מערכת לניהול חנות צילום

מגיש: רועי רפאלי

מדריך: עוז ארווץ

בית ספר: מקיף י"א אשדוד

כיתה: יב'4

תעודת זהות: 212766281

תוכן

מבוא

נושא הפרוייקט:

הקמה של מערכת ממוחשבת כללית למודל חנות ציוד צילום.

מהות התוכנה:

תוכנה זו נכתבה במטרה לייעל את דרכי ההתנהלות של חנויות לציוד צילום.

תהליכים מרכזיים

לנהל מכירות

בעזרת תהליך זה יכולה החנות לרשום את פרטי המכירות.

לנהל מלאי

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

לנהל עובדים

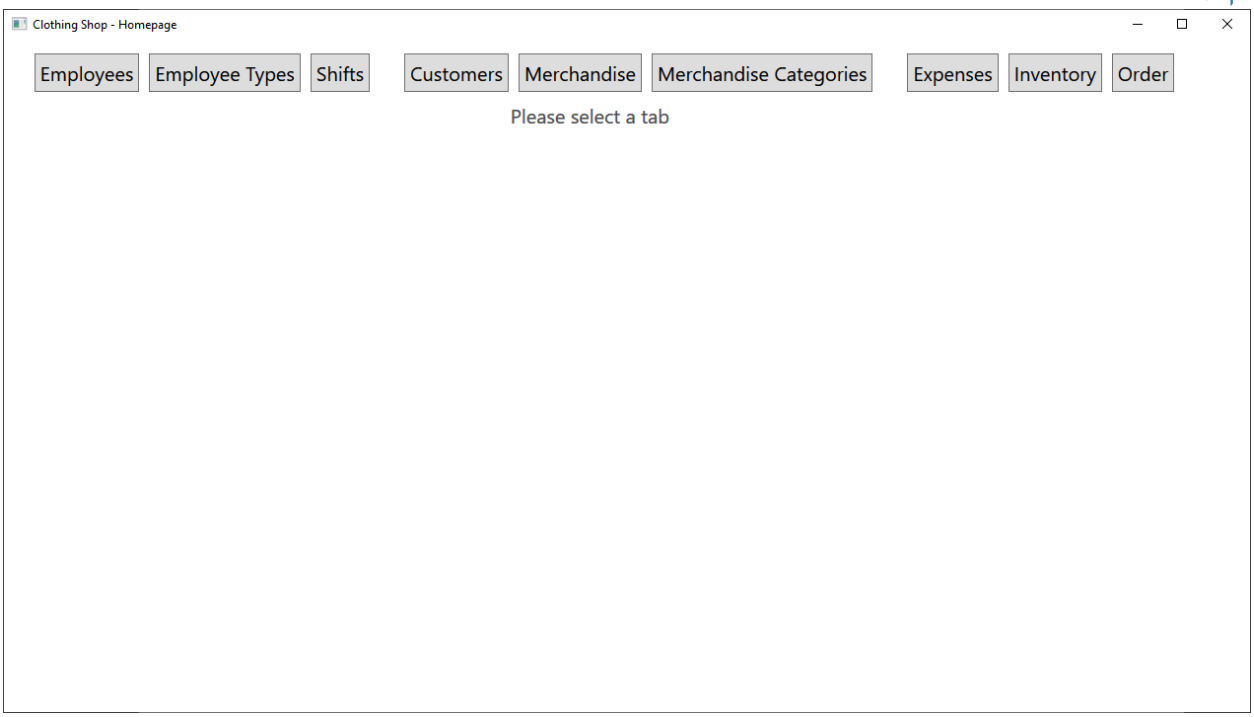
בעזרת תהליך זה יכולה החנות לרשום את עובדיו לעסק, לפטרם וכן לערוך פרטיהם.

לנהל לקוחות

בעזרת תהליך זה יכולה החנות לשמור את הפרטים של הלקוחות.

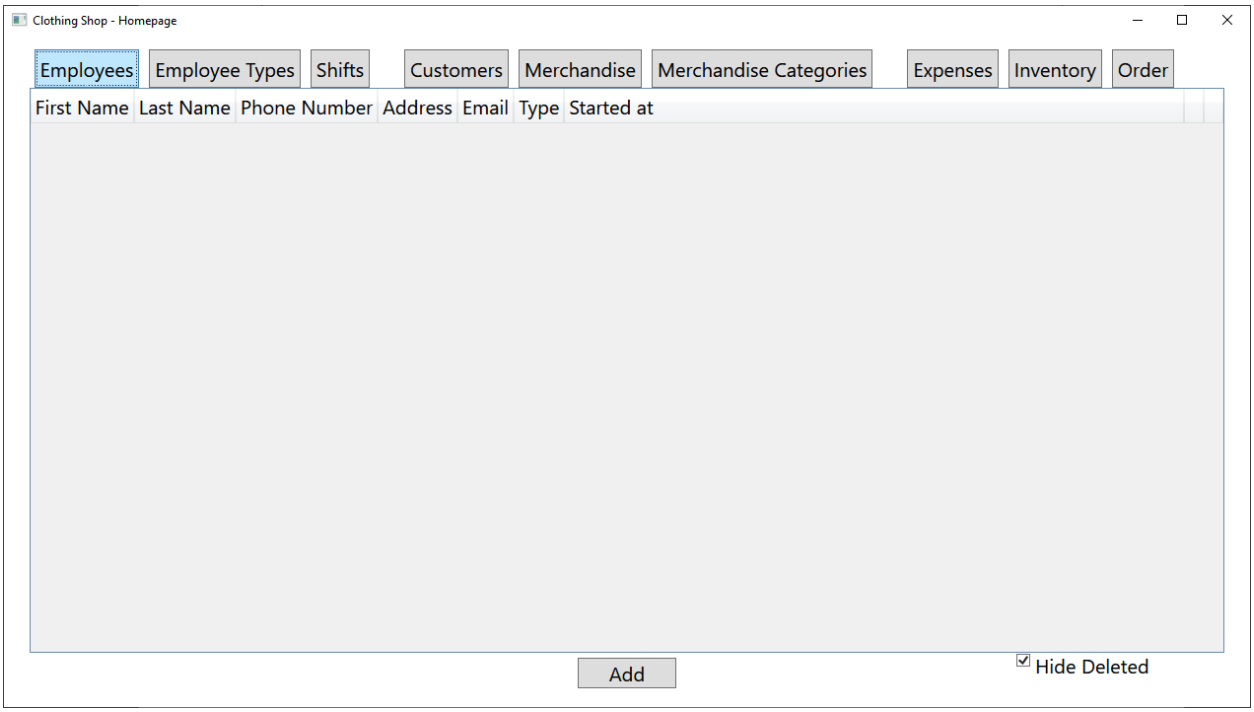
מדריך למשתמש

דף ראשי



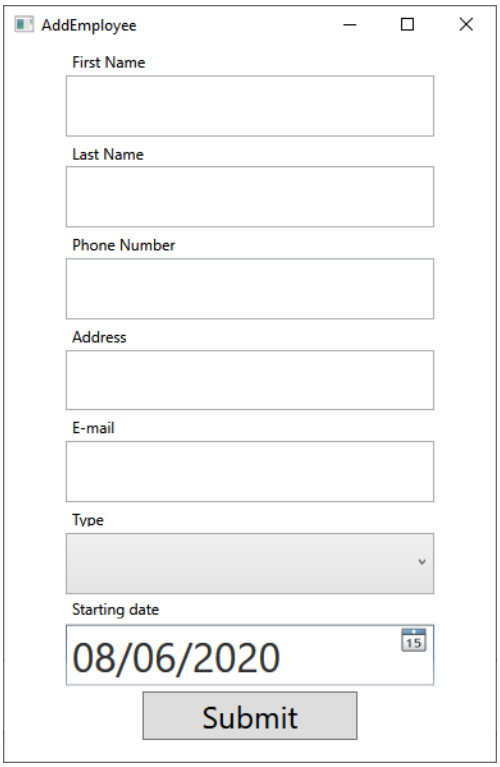
דף זה משמש כמסך הבית של התוכנה, דרך דף זה אפשר להגיע לכל האפשרויות בתוכנה

עובדים

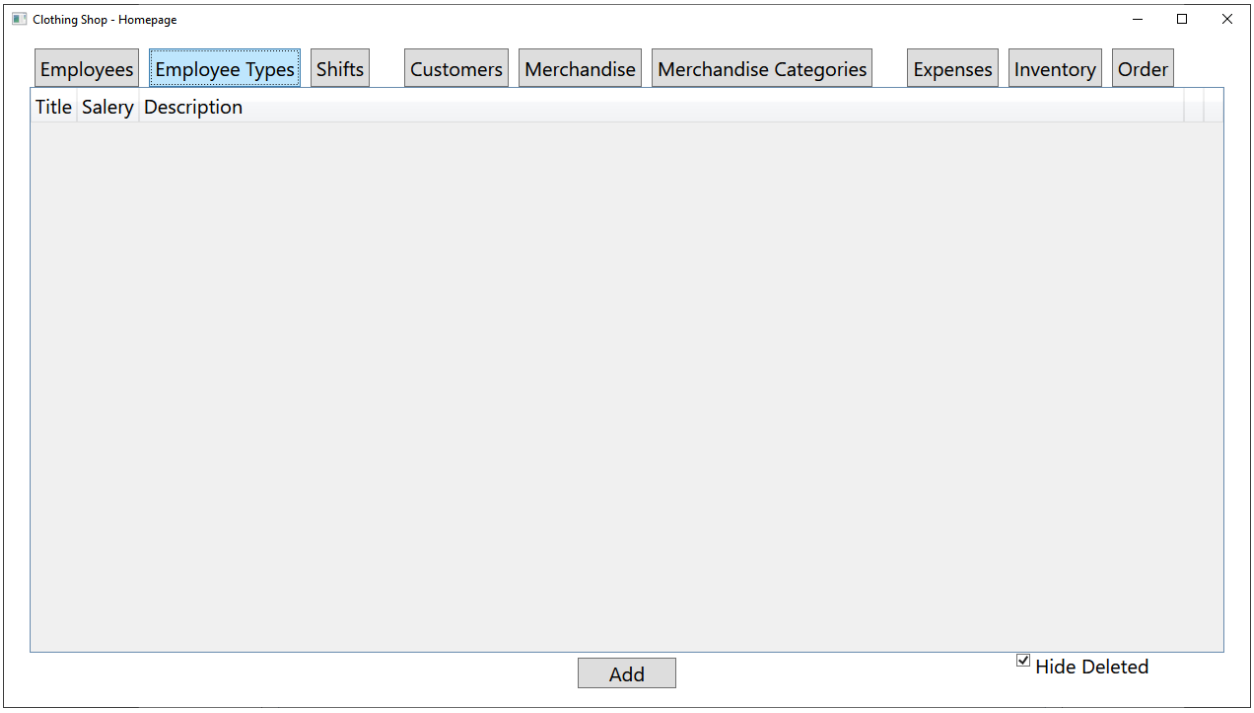


דף העובדים, בדף זה אפשר להוסיף, לערוך או להסיר עובדים.

הוספת עובד



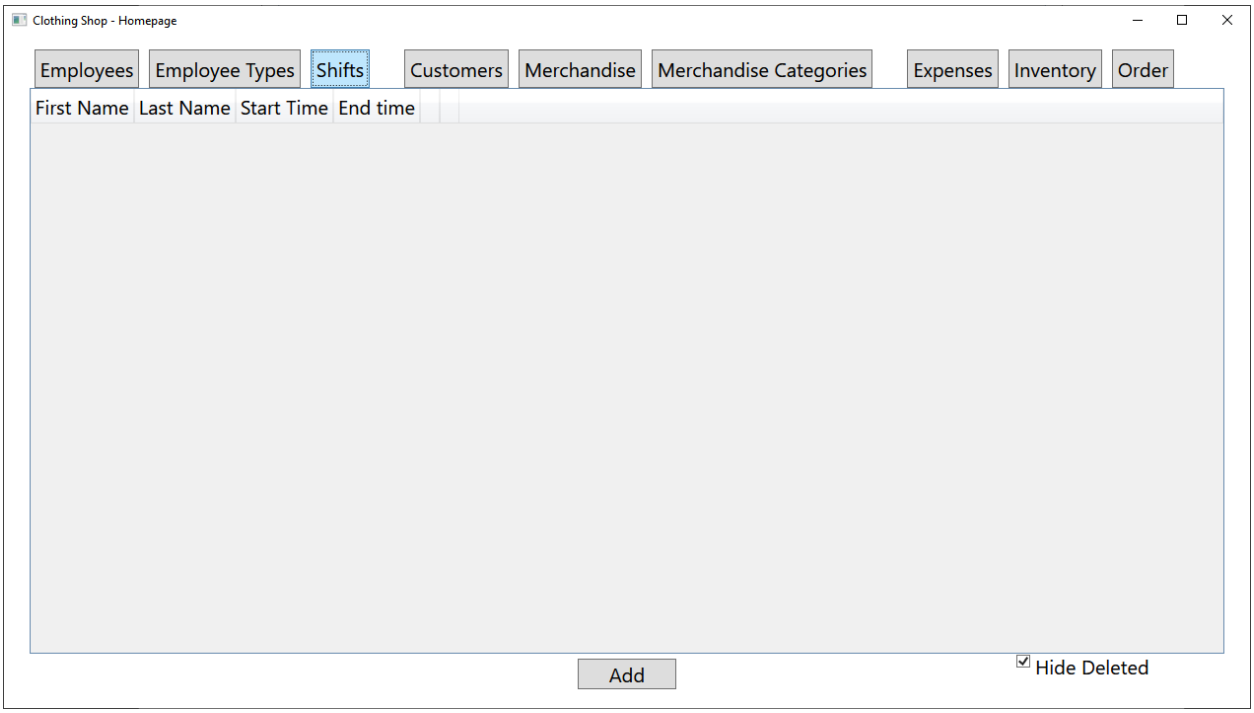
סוגי עובדים



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

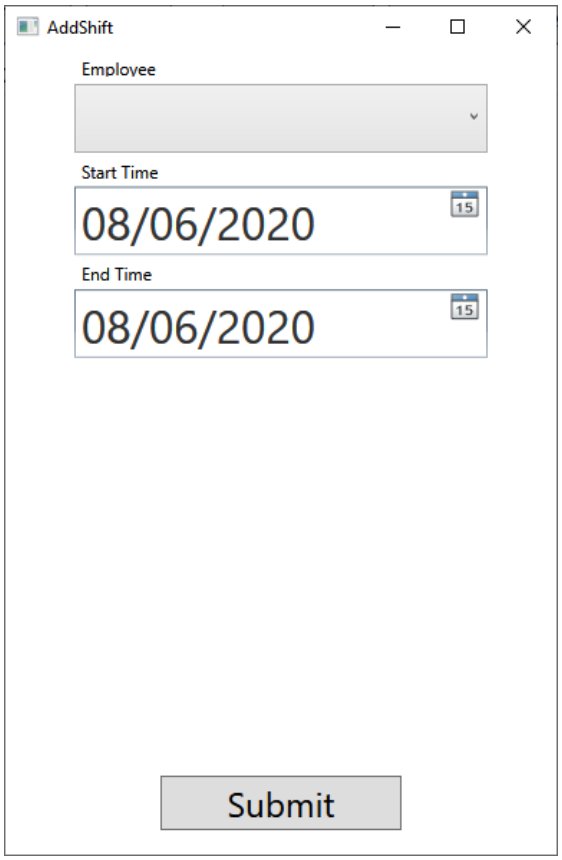
הוספת סוג עובד

משמרות

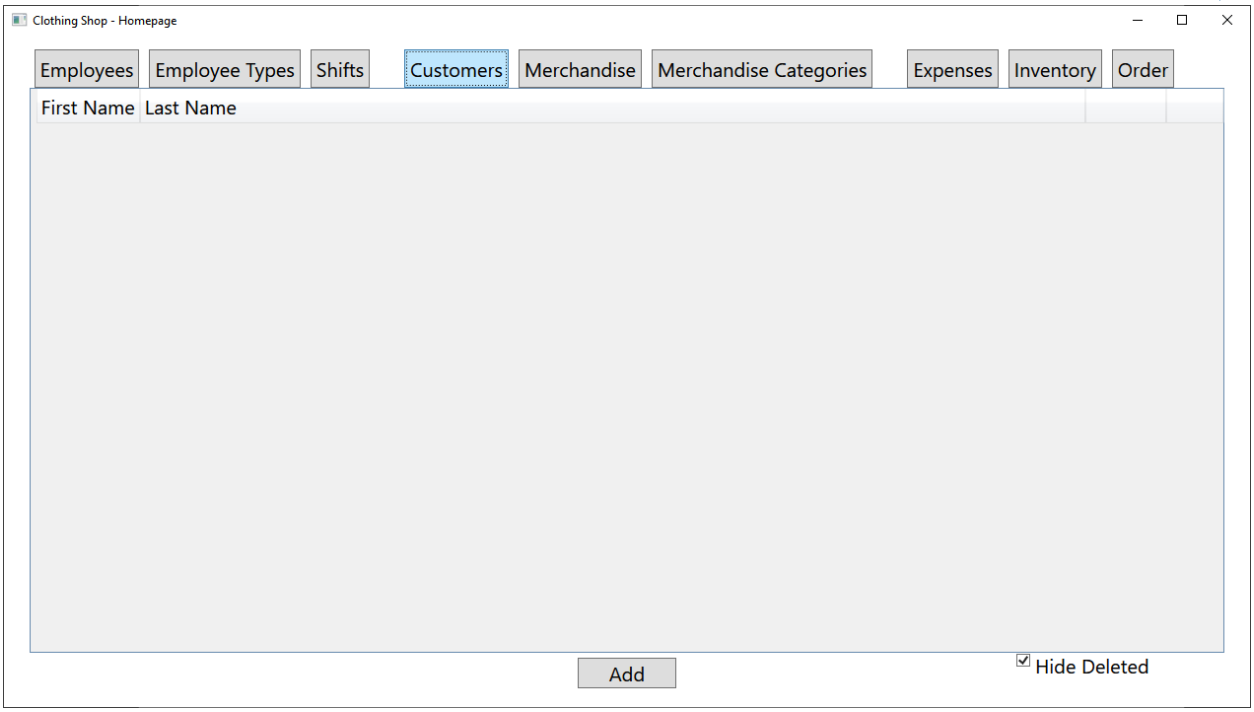


דף משמרות, בדף זה אפשר להוסיף, לערוך או להסיר משמרות לפי העובדים הקיימים.

הוספת משמרת

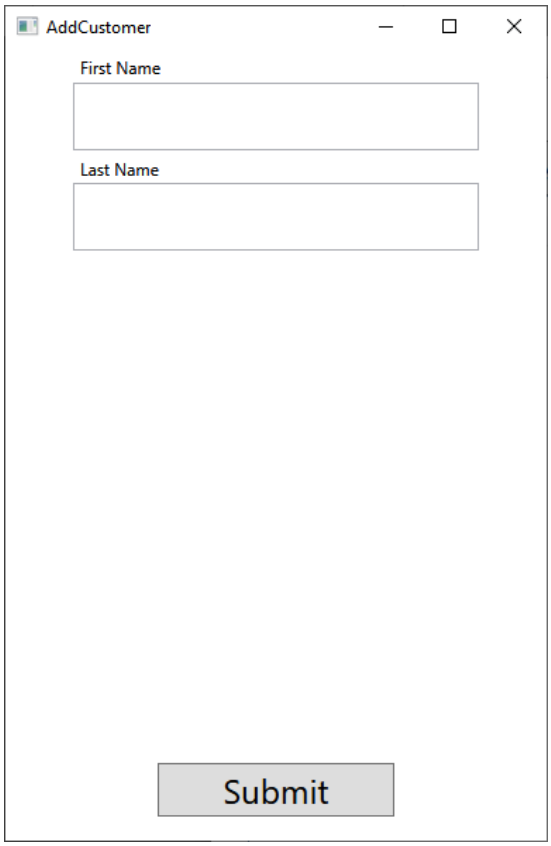


לקוחות

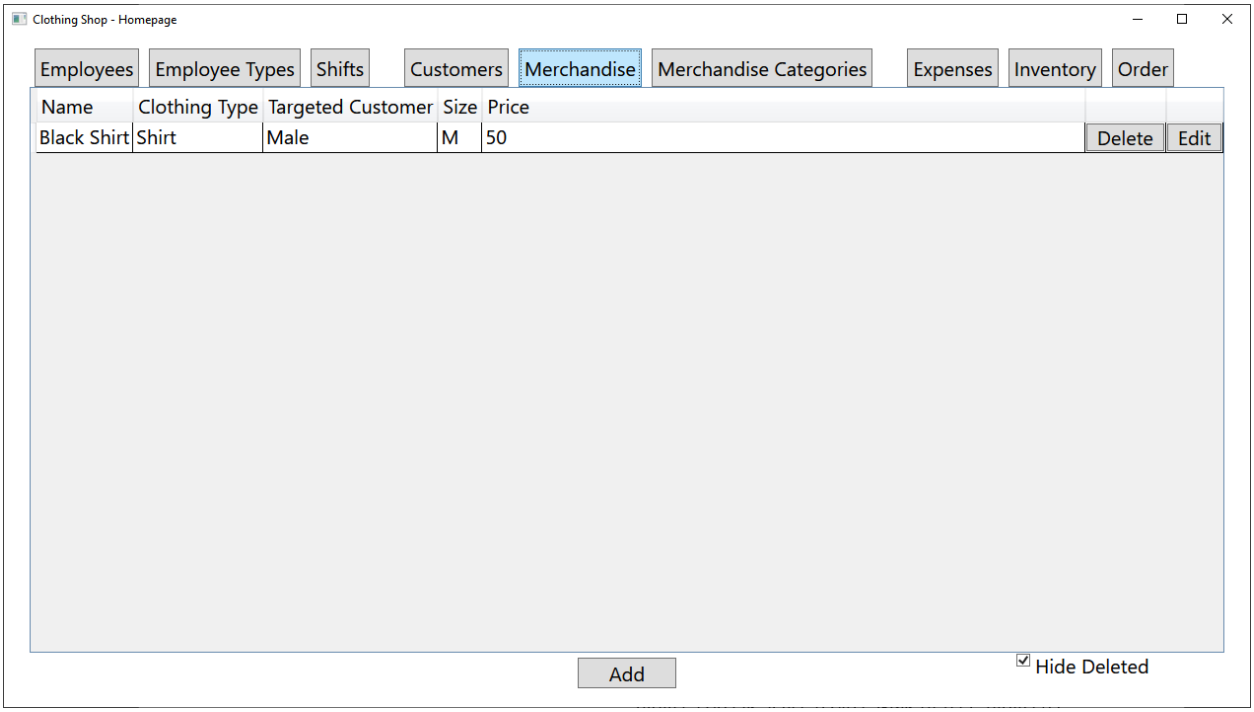


דף לקוחות, בדף זה אפשר להוסיף, לערוך או להסיר לקוחות.

הוספת לקוח

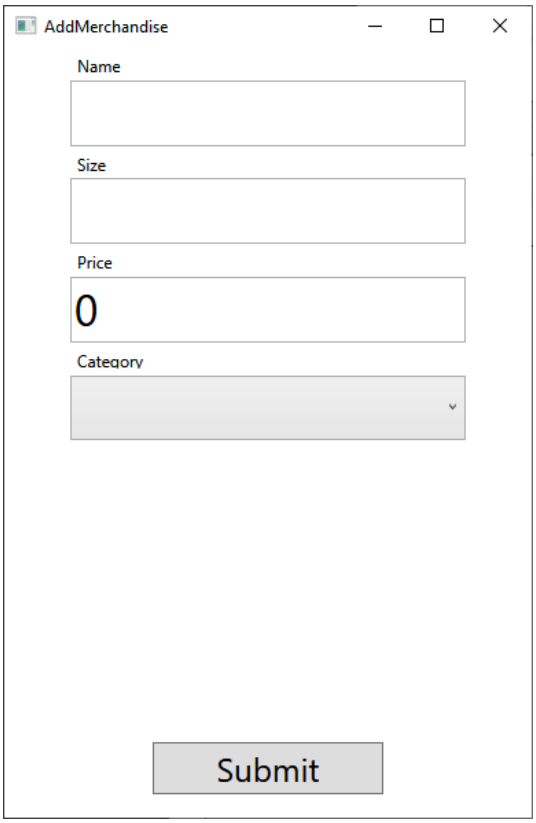


מוצרים

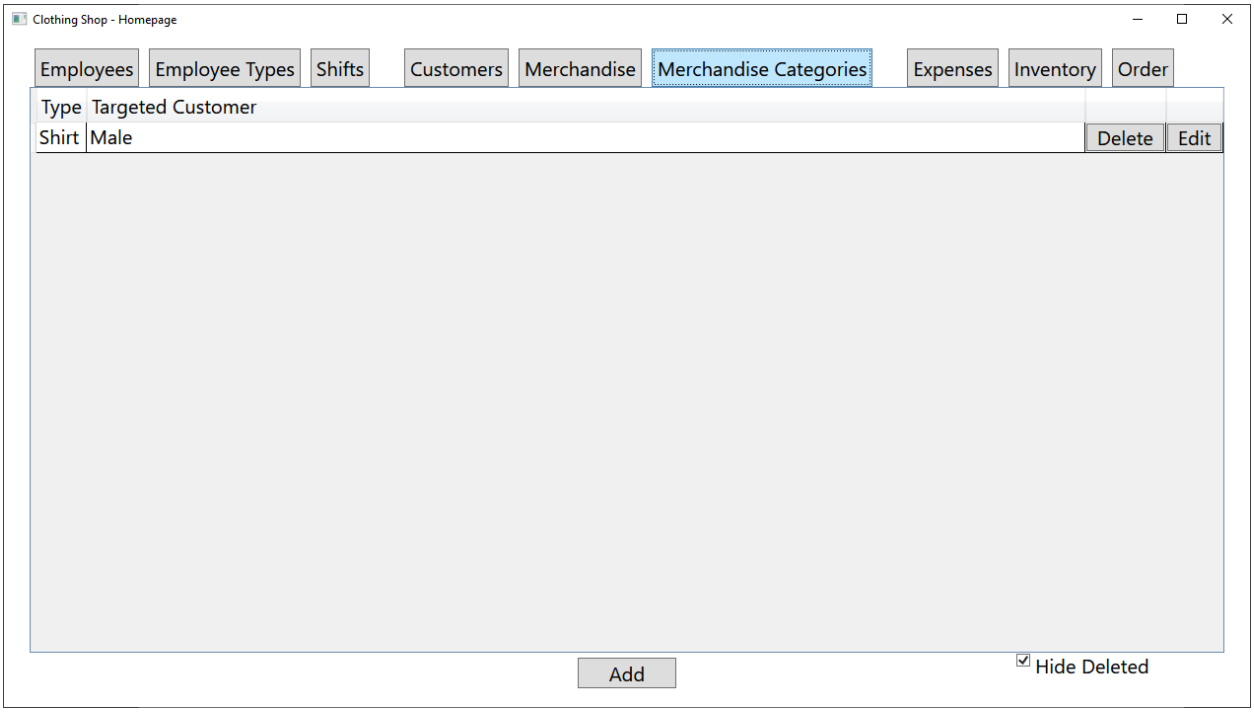


דף מוצרים, בדף זה אפשר להוסיף, לערוך או להסיר מוצרים. המוצרים לא בהכרח קיימים בחנות, אלא היו קיימים או בדרך לחנות.

הוספת מוצר

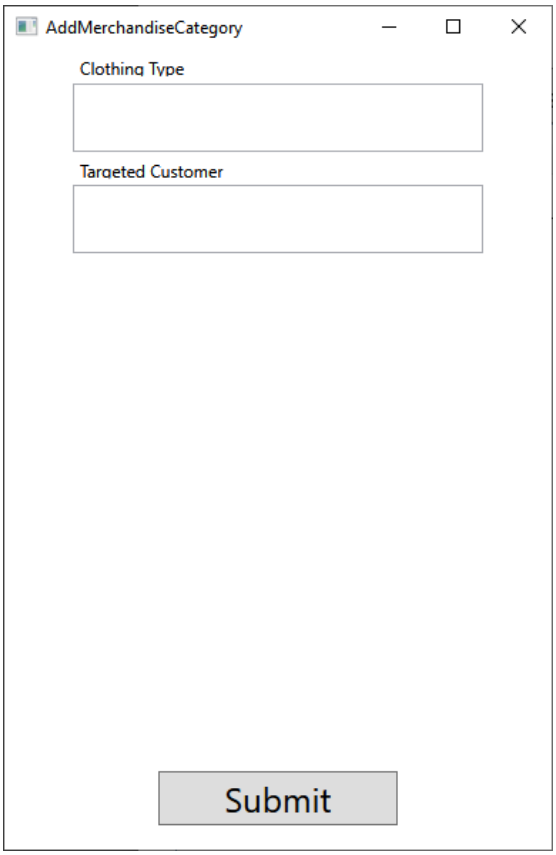


סוגי מוצרים

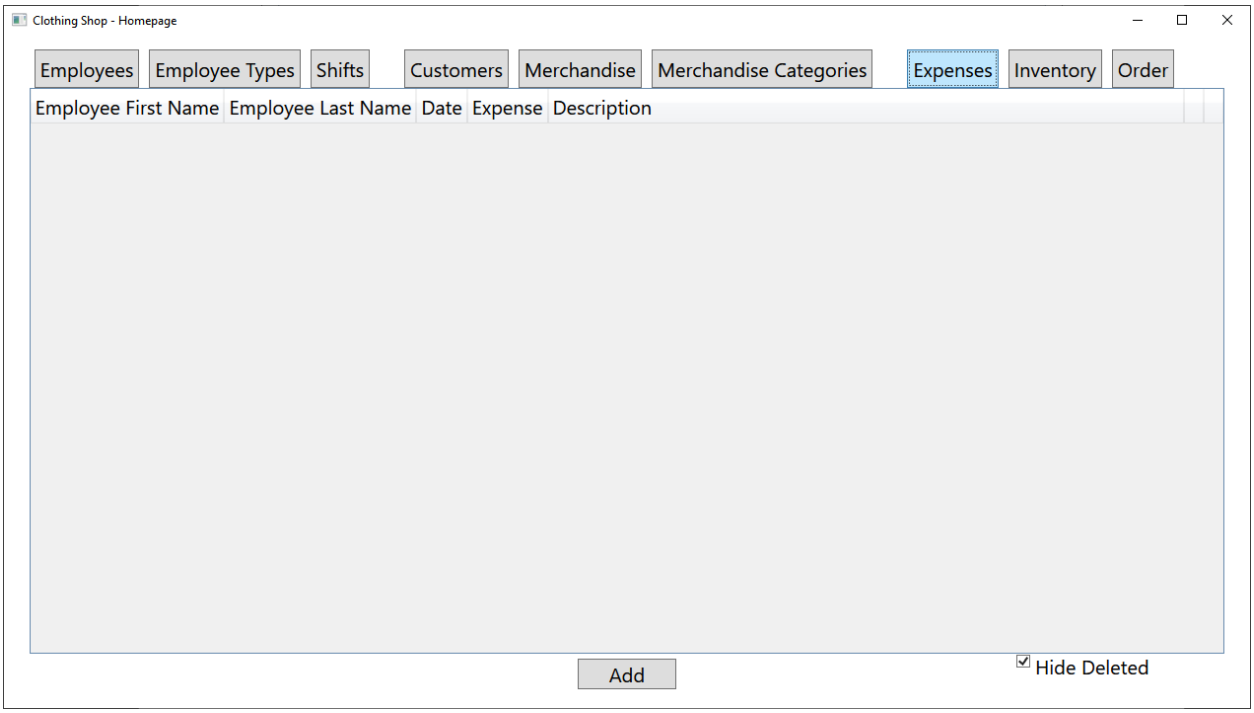


דף סוגי מוצרים, בדף זה אפשר להוסיף, לערוך או להסיר סוגי מוצרים.

הוספת סוג מוצר

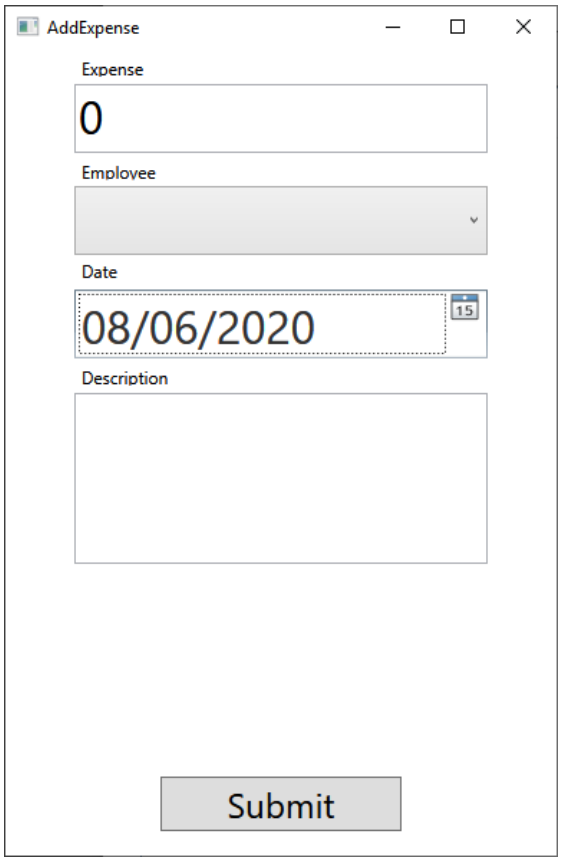


הוצאות

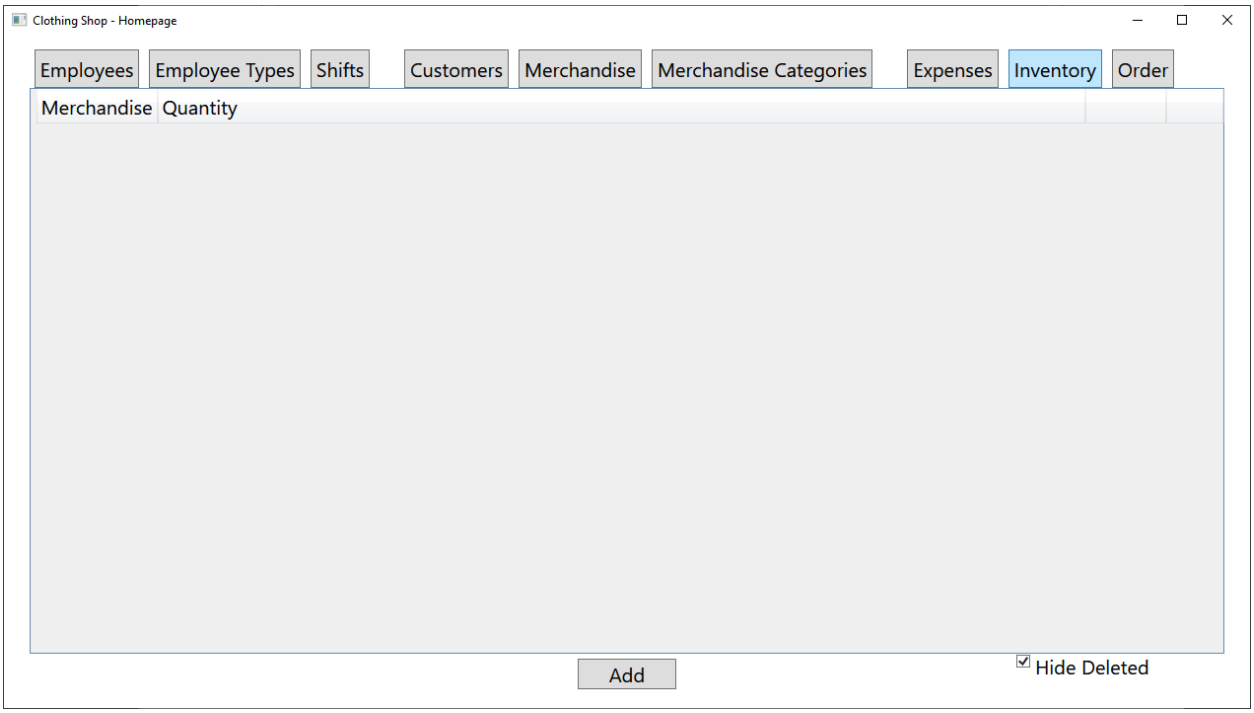


דף ההוצאות, בדך זה אפשר להוסיף, לערוך או להסיר הוצאות. דף זה יציג הוצאות אינן שוטפות.

הוספת הוצאה

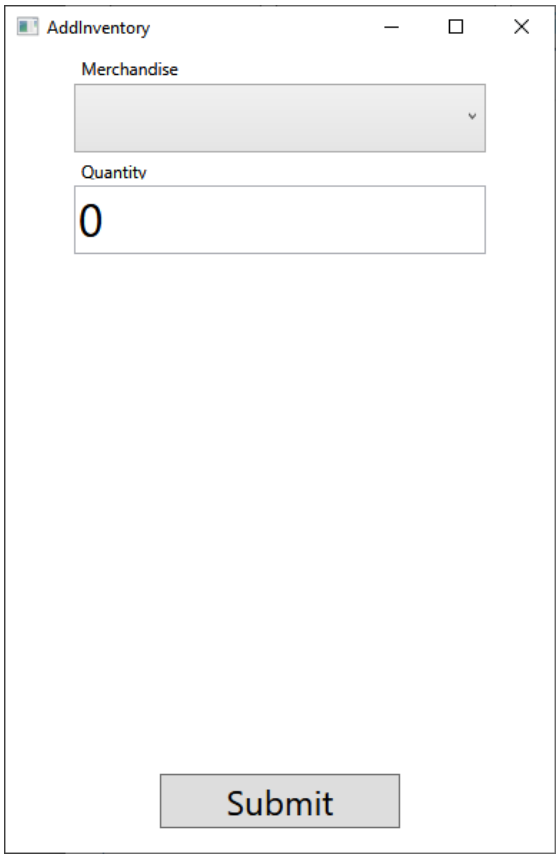


מלאי

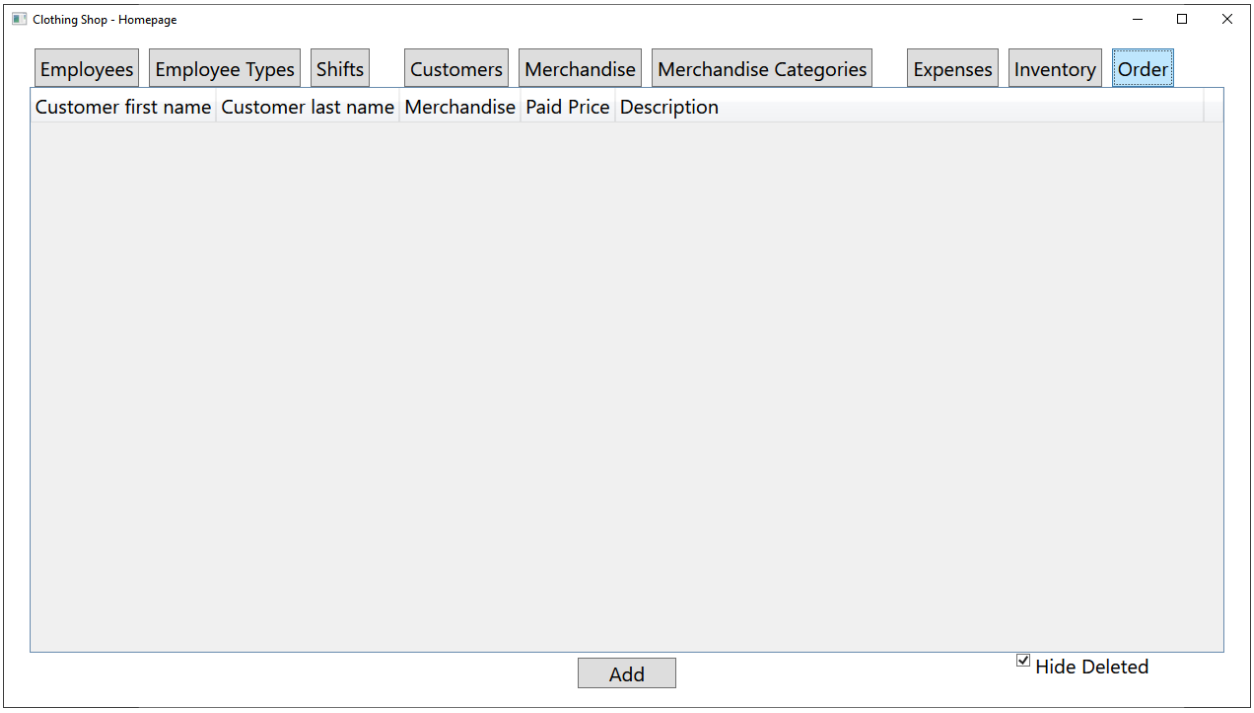


דף מלאי, בדך זה אפשר להוסיף, לערוך או להסיר מלאי קיים.

הוספה למלאי

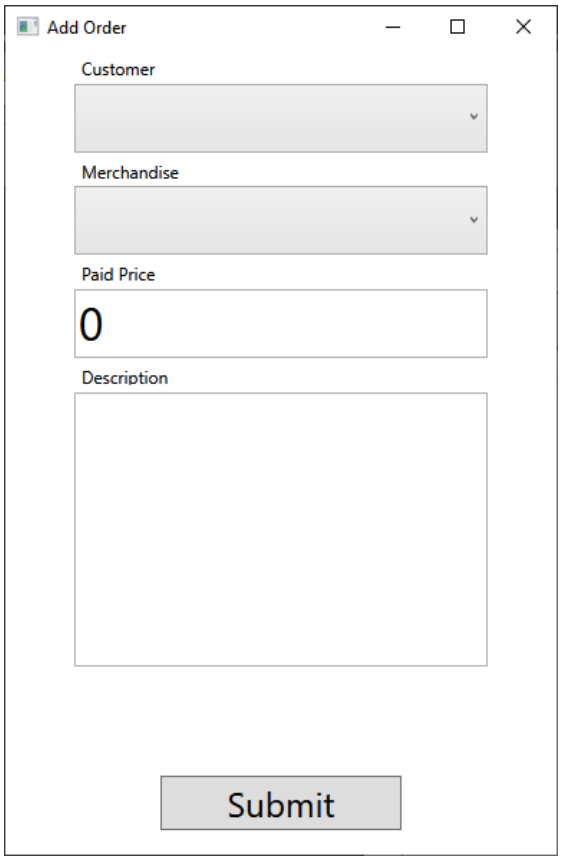


מכירות



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

הוספת מכירה

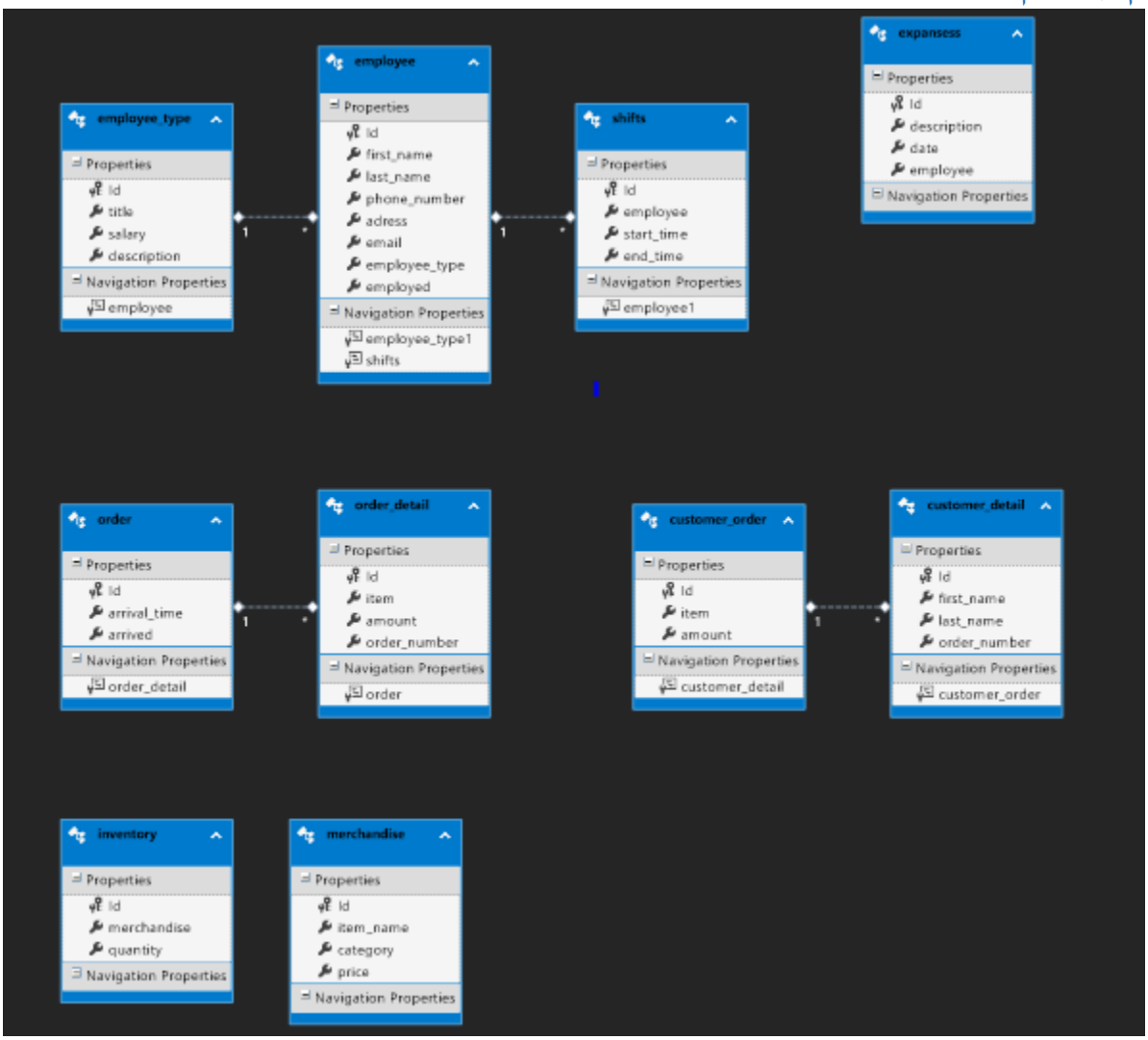


מדריך למתכנת

הסבר על שיטת השכבות

בתוכנה זו אני עושה שימוש ב**framework** בשם **caliburn.micro**. המערכת הזו משתמשת בשיטת השכבות **MVVM** (**Model, View, View-Model**), כאשר המודל מובא לי על ידי ה**database** 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 ב**viewmodel** מתוך ה **XAML** של ה**view** בצורת **binding** פשוטה או אפילו בשינוי השם של הרכיב לשם של המשתנה ב**viewmodel**.

קשרים בין הטבלאות



טבלת העובדים

employee

Properties	
Id	עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה
first_name	עמודה זו מייצגת את השם הפרטי של העובד
last_name	עמודה זו מייצגת את שם המשפחה של העובד
phone_number	עמודה זו מייצגת את מספר הטלפון של העובד
adress	עמודה זו מייצגת את הכתובת של העובד
email	עמודה זו מייצגת את כתובת האימייל של העובד
employee_type	עמודה זו מייצגת את הId של סוג העובד
employed	עמודה זו מציגה 0 אם העובד קיים או 1 אם העובד נמחק

טבלת סוגי העובדים

employee_type

Properties	
Id	עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה
title	עמודה זו מייצגת את סוג העובד
salary	עמודה זו מייצגת את המשכורת שסוג עובד זה מקבל
description	עמודה זו מייצגת את תיאור סוג עובד זה
Navigation Properties	
employee	

טבלת משמרות

shifts	
Properties	
Id	עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה
employee	עמודה זו מייצגת את הId של העובד
start_time	עמודה זו מייצגת את הזמן של התחלת המשמרת
end_time	עמודה זו מייצגת את הזמן של סיום המשמרת

טבלת לקוחות

customer_detail	
Properties	
Id	עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה
first_name	עמודה זו מייצגת את השם הפרטי של הלקוח
last_name	עמודה זו מייצגת את שם המשפחה של הלקוח
order_number	עמודה זו מייצגת את מספר ההזמנה של הלקוח

טבלת מוצרים

merchandise	
Properties	
Id	עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה
item_name	עמודה זו מייצגת את שם המוצר
category	עמודה זו מייצגת את הId של הקטגוריה של המוצר
price	עמודה זו מייצגת את המחיר של המוצר

טבלת הוצאות

expansess	
Properties	
Id	עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה
description	עמודה זו מייצגת את תיאור ההוצאה
date	עמודה זו מייצגת את תאריך ההוצאה
employee	עמודה זו מייצגת את הId של העובד האחראי על ההוצאה

טבלת מלאי

inventory	
Properties	
Id	עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה
merchandise	עמודה זו מייצגת את הId של המוצר
quantity	עמודה זו מייצגת הכמות שיש מהמוצר במלאי

טבלת המכירות

order	
Properties	
Id	עמודה ייחודית שגדלה באחד בכל שורה חדשה בטבלה
arrival_time	עמודה זו מייצגת את זמן ההגעה של ההזמנה
arrived	עמודה זו מייצגת 0 אם ההזמנה לא הגיעה ו1 אם ההזמנה הגיעה

קוד התוכנה

קוד משני

Bootstrapper

```
using FinalProject.ViewModels;
using Caliburn.Micro;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace FinalProject
{
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 FinalProject.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 FinalProject.ViewModels;
using Caliburn.Micro;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace FinalProject.ViewModels
{
class ShellViewModel : Conductor<object>
{
public Dictionary<string, Screen> ViewsDict { get; set; } = new Dictionary<st
ring, Screen>
{
{ "Employees", new EmployeesViewModel() },
{ "EmployeeTypes", new EmployeeTypesViewModel() },
{ "Shifts", new ShiftsViewModel() },
{ "MerchandiseCategories", new MerchandiseCategoriesViewModel() },
{ "Merchandise", new MerchandiseViewModel() },
{ " expansess ", new expansess () },
{ "Customers", new CustomerViewModel() },
{ "Inventory", new InventoryViewModel() },
{ "Orders", new OrderViewModel() },
{ "HomePage", new HomePageViewModel() }
};

public ShellViewModel()
{
ActivateItem(ViewsDict["HomePage"]);
}

public void ActivateView(string viewModelName)
{
DisplayName = $"Final Project- {viewModelName}";

ActivateItem(ViewsDict[viewModelName]);
}
}
}
```

CustomerViewModel

```
using Caliburn.Micro;
using FinalProject.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.ViewModels
{
class CustomerViewModel : Screen, INotifyPropertyChanged
{
public DataHandlers.CustomerDataHandler dataHandler = new DataHandlers.Custom
erDataHandler();

public List<customer> Customers { get { return dataHandler.GetData(HideDelete
d); } }
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.Sto
reWins, cust);
}
NotifyOfPropertyChange("Customers");

}
}
}
```

EmployeesViewModel

```
using FinalProject.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 FinalProject.ViewModels
{
class EmployeesViewModel : Screen, INotifyPropertyChanged
{
public DataHandlers.EmployeeDataHandler dataHandler = new DataHandlers.Employ
eeDataHandler();

public List<employee> Employees { get { return dataHandler.GetData(HideDelete
d); } }
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 FinalProject.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.ViewModels
{
class EmployeeTypesViewModel : Screen, INotifyPropertyChanged
{
public DataHandlers.EmployeeTypesDataHandler dataHandler = new DataHandlers.E
mployeeTypesDataHandler();

public List<employee_type> EmployeeTypes { get { return dataHandler.GetData(H
ideDeleted); } }
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.Sto
reWins, empType);
}
NotifyOfPropertyChange("EmployeeTypes");
}
}
}
```

expansess eViewModel

```
using Caliburn.Micro;
using FinalProject.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.ViewModels
{
class expansess: Screen, INotifyPropertyChanged
{
public DataHandlers. expansessDataHandler dataHandler = new DataHandlers.
expansess ();

public List< expansess > expansess
{ get { return dataHandler.GetData(HideDeleted); } }
public bool HideDeleted { get; set; } = true;

public void AddExpense()
{
expansess exp = new expansess ();
expansess addWindow = new expansess ();

exp.date = DateTime.Now;

// Sets the window's context and gives it the types available, that aren'
t deleted
addWindow.DataContext = new { expansess
= exp, employees = dataHandler.GetEntities().employee.ToList().FindAll(type => { retu
rn !type.deleted; }) };
addWindow.ShowDialog();
```

```
try
{
dataHandler.AddOrUpdate(exp);
}
catch (Exception e)
{
System.Windows.MessageBox.Show("Error: " + e.Message);
}
NotifyOfPropertyChange("expansess ");

}

public void Delete(expansess obj)
{
dataHandler.RemoveData(obj);
NotifyOfPropertyChange("expansess ");
}

public void Modify(expense exp)
{
expansess addWindow = new expansess ();

// Sets the window's context and gives it the types available, that aren'
t deleted
addWindow.DataContext = new { expansess
= exp, employees = dataHandler.GetEntities().employee.ToList().FindAll(type => { retu
rn !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.Sto
reWins, exp);
}
NotifyOfPropertyChange("expansess ");

}
}
}
```

HomePageViewModel

```
using Caliburn.Micro;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.ViewModels
{
 class HomePageViewModel : Screen
{
}
}
```

InventoryViewModel

```
using Caliburn.Micro;
using FinalProject.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.ViewModels
{
class InventoryViewModel : Screen, INotifyPropertyChanged
{
public DataHandlers.InventoryDataHandler dataHandler = new DataHandlers.Inven
toryDataHandler();

public List<inventory> Inventory { get { return dataHandler.GetData(HideDelet
ed); } }
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.Sto
reWins, inv);
}
NotifyOfPropertyChange("Inventory");

}
}
}
```

MerchandiseCategoriesViewModel

```
using Caliburn.Micro;
using FinalProject.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.ViewModels
{
class MerchandiseCategoriesViewModel : Screen, INotifyPropertyChanged
{
```

```
        public DataHandlers.MerchandiseCategoryDataHandler dataHandler = new DataHand
                                                lers.MerchandiseCategoryDataHandler();

        public List<merchandise_category> MerchandiseCategories { get { return dataHa
                                                        ndler.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.Sto
                                                                    reWins, merchCategory);
                                                                                        }
                                        NotifyOfPropertyChange("MerchandiseCategories");

                                                                                        }
                                                                                        }
                                                                                        }
```

MerchandiseViewModel

```
                                                                    using Caliburn.Micro;
                                                                 using FinalProject.Views;
                                                                             using System;
                                                         using System.Collections.Generic;
                                                              using System.ComponentModel;
                                                                        using System.Linq;
                                                                        using System.Text;
                                                             using System.Threading.Tasks;

                                                         namespace FinalProject.ViewModels
                                                                                        {
                          class MerchandiseViewModel : Screen, INotifyPropertyChanged
                                                                                        {
        public DataHandlers.MerchandiseDataHandler dataHandler = new DataHandlers.Mer
                                                                    chandiseDataHandler();

        public List<merchandise> Merchandise { get { return dataHandler.GetData(HideD
                                                                                eleted); } }
                                        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.GetEnt
  ities().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 FinalProject.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.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 = dataHan
dler.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 FinalProject.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.ViewModels
{
class ShiftsViewModel : Screen, INotifyPropertyChanged
{
public DataHandlers.ShiftDataHandler dataHandler = new DataHandlers.ShiftData
Handler();

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.GetEntit
ies().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.GetEntit
                ies().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.Sto
                                                                                        reWins, 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 FinalProject
                                                                                                {
                                                                    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).GetPropert
                y("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 FinalProject.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 FinalProject.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 FinalProject.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");
}
}
}
}
```

expansessDataHandler

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.DataHandlers
{
class expansesseDataHandler : DataHandler<expansess>
{
protected override void VerifyItem(expense item)
{
}
}
}
```

InventoryDataHandler

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FinalProject.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 FinalProject.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 FinalProject.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 FinalProject.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;

.DataHandlers namespace FinalProject
{
class ShiftDataHandler : DataHandler<shift>
{
protected override void VerifyItem(shift item)
{
}
}
}
```

סיכום

במהלך בניית הפרויקט למדתי רבות על הנושא. למדתי כיצד להתמודד עם בעיות ובאגים שיש עם הפרויקט, כיצד להשתמש במערכת caluburn.micro, בשיטת השכבות mvvm והעשרתי את הידע שלי על wpf ו#C. לסיכום בניית פרויקט זה היה חוויה מדהימה ומאוד מעשירה וחינוכית שבטח ולבטח תעזור לי להמשך החיים.