Diseño de la Base de Datos

03/11/2023

Carlos Rodríguez del Toro
Antonio Medina Santana

Índice

Arquitectura

La arquitectura del proyecto será MVVM (Model View ViewModel). Por lo tanto, la capa de persistencia se sitúa en el modelo.

Los modelos representan los datos que se van a almacenar en la base de datos. Entidades: 'Usuario' y 'Libro'

Base de Datos

La tecnología seleccionada para la base de datos de la aplicación será Room. Este motor de base de datos está específicamente diseñado para aplicaciones Android por lo que será más sencilla su integración en Android Studio (software en el que se va a implementar la app).

Se podrá aplicar Kotlin para las consultas de la base de datos. Una de las grandes ventajas de Room respecto a otros motores es la eliminación del conocido como 'boilerplate code' que obliga a la repetición constante de código muy similar. Esto permitirá emplear más tiempo en el diseño de la interfaz de la aplicación y añadir y perfeccionar funcionalidades. Hay que tener en cuenta que el tiempo para desarrollar la aplicación será escueto (aproximadamente dos meses) por lo que el tiempo será un recurso muy valioso.

Room también permite modificar el esquema de la base de datos sin perder información e integra el ViewModel que hace más sencilla la actualización de la interfaz cuando la base de datos sufre modificaciones en tiempo real.

Finalmente, existe un conocimiento por parte de los desarrolladores de la app tras realizar el CodeLab tutorial de la implementación de Room en una aplicación de Android. Por ende, será más sencillo y no será necesario obtener conocimientos de cómo implementar y mantener otro entorno.

Capa de Persistencia

La capa de persistencia en nuestro modelo es esencial en la gestión de datos, para almacenar y recuperar información relevante para el funcionamiento de la aplicación. Más allá de los datos usuales, como nombres de usuario y contraseñas, nuestra base de datos se centra principalmente en registrar información relacionada con las búsquedas de libros realizadas por los usuarios.

Para lograr esto, diseñamos una estructura de tablas que contendrá un registro de todos los libros escaneados por cada usuario, junto con detalles adicionales sobre estos libros, como título, autor, género, fecha de escaneo, y otros atributos relevantes. Esto nos proporcionará un histórico detallado de las búsquedas de libros, como se pretendía para la funcionalidad de biblioteca.

Además de las tablas dedicadas a los libros y las búsquedas, nuestra base de datos incluirá tablas para gestionar la información de usuarios y contraseñas, asegurando la autenticación y seguridad de la plataforma. También tenemos pensado crear una tabla complementaria para la funcionalidad de agregar amigos.

Es importante recordar que toda esta infraestructura de base de datos se implementará en la capa del modelo de la aplicación. Aquí es donde estableceremos la conexión con la base de datos de Room. Esto asegura una gestión eficiente y segura de los datos, permitiendo a nuestra aplicación acceder y manipular la información de manera adecuada, respaldando así la funcionalidad central de la plataforma.

Cambios Futuros

Existen diversos motivos por los que podría ser necesario o recomendable modificar el motor de la base de datos. Por ejemplo, la limitación de rendimiento y la escalabilidad. En caso de que la aplicación crezca drásticamente, es posible que el volumen de datos que se maneje sea demasiado grande como para ser gestionado por Room. Por lo tanto, la escritura, lectura o consulta de la base de datos podría tener un rendimiento poco óptimo que obligue al cambio.

Otro motivo podría ser la necesidad de nuevas funcionalidades. La aplicación podría variar y evolucionar, adoptando nuevas características que Room no sería capaz de satisfacer. Por ejemplo, la aplicación podría empezar a tratar y manipular datos geoespaciales para la ubicación del usuario o la búsqueda de puntos de interés cercano como bibliotecas o librerías. Room no ofrece, de momento, soporte nativo para este tipo de consultas mientras que otros motores como PostGIS sí lo hacen. Por último, otros motivos podrían ser los altos costes de licencias y mantenimiento o la aparición de nuevos motores de base de datos revolucionarios o más apropiados.

En caso de cambiar de motor, los mayores desafíos se encontrarán en la migración y los cambios en el código. En lo que a la migración concierne, habrá que trasladar todos los datos almacenados en la base de datos obsoleta a la nueva base de datos. Esto se deberá hacer de manera eficiente y con garantía de seguridad ya que podría ser dramática la pérdida de información importante. Además del riesgo, habrá que dedicar tiempo no solo para que los datos se transfieran (tiempo que podría hacer a la app no estar operativa) sino que habrá que realizar pruebas exhaustivas para asegurar que las operaciones funcionan de manera óptima en el nuevo entorno. La reescritura de código también podría ser necesaria ya que diferentes motores de base de datos requieren diferentes lógicas y aplicaciones en la capa de persistencia. Por ello, para acceder a la base de datos, los desarrolladores tendrían que modificar el código existente para adaptarlo a las necesidades del nuevo motor.

Una técnica para minimizar el impacto de cambiar el motor de la base de datos es la abstracción de la lógica de la base de datos. Esto quiere decir que usando patrones de diseño, se podría aislar la lógica de acceso a datos de las capas superiores de la aplicación. Por lo tanto, las modificaciones se limitarían a repositorios, proporcionando flexibilidad y

facilidad en el cambio. También, la producción de una documentación clara. Esto simplificará la labor de localización de qué partes del código han de modificar los desarrolladores. Si el código no está bien explicado, puede ser lioso identificar dónde se encuentran los detalles que hay que modificar para realizar la migración.