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

* **Funny Post**הפיצ'ר מציג למשתמש גלריית תמונות מהאתר הבא:

[http://alltheragefaces.comHYPERLINK "http://alltheragefaces.com/"/](http://alltheragefaces.com/)

כאשר המשתמש יכול לבחור תמונה מהגלרייה ולשמור אותה למחשב או לפתוח אותה בצייר או להעלות אותה לפייסבוק שלו.

בנוסף יש לו אפשרות להוסיף לתמונה פילטר שמשתנה בכל יום מהאתר הבא : [http://fonxvard.comHYPERLINK "http://fonxvard.com/"/](http://fonxvard.com/)

הקוד הרלוונטי נמצא בקבצים : RageWebController, RageImageWebController, FunnyForm, RagePersonJson.

* **Statistics**

המשתמש בוחר קבוצת חברים מתוך רשימת חבריו לגביהם הוא מעוניין לדעת כמה לייקים העניקו לו בשתי קטגוריות : תמונות ופוסטים.

לאחר חישוב קל של המערכת תוצג למשתמש הסטטיסטיקה אודות כמות הלייקים שביצע כל אחד מהחברים הנבחרים לפי הקטגוריה הרלוונטית. המשתמש יוכל לבחור האם למיין את התוצאות.

הקוד הרלוונטי נמצא בקבצים StatisticsUser, StatisticsLogic , StatisticsForm, StatisticsComputingService.

בתרגיל זה הוספנו עוד 2 פיצ'רים נוספים :

1. אפשרות לקבל עובדה אקראית על תאריך יומולדת של אחד החברים ולהעלות עובדה זו לקיר שלו בפייסבוק. הקוד הרלוונטי נמצא ב FactForm
2. אפשרות לקבל תמונה של חתול רנדומלי מהאינטרנט ולהעלות אותו לקיר של החברים שלך בפייסבוק. הקוד הרלוונטי נמצא ב CatForm.

**תבנית מס' 1 – [FACADE]**

* סיבת הבחירה / שימוש בתבנית:

[הסיבה שבה בחרנו להשתמש בתבנית זו היא העובדה שישנן פעולות מורכבות בכדי לבצע חישובים מסוימים, ובחרנו לתת ל-client ממשק נוח להפעלת הפונקציות הרלוונטיות עבורו.

המתודות שמחלקת ה-FACADE מציעה מאוד נוחות לשימוש, אך מימושן כאמור מורכב.]

* אופן המימוש:

[מחלקת ה-FACADE שלנו היא מחלקת ה-StatisticsComputingService, והיא מוכלת בתוך ה-StatisticsForm ומכילה אובייקט של מחלקת StatisticsLogic באמצעותו היא חושפת ל-client את האופציה להשיג את הסטטיסטיקה הרלוונטית וגם למיין את התוצאה הסופית, זאת תוך שהיא מסתירה את הפעולות המורכבות בהן היא משתמשת.]

* 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 2 - SINGLETON**

* סיבת הבחירה / שימוש בתבנית:

הסיבה בה השתמשנו בתבנית זו היא מכוון שהיא מאפשרת לנו לדעת ולשמור על המצב בו יש רק אוביקט אחד של משתמש פייסבוק יחיד המחובר למערכת בזמן נתון.

* אופן המימוש:

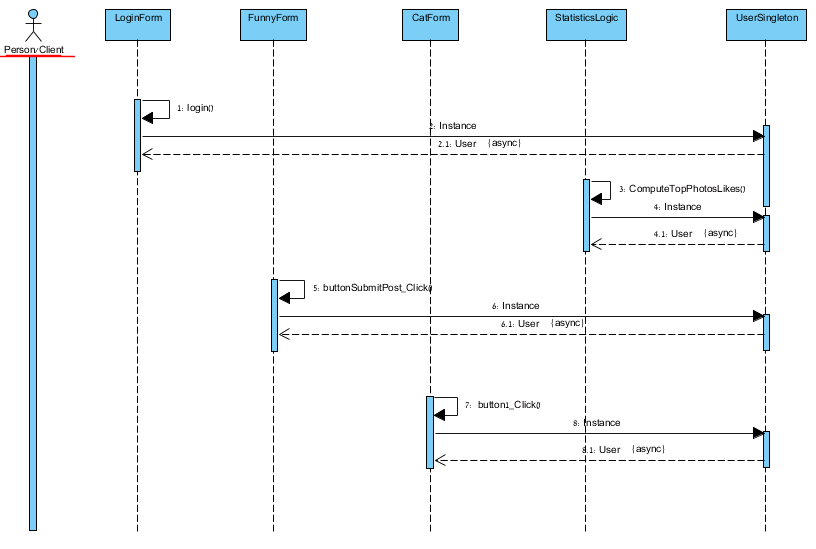
מחלקת ה UserSingleton משמש כסינגלטון במערכת והוא שומר בתוכו את המשתמש היחיד המחובר. המחלקה מאפשרת יצירה של אוביקט אחד מסוג זה. לפני שהאוביקט הראשון נוצר המחלקה קוראת לפונקציה הסטטית FacebookService.Login ומתחילה את תהליך הכניסה של המשתמש למערכת, ורק אם המשתמש התחבר לפייסבוק אז נוצר מופע של המחלקה וכן שומרת בתוכה את משתמש הפייסבוק הנוכחי שהתחבר בהצלחה. תהליך זה יכול לקרות שוב לאחר שאוביקט אחד נוצר במערכת.

הסינגטון מומש בעזרת private constructor כמו סינגלטון קלאסי שמונע יצירה של מופע מבחוץ,

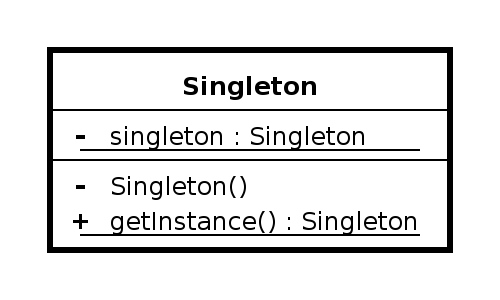
הסינגלטון לא מוגן ע"י מנעול בקטע קריטי, שכן לא ראינו צורך לכך.

הסינגלטון לא מאפשר יצירה של מופע חדש של הסינגלטון וכן לא מאפשר השמה חוזרת לאוביקט User שהוא מכיל.

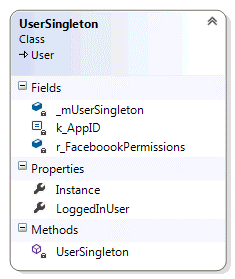
* Sequence Diagram



* Class Diagram

A classic Singleton Class Diagram

Our UserSingleton Class Diagram



**תבנית מס' 3 – [שם התבנית]**

* סיבת הבחירה / שימוש בתבנית:

[תיאור הסיבה / הצורך בשימוש בתבנית במערכת שלכם]

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

* 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

**שימוש ב-עבודה אסינכרונית**

ביצענו שימוש בעבודה אסינכרונית במספר מקומות בקוד:

* בקובץ StatisticsForm, כאשר המערכת מחשבת את הנתונים (פוסטים ו/או תמונות של המשתמש), הפעולה לוקחת זמן. לכן, דאגנו שהיא תתבצע ב-Thread אחר (computeThread), על מנת שהמשתמש עדיין יוכל להזיז את חלון האפליקציה.

כמו כן, בזמן חישוב הנתונים, רץ Thread נוסף (ProgressBarThread) שתפקידו להראות שהמערכת מעבדת את הנתונים ע"י שימוש ב-progress bar. כאשר ה-Thread של עיבוד הנתונים מסתיים, אנו מקבלים אינדיקציה לכך שניתן להציג את התוצאות, ואז מוסיפים את התוצאות ל-listview ב- ProgressBarThread, ולכן אנו נאלצים להשתמש ב-Inovke בכדי להוסיף את התוצאות ל-listview שנוצר ב-Thread הראשי.

(מטפלים בבעית ה-cross-thread-operation) .

הקוד הרלוונטי נמצא ב-StatisticsForm במתודה:

private void buttonGetLikesStatistics\_Click(object sender, EventArgs e).

* במחלקה WebController יש שתי פונקציות שמטרתם להשיג מידע ע"פ כתובת Uri בצורה אסינכרונית.
* public void DownloadWebDataAsync(Uri i\_Uri)
* public void DownloadWebStringAsync(Uri i\_Uri)

המחלקה הנ"ל מממשת את הInterface IWebControllerשמוגדרת בפרויקט ומהווה את הממשק להשגת מידע מהאינטרנט. המחלקה מכילה אוביקט מטיפוס WebClient וע"י שימוש בפעולות אסינכרוניות שמוגדרות לפיה כמו :

m\_WebClient.DownloadDataAsync(i\_Uri);

m\_WebClient.DownloadStringAsync(i\_Uri);

וע"י השמת פונקציה אנונימית שתופעל ברגע שהמידע הגיע מהאינטרנט, תגרום להפעלת Events שהמחלקה מכילה בתוכה.

**שימוש ב-Data Binding**

נעשה שימוש ב DataBinding בשתי מחלקות :

FactForm בה מוצגת רשימת חברים שהמשתמש יכול לבחור מתוכם ועבור כל חבר שנחבר מוצג המידע שלו אוטומטית ע"י UserDataBinding..

CatForm בשימוש דומה למה שיש ב .FactForm