Entity_Framework

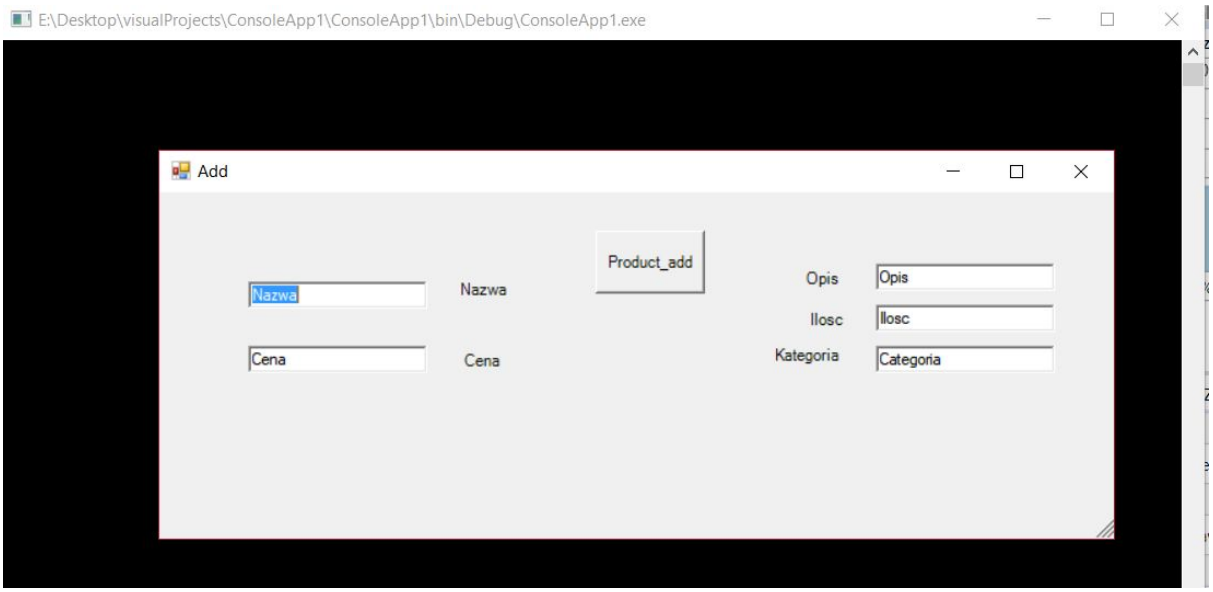
Celem zadania domowego było rozszerzenie funkcjonalności aplikacji według własnego pomysłu.
Zdecydowałem się na rozszerzanie funkcjonalności początkowych formularzy i na dodawanie nowych formularzy, które mają dodać nowe możliwości wykorzystania aplikacji.

Pierwszym usprawnieniem jest dodanie nowego formularza, odpowiedzialnego za dodawanie nowych produktów.

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Data.Entity;
6.  using System.Drawing;
7.  using System.Linq;
8.  using System.Text;
9.  using System.Threading.Tasks;
10. using System.Windows.Forms;
11.
12. namespace ConsoleApp1
13. {
14.     public partial class Form2 : Form
15.     {
16.         int ProductId;
17.         int CategoryId;
18.
19.         public Form2()
20.         {
21.             InitializeComponent();
22.         }
23.
24.         private void return_to_start()
25.         {
26.
27.             categoryId_box.Text = "CategoryId";
28.             productName_box.Text = "ProductName";
29.             description_box.Text = "Decription";
30.             unitprice_box.Text = "Unitprice";
31.             unitsinstock_box.Text = "UnitsInStock";
32.         }
33.
34.
35.         private void addButton_Click(object sender, EventArgs e)
```

```
36.       {
37.
38.           string category_ = categoryId_box.ToString();
39.           string name = productName_box.ToString();
40.           string description = description_box.ToString();
41.           decimal unitprice = Decimal.Parse(unitprice_box.Text);
42.           int unitsinstock = Int32.Parse(unitsinstock_box.Text);
43.
44.           using (var prodContext = new ProdContext())
45.           {
46.               var category_id =
47.                   from c in prodContext.Categories
48.                   where c.Name == category_
49.                   select c.CategoryId;
50.
51.               if (category_id == null)
52.               {
53.                   Console.WriteLine("Nowa kategoria");
54.                   this.CategoryId = prodContext.Categories.Select(c => c.CategoryId).Max();
55.                   this.ProductId = prodContext.Products.Select(p => p.ProductId).Max();
56.                   prodContext.Categories.Add(new Category { CategoryId = CategoryId + 1,
    Name = category_, Products = { new Product {ProductId = this.ProductId + 1, CategoryID
    = CategoryId + 1, Description = description, Name = name, Unitprice = unitprice,
    UnitsInStock =unitsinstock } } });
57.                   prodContext.Products.Add(new Product { ProductId = this.ProductId + 1,
    CategoryID = category_id.First(), Description =description, Name = name, Unitprice =
    unitprice, UnitsInStock = unitsinstock });
58.                   return_to_start();
59.                   return;
60.               }
61.
62.               this.ProductId = prodContext.Products.Select(p => p.ProductId).Max();
63.
64.               prodContext.Products.Add(new Product { ProductId = ProductId +1 , CategoryID
    = category_id.First(), Description =description, Name = name, Unitprice = unitprice,
    UnitsInStock = unitsinstock });
65.               return_to_start();
66.
67.           }
68.
69.
70.       }
71.   }
72. }
```

Założyłem tutaj, że jeżeli w bazie danych nie ma kategorii, którą klient podaje, to ja tworzę.



Kolejnym ważnym usprawnieniem dodanym przeze mnie do bazy danych jest większa ilość filtrów i statystyk, które są dostępne w formularzu Produktów.

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Data.Entity;
6.  using System.Data.Entity.Core.Objects;
7.  using System.Drawing;
8.  using System.Linq;
9.  using System.Text;
10. using System.Threading.Tasks;
11. using System.Windows.Forms;
12.
13.
14. namespace ConsoleApp1
15. {
16.     public partial class CategoryForm : Form
17.     {
18.
19.         public ProdContext prodContext;
20.         public DataGridView dataGridView1;
21.         private BindingList<Category> categoryBindingSource;
22.         private BindingList<Product> productBindingSource;
23.         private BindingList<Customer> customerBindingSource;
24.
25.         public CategoryForm()
26.         {
```

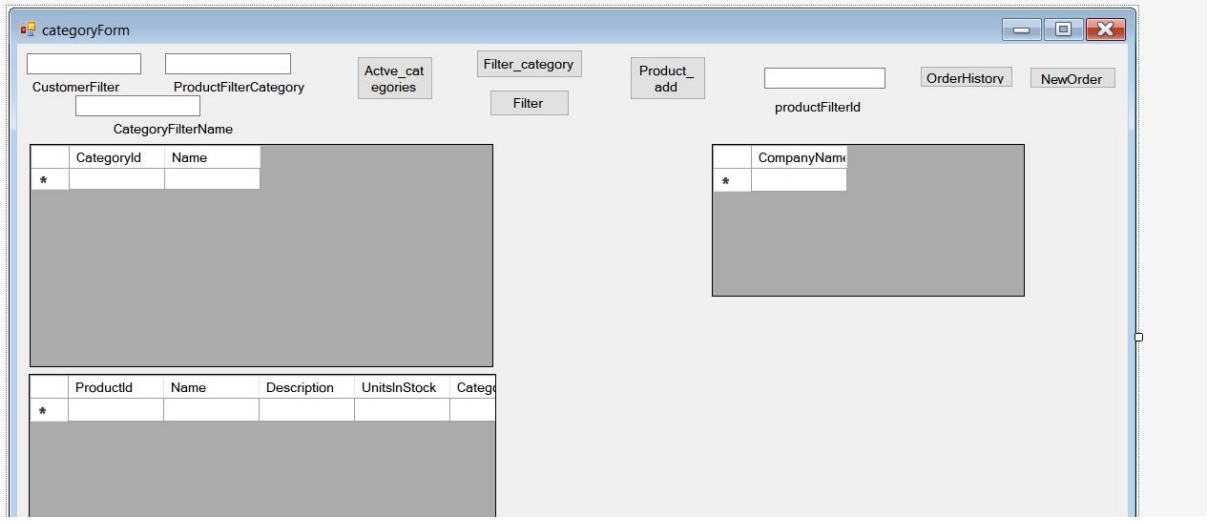
```
27.             InitializeComponent();
28.
29.         }
30.
31.         public void CategoryForm_Load()
32.         {
33.             BindingSource categoryBindingSource = new BindingSource();
34.             ProdContext prodContext = new ProdContext();
35.             prodContext.Categories.Load();
36.             prodContext.Customers.Load();
37.             this.categoryBindingSource1.DataSource =
    prodContext.Categories.Local.ToBindingList();
38.             this.productBindingSource1.DataSource =
    prodContext.Products.Local.ToBindingList();
39.             this.customerBindingSource1.DataSource =
    prodContext.Customers.Local.ToBindingList();
40.             dataGridView1.AutoGenerateColumns = true;
41.             dataGridView1.DataSource = categoryBindingSource;
42.
43.         }
44.
45.         private void categoryBindingNavigatorSaveItem_Click(object sender, EventArgs e)
46.         {
47.             if (prodContext != null)
48.             {
49.
50.                 prodContext.SaveChanges();
51.
52.             }
53.         }
54.
55.         private void setProductsDataSource(DataGridViewCellEventArgs e)
56.         {
57.             if (e.RowIndex < 0 || e.ColumnIndex < 0) return;
58.             string filter = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
59.             this.dataGridView1.DataSource =
60.             prodContext.Products
61.                 .Where(product => product.CategoryID.ToString() == filter)
62.                 .Select(product => new
63.                 {
64.                     ProductId = product.ProductId,
65.                     Name = product.Name,
66.                     Price = product.Unitprice,
67.                 }).ToList();
68.         }
69.
70.         private void setProductsDatSource2(DataGridViewCellEventArgs e)
71.         {
72.             string filter = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
73.             IEnumerable<Product> query =
74.                 from p in prodContext.Products
75.                 where p.CategoryID.ToString() == filter
76.                 select p;
77.             this.dataGridView1.DataSource = query.ToList();
```

```
78.       }
79.
80.       private void blogDataGridView_CellContentClick(object sender,
    DataGridViewCellEventArgs e)
81.       {
82.          setProductsDataSource(e);
83.          this.dataGridView1.Update();
84.          this.dataGridView1.Refresh();
85.       }
86.
87.       private void filterButton_Click(object sender, EventArgs e)
88.       {
89.          prodContext.Categories.Load();
90.          this.dataGridView1.DataSource =
91.          prodContext.Products
92.             .Where(prod =>
93.             (productFilterID.Text != "" &&
    prod.CategoryID.ToString().Contains(productFilterID.Text) == true)
94.             &&
95.             (productFilterName.Text != "" &&
    prod.Name.ToString().Contains(productFilterName.Text) == true))
96.             .Select(prod => new
97.             {
98.                ProductId = prod.ProductId,
99.                Name = prod.Name,
100.                    Description = prod.Description,
101.                    Unitprice = prod.Unitprice,
102.                    UnitsInStock = prod.UnitsInStock
103.                }).ToList();
104.             this.categoryDataGridView1.Update();
105.             this.productDataGridView1.Refresh();
106.             this.productDataGridView1.Update();
107.             this.categoryDataGridView1.Refresh();
108.
109.          }
110.          private void filterButton_Click1(object sender, EventArgs e)
111.          {
112.
113.             this.dataGridView1.DataSource =
114.          prodContext.Categories
115.             .Where(category =>
116.             (categoryFilterName.Text != "" &&
    category.Name.ToString().Contains(categoryFilterName.Text) == true))
117.             .Select(c => new
118.             {
119.                CategoryId = c.CategoryId,
120.                Name = c.Name
121.             }).ToList();
122.                this.categoryDataGridView1.Update();
123.                this.productDataGridView1.Refresh();
124.                this.productDataGridView1.Update();
125.                this.categoryDataGridView1.Refresh();
126.             }
127.
```

```
128.
129.
130.
131.         private void categoriesDataGridView_CellContentClick(object sender,
    DataGridViewCellEventArgs e)
132.         {
133.             setcategoriesDataSourceQuerySyntax(e);
134.             this.productDataGridView1.Update();
135.             this.productDataGridView1.Refresh();
136.         }
137.
138.         private void customerDataGridView_CellContentClick(object sender,
    DataGridViewCellEventArgs e)
139.         {
140.             setcategoriesDataSourceQuerySyntax(e);
141.             this.categoryDataGridView1.Update();
142.             this.categoryDataGridView1.Refresh();
143.         }
144.
145.         private void setcategoriesDataSourceQuerySyntax(DataGridViewCellEventArgs e)
146.         {
147.             string filter =
    customerDataGridView.Rows[e.RowIndex].Cells[0].Value.ToString();
148.             IEnumerable<Category> query =
149.                 from c in prodContext.Categories
150.                 where c.Name.ToString() == filter
151.                 select c;
152.             this.categoryDataGridView.DataSource = query.ToList();
153.         }
154.
155.         private void categoryWithCount(object sender, EventArgs e)
156.         {
157.             IEnumerable<Category> categories = prodContext.Categories;
158.             IEnumerable<Product> products = prodContext.Products;
159.
160.             this.dataGridView1.DataSource =
161.                 categories.GroupJoin
162.                 (products, product => product.CategoryId, category =>
    category.CategoryID,
163.                 (category, categoryGroup) =>
164.                 new
165.                 {
166.                     CategoryName = category.Name,
167.                     CategoryCount = categoryGroup.Count()
168.
169.
170.                 }).ToList();
171.
172.         }
173.
174.
175.         private void categoryWithCount2(object sender, EventArgs e)
176.         {
177.             IEnumerable<Category> categories = prodContext.Categories;
```

```
178.             IEnumerable<Product> products = prodContext.Products;
179.
180.             this.dataGridView1.DataSource =
181.                 from category in categories
182.                 join product in products
183.                 on category.CategoryId equals
184.                 product.CategoryID into categoryGroup
185.                 select new
186.                 {
187.                     CategoryName = category.Name,
188.                     CategoryCount = categoryGroup.Count()
189.                 };
190.
191.         }
192.
193.         private void filterButton_Click2(object sender, EventArgs e)
194.         {
195.
196.             this.dataGridView1.DataSource =
197.         prodContext.Customers
198.             .Where(customer =>
199.             (customerFilterName.Text != "" &&
    customer.CompanyName.Contains(customerFilterName.Text) == true))
200.             .Select(c => new
201.             {
202.                 CompanyName = c.CompanyName
203.             }).ToList();
204.             this.categoryDataGridView1.Update();
205.             this.productDataGridView1.Refresh();
206.             this.productDataGridView1.Update();
207.             this.categoryDataGridView1.Refresh();
208.         }
209.
210.         private void AddButton_Click(object sender, EventArgs e)
211.         {
212.             Form2 addform = new Form2();
213.             addform.ShowDialog();
214.
215.         }
216.
217.         private void Show_Active_Categories(object sender, EventArgs e)
218.         {
219.             this.dataGridView1.DataSource =
220.                 prodContext.Categories.Select(c => c.Products).Where(p => p.Count() >
    0).ToList();
221.             this.categoryDataGridView1.Update();
222.             this.productDataGridView1.Refresh();
223.             this.productDataGridView1.Update();
224.             this.categoryDataGridView1.Refresh();
225.
226.
227.         }
228.
229.         private void Show_Active_Categories2(object sender, EventArgs e)
```

```
230.         {
231.             IEnumerable<Category> categories = prodContext.Categories;
232.             this.dataGridView1.DataSource =
233.                 from c in categories
234.                 where c.Products.Count() > 0
235.                 select c;
236.
237.
238.
239.
240.         }
241.
242.         private void button1_Click(object sender, EventArgs e)
243.         {
244.             OrderHistory orderHistory = new OrderHistory();
245.             orderHistory.ShowDialog();
246.         }
247.
248.         private void button2_Click(object sender, EventArgs e)
249.         {
250.             Zamowienie zamowienie = new Zamowienie();
251.             zamowienie.ShowDialog();
252.         }
253.     }
254. }
```



Funkcje zaimplementowane zostały na 2 sposoby (query i method syntax).

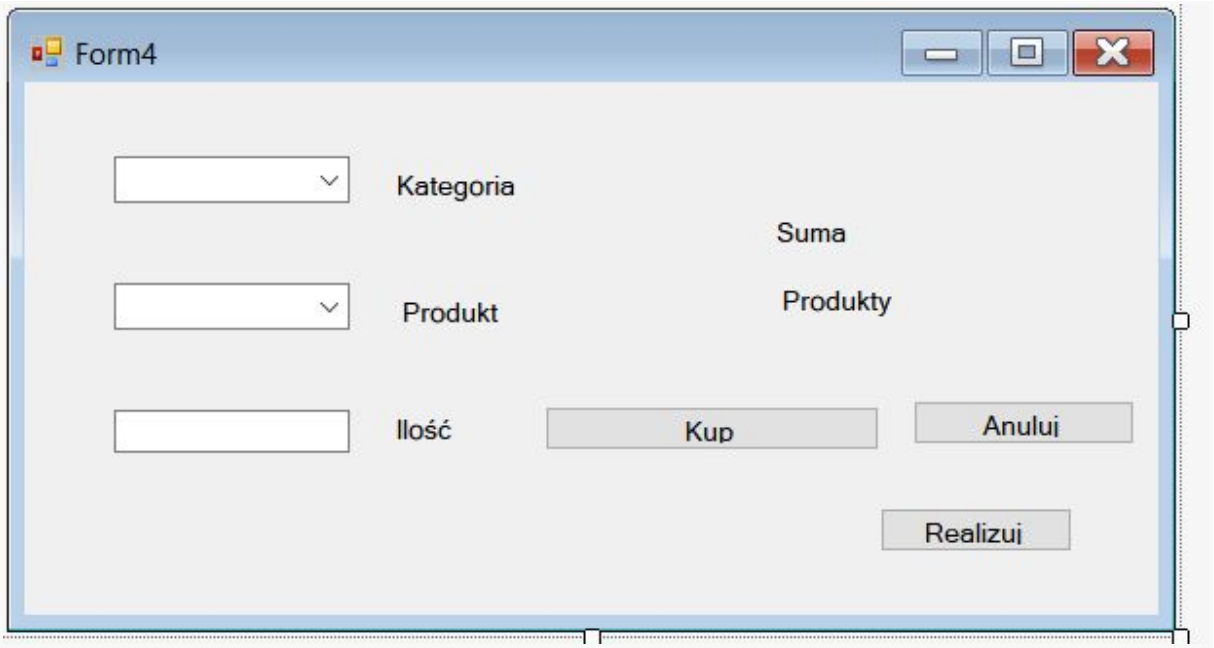
Następnym usprawnieniem, na które się zdecydowałem jest składanie zamówień
na produkty(ten proces jest podzielony na 2 części:

w pierwszej wybieramy interesujące nas produkty i dodajemy do koszyka, natomiast w drugiej podajemy nasze dane osobowe i dokonujemy potwierdzenia zamówienia). Aby zrealizować to w przejrzysty sposób utworzyłem 2 formularze.

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Data.Entity;
6.  using System.Drawing;
7.  using System.Linq;
8.  using System.Text;
9.  using System.Threading.Tasks;
10. using System.Windows.Forms;
11.
12. namespace ConsoleApp1
13. {
14.     public partial class Zamowienie : Form
15.     {
16.
17.         ProdContext context;
18.         int Category_Id;
19.         string productName;
20.         int available;
21.         int ammount;
22.         decimal price;
23.         int ProductId;
24.         decimal sum;
25.         List<Product> productList = new List<Product>();
26.
27.         public Zamowienie()
28.         {
29.
30.             InitializeComponent();
31.             context = new ProdContext();
32.             comboBox1.DataSource = context.Categories.Select(category =>
    category.Name).ToList();
33.             comboBox1.DisplayMember = "Category Name";
34.             sum = 0;
35.
36.         }
37.
38.         private void Zamowienie_Load(object sender, EventArgs e)
39.         {
40.             context.Categories.Load();
41.             context.Products.Load();
42.
43.         }
44.
45.
46.         private void textBox1_TextChanged(object sender, EventArgs e)
47.         {
48.             if (int.TryParse(textBox1.Text, out ammount))
```

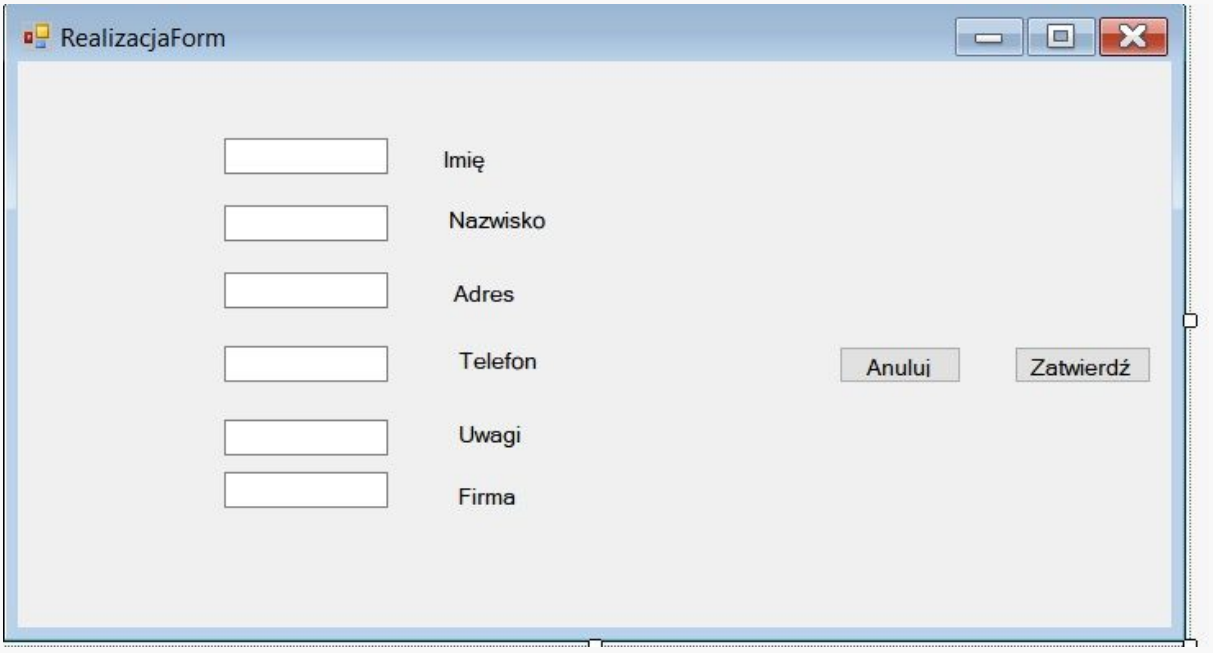
```
49.         {
50.             if ( ammount > available || ammount <=0)
51.             {
52.                 return;
53.             }
54.         }
55.         else
56.         {
57.             textBox1.Text = String.Empty;
58.             return;
59.         }
60.     }
61.
62.     private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
63.     {
64.         string selected = comboBox1.GetItemText(comboBox1.SelectedItem);
65.
66.         Category_Id = context.Categories.Where(category => category.Name ==
    selected).
67.             Select(category => category.CategoryId).FirstOrDefault();
68.
69.         comboBox2.DataSource = context.Products.Where(p => p.CategoryID ==
    Category_Id).
70.             Select(p => p.Name).ToList();
71.         comboBox2.DisplayMember = "Name";
72.     }
73.
74.     private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
75.     {
76.         productName = comboBox2.GetItemText(comboBox2.SelectedItem);
77.
78.         available =context.Products.Where(p => p.Name == productName).
79.             Select(p => p.UnitsInStock).FirstOrDefault();
80.     }
81.
82.     private void button1_Click(object sender, EventArgs e)
83.     {
84.
85.         price = context.Products.Where(p => p.Name == productName).Select(p
    => p.Unitprice).First();
86.
87.         ProductId = context.Products.Where(p => p.Name ==
    productName).Select(p => p.ProductId).First();
88.
89.         decimal val = ammount * price;
90.         label2.Text += ammount.ToString() + "x " + productName.ToString() + "  "
    + val.ToString() + "zł\n";
91.
92.
93.         sum += val;
94.         label1.Text = sum.ToString() + " zł";
95.
96.         Product prod = new Product();
97.         prod.ProductId = ProductId;
```

```
98.         prod.Name = productName;
99.         prod.UnitsInStock = ammount;
100.            prod.Unitprice = price;
101.            productList.Add(prod);
102.         }
103.
104.         private void button3_Click(object sender, EventArgs e)
105.         {
106.            int OrderId = context.Orders.Select(o => o.OrderID).Max();
107.            RealizacjaForm realizacja = new RealizacjaForm();
108.            realizacja.context = this.context;
109.            realizacja.price = this.price;
110.            realizacja.Products = productList;
111.            realizacja.OrderId = OrderId;
112.            realizacja.ShowDialog();
113.         }
114.
115.         private void button2_Click(object sender, EventArgs e)
116.         {
117.            this.Close();
118.         }
119.
120.      }
121.   }
```




```
1. using System;
2. using System.Collections.Generic;
3. using System.ComponentModel;
4. using System.Data;
5. using System.Drawing;
6. using System.Linq;
7. using System.Text;
8. using System.Threading.Tasks;
```

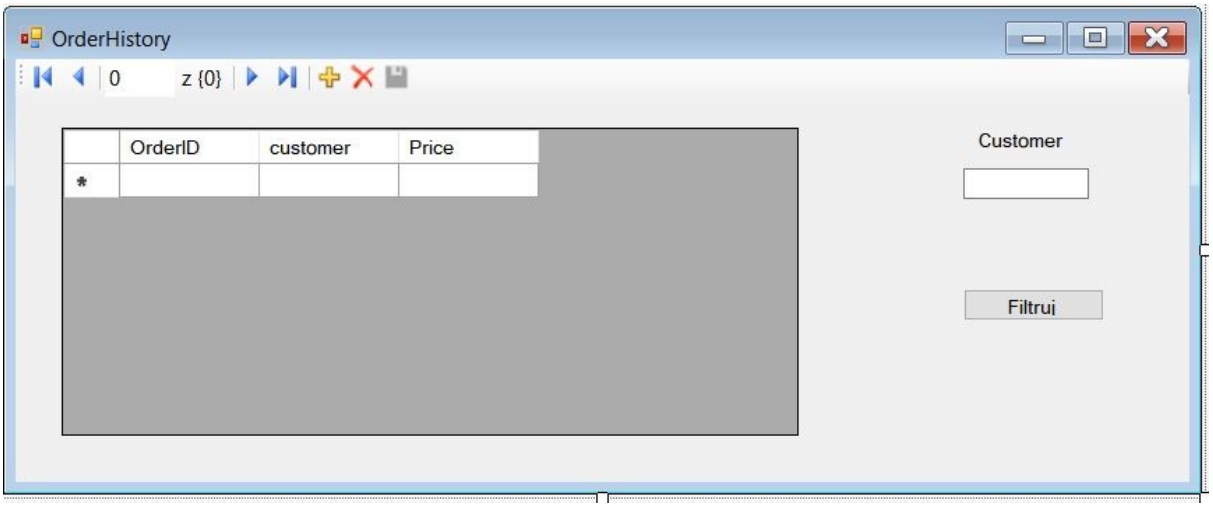
```
9.  using System.Windows.Forms;
10.
11. namespace ConsoleApp1
12. {
13.     public partial class RealizacjaForm : Form
14.     {
15.
16.         public int OrderId;
17.         public List<Product> Products;
18.         public ProdContext context;
19.         public Order orderHistory;
20.         public decimal price;
21.
22.         public RealizacjaForm()
23.         {
24.             InitializeComponent();
25.         }
26.
27.         private void button1_Click(object sender, EventArgs e)
28.         {
29.             if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" &&
    textBox4.Text != ""
30.                 && textBox5.Text != "" && textBox6.Text != "" )
31.             {
32.                 MessageBox.Show("Dziękujemy za zamówienie");
33.                 string companyName = textBox6.Text;
34.                 string s = context.Customers.Select(c => c.CompanyName).ToString();
35.                 if(s == null)
36.                 {
37.                     context.Customers.Add(new Customer { CompanyName =
    companyName });
38.                 }
39.
40.                 context.Orders.Add(new Order
41.                 {
42.                     OrderID = OrderId,
43.                     customer = new Customer { CompanyName = companyName },
44.                     Price = price,
45.                     Products =
46.                     this.Products
47.                 });
48.
49.             }
50.             return;
51.         }
52.
53.         private void button2_Click(object sender, EventArgs e)
54.         {
55.             MessageBox.Show("Zamówienie anulowane");
56.             return;
57.         }
58.     }
59. }
```



Kolejnym usprawnieniem było też stworzenie nowej klasy - Zamówienie i formularza Historia Zamówień, gdzie będziemy zapisywać każde zrealizowane zamówienie(klient, wartość zamówienia, lista zamówionych produktów)

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Data.Entity;
6.  using System.Drawing;
7.  using System.Linq;
8.  using System.Text;
9.  using System.Threading.Tasks;
10. using System.Windows.Forms;
11.
12. namespace ConsoleApp1
13. {
14.     public partial class OrderHistory : Form
15.     {
16.         public ProdContext context;
17.
18.
19.         public OrderHistory()
20.         {
21.             InitializeComponent();
22.         }
23.
24.         public void OrderHistory_Load()
25.         {
26.             this.context = new ProdContext();
```

```
27.             this.context.Orders.Load();
28.             this.orderBindingSource.DataSource = context.Orders.Local.ToBindingList();
29.         }
30.
31.
32.         private void button1_Click(object sender, EventArgs e)
33.         {
34.             this.orderDataGridView.DataSource =
35.                 context.Orders.Select(o => o.customer).Where(c => c.CompanyName ==
    textBox1.Text.ToString()).Select(company =>company).ToList();
36.
37.             this.orderDataGridView.Update();
38.             this.orderDataGridView.Refresh();
39.         }
40.     }
41. }
```



Na koniec zamieszczam też kilka funkcji, które mogą się przydać w dalszym rozszerzaniu aplikacji.

```
1.  private void _show_(ProdContext context)
2.     {
3.
4.         var query = context.Categories.Select(item => item.Name).ToList();
5.
6.         foreach (var item in query)
7.         {
8.             Console.WriteLine(item);
9.         }
10.    }
11.
12.    private void show_categories_with_products(ProdContext context)
13.    {
14.        var result = context.Categories
```

```
15.          .Join(context.Products,
16.             c => c.CategoryId,
17.             p => p.CategoryID,
18.             (c, p) => new { c, p })
19.       .Select(n => new
20.       {
21.          CategoryId = n.c.CategoryId,
22.          CategoryName = n.c.Name,
23.          ProductId = n.p.ProductId,
24.          ProductName = n.p.Name,
25.          Description = n.p.Description,
26.          Unitprice = n.p.Unitprice,
27.          UnitsInStock = n.p.UnitsInStock
28.       });
29.
30.       foreach (var item in result)
31.       {
32.          Console.WriteLine(item);
33.       }
34.    }
35.
36.    private static void show_categories_with_products2(ProdContext context)
37.    {
38.       var result =
39.                from c in context.Categories
40.                join p in context.Products
41.                on c.CategoryId equals p.CategoryID
42.                select new
43.                {
44.                   CategoryId = c.CategoryId,
45.                   CategoryName = c.Name,
46.                   ProductId = p.ProductId,
47.                   ProductName = p.Name,
48.                   UnitPrice = p.Unitprice,
49.                   UnitsInStock = p.UnitsInStock
50.                };
51.
52.       foreach (var item in result)
53.       {
54.          Console.WriteLine(item);
55.       }
56. }
57.
58.
59.    private static void show_categories_with_ammount(ProdContext context)
60.    {
61.
62.       var result = context.Categories
63.     .GroupJoin(context.Products,
64.        c => c.CategoryId,
65.        p => p.CategoryID,
66.        (c, categorygroup) =>
67.     new
68.     {
```

```
69.         Category = c.CategoryId,
70.         Ammount = categorygroup.Count()
71.     });
72.
73.     foreach (var item in result)
74.     {
75.         Console.WriteLine(item);
76.     }
77. }
78.
79.
80. private void show_categories_with_ammount2(ProdContext context)
81. {
82.     var query = from c in context.Categories
83.                 join p in context.Products
84.                 on c.CategoryId equals p.CategoryID
85.                 into categorygroup
86.                 select new
87.                 {
88.                     Category = c.CategoryId,
89.                     Ammount = categorygroup.Count()
90.                 };
91.
92.     foreach (var item in query)
93.     {
94.         Console.WriteLine(item);
95.     }
96. }
```