|  |
| --- |
|  |
|  |
|  |



ÍNDICE

[Introduccion 2](#_Toc214570351)

[Planteamiento del problema 3](#_Toc214570352)

[Panel Trello y Scrum Master 4](#_Toc214570353)

[Repositorio GitHub 6](#_Toc214570354)

[Requisitos Funcionales y No Funcionales 7](#_Toc214570355)

[Análisis y Diseño de la BBDD 10](#_Toc214570356)

[Diagrama de entidad relación 11](#_Toc214570357)

[Diagrama de clases 13](#_Toc214570358)

[Diagrama de casos de uso 15](#_Toc214570359)

[Arquitectura General del Sistema 16](#_Toc214570360)

## Introduccion

Este documento tiene como objetivo recopilar y presentar de manera organizada toda la documentación generada por el equipo durante la primera fase del proyecto. Su finalidad es mostrar el estado actual del desarrollo de los análisis, diseños y herramientas de gestión utilizados hasta el momento.

En este documento se incluyen: los requisitos funcionales y no funcionales de ambas aplicaciones, el análisis y diseño de la base de datos, los diagramas de entidad-relación, de clases y de casos de uso, así como la arquitectura general del sistema y los endpoints a implementar. Además, se presentan el panel de Trello y el repositorio de GitHub, con capturas de pantalla y explicaciones sobre la organización y distribución de las tareas dentro del equipo.

Este documento sirve tanto como memoria de lo realizado hasta ahora como guía para la planificación de las siguientes fases del proyecto, garantizando una visión clara y estructurada del trabajo realizado y facilitando la evaluación de los avances alcanzados.

## Planteamiento del problema

Nuestro proyecto se centra en el desarrollo de un sistema de compraventa de productos de segunda mano que incluye dos aplicaciones principales: el ERP para empleados (basado en Odoo con PostgreSQL) y una app móvil desarrollada en Kotlin para usuarios, nos estamos replanteando emplear Ionic 8 con Angular lo cual nos permitiría el desarrollo multiplataforma y empleando Capacitor poder hacerla nativa para Android y con ello que sea una app móvil optima. El objetivo es facilitar y moderar la compraventa, centralizando la gestión mediante una arquitectura robusta y fácilmente operable por todos los roles implicados.

## Panel Trello y Scrum Master

Para organizar el trabajo, hemos utilizado Trello, donde hemos estructurado el tablero en listas como Product Backlog, Sprint Backlog, To Do, In Progress y Done. En cada tarjeta hemos descrito las tareas y subtareas, asignando responsables y fijando fechas límite según la temporalización establecida.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

También podremos observar las tareas de un Sprint Backlogque han sido segmentadas a raíz de nuestro Product Backlog, para con ello poder repartírnoslas entre ambos compañeros y así repartir la carga de trabajo, al terminarlas se archivan y tachamos la checklist dentro de los ítems del Product Backlog declarando que esa funcionalidad esta cerrada.

Interfaz de usuario gráfica, Texto, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

Ademas nos hemos repartido las tareas por roles: uno como Product Owner y otro como Scrum Master, asegurando la revisión diaria del avance y la actualización del tablero tras cada reunión, a la vez por lo reducido que es nuestro equipo ambos también ejercemos el rol de Desarrolladores.

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

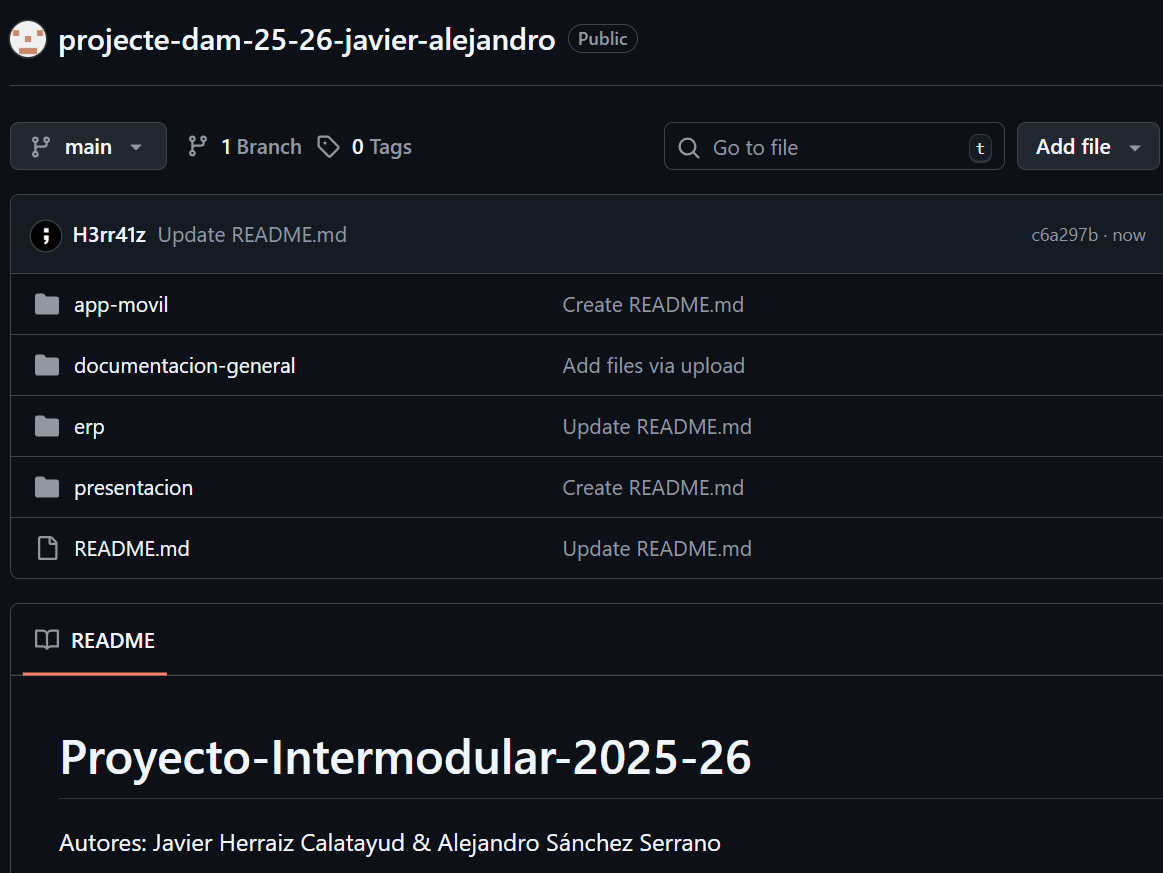
## Repositorio GitHub

El código y la documentación se almacenan en el repositorio GitHub, estructurado en carpetas para la app móvil, el entorno dockerizado de Odoo y la base de datos, además de una carpeta documental para UML y análisis.

Nuestro repositorio principal está alojado en GitHub bajo la dirección: [Repositorio GitHub Proyecto](https://github.com/Alejandro-WOU/projecte-dam-25-26-javier-alejandro.git)

En él se encuentra toda la documentación y el código fuente hasta la fecha. La organización se ha realizado por carpetas diferenciadas para cada módulo (ERP, móvil, documentación) y mantenemos un README actualizado con explicación de la estructura y los pasos de despliegue. Los colaboradores principales son Javier (H3rr41z) Herraiz Calatayud y Alejandro (Alejandro-WOU) Sánchez Serrano, y vamos actualizando el contenido mediante ramas y commits que reflejan la evolución y división de tareas

* **erp/:** Abarca todo lo relativo a Odoo, su configuración y cualquier personalización (módulos, scripts, contenedores).
* **app-movil/:** Reúne el código y la documentación de la app móvil en Kotlin, separado claramente del ERP.
* **documentación-general/:** Centraliza los documentos de requisitos, diagramas, arquitectura y análisis de base de datos que aplican al conjunto del proyecto.

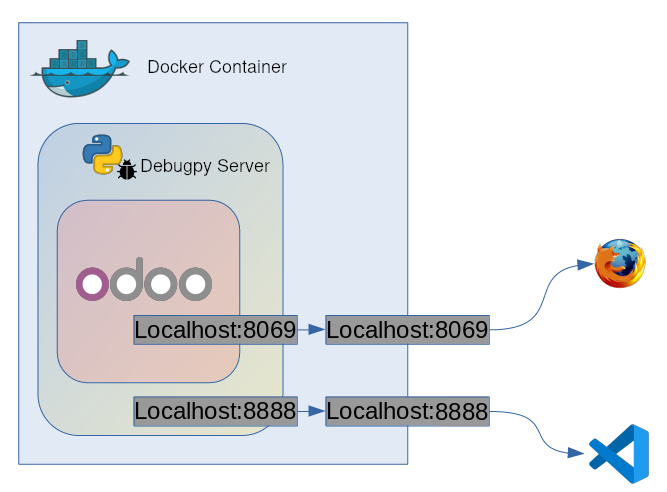


## Requisitos Funcionales y No Funcionales

Para nuestro proyecto con ERP basado en Odoo y la app móvil, definimos los siguientes requisitos para asegurar un sistema completo y robusto:

**Odoo ERP (dockerizado)**

* Gestión integral de clientes y empleados (alta, baja, modificación, roles).
* Moderación y administración de productos y comentarios (eliminar, reportar).
* Administración de categorías, etiquetas y filtros.
* Generación automática de reportes y estadísticas.
* Control de inventarios y órdenes de compra/venta.
* Gestión contable y financiera integrada.
* Acceso multiusuario con permisos personalizados.
* Integración mediante API REST para sincronización con la app móvil.



**Aplicación móvil**

* Registro e inicio de sesión seguro de usuarios.
* Publicación y edición de productos con fotos, descripción, estado y categoría.
* Búsqueda avanzada de productos con filtros múltiples (precio, distancia, categoría).
* Sistema de chat vinculado a productos para comunicación comprador-vendedor.
* Visualización de valoraciones y posibilidad de emitirlas tras compras.
* Recepción de notificaciones para mensajes, comentarios y alertas.
* Reporte de contenido ofensivo o inapropiado para moderación.
* Sincronización con ERP vía API para datos actualizados.

**Icono

El contenido generado por IA puede ser incorrecto.**

**Requisitos no funcionales**

* **Seguridad robusta:** cifrado de contraseñas, sesiones seguras y control de accesos.
* **Rendimiento optimizado:** respuestas rápidas, gestión eficiente de datos y conexiones.
* **Usabilidad:** interfaces intuitivas y accesibles tanto en móvil como en web.
* **Escalabilidad:** capacidad de crecer en usuarios y volumen de datos.
* **Mantenibilidad:** código modular, documentado y fácilmente ampliable.
* **Compatibilidad:** comunicación fiable entre móvil y ERP con formatos estándar (JSON, REST).
* **Disponibilidad:** alta estabilidad y mínimas caídas, especialmente en funciones críticas.

Este conjunto cubre el núcleo necesario para abordar el desarrollo intermodular con la base fuerte que da Odoo y la flexibilidad de Kotlin o Ionic para la app móvil, asegurando calidad, seguridad y usabilidad en ambos extremos.

## Análisis y Diseño de la BBDD

El diseño y análisis de la base de datos para nuestro proyecto se basa en el uso de PostgreSQL, que es la base de datos que utiliza Odoo y que ya tenemos dockerizada. Odoo utiliza un ORM (Object-Relational Mapping) que traduce los modelos y objetos definidos en el ERP en tablas y relaciones dentro de PostgreSQL. Cada entidad relevante para el negocio (usuarios, productos, categorías, transacciones, mensajes, valoraciones, etc.) se convierte en una tabla con sus respectivos campos y relaciones bien definidas mediante claves primarias y foráneas.

Para soportar ambas aplicaciones, tanto el ERP como la app móvil, la base de datos debe estar diseñada para ser coherente, escalable y optimizada para consultas frecuentes, con índices en campos claves para filtros y búsquedas eficientes.

En cuanto a la conexión entre la base de datos y la app móvil, no se accede directamente a la base de datos, sino que se utiliza una API REST intermedia. Esta API expone los datos y funcionalidades necesarias mediante endpoints seguros y bien definidos que la app móvil consume usando llamadas HTTP. La API además gestiona la autenticación, controles de acceso, validación de datos y asegura la integridad y seguridad del sistema.

De esta manera, la app móvil en Kotlin interactúa con la base de datos de PostgreSQL a través de la API REST, que se integra con Odoo y permite sincronizar datos, realizar operaciones CRUD (crear, leer, actualizar, eliminar) y recibir notificaciones o mensajes. Esto garantiza un diseño modular, seguro, mantenible y escalable para el proyecto.

Imagen que contiene interior, tabla, camioneta, diversos

El contenido generado por IA puede ser incorrecto.

## Diagrama de entidad relación

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.El siguiente diagrama representa el modelo entidad-relación diseñado para la aplicación móvil, mostrando las principales entidades, atributos y relaciones entre ellas:

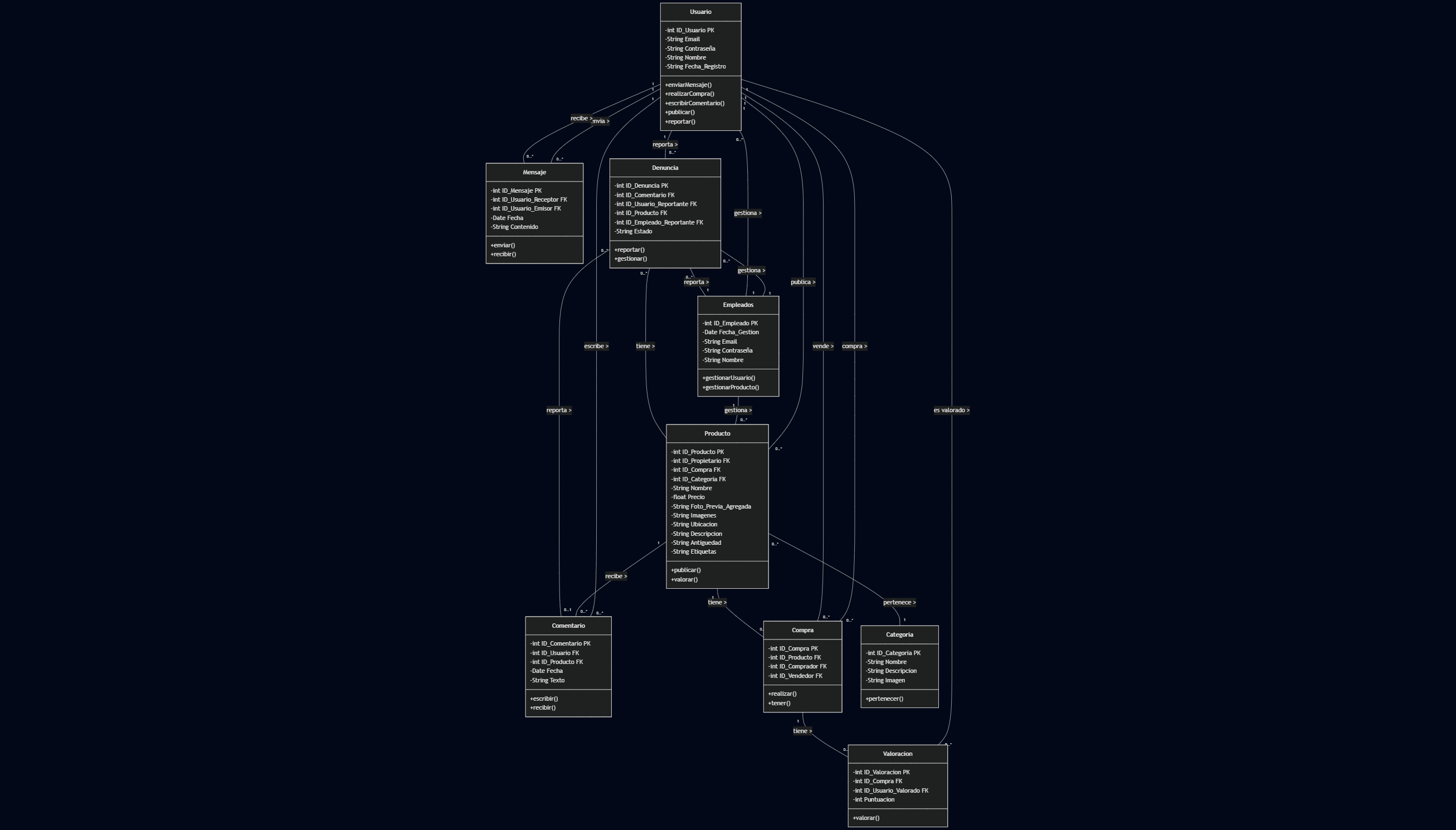
**Explicación breve**

El diagrama incluye las entidades clave del sistema (Usuario, Producto, Comentario, Compra, Valoración, Mensaje, Denuncia, Empleados, Categoría), y describe con claridad la relación entre ellas:

* Cada usuario puede publicar productos, realizar compras, escribir comentarios y emitir denuncias.
* Los productos se relacionan con categorías y pueden recibir comentarios, valoraciones y ser gestionados por empleados.
* Las relaciones muestran la cardinalidad adecuada; por ejemplo, una compra une un usuario y un producto, y sólo los usuarios compradores pueden valorar productos adquiridos.
* La entidad Empleados gestiona tanto usuarios como productos dentro del sistema, permitiendo la moderación y administración cuando sea necesario.
* Se han definido las claves primarias y foráneas para asegurar integridad referencial y poder implementar restricciones en la base de datos.

Este diagrama ER ofrece una visión clara y estructurada del modelo de datos de la app, facilitando la transición al modelo relacional y al diseño físico de la base de datos.

## Diagrama de clases

El siguiente diagrama de clases representa la estructura básica de los objetos principales que forman la lógica de negocio en nuestra app y el ERP:

**Explicación breve**

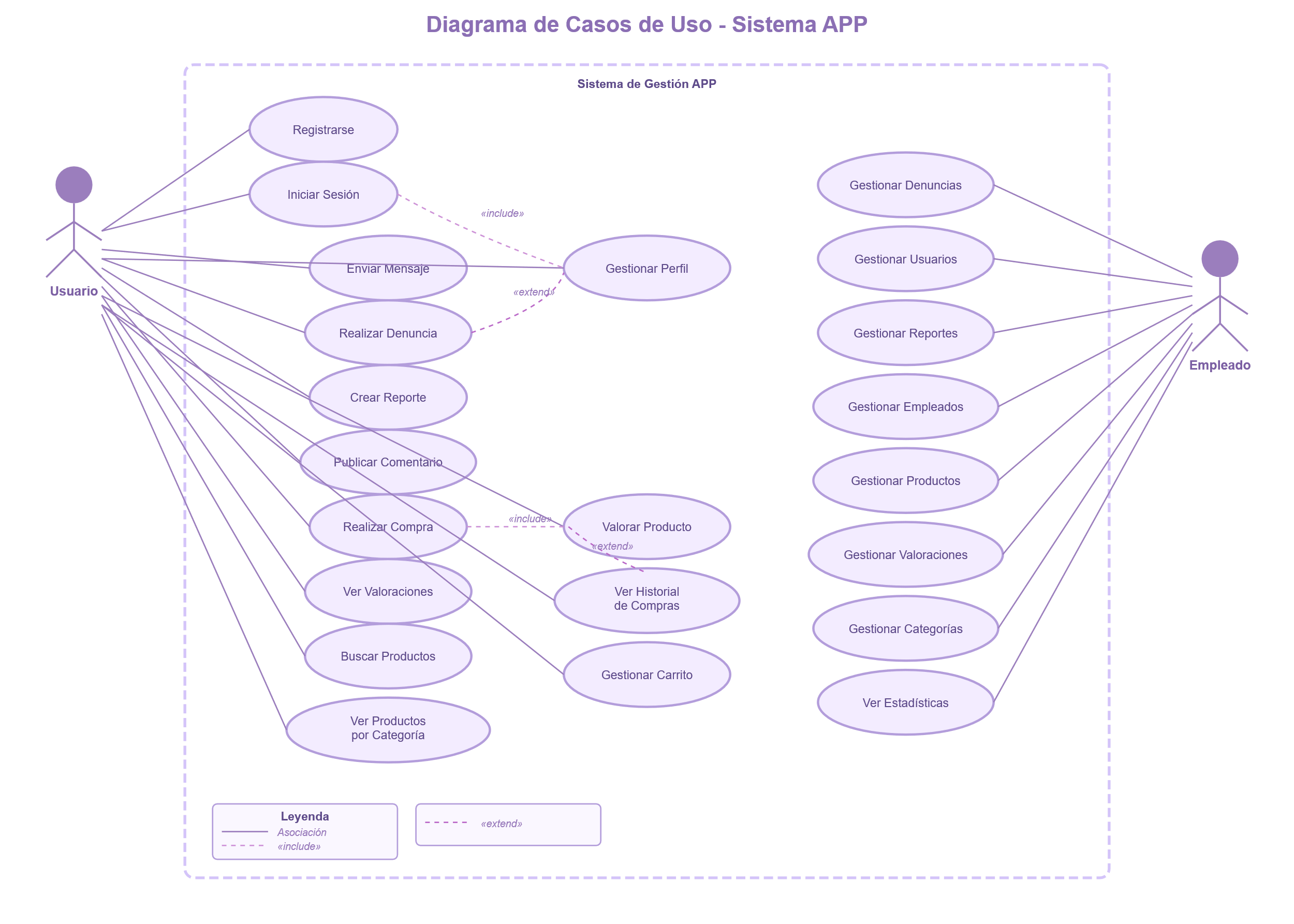
El modelo de clases está compuesto por las siguientes entidades principales:

* **Usuario:**  
  Identificado por su ID, email, contraseña, nombre y fecha de registro. Incluye métodos para enviar mensajes, realizar compras, publicar y reportar contenido, así como para escribir comentarios.
* **Mensaje:**  
  Asocia un emisor y un receptor y contiene la fecha y el contenido. Dispone de funciones para enviar y recibir comunicación entre usuarios.
* **Denuncia:**  
  Permite reportar productos o comentarios, vinculando al usuario reportante, empleado moderador y estado de la denuncia. Gestiona la moderación y seguimiento de incidentes.
* **Empleados:**  
  Gestores del sistema, con métodos para administrar usuarios y productos, manteniendo un registro de las acciones mediante fecha y permisos.
* **Producto:**  
  Propiedad de un usuario, puede pertenecer a una compra y a una categoría. Almacena nombre, precio, imágenes, ubicación, descripción y otros metadatos. Implementa métodos para publicar y valorar productos.
* **Comentario:**  
  Relaciona usuario y producto, incluye fecha y texto del comentario, y funciona para escribir y recibir aportaciones sobre los artículos.
* **Compra:**  
  Permite registrar transacciones, asociando comprador, vendedor y producto. Incluye métodos para efectuar la compra y gestionar la propiedad.
* **Categoría:**  
  Clasifica los productos y dispone de nombre, imagen y descripción.
* **Valoración:**  
  Registra las valoraciones realizadas por usuarios sobre las compras, asignando puntuación y descripción.

**Relación entre clases**

Las entidades están conectadas mediante asociaciones que reflejan tanto la dinámica de compraventa (usuarios-publican-productos, usuarios-compran-productos, usuarios-comentan-productos), como la gestión administrativa (empleados-moderan-denuncias, empleados-gestionan-productos y usuarios). Se han definido los métodos principales para cada clase, permitiendo la implementación efectiva de la lógica en el sistema y un diseño modular fácilmente extensible.

## Diagrama de casos de uso



**Explicación breve**

El diagrama representa las interacciones principales entre los dos actores del sistema (Usuario y Empleado) y los distintos casos de uso disponibles dentro de la aplicación móvil. Los usuarios pueden gestionar su cuenta, navegar productos, realizar compras y valoraciones, interactuar mediante comentarios y reportes, así como enviar mensajes.  
Por otro lado, el empleado actúa como administrador, gestionando denuncias, usuarios, reportes, productos, valoraciones, estadísticas y empleados

Se han utilizado varias relaciones «include» para representar comportamientos que forman parte obligatoria del flujo principal de otros casos de uso. Estas relaciones se emplean cuando una funcionalidad es común, necesaria y debe ejecutarse siempre dentro del caso de uso principal.

también se utilizan relaciones «extend», las cuales representan comportamientos opcionales o complementarios que se ejecutan únicamente bajo ciertas condiciones.

A diferencia del «include», que es obligatorio, el «extend» indica que el caso de uso extendido no siempre ocurre, sino que se activa solo si se cumplen determinadas circunstancias dentro del flujo principal.

## Arquitectura General del Sistema

Para garantizar una comunicación eficiente y segura entre el ERP (Odoo) y la app móvil desarrollada en Kotlin o Ionic, hemos optado por implementar una arquitectura basada en API REST como capa intermedia principal. La API REST expone los recursos y operaciones mediante endpoints bien definidos y seguros, permitiendo que las dos aplicaciones se mantengan sincronizadas y sean fácilmente escalables.

**Motivos de la elección de API REST**

Es el estándar en sistemas modernos por su simplicidad y compatibilidad con múltiples plataformas y lenguajes.

* Permite utilizar los métodos HTTP (GET, POST, PUT, DELETE) para la gestión de los datos.
* Facilita el mantenimiento, la escalabilidad y la integración con futuras aplicaciones o servicios.
* Su uso es ampliamente soportado por los frameworks tanto en Odoo como en Kotlin.

Los endpoints están organizados de modo que permiten manipular todos los recursos del sistema, mantener la integridad de los datos y facilitar la gestión tanto de la app móvil como del ERP.

**Seguridad y control**

* Todos los endpoints requieren autenticación (token JWT), y algunos validan permisos antes de permitir operaciones críticas.
* Las operaciones de gestión (moderación, administración) sólo están disponibles para empleados con el rol adecuado.

**Flexibilidad**

El sistema puede evolucionar para incluir otros estilos de comunicación (como Json-RPC) si en el futuro se necesitara optimizar ciertas operaciones, aunque actualmente REST cubre todos los requisitos del proyecto.

**Endpoints principales definidos**

| **Recurso** | **Método** | **Endpoint** | **Descripción** |
| --- | --- | --- | --- |
| Usuario | POST | /api/usuarios | Crear nuevo usuario |
| Usuario | GET | /api/usuarios/{id} | Consultar perfil del usuario |
| Usuario | PUT | /api/usuarios/{id} | Modificar datos del usuario |
| Producto | POST | /api/productos | Publicar nuevo producto |
| Producto | GET | /api/productos | Listar productos |
| Producto | GET | /api/productos/{id} | Detalle de un producto |
| Producto | PUT | /api/productos/{id} | Editar producto |
| Compra | POST | /api/compras | Registrar compra |
| Compra | GET | /api/compras/{id} | Consultar compra específica |
| Comentario | POST | /api/productos/{id}/comentarios | Añadir comentario a un producto |
| Comentario | GET | /api/productos/{id}/comentarios | Ver comentarios de un producto |
| Valoración | POST | /api/valoraciones | Crear nueva valoración sobre compra |
| Mensaje | POST | /api/mensajes | Enviar mensaje privado |
| Mensaje | GET | /api/mensajes | Listar mensajes para usuario |
| Categoría | GET | /api/categorias | Listar categorías existentes |
| Denuncia | POST | /api/denuncias | Reportar producto, comentario o usuario |
| Empleado | GET | /api/empleados | Consultar empleados (desde ERP, para administración) |
| Gestión | POST | /api/gestion/usuario | Acciones de gestión por parte de empleados |
| Gestión | POST | /api/gestion/producto | Moderación y gestión de productos |