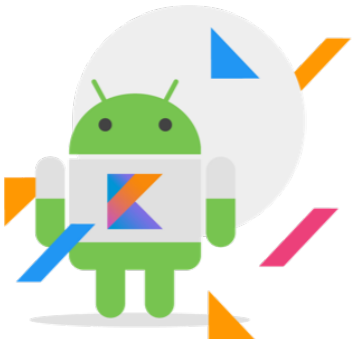
IES DR. LLUÍS SIMARRO

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

CURS 2025-2026

UD9: Persistencia de datos II: Base de datos local con Room



Profesorado	
Ana Sanchis	ag.sanchisperales@edu.gva.es
Juan Mateu	j.mateuruzafa@edu.gva.es
Joan Carrillo	jr.carrilloforonda@edu.gva.es

UD09 -> T9_A2. PERSISTENCIA DE DATOS II: BASE DE DATOS LOCAL CON ROOM

Instrucciones iniciales

- Las tareas se subirán a la plataforma Aules en la fecha establecida y con el formato indicado en los siguientes puntos.
- A partir de estas prácticas, se recomienda enviar la URL al repositorio de **GitHub** con la práctica solicitada.
- En caso de no subirlo a través de Git/GitHub, se subirá un fichero zip con el siguiente nombre:

T9a2-PrimerApellido-Nombre.zip

Objetivos de la actividad

1. Aprender la arquitectura y funcionamiento de Room como una capa de abstracción sobre SQLite.
2. Implementar bases de datos locales en aplicaciones Android utilizando Room y Kotlin.

Temporalización

La tarea tiene una estimación aproximada de tres sesiones lectivas de 55 minutos cada sesión.

BASE DE DATOS CON ROOM: AGENCIA DE VIAJES OFFLINE

BASE DE DATOS - Agenda de Viajes Offline"

1. DESCRIPCIÓN DEL PROYECTO

El objetivo es desarrollar una aplicación Android nativa utilizando **Kotlin** que permita a los usuarios gestionar sus futuros viajes. La aplicación debe funcionar sin conexión a internet (**Offline First**), almacenando toda la información en una base de datos local relacional (**SQLite**) gestionada mediante la librería **Android Room**.

2. ARQUITECTURA DEL PROYECTO (Obligatorio)

El alumno deberá respetar estrictamente la siguiente estructura de paquetes y archivos. Cualquier desviación en la ubicación de los archivos penalizará la nota.

📂 Paquete com.example.appviatges.data

Este paquete contendrá exclusivamente la lógica de base de datos:

1. **AppDatabase.kt**: Clase abstracta que hereda de RoomDatabase.
2. **ViajeAplication**.kt Heredará de Application. Debe implementar el patrón **Singleton**. Sobreescribiremos el onCreate
3. **Destino.kt**: Entidad (Tabla) con los campos idDestino (PK), nombre y pais.
4. **Viaje.kt**: Entidad (Tabla) con los campos idViaje (PK), titulo, tipo, descripcion y destinoId.
    - *Nota:* Está **prohibido** guardar el nombre del destino como texto en la tabla de viajes. Se debe usar una **Clave Foránea** (Foreign Key).
5. **ViajeConDestino.kt**: Clase de datos para modelar la relación 1:N. Debe usar las anotaciones @Embedded y @Relation.
6. **DestinoDao.kt**: Interfaz con operaciones CRUD básicas (Insert, Delete, Query).
7. **ViajeDao.kt**: Interfaz con operaciones CRUD y una consulta COUNT para verificar integridad.
8. **ViajeDestinoDao.kt**: Interfaz que contiene la consulta @Transaction para obtener la lista combinada de viajes con sus destinos.

📂 Paquete com.example.appviatges.ui

La interfaz se dividirá en dos sub-paquetes funcionales:

📁 ui.destinos

- **DestinosFragment.kt**: Pantalla que muestra el catálogo de destinos disponibles.

- **DestinosAdapter.kt**: Adaptador para el RecyclerView de destinos.

📁 **ui.viajes**

- **ViajesListFragment.kt**: Pantalla principal que muestra la lista de viajes creados.
- **ViajeFormFragment.kt**: Pantalla de formulario para crear un nuevo viaje.
- **ViajesAdapter.kt**: Adaptador para el RecyclerView de viajes. **Importante:** Este adaptador debe recibir una lista de objetos ViajeConDestino.

---

3. REQUISITOS FUNCIONALES

A. Gestión de Destinos (Catálogo)

- **Visualización:** Lista de destinos con imagen (usar librería externa como Picasso y nombre.
- **Integridad Referencial (Borrado Seguro):**
  - Al intentar borrar un destino, la aplicación **NO** debe preguntar al usuario, sino a la base de datos.
  - Se debe consultar si existen viajes asociados a ese destino.
  - **Si existen viajes:** Mostrar un error (Toast) indicando el número de viajes afectados y **bloquear** el borrado.
  - **Si no existen:** Eliminar el destino directamente.

B. Gestión de Viajes (Principal)

- **Listado:**
  - **Visualización:** Lista de destinos con imagen (usar librería externa como Picasso y nombre).
  - Las tarjetas de viaje deben mostrar el título y el **nombre del destino** (obtenido mediante la relación entre tablas) así como el tipo de viaje que es y la fecha del viaje.

  - Diseño cuidado utilizando MaterialCardView y ConstraintLayout.

- **Creación (Formulario):**
  - Debe incluir una **imagen de cabecera** en la parte superior.
  - **Selector de Destino:** El usuario debe seleccionar el destino mediante un componente **Spinner**.
  - El Spinner debe cargarse dinámicamente con los datos de la tabla destinos.
- **Borrado:**
  - Al eliminar un viaje, se debe solicitar confirmación explícita mediante un **AlertDialog** del sistema.

UD09 -> T9_A2. PERSISTENCIA DE DATOS II: BASE DE DATOS LOCAL CON ROOM

Capturas de pantalla