Protokoll 26.3.2020:

Wiederholung und Theorie

In der Wiederholung vom 6.3 ging es um Fragments und Transactions mit denen man mithilfe von Navigation Item Listenern auf das richtige Fragment wechselt.

Das Speichern von Daten ist auf mehreren verschiedenen Wegen möglich. Interner Speicher ist spezifisch für die App zur Verfügung gestellt und wird beim Löschen der App ebenfalls entfernt. Eine andere Möglichkeit ist der externe Speicher, bei dem Berechtigungen benötigt werden, aber die Daten nach dem Löschen der App erhalten bleibt. Wenn diese Varianten nicht ausreichen, gibt es auch die Möglichkeit von Datenbanken oder Shared preferences. Shared Preferences sind Key-Value Pairs in einem internen XML File. Werte können hier über *.apply* (im Hintergrund) oder *.commit* (im Main Thread) gespeichert werden.

Wenn eine Datenbank als Speichermethode gewählt wurde kann SQLite verwendet werden, was ohne Konfiguration oder die Verwendung von einem Server verfügbar ist, da es bereits in Android eingebaut ist. Die Room Persistence Library kann stattdessen verwendet werden und bietet einen einfacheren und abstrakteren Zugriff auf die Datenbank. Room besteht aus der Datenbank, Entities und einem DAO. Mithilfe von Annotations können die gewünschten Daten persistiert werden.

Für das Lesen von Files aus dem Speicher werden Input und Output Streams benötigt, um die Daten aus der Datei zu lesen. Auch dafür werden ebenfalls Berechtigungen benötigt, die wiederrum im Manifest angegeben werden.

Praxis

Es wurde das aktuelle Fragment in Shared Prefernces gespeichert, um beim Öffnen der App wieder auf die Seite zurückkehren zu können. Danach wurde Room eingebaut, um es zu erleichtern Daten zu speichern. Dafür müssen zunächst die Abhängigkeiten für Room hinzugefügt werden. Danach wird die Location Klasse annotiert und eine LocationDAO Klasse erstellt, wo Abfragen auf die Datenbank durchgeführt werden. Zuletzt wird noch eine Datenbank Klasse benötigt, welches uns die DAO zu Verfügung stellt. Diese Klasse extends RoomDatabase und wird als Singleton implementiert, damit das Datenbank Objekt nicht ständig neu erstellt wird.