מערכת לניהול חנות בגדים

מגיש: איתי בנבניסטי

תעודת זהות: 209267582

תוכן

[מבוא 5](#_Toc42526994)

[נושא הפרוייקט: 5](#_Toc42526995)

[מהות התוכנה: 5](#_Toc42526996)

[תהליכים מרכזיים 5](#_Toc42526997)

[מדריך למשתמש 6](#_Toc42526998)

[דף ראשי 6](#_Toc42526999)

[עובדים 7](#_Toc42527000)

[הוספת עובד 7](#_Toc42527001)

[סוגי עובדים 8](#_Toc42527002)

[הוספת סוג עובד 8](#_Toc42527003)

[משמרות 9](#_Toc42527004)

[הוספת משמרת 9](#_Toc42527005)

[לקוחות 10](#_Toc42527006)

[הוספת לקוח 10](#_Toc42527007)

[מוצרים 11](#_Toc42527008)

[הוספת מוצר 11](#_Toc42527009)

[סוגי מוצרים 12](#_Toc42527010)

[הוספת סוג מוצר 12](#_Toc42527011)

[הוצאות 13](#_Toc42527012)

[הוספת הוצאה 13](#_Toc42527013)

[מלאי 14](#_Toc42527014)

[הוספה למלאי 14](#_Toc42527015)

[מכירות 15](#_Toc42527016)

[הוספת מכירה 15](#_Toc42527017)

[מדריך למתכנת 16](#_Toc42527018)

[הסבר על שיטת השכבות 16](#_Toc42527019)

[קשרים בין הטבלאות 16](#_Toc42527020)

[טבלת העובדים 17](#_Toc42527021)

[טבלת סוגי העובדים 17](#_Toc42527022)

[טבלת משמרות 18](#_Toc42527023)

[טבלת לקוחות 19](#_Toc42527024)

[טבלת מוצרים 19](#_Toc42527025)

[טבלת סוגי מוצרים 20](#_Toc42527026)

[טבלת הוצאות 20](#_Toc42527027)

[טבלת מלאי 21](#_Toc42527028)

[טבלת המכירות 21](#_Toc42527029)

[טבלת הזמנות מספק 22](#_Toc42527030)

[טבלה קישורת בין הזמנה מספק למוצרים בהזמנה 22](#_Toc42527031)

[קוד התוכנה 23](#_Toc42527032)

[קוד משני 23](#_Toc42527033)

[Bootstrapper 23](#_Toc42527034)

[Bool Converter 23](#_Toc42527035)

[ViewModels 24](#_Toc42527036)

[ShellViewModel 24](#_Toc42527037)

[CustomerViewModel 25](#_Toc42527038)

[EmployeesViewModel 26](#_Toc42527039)

[EmployeeTypesViewModel 28](#_Toc42527040)

[ExpenseViewModel 29](#_Toc42527041)

[HomePageViewModel 30](#_Toc42527042)

[InventoryViewModel 31](#_Toc42527043)

[MerchandiseCategoriesViewModel 32](#_Toc42527044)

[MerchandiseViewModel 34](#_Toc42527045)

[OrderViewModel 35](#_Toc42527046)

[ShiftsViewModel 37](#_Toc42527047)

[DataHandlers 38](#_Toc42527048)

[Abstract dataHandler 38](#_Toc42527049)

[CustomerDataHandler 39](#_Toc42527050)

[EmployeeDataHandler 40](#_Toc42527051)

[EmployeeTypesDataHandler 41](#_Toc42527052)

[ExpenseDataHandler 42](#_Toc42527053)

[InventoryDataHandler 42](#_Toc42527054)

[MerchandiseCategoryDataHandler 42](#_Toc42527055)

[MerchandiseDataHandler 43](#_Toc42527056)

[OrderDataHandler 44](#_Toc42527057)

[ShiftsDataHandler 44](#_Toc42527058)

# מבוא

## נושא הפרוייקט:

הקמה של מערכת ממוחשבת כללית למודל חנות בגדים.

## מהות התוכנה:

תוכנה זו נכתבה במטרה לייעל את דרכי ההתנהלות של חנויות בגדים.

# תהליכים מרכזיים

דוחות

בעזרת תהליך זה יכולה החנות לעקוב אחר ההכנסות שלה, וכן גם ההוצאות.

ניהול מלאי

בעזרת תהליך זה יכולה החנות לבדוק מה מצב המלאי בעסק וכן לחדש את המלאי כאשר מקבל הזמנה חדשה.

ניהול לקוחות

בעזרת תהליך זה יכולה החנות לשמור את הפרטים של הלקוחות.

ניהול מכירות

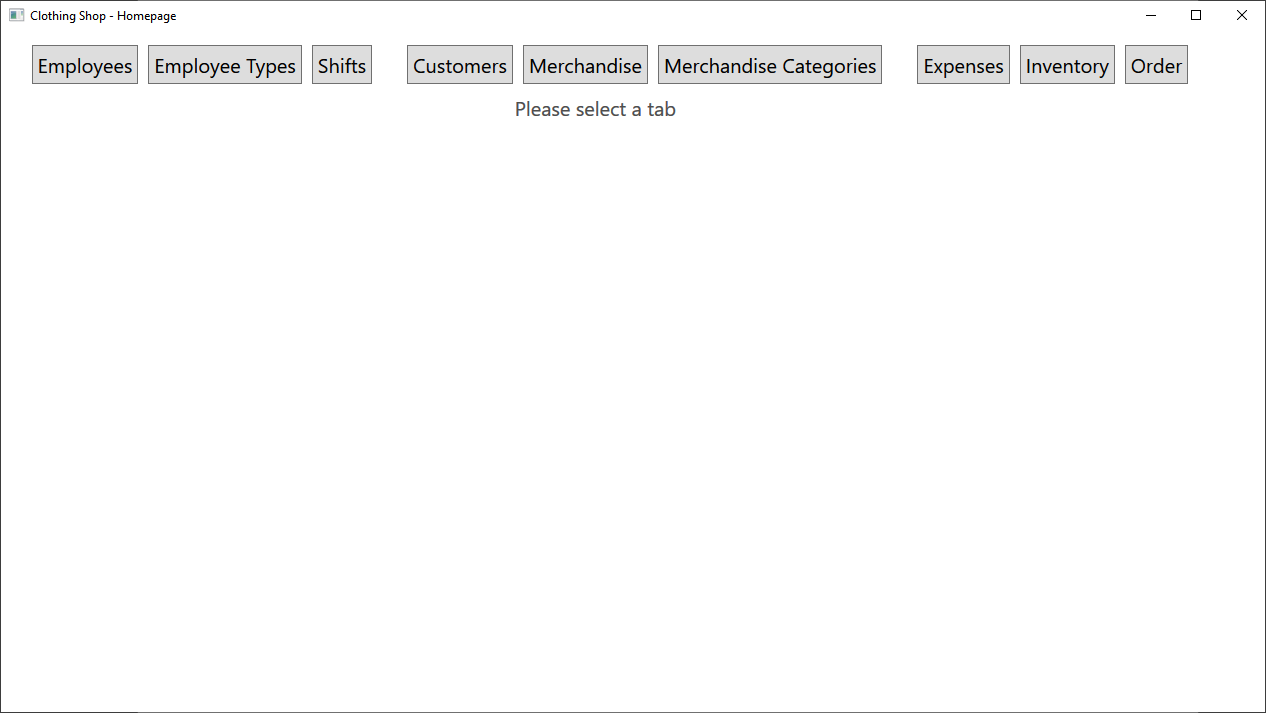
בעזרת תהליך זה יכולה החנות לרשום את פרטי המכירות.

ניהול עובדים

בעזרת תהליך זה יכולה החנות לרשום את עובדיו לעסק, לפטרם וכן לערוך פרטיהם.

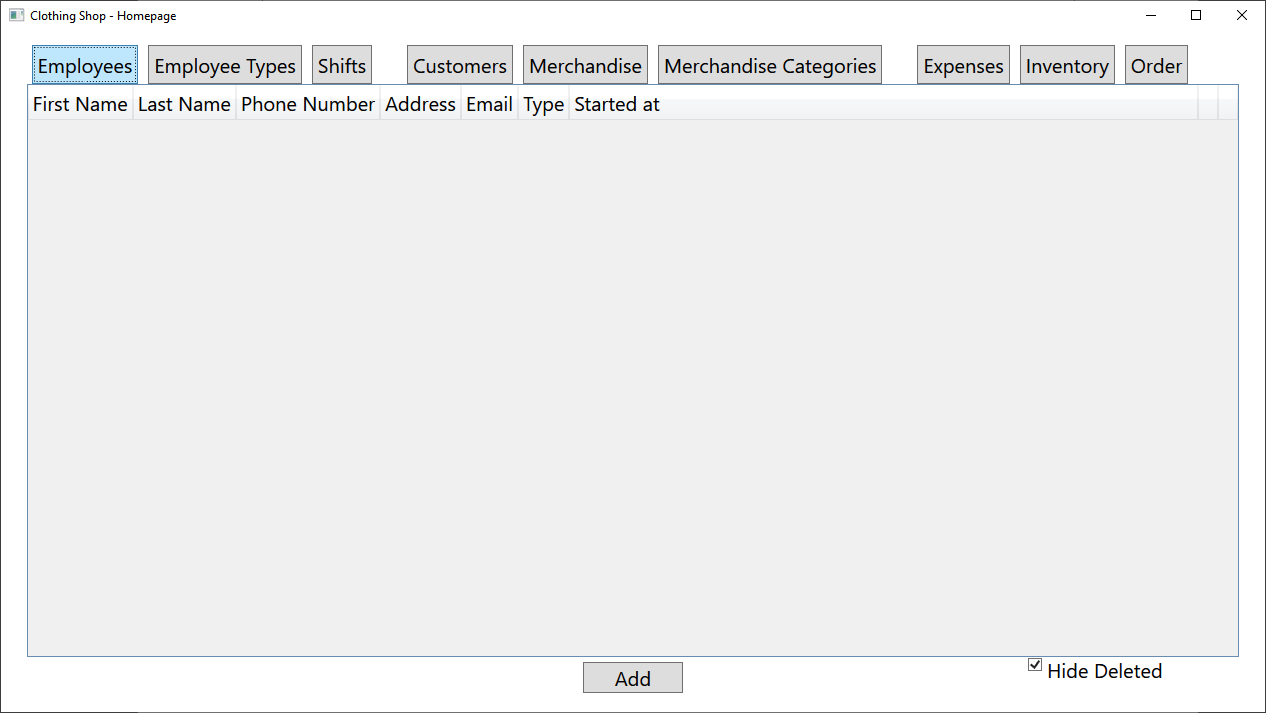
# מדריך למשתמש

## דף ראשי



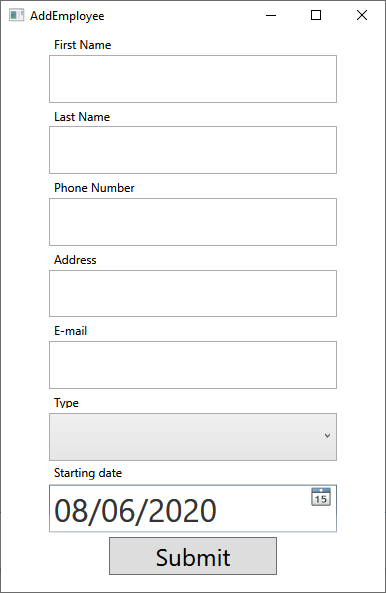
דף זה משמש כמסך הבית של התוכנה, דרך דף זה אפשר להגיע לכל האפשרויות בתוכנה

## עובדים

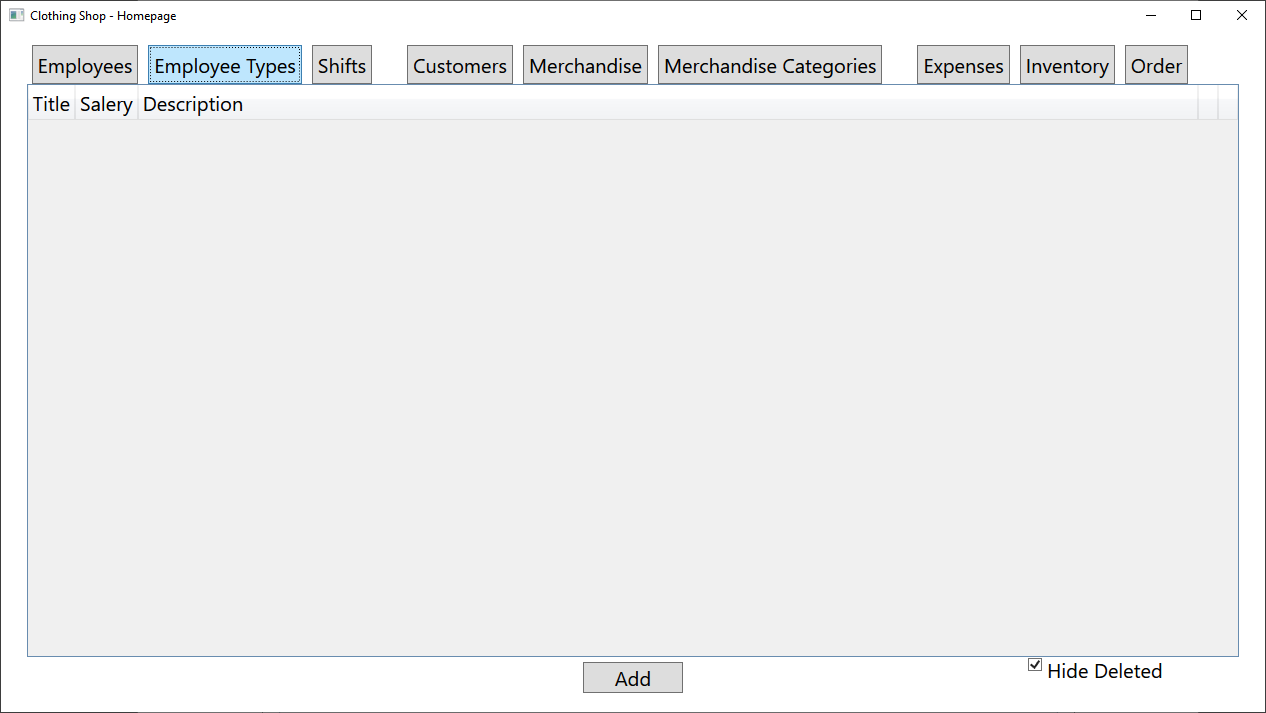


דף העובדים, בדף זה אפשר להוסיף, לערוך או להסיר עובדים.

## הוספת עובד

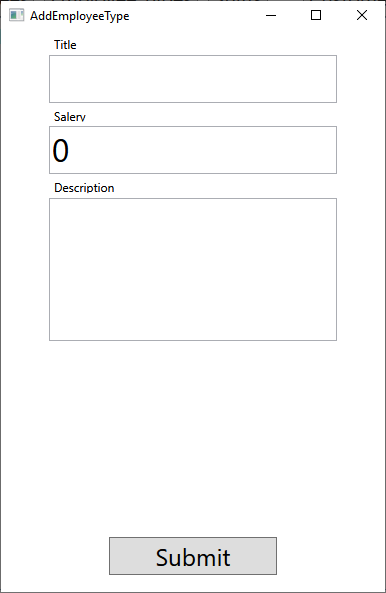


## סוגי עובדים

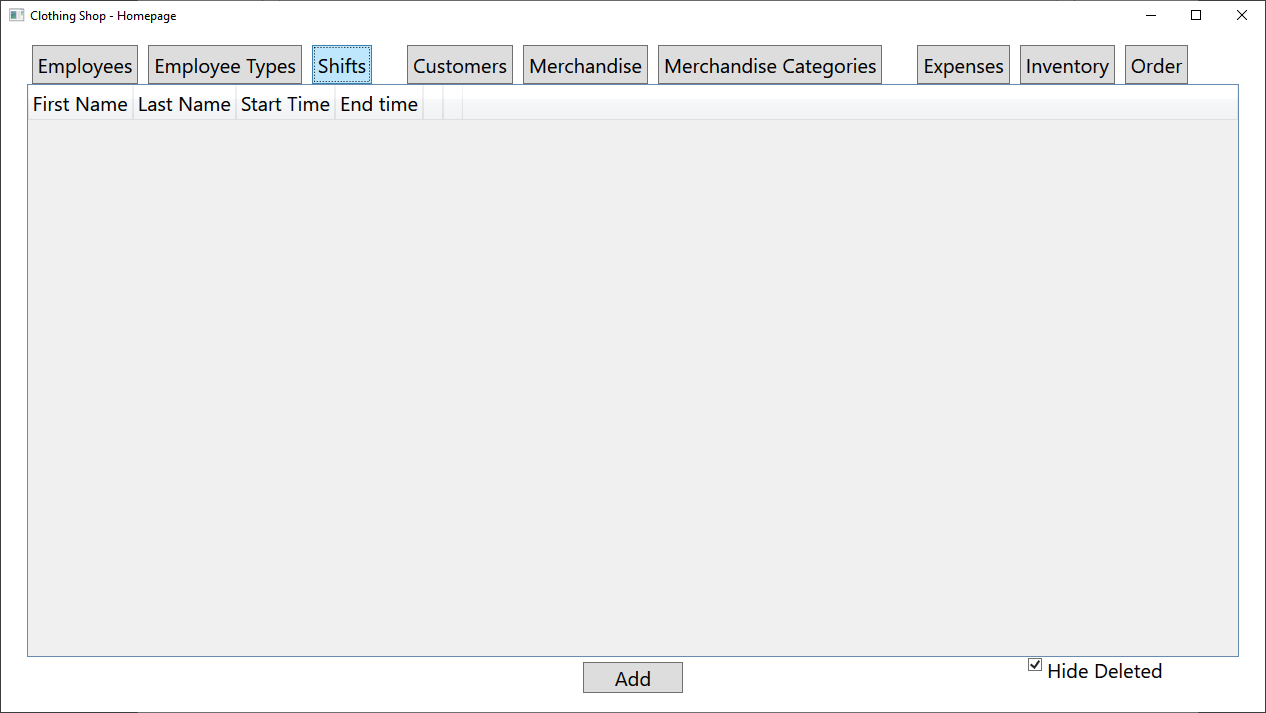


דף סוגי העובדים, בדף זה אפשר להוסיף, לערוך או להסיר סוגי עובדים.

## הוספת סוג עובד

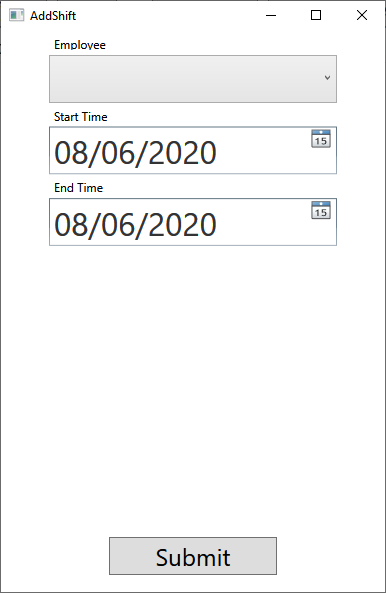


## משמרות

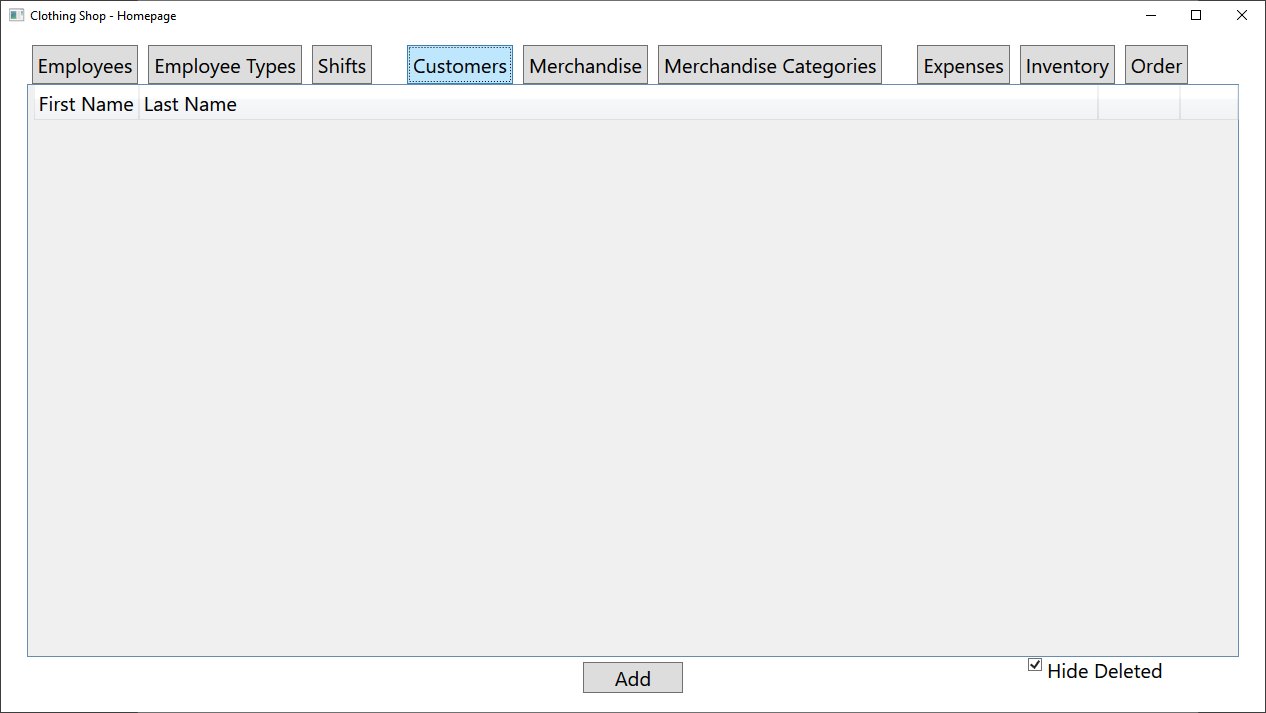


דף משמרות, בדף זה אפשר להוסיף, לערוך או להסיר משמרות לפי העובדים הקיימים.

## הוספת משמרת

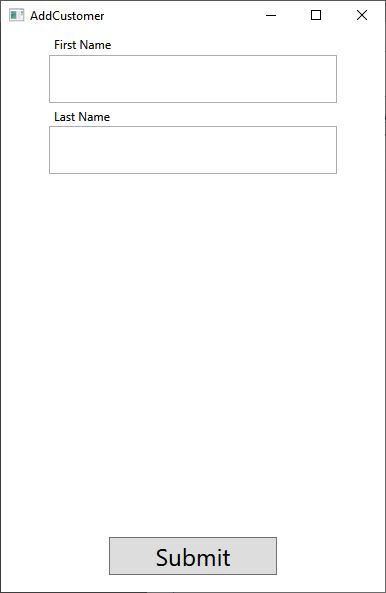


## לקוחות

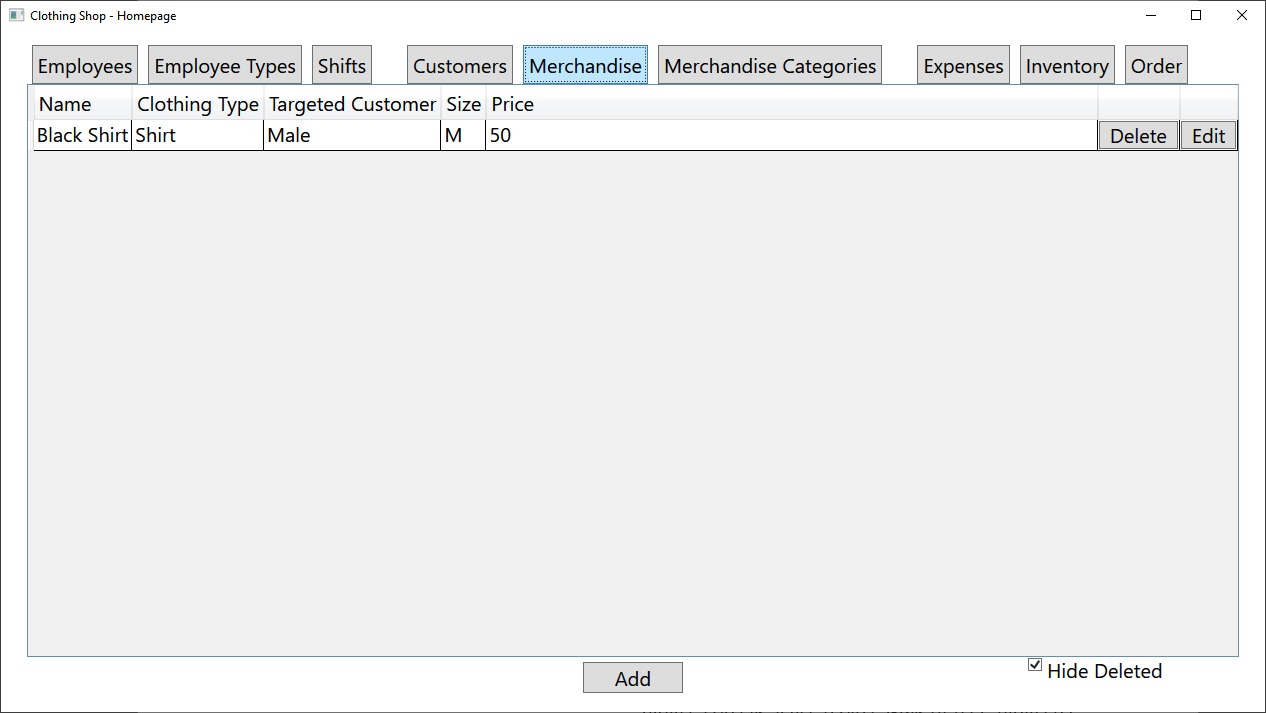


דף לקוחות, בדף זה אפשר להוסיף, לערוך או להסיר לקוחות.

## הוספת לקוח

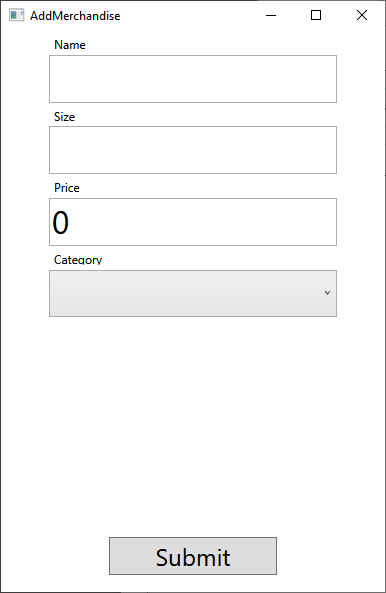


## מוצרים

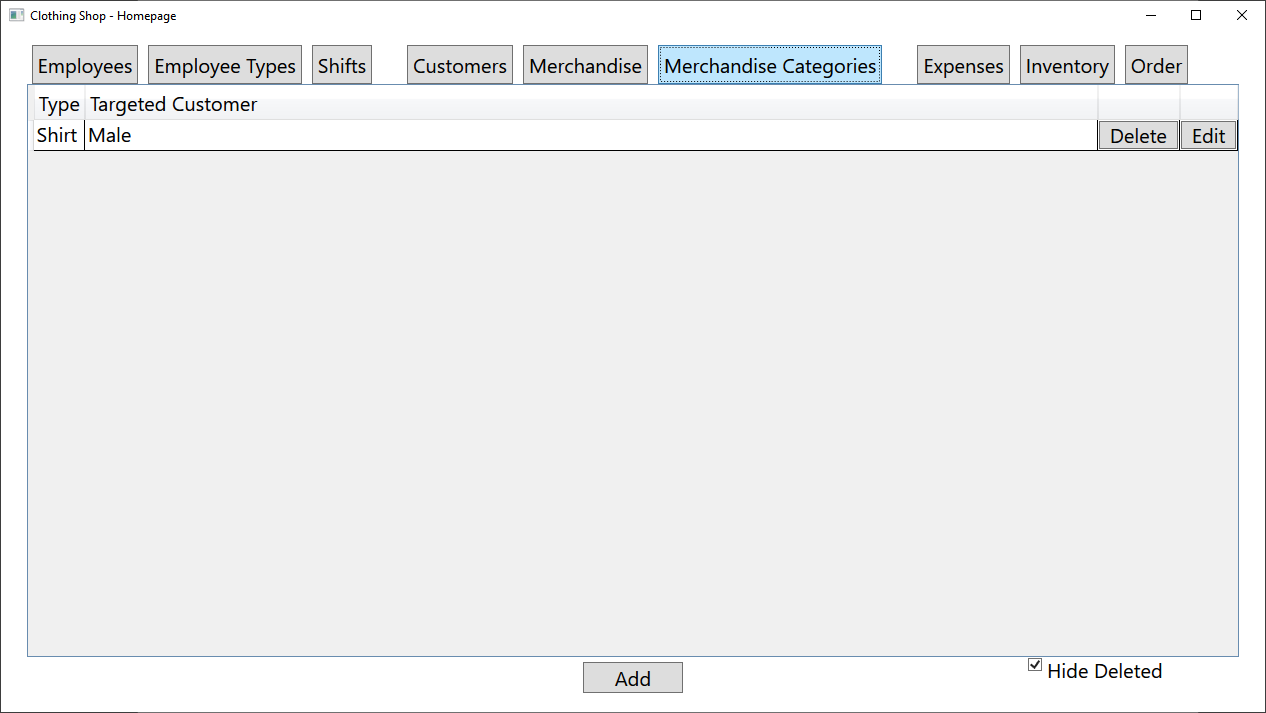


דף מוצרים, בדף זה אפשר להוסיף, לערוך או להסיר מוצרים. המוצרים לא בהכרח קיימים בחנות, אלא היו קיימים או בדרך לחנות.

## הוספת מוצר

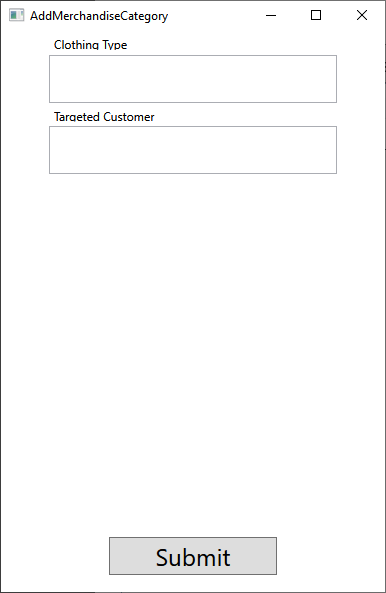


## סוגי מוצרים

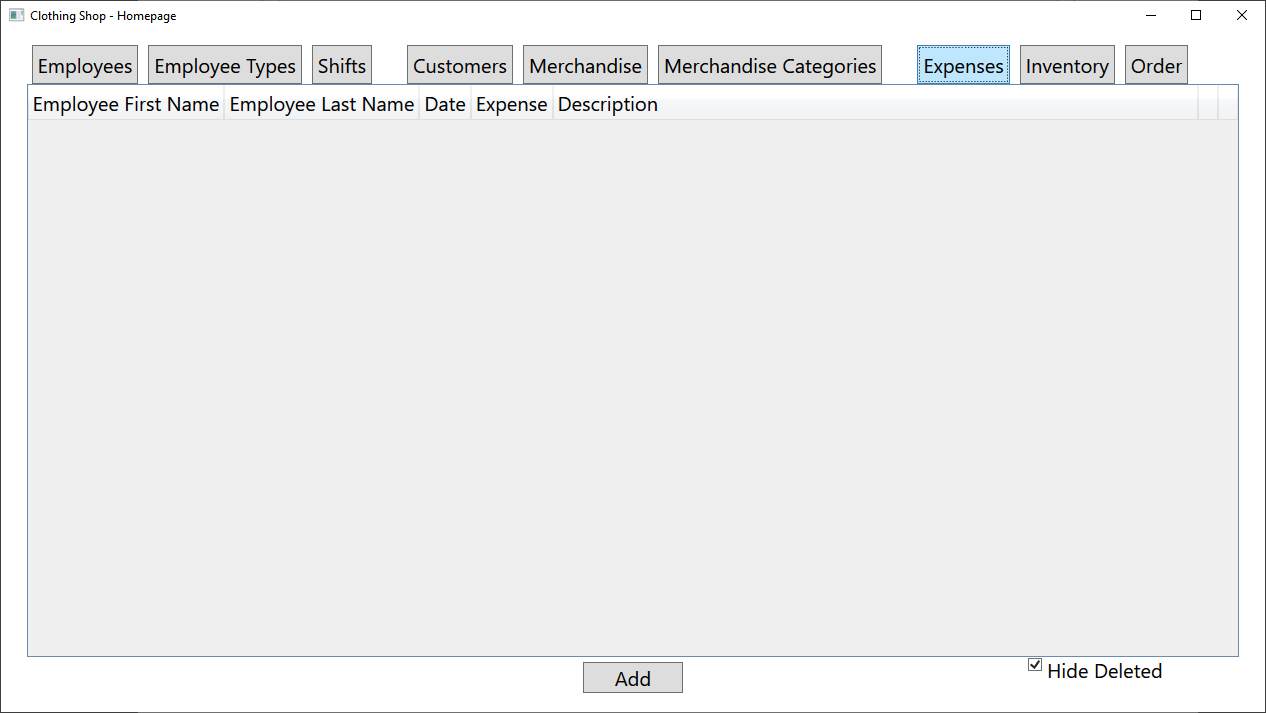


דף סוגי מוצרים, בדף זה אפשר להוסיף, לערוך או להסיר סוגי מוצרים.

## הוספת סוג מוצר

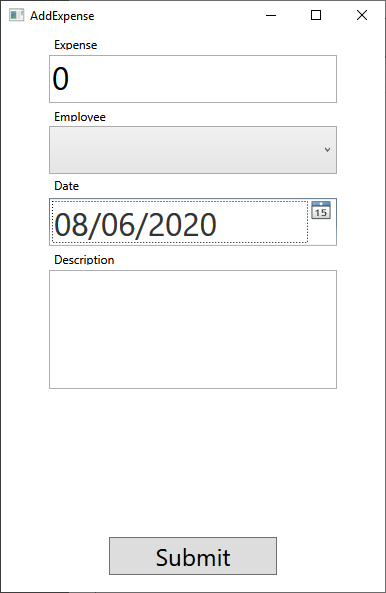


## הוצאות

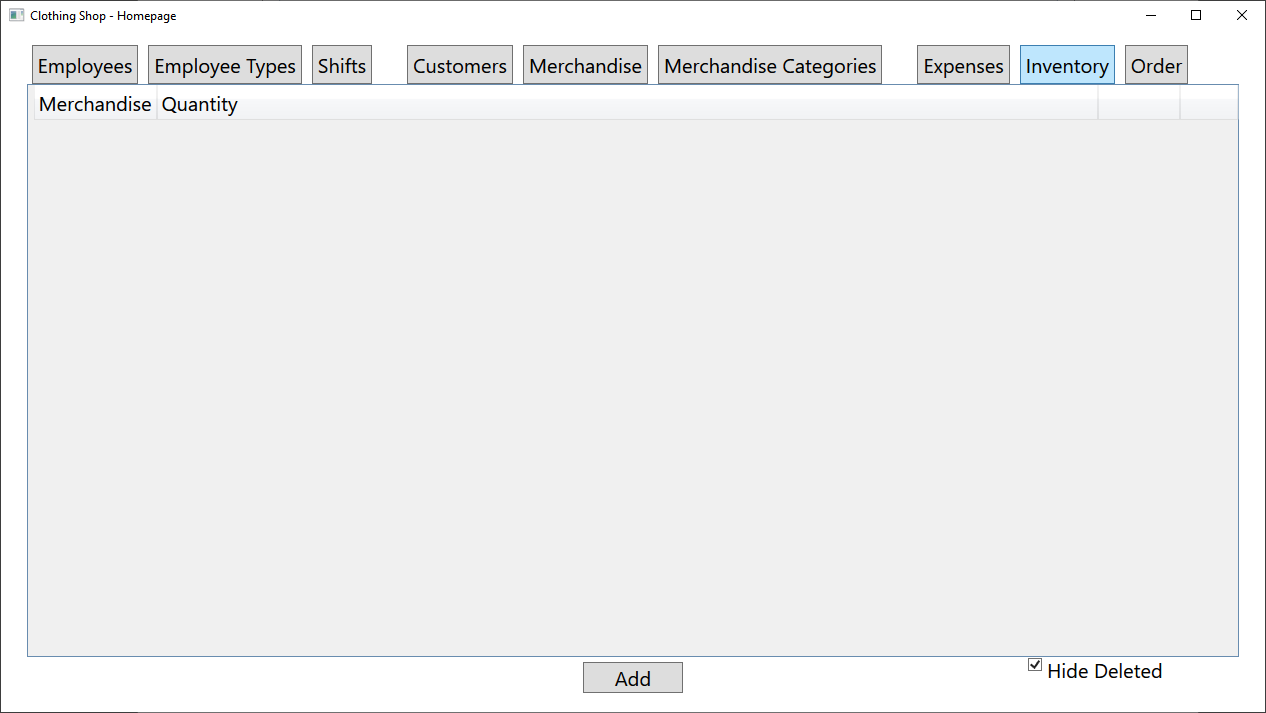


דף ההוצאות, בדך זה אפשר להוסיף, לערוך או להסיר הוצאות. דף זה יציג הוצאות אינן שוטפות.

## הוספת הוצאה

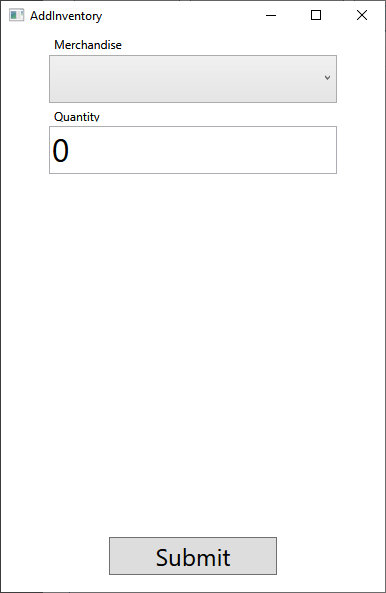


## מלאי

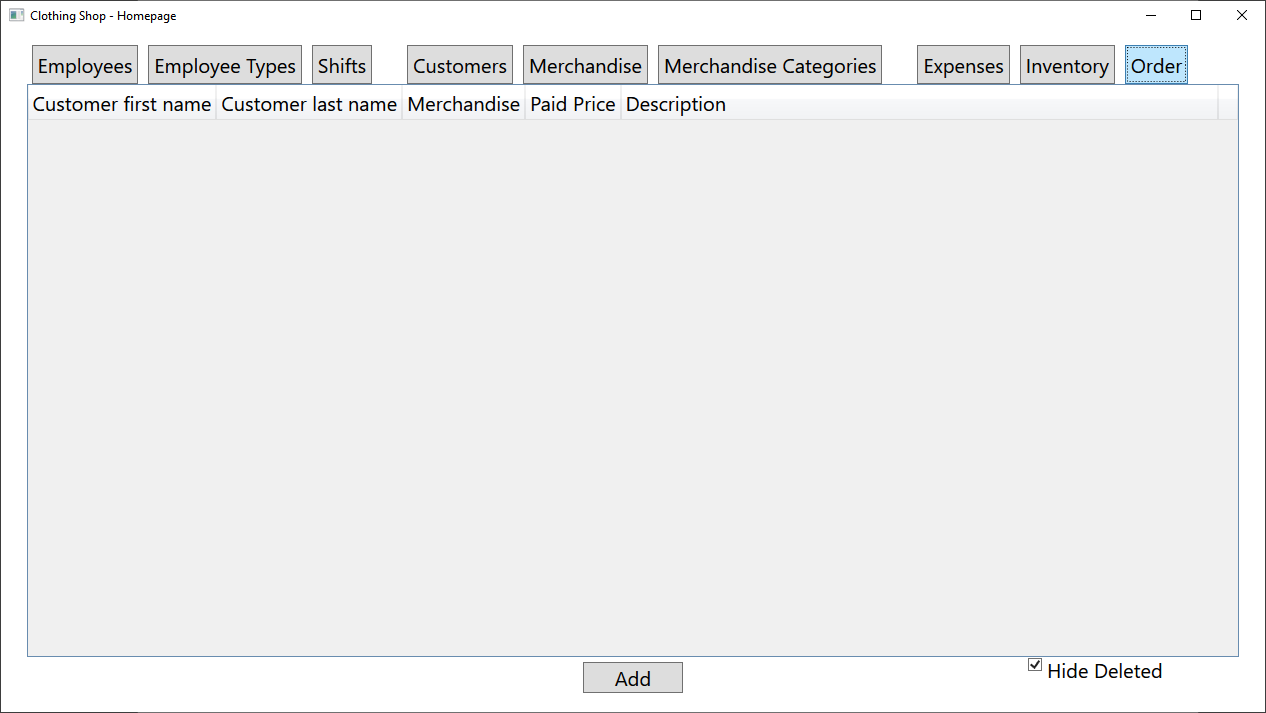


דף מלאי, בדך זה אפשר להוסיף, לערוך או להסיר מלאי קיים.

## הוספה למלאי

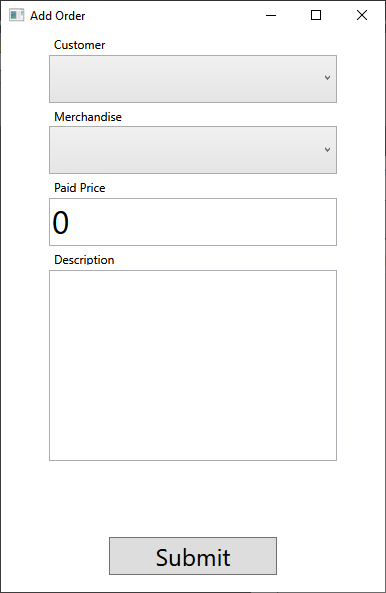


## מכירות



דף מכירות, בדך זה אפשר להוסיף, לערוך או להסיר מכירות.

## הוספת מכירה

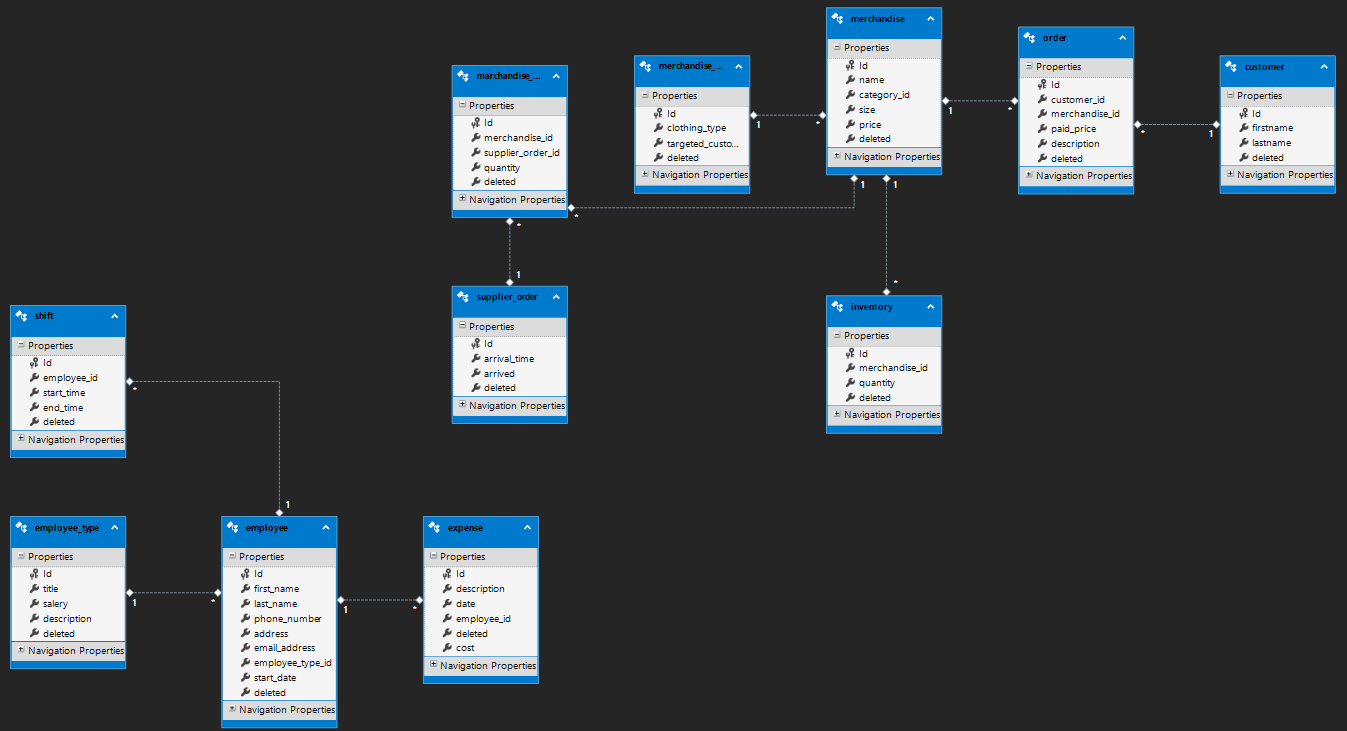


# מדריך למתכנת

## הסבר על שיטת השכבות

בתוכנה זו אני עושה שימוש בframework בשם caliburn.micro. המערכת הזו משתמשת בשיטת השכבות MVVM (Model, View, View-Model), כאשר המודל מובא לי על ידי הdatabase בצורת classes לכל טבלה שמציגים שורה מן הטבלה. שכבת ה view היא חלון userControl שמוכנס לחלון הראשי כאשר המשתמש בוחר בהצגתו. בשכבה זו קיימים קשרים בין ה XAML לשכבה הבאה, ולא נמצא קוד פונקציונלי בכלל בשכבה זו. שכבת ה view-model היא שכבת הקוד המקשרת בין ה view ל model וה database. בשכבה זו נמצא כל הקוד של הוספה, עריכה ומחיקה. Caliburn.micro נותן לי גישה מהירה למשתנים בviewmodel מתוך ה XAML של ה view בצורת binding פשוטה או אפילו בשינוי השם של הרכיב לשם של המשתנה בviewmodel.

## קשרים בין הטבלאות



## טבלת העובדים



עמודה זו מייצגת את תראיך התחלתו של העובד בחנות

עמודה זו מייצגת את ה Idשל סוג העובד

עמודה זו מייצגת את כתובת האימייל של העובד

עמודה זו מייצגת את הכתובת של העובד

עמודה זו מייצגת את שם המשפחה של העובד

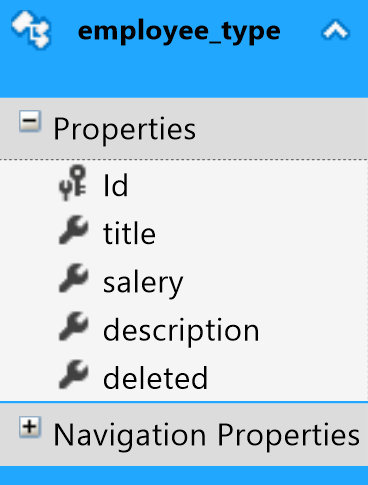
עמודה זו מייצגת את שם המשפחה של העובד

עמודה זו מייצגת את השם הפרטי של העובד

עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה

עמודה זו מציגה 0 אם העובד קיים או 1 אם העובד נמחק

## טבלת סוגי העובדים



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת את תיאור סוג עובד זה



עמודה זו מייצגת את המשכורת שסוג עובד זה מקבל



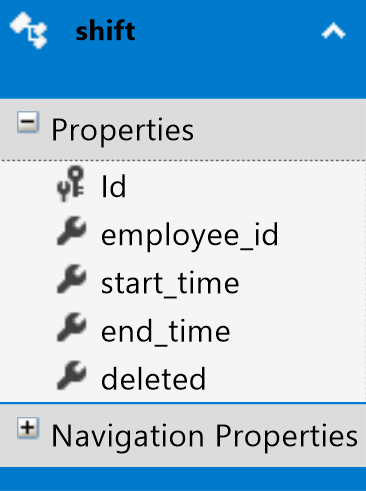
עמודה זו מייצגת את סוג העובד



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלת משמרות



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת את הזמן של סיום המשמרת



עמודה זו מייצגת את הזמן של התחלת המשמרת



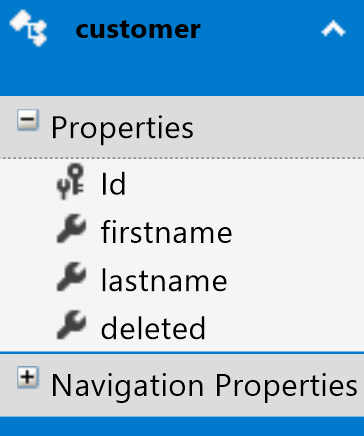
עמודה זו מייצגת את ה Idשל העובד



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלת לקוחות



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת את שם המשפחה של הלקוח



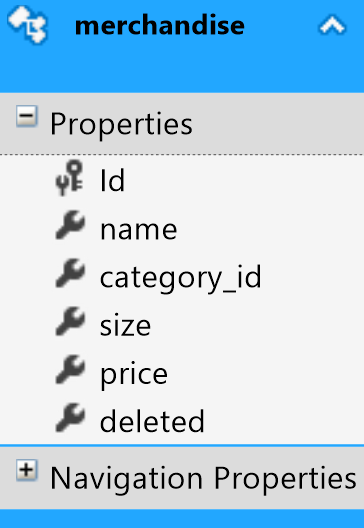
עמודה זו מייצגת את השם הפרטי של הלקוח



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלת מוצרים



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת את המחיר של המוצר



עמודה זו מייצגת את הגודל של המוצר



עמודה זו מייצגת את הId של הקטגוריה של המוצר



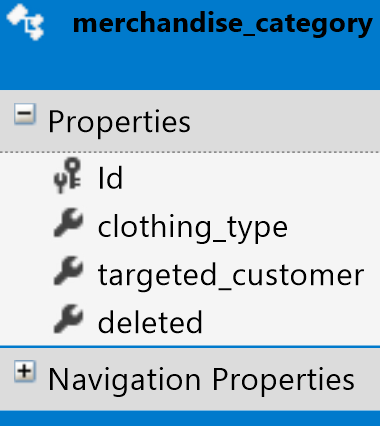
עמודה זו מייצגת את שם המוצר



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלת סוגי מוצרים



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת את הלקוח שעבורו ביגוד זה יוצר



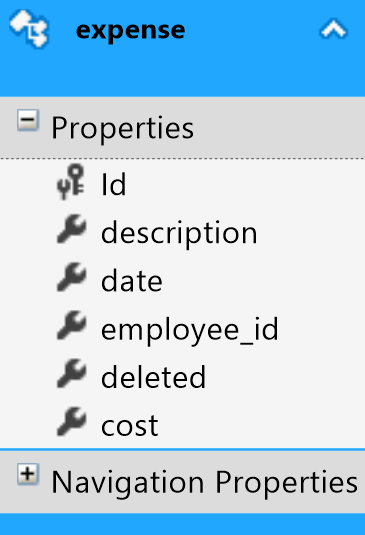
עמודה זו מייצגת את סוג הביגוד



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלת הוצאות



עמודה זו מייצגת את ערך ההוצאה



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת את הId של העובד האחראי על ההוצאה



עמודה זו מייצגת את תאריך ההוצאה



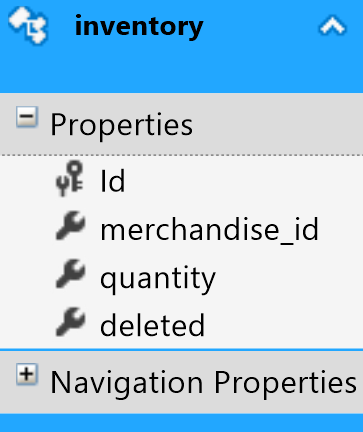
עמודה זו מייצגת את תיאור ההוצאה



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלת מלאי



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת הכמות שיש מהמוצר במלאי



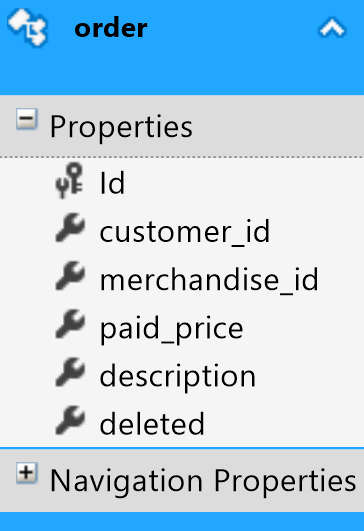
עמודה זו מייצגת את הId של המוצר



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלת המכירות



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת תיאור אופציונלי עבור מקרים מיוחדים



עמודה זו מייצגת את המחיר ששולם



עמודה זו מייצגת את הId של המוצר שנקנה



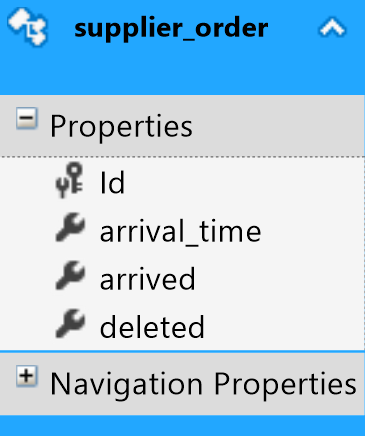
עמודה זו מייצגת את הId של הלקוח



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלת הזמנות מספק



עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת את אם ההזמנה הגיעה



עמודה זו מייצגת את הזמן שבו הגיע ההזמנה



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



## טבלה קישורת בין הזמנה מספק למוצרים בהזמנה

עמודה זו מייצגת 0 אם העובד קיים או 1 אם העובד נמחק



עמודה זו מייצגת את הכמות מהמוצר



עמודה זו מייצגת את הId של ההזמנה מהספק



עמודה זו מייצגת את הId של המוצר



עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה



# קוד התוכנה

## קוד משני

### Bootstrapper

using ClothingShop.ViewModels;

using Caliburn.Micro;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace ClothingShop

{

    class Bootstrapper : BootstrapperBase

    {

        public Bootstrapper()

        {

            Initialize();

        }

        protected override void OnStartup(object sender, StartupEventArgs e)

        {

            DisplayRootViewFor<ShellViewModel>();

        }

    }

}

### Bool Converter

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Data;

namespace ClothingShop.misc

{

    [ValueConversion(typeof(bool), typeof(bool))]

    public class InverseBooleanConverter : IValueConverter

    {

        public object Convert(object value, Type targetType, object parameter,

            System.Globalization.CultureInfo culture)

        {

            if (targetType != typeof(bool))

                throw new InvalidOperationException("The target must be a boolean");

            return !(bool)value;

        }

        public object ConvertBack(object value, Type targetType, object parameter,

            System.Globalization.CultureInfo culture)

        {

            throw new NotSupportedException();

        }

    }

}

## ViewModels

### ShellViewModel

using ClothingShop.ViewModels;

using Caliburn.Micro;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace ClothingShop.ViewModels

{

    class ShellViewModel : Conductor<object>

    {

        public Dictionary<string, Screen> ViewsDict { get; set; } = new Dictionary<string, Screen>

        {

            { "Employees", new EmployeesViewModel() },

            { "EmployeeTypes", new EmployeeTypesViewModel() },

            { "Shifts", new ShiftsViewModel() },

            { "MerchandiseCategories", new MerchandiseCategoriesViewModel() },

            { "Merchandise", new MerchandiseViewModel() },

            { "Expenses", new ExpenseViewModel() },

            { "Customers", new CustomerViewModel() },

            { "Inventory", new InventoryViewModel() },

            { "Orders", new OrderViewModel() },

            { "HomePage", new HomePageViewModel() }

        };

        public ShellViewModel()

        {

            ActivateItem(ViewsDict["HomePage"]);

        }

        public void ActivateView(string viewModelName)

        {

            DisplayName = $"Clothing Shop - {viewModelName}";

            ActivateItem(ViewsDict[viewModelName]);

        }

    }

}

### CustomerViewModel

using Caliburn.Micro;

using ClothingShop.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class CustomerViewModel : Screen, INotifyPropertyChanged

    {

        public DataHandlers.CustomerDataHandler dataHandler = new DataHandlers.CustomerDataHandler();

        public List<customer> Customers { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddCustomer()

        {

            customer cust = new customer();

            AddCustomer addWindow = new AddCustomer();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { customer = cust };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(cust);

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("Customers");

        }

        public void Delete(customer obj)

        {

            dataHandler.RemoveData(obj);

            NotifyOfPropertyChange("Customers");

        }

        public void Modify(customer cust)

        {

            AddCustomer addWindow = new AddCustomer();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { customer = cust };

            if (addWindow.ShowDialog() == true)

            {

                try

                {

                    dataHandler.AddOrUpdate(cust);

                }

                catch (Exception e)

                {

                    System.Windows.MessageBox.Show("Error: " + e.Message);

                }

            }

            else

            {

                ((System.Data.Entity.Infrastructure.IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.Refresh(System.Data.Entity.Core.Objects.RefreshMode.StoreWins, cust);

            }

            NotifyOfPropertyChange("Customers");

        }

    }

}

### EmployeesViewModel

using ClothingShop.Views;

using Caliburn.Micro;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel;

using System.Diagnostics;

using System.Windows.Input;

namespace ClothingShop.ViewModels

{

    class EmployeesViewModel : Screen, INotifyPropertyChanged

    {

        public DataHandlers.EmployeeDataHandler dataHandler = new DataHandlers.EmployeeDataHandler();

        public List<employee> Employees { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddEmployee()

        {

            employee emp = new employee();

            AddEmployee addWindow = new AddEmployee();

            emp.start\_date = DateTime.Now;

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { employee = emp, types = dataHandler.GetEntities().employee\_type.ToList().FindAll(type => { return !type.deleted; }) };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(emp);

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("Employees");

        }

        public void Delete(employee obj)

        {

            dataHandler.RemoveData(obj);

            NotifyOfPropertyChange("Employees");

        }

        public void Modify(employee emp)

        {

            AddEmployee addWindow = new AddEmployee();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { employee = emp, types = dataHandler.GetEntities().employee\_type.ToList().FindAll(type => { return !type.deleted; }) };

            if (addWindow.ShowDialog() == true)

            {

                try

                {

                    dataHandler.AddOrUpdate(emp);

                } catch (Exception e)

                {

                    System.Windows.MessageBox.Show("Error: " + e.Message);

                }

            }

            else

            {

                ((System.Data.Entity.Infrastructure.IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.Refresh(System.Data.Entity.Core.Objects.RefreshMode.StoreWins, emp);

            }

            NotifyOfPropertyChange("Employees");

        }

    }

}

### EmployeeTypesViewModel

using Caliburn.Micro;

using ClothingShop.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class EmployeeTypesViewModel : Screen, INotifyPropertyChanged

    {

        public DataHandlers.EmployeeTypesDataHandler dataHandler = new DataHandlers.EmployeeTypesDataHandler();

        public List<employee\_type> EmployeeTypes { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddEmployeeType()

        {

            employee\_type empType = new employee\_type();

            AddEmployeeType addWindow = new AddEmployeeType();

            addWindow.DataContext = new { employeeType = empType };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(empType);

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("EmployeeTypes");

        }

        public void Delete(employee\_type obj)

        {

            dataHandler.RemoveData(obj);

            NotifyOfPropertyChange("EmployeeTypes");

        }

        public void Modify(employee\_type empType)

        {

            AddEmployeeType addWindow = new AddEmployeeType();

            addWindow.DataContext = new { employeeType = empType };

            if(addWindow.ShowDialog() == true)

            {

                try

                {

                    dataHandler.AddOrUpdate(empType);

                }

                catch (Exception e)

                {

                    System.Windows.MessageBox.Show("Error: " + e.Message);

                }

            }

            else

            {

                ((System.Data.Entity.Infrastructure.IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.Refresh(System.Data.Entity.Core.Objects.RefreshMode.StoreWins, empType);

            }

            NotifyOfPropertyChange("EmployeeTypes");

        }

    }

}

### ExpenseViewModel

using Caliburn.Micro;

using ClothingShop.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class ExpenseViewModel: Screen, INotifyPropertyChanged

    {

        public DataHandlers.ExpenseDataHandler dataHandler = new DataHandlers.ExpenseDataHandler();

        public List<expense> Expenses { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddExpense()

        {

            expense exp = new expense();

            AddExpense addWindow = new AddExpense();

            exp.date = DateTime.Now;

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { expense = exp, employees = dataHandler.GetEntities().employee.ToList().FindAll(type => { return !type.deleted; }) };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(exp);

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("Expenses");

        }

        public void Delete(expense obj)

        {

            dataHandler.RemoveData(obj);

            NotifyOfPropertyChange("Expenses");

        }

        public void Modify(expense exp)

        {

            AddExpense addWindow = new AddExpense();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { expense = exp, employees = dataHandler.GetEntities().employee.ToList().FindAll(type => { return !type.deleted; }) };

            if (addWindow.ShowDialog() == true)

            {

                try

                {

                    dataHandler.AddOrUpdate(exp);

                }

                catch (Exception e)

                {

                    System.Windows.MessageBox.Show("Error: " + e.Message);

                }

            }

            else

            {

                ((System.Data.Entity.Infrastructure.IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.Refresh(System.Data.Entity.Core.Objects.RefreshMode.StoreWins, exp);

            }

            NotifyOfPropertyChange("Expenses");

        }

    }

}

### HomePageViewModel

using Caliburn.Micro;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class HomePageViewModel : Screen

    {

    }

}

### InventoryViewModel

using Caliburn.Micro;

using ClothingShop.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class InventoryViewModel : Screen, INotifyPropertyChanged

    {

        public DataHandlers.InventoryDataHandler dataHandler = new DataHandlers.InventoryDataHandler();

        public List<inventory> Inventory { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddInventory()

        {

            inventory inv = new inventory();

            AddInventory addWindow = new AddInventory();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { inventory = inv, merchandise = dataHandler.GetEntities().merchandise.ToList().FindAll(type => { return !type.deleted; }) };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(inv);

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("Inventory");

        }

        public void Delete(inventory obj)

        {

            dataHandler.RemoveData(obj);

            NotifyOfPropertyChange("Inventory");

        }

        public void Modify(inventory inv)

        {

            AddInventory addWindow = new AddInventory();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { inventory = inv, merchandise = dataHandler.GetEntities().merchandise.ToList().FindAll(type => { return !type.deleted; }) };

            if (addWindow.ShowDialog() == true)

            {

                try

                {

                    dataHandler.AddOrUpdate(inv);

                }

                catch (Exception e)

                {

                    System.Windows.MessageBox.Show("Error: " + e.Message);

                }

            }

            else

            {

                ((System.Data.Entity.Infrastructure.IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.Refresh(System.Data.Entity.Core.Objects.RefreshMode.StoreWins, inv);

            }

            NotifyOfPropertyChange("Inventory");

        }

    }

}

### MerchandiseCategoriesViewModel

using Caliburn.Micro;

using ClothingShop.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class MerchandiseCategoriesViewModel : Screen, INotifyPropertyChanged

    {

        public DataHandlers.MerchandiseCategoryDataHandler dataHandler = new DataHandlers.MerchandiseCategoryDataHandler();

        public List<merchandise\_category> MerchandiseCategories { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddMerchandiseCategory()

        {

            merchandise\_category merchCategory = new merchandise\_category();

            AddMerchandiseCategory addWindow = new AddMerchandiseCategory();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { merchCategory };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(merchCategory);

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("MerchandiseCategories");

        }

        public void Delete(merchandise\_category obj)

        {

            dataHandler.RemoveData(obj);

            NotifyOfPropertyChange("MerchandiseCategories");

        }

        public void Modify(merchandise\_category merchCategory)

        {

            AddMerchandiseCategory addWindow = new AddMerchandiseCategory();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { merchCategory };

            if (addWindow.ShowDialog() == true)

            {

                try

                {

                    dataHandler.AddOrUpdate(merchCategory);

                }

                catch (Exception e)

                {

                    System.Windows.MessageBox.Show("Error: " + e.Message);

                }

            }

            else

            {

                ((System.Data.Entity.Infrastructure.IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.Refresh(System.Data.Entity.Core.Objects.RefreshMode.StoreWins, merchCategory);

            }

            NotifyOfPropertyChange("MerchandiseCategories");

        }

    }

}

### MerchandiseViewModel

using Caliburn.Micro;

using ClothingShop.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class MerchandiseViewModel : Screen, INotifyPropertyChanged

    {

        public DataHandlers.MerchandiseDataHandler dataHandler = new DataHandlers.MerchandiseDataHandler();

        public List<merchandise> Merchandise { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddMerchandise()

        {

            merchandise merch = new merchandise();

            AddMerchandise addWindow = new AddMerchandise();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { merch, merchCategories = dataHandler.GetEntities().merchandise\_category.ToList().FindAll(type => { return !type.deleted; }) };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(merch);

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("Merchandise");

        }

        public void Delete(merchandise obj)

        {

            dataHandler.RemoveData(obj);

            NotifyOfPropertyChange("Merchandise");

        }

        public void Modify(merchandise merch)

        {

            AddMerchandise addWindow = new AddMerchandise();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { merch, merchCategories = dataHandler.GetEntities().merchandise\_category.ToList().FindAll(type => { return !type.deleted; }) };

            if (addWindow.ShowDialog() == true)

            {

                try

                {

                    dataHandler.AddOrUpdate(merch);

                }

                catch (Exception e)

                {

                    System.Windows.MessageBox.Show("Error: " + e.Message);

                }

            }

            else

            {

                ((System.Data.Entity.Infrastructure.IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.Refresh(System.Data.Entity.Core.Objects.RefreshMode.StoreWins, merch);

            }

            NotifyOfPropertyChange("Merchandise");

        }

    }

}

### OrderViewModel

using Caliburn.Micro;

using ClothingShop.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class OrderViewModel : Screen, INotifyPropertyChanged

    {

        public DataHandlers.OrderDataHandler dataHandler = new DataHandlers.OrderDataHandler();

        public List<order> Orders { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddOrder()

        {

            order order = new order();

            AddOrder addWindow = new AddOrder();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { order, customers = dataHandler.GetEntities().customer.ToList().FindAll(type => { return !type.deleted; }), merchandise = dataHandler.GetEntities().merchandise.ToList().FindAll(type => { return !type.deleted; }) };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(order);

                // Deduct from inventory

                List<inventory> invList = dataHandler.GetEntities().inventory.ToList();

                foreach (inventory inv in invList)

                {

                    if(inv.merchandise\_id == order.merchandise\_id)

                    {

                        inv.quantity -= 1;

                        dataHandler.GetEntities().SaveChanges();

                        NotifyOfPropertyChange("Inventory");

                        break;

                    }

                }

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("Orders");

        }

        public void Delete(order obj)

        {

            dataHandler.RemoveData(obj);

            NotifyOfPropertyChange("Orders");

        }

    }

}

### ShiftsViewModel

using Caliburn.Micro;

using ClothingShop.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.ViewModels

{

    class ShiftsViewModel : Screen, INotifyPropertyChanged

    {

        public DataHandlers.ShiftDataHandler dataHandler = new DataHandlers.ShiftDataHandler();

        public List<shift> Shifts { get { return dataHandler.GetData(HideDeleted); } }

        public bool HideDeleted { get; set; } = true;

        public void AddShift()

        {

            shift s = new shift();

            AddShift addWindow = new AddShift();

            s.end\_time = DateTime.Now;

            s.start\_time = DateTime.Now;

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { shift = s, employees = dataHandler.GetEntities().employee.ToList().FindAll(type => { return !type.deleted; }) };

            addWindow.ShowDialog();

            try

            {

                dataHandler.AddOrUpdate(s);

            }

            catch (Exception e)

            {

                System.Windows.MessageBox.Show("Error: " + e.Message);

            }

            NotifyOfPropertyChange("Shifts");

        }

        public void Delete(shift s)

        {

            dataHandler.RemoveData(s);

            NotifyOfPropertyChange("Shifts");

        }

        public void Modify(shift s)

        {

            AddShift addWindow = new AddShift();

            // Sets the window's context and gives it the types available, that aren't deleted

            addWindow.DataContext = new { shift = s, employees = dataHandler.GetEntities().employee.ToList().FindAll(type => { return !type.deleted; }) };

            if (addWindow.ShowDialog() == true)

            {

                try

                {

                    dataHandler.AddOrUpdate(s);

                }

                catch (Exception e)

                {

                    System.Windows.MessageBox.Show("Error: " + e.Message);

                }

            }

            else

            {

                ((System.Data.Entity.Infrastructure.IObjectContextAdapter)dataHandler.GetEntities()).ObjectContext.Refresh(System.Data.Entity.Core.Objects.RefreshMode.StoreWins, s);

            }

            NotifyOfPropertyChange("Shifts");

        }

    }

}

## DataHandlers

### Abstract dataHandler

using System;

using System.Collections.Generic;

using System.Data.Entity.Migrations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop

{

    abstract class DataHandler<T> where T : class

    {

        static protected dbEntities db = new dbEntities();

        public virtual List<T> GetData(bool hideDeleted)

        {

            if (hideDeleted)

            {

                return db.Set<T>().ToList().FindAll(o => !(bool)(typeof(T).GetProperty("deleted").GetValue(o)));

            }

            else

            {

                return db.Set<T>().ToList();

            }

        }

        public virtual void RemoveData(T item)

        {

            // Set the data item as deleted

            typeof(T).GetProperty("deleted").SetValue(item, true);

            // Notify db that the item was changed

            db.Entry(item).State = System.Data.Entity.EntityState.Modified;

            // Save the database

            db.SaveChanges();

        }

        public virtual void AddOrUpdate(T item)

        {

            try

            {

                VerifyItem(item);

                // Add item to database

                db.Set<T>().AddOrUpdate(item);

                // Save the database

                db.SaveChanges();

            }

            catch(Exception e)

            {

                throw new Exception(e.Message);

            }

        }

        public dbEntities GetEntities()

        {

            return db;

        }

        protected abstract void VerifyItem(T item);

    }

}

### CustomerDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class CustomerDataHandler : DataHandler<customer>

    {

        private bool VerifyString(string name)

        {

            Regex nameRegex = new Regex(@"^([a-zA-Z]+?)([-\s'][a-zA-Z]+)\*?$");

            return nameRegex.IsMatch(name);

        }

        protected override void VerifyItem(customer item)

        {

            if (!VerifyString(item.firstname))

            {

                throw new Exception("First name incorrect");

            } else if (!VerifyString(item.lastname))

            {

                throw new Exception("Last name incorrect");

            }

        }

    }

}

### EmployeeDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class EmployeeDataHandler : DataHandler<employee>

    {

        private bool VerifyString(string name)

        {

            Regex nameRegex = new Regex(@"^([a-zA-Z]+?)([-\s'][a-zA-Z]+)\*?$");

            return nameRegex.IsMatch(name);

        }

        private bool VerifyPhone(string phone)

        {

            Regex phoneRegex = new Regex(@"^\+?(972|0)(\-)?0?(([23489]{1}\d{7})|[5]{1}\d{8})$");

            return phoneRegex.IsMatch(phone);

        }

        protected override void VerifyItem(employee item)

        {

            if(!VerifyString(item.first\_name))

            {

                throw new Exception("First name incorrect");

            }

            else if (!VerifyString(item.last\_name))

            {

                throw new Exception("Last name incorrect");

            }

            else if (!VerifyPhone(item.phone\_number))

            {

                throw new Exception("Phone number incorrect");

            }

            else if (!VerifyString(item.address))

            {

                throw new Exception("Address incorrect");

            }

        }

    }

}

### EmployeeTypesDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class EmployeeTypesDataHandler : DataHandler<employee\_type>

    {

        private bool VerifyString(string name)

        {

            Regex nameRegex = new Regex(@"^([a-zA-Z]+?)([-\s'][a-zA-Z]+)\*?$");

            return nameRegex.IsMatch(name);

        }

        private bool VerifySalery(double salery)

        {

            return salery != 0;

        }

        protected override void VerifyItem(employee\_type type)

        {

            if (!VerifyString(type.title))

            {

                throw new Exception("title incorrect");

            }

            else if (!VerifySalery(type.salery))

            {

                throw new Exception("Salery Incorrect");

            }

        }

    }

}

### ExpenseDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class ExpenseDataHandler : DataHandler<expense>

    {

        protected override void VerifyItem(expense item)

        {

        }

    }

}

### InventoryDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class InventoryDataHandler : DataHandler<inventory>

    {

        protected override void VerifyItem(inventory item)

        {

        }

    }

}

### MerchandiseCategoryDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class MerchandiseCategoryDataHandler : DataHandler<merchandise\_category>

    {

        private bool VerifyString(string name)

        {

            Regex nameRegex = new Regex(@"^([a-zA-Z]+?)([-\s'][a-zA-Z]+)\*?$");

            return nameRegex.IsMatch(name);

        }

        protected override void VerifyItem(merchandise\_category item)

        {

            if (!VerifyString(item.clothing\_type))

            {

                throw new Exception("Clothing type incorrect");

            }

            else if (!VerifyString(item.targeted\_customer))

            {

                throw new Exception("Targeted customer incorrect");

            }

        }

    }

}

### MerchandiseDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class MerchandiseDataHandler : DataHandler<merchandise>

    {

        private bool VerifyString(string name)

        {

            Regex nameRegex = new Regex(@"^([a-zA-Z]+?)([-\s'][a-zA-Z]+)\*?$");

            return nameRegex.IsMatch(name);

        }

        protected override void VerifyItem(merchandise item)

        {

            if (!VerifyString(item.name))

            {

                throw new Exception("Name incorrect");

            }

            else if (!VerifyString(item.merchandise\_category.clothing\_type))

            {

                throw new Exception("Clothing type incorrect");

            }

            else if (!VerifyString(item.merchandise\_category.targeted\_customer))

            {

                throw new Exception("targeted customer incorrect");

            }

            else if (!VerifyString(item.size))

            {

                throw new Exception("size incorrect");

            }

        }

    }

}

### OrderDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class OrderDataHandler : DataHandler<order>

    {

        protected override void VerifyItem(order item)

        {

        }

    }

}

### ShiftsDataHandler

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClothingShop.DataHandlers

{

    class ShiftDataHandler : DataHandler<shift>

    {

        protected override void VerifyItem(shift item)

        {

        }

    }

}

# סיכום

בכל מהלך בניית התוכנה למדתי רבות. למדתי איך להתמודד עם בעיות ותקלות אפילו אם הן קשות ומסובכות, איך משתמשים במערכת caliburn.micro, בשיטת השכבות MVVM והעמקתי המון את הידע שלי על WPF וC# ובנוסף למדתי גם את הבסיס של Regex, אך בעיקר למדתי איך ללמוד, להתמודד בעצמי ולחפש בעצמי פתרון. לסיכום בניית תוכנה זו הייתה חוויה מהנה ומאוד מעשירה להמשך החיים.