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

* **Memories game:**

**פיצ'ר המאפשר למשתמש לשחק במשחק הזיכרון ברמות קושי שונות, כאשר הקלפים במשחק נמשכים באופן אקראי מאלבומי התמונות של המשתמש. הפיצ'ר נועד לספק למשתמש חווית משחק מהנה המשלבת זיכרונות נוסטלגיים מהעבר.**

* **Grouper:**

**פיצ'ר המאפשר למשתמש ליצור רשימות חברים (friend lists) חדשות על בסיס מכנה משותף שלו איתם, מתוך רשימת קטגוריות אפשריות קבועות מראש:**

1. **עובדים \ עבדו יחד.**
2. **לומדים \ למדו יחד.**
3. **נמצאים באותו סטטוס רומנטי.**
4. **אוהדים את אותה קבוצה \ קבוצות.**
5. **גרים באותה העיר.**
6. **בני אותו שכבת גיל.**

**הפיצ'ר יציג למשתמש את רשימת חבריו המתאימים לקטגוריה הנבחרה, וכל מה שנותר יהיה למשתמש לעשות הוא לבחור מתוך הרשימה מי מהחברים הוא רוצה לצרף ל"friend list" וללחוץ על כפתור יצירת הרשימה, ורשימה בעלת שם מתאים תיווצר עבורו.**

### תבנית מס' 1 – Singleton

* **סיבת הבחירה / שימוש בתבנית:**

בפרויקט שלנו המחלקה שמממשת **singleton** היא מחלקה הנקראת ‘***FaceBookUserManager’****.* תפקיד המחלקה הוא להחזיק את פרטי המשתמש המחובר, לבצע את לוגיקת ההתחברות וההתנתקות מהמערכת, ולשמור את הגדרות האפליקציה. פרטי המשתמש וההגדרות הם יחידים לאפליקציה, ואין שום היגיון בהחזקת מספר משתמשים פעיליםבמקביל. בנוסף השימוש במחלקה הוא תדיר מפני שכמעט בכל רכיב באפליקציה יש צורך במשיכת מידע אודות המשתמש, ובבדיקה אם מחובר אחד כזה. כל אלו נשמרים ב‘***FaceBookUserManager’****,* ולכן היינו רוצים שמופע המחלקה היחיד שמחזיק את המידע הזה יהיה נגיש למשיכה בכל רכיב במערכת מיוזמתו, מבלי צורך לכפות תלויות בין רכיבים במערכת.

בגלל כל הסיבות הנ"ל בחרנו לממש את ‘***FaceBookUserManager’*** כ-**Singleton** שכן תבנית זו פותרת בדיוק את הבעיות ההנדסיות שציינו.

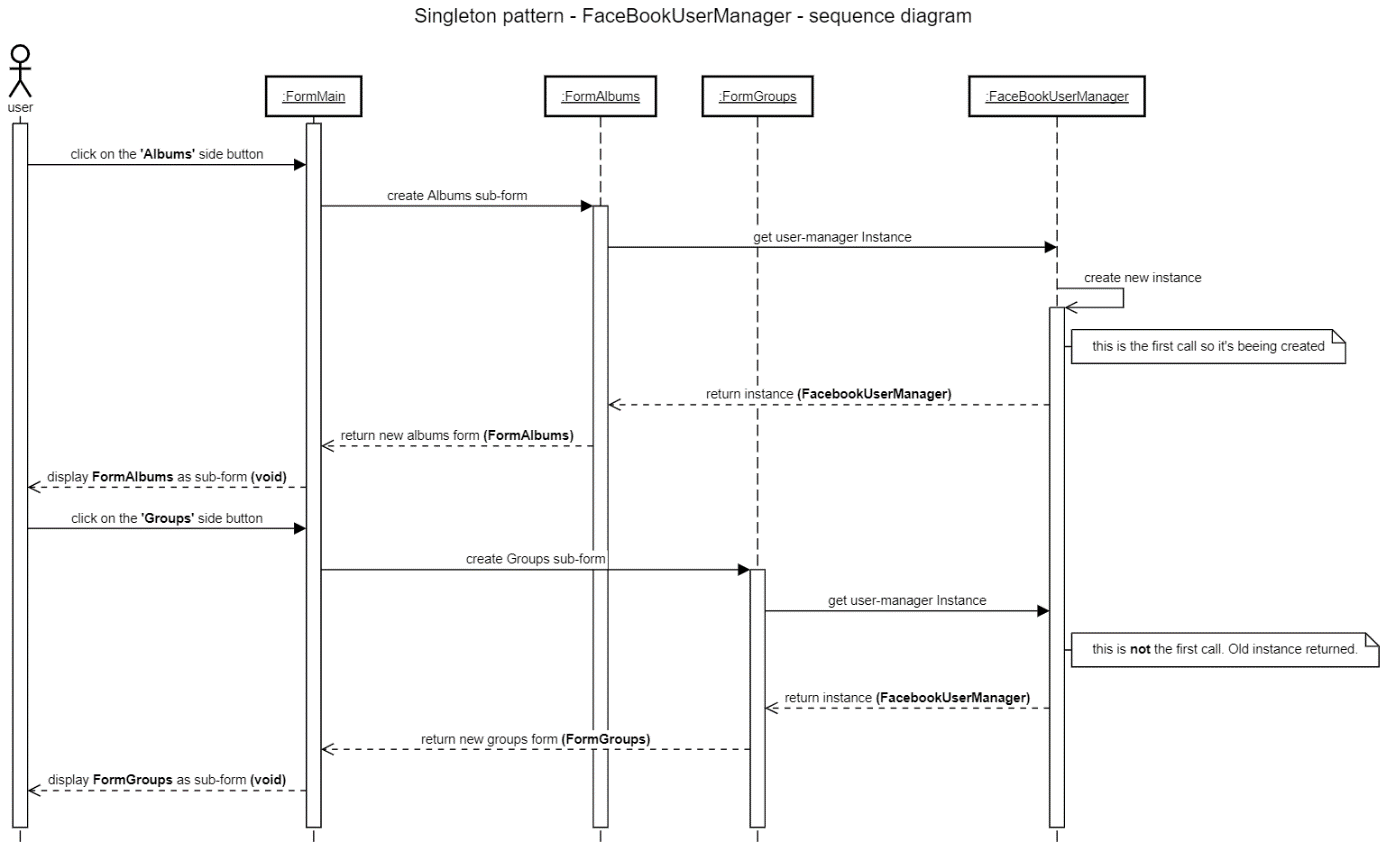
* **אופן המימוש:**

המחלקה ‘***FaceBookUserManager’*** מממשת את תבנית **singleton** באופן הבא:

* המחלקה ממומשת כ - **sealed class**.
* בנאי המחלקה הוא .**private**
* המחלקה מחזיקה **public static property** בשם ‘***instance’*** המחזיק את המופע היחיד של המחלקה. ‘***instance’*** מאותחל להיות **null**, ונוצר באופן ‘עצלן’, כלומר נוצר רק בזמן הבקשה הראשונה. תהליך היצירה מוגן משגיאות היכולות להיווצר כתוצאה מפניות מרובות מתהליכונים שונים באמצעות מנגנון **double** **check lock**.

ניתן למצוא את מימוש התבנית בקוד במחלקה ‘***FaceBookUserManager’*** שבפרויקט.‘***FacebookAppWinformsUI’*** ואת השימוש בה במחלקות השונות באותו הפרויקט תחת התיקיה **SubForms**.

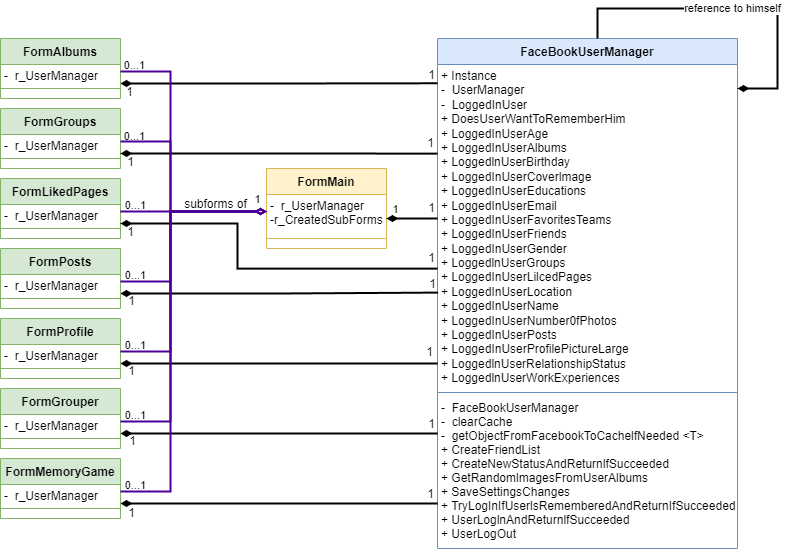
* **Sequence Diagram:**



* **Class Diagram:**

כאמור ‘***FaceBookUserManager***’ היא המחלקה המממשת את התבנית **singleton**.

הטפסים השונים משמשים כ**’client’**, והם אלו שמשתמשים במופע היחיד של ‘***FaceBookUserManager***’.



### תבנית מס' 2 – Factory method

* **סיבת הבחירה / שימוש בתבנית:**

השתמשנו בתבנית ‘**Factory method’** על מנת ליצור את ה-sub forms השונים.

כל תתי הטפסים הם פולימורמים (מעצם ירושתם מ***’Form’***), ונתקלנו בצורך ליצור בכל פעם form אחר, בהתאם לבחירת המשתמש. כמו כן רצינו שתהליך יצירת הטפסים יתבצע במקום מרוכז, בצורה שתתמוך באופן מירבי בהוספה עתידית של טופס נוסף בצורה נוחה.

בגלל כל הסיבות הנ"ל בחרנו לממש את תהליך יצירת הקבצים בעזרת ‘**Factory method’** שכן תבנית זו פותרת בדיוק את הבעיות ההנדסיות שציינו.

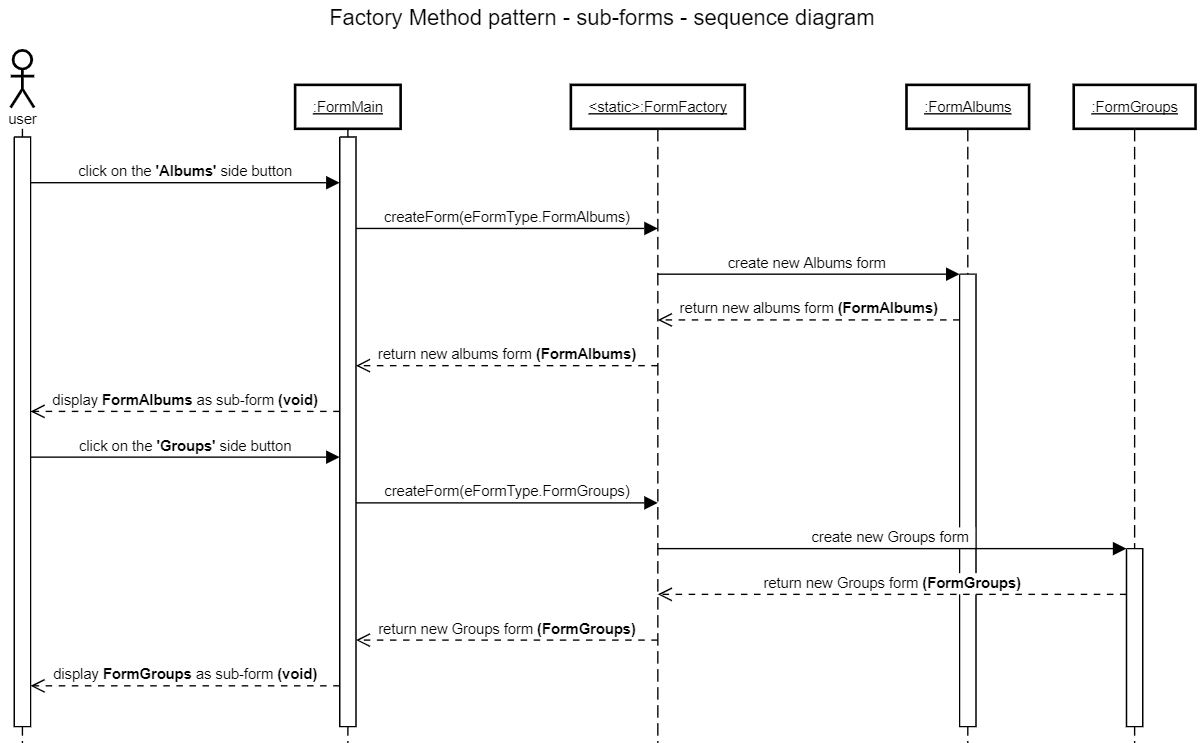
* **אופן המימוש:**

תהליך יצירת תתי הטפסים נעזר בתבנית ‘**Factory method’** באופן הבא:

* יצרנו מחלקה סטטית ‘***FormFactory’***. המחלקה משמשת בתור המפעל בתבנית, כלומר, הרכיב שיוצר את הטפסים השונים ומחזיר אותם לclient. המחלקה מחזיקה **Enum** בשם ‘***eFormType’***, שם שמורים סוגי הטפסים אותם המפעל יכול לייצר, ומטודה סטטית בשם ‘***CreateForm***’ שמקבלת ערך מ‘***eFormType’,*** ומחזירה מופע חדש מסוג הטופס המתאים.
* ***ה’MainForm’*** משמש כclient, ומבקש מהמפעל ליצור את הטופס המתאים בהתאם לבקשת המשתמש.

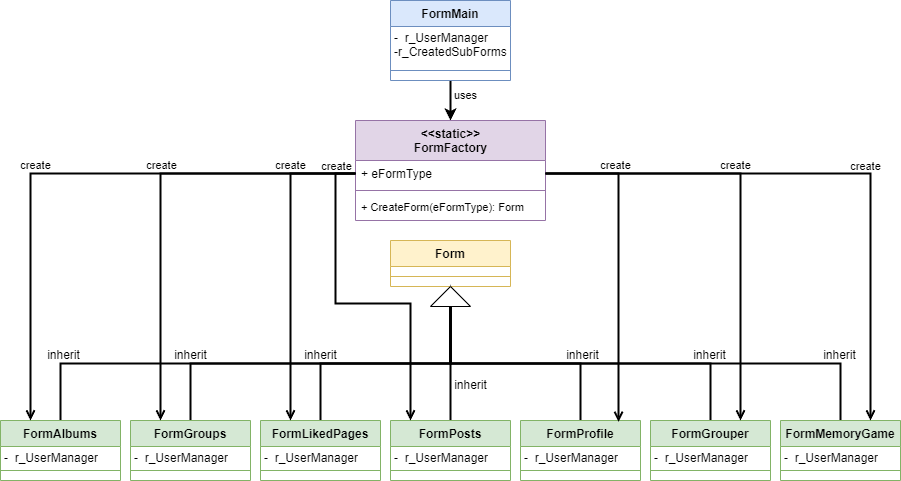
ניתן למצוא את מימוש התבנית בקוד במחלקה ‘***FormFactory’***שבפרויקט.‘***FacebookAppWinformsUI’*** ואת השימוש בה במחלקה ‘FormMain’ באותו הפרויקט.

* **Sequence Diagram:**



* **Class Diagram:**

כאמור ‘***FormFactory’*** היא המחלקה שמשמשת כ’מפעל’, הטפסים השונים הם המשפחה הפולימורפית, והטופס הראשי הוא הקליינט.



### תבנית מס' 3 – Caching Proxy

* **סיבת הבחירה / שימוש בתבנית:**

בחרנו להשתמש במחלקה ‘***FaceBookUserManager***’ בתור **Caching Proxy** השומר את המידע על המשתמש המחובר, מידע הנמצא בשרת הפייסבוק.

בנקודות רבות בפרויקט אנו צריכים לפנות לשרת הפייסבוק על מנת לקבל מידע על הuser, ופעולה זאת לוקחת המון זמן. רצינו להיות מסוגלים לשמור את המידע המבוקש בפעם הראשונה בה אנו פונים לשרת (on first demand), ובכך לשפר את ביצועי המערכת מבחינת זמנים.

כמו כן, לפעמים רכיבים שונים צריכים לקבל את אותם הפרטים על המשתמש ורצינו למנוע שכפול של אותם פרטים, בכך שנשמור אותם במקום אחד הנגיש לכלל הרכיבים ובכך לשפר את ביצועי המערכת בכל הנוגע לזיכרון.

בגלל כל הסיבות הנ"ל בחרנו לממש את ‘***FaceBookUserManager’*** כ- **Caching Proxy**שכן תבנית זו פותרת בדיוק את הבעיות ההנדסיות שציינו.

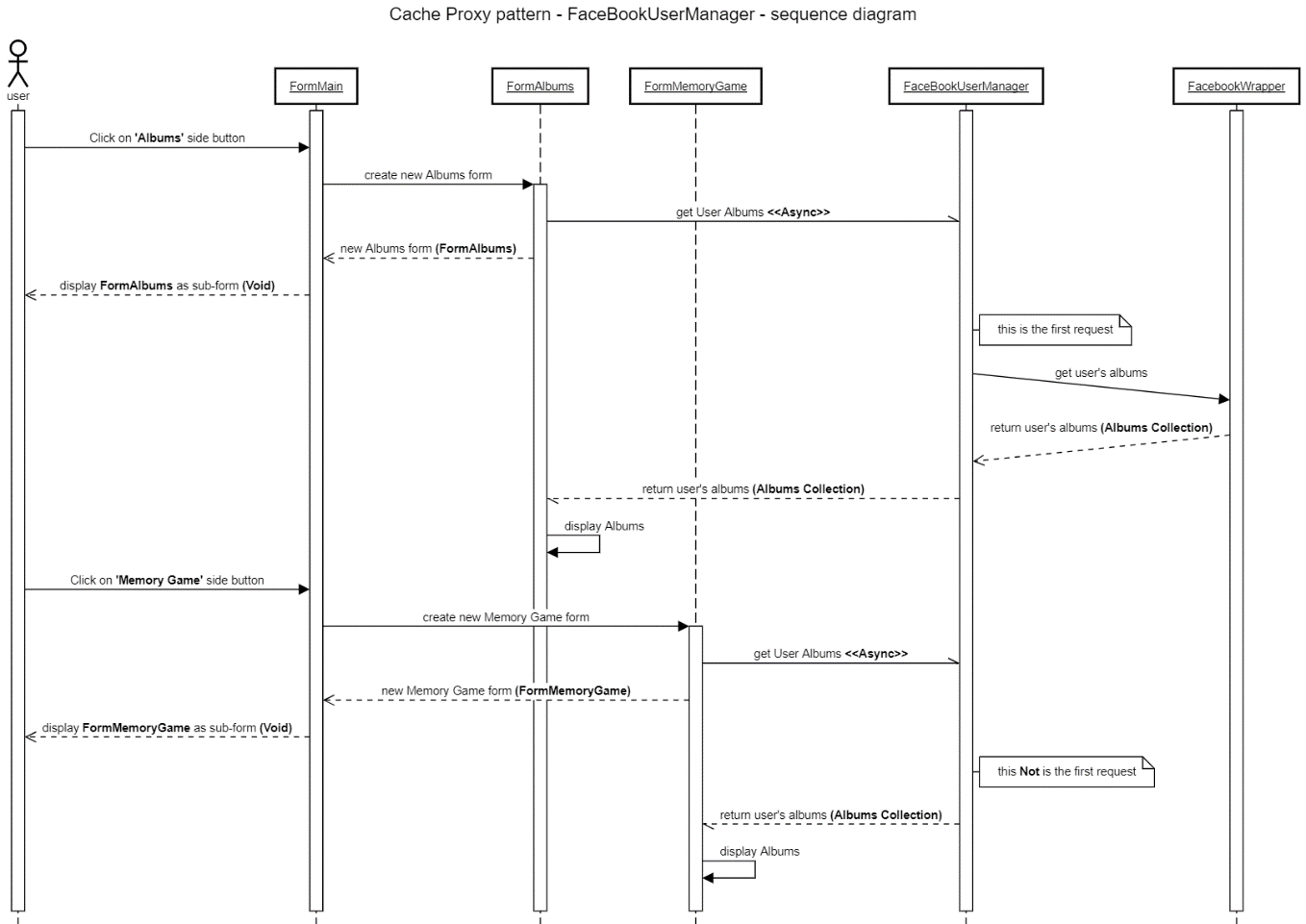
* **אופן המימוש:**

המחלקה ‘***FaceBookUserManager’*** מממשת את תבנית **Caching Proxy** באופן הבא:

* המחלקה מחזיקה **fields Privet** של פרטים רלוונטיים על המשתמש המחובר, המאותחלים לnull. כמו כן המחלקה מנגישה **Public properties** (עם Privet sets) דרכם הclient יכול לקבל את אותם הפרטים על המשתמש.
* במידה וזוהי הבקשה הראשונה לקבלת פרט כלשהו על המשתמש (כלומר השדה המתאים הוא null) המחלקה תיגש לשרת הפייסבוק (דרך***’FacebookWrapper’*** ), תמשוך את המידע ותחזיר אותו לclient. החל מהפניה השנייה לקבלת המידע הנ"ל המחלקה תחזיר את הערך השמור בה בלי לפנות לשרת.
* במידה והמשתמש מתנתק מהמערכת המחלקה מפעילה את הפקודה ‘***ClearCache’*** המאתחלת את כל השדות לnull.

ניתן למצוא את מימוש התבנית בקוד במחלקה ‘***FaceBookUserManager’*** שבפרויקט.‘***FacebookAppWinformsUI’*** ואת השימוש בה במחלקות השונות באותו הפרויקט תחת התיקיה **SubForms**.

* **Sequence Diagram:**



* **Class Diagram:**

כאמור ‘***FaceBookUserManager***’ היא המחלקה המממשת את התבנית **Cache Proxy**.

הטפסים השונים משמשים כ**’client’**, והם אלו שפונים לProxy במקום לפנות לשרת – כלומר ל ‘***FacebookWrapper***’.



### עבודה אסינכרונית

בכל הטפסים, כאשר יש צורך לפנות לשרת להביא מידע על המשתמש (פעולה שלוקחת זמן), הבקשה מתבצעת בעזרת thread נוסף. בעזרת השימוש בthreads הנוספים, המסך נשאר רספונסיבי לאורך כל השימוש (הthrad- הראשי לא נתקע בפעולות שלוקחות הרבה זמן), והמידע מגיע מהשרת באופן הדרגתי. בצורה זו המשתמש באפליקציה יכול ליהנות מממשק ללא תקיעות, כאשר הוא רואה את ההתקדמות של הפעולה שביקש בעיניים.

ניתן לראות שימוש בעבודה אסינכרונית בקוד במחלקות הבאות:

***FormMain, FormAlbums, FormGroups, FormLikedPages, FormPosts, FormProfile, FormMemoryGame.***

### עבודה עם Data Binding

קישרנו באמצעות Data Binding חד כיווני על מנת להציג מידע על הGroup הנבחר ועל הLiked Page הנבחר מהשרת. ניתן לראות שימוש ב-Data Binding במחלקות הבאות:

***FormGroups, FormLikedPages.***