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

* **חברים רווקים:**

הצגת כל החברים הרווקים שלי ברשימה ובנוסף – תמונה, גיל, עיר מגורים פלוס הצגת מפה של google maps שמציגה את העיר.

* **ימי הולדת:**

מציג רשימה של כלל החברים שיש להם יום הולדת היום עם אפשרות לכתוב ברכת יום הולדת שתשלח לכולם בבת אחת.

### תבנית מס' 1 – Factory method

* סיבת הבחירה / שימוש בתבנית:

ה-form הראשי של המערכת (facebookForm) מייצר formים שמשמשים כ"מסכים מתחלפים" בהתאם לבחירת המשתמש באופן שבו בכל פעם שנלחצת אחת האפשרויות נוצר form חדש.

מספר הformים הוא תמיד סופי ולכן הוגדר בתוך eNum

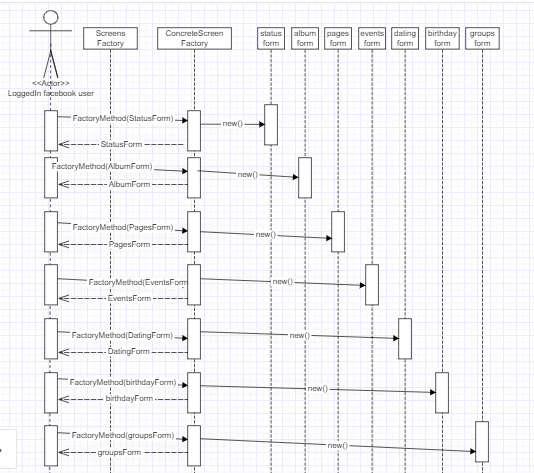
זיהינו צורך בשימוש בקלאס שיהיה אחראי על היצירה של ה-formים האלו, התמודדות עם האקספשנים שיצוצו וכן שילוב תתי המערכות והפרמטרים להם הפורמים זקוקים.

* אופן המימוש:

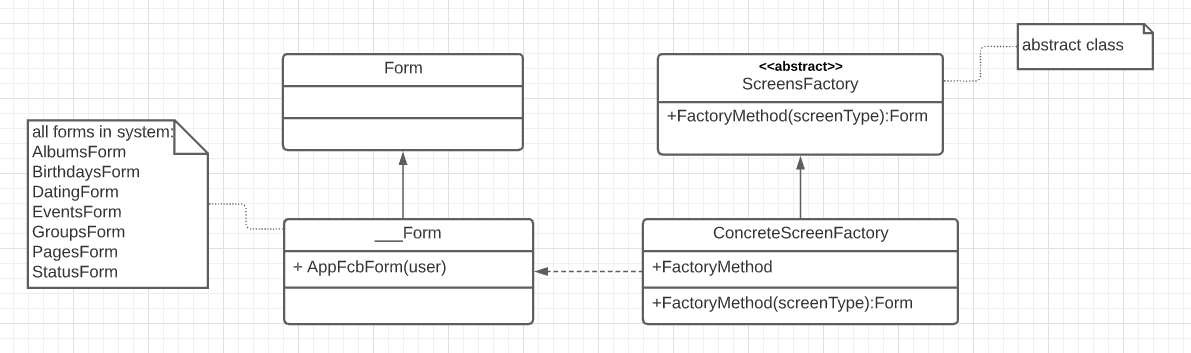
הפטרן ממומש בשני קבצים – האחד ScreensFactory שמהווה מחלקה אבסטרקטית בעלת מטודת factoryMethode שמקבלת סוג פורם ליצירה וUSER והשני ConcreteScreenFactory שיורש מהמחלקה האבסטרקית (נועד לתחזוקתיות עתידית) ומממש את המתודה.

בתוך המטודה יש switch case שמתאים את אופן הבניה שלו בהתאם לenum שקיבל.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Proxy

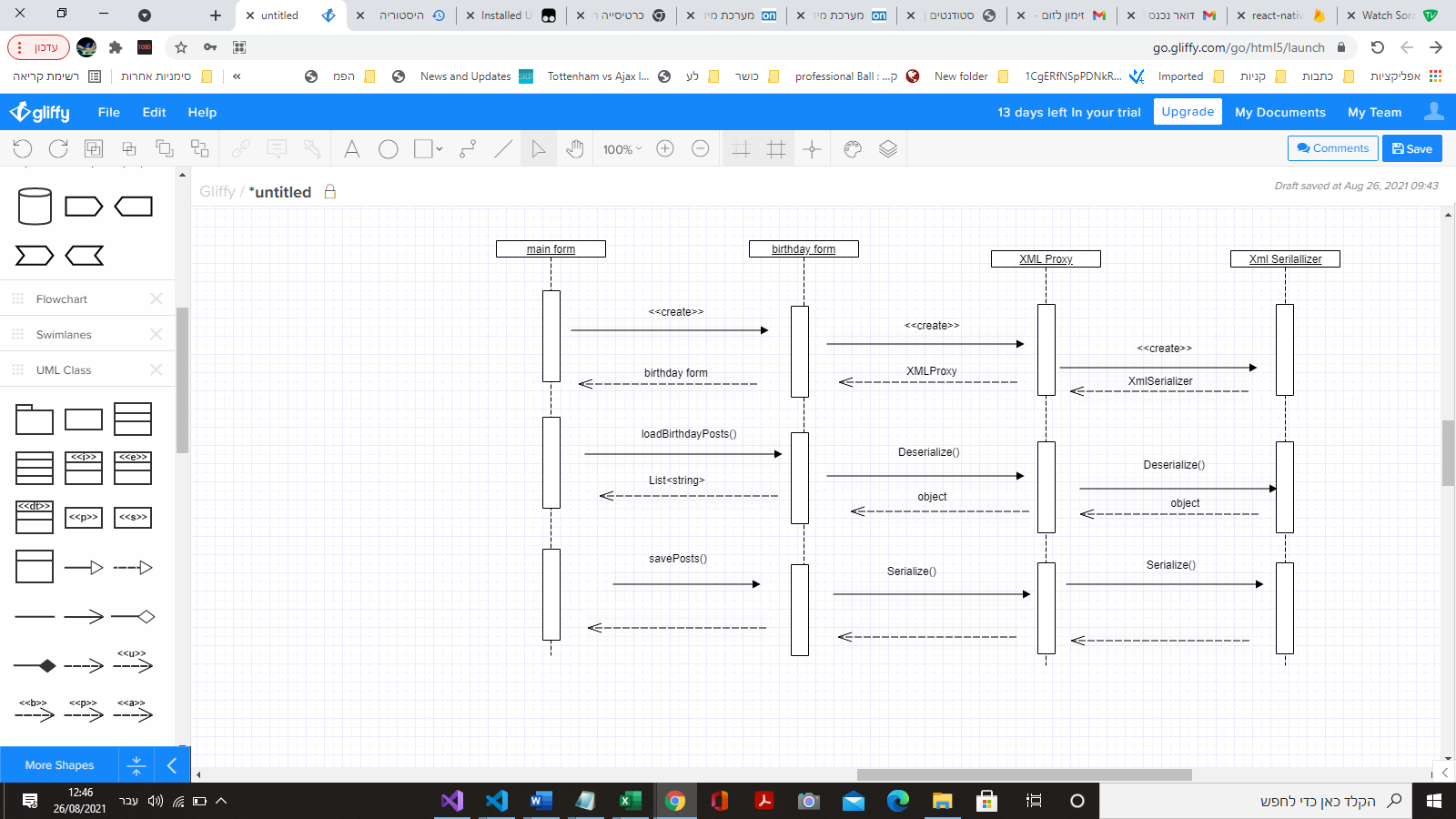
* סיבת הבחירה / שימוש בתבנית:

אנו שומרים את היסטוריית הפוסטים שאדם שולח עבור יום הולדת לחבריו דרך הפיצ'ר שעשינו בקובץ XML עי שימוש בXmlSerializer שהינו רכיב Legacy ויכול לזרוק שגיאות במהלך העבודה.  
שימוש בPROXY ימנע מהמשתמש לטפל בשגיאות תוך חשיפה לממשק שנראה כמו XmlSerializer

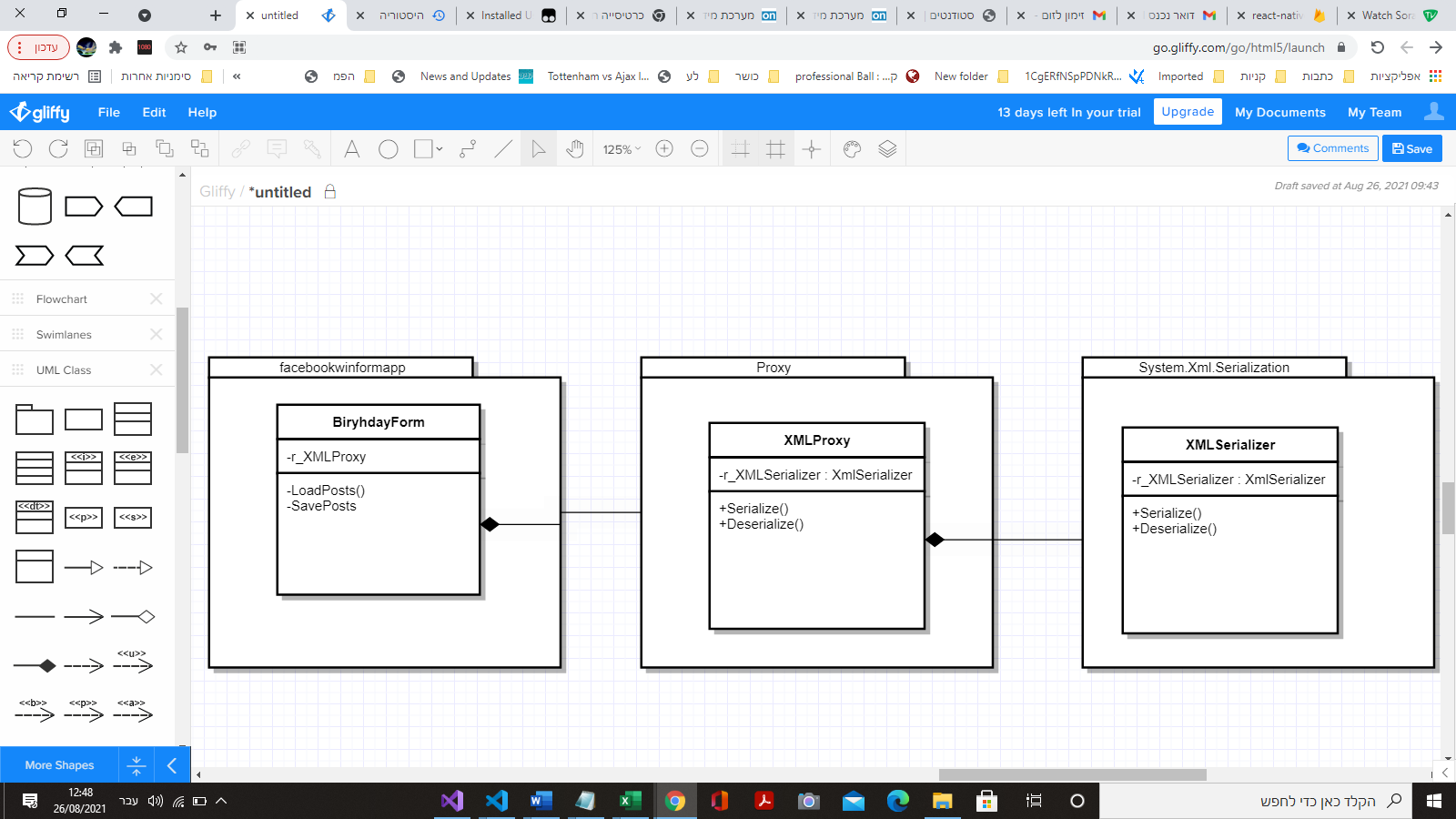
* אופן המימוש:

ממשנו מחלקה חיצונית בשם XMLProxy שחושפת את המתודות ששייכות ל XmlSerializer- Serialize וDeserialize ולכן הטיפול בשגיאות מתבצע בתוך המחלקה והמחלקה מחזירה למשתמש את המידע הרלוונטי בהינתן כישלון או הצלחה של הבקשה.

* Sequence Diagram



* Class Diagram



1.המקביל ל"Client" היא "BirthdayForm".

2.המקביל ל"Proxy" היא "XMLProxy".

3.המקביל ל"Real Subject" היא "XmlSerializer".

### תבנית מס' 3 – Facade

* סיבת הבחירה / שימוש בתבנית:

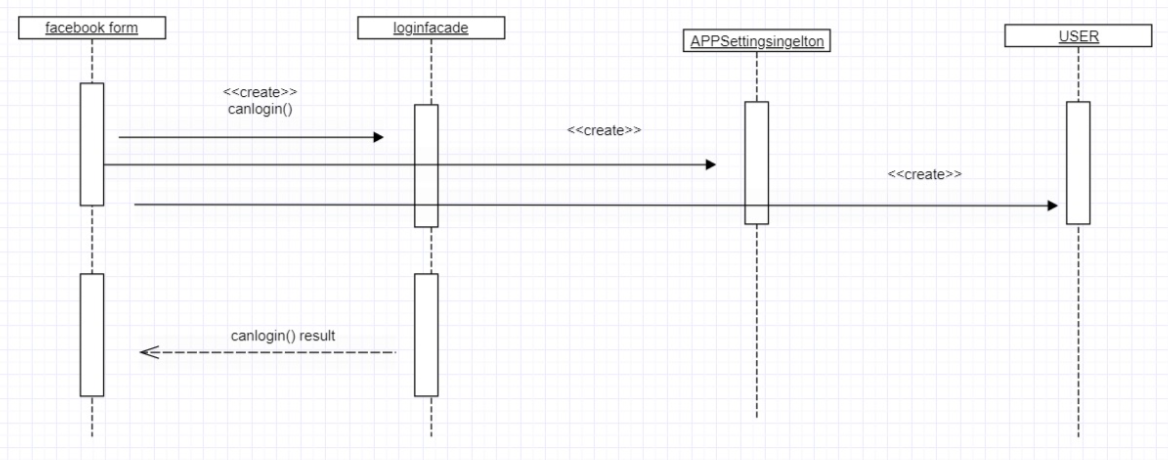
1. המערכת הלוגית המשמשת את אפליקציה זו מורכבת מכמה מחלקות. בכדי לפשט את העבודה אל מול הלוגיקה יצרנו ממשק יחיד העובד עם כמה ממשקים שונים.
2. בכדי למנוע שימוש לא נכון במחלקות הלוגיות השונות הגדרנו את ממשק זה המונע מהמשתמש בו סרבול וטעויות.
3. שימוש בfacade תורם לשימוש חוזר ותחזוקתיות בכך שהfacade חושף ממשק משתמש ומאפשר שינוי/הוספה של רכיבים לוגיים לממשקים השונים.

* אופן המימוש:

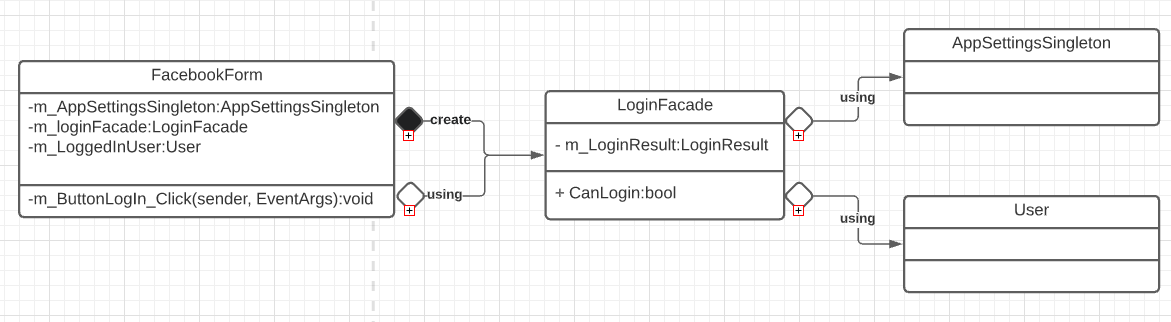
המחלקה FacadeLogin מהווה את הfacade האחראי על מסך ה-log in באפליקציה שלנו.

החלטנו לממש את FacadeLogin כfacade לא אטום מכיוון שרצינו לאפשר למשתמש גישה למידעים ספציפיים בנוסף לגישה דרך Logic בה מקבל את כלל המידעים בצורה מרוכזת.

* Sequence Diagram



* Class Diagram



• המקבילות ל "Client" היא “facebookForm”.

• המקבילה ל"Façade" היא “LoginFacade”

• המקבילות ל"SubsystemA" ו"SubsystemB" בהקבלה הן "APPSETTING" ו"USER".

### עבודה אסינכרונית:

המקומות בהם השתמשנו בתכנות אסינכרוני הם:

**StatusForm.fetchPosts()**

**fetchAlbums.Start()**

**fetchLikedPages.Start()**

פעולות אלו דורשות הבאת מספר רב של נתונים משרת הפייסבוק באופן שמורגש ע"י המשתמש ולכן מופעלות בThread נפרד.

המתודות לעיל מופעלות ע"י Threadים שונים ובתורן צריכות לעדכן ו/או להוסיף פקדים לפאנלים ו/או לאובייקטים שנוצרו קודם לכן ע"י ה- Main-Thread.

המתודה Invoke() , היא מתודה שמשתמשת ב-Delegate ששמו Action על מנת להפעיל את אותן מתודות האחראיות על תחזוקה קונקרטית עבור ה-Thread הנוכחי.

מתודה זו מכילה את קטע הקוד הדרוש ע"י Lambda Expression.

כל זאת כיוון שרק ה-Thread שעידכן/הוסיף את האובייקטים רשאי לשנות/לעדכן אותם.

### עבודה עם DataBinding:

המקומות בהם השתמשנו ב-DataBinding הם:

**AlbumsForm**

**EventsForm**

**BirthdaysForm**

**PagesForm**

**StatusForm**

ה- DataBindings לעיל נועדו על מנת להציג חיבור בין data לבין פקדים ב-UI באמצעות BindingSource רלוונטים. ה- BindingSources הללו, ברגע שמכילים reference לקולקציה של נתונים/לאוסף נתונים, בכל רגע נתון הם מחזיקים אחד כזה וקוראים לו Current.

ה-Current של אותו BindingSource הוא זה שהנתונים שלו יוצגו בפקדים ב-UI.

ברגע שנבחר איזשהו Item ב-listBox כלשהו ויהיה איזשהו אירוע של SelectedIndexChanged, אז ה-listBox ילך ל-DataSource ויודיע לו שה-Current שלו השתנה (של ה-DataSource), וברגע שה-Current השתנה הפקדים יאוכלסו בנתונים של Current.

מאחורי הקלעים הקוד ממומש ע"י המתודה InitializeComponent(), בעזרת Reflection.

בנוסף, מאחורי הקלעים ה- Visual Designer שלנו מאפשר לנו לשלוט על לא מעט קונפיגורציות של ה-DataBinding.