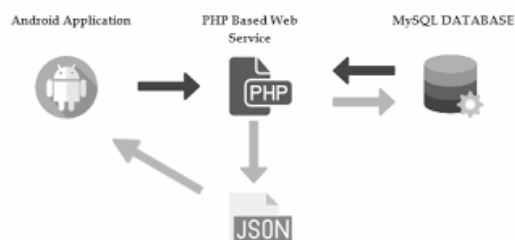


סדנה בתכנות מתקדם ב-JAVA

מסמך תכנון סופי לפרויקט שרת משחק "צוללות"

מבנה המערכת, תיאור המחלקות השונות והקשרים בניהן

מיטל קו'ידוב 318995842
לתמר בניה, אפריל 21



המערכת מורכבת משלושה חלקים שעובדים ביחד כך שצד השרת מתווך בין צד הלקוח לבסיס הנתונים:

מחלקות המערכת הן:

צד הלקוח - אפליקציית אנדרואיד של משחק "צוללות" לשני שחקנים מרוחקים שמציגה ממשק משתמש אינטראקטיבי ומאפשרת למשתמש להירשם למערכת, להתחבר למערכת, להתחיל משחק חדש, לראות את תוצאות עשרת משחקיו האחרונים, לראות את עשר השחקנים המובילים במערכת וכו'.

קוד האפליקצייה נכתב ב-Java והפרונט נכתב ב-xml. סביבת הפיתוח היא **Android Studio**. שולח בקשות HTTP לשרת בעזרת קריאות **REST API**, כלומר retrofit calls של POST, GET לקבצי ה-api של השרת. התשובה מתקבלת בעזרת retrofit client שניגש ל-api של השרת ומתרגם את ה-gson ל-retrofit.

צד השרת - ממומש באמצעות **apache server**. מקשר בין הלקוח לבין בסיס הנתונים באמצעות ה-api שנכתב ב-php (בתוך קבצי ה-php מתבצעות בקשות מבסיס הנתונים שכתובת ב-sql). מחזיר תשובות gson אל הלקוח שקרא לו. צד השרת ובסיס הנתונים מחובר בניהם בעזרת שימוש ב-XAMPP שמחבר בין server apache לבין בסיס נתונים של MySQL ע"י **PhpMyAdmin**.

בסיס הנתונים - בסיס נתונים בשם battleship מסוג MySQL שמכיל מידע על המשתמשים שנרשמו למשחק, מידע על עשרת המשחקים האחרונים של משתמש, מידע על המשתמשים שכרגע ממתינים למשחק ומידע על המשחקים שמתרחשים כרגע. מכיל 4 טבלאות - users, last_games, active_players, active_games.

מבנה המערכת: החלקים והקשרים בניהם יתוארו בעמ' הבאים-

- 2..... צד הלקוח: אפליקציית אנדרואיד.
- 6..... צד השרת: גישה לבסיס הנתונים באמצעות PHP REST API.
- 9..... בסיס הנתונים Battleship.

צד הלקוח: אפליקציית אנדרואיד

אפליקציית Battleship היא אפליקציית אנדרואיד שנכתבה בסביבת הפיתוח Android Studio, עם SDK מינימלי של API 28 (Android 9.0). הקוד של האפליקציה נכתב ב-Java והפרונט נכתב ב-xml.

האפליקציה כוללת את קבצי ה-Java הבאים:

הגדרת הקשר ל-api של השרת:

- **Api:** זהו ממשק שמגדיר באמצעות מתודות את בקשות ה-HTTP שיתבצעו בתוכנית ומגדיר את ה-api שבו הן ישתמשו לצורך מימושן, כלומר קבצי php להם נשלחות הבקשות.
- **ApiClient:** זוהי מחלקה שמממשת את Api. יוצרת לקוח retrofit לתשובות שמתקבלות מה-api של השרת (התשובות מתקבלות ב-gson ומומרות ע"י retrofit) שבמקרה שלנו מוגדר בכתובת- "<http://192.168.14.143/battleship/api/>". כתובת זו מכילה את קבצי ה-php אותם הגדרנו במתודות POST ו-GET בממשק Api.

ה-Activities:

- **MainActivity:** המסך שעולה ראשון בלחיצה על האפליקציה (אם המשתמש לא מחובר משימוש קודם). לאחר שהמשתמש לוחץ על כפתור "log in", הוא מועבר באמצעות intent ל-LoginActivity. המקרה סימטרי עבור לחיצה על "sign in".
- **LoginActivity:** אחרי שהמשתמש לוחץ על כפתור השליחה, נבדוק שתיבות הטקסט username ו-password אינן ריקות וש-username לא קטן מ-4, ואז נקבל ע"י קריאת retrofit ל-login.php בשליחת הפרמטרים username, password אם הפרטים של המשתמש נמצאים במערכת ונכונים. אם כן, המשתמש מחובר- נשמור אותו כמחובר ב-UserPref. נעביר את המשתמש למסך UserMenuActivity.
- **RegisterActivity:** אחרי שהמשתמש לוחץ על כפתור השליחה, נבדוק שתיבות הטקסט username ו-password עומדות בתנאים- שם משתמש חוקי (מינימום 4) וסיסמא חוקית (מינימום 6, אות גדולה אות קטנה מספר וסימן מיוחד). לאחר מכן נקבל ע"י קריאת retrofit ל-register.php בשליחת הפרמטרים username, password אם הפרטים של המשתמש נשמרו במערכת. אם כן, המשתמש מחובר- נשמור אותו כמחובר ב-UserPref. נעביר את המשתמש למסך UserMenuActivity. אם לא- נציג למשתמש את הודעת השגיאה שמוחזרת מהשרת, שהיא או שקיים שם המשתמש הזה כבר במערכת, או שלא הצליח לשמור בגלל תקלה.
- **UserMenuActivity:** מציג למשתמש הודעת ברכה בצירוף שם המשתמש שלו ואת מס' ניצחונותיו. את שם המשתמש שלו הוא מקבל ע"י קריאה למחלקה UserPref. את מס' ניצחונותיו הוא מקבל ע"י retrofit GET call ל-get_wins.php. הוא גם מציג למשתמש את התפריט:

כפתור להתחלת משחק חדש- מבצע קריאת retrofit ל-`search_opponent.php` בה נבדק אם קיים פרטנר למשחק- אם נמצא כזה, נעבור למסך `GameActivity` תוך צירוף הודעה על שמות השחקן היריב ומי מתחיל.
אם לא נמצא כזה- נציג למשתמש הודעת `toast` שמודיעה על חיפוש יריב ונמשיך לחפש באמצעות יריב טיימר וקריאות retrofit. אם המשתמש לחץ על אחד הכפתורים האחרים בתפריט- נפסיק לחפש לו יריב: נבצע קריאת retrofit ל-`delete_active.php` כדי למחוק את השחקן מטבלת הממתינים.
כפתור למעבר ל-10 הגדולים: פותח את `TopTenActivity`
כפתור למעבר ל-10 האחרונים: פותח את `LastTenActivity`
כפתור להתנתקות- לחיצה על `Log out` גוררת `log out` בתוך `UserPref` והמשתמש לא מחובר יותר למערכת. הוא מועבר אוטומטית ל-`LoginActivity`.

- **GameActivity:** מציג לשחקן לוח 5 על 5 ו-4 סירות מתחתיו. לאחר סידור הסירות על הלוח המשתמש לוחץ על הכפתור "ready". אם לא כל הסירות נמצאות במקומות החוקיים: נציג הודעת `toast` למשתמש. אחרת- אם המשתמש הוא השחקן שמתחיל- תורו עכשיו. הלוח הופך להיות לחיץ. בלחיצה על הלוח מתבצעת קריאת retrofit ל-`update_game.php` שמעדכנת בבסיס הנתונים איפה המשתמש ירה ואת `last_played` לשחקן הראשון.
אחרת, זה השחקן השני- מתבצע קריאת retrofit ל-`status_game` שקוראת מבסיס הנתונים ומחזירה את מקום הירייה של השחקן הראשון.
מכאן תורו של השחקן השני- מתבצעת קריאת retrofit ל-`update_game.php` שמעדכנת את `last_hitOrMiss` אם המשתמש הראשון הצליח לפגוע או החטיא, `hit_location`- המיקום שהמשתמש השני ירה ואת `last_played` למשתמש השני.
כך המשחק ממשיך לפי תורות- שחקן יודע כשתורו מגיע באמצעות קריאת שדה ה-`last_played` מהתשובה שמתקבלת מהקריאה ל-`status_game`: כאשר הוא לא האחרון ששיחק- תורו.
המשחק מסתיים כאשר אחד השחקנים מגיע ל-6 פגיעות- המשתמש מגלה שהשחקן השני הגיע ל-6 פגיעות כשמגיע תורו, ולכן קורא ל-`update_game_stat.php` כדי לעדכן את `last_status` ל-"win" כדי שהשחקן השני ידע שניצח. בנוסף, מציג הודעת הפסד לשחקן.
השחקן השני בתורו מגלה בקריאת ה-`retrofit` שניצח ומציג זאת, וקורא ל-`delete_game.php` כדי לשמור ניצחון נוסף לעצמו בבסיס הנתונים וגם כדי למחוק את המשחק הזה ממנו.
לחיצה על `return` במהלך המשחק היא עוד דרך לסיים אותו. כאשר משתמש יוצא באמצע המשחק, כלומר לוחץ על `return` ומועבר ל-`UserMenuActivity`, המשחק נמחק מבסיס הנתונים באמצעות קריאה ל-`delete_game.php`. כאשר השחקן השני מנסה לחפש את המשחק בקריאה ל-`status_game` הוא יקבל תשובה מהשרת של "exit" בשדה `last_status` כדי לעדכן אותו שהשחקן האחר יצא. הוא יועבר באופן אוטומטי ל-`UserMenuActivity` תוך הצגת `toast` שהמשתמש האחר עזב את המשחק.

- **TopTenActivity**: מציג למשתמש את טבלת 10 השחקנים בעלי מס' הניצחונות הגבוה ביותר בDB כך: מבצע קריאת retrofit GET ל-top_ten.php, ומבצע פעולות בתוך המחלקה Player שאליה מתקבלת התשובה מהקריאה כך שהטבלה תוצג כרצוננו.
- **LastTenActivity**: מציג למשתמש את תוצאות עשר המשחקים האחרונים שלו כך: מבצע קריאת retrofit GET ל-last_ten.php, ומבצע פעולות בתוך המחלקה Player שאליה מתקבלת התשובה מהקריאה כך שהטבלה תוצג כרצוננו.

מחלקות POJO אליהן מתקבל ה-response בביצוע api call:

- **Player**: משמש את ה-api calls הבאות:
 delete_active.php, login.php, register.php - מחזירות את השדות
 isSuccess,message
 top_ten.php - מחזירה result,isSuccess,message
 last_ten.php - מחזירה games,isSuccess,message
- **Game**:
 create_game - מחזיר את כל השדות:
 opponent1,opponent2,isSuccess,message
 game_update,update_game_stat,delete_game,save_last
 isSuccess,message
- **GameHandler**: מייצג את התשובה של retrofit GET call ל-game_status.php.
 מכיל את השדות last_status,last_hitOrMiss,last_hit_location,last_played
 last_played: השחקן האחרון ששיחק
 hit_location: מקום הירייה של השחקן האחרון ששיחק
 last_hitOrMiss: אם השחקן שאינו last_played הצליח לפגוע - "hit". אחרת -
 "miss".
- **last_status**: ריק במהלך המשחק. מעודכן אשר השחקן last_played הפסיד - אז
 מכניס לשם "win" כדי שהאחר ידע שניצח.

מחלקות להגדרת אובייקטים לצורך GameActivity:

- **BigShip**: מגדירה סירה גדולה- תופסת שני ריבועים בלוח. המחלקה שומרת את שני מיקומי הסירה.
- **SmallShip**: מגדירה סירה קטנה- תופסת ריבוע אחד. המחלקה שומרת את מיקום הסירה.
- **PlayerShips**: שומרת שתי סירות גדולות ושתי סירות קטנות לפי איך שהמשתמש מיקם אותן.
- **SetPlayerShips**: בודקת אם מיקום המשתמש לסירה ספציפית הוא נכון וחוקי- אם כן, שומרת את הספינה ב-instance של PlayerShips, יחד עם יתר הספינות החוקיות.

מחלקה לצורך שמירת מצב המשתמש (מחובר או לא)

- **SharedPref:** שומרת את שם המשתמש של המשתמש, ובכך מאפשרת למשתמש שכבר התחבר להיכנס אוטומטית ל-UserMenuActivity ללא צורך להתחבר שוב למערכת. בנוסף, ה-Activities משתמשים במחלקה זו כדי לקבל את שם המשתמש.

קבצי ה-xml

לכל Activity יש קובץ xml לצורך עיצוב ממשק המשתמש:










- **Activity_main:** MainActivity מקושר אליו. מגדיר הודעת כניסה ושני כפתורים של login ו-register שבלחיצה עליהם מופעלות מתודות המוגדרות ב-MainActivity.
 - **Activity_login:** LoginActivity מקושר אליו. מגדיר שתי תיבות טקסט - אחת לשם משתמש ואחת לסיסמא. מוגדרים שני כפתורים שבלחיצה עליהם מופעלות מתודות המוגדרות ב-LoginActivity.
 - **Activity_register:** סימטרי ל-activity_login עבור RegisterActivity.
 - **Activity_user_menu:** מגדיר שדה טקסט ש-UserMenuActivity מציג בו הודעה למשתמש ו-4 כפתורים.
 - **Activity_game:** GameActivity מקושר אליו. מכיל שדה טקסט, שני כפתורים, טבלה (TableLayout) ריקה ש-GameActivity מבצע בה שינויים כדי להציג את לוח המשחק, ו-ImageView של "Game Over" שמוצגת בסיום המשחק.
 - **Activity_last_ten:** LastTenActivity מקושר אליו. מכיל שדה טקסט עם הודעה, ושדה טקסט נוסף ש-LastTenActivity ימלא אותו. בנוסף מכיל כפתור לחזרה.
 - **Activity_top_ten:** סימטרי ל-Activity_last_ten עבור TopTenActivity.
- (בנוסף, האפליקציה כוללת שני קבצי xml של אנימציות בתוך התיקייה anim: fade_in, fade_out).

צד השרת: גישה לבסיס הנתונים באמצעות PHP REST API

כתובת הגישה - "<http://192.168.14.143/battleship/api/>"

ה-Rest Api שלנו מוגדר ע"י קבצי ה-php ששמורים בשרת בכתובת הגישה ומבצעים sql queries לטבלאות שבבסיס הנתונים Battleship. אלו הם קבצי ה-php:

Index of /battleship/api

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 db.php	2021-04-21 22:17	261	
 delete_active.php	2021-04-21 22:19	629	
 delete_game.php	2021-04-21 22:28	1.2K	
 get_wins.php	2021-04-21 22:29	511	
 last_ten.php	2021-04-21 22:30	667	
 login.php	2021-04-21 22:31	762	
 register.php	2021-04-21 22:34	1.6K	
 save_last_games.php	2021-04-22 00:54	1.3K	
 search_opponent.php	2021-04-21 22:40	2.3K	
 status_game.php	2021-04-21 22:43	1.1K	
 top_ten.php	2021-04-21 22:44	583	
 update_game.php	2021-04-21 22:52	1.1K	
 update_game_stat.php	2021-04-21 22:57	631	

Apache/2.4.46 (Win64) OpenSSL/1.1.1j PHP/8.0.3 Server at 192.168.14.143 Port 80

- **db.php:** התחברות לבסיס הנתונים Battleship לפי שם המשתמש והסיסמא של האדמין והפורט. אם נכשל, מחזיר את השגיאה.
- **register.php:** מחפש בטבלה users בעמודה username את הפרמטר שהתקבל "username". אם מצא, מחזיר שהמשתמש קיים. אחרת, מוסיף שורה בטבלה users עם הפרמטרים "username" ו"password" שסופקו ע"י המשתמש בעמודות המתאימות.
בנוסף, גם מוסיף שורה בטבלת "last_games" עם הפרמטר "username" בעמודה המתאימה. מחזיר אם הצליח או לא.

- **login.php**: מחפש שורה ב-"users" שהעמודות "username" ו-"password" שלה זהות לפרמטרים שמסופקים ע"י המשתמש. מחזיר למשתמש האם נמצא או לא.
- **Get_wins.php**: מחפש שורה ב-"users" שהעמודה "username" שלה זהה לפרמטר "username" שמסופק ע"י המשתמש. מחזיר למשתמש את העמודה wins מהשורה.
- **Search_opponent.php**: מחפש שורה כלשהי ב-active_players ש-username שלה אינו ה-"username" שהתקבל כפרמטר מהמשתמש. אם אין כזו, מחפש שורה בה ה-username כן "username" בשביל לעדכן את המשתמש שעדיין לא נמצא שחקן נוסף. אם אין כזו, מוסיף את "username" שסופק לו לטבלה. אם נמצאה שורה בה ה-username אינו "username": מחזיר את העמודה username מהשורה, מוחק את השורה וגם יוצר שורה חדשה בטבלה active_games עם העמודות op1 (ה-"username" בשורה שמצא), op2 (ה-"username" שסופק ע"י המשתמש) ו-last_played (מאתחל כ-"not yet"). שאר עמודות השורה נשארות ריקות.
- **Delete_active.php**: מוחק את השורה ב-active_player שבה username הוא "username" (פרמטר שסופק ע"י המשתמש). משמש ע"י השחקן השני שמצטרף למשחק. הוא קורא לקובץ זה כדי למחוק את השחקן הראשון שחיכה לו מהטבלה.
- **Update_game.php**: מעדכן את השורה בה last_played שווה ל-"otherOp" (פרמטר שסופק ע"י המשתמש) בטבלה active_games עם שאר הפרמטרים שסופקו מהמשתמש. (למעט בירייה הראשונה במשחק, כלומר כאשר הפרמטר hitOrMiss המסופק ע"י המשתמש הוא "", אז מתבצע עדכון לשורה בה last_played הוא "not yet" ו-op1 הוא "username").
- **Update_game_stat.php**: מעדכן את העמודה last_status בשורה בה last_played שווה ל-"otherOp" (פרמטר שסופק ע"י המשתמש) בטבלה active_games ל-"win". (מסמן לשחקן השני שהוא ניצח)
- **Status_game.php**: שולף מהטבלה active_games את השורה בה op1 או op2 הוא "username" בהתאם לפרמטר "whichOp" (שניהם מסופקים ע"י המשתמש). אם לא נמצאה שורה כזו- המשחק לא קיים במערכת (נמחק). נחזיר למשתמש "exit" ב-last_status. אחרת, נחזיר את המידע בשורה למשתמש.
- **Save_last_game.php**: מעדכן את השורה שבה העמודה username שווה ל-"username" (פרמטר שסופק ע"י המשתמש) בטבלה last_games עם תוצאת המשחק החדש. עושה זאת ע"י קריאת השורה, הזזת המשחקים "אחד למעלה" (המשחק התשיעי הופך לעשירי... המשחק הראשון הופך לשני) והוספת ה-"result" שהתקבל כפרמטר מהמשתמש לעמודה "game1".

- **Delete_game.php:** מוחק את השורה בה op1 או op2 שווים ל-"username" ב-active_players (בהתאם לפרמטר whichOp שסופק ע"י המשתמש). בנוסף, אם ה-"status" שסופק ע"י המשתמש הוא "win", קורא מהטבלה "users" בשורה בה העמודה username שווה ל-"username" את העמודה "wins", מוסיף לה 1 ושומר אותה מחדש בטבלה. אם המשחק לא נמצא, סימן שהוא כבר נמחק מהטבלה ע"י המשתמש השני.
- **Last_ten.php:** מחזיר למשתמש את השורה ב-last_games בה username שווה ל-"username" שסופק ע"י המשתמש.
- **Top_ten.php:** מחזיר למשתמש את 10 השורות בטבלת users בהן עמודת win מכילה את המס' הגבוהים ביותר ביחס לשאר המשתמשים.

בסיס הנתונים battleship

ניגשים אליו דרך Admin->MySQL->XAMPP (PhpMyAdmin) או בכתובת
<http://192.168.14.143/phpmyadmin/index.php>
(הגישה ניתנת רק למחשב האדמין וחסומה ממכשירים אחרים, שיכולים לגשת לבסיס הנתונים רק ע"י קריאה ל-api של השרת)

טבלאות בסיס הנתונים:



users: טבלת פרטי משתמשי האפליקציה
last_games: טבלת תוצאות עשרת המשחקים האחרונים של users
active_players: טבלת המשתמשים המחוברים ומכילים אקטיביות לפרטנר למשחק
active_games: טבלת המשחקים שטרם נגמרו

תיאור הטבלאות והקשר בניהן:

❖ **users:** טבלת המשתמשים במערכת. מכילה את כל המשתמשים שנרשמו באפליקציה. העמודות הן- שם המשתמש, סיסמא (אחרי הצפנה) ומס' ניצחונות.

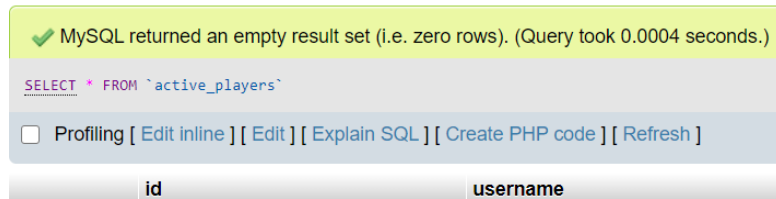
id	username	password	wins
1	jkjk	83c90566f9c998d6860e0861412400fa	0
2	mmmm	83c90566f9c998d6860e0861412400fa	0
3	tttt	83c90566f9c998d6860e0861412400fa	0
4	papa	83c90566f9c998d6860e0861412400fa	0
5	baba	83c90566f9c998d6860e0861412400fa	0
6	mmmj	a94d717c0b94ea8fa63793e0b8e89f7f	0
7	babaj	83c90566f9c998d6860e0861412400fa	0
8	jjjj	f3909461bb995ae5baaafb53c97060f3	0
9	ttsss	83c90566f9c998d6860e0861412400fa	0
10	a1111	83c90566f9c998d6860e0861412400fa	0
11	wtfff	83c90566f9c998d6860e0861412400fa	0
12	ytyty	83c90566f9c998d6860e0861412400fa	0
13	abc1	83c90566f9c998d6860e0861412400fa	23
14	acds1	83c90566f9c998d6860e0861412400fa	0
15	rrrr	83c90566f9c998d6860e0861412400fa	0
16	bbb1	83c90566f9c998d6860e0861412400fa	22
17	tytyt	83c90566f9c998d6860e0861412400fa	0
18	mmmmmy	000b085b1a3bc6aea117a29de08f6524	0
19	romba	052f1179bf449cdab77c516239cd477d	1

❖ last_games: טבלת תוצאות עשרת המשחקים האחרונים של משתמשי המערכת. 0 מסמל שעדיין לא שוחק המשחק, 1 מסמל ניצחון ו-2 הפסד:

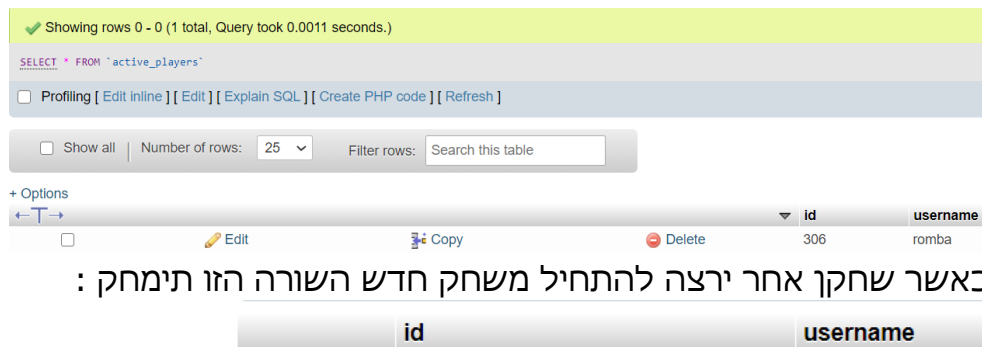
id	username	game1	game2	game3	game4	game5	game6	game7	game8	game9	game10
1	jkjk	0	0	0	0	0	0	0	0	0	0
2	mmmm	0	0	0	0	0	0	0	0	0	0
3	tttt	0	0	0	0	0	0	0	0	0	0
4	papa	0	0	0	0	0	0	0	0	0	0
5	baba	0	0	0	0	0	0	0	0	0	0
6	mmmj	0	0	0	0	0	0	0	0	0	0
7	babaj	0	0	0	0	0	0	0	0	0	0
8	jjjj	0	0	0	0	0	0	0	0	0	0
9	ttss	0	0	0	0	0	0	0	0	0	0
10	a1111	0	0	0	0	0	0	0	0	0	0
11	wttf	0	0	0	0	0	0	0	0	0	0
12	ytyty	0	0	0	0	0	0	0	0	0	0
13	abc1	2	1	1	1	2	1	1	1	1	1
14	acds1	0	0	0	0	0	0	0	0	0	0
15	rrrr	0	0	0	0	0	0	0	0	0	0
16	bbb1	1	2	1	2	2	2	1	2	2	2
17	tytyt	0	0	0	0	0	0	0	0	0	0
18	mmmmj	0	0	0	0	0	0	0	0	0	0
19	romba	2	1	0	0	0	0	0	0	0	0

השרת דואג שכאשר מתווספת שורה לטבלת `users` עבור `username`, תיווסף שורה ל-`last_games` עבור אותו `username`.

❖ active_players: טבלת השחקנים שמחכים לפרטנר למשחק, כלומר שלחצו באפליקציה על "new game" וטרם נמצא להם פרטנר. כאשר שחקן רוצה להתחיל משחק חדש, הוא קודם כל מחפש שורה בטבלה זו- אם מוצא, מוחק את השורה הזו והמשחק מתחיל. אם הטבלה ריקה- מוסיף עצמו לטבלה ומחכה.
- כאשר אין שחקנים שמחכים הטבלה ריקה:



כאשר שחקן רוצה להתחיל משחק חדש והטבלה ריקה תיתווסף שורה בטבלה, לדוגמא:



כאשר שחקן אחר ירצה להתחיל משחק חדש השורה הזו תימחק :

id	username
----	----------

ותיתווסף שורה חדשה ב-`active_games`.

❖ Active_games: כאשר מתחיל משחק חדש, כלומר כאשר כפי שמוסבר למעלה השחקן השני שרצה לשחק קורא למחיקת השחקן הראשון מהטבלה active_players, מתווספת שורה ב active_games לשמירת מצב המשחק שנוצר: בדוגמא שלנו-

id	op1	op2	last_played	last_hitOrMiss	hit_location	last_status
276	romba	bbb1	not yet		0	

הסבר עמודות הטבלה active_games:

Op1 הוא השחקן הראשון- כלומר השחקן שרצה לשחק קודם, השחקן שהופיע בטבלת active_games.
op2 הוא השחקן השני.
השחקנים מוצאים את שורת המשחק שלהם בטבלה ע"י שדות אלו והם לא משתנים במהלך המשחק.

last_played- השחקן ששיחק אחרון. בהתחלה מוגדר כ"not yet" כדי לסמן שטרם מישהו שיחק, והשחקן הראשון מתחיל.
במהלך המשחק- תורו של שחקן הוא כאשר last_played הוא השחקן השני. כאשר תורי במשחק – אני קוראת מהשורה של המשחק בטבלה הזו ולאחר מכן כשאני לוחצת על לוח המשחק (יורה) אני מכניסה את המידע החדש לטבלה, כולל עדכון last_played לשם המשתמש שלי.

last_hitOrMiss -hit מסמן פגיעה ו-miss החטאה.
בקריאה מהטבלה: אני קוראת אם הירייה הקודמת שיריתי הייתה פגיעה או החטאה. לאחר מכן, אני מעדכנת בטבלה אם הירייה של השחקן השני היא hit או miss.

Hit_location- מסמן מה מקום הירייה של השחקן last_played. כל שחקן בתורו מעדכן את hit_location אם מיקום הירייה שלו ואת last_hitOrMiss לפי הירייה שקרא לפני זה מהטבלה (אם היא hit או miss).

Last_status- ריק במהלך המשחק. משתנה רק כאשר אחד השחקנים הפסיד, ולכן הוא מעדכן בטבלה את הסטטוס כ"win" כדי שהשחקן השני ייודע לניצחונו.

דוגמא לטבלה active games במהלך משחק:

בתחילת המשחק-

id	op1	op2	last_played	last_hitOrMiss	hit_location	last_status
287	romba	bbb1	not yet		0	

אחרי ש-romba (השחקן שמתחיל-op1) משחק לראשונה:
romba ירה בריבוע 22 ולכן hit_location מתעדכן ל-22 כמו גם ה-last_played ל-romba.

id	op1	op2	last_played	last_hitOrMiss	hit_location	last_status
287	romba	bbb1	romba		22	

אחרי ש-bbb1 (op2) משחק לראשונה אחרי התור של romba:
bbb1 מעדכן שהירייה של romba היא hit כלומר מעדכן hit בעמודה last_hitOrMiss בנוסף מעדכן את מיקום הירייה שלו- הוא ירה בריבוע 13 לכן hit_location מתעדכן ל-13, וה-last_played ל-bbb1.

id	op1	op2	last_played	last_hitOrMiss	hit_location	last_status
287	romba	bbb1	bbb1	hit	13	

תורו של romba שוב, והפעם הוא מחטיא:

id	op1	op2	last_played	last_hitOrMiss	hit_location	last_status
287	romba	bbb1	romba	miss	3	

מילוי הטבלה ממשיך בצורה הנוכחית עד שהמשחק נגמר:

- שחקן כלשהו מגלה שהשחקן השני ניצח ולכן מעדכן את last_status ל-"win":
לאחר מס' שלבים במשחק, בהנחה שאף שחקן לא יצא באמצע, אחד השחקנים יגיע ל-6 פגיעות: כאשר שחקן א' מגלה ששחקן ב' הגיע ל-6 פגיעות במקום לעדכן את השדות הרגילים hit_location, last_hitOrMiss, last_played הוא מעדכן את השדה last_status בלבד ל-"win" כדי ששחקן ב' ידע שניצח. לאחר מכן- שחקן ב' שקורה את השורה הזו בטבלה מוחק אותה כי המשחק נגמר, תוך כדי הוספת 1 לעמודת wins בטבלת users בשורה של שחקן ב'. במשחק id=287 בין bbb1 ל-romba שהודגם כאן הסתיים בניצחון ל-romba, ולכן השורה של romba בטבלת users אחרי העדכון:

19	romba	052f1179bf449cdab77c516239cd477d	2
----	-------	----------------------------------	---

- בכל שלב במשחק המשתמש יכול לצאת ממנו ואז השורה של המשחק תימחק מהטבלה.

הטבלה ריקה כאשר כל המשחקים הפעילים נגמרו:

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)

SELECT * FROM `active_games`

☐ Profiling [Edit Inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

id	op1	op2	last_played	last_hitOrMiss	hit_location	last_status
----	-----	-----	-------------	----------------	--------------	-------------