Sprawozdanie 2 **data laboratorium: 30.10.19 r.**
Bazy danych
Entity Framework, LINQ2Entities

Zadanie polegało na dokończeniu rozpoczętej aplikacji, rozszerzając ją m.in. o możliwość składania zamówień. Postanowiłem kontynuować projekt w formie aplikacji WindowsFormowej.

Dokończenie zadania z laboratorium:

a) **Funkcja wyświetlająca kategorie razem z produktami**

```
private static void showProductsWithCategories(ProdContext context)
        {
            var query = from product in context.Products
                        join category in context.Categories
                        on product.CategoryID equals category.CategoryID
                        select new
                        {
                            CategoryId = category.CategoryID,
                            CategoryName = category.Name,
                            ProductId = product.ProductId,
                            ProductName = product.Name,
                            Unitprice = product.UnitPrice,
                            UnitsInStock = product.UnitsInStock
                        };

            foreach(var item in query)
            {
                Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}\t{5}",
                    item.CategoryId,
                    item.CategoryName,
                    item.ProductId,
                    item.ProductName,
                    item.Unitprice,
                    item.UnitsInStock
                    );
            }
        }

        private static void showProductsWithCategories2(ProdContext
context)
        {
```

```
            var query = context.Categories.Join(context.Products, category
=> category.CategoryID, product => product.CategoryID,
                (category, product) => new
                {
                    CategoryId = category.CategoryID,
                    CategoryName = category.Name,
                    ProductId = product.ProductId,
                    ProductName = product.Name,
                    Unitprice = product.UnitPrice,
                    UnitsInStock = product.UnitsInStock
                });

            foreach (var item in query)
            {
                Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}\t{5}",
                    item.CategoryId,
                    item.CategoryName,
                    item.ProductId,
                    item.ProductName,
                    item.Unitprice,
                    item.UnitsInStock
                    );
            }
        }
```

Funkcję napisałem dwa razy - za pomocą query syntax i method syntax.

b) Funkcja wyświetlająca kategorie razem z liczbą produktów danej kategorii.

```
private static void showCategoriesWithAmout(ProdContext context)
       {
           var query = from category in context.Categories
                       join product in context.Products
                       on category.CategoryID equals product.CategoryID
                       into categorygroup
                       select new
                       {
                           Category = category.CategoryID,
                           Ammount = categorygroup.Count()
                       };
```

```
            foreach (var item in query)
            {
                Console.WriteLine(item);

            }
        }

        private static void showCategoriesWithAmout2(ProdContext context)
        {
            var query = context.Categories.GroupJoin(context.Products,
                category => category.CategoryID, product => product.CategoryID,
                (category, categoryGroup) => new
                {
                    Category = category.CategoryID,
                    Ammout = categoryGroup.Count()
                });

            foreach(var item in query)
            {
                Console.WriteLine(item);
            }
        }
```

Podobnie jak wcześniej funkcje napisałem na dwa sposoby.

Głównym usprawnieniem była rozbudowa CategoryForm. Pozwala na wyświetlenie produktów wybranej kategorii, filtrowania produktów i kategorii po nazwie, wyszukiwanie produktów w określonej cenie czy też pokazanie dostępnych produktów. Przyciski z boku okienka otwierają inne formularze.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
```

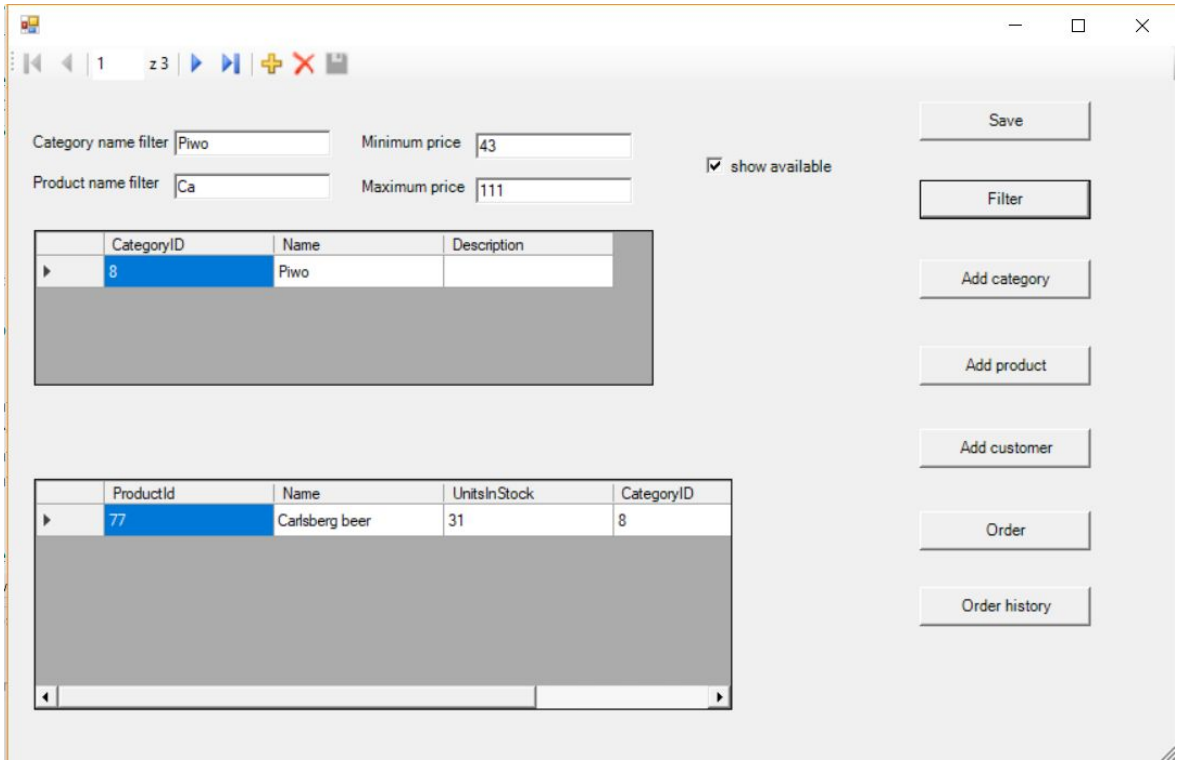
```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyDataBase
{
    public partial class CategoryForm : Form
    {

        public ProdContext prodContext;
        public DataGridView dataGridView1;

        public CategoryForm()
        {
            InitializeComponent();
        }


        private void saveButton_Click(object sender, EventArgs e)
        {
            prodContext.SaveChanges();
            prodContext.Categories.Load();
            prodContext.Products.Load();

            this.categoryDataGridView.Refresh();
            this.productsDataGridView.Refresh();
        }

        private void CategoryForm_Load(object sender, EventArgs e)
        {
            BindingSource categoryBindingSource = new BindingSource();
            prodContext = new ProdContext();
            prodContext.Categories.Load();
            prodContext.Products.Load();
            this.categoryBindingSource.DataSource =

prodContext.Categories.Local.ToBindingList();
            this.productsBindingSource.DataSource =
prodContext.Products.Local.ToBindingList();

        }

        private void categoryDataGridView_CellContentClick(object sender,
```

```
DataGridViewCellEventArgs e)
        {
            int row = e.RowIndex;
            Category category =
(Category)this.categoryDataGridView.Rows[row].DataBoundItem;
            if (category != null)
            {
                this.productsBindingSource.DataSource = new
BindingList<Product>(
                    prodContext.Products.Where(p => p.CategoryID ==
category.CategoryID).ToList());

            }

        }


        private void categoryDataGridView_CellContentClick2(object sender,
DataGridViewCellEventArgs e)
        {
            int row = e.RowIndex;
            Category category =
(Category)this.categoryDataGridView.Rows[row].DataBoundItem;
            if (category != null)
            {
                this.productsBindingSource.DataSource =  (from p in
prodContext.Products
               where p.CategoryID == category.CategoryID
               select p).ToList();

            }

        }

        private void categoryFilterButton_Click(object sender, EventArgs e)
        {
            ProdContext prodContext = new ProdContext();
            Boolean parse1;
            Boolean parse2;
            Decimal minPrice = 0;
            Decimal maxPrice = Decimal.MaxValue;
            if (this.minPriceTextBox.Text == "" || this.maxPriceTextBox.Text
```

```
== "")
            {
                if (minPriceTextBox.Text == "")
                {
                    minPrice = 0;
                }
                if (maxPriceTextBox.Text == "")
                {
                    maxPrice = Decimal.MaxValue;
                }
            }
            else if (!Decimal.TryParse(this.minPriceTextBox.Text, out
minPrice) ||
                !Decimal.TryParse(this.maxPriceTextBox.Text, out maxPrice) ||
minPrice > maxPrice)
            {
                MessageBox.Show("Bad price filters");
            };



            this.categoryDataGridView.DataSource =
prodContext.Categories.Where(category => (filterButton.Text != "" &&

category.Name.ToString().Contains(categoryNameFilterTextBox.Text) ==
true)).ToList();

            if (checkBox1.Checked == true)
            {
                this.productsDataGridView.DataSource =
prodContext.Products.Where(product =>
((product.Name.ToString().Contains(this.productNameFiltertextBox.Text)) ==
true) &&
                    (product.UnitPrice > minPrice) && product.UnitPrice <
maxPrice && product.UnitsInStock > 0).ToList();
            }
            else
            {
                this.productsDataGridView.DataSource =
prodContext.Products.Where(product =>
((product.Name.ToString().Contains(this.productNameFiltertextBox.Text)) ==
true) &&
```

```
                        (product.UnitPrice > minPrice) && product.UnitPrice <
maxPrice).ToList();
            }

            this.categoryDataGridView.Update();
            this.productsDataGridView.Refresh();
            this.productsDataGridView.Update();
            this.categoryDataGridView.Refresh();

        }

        private void categoryNameFilterTextBox_TextChanged(object sender,
EventArgs e)
        {
            this.categoryDataGridView.DataSource =
prodContext.Categories.Where(category => (
      category.Name.ToString().Contains(this.categoryNameFilterTextBox.Text)
== true)).ToList();
        }

        private void productNameFiltertextBox_TextChanged(object sender,
EventArgs e)
        {
            this.productsDataGridView.DataSource =
prodContext.Products.Where(prod => (
            prod.Name.ToString().Contains(this.productNameFiltertextBox.Text)
== true)).ToList();
        }

        private void orderButton_Click(object sender, EventArgs e)
        {
            AddOrderForm addOrderForm = new AddOrderForm();
            addOrderForm.ShowDialog();

        }

        private void addProductButton_Click(object sender, EventArgs e)
        {
            AddProductForm addProductForm = new AddProductForm();
            addProductForm.ShowDialog();
        }
```

```
        private void addCategoryButton_Click(object sender, EventArgs e)
        {
            AddCategoryForm addCategoryForm = new AddCategoryForm();
            addCategoryForm.ShowDialog();
        }

        private void addCustomer_Click(object sender, EventArgs e)
        {
            AddCustomer addCustomer = new AddCustomer();
            addCustomer.ShowDialog();
        }

        private void orderHistoryButton_Click(object sender, EventArgs e)
        {
            OrderHistoryForm orderHistoryForm = new OrderHistoryForm();
            orderHistoryForm.ShowDialog();
        }
    }
}
```

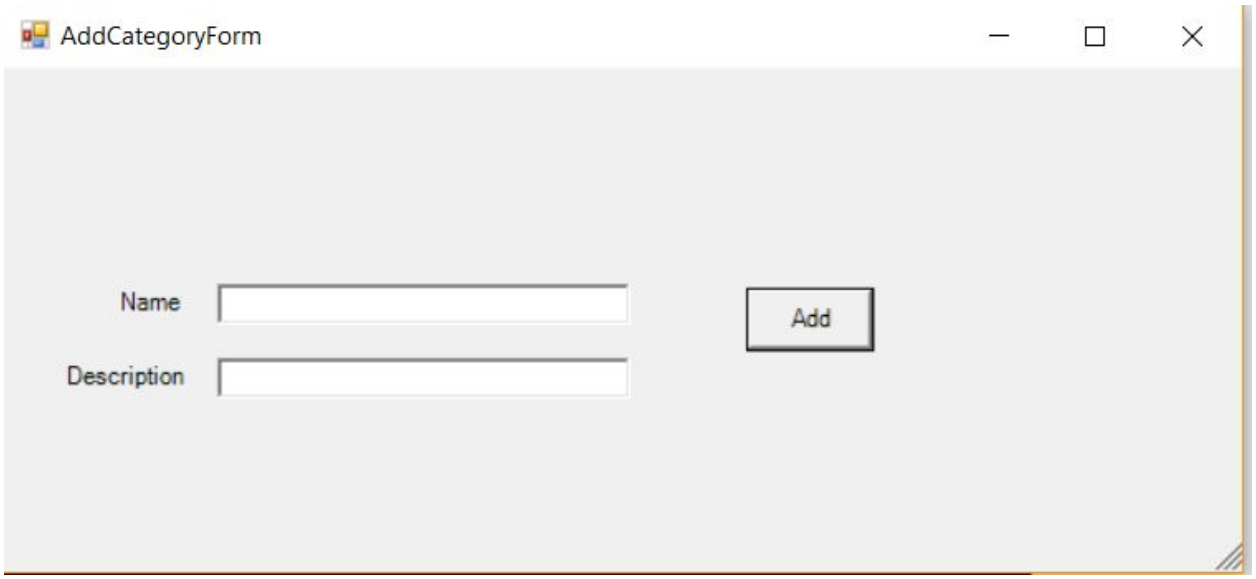


Jednym z nowych formularzy jest formularz dodania nowej kategorii.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyDataBase
{
    public partial class AddCategoryForm : Form
    {
        public AddCategoryForm()
        {
            InitializeComponent();
        }
```

```
        private void resetBoxes()
        {
            categoryNameTextBox.Text = "";
            descriptionTextBox.Text = "";
        }



        private void AddCategoryForm_Load(object sender, EventArgs e)
        {

        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            string categoryName1 = categoryNameTextBox.Text;
            string description1 = descriptionTextBox.Text;

            using(var db = new ProdContext())
            {
                var catID = from c in db.Categories
                            where c.Name == categoryName1
                            select c.CategoryID;

                if (catID == null)
                {
                    Category category = new Category
                    {
                        Name = categoryName1,
                        Description = description1
                    };
                    db.Categories.Add(category);
                    MessageBox.Show("Category added");
                    db.SaveChanges();
                    this.resetBoxes();
                }
                else
                {
                    MessageBox.Show("Category exists");

                }
```

```
            }
        }

        private void categoryNameTextBox_Click(object sender, EventArgs e)
        {

        }
    }
}
```

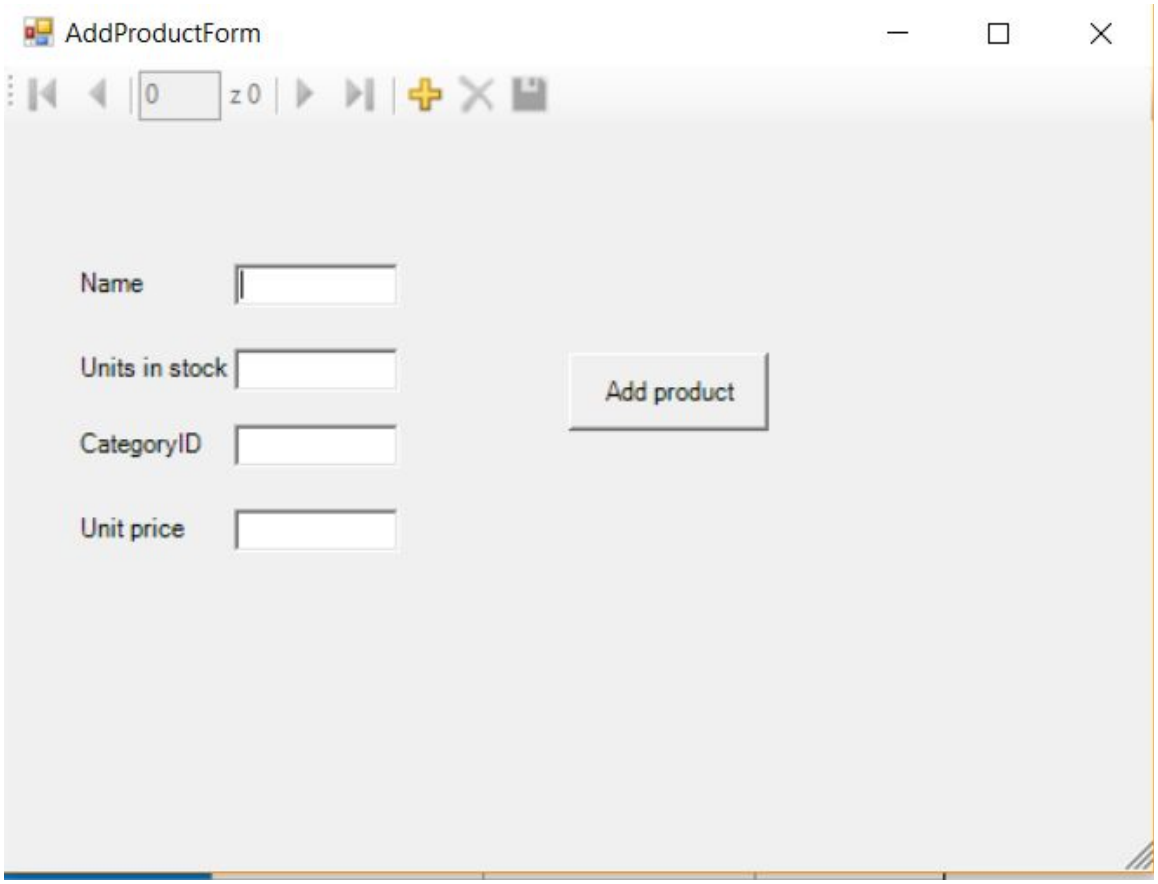


Kolejny formularz umożliwia dodanie nowego produktu. Sprawdzam poprawność wpisywanych danych.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyDataBase
{
    public partial class AddProductForm : Form
    {
        public AddProductForm()
        {
            InitializeComponent();
        }


        private void resetBoxes()
        {
            nameTextBox.Text = "";
            unitPriceTextBox.Text = "";
            categoryNameTextBox.Text = "";
            unitPriceTextBox.Text = "";
        }

        private void addProductButton_Click(object sender, EventArgs e)
        {
            String productName1 = nameTextBox.Text;
            int unitsInStock1 = 0;
            while (true)
            {
                if (!int.TryParse(unitsInStockTextBox.Text, out
unitsInStock1))
                {
                    Console.WriteLine("Invalid input units in stock - must
be a number");
                }
                else
                    break;
            }
            decimal unitPrice1 = 0;
            while (true)
            {
                if (!Decimal.TryParse(unitPriceTextBox.Text, out
```

```
unitPrice1))
                {
                    Console.WriteLine("Invalid input units in stock - must
be a number");
                }
                else
                    break;
            }

            String categoryName1 = categoryNameTextBox.Text;

            using (var db = new ProdContext())
            {
                var catID = from c in db.Categories
                            where c.Name == categoryName1
                            select c.CategoryID;
                if (catID == null)
                {
                    Console.WriteLine("Invalid category name");
                    this.resetBoxes();
                }
                else
                {
                    Product product = new Product
                    {
                        Name = productName1,
                        UnitsInStock = unitsInStock1,
                        CategoryID = catID.First(),
                        UnitPrice = unitPrice1
                    };

                    db.Products.Add(product);
                    MessageBox.Show("Product added");
                    db.SaveChanges();
                    this.resetBoxes();
                }

            }

        }
    }
}
```

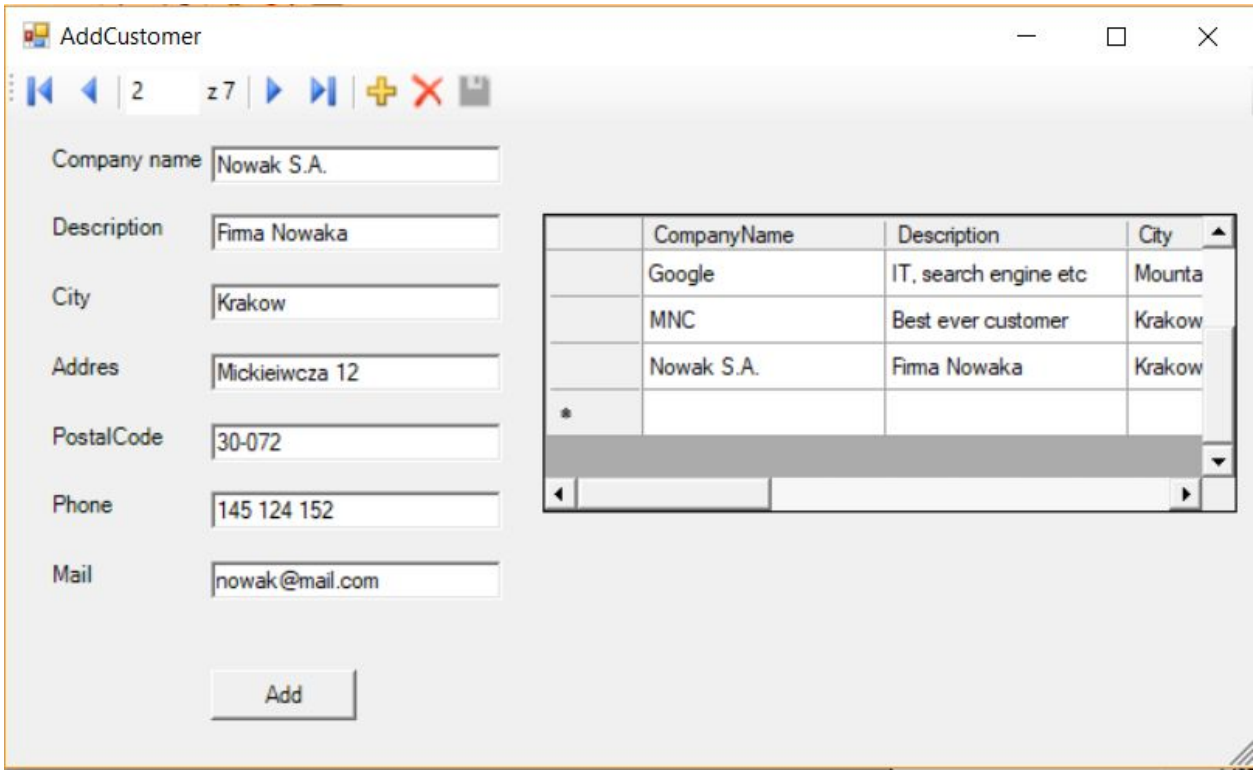


Następny formularz służy dodaniu klienta. Po poprawnym wypełnieniu danych ukaże się tabela z klientami.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyDataBase
{
    public partial class AddCustomer : Form
```

```
    {
        public ProdContext prodContext;
        public AddCustomer()
        {
            InitializeComponent();
            this.prodContext = new ProdContext();
        }

        private void AddCustomer_Load(object sender, EventArgs e)
        {
            prodContext = new ProdContext();
            prodContext.Customers.Load();
            this.customerBindingSource.DataSource =
prodContext.Customers.Local.ToBindingList();

        }

        private void addCustomerButton_Click(object sender, EventArgs e)
        {
            if(this.companyNameTextBox.Text == "" | this.cityTextBox.Text
== "" | this.addressTextBox.Text =="" |
               this.cityTextBox.Text == "" | this.postalCodeTextBox.Text ==
"" )
            {
                MessageBox.Show("Fullfill all boxes");

            }

            Customer customer = (from c in prodContext.Customers
                                 where c.CompanyName ==
companyNameTextBox.Text
                                 select c).FirstOrDefault();
            if(customer != null)
            {
                MessageBox.Show("This customer already exists");
            }

            Customer customer1 = new Customer
            {
                CompanyName = this.companyNameTextBox.Text,
                Description = this.descriptionTextBox.Text,
                Address = this.addressTextBox.Text,
```

```
                PostalCode = this.postalCodeTextBox.Text,
                City = this.cityTextBox.Text,
                Mail = this.cityTextBox.Text,
                Phone = this.cityTextBox.Text
            };

            prodContext.Customers.Add(customer1);
            prodContext.SaveChanges();
            this.customerDataGridView.Visible = true;
            this.customerDataGridView.Update();
            this.customerDataGridView.Refresh();
            //prodContext.SaveChanges();
            MessageBox.Show("Customer added");


        }
    }
}
```



Kolejny formularz służy do składania zamówień na produkty. Za pomocą ComboBoxów możemy wybrać kategorię i produkt, w okienku pojawia się ilość dostępnych sztuk. Po naciśnięciu przycisku buy cena jest doliczana przy labelu Sum a produkt dodany do listy. Jeśli naciśniemy order, zamówienie zostanie dodane - tworzymy obiekt klasy Order. Oczywiście sprawdzam poprawność danych i aktualizuję dostępne produkty jeśli kupimy produkt.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyDataBase
{
    public partial class AddOrderForm : Form
    {

        private int categoryID;
        private int productID;
        public string companyName1;
        public ProdContext prodContext;
        private decimal sum;
        private decimal price;
        private int available;
        private int quantity;
        List<Product> products = new List<Product>();
        public AddOrderForm()
        {
            InitializeComponent();
            this.prodContext = new ProdContext();
            this.categoryComboBox.DataSource =
prodContext.Categories.ToList();
            sum = 0;
        }

        private void categoryComboBox_SelectedIndexChanged(object sender,
EventArgs e)
```

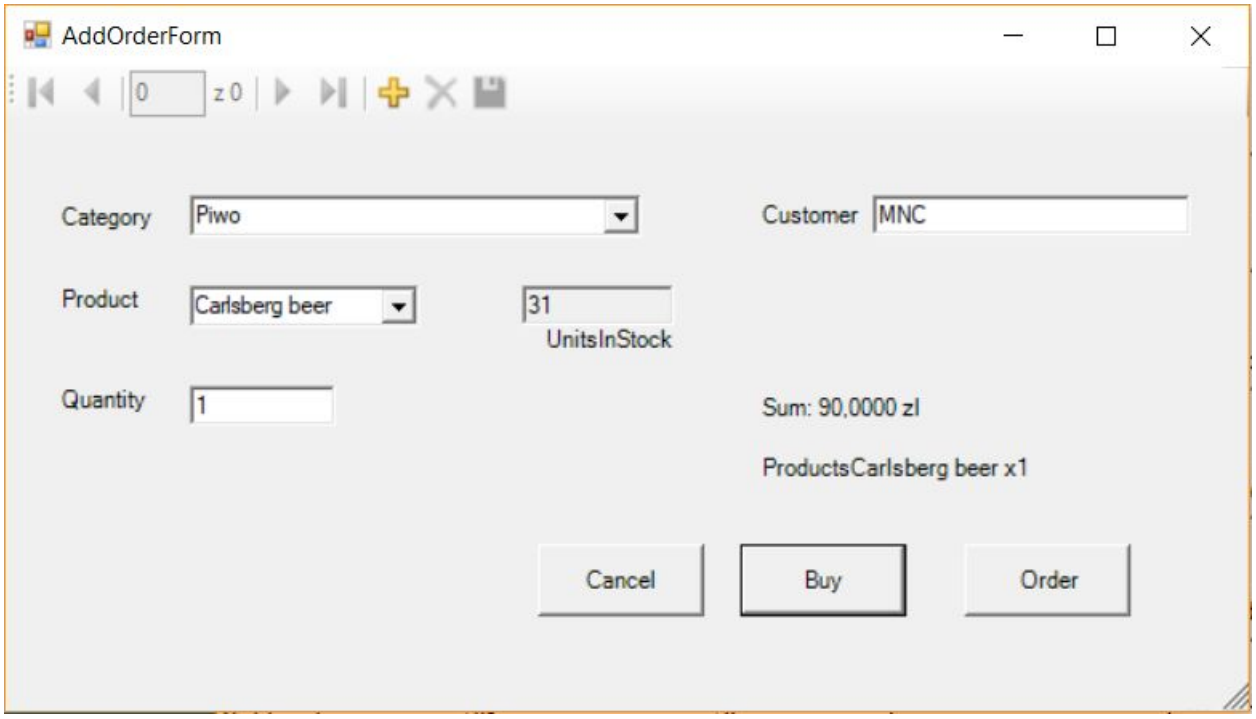
```
        {
            string selectedCategory =
categoryComboBox.GetItemText(categoryComboBox.SelectedItem);
            this.categoryID = prodContext.Categories.Where(c => c.Name ==
selectedCategory).Select(c => c.CategoryID).FirstOrDefault();

            this.productsComboBox.DataSource = prodContext.Products.Where(p
=> p.CategoryID == categoryID).Select(p => p).ToList();



        }

        private void productsComboBox_SelectedIndexChanged(object sender,
EventArgs e)
        {
            string selectedProduct =
productsComboBox.GetItemText(productsComboBox.SelectedItem);
            this.productID = prodContext.Products.Where(p => p.Name ==
selectedProduct).Select(p => p.ProductId).FirstOrDefault();
            available = prodContext.Products.Where(p => p.ProductId ==
productID).Select(p => p.UnitsInStock).FirstOrDefault();

            this.unitsInStockTextBox.Text = available.ToString();
            Console.WriteLine(available);

        }

        private void AddOrderForm_Load(object sender, EventArgs e)
        {
            prodContext.Categories.Load();
            prodContext.Products.Load();
        }

        private void buyButton_Click(object sender, EventArgs e)
        {
            if (this.customerNameTextBox.Text == null)
            {
                MessageBox.Show("Put your customer data");
            }
            quantity = int.Parse(this.quantityTextBox.Text);
            if (quantity > this.available | quantity < 0)
            {
```

```
                MessageBox.Show("Invalid quantity");
            }

            companyName1 = customerNameTextBox.Text;
            Customer company = prodContext.Customers.Where(c =>
c.CompanyName == companyName1).Select(c => c).FirstOrDefault();
            if (company == null)
            {
                MessageBox.Show("Invalid company name");
                return;
            }

            price = prodContext.Products.Where(p => p.ProductId ==
productID).Select(p => p.UnitPrice).FirstOrDefault();
            string prodName = prodContext.Products.Where(p => p.ProductId
== productID).Select(p => p.Name).FirstOrDefault();
            decimal value = price * quantity;
            this.sum += value;


            sumLabel1.Text = "Sum: " + sum.ToString() + " zl ";

            this.productsLabel.Text += prodName + " x" +
quantity.ToString();

            Product product2 = new Product();
            product2.ProductId = productID;
            product2.Name = prodName;
            product2.UnitsInStock = quantity;
            product2.UnitPrice = price;
            products.Add(product2);

            Product product = (from p in prodContext.Products
                               where p.ProductId == productID
                               select p).FirstOrDefault();
            product.UnitsInStock = available - quantity;
            prodContext.SaveChanges();
        }



        private void orderButton_Click(object sender, EventArgs e)
```

```
        {
            Order order = new Order
            {
                Customer = new Customer { CompanyName = companyName1 },
                Products = products,
                Quantity = quantity,
                Price = this.price
            };

            prodContext.Orders.Add(order);
            prodContext.SaveChanges();
            MessageBox.Show("Dokonano zamówienia");
        }

        private void Cancell_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Zamowienie anulowane");
            return;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyDataBase
{
    public class Order
    {
        public int OrderID { get; set; }
        public List <Product> Products { get; set; }
        public int Quantity { get; set; }

        public decimal Price { get; set; }

        [ForeignKey("Customer")]
        public string CompanyName { get; set; }
```

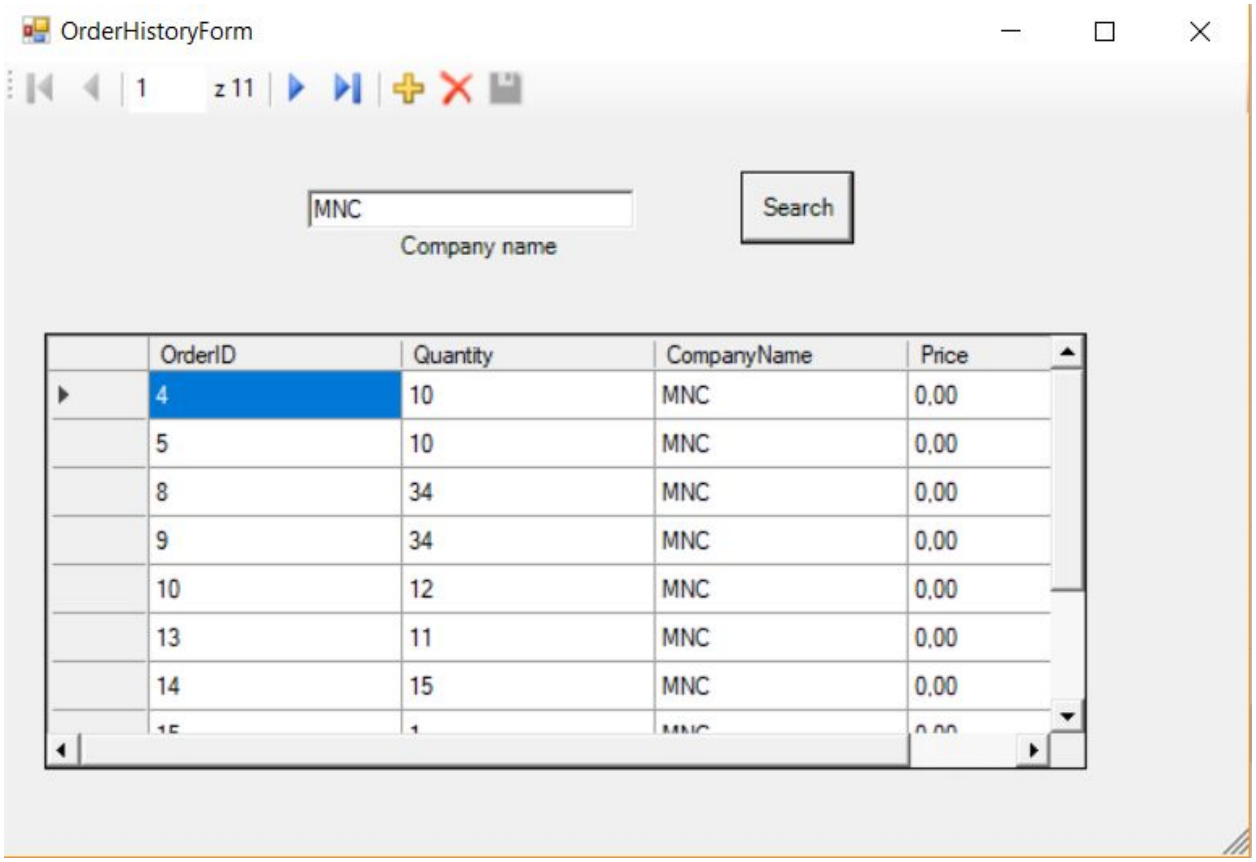
```
        public virtual Customer Customer { get; set; }

    }
}
```



Zamówienia dla danego klienta pozwala nam zobaczyć formularz OrderHistory.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MyDataBase
```

```
{
    public partial class OrderHistoryForm : Form
    {
        public ProdContext prodContext;
        public OrderHistoryForm()
        {
            InitializeComponent();
        }

        private void searchButton_Click(object sender, EventArgs e)
        {
            this.orderDataGridView.DataSource =
                prodContext.Orders.Select(o => o).Where(c => c.CompanyName
==
                categoryNameTextBox.Text.ToString()).Select(company =>
company).ToList();

            this.orderDataGridView.Update();
            this.orderDataGridView.Refresh();
        }

        public void OrderHistoryForm_Load(object sender, EventArgs e)
        {
            this.prodContext = new ProdContext();
            this.prodContext.Orders.Load();
            this.orderBindingSource.DataSource =
prodContext.Orders.Local.ToBindingList();
        }
    }
}
```


OrderHistoryForm
1
z 11
MNC
Company name
Search
OrderID
Quantity
CompanyName
Price
4
10
MNC
0,00
5
10
MNC
0,00
8
34
MNC
0,00
9
34
MNC
0,00
10
12
MNC
0,00
13
11
MNC
0,00
14
15
MNC
0,00