Ljrkkkkffff

Здравствуйте уважаемые слушатели.

Сегодня я хотел бы рассказать о моем мобильном приложении помощник учителя.

Расскажу в кратце о чем приложение:

Приложение создавалось для учителей, и представляет из себя электронные заметки для преподавателя.

Приложение позволяет вносить в него классы учеников в виде списков проставлять им оценки.

Также в приложении можно перенести кабинеты, в которых работает учитель. Это нужно для того, чтобы на уроке учитель мог открыть на экране телефона список учеников и удобно выставлять им оценки, а также отмечать присутствующих.

В приложении можно внести поурочное расписание, чтобы в нужный момент открыть нужный урок.

Теперь расскажу о базе данных, встроенной в приложение.

Поскольку фактически это оффлайн заметки, то в приложении необходимо было использовать какое-то внутреннее хранилище. Я выбрал СУБД SQLite3. На момент начала работы над приложением, а это 2016 год, SQLite3 имела огромное количество примеров работы с ней и альтернатив на тот момент почти не было. Эта СУБД входит в стандартный пакет инструментов разработки, а это значит, что она уже настроена и готова для использования в коде… в теории.

Структуру базы данных я разрабатывал в процессе создания приложения, скажем так "под интерфейс". И по этой причине приложение скорее представляет из себя обёртку вокруг БД.

....Дальше рассказ по схеме....

Поскольку это приложение я защищал как диплом, то я успел описать его буквально со всех сторон. За исключением одного аспекта, я не описывал ошибки и проблемы с которыми я столкнулся

Начал я его делать в 16м году

Room - это библиотека, представленная на Google I/O в 2017 году

Работает она через аннотации и позволяет описать модель базы данных через классы. И вся прелесть его как раз в удобной работе с данными. Тк база данных создается на основе классовой модели данных, а соответственно выгрузка и сохранение данных становится очень удобными.

Не приходится на прямую писать Sql запросы с подставлением значений, а потом выяснять где же вы конкретно не поставили запятую или перепутали местами подставляемые значения.

Но на тот момент я был не очень-то и опытен, а база данных уже была наполовину написана, да и туториалов на эту тему было не много, и основная их часть появилась в 18м.

Сейчас я хорошо понимаю, что нужно было бы сразу писать базу данных так чтобы не возиться с сырой и местами неповоротливой библиотекой SQLite, а воспользоваться удобной официальной надстройкой от google.

Сейчас я смотрю на код перегруженный огромным количеством неизменяемых констант которые уже находятся на продакшн и понимаю что стоило действительно взять какую-нибудь аннотационную модель.

Сейчас в приложении более 35 тысяч строк кода, завязанного с большой базой данных, и я не хотел переписывать почти всё приложение. В общем не повторяйте моих ошибок.. :)

Room - это ORM, библиотека объектно-реляционного отображения. Другими словами, Room сопоставит объекты нашей базы данных с объектами Java. Room предоставляет уровень абстракции поверх SQLite, чтобы обеспечить свободный доступ к базе данных, используя при этом всю мощь SQLite.

Разница между SQLite и Room persistence library:-

В случае SQLite нет проверки исходных запросов SQLite во время компиляции. Но в Room есть проверка SQL во время компиляции.

По мере изменения вашей схемы вам необходимо обновлять соответствующие SQL-запросы вручную. Комната решает эту проблему.

Вам нужно использовать много шаблонного кода для преобразования между SQL-запросами и объектами данных Java. Но Room сопоставляет объекты нашей базы данных с Java Object без шаблонного кода.

Room создан для работы с LiveData и RxJava для наблюдения за данными, в то время как SQLite этого не делает.

Аннотации Room и основные компоненты:

@Entity — Определяют таблицы нашей базы данных

@DAO — Предоставляют API для чтения и записи данных

@Database — Представляют владельца базы данных

Вот ссылка на статью medium, в которой подробно объясняется использование и преимущества Room persistence library.

Редактировать 1: Вы можете обратиться к документам Google для разработчиков, в которых четко объясняется, как сохранять данные в локальной базе данных с помощью room. Ссылка на документы Google для разработчиков

Пример реализации

На этом слайде представлен пример реализации базы данных Room с одним объектом данных и одним DAO.

--Объект данных--

Следующий код определяет объект данных User. Каждый экземпляр User представляет строку в пользовательской таблице в базе данных приложения.

--Объект доступа к данным (DAO)--

Следующий код определяет DAO с именем UserDao. UserDao предоставляет методы, которые использует остальная часть приложения для взаимодействия с данными в пользовательской таблице.

И вообще для андроид разработки Google IO очень полезный источник информации.

**История развития приложки**

**Косяки на проде**

Room is an ORM, Object Relational Mapping library. In other words, Room will map our database objects to Java objects. Room provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite.

Difference between SQLite and Room persistence library:-

In case of SQLite, There is no compile time verification of raw SQLite queries. But in Room there is SQL validation at compile time.

As your schema changes, you need to update the affected SQL queries manually. Room solves this problem.

You need to use lots of boilerplate code to convert between SQL queries and Java data objects. But, Room maps our database objects to Java Object without boilerplate code.

Room is built to work with LiveData and RxJava for data observation, while SQLite does not.

Room annotations and main components:

@Entity — Define our database tables

@DAO — Provide an API for reading and writing data

@Database — Represent a database holder

Here is the link to the medium article which explains in detail the usage and benefits of Room persistence library.

Edit 1: You can refer to Google developer docs, which clearly explains how to save data in a local database using room. Link to Google Developer Docs



Рисунок 26 – Инфологическая модель базы данных