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

* **"שמירה במועדפים"** – פיצ'ר זה מאפשר לשמור פוסטים שהמשתמש יצר.   
  הפוסט נכתב לקובץ JSON ורשימת הפוסטים השמורים נטענת ברגע שהאפליקציה עולה שוב.
* **סטטיסטיקות אישיות** – פיצ'ר זה מאפשר למשתמש באפליקציה להציג עליו פרטים וסטטיסטיקות כלפי השימוש שלו בפייסבוק, למשל הצגת מספר חברים, כמות פוסטים וכו במקום מרוכז אחד.

### תבנית מס' 1 – Static Factory Method

* סיבת הבחירה / שימוש בתבנית:  
  באפליקציה שלנו, יצרנו מספר מסכים(Forms) שונים בעזרתם אנו מבצעים הפרדה בין תצוגות שלא תלויות אחת בשנייה(למשל form ייעודי לרשימת החברים, וform ייעודי לרשימת האלבומים של המשתמש), וכן כל Form הינו בקובץ ייעודי ויחיד.  
  ניתן לשים לב כי כל הForms בהם אנו משתמשים יורשים ממחלקת Forms של System.Windows.Forms, ולכן החלטנו לממש את תבנית ה-Factory.  
  בזכות השימוש בFactory שייצרנו, הפרדנו את מלאכת הבחירה והיצירה מהקליינט, לתוך מחלקה ייעודית נפרדת. בעזרת תבנית זו, תהליך יצירת המסכים השונים באפליקציה נמצא במקום ריכוזי אחד, וכתוצאה מכך המערכת ניתנת לתחזוקה טובה יותר.
* אופן המימוש:  
  יצרנו מחלקה חדשה בשם FormFactory, ובה כתבנו את לוגיקת הבחירה והיצירה של הForm המתאים בהתאם למה שהמשתמש בחר. בתוך המחלקה יש פונקציה סטטית בשם CreateFormByType, שמקבלת enum של איזה form רוצים ליצור, ותפקידה הוא לייצר את הform המתאים ולהחזיר אותו.
* Diagram

  Description automatically generatedSequence Diagram
* Diagram

  Description automatically generatedClass Diagram

### תבנית מס' 2 – Singleton Facade\*

* סיבת הבחירה / שימוש בתבנית:  
  באפליקציה שלנו יש שימוש רחב במתודות המונגשות לנו על ידי ספריית   
  ה-FbGraphApiWrapper, החל מהתחברות לאפליקציה ועד לטעינת פוסטים ותמונות.   
  השימוש במתודות אלו נמצא בכל מסכי האפליקציה, וכתוצאה מכך נוצרה בעיה: מה יקרה אם בעתיד נצטרך לשנות את העבודה מול הספרייה של פייסבוק?  
  במידה וסיטואציה כזאת אכן מתממשת, אנו ניאלץ ללכת לכל המקומות בקוד בהן הפעלנו מתודות של הספרייה, ולעדכן שם את הקוד בהתאם. כלומר, יצרנו קוד שהוא tightly coupled, כלומר קוד בעל תלות גבוהה בספרייה של פייסבוק לכל אורך התוכנית,   
  ולכן זהו קוד לא תחזוקתי. בנוסף, קוד זה מממש אצלו ישירות פעולות לוגיות, דבר נוסף שאסור שייקרה כיוון שממשק המשתמש לא צריך להכיר את העולם הלוגי.  
  בעזרת השימוש בתבנית ה-Facade, אנו יכולים להגדיר מקום ריכוזי אחד דרכו אנו מתקשרים עם הספרייה של פייסבוק, וכתוצאה מכך אם בעתיד הספרייה של פייסבוק תשתנה ולמשל תנגיש נתונים בצורה שונה, נצטרך לשנות את התפעול והקריאות למתודות של הספרייה רק במקום אחד, ב-Facade.   
  כתוצאה מכך הקוד שלנו לאורך כל התוכנית יעבוד רק עם המתודות המונגשות על ידי ה-Facade, ולא ישירות מול הספרייה של פייסבוק.   
  בעזרת תבנית זו יצרנו קוד שהוא loosely coupled, כלומר קוד שקל לתחזק.   
  בנוסף, אנו משפרים את הקריאות של הקוד ויצרנו חלוקה נכונה למחלקות, בכך שלוגיקות מורכבות יושבות במקום אחד, ומסכי האפליקציה אינן צריכות לממש לוגיקות אלו אצלן.
* אופן המימוש:  
  יצרנו מחלקה חדשה בשם FacebookApiFacade, בתוכה אנו מממשים את כל הפעולות המתקשרות עם הספרייה של פייסבוק. בנוסף, במסכי האפליקציה אנו יוצרים מופעים**\*** של מחלקה זו ומפעילים בעת הצורך פונקציות של המחלקה.   
    
  **\*** לא ציינו במהלך הפירוט על אופן המימוש וסיבת הבחירה בתבנית על השימוש ב-Singleton, ואנו נפרט זאת כאן. אנו מאמינים כי מחלקת ה-Facade שתפקידה לרכז את כל התקשורת מול ספריית פייסבוק צריכה להיות סינגלטונית. אין היגיון בלממש יותר ממופע אחד של מחלקה זו, כיוון שבעזרת מופע אחד בלבד אנו יכולים לממש את כל התכונות הפונקציונאליות של ממשק המשתמש. כתוצאה מכך אנו חוסכים בזיכרון של מימוש של אובייקטים מיותרים מאותה מחלקה, ומונעים פוטנציאל של באגים. לכן אנו מגדירים במערכת מופע אחד ריכוזי של ה-Facade, וכל הForms מתקשרים רק איתו.
* Sequence Diagram

Diagram

Description automatically generated

* Class Diagram

Diagram

Description automatically generated

### תבנית מס' 3 – Caching Proxy

* סיבת הבחירה / שימוש בתבנית:  
  באפליקציה שלנו, אנו מנגישים למשתמש בעזרת הספרייה של פייסבוק פונקציונאליות כמו טעינת אלבומים או פוסטים שהמשתמש יצר. בכל פעם שאנו רוצים לראות, למשל, את הפוסטים שייצרנו, המערכת תבצע טעינה של כל הפוסטים שייצרנו אי פעם.   
  אולם נוצרה בעיה ביעילות של האפליקציה: בכל פעם שאנו רוצים לראות את הפוסטים שייצרנו, או את החברים שלנו, הקבוצות וכו', אנו נאלצים לטעון מהשרת של פייסבוק המון מידע אודות המשתמש שלנו, בכל פעם מחדש. כתוצאה מכך משתמשים באפליקציה בעלי אינטרנט איטי ייסבלו כתוצאה מטעינה שתיקח המון זמן כיוון ומדובר בהמון מידע לטעון כל פעם מחדש, ובנוסף אם השרתים של פייסבוק יפסיקו לעבוד, לא נוכל לעבוד עם האפליקציה שלנו, כיוון שאנו לא יכולים לגשת לשרתים של פייסבוק ולקחת משם מידע.   
  לכן החלטנו לממש את התבנית Caching Proxy בתוכנית שלנו.   
  בעזרת תבנית זו, אנו יכולים לייצר cache לנתונים שאנו לוקחים מפייסבוק, ולשמור אותם אצלנו כל עוד התוכנית רצה. כעת, בכל פעם שנרצה לטעון מידע אודות המשתמש, קודם כל נבדוק האם המידע כבר שמור אצלנו במערכת, ואם כן נציג אותו ישירות. אם לא, נבצע טעינה מהשרתים של פייסבוק ואז נשמור את המידע שקיבלנו אצלנו במערכת לשימוש חוזר בעתיד. כתוצאה מכך אנו "משלמים" בטעינת נתונים **באופן חד פעמי**, ונוכל להציג מידע למשתמש במהירות, ביעילות ומבלי טעינות נתונים מיותרות, ובכך שיפרנו את יעילות המערכת.
* אופן המימוש:  
  השינויים שהיינו צריכים לבצע על מנת להטמיע את ה-Caching Proxy נמצאים בקובץ FacebookApiFacade. יצרנו משתנה מקומי לכל אחד מהאובייקטים אותם אנחנו מעוניינים לשמור, ובעת הפעלה של המתודות במחלקה זו אנו בודקים האם המשתנה המקומי המתאים לא ריק. אם הוא לא ריק, אנחנו פשוט מחזיקים את מה שיש בו, ואם הוא ריק אנחנו מבצעים טעינה מהשרתים של פייסבוק, ובעזרת המידע שקיבלנו חזרה אנו מבצעים השמה למשתנה המקומי ואז מחזירים אותו.  
  למשל, כאשר המשתמש רוצה לראות את רשימת הפוסטים שלו, אנו בודקים האם רשימת הפוסטים ריקה. אם כן, אנו טוענים לתוכה את הנתונים מפייסבוק, ואז מחזירים אותה. בעתיד כשנרצה שוב לראות את הפוסטים שלנו, רשימת הפוסטים כבר לא ריקה ואנו נחזיר אותה במיידי.
* Diagram

  Description automatically generatedSequence Diagram
* Diagram, calendar

  Description automatically generatedClass Diagram

### עבודה עם DataBinding:

המחלקות אשר עשינו שימוש ביכולות ה-DataBinding הן:

1. FormEvents
2. FormFriends
3. FormGroups
4. FormLikedPages

בעת השימוש בDataSources הרלוונטיים ל-Forms אלו, ביצענו חיבור בין רשימת ה-ListBox-ים ל-BindingSource המתאימים בכל Form, ובנוסף בעת אירוע לחיצה על איבר ברשימה, הנתונים עליו נטענים אוטומטית לשדות שנוספו ל-Form בעת יצירת ה-DataSources.

### עבודה אסינכרונית:

המקומות בהם השתמשנו בתכנות אסינכרוני הם:

1. FormEvents
2. FormFriends
3. FormGroups
4. FormLikedPages

ב-Forms אלו אנו מבצעים טעינה של כמות נתונים גדולה מהשרתים של פייסבוק, ולכן אם לא נבצע את פעולות אלו ב-Thread נפרד האפליקציה תיתקע עד שכל הנתונים התקבלו בהצלחה.   
לכן היה צורך בהפרדת הלוגיקה הכבדה ל-Thread נפרד, כדי שהאפליקציה תוכל לרוץ מבלי להיתקע, בזמן שברקע המערכת טוענת את הנתונים מפייסבוק, ובכך למנוע תסכול של המשתמש.  
בנוסף, חשוב לציין כי בעזרת ה-Caching Proxy אנו מבצעים טעינה מהשרתים של פייסבוק רק פעם אחת כפי שהוסבר בסיבת הבחירה של תבנית זו, ולכן בפעמים העתידיות בהן נרצה לעבוד עם מידע שטענו ונמצא ב-Cache של האפליקציה, המידע ייטען ב-Thread נפרד והאפליקציה לא תיתקע גם אם פעולת הטעינה מה-Cache תהיה מהירה יותר.