Brain Battle  
מגיש: יהודה אליסף

ת.ז. 328193602

מורה: חגי סוויד

חלופה: תכנות לטלפונים ניידים

תאריך הגשה: 12.6.2023

תוכן

[מבוא 4](#_Toc137342729)

[הרקע לפרויקט 4](#_Toc137342730)

[תהליך המחקר 4](#_Toc137342731)

[טכנולוגיות 4](#_Toc137342732)

[אתגרים מרכזיים 4](#_Toc137342733)

[תיאור תחום הידע 5](#_Toc137342734)

[אובייקטים נחוצים 5](#_Toc137342735)

[סוגי נתונים 5](#_Toc137342736)

[פעולות על המידע 5](#_Toc137342737)

[ארכיטקטורה 6](#_Toc137342738)

[מסכי הפרויקט 6](#_Toc137342739)

[מסך הרשמה והתחברות (מסך הפתיחה) 6](#_Toc137342740)

[מסך יצירת משחק 7](#_Toc137342741)

[מסך קוד המשחק 8](#_Toc137342742)

[מסך הצטרפות למשחק 9](#_Toc137342743)

[מסך המשחק 10](#_Toc137342744)

[מסך סיום משחק 11](#_Toc137342745)

[מסך ניקוד 12](#_Toc137342746)

[מסך הגדרות 13](#_Toc137342747)

[תרשים מסכים 14](#_Toc137342748)

[14](#_Toc137342749)

[14](#_Toc137342750)

[14](#_Toc137342751)

[מחלקות הפרויקט 15](#_Toc137342752)

[מימוש הפרויקט 16](#_Toc137342753)

[מחלקות מסכים (Fragment/Activity) 16](#_Toc137342754)

[LoginActivity 16](#_Toc137342755)

[MainMenuActivity 18](#_Toc137342756)

[NewGameFragment 20](#_Toc137342757)

[JoinGameFragment 23](#_Toc137342758)

[ScoreFragment 24](#_Toc137342759)

[GameIdFragment 26](#_Toc137342760)

[GameActivity 27](#_Toc137342761)

[EndGameFragment 32](#_Toc137342762)

[מחלקות עזר 33](#_Toc137342763)

[Game 33](#_Toc137342764)

[Question 33](#_Toc137342765)

[User 34](#_Toc137342766)

[Player 34](#_Toc137342767)

[HttpQuestionFetcher 34](#_Toc137342768)

[GameViewModel 38](#_Toc137342769)

[פעולות 38](#_Toc137342770)

[ScoreListViewHolder 40](#_Toc137342771)

[ScoreListAdapter 40](#_Toc137342772)

[קבצי תצורה 41](#_Toc137342773)

[קבצי Layout 41](#_Toc137342774)

[קבצי Menu 41](#_Toc137342775)

[קבצי Navigation 42](#_Toc137342776)

[בסיסי נתונים 42](#_Toc137342777)

[SharedPreferences 42](#_Toc137342778)

[Firestore 43](#_Toc137342779)

[Open Trivia DB 44](#_Toc137342780)

[מדריך למשתמש 45](#_Toc137342781)

[תיאור כללי 45](#_Toc137342782)

[הגבלות 45](#_Toc137342783)

[מידע נוסף 45](#_Toc137342784)

[הרשאות 45](#_Toc137342785)

[הצהרות 45](#_Toc137342786)

[גרסת Android מינימלית 45](#_Toc137342787)

[גרסאות ומכשירים שעליהם נבדקה האפליקציה 45](#_Toc137342788)

[רפלקציה 46](#_Toc137342789)

[ביביליוגרפיה 47](#_Toc137342790)

[נספחים 48](#_Toc137342791)

# מבוא

## הרקע לפרויקט

הפרויקט Brain Battle (קרב מוחות) הוא משחק טריוויה ל2 שחקנים. השחקנים מתחרים ביניהם אונליין, עונים על שאלות טריוויה שונות וצוברים נקודות. בסיום המשחק, השחקנים יכולים לראות סטטיסטיקות שונות כמו מספר התשובות הנכונות והשגויות, ואת מצב הניקוד שלהם ביחס לשחקנים אחרים.

המשחק מיועד לטווח גילאים רחב, משום שיש אפשרות לבחור את רמת הקושי הרצויה (כמובן שרמת הקושי משפיעה גם על הניקוד).להערכתי, תחום הגילאים המומלץ הוא 10 ומעלה.

בחרתי ליצור דווקא משחק כי רציתי ליהנות כמה שיותר מיצירת הפרויקט, ויצירת משחקים מוסיפה גורם של הנאה לפיתוח. בחרתי דווקא בטריוויה כי אני אוהב את המשחק, ונהנה לבחון את הידע הכללי שלי בתחומים שונים.

## תהליך המחקר

כיום, ישנן אפליקציות טריוויה רבות בשוק, גם ל2 שחקנים וגם למספרי שחקנים שונים. עם זאת, זה לא מפריע לי, כי המטרה העיקרית שלי בפיתוח במשחק היא להעמיק את הידע שלי בפיתוח אפליקציות.

## טכנולוגיות

במהלך יצירת הפרויקט השתמשתי במספר טכנולוגיות שאינן חלק מתוכנית הלימודים:

1. HTTP – פרוטוקול תקשורת שמיועד לשליחת וקבלת מידע לאתרי אינטרנט ומהם. השתמשתי בHTTP כדי להוריד שאלות טריוויה ממסד נתונים בשם [Open Trivia Database](https://opentdb.com/), במטרה להשיג כמות גדולה של שאלות בנושאים מגוונים ורמות קושי שונות. פרמטרי כמו מספר השאלות, רמת הקושי וכו' נשלחים בHTTP בעזרת פרמטרי GET.
2. JSON Deserialization – השאלות במסד הנתונים מתקבלות בפורמט JSON. כדי להפוך את השאלות שהתקבלו לאובייקט Question (אובייקט שיצרתי שמייצג שאלה, עליו אפרט בהמשך) יש צורך בדה-סריאליזציה של המידע, כלומר הפיכה שלו מJSON לאובייקט קוד.
3. Firebase – שירות של גוגל שמאפשר מספר פיצ'רים כמו ניהול ההתחברות וההרשמה של משתמשים, ושרת אחסון בשם Firestore שמאפשר לשמור מידע בענן. השתמשתי בFirestore בשביל לשמור את נתונים המשחקים ולסנכרן אותם בזמן אמת, ובשביל לשמור את רשימת המשתמשים ופרטים שלהם.

## אתגרים מרכזיים

היו לי לא מעט אתגרים בזמן העבודה על הפרויקט.

בגלל שהמשחק נערך אונליין בין 2 שחקנים, ואף אחד מהשחקנים לא יכול לשחק בלי השני, הייתי צריך להשתמש ב2 מכשירי פלאפון. לכן, יצרתי אמולטור שמדמה מכשיר פלאפון, והרצתי את האפליקציה גם בו וגם במכשיר הפיזי.

קושי נוסף שהתמודדתי איתו הוא נגרם מכך שהמשחק דורש חיבור אינטרנט רציף. אם באמצע המשחק אחד השחקנים מתנתק, האפליקציה יכולה להיתקע או לקרוס. כדי לפתור את הבעיה, הוספתי לכל תקשורת עם הFirestore פונקציית OnFailure שקובעת מה לעשות במידה וחיבור האינטרנט קרס (למשל: הצגת הודעת שגיאה למשתמש וחזרה למסך הראשי). בנוסף, ההכנסתי את בקשת הHTTP לבלוק try catch, ובמידה וקרתה שגיאה – הודעתי למשתמש שלא ניתן להוריד את השאלות.

# תיאור תחום הידע

## אובייקטים נחוצים

Game: אובייקט הכולל את כל המידע של המשחק. כולל בתוכו את 2 השחקנים ורשימה של שאלות (על מבנה השחקנים והשאלות אפרט בהמשך), ואת הID של המשחק.

User: מייצג משתמש. כולל בתוכו את המידע של המשתמש (שם משתמש, ID וסטטיסטיקות של ניקוד).

Player: מייצג שחקן במשחק. יורש מUser. מוסיף למשתמש רשימה של השאלות הנכונות והשגויות שהוא ענה במשחק.

Question: מייצג שאלה. כולל בתוכו את התשובות השגויות והתשובה הנכונה, קטגוריית השאלה ורמת הקושי שלה.

## סוגי נתונים

בתוך מחלקת Game, השאלות שמורות ברשימה של אובייקטים מסוג Question.

במחלקת Question, התשובות שמורות ברשימה של סטרינגים, והאינדקס של התשובה הנכונה שמור בint. הקטגוריה ורמת הקושי שמורות כל אחת בenum משל עצמה:

enum DifficultyLevel{

EASY,

MEDIUM,

HARD

}

enum Category{

ALL,

GENERAL\_KNOWLEDGE,

SCIENCE,

COMPUTER\_SCIENCE

{

## פעולות על המידע

Game: יצירת משחק, סיום משחק, גטרים וסטרים.

User: יצירת משתמש, מחיקת משתמש, גטרים וסטרים.

Player: יצירת שחקן, חישוב של כמות התשובות הנכונות/שגויות במשחק, מתודת equals(), גטרים וסטרים.

Question: יצירת שאלה, גטרים וסטרים.

# ארכיטקטורה

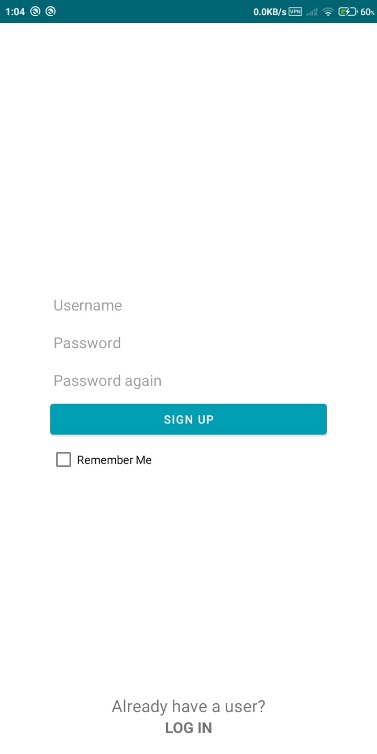
## מסכי הפרויקט

### מסך הרשמה והתחברות (מסך הפתיחה)

במצב התחברות, המסך מציג 2 EditTextים בשביל שם משתמש וסיסמא. במצב הרשמה, המסך מציג EditText נוסף שמיועד להכנסת הסיסמא שוב.

Button

מעבר בין המצבים מתבצע באמצעות לחיצה על הlabel שבתחתית המסך (SIGN UP/LOG IN).

תמונה שמכילה טקסט, צילום מסך, תוכנה, מערכת הפעלה

התיאור נוצר באופן אוטומטי

Button

הרשמה

בודק אם השם משתמש והסיסמא תקינים, ואם כן יוצר משתמש ומעביר למסך יצירת משחק

Button

התחברות

בודק בפיירבייס אם השם משתמש והסיסמא נכונים, ואם כן מעביר למסך יצירת משחק

EditText

הכנסת סיסמא פעם נוספת

EditTexts

הכנסת שם משתמש וסיסמא

CheckBox

קובע האם לשמור את שם המשתמש והסיסמא בהתחברות הבאה

TextView (Clickable)

מעבר ממצב הרשמה למצב התחברות ולהיפך

### מסך יצירת משחק

מאפשר למשתמש ליצור משחק חדש, ולבחור את הקטגוריה, רמת הקושי ומספר השאלות.

אם בפתיחת האפליקציה המשתמש כבר מחובר, זה יהיה מסך הפתיחה.

Buttons

בחירת הקטגוריה של המשחק מבין הבאות:

כל הקטגוריות, ידע כללי, מדע, מדעי המחשב

מסך זה הוא Fragment בתוך הActivity של בתפריט הראשי,

מעבר לFragmentים אחרים מתבצע באמצעות לחיצה על כפתורי הניווט למטה

או החלקה ימינה ושמאלה.

תמונה שמכילה טקסט, צילום מסך, תוכנה, דף אינטרנט

התיאור נוצר באופן אוטומטי

Buttons

בחירת רמת הקושי של המשחק:

קל, בינוני או קשה

TextView

הצגת כמות השאלות:

מינימום 2, מקסימום 20.

Buttons

בחירת כמות השאלות:

כפתור להגדלה וכפתור להקטנה של מספר השאלות

Buttons

בחירת רמת הקושי של המשחק:

קל, בינוני או קשה

Button

מעבר למסך ניקוד

Button

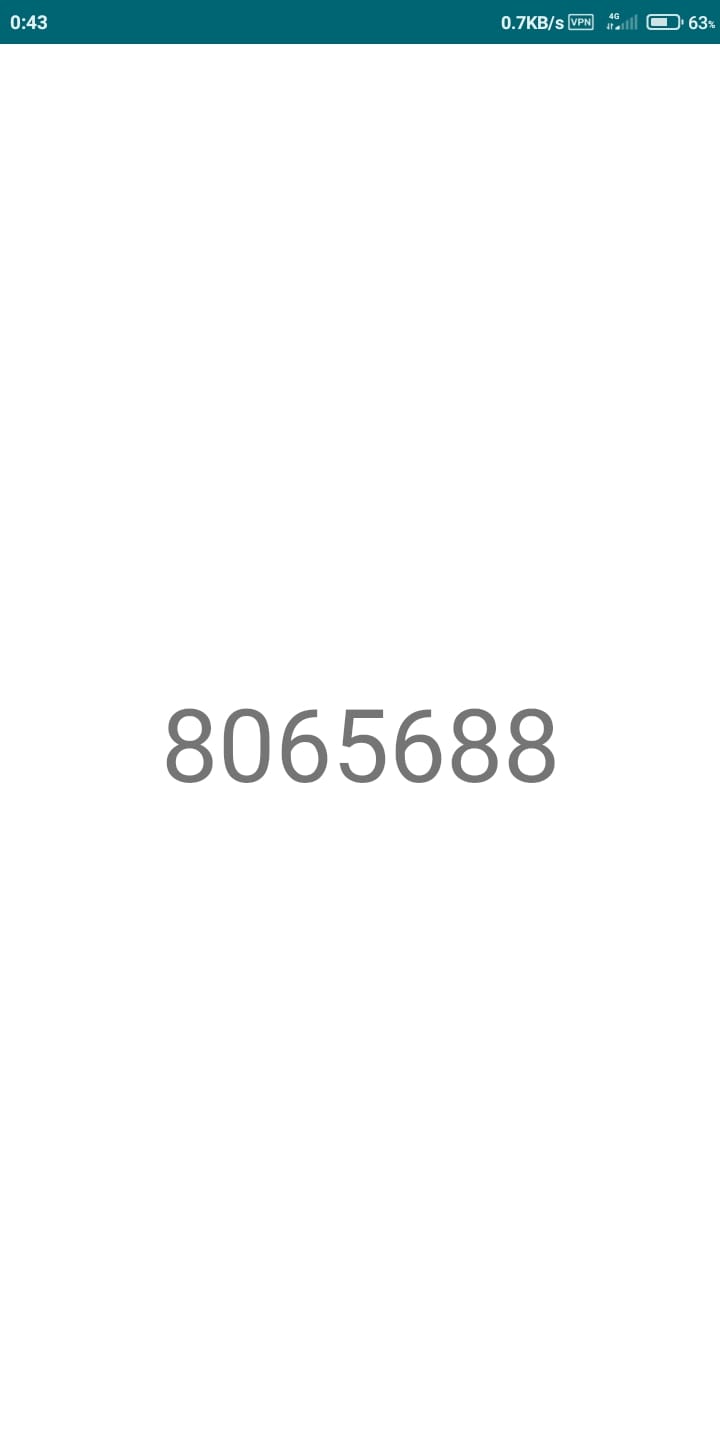
מעבר למסך הצטרפות למשחק

Button

מעבר למסך הגדרות

## מסך קוד המשחק

לאחר יצירת המשחק, יופיע מסך ובו קוד אותו צריך המשתמש השני להכניס כדי להתחבר.

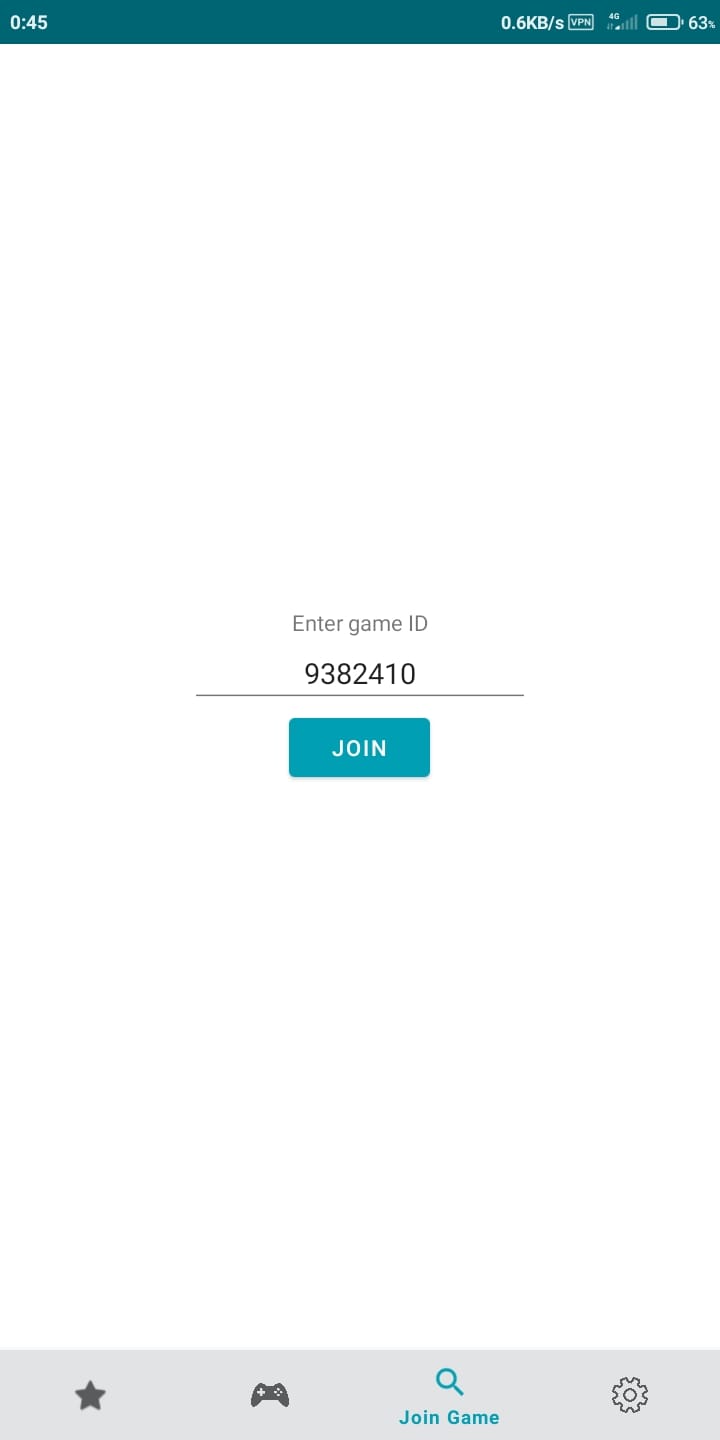


TextView

הצגת קוד המשחק

## מסך הצטרפות למשחק

אחרי שהשחקן הראשון יצר את המשחק, השחקן השני יכניס במסך זה את קוד המשחק כדי להצטרך אליו.



Button

הצטרפות למשחק

EditText

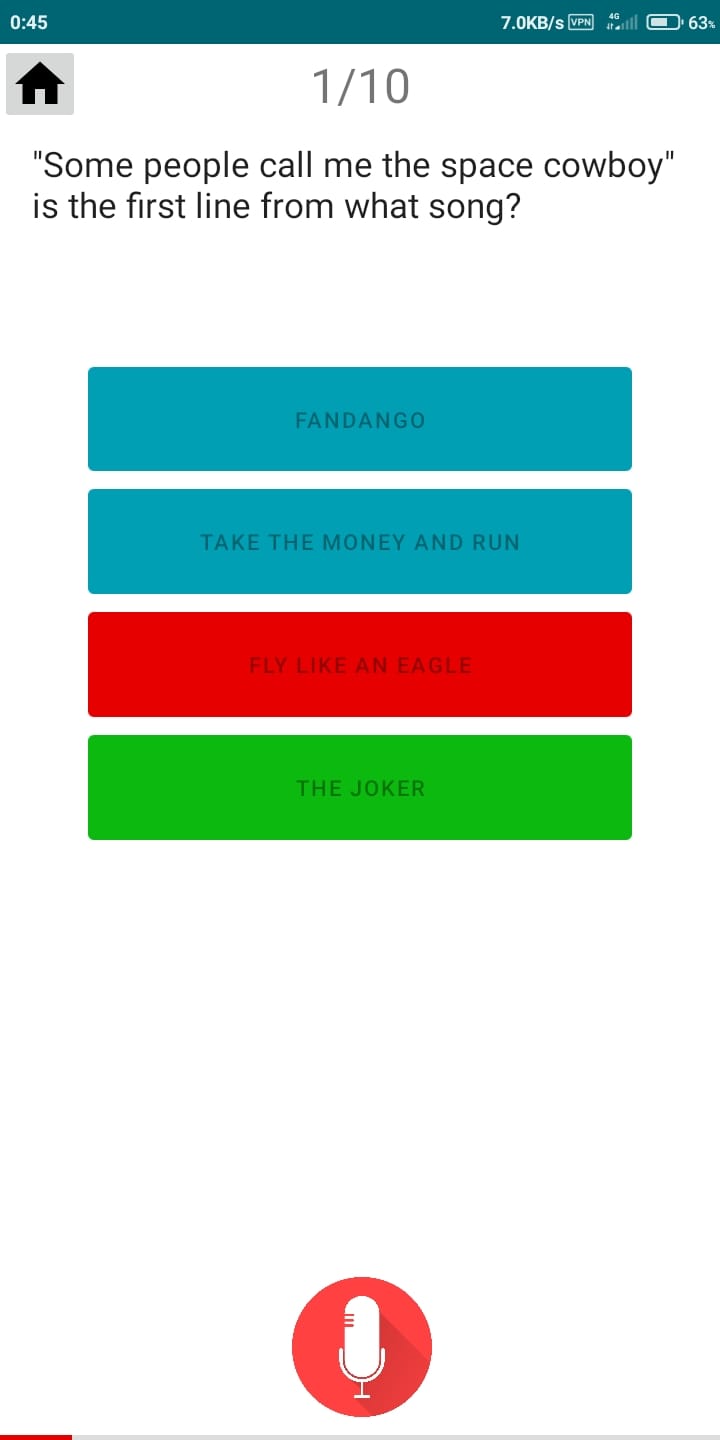
הכנסת קוד המשחק

## מסך המשחק

כששני השחקנים מחוברים, הם מועברים למסך זה, בו המשחק עצמו מתנהל.

Button

חזרה לתפריט הראשי



ImageButton

הקלטת תשובה

המשתמש צריך ללחוץ על הכפתור ולומר את מספר התשובה

(one, two, three, four)

הפיצ'ר עדיין לא ממומש

Canvas

הצגת ההתקדמות של היריב והאם הוא צדק או טעה (ירוק/אדום בהתאמה)

TextView

הצגת השאלה

TextView

הצגת מספר השאלה הנוכחית וסך השאלות

Buttons

כפתורי תשובות

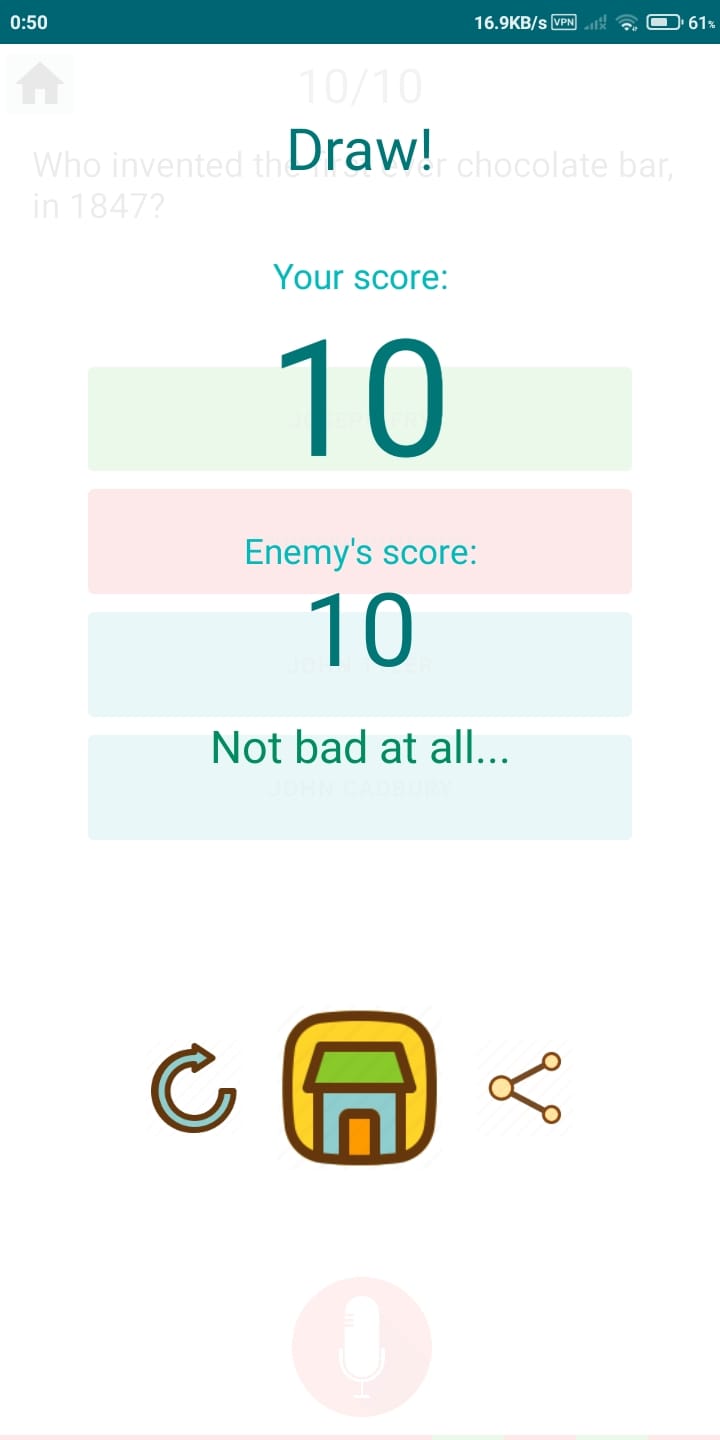
המשתמשים צריכים ללחוץ על התשובה הנכונה. לאחר מכן, הכפתור ייצבע באדום או ירוק בהתאם לתשובה

## מסך סיום משחק

בסוף המשחק, יופיע מסך זה, המציג את הניקוד ואת המנצח במשחק.

TextView

הצגת השחקן המנצח



ImageButton

שיתוף תוצאות משחק

ImageButton

חזרה למסך הבית

ImageButton

יצירת משחק חדש עם אותם פרמטרים (כמות שאלות וכו')

TextView

הצגת הניקוד של היריב

TextView

הצגת הניקוד שלך

## מסך ניקוד

במסך זה המשתמש יכול לראות את מצב הנקודות שלו ושל אחרים, וסטטיסטיקות נוספות.

TextView

סה"כ תשובות נכונות

TextView

סך הניקוד שלך

תמונה שמכילה טקסט, צילום מסך, תוכנה, סמל מחשב

התיאור נוצר באופן אוטומטי

RecyclerView

הצגת הניקוד שלך מול משתמשים אחרים

ProgressBar

הצגת אחוז התשובות הנכונות מבין כל התשובות

TextView

סה"כ תשובות שגויות

## מסך הגדרות

במסך זה אפשר לערוך את הגדרות האפליקציה והגדרות המשתמש.

TextView

שם המשתמש

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטי

ImageButton

שינוי שם המשתמש

ImageButton

מחיקת המשתמש

ProgressBar + ImageButton

שינוי עוצמת מוזיקת הרקע

שם המשתמש

## תרשים מסכים

## 

LoginActivity

## 

## 

הצגת הניקוד

חזרה לתפריט הראשי

משחק חוזר

התחלת משחק

התחברות למשתמש

התנתקות מהמשתמש

MainMenuActivity

GameIdFragment

GameActivity

EndGameFragment

NewGameFragment

JoinGameFragment

SettingsFragment

ScoreFragment

## מחלקות הפרויקט



;;

;;

;;

# מימוש הפרויקט

## מחלקות מסכים (Fragment/Activity)

### LoginActivity

הערה: כל התכונות מוגדרות כprivate אלא אם נכתב אחרת.

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| static final int MIN\_PASSWORD\_LENGTH = 6 | קבוע: מספר הספרות המינימלי בסיסמא |
| static final String LOGIN\_PREFERENCES\_FILE = "loginSp" | שם הקובץ ששומר את פרטי ההתחברות (שם המשתמש והסיסמא) בSharedPreferences |
| static final String LOGIN\_PREFERENCES\_USERNAME = "username" | הkey של שם המשתמש בSharedPreferences |
| static final String LOGIN\_PREFERENCES\_PASSWORD = "password" | הkey של הסיסמא בSharedPreferences |
| Button loginButton | כפתור התחברות/יצירת משתמש |
| TextView toggleLoginModeLbl | כותרת המעבר ממצב התחברות למצב הרשמה |
| TextView toggleLoginModeLink | לייבל לחיץ (Clickable) למעבר ממצב התחברות למצב הרשמה ובחזרה |
| EditText usernameTxt | שורה להכנסת שם המשתמש |
| EditText passwordTxt | שורה להכנסת הסיסמא |
| EditText passwordAgainTxt | שורה להכנסת הסיסמא שוב (במצב הרשמה) כדי לוודא שאין שגיאת הקלדה |
| CheckBox rememberMeCb | קובע האם לזכור את שם המשתמש והסיסמא להתחברות הבאה ולהתחבר באופן אוטומטי |
| TextView loginStatusLbl | מודיע אם ההתחברות/הרשמה הצליחה. אם לא כותב מה השגיאה |
| Mode mode | המצב הנוכחי – התחברות או הרשמה  enum Mode {  LOGIN,  SIGNUP  { |

שמירת המידע בבסיס הנתונים:

המשתמשים נשמרים בפיירסטור בקולקציה (מעין HasMap<>) של אובייקטים מסוג User.

המזהה של כל משתמש הוא שם המשתמש שלו.

פעולות עיקריות

(פעולות של שמירת המידע לsharedPreference יפורטו בחלק "בסיסי נתונים")

התחברות:

private void login(String username, String password) {

**בדיקה אם שם המשתמש והסיסמא תקינים**

**(אורך תקין, אין תווים לא חוקיים וכו')**

if(!validateUsernameAnsPassword(username, password))

return;

**אם הם תקינים - התחברות**

**פנייה לפיירבייס לבקשת התחברות. בסוף ההתחברות נקראת פעולת onComplete**

FirebaseAuth.getInstance().signInWithEmailAndPassword(username, password).addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if(task.isSuccessful()){

**התחברות הצליחה**

**מעבר לתפריט הראשי (מסך יצירת משחק)**

startMainMenuActivity();

//TODO: if user isn't listed in "users" list, then create user and add it

//can happen if user was signed in and had an connection error before he was added to list

}

Else

**התחברות נכשלה**

**צריך להודיע למשתמש מה הייתה השגיאה**

loginStatusLbl.setText(task.getException().getMessage());

}

});

}

הרשמה:

private void signup(String username, String password, String passwordAgain) {

**בדיקה אם 2 הסיסמאות שהוכנסו זהות**

if(!password.equals(passwordAgain)){

**אם הן שונות, מודיעים למשתמש ויוצאים מהפעולה**

loginStatusLbl.setText("Passwords doesn't match");

passwordTxt.setText("");

passwordAgainTxt.setText("");

setWrongColors(passwordTxt);

setWrongColors(passwordAgainTxt);

validateUsernameAnsPassword(username, password);

return;

}

**בודקים אם שם המשתמש והסיסמא תקינים**

**אם לא, יוצאים מהפעולה**

if(!validateUsernameAnsPassword(username, password))

return;

String email = username;

**פנייה לפיירבייס לבקשת יצירת משתמש**

FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

**אחרי ביצוע הבקשה בודקים אם ההרשמה הצליחה**

if(task.isSuccessful()){

//add user to users list

**אם היא הצליחה, מוסיפים את המשתמש לרשימת המשתמשים**

User user = new User(username, FirebaseAuth.getInstance().getUid(), 0, 0, 0);

FirebaseFirestore.getInstance(). collection(GameActivity.USERS\_COLLECTION\_PATH).document(email).set(user).

addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(task.isSuccessful())

**אחרי שמוסיפים את המשתמש לרשימת המשתמשים, עוברים לתפריט הראשי**

startMainMenuActivity();

else

**אם הרשמת המשתמש/ההוספה שלו לרשימת המשתמשים נכשלה, מודיעים למשתמש מה הייתה השגיאה**

loginStatusLbl.setText(task.getException().getMessage());

}

});

}

else

loginStatusLbl.setText(task.getException().getMessage());

}

});

}

### MainMenuActivity

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| GestureDetectorCompat swipeDetector | רכיב שמזהה החלקות על המסך. אם המשתמש החליק ימינה או שמאלה לאורך מרחק מסויים, עוברים מסך ימינה/שמאלה בהתאמה |
| static final int MIN\_SWIPE\_LENGTH = 150 | המרחק המינימלי של החלקה אופקית בשביל לעבור מסך. אם המשתמש החליק על המסך למרחק קטן יותר, לא יקרה כלום |
| BottomNavigationView navigationView | תפריט בתחתית המסך |
| NewGameFragment newGameFragment | ארבעת הפרגמנטים אליהם ניתן לעבור בתפריט  ביצירת המסך, הפרגמנט המוצג הוא  NewGameFragment |
| JoinGameFragment joinGameFragment |
| ScoreFragment scoreFragment |
| SettingsFragment settingFragment |

פעולות

מעבר בין מסכים בהחלקה:

public boolean onFling(@NonNull MotionEvent e1, @NonNull MotionEvent e2, float velocityX, float velocityY) {

**מציאת הפרגמנט שמוצג עכשיו**

Fragment currentFragment = getSupportFragmentManager().findFragmentById(R.id.mainFragmentContainer);

FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();

**בדיקה אם ההחלקה מספיק גדולה בשביל לעבור מסך**

if(Math.abs(e2.getX() - e1.getX()) > MIN\_SWIPE\_LENGTH){

//swipe

**בדיקה אם ההחלקה הייתה ימינה או שמאלה**

if(e2.getX() > e1.getX()){

//right swipe

**בהחלקה ימינה: מעבר מסך אחד שמאלה**

if(currentFragment instanceof JoinGameFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, newGameFragment).commit();

navigationView.setSelectedItemId(R.id.newGameFragment);

}

else if(currentFragment instanceof NewGameFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, scoreFragment).commit();

navigationView.setSelectedItemId(R.id.scoreFragment);

}

else if(currentFragment instanceof SettingsFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, joinGameFragment).commit();

navigationView.setSelectedItemId(R.id.joinGameFragment);

}

}

else{

**בהחלקה שמאלה: מעבר מסך אחד ימינה**

//left swipe

if(currentFragment instanceof ScoreFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, newGameFragment).commit();

navigationView.setSelectedItemId(R.id.newGameFragment);

}

else if(currentFragment instanceof NewGameFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, joinGameFragment).commit();

navigationView.setSelectedItemId(R.id.joinGameFragment);

}

else if(currentFragment instanceof JoinGameFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, settingFragment).commit();

navigationView.setSelectedItemId(R.id.settingsFragment);

}

}

}

**החזרה: הפעולה הסתיימה בהצלחה**

return true;

}

הערה: מבחינה תכנותית, נכון יותר לשים את הFragmentים במערך. בכתיבת הקוד, היו בהתחלה רק 2 מסכים, אח"כ 3 ובסוף 4, ולכן עשיתי בדרך הזו.

### NewGameFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| static final float UNSELECTED\_BUTTON\_ALPHA = (float) 0.4 | אם בוחרים כפתור (למשל: כפתור של רמת קושי קלה), כל הכפתורים האחרים שאיתו (רמת קושי בינונית וקשה) נעשים שקופים קצת. זו רמת השקיפות שלהם. |
| static final int MIN\_QUESTION\_COUNT = 2 | מספר השאלות המינימלי |
| static final int MAX\_QUESTION\_COUNT = 15 | מספר השאלות המקסימלי |
| Button allCategoriesBtn | כפתור בחירת קטגוריה: כל הקטגוריות |
| Button generalKnowledgeCategoryBtn | כפתור בחירת קטגוריה: ידע כללי |
| Button scienceCategoryBtn | כפתור בחירת קטגוריה: מדע |
| Button computerScienceCategoryBtn | כפתור בחירת קטגוריה: מדעי המחשב |
| Category category | הקטגוריה שנבחרה  enum Category{  ALL,  GENERAL\_KNOWLEDGE,  SCIENCE,  COMPUTER\_SCIENCE  { |
| Button easyDifficultyLevelBtn | כפתור בחירת רמת קושי: קל |
| Button mediumDifficultyLevelBtn | כפתור בחירת רמת קושי: בינוני |
| Button hardDifficultyLevelBtn | כפתור בחירת רמת קושי: קשה |
| DifficultyLevel difficultyLevel | רמת הקושי שנבחרה  enum DifficultyLevel{  EASY,  MEDIUM,  HARD  { |
| Button questionCountDecBtn | הקטנת כמות השאלות למשחק ב1 |
| Button questionCountIncBtn | הגדלת כמות השאלות למשחק ב1 |
| TextView questionCountValueLbl | הצגת מספר השאלות שנבחר למשתמש |
| int questionCount | מספר השאלות שנבחר |
| Button playBtn | כפתור התחלת המשחק |

שמירת המידע בבסיס הנתונים:

המשחקים נשמרים בפיירסטור בקולקציה (מעין HasMap<>) של אובייקטים מסוג Game.

המזהה של כל משחק הוא שדה הID שלו.

תרשים זרימה

onCreate()

showCurrentQuestion()

showLoadingFragment()

(רץ ברקע)

יצירת מסך

רענון מסך (למשל אם סובבו את המסך)

showCurrentQuestion()

getQuestionsAsync.execute()

יצירת משחק חדש

הצטרפות למשחק קיים

joinGameAsync.execute()

questionFetcher.getQuestions()

sendGameToFirestore()

פעולות

הפעולה שנקראת כשנלחץ כפתור במסך:

public void onClick(View v) {

**בדיקה איזה כפתור נלחץ**

switch (v.getId()){

**אם זה אחד מכפתורי בחירת רמת הקושי, בוחרים את רמת הקושי הזו.**

**קוראת לפעולה setDifficultyLevel, שבהמשך אפרט מה היא עושה.**

case R.id.easyDifficultyLevelBtn:

setDifficultyLevel(EASY);

break;

case R.id.mediumDifficultyLevelBtn:

setDifficultyLevel(MEDIUM);

break;

case R.id.hardDifficultyLevelBtn:

setDifficultyLevel(HARD);

break;

**אם זה אחד מכפתורי בחירת הקטגוריה: מסמנים את הקטגוריה הזו.**

case R.id.allCategoriesBtn:

setCategory(ALL);

break;

case R.id.generalKnowledgeCategoryBtn:

setCategory(GENERAL\_KNOWLEDGE);

break;

case R.id.scienceCategoryBtn:

setCategory(SCIENCE);

break;

case R.id.computerScienceCategoryBtn:

setCategory(COMPUTER\_SCIENCE);

break;

**אם נלחץ כפתור הוספת/הורדת שאלה, מוסיפים או מורידים שאלה בהתאמה.**

case R.id.questionCountDecBtn:

questionCountDec();

break;

case R.id.questionCountIncBtn:

questionCountInc();

break;

**אם נלחץ כפתור התחלת משחק, עוברים למסך יצירת משחק**

case R.id.playBtn:

**משתמשים בIntent כדי להעביר למסך את רמת הקושי, מספר השאלות והקטגוריה**

Intent intent = new Intent(getActivity(), GameActivity.class);

int extras[] = new int[3];

extras[QUESTIONS\_COUNT\_INDEX] = questionCount;

extras[DIFFICULTY\_LEVEL\_INDEX] = difficultyLevel.ordinal();

extras[CATEGORY\_INDEX] = category.ordinal();

intent.putExtra(GameActivity.NEW\_GAME\_EXTRAS, extras);

intent.putExtra(IS\_NEW\_GAME\_EXTRA, true);

**מתחילים את GameActivity**

startActivity(intent);

getActivity().finish();

}

{

פעולה שקובעת את רמת הקושי לרמת הקושי שנבחרה:

void setDifficultyLevel(DifficultyLevel newDifficultyLevel){

**מאפסים את אחוז השקיפות של כל כפתורי קביעת רמת הקושי**

easyDifficultyLevelBtn.setAlpha(UNSELECTED\_BUTTON\_ALPHA);

mediumDifficultyLevelBtn.setAlpha(UNSELECTED\_BUTTON\_ALPHA);

hardDifficultyLevelBtn.setAlpha(UNSELECTED\_BUTTON\_ALPHA);

**בודקים מה רמת הקושי שנבחרה מסמנים את הכפתור שלה**

**וקובעים את רמת הקושי להיות זו שנבחרה.**

difficultyLevel = newDifficultyLevel;

switch (newDifficultyLevel){

case EASY:

easyDifficultyLevelBtn.setAlpha(1);

break;

case MEDIUM:

mediumDifficultyLevelBtn.setAlpha(1);

break;

case HARD:

hardDifficultyLevelBtn.setAlpha(1);

break;

}

{

}

### JoinGameFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| public static final int GAME\_ID\_LENGTH = 7 | אורך הID של משחק  מוגדר כpublic כי גם ביצירת משחק צריך לדעת את האורך של הID |
| EditText gameIdTxt | שורה בה כותבים את הID |
| Button joinGameBtn | הצטרפות למשחק |

פעולות

הצטרפות למשחק

private void joinGame(){

String id = gameIdTxt.getText().toString();

int intId;

try{

**הפיכת הID לint.**

intId = Integer.parseInt(id);

**אם אי אפשר להפוך לint, הID לא חוקי (ID יכול לכלול רק מספרים).**

if(id.length() != GAME\_ID\_LENGTH)

**אם אי אפשר להפוך לint, הID לא חוקי (ID יכול לכלול רק מספרים).**

throw new Exception();

}

**ID לא חוקי. מודיעים למשתמש שהID לא חוקי.**

catch (Exception e){

Toast.makeText(getContext(), "Invalid ID entered!", Toast.LENGTH\_SHORT).show();

gameIdTxt.setText("");

return;

}

**אם הID חוקי, עוברים למסך המשחק.**

**בintent, מודיעים שהמשתמש הצטרף למשחק (ולא יצר אותו),**

**ומעבירים את הID של המשחק.**

Intent intent = new Intent(getActivity(), GameActivity.class);

intent.putExtra(GameActivity.IS\_NEW\_GAME\_EXTRA, false);

intent.putExtra(GameActivity.GAME\_ID\_EXTRA, intId);

startActivity(intent);

}

{

### ScoreFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| RecyclerView scoreRv | רשימת המשתמשים |
| CircularProgressBar successPercentagePb | סרגל התקדמות מעגלי, שמציג כמה אחוז מהתשובות נכונות וכמה שגויות |
| TextView totalScoreLbl | לייבל שמציג את הנקודות של השחקן |
| TextView totalCorrectLbl | לייבל שמציג את סה"כ התשובות הנכונות |
| TextView totalWrongLbl | לייבל שמציג את סה"כ התשובות השגויות |
| TextView successPercentageLbl | לייבל שמציג את אחוז התשובות הנכונות |
| ArrayList<User> users | רשימת משתמשים |
| User currentUser | המשתמש הנוכחי |

פעולות

**יצירת הפרגמנט וקבלת נתוני המשתמש מFireStore**

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

**קביעת השדות של המחלקה לערך שלהם בקובץ הlayout**

// Inflate the layout for this fragment

View v = inflater.inflate(R.layout.fragment\_score, container, false);

scoreRv = v.findViewById(R.id.scoreRv);

successPercentagePb = v.findViewById(R.id.successPercentagePb);

totalScoreLbl = v.findViewById(R.id.totalScoreLbl);

totalCorrectLbl = v.findViewById(R.id.totalCorrectLbl);

totalWrongLbl = v.findViewById(R.id.totalWrongLbl);

successPercentageLbl = v.findViewById(R.id.successPercentageLbl);

**הצגת פרגמנט טעינה**

Fragment loadingFragment = new LoadingFragment();

getChildFragmentManager().beginTransaction().replace(R.id.scoreLayout, loadingFragment).commit();

**הורדת רשימת המשתמשים מפיירסטור**

//fetch user list

FirebaseFirestore.getInstance().

collection(GameActivity.USERS\_COLLECTION\_PATH).get().addOnFailureListener(new OnFailureListener() {

@Override

**במקרה שהורדת הרשימה נכשלה – הצגת הודעת שגיאה למשתמש**

**וחזרה למסך יצירת משחק**

public void onFailure(@NonNull Exception e) {

Toast.makeText(getContext(), "Connection error!", Toast.LENGTH\_SHORT).show();

//back to main manu

FragmentTransaction ft = getChildFragmentManager().beginTransaction();

ft.replace(R.id.mainFragmentContainer, new NewGameFragment()).commit();

}

}).addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {

@Override

public void onSuccess(QuerySnapshot queryDocumentSnapshots) {

**במקרה שההורדה הצליחה – אתחול רשימת המשתמשים לרשימה שהתקבלה.**

ArrayList<User> fetchedUserList = new ArrayList<>();

for(DocumentSnapshot documentSnapshot : queryDocumentSnapshots.getDocuments())

if(documentSnapshot.exists())

fetchedUserList.add(documentSnapshot.toObject(User.class));

users = fetchedUserList;

**מציאת המשתמש הנוכחי לצורך הצגת הסטטיסטיקות שלו**

//find current user

for(User user : users)

if(user.getUid().equals(FirebaseAuth.getInstance().getUid()))

currentUser = user;

if(currentUser == null)

**אם המשתמש הנוכחי לא קיים (לא צריך לקרות) – מציגים הודעת שגיאה.**

Toast.makeText(getContext(), "Current user not found!", Toast.LENGTH\_SHORT).show();

**הסתרת מסך הטעינה** getChildFragmentManager().beginTransaction().hide(loadingFragment).commit();

**קריאה לפעולה שמציגה את הסטטיסטיקות על המסך (אפרט עליה בהמשך)**

showScore();

}

});

return v;

}

הצגת הסטטיסטיקות על המסך:

private void showScore() {

**הצגת הניקוד, מספר התשובות הנכונות והשגויות של המשתמש הנוכחי**

totalScoreLbl.setText(Integer.toString(currentUser.getScore()));

totalCorrectLbl.setText(Integer.toString(currentUser.getTotalCorrect()));

totalWrongLbl.setText(Integer.toString(currentUser.getTotalWrong()));

**הצגת אחוז התשובות הנכונות**

//show correct percentage progress bar

successPercentagePb.setProgressBarColor(MyColor.CORRECT\_GREEN);

successPercentagePb.setProgressBarWidth(15);

successPercentagePb.setBackgroundProgressBarColor(MyColor.WRONG\_RED);

successPercentagePb.setBackgroundProgressBarWidth(10);

int totalAnswers = currentUser.getTotalCorrect() + currentUser.getTotalWrong();

int progress;

if(totalAnswers > 0)

progress = (100 \* currentUser.getTotalCorrect()) / totalAnswers;

else

//avoid division by zero

progress = 0;

successPercentagePb.setProgressWithAnimation(progress, (long)1000);

successPercentageLbl.setText(Integer.toString(progress) + “%”);

**מיון רשימת המשתמשים לפי הניקוד שלהם**

//show user list

users.sort((o1, o2) -> (int)(o2.getScore() - o1.getScore()));

**הצגת רשימת המשתמשים והניקוד שלהם**

ScoreListAdapter scoreListAdapter = new ScoreListAdapter(getContext(), users);

scoreRv.setAdapter(scoreListAdapter);

scoreRv.setLayoutManager(new LinearLayoutManager(getContext()));

{

### GameIdFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| TextView startGameGameIdLbl | מציג למשתמש את הID של המשחק, כדי שהשחקן השני ידע להתחבר אליו |

פעולות

אתחול המסך:

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

View view = inflater.inflate(R.layout.fragment\_game\_id, container, false);

**מציאת הלייבל שמציגה את הID**

startGameGameIdLbl = view.findViewById(R.id.startGameGameIdLbl);

**מציאת הID של המשחק**

GameViewModel gameViewModel = new ViewModelProvider(getActivity()).get(GameViewModel.class);

int gameId = gameViewModel.getGame().getId();

**הצגת הID למשתמש**

startGameGameIdLbl.setText(Integer.toString(gameId));

return view;

{

### GameActivity

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| public static final String GAMES\_COLLECTION\_PATH = "games" | השם של אוסף המשחקים בפיירסטור |
| public static final String USERS\_COLLECTION\_PATH = "users" | השם של אוסף המשתמשים בפיירסטור |
| public static final String NEW\_GAME\_EXTRAS = "extras" | שם הextra בintent שכולל פרמטרים של המשחק החדש – קטגוריה, רמת קושי ומספר שאלות, כמערך של int. |
| public static final String IS\_NEW\_GAME\_EXTRA = "isNewGame" | שם הextra בintent שאומר למחלקה האם צריך ליצור משחק חדש (כלומר – האם המשתמש הזה יוצר את המשחק או מתחבר למשחק קיים). |
| public static final String GAME\_ID\_EXTRA = "gameId" | שם הextra בintent שכולל את הID של המשחק (בהתחברות למשחק קיים) |
| public static final int QUESTIONS\_COUNT\_INDEX = 0 | בAsyncTask, הפרמטרים מתקבלים במערך שלIntegerים.  אלו האינדקסים של פרמטרים שונים בGetQuestionsAsync.  בהמשך אראה את אופן השימוש בהם. |
| public static final int DIFFICULTY\_LEVEL\_INDEX = 1 |
| public static final int CATEGORY\_INDEX = 2 |
| TextView currentQuestionLbl | לייבל שמציגה את מספר השאלה הנוכחית מתוך כל השאלות |
| TextView questionLbl | לייבל שמציגה את גוף השאלה |
| Button[] answerButtons | 4 כפתורים שעליהם לוחצים כדי לבחור בתשובה |
| ImageButton homeImgBtn | כפתור חזרה למסך הבית |
| Canvas pbCanvas | קנבס שמציג את ההתקדמות של היריב, ואת התשובות הנכונות והשגויות שלו |
| ImageView progressImg | מקום בו נמצא הקנבס הנ"ל |
| Bitmap progressBitmap | ביטמאפ שנצרך בשביל הקנבס (בהמשך אראה למה צריך את שלושתם) |
| ImageButton recordImgBtn | כפתור הקלטת תשובה (עדיין לא ממומש) |
| GameViewModel gameVM | רכיב ששומר נתונים של המשחק |
| Fragment loadingFragment | פרגמנט שמראה את טעינת המסך |
| Fragment gameIdFragment | פרגמנט שמציג את הID של המשחק לפני ההתחלה, כדי שהמשתמש השני יוכל להתחבר |
| FirebaseFirestore firestore | האובייקט המרכזי של פיירסטור שמאפשר חיבור לDB. |

פעולות

הורדת השאלות מהשרת, ביצירת משחק (משתמש בAsyncTask, מחלקה שמאפשרת להריץ פעולות ברקע תוך כדי ריצת הת'רד הראשי):

private class GetQuestionsAsync extends AsyncTask<Integer, Integer, ArrayList<Question>> {

@Override

//params: question count, difficulty level, category

protected ArrayList<Question> doInBackground(Integer... integers) {

**יצירת אובייקט של המחלקה שמורידה את השאלות וקבלת השאלות ממנו.**

**פרמטרים כמו קטגוריה וכו' מתקבלים בפרמטר integers.**

HttpQuestionFetcher questionFetcher = new HttpQuestionFetcher();

return questionFetcher.getQuestions(integers[QUESTIONS\_COUNT\_INDEX], integers[DIFFICULTY\_LEVEL\_INDEX], integers[CATEGORY\_INDEX]);

}

**פעולה שרצה לאחר קבלת השאלות מהשרת**

@Override

protected void onPostExecute(ArrayList<Question> questions) {

**בדיקה אם התקבלו שאלות. אם לא, הצגת הודעת שגיאה וחזרה למסך הראשי.**

if (questions == null || questions.size() == 0) {

Toast.makeText(getBaseContext(), "Couldn't fetch questions!", Toast.LENGTH\_SHORT).show();

backToMainMenu();

} else {

**אם התקבלו שאלות בהצלחה, שומרים אותן בViewModel.**

gameVM.setQuestions(questions);

**שליחת המשחק שנוצר לפיירסטור. פעולה זו גם מציגה את GameIdFragment עד שיתחבר המשתמש.**

sendGameToFirestore();

}

}

{

קבלת הGame מהשרת, במצב התחברות למשחק:

private class JoinGameAsync extends AsyncTask<Integer, Integer, Void>{

@Override

//params: id

**הפעולה מקבלת כפרמטר את הID של המשחק.**

protected Void doInBackground(Integer... integers) {

**שמירת השחקן הנוכחי, כי הורדת המשחק מפיירסטור תמחק את המשתמש הנוכחי מGame, כי הוא עדיין לא נמצא בGame שבDB.**

Player myPlayer = gameVM.getMyPlayer(); //getting the game from firestore overrides player2 to null

**הפיכת הID לסטרינג**

int id = integers[0];

String strId = Integer.toString(id);

**הורדה של המשחק מפיירסטור**

firestore.collection(GAMES\_COLLECTION\_PATH).document(strId).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {

@Override

public void onSuccess(DocumentSnapshot documentSnapshot) {

//game loaded successfully

if(documentSnapshot.exists()){

**אם ההורדה הצליחה, שמירת המשחק בviewModel:**

Game game = documentSnapshot.toObject(Game.class);

gameVM.setGame(game);

gameVM.enableGameSyncWithFirestore(GameActivity.this);

gameVM.setMyPlayer(myPlayer);

hideLoadingFragment(getSupportFragmentManager(), loadingFragment);

**הצגת השאלה הראשונה.**

showCurrentQuestion();

}

else{

**אם המשחק לא קיים, הצגת הודעת שגיאה וחזרה לתפריט הראשי**

Toast.makeText(GameActivity.this, "Wrong game ID!", Toast.LENGTH\_SHORT).show();

backToMainMenu();

}

}

**אם ההורדה נכשלה, הצגת הודעת שגיאה וחזרה לתפריט הראשי**

}).addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

Toast.makeText(GameActivity.this, "Connection error!", Toast.LENGTH\_SHORT).show();

backToMainMenu();

}

});

return null;

}

{

בכל פעם שהמשתמש בוחר תשובה, מציגים אם היא נכונה למשך שניה,

ואחר כך מציגים את השאלה הבאה:

private void sendAnswer(int answerIndex, Button answerButton) {

**בדיקה אם התשובה שנענתה נכונה**

int currentQuestionIndex = gameVM.getMyPlayer().getCurrentQuestionIndex();

oolean isCorrect = gameVM.getQuestions().get(currentQuestionIndex).correctAnswer == answerIndex;

**הוספה לרשימת התשובות הנכונות/שגויות של השחקן את התשובה שהוא ענה.**

ArrayList<Boolean> isCorrectList = gameVM.getMyPlayer().getIsCorrectList();

isCorrectList.add(isCorrect);

gameVM.setMyIsCorrectList(isCorrectList);

if (isCorrect) {

**אם התשובה נכונה, מציגיםם אותה בירוק**

answerButton.setBackgroundColor(MyColor.CORRECT\_GREEN);

} else {

**אם לא, מציגים אותה באדום.**

answerButton.setBackgroundColor(MyColor.WRONG\_RED);

answerButtons[gameVM.getQuestions().get(currentQuestionIndex).correctAnswer].setBackgroundColor(MyColor.CORRECT\_GREEN);

}

**מציירים האם התשובה נכונה/שגויה על הקנבס בתחתית המסך.**

**הפעולה DrawISCorrectOnProgressBar תוסבר בפירוט בהמשך.**

**(כרגע מציג את ההתקדמות של השחקן הנוכחי, בהמשך אשנה כך שיציג את ההתקדמות של היריב).**

drawIsCorrectOnProgressBar(isCorrect, currentQuestionIndex);

**הגדלת אינדקס השאלה הנוכחית ב1.**

gameVM.setMyCurrentQuestionIndex(currentQuestionIndex + 1);

**כיבוי של כפתורי התשובה בזמן הצגת התשובה הנכונה.**

for (Button answerBtn : answerButtons)

answerBtn.setEnabled(false);

recordImgBtn.setEnabled(false);

**לחכות שניה אחת לפני הצגת השאלה הבאה**

new Handler().postDelayed(new Runnable() {

@Override

public void run() {

**אם אין עוד שאלות (נגמר המשחק), מפעילים AsyncTask שיציג את מסך סיום המשחק**

if (currentQuestionIndex == gameVM.getQuestions().size() – 1) {

//game ended

EndGameAsync endGameAsync = new EndGameAsync();

endGameAsync.execute();

**אם יש עוד שאלות, מציגים את השאלה הבאה.**

} else {

showCurrentQuestion();

}

}

}, 1000);

{

ציור על הקנבס שמראה אם התשובה נכונה או שגויה:

private void drawIsCorrectOnProgressBar(boolean isCorrect, int currentQuestion) {

**מציאת מספר השאלות במשחק**

int totalQuestions = gameVM.getQuestions().size();

**יצירת צייר ירוק/אדום בהתאם לתשובה נכונה/שגויה**

Paint paint = new Paint();

if (isCorrect)

paint.setColor(MyColor.CORRECT\_GREEN);

else

paint.setColor(MyColor.WRONG\_RED);

**חישוב המקום בו צריך לצייר (בציר האופקי)**

int start = currentQuestion \* (pbCanvas.getWidth() / totalQuestions);

int end = (currentQuestion + 1) \* (pbCanvas.getWidth() / totalQuestions);

**ציור של מלבן על הקנבס**

pbCanvas.drawRect(start, 0, end, pbCanvas.getHeight(), paint);

**הכנסה של הקנבס החדש לUI.**

progressImg.setImageBitmap(progressBitmap);

{

סיום המשחק:

private void endGame() {

//TODO: delete game from firestore

**הוספה של התשובות הנכונות והשגויות לסטטיסטיקות של השחקן**

//update user score

gameVM.getMyPlayer().setTotalCorrect(gameVM.getMyPlayer().getTotalCorrect() + gameVM.getMyPlayer().getTotalCorrectInGame());

gameVM.getMyPlayer().setTotalWrong(gameVM.getMyPlayer().getTotalWrong() + gameVM.getMyPlayer().getTotalWrongInGame());

**חישוב הנקודות של 2 המשתמשים**

int myPoints = gameVM.getMyPlayer().calculatePoints();

int otherPoints = gameVM.getOtherPlayer().calculatePoints();

if(myPoints > otherPoints)

**הוספת הניקוד של המנצח לסך הנקודות שלו (רק המנצח מקבל נקודות)**

//you won, add points to total score

gameVM.getMyPlayer().setScore(gameVM.getMyPlayer().getScore() + myPoints);

**המרה של הPlayer לUser כדי להכניס אותו לרשימת השחקנים בפיירסטור**

User myPlayerAsUser = gameVM.getMyPlayer();

//send new user data to users list

**הכנסה של המשתמש לרשימה**

firestore.collection(USERS\_COLLECTION\_PATH).document(gameVM.getMyPlayer().getEmail())

.set(myPlayerAsUser).addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(!task.isSuccessful()){

**אם ההכנסה נכשלה, מציגים הודעת שגיאה וחוזרים לתפריט הראשי**

Toast.makeText(GameActivity.this, "Connection error!", Toast.LENGTH\_SHORT).show();

backToMainMenu();

}

else{

**אם ההנסה הצליחה, עוברים לפרגמנט של סיום המשחק שמציג את התוצאה לשחקנים**

//game updated successfully

showEndGameFragment();

}

}

});

}

### EndGameFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| GameViewModel gameVM | הViewModel של המשחק – מאפשר לגשת למידע של המשחק מהפרגמט |
| TextView yourScoreCountLbl | מציג את הניקוד שלך |
| TextView enemyScoreCountLbl | מציג את הניקוד של היריב |
| TextView winnerLbl | מציג את המשתמש שניצח |
| ImageButton endGameReplayBtn | כפתור למשחק חוזר |
| ImageButton endGameHomeBtn | כפתור לחזרה למסך הבית |
| ImageButton endGameShareBtn | כפתור לשיתוף תוצאות המשחק |

פעולות

הצגת התוצאות:

private void showResults(View view) {

**חישוב הנקודות של השחקנים**

int yourScore = gameVM.getMyPlayer().calculatePoints();

int enemyScore = gameVM.getOtherPlayer().calculatePoints();

yourScoreCountLbl.setText(Integer.toString(yourScore));

enemyScoreCountLbl.setText(Integer.toString(enemyScore));

**הצגת המשתמש המנצח**

if(yourScore > enemyScore) {

//you won

winnerLbl.setText("You won!");

**הלייבל שמציג את הניקוד של המנצח יהיה גדול יותר**

yourScoreCountLbl.setTextSize(TypedValue.COMPLEX\_UNIT\_SP, 84);

}

else if(yourScore < enemyScore){

//you lost

winnerLbl.setText(gameVM.getOtherPlayer().getUsername() + " won!");

enemyScoreCountLbl.setTextSize(TypedValue.COMPLEX\_UNIT\_SP, 84);

}

else{

**אם יש תיקו, אין מנצח**

//draw

winnerLbl.setText("Draw!");

yourScoreCountLbl.setTextSize(enemyScoreCountLbl.getTextSize());

}

{

## מחלקות עזר

### Game

מחלקה שכוללת את כל המידע של המשחק

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| ArrayList<Question> questions | רשימת השאלות |
| Player player1, player2 | השחקנים |
| int id | הID של המשחק |

פעולות

אין פעולות מיוחדות, רק get, set.

### Question

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| String question | גוף השאלה |
| ArrayList<String> answers | מערך של התשובות האפשריות |
| int correctAnswer | האינדקס של התשובה הנכונה |
| Category category | קטגוריית השאלה |
| DifficultyLevel difficultyLevel | רמת הקושי |
|  |  |

פעולות

אין פעולות מיוחדות, רק get, set.

### User

במחלקה זו, כל השדות הם מסוג protected כי player יורשת ממנה

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| Protected String username | שם המשתמש |
| Protected String uid | הID של המשתמש (נקרא UID ולא ID כי ככה זה מכונה בפיירבייס) |
| Protected int score | הניקוד |
| Protected int totalCorrect | סך התשובות הנכונות |
| Protected int totalWrong | סך התשובות השגויות |

פעולות

אין פעולות מיוחדות, רק get, set.

### Player

יורשת מUser

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| int currentQuestionIndex | מספר של השאלה הנוכחית במשחק |
| ArrayList<Boolean> isCorrectList | רשימה שמציגה עבור כל תשובה שהמשתמש ענה אם היא נכונה או לא |

פעולות

חישוב הנקודות:

**הנוסחא לחישוב הניקוד קצת מורכבת. היא בנויה כך שלתשובה נכונה יש משקל גדול יותר מלתשובה שגויה. ככל שיש יותר תשובות נכונות, הניקוד גבוה יותר, ולהיפך.**

**מוסיפים 1 לסכום התשובות השגויות כדי שבמקרה ואין תשובה שגויה, לא תהיה חלוקה ב0.**

**יש המרה לint והכפלה ב10 כדי שהנקודות יהיו תמיד כפולה של 10.**

public int calculatePoints() {

return 10 \* (int)(5 \* (Math.pow(getTotalCorrectInGame(), 1.2)) / (getTotalWrongInGame() + 1));

}

בנוסף, מומשה פעולת equals (באופן אוטומטי) שמאפשרת להשוות בין השדות של 2 שחקנים ומחזירה אם הם שווים.

### HttpQuestionFetcher

הורדת השאלות מהאינטרנט

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| static final String DATABASE\_URL\_ADDRESS = "https://opentdb.com/api.php" | כתובת האתר שממנו מורידים את השאלות |
| static final int OK\_RESPONSE\_CODE = 0 | קוד תגובה מהשרת, שמשמעותו שהבקשה בוצעה בהצלחה |
| static final int GENERAL\_KNOWLEDGE\_CATEGORY\_ID = 9 | כדי לקבל שאלה בקטגוריה מסויימת מהשרת, צריך לשלוח כפרמטר את הID של הקטגוריה הרצויה. |
| static final int SCIENCE\_CATEGORY\_ID = 17 |
| static final int COMPUTER\_SCIENCE\_CATEGORY\_ID = 18 |
| String response | התגובה מהשרת |

פעולות

קבלת השאלות מהשרת:

public ArrayList<Question> getQuestions(int questionCount, int difficultyLevel, int category) {

response = "";

**תהליכים שדורשים חיבור אינטרנט באנדרואיד צריכים להיות בת'רד נפרד,**

**לכן יוצרים להם ת'רד**

//run in other thread, because networking isn't allowed in main thread

Thread sendThread = new Thread(new Runnable() {

@Override

public void run() {

try {

**מוסיפים לכתובת הURL את הפרמטרים שמועברים לשרת.**

**URL לדוגמא:**

**https://opentdb.com/api.php?amount=10&category=9&difficulty=medium**

URL url = new URL(DATABASE\_URL\_ADDRESS + getParams(questionCount, difficultyLevel, category));

**יוצרים חיבור HTTP**

HttpURLConnection connection = (HttpURLConnection) url.openConnection();

**מגדירים שהפרמטרים יעברו כחלק מכתובת הURL**

connection.setRequestMethod("GET");

if (connection.getResponseCode() != HttpURLConnection.HTTP\_OK)

**אם ההורדה נכשלה, זורקים שגיאה**

throw new Exception(“HTTP returned response code “ + connection.getResponseCode());

**אם ההורדה הצליחה, שמים את התגובה במשתנה response**

BufferedReader reader = new BufferedReader(new InputStreamReader((connection.getInputStream())));

String line = reader.readLine();

while (line != null) {

response += line;

line = reader.readLine();

}

**סוגרים את החיבור**

reader.close();

} catch (Exception e) {

response = “”;

}

}

});

try {

sendThread.start();

**מחכים עד לסוף ריצת הת'רד כדי להמשיך**

sendThread.join();

} catch (Exception e) {

return null;

}

**אם התגובה ריקה, הייתה שגיאת רשת. לכן לא מחזירים כלום**

**(מי שקורא לפעולת getQuestions מוודא שלא חזר null)**

if (response.equals(""))

//fetching failed

return null;

**הופכים את השאלות שחוזרות בJSON לArrayList<Qeustion>**

**בפעולה נפרדת שתפורט בהמשך**

return deserializeQuestions(DifficultyLevel.values()[difficultyLevel], Category.values()[category]);

{

המרה של השאלות מJSON לרשימת שאלות:

private ArrayList<Question> deserializeQuestions(DifficultyLevel difficultyLevel, Category category) {

**יוצרים רשימת שאלות**

ArrayList<Question> questions = new ArrayList<Question>();

try {

JSONObject jsonObject = new JSONObject(response);

int responseCode = Integer.parseInt(jsonObject.getString("response\_code"));

**בודקים אם התגובה מהשרת היא שגיאב, אם כן מחזירים null**

if (responseCode != OK\_RESPONSE\_CODE)

return null;

JSONArray questionsJsonObject = jsonObject.getJSONArray("results");

**הופכים את הטקסט של כל שאלה לאובייקט JSON של שאלה**

for (int i = 0; i < questionsJsonObject.length(); i++) {

JSONObject questionJsonObject = questionsJsonObject.getJSONObject(i);

**יוצרים שאלה**

Question question = new Question();

**מאתחלים את הערכים של השאלה עם הקטגוריה ורמת הקושי שהמשתמש ביקש**

question.setCategory(category);

question.setDifficultyLevel(difficultyLevel);

**מאתחלים את השדות האחרים של השאלה בערכים שלהם מתוך הJSON**

String questionString = questionJsonObject.getString("question");

question.setQuestion(questionString);

String correctAnswer = questionJsonObject.getString("correct\_answer");

JSONArray incorrectAnswersJsonArray = questionJsonObject.getJSONArray("incorrect\_answers");

ArrayList<String> answers = new ArrayList<>();

for (int j = 0; j < incorrectAnswersJsonArray.length(); j++)

answers.add(incorrectAnswersJsonArray.getString(j));

answers.add(correctAnswer);

**תווים מיוחדים בJSON נשמרים בפורמט שונה. לכן, ממירים אותם בחזרה לפורמט טקסט**

for (int j = 0; j < answers.size(); j++)

answers.set(j, Html.fromHtml(answers.get(j), Html.FROM\_HTML\_MODE\_LEGACY).toString());

question.setQuestion(Html.fromHtml(question.getQuestion(), Html.FROM\_HTML\_MODE\_LEGACY).toString());

**מערבבים את רשימת השאלות**

Collections.shuffle(answers);

question.setAnswers(answers);

**מוצאים את האינדקס של התשובה הנכונה**

//check the correct answer’s index

for(int j=0;j<answers.size();j++)

if(answers.get(j).equals(correctAnswer))

question.setCorrectAnswer(j);

**מוסיפים את השאלה הראשונה**

questions.add(question);

}

**אם הייתה שגיאה, מחזירים null**

} catch (Exception e) {

return null;

}

**אם לא, המחזירים את השאלות**

return questions;

}

}

### GameViewModel

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| MutableLiveData<Game> game | המשחק |
| boolean isCreator | האם המשתמש הזה הוא יוצר המשחק  (אם לא, הוא הצטרף למשחק שאחר יצר) |

### פעולות

הפעלת סנכרון עם המשחק בפיירסטור, כדי שכל שינוי שהמשתמש האחר עשה יעודכן אצלנו, וכל שינוי שנעשה בGame יישלח לשרת:

**הגדרת משתנה שישמור את הערך הקודם של myPlayer כדי לדעת אם הוא השתנה**

private Player previousMyPlayer;

public void enableGameSyncWithFirestore(Context context){

previousMyPlayer = getMyPlayer();

**המרה של הID לסטרינג**

String gameId = Integer.toString(getGame().getId());

//when other player is changed, update Game locally

FirebaseFirestore firestore = FirebaseFirestore.getInstance();

**הוספת האזנה לשינויים במשחק בפיירסטור** firestore.collection(GameActivity.GAMES\_COLLECTION\_PATH).document(gameId).addSnapshotListener(new EventListener<DocumentSnapshot>() {

@Override

public void onEvent(@Nullable DocumentSnapshot snapshot, @Nullable FirebaseFirestoreException error) {

if(error == null && snapshot != null && snapshot.exists()){

//no exception

**כאשר יש שינוי, מקבלים את המשחק אחרי השינוי**

Game newGame = snapshot.toObject(Game.class);

if(newGame.getPlayer2() != null){

**אם השחקן השני ריק, עדיין לא התחיל המשחק, ולכן לא עושים כלום**

//if player2 is null, game hasn't yet started

Game gameValue = getGame();

**אם השחקן הנוכחי הוא היוצר של המשחק, והשחקן שהצטרף (player2) הוא זה שהשתנה,**

**מעדכנים את הערך של שחקן 2 אצלנו.**

if(isCreator && !newGame.getPlayer2().equals(gameValue.getPlayer2())){

//player 2 has changed

gameValue.setPlayer2(newGame.getPlayer2());

game.setValue(gameValue);

}

**אם השחקן הנוכחי הוא השחקן שהצטרף, והשחקן שיצר את המשחק הוא זה שהשתנה,**

**מעדכנים את הערך של שחקן 1**

if(!isCreator && !newGame.getPlayer1().equals(gameValue.getPlayer1())){

//player 1 has changed

gameValue.setPlayer1(newGame.getPlayer1());

game.setValue(gameValue);

}

}

}

}

});

**הוספת האזנה לשינוי בשחקן הנוכחי, וכשהוא משתנה – עדכון של הפיירסטור**

//when my player changed, update in firestore

game.observe((LifecycleOwner) context, new Observer<Game>() {

@Override

public void onChanged(Game newGame) {

**צריך לבדוק אם השחקן שלנו השתנה או שהיריב השתנה. אם היריב הוא זה שהשתנה, לא צריך לעדכן בפיירסטור כי זה כבר מעודכן.**

**אם המשתנה ששומר את המצב הקודם של השחקן ריק, והשחקן לא ריק, השחקן השתנה הנוכחי השתנה**

**גם השחקן הנוכחי שונה מהמצב הקודם שלו, הוא השתנה**

**לכן, צריך לשלוח את game לפיירסטור**

if((previousMyPlayer == null && getMyPlayer() != null) ||

(previousMyPlayer != null && !previousMyPlayer.equals(getMyPlayer())))

//player has changed

firestore.collection(GameActivity.GAMES\_COLLECTION\_PATH).document(gameId).set(newGame).

addOnFailureListener(new OnFailureListener() {

@Override

**אם העדכון נכשל, הייתה שגיאת רשת. לכן, מעדכנים את המשתמש ויוצאים מהמשחק.**

Public void onFailure(@NonNull Exception e) {

Toast.makeText(context, “Connection error!”, Toast.LENGTH\_SHORT).show();

//TODO: exit screen

}

});

**מעדכנים את הערך ה"קודם" של השחק הנוכחי,**

**(בפעם הבאה שיהיה שינוי, השחקן הנוכחי יהיה הקודם).**

if(getMyPlayer() == null)

//copy constructor doesn't work with null

previousMyPlayer = null;

else

previousMyPlayer = new Player(getMyPlayer());

}

});

}

### ScoreListViewHolder

רכיב שממנו מורכבת כל שורה ברשימת המשתמשים במסך הניקוד. כולל את הView שמציגים את המידע על השחקן.

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| TextView place | מציג את המקום של השחקן ברשיממת המשתמשים |
| TextView username | מציג את שם המשתמש |
| TextView score | מציג את הניקוד |

פעולות

יצירת הViewHolder:

public ScoreListViewHolder(@NonNull View itemView) {

super(itemView);

**מוצאים את כל אחד מהViews שבXML, ושמים את הערך שלו בשדה המתאים**

place = itemView.findViewById(R.id.placeLbl);

username = itemView.findViewById(R.id.usernameLbl);

score = itemView.findViewById(R.id.scoreLbl);

}

### ScoreListAdapter

מחלקה שאחראית על ניהול תוכן רשימת המשתמשים במסך הניקוד

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| Context context | הקונטקסט של הActivity.  הכרחי בשביל פעולת onCreateViewHolder |
| ArrayList<User> users | רשימת המשתמשים שהתקבלה מפיירסטור, ובה נמצא הניקוד של כל משתמש |

פעולות

יצירה של ViewHolder:

public ScoreListViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

LayoutInflater layoutInflater = LayoutInflater.from(context);

**יצירה של שורה חדשה ברשימה**

View view = layoutInflater.inflate(R.layout.recycler\_view\_score, parent, false);

**יצירת הViewHolder והחזרה שלו**

return new ScoreListViewHolder(view);

}

**קישור בין ViewHolder לבין המשתמש והכנסת המידע על המשתמש לViewHolder**

public void onBindViewHolder(@NonNull ScoreListViewHolder holder, int position) {

User user = users.get(position);

**בכל View בשורה של המשתמש, מכניסים את הערך (מקום/שם משתמש/ניקוד) של המשתמש שברשימה**

holder.place.setText(Integer.toString(position + 1));

holder.username.setText(user.getUsername());

holder.score.setText(Integer.toString((int) user.getScore()));

{

## קבצי תצורה

### קבצי Layout

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| activity\_game.xml | קובץ העיצוב של מסך המשחק |
| activity\_login.xml | קובץ העיצוב של מסך ההתחברות |
| activity\_main\_menu.xml | קובץ העיצוב של מסך התפריט הראשי (בתוכו נמצאים כfragments מסכי הניקוד, יצירת משחק, התחברות למשחק והגדרות) |
| fragment\_end\_game.xml | קובץ העיצוב של מסך סיום המשחק |
| fragment\_game\_id.xml | קובץ העיצוב של מסך הצגת הID של המשחק |
| fragment\_join\_game.xml | קובץ העיצוב של מסך ההצטרפות למשחק |
| fragment\_loading.xml | קובץ העיצוב של מסך הטעינה  (לא מופיע ברשימת המסכים כי הוא כולל עיצוב בלבד ואין לו פעולות למעט OnCreate) |
| fragment\_new\_game.xml | קובץ העיצוב של מסך יצירת משחק |
| fragment\_score.xml | קובץ העיצוב של מסך הניקוד |
| fragment\_settings.xml | קובץ העיצוב של מסך ההגדרות |
| fragment\_wait\_to\_enemy.xml | קובץ העיצוב של מסך ההמתנה למשתמש בסוף המשחק  (לא מופיע ברשימת המסכים כי הוא כולל עיצוב בלבד ואין לו פעולות למעט OnCreate) |
| recycler\_view\_score.xml | קובץ העיצוב של כל שורה ברשימת המשתמשים במסך הניקוד |

### קבצי Menu

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| main\_menu.xml | קובץ בו מוגדרות האפשרויות לבחירה שבתפריט הראשי |

### קבצי Navigation

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| main\_nav.xml | קובץ בו מוגדרים הfragments שניתן לעבור אליהן בעזרת התפריט הראשי |

## בסיסי נתונים

בפרויקט השתמשתי ב3 בסיסי נתונים שונים. אפרט כאן על צורת השימוש בהם והמטרות של כל אחד.

### SharedPreferences

זהו בסיס נתונים לוקאלי מובנה בAndroid. השתמשתי בו כדי לשמור את שם המשתמש והסיסמא בהתחברות, כך שבהתחברות הבאה הם יופיעו אוטומטית (אם המשתמש מעוניין בכך).

פעולות:

שמירת שם המשתמש והסיסמא:

private void saveLoginDataToSharedPreferences() {

**יצירת אובייקט שבו נשתמש בשביל לגשת לSharedPreferences**

SharedPreferences sharedPreferences = getSharedPreferences(LOGIN\_PREFERENCES\_FILE, MODE\_PRIVATE);

**אם המשתמש סימן "זכור אותי" – שומרים את שם המשתמש והסיסמא:**

if (rememberMeCb.isChecked()) {

//save to shared preferences

**יצירת "עורך"**

SharedPreferences.Editor sharedPreferencesEditor = sharedPreferences.edit();

**הכנסה של שם המשתמש**

sharedPreferencesEditor.putString(LOGIN\_PREFERENCES\_USERNAME, usernameTxt.getText().toString());

**הכנסה של הסיסמא**

sharedPreferencesEditor.putString(LOGIN\_PREFERENCES\_PASSWORD, passwordTxt.getText().toString());

**שמירת השינויים**

//apply changes

sharedPreferencesEditor.apply();

} else if (sharedPreferences.getAll().size() != 0) { //there is already shared preferences file

**אם המשתמש לא סימן "זכור אותי" – מוחקים את שם המשתמש והסיסמא ששמורים (אם שמורים)**

//delete shared preferences

sharedPreferences.edit().clear().apply();

}

}

שליפת שם המשתמש והסיסמא (אם נשמרו):

private void loadLoginDataFromSharedPreferences() {

//try load the data

**יצירת אובייקט שבו נשתמש בשביל לגשת לSharedPreferences**

SharedPreferences sharedPreferences = getSharedPreferences(LOGIN\_PREFERENCES\_FILE, MODE\_PRIVATE);

**קריאת שם המשתמש**

String username = sharedPreferences.getString(LOGIN\_PREFERENCES\_USERNAME, null);

**קריאת הסיסמא**

String password = sharedPreferences.getString(LOGIN\_PREFERENCES\_PASSWORD, null);

**הכנסה של שם המשתמש והסיסמא לתצוגה**

//if shared preferences was found, show saved username and password in EditTexts

if (username != null)

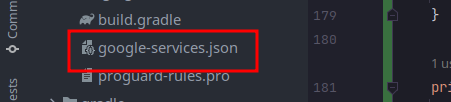
usernameTxt.setText(username);

if (password != null)

passwordTxt.setText(password);

}

### Firestore

בסיס נתונים זה, כמו שכבר כתבתי, הוא בסיס נתונים בענן שניתן כחלק משירות Firebase של גוגל. כדי לשמור בו מידע, חיברתי את האפליקציה לפיירבייס באמצעות הורדה של הקובץ google-services.json והדבקה שלו באפליקציה. 

בנוסף, הצטרכתי להוסיף לקובץ build.gradle של האפליקציה את הdependencies הבאים:

implementation platform('com.google.firebase:firebase-bom:32.0.0')  
implementation 'com.google.firebase:firebase-firestore-ktx'  
implementation 'com.google.firebase:firebase-auth-ktx'  
implementation 'com.google.api:api-common:2.2.1'

כדי לשמור מידע בפיירסטור, צריך להשתמש באובייקט

FirebaseFirestore.getInstance()

המידע נשמר באוספים (מבנה נתונים שמתפקד כמו HashMap) של אובייקט מסוג מסויים (במקרה שלי – User או Game).

כדי לשמור את המידע, משתמשים באובייקט של פיירסטור:

firestore.collection(שם האוסף).document(המפתח של הערך).set(הערך שמכניסים);

כדי לוודא שהשמירה הצליחה, צריך להשתמש במתודה

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(task.isSuccessful()){

**כאן נמצא הקוד שרץ אם השמירה הצליחה**

}

else{

**כאן נמצא הקוד שרץ אם השמירה נכשלה**

}

}

});

שליפת מידע מהפיירסטור מתבצעת כך:

firestore.collection(שם האוסף).document(המפתח של הערך).get();

כדי לקרוא את המידע, צריך להוסיף גם כאן listener. לרוב, העדפתי להשתמש בaddOnSuccessListener:

.addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {

@Override

public void onSuccess(DocumentSnapshot documentSnapshot) {

**קריאת האובייקט שהתקבל**

Object object = documentSnapshot.toObject(Object.class);

});

אלו דרכי שמירת ושליפת המידע המרכזיות בפיירסטור. אפשר לעשות איתו עוד מספר פעולות דומות (כמו: שליפה של אוסף שלם במקום ערך מסויים, הוספת listener שיודיע על שינויים בערך ועוד). על כל אחת מהפעולות השונות שהשתמשתי בהן פירטתי יותר באריכות במקום שהשתמשתי בה. (למשל: דוגמא להאזנה לשינוי של משתנה אפשר למצוא במחלקת GameViewModel.

### Open Trivia DB

בבסיס נתונים זה השתמשתי רק כדי להוריד את השאלות, ולא שמרתי בו מידע בעצמי. לכן, לא אפרט עליו כאן. אופן השימוש המלא בו מפורט לעיל במחלקת HttpQuestionFetcher.

# מדריך למשתמש

## תיאור כללי

המשחק Brain Battle הוא משחק טריוויה אונליין ל2 שחקנים. השחקנים מתחרים זה בזה, עונים על שאלות וצוברים נקודות. המנצח הוא השחקן בעל מספר הנקודות הגדול ביותר.

האפליקציה כוללת את הפיצ'רים הבאים:

* יצירת משתמש ומחיקתו, התנתקות והתחברות
* יצירת משחק
* התחברות למשחק קיים
* צפייה בסטטיסטיקות ובניקוד
* שליטה במוסיקת הרקע
* הקלטת תשובה באמצעות המיקרופון

## הגבלות

שם המשתמש חייב לכלול לפחות תו אחד. הסיסמא צריכה לכלול לפחות 6 תווים.

## מידע נוסף

### הרשאות

חיבור לאינטרנט:

### <uses-permission android:name="android.permission.INTERNET" />

### הצהרות

### גרסת Android מינימלית

Minimum SDK: 24

Target SDK: 32

### גרסאות ומכשירים שעליהם נבדקה האפליקציה

Xiaomi Redmi 7A, Android Version 10.0, API 29 – מכשיר פיזי

FWVGA, Android Version 11.0, API 30 - אמולטור

# רפלקציה

# ביביליוגרפיה

הAPI שלי בסיס הנתונים ממנו שולפיםם את השאלות - <https://opentdb.com/api_config.php>

סטאק אוברפלואו - <https://stackoverflow.com/>

יוטיוב - <https://www.youtube.com/>

התיעוד הרשמי של אנדרואיד - <https://developer.android.com/docs>

התיעוד הרשמי של פיירבייס - <https://firebase.google.com/docs>

# נספחים