תיכון ע"ש ב.אוסטרובסקי

מצוינות • מנהיגות • יזמות • חדשנות

סמל ביה"ס: 440024

עבודת גמר 5 יח"ל

תכנון ותכנות מערכות

סייבר – תכנות אפליקציות אסינכרוניות

פרוייקט בנושא:

משחק טאקי

שם התלמיד : סמואל ארביב

תעודת זהות [redacted]

כיתה : י"ב 2

מורה מלווה : דורית בן-דוד

מאי, 2019, אייר, תשע"ט

תוכן עניינים

מבוא 3
מטרת האפליקציה 3
תיאור המערכת 3
קהל יעד 4
מבנה מסד הנתונים 5
קשרי הגומלין במסד הנתונים 5
טבלאות מתוך מסד הנתונים 6
צד שרת 8
מבנה התיקיות 8
פרוייקט ה- BL (Business Logic) 9
מבנה הפרוייקט 9
מחלקת BL 9
פרוייקט Host 11
מבנה הפרוייקט 11
מחלקה לדוגמה 11
מחלקת MainWindow.xaml 11
הגדרת הרשת של השרת 12
פרוייקט Model 13
מבנה הפרוייקט 13
UML 14
מחלקות דוגמה 15
מחלקת BaseEntity 15
מחלקת User 15
מחלקת PlayersList 15
מחלקת Message 16
פרוייקט ViewModel 17
מבנה התיקיות 17
מחלקות דוגמה 17
מחלקת BaseDB 17
מחלקת ChangeEnitity 21
מחלקת GameDB 21
מחלקת PlayerCardDB 25

שירותי הרשת (פרוייקט ה-WcfServiceLibrary) 29
מבנה התיקיות 29
ממשק שירות הרשת IService 29
מחלקת Service 31
הלקוח - סביבת WPF 34
תרשים זרימת חלונות 34
הגדרת השרת – נמצא בקובץ (App.config) 35
מבנה התיקייה 35
קוד לדוגמה 36
דף Login 36
דף UserPage 40
דף LoadingPage 44
דף GamePage 48
קוד לדוגמה 64
דוגמה לקוד דיאלוג : 67
הלקוח - סביבת Web 69
תרשים זרימת החלונות 69
הגדרת השרת באתר – בקובץ (Web.config) 69
מבנה התיקייה: 70
קוד לדוגמה: 70
דף Register: 70
דף Results 73
הלקוח – סביבת Android(Xamarin) 76
תרשים זרימת הדפים: 76
מבנה התיקיה 77
דוגמה לקוד 77
דף Login 77
Read Me 80
לקוח WCF / Android 80
לקוח Web 81
ביבליוגרפיה 82

מבוא

מטרת האפליקציה

מטרת האפליקציה היא משחק קלפים המבוסס על המשחק הישראלי טאקי, כמשחק רב-משתמשים הקיים על מספר פלטפורמות, תוך שימוש בטכנולוגית שרת-לקוח.
המשחק נבנה כך שהשימוש בו יהיה כמה שיותר פשוט. האפליקציה היא אינטואיטיבית, קלה לשימוש, פשוטה ומהירה.
קיימת מערכת שלמה של משתמשים, להם חברים, היסטוריית משחקים, נקודות, רמות ועוד, ומגובה ומנוהלת כולה במסד נתונים, שבו זמנית גם מכיל את כל מידע אודות כל משחק שאי פעם התקיים, בכל רגע נתון.

תיאור המערכת

המערכת מבוססת על מבנה של שרת/לקוח.

השרת מכיל את מערכת המשחק – הגדרות כלל המחלקות הבונות את המערכת כולה, הקישורים השונים הקיימים ביניהן, את הקישור שלהן למסד הנתונים, את החוזה בין השרת ללקוח המכיל את כלל הפונקציונאליות המרכיבה את ממשק המשתמש, היחסים בין השחקנים השונים, ואת מהלך המשחק עצמו.

בנוסף לכך, השרת מכיל את שכבת הלוגיקה (Business Logic) המכיל פונקציות פנימיות מרובות, אשר אחראי על הלוגיקה המרכיבה את האפליקציה כולה, מלבד לוגיקת המשחק עצמה אשר קיימת בצד הלקוח מטעמי יעילות.

המערכת יכולה להתקיים בידי כל משתמש רשום, כאשר קיים משתמש בעל סמכות של מנהל, ולו קיימות פונקציות מיוחדות.

המערכת קיימת בסביבה חלונאית במערכת ווינדוס בסביבת WPF, וכמו כן קיימת גם בסביבות טלפונים ניידים iOS ו- Android בשימוש בטכנולוגיית Xamarin.

המערכת פועלת באופן דומה בכל הסביבות השונות, כאשר כולן מחוברות לשרת בפורטל WCF, אשר מכיל את כל הפונקציות השונות, והוא היוצר את רמת הרב-משתמשים של המשחק.

השרת, שהוא ליבת המערכת, אחראי על יצירת עצמים מתאימים ושליחתם ללקוחות בהתאם לבקשותיהם.

בנוסף למערכות המשחק הקיימות במספר פלטפורמות, קיים אתר אינטרנט המקושר גם הוא לשרת, אך אינו מכיל משחק. האתר מיועד לניהול משתמשים בלבד, ומיועד בעיקר, אך לא מוגבל, למשתמשים בעלי סמכות מנהל (Admin).

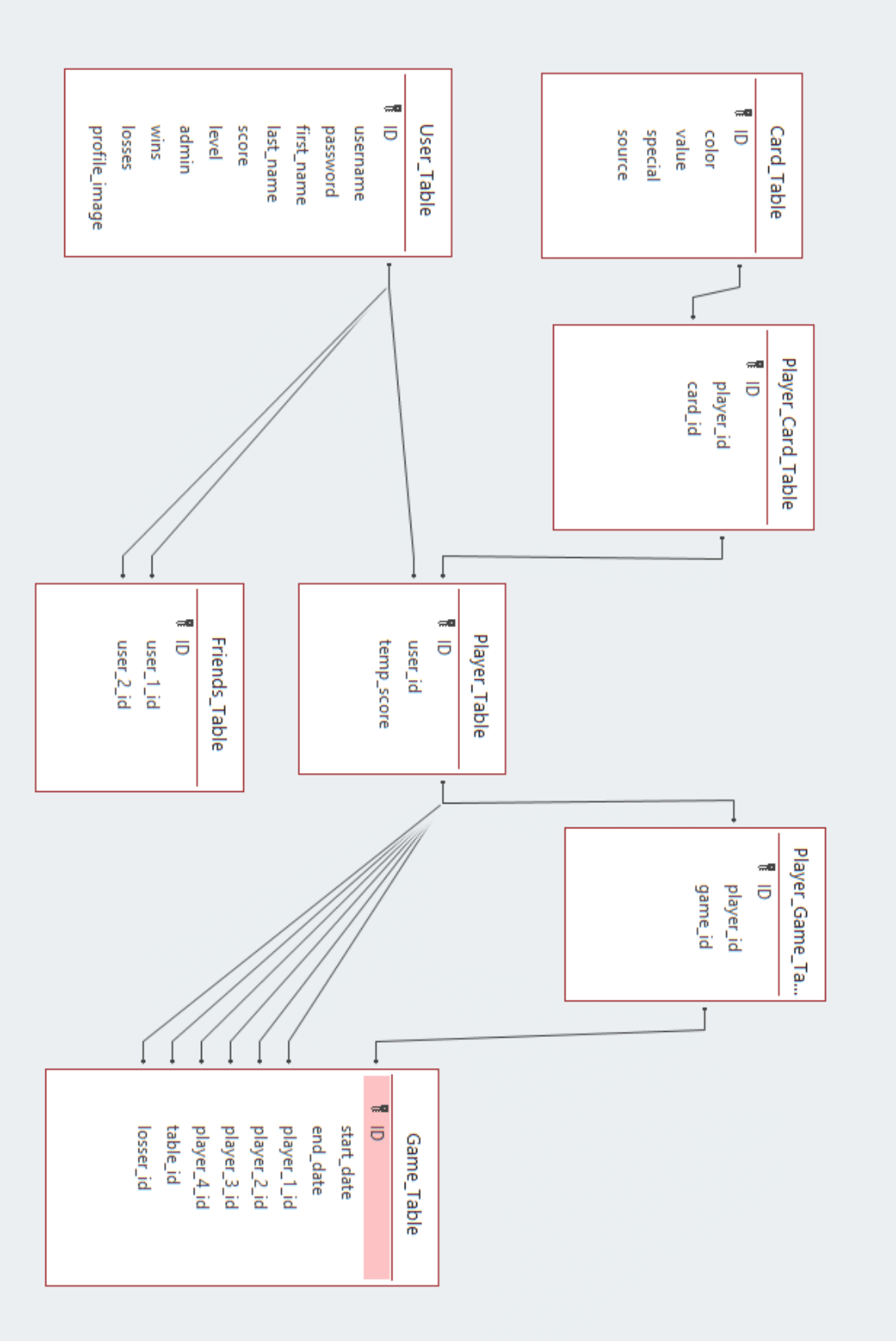
במערכת זו, יכולים שחקנים המשחקים מפלטפורמות שונות לשחק זה עם זה במשחק טאקי, המנוהל באופן מסונכרן בין כל השחקנים באמצעות השרת.
בנוסף למשחק, יכול כל שחקן להוסיף חברים, לראות את נתוני המשחק שלו (נקודות שצבר, רמה), נתוני המשחק של חבריו, היסטוריית המשחקים שלו ואת היסטוריית המשחקים המשותפת שלו עם כל חבר.

קהל יעד

המשחק אינו מיועד לטווח גילאים מסוים, אולם הוא נבנה בצורה אשר מתאימה גם לילדים וגם למבוגרים בכך שהוא פשוט וקל לשימוש, אינטואיטיבי, ומעוצב בצורה מעניינת.
נוסף לכך, חוקי המשחק עצמו פשוטים דיים בכדי להתאים לילדים, אך מצריכים רמה מסוימת של חשיבה וטקטיקה אשר עשויה לבדר גם את הקהל המבוגר יותר.

מבנה מסד הנתונים

קשרי הגומלין במסד הנתונים



טבלאות מתוך מסד הנתונים

1.טבלת User_Table

Field Name	Data Type
ID	AutoNumber
username	Short Text
password	Short Text
first_name	Short Text
last_name	Short Text
score	Number
level	Number
admin	Yes/No
wins	Number
losses	Number
profile_image	Short Text

טבלה לאכסון מידע אודות כל המשתמשים הרשומים במערכת. מכילה את פרטיהם השונים:
מספר מזהה, שם משתמש, סיסמא, שם מלא, ניקוד, רמה, האם הם מסוג מנהל, מספר נצחונות וההפסדים, ותמונת פרופיל.

2.טבלת Player_Table

Field Name	Data Type
ID	AutoNumber
user_id	Number
temp_score	Number

טבלה לאכסון מידע אודות השחקנים במשחק. בכל פעם שמשתמש נכנס למשחק, נוצר בדמותו עצם מסוג שחקן, שנועד לייצוג פשוט יותר של המשתמש בפרק זמן המשחק שלו.
הטבלה מכילה פרטים הנוספים לעצם מסוג User ביצירת עצם Player כמו: מספר מזהה מיוחד לכל session, שמירת המספר המזהה המקורי והקבוע של המשתמש, והניקוד הזמני של השחקן בתהליך המשחק הנוכחי בו הוא נמצא.

3.טבלת Game_Table

Field Name	Data Type
ID	AutoNumber
start_date	Date/Time
end_date	Date/Time
player_1_id	Number
player_2_id	Number
player_3_id	Number
player_4_id	Number
table_id	Number
losser_id	Number

טבלה זו שומרת את כל המשחקים שמתקיימים ושהתקיימו במערכת.
טבלה זו מתעדכנת בזמן אמת במקרה הצורך (למשל זמן סיום, מפסיד), ומכילה את המידע הבא:
מספר מזהה של המשחק, זמן ההתחלה, זמן הסיום, מספר מזהה של כל השחקנים המשתתפים במשחק (עד ארבעה שחקנים), מספר מזהה של שולחן המשחק (הבנוי בדמות של שחקן), ומספר מזהה של השחקן המפסיד.

4. **טבלת** Card_Table

Field Name	Data Type
ID	Number
color	Short Text
value	Number
special	Yes/No
source	Short Text

טבלה המכילה את המידע אודות כל הקלפים הקיימים במשחק. במשחק קיימים 67 קלפים, והמידע על כולם נשמר בטבלה זו. מידע זה כולל: מספר מזהה, צבע, ערך, האם הקלף מיוחד, ולינק לתמונת הקלף.

5. **טבלת** Friends_Table

Field Name	Data Type
ID	AutoNumber
user_1_id	Number
user_2_id	Number

טבלה זו מקשרת בין משתמשים בצורה של חברות. עצם של חברות נשמר בטבלה זו, כאשר מספר המזהה של החברות, והמספר המזהה של המשתמשים החברים.

6.**טבלת** Player_Card_Table

Field Name	Data Type
ID	AutoNumber
player_id	Number
card_id	Number

טבלה זו מקשרת בין כל שחקן במשחק לבין כל הקלפים אותם הוא מחזיק. טבלה זו מתעדכנת בזמן אמת, ומכילה את המספר המזהה של השחקן, של אחד הקלפים אותם הוא מחזיק, ומספר מזהה של הקשר ביניהם. כלומר, עבור כל שחקן, מספר שורות המקשרות בינו לבין קלף מסוים יהיה כמספר הקלפים אותם הוא מחזיק באותה העת.

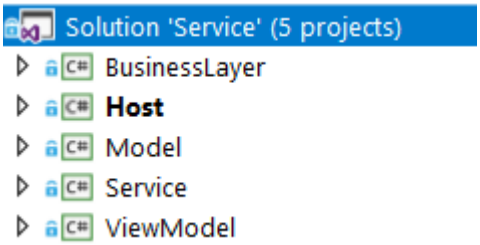
7. **טבלת** Player_Game_Table

Field Name	Data Type
ID	AutoNumber
player_id	Number
game_id	Number

טבלה זו מקשרת בין כל שחקן לכל משחק בו השתתף. טבלה זו מכילה עבור כל קשר בין שחקן למשחק את המספר המזהה של הקשר, של השחקן ושל המשחק.

צד שרת

מבנה התיקיות



צד השרת מורכב מ-5 פרוייקטים שונים, אשר לכל אחד קשר מיוחד למשנהו ופונקציונאליות מסויימת.

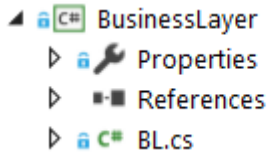
שם הפרוייקט	תיאור
BL (Business Logic)	שכבת הלוגיקה של המערכת. מכילה את כל הפונקציות החישוביות של המערכת, חוץ מלוגיקת המשחק עצמה, אשר קיימת בצד הלקוח.
Host	השכבה אשר מחזיקה את השירוץ Service באוויר.
Model	פרוייקט המכיל את ההגדרות של כל המחלקות אשר מרכיבות את המערכת. מערכת זו, אשר בנויה בצורת OOP, כלומר מונחה עצמים, מורכת ממספר עצמים שונים, אשר כל עצם מייצג חלק אחר במערכת. לכל טבלה במסד הנתונים תהיה מחלקה תואמת בפרוייקט זה.
Service	שכבה השירות, אשר מנהלת את היחסים שבין הלקוח לשרת, מכילה את כל הפונקציונאליות של הלקוח, ומכריעה עליה באמצעות חוזה בין הלקוח לשרת.
ViewModel	השכבה האחראית ליחסים שבין מערכת השרת לבין מסד הנתונים. בשכבה זו מומר המידע ממסד הנתונים אל עצמים המרכיבים את המערכת, וכמו כן מסונכרנים כל השינויים הנעשים למסד הנתונים.

פרוייקט ה -BL (Business Logic)

שכבה זו אחראית על לוגיקת המערכת, עיבוד מידע, חישובים שונים והעברת מידע מעובד אל הלקוח.

שכבה זו אומנם מכילה את לוגיקת המערכת, אך אינה מכילה את לוגיקת המשחק עצמה, אשר קיימת בצד הלקוח מטעמי יעילות ופשטות.

מבנה הפרוייקט



מחלקת BL

מחלקה זו מכילה מגוון רב של פונקציות. פונקציות רבות מועברות ממחלקת השירות אל מחלקה זו, זאת על מנת לשמור על מחלקת השירות כמחלקה האחראית על מעבר מידע בלבד ולא על חישובים ועיבוד מידע.

בין הפונקציות שהיא מכילה, הפונקציה `BlStartGame` האחראית לשדך מספר שחקנים שונים אל תוך משחק אחד.

```
// creates a new game and returns it
 public Game BlStartGame(Player p, int playerCount)
 {
     //if there is a game in gameList containing this player and if game is active
     Game temp = ActiveGameList.Find(g => g.Players.Find(q => q.UserId == p.UserId) != null);

     // return the game to the player!
     if (temp != null)
     {
         if (WaitingList[playerCount - 2].Count > 0) WaitingList[playerCount - 2].Clear();
         return temp;
     }

     // if the list contains a table, clear it! its old!
     if (WaitingList[playerCount - 2].Count > playerCount)
                                             WaitingList[playerCount - 2].Clear();


     // if the player isn't in the waiting list add him.
     if (WaitingList[playerCount - 2].Find(q => q.UserId == p.UserId) == null)
         WaitingList[playerCount - 2].Add(p);


     // if the player list is the size wanted including the requesting player,
     if (WaitingList[playerCount - 2].Count == playerCount &&
         p.UserId == WaitingList[playerCount - 2][playerCount - 1].UserId)
     {
         // create a new game containing all the players on the last player's request
         _game = new Game(WaitingList[playerCount - 2]); //create a new game with the players
```

```
            foreach (var t in _game.Players)
            {
                t.Hand = BuildShuffledHand(6, false);
            }

            _game.Players.Add(new Player()
                                    {Username = "table"}); // adding the table as a player

            // giving the table 100 shuffled cards
            _game.Players[playerCount].Hand = BuildShuffledHand(59, true);

            _game = BlStartGameDatabase(_game); // add this game to the database!

            // if no active games exist, no background worker is active.
            // so set one:
        if (ActiveGameList.Count == 0)
            {
                SetSaveChanges();
            }

            ActiveGameList.Add((Game) _game.Clone()); // add this game to the active games list

            _game.StartTime = DateTime.Now; // start the game

            return _game; // return this game
        }

        return null;
}
```

פעולה זו מקבלת בקשות להצטרפות למשחק משחקנים כל פרק זמן מסויים, ושומרת בקשות אלו בצורה זמנית. כאשר ישנו מספר מסוים של שחקנים אשר מחפשים לשחק עם מספר משותף של שחקנים, הפעולה יוצרת משחק חדש עבורם אותם ומחלקת לכל אחד מהם את הקלפים שלו. לסיום מוחזר העצם מסוג משחק לשחקנים שהמתינו לו.

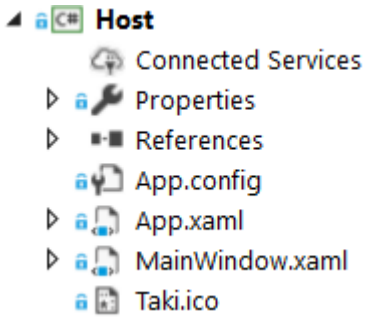
בנוסף לפעולה זו, קיימות פעולות נוספות, וביניהן:

-פעולה אשר בונה יד קלפים רנדומלית לכל שחקן
-פעולה הבודקת את כניסתם של משתמשים למערכת (Login)
-פעולה שבודקת נתונים ליצירת משתמש חדש (Register)
-פעולה ששומרת את השינויים שנעשו במהלך המשחק במסד הנתונים, בכל פרק זמן מסויים.

פרוייקט Host

שכבה זו אחראית על החזקת השירות Service באוויר, בשימוש ובקישור בין הלקוח לשרת.

מבנה הפרוייקט



פרוייקט זה מכיל גם את הלוגו של השרת המורץ בצורה חלונאית בווינדווס.

מחלקה לדוגמה

מחלקת MainWindow.xaml

מחלקה זו אחראית על פתיחת הקישור בין הלקוח לשרת, ועל איתחול Service והחזקתו באוויר.

```
namespace Host
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            ServiceHost service = new ServiceHost(typeof(Service.Service));
            service.Open();
        }
    }
}
```

הגדרת הרשת של השרת

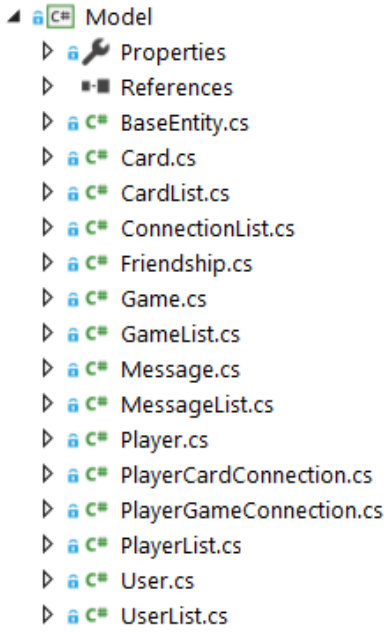
הגדרות הרשת שמורות בקובץ App.config הנמצא בפרוייקט Host.
בהגדרות אלו מתרחש החלק של החיבור החיצוני בעזרת ABC : **A**dress **B**inding **C**ontract.
ניתן לראות שפרוייקט ה-WCF מוגדר כנותן השירות המאזין לפורט הכתוב ב baseAdress, במקרה זה http://10.116.2.90:8733/Design_Time_Addresses/Service/Service/, ושהוא מאזין באמצעות פרוטוקול http בסיסי כפי שכתוב ב-binding של ה-endpoint, ומכיל גם חוזה המקבץ את הפעולות הנגישות ללקוח והוא הממשק interface שכתבנו Iservice.

```
<?xml version="1.0" encoding="utf-8"?>

<configuration>

  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" />
  </system.web>
  <!-- When deploying the service library project, the content of the config file must be added to the
host's
  app.config file. System.Configuration does not support config files for libraries. -->
  <system.serviceModel>
    <services>
      <service name="Service.Service">
        <host>
          <baseAddresses>
            <add baseAddress="http://10.116.2.90:8733/Design_Time_Addresses/Service/Service/" />
            <!--<add baseAddress = "http://192.168.0.18:8733/Design_Time_Addresses/Service/Service/" />-->
          </baseAddresses>
        </host>
        <!-- Service Endpoints -->
        <!-- Unless fully qualified, address is relative to base address supplied above -->
        <endpoint address="" binding="basicHttpBinding" contract="Service.IService">
          <!--
              Upon deployment, the following identity element should be removed or replaced to reflect the
              identity under which the deployed service runs.  If removed, WCF will infer an appropriate
identity
              automatically.
          -->
          <identity>
            <dns value="localhost" />
          </identity>
        </endpoint>
        <!-- Metadata Endpoints -->
        <!-- The Metadata Exchange endpoint is used by the service to describe itself to clients. -->
        <!-- This endpoint does not use a secure binding and should be secured or removed before
deployment -->
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-- To avoid disclosing metadata information,
          set the values below to false before deployment -->
          <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True" />
          <!-- To receive exception details in faults for debugging purposes,
          set the value below to true.  Set to false before deployment
          to avoid disclosing exception information -->
          <serviceDebug includeExceptionDetailInFaults="True" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

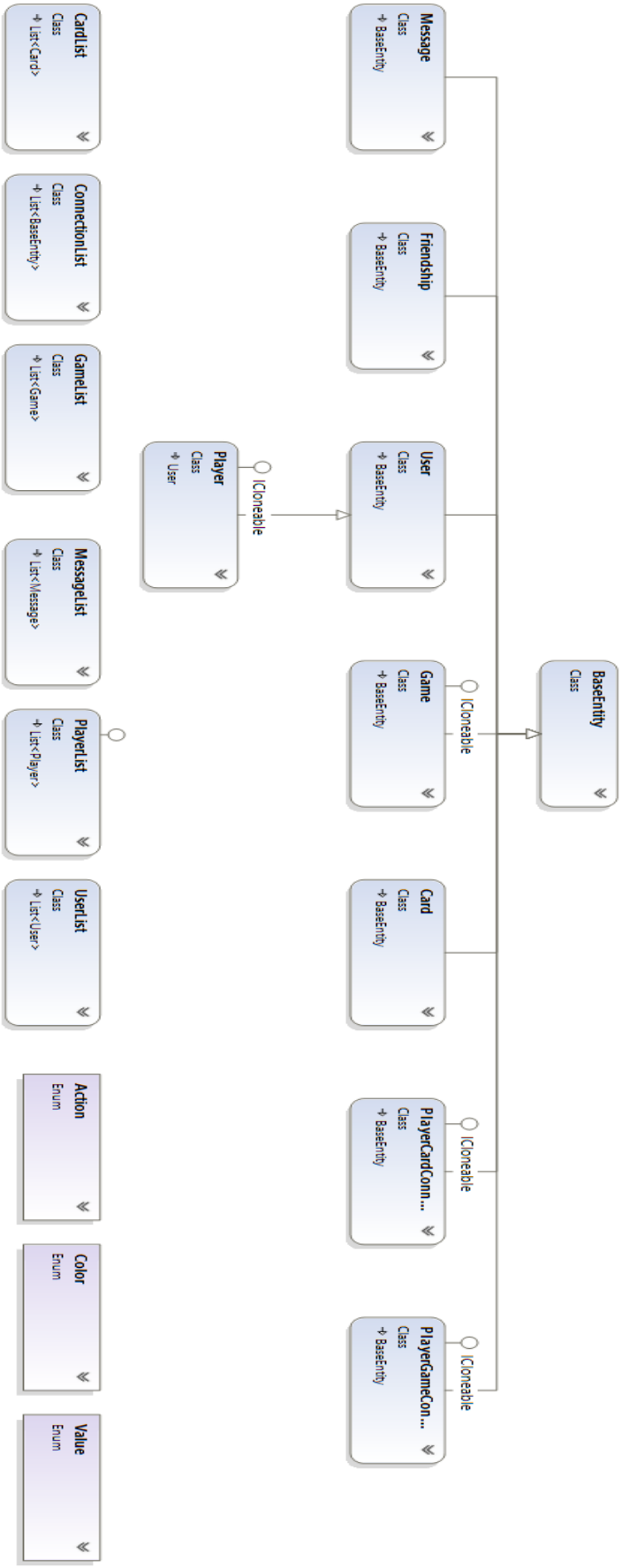
פרוייקט Model

שכבת המודל משקפת את מבנה הטבלאות במסד הנתונים, ובאמצעות מחלקות אלו נמיר את המידע השמור במסד הנתונים אל עצמים המתאימים לקוד בייצוג OOP(Object Oriented Programming).
לכל טבלה במסד הנתונים תהיה מחלקה רלוונטית ב Model, המייצגת עצם המבוסס על המאפיינים בטבלה החופפת במסד הנתונים.
בין מחלקות אלו יהיו קשרי גומלין וירושה החופפים לאלו שבמסד הנתונים.

מבנה הפרוייקט



UML


BaseEntity
Class
ICloneable
ICloneable
ICloneable
Message
Class
+ BaseEntity
Friendship
Class
+ BaseEntity
User
Class
+ BaseEntity
Game
Class
+ BaseEntity
Card
Class
+ BaseEntity
PlayerCardConn...
Class
+ BaseEntity
PlayerGameCon...
Class
+ BaseEntity
ICloneable
Player
Class
+ User
CardList
Class
+ List<Card>
ConnectionList
Class
+ List<BaseEntity>
GameList
Class
+ List<Game>
MessageList
Class
+ List<Message>
PlayerList
Class
+ List<Player>
UserList
Class
+ List<User>
Action
Enum
Color
Enum
Value
Enum

מחלקות דוגמה

מחלקת BaseEntity

מחלקה זו היא מחלקת הבסיס של כל המחלקות המרכיבות את הפרוייקט Model.
התכונה המשותפת לכל עצם (entity) היא id, ולכן יהיה זה המאפיין היחיד במחלקה זו.

```
namespace Model
{
    public class BaseEntity
    {
        public int Id { get; set; }
    }
}
```

מחלקת User

מחלקה זו יורשת ישירות מהמחלקה BaseEnitity, ומייצגת עצם מסוג משתמש.
לכל עצם מוגדרות פעולות Get-ו Set ישירות, אשר מאפשרות קבלת ושינוי מידע אודות העצם מכל מקום בקוד אשר מחזיק באותו העצם.
כיוון שמחלקה זו יורשת ממחלקת האב BaseEntity, בדומה לכל מחלקה אחרת ב-Model, היא תכיל אוטומטית את המאפיין id.

```
namespace Model
{
    public class User : BaseEntity
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public string ProfileImage { get; set; }
        public int Score { get; set; }
        public int Level { get; set; }
        public bool Admin { get; set; }
        public int Wins { get; set; }
        public int Losses { get; set; }
    }
}
```

מחלקת PlayersList

מחלקת אוסף זו יורשת מהמחלקה הגנרית של #c, List<T>. נשתמש במחלקת אוסף שכזו כדי להעביר אוסף של עצמים בתקשורת מהשרת ללקוח.
יש לציין, כי מחלקת Player יורשת ממחלקת User את כלל תכונותיה, כולל, בין היתר, את ה-id שמקורו במחלקת BaseEntity.

```
namespace Model
{
    [CollectionDataContract]
    public class PlayerList : List<Player>
    {
        public PlayerList(){}

        public PlayerList(IEnumerable<Player> list) : base(list){}
```

```
        public PlayerList(IEnumerable<BaseEntity> list) :
                                                base(list.Cast<Player>().ToList()){}

    }
```

מחלקת Message

מחלקה זו היא חלק חשוב בהתנהלות המשחק, וחלק הכרחי ביחסים שבין הלקוח לשרת.
מחלקה זו היא מחלקת "הודעה" – בכל פעם ששחקן בונה בקשה לעשות מהלך מסויים, או שמשתנה מצבו של שחקן זה ביחס למשחק(תורו נגמר, הוא עזב את המשחק, הוא ניצח וכו'), בונה הלקוח עצם מסוג Message.
עצם זה, המכיל מידע אודות הפעולה שהלקוח מבקש לעשות או שכבר עשה בשכבת הלקוח, מועברת אל השרת באמצעות ה Service, ומעובדת במחלקת BL(BussinessLayer).
כמו כן, הודעה זו מועברת אל שאר השחקנים במשחק, כך שיוכלו לעדכן את תמונת המצב של המשחק בעצם המשחק המוצג אצלם, כך שכל השחקנים מעודכנים במעשים של משניהם.

כל הודעה מורכבת מכמה פרמטרים:

-מוען ההודעה (מסוג שחקן)
-הפעולה אותה הוא מבקש לבצע (מסוג Action, שהוא enum אשר יצרתי למטרה זו).
-הקלף עליו הוא מבקש לבצע פעולה זו (מסוג Card, אך כשקלף אינו מעורב בפעולה הוא נשאר null).
-מספר מזהה של המשחק הנוכחי (מסוג int, לצורכי זיהוי).
-נמען ההודעה (מסוג int, ובמקרה הצורך הוא המספר המזהה של השחקן אליו הפעולה מכוונת).

```
namespace Model
{
    [DataContract]
    public enum Action
    {
        Add,
        Remove,
        NextTurn,
        PlayerQuit,
        SwitchRotation,
        SwitchHand,
        PlusTwo,
        Win,
        Loss
    }

    [DataContract]
    public class Message : BaseEntity
    {
        public Player Target { get; set; }
        public Action Action { get; set; }
        public Card Card { get; set; }
        public int GameId { get; set; }
        public int Reciever { get; set; }
    }
}
```

Enum זה מכיל את כלל הפעולות אותן יכול השחקן לבצע.

פרוייקט ViewModel

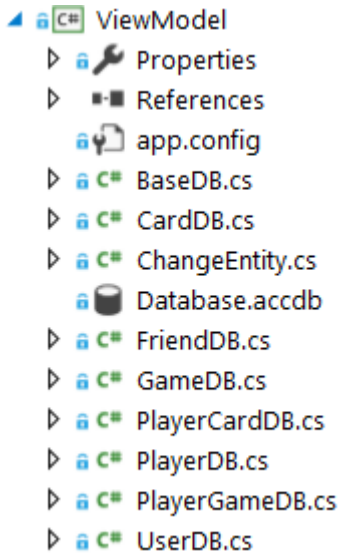
מחלקת ViewModel היא מחלקת המיפוי, היא המחלקה המקשרת בין מחלקת המודל לבין הטבלה המתאימה לכל מחלקה במסד הנתונים.
באמצעותה נשלוף מידע ממסד הנתונים, ומהם ניצור עצמים המכילים מידע זה ממחלקת Model, נוסיף נעדכן ונמחק מידע ממסד הנתונים באמצעות עצמים קיימים.

מחלקה זו מאפשרת CRUD (**C**reate, **R**ead, **U**pdate, **D**elete) באופן שלם למידע הקיים במסד הנתונים.

עבור כל טבלה ניצור מחלקה אשר תממש את הצרכים של התכנית מאותה טבלה, ותבנה עצם מהסוג התואם מפרוייקט Model.

כל מחלקה ב-ViewModel מממשת בקוד את קשרי הגומלין התואמים ממחלקת Model.

מבנה התיקיות



מחלקות דוגמה

מחלקת BaseDB

מחלקה זו, בדומה למחלקת BaseEnitity אותה היא מייצגת ביחס בין Model למסד הנתונים, היא מחלקת בסיס.

מחלקה זו אינה תלויה בטבלה כלשהי, היא כללית מאוד ומתאימה לכל פרוייקט אשר יתבסס על ארכיטקטורה זו.

מחלקה זו מכילה את הפעולות הבסיסיות של ניהול מסד הנתונים, וכל מחלקה נוספת מפרוייקט זה תקבל את פעולות אלו ותתאים אותן אליהם בירושה.

```
namespace ViewModel
{
    public abstract class BaseDb
    {


        protected OleDbCommand Command;
        protected OleDbConnection Con;
        protected string ConnectionString;
        protected OleDbDataReader Reader;

        /*-----------------------------------------------------------------*/

        public BaseDb()
        {
            string s1 = "";

            if (ConnectionString == null)
            {
                string[] arguments = Environment.GetCommandLineArgs();
                string s;
                if (arguments.Length == 1)
                {
                    s = arguments[0];
                }
                else
                {
                    s = arguments[1];
                    s = s.Replace("/service:", "");
                }

                string[] ss = s.Split('\\');

                int x = ss.Length - 4;
                ss[x] = "ViewModel";
                Array.Resize(ref ss, x + 1);

                s1 = string.Join("\\", ss);
            }

            ConnectionString =
                @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + s1 +
                @"\Database.accdb; Persist Security Info = True";

            Con = new OleDbConnection(ConnectionString);
            Command = new OleDbCommand();
        }


        //base entity
        protected abstract BaseEntity NewEntity();


        //base model
        protected abstract BaseEntity CreateModel(BaseEntity entity);


        public abstract void CreateInsertSql(BaseEntity entity, OleDbCommand command);
        public abstract void CreateUpdateSql(BaseEntity entity, OleDbCommand command);
        public abstract void CreateDeleteSql(BaseEntity entity, OleDbCommand command);

       //executed update, insert, delete lists. (delete in updated)
        protected static List<ChangeEntity> Inserted = new List<ChangeEntity>();
        protected static List<ChangeEntity> Updated = new List<ChangeEntity>();
```

תכונות כלליות,
מיועדות לעבודה מול מסד
הנתונים מסוג אקסס

מתודה שנועדה למצוא וליצור את ה-PATH העדכני של
מסד הנתונים.
זאת על מנת שלא "נאבד" את מסד הנתונים כאשר נזיז
את הפרוייקט ממחשב למחשב.

```
        public virtual void Insert(BaseEntity entity)
        {
            BaseEntity reqEntity = NewEntity();
            if (entity != null && entity.GetType() == reqEntity.GetType())
                Inserted.Add(new ChangeEntity(CreateInsertSql, entity));
        }

        public virtual void Update(BaseEntity entity)
        {
            BaseEntity reqEntity = NewEntity();
            if (entity != null && entity.GetType() == reqEntity.GetType())
                Updated.Add(new ChangeEntity(CreateUpdateSql, entity));
        }

        public virtual void Delete(BaseEntity entity)
        {
            BaseEntity reqEntity = NewEntity();
            if (entity != null && entity.GetType() == reqEntity.GetType())
                Updated.Add(new ChangeEntity(CreateDeleteSql, entity));
        }

        /*-----------------------------------------------------------------*/

        public int SaveChanges()
        {
            OleDbCommand command = new OleDbCommand();
            OleDbTransaction trans = null;

            int recordsAffected = 0;
            int errorIndex = 0;
            try
            {
                Con.Open();
                trans = Con.BeginTransaction();
                command.Connection = Con;
                command.Transaction = trans;

                //inserted
                foreach (var item in Inserted)
                {
                    command.Parameters.Clear();
                    item.CreateSql(item.Entity, command);
                    recordsAffected += command.ExecuteNonQuery();


                    command.CommandText = "SELECT @@Identity"; //get last ID on this
session
                    int temp = (int) command.ExecuteScalar();
                    item.Entity.Id = temp;

                    errorIndex++;
                }

                Inserted.Clear();

                //updated, deleted
                foreach (var item in Updated)
                {
                    command.Parameters.Clear();
                    item.CreateSql(item.Entity, command);
                    recordsAffected += command.ExecuteNonQuery();
                }
```

מתודה זו היא היחידה אשר מבצעת בפועל שינויים במסד הנתונים.
המתודה, כאשר נקראת ממקום כלשהו בפרוייקט, עוברת בלולאה על הפעולות הנתבקשות להיעשות במסד הנתונים ומבצעת אותן.

שיטה זו מאפשרת הקבצה של מספר פעולות לתוך מהלך אחד, זאת על מנת לאפשר אסינכרוניות, כיוון שעד שמתודה זו לא נקראת אף פעולה אינה נעשית.

כך ניתן לשנות נתונים אודות אופן כל פעולה, במהלך ביצועה התיאורתי בצד לקוח – עד ביצוע הממשי בצד השרת.

```
            Updated.Clear();

            trans.Commit();
        }
        catch (Exception e)
        {
            Debug.WriteLine("\n" + e.Message + "\nSQL:" + command.CommandText);

            try
            {
                trans.Rollback();
            }
            catch (Exception ex2)
            {
                // This catch block will handle any errors that may have occurred
                // on the server that would cause the rollback to fail, such as
                // a closed connection.
                Console.WriteLine("Rollback Exception Type: {0}", ex2.GetType());
                Console.WriteLine("  Message: {0}", ex2.Message);
            }
        }
        finally
        {
            Reader?.Close();

            if (Con.State == ConnectionState.Open) Con.Close();
        }

        return recordsAffected;
    }

    /*-----------------------------------------------------------------*/

    protected List<BaseEntity> Select()
    {
        List<BaseEntity> list = new List<BaseEntity>();

        try
        {
            Command.Connection = Con;
            Con.Open();
            Reader = Command.ExecuteReader();

            while (Reader.Read())
            {
                BaseEntity entity = NewEntity();
                list.Add(CreateModel(entity));
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Error! " + e.Message);
        }
        finally
        {
            Reader?.Close();

            if (Con.State == ConnectionState.Open) Con.Close();
        }

        return list;
    }
}
```

```
}
```

מחלקת ChangeEnitity

מחלקה המכילה את המתודה המייצרת את ה-SQL הנדרש ליצירת משפט ה-SQL ואת הנתונים לעדכון.
ממחלקה זו נייצר בפעולות Insert/ Update/ Delete, את שלושת האוספים בהם נשתמש בפעולה SaveChnages.

```
namespace ViewModel
{
    //create SQL
    public delegate void CreateSql(BaseEntity entity, OleDbCommand command);

    public class ChangeEntity
    {
        public ChangeEntity(CreateSql createSql, BaseEntity entity)
        {
            Entity = entity;
            CreateSql = createSql;
        }

        public BaseEntity Entity { get; set; }

        public CreateSql CreateSql { get; set; }
    }
}
```

מחלקת GameDB

מחלקה המקשרת בין מחלקת Game לבין טבלת מסד הנתונים Game_Table.
באמצעותה נשלוף מידע עבור מחלקת Game, נעדכן מידע אודות משחקים המתרחשים בכל עת, נוסיף ונמחק רשימות.
ניתן לראות במחלקה זו פעולות המקשרות בין עצמים שונים לבין כל משחק (שחקן מסוג Player, ועצם מסוג PlayerGameConnection המקשר בין כל שחקן למשחק)

נוסף לכך, ניתן לראות את הירושה מ BaseDB, כך שמחלקה זו מממשת רק את הדברים הייחודיים לה.

```
namespace ViewModel
{
    public class GameDb : BaseDb
    {
        protected override BaseEntity NewEntity()
        {
            return new Game();
        }
```

```
protected override BaseEntity CreateModel(BaseEntity entity)
{
    PlayerDb db = new PlayerDb();
    Game game = entity as Game;

    game.Id = (int) Reader["ID"];


    game.Players.Add((Player) db.GetPlayerById((int) Reader["player_1_id"]));
    game.Players.Add((Player) db.GetPlayerById((int) Reader["player_2_id"]));
    game.Players.Add((Player) db.GetPlayerById((int) Reader["player_3_id"]));
    game.Players.Add((Player) db.GetPlayerById((int) Reader["player_4_id"]));
    game.Players.Add((Player) db.GetPlayerById((int) Reader["table_id"]));

    game.StartTime = (DateTime) Reader["start_date"];
    game.EndTime = (DateTime) Reader["end_date"];
    game.Loser = (int) Reader["losser_id"];

    return game;
}


public GameList SelectAll()
{
    Command.CommandText = "SELECT * FROM Game_Table";
    GameList temp = new GameList(Select());
    return temp;
}


public Game GetLastGame()
{
    Command.CommandText =
             "SELECT * FROM Game_Table WHERE ID = (SELECT MAX(ID) FROM Game_Table)";
    GameList temp = new GameList(Select());
    return temp.Count > 0 ? temp[0] : new Game(0);
}


public Game GetGameById(int id)
{
    Command.CommandText = "SELECT * FROM Game_Table WHERE [ID] = " + id + "";
    GameList temp = new GameList(Select());
    return temp.Count > 0 ? temp[0] : null;
}


public GameList SelectByUserId(int userId)
{
    Command.Parameters.Clear();

    Command.CommandText = "SELECT * FROM Game_Table WHERE(ID IN " +
                          "(SELECT Player_Game_Table.game_id" +
                          " FROM(Player_Table INNER JOIN" +
                          " Player_Game_Table ON Player_Table.ID =
                                                Player_Game_Table.player_id)" +
                          " WHERE (Player_Table.user_id = @Id)))";

    Command.Parameters.Add(new OleDbParameter("@Id", userId));

    GameList temp = new GameList(Select());
    return temp;
}
```

```
public GameList SelectByUsersId(int u1, int u2)
{
    GameList u1Games = this.SelectByUserId(u1);
    GameList u2Games = this.SelectByUserId(u2);

    GameList mutualGames = new GameList();

    foreach (Game g1 in u1Games)
    {
        foreach (Game g2 in u2Games)
        {
            if (g1.Id == g2.Id)
            {
                mutualGames.Add(g1);
            }
        }
    }
    return mutualGames;
}


public int GetLastGameId()
{
    Command.Parameters.Clear();

    Command.CommandText = "SELECT MAX(ID) FROM Game_Table";
    GameList temp = new GameList(Select());
    return temp.Count > 0 ? temp[0].Id : new Game(0).Id;
}


public override void Insert(BaseEntity entity)
{
    if (entity is Game u) Inserted.Add(new ChangeEntity(CreateInsertSql, entity));
}


public override void Update(BaseEntity entity)
{
    if (entity is Game u) Updated.Add
       (new ChangeEntity(CreateUpdateSql, entity));
}


public override void Delete(BaseEntity entity)
{
    Game game = entity as Game;

    //delete all connections related to this game
    PlayerGameDb pGdb = new PlayerGameDb();

    ConnectionList pg = pGdb.SelectByGame(game);

    if (pg != null)
    {
      //delete all player-games connections to this Game id
                                        using PlayerGameDB.CreateDeleteSql
        foreach (PlayerGameConnection c in pg)
            Updated.Add(new ChangeEntity(pGdb.CreateDeleteSql, c));

        //delete the player itself
        Updated.Add(new ChangeEntity(CreateDeleteSql, entity));
    }
}
```

מחיקת משחק גוררת איתה פעולות נוספת אותן צריך לבצע עקב קיום קשרי הגומלין בין העצמים ובין הטבלות במסד הנתונים.
נדרוס את פעולת ה-Delete הכללית אותה אנו יורשים ממחלקת האב BaseDB, ונתאים אותה כך שנמחק גם את העצמים הקשורים לאותו משחק.
כך למשל, אנו מוחקים במקרה זה את כל קשר בין שחקן לאותו המשחק באמצעות המחלקה PlayerGameDB.

```
        public override void CreateDeleteSql(BaseEntity entity, OleDbCommand command)
        {
            command.Parameters.Clear();

            Game game = entity as Game;

            command.CommandText = "DELETE FROM Game_Table WHERE ID = @game_id";

            //PlayerGameDB playerGame = new PlayerGameDB();

            //parameters

            command.Parameters.Add(new OleDbParameter("@game_id", game.Id));

            Console.WriteLine("All Connections and games related to this game have been
deleted");
        }


        public override void CreateInsertSql(BaseEntity entity, OleDbCommand command)
        {
            command.Parameters.Clear();

            Game g = entity as Game;

            command.CommandText =
                "INSERT INTO Game_Table ([start_date], [end_date], [player_1_id],
[player_2_id], [player_3_id], [player_4_id], [table_id], [losser_id]) VALUES (" +
                "'" + g.StartTime.ToString("G") + "', '" + g.EndTime.ToString("G") +
                "', @p1, @p2, @p3, @p4, @table, @loss)";

            command.Parameters.AddWithValue("@p1", g.Players[0].Id);
            command.Parameters.AddWithValue("@p2", g.Players[1].Id);

            switch (g.Players.Count)
            {
                case 3:
                    command.Parameters.AddWithValue("@p3", int.Parse("0"));
                    command.Parameters.AddWithValue("@p4", int.Parse("0"));
                    command.Parameters.AddWithValue("@table", g.Players[2].Id);
                   break;

                case 4:
                    command.Parameters.AddWithValue("@p3", g.Players[2].Id);
                    command.Parameters.AddWithValue("@p4", int.Parse("0"));
                    command.Parameters.AddWithValue("@table", g.Players[3].Id);
                    break;

                case 5:
                    command.Parameters.AddWithValue("@p3", g.Players[2].Id);
                    command.Parameters.AddWithValue("@p4", g.Players[3].Id);
                    command.Parameters.AddWithValue("@table", g.Players[4].Id);
                    break;
            }

            //parameters
            command.Parameters.AddWithValue("@loss", int.Parse("0"));

            Console.WriteLine("Game [" + g.Id + "] Added to the Database");
        }
```

```
        public override void CreateUpdateSql(BaseEntity entity, OleDbCommand command)
        {
            command.Parameters.Clear();

            Game g = entity as Game;

            command.CommandText =
                "UPDATE Game_Table SET start_date = '" + g.StartTime.ToString("G") + "',
end_date = '" +
                g.EndTime.ToString("G") +
                "', player_1_id = @p1, player_2_id = @p2, player_3_id = @p3, player_4_id =
@p4, table_id = @table, losser_id = @losser";

            command.Parameters.AddWithValue("@p1", g.Players[0].Id);
            command.Parameters.AddWithValue("@p2", g.Players[1].Id);

            switch (g.Players.Count)
            {
                case 3:
                    command.Parameters.AddWithValue("@p3", int.Parse("0"));
                    command.Parameters.AddWithValue("@p4", int.Parse("0"));
                    command.Parameters.AddWithValue("@table", g.Players[2].Id);
                    break;

                case 4:
                    command.Parameters.AddWithValue("@p3", g.Players[2].Id);
                    command.Parameters.AddWithValue("@p4", int.Parse("0"));
                    command.Parameters.AddWithValue("@table", g.Players[3].Id);
                    break;

                case 5:
                    command.Parameters.AddWithValue("@p3", g.Players[2].Id);
                    command.Parameters.AddWithValue("@p4", g.Players[3].Id);
                    command.Parameters.AddWithValue("@table", g.Players[4].Id);
                    break;
            }
```

מחלקת PlayerCardDB

מחלקה זו מקשרת בין PlayerGameConnection שבפרוייקט Model לבין טבלת Player_Game_Table שבמסד הנתונים.

טבלה זו אחראית על שליפת מידע אודות הקשר של שחקנים למשחקים, ויצירת קשרים מסוג PlayerGameConnection היורשים ממחלקת Connection בין כל שחקן למשחק בו הוא משתתף.

בטבלה זו מתודות המופעלות בתגובה למהלכים של שחקן כלשהו במהלך המשחק. כך למשל, כאשר שחקן מעביר קלף כלשהו לשולחן, הקשר בין אותו השחקן לאותו קלף ימחק, ויתסווף קשר חדש בין השולחן לאותו קלף למסד הנתונים.

```
namespace ViewModel
{
    public class PlayerCardDb : BaseDb
    {
        protected override BaseEntity NewEntity()
        {
            return new PlayerCardConnection();
        }
```

```
        protected override BaseEntity CreateModel(BaseEntity entity)
        {
            PlayerDb playerDb = new PlayerDb();
            CardDb cardDb = new CardDb();

            PlayerCardConnection con = entity as PlayerCardConnection;
            con.Id = (int) Reader["ID"];
            con.Player = playerDb.GetPlayerById((int) Reader["player_id"]);
            con.Card = cardDb.SelectById((int) Reader["card_id"]);
            return con;
        }


        public ConnectionList SelectByPlayer(Player player)
        {
            Command.CommandText = "SELECT * FROM Player_Card_Table WHERE [player_id] = @id";

            //parameters
            Command.Parameters.Add(new OleDbParameter("@id", player.Id));

            ConnectionList conList = new ConnectionList(Select());
            return conList;
        }


        public PlayerCardConnection GetConnectionByPlayerIdAndCardId(Player player, Card card)
        {
            Command.CommandText =
                "SELECT * FROM Player_Card_Table WHERE [player_id] = @playerId AND [card_id] =
@cardId";

            // parameters
            Command.Parameters.Add(new OleDbParameter("@playerId", player.Id));
            Command.Parameters.Add(new OleDbParameter("@cardId", card.Id));

            ConnectionList conList = new ConnectionList(Select());
            return (PlayerCardConnection) conList[0];
        }


        public ConnectionList GetConnectionsByPlayerId(Player player)
        {
            Command.CommandText = "SELECT * FROM Player_Card_Table WHERE [player_id] =
@playerId";

            // parameters
            Command.Parameters.Add(new OleDbParameter("@playerId", player.Id));

            ConnectionList conList = new ConnectionList(Select());
            return conList;
        }


        public void SwitchConnectionsByPlayersId(Player player1, Player player2)
        {
            ConnectionList hand1 = GetConnectionsByPlayerId(player1);
            ConnectionList hand2 = GetConnectionsByPlayerId(player2);

            foreach (PlayerCardConnection c in hand1)
            {
               // switch the owner of the cards to the other player
                c.Player = player1;
                Update(c); // update
            }
```

מתודה זו אחראית על ביצוע הפעולה
של החלפת הקלפים בין שחקנים
הקיימת במשחק טאקי.
על מנת שהקשרים בין השחקנים
לקלפים במסד הנתונים ישנו בהתאם
למתרחש בשכבת הלוגיקה של המשחק
(שכבת פרוייקט ה BL
(BussinessLogic)), הקשרים הללו לא
ימחקו, אלה יעודכנו בלבד כך
שהמספרים המזהים של השחקנים
שבקשרים האלו יתחלפו זה בזה.

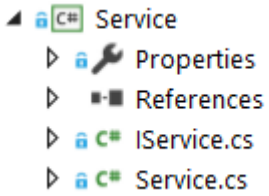
```
            foreach (PlayerCardConnection c in hand2)
            {
                c.Player = player2; // switch the owner of the cards to the other player
                Update(c); // update
            }
        }


        public void InsertList(ConnectionList entity)
        {
            ConnectionList cl = entity;
            foreach (var playerCardConnection in cl)
                if (playerCardConnection != null)
                    Inserted.Add(new ChangeEntity(CreateInsertSql, playerCardConnection));
        }


        public void Insert(Game game)
        {
            foreach (Player p in game.Players)
            {
                foreach (Card c in p.Hand)
                {
                    Inserted.Add(new ChangeEntity(CreateInsertSql, new PlayerCardConnection()
                    {
                        Player = p,
                        Card = c
                    }));
                }
            }
        }


        public override void Delete(BaseEntity entity)
        {
            if (entity != null)
            {
                Updated.Add(new ChangeEntity(CreateDeleteSql, entity));
            }
        }


        public override void CreateInsertSql(BaseEntity entity, OleDbCommand command)
        {
            PlayerCardConnection con = entity as PlayerCardConnection;

            command.CommandText = "INSERT INTO Player_Card_Table (player_id, card_id) VALUES
(@player_id, @card_id)";

            //parameters

            command.Parameters.Add(new OleDbParameter("@player_id", con.Player.Id));
            command.Parameters.Add(new OleDbParameter("@card_id", con.Card.Id));

            Console.WriteLine("PlayerCardConnection between player [" + con.Player.Id + "] and
card [" + con.Card.Id +
                              "] INSERTED");
        }

        public override void CreateDeleteSql(BaseEntity entity, OleDbCommand command)
        {
            PlayerCardConnection con = entity as PlayerCardConnection;

            command.CommandText = "DELETE FROM Player_Card_Table WHERE [ID] = @id ";
```

```
            //parameters

            command.Parameters.Add(new OleDbParameter("@id", con.Id));

            Console.WriteLine("PlayerCardConnection between player [" +
                              con.Player.Id + "] and card [" + con.Card.Id +"] DELETED");
        }


        public override void CreateUpdateSql(BaseEntity entity, OleDbCommand command)
        {
            PlayerCardConnection con = entity as PlayerCardConnection;

            command.CommandText = "UPDATE Player_Card_Table SET [player_id] = @player_id WHERE
[ID] = @id";

            //parameters

            command.Parameters.Add(new OleDbParameter("@player_id", con.Player.Id));
            command.Parameters.Add(new OleDbParameter("@id", con.Id));

            Console.WriteLine("PlayerCardConnection between player [" +
                              con.Player.Id + "] and card [" + con.Card.Id +"] UPDATED");
        }
    }
}
```

שירותי הרשת (פרוייקט ה-WcfServiceLibrary)

בפרוייקט זה נמצא ממשק IService המכיל את כל בפעולות הנמצאות בשרת, שהלקוח יכול להשתמש בהן. הוא מגדיר את כל הפונקציונאליות של השירות, בהן יכול להשתמש הלקוח.
לממשק מחלקה Service הממשת אותו.

מבנה התיקיות



ממשק שירות הרשת IService

```
namespace Service
{
    [ServiceContract]
    public interface IService
    {
        //User Management

        [OperationContract]
        User Login(string username, string password);

        [OperationContract]
        bool Logout(int userId);

        [OperationContract]
        bool Register(string firstName, string lastName, string username, string
password);

        [OperationContract]
        int DeleteUser(User user);

        [OperationContract]
        int UpdateUser(User user);

        [OperationContract]
        bool PasswordAvailable(string password);

        [OperationContract]
        GameList GetAllUserGames(int userId);

        [OperationContract]
        User GetUserByUsername(string username);

        [OperationContract]
        User GetUserById(int id);

        [OperationContract]
        bool UsernameAvailable(string username);
```

```
        [OperationContract]
        UserList GetAllUserFriends(int userId);

        [OperationContract]
        GameList GetMutualGames(int u1, int u2);

        [OperationContract]
        bool AreFriends(int user1Id, int user2Id);

        [OperationContract]
        void MakeFriends(User u1, User u2);

        [OperationContract]
        void RemoveFriend(User u1, User u2);

        [OperationContract]
        UserList GetAllUsers();

        //cards management

        [OperationContract]
        CardList GetCardList();

        [OperationContract]
        CardList BuildDeck();

        //game management

        [OperationContract]
        Game StartGame(Player p, int playerCount);

        [OperationContract]
        PlayerList GetPlayerList();

        [OperationContract]
        void AddAction(Message m);

        [OperationContract]
        void AddActions(MessageList ml);

        [OperationContract]
        MessageList DoAction(int gameId, int playerId);

        [OperationContract]
        bool StopSearchingForGame(Player p);

        [OperationContract]
        int GetPlayersFound(int playerCount);

        [OperationContract]
        bool PlayerQuit(Player p);

        [OperationContract]
        int SaveChanges();
    }
}
```

מחלקת Service

מחלקה זו מממשת את הממשק Iservice.
תפקידה של מחלקה זו הוא לקבל את בקשות הלקוח, ולהפנות ולהעביר אותן למקום המתאים.
פעולה זו מקבלת בקשה בצד הלקוח, מייצרת עצם מסוג BL שהוא חלק ממחלקת ה- BussinessLayer,
שם הנתונים מועבדים ומוחזרת תשובה בהתאם לצורך.

```
namespace Service
{
    public class Service : IService
    {
        public static MessageList PendingChanges { get; set; } = new MessageList();

        public CardList BuildDeck()
        {
            Bl bl = new Bl();
            return bl.BlBuildDeck();
        }

        public UserList GetAllUsers()
        {
            Bl bl = new Bl();
            return bl.BlGetAllUsers();
        }

        public User GetUserByUsername(string username)
        {
            Bl bl = new Bl();
            return bl.BlGetUserByUsername(username);
        }

        public User GetUserById(int id)
        {
            Bl bl = new Bl();
            return bl.BlGetUserById(id);
        }

        public User Login(string username, string password)
        {
            Bl bl = new Bl();
            return bl.BlLogin(username, password);
        }

        public bool Logout(int userId)
        {
            Bl bl = new Bl();
            return bl.BlLogout(userId);
        }

        public bool Register(string firstName, string lastName, string username, string password)
        {
            Bl bl = new Bl();
            return bl.BlRegister(firstName, lastName, username, password);
        }

        public int DeleteUser(User user)
        {
            Bl bl = new Bl();
            return bl.BlDeleteUser(user);
        }

        public int UpdateUser(User user)
        {
            Bl bl = new Bl();
            return bl.BlUpdateUser(user);
        }

        public bool PasswordAvailable(string password)
        {
            Bl bl = new Bl();
            return bl.BlPasswordAvailable(password);
        }
```

```
public bool UsernameAvailable(string username)
{
    Bl bl = new Bl();
    return bl.BlUsernameAvailable(username);
}

public GameList GetAllUserGames(int userId)
{
    Bl bl = new Bl();
    return bl.BlGetAllUserGames(userId);
}

public UserList GetAllUserFriends(int userId)
{
    Bl bl = new Bl();
    return bl.BlGetAllUserFriends(userId);
}

public GameList GetMutualGames(int u1, int u2)
{
    Bl bl = new Bl();
    return bl.BlGetMutualGames(u1, u2);
}

public bool AreFriends(int user1Id, int user2Id)
{
    Bl bl = new Bl();
    return bl.BlAreFriends(user1Id, user2Id);
}

public void MakeFriends(User u1, User u2)
{
    Bl bl = new Bl();
    bl.BlMakeFriends(u1, u2);
}

public void RemoveFriend(User u1, User u2)
{
    Bl bl = new Bl();
    bl.BlRemoveFriends(u1, u2);
}

public Game StartGame(Player p, int playerCount)
{
    Bl bl = new Bl();
    Game g = bl.BlStartGame(p, playerCount);

    return g;
}

public int GetPlayersFound(int playerCount)
{
    Bl bl = new Bl();
    return bl.BlGetPlayersFound(playerCount);
}

public bool StopSearchingForGame(Player p)
{
    Bl bl = new Bl();
    return bl.BlStopSearchingForGame(p);
}

public bool PlayerQuit(Player p)
{
    Bl bl = new Bl();
    return bl.BlPlayerQuit(p);
}

public PlayerList GetPlayerList()
{
    return null;
}

public void AddAction(Message m)
{
```

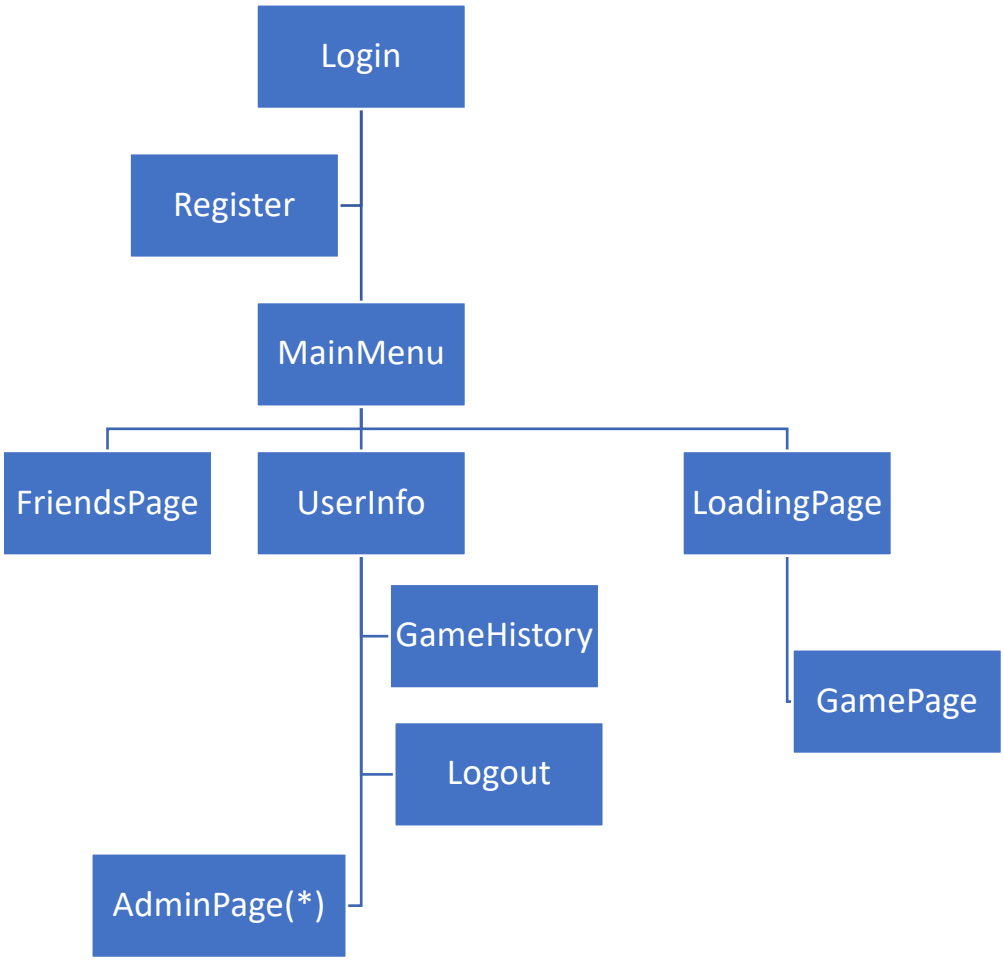
```
            PendingChanges.Add(m);
        }

        public void AddActions(MessageList ml)
        {
            PendingChanges.AddRange(ml);
        }

        public int SaveChanges()
        {
            Bl bl = new Bl();
            return bl.SaveChanges();
        }

        public MessageList DoAction(int gameId, int playerId)
        {
            MessageList temp = new MessageList();

            if (PendingChanges == null || PendingChanges.Count == 0) return null;
            else
            {
                foreach (Message m in PendingChanges.ToList())
                {
                    if (m.GameId == gameId && m.Reciever == playerId)
                    {
                        temp.Add(m);
                        PendingChanges.Remove(m);

                        if (m.Reciever == m.Target.Id) // make sure this happens once for each message
                        {
                            switch (m.Action)
                            {
                                case Model.Action.Add:
                                    Bl bl = new Bl();
                                    bl.BlAddCard(m);
                                    break;

                                case Model.Action.Remove:
                                    bl = new Bl();
                                    bl.BlRemoveCard(m);
                                    break;

                                case Model.Action.SwitchHand:
                                    bl = new Bl();
                                    bl.BlSwitchHands(m);
                                    break;

                                case Model.Action.Win:
                                    bl = new Bl();
                                    bl.BlWin(m);
                                    break;

                                case Model.Action.Loss:
                                    bl = new Bl();
                                    bl.BlLoss(m);
                                    break;
                            }
                        }
                    }
                }
            }

            return temp;
        }
    }
}
```

המתודה DoAction למעשה מתרגמת את העצמים מסוג Message המתבצרים מבקשות הלקוחות, לפעולה.

מתודה זו בדומה למתודת SaveChnages מאפשרת פעילות א-סינכרונית.

הלקוח - סביבת WPF

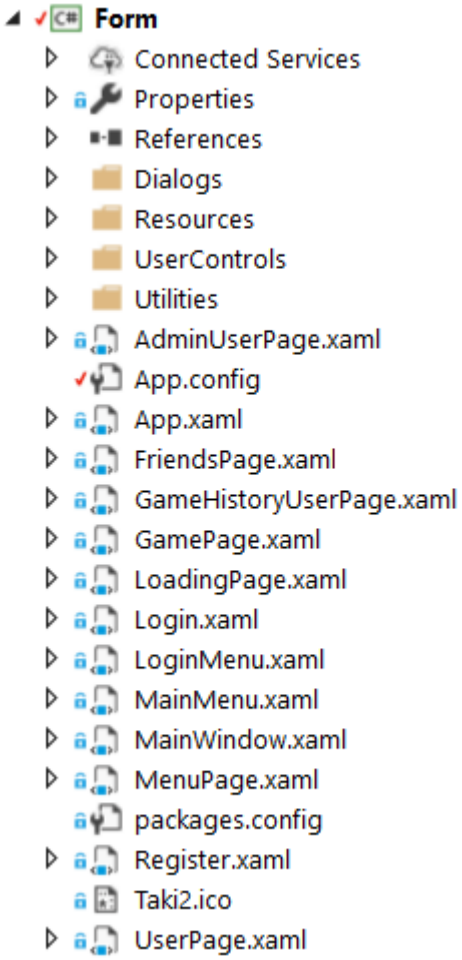
תרשים זרימת חלונות



הגדרת השרת – **נמצא בקובץ** (App.config)

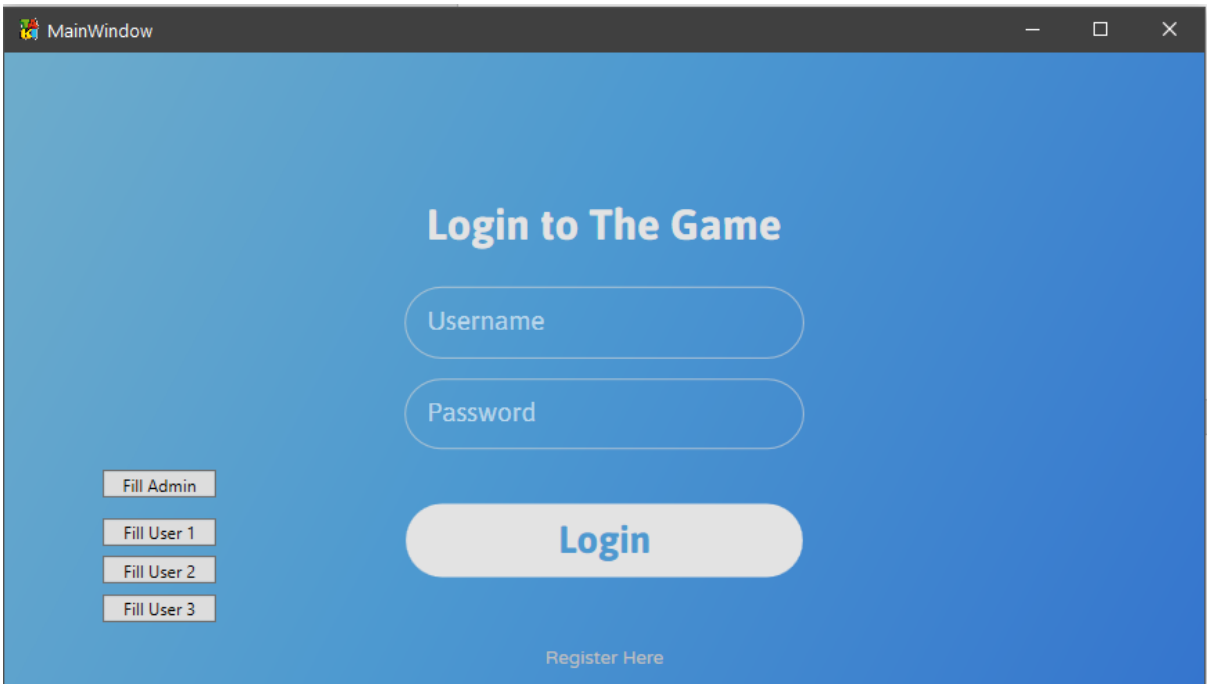
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />
    </startup>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_IService" />
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint
address="http://192.168.0.18:8733/Design_Time_Addresses/Service/Service/"
                binding="basicHttpBinding"
bindingConfiguration="BasicHttpBinding_IService"
                contract="TakiService.IService" name="BasicHttpBinding_IService" />
        </client>
    </system.serviceModel>
</configuration>
```

מבנה התיקייה



קוד לדוגמה

דף Login



XAML:

```
<Page x:Class="Form.Login"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:Form"
      xmlns:utilities="clr-namespace:Form.Utilities"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="Login">
    <Page.Resources>

        <ControlTemplate x:Key="TextBoxBaseControlTemplate" TargetType="{x:Type TextBoxBase}">
            <Border Background="{TemplateBinding Background}"
                    x:Name="Bd"
                    BorderThickness="1" BorderBrush="#FFD7D7D7" CornerRadius="24">
                <ScrollViewer x:Name="PART_ContentHost"/>
            </Border>
        </ControlTemplate>

        <ControlTemplate x:Key="PasswordBoxControlTemplate" TargetType="{x:Type PasswordBox}">
            <Border Background="{TemplateBinding Background}"
                    x:Name="Bd"
                    BorderThickness="1" BorderBrush="#FFD7D7D7" CornerRadius="24">
                <ScrollViewer x:Name="PART_ContentHost"/>
            </Border>
        </ControlTemplate>

        <ControlTemplate x:Key="ButtonBaseControlTemplate" TargetType="{x:Type ButtonBase}">
            <Grid x:Name="grid">
                <Border Background="{TemplateBinding Background}"
                    x:Name="Bd"
```

```
                    BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="24">
                </Border>
            </Grid>
        </ControlTemplate>

        <Style TargetType="TextBlock">
            <Setter Property="HorizontalAlignment" Value="Center" />
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontFamily" Value="/Form;component/Resources/Fonts/Asap/#Asap"/>
        </Style>

        <Style TargetType="TextBox">
            <Setter Property="FontFamily" Value="/Form;component/Resources/Fonts/Asap/#Asap"/>
            <Setter Property="FontWeight" Value="SemiBold"/>
            <Setter Property="VerticalContentAlignment" Value="Center" />
            <Setter Property="FontSize" Value="17"/>
            <Setter Property="Padding" Value="15,7,0,0"/>
            <Setter Property="Foreground"  Value="White"/>
            <Setter Property="Opacity" Value="0.6"/>
        </Style>
        <Style TargetType="PasswordBox">
            <Setter Property="FontFamily" Value="/Form;component/Resources/Fonts/Asap/#Asap"/>
            <Setter Property="VerticalContentAlignment" Value="Center" />
            <Setter Property="FontSize" Value="17"/>
            <Setter Property="Padding" Value="15,7,0,0"/>
            <Setter Property="Foreground"  Value="white"/>
            <Setter Property="Opacity"  Value="0.6"/>
        </Style>

    </Page.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="131*"/>
            <RowDefinition Height="24*"/>
            <RowDefinition Height="48*"/>
            <RowDefinition Height="13*"/>
            <RowDefinition Height="47*"/>
            <RowDefinition Height="35*"/>

            <RowDefinition Height="121*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="5*"/>
            <ColumnDefinition Width="5*"/>
            <ColumnDefinition Width="5*"/>
        </Grid.ColumnDefinitions>
        <TextBlock Text="Login to The Game" FontSize="28" Grid.Row="0"
HorizontalAlignment="Center" Height="32" Foreground="#FFE3E3E3"
FontFamily="/Form;component/Resources/Fonts/Asap/#Asap" Margin="266,0,266.698,0.66"
FontWeight="Bold" VerticalAlignment="Bottom" Grid.ColumnSpan="3"/>

        <AdornerDecorator Grid.Row="2" Grid.Column="1">
            <TextBox Template="{StaticResource TextBoxBaseControlTemplate}" x:Name="Username"
Background="{x:Null}" Padding="15,-2,0,0">
                <utilities:WatermarkService.Watermark>
                    <TextBlock FontSize="17" VerticalAlignment="Center"
FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Foreground="White">Username</TextBlock>
                </utilities:WatermarkService.Watermark>
            </TextBox>
        </AdornerDecorator>

        <AdornerDecorator Grid.Row="4" Grid.Column="1">
            <PasswordBox Template="{StaticResource PasswordBoxControlTemplate}"
x:Name="Password" Background="{x:Null}" Padding="15,-2,0,0">
                <utilities:WatermarkService.Watermark>
```

```
                    <TextBlock FontSize="17"  VerticalAlignment="Center"
FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Foreground="White">Password</TextBlock>
                </utilities:WatermarkService.Watermark>
            </PasswordBox>
        </AdornerDecorator>


        <Button Template="{StaticResource ButtonBaseControlTemplate}" x:Name="LoginButton"
Grid.Row="6" Background="#FFE3E3E3" Height="50" Click="LoginButton_Click" Grid.Column="1"
VerticalAlignment="Top"/>

        <TextBlock FontFamily="/Form;component/Resources/Fonts/Asap/#Asap" FontWeight="Bold"
Grid.Row="6" HorizontalAlignment="Center" Foreground="#FF4D99D0" Height="30" Text="Login"
Grid.Column="1" VerticalAlignment="Top" TextAlignment="Center" Margin="0,10,0,0" FontSize="25"
/>

        <Button Grid.Row="6" Grid.Column="1" Width="120" Height="15" Padding="-10"
VerticalAlignment="Bottom" Content="Register Here" Background="{x:Null}"
Click="RegisterButton_Click" BorderThickness="0"  Foreground="#FFC2C0C0" FontSize="12"
FontFamily="/Form;component/Resources/Fonts/Varela_Round/#Varela Round" Margin="0,0,0,10" />

        <TextBlock x:Name="noUserError" Grid.Row="5" Foreground="#FFC25252"
HorizontalAlignment="Stretch" FontSize="12" TextWrapping="Wrap" Grid.Column="2"/>

        <Button Content="Fill Admin" HorizontalAlignment="Left" Margin="64,13.585,0,0"
Grid.Row="5" VerticalAlignment="Top" Width="75" Click="AdminFillButton_Click" Height="18"/>
        <Button Content="Fill User 1" HorizontalAlignment="Left" Margin="64,10.245,0,0"
Grid.Row="6" VerticalAlignment="Top" Width="75" Click="User1FillButton_Click" Height="18"/>
        <Button Content="Fill User 2" HorizontalAlignment="Left" Margin="64,35.245,0,0"
Grid.Row="6" VerticalAlignment="Top" Width="75" Click="User2FillButton_Click" Height="18"/>
        <Button Content="Fill User 3" HorizontalAlignment="Left" Margin="64,60.245,0,0"
Grid.Row="6" VerticalAlignment="Top" Width="75" Click="User3FillButton_Click" Height="18"/>
    </Grid>
</Page>
```

C# Code:

```
namespace Form
{
    public partial class Login : Page
    {

        public Login()
        {
            InitializeComponent();
        }

        private void RegisterButton_Click(object sender, RoutedEventArgs e)
        {
            LoginMenu.LoginFrame.Navigate(new Register());
        }

        private void LoginButton_Click(object sender, RoutedEventArgs e)
        {
            string usernameValue = Username.Text;
            string passwordValue = Password.Password;

            List<Control> textBoxes = new List<Control>();
            textBoxes.Add(Username);
            textBoxes.Add(Password);
```

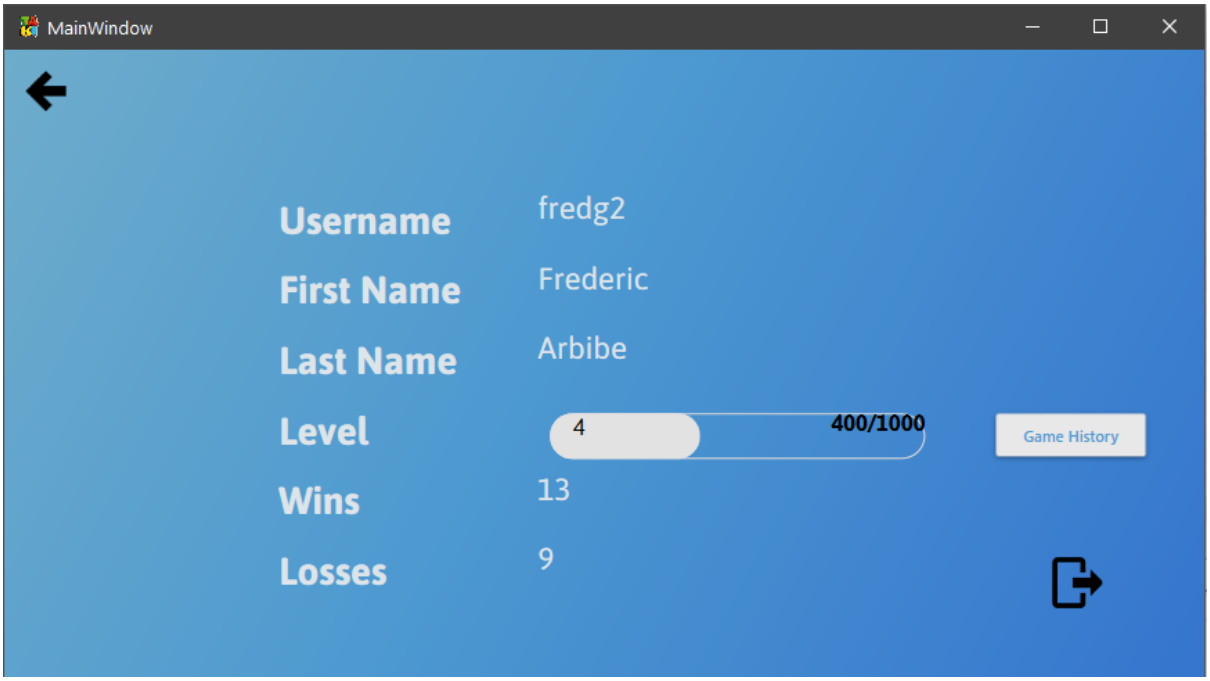
```
            if (Check.NullCheck(textBoxes))
            {
                User currentUserCheck = MainWindow.Service.Login(usernameValue, passwordValue);

                if (currentUserCheck != null)
                {
                    noUserError.Text = "";
                    MainWindow.CurrentUser = currentUserCheck;
                    MainWindow.BigFrame.Navigate(new MainMenu());
                }
                else
                {
                    noUserError.Text =
                    "Your Password or Username are incorrect. Try again";
                }
            }
            else
            {
                noUserError.Text = "Please fill up all the fields!";
            }
        }


        private void AdminFillButton_Click(object sender, RoutedEventArgs e)
        {
            Username.Text = "Samuelov1";
            Password.Password = "123";

            LoginButton_Click(null, null);
        }


        private void User1FillButton_Click(object sender, RoutedEventArgs e)
        {
            Username.Text = "fredg2";
            Password.Password = "Israel123";

            LoginButton_Click(null, null);
        }


        private void User2FillButton_Click(object sender, RoutedEventArgs e)
        {
            Username.Text = "itai";
            Password.Password = "menes";

            LoginButton_Click(null, null);
        }

        private void User3FillButton_Click(object sender, RoutedEventArgs e)
        {
            Username.Text = "el_primo";
            Password.Password = "123456";

            LoginButton_Click(null, null);
        }
    }
}
```

בקריאה Service.Login מתממשת הקישוריות בין הלקוח לשרת בפעם הראשונה.
הלקוח מבקש להיכנס עם שם המשתמש והסיסמה האלה, אם הם נכונים יוחזר ללקוח עצם מסוג משתמש המייצג אותו, ואם לא – יחזיר null, וכך נדע שהסיסמה/ שם המשתמש אינם נכונים.

AdminButton

דף UserPage

דף זה הוא דף המידע אודות המשתמש.
דף זה מכיל מידע כגון: השם הפרטי, שם המשפחה ושם המשתמש, רמת המשתמש במשחק וניקודו הכללי, מספר הניצחונות שלו ומספר ההפסדים שלו.
מדף זה ניתן להגיע אל דף היסטוריית המשחקים, לצאת מהמשתמש (Logout), ובמקרה והמשתמש גם מנהל ישנה האופציה להגיע אל מסך המנהלים באמצעות לחיצה על הכפתור הבא אשר יופיע בהתאם:



Xaml:

```
<Page x:Class="Form.UserPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:Form"
      xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="UserPage">
    <Page.Resources>
        <ControlTemplate x:Key="ButtonBaseControlTemplate" TargetType="{x:Type
ButtonBase}">
            <Grid x:Name="grid">
                <Border Background="{TemplateBinding Background}"
                    x:Name="Bd"
                    BorderThickness="{TemplateBinding BorderThickness}"
CornerRadius="24">
                </Border>
            </Grid>
        </ControlTemplate>
    </Page.Resources>
```

```
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="2*"/>
            <ColumnDefinition Width="2*"/>
            <ColumnDefinition Width="3*"/>
            <ColumnDefinition Width="2*"/>
        </Grid.ColumnDefinitions>

        <Label Grid.Column="1" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
FontWeight="Bold" Grid.Row="2" FontSize="25" Content="Username" Foreground="#DDECECEC"/>
        <Label Grid.Column="1" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
FontWeight="Bold" Grid.Row="3"  FontSize="25" Content="First Name"
Foreground="#DDECECEC"/>
        <Label Grid.Column="1" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
FontWeight="Bold" Grid.Row="4"  FontSize="25" Content="Last Name"
Foreground="#DDECECEC"/>
        <Label Grid.Column="1" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
FontWeight="Bold" Grid.Row="5"  FontSize="25" Content="Level" Foreground="#DDECECEC"/>
        <Label Grid.Column="1" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
FontWeight="Bold" Grid.Row="6"  FontSize="25" Content="Wins" Foreground="#DDECECEC"/>
        <Label Grid.Column="1" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
FontWeight="Bold" Grid.Row="7"  FontSize="25" Content="Losses" Foreground="#DDECECEC"/>

        <TextBlock FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Grid.Column="2" Grid.Row="2" FontSize="20" TextWrapping="Wrap" Text="{Binding
Path=Username}" Foreground="#DDECECEC"/>
        <TextBlock FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Grid.Column="2" Grid.Row="3" FontSize="20" TextWrapping="Wrap" Text="{Binding
Path=FirstName}" Foreground="#DDECECEC"/>
        <TextBlock FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Grid.Column="2" Grid.Row="4" FontSize="20" TextWrapping="Wrap" Text="{Binding
Path=LastName}" Foreground="#DDECECEC"/>
        <ProgressBar x:Name="LevelProgressBar" Grid.Column="2" Grid.Row="5" Minimum="0"
Maximum="1000" Margin="8"/>
        <TextBlock FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Grid.Column="2" Grid.Row="5" FontSize="15" TextWrapping="Wrap" Text="{Binding
Path=Level}" Padding="8" Margin="15,0,0,0"/>
        <TextBlock FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
x:Name="ShowScore" Grid.Column="2" Grid.Row="5" FontSize="15" Height="Auto"
FontWeight="Bold" VerticalAlignment="Bottom" HorizontalAlignment="right" Padding="8"
Margin="0,0,0,15"/>
        <TextBlock FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Grid.Column="2" Grid.Row="6" FontSize="20" TextWrapping="Wrap" Text="{Binding Path=Wins}"
Foreground="#DDECECEC"/>
        <TextBlock FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Grid.Column="2" Grid.Row="7" FontSize="20" TextWrapping="Wrap" Text="{Binding
Path=Losses}" Foreground="#DDECECEC"/>

        <Button Grid.Column="0" Grid.Row="0" x:Name="BackButton"
BorderBrush="Transparent" Click="BackButton_Click" Background="Transparent"
                Margin="5,5,0,0" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
Height="40"
                VerticalAlignment="Top" HorizontalAlignment="Left" Width="40">
```

```
            <materialDesign:PackIcon Kind="ArrowLeftThick" Width="40" Height="40" />
        </Button>

        <Button Grid.Column="3" Grid.Row="7" Click="Logout_Button_Click"
BorderBrush="Transparent"  Background="Transparent"
                Margin="5,5,0,0" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
                VerticalAlignment="Top" HorizontalAlignment="Center" Width="40"
ToolTip="Logout">
            <materialDesign:PackIcon  Kind="Logout" Width="40" Height="40" />
        </Button>

        <Button
            x:Name="GamesButton"
            Content="Game History"
            Grid.Column="3" Grid.Row="5"
            Click="Game_Button_Click"
            Foreground="#FF4D99D0"
            Background="#FFE8E8E8"
            Style="{StaticResource MaterialDesignRaisedLightButton}"
            Height="30"
            Width="100"
            FontSize="10"
            ToolTip="View Your Game History" BorderBrush="{x:Null}">
        </Button>
        <Button
            x:Name="AdminButton" Content="AdminButton" Grid.Column="3" Grid.Row="6"
Click="Admin_Button_Click"
            Foreground="#FF4D99D0"
            Background="#FFE8E8E8"
            Style="{StaticResource MaterialDesignRaisedLightButton}"
            Height="30"
            Width="100"
            FontSize="10"
            BorderBrush="{x:Null}">
        </Button>

    </Grid>
</Page>
```

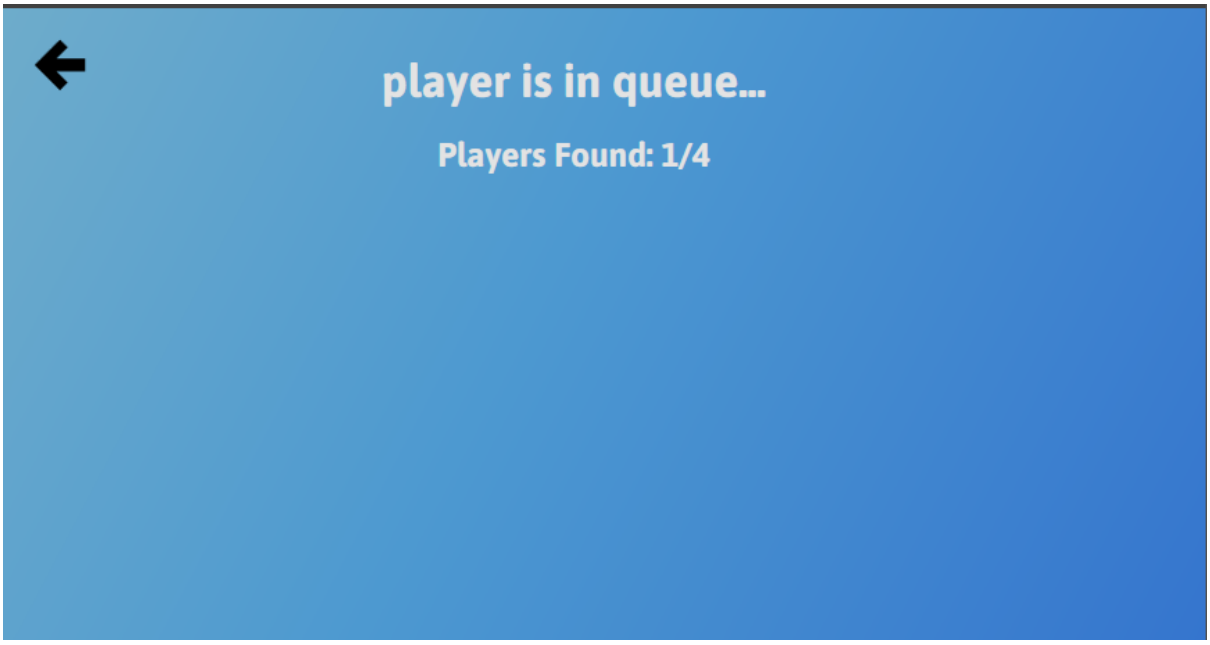
C# Code:

```
namespace Form
{
    /// <summary>
    /// Inter_action logic for UserPage.xaml
    /// </summary>
    public partial class UserPage : Page
    {
        private User _cu = MainWindow.CurrentUser;

        public UserPage()
        {
            InitializeComponent();

            DataContext = _cu;

            LevelProgressBar.Value = _cu.Score % 1000;
            ShowScore.Text = (_cu.Score % 1000) + "/1000";

            if (_cu.Admin)
            {
                AdminButton.Visibility = Visibility;
            }
            else
            {
                AdminButton.Visibility = Visibility.Hidden;
            }
        }


        private void BackButton_Click(object sender, RoutedEventArgs e)
        {
            if (NavigationService.CanGoBack)
            {
                NavigationService.GoBack();
            }
        }


        private void Logout_Button_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.CurrentUser = null;
            MainWindow.Service.Logout(_cu.Id);
            MainWindow.CurrentUser = null;
            MainWindow.BigFrame.Navigate(new LoginMenu());
        }


        private void Admin_Button_Click(object sender, RoutedEventArgs e)
        {
            MainMenu.MenuFrame.Navigate(new AdminUserPage());
        }


        private void Game_Button_Click(object sender, RoutedEventArgs e)
        {
            MainMenu.MenuFrame.Navigate (new
GameHistoryUserPage(MainWindow.CurrentUser.Id));
        }
    }
}
```

דף LoadingPage

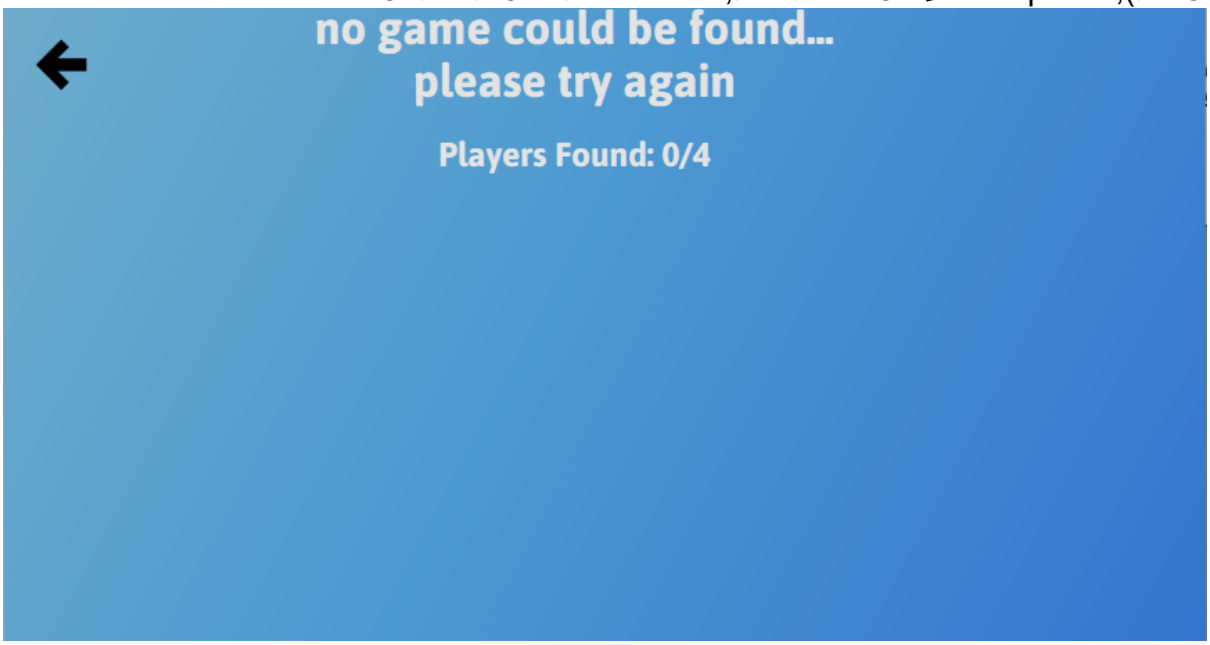
הלקוח מגיע אל דף לאחר שבחר את מספר השחקנים איתם הוא רוצה לשחק. לאחר שבחר עם כמה שחקנים הוא רוצה לשחק, דף זה אחראי לחפש עבורו מספר שחקנים שרוצים לשחק עם אותו מספר של שחקנים, וליצור מהם משחק.

דף זה הוא למעשה דף המעבר אל שולחן המשחק עצמו.



כאשר השרת מחפש עבור השחקן משחק, הוא מעדכן אותו בזמן אמת על מספר השחקנים שהוא כבר מצא.

במקרה והשרת לא הצליח למצוא משחק מתאים לאותו משתמש לאחר זמן מסויים (במקרה זה 10 שניות), הוא יקבל הודעה שמציינת זאת, ומזמינה אותו לנסות אותו שוב.



XAML:

```
<Page x:Class="Form.LoadingPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:Form"
      xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="LoadingPage">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="73*"/>
            <ColumnDefinition Width="93*"/>
            <ColumnDefinition Width="185*"/>
            <ColumnDefinition Width="59*"/>
            <ColumnDefinition Width="182*"/>
            <ColumnDefinition Width="207*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="70*"/>
            <RowDefinition Height="48*"/>
            <RowDefinition Height="146*"/>
            <RowDefinition Height="57*"/>
            <RowDefinition Height="39*"/>
            <RowDefinition Height="57*"/>
            <RowDefinition Height="34*"/>
        </Grid.RowDefinitions>
        <TextBlock x:Name="status" FontSize="30" TextWrapping="Wrap" Text="Searching for
a Game..." FontFamily="/Form;component/Resources/Fonts/Asap/#Asap" TextAlignment="Center"
Grid.ColumnSpan="3" Foreground="#FFE3E3E3" Grid.Column="2" FontWeight="Bold"
VerticalAlignment="Bottom"/>
        <Button Visibility="Hidden"  x:Name="SearchAgainButton" Content="Search Again"
Click="SearchAgainButton_Click" Grid.Row="5" Background="{x:Null}" BorderBrush="White"
Foreground="DarkGray" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
FontSize="16" Grid.ColumnSpan="2" Margin="4.309,0.436,59.055,0.291" Grid.Column="2" />
        <Button x:Name="CancelSearchButton" BorderBrush="Transparent"
Click="CancelSearchButton_Click" Background="Transparent"  Margin="10,10,0,0"
FontFamily="/Form;component/Resources/Fonts/Asap/#Asap" Height="50"
VerticalAlignment="Top" HorizontalAlignment="Left" Width="50" >
            <materialDesign:PackIcon Kind="ArrowLeftThick" Width="50" Height="50"/>
        </Button>
        <TextBlock x:Name="PlayersFound" FontSize="22" TextWrapping="Wrap" Text="Players
Found: 5/5" FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"
TextAlignment="Center" Grid.ColumnSpan="3" Foreground="#FFE3E3E3" Grid.Row="1"
Grid.Column="2" FontWeight="Bold" VerticalAlignment="Bottom"/>
    </Grid>
</Page>
```

C# code:

```
namespace Form
{
    /// <summary>
    /// Inter_action logic for LoadingPage.xaml
    /// </summary>
    public partial class LoadingPage : Page
    {
        private Game _game;
        private int _playerCount;
        private Player _p;
        private User _cu;
        private int _counter;
        private bool _gameNotFound;
        private System.Windows.Threading.DispatcherTimer _dispatcherTimer;


        public LoadingPage(int playerCount)
        {
            _counter = 0;
            _playerCount = playerCount;
            _cu = MainWindow.CurrentUser;
            _gameNotFound = false;

            InitializeComponent();

            SearchGame();
        }

        private void SearchGame()
        {
            _counter = 0;
            _p = new Player()
            {
                UserId = _cu.Id,
                FirstName = _cu.FirstName,
                LastName = _cu.LastName,
                Username = _cu.Username,
                Password = _cu.Password,
                Level = _cu.Level,
                ProfileImage = _cu.ProfileImage,
                Score = _cu.Score,
                Admin = _cu.Admin,
                Wins = _cu.Wins,
                Losses = _cu.Losses,
                TempScore = 0
            };

            //_game = MainWindow.Service.StartGame(_p, _playerCount);

            _dispatcherTimer = new System.Windows.Threading.DispatcherTimer();
            _dispatcherTimer.Tick += FindGame;
            _dispatcherTimer.Interval = new TimeSpan(0, 0, 0, 0, 500); //send request
five times every second
            _dispatcherTimer.Start();
```

עם הבקשה למצוא משחק מתאים, יוצרת המערכת עצם מסוג שחקן, שהוא עצם זמני אשר ייצג את המשתמש באותו המשחק.

הלקוח למעשה שולח בקשת תמונת מצב מהשרת בכל זמן קצוב, במקרה זה בכל חצי שניה.
אם השרת מעדכן שטרם נמצא משחק, הלקוח מנסה פעם נוספת.

```
            void FindGame(object sender, EventArgs e)
            {
                if (_game == null)
                {
                    status.Text = "player is in queue...";
                    PlayersFound.Text = "Players Found: 0/" + _playerCount;
```

לולאת חיפוש זו יכולה לקרות 20 פעם, כאשר יש הפסקה של חצי שניה בין כל שליחת בקשה לשרת. כך למעשה, החיפוש מתמשך במשך 10 שניות, ואם בסופו טרם נמצא משחק תוחזר הועדה למשתמש שתציין זאת.

```
                    if (_counter < 20)
                    {
                        _game = MainWindow.Service.StartGame(_p, _playerCount);
                        _counter++;
                        PlayersFound.Text = "Players Found: "+
MainWindow.Service.GetPlayersFound(_playerCount) + "/" + _playerCount;
                    }
                    else
                    {
                        _dispatcherTimer.Stop();
                        status.Text = "no game could be found... please try again";
                        _gameNotFound = true;
                    }
                }
                else // if game is found
                {
                    MainWindow.BigFrame.Navigate(new GamePage(_game));
                    _dispatcherTimer.Stop();// stop timer loop
                }
            }
        }


        private void SearchAgainButton_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.BigFrame.Navigate(new LoadingPage(_playerCount));
        }


        private void CancelSearchButton_Click(object sender, RoutedEventArgs e)
        {
            _dispatcherTimer.Stop();
            MainWindow.Service.StopSearchingForGame(_p);
            MainWindow.BigFrame.Navigate(_gameNotFound ? new MainMenu(true) : new
MainMenu());
        }
    }
}
```

דף GamePage

זהו הדף בו מתרחש המשחק עצמו.
לאחר שנמצא משחק בדף הקודם LoadingPage, והמשתמש קיבל עצם משחק הכולל עצמים מסוג שחקנים, שהוא אחד מהם, מתחיל המשחק.

על מנת למנוע עומס על השרת, ומכיוון שלוגיקת המשחק פשוטה יחסית, החלטתי להציב את לוגיקת המשחק בצד בלקוח, בדף זה.

כל מהלך שמנסה השחקן לעשות נבדק על ידי האלגוריתם בדף זה, ובמקרה והמהלך תקין הוא מועבר אל השרת בצורה של Message, כלומר עצם מסוג הודעה המכיל את בקשת המהלך של אותו השחקן.

כאשר ההודעה הזו מגיעה אל השרת, השרת עושה מספר דברים:

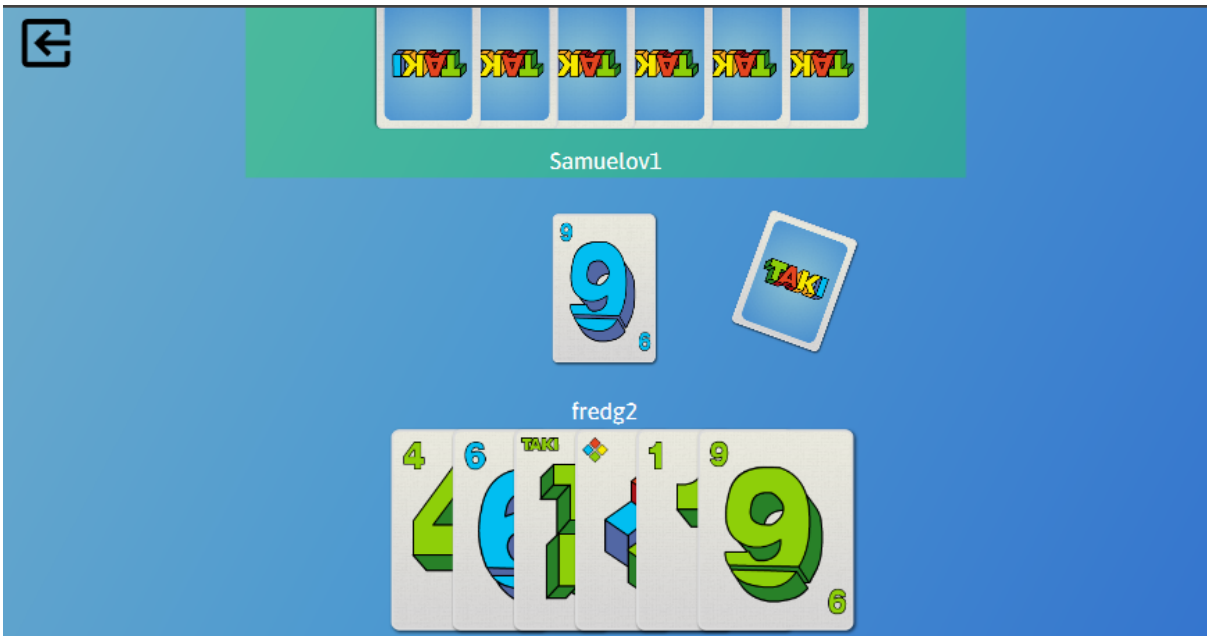
1. מבצע את המהלך במסד הנתונים, כלומר שומר את השלכות המהלך בטבלאות המתאימות.
כך למשל, כאשר שחקן יקח קלף מהחפיסה, הקשר בין השולחן לבין הקלף הזה בטבלת Player_Card_Table יוסר, ויתווסף קשר חדש מסוג זה בין השחקן לבין הקלף שלקח.

2. מעביר את אותה ההודעה אל כל השחקנים.

לאחר שכל השחקנים קיבלנו את אותה ההודעה, הם מבצעים את השינויים המתאימים בעצם המשחק שלהם.

למעשה, המידע אודות כל השחקנים במשחק והמשחק בכלל שמור אצל כל המשתמשים המשחקים, כאשר ההבדל היחידי בין כל שחקן הוא שכל אחד יכול לראות אך ורק את הקלפים שלו, ויכול ליצור אינטרקציה רק איתם.

מסך של שחקן 1



מסך של שחקן 2

XAML:

```
<Page x:Class="Form.GamePage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:Form"
      xmlns:userControls="clr-namespace:Form.UserControls"
      xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="GamePage">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
```

```
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
        </Grid.ColumnDefinitions>

        <Button x:Name="ExitGameButton" Click="ExitGameButton_Click"
Background="{x:Null}" BorderBrush="{x:Null}">
            <materialDesign:PackIcon Kind="ExitToApp" Width="Auto" Height="Auto"
RenderTransformOrigin="0.5,0.5" >
                <materialDesign:PackIcon.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform ScaleY="1" ScaleX="-1"/>
                        <SkewTransform AngleY="0" AngleX="0"/>
                        <RotateTransform Angle="0"/>
                        <TranslateTransform/>
                    </TransformGroup>
                </materialDesign:PackIcon.RenderTransform>
            </materialDesign:PackIcon>
        </Button>

        <userControls:Player1Uc x:Name="uc1" Grid.ColumnSpan="10" Grid.RowSpan="4"
Grid.Column="2" Grid.Row="5" Margin="42,23,43,0"/>
        <userControls:Player2Uc x:Name="uc2" Grid.Column="9" Grid.RowSpan="4"
RenderTransformOrigin="0.5,0.5" Grid.ColumnSpan="5" Grid.Row="2" Margin="1,48,-134,0">
            <userControls:Player2Uc.RenderTransform>
                <TransformGroup>
                    <ScaleTransform/>
                    <SkewTransform/>
                    <RotateTransform Angle="270"/>
                    <TranslateTransform/>
                </TransformGroup>
            </userControls:Player2Uc.RenderTransform>
        </userControls:Player2Uc>
        <userControls:Player3Uc x:Name="uc3" Grid.Column="2" Grid.RowSpan="3"
Grid.ColumnSpan="10" Margin="46,0,45,27" RenderTransformOrigin="0.5,0.5">
            <userControls:Player3Uc.RenderTransform>
                <TransformGroup>
                    <ScaleTransform/>
                    <SkewTransform/>
                    <RotateTransform Angle="180"/>
                    <TranslateTransform/>
                </TransformGroup>
            </userControls:Player3Uc.RenderTransform>
        </userControls:Player3Uc>
        <userControls:Player4Uc x:Name="uc4" Grid.Row="3" Grid.RowSpan="4"
Grid.ColumnSpan="5" RenderTransformOrigin="0.5,0.5" Margin="-108,18,15,36">
            <userControls:Player4Uc.RenderTransform>
                <TransformGroup>
                    <ScaleTransform/>
                    <SkewTransform/>
                    <RotateTransform Angle="90"/>
                    <TranslateTransform/>
                </TransformGroup>
            </userControls:Player4Uc.RenderTransform>
        </userControls:Player4Uc>
        <userControls:TableUc x:Name="uctable" Grid.ColumnSpan="8" Grid.RowSpan="4"
Grid.Column="3" Grid.Row="2" Margin="10"/>
    </Grid>
</Page>
```

C# Code:

```
namespace Form
{
    /// <summary>
    /// Inter_action logic for GamePage.xaml
    /// </summary>
    public partial class GamePage
    {

        private bool _myTurn;
        private PlayerList PlayersList { get; set; } = new PlayerList();
        private Player Table { get; set; }
        private Player CurrentPlayer { get; set; }
        private User CurrentUser { get; set; } = MainWindow.CurrentUser;
        private Game CurrentGame { get; set; }

        public bool MyTurn
        {
            get => _myTurn;
            set
            {
                _myTurn = value;

                if (MyTurn)
                {
                    uctable.CanTakeCardFromDeck();
                }
                else uctable.CannotTakeCardFromDeck();
            }
        }
        public bool Active { get; set; }
        public bool ClockWiseRotation { get; set; }
        public int Turn { get; set; }
        public int CurrentPlayerIndex { get; set; }
        public int PlayersCount { get; set; }
        public int PlusValue { get; set; }
        public Card OpenTaki { get; set; }
        public List<Card> Deck { get; set; }
        public MessageList LocalMessageList { get; set; }
        public BackgroundWorker BackgroundProgress { get; set; }

        public GamePage(Game game)
        {
            InitializeComponent();

            MainWindow.CurrentGamePage = this;

            CurrentGame = game;

            SetBackgroundWorker();

            ClockWiseRotation = true;
            uc1.SetAsNonActive();
            OpenTaki = null;
            MyTurn = false;
            Active = true;
            PlusValue = 0;

            int firstPlayerUserId = CurrentGame.Players[0].UserId;
            ReorderPlayerList();

            //the first player is the one to request changes saving in the database every x seconds
            if (CurrentUser.Id == firstPlayerUserId)
            {
                InitialTurn();// broadcast that self is the first player in the game's players list
            }

            uctable.TakeCardFromDeckButtonClicked += TakeCardFromDeck;

            uctable.PassCardToStackButtonClicked += PassCardToDeck;

            Deck = PlayersList[PlayersList.Count - 1].Hand.ToList();

            DataContext = PlayersList;
```

```
        uc1.SetCurrentPlayer(CurrentPlayer);

        switch (CurrentGame.Players.Count)
        {
            case 3:
                uc2.Visibility = Visibility.Hidden;
                uc3.SetCurrentPlayer(PlayersList[1]);
                uc4.Visibility = Visibility.Hidden;
                uctable.SetCurrentPlayer(Table);
                break;
            case 4:
                uc2.SetCurrentPlayer(PlayersList[1]);
                uc3.SetCurrentPlayer(PlayersList[2]);
                uc4.Visibility = Visibility.Hidden;
                uctable.SetCurrentPlayer(Table);
                break;
            case 5:
                uc2.SetCurrentPlayer(PlayersList[1]);
                uc3.SetCurrentPlayer(PlayersList[2]);
                uc4.SetCurrentPlayer(PlayersList[3]);
                uctable.SetCurrentPlayer(Table);
                break;
        }
    }

    private void SetBackgroundWorker()
    {
        BackgroundProgress = new BackgroundWorker();
        BackgroundProgress.DoWork += FetchChanges;
        BackgroundProgress.RunWorkerCompleted += BackgroundProcess_RunWorkerCompleted;

        BackgroundProgress.RunWorkerAsync();
    }

    private void FetchChanges(object sender, DoWorkEventArgs e)
    {
        Thread.Sleep(100);

        LocalMessageList = MainWindow.Service.DoAction(CurrentGame.Id, CurrentPlayer.Id);
    }


    private void BackgroundProcess_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
    {
        if (Active)
        {
            // Update local variables
            if (LocalMessageList != null && LocalMessageList.Count != 0)
            {
                foreach (Message m in LocalMessageList)
                {
                    switch (m.Action)
                    {
                        case TakiService.Action.Add:


                            PlayersList.Find(p => p.Id == m.Target.Id).Hand.Add(m.Card);
                            break;

                        case TakiService.Action.Remove:

                            // the target player
                            Player tempPlayer = PlayersList.Find(p => p.Id == m.Target.Id);
                            // the target card
                            Card tempCard = tempPlayer.Hand.Find(c => c.Id == m.Card.Id);
                            // remove the target card from the target player's hand
                            tempPlayer.Hand.Remove(tempCard);

                            break;

                        case TakiService.Action.Win:

                            // the winning player
                            Player winningPlayer = PlayersList.Find(p => p.Id == m.Target.Id);
```

```
            if (CurrentPlayer.Id == m.Target.Id)
            {
                PlayerWin();
                PlayerQuit();
                MainWindow.BigFrame.Navigate(new MainMenu());
            }
            else
            {
                PlayerWin pw = new PlayerWin(winningPlayer.Username);
                pw.ShowDialog();

                if (PlayersList.Count == 3)
                {
                    PlayerLoss();

                    MainWindow.Service.AddAction(new Message()
                    {
                        Action = TakiService.Action.Loss,
                        Target = CurrentPlayer,
                        Reciever = CurrentPlayer.Id,
                        GameId = CurrentGame.Id
                    });
                }
            }

            break;

        case TakiService.Action.SwitchHand:

            CardList l1 = PlayersList.Find(p => p.Id == m.Target.Id).Hand;

            PlayersList.Find(p => p.Id == m.Target.Id).Hand =
                                    PlayersList.Find(p => p.Id == m.Card.Id).Hand;
            PlayersList.Find(p => p.Id == m.Card.Id).Hand = l1;

            break;

        case TakiService.Action.PlusTwo:

            PlusValue = m.Card.Id;

            if (CurrentPlayer.Id == m.Target.Id && PlusValue != 0)
            {
                if (CurrentPlayer.Hand.Find(c => c.Value == Value.PlusTwo) == null)
                {
                    TakeMultipleCardsFromDeck(PlusValue);
                }
            }
            break;

        case TakiService.Action.NextTurn:

            Turn = PlayersList.FindIndex(p => p.Id == m.Target.Id);
            MyTurn = (m.Target.Id == CurrentPlayer.Id);

            Win();

            DeclareTurn();

            break;


        case TakiService.Action.SwitchRotation:

            if (ClockWiseRotation) ClockWiseRotation = false;
            else ClockWiseRotation = true;

            break;
```

```
                    case TakiService.Action.PlayerQuit:

                        int prevChangesSaverId = PlayersList[0].UserId;

                        Player quitter = PlayersList.Find(p => p.Id == m.Target.Id);

                        // remove the quitting player from the local players list
                        PlayersList.Remove(quitter);

                        PlayersList.TrimExcess();
                        CurrentGame.Players.TrimExcess();

                        if (CurrentUser.Id == PlayersList[0].UserId)
                        {
                            // broadcast that self is the first player in the game's players list
                            InitialTurn();

                            uc1.SetAsActive();
                        }
                        else
                        {
                            MyTurn = false;
                        }

                        DeclareTurn();

                        break;
                    }
                }

                UpdateUi();
            }

            // Run again
            // This will make the BgWorker run again, and never runs before it is completed.
            BackgroundProgress.RunWorkerAsync();
        }
    }


    public void PrintCards()
    {
        string cards = "\n \n -------------------------------------";
        foreach (Player p in PlayersList)
        {
            cards += "\n \n Player " + p.Username + ":";
            foreach (Card c in p.Hand)
            {
                cards += "\n value:" + c.Value + ", color:" + c.Color;
            }
        }
        Console.Write(cards);
    }



    // this function re-arranges the player in a particular way, to make sure that:
    // - list[First] is the current player
    // - all the players after him are arranged in order
    // - list[Last] is the table

    private void ReorderPlayerList()
    {
        PlayersCount = CurrentGame.Players.Count;

        CurrentPlayerIndex = CurrentGame.Players.FindIndex(p => p.UserId == CurrentUser.Id);

        for (int i = 0; i < PlayersCount - 1; i++)
        {
            if (CurrentPlayerIndex >= PlayersCount - 1)
            {
                PlayersList.Add(CurrentGame.Players[CurrentPlayerIndex % (PlayersCount - 1)]);
            }
            else
            {
```

```
            PlayersList.Add(CurrentGame.Players[CurrentPlayerIndex]);
        }

        CurrentPlayerIndex++;
    }

    PlayersList.Add(CurrentGame.Players.Find(q => q.Username == "table"));

    CurrentPlayer = PlayersList[0];

    Table = PlayersList[PlayersCount - 1];
}


private void ExitGameButton_Click(object sender, RoutedEventArgs e)
{
    ExitDialog dialog = new ExitDialog
    {
        Owner = Application.Current.MainWindow
    };

    if (dialog.ShowDialog() == true)
    {
        PlayerQuit();
        MainWindow.BigFrame.Navigate(new MainMenu());
    }
}


private void UpdateUi()
{
    uc1.UpdateUi(PlayersList[0]);

    switch (PlayersList.Count)
    {
        case 2:
            uc2.Visibility = Visibility.Hidden;
            uc3.Visibility = Visibility.Hidden;
            uc4.Visibility = Visibility.Hidden;
            uctable.UpdateUi(PlayersList[1]);

            ForceQuit dialog = new ForceQuit
            {
                Owner = Application.Current.MainWindow
            };

            if (dialog.ShowDialog() == true)
            {
                PlayerQuit();
                MainWindow.BigFrame.Navigate(new MainMenu());
            }
            break;

        case 3:
            uc2.Visibility = Visibility.Hidden;
            uc3.UpdateUi(PlayersList[1]);
            uc4.Visibility = Visibility.Hidden;
            uctable.UpdateUi(PlayersList[2]);
            break;

        case 4:
            uc2.UpdateUi(PlayersList[1]);
            uc3.UpdateUi(PlayersList[2]);
            uc4.Visibility = Visibility.Hidden;
            uctable.UpdateUi(PlayersList[3]);
            break;

        case 5:
            uc2.UpdateUi(PlayersList[1]);
            uc3.UpdateUi(PlayersList[2]);
            uc4.UpdateUi(PlayersList[3]);
            uctable.UpdateUi(PlayersList[4]);
            break;
    }
}
```

```
        private void PassCardToDeck(object sender, EventArgs e)
        {
            Player currentPlayer = PlayersList.First();
            Player table = PlayersList.Last();
            MessageList temp = new MessageList();
            Card givenCard = uc1.SelectedCard();

            if (givenCard != null)
            {
                if (givenCard.Value == Value.TakiAll || givenCard.Value == Value.Taki)
                {
                    OpenTaki = givenCard;
                }

                if (CheckPlay(givenCard, table.Hand[table.Hand.Count - 1]))
                {
                    for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the
table
                    {
                        if (givenCard.Value != Value.SwitchHandAll) // don't add "SwitchHandsAll" card to
table
                        {
                            temp.Add(new Message() // add the top card of the table to the current player
                            {
                                Action = TakiService.Action.Add,
                                Target = table, // the person who's hand is modified
                                Reciever = PlayersList[i].Id, // the peron who this message is for
                                Card = givenCard, // the card modified
                                GameId = CurrentGame.Id // the game modified
                            });

                            temp.Add(new Message() // add the top card of the table to the current player
                            {
                                Action = TakiService.Action.Remove,
                                Target = CurrentPlayer, // the person who's hand is modified
                                Reciever = PlayersList[i].Id, // the peron who this message is for
Card = givenCard, // the card modified
                                GameId = CurrentGame.Id // the game modified
                            });
                        }
                    }

                    MainWindow.Service.AddActions(temp);

                    Switch(givenCard);
                }
            }
        }


        private void TakeCardFromDeck(object sender, EventArgs e)
        {
            if (PlusValue != 0)
            {
                TakeMultipleCardsFromDeck(PlusValue);
            }
            else
            {
                Player currentPlayer = PlayersList.First();
                MessageList temp = new MessageList();
                Card takenCard = uctable.GetCardFromStack(); // get a random card

                for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the table
                {
                    temp.Add(new Message()// add the top card of the table to the current player
                    {
                        Action = TakiService.Action.Add,
                        Target = CurrentPlayer, // the person who's hand is modified
                        Reciever = PlayersList[i].Id, // the peron who this message is for
                        Card = takenCard, // the card modified
                        GameId = CurrentGame.Id // the game modified
                    });

                    temp.Add(new Message()// add the top card of the table to the current player
                    {
                        Action = TakiService.Action.Remove,
```

```
                    Target = Table, // the person who's hand is modified
                    Reciever = PlayersList[i].Id, // the peron who this message is for
                    Card = takenCard, // the card modified
                    GameId = CurrentGame.Id // the game modified
                });
            }

            MainWindow.Service.AddActions(temp);

            TurnFinished(Value.Nine); // give turn to next player
        }
    }


    private void TakeMultipleCardsFromDeck(int num)
    {
        Player currentPlayer = PlayersList.First();
        MessageList temp = new MessageList();

        for (int i = 0; i < num; i++)
        {
            for (int j = 0; j < PlayersList.Count; j++) //add for each player, not including the table
            {
                temp.Add(new Message()// add the top card of the table to the current player
                {
                    Action = TakiService.Action.Add,
                    Target = CurrentPlayer, // the person who's hand is modified
                    Reciever = PlayersList[j].Id, // the peron who this message is for
                    Card = uctable.GetCardFromStack(), // the card modified
                    GameId = CurrentGame.Id // the game modified
                });
            }
        }

        MainWindow.Service.AddActions(temp);

        PlusTwoMessage(0); // finished taking cards

        TurnFinished(Value.Nine); // give turn to next player
    }

    private int GetUserControlOfPlayer(int playerIndex)
    {
        if (uc1.CurrentPlayer != null && uc1.CurrentPlayer.Id == PlayersList[playerIndex].Id) return
1;
        if (uc2.CurrentPlayer != null && uc2.CurrentPlayer.Id == PlayersList[playerIndex].Id) return
2;
        if (uc3.CurrentPlayer != null && uc3.CurrentPlayer.Id == PlayersList[playerIndex].Id) return
3;
        return 4;
    }

    private void DeclareTurn() // declare the player with the turn as active
    {
        switch (GetUserControlOfPlayer(Turn))
        {
            case 1:
                uc1.SetAsActive();
                uc2.SetAsNonActive();
                uc3.SetAsNonActive();
                uc4.SetAsNonActive();
                break;

            case 2:
                uc1.SetAsNonActive();
                uc2.SetAsActive();
                uc3.SetAsNonActive();
                uc4.SetAsNonActive();
                break;

            case 3:
                uc1.SetAsNonActive();
                uc2.SetAsNonActive();
                uc3.SetAsActive();
                uc4.SetAsNonActive();
                break;
```

```
                case 4:
                    uc1.SetAsNonActive();
                    uc2.SetAsNonActive();
                    uc3.SetAsNonActive();
                    uc4.SetAsActive();
                    break;
            }
        }


        public void TurnFinished(Value value)
        {
            MessageList temp = new MessageList();

            for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the table
            {
                temp.Add(new Message()
                {
                    Action = TakiService.Action.NextTurn, // give next turn to
                    Target = PlayersList.Find(p => p.Id == GetNextPlayerId(value)),
                    Reciever = PlayersList[i].Id,
                    GameId = CurrentGame.Id
                });
            }

            MainWindow.Service.AddActions(temp);

            MyTurn = false;
        }


        private void InitialTurn()
        {
            uc1.SetAsActive();

            MessageList temp = new MessageList();

            for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the table
            {
                temp.Add(new Message()
                {
                    Action = TakiService.Action.NextTurn, // give next turn to self
                    Target = CurrentPlayer,
                    Reciever = PlayersList[i].Id,
                    GameId = CurrentGame.Id
                });
            }

            MainWindow.Service.AddActions(temp);

            MyTurn = true;
        }



        public void PlayerQuit()
        {
            Console.WriteLine("Player" + CurrentPlayer.Username + " removed from the game in the gameList:
" + MainWindow.Service.PlayerQuit(CurrentPlayer));

            MessageList temp = new MessageList();

            for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the table
            {
                temp.Add(new Message()
                {
                    Action = TakiService.Action.PlayerQuit,
                    Target = CurrentPlayer,
                    Reciever = PlayersList[i].Id,
                    GameId = CurrentGame.Id
                });
            }

            MainWindow.Service.AddActions(temp);
```

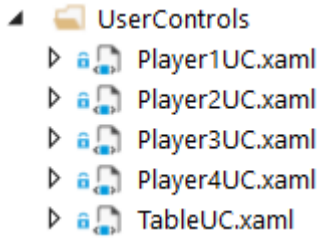
```
            if (MyTurn) TurnFinished(Value.Nine);
            Active = false;
        }


        private bool CheckPlay(Card given, Card table)
        {
            if (PlusValue != 0)
            {
                if (given.Value == table.Value) return true;
            }
            // not Multi-color
            if (given.Color == Color.Multi || table.Value == Value.TakiAll ||
                given.Color == table.Color || given.Value == table.Value) return true;

            return false;
        }



        private void Switch(Card givenCard)
        {
            if (OpenTaki != null)
            {
                int colorCount = CurrentPlayer.Hand.FindAll (c => c.Color == OpenTaki.Color || c.Value ==
Table.Hand[Table.Hand.Count-1].Value).Count - 1;

                if (givenCard.Value == Value.TakiAll)
                {
                    colorCount = 2;
                }

                if (OpenTaki.Value == Value.TakiAll && givenCard.Value != Value.TakiAll)
                {
                    switch (givenCard.Color)
                    {
                        case Color.Blue:
                            OpenTaki = new Card()
                            {
                                Id = 61,
                                Color = Color.Blue,
                                Image = "../Resources/Cards/card0061.png",
                                Special = true,
                                Value = Value.Taki
                            };
                            break;
                        case Color.Green:
                            OpenTaki = new Card()
                            {
                                Id = 13,
                                Color = Color.Green,
                                Image = "../Resources/Cards/card0013.png",
                                Special = true,
                                Value = Value.Taki
                            };
                            break;

                        case Color.Red:
                            OpenTaki = new Card()
                            {
                                Id = 29,
                                Color = Color.Red,
                                Image = "../Resources/Cards/card0029.png",
                                Special = true,
                                Value = Value.Taki
                            };
                            break;
                        case Color.Yellow:
                            OpenTaki = new Card()
                            {
                                Id = 45,
                                Color = Color.Yellow,
                                Image = "../Resources/Cards/card0045.png",
                                Special = true,
                                Value = Value.Taki
                            };
```

```
                    break;
                }
            }

            if (colorCount == 0)
            {
                OpenTaki = null;
                TurnFinished(Value.Nine);
            }
            else
            {
                if (colorCount == 1) // if one playing options is left in open taki
                {
                    OpenTaki = null;
                }

                TurnFinished(Value.Plus);
            }
        }
        else
        {
            switch (givenCard.Value)
            {
                case Value.SwitchColor:
                case Value.SwitchColorAll:

                    SwitchColor dialog = new SwitchColor
                    {
                        Owner = Application.Current.MainWindow
                    };

                    dialog.ShowDialog();

                    SwitchColorMessage(dialog.SelectedColor);

                    break;

                case Value.PlusTwo:

                    PlusTwoMessage(PlusValue + 2);

                    break;

                case Value.SwitchDirection:

                    ChangeRotationMessage();

                    break;

                case Value.SwitchHandAll:

                    SwitchHandsMessage();

                    break;
            }

            TurnFinished(givenCard.Value); // GetNextPlayerId will handle this
        }
    }


    private void SwitchHandsMessage()
    {
        MessageList temp = new MessageList();

        for (int i = 0; i < (PlayersList.Count - 1); i++) //add for each player, not including the
table
        {
            temp.Add(new Message() // add the top card of the table to the current player
            {
                Action = TakiService.Action.Remove,
                Target = CurrentPlayer, // the person who's hand is modified
                Reciever = PlayersList[i].Id, // the peron who this message is for
                Card = new Card() {Id = 67}, // the card modified
                GameId = CurrentGame.Id // the game modified
            });
```

```
                temp.Add(new Message()
                {
                    Action = TakiService.Action.SwitchHand,
                    Target = CurrentPlayer,
                    Card = new Card() { Id = GetNextPlayerId(Value.Nine) }, // pass the other Player's ID
through the card field.
                    Reciever = PlayersList[i].Id,
                    GameId = CurrentGame.Id
                });
            }
            MainWindow.Service.AddActions(temp);
        }


        private void SwitchColorMessage(Card selectedColorCard)
        {
            MessageList temp = new MessageList();

            for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the table
            {
                temp.Add(new Message()
                {
                    Action = TakiService.Action.Add,
                    Target = Table,
                    Reciever = PlayersList[i].Id,
                    GameId = CurrentGame.Id,
                    Card = selectedColorCard
                });
            }


            MainWindow.Service.AddActions(temp);
        }

        private void PlusTwoMessage(int num)
        {
            MessageList temp = new MessageList();

            for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the table
            {
                temp.Add(new Message()
                {
                    Action = TakiService.Action.PlusTwo,
                    Target = PlayersList.Find(p => p.Id == GetNextPlayerId(Value.Nine)), // get the next
player
                    Reciever = PlayersList[i].Id,
                    GameId = CurrentGame.Id,
                    Card = new Card() { Id = num }// the card's id represents the PlusValue Build-up
                });
            }

            MainWindow.Service.AddActions(temp);
        }




        private void ChangeRotationMessage()
        {
            MessageList temp = new MessageList();

            for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the table
            {
                temp.Add(new Message()
                {
                    Action = TakiService.Action.SwitchRotation,
                    Target = CurrentPlayer,
                    Reciever = PlayersList[i].Id,
                    GameId = CurrentGame.Id
                });
            }


            MainWindow.Service.AddActions(temp);
        }
```

```
        private int GetNextPlayerId(Value value)
        {
            // special card switch - case
            switch (value)
            {
                case Value.Stop:
                    if (PlayersList.Count > 3)
                    {
                        if (ClockWiseRotation) return PlayersList[PlayersList.Count - 3].Id;
                        return PlayersList[2].Id;
                    }
                    return CurrentPlayer.Id;

                case Value.Plus:
                case Value.Taki:
                case Value.TakiAll:
                    return CurrentPlayer.Id;

                case Value.SwitchDirection:
                    {
                        if (!ClockWiseRotation) return PlayersList[PlayersList.Count - 2].Id;
                        return PlayersList[1].Id;
                    }

            }

            if (ClockWiseRotation) return PlayersList[PlayersList.Count - 2].Id;
            return PlayersList[1].Id;

        }


        private void Win()
        {
            if (CurrentPlayer.Hand.Count == 0)
            {
                MessageList temp = new MessageList();

                for (int i = 0; i < PlayersList.Count; i++) //add for each player, not including the table
                {
                    temp.Add(new Message()
                    {
                        Action = TakiService.Action.Win,
                        Target = CurrentPlayer,
                        Reciever = PlayersList[i].Id,
                        GameId = CurrentGame.Id
                    });
                }
                MainWindow.Service.AddActions(temp);
            }
        }


        private void PlayerWin()
        {
            CurrentUser.Score += 500;
            CurrentUser.Wins += 1;
            CurrentUser.Level = (CurrentUser.Score - CurrentUser.Score % 1000) / 1000;
        }

        private void PlayerLoss()
        {
            CurrentUser.Score += 200;
            CurrentUser.Losses += 1;
            CurrentUser.Level = (CurrentUser.Score - CurrentUser.Score % 1000) / 1000;
        }
    }
}
```

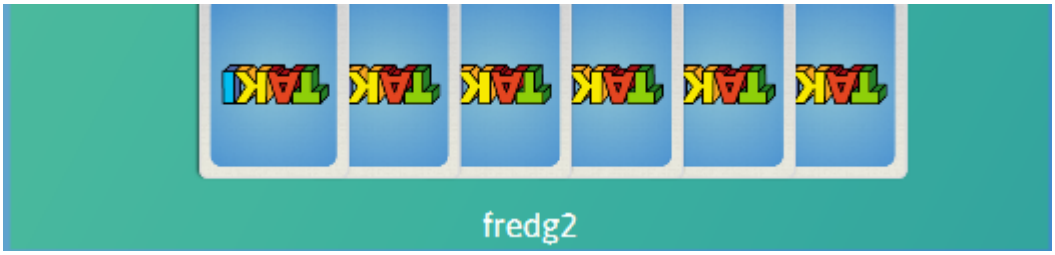
דף המשחק בנוי ממספר UserControl שהם רכיבים אותם ניתן לשכפל מספר פעמים על דף המשחק.



ישנם 5 רכיבים כאלו, האחראים על הצגת הקלפים של כל שחקן, וכמו כן גם את השולחן.

הראשון Player1UC הוא הרכיב היחיד מבין 4 הרכיבים המייצגים שחקנים במיקומים שונים בו ניתן לראות את הקלפים עצמם, ובו יופיע השחקן של המשתמש.

כך נראה רכיב Player1UC



Player3UC

TableUC

קוד לדוגמה

הרכיבים הללו מאוד דומים בקוד, אדגים את הקוד של Player1UC:

XAML:

```
<UserControl x:Class="Form.UserControls.Player1Uc"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Form"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">

    <Grid x:Name="MyGrid">

        <Rectangle Panel.ZIndex="-1" x:Name="BackgroundActive"/>

        <StackPanel>
            <Button FontSize="15" x:Name="Username" Click="Username_Click"
Background="Transparent" BorderBrush="{x:Null}">
                <TextBlock Text="{Binding Username}" HorizontalAlignment="Left"
VerticalAlignment="Top" Foreground="White"
FontFamily="/Form;component/Resources/Fonts/Asap/#Asap"></TextBlock>
            </Button>

            <ListView BorderThickness="0" x:Name="HandView" Background="Transparent"
HorizontalAlignment="Stretch">
                <ListView.ItemsPanel>
                    <ItemsPanelTemplate>
                        <StackPanel Margin="-40,0,0,0" Orientation="Horizontal"
HorizontalAlignment="Center" IsItemsHost="True" VerticalAlignment="Top"/>
                    </ItemsPanelTemplate>
                </ListView.ItemsPanel>

                <ListView.ItemTemplate>
                    <DataTemplate>
                        <Image Source="{Binding Image}" Height="135"
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackPanel>
    </Grid>
</UserControl>
```

ההבדל העיקרי בין רכיב Player1UC לשאר רכיבי השחקנים, הוא שרכיב זה מראה את התמונות של הקלפים אותם מחזיק השחקן, בעוד שבשאר הרכיבים מוצגת תמונה של גב הקלף לכל הקליפ בהם מחזיק השחקן.

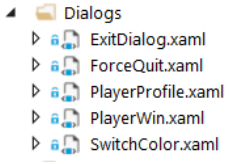
C# Code:

```
namespace Form.UserControls
{
    public partial class Player1Uc : UserControl
    {
        public Player1Uc() {

            InitializeComponent();
        }

        public Player CurrentPlayer { get; set; }

        public CardList Hand { get; set; }
```

```
        public Card SelectedCard()
        {
            return (Card) HandView?.SelectedItems[0];
        }

        public void UpdateUi(Player p)
        {
            CurrentPlayer = p;

            DataContext = CurrentPlayer;

            HandView.ItemsSource = null;

            HandView.ItemsSource = CurrentPlayer.Hand;
        }

        public void SetAsActive()
        {
            BackgroundActive.Fill = new
SolidColorBrush(System.Windows.Media.Color.FromArgb(90, 0, 250, 0));
        }

        public void SetAsNonActive()
        {
            BackgroundActive.Fill = null;
        }

        public void SetCurrentPlayer(Player currentPlayer)
        {

            CurrentPlayer = currentPlayer;

            Hand = currentPlayer.Hand;

            DataContext = CurrentPlayer;

            HandView.ItemsSource = CurrentPlayer.Hand;
        }
    }
}
```

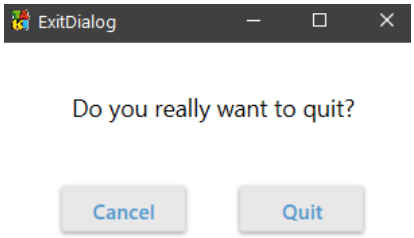
כאשר מתקיים שינוי כלשהו בהרכב הקלפים של שחקן, הרכיב מתעדכן מחדש, על מנת שיוצגו התמונות העדכניות של הקלפים המתאימים.

חלק חשוב נוסף מהדף GamePage הוא ה-Dialogs, האחראים על חלק מהאינטרקציות עם המשתמש.

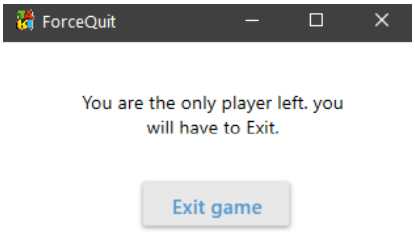


<u>בדף זה 4 דיאלוגים שונים:</u>

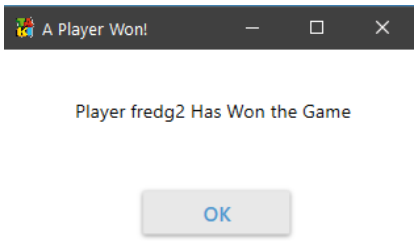
1. דיאלוג אישור היציאה – דיאלוג זה צץ כאשר שחקן מבקש לצאת מהמשחק. דיאלוג זה מאשר את כוונתו של השחקן



2. דיאלוג כפיית יציאה - דיאלוג זה צץ כאשר השחקן הוא האחרון שנותר במשחק, ולכן הוא מוכרח לצאת מהמשחק.

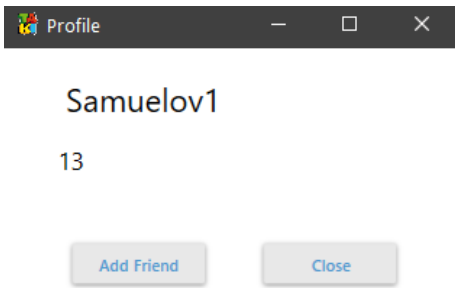


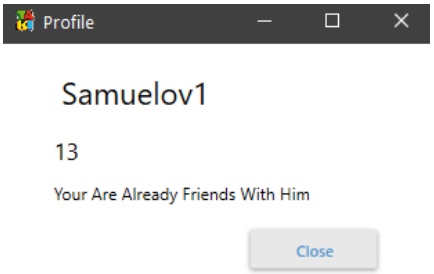
3. דיאלוג הכרזת ניצחון – דיאלוג זה צץ כאשר שחקן מנצח את המשחק.



4. דיאלוג בחירת צבע – דיאלוג זה מופיע כאשר שחקן שם קלף שנה צבע.

5. דיאלוג פרופיל – כאשר לוחצים על שמו של שחקן אחר, מופיע דיאלוג עם פרטים אודותיו, ואם הם כבר חברים יצויין כך, ואם לא תינתן האפשרות להוסיף את אותו השחקן כחבר.





דוגמה לקוד דיאלוג :

XAML:

```
<Window x:Class="Form.Dialogs.ExitDialog"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Form"
        mc:Ignorable="d"
        Title="ExitDialog" Height="200" Width="300">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="35*"/>
            <RowDefinition Height="33*"/>
            <RowDefinition Height="36*"/>
            <RowDefinition Height="35*"/>
            <RowDefinition Height="37*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="48*"/>
            <ColumnDefinition Width="95*"/>
            <ColumnDefinition Width="39*"/>
            <ColumnDefinition Width="95*"/>
            <ColumnDefinition Width="41*"/>
        </Grid.ColumnDefinitions>
        <TextBlock Text="Do you really want to quit?" x:Name="ResponseTextBox"
Grid.Column="1" Grid.ColumnSpan="3" Grid.Row="1" TextAlignment="Center" FontSize="16" />
        <Button
            Content="Cancel" Grid.Column="1" Grid.Row="3" Click="CancelButton_Click"
            Style="{StaticResource MaterialDesignRaisedButton}"
            Height="Auto"
            ToolTip="Resource name: MaterialDesignRaisedButton"
            Foreground="#FF4D99D0"
            Background="#FFE8E8E8" BorderBrush="{x:Null}">
        </Button>
        <Button
            Content="Quit" Grid.Column="3" Grid.Row="3" Click="QuitButton_Click"
            Style="{StaticResource MaterialDesignRaisedLightButton}"
            Height="Auto"
            ToolTip="Resource name: MaterialDesignRaisedLightButton"
            Foreground="#FF4D99D0"
            Background="#FFE8E8E8" BorderBrush="{x:Null}">
        </Button>
```

```
    </Grid>
</Window>
```

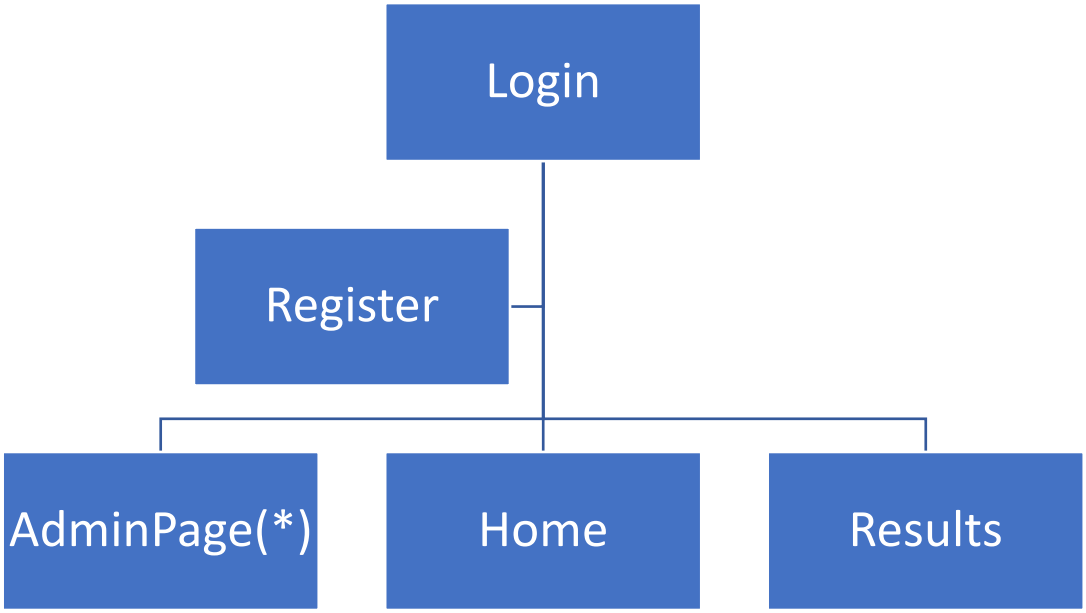
C# Code:

```
namespace Form.Dialogs
{
    public partial class ExitDialog : Window
    {
        public ExitDialog()
        {
            InitializeComponent();
        }

        private void QuitButton_Click(object sender, RoutedEventArgs e)
        {
            DialogResult = true;
        }

        private void CancelButton_Click(object sender, RoutedEventArgs e)
        {
            DialogResult = false;
        }
    }
}
```

הלקוח - סביבת Web

סביבה זו מיועדת לשחקנים ולמנהלים. בסביבה פשוטה זו ניתן לקבל ...

תרשים זרימת החלונות

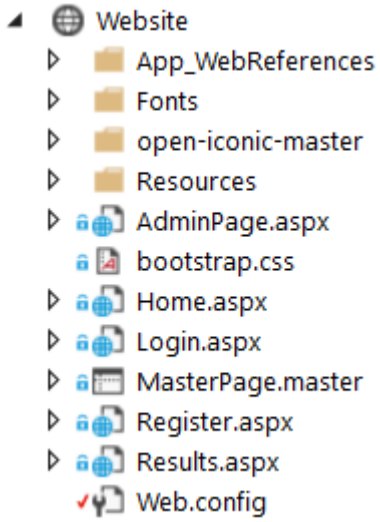


* העמוד AdminPage ניתן לכניסה אך ורק בידי מנהלים.

הגדרת השרת באתר – **בקובץ** (Web.config)

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.6.1"/>
    <httpRuntime targetFramework="4.6.1"/>
  </system.web>
  <system.webServer>
    <defaultDocument>
      <files>
        <add value="Login.aspx"/>
      </files>
    </defaultDocument>
    <directoryBrowse enabled="false"/>
  </system.webServer>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_IService" />
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://192.168.0.18:8733/Design_Time_Addresses/Service/Service/"
        binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IService"
        contract="TakiService.IService" name="BasicHttpBinding_IService" />
    </client>
  </system.serviceModel>
</configuration>
```

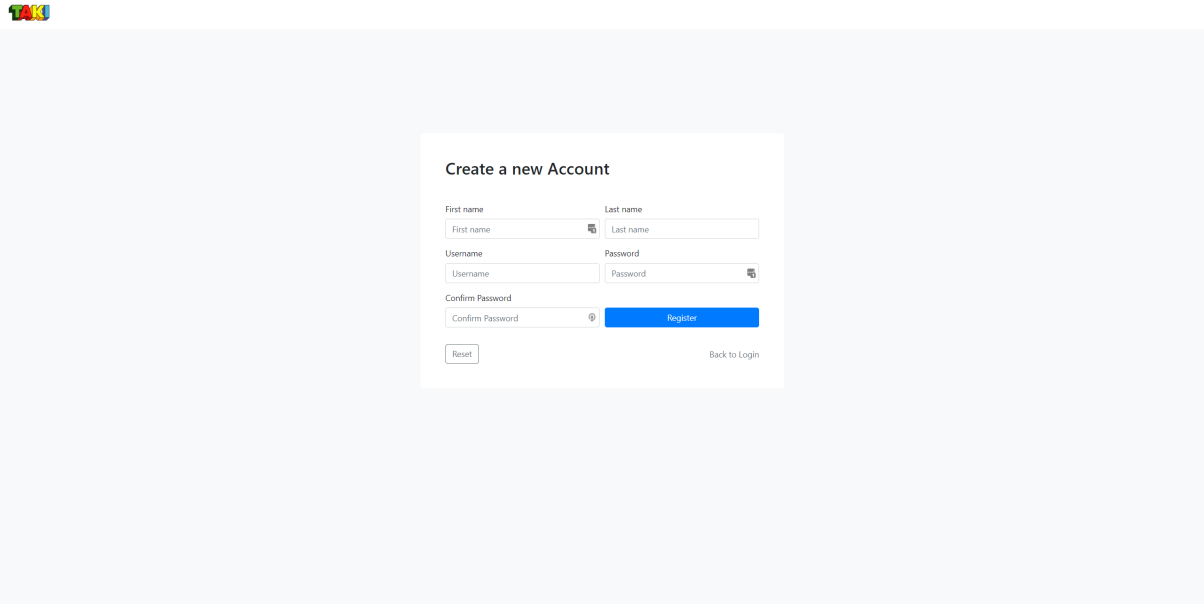
מבנה התיקייה:



קוד לדוגמה:

דף Register:

דף זה הוא דף אליו מגיעים אנשים להם אין משתמש, על מנת ליצור משתמש חדש.



Aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
CodeFile="Register.aspx.cs" Inherits="Register" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
        <div class="container bg-white p-5 rounded" style="width:700px;margin-top:200px;"
runat="server">
            <div class="form-group pb-4">
                <h2>Create a new Account</h2>
            </div>
          <div class="form-row" runat="server">
            <div class="col-md-6 mb-3" runat="server">
              <label for="validationDefault01">First name</label>
              <input type="text" class="form-control" id="validationDefault01"
placeholder="First name" required runat="server">
            </div>
            <div class="col-md-6 mb-3" runat="server">
              <label for="validationDefault02">Last name</label>
              <input type="text" class="form-control" id="validationDefault02"
placeholder="Last name" required runat="server">
            </div>
          </div>
          <div class="form-row" runat="server">
            <div class="col-md-6 mb-3" runat="server">
              <label for="validationDefault03">Username</label>
              <input type="text" class="form-control" id="validationDefault03"
placeholder="Username" required runat="server">
            </div>
            <div class="col-md-6 mb-3" runat="server">
              <label for="validationDefault04">Password</label>
              <input type="Password" class="form-control" id="validationDefault04"
placeholder="Password" required runat="server">
            </div>
          </div>
          <div class="form-row" runat="server">
            <div class="col-md-6 mb-3" runat="server">
              <label for="validationDefault05">Confirm Password</label>
              <input type="Password" class="form-control" id="validationDefault05"
placeholder="Confirm Password" required runat="server">
            </div>
            <div class="col-md-6 mb-3" runat="server">
              <label for="submit"> </label>
              <asp:button ID="submit" onclick="Register_Button_Click" class="form-control btn
btn-primary mt-2" runat="server" Text="Register" />
            </div>
          </div>
            <div class="form-group" runat="server">
                <div visible="false" ID="ErrorDiv" class="alert alert-danger" role="alert"
runat="server">
                    <span ID="noUserError" runat="server"></span>
                </div>
            </div>
          <button type="reset" class="btn btn-outline-secondary">Reset</button>
          <a href="Login.aspx" class="text-muted float-sm-right pt-2" ID="Text"
runat="server">Back to Login</a>
        </div>
</asp:Content>
```

C# code:

```
public partial class Register : Page
{
    ServiceClient _service;

    protected void Page_Load(object sender, EventArgs e)
    {

        _service = new ServiceClient();

        ErrorDiv.Visible = false;

        if (_service == null)
        {
            ErrorDiv.Visible = true;
            noUserError.InnerText = "Not Connected to server.";
        }
    }


    protected void Register_Button_Click(object sender, EventArgs e)
    {
        string firstNameValue = validationDefault01.Value;
        string lastNameValue = validationDefault02.Value;
        string usernameValue = validationDefault03.Value;
        string passwordValue = validationDefault04.Value;
        string confirmPasswordValue = validationDefault05.Value;

        if (passwordValue.Equals(confirmPasswordValue)) //checks if password and confirm
password are equal
        {
            if (_service.PasswordAvailable(passwordValue) &&
_service.UsernameAvailable(usernameValue)) //checks if this password is used or not
            {
                ErrorDiv.Visible = false;
                noUserError.InnerText = "";

                if (_service.Register(firstNameValue, lastNameValue, usernameValue,
passwordValue)) //checks if registration succeeded
                {
                    noUserError.InnerText = "";
                    ErrorDiv.Visible = false;
                    Response.Redirect("Login.aspx");
                }
                else
                {
                    ErrorDiv.Visible = true;
                    noUserError.InnerText = "There was an error. we couldn't register
you.";
                }
            }
            else
            {
                ErrorDiv.Visible = true;
                noUserError.InnerText = "password already in use.";
            }
        }
        else
        {
            ErrorDiv.Visible = true;
            noUserError.InnerText = "passwords don't match";
        }
```

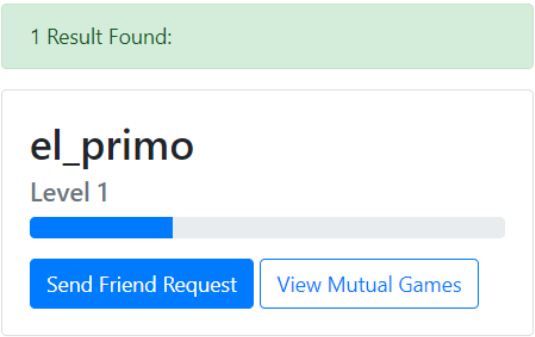
```
    }
}
```

דף Results

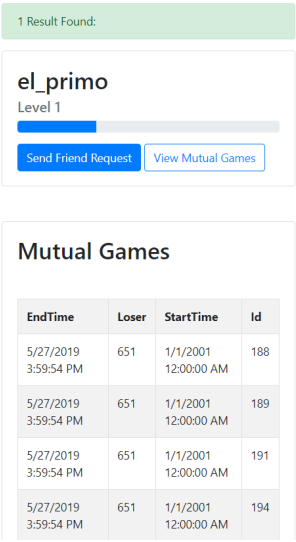
דף זה משמש כדף תוכן באתר.

דרך ראשונה להגיע לדף זה היא דרך חיפוש שחקנים בתיבת החיפוש בפינה הימנית העליונה של האתר רק לאחר כניסת המשתמש למערכת:

לאחר חיפוש שם של שחקן מסויים, יוצגו תוצאות חיפוש, ובמקרה ונמצא השחקן יוצג אודותיו מידע.



לאחר לחיצה על הכפתור View Mutual Games, יוצגו מתחת כל המשחקים המשותפים של שחקן זה והמשתמש עצמו.



Aspx:

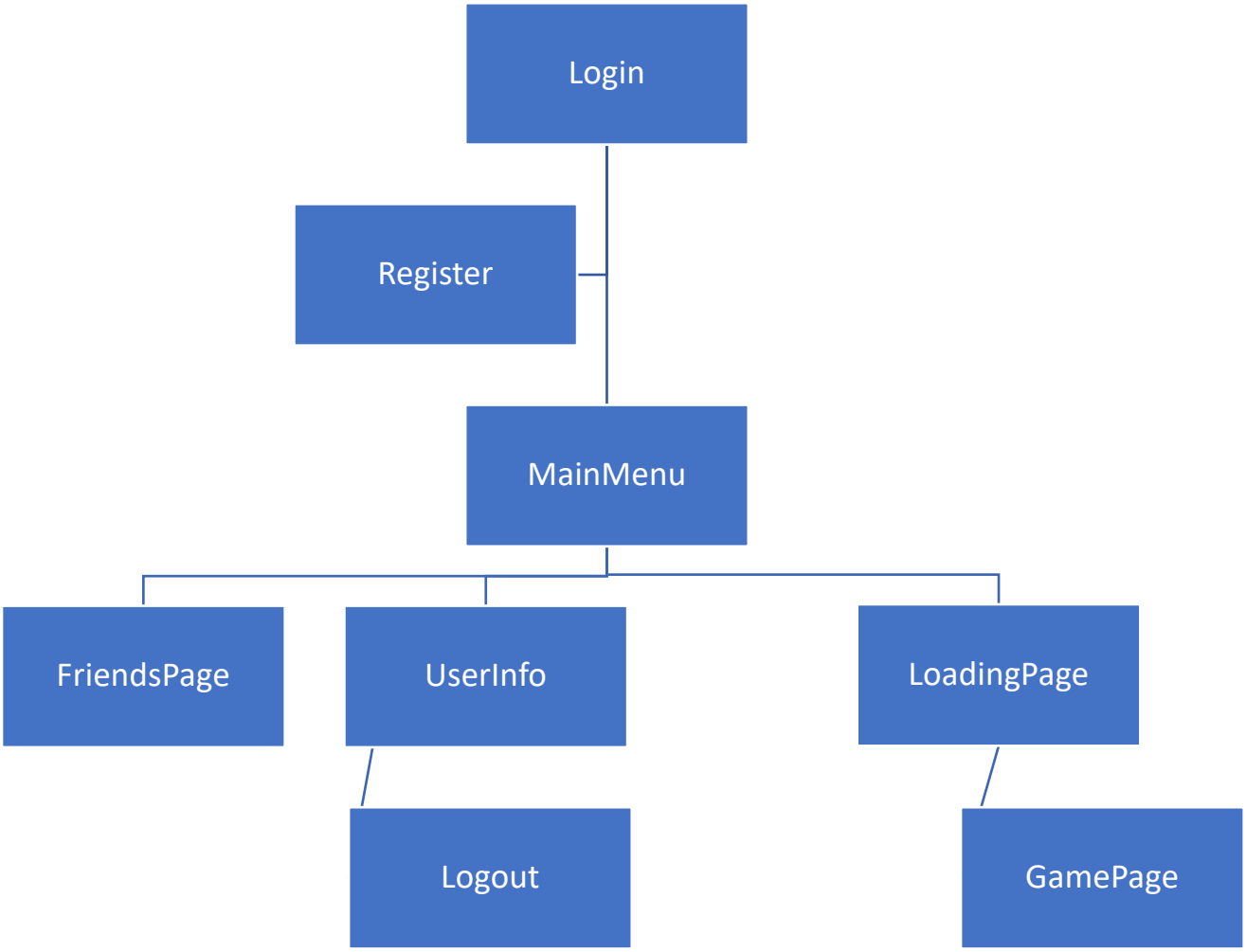
```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Results.aspx.cs" Inherits="Results" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <div class="container bg-white p-5 rounded" style="width:500px;margin-top:200px;"
Runat="Server">
        <div visible="false" ID="ErrorDiv" class="alert alert-danger" role="alert"
runat="server">
            <span ID="noUserError" runat="server">No Results.</span>
        </div>
        <div visible="false" ID="SuccessDiv" class="alert alert-success" role="alert"
runat="server">
            <span ID="Span1" runat="server">1 Result Found:</span>
        </div>
        <div class="card" visible="false" id="ResultDiv"  runat="server">
          <div class="card-body"  runat="server">
            <h2 class="card-title" id="Username" runat="server"></h2>
            <h5 class="card-subtitle mb-2 text-muted" id="Level" runat="server"></h5>
            <div class="progress mb-3"  Runat="Server">
                <div id="ProgressBar" class="progress-bar" role="progressbar"
style="width:30%" aria-valuenow="30" aria-valuemin="0" aria-valuemax="100"
Runat="Server">
                </div>
            </div>
              <div ID="IsFriend" Visible="false" runat="server">
                <span>Friend</span>
                  <img src="open-iconic-master/svg/check.svg" style="fill: #46A346;
Height: 13px; Width: 13px; padding-left: 5px"/>
              </div>
              <asp:button ID="AddFriendButton" class="btn btn-primary"
OnClick="SendFriendRequestButton_Clicked" runat="server" Text="Send Friend Request" />
              <asp:button ID="ViewMutualGames" OnClick="ViewMutualGamesButton_Clicked"
class="btn btn-outline-primary" runat="server" Text ="View Mutual Games"/>
          </div>
        </div>
        <div visible="false" ID="NoGamesDiv" class="alert alert-warning mt-3"
role="alert" runat="server">
            <span ID="Span2" runat="server">You Have No Mutual Games.</span>
        </div>
        <div class="card mt-5" visible="false" id="MutualGamesDiv"  runat="server">
            <div class="card-body"  runat="server">
                <h2 class="card-title" id="H1" runat="server">Mutual Games</h2>
                <asp:GridView ID="GridView1" class="table table-striped table-bordered
mt-5" runat="server">

                </asp:GridView>
            </div>
        </div>
    </div>
</asp:Content>
```

C# code:

```
public partial class Results : Page
{
    private ServiceClient service;
    User u1;
    User u2;

    protected void Page_Load(object sender, EventArgs e)
    {
        if (MasterPage.CurrentUser == null)
        {
            Response.Redirect("Login.aspx");
        }
        else
        {
            if (MasterPage.SearchResults == null)
            {
                ErrorDiv.Visible = true;
                ResultDiv.Visible = false;
                SuccessDiv.Visible = false;
            }
            else
            {
                ErrorDiv.Visible = false;
                ResultDiv.Visible = true;
                SuccessDiv.Visible = true;

                Username.InnerText = MasterPage.SearchResults.Username;
                Level.InnerText = "Level " + MasterPage.SearchResults.Level.ToString();


                service = new ServiceClient();

                u1 = MasterPage.SearchResults;
                u2 = MasterPage.CurrentUser;

                // check if are friends
                if (service.AreFriends(u1.Id, u2.Id))
                {
                    AddFriendButton.Visible = false;
                    IsFriend.Visible = true;
                }
            }
        }
    }

    protected void SendFriendRequestButton_Clicked(Object sender, EventArgs e)
    {
        service.MakeFriends(u1, u2);
        AddFriendButton.Visible = false;
    }

    protected void ViewMutualGamesButton_Clicked(Object sender, EventArgs e)
    {
        GameList gl = service.GetMutualGames(u1.Id, u2.Id);

        if (gl.Count > 0)
        {
            GridView1.DataSource = gl;
            GridView1.DataBind();

            MutualGamesDiv.Visible = true;
        }
        else
        {
            NoGamesDiv.Visible = true;
        }
    }
}
```

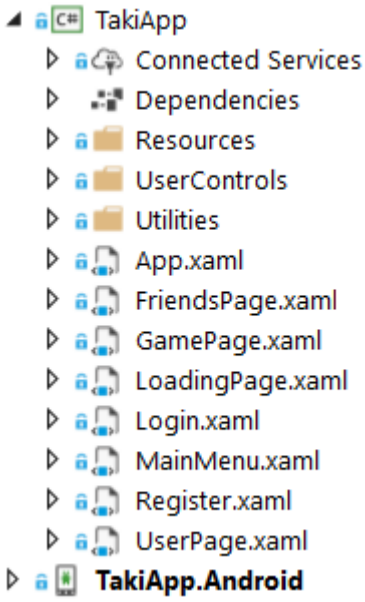
הלקוח – סביבת (Xamarin)Android

תרשים זרימת הדפים:



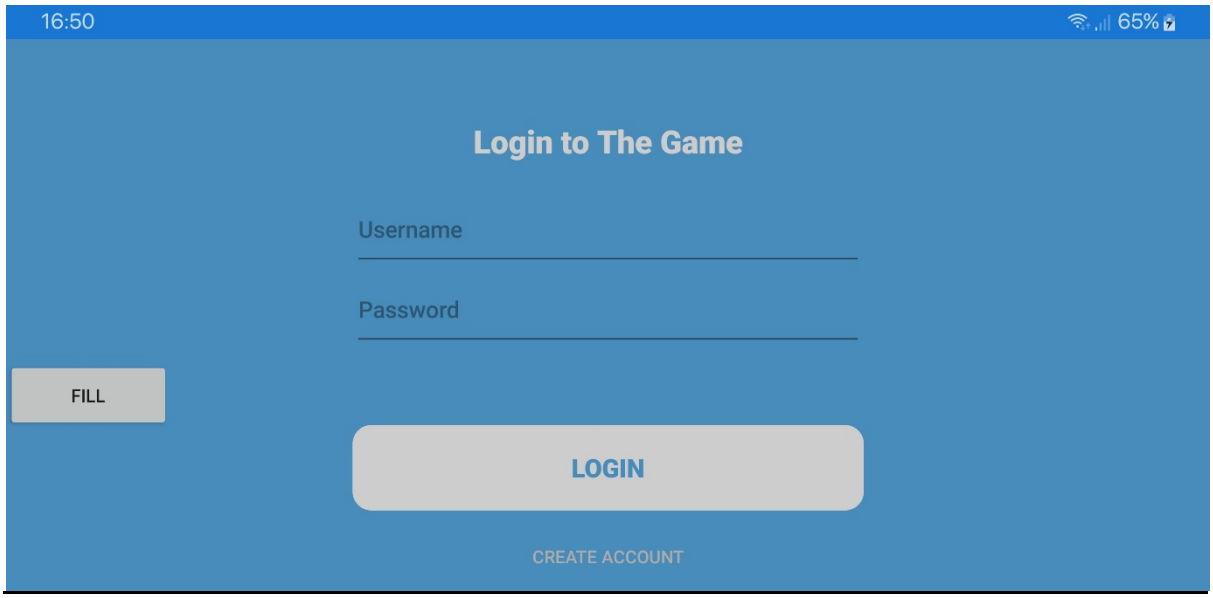
פרויקט זה הוא בעצם זהה לפרויקט הלקוח סביבת WPF, אך עם התאמות לסביבה הניידת בשימוש בטכנולוגיית Xamarin.

מבנה התיקיה



דוגמה לקוד

דף Login



XAML:

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="TakiApp.Login">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
```

```
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <Label FontSize="25" TextColor="#e3e3e3" FontAttributes="Bold"
HorizontalTextAlignment="Center" Grid.Row="1" Grid.Column="2" Grid.ColumnSpan="3"
Text="Login to The Game"/>
        <Entry x:Name="Username" TextColor="#e3e3e3" Placeholder="Username" Grid.Row="2"
Grid.Column="2" Grid.ColumnSpan="3"/>
        <Entry x:Name="Password" TextColor="#e3e3e3" IsPassword="true"
Placeholder="Password" Grid.Row="3" Grid.Column="2" Grid.ColumnSpan="3"/>
        <Label x:Name="NoUserError" TextColor="Silver" IsVisible="true"
HorizontalTextAlignment="Center" VerticalTextAlignment="End" Margin="10" Grid.Row="4"
Grid.Column="2" Grid.ColumnSpan="3" />

        <Button x:Name="LoginButton" BackgroundColor="#e3e3e3" HeightRequest="60"
TextColor="#FF509BD0" Padding="15" CornerRadius="13" Clicked="LoginButton_Click"
Grid.Row="4" Grid.RowSpan="2" VerticalOptions="End" Grid.Column="2" Grid.ColumnSpan="3"
Text="Login" FontSize="20" FontAttributes="Bold"/>
        <Button x:Name="RegisterButton" VerticalOptions="End"
Clicked="RegisterButton_Click" BackgroundColor="Transparent" TextColor="Silver"
Grid.Row="6" Grid.Column="2" Grid.ColumnSpan="3" Text="Create Account" />

        <Button x:Name="AdminFillButton" Clicked="AdminFillButton_Clicked" Text="Fill"
Grid.Row="4" Grid.Column="0"/>
    </Grid>
</ContentPage>
```

C# Code:

```
namespace TakiApp
{
    public partial class Login : ContentPage
    {
        ServiceClient _service;
        private string _usernameValue;
        private string _passwordValue;

        public Login()
        {
            InitializeComponent();
            BackgroundColor = Xamarin.Forms.Color.FromRgb(80, 155, 208);
```

```
            _service = new ServiceClient();
            _service.LoginCompleted += Serv_LoginCompleted;
        }

        private void LoginButton_Click(object sender, EventArgs e)
        {
            _usernameValue = Username.Text;
            _passwordValue = Password.Text;

            if (_usernameValue != null && _passwordValue != null)
            {
                _service.LoginAsync(_usernameValue, _passwordValue);
            }
            else
            {
                NoUserError.Text = "Please fill up all the fields!";
            }
        }

        private void Serv_LoginCompleted(object sender, LoginCompletedEventArgs e)
        {

            Device.BeginInvokeOnMainThread(async () =>
            {
                MainMenu.CurrentUser = null;
                string msg = null;

                if (e.Error != null) { msg = e.Error.Message;
                    Console.WriteLine(msg);
                    }
                else if(e.Result == null) { msg = "Wrong Username or Password"; Console.WriteLine(msg); }
                else if (e.Cancelled) { msg = "Didn't Work!"; Console.WriteLine(msg); }
                else
                {
                    MainMenu.CurrentUser = e.Result as User;
                    await Navigation.PushModalAsync(new MainMenu());
                }
                NoUserError.Text = msg;
            });
        }

        private void RegisterButton_Click(object sender, EventArgs e)
        {
            Navigation.PushModalAsync(new Register());
        }

        private void AdminFillButton_Clicked(object sender, EventArgs e)
        {
            Username.Text = "Samuelov1";
            Password.Text = "123";

            LoginButton_Click(null, null);
        }
    }
}
```

Read Me

לקוח WCF / Android

1. יש לבצע כניסה למערכת על ידי לחיצת הכפתור Fill, או על ידי מילוי
Username: “Samuelov1”
Password: “123”
2. ליצירת משתמש חדש, יש ללחוץ על הכפתור Register בקצה התחתון של הדף, ולמלא פרטים.
3. למידע אודות השחקן, יש ללחוץ על שם השחקן המופיע בפינה הימנית למעלה.
4. להיסטוריית המשחקים, יש ללחוץ על הכפתור Game History (WPF בלבד)
5. על מנת לנהל את החברים במשחק, יש ללחוץ על הכפתור הימני התחתון.
6. <u>למנהלים בלבד:</u>
על מנת להגיע לעמוד המנהלים יש ללחוץ על כפתור הפרופיל (שלב 2), ואז ללחוץ על הכפתור Admin Button.
בהגיעכם לעמוד המנהל, ניתן לנהל את המשתמשים הרשומים:
בכדי למחוק – יש לבחור משתמש וללחוץ על כפתור המחיקה.
בכדי לעדכן - יש ללחוץ על שדה אצל משתמש, לשנות את הערך וללחוץ על עדכן.
בכדי להוסיף – יש ללחוץ על כפתור ההוספה, למלא את השדות הריקים, וללחוץ על העלאה.

7. <u>להתחלת משחק:</u>
יש ללחוץ על הכפתור Multiplayer, ולאחר מכן לבחור את מספר השחקנים איתם רוצים לשחק (2,3,4).

<u>כיצד משחקים:</u>
כאשר הקלפים שלך מוצגים על רקע ירוק, זהו תורך. על מנת לשחק, יש לבחור קלף, ואז ללחוץ על הקלף שבמרכז השולחן, שהוא ערימת הקלפים.
בכדי לקחת קלף מהקופה, יש ללחוץ על הכפתור האלכסון בצורת קלף.

למידע אודות שחקנים אחרים ולהוספתם כחברים, יש ללחוץ על שמם(WPF) או על תמונתם(Android).

לעזיבת המשחק יש ללחוץ על כפתור היציאה בפינה הימנית העליונה.

8. להתנתקות מהמערכת, יש להגיע לדף הפרופיל (שלב 2), וללחוץ על כפתור היציאה בפינה הימנית התחתונה.

לקוח Web

1. יש לבצע כניסה למערכת על ידי לחיצת הכפתור Fill, או על ידי מילוי
   Username: “Samuelov1”
   Password: “123”
2. ליצירת משתמש חדש, יש ללחוץ על הכפתור Register בקצה התחתון של הדף, ולמלא פרטים.
3. לחיפוש שחקנים אחרים, יש למלא את תיבת החיפוש בפינה העליונה למעלה בשם, וללחוץ על הכפתור Search.
4. <u>למנהלים בלבד:</u>
   למידע אודות כלל המשתמשים, יש ללחוץ על הכפתור Admin שבפינה הימנית העליונה.
5. להתנתקות מהמערכת, יש ללחוץ על הכפתור האדום Logout בפינה הימנית העליונה.

ביבליוגרפיה

1. אתר StackOverflow https://stackoverflow.com
2. אתר msdn https://msdn.microsoft.com/en-us
3. ממשק Bootstrap https://getbootstrap.com
4. סביבת Xamarin https://docs.microsoft.com/en-us/xamarin
5. תמונות Google Images https://www.google.com/imghp?hl=en
6. והכי חשוב, דורית הכוכבת שעזרה יותר מכל הנ"ל