

eman ta zabal zazu



Universidad  
del País Vasco      Euskal Herriko  
                              Unibertsitatea

# Memoria de Proyecto: SkyVisit

Aitor Gonzalo

22 de julio de 2025

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Repositorio de Código</b>	<b>3</b>
<b>3. Elementos de Valoración y Funcionalidades</b>	<b>3</b>
3.1. Elementos obligatorios (requisitos mínimos) . . . . .	3
3.2. Elementos opcionales (funcionalidades avanzadas) . . . . .	4
<b>4. Descripción de Clases y Bases de Datos</b>	<b>5</b>
4.1. Arquitectura . . . . .	5
4.2. Clases del proyecto . . . . .	6
4.3. Diagrama de la Base de Datos . . . . .	9
<b>5. Manual de Usuario</b>	<b>10</b>
<b>6. Dificultades Encontradas</b>	<b>16</b>
<b>7. Fuentes Utilizadas</b>	<b>16</b>
<b>8. Conclusiones</b>	<b>17</b>

# Índice de figuras

1.	Arquitectura de la Aplicación . . . . .	6
2.	Diagrama de Clases de la Aplicación . . . . .	8
3.	Pantalla Bienvenida . . . . .	10
4.	Pantalla Inicio . . . . .	11
5.	Cercanía . . . . .	11
6.	Wifi . . . . .	11
7.	Pantalla Mapa . . . . .	12
8.	Añadir a Galería . . . . .	12
9.	Galería . . . . .	13
10.	Pantalla Añadir Lugar . . . . .	13
11.	Método Foto . . . . .	14
12.	Datos Completos . . . . .	14
13.	Widget1 . . . . .	15
14.	Widget2 . . . . .	15

# 1. Introducción

Esta memoria documenta el desarrollo de una aplicación móvil nativa para Android, desarrollada con Android Studio. El tema de la aplicación es **Lugares de Interés**; en ella podrás ver la localización de distintos sitios relevantes. La aplicación permite visualizar la ubicación de cada lugar en un mapa interactivo, acceder a detalles específicos de cada sitio y gestionar la información mediante una base de datos remota(Mysql+PHP). El objetivo principal ha sido poner en práctica conocimientos avanzados de desarrollo móvil, tales como la integración con servicios web, la gestión de permisos y recursos del sistema, el uso de servicios en segundo plano, widgets, notificaciones y content providers. Además, se ha buscado ofrecer una experiencia de usuario fluida y una interfaz atractiva. La aplicación está desarrollada en Android Studio, orientada a dispositivos con Android 10 (API 29) o superior.

## 2. Repositorio de Código

El proyecto completo se encuentra disponible en el siguiente repositorio:

[https://github.com/aitorGonzalo/SkyVist\\_v2](https://github.com/aitorGonzalo/SkyVist_v2)

## 3. Elementos de Valoración y Funcionalidades

La aplicación incorpora una serie de elementos que permiten cumplir tanto con los requisitos mínimos obligatorios, como con funcionalidades avanzadas que aportan valor añadido. A continuación, se describen estos elementos, indicando para qué se utilizan y su categorización.

### 3.1. Elementos obligatorios (requisitos mínimos)

- **Base de datos remota para registro e identificación de usuarios.**  
Implementado mediante un servidor en AWS que ejecuta scripts PHP conectados a una base de datos MySQL. La aplicación permite registrar y autenticar usuarios, guardando el identificador en `SharedPreferences`.
- **Integración de geolocalización y mapas.**  
Se utiliza **OpenStreetMap** (mediante la librería `osmdroid`) para visualizar lugares de interés en un mapa. La ubicación actual del usuario se obtiene mediante `FusedLocationProviderClient`, gestionando permisos en tiempo de ejecución.
- **Captura de imágenes, subida al servidor y visualización.**  
El usuario puede tomar fotos desde la cámara o seleccionarlas desde la galería. Las imágenes se convierten en archivos temporales y se suben mediante una petición `multipart/form-data`. Posteriormente, se muestran con la librería `Picasso` en una lista de lugares.

### 3.2. Elementos opcionales (funcionalidades avanzadas)

- **Uso de Content Provider.**

Se utiliza `MediaStore.Images.Media` para realizar operaciones de inserción, consulta y eliminación de imágenes, tanto desde la actividad principal como desde el widget.

- **Servicio en primer plano y gestión de broadcast.**

La aplicación incluye un servicio en primer plano (`NetworkMonitorService`) que detecta pérdida de conexión a Internet. Utiliza un `BroadcastReceiver` para mostrar una notificación cuando se pierde la conectividad.

- **Widget con actualización automática.**

Se ha desarrollado un widget que muestra el número de lugares guardados y el total de imágenes almacenadas. Este widget se actualiza cada 30 segundos mediante una alarma programada.

- **Alarma programada (no perteneciente al widget).**

La aplicación programa una alarma de cercanía que se ejecuta mientras la app está abierta. La alarma de cercanía se programa con `AlarmManager` para ejecutarse cada 10 segundos. En cada activación, se obtiene la ubicación actual del usuario mediante `FusedLocationProviderClient`, y se consulta desde el servidor la lista de lugares registrados. Para cada lugar, se calcula la distancia con `Location.distanceBetween(...)`. Si la distancia es menor a 100 metros, se lanza una notificación advirtiendo que el usuario está cerca de ese lugar.

## 4. Descripción de Clases y Bases de Datos

### 4.1. Arquitectura

La aplicación sigue un enfoque basado en la Arquitectura en Tres Capas (Three-Tier Architecture), lo que permite separar la presentación, la lógica de negocio y el acceso a datos. Esta separación facilita la mantenibilidad, la escalabilidad y la modularidad del sistema.

#### 1. Capa de Presentación (Interfaz de Usuario)

Esta capa se encarga de mostrar información al usuario y recoger sus interacciones.

- Compuesta por las **Activities** principales: `MainActivity`, `LugaresActivity`... entre otras.
- Incluye los archivos XML que definen la interfaz de usuario: botones, formularios, listas, mapas, etc.
- El `SkyVisitWidget` forma parte también de esta capa como una interfaz adicional desde el escritorio del dispositivo.

#### 2. Capa de Lógica de Negocio

Encargada de procesar datos, aplicar reglas y coordinar las funcionalidades.

- La clase `LugaresAlarmReceiver` gestiona la alarma de cercanía y lanza notificaciones.
- El servicio en primer plano `NetworkMonitorService` detecta pérdidas de conexión a Internet.
- Las actividades también contienen controladores de eventos (listeners), validaciones y gestión de flujos.
- Se coordina la subida de imágenes, la obtención de la ubicación y el manejo de permisos en tiempo de ejecución.

#### 3. Capa de Acceso a Datos

Gestiona la interacción con el servidor y los recursos del sistema.

- Comunicación con el servidor web mediante peticiones HTTP (`HttpURLConnection`) a ficheros PHP.
- Acceso al sistema de archivos mediante el `Content Provider MediaStore.Images.Media` para guardar, contar y eliminar imágenes.
- Uso de `SharedPreferences` para almacenar configuraciones locales como el ID del usuario o estados de imágenes.

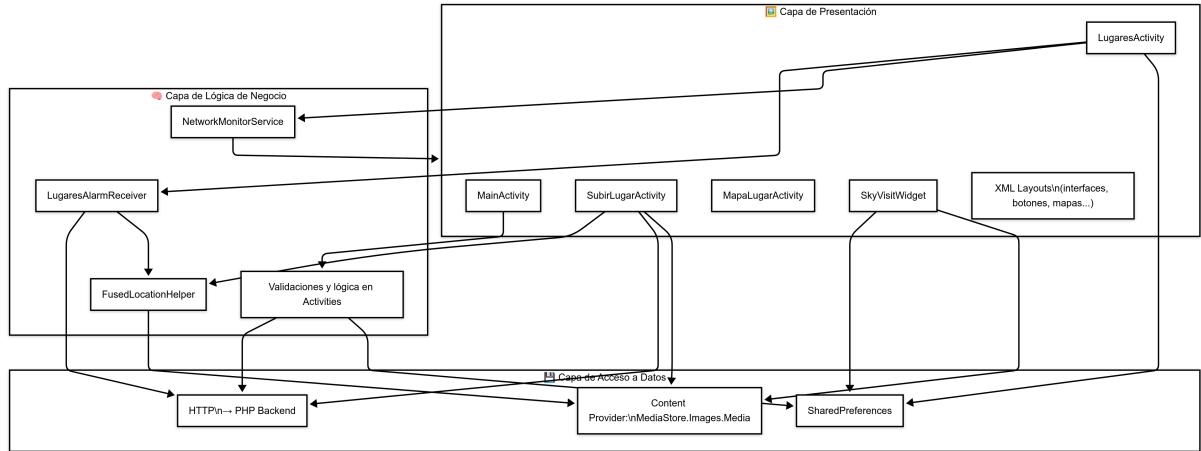


Figura 1: Arquitectura de la Aplicación

## 4.2. Clases del proyecto

A continuación se describen las clases principales de la aplicación **SkyVisit**, organizadas por su responsabilidad y función en la arquitectura.

**MainActivity** Esta clase extiende de `AppCompatActivity` y actúa como la pantalla de inicio de la aplicación. Es la encargada de gestionar el formulario de inicio de sesión y registro de usuario. Recoge los datos introducidos por el usuario (usuario y contraseña), los empaqueta en un objeto JSON y los envía al servidor mediante peticiones HTTP. En caso de éxito, guarda el ID del usuario en `SharedPreferences` para ser reutilizado más adelante y redirige a la actividad principal de la app: `LugaresActivity`.

**LugaresActivity** También hereda de `AppCompatActivity` y representa la pantalla principal tras el inicio de sesión. Muestra un listado de lugares turísticos recuperados desde el servidor y permite al usuario añadir nuevos lugares o ver imágenes guardadas. Utiliza un `RecyclerView` para mostrar la lista y se encarga de lanzar servicios como `NetworkMonitorService`, así como de programar la alarma de cercanía. Además, comproueba permisos y fuerza la actualización del widget mediante un broadcast.

**SubirLugarActivity** Extiende de `AppCompatActivity` y permite al usuario capturar o seleccionar una foto, obtener la ubicación actual y subir toda la información al servidor. Gestiona los permisos de cámara y localización, construye una petición `multipart/form-data` y sube la imagen junto con los datos del lugar. También valida que los campos estén completos antes de enviar la información.

**Lugar** Esta clase representa un modelo de datos que encapsula la información de un lugar turístico. Contiene los atributos nombre, URL de la foto, latitud y longitud. No extiende de ninguna clase personalizada y simplemente sirve como contenedor de datos para ser usado en listas o serializaciones.

**LugarAdapter** Es una clase que extiende de `RecyclerView.Adapter` y se encarga de adaptar la lista de objetos `Lugar` para ser mostrada en un `RecyclerView`. Gestiona la visualización de cada lugar, permite guardar o eliminar imágenes en la galería mediante el `ContentProvider` del sistema, y reacciona a los clics sobre los elementos para abrir un mapa con la ubicación del lugar.

**MapaLugarActivity** Clase que hereda de `AppCompatActivity` y muestra un mapa centrado en la ubicación de un lugar concreto. Utiliza la biblioteca `osmdroid` para renderizar el mapa y colocar un marcador sobre la ubicación recibida por `Intent`. Sirve como complemento visual para identificar rápidamente la localización del sitio fotografiado.

**FusedLocationHelper** Es una clase utilitaria que no extiende de ninguna clase específica de Android. Su propósito es abstraer el uso de `FusedLocationProviderClient` y proporcionar una forma sencilla de obtener la última ubicación conocida. Utiliza una interfaz interna para devolver el resultado de la ubicación al llamador.

**LugaresAlarmReceiver** Extiende de `BroadcastReceiver` y se activa mediante una alarma programada con `AlarmManager`. Cada vez que se lanza, obtiene la ubicación actual y la compara con las coordenadas de los lugares almacenados en el servidor. Si detecta que el usuario está cerca de alguno (a menos de 100 metros), lanza una notificación de proximidad usando el sistema de notificaciones de Android.

**NetworkMonitorService** Esta clase extiende de `Service` y se ejecuta en primer plano (*foreground*) para evitar que el sistema la detenga. Dentro de ella se registra un `BroadcastReceiver` que detecta cambios en la conectividad de red. En caso de pérdida de conexión, se muestra una notificación de advertencia al usuario. Incluye también una clase interna `MiBinder` para permitir su vinculación desde una `Activity`.

**SkyVisitWidget** Extiende de `AppWidgetProvider` y representa un widget de escritorio que muestra el número de lugares y de imágenes guardadas en la galería. Se actualiza de forma periódica mediante una alarma.

**SkyVisitWidgetUpdateReceiver** Clase sencilla que extiende de `BroadcastReceiver`. Su única función es recibir las alarmas periódicas programadas para el widget y reenviar un intent que obliga al sistema a actualizar su contenido.

A continuación se muestra un diagrama de clases que resume el flujo y la relación entre las clases de la aplicación:

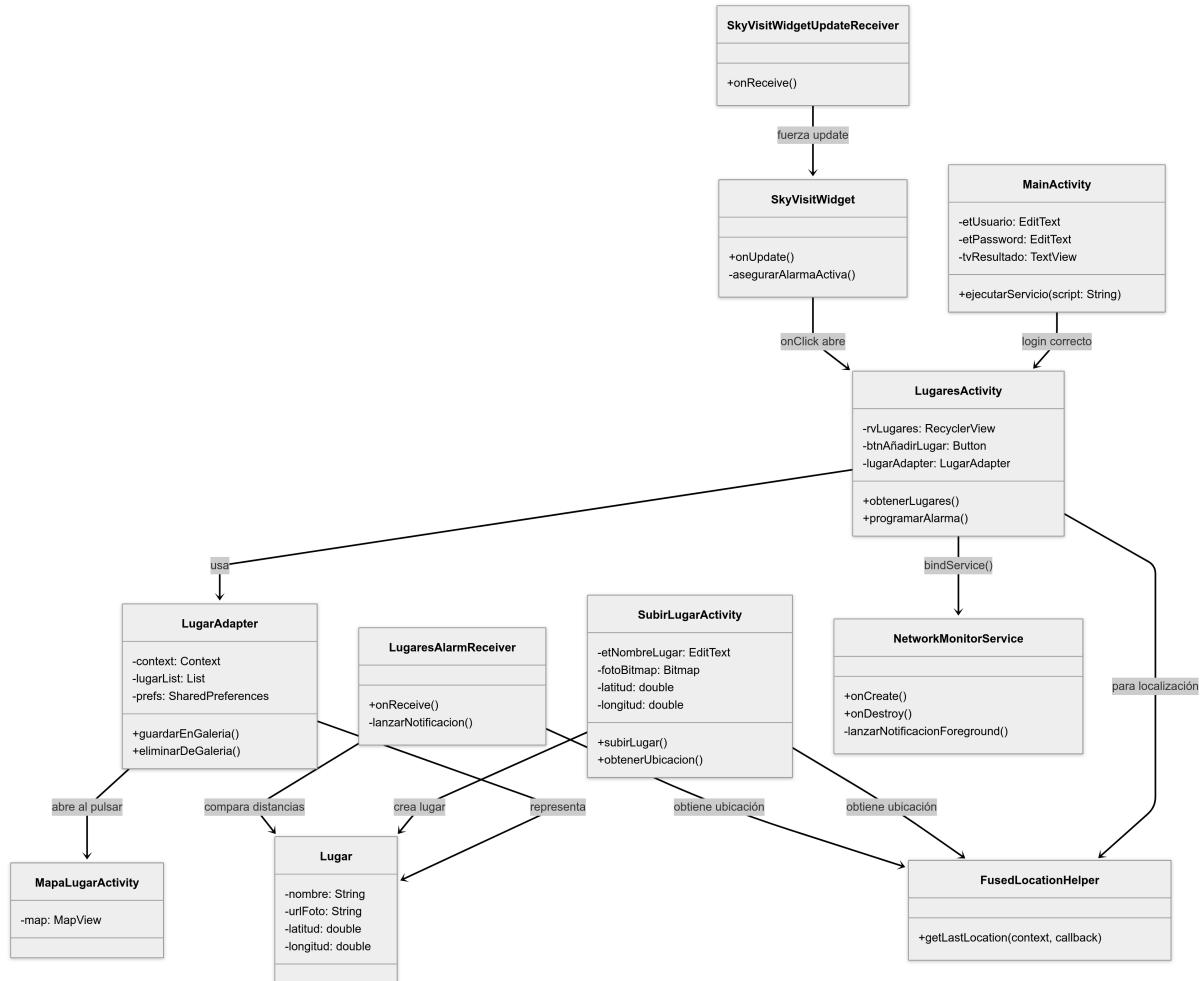


Figura 2: Diagrama de Clases de la Aplicación

### 4.3. Diagrama de la Base de Datos

La base de datos remota utilizada por la aplicación está alojada en un servidor web accesible a través de peticiones HTTP a ficheros PHP. Contiene una única tabla principal denominada **lugares**, donde se almacenan los lugares subidos por los usuarios. Cada registro representa un sitio visitado, incluyendo su nombre, ubicación geográfica, la ruta a la imagen asociada y el identificador del usuario que lo subió.

Campo	Tipo	Descripción
id	INTEGER (PRIMARY KEY AUTOINCREMENT)	Identificador único del lugar
nombre	TEXT NOT NULL	Nombre del lugar
latitud	REAL NOT NULL	Latitud del lugar
longitud	REAL NOT NULL	Longitud del lugar
foto	TEXT NOT NULL	Nombre de la foto
usuario_id	INTEGER NOT NULL	ID del usuario

Además, existe otra tabla auxiliar llamada **usuarios** para gestionar el acceso a la aplicación:

Campo	Tipo	Descripción
id	INTEGER (PRIMARY KEY AUTOINCREMENT)	Identificador único del usuario
usuario	TEXT NOT NULL UNIQUE	Nombre de usuario
password	TEXT NOT NULL	Contraseña

## 5. Manual de Usuario

Esta sección proporciona una guía práctica para los usuarios de la aplicación **SkyVisit**. El objetivo es facilitar el uso de las funcionalidades principales de la app, detallando los pasos necesarios para interactuar correctamente con la interfaz, registrar lugares de interés, consultar la galería, recibir notificaciones y aprovechar otras características disponibles.

El manual está orientado a usuarios finales con conocimientos básicos en el manejo de dispositivos Android, y explica de forma clara y accesible cada una de las pantallas y acciones que se pueden realizar en la aplicación.

Nada más entrar a la app vemos esta interfaz, si es la primera vez que accedemos nos registramos o se puede usar el usuario ya creado(aitor,1234). Si tratamos de registrarnos con un usuario ya creado mostrará un error por pantalla.

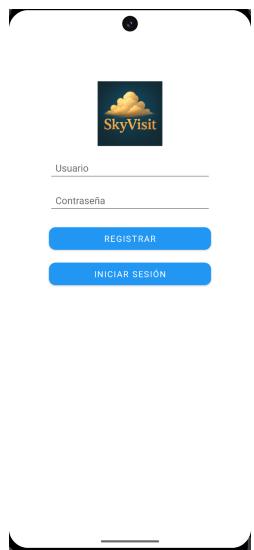


Figura 3: Pantalla Bienvenida

Al iniciar sesión, llegamos a la pantalla principal donde se listan todos los lugares y podemos llevar a cabo las diferentes acciones de la web.

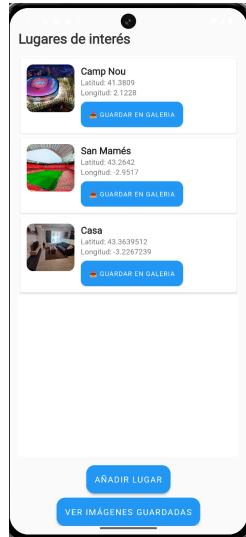


Figura 4: Pantalla Inicio

Antes de empezar con las acciones, muestro las notificaciones(fotos de dispositivo real). Si estamos cerca de algún lugar registrado, nos llegará esta notificación.

**Nota:** La alarma de cercanía está configurada para activarse únicamente mientras la aplicación está abierta, mientras que el widget y el control de conexión permanecen activos en todo momento.

En la Figura 5 podemos ver la alarma de cercanía y en la Figura 6 podemos ver la alarma que salta al monitorizar el wifi y cuando se desactiva.



Figura 5: Cercanía



Figura 6: Wifi

En la interfaz principal, si pulsamos en un lugar, nos lleva a su localización en el mapa.

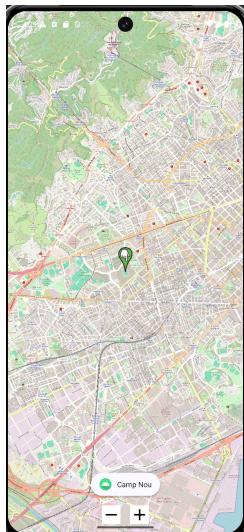


Figura 7: Pantalla Mapa

Si pulsamos en los botones de guardar en galería, cambiara su aspecto a eliminar de galería.

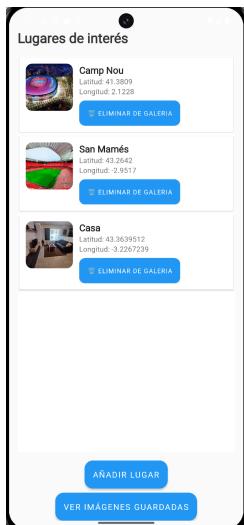


Figura 8: Añadir a Galeria

Al pulsar en el botón de ver imágenes en galería, se nos abre la galería y podemos ver las imágenes que hayamos guardado desde la interfaz de la web.

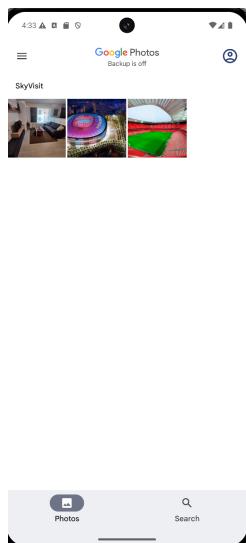


Figura 9: Galería

Si deseamos añadir un lugar nuevo, damos al botón de añadir lugar y nos lleva a esta interfaz.



Figura 10: Pantalla Añadir Lugar

Desde aquí podemos añadir una imagen, ya sea por cámara o desde la galería 11, obtener nuestra ubicación actual, establecer el nombre del lugar 12 y subirlo al servidor y base de datos.



Figura 11: Método Foto

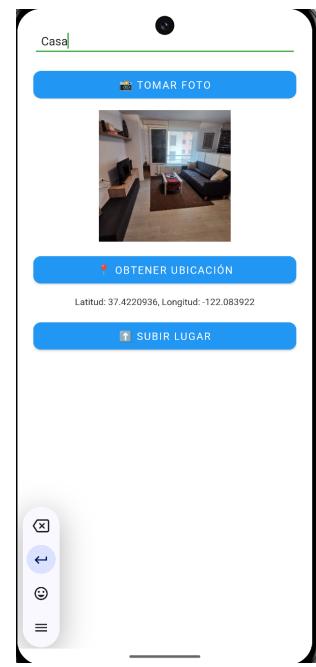


Figura 12: Datos Completos

Por ultimo,en el widget podemos ver como se van actualizando el numero de lugares de la base de datos y el numero de elementos guardados en galería. Si probamos a eliminar de galeria las fotos de los 3 lugares que hemos añadido antes, veremos como se actualiza al de 30 segundos aproximadamente.

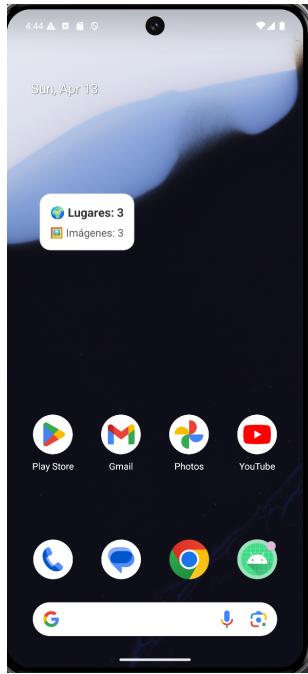


Figura 13: Widget1

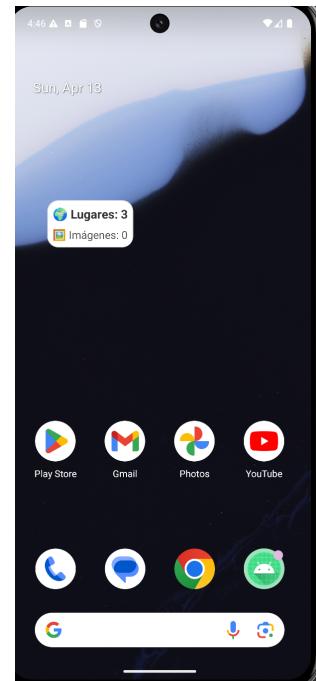


Figura 14: Widget2

## 6. Dificultades Encontradas

Durante el desarrollo de la aplicación **SkyVisit**, se presentaron varios desafíos técnicos, especialmente debido a la cantidad de funcionalidades obligatorias que debían implementarse y coordinarse entre sí.

Una de las principales dificultades fue la implementación del **servicio en primer plano** para la monitorización de red, ya que implicaba gestionar correctamente el ciclo de vida del servicio, crear canales de notificación, y registrar correctamente el **BroadcastReceiver** para detectar la pérdida de conexión. Además, se debía garantizar que este servicio se ejecutara en segundo plano sin ser detenido por el sistema.

También supuso un reto la gestión de permisos en tiempo de ejecución, especialmente en versiones recientes de Android (como Android 13+), que requieren permisos explícitos para mostrar notificaciones. Esto afectaba tanto al servicio como a la **alarma de cercanía**, ya que si el permiso era denegado, las notificaciones no se mostraban aunque la lógica funcionara correctamente.

Otra dificultad importante fue el uso del **MediaStore** para cumplir con el requisito de utilizar un **Content Provider** para operaciones CRUD. El guardado y eliminación de imágenes desde la galería implicaba manejar correctamente los **ContentValues**, rutas relativas, y comprobar los permisos necesarios, lo que requirió múltiples pruebas.

Finalmente, la implementación del **widget** y su actualización periódica también requirió tiempo y experimentación. Fue necesario coordinar correctamente la lectura de datos desde las **SharedPreferences** y el acceso a la galería mediante queries al **ContentProvider**, además de configurar adecuadamente la **AlarmManager** para forzar su actualización cada 30 segundos.

## 7. Fuentes Utilizadas

Las siguientes fuentes han sido de gran ayuda durante el desarrollo del proyecto:

- **Documentación Oficial de Android:** [3]
- **Tutoriales y Blogs Especializados:** Recursos en línea sobre Widget, conexiones remotas, servicios... [2].
- **Foros de Desarrollo:** Stack Overflow, GitHub y otras comunidades que han aportado soluciones a problemas encontrados [4].
- **Egela:** Se han usado los apuntes del contenido visto en clase así como los guiones de laboratorios y recursos de ayuda para la documentación [1]

## 8. Conclusiones

El desarrollo de **SkyVisit** ha supuesto un gran paso adelante en mi formación como desarrollador de aplicaciones Android. A lo largo del proyecto, he consolidado conocimientos fundamentales sobre el ciclo de vida de las actividades, la gestión de permisos en tiempo de ejecución y el trabajo con interfaces diseñadas en XML. Además, he tenido la oportunidad de integrar funcionalidades más avanzadas y propias de aplicaciones reales del mercado.

Entre los aprendizajes más destacables se encuentra la implementación de un **servicio en primer plano**, el uso de **BroadcastReceiver** para alarmas y notificaciones, la manipulación de imágenes mediante el **ContentProvider MediaStore**, y la integración de un **widget** funcional con actualización periódica automatizada. También he profundizado en la comunicación con un **servidor remoto**, la subida de imágenes al backend, y el uso de librerías externas como **Picasso** y **osmdroid** para mejorar la experiencia del usuario.

Gracias a este proyecto he aprendido a coordinar múltiples componentes de Android en un mismo flujo de trabajo, enfrentándome a problemas reales de integración, compatibilidad entre versiones del sistema, y buenas prácticas en el desarrollo móvil.

## Referencias

- [1] Apuntes. <https://egela.ehu.eus/course/view.php?id=96452&section=0#tabs-tree-start>. Consultado el 7-15 de abril de 2025.
- [2] Como crear un widget desde cero. <https://www.youtube.com/watch?v=GkyG1KftQWw>. Consultado el 12 de abril de 2025.
- [3] Documentación oficial de android. <https://developer.android.com/>. Consultado el 7-15 de abril de 2025.
- [4] Stack overflow. <https://stackoverflow.com/>. Consultado el 7-15 de abril de 2025.