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

* פיצ'ר ראשון: הפיצ'ר מציג למשתמש המחובר בלחיצת כפתור את תחזית המזל היומית שלו, יחד עם המצב רוח שלו, מספר המזל שלו.

הפיצ'ר מקבל את המידע מ API חיצוני באמצעות HttpRequest מהלינק:

<https://aztro.sameerkumar.website>

בתרגיל הראשון מימשנו אותו במחלקה של Horoscope וכעת העברנו אותו

לCache Proxy אשר מחזיק בתוכו RemoteAPI שבו מבצעים את הRequest.

* פיצ'ר שני: הפיצ'ר מציג למשתמש המחובר בלחיצת כפתור כמה ימים נותרו עד יום ההולדת הבא שלו ואיזה גיל הוא יהיה.

הפיצ'ר ממומש בAppEngine.

### תיאור Data Binding:

מימשנו Data Binding שמקבלת מידע מאובייקט FacebookWrapper.Object.Album לאחר שהמשתמש התחבר בהצלחה (ניתן למצוא זאת במתודה buttonLogin\_Click()).

השדות שה Data Binding מציג הם Name, Description של האלבום הרלוונטי. ניתן להשתמש

בData Binding הנ"ל על מנת לערוך את ה Description של כל אלבום באמצעות לחיצה על תמונה של אלבום מסוים.

### תיאור עבודה אסינכרונית:

מימשנו RemoteAPI במתודה בשם SetHoroscope() שהיא מתודה מסוג async ובתוכה מתבצע קריאה ל API חיצוני באמצעות HttpRequest עם שימוש ב await.

ביצענו קריאה אסינכרונית כיוון שפעולה כזו לוקחת זמן ולא היינו רוצים שהלקוח ימתין לכך שתתקבל התשובה של הקריאה של הHttpRequest, כך באופן זה אנו חוסכים מהמשתמש זמן ודואגים שאם הוא מבצע פעולה אחרת במקביל שהיא תוכל לקרות.

### תבנית מס' 1 – [Singletone]

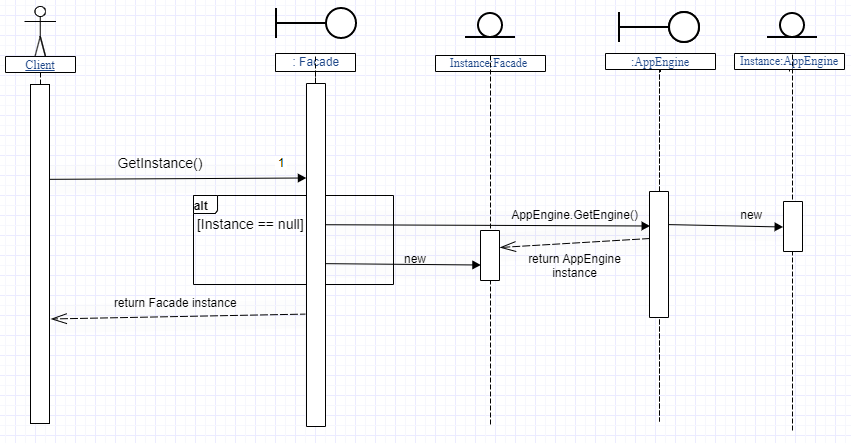
* שימוש בתבנית :

השתמשנו בתבנית עבור מחלקות AppEngine,Façade משום שאנו מעוניינים שיהיה מופע אחד ויחיד עבור המערכת כדי שכל שאר המחלקות יוכלו להשתמש בו ללא צורך ביצירה מרובה שלהם שתיצור בעיות. בנוסף המחלקה AppEngine היא סינגלטון שניתן לגשת אליה רק ממחלקות הDLL של Engine כיוון שהיא internal, ומחלקת ה Façade כולם יכולים לגשת אליה כיוון שהיא public.

* אופן המימוש:

יש לנו מחלקות בשם AppEngine,Façade.

שני המופעים נוצרים באופןlazy רק לאחר קריאה ראשונית על ידי בנאי פרטי ויצירה בשדה סטטי.

* Sequence Diagram
* Class Diagram

Diagram

Description automatically generated

### תבנית מס' 2 – [Facade]

* סיבת הבחירה / שימוש בתבנית:

השימוש בתבנית הוא על מנת להקל על צד הלקוח שמשתמש במערכת שלנו

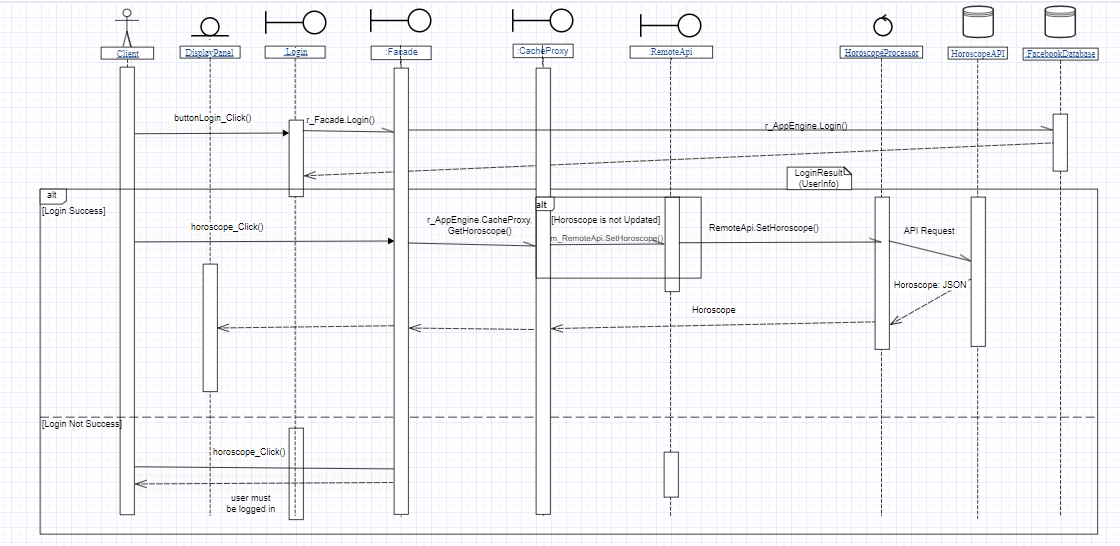
AppEngine אשר אטומה ללקוח ובמקומה הוא מקבל את ה Façade אשר נותנת לו צורת שימוש נוחה יותר. במיוחד אם בעתיד נרצה להמשיך לשפר ולשדרג את המערכת, בנוסף הדבר מגדיל את בטיחות השימוש במערכת בכך שללקוח אין גישה ישירה למערכת המורכבת יותר אלא רק למה שהוא צריך.

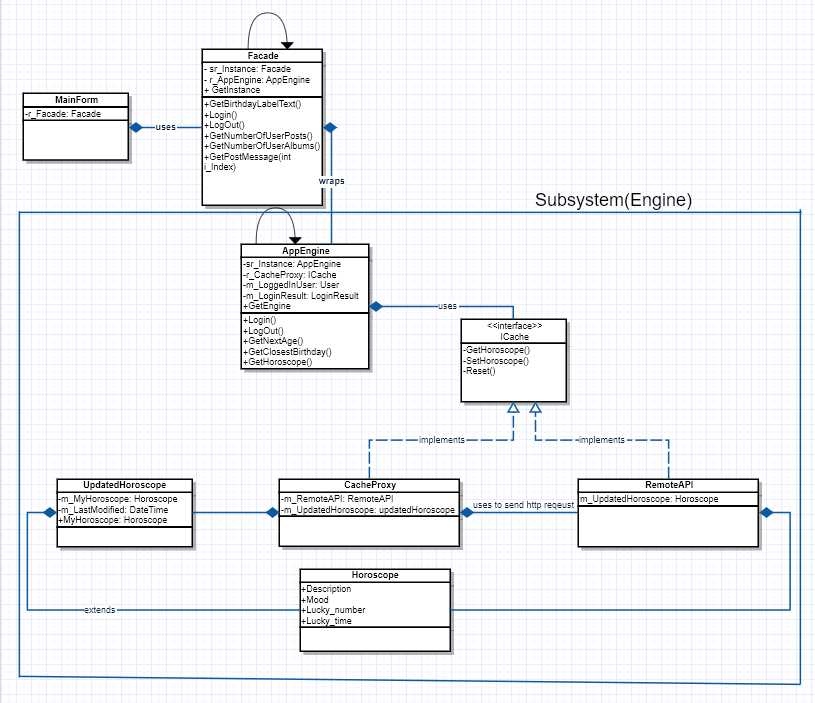
* אופן המימוש:

יצרנו מחלקה בשם Façade אשר מחזיקה את המופע הסינגלטוני של AppEngine,

ומממשת את כל המתודות שדורשות כמה פעולות שונות מ AppEngine וכל המתודות אשר הלקוח ירצה להשתמש בהם.

* Sequence Diagram



* Class Diagram

### תבנית מס' 3 – [Cache Proxy]

* סיבת הבחירה / שימוש בתבנית:

בפיצ'ר הראשון שלנו אנו משתמשים בHttpRequest בעזרת API חיצוני.

על מנת לחסוך בכמות הHttpRequest אשר לוקחות זמן ומכבידות על המערכת שלנו ובנוסף בתעבורה והשרת. בעזרת תבנית ה Cache Proxy אנו שומרים את המידע שקיבלנו בקריאה הראשונה מידע אשר לא משתנה באופן רציף אלא רק באופן יומי. ולכן בפעם הבאה שהמשתמש יבקש לצפות במידע במידה והבקשה נוצרה עדיין באותו היום ולכן מידע זה לא ישתנה הוא יקבל את המידע ששמרנו בעזרת התבנית ללא HTTP REQUEST נוספת , אשר ישפר את הביצועים ומהירות הפיצ'ר.

* אופן המימוש:

יצרנו Interface בשם ICache

אותו מממשות המחלקות: RemoteAPI , CacheProxy.

בנוסף יצרנו מחלקה של updatedHoroscope המוודאת שההורוסוקופ השמור הוא אכן המעודכן, אם כן מחזירה אותו ואם לא אז מחזירה שאין כזה.

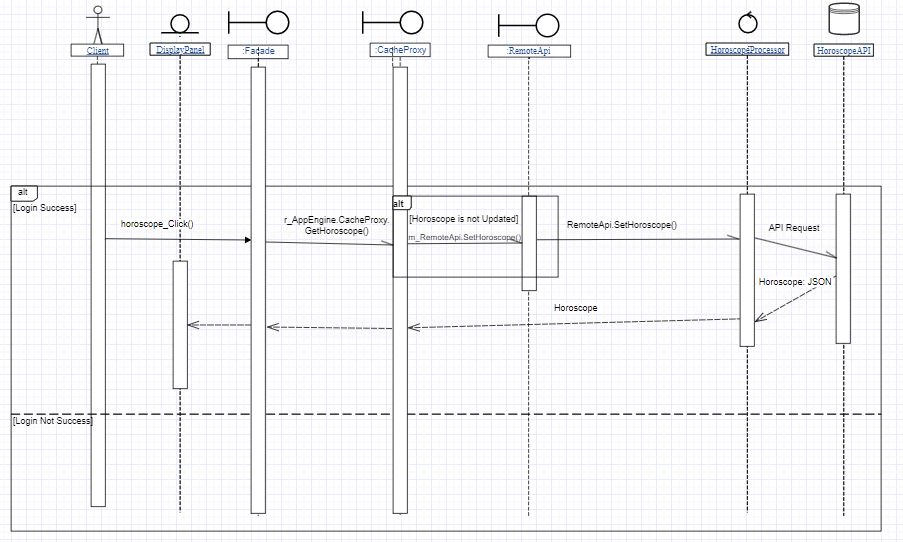
החזקנו מופע של updatedHoroscope ,RemoteAPI, ב CacheProxy-.

הבקשה של ה HttpRequest נשלחת מתוך RemoteAPI.

ונשמרת ב updatedHoroscopeשמוחזק ב CacheProxy.

כאשר הלקוח משתמש רק ב CacheProxy דרך ה Façade וככה מובטח צמצום מירבי של בקשות HTTP.

* Sequence Diagram



* Class Diagram Diagram

  Description automatically generated