ULPGC

Escuela de
Ingeniería Informática

# *Programación de Aplicaciones Móviles Nativas*

## Práctica 5: Diseño de la Base de Datos

Guillermo Marion López,
Karl Christian Deilmann,
Louka David Vanhoucke

**1.- Revisa con tu equipo la arquitectura de tu proyecto final, identifica y resalta los modelos y componentes relacionados con la capa de persistencia.**

**Modelos**

- **Ruta:** Representa una ruta que ha sido rastreada por un usuario. Incluye información como la fecha y hora de inicio y finalización, la duración, la distancia recorrida y los puntos de interés visitados.
- **Ubicación destacada:** Representa una ubicación que ha sido destacada por un usuario. Incluye información como la ubicación geográfica, el nombre, la descripción y las imágenes.
- **Feedback:** Representa un comentario o sugerencia sobre un lugar. Incluye información como el nombre del usuario, la fecha, el contenido del feedback y la calificación.

**Componentes**

- **Repositorio:** Es un componente que proporciona acceso a los datos almacenados en la capa de persistencia.
- **Servicio de persistencia:** Es un componente que implementa las operaciones de persistencia, como el almacenamiento, la recuperación, la actualización y la eliminación de datos.

**2.- Selecciona una tecnología para la base de datos de tu aplicación (Room, Firebase, SQLite, Realm, DataStore ...etc). Justifica brevemente tu elección, considerando las necesidades de tu aplicación.**

Para poder elegir una tecnología para la base de datos de nuestra aplicación, tenemos que saber que es realmente cada tecnología lo describimos brevemente abajo

**Firebase:** Firebase es una plataforma de desarrollo de aplicaciones en la nube de Google que ofrece una variedad de servicios, incluyendo una base de datos en tiempo real, autenticación, almacenamiento en la nube, análisis y más, diseñada para facilitar el desarrollo y escalabilidad de aplicaciones móviles y web.

**SQLite:** SQLite es una base de datos relacional ligera y sin servidor que se utiliza en aplicaciones locales para el almacenamiento de datos en dispositivos móviles y de escritorio. Es ampliamente utilizado en aplicaciones móviles y admite consultas SQL.

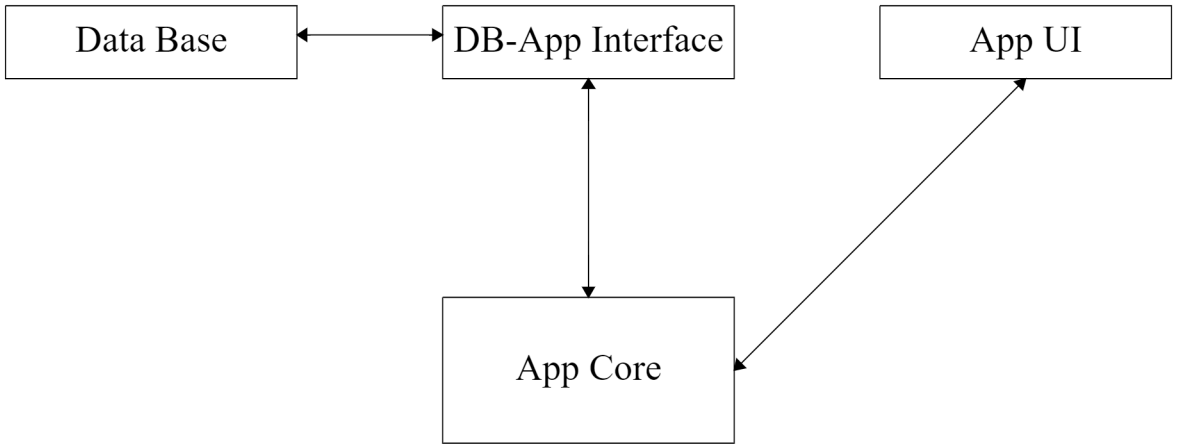
**Realm:** Realm es una base de datos orientada a objetos diseñada para aplicaciones móviles. Proporciona una API de alto rendimiento y un modelo de datos más intuitivo en comparación con SQLite y se sincroniza en tiempo real en dispositivos.

**DataStore:** DataStore es una biblioteca de Jetpack en Android que proporciona una forma eficiente y segura de almacenar datos en dispositivos móviles utilizando una arquitectura basada en objetos y una API de tipo clave-valor. Está diseñada para reemplazar SharedPreferences y es adecuada para aplicaciones Android modernas.

Hemos llegado a la conclusión de usar **Firebase** para nuestra aplicación debido a su sincronización en tiempo real, escalabilidad y amplio conjunto de funciones, lo que simplifica el desarrollo y ofrece una experiencia de usuario mejorada en nuestra aplicación. Además podemos añadir que los miembros del proyecto tienen conocimientos y experiencia previa con Firebase.

**3.- Define cómo se integrará la lógica que gestionará la capa de persistencia en tu proyecto. Puedes complementar el diagrama empleado en la actividad sobre la arquitectura de tu aplicación. Esto puede incluir entidades, atributos, relaciones, DAOs, repositorios.**
La lógica de persistencia se integrará en el núcleo de la aplicación. El núcleo de la aplicación utilizará un API de acceso a datos (DAO) para acceder a la base de datos. La interfaz de la base de datos a la aplicación (DB-App Interface) proporcionará un mecanismo para que la aplicación interactúe con el DAO.

En resumen, la lógica de persistencia se implementará como un DAO que proporcionará métodos para realizar operaciones de almacenamiento y recuperación de datos. La aplicación utilizará la interfaz DB-App Interface para interactuar con el DAO.




**4.- Reflexiona sobre cambios futuros:**
**- ¿Qué pasa si en un futuro se quisiera cambiar el motor de base de datos?**
Cambiar el motor de base de datos en el futuro requeriría modificar las consultas y la lógica de acceso a la base de datos en la aplicación, por lo tanto es importante elegir en el principio el motor de la base de datos

**- ¿Qué partes de tu aplicación tendrías que modificar?**
Las dificultades anticipadas incluyen la necesidad de migrar datos existentes y garantizar la compatibilidad con el nuevo motor, ya que no todos los motores funcionan de la misma manera.

**- ¿Qué dificultades anticipas?**
Algunas dificultades anticipadas al cambiar el motor de base de datos pueden incluir la necesidad de migrar datos existentes en la base de datos, ajustar consultas y adaptar la lógica de acceso a la nueva tecnología, lo que puede ser un proceso complejo y propenso a errores.

**- ¿Cómo podrías diseñar tu aplicación para minimizar el impacto de tal cambio?**
Usar interfaces para el acceso al base de datos => solo hace falta a modificar los interfaces en caso de cambio de base de datos