Quiz App
Programmablauf

Android App

Handling des FCM Tokens

Erklärung FCM Token

Ein FCM Token ist eine pro Android App pro Android Gerät eindeutige Zeichenkette. Sie wird verwendet, damit eine App auf einem Smartphone eindeutig referenziert werden kann. Benötigt wird das zum Senden von Push-Notifications an ein bestimmtes Gerät.

Aus diesem Grund muss der FCM Token zu jedem User der App gespeichert werden (in der Datenbanktabelle *user*).

Der FCM Token kann sich nach der Zeit ändern (selten, aber doch). Aber auch ein Handy-Wechsel eines registrierten Users führt zur Änderung des Tokens.
Sobald er sich geändert hat, muss er auch in der Datenbank geändert werden.

Holen des FCM Tokens

Durch die Google Firebase Library (auf Android) wird ein Task erstellt, der den FCM Token holt. Dieser funktioniert asynchron. Deshalb wurde ein Listener erstellt (*FCMTokenTaskCompleteListener*). Dieser wird benachrichtigt, wenn

- der FCM Token ermittelt wurde (nach dem Start der App)
- der FCM Token sich geändert hat (irgendwann)

Speichern des FCM Tokens

- **User ist bereits eingeloggt**
  FCM Token wird im User-Objekt des derzeit eingeloggten Users aktualisiert (User.getLoggedInUser())
- **User ist noch nicht eingeloggt**
  FCM Token wird in einer statischen Variabel zwischengespeichert (User.setLonelyFcmToken(„…“)).
  Sobald der User eingeloggt wurde (oder sich registriert hat), wird der Token auf das neue User Objekt übertragen bzw. in der Datenbank aktualisiert (falls nötig).

Aufbau der GUI

Die App-GUI wurde mit XML designt. Zu jeder XML-Datei gibt es eine Java-Klasse, die die Funktionalitäten der GUI umsetzt. Dabei gibt es zwei unterschiedliche „Typen“ von XML-Designs:

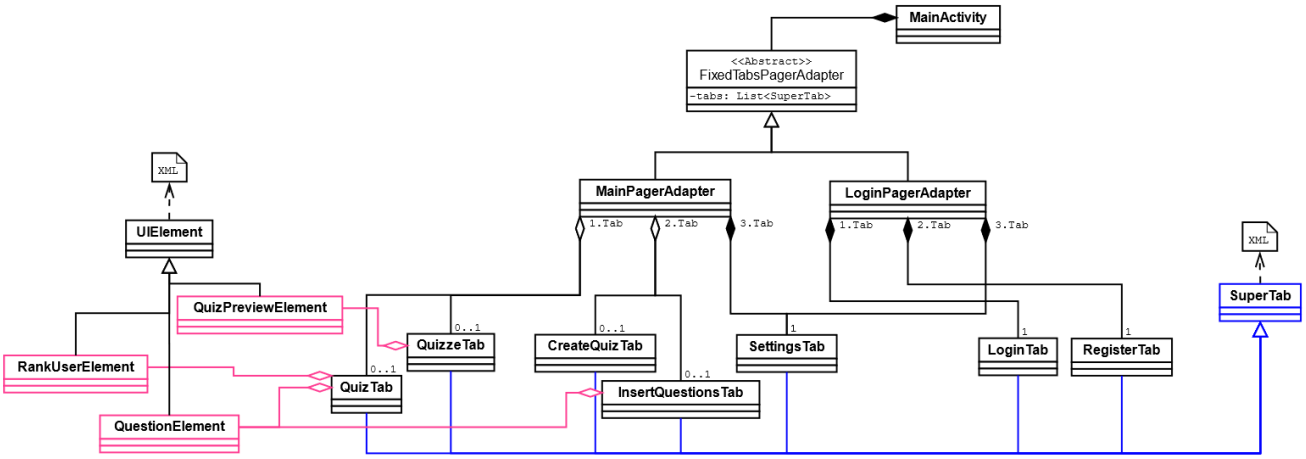
1. **Tab/Activity**
   Stellt einen ganzen Tab dar. Jeder Tab hat/ist eine Klasse, die von der Superklasse „*SuperTab*“ erbt.
   z.B. „*activity_login.xml*“, „*activity_quiz.xml*“, …
2. **Element**
   Stellt einen Teil einer GUI dar, der mehrmals verwendet wird. Jedes Element hat/ist eine Klasse, die von der Superklasse „*UIElement*“ erbt.
   „*UIElement*“ schreibt die Methode „*show*“ vor, die von den Sub-Klassen implementiert werden muss. Hier wird aus dem XML ein Android-GUI-Objekt erstellt und die benötigten Daten befüllt.
   Beispiele:
   - „*element_question.xml*“ für einen Frageblock mit Antwortmöglichkeiten
   - „*element_quizpreview.xml*“ für die Kurz-Darstellung eines Quiz im *QuizTab*.

Alles spielt sich in einer einzigen Activity ab (*MainActivity*). Dort gibt es 2 Haupt-Ansichten:

1. Login-Ansicht (alles **vor** dem Login/Registrieren)
2. Main-Ansicht (alles **nach** dem Login/ Registrieren)

Eine Haupt-Ansicht wird durch einen „*FixedTabsPagerAdapter*“ realisiert. Dieser verwaltet das Switchen und Anzeigen von verschiedenen Tabs. Für jede Haupt-Ansicht gibt es eine Sub-Klasse von „*FixedTabsPagerAdapter*“.

Jede Haupt-Ansicht hat zu jeder Zeit 3 Tabs (könnte aber in der Zukunft problemlos erweitert werden). Während des Programmablaufs können Tabs mit anderen (neuen) Tabs ausgetauscht werden.



In weiterer Folge gehört ein „*FixedTabsPagerAdapter*“ zu einem „*ViewPager*“.

Verbindung App -> Server

Die App verbindet sich mit *httpURLConnections* zum Server. Diese werden **asynchron** ausgeführt (bei **synchroner** Ausführung würde sich die App aufhängen während Daten vom Server geholt werden).

Für jede asynchrone Server-Anfrage gibt es eine eigene Klasse, die von „*SendToServletTask*“ erbt. Dort werden der Servletname und die benötigten Parameter angegeben.
Zusätzlich kann die „*onPostExecute*“ Methode überschrieben werden. Dort wird festgelegt, was passieren soll, nachdem der Task abgeschlossen wurde.

Connection Parameter

Daten, die die App an den Server sendet, sind URL-encodiert.

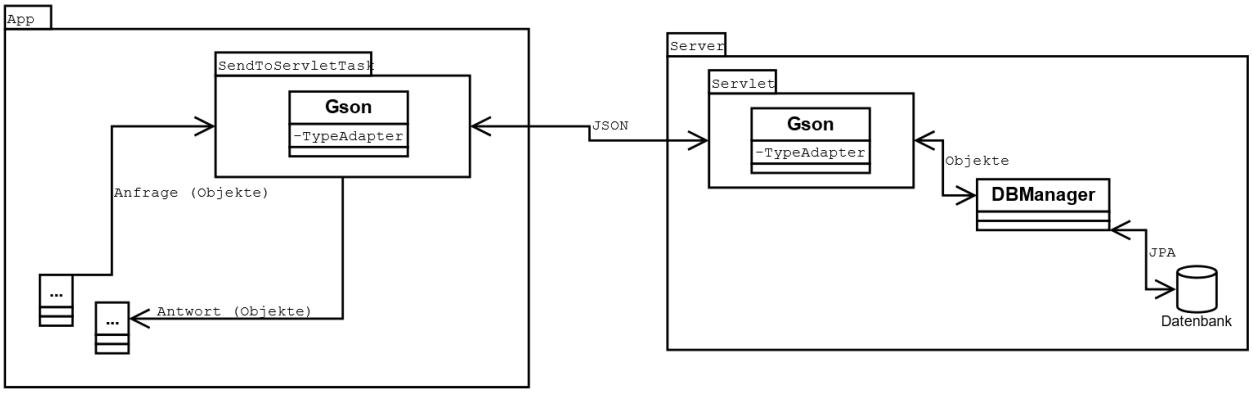
Beispiel: variable1=wert1&variable2=wert2

Steuerung der Tasks

Die Tasks können von einer beliebigen Stelle im Programm ausgeführt werden.

Beispiel:

> Nachdem alle Daten eines neuen Quiz vom User eingegeben wurden, wird der *CreateQuizTask* von einem Button-Listener ausgeführt.
>
> Nachdem der Task fertig ist, wird seine „*onPostExecute*“ Methode ausgeführt. Dort wird dann eine kurze Popup-Nachricht für den User angezeigt, ein anderer Tab geladen und ein weiterer Task ausgeführt (zum Aktualisieren der Quizze-Liste).



Datenaustausch Server <-> App

Der Datenaustausch zwischen Server und App erfolgt im **JSON Format**. Objekte werden also in einen JSON String umgewandelt, an das Zielgerät (Server bzw. App) gesendet, und dort zur Weiterverarbeitung wieder in ein Objekt umgewandelt.

Dafür wird die *GSON* Library verwendet.

Steuerung der JSON Umwandlung

Nicht in jedem Use-Case werden alle Variablen eines Objektes benötigt.
Zum Beispiel hat ein User-Objekt aus der Datenbank eine Liste mit allen Antworten, die er jemals bei irgendeinem Quiz angekreuzt hat. Und jedes dieser Antwort-Objekte hat wiederum ein dazugehöriges Frage-Objekt, usw.

Deshalb werden *TypeAdapter* (von der GSON Library) verwendet. Eine Klasse, die von *TypeAdapter* erbt, gibt an, wie ein bestimmter Datentyp in ein JSON geschrieben werden soll und wie ein JSON String wieder zurück in den Datentyp gewandelt werden soll.

Für die meisten Use-Cases wurde ein neuer *TypeAdapter* erstellt. Ein *TypeAdapter* muss bei einem *GSON* Objekt registriert werden, damit er verwendet wird.