



QUIZAPP

ארכיטקטורה והסבר על הקוד

תוכן עניינים

| | |
|---|----------------------------------|
| 2 | תוספים |
| 3 | FireBase |
| 4 | פרויקט FireBase Functions |
| 5 | המחלקה QuizApp |
| 6 | המחלקה FirebaseAuth |
| 7 | המחלקה BroadcastReceiver Battery |
| 7 | המחלקה AlertDialog |
| 7 | המחלקה Question |
| 8 | המחלקה Quiz |
| 8 | המחלקה QuestionFragment |
| 8 | המחלקה ScoreFragment |
| 9 | המחלקה MainActivity |
| 9 | המחלקה QuestionListActivity |
| 9 | המחלקה QuestionAdapter |

| Project | 2xActivities + | Listeners: two methods | Dialog | Menu with two entries | Adapter | Long term data | Broadcast | Service |
|---------|-------------------------------------------|----------------------------------------------------------------------------------------|------------|-------------------------------------------------------------|----------------|----------------|-------------|--------------|
| QuizApp | 1. Question Quiz 2. All questions view | 1. QuestionFragment: class implements listener. 2. FireBaseHandler: new listener... | Everywhere | -Load new quiz. -Reset all. -Report mistake. -Exit | Question build | Firebase | סוללה נמוכה | notification |

תוספים

להלן התוספים שהשתמשנו בהם.

- [Zoomage](#) – תוסף המאפשר תצוגת תמונות באופן חכם: הגדלה / הקטנה והזזה שלהם.
- [Android Universal Image Loader](#) – תוסף המאפשר טעינת תמונות מהירה וקלה.
- [Firebase](#) – טעינת שאלות ממסד, קבלת התראות בעת עדכון המסד.

FireBase

הגישה ל FireBase הינה דרך: <https://console.firebase.google.com>

החשבון שאיתו נעשה שימוש הינו:

RKCSQuizApp@gmail.com

quizapp12

גישה לשאלות דרך סרגל כלים מצד שמאל: **Develop -> Database**

השאלות נשמרות בצורת קובץ json. דוגמא לשאלה:

```
{
  "choices" : {
    "A" : "עקביות (Consistency).",
    "B" : "אפשרות (Affordance).",
    "C" : "משוב (Feedback).",
    "D" : "מיפוי (Mapping).",
  },
  "correct" : "עקביות (Consistency).",
  "explanation" : "נעשה שימוש לא עקבי בפונטים, יישור טקסט לא אחיד, שימוש באייקונים לא ברורים",
  "image" : "https://raw.githubusercontent.com/Romansko/HCI/master/img/q/sign.jpg",
  "question" : "איזה עקרון שימושיות של נורמן, מופר באופן מובהק במסך הבא:"
}
```

הסבר לשאלה / תמונה לא חובה ואם השדות ריקים, פשוט לא יוצגו באפליקציה.

כל השאלות שמורות תחת צומת יחיד (Root Node) בשם המפתח **Hebrew**.

בנוסף, נעשה שימוש ב **FireBase Cloud Messaging (FCM) Functions** על מנת לקבל התראה על עדכון

במסד. אל פונקציות אלו, ניתן לגשת גם דרך סרגל הכלים מצד שמאל: **Develop -> Functions**

שם נראה כי קיימת פונקציה **dataBaseUpdateNotification** (שאנו הגדרנו) שמאזינה לכל כתיבה אל מסד השאלות:

| Function | Trigger |
|-----------------------|-----------------------------------------------------------------------------------------------------------|
| dataBaseUpdateNoti... |  ref.write hebrew |

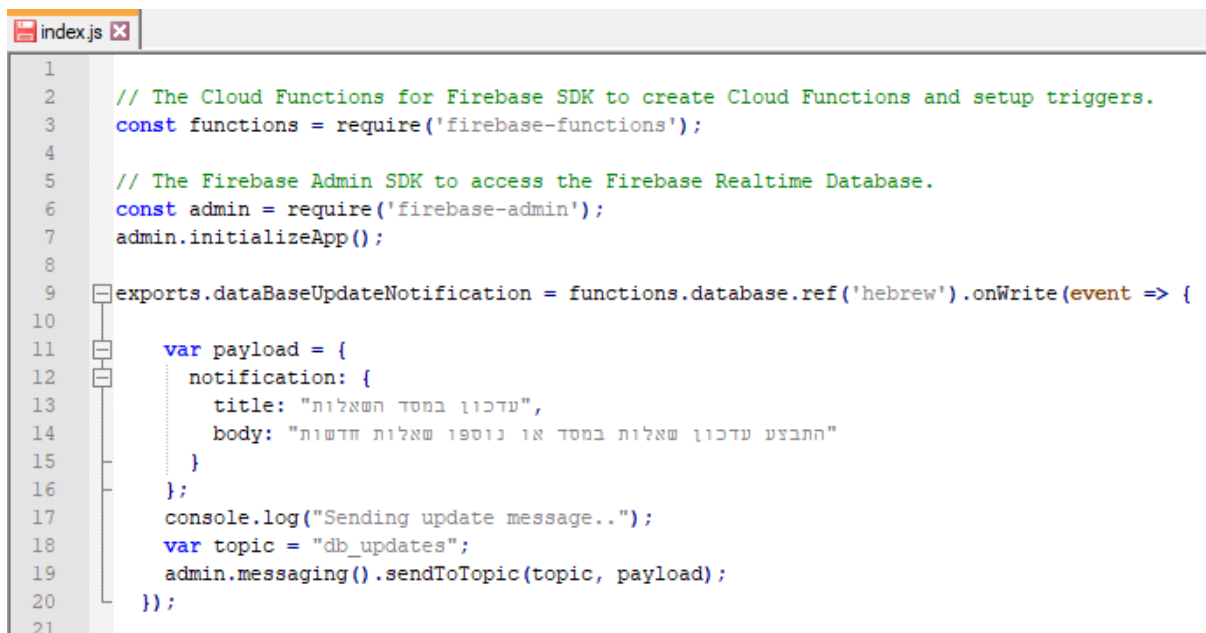
פרויקט Firebase Functions

בתוך הפרויקט של QuizApp, יש פרויקט (לא Android) תחת תיקיית Firebase שנוצר בכדי להוסיף את הפונקציה שמאזינה על מסד השאלות.

הפרויקט נוצר ע"י הפקודות הבאות בTerminal (שמופנה לתיקייה הנ"ל):

```
npm install -g firebase-tools
firebase login --interactive
firebase init
```

את הפונקציה **dataBaseUpdateNotification** הוספנו בקובץ **index.js** QuizApp/Firebase/functions/



```
1
2 // The Cloud Functions for Firebase SDK to create Cloud Functions and setup triggers.
3 const functions = require('firebase-functions');
4
5 // The Firebase Admin SDK to access the Firebase Realtime Database.
6 const admin = require('firebase-admin');
7 admin.initializeApp();
8
9 exports.dataBaseUpdateNotification = functions.database.ref('hebrew').onWrite(event => {
10
11     var payload = {
12         notification: {
13             title: "עדכון במסד השאלות",
14             body: "התבצע עדכון שאלות במסד או נוספו שאלות חדשות"
15         }
16     };
17     console.log("Sending update message..");
18     var topic = "db_updates";
19     admin.messaging().sendToTopic(topic, payload);
20 });
21
```

נשים לב למפתח "db_updates" (Topic). באפליקציה שלנו הגדרנו מאזין לפי Topic ולכן המפתח חשוב. (Firebase תומכת באפשרות לעשות גם האזנה לפי Token או לפי Groups).

המחלקה QuizApp

```
package com.rkcodesolution.quizapp;
import android.app.Application;
import com.nostr3l3.universallimageloader.core.ImageLoader;
import com.nostr3l3.universallimageloader.core.ImageLoaderConfiguration;

public class QuizApp extends Application {
    public static final String QATAG = "QATAG";           // Tag for debugging.
    public static ImageLoader sImageLoader;               // for loading images.

    @Override
    public void onCreate() {
        super.onCreate();
        ImageLoaderConfiguration config = new ImageLoaderConfiguration
            .Builder(this)
            .build();
        sImageLoader = ImageLoader.getInstance();
        sImageLoader.init(config);
    }
}
```

זוהי נקודת הכניסה לאפליקציה. השימוש במחלקה זאת הוא בכדי לאתחל **ImageLoader** (תוסף) סטטי בשביל טעינת תמונות קלה יותר בהמשך.

כמו כן, כאן שמורה המחרוזת **QATAG** בשביל ה-Log.

הגישה לטוען התמונות ולמחרוזת היא בצורה סטטית.

המחלקה FirebaseHandler

המחלקה יורשת מ-FirebaseMessagingService והיא אחראית לתקשורת בין האפליקציה לבין מסד FireBasen.

הפונקציות העיקריות של המחלקה היא טעינת שאלות משרת FireBasen. זה נעשה על ידי הפונקציה `getQuestions`.

```
public static void getQuestions(final IResponse caller, boolean hardReset)
```

פונקציה זו אחראית על קריאת השאלות מהשרת באינטרנט. `IResponse` זהו ממשק של המחלקה שצריך לממש כאשר מחלקה אחרת קוראת לפונקציה בכדי לקבל את השאלות. **הממשק** נראה כך:

```
public interface IResponse {  
    void onFirebaseResponse(ArrayList<Question> questions);  
}
```

כאשר הפונקציה מקבלת את השאלות בצורת אובייקטים גנריים מהמסד, היא בונה אותם לפי צרכי האפליקציה באמצעות הפונקציה הסטטית `parseQuestions`. מוחזר מערך מסוג `Question`.

```
private static ArrayList<Question> parseQuestions(Object fbObject)
```

כאשר השאלות מוכנות, המחלקה שומרת עותק של מערך השאלות בצורה סטטית, כך שבפעם הבאה שהשאלות יתבקשו, היא תחזיר עותק זה ולא תיגש שוב לאינטרנט ותבזבז זמן על תקשורת ובניית אובייקטים.

אם הדגל `hardReset` הוא `true`, אז תתבצע גישה למסד האינטרנטי והעותק השמור ידרס.

הפונקציות השנייה של המחלקה היא האזנה לשינויים במסד FireBasen. זה נעשה ע"י הפונקציה `subscribeToUpdate` שרושמת את האפליקציה להאזנה עם המפתח `"qb_updates"`.

כאשר האפליקציה פתוחה וישנו שינוי בשרת, הפונקציה שתקרא במחלקה היא `onMessageReceived`. זוהי פונקציה שנדרסת מהמחלקה `FirebaseMessagingService`.

`onMessageReceived` תודיע למשתמש שהיה שינוי במסד ותקפיץ popup אם לטעון מחדש את השאלות או לא. אם המשתמש יבחר לא, לא יקרה כלום. אם כן, אז השאלות יטענו מחדש מהמסד. יש צורך לממש את הממשק הפנימי `ISubscribeHandler` בשביל הפונקציות הזו. השאלה נשאלת והטעינה לא מבוצעת אוטומטית בשביל לא להרוס למשתמש את ההתקדמות שביצע בסבב השאלות.

כאשר האפליקציה **כבויה**, אם היה שינוי בשרת אז הפלאפון יקבל `push notification` המודיע על שינויים במסד. זה נעשה ע"י service שהוגדר ב `manifest`:

```
<service android:name=".FirebaseHandler">  
    <intent-filter>  
        <action android:name="com.google.firebase.MESSAGING_EVENT" />  
    </intent-filter>  
</service>
```

המחלקה BatteryBroadcastReceiver

מחלקה זו יורשת מ **BroadcastReceiver** והיא אחראית להודיע למשתמש על אחוז סוללה נמוך. האחוז מוגדר HardCoded דרך השדה **THRESHOLD** אשר כרגע מוגדר להיות 50.

הרישום להאזנה על הסוללה נעשה דרך הבנאי של המחלקה שמקבל כפרמטר את ה **Activity** שקורא לו.

כל שינוי בסוללה יקרא לפונקציה **onReceive**. שם, אם האחוז קטן מ **THRESHOLD** שהוגדר, תוצג למשתמש הודעת **popup** שתתריע על האחוז הנמוך ותשאל אותו האם הוא רוצה לצאת מאפליקציה או להמשיך.

אם המשתמש יבחר להמשיך, הרישום למאזין יתבטל ולא יתחדש עד שהאפליקציה תופעל מחדש.

המחלקה ExitDialog

מחלקה זו יורשת מ **DialogFragment**. בעת יצירת אובייקט חדש מסוג זה, יוצר דיאלוג שיוודא שהמשתמש אכן רוצה לצאת. אם ילחץ כן, האפליקציה תיסגר. לשם כך יש את הממשק הפנימי **IExitListener** המכיל את הפונקציה **onExitPressed** שתיקרא כאשר המשתמש יבחר לצאת.

המחלקה Question

מחלקה זו מייצגת שאלה במערכת. היא מכילה את מחזורות השאלה, תשובות, הסבר וקישור לתמונה.

המחלקה מממשת את הממשק **Clonable** עבור העתקת השאלה. המחלקה מממשת את הממשק **Parcelable** בכדי שיהיה ניתן לשמור שאלות כ **Bundle** בעת העברת ל **Activity** או **Fragment** אחר.

המחלקה Quiz

זוהי מחלקת הליבה של האפליקציה. היא מכילה את הלוגיקה העיקרית. מחלקה זו מנהלת את הבחן הנוכחי שהמשתמש נבחן עליו.

הבנאי של המחלקה הוא פרטי וקבלת עותק נעשה ע"י הפונקציה `getInstance`. המחלקה אחראית לקרוא את העדפת המשתמש לגבי מספר שאלות בבוחן. אם לא הוגדר, ברירת המחדל היא 10.

המחלקה אחראית למלא בוחן בשאלות שהמשתמש עדיין לא נבחן בהם, כאשר סבב השאלות נגמר, המחלקה טוענת מחדש את העותק הסטטי של השאלות שנשמר מהמסד. בנוסף, המחלקה מציגה את השאלות והתשובות בסדר אקראי.

המחלקה QuestionFragment

מחלקה זו יורשת מ-`Fragment` והיא אחראית להציג שאלה למשתמש. כלומר היא מציגה את השאלה, תשובות ותמונה אם יש. אחרי מענה המשתמש, אם קיים הסבר לשאלה אז גם הוא יוצג.

ניתן למלא את ה-GUI ע"י הפונקציה `populateQuestion`.

יש שני קבצי XML שמשויכים עם מחלקה זו. אחד עבור `portrait` ואחד עבור `landscape`. שניהם קרויים בשם `fragment_question.xml`. קבצי XML אלו, משתמשים בקובץ `zoomage.xml` שהוא בעצם מעטפת לתוסף `zoomage` ומיועד עבור הצגת תמונות. הוספת `zoomage` נעשית באופן דינאמי.

למחלקה יש מחלקה פנימית בשם `ChoiceClickListener` אשר מממשת את הממשק `View.OnClickListener` ומטרתה היא לצבוע בכחול תשובות שהמשתמש לוחץ עליהם.

כמו כן למחלקה יש ממשק פנימי בשם `ICorrectAnswerListener` המאזין ללחיצת תשובה נכונה ומפעיל את הפונקציה `correctQuestionPressed`.

המחלקה ScoreFragment

מחלקה זו יורשת מ-`Fragment`. היא אחראית להציג את התוצאה (ניקוד) של הבוחן בסיומו.

קבלת אובייקט מסוג זה נעשה ע"י הפונקציה `newInstance`.

המחלקה משתמשת בקבצי `fragment_score.xml` עבור `portrait / landscape`.

המחלקה MainActivity

זהו ה-Activity הראשי של האפליקציה. הוא משתמש בקבצי `activity_main.xml` עבור `portrait / landscape`. קבצי ה-`Fragment` של ה-`Score / Question` מוכלים בקבצי XML אלה באופן דינאמי.

המחלקה מממשת את הממשק `FireBaseHandler.IResponse` עבור קבלת השאלות מ-Firebase, ובעת קבלה היא יוצרת אובייקט מסוג `Quiz` חדש.

המחלקה מממשת את הממשק `FireBaseHandler.ISubscribeHandler` עבור הצגת דיאלוג כאשר יש שינוי במסד.

המחלקה מממשת את `ExitDialog.IExitListener` עבור האזנה ליציאה מהאפליקציה.

המחלקה מממשת את `QuestionFragment.ICorrectAnswerListener` עבור טיפול בלחיצה על תשובה נכונה – הוספת ניקוד דרך `Quiz`. כלומר ה-`QuestionFragment` מתריע ל-`MainActivity` שהמשתמש לחץ תשובה נכונה, וזו מעבירה את זה למחלקת `Quiz` ששומרת את הניקוד והתוצאה.

המחלקה הזו גם שומרת את לוגיקת ה-`Menu` בתוכה. נעשה שימוש בקובץ `menu.xml` ובעת לחיצה על אחד מהפריטים של התפריט, הקוד המטפל נמצא בתוך מחלקה זו.

המחלקה QuestionListActivity

מחלקה זו אחראית לפרוס את השאלות בצורה כללית כמתואר בהצעה:

עבור פריט בודד של שאלה, נעשה שימוש בקובץ `question_view.xml`.

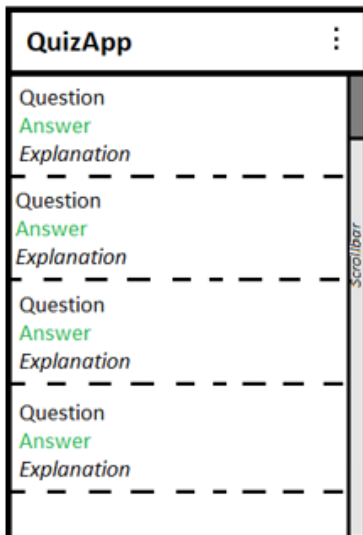
המחלקה פורסת את השאלות, התשובות ואת ההסברים ותמונות אם קיימים ב-`ListActivity` חדש.

אם קיימת תמונה, בעת לחיצה עליה יפתח התוסף `zoomage` שציג אותה ויאפשר משחק עם התמונה. (בתצוגה הרגילה לא ניתן כיוון שזה תמונה רגילה ולא `zoomage`). הוספת מאזין הלחיצה נקבע במחלקה זו.

המחלקה הזו עושה שימוש במחלקת `QuestionAdapter`.

העברת השאלות לתצוגה מ-`MainActivity` ל-`ListActivity` דרשה שהשאלות יהיו מסוג `Parcelable`.

(תצוגת כל השאלות)



המחלקה QuestionAdapter

מחלקה זו עובדת עבור מחלקת `QuestionListActivity`. היא אחראית על לוגיקת פריט שאלה בודד בתוך כל הרשימה. כלומר מחלקה זו אחראית להציג עבור כל שאלה את מחרוזת השאלה, תשובה הסבר ותמונות אם יש. כמו כן שם נקבע צבע ה-`Background` עבור כל פריט: אם הפריט במיקום זוגי – צבע לבן. אחרת, תכלת.