Mittuniversitetet

MID SWEDEN UNIVERSITY

DT133G, Final Project

Project Proposal

MITTUNIVERSITETET

DSV Östersund

Examiner	Dr. Felix Dobslaw	Felix.Dobslaw@miun.se
Supervisor	Raja-Khurram Shahzad	Raja-Khurram.Shahzad@miun.se
Author	Bernard Che Longho	lobe1602@student.miun.se
Programme	Programvaruteknik, 180hp	
Course	DT133G, Final Project	
Field of study	Computer Engineering	
Semester, year	VT, 2019	

Table of Contents

Project Proposal

Title	Room vs greenDAO for Android	
Subtitle	A comparative analysis of performance of Room and greenDAO on Android	
Ev. client		
Ev. contact		
Time Period	25th March 2019	6th June 2019

Background and Motivation

Mobile phone use have witnessed a steady increase in the past few years with the number of users estimated to reach five billion in 2019, amounting to about 67% of the world's population [1]. These devices often have applications that can be used to create, store and share almost any kind of information, from text, picture, or videos. Such data are often stored both during an application session and between sessions. This ability to store and retrieve data between application sessions is referred to as persistence [2]. There are four major ways of achieving persistence in Android applications [3], [4]. One such way is using Shared Preferences [5], which is used to store a relatively small amount of data like application preferences in key-value pairs and provides simple methods to read them [6]. Another way of persisting data is using the device's file system. This is suitable for storing such information like images, audio and video files. In this form of storage, one can either utilize the internal storage or the external storage [7]. Saving on the external storage can only be possible for devices that have an externally mounted micro Secure Digital (SD) card. A third option of local data storage for Android devices is by using a database [8], [9]. For database storage, Android developers have many persistent frameworks to choose from [10]. Some of these frameworks are the built-in SQLite [11], Room Persistence Library(Room) [12]. ActiveAndroid [13], greenDAO [14], Realm [15], and OrmLite [15]. All these mentioned frameworks are object-relational mapping (ORM) frameworks except SQLite. ORM frameworks are designed to relief the developer from writing raw SQL and to ease write and read of data to and from a database [17]. This plethora of database libraries implies Android developers must make a choice for storing structured data. In order to make this choice, developers have increasingly considered application performance as a major factor [18]. This is because application performance, in terms of response time and memory consumption (CPU and RAM) plays an important role in user-perceived quality of apps[19], [20],[21]. Using these third-party libraries can also increase the overall application size which in turn increases the memory usage of the application [22].

There is limited empirical studies that compare the performance of the different Android database frameworks. In a study to understand the energy, performance and programming effort trade-offs of SQLite, ActiveAndroid, greenDAO, Realm and OrmLite, Pu J. et al., [22] found that SQLite performs best while ActiveAndroid performs worst. They also concluded that greenDAO performed best among the ORMs. They however never compared the performance of Room which is the recommended framework by Google. This project adds to this knowledge of performance analysis by analyzing the performance of Room and greenDAO. It also seeks to know how much more of the application size is increased when using any of these frameworks.

Problem Statement

In order to persist data in their apps, Android developers often have to make the choice among the above mentioned database libraries. Both Room and greenDAO map objects to SQLite databases yet Google prefers Room and SQLite [8], [9]. As of the writing of this report, greenDAO is the forth most used library for database management in Android [24], coming after Firebase [25], Android Architecture Components (LiveData, ViewModel, Room) [26], and Realm [15]. To the best of the author's knowledge, there is no empirical study that analyses the performance benefit of Room compared to greenDAO. This lack of empirical study makes it difficult for Android developers to choose which framework to use when designing apps that require local data persistence. More so, using any of these libraries can also contribute to an increase in the application size which in turn contributes to the memory need of an application, a factor that has negative impact on application performance [22].

Research question

Does the use of Room for data persistence confer any performance benefit to an Android application compared to when using greenDAO for the same purpose?

In order to answer this research question, the following specific questions will analyzed.

i. How much time time does Room take to perform database CRUD(create, read, update and delete) operations compared to greenDAO?
ii. How much RAM and CPU is used by an Android application when executing database operations when Room is used as the database framework compared to when greenDAO is used as database framework?
iii. How does the use of Room or greenDAO affect the Android application size?

Proposed Methodology

In order to answer the research question three versions of an Android application containing a relational model shall be designed and developed. One version of the application shall persist the data using SQLite, another using greenDAO and the other using Room. The application using SQLite to persist the data is considered the control(reference) application. This means that all other measurements shall be compared to the measurements obtained from this application. The applications shall be installed into two Android Virtual Device (AVD) with varying RAM and CPU capabilities. This variation in device memory will control for any variations due to available memory.

On the question of time and memory measurements, these shall be done five times for each app and device and the average time and memory for the runs recorded. This number of runs is arbitrarily chosen. This number of runs limits errors due to cold start of the Java Virtual Machine(JVM). TimingLogger [27] shall be used to measure the time taken to perform the CRUD operations. TimingLogger is a utility class in the Android Software Development Kit (SDK) that enables timing of an application method. It allows for the use of named tags, records time in a similar fashion as a conventional timer with splits, and gives a fine print of the execution time of any method. The ability to use the tag makes it easier to filter the data. With regards to memory usage, the Android Profiler shall be started whenever the application starts and the memory usage recorded. This tool is recommended by Google [28] and has been used to measure the memory performance of Android apps [29]. As regards the application size, an Android Application Package (apk) shall be generated for each application using Android Studio with the steps `Build->Build bundle(s)/APK(s)) -> Build APK(s))`. When this is done, the apk sizes will be estimated using the bash command **`du -h <apk-file>`** (where apk-file is the apk file generated from the application).

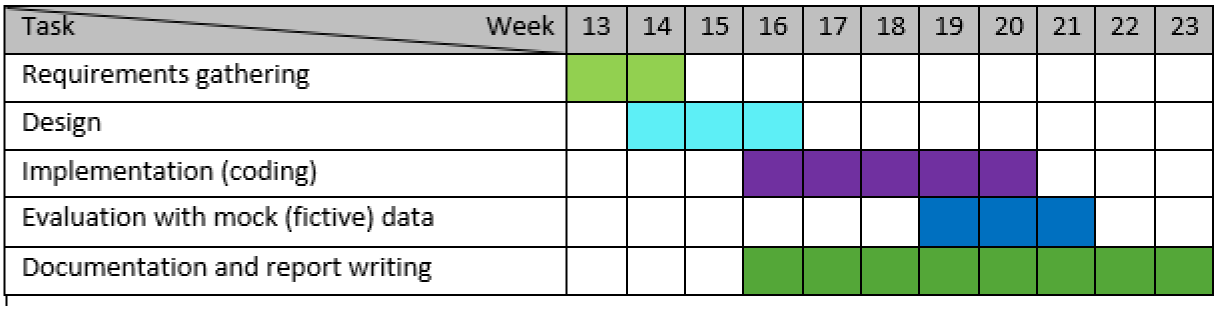
Intended Result

- Three android applications, one using SQLite, the other using greenDAO as database library and another using Room as database library.

- Data from the measurements described in the methodology section.

Time Plan

The thesis shall span between week13 to week 23 of 2019 and the work-load shall the divided as follows.



Task \ Week	13	14	15	16	17	18	19	20	21	22	23
Requirements gathering	🟩	🟩									
Design		🟦	🟦	🟦							
Implementation (coding)				🟪	🟪	🟪	🟪	🟪			
Evaluation with mock (fictive) data							🟦	🟦	🟦		
Documentation and report writing				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

References

[1] ‘Number of mobile phone users worldwide 2015-2020’, *Statista*. [Online]. Available: https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/. [Accessed: 04-May-2019].

[2] ‘What is Persistence? - Definition from Techopedia’, *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/8842/persistence-computing. [Accessed: 04-May-2019].

[3] ‘How to store data locally in an Android app - Android Authority’. [Online]. Available: https://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/. [Accessed: 09-Apr-2019].

[4] ‘Android From Scratch: How to Store Application Data Locally’, *Code Envato Tuts+*. [Online]. Available: https://code.tutsplus.com/tutorials/android-from-scratch-how-to-store-application-data-locally--cms-26853. [Accessed: 09-Apr-2019].

[5] ‘Save key-value data’, *Android Developers*. [Online]. Available: https://developer.android.com/training/data-storage/shared-preferences. [Accessed: 09-Apr-2019].

[6] ‘SharedPreferences’, *Android Developers*. [Online]. Available: https://developer.android.com/reference/android/content/SharedPreferences. [Accessed: 09-Apr-2019].

[7] ‘Save files on device storage | Android Developers’. [Online]. Available: https://developer.android.com/training/data-storage/files. [Accessed: 09-Apr-2019].

[8] ‘Save data in a local database using Room | Android Developers’. [Online]. Available: https://developer.android.com/training/data-storage/room. [Accessed: 09-Apr-2019].

[9] ‘Save data using SQLite’, *Android Developers*. [Online]. Available: https://developer.android.com/training/data-storage/sqlite. [Accessed: 09-Apr-2019].

[10] ‘Database libraries’, *AppBrain*. [Online]. Available: https://www.appbrain.com/stats/libraries/tag/database/database-libraries. [Accessed: 04-May-2019].

[11] ‘SQLite Home Page’. [Online]. Available: https://www.sqlite.org/index.html. [Accessed: 09-Apr-2019].

[12] ‘Room Persistence Library | Android Developers’. [Online]. Available: https://developer.android.com/topic/libraries/architecture/room. [Accessed: 09-Apr-2019].

[13] ‘Activeandroid by pardom’. [Online]. Available: http://www.activeandroid.com/. [Accessed: 17-Apr-2019].

[14] ‘greenDAO: Android ORM for your SQLite database’, *Open Source by greenrobot*. .

[15] ‘Realm Database’. [Online]. Available: https://realm.io/products/realm-database/. [Accessed: 17-Apr-2019].

[16] ‘OrmLite - Lightweight Object Relational Mapping (ORM) Java Package’. [Online]. Available: http://ormlite.com/. [Accessed: 17-Apr-2019].

[17] C. Xia, G. Yu, and M. Tang, ‘Efficient implement of ORM (object/relational mapping) use in J2EE framework: Hibernate’, presented at the 2009 International Conference on Computational Intelligence and Software Engineering, 2009, pp. 1–3.

[18] M. Linares-Vásquez, C. Vendome, Q. Luo, and D. Poshyvanyk, ‘How developers detect and fix performance bottlenecks in Android apps’, in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015, pp. 352–361.

[19] E. Noei, M. D. Syer, Y. Zou, A. E. Hassan, and I. Keivanloo, ‘A study of the relation of mobile device attributes with the user-perceived quality of Android apps’, *Empir. Softw. Eng.*, vol. 22, no. 6, pp. 3088–3116, Dec. 2017.

[20] S. Yang, D. Yan, and A. Rountev, ‘Testing for poor responsiveness in Android applications’, presented at the 2013 1st International Workshop on the Engineering of Mobile-Enabled Systems (MOBS), 2013, pp. 1–6.

[21] S. Ickin, K. Petersen, and J. Gonzalez-Huerta, ‘Why Do Users Install and Delete Apps? A Survey Study’, presented at the International Conference of Software Business, 2017, pp. 186–191.

[22] Google, ‘Manage your app’s memory’, *Android Developers*. [Online]. Available: https://developer.android.com/topic/performance/memory#reduce. [Accessed: 09-May-2019].

[23] J. Pu, Z. Song, and E. Tilevich, ‘Understanding the energy, performance, and programming effort trade-offs of Android persistence frameworks’, presented at the 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2016, pp. 433–438.

[24] ‘greenDAO - Android SDK statistics’, *AppBrain*. [Online]. Available: https://www.appbrain.com/stats/libraries/details/greendao/greendao. [Accessed: 04-May-2019].

[25] ‘Firebase - Android SDK statistics’, *AppBrain*. [Online]. Available: https://www.appbrain.com/stats/libraries/details/firebase/firebase. [Accessed: 04-May-2019].

[26] ‘Android Architecture Components - Android SDK statistics’, *AppBrain*. [Online]. Available: https://www.appbrain.com/stats/libraries/details/android_arch/android-architecture-components. [Accessed: 04-May-2019].

[27] Google, ‘TimingLogger’, *Android Developers*. [Online]. Available: https://developer.android.com/reference/android/util/TimingLogger. [Accessed: 18-Apr-2019].

[28] ‘Profile your app performance | Android Developers’. [Online]. Available: https://developer.android.com/studio/profile. [Accessed: 09-Apr-2019].

[29] A. E. Del La Torre and Y. Cheon, ‘Impacts of Java Language Features On the Memory Performance of Android Apps’, 2017.