CONSEJERÍA DE EDUCACIÓN
**Comunidad de Madrid**

# IES ENRIQUE TIERNO GALVAN

**Parla**

# CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

**Curso 2024/2025**

## Proyecto DAM

***TITULO: Tasknelia***


***Alumno*: Laura Salas Ávila**

***Tutor*: Julián Parra Perales**


Junio de 2025

# Contenido

**Contexto de la aplicación.** 4

Mundo real del problema. 4

Qué aplicaciones existen. 4

Justificación de este proyecto y diferencia del mercado. 5

Requisitos funcionales y no funcionales. 6

Casos de Uso. 7

**Diseño.** 8

GUI. 8

UI. 8

UX. 11

Diagrama de navegación. 13

Reutilización. 13

Arquitectura. 14

Despliegue. 15

Componentes. 16

Base de datos. 18

Módulos de Odoo utilizados. (¿revisar, reuniones?) 18

Uso de la base de datos en la aplicación 18

Paquetes, Interfaces y Clases. 18

Plan de pruebas. 19

**Implementación de la aplicación.** 21

Entorno de desarrollo 21

Implantación/Puesta en producción 21

**Capturas de la ejecución de la funcionalidad.** 22

Ejecución de pruebas 22

**Herramientas usadas en el proyecto para su ejecución.** 22

**Elementos destacables del desarrollo.** 24

Problemas a la hora del desarrollo. ..... 24

Error 500 en Odoo. 24

Error al intentar realizar conexión HTTPS. 24

Problemas al realizar pantallas. 25

Problemas al emular la aplicación en dispositivos móviles. 25

Innovaciones del proyecto. ..... 25

**Líneas de trabajo futuro.** 26

Requisitos pendientes e ideas pensadas para las siguientes versiones..... 26

**Conclusiones.** 27

**Bibliografía/Enlaces de interés.** 29

**Anexos.** 29

GitHub ..... 29

(revisar)

# Contexto de la aplicación.

## Mundo real del problema.

Actualmente las empresas prefieren estar más cerca de lo digital, la comodidad y también la movilidad para poder gestionarse de una manera más eficiente y rápida, ya no sirve con tener las cosas a papel o en un dispositivo fijo en la oficina, es por eso que las empresas buscan herramientas o sistemas que las permitan organizarse y tener de forma más accesible sus tareas, pedidos, reuniones con sus clientes o proveedores, incluso saber qué debe de hacer cada trabajador en cualquier momento y desde cualquier dispositivo. Es por esto que las aplicaciones ERP pueden llegar a ser muy utilizadas por las empresas para intentar tener o mejorar esa gestión de información, pero muchas de estas soluciones ERP llegan a ser algo difíciles de usar o no se llegan a necesitar de manera completa por muchos de los trabajadores, ya que ellos no necesitan usar un ERP completo, sino que necesitan usar una herramienta que les permita ver de forma rápida lo que realmente quieren. Además de que muchas de estas soluciones llegan a ser caras o están pensadas para grandes empresas, muchas de ellas no llegan a poder ser usadas de manera multiplataforma o si lo llegan a ser, no suelen ser muy agradables para los usuarios. Es por esto que se ha decidido solucionar este problema que iremos explicando en este documento, donde se creará una solución más sencilla, rápida y eficiente de una aplicación centrada en la gestión diaria de una empresa.

## Qué aplicaciones existen.

Se ha realizado un pequeño análisis del mercado para ver qué aplicaciones existen actualmente que intenten realizar o se parezcan a lo que se está intentando desarrollar en este proyecto, es decir, una aplicación con el objetivo de ayudar a las empresas en su gestión diaria de una forma más sencilla y accesible. Hay bastantes aplicaciones en el mercado que permiten a los usuarios organizar sus tareas, hacer seguimientos, repartir trabajos, etc... Pero muchas de ellas no están pensadas para poder conectarse a un ERP, ser multiplataforma o que no llegan a ser tan prácticas para el tipo de uso que se quiere dar a este proyecto. Se han escogido para realizar una comparación de nuestra idea las siguientes aplicaciones de gestión:

• **Trello y Asana:** Conocidas para organizar tareas de una manera visual, se usan en empresas para proyectos de tableros incluso asignaciones, pero no están pensadas para empresas que tienen pedidos, productos, clientes…

• **Notion:** Para organizar casi cualquier cosa con ella, pero todo hay que montarlo a mano y no tiene automatización ni conexión con otros sistemas como un ERP. Para cosas más personales es perfecta, pero para gestionar una empresa con tareas reales no es suficiente ni adecuada.

• **Monday.com y Zoho Creator:** Permiten montar soluciones un poco más a medida que son parecidas a pequeñas aplicaciones de gestión, pero muchas opciones son de pago, pueden ser complicadas de configurar y no están pensadas para centrarse en tareas como pedidos, trabajadores, entregas...

Como se puede ver en estos ejemplos, falta una aplicación más centrada en tareas reales que pueda llegar a tener una empresa en el día a día, que se pueda usar de manera más fácil e intuitiva, que no tenga muchas opciones poco útiles o altos precios, que tenga un diseño más agradable y que además se pueda conectar a uno o varios sistemas externos sin complicaciones. Por eso en este proyecto se busca ofrecer justo eso: una aplicación de gestión sencilla, que al igual que estas sea multiplataforma, pero que sea más intuitiva de usar y con un mejor diseño, accesible para cualquier tipo de empresa y que esté pensada especialmente para aquellos que necesitan organizar su trabajo diario sin tener que hacer uso de herramientas más complicadas o caras, además, a diferencia de otras soluciones, esta aplicación estará diseñada para integrarse fácilmente con un sistema ERP, permitiendo así que las empresas puedan gestionar tareas reales como pedidos, reuniones o asignaciones sin perder el tiempo navegando entre opciones que no necesitan.

## Justificación de este proyecto y diferencia del mercado.

Es por esto que en este proyecto vamos a dar solución a los problemas explicados anteriormente y vamos a diferenciarnos del resto de sistemas de gestión permitiendo a las empresas tener una forma mucho más fácil, cómoda y multiplataforma de acceder o interactuar con sus datos desde cualquier sitio y con cualquier dispositivo para poder gestionarse. La idea principal es que cualquier usuario, ya sea un trabajador o un

administrador pueda ver sus tareas, reuniones, asignaciones, pedidos o incluso más sin tener que entrar a sistemas o herramientas más grandes o complicadas de usar que muchas veces no sirven para el uso diario.

Esta solución no pretende ser un ERP, sino una aplicación de gestión que será más sencilla y rápida que se conectará a uno existente (que en este caso será Odoo) para hacer más fácil el trabajo de las personas que solo necesitan gestionar su parte. Con esto, los usuarios podrán centrarse solo en lo que necesitan hacer, sin tener que navegar por menús complicados con mucha información o entender cómo funciona un ERP. Además, no todas las empresas pueden pagar licencias caras o aplicaciones que exigen tener un gran conocimiento sobre cómo usarlas, por eso, en este proyecto también se busca que sea una opción accesible, gratuita y que pueda adaptarse a cualquier tamaño de empresa.

En resumen, en esta idea se busca el objetivo de ayudar a las empresas a organizarse mejor en su día a día sin tener que usar sistemas complicados, costosos o poco adaptables y cómodos. (revisar)

## Requisitos funcionales y no funcionales.

Repasando todo lo anterior, se han encontrado tanto los requisitos funcionales como los no funcionales, todos estos requisitos son necesarios para el desarrollo de este proyecto y vamos a mencionarlos por partes: (¿poner más texto o cambiar?)

- **Funcionales:**
  - Crear, eliminar, modificar y consultar tareas, reuniones o pedidos asignados. (¿dejar junto o separar?)

- **No funcionales:**
  - La aplicación debe poder ejecutarse en dispositivos móviles, escritorio y navegadores web.
  - La interfaz debe ser intuitiva, visualmente agradable y fácil de usar.
  - Las acciones principales deben poder realizarse en pocos pasos.
  - El sistema debe mostrar mensajes claros en caso de error o éxito.
  - El sistema debe permitir visualizar las tareas en modo lista o calendario.

- El sistema debe permitir a los usuarios visualizar información detallada de las tareas.
- El sistema debe mantener un rendimiento fluido.
- La aplicación debe usar HTTPS para cifrar las comunicaciones entre cliente y servidor.
- El sistema debe permitir a los usuarios iniciar y cerrar sesión de forma segura en cualquier momento.
- La arquitectura del sistema debe permitir futuras ampliaciones.
- Las funcionalidades deben estar restringidas según el tipo de usuario.
- El sistema debe permitir al administrador asignar tareas o pedidos a uno o varios trabajadores. (¿ningún lado o no funcional?)
- El sistema debe permitir a los trabajadores marcar sus tareas como completadas, en curso o pendientes. (¿ningún lado o no funcional?) (revisar).

## Casos de Uso.

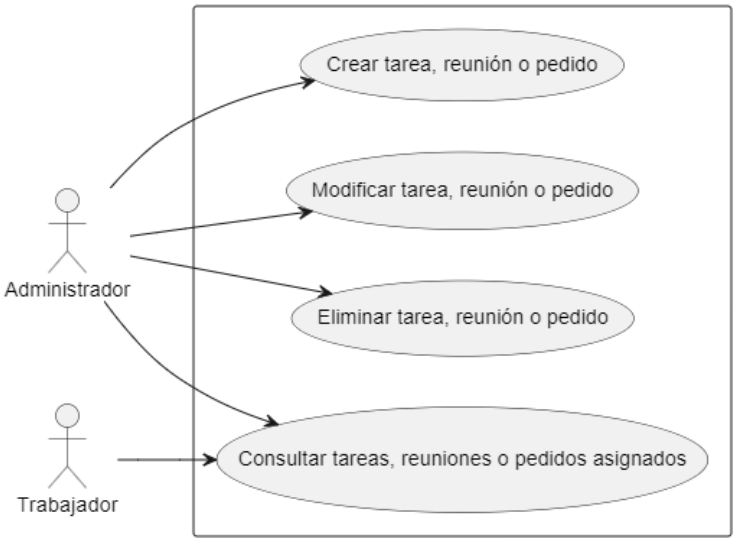
Los casos de uso se han definido a partir de los requisitos funcionales mencionados anteriormente, gracias a ellos se puede entender cómo van a interactuar los distintos actores del sistema con la aplicación, enseñando de una forma clara cuáles son las principales funcionalidades de las personas que la van a utilizar.

En este proyecto, los actores que van a interactuar con la aplicación son dos: el *Administrador*, que será quien pueda crear, asignar, modificar o eliminar tareas, reuniones o pedidos; y el *Trabajador*, que podrá consultar la información que tiene asignada, cambiar el estado de sus tareas y ver los detalles correspondientes a cada una.

(revisar)

Los dos actores tienen acceso a funcionalidades comunes como iniciar y cerrar sesión, visualizar la agenda o consultar tareas en el modo de vista calendario, anteriormente se han ido mencionado estas funcionalidades, pero en este apartado quedan organizadas de forma más visual y resumida en un diagrama de casos de uso, donde se puede ver la relación entre cada actor y las acciones que puede realizar dentro de la aplicación.

Este modelo también nos ayuda a tener una visión más clara de qué partes del sistema han sido desarrolladas, cuáles podrían añadirse o mejorar en el futuro y qué tipo de usuario necesita cada funcionalidad. (revisar)




# Diseño.

## GUI.

Se han realizado unos bocetos sobre cuál sería la idea principal para la interfaz del usuario, como se va a poder ver, es un diseño bastante agradable y simple, en cada pantalla haremos una pequeña explicación y más adelante se hablará de los aspectos de usabilidad que se han tenido en cuenta a la hora de realizar los diseños. (revisar)

## UI.

El diseño de la interfaz de usuario se ha realizado teniendo en cuenta tanto la funcionalidad como lo visual (revisar). Se han creado unos bocetos básicos usando Canva y estos están diseñados con un estilo limpio, usando los mismos colores, iconos y tipografías en cada uno de ellos, ya que el objetivo es ofrecer una experiencia visual clara, agradable y fácil de entender. Estos bocetos están centrados principalmente en los dispositivos móviles, que serán los más usado por los usuarios, aunque la aplicación también será multiplataforma, por lo que el diseño en otras pantallas será del mismo estilo, pero con diferente organización. (revisar)

A continuación, se incluyen las pantallas que se han diseñado con una pequeña explicación de que hay en cada una:

- **Pantalla de inicio de sesión**


Esta pantalla incluye los campos de correo y contraseña con iconos, y un botón de acceso bastante visible. También se ha incluido un encabezado visual con una imagen de perfil.

- **Pantalla principal (menú de navegación)**


Enseña las opciones de navegación (que serán Tareas, Pedidos, Reuniones y Agenda). Los botones están organizados de manera vertical, con gran tamaño y buena visibilidad. Se ha añadido también un calendario y un icono de chat, aunque esta funcionalidad está prevista para futuras versiones. La cuenta del usuario es visible en la zona superior.

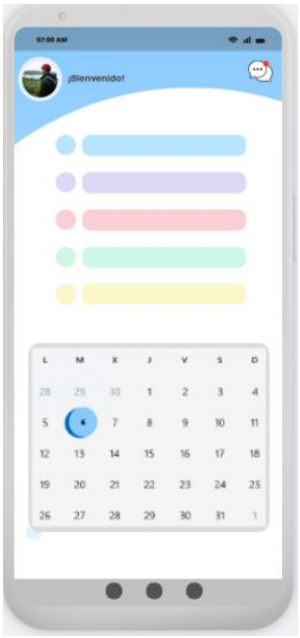
- **Pantalla de creación de tareas/pedidos/reuniones**


Esta es la pantalla que está pensada para crear tareas, reuniones o pedidos. Los campos están ordenados de forma lógica (nombre, persona asignada, fecha, descripción), y se ha buscado que el formulario sea sencillo y rápido de rellenar.

- **Pantalla de agenda (lista)**

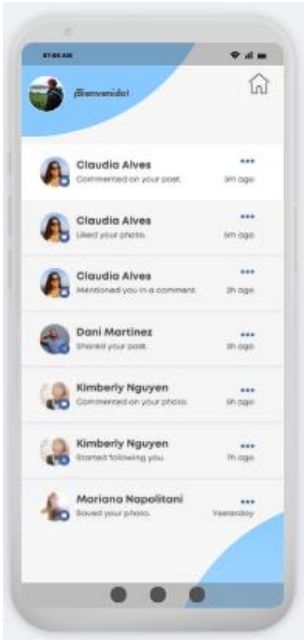

En esta vista se podrá consultar una lista ordenada de todas las tareas, reuniones y pedidos con su estado y fechas. Ha sido pensada para tener una visión rápida y organizada del trabajo, ofreciendo una forma clara de ver el trabajo pendiente.

- **Pantalla de calendario**



Esta pantalla representa una vista tipo calendario donde se pueden visualizar las tareas y eventos según la fecha, y usando colores se puede distinguir los tipos de tareas. Al hacer clic en un día, se muestran los eventos correspondientes.

- **Pantalla de chat**



Aunque esta pantalla aún no es funcional, se ha diseñado para enseñar cómo sería la comunicación entre trabajadores de la misma empresa. Está pensada para próximas versiones en la que se permitirá el envío de mensajes entre compañeros.

## UX.

La experiencia de usuario es una parte muy importante en el diseño de una aplicación, es por eso en esta se ha pensado que…. (revisar, no me gusta) Tal como se indica en los

principios de usabilidad, la UX debe garantizar efectividad, eficiencia y satisfacción en el uso de la aplicación, y todo eso se ha tenido en cuenta a la hora de realizar el diseño de la interfaz. A continuación, explicaremos los elementos principales que se han tenido en cuenta a la hora de realizar el diseño: (revisar, algo como que no)

- **Visual simple**: Se ha tenido en cuenta un diseño simple, evitando decorar de manera innecesaria y que pueda provocar confusión. Cada pantalla se centra en lo que el usuario necesita hacer.
- **Facilidad de aprendizaje:** La aplicación está pensada para que los nuevos usuarios puedan aprender a usarla rápidamente sin necesidad de documentación o una formación, ya que las pantallas tienen una organización fácil y repetitiva.
- **Navegación clara y ordenada:** El acceso a cada opción (tareas, reuniones, pedidos, agenda…) está diseñado para realizarse con pocos pasos. Los botones están claramente visibles, organizados, y con un visual coherente. (revisar)
- **Retroalimentación constante:** La aplicación se asegura de mandar mensajes claros y visibles cuando se realiza una acción, como guardar, borrar o cambiar el estado de una tarea, de esta forma nos encargamos de la necesidad de mantener informado al usuario en todo momento.
- **Consistencia visual:** Se ha mantenido un mismo estilo gráfico en todo el sistema: los mismos colores, iconos, tipografía, etc...
- **Accesibilidad y contraste**: Las pantallas están adaptadas para su uso en dispositivos móviles. Se han usado tamaños de letra adecuados, buen contraste entre texto y fondo, y botones de gran tamaño, pensando también en personas con dificultades visuales. (revisar, tema multiplataforma)
- **Satisfacción subjetiva:** Estos diseños buscan ofrecer a los usuarios una experiencia agradable, usando un lenguaje cercano, botones visuales, y tiempos de respuesta rápidos.

En resumen, se ha intentado diseñar una aplicación útil, eficiente y fácil de usar, cumpliendo con las recomendaciones de usabilidad establecidas y como se explicó en los apartados anteriores, toda la estructura está pensada para poder adaptarse a futuras mejoras sin romper esta experiencia centrada en el usuario. (revisar, algo como que no)

## Diagrama de navegación.

A continuación, se muestra el diagrama de navegación de las pantallas de nuestra aplicación, donde se puede ver cómo se conectan entre sí y cómo nos desplazaríamos por cada una de ellas. (¿muy pequeño?)


La navegación empieza en la pantalla de inicio de sesión y si el usuario introduce correctamente sus datos puede acceder a la pantalla principal.

Desde la pantalla principal puede ir a:

- La pantalla de creación de tareas/reuniones/pedidos.
- La agenda, donde verá una lista con todas sus tareas y eventos.
- El calendario, con una vista visual organizada por fechas.
- La pantalla de chat (no funcional en esta versión).
- Cada pantalla tiene opción de volver a la principal, permitiendo una navegación simple y sin pérdida de contexto.

## Reutilización.

(no sé muy bien que poner aquí)

## Arquitectura.

En este proyecto se ha desarrollado una solución formada por varios sistemas y subsistemas que trabajan juntos para poder permitir la gestión de tareas, pedidos, asignaciones y más de una manera sencilla, eficiente y desde cualquier dispositivo. La arquitectura se ha pensado para que sea clara, escalable y mantenible, y para ello se va a separar lo máximo posible los distintos elementos que formarán este sistema. Aunque algunas partes de esta, como el ERP o la base de datos, no se hayan desarrollado desde cero, sí se han configurado y conectado en el sistema. A continuación, vamos a nombrar y explicar cómo se compone esta arquitectura y cuáles son cada uno de sus componentes.

1. **Aplicación** (subsistema desarrollado):

   La aplicación ha sido desarrollada en Flutter y es la interfaz principal del sistema. Funciona como una app multiplataforma que se puede ejecutar en dispositivos móviles, escritorios, navegadores…Este subsistema ha sido desarrollado desde cero.

2. **Backend – Odoo** (subsistema independiente)

   Odoo actúa como sistema de gestión de datos y de lógica empresarial. No se ha programado desde cero, pero se ha desplegado y configurado activando los módulos necesarios para habilitar las funcionalidades necesarias para este proyecto.

3. **Base de datos – PostgreSQL** (componente interno)

   Odoo utiliza PostgreSQL como sistema de gestión de bases de datos en el que almacena todos los datos del sistema: las tareas, pedidos, usuarios, categorías, etc. A esta base de datos no se accede directamente desde la aplicación, sino a través del backend de Odoo.

4. **Capa de comunicación** (API REST sobre HTTPS)

   Toda la comunicación entre la aplicación y Odoo se realiza mediante peticiones HTTPS gracias a la configuración de un servidor proxy NGINX. Este proxy gestiona el cifrado SSL para asegurar que toda la información intercambiada esté protegida.

Con este uso de HTTPS se asegura de que el sistema sea seguro incluso cuando se accede desde otras redes externas u otros dispositivos móviles.

## Despliegue.

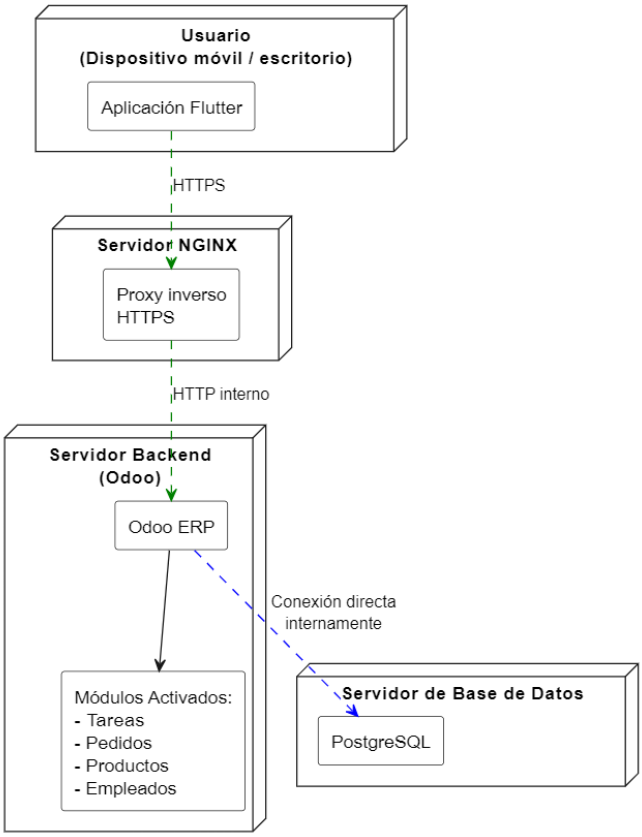
Todo el sistema ha sido desplegado mediante contenedores Docker organizados. Esta virtualización nos permite:

- **Ejecutar** Odoo, PostgreSQL y NGINX en contenedores separados pero interconectados.
- **Asegurar** de que el sistema funcione igual en cualquier máquina.

La estructura del despliegue es la siguiente:

- **Contenedor de PostgreSQL**: que contiene la base de datos.
- **Contenedor de Odoo**: ejecuta el ERP.
- **Contenedor NGINX**: actúa como proxy inverso, cifrando todas las comunicaciones del cliente a través de HTTPS.
- **Aplicación del cliente**: desarrollada en Flutter, esta se ejecuta por separado (osea fuera de Docker), conectándose al backend mediante HTTPS.

Gracias a esta configuración cada parte del sistema puede reiniciarse o actualizarse de forma separada sin afectar a los demás.




## Componentes.

A nivel de componentes individuales, el sistema desarrollado se puede dividir en las siguientes partes, cada una con una función específica dentro de esta arquitectura:

- **Frontend** (Flutter):
    - Es la interfaz principal con la que interactúan los usuarios y está diseñada para funcionar en múltiples plataformas (móvil, web, escritorio...) y permite:
        - Iniciar sesión según el tipo del usuario.
        - Visualizar tareas, pedidos, reuniones, asignaciones…
        - Crear o modificar tareas y asignarlas a otros empleados.
        - Ver la información en formato de lista o calendario.
        - Cambiar el estado de una tarea.

- **Controlador de la comunicación** (API REST)
    - Es la capa encargada de conectar la app con el backend. Se encarga de:

        - Enviar solicitudes HTTP al servidor (Odoo).
        - Interpretar respuestas JSON.
        - Gestionar errores de conexión o autenticación.
        - Garantizar que los datos estén actualizados entre el cliente y servidor.

- **Backend** (Odoo ERP)
    - Odoo funciona como backend. En este componente se han configurado:
        - Módulos de tareas, productos, empleados, contactos...
        - Control de acceso por tipo de usuario.

- **Base de datos** (PostgreSQL)
    - Gestionada internamente por Odoo, es el almacenamiento de todos los datos de este proyecto:
        - Tareas, productos, usuarios, reuniones, asignaciones....
        - Solo accesible mediante Odoo.

- **Servidor NGINX**
    - Funciona como proxy inverso y se encarga de:
        - Redirigir el tráfico externo al backend de Odoo.
        - Asegurar que las comunicaciones sean HTTPS.
        - Mejorar la seguridad del sistema al controlar el acceso.

- **Entorno de despliegue** (Docker)
    - Todos los servicios anteriores se han desplegado en contenedores Docker, y esto permite:
        - Ejecutar Odoo, PostgreSQL y NGINX.
        - Iniciar el entorno de desarrollo fácilmente.
        - Asegurar que todo funcione igual en distintos entornos.

## Base de datos.

Nuestro proyecto hace uso de Odoo como ERP, y este utiliza una base de datos PostgreSQL que se encarga de guardar toda la información de los módulos que tenga Odoo. Esta base de datos no fue creada desde cero ya que al hacer la dockerización de Odoo, este se encargó de realizar las tablas y las relaciones necesarias dependiendo de los módulos que se van activando. Asi que, el diseño de la base de datos no forma parte directa del trabajo que se está desarrollando, pero sí es importante saber qué datos utiliza la aplicación y cómo se accede a ellos.

## Módulos de Odoo utilizados. (¿revisar, reuniones?)

### Uso de la base de datos en la aplicación

Desde la aplicación, se accede a los datos de Odoo mediante su API. Estos datos son de cada una de las tablas internas que han sido creadas por los módulos que se han activado. Como ejemplo:

- Cuando un usuario consulta sus tareas asignadas, la app accede a la información de los módulos de pedidos y empleados.
- Al crear una nueva tarea, se registra automáticamente en el sistema de Odoo.
- Si se actualiza el estado de una entrega, se modifica directamente el campo necesario en la base de datos de Odoo.

En resumen, la aplicación no tiene control de una manera directa sobre la estructura de la base de datos, pero sí la utiliza a través del backend, por lo tanto, el diseño de base de datos en este proyecto se basa en saber qué módulos se van a necesitar, qué entidades se van a utilizar y cómo se relacionan entre ellas dentro de Odoo.

## Paquetes, Interfaces y Clases.

(¿revisar? No sé bien que poner)

## Plan de pruebas.

El plan de pruebas que se usará para poder darle el visto bueno a nuestra solución está formado por una serie de pruebas que se deben de cumplir y funcionar correctamente, estas pruebas que explicaremos a continuación nos ayudarán a descubrir posibles errores en la aplicación:

1. **Pruebas de arranque y conexión:**
    - Verificar que la aplicación inicia correctamente desde distintos dispositivos (móvil, web, escritorio…).
    - Comprobar que se establece la conexión con el sistema de gestión (backend).
    - En el caso de error de conexión, la aplicación debe mostrar un mensaje al usuario.
2. **Pruebas de acceso y control de usuarios:**
    - El sistema debe permitir el inicio de sesión con usuarios registrados.
    - Según el tipo de usuario (administrador o trabajador), se debe ver el contenido y las funcionalidades que le correspondan.
    - Probar que un trabajador no puede acceder a funciones que no le correspondan.
3. **Pruebas de gestión de tareas:**
    - Crear nuevas tareas de distintos tipos (pedido, reunión…).
    - Asignar tareas a trabajadores desde el administrador.
    - Cambiar el estado de una tarea (por ejemplo, de pendiente a completada) y comprobar que se actualiza correctamente.
    - Visualizar tareas asignadas por fecha o por tipo. (ir viendo)
4. **Visualización y navegación:**
    - Asegurarse de que los datos (usuarios, tareas, productos...) se ven correctamente en la aplicación.

- Confirmar que el calendario o la lista de tareas muestra la información que se espera.
- Probar la navegación entre pantallas para asegurar que no hay errores ni bloqueos.

5. **Comunicación entre cliente y servidor:**
   - Confirmar que los datos creados en la app se ven correctamente en Odoo.
   - Comprobar que los datos modificados desde Odoo se actualizan en la aplicación.
   - Simular una desconexión o un fallo en la red para ver cómo funcionaría el sistema.
6. **Pruebas en diferentes plataformas:**
   - Ejecutar la aplicación en dispositivos móviles, navegadores web y escritorio para comprobar la compatibilidad y el correcto funcionamiento de cada uno de ellos.
   - Verificar que el diseño se adapta bien a los distintos tamaños de pantalla.
7. **Pruebas de rendimiento:**
   - Crear una gran cantidad de tareas y comprobar que la aplicación sigue funcionando sin problemas.
   - Medir el tiempo que tarda en cargar una lista de tareas larga.
8. **Pruebas de usabilidad:**
   - Validar que la interfaz es clara y fácil de usar para un usuario sin conocimientos.
   - Comprobar que las acciones se intentan realizar en pocos pasos.
9. **Pruebas de seguridad y control:**
   - Confirmar que los datos de los usuarios no se comparten entre perfiles distintos.

- Comprobar que solo los usuarios autorizados puedan realizar ciertas acciones (crear tareas, asignar, eliminar...).

# Implementación de la aplicación.

La implementación de este proyecto ha seguido una planificación ordenada basada en el análisis y diseño realizados previamente. Aunque no se han desarrollado todos los casos de uso al completo, se ha creado la base funcional necesaria para comprobar que el sistema cumple su objetivo y se puede ampliar en el futuro sin problemas. A continuación, se explican dos aspectos clave de la implementación: el entorno de desarrollo utilizado y la futura puesta en producción del sistema.

## Entorno de desarrollo

Para desarrollar esta aplicación se ha utilizado Flutter como framework principal y Dart como lenguaje, el entorno de desarrollo ha sido Visual Studio Code que se ha configurado con las extensiones necesarias para trabajar con Flutter, Dart y GitHub.

(realizar desarrollo con buenas prácticas siguiendo una estructura limpia y organizada del código, uso de control de versiones mediante GitHub, y organización del trabajo. Ir añadiendo y explicando)

La estructura del proyecto incluye:

- lib/: carpeta principal del código fuente.

El backend utiliza Odoo, que se ha desplegado junto con la base de datos PostgreSQL mediante Docker usando un archivo docker-compose.yml, lo que facilita el arranque y la configuración del entorno sin tener que instalar manualmente cada servicio.

El backend utiliza Odoo y se ha desplegado junto con la base de datos PostgreSQL mediante Docker usando un archivo docker-compose.yml, lo que facilita el arranque y la configuración del entorno.

## Implantación/Puesta en producción

El proyecto ha sido diseñado para que su despliegue y ejecución sea sencillo y puedes ejecutarse en distintos entornos. Para ello, se han tomado en cuenta los siguientes aspectos: (ir añadiendo)

El proyecto ha sido diseñado para que su despliegue y ejecución sea sencilla y pueda ejecutarse en distintos entornos (desarrollo, pruebas, producción...). Para ello, se han tenido en cuenta los siguientes aspectos:

- Se ha utilizado Docker para encapsular tanto Odoo como PostgreSQL y facilitar la portabilidad del sistema.
- Se ha intentado configurar un servidor **NGINX** como **proxy inverso** para asegurar la conexión HTTPS, aunque en esta primera versión no se ha conseguido dejar del todo funcional por problemas de red y puertos. Aun así, la estructura está preparada.
- La aplicación Flutter puede compilarse tanto para Android como para escritorio o web, lo que facilita su despliegue en distintos dispositivos.
- El backend queda preparado para integrarse con la app mediante peticiones HTTPS a su API REST.

Por último, como recomendación, se han dejado preparados los archivos necesarios para que otras personas puedan clonar el repositorio, levantar los contenedores de Docker y ejecutar la aplicación sin complicaciones. También se han documentado algunas instrucciones básicas en el propio README del repositorio de GitHub.

## Capturas de la ejecución de la funcionalidad.

### Ejecución de pruebas

## Herramientas usadas en el proyecto para su ejecución.

Para el desarrollo de este sistema de gestión he utilizado una serie de herramientas que permiten que la aplicación funcione correctamente y pueda cumplir con los objetivos propuestos en los apartados anteriores. A continuación, se mencionarán cada una de ellas con una pequeña explicación de su uso:

- **Flutter:**

    - Framework utilizado para el desarrollo de la interfaz del usuario (frontend). Permite crear aplicaciones multiplataforma desde un único código.
- **Dart:**
    - Lenguaje de programación usado por Flutter para implementar la lógica y el comportamiento de la aplicación.
- **Odoo:**
    - ERP de código abierto que se ha utilizado como backend para gestionar los datos de productos, tareas, usuarios, etc.
- **PostgreSQL:**
    - Sistema de gestión de bases de datos que utiliza Odoo internamente para almacenar la información.
- **Docker:**
    - Plataforma de contenedores utilizada para desplegar Odoo y su base de datos de forma aislada y replicable.
- **NGINX:**
    - Servidor web que se ha utilizado como proxy inverso para habilitar HTTPS y mejorar la seguridad de las conexiones.

Además de estas herramientas necesarias para la ejecución, durante el desarrollo del proyecto se han utilizado otras que, aunque no forman parte directa del funcionamiento final, han sido fundamentales para su implementación:

- **Visual Studio Code:** Entorno de desarrollo utilizado para programar y organizar todo el código fuente del proyecto.
- **Git y GitHub:** Control de versiones y plataforma de almacenamiento en la nube que ha permitido gestionar el progreso del proyecto de forma ordenada.
- **Canva:** Herramienta online de diseño gráfico utilizada para crear los prototipos de interfaz de usuario (UI), diagramas de navegación, y elementos visuales incluidos en la documentación. (revisar)

# Elementos destacables del desarrollo.

A medida que se ha ido realizando el proyecto, fueron surgiendo innovaciones y varios problemas que se fueron solucionando poco a poco, vamos a explicar cada uno de ellos en diferentes apartados y en el caso de los problemas, serán explicados en el orden en el que aparecieron mientras se iba avanzando.

## Problemas a la hora del desarrollo.

### Error 500 en Odoo.

Al terminar de montar Odoo y su base de datos en Docker e intentar acceder desde el navegador me aparecía el error 500 (un error interno del servidor) que no me dejaba acceder a él. Después de revisar los mensajes de error que salían en la consola e investigar un poco, me di cuenta de que el problema venía por dos cosas:

- **Por algunos puertos** que estaba intentando usar y ya estaban ocupados, así que Odoo no podía iniciarse bien.
- Y **por la base de datos** que no estaba configurándose del todo bien al arrancar, así que Odoo no podía conectarse a ella correctamente.

Después de comprobarlo, lo que hice para solucionarlo fue cambiar los puertos para que no hubiera conflicto y añadir unas líneas de código que me faltaban en el archivo de configuración que usé para el montaje (llamado docker-compose) para arrancar de forma correcta la base de datos.

Cuando terminé de configurarlo, volví a arrancar el sistema y ahora sí funcionaba todo como debería, permitiéndome acceder a Odoo y tener la base de datos.

### Error al intentar realizar conexión HTTPS.

Al intentar configurar la conexión segura mediante HTTPS para mi aplicación utilizando un proxy inverso con NGINX, hubo varios problemas que hicieron que la conexión no funcionara. El objetivo era poder acceder a Odoo de forma segura, usando un dominio personalizado con un certificado SSL para cifrar los datos, pero esto acabó siendo más complicado de lo que se pensaba y terminó en error. Después de revisar los

mensajes de error e investigar un poco, me di cuenta de que el problema venía por estas dos cosas:

- **Por los puertos de nuevo**, los puertos que se estaban intentando usar no estaban abiertos por lo que terminaba en error todo el rato.
- Por el **fallo en la herramienta certbot** que nos permitiría tener un certificado SSL gratuito usando Let's Encrypt, este error era debido a los puertos, ya que al no estar abiertos no se podía validar el dominio por la falta de acceso a estos y el certificado no pudo generarse.

Este problema por el momento aún no se ha terminado de solucionar, pero esta todo configurado (NGINX, Docker, Certbot y dominio con DuckDNS) y preparado para su funcionamiento.

## Problemas al realizar pantallas.

Al estar realizando las pantallas hubo un pequeño problema con

## Problemas al emular la aplicación en dispositivos móviles.

Al intentar probar la aplicación en un emulado móvil, la aplicación no aparecía en pantalla, pero si intentaba emular la aplicación desde una página web o en Windows, sí que funcionaba, investigando el por qué la aplicación no iba en los emuladores móviles me di cuenta de que los problemas eran los siguientes:

## Innovaciones del proyecto.

En este proyecto se han usado varias herramientas que se consideran innovaciones si se tiene en cuenta otros desarrollos que se han realizado anteriormente. Estas innovaciones no solo han hecho que el proyecto sea algo más actual, sino que también sea más completo, entretenido e interesante. Las principales innovaciones que se han realizado en este proyecto han sido las siguientes:

- **Flutter como framework multiplataforma:**

  Se ha usado Flutter para desarrollar la aplicación, permitiendo que funcione tanto en Android como en escritorio, web... Esta tecnología es actual, bastante

usada y está en crecimiento. Aunque no se había trabajado antes con Flutter, se ha aprendido a usarlo durante el proyecto.

- **Despliegue con Docker:**

Para facilitar el despliegue del backend (Odoo + PostgreSQL), se ha utilizado Docker y docker-compose. Esto permite desplegar todo el entorno en solo unos segundos y se encarga de asegurarse de que funcione igual en distintos dispositivos.

- **Conexión segura mediante HTTPS y NGINX:**

Se ha configurado un proxy inverso con NGINX para habilitar el acceso seguro (HTTPS) a la aplicación, haciendo uso de certificados SSL y el manejo de puertos. Aunque se ha hecho algo difícil su uso, ahora se ha entendido cómo poder usar este tipo de seguridad en aplicaciones reales.

- **Uso de dominio dinámico DuckDNS:**

Para poder acceder a la aplicación desde fuera de la red local sin tener un dominio de pago se ha usado DuckDNS que es un servicio gratuito de DNS dinámico, con esta herramienta se ha podido probar y desplegar el sistema con un dominio propio sin la necesidad de pagar.

# Líneas de trabajo futuro.

## Requisitos pendientes e ideas pensadas para las siguientes versiones.

En esta primera versión de la aplicación solo se pondrá en funcionamiento alguno de los apartados de esta, pero en un desarrollo futuro todas las opciones estarían disponibles, mejoradas e incluso se podrían añadir más, logrando que la aplicación sea mucho mejor. Algunos de los requisitos o ideas pensadas para las futuras versiones de la aplicación son las siguientes:

• Mejorar los niveles de seguridad de la aplicación.

• Que la aplicación sea muy escalable y no dé errores.

• Tener una mejor accesibilidad para personas con discapacidades.

• Poder realizar actualizaciones y añadir nuevas opciones/módulos.

• Añadir automatización en ciertos apartados de la aplicación, incluso hacer

uso de inteligencia artificial para crear diagramas o analizar datos que den reportes al usuario.

• Tener notificaciones en tiempo real e incluso chats entre usuarios de la misma empresa para el paso de información.

• Mejorar el seguimiento de las tareas y sus estados.

• Mejorar los roles de los usuarios y todos sus permisos.

## Conclusiones.

Como conclusión final de este proyecto he preparado un pequeño análisis DAFO para resumir de una forma más clara todo lo que se ha aprendido durante el desarrollo de este y también los puntos más importantes del proceso que se ha realizado. Este análisis me ha servido para ver de manera más rápida y organizada tanto los puntos fuertes como las cosas que se pueden ir mejorando en futuras versiones o en futuros proyectos. Las conclusiones finales son las siguientes:

- **Fortalezas:**
    - El proyecto es multiplataforma y está bien estructurado.
    - Se han utilizado tecnologías que son consideradas actuales (Flutter, Docker, Odoo...).
    - La solución que se ha desarrollado es intuitiva, sencilla y fácil de usar.
    - La arquitectura del proyecto permite añadir en el futuro nuevas funcionalidades fácilmente.

- **Debilidades:**
    - Algunas configuraciones como la del proxy HTTPS han sido difíciles y complicadas de realizar.

- No se ha llegado a usar toda la lógica del backend, ya que solo se ha usado la necesaria para realizar una prueba en este proyecto.
- No tener conocimientos en Flutter al inicio, cosa que ha afectado un poco al desarrollo.

- **Oportunidades:**
  - Puede seguir desarrollándose en el futuro y convertirse en una herramienta real que puedan usar las empresas.
  - En el futuro puede tener nuevas funcionalidades como las notificaciones, automatizaciones con IA o conexiones con otros sistemas externos que harán que la aplicación sea mucho mejor.
  - Servirá como base para otros proyectos que puedan ser parecidos.

- **Amenazas:**
  - Problemas de compatibilidad en Flutter o Docker si no se mantiene actualizado.
  - La necesidad de tener conocimientos técnicos para modificar o mejorar la aplicación si lo usaran otras personas.

En resumen, este proyecto creo que me ha ayudado mucho a mejorar a nivel técnico y personal. He aprendido muchas cosas y a trabajar con herramientas que no había usado antes, como Flutter y Docker, también he tenido que investigar, resolver problemas reales por mi cuenta… Aunque al principio algunas cosas se me hicieron complicadas, como la configuración del proxy HTTPS o tener una estructura limpia en Flutter, al final he conseguido desarrollar la aplicación con la idea que tenía en mente. Creo que el proyecto puede seguir mejorando en futuras versiones y me quedo con todo lo que he aprendido durante el proceso, tanto las cosas buenas como las malas, ya que me han servido para entender mejor cómo se desarrolla una aplicación desde cero con otro lenguaje junto con otras herramientas y que se conectara a un sistema real. Ha sido un proyecto muy entretenido de hacer por toda la investigación realizada y estoy deseando seguir mejorando con estas herramientas en el futuro.

# Bibliografía/Enlaces de interés.

**Capterra.** (25 de 3 de 2025). *Best Project Management Software*. Obtenido de Capterra.com: https://www.capterra.com/project-management-software/

**Cloudbooklet**. (30 de 4 de 2025). *Install Odoo 17 on Ubuntu with Docker, NGINX and SSL*. Obtenido de Cloudbooklet.com: https://www.cloudbooklet.com/developer/install-odoo-17-on-ubuntu-with-docker-nginx-and-ssl/

**DigitalOcean**. (10 de 11 de 2024). *How To Secure Nginx with Let's Encrypt on Ubuntu 22.04*. Obtenido de DigitalOcean.com: https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-22-04

**FilledStacks**. (25 de 2 de 2025). *Flutter and Provider Architecture using Stacked*. Obtenido de FilledStacks.com: https://www.filledstacks.com/post/flutter-and-provider-architecture-using-stacked/

**Google**. (21 de 5 de 2025). *Architecture recommendations*. Obtenido de Flutter.dev: https://docs.flutter.dev/app-architecture/recommendations

**Google**. (18 de 5 de 2025). *Flutter documentation*. Obtenido de Flutter.dev: https://docs.flutter.dev

**Reso Coder**. (12 de 3 de 2025). *Flutter Clean Architecture & TDD Course*. Obtenido de Reso Coder: https://resocoder.com/flutter-clean-architecture-tdd/

# Anexos.

## GitHub

Repositorio de GitHub con el proyecto subido: LauraSA29/Proyecto_TFG_Laura