Инженерный вестник Дона, №1 (2023)
ivdon.ru/ru/magazine/archive/n1y2023/8152

К вопросу использования библиотеки Room из набора Android Jetpack и упрощения разработки мобильных приложений для Android

В.А. Жжонов, В.А. Евсина, С.Н. Широбокова

Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова, Новочеркасск

Аннотация: Мобильные приложения сегодня – это необходимый инструмент для работы, учебы, развлечений и связи со всем информационным миром. С каждым годом возрастают требования к приложениям, а также потребность в стабильных и многофункциональных программных средствах, которые будут способны быстро выполнять поставленные перед ними задачи. Хоть большинству приложений и необходим доступ в интернет для связи с сервисами, требуется также обеспечить и сохранность данных на самом устройстве, чтобы была возможность оффлайн - доступа к данным. Для решения этой задачи при разработке мобильных приложений существует множество различных средств, но наиболее распространённым является библиотека *Room*, входящая в пакет «*Android Jetpack*». В статье приведено краткое описание функциональных возможностей данной библиотеки. Рассмотрена работа всех основных компонентов с базовыми аннотациями. Также схематично представлено взаимодействие основных компонентов библиотеки и показан пример реализации в мобильном приложении для операционной системы *Android*.
Ключевые слова: база данных, *SQLite*, *Android Jetpack*, *Room*, *Android*.

Мобильные приложения в современном мире – это не только наиболее удобный способ взаимодействия с информационными сервисами, но и необходимый инструмент для большого количества сфер деятельности [1,2]. Сегодня существует множество многофункциональных программ, решающих различные задачи [2,3]. Постоянное повышение требований к безопасности, скорости и многим другим параметрам заставляют разработчиков создавать различные инструменты для обеспечения поставленных задач. Так, вопрос об эффективном построении базы данных в мобильных приложениях на операционной системе *Android* привел к разработке удобной и многофункциональной библиотеки *Room*, входящей в состав пакета «*Android Jetpack*» [4]. Подобные решения позволяют значительно упростить и ускорить разработку различных приложений [5-7].

Наиболее используемой базой данных в мобильных приложениях является *SQLite*. Но работать напрямую с базой данных – довольно сложная задача, поскольку требуется, как минимум, вручную создавать необходимые

сущности и атрибуты, не говоря уже о дополнительной обработке данных при их получении и прочем. Чтобы значительно упростить процесс работы с базой данных, были созданы специальные надстройки над базовым *SQLite*. К таким надстройкам и относится специальная библиотека *Room*, которая позволяет получать доступ к данным, используя все функциональные возможности *SQLite*, но в гораздо более удобной форме. Библиотека предоставляет такие преимущества работы с базой, как [8]:

– упрощенный процесс миграции базы данных;

– возможность проверки *SQL*-запросов во время компиляции проекта;

– использование аннотаций, что минимизирует повторение шаблонного кода.

Можно выделить три основных компонента *Room* [8]:

1. Класс базы данных – является основным для доступа к методам взаимодействия с данными, сохраненными в базе.

2. Объект данных – представляет собой таблицу.

3. Объект доступа к данным (*DAO*) – описывает методы, которые могут быть использованы для операций, совершаемых с данными в базе.

Для работы с данной библиотекой необходимо создать требуемый класс объекта данных, который будет содержать все нужные атрибуты таблицы [8]. Далее, для выбранного объекта создается *DAO* – специальный интерфейс, в котором необходимо указать методы для работы с объектом данных [9]. В конечном счете, в классе базы данных мы определяем сущности, указывая созданные классы объектов данных, а также объекты *DAO*, к которым будем обращаться для работы с необходимым сущностями. На рис. 1 показана архитектура библиотеки *Room*.

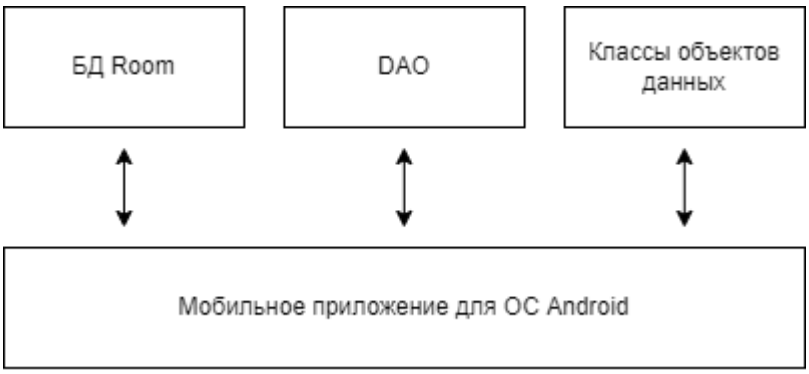

Рис. 1. – Схема архитектуры библиотеки Room

Рассмотрим основные компоненты на примере. На рис. 2 представлен класс объекта данных «*UserInform*». Как можно заметить, перед объявлением класса указывается специальная аннотация «*@Entity*», которая показывает, что данный класс представляет собой сущность базы данных. Для обозначения первичного ключа используется аннотация «*@PrimaryKey*», опционально можно включить автогенерацию ключа. Обычно названия столбцов таблицы формируются по названию соответствующего атрибута класса, но также можно изменить его название на произвольное с помощью параметра «*name*» аннотации «*@ColumnInfo»*. На рис. 3 показан интерфейс, описывающий методы доступа к данным. Перед объявлением интерфейса указывается специальная аннотация *«@Dao»*. Чтобы определить свой собственный запрос к таблице, используется аннотация «*@Query*». Для обычных операций вставки, обновления и удаления используются такие аннотации, как «*@Insert*», «*@Update*», «*@Delete*». После выполнения запроса на получение информации, данные формируются в соответствующий объект автоматически. Кроме того, существует возможность интеграции различных фреймворков для выполнения асинхронных запросов, например, использование компонента *Single* из *RxJava*, что показано на рис. 3 [10].

```
@Entity
public class UserInform {
    @PrimaryKey (autoGenerate = true) @NonNull
    public Integer id;
    @ColumnInfo(name = "full_name")
    public String fio;
    public String email;
    public Integer cityId;
    public Date birthday;
```

Рис. 2. – Класс объекта данных

```
@Dao
public interface UserInformDao {

    @Query("SELECT * FROM UserInform")
    List<AccessUserInform> getAll();

    @Query("SELECT * FROM UserInform WHERE Id = :id")
    Single<AccessUserInform> get(Integer id);

    @Insert
    void insert(AccessUserInform employee);

    @Update
    void update(AccessUserInform employee);

    @Delete
    void delete(AccessUserInform employee);

}
```

Рис. 3. – Объект доступа к данным (*DAO)*

На рис. 4 представлен абстрактный класс базы данных, представляющий собой расширение класса «*RoomDatabase*». Перед объявлением класса в аннотации «*@Database*» необходимо указать классы объектов, которые будут представлять собой таблицы в базе данных, а также

версию базы данных. Далее указываются *DAO*, при обращении к которым будет возможно взаимодействовать с сущностями базы данных.

```
@Database(entities = {UserInform.class}, version = 1)
public abstract class MainData extends RoomDatabase {
    public abstract UserInformDao userInformDao();
}
```

Рис. 4. – Класс базы данных

Как можно заметить, использование данной библиотеки позволяет значительно упросить процесс работы с базой. Запросы выборки автоматически преобразуют полученные данные в соответствующий класс, благодаря чему возможно сразу работать с полученными данными, в отличие от работы с базой напрямую, где необходимо вручную описывать методы получения *Data*-классов посредством обработки курсоров. Таким образом, использование данной библиотеки значительно упрощает процесс конструирования базы данных, а также работы с ней, избавляя от необходимости применять множество однотипного и избыточного кода. В целом уменьшается время, затрачиваемое на написание кода, и ускоряется процесс разработки проекта, что на сегодняшний день является очень важным аспектом успешности продукта.

Литература

1. Joseph D. Blackburn, Gary D. Scudder, Luk Van Wassenhove. Improving Speed and Productivity of Software Development: A Global Survey of Software Developers // IEEE Transactions on Software Engineering, 1997. URL: researchgate.net/publication/3187839_Improving_Speed_and_Productivity_of_Software_Development_A_Global_Survey_of_Software_Developers.

2. Александровский В.Г. Мобильные технологии в строительстве. Программное обеспечение на платформе Android. Часть 1 // Инженерный вестник Дона, 2019, №4. URL: ivdon.ru/ru/magazine/archive/n4y2019/5874.

3. Mentsiev A.U., Alams M.T. Mobile forensic tools and techniques: Android data security // Инженерный вестник Дона, 2019, №2. URL: ivdon.ru/ru/magazine/archive/n2y2019/5766.

4. Android Jetpack. URL: developer.android.com/jetpack.

5. Natoli Paula. Upwardly mobile—mobile technologies span the supply chain // Supply & Demand Chain Executive, 2015. URL: sdcexec.com/warehousing/article/12091968/upwardly-mobilemobile-technologies-span-the-supply-chain.

6. Destefano Robert. Four steps to successfully deploy android in your supply chain // Supply & Demand Chain Executive, 2018. URL: sdcexec.com/software-technology/blog/20998353/ivanti-supply-chain-4-steps-to-successfully-deploy-android-in-your-supply-chain.

7. Harrison Rachel, Flood Derek, Duce David. Usability of mobile applications: literature review and rationale for a new usability model // Supply & Journal of Interaction Science, 2013. URL: journalofinteractionscience.springeropen.com/articles/10.1186/2194-0827-1-1.

8. Defining data using Room entities. URL: developer.android.com/training/data-storage/room/defining-data.

9. Accessing data using Room DAOs. URL: developer.android.com/training/data-storage/room/accessing-data.

10. Write asynchronous DAO queries. URL: developer.android.com/training/data-storage/room/async-queries.

References

1. Joseph D. Blackburn, Gary D. Scudder, Luk Van Wassenhove. IEEE Transactions on Software Engineering, 1997. URL:

researchgate.net/publication/3187839_Improving_Speed_and_Productivity_of_Software_Development_A_Global_Survey_of_Software_Developers.

2. Alexandrovsky V.G. Inzhenernyj vestnik Dona, 2019, №4. URL: ivdon.ru/ru/magazine/archive/n4y2019/5874.

3. Mentsiev A.U., Alams M.T. Inzhenernyj vestnik Dona, 2019, №2. URL: ivdon.ru/ru/magazine/archive/n2y2019/5766.

4. Android Jetpack. URL: developer.android.com/jetpack.

5. Natoli Paula. Supply & Demand Chain Executive, 2015. URL: sdcexec.com/warehousing/article/12091968/upwardly-mobilemobile-technologies-span-the-supply-chain.

6. Destefano Robert. Supply & Demand Chain Executive, 2018. URL: sdcexec.com/software-technology/blog/20998353/ivanti-supply-chain-4-steps-to-successfully-deploy-android-in-your-supply-chain.

7. Harrison Rachel, Flood Derek, Duce David. Supply & Journal of Interaction Science, 2013. URL: journalofinteractionscience.springeropen.com/articles/10.1186/2194-0827-1-1.

8. Defining data using Room entities. URL: developer.android.com/training/data-storage/room/defining-data.

9. Accessing data using Room DAOs. URL: developer.android.com/training/data-storage/room/accessing-data.

10. Write asynchronous DAO queries. URL: developer.android.com/training/data-storage/room/async-queries.