Гуров М.В.

студент,

Научный руководитель: **Олешко О.И.**

кандидат технических наук, доцент,

Харьковский национальный университет
имени В.Н. Каразина

АНАЛИЗ SQL И NOSQL РЕШЕНИЙ ДЛЯ ХРАНЕНИЯ ДАННЫХ В ОС ANDROID

По статистике, устройства под управлением мобильных операционных систем являются одними из самых используемых человеком в современной повседневной жизни. Среди мобильных операционных систем в мире преобладают: Android (76.82%), iOS (20.45%) и другие (2.28%) [1]. Все эти устройства позволяют выполнять множество разнообразных задач: просмотр прогноза погоды, управление банковским счетом, заказ такси, покупка билетов, просмотр и создание медиа контента, а также многое другое. Все эти задачи объединяет то, что для их выполнения необходим интернет. Сейчас уже сложно найти новое устройство без соответствующих модулей связи для подключения к интернету.

Рост интереса к мобильным устройствам влечет за собой большой спрос на разработку и поддержку мобильных приложений, что в свою очередь увеличивает предъявляемые к ним требования. Актуально рассмотреть операционную систему Android, поскольку она занимает первое место по количеству активных устройств. Каждое приложение, в той или иной мере, нуждается в хранении каких-либо данных. Это могут быть пользовательские настройки, состояние элементов интерфейса, промежуточные данные каких-либо вычислений или медиа контент.

Вопрос хранения данных стоит, как никогда остро, ведь их количество растет с каждым днем. В Android SDK изначально поставляется несколько механизмов хранения данных, такие как: *SQLite*, *SharedPreferences*. Все они доступны с самой первой версии API. Среди сторонних библиотек логично рассмотреть их аналоги.

SQLite

Компактная встраиваемая и очень легковесная СУБД. Все манипуляции с данными выполняются средствами языка SQL. Не требует дополнительных зависимостей, является одной из самых быстрых баз данных доступных под ОС Android. Одновременно достоинством и недостатком является поддержка SQL, с одной стороны, это очень функциональный инструмент для манипуляций с данными, а с другой, это дополнительный язык программирования для разработчика. Среди недостатков также стоит выделить: большое количество шаблонного кода, отсутствие безопасности хранимых данных, отсутствие механизма кэширования ответов, отсутствие эффективной параллельной записи данных с нескольких потоков / процессов в пределах одной базы данных (блокировка на файл базы данных), отсутствие проверок времени компиляции

(например, для SQL запросов), отсутствие возможности тестировать без использования интеграционных тестов и необходимость вынесения работы с базой на фоновый поток.

SharedPreferences

Простая реализация NoSQL хранилища «ключ-значение». Ключами могут быть строки, значения которых представлены исключительно примитивами (*int*, *long*, *boolean*, *float*, *String*). Данные хранятся в XML файлах, расположенных в приватной поддиректории приложения «*shared-prefs*» (в *Internal storage*). Реализовано кэширование и поддержка как синхронной, так и асинхронной записи, т.е. есть возможен доступ напрямую из графического потока. Использование ограничено низкой производительностью и отсутствием возможности хранения произвольных классов данных. Хранимые данные не защищены и имеют текстовый формат, тестирование исключительно через интеграционные тесты, хотя подобной надобности на практике не возникает в виду очень простого функционала. Может быть использовано для хранения пользовательских настроек и состояния графического интерфейса между сессиями работы приложения, т.е. данных потеря, подмена или утечка которых допускается.

Также существует ряд сторонних решения для ОС Android, такие как: *Room*, разрабатываемая компанией Google и предоставляемая в пакете Android Architecture Components как дополнение к Android SDK, *Paper*, *GreenDAO*, *BinaryPreferences*, *ORMLite*, *Requery*, *Realm*.

Room

Библиотека представляет собой высокоуровневый интерфейс для низкоуровневых привязок SQLite, обеспечивает более надежный доступ, избавляет от шаблонного кода при работе с SQLite (классом *SQLiteOpenHelper*) и при этом оставляя возможность использовать все возможности SQL. В первую очередь, Room позиционируется, как альтернатива и замена SQLite поставляемого в Android SDK, являясь надстройкой последнего. Работает с использованием Java Annotation Processing [2], т. е. большая часть работы происходит на этапе компиляции. При сравнимой с SQLite производительностью, Room обладает рядом неоспоримых преимуществ таких как: статическая проверка SQL на этапе компиляции, автоматическое преобразование данных, поддержка JavaRX, инструменты тестирования миграций схемы данных.

Таблица 1

Сравнительный тест производительности Room и SQLite

Библиотека	Write(s)	Read(s)	Update(s)	Delete(s)
SQLite	48	425	64	81
Room	134	687	170	109

(Запущен на OnePlus 5T/6, вставка 1000 объектов без оптимизаций)

BinaryPreferences

Key-Value база данных, которая предоставляет легковесное решение, нацеленное на высокую производительность, низкие затраты ресурсов и расширение возможностей стандартной реализации SharedPreferences. Реализует интерфейс SharedPreferences, что стоит отнести как к минусам, так и плюсам (например, при миграции со стандартной реализации). К достоинствам относится: относительно маленький размер библиотеки и отсутствие внешних зависимостей, производительность, превосходящая существующие NoSQL решения, возможность отслеживания изменения данных, синхронизация при работе в нескольких процессах. Библиотека поддерживает работу с произвольными типами (*классами*), предоставляет механизмы миграции схемы данных и реализации шифрования как ключей и значений. Из недостатков это большое количество шаблонного кода при работе с произвольными классами, накладные расходы используемой постоянной памяти при хранении примитивов в связи с блочным строением файловых систем.

По статистике [3] 40% опрошенных разработчиков предпочитают не использовать сторонние решения баз данных, а отдают предпочтения API предоставляемому Android SDK. На альтернативы в лице NoSQL приходится всего около 1-2% случаев использования благодаря тому, что большинство приложений достаточно типичны. Но, встречаются задачи, когда SQL базы данных избыточны и на много лучше воспользоваться NoSQL решением.

В стандартной поставке Android SDK присутствуют инструменты призванные решить большую часть задач, с которыми сталкиваются разработчики. Требования растут значительно быстрее чем успевают развиваться их API. Все это заставляет разработчиков Android приложений искать новые инструменты или даже самостоятельно разрабатывать их под свои нужды.

Список использованных источников:

1. Mobile Operating System Market Share Worldwide. – [Электронный ресурс] Режим доступа: <http://gs.statcounter.com/os-market-share/mobile/worldwide>.
2. Java Annotation Processing. – [Электронный ресурс] Режим доступа: <https://www.javacodegeeks.com/2015/09/java-annotation-processors.html>.
3. SQLite, ORMs, NoSQL: what Android developers use and why. – [Электронный ресурс] Режим доступа: <http://greenrobot.org/android/sqlite-orms-nosql-what-android-developers-use-and-why/>.