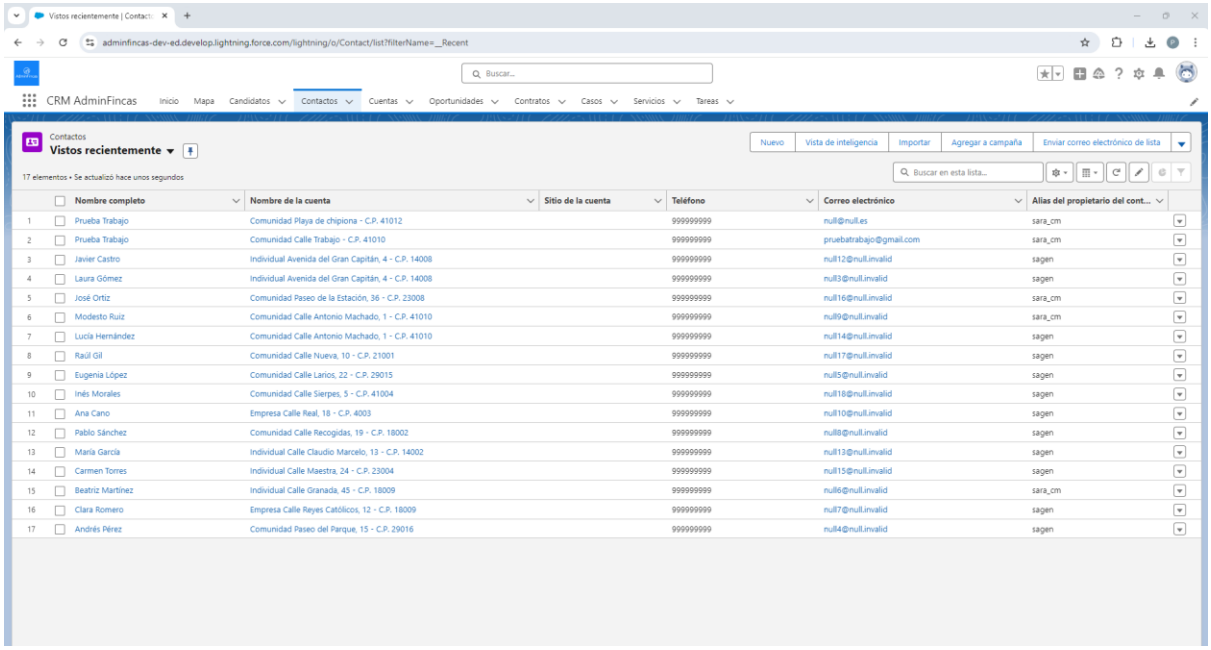


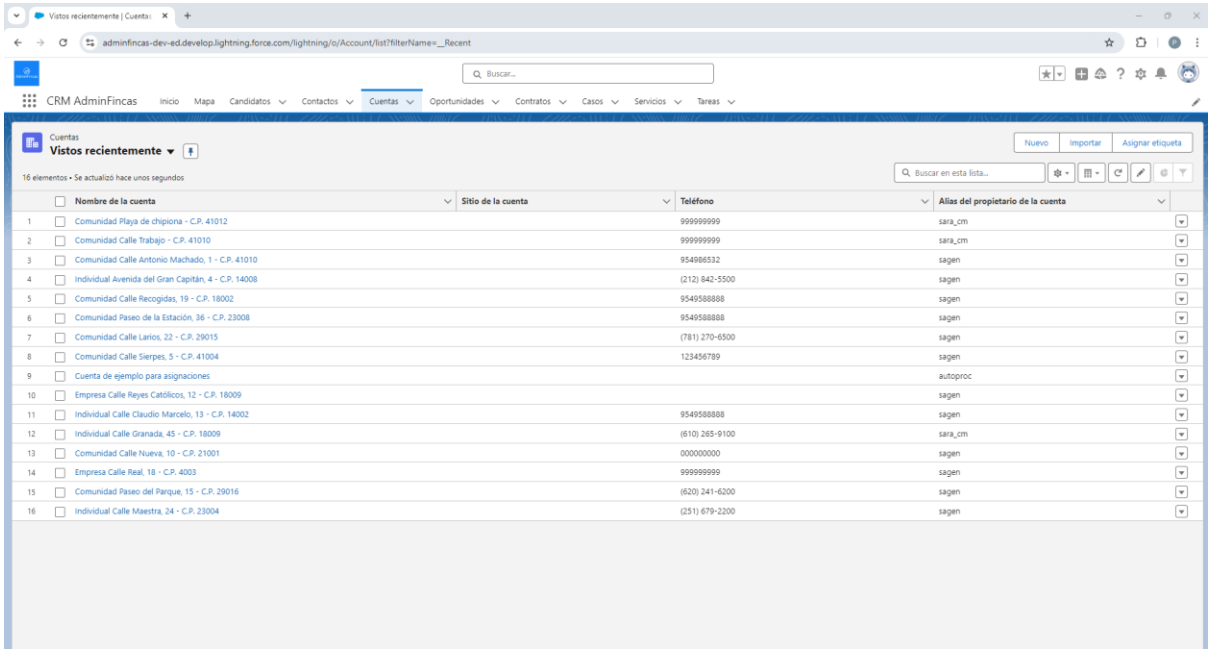

Vistos recientemente | Contacts
adminfincas-dev-ed.develop.lightning.force.com/lightning/o/Contact/list?filterName=__Recent
CRM AdminFincas
Inicio
Mapa
Candidatos
Contactos
Cuentas
Oportunidades
Contratos
Casos
Servicios
Tareas
Contactos
Vistos recientemente
Nuevo
Vista de inteligencia
Importar
Agregar a campaña
Enviar correo electrónico de lista
16 elementos • Se actualizó hace unos segundos
Buscar en esta lista...
Nombre completo
Nombre de la cuenta
Sitio de la cuenta
Teléfono
Correo electrónico
Alias del propietario del cont...
1 Prueba Trabajo Comunidad Calle Trabajo - C.P. 41010 999999999 pruebatrabajo@gmail.com sara_cm
2 Javier Castro Individual Avenida del Gran Capitán, 4 - C.P. 14008 999999999 null12@null.invalid sagen
3 Laura Gómez Individual Avenida del Gran Capitán, 4 - C.P. 14008 999999999 null3@null.invalid sagen
4 José Ortiz Comunidad Paseo de la Estación, 36 - C.P. 23008 999999999 null16@null.invalid sara_cm
5 Modesto Ruiz Comunidad Calle Antonio Machado, 1 - C.P. 41010 999999999 null9@null.invalid sara_cm
6 Lucía Hernández Comunidad Calle Antonio Machado, 1 - C.P. 41010 999999999 null14@null.invalid sagen
7 Raúl Gil Comunidad Calle Nueva, 10 - C.P. 21001 999999999 null17@null.invalid sagen
8 Eugenia López Comunidad Calle Larios, 22 - C.P. 29015 999999999 null5@null.invalid sagen
9 Inés Morales Comunidad Calle Sierpes, 5 - C.P. 41004 999999999 null18@null.invalid sagen
10 Ana Cano Empresa Calle Real, 18 - C.P. 4003 999999999 null10@null.invalid sagen
11 Pablo Sánchez Comunidad Calle Recogidas, 19 - C.P. 18002 999999999 null8@null.invalid sagen
12 María García Individual Calle Claudio Marcelo, 13 - C.P. 14002 999999999 null13@null.invalid sagen
13 Carmen Torres Individual Calle Maestra, 24 - C.P. 23004 999999999 null15@null.invalid sagen
14 Beatriz Martínez Individual Calle Granada, 45 - C.P. 18009 999999999 null6@null.invalid sara_cm
15 Clara Romero Empresa Calle Reyes Católicos, 12 - C.P. 18009 999999999 null7@null.invalid sagen
16 Andrés Pérez Comunidad Paseo del Parque, 15 - C.P. 29016 999999999 null4@null.invalid sagen


Vistos recientemente | Cuentas
adminfincas-dev-ed.develop.lightning.force.com/lightning/o/Account/list?filterName=__Recent
CRM AdminFincas
Inicio
Mapa
Candidatos
Contactos
Cuentas
Oportunidades
Contratos
Casos
Servicios
Tareas
Cuentas
Vistos recientemente
Nuevo
Importar
Asignar etiqueta
15 elementos • Se actualizó hace unos segundos
Buscar en esta lista...
Nombre de la cuenta
Sitio de la cuenta
Teléfono
Alias del propietario de la cuenta
1 Comunidad Calle Trabajo - C.P. 41010 999999999 sara_cm
2 Comunidad Calle Antonio Machado, 1 - C.P. 41010 954986532 sagen
3 Individual Avenida del Gran Capitán, 4 - C.P. 14008 (212) 842-5500 sagen
4 Comunidad Calle Recogidas, 19 - C.P. 18002 9549588888 sagen
5 Comunidad Paseo de la Estación, 36 - C.P. 23008 9549588888 sagen
6 Comunidad Calle Larios, 22 - C.P. 29015 (781) 270-6500 sagen
7 Comunidad Calle Sierpes, 5 - C.P. 41004 123456789 sagen
8 Cuenta de ejemplo para asignaciones autoproc
9 Empresa Calle Reyes Católicos, 12 - C.P. 18009 sagen
10 Individual Calle Claudio Marcelo, 13 - C.P. 14002 9549588888 sagen
11 Individual Calle Granada, 45 - C.P. 18009 (610) 265-9100 sara_cm
12 Comunidad Calle Nueva, 10 - C.P. 21001 000000000 sagen
13 Empresa Calle Real, 18 - C.P. 4003 999999999 sagen
14 Comunidad Paseo del Parque, 15 - C.P. 29016 (620) 241-6200 sagen
15 Individual Calle Maestra, 24 - C.P. 23004 (251) 679-2200 sagen

- Pasar candidato a apto y se crea cuenta y contacto







- Se crea tarea (correo)

Candidato pendiente de cualificación

SM **Sara Cabeza Muñoz <sarcabmun@alum.us.es>**
10:11

Para: SARA CABEZA MUÑOZ

Tarea

Para: Sara Analista

Sara Cabeza Muñoz le ha asignado la siguiente tarea:

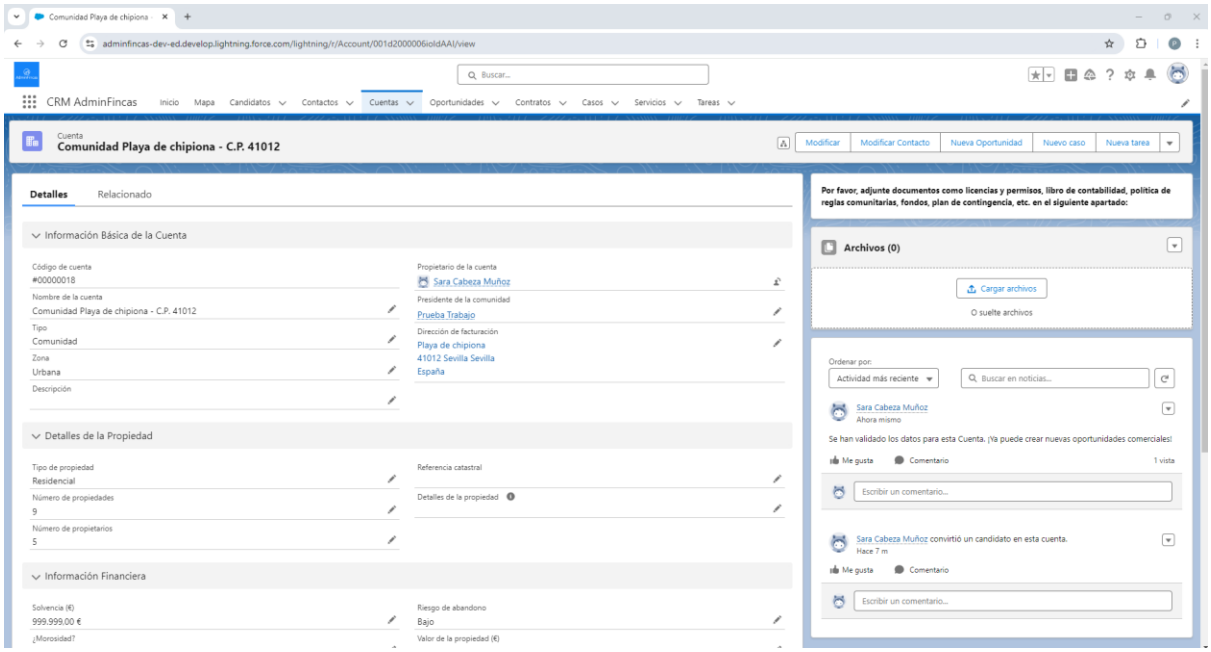
Asunto: Candidato pendiente de cualificación
Fecha de vencimiento: 14/9/2024
Prioridad: Normal

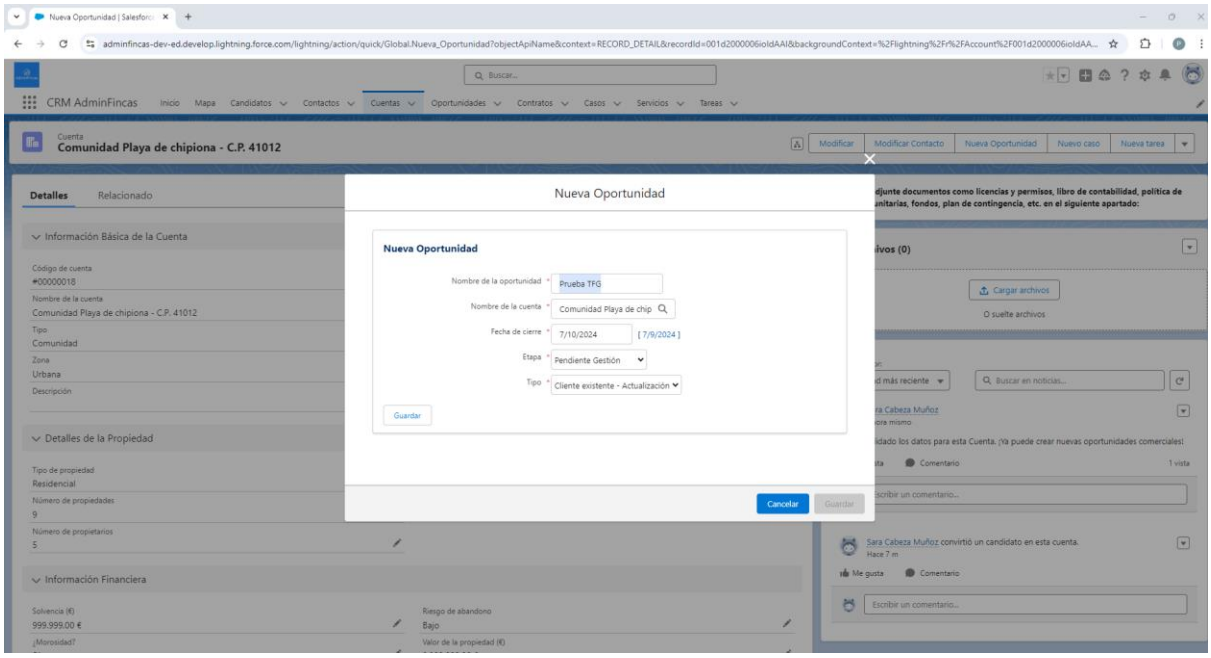
Si desea más información, haga clic en el siguiente vínculo:

https://adminfincas-dev-ed.develop.my.salesforce.com/00Td2000000bPKP

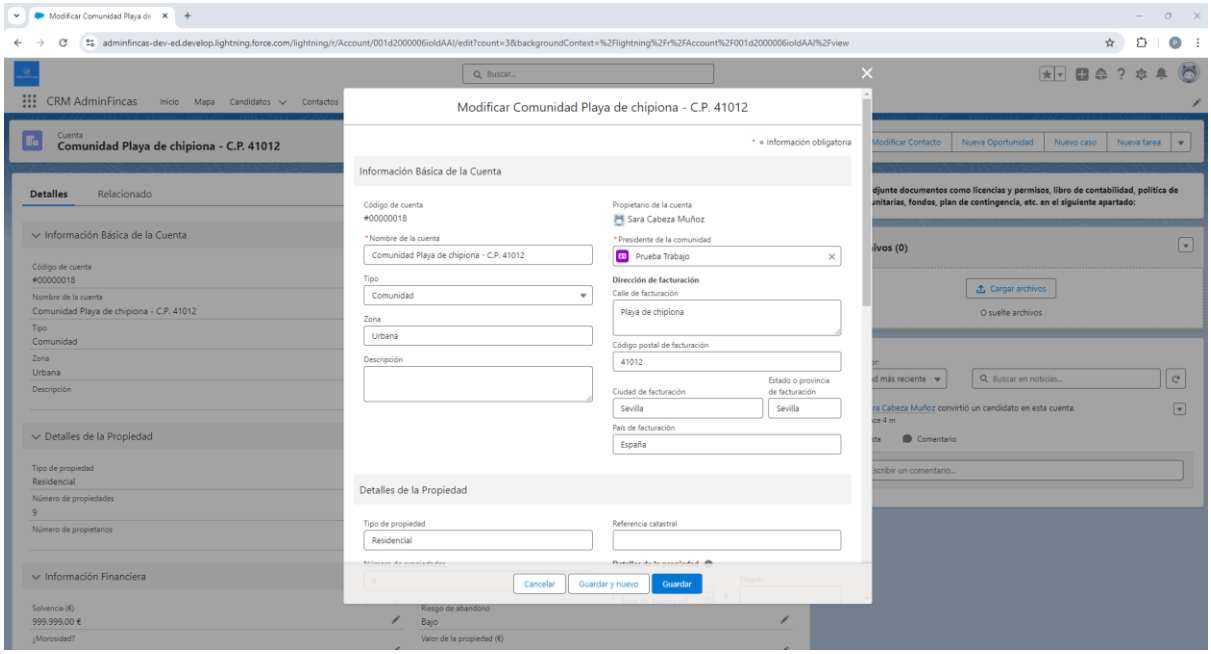
2. CUENTA

- Ref catastral obligatoria

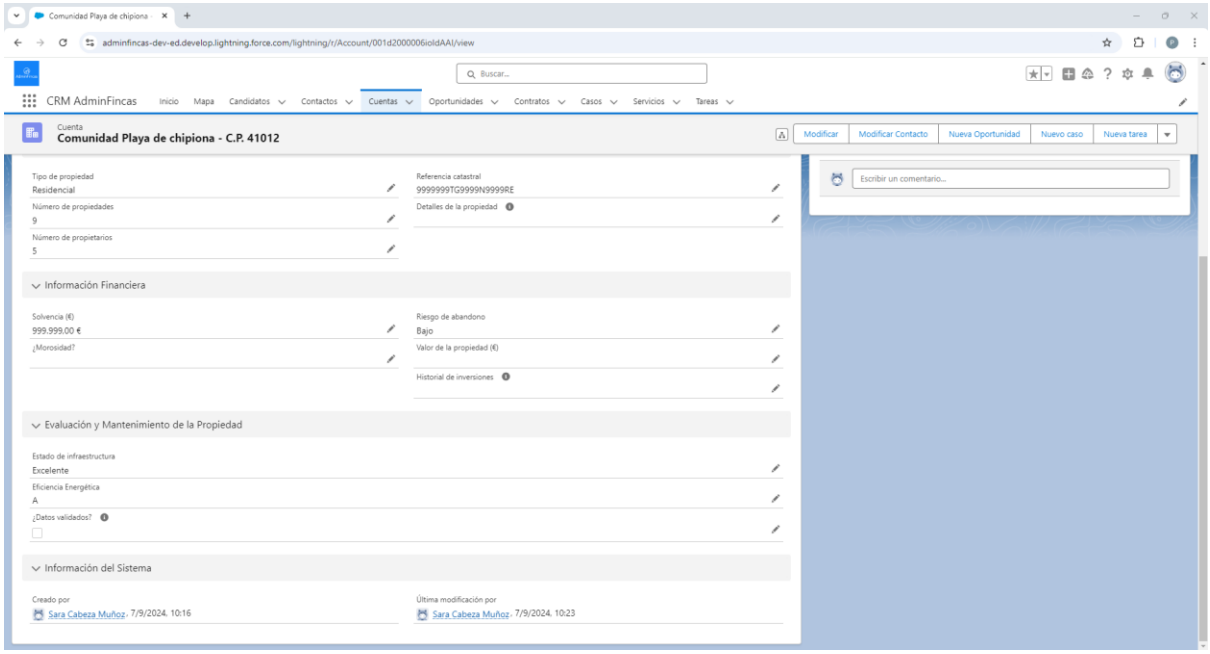


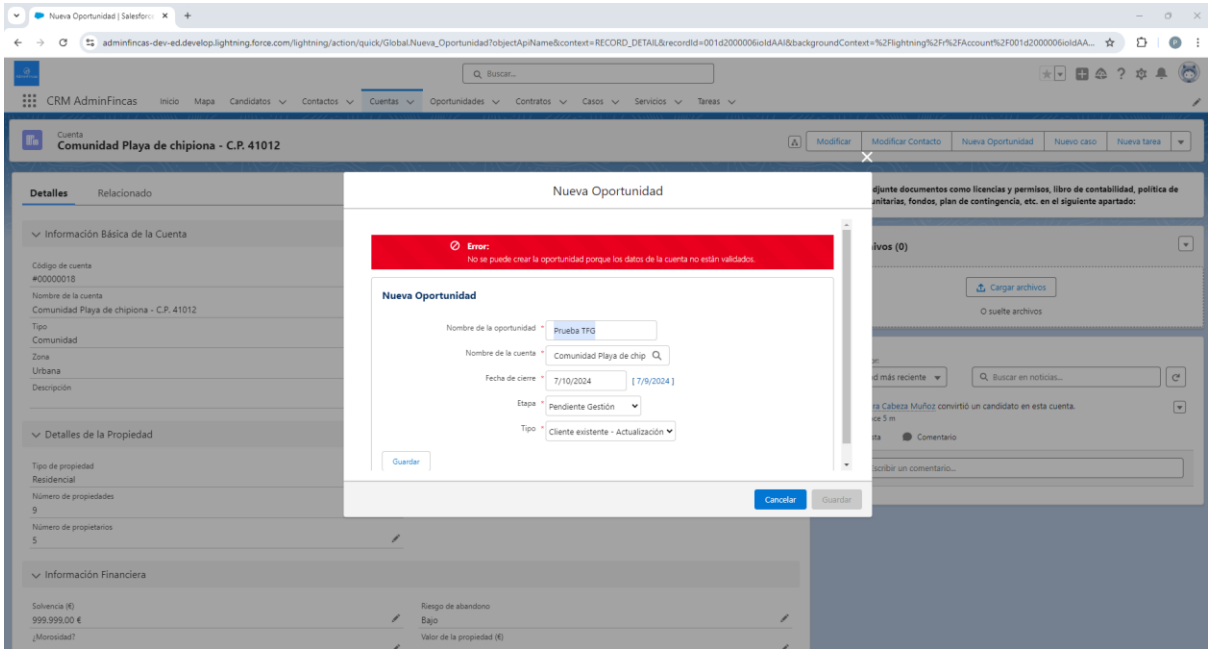


- Modificar cuenta

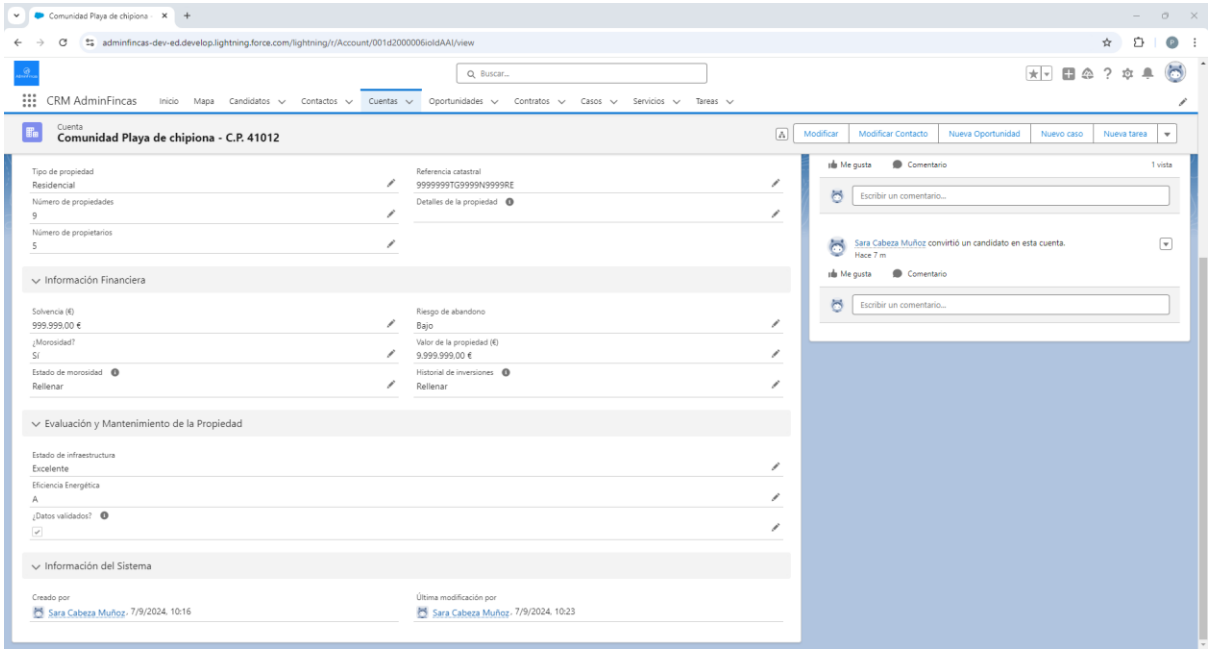


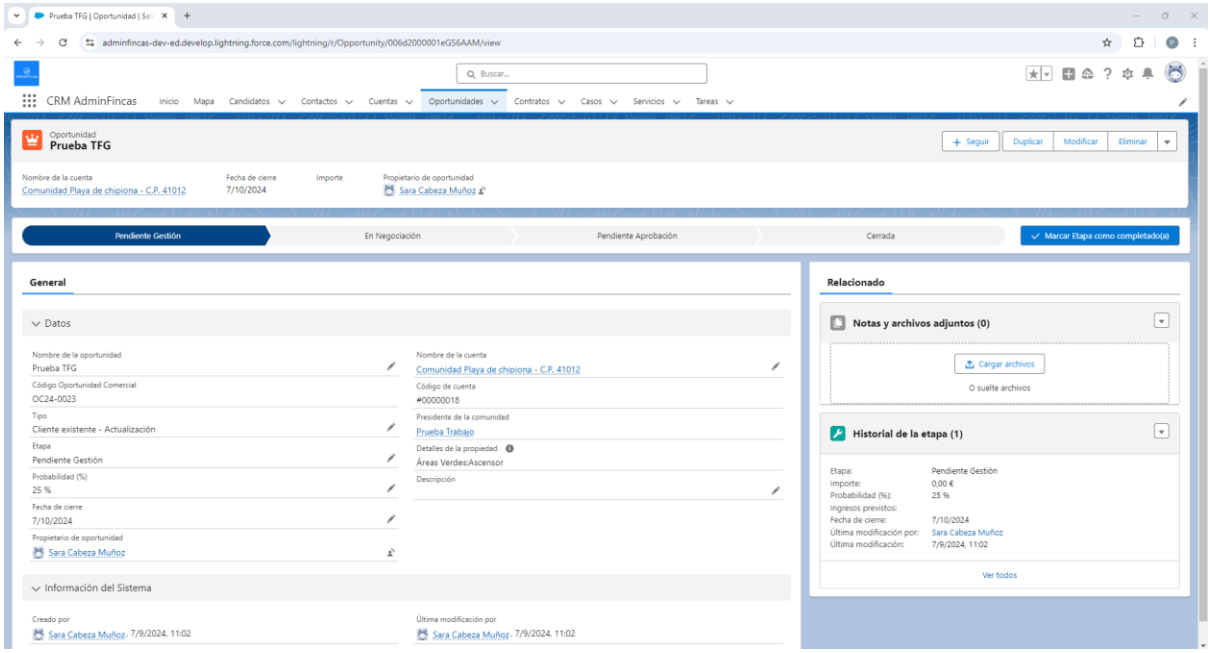
- Crear nueva oportunidad sin validación de datos



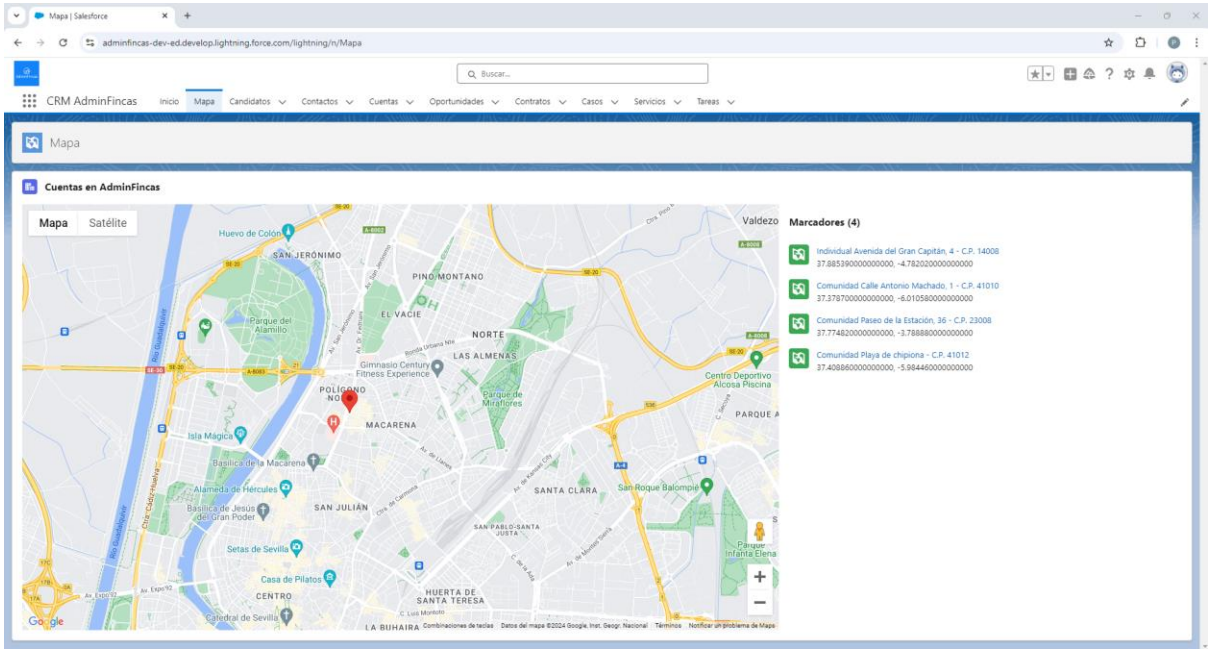


- Crear nueva oportunidad con validación de datos



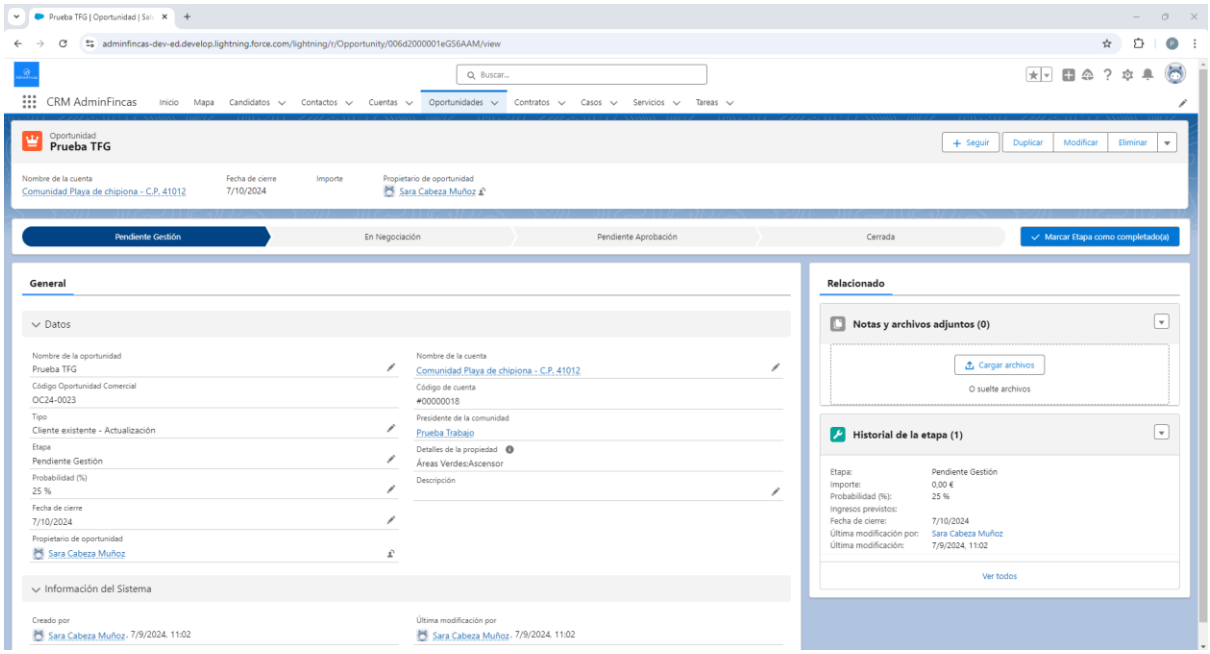


- Localización en mapa

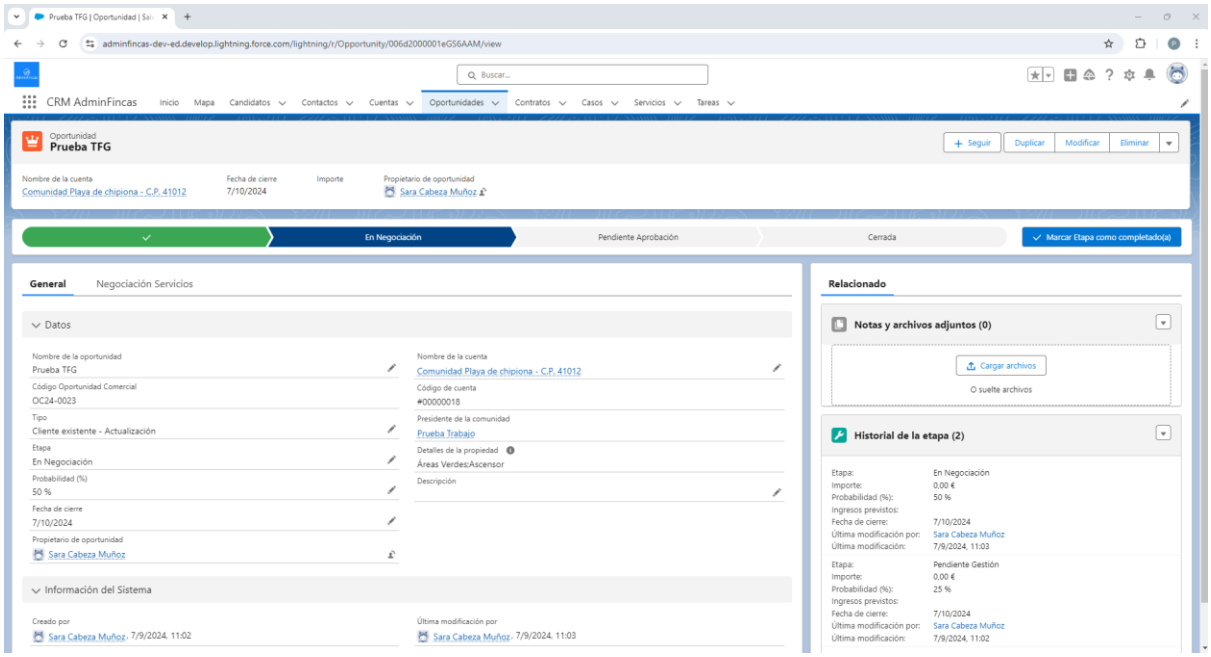


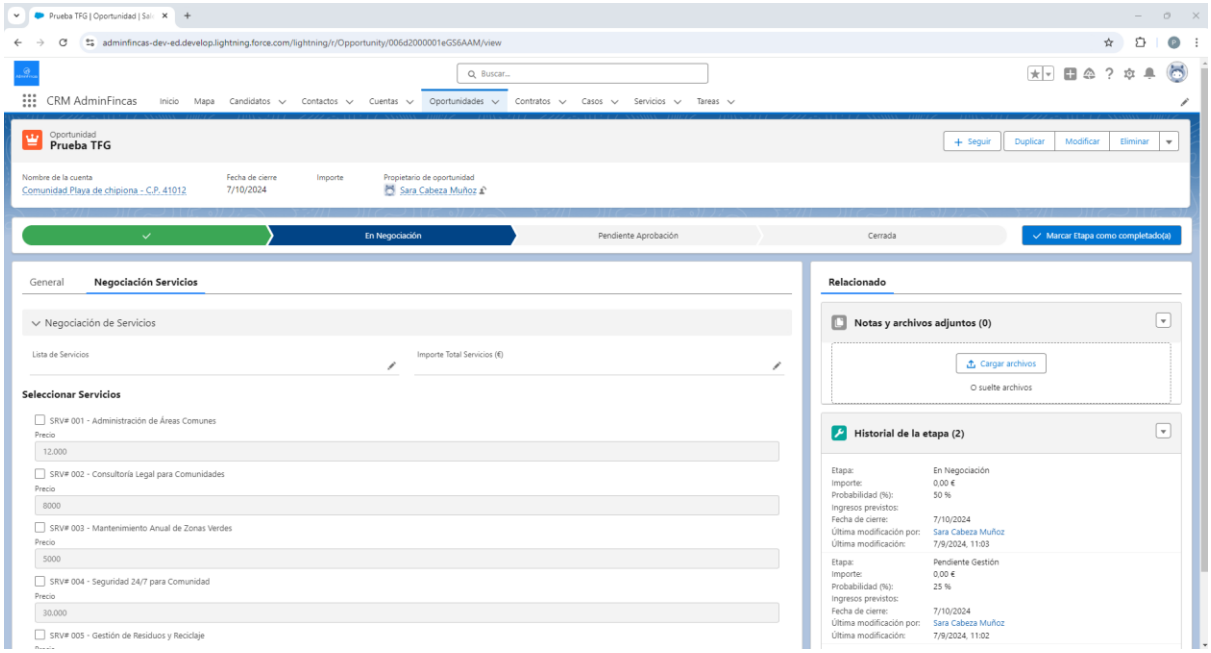
3. OPORTUNIDAD

- Crear oportunidad en pendiente gestión

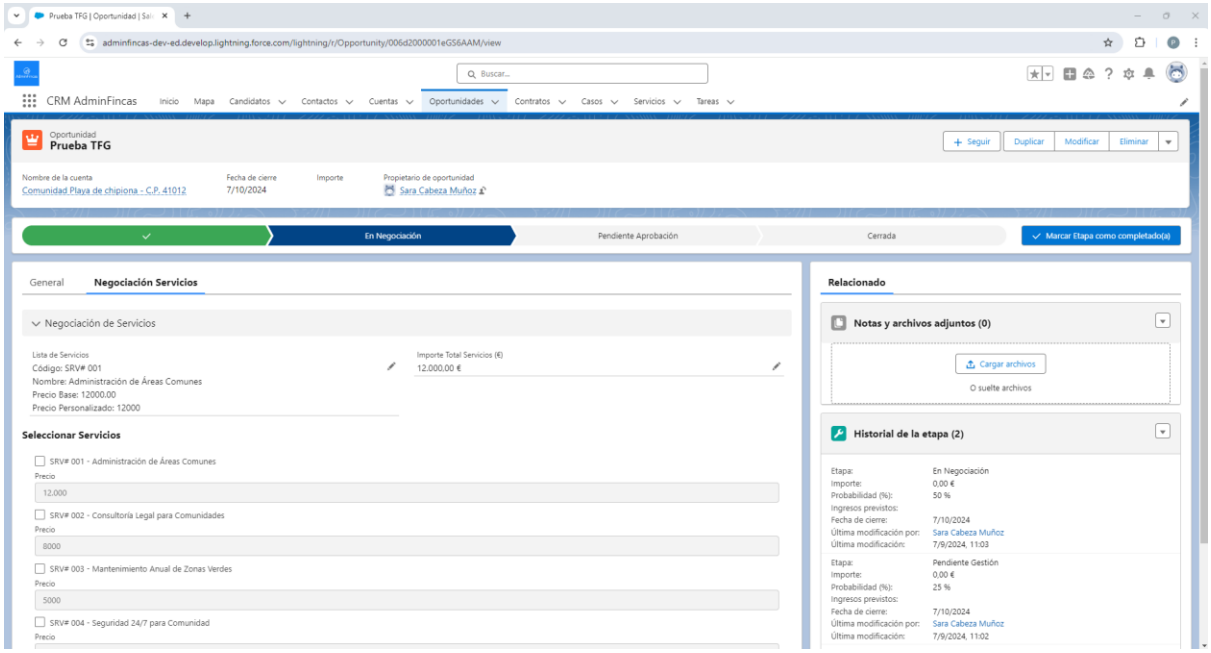


- Fase de negociación

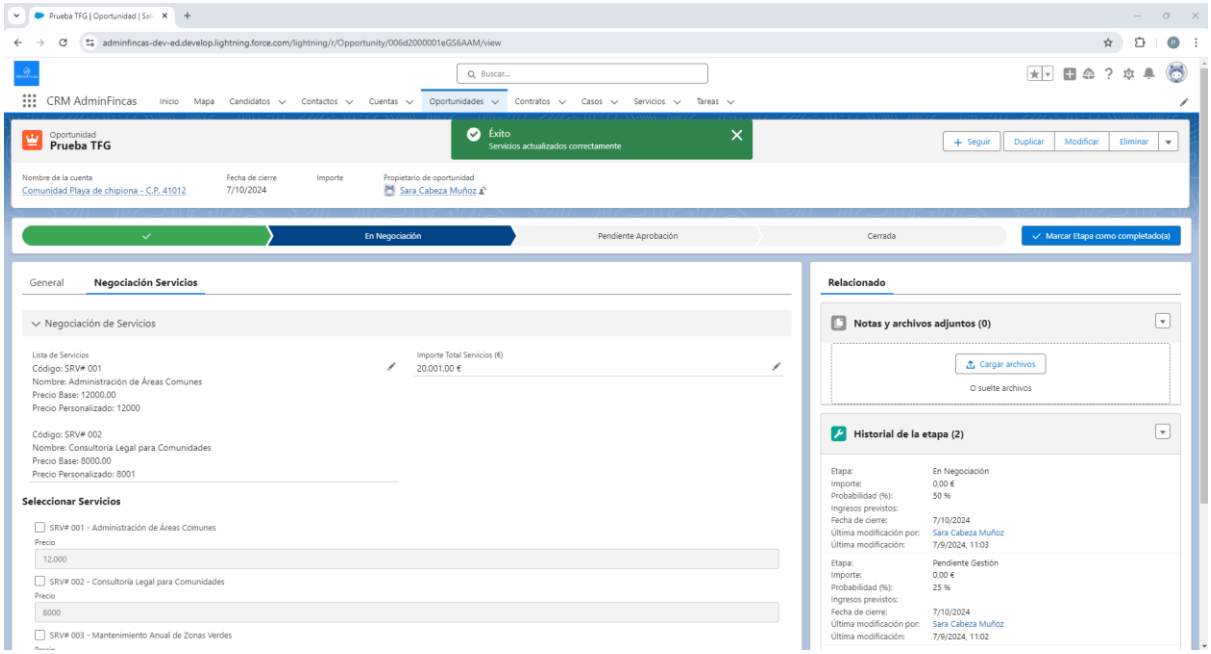




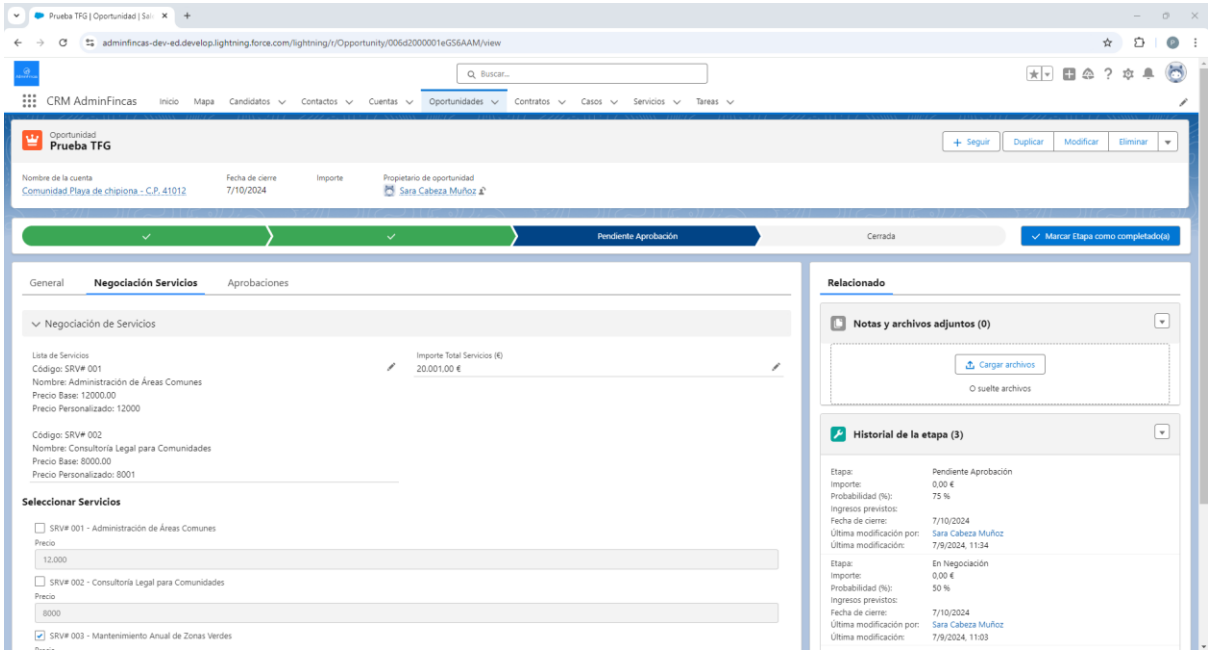
- Incluir un servicio



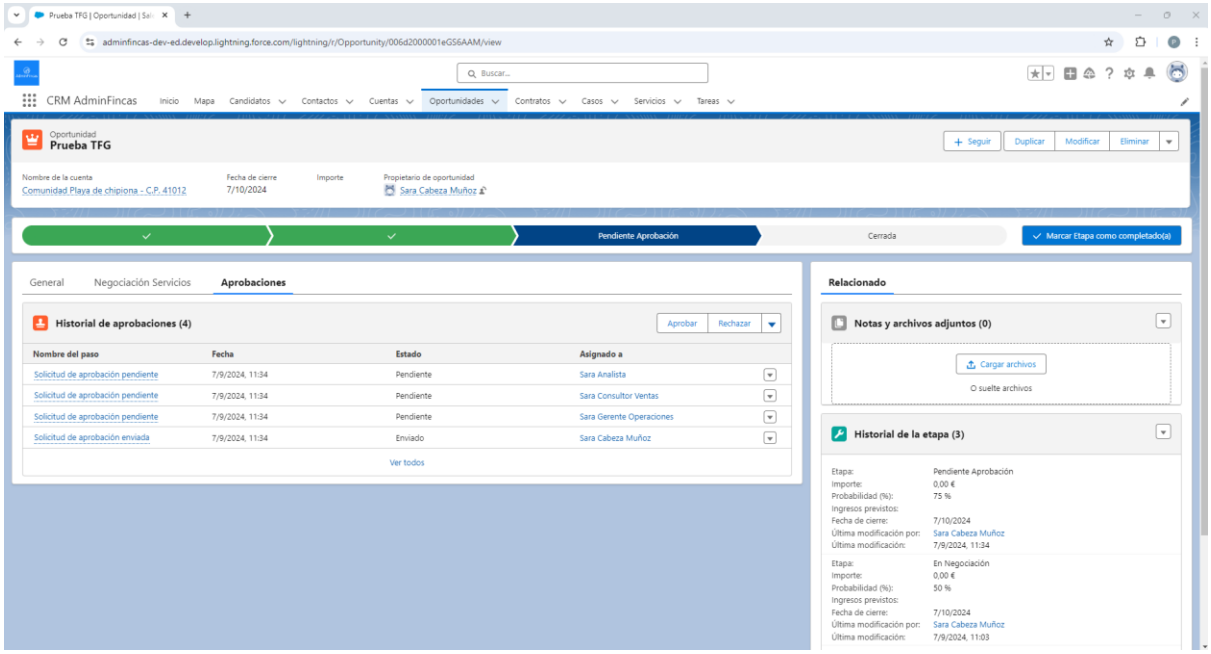
- Modificar importe de un servicio e incluir otro

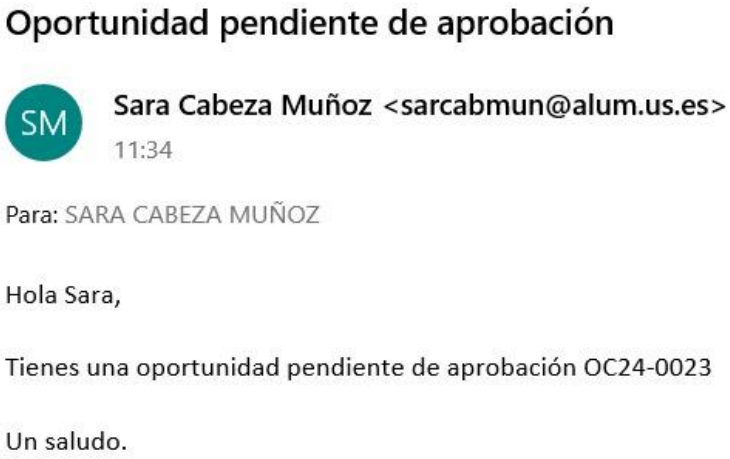


- Fase pendiente aprobación

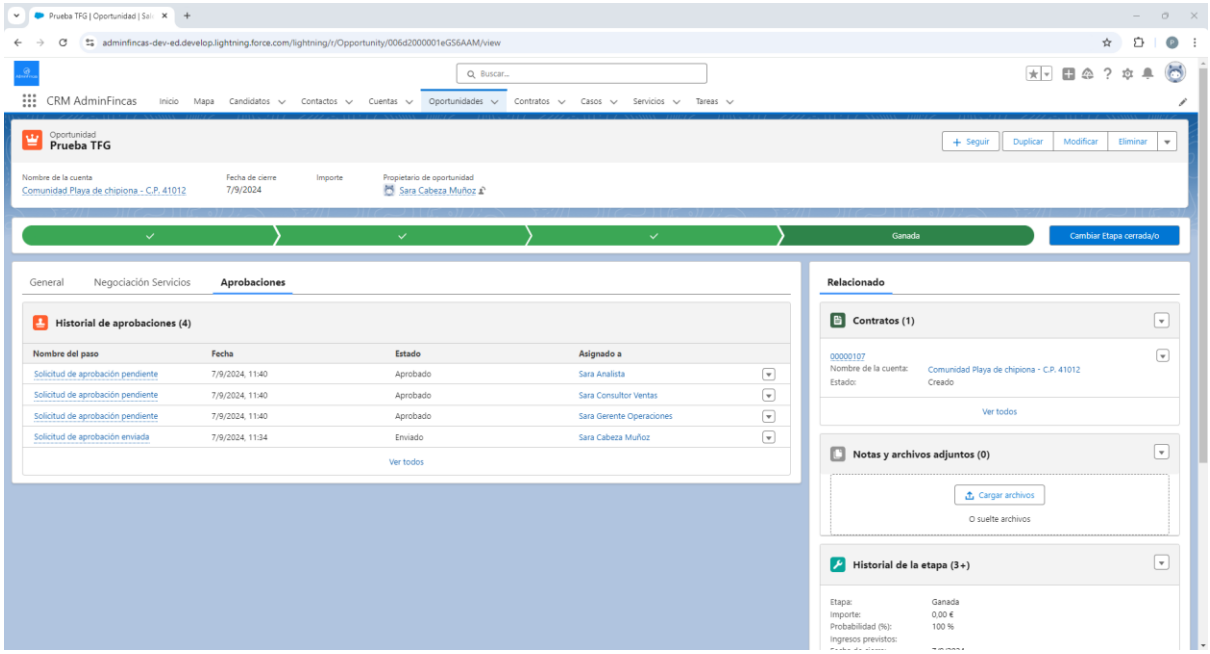


- Aprobaciones





- Aprobación de oportunidades



Nuevo contrato pendiente de firma

SM **Sara Cabeza Muñoz <sarcabmun@alum.us.es>**
11:41

Para: SARA CABEZA MUÑOZ

Nueva tarea

Para: Sara Gestor Activos

Sara Cabeza Muñoz le ha asignado la siguiente nueva tarea:

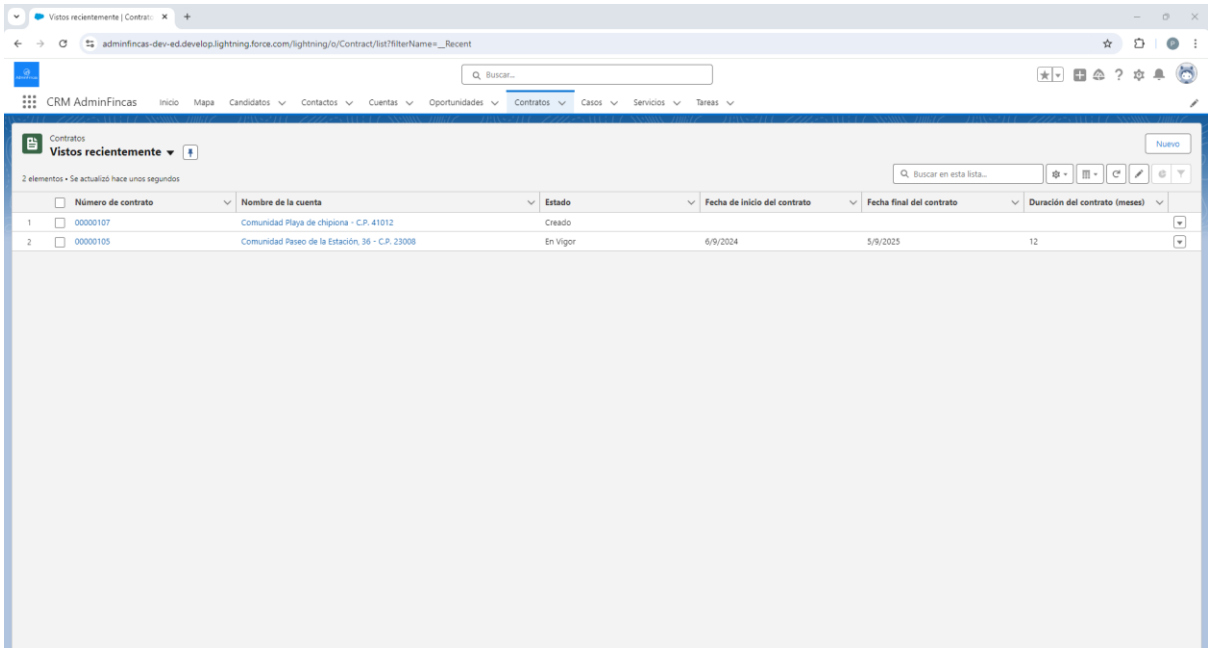
Asunto: Nuevo contrato pendiente de firma
Fecha de vencimiento: 12/9/2024
Prioridad: Normal

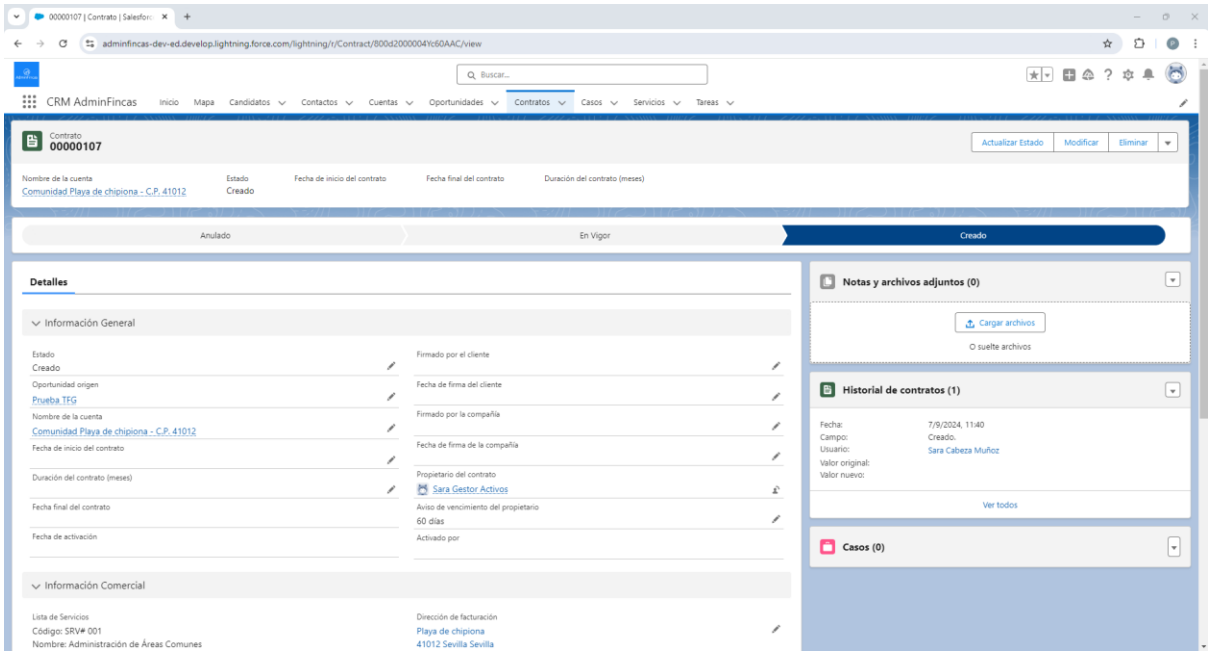
Si desea más información, haga clic en el siguiente vínculo:

https://adminfincas-dev-ed.develop.my.salesforce.com/00Td2000000bPp3

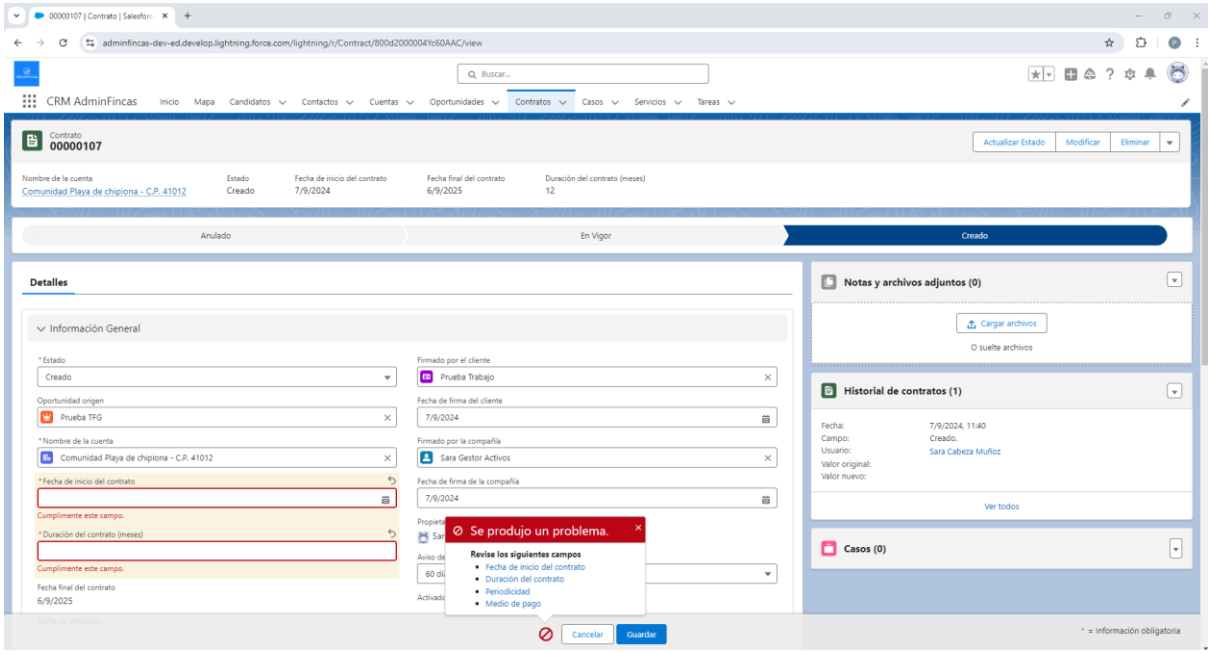
4. CONTRATO

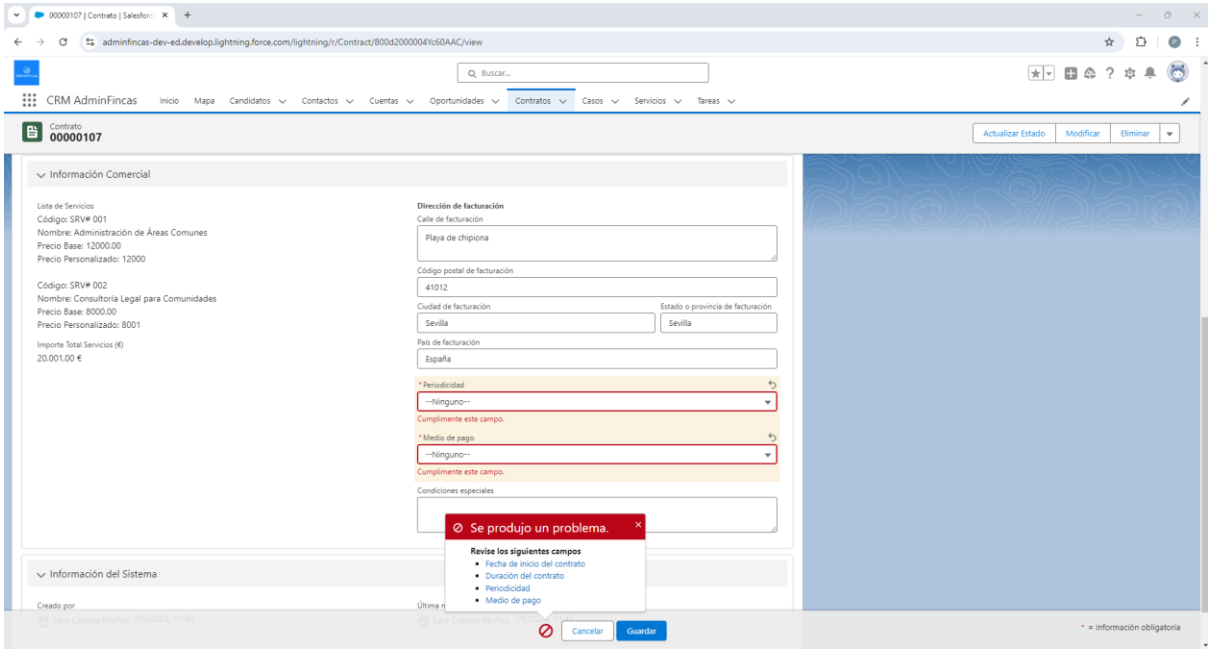
- Generación contrato

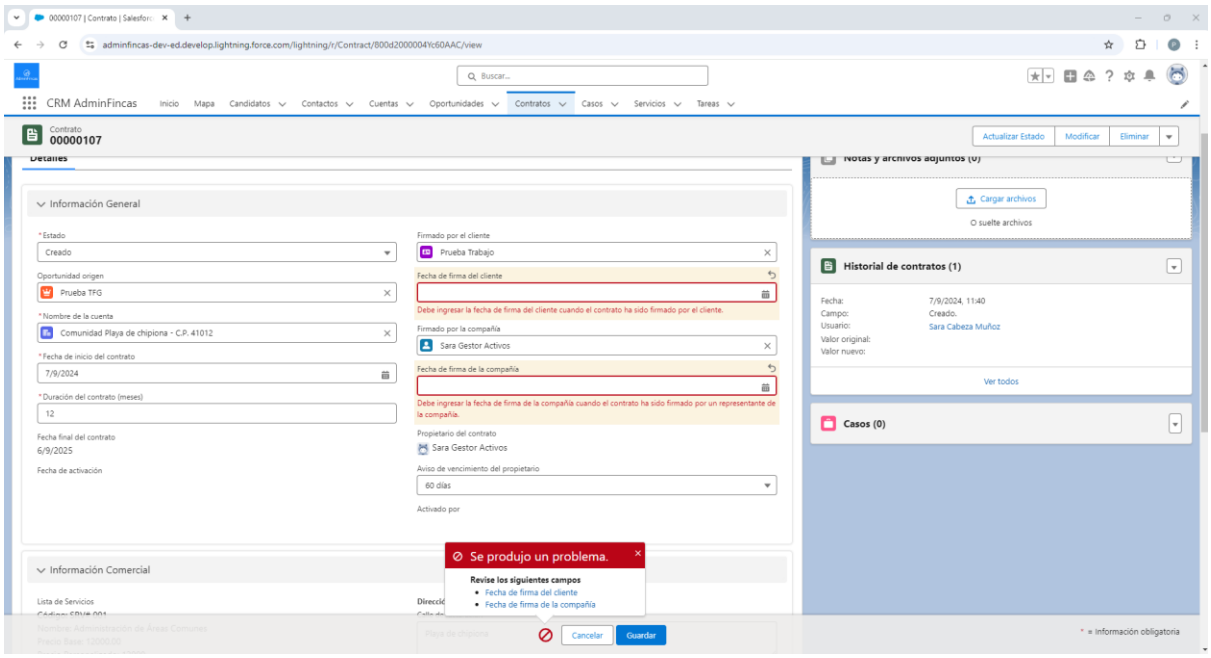


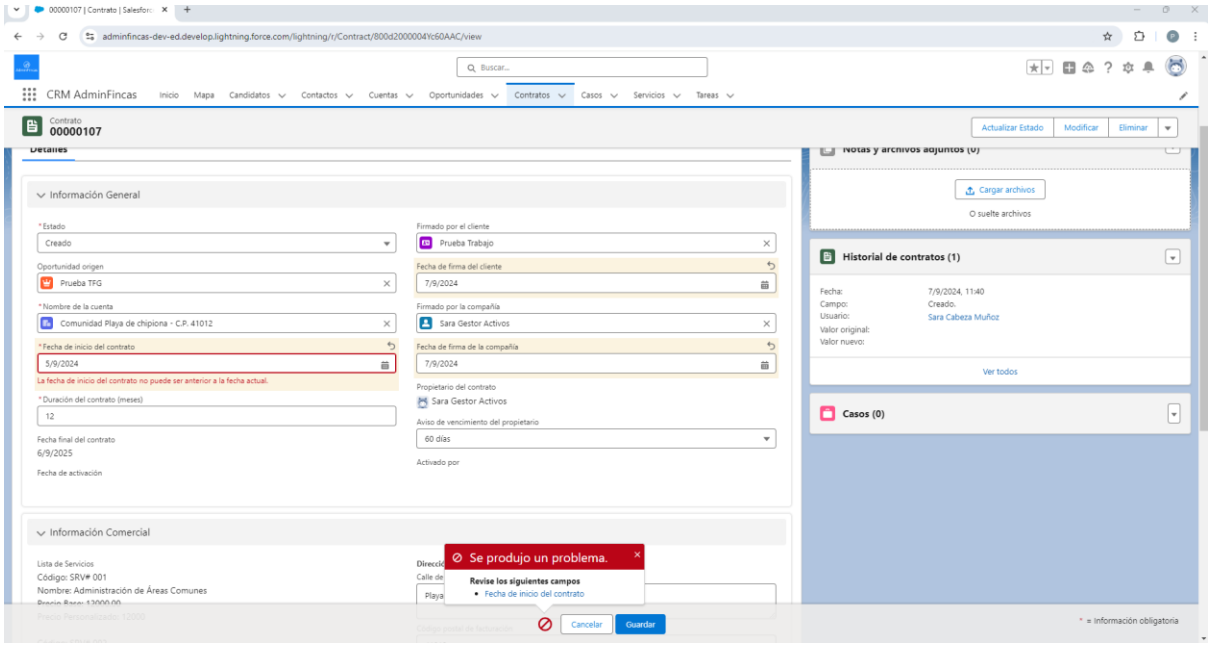


- Datos obligatorios

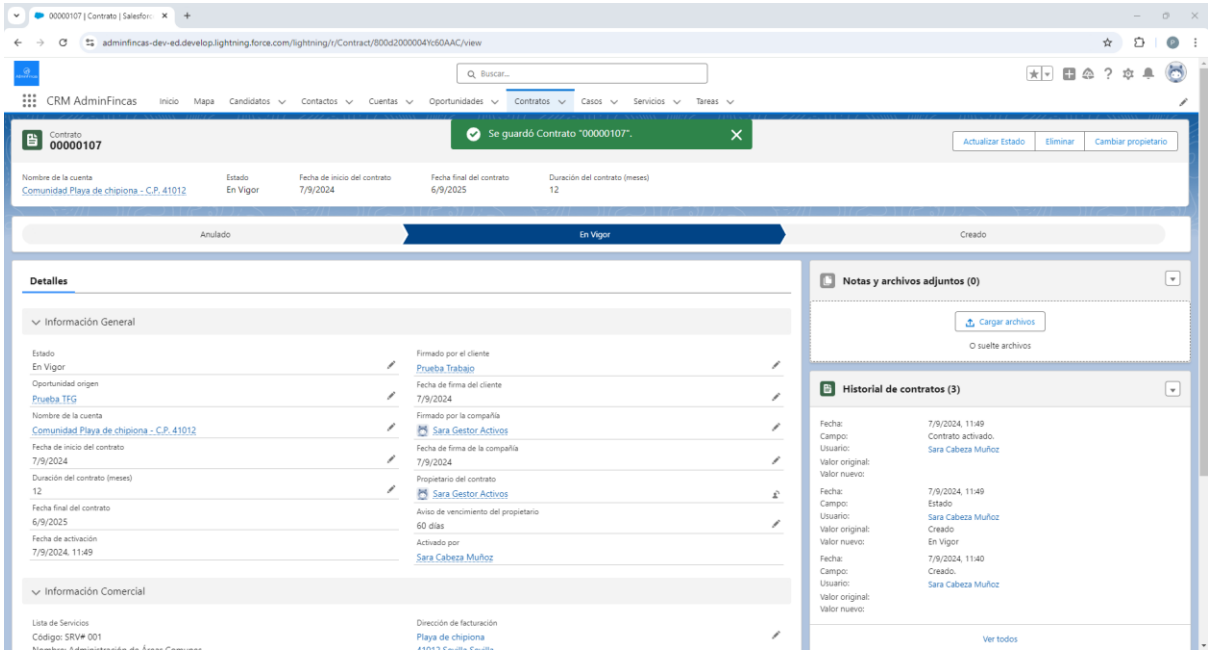




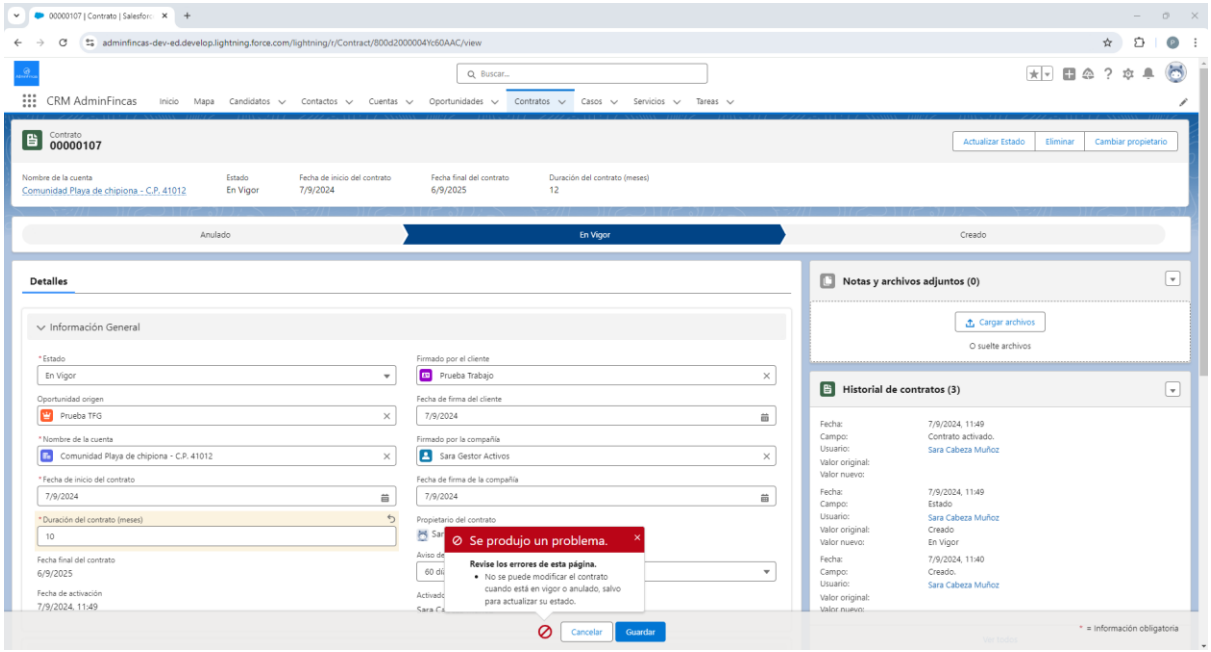

CRM AdminFincas
Contrato
00000107
Actualizar Estado
Modificar
Eliminar
Información General
Estado
Creado
Oportunidad origen
Prueba TFG
Nombre de la cuenta
Comunidad Playa de chipiona - C.P. 41012
Fecha de inicio del contrato
7/9/2024
Duración del contrato (meses)
12
Fecha final del contrato
6/9/2025
Fecha de activación
Firmado por el cliente
Prueba Trabajo
Fecha de firma del cliente
Debe ingresar la fecha de firma del cliente cuando el contrato ha sido firmado por el cliente.
Firmado por la compañía
Sara Gestor Activos
Fecha de firma de la compañía
Debe ingresar la fecha de firma de la compañía cuando el contrato ha sido firmado por un representante de la compañía.
Propietario del contrato
Sara Gestor Activos
Aviso de vencimiento del propietario
60 días
Activado por
Se produjo un problema.
Revise los siguientes campos
Fecha de firma del cliente
Fecha de firma de la compañía
Cancelar
Guardar
Información Comercial
Cargar archivos
O suelte archivos
Historial de contratos (1)
Fecha: 7/9/2024, 11:40
Campo: Creado.
Usuario: Sara Cabeza Muñoz
Valor original:
Valor nuevo:
Ver todos
Casos (0)
* = Información obligatoria


CRM AdminFincas
Contrato
00000107
Actualizar Estado
Modificar
Eliminar
Información General
Estado
Creado
Oportunidad origen
Prueba TFG
Nombre de la cuenta
Comunidad Playa de chipiona - C.P. 41012
Fecha de inicio del contrato
5/9/2024
La fecha de inicio del contrato no puede ser anterior a la fecha actual.
Duración del contrato (meses)
12
Fecha final del contrato
6/9/2025
Fecha de activación
Firmado por el cliente
Prueba Trabajo
Fecha de firma del cliente
7/9/2024
Firmado por la compañía
Sara Gestor Activos
Fecha de firma de la compañía
7/9/2024
Propietario del contrato
Sara Gestor Activos
Aviso de vencimiento del propietario
60 días
Activado por
Se produjo un problema.
Revise los siguientes campos
Fecha de inicio del contrato
Cancelar
Guardar
Información Comercial
Lista de Servicios
Código: SRV# 001
Nombre: Administración de Áreas Comunes
Cargar archivos
O suelte archivos
Historial de contratos (1)
Fecha: 7/9/2024, 11:40
Campo: Creado.
Usuario: Sara Cabeza Muñoz
Valor original:
Valor nuevo:
Ver todos
Casos (0)
* = Información obligatoria

- Contrato en vigor

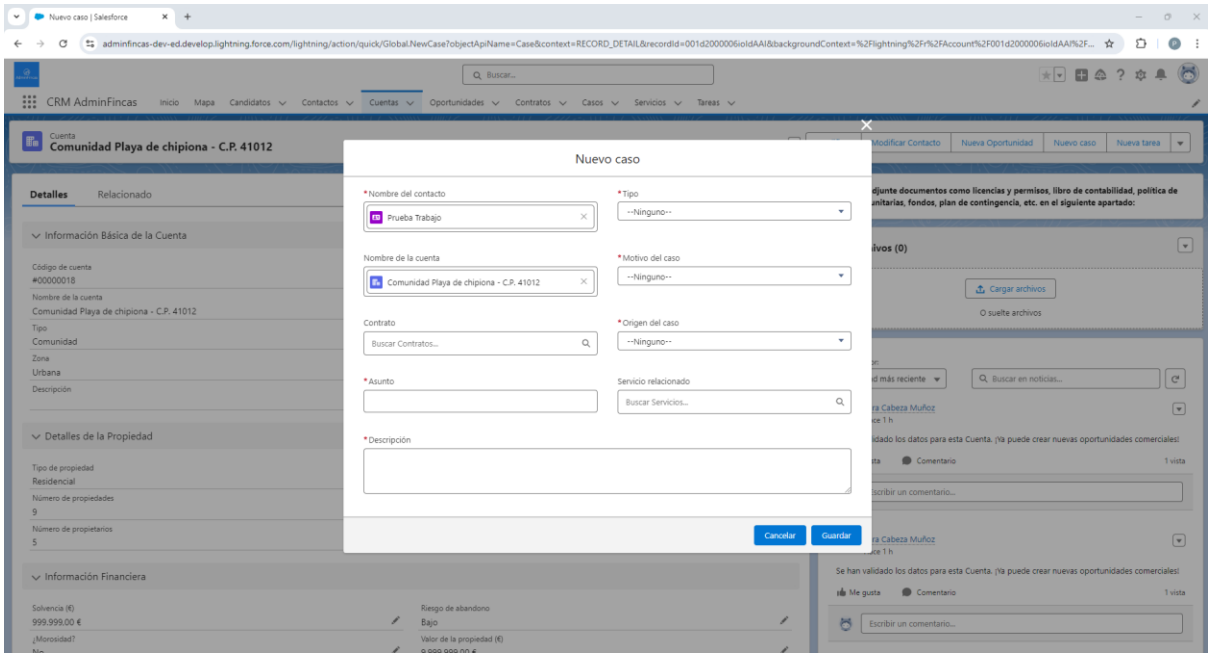


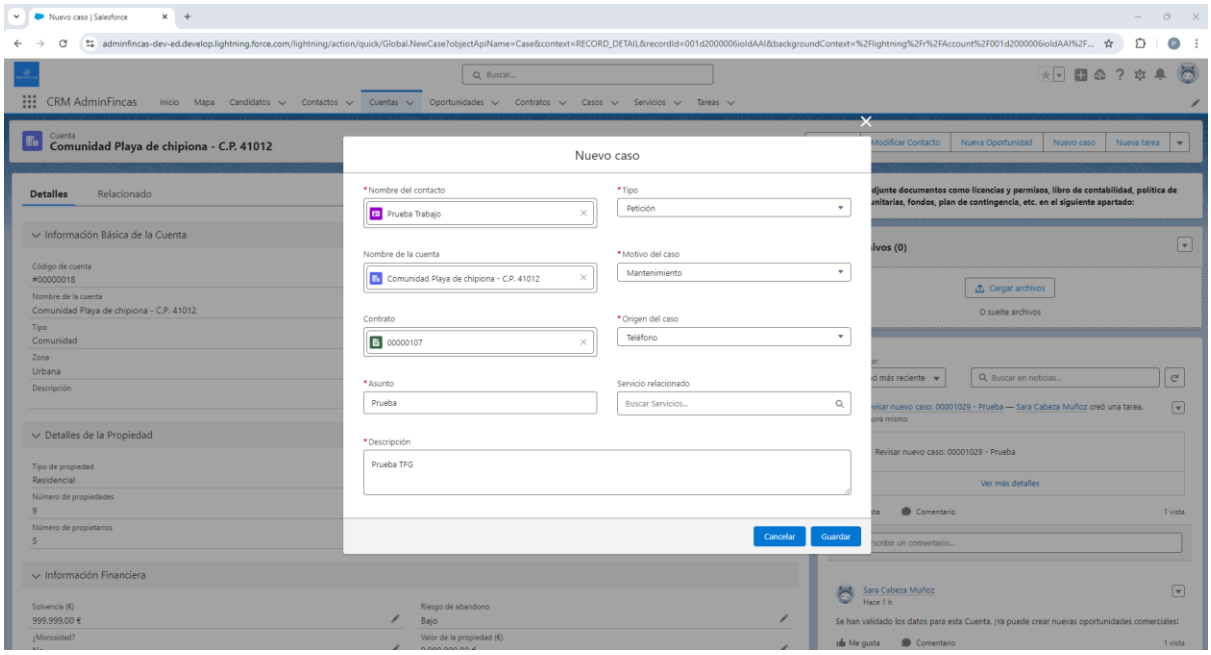
- No deja modificar datos

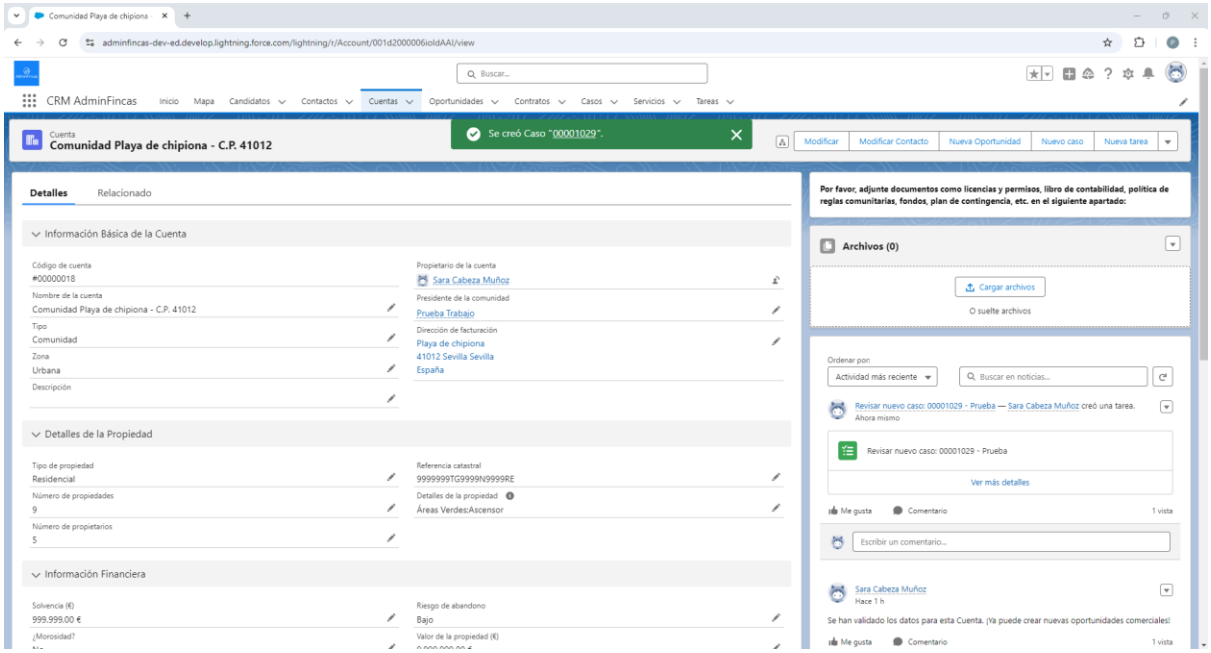


5. CASOS

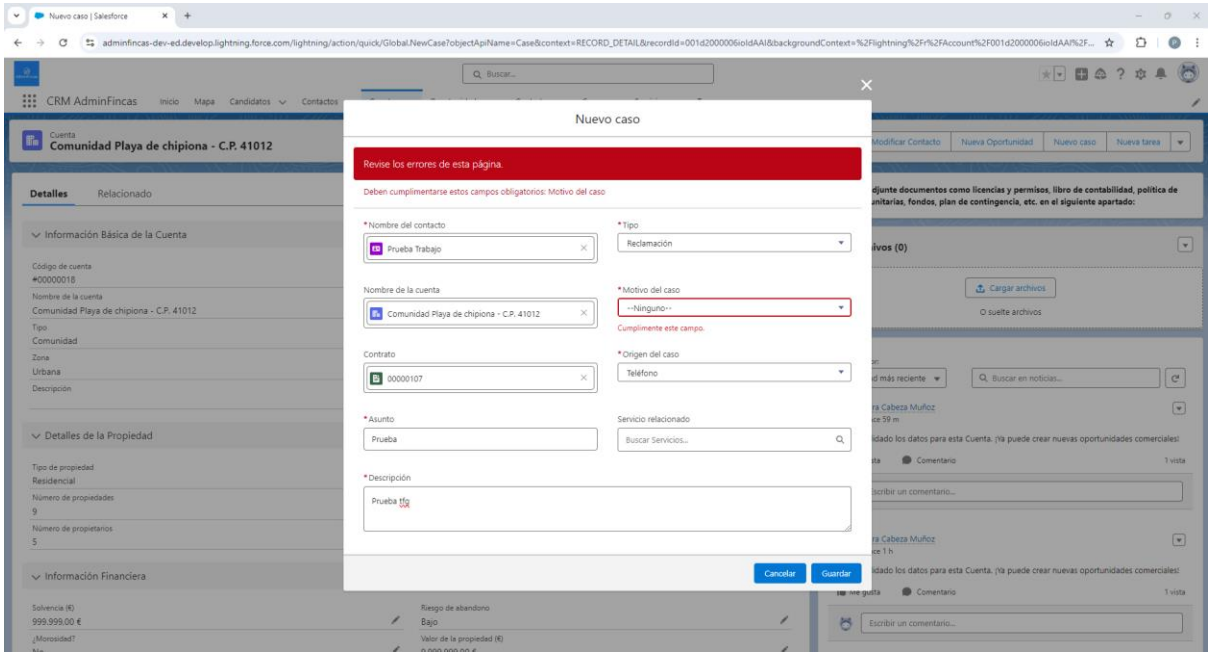
- Creación de caso



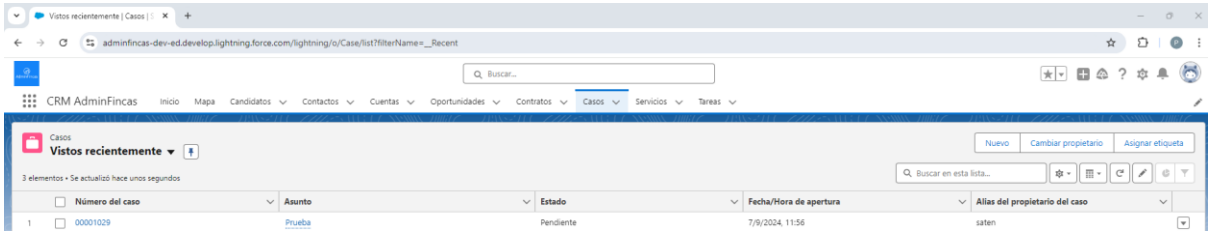


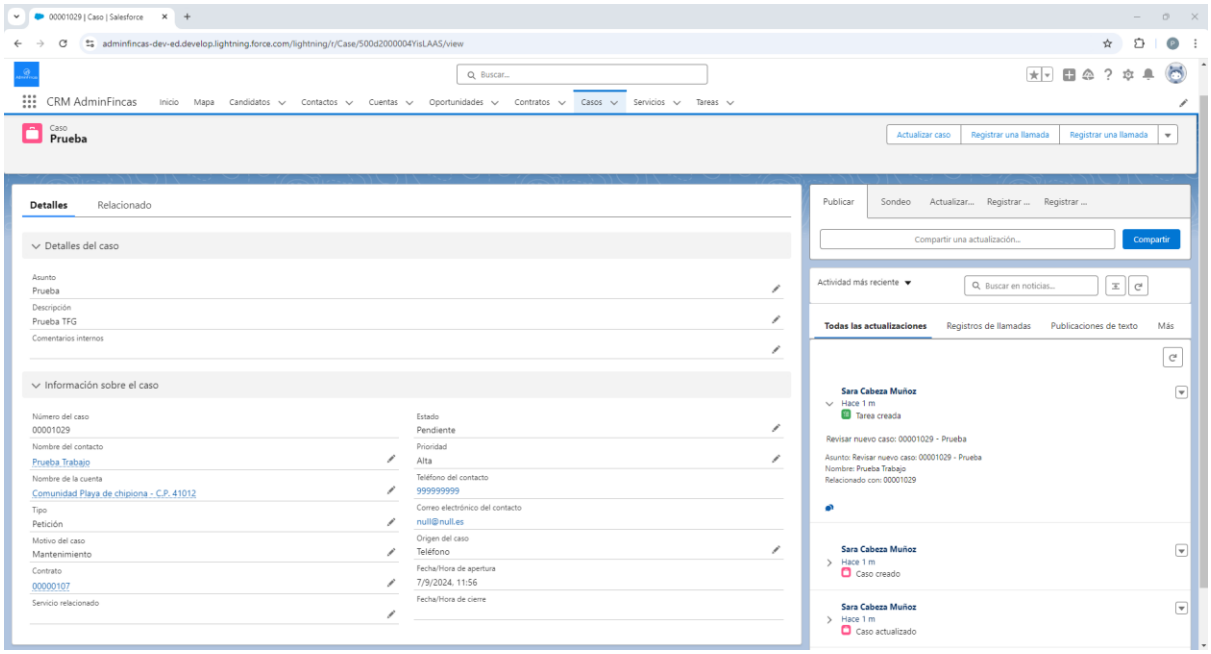


- Datos obligatorios

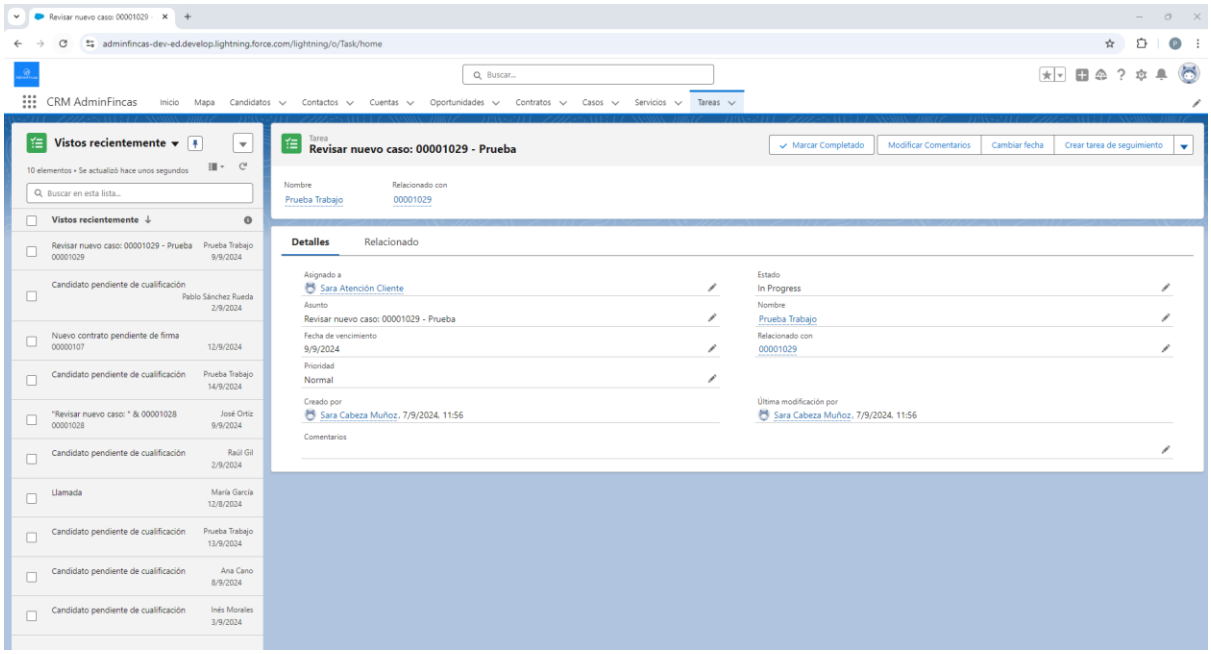


- Prioridad del caso alta y estado pendiente





- Asignado a Atención al cliente y creación de tarea



Revisar nuevo caso: 00001029 - Prueba

SM **Sara Cabeza Muñoz <sarcabmun@alum.us.es>**
11:57

Para: SARA CABEZA MUÑOZ

Nueva tarea

Para: Sara Atención Cliente

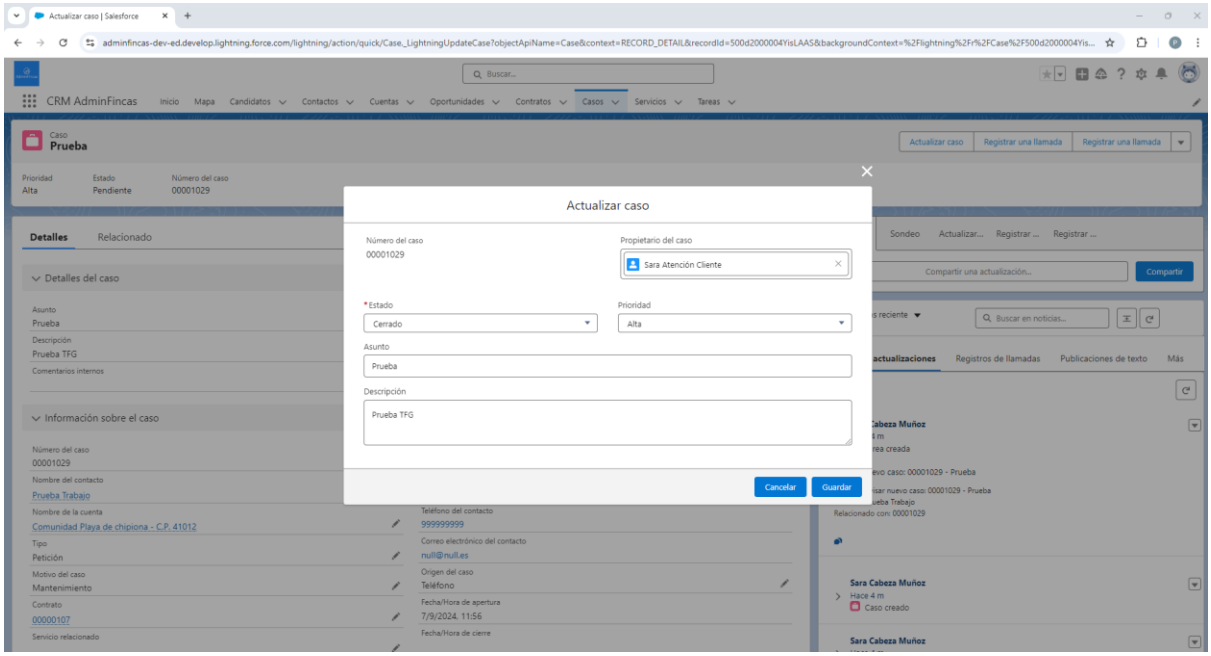
Sara Cabeza Muñoz le ha asignado la siguiente nueva tarea:

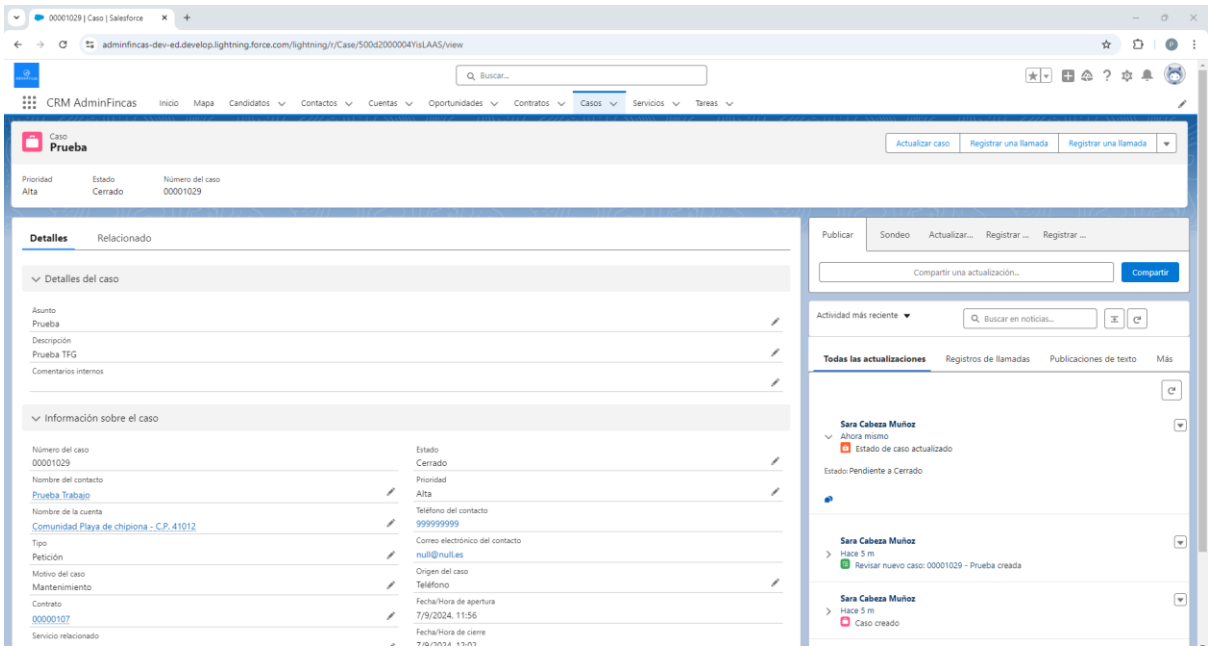
Asunto: Revisar nuevo caso: 00001029 - Prueba
Fecha de vencimiento: 9/9/2024
Prioridad: Normal

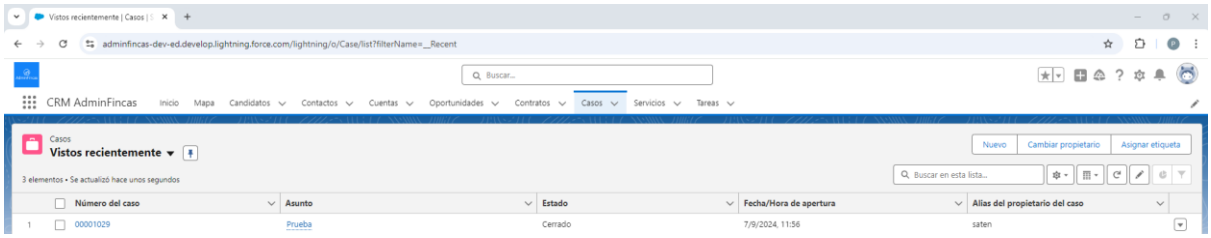
Si desea más información, haga clic en el siguiente vínculo:

https://adminfincas-dev-ed.develop.my.salesforce.com/00Td2000000bPtt

- Actualizar caso a Cerrado




00001029 | Caso | Salesforce
adminfincas-dev-ed.develop.lightning.force.com/lightning/r/Case/500d2000004YisLAAS/view
CRM AdminFincas
Inicio
Mapa
Candidatos
Contactos
Cuentas
Oportunidades
Contratos
Casos
Servicios
Tareas
Caso
Prueba
Actualizar caso
Registrar una llamada
Registrar una llamada
Prioridad
Alta
Estado
Cerrado
Número del caso
00001029
Detalles
Relacionado
Detalles del caso
Asunto
Prueba
Descripción
Prueba TFG
Comentarios internos
Información sobre el caso
Número del caso
00001029
Nombre del contacto
Prueba Trabajo
Nombre de la cuenta
Comunidad Playa de chipiona - C.P. 41012
Tipo
Petición
Motivo del caso
Mantenimiento
Contrato
00000107
Servicio relacionado
Estado
Cerrado
Prioridad
Alta
Teléfono del contacto
999999999
Correo electrónico del contacto
null@null.es
Origen del caso
Teléfono
Fecha/Hora de apertura
7/9/2024, 11:56
Fecha/Hora de cierre
Publicar
Sondeo
Actualizar...
Registrar ...
Registrar ...
Compartir una actualización...
Compartir
Actividad más reciente
Buscar en noticias...
Todas las actualizaciones
Registros de llamadas
Publicaciones de texto
Más
Sara Cabeza Muñoz
Ahora mismo
Estado de caso actualizado
Estado: Pendiente a Cerrado
Sara Cabeza Muñoz
Hace 5 m
Revisar nuevo caso: 00001029 - Prueba creada
Sara Cabeza Muñoz
Hace 5 m
Caso creado


Vistos recientemente | Casos |
adminfincas-dev-ed.develop.lightning.force.com/lightning/o/Case/list?filterName=__Recent
CRM AdminFincas
Inicio
Mapa
Candidatos
Contactos
Cuentas
Oportunidades
Contratos
Casos
Servicios
Tareas
Casos
Vistos recientemente
3 elementos • Se actualizó hace unos segundos
Nuevo
Cambiar propietario
Asignar etiqueta
Buscar en esta lista...
Número del caso
Asunto
Estado
Fecha/Hora de apertura
Alias del propietario del caso
1
00001029
Prueba
Cerrado
7/9/2024, 11:56
saten