**מערכת לניהול חברה**

**לציוד רפואי**



**מגיש:** אלמוג חמדני

**ת.ז:** 212940191

**מנחה:** עוז ארווץ

**בית ספר:** מקיף יא' אשדוד

**שם החלופה:** תכנון ותכנות מערכות

**תאריך הגשה:** 20/5/2020

תוכן עניינים

[מבוא 6](#_Toc40885121)

[נושא הפרויקט 6](#_Toc40885122)

[רקע כללי על החברה 6](#_Toc40885123)

[מהות התוכנה 6](#_Toc40885124)

[תהליכים מרכזיים 7](#_Toc40885125)

[ניהול עובדים 7](#_Toc40885126)

[ניהול לקוחות 7](#_Toc40885127)

[ניהול ספקים 8](#_Toc40885128)

[ניהול מלאי 8](#_Toc40885129)

[מדריך למשתמש 9](#_Toc40885130)

[מסך התחברות 9](#_Toc40885131)

[מסך ראשי 10](#_Toc40885132)

[תפריט התוכנה 11](#_Toc40885133)

[ניהול עובדים 12](#_Toc40885134)

[הצגת עובדים הקיימים במערכת 12](#_Toc40885135)

[הוספת עובד חדש 13](#_Toc40885136)

[הצגת סוגי עובדים הקיימים במערכת 14](#_Toc40885137)

[הוספת סוג עובד חדש 15](#_Toc40885138)

[הצגת כל ההוצאות הלא שוטפות 16](#_Toc40885139)

[הוספת הוצאה לא שוטפת של עובד 17](#_Toc40885140)

[ניהול לקוחות 18](#_Toc40885141)

[הצגת לקוחות הקיימים במערכת 18](#_Toc40885142)

[הוספת לקוח חדש 19](#_Toc40885143)

[הצגת כל הזמנות הלקוח השונות והפריטים של ההזמנה 20](#_Toc40885144)

[הוספת הזמנת לקוח 21](#_Toc40885145)

[הוספת פריט להזמנת לקוח 22](#_Toc40885146)

[ניהול ספקים 23](#_Toc40885147)

[הצגת ספקים הקיימים במערכת 23](#_Toc40885148)

[הוספת ספק חדש 24](#_Toc40885149)

[הצגת כל ההזמנות מספקים והפריטים של ההזמנה 25](#_Toc40885150)

[הוספת הזמנה מספק 26](#_Toc40885151)

[הוספת פריט להזמנה מספק 27](#_Toc40885152)

[ניהול מלאי 28](#_Toc40885153)

[הצגת מחסנים הקיימים במערכת 28](#_Toc40885154)

[הוספת מחסן חדש 29](#_Toc40885155)

[הצגת אצוות הקיימות במערכת 30](#_Toc40885156)

[הוספת אצווה חדשה 31](#_Toc40885157)

[הצגת הפריטים הקיימים במערכת 32](#_Toc40885158)

[הוספת פריט חדש 33](#_Toc40885159)

[הצגת כל המלאי הזמין 34](#_Toc40885160)

[הוספת מלאי חדש 35](#_Toc40885161)

[דו"חות 36](#_Toc40885162)

[דו"ח התפלגות הוצאות לא שוטפות בין עובדים 36](#_Toc40885163)

[דו"ח המוצרים הנמכרים ביותר 37](#_Toc40885164)

[דו"ח הזמנות לקוח חודשיות 38](#_Toc40885165)

[דו"ח הזמנות מספק חודשיות 39](#_Toc40885166)

[דו"ח מאזן כספים חודשי 40](#_Toc40885167)

[בסיס הנתונים 41](#_Toc40885168)

[דיאגרמת מבנה הנתונים 41](#_Toc40885169)

[טבלת employee 42](#_Toc40885170)

[טבלת employee\_type 42](#_Toc40885171)

[טבלת customer 42](#_Toc40885172)

[טבלת vendor 42](#_Toc40885173)

[טבלת part 43](#_Toc40885174)

[טבלת expense 43](#_Toc40885175)

[טבלת lot 43](#_Toc40885176)

[טבלת address 43](#_Toc40885177)

[טבלת warehouse 43](#_Toc40885178)

[טבלת inventory 44](#_Toc40885179)

[טבלת sale\_order 44](#_Toc40885180)

[טבלת purchase\_order 44](#_Toc40885181)

[טבלה sale\_order\_part 44](#_Toc40885182)

[טבלה sale\_order\_part 44](#_Toc40885183)

[מדריך למפתח 45](#_Toc40885184)

[ספריות ליבה 45](#_Toc40885185)

[WPF 45](#_Toc40885186)

[MVVM 45](#_Toc40885187)

[Material Design 45](#_Toc40885188)

[OxyPlot 45](#_Toc40885189)

[מרכיבי ליבה 45](#_Toc40885190)

[DataHandler 45](#_Toc40885191)

[DataTableViewModel 46](#_Toc40885192)

[DualDataTableViewModel 46](#_Toc40885193)

[מחלקות הפרויקט 47](#_Toc40885194)

[DataTableViewModel 47](#_Toc40885195)

[IDataHandler 51](#_Toc40885196)

[DualDataTableViewModel 53](#_Toc40885197)

[ConnectionViewModel 54](#_Toc40885198)

[ShellViewModel 56](#_Toc40885199)

[CustomersViewModel 58](#_Toc40885200)

[EmployeesViewModel 59](#_Toc40885201)

[EmployeeTypesViewModel 60](#_Toc40885202)

[ExpensesViewModel 61](#_Toc40885203)

[InventoryViewModel 62](#_Toc40885204)

[LotsViewModel 63](#_Toc40885205)

[PartsViewModel 64](#_Toc40885206)

[PurchaseOrderPartsViewModel 65](#_Toc40885207)

[PurchaseOrdersViewModel 66](#_Toc40885208)

[SaleOrderPartsViewModel 68](#_Toc40885209)

[SaleOrdersViewModel 69](#_Toc40885210)

[VendorsViewModel 71](#_Toc40885211)

[WarehousesViewModel 72](#_Toc40885212)

[ExpensesDistributionViewModel 73](#_Toc40885213)

[MonthlyBalanceViewModel 74](#_Toc40885214)

[MonthlyPurchaseOrdersViewModel 76](#_Toc40885215)

[MonthlySaleOrdersViewModel 78](#_Toc40885216)

[SoldGoodsViewModel 80](#_Toc40885217)

[OrdersViewModel 82](#_Toc40885218)

[CustomersDataHandler 83](#_Toc40885219)

[EmployeesDataHandler 84](#_Toc40885220)

[EmployeeTypesDataHandler 85](#_Toc40885221)

[ExpensesDataHandler 86](#_Toc40885222)

[InventoryDataHandler 87](#_Toc40885223)

[LotsDataHandler 88](#_Toc40885224)

[PartsDataHandler 89](#_Toc40885225)

[PurchaseOrderPartsDataHandler 90](#_Toc40885226)

[PurchaseOrdersDataHandler 91](#_Toc40885227)

[SaleOrderPartsDataHandler 92](#_Toc40885228)

[SaleOrdersDataHandler 93](#_Toc40885229)

[VendorsDataHandler 95](#_Toc40885230)

[WarehousesDataHandler 96](#_Toc40885231)

[BoolToStringConverter 97](#_Toc40885232)

[BoolToVisibilityConverter 98](#_Toc40885233)

[DateTimeHelper 99](#_Toc40885234)

[FloatConverter 100](#_Toc40885235)

[LotTypeEnumToStringConverter 101](#_Toc40885236)

[PartUnitEnumToStringConverter 102](#_Toc40885237)

[PasswordBoxHelper 103](#_Toc40885238)

[RelayCommand 104](#_Toc40885239)

[Utils 105](#_Toc40885240)

[סיכום אישי 107](#_Toc40885241)

# מבוא

## נושא הפרויקט

הפרויקט הוא הקמה של מערכת ממוחשבת לניהול חברה לציוד רפואי (בעיקר תחבושות רפואיות).

## רקע כללי על החברה

Core Scientific Creations נוסדה בשנת 2012. המשרדים והמפעל ממוקמים בפתח תקווה. החברה הינה חברה לציוד רפואי המייצרת תחבושות המוסטטיות לעצירת דימום. החברה מוכרת את מוצריה בארץ ובעולם לבתי חולים, גופים צבאיים וגופים אזרחיים.

## מהות התוכנה

התכנה נכתבה במטרה לייעל את דרכי ההתנהלות של החברה. מטרתה של התוכנה היא שמירת היסטוריה רחבה של כל נתוני החברה ומתן דרך ארגון נתוני החברה בדרך נוחה ונגישה. כמו כן, לתוכנה מטרה נוספת שהיא מניעת איבוד מידע של נתוני החברה שכן כרגע החברה משתמשת בתוכנה לא יציבה אשר ישנם מקרים בהם היא מאבדת את נתוני החברה. לתוכנה יש מטרה נוספת שהיא מתן דרך מהירה לחיפוש ושליפת נתונים על לקוחות, עובדים, ספקים ועוד נתוני חברה. יתר על כן, לתוכנה יש יתרון נוסף שהוא שמירת הנתונים במסד נתונים שדורש אותנטיקציה ובכך ניתן להגן על הנתונים.

# תהליכים מרכזיים

## ניהול עובדים

בעזרת תהליך זה יכול מנהל העסק לרשום את עובדיו לעסק, לפטרם וכן לערוך פרטיהם.

## ניהול לקוחות

בעזרת תהליך זה יכולים עובדי החברה לנהל את הלקוחות השונים בחברה (הוספת לקוח, מחיקת לקוח וכו') ולקבל הזמנות מן הלקוחות ולבצע שליחת ההזמנה המוכנה אל הלקוח.

## ניהול ספקים

בעזרת תהליך זה יכולים עובדי החברה לנהל את הספקים השונים בחברה (הוספת ספק, מחיקת ספק וכו') ולשלוח הזמנות לספקים ולאשר הזמנות שמתקבלים במחסני החברה.

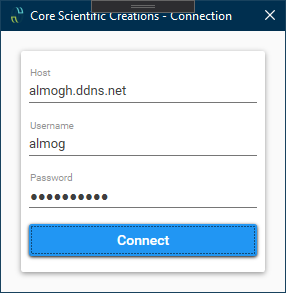
## ניהול מלאי

בעזרת תהליך זה יכולים עובדי החברה לנהל את הפריטים השונים המוכרים במערכת החברה, לנהל את האצוות השונות, את המחסנים השונים ולבדוק מהו מצב המלאי הזמין בעסק וכן לחדש את המלאי כאשר מתקבלת הזמנת לקוח חדשה.

# מדריך למשתמש

## מסך התחברות

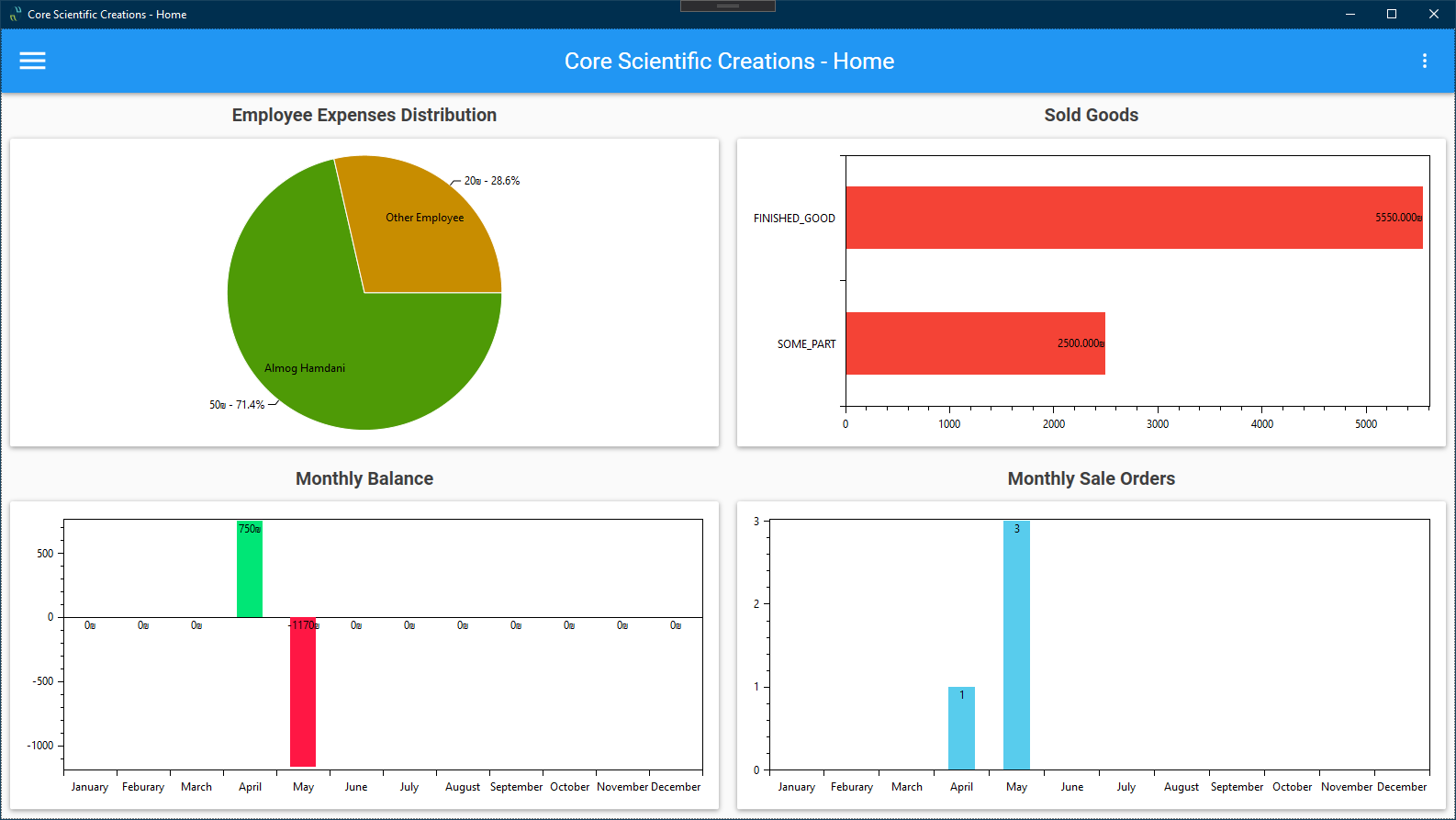
זה המסך ההתחלתי של התוכנה. במסך זה יש לבחור את השרת (דומיין או IP) שבו נמצא מסד הנתונים. כמו כן, מסך זה כולל את השם משתמש והסיסמא שאיתם הקליינט מתחבר אל מסד הנתונים.



## מסך ראשי

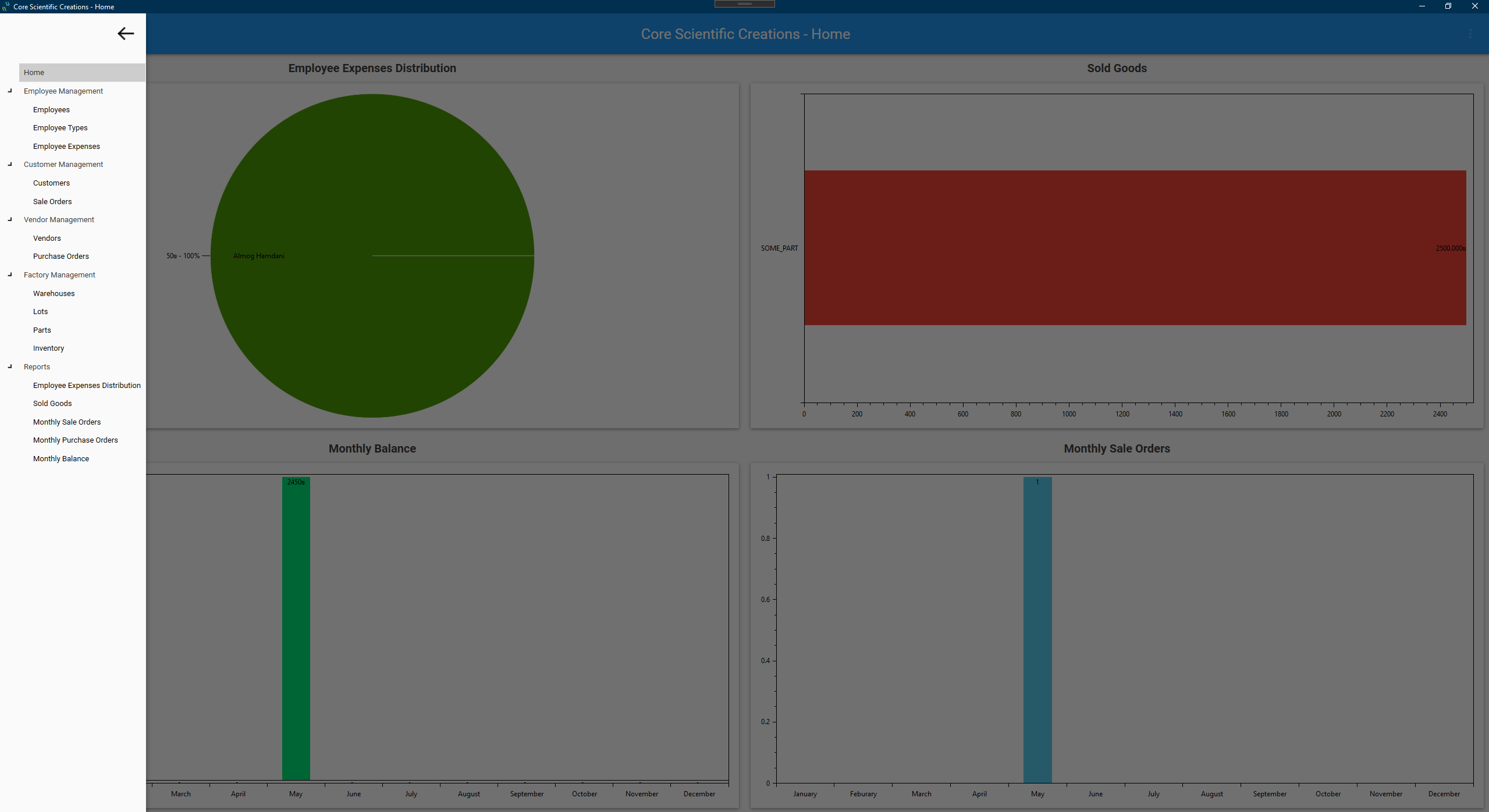
זה הוא המסך הראשי של התוכנה. ניתן לראות כי במסך זה מופיעים מספר מהדוחות שהתוכנה מציעה כמו הוצאות לא שוטפות של עובדים, מאזן חודשי, הזמנות לקוח חודשיות וכו'.

ניתן לראות כי בצד ימין מופיע כפתור אשר פותח את תפריט התוכנה.



## תפריט התוכנה

ניתן לראות בתפריט התוכנה את כל התהליכים השונים שמציעה התוכנה.

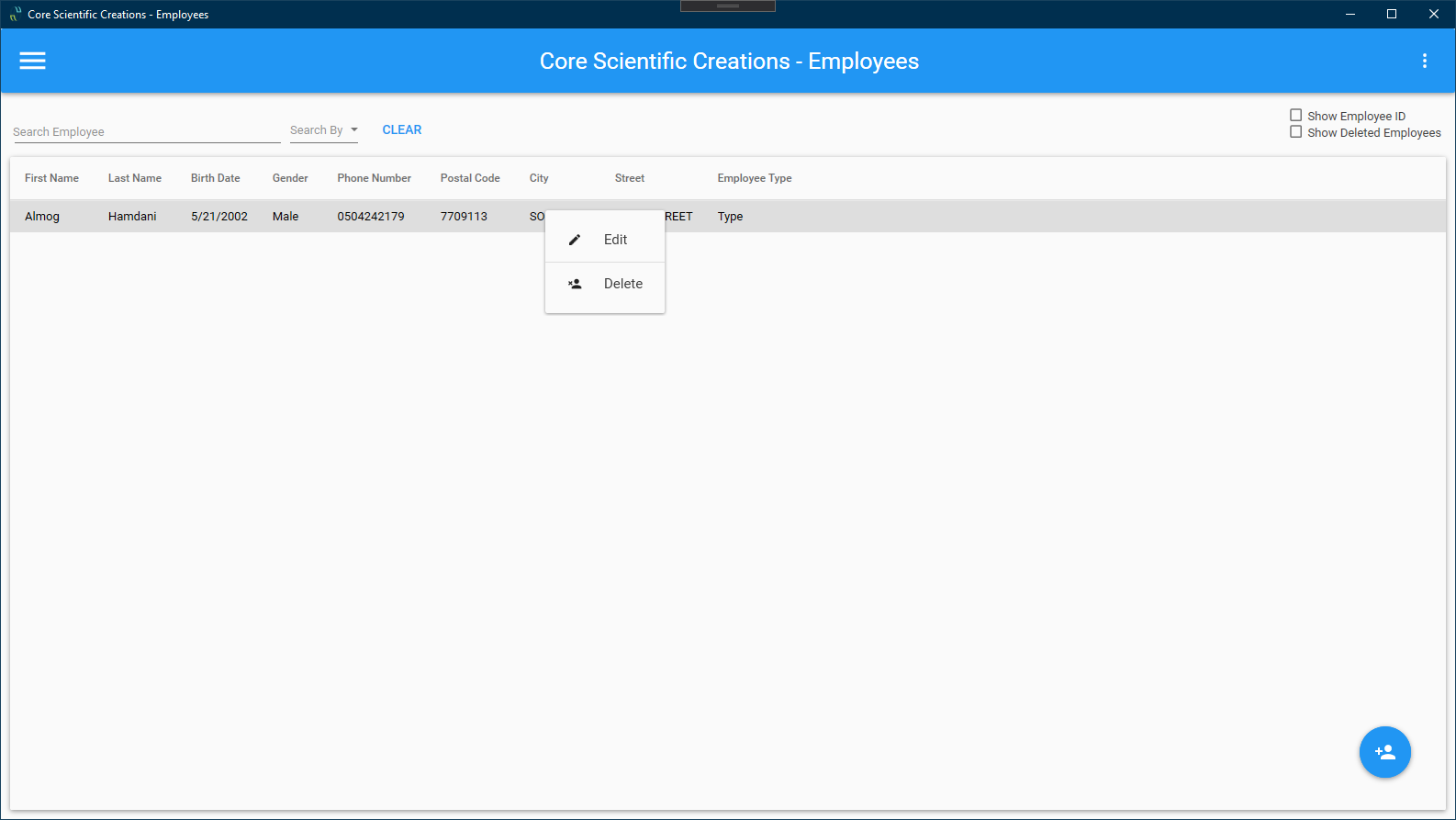


## ניהול עובדים

### הצגת עובדים הקיימים במערכת

במסך זה ניתן לראות את כל העובדים הקיימים במערכת.

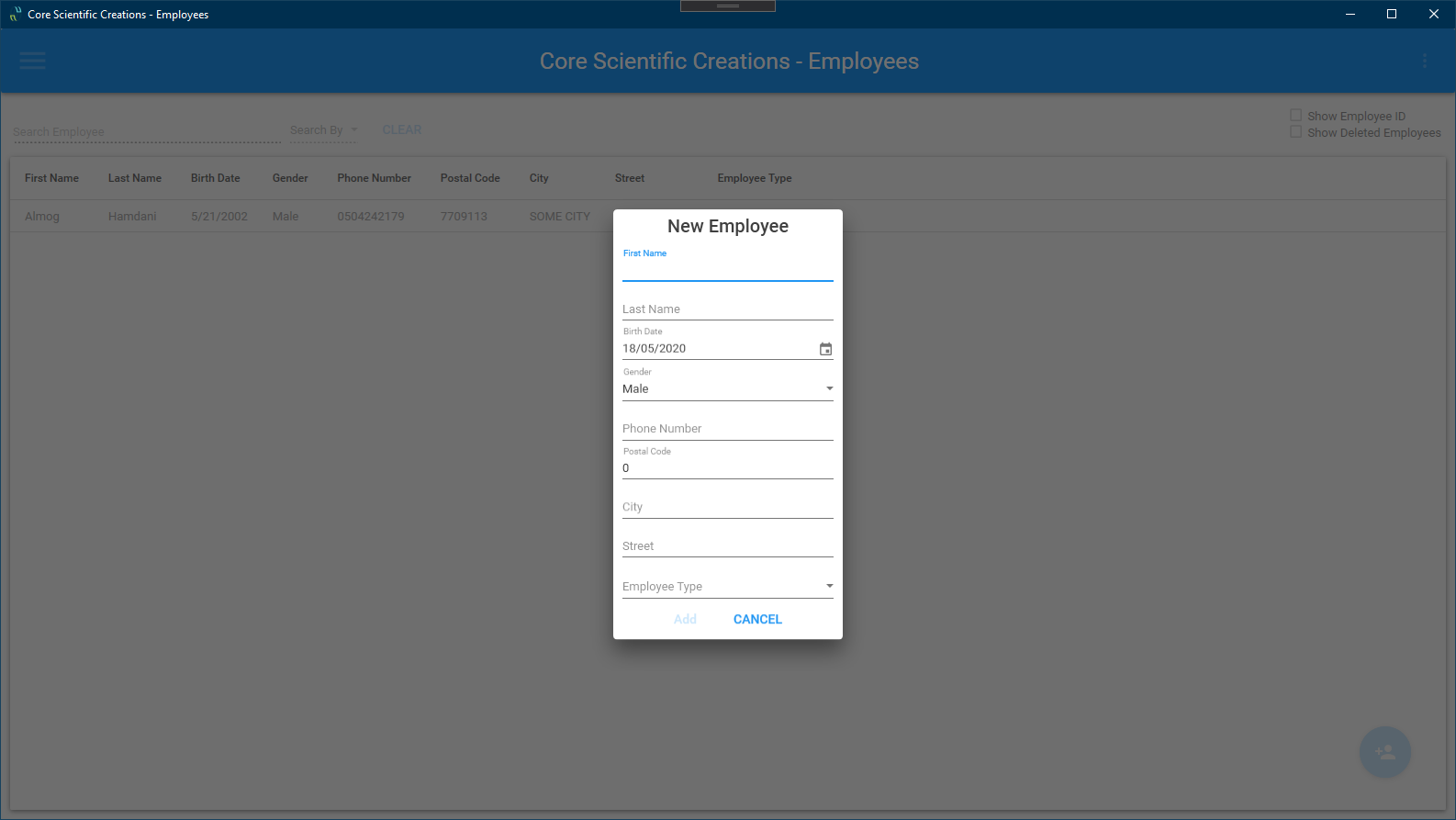
ניתן לראות כי קיימות אופציות שונות כמו חיפוש עובדים לפי השדות השונים, הסתרת מספר העובד והסתרת עובדים מפוטרים. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת עובד חדש וכי לחיצת מקש ימני על אחד העובדים תגרום לפתיחת תפריט המאפשר מחיקת העובד או עריכת הפרטים שלו.



### הוספת עובד חדש

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של המסך.

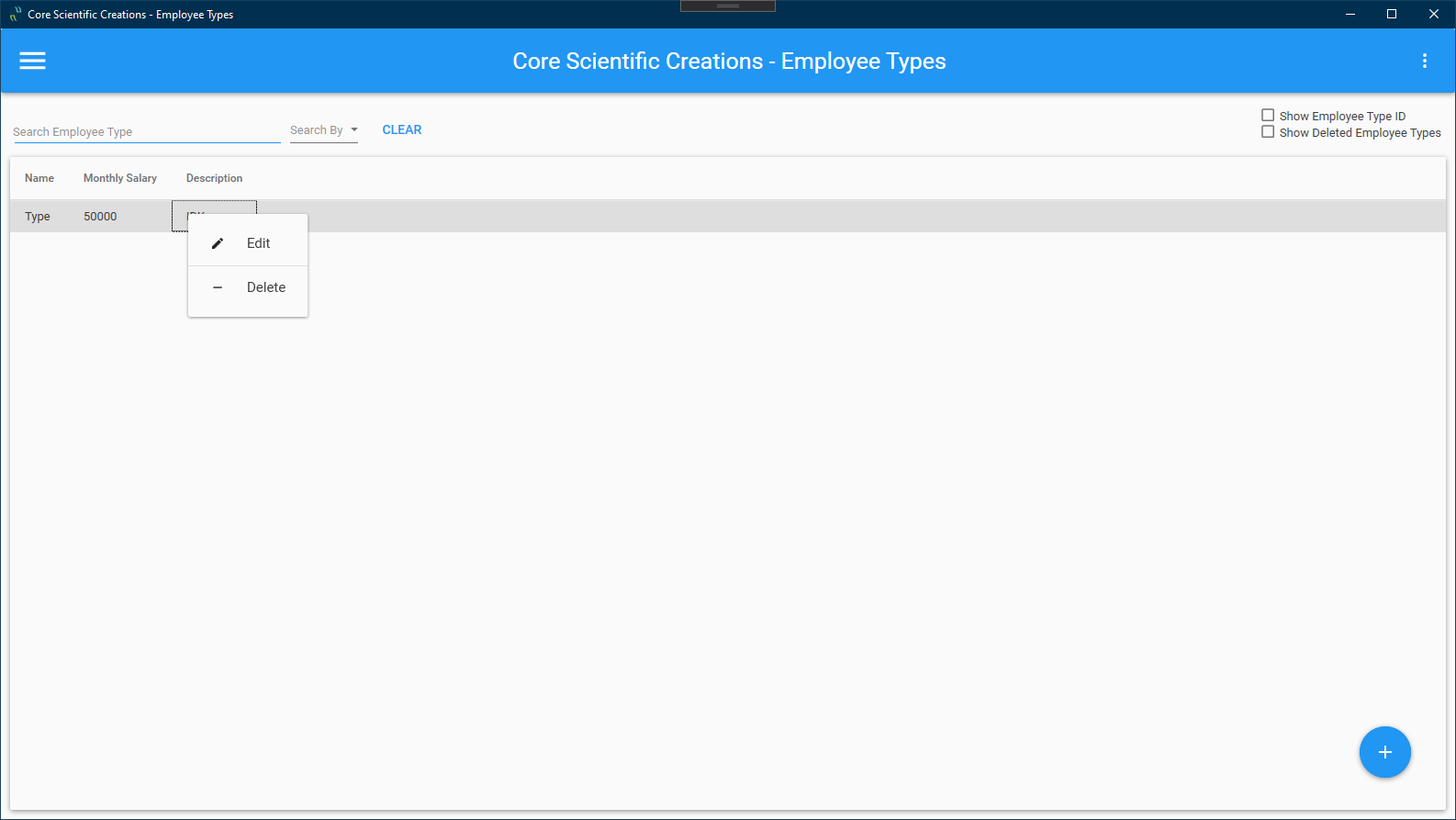
ניתן לראות כי במסך זה ניתן ליצור עובד לפי השדות השונים ולבחור לו סוג עובד.



### הצגת סוגי עובדים הקיימים במערכת

במסך זה ניתן לראות את כל סוגי העובדים הקיימים במערכת.

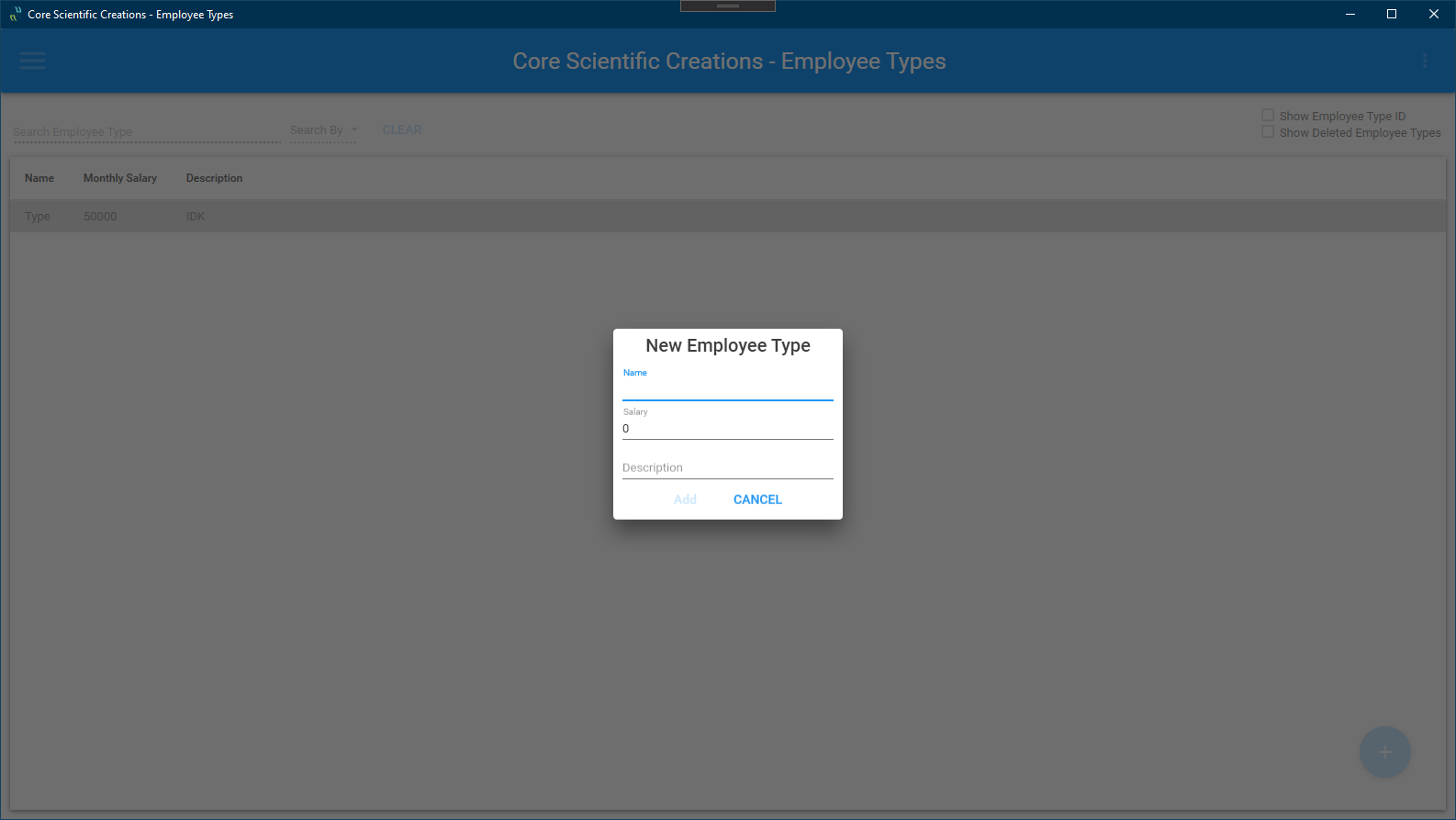
ניתן לראות כי קיימות אופציות שונות כמו חיפוש סוגי עובדים לפי השדות השונים, הסתרת מספר סוג העובד והסתרת סוג עובדים מחוקים. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת סוג עובד חדש וכי לחיצת מקש ימני על אחד מסוגי העובדים תגרום לפתיחת תפריט המאפשר מחיקת הסוג או עריכת הפרטים שלו.



### הוספת סוג עובד חדש

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של המסך.

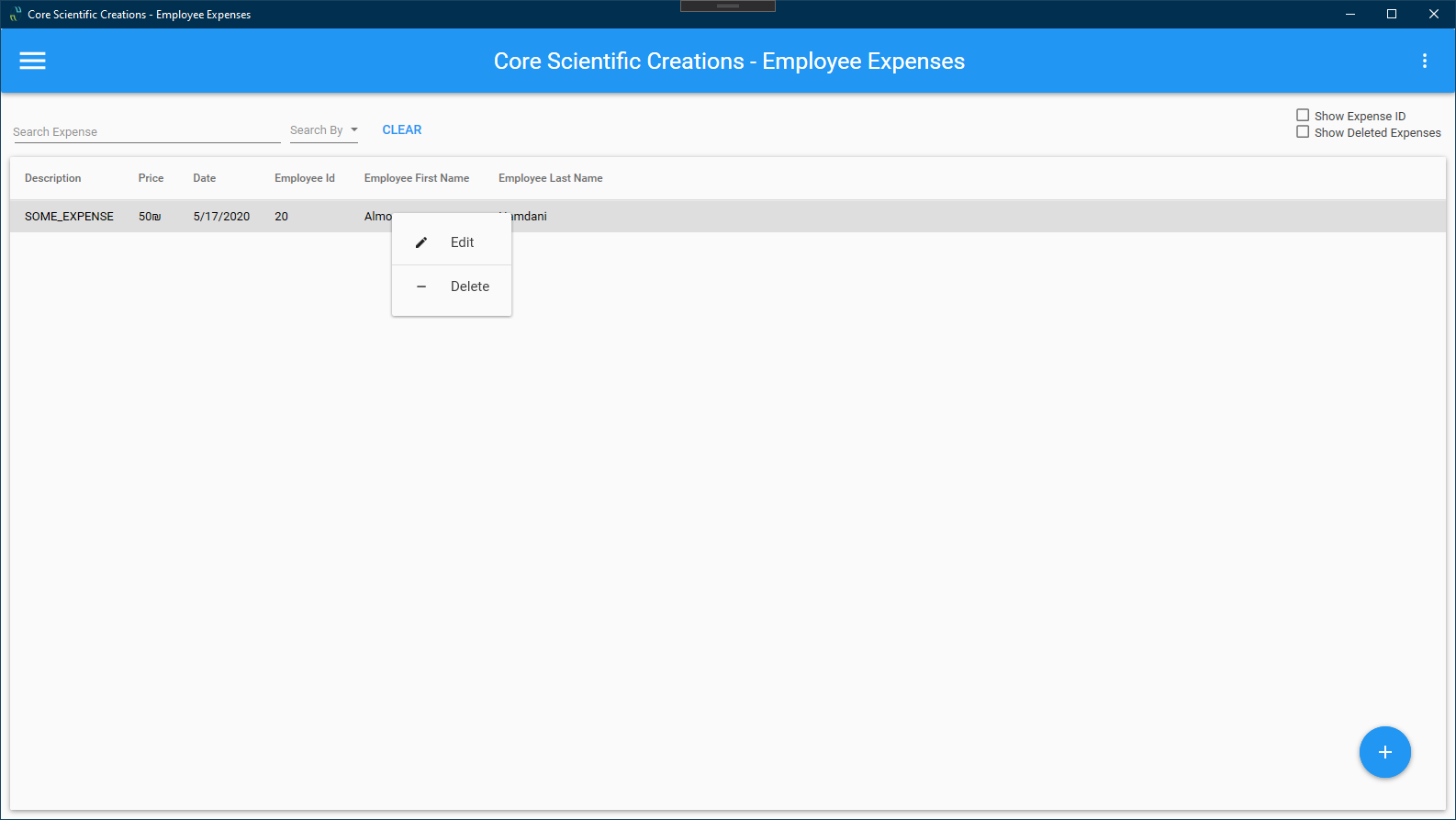
ניתן לראות כי במסך זה ניתן ליצור סוג עובד לפי השדות השונים.



### הצגת כל ההוצאות הלא שוטפות

במסך זה ניתן לראות את כל ההוצאות הלא שוטפות הקיימות במערכת.

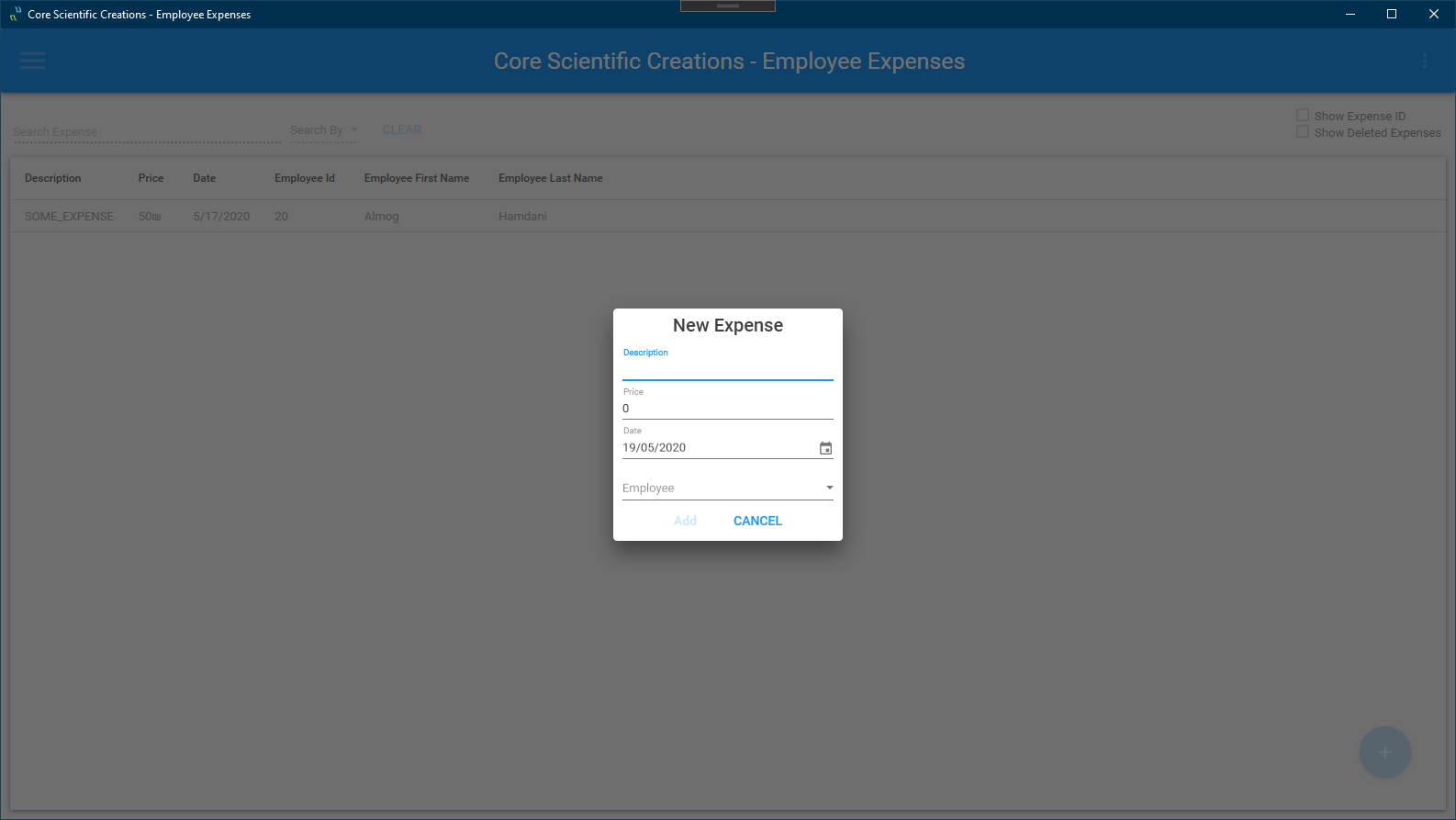
ניתן לראות כי קיימות אופציות שונות כמו חיפוש הוצאות לא שוטפות לפי השדות השונים, הסתרת מספר ההוצאה הלא שוטפת והסתרת הוצאות מחוקות. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת הוצאה לא שוטפת חדש וכי לחיצת מקש ימני על אחד מההוצאה הלא השוטפות תגרום לפתיחת תפריט המאפשר מחיקת ההוצאה או עריכת הפרטים שלה.



### הוספת הוצאה לא שוטפת של עובד

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של המסך.

ניתן לראות כי במסך זה ניתן ליצור הוצאה לא שוטפת לפי השדות השונים.

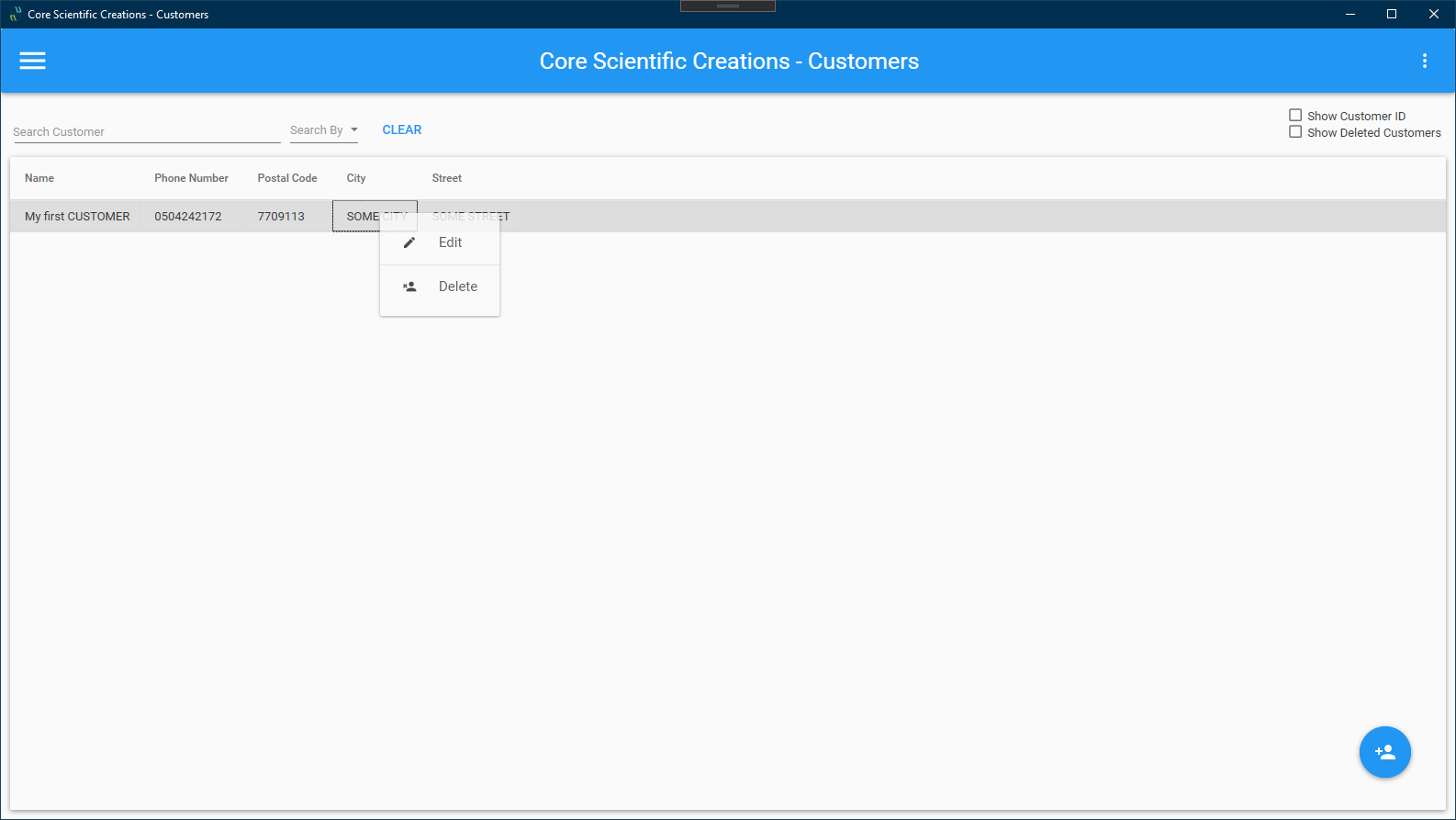


## ניהול לקוחות

### הצגת לקוחות הקיימים במערכת

במסך זה ניתן לראות את כל הלקוחות הקיימים במערכת.

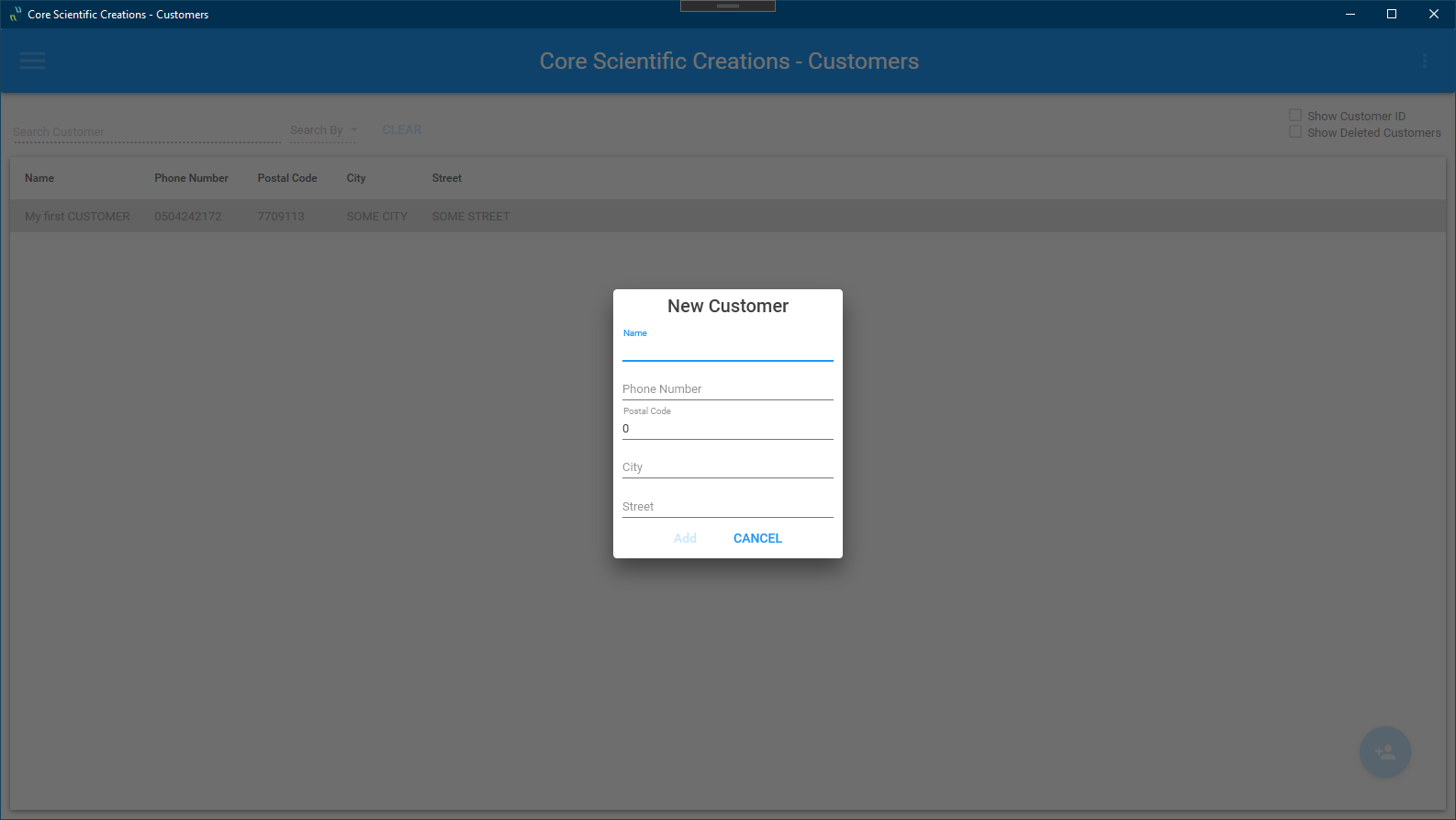
ניתן לראות כי קיימות אופציות שונות כמו חיפוש לקוחות לפי השדות השונים, הסתרת מספר הלקוח והסתרת לקוחות מחוקים. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת לקוח חדש וכי לחיצת מקש ימני על אחד מהלקוחות תגרום לפתיחת תפריט המאפשר מחיקת הלקוח או עריכת הפרטים שלו.



### הוספת לקוח חדש

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של המסך.

ניתן לראות כי במסך זה ניתן להוסיף לקוח חדש לפי השדות השונים.

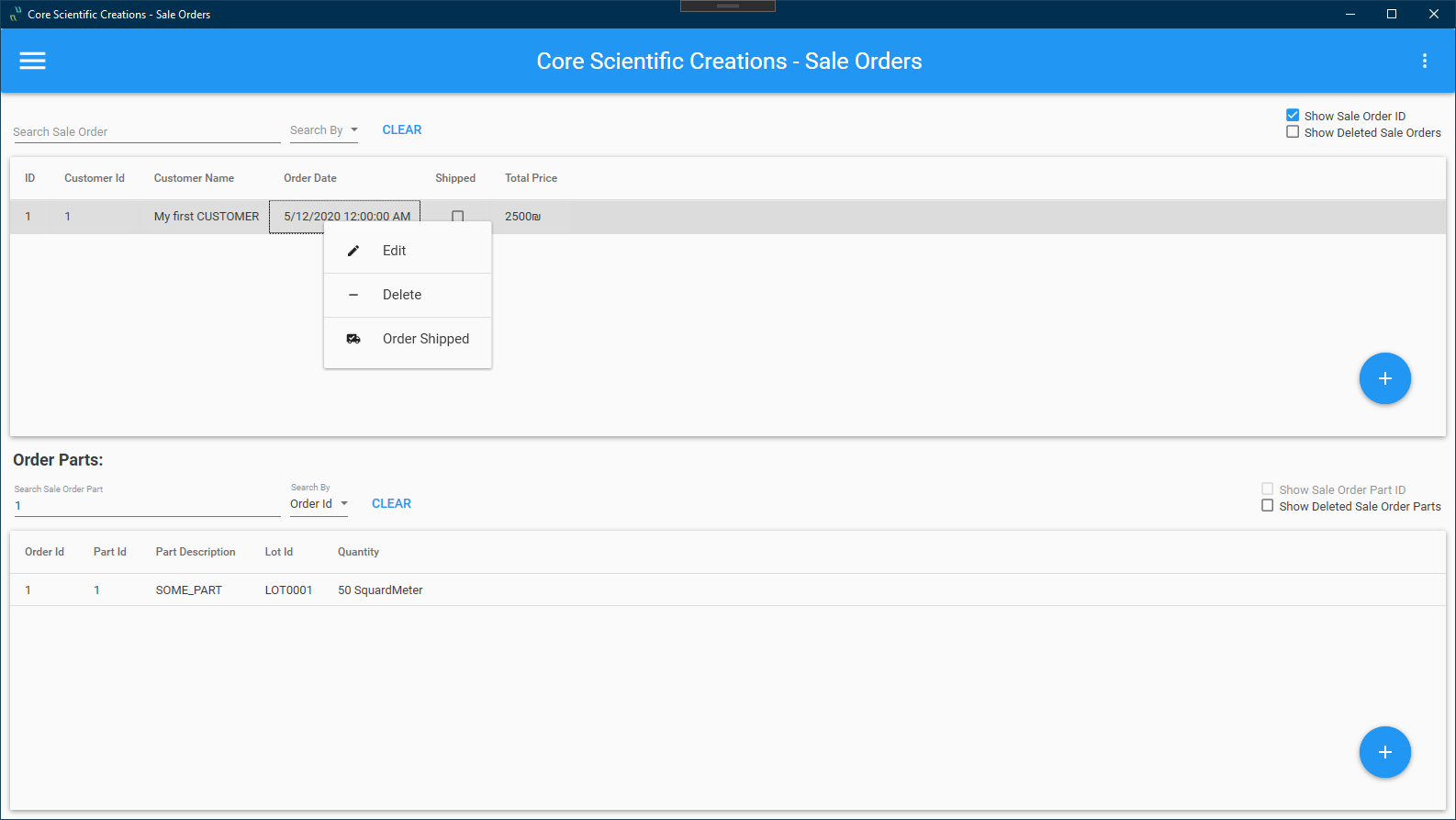


### הצגת כל הזמנות הלקוח השונות והפריטים של ההזמנה

במסך זה ניתן לראות את כל הזמנות הלקוח השונות הקיימים עבור כל לקוח. לאחר בחירה של אחת ההזמנות, הפריטים של ההזמנה יופיעו במסך השני.

ניתן לראות כי קיימות אופציות שונות כמו חיפוש הזמנות לקוח לפי השדות השונים, הסתרת מספר ההזמנה והסתרת הזמנות מחוקות. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת הזמנת לקוח חדשה וכי לחיצת מקש ימני על אחד מההזמנות תגרום לפתיחת תפריט המאפשר מחיקת ההזמנה או עריכת הפרטים שלה. יתר על כן, קיימת אופציה המאפשרת לסמן את ההזמנה כנשלחה ללקוח אשר תגרום לפריטים של ההזמנה לרדת מן המלאי.

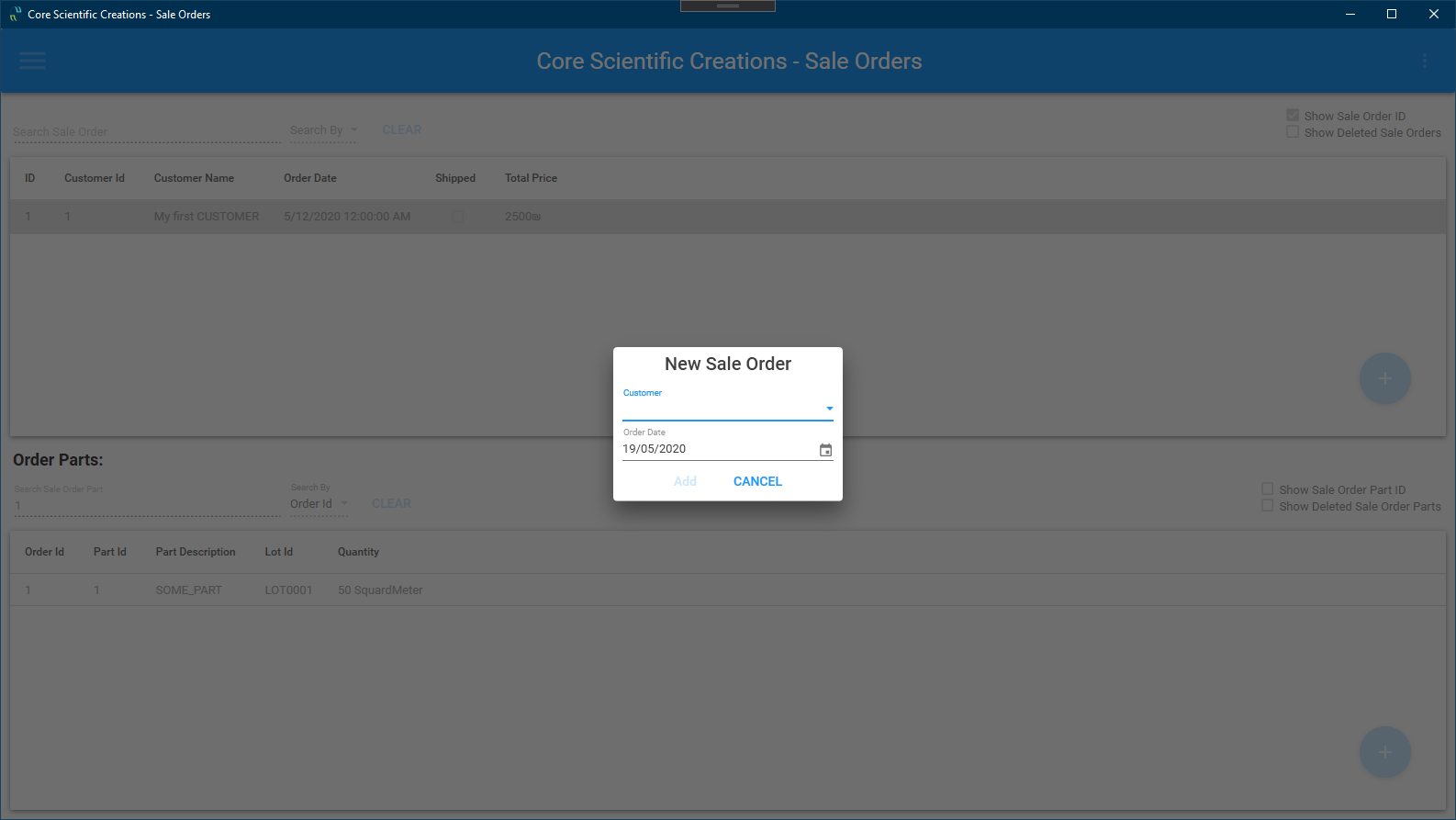
כמו כן, ניתן לראות כי במסך השני יש את אותן האופציות עבור הפריטים של אותה הזמנה שנבחרה.



### הוספת הזמנת לקוח

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של מסך ההזמנות.

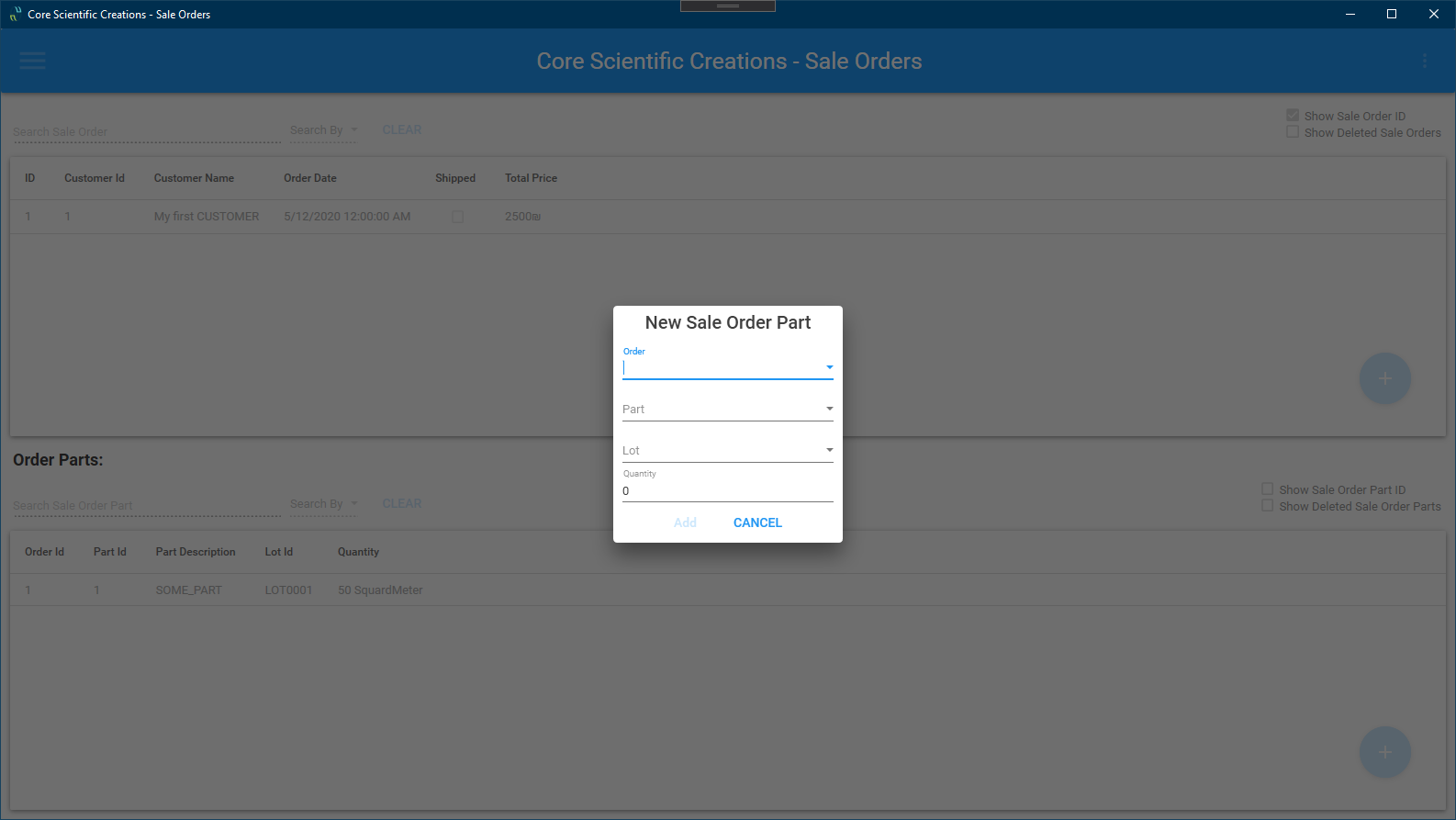
ניתן לראות כי במסך זה ניתן להוסיף הזמנת לקוח חדשה לפי השדות השונים.



### הוספת פריט להזמנת לקוח

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של מסך הפריטים.

ניתן לראות כי במסך זה ניתן להוסיף פריט חדש להזמנת לקוח לפי השדות השונים.

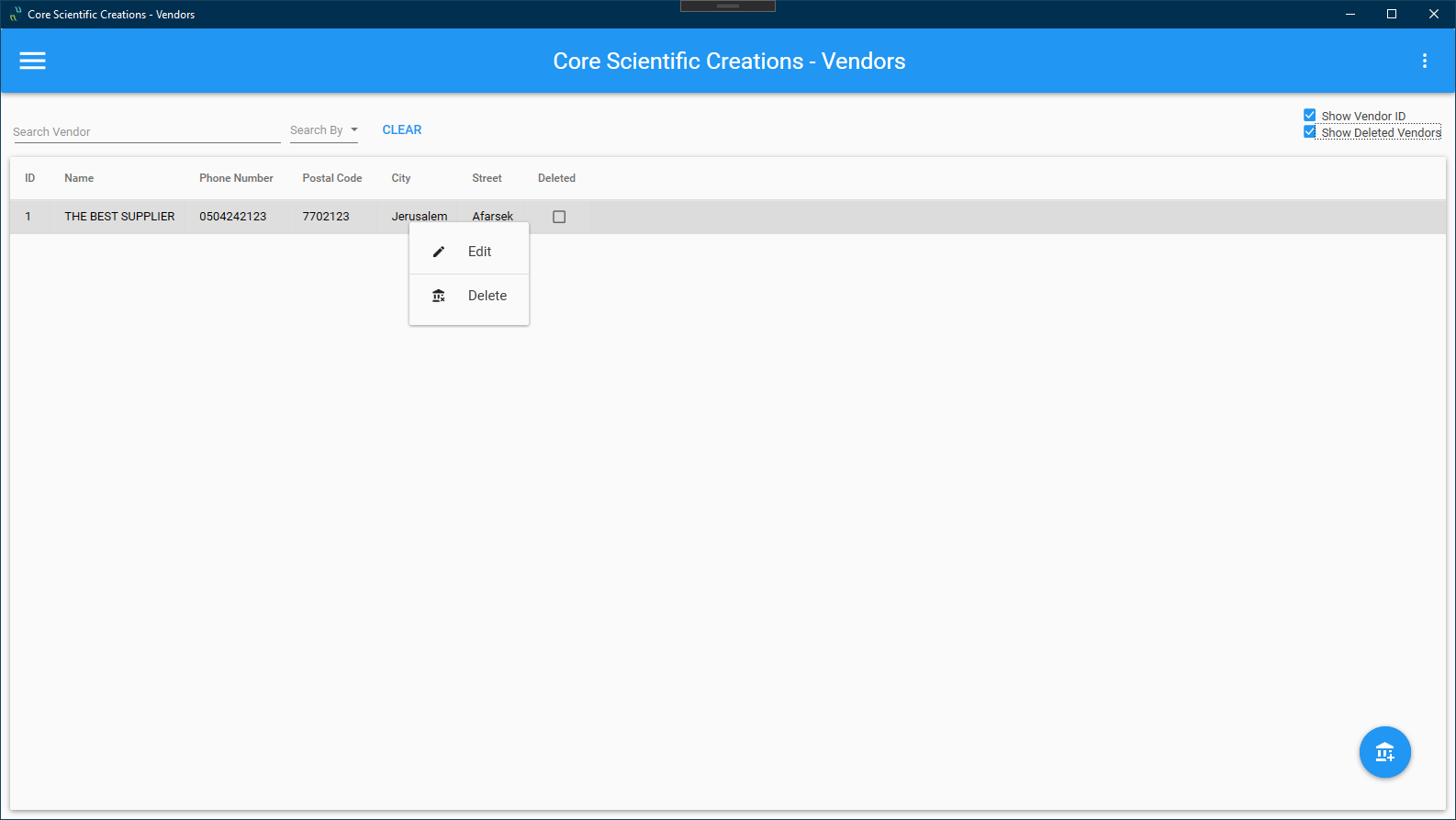


## ניהול ספקים

### הצגת ספקים הקיימים במערכת

במסך זה ניתן לראות את כל הספקים הקיימים במערכת.

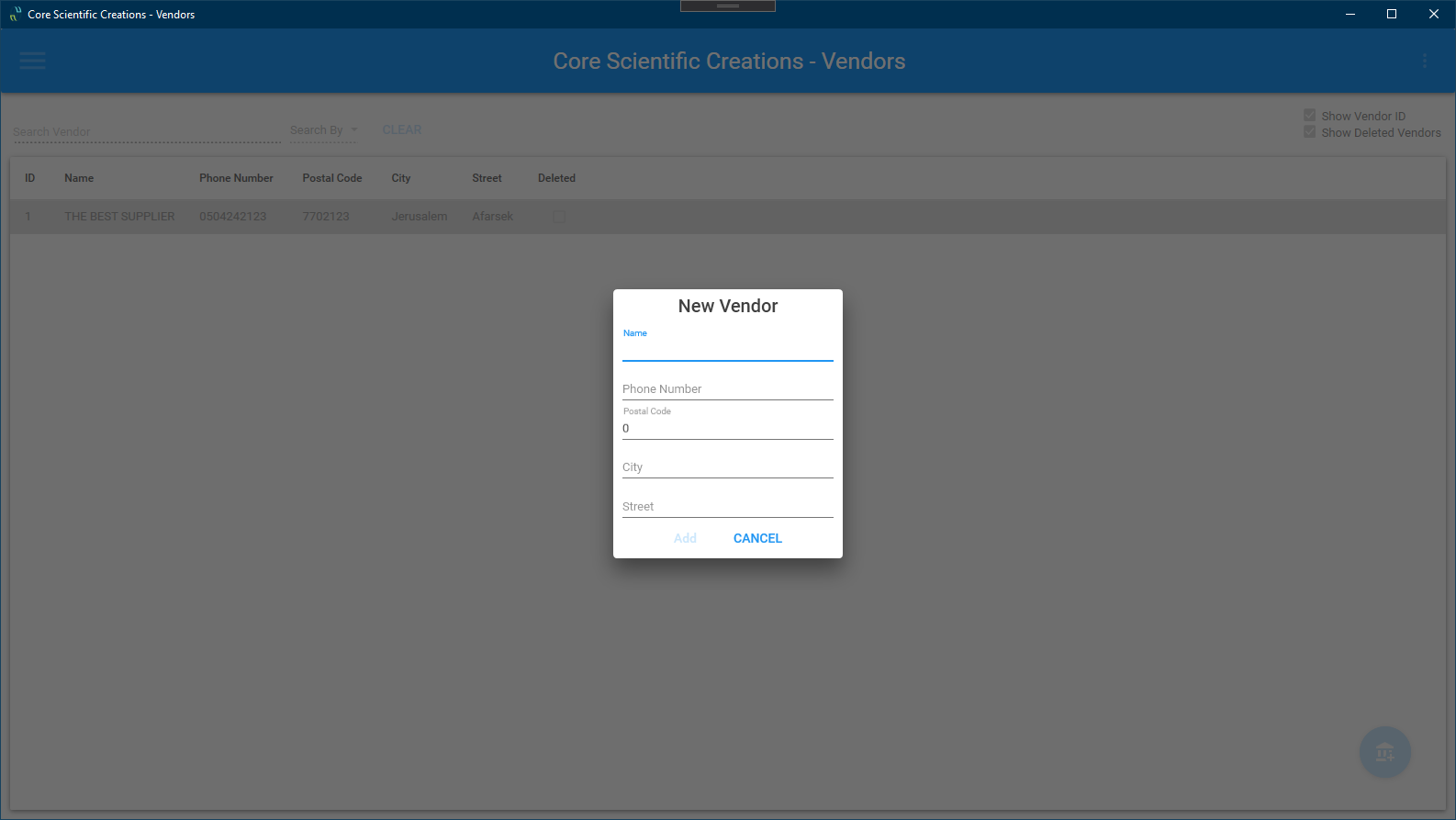
ניתן לראות כי קיימות אופציות שונות כמו חיפוש ספקים לפי השדות השונים, הסתרת מספר הספק והסתרת ספקים מחוקים. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת ספק חדש וכי לחיצת מקש ימני על אחד מהספקים תגרום לפתיחת תפריט המאפשר מחיקת הספק או עריכת הפרטים שלו.



### הוספת ספק חדש

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של המסך.

ניתן לראות כי במסך זה ניתן להוסיף ספק חדש לפי השדות השונים.

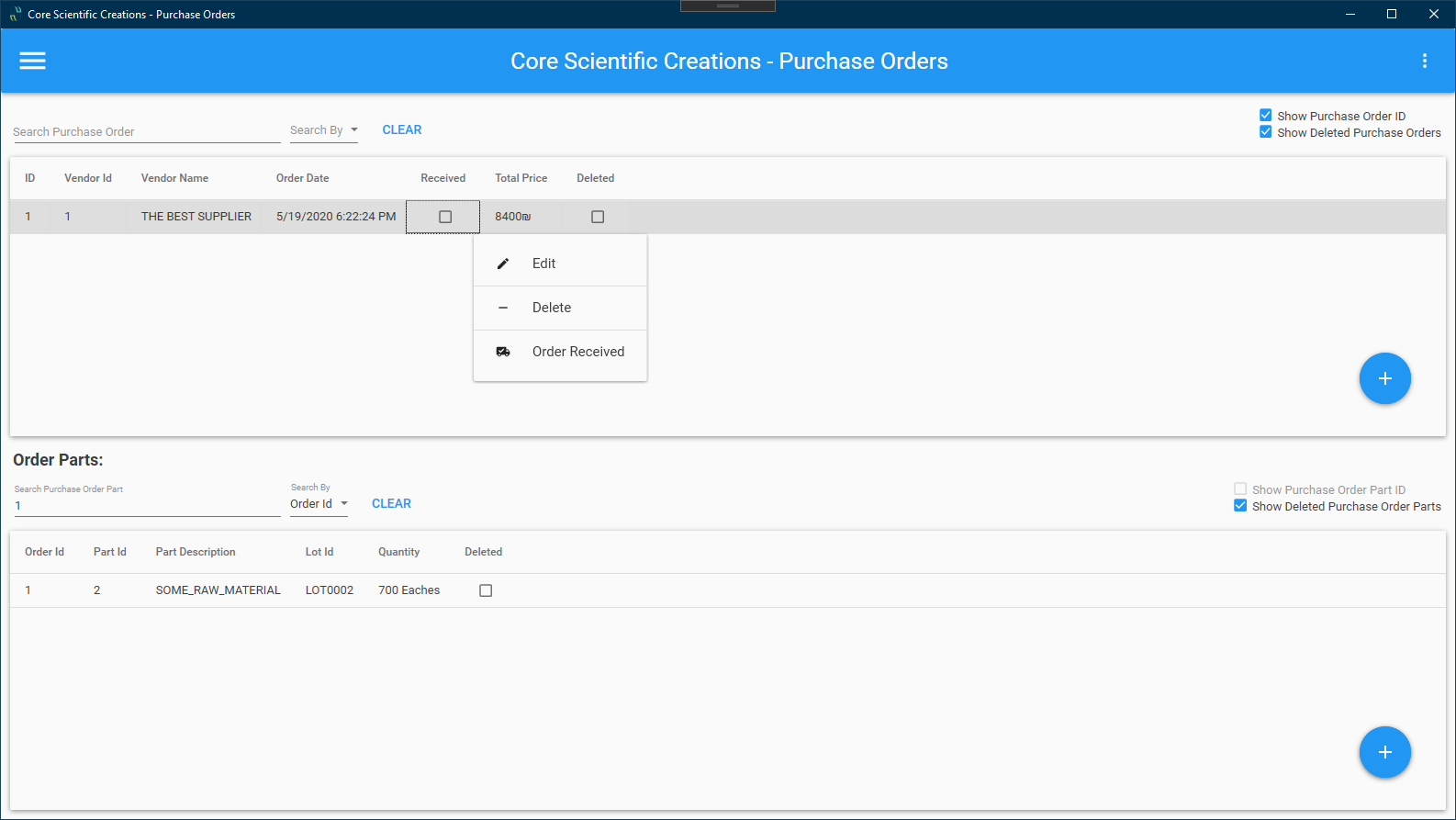


### הצגת כל ההזמנות מספקים והפריטים של ההזמנה

במסך זה ניתן לראות את כל ההזמנות מספקים הקיימים. לאחר בחירה של אחת ההזמנות, הפריטים של ההזמנה יופיעו במסך השני.

ניתן לראות כי קיימות אופציות שונות כמו חיפוש הזמנות מספקים לפי השדות השונים, הסתרת מספר ההזמנה והסתרת הזמנות מחוקות. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת הזמנה מספק חדשה וכי לחיצת מקש ימני על אחד מההזמנות תגרום לפתיחת תפריט המאפשר מחיקת ההזמנה או עריכת הפרטים שלה. יתר על כן, קיימת אופציה המאפשרת לסמן את ההזמנה כ-התקבלה מן הספק אשר יגרום לפריטים של ההזמנה להיכנס למלאי במערכת.

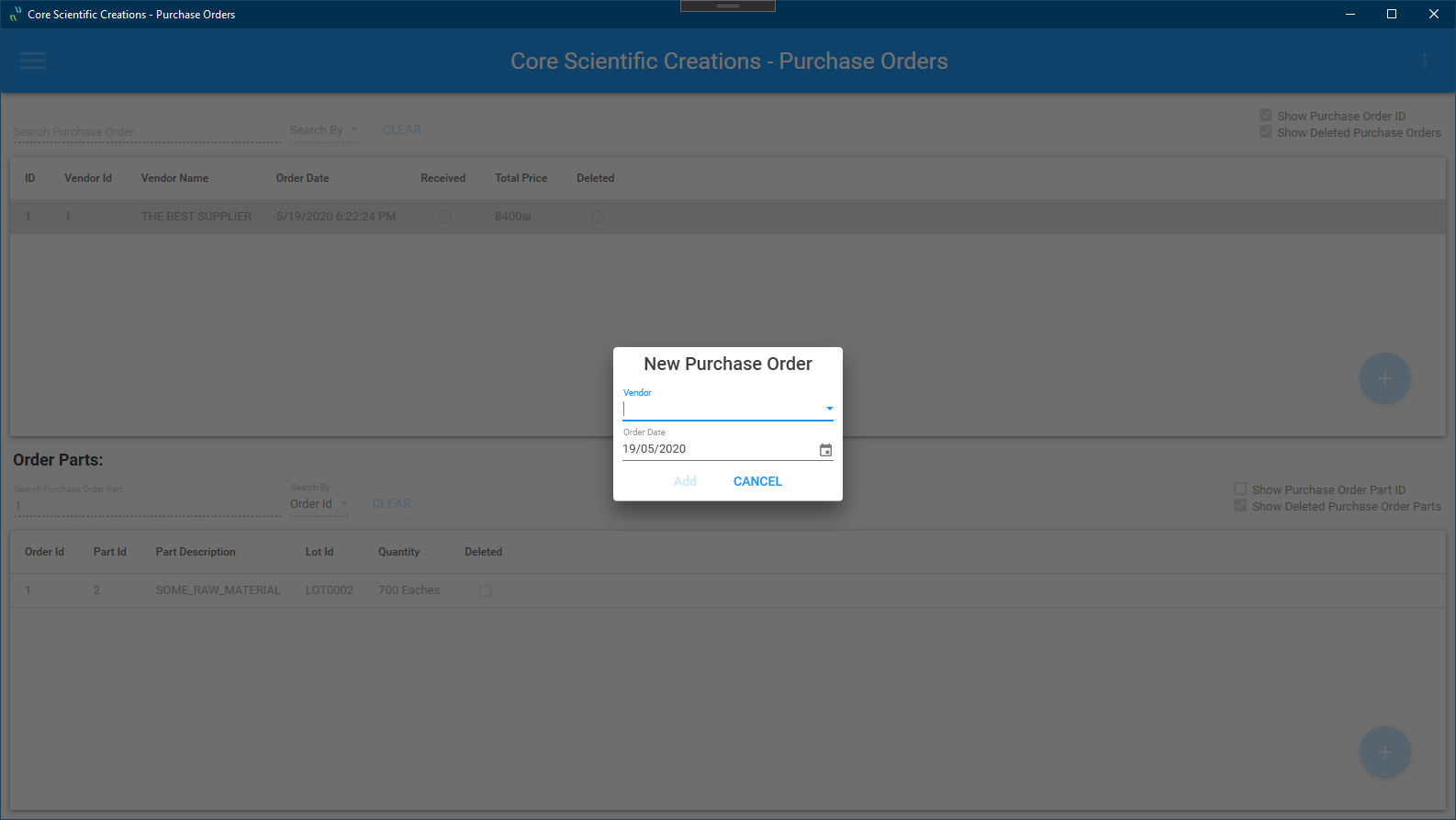
כמו כן, ניתן לראות כי במסך השני יש את אותן האופציות עבור הפריטים של אותה הזמנה שנבחרה.



### הוספת הזמנה מספק

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של מסך ההזמנות.

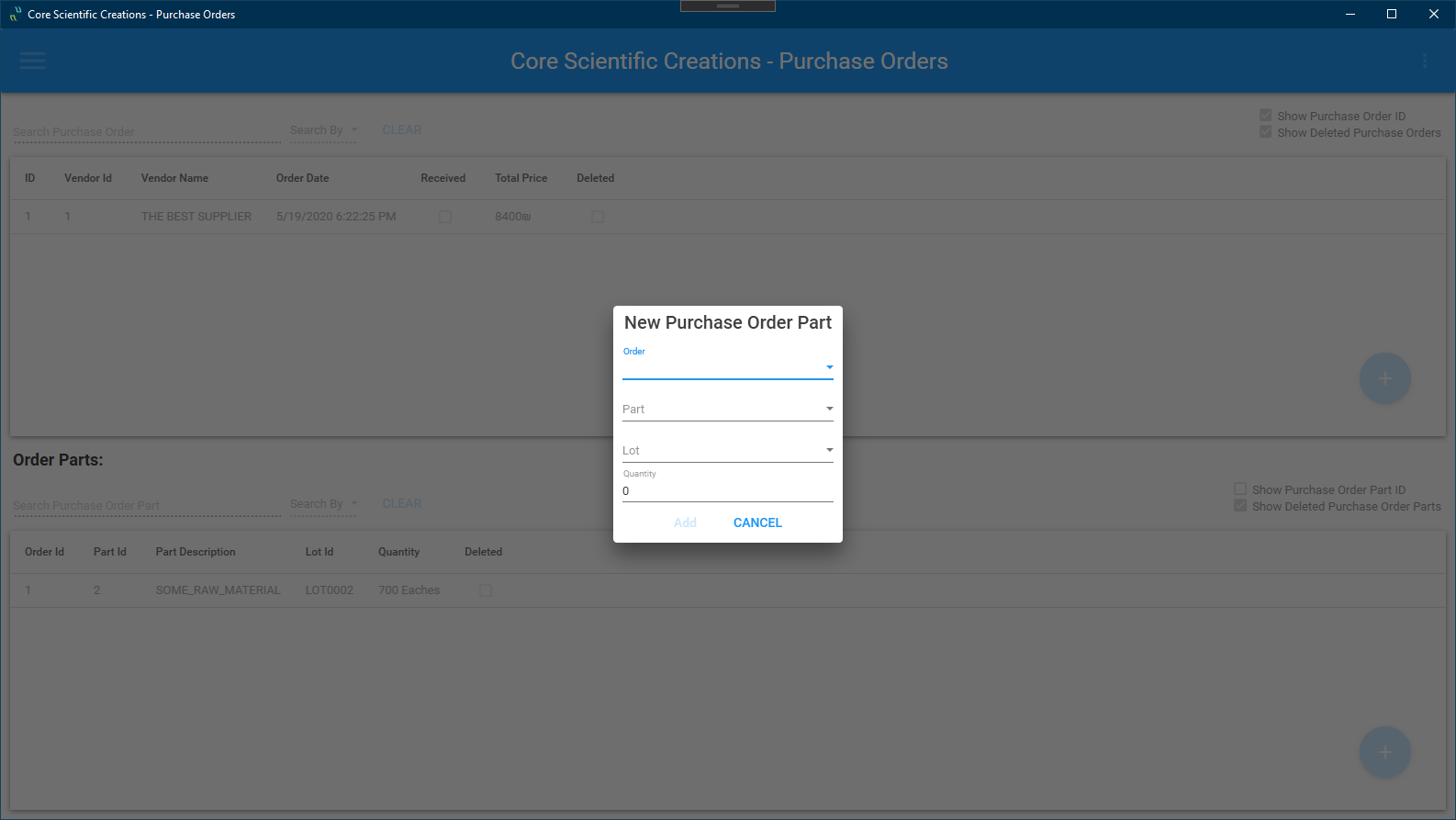
ניתן לראות כי במסך זה ניתן להוסיף הזמנה מספק חדשה לפי השדות השונים.



### הוספת פריט להזמנה מספק

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה של מסך הפריטים.

ניתן לראות כי במסך זה ניתן להוסיף פריט חדש להזמנה מספק לפי השדות השונים.

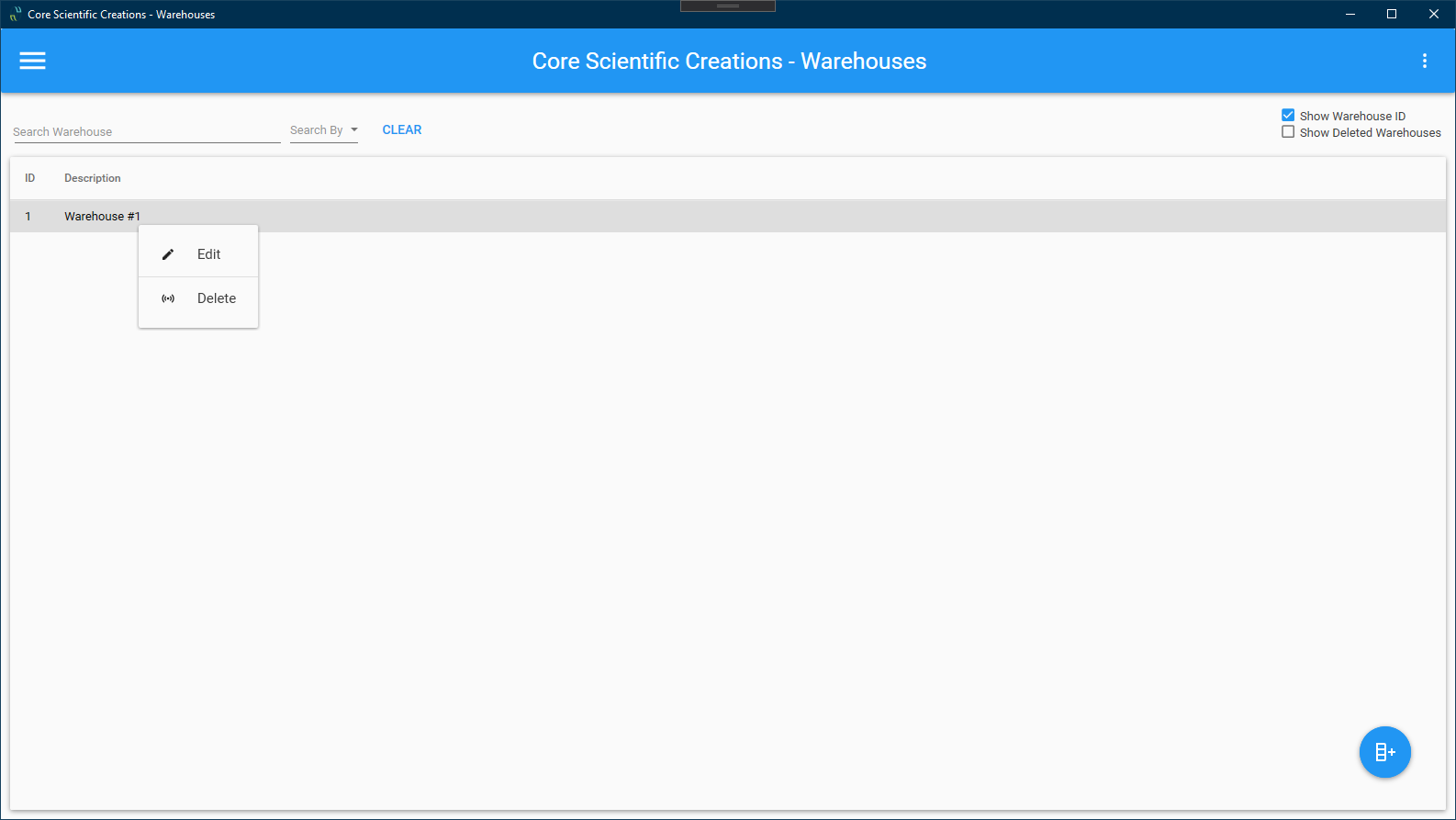


## ניהול מלאי

### הצגת מחסנים הקיימים במערכת

במסך זה ניתן לראות את כל המחסנים הקיימים במערכת.

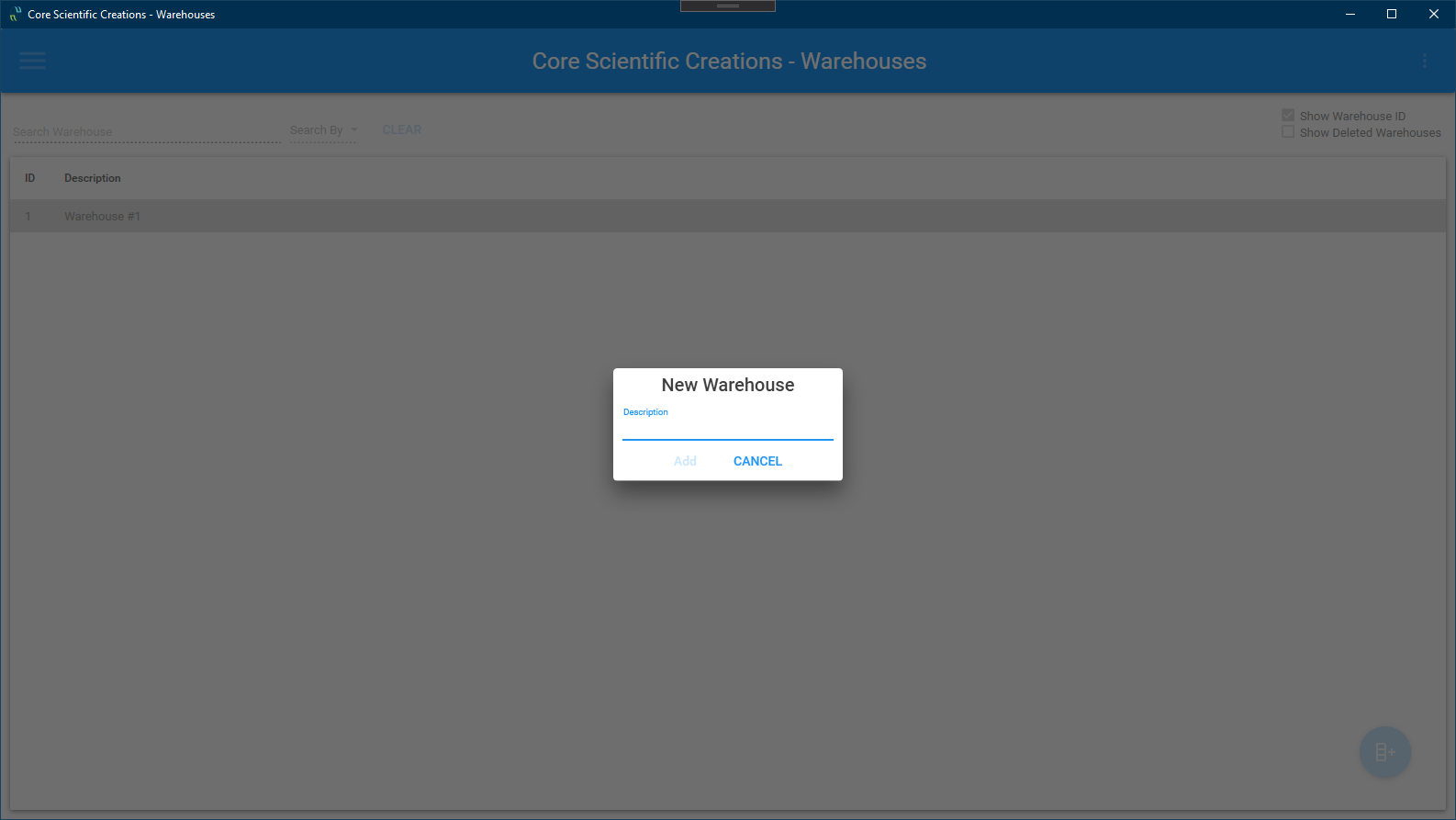
ניתן לראות כי קיימות אופציות שונות כמו חיפוש מחסנים לפי השדות השונים, הסתרת מספר המחסן והסתרת מחסנים מחוקים. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת מחסן חדש וכי לחיצת מקש ימני על אחד מהמחסנים תגרום לפתיחת תפריט המאפשר מחיקת המחסן או עריכת הפרטים שלו.



### הוספת מחסן חדש

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה.

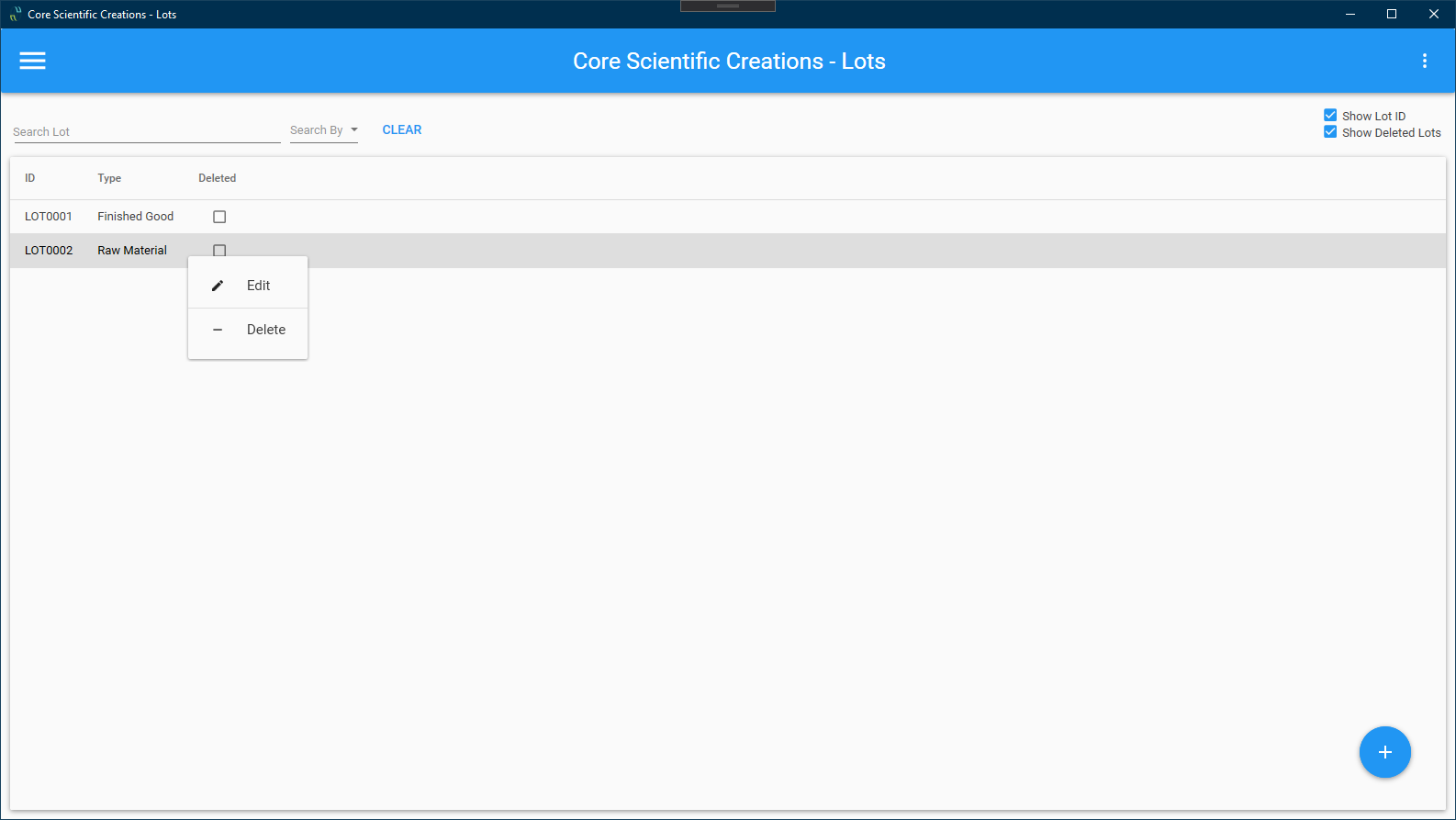
ניתן לראות כי במסך זה ניתן להוסיף מחסן חדש לפי השדות השונים.



### הצגת אצוות הקיימות במערכת

במסך זה ניתן לראות את כל האצוות הקיימות במערכת.

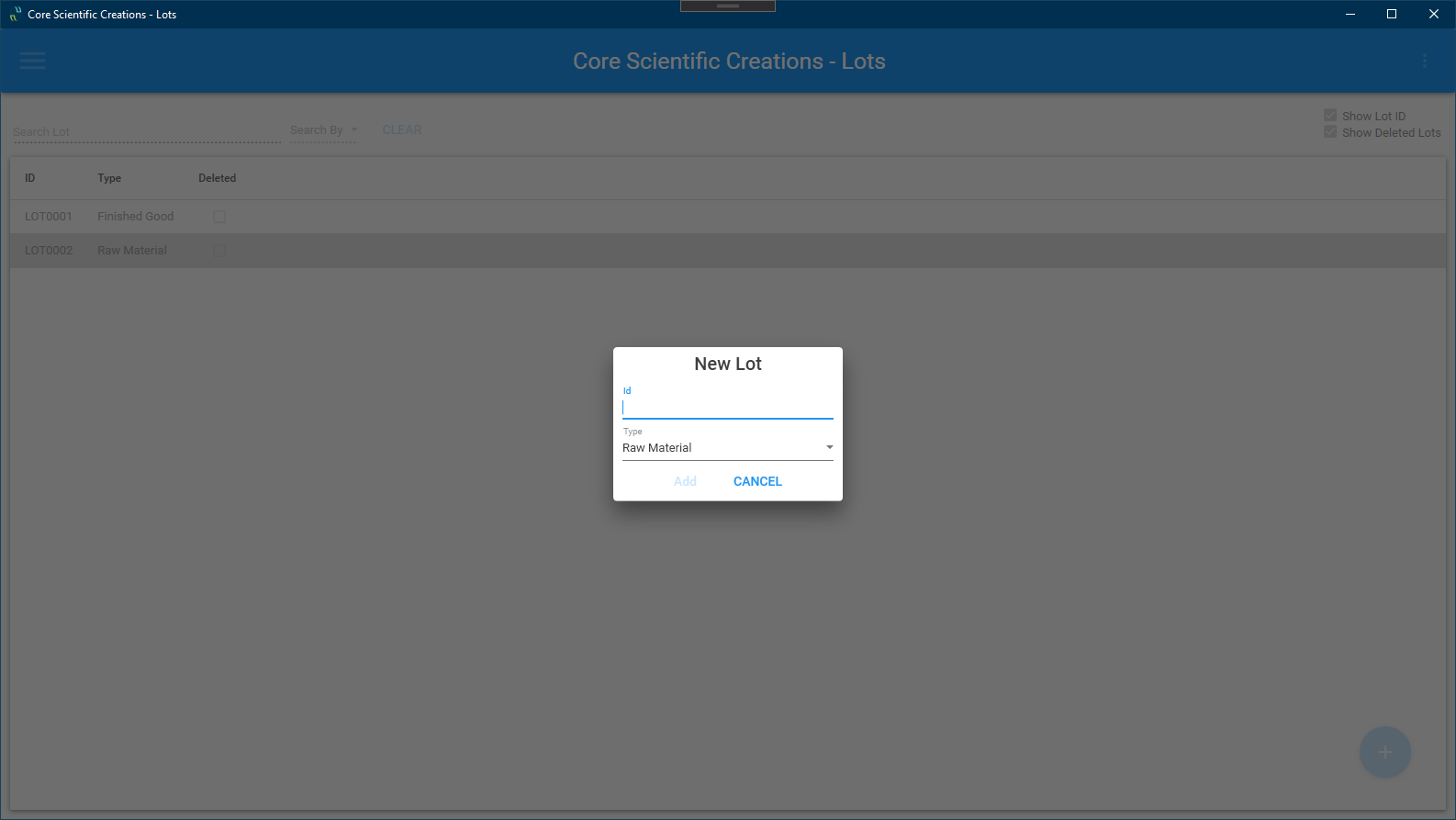
ניתן לראות כי קיימות אופציות שונות כמו חיפוש אצוות לפי השדות השונים, הסתרת מספר האצווה והסתרת אצוות מחוקות. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת אצווה חדש וכי לחיצת מקש ימני על אחד מהאצוות תגרום לפתיחת תפריט המאפשר מחיקת האצווה או עריכת הפרטים שלו.



### הוספת אצווה חדשה

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה.

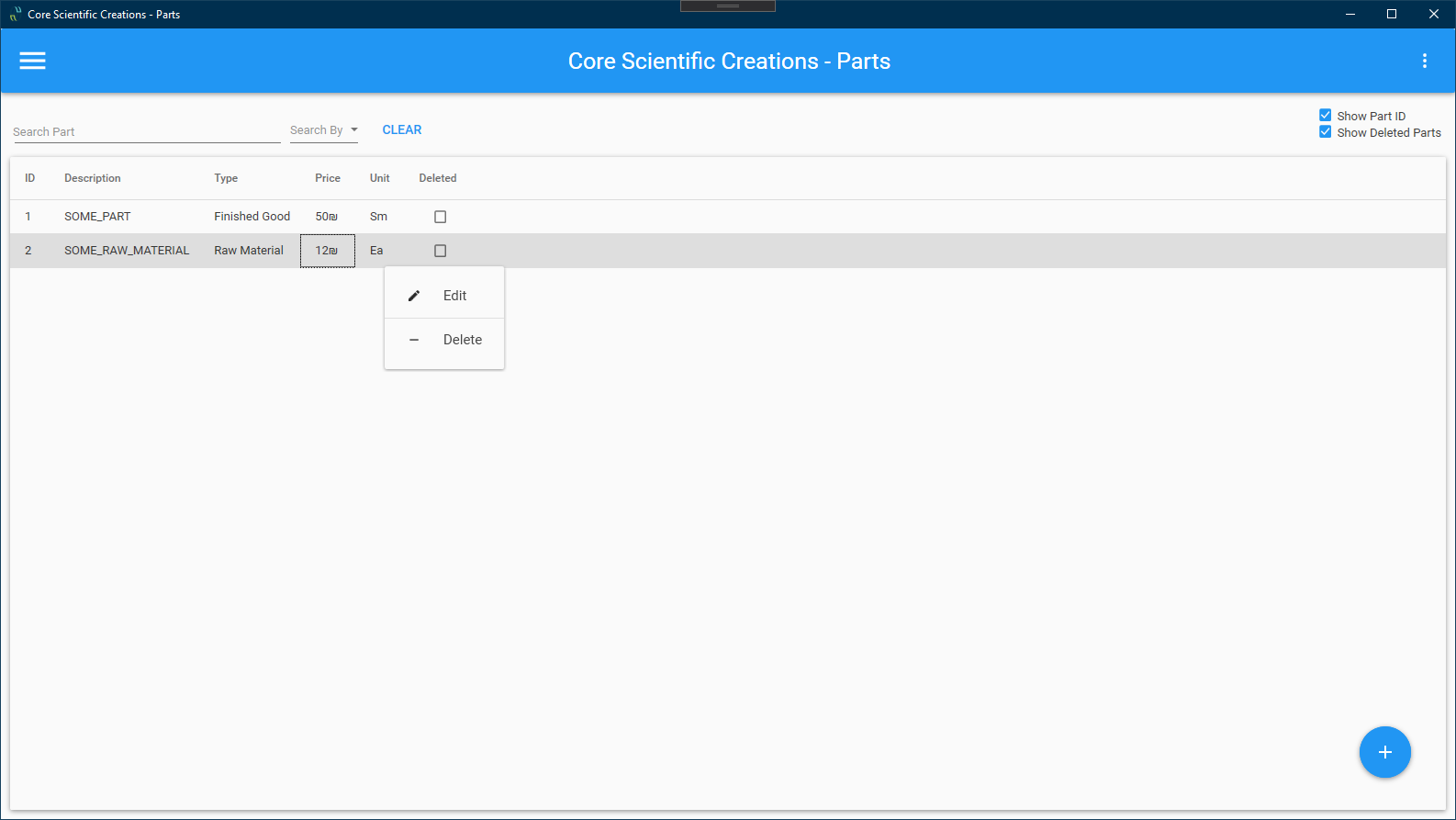
ניתן לראות כי במסך זה ניתן להוסיף אצווה חדשה לפי השדות השונים.



### הצגת הפריטים הקיימים במערכת

במסך זה ניתן לראות את כל הפריטים הקיימים במערכת.

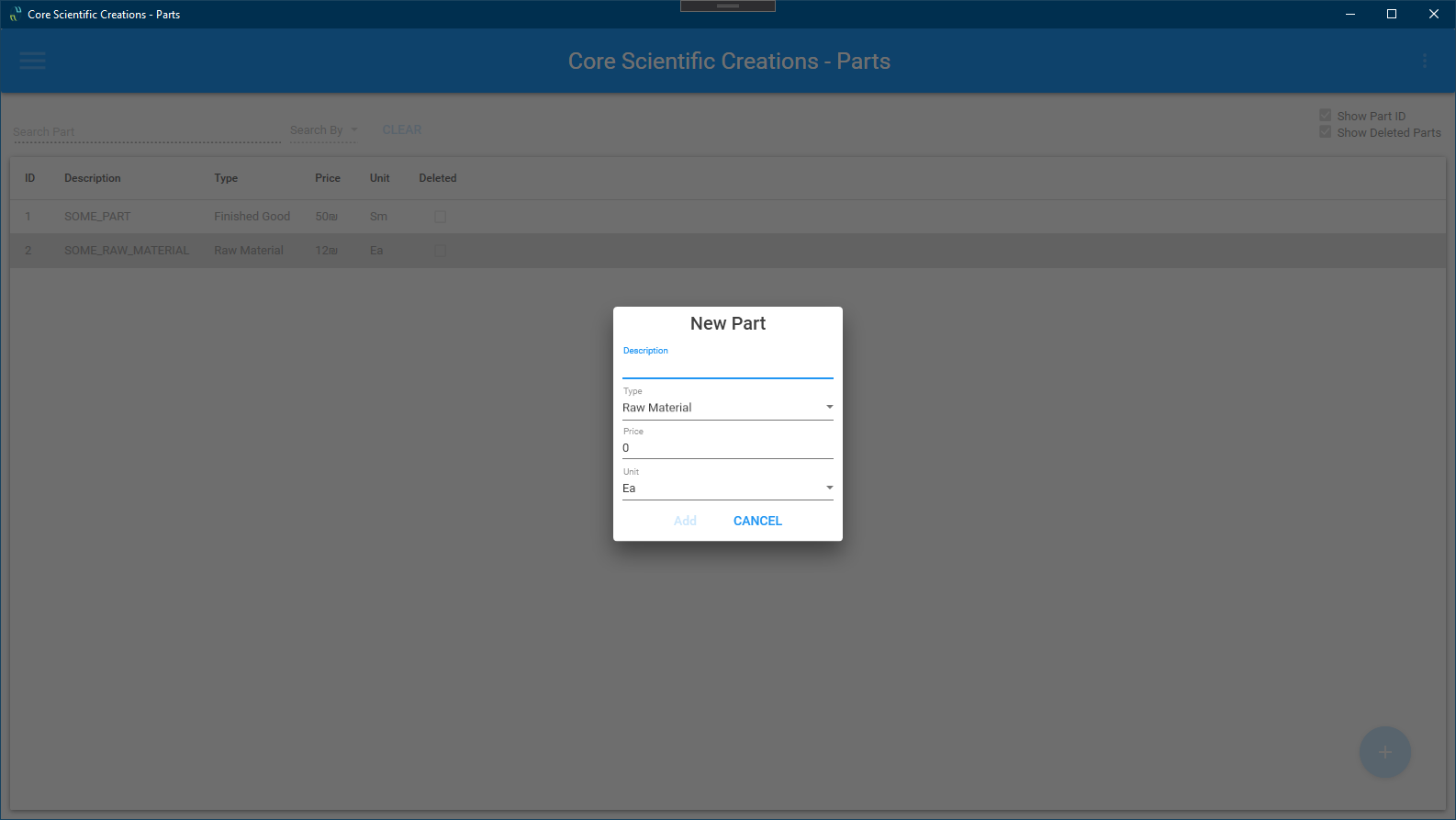
ניתן לראות כי קיימות אופציות שונות כמו חיפוש פריטים לפי השדות השונים, הסתרת מספר הפריט והסתרת פריטים מחוקים. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת פריט חדש וכי לחיצת מקש ימני על אחד מהפריטים תגרום לפתיחת תפריט המאפשר מחיקת הפריט או עריכת הפרטים שלו.



### הוספת פריט חדש

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה.

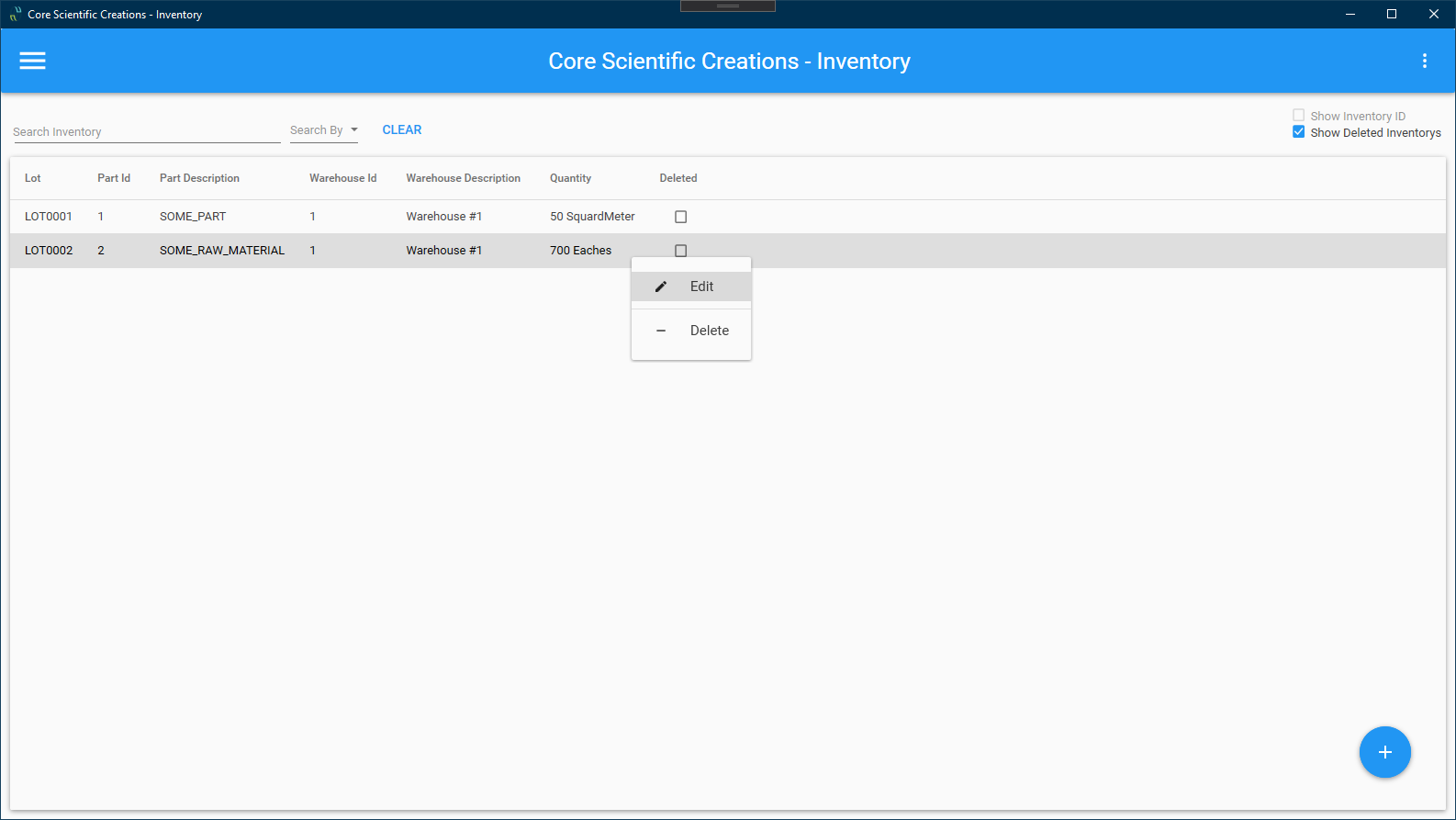
ניתן לראות כי במסך זה ניתן להוסיף פריט חדש לפי השדות השונים.



### הצגת כל המלאי הזמין

במסך זה ניתן לראות את כל המלאי הזמין במערכת.

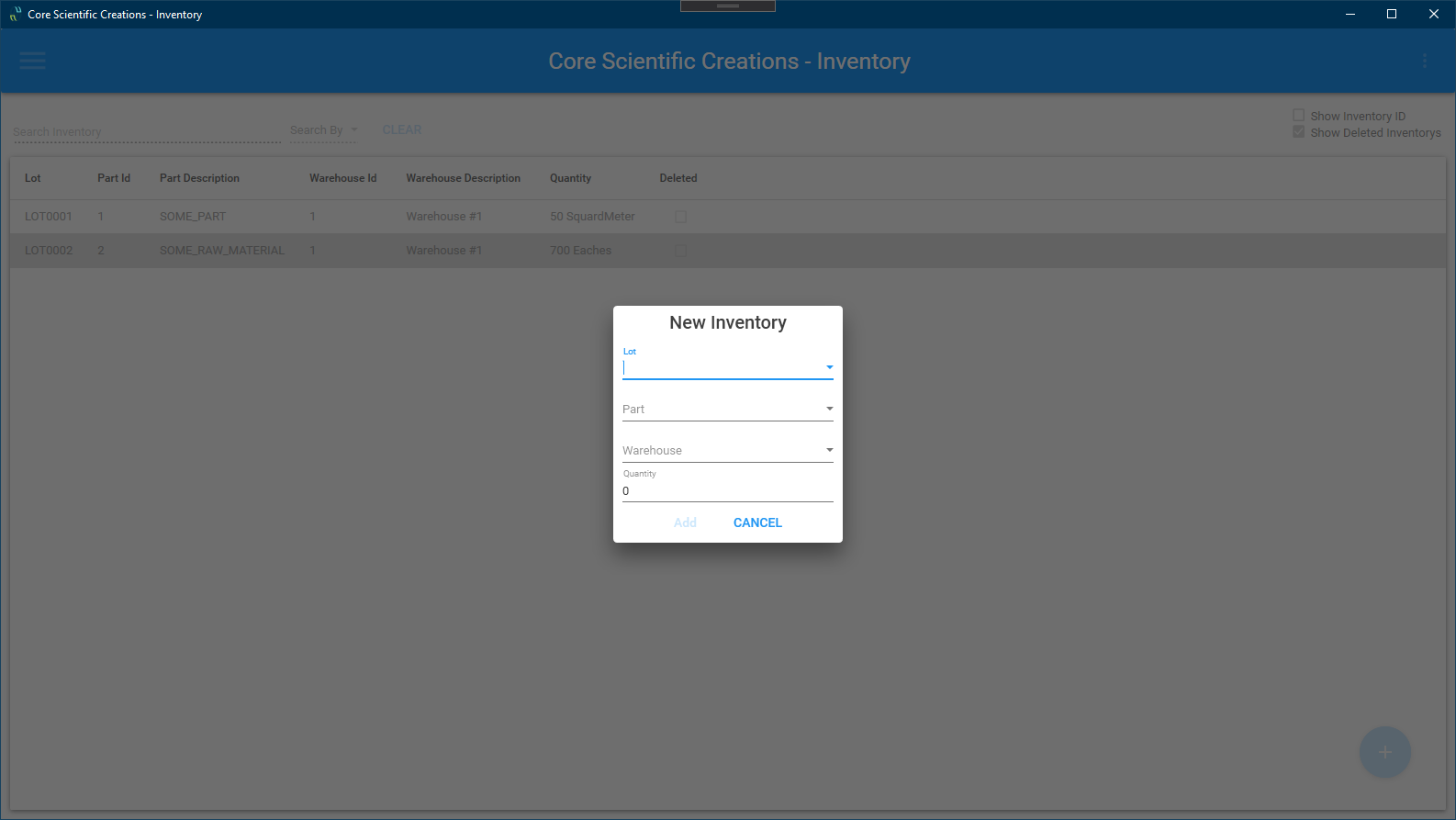
ניתן לראות כי קיימות אופציות שונות כמו חיפוש פריטים לפי השדות השונים והסתרת פריטים מחוקים. כמו כן, ניתן לראות כי ישנו כפתור המאפשר הוספת פריט חדש וכי לחיצת מקש ימני על אחד מהפריטים תגרום לפתיחת תפריט המאפשר מחיקת הפריט או עריכת הפרטים שלו.



### הוספת מלאי חדש

ניתן לפתוח את מסך זה באמצעות כפתור ההוספה בצד ימין למטה.

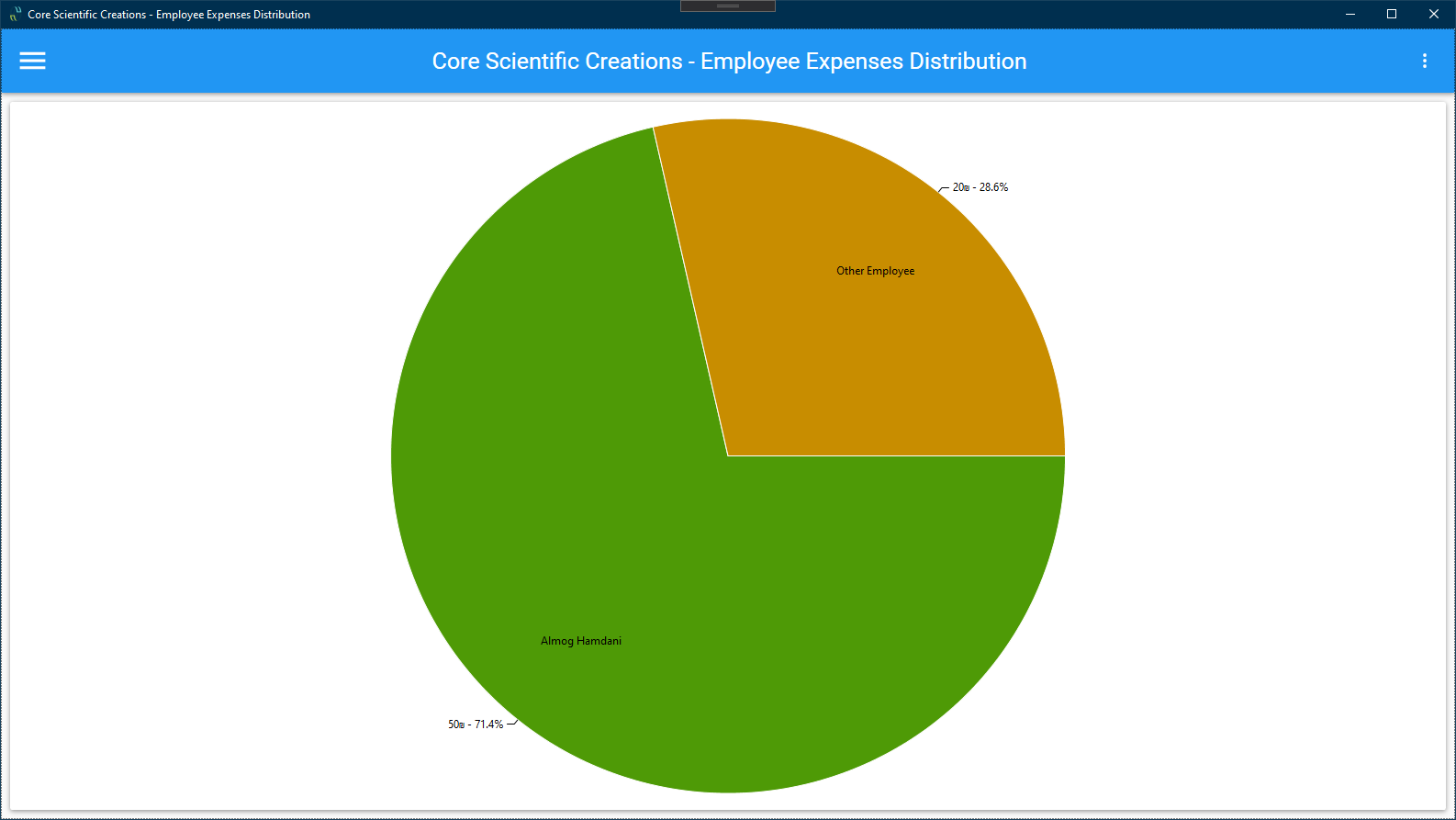
ניתן לראות כי במסך זה ניתן להוסיף מלאי חדש לפי השדות השונים.



## דו"חות

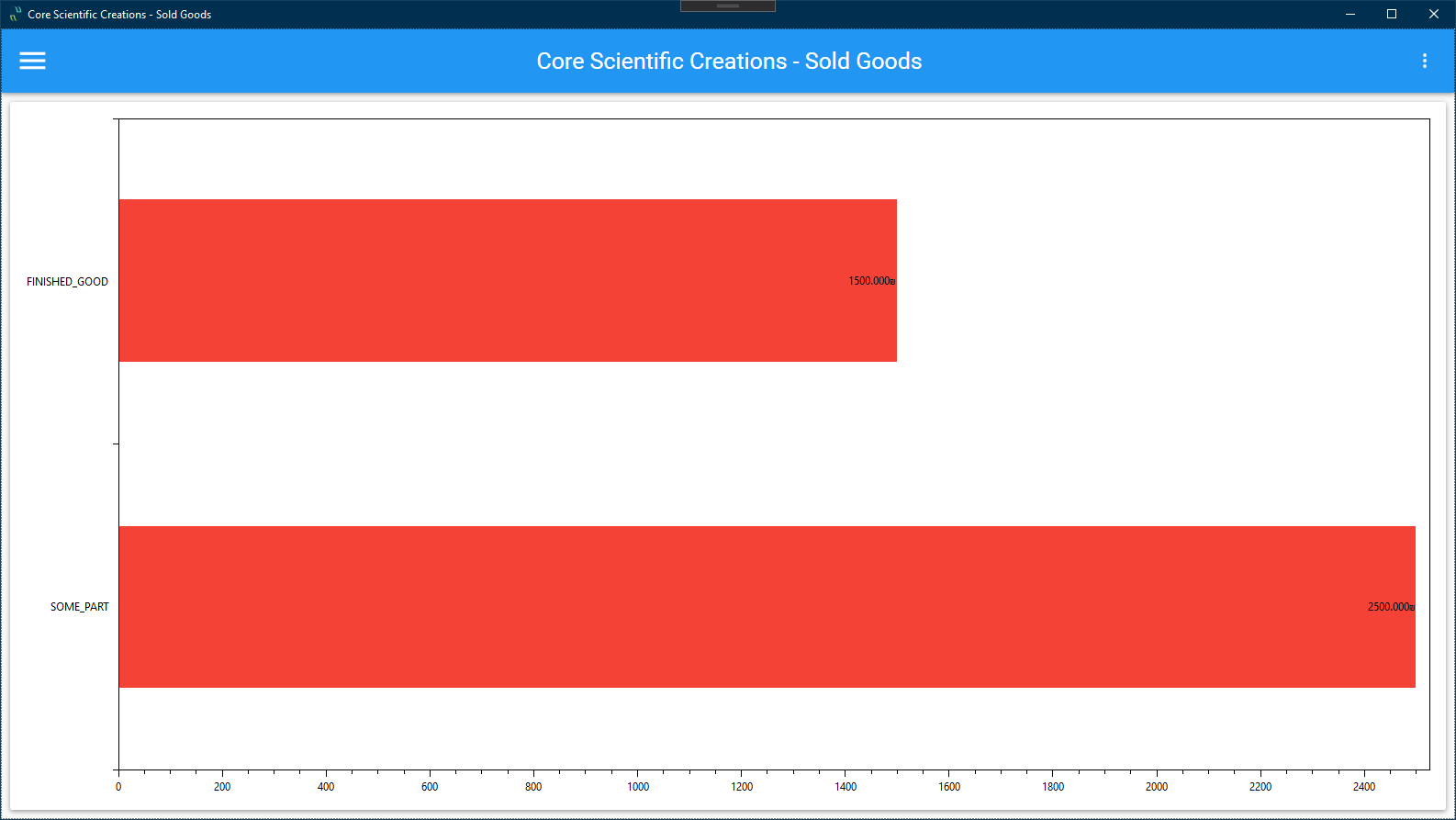
### דו"ח התפלגות הוצאות לא שוטפות בין עובדים

ניתן לראות כי במסך זה ישנו דו"ח המתואר באמצעות תרשים בצורת פאי המתאר את התפלגות ההוצאות הלא שוטפות של החברה לכל אורך תקופת קיום החברה. ה"חתיכות" השונות בתרשים מסמנים עובדים שונים האחראים על ההוצאות הלא שוטפות של החברה. על כל "חתיכה" מצוין את מספר הכסף שהעובד הוציא ואת האחוזים שההוצאות של עובד זה מהוות מתוך ההוצאות הלא השוטפות הכלליות של החברה.



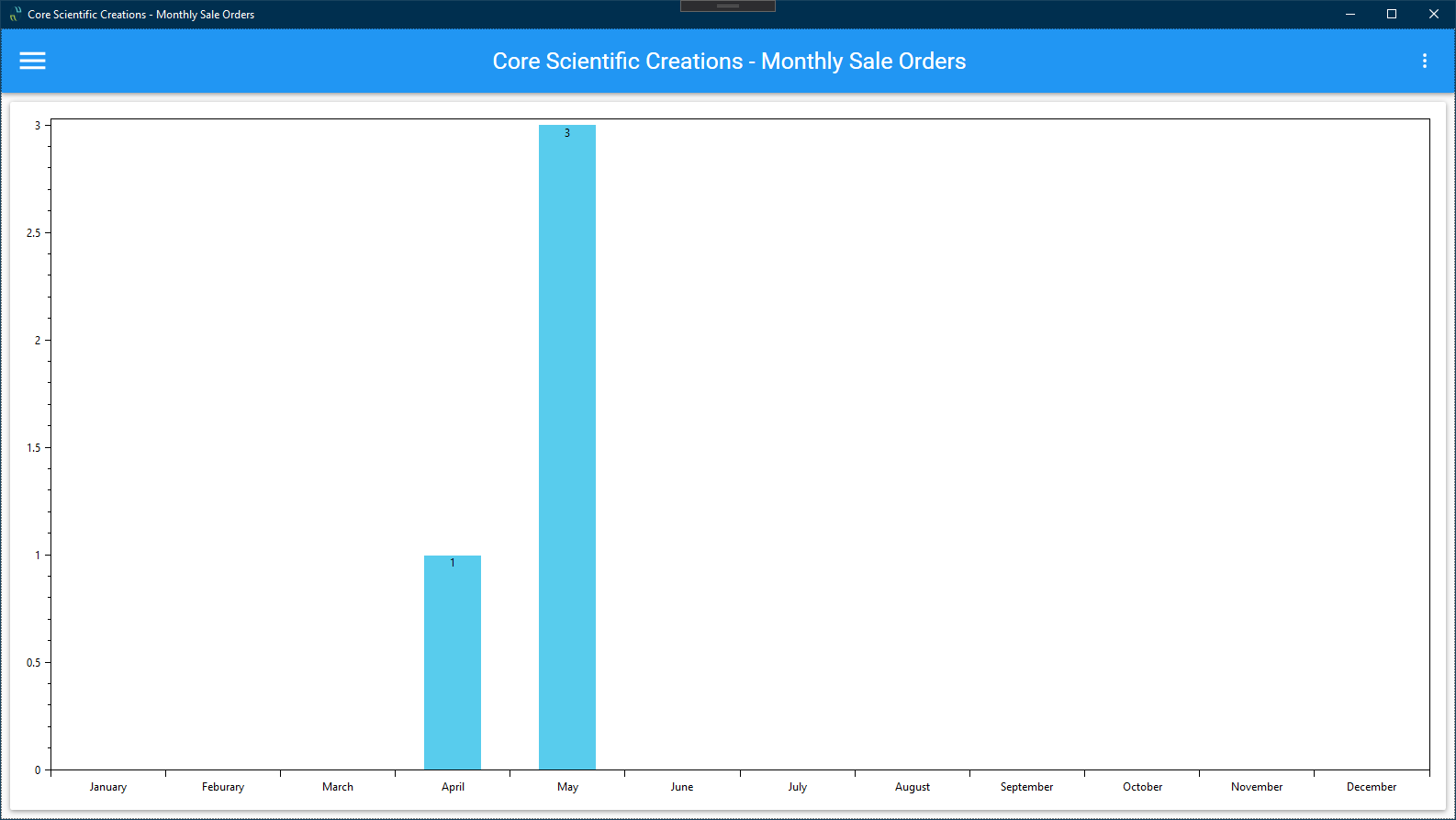
### דו"ח המוצרים הנמכרים ביותר

ניתן לראות כי במסך זה ישנו דו"ח המתואר באמצעות תרשים עמודות המתאר את המוצרים המוגמרים הנמכרים ביותר של החברה. ניתן לראות כי כל עמודה מסמנת פריט מוגמר של החברה וכי הערך שכתוב בתוך העמודה הוא הסכום שהחברה הרוויחה מהמחירה של המוצר ללקוחות.



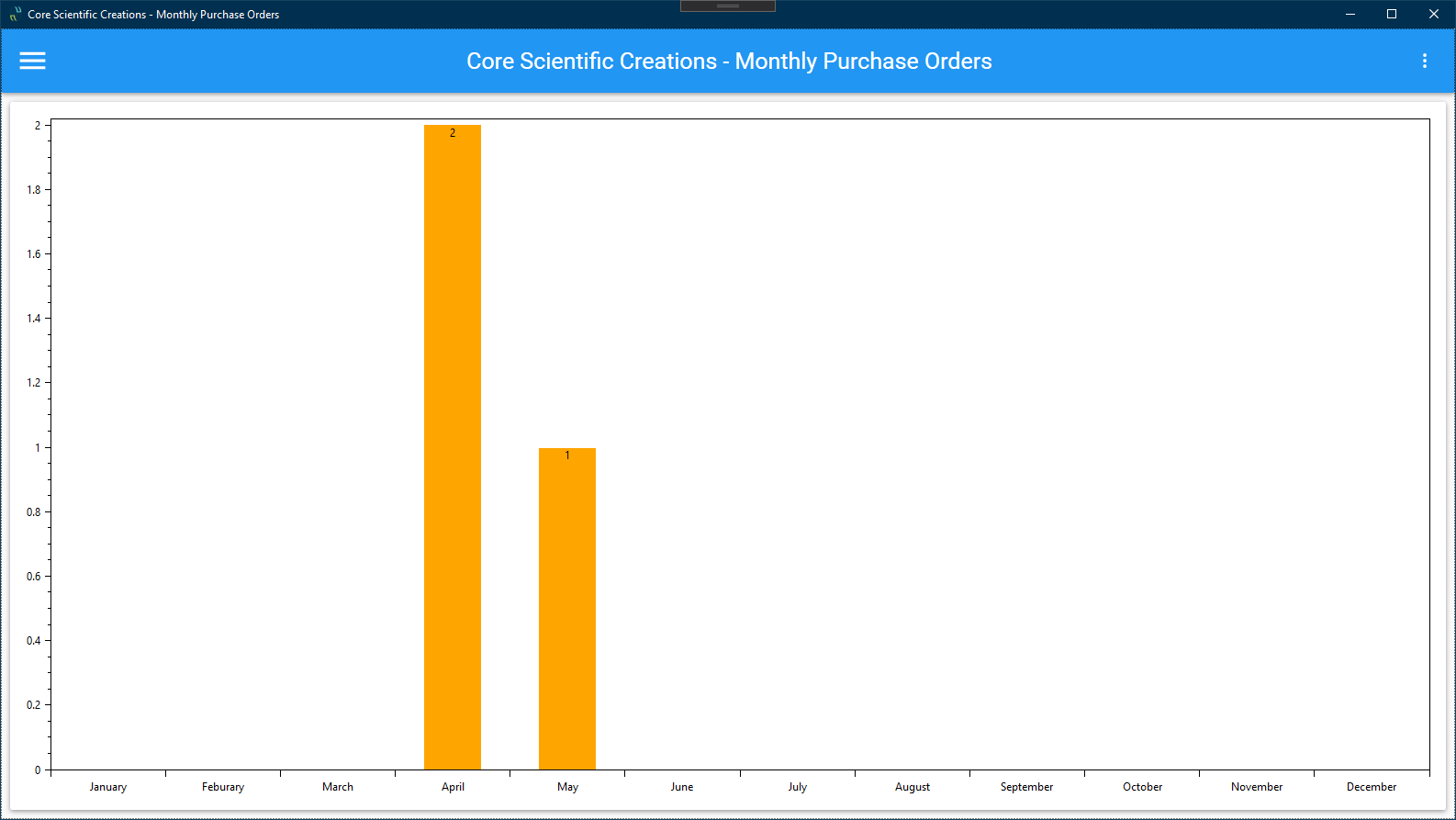
### דו"ח הזמנות לקוח חודשיות

במסך זה ניתן לראות דו"ח המתאר את כמות הזמנות לקוח שהחברה מוכרת בכל חודש במשך השנה האחרונה. ניתן לראות כי בציר האנכי מופיעים כל חודשי השנה וכי כל עמודה מתארת את כמות הזמנות הלקוח שהיה באותו החודש.



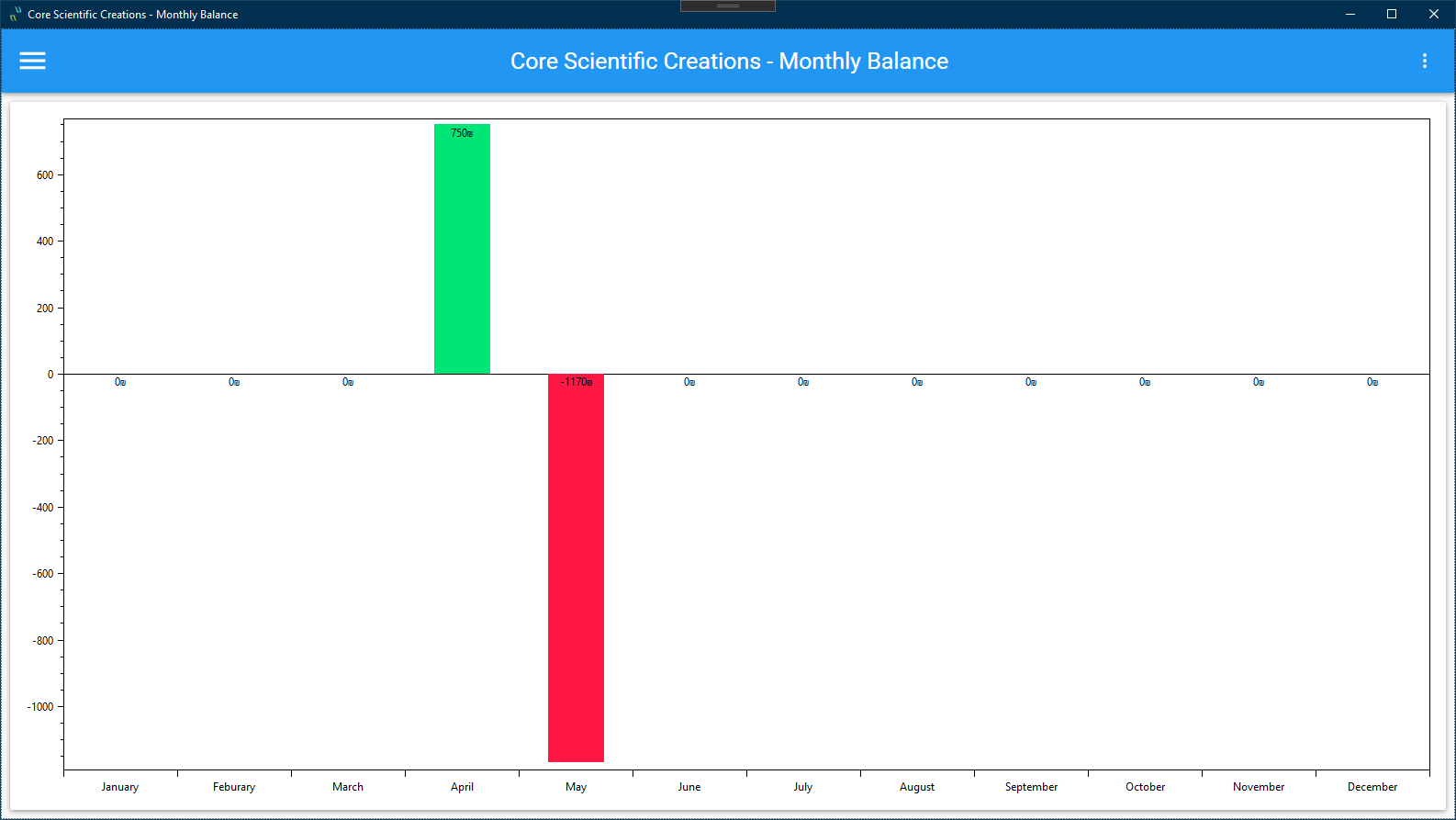
### דו"ח הזמנות מספק חודשיות

במסך זה ניתן לראות דו"ח המתאר את כמות ההזמנות מספק שהחברה קונה בכל חודש במשך השנה האחרונה. ניתן לראות כי בציר האנכי מופיעים כל חודשי השנה וכי כל עמודה מתארת את כמות ההזמנות מספק שהיה באותו החודש.



### דו"ח מאזן כספים חודשי

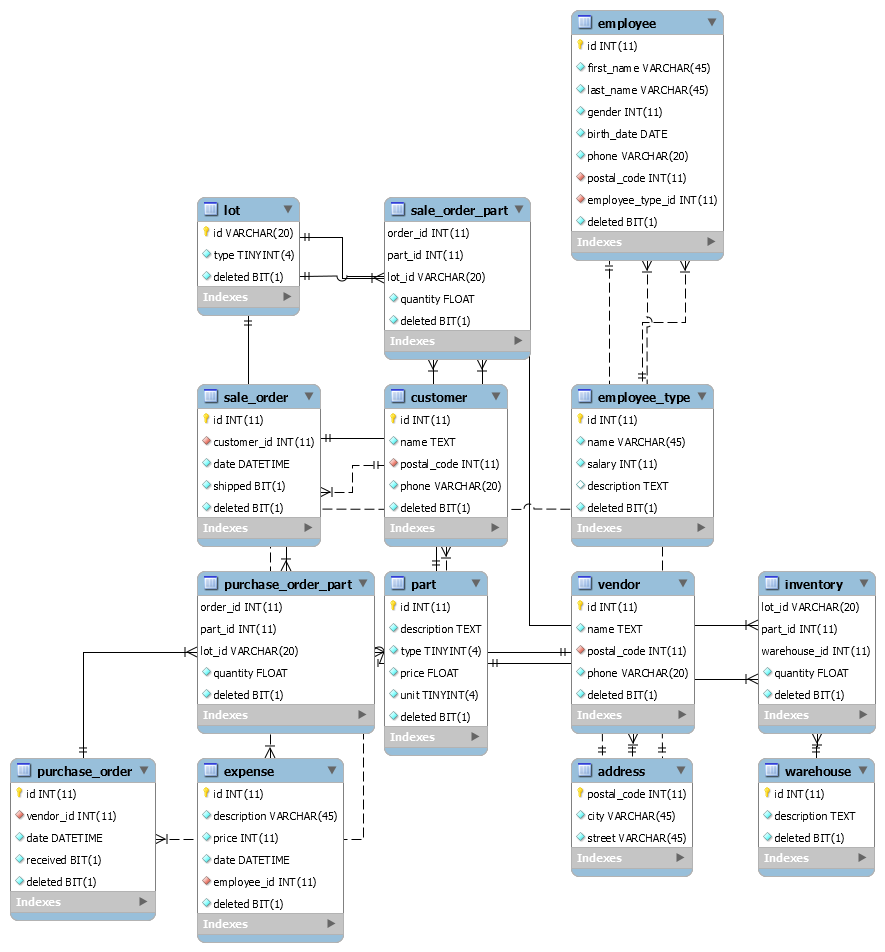
במסך זה ניתן לראות דו"ח מאזן כספים חודשי במשך השנה האחרונה המוצג באמצעות גרף עמודות. ניתן לראות כי בציר האנכי מופיעים כל חודשי השנה וכי כל עמודה מתארת את המאזן החודשי של אותו החודש. ניתן לראות כי צבע העמודה נקבע על פי האם המאזן החודשי של החברה חיובי או שלילי. המאזן החודשי נקבע על פי כמות הכסף שהחברה הרוויחה מהזמנות לקוח, כמות הכסף שהחברה הוציאה על הזמנות מספקים ומהוצאות לא שוטפות של החברה.



# בסיס הנתונים

## דיאגרמת מבנה הנתונים

בתרשים זה ניתן לראות את הטבלות השונות הנמצאות במבנה הנתונים:



## טבלת employee

* Id – המספר הסידורי של העובד (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* first\_name – השם פרטי של העובד.
* last\_name – השם משפחה של העובד.
* gender – המגדר של העובד (0 עבור זכר, 1 עבור נקבה).
* birth\_date – התאריך לידה של העובד.
* phone – מספר הטלפון של העובד.
* postal\_code – המיקוד של העובד. שדה זה הוא מצביע ל postal\_code של הטבלה address.
* employee\_type\_id – המספר הסידורי של סוג העובד. שדה זה הוא מצביע ל id של הטבלה employee\_type.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת employee\_type

* Id – המספר הסידורי של הסוג עובד (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* name – השם של הסוג עובד.
* salaray – המשכורת החודשית של הסוג עובד.
* description – תיאור סוג העובד.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת customer

* Id – המספר הסידורי של הלקוח (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* name – השם של הלקוח.
* postal\_code – המיקוד של הלקוח. שדה זה הוא מצביע ל postal\_code של הטבלה address.
* phone – מספר הטלפון של הלקוח.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת vendor

* Id – המספר הסידורי של הספק (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* name – השם של הספק.
* postal\_code – המיקוד של הספק. שדה זה הוא מצביע ל postal\_code של הטבלה address.
* phone – מספר הטלפון של הספק.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת part

* Id – המספר הסידורי של הפריט (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* description – תיאור הפריט.
* type – סוג הפריט (תוצרת גמורה או חומר גלם).
* price- מחיר הפריט ליחידה אחת.
* unit – היחידות בהן נמדד הפריט (ליטרים, מטר מרובע ויחידות).
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת expense

* Id – המספר הסידורי של ההוצאה הלא שוטפת (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* description – תיאור ההוצאה הלא שוטפת.
* price – המחיר של ההוצאה.
* date – התאריך בו התרחשה ההוצאה.
* employee\_id – המספר הסידורי של העובד שביצע את ההוצאה. שדה זה מצביע ל id של הטבלה employee.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת lot

* id – המספר אצווה (רק במקרה של אצווה זה מחרוזת).
* type - סוג האצווה (תוצרת גמורה או חומר גלם).
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת address

* postal\_code – המיקוד של הכתובת.
* city – העיר של הכתובת.
* street – הרחוב של הכתובת.

## טבלת warehouse

* Id – המספר הסידורי של ההוצאה הלא שוטפת (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* description – תיאור המחסן (שם המחסן).
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת inventory

* lot\_id – המספר הסידורי של האצווה בה נמצא הפריטים. שדה זה מצביע ל id של הטבלה lot.
* part\_id – המספר הסידורי של הפריט של המלאי. שדה זה מצביע ל id של הטבלה part.
* warehouse\_id – המספר הסידורי של המחסן בו נמצאים הפריטים. שדה זה מצביע ל id של הטבלה warehouse.
* quantity – כמות הפריטים.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת sale\_order

* Id – המספר הסידורי של ההזמנה (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* customer\_id – המספר הסידורי של הלקוח שביצע את ההזמנה. שדה זה מצביע ל id של הטבלה customer.
* date – התאריך בו התרחשה ההזמנה.
* shipped – שדה זה מסמן האם ההזמנה נשלחה אל הלקוח והפריטים שלה ירדו מהמלאי.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלת purchase\_order

* Id – המספר הסידורי של ההזמנה (מיוצר באופן אוטומטי על ידי מסד הנתונים).
* vendor\_id – המספר הסידורי של הספק שממנו בוצעה ההזמנה. שדה זה מצביע ל id של הטבלה vendor.
* date – התאריך בו התרחשה ההזמנה.
* recieved – שדה זה מסמן האם ההזמנה התקבלה באחד מהמחסנים של החברה והפריטים של ההזמנה נכנסו למלאי.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלה sale\_order\_part

* order\_id – המספר הסידורי של ההזמנה. שדה זה מצביע ל id של הטבלה sale\_order.
* part\_id – המספר הסידורי של הפריט בהזמנה. שדה זה מצביע ל id של הטבלה part.
* lot\_id – המספר אצווה. שדה זה מצביע ל id של הטבלה lot.
* quantity – כמות הפריטים.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

## טבלה sale\_order\_part

* order\_id – המספר הסידורי של ההזמנה. שדה זה מצביע ל id של הטבלה sale\_order.
* part\_id – המספר הסידורי של הפריט בהזמנה. שדה זה מצביע ל id של הטבלה part.
* lot\_id – המספר אצווה. שדה זה מצביע ל id של הטבלה lot.
* quantity – כמות הפריטים.
* deleted – שדה זה מסמן האם הרשומה מחוקה או לא.

# מדריך למפתח

## ספריות ליבה

את הפרויקט הזה בניתי באמצעות מספר Frameworks ששילובם אפשר את הבנייה של פרויקט זה.

### WPF

Windows Presentation Foundation (בקיצור WPF) הוא מודל וכלי פיתוח תוכנה מבית מיקרוסופט למערכת ההפעלה חלונות, עם דגש על עיצוב גרפי מתקדם של ממשק המשתמש. WPF הינו הכלי בו השתמשתי על מנת לפתח את ה UI של הפרויקט ובעזרתו בניתי ממשק נוח למשתמש.

### MVVM

Model View ViewModel או MVVM היא תבנית ארכיטקטורה בהנדסת תוכנה שתוכננה על ידי מיקרוסופט.

עקרונות התבנית הם מבנה של:

* + **Model** - מודל מתאר על האובייקטים שמהווים את ה-Data של התוכנית.
  + **View** - החלק האחראי על הגרפיקה וממשק המשתמש.
  + **ViewModel** - החלק שמהווה רכיב תקשורת בין החלקים ומחזיק רכיבי לוגיקה של UI.

MVVM היא הארכיטקטורה שבעזרתה בניתי את הפרויקט הזה והיא תרמה לי רבות בפיתוח יעיל של הפרויקט. זאת משום שהארכיטקטורה מתאימה לפיתוח UI באמצעות Xaml (כמו ש WPF עושה) ולכן היא התאימה לשימוש בפרויקט זה.

### Material Design

Material Design (או Quantum Paper) היא שפת עיצוב שמפותחת על ידי גוגל. את פרויקט זה פיתחתי באמצעות הספרייה *Material Design in XAML* אשר אפשרה לי לפתח את הפרויקט באמצעות שפת העיצוב המתקדמת של גוגל ובכך העלתה לי את הרמה של ממשק המשתמש.

### OxyPlot

OxyPlot היא ספרייה המאפשרת הצגת תרשימים שונים (תרשים פאי, תרשים עמודות וכו') ב WPF. אני השתמשתי בספרייה זו על מנת לבנות את הדו"חות השונים בפרויקט.

## מרכיבי ליבה

פרויקט זה בנוי על בסיס מספר מרכיבי ליבה המאפשרים את הפיתוח המהיר והיעיל של הפרויקט.

### DataHandler

הפרויקט בנוי על בסיס המחלקה IDataHandler שמציעה ממשק גנרי לביצוע פעולות על מופעים של האובייקטים השונים כמו: הוספה מופע חדש למסד הנתונים, עדכון מופע קיים במסד הנתונים, מחיקה של מופע במסד הנתונים, שליפת כל המופעים ממסד הנתונים וכו'. בפרויקט קיים תת-מחלקה עבור כל אובייקט אשר יורש מ IDataHandler ומתאים את הפעילות של המחלקה לאובייקטים השונים. כמו כן, במחלקה זו מתממש הבדיקות על תקינות התוכן שהמשתמש מכניס. באמצעות שימוש במחלקה גנרית זו אני חוסך בכתיבת קוד כפול ומייעל את אופן הפעולה של הפרויקט.

### DataTableViewModel

מחלקה זו מהווה את ה ViewModel (החלק שמהווה רכיב תקשורת בין החלקים ומחזיק רכיבי לוגיקה של UI) הבסיסי שממנו יורשים כל ה ViewModels. מחלקה זו היא מחלקה גנרית המאפשרת ייצוג מלא ופעולות שונות על האובייקט הגנרי (כמו: הצגת המופעים של האובייקט, הוספה, מחיקה, עריכה, חיפוש וכו'). כמו כן, מחלקה זו מקבלת את הדיאלוג המיוחד של אותו האובייקט (באמצעות הדיאלוג ניתן להוסיף ולערוך מופעים של האובייקט). באמצעות שימוש במחלקה גנרית זו אני חוסך בכתיבת קוד כפול ומייעל את אופן הפעולה של הפרויקט.

### DualDataTableViewModel

מחלקה זו מהווה הרחבה לDataTableViewModel בכך שהיא מאפשרת הצגה של שני DataTableView אחד מעל השני. מחלקה בסיסית זו שימושית כאשר בניתי את ההזמנות לקוח ואת ההזמנות מספק.

## מחלקות הפרויקט

### DataTableViewModel

כפי שתיארתי במרכיבי ליבה, מחלקה זאת אחראית על הצגת המופעים של האובייקט באופן גנרי ואפשור ביצוע פעולות שונות על האובייקט.

|  |
| --- |
| using Caliburn.Micro;  using MaterialDesignThemes.Wpf;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data.Entity.Core.Objects;  using System.Data.Entity.Infrastructure;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  public class DataItemSelectedEventArgs<T> : EventArgs  {  public T DataItem { get; set; }  }  abstract class DataTableViewModel<T, U, V> : Screen, INotifyPropertyChanged where T : class, new() where U : Interfaces.IDataHandler<T>, new() where V : Dialogs.Dialog, new()  {  protected Interfaces.IDataHandler<T> dataHandler = new U();  public string DataItemName { get; } = Regex.Replace(typeof(T).Name, "(\\B[A-Z])", " $1");  public virtual bool HasId { get; set; } = true;  public virtual bool ShowDataItemId { get; set; } = false;  public bool ShowDeletedItems { get; set; } = false;  public List<Misc.Column> SearchableColumns { get => GetColumns().FindAll(column => column.AllowSearch); }  public int SearchColumnIndex { get; set; } = -1;  public string SearchText { get; set; }  public event EventHandler<DataItemSelectedEventArgs<T>> ItemSelected;  public event EventHandler DataUpdated;  public List<T> Data  {  get  {  List<T> OriginalData = ShowDeletedItems ? dataHandler.GetData() : dataHandler.GetData().FindAll(item => (bool)typeof(T).GetProperty("Deleted").GetValue(item) == false);  // If no search text or column, return original data  if (SearchText == null || SearchColumnIndex == -1 || SearchText.Length == 0)  {  return OriginalData;  }  else  {  Misc.Column searchColumn = SearchableColumns[SearchColumnIndex];  return OriginalData.FindAll(item => Misc.Utils.GetPropertyValue(typeof(T), searchColumn.PropertyBinding is Binding ? ((Binding)searchColumn.PropertyBinding).Path.Path : ((Binding)((MultiBinding)searchColumn.PropertyBinding).Bindings[0]).Path.Path, item).ToString().Contains(SearchText));  }  }  }  public virtual string AddButtonIcon { get; set; } = "Add";  public virtual string RemoveButtonIcon { get; set; } = "Remove";  public virtual bool HasDataContextMenuExtraButton { get; } = false;  public virtual string DataContextMenuExtraButtonText { get; }  public virtual string DataContextMenuExtraButtonIcon { get; }  public virtual bool DataContextMenuExtraButtonEnabled { get; } = true;  protected override void OnViewReady(object view)  {  base.OnViewReady(view);  DataGrid dataGrid = (view as Views.DataTableView).Data;  // Add each column to the data grid  foreach (var column in GetColumns())  {  // Check for checkbox column  if (column.PropertyBinding is Binding && Misc.Utils.FollowPropertyPath(typeof(T), ((Binding)column.PropertyBinding).Path.Path) == typeof(bool))  {  // Insert before the deleted field  dataGrid.Columns.Insert(dataGrid.Columns.Count - 1, new DataGridCheckBoxColumn { Header = column.Name, Binding = column.PropertyBinding, ElementStyle = Application.Current.Resources["MaterialDesignDataGridCheckBoxColumnStyle"] as Style });  }  else  {  // Insert before the deleted field  dataGrid.Columns.Insert(dataGrid.Columns.Count - 1, new DataGridTextColumn { Header = column.Name, Binding = column.PropertyBinding });  }  }  }  public bool CanChangeDataItemDeleteStatus(T dataItem)  {  // Check if the data item isn't null  return dataItem != null;  }  public void ChangeDataItemDeleteStatus(T dataItem)  {  // If the item isn't deleted yet  if ((bool)typeof(T).GetProperty("Deleted").GetValue(dataItem) == false)  {  // Remove the data item from the data set  dataHandler.RemoveDataItem(dataItem);  } else  {  // Restore the data item to the data set  dataHandler.RestoreDataItem(dataItem);  }  // Update the table  UpdateTable();  }  public bool CanEditDataItem(T dataItem)  {  // Check if the data item isn't null  return dataItem != null;  }    public void EditDataItem(T dataItem)  {  // Show the data item dialog of the item  ShowDataItemDialog(dataItem);  }  public async void ShowDataItemDialog(T dataItem)  {  bool update = dataItem != null;  V dataItemDialog = null;  // If the dialog isn't for updating, init the data item  if (!update) {  InitDataItem(ref dataItem);  }  // Initialize the data item dialog and the new item object  InitDataItemDialog(ref dataItemDialog, ref dataItem);  // Set the dialog as update dialog if needed  dataItemDialog.UpdateDialog = update;  // Show the data item dialog  await DialogHost.Show(dataItemDialog, (object \_, DialogClosingEventArgs args) =>  {  // Check if the dialog closing is a message dialog  if (args.Session.Content.GetType() == typeof(Dialogs.MessageDialog))  {  // Cancel the close of the dialog  args.Cancel();  // Show the data item dialog again  args.Session.UpdateContent(dataItemDialog);  }  else  {  // If not cancelled  if (args.Parameter.GetType() == typeof(bool) && (bool)args.Parameter)  {  try  {  if (update)  {  // Update the data item  UpdateDataItem(dataItem);  } else  {  // Add the data item  AddDataItem(dataItem);  }  }  catch (Exception ex)  {  // Cancel the close of the dialog  args.Cancel();  // Show the error  args.Session.UpdateContent(new Dialogs.MessageDialog { Message = ex.Message });  }  // Update the data table  UpdateTable();  } else  {  // Refresh the entity in case it changed  ((IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.RefreshAsync(RefreshMode.StoreWins, dataItem);  // Update the data table  UpdateTable();  }  }  });  }  public void ClearSearch()  {  SearchText = null;  SearchColumnIndex = -1;  }  public void DataSelectionChanged()  {  EventHandler<DataItemSelectedEventArgs<T>> handler = ItemSelected;  handler?.Invoke(this, new DataItemSelectedEventArgs<T>  {  DataItem = (T)(GetView() as Views.DataTableView).Data.SelectedItem  });  NotifyOfPropertyChange("DataContextMenuExtraButtonEnabled");  }  public virtual void DataExtraButtonClicked(T dataItem)  {  }  private void AddDataItem(T dataItem)  {  // Add the data item to the data set  dataHandler.AddDataItem(dataItem);  // Update the table  UpdateTable();  }  private void UpdateDataItem(T dataItem)  {  // Update the data item  dataHandler.UpdateDataItem(dataItem);  // Update the table  UpdateTable();  }  protected void UpdateTable()  {  NotifyOfPropertyChange("Data");  DataUpdated?.Invoke(this, null);  }  protected virtual void InitDataItem(ref T dataItem)  {  dataItem = Misc.Utils.Create<T>();  }  protected virtual void InitDataItemDialog(ref V dialog, ref T dataItem)  {  dialog = new V  {  DataContext = dataItem  };  }  protected abstract List<Misc.Column> GetColumns();  }  } |

### IDataHandler

כפי שתיארתי במרכיבי ליבה, מחלקה זו אחראית על ביצוע פעולות שונות על מופעים של אובייקט גנרי במסד נתונים.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Data.Entity.Migrations;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.Interfaces  {  abstract class IDataHandler<T> where T : class  {  static protected dbEntities db = new dbEntities();  public virtual List<T> GetData()  {  return db.Set<T>().ToList();  }  public virtual void RemoveDataItem(T dataItem)  {  // Set the data item as deleted  typeof(T).GetProperty("Deleted").SetValue(dataItem, true);  // Update the data item  UpdateDataItem(dataItem);  }  public virtual void RestoreDataItem(T dataItem)  {  // Set the data item as not deleted  typeof(T).GetProperty("Deleted").SetValue(dataItem, false);  // Update the data item  UpdateDataItem(dataItem);  }  public virtual void AddDataItem(T dataItem)  {  // Verify the data item's content  VerifyDataItem(dataItem);  // Add the data item  try  {  db.Set<T>().Add(dataItem);  }  catch  {  throw new ArgumentException("Data item already exists in table!");  }  // Save changes to the database  db.SaveChanges();  }  public virtual void UpdateDataItem(T dataItem)  {  // Verify the data item's content  VerifyDataItem(dataItem);  // Update the entity  try  {  db.Set<T>().AddOrUpdate(dataItem);  }  catch  {  throw new ArgumentException("Data item already exists in table!");  }  // Save the database  db.SaveChanges();  }  public dbEntities GetEntities()  {  // Reload entities  db = new dbEntities();  return db;  }  protected abstract void VerifyDataItem(T dataItem);  }  } |

### DualDataTableViewModel

כפי שתיארתי במרכיבי הליבה, מחלקה זו אחראית על הצגת שני DataTableView בחלון אחד.

|  |
| --- |
| using Caliburn.Micro;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.ViewModels  {  class DualDataTableViewModel<T, U> : Conductor<object>.Collection.AllActive where T : Screen, new() where U : Screen, new()  {  public T FirstViewModel { get; } = new T();  public U SecondViewModel { get; } = new U();  public virtual string SecondViewName { get; set; }  protected override void OnViewReady(object view)  {  base.OnViewReady(view);  ActivateItem(FirstViewModel);  ActivateItem(SecondViewModel);  }  }  } |

### ConnectionViewModel

מחלקה זו אחראית על מימוש ה ViewModel עבור החלון של התחברות אל השרת ובכך מממשת את האותנטיקציה מול המסד נתונים.

|  |
| --- |
| using Caliburn.Micro;  using MySql.Data.MySqlClient;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Configuration;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  namespace CSCProject.ViewModels  {  class ConnectionViewModel : Screen, INotifyPropertyChanged  {  private WindowManager windowManager = new WindowManager();  public static string ConnectionString { get; set; } = ((ConnectionStringsSection)ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None).GetSection("connectionStrings")).ConnectionStrings["dbEntities"].ConnectionString;  public string Host { get; set; }  public string Username { get; set; }  public string Password { get; set; }  public bool CanConnectToServer => Host != null && Host.Length > 0 && Username != null && Username.Length > 0 && Password != null && Password.Length > 0;  public void ConnectToServer()  {  string[] HostDetails = Host.Split(':');  string connString = $"server={HostDetails[0]};port={(HostDetails.Count() > 1 ? int.Parse(HostDetails[1]) : 3306)};user id={Username};password={Password};database=project";  try  {  // Try to connect to the server  MySqlConnection connection = new MySqlConnection(connString);  connection.Open();  } catch (Exception ex)  {  MessageBox.Show(ex.Message, "Connection Error", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Save the connection string  SaveConnectionString();  // Show the shell view model  windowManager.ShowWindow(new ShellViewModel());  // Close the window  TryClose();  }  private void SaveConnectionString()  {  string[] HostDetails = Host.Split(':');  Configuration config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);  ConnectionStringsSection connectionStringsSection = (ConnectionStringsSection)config.GetSection("connectionStrings");  string connectionString = connectionStringsSection.ConnectionStrings["dbEntities"].ConnectionString;  // Replace connection info  connectionString = connectionString.Replace("localhost", HostDetails[0]);  connectionString = connectionString.Replace("3306", HostDetails.Count() > 1 ? HostDetails[1] : "3306");  connectionString = connectionString.Replace("server", Username);  connectionString = connectionString.Replace("123456789", Password);  // Set the connection string  ConnectionString = connectionString;  }  }  } |

### ShellViewModel

מחלקה זו אחראית על מימוש ה ViewModel עבור החלון המרכזי של התוכנה. מחלקה זו כוללת את כל ה ViewModels השונים עבור תוכניות התוכנה והיא מאפשרת את המעבר בין החלונות השונים.

|  |
| --- |
| using Caliburn.Micro;  using CSCProject.Views;  using MaterialDesignThemes.Wpf;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class ShellViewModel : Conductor<object>, INotifyPropertyChanged  {  public bool ViewsDrawerOpen { get; set; } = false;  public string CurrentViewName { get; set; } = "";  public List<TreeViewItem> TreeViewItems { get; set; } = new List<TreeViewItem>  {  new TreeViewItem { Header = "Home", Tag = new HomeViewModel() },  new TreeViewItem { Header = "Employee Management", ItemsSource = new List<TreeViewItem> {  new TreeViewItem { Header = "Employees", Tag = new EmployeesViewModel() },  new TreeViewItem { Header = "Employee Types", Tag = new EmployeeTypesViewModel() },  new TreeViewItem { Header = "Employee Expenses", Tag = new ExpensesViewModel() }  }  },  new TreeViewItem { Header = "Customer Management", ItemsSource = new List<TreeViewItem> {  new TreeViewItem { Header = "Customers", Tag = new CustomersViewModel() },  new TreeViewItem { Header = "Sale Orders", Tag = new SaleOrdersCombinedViewModel() }  }  },  new TreeViewItem { Header = "Vendor Management", ItemsSource = new List<TreeViewItem> {  new TreeViewItem { Header = "Vendors", Tag = new VendorsViewModel() },  new TreeViewItem { Header = "Purchase Orders", Tag = new PurchaseOrdersCombinedViewModel() }  }  },  new TreeViewItem { Header = "Factory Management", ItemsSource = new List<TreeViewItem> {  new TreeViewItem { Header = "Warehouses", Tag = new WarehousesViewModel() },  new TreeViewItem { Header = "Lots", Tag = new LotsViewModel() },  new TreeViewItem { Header = "Parts", Tag = new PartsViewModel() },  new TreeViewItem { Header = "Inventory", Tag = new InventoryViewModel() }  }  },  new TreeViewItem { Header = "Reports", ItemsSource = new List<TreeViewItem> {  new TreeViewItem { Header = "Employee Expenses Distribution", Tag = new ExpensesDistributionViewModel() },  new TreeViewItem { Header = "Sold Goods", Tag = new SoldGoodsViewModel() },  new TreeViewItem { Header = "Monthly Sale Orders", Tag = new MonthlySaleOrdersViewModel() },  new TreeViewItem { Header = "Monthly Purchase Orders", Tag = new MonthlyPurchaseOrdersViewModel() },  new TreeViewItem { Header = "Monthly Balance", Tag = new MonthlyBalanceViewModel() }  }  },  };  public ShellViewModel()  {  // Transition to the initial view (Home view)  TransitionToView("Home", (Screen)TreeViewItems[0].Tag);  }  public void TransitionToView(string ViewName, Screen ViewScreen)  {  if (ViewScreen == null)  {  return;  }  // Set the current view name  CurrentViewName = ViewName;  // Set the new display name of the app  DisplayName = $"Core Scientific Creations - {ViewName}";  // Activate the item  ActivateItem(ViewScreen);  // Close the views drawer if open  ViewsDrawerOpen = false;  }  public async void ShowAboutDialog()  {  System.Reflection.Assembly assembly = System.Reflection.Assembly.GetExecutingAssembly();  FileVersionInfo fvi = FileVersionInfo.GetVersionInfo(assembly.Location);  string version = fvi.FileVersion;  // Show the about message dialog  await DialogHost.Show(new Dialogs.MessageDialog { Message = $"CSC Database\nVersion {version}\n© 2020 Almog Hamdani.\nAll rights reserved." });  }  public void Exit()  {  Application.Current.Shutdown();  }  }  } |

### CustomersViewModel

מחלקה זו מממשת את החלון של ה Customers באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class CustomersViewModel : DataTableViewModel<Customer, DataHandlers.CustomersDataHandler, Dialogs.CustomerDialog>  {  public override string AddButtonIcon { get; set; } = "UserAdd";  public override string RemoveButtonIcon { get; set; } = "UserRemove";  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Name", PropertyBinding = new Binding("Name"), AllowSearch = true },  new Misc.Column { Name = "Phone Number", PropertyBinding = new Binding("Phone"), AllowSearch = true },  new Misc.Column { Name = "Postal Code", PropertyBinding = new Binding("Address.PostalCode"), AllowSearch = true },  new Misc.Column { Name = "City", PropertyBinding = new Binding("Address.City"), AllowSearch = true },  new Misc.Column { Name = "Street", PropertyBinding = new Binding("Address.Street"), AllowSearch = true }  };  }  }  } |

### EmployeesViewModel

מחלקה זו מממשת את החלון של ה Employees באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class EmployeesViewModel : DataTableViewModel<Employee, DataHandlers.EmployeesDataHandler, Dialogs.EmployeeDialog>  {  public override string AddButtonIcon { get; set; } = "UserAdd";  public override string RemoveButtonIcon { get; set; } = "UserRemove";  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "First Name", PropertyBinding = new Binding("FirstName"), AllowSearch = true },  new Misc.Column { Name = "Last Name", PropertyBinding = new Binding("LastName"), AllowSearch = true },  new Misc.Column { Name = "Birth Date", PropertyBinding = new Binding("BirthDate") { StringFormat = "d" }, AllowSearch = true },  new Misc.Column { Name = "Gender", PropertyBinding = new Binding("Gender") },  new Misc.Column { Name = "Phone Number", PropertyBinding = new Binding("Phone"), AllowSearch = true },  new Misc.Column { Name = "Postal Code", PropertyBinding = new Binding("Address.PostalCode"), AllowSearch = true },  new Misc.Column { Name = "City", PropertyBinding = new Binding("Address.City"), AllowSearch = true },  new Misc.Column { Name = "Street", PropertyBinding = new Binding("Address.Street"), AllowSearch = true },  new Misc.Column { Name = "Employee Type", PropertyBinding = new Binding("EmployeeType.Name"), AllowSearch = true }  };  }  protected override void InitDataItem(ref Employee dataItem)  {  dataItem = new Employee  {  EmployeeTypeId = -1,  BirthDate = Misc.DateTimeHelper.Yesterday,  Address = new Address()  };  }  protected override void InitDataItemDialog(ref Dialogs.EmployeeDialog dialog, ref Employee dataItem)  {  dialog = new Dialogs.EmployeeDialog  {  DataContext = new Dialogs.EmployeeDialogContext  {  Employee = dataItem,  EmployeeTypes = dataHandler.GetEntities().EmployeeTypes.ToList().FindAll(type => !type.Deleted)  }  };  }  }  } |

### EmployeeTypesViewModel

מחלקה זו מממשת את החלון של ה Employee Types באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class EmployeeTypesViewModel : DataTableViewModel<EmployeeType, DataHandlers.EmployeeTypesDataHandler, Dialogs.EmployeeTypeDialog>  {  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Name", PropertyBinding = new Binding("Name"), AllowSearch = true },  new Misc.Column { Name = "Monthly Salary", PropertyBinding = new Binding("Salary"), AllowSearch = true },  new Misc.Column { Name = "Description", PropertyBinding = new Binding("Description"), AllowSearch = true }  };  }  }  } |

### ExpensesViewModel

מחלקה זו מממשת את החלון של ה Expenses באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class ExpensesViewModel : DataTableViewModel<Expense, DataHandlers.ExpensesDataHandler, Dialogs.ExpenseDialog>  {  public override string AddButtonIcon { get; set; } = "Add";  public override string RemoveButtonIcon { get; set; } = "Remove";  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Description", PropertyBinding = new Binding("Description"), AllowSearch = true },  new Misc.Column { Name = "Price", PropertyBinding = new Binding("Price") { StringFormat = "{0}₪" }, AllowSearch = true },  new Misc.Column { Name = "Date", PropertyBinding = new Binding("Date") { StringFormat = "d" }, AllowSearch = true },  new Misc.Column { Name = "Employee Id", PropertyBinding = new Binding("Employee.Id") },  new Misc.Column { Name = "Employee First Name", PropertyBinding = new Binding("Employee.FirstName") },  new Misc.Column { Name = "Employee Last Name", PropertyBinding = new Binding("Employee.LastName") },  };  }  protected override void InitDataItem(ref Expense dataItem)  {  dataItem = new Expense  {  EmployeeId = -1,  Date = DateTime.Now  };  }  protected override void InitDataItemDialog(ref Dialogs.ExpenseDialog dialog, ref Expense dataItem)  {  dialog = new Dialogs.ExpenseDialog  {  DataContext = new Dialogs.ExpenseDialogContext  {  Expense = dataItem,  Employees = dataHandler.GetEntities().Employees.ToList().FindAll(employee => !employee.Deleted)  }  };  }  }  } |

### InventoryViewModel

מחלקה זו מממשת את החלון של ה Inventory באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class InventoryViewModel : DataTableViewModel<Inventory, DataHandlers.InventoryDataHandler, Dialogs.InventoryDialog>  {  public override bool HasId { get; set; } = false;  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Lot", PropertyBinding = new Binding("LotId"), AllowSearch = true },  new Misc.Column { Name = "Part Id", PropertyBinding = new Binding("PartId"), AllowSearch = true },  new Misc.Column { Name = "Part Description", PropertyBinding = new Binding("Part.Description"), AllowSearch = true },  new Misc.Column { Name = "Warehouse Id", PropertyBinding = new Binding("WarehouseId"), AllowSearch = true },  new Misc.Column { Name = "Warehouse Description", PropertyBinding = new Binding("Warehouse.Description"), AllowSearch = true },  new Misc.Column { Name = "Quantity", PropertyBinding = new MultiBinding() { Bindings = { new Binding("Quantity"), new Binding("Part.Unit") }, StringFormat="{0} {1}" }, AllowSearch = true }  };  }  protected override void InitDataItem(ref Inventory dataItem)  {  dataItem = new Inventory  {  LotId = "",  PartId = -1,  WarehouseId = -1  };  }  protected override void InitDataItemDialog(ref Dialogs.InventoryDialog dialog, ref Inventory dataItem)  {  dialog = new Dialogs.InventoryDialog  {  DataContext = new Dialogs.InventoryDialogContext  {  Inventory = dataItem,  Lots = dataHandler.GetEntities().Lots.ToList().FindAll(type => !type.Deleted),  Parts = dataHandler.GetEntities().Parts.ToList().FindAll(type => !type.Deleted),  Warehouses = dataHandler.GetEntities().Warehouses.ToList().FindAll(type => !type.Deleted)  }  };  }  }  } |

### LotsViewModel

מחלקה זו מממשת את החלון של ה Lots באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class LotsViewModel : DataTableViewModel<Lot, DataHandlers.LotsDataHandler, Dialogs.LotDialog>  {  public override bool ShowDataItemId { get; set; } = true;  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Type", PropertyBinding = new Binding("Type") { Converter = new Misc.LotTypeEnumToStringConverter() }, AllowSearch = true }  };  }  }  } |

### PartsViewModel

מחלקה זו מממשת את החלון של ה Parts באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class PartsViewModel : DataTableViewModel<Part, DataHandlers.PartsDataHandler, Dialogs.PartDialog>  {  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Description", PropertyBinding = new Binding("Description"), AllowSearch = true },  new Misc.Column { Name = "Type", PropertyBinding = new Binding("Type") { Converter = new Misc.LotTypeEnumToStringConverter() }, AllowSearch = true },  new Misc.Column { Name = "Price", PropertyBinding = new Binding("Price") { StringFormat = "{0}₪" }, AllowSearch = true },  new Misc.Column { Name = "Unit", PropertyBinding = new Binding("Unit") { Converter = new Misc.PartUnitEnumToStringConverter() }, AllowSearch = true },  };  }  }  } |

### PurchaseOrderPartsViewModel

מחלקה זו מממשת את החלון של ה Purchase Order Parts באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class PurchaseOrderPartsViewModel : DataTableViewModel<PurchaseOrderPart, DataHandlers.PurchaseOrderPartsDataHandler, Dialogs.PurchaseOrderPartDialog>  {  public override bool HasId { get; set; } = false;  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Order Id", PropertyBinding = new Binding("OrderId"), AllowSearch = true },  new Misc.Column { Name = "Part Id", PropertyBinding = new Binding("PartId"), AllowSearch = true },  new Misc.Column { Name = "Part Description", PropertyBinding = new Binding("Part.Description"), AllowSearch = true },  new Misc.Column { Name = "Lot Id", PropertyBinding = new Binding("LotId"), AllowSearch = true },  new Misc.Column { Name = "Quantity", PropertyBinding = new MultiBinding() { Bindings = { new Binding("Quantity"), new Binding("Part.Unit") }, StringFormat="{0} {1}" }, AllowSearch = true }  };  }  protected override void InitDataItem(ref PurchaseOrderPart dataItem)  {  dataItem = new PurchaseOrderPart  {  LotId = "",  OrderId = -1,  PartId = -1  };  }  protected override void InitDataItemDialog(ref Dialogs.PurchaseOrderPartDialog dialog, ref PurchaseOrderPart dataItem)  {  dialog = new Dialogs.PurchaseOrderPartDialog  {  DataContext = new Dialogs.PurchaseOrderPartDialogContext  {  PurchaseOrderPart = dataItem,  Orders = dataHandler.GetEntities().PurchaseOrders.ToList().FindAll(order => !order.Deleted),  Parts = dataHandler.GetEntities().Parts.ToList().FindAll(part => !part.Deleted && part.Type == LotType.RawMaterial),  Lots = dataHandler.GetEntities().Lots.ToList().FindAll(lot => !lot.Deleted && lot.Type == LotType.RawMaterial)  }  };  }  }  } |

### PurchaseOrdersViewModel

מחלקה זו מממשת את החלון של ה Purchase Orders באמצעות ה DataTableViewModel.

|  |
| --- |
| using CSCProject.DataHandlers;  using MaterialDesignThemes.Wpf;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class PurchaseOrdersViewModel : DataTableViewModel<PurchaseOrder, DataHandlers.PurchaseOrdersDataHandler, Dialogs.PurchaseOrderDialog>  {  public override bool ShowDataItemId { get; set; } = true;  public override bool HasDataContextMenuExtraButton { get; } = true;  public override string DataContextMenuExtraButtonText { get; } = "Order Received";  public override string DataContextMenuExtraButtonIcon { get; } = "TruckCheck";  public override bool DataContextMenuExtraButtonEnabled { get => (GetView() as Views.DataTableView).Data.SelectedItem != null && !((PurchaseOrder)(GetView() as Views.DataTableView).Data.SelectedItem).Received; }  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Vendor Id", PropertyBinding = new Binding("VendorId"), AllowSearch = true },  new Misc.Column { Name = "Vendor Name", PropertyBinding = new Binding("Vendor.Name"), AllowSearch = true },  new Misc.Column { Name = "Order Date", PropertyBinding = new Binding("Date"), AllowSearch = true },  new Misc.Column { Name = "Received", PropertyBinding = new Binding("Received"), AllowSearch = true },  new Misc.Column { Name = "Total Price", PropertyBinding = new Binding("Price") { StringFormat = "{0}₪" }, AllowSearch = true }  };  }  protected override void InitDataItem(ref PurchaseOrder dataItem)  {  dataItem = new PurchaseOrder  {  VendorId = -1,  Date = DateTime.Now,  Received = false  };  }  protected override void InitDataItemDialog(ref Dialogs.PurchaseOrderDialog dialog, ref PurchaseOrder dataItem)  {  dialog = new Dialogs.PurchaseOrderDialog  {  DataContext = new Dialogs.PurchaseOrderDialogContext  {  PurchaseOrder = dataItem,  Vendors = dataHandler.GetEntities().Vendors.ToList().FindAll(type => !type.Deleted)  }  };  }  public override async void DataExtraButtonClicked(PurchaseOrder dataItem)  {  // If the order is empty  if (dataItem.Parts.Count == 0)  {  await DialogHost.Show(new Dialogs.MessageDialog { Message = "Order is empty" });  return;  }  Dialogs.SelectWarehouseDialog selectWarehouseDialog = new Dialogs.SelectWarehouseDialog  {  DataContext = new Dialogs.SelectWarehouseDialogContext  {  WarehouseId = -1,  Warehouses = dataHandler.GetEntities().Warehouses.ToList().FindAll(type => !type.Deleted)  }  };  // Show the data item dialog  await DialogHost.Show(selectWarehouseDialog, (object \_, DialogClosingEventArgs args) =>  {  // Check if the dialog closing is a message dialog  if (args.Session.Content.GetType() != typeof(Dialogs.MessageDialog))  {  // If not cancelled  if (args.Parameter.GetType() == typeof(bool) && (bool)args.Parameter)  {  try  {  (dataHandler as PurchaseOrdersDataHandler).RecieveOrder(dataItem, (selectWarehouseDialog.DataContext as Dialogs.SelectWarehouseDialogContext).WarehouseId);  UpdateTable();  NotifyOfPropertyChange("DataContextMenuExtraButtonEnabled");  }  catch (Exception ex)  {  // Cancel the close of the dialog  args.Cancel();  // Show the error  args.Session.UpdateContent(new Dialogs.MessageDialog { Message = ex.Message });  }  }  }  });  }  }  } |

### SaleOrderPartsViewModel

מחלקה זו מממשת את החלון של ה Sale Order Parts באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class SaleOrderPartsViewModel : DataTableViewModel<SaleOrderPart, DataHandlers.SaleOrderPartsDataHandler, Dialogs.SaleOrderPartDialog>  {  public override bool HasId { get; set; } = false;  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Order Id", PropertyBinding = new Binding("OrderId"), AllowSearch = true },  new Misc.Column { Name = "Part Id", PropertyBinding = new Binding("PartId"), AllowSearch = true },  new Misc.Column { Name = "Part Description", PropertyBinding = new Binding("Part.Description"), AllowSearch = true },  new Misc.Column { Name = "Lot Id", PropertyBinding = new Binding("LotId"), AllowSearch = true },  new Misc.Column { Name = "Quantity", PropertyBinding = new MultiBinding() { Bindings = { new Binding("Quantity"), new Binding("Part.Unit") }, StringFormat="{0} {1}" }, AllowSearch = true }  };  }  protected override void InitDataItem(ref SaleOrderPart dataItem)  {  dataItem = new SaleOrderPart  {  LotId = "",  OrderId = -1,  PartId = -1  };  }  protected override void InitDataItemDialog(ref Dialogs.SaleOrderPartDialog dialog, ref SaleOrderPart dataItem)  {  dialog = new Dialogs.SaleOrderPartDialog  {  DataContext = new Dialogs.SaleOrderPartDialogContext  {  SaleOrderPart = dataItem,  Orders = dataHandler.GetEntities().SaleOrders.ToList().FindAll(order => !order.Deleted),  Parts = dataHandler.GetEntities().Parts.ToList().FindAll(part => !part.Deleted && part.Type == LotType.FinishedGood),  Lots = dataHandler.GetEntities().Lots.ToList().FindAll(lot => !lot.Deleted && lot.Type == LotType.FinishedGood)  }  };  }  }  } |

### SaleOrdersViewModel

מחלקה זו מממשת את החלון של ה Sale Orders באמצעות ה DataTableViewModel.

|  |
| --- |
| using CSCProject.DataHandlers;  using MaterialDesignThemes.Wpf;  using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class SaleOrdersViewModel : DataTableViewModel<SaleOrder, DataHandlers.SaleOrdersDataHandler, Dialogs.SaleOrderDialog>  {  public override bool ShowDataItemId { get; set; } = true;  public override bool HasDataContextMenuExtraButton { get; } = true;  public override string DataContextMenuExtraButtonText { get; } = "Order Shipped";  public override string DataContextMenuExtraButtonIcon { get; } = "TruckCheck";  public override bool DataContextMenuExtraButtonEnabled { get => (GetView() as Views.DataTableView).Data.SelectedItem != null && !((SaleOrder)(GetView() as Views.DataTableView).Data.SelectedItem).Shipped; }  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Customer Id", PropertyBinding = new Binding("CustomerId"), AllowSearch = true },  new Misc.Column { Name = "Customer Name", PropertyBinding = new Binding("Customer.Name"), AllowSearch = true },  new Misc.Column { Name = "Order Date", PropertyBinding = new Binding("Date"), AllowSearch = true },  new Misc.Column { Name = "Shipped", PropertyBinding = new Binding("Shipped"), AllowSearch = true },  new Misc.Column { Name = "Total Price", PropertyBinding = new Binding("Price") { StringFormat = "{0}₪" }, AllowSearch = true }  };  }  protected override void InitDataItem(ref SaleOrder dataItem)  {  dataItem = new SaleOrder  {  CustomerId = -1,  Date = DateTime.Now,  Shipped = false  };  }  protected override void InitDataItemDialog(ref Dialogs.SaleOrderDialog dialog, ref SaleOrder dataItem)  {  dialog = new Dialogs.SaleOrderDialog  {  DataContext = new Dialogs.SaleOrderDialogContext  {  SaleOrder = dataItem,  Customers = dataHandler.GetEntities().Customers.ToList().FindAll(type => !type.Deleted)  }  };  }  public override async void DataExtraButtonClicked(SaleOrder dataItem)  {  // If the order is empty  if (dataItem.Parts.Count == 0)  {  await DialogHost.Show(new Dialogs.MessageDialog { Message = "Order is empty" });  return;  }  // Try to ship the order  try  {  (dataHandler as SaleOrdersDataHandler).ShipOrder(dataItem);  UpdateTable();  NotifyOfPropertyChange("DataContextMenuExtraButtonEnabled");  }  catch (Exception ex)  {  // Show the error  await DialogHost.Show(new Dialogs.MessageDialog { Message = ex.Message });  }  }  }  } |

### VendorsViewModel

מחלקה זו מממשת את החלון של ה Vendors באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class VendorsViewModel : DataTableViewModel<Vendor, DataHandlers.VendorsDataHandler, Dialogs.VendorDialog>  {  public override string AddButtonIcon { get; set; } = "BankAdd";  public override string RemoveButtonIcon { get; set; } = "BankRemove";  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Name", PropertyBinding = new Binding("Name"), AllowSearch = true },  new Misc.Column { Name = "Phone Number", PropertyBinding = new Binding("Phone"), AllowSearch = true },  new Misc.Column { Name = "Postal Code", PropertyBinding = new Binding("Address.PostalCode"), AllowSearch = true },  new Misc.Column { Name = "City", PropertyBinding = new Binding("Address.City"), AllowSearch = true },  new Misc.Column { Name = "Street", PropertyBinding = new Binding("Address.Street"), AllowSearch = true }  };  }  }  } |

### WarehousesViewModel

מחלקה זו מממשת את החלון של ה Warehouses באמצעות ה DataTableViewModel.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.ViewModels  {  class WarehousesViewModel : DataTableViewModel<Warehouse, DataHandlers.WarehousesDataHandler, Dialogs.WarehouseDialog>  {  public override string AddButtonIcon { get; set; } = "TableColumnAddAfter";  public override string RemoveButtonIcon { get; set; } = "TableColumnRemoveAfter";  public override bool ShowDataItemId { get; set; } = true;  protected override List<Misc.Column> GetColumns()  {  return new List<Misc.Column> {  new Misc.Column { Name = "Description", PropertyBinding = new Binding("Description"), AllowSearch = true }  };  }  }  } |

### ExpensesDistributionViewModel

מחלקה זו מממשת את החלון של הדו"ח התפלגות הוצאות לא שוטפות באמצעות הספרייה OxyPlot.

|  |
| --- |
| using Caliburn.Micro;  using OxyPlot;  using OxyPlot.Series;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.ViewModels  {  class ExpensesDistributionViewModel : Screen, INotifyPropertyChanged  {  public PlotModel Model {  get  {  DataHandlers.ExpensesDataHandler dataHandler = new DataHandlers.ExpensesDataHandler();  PlotModel model = new PlotModel();  dynamic pieSeries = new PieSeries { InsideLabelPosition = 0.7, OutsideLabelFormat = "{0}₪ - {2:0.#}%", Font = "Roboto" };  List<Expense> expenses = dataHandler.GetData().FindAll(expense => !expense.Deleted);  Dictionary<Employee, int> employeesExpenses = new Dictionary<Employee, int>();  // For each expense, add it to the employee's total expense  foreach (Expense expense in expenses)  {  // Check if the employee key exists  if (!employeesExpenses.ContainsKey(expense.Employee))  {  employeesExpenses[expense.Employee] = 0;  }  // Add the expense price to the employee's expense  employeesExpenses[expense.Employee] += expense.Price;  }  // For each employee, create it's pie slice  foreach (var employeeTotalExpense in employeesExpenses)  {  pieSeries.Slices.Add(new PieSlice(string.Format("{0} {1}", employeeTotalExpense.Key.FirstName, employeeTotalExpense.Key.LastName), employeeTotalExpense.Value));  }  // Add the pie series to the model1  model.Series.Add(pieSeries);  return model;  }  }  }  } |

### MonthlyBalanceViewModel

מחלקה זו מממשת את החלון של הדו"ח מאזן חודשי באמצעות הספרייה OxyPlot.

|  |
| --- |
| using Caliburn.Micro;  using OxyPlot;  using OxyPlot.Axes;  using OxyPlot.Series;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.ViewModels  {  class MonthlyBalanceViewModel : Screen, INotifyPropertyChanged  {  public PlotModel Model  {  get  {  DataHandlers.ExpensesDataHandler expensesDataHandler = new DataHandlers.ExpensesDataHandler();  DataHandlers.SaleOrdersDataHandler saleOrdersDataHandler = new DataHandlers.SaleOrdersDataHandler();  DataHandlers.PurchaseOrdersDataHandler purchaseOrdersDataHandler = new DataHandlers.PurchaseOrdersDataHandler();  PlotModel model = new PlotModel();  ColumnSeries columnSeries = new ColumnSeries { Font = "Roboto", LabelPlacement = LabelPlacement.Inside, LabelFormatString = "{0}₪" };  List<PurchaseOrder> purchaseOrders = purchaseOrdersDataHandler.GetData().FindAll(order => !order.Deleted);  List<SaleOrder> saleOrders = saleOrdersDataHandler.GetData().FindAll(order => !order.Deleted);  List<Expense> expenses = expensesDataHandler.GetData().FindAll(expense => !expense.Deleted);  float[] balance = new float[12];  // For each purchase order, check if it's in the last year  foreach (PurchaseOrder purchaseOrder in purchaseOrders)  {  if (purchaseOrder.Date.Year == DateTime.Today.Year)  {  balance[purchaseOrder.Date.Month - 1] -= purchaseOrder.Price;  }  }  // For each expense, check if it's in the last year  foreach (Expense expense in expenses)  {  if (expense.Date.Year == DateTime.Today.Year)  {  balance[expense.Date.Month - 1] -= expense.Price;  }  }  // For each sale order, check if it's in the last year  foreach (SaleOrder saleOrder in saleOrders)  {  if (saleOrder.Date.Year == DateTime.Today.Year)  {  balance[saleOrder.Date.Month - 1] += saleOrder.Price;  }  }  // For each month, add it's value to the column series items  for (int month = 0; month < 12; month++)  {  columnSeries.Items.Add(new ColumnItem { Value = balance[month], Color = balance[month] > 0 ? OxyColor.FromRgb(0, 230, 118) : OxyColor.FromRgb(255, 23, 68) });  }  // Add the column series to the model1  model.Series.Add(columnSeries);  // Add axes  model.Axes.Add(new CategoryAxis  {  Position = AxisPosition.Bottom,  Key = "CakeAxis",  ItemsSource = new[] {  "January",  "Feburary",  "March",  "April",  "May",  "June",  "July",  "August",  "September",  "October",  "November",  "December"  }  });  model.Axes.Add(new LinearAxis  {  Position = AxisPosition.Left,  ExtraGridlines = new Double[] { 0 }  });  return model;  }  }  }  } |

### MonthlyPurchaseOrdersViewModel

מחלקה זו מממשת את החלון של הדו"ח הזמנות מספק חודשיות באמצעות הספרייה OxyPlot.

|  |
| --- |
| using Caliburn.Micro;  using OxyPlot;  using OxyPlot.Axes;  using OxyPlot.Series;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.ViewModels  {  class MonthlyPurchaseOrdersViewModel : Screen, INotifyPropertyChanged  {  public PlotModel Model  {  get  {  DataHandlers.PurchaseOrdersDataHandler dataHandler = new DataHandlers.PurchaseOrdersDataHandler();  PlotModel model = new PlotModel();  ColumnSeries columnSeries = new ColumnSeries { Font = "Roboto", LabelPlacement = LabelPlacement.Inside, LabelFormatString = "{0}" };  List<PurchaseOrder> purchaseOrders = dataHandler.GetData().FindAll(order => !order.Deleted);  int[] monthlyOrders = new int[12];  // For each purchase order, check if it's in the last year  foreach (PurchaseOrder purchaseOrder in purchaseOrders)  {  if (purchaseOrder.Date.Year == DateTime.Today.Year)  {  monthlyOrders[purchaseOrder.Date.Month - 1]++;  }  }  // For each month, add it's value to the column series items  for (int month = 0; month < 12; month++)  {  columnSeries.Items.Add(new ColumnItem { Value = monthlyOrders[month], Color = OxyColor.FromRgb(255,165,0) });  }  // Add the column series to the model1  model.Series.Add(columnSeries);  // Add axes  model.Axes.Add(new CategoryAxis  {  Position = AxisPosition.Bottom,  Key = "CakeAxis",  ItemsSource = new[] {  "January",  "Feburary",  "March",  "April",  "May",  "June",  "July",  "August",  "September",  "October",  "November",  "December"  }  });  model.Axes.Add(new LinearAxis  {  Position = AxisPosition.Left,  AbsoluteMinimum = 0  });  return model;  }  }  }  } |

### MonthlySaleOrdersViewModel

מחלקה זו מממשת את החלון של הדו"ח הזמנות לקוח חודשיות באמצעות הספרייה OxyPlot.

|  |
| --- |
| using Caliburn.Micro;  using OxyPlot;  using OxyPlot.Axes;  using OxyPlot.Series;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.ViewModels  {  class MonthlySaleOrdersViewModel : Screen, INotifyPropertyChanged  {  public PlotModel Model  {  get  {  DataHandlers.SaleOrdersDataHandler dataHandler = new DataHandlers.SaleOrdersDataHandler();  PlotModel model = new PlotModel();  ColumnSeries columnSeries = new ColumnSeries { Font = "Roboto", LabelPlacement = LabelPlacement.Inside, LabelFormatString = "{0}" };  List<SaleOrder> saleOrders = dataHandler.GetData().FindAll(order => !order.Deleted);  int[] monthlyOrders = new int[12];  // For each sale order, check if it's in the last year  foreach (SaleOrder saleOrder in saleOrders)  {  if (saleOrder.Date.Year == DateTime.Today.Year)  {  monthlyOrders[saleOrder.Date.Month - 1]++;  }  }  // For each month, add it's value to the column series items  for (int month = 0; month < 12; month++)  {  columnSeries.Items.Add(new ColumnItem { Value = monthlyOrders[month], Color = OxyColor.FromRgb(88,204,237) });  }  // Add the column series to the model1  model.Series.Add(columnSeries);  // Add axes  model.Axes.Add(new CategoryAxis  {  Position = AxisPosition.Bottom,  Key = "CakeAxis",  ItemsSource = new[] {  "January",  "Feburary",  "March",  "April",  "May",  "June",  "July",  "August",  "September",  "October",  "November",  "December"  }  });  model.Axes.Add(new LinearAxis  {  Position = AxisPosition.Left,  AbsoluteMinimum = 0  });  return model;  }  }  }  } |

### SoldGoodsViewModel

מחלקה זו מממשת את החלון של הדו"ח מוצרים נמכרים ביותר באמצעות הספרייה OxyPlot.

|  |
| --- |
| using Caliburn.Micro;  using OxyPlot;  using OxyPlot.Axes;  using OxyPlot.Series;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.ViewModels  {  class SoldGoodsViewModel : Screen, INotifyPropertyChanged  {  public PlotModel Model  {  get  {  DataHandlers.SaleOrderPartsDataHandler dataHandler = new DataHandlers.SaleOrderPartsDataHandler();  PlotModel model = new PlotModel();  List<SaleOrderPart> saleOrderParts = dataHandler.GetData().FindAll(part => !part.Deleted);  List<BarItem> barItems = new List<BarItem>();  Dictionary<Part, float> soldGoods = new Dictionary<Part, float>();  // For each sale order part, add it to the part's total sale price  foreach (SaleOrderPart saleOrderPart in saleOrderParts)  {  // Check if the part key exists  if (!soldGoods.ContainsKey(saleOrderPart.Part))  {  soldGoods[saleOrderPart.Part] = 0;  }  // Add the sale price to the part's price  soldGoods[saleOrderPart.Part] += saleOrderPart.Part.Price \* saleOrderPart.Quantity;  }  // For each sold good, add it to the items source  foreach (var soldGood in soldGoods)  {  barItems.Add(new BarItem { Value = soldGood.Value, Color = OxyColor.FromRgb(244, 67, 54) });  }  // Add the bar series to the model1  model.Series.Add(new BarSeries  {  Font = "Roboto",  ItemsSource = barItems,  LabelPlacement = LabelPlacement.Inside,  LabelFormatString = "{0:.000}₪"  });  // Add axes  model.Axes.Add(new CategoryAxis  {  Position = AxisPosition.Left,  Key = "CakeAxis",  ItemsSource = soldGoods.Keys.ToList().Select(p => p.Description)  });  return model;  }  }  }  } |

### OrdersViewModel

מחלקה זו מממשת את החלון הגנרי עבור הזמנות (הזמנות מספק או הזמנות לקוח) באמצעות DualDataTableView.

|  |
| --- |
| using Caliburn.Micro;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.ViewModels  {  class OrdersViewModel<T, U, T1, T2, T3, U1, U2, U3> : DualDataTableViewModel<T, U>  where T : DataTableViewModel<T1, T2, T3>, new() where U : DataTableViewModel<U1, U2, U3>, new()  where T1 : class, new() where T2 : Interfaces.IDataHandler<T1>, new() where T3 : Dialogs.Dialog, new()  where U1 : class, new() where U2 : Interfaces.IDataHandler<U1>, new() where U3 : Dialogs.Dialog, new()  {  public override string SecondViewName { get; set; } = "Order Parts:";  protected override void OnViewLoaded(object view)  {  base.OnViewLoaded(view);  FirstViewModel.ItemSelected += FirstViewModel\_ItemSelected;  // Update first view model on second view model update  SecondViewModel.DataUpdated += delegate { FirstViewModel.NotifyOfPropertyChange("Data"); };  }  private void FirstViewModel\_ItemSelected(object sender, DataItemSelectedEventArgs<T1> e)  {  if (e.DataItem != null)  {  SecondViewModel.SearchColumnIndex = 0; // This should be the order id column  SecondViewModel.SearchText = e.DataItem.GetType().GetProperty("Id").GetValue(e.DataItem).ToString();  } else  {  SecondViewModel.ClearSearch();  }  }  }  class PurchaseOrdersCombinedViewModel : OrdersViewModel<PurchaseOrdersViewModel, PurchaseOrderPartsViewModel, PurchaseOrder, DataHandlers.PurchaseOrdersDataHandler, Dialogs.PurchaseOrderDialog, PurchaseOrderPart, DataHandlers.PurchaseOrderPartsDataHandler, Dialogs.PurchaseOrderPartDialog> { }  class SaleOrdersCombinedViewModel : OrdersViewModel<SaleOrdersViewModel, SaleOrderPartsViewModel, SaleOrder, DataHandlers.SaleOrdersDataHandler, Dialogs.SaleOrderDialog, SaleOrderPart, DataHandlers.SaleOrderPartsDataHandler, Dialogs.SaleOrderPartDialog> { }  } |

### CustomersDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט Customer באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class CustomersDataHandler : Interfaces.IDataHandler<Customer>  {  public override void AddDataItem(Customer dataItem)  {  // Set the address of the customer  dataItem.PostalCode = dataItem.Address.PostalCode;  // Check if the address already exists in the db  if (Misc.Utils.CheckIfAddressExists(db, dataItem.PostalCode))  {  // Remove address to prevent creation of new address  dataItem.Address = null;  }  // Call base add data item  base.AddDataItem(dataItem);  }  protected override void VerifyDataItem(Customer dataItem)  {  Address address = dataItem.Address == null ? db.Addresses.Find(dataItem.PostalCode) : dataItem.Address;  // Check if the name of the customer is valid  if (!Misc.Utils.VerifyName(dataItem.Name))  {  throw new ArgumentException("Invalid vendor name");  }  // Check for valid phone number  if (!Misc.Utils.VerifyPhone(dataItem.Phone))  {  throw new ArgumentException("Invalid phone number");  }  // Check for valid address  if (!Misc.Utils.VerifyName(address.City))  {  throw new ArgumentException("Invalid city name");  }  }  }  } |

### EmployeesDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט Employee באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class EmployeesDataHandler : Interfaces.IDataHandler<Employee>  {  public override void AddDataItem(Employee dataItem)  {  // Set the address of the vendor  dataItem.PostalCode = dataItem.Address.PostalCode;  // Check if the address already exists in the db  if (Misc.Utils.CheckIfAddressExists(db, dataItem.PostalCode))  {  // Remove address to prevent creation of new address  dataItem.Address = null;  }  // Call base add data item  base.AddDataItem(dataItem);  }  protected override void VerifyDataItem(Employee dataItem)  {  Address address = dataItem.Address == null ? db.Addresses.Find(dataItem.PostalCode) : dataItem.Address;  // Check if the name of the employee is valid  if (!Misc.Utils.VerifyName(dataItem.FirstName) || !Misc.Utils.VerifyName(dataItem.LastName))  {  throw new ArgumentException("Invalid employee name");  }  // Check for valid phone number  if (!Misc.Utils.VerifyPhone(dataItem.Phone))  {  throw new ArgumentException("Invalid phone number");  }  // Check for valid address  if (!Misc.Utils.VerifyName(address.City))  {  throw new ArgumentException("Invalid city name");  }  // Check for valid birthdate  if (DateTime.Today.AddYears(-6) < dataItem.BirthDate)  {  throw new ArgumentException("Invalid birth date");  }  }  }  } |

### EmployeeTypesDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט EmployeeType באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class EmployeeTypesDataHandler : Interfaces.IDataHandler<EmployeeType>  {  protected override void VerifyDataItem(EmployeeType dataItem)  {  // Check if the name of the employee type is valid  if (!Misc.Utils.VerifyName(dataItem.Name))  {  throw new ArgumentException("Invalid employee type name");  }  // Check if the salary is a valid integer  if (dataItem.Salary <= 0)  {  throw new ArgumentException("Invalid salary");  }  }  }  } |

### ExpensesDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט Expense באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class ExpensesDataHandler : Interfaces.IDataHandler<Expense>  {  protected override void VerifyDataItem(Expense dataItem)  {  // Check if the price of the expense is a valid price  if (dataItem.Price <= 0)  {  throw new ArgumentException("Invalid expense price");  }  }  }  } |

### InventoryDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט Inventory באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class InventoryDataHandler : Interfaces.IDataHandler<Inventory>  {  public override void AddDataItem(Inventory dataItem)  {  Inventory existingItem = db.Inventories.ToList().FindAll(i => i.LotId == dataItem.LotId && i.PartId == dataItem.PartId && i.WarehouseId == dataItem.WarehouseId).FirstOrDefault();  // If there is an existing item with this information, update it's quantity instead  if (existingItem != null && dataItem.Quantity > 0)  {  existingItem.Quantity += dataItem.Quantity;  UpdateDataItem(existingItem);  } else  {  base.AddDataItem(dataItem);  }  }  protected override void VerifyDataItem(Inventory dataItem)  {  Lot lot = db.Lots.ToList().FindAll(l => l.Id == dataItem.LotId).FirstOrDefault();  Part part = db.Parts.ToList().FindAll(p => p.Id == dataItem.PartId).FirstOrDefault();  // Check for valid quantity  if (dataItem.Quantity < 1)  {  throw new ArgumentException("Invalid quantity");  }  // Check if the the lot type is different from the part type  if (lot.Type != part.Type)  {  throw new ArgumentException("Lot and Part type mismatch");  }  }  }  } |

### LotsDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט Lot באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class LotsDataHandler : Interfaces.IDataHandler<Lot>  {  protected override void VerifyDataItem(Lot dataItem)  {  // Check if the id of the lot is valid  if (!Misc.Utils.VerifyLotId(dataItem.Id))  {  throw new ArgumentException("Invalid lot id");  }  }  }  } |

### PartsDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט Part באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class PartsDataHandler : Interfaces.IDataHandler<Part>  {  protected override void VerifyDataItem(Part dataItem)  {  // Check if the price of the data item is invalid  if (dataItem.Price <= 0)  {  throw new ArgumentException("Invalid price");  }  }  }  } |

### PurchaseOrderPartsDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט PurchaseOrderPart באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class PurchaseOrderPartsDataHandler : Interfaces.IDataHandler<PurchaseOrderPart>  {  public override void AddDataItem(PurchaseOrderPart dataItem)  {  PurchaseOrderPart existingItem = db.PurchaseOrderParts.ToList().FindAll(p => p.OrderId == dataItem.OrderId && p.PartId == dataItem.PartId && p.LotId == dataItem.LotId).FirstOrDefault();  // If there is an existing item with this information, update it's quantity instead  if (existingItem != null)  {  existingItem.Quantity += dataItem.Quantity;  base.UpdateDataItem(existingItem);  } else  {  base.AddDataItem(dataItem);  }  }  protected override void VerifyDataItem(PurchaseOrderPart dataItem)  {  Part part = db.Parts.ToList().FindAll(p => p.Id == dataItem.PartId).FirstOrDefault();  Lot lot = db.Lots.ToList().FindAll(l => l.Id == dataItem.LotId).FirstOrDefault();  // Check for valid quantity  if (dataItem.Quantity <= 0)  {  throw new ArgumentException("Invalid quantity");  }  // Verify that the part is a raw material  if (part.Type != LotType.RawMaterial) {  throw new ArgumentException("Part must be raw material");  }  // Verify that the lot is a raw material  if (lot.Type != LotType.RawMaterial)  {  throw new ArgumentException("Lot must be raw material");  }  }  }  } |

### PurchaseOrdersDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט PurchaseOrder באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Data.Entity.Core.Objects;  using System.Data.Entity.Infrastructure;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class PurchaseOrdersDataHandler : Interfaces.IDataHandler<PurchaseOrder>  {  public override List<PurchaseOrder> GetData()  {  return base.GetData().Select(o => CalcPrice(o)).ToList();  }  public void RecieveOrder(PurchaseOrder order, int warehouseId)  {  InventoryDataHandler inventoryDataHandler = new InventoryDataHandler();  if (db.Warehouses.Find(warehouseId) == null)  {  throw new ArgumentException("Warehouse not found");  }  foreach (var part in order.Parts)  {  if (!part.Deleted)  {  inventoryDataHandler.AddDataItem(new Inventory { LotId = part.LotId, PartId = part.PartId, Quantity = part.Quantity, WarehouseId = warehouseId });  }  }  // Set the order as recieved  order.Received = true;  UpdateDataItem(order);  }  private static PurchaseOrder CalcPrice(PurchaseOrder order)  {  float price = 0;  // Reload parts collection  db.Entry(order).Collection(o => o.Parts).Load();  foreach (var part in order.Parts)  {  ((IObjectContextAdapter)db).ObjectContext.Refresh(RefreshMode.StoreWins, part);  if (!part.Deleted)  {  price += part.Part.Price \* part.Quantity;  }  }  order.Price = price;  return order;  }  protected override void VerifyDataItem(PurchaseOrder dataItem)  {  }  }  } |

### SaleOrderPartsDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט SaleOrderPart באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class SaleOrderPartsDataHandler : Interfaces.IDataHandler<SaleOrderPart>  {  public override void AddDataItem(SaleOrderPart dataItem)  {  SaleOrderPart existingItem = db.SaleOrderParts.ToList().FindAll(p => p.OrderId == dataItem.OrderId && p.PartId == dataItem.PartId && p.LotId == dataItem.LotId).FirstOrDefault();  // If there is an existing item with this information, update it's quantity instead  if (existingItem != null)  {  existingItem.Quantity += dataItem.Quantity;  base.UpdateDataItem(existingItem);  }  else  {  base.AddDataItem(dataItem);  }  }  protected override void VerifyDataItem(SaleOrderPart dataItem)  {  Part part = db.Parts.ToList().FindAll(p => p.Id == dataItem.PartId).FirstOrDefault();  Lot lot = db.Lots.ToList().FindAll(l => l.Id == dataItem.LotId).FirstOrDefault();  // Check for valid quantity  if (dataItem.Quantity <= 0)  {  throw new ArgumentException("Invalid quantity");  }  // Verify that the part is a finished good  if (part.Type != LotType.FinishedGood)  {  throw new ArgumentException("Part must be finished good");  }  // Verify that the lot is a finished good  if (lot.Type != LotType.FinishedGood)  {  throw new ArgumentException("Lot must be finished good");  }  }  }  } |

### SaleOrdersDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט SaleOrder באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Data.Entity.Core.Objects;  using System.Data.Entity.Infrastructure;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class SaleOrdersDataHandler : Interfaces.IDataHandler<SaleOrder>  {  public override List<SaleOrder> GetData()  {  return base.GetData().Select(o => CalcPrice(o)).ToList();  }  public void ShipOrder(SaleOrder order)  {  InventoryDataHandler inventoryDataHandler = new InventoryDataHandler();  List<Inventory> inventory = inventoryDataHandler.GetData();  foreach (var part in order.Parts)  {  if (!part.Deleted)  {  Inventory partInventory = inventory.Find(i => i.PartId == part.PartId && i.LotId == part.LotId);  // If the part is missing or insufficient in the inventory  if (partInventory == null || partInventory.Quantity < part.Quantity)  {  throw new ArgumentException($"Insufficient goods\nMissing '{(partInventory == null ? part.Quantity : part.Quantity - partInventory.Quantity)} {part.Part.Unit}' of Part '{part.Part.Description}'");  }  if (partInventory.Quantity == part.Quantity)  {  inventoryDataHandler.RemoveDataItem(partInventory);  }  else  {  partInventory.Quantity -= part.Quantity;  inventoryDataHandler.UpdateDataItem(partInventory);  }  }  }  // Set the order as shipped  order.Shipped = true;  UpdateDataItem(order);  }  private static SaleOrder CalcPrice(SaleOrder order)  {  float price = 0;  // Reload parts collection  db.Entry(order).Collection(o => o.Parts).Load();  foreach (var part in order.Parts)  {  ((IObjectContextAdapter)db).ObjectContext.Refresh(RefreshMode.StoreWins, part);  if (!part.Deleted)  {  price += part.Part.Price \* part.Quantity;  }  }  order.Price = price;  return order;  }  protected override void VerifyDataItem(SaleOrder dataItem)  {  }  }  } |

### VendorsDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט Vendor באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class VendorsDataHandler : Interfaces.IDataHandler<Vendor>  {  public override void AddDataItem(Vendor dataItem)  {  // Set the address of the vendor  dataItem.PostalCode = dataItem.Address.PostalCode;  // Check if the address already exists in the db  if (Misc.Utils.CheckIfAddressExists(db, dataItem.PostalCode))  {  // Remove address to prevent creation of new address  dataItem.Address = null;  }  // Call base add data item  base.AddDataItem(dataItem);  }  protected override void VerifyDataItem(Vendor dataItem)  {  Address address = dataItem.Address == null ? db.Addresses.Find(dataItem.PostalCode) : dataItem.Address;  // Check if the name of the customer is valid  if (!Misc.Utils.VerifyName(dataItem.Name))  {  throw new ArgumentException("Invalid vendor name");  }  // Check for valid phone number  if (!Misc.Utils.VerifyPhone(dataItem.Phone))  {  throw new ArgumentException("Invalid phone number");  }  // Check for valid address  if (!Misc.Utils.VerifyName(address.City))  {  throw new ArgumentException("Invalid city name");  }  }  }  } |

### WarehousesDataHandler

מחלקה זו מממשת את הממשק למסד הנתונים של האובייקט Warehouse באמצעות IDataHandler.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.DataHandlers  {  class WarehousesDataHandler : Interfaces.IDataHandler<Warehouse>  {  protected override void VerifyDataItem(Warehouse warehouse)  {  }  }  } |

### BoolToStringConverter

מחלקה זו אחראית על מימוש IValueConverter (המאפשר המרה ב UI של Binding Values) בין Bool ל String.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.Misc  {  public class BoolToStringConverter : IValueConverter  {  public string FalseValue { get; set; }  public string TrueValue { get; set; }  public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)  {  if (value == null)  return FalseValue;  else  return (bool)value ? TrueValue : FalseValue;  }  public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)  {  return value != null ? value.Equals(TrueValue) : false;  }  }  } |

### BoolToVisibilityConverter

מחלקה זו אחראית על מימוש IValueConverter (המאפשר המרה ב UI של Binding Values) בין Bool ל Visibility.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Globalization;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Data;  namespace CSCProject.Misc  {  public class BoolToVisibilityConverter : IValueConverter  {  public object Convert(object value, Type targetType, object parameter, CultureInfo info)  {  return (value is bool && (bool)value) ? Visibility.Visible : Visibility.Collapsed;  }  public object ConvertBack(object value, Type targetType, object parameter, CultureInfo info)  {  return value is Visibility && (Visibility)value == Visibility.Visible;  }  }  } |

### DateTimeHelper

מחלקה זו מספקת עזרים ל UI להשיג באופן סטטי את התאריכים של מחר ושל אתמול.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CSCProject.Misc  {  public static class DateTimeHelper  {  public static DateTime Yesterday  {  get { return DateTime.Today.AddDays(-1); }  }  public static DateTime Tomorrow  {  get { return DateTime.Today.AddDays(1); }  }  }  } |

### FloatConverter

מחלקה זו אחראית על מימוש IValueConverter (המאפשר המרה ב UI של Binding Values) בין String ל Float ובכך מתקנת בעיות שנגרמות עקב ההמרה הסטנדרטית של Binding.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Globalization;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.Misc  {  public class FloatConverter : IValueConverter  {  public object Convert(object value, Type targetType, object parameter, CultureInfo culture)  {  return value;  }  public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)  {  // return an invalid value in case of the value ends with a point  return value.ToString().EndsWith(".") ? "." : value;  }  }  } |

### LotTypeEnumToStringConverter

מחלקה זו אחראית על מימוש IValueConverter (המאפשר המרה ב UI של Binding Values) בין LotType ל String.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Globalization;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.Misc  {  public class LotTypeEnumToStringConverter : IValueConverter  {  public object Convert(object value, Type targetType, object parameter, CultureInfo culture)  {  switch((LotType)value)  {  case LotType.RawMaterial:  return "Raw Material";  case LotType.FinishedGood:  return "Finished Good";  default:  return "Unknown";  }  }  public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)  {  throw new NotSupportedException();  }  }  } |

### PartUnitEnumToStringConverter

מחלקה זו אחראית על מימוש IValueConverter (המאפשר המרה ב UI של Binding Values) בין PartUnit ל String.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Globalization;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Data;  namespace CSCProject.Misc  {  public class PartUnitEnumToStringConverter : IValueConverter  {  public object Convert(object value, Type targetType, object parameter, CultureInfo culture)  {  switch((PartUnit)value)  {  case PartUnit.Eaches:  return "Ea";  case PartUnit.SquardMeter:  return "Sm";  case PartUnit.Litter:  return "Lt";  default:  return "Unknown";  }  }  public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)  {  throw new NotSupportedException();  }  }  } |

### PasswordBoxHelper

מחלקה זו אחראית על סיפוק עזרים ל PasswordBox על מנת שיהיה אפשר לעשות Binding ל Password.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Reflection;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Controls;  namespace CSCProject.Misc  {  public static class PasswordBoxHelper  {  public static readonly DependencyProperty BoundPasswordProperty =  DependencyProperty.RegisterAttached("BoundPassword",  typeof(string),  typeof(PasswordBoxHelper),  new FrameworkPropertyMetadata(string.Empty, OnBoundPasswordChanged));  public static string GetBoundPassword(DependencyObject d)  {  var box = d as PasswordBox;  if (box != null)  {  // this funny little dance here ensures that we've hooked the  // PasswordChanged event once, and only once.  box.PasswordChanged -= PasswordChanged;  box.PasswordChanged += PasswordChanged;  }  return (string)d.GetValue(BoundPasswordProperty);  }  public static void SetBoundPassword(DependencyObject d, string value)  {  if (string.Equals(value, GetBoundPassword(d)))  return; // and this is how we prevent infinite recursion  d.SetValue(BoundPasswordProperty, value);  }  private static void OnBoundPasswordChanged(  DependencyObject d,  DependencyPropertyChangedEventArgs e)  {  var box = d as PasswordBox;  if (box == null)  return;  box.Password = GetBoundPassword(d);  }  private static void PasswordChanged(object sender, RoutedEventArgs e)  {  PasswordBox password = sender as PasswordBox;  SetBoundPassword(password, password.Password);  // set cursor past the last character in the password box  password.GetType().GetMethod("Select", BindingFlags.Instance | BindingFlags.NonPublic).Invoke(password, new object[] { password.Password.Length, 0 });  }  }  } |

### RelayCommand

מחלקה זו אחראית על סיפוק מימוש בסיסי של ICommand באמצעות פרמטרים שהמחלקה מקבלת דרך הפעולה בונה שלה.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Input;  namespace CSCProject.Misc  {  public class RelayCommand : ICommand  {  private Action<object> execute;  private Func<object, bool> canExecute;  public event EventHandler CanExecuteChanged  {  add { CommandManager.RequerySuggested += value; }  remove { CommandManager.RequerySuggested -= value; }  }  public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)  {  this.execute = execute;  this.canExecute = canExecute;  }  public bool CanExecute(object parameter)  {  return this.canExecute == null || this.canExecute(parameter);  }  public void Execute(object parameter)  {  this.execute(parameter);  }  }  } |

### Utils

מחלקה זו אחראית על מימוש עזרים לכל המחלקות בתוכנית. מחלקה זו כוללת בדיקה רקורסיבית האם אובייקט גנרי הוא ריק (הפונקציה IsAnyNullOrEmpty) או יצירה גנרית של אובייקט או טיפוס או מימוש של בדיקות תקינות קלט.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Reflection;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  namespace CSCProject.Misc  {  public static class Utils  {  public static bool IsAnyNullOrEmpty(object myObject)  {  if (myObject == null)  {  return false;  }  foreach (PropertyInfo pi in myObject.GetType().GetProperties())  {  object value = pi.GetValue(myObject);  if (pi.PropertyType == typeof(string))  {  if (string.IsNullOrEmpty((string)value))  {  return true;  }  }  else if (pi.PropertyType == typeof(DateTime))  {  return false;  }  else  {  if (IsAnyNullOrEmpty(value))  {  return true;  }  }  }  return false;  }  public static object GetPropertyValue(Type type, string path, object obj)  {  object propertyObject = obj;  Type currentType = type;  foreach (string propertyName in path.Split('.'))  {  PropertyInfo property = currentType.GetProperty(propertyName);  currentType = property.PropertyType;  propertyObject = property.GetValue(propertyObject);  }  return propertyObject;  }  public static Type FollowPropertyPath(Type type, string path)  {  Type currentType = type;  foreach (string propertyName in path.Split('.'))  {  PropertyInfo property = currentType.GetProperty(propertyName);  currentType = property.PropertyType;  }  return currentType;  }  public static bool CheckIfAddressExists(dbEntities db, int postalCode)  {  return db.Addresses.Count(a => a.PostalCode == postalCode) != 0;  }  public static bool VerifyName(string name)  {  Regex nameRegex = new Regex(@"^([a-zA-Z]+?)([-\s'][a-zA-Z]+)\*?$");  return nameRegex.IsMatch(name);  }  public static bool VerifyPhone(string phone)  {  Regex phoneRegex = new Regex(@"^\+?(972|0)(\-)?0?(([23489]{1}\d{7})|[5]{1}\d{8})$");  return phoneRegex.IsMatch(phone);  }  public static bool VerifyLotId(string id)  {  Regex lotIdRegex = new Regex(@"^[a-zA-Z0-9]\*$");  return lotIdRegex.IsMatch(id) && id.Length < 20;  }  public static T Create<T>()  {  var type = typeof(T);  return (T)Create(type);  }  public static object Create(Type type)  {  var obj = Activator.CreateInstance(type);  foreach (var property in type.GetProperties())  {  var propertyType = property.PropertyType;  if (propertyType.IsClass  && string.IsNullOrEmpty(propertyType.Namespace)  || (!propertyType.Namespace.Equals("System")  && !propertyType.Namespace.StartsWith("System.")))  {  var child = Create(propertyType);  property.SetValue(obj, child);  }  }  return obj;  }  }  } |

# סיכום אישי

פרויקט זה לימד אותי רבות אודות WPF, MVVM, Material Design, OxyPlot שבעזרתם ושילובם בניתי את הפרויקט הזה. בפרויקט זה למדתי כיצד לפתח מערכת לניהול מפעלים ברמה גבוהה ובתהליך למדתי גם כיצד מפעלים שונים פועלים. כמו כן, בפרויקט זה למדתי כיצד לשלב מסד נתונים (כמו MySQL שהשתמשתי בפרויקט זה) עם Frontend Desktop Application.

יתר על כן, פרויקט זה לימד איך לנהל את הזמן שלי באופן נכון משום שהיה עליי לשלב פעילות על פרויקט שונים ובגרויות בזמן העבודה על פרויקט, ולכן אני חושב שבזכות פרויקט זה למדתי כיצד לנהל את זמני ואני חושב שדבר זה יעזור לי גם בעתיד שלי בפיתוח תוכנות.