

# AGENDA SATUR

## Aplicación Android - Desarrollo Avanzado de Software

**Titulación:**

Grado en Ingeniería Informática de Gestión y Sistemas de Información

**Curso:**

4º Curso (2º Cuatrimestre)

**Fecha:**

16 de marzo de 2025

*Aritz Blasco*

## Índice

<b>1. Enlace al repositorio Git.....</b>	<b>3</b>
<b>2. Diagramas de clases.....</b>	<b>4</b>
<b>3. Enumeración de elementos utilizados y funcionalidades.....</b>	<b>7</b>
3.1. Descripción de las clases.....	7
3.2. Descripción de los archivos .xml.....	11
3.3. Descripción de los values.....	13
<b>4. Manual de Usuario.....</b>	<b>14</b>
Inicio de la Aplicación.....	14
Menú Lateral (Drawer).....	15
Pantalla del Calendario.....	16
Pantalla de Tareas.....	17
<b>5. Dificultades encontradas.....</b>	<b>18</b>
<b>6. Fuentes utilizadas.....</b>	<b>20</b>

## Índice de Figuras

<a href="#">Figura 1. Diagrama de clases de la Base de Datos.....</a>	<a href="#">4</a>
<a href="#">Figura 2. Diagrama de clases de la UI y las Notificaciones.....</a>	<a href="#">5</a>
<a href="#">Figura 3. Diagrama de clases general de la aplicación.....</a>	<a href="#">6</a>
<a href="#">Figura 4. Pantalla de Inicio.....</a>	<a href="#">14</a>
<a href="#">Figura 5. Drawer de la aplicación.....</a>	<a href="#">15</a>
<a href="#">Figura 6. Pantalla del Calendario.....</a>	<a href="#">16</a>
<a href="#">Figura 6. Pantalla tareas.....</a>	<a href="#">17</a>
<a href="#">Figura 7. Añadir tarea.....</a>	<a href="#">17</a>
<a href="#">Figura 8. Detalles de la tarea.....</a>	<a href="#">17</a>

## 1. Enlace al repositorio Git

<https://github.com/Surtur9/AgendaSatur/tree/main>

## 2. Diagramas de clases

En este punto se exponen 3 diagramas diferentes uno específico sobre la base de datos ([ver Figura 1](#)), uno específico para la UI y las notificaciones ([ver Figura 2](#)) y uno último general de todo el proyecto ([ver Figura 3](#)). Todos estos archivos están subidos en el github del proyecto a mayor calidad.

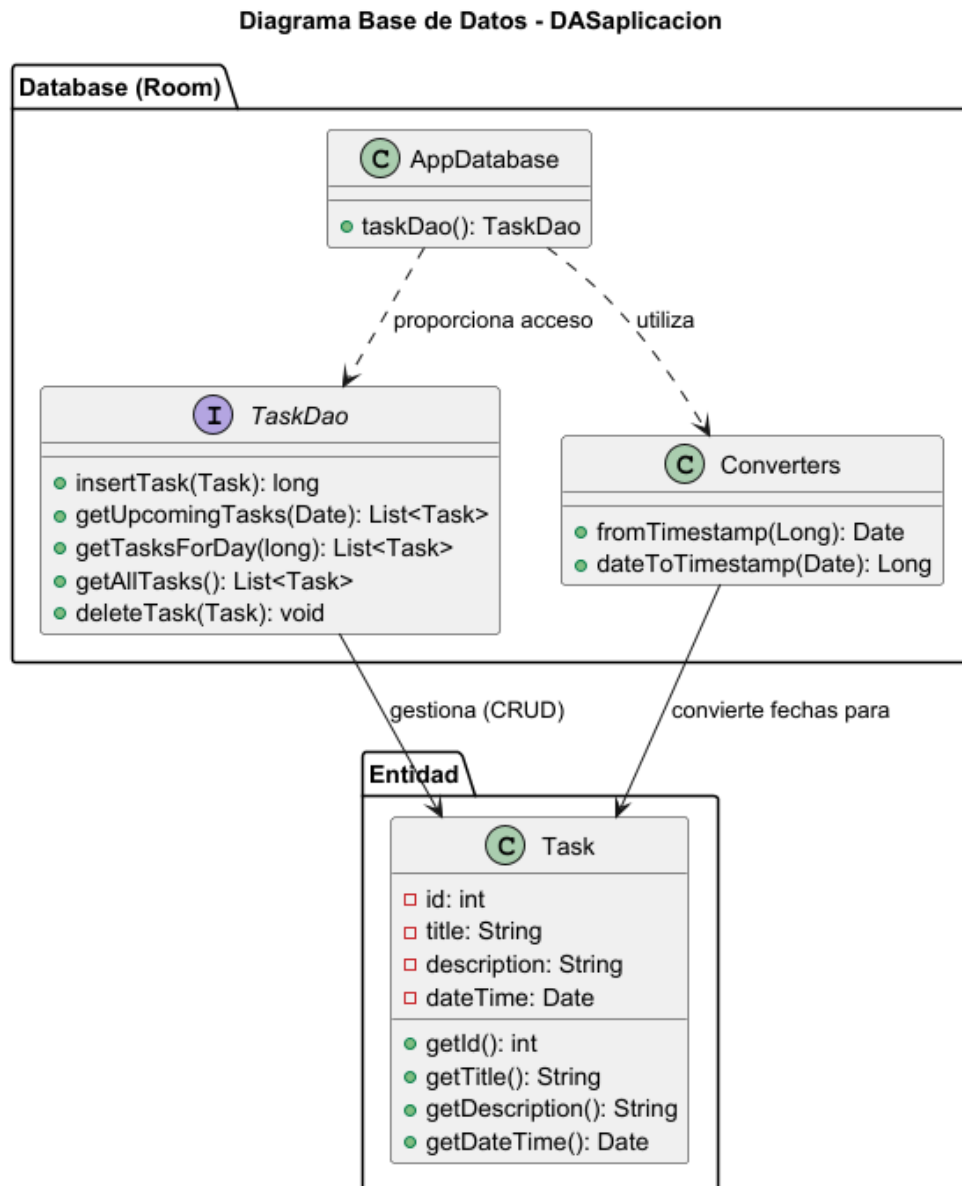


Figura 1. Diagrama de clases de la Base de Datos.

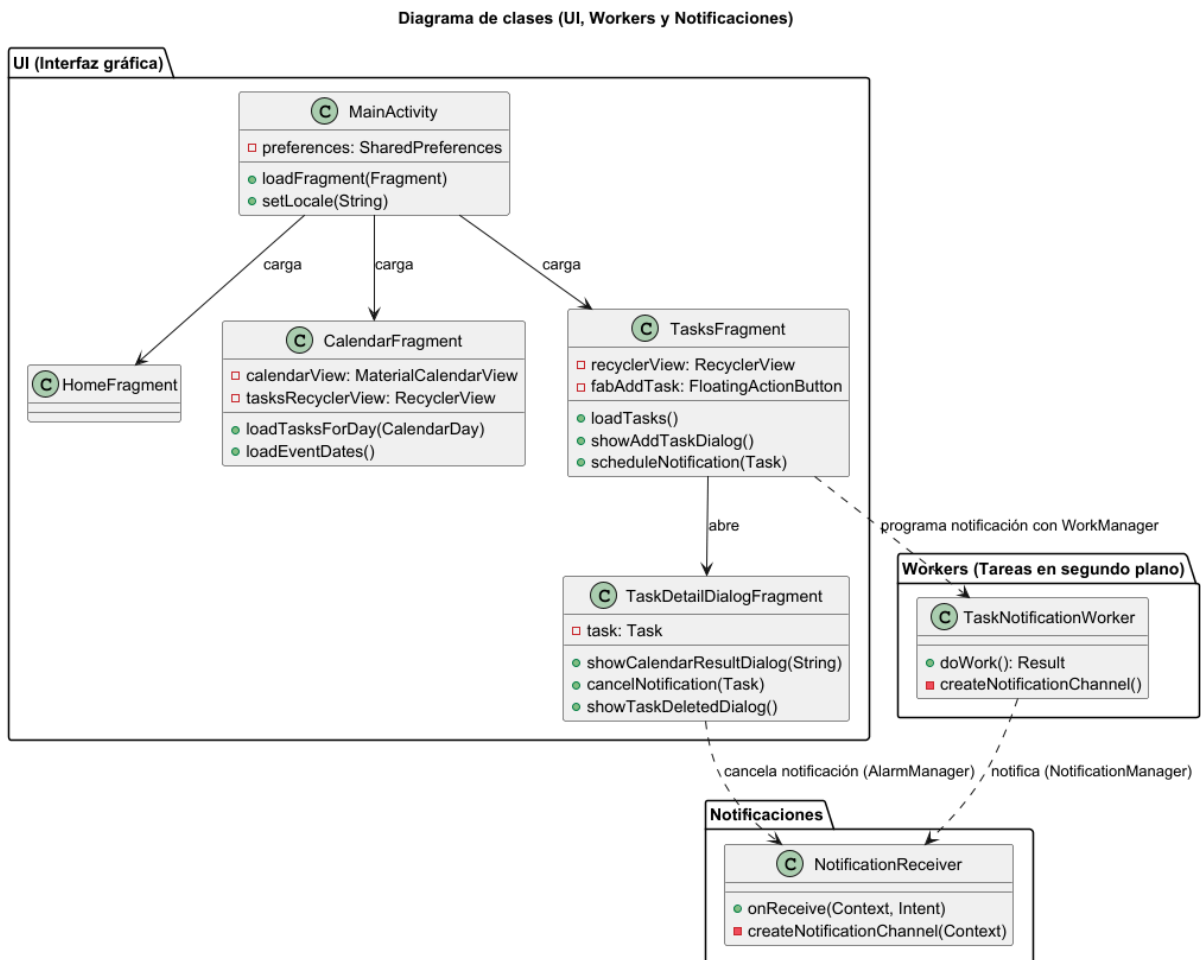


Figura 2. Diagrama de clases de la UI y las Notificaciones.

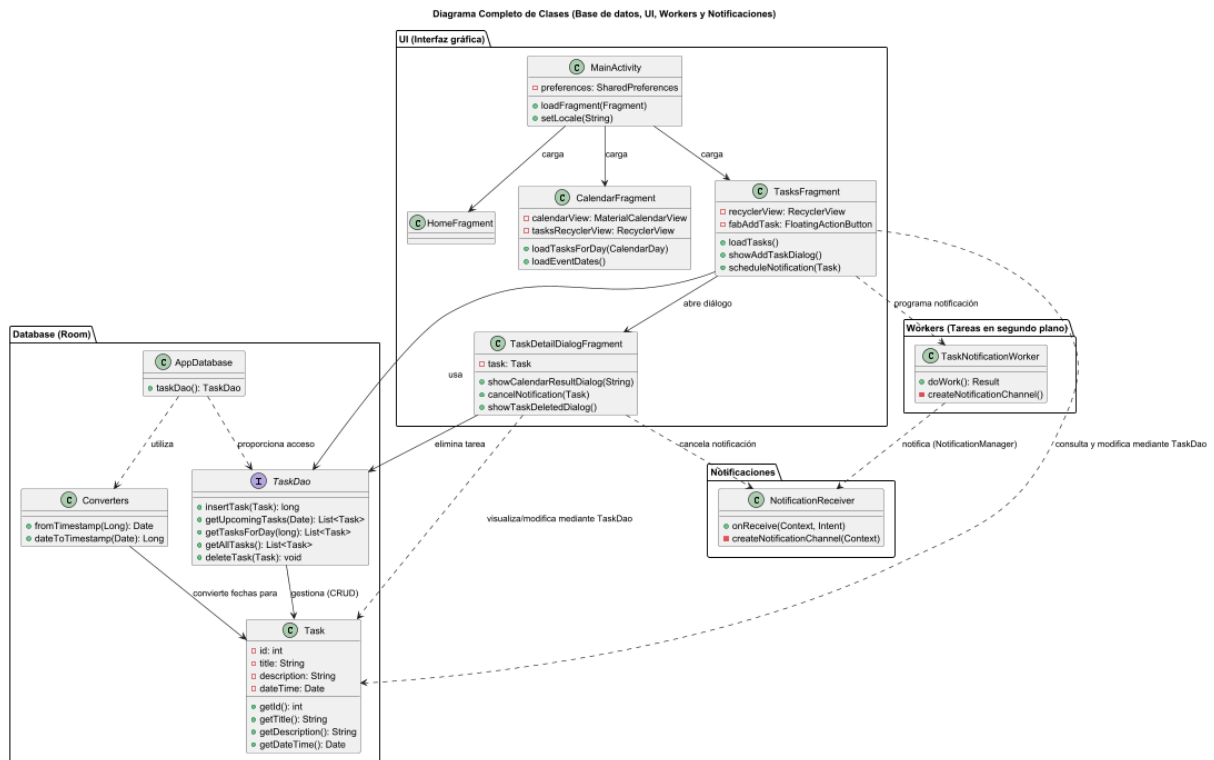


Figura 3. Diagrama de clases general de la aplicación

## 3. Enumeración de elementos utilizados y funcionalidades

### 3.1. Descripción de las clases

#### MainActivity.java

Es la actividad principal y punto de entrada de la aplicación. Gestiona principalmente la navegación entre los distintos fragmentos (`HomeFragment`, `CalendarFragment`, `TasksFragment`) mediante la barra de navegación inferior (`BottomNavigationView`). Controla también la configuración general del usuario almacenada en `SharedPreferences`, incluyendo el idioma seleccionado, el tema visual (oscuro o claro), y las notificaciones. Esta clase verifica y solicita dinámicamente los permisos necesarios para notificaciones y acceso al calendario del dispositivo al iniciar la aplicación, asegurando así que la aplicación pueda integrarse correctamente con las funcionalidades nativas del sistema Android.

Principales responsabilidades:

- Manejo dinámico del idioma y del tema visual.
- Gestión de permisos en tiempo de ejecución.
- Navegación mediante `BottomNavigationView`.
- Configuración persistente mediante `SharedPreferences`.

#### HomeFragment.java

Este fragmento representa la pantalla principal (Home) de la aplicación, ofreciendo al usuario información general o introductoria. Su contenido suele incluir instrucciones básicas sobre el funcionamiento o características destacadas de la aplicación. No maneja interacción compleja ni lógica de negocio, siendo principalmente informativa y facilitando una experiencia de bienvenida amigable al usuario. Utiliza recursos en múltiples idiomas desde archivos `strings.xml`, adaptándose dinámicamente al idioma seleccionado por el usuario desde la configuración.

Principales responsabilidades:

- Visualización informativa inicial.
- Adaptación dinámica de textos según idioma.

## CalendarFragment.java

Este fragmento contiene una visualización avanzada e interactiva del calendario mediante la librería **MaterialCalendarView**. Permite al usuario explorar las tareas asociadas a cada día del mes. Utiliza decoradores personalizados para resaltar visualmente días específicos en los que hay tareas almacenadas en la base de datos. Además, se adapta automáticamente al idioma seleccionado mostrando días de la semana y meses en español, inglés o euskera según la preferencia del usuario.

Principales responsabilidades:

- Integración visual e interactiva del calendario.
- Marcado dinámico de eventos/tareas en calendario.
- Carga dinámica de tareas desde la base de datos **Room**.
- Soporte multi-idioma con recursos dinámicos.

## TasksFragment.java

Gestiona una lista detallada y organizada cronológicamente de las tareas futuras creadas por el usuario. Facilita la creación de nuevas tareas mediante un formulario personalizado (diálogo), permitiendo al usuario especificar título, descripción, fecha y hora para cada tarea. Además, maneja la programación automática de notificaciones mediante **WorkManager**, notificando al usuario una hora antes del inicio de la tarea. Muestra cada tarea agrupada bajo un encabezado indicando la fecha correspondiente de forma clara y organizada.

Principales responsabilidades:

- Gestión completa de tareas (crear, mostrar y eliminar).
- Programación automática de notificaciones mediante **WorkManager**.
- Uso avanzado de diálogos personalizados para introducción de datos.
- Soporte multi-idioma, usando recursos dinámicos para días y meses.
- Integración eficiente con base de datos **Room** para persistencia de tareas.



## TaskDetailDialogFragment.java

Diálogo interactivo personalizado que aparece al pulsar sobre una tarea específica en la lista de tareas. Permite al usuario ver detalles completos de la tarea (título, descripción, fecha/hora). Además, posibilita acciones adicionales como añadir fácilmente la tarea al calendario personal del dispositivo (utilizando intents del sistema operativo) o eliminar permanentemente la tarea, cancelando automáticamente la notificación programada previamente. Todas estas acciones se comunican eficientemente con la base de datos Room y otros fragmentos mediante callbacks (listeners).

Principales responsabilidades:

- Visualización detallada de tareas individuales.
- Integración directa con el calendario nativo del dispositivo.
- Eliminación de tareas con cancelación automática de notificaciones.
- Comunicación eficiente con otros fragmentos mediante callbacks.

## NotificationReceiver.java

Clase crítica encargada de recibir y gestionar las notificaciones programadas de tareas mediante un BroadcastReceiver. Cuando el sistema dispara una alarma programada previamente, esta clase se encarga de construir y mostrar una notificación clara e informativa al usuario. Integra la información detallada de la tarea correspondiente (como título y descripción) en la notificación, permitiendo que el usuario sea alertado oportunamente sobre las tareas próximas en su agenda.

Principales responsabilidades:

- Recepción y gestión de notificaciones mediante BroadcastReceiver.
- Construcción dinámica de notificaciones informativas.
- Integración directa con AlarmManager y PendingIntent.

## AppDatabase.java

Clase central que representa la base de datos local, implementada con Room, un ORM de SQLite recomendado por Google para Android. Define explícitamente la entidad **Task**, además de especificar los conversores necesarios (**Converters**) para transformar tipos de datos complejos, como fechas, en formatos compatibles con SQLite. Actúa como punto único de acceso global a la base de datos y facilitando operaciones consistentes.

Principales responsabilidades:

- Definición centralizada de entidades y DAOs.
- Manejo de conversión de tipos de datos complejos.
- Facilita acceso consistente y eficiente a datos persistentes.

## TaskDao.java

Interfaz DAO (Data Access Object) para definir operaciones concretas sobre las tareas almacenadas en la base de datos Room. Proporciona métodos claramente definidos para insertar, eliminar, y consultar tareas, con especial énfasis en consultas dinámicas para tareas futuras. Facilita la separación clara de responsabilidades entre la capa de datos y la lógica de negocio en los fragmentos de la aplicación.

Principales responsabilidades:

- Definición precisa de consultas SQL mediante anotaciones Room.
- Operaciones CRUD específicas para tareas.
- Consultas dinámicas y filtradas por fecha y hora.

## Task.java

Clase modelo representando la entidad de una tarea concreta en la aplicación. Contiene atributos clave como título, descripción y fecha/hora de ejecución. Está claramente definida con anotaciones Room para indicar la clave primaria (**id**) generada automáticamente, facilitando su persistencia eficiente en la base de datos local. Actúa como estructura básica en torno a la cual gira gran parte de la lógica de negocio y visualización de la aplicación.

Principales responsabilidades:

- Modelo de datos centralizado para tareas.
- Facilita persistencia clara y eficiente mediante Room.
- Base estructural para gestión visual y lógica del proyecto.

## Converters.java

Clase de soporte que proporciona métodos de conversión especializados necesarios para almacenar tipos de datos complejos (como objetos **Date**) en la base de datos SQLite mediante Room. Define métodos claros para transformar fechas en valores primitivos (Long) y viceversa, facilitando así su almacenamiento eficiente y su recuperación sencilla desde la base de datos local.

Principales responsabilidades:

- Conversión clara y eficiente de tipos complejos (fechas).
- Facilita la integración sin problemas con SQLite mediante Room.

## 3.2. Descripción de los archivos .xml

### **activity\_main.xml**

Define la estructura principal de la aplicación. Utiliza un `DrawerLayout` que incluye un `ConstraintLayout` con una barra de navegación inferior (`BottomNavigationView`) y un contenedor para cargar los fragmentos. También contiene un `NavigationView` lateral (Drawer) con un encabezado personalizado para preferencias rápidas como idioma, tema y notificaciones.

### **custom\_toolbar.xml**

Define una barra de herramientas personalizada, mostrando un título alineado a la izquierda y un logotipo a la derecha. Se utiliza consistentemente en toda la aplicación para mejorar la experiencia visual y de navegación del usuario.

### **dialog\_add\_task.xml**

Representa el formulario en un diálogo emergente para añadir nuevas tareas. Incluye campos para título, descripción, y botones para seleccionar la fecha y hora. Está optimizado visualmente tanto para el modo claro como para el oscuro.

### **dialog\_task\_detail.xml**

Diálogo con diseño en `CardView` que muestra los detalles completos de una tarea específica. Contiene botones para cerrar el diálogo, añadir la tarea al calendario personal o eliminarla. Utiliza un scroll para la descripción, facilitando la visualización de textos largos.

### **fragment\_calendar.xml (orientación vertical)**

Pantalla principal del calendario, con un diseño vertical que muestra un calendario interactivo (`MaterialCalendarView`) en la parte superior, y debajo una lista (`RecyclerView`) de las tareas del día seleccionado. Adaptable según el idioma seleccionado.

## **fragment\_calendar.xml (orientación horizontal)**

Variante adaptada para orientación horizontal. Divide la pantalla en dos partes iguales, mostrando el calendario a la izquierda y la lista de tareas a la derecha. Ideal para dispositivos con pantallas más anchas como tablets.

## **fragment\_home.xml**

Layout sencillo con texto centralizado que muestra una breve descripción o bienvenida al usuario. El contenido es dinámico y multi-idioma, cambiando según la selección del usuario.

## **fragment\_tasks.xml**

Pantalla para gestionar tareas. Muestra un título en la parte superior, una lista dinámica (**RecyclerView**) para visualizar las tareas pendientes ordenadas por fecha, y un botón flotante (FAB) para agregar nuevas tareas de forma intuitiva.

## **item\_task.xml**

Define el diseño individual para cada elemento de la lista de tareas. Usa **CardView** para mostrar título y fecha/hora de cada tarea. También incluye un encabezado opcional para agrupar tareas por día de forma clara y visualmente agradable.

## **nav\_header.xml**

Layout personalizado del encabezado del menú lateral (drawer). Incluye un logo central, switches para activar/desactivar notificaciones y tema oscuro, y un selector desplegable para elegir el idioma. Permite una configuración rápida y cómoda para el usuario.

### 3.3. Descripción de los valores

#### **arrays.xml (Español, Inglés y Euskera)**

Almacena arrays de cadenas de texto utilizados para mostrar listas y selecciones, tales como días de la semana, meses del año y opciones de idioma. Proporciona soporte completo para multi-idioma en la aplicación.

#### **attrs.xml**

Define atributos personalizados para temas visuales, permitiendo configurar colores específicos según el modo (claro/oscuro) en la interfaz de usuario. Facilita una personalización eficiente y unificada de los estilos visuales.

#### **colors.xml**

Define los colores básicos usados en la aplicación, incluyendo fondos, textos y elementos visuales destacados. Asegura la coherencia visual en toda la interfaz de usuario.

#### **strings.xml (Español, Inglés y Euskera)**

Contiene todas las cadenas de texto estáticas usadas en la aplicación, como etiquetas, títulos, mensajes informativos, notificaciones y descripciones. Este archivo es clave para el soporte multi-idioma, permitiendo mostrar automáticamente los textos en el idioma preferido del usuario.

#### **styles.xml**

Define estilos específicos reutilizables en componentes visuales particulares, como texto del calendario y diálogos redondeados personalizados. Permite mantener uniformidad visual en elementos comunes a través de toda la aplicación.

#### **themes.xml (modo claro y oscuro)**


Establece el tema visual general de la aplicación utilizando Material Design 3. Especifica configuraciones base, colores de fondo, superficie y elementos visuales, proporcionando soporte integrado para modos claro y oscuro.

## 4. Manual de Usuario

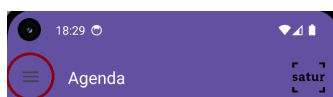
AgendaSatur es una aplicación móvil para gestionar tareas de forma local, integrada con un calendario interactivo que permite visualizar, crear y eliminar tareas, además de notificaciones personalizadas.

### Inicio de la Aplicación

Al abrir la aplicación, se muestra la pantalla principal (Inicio) que presenta una descripción breve sobre las funcionalidades disponibles en la app.

En esta pantalla, el usuario puede acceder al drawer pulsando () , además de poder utilizar la barra de navegación inferior para moverse rápidamente entre las secciones principales ([ver Figura 4](#)):

- **Calendario (2)**
- **Inicio (3)**
- **Tareas (4)**



1

AgendaSatur es una aplicación de agenda y gestión de tareas que funciona de forma local.

Entre sus funcionalidades se encuentran:

- Listado de tareas con notificaciones.
- Calendario interactivo que marca los días con eventos.
- Creación, modificación y eliminación de tareas.
- Modo oscuro y claro.
- Soporte multi-idioma.

Desarrollada por Aritz Blasco para la asignatura de Desarrollo Avanzado de Software.



Figura 4. Pantalla de Inicio.

## Menú Lateral (Drawer)

Accesible desde el icono superior izquierdo (☰), muestra opciones adicionales [\(ver Figura 5\)](#) para:

- Activar o desactivar **Notificaciones**.
- Activar o desactivar el **modo oscuro**.
- Cambiar el **idioma** entre Castellano, Euskera e Inglés.

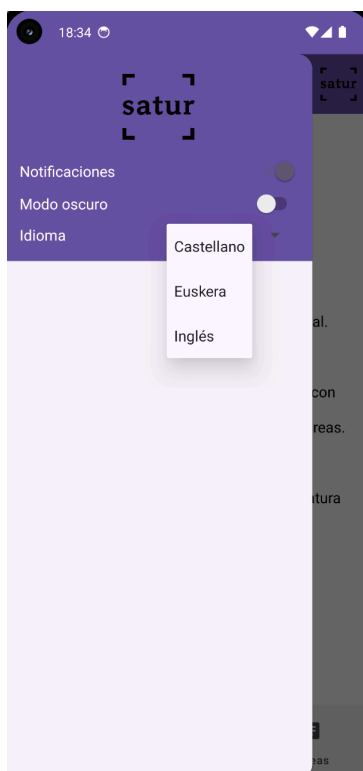


Figura 5. Drawer de la aplicación.

## Pantalla del Calendario

En esta sección, el usuario podrá visualizar un calendario interactivo donde aparecen marcadas aquellas fechas en las que existe alguna tarea pendiente [\(ver Figura 6\)](#).

- Al seleccionar un día específico del calendario, la parte inferior de la pantalla mostrará una lista con todas las tareas programadas para ese día.
- Pulsando sobre una tarea específica, se mostrarán los detalles completos de la misma.

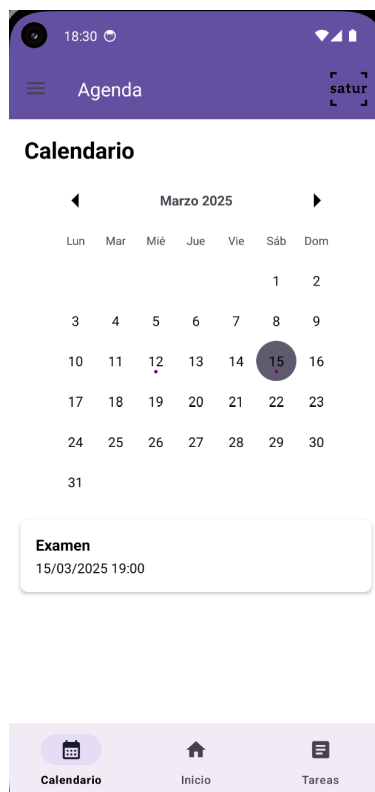


Figura 6. Pantalla de Calendario.



## Pantalla de Tareas

Esta pantalla ([ver Figura 7](#)) permite gestionar todas las tareas pendientes de una forma sencilla:

- Se presenta una lista ordenada cronológicamente con las tareas existentes.
- Para crear una nueva tarea, el usuario debe pulsar el botón circular ubicado en la esquina inferior derecha con el símbolo "+". Se abrirá un diálogo ([ver Figura 8](#)) a rellenar para agregar una tarea nueva. Al confirmar pulsando **"Agregar"**, la tarea queda almacenada y programada para notificar al usuario una hora antes de su vencimiento.
- Al pulsar sobre una tarea, se abre un diálogo mostrando los detalles ([ver Figura 9](#)), donde se puede:
  - **Añadir al calendario personal del dispositivo** la tarea seleccionada.
  - **Eliminar la tarea** del listado.

Además, incluye un botón flotante en la esquina inferior derecha para agregar nuevas tareas.



Figura 7. Pantalla tareas.

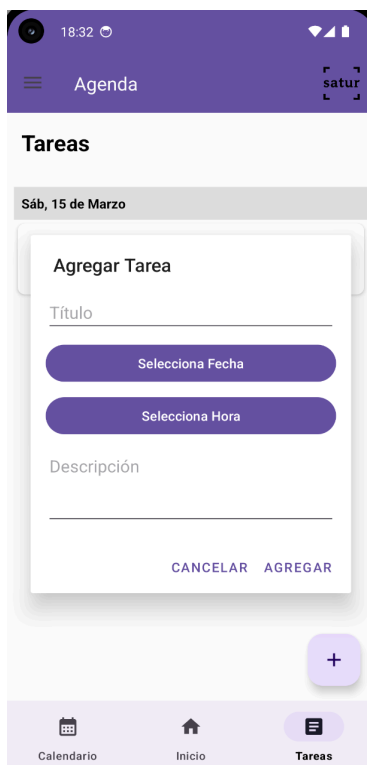


Figura 8. Añadir tarea.

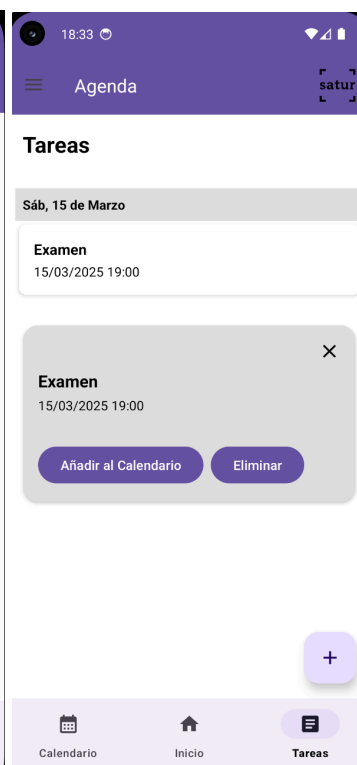


Figura 9. Detalles de la Tarea.

## 5. Dificultades encontradas

### Gestión dinámica del idioma y tema visual

- Complejidad al cambiar dinámicamente idiomas y aplicar inmediatamente los cambios en la interfaz de usuario.
- Dificultades para evitar que la actividad se reiniciara constantemente tras los cambios, lo que requería una gestión cuidadosa mediante `SharedPreferences` y configuraciones de la aplicación.

### Manejo de permisos en tiempo de ejecución

- Implementación de la solicitud dinámica de permisos para notificaciones y acceso al calendario, especialmente debido a restricciones y cambios en políticas introducidas en versiones recientes de Android.
- Problemas iniciales con la gestión de distintos estados de permisos (concedidos, rechazados, permanentemente rechazados).

### Integración con el calendario del dispositivo

- Desafío en la implementación adecuada del intent `ACTION_INSERT` para insertar eventos directamente en el calendario nativo, especialmente gestionando distintos comportamientos según fabricantes y versiones de Android.
- Gestión de posibles errores cuando no hay ninguna aplicación compatible instalada.

### Uso avanzado del componente `MaterialCalendarView`

- Dificultad inicial para aplicar decoradores personalizados para señalar visualmente los días con tareas existentes en la base de datos.
- Integración eficiente del calendario interactivo con Room, asegurando rendimiento fluido al cargar eventos y tareas desde la base de datos local.

### Persistencia eficiente con Room

- Manejo correcto de tipos complejos (como fechas) mediante conversores personalizados, inicialmente causando errores debido a conversiones incorrectas o falta de definiciones adecuadas.
- Problemas de rendimiento al cargar datos inicialmente en hilos separados, especialmente al manejar grandes cantidades de datos simultáneamente.

### Notificaciones mediante `WorkManager` y `AlarmManager`

- Inicialmente fue complicado coordinar correctamente el uso de `AlarmManager` y `WorkManager` para notificaciones, causando retrasos o notificaciones fallidas.
- Complejidad adicional al gestionar la cancelación automática de notificaciones al eliminar tareas, requiriendo una correcta gestión de `PendingIntents`.

### **Gestión eficiente de diálogos personalizados**

- Problemas iniciales al aplicar esquinas redondeadas a todos los diálogos de forma consistente, requiriendo la creación y aplicación cuidadosa de estilos personalizados globales.

### **Soporte multi-idioma completo**

- Necesidad de mantener sincronizados múltiples recursos (strings.xml y arrays.xml) en distintos idiomas, lo que incrementó la posibilidad de errores de traducción o referencias incorrectas entre versiones.

Estas dificultades fueron abordadas satisfactoriamente mediante la documentación oficial de Android, consultas en foros técnicos como Stack Overflow y pruebas extensivas en múltiples configuraciones y versiones del sistema operativo.

## 6. Fuentes utilizadas

### 1. Documentación oficial de Android

- Guías para desarrolladores Android.
  - Manejo de permisos en tiempo de ejecución.
  - Uso de preferencias compartidas (SharedPreferences).
  - Implementación de navegación con NavigationView y BottomNavigationView.

### 2. Android Jetpack: Room Persistence Library

- Documentación oficial sobre la biblioteca Room.
  - Creación y definición de entidades, DAO y consultas avanzadas.
  - Implementación de conversores personalizados para tipos de datos complejos.

### 3. MaterialCalendarView

- Documentación y repositorio oficial.
  - Configuración del calendario interactivo.
  - Creación y aplicación de decoradores para eventos personalizados.

### 4. WorkManager y AlarmManager

- Documentación oficial sobre WorkManager (parte de Android Jetpack).
  - Gestión eficiente de tareas en segundo plano y notificaciones programadas.
- Documentación oficial sobre AlarmManager.
  - Configuración de alarmas para notificaciones precisas en fechas y horas determinadas.

### 6. Soporte multi-idioma

- Documentación oficial sobre internacionalización y localización en Android.
  - Organización estructurada de recursos para diferentes idiomas.
  - Mejores prácticas para soporte multi-idioma en aplicaciones Android.