Technologic Papers

# איך להסתפר מבלי לחכות בתור - מחקר אפליקציית EasyTor

מאת טל סעדי

## הקדמה

המאמר הבא הוא השני בסדרה של מאמרים שמטרתה להראות דרכי ניצול של אפליקציות יומיומיות לטובת השגת מידע רגיש של משתמשים או לטובת ניצול לרעה של דרכי פעולה לגיטימיות של האפליקציה. הקו המנחה של המאמרים הוא מציאת חולשות לוגיות, שלעתים אינן חולשות כלל, שכל מה שדרוש על מנת לנצל אותן זה הבנה בסיסית של דרך הפעולה של הלקוח וכמה כלי מחקר פשוטים. אם נזקק את כל זה למשפט אחד, אז אפשר להגיד שלפעמים כדי למצוא תפוח זהב, עדיף לחפש על הרצפה מאשר לנסות לטפס על העץ. מי שלא קרה את המאמר הראשון בסידרה ומעוניין - יכול לעשות זאת כאן.

## רקע

באחד מהביקורים האחרונים שלי בעיר באר שבע, הסתובבתי ביחד עם חבר בין השכונות החביבות של העיר, כשלפתע משום מקום הוא פנה אליי עם המשפט הבא: "אני חייב ללכת להסתפר, עכשיו!". ברור שבתור חבר רציונלי, עזרתי לו מיד למצוא את המספרה הקרובה ביותר. צעדנו לנו עד לפתח המספרה ושאלנו את הספר את השאלה המתבקשת שהוא שומע כמה פעמים טובות ביום: "כמה יש בתור?". ההפתעה האמיתית הגיעה כאשר במקום לקבל תשובה קלאסית של "שני הילדים שפה ואז הבחורצ'יק מאחורה" או "מפוצץ עכשיו חברה, תחזרו בצהריים", הוא ענה לנו את התשובה הבאה:

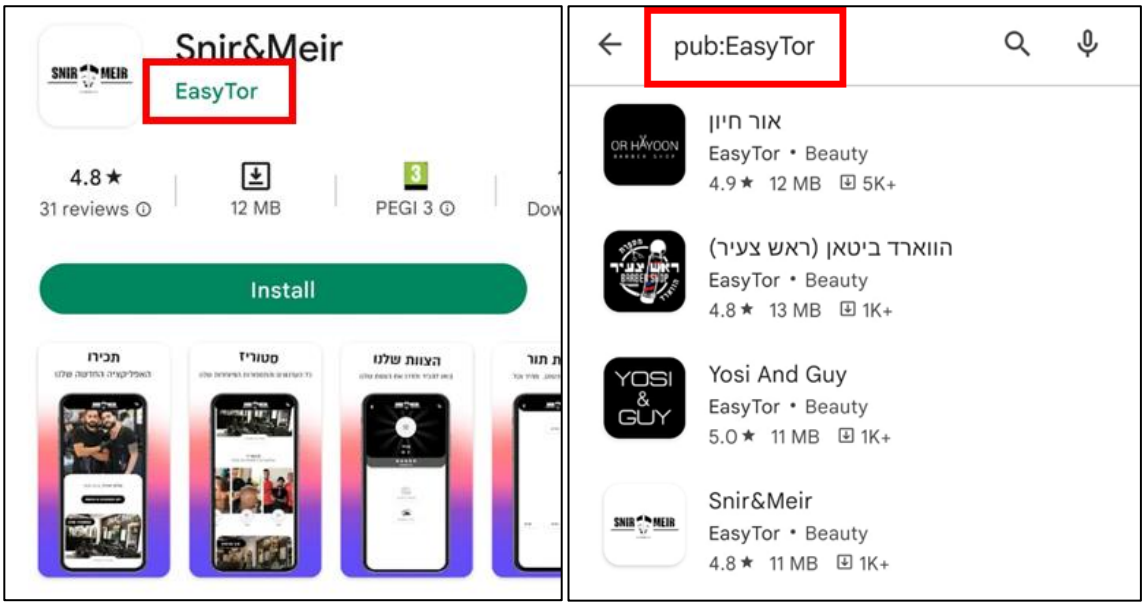
"אתם לא יכולים להיכנס ככה, חייבים להזמין תור באפליקציה". שנינו הסתכלנו אחד על השני במעין מבט שתוהה האם אנחנו בניו יורק של 2030 או בבאר שבע של 2020, אבל קיבלנו את הבקשה והלכנו להוריד את האפליקציה. רק בדיעבד, בשיחות עם חברים נוספים, נגלה אליי הטרנד החדש של עולם הספרות ושל עולם העצמאים בכלל, והוא אפליקציות מותאמות לניהול תורים לעסק הפרטי.

## EasyTor

חברת EasyTor היא בין האחראיות הגדולות למהפכת האפליקציות לניהול תורים שהוטמעו בשוק הישראלי בצורה רחבה לאחרונה. כבר בחנות האפליקציות, בחיפוש אחרי האפליקציה של הספר שלנו, אפשר לראות את היקף הלקוחות של החברה ואת כמות האפליקציות שהיא סיפקה ללקוחות פרטיים שרוצים לנהל את התורים שלהם.

בעמוד האפליקציה של הספר שלנו אפשר לראות את שם החברה שפיתחה את האפליקציה, ובלחיצה על שם החברה ניכנס למסך סינון שמראה לנו את כל האפליקציות האחרות שפותחו על ידי אותה חברה.

הצילום הזה מכיל רק את התוצאות הראשונות, שכוללות גם הן שמות גדולים בתחום הספרות כמו Snir & Meir שידועים בתור מעצבי שיער של סלבס ושחקני כדורגל מפורסמים, אבל בגלילה למטה בעמוד אפשר לראות עוד עשרות אפליקציות שונות שהחברה פיתחה שמגיעות ביחד לעשרות אלפי הורדות.



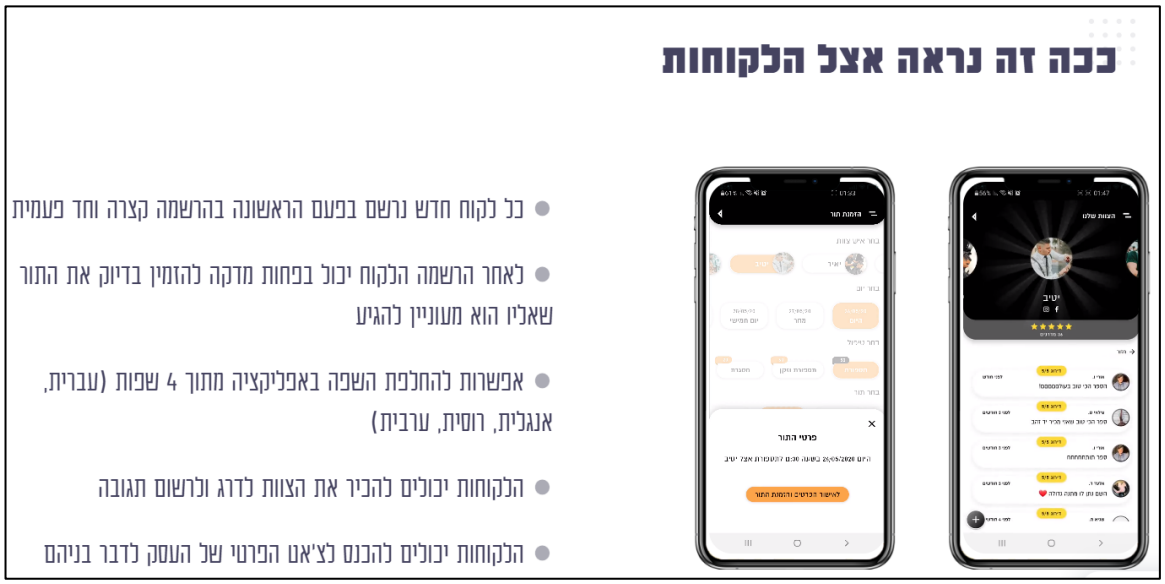

## המוצר

לפי אתר החברה, המוצר שהיא מספקת ללקוחות מוגדר כך:

*"מערכת ייחודית לניהול תורים בצורה פשוטה ויעילה שמגיעה עם אפליקציה ייחודית לאיפון ואנדרואיד לכל בעל עסק עם המון פונקציונליות ושליטה מלאה על הכל מתוך ממשק הניהול! כל מה שנשאר לעשות זה להפנות את הלקוחות לאפליקציה הייחודית של העסק שלך ומשם בצורה קלה ופשוטה הם יוכלו להזמין תור בכל רגע נתון, לקבל תזכורות, לראות עבודות שלך, להחליף תורים בניהם בצ'אט ולהכיר את הצוות של העסק."*

למוצר של החברה יש בעצם שני צדדים של לקוחות, יש קודם כל את בעל העסק, שהוא גם לקוח של החברה אבל גם נותן שירות ללקוחות אחרים, ויש את הלקוחות שלו. לכן גם לאפליקציה של החברה יש שני צדדים, ממשק לבעל העסק לניהול העסק שלו, וממשק ללקוחות העסק לקביעת תורים קלים:

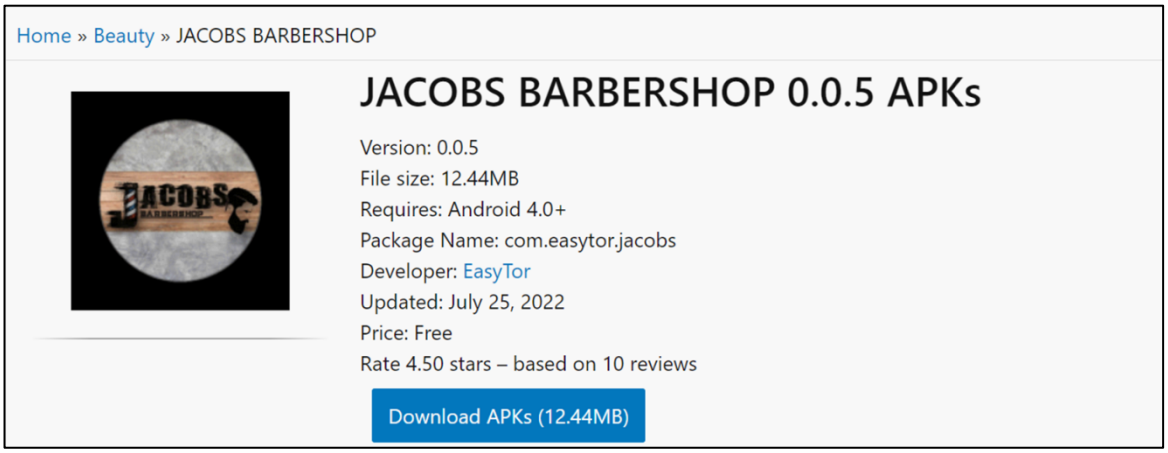
[הצד של בעל העסק]



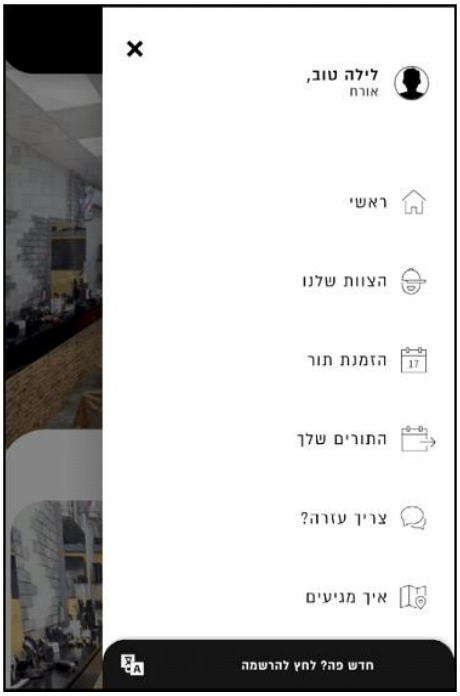

[הצד של הלקוחות בעסק]

## אז איך מתחילים לחקור?

למי שלא קרא את המאמר הקודם, הצגתי בו הקדמה לגבי מחקר אנדרואיד באופן כללי, כולל שיטות וכלים שאפשר להשתמש בהם. לכן בשביל להמשיך לחלק הבא של המאמר, אמליץ לסיים לקרוא את המאמר הקודם בקישור הבא, ואז לחזור אלינו. אז נתחיל כמובן בליצור emulator חדש ב-Genymotion ונוריד את ה-APK של הספר שלנו מאתר Mirror לבחירתנו (זאת הוכחה שבאתרים האלה יש באמת הכל...):




הקטע המיוחד במחקר שלנו עכשיו, הוא שלחברת EasyTor אין אפליקציה אחת שאנחנו יכולים לחקור, אלא שיש לה אפליקציה לכל לקוח ולכן נחקור אפליקציה אחת מבין רבות שהיא יצרה. החלק הטוב הוא שלחברה יש Template גנרי של אפליקציה שעליו היא מלבישה עיצוב ספציפי לכל לקוח, ולכן מספיק לחקור לקוח אחד כדי להבין את כל המוצר של החברה, ונראה זאת גם בהמשך. עכשיו אחרי שהתקנו את האפליקציה על האימולטור, נוכל לראות את את המסך הראשי ואת התפריט הצדדי שמאפשר לנו לעשות פעולות:



נפתח במקביל את ה-APK ב-Jadx ונתחיל כמובן לעבור על ה-Manifest. כמו שאמרתי במאמר הקודם, אני אוהב להתחיל לעבור על Flow-ים שמתחילים ב-MainActivity ולכן נמצא שוב את ה-Activity שמוגדר ראשון תחת הגדרת האפליקציה ב-AndroidManifest.xml:

```
<application android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:hardwareAccelerated="true" android:supportsRtl="true" androic
    <activity android:theme="@style/Theme.AppCompat.NoActionBar" android:label="@string/activity_name" android:name="com.easytor.jacobs.MainActivity"
        <intent-filter android:label="@string/launcher_name">
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
```

אבל, למרבה ההפתעה, אחרי שנכנסים ל-MainActivity רואים שהוא מכיל פונקציה אחת עם מספר שורות קוד בודדות:

```
/* loaded from: classes.dex */
public class MainActivity extends CordovaActivity {
    @Override // org.apache.cordova.CordovaActivity, androidx.fragment.app.FragmentAct
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Bundle extras = getIntent().getExtras();
        if (extras != null && extras.getBoolean("cdvStartInBackground", false)) {
            moveTaskToBack(true);
        }
        loadUrl(this.launchUrl);
    }
}
```

ולא רק זה, אם מסתכלים בתיקיית הפרויקט, רואים שיש שם 3 קבצים בודדים בלבד, ששניים מהם הם בכללי קבצי קונפיגורציה וקישור:

אז מה קורה פה? איך יכול להיות שיש רק קובץ אחד עם Activity אחד שמכיל כמה שורות בודדות? איפה כל האפליקציה? הדבר היחיד שאנחנו כן רואים בינתיים הוא שה-MainActivity יורש ממשהו שנקרא CordovaActivity וקורא לפונקציה אחת בשם loadUrl.

## Apache Cordova

האפליקציה שאנחנו מסתכלים עליה משתמשת ב-Framework שנקרא Apache Cordova. ה-Framework הזה נועד לפיתוח אפליקציות מובייל בעזרת פיתוח Web (HTML, CSS, JS) והוא מאפשר למפתחים ליצור יישומים לאנדרואיד ול-IOS בזמן שרוב התוכן של האפליקציה מגיע מהרשת ומוצג בדפדפן פנימי, עם מעטפת שמאפשרת גישה לתכונות שזמינות לאפליקציות מובייל רגילות, כמו התראות, גישה למיקום, מסך טעינה ועוד. קורדובה מספקת לנו את המעטפת שמתקשרת עם מערכת ההפעלה ונותנת לנו להגדיר מה יהיה המידע שהאפליקציה תפתח בדפדפן הפנימי.

מצד אחד הביצועים של אפליקציות כאלו יכולים להיות מוגבלים ביחס לאפליקציות רגילות, אבל מצד שני זה נותן לנו יכולות טובות יותר מאתרים פשוטים בעיקר בגלל נראות של אפליקציה שמותקנת על המכשיר ויכולת התראות שלא קיימת באתרים רגילים שנפתחים על ידי דפדפן.

כיום ה-Framework הזה מנוהל על ידי חברת Adobe, והיא משווקת גם שירות שנקרא Apache Cordova Build שמאפשר למתכנתים להעלות קוד מקור של Html, CSS ו-Java Script ל-"Cloud Compiler" המייצר אפליקציות לכל פלטפורמה נתמכת. ה-Framework של Adobe נחשב ליחסית מיושן (למרות שעדיין יש כמות גדולה מאוד של אפליקציות שמשתמשות בו) והמחליף העכשווי שלו הוא React-Native שמאפשר גם הוא פיתוח אפליקציות Cross-Platform.

ועכשיו אחרי שאנחנו מבינים מה זה בכלל Cordova ואיך יכול להיות שאין כמעט קוד באפליקציה שלנו, אפשר גם להסיק למה החברה בחרה להשתמש בתשתית הזאת. המקרה של EasyTor הוא דוגמא קלאסית למקום שבו נצטרך אפליקציה גנרית מינימלית שנוכל להתאים בצורה קלה ומהירה לכל הלקוחות, ומה יותר נוח מלהתאים טיפה את עמודי ה-HTML ו-CSS כך שיתאימו לעיצוב שהלקוח שלנו רוצה? אז עכשיו השאלה היחידה שנשארה היא...

## מאיפה טוענים את האתר?

בואו נצלול חזרה לקוד. מה שאנחנו רואים שקורה ב-MainActivity זה טעינה של האתר בעזרת הפונקציה loadUrl שמקבלת כפרמטר את המשתנה launchUrl. המשתנה launchUrl יכול להיות כתובת שנמצאת באינטרנט שאותה נטען בדפדפן הפנימי שלנו ונציג ככל אתר רגיל, אבל במקרה שלנו הכתובת launchUrl מצביעה דווקא על קובץ מקומי... כדי למצוא את הערך של המשתנה התחלתי בתהליך הבא:

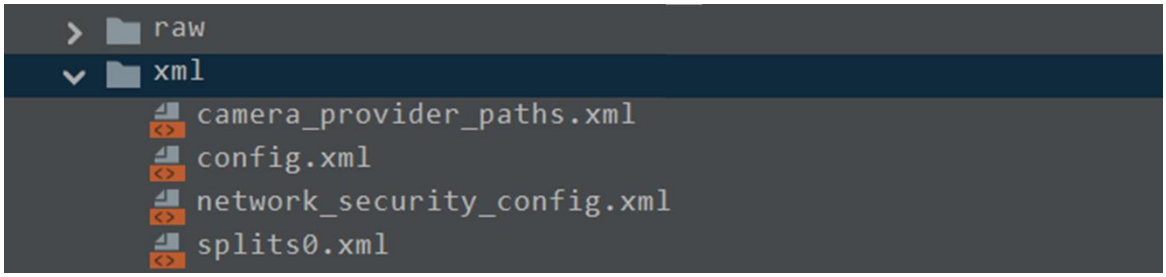
חיפשתי את כל המקומות שבהם משתמשים במשתנה ומצאתי את שתי הפונקציות שרואים בתמונה. נכנסתי לפונקציה שעושה השמה למשתנה עם ערך מהקונפיגורציה ונקראת loadConfig.

```
protected void loadConfig() {
    ConfigXmlParser configXmlParser = new ConfigXmlParser();
    configXmlParser.parse(this);
    CordovaPreferences preferences = configXmlParser.getPreferences();
    this.preferences = preferences;
    preferences.setPreferencesBundle(getIntent().getExtras());
    this.launchUrl = configXmlParser.getLaunchUrl();
    this.pluginEntries = configXmlParser.getPluginEntries();
    Config.parser = configXmlParser;
}
```

לאחר מכן עברתי להסתכל על הפונקציה parse של האובייקט ConfigXmlParser כדי לראות מאיפה טוענים את הקונפיגורציה:

```
public void parse(Context action) {
    int identifier = action.getResources().getIdentifier("config", "xml", action.getClass().getPackage().getName());
    if (identifier == 0 && (identifier = action.getResources().getIdentifier("config", "xml", action.getPackageName()))
        LOG.e(TAG, "res/xml/config.xml is missing!");
        return;
    }
    this.pluginEntries.add(new PluginEntry(AllowListPlugin.PLUGIN_NAME, "org.apache.cordova.AllowListPlugin", true));
    parse(action.getResources().getXml(identifier));
}
```

ראיתי שהפונקציה טוענת את הקונפיגורציה מקובץ שנמצא בנתיב res/xml/config.xml. מצאתי את הקובץ בתוך ה-APK ופתחתי אותו:




ראיתי שבקובץ באמת יש הגדרות שונות כמו השם של האפליקציה והכותרת במסך הבית, אבל לא מצאתי שום משתנה שנקרא launchUrl.

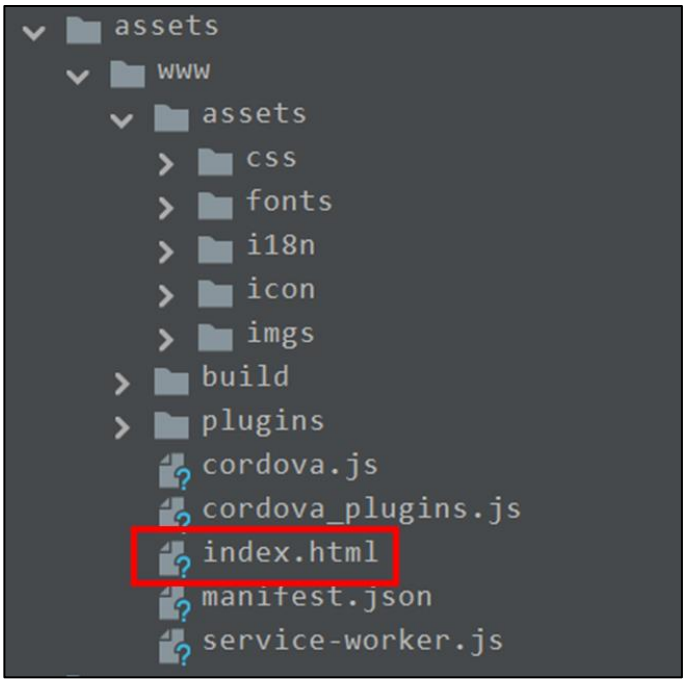
חזרתי להסתכל על הפונקציה getLaunchUrl של האובייקט ConfigXmlParser וראיתי שבמקרה שאין משתנה עם הנתיב בקובץ הקונפיגורציה, לוקחים את הנתיב ממשתנה אחר בשם contentSrc:

```
public String getLaunchUrl() {
    if (this.launchUrl == null) {
        setStartUrl(this.contentSrc);
    }
    return this.launchUrl;
}
```

המשתנה הזה דווקא כן קיים בקונפיגורציה והאפליקציה מפרסרת אותו מהתגית content:

```
<name>JACOBS BARBERSHOP
</name>
<description>An awesome Ionic/Cordova app.
</description>
<author email="hi@ionicframework.com" href="http://ionicframework.com/">
</author>
<content src="index.html"/>
<allow-intent href="http://*/*"/>
<allow-intent href="https://*/*"/>
<allow-intent href="tel:*"/>
<allow-intent href="sms:*"/>
<allow-intent href="mailto:*"/>
```

אבל מה זה אומר שהנתיב שלנו זה index.html? איפה נמצא התוכן של האתר אם הנתיב שלנו הוא קובץ לוקאלי? אז במקרה שלנו, כל האתר מגיע דחוס ביחד עם ה-APK תחת תיקיית ה-Assets ולא נמצא בכלל על שרת באינטרנט! לכן, נוכל לחלץ את קוד המקור של האתר מהAPK ולהתחיל לעבור על הקוד ולחקור אותו.




## איך חוקרים אתר?

אז המחקר שלנו קיבל תפנית קצת שונה וממחקר אפליקציות עברנו למחקר אתרים, אבל מחקר זה מחקר ואנחנו מסוגלים להכל. הדבר הראשון שנרצה לחפש בקוד של האתר כמו בקוד של אפליקציה זה את התקשורת מול השרת ואת ה-API שבו הלקוח משתמש כדי לבצע בקשות שונות. לצורך החלק הזה אני רוצה להציג עוד כלי חביב נוסף שיכול לעזור לנו מאוד עם הסיטואציה.

## HTTP Toolkit

בקהילת חוקרי התקשורת יש איזשהו קונצנזוס לגבי כלי שחלקכם יכירו טוב שנקרא BurpSuite, ובצדק. זה אחד הכלים הכי טובים להסנפות תעבורה ולמחקר API ואפשר לכתוב עליו מאמר שלם, אבל היום אני רוצה לתמוך בעסקים הקטנים ולהציג כלי חדש ופחות מוכר שהתגלה כאחד הכלים הכי טובים שיצא לי להשתמש בהם. בגדול HTTP Toolkit זה כלי דומה מאוד ל-BurpSuite שמכיל יכולות דומות ובעצם משמש לנו כמעין Proxy בין הטלפון שלנו לשרת שתופס את כל הבקשות שעוברות.

הכלי מציג לנו את הבקשות בצורה נוחה ונותן לנו לבצע כל מיני פעולות שונות. החלק המדהים ב-HTTP Toolkit הוא קודם כל קלות התפעול וההתקנה שלו על המכשירים, והתצוגה הברורה מאוד שהוא מספק למשתמש, מה שמראה שבאופן כללי שהדגש העיקרי של המפתחים היה הנוחות של המשתמש.

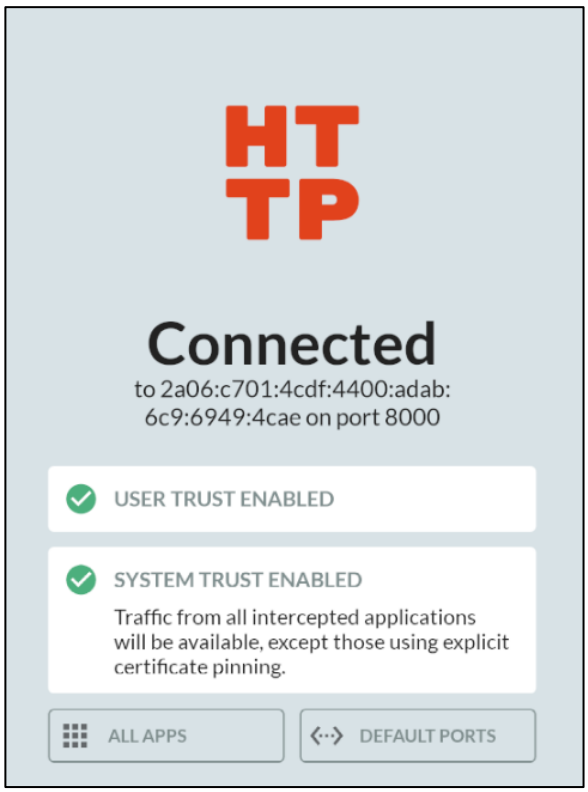
BurpSuite הוא כלי מטורף עם יכולות בלתי נגמרות, אבל בשביל לשים Proxy על מכשיר דרכו נצטרך לעבור ב-5 מדריכים שונים באינטרנט וגם אחרי שנצליח התצוגה שלו תהיה לא ברורה בכלל לאנשים שמשתמשים בו בפעם הראשונה.

אז בואו נתחיל ונראה מה נוכל לתפוס בתעבורה של האפליקציה הפעם. נרים Proxy על המכשיר בעזרת הכפתור הפשוט הבא:

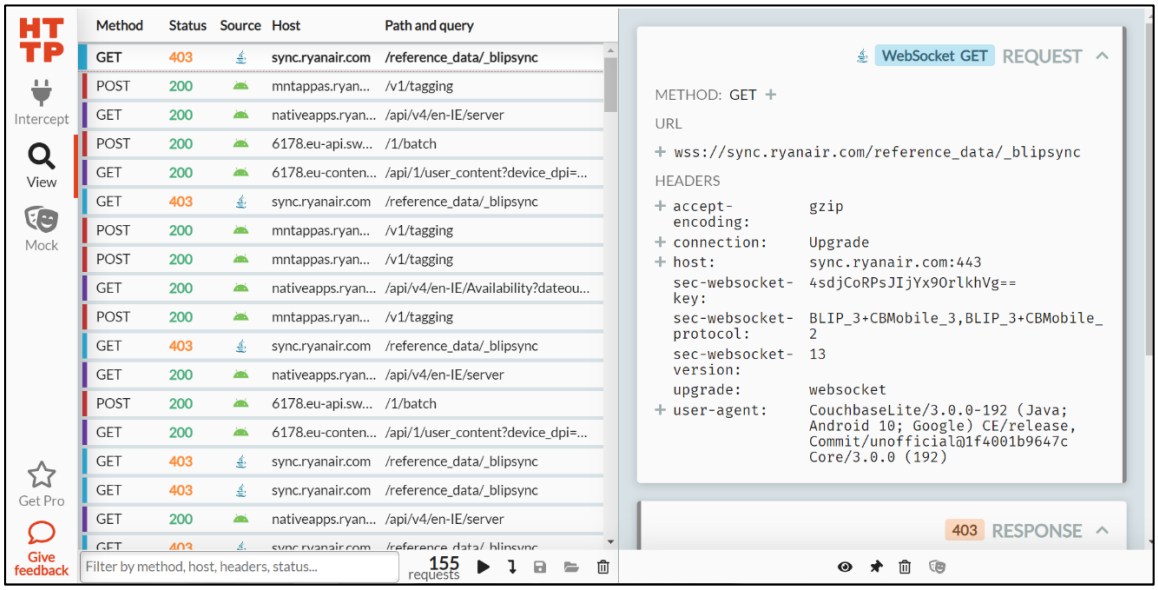
בתוך שניות נראה את הקסם קורה ודברים יקרו מעצמם על מסך הטלפון שלנו. מה שבעצם קורה ברקע זה שהתוכנה מריצה פקודות על המכשיר בעזרת ADB ומקנפגת לנו שני Certificate-ים שהיא יצרה.

Certificate אחד היא מתקינה בהרשאות User למקרה שהמכשיר שלנו ללא Root, וCertificate אחד היא מתקינה בהרשאות System במקרה שהמכשיר שלנו הוא Rooted או Emulator שמגיע עם הרשאות גבוהות. ה-Certificate-ים האלו נועדו כדי שנוכל לקלף את שכבת ההצפנה בפרוטוקל HTTPS כדי לראות ממש את התוכן שנשלח בתוך הפקטות. יש אפליקציות שיזרמו עם כל User Certificate שניתן להם, אבל יש אפליקציות שיקשו עלינו את החיים ולכן נצטרך להתקין System Certificate שיקבל את כל התעבורה של המכשיר.

מיד בסיום של ה-Setup האפליקציה תפתח לנו את המסך הבא בו נוכל לראות את הסטטוס של הProxy ולקנפג פורטים ואפליקציות מתאימות שנרצה:




החלק המהפכני פה הוא שאת כל התהליך הזה היינו צריכים לעשות בצורה ידנית אם היינו משתמשים ב-BurpSuite וכאן הכל קורה לבד. עכשיו תוכנת ה-HTTP Toolkit שלנו שמותקנת במחשב תעבור גם היא בצורה אוטומטית לעמוד חדש שבו נוכל לראות בצורה יפה את כל הבקשות שנשלחות מהמכשיר, כולל ה-Method, הסטטוס, ה-Source, ה-Host שאליו שולחים, והנתיב שאותו אנחנו מחפשים:



בצד שמאל נוכל לראות את הבקשות ובלחיצה על בקשה מסויימת יפתח לנו בצד ימין תפריט שנותן לנו לראות עוד פרטים על הבקשה כולל כל ה-Header-ים שלה והתוכן, וכנ"ל גם על הפרטים מהתשובה של השרת. עכשיו נוכל לפתוח את האפליקציה של EasyTor ולהתחיל לראות איזה בקשות ה-Client שולח בעליה ובפעולות מסוימות שאנחנו עושים. מההתחלה אפשר לראות בהסנפה שהאפליקציה שולחת הרבה בקשות לכתובת app.eztor.io/api. זאת הכתובת שדרכה מתאמתים ומבקשים את המידע של הלקוח מתאים, אבל השאלה היא איך השרת יודע לזהות איזו אפליקציה של איזה לקוח מדברת איתו? בשביל זה פתחתי שוב את הקוד של האתר והתחלתי לחפש איפה בקוד שולחים בקשות לדומיין שראינו בהסנפה ואיך עושים את זה.

אחרי חיפוש קצר גיליתי שהדומיין מופיע פעם אחת בקוד במחלקה הזו:

```
var MainServiceProvider = /** @class */ (function () {
    function MainServiceProvider(http, transfer, LoginService) {
        this.http = http;
        this.transfer = transfer;
        this.LoginService = LoginService;
        this.request = {};
        this.constants = __WEBPACK_IMPORTED_MODULE_3__shared_constants__["a" /* Constants */];
        console.log('Hello MainServiceProvider Provider');
        this.initiate();
    }
```

המחלקה MainServiceProvider היא האחראית על בקשות ה-API, והתקשורת של הלקוח עם השרת והיא זאת שמגדירה את כל הבקשות. ממש בתחילת המחלקה, אפשר לראות שמגדירים את המבנה של Request-ים באופן כללי במשתנה שנקרא request ונראה כך:

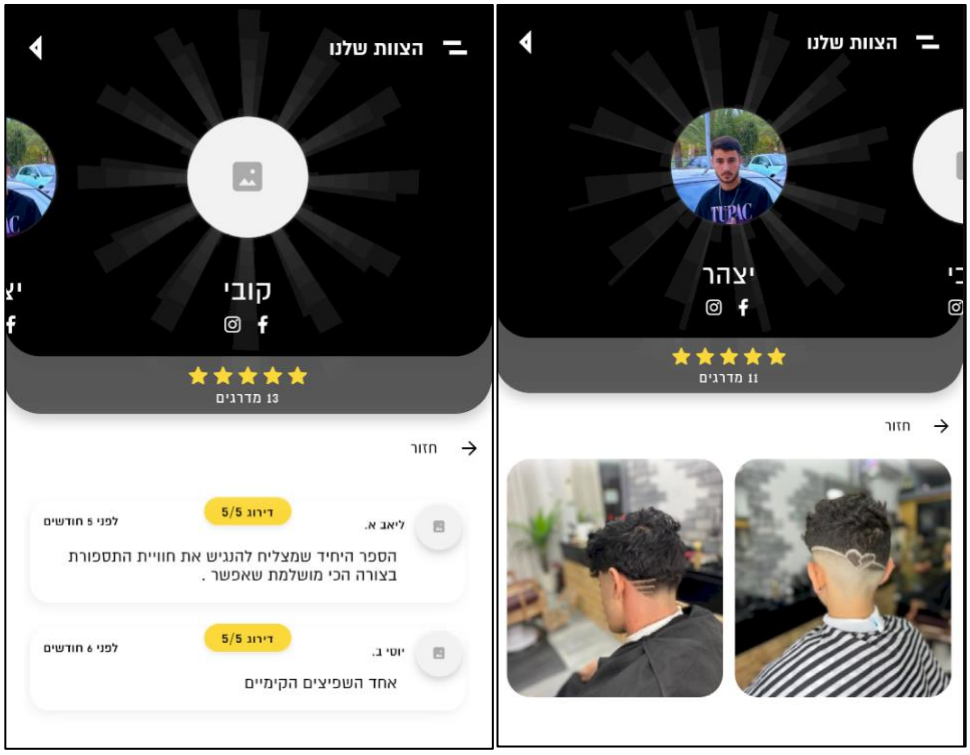
```
MainServiceProvider.prototype.initiate = function () {
    this.request = {
        url: "https://app.eztor.io/api",
        // url: "http://localhost:4000/api",
        wp_api_url: "",
        headers: {
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': 'POST, GET, OPTIONS, PUT',
            'uniqeabc': __WEBPACK_IMPORTED_MODULE_3__shared_constants__["a" /* Constants */].app.uniqeabc
        },
        params: {
            'appId': __WEBPACK_IMPORTED_MODULE_3__shared_constants__["a" /* Constants */].app.appId,
        }
        /*headers: {
          "Content-Type": 'application/json; charset=utf-8',
          "Access-Control-Allow-Credentials": "true",
          "Access-Control-Allow-Headers": "Content-Type,Authorization,Upgrade-Insecure-Requests"
        }*/
        /*params: {
          'appId': Constants.app.appId,
          'appSecret': Constants.app.appSecret
        }*/
    };
};
```

ה-Url של הבקשה הוא הדומיין שראינו בהסנפה (ועוד url שנשאר בהערה מפיתוח בסביבה לוקאלית ב-localhost:4000). בבקשה מוגדרים גם ה-Header-ים שביניהם שני Header-ים שאחראים על הגדרות ה-

Access Control ועוד Header אחד לא מזוהה בשם uniqeabc שכדאי שתזכרו להמשך... חוץ מזה מעבירים עוד פרמטר לבקשה בשם appId ואפשר לראות בהערה שאולי היה בתכנון להעביר גם משתנה בשם appSecret. לאחר מכן מתחילים להגדיר במחלקה את ה-Endpoints השונים לכל סוגי הבקשות כמו סטורי למסך הבית או פרטים של העובדים:

```
////////// Stories to home page//////////
MainServiceProvider.prototype.getStories = function () {
    return this.http.get(this.request.url + '/GetAllStories', {
        params: this.request.params
    }).map(this.extractData)
        .catch(this.catchError);
};
MainServiceProvider.prototype.updateStorySeen = function (storyObj) {
    return this.http.post(this.request.url + '/StorySeen', {
        storyObj: storyObj
    }).map(this.extractData)
        .catch(this.catchError);
};
MainServiceProvider.prototype.GetOrder_ = function (id) {
    return this.http.post(this.request.url + '/GetOrder', {
        id: id
    }).map(this.extractData)
        .catch(this.catchError);
};
////////// EmployyeInfo //////////
MainServiceProvider.prototype.getEmployyessCommentsAndGallery = function () {
    return this.http.get(this.request.url + '/getEmployyessCommentsAndGallery', {
        params: this.request.params
    }).map(this.extractData)
        .catch(this.catchError);
};
```

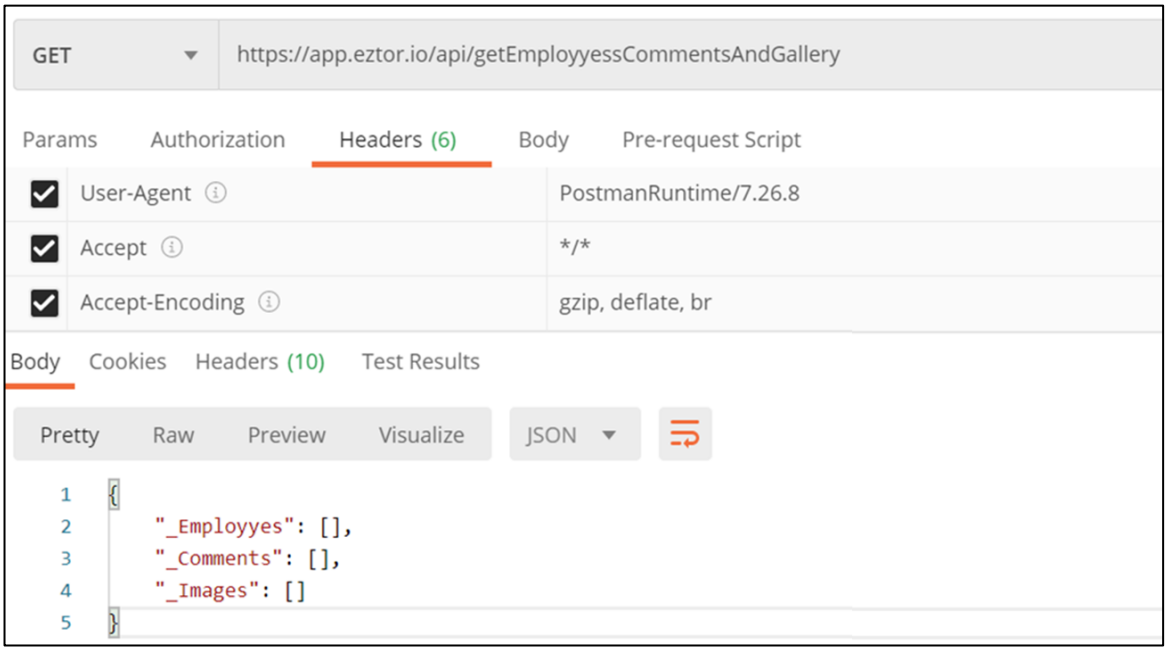
יש במחלקה עוד כמה עשרות Endpoints שונים שכל אחד מהם מקבל פרמטרים קצת שונים ומחזיר תשובות בהתאם.אז החלטתי לנסות לשחזר בקשה בעצמי ב-Postman ולראות איזה שדות השרת דורש כדי להחזיר לי את המידע הרצוי. בחרתי להתחיל עם פיצ'ר שקיים באפליקציה שמאפשר לקבל מידע על העובדים בחנות ולראות דירוג שלהם עם תגובות של משתמשים:



ה-API של הבקשה שמכילה את המידע בעמוד הזה נראה ככה:

```
////////// EmployyeInfo ////////////
MainServiceProvider.prototype.getEmployyessCommentsAndGallery = function () {
    return this.http.get(this.request.url + '/getEmployyessCommentsAndGallery', {
        params: this.request.params
    }).map(this.extractData)
        .catch(this.catchError);
};
```

ועכשיו ננסה לבנות אותה ב-Postman:




התחלתי מלשלוח בקשה פשוטה ל-Endpoint הרלוונטי בלי להוסיף שום header-ים של האפליקציה ובתשובה השרת החזיר לי רשימות ריקות שזה הגיוני בהתחשב בעובדה שהוא לא יודע מאיזה אפליקציה הבקשה.

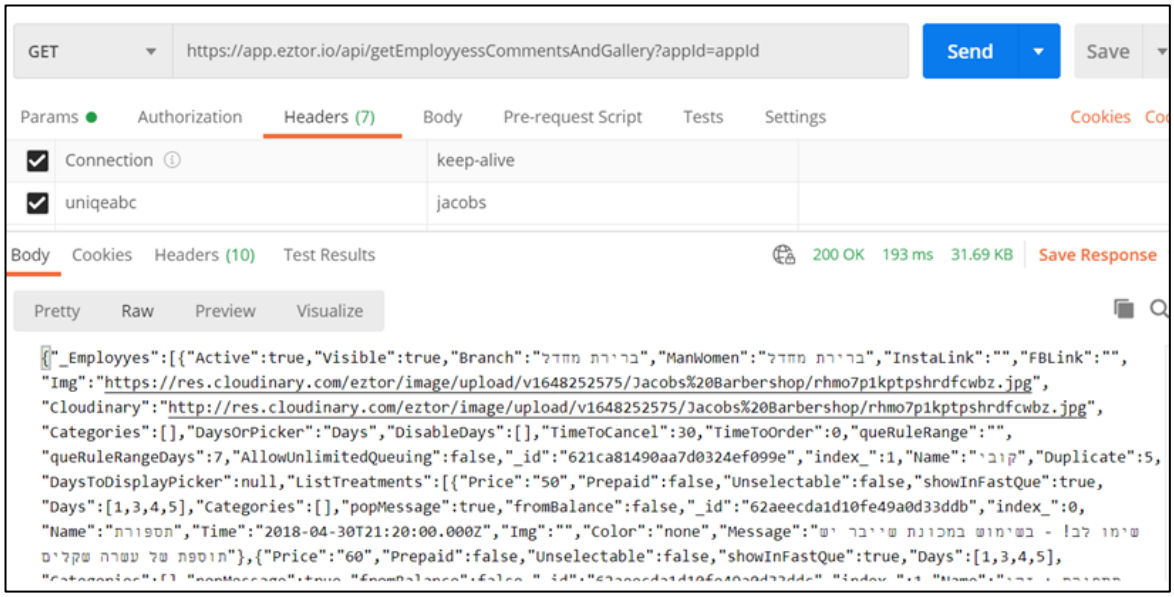
המסקנה המתבקשת מכל המידע שאספנו עד עכשיו הוא שאם הדבר היחיד שנשלח בבקשה זה header בשם uniqeabc כנראה שזה מה שמאמת אותנו מול השרת ואומר לו איזה מידע להחזיר לנו ומה אנחנו מורשים לראות. במילים אחרות ה-header הזה הוא בעצם ה-token שלנו לתקשורת מול השרת, זה הגיוני גם בהתחשב בעובדה שיש מסך הרשמה באפליקציה ואת כל הפעולות אפשר לעשות רק אחרי שנרשמיםומתאמתים עם מספר טלפון וסמס.

אז מה החלק המעניין פה? שכל זה בכלל לא נכון. בעצם אני משקר, רק חלק מזה לא נכון, ה-token הזה באמת אומר לשרת מאיזו אפליקציה אנחנו שולחים את הבקשה, אבל אין בו שום מידע לגבי הלקוח ששולח את הבקשה!

אפשר לראות שהשדה הזה מגיע מהמקום בו מגדירים קבועים באתר:

```
Constants.app = {
    appId: 'AppId',
    appName: 'Jacobs',
    uniqeabc: 'jacobs',
    appSecret: 'AppSecret',
    sender_id: '423726850901',
    oneSignalAppId: 'd0ff1efb-9342-4487-ac20-ce4cf53055d7',
    appSmsName: "Jacobs",
    jsonApiUrl: "https://jacobs.eztor.co.il/files/",
    restApiUrl: "https://jacobs.eztor.co.il/api",
    android_market: 'market://details?id=com.easytor.jacobs',
    ios_market: 'market://details?id=com.easytor.jacobs',
    youtubeLink: '',
    whatsappLink: 'https://wa.me/+972546783555',
    instagramLink: 'https://www.instagram.com/jacobsbarbershop1/',
    facebookLink: 'https://www.facebook.com/people/Jacobs-Barbershop/100063601835866/',
    tiktokLink: '',
    attentionMessage: '',
```

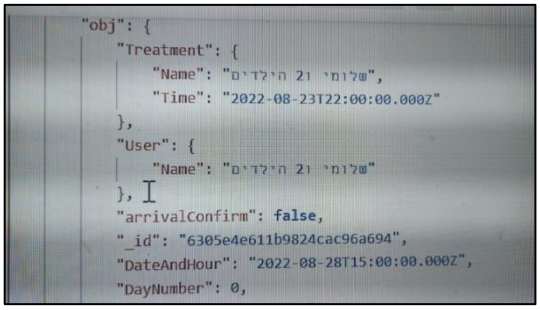
ה-header הזה קבוע לכל הלקוחות של האפליקציה והוא מכיל את השם של הספר ב-plain text! גם שאר השדות כמו appId ו-appSecret קבועים ומכילים מידע בנאלי, ולכן כל אחד יכול לשלוח את הבקשה הזאת. אבל רגע, כמו שאתם זוכרים מההקדמה שעשינו, לכל אפליקציה יש שני לקוחות, גם הלקוחות בעסק שמזמינים תורים, אבל גם... בעל העסק! אם אין שום דבר שמאמת את המשתמשים בבקשות לשרת, כנראה שאנחנו יכולים לשלוח בקשות גם בתור בעל העסק ואנחנו רק צריכים למצוא את ה-API הרלוונטי.



דוגמא לבקשה עם ה-uniqeabc header אז לאחר מעבר חוזר על הבקשות שמוגדרות בקוד, מצאתי בקשה שנקראת getFutureOrdersById:

```
MainServiceProvider.prototype.futureOrdersOfUser = function (userId) {
    console.log("futureOrdersOfUser  - " + userId);
    return this.http.post(this.request.url + '/getFutureOrdersById', {
        _id: userId
    }).map(this.extractData)
        .catch(this.catchError);
};
```

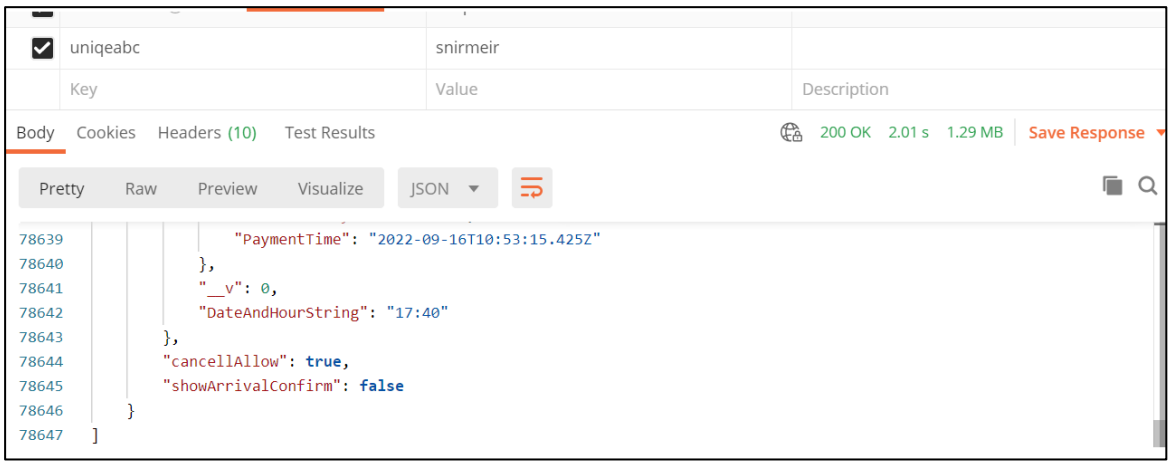
באפליקציה, את הטיפולים שהם ביקשו ואת הזמנים בהם הם יבואו. בעיקרון הבקשה מקבלת פרמטר בשם id אבל השרת לא מתייחס אליו בכלל ומחזיר תמיד את רשימת כל התורים של הספר. נוכל למשל לראות את טיפול ה-premium ב-100 שקלים שהזמין בן לקראת החתונה שלו, או את התור שהזמין שלומי לעצמו ולשני הילדים:



אבל זה לא הסוף, כמו שאנחנו יכולים לראות את רשימת התורים, אנחנו יכולים לעשות עוד הרבה דברים. אנחנו יכולים לבטל תורים, אנחנו יכולים לאשר תורים בשם אחרים, אנחנו יכולים להזיז תורים של אנשים, אנחנו יכולים להוסיף ולהוריד משתמשים מרשימת ההמתנה ובגדול לעשות את כל מה שבעל העסק יכול לעשות. רוצים תור דחוף לספר והכל תפוס? אתם כבר יודעים מה לעשות...

אז אומנם המספרה החביבה שלנו בבאר שבע היא לא מוקד עליה לרגל ואין בה הרבה תורים, אבל זוכרים את Snir&Meir מהמערכה הראשונה? אלה שמספרים את כל הסלבס והכדורגלנים? נשלח בקשה עם

הערך snirmeir (זה היה הניחוש השלישי שלי) בשדה Uniqeabc ונקבל חזרה json בגודל 1.3 מגה ו-80 אלף שורות של כל ההזמנות העתידיות שלהם:




אה, ומישהו אמר נבחרת ישראל בכדורגל?

נקווה שההגנה של הנבחרת במשחקים הקרובים תהיה יותר טובה מההגנה על ה-API של השרת הזה...

[שניר ומאיר עם שחקני הנבחרת בתספורת לפני משחק]

עדיין לא מספיק? רוצים לראות מה עוד אפשר לעשות? אז ככה. באפליקציה יש אפשרות לצ'אט בין הספר ללקוח למקרה שרוצים לתאם דברים נוספים, לשאול שאלות או סתם לדבר על החיים.

```
MainServiceProvider.prototype.getMessagesChat = function (user_id) {
    return this.http.post(this.request.url + '/getMessagesChat', {
        userId: user_id
    }).map(this.extractData)
        .catch(this.catchError);
};
```

רוצים לקבל את כל ההודעות מהשיחה הזו? מספיק להוסיף את ה-id של המשתמש וקיבלנו הכל. אבל איך נשיג את ה-id הזה? בקלות, זוכרים את ה-API לקבלת מידע על העובדים ותגובות / דירוגים שהשאירו עליהם משתמשים אחרים? אז יש שם עוד דבר אחד מעניין:


```
{
    "Comment": "",
    "_id": "5ecbe552ea18732560e67666",
    "Rating": 5,
    "User": {…
    },
    "Employye": "5e4c0968b5953d0dec2786c0",
    "SendOn": "2020-05-25T15:33:38.911Z",
    "__v": 0
},
```


בכל אובייקט של תגובה יש כמה שדות:

- התוכן עצמו של התגובה
- המזהה של התגובה
- הדירוג המספרי שהמשתמש נתן
- המשתמש ששלח את התגובה
- העובד שאותו דירגו
- התאריך של התגובה

אבל איזה מידע יש בתוך האובייקט של המשתמש ששלח את התגובה? כל מה שרק תחלמו עליו. יש לנו את השם המלא של המשתמש, מספר הטלפון שלו, מגדר, הביקור האחרון שלו, ההתחברות האחרונה שלו לאפליקציה, כל הגדרות הפרטיות שלו, תאריך הלידה שלו, תאריך ההרשמה, התמונה שלו, והכי חשוב, כל המזהים שלו באפליקציה! תמונות בעמוד הבא...

```
"User": {
    "RegistrationDate": "2020-05-25T15:32:17.717Z",
    "AccountClosed": false,
    "Img": "",
    "Cloudinary": "",
    "DateBirth": "1992-05-17T00:00:00.000Z",
    "Blocked": [],
    "NotComing": 0,
    "Status": 1,
    "Notify": true,
    "MultipleOrders": false,
    "LastVisit": "2020-07-07T15:08:43.773Z",
    "AllowRespond": true,
    "sendVoiceReminder": false,
    "Balance": 0,
    "_id": "[redacted]",
    "FName": "[redacted]",
    "LName": "[redacted]",
    "Phone": "[redacted]",
    "Gender": "זכר",
    "__v": 0
```

```
"OnesignalIds": [
    {
        "userId": "[redacted]",
        "pushToken": "[redacted]",
        "oneSignalAppId": "[redacted]"
    }
],
```


כן כן אתם רואים נכון, יש פה גם את ה-userId, אותו מזהה שחיפשנו מקודם בשביל לשלוף את כל ההודעות של המשתמש עם הספר.

## סיכום

זה הזמן גם להדגיש שהחולשות הללו דווחו למערך הסייבר הלאומי ששוב עזר מאוד בטיפול הבעיה ויצר קשר עם החברה שהספיקה מאז להוריד את המידע הפרטי על המשתמשים בכל הEndpoints השונים, ולחסום את האפשרות לבצע פעולות מול משתמשים תמימים.

אני אשמח שמי שהחזיק מעמד עד כאן יצא מכאן עם כמה מסקנות:

1. תחקרו כמה שיותר, אבל למטרות טובות - זה כיף, זה מלמד, וזה עוזר לשמור על האפליקציות שלנו בטוחות יותר.
2. אפשר למצוא דברים משמעותיים גם במקומות ה"ברורים מאליהם" שלכאורה אין סיכוי שנמצא בהם משהו משמעותי.
3. שוב, אל תניחו הנחות יסוד. אל תיפסלו כיוונים ואל תשללו דברים שאתם חושבים שבטוח מישהו אחר כבר חשב עליהם.
4. תאמינו בעצמכם, הכל פתיר ☺

ניתן ליצור עימי קשר בכתובת האימייל הבאה:

talsaadi36@gmail.com