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

* **Most Attractive Feature:**

הפיצ'ר נמצא בform משלו, שניתן להגיע אליו בלחיצה על ה button שמוגדר כ Most Attractive feature, בform מוצגים 3 פיצ'רים שהם:

א+ב מציגים למשתמש את הפוסט עם הכי הרבה לייקים והפוסט עם הכי הרבה תגובות.

ג. מציג את התמונה עם הכי הרבה תגובות.

* **Best Friend Feature:**

א. ניתן למצוא את הפיצ'ר בחלון הראשי מיד לאחר התחברות בחלון ה login. לאחר לחיצה על הכפתור יפתח טופס חדש Best Friend Feature ובטופס מופיעים שני כפתורים שמחולקים לפי החבר שעשה הכי הרבה לייקים, והחבר שהגיב הכי הרבה תגובות.

ב. הסבר קצר על הפיצ'רים , הפיצ'ר מאפשר בעצם לראות מי החבר הכי טוב שלך מבחינת פייסבוק , מכיוון שתמיכה בפייסבוק נמדדת על פי לייקים ותגובות.

**עבודה אסינכרונית – Threads:**

הפרוייקט משתמש בשרת של פייסבוק כדי להביא מידע , הבאת המידע ועיבודו מכבידה על הפרוייקט ויוצרת פרקי זמן ארוכים לטעינת מתודה / העלאת חלון.

לכן השתמשנו בעבודה אסינכרונית במקרים בהם היה את הצורך , דבר זה איפשר לנו לקצר את זמני ההמתנה למידע ועזר לנו עם העלאת הטפסים.

השימוש בטרדים איפשר לנו לבצע כמה תהליכים במקביל , בנוסף סיפק למשתמש חוויה נוחה יותר , כך שהמסך לא נתקע , המשתמש יכול להזיז את חלון התצוגה בזמן קבלת הנתונים , או לבצע פעולות נוסופות בזמן שמתבצעת גישה לשרת הפייסבוק.

השימוש נעשה בעזרת יצירת thread חדש , הפקודה invoke בעצם גורמת להפעלת המתודה בThread שבו הfetch נמצא כדי ליצור בעצם פעולות סינכרוניות .

ניתן למצוא את השימוש בעבודה אסינכרונית במקומות הבאים :

ב UI :

בכל אחד מהForm הבאים , כאשר מבצעים fetch למידע משרת הפייסבוק:

FormHomePage ,FormBestFriend ,FormMostAttractive.

**עבודה עם Data Binding:**

ניתן לראות את המימוש עבור – Two-Way Data binding במקומות הבאים:

FormEvents:

בForm זה מוצגים שלושה List Boxs שכל אחד מהם מתקשר עם פאנל שונה שבו יושב ה- Binding Source , ומקושר לרשימה שלו . (מדובר על הצגה של Events שמחולקים באמצעות צבעים – ירוק,צהוב,ואדום (רמזור קורונה) לפי מספר המשתתפים באירוע).

עבור כל אירוע שנמצא ברישמה , ניתן לראות בpanel :

* שם האירוע.
* מס' משתתפים.
* תיאור האירוע.
* תאריך התחלה של האירוע.
* תאריך הסיום של האירוע.

FromHomePage:

בForm זה נמצא ListBox של פוסטים , ומתחתיו יש פנאל שמוקשר אליו , עבור כל פוסט ברשימה ניתן לראות:

* תאריך של יצירת הפוסט.
* תאריך של עידכון הפוסט.

### תבנית מס' 1 – [Factory Method]

* סיבת הבחירה / שימוש בתבנית:

השתמשנו ב design pattern factory method על מנת ליצור מופע של משפחה פולימורפית היורשת מlistBox , משפחה זו עוזרת לנו בתיוג ואפיון הEvents לפי מספר המשתתפים.

הצורך בשיטה זו נבעה מהרצון לייחד כל List Box בצבע היחודי שלו , כדי שהשתמש יוכל להבחין בצורה קלה יותר בהבדלים.

* אופן המימוש:

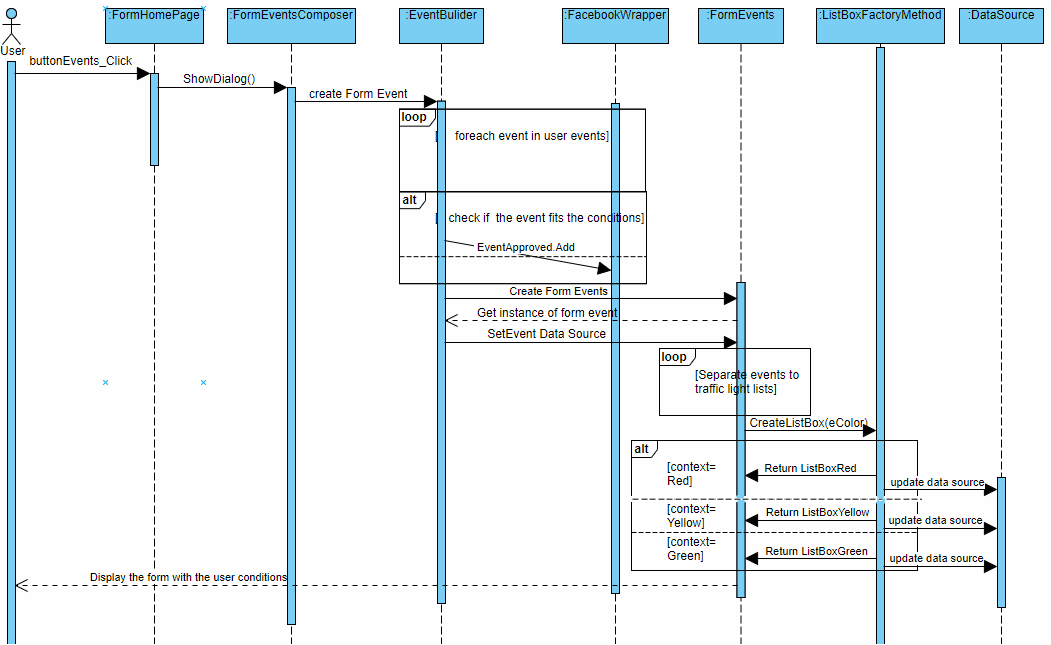
בטופס FormEvent עבור כל אירוע שעבר סינון ע"י ה- Client , הטופס FormEvent מייצר שלושה List Boxs על פי תוכנית רמזור הקורונה , כאשר כל אירוע משוייך לList Box עם הצבע המתאים , יצירת הListBox מתבצעת ע"י מחלקה יעודית ListBoxFactoryMethod

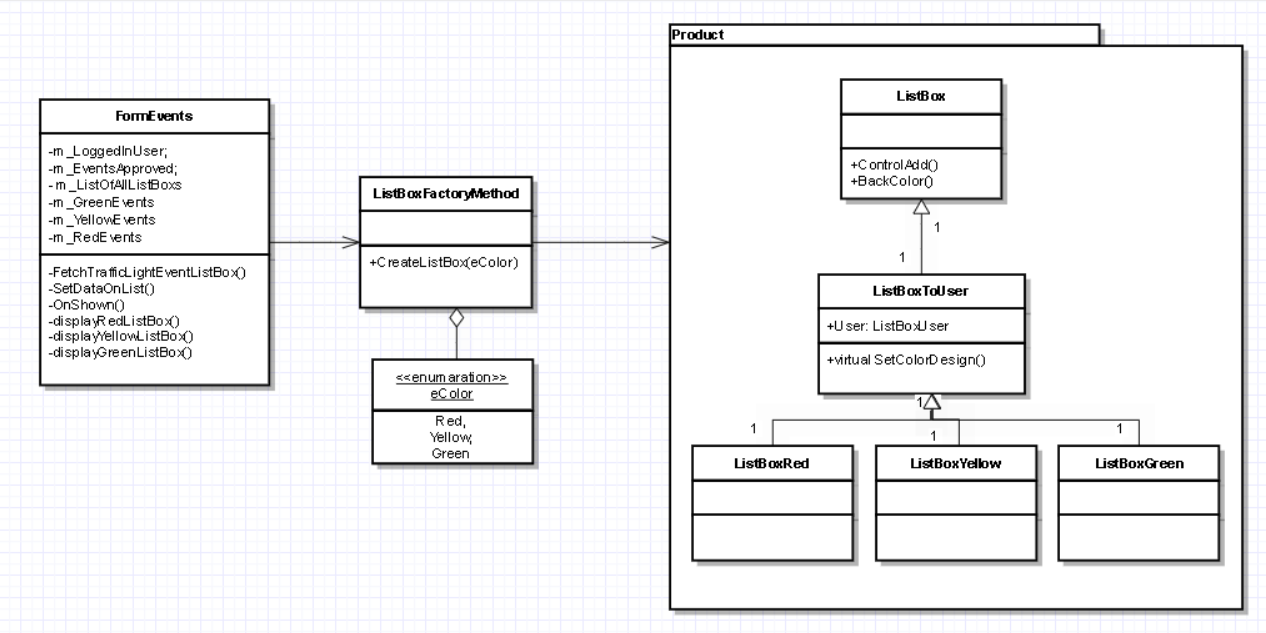
אשר מבצעת את הFactory Method.

רצינו ליצור שוני בין הList Boxs לפי צבעים של תוכנית רמזור הקורונה .

לאחר שיצרנו את הList Box זה (ListBoxToUser), שאר הList Boxs ירשו ממנו וישנו את הצבע של הListBox לפי מספר המשתתפים שבעזרתם הם הותאמו ע"י דריסת המתודה שאחראית על עיצוב הList Box שקיבלו בירושה.

המחלקות שמהוות את המשפחה הפולימורפית הן:

* ListBoxRed
* ListBoxGreen
* ListBoxYellow
* Sequence Diagram
* Class Diagram



### תבנית מס' 2 – [Pattern Builder]

* סיבת הבחירה / שימוש בתבנית:

הצורך בתבנית: החלטנו לממש את התבנית בטופס "אירועים" מהסיבה שאנו חיים בתקופת הקורונה, לאנשים חשוב להיות באירועים שלא מכילים מס' רב של משתתפים, בנוסף ניתנת למשתמש האפשרות לסנן אירועים שאינם רלוונטיים עבורו ע"פ תאריך.

* אופן המימוש:

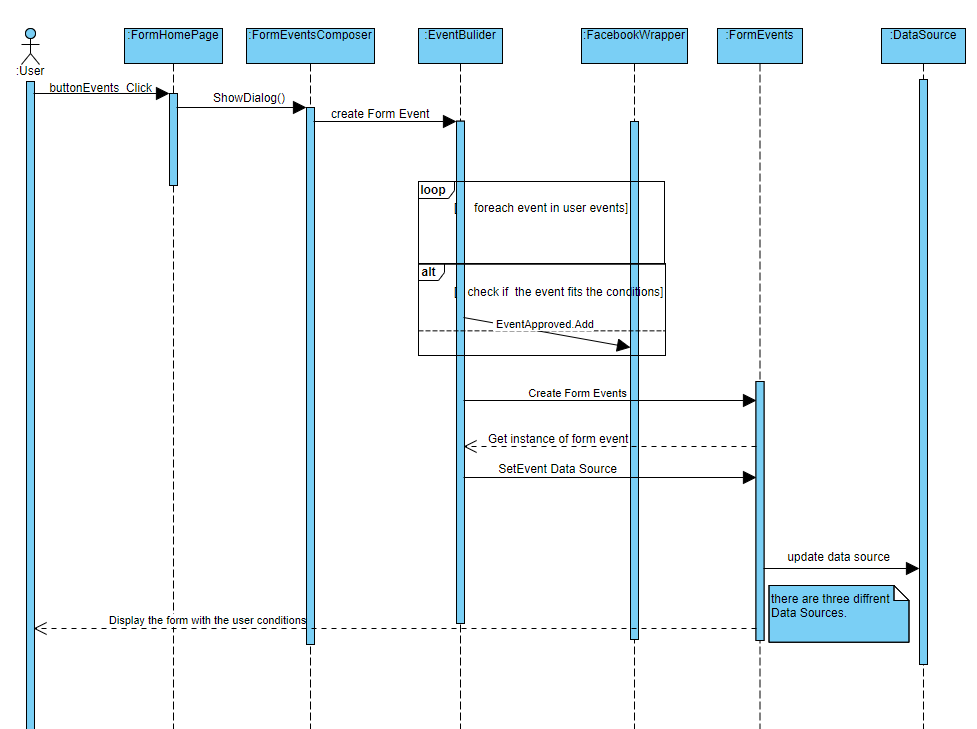
ה**Client -** הוא משתמש הפייסבוק היוזר, הוא מיוצג ע"י בחירה של אירועים בטופס הראשי – FormHomePage ,

ה-**Composer** שמקבל את הנתונים והדרישות מהלקוח, ממומש בטופסFormEventComposer .

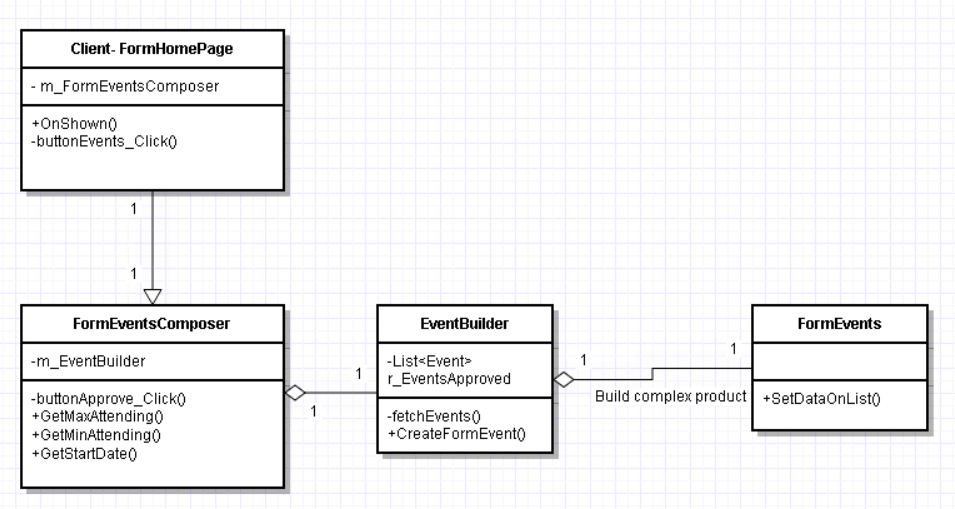
ה- **Builder** מקבל את הדרישות מהComposer, הוא זה שמבצע את הסינון ואורז הכל במקשה אחת, כל זאת מתרחש במחלקה EventBuilder

ה-**Complex Product** הוא המוצר המוגמר שמתקבל ע"י הBuilder ומיוצג בחלון – FormEvents.

* Sequence Diagram



* Class Diagram



### תבנית מס' 3 – [Cache Proxy]

* סיבת הבחירה / שימוש בתבנית:

מימשנו Proxy ש"מתחזה" לposts של המשתמש , ה-Proxy שומר אצלו את כל הפוסטים בפעם הראשונה , זאת על מנת למנוע פניות מרובות שוב ושוב לפייסבוק (במידה וכמות הפוסטים לא השתנתה). כך ניתן לעבד את המידע כמה שנדרש ולחסוך תעבורת מידע ברשת.

השתמשנו בתבנית זו , מכיוון שבפרוייקט שלנו , אנו ניגשים הרבה פעמים לפוסטים של המשתמש לצרכים שונים.

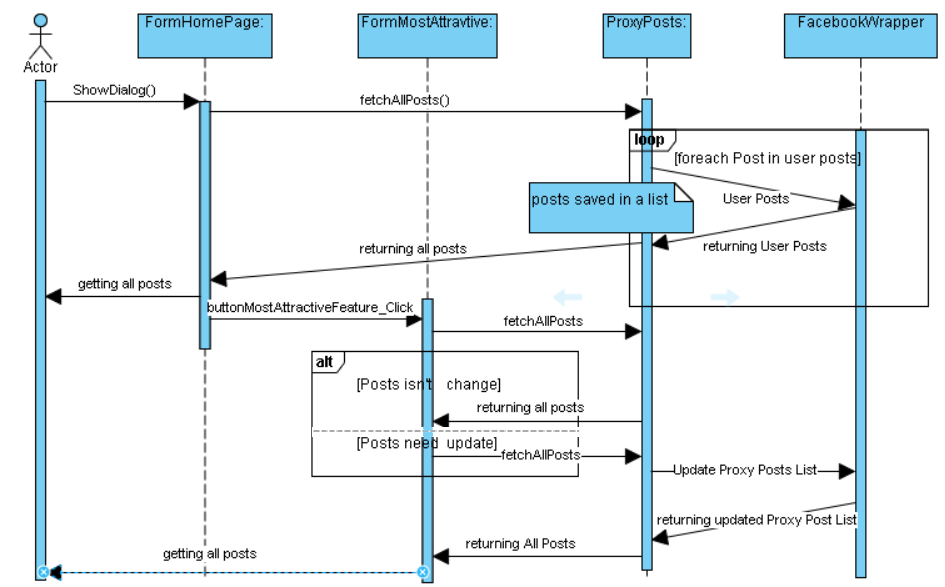
* אופן המימוש:

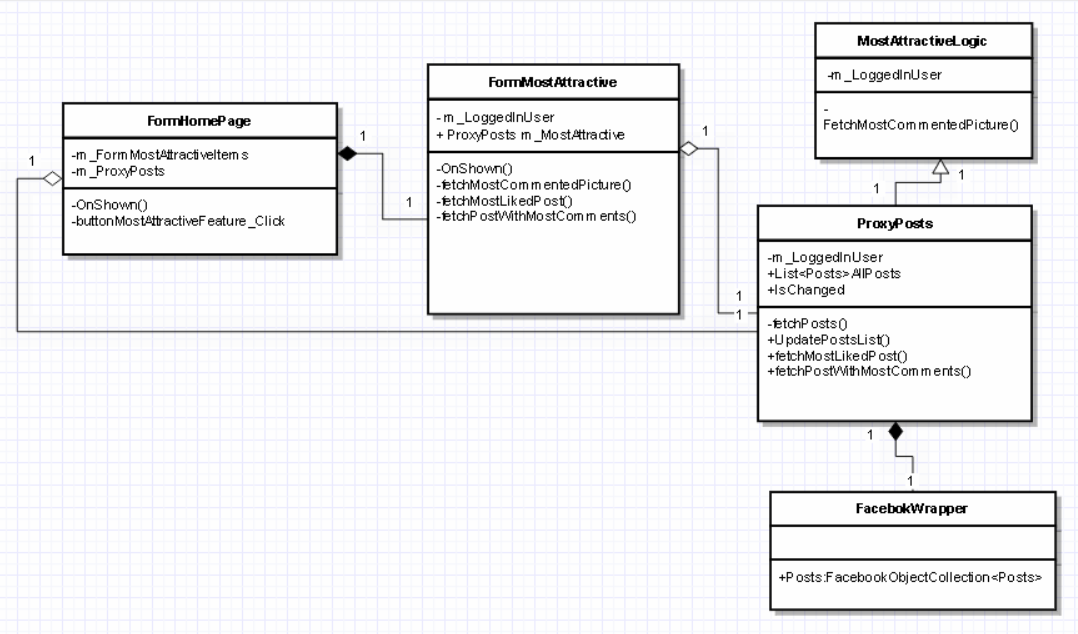
ברגע שנפתח הטופס FormHomePage לאחר ההתחברות , מתבצעת גישה דרך ה - ProxyPosts באמצעות הconstartor שלו , שם המידע נשאב לראשונה , ולאחר מכן בכל פעם שאנו רוצים לגשת לפוסטים , אנו ניגשים למתודה UpdatePostList שבודקת האם נוספו עוד פוסטים , ואם לא , היא מחזירה את רשימת הפוסטים שנשמרו.

המחלקה נמצאת בחלק הלוגי של המערכת כיוון שהיא לא מתייחסת לממשק המשתמש אלא מתעסקת בקוד המתבצע מאחורי הקלעים עבור כל ממשק שהוא.

דוגמה לטופס שמשתמש בProxyPost - FormMostAttracticve משתמש כדי למצוא את הפוסטים הכי אהובים .

* Sequence Diagram



* Class Diagram