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

* **הפיצ'ר Photos Select By Dates**

בפיצ'ר זה המשתמש יוכל לצפות בתמונות שפרסם בין תאריך התחלה ותאריך סיום שיבחר , וכן יוכל לבחור האם לצפות בתמונות ממוינות לפי תאריך הפרסום או לפי מספר הלייקים של כל תמונה ,בנוסף יוכל המשתמש להוריד את התמונות שבחר למחשב ולבחור האם שמות התמונות במחשב יהיו לפי תאריך היצירה שלהם או לפי הid שלהם בפייסבוק.

* **הפיצ'ר Post Analyst**

בפיצ'ר זה המשתמש יוכל לבחור מילים כך שיוצגו לו פוסטים המכילים את אותם מילים, ויוצג לו סיכום סה"כ כמה פוסטים הכילו כל מילה שבחר ,וכן לצפות בפוסטים ממוינים לפי תאריך הפרסום או לפי מספר הלייקים של כל פוסט ולבחור מה האורך המינימלי של פוסט על מנת שיוצג , בנוסף המשתמש יוכל לקבל ניתוח של הפוסטים לפי הנושאים : פוליטיקה , מדעים , אחר. ויוצג לו בדיאגרמה אחוזי הפעילות שלו בכל אחד מנושאים אלו.

### תבנית מס' 1 – [Strategy]

* סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית מכיוון שבפיצ'ר של בחירת תמונות והורדתם למחשב רצינו לאפשר למשתמש לבחור בזמן ריצת האפליקציה האם שמות התמונות שהוא מוריד יהיו לפי תאריך יצירה שלהם או לפי הid שלהם בפייסבוק (ConcreteStrategy), על ידי כך שייצרנו את המטודה לבחירת שם לתמונה(Strategy) עם יכולת להחליף אותה (ברכיב PhotosDownloader) כמו שהתבנית ממליצה יצרנו מערכת פריקה ומודלרית שניתנת להחלפה ולכן ניתן לעשות שימוש חוזר במחלקה PhotosDownloader כך שאם בעתיד נרצה לשנות את אסטרטגיית בחירת שם לתמונה נוכל לעשות זאת גם אם לא יהיה לנו את הקוד והמחלקה PhotosDownloader תיהיה legasy component .

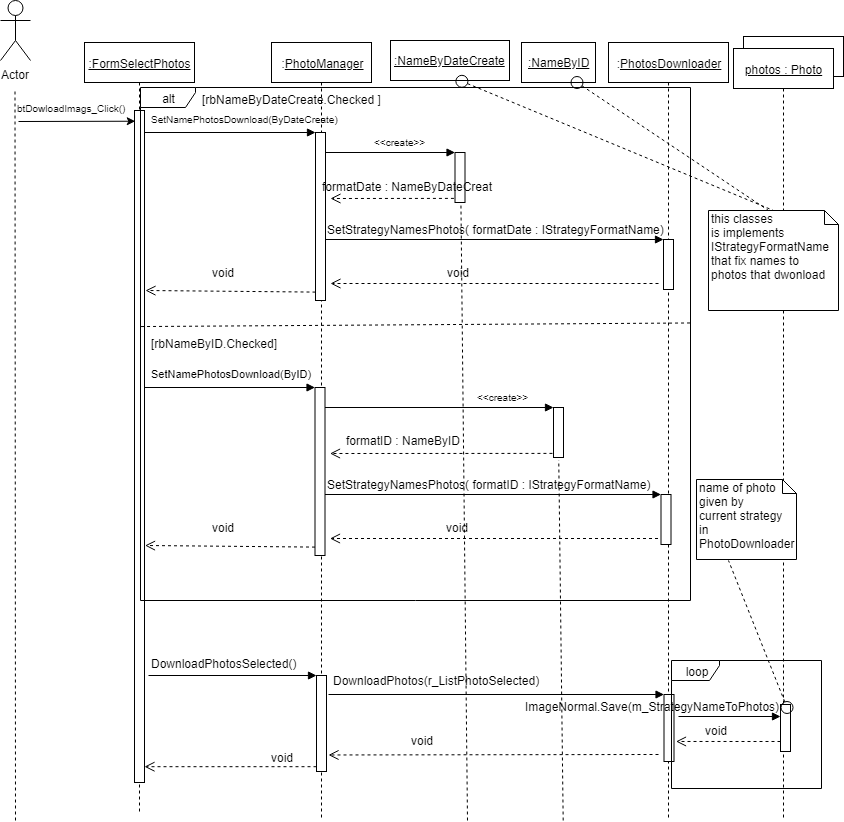
* אופן המימוש:

הוספנו לרכיב PhotosDownloader **מצביע למטודה** שמקבלת Photo ומחזירה string (Strategy), וכן הוספנו מטודה SetStrategyNamePhotos שמקבלת מצביע למטודה שמקיימת את הממשק האחיד מPhoto ל string **ומשנה את אסטרגיית** קביעת השמות למטודה שהיא קיבלה,

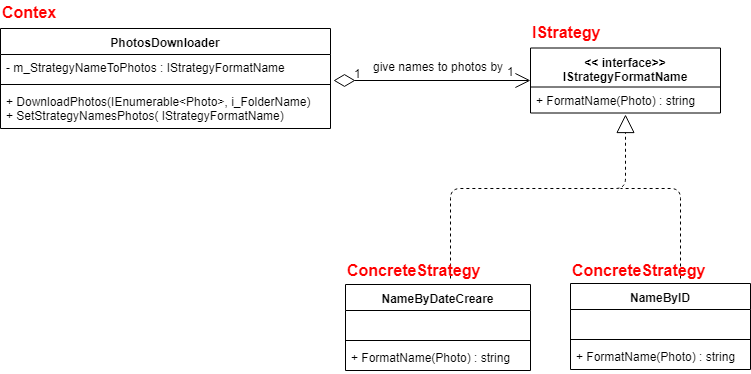
בזמן הורדת התמונה הרכיב מפעיל את **המטודה הנוכחית** על התמונה והstring שיוחזר יהיה שם התמונה.

- ניתן למצא בקוד בקבצים: PhotosDownloader, PhotoManager , FormSelectedPhotos

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – [Observer]

* סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית Observer מכיוון שרצינו לאפשר למשתמש התנתקות ויציאה מהאפליקציה בכל Form שבו הוא נימצא, כך שהאפילקציה תחזור לחלון התחלתי ותתאפס.

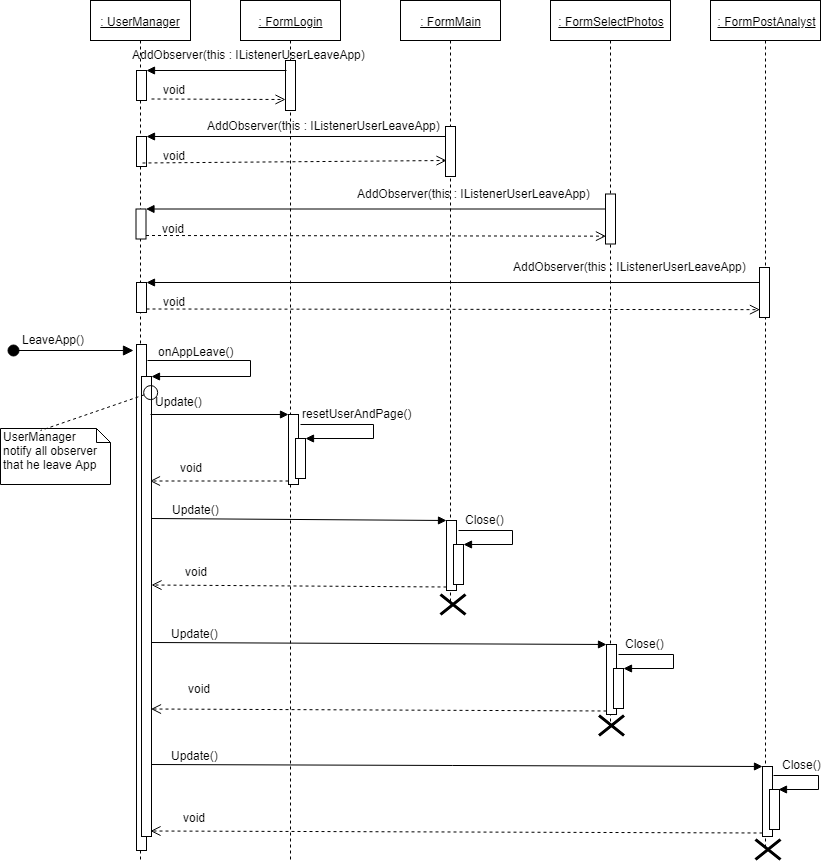
הצורך בתבנית נבע מכך שרצינו שכשהמשתמש מתנתק ויוצא מהאפליקציה (**השינוי**) הForms - FormLogin, FormMain, FormPostAnalyst, FormSelectPhotos שקיימים במערכת (**Listeners**) ידעו מכך ויפעלו בהתאם לצרכיהם (יסגרו או יתאפסו), מכיוון שהרכיב UserManager (**Notifier**) המייצג את המשתמש שמחובר למערכת הוא סינגלטוני אם בעתיד רכיבים נוספים ירצו לדעת כשהמשתמש מתנתק ויוצא מהאפליקציה הם יוכלו להוסיף את עצמם כ Listener של אירוע זה ללא שינוי קוד ובצורה דומה ולפעול בהתאם לצרכיהם.

* אופן המימוש:

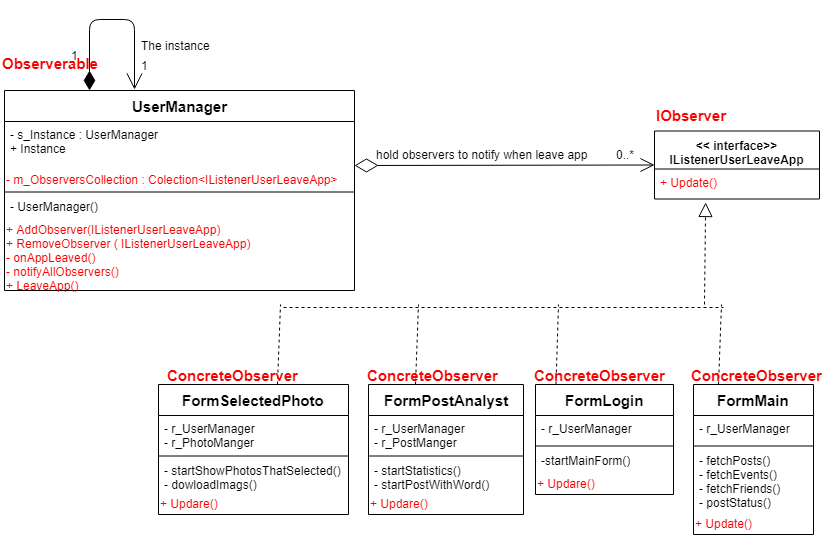
הוספנו למחלקה UserManager (notifier) שמייצגת את המשתמש שמחובר למערכת **אירוע** AppLeave שמופעל ומודיע לכל המאזינים כשהמשתמש עוזב את האפליקציה, הרכיבים FormLogin, FormMain, FormPostAnalyst, FormSelectPhotos (Listeners) נרשמים כמאזינים לאירוע זה על ידי שנותנים לו מטודה שלהם/ lambda exsprisson שיופעלו בעת התנתקות המשתמש ויציאתו מהאפליקציה (נעשה שימוש בפיצ'ר של השפה ע"י הגדרת האירוע public event ההרשמה מתבצעת ע"י =+) , וכן הוספנו לUserManager מטודה LeaveApp שבה המשתמש עוזב את האפליקציה ומפעיל את האירוע AppLeave **ומודיע למאזינים** שהוא עוזב את האפליקציה ומבצע Logout מהמערכת.

- ניתן למצא בקוד בקבצים: UserManager ,FormLogin, FormMain, FormPostAnalyst FormSelectPhotos

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – [Iterator]

* סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית מכיוון שאצלנו במערכת קיים רכיב PostsTextManager שמכיל בתוכו את הפוסטים של המשתמש והרכיב FormPostAnalyst מציג את פוסטים אלו ב . listBoxPosts הצורך בתבנית עלה מכיוון שרצינו שהרכיב PostsTextManager ((Aggregte יוכל לתת ל FormPostAnalyst ((Client את היכולת לסרוק את אוסף הפוסטים שהוא מחזיק **מבלי לחשוף את מבנה הנתונים** שבו הוא מחזיק אותם, על מנת שאם בעתיד נרצה להחליף את מבנה הנתונים שמחזיק את אוסף הפוסטים נוכל לעשות זאת ללא שינויים בClient , בנוסף רצינו לתת את האופציה של פילטור המידע כך שהאיטרטור יחזיר רק אובייקטים שעוברים את הפילטר שהClinet הגדיר (Filter iterator), התבנית Iterator עונה על צורך זה.

* אופן המימוש:

הגדרנו את הרכיב PostsTextManager (Aggregte) כמיישם את האינטרפייס IEnumrable<Post> ומכיל מטודה GetEnumerator המחזירה <IEnumerator<Post

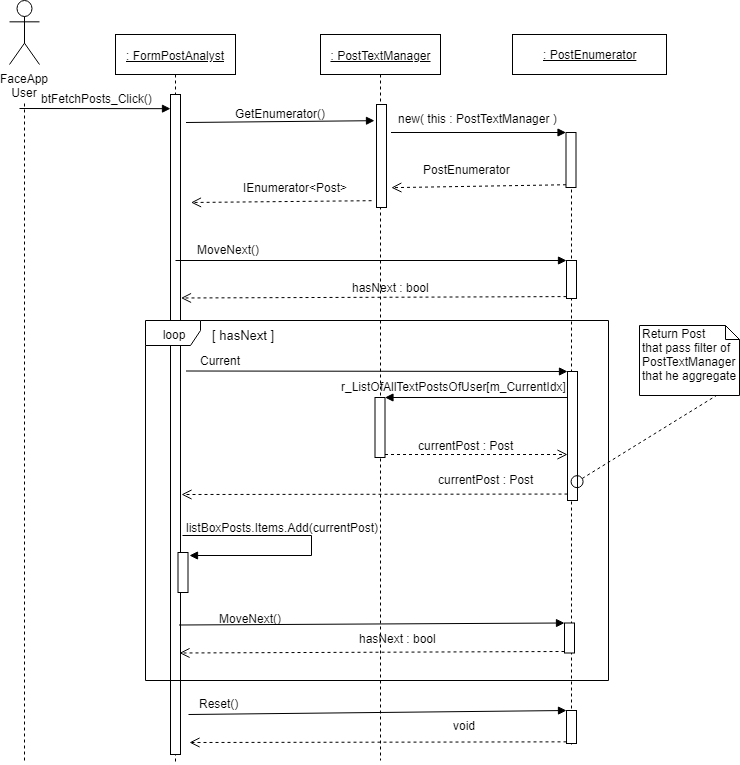
כך ש<IEnumerator<Post מכיל את האובייקט PostsTextManager ובעזרתו ניתן לסרוק את אוסף הפוסטים שקיים באובייקט על ידי קריאה למטודות MoveNext,Current,Reset ללא חשיפת מבנה הנתונים שמחזיק את האוסף.

וכן הוספנו מצביע למטודה שמקבלת Post ומחזירה bool כך שמטודה זו ניתנת לשינוי מהClient ומהווה את הפילטר של האיטרטור.

עשינו שימוש בפיצ'ר של השפה yield return שבונה עבורנו את המחלקה PostEnumerator (Iterator) כך שהאיטרטור הוא Protective וזורק Exception אם נעשה שינוי בקולקציה במהלך הסריקה(שינוי גודל וכדומה).

- ניתן למצא בקוד בקבצים : PostsTextManager, FormPostAnalyst

* Sequence Diagram



* Class Diagram

