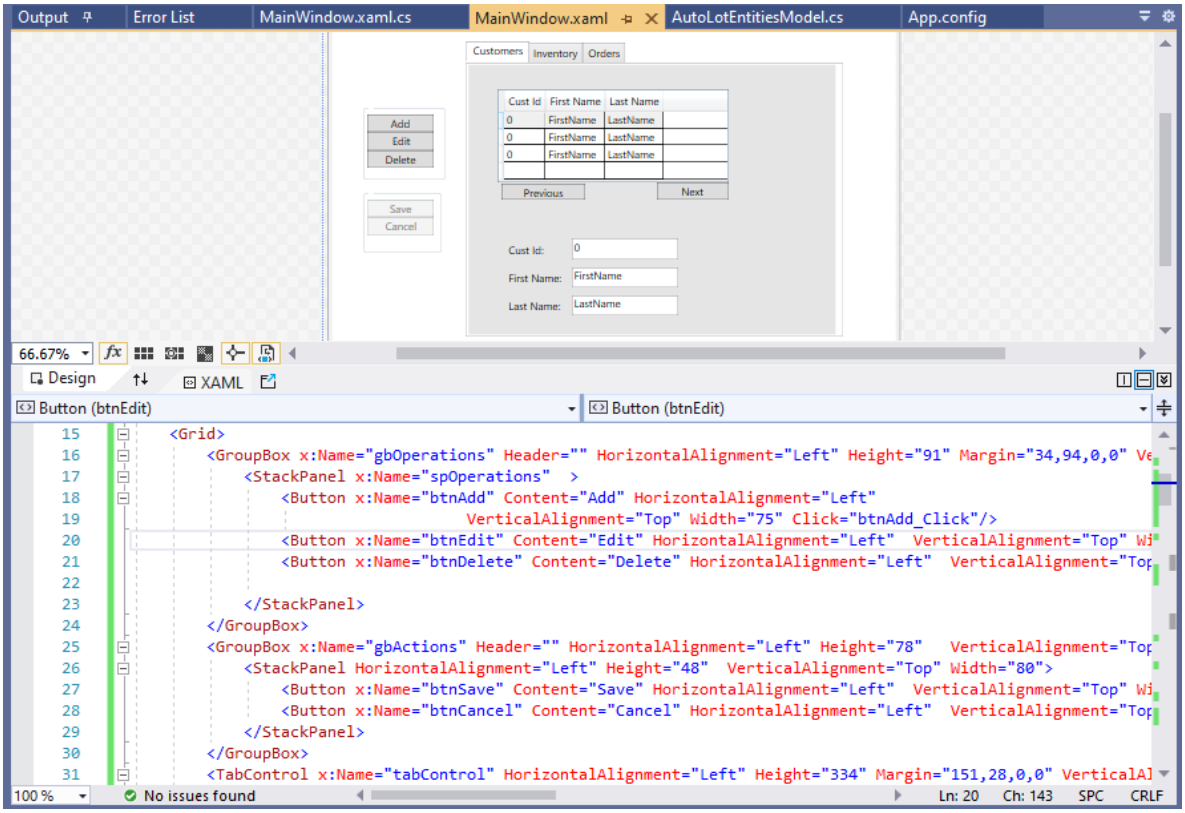
Laborator 6 – Aplicatie WPF folosind Entity Framework - Code First from DataBase (continuare Lab 5)

1. Pentru a putea realiza operatiile CRUD unitar, vom scoate butoanele aferente operatiilor in afara elementului TabControl conform imaginii de mai jos. Vom grupa butoanele in doua GroupBox-uri cu numele gbOperations pentru butoanele btnNew, btnEdit, btnDelete repectiv un GroupBox cu numele gbActions pentru butoanele btnSave si btnCancel. In interiorul fiecarui GroupBox vom adauga un StackPanel conform imaginii de mai jos.



2. În fişierul de code-behind al formularului (**MainWindow.xaml.cs**) adăugăm următoarea structură enum:

```
namespace Nume_Pren_Lab5
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    ///
    enum ActionState
    {
        New,
        Edit,
        Delete,
        Nothing
    }
    public partial class MainWindow : Window
    {
```

```
ActionState action = ActionState.Nothing;
```

3. Observam ca in MainWindow.xaml.cs s-a creat handlerul de eveniment Window_Loaded pentru obiectul Window cu urmatorul continut:

```
0 references
public MainWindow()
{
    InitializeComponent();
}
2
0 references
private void Window_Loaded(object sender, RoutedEventArgs e)
{

    System.Windows.Data.CollectionViewSource customerViewSource =
        ((System.Windows.Data.CollectionViewSource)(this.FindResource("customerViewSource")));
    // Load data by setting the CollectionViewSource.Source property:
    // customerViewSource.Source = [generic data source]
}
}
```

4. Vom seta contextul in constructorul clasei de code-behind si vom modifica codul din handlerul de eveniment Window_Loaded astfel:

```
public partial class MainWindow : Window
    {
        //using AutoLotModel;
        ActionState action = ActionState.Nothing;
        AutoLotEntitiesModel ctx = new AutoLotEntitiesModel();
        CollectionViewSource customerVSource;
        public MainWindow()
        {
            InitializeComponent();
            DataContext = this;
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            //using System.Data.Entity;
            customerVSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("customerViewSource")));
            customerVSource.Source = ctx.Customers.Local;
            ctx.Customers.Load();
        }
    }
}
```

5. Vom crea handlerele de evenimente pentru butoanele btnAdd, btnEdit si btnDelete cu urmatorul continut:

```
private void btnAdd_Click(object sender, RoutedEventArgs e)
        {
            action = ActionState.New;
        }

private void btnEditO_Click(object sender, RoutedEventArgs e)
        {
            action = ActionState.Edit;
        }
```

```
        private void btnDeleteO_Click(object sender, RoutedEventArgs e)
        {
            action = ActionState.Delete;
        }
```

6. Handerele de eveniment de la butoanele Next si Previous vor avea urmatorul continut:

```
  private void btnNext_Click(object sender, RoutedEventArgs e)
        {
            customerVSource.View.MoveCurrentToNext();
        }

   private void btnPrevious_Click(object sender, RoutedEventArgs e)
       {
           customerVSource.View.MoveCurrentToPrevious();
       }
```

7. Vom crea o metoda SaveCustomers cu urmatorul continut:

```
private void SaveCustomers()
        {
            Customer customer = null;
            if (action == ActionState.New)
            {
                try
                {
                    //instantiem Customer entity
                    customer = new Customer()
                    {
                        FirstName = firstNameTextBox.Text.Trim(),
                        LastName = lastNameTextBox.Text.Trim()
                    };
                    //adaugam entitatea nou creata in context
                    ctx.Customers.Add(customer);
                    customerVSource.View.Refresh();
                    //salvam modificarile
                    ctx.SaveChanges();
                }
                //using System.Data;
                catch (DataException ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }

            else
if (action == ActionState.Edit)
            {
                try
                {
                    customer = (Customer)customerDataGrid.SelectedItem;
                    customer.FirstName = firstNameTextBox.Text.Trim();
                    customer.LastName = lastNameTextBox.Text.Trim();
```

```
                    //salvam modificarile
                    ctx.SaveChanges();
                }
                catch (DataException ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }

            else if (action == ActionState.Delete)
            {
                try
                {
                    customer = (Customer)customerDataGrid.SelectedItem;
                    ctx.Customers.Remove(customer);
                    ctx.SaveChanges();
                }
                catch (DataException ex)
                {
                    MessageBox.Show(ex.Message);
                }
                customerVSource.View.Refresh();
            }

        }
```

8. Reluati pasii 6 si 7 pentru TabItem-ul Inventory, creand handler-ele de evenimente pentru butoanele btnNext si btnPrevious precum si metoda SaveInventory() similar cu exemplul de la Customers

9. Pentru a activa/dezactiva corespunzator butoanele aferente operatiilor CRUD vom utiliza un eveniment rutat specificand pentru GroupBox-ul gbOperations faptul ca la declasansarea evenimentului Click de catre un buton din interiorul sau se va executa handler-ul gbOperations_Click:

```
<GroupBox x:Name="gbOperations"  Button.Click="gbOperations_Click" …>

        …
```

10. Handler-ul gbOperations_Click va dezactiva toate butoanele in afara de cel care a declansat evenimentul Click si va avea urmatorul continut:

```
private void gbOperations_Click(object sender, RoutedEventArgs e)
        {

            Button SelectedButton = (Button)e.OriginalSource;
            Panel panel = (Panel)SelectedButton.Parent;

                foreach (Button B in panel.Children.OfType<Button>())
                {
                    if (B != SelectedButton)
                        B.IsEnabled = false;
                }

            gbActions.IsEnabled = true;
        }
```

11. Pentru a reinitializa butoanele aferente operatiilor vom crea metoda Reinitialize astfel:

```
private void ReInitialize()
        {

            Panel panel = gbOperations.Content as Panel;

            foreach (Button B in panel.Children.OfType<Button>())
            {

                B.IsEnabled = true;
            }

            gbActions.IsEnabled = false;
        }
```

12. Vom apela metoda Reinitialize in handler-ul de eveniment btnCancel_Click astfel :

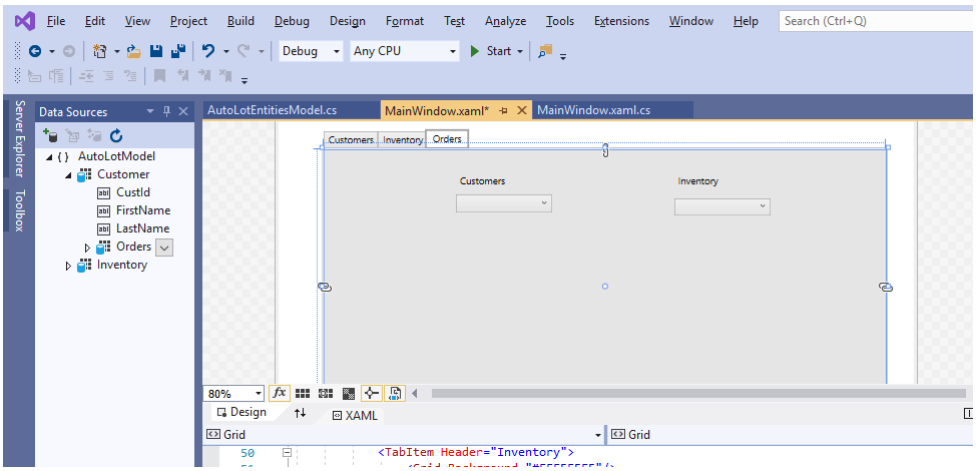
```
private void btnCancel_Click(object sender, RoutedEventArgs e)
        {
            ReInitialize();
        }
```

13. Vom apela crea handler-ul de eveniment btnSave_Click care va apela metoda corespunzatoare TabItem-ului selectat pentru salvarea operatiilor iar apoi va reinitializea butoanele aferente operatiilor :

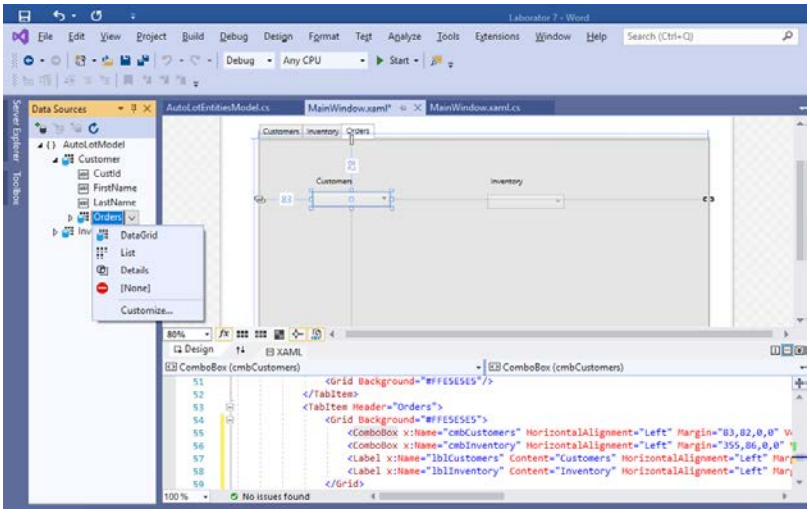
```
      private void btnSave_Click(object sender, RoutedEventArgs e)
        {

            TabItem ti = tbCtrlAutoLot.SelectedItem as TabItem;

            switch (ti.Header)
                {
                case "Customers":
                    SaveCustomers();
                    break;
                case "Inventory":
                    SaveInventory();
                    break;
                case "Orders":
                    break;

            }
            ReInitialize();
        }
```

Rulam aplicatia si verificam faptul ca toate operatiile functioneaza corespunzator pentru Customers si Inventory.

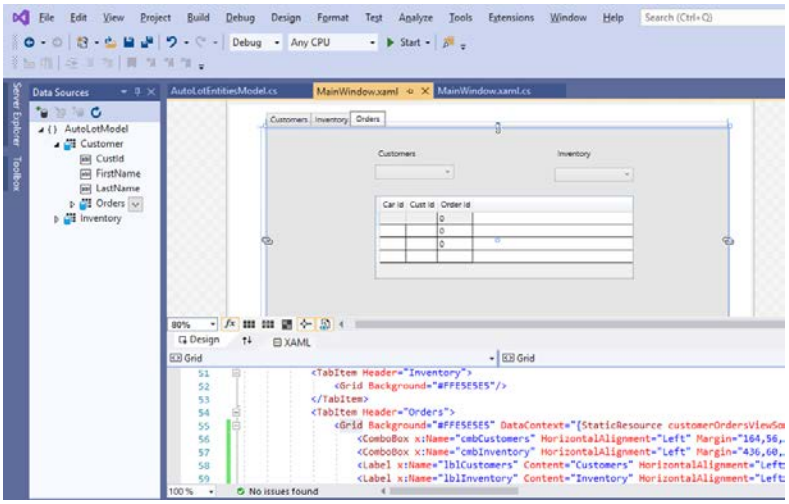
14. Pentru TabItem-ul Orders vom adauga pe interfata doua combobox-uri aferente Customers si Inventory pentru care vom seta proprietatea name cmbCustomers respectiv cmbInventory



15. In fereastra DataSources facem click pe sageata din fata de la Customers si apoi click pe sageata din dreapta de la Orders - alegem **DataGrid**



16. Facem apoi drag&drop la Orders pe MainWindow.xaml



17. Vom modifica codul din handlerul de eveniment Window_Loaded astfel:

```
CollectionViewSource customerOrdersVSource;
        public MainWindow()
        {
            InitializeComponent();
            DataContext = this;
        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            customerVSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("customerViewSource")));
            customerVSource.Source = ctx.Customers.Local;
            ctx.Customers.Load();


customerOrdersVSource=
((System.Windows.Data.CollectionViewSource)(this.FindResource("customerOrdersViewSource")));

            customerOrdersVSource.Source = ctx.Orders.Local;
            ctx.Orders.Load();

            ctx.Inventories.Load();

            cmbCustomers.ItemsSource = ctx.Customers.Local;
            cmbCustomers.DisplayMemberPath = "FirstName";
            cmbCustomers.SelectedValuePath = "CustId";

            cmbInventory.ItemsSource = ctx.Inventories.Local;
            cmbInventory.DisplayMemberPath = "Make";
            cmbInventory.SelectedValuePath = "CarId";

        }
    }
}
```

18. Vom crea metoda SaveOrders() cu urmatorul continut:

```
private void SaveOrders()
        {
            Order order = null;
            if (action == ActionState.New)
            {
                try
                {
                    Customer customer = (Customer)cmbCustomers.SelectedItem;
                    Inventory inventory = (Inventory)cmbInventory.SelectedItem;

                    //instantiem Order entity
                    order = new Order()
                    {

                    CustId = customer.CustId,
                    CarId = inventory.CarId
                    };
                    //adaugam entitatea nou creata in context
```

```
                    ctx.Orders.Add(order);
                    //salvam modificarile
                    ctx.SaveChanges();
                    BindDataGrid();
                }
                catch (DataException ex)
                {
                    MessageBox.Show(ex.Message);
                }
}
else

 if (action == ActionState.Edit)
 {
 dynamic selectedOrder = ordersDataGrid.SelectedItem;
 try
 {
 int curr_id = selectedOrder.OrderId;

 var editedOrder = ctx.Orders.FirstOrDefault(s => s.OrderId == curr_id);
 if (editedOrder != null)
 {

 editedOrder.CustId = Int32.Parse(cmbCustomers.SelectedValue.ToString());

 editedOrder.CarId = Convert.ToInt32(cmbInventory.SelectedValue.ToString());
 //salvam modificarile
 ctx.SaveChanges();
 }
 }
 catch (DataException ex)
 {
 MessageBox.Show(ex.Message);
 }

 BindDataGrid();
 // pozitionarea pe item-ul curent
 customerOrdersVSource.View.MoveCurrentTo(selectedOrder);
 }

 else if (action == ActionState.Delete)
 {

 try
 {
 dynamic selectedOrder = ordersDataGrid.SelectedItem;

 int curr_id = selectedOrder.OrderId;
 var deletedOrder = ctx.Orders.FirstOrDefault(s => s.OrderId == curr_id);
 if (deletedOrder != null)
 {

 ctx.Orders.Remove(deletedOrder);
 ctx.SaveChanges();
 MessageBox.Show("Order Deleted Successfully", "Message");
 BindDataGrid();
 }
```

```
    }
    catch (DataException ex)
    {
    MessageBox.Show(ex.Message);
    }

    }

  }
```

19. Pentru Tab-ul Orders vom imbunatatii modul de afisare din combobox-uri. Pentru Combobox-ul Customers dorim sa se afiseze atat FirstName cat si LastName (momentan se afiseaza doar FirstName).In handler-ul de eveniment Window_Loaded, comentam linia de mai jos:

```
    private void Window_Loaded(object sender, RoutedEventArgs e)
    {

    …


    cmbCustomers.ItemsSource = ctx.Customers.Local;
    //cmbCustomers.DisplayMemberPath = "FirstName";
    cmbCustomers.SelectedValuePath = "CustId";


    …

    }
```

20. In fisierul Window.xaml vom adauga un DataTemplate pentru combox-ul cmbCustomers astfel:

```
    <ComboBox x:Name="cmbCustomers" HorizontalAlignment="Left" Margin="164,56,0,0"
    VerticalAlignment="Top" Width="120" >
                          <ComboBox.ItemTemplate>
                           <DataTemplate>
                               <TextBlock>
                                   <TextBlock.Text>
                                           <MultiBinding StringFormat="{}{0} {1}">
                                           <Binding Path="FirstName"/>
                                           <Binding Path="LastName"/>
                                       </MultiBinding>
                                   </TextBlock.Text>
                               </TextBlock>
                           </DataTemplate>
                           </ComboBox.ItemTemplate>
                      </ComboBox>
```

21. Pentru combox-ul Inventory dorim sa se afiseze atat marca masinii cat si culoarea acesteia. In fisierul MainWindow.xaml.cs, in handler-ul de eveniment Window_Loaded comentam linia de mai jos:

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{

…

cmbInventory.ItemsSource = ctx.Inventories.Local;
//cmbInventory.DisplayMemberPath = "Make";
cmbInventory.SelectedValuePath = "CarId";
…

}
```

22. In fisierul Window.xaml vom adauga un DataTemplate pentru combox-ul cmbInventory astfel:

```
<ComboBox x:Name="cmbInventory" HorizontalAlignment="Left" Margin="436,60,0,0"
VerticalAlignment="Top" Width="120">
                        <ComboBox.ItemTemplate>
                            <DataTemplate>
                                <TextBlock>
                                    <TextBlock.Text>
                                        <MultiBinding StringFormat="{}{0} - {1}">
                                            <Binding Path="Make"/>
                                            <Binding Path="Color"/>
                                        </MultiBinding>
                                    </TextBlock.Text>
                                </TextBlock>
                            </DataTemplate>
                        </ComboBox.ItemTemplate>
                    </ComboBox>
```

23. Dorim ca in Datagrid-ul **ordersDataGrid** sa fie afisat in loc de id-ul clientului si id-ul masinii comandate, sa se afiseze nume si preume client respectiv marca si culoare masina comandata. Creem in fisierul MainWindow.xaml.cs o metoda BindDataGrid() cu urmatorul continut:

```
private void BindDataGrid()
        {
            var queryOrder = from ord in ctx.Orders
                             join cust in ctx.Customers on ord.CustId equals
                             cust.CustId join inv in ctx.Inventories on ord.CarId
                             equals inv.CarId select new { ord.OrderId, ord. CarId,
                             ord.CustId, cust.FirstName,  cust.LastName, inv.Make,
                             inv.Color };
          customerOrdersVSource.Source = queryOrder.ToList();
        }
```

24. In handler-ul de eveniment Window_Loaded, comentam linia de mai jos si apelam metoda `BindDataGrid()`

```
    private void Window_Loaded(object sender, RoutedEventArgs e)
    {

    …

    // customerOrdersVSource.Source = ctx.Orders.Local;
    …

    BindDataGrid();
    }
```

25. In MainWindow.xaml, modificam coloanele din DataGrid pentru Orders astfel incat sa afiseze datele obtinute la integogarea de la punctul 23

```
   <DataGrid x:Name="ordersDataGrid" AutoGenerateColumns="False"
   EnableRowVirtualization="True" ItemsSource="{Binding}"
   RowDetailsVisibilityMode="VisibleWhenSelected" IsSynchronizedWithCurrentItem="True">
           <DataGrid.Columns>
             <DataGridTextColumn x:Name="FirstNameColumn" Binding="{Binding FirstName}"
Header="First Name" Width="SizeToHeader"/>
             <DataGridTextColumn x:Name="LastNameColumn" Binding="{Binding LastName}"
Header="Last Name" Width="SizeToHeader"/>
             <DataGridTextColumn x:Name="MakeColumn" Binding="{Binding Make}"
Header="Make" Width="SizeToHeader"/>
             <DataGridTextColumn x:Name="ColorColumn" Binding="{Binding Color}"
Header="Color" Width="SizeToHeader"/>
           </DataGrid.Columns>
    </DataGrid>
```

26. Pentru ca in TabItem Orders pentru cele doua ComboBox-uri sa se modifice elementul curent atunci cand selectam un rand din ordersDataGrid vom face bind cu acesta astfel:

```
   <ComboBox x:Name="cmbCustomers" HorizontalAlignment="Left" VerticalAlignment="Top"
   Width="120" Margin="165,68,0,0" SelectedValue="{Binding
   ElementName=ordersDataGrid,Path=SelectedItem.CustId,Mode=OneWay}">
                            <ComboBox.ItemTemplate>
                                <DataTemplate>
                                    <TextBlock>
                                        <TextBlock.Text>
                                            <MultiBinding StringFormat="{}{0} {1}">
                                                <Binding Path="FirstName"/>
                                                <Binding Path="LastName"/>
                                            </MultiBinding>
                                        </TextBlock.Text>
                                    </TextBlock>
                                </DataTemplate>
                            </ComboBox.ItemTemplate>
                        </ComboBox>

   <ComboBox x:Name="cmbInventory" HorizontalAlignment="Left" Margin="350,70,0,0"
   VerticalAlignment="Top" Width="120" SelectedValue="{Binding
   ElementName=ordersDataGrid,Path=SelectedItem.CarId,Mode=OneWay}">
                            <ComboBox.ItemTemplate>
                                <DataTemplate>
                                    <TextBlock>
                                        <TextBlock.Text>
                                            <MultiBinding StringFormat="{}{0} - {1}">
                                                <Binding Path="Make"/>
                                                <Binding Path="Color"/>
```

```
                                        </MultiBinding>
                                    </TextBlock.Text>
                                </TextBlock>
                            </DataTemplate>
                        </ComboBox.ItemTemplate>
                    </ComboBox>
```