### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

**Friends With Common Interest:**

פיצ'ר זה מציג רשימה של החברים איתם יש למשתמש באפליקציה עמודי פייסבוק משותפים (עמודים להם המשתמש והחבר עשו "לייק"). ברשימה זו יוצגו כל החברים עם עמוד או יותר משותפים.

הפונקציה שמבצעת זאת מופיעה בקוד ב-MainLogic, וה-form הראשי של האפליקציה עושה בה שימוש בעת לחיצה על כפתור "Click here to see them!".

**Artists Recommendations:**

פיצ'ר זה עושה שימוש ב-API חיצוני של last.fm.

האפליקציה בודקת מי האמנים האהובים על המשתמש, ובעבור כל אחד מהם ממליצה לו על X אמנים דומים (הכמות לבחירתו).

עיקר ההופעה בקוד היא במחלקה LastFmApi, במחלקה קיימת מתודה שמבצעת קריאת API לשרת של lastfm באופן א-סינכרוני. הקריאה מחזירה תשובה בפורמט XML עם מידע על אומנים דומים לאמן המבוקש. מתוך המידע אנו מסננים רק את שמות האמנים ומחזירים רשימה שלהם.

**שימוש ב-Data Binding:**

עשינו שימוש בפיצ'ר זה בטבלת ה-events שלנו. כמו כן קישרנו בין שמות ה-events לחלק מהשדות שלו והם מוצגים בהתאם ל-event הנוכחי שהמשתמש בוחר לראות מידע עליו.

Graphical user interface, application

Description automatically generated

השימוש בקוד הוא במחלקת ה-MainForm, במתודה fetchEvents() שמופעלת בעקבות הלחיצה על כפתור 'Click here to get Upcoming Events'.

**שימוש בתכנות אסינכרוני:**

בקוד יש 2 שימושיים עיקריים בתכנות אסינכרוני:

1. בפיצ'ר Artist Recommendations הנ"ל, המתודה GetArtistRecommendations ב-Façade שיוזכר בהמשך (ובהתאמה המתודה fetchRecommendationsאשר מפעילה אותה) היא מתודה אסינכרונית והיא משתמשת במתודה אסינכרונית ספציפית אחרת שנמצאת במחלקה LastFmApi.

השימוש בה נעשה על מנת לאפשר למשתמש פונקציונליות של שימוש שוטף ורציף בעת טעינת ההמלצות, שלוקח זמן מכיוון והוא תלוי בתשובה של שרת חיצוני (של lastfm).

השימוש נעשה באמצעות async ו-await.

1. ב-MainForm, כלל הפונקציות שתפקידן להביא מידע ולהציג אותו למשתמש עושות שימוש בתכנות אסינכרוני, שכן המידע מיובא משרת חיצוני של פייסבוק. זאת באמצעות שימוש ב-threads – כמובן עם שימוש בפעולות מהצורה Invoke new Action כדי שלא תהיה חפיפה בין ה-threads (Cross thread operation). השימוש בתכנות אסינכרוני באופן דומה נעשה כדי לאפשר למשתמש להינות משימוש שוטף באפליקציה בעת שהנתונים נטענים.

### תבנית מס' 1 – [Singelton]

* סיבת הבחירה / שימוש בתבנית:

השתמשנו בסינגלטון עבור מחלקת הלוגיקה בתוכנית. מחלקה זו מחזיקה נתונים ומבצעת פעולות לוגיות עבור כלל הטפסים השונים במערכת. בחרנו להשתמש בסינגלטון עבור מחלקה זו כי רצינו לוודא שיש מקור יחיד של אמת עבור הנתונים השונים בלוגיקה של התוכנית, וכדי למנוע מצב בו יש כפילויות של נתונים אודות המשתמש או עבור תוצאות פעולות בלוגיקה. בנוסף רצינו שהנתונים והמופע של מחלקת הלוגיקה יהיה נגיש מכל טופס ללא צורך "להעביר" את המופע הזה בין הטפסים – גישה נוחה מנקודות שונות במערכת אשר תבנית סינגלטון מאפשרת.

* אופן המימוש:

המימוש במחלקת AppLogic (ובנוסף בטפסים המבצעים AppLogic.instance) :

* הבנאי נמצא בprivate.
* מופע סטטי בתוך המחלקה.
* פרופרטי סטטי שמחזיר את המופע הסטטי , ורק אם המופע עדיין לא נוצר , הוא יוצר כזה.
* Thread Safe : שימוש בlock.
* "שחקנים" :

Singelton:AppLogic

Clients:The forms- MainForm,StartForm.

* Sequence Diagram

Diagram, schematic

Description automatically generated

* Class Diagram

Diagram

Description automatically generated

### תבנית מס' 2 – [Proxy]

* סיבת הבחירה / שימוש בתבנית:

עבודה בטוחה ומוסדרת של MainForm מול ממשק חיצוני - Facebook User .

זהו -

Protective Proxy

כיוון שFacebook User הינו ממשק חיצוני המשתמש בשרתי פייסבוק , אנו מעוניינים להגן על המשתמש ,בדוגמה שלנו MainForm

מפני תקלות שיכולות להיות בשרת. הגדרנו בממשק מוסדר מהן המתודות שבטוח להשתמש בהן. הבעיה יכולה להיות למשל מכך שישנן מתודות שקיימות , אך בפועל בעבודה מול פייסבוק עלולות להיות תקלות והמידע לא מתקבל , או לא מתקבל בצורה תקינה.

ביצוע של אינקפסולציה – Facebook User חושף באמצעות Proxy והממשק את המתודות אליהן יש התאמה , ובכך גם משיג בטיחות שימוש – הMainForm ניגש למתודות דרך הממשק.

* אופן המימוש:

ממשק IFacebookUser מגדיר את המתודות שעל Proxy לממש.

MainForm מכיל מופע של הממשק – IFacebookUser.

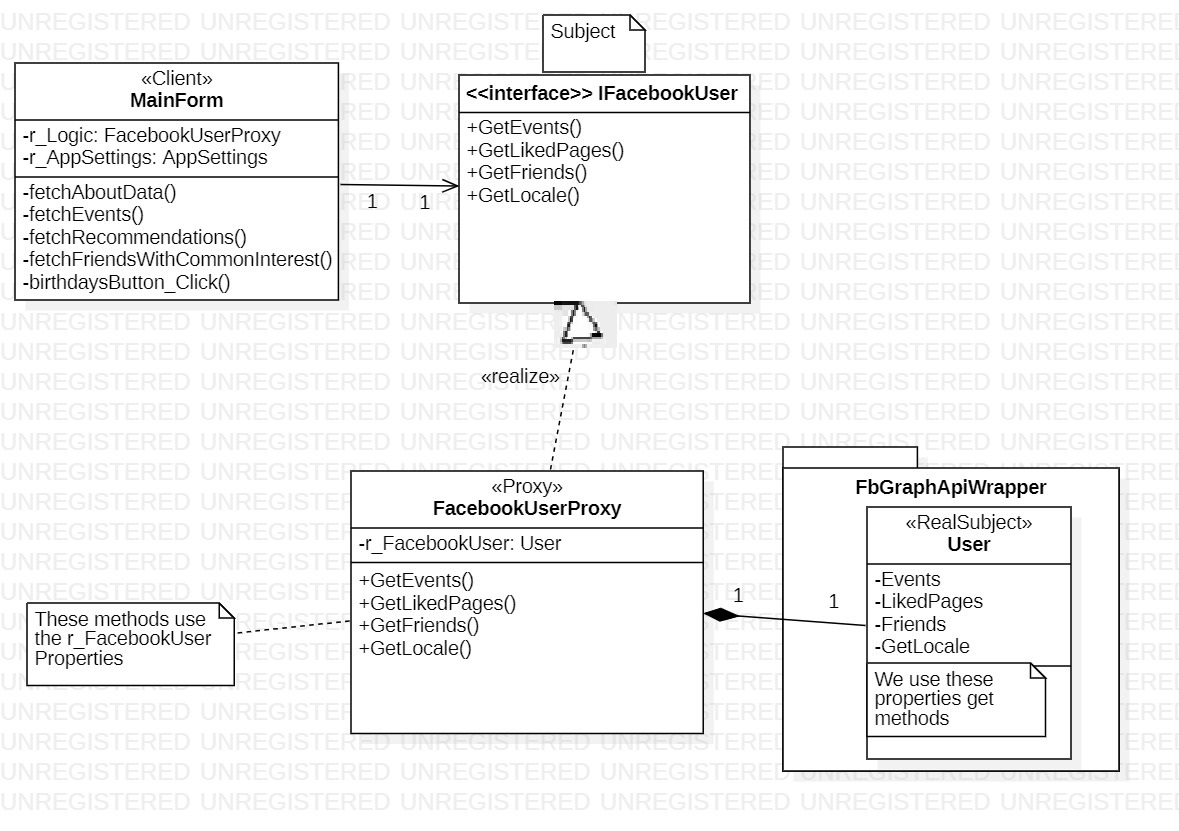
FacebookUserProxy מכיל מופע של User , וממש את הממשק IFacebookUser באמצעות שימוש בProperties של User.

וכך MainForm מממש את המתודות שמשמשות לעבודה מול Facebook User תוך שימוש בממשק , הממומש ע"י Proxy , אשר מכיל מופע של User.

**Sequence diagram:**

Diagram

Description automatically generated

* **Class** **Diagram**

### תבנית מס' 3 – [Façade]

* סיבת הבחירה / שימוש בתבנית:

השתמשנו בתבנית זו כדי לאפשר לטופס הראשי עבודה מול Api חיצוני המספק המלצות מוזיקליות.

ביצענו זאת מכמה סיבות: לצורך בטיחות בשימוש בעבודה מול Api חיצוני שעלולות להיות בו תקלות . כמו כן לצורך פישוט העבודה של הטופס מול הApi (הFacade מחזיר בצורה נוחה את הנתונים הרלוונטים לטופס). לצורך תחזוקתיות ופולימורפיזם – אם בעתיד נרצה להשתמש בApi אחר של המלצות במקום הנוכחי(הנוכחי הוא (lastFm

אז השינויים שנצטרך לבצע יהיו רק בFacade ולא בשכבה המשתמשת בו.

* אופן המימוש:

במחלקת RecommendationsFacade (והשימוש בה בmainForm

המימוש הוא של Facade אטום:

מחלקת lastFmApi ) שאיתה הfacade עובד , מוגדרת internal , ומחלקת הfacade נמצאת

באותו dll .

הmainForm מחזיק מופע של RecommendationsFacade.

הfacade מחזיק מופע של המחלקה lastfm – ומשתמש במתודות שלו כדי למממש את המתודה GetArtistRecommendations שמונגשת באופן נוח לmainForm – המשתמש בfacade.

"שחקנים":

RecommendationsFacade- Façade

mainFrom – client

lastFmApi – package.

**Sequence diagram:**

Diagram, schematic

Description automatically generated

* Graphical user interface, text, application

  Description automatically generated with medium confidenceClass Diagram