Desarrollo de aplicaciones móviles I

Tema Nº12:Almacenamiento en Android.

Indicador de logro Nº12:Desarrolla una aplicación que persista datos, teniendo en cuenta las formas que nos permite el sistema operativo Android.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº12:**

Almacenamiento en Android.

**Subtema 12.1:**

Tipos de almacenamiento en Android

**Ejemplos:**

Android utiliza un sistema de archivos que es similar a los sistemas de archivos basados en discos de otras plataformas. El sistema proporciona varias opciones para que guardes los datos de tu app:

* Almacenamiento específico de la app: Almacena archivos diseñados solo para tu app, ya sea en directorios dedicados dentro de un volumen de almacenamiento interno o en directorios dedicados diferentes dentro del almacenamiento externo. Usa los directorios del almacenamiento interno para guardar información sensible a la que otras apps no deberían acceder.
* Almacenamiento compartido: Almacena archivos que tu app pretenda compartir con otras apps, incluidos archivos multimedia, documentos y otros.
* Preferencias: Almacena datos primitivos y privados en pares clave-valor.
* Bases de datos: Almacena datos estructurados en una base de datos privada mediante la biblioteca de persistencias Room.

**Subtema 12.2:**

Implementando Shared Preferences en Android

**Ejemplos:**

Puedes crear un nuevo archivo de preferencias compartidas o acceder a uno existente llamando a uno de estos dos métodos:

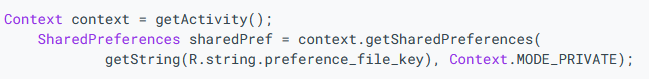
* getSharedPreferences(): utiliza este método si necesitas varios archivos de preferencias compartidas identificados por nombre, que especificas con el primer parámetro. Puedes llamar a este método desde cualquier instancia de Context en tu app.
* getPreferences(): utiliza este método desde una instancia de Activity si necesitas utilizar un solo archivo de preferencias compartidas para la actividad. Como este método recupera un archivo de preferencias compartidas predeterminado que pertenece a la actividad, no necesitas indicar un nombre.

Con Shared Preferences podemos almacenar y recuperar en el formato clave-valor información como texto, booleanos y números; lo que lo convierte en potencial para almacenar configuraciones del usuario como: estilos, preferencias, etc.

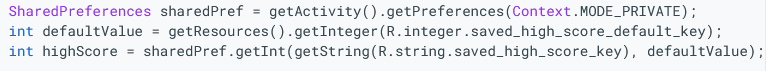
Los modos de acceso posibles son:

* MODE\_PRIVATE: Sólo nuestra aplicación tiene acceso a estas preferencias.
* MODE\_WORLD\_READABLE: Todas las aplicaciones pueden leer estas preferencias, pero sólo la nuestra puede modificarlas (deprecated desde el API 17).

Su uso es sencillo; si queremos recuperar una "preferencia" que hemos llamado "MiPreferencia" basta con emplear el siguiente código:



Para obtener la información de la preferencia:



Los parámetros consisten en:

* El nombre de la preferencia que queremos recuperar.
* Valor por defecto que será utilizado si la preferencia no existe en el sistema.

Para guardar o modificar información de la preferencia:

Texto

Descripción generada automáticamente

**Subtema 12.3:**

Android y SQLite

**Ejemplos:**

Guardar datos en una base de datos es ideal para los datos estructurados o que se repiten, como la información de contacto. En esta página, en la que se asume que estás familiarizado con las bases de datos SQL en general, encontrarás información que te ayudará a comenzar a usar bases de datos SQLite en Android.

Cómo definir un esquema y un contrato

Uno de los principios fundamentales de las bases de datos SQL es el esquema: una declaración formal de la manera en la que la base de datos está organizada. El esquema se refleja en las instrucciones de SQL que utilizas para crear la base de datos. Tal vez te resulte útil crear una clase complementaria, denominada clase de contratos, que indique explícitamente el diseño del esquema de forma sistemática y autodocumentada.

La clase de contratos es un contenedor de constantes que definen nombres de URI, tablas y columnas. Esta clase te permite utilizar las mismas constantes en todas las otras clases del mismo paquete, por lo que puedes cambiar el nombre de una columna en un lugar y propagar ese cambio en todo el código.

Una forma adecuada de organizar una clase de contratos consiste en incluir definiciones que sean globales para toda la base de datos en el nivel raíz de la clase. Luego, se debe crear una clase interna para cada tabla. Cada clase interna enumera las columnas de tabla correspondientes.

Características interesantes de SQLite

* El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host.
* SQLite usa un sistema de tipos inusual. En lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede insertar una cadena de texto en una columna de tipo entero (a pesar de que SQLite tratará en primera instancia de convertir la cadena en un entero).

**Subtema 12.4:**

Android y acceso a la base de datos.

**Ejemplos:**

Room proporciona una capa de abstracción sobre SQLite que permite acceder a la base de datos sin problemas y, al mismo tiempo, aprovechar toda la potencia de SQLite.

Las apps que controlan grandes cantidades de datos estructurados pueden beneficiarse con la posibilidad de conservar esos datos localmente. El caso práctico más común es almacenar en caché datos relevantes. De esa manera, cuando el dispositivo no puede acceder a la red, el usuario de todos modos puede explorar ese contenido mientras está desconectado. Cualquier cambio de contenido iniciado por el usuario se sincroniza con el servidor una vez que el dispositivo vuelve a estar en línea.

Para usar Room en tu app, agrega las siguientes dependencias al archivo build.gradle de tu app:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Estos son los 3 componentes principales de Room:

* Base de datos: Contiene el titular de la base de datos y sirve como punto de acceso principal para la conexión subyacente a los datos persistentes y relacionales de tu app.
  + La clase anotada con @Database debe cumplir con las siguientes condiciones:
  + Ser una clase abstracta que extiende RoomDatabase
  + Incluir la lista de entidades asociadas con la base de datos dentro de la anotación
  + Contener un método abstracto que tenga 0 argumentos y muestre la clase anotada con @Dao

En el entorno de ejecución, puedes adquirir una instancia de Database llamando a Room.databaseBuilder() o Room.inMemoryDatabaseBuilder().

* Entidad: Representa una tabla dentro de la base de datos.
* DAO: Contiene los métodos utilizados para acceder a la base de datos.

La app usa la base de datos de Room para obtener los objetos de acceso a los datos (DAO) asociados con esa base de datos. Luego, la app usa cada DAO para obtener entidades de la base de datos y guardar los cambios realizados en esas entidades en la base de datos. Por último, la app usa una entidad para obtener y configurar valores que corresponden a columnas de tabla dentro de la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

En el siguiente fragmento de código, se muestra una configuración de base de datos de ejemplo con una entidad y un DAO:

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Después de crear los archivos anteriores, obtendrás una instancia de la base de datos creada con el siguiente código:



* Los componentes de arquitectura son una colección de bibliotecas que ayudan a desarrollar aplicaciones sólidas
* Permite administrar el ciclo de vida de la aplicación de una manera sencilla, apoyándose en las siguientes tecnologías:
  + LiveData para notificar a la vista de cambios en el modelo
  + ViewModel para almacenar el estado de la aplicación
  + Room, biblioteca de persistencia a una base de datos local en SQLite
* Estos componentes forman parte del conjunto de bibliotecas de JetPack

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

**Actividad:**

1. CUESTIONARIO TÉCNICO

* ¿Qué es SQLite?
* ¿Para qué sirve ROOM?
* Desarrollar un proyecto de Android que al ejecutarlo genere una base de datos en SQLite llamada tienda que contenga una tabla llamada producto
* Al iniciar la aplicación se debe mostrar en un RecyclerView una lista de productos previamente cargados en la base de datos
* Realizar la carga inicial de datos desde el archivo de configuración de la conexión

1. CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_