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

* פיצ'ר שמראה הצעות לניצול זמן טוב יותר. כאשר עוברים לTAB של Timers, מופיע טיימר שמראה כמה זמן המשתמש "בזבז" באפליקציה, בבנוסף יש כפתור כשלוחצים עליו החלון יציג למשתמש משפט של מה המשתמש יכל לעשות בהתאם לזמן שבזבז באפליקציה, המשפטים בהתאם לזמן שאפליקציה פתוחה נמצאים במילון תחת המחלקה TimeWellSpend אשר נמצא בModel.
* פיצ'ר עריכת תמונות מהאלבומים ושמירתם. כאשר פותחים את TAB האלבומים, ניתן לערוך את התמונה באמצעות כפתור Edit וגם ניתן לשמור לוקאלית את התמונה שנערכה. ניתן למצוא בקוד: בView תחת חלון FormEditPicture.

### תבנית מס' 1 – [Iterator]

* סיבת הבחירה / שימוש בתבנית:

באפליקציה שלנו, אנו מאפשרים למשתמש לעבור על האלבומים שאותם העלה לפייסבוק, על מנת לאפשר למשתמש מעבר על האוספים של התמנות שלו, וכדי להפריד בין מבנה הנתונים של האוספים למעבר עליהם, יצרנו איטרטור, אשר מאפשר לעבור על אוספים שונים של תמונות בעזרת 'ממשק' פשוט ונוח לשימוש חוזר.

* אופן המימוש:

המימוש של התבנית הוא באמצעות יצירת מחלקה ImageURLIterator אשר מממשת ממשק של IEnumerator. מחלקה זו מקבלת בקונסטרקטור אוסף של סטרינגים, וכך היא חושפת ממשק לעבודה נוחה עם אותו אוסף. כמו כן יצרנו מחלקה של ImageNavigator אשר ממש את IAggregate, והוא יוצר את האיטרטור.

#את מחלקת ImageURLIterator אפשר למצוא בקוד תחת FacebookViewModel ואת מחלקת ImageNavigator ניתן למצוא תחת . FacebookViewואת IAggregate ניתן למצוא תחת CommonContract

* gdsgdgdsg


  Sequence Diagram

m\_ImageURLIterator:

:Form

m\_ImageNavigaA picture containing text, screenshot, line, diagram

Description automatically generatedtor:

* Class Diagram



1

0…\*

@

ה"שחקנים":

IEnumerator<T> as Iterator \*

ImageURLIterator as ConcreteIterator \*

IAggregate as Aggregate \*

ImageNavigator as ConcreteAggregate \*

FormMain as Client\*

### תבנית מס' 2 – [Command]

* סיבת הבחירה / שימוש בתבנית:

במערכת שלנו יש את הTabControl שמכיל מספר טאבים (פוסטים, אלבומים...).

בכל טאב מתבצע משהו שונה, רצינו שבעתיד יהיה ניתן להוסיף טאבים בצורה פשוטה יותר,

כך שכל הלוגיקה ויצירת הTAB יתבצעו ע"י פקודה אחת (שונה לכל TAB).

בגלל שהכל עטוף תחת פקודה אחת דבר זה מאפשר הפרדה בן ממשק המשתמש ללוגיקה אשר מתבצעת והרחבת או שינוי התפריט בצורה קלה יותר.

* אופן המימוש:

יצרנו תפריט מסוג tabControl אשר מכיל אוסף של TabPages. באמצעות הפורופרטי של TabPage ששמו Tag הצמדנו לכל טאב ActionCommand, ולכל ActionCommand יש דליגייט.

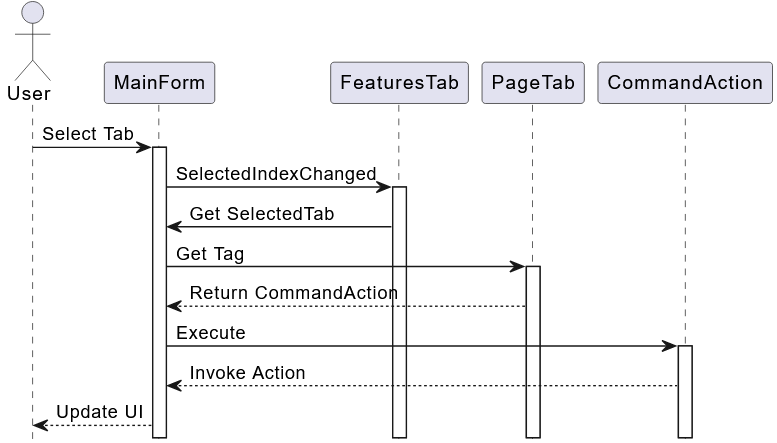
יצרנו מטודות אשר אחראיות ליישם את הלוגיקה של כל טאב כאשר הוא נבחר.

במידה ונרצה להוסיף טאב נוסף נוכל גם ליצור את התוכן שבתוכו בתוך אותה מטודה. (כאשר הטאב יבחר התוכן שלו יווצר והלוגיקה הרצוייה תתבצע) בתחילת התכנית נרשמנו בהתאמה לדליגייט של כל טאב (בעצם לדליגייט של actionCommand אשר מוצמד לטאב) וכשTAB מסויים נבחר מופעלת המטודה של tabControl (התפריט) SelectedIndexChanged.

במטודה זו יש וולידציה שאכן מה שהוצמד הוא ActionCommand והדליגיט שלו נקרא באמצעות Execute.

#אפשר למצוא בקוד את ActionCommand תחת FacebookView ואת ICommand תחת CommonContracts.

* Sequence Diagram



TabPage

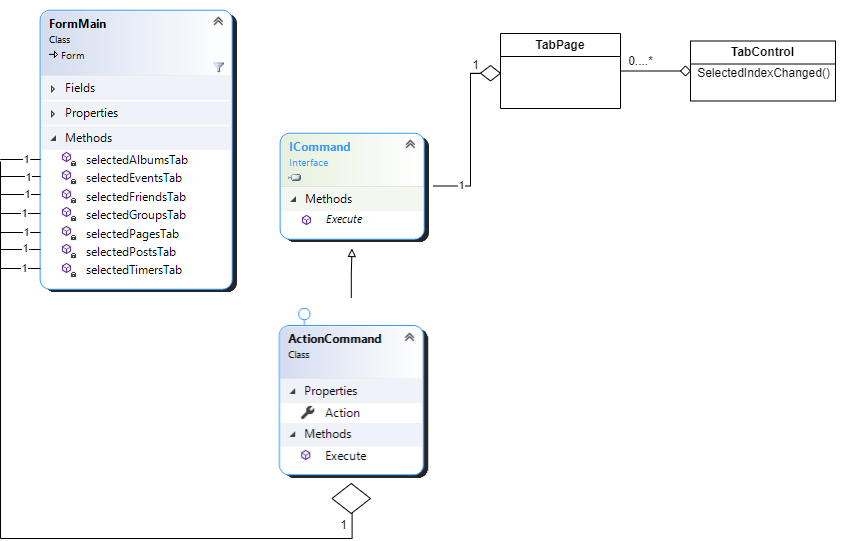
FormMain:Form

ActionCommand:

Action

tabControlFeatures : tabControl

TabPage



* Class Diagram

@ ה"שחקנים":

FormMain as Client \*

ICommand as **Command Interface**\*

ActionCommand as ConcreteCommand\*

FormMain as **Invoker \***

TabPage as UIItem\*

TabControl as Menu\*

### תבנית מס' 3 – [Observer]

* סיבת הבחירה / שימוש בתבנית:

יש אפשרות למשתמש באפליקציה שלנו להזין פוסטים חדשים,

כשהמשתמש מזין פוסט חדש, ממשק המשתמש צריך להתעדכן ולהראות את רשימת הפוסטים המעודכנת שכוללת את הפוסט החדש.

* אופן המימוש:

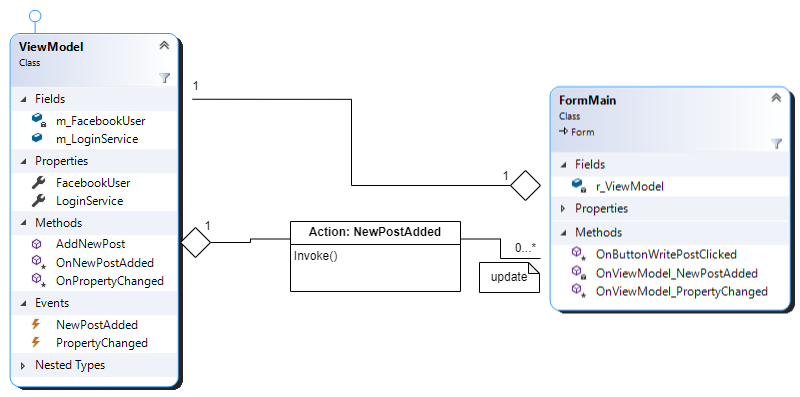
יצרנו דליגייט(NewPostAdded מסוג Action) בViewModel ונרשמנו אליו עם פונקציה שנמצאת בFormMain.

כשמשתמש כותב פוסט, הViewModel מוסיף את הפוסט החדש לאוסף הפוסטים של המשתמש שמוחזק בFacebookUser (דבר זה בעצם מדמה שליחת בקשת פוסט עם הפוסט אותו כתב המשתמש לשרתים של פייסבוק)

כך שהמשתמש מזין פוסט חדש מופעל הדליגייט ומעדכן את כל "הנרשמים" שלו על השינוי, במקרה שלנו מופעלת הפונקציה OnViewModel\_NewPostAdded שמעדכנת את הlistboxPosts כך שיופיע בו הפוסט החדש.

# אפשר למצוא בקוד תחת FACEBOOK View MODEL את NewPostAdded Action ואת הפונקציות של כתיבת ושליחת פוסט ב FormMain

* Class Diagram



* Sequence Diagram

m\_ViewModel:

**A screenshot of a computer

Description automatically generated with medium confidence**

:Form

:Action

@ה"שחקנים":

ViewModel as the "Subject" \*

FormMain as the "Observer" \*