

## הוספת ספק

המערכת בנויה כך שלכל ספק **שאינו** ספק פרטי יש כמה פונקציות וקבצים הנפרדים משאר הספקים זאת על מנת לאפשר גמישות בשינויים בניהול המידע של אותו ספק, כמו למשל שינוי סוג או מבנה קובץ המתאר את החיובים של הרשות מול אותו ספק או שינוי המתבקש על ידי רשות בצורת הצגה של המידע ועוד.

מצב זה אכן יוצר שכפול קוד, אך מאפשר למערכת להיות גמישה ונוטה לשינויים במידת הצורך.

## חשוב לציין:

ספקים שאינם פרטיים מתארים ספקים שהינם נותנים שירות לכל הרשויות בארץ, לרוב מדובר בשירותי תשתיות. הספק שעל בסיסו נוצרה המערכת הינו 'בזק', המשקף בצורה מעולה את הרעיון הנ"ל

## צד-שרת:

### קריאת קובץ:

1. הוספת ספק בקובץ Enums הנמצא בפרוייקט Entites , הספק צריך להתווסף תחת public enum Suppliers, כאשר ה.פ של אותו ספק משמש כערך המספרי של שם הספק
2. הוספת קובץ שמטרתו היחידה היא קריאת המידע מהקובץ, זאת נעשה בצורה הזו משום שבין הספקים אין רק הבדל של סוג קובץ והסדר בתוכו, אלא גם בסוג המידע שהוא מספק. שם הקובץ צריך להיות בנוי בצורה הבאה: **SupplierNameFileReaderService**, כאשר הגישה אליו נעשית דרך interface בלבד! לאחר הוספת הקובץ וחיבורו לממשק יש ללכת לקובץ startup.cs היושב תחת פרוייקט ה ManagingDebts, ולהוסיף תחת services.AddScoped, את הקבצים החדשים שנוצרו לאחר מכן יש למלא את הקובץ לפי לוגיקת הקריאה של הקובץ ופיזור הנתונים בטבלה המתאימה (ראה חלק בסיס-מידע לפירוט בנושא)
3. תחת תיקיית controllers אשר יושבת בפרוייקט ManagingDebts, יש לגשת לקובץ UploadFileController, בקובץ הנ"ל יש להוסיף את הממשק מסעיף 2 בתחילת הדף ולהוסיפו לconstructor של controllern, לאחר מכן תחת פונקציית uploadFile, קיים switch כאשר כל case בswitch מייצג קריאה לממשק של הספק המקושר אליו, יש להוסיף שם case חדש תוך כדי שימוש בenum מסעיף 1

### וידוי נתונים:

1. הוספת קובץ יישות תחת Entities הממש את כל השדות מבסיס המידע, מומלץ לבצע העתק הדבק מהקובץ של היישות תחת תיקיית models הנמצאת בפרוייקט dataModel, יש לוודא שקיים שדה AmountAfterTax, שבהתאם לשמו מייצגת את הסכום לאחר חישוב מס
2. הוספת קובץ שמטרתו היא קבלת נתונים מהבסיס המידע והטיפול בהם, לשם כך יש להוסיף קובץ בעל המבנה השמי **SupplierNameDataConformtion**, וממשק התואם לשם הקובץ.
3. בקובץ יש להוסיף את הקריאה מבסיס הנתונים ואת הפונקציות המכינות את הdictionary הנשלח לצד-לקוח לצורך פענוח בעמוד ווידוי נתונים באפליקציה, dictionary בנוי בצורה הבאה:

- מפתח שהינו json string של סיכום החיובים תחת חוזה מסויים, ניתן לראות דוגמאות בקבצי בזק וחשמל, וערך שהינו מייצג מראה של הנתונים מהקובץ כמו שהם.
4. בקובץ יש להוסיף פונקציית match, שמטרתה לקבל מערך של נתוני קובץ ולעדכן את המצב שלהם, ניתן לראות דוגמאות בקבצי בזק וחשמל
  5. לאחר הוספת הקובץ וחיבורו לממשק יש ללכת לקובץ startup.cs היושב תחת פרוייקט ה ManagingDebts, ולהוסיף תחת services.AddScoped, את הקבצים החדשים שנוצרו
  6. תחת תקיית controllers הנמצאת בפרוייקט ManagingDebts יש לגשת לקובץ DataConfirmationController ולהוסיף בחלק העליון של הדף את הקריאה לממשק שכרגע יצרנו, לא לשכוח להוסיף קריאה לממשק בconstructor.
  7. יש לממש את הפונקציות בswitch הנמצא בפונקציית GetDataBySummary, ולהוסיף פונקציה חדשה בשם, MatchSupplierName, הקוראת לפונקציית ההתאמה שיצרנו בקובץ (הכל דרך הממשקים!)  
דוחות:
  1. הוספת case בשני הswitch הקיימים בקובץ dashboardServie
  2. הוספת פונקציה אחת שמטרה היא הבאת כל המידע הקיים במערכת לקבץ אותו לחוזים וליצור יישות הממופה לטבלה chartEntity, ניתן לראות דוגמאות איך ניתן לעשות זאת בשורה אחת של קוד על פי פונקציות קודמות

#### יצירת פקודת יומן:

1. יש להוסיף פונקצייה אחת תחת sliService, בשם createPkudatYomanSupplierName, הפונקצייה אמורה לקבץ את המידע מהבסיס מידע על בסיס מספר חשבונים, ובכך ליצור סוג של dictionary כאשר מספר החשבונות הוא המפתח לאחר מכן יש לעבור על כל מידע המקובץ על ידי חשבונות ולקבץ אותו שוב על סמך מספר חוזה לאחר מכן יש לוודא את הקשר הלוגי בין החוזה לסעיף התקציבי וליצור/לסכום את את פרטי החוזה לתוך השורת פקודה של הסעיף התקציבי, הסבר יותר מפורט של הלוגיקה בנספח א' לבסוף יש להמיר הכל למערך ולהעביר חזרה לפקודת היומן.
- חשוב לציין:** לא מומלץ לשנות יותר מידי, מרבית השוני בין הפונקציות של יצירת תוכן הפקודה הוא השליפה של המידע מבסיס המידע וצורת הקיבוץ שלו.
2. לאחר יצירת הפונקציה יש להוסיף case בswitch לפי enum הקיים במערכת.

#### צד-לקוח:

##### קריאת קובץ:

1. יש להוסיף לקובץ billing-summary.service.ts, פונקציה שמטרתה להחזיר מערך של TableHeader, שם הפונקציה צריך להיות getSupplierNameTableHeaders, יש לממש זאת על פי השדות הסיכום הנמצאות בשימוש הרב ביותר לפי מבנה הקובץ לדוגמא בקובץ בזק יש שימוש בכל העמודות הקיימות בטבלת GeneralBillingSummary, אך לא תחת חברת חשמל

ולאחר הוספת הפונקציה לממש קריאה של על בסיס הswitch הקיים באותו קובץ

##### וידוי נתונים:

1. יש להוסיף לקובץ bank-matching.service.ts, פונקציה שמטרתה להחזיר מערך של TableHeader, שם הפונקציה צריך להיות getSupplierNameTableHeaders, יש לממש זאת על פי השדות הקיימים בטבלה המייצגת את המידע מהקובץ

2. באותו קובץ יש להוסיף לswitch תחת פונקציית match ערך url בהתאם לפונקציה בצד-שרת  
matchSupplierName

כמובן לא לשכוח ליצור עבור הרשות/רשויות הרלוונטיות את הספק , פעולה זאת נמצאת בתפריט תחת  
"מערכת" -> "ספקים"

בסיס נתונים:

יצירת טבלה לספק:

הטבלה הנמצאת בבסיס המידע אמורה לשמש כמו מראה לשדות הקובץ של הספק, **לא זורקים מידע**  
מעבר לכך יש להוסיף את השדות הבאים, נא לדייק בשם

**Customer\_id,general\_row\_id,row\_id,is\_matched,journal\_entry\_number**

השילוב של השדות המודגשים צריכים להיות המפתח הראשי בטבלה,  
כאשר קיים קשר של אחד לרבים בין general\_billing\_summary.row\_id לטבלה שעכשיו יצרנו והשדה  
general\_row\_id

עדכון צד שרת:

קיים קובץ עם דוגמא לפקודה בתוך הפרוייקט dataModel , יש למלא את הנתונים בפקודה בהתאם,  
ולבצע דרך Nuget console.  
לאחר ביצוע הפקודה ייוצרו מלא קבצים בפרוייקט ManagingDebts , יש למחוק את הקבצים שנוצרו זה  
זה סתם שכפול  
לאחר מכן יש להוסיף את הדברים הבאים:

<http://prntscr.com/re6h64>

<http://prntscr.com/re6hf>

זה קורה כי הcontext מתרענן גם הוא בעדכון db , ועל מנת שהפרוייקט ישען על appsettings צריך  
לבצע את זה

וזהו...

## נספח א'

### מבנה פקודת יומן:

פקודת יומן מייצגת תנועה של כסף בתוך הנהלת חשבונות. מכון שלא מדובר בהצהרה של בעל העסק (במקרה הזה עירייה כזו או אחרת) עבור תשלום עתידי, אלא הצהרה על תנועה כספית שכבר בוצעה, הפקודה חייבת להיות מאוזנת בזכות וחובה

אפשר לראות זכות וחובה כמו פעולת פלוס ומינוס, כאשר חובה היא פעולת מינוס וזכות היא פעולת פלוס. הסבר יותר נכון הוא: כאשר בעל עסק מדווח על תנועה כספית, הוא בעצם **מחייב** את עצמו, ומזכה את המוטב בעסקה

### דוגמא:

כאשר עיריית בני ברק משלמת לבזק עבור שירותי תשתית כאלו ואחרים, עיריית בני ברק **מחייבת** את עצמה **ומזכה** את בזק וכך בהנהלת החשבונות מתעדכן הסכום שהעירייה שילמה לספק מסוים, ומאיפה היא שילמה אותו

ולכן כאשר אנחנו מעבירים פקודת יומן היא חייבת להיות מאוזנת, מיכון שהסכום **שזוכה** (זאת אומרת שולם) לספק חייב להיות קיים גם **בחייב** בעירייה

### חלוקה לסעיפים תקציביים:

כל עירייה מתנהלת עם סעיפים תקציביים בכדי לדווח ולנהל את ההוצאות שלה, למשל לכל עירייה יש סעיף תקציבי שמתעסק בחינוך, או בתשתיות או באבטחה וכו' בכל שנה הסעיף התקציבי הזה מתוקצב בכמות מסויימת של כסף, שזו הכמות אותה הוא יכול "לבזבז" בשנת הכספים

ולכן עבור כל ספק קיים סעיף תקציבי אשר **מחוייב** על סך העסקאות מולו.

מהצד של הספק, כל גביה של כסף מתבצעת לרוב אל מול חוזה מסויים, כאשר החוזה יכול לייצג מקום אספקה או סוג מסויים של שירות ועוד

ולכן הקשר הלוגי בין חוזים לסעיפים תקציביים הוא כדלקמן: "לכל חוזה קיים לפחות סעיף תקציבי אחד, ותחת סעיף תקציבי אחד יכולים להיות כמה חוזים" כלומר, קשר של רבים לרבים.

כאשר יוצרים פקודת יומן אנחנו צריכים להבין מי הסעיף התקציבי **המחוייב** עבור חוזה מסויים, ומומלץ לבצע הפרדה של החיובים בכל חשבונית וחשבונית כלומר, אם סעיף תקציבי מסויים מחוייב כמה פעמים בכמה חשבוניות שונות, מומלץ שתהיה שורת **חובה** של הסעיף עבור כל חשבונית

ולבסוף כל שורות החובה האלו צריכות להיות שוות לתנועת **זכות** אחת של הספק כלומר, **חייבנו** את העירייה וחילקנו את **החובה** הזו לסעיפים תקציביים שונים **וזיכנו** ספק יחיד אל מול הסכום **שחייבנו** את העירייה.

