

פרויקט גמר – מדעי המחשב

ניתוח שיטות להנדסה לאחור (Revers Engineering) של אפליקציות
באנדרואיד תוך כדי התעמקות באפליקציה מוכרת ותעבורת <https>.

דולב פרנקו



Grader

פרטי התלמיד:

שם:

תעודת זהות:

נייד:

תאריך לידה:

כתובת מייל: dolevfranco@gmail.com

מקום מגורים:

פרטי בית הספר:

מוסד לימודי:

סמל המוסד הלימודי:

כתובת:

טלפון בית הספר:

פרטי המנחה:

שם המנחה: אור

תעודת זהות:

תחום עבודה (מקצוע):

פרטי תואר המנחה:

טלפון:

דוא"ל:

שם המורה המלווה:

תוכן עניינים

4.....	מבוא.....
5	חלק תיאורטי
6.....	מבוא להנדסה לאחר
9.....	מבוא לרשתות
12.....	מבוא לשפת Dart
13.....	מבוא לFlutter
17	חלק מעשי
18.....	הצגת תכנון ומבנה הפרויקט
19.....	הצגת כלי עבודה למימוש הפרויקט
20.....	תיאור שלבי העבודה על הפרויקט
21.....	המחשה ויזואלית של המוצר העובד
24.....	בעיות בתהליך מימוש האפליקציה
26.....	שלב ראשון – הנדסה לאחר
30.....	שלב שני – הבנת התעבורה של משוב
34.....	שלב שלישי – כתיבת API למשוב
42.....	שלב רביעי – הכנת האפליקציה
46.....	ספריות בהם השתמשתי ביצירת האפליקציה
47.....	הצעת נושא לעבודת גמר במדעי המחשב
49.....	ביבליוגרפיה
50	הקוד

מבוא

מטרת העבודה : יצירת אפליקציה לאנדרואיד, iOS ולweb המחקה את האפליקציה המוכרת של משרד החינוך – "משוב" לצורך הגברת נגישות התלמידים לנתונייהן הלימודיים.

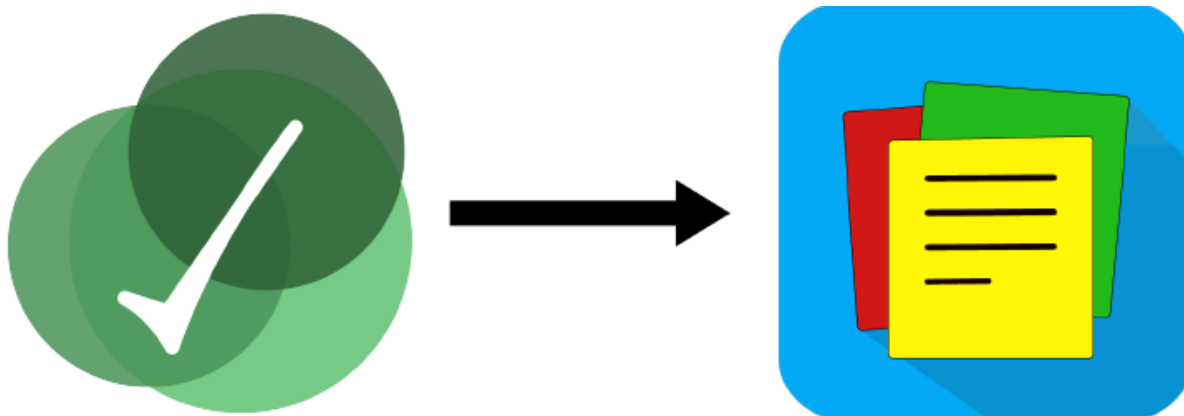
עבודת הגמר תעסוק בניתוח הנדסה לאחר של אפליקציות באנדרואיד תוך כדי התעמקות באפליקציית "משוב" ותעבורת <https> בין לקוח לשרת.

הרעיון הראשוני של האפליקציה עלה כאשר אני וחברים שלי בכיתה ט, ניסנו לבדוק למי יש את הממוצע הכי גבוה. התחלקנו לזוגות וכל פעם מישהו אחד היה עם מחשבון והשני היה עם משוב וכך רק הצלחנו לחשב את הממוצע של אחד. תהליך זה לקח לפחות רבע שעה ויצר אופציה לביצוע טעויות בממוצע עד כדי כך שהיה צריך לחשב הכל מחדש. יתר על כן, תלמידים רבים נתקלו כל הזמן בבעיות במשוב, כגון : איטיות ובאגים קטנים אך מורגשים, אי התאמה של משוב לטלפון (טקסט חתוך), עיצוב לא עדכני ופונקציות רבות חסרות שימוש. בעזרת יצירת האפליקציה "גריידר" אנסה לענות ולספק פתרון לכל הבעיות שהעליתי לעיל.

העבודה על יצירת האפליקציה חשפה בפניי לתחומים מגוונים במדעי המחשב שלא הכרתי וגרמה לי להתפתח הן בחשיבה האלגוריתמית שלי והן במחשיבה היצירתית שלי. העבודה עזרה לי להבין כיצד באמת עובדים נושאים חשובים ובסיסיים במדעי המחשב ובעולם כולו.

בעבודה זו יש סקירה עמוקה של נושאים שונים במחשבים, בניהם : רשתות, הנדסה לאחר, כתיבת קוד לניידים, תקשורת בין לקוח לשרת ועוד, כאשר התוצר הסופי יהיה אפליקציה משודרגת למשוב – ואף טובה יותר. העבודה מומשה בשפת Dart, Flutter מכיוון ששפה זו היא שפה מוכרת לבניית אפליקציות וניתן ליישם בה את הנושאים הנ"ל.

על מנת להבין את הפרויקט תחילה יש להבין היטב את כל הנושא של הנדסה לאחר. אתחיל במבוא קצר על הנדסה לאחר, לאחר מכן מבוא לרשתות, מבוא לשפת dart ולflutter, ולבסוף אסביר על המחלוקות השונות שנתקלתי בהם ואראה את מימוש התוצר הסופי.



חלק תיאורטי



מבוא להנדסה לאחור

הנדסה לאחור הינו תהליך גילוי עקרונות טכנולוגיים של מכשיר, אובייקט או מערכת באמצעות ניתוח המבנה, תפקידו ופועלו. הדבר כרוך בלקיחת מכשיר, מערכת או תוכנה כלשהם ושבירתם, ניתוחם והסקת אופן פעולתם. שימוש בהנדסה לאחור יכול להיות שימושי עבור מגוון רחב של מטרות:

- מציאת קוד זדוני.
- תיעוד מערכת המתועדת בצורה לא טובה בזמן שהמתכננים כבר אינם זמינים.
- לאחזר קוד מקור שאבד ולתקן בעיות.
- ליצור פרויקט תחרותי הדומה לפרויקט המקורי.
- הבנת פרוטוקול תקשורת בין מערכות.
- להתגבר על הגנה.

הנקודות שכמובן רלוונטיות ביותר עבור פרויקט זה הם הבנת פרוטוקול תקשורת בין מערכות וגם יצירת פרויקט תחרותי.

שימוש ותרגול בהנדסה לאחור

ישנם מספר תחומים בהם משתמשים בהנדסה לאחור לעתים קרובות. אחת החשובות ביותר היא יכולת ההדדיות של מערכות. מכיוון שמערכות מורכבות מיוצרות בדרך כלל על ידי כמה תתי מערכות, יש צורך לחברן. לא תמיד מעצבי מערכות חושבים על חיבור מערכת למערכת גדולה יותר, ולא תמיד למערכת יש ממשקים מתועדים היטב. במקרה זה מהנדסי הרוורס צריכים לנתח את המערכת ולהגיע עם ממשקים שיאפשרו שילוב במערכת גדולה יותר. אחד היישומים החשובים ביותר של הנדסה הפוכה הוא בניתוח מערכות אבטחה וגם בניתוח של תוכנות זדוניות. כמעט לא היינו יכולים לדבר על תחום זה ללא הנדסה לאחור. גם התוקפים וגם המגינים משתמשים בהנדסה לאחור. התוקפים משתמשים בהנדסה לאחור כדי להבין את נקודות התורפה של מערכות שהם יכולים לתקוף. כמו כן הם עשויים לנתח תכונות ואפשרויות של מערכת ההפעלה ללא תיעוד לצורך בניית יישומים זדוניים. בצד השני, המגינים גם מנתחים ומצמצמים את נקודות התורפה של מערכות. שימושים אחרים הם גם אם מישהו מעוניין לפתח מוצר תחרותי דומה למוצר קיים. בנוסף לכך, חברות גדולות משתמשות בהנדסה לאחור כדי לבדוק אם מוצרים מסוימים מפריס את הפטנטים או זכויות היוצרים שלהם. בודקי תוכנה גם משתמשים לפעמים בהנדסה לאחור כדי לאתר פגמים ושגיאות בתוכנה. במקרים מסוימים ההנדסה הפוכה אסורה על פי חוק, אך עדיין משתמשים בה, לדוגמא: באיחוד האירופי מותר להשתמש בהנדסה הפוכה לצורך יכולת פעולה הדדית, אך אסור להשתמש בו ליצירת מוצר במקביל.

טכניקות של הנדסה לאחור

כאשר אנו מדברים על הנדסת תוכנה ישנן מספר גישות להנדסה לאחור. למעשה אנו יכולים להפריד אותן לניתוח דינמי וסטטי של תוכנות מסוימות. **ניתוח דינמי** כולל הפעלה של תוכנות והתבוננות מה התוכנה הזו עושה. ניתוח

דינמי כולל מספר דברים כמו התבוננות ב- GUI¹ של תוכנות, פלט ותכונות התוכנה, שינויי דיסק, קריאה וכתובה של דיסקים, שינויים בזיכרון, קריאה וכתובה בזיכרון, תעבורת רשת, שימוש במעבד, פלט באגים אם קיים, יישומים ויומני מערכת.

ניתוח סטטי כולל פירוק האפליקציה וניתוח קוד המקור של היישום. לעתים קרובות אי אפשר לאחזר את הקוד המקורי, מכיוון שהיישום נאסף בשפות כגון C, C++, Delphi או כל מיני שפות תכנות נמוכות² אחרות.

ניתוח קוד המכלול הרבה יותר קשה, אבל לא בלתי אפשרי. עבור כמה שפות תכנות גבוהות יותר כגון שפות NET או JAVA אפשר לאחזר קוד מקור מקורי או כמעט מקורי באמצעות מפענחים מ-bytecode.

הנדסה לאחור בעולם הניידים

נבחן הנדסה לאחור על ידי פירוק של פלטפורמת מובייל מרכזית - אנדרואיד.

אנדרואיד משתמשת בקבצי apk ליישומים. יישומי אנדרואיד לרוב נכתבים ב-Java ומקומפלים ל-bytecode בשביל ה-Java Virtual Machine. בהמשך הם מתורגמים ל-Dalvik bytecode המאוחסן בתוך קובץ עם סיומת dex שיכול לרוץ על תוכנת אנדרואיד. יש לציין שבגרסאות אנדרואיד חדשות נעשה שימוש ב-ART שהחליף את Dalvik VM אבל לא נתמקד בכך בעבודה זו מכיוון שזה לא העיקר. כדי לנתח לאחור קובץ apk, כל מה שצריך זה [Java Decompiler](#) וכלי שנקרא [dex2jar](#). באמצעות dex2jar אפשר לשנות קובץ dex (נרחיב עליו בהמשך) ל-jar.

יש מספר שלבים בניתוח קובץ apk :

1. שינוי השם של קובץ apk לzip וחילוץ. קובץ apk הינו קובץ zip (ניתן לראות שארבעת הבתים הראשונים של ה-header של פתיחת כל קובץ יהיה כמו של zip – 0x04034d50). בשל כך, כדי לראות את תוכנו, מספיק לשנות את הסיומת שלו ל-zip ולחלץ את הקובץ כמו zip רגיל.
2. לקיחת הקובץ classes.dex והפיכתו לקובץ jar בעזרת dex2jar.
3. פתיחת קובץ jar באמצעות תוכנה מתאימה.

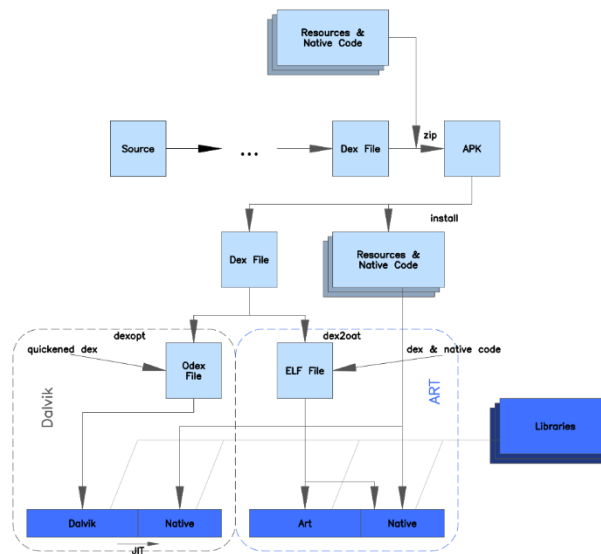
קבצי Dex

dex הינו קיצור של Dalvik Executable format. קובץ dex מכיל קוד שבסופו של דבר מבוצעת על ידי Android Runtime (זמן הריצה המנוהל המשמש את היישומים וכמה שירותים במערכת אנדרואיד). לכל קובץ apk יש קובץ classes.dex אחד לפחות המתייחס לכל המחלקות או הפעולות המשמשים בתוך אפליקציה. בעיקרו של דבר, כל

¹ Graphical User Interface – עיצוב של התוכנה

² שפות שתלויות בחומרה של המחשב ויש יותר אופציות רחבות עם המכונה עצמה.

פעילות או אובייקט המשמשים בתוך קוד הבסיס, יהפכו לביטים (bytes) בתוך קובץ dex שניתן להריצו כאפליקציית אנדרואיד.



תמונה 1 – מסגרת יישום אנדרואיד

כל קבצי המקור של Java בפרויקט אנדרואיד נאספים לראשונה, על ידי הקומפיילר, לקבצי .class, המורכבים מהוראות בית-קוד. ביישום Java מסורתי, הוראות אלה יבוצעו על JVM (Java virtual machine). עם זאת, אפליקציות אנדרואיד מבוצעות ב- Android Runtime, המשתמשת בקודים לא תואמים, ולכן נדרש שלב DEXING נוסף, שבו קבצי .class הופכים לקובץ dex יחיד.

מקובץ dex לצורה קריאה

כפי שהוסבר לעיל, ה-APK מכיל קבצי dex אשר מכילים Dalvik bytecode. המערכת מבינה את פורמט זה ויודעת להתמודד עמו. אולם, לא פשוט לקרוא את ה-bytecode ולכן, יש כלי הנקרא dex2jar שמאפשר להעביר את ה-bytecode לקובץ יותר קריא שאדם יכול להבין. צורה זו נקראת "jar" וניתן לראותה בעזרת התוכנה Jd-gui. לדוגמה, אם נרשום ב-java את הקוד הבא: `int x = 42`, ה-bytecode שלו יראה כך: `2A 00 00 13`, זהו קוד לא קריא. אך אם נפעיל על הבתים הללו את dex2jar ואז נפתח אותו באמצעות התוכנה Jd-gui, נקבל את הקוד המקורי: `int x = 42`. כלומר, עם הכלים הנכונים קל מאוד להפוך קובץ dex לקוד java קריא ומובן. יש לציין שניתן לעבור ישירות מקובצי apk לקבצים מסוג smali אך קוד מסוג זה אינו מספיק מפורט ומובן בשביל להבין את הפעולה של אפליקציית משוב.

בשביל לעבור מ-dex לקובץ jar מספיק להריץ את הפקודה הפשוטה הבאה: `d2j-dex2jar.bat classes.dex`

כדי לפתוח את הקובץ החדש שנוצר לנו, יש להפעיל את התוכנה Jd-gui ולפתוח את הקובץ jar החדש שנוצר לנו. כך, נוכל לראות את כל הקבצי java המצויים בקובץ ה-dex ולקרוא אותן בצורה מסודרת ונוחה.

מבוא לרשתות

מתחילת המהפכה התעשייתית, במאה ה-19, התקשורת החשמלית החלה להתפתח. בעבר (לפני כמאה שנה) כשאנשים רצו לתקשר אחד עם השני הם היו צריכים להעביר מכתבים, שלקח להם הרבה זמן להגיע ולא היה ניתן להבטיח את הגעתם ליעד. כיום, אנחנו רק צריכים לפתוח את מכשירנו האישי ונוכל לשלוח ולהעביר מידע לכל האנשים בעולם בשניות. איך תהליך זה קורה?

בזכות הרשתות שאנחנו מכירים כיום, רשת האינטרנט בעיקר, אנחנו יכולים להעביר מידע ממחשב אחד למחשב אחר. הרשתות פרוסות בכל רחבי העולם והם מבוססת על חבילת פרוטוקולי התקשורת.

מה הוא האינטרנט?

לרוב האנשים האינטרנט היא ענן אשר מאפשר להעביר קבצים ממקום אחד לשני. במציאות, האינטרנט הוא עשרות אלפי כבלים אופטיים מתחת לאדמה וליים אשר מחוברים זה לזה במיליוני קילומטרים ובניהם עובר המידע הדיגיטלי.

פרויקט זה משלב בתוכו נושאים המתקשרים לרשתות ולכן כדי להבינו יש להבין כיצד פועלות הרשתות.

מושגים חשובים להבנת הרשתות:

- **כתובת IP** – כותבת המורכבת מארבעה בתים, המייצגת נקודת קצה ברשת. לכל נקודת קצה ברשת יש כתובת IP וכך אפשר לשלוח אליה או לקבל ממנה מידע.
- **מודל חמשת השכבות** – המודל מספק הסבר והדרכה כללית על מרכיביה ותפקידיה השונים של הרשת. המודל נוצר על ידי ארגון התקינה הבינלאומי (ISO) המראה ומסביר כיצד צריכה להיראות תקשורת בין מערכת מחשב אחת לשנייה, ללא תלות בייצרן של אותה מערכת. כל שכבה במודל מספקת שירות רק לרמה שמעליה, מבלי לחשוף אותה לאופן בו השירות שלה ממומש. להלן שכבת המודל ותפקידיהם העיקריים:
 1. **השכבה הפיזית** – השכבה אחראית על העברת ביטים (Bits) ממקום אחד למקום אחר על ידי כבלי רשת, סיבים אופטיים, גלים אלקטרומגנטיים ועוד.
 2. **שכבת הקו** – מטרת השכבה היא להעביר מידע בצורה פיזית בין שני ישויות סמוכות.
 3. **שכבת הרשת** – תפקידה של שכבה זו היא למצוא את המסלול הטוב ביותר מנקודת קצה אחת אל השנייה בעזרת נתיבים אשר מנתבים את הפקטות בין הרשתות השונות. (פרוטוקול IP)
 4. **שכבת התעבורה** – שכבה האחראית על שליחת הפקטה אל יעדה ובדיקת אמינות נקודות הקצה. בנוסף השכבה משתמשת בפורטים העוזרים לנתב כל שירות ליעודו. (פרוטוקול TCP)
 5. **שכבת האפליקציה** – משמשת לשימושים שונים לצורכי האפליקציה ומעבירה את המידע בהתאם לפרוטוקולים השונים.

חשוב מאוד להבין את תהליך ואופן פעולת שליחת פקטה באינטרנט גם בשביל הבנת אופן הפעולה של אפליקציית משוב וגם זהו ידע בסיסי לכל אדם העוסק בתכנות בכל צורה שהיא. על מנת להבין טוב יותר המושגים הנ"ל וכיצד עובד האינטרנט, אראה דוגמא ואסביר מה התהליך שקורה מרגע בו

רושמים בדפדפן את הכתובת www.facebook.com, ועד שמופיע לנו על הדפדפן העמוד של פייסבוק.

רשימת השלבים:

1. המשתמש כותב את הכתובת בדפדפן.
2. האינטרנט בנוי על כתובות IP בלבד, אין לדפדפן דרך לדעת איפה נמצאים השרתים של פייסבוק לפי כתובת ה-URL. לכן הדפדפן מתחיל בתהליך הנקרא DNS Query:
 - הדפדפן מוציא את כתובת הדומיין מתוך ה-URL (למשך מתוך <https://www.facebook.com/hakfar.hayarok> את הדומיין facebook.com) ומחפש במקום ב-DNS Cache האם הכתובת קיימת. ה-DNS Cache – הוא מקום בו מאוחסנים צמדי דומיין ו-IP ששמורים במחשב.
 - אם דומיין קיים והדפדפן מוצא את ה-IP, הדפדפן עובר לשלב הבא.
 - אם הדומיין לא קיים הוא יפנה ל-DNS Server, שיעביר בצורה רקורסיבית לעוד שרתים כדי שימצא או לא את כתובת ה-IP.
 - במידה והדומיין לא נמצא, תחזור הודעת שגיאה.
3. HTTP – בהנחה שהוחזר משרתי ה-DNS כתובת IP נכונה שמייצגת מחשב ברחבי הרשת, בשלב הבא נבנית פקטת HTTP בתוך הדפדפן. במקרה שלנו הבקשה בקבצי ה-Header תכיל את ה, name host ותבקש מפייסבוק להחזיר לנו את דף הבית.
4. בניית נתיב לפקטה עד הגעתה ליעד.
- שימוש ב: Table NAT כל מחשב ברשת הפנימית מקבל כתובת פרטית משלו על ידי הראוטר. לראוטר יש כתובת חיצונית אחת שלרוב היא קבועה. כאשר הראוטר מקבל מידע מאחד המחשבים שמחוברים אליו, נעשה שימוש ב-Table Nat. מטרתה היא להעביר פקטות ממחשבים בתוך הרשת למחוץ לה. הראוטר מקבל מידע מסוים מהמחשבים אליו הוא מחובר, כתובת ה-IP של המחשב מוחלפת בכתובת הראוטר וכן הלאה. הכתובות וה-ports המוחלפים יאוחסנו בטבלה הנקראת Table NAT כדי שנדע לאן צריך להחזיר את המידע שמתקבל.
5. כך, באמצעות Tables Routing ומושגים שהוגדרו קודם לכן הפקטה מגיעה עד לשרת המבוקש עם הכתובת ששייכת ל-facebook.com. שם היא נבנית, ומוחזרת תשובה.
6. התשובה מגיעה אלינו והדפדפן מנתח את התשובה ומציג אותה.

Proxy server וכיצד הוא עובד

עלינו להבין גם כיצד שרת proxy עובד משום שבמהלך העבודה נשתמש באפליקציית Burp Suite ובתוכה נשתמש בפונקציית proxy browser שבעצם יוצר שרת proxy בין הדפדפן לבין האפליקציות שנשתמש בהם, במקרה שלנו – ["משוב"](#).

התוכנה Burp Suite נותנת את האופציה ליירט, לבדוק ולשנות את התעבורה הגולמית החולפת בין שני הכיוונים.

הגדרה

שרת proxy הוא כל מכונה המתגרמת תנועה בין רשתות או פרוטוקולים. זהו שרת המפריד בין לקוחות משתמשי קצה לבין היעדים בהם הם גולשים. שרתי ה-proxy יכולים לספק רמות שונות של פונקציונליות, אבטחה ופרטיות בהתאם למקרה השימוש שלך, לצרכים או למדיניות החברה.

כיצד פועל שרת proxy?

לכל מחשב יש כתובת IP ייחודית, כפי שהסברנו [כאן](#). הכתובת היא המזהה של מחשב שלך, ואיתה השרת שאליו אתה שולח הודעה יודע להחזיר לך את התשובה.

שרת proxy הוא בעצם מחשב עם כתובת IP משלו שהמחשב שלך מכיר. כשאתה שולח בקשת אינטרנט, הבקשה שלך תחילה תגיע לשרת proxy. לאחר מכן, שרת ה-proxy מבקש את בקשת האינטרנט שלך בשמך, אוסף את התגובה משרת האינטרנט ומעביר לך את נתוני דף האינטרנט שתוכל לראות את הדף בדפדפן שלך.

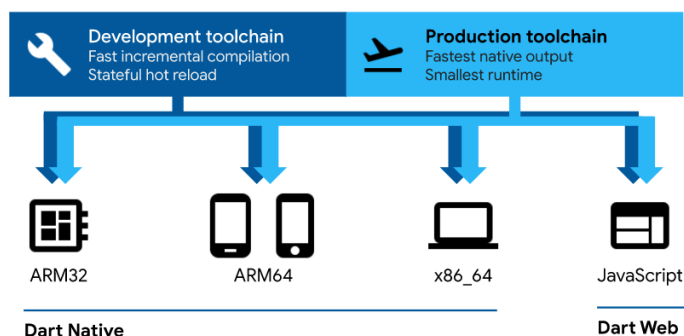
כאשר שרת ה-proxy מעביר את בקשות האינטרנט שלך, הוא יכול לבצע שינויים בנתונים שאתה שולח ועדיין להביא לך את המידע שאתה מצפה לראות. שרת proxy יכול לשנות את כתובת ה-IP שלך, כך ששרת האינטרנט לא יודע בדיוק היכן אתה נמצא בעולם. השרת יכול להצפין את הנתונים שלך, כך שהנתונים שלך אינם קריאים במעבר. ולבסוף, שרת proxy יכול לחסום גישה לדפי אינטרנט מסוימים, בהתבסס על כתובת ה-IP.

מבוא לשפת Dart

Dart היא שפה המותאמת לפיתוח אפליקציות מהירות בכל פלטפורמה. Dart היא שפת תכנות חדשה אשר נוצרה ופותחה על ידי גוגל בשנת 2011. השפה היא שפה מונחת עצמים, ובעלת אופן קוד דומה לשפת C. בשנים האחרונות dart היא אחת מהשפות הפופולריות ביותר בעולם לייצוא אפליקציות לאנדרואיד, iOS ול web (GitHub קבע ששפת Dart היא השפה בעלת קצב הצמיחה הגבוה ביותר לשנת 2019). הסיבה העיקרית לפופולריות של Dart היא העבודה שגוגל בנו בעזרתה את [Flutter](#) – ספרייה המאפשרת לתכנת לאנדרואיד, iOS ול web בצורה נוחה במיוחד.

dart יכולה להתקמפל בשני דרכים עיקריות:

- **Native platform** – אפליקציות למכשירים ניידים ושולחניים. dart כוללת Dart VM עם just-in-time (JIT) קומפיילר ו-ahead-of-time (AOT) קומפיילר. השפה מתורגמת לשפת מכונה – ARM32, ARM64, x86_64.
 - Just-in-time קומפיילר - מאפשר גישה מהירה למפתח לגשת למדדים חיים של האפליקציה בעת פיתוח – מאפשר טעינה חמה (מאתחל את האפליקציה בלחיצה אחת), הפעלת Dev Tools (כלים לראיית הזיכרון, CPU ועוד).
 - Ahead-of-time קומפיילר – מאפשר קימפול מראש של קוד מחשב (ARM) או קוד מכונה מקורי (x64). בזכות קומפיילר זה, dart יש זמן פתיחה (פתיחה ראשונית של האפליקציה) קצר ויעיל.
- **Web platform** – לאפליקציות הממוקדות לרשת. dart כולל development-time קומפיילר ו production-time קומפיילר. שני הקומפיילרים מתרגמים את dart לשפת JavaScript.
 - Development-time קומפיילר (dartdevc) – מאפשר להריץ את שפת dart ולנפות באגים בדפדפן chrome.
 - Production-time קומפיילר (dart2js) – אוסף קוד dart לקוד ב JavaScript מהיר, קומפקטי ופרוס באמצעות טכניקות כגון חיסול קוד מת.



תמונה 2 – אופן עבודה של השפה dart נלקח מהאתר flutter.dev

מבוא לFlutter

פלאטר היא ערכת פיתוח תוכנה אשר נועדה לבניית ממשק משתמש לפיתוח אפליקציות עבור אנדרואיד, Web, iOS, Mac, Linux, Windows ועוד. השפה נוצרה ופותחה על ידי גוגל ושחררה לראשונה בשנת 2015 (הגרסה היציבה הראשונה שוחררה רק ב-2018). יש לציין כי פלאטר היא השפת תכנות עם הקצב גדילה הכי גדול בעולם לפי GitHub (כרגע עם 132,000 כוכבים – נמצאת במקום ה-15 ב-GitHub)

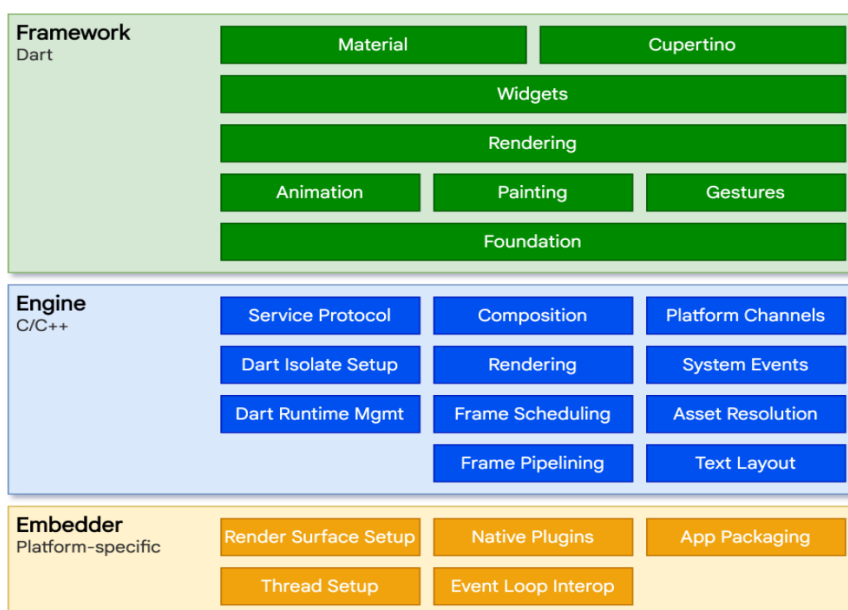
המטרה העיקרית של פלאטר היא לאפשר למתפתחים לספק אפליקציות בעלות ביצועים גבוהים שמרגישות טבעי בפלטפורמות השונות (לעומת שאר שפות התכנות לאנדרואיד, היתרון המרכזי של פלאטר שהיא תיראה ותרגיש בדיוק אותו דבר בכל הפלטפורמות).

פלאטר כתובה ב-dart לכן כל אפליקציה שנכתוב בפלאטר תתקמפל ישירות לשפת מכונה (ARM או x64) או ל-JavaScript.

בשביל להבין את העקרונות של השפה, להימנע משגיאות מיותרות ובשביל לעשות דברים בדרך הכי יעילה שרק אפשר חשוב לדעת מה עומד "מאחורי הקלעים", כלומר חשוב להבין לעומק מה באמת קורה כאשר אנו מריצים תכנית בשפת flutter.

שכבות אדריכליות

פלאטר מתוכננת כמערכת שכבות הניתנת להרחבה. המערכת בנויה מכמה ספריות עצמאיות שכל אחת מהן תלויה בשכבה הבסיסית. לאף שכבה אין גישה מיוחדת לשכבה למטה, וכל חלק ברמת המסגרת נועד להיות אופציונלי וניתן להחלפה.



תמונה 3 – אופן הבנייה של פלאטר לפי שכבות אדריכליות נלקח מהאתר flutter.dev

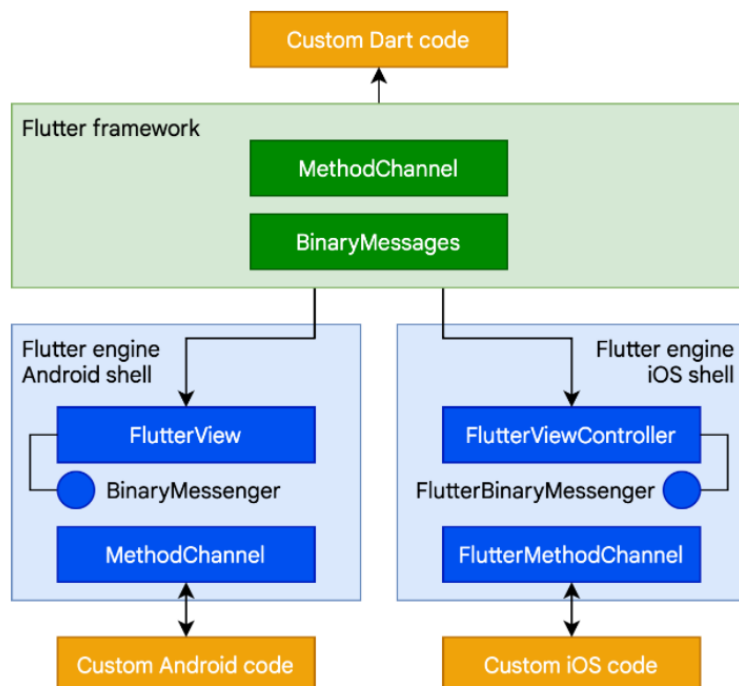
1. **Embedder** – מנהל את הטמעה הספציפית של פלטפורמה מסוימת. מספק נקודות כניסה, מתאם עם מערכת ההפעלה הבסיסית את הגישה לשירותים כמו משטחי עיבוד, נגישות וקלט. Embedder יהיה כתוב בשפה המתאימה לכל פלטפורמה: Java ו-C++ לאנדרואיד, Objective-C או Objective-C++ ל-iOS ול-macOS, C++ ל-Windows או Linux. באמצעות Embedder ניתן לשלב קוד בפלאטר באפליקציה קיימת כמודול, או שהקוד עשוי להיות כל תוכן האפליקציה.
2. **Flutter engine** – בלב ליבה של פלאטר נמצא המנוע של פלאטר שנכתב ברובו ב-C++ ותומך בכלים הדרושים לתמיכה בכל יישומי פלאטר. המנוע כולל גרפיקה, פריסת טקסט, קלט/פלט של קבצים ורשתות, תמיכה, נגישות ארכיטקטורות נוספות וזמן ריצה וערכת כלים של Dart.
3. **Flutter framework** – בדרך כלל, רוב המפתחים מתקשרים עם פלאטר באמצעות flutter framework, המספקת מסגרת מודרנית ותגובתית הכתובה ב-Dart. הוא כולל מערך עשיר של פלטפורמות וספריות יסוד המורכבות מסדרה של שכבות. להלן השכבות מלמטה למעלה:
 1. **Foundation** – מחלקות ושירותים יסודיים כגון אנימציה, ציור, ומחווות יד המציעים הפשטות נפוצות של הבסיס.
 2. **Rendering** – שכבת העיבוד מספקת הפשטה להתמודדות עם פריסה של מחלקות. בעזרת שכבה זו אפשר לבנות עץ של אובייקטים הניתנים לעיבוד. אפשר לתפעל את האובייקטים האלה באופן דינמי, כאשר העץ מעדכן אוטומטית את הפריסה ובכך ישקף את השינויים שנוצרו.
 3. **Widget** - שכבת הווידג'טים היא הפשטת קומפוזיציה. לכל אובייקט עיבוד בשכבת העיבוד יש מחלקה מתאימה בשכבת הווידג'טים. בנוסף, שכבת הווידג'טים מאפשרת לך להגדיר שילובי מחלקות שתוכל לעשות בהן שימוש חוזר. זו השכבה שבה מוצג באופן מוחשי מודל התכנות.
 4. **Material & Cupertino** - מציעות קבוצות מקיפות של פקדים שמשתמשים בפרמיטיביים של שכבת הווידג'ט כדי ליישם את שפות העיצוב של Material או iOS.

שילוב קוד משפה אחרת

פלאטר מספק מגוון מנגנוני יכולת פעולה הדדית, בין אם אתה ניגש לקוד או לממשקי API שנכתבו בשפה כמו Kotlin או Swift, קורא ל-API מבוסס C.

עבור אפליקציות, פלאטר מאפשר לך להכניס לקוד מותאם אישית באמצעות ערוץ פלטפורמה, שהוא מנגנון פשוט לתקשורת בין קוד ה-Dart שלך לבין הקוד הספציפי לפלטפורמה של האפליקציה המארחת שלך. על ידי יצירת ערוץ משותף, תוכל לשלוח ולקבל הודעות בין dart לרכיב פלטפורמה הכתוב בשפה כמו Kotlin או Swift. הנתונים

מסודרים מסוג של Dart כמו מפה לתבנית סטנדרטית, ולאחר מכן מסוללים מחדש לייצוג שווה ערך ב-Kotlin (כגון HashMap) או Swift (כגון מילון).



תמונה 4 – שילוב קוד משפה אחרת ב-flutter נלקח מהאתר flutter.dev

להלן דוגמה פשוטה לערוץ פלטפורמה לשיחת Dart למנהל אירועים קולט ב-Kotlin (אנדרואיד) או Swift (iOS):

```
// Dart side
const channel = MethodChannel('foo');
final String greeting = await channel.invokeMethod('bar', 'world');
print(greeting);

// Android (Kotlin)
val channel = MethodChannel(flutterView, "foo")
channel.setMethodCallHandler { call, result ->
    when (call.method) {
        "bar" -> result.success("Hello, ${call.arguments}")
        else -> result.notImplemented()
    }
}

// iOS (Swift)
let channel = FlutterMethodChannel(name: "foo", binaryMessenger: flutterView)
channel.setMethodCallHandler {
    (call: FlutterMethodCall, result: FlutterResult) -> Void in
    switch (call.method) {
        case "bar": result("Hello, \(call.arguments as! String)")
        default: result(FlutterMethodNotImplemented)
    }
}
```

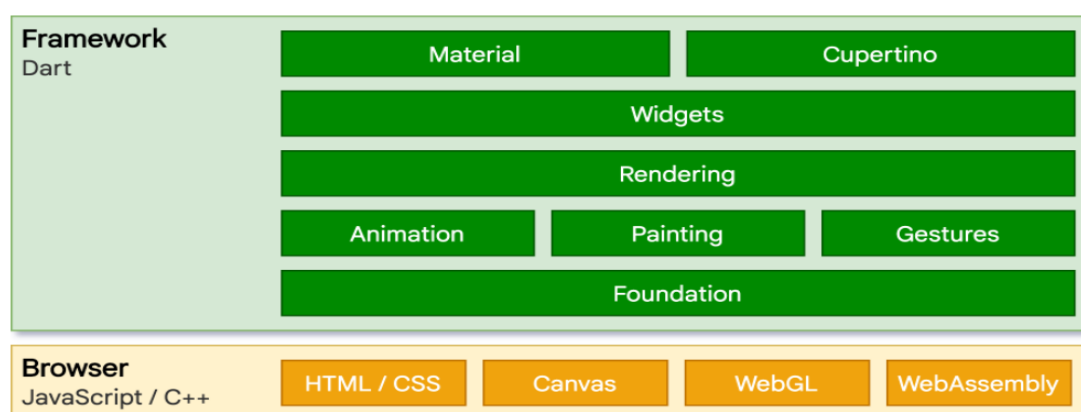
תמונה 5 – דוגמאות לשילוב קוד משפה אחרת ב-flutter נלקח מהאתר flutter.dev

תמיכה ב-Web

Dart אוסף ל- JavaScript כל עוד השפה קיימת, עם כלי עבודה מותאם למטרות פיתוח וייצור כאחד. אפליקציות חשובות רבות נאספות מ- Dart ל- JavaScript ופועלות כיום בייצור, כולל כלי המפרסם עבור Google Ads. מכיוון שמסגרת ה- Flutter כתובה ב- Dart, עריכתה ל- JavaScript הינה פשוטה יחסית.

עם זאת, מנוע פלאטר, הכתוב ב- C++, נועד להתממשק עם מערכת ההפעלה הבסיסית ולא עם דפדפן אינטרנט. לכן נדרשת גישה אחרת. באינטרנט, פלאטר מספקת מימוש מחדש של המנוע על גבי ממשקי API רגילים לדפדפן. כרגע יש שתי אפשרויות לעיבוד תוכן Flutter באינטרנט: HTML ו- WebGL. במצב HTML, פלאטר משתמשת ב- HTML, CSS, Canvas ו- SVG. כדי לעבד ל- WebGL, פלאטר משתמשת בגרסה של Skia שהורכבה ל- WebAssembly בשם CanvasKit. בעוד מצב HTML מציע את מאפייני גודל הקוד הטובים ביותר, CanvasKit מספק את הדרך המהירה ביותר למחסנית הגרפיקה של הדפדפן, ומציע נאמנות גרפית גבוהה במקצת עם היעדים הניידים המקומיים.

השכבה האדריכלית של פלאטר באינטרנט:



תמונה 6 – אופן הבנייה של flutter ל-web נלקח מהאתר flutter.dev

חלק מעשי

```
document.getElementById(div).innerHTML = errorMessage;
else if (i==2)
{
    var atpos=inputs[i].indexOf("@");
    var dotpos=inputs[i].lastIndexOf(".");
    if (atpos<1 || dotpos<atpos+2 || dotpos>inputs[i].length-1)
        document.getElementById('errEmail').innerHTML = "Email is not valid";
    else
        document.getElementById(div).innerHTML = "OK";
}
else if (i==5)
{
    var atpos=inputs[i].indexOf("@");
    var dotpos=inputs[i].lastIndexOf(".");
    if (atpos<1 || dotpos<atpos+2 || dotpos>inputs[i].length-1)
        document.getElementById('errPass').innerHTML = "Password is not valid";
    else
        document.getElementById(div).innerHTML = "OK";
}
```

הצגת תכנון ומבנה הפרויקט

הפרויקט כתוב בשפת dart. לפני תחילת כתיבת הקוד תכננתי את הפרויקט וקבעתי באופן כללי את היעדים שאליהם ארצה להגיע. את הפרויקט תכננתי כך שיהיה נוח להוסיף לו עוד דברים ולשנות פונקציונליות בקלות. בנוסף על כך, הקפדתי על קוד נקי ומסודר.

בתכנות עבדתי בשלבים – קודם כל הבנה עמוקה של אפליקציית משוב (כולל ניתוח הקוד והתעבורה של משוב) ורק לאחר מכן מימוש האפליקציה.

בכל שלב עבדתי מלמטה למעלה, לדוגמה: אם הייתי צריך לבצע תקשורת עם השרת של משוב אז קודם כל בניתי מחלקת תקשורת בסיסית עם האינטרנט ולאחר מכן בניתי לה מעטפת, קלה יותר לשימוש, ואז מעל הכל מחלקה שתכיל את הקישורים הנרשים והפעולות, ותתקשר באופן עצמאי עם המחלקות הנ"ל.

שם האפליקציה – Grader for Mashov (או בקיצור גריידר). גריידר תכיל פונקציות ויכולות רבות, יהיה ניתן להכניס את שם בית הספר, שם המשתמש, הסיסמא (או באמצעות SMS) ולהתחבר לחשבון שלך (שנמצא בשרת של משוב). גריידר ישלח בקשות לשרת של משוב ויהיו לו את כל היכולות של משוב יש. גריידר תציג את הנתונים בצורה יפה יותר, נעימה לעין ויעילה יותר. אפרט על מה בדיוק התוכנה עושה בהמשך באמצעות הסברים ותמונות.

האפליקציה עובדת, וכל הרוצה להשתמש בה יכול להורידה באופן חינמי ב-[Google play](https://play.google.com/store/apps/details?id=com.graderfor) וב-[App store](https://apps.apple.com/us/app/grader-for-mashov/id1444444444) או להשתמש ב-[web](https://web.graderfor.com/). בנוסף, מי שרוצה להשתמש בקוד יכול לעשות זאת גם, הקוד הינו קוד פתוח הנמצא ב-[GitHub](https://github.com/graderfor).

במהלך יצירת הפרויקט הקפדתי על קוד קריא, נוח וגמיש ונעשה שימוש רב באלגוריתמים שיצרתי ובהורשה.

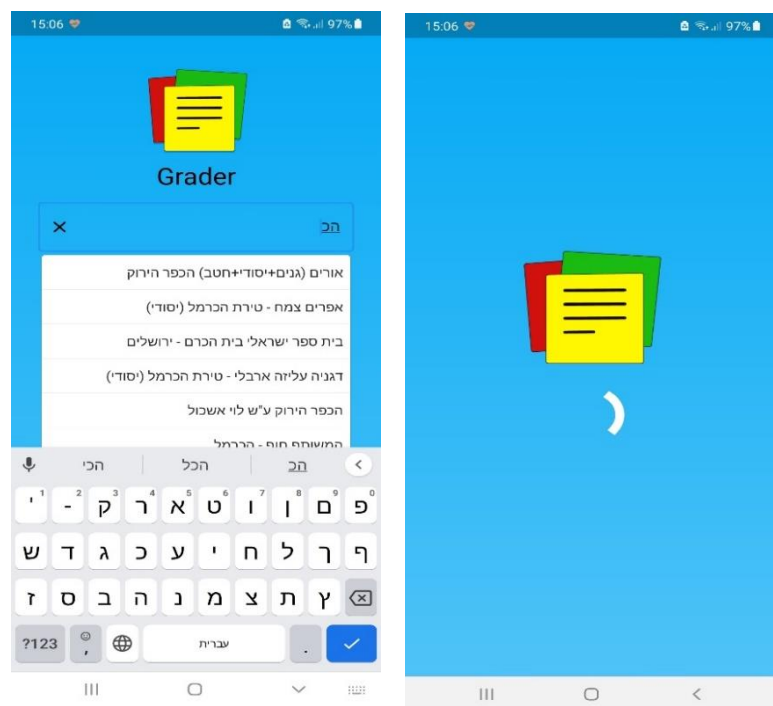
הצגת כלי עבודה למימוש הפרויקט

- **Dex2jar** – כלי עבודה המאפשר לנו לעבוד עם קבצי אנדרואיד עם סיומת של dex ו class. נוכל להעביר בעזרתו את הקבצים עם סיומות הנ"ל למספר אופציות שנרצה.
- **JD-GUI** – כלי גרפי עצמאי המציג קוד מקור של java עם הסיומת class. בזכותו ניתן לעניין בקוד המשוחזר (לאחר שהעברנו את הקובץ לסיומת של jar). בקלות וביעילות.
- **Burp Suite** – תוכנה המאפשרת להקליט את התעבורה של אתר מסוים בתור שרת Proxy. בתוכנה זו השתמשתי הרבה במהלך הפרויקט. במהלך עבודתי, הייתי צריך להבין כיצד עובדת התעבורה בין הלקוח של משוב ([האתר של משוב](#)) לבין השרת של משוב ([השרת של משוב](#)). בזכות התוכנה הצלחתי להבין מה בדיוק headers שצד- הלקוח שולח ומה בדיוק אני צריך לעשות כדי להתחזות לתוכנת משוב.
- **Android Studio** – משמש כ-IDE של מערכת הפעלה באנדרואיד של גוגל, בנויה על תוכנת IDEA ו IntelliJ ותוכנה במיוחד לפיתוח אפליקציות לאנדרואיד. אני אוכל לפתח בעזרתה את גריידר.
- **PyCharm** – משמש כ-IDE לכתיבת קוד בpython. ניצור בעזרתו את השרת של תחרות ממוצע הציונים של גריידר.
- **MongoDB** – דטה בייס זה הוא חינומי ומעניק שטח אחסון בענן. נאחסן בעזרתו את כל המשתמשים שנרשמו לתחרות ממוצע הציונים של גריידר.
- **VMware** – תוכנת הדמיה. נוריד את מערכת ההפעלה של macOS ונריץ אותה על המכונה מדומה. נעשה זאת משום שהרצה של אפליקציה על מכשיר אייפון יכולה להגיע אך ורק ממחשב mac (לא היה לי מחשב mac בבית לכן הפתרון היחידי האפשרי היה מכונה מדומה).

תיאור שלבי העבודה על הפרויקט

1. הבנת אופן הפעולה של הנדסה לאחור של קבצים מסוג – apk.
 - פירוט שיטות של reverse engineering.
 - קריאה והבנה עמוקה על כל אחת ואחת מהשיטות.
 - מימוש אחת מהשיטות על אפליקציית "משוב".
2. חקירה על אפליקציית משוב.
 - פתיחת קוד המקור של משוב ועיון בו.
 - הבנה כיצד משוב פועל מבפנים וכיצד יש לעשות כדי לחקותו.
3. חקירה על התעבורה של אפליקציית משוב עם השרת.
 - קריאה והבנה על פרוטוקול HTTP ועל כיצד עובד Proxy.
 - שימוש ב-burp suite ותיעוד התעבורה בין צד-לקוח לצד-שרת.
 - הבנה של כל אחת ואחת מהבקשות הנשלחות לשרת של משוב.
 - ניסוי וטעייה על ידי שליחת קישורים עם headers שונים כדי לבדוק מה נחוץ ומה לא.
4. הבנה עמוקה של שפת dart ו-Flutter ועקרונות ה-OPP מספרים, סרטונים, מאמרים ואתרי אינטרנט.
 - יצירת אפליקציות פשוטות להבנה של השפה.
5. יצירת API למשוב באמצעות השפה dart.
 - מימוש ראשוני של בקשות למשוב (יצירת מחלקות לבדיקות).
 - קבלת התשובות מהשרת של משוב ופירסור הנתונים למחלקות שונות (שימוש ב-json וב-Regex).
6. יצירת האפליקציה.
 - שמירת הנתונים בזיכרון האישי של הטלפון.
 - עדכון האפליקציה (שינוי State) כאשר הנתונים המבוקשים התקבלו.
 - יצירת עיצוב לאפליקציה והצגה נוחה של הנתונים המובאים משרת המשוב.
 - העלאת האפליקציה ל-Google play, App store ול-web.

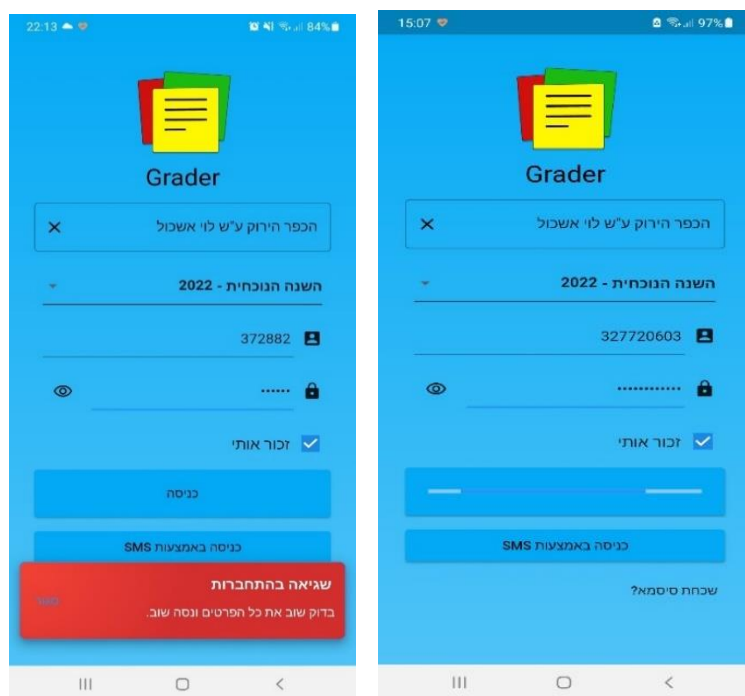
המחשה ויזואלית של המוצר העובד



1. כניסה לאפליקציה:

המשתמש נכנס לאפליקציה, בתחילה האפליקציה טוענת את כל הנתונים השמורים בזיכרון (כגון: האם המשתמש מחובר כבר? האם הוא נמצא במצב כהה? ועוד). לאחר מכן האפליקציה שולחת בקשה ראשונית לשרת המשוב ומבקשת רשימה של בתי הספר. אם המשתמש כבר מחובר, נשלחת בקשה להתחברות למשוב והמשתמש מועבר לעמוד הבית, אם לא, הוא מועבר לעמוד ההתחברות.

תמונה 7 – צילומי מסך ממסך הטעינה וההתחברות מאפליקציית גריידר



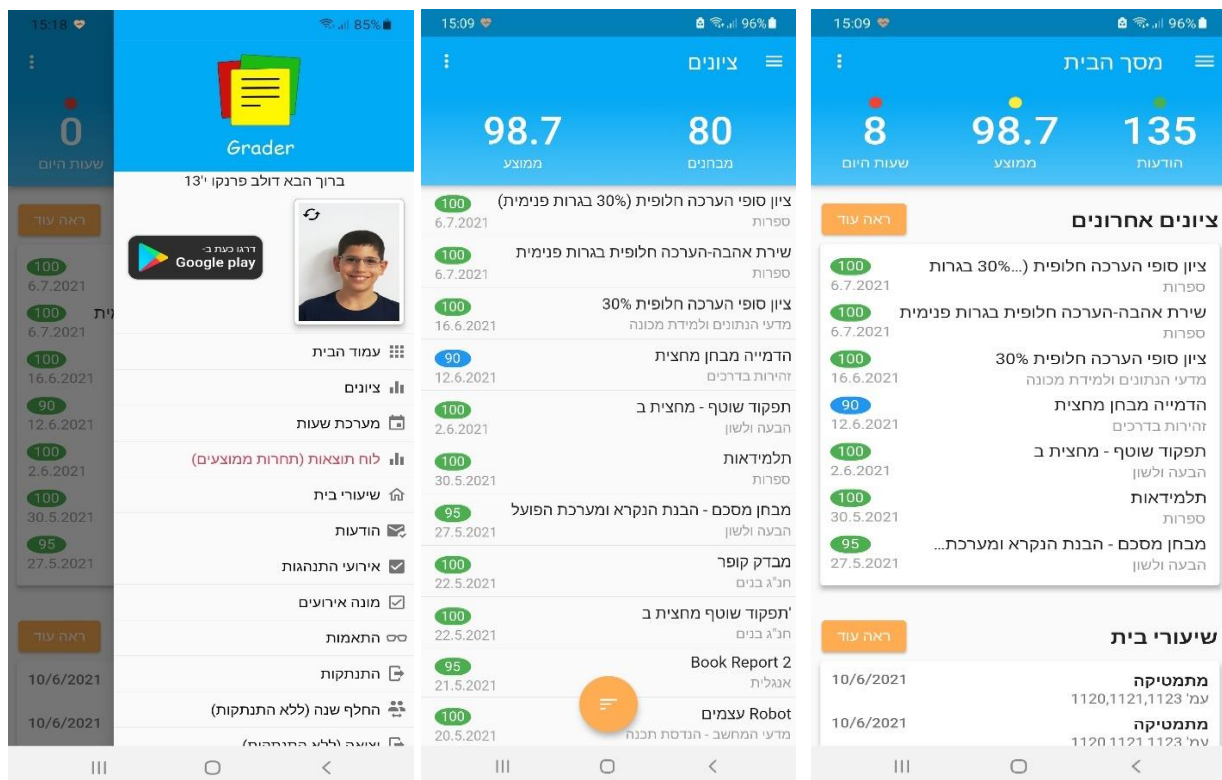
2. התחברות:

על המשתמש לכתוב את שם בית הספר שלו, את שנה אליה הוא רוצה להתחבר ושם המשתמש. יש שתי אפשרויות להתחבר למשוב, אחת עם סיסמא רגילה, השנייה עם מספר הטלפון (במקום סיסמא אתה כותב את המספר הטלפון שלך ומשוב שולח לך קוד כניסה חד פעמי להתחברות). לאחר שהמשתמש מילא את כל הפרטים הנכונים עליו ללחוץ "כניסה", אם אחד מהמפרטים לא נכון, גריידר יראה הודעת שגיאה ויבקש מהמשתמש להכניס את פרטיו שוב. אם מילא נכון, יעבור לעמוד הבית.

תמונה 8 - צילומי מסך ממסך ההתחברות מאפליקציית גריידר

3. האפליקציה :

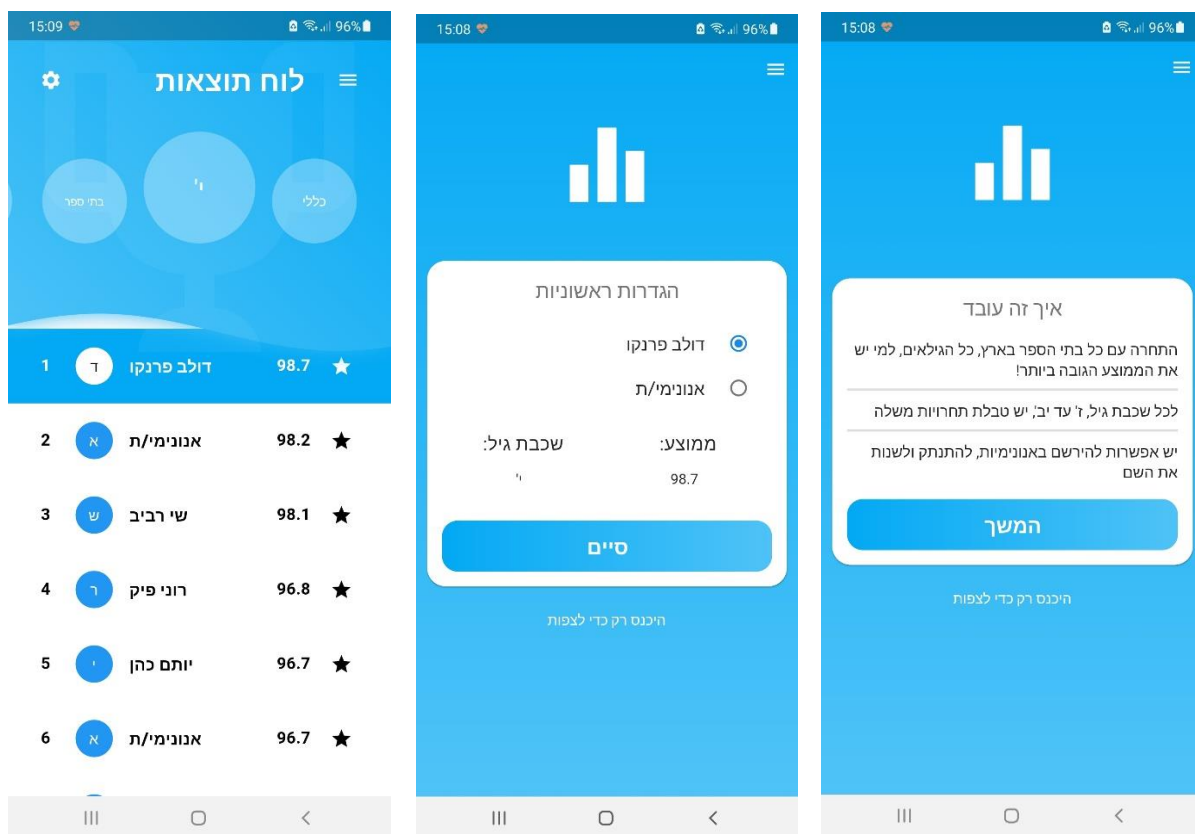
כאשר נכנסים לאפליקציה, מופיע מסך הבית, בו נמצאים הנתונים החשובים שכל תלמיד.ה רוצה לראות – ציונים, שיעורי בית ומערכת שעות יומית. בראש האפליקציה ניתן לראות את הממוצע, מספר ההודעות שהשתמש אינו קרא ומספר שעות לימודיות יומי. בצד ימין של האפליקציה יש מגירת אפשרויות, לכל האופציות שגריידר מציעה: ציונים, שיעורי בית, מערכת שעות, אירועי התנהגות, סיכום אירועי התנהגות, מונע אירועים, הודעות, לוח תוצאות (תחרות ממוצע ציונים) והתאמות.



תמונה 9 - צילומי מסך ממסך הבית מאפליקציית גריידר

4. תחרות ממוצע ציונים :

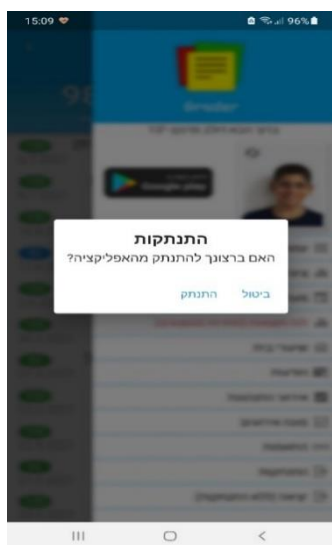
התחרות הינה תחרות בין תלמידים על למי יש את הממוצע הגבוהה ביותר בין כל שכבות הגיל ובתי הספר המשתמשים בגריידר. התחרות אינה קשורה לשרת של משוב לכן אין עלי הפירוט בשאר העבודה. יצרתי שרת הכתוב ב-python בשימוש בספריית Flask. לשרת כמה פונקציות: רישום תלמיד חדש לתחרות, עדכון הממוצע, ניתוק תלמיד מהתחרות, שינוי שמו (אנונימי או שמו האמיתי) – יש גישה לשמו מהתשובה שקיבלנו מההתחברות (למשוב) ומידע על המשתמש – איזה מיקום הוא בשכבת הגיל שלו, איזה מיקום הוא בכללי. ניתן לראות בתמונה 10 שיש כמה אופציות – תחרות כללית בין כל החוברים למשחק, תחרות בין בתי הספר (לאיזה בית ספר יש ממוצע גבוהה ביותר) ולכל שכבת גיל יש תחרות ממוצע ציונים משלה.



תמונה 10 - צילומי מסך מתחרות ממוצע הציונים מאפליקציית גריידר

5. התנתקות:

יש אופציה במגירת האפשרויות להתנתק בכל רגע נתון, והאפליקציה תחזור למסך ההתחלתי שלה – מסך ההתחברות.



תמונה 11 - צילום מסך מאפליקציית גריידר

הערה: האפליקציה לא אוספת שום נתונים על המשתמש בלי הסכמתו, ולא שומרת את הסיסמאות בענן, כמובן שמיד יהיו אנשים שלא יסמכו על האפליקציה, לכן יצאתי קוד פתוח [GitHub](https://github.com). כמו כן – עשיתי הדמייה בסרטון של כל הפיצורים באפליקציה, [לחצו כאן](#).

בעיות בתהליך מימוש האפליקציה

במהלך עבודתי על עבודת הגמר נתקלתי בהרבה אתגרים, הן תכנותיים והן מחשבתיים אשר הקדשתי להן מחשבה רבה. בחלק זה של הפרויקט אתאר את האתגרים שהיו לי במהלך התהליך וכיצד פתרתי אותן.

אתגר 1: בהתחלה כאשר ניגשתי לפרויקט, קראתי על המושגים והתהליכים שרציתי לדעת – רשתות, והנדסה לאחר מהאתרים הרשמיים. נוכחתי לדעת שההסברים אינם נוחים לקריאה ואינם מובנים. למרות קריאה חוזרת ונישנית לא הבנתי מאיפה להתחיל ולא ידעתי מה אני אמור לעשות, מתי וכיצד.

פתרון: לאחר זמן מה של ניסיון קריאה והבנה מן ההסברים הרשמיים, עברתי להסברים הלא רשמיים ולמידה אינטראקטיבית ונוכחתי לדעת כי אני מבין יותר כיצד כל התהליך קורה. בנוסף, הורדתי את הכלים הנדרשים בעצמי והתנסיתי בעצמי בתפעולם.

אתגר 2: כאשר לראשונה התחלתי לכתוב בשפת Dart, כתבתי פונקציית התחברות למשוב אך היא לא עבדה כלל והחזירה לי שגיאה. עברתי מספר פעמים על מה שאני שולח ואיך אני שולח, אך הפעם רק בעזרת הדפסות המשתנים ודיבוג של התוכנה. אמנם כל הבקשות היו על פי הצפוי, בדומה למשוב, אך ידעתי שאם הן היו באמת אותו דבר הפונקציה היית עובדת ויש בעיה.

פתרון: הרצתי את האפליקציה על טלפון מדומה במחשב (דרך אנדרואיד סטודיו), הגדרתי proxy בקוד של גריידר והסנפתי בעזרת Burp Suite את כל מה שגריידר שולח לשרת המשוב. הבנתי כי גוף הבקשה ששלחתי בבקשת postn להתחברות אינו מותאם למבנה של מילון כמו שמקובל לשלוח בבקשות post. כשהגדרתי את המילון באפליקציה ושלחתי אותו לשרת, השתמשתי בפונקציה toString במקום להשתמש בjson.encode. תיקנתי מיד את הטעות והפונקציה עבדה כצפוי. שמחתי על כך שלמדתי באתגר זה משהו חדש ולמדתי שיטה להתגבר עליו.

אתגר 3: כשהתחברתי למשוב, דרך הפונקציה שכתבתי, משוב שלך הודעה למייל שלי: "כניסה למשוב ממשיך חדש, נא לבדוק שהפעולה נעשתה על ידך...". חשבתי ששלחתי משהו לא בסדר ועברתי מספר פעמים על הבקשה שלי לראות שהיא אותו דבר והיא אכן נראתה לי אותו דבר.

פתרון: עברתי על כל headers שמשוב שולח לשרת שלו ברגע ההתחברות, הבנתי ששכחתי את פרמטר ה-uniqueID ובשל כך, משוב כל פעם חושב שאני משתמש חדש ולכן הוא שולח מייל אזהרה. התחלתי לשמור את ה-uniqueID בזיכרון הטלפון ושלחתי אותו בכל פעם שהמשתמש נכנס לגריידר, כך השרת לא יזהה אותו כמשתמש חדש. פתרון זה עבד ומשוב כבר לא שלח מיילים בכל כניסה, אלא רק בכניסה הראשונה כשפרמטר ה-uniqueID עדיין לא היה ידוע.

אתגר 4: התשובות ששרת המשוב מחזיר מורכבות ממילונים עם תוכן שונה, אך בסידור דומה. יכולתי כמובן ליצור פונקציה נפרדת לכל מילון ולנתח בצורה יחידנית ושונה כל בקשה אך לא רציתי ליצור כפילויות קוד לא נחוצות.

פתרון: כתבתי פונקציה הנקראת process אשר מקבלת מחלקה שאליה אנו רוצים לפרסר את התשובה. הפונקציה תחילה מנתחת את התשובה עם אותו מבנה אך כאשר היא מגיעה למילון היחידני ששונה בכל תשובה ותשובה (כאשר שמות הפרמטרים שונים), היא משתמשת במחלקה שהיא קיבלה ומעבירה את התשובה המנותחת אליה.

אתגר 5: כשרצייתי לפתח ולהריץ את האפליקציה על איפון נוכחתי לדעת שניתן לעשות זאת אך ורק ממחשב מסוג mac של אפל, ולי יש מחשב ווינדוס. לאף אחד מבני משפחתי אין מחשב מסוג זה, וגם לא מכשיר איפון שהוא אומנם אופציונלי, כלומר יש אפשרות לעשות הדמיה על מחשב – (סימולטור) אך עדיין עדיף עם מכשיר פיזי.

פתרון: בדקתי וניסיתי עשרות פתרונות והבנתי שהורדת הכלי VMware (מכונה מדומה) ואת מערכת ההפעלה macOS הינו הפתרון הטוב ביותר. נתקלתי בעשרות בעיות, קריסות ועדכוני התוכנה אך לאחר כמה ימים של עבודה רצופה בהתמדה הצלחתי להתקין על המחשב שלי macOS שעובד. הורדתי אנדרואיד סטודיו Xcode והתחלתי לעבוד ולהתאים את הקוד ל-iOS.

אתגר 6: בתחרות ממוצע ציונים, היה צורך ליצור לכל משתמש קוד זיהוי משלו כדי שהוא ידע איזה מקום הוא בתחרות ממוצע ציונים (אי אפשר לשמור את השם שלו כי יכול להיות כפילויות בשם וגם יש אופציה להירשם כאנונימיים). בנוסף לכך, רצייתי גם שהתחרות תשמור את הממוצעים של כל השנים של התלמיד לכן נדרשתי ליצור קוד זיהוי יחיד לתלמיד בלי קשר לשנה, לשמו ולבית ספרו.

פתרון: נזכרתי כי משוב כבר בנה userId לכל תלמיד אך לא ידעתי האם הוא שונה מתלמיד לתלמיד או האם הוא נשמר לאורך כל השנים. בדקתי עם חשבונות של חמישה חברים שלי ונוכחתי לדעת שהuserId רנדומלי בין כל אחד ונשמר לאורך השנים לכן החלטתי להשתמש בו בתור מזהה התלמיד.

אתגר 7: בתחרות ממוצע ציונים, הייתי צריך ליצור שרת כדי לנהל התחברויות, התעדכנויות ועוד. כמו כן, הייתי צריך להשתמש במאגר מידע אינטרנטי (מאוחסן בענן) כדי לשמור את כל התלמידים בתחרות. כלומר היה צריך למצוא הייתי צריך למצוא מאגר מידע חינוכי שעובד טוב עם שפת פייתון (שפת השרת) ויענה על צרכים אילו.

פתרון: לאחר חיפושים וניסיונות רבים, נוכחתי לדעת כי השימוש ב-mongoDB הוא הפשוט ביותר, המהיר ביותר והיעיל ביותר. יצרתי קישוריות בין השרת למאגר וכל פעם שיש בקשה לשרת, מאגר מידע זה היה מעורב בכך.

אתגר 8: לאחר שיצרתי את האפליקציה לאנדרואיד ולאייפון, רצייתי להרחיב אופקים ולהכין את גריידר גם לweb. בפעם הראשונה כשהרצתי אותה על הענן (לאחר התאמת הקוד) קיבלתי שגיאה כאשר ניסיתי לגשת לשרת המשוב:

"Access to fetch at https... from origin ... has been blocked by CORS policy: No Access control allow origin header"

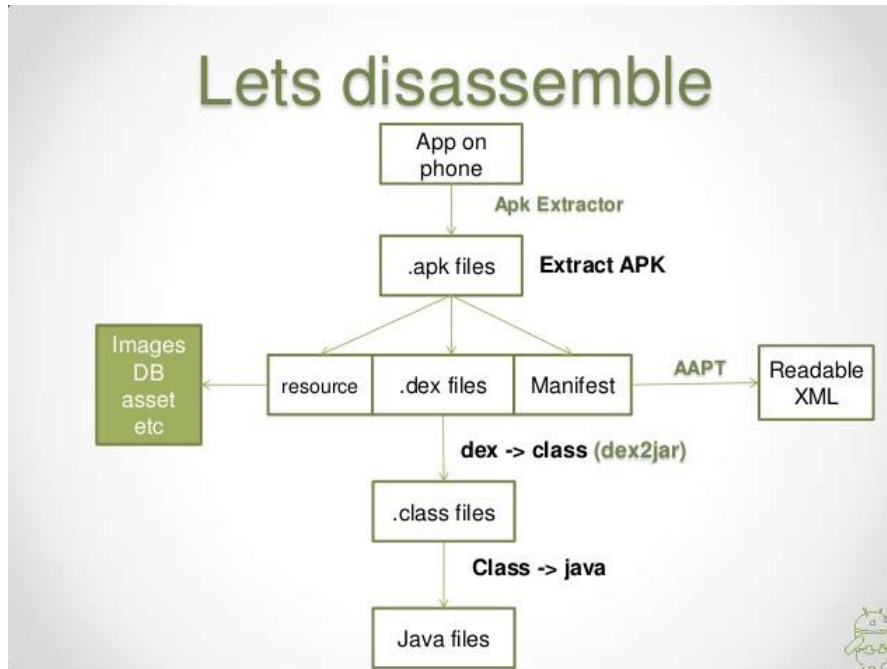
ניסיתי לטפל בבעיה זו, חיפשתי וחקרתי מספר רב של ימים וניסיתי מספר פתרונות שנראו לי נכונים אך שום דבר לא עבד. הבנתי את הבעיה לעומק והבנתי כי אין פתרון למעבר ישירות בין האתר שלי לבין השרת של משוב. הבעיה הייתה שלא ניתן לגשת לשרת מדומיין³ אחד כשאתה נמצא בדומיין אחר.

פתרון: יצרתי בפייתון שרת (שונה משרת תחרות ממוצע הציונים) הנקרא שרת פרוקסי. שרת פרוקסי הוא שרת אשר מקבל פעולה מסוימת ודומיין (לדוגמה בקשת get עם הפרמטרים age=3 לדומיין www.example.com), שולח את אותה בקשה בעצמו ומחזיר את התשובה למי ששלח לו את הפקודה. כעת כל מה שאני שולח באתר של גריידר, מועבר לשרת פרוקסי, נשלח לשרת של משוב והתשובה מוחזרת על ידי שרת פרוקסי לגריידר.

³ שם ייחודי המזהה אתר כגון: www.google.com

שלב ראשון – הנדסה לאחור

בשלב הראשון של הפרויקט ביצעתי הנדסה לאחור לקובץ apk של אפליקציית משוב. לאחר מכן קיבלתי גישה לקוד המקור של משוב ויכולתי להבין טוב יותר כיצד משוב עובד ואיך אוכל לחקות את פעולתו לאפליקציה משלי.



תמונה 10 – שלבי הנדסה לאחור של קובץ apk מתוך גוגל

הורדת קובץ apk ושינוי הסימט zip

הורדתי את קובץ apk של משוב דרך האתר [Apk pure](#).

פתחתי את סייר הקבצים במחשב, ואראה שכעת הורד הקובץ - Mashov.apk. החלפתי את סימט apk לסימט zip כדי לקבל את הקובץ classes.dex שבו נמצא קוד המקור של משוב. חילצתי את הקובץ וראיתי את הקבצים הבאים :

	תיקיית קבצים	14/08/2021 11:18	assets
	תיקיית קבצים	14/08/2021 11:18	META-INF
	תיקיית קבצים	14/08/2021 11:18	res
13 KB	XML Document	14/08/2021 11:18	AndroidManifest.xml
2,547 KB	קובץ DEX	14/08/2021 11:18	classes.dex
1 KB	קובץ PROPERTIES	14/08/2021 11:18	firebase-common.properties
1 KB	קובץ PROPERTIES	14/08/2021 11:18	firebase-iid.properties
1 KB	קובץ PROPERTIES	14/08/2021 11:18	firebase-iid-interop.properties
1 KB	קובץ PROPERTIES	14/08/2021 11:18	firebase-measurement-connector.proper...
1 KB	קובץ PROPERTIES	14/08/2021 11:18	firebase-messaging.properties
1 KB	קובץ PROPERTIES	14/08/2021 11:18	play-services-base.properties
1 KB	קובץ PROPERTIES	14/08/2021 11:18	play-services-basement.properties
1 KB	קובץ PROPERTIES	14/08/2021 11:18	play-services-stats.properties
1 KB	קובץ PROPERTIES	14/08/2021 11:18	play-services-tasks.properties
161 KB	קובץ ARSC	14/08/2021 11:18	resources.arsc

תמונה 11 – הקבצים שנמצאים לאחר שהחלפנו את הסיומת של apk בzip

} הקבצים בהם יש את כל התמונות וקבצי הxml. בהמשך גיליתי שיש בהם דברים נוספים להבנת פעולת משוב.

} הקובץ שמכיל את כל הקוד – כפי שניתן לראות הוא גם הכי גדול, ועליו אעבוד. קובץ זה שסיומת שלו היא dex מכיל את הקוד המקומפל ורץ על פלטפורמת אנדרואיד.

} כל שאר הספריות החיצוניות שמשמש בהם לדוגמה: פיירבייס, או שירותי גוגל פליי.

מקובץ dex לקובץ jar

בשלב זה לקחתי את קובץ ה- classes.dex מהשלב הראשון, והפכתי אותו לקובץ מסוג jar כדי שיהיה זמין לקריאה. נעזרתי בכלי הנקרא [dex2jar](#).

מתוך כל הכלים שהורדתי השתמשתי רק באחד מהם הנקרא d2j-dex2jar. הוא יעזור להמיר מקובץ עם סיומת dex לקובץ עם סיומת jar.

פתחתי cmd⁴ והרצתי את הפקודה הבאה:

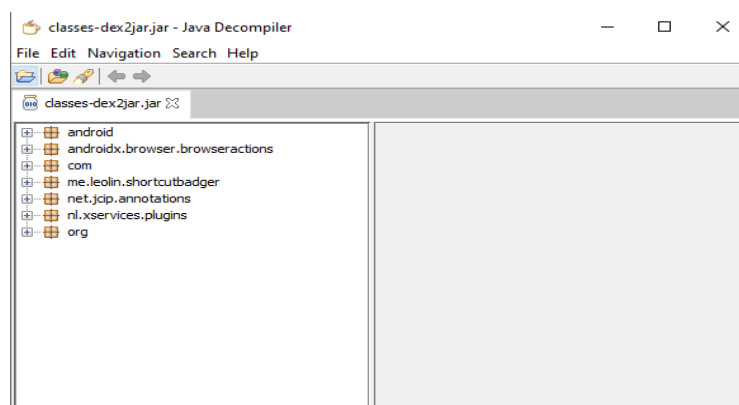
```
d2j-dex2jar.bat classes.dex
```

נוצר קובץ חדש – classes-dex2jar.jar:

2,900 KB Executable Jar File 14/08/2021 11:42 classes-dex2jar.jar

פתיחת קובץ jar

כדי לפתוח קובץ jar נעזרתי בכלי [JD GUI](#). לאחר שהתוכנה נפתחה, יש אפשרות לייבוא קובץ jar. פתחתי את classes-dex2jar.jar שיצרתי בשלב הקודם וראיתי את הקוד של משוב באופן קריא ומובן.



תמונה 12 – צילום מסך מתוך התוכנה Jd-Gui

⁴ ממשק שורת פקודה של מערכות הפעלה הכוללות גם ווינדאוס.

נכנסתי לתיקייה : com.mashov.main ושם מצאתי את MainActivity.java, הקובץ הראשוני של אפליקציית אנדרואיד.

```
package com.mashov.main;

import android.os.Bundle;
import org.apache.cordova.CordovaActivity;

public class MainActivity extends CordovaActivity {
    public void onCreate(Bundle paramBundle) {
        super.onCreate(paramBundle);
        paramBundle = getIntent().getExtras();
        if (paramBundle != null && paramBundle.getBoolean("cdvStartInBackground", false))
            moveTaskToBack(true);
        loadUrl(this.launchUrl);
    }
}
```

תמונה 13 – צילום מסך של הקובץ MainActivity.java מתוך התוכנה Jd-Gui

ניתן לראות שלא הרבה קורה בקובץ הראשוני, אך יש משהו יוצא דופן: שימוש בספרייה הנקראת **Cordova**. Cordova היא ספריית אנדרואיד הנותנת אפשרות להריץ קוד ב HTML, CSS, JavaScript ל Java לאנדרואיד - בעצם קוד של web.

הבנתי שרוב הקוד של משוב לא ימצא ב com.mashov.main אלא בקבצים עם סיומת של html או של js (JavaScript). חיפשתי בקבצים של com.microsoft.cordova אזכר לכל קובץ המותאם ל web.

לא מצאתי שום אזכר לקישור של לקובץ, אך מצאתי את הקישור לתיקייה : file:///android_asset/www/.

```
private static final String WWW_ASSET_PATH_PREFIX = "file:///android_asset/www/";
```

תחילה חשבתי שזה מבוי סתום אך החלטתי לבדוק אזכר זה. חזרתי לתיקייה של משוב שהורדתי מקודם, נכנסתי לתיקיית assets ואז ל www ומצאתי שם מספר קבצים מסוג html ו js עם אזכרים לספריית Cordova.

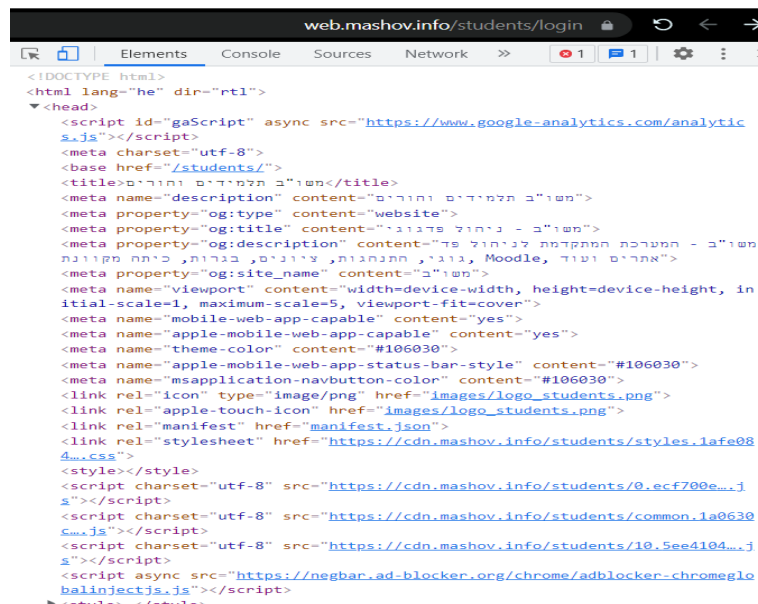
	תיקיית קבצים	14/08/2021 11:18	cordova-js-src
	תיקיית קבצים	14/08/2021 11:18	css
	תיקיית קבצים	14/08/2021 11:18	i18n
	תיקיית קבצים	14/08/2021 11:18	images
	תיקיית קבצים	14/08/2021 11:18	js
	תיקיית קבצים	14/08/2021 11:18	plugins
	תיקיית קבצים	14/08/2021 11:18	scripts
	תיקיית קבצים	14/08/2021 11:18	vendor
62 KB	JavaScript File	14/08/2021 11:18	cordova.js
15 KB	JavaScript File	14/08/2021 11:18	cordova_plugins.js
3 KB	Chrome HTML Do...	14/08/2021 11:18	index.html
3 KB	Chrome HTML Do...	14/08/2021 11:18	index-app.html
1 KB	קובץ JSON	14/08/2021 11:18	manifest.json

תמונה 14 – הקבצים שיש בתוך קובץ ה assets/www של משוב

הורדתי תוכנה המאפשרת לראות קבצי `jsi html` בצורה יותר נוחה – `sublime`, והבנתי שהקוד המוצג הוא אותו קוד של אתר משוב.

[illegible]

תמונה 15- הקוד בindex.html (מתוך אפליקציית משוב)



תמונה 16- הקוד באתר של משוב לאחר שלוחצים F12 (ראיית הקוד של האתר)

ניתן לראות שיש הרבה דמיון בין שני קטעי הקוד.

הערות:

- השפה עברית לא מוצגת טוב ב-sublime חלק מהקוד הכתוב בעברית הוא: "משוי"ב לתלמידים ולהורים." – כמו באתר משוב המקורי.
- ניתן לראות את הקישורים שמשוב שולח לשרת דרך קבצי js שמוצגים בתיקייה הנ"ל אך זוהי דרך יותר מסובכת וקשה להבנה ולכן החלטתי להשתמש בשיטה אחרת.

כעת כאשר הובן שאפליקציית משוב כתובה עם הקוד של אתר משוב, ניתן לעבור לשלב הבא של העבודה: הבנת התעבורה בין אתר משוב (צד-לקוח) לבין שרת משוב (צד-שרת).

שלב שני – הבנת התעבורה של משו

כדי להבין את התקשורת נעזרתי בתוכנת Burp Suite.

הערה: בכל התמונות שאציג מתוכנת Burp Suite, רואים בצד שמאל את הבקשה לשרת משו, ומצד ימין את התשובה (ראה תמונות 17-22).

הבנה בסיסית של אופן פעולת משו

תחילה, מופיע דף התחברות בו צריך להזין את שם בית הספר, שם המשתמש (ת.ז.) וסיסמא. יש כמה אפשרויות להתחברות: סיסמא, או בעזרת קוד ממספר הטלפון או בעזרת קוד מהאימיל.

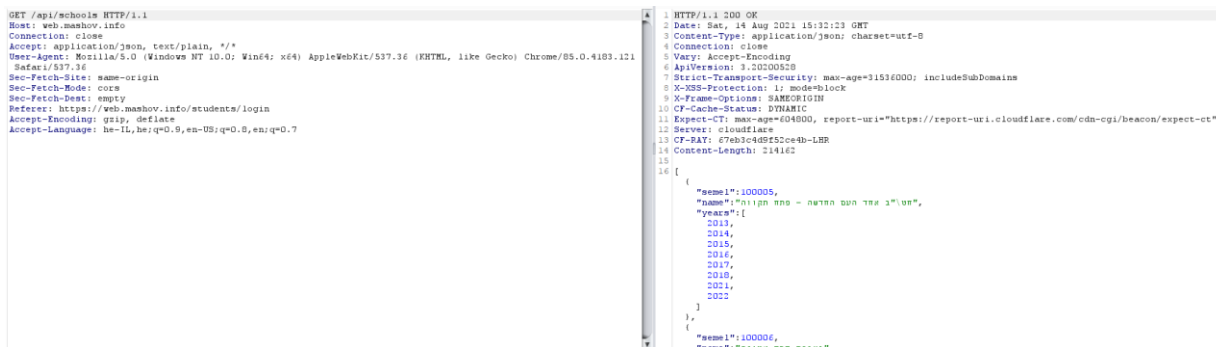
לאחר הכניסה (בהנחה שאין שגיאות), המשתמש מועבר לדף הבית בו יש לו כמה נתונים חשובים כמו מספר הודעות, אירועי התנהגות וציונים אחרונים.

בצד ימין למעלה יש מגירת אפשרויות, בהן ניתן לגשת לכל האפשרויות שמשוב מציע: ציונים, אירועי התנהגות, התאמות, שיעורי בית, מערכת שעות, מונה אירועים ועוד.

התחברות למשו

פתחתי את Burp Suite ואת דפדפן proxy של כרום. לאחר מכן נכנסתי לאתר של משו וניטרתי את בקשות משו מיד מהכניסה אליו.

לראשונה כשנכנסים לאתר, האתר שולח בקשת ⁵get לקישור של ה-API של משו (זו כתובת השרת) ומקבל חזרה את רשימת בתי הספר הרשומים למשו.



```
GET /api/schools HTTP/1.1
Host: web.mashov.info
Connection: close
Accept: application/json, text/plain, */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://web.mashov.info/students/login
Accept-Encoding: gzip, deflate
Accept-Language: he-IL,he;q=0.9,en-US;q=0.8,en;q=0.7

1 HTTP/1.1 200 OK
2 Date: Sat, 14 Aug 2021 15:32:23 GMT
3 Content-Type: application/json; charset=utf-8
4 Connection: close
5 Vary: Accept-Encoding
6 Ap-Version: 3.20200328
7 Strict-Transport-Security: max-age=31536000; includeSubDomains
8 X-XSS-Protection: 1; mode=block
9 X-Frame-Options: SAMEORIGIN
10 CF-Cache-Status: DYNAMIC
11 Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
12 Server: cloudflare
13 CF-RAY: 67eb3c4d5f52ce4b-LHR
14 Content-Length: 214162
15
16 {
  "seme1":1000005,
  "name":"בית הספר החדשה - פתח תקווה",
  "years":[
    2013,
    2014,
    2015,
    2016,
    2017,
    2018,
    2021,
    2022
  ]
},
  "seme1":1000006,
  "name":"בית הספר החדשה - פתח תקווה"
}
```

תמונה 17- צילום מסך של בקשת למשו לקבלת רשימת בתי הספר מתוך התוכנה Burp Suite

רשימת בתי הספר מיוצגים על ידי רשימת מילונים⁶ - לכל בית ספר יש מילון משלו. המילון מכיל את שם בית הספר, סמל המוסד ושנות פעולתו.

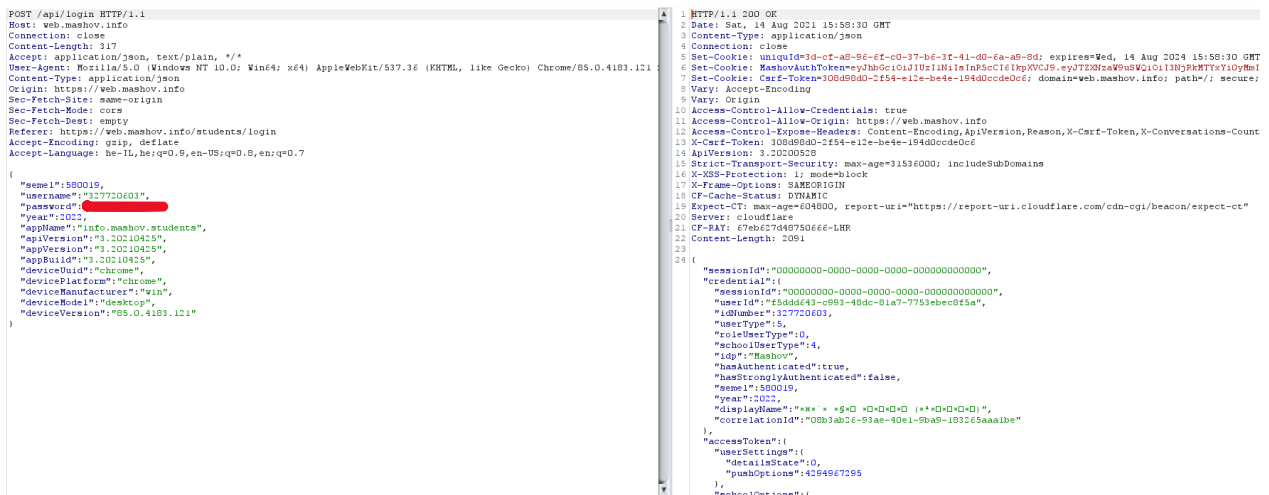
על המשתמש למלא את כל הפרטים, וללחוץ "כניסה".

⁵ בקשת get היא הנפוצה ביותר ברחבי האינטרנט ומשתמשים בה כאשר רוצים לבקש מידע ממקום מסוים.

⁶ מילונים משמשים לאחסון ערכי נתונים בזוגות של - מפתח: ערך.

בשביל להתחבר, משוב שולח בקשת ⁷post לשרת, ובכך מבקש לחבר את המשתמש שהוזן. הוא שולח בתוך גוף הבקשה את שם המשתמש, הסיסמא, סמל בית הספר, מאיפה הוא מתחבר ועוד (בלי הצפנה מכל סוג שהיא). לאחר קבלת התשובה, השרת מחזיר 2 דברים חשובים:

1. בגוף התשובה נמצא הuserID של המשתמש. מזהה זה משמש לפעולות במשוב. בנוסף לכך גוף התשובה מכיל גם את הפרטים האישיים שיש למשוב על המשתמש: שם המשתמש, מגדר, כמה שנים לומד בבית ספר מסוים, שכבת גיל, מספר כיתה ועוד.
2. בראש התשובה (Headers) נמצא את כל פרטי cookies (כדי לשמור על המשתמש מחובר לאותו session⁸) שמשמש בו על מנת לזהות את המשתמש המחובר. כדי לגשת לקישורים הבאים של משוב ולהישאר ב-session אחד (שבו אנו כבר מחוברים למשתמש) יש צורך לשמור את כל cookies המופיעים בראש התשובה על ידי האפליקציה, בזיכרון הטלפון, על מנת להכניס את המשתמש פעם שנייה בלי שיצטרך לכתוב שוב את פרטיו האישיים.



תמונה 18- צילום מסך של בקשת התחברות למשוב מתוך התוכנה Burp Suite

לאחר שנוצר חיבור ויש את מזהה המשתמש, בזכותו נוכל לגשת לפרטי המשתמש האישיים כמו: ציונים, מערכת שעות ועוד.

חשוב לציין כי מעכשיו ישלחו בכל הבקשות כל קבצי cookie ששמרנו מהכניסה של המשתמש: unquid, MashovAuthToken, Csrf-token כדי להגיד לשרת: "הנה המזהה שלי, אני כבר מחובר, שלח את פרטיי".

⁷ בקשת post שימושית כאשר יש צורך לשלוח או לעלות קובץ או מידע לשרת מסוים.

⁸ החלפת מידע קבוע בין השרת ללקוח. כאשר אנו מחוברים ונמצאים בששן מסוים, השרת יודע זאת, וכבר בבקשה הבאה הוא ניגש לדפים הנראים רק על ידי משתמש ספציפי.

קבלת הנתונים של המשתמש

רוב הנתונים של התלמיד מבוקשים ומקובלים באותה דרך (רק הקישור הינו שונה). להלן דוגמא של קבלת ציוני המשתמש :

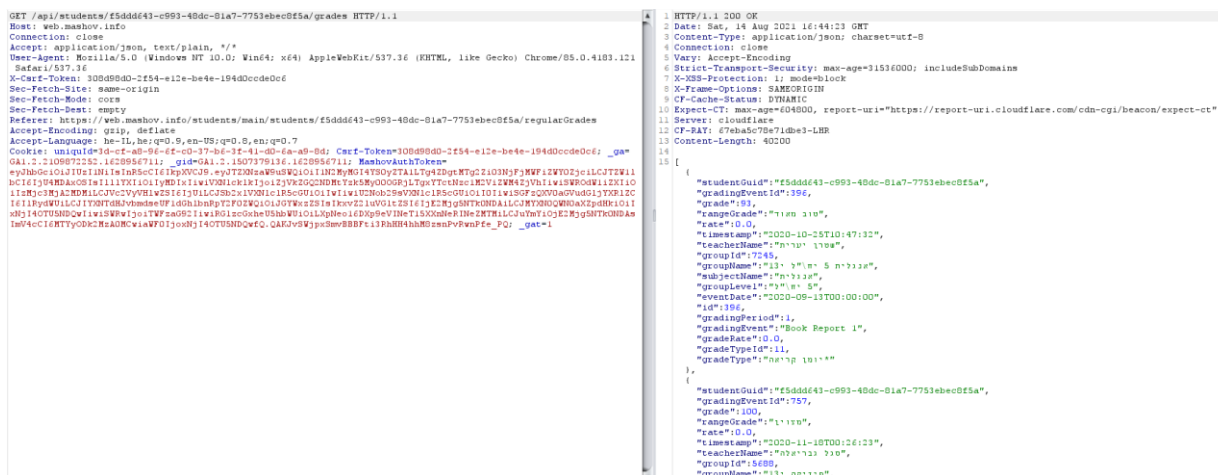
הקישור אליו צריך לשלוח בקשת get מורכב משלושה פרמטרים:

- **הדומיין של שרת המשוב:** `web.mashov.info/API/students/`
- **ה id של המשתמש** (במקרה הנוכחי: `f5ddd643-c993-48dc-81a7-7753ebec8f5a`)
- **הנתון הנחוץ** (לדוגמא: `grades` או `tableTime`).

בראש הבקשה יש לכתוב את הפרמטרים הידועים כמו : Accept-Encoding, Accept, Accept-Language. בנוסף לכך, גם את כל cookies ששמרנו מרגע ההתחברות. כל ה"עוגיות" יכתבו תחת פרמטר Cookien כאשר בניהם נקודה פסיק.

התשובה המתקבלת תהיה מערך המכיל משתנה מסוג מילון המכיל את הנתונים של המשתמש. לדוגמה: קבלת הציונים תכיל מערך המורכב ממילון ציונים, שיכיל: ציון, שם המורה, שם המקצוע, הערה, תאריך, id ועוד.

האפליקציה תפרסר את מערך המילונים למחלקות מתאימות, יורחב על כך בהמשך.



תמונה 19- צילום מסך של בקשה לקבלת הציונים של המשתמש מתוך התוכנה Burp Suite

התנתקות ממשוב

כדי להתנתק ממשוֹב יש לשלוח בקשה שונה במעט משאר הבקשות. הבקשה תשלח לדומיין של השרת ותוסיף בסוף את המילה logout. הבקשה (בקשת get) תכיל בתוכה את כל cookies שאספנו בהתחברות. השרת מבטל את האופציה לקבלת הנתונים דרך cookies שנוצרו בהתחברות, מבטל קוד המזהה (יש לציין כי קוד זה מתבטל באופן אוטומטי לאחר זמן מה כאשר המשתמש לא פעיל). התשובה תכיל את זמני ההתנתקות ואת כל cookies שפג תוקפם.

[illegible]

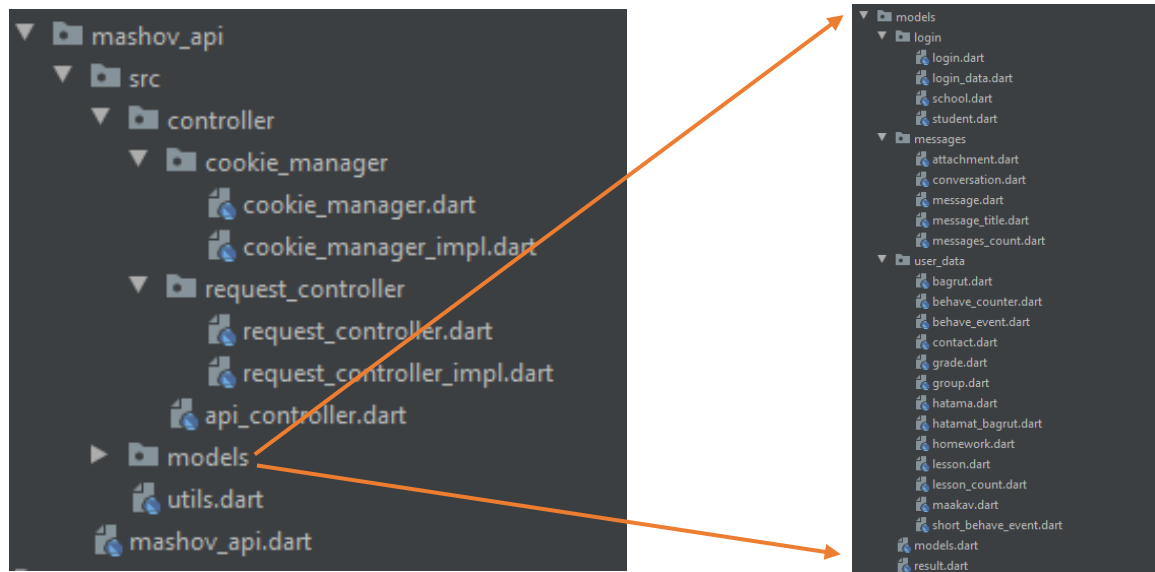
תמונה 17- צילום מסך של בקשה למשוב להתנתקות המשתמש מתוך התוכנה Burp Suite

שלב שלישי – כתיבת API למשוב

כדי לקחת את הנתונים ממשוב, יש ליצור API (מחלקות עזר) שינהלו את כל מה שקורה מרגע ההתחברות למשוב, עד ההתנתקות – שמירת cookies, ניהול שליחת הבקשות, פירסור התשובות המתקבלות ועוד.

מבנה ה-API

כדי שה-API יהיה כתוב בצורה יפה ומסודרת הכנתי קודם את המבנה שלו.



תמונה 20- צילום מסך של המבנה של mashov_api מתוך התוכנה אנדרואיד סטודיו

החלוקה של ה-API מורכבת מכמה חלקים.

1. ניהול שליחת הבקשות - ספריות המנהלות את שליחת הבקשות, ניהול ופירסור נתוני cookie.
2. מחלקות עזר שיעבירו בין נתונים, לדוגמא: לפרסר את headers למחלקות שונות.
3. מחלקות שיכילו בצורה מסודרת את הפרמטרים מהמילונים.
4. מחלקה ראשית – תנהל את כל התכונות של גריידר, תכיל את הקישורים ותבצע תיאום בין כל מחלקות העזר.
5. מחלקה מסכמת – אשר ניתן לייבאה כדי להשתמש ב-API של משוב.

להלן הסבר מפורט על חלקי המבנה:

1. ניהול שליחת הבקשות

שני הקבצים הקשורים בשליחת הבקשות – request_controller.dart, request_controller_impl.dart.

א. request_controller.dart

מחלקה אבסטרקטית (abstract class) בה מבנה בסיסי של כל מה שצריך שיהיה בשליחת הבקשות:

1. פונקציה אבסטרקטית של שליחת בקשת get לכל הבקשות להבאת הנתונים חוץ מהתחברות.

```
Future<http.Response> get(String url, Map<String, String> headers);
```

2. פונקציה אבסטרקטית של שליחת בקשת post להתחברות למשוב.

```
Future<http.Response> post(String url, Map<String, String> headers, Object body);
```

3. יצירת לקוח של ספריית http כדי לשמור את כל הבקשות תחת לקוח אחד.

ב. request_controller_impl.dart

המחלקה מיישמת את הפעולות במחלקה RequestController ומממשת את הפעולות בעזרת ספריית http ובעזרת הלקוח שנבנה קודם.

```
class RequestControllerImpl implements RequestController {
```

2. ניהול ופירסור של נתוני cookie

שני הקבצים הקשורים בניהול נתוני cookie – cookie_manager.dart, cookie_manager_impl.dart.

א. cookie_manager.dart

בדומה לניהול הבקשות, גם כאן יצרתי מחלקה אבסטרקטית בה יש כל מה שצריך בניהול נתוני cookie.

1. כפי שניתן לראות [בהסנפה של משוב](#) צריך לשמור שלושה נתוני cookie משמעותיים – unquid,

.Csrf-token, MashovAuthToken.

2. פונקציה אבסטרקטית processHeaders הלוקחת את נתוני cookie מראש התשובה שמשוב שולח לאחר ההתחברות.

```
void processHeaders(Map<String, List<String>> headers);
```

3. פונקציה אבסטרקטית clearAll שמטרתה לנקות את כל הנתונים השמורים במחלקה – היא נקראת כאשר המשתמש רוצה להתנתק מגריידר או להחליף שנת התחברות.

ב. cookie_manager_impl.dart

המחלקה מיישמת את המחלקה CookieManager ומממשת את נתוני cookie והפונקציות. כדי להשיג את הנתונים מראש התשובה יש צורך בשימוש של פקודות בסיסיות על מחרוזת כגון: split, contains.

```

if (headers.containsKey("set-cookie")) {
  var cookie = headers["set-cookie"];

  uniqueId = cookie
    !.firstWhere((header) => header.contains("uniqueId"))
    .split("=")
    .last;
  mashovAuthToken = cookie
    .firstWhere((header) => header.contains("MashovAuthToken"))
    .split("=")
    .last;
}

```

תמונה 21- צילום מסך של ניתוח ה"עוגיות" של משוב מתוך התוכנה אנדרואיד סטודיו

3. יצירת מחלקת עזר – `utils.dart`

המחלקה מכילה דברים בסיסיים אך נחוצים כגון: בקרה עם נתוני null (שלא יהיו שגיאות מיותרות), עזרה בפירסור `headers`, שילוב מילונים, הפיכת רשימה למחרוזת ועוד. כמו כן, כל הפעולות הן סטטיות.

```

///Returns an empty string if value is null, the value itself otherwise.
static String string(String? value) => value ?? "";

///Returns 0 if value is null, the value itself otherwise.
static int integer(int? value) => value ?? 0;

///Returns false if value is null, the value itself otherwise.
static bool boolean(bool? value) => value ?? false;

```

תמונה 22- צילום מסך של חלק מן הפעולות במחלקה `Utils` מתוך התוכנה אנדרואיד סטודיו

4. יצירת מודלים לכל המילונים השונים

יצרתי מחלקה לכל מילון, כדי לעבוד עם התשובות שמשוב מחזיר (המילונים) בצורה מסודרת. כל מחלקה תכיל כמה פרמטרים בסיסיים:

- משתנים התואמים כל מחלקה (לדוגמא מחלקת בית הספר תקבל `id`, שם בית הספר ושנות פעילות).
- בנאי מחלקה (constructor).
- `toJson` – פונקציה שממירה את המחלקה למילון ומחזירה אותו.
- `fromJson` – מקבל מילון של המחלקה ומחזירה את המחלקה.
- פעולות התואמות כל מחלקה (לדוגמא למחלקת בית הספר תהיה פעולה `getYears` המחזירה מחרוזת של השנים).

בנוסף לכך, יש גם במודלים מחלקה של תוצאה. מחלקה זו תוחזר עם כל תשובה. מחלקת התוצאה תכיל את המחלקה המוחזרת (יכול להיות מחלקת ציונים או מחלקת אירועי התנהגות), סטטוס הקוד (לדוגמא 200 או 400), ושגיאה. כמו כן, ניתן לדעת כל פעם שתקבל תשובה ממשוב האם הבקשה הצליחה או נכשלה, כך יש אפשרות לפתור ולטפל בשגיאות בצורה טובה יותר.

```

class Result<E> {
  dynamic exception;
  E? value;
  int? _statusCode;

  int? get statusCode => _statusCode;

  Result({this.exception, this.value, int? statusCode}) {
    _statusCode = statusCode;
  }

  bool get isSuccess => exception == null && value != null && isOk;
  bool get isUnauthorized => statusCode == 401;
  bool get isInternalServerError => statusCode == 500;
  bool get isNeedToLogin => isUnauthorized || isInternalServerError;
  bool get isForbidden => statusCode == 403;
  bool get isOk => statusCode == 200;
}

```

```

class School {
  int id;
  String name;
  List<int> years;

  String getYears() => Utils.ListToString(years);

  School({required this.id, required this.name, required this.years});

  static School fromJson(Map<String, dynamic> json) => School(
    id: json['semel'], name: json['name'], years: json['years'].cast<int>());

  static List<School> listFromJson(String src) =>
    (json.decode(src) as List)
      .map((school) => School.fromJson(school))
      .toList();

  Map<String, dynamic> toJson() => {'semel': id, 'name': name, 'years': years};
}

```

תמונה 23- צילום מסך של המחלקות School ו Result מתוך התוכנה אנדרואיד סטודיו

5. מחלקה ראשית – api_controller.dart

המחלקה הראשית מכילה את כל התכונות בגריידר – התחברות, קבלת ציונים, קבלת מערכת שעות, התנתקות ועוד.

א. פרמטרים ראשוניים

המחלקה מקבלת בבנייה הראשונה שלה ארבעה פרמטרים – שניים חובה ושניים אופציונליים:

- **cookieManager** (חובה) – המחלקה שמנהלת את נתוני cookie.
- **requestController** (חובה) – המחלקה שמנהלת את הבקשות לשרת.
- **dataProcessor** (אופציונלי) – פונקציה שמקבלת נתון (מחרוזת) וסוג הנתון (ציונים, אירועי התנהגות), אם המשתמש בוחר לממש את הפונקציה, הנתונים המתקבלים על ידי השרת יעברו דרך הפונקציה.
- **rawDataProcessor** (אופציונלי) – פונקציה שמקבלת מחרוזת וסוג הנתון, אם המשתמש בוחר לממש את הפונקציה הנתונים, המתקבלים על ידי השרת יעברו דרך הפונקציה.

בנוסף לכך, למחלקה יהיה עוד משתנה גלובלי – **jsonHeader**. משתנה זה יכיל את הheaders הבסיסים שצריכים להישלח עם כל בקשה כגון: **content-type**, **accept** ועוד. הוא יוגדר אוטומטית ביצירת האיזכור למחלקה.

ב. פעולות בסיסיות

המחלקה מכילה כמה פונקציות בסיסיות שיעזרו בשליחת וקבלת הנתונים.

- **processResponse** – הפעולה תקבל את התשובה של משוב לאחר ההתחברות, תפרסר את הheaders בעזרת מחלקת **utils** ותשלח למחלקת **CookieManager** הוראה לשמור את הנתונים החשובים.

- **authHeader** - פעולה המחזירה את headers שצריכים בכל בקשה. היא תוסיף את פרמטר **jsonHeader** ותוסיף גם את נתוני cookie ממחלקת CookieManager. הפעולה תיקרא בכל פעם שנרצה לשלוח בקשה למשוב.
- **loginHeader** – פעולה נפרדת לקבלת headers להתחברות למשוב, משום שלהתחברות נצטרך headers שונים משאר הבקשות.
- **enum**, מאפשר ליצור טיפוס חדש עם ערכים חדשים שהוא יכול לקבל מחוץ למחלקה. הוא יקרא API ויכיל את כל המחלקות שיש בmodels – schools, grades ועוד.

ג. פעולות מתקדמות

- הפעולות המתקדמות ישמשו לפירסור הנתונים המתקבלים במשוב ולניתוח מילונים למחלקות השונות.
- **process** – הפעולה תקבל פונקציה שעדיין לא נקראה (כלומר שעדיין צריך לחכות עד לתשובה שלה), וAPI. הפעולה תבדוק אם dataProcessor קיים, אם כן היא תחכה עד שהפונקציה שהיא קבלה תיקרא, ואז תיקרא לdataProcessor, אם לא היא תחזיר את הפונקציה שהיא קיבלה בלי שינויים.
 - **authList** – הפעולה תקבל קישור, סוג פירסור (פעולה הלוקחת מילון ומחזירה מחלקה), וAPI. הפעולה תוסיף את headers המתאימים, ותקרא לבקשת get מהrequestController. לאחר שהיא קיבלה תשובה היא תקרא לפעולה parseListResponse שתעבד את התשובה שהיא קיבלה.
 - **parseListResponse** – הפעולה תקבל תשובה (http.Response מספריית http), סוג פירסור, וAPI. הפעולה תהיה עטופה כולה בtry catch (פעולות אלה הם פעולות מובנות בdart ויודעות לזהות ולתפוס כל שגיאה שיש מבלי שהאפליקציה תקרוס). בתוך try נעביר את התשובה שהתקבלה (כרגע היא מחרוזת) למילון בעזרת הפקודה json.decode מספריית dart:convert. לאחר מכן נעבור על כל הערכים במילון ונעביר אותם בעזרת סוג הפירסור שהתקבל למחלקה הרצויה. הכל יוחזר בעזרת מחלקת תוצאה.
 - **auth** – הפעולה תקבל את אותם פרמטרים כמו authList ותעשה את אותם דברים. ההבדל בין שתי הפעולות הוא שאחת מהם תקרא לפעולה המעבדת תשובה מסוג רשימה והשנייה תשובה מסוג שלא מכילה רשימה (כמו קבלת הודעה ממשוב).
 - **parseResponse** – פעולה זו תהיה כתובה כמו parseListResponse פרט ללולאה שעוברת על הערכים של התשובה. כעת לא תהיה לולאה וישר נקרא לסוג הפירסור שהתקבל.

ד. קישורים

כל פרמטרי הקישורים יהיו סטטיים. הקישורים יתחילו בקישור אחד בסיסי - <https://web.mashov.info/api> - הקישור של השרת ויקבלו את הuserId.

```
static String _gradesUrl(String userId) =>
    _baseUrl + "students/$userId/grades";
```

יתר על כן, ישנם גם קישורים שלא צורכים את הuserId, כגון: בקשת ההתחברות וקבלת רשימת בתי הספר.

```
static const String _schoolsUrl = _baseUrl + "schools";
static const String _loginUrl = _baseUrl + "login";
```

ה. פעולות ראשיות

הפעולות הראשיות יכילו את כל התכונות של גריידר וישתמשו בכל פעולות העזר. רוב הפעולות הראשיות יקבלו userId (לקישור) ויחזירו מחלקה או רשימה של מחלקות (לדוגמה בקבלת הציונים תוחזר רשימה של מחלקה מסוג ציון).

ישנן פעולות יוצאי דופן שיכולות לקבל מספר פרמטרים, לדוגמה: בפעולה של התחברות למשוב, הפעולה תקבל את שם המשתמש, הסיסמא, שנת התחברות וסמל בית הספר. לעומת זאת, בפעולה המחזירה רשימה של בתי הספר, לא יתקבלו פרמטרים משום [שהקישור של בתי הספר](#) לא צריך userId.

הערות:

- כל הפעולות יחזירו מחלקה מסוג [תוצאה](#) אשר תכיל את הנתונים המוחזרים על ידי הפעולה.
- בציון המילים "רוב הפעולות" אני מכליל את כל הפעולות חוץ מקבלת רשימת בתי הספר, התחברות, שליחת קוד התחברות, קבלת מסמך מהודעה, קבלת התמונה של המשתמש והתנתקות.
- כל הפעולות יהיו פעולות Future כלומר פעולות מסוג זה נותנות את האופציה לשימוש בawait, שאומר לפונקציה לחכות עד שהסתיימה פעולה שיכולה לקחת זמן (במקרה שלנו שליחה בקשה לשרת).

מבנה רוב הפעולות:

- הפעולות יקבלו מחרוזת userId.
 - לאחר מכן יתקבלו הנתונים ממשוב בסדר דומה:
1. יצירת משתנה headers בשימוש [בפעולה הבסיסית](#) authHeader.
 2. ביקוש בקשת get מהrequestController עם שני פרמטרים: הקישור והheaders.
 3. קבלת מחלקה מסוג תוצאה בשימוש [בפעולה המתקדמת](#) parseListResponse. אפשרות שנייה, שימוש בשתי פעולות מתקדמות: authList וprocess.

4. הפעולה תחזיר את התוצאה.

```
///Returns a List of grades.
Future<Result<List<Grade>>> getGrades(String userId) {
  Map<String, String> headers = _authHeader();

  return _requestController.get(_gradesUrl(userId), headers).then((value) =>
    _parseListResponse<Grade>(value, Grade.fromJson, Api.grades));
}

///Returns a List of behave events.
Future<Result<List<BehaveEvent>>>? getBehaveEvents(String userId) => _process(
  _authList<BehaveEvent>(
    _behaveUrl(userId), BehaveEvent.fromJson, Api.behaveEvents),
  Api.behaveEvents);
```

תמונה 24- צילום מסך של שתי פעולות במחלקה ApiController מתוך התוכנה אנדרואיד סטודיו

מבנה פעולות יוצאות דופן :

1. **קבלת בתי הספר** – הפעולה רק תקרא [לפעולה המתקדמת](#) authList ותחזיר את התשובה : רשימה של מחלקת בית הספר.

2. **התחברות** – הפעולה תקבל סמל בית ספר, שם משתמש, סיסמא, שנת לימוד וuniqueId. בתוך הפעולה ניצור את הבקשה שלנו. לבקשה מסוג post צריך להיות גוף, ניצור את הגוף עם הפרמטרים שקיבלנו ומשתנים נוספים. נשלח בקשת post מהrequestController ונקרא [לפעולה הבסיסית](#) processResponse.

3. **שליחת קוד התחברות** – נקבל בית ספר, שם משתמש, ומספר טלפון. נכין את גוף הבקשה עם הפרמטרים שקיבלנו ונשלח בקשת post. נחזיר משתנה בוליאני – האם שליחת הקוד הצליחה או לא.

```
Future<File> getPicture(String userId, File file) {
  Map<String, String> headers = _authHeader();

  return _requestController
    .get(_pictureUrl(userId), headers)
    .then((response) {
      return response.bodyBytes;
    }).then((picture) {
      return file.writeAsBytes(picture);
    });
}
```

4. **קבלת תמונה** – הפעולה תקבל את userId וFile שעליו יהיה אפשר לכתוב את הביטים של התמונה שנקבל. נקרא לבקשת get עם headers הבסיסים. לאחר קבלת התשובה נכתוב את הביטים מהתשובה על הביטים של המסמך שקיבלנו.

5. **קבלת מסמך מהודעה** – לכל הודעה יש רשימה של מסמכים שהיא מכילה, לפעמים 0 ולפעמים 10. כשאנחנו רוצים לקבל את המסמך אנחנו בעצם צריכים להוריד אותו לטלפון שלנו ולפתוח אותו. נקבל את הid של ההודעה, הid של המסמך ושמו. נוריד את המסמך לטלפון תוך כדי שימוש בספריות (כולן מפורטות בהמשך).

6. **התנתקות** – פעולת ההתנתקות תהיה פעולה מאוד פשוטה. נשלח בקשת get עם הקישור של ההתנתקות והheaders עם נתוני cookie. בנוסף לכך לאחר ההתנתקות מהשרת, נקרא לפעולה clearAll בתוך requestController כדי לנקות כל נתוני cookie השמורים בתוך המחלקה.

6. מחלקה כוללת המאחדת את כל ה-API – mashov_api.dart

כדי לגשת למחלה הראשית (ApiController) יש לקבל כל מיני פרמטרים ואזכורים של מחלקות אחרות. לכן המחלקה MashovAPI תקל על התהליך ועם פקודה סטטית אחת נוכל לקבל את הcontroller שלנו ומשם לקבל ציונים ועוד.

בנוסף לכך היא תכריז על ה-API כספרייה ותייצא את כל המחלקות הקשורות ל-API.

```
library mashov_api;

import 'src/controller/api_controller.dart';
import 'src/controller/cookie_manager/cookie_manager_impl.dart';
import 'src/controller/request_controller/request_controller_impl.dart';

export 'src/controller/api_controller.dart';
export 'src/controller/cookie_manager/cookie_manager.dart';
export 'src/controller/cookie_manager/cookie_manager_impl.dart';
export 'src/controller/request_controller/request_controller.dart';
export 'src/controller/request_controller/request_controller_impl.dart';
export 'src/models/models.dart';

class MashovApi {
  static ApiController? _controller;

  static ApiController? getController() {
    _controller ??= ApiController(CookieManagerImpl(), RequestControllerImpl());
    return _controller;
  }
}
```

תמונה 25- צילום מסך של הקובץ mashov_api.dart מתוך התוכנה אנדרואיד סטודיו

שלב רביעי – הכנת האפליקציה

הורדתי את התוכנה אנדרואיד סטודיו ופתחתי פרויקט flutter חדש.

הפרויקט יבנה בשלבים כמו בניין: נבנה קודם כל את הבסיס ואז נבנה עליו את כל שאר הקומות.

ראשית, הוספתי את כל הספריות שהאפליקציה תצטרך לקובץ בשם `pubspec.yaml`, באותו קובץ גם צריך לכתוב את הגרסה של האפליקציה ואת גרסת הבנייה של האפליקציה. כרגע הן יהיו גרסה – 1.0.0 וגרסת בנייה – 1 (גרסת הבנייה משמשת לגוגל פליי ולאפ סטור לעקוב אחר הגרסאות שמועלות לענן).

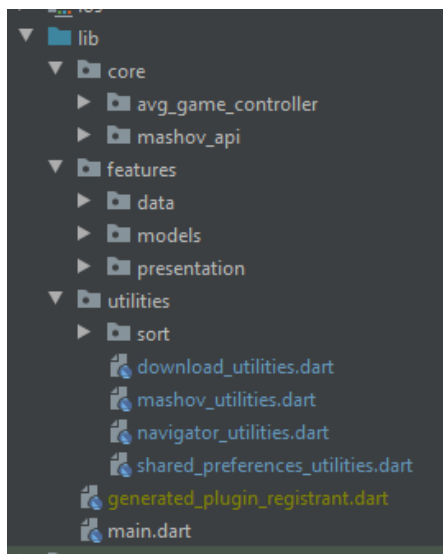
שנית, הגדרתי את כל הפרטים הבסיסיים של אנדרואיד, אייפון `web`. יצרתי אייקון לאפליקציה, וכתבתי ב`AndroidManifest.xml` וב`Info.plist` את ההרשאות כגון: להורדת קבצים ולגישה לאינטרנט.

יצרתי נתיבים חדשים באפליקציה – `assets` ו`fonts` הנתיבים יכילו את התמונות והפונטים שהשתמשתי בהם באפליקציה.

בנוסף לכך, הוספתי את השירותים של פיירבייס (`firebase`) – שירותי ענן העוזרים לך לגלות באגים וקריסות באפליקציה) כדי ליצור למשתמשים חווית שימוש טובה.

לאחר שעשיתי את השלבים הראשונים נוכל להתחיל בבניית האפליקציה עצמה.

חילקתי את האפליקציה לשלושה חלקים עיקריים.



- **core** – הליבה של האפליקציה: מורכבת מהAPI של משוב וניהול תחרות ממוצע הציונים. בעצם כל ההתעסקות של האפליקציה עם הענן.
- **features** – העיצוב והגרפיקה של האפליקציה (ווידג'טים⁹, מסכים ועוד) והמודלים הקשורים לכך.
- **utilities** – מחלקות עזר (לדוגמא מחלקה שתעזור לנו לשמור פרמטרים בזיכרון של הטלפון: פרטי התחברות, מצב כהה וכדומה).

תמונה 26- המבנה הבסיסי של

האפליקציה מתוך התוכנה אנדרואיד סטודיו

⁹ מחלקות שונות של עיצוב.

הקובץ main.dart הוא הקובץ העיקרי של האפליקציה, כאשר האפליקציה תרוץ היא תתחיל מקובץ זה. כתבתי בקובץ הזה את ההגדרות הבסיסיות של העיצוב של האפליקציה לדוגמא: כותרת האפליקציה, הגדרה שהאפליקציה תהיה רק מאוזנת ולא מאונכת, הצבע העיקרי של האפליקציה, כיוון הכתיבה ועוד. כעת אציג את השלבים העיקריים שהאפליקציה עוברת עם תחילת ההרצה שלה.

1. שלב ראשון – דף ראשוני, התחברות

המחלקה הראשונה של האפליקציה היא מחלקה הנקראת `LoadingPage`. מחלקה זו תציג את המסך הראשון של האפליקציה – מסך טעינה. הטעינה תעשה בשני שלבים:

- טעינה של כל הפרטים השמורים בזיכרון בעזרת מחלקת העזר `SharedPreferencesUtilities`. עד שזיכרון נטען יופיע רק הלוגו של גריינדר.
 - כעת יש שני אפשרויות
 - המשתמש כבר מחובר לאפליקציה, נבקש מה-API להתחבר עם הפרטים השמורים לנו בזיכרון ונעבור למסך `HomePage`.
 - המשתמש נכנס פעם ראשונה לאפליקציה. נבקש מה-API את רשימת בתי הספר ונעבור למסך `LoginPage`.
- עד שאחד מהשלבים הללו יטען (לוקח זמן עד שאנחנו מקבלים את התשובה ממשוב), יופיע הלוגו של גריינדר עם ווידג'ט טעינה מתחתיו.

אם המשתמש הועבר למסך ההתחברות, הוא יצטרך לכתוב את כל הפרטים שלו – בית ספר, שנת לימוד, שם משתמש וסיסמא וללחוץ על כפתור הכניסה, אם כל פרטיו נכונים הוא יועבר ל-`HomePage`, אם לא, יחוייב להכניס את פרטיו שוב.

2. שלב שני – מסכים באפליקציה

יצרתי מחלקה אבסטרקטית הנקראת `BaseScreen`. מחלקה זו תגדיר כיצד יש לממש כל מסך באפליקציה (למעט תחרות ממוצע שהציונים שהייתי צריך לממש בצורה יותר כיפית ומשחקית), כל מסך חדש באפליקציה, יירש את `BaseScreen`.
פרטי ה-`BaseScreen`:

- בכניסה אל המסך תקרא הפונקציה `getMashovData`, פעולה אבסטרקטית. הפעולה תהיה פעולת `Future` וכל מחלקת מסך תצטרך לממשה בעצמה – פעם אחת קבלת הציונים, פעם אחרת קבלת מערכת שעות ועוד.
- כאשר הפעולה הנ"ל בטעינה, יופיע מסך טעינה היכול את ווידג'ט הטעינה.
- המחלקה תהיה עטופה בווידג'ט `Theme` ובעזרתו נוכל להעביר את האפליקציה למצב כהה במהירות.

- הווידיג'ט של המחלקה מכיל header – ראש האפליקציה, body – גוף האפליקציה (תמיד יהיה ניתן לגלילה), bottom – תחתית האפליקציה (לא השתמשתי בה למעט מסך ההודעות).
- למסכים יהיה שלוש נקודות מצד השמאלי ומשמ יתה אפשר לרענן, או כל פעולה אחרת שהמסך יממש בעצמו לדוגמה: במסך הציונים, יהיה אופציה לחלק את הציונים למקצועות או לחשב ממוצע בלי מקצוע מסוים.

מסך הבית

במסך הבית יופיעו לנו פרטים ראשוניים של המשתמש – הפרטים שהתלמיד הכי מתעניין בהם. נחלק את מסך הבית לשני חלקים, הגוף והראש.

• הראש:

- באמצע בגדול יהיה את ממוצע התלמיד.
- מצדדי הממוצע יהיו מספר השעות שיש לו היום ומספר ההודעות שלא נקראו.
- בראש העמוד יהיו מגירת האפשרויות, שם המסך (כרגע: "מסך הבית") ושלוש נקודות המובילות אותך לאפשרויות מתקדמות כגון: רענון, שינוי סידור, שינוי ערכת צבע והסרת הערות.
- הראש יהיו מסוג sliver app bar – כאשר אנו גוללים למטה ראש האפליקציה נעלם.

• הגוף:

- הגוף מורכב מכרטיסיות. יהיו ארבעה כרטיסיות: ציונים אחרונים, שיעורי בית, מערכת שעות יומית ומיקום בתחרות ממוצע ציונים.

הערה: אני לא אפרט על כל מסך ומסך כיצד הכנתי אותו משום שישנם עשרות מסכים. רוב המסכים בנויים דומה מאוד משום שכולם יורשים מהמחלקה BaseScreen.

3. שלב שלישי – מגירת האפשרויות

כדי לנווט באפליקציה הכנתי את מגירת האפשרויות. מבנה מגירת האפשרויות:

- בראש המגירה יהיה תמונה של הלוגו של גרייזר על מסך תכלת – מעשיר את העיצוב של האפליקציה.
- התמונה של המשתמש ממושב (יש אפשרות לשנות אותה בלחיצה על התמונה).
- רשימה של כל האפשרויות באפליקציה: עמוד הבית, ציונים, מערכת שעות, לוח תוצאות (תחרות ממוצעים), שיעורי בית, הודעות, אירועי התנהגות, מונה אירועים, התאמות ושני פעולות: התנקות ויציאה (ללא התנתקות). לכל אפשרות יש אייקון כדי שאנשים יוכלו להבין גם בלי לקרוא את האפשרות.
- לחיצה על אחת מאפשרויות תוביל אותך למסך שלה. לחיצה על חזור מכל מקום, תחזיר אותך אל מסך הבית.
- ישנם אופציות גם להחלפת שנה (אם המשתמש היה נמצא במשוב גם בשנים שעברו, הוא יוכל להתחבר לשנה זו במידיות), יציאה ללא התנתקות והתנקות.

4. שלב רביעי – תחרות ממוצע ציונים

התחרות מאפשרת לכל התלמידים הנמצאים בגריידר להתחבר עם שמם, בית ספרם וציונם ולהתחרות על הממוצע הטוב ביותר.

כפותחים דף זה לראשונה, האפליקציה מסבירה לתלמיד כיצד עובדת התחרות ומבקשת ממנו לאשר שאכן הוא רוצה לשתף את הממוצע שלו עם שאר האנשים. לאחר ההסכמה, נשלח בקשה לשרת שיצרתי (לא הרחבתי עליו משום שזה לא עיקר העבודה) שירשום תלמיד חדש לתחרות.

לאחר ההרשמה, המשתמש מגיע למסך בו הוא יכול לראות את כל המשתמשים המחוברים לתחרות ולאיזה מקום הוא הגיע. יש מספר תחרויות:

- **כללי** – כל התלמידים בכל השכבות בתחרות אחת
- **בתי ספר** – תחרות של בתי הספר למי יש את הממוצע, בין הממוצעים של התלמידים מאותו בית ספר, ההכי גבוה.
- **כל שכבות הגיל** – לכל שכבת גיל יש תחרות משלה, כל תלמיד יכול לראות תחרויות של שכבות אחרות אך להשתתף הוא יכול רק בשכבה שלו.

ניתן להשתתף בכמה תחרויות בו זמנית – גם עם הממוצע שלך בשכבות גיל אחרות (אם יש לך).

שמרתי את המשתמשים עם מזהה `userId` שמשוב נותן להם, `userId`. מזהה זה לא משתנה עם השנים והוא נשאר שלך לנצח לכן המשתמש נשמר כאותו משתמש גם עם עברה שנה.

5. שלב חמישי – התנתקות

על ידי לחיצה על כפתור ההתנתקות במגירת האפשרויות יש אפשרות להתנתק מאפליקציה.

גריידר ימחוק מהזיכרון את כל הפרטים שלך, ויוביל אותך לעמוד ההתחברות – העמוד ההתחלתי.

ספריות בהם השתמשתי ביצירת האפליקציה

עיצוב

- [Like button](#) – כפתור עם אנימציה לתחרות ממוצע הציונים.
- [Wave](#) – עיצוב הכותרת לתחרות בצורה יפה ומעניינת.
- [Cupertino icons](#) – אייקונים.
- [Page transition](#) – לעבור בין מסך למסך עם אנימציה.
- [Flash](#) – עיצוב של snack bar לתקלה בהתחברות.
- [Charts flutter](#) – הצגת גרפים של הציונים של המשתמש מקצוע מסוים.
- [Simple animations](#) – אנימציות לאורך האפליקציה.
- [Auto size text](#) – שינוי גודל הטקסט בהתאם לגודל המסך.
- [Fab circular menu](#) – כפתור בצורת עיגול הנפתח עם אנימציה.

הצגת הנתונים והתוכן של האפליקציה

- [Wakelock](#) – שמירת המסך של הטלפון דלוק.
- [Connectivity](#) – נותן לדעת האם למשתמש יש אינטרנט פעיל.
- [url_launcher](#) – פתיחת קישורים מן האפליקציה לאתר באינטרנט.
- [Flutter html](#) – הצגת קוד html בתוך האפליקציה.
- [Image picker](#) – שולח את המשתמש לבחירת תמונה מהגלריה שלו.

הורדה ושליחת הודעות

- [downloads_path_provider 28](#) – גישה לשמירה לדרך מסוימת בזיכרון הטלפון.
- [dio](#) – לקיחת מסמך מהאינטרנט ושמירה בקובץ.
- [path_provider](#) – קבלת הנתביב לזיכרון של הטלפון.
- [permission_handler](#) – ניהול ההרשאות שיש לאפליקציה.
- [flutter_local_notifications](#) – שליחת הודעות מהאפליקציה.
- [open_file](#) – לאחר לחיצה על ההודעה, הספרייה מאפשרת לפתוח את הקובץ שהורד בטלפון.

פירסור הנתונים

- [Dart:convert](#) – מאפשרת להשתמש בפקודות של json (מעביר משתנה מסוג מחרוזת למילון ולהפך. השימוש בjson פועל באופן גלובלי בכל השפות).

שליחת הנתונים

- [http](#) – יצירת http client כדי לאפשר ליצור session אחד שממנו נשלחים כל הבקשות get ו post.
- [dart:async](#) – מאפשר לפונקציה להיות async.

הצעת נושא לעבודת גמר במדעי המחשב

מבוא

הבעיה המרכזית איתה אני מתמודד בעבודה היא הנדוס לאחר והבנה עמוקה של אופן הפעולה של אפליקציה מוכרת ב-google play הנקראת "משוב". אעשה זאת על ידי מציאה וניתוח שיטות להנדסה לאחר, התעמקות בפרוטוקולי תקשורת כגון https, הבנת דרכי הפעולה של אפליקציות צד-שרת וצד-לקוח ועוד. ברצוני לפתח אפליקציה שמתנהגת כמו "משוב" אך גם מציעה עיצוב נקי וחדשני. החלטתי לקחת על עצמי פרויקט, לחקור את אפליקציית משוב מכל ההיבטים השונים – לנתח ולהבין את הקוד של "משוב", את התקשורת שלו עם שרתיו ועוד. לאחר שאני אסיים לכתוב את עבודת הגמר אני חושב שהרבה תלמידים ירצו להשתמש באפליקציה שאכין וגם יוכלו להבין בקלות יותר את עקרונותיהם של הנדסה לאחר ותעבורת https. כדי להציג פתרון לבעיה שלי, עלי ללמוד על שיטות להנדסה לאחר, לבצע הנדסה לאחר בעצמי, לנתח את הקוד (על ידי למידת השפה smali) ולהבין את דרכי פעולה של "משוב" ואפליקציות צד-שרת וצד לקוח. לבסוף, אני אצטרך ליצור אפליקציה (גם לאנדרואיד, לאיפון ול-web) על ידי השפה dart.

מטרות מרכזיות של העבודה:

- הבנת אופן הפעולה של הנדסה לאחר של קבצים מסוג apk.
 - פירוט שיטות של reverse engineering.
 - הבנת כל אחת ואחת מן השיטות לעומק.
 - שילוב השיטות על אפליקציית "משוב".
- הבנת אופן הפעולה של אפליקציות צד-רשת.
- שימוש והבנה של העקרונות של https והנדסה לאחר.
- הבנת עמוקה של הפרוטוקולים HTTP ו-TCP.
- מחקר והבנה על אופן פעולות רשתות שונות באינטרנט.
- רכישת ידע תיאורטי לגבי שפת התכנות dart והיכרות עם ארכיטקטורת אנדרואיד.
- רכישת ידע מעשי ולמידת שפת התכנות dart.
- מימוש והגעה למוצר סופי עובד - אפליקציה עובדת שמתנהגת כמו משוב, טובה יותר.

תיאור תהליך העבודה:

על מנת להכין את עבודת החקר שלי, ראשית אני צריך להבין את אופן הפעולה של הנדסה לאחר של קבצי apk ולהעמיק בנושא. אוכל ללמוד זאת בעזרת הספר "Reversing: Secrets of Reverse Engineering", ועוד אתרים ברשת. אצטרך לבצע אחת מהשיטות שלמדתי על אפליקציית "משוב". שנית, אני צריך ללמוד בצורה כללית על אופן הפעולה של הרשתות. זאת אוכל ללמוד מהספר "רשתות מחשבים" מאתר הסייבר הישראלי, ומהספר "Network basics CCNA". לאחר למידה כללית על הנושא, אעבור להתמקד בפרוטוקולים העיקריים HTTP ו-

TCP. עליי ללמוד מודלים עיקריים שיש ברשת (מכיוון שהפרוטוקול משלב את שניהם). המודלים העיקריים הם : Client ו-Server. את הבסיס התיאורטי למודלים אלה אוכל ללמוד מספרים ומקורות שונים ברשת. בהמשך, אצטרך ללמוד על תכנות אפליקציות בשפת התכנות של גוגל – dart, זאת אוכל ללמוד מהספר "Beginning Flutter: A Hands On Guide to App Development", "Flutter in Action" ועוד המון מקורות מידע ואתרים נוספים מן האינטרנט. לבסוף, אוכל לשלב את כל מה שלמדתי ליצירת אפליקציה המתנהגת כמו משוב, ואף יותר טובה. במהלך כל עבודתי אנסה להרחיב את הידע שלי בתחומים רבים מהתחום של מדעי המחשב מעבר למה שרשמתי. כל המקורות שצינתי יופיעו בביבליוגרפיה עם קישור אליהם במידת הצורך, וכמובן שישנם עוד ספרים נוספים רבים שמהם אוכל לשאוב מידע, ומספר מקורות עצום עם מידע חיוני ברחבי האינטרנט.

פירוט ההיבט המחקרי של העבודה

תהליך העבודה המחקרי שלי יעשה בשלבים : בתחילה, אני אמצא שיטות להנדסה לאחר של קבצי apk, אלמד אותם ואנסה ליישם אחת מהדרכים על אפליקציית "משוב". לאחר מכן, אלמד על תעבורת https לעומק ואנסה להבין את התעבורה של משוב על ידי הסנפת האתר של משוב (<https://web.mashov.info/students/login>). לאחר מכן, אלמד את שפת התכנות dart בעזרת ספרים ואתרים באינטרנט ואצור כמה קטעי קוד נפרדים : אחד להתחברות למשוב, שמירת cookies, לקיחת הציונים ועוד. לבסוף, אני אשלב את כל קטעי הקוד השונים ואצור את האפליקציה הסופית המתנהגת כמו משוב על סמך כל המחקר שעשיתי.

לוח זמנים :

- ינואר-פברואר 2021 – ניסוח שאלת המחקר והצגת הבסיס התיאורטי.
- מרץ-אפריל – מחקר והבנת הבסיס התיאורטי הדרוש לעבודה : מפורט למעלה.
- מאי-יוני – מידול והכנת פרויקטים קטנים כחלק מהפרויקט הגדול לשם הבנה.
- יולי – מימוש החלק הראשון של הפרוטוקול (ניתוח הקוד של משוב בעזרת הנדסה לאחר)
- אוגוסט – מימוש חלק שני (חקירת תעבורת https של משוב ויצירת)
- ספטמבר-אוקטובר – מימוש מלא של הפרוטוקול, חיבור חלקי הקוד ויצירת אפליקציה.
- נובמבר – בדיקה, שיפור הקוד.
- דצמבר 2021 – סיום כתיבת עבודת החקר וליטוש סופי.

1. Odom Wendell, and Tom Knott (2006). Networking basics: CCNA 1 companion guide. Indianapolis, IN: Cisco Press, March 22, print.
2. Eric windmill (2019). "Flutter in Action".
3. Marco L. Napoli (2019). "Beginning Flutter: A Hands-on Guide to App Development".
4. Eldad Eilam (2011). Reversing: Secrets of Reverse Engineering.
5. עומר רוזנבוים, שלומי הוד וברק גונן (2020). "רשתות ומחשבים".
6. ברק גונן (2021). "תכנות בשפת פייתון".
7. דוד א (2014). "Reverse Engineering – לגלות את הסודות החבויים". גיליון 52.
8. דר' עמית רש (2019). "הנדסה לאחור / הנדסה הפוכה - Reverse Engineerin".

mashov_api.dart

```
library mashov_api;
```

```
import 'src/controller/api_controller.dart';
```

```
import 'src/controller/cookie_manager/cookie_manager_impl.dart';
```

```
import 'src/controller/request_controller/request_controller_impl.dart';
```

```
export 'src/controller/api_controller.dart';
```

```
export 'src/controller/cookie_manager/cookie_manager.dart';
```

```
export 'src/controller/cookie_manager/cookie_manager_impl.dart';
```

```
export 'src/controller/request_controller/request_controller.dart';
```

```
export 'src/controller/request_controller/request_controller_impl.dart';
```

```
export 'src/models/models.dart';
```

```
class MashovApi {
```

```
  static ApiController? _controller;
```

```
  static ApiController? getController() {
```

```
    _controller ??= ApiController(CookieManagerImpl(), RequestControllerImpl());
```

```
    return _controller;
```

```
  }
```

```
}
```

utils.dart

```
import 'models/models.dart';
```

```
class Utils {
```

```
  /// Returns an empty string if value is null, the value itself otherwise.
```

```
  static String string(String? value) => value ?? "";
```

```
  /// Returns 0 if value is null, the value itself otherwise.
```

```
  static int integer(int? value) => value ?? 0;
```

```
  /// Returns false if value is null, the value itself otherwise.
```

```
  static bool boolean(bool? value) => value ?? false;
```

```
  /// Parses a list of attachments.
```

```
  static List<Attachment> attachments(List? src) =>
```

```
    src == null
```

```
      ? []
```

```
      : src.map<Attachment>((s) => Attachment.fromJson(s)).toList();
```

```
  /// Takes a list and turns it into the format:
```

```
  /// "[ a1, a2, a3, a4, ... ]"
```

```
  /// Used to convert a list to a nicely formatted json.
```

```
  static String listToString(List<dynamic> strings) {
```

```
    if (strings.isEmpty) return "[ ]";
```

```
    String string = '[$ {strings[0]}';
```

```

if (strings.length == 1) {

    string += ']';

} else {

    for (int i = 1; i < strings.length; i++) {

        string += ', ${strings[i]}';

    }

    string += ']';

}

return string;

}

/// Turns headers into something a bit more readable.

static Map<String, List<String>> decodeHeaders(Map<String, String> headers) =>

    headers.map((key, value) => MapEntry(key, value.split(';')));

/// merges two maps, with an advantage to the first map(in case of same key and value,

/// the value of the first map will be taken.

static Map<dynamic, dynamic> mergeMaps(Map<dynamic, dynamic> map1,

    Map<dynamic, dynamic> map2) {

    Map merged = Map.from(map1);

    for (String key in map2.keys) {

        merged.putIfAbsent(key, () => map2[key]);

    }

    return merged;

}

}

```

cookie_manager.dart

```
abstract class CookieManager {
```

```
    late String csrfToken, mashovAuthToken, uniqueId;
```

```
    ///Saves the csrf-token, session id and uniqueId.
```

```
    void processHeaders(Map<String, List<String>> headers);
```

```
    ///Attach a listener to cookie manager - whenever one of the variables is updated, these listeners are triggered
```

```
    void attachListener(int id, Function listener);
```

```
    ///Detach a listener from the cookie manager.
```

```
    void detachListener(int id);
```

```
    ///Clear all data saved
```

```
    void clearAll();
```

```
}
```

cookie_manager_impl.dart

```
import 'cookie_manager.dart';
```

```
class CookieManagerImpl implements CookieManager {
```

```
  @override
```

```
  void clearAll() {
```

```
    _csrfToken = "";
```

```
    _mashovSessionId = "";
```

```
    _uniqueId = "";
```

```
    _listeners = {};
```

```
  }
```

```
  String _csrfToken = "";
```

```
  @override
```

```
  String get csrfToken => _csrfToken;
```

```
  @override
```

```
  set csrfToken(String csrfToken) {
```

```
    _csrfToken = csrfToken;
```

```
    trigger();
```

```
  }
```

```
  String _mashovSessionId = "";
```

@override

String get mashovAuthToken => _mashovSessionId;

@override

set mashovAuthToken(String mashovSessionId) {

 _mashovSessionId = mashovSessionId;

 trigger();

}

String _uniqueId = "";

@override

String get uniqueId => _uniqueId;

@override

set uniqueId(String uniqueId) {

 _uniqueId = uniqueId;

 trigger();

}

Map<int, Function> _listeners = {};

@override

void attachListener(int id, Function listener) {

```

    _listeners[id] = listener;
}

```

```

@Override

void detachListener(int id) {

    _listeners.remove(id);
}

```

```

@Override

void processHeaders(Map<String, List<String>> headers) {

    if (csrfToken.isEmpty() &&
        mashovAuthToken.isEmpty() &&
        uniqueId.isEmpty()) {

        return;
    }

    var cToken = headers["x-csrf-token"];

    if (cToken != null) {

        csrfToken = cToken[0];
    }

    if (headers.containsKey("set-cookie")) {

        var cookie = headers["set-cookie"];

        uniqueId = cookie

            .firstWhere((header) => header.contains("uniqueId"))

            .split("=")

```



```

        .last;

    mashovAuthToken = cookie

        .firstWhere((header) => header.contains("MashovAuthToken"))

        .split("=")

        .last;

    }

}

void trigger() => _listeners.values.forEach((f) => f());

}

```

request_controller.dart

```
import 'dart:async';

import 'package:http/http.dart';

import 'package:http/http.dart' as http;

///I actually did think this class was gonna be more useful than that.

///But uhm, maybe it will have some meaning in the future??

///Who knows. I'm keeping it.

abstract class RequestController {

  ///create a client

  late http.Client client;

  ///sends a get request.

  Future<http.Response> get(String url, Map<String, String> headers);

  ///sends a post request.

  Future<http.Response> post(String url, Map<String, String> headers, Object body);

}
```

request_controller_impl.dart

```
import 'dart:async';

import 'package:http/http.dart' as http;

import 'request_controller.dart';

class RequestControllerImpl implements RequestController {

  RequestControllerImpl();

  @override

  Future<http.Response> get(String url, Map<String, String> headers) async {

    return client.get(Uri.parse(url), headers: headers);

  }

  @override

  Future<http.Response> post(String url, Map<String, String> headers,

    Object body) async {

    return client.post(Uri.parse(url), headers: headers, body: body);

  }

  @override

  http.Client client = http.Client();

}
```

api_controller.dart

```
import 'dart:async';

import 'dart:convert';

import 'dart:io';

import
'package:grader_for_mashov_new/features/presentation/widgets/custom_dialog/custom_dialog.dart';

import 'package:grader_for_mashov_new/utilities/download_utilities.dart';

import 'package:http/http.dart' as http;

import 'cookie_manager/cookie_manager.dart';

import 'request_controller/request_controller.dart';

import '../models/models.dart';

import '../utils.dart';

typedef Parser<E> = E Function(Map<String, dynamic> src);

typedef DataProcessor = void Function(dynamic data, Api api);

typedef RawDataProcessor = void Function(String json, Api api);

class ApiController {

  final CookieManager _cookieManager;

  final RequestController _requestController;

  DataProcessor? _dataProcessor;

  RawDataProcessor? _rawDataProcessor;

  detachDataProcessor() {

    _dataProcessor = null;
```

```
}
```

```
attachDataProcessor(DataProcessor? processor) {
```

```
    if (processor != null) {
```

```
        _dataProcessor = processor;
```

```
    }
```

```
}
```

```
detachRawDataProcessor() {
```

```
    _rawDataProcessor = null;
```

```
}
```

```
attachRawDataProcessor(RawDataProcessor? processor) {
```

```
    if (processor != null) {
```

```
        _rawDataProcessor = processor;
```

```
    }
```

```
}
```

```
ApiController(this._cookieManager, this._requestController)
```

```
    : _rawDataProcessor = null,
```

```
      _dataProcessor = null,
```

```
      jsonHeader = _setJsonHeader();
```

```
///returns a list of schools.
```

```
Future<Result<List<School>>> getSchools() =>
```

```
_authList(_schoolsUrl, School.fromJson, Api.schools);
```

```
///Send one time password
```

```
Future<bool> sendLoginCode(School school, String id, String cellphone) async {
```

```
  var body = {
```

```
    "cellphone": cellphone,
```

```
    "semel": school.id,
```

```
    "username": id,
```

```
  };
```

```
  Map<String, String> headers = {"Content-Type": "application/json"};
```

```
  var res = await _requestController.post(
```

```
    _cellphoneUrl, headers, json.encode(body));
```

```
  return res.statusCode == 200;
```

```
}
```

```
///logs in to the Mashov API.
```

```
Future<Login?> login(School school, String id, String password, int year,
```

```
  {String? uniqueId}) async {
```

```
  var body = {
```

```
    "semel": school.id,
```

```
    "year": year,
```

```
    "username": id,
```

```
    "password": password,
```

```
    "appName": "info.mashov.students",
```

```
    "appVersion": _apiVersion,
```

```

"apiVersion": _apiVersion,
"appBuild": _apiVersion,
"deviceUuid": "chrome",
"devicePlatform": "chrome",
"deviceManufacturer": "win",
"deviceModel": "desktop",
"deviceVersion": "85.0.4183.121"
};

```

Login? login;

```
var headers = jsonHeader;
```

```
headers.addAll(_loginHeader(uniqueId: uniqueId));
```

```
await _requestController
```

```
    .post(_loginUrl, headers, json.encode(body))
```

```
    .then((response) {
```

```
        if (response.statusCode == 200) {
```

```
            String uniqueId = response.headers["set-cookie"]!
```

```
                .split("uniqueId=")
```

```
                .last
```

```
                .split(";")[0];
```

```
            processResponse(response);
```

```
            login = Login.fromJson(
```

```
                json.decode(utf8.decode(response.bodyBytes)), uniqueId);
```

```
        }
```

```

});

return login;
}

```

```

Future<List<dynamic>> getBells() async {

  Map<String, String> headers = _authHeader();

  List bellsMap =

    await _requestController.get(_bellsUrl, headers).then((response) {

      return json.decode(response.body);

    });

  return bellsMap;
}

```

///Returns a list of grades.

```

Future<Result<List<Grade>>> getGrades(String userId) => _process(

  _authList<Grade>(_gradesUrl(userId), Grade.fromJson, Api.grades),

  Api.grades);

```

///Returns a list of behave events.

```

Future<Result<List<BehaveEvent>>>? getBehaveEvents(String userId) => _process(

  _authList<BehaveEvent>(

    _behaveUrl(userId), BehaveEvent.fromJson, Api.behaveEvents),

  Api.behaveEvents);

```

/// Returns the messages count - all, inbox, new and unread.


```

Future<Result<MessagesCount>> getMessagesCount() => _process(
    _auth(_messagesCountUrl, MessagesCount.fromJson, Api.messagesCount),
    Api.messagesCount)
.then((r) => Result(
    exception: r.exception,
    statusCode: r.statusCode,
    value: r.value == null ? null : r.value as MessagesCount));

```

///Returns a list of conversations.

```

Future<Result<List<Conversation>>> getMessages(int skip) => _process(
    _authList(_messagesUrl(skip), Conversation.fromJson, Api.messages),
    Api.messages);

```

///Returns a specific message.

```

Future<Result<Message>> getMessage(String messageId) => _process(
    _auth(_messageUrl(messageId), Message.fromJson, Api.message),
    Api.message)
.then((r) => Result(
    exception: r.exception,
    statusCode: r.statusCode,
    value: r.value == null ? null : r.value as Message));

```

///Returns the user timetable.

```

Future<Result<List<List<Lesson>>>> getTimeTable(String userId) async {
//  try {

```

```
Map<String, String> headers = _authHeader();
```

```
List lessonsMap = await _requestController
```

```
    .get(_timetableUrl(userId), headers)
```

```
    .then((response) {
```

```
        List map;
```

```
        try {
```

```
            map = json.decode(response.body);
```

```
        } catch (e) {
```

```
            return [];
```

```
        }
```

```
        return map;
```

```
    });
```

```
List bellsMap =
```

```
    await _requestController.get(_bellsUrl, headers).then((response) {
```

```
        return json.decode(response.body);
```

```
    });
```

```
List maps = [];
```

```
for (var l in lessonsMap) {
```

```
    int num = l["timeTable"]["lesson"];
```

```
    Map? matchingBell = bellsMap.firstWhere(
```

```
        (m) => Utils.integer(m["lessonNumber"]) == num,
```

```
        orElse: () => null);
```

```
    if (matchingBell == null) {
```

```

        debugPrint(
            "we've got a serious error here: no bell matched for lesson number $num, bells length is
            ${bellsMap.length}");
    }

    maps.add(matchingBell == null
        ? 1
        : Utils.mergeMaps(1, {
            "startTime": matchingBell["startTime"],
            "endTime": matchingBell["endTime"]
        }));
}

List<Lesson> timetable = maps.map((m) => Lesson.fromJson(m)).toList();

if (_dataProcessor != null) _dataProcessor!(timetable, Api.timetable);

return Result<List<List<Lesson>>>>(
    exception: null,
    value: processTableTimeDate(timetable, bellsMap),
    statusCode: 200);
}

```

```

List<List<Lesson>> processTableTimeDate(
    List<Lesson> times, List<dynamic> bells) {
    List<List<Lesson>> newTableTime = [];

    List<Lesson> oneDay = [];

```

```

for (int i = 0; i < 6; i++) {

    for (int j = 0; j < times.length; j++) {

        if (times[j].day == i + 1) {

            oneDay.add(times[j]);

        }

    }

    oneDay.sort((a, b) => a.lesson.compareTo(b.lesson));

    newTableTime.insert(0, oneDay);

    oneDay = [];

}

```

```

for (int i = 0; i < newTableTime.length; i++) {

    List<Lesson> okTest = newTableTime[i];

    List<int> jo = [];

    if (okTest.isEmpty()) {

        for (int k = 1; k < (okTest.last.lesson) + 1; k++) {

            bool contains = false;

            for (int o = 0; o < okTest.length; o++) {

                if (okTest[o].lesson == k) {

                    contains = true;

                }

            }

            if (!contains) {

                jo.add(k);

            }

        }

    }

}

```

```

    }
}

for (int y = 0; y < jo.length; y++) {

    okTest.insert(

        jo[y],

        Lesson(

            subject: "שיעור חופשי!",

            lesson: jo[y],

            endTime: bells[jo[y] - 1]["endTime"],

            startTime: bells[jo[y] - 1]["startTime"],

            teachers: [""],

            room: "",

            groupId: 1,

            day: okTest.last.day));

    }

    okTest.sort((a, b) => a.lesson.compareTo(b.lesson));

}

}

newTableTime.add([]);

return newTableTime;

}

//Returns a list of the Alfon Groups.

//The class group is a different address, so we use an id -1 to access it.

```

```

Future<Result<List<Group>>> getGroups(String userId) async {

  Result<List<Group>> groups =

    await _authList<Group>(_groupsUrl(userId), Group.fromJson, Api.groups);

  groups.value!.add(Group(id: -1, teachers: [], subject: "כיתת אם"));

  if (_dataProcessor != null) {

    _dataProcessor!(groups.value, Api.groups);

  }

  return groups;

}

```

///returns an Alfon Group contacts.

```

Future<Result<List<Contact>>> getContacts(String userId, String groupId) =>

  _process(

    _authList(_alfonUrl(userId, groupId), Contact.fromJson, Api.alfon),

    Api.alfon);

```

///Returns a list of Maakav reports.

```

Future<Result<List<Maakav>>> getMaakav(String userId) => _process(

  _authList(_maakavUrl(userId), Maakav.fromJson, Api.maakav), Api.maakav);

```

///Returns a list of homework.

```

Future<Result<List<Homework>>> getHomework(String userId) => _process(

  _authList(_homeworkUrl(userId), Homework.fromJson, Api.homework),

  Api.homework);

```

///Returns a list of bagrut exams, combining both dates, times, room and grades.

///This will NOT go through raw data processor as it takes the data

///from two sources.

```
Future<Result<List<Bagrut>>> getBagrut(String userId) async {

  try {

    Map<String, String> headers = _authHeader();

    List gradesMaps = await _requestController

      .get(_bagrutGradesUrl(userId), headers)

      .then((response) => json.decode(response.body));

    List timesMaps = await _requestController

      .get(_bagrutTimeUrl(userId), headers)

      .then((response) => json.decode(response.body));

    List<Map> maps = [];

    for (var g in gradesMaps) {

      int semel = Utils.integer(g["semel"]);

      Map? matchingTime = timesMaps.firstWhere(

        (m) => Utils.integer(m["semel"]) == semel,

        orElse: () => null);

      maps.add(matchingTime == null ? g : Utils.mergeMaps(g, matchingTime));

    }

    List<Bagrut> bagrut = maps.map((m) => Bagrut.fromJson(m)).toList();

    if (_dataProcessor != null) _dataProcessor!(bagrut, Api.bagrut);

    return Result<List<Bagrut>>>(

      exception: null, value: bagrut, statusCode: 200);

  }
```

```

    } catch (e) {

        return Result(exception: e, value: null, statusCode: 200);

    }

}

//Returns the user's hatamot.

Future<Result<List<Hatama>>> getHatamot(String userId) => _process(

    _authList(_hatamotUrl(userId), Hatama.fromJson, Api.hatamot),

    Api.hatamot);

///Returns the user's Hatamot bagrut.

Future<Result<List<HatamatBagrut>>> getHatamotBagrut(String userId) =>

    _process(

        _authList(_hatamotBagrutUrl(userId), HatamatBagrut.fromJson,

            Api.hatamotBagrut),

            Api.hatamotBagrut);

///Return the lesson count

Future<List<BehaveCounter>> getEventsCounter(String userId) async {

    Result<List<BehaveEvent>> behaves2 = await _process(

        _authList<BehaveEvent>(

            _behaveUrl(userId), BehaveEvent.fromJson, Api.behaveEvents),

            Api.behaveEvents);

    Result<List<Group>> groups2 =

```



```

    await _authList<Group>(_groupsUrl(userId), Group.fromJson, Api.groups);

Result<List<LessonCount>?> lessonCounts = await _authList(
    _lessonCountUrl(userId), LessonCount.fromJson, Api.lessonCount);

List<BehaveEvent>? behaves = behaves2.value;

List<Group>? groups = groups2.value;

List<BehaveCounter> behaveCounter = [];

if (groups == null) return [];

for (int i = 0; i < groups.length; i++) {
    String sub = groups[i].subject;

    LessonCount lessonCount;

    int lessonsCount;

    int weeklyHours;

    List<ShortBehaveEvent> events = [];

    try {
        lessonCount = lessonCounts.value!

        .where((element) => element.groupId == groups[i].id)

        .first;

        lessonsCount = lessonCount.lessonsCount;

        weeklyHours = lessonCount.weeklyHours;

    } catch (e) {

        weeklyHours = 0;
    }
}

```

```

    lessonsCount = 0;
}

var totalEvents =

    behaves!.where((element) => element.groupId == groups[i].id);

if (totalEvents.isNotEmpty) {
    for (int j = 0; j < totalEvents.length; j++) {
        BehaveEvent behaveEvent = totalEvents.elementAt(j);
        events.add(ShortBehaveEvent(
            subject: behaveEvent.subject,
            justification: behaveEvent.justification,
            justificationId: behaveEvent.justificationId,
        ));
    }
}

behaveCounter.add(BehaveCounter(
    subject: sub,
    lessonsCount: lessonsCount,
    weeklyHours: weeklyHours,
    events: events,
    totalEvents: totalEvents.length));
}

```

```

return behaveCounter;

}

```

///Returns the user profile picture into given file parameter.

```

Future<File> getPicture(String userId, File file) {

```

```

    Map<String, String> headers = _authHeader();

```

```

    return _requestController

```

```

        .get(_pictureUrl(userId), headers)

```

```

        .then((response) {

```

```

            return response.bodyBytes;

```

```

        }).then((picture) {

```

```

            return file.writeAsBytes(picture);

```

```

        });

```

```

    }

```

///log out from the session. might want to reset the cookies, but anyway

///The application should make sure these are reset.

```

Future<void> logout() async {

```

```

    Map<String, String> headers = {

```

```

        "Cookie":

```

```

        "uniqueId=${_cookieManager.uniqueId};MashovAuthToken=${_cookieManager.mashovAuthToken}";
    },

```

```

    };

```

```

headers.addAll(_authHeader());

await _requestController.get(_logoutUrl, _authHeader());

_cookieManager.clearAll();
}

```

```

Future<Map<String, dynamic>?> getAttachment(
    String messageId, String fileId, String name) async {
    Map<String, String> headers = _authHeader();
    headers.addAll(_authHeader());

    return await DownloadUtilities(
        link: _attachmentUrl(messageId, fileId, name),
        fileName: name,
        headers: headers)
        .startDownload();
}

```

///Returns a given maakav attachment.

```

Future<File> getMaakavAttachment(
    String maakavId, String userId, String fileId, String name, File file) {
    Map<String, String> headers = _authHeader();
    headers.addAll(_authHeader());

    return _requestController
        .get(_maakavAttachmentUrl(maakavId, userId, fileId, name), headers)
        .then((response) => response.bodyBytes)

```

```

        .then((attachment) => file.writeAsBytes(attachment));
    }

    ///Returns a list of E, using an authenticated request.
    Future<Result<List<E>>> _authList<E>(
        String url, Parser<E> parser, Api api) async {
        Map<String, String> headers = _authHeader();
        if (url != _schoolsUrl) {
            /// we don't need authentication when getting schools.
            headers.addAll(_authHeader());
        }
        return _requestController
            .get(url, headers)
            .then((response) => _parseListResponse(response, parser, api));
        //.then((body) => body.map<E>((e) => parser(e)).toList());
    }

    ///Returns E, using an authenticated request.
    Future<Result<E>> _auth<E>(String url, Parser parser, Api api) async {
        Map<String, String> headers = _authHeader();
        return _requestController
            .get(url, headers)
            .then((response) => _parseResponse(response, parser, api));
    }

```

```

Result<E> _parseResponse<E>(http.Response response, Parser parser, Api api) {
    try {
        Map<String, dynamic> src = json.decode(response.body);

        Result<E> result = Result(
            exception: null, value: parser(src), statusCode: response.statusCode);

        //if it had not crashed, we know the data is good.

        if (_rawDataProcessor != null) {
            _rawDataProcessor!(response.body, api);
        }

        return result;
    } catch (e) {
        return Result(exception: e, value: null, statusCode: response.statusCode);
    }
}

```

```

Result<List<E>> _parseListResponse<E>(
    http.Response response, Parser parser, Api api) {
    try {
        List src = json.decode(response.body);

        Result<List<E>> result = Result(
            exception: null,
            value: src.map<E>((e) => parser(e)).toList(),
            statusCode: response.statusCode);
    }
}

```

```

if (_rawDataProcessor != null) {

    _rawDataProcessor!(response.body, api);

}

return result;

} catch (e) {

    return Result(exception: e, value: null, statusCode: response.statusCode);

}

}

```

```

void processResponse(http.Response response) {

    _cookieManager.processHeaders(Utils.decodeHeaders(response.headers));

}

```

///The authentication header.

```

Map<String, String> _authHeader() {

    Map<String, String> headers = {}..addAll(jsonHeader);

```

///uniqueId is NOT a typo!

///I...I really don't know why they named it that way.

///just... go on

```

headers["cookie"] =

```

```

    "uniqueId=${_cookieManager.uniqueId};

```

```

MashovAuthToken=${_cookieManager.mashovAuthToken}; Csrf-
Token=${_cookieManager.csrfToken}";

```

```

headers["x-csrf-token"] = _cookieManager.csrfToken;

```

```

return headers;

}

Map<String, String> _loginHeader({String? uniqueId}) {

    Map<String, String> headers = { };

    headers["x-csrf-token"] = _cookieManager.csrfToken;

    headers["accept-encoding"] = "gzip";

    headers["Host"] = "web.mashov.info";

    if (uniqueId != null) {

        headers["Cookie"] = "uniqueId=${uniqueId}";

    }

    return headers;

}

```

```

Map<String, String> jsonHeader;

static const String _baseUrl = "https://web.mashov.info/api/";

static const String _schoolsUrl = _baseUrl + "schools";

static const String _loginUrl = _baseUrl + "login";

static const String _messagesCountUrl = _baseUrl + "mail/counts";

static const String _cellphoneUrl = _baseUrl + "user/otp";

static String _gradesUrl(String userId) =>

    _baseUrl + "students/$userId/grades";

static String _bagrutGradesUrl(String userId) =>

```



```
_baseUrl + "students/$userId/bagrut/grades";
```

```
static String _bagrutTimeUrl(String userId) =>
```

```
_baseUrl + "students/$userId/bagrut/sheelonim";
```

```
static String _hatamotUrl(String userId) =>
```

```
_baseUrl + "students/$userId/hatamot";
```

```
static String _hatamotBagrutUrl(String userId) =>
```

```
_baseUrl + "students/$userId/bagrut/hatamot";
```

```
static String _behaveUrl(String userId) =>
```

```
_baseUrl + "students/$userId/behave";
```

```
static String _messagesUrl(int skip) =>
```

```
_baseUrl + "mail/inbox/conversations?skip=$skip";
```

```
static String _messageUrl(String messageId) =>
```

```
_baseUrl + "mail/messages/$messageId";
```

```
static String _timetableUrl(String userId) =>
```

```
_baseUrl + "students/$userId/timetable";
```

```
static const String _bellsUrl = _baseUrl + "bells";
```

```

static String _groupsUrl(String userId) =>
    _baseUrl + "students/$userId/groups";

static String _maakavUrl(String userId) =>
    _baseUrl + "students/$userId/maakav";

static String _homeworkUrl(String userId) =>
    _baseUrl + "students/$userId/homework";

static String _pictureUrl(String userId) => _baseUrl + "user/$userId/picture";

static String _attachmentUrl(String messageId, String fileId, String name) =>
    _baseUrl + "mail/messages/$messageId/files/$fileId/download/$name";

static String _maakavAttachmentUrl(
    String maakavId, String userId, String fileId, String name) =>
    _baseUrl + "students/$userId/maakav/$maakavId/files/$fileId/$name";

static String _lessonCountUrl(String userId) =>
    _baseUrl + "students/$userId/lessonsCount";

static String _alfonUrl(String userId, String groupId) =>
    _baseUrl +
    (groupId == "-1" ? "students/$userId/alfon" : "groups/$groupId/alfon");

```

```
static const String _logoutUrl = _baseUrl + "logout";
```

```
static const String _apiVersion = "3.20210425";
```

```
static Map<String, String> _setJsonHeader() {
```

```
    Map<String, String> jsonHeader = { };
```

```
    jsonHeader["content-type"] = "application/json;charset=UTF-8";
```

```
    jsonHeader["accept"] = "application/json, text/plain, */*";
```

```
    jsonHeader["accept-language"] = "en-US,en;q=0.9;he;q=0.8";
```

```
    return jsonHeader;
```

```
}
```

```
Future<Result<E>> _process<E>(Future<Result<E>> data, Api api) {
```

```
    if (_dataProcessor != null) {
```

```
        if (data is Future<Result>) {
```

```
            data.then((result) {
```

```
                if (result.isSuccess) {
```

```
                    _dataProcessor!(result.value, api);
```

```
                }
```

```
            });
```

```
        } else {
```

```
            debugPrint(
```

```
                "Api controller data processor recieved data which is not of type Future<Result>");
```

```
            _dataProcessor!(data, api);
```

```
        }
```

```
    }  
    return data;  
}  
}
```

```
enum Api {  
    schools,  
    login,  
    grades,  
    bagrut,  
    behaveEvents,  
    groups,  
    timetable,  
    alfon,  
    messages,  
    message,  
    details,  
    messagesCount,  
    maakav,  
    homework,  
    hatamot,  
    hatamotBagrut,  
    lessonCount  
}
```

result.dart

```
class Result<E> {  
  
  dynamic exception;  
  
  E? value;  
  
  int? _statusCode;  
  
  int? get statusCode => _statusCode;  
  
  Result({this.exception, this.value, int? statusCode}) {  
  
    _statusCode = statusCode;  
  
  }  
  
  bool get isSuccess => exception == null && value != null && isOk;  
  
  bool get isUnauthorized => statusCode == 401;  
  
  bool get isInternalServerError => statusCode == 500;  
  
  bool get isNeedToLogin => isUnauthorized || isInternalServerError;  
  
  bool get isForbidden => statusCode == 403;  
  
  bool get isOk => statusCode == 200;  
  
}
```

models.dart

export 'result.dart';

export 'login/login.dart';

export 'login/login_data.dart';

export 'login/school.dart';

export 'login/student.dart';

export 'messages/attachment.dart';

export 'messages/conversation.dart';

export 'messages/message.dart';

export 'messages/message_title.dart';

export 'messages/messages_count.dart';

export 'user_data/bagrut.dart';

export 'user_data/behave_counter.dart';

export 'user_data/behave_event.dart';

export 'user_data/contact.dart';

export 'user_data/grade.dart';

export 'user_data/group.dart';

export 'user_data/hatama.dart';

export 'user_data/hatamat_bagrut.dart';

export 'user_data/homework.dart';

export 'user_data/lesson.dart';

export 'user_data/lesson_count.dart';

```
export 'user_data/maakav.dart';  
  
export 'user_data/short_behave_event.dart';
```

short_behave_event.dart

```
class ShortBehaveEvent {  
  
  String subject, justification;  
  
  int justificationId;  
  
  ShortBehaveEvent({  
  
    required this.subject,  
  
    required this.justification,  
  
    required this.justificationId  
  
  });  
  
}
```

maakav.dart

```
import 'package:grader_for_mashov_new/core/mashov_api/src/models/messages/attachment.dart';
```

```
import '../utils.dart';
```

```
class Maakav {
```

```
    DateTime date;
```

```
    String message, reporter, id;
```

```
    List<Attachment> attachments;
```

```
    Maakav({required this.id, required this.date, required this.message, required this.reporter, required this.attachments});
```

```
    static Maakav fromJson(Map<String, dynamic> src) => Maakav(
```

```
        id: Utils.integer(src["maakavId"]).toString(),
```

```
        date: DateTime.parse(src["maakavDate"]),
```

```
        message: Utils.string(src["message"]),
```

```
        reporter: Utils.string(src["reporterName"]),
```

```
        attachments: Utils.attachments(src["filesMetadata"]));
```

```
    }
```


lesson_count.dart

```
import '../utils.dart';
```

```
class LessonCount {
```

```
  int groupId, lessonsCount, weeklyHours;
```

```
  LessonCount({
```

```
    required this.groupId,
```

```
    required this.lessonsCount,
```

```
    required this.weeklyHours
```

```
  });
```

```
  static LessonCount fromJson(Map<String, dynamic> src) => LessonCount(
```

```
    groupId: Utils.integer(src["groupId"]),
```

```
    lessonsCount: Utils.integer(src["lessonsCount"]),
```

```
    weeklyHours: Utils.integer(src["weeklyHours"]),
```

```
  );
```

```
  @override
```

```
  String toString() {
```

```
    return super.toString() +
```

```
      " => {$groupId, $lessonsCount, $weeklyHours}";
```

```
  }
```

```
}
```

lesson.dart

```
import '../utils.dart';

class Lesson {

  int groupId, day, lesson;

  String subject, room, startTime, endTime;

  List<String> teachers;

  Lesson(

    {required this.groupId,

      required this.day,

      required this.lesson,

      required this.startTime,

      required this.endTime,

      required this.subject,

      required this.teachers,

      required this.room,});

  static Lesson fromJson(Map<dynamic, dynamic> src) {

    var tableData = src["timeTable"];

    var details = src["groupDetails"];

    bool tableDataNull = tableData == null;

    bool detailsNull = details == null;

    return Lesson(

      groupId: Utils.integer(tableDataNull ? null : tableData["groupId"]),
```

```

    day: Utils.integer(tableDataNull ? null : tableData["day"]),
    lesson: Utils.integer(tableDataNull ? null : tableData["lesson"]),
    startTime: Utils.string(src["startTime"]),
    endTime: Utils.string(src["endTime"]),
    subject: Utils.string(detailsNull ? null : details["subjectName"]),
    teachers: detailsNull
        ? null
        : details["groupTeachers"]
        .map<String>((t) => "${t["teacherName"]}")
        .toList(),
    room: Utils.string(tableDataNull ? null : tableData["roomNum"]));
}

@override
String toString() =>
    super.toString() +
        " => { $groupId, $day, $lesson,$startTime-$endTime,$subject, ${teachers
            .join(", ")} , $room }";
}

```

homework.dart

```
import '../utils.dart';

class Homework {

  String message, subject;

  DateTime date;

  Homework({required this.message, required this.subject, required this.date});

  static Homework fromJson(Map<String, dynamic> src) => Homework(

    message: Utils.string(src["homework"]),

    date: DateTime.parse(src["lessonDate"]),

    subject: Utils.string(src["subjectName"]));

}
```

hatamat_bagrut.dart

```
import '../utils.dart';

class HatamatBagrut {

  String hatama, moed, name, semel;

  HatamatBagrut({required this.hatama, required this.moed, required this.name, required this.semel});

  static HatamatBagrut fromJson(Map<String, dynamic> src) =>

    HatamatBagrut(
```

```
    hatama: Utils.string(src["hatama"]),  
    moed: Utils.string(src["moed"]),  
    name: Utils.string(src["name"]),  
    semel: Utils.string(src["semel"]));  
}
```

hatama.dart

```
import '../utils.dart';  
  
class Hatama {  
    int code;  
    String name, remark;  
  
    Hatama({required this.code, required this.name, required this.remark});  
  
    static Hatama fromJson(Map<String, dynamic> src) =>  
        Hatama(  
            code: Utils.integer(src["code"]),  
            name: Utils.string(src["name"]),  
            remark: Utils.string(src["remark"]));  
}
```

group.dart

```
import '../utils.dart';
```

```
class Group {
```

```
  int id;
```

```
  String subject;
```

```
  List<String>? teachers;
```

```
  Group({required this.id, required this.subject, this.teachers}) {
```

```
    teachers ??= [];
```

```
  }
```

```
  static Group fromJson(Map<String, dynamic> src) {
```

```
    return Group(
```

```
      id: Utils.integer(src["groupId"]),
```

```
      subject: Utils.string(src["subjectName"]).replaceAll("", "\\\""),
```

```
      teachers: src["groupTeachers"]
```

```
        .map<String>((t) => "${t["teacherName"]}")
```

```
        .toList(),
```

```
    );
```

```
  }
```

```
  @override
```

```
  String toString() =>
```

```
    super.toString() +
```

```

" => { id: $id, $subject" +
  (teachers!.isEmpty ? " - [{teachers!.join(", ")}]" : "") +
  " }";

}

```

grade.dart

```

import '../utils.dart';

class Grade {

  String teacher, subject, event;

  DateTime eventDate;

  int groupId, type, grade;

  Grade(

    {required this.teacher,

    required this.groupId,

    required this.subject,

    required this.eventDate,

    required this.event,

    required this.type,

    required this.grade});

  static Grade fromJson(Map<String, dynamic> src) {

```

```

return Grade(

    teacher: Utils.string(src["teacherName"]),

    groupId: Utils.integer(src["groupId"]),

    subject: Utils.string(src["subjectName"]),

    eventDate: DateTime.parse(src["eventDate"]),

    event: Utils.string(src["gradingEvent"]),

    type: Utils.integer(src["gradeTypeId"]),

    grade: Utils.integer(src["grade"]));

}

```

```

static List<Grade> fromJsonArray(List<dynamic> src) =>

    src.map((g) => fromJson(g)).toList();

```

```
@override
```

```

String toString() {

    return super.toString() +

        "-> { teacher: $teacher, groupId: $groupId, subject: $subject, eventDate:

        ${eventDate.toIso8601String()}(${eventDate.millisecondsSinceEpoch}), event: $event, type: $type,

        grade: $grade }";

}

}

```


contact.dart

```
import '../utils.dart';
```

```
class Contact {
```

```
    String name, parentClass, address, phone;
```

```
    Contact({required this.name, required this.parentClass, required this.address, required this.phone});
```

```
    static Contact fromJson(Map<String, dynamic> src) {
```

```
        var city = Utils.string(src["city"]);
```

```
        var street = Utils.string(src["address"]);
```

```
        return Contact(
```

```
            name: Utils.string(src["privateName"]) +
```

```
                " " +
```

```
                Utils.string(src["familyName"]),
```

```
            phone: Utils.string(src["cellphone"]),
```

```
            parentClass: Utils.string(src["classCode"]) +
```

```
                Utils.integer(src["classNum"]).toString(),
```

```
            address: street +
```

```
                (city.isEmpty ? "" : (street.isEmpty ? city : ", " + city)));
```

```
        }
```

```
    @override
```

```
String toString() =>
    super.toString() + " => { $name, $parentClass, $address, $phone }";
}
```

behave_event.dart

```
import '../utils.dart';

class BehaveEvent {

    int groupId, lesson, type, justificationId;

    String text, justification, reporter, subject;

    DateTime date;

    BehaveEvent(
        {required this.groupId,
        required this.lesson,
        required this.date,
        required this.type,
        required this.text,
        required this.justificationId,
        required this.justification,
        required this.reporter,
        required this.subject});
```

```
static BehaveEvent fromJson(Map<String, dynamic> src) => BehaveEvent(
```

```
    groupId: Utils.integer(src["groupId"]),
```

```
    lesson: Utils.integer(src["lesson"]),
```

```
    date: DateTime.parse(src["lessonDate"]),
```

```
    type: Utils.integer(src["lessonType"]),
```

```
    text: Utils.string(src["achvaName"]),
```

```
    justificationId: Utils.integer(src["justificationId"]),
```

```
    justification: Utils.string(src["justification"]),
```

```
    reporter: Utils.string(src["reporter"]),
```

```
    subject: Utils.string(src["subject"]));
```

```
@override
```

```
String toString() {
```

```
    return super.toString() +
```

```
        " => {$groupId, $lesson, ${date.toIso8601String()}, $type, $text, $justificationId, $justification,  
$reporter, $subject";
```

```
}
```

```
}
```

behave_counter.dart

```
import 'short_behave_event.dart';
```

```
class BehaveCounter {
```

```
    String subject;
```

```
    int lessonsCount, weeklyHours, totalEvents;
```

```
List<ShortBehaveEvent> events;
```

```
BehaveCounter({  
  required this.totalEvents,  
  required this.subject,  
  required this.weeklyHours,  
  required this.lessonsCount,  
  required this.events  
});  
}
```

bagrut.dart

```
import '../utils.dart';
```

```
class Bagrut {  
  String semel, name, moed, date, room, examStartTime, examEndTime;  
  int finalGrade, yearGrade, testGrade;
```

```
Bagrut(  
  {required this.semel,  
   required this.name,  
   required this.date,  
   required this.moed,  
   required this.examStartTime,  
   required this.examEndTime,
```

```
required this.room,  
required this.finalGrade,  
required this.yearGrade,  
required this.testGrade});
```

```
String stringify() => ""{  
    semel:    $semel,  
    name:     $name,  
    date:     $date,  
    moed:     $moed,  
    room:     $room,  
    finalGrade: $finalGrade,  
    yearGrade: $yearGrade,  
    testGrade: $testGrade,  
    times:    $examStartTime-$examEndTime,  
  
}"";
```

```
static Bagrut fromJson(Map<dynamic, dynamic> src) =>  
    Bagrut(  
        semel: Utils.string(src["semel"]),  
        name:  Utils.string(src["name"]),  
        date:  Utils.string(src["examDate"]),  
        moed:  Utils.string(src["moed"]),  
        room:  Utils.string(src["examRoomNumber"]),
```

```
    examStartTime: Utils.string(src["examStartTime"]),  
    examEndTime: Utils.string(src["examEndTime"]),  
    finalGrade: Utils.integer(src["final"]),  
    yearGrade: Utils.integer(src["shnaty"]),  
    testGrade: Utils.integer(src["test"]);  
}
```

messages_count.dart

```
import '../utils.dart';
```

```
class MessagesCount {
```

```
  int allMessages, inboxMessages, newMessages, unreadMessages;
```

```
  MessagesCount(
```

```
    {required this.allMessages,  
    required this.inboxMessages,  
    required this.newMessages,  
    required this.unreadMessages});
```

```
  static MessagesCount fromJson(Map<String, dynamic> src) => MessagesCount(
```

```
    allMessages: Utils.integer(src["allMessages"]),  
    inboxMessages: Utils.integer(src["inboxMessages"]),  
    newMessages: Utils.integer(src["newMessages"]),  
    unreadMessages: Utils.integer(src["unreadMessages"]));
```

```
  @override
```

```
  String toString() {
```

```
    return super.toString() +
```

```
      "{ $allMessages, $inboxMessages, $newMessages, $unreadMessages }";
```

```
  }
```

```
}
```

message_title.dart

```
import '../utils.dart';
```

```
class MessageTitle {
```

```
  String messageId, subject, senderName;
```

```
  DateTime sendDate;
```

```
  bool isNew, hasAttachment;
```

```
  MessageTitle(
```

```
    { required this.messageId,
```

```
    required this.subject,
```

```
    required this.senderName,
```

```
    required this.sendDate,
```

```
    required this.isNew,
```

```
    required this.hasAttachment});
```

```
  static MessageTitle fromJson(Map<String, dynamic> src) => MessageTitle(
```

```
    messageId: Utils.string(src["messageId"]),
```

```
    subject: Utils.string(src["subject"]),
```

```
    senderName: Utils.string(src["senderName"]),
```

```
    sendDate: DateTime.parse(src["sendTime"]),
```

```
    isNew: Utils.boolean(src["isNew"]),
```

```
    hasAttachment: Utils.boolean(src["hasAttachments"]));
```

```
  @override
```

```
  String toString() {
```



```

return super.toString() +

    " => { $messageId, $subject, $senderName, ${sendDate

        .toIso8601String()

        .split(".")

        .first}, $isNew, $hasAttachment";

    }

}

```

message.dart

```

import '../utils.dart';

import 'attachment.dart';

class Message {

    String messageId, subject, sender, body;

    DateTime sendDate;

    List<Attachment> attachments;

    Message(

        {required this.messageId,

        required this.sendDate,

        required this.subject,

        required this.body,

        required this.sender,

        required this.attachments});

    static Message fromJson(Map<String, dynamic> src) {

```

```

return Message(

  messageId: Utils.string(src["messageId"]),

  sendDate: DateTime.parse(src["sendTime"]),

  subject: Utils.string(src["subject"]),

  body: Utils.string(src["body"]),

  sender: Utils.string(src["senderName"]),

  attachments: Utils.attachments(src["files"]));

}

@override

String toString() {

  return super.toString() +

    " => { $messageId, $subject, $sender, ${sendDate.toIso8601String()}, ${attachments.length}
attachments";

}

}

```

conversation.dart

```

import '../utils.dart';

import 'message_title.dart';

class Conversation {

  String conversationId, subject;

  DateTime? sendTime;

  List<MessageTitle> messages;

  bool preventReply, isNew, hasAttachments;

```

```

Conversation(
    {required this.conversationId,
     required this.subject,
     this.sendTime,
     required this.messages,
     required this.preventReply,
     required this.isNew,
     required this.hasAttachments});

static Conversation fromJson(Map<String, dynamic> src) => Conversation(
    conversationId: Utils.string(src["conversationId"]),
    subject: Utils.string(src["subject"]),
    hasAttachments: Utils.boolean(src["hasAttachments"]),
    isNew: Utils.boolean(src["isNew"]),
    messages: src["messages"]
        .map<MessageTitle>((m) => MessageTitle.fromJson(m))
        .toList(),
    preventReply: Utils.boolean(src["preventReply"]));
}

```

attachment.dart

```
import '../utils.dart';
```

```
class Attachment {
```

```
    String? id, name;
```

```
    Attachment({ this.id, this.name });
```

```
    static Attachment fromJson(Map<String, dynamic> src) => Attachment(
```

```
        id: Utils.string(src["fileId"]), name: Utils.string(src["fileName"]));
```

```
    @override
```

```
    String toString() => super.toString() + " => { $id, $name }";
```

```
}
```

student.dart

```
class Student {

  Student({

    required this.id,

    required this.familyName,

    required this.privateName,

    required this.classCode,

    required this.classNum

  });

  String id, familyName, privateName;

  String? classCode;

  int? classNum;

  factory Student.fromJson(Map<String, dynamic> json) {

    return Student(

      id: json['childGuid'],

      familyName: json['familyName'],

      privateName: json['privateName'],

      classCode: json['classCode'],

      classNum: json['classNum']

    );

  }

}
```

```

Map<String, dynamic> toJson() => {
  'childGuid': id,
  'familyName': familyName,
  'privateName': privateName,
  'classCode': classCode,
  'classNum': classNum
};

@override
String toString() =>
  "Student(id=$id,privateName=\"$privateName\",familyName=\"$familyName\"";
}

```

school.dart

```

import 'dart:convert';
import '../utils.dart';

class School {
  int id;
  String name;
  List<int> years;

  String getYears() => Utils.listToString(years);

  School({required this.id, required this.name, required this.years});
}

```

```

static School fromJson(Map<String, dynamic> json) => School(
    id: json['semel'], name: json['name'], years: json['years'].cast<int>());

static List<School> listFromJson(String src) =>
    (json.decode(src) as List)
        .map((school) => School.fromJson(school))
        .toList();

Map<String, dynamic> toJson() => {'semel': id, 'name': name, 'years': years};
}

```

login_data.dart

```

class LoginData {
    LoginData(
        {required this.sessionId,
        required this.userId,
        required this.id,
        required this.userType,
        required this.schoolUserType,
        required this.schoolId,
        required this.year,
        required this.correlationId,
        required this.uniqueId});
}

```

```
String sessionId, userId, id, correlationId, uniqueId;
```

```
int userType, schoolUserType, schoolId, year;
```

```
factory LoginData.fromJson(Map<String, dynamic> json, String uniqueId) => LoginData(  
    sessionId: "",  
    userId: json['userId'],  
    id: "${json['idNumber']}",  
    userType: json['userType'],  
    schoolUserType: json['schoolUserType'],  
    schoolId: json['schoolId'],  
    year: json['year'],  
    correlationId: json['correlationId'],  
    uniqueId: uniqueId);
```

```
Map<String, dynamic> toJson() => {  
    'sessionId': "",  
    'userId': userId,  
    'id': id,  
    'userType': userType,  
    'schoolUserType': schoolUserType,  
    'schoolId': schoolId,  
    'year': year,  
    'correlationId': correlationId,  
    'uniqueId' : uniqueId  
};
```



```
@override

String toString() {

    return 'LoginData{sessionId: $sessionId, userId: $userId, id: $id, correlationId: $correlationId,
uniqueId: $uniqueId, userType: $userType, schoolUserType: $schoolUserType, schoolId: $schoolId,
year: $year}';

}

}
```

login.dart

```
import 'login_data.dart';

import 'student.dart';

class Login {

  Login({required this.data, required this.students, required this.userSchoolYears});

  LoginData data;

  List<Student> students;

  List<int> userSchoolYears;

  factory Login.fromJson(Map<String, dynamic> src, String uniqueId) {

    var credential = src["credential"];

    var token = src["accessToken"];

    if (token != null) {

      List<dynamic> children = token["children"];

      LoginData data = LoginData.fromJson(credential, uniqueId);

      List<Student> st = children.map((student) {

        return Student.fromJson(student as Map<String, dynamic>);

      }).toList();

      List<int> uSY = List<int>.from(token["userSchoolYears"]);

      return Login(data: data, students: st, userSchoolYears: uSY);

    } else {

      throw Exception("token is null");

    }

  }

}
```

```

    }

}

static List<String> listToStringsList(List list) {

    List<String> strings = [];

    for (int i = 0; i < list.length; i++) {

        strings.add(list[i].toString());

    }

    return strings;

}

@Override

String toString() {

    return 'Login{ data: $data, students: $students, userSchoolYears: $userSchoolYears}';

}

}

```