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

1. Contexto de la aplicación

1.1 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 forma 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.

1.2 ¿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 o 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 forma visual, se usan en empresas para proyectos de tableros incluso tienen 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 poder integrarse de forma fácil con un sistema ERP, permitiendo así que las empresas puedan gestionar tareas reales como pedidos, reuniones o asignaciones sin tener que perder el tiempo navegando entre opciones que no necesitan.

1.3 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 a 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 para usarlo. Además, no todas las empresas pueden pagar por licencias caras o aplicaciones en las que se necesita 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.

Resumiendo, en esta idea se busca el objetivo de destacar y ayudar a las empresas a organizarse mejor en su día a día sin tener que usar sistemas complicados, muy caros o poco adaptables y cómodos.

1.3.1 Requisitos funcionales y no funcionales

Repasando todo lo anterior, se han encontrado tanto requisitos funcionales como los no funcionales que son necesarios para el desarrollo y el funcionamiento de la aplicación. Estos requisitos se han definido a partir del análisis y todos los objetivos mencionados anteriormente, se explicarán por separado para dejar más claro qué se espera que haga el sistema y cómo debe comportarse en general.

- **Funcionales:**
    - Crear, eliminar, modificar y consultar tareas.
    - Gestionar reuniones.
    - Gestionar pedidos asignados.
    - El sistema debe permitir al administrador asignar tareas, reuniones o pedidos a uno o varios trabajadores.

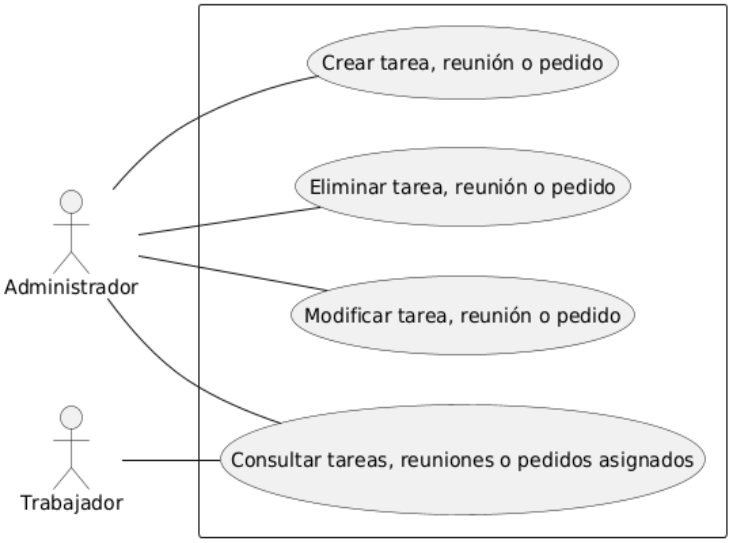
- **No funcionales:**
    - La aplicación debe poder ejecutarse en dispositivos móviles, escritorio y navegadores web.
    - La interfaz debe ser intuitiva, visualmente agradable y fácil de usar.
    - Las acciones principales deben poder realizarse en pocos pasos.
    - El sistema debe enseñar mensajes claros en caso de error o éxito al usuario.
    - 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 o mejoras.
    - Las funcionalidades deben estar restringidas según el tipo de usuario que esté usando la aplicación.

1.3.2 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 usuarios del sistema con la aplicación, enseñando de una forma clara y gráfica cuáles son las principales funcionalidades de las personas que la van a utilizar.

En este proyecto, los usuarios que van a interactuar con la aplicación son dos: el *administrador*, que será quien pueda crear, asignar, modificar, cambiar 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 de cada una. Los dos usuarios también tienen acceso a funcionalidades comunes como el iniciar y cerrar sesión, visualizar la agenda o consultar tareas en el modo de vista calendario…

A continuación, este diagrama de casos de uso nos ayudará a tener una visión mucho 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.



2. Diseño

2.1 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 de qué contiene y más adelante se hablará de los aspectos de usabilidad que se han tenido en cuenta a la hora de realizar los diseños.

2.1.1 UI

El diseño de la interfaz de usuario se ha realizado teniendo en cuenta tanto la funcionalidad como lo visual. 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 lograr ofrecer al usuario una experiencia visual que sea clara, agradable y fácil de entender. Estos bocetos están diseñados principalmente para los dispositivos móviles, que serán los más usado por los usuarios, aunque como ya se explicó antes, la aplicación también será multiplataforma, por lo que el diseño en otras pantallas será del mismo estilo, pero con diferente tamaño.

A continuación, se incluyen las pantallas diseñadas con una pequeña explicación sobre cada una de ellas:

- **Pantalla de inicio de sesión:**

Esta es la pantalla de inicio de sesión, que incluye los campos de correo y contraseña junto con iconos, un botón de acceso bastante visible y también se ha incluido un encabezado que tendrá la imagen de perfil del usuario.

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

En la pantalla principal están 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.

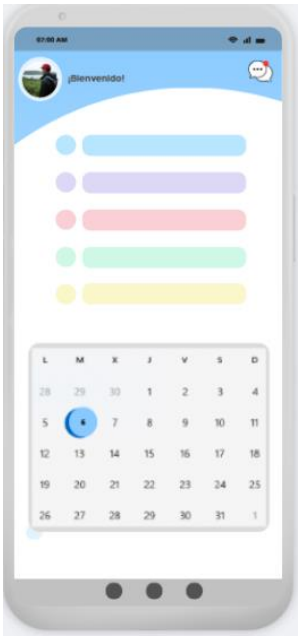
- **Pantalla de creación de tareas/pedidos/reuniones:**

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 para el usuario.

- **Pantalla de agenda (lista):**

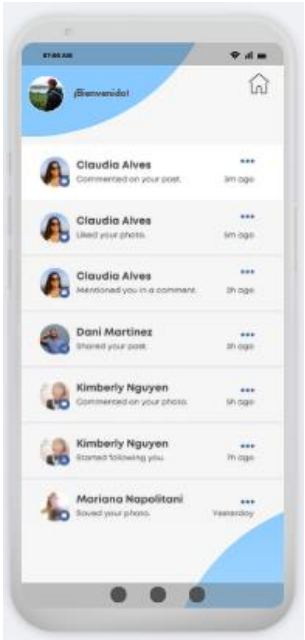
Esta es la pantalla de agenda, donde se podrá consultar una lista ordenada de todas las tareas, reuniones y pedidos con su estado y fechas. Ha sido pensada para tener una visión rápida y organizada del trabajo, logrando ofrecer a los usuarios una forma clara de ver el trabajo que tienen pendiente o hecho.

- **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.

2.1.2 UX

La experiencia de usuario es una parte muy importante en el diseño de una aplicación, y es por eso que se ha tenido muy en cuenta en este proyecto. Como se indica en los principios de la usabilidad, la UX debe de garantizar la efectividad, eficiencia y satisfacción al usuario al usar la aplicación, y todo eso se ha tenido en cuenta a la hora de realizar el diseño de esta interfaz. A continuación, se explican los elementos más importantes que se han tenido en cuenta a la hora de realizar el diseño:

- **Visual simple**: Se ha tenido en cuenta un diseño simple, evitando decorar de una manera muy excesiva y que pueda provocar confusión o incomodidad al usuario. Cada pantalla se centra solo en lo que el usuario necesita hacer.

- **Facilidad de aprendizaje:** La aplicación está pensada para que los usuarios nuevos puedan aprender a usarla rápidamente sin la necesidad de documentación o una formación, ya que las pantallas tienen una organización fácil, simple y repetitiva.

- **Navegación clara y ordenada:** El acceso a cada opción de la aplicación (tareas, reuniones, pedidos, agenda…) está diseñado para poder realizarse en pocos pasos. Los botones están claramente visibles, organizados, y visualmente son agradables.

- **Retroalimentación constante:** La aplicación se asegura de mandar mensajes claros y visibles al usuario 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**: Se han usado tamaños de letra adecuados, buen contraste de colores entre texto y fondo, y botones de tamaño adecuado, pensando también en personas con dificultades visuales.

- **Satisfacción:** Estos diseños buscan como objetivo poder ofrecer a los usuarios una experiencia que les sea agradable, usando un lenguaje cercano y simple, con botones visuales, y tiempos de respuesta que son rápidos.

2.1.3 Diagrama de navegación

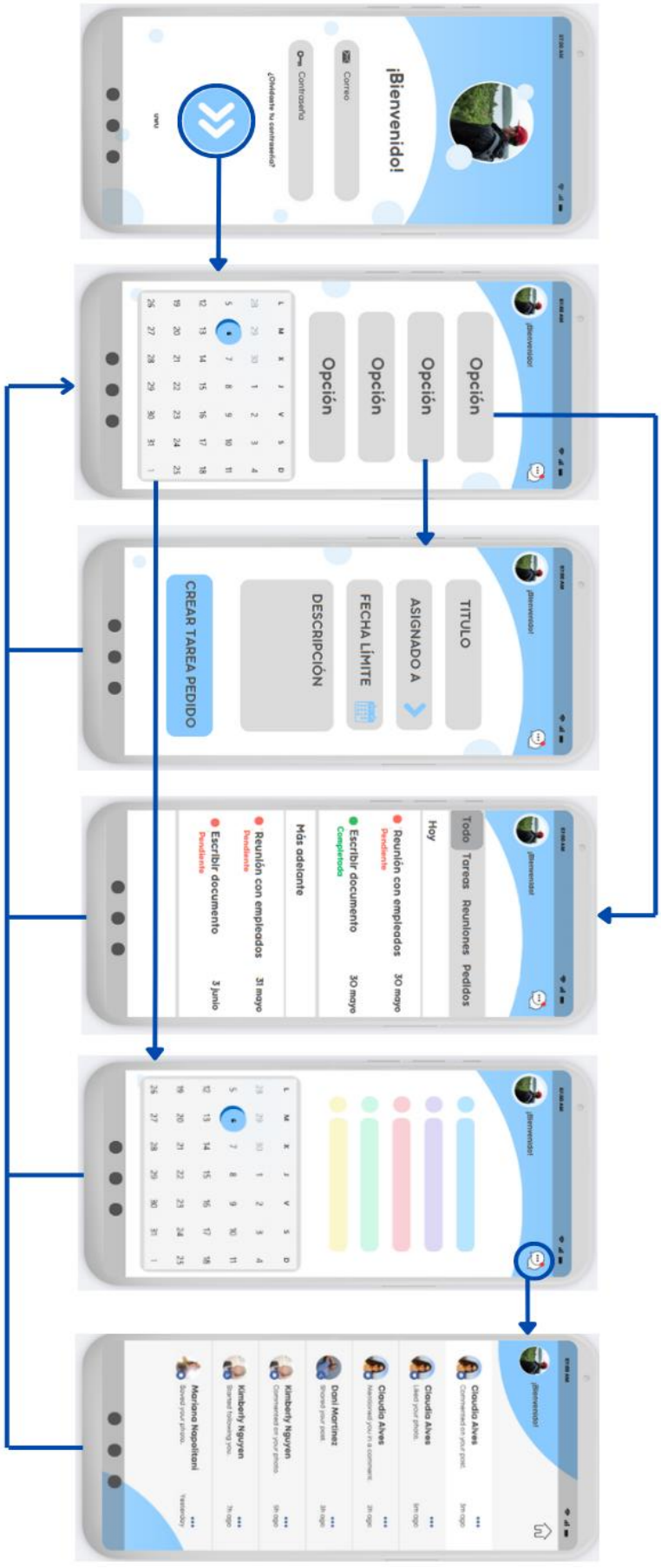
A continuación, se muestra el diagrama de navegación de las pantallas de nuestra aplicación, donde se puede ver cómo se conectan entre sí y cómo nos desplazaríamos por cada una de ellas.

(Este diagrama debido a la falta de espacio y la necesidad de poder verlo correctamente, ha sido colocado en la siguiente página.)

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 y colores.
- La pantalla de chat.
- Cada pantalla tiene opción de volver a la principal, permitiendo una navegación simple y sin la pérdida de contexto para el usuario.


¡Bienvenido!
Correo
Contraseña
¿Olvidaste tu contraseña?
www
¡Bienvenido!
Opción
Opción
Opción
Opción
¡Bienvenido!
TITULO
ASIGNADO A
FECHA LÍMITE
DESCRIPCIÓN
CREAR TAREA PEDIDO
¡Bienvenido!
Todo
Tareas
Reuniones
Pedidos
Hoy
Reunión con empleados
Pendiente
30 mayo
Escribir documento
Completado
30 mayo
Más adelante
Reunión con empleados
Pendiente
31 mayo
Escribir documento
Pendiente
3 junio
¡Bienvenido!
¡Bienvenido!
Claudia Alves
Commented on your post.
3m ago
Claudia Alves
Liked your photo.
3m ago
Claudia Alves
Mentioned you in a comment.
3h ago
Dani Martinez
Shared your post.
3h ago
Kimberly Nguyen
Commented on your photo.
3h ago
Kimberly Nguyen
Started following you.
7m ago
Mariano Napolitani
Shared your photo.
Yesterday

2.1.4 Reutilización

Durante el desarrollo se han llegado a reutilizar varias partes en el diseño con el objetivo de evitar duplicar código, agilizar el proceso y mantener una estructura más limpia. Este apartado se dividirá en dos partes, en las cosas reutilizadas y en las cosas que deberían ser reutilizadas en el futuro, a continuación, se irán explicando cada uno de ellos:

- **Partes reutilizadas:**
    - **Colores:** Una de las cosas más reutilizadas ha sido la paleta de colores de la aplicación, ya que se creó una clase llamada Colores donde están definidos todos los colores principales que se usan en la aplicación. De esta manera, cada vez que se necesita un color (para los fondos, textos o campos…), se utiliza desde esta clase, facilitando futuros cambios.

    - **Encabezado:** Otra parte que se ha reutilizado bastante es el encabezado con la imagen del usuario que aparece en varias pantallas. Este encabezado muestra el nombre del usuario, su imagen de perfil y los botones de chat y cerrar sesión/volver. En resumen, se ha hecho un widget propio con diseño personalizado y se usa en varias pantallas sin tener que repetir todo el código.

    - **Rutas:** También se ha aprovechado el sistema de rutas centralizadas para evitar tener que escribir a mano cada ruta de navegación. Esto permite tenerlas todas organizadas en un solo archivo y hace más fácil navegar entre pantallas, cambiar o añadir rutas si fuera necesario más adelante.

- **Partes que deberían ser reutilizadas en el futuro:**
    - Por último, en este apartado se hablará de ciertas partes de la aplicación que deberían ser reutilizadas en el futuro ya que no se han podido realizar en esta versión.

    - **Pantallas**: hay pantallas que comparten estructuras similares (por ejemplo, la de creación de tareas/reuniones/pedidos o de usuarios y

cambio de contraseña, la pantalla de crear/editar tarea, la imagen de fondo…). Soy consciente de que estos apartados deben de ser reutilizados y lo serán en futuras versiones de la aplicación.

Gracias a estas reutilizaciones o futuras reutilizaciones, el desarrollo ha podido ser un poco más rápido, el código más claro y ahora es más sencillo mantener o modificar partes de la aplicación sin tener que hacer los mismos cambios en varios sitios distintos.

2.2 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 van a formar 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 cada una de ellas y se han conectado al sistema. A continuación, vamos a nombrar y a explicar de forma rápida cómo se compone esta arquitectura y cuáles son cada uno de sus componentes.

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

   La aplicación desarrollada en Flutter, que actúa como interfaz principal para los usuarios. Es una aplicación multiplataforma que puede ejecutarse en dispositivos móviles, escritorio o navegadores. Este subsistema ha sido desarrollado desde cero en el proyecto.

2. **Backend - Odoo (subsistema independiente):**

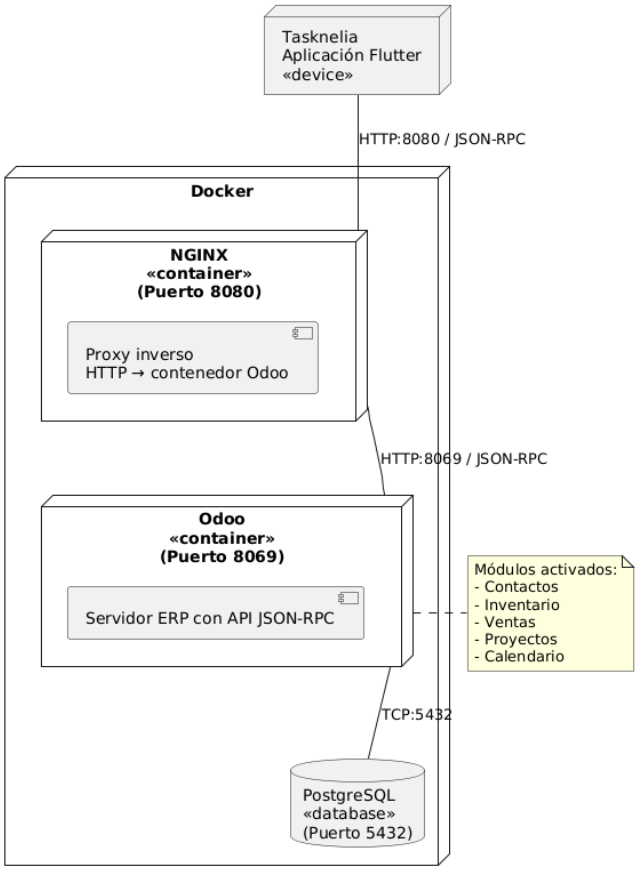
   Es un sistema de gestión empresarial, Odoo se encarga de la lógica de negocio y del acceso a los datos. No ha sido programado desde cero, pero se ha desplegado y configurado para activar los módulos necesarios que se debían de usar en la aplicación.

3. Base de datos - PostgreSQL (componente interno):

Odoo utiliza PostgreSQL como sistema de gestión de bases de datos para almacenar toda la información importante: tareas, pedidos, usuarios, etc... No se accede directamente desde la aplicación a esta base de datos, sino que se accede a través de Odoo.

4. Capa de comunicación (JSON-RPC sobre HTTP):

Toda la comunicación entre la aplicación Flutter y el backend de Odoo se realiza mediante el protocolo JSON-RPC, que es uno de los métodos que ofrece Odoo para consultar, crear o modificar datos. Este tipo de comunicación se realiza a través de llamadas HTTP en formato JSON y para facilitar esta conexión, se ha configurado un servidor NGINX como proxy inverso, el cual redirige las peticiones de la aplicación al backend de Odoo. Aunque no se ha llegado a configurar un certificado SSL válido para el uso de HTTPS, se ha preparado toda la estructura para que esta conexión pueda realizarse de forma segura en futuras versiones.



Sobre el despliegue del sistema, Tasknelia se ha diseñado para que su despliegue y ejecución sea sencillo y para ello, se ha utilizado Docker como herramienta principal, permitiendo separar los distintos servicios que componen el sistema y ejecutándolos de forma independiente pero conectada.

Componentes del sistema desplegados:

- **Cliente (Aplicación Tasknelia):**
  - Tecnología usada: Flutter.
  - Se puede ejecutar como app en un móvil o abrir desde el navegador.
  - La conexión con el backend, se realiza mediante peticiones JSON-RPC a través de HTTP usando un proxy intermedio configurado con NGINX.

- **NGINX (Proxy Inverso):**
  - Redirige las peticiones externas que llegan desde Flutter al backend de Odoo.
  - Se ejecuta en un contenedor Docker.
  - Puerto utilizado: 8080 (redirige internamente al puerto 8069 de Odoo).

- **Odoo (Backend ERP):**
  - Gestiona la lógica de negocio y proporciona acceso a los datos.
  - Módulos activados: Contactos, Ventas, Inventario, Proyecto y Calendario.
  - Odoo corre dentro de un contenedor Docker personalizado y configurado con una base de datos propia.

- **PostgreSQL (Base de datos):**
  - Almacena todos los datos que utiliza Odoo.
  - Esta dentro de un contenedor Docker.
  - Su conexión solo es accesible de forma interna por Odoo.

- **Comunicación entre contenedores:**
    - Todos los contenedores de lo mencionado anteriormente (Odoo, PostgreSQL y NGINX) están conectados en una misma red Docker privada. Esto facilita que se comuniquen entre sí de forma segura.

Gracias a esta estructura, el sistema puede desplegarse en distintos equipos sin necesidad de configurar todo desde cero. Solo se necesita Docker y ejecutar el archivo docker-compose.yml, que ya contiene todo lo necesario para levantar el entorno.

2.3 Base de datos

Este 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 como se comentó antes en otro apartado, 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. Así que, el diseño de la base de datos en sí, no forma directamente parte del trabajo que se está desarrollando, pero sí que es importante saber qué datos utiliza la aplicación y cómo se accede a ellos.

2.3.1 Módulos de Odoo utilizados

Para que la aplicación funcione correctamente y tenga acceso a los datos que se van a necesitar, se han activado varios módulos dentro de Odoo. Estos módulos son los encargados de crear y mantener las tablas que el sistema usará. Los principales módulos que van a ser utilizados teniendo en cuenta el uso que se le dará a nuestra aplicación son los siguientes:

- **Contactos**: Para la gestión de los clientes, proveedores y empleados.
- **Inventario**: Para la gestión de los productos, categorías y cantidades.
- **Ventas**: Para registrar los pedidos de los clientes y su estado.
- **Proyectos**: Para crear las tareas que se asignan a los trabajadores.

- **Calendario**: Para la planificación y gestión de reuniones y fechas.

Cada uno de estos módulos se activa y mantiene una serie de tablas en PostgreSQL que son utilizadas por la aplicación. Aunque no se ha diseñado directamente esta base de datos, se han analizado estas tablas para entender qué campos son los que se usarían en la app.

2.4 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 ido activado. Estos son ejemplos de cómo la aplicación interactúa con la base de datos:

- Cuando un usuario consulta sus tareas asignadas, la app accede a la información de los módulos de proyectos y empleados.
- Al crear una nueva tarea, se registra automáticamente en el sistema de Odoo, que a su vez la almacena en su base de datos PostgreSQL.
- Si se actualiza el estado de una tarea (de "pendiente" a "completada"), 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, nuestro diseño de base de datos en este proyecto se basa más en saber qué módulos se van a necesitar, qué entidades se van a utilizar y cómo se relacionan entre ellas dentro de Odoo.

2.5 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.

4. **Visualización y navegación:**
   - Asegurarse de que los datos (usuarios, tareas, reuniones...) 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 asegurarse de 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.
   - Comprobar 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.
   - Ver el tiempo que tarda en cargar una lista de tareas larga.

8. **Pruebas de usabilidad:**

   - Confirmar que la interfaz es clara y fácil de usar para un usuario sin conocimientos.
   - Comprobar que las acciones se puedan realizar en pocos pasos.

9. **Pruebas de seguridad y control:**

- Confirmar que los datos de los usuarios no se mezclen entre perfiles distintos.
- Comprobar que solo los usuarios autorizados puedan realizar ciertas acciones (crear tareas, asignar, eliminar...).

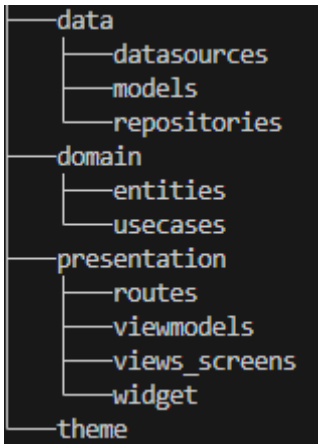
3. Implementación de la aplicación

La implementación de este proyecto ha seguido una planificación ordenada basada en el análisis y el diseño realizados anteriormente. 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 puntos importantes de la implementación: el entorno de desarrollo y la futura puesta en producción del sistema.

3.1 Entorno de desarrollo

El desarrollo de esta aplicación se ha realizado usando Flutter como framework principal y Dart como lenguaje de programación. El entorno de desarrollo ha sido Visual Studio Code, configurado con las extensiones necesarias para trabajar con Flutter, Dart, Git y GitHub, facilitando así el desarrollo y el control de versiones.

En este proyecto se ha seguido una estructura limpia, basada en buenas prácticas recomendadas para Flutter, como el uso de widgets reutilizables, separación por capas (vistas, lógica…) y uso de rutas organizadas. El código fuente se encuentra dentro de la carpeta **lib/**, siguiendo la siguiente estructura:



Para el control de versiones se ha llevado a cabo mediante GitHub, lo que ha permitido ir guardando los avances o volver atrás en el caso de que fuera necesario y se han llegado a desarrollar y probar el inicio de sesión, tareas, agenda...

Para el backend se ha usado Odoo, desplegado mediante Docker junto con una base de datos PostgreSQL. Esto se ha realizado utilizando un archivo docker-compose.yml, que hace más fácil el arranque y la configuración de los contenedores de forma automática, sin necesidad de instalar cada componente de forma manual.

El backend completo está compuesto por:

- **Odoo (ERP):** lógica de negocio y acceso a datos.
- **PostgreSQL:** base de datos interna de Odoo.
- **NGINX:** configurado como proxy inverso para gestionar la comunicación entre Flutter y Odoo y evitar errores CORS.

3.2 Implantación/Puesta en producción

La puesta en funcionamiento del sistema se realiza de forma separada para los dos desarrollos: la aplicación en Flutter y el backend con Odoo.

- **Backend (Odoo + PostgreSQL + NGINX):**

- **Docker Compose:** Todo el entorno backend se levanta con un único comando gracias al archivo docker-compose.yml, que contiene la configuración de los contenedores para Odoo, PostgreSQL y NGINX.

- **Red interna en Docker:** Los contenedores se comunican entre sí a través de una red Docker privada, lo que garantiza seguridad y aislamiento.

- **Proxy Inverso (NGINX):** Se encarga de redirigir las peticiones externas (desde Flutter) hacia Odoo, solucionando errores CORS y facilitando la comunicación en el entorno web.

- **Backup y recuperación:** Gracias a Docker, la creación de copias de seguridad de la base de datos y la configuración es sencilla y rápida.

- **Frontend (Aplicación Flutter):**

  - **Ejecución multiplataforma**: La aplicación puede ejecutarse en móvil, como aplicación de escritorio o desde el navegador web.

  - **Conexión al backend:** La aplicación se comunica con Odoo mediante peticiones JSON-RPC sobre HTTP, que pasan por el servidor NGINX.

  - **Despliegue flexible:** Al ser una aplicación Flutter, se puede compilar y desplegar en distintos entornos fácilmente, lo que facilita su uso tanto dentro de la red de una empresa como de forma externa si se configurará el dominio con DuckDNS.

- **Resumen del proceso de puesta en producción:**
  - Clonar el repositorio desde GitHub.
  - Ejecutar docker-compose up para desplegar el backend.

- Ejecutar la aplicación Flutter en el entorno deseado (móvil, web, escritorio).

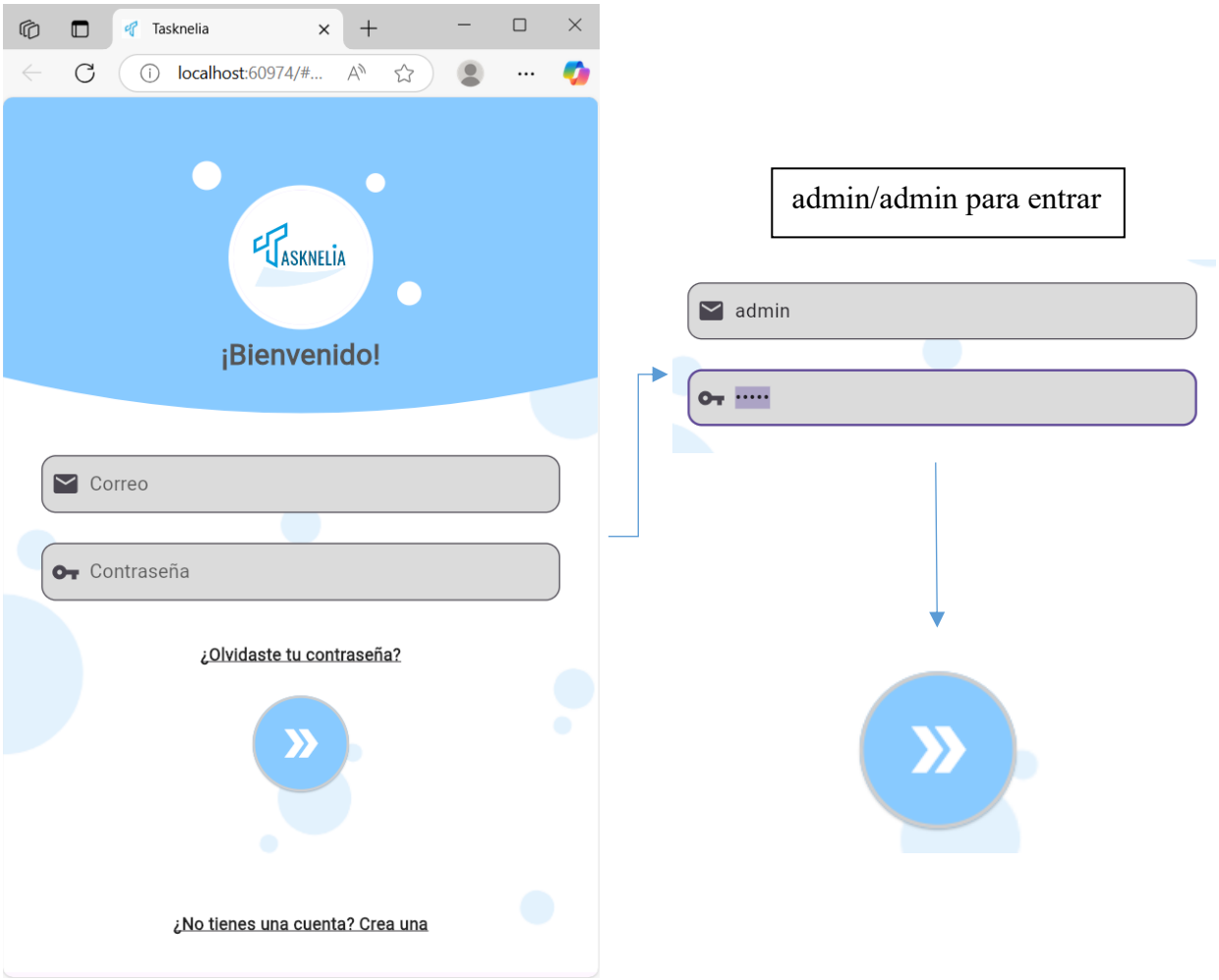
4. Capturas de la ejecución

4.1 Funcionalidad

A continuación, se muestran capturas de ejecución de la funcionalidad de la aplicación, como ejemplo se han puesto capturas de ejecución en Windows y en Chrome

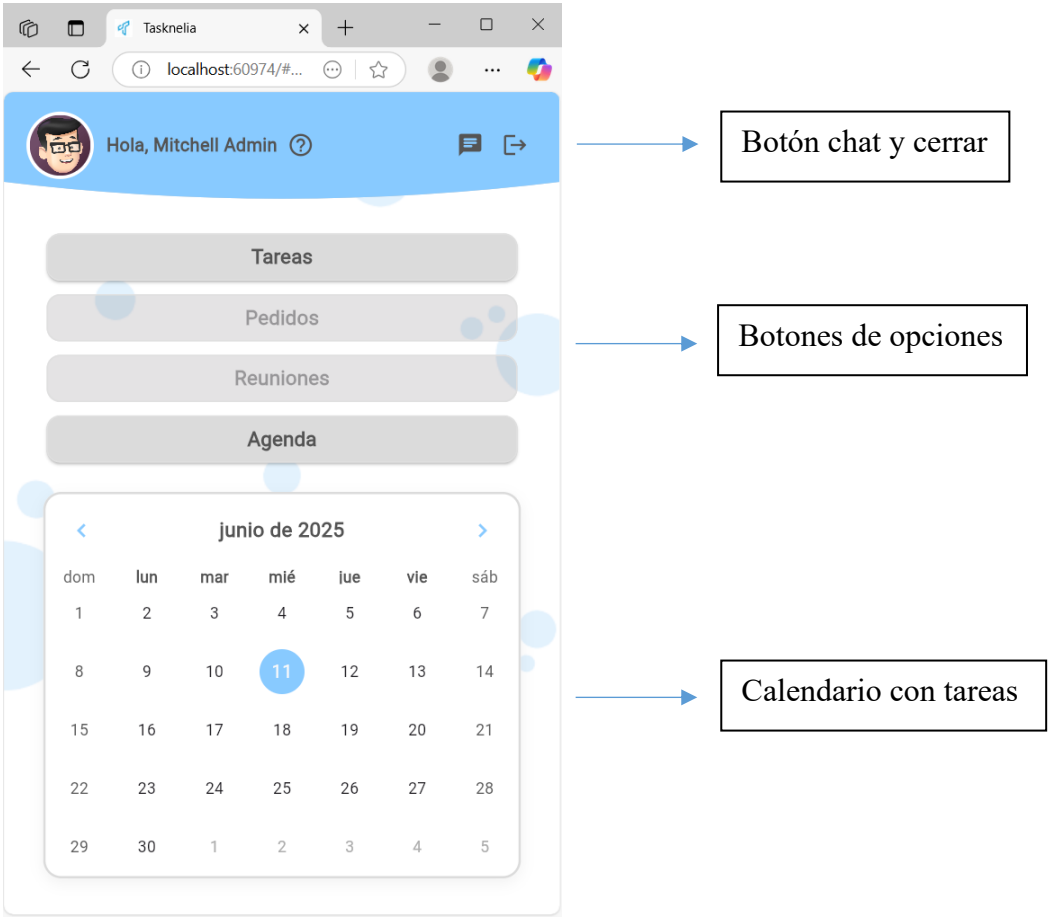
4.1.1 Inicio de sesión

Esta es la pantalla de inicio de sesión ejecutada en Chrome, donde rellenamos los datos y le damos al botón



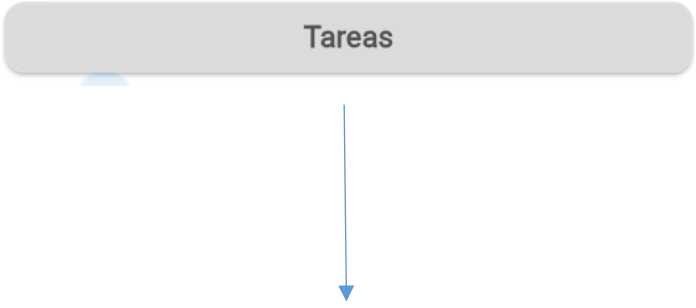
4.1.2 Pantalla principal

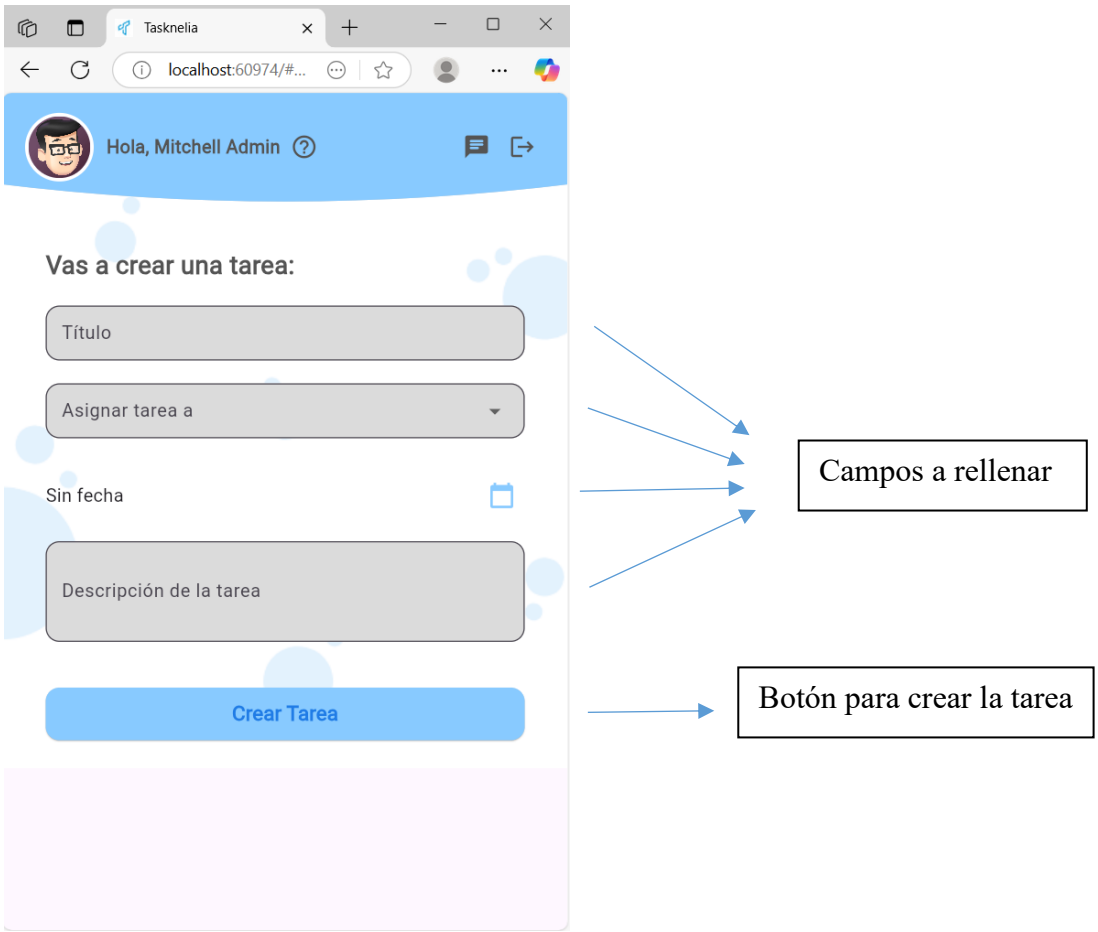
Esta es la pantalla principal ejecutada en Windows, donde vemos los datos del usuario, el botón de cerrar sesión, el botón de chat, el calendario y las opciones de las cuales dos de ellas no están disponibles.



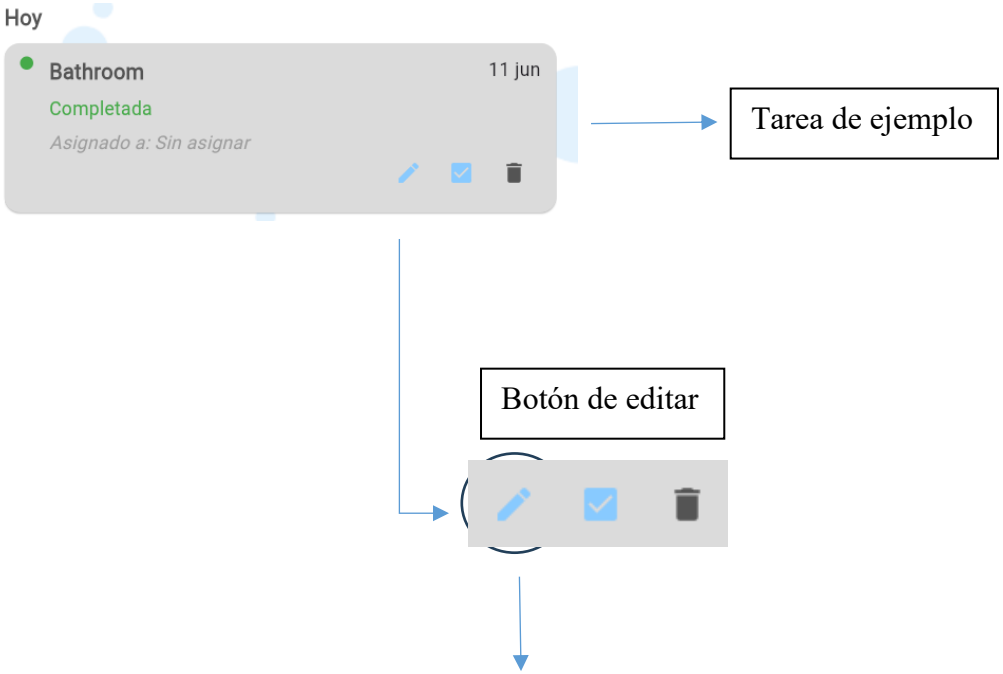
4.1.3 Creación de tareas

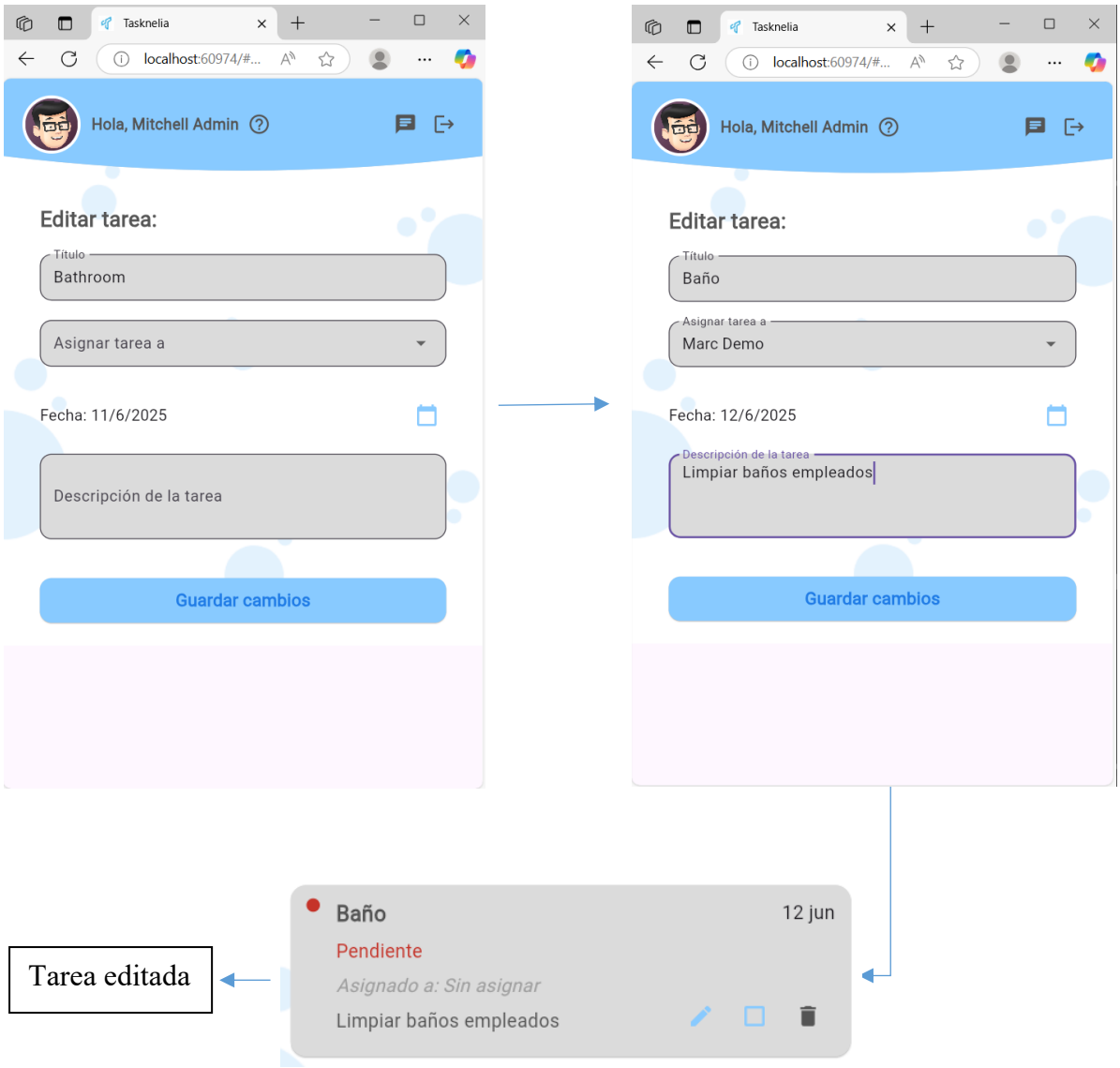
Esta es la pantalla de creación de tarea en Chrome, como se puede ver tiene ciertos apartados para rellenar.





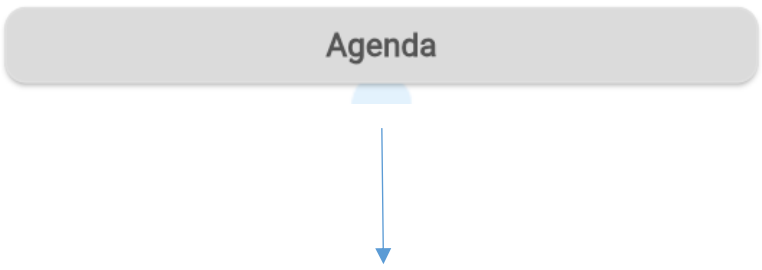
4.1.4 Editor de tareas

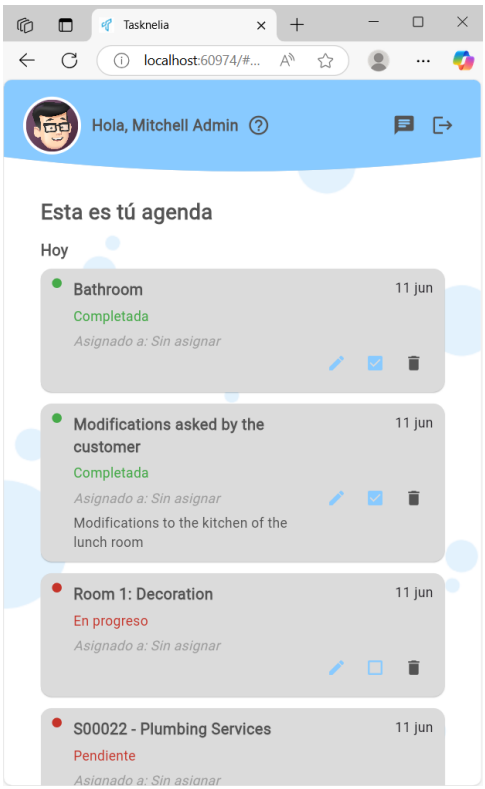




4.1.5 Agenda

Esta es la pantalla de agenda ejecutada en Windows donde podrás ver todas tus tareas/pedidos/reuniones y tú podrás borrarlas, editarlas o marcarlas como completadas.

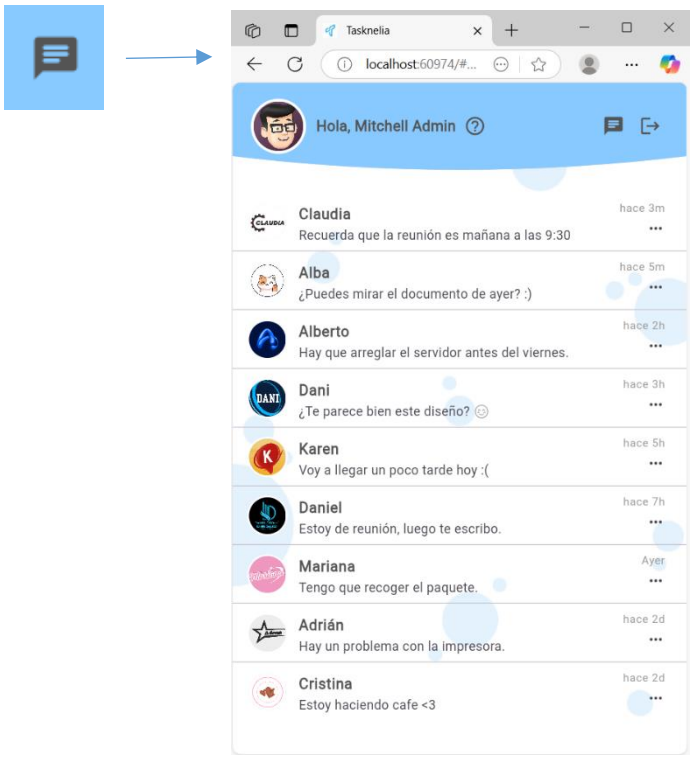




Aquí veras todas tus tareas

4.1.6 Chat

Esta es la pantalla de chat ejecutada en Chrome, no es funcional, solo decorativa.



La pantalla de chat

5. Herramientas usadas en el proyecto para su ejecución

Para el desarrollo de este sistema de gestión se han usado una serie de diferentes herramientas que permiten que la aplicación funcione correctamente y pueda cumplir con todos los objetivos que se han mencionado en los apartados anteriores. A continuación, se mencionarán cada una de estas herramientas con una pequeña explicación de su uso:

- **Flutter:**
  - Framework que se ha utilizado para el desarrollo de la interfaz del usuario (frontend). Permite crear aplicaciones multiplataforma desde un solo código.

- **Dart:**
  - Un lenguaje de programación usado por Flutter para poder implementar la lógica y el comportamiento de la aplicación.

- **Odoo:**
  - Es un ERP de código abierto que se ha utilizado como backend de nuestro sistema de gestión para poder 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:**
  - Una plataforma de contenedores usada para desplegar Odoo y su base de datos de forma aislada.

- **NGINX:**
  - Es un servidor web que se ha usado como proxy inverso para poder redirigir las peticiones externas al contenedor de Odoo, permitiendo así que la aplicación Flutter pueda comunicarse con Odoo de forma

indirecta, resolviendo así problemas como los errores de CORS y facilitando el acceso a través de un único puerto.

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

- **Visual Studio Code:** Un entorno de desarrollo utilizado para programar y organizar todo el código y archivos del proyecto.

- **Git y GitHub:** Para el control de versiones y almacenamiento en la nube que me ha permitido gestionar el progreso del proyecto de una forma más ordenada.

- **Canva:** Una herramienta online de diseño gráfico que ha sido usada para crear los bocetos de la interfaz de usuario y diagramas de navegación.

6. 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.

6.1 Problemas a la hora del desarrollo

6.1.1 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.

6.1.2 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 del uso de este proxy inverso 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) en una carpeta del proyecto llamada https no usable y preparado para su funcionamiento.

6.1.3 Problemas al iniciar la aplicación después de organizar el proyecto

En cierto punto del desarrollo se volvió a organizar todo el proyecto para mejorar su estructura, pero después de realizar esto e intentar iniciar la aplicación, Flutter dio un error que impedía el arranque del proyecto. Investigando se descubrió que este error era debido a que Flutter (en concreto CMake, en proyectos de escritorio) guardaba en su caché rutas absolutas internas de compilación, como el proyecto había sido reorganizado, las rutas guardadas ya no coincidían con la nueva ubicación real, y por eso fallaba la ejecución. Para solucionarlo simplemente se realizó lo siguiente:

- **Ejecutar el comando flutter clean**, que se encarga de borrar la carpeta de compilación y limpiar todo el caché del proyecto.

Después de hacer esto y volver a compilar, la aplicación se ejecutó correctamente.

6.1.4 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 no tenía correctamente configurados los dispositivos.

6.1.5 Problemas en la conexión de la aplicación y Odoo

Durante el desarrollo de la aplicación, uno de los problemas más grandes fue conseguir que la aplicación pudiera comunicarse correctamente con el backend de Odoo. Aunque en local todo parecía estar configurado correctamente, al intentar hacer peticiones desde la aplicación (sobre todo desde la versión web), aparecían errores relacionados con CORS.

Estos errores impedían que la aplicación accediera a los datos de Odoo, ya que el navegador bloqueaba las peticiones por no estar autorizadas entre orígenes distintos. Aunque esto es normal por temas de seguridad, para el proyecto era un gran problema ya que no permitía avanzar con las pruebas de conexión.

La solución a este problema fue investigar y **descubrir que NGINX por sí solo podía solucionar este problema**, es por ello que lo configuré sin el HTTPS. Gracias a esto, las peticiones ahora pasan primero por NGINX (usando el puerto 8080), que redirige las peticiones al puerto interno de Odoo, logrando que el navegador ya no bloquee las peticiones porque se hacen dentro del mismo dominio configurado, evitando así los errores de CORS.

Con esta rápida solución, la aplicación ahora puede acceder correctamente al backend sin errores y se ha logrado usar NGINX de todas maneras, aunque no como se esperaba.

6.1.5 Problemas en las pantallas, creación y muestra de tareas por Git

Hubo un problema de última hora con el proyecto, se realizó un guardado totalmente funcional del proyecto, el problema es que al continuar el día siguiente y realizar un guardado para guardar la configuración HTTPS junto con algunas fotos, se probó a ejecutar la aplicación de nuevo pero la creación y la muestra de tareas ya no iban, no sé sabe con exactitud cuál fue el motivo del cambio y desaparición de líneas en el código al hacer un guardado en Git, se intentó recuperar sin suerte. Por lo que se ha realizado otra vez parte de los diseños, se han cambiado iconos y se ha vuelto a poner en funcionamiento ciertas clases.

6.2 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 no solo han hecho que el proyecto sea algo más actual, sino que también sea más completo, entretenido e interesante. Las principales innovaciones (aunque algunas de ellas al final no terminaran por usarse en el proyecto) 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. No se tenía mucho conocimiento sobre Docker, pero se logró aprender a usarlo fácilmente.

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

Se intentó configurar una conexión segura mediante HTTPS utilizando un proxy inverso con NGINX, certificados SSL y la herramienta Certbot. Aunque se realizaron todos los pasos, el sistema no pudo validar el certificado debido a problemas con los puertos y restricciones de red, se dejó preparada la configuración en otra carpeta específica del proyecto. Aun así, esta parte del proyecto me sirvió para entender cómo funciona realmente la configuración de HTTPS en aplicaciones reales y cómo usar NGINX como proxy inverso para mejorar la seguridad o arreglar problemas CORS.

- **Uso de dominio dinámico con DuckDNS:**

Durante el desarrollo del proyecto se intentó utilizar un dominio dinámico de DuckDNS para poder acceder a la aplicación desde fuera de la red local sin la necesidad tener un dominio de pago. DuckDNS es un servicio gratuito de DNS dinámico, aunque la conexión segura mediante HTTPS no llegó a completarse correctamente, sí se consiguió crear el subdominio. Esta configuración servirá a futuro en otra versión para poder realizar el posible despliegue accesible desde fuera.

7. Líneas de trabajo futuro

7.1 Requisitos pendientes e ideas pensadas para las siguientes versiones

En esta primera versión de la aplicación solo se pondrá en funcionamiento los apartados de tarea y agenda, 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 logrando realizar la conexión HTTPS.
- Que la aplicación sea muy escalable y no dé errores.
- Tener una mejor accesibilidad para personas con discapacidades.
- Poder tener ajustes como el tema oscuro, cambiar el idioma o el tamaño de la letra.
- Añadir un apartado de configuración de cuenta de usuario completo.
- 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 administrador sobre la eficiencia de sus trabajadores o de sí mismo.
- Tener notificaciones en tiempo real para avisar de nuevas tareas/reuniones/pedidos e incluso chats entre usuarios de la misma empresa para el paso de información y la mejora de la privacidad.
- Mejorar el seguimiento de las tareas y sus estados.
- Mejorar los roles de los usuarios y todos sus permisos.
- Mejorar el diseño de la aplicación y la reutilización.
- Y de final, hacer que el manual de usuario sea una página web, donde los usuarios podrán leerlo junto con nuevos apartados como:
    - Noticias sobre nuevas actualizaciones o parches que vayan a salir.
    - Un apartado para poder dejar comentarios y así ver sus opiniones sobre la aplicación y que aspectos se podrían mejorar e incluso saber de nuevos errores que hayan descubierto ellos.
    - De esta manera se formaría una pequeña comunidad, mejorando así la cercanía con los usuarios, el manual de usuario y la aplicación.

8. Conclusiones

Como conclusión final quiero decir que este proyecto ha sido un gran reto para mí desde el principio, tanto a nivel técnico como personal, ya que al empezar no tenía conocimientos en Flutter ni había trabajado con herramientas como Docker o NGINX y yo suelo ser alguien que va a su propio ritmo a la hora de investigar y aprender sobre nuevos temas o a la hora de programar. Aun así, he sido capaz de aprender poco a poco y lograr construir una aplicación funcional con la idea que tenía en mente con todas estas herramientas.

Algunas cosas me han costado bastante, como intentar configurar la conexión HTTPS que al final no pude lograr o tener que estructurar el código siguiendo una arquitectura limpia, ya que es algo que me ha costado al principio acostumbrarme debido a que tenía que trabajar con muchos archivos, y sin mencionar lo duro que ha sido conectar todo sin errores. Sin embargo, aunque surgiera problema tras problema, los he ido solucionando por mi cuenta, investigando mucho y haciendo muchos cambios o pruebas, así que estoy algo orgullosa de haber podido arreglarlos (aunque sienta que he estado más solucionando errores del código o de la conexión que programando la aplicación).

Admito que estoy contenta y a la vez no con el resultado final, la aplicación funciona, tiene su conexión, es intuitiva, tiene un diseño agradable, está organizada por dentro…Pero siento que, si no hubiera sido por tantos fallos y errores que para mí no tenían sentido, haberme centrado tanto en intentar lograr que funcionara la conexión HTTPS o en el documento, podría haber dado más de la aplicación y hacerla tan funcional como he estado comentando en los apartados.

Aun así, sé que después de esto puedo seguir ampliando y mejorando esta aplicación, algo que me anima ya que ha sido un proyecto que me ha gustado hacer y me gustaría poder seguir mejorándolo personalmente. Además, me ha servido para entender mucho mejor cómo se conectan los distintos sistemas en un proyecto real: frontend, backend, base de datos, red, etc. También algo que agradezco y me ha ayudado a seguir es toda la ayuda que he recibido tanto de profesores como de la gente de la empresa en que he estado realizando las practicas ya que me han enseñado un montón y gracias a ellos he podido usar ciertas herramientas de este proyecto con un poco más de rapidez.

Creo que este proyecto puede tener futuro ya que no siento que sea una mala idea, con tiempo y trabajo podría evolucionar hasta convertirse en una herramienta útil de verdad para alguna pequeña empresa o al menos servirme de inspiración para crear una versión totalmente diferente y mejorada que algún día pueda publicar. También sé que la realización de este proyecto me puede abrir puertas a futuros proyectos similares donde pueda aplicar todo lo que he aprendido (tanto las herramientas usadas, las ideas como la resolución de problemas).

En resumen, este proyecto me ha ayudado a crecer mucho como estudiante y posible futura desarrolladora. He pasado de no tener experiencia con estas herramientas a saber cómo montar un entorno casi completo de desarrollo, desplegarlo con Docker, comunicarlo con NGINX e intentar que tuviera seguridad, y crear una app en Flutter desde cero. De este proyecto me quedo tanto con las cosas buenas como las malas y, aunque aún me queda mucho por aprender, ahora me siento mucho más preparada para poder afrontar proyectos similares o incluso más complicados en el futuro.

9. 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

Khomula, T. (22 de 10 de 2024). *Clean Architecture in Flutter*. Obtenido de Medium: https://medium.com/flutter-community/clean-architecture-fcd46c98e089

Overflow, S. (s.f.). *Stack Overflow Questions*. Obtenido de Stack Overflow: https://stackoverflow.com/questions

Packages, D. (s.f.). *Pub.dev*. Obtenido de Pub.dev: https://pub.dev/

Poudel, M. (14 de Junio de 2020). *What is copyWith and how can I use it in Flutter and what are some of its use-cases?* Obtenido de Stack Overflow: https://stackoverflow.com/questions/62372580/what-is-copywith-and-how-can-i-use-it-in-flutter-and-what-are-some-of-its-use-c

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

Syncfusion. (s.f.). *syncfusion_flutter_pdfviewer*. Obtenido de Pub.dev: https://pub.dev/packages/syncfusion_flutter_pdfviewer

Zagliz, Y. (30 de Septiembre de 2024). *Changing Flutter Date Picker Locale*. Obtenido de Stack Overflow: https://stackoverflow.com/questions/61529343/how-to-change-language-of-show-date-picker-in-flutter

10. Anexos

10.1 GitHub

Repositorio de GitHub con el proyecto subido: LauraSA29/Proyecto_TFG_Laura

(Poudel, 2020) (Zagliz, 2024) (Syncfusion, s.f.) (Packages, s.f.) (Overflow, s.f.)

10.2 Docker-compose

Para poder ver el Docker-compose usado para este proyecto:
Proyecto_TFG_Laura/backend/docker_laura at main · LauraSA29/Proyecto_TFG_Laura

10.3 Manual de usuario

El manual de usuario de esta aplicación se encuentra en un documento aparte, esta es la ruta: Proyecto_TFG_Laura\docs\Manual usuario Tasknelia