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

•FrienDiscover - האופציה להכיר חברים חדשים דרך חברים קיימים:

המערכת תגריל חבר קיים של המשתמש הרשום, ומתוך רשימת החברים של החבר שהוגרל נגריל משתמש זר למשתמש הרשום.

במסך האפליקציה יופיעו פרטיו של המשתמש השני שהוגרל, הכוללים: תמונת פרופיל, גיל ואודות.

המשתמש הרשום יכול לבחור האם לבקר בפרופיל של המשתמש המוגרל השני במטרה להוסיפו כחבר או לבצע הגרלה מחדש לפגישת משתמש אחר.

ניתן למצוא פיצ'ר זה ב- FrienDiscoverForm.cs.

•סטטיסטיקה על אופן השימוש של המשתמש הרשום בפייסבוק:

המערכת מציעה שני סוגי סטטיסטיקות להצגה:

1. התפלגות המדינות בהם המשתמש ביצע checkin.

2. התפלגות תיוג החברים של המשתמש בפוסטים שלו.

המידע מוצג ע"י תרשימי עוגה שמראים את התפלגות הנתונים השונים.

בנוסף, ניתן להוסיף עוד סטטיסטיקות בעתיד, וזאת הודות לרכיב התכנות האחראי על מילוי הנתונים בתרשים העוגה - PieChartDataLoadingComponent.cs.

ניתן למצוא פיצ'ר זה ב- StatisticsForm.cs.

### תבנית מס' 1 – [Factory Method]

* סיבת הבחירה / שימוש בתבנית:

בפרויקט נעשה שימוש במחלקות המבצעות טעינה דומה של נתונים מסוגים שונים (למשל תמונות, טקסט ונתוני תרשים עוגה).

החלטנו להשתמש בתבנית העיצוב Factory method כדי לעבוד עם רכיב יחיד המבצע את יצירת מופעי המחלקות השונות, ורק דרכו ניתן לקבל את המופעים שנוצרו.

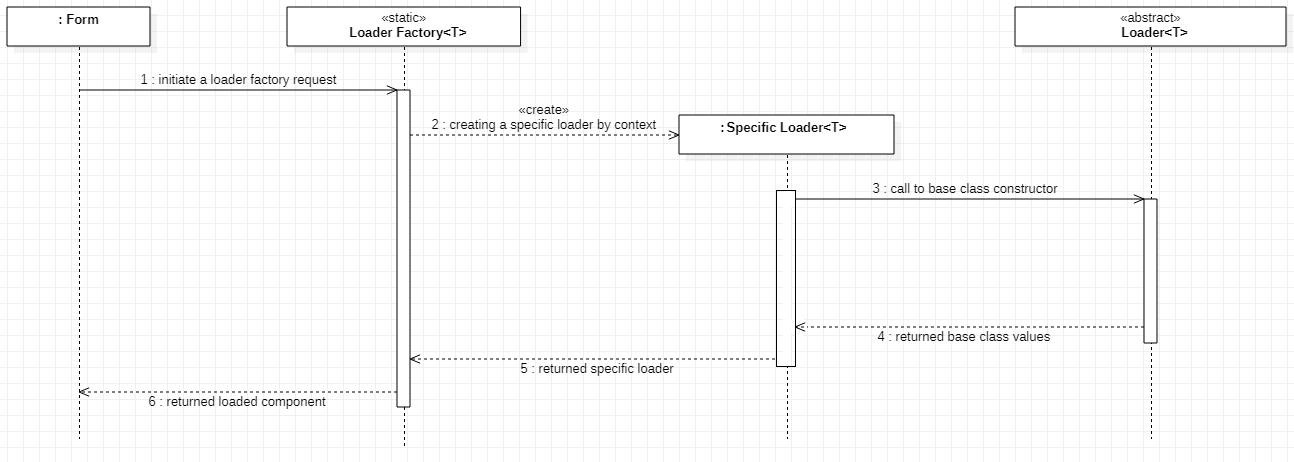
* אופן המימוש:

בגלל שמדובר במחלקות המבצעות פעולה דומה של טעינת נתונים שונים, הוחלט ליצור מחלקת בסיס אבסטרקטית בשם Loader שממנה ירשו מחלקות הטעינה השונות.

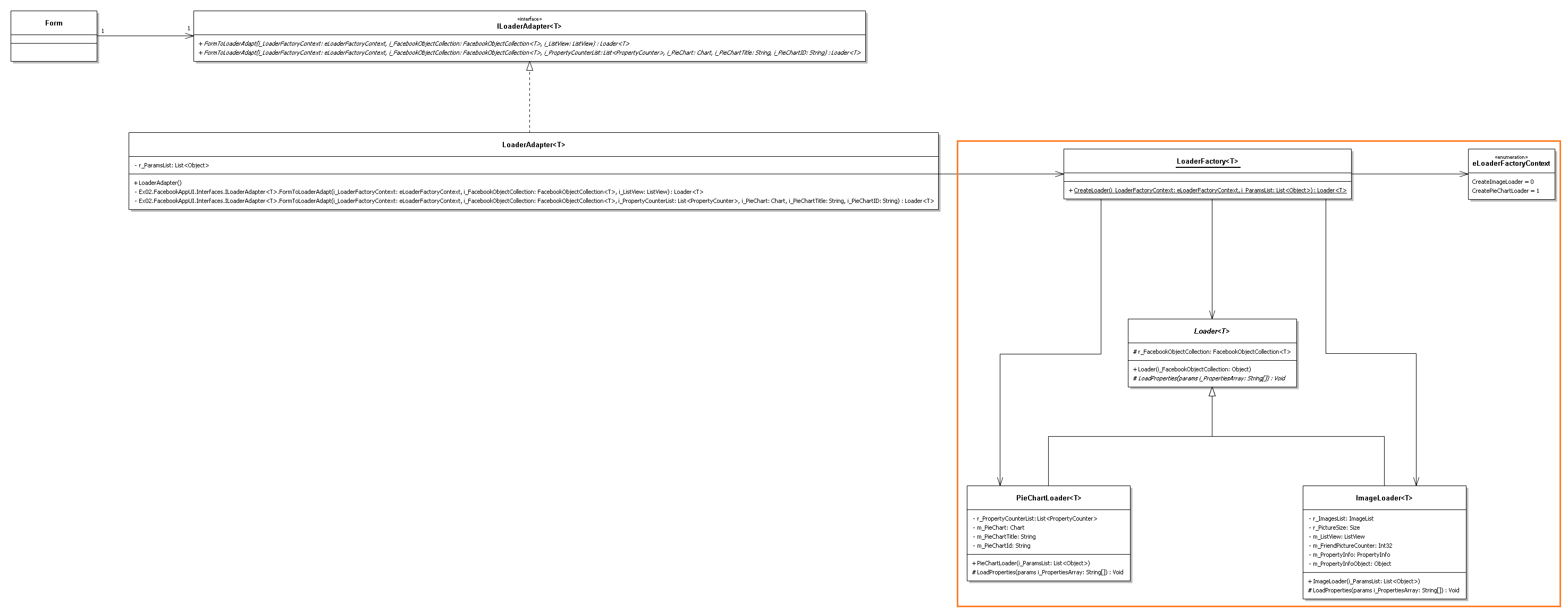
החלטנו לממש את Factory method כמחלקה סטטית בשם LoaderFactory המכילה את המתודה הסטטית האחראית על יצירת האובייקטים השונים בהינתן קונטקסט ורשימת פרמטרים.

כל רכיב במערכת שיצטרך רכיב טעינה, יפנה אל ה- LoaderFactoryויעביר אליו את קונטקסט האובייקט הרצוי יחד עם רשימת פרמטרים מתאימה.

* Sequence Diagram - החלטנו להתמקד בתקשורת בין טופס (Form) לבין המפעל (LoaderFactory<T>), ללא התייחסות למימוש האדפטר בדיאגרמה זו (כפי שנראה בתרשים המחלקות מטה). ההסבר למימוש האדפטר ינתן בתבנית מספר 3.



* Class Diagram - החלק הרלוונטי לתבנית העיצוב מסומן בריבוע כתום, אך חשוב לציין גם את האדפטר שמשתתף בתהליך.



פירוט מחלקות

* המחלקה Form מהווה שם כללי לכלל הטפסים השונים (Forms) שישתמשו באדפטר ( - ILoaderAdapter<T>יוסבר בתבנית מספר 3), אשר מגשר בין הטופס לבין המפעל ע"י שליחת הפרמטרים ליצירת האובייקט הרצוי.
* המחלקה LoaderFactory<T> הינה מחלקה סטטית המהווה את המפעל היוצר את האובייקטים השונים.
* המחלקה Loader<T> מהווה את מחלקת הבסיס עבור האובייקטים השונים שניתן לייצר במפעל.
* המחלקה ImageLoader<T> מהווה מחלקה האחראית על טעינת תמונות ביישום.
* המחלקה PieChartLoader<T> מהווה מחלקה האחראית על טעינת נתונים לתוך תרשימי עוגה ביישום.
* האינום eLoaderFactoryContext מהווה את הקונטקסט לפיו המחלקה LoaderFactory<T> יוצרת אובייקטים.

### תבנית מס' 2 – [Façade]

* סיבת הבחירה / שימוש בתבנית:

במטלה הקודמת, מימשנו את מחלקת FrienDiscoverForm כמחלקה המכילה גם לוגיקה וגם ממשק משתמש.

במטלה הנוכחית, החלטנו להפריד בין החלק הלוגי לבין ממשק המשתמש על מנת להפוך את החלק הלוגי ל-reusable.

לאור הפרדת החלק הלוגי, החלטנו להשתמש בתבנית העיצוב Façade על מנת להגביר את בטיחות ונוחות השימוש ולייעל את נוחות התחזוקה.

* אופן המימוש:

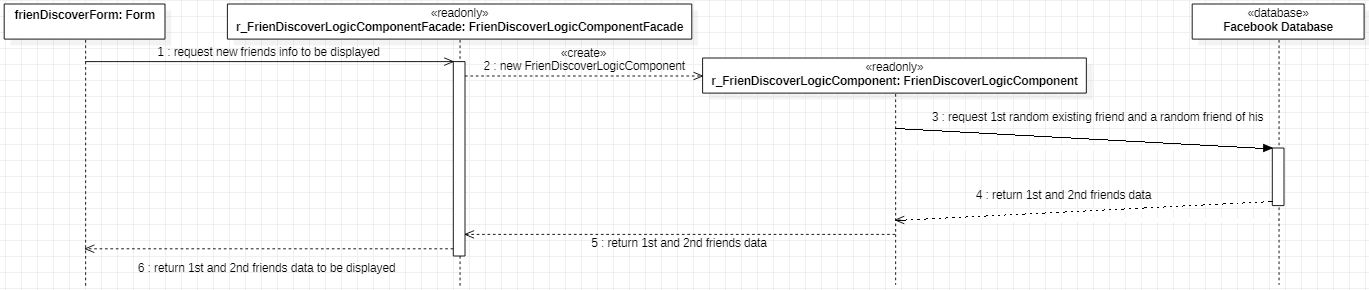
כפי שנאמר בסעיף הקודם, החלטנו להפריד את הרכיב הלוגי לתוך מחלקה חדשה (FrienDiscoverLogicComponent).

את המחלקה החדשה הגדרנו כמחלקה פנימית (internal) בתוך הפרויקט Ex02.FacebookAppLogic.

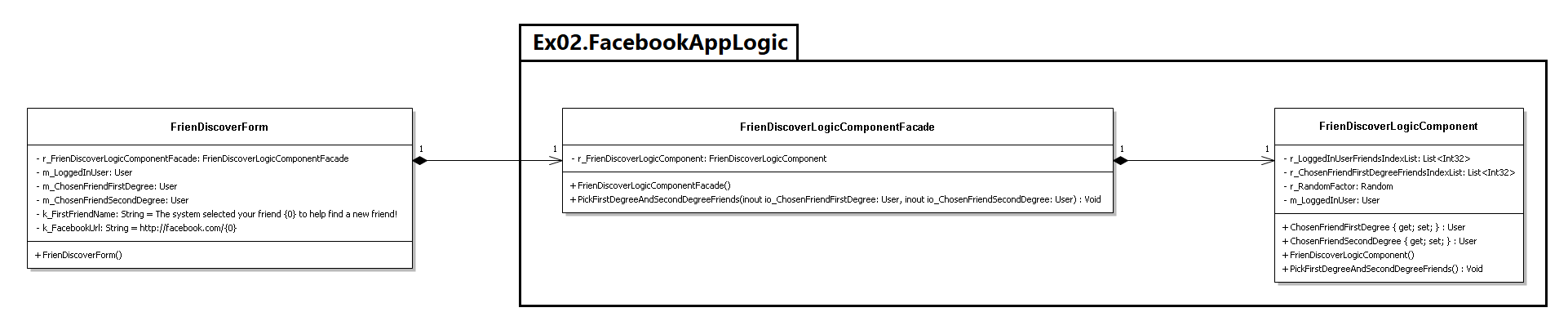
בנוסף הגדרנו מחלקת גישה ציבורית (FrienDiscoverLogicComponentFacade) שרק דרכה אפשר לפנות לרכיב הלוגי.

עצם העובדה שמחלקת הרכיב הלוגי הוגדרה כפנימית ומחלקת הגישה הציבורית הינה החוצץ לשאר המערכת, מהווה מימוש ל-Façade אטום.

* Sequence Diagram



* Class Diagram



פירוט מחלקות

* מחלקת FrienDiscoverForm מהווה את הקליינט אשר משתמש בתבנית העיצוב .Façade
* מחלקת FrienDiscoverLogicComponentFacade מהווה את החוצץ בין הקליינט לבין החלק הלוגי.
* מחלקת FrienDiscoverLogicComponent מהווה את החלק הלוגי אותו אנו רוצים להכמיס מפני שאר המערכת פרט ל-Façade.

### תבנית מס' 3 – [Adapter]

* סיבת הבחירה / שימוש בתבנית:

במערכת שלנו, ניתקלנו במצב בו קיימים שני רכיבים בעלי ממשק שונה אשר צריכים לעבוד ביחד:

1. הרכיב הראשון הוא כל טופס אשר רוצה להשתמש במחלקת טעינה כלשהי (ImageLoader<T>, PieChartLoader<T>) ומחזיק פרמטרים מסוגים שונים.
2. הרכיב השני הוא המפעל ליצירת מחלקות טעינה (LoaderFactory<T>) אשר משתמש ברשימות אובייקטים (List<object>) בתור ארגומנטים לפונקציות הבנאי של מחלקות הטעינה השונות, וכך נוצר מצב פולימורפי בו נוצרים אובייקטים שונים באופן פעולה זהה.

לשני הרכיבים קיים ממשק שונה, ולכן כדי לאפשר לטופס להשתמש בשירותי המפעל, החלטנו להשתמש בתבנית העיצוב אדפטר.

* אופן המימוש:

הוחלט להגדיר ממשק (ILoaderAdapter<T>) אשר יכיל שתי הכרזות של מתודות, כאשר כל מתודה מטפלת בסוג אחד של מחלקת טעינה - חשוב לציין כי בחרנו בשיטת המימוש object adapter, כפי שיתואר בהמשך.

כל הכרזה כזו תתווך בין הטופס לבין המפעל, בשימוש קונטקסט למחלקת הטעינה הרצויה, יחד עם רשימת פרמטרים שונים שהטופס מעביר לשם יצירת האובייקט.

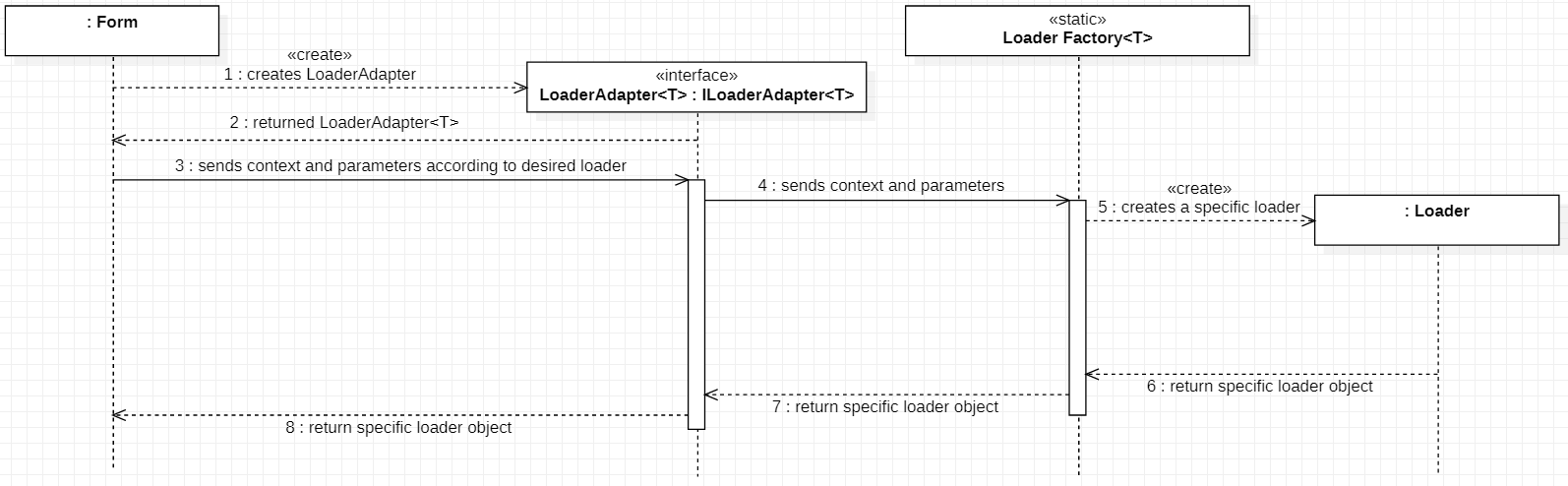
המחלקה אשר תממש את הממשק המדובר היא המחלקה LoaderAdapter<T>.

כל מחלקה אשר תרצה להשתמש בשירותיו של המפעל, תחזיק כשדה משתנה מסוג ILoaderAdapter<T>, ודרכו תפנה למפעל ותקבל את האובייקט הרצוי בהתאם לקונטקסט והפרמטרים שנשלחו.

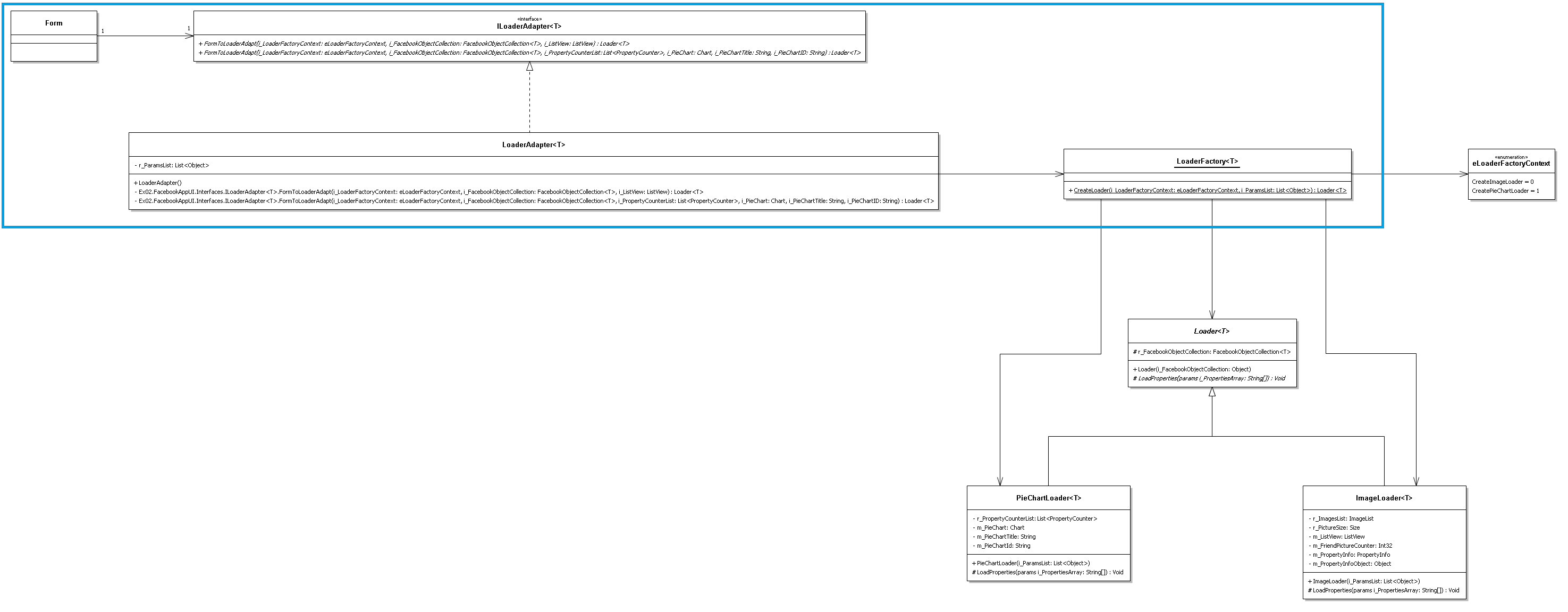
בנוסף, נציין כי הכרזות המתודות בממשק הן בעלות שם זהה.

הוחלט לפעול כך, כדי לגרום למצב של העמסת מתודות (method overloading) ובכך ליצור נוחות שימוש עבור המשתמש, אשר לא צריך "לזכור" שם מתודה ספציפית, אלא רק להעביר קונטקסט ופרמטרים מתאימים.

* Sequence Diagram



* Class Diagram - החלק הרלוונטי לתבנית העיצוב מסומן בריבוע כחול, אך חשוב לציין גם את המפעל שמשתתף בתהליך.



פירוט מחלקות

* המחלקה Form מהווה שם כללי לכלל הטפסים השונים (Forms) שישתמשו באדפטר.
* הממשק ILoaderAdapter<T> מהווה את ממשק האדפטציה המקשר בין טופס למפעל.
* המחלקה LoaderAdapter<T>מממשת את הממשק ILoaderAdapter<T>.
* המחלקה הסטטית LoaderFactory<T> מהווה את המפעל ליצירת מחלקות הטעינה.

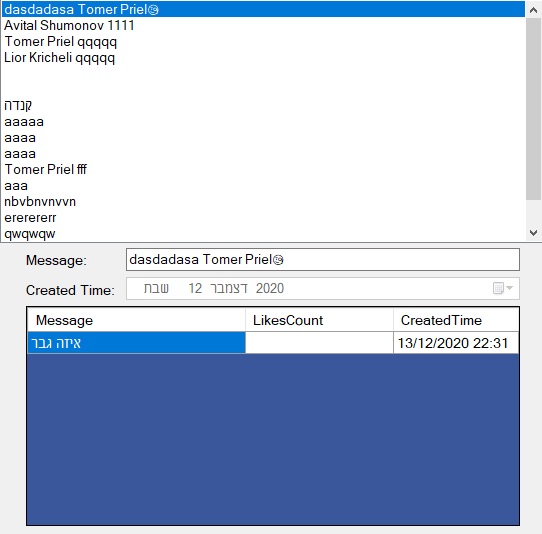
עבודה עם Data binding

ביצענו שימוש ב- Data binding במספר טפסים במערכת והם:

* EventsForm
* GroupsForm
* PostsForm
* LikedPagesForm

עבור כל אחד מהטפסים בוצע מימוש דומה, ובו נשלפים ערכים לתצוגה נוחה ודינמית בתוך ה-UI שנוצר באופן אוטומטי ע"י ה-Data binding.

דוגמה לשימוש ב- Data binding בטופסPostsForm :



עבודה אסינכרונית

יישמנו עקרונות עבודה אסינכרונית במספר טפסים (Forms) במערכת:

* AlbumsForm
* FrienDiscoverForm
* FriendsForm
* SelectedAlbumForm
* StatisticsForm

הסיבה לכך שיושמה עבודה אסינכרונית בטפסים הנ"ל, היא העובדה שבכל אחד מטפסים אלה מתבצעת טעינת נתונים אשר אורכת זמן (תמונות, נתונים לתרשימי עוגה ומידע הנשלף משרת מרוחק).  
באמצעות עבודה אסינכרונית, אנו חוסכים למשתמש את זמן ההמתנה עד טעינת הנתונים, ובו בזמן מאפשרים שימוש במערכת באופן רספונסיבי.

בכל טופס שצוין לעיל, נפתח תהליכון (thread) נוסף שבו מתבצעת פעולת הטעינה עבור הטופס הספציפי בלבד.

במערכת, אנו משתמשים בשתי מחלקות העוסקות בטעינת נתונים שונים:

* ImageLoader<T> - מחלקה האחראית על טעינת תמונות.
* PieChartLoader<T> - מחלקה האחראית על טעינת תרשימי עוגה.

בשתי מחלקות אלו, מתבצעות קריאות invoke בכדי לגשת ל-thread המקורי בו נוצר הפקד המוצג בממשק המשתמש. בדרך זו אנו מאפשרים את שינוי תוכן הפקד תוך שמירה על thread safety.