TFC DAM

UNIR FORMACION PROFESIONAL

TASKSPHERE

COMPANY

TUTOR:
RAQUÉL CERDÁ

INTEGRANTES DEL GRUPO:
ALFONSO LOBATO DE LA SIERRA
AZAHARA FONSECA ESCUDERO
ALEJANDRO MARTIN FERNANDEZ PRIETO

INTRODUCCIÓN:

TaskSphere es una aplicación móvil innovadora diseñada para revolucionar la forma en que los comercios gestionan sus tareas diarias, equipos, horarios y vacaciones. Combina una interfaz intuitiva con poderosas herramientas de gestión, convirtiéndose en una solución integral para negocios que buscan optimizar sus operaciones y mejorar la productividad de sus empleados.

En el actual entorno empresarial, caracterizado por la rápida evolución tecnológica y la creciente competencia, los pequeños y medianos comercios enfrentan múltiples desafíos. Uno de los más importantes es la falta de acceso a herramientas tecnológicas adaptadas a sus necesidades, especialmente en lo que respecta a la gestión de empleados y la asignación y seguimiento de tareas.

Las grandes corporaciones suelen contar con sistemas integrados y aplicaciones móviles personalizadas que les permiten monitorear la productividad de sus equipos y gestionar diversas labores de manera eficiente, sin embargo, muchas PyMEs se ven obligadas a depender de métodos tradicionales que pueden ser ineficientes y propensos a error.

Esta brecha tecnológica no solo impide que las PyMEs maximicen su productividad y eficiencia, sino que también limita su capacidad para competir en igualdad de condiciones en el mercado.

Este escenario subraya la necesidad de desarrollar soluciones tecnológicas accesibles, personalizables y escalables que se adapten a las realidades y limitaciones específicas de las PyMEs. Al hacerlo, no solo se estaría potenciando la capacidad de estas empresas para gestionar de manera más efectiva a sus equipos y recursos, sino que también se estaría contribuyendo a igualar su nivel de competitividad en el mundo empresarial, donde la eficiencia operativa y la capacidad de adaptación rápida a los cambios del mercado son claves para el éxito sostenido.

OBJETIVOS:

Algunas de las principales funcionalidades de esta aplicación serán:

- Generación y delegación de tareas: permitirá a los gerentes crear tareas y asignarlas a uno o varios trabajadores.
- Establecimiento de niveles de prioridad: los usuarios podrán definir la importancia de cada tarea para una gestión más eficiente del tiempo.
- Supervisión y seguimiento detallado: los gerentes podrán monitorear el progreso de las actividades de manera minuciosa.
- Visión global de la productividad: la aplicación ofrecerá una perspectiva general de la productividad tanto a nivel empresarial como individual de cada trabajador.
- Administración de horarios y turnos de los trabajadores: facilitará la gestión de los horarios laborales y los turnos del personal.
- Gestión de vacaciones y horas extras: los gerentes podrán administrar las solicitudes de vacaciones y las horas extras del equipo de trabajo de manera eficiente.
- Envío de notificaciones y comunicados: se implementará una función para enviar comunicados importantes y notificaciones a todo el personal de forma rápida y sencilla.

INDICE:

MÓDULOS FORMATIVOS APLICADOS EN EL TRABAJO.

A continuación, se detallan las asignaturas del ciclo en las que nos hemos apoyado para realizar el proyecto:

ENTORNOS DE DESARROLLO

Hemos hecho uso de **Git y GitHub** ya que son esenciales para la colaboración en equipo y el control de versiones.

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

Trabajaremos con **Android** sobre el lenguaje de programación java como cliente ya que toda nuestra aplicación será para Android por ser el sistema operativo móvil más utilizado a nivel mundial. Además, en cuanto a su desarrollo, esta plataforma ofrece varias ventajas:

- Acceso a un **Mercado Amplio**: al desarrollar para Android, se tiene acceso a una amplia base de usuarios potenciales en todo el mundo.
- **Flexibilidad y Personalización**: ofrece una gran flexibilidad en términos de personalización de la interfaz de usuario y la integración con las funciones del dispositivo.
- **Soporte de Google**: al ser desarrollado por Google, se beneficia de un fuerte soporte, incluyendo herramientas de desarrollo como Android Studio, documentación extensa y nuevas tecnologías para facilitar el desarrollo.

DESARROLLO DE INTERFACES

En la aplicación se han implementados Los principios de "**Experiencia de Usuario (UX) e Interfaz de Usuario (UI)"** vistos en la asignatura ya que son cruciales para el éxito de cualquier aplicación móvil.

DEVOPS

Combinación de los términos "development" y "operations". Es la unión de personas, procesos y tecnología, para ofrecer valor a los clientes de forma constante. Los equipos que adoptan la cultura, las prácticas y las herramientas de DevOps mejoran el rendimiento y crean productos de mayor calidad en menor tiempo, aumentando la satisfacción de los clientes.

Se han desplegado alguna de las funcionalidades de la aplicación en forma de contenedores en Docker.

HERRAMIENTAS/LENGUAJES UTILIZADOS:

En cuanto a tecnologías y herramientas, se han empleado las siguientes:

JAVA ENTERPRISE EDITION

Se ha empleado el **lenguaje de programación Java** por ser uno de los más utilizados para el desarrollo de aplicaciones Android.

FIREBASE

Proporciona un backend listo para usar, facilitando servicios esenciales como autenticación, base de datos en tiempo real, y almacenamiento, lo que acelera el desarrollo y permite a los desarrolladores concentrarse en mejorar la experiencia del usuario. Se ha utilizado sobre todo para la gestión de base de datos ya que la mayoría de los servicios que ofrece la aplicación se persisten en Firebase Firestore. Tambien todas las funcionalidades de login y registro vienen de la mano de firebase.

GITHUB

Mejora la colaboración en equipo a través de la gestión de código fuente y control de versiones, facilitando la revisión de código, el seguimiento de problemas y la automatización de flujos de trabajo, lo que resulta en un proceso de desarrollo más organizado y eficiente. Durante todo el proyecto se han ido subiendo al repositorio remoto los diferente commits y fases del proceso de la app en cuanto a manejo de código.

ANDROID STUDIO

Como el IDE oficial para el desarrollo de Android, ofrece herramientas específicas para esta plataforma, incluyendo emuladores, depuradores, y un editor de UI, lo que simplifica la creación, prueba y depuración de aplicaciones. Se ha utilizado el IDE de Android Studio para toda la parte del cliente de Android.

XML (EXTENSIBLE MARKUP LANGUAGE)

Lenguaje de marcado similar a HTML. En aplicaciones Android, se utiliza normalmente para definir recursos como cadenas, colores, estilos y layouts. Los archivos XML son fundamentales en la construcción de la UI, ya que en ellos se definen la estructura y apariencia de las pantallas que componen la aplicación.

RECYCLERVIEW

Herramienta fundamental para el desarrollo de interfaces en Android Studio. Permite mostrar grandes conjuntos de datos de forma eficiente reutilizando vistas e implementar diferentes disposiciones de elementos y animaciones personalizadas.

ADAPTER

Actúa como un puente entre los datos y la UI en un RecyclerView. Crea nuevas vistas para los elementos de datos y enlaza dichos datos con las vistas existentes.

WEBSOCKETS (JAVA)

Permite construir aplicaciones interactivas y responsivas con comunicación bidireccional en tiempo real, se ha optado por un sistema de websockets para realizar la funcionalidad de chat de todos los empleados de la empresa.

SPRINGBOOT

Framework de Java basado en Spring que se utiliza para crear aplicaciones independientes y de producción de manera rápida y sencilla. Muy utilizado en aplicaciones empresariales debido a su capacidad para reducir el tiempo de desarrollo y simplificar la configuración.

FIREBASE STORAGE

Servicio de almacenamiento en la nube proporcionado por Firebase, Se utiliza principalmente para almacenar y servir contenido generado por los usuarios, como fotos, vídeos y otros tipos de archivos.

POSTGRESQL

Sistema de gestión de bases de datos relacional y objeto-relacional de código abierto. Se caracteriza por su robustez, flexibilidad y conformidad con los estándares SQL.

JSON

(JavaScript Object Notation) formato ligero de intercambio de datos. Es fácil de leer y escribir para los humanos y fácil de parsear y generar para las máquinas. Es un subconjunto de la notación de objetos de JavaScript, pero es independiente del lenguaje, por ello es ampliamente utilizado en programación web y en la comunicación entre aplicaciones.

FRAGMENT

Parte modular de una actividad para construir interfaces flexibles. Permite combinar múltiples fragmentos en una sola actividad o incluso reutilizar fragmentos en otras actividades. De esta manera, cada fragmento tendrá su propio ciclo de vida, recibirá sus propios eventos de entrada y se podrá agregar o eliminar mientras la activity de acogida esté en marcha.

La integración de estas tecnologías proporciona una base sólida para el desarrollo de aplicaciones móviles, optimizando tanto el proceso de desarrollo, como la calidad del producto final.

COMPONENTES DEL EQUIPO Y SU APORTACIÓN:

AZAHARA FONSECA ESCUDERO: ha contribuido al proyecto, enfocándose en el diseño y desarrollo de las pantallas de inicio de sesión, creación de nuevas cuentas de usuario y gestión de comunicados, creando una interfaz de usuario intuitiva y accesible que permite a los usuarios acceder de manera sencilla y segura a la aplicación e implementando una plataforma eficiente para la gestión de comunicados, que permite la creación, edición y eliminación de los mismos, asegurando una difusión de información clara y controlada dentro de la organización, mejorando la comunicación interna y la eficiencia operativa.

ALFONSO LOBATO DE LA SIERRA: ha contribuido al proyecto, enfocándose en el diseño y desarrollo de múltiples componentes clave. Ha desarrollado el apartado de realizar fichajes y la visualización de los mismos, permitiendo a los usuarios registrar y consultar sus horas de manera sencilla y precisa. También ha creado la pantalla para ver a los empleados del equipo y gestionar sus roles, además de diseñar y desarrollar la pantalla principal.

Ha implementado un sistema de gestión de tareas tanto para administradores como para empleados, que permite la creación y asignación de tareas. En la pantalla de perfil, ha mejorado la funcionalidad para subir fotos de perfil al servidor utilizando Firestore Storage. Ha desarrollado la pantalla de notificaciones y gestionado su funcionamiento mediante el servicio de Firestore Cloud. Además, ha integrado un microservicio en Spring Boot que proporciona funcionalidad al chat de empleados mediante WebSockets, mejorando así la comunicación interna dentro de la organización.

ALEJANDRO MARTÍN FERNÁNDEZ PRIETO: Ha contribuido al proyecto diseñando el logo de la aplicación, creando la pantalla de apertura de la aplicación (splash screen) y desarrollando tanto la interfaz gráfica de la actividad del calendario como su funcionalidad. Asimismo, ha sido responsable del desarrollo de la pantalla de gestión de las solicitudes de vacaciones. Además, ha implementado la conexión entre la aplicación y la base de datos para el almacenamiento de los datos de los nuevos usuarios, así como para las solicitudes de días de vacaciones.

FASES DEL PROYECTO:

PLANIFICACIÓN:

- **MARZO:**

  Durante este mes se puso el foco en la fase del anteproyecto, donde se establecieron los fundamentos de la iniciativa.

  La propuesta consiste en el desarrollo de una aplicación móvil diseñada para simplificar la administración de tareas y la gestión de personal en pequeñas y medianas empresas (PyMEs), con la particularidad de ofrecerla de manera gratuita, pues reconocimos la existencia de aplicaciones similares en el mercado, pero observamos que están principalmente dirigidas a empresas de gran envergadura y además suelen tener costos significativos.

- **ABRIL:**

  Durante este mes, se llevaron a cabo múltiples reuniones para asignar responsabilidades entre los miembros del equipo y coordinar el desarrollo de diferentes partes del proyecto.

  Los primeros días, el equipo se centró en el diseño de la aplicación y en la creación de las diferentes pantallas de las que dispondría. Una vez configuradas, se procedió a dotarlas de la funcionalidad adecuada.

  Estas reuniones también sirvieron para solucionar dudas que iban surgiendo, así como para revisar y comentar los progresos realizados hasta el momento.

  El objetivo principal fue mantener una comunicación fluida y efectiva entre todos los miembros del equipo para garantizar un avance coherente y exitoso del proyecto.

  En los días restantes, se definieron las nuevas funcionalidades que se deseaban agregar a la aplicación y se retocaron pequeños detalles de aquellas que ya estaban previamente establecidas.

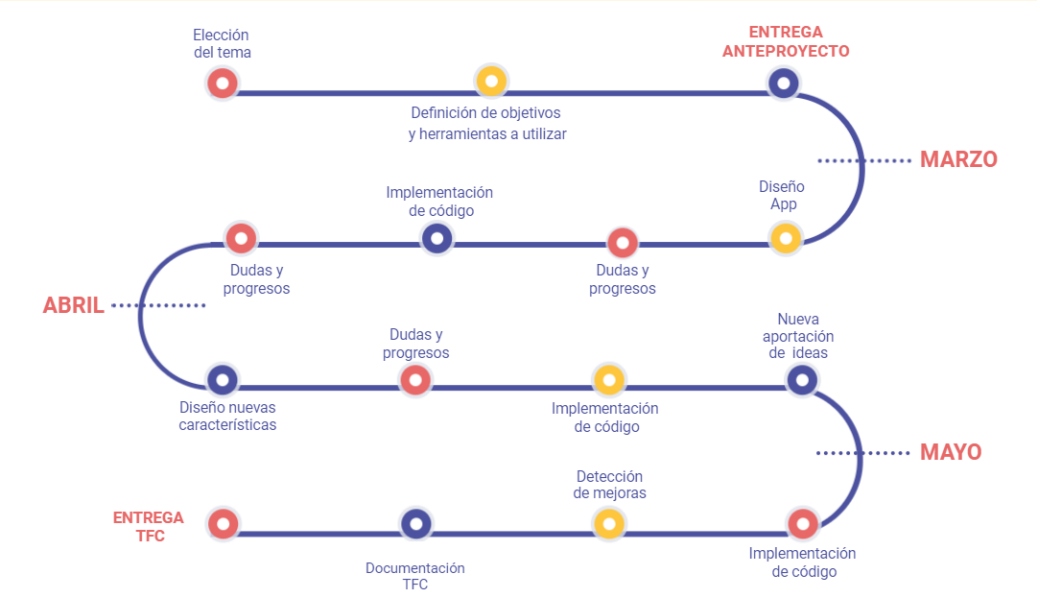
  A partir de esta definición, el equipo se dedicó al diseño y desarrollo de las nuevas características, asegurando su correcta implementación y funcionalidad en la aplicación.

- **MAYO:**

A lo largo de este mes, surgió alguna idea innovadora que no se había contemplado inicialmente y que además podría hacer la aplicación más atractiva para los futuros usuarios, por lo que se procedió a su desarrollo.

Se identificaron áreas de mejora a medida que se probaba la aplicación, lo que llevó a la modificación de ciertos aspectos para optimizar la experiencia del usuario.

Por último, se completó la redacción de la memoria del proyecto, documentando todo el proceso de desarrollo, desde la concepción de la idea, hasta las últimas modificaciones realizadas, así como los desafíos enfrentados y las lecciones aprendidas durante el proceso.



MODELO DE DATOS UTILIZADO:

En el presente apartado se describe la estructura de datos utilizada en Firebase Firestore para la aplicación. Firestore es una base de datos NoSQL en tiempo real proporcionada por Google, diseñada para almacenar y sincronizar datos entre usuarios y dispositivos a escala global.

ORGANIZACIÓN GENERAL DE LA BASE DE DATOS

La base de datos se organiza en colecciones y documentos, donde cada colección puede contener múltiples documentos, y cada documento puede tener varios campos y subcolecciones. A continuación, se detalla la estructura específica implementada para la aplicación.

COLECCIÓN: `COMUNICADOS`

La colección `Comunicados` almacena mensajes enviados por los usuarios. Cada documento en esta colección representa un comunicado individual y tiene los siguientes campos:

- id_usuario (String): Identificador único del usuario que creó el comunicado.
- fecha_creacion (Timestamp): Fecha y hora en que se creó el comunicado.
- descripcion (String): Descripción detallada del comunicado.
- titulo (String): Título del comunicado.

COLECCIÓN: `ROLES`

La colección `roles` contiene información sobre los diferentes roles de usuario en la aplicación. Cada documento representa un rol específico y tiene el siguiente campo:

- rolName (String): Nombre del rol (por ejemplo, "Empleado", "RRHH", "Programador", "Administrador").

COLECCIÓN: `SOLICITUDES`

La colección `solicitudes` gestiona las solicitudes para los días de vacaciones de los usuarios. Cada documento representa una solicitud individual y contiene los siguientes campos:

- fecha (String): Fecha de la solicitud en formato "DD/MM/YYYY".
- usuario (String): Identificador único del usuario que realizó la solicitud.
- estado (String): Estado actual de la solicitud (por ejemplo, "aprobada", "pendiente").

COLECCIÓN: `TAREAS`

La colección `tareas` administra las tareas asignadas a los usuarios. Cada documento representa una tarea individual y contiene los siguientes campos:

- asignadaA (String): Identificador único del usuario al que se le asignó la tarea.
- nombreTarea (String): Nombre de la tarea.
- fechaCreacion (Timestamp): Fecha y hora en que se creó la tarea.
- descripcionTarea (String): Descripción detallada de la tarea.
- fechaInicio (String, nullable): Fecha y hora de inicio de la tarea en formato "yyyy-MM-dd HH:mm:ss" (puede ser nulo).
- fechaFinalizacion (String, nullable): Fecha y hora de finalización de la tarea en formato "yyyy-MM-dd HH:mm:ss" (puede ser nulo).

COLECCIÓN: `USERS`

La colección `users` almacena información detallada de cada usuario registrado en la aplicación. Cada documento representa un usuario individual y contiene los siguientes campos:

- apellidos (String): Apellidos del usuario.
- fechaNacimiento (String): Fecha de nacimiento del usuario en formato "dd/MM/yyyy".
- ciudad (String): Ciudad de residencia del usuario.
- direccion (String): Dirección del usuario.
- telefono (String): Número de teléfono del usuario.

- nombre (String): Nombre del usuario.
- email (String): Correo electrónico del usuario.
- dni (String): Documento Nacional de Identidad del usuario.
- token (String): Token para notificaciones push.
- rollId (String): Identificador único del rol asignado al usuario.
- rol (String): Nombre del rol del usuario.
- vacaciones (Integer): Número de días de vacaciones disponibles para el usuario.
- biografia (String): Biografía del usuario.

Además, la colección `users` puede contener subcolecciones adicionales como `notificaciones` y `fichajes`.

SUBCOLECCIÓN: `NOTIFICACIONES`

Cada usuario puede tener una subcolección de notificaciones, donde cada documento representa una notificación individual con los siguientes campos:

- descripcion (String): Descripción de la notificación.
- categoria (String): Categoría de la notificación (por ejemplo, "Tareas", "Equipo").
- titulo (String): Título de la notificación.
- fechaCreacion (Timestamp): Fecha y hora en que se creó la notificación.

SUBCOLECCIÓN: `FICHAJES`

Cada usuario puede tener una subcolección de fichajes, donde cada documento representa los fichajes de un día específico con los siguientes campos:

- exists (Boolean): Indica si existen fichajes para la fecha.
- listafichajedia (Map): Subdocumentos que representan cada fichaje dentro del día, con los siguientes campos:
- fechaEmpiezo (Timestamp): Fecha y hora de inicio del fichaje.
- fechaFin (Timestamp, nullable): Fecha y hora de fin del fichaje.

RELACIÓN ENTRE COLECCIONES

Las colecciones en Firebase Firestore pueden relacionarse entre sí mediante referencias de identificadores únicos. A continuación, se describen las principales relaciones entre las colecciones en esta estructura de datos:

- **Usuarios y Comunicados:**

  La colección `Comunicados` contiene un campo `id_usuario` que es una referencia al identificador único del usuario en la colección `users`. Esto permite identificar qué usuario creó cada comunicado.

- **Usuarios y Solicitudes:**

  La colección `solicitudes` tiene un campo `usuario` que hace referencia al identificador único del usuario en la colección `users`. De esta manera, se puede rastrear qué solicitudes fueron realizadas por cada usuario.

- **Usuarios y Tareas:**

  La colección `tareas` contiene un campo `asignadaA`, que es una referencia al identificador único del usuario en la colección `users`. Esto permite asignar tareas a usuarios específicos y rastrear a quién se le asignó cada tarea.

- **Roles y Usuarios:**

  La colección `users` tiene campos `rolId` y `rol`, que hacen referencia a los documentos en la colección `roles`. El campo `rolId` es el identificador único del rol, mientras que `rol` es el nombre del rol, permitiendo así asignar y gestionar los roles de los usuarios.

- **Notificaciones y Usuarios:**

  Cada documento en la subcolección `notificaciones` dentro de `users` está asociado a un usuario específico, permitiendo enviar y almacenar notificaciones personalizadas para cada usuario.

- **Fichajes y Usuarios:**

  La subcolección `fichajes` dentro de `users` almacena los registros de fichajes diarios para cada usuario, facilitando el seguimiento de las horas de trabajo y asistencia.

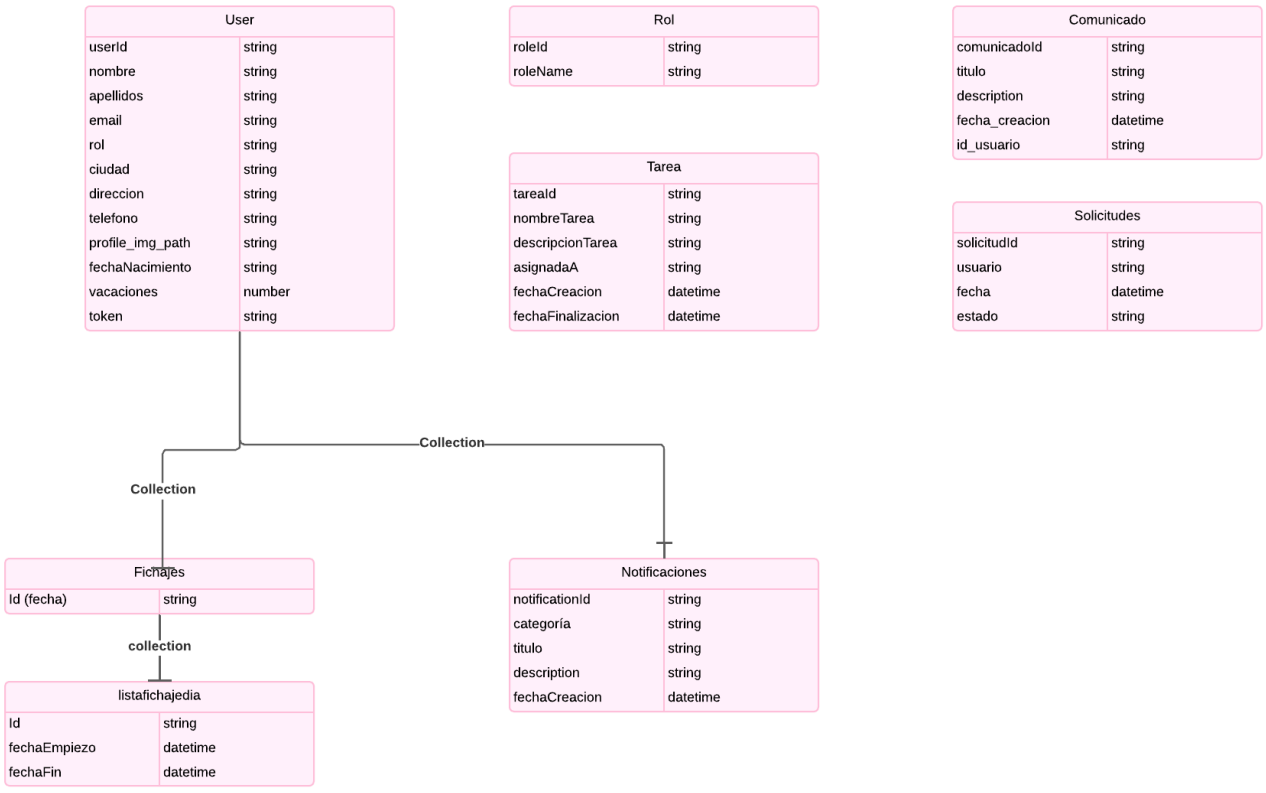
CONCLUSIÓN:

Esta estructura de datos en Firebase Firestore, permite almacenar y gestionar de manera eficiente la información necesaria para el buen funcionamiento de la aplicación.

Las relaciones entre las colecciones están claramente definidas mediante el uso de identificadores únicos, lo que facilita la integración y el acceso a los datos.

Esta organización modular y flexible se adapta bien a las características y requisitos dinámicos de las aplicaciones móviles modernas, asegurando una experiencia de usuario robusta y eficiente.

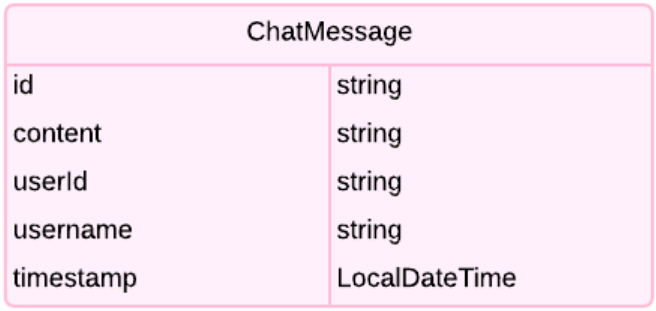
Aquí podemos observar un pequeño diagrama del modelo de datos basado en NoSQL utilizado en Firebase:



MODELO PARA GUARDAR EL HISTÓRICO DEL SERVICIO DE CHAT DE EMPLEADOS:

El siguiente modelo de datos representa un mensaje de chat en una aplicación y se utiliza para almacenar el historial de chat en una base de datos PostgreSQL. Se ha creado una única tabla ChatMessage donde cada registro tendrá los siguientes campos:

- **id:** este campo sirve como identificador único de cada mensaje de chat.
- **content:** aquí se guarda el contenido del mensaje de chat, es decir, el texto del mensaje enviado por el usuario. Este campo se utiliza para almacenar el cuerpo del mensaje y proporciona el contexto del chat.
- **userId:** este campo almacena el identificador del usuario que envió el mensaje. Sirve para asociar cada mensaje con el usuario que lo envió, lo que facilita la recuperación de los mensajes de un usuario específico o la identificación del remitente de un mensaje en el historial de chat.
- **username:** guarda el nombre de usuario del remitente del mensaje. Este campo se utiliza para mostrar el nombre del remitente junto con el mensaje en la interfaz de usuario del chat, lo que ayuda a los usuarios a identificar quién envió cada mensaje.
- **timestamp:** este campo registra la marca de tiempo en la que se envió el mensaje. Utiliza el tipo de dato LocalDateTime, que almacena la fecha y hora precisas en que se envió el mensaje. Esto es fundamental para ordenar los mensajes cronológicamente y mostrar la hora en que se envió cada mensaje en la interfaz de usuario del chat, proporcionando un contexto temporal al historial de chat.



FUNCIONALIDAD DEL PROYECTO:

(LOGIN – REGISTRO – CIERRE DE SESIÓN)

La pantalla de inicio de sesión es la primera interacción que los usuarios tienen con la aplicación TaskSphere. Su diseño está pensado para ser intuitivo y fácil de usar, facilitando el acceso a las distintas funcionalidades de la aplicación.

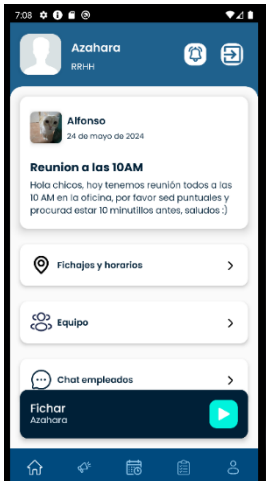
Acceso inicial: al abrir la aplicación, el usuario se encuentra con la pantalla de inicio de sesión, en la cual debe introducir su correo electrónico y su contraseña. Tras introducir las credenciales, el usuario puede optar por marcar el "Checkbox" para recordar su sesión en el futuro.

Iniciar sesión: al hacer clic en el botón "Login", la aplicación valida las credenciales introducidas. Si son correctas, el usuario es redirigido a la pantalla principal de la aplicación.

Cerrar sesión: al hacer clic en el botón de "Cerrar Sesión", se realiza el cierre de sesión y el usuario es redirigido a la pantalla de login.

Creación de cuenta: si el usuario no tiene una cuenta, puede hacer clic en el botón "Crear una cuenta" que le redirigirá a la pantalla de registro, donde deberá completar los campos solicitados para poder registrarse. Una vez registrado, deberá esperar a que el administrador le asigne un rol para poder acceder a la aplicación.

Pantalla de Login



Pantalla principal

Pantalla de registro

(ACTIVIDAD PRINCIPAL)

Una vez que el usuario se ha logeado, pasará por esta actividad. Al iniciarse, la actividad configura la interfaz de usuario y las funcionalidades básicas de la aplicación, como la inicialización de Firebase para la autenticación y el acceso a la base de datos, y la configuración de la navegación entre las diferentes secciones de la aplicación.

En esta actividad se lleva a cabo la inicialización de la barra de navegación inferior, donde la actividad busca el diseño ubicado en el archivo correspondiente de formato XML (BottomNavigationView).

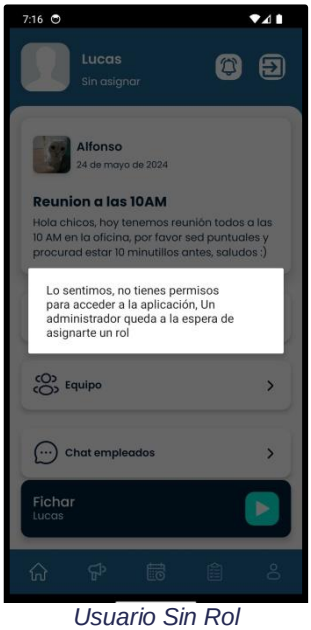
Dentro de este XML, previamente se ha configurado un menú con cada uno de los elementos que conforman nuestra barra de navegación incluyendo cada uno de los iconos correspondientes a cada uno de ellos.

Posteriormente se hace uso de fragment para representar cada una de las pantallas correspondientes a los iconos de la barra de navegación.

Un fragment en Android es como una parte de una pantalla o una sección de la interfaz de usuario que puede contener su propio diseño y funcionalidad. En la actividad principal (MainActivity), se utiliza un fragment para representar y gestionar cada una de las secciones principales de la aplicación, en nuestro caso para la pantalla de Inicio, la pantalla de comunicados, pantalla de calendarios, pantalla de tareas y pantalla de perfil.

Posteriormente, se asigna a cada uno de los botones de la barra de navegación el fragment correspondiente al que tienen que llevar cuando se hace click en ellos.

Y por último, en la "MainActivity" se obtienen los datos del usuario almacenados previamente durante el logeo, en las preferencias compartidas "SharedPreferences" explicadas con anterioridad. Se obtiene el usuario para comprobar si tiene un Rol asignado ya que sino, no podrá acceder a la aplicación y se muestra un dialogo exponiendo que el usuario no tiene un Rol asignado y que está a la espera que un administrador le asigne un Rol.



Usuario Sin Rol

(HOME FRAGMENT)

El fragmento Home es el fragmento inicial de la aplicación y el cual se muestra cuando estamos logeados de forma principal.

En la parte superior del Home Fragment, hay un botón que muestra las notificaciones pendientes. Donde se recogen el conteo de las notificaciones no leídas y lo muestra como un número flotante alrededor del botón. Al hacer clic en este botón, se abre la actividad de notificaciones donde el usuario puede ver todas las notificaciones recibidas.

También en la parte superior del Home Fragment, se encuentra un botón que permite al usuario cerrar sesión en la aplicación. Al hacer clic en este botón, se cierra la sesión actual y se redirige al usuario a la pantalla de inicio de sesión.

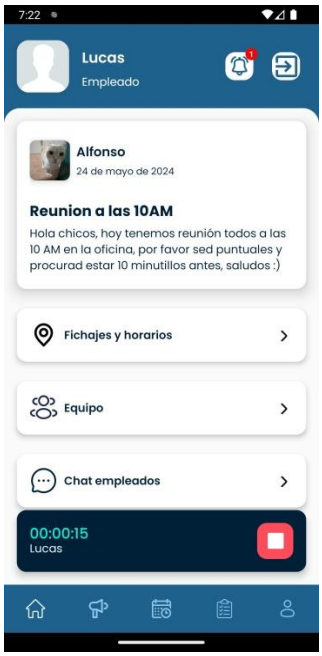
En la pantalla principal del Home Fragment, se muestra el último comunicado emitido. Esto puede ser un mensaje o anuncio importante para todos los usuarios de la aplicación. La visualización del comunicado puede incluir su título, contenido y la fecha en que se emitió.

Botones de acceso rápido: debajo del último comunicado, hay tres botones que proporcionan acceso rápido a diferentes partes de la aplicación:

- Botón de fichajes: Este botón lleva al usuario a la actividad de fichajes, donde puede registrar sus horas de entrada y salida.
- Botón de equipo: Este botón redirige al usuario a la actividad del equipo, donde puede ver información sobre sus compañeros de trabajo, como sus perfiles y contactos.
- Botón de chat global: Al hacer clic en este botón, se abre el chat global de empleados, donde los usuarios pueden comunicarse entre sí en tiempo real.

Y por último en este fragmento se incluye toda la funcionalidad donde los usuarios pueden hacer los fichajes. En la parte inferior de la pantalla, hay un botón flotante que permite al usuario fichar su entrada o salida del trabajo. Antes de mostrar este botón, la actividad comprueba si hay fichajes activos. Si hay un fichaje activo, se muestra un contador de tiempo que indica cuánto tiempo ha estado activo el fichaje y un botón de “Stop” para terminar el fichaje. Habrá un fichaje activo siempre que alguno de los fichajes tenga fecha de inicio, pero no tenga fecha de fin. Si no hay fichajes activos, se muestra un botón de "Play" que permite al usuario iniciar un nuevo fichaje.

Pantalla de Inicio.
Usuario no ha fichado



Pantalla de Inicio.
Usuario ha fichado

(COMUNICADOS FRAGMENT)

La sección de comunicados proporciona una plataforma eficiente para la gestión de comunicados dentro de la organización, facilitando la difusión de información importante y manteniendo a los usuarios informados en todo momento.

Su diseño permite ver una lista de comunicados (con la foto de perfil de su autor y la fecha de publicación, así como el título y la descripción de dicho comunicado). Además, se muestran distintos botones que permiten:

- **Añadir comunicados:** aparecerá una nueva ventana que permitirá añadir un título y un mensaje. Ambos campos deben ser completados para poder agregar el comunicado. Una vez completados, se pulsará el botón “agregar comunicado” y éste será automáticamente añadido a la lista de comunicados. En caso contrario se pulsará el botón “cancelar” y se cerrará la ventana.

- **Editar comunicados:** se mostrará una nueva ventana que permitirá editar el título, el mensaje, o ambos. En caso de realizar cambios se pulsará el botón de “editar comunicado”. En caso contrario se pulsará el botón “cancelar” y se cerrará la ventana.

- **<u>Eliminar comunicados:</u>** aparecerá un diálogo de confirmación, en el que se preguntará al usuario si realmente desea eliminar el comunicado. En caso afirmativo marcará la opción “si” y el comunicado desaparecerá de la lista. En caso contrario, marcará la opción “cancelar” y se cerrará el diálogo.

Dependiendo del rol que el usuario tenga asignado se tendrá acceso a distintas funcionalidades.

Administrador o gerente: como puede observarse en la siguiente imagen, los administradores tendrán acceso a todas las funcionalidades, es decir, podrán añadir nuevos comunicados y editar o eliminar los existentes (tanto los suyos propios, como los que pertenezcan a los distintos usuarios).

Lista Comunicados

Otros roles: el resto de los usuarios también podrán añadir nuevos comunicados, sin embargo, únicamente podrán editar y/o borrar sus propios comunicados.

En las siguientes imágenes se puede observar cómo teniendo asignado el rol de Recursos Humanos (RR.HH), al usuario no se le muestran los botones de editar y eliminar en los comunicados de otros usuarios, sino que sólo aparecen en los suyos propios.

Lista Comunicados

Añadir Comunicado

Comunicado Añadido

Editar Comunicado

Comunicado Editado

Borrar Comunicado

(TAREAS)

Cuando un usuario accede al TasksFragment, se muestra una lista de todas las tareas, ordenadas por orden de creación. Esto proporciona una visión general de todas las actividades en curso y las pendientes.

Se incluyen botones de filtro para permitir que el usuario vea las tareas según su estado: pendientes, sin asignar y terminadas. Esto facilita la gestión y la organización de las tareas, permitiendo al usuario enfocarse en un conjunto específico de actividades según sea necesario.

En este fragmento tenemos varios flujos y depende del rol que asignado se tienen acceso a distintas funcionalidades.

- **Administradores o Gerentes.**

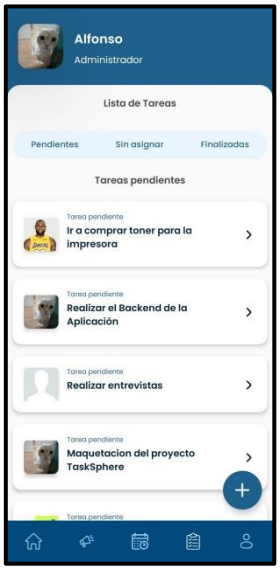
Visualización de tareas:

El usuario con este rol podrá ver todas las tareas de los Empleados, y podrá filtrarlas según los botones de filtro correspondientes. Junto a la tarea en este caso se muestra la foto de perfil del usuario al que se le asignó la tarea en caso de que este asignada a un usuario. Esto proporciona una mayor contextualización y ayuda a los administradores o gerentes a identificar rápidamente quién es responsable de cada tarea.

Añadir nuevas tareas:

Solo los administradores o gerentes tienen acceso al botón de añadir tareas. Al hacer clic en este botón, se abre un diálogo donde pueden ingresar los detalles de una nueva tarea, como el título y la descripción. Además, se incluye un menú desplegable que muestra la lista de todos los empleados disponibles en la empresa para asignar la tarea a un usuario específico. Si la tarea no se asigna a un usuario, se mostrará como "sin asignar".

Antes de agregar una nueva tarea, se verifica que los campos obligatorios, como el título y la descripción, no estén vacíos. Esto garantiza que todas las tareas agregadas contengan la información necesaria para su seguimiento y ejecución.



Pantalla de Tareas
Rol Administrador
Se muestra el botón de añadir

Pantalla de Tareas.
Rol Administrador
Asignando tarea a un Empleado

- **Flujo para otros roles de usuario:**

Visualización de tareas asignadas:

Para usuarios con roles diferentes al de administrador o gerente, el TasksFragment muestra únicamente las tareas asignadas al propio usuario logueado, así como las tareas sin asignar. Esto simplifica la experiencia del usuario al presentar solo las actividades relevantes para ellos.

Filtrado y ordenación de tareas:

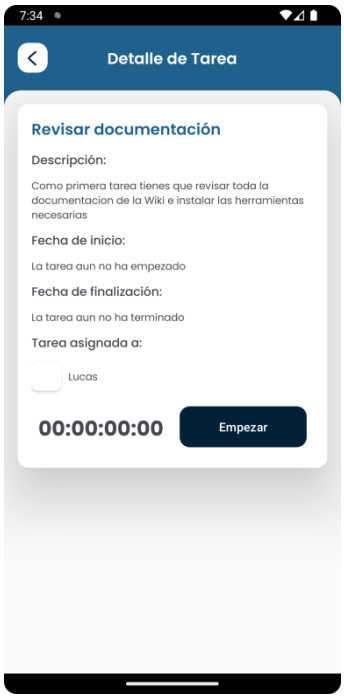
Se proporcionan las mismas opciones de filtro y ordenación que en el flujo para administradores o gerentes, permitiendo así a los usuarios personalizar la visualización de las tareas según sus necesidades y preferencias.

Pantalla de Tareas
Rol Empleado

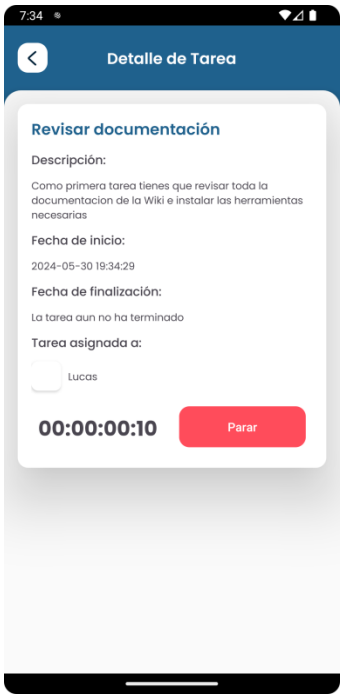
Pantalla de Tareas
Filtro de tareas finalizadas

En ambos casos se puede acceder a los detalles de las tareas al hacer clic en una tarea de la lista, el usuario es dirigido a la actividad de detalle de tareas correspondiente. Aquí, pueden ver información detallada sobre la tarea seleccionada, incluyendo el título, la descripción, el usuario asignado que se muestra mediante la foto de perfil del usuario y su nombre, fecha de inicio, fecha de finalización, y por último aparece un botón para empezar la tarea cuando se quiera empezar, en este caso si la tarea está empezada (pendiente) cuando se entra a la actividad, el contador recogerá el tiempo que lleva

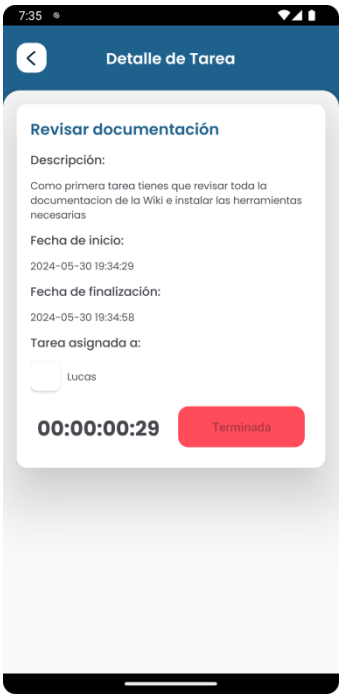
desde que se empezó la actividad, también una vez empezada se dispone del botón para terminar la tarea, que marcará la tarea como finalizada y el contador parará de contar indicando el tiempo total que ha llevado esa tarea, lo que facilita el seguimiento y la gestión de las actividades asignadas al usuario.



Tarea sin empezar



Tarea empezada



Tarea terminada

(VER PERFIL)

El ProfileFragment muestra todos los datos del usuario que está logueado, organizados en dos secciones principales: datos personales y datos laborales. En la sección de datos personales, se incluyen campos como nombre, apellidos, DNI, fecha de nacimiento, dirección, localidad y biografía. En la sección de datos laborales, se muestran campos como empresa, rol y fecha de incorporación.

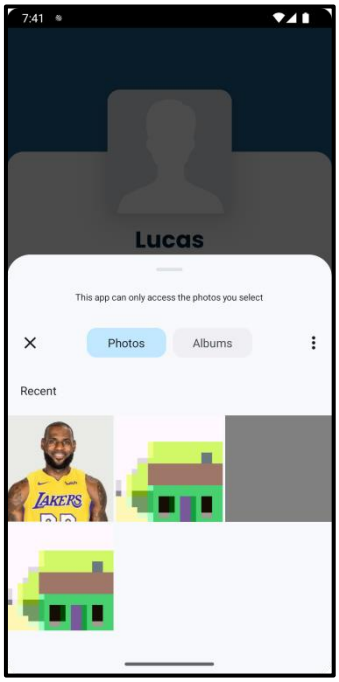
El usuario puede editar sus datos personales haciendo clic en el botón de "editar datos". Esto abre un diálogo personalizado donde cada campo es editable, lo que facilita la modificación de la información. El usuario puede realizar

Pantalla Perfil

cambios en su nombre, apellidos, DNI, fecha de nacimiento, dirección, localidad y biografía según sea necesario.

Al hacer clic en la foto de perfil, se abre un modal que muestra un selector de fotos de la galería. Una vez seleccionada la foto, se utiliza el plugin “Ucrop” que proporciona una interfaz de usuario intuitiva para recortar la imagen en un formato cuadrado. Este paso ayuda a mantener una apariencia uniforme para las fotos de perfil. Una vez recortada la imagen, el propio plugin también tiene unos métodos para comprimir la imagen y así reducir su tamaño y luego se codifica en binario para poder subirla al servidor de Firebase Storage. La URL de la imagen se guarda en el campo profile_img_path de la base de datos de Firebase, lo que permite recuperar y mostrar la foto de perfil del usuario en diferentes partes de la aplicación.

Cualquier cambio realizado en los datos personales o en la foto de perfil se refleja instantáneamente en la pantalla de perfil. Esto se debe a la integración con Firebase, que proporciona una actualización en tiempo real de los datos almacenados en la base de datos.



Seleccionando Imagen

*Recortando Imagen con **Ucrop***

Nueva foto perfil

(VER EQUIPO)

A la actividad "Equipo" se accede desde la pantalla de inicio y muestra una lista de todos los miembros del equipo junto con sus roles. Al hacer clic en un miembro, se abre su perfil, mostrando información personal y laboral.

Si el usuario logueado es un administrador, la pantalla de perfil de cualquier usuario incluirá opciones adicionales para modificar los datos y el rol del usuario.

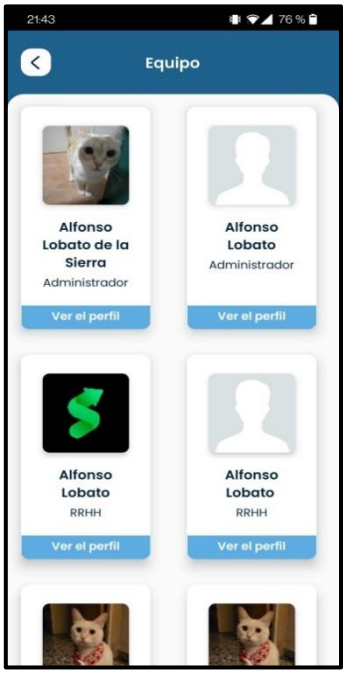
Los administradores pueden cambiar el rol del usuario mediante un selector desplegable (combobox) que muestra todos los roles disponibles. Este cambio se guarda automáticamente en la base de datos una vez seleccionado.

Además, los administradores pueden editar otros datos del usuario, como nombre, dirección y demás información personal, utilizando un diálogo personalizado similar al de la edición de su propio perfil.

Estas modificaciones permiten a los administradores gestionar eficientemente la información del equipo y mantener actualizados los datos de la organización.

Para empleados, esta pantalla solo permite ver la información del perfil de otros usuarios sin capacidad de edición. Esto asegura que solo los administradores puedan realizar cambios, protegiendo la integridad de los datos.

La actividad está diseñada con una interfaz intuitiva, utilizando controles de seguridad y permisos adecuados para diferenciar entre administradores y empleados. Esto facilita una gestión organizada y segura de la información del equipo dentro de la aplicación.



Ver Equipo

Editar Rol
(Solo Administradores)

(VER FICHAJES)

A la actividad "Fichajes" se puede acceder desde la pantalla de inicio y permite a los usuarios ver los registros de fichajes diarios. Dependiendo del rol del usuario, la funcionalidad varía.

Para **administradores**, la actividad muestra un calendario que permite seleccionar un día específico. También aparece un combo box para seleccionar a un empleado. Una vez seleccionados el día y el empleado, se despliega una lista con todos los fichajes realizados por ese empleado en el día seleccionado. Cada fichaje muestra la fecha de inicio, la fecha de finalización y el tiempo total del fichaje. Además, se calcula y se muestra el total de horas fichadas en ese día, sumando el tiempo de todos los fichajes realizados.

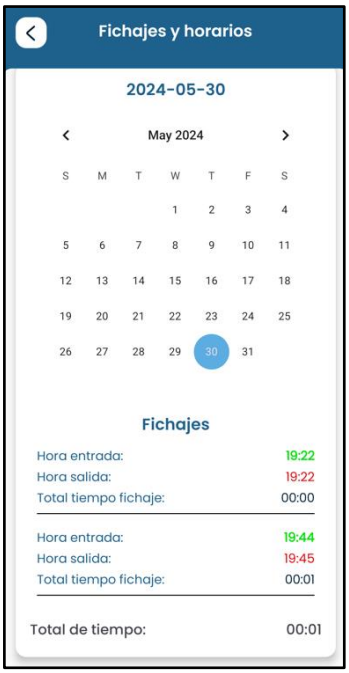
Ver Fichajes
Rol Administrador

Seleccionar empleado
Rol Administrador

Fichajes de Empleado
Rol administrador

Para **empleados**, la actividad permite ver únicamente los propios fichajes. No se muestra el combo box para seleccionar a otros empleados. Al seleccionar un día en el calendario, el empleado puede ver una lista con sus fichajes de ese día. Cada fichaje incluye la fecha de inicio, la fecha de finalización y el tiempo total del fichaje. También se muestra el total de horas fichadas en el día seleccionado, sumando el tiempo de todos los fichajes del empleado.

Los administradores pueden revisar y gestionar los fichajes de cualquier empleado, mientras que los empleados solo tienen acceso a sus propios registros, garantizando la privacidad y seguridad de los datos.



Ver Fichajes
Rol Empleado

(NOTIFICACIONES)

En la actividad "Notificaciones" accesible desde la pantalla de Inicio, se muestran todas las notificaciones recibidas por cada usuario. Los usuarios pueden ver sus notificaciones y tienen la opción de borrar cualquier notificación desde esta pantalla.

Nuestra aplicación maneja varios tipos de notificaciones, cada una con un propósito específico:

- Notificación a los administradores cuando un usuario se registra: esta notificación solicita a los administradores que asignen un rol al nuevo usuario registrado en la aplicación.
- Notificación al usuario cuando se le ha asignado un rol: informa al usuario de que se le ha asignado un rol específico, permitiéndole acceder a las funcionalidades correspondientes dentro de la aplicación.

- Notificación a los empleados cuando un administrador les ha asignado una tarea: notifica a los empleados sobre nuevas tareas asignadas por los administradores, facilitando la gestión y el seguimiento de sus responsabilidades.

- Notificación a los administradores cuando un usuario ha solicitado vacaciones: alerta a los administradores de que un usuario ha hecho una solicitud de vacaciones, lo que requiere su revisión y aprobación.

- Notificación a todos los usuarios cuando alguien ha escrito un comunicado: informa a todos los usuarios sobre nuevos comunicados publicados, asegurando que estén al tanto de anuncios importantes dentro de la organización.

Pantalla notificaciones

Notificaciones

(CHAT DE EMPLEADOS)

En la actividad "Chats" accesible desde la pantalla de Inicio, se muestra un chat en tiempo real para todos los empleados de la empresa. Para el chat se ha optado por desplegar un microservicio hecho con el framework de Springboot en Java, el cual ofrece diferentes endpoints para realizar la comunicación. Se ha integrado una base de

datos PostgreSQL para persistir cada uno de los mensajes enviados por los usuarios. PostgreSQL es una base de datos relacional ampliamente utilizada y compatible con Spring Boot.

Al entrar a la actividad, desde el cliente android se realiza una petición HTTP al endpoint chat/history del servidor. Esta solicitud se realiza al servidor para recuperar todos los mensajes almacenados en la base de datos. Una vez obtiene los mensaje se muestran de manera ordenada por fecha en la aplicación.

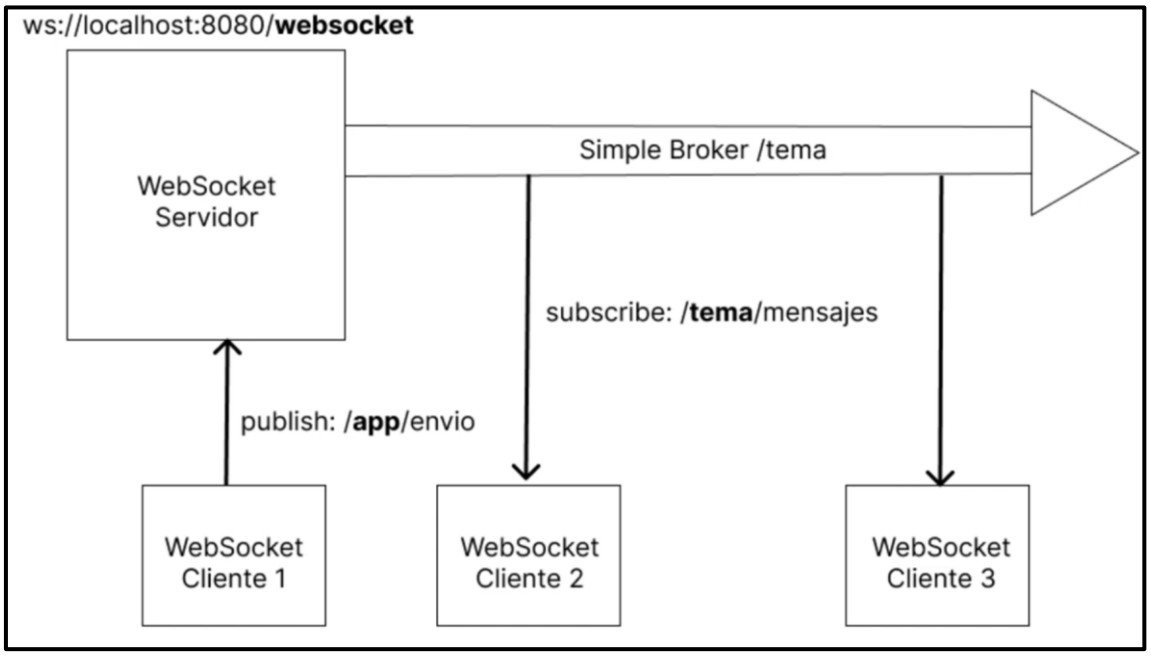
Una vez se han obtenido todos los mensajes, desde el cliente android se establece la comunicación entre cliente y servidor mediante websockets desde el endpoint /chat.

Una vez que se establece la conexión, cada uno de los clientes indican al servidor que se van suscribirán al bróker de mensajería y este es el responsable de mandar los mensajes a cada uno de los clientes que se haya suscrito al tema específico, en este caso los clientes se suscriben a "/topic". El broker de mensajes actúa como un intermediario entre los clientes y el tema, facilitando la distribución eficiente de mensajes en tiempo real.

Para el envío de mensajes, se crea un objeto que representa el mensaje que el usuario desea enviar. Este objeto contiene la información con el contenido del mensaje, el userId que lo hizo, la fecha… Posteriormente este objeto se serializa en formato Json y es enviado por el canal de mensajería habilitado para ello desde el servidor, en este caso /chat.sendMessage, el controlador se encarga de persistir el mensaje en la base de datos, y de enviar el mensaje por el bróker de mensajeria "/topic". El mensaje llega por el bróker de mensajes y este se encarga de mandar el mensaje a todos los usuarios suscritos.

De esta forma usando WebSockets, se establece una conexión persistente entre el cliente y el servidor. Esta conexión bidireccional permite que tanto el cliente como el servidor envíen mensajes activamente en cualquier momento, lo que facilita la comunicación en tiempo real sin la latencia asociada con la creación y el cierre de conexiones. Que en comparación con este implica solicitudes y respuestas individuales para cada interacción lo cual ralentiza la comunicación.

A continuación, se muestra un esquema del funcionamiento:



Despliegue del servicio.

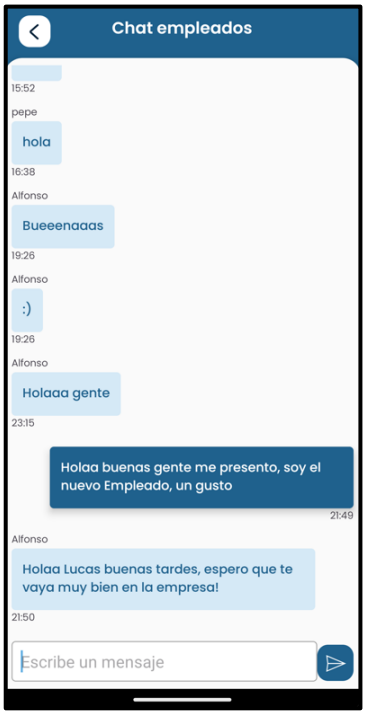
El despliegue del servicio se ha realizado utilizando Docker, una plataforma de contenedores que simplifica la implementación y la gestión de aplicaciones. Para desplegar la aplicación de Spring Boot en Docker, se han realizado los siguientes pasos:

1. Preparación del Contenedor Docker: se ha creado un archivo de configuración llamado Dockerfile que define cómo se debe construir el contenedor Docker. Este archivo incluye instrucciones para copiar los archivos de la aplicación Spring Boot, configurar el entorno de ejecución que va a ejecutarse sobre la imagen de Java y también exponer los puertos necesarios.

2. Construcción de la Imagen Docker: utilizando el Dockerfile, se construye una imagen Docker que contiene todos los componentes necesarios para ejecutar la aplicación Spring Boot, incluidas las dependencias, el servidor de aplicaciones y el código de la aplicación en sí.
3. Exportación de la Aplicación Spring Boot: la aplicación Spring Boot se compila y se empaqueta en un archivo ejecutable, como un archivo JAR. Este archivo se copia dentro del contenedor Docker durante el proceso de construcción de la imagen.

4. Creación del Contenedor Docker: una vez construida la imagen Docker, se ejecuta un contenedor Docker a partir de esta imagen. El contenedor Docker es una instancia en ejecución de la imagen Docker, que contiene todos los componentes necesarios para ejecutar la aplicación Spring Boot de manera independiente y aislada.

5. Despliegue del Contenedor Docker en la nube: para el despliegue en la nube se ha hecho uso de la web de https://zeabur.com/ que ofrece una capa gratuita. Este proporciona un servidor en el que ejecutar nuestro servicio. En este caso para, se vincula la URL del repositorio de Github que contiene el servicio y el propio servidor es el que se encarga mediante Docker de construir la imagen haciendo uso del Dockerfile y posteriormente ejecutándola. En este también se despliega la base de datos PostgresSQL que va a consumir la aplicación.



Chat Empleados

(CALENDARIO)

Es una sección dentro de la aplicación que permite a los usuarios ver y gestionar sus días de vacaciones y otras fechas importantes. Esta pantalla está diseñada para ofrecer una vista clara y sencilla del calendario y las vacaciones disponibles.

Visualización de Perfil y Rol:

- En la parte superior de la pantalla, el usuario puede ver su **nombre de usuario** y su **rol** dentro de la organización.
- También se muestra una **imagen de perfil** dentro de una tarjeta (CardView).

Gestión de Fechas y Vacaciones:

- **Seleccionar Fecha**: un botón permite a los usuarios seleccionar una fecha en el calendario. Esto es útil para marcar días específicos como días de vacaciones.
- **Información de Vacaciones**: la pantalla muestra información relevante sobre los días de vacaciones disponibles y solicitados.
- **Solicitar Día de Vacaciones**: un botón que permite a los usuarios solicitar un día de vacaciones directamente desde la pantalla.
- **Gestionar Vacaciones**: otro botón que permite a los usuarios con rol de administrador acceder a la gestión de las vacaciones del resto de usuarios para ver un resumen o modificar solicitudes pendientes.

Vista Pantalla Calendario

(GESTIÓN DE VACACIONES)

Está destinada a la gestión de aprobaciones de solicitudes de vacaciones por parte de los usuarios con rol de administrador dentro de la aplicación. Aquí se detallan las funcionalidades disponibles para el usuario en esta pantalla:

- **Visualización de Solicitudes Pendientes**: la actividad incluye un RecyclerView que muestra un listado de todas las solicitudes pendientes de aprobación.
- **Acción de Aceptar o Denegar Solicitudes**: cada solicitud tiene opciones para ser aceptada o denegada, permitiendo a los administradores gestionar las solicitudes de manera eficiente.

Cada ítem en la lista de solicitudes incluye los siguientes elementos:

- **Nombre del Usuario**: un TextView para mostrar el nombre del usuario que hizo la solicitud.
- **Fecha de Solicitud**: un TextView para mostrar la fecha en la que se realizó la solicitud.
- **Botones de Aceptar y Denegar**: botones para realizar acciones de aceptación o denegación de la solicitud.

Vista Pantalla Gestión Vacaciones

CONCLUSIONES Y MEJORAS DEL PROYECTO:

Este proyecto ha demostrado ser una herramienta integral capaz de revolucionar la forma en que las PyMEs operan diariamente, ofreciendo una interfaz intuitiva y con funcionalidades potentes que mejoran su productividad y eficiencia.

A través de TaskSphere, los comercios pueden ahora delegar tareas, establecer prioridades, supervisar el progreso, gestionar turnos y horarios, vacaciones y horas extras, así como, enviar notificaciones y comunicados de manera centralizada.

Estas funcionalidades no solo cierran la brecha tecnológica existente respecto a las grandes corporaciones, sino que, además, proporcionan una solución escalable y personalizable adaptada a las necesidades de estos negocios.

PROPUESTAS DE MEJORA:

- **Asignación de turnos**: permitir que los administradores agreguen y modifiquen los turnos, incluyendo opciones para definir horarios específicos, días de la semana y turnos recurrentes.
- **Autenticación social:** implementar opciones de inicio de sesión mediante cuentas de Facebook o Google, asegurando un proceso de "login" seguro y fluido.
- **Notificaciones globales para nuevos comunicados:** configurar un sistema de notificaciones que se active automáticamente al publicar un nuevo comunicado, garantizando que todos los empleados reciban la información.
- **Integrar módulos adicionales:** reportes personalizados, integración de herramientas de contabilidad y CRM o soporte para múltiples idiomas.
- **Fortalecer las medidas de seguridad:** para proteger la información sensible de los comercios y sus empleados.
- **Desarrollar versiones optimizadas**: para iOS y web y garantizar la accesibilidad para personas con discapacidades siguiendo las pautas WCAG.
- **Feedback:** implementar encuestas de satisfacción y un sistema de retroalimentación dentro de la aplicación, para mantener una comunicación constante con los usuarios que permita identificar áreas de mejora.
- **Fichajes: i**mplementar una opción para que lea datos biométricos como la huella dactilar a la hora de hacer un fichaje para que otros usuarios no puedan hacerlo en tu nombre dándole el usuario y la contraseña. También para este apartado se pensó la

posibilidad de que a la hora de fichar guardar los datos de la ubicación, para saber si se está fichando desde la ubicación de la empresa.

- **Chat:** posibilidad de adjuntar archivos al chat y mandar audios.

Estas propuestas no sólo incrementarán la satisfacción del usuario final, sino que también asegurarán que TaskSphere se mantenga como una herramienta indispensable y competitiva en el mercado de gestión empresarial.

BIBLIOGRAFÍA:

- Campus FP UNIR. (s.f.). *Campus FP UNIR*. Recuperado de https://campusfp.unir.net/
- Flaticon. (s.f.). *Flaticon* Recuperado de https://www.flaticon.es/
- GitHub. (s.f.). *GitHub*. Recuperado de https://github.com/
- Stack Overflow en español. (s.f.). *Stack Overflow en español*. Recuperado de https://es.stackoverflow.com/
- YouTube. (s.f.). *YouTube*. Recuperado de https://www.youtube.com/
- Firebase (s.f.). *Firebase*. Recuperado de https://firebase.google.com/?hl=es
- Zeabur. (s.f). *Zeabur*. Recuperado de https://zeabur.com/

ANEXOS:

El código perteneciente a todo el desarrollo de la aplicación se encuentra en el siguiente repositorio:

- https://github.com/alejandromfp/TaskSphere.git

En él se encuentran dos carpetas principales:

- **TaskSphere:** contiene el proyecto de Android Studio con todo el código principal de la aplicación.
- **Chat-TaskSphere:** contiene el microservicio de Chat hecho con SpringBoot y con él un archivo Dockerfile para poder generar el contenedor en Docker para desplegar el servicio.