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

# פיצר 1 - LikeAnalyzer

*הפיצר מאפשר למשתמש פייסבוק לראות מי מבין חבריו אחראי לכמות הלייקים המרובה ביותר בין כל הפרסומים שלו.*

*בנוסף מאפשר לו לראות רשימה לפי כמות הלייקים, תמונה של אותו חבר, אפשרות לעשות לחבר לייק חוזר וחיתוכים נוספים כמו הגבלת מספר הלייקים לחיפוש.*

*את הפיצר ניתן למצוא תחת הקלאס* likeanalyzer

# פיצר 2 - MusicForm

הפיצר למעשה מקשר בין שני שירותים פופולריים: פייסבוק ויוטיוב.

הפיצר מושך את כל הדפים שהמשתמש עשה להם לייק בעבר, חותך את העמודים בקטוגריית מוסיקה בלבד ומציגם למשתמש. בעת לחיצה על אופן שמשקף דף שהמשתמש אהב האפליקציה מחפשת את ערוץ היוטיוב של אותו אומן ואת החיפושים הכי רלוונטיים אליו ביוטיוב. מציגה למשתמש את התוצאות ברשימה ומאפשר לו את האופציות הבאות: לנגן את השיר באפליקציה, לנגן את השיר בדפדפן, לפתוח את עמוד הפייסבוק של האומן או לפתוח את עמוד היוטיוב שלו.

### תבנית מס' 1 – [Static Factory Class]

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

לשני הפיצ'רים בתכנית קיים Form ייעודי בהם הם מומשו. ייתכן ובהמשך יתווספו Forms נוספים למערכת, אך בהכרח ירשו Form, לכן את לוגיקת היצירה של Form הוצאנו לקלאס חיצוני אשר מייצר את Form המבוקש ומחזיר מופע שלו.

כך המשתמש איננו חשוף ליצירת המופע וללוגיקה שביצירתו.

בהינתן מצב עתידי שבו יתווספו Forms נוספים למערכת, שינוי הקוד על מנת לתמוך בהם יהיה מאוד קטן ועיקרו יהיה ב Static Factory Class.

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

המימוש התבצע בכך שיצרנו מחלקת סטטית ובה מתודה סטטית, מתודה זו מחזירה Form שזהו הבסיס המשותף לכל הטפסים באפליקציה. מחלקה זו מקבלת כארגומנט את סוג הטופס הרצוי ומחזירה מופע שלו. ניתן למצוא את המחלקה בקוד בתור מחלקה בפני עצמה (זו איננה מחלקה ששוכנת במחלקה אחרת).

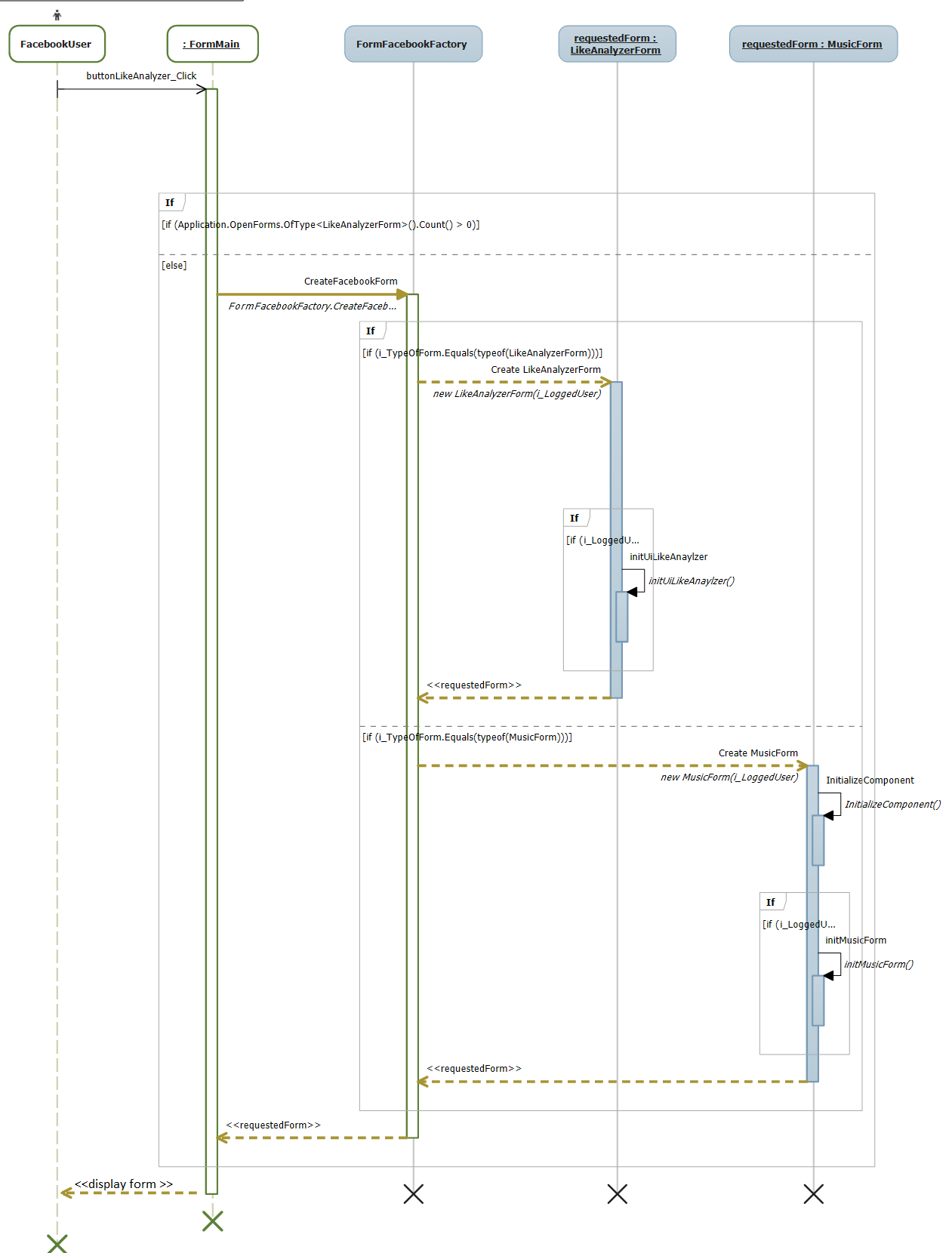
את השימוש בו ניתן למצוא FormMain במתודות:

private void buttonLikeAnalyzer\_Click(object sender, EventArgs e)

private void buttonMusicView\_Click(object sender, EventArgs e)

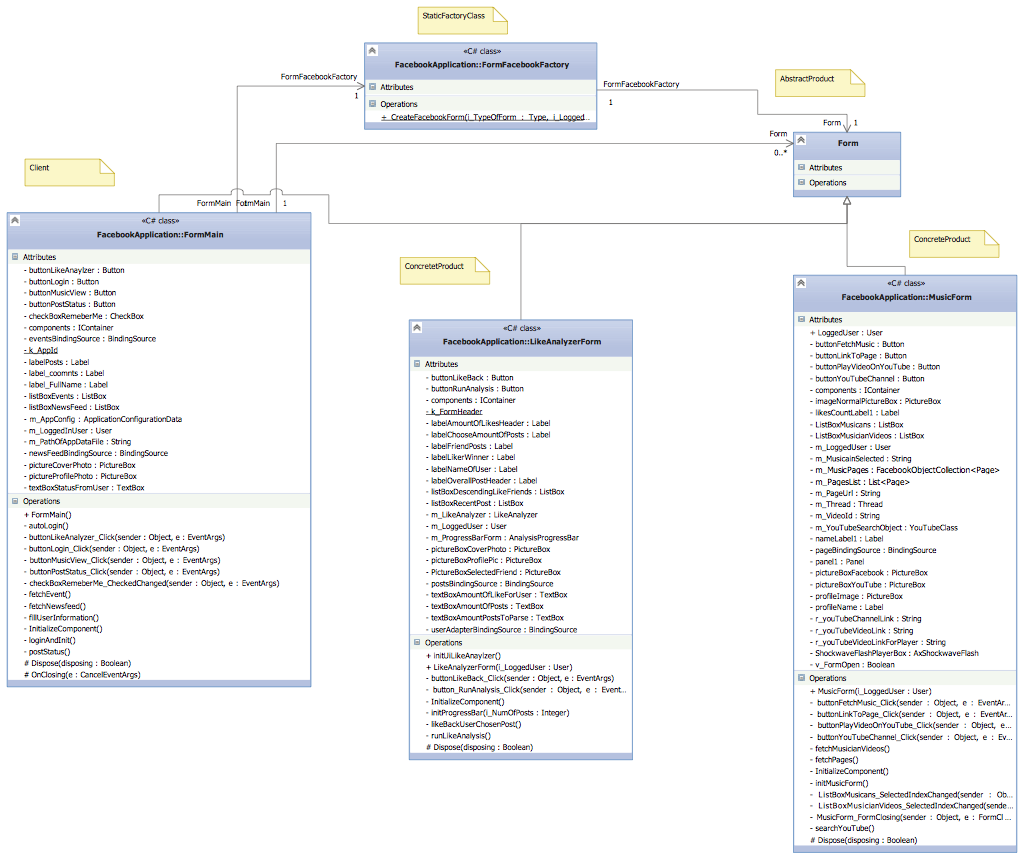
* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]



* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם



### תבנית מס' 2 – [Adapter]

* סיבת הבחירה / שימוש בתבנית:
* [תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

רצינו לעשות Data Binding ב LikeAnalyzerForm.

הטופס כולל\דורש בפעולתו את שמות המשתמשים, עבור כל משתמש את הפוסטים שלו ומספר הלייקים שלו (כלומר מספר הלייקים שעשה למשתמש שכרגע מחובר לאפליקציה). מאחר האובייקט User מציע את כל הפרטים הנחוצים פרט לאחרון, מספר הלייקים שביצע. על מנת שיהיה אפשר לעשות Data Binding עצמאי, כלומר בלי צורך לעדכן את הUI בשינויים כלשהם (לדוגמא בבחירת משתמש כלשהו והצגת מספר הלייקים שלו) בחרנו להשתמש בAdapter על מנת לסייע בכך.

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

יצרנו קלאס חדש ששמו UserAdapter, הוא מחזיק רפרנס לאובייקט User וקיימים לו getters רק עבור המידע הרלוונטי (less is more) ל Data Binding ב- LikeAnalyzerForm .

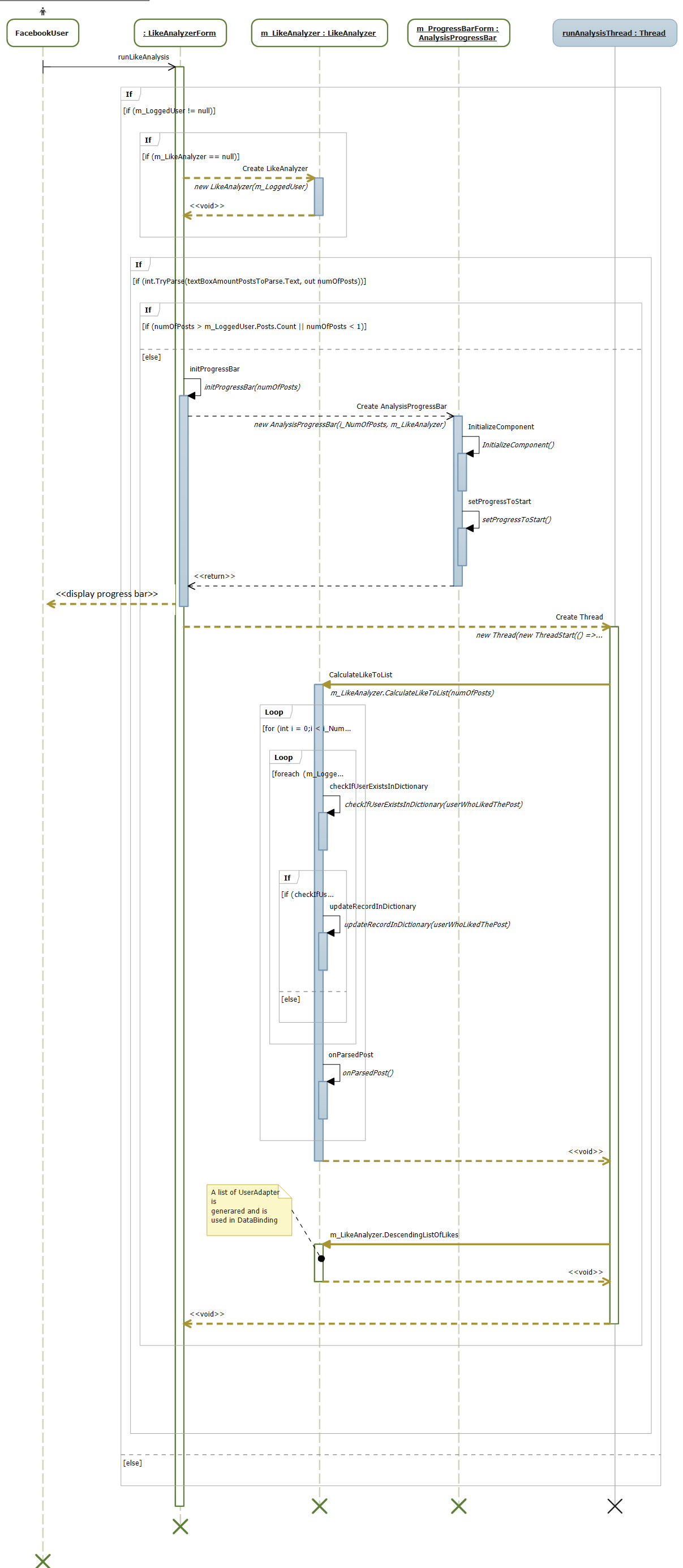
כך יכולנו לבצע את ה Data Binding על ה UserAdapter בכך שעשינו getters לPOST לNAME והערך המוסף הנחוץ, מספר הלייקים שביצע.

ניתן למצוא את המחלקה בקוד לצד כל המחלקות האחרות (מחלקה זו איננה שוכנת בתוך מחלקה אחרת).

את השימוש בה ניתן למצוא במחלקה LikeAnalyzer ב Getter ל DescendingListOfLikes.

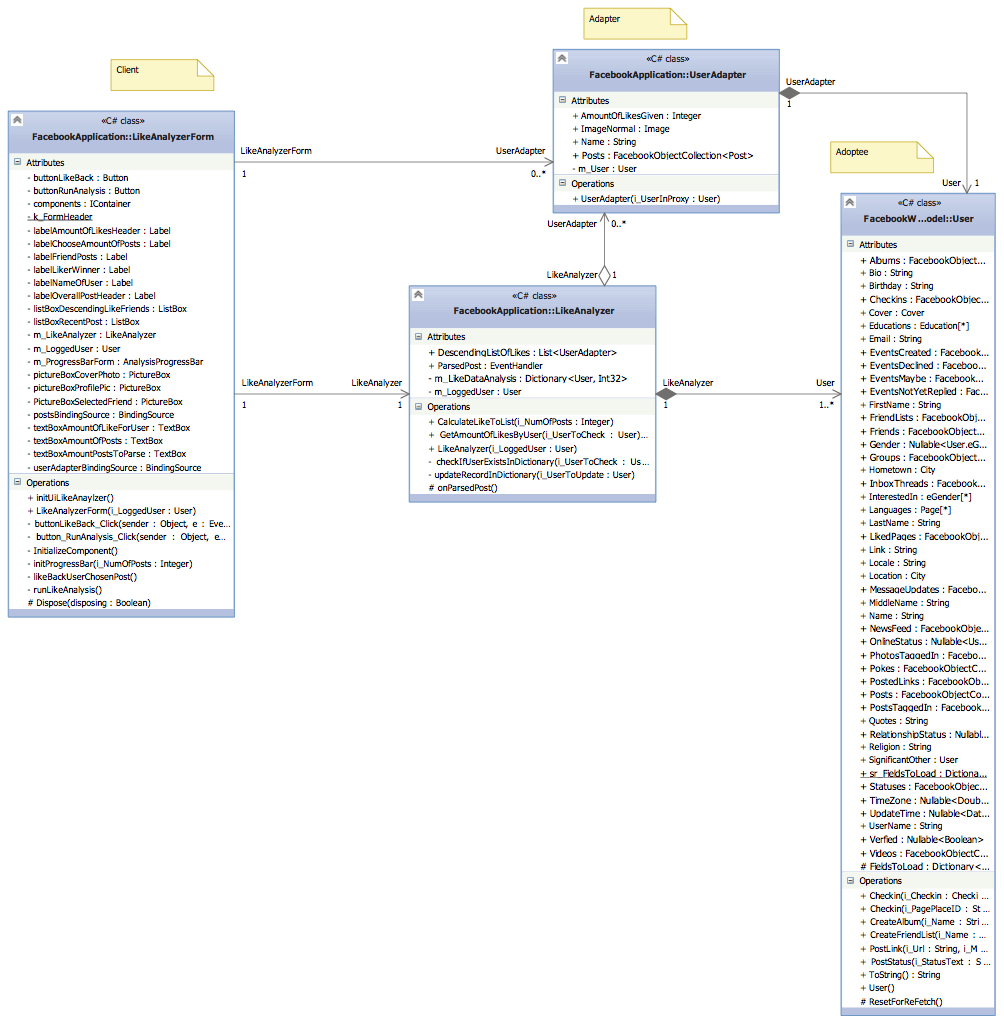
* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]



* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם



### 

### 

### תבנית מס' 3 – [Proxy]

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

רצף הפעולות במערכת, ב- FromMusic אומר שכל אומן מהפייסבוק שהמשתמש מתעניין בו ולחץ עליו תופיע רשימה של השירים של אותו אומן מהיוטיוב. סיבת הבחירה בפרוקסי היא למעשה לחקות את הרכיב שמבצע את הקישור ליוטיוב ולשמור מבנה נתונים פנימי.

כך שבמהלך הריצה אם המשתמש יחזור לאותו אומן שבחר קודם השליפה תהיה מקומית ולא יתבצע חיפוש ביוטיוב שוב.

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

מומש באמצעות קלאס YouTubeProxy ורפרנס ושימוש בו ב- FormMusic כשהאירועים הללו קורים:

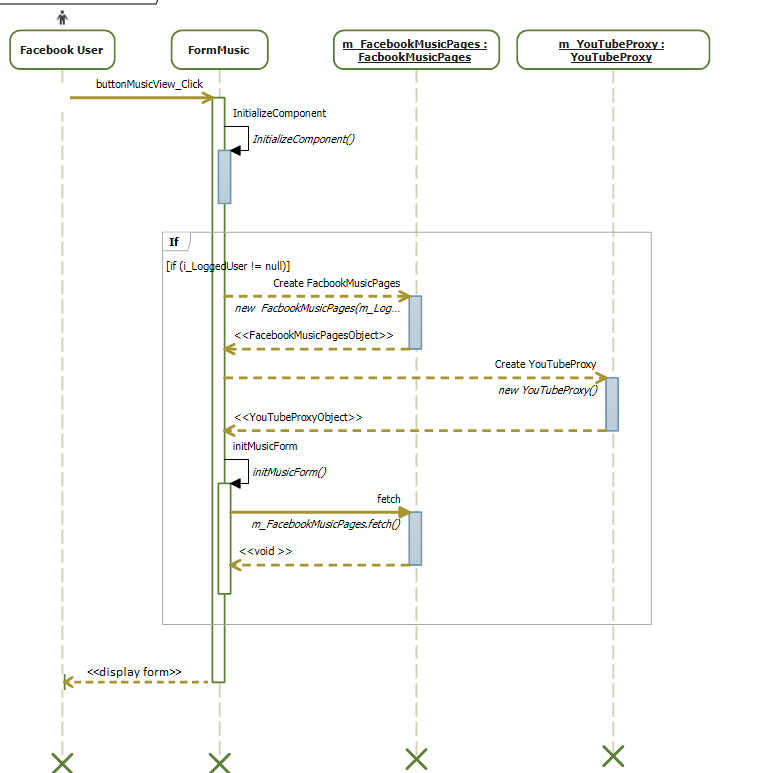
youTubeProxyVideoListBindingSource CurrentChanged

pageBindingSource CurrentChanged

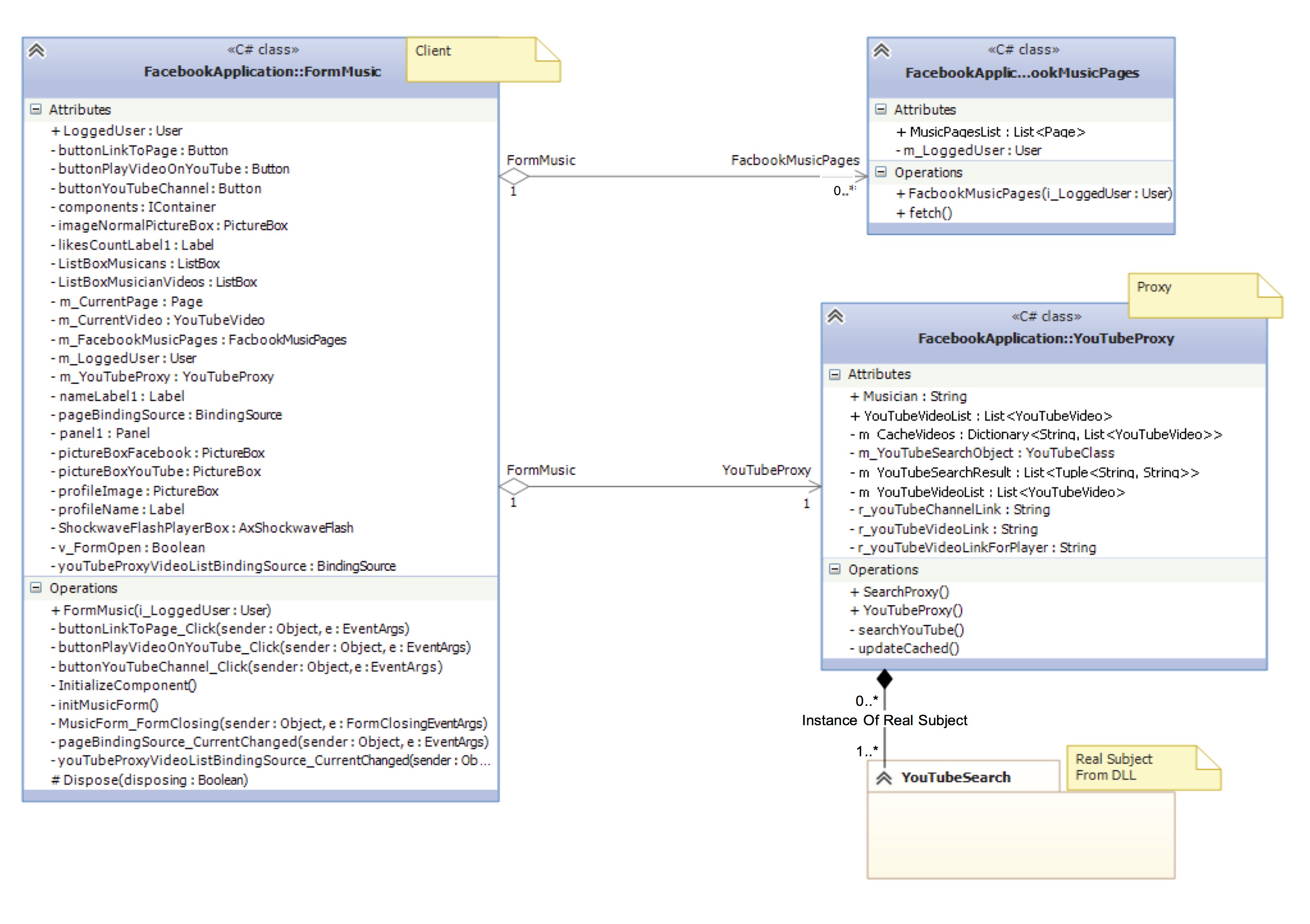
בנוסף יש קישור DATA BINDING דפי המוסיקה שמתקבלים מאובייקט FacebookMusicPages שמפלטר רק את הדפים שקשורים למוסיקה.

DATA BINDING נוסף מתבצע בין לYOUTUBE פרוקסי והצגת רשימת הוידאו, בנוסף כל פעם שהמשתמש בוחר אומן חדש מתעדכן ה- datasource של הפרוקסי.

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

**עבודה אסינכרונית**

עליכם להפגין שימוש בתיכנות אסינכרוני. יש לציין במסמך שאתם מגישים **היכן בקוד** עשיתם שימוש

בתיכנות אסינכרוני, **כיצד**, **ומדוע**.

דאגנו לכך שהת'רד הראשי יעבוד ללא הפרעה כך שחוויית המשתמש תהיה חלקה, לכן, כל פעולה חישובית כלשהי שעלולה להקפיא את הUI הפעלנו בת'רד וכך ממשק המשתמש חלק וכל החישובים שדורשים זמן (הבאת נתונים מפייסבוק, אנליזה של מידע לוקלי) מתבצעת ברקע.

דוגמאות לכך ניתן למצוא במקומות הבאים:

במחלקה LikeAnalyzerForm –

private void runLikeAnalysis() – הרצת האנליזה לרשימת הלייקים.

private void likeBackUserChosenPost() – החזרת לייק למשתמש.

במחלקה FormMain –

private void autoLogin() – עדכון פרטי המשתמש בUI.

private void fillUserInformation() – משיכת הנתונים מפייסבוק.

private void postStatus() – העלאת פוסט לפייסבוק.

**עבודה עם Data Binding**

עליכם להפגין שימוש ב- Data Binding . יש לציין במסמך שאתם מגישים **היכן בקוד** עשיתם שימוש.

במחלקה LikeAnalyzerForm

private void runLikeAnalysis()

במחלקה FormMain

private void fetchEvent()

private void fetchNewsfeed()

במחלקה FormMusic

private void initMusicForm()

private void pageBindingSource\_CurrentChanged(object sender, EventArgs e)