**חיל האוויר**
**ביה"ס לפיקוד**
**ומנהיגות**
**תכנית תלפיות**
15 נובמבר 2020
כ"ח חשון תשפ"א

תכנית תלפיות

# הנדון: העברת מקל לתוכנת מעבדות חבלה

## תוכן עניינים

| | |
|---|---|
| **תיאור התוכנה בקצרה** | **1** |
| **הוראות למשתמש ומפרט תכולות** | **1** |
| **מדריך הרצה** | **7** |
| **תקלות בהרצה או בהרמת הדוקרים** | **7** |
| **מפרט טכני** | **8** |
| **רשימת פערים ומשימות להמשך פיתוח** | **8** |
| **עריכה בסיסית מאוד של התוכנה** | **8** |
| **פירוט קבצים חשובים בקוד המקור** | **12** |

## 1. *תיאור התוכנה בקצרה*

התוכנה נועדה לשימור וניהול ידע עבור חוקרי מעבדות החבלה במשטרת ישראל. המערכת מאפשרת להם לשמור את תיקי החקירות, ותיעוד של הראיות הנמצאות בזירות השונות, בצורה פשוטה וקלה לשימוש. ניתן להשתמש בתוכנה על מנת לבצע חיפושים חכמים בתיקי עבר ולמצוא אותם על בסיס המידע הנשמר בתוכנה. בנוסף לכך, התוכנה מאפשרת אוטומציה של יצירת קבצים נחוצים עבור החוקרים.

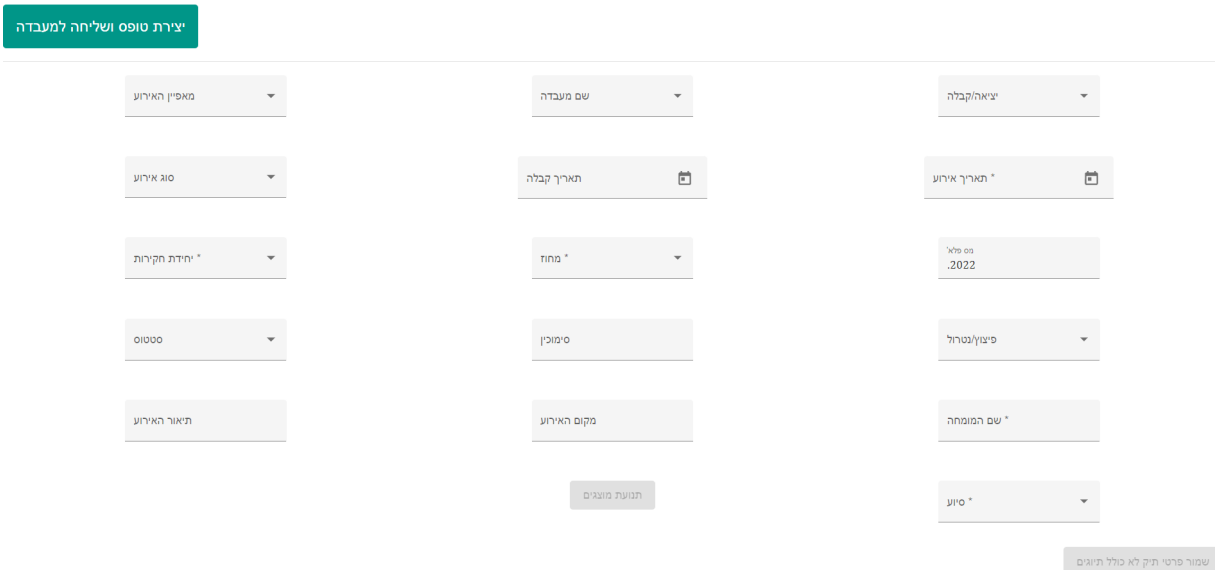
חלק זה של המדריך מיועד למשתמשי המוצר (חוקי מעבדות החבלה).

## 2. *הוראות למשתמש ומפרט תכולות*

2.1. מסך ראשי

מאפשר לגשת לכלל המסכים בתוכנה, ניתן להגיע ממנו לפתיחת תיק, הורדת קבצי Excel, סיכומים, ושאילתות.

## 2.2. פתיחת תיק



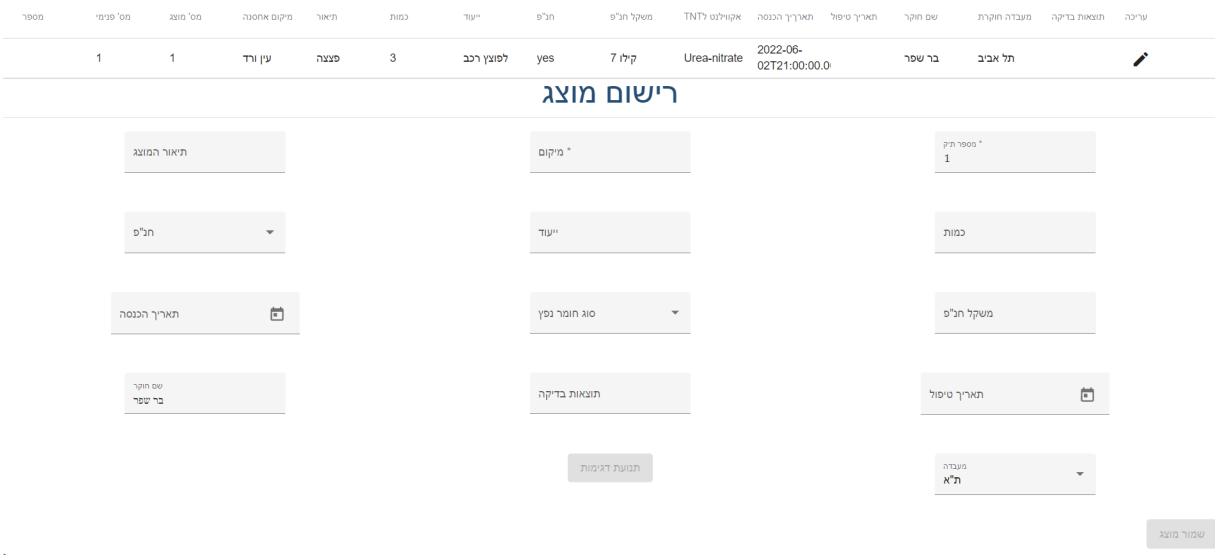

מאפשר לפתוח תיק חדש, לקבל מספר תיק דרך כפתור ייעודי בחלקו העליון של המסך. ניתן למלא פרטים אודות התיק, כשאר ישנם פרמטרי חובה (מסומנים בכוכבית) שרק לאחר מהמילוי שלהם יהיה ניתן לשמור את התיק. ממסך זה ניתן לגשת למסך רישום מוצגים דרך הכפתור "תנועת מוצגים", לאחר שמירת התיק.

בתחתית המסך מופיעה אפשרות להוסיף לתיק תיוגים, ולשמור אותם. יש לשים לב שישנו כפתור ייעודי לשמירת התיוגים, והכפתור של שמירת התיק בלבד לא שומר את התיוגים.

במידה ורוצים לשמור תיוגים, קודם כל יש למשור את התיק בכפתור העליון מבין השניים.

## 2.3. תנועת מוצגים

מסך הוספת מוצגים, בסיום ההוספה יש לשמור את המוצג דרך הכפתור, וניתן לעבור למסך הדגימות.
מוצגי שהוזנו בתיק יופיעו בחלקו העליון של המסך.



## 2.4. תנועת דגימות

| עריכה | מספר שקית | מספר טלפון | שם חוקר | סימוכין | שם יחידה | תאריך | הערות | תוצאות | אריזה | תאריך קבלה | תאריך שליחה | העבר למעבדה | מאיפה נדגם | מה נדגם | מס דגימה | מס מוצג | מס תיק |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 123456789 | 0500000000 | בר שפר | 2 | default | 2022-06-02T21:00:00. | default | אין טביעות אצבע | שקית שקופה | 2022-06-02T21:00:00. | 2022-06-02T21:00:00. | ט"א | הפצצה | חומר נפץ | 1 | 1 | 1 |

### רישום דגימה

* מספר תיק
1

מאיפה נדגם

תאריך קבלה

הערות

סימוכין

מספר שקית

* מספר מוצג
1

* מעבדת הבדיקה

אריזה

תאריך

שם חוקר
בר שפר

* מה נדגם

תאריך שליחה

תוצאות

* מעבדה שולחת
תל אביב

מספר טלפון

מסך רישום והצגת דגימות, בדומה למסך המוצגים. דגימות שהוזנו יופיעו בחלקו העליון של המסך.

## 2.5. הורדת תיקים ומוצגים

ממסך הבית, ניתן ללחוץ על כפתור "הורדת תיקים", או על כפתור "הורדת מוצגים", ולקבל קובץ Excel עם כלל התיקים או המוצגים שהוזנו לתוכנה.

## 2.6. סיכום חודשי ושנתי

ניתן להגיע אליהם דרך הכפתורים במסך הבית. מאפשרים קבלת סטטיסטיקות על החודש / השנה האחרונה. דרך הסיכום השנתי ניתן להגיע לכל אחד מהסיכומים החודשיים בשנה הנוכחית.

### סיכום שנתי

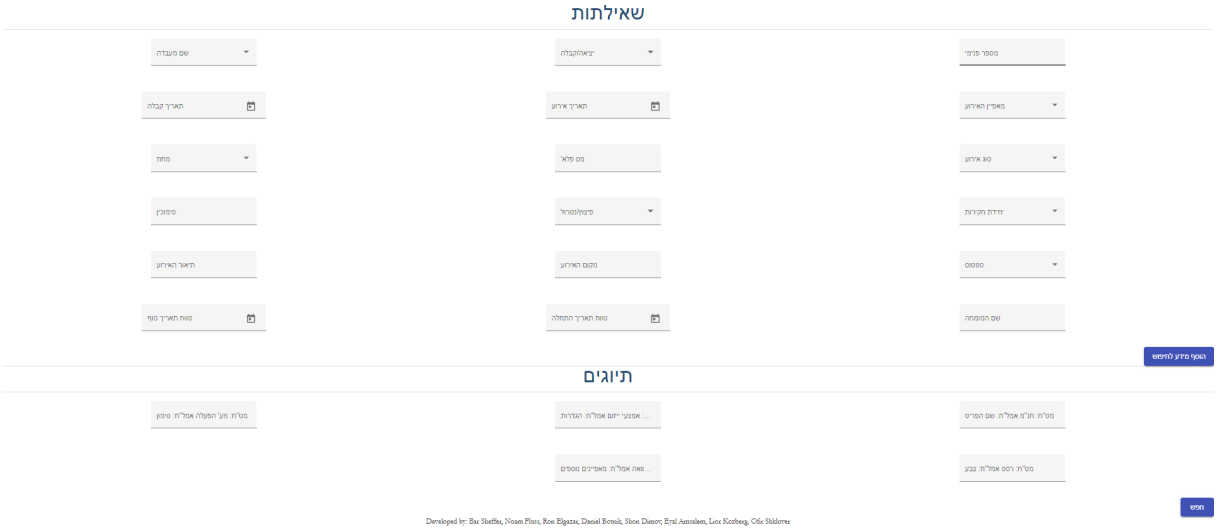
סה"כ תיקים פתוחים: 1
סה"כ תיקים בסטטוס פתוח השנה: 1
תיקים שנפתחו השנה: 0
תיקים שנסגרו השנה: 1
סה"כ אירועים ללא זירות: 0
סה"כ יציאה לזירות אירוע: 0
**פירוט אירועים עפ"י קטגוריות**
אמל"ח: 0
מטען חבלה: 1
זיקוקין: 1
בדיקות/שאילתות: 0

ינואר | פברואר | מרץ | אפריל | מאי | יוני
יולי | אוגוסט | ספטמבר | אוקטובר | נובמבר | דצמבר

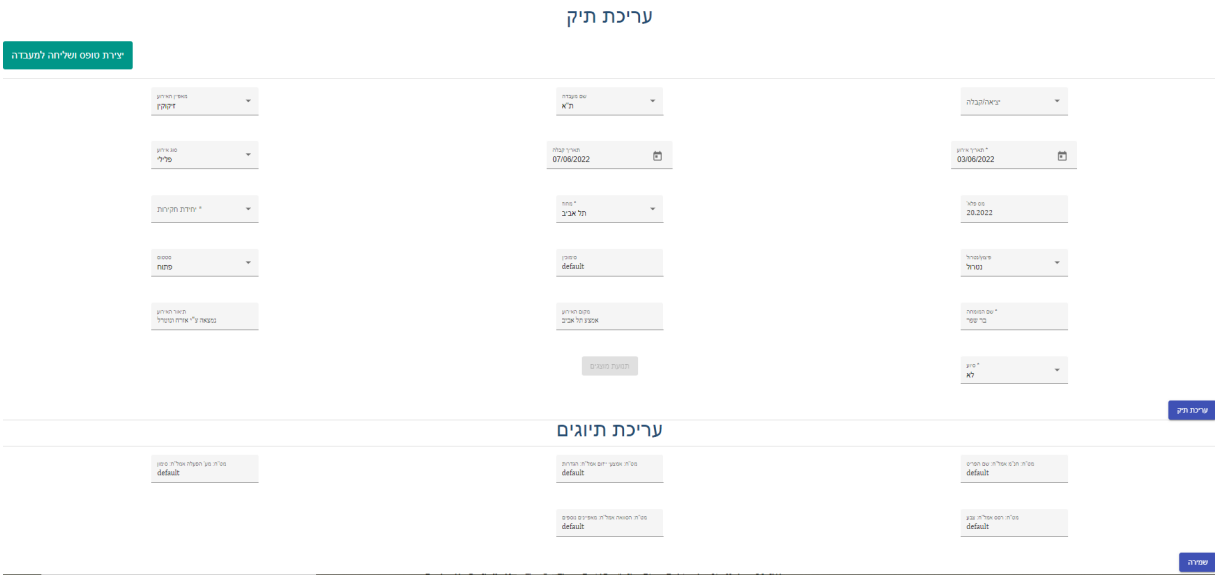
Developed by: Bar Sheffer, Noam Fluss, Ron Elgazar, Daniel Botnik, Shon Dimov, Eyal Amsalem, Lior Kozberg, Ofir Shklover

2.7. שאילתות

דרך מסך הבית ניתן להגיע למסך השאילתות. מסך זה מאפשר חיפוש תיקים שהוזנו למערכת, ע"פ פרמטרים קבועים, או ע"פ התיוגים של התיק שנעשו בעת יצירתו / עריכתו. אם נרצה לחפש ע"פ פרמטרים קבועים, נמלא אותם בחלקו העליון של המסך, ונלחץ על כפתור "הוסף מידע לחיפוש". לאחר מכן ניתן להוסיף תיוגים לחיפוש (ניתן גם לחפש רק ע"פ תיוגים). בסופו של דבר נלחץ על כפתור החיפוש.
מציאת כלל התיקים תעשה על ידי חיפוש ישיר ללא הזנת פרמטרים.




2.8. עריכת תיקים / מוצגים / דגימות




כלל מסכי העריכה דומים מאוד. ניתן לגשת לאובייקט מסוים (תיק / מוצג / דגימה), לאחר העריכה (אין חובה לערוך, במידה ורק רוצים לחפש בתיק מידע), ניתן ללחוץ על כפתור העריכה, ומכאן תיפתח האפשרות להגיע למסך תנועת המוצגים / הדגימות. חשוב: חייבים ללחוץ על כפתור השמירה בתחתית המסך כדי לשמור את השינוים, לא מפסיק ללחוץ על כפתור עריכת התיק.

2.9. יצירת טופס שליחה למעבדה

מופיע במסך יצירת התיק ובמסך עריכת התיק. מבצע הורדה של טופס העברת חומר למעבדה, ע"פ הפרטים שהוזנו לתוכנה.

יצירת טופס ושליחה למעבדה

משטרת ישראל

מעבדת חבלה

מעבדת חבלה צפון
תאריך: 2022 / 06 / 02
טלפון: 000000
פנימי מעבדה: 2

## טופס העברת חומר

אל: ט"א

| דחיפות | סיכונים | מוצגים |
|---|---|---|
| ☒ רגיל | ☐ סיכון ביולוגי | ☒ רגיל |
| ☐ דחוף | ☐ חומר רעיל | ☐ נוסף |
| ☐ דחוף מעצר | ☒ מוצגים חדים | ☐ חוזר |

יחידה: TEST נוסף: מידע

תיאור המוצג/ים:

1. מסמרים וברגים ממוצג מס' 1 בדוח התפיסה הוכנסו לשקית צלף שסומנה "שקית שקופה" והוכנסה לשקית מאובטחת לשימוש חד פעמי שמספרה 1 וסומנה "-2022-06 21T21:00:00.000Z" מע' חבלה דרום מס' 10.2022 ר.ז כ.פ
2. גוף הפצצה ממוצג מס' 1 בדוח התפיסה הוכנסו לשקית צלף שסומנה "שקית שקופה" והוכנסה לשקית מאובטחת לשימוש חד פעמי שמספרה 1 וסומנה "-2022-06 19T21:00:00.000Z" מע' חבלה דרום מס' 10.2022 ר.ז כ.פ

תיאור האירוע:
טקסט

מהות הבדיקה

הערות:
הערה לדוגמא

פרטי השולח: שם: נועם פלוס דרגה: סמל מס' אישי: 000000000 חתימה: _____

**אישור קבלה**

תאריך _____

שם המקבל _____ תפקיד _____ דרגה _____ חתימה _____

2.10. הגבלות
ישנה הגבלה משמעותי אחת על השימוש בתוכנה, אסור ללחוץ על כפתור החזרה אחורה בדפדפן ממסך הבית, הדבר גורם לשגיאות בתוכנה. בכל דף אחר בתוכנה, הדבר אפשרי ללא שגיאות.

מנקודה זו והלאה המדריך מיועד למפתחים.

## 3. *מדריך הרצה*

קישור לתיקיית גיטהאב

קישור לדוקרים בדוקרהאב: frontend, backend

3.1. מדריך לקימפול הקוד והרצה מקומית - בגיטהאב, ב-main, בקובץ run_the_app_instructions.txt.

3.2. מדריך להרמת הדוקרים - בגיטהאב, ב-main, בקובץ create_dockers.txt.

## 4. *תקלות בהרצה או בהרמת הדוקרים*

4.1. backend לא עולה

4.1.1. השגיאה היא שכתובת TCP/IP לא נמצאה, ייתכן וכתובת ה-db לא נכונה, וודאו שכל משתני הסביבה בקובץ env_variables נכונים עבור ה-db שלכם. (פתרון אפשרי נוסף הוא לבדוק את נכונות המשתנים בקובץ settings.py שנמצא בSUPER_API, תחת המשתנה DATABASES).

4.1.2. אם השגיאה היא בהתקנת ספריות (אפשרי בהרצה ללא דוקרים בלבד), יש לוודא שכל הספריות המופיעות בקובץ requirements.txt קיימות.

4.1.3. יש לוודא שלפני פקודת runserver הרצנו את פקודות המיגרציה שממלאות את הdb

Python manage.py makemigrations

Python manage.py migrate

4.2. frontend לא מגיב לשימוש אבל עולה

4.2.1. יש לוודא תקשורת בין הdb לbackend לfrontend. ניתן לעשות זאת בקלות בעזרת משחק קצר עם התוכנה, ובדיקה של הלוגים המתקבלים מהדוקר. לדוגמה, נצפה לראות לוגים frontend כאשר נרפרש את התוכנה, ונצפה לראות לוגים מ backend כאשר נמלא את כל

השדות המתאימים וננסה ליצור קובץ word בצורה אוטמטית. כמו כן תמיד אפשר לבדוק את התעבורה שעוברת דרך frontend בעזרת מקש f12, בתפריט network.

4.2.2. יש לוודא שה-API_URL של backend מוגדר נכונה בקובץ docker-compose.yml.

## 5. *מפרט טכני*

5.1. Database - PostgreSQL

5.2. Backend - Python Django 3.2

5.2.1. הספריות וגרסותיהן מפרטות בקובץ requirements.txt.

5.3. Frontend - Angular

## 6. *רשימת פערים ומשימות להמשך פיתוח*

תמיכה במניעת עריכה בו זמנית

סיכום חודשי ושנתי - פר מעבדה

הגדרת פרמטרים דיפולטיבים לכל שדה

## 7. *עריכה בסיסית מאוד של התוכנה*

7.1. עריכת שמות שדות -

ב-**services** יש פונקציות ליצירת השדות למסכים השונים. כדי לערוך שדה ספציפי ניתן לגשת ל-**service** המתאים ולשנות את התאים במערך שמיוצר שם. דוגמא עם **cases.service.ts**:

```
getQuestions() {
  const questions: FormFieldBase<string>[] = [
    new DropdownField({
      key: 'received_or_go',
      label: 'יציאה/קבלה',
      required: true,
      options: [
        { key: 'יציאה לאירוע', value: 'יציאה לאירוע' },
        { key: 'קבלת אירוע', value: 'קבלת אירוע' },
      ],
    }),
    new DropdownField({
      key: 'lab_name',
      label: 'שם מעבדה ',
      required: true,
      options: [
        { key: 'דרום', value: 'דרום' },
        { key: 'תל אביב', value: 'ת"א' },
        { key: 'צפון', value: 'צפון' },
        { key: 'מטא"ר', value: 'מטא"ר' },
      ],
    }),
    new DropdownField({
```

ניתן לשנות את שם השדה של "שם מעבדה" ל"מעבדה" איפה ש-**label** נמצא.

7.2. הוספת שדות -

כל service מייצר מערך שדות ומחזיר אותו למסך שמשתמש בו. ניתן להוסיף תא נוסף למערך עם ערכים מתאימים. ניתן לייצר **DatePickerField**, **TextboxField**, **DropdownField** וכו.

```
new DropdownField({
  key: 'event_characteristic',
  label: 'מאפיין האירוע',
  required: true,
  options: [
    { key: 'weapons', value: 'אמל"ח' },
    { key: 'explosive_device', value: 'מטען חבלה' },
    { key: 'fireworks', value: 'זיקוקין' },
    { key: 'query', value: 'בדיקות/שאילתה' },
  ],
}),
new DatePickerField({
  key: 'event_date',
  label: 'תאריך אירוע',
  type: 'text',
  required: true,
}),
new TextboxField({
  key: 'pele_number',
  required: false,
  label: "'מס" + ' פלא',
  type: 'text',
  value: '' + '.' + this.getFullYear().toString(),
```

שם השדה (**key**) שנשלח ל-**backend**

שם השדה המוצג במסך (**label**)

סוג הנתון (**type**) יכול להיות מחרוזת, מספר וכו.

האם חובה למלא (**required**)

ערך דיפולטי (**value**)

ל-**DropdownField** אפשרויות שונות (**options**) כמוצג בתמונה:

**key** - שם שנשלח ל-**backend**.

**value** - הערך שמולא.

7.3. מחיקת שדה -

ניתן למחוק שדה על ידי הורדת התא שלו מן המערך.

7.4. שינוי סיכום חודשי/שנתי -

כדי לשנות את התצוגה של שמות השדות בסיכום חודשי/שנתי ניתן לעצב את קבצי ה-**html**:

**yearly-summary-screen.component.html**

**monthly-summary-screen.component.html**

כל מה שקשור בערכים תלוי ב-**backend**, המידע חוזר בתור רשימה.

7.5. עריכת ערך דיפולטיבי -

במסכים מסוימים ערכים מתמלאים בצורה דיפולטיבית כאשר המסך עולה. המידע מוצא מה-localStorage שנשמר במסך קודם.

במידה ויש צורך לשנות את הערך הדיפולטיבי צריך לשנות אותו במסך הקודם. דוגמא עם **register-exhibit-screen.component.ts**, **open-case-screen.component.ts**

```
formRawValue.internal_number = this.internal_number;
localStorage.setItem('case', JSON.stringify(formRawValue));
```

המידע של השדות נשמר כ-"case" ב-local storage. ניתן לראות שיש שינוי של אחד השדות (מספר פנימי של תיק) בתור דוגמא לשינויים עתידיים.

```
// Gets case's data from local storage
const localCase = JSON.parse(localStorage.getItem('case') || '[]');
// Convert all fields into array and auto fills some fields with values from case.
var values = Array.from(this.fields$.values());
values[0]['value'] = localCase.internal_number;
values[11]['value'] = localCase.sender_name;
values[12]['value'] = localCase.lab_name;
```

ב-constructor יש קריאה של המידע מה-local storage והצבת נתונים באופן דיפולטיבי, מפני שהשדות מסודרות במערך ניתן לגשת לכל שדה בעזרת **index** וגישה ל-**value** שלו כמו שמוצג בדוגמא. ***מומלץ להשתמש בקבועים למקומות בעלי שמות משמעותיים כדי לא להתבלבל במיקום של כל שדה.***

7.6. הוספת אופציה לשדה רב ברירתי - במידה ורוצים לשנות או להוסיף שדה מסוים בתוכנה (נגיד להוסיף מעבדה לרשימת המעבדות), ניתן לעשות זאת דרך תיקיית src\frontend -> src -> app -> core -> services.

שמות השדות המופיעים בתוכנה, מופיעים בכל הקוד בשמות המשתנים המרוכזים בקובץ הבא.

השדות המופיעים שם מרכיבים את כלל הפיצ'רים בתוכנה:

תיקים

מוצגים

דגימות

סיכום חודשי

סיכום שנתי

7.7 עריכת בסיסית של התוכנה בצד הbackend

כאשר רוצים לערוך שם של שדה כלשהוא בצד השרת יש להוסיף\לשנות את שמו באנגלית תחת השם של המחלקה שלו(דגימה,תיק וכו'....) בקובץ serializers.py:

לדוג' להוסיף\לשנות שדה בתיק:

```
class CaseSerializer(serializers.ModelSerializer):
    class Meta:
        model = Case
        fields = (
            "internal_number",
            "received_or_go",
            "lab_name",
            "event_characteristic",
            "event_date",
            "received_date",
            "event_type",
            "pele_number",
            "district",
            "investigating_unit",
            "explosion_or_disarm",
            "reference_number",
            "status",
            "sender_name",
            "event_location",
            "event_description",
            "weapon_name",
            "explosive_device_material",
            "explosive_device_means",
            "weapon_options",
            "explosive_device_operating_system",
            "weapon_mark",
            "explosive_device_spray",
            "weapon_color",
            "explosive_device_camouflage",
            "weapon_additional_characteristics")
```

בנוסף יש להתאים(לכתוב את אותו השם בדיוק) בקובץ models.py

לדוג' כאשר רוצים לשנות\להוסיף שדה בתיק:

*שמים את אותו השם **בדיוק** ששמנו בקובץ serializres.py ושמים את האורך המקסימלי של הstring

בצורה הבאה:

```
class Case(models.Model):
    # class Meta:
        # get_latest_by = 'internal_number'
    internal_number = models.CharField(max_length=100, primary_key=True)
    received_or_go = models.CharField(max_length=100)
    lab_name = models.CharField(max_length=256)
    event_characteristic = models.CharField(max_length=100)
    event_date = models.CharField(max_length=100)
    received_date = models.CharField(max_length=100)
    event_type = models.CharField(max_length=100)
    pele_number = models.CharField(max_length=100)
    district = models.CharField(max_length=100)
    investigating_unit = models.CharField(max_length=100)
    explosion_or_disarm = models.CharField(max_length=100)
    reference_number = models.CharField(max_length=100)
    status = models.CharField(max_length=100)
    event_location = models.CharField(max_length=100)
    event_description = models.CharField(max_length=256)
    sender_name = models.CharField(max_length=50)

    weapon_name = models.CharField(max_length=256)
    explosive_device_material = models.CharField(max_length=256)
    explosive_device_means = models.CharField(max_length=256)
    weapon_options = models.CharField(max_length=256)
    explosive_device_operating_system = models.CharField(max_length=256)
    weapon_mark = models.CharField(max_length=256)
    explosive_device_spray = models.CharField(max_length=256)
    weapon_color = models.CharField(max_length=256)
    explosive_device_camouflage = models.CharField(max_length=256)
    weapon_additional_characteristics = models.CharField(max_length=256)
```

**חשוב לציין שאין ממש חשיבות לשמה של התיבה שכותבים בצד השרת שכן היא לא מופיעה בUI הסופי, אך חייב שיהיה את אותו השם בשני הקבצים serializers.py ו models.py

## 8. *פירוט קבצים חשובים בקוד המקור*

### 8.1 מודול frontend

**app**

**app.module.ts**

המודול הראשי, מכיל את הפרויקט ואת כל התכונות שלו (**components**).

**app.component.ts**

תכונה ראשית, משמשת להראות מסכים אחרים עליה.

**app.component.html**

מציג מסכים אחרים בתכנה בעזרת **<router-outlet>** שממלא בצורה דינמית את המסך הנכון לפי הדרישה.

**app-routing.module.ts**

מקשר את כל הניתובים למסכים השונים בתוכנה.

## dynamic components

### dynamic-form

תכונה דינמית לתצוגה של מסך מילוי שדות במסכים שונים, מקבלת פרמטר **[fields]** המכיל את השדות. שדות מוכנים על ידי **services** שונים. קובץ ה-html רץ על רשימת השדות (fields) ומציג אותם על המסך. פרמטרים נוספים הם **[buttonText]** ו-**[onSubmit]**, כפתור שקורא לפונקציה **onSubmit** ושולח לה form המכיל את כל השדות במסך (ממולאים או ריקים).

### dynamic-form-field

תכונה דינמית לייצוג שדה יחיד בתכונה **dynamic-form**, מקבלת פרמטר [field], נוצר על ידי **dynamic-for** בריצה על התכונה fields.

## services

### summary.service.ts

שולח קריאת API ל-backend על מנת לקבל סיכום חודשי/שנתי של אירועים.

### cases.service.ts

אחראי על יצירת שדות מילוי בשביל התכונה **dynamic-form** במסכי פתיחת ועריכת תיקים (**open-case-screen**, **edit-case-screen**), בנוסף מכיל קריאות API ל-backend בנוגע להוספה/עריכה/מחיקה של תיקים.

### exhibits.service.ts

אחראי על יצירת שדות מילוי בשביל התכונה **dynamic-form** במסכי יצירת ועריכת מוצגים (**register-exhibit-screen**, **edit-exhibit-screen**), בנוסף מכיל קריאות API ל-backend בנוגע להוספה/עריכה/מחיקה של מוצגים.

### samples.service.ts

אחראי על יצירת שדות מילוי בשביל התכונה **dynamic-form** במסכי יצירת ועריכת דגימות (**samples-screen**, **edit-samples-screen**), בנוסף מכיל קריאות API ל-backend בנוגע להוספה/עריכה/מחיקה של דגימות.

### lab-form.service.ts

אחראי על יצירת שדות מילוי בשביל התכונה **dynamic-form** במסך יצירת קובץ ושליחה למעבדה (**gen-lab-form-screen**).

### generate-docx.service.ts

מבצע קריאת API ל-backend ומוריד את קובץ ה-docx שנשלח בחזרה.

### download-exhibits.service.ts

מבצע קריאת API ל-backend ומוריד קובץ csv שנשלח בחזרה המכיל מידע על המוצגים.

## constants

### constants.ts

שומר את קריאת הAPI ל-backend:

בגרסה לוקלית:

"**http://localhost:12580**"

בגרסה שמופיעה בגיט:

המשתנה הוא משתנה סביבה שמוגדר דרך ה-docker-compose.

## components

### main-screen

המסך הראשי, ב-html יש ניתוב לכל מסך אחר בתוכנה:

- פתיחת אירוע חדש (**open-case-screen**)
- עדכון אירוע קיים (**edit-case-screen**)
- סיכום חודשי (**monthly-summary-screen**)
- סיכום שנתי (**yearly-summary-screen**)
- שאילתות (**search-case-screen**)
- הורדת מוצגים (מוריד מוצגים - **download-exhibits.service.ts**)

### edit-case-screen | open-case-screen

מסכי פתיחת ועריכת תיקים, מציגים ב-html שדות מילוי בעזרת **dynamic-form** ו-**cases.service.ts**.

### edit-exhibit-screen | register-exhibit-screen

מסכי יצירת ועריכת מוצגים בתיק, מציגים ב-html שדות מילוי בעזרת **dynamic-form** ו-**exhibits.service.ts**.

### edit-samples-screen | samples-screen

מסכי יצירת ועריכת דגימות במוצג, מציגים ב-html שדות מילוי בעזרת **dynamic-form** ו-**samples.service.ts**.

### gen-lab-form-screen

מסך ליצירת קובץ ושליחה למעבדה, מציגים ב-html שדות מילוי בעזרת **dynamic-form** ו-**generate-docx.service.ts**.

### monthly-summary-screen | yearly-summary-screen

מסכי סיכום חודשי/שנתי, מציגים סיכום לגבי אירועים בטווחי זמן מתבקשים, המידע מושג בעזרת **summary.service.ts**

**מילוי אוטומטי של שדות**

במסך פתיחת תיק (**open-case-screen**) ישנה אפשרות לשמירה הנתונים על התיק, נתונים אלה ישמרו ב-localStorage כשהמפתח תחת השם "case" והערך הוא מחרוזת במבנה json בעלת כל הנתונים על התיק.

במסך פתיחת מוצג (**register-exhibit-screen**) ישנה קריאה מה-localStorage על מנת לקבל את נתוני התיק ולמלא אותם באופן אוטומטי בשדות. בנוסף למילוי שדות ישנה אפשרות לשמור מוצג ב-localStorage גם כן, נשמר תחת שם המפתח "exhibit" כשהערך הוא מחרוזת במבנה json בעלת כל הנתונים על המוצג.

במסך יצירת דגימה (**samples-screen**) ישנה קריאה מה-localStorage על מנת לקבל את נתוני המוצג ולמלא אותן באופן אוטומטי בשדות.

**מילוי אוטומטי של שדות בלחיצת כפתור**

ניתן לדחוף ל-local storage מידע ואז לבצע rerendering לעמוד על מנת לקרוא אותו בבניה החדשה של העמוד ובכך למלא את השדה לאחר לחיצת כפתור.

8.2. מודול backend

**app.py**

מכיל את הקונפיגורציה הבסיסית של צד השרת.

**create_deafult_values**

מכיל פונקציה אחת שמקבלת מילון(עם מידע) וסריאלייזר(ראה תחת הכותרת serializers.py מה הוא מכיל) ובהתאם למידע מעדכנת את הערכים שדרושים לתיקון("שאינם תקינים") לערכי ברירת מחדל.

**models.py**

בקובץ זה כתובים כל המודלים וערכיהם, כאשר כל מודל הוא מחלקה בפני עצמה, וכל מודל כזה מכיל מספר ערכים בהתאם למודל(לדוג': המודל CASE כלומר תיק, מכיל מספר רב של ערכים כמו: שם המעבדה, תאריך האירוע וכו'...). כאשר בגדול ישנם שלושה מודלים: תיק, מיצג ודגימה ולכל אחד ערכים משלו כפי שצויין בדוגמה.

**serializers.py**

בקובץ נמצאים **אך ורק שמות** הערכים של כל מודל.

***חשוב לציין שחייב להיות תיאום מלא בין קובץ זה לקובץ models.py, כלומר שיהיו אותם שמות לערכים(אחרת עלולים להיווצר מספר בעיות).

**tests.py**

קובץ ריק שנועד לבדיקות QA.

**urls.py**

קובץ זה הוא למעשה הקישור בין הפונקציות שנכתבות בצד השרת, לצד הלקוח. הURL הוא למעשה הכתובת של פונקציה בצד השרת באמצעות הURL יש לצד הלקוח גישה לפונקציות בצד השרת, כאשר לכל URL יש 2 שדות, הראשון מכיל את שם הכתובת והוא חייב להתחיל ב"r^"(מסוג str), והשדה שני מכיל את מיקום הפונקציה בצד השרת בצורה הבאה:file_name.function_name (במילים אחרות מיקום הפונקציה ושמה. לדוג': views.idApi (כל הפונקציות העיקריות נכתבות בקובץ views שאפרט עליו בכותרת הבאה).

דוגמה:

```
url(r'^exhibits/query/$', views.exhibitQuery),
    url(r'^exhibits/dwnld/$', views.exhibitDwnld),
```

**views.py**

בקובץ זה נמצאות כל הפונקציות העיקריות שבשורה התחתונה מחזירות ערכים שצד הלקוח דורש, בנוסף נמצאות שם גם פונקציות עזר שמבצעות אלגוריתמי חיפוש ועריכת משתנים לטובת הפונקציות העיקריות.

### 8.3. ג'ינרוט קבצי office

**docx_generator.py**

בקובץ זה נמצאת הפונקציה generate_docx אשר מייצרת זרם ביטים של קובץ וורד לפי נתונים שנשלחים אלייה ומחזירה אותו.

**template.docx**

קובץ זה הוא התבנית שבעצרת נוצר קובץ העבר חומר. כדי לערוך אותו או לייצר תבנית חדשה יש להחליף את קובץ זה בקובץ טמפלייט החדש ולשנות בקוד הגנרוט כך שמיקום הפסקאות והשורות יהיה תואם לקובץ החדש.

לדוגמא בשביל טסקסט:

```
doc.tables[0].rows[1].cells[0].paragraphs[6].runs[0].text =
```

ובשביל תייבת סימון:

```
doc.tables[0].rows[0].cells[1].paragraphs[1].runs[0]._r.xpath(CHECKBOX_PATH)[0].insert(2,
                                                                                OxmlElement('w:checked'))
```

בנוסף לכך בתנאי שמחליפים את כל קובץ הטמפלייט לקובץ חדש אז נצטרך גם לשנות את הxpath ששמורה במשתנה `CHECKBOX_PATH` בראש הקובץ לxpath של הקובץ החדש.

**views.py**

בקובץ זה נמצאת הפונקציה exhibitDwnld אשר אחראית על החזרת קובץ אקסל עם כל המוצגים במסד הנתונים. היא נעזרת בפונקציה writeToExcelExb שבתוכה נמצא הקוד אשר אחראי על יצרית קובץ האקסל והכנסת הנתונים אשר מועברים לפונקציה לתוכו.

בהתאם להגיון זה עובדת גם הפונקציה caseDwnld אשר משתמשת בwriteToExcelCase

**9.** אנשי קשר:

בר שפר - 058-5172002

נועם פלוס - 055-2239131

**10.** בברכה,

**בר שפר**, ראש צוות

**נועם פלוס**, ראש צוות

**רון אלגזר**, מודול frontend

**שון דימוב**, מודול frontend

**דניאל בוטניק**, מודול frontend

**אופיר שקלובר**, מודול backend

**ליאור קוזברג**, מודול backend

**אייל אמסלם**, מודול backend