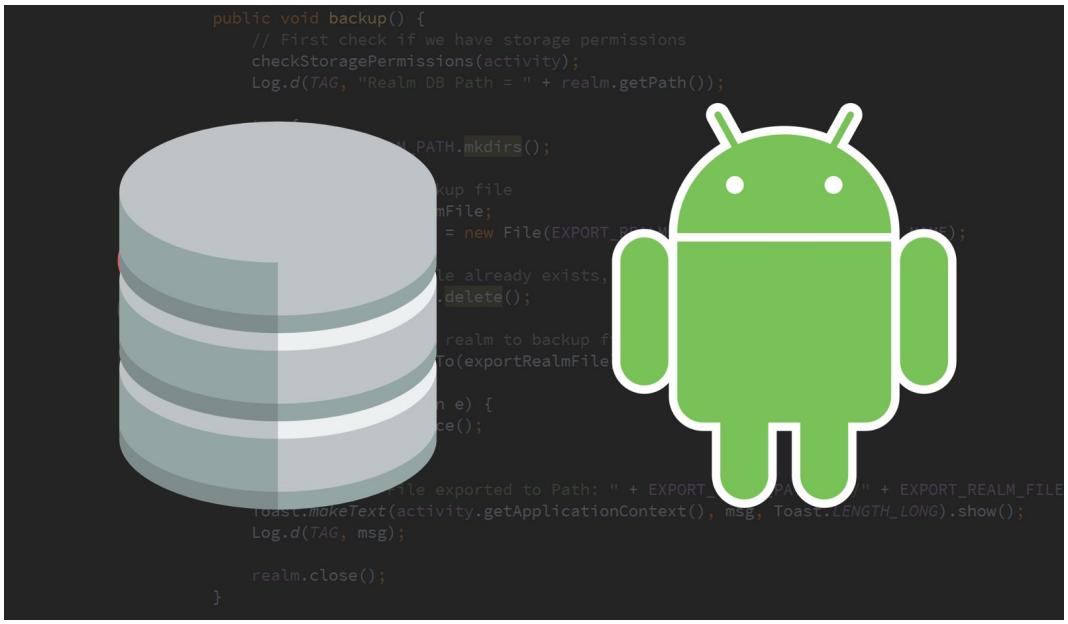
ULPGC · UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Práctica 6

Diseño de la Base de Datos



Programación de Aplicaciones Móviles Nativas
05 de Noviembre de 2023

Autores:

Ana del Carmen Santana Ojeda (ana.santana152@alu.ulpgc.es)

Alejandro David Arzola Saavedra (alejandro.arzola101@alu.ulpgc.es)

Índice

1. Introducción

En esta actividad, hemos revisado **la arquitectura del proyecto**, destacando modelos y componentes de la **capa de persistencia**. Tambien eligimos una tecnología para la base de datos **justificando la elección**.

Definiremos **la integración de la lógica** de persistencia en el proyecto, detallando entidades, atributos, relaciones etc...

Reflexionamos sobre cambios futuros, como la posibilidad de cambiar el motor de la base de datos. **Identificamos partes a modificar, para anticiparnos a dificultades** y propúsimos estrategias para minimizar el impacto de tales cambios en la aplicación. Este análisis garantizará la **adaptabilidad del proyecto a futuras evoluciones tecnológicas**.

Imagen 1: Persistencia en Android

2. Arquitectura del proyecto final

En cuanto a la capa de persistencia en nuestra aplicación musical, la gestión segura y eficiente de **la autenticación de usuarios es crucial** para proporcionar una **experiencia personalizada y segura**.

Para abordar esta necesidad, hemos decidido que lo integraremos con **Firebase Authentication**, una solución de **autenticación en la nube** proporcionada por **Firebase**, como el componente de persistencia clave en las vistas de login y registro.

Hemos seleccionado Firebase como nuestra base de datos por su **facilidad de uso y la familiaridad que nuestro equipo tiene con esta tecnología**. La elección de Firebase se fundamenta en la simplicidad que ofrece, lo cual facilita significativamente el desarrollo y la gestión de la base de datos.

Imagen 2: Componentes relacionados con la capa de persistencia

3. Tecnología para la base de datos

Hasta el momento, hemos decidido utilizar **Firebase como nuestra base de datos principal en la nube**. Esto permitirá a los usuarios **registrarse y almacenar información** o enviar notificaciones relacionada con las canciones.

Imagen 3: Firebase

Además, hemos optado por emplear **Room para gestionar las diversas listas de favoritos y 'me gusta' del usuario que haya iniciado sesión**. Estos datos se almacenarán **localmente para garantizar un acceso rápido y eficiente**. La sincronización periódica con Firebase facilitará la **actualización de estos datos entre el almacenamiento local y la nube**.

Imagen 4: Room

4. Integración de la lógica que gestionará la capa de persistencia

En cuanto a la lógica que gestiona la persistencia, específicamente en lo que respecta al **registro y login de la aplicación**, es necesario tener en cuenta diversas partes:

Entidades: Definimos las entidades que representarán los datos relevantes para el **módulo de autenticación**, como por ejemplo: **Usuario**.

DAO: Se crearan los DAOs para realizar **operaciones de acceso a datos** relacionadas con la autenticación.

Repositorios: Se implementaran repositorios que actúen como **capa intermedia entre los DAOs y la lógica de la aplicacion**.

ViewModels: Se utilizaran ViewModels para **gestionar la lógica y la comunicación entre la interfaz de usuario y los repositorios**.

Este enfoque modular permite una **separación clara de responsabilidades**.

5. Diagrama de la arquitectura de la aplicacion

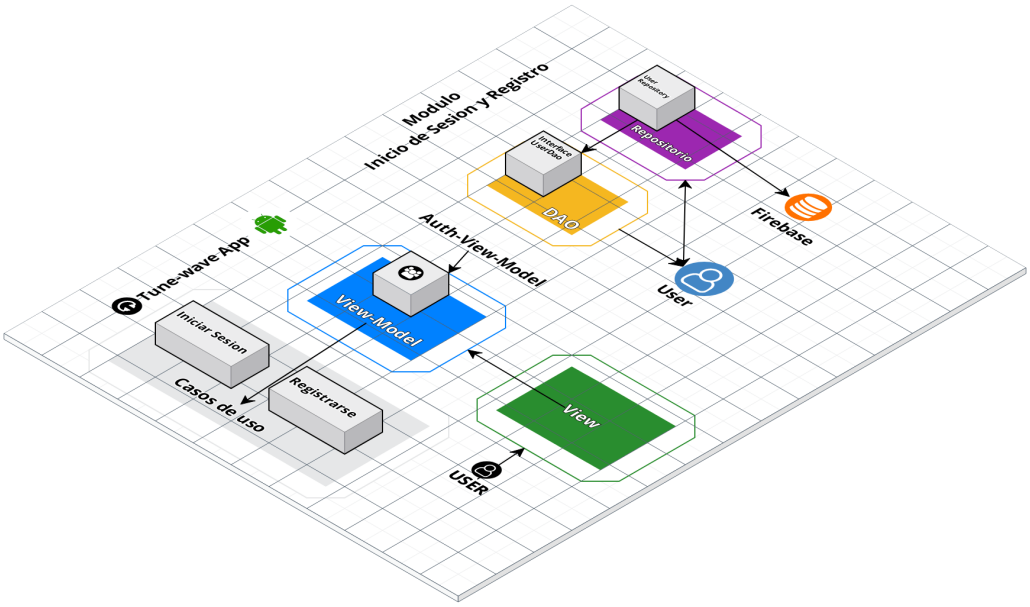


Imagen 5: Arquitectura de los modulos de la aplicacion

6. Reflexion de cambios futuros

6.1. ¿Qué pasa si en un futuro se quisiera cambiar el motor de base de datos?

Si se cambiara el motor de base de datos, es posible que tengamos algunos desafíos. La transición de un motor a otro podría requerir **cambios significativos en la lógica de persistencia** de la aplicación y **afectar a varias capas de la arquitectura**.

6.2. ¿Qué partes de tu aplicación tendrías que modificar?

Las áreas principales que podrían requerir modificaciones incluyen:

Capa de Datos:

Los DAOs y la lógica de **acceso a datos específicos del motor de base de datos actual**. La configuración de la conexión con el motor de base de datos actual.

Repositorios:

El repositorio que utiliza las **operaciones específicas del motor de base de datos** actual.

Entidades:

Si hay diferencias en la estructura de datos admitida por el nuevo motor de base de datos, **las entidades podrían necesitar ajustes**.

6.3. ¿Qué dificultades anticipas?

Compatibilidad de Funciones:

Diferentes motores de base de datos pueden tener características y funciones específicas. La **adaptación de estas funciones puede ser complicada** y tener que manejar **bases de datos en local y en remoto** puede llevar un poco de complejidad a la hora de programarlo. El objetivo nuestro es ofrecer una **experiencia de usuario lo mas gratificante** posible, y para ello intentar que la aplicacion pueda ser parte de ella **offline-first**.

Rendimiento y Optimizaciones:

Los motores de base de datos tienen rendimientos y optimizaciones específicas. La transición podría afectar el rendimiento de la aplicación, y se necesitarán ajustes para **optimizarla** para el motor que vamos a usar.

6.4. ¿Cómo podrías diseñar tu aplicación para minimizar el impacto de tal cambio?

Capa de Abstracción:

Utilizar **interfaces o capas de abstracción para acceder a la capa de datos**. De esta manera, podrías cambiar la implementación de la capa de datos **sin afectar directamente al resto de la aplicación**.

Patrón Repositorio:

Aplicar **el patrón de repositorio** de manera efectiva para aislar la lógica de acceso a datos. Siempre accede a los datos a través de la **interfaz del repositorio**, permitiendo **cambiar la implementación del repositorio según el motor de base de datos**.

Inyección de Dependencias:

Usar la **inyección de dependencias** para suministrar las implementaciones de los componentes relacionados con la base de datos. Esto facilitará la **sustitución de una implementación por otra** sin cambiar el código del cliente.

Pruebas Unitarias:

Implementar **pruebas unitarias y CI/CD** sólidas para las capas de lógica de dominio y acceso a datos. Las pruebas facilitan la **identificación de problemas durante el uso de la aplicacion o del motor de base de datos y proporcionan una capa de seguridad** durante todo el desarrollo del producto.

Documentación Clara:

Documentar detalladamente la estructura del proyecto, consultas y operaciones utilizadas facilitará tanto el desarrollo como **los cambios que se efectúen** durante el desarrollo del producto.

7. Conclusión

En esta práctica de Diseño de la Base de Datos, hemos abordado de manera integral **la arquitectura y la elección de tecnologías** para la **capa de persistencia** en nuestro proyecto de aplicaciones móviles.

La decisión de utilizar **Firebase**, **Room** y la integración de la lógica de persistencia consideramos que hara que **nuestra aproximación prospera, asegurando una base sólida para el crecimiento y la evolución de nuestro proyecto**.