

SKLEP INTERNETOWY

Skład grupy:

- Bartłomiej Umiński (105792),
- Kacper Siegieńczuk (105741),
- Michał Kozikowski (105591),
- Jakub Kozłowski (105593).

Grupa PS2.

Spis treści:

1. WSTĘP	2
2. ANALIZA WYMAGAŃ SYSTEMU	2
2.1. WYMAGANIA FUNKCJONALNE.....	2
2.2. WYMAGANIA NIEFUNKCJONALNE	2
2.3. DIAGRAM PRZYPADKÓW UŻYCIA.....	3
3. WYKORZYSTANE TECHNOLOGIE	3
4. PROJEKT APLIKACJI.....	3
4.1. ARCHITEKTURA APLIKACJI	3
4.2. PROJEKT KONCEPCYJNY BAZY DANYCH.....	4
4.3. PROJEKT SCHEMATU RELACYJNEGO	4
4.4. MAPOWANIE KLAS NA TABELĘ BAZODANOWE.....	4
5. FUNKCJONALNOŚĆ APLIKACJI.....	29
6. INTERFEJS UŻYTKOWNIKA.....	29
7. PODSUMOWANIE.....	34
8. ETAPY TWORZENIA APLIKACJI	34
9. DOKŁADNY OPIS TABEL BAZY DANYCH.....	35
10. DODATEK A: SKRYPTY TWORZĄCE OBIEKTY BAZ DANYCH	36

1. WSTĘP

Aplikacja ma służyć do zarządzania danymi i zasobami sklepu. Umożliwia ona dokonywania zamówień online przez klientów sklepu. Klient może wyświetlić listę produktów z danej kategorii i dodać je do koszyka, aby następnie złożyć zamówienie. Całość kontrolowana jest przez administratorów systemu, którzy mają możliwość dodawania, usuwania i edytowania produktów i kategorii, modyfikowania i przeglądania zamówień oraz zarządzania kontami użytkowników. Istnieje również możliwość użycia kodu rabatowego przez klienta.

2. ANALIZA WYMAGAŃ SYSTEMU

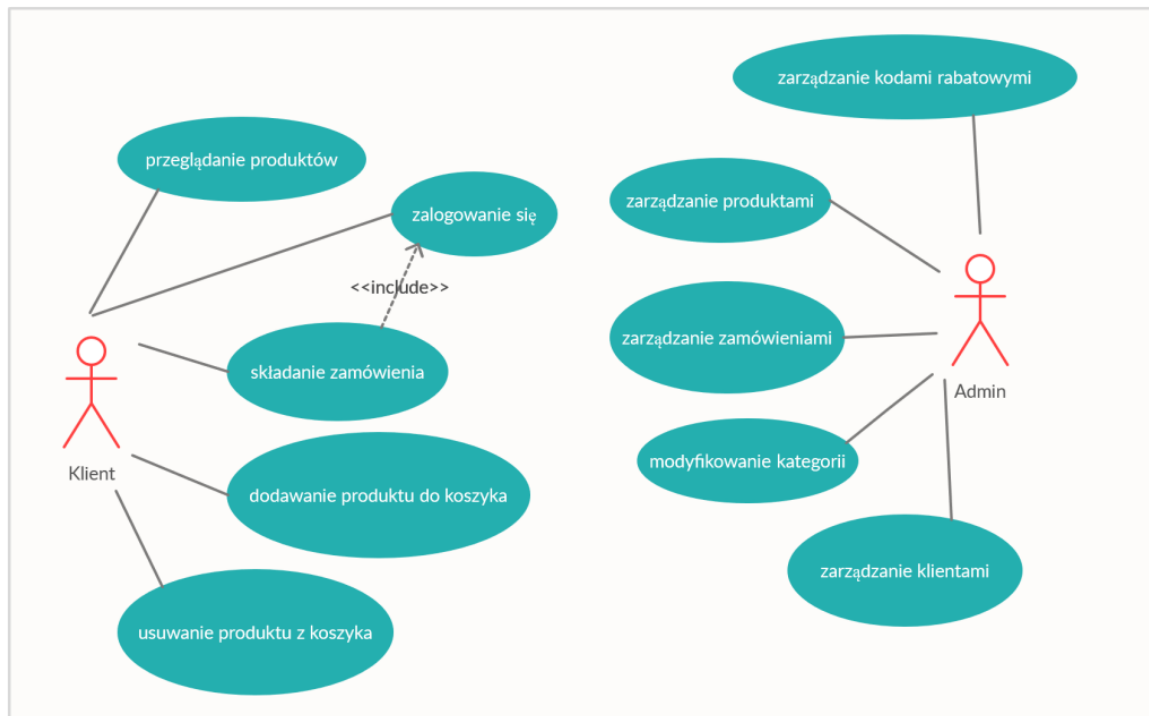
2.1. WYMAGANIA FUNKCJONALNE

- możliwość zakładania konta i zalogowania się za jego pomocą w aplikacji,
- możliwość przeglądania produktów (w postaci listy) dostępnych w sklepie,
- możliwość przeglądania produktów wg. kategorii,
- możliwość dodawania produktów do wirtualnego koszyka,
- możliwość zakupienia produktów wybierając preferowany sposób płatności oraz dostawy,
- możliwość dodawania, usuwania oraz edytowania produktów dostępnych w bazie danych sklepu jeśli użytkownik posiada prawa administracyjne,
- możliwość edycji tabel ukrytych przed zwykłymi użytkownikami przez użytkowników z prawami administracyjnymi.

2.2. WYMAGANIA NIEFUNKCJONALNE

- wygodne kupowanie sprzętu przez internet,
- przejrzystość aplikacji,
- szybka reakcja na działanie użytkownika
- zapewnienie odpowiednich narzędzi do zarządzania produktami w sklepie,
- prostota w edytowaniu bazy danych.

2.3. DIAGRAM PRZYPADKÓW UŻYCIA



3. WYKORZYSTANE TECHNOLOGIE

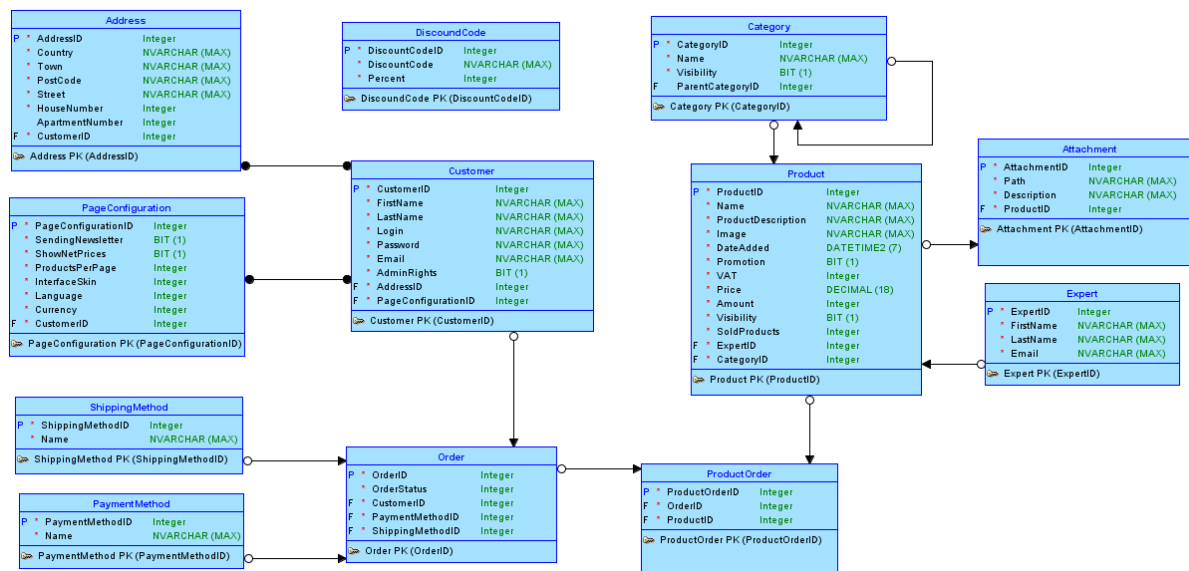
Podczas realizacji projektu wykorzystaliśmy program *sqldeveloper* do wygenerowania diagramu ER i diagramu relacyjnego. Do implementacji projektu wybraliśmy język C# i środowisko *VisualStudio*. Stworzyliśmy lokalną bazę danych z której korzysta nasza aplikacja. Baza danych jest postawiona na lokalnym serwerze SQL Server. Aplikację wykonaliśmy w technologii *.NET Core MVC*. Mapowanie modeli na tabele bazy danych wykonaliśmy przy pomocy narzędzia *Entity Framework*. Jest to platforma ORM, dzięki której można mapować modele na tabele bazodanowe.

4. PROJEKT APLIKACJI

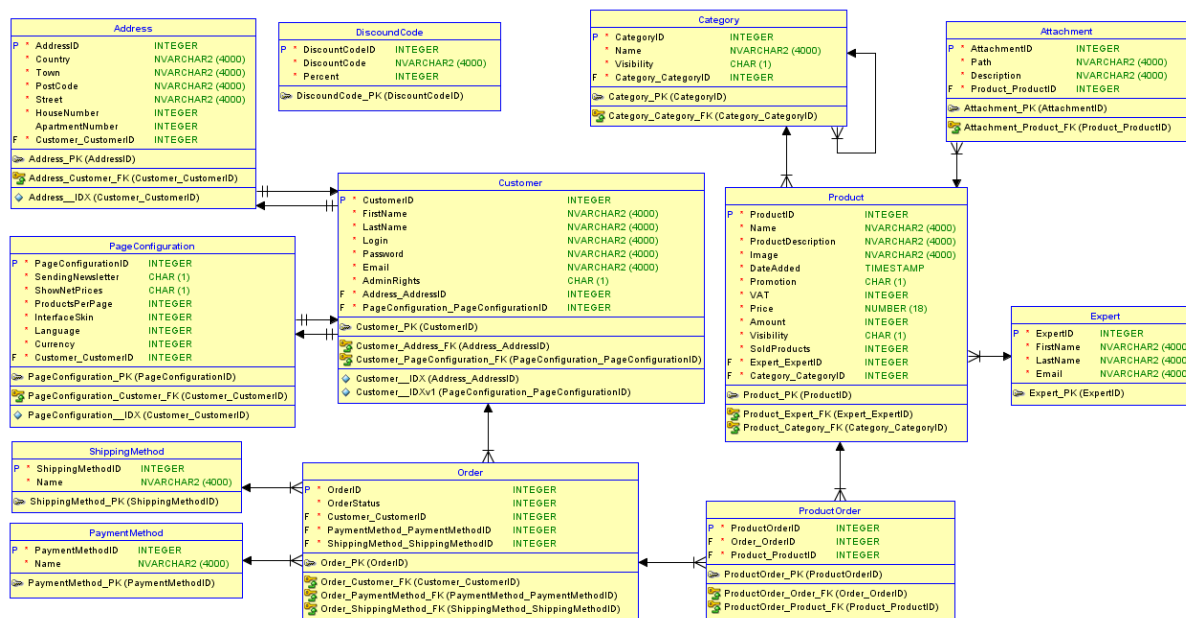
4.1. ARCHITEKTURA APLIKACJI

Architektura aplikacji opiera się na wzorcu MVC (Model-View-Controller). Użytkownik ma możliwość poruszania się na serwisie po różnych widokach stron. Połączone są one z odpowiadającymi im kontrolerami, które mogą modyfikować dane w bazie danych, korzystając z odpowiednich modeli. Po uaktualnieniu danych zmieniają konkretne widoki, które następnie są prezentowane użytkownikowi. Aplikacja nie wykorzystuje zewnętrznego serwera, a korzysta z lokalnej bazy danych. Dodatkowo, wykorzystuje ona *Entity Framework* do wykonania procesu mapowania i korzystania z bazy danych oraz z *Identity* do zaimplementowania mechanizmu logowania.

4.2. PROJEKT KONCEPCYJNY BAZY DANYCH



4.3. PROJEKT SCHEMATU RELACYJNEGO



4.4. MAPOWANIE KLAS NA TABELY BAZODANOWE

Mapowanie zostało wykonane za pomocą narzędzia Entity Framework. Wykorzystaliśmy do tego mechanizm migracji modeli na tabeli bazy danych. Mapowanie odbywa się poprzez umieszczenie dodatkowych pól i atrybutów w modelach. To za ich pomocą definiujemy relacje między modelami i wpływamy na szczegóły wygenerowanej bazy danych.

W naszym projekcie wykorzystaliśmy atrybuty takie jak:

- Key - klucz główny
- ForeignKey - klucz obcy

- Required - pole wymagane (np. przy dodaniu nowego rekordu do bazy danych)
- DisplayFormat - określenie jak dane mają być formatowane w danym polu

Do stworzenia relacji jeden do jednego, jeden do wielu i wiele do wielu należy w modelach utworzyć pola lub kolekcje (listę) modeli z którymi chcemy stworzyć relację.

Przykłady relacji użytych w naszym programie:

- 1 do 1 - relacja między Adresem (*Address*) i Klientem (*Customer*) - w założeniu naszego projektu klient może posiadać tylko jeden adres, więc użyliśmy w tym przypadku relacji 1 do 1. Poniżej pokazane są potrzebne pola do stworzenia relacji jeden do jednego między adresem i klientem.

```
public class Address
{
    /* POLA */
    [Key]
    public int AddressID { get; set; }
    [Required]
    public int CustomerID { get; set; }
    [Required]
    public string Country { get; set; }
    [Required]
    public string Town { get; set; }
    [Required]
    public string PostCode { get; set; }
    [Required]
    public string Street { get; set; }
    [Required]
    public int HouseNumber { get; set; }
    public int? ApartmentNumber { get; set; }
}
```

```
public class Customer
{
    /* POLA */
    [Key]
    public int CustomerID { get; set; }
    [Required]
    [ForeignKey("Address")]
    public int AddressID { get; set; }
    [Required]
    [ForeignKey("PageConfiguration")]
    public int PageConfigurationID { get; set; }
    [Required]
    public string FirstName { get; set; }
    [Required]
    public string LastName { get; set; }
    [Required]
    public string Login { get; set; }
    [Required]
    public string Password { get; set; }
    [Required]
    public string Email { get; set; }
    [Required]
    public bool AdminRights { get; set; }

    /* POLA - ENTITY FRAMEWORK */
    // [ForeignKey("AddressID")]
    public Address Address { get; set; }
    // [ForeignKey("PageConfigurationID")]
    public PageConfiguration PageConfiguration { get; set; }
    public ICollection<Order> Orders { get; set; }
}
```

- 1 do wielu - relacja między Klientem (*Customer*) a Zamówieniem (*Order*) - jeden klient może mieć wiele zamówień. W tym przypadku klient musi posiadać kolekcję zamówień, a zamówienie to tylko pole z identyfikatorem klienta i pole z klientem.

```
public class Customer
{
    /* POLA */
    [Key]
    public int CustomerID { get; set; }
    [Required]
    [ForeignKey("Address")]
    public int AddressID { get; set; }
    [Required]
    [ForeignKey("PageConfiguration")]
    public int PageConfigurationID { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Login { get; set; }
    public string Password { get; set; }
    public string Email { get; set; }
    public bool AdminRights { get; set; }

    /* POLA - ENTITY FRAMEWORK */
    [ForeignKey("AddressID")]
    public Address Address { get; set; }
    [ForeignKey("PageConfigurationID")]
    public PageConfiguration PageConfiguration { get; set; }
    public ICollection<Order> Orders { get; set; }
}
```

```
public class Order
{
    /* POLA */
    [Key]
    public int OrderID { get; set; }
    [Required]
    [ForeignKey("Customer")]
    public int CustomerID { get; set; }
    [Required]
    [ForeignKey("ShippingMethod")]
    public int ShippingMethodID { get; set; }
    [Required]
    [ForeignKey("PaymentMethod")]
    public int PaymentMethodID { get; set; }
    [EnumDataType(typeof(State))]
    public State OrderStatus { get; set; }

    /* POLA - ENTITY FRAMEWORK */
    [ForeignKey("CustomerID")]
    public Customer Customer { get; set; }
    [ForeignKey("ShippingMethodID")]
    public ShippingMethod ShippingMethod { get; set; }
    [ForeignKey("PaymentMethodID")]
    public PaymentMethod PaymentMethod { get; set; }
    public ICollection<ProductOrder> ProductOrders { get; set; }
}
```

- wiele do wielu - relacja między Zamówieniem (*Order*) a Produktem (*Product*) - w jednym zamówieniu może być wiele produktów. Jeden produkt może być w wielu zamówieniach. Do stworzenia tej relacji utworzyliśmy model *ProductOrder* na podstawie którego tworzy się tabela pośrednia między tabelą *Product* i tabelą *Order*. W tym przypadku w modelach produktu i zamówienia dodane zostały listy na obiekty *ProductOrder*. W modelu *ProductOrder* natomiast dodane zostały pola z identyfikatorami produktu i zamówienia oraz pola przechowujące produkt i zamówienie.

```

public class Product
{
    /* POLA */
    [Key]
    Odwołania: 25
    public int ProductID { get; set; }
    [Required]
    [ForeignKey("Category")]
    Odwołania: 12
    public int CategoryID { get; set; }
    [Required]
    [ForeignKey("Expert")]
    Odwołania: 12
    public int ExpertID { get; set; }
    [Required]
    Odwołania: 18
    public string Name { get; set; }
    [Required]
    Odwołania: 16
    public string ProductDescription { get; set; }
    [Required]
    Odwołania: 20
    public string Image { get; set; }
    [Required]
    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    Odwołania: 16
    public DateTime DateAdded { get; set; }
    [Required]
    Odwołania: 14
    public bool Promotion { get; set; }
    [Required]
    Odwołania: 16
    public int VAT { get; set; }
    [Required]
    [Column(TypeName = "decimal(18,2)")]
    Odwołania: 20
    public decimal Price { get; set; }
    [Required]
    Odwołania: 16
    public int Amount { get; set; }
    [Required]
    Odwołania: 14
    public bool Visibility { get; set; }
    [Required]
    Odwołania: 16
    public int SoldProducts { get; set; }

    /* POLA - ENTITY FRAMEWORK */
    //[ForeignKey("CategoryID")]
    Odwołania: 13
    public Category Category { get; set; }
    //[ForeignKey("ExpertID")]
    Odwołania: 12
    public Expert Expert { get; set; }
    Odwołania: 0
    public ICollection<Attachment> Attachments { get; set; }
    Odwołania: 0
    public ICollection<ProductOrder> ProductOrders { get; set; }
}

```

```

public class Order
{
    /* POLA */
    [Key]
    Odwołania: 17
    public int OrderID { get; set; }
    [Required]
    [ForeignKey("Customer")]
    Odwołania: 14
    public int CustomerID { get; set; }
    [Required]
    [ForeignKey("ShippingMethod")]
    Odwołania: 14
    public int ShippingMethodID { get; set; }
    [Required]
    [ForeignKey("PaymentMethod")]
    Odwołania: 14
    public int PaymentMethodID { get; set; }
    [Required]
    [EnumDataType(typeof(State))]
    Odwołania: 16
    public State OrderStatus { get; set; }

    /* POLA - ENTITY FRAMEWORK */
    //[ForeignKey("CustomerID")]
    Odwołania: 9
    public Customer Customer { get; set; }
    //[ForeignKey("ShippingMethodID")]
    Odwołania: 9
    public ShippingMethod ShippingMethod { get; set; }
    //[ForeignKey("PaymentMethodID")]
    Odwołania: 9
    public PaymentMethod PaymentMethod { get; set; }
    Odwołania: 9
    public ICollection<ProductOrder> ProductOrders { get; set; }
}

```

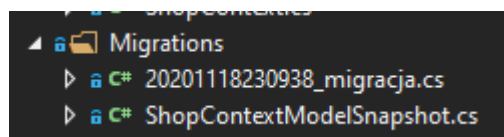
```

public class ProductOrder
{
    /* POLA */
    [Key]
    Odwołania: 13
    public int ProductOrderID { get; set; }
    [Required]
    [ForeignKey("Product")]
    Odwołania: 14
    public int ProductID { get; set; }
    [Required]
    [ForeignKey("Order")]
    Odwołania: 14
    public int OrderID { get; set; }

    /* POLA - ENTITY FRAMEWORK */
    //[ForeignKey("OrderID")]
    Odwołania: 9
    public Order Order { get; set; }
    //[ForeignKey("ProductID")]
    Odwołania: 9
    public Product Product { get; set; }
}

```

Po stworzeniu wszystkich relacji dodaliśmy migrację, by zsynchronizować nasze modele z bazą danych. Użyliśmy do tego polecenia: `add-migration <migration_name>`. Spowodowało to wygenerowanie plików odpowiedzialnych za stworzenie bazy danych do naszego projektu.



W pliku o nazwie `20201118230938_migracja.cs` znajdują się klasa o nazwie `migracja`, w której zdefiniowane są dwie metody `Up` i `Down`. Metoda `Up` jest wywoływana podczas tworzenia bazy danych, a metoda `Down` podczas usuwania bazy danych. W metodzie `Up` za pomocą metod `CreateTable` tworzymy wszystkie tabele. W tej metodzie zdefiniowane są nazwy tabel, kolumny tabel, nazwy kolumn tabel, typy kolumn tabel, oraz definicje kluczy głównych i obcych w tabelach.

W podanym poniżej fragmencie kodu tworzymy tabelę *ProductOrder*, która ma trzy kolumny *ProductOrderID*, *ProductID* i *OrderID*. Są to kolumny typu *int* i nie mogą przyjmować wartości *null*. Kolumna *ProductOrderID* jest ustawiona jako Identity, czyli jest to kolumna z automatycznie generowanymi identyfikatorami. Pod definicją kolumn mamy też definicję kluczy głównych i obcych potrzebnych do prawidłowego działania relacji w bazie danych. W tym przypadku kluczem głównym jest *ProductOrderID*, a kluczami obcymi są *ProductID* i *OrderID*. Zdefiniowane mamy też nazwy tabel i kolumn w których znajdują się klucze obce.

```
migrationBuilder.CreateTable(
    name: "ProductOrder",
    columns: table => new
    {
        ProductOrderID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        ProductID = table.Column<int>(nullable: false),
        OrderID = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_ProductOrder", x => x.ProductOrderID);
        table.ForeignKey(
            name: "FK_ProductOrder_Order_OrderID",
            column: x => x.OrderID,
            principalTable: "Order",
            principalColumn: "OrderID",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_ProductOrder_Product_ProductID",
            column: x => x.ProductID,
            principalTable: "Product",
            principalColumn: "ProductID",
            onDelete: ReferentialAction.Cascade);
    });
```

W metodzie *Up* odbywa się też tworzenie indeksów, są one używane do szybkiego wyszukiwania danych w bazie na podstawie wartości klucza. Na przykład poniżej tworzony jest indeks w tabeli *Order* do kolumny *CustomerID*.

```
migrationBuilder.CreateIndex(
    name: "IX_Order_CustomerID",
    table: "Order",
    column: "CustomerID");
```


W metodzie *Down* znajdują się instrukcje służące do usunięcia wszystkich tabel z bazy. Na poniższym przykładzie widzimy usunięcie tabeli zamówień i produktów.

```
migrationBuilder.DropTable(  
    name: "Order");  
  
migrationBuilder.DropTable(  
    name: "Product");
```

W pliku o nazwie *ShopContextModelSnapshot.cs* znajduje się klasa o nazwie *ShopContextModelSnapshot*. W tej klasie w metodzie *BuildModel* tworzymy model bazy danych. Za pomocą *modelBuilder'a* i metody *Entity* konfigurujemy encje naszej bazy danych. Między innymi określamy typy i nazwy kolumn encji oraz ich dodatkowe parametry takie jak automatyczne generowanie identyfikatorów. Na przykład poniżej określamy kolumny i typ encji *Order*. Dodajemy też używane przez tą tabelę indeksy i klucz podstawowy.

```
modelBuilder.Entity("ProjektSklep.Models.Order", b =>  
{  
    b.Property<int>("OrderID")  
        .ValueGeneratedOnAdd()  
        .HasColumnType("int")  
        .HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);  
  
    b.Property<int>("CustomerID")  
        .HasColumnType("int");  
  
    b.Property<int>("OrderStatus")  
        .HasColumnType("int");  
  
    b.Property<int>("PaymentMethodID")  
        .HasColumnType("int");  
  
    b.Property<int>("ShippingMethodID")  
        .HasColumnType("int");  
  
    b.HasKey("OrderID");  
  
    b.HasIndex("CustomerID");  
  
    b.HasIndex("PaymentMethodID");  
  
    b.HasIndex("ShippingMethodID");  
  
    b.ToTable("Order");  
});
```

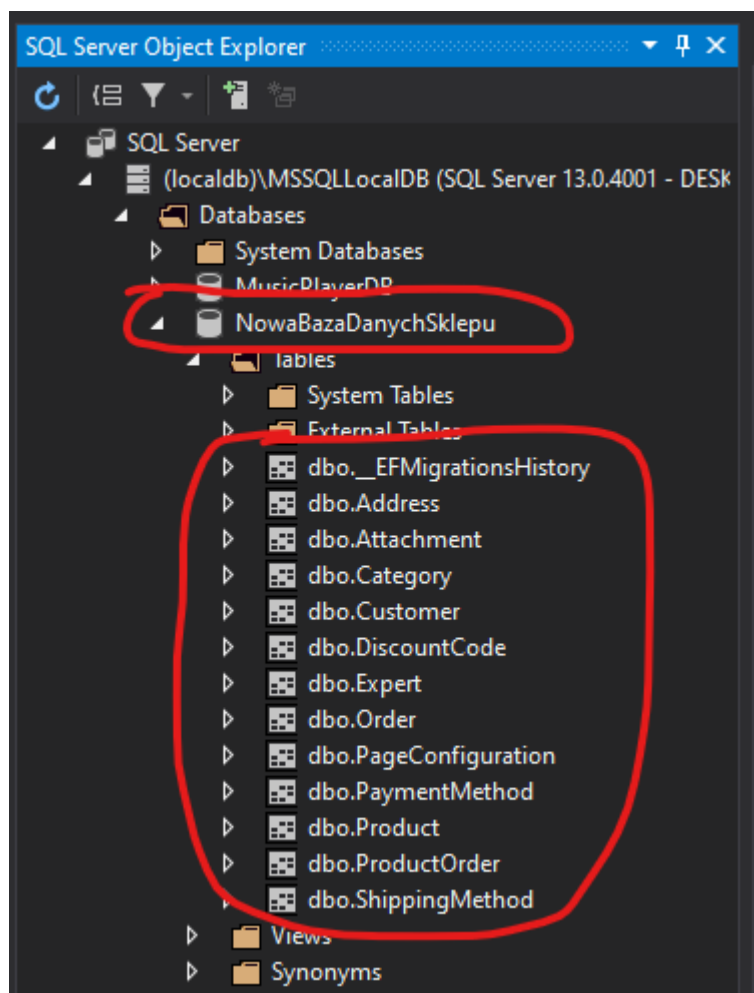
W pliku tym tworzone są też relacje. Na poniższym przykładzie mamy model zamówienia w którym określiliśmy połączenia z klientem, metodą płatności i metodą dostawy. Zamówienie posiada trzy relacje jeden do wielu. Dlatego, że jeden klient może wykonać wiele zamówień. Wiele zamówień może mieć tą samą metodę płatności lub dostawy. Dlatego poniżej mamy określone *HasOne*("Customer") i *WithMany*("Orders"), co oznacza że jeden klient może mieć wiele zamówień. Określony tu jest też klucz obcy użyty w relacji, to czy pola są wymagane, i zachowanie podczas usunięcia elementów.

```
modelBuilder.Entity("ProjektSklep.Models.Order", b =>
{
    b.HasOne("ProjektSklep.Models.Customer", "Customer")
        .WithMany("Orders")
        .HasForeignKey("CustomerID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();

    b.HasOne("ProjektSklep.Models.PaymentMethod", "PaymentMethod")
        .WithMany("Orders")
        .HasForeignKey("PaymentMethodID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();

    b.HasOne("ProjektSklep.Models.ShippingMethod", "ShippingMethod")
        .WithMany("Orders")
        .HasForeignKey("ShippingMethodID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();
});
```

Po przejrzaniu powyżej omówionych plików i sprawdzenia ich zgodności z naszymi oczekiwaniami. Wygenerowaliśmy bazę danych poleceniem *update-database*. Poniżej zrzut ekranu pokazujący tabele wygenerowanej bazy danych:



Do korzystania z utworzonej bazy potrzebna nam klasa kontekstu. Stworzyliśmy klasę *ShopContext*, w niej zawarliśmy deklaracje *DbSet*-ów czyli klas reprezentujących zbiory encji, które mogą być użyte do dodawania, usuwania, edytowania i odczytywania obiektów z bazy danych. W klasie tej mamy też definicje *ConnectionStringa* służącego jako połączenie z naszą lokalną bazą danych. W metodzie *OnModelCreating* określamy nazwy tabel do jakich mają się mapować dane klasy. Poniżej podany został fragment klasy kontekstu.

```

Odwolania: 36
public class ShopContext : DbContext
{
    Odwolania: 3
    public ShopContext()
    {
    }

    Odwolania: 0
    public ShopContext(DbContextOptions<ShopContext> options) : base(options)
    {
    }

    Odwolania: 10
    public DbSet<DiscountCode> DiscountCodes { get; set; }
    Odwolania: 13
    public DbSet<Address> Addresses { get; set; }
    Odwolania: 13
    public DbSet<Customer> Customers { get; set; }
    Odwolania: 13
    public DbSet<PageConfiguration> PageConfigurations { get; set; }
    Odwolania: 14
    public DbSet<Order> Orders { get; set; }
    Odwolania: 16
    public DbSet<ShippingMethod> ShippingMethods { get; set; }
    Odwolania: 16
    public DbSet<PaymentMethod> PaymentMethods { get; set; }
    Odwolania: 20
    public DbSet<Product> Products { get; set; }
    Odwolania: 19
    public DbSet<Category> Categories { get; set; }
    Odwolania: 9
    public DbSet<Attachment> Attachments { get; set; }
    Odwolania: 13
    public DbSet<Expert> Experts { get; set; }
    Odwolania: 10
    public DbSet<ProductOrder> ProductOrders { get; set; }

    Odwolania: 0
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer("Server=(localdb)\\MSSQLLocalDB;Database=NowaBazaDanychSklepu;Trusted_Connection=True;");
        base.OnConfiguring(optionsBuilder);
    }
    Odwolania: 0
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<DiscountCode>().ToTable("DiscountCode");
        modelBuilder.Entity<Address>().ToTable("Address");
        modelBuilder.Entity<Customer>().ToTable("Customer");
    }
}

```

W naszym programie inicjalizujemy bazę danych danymi, które dodajemy w klasie *DbInitializer*. Na przykład poniższy fragment kodu dodaje kody rabatowe do bazy danych.

```

1 odwołanie
public static class DbInitializer
{
    1 odwołanie
    public static void Initialize(ShopContext context)
    {
        context.Database.EnsureCreated();

        /* DiscountCodes */
        if (context.DiscountCodes.Any())
        {
            return;
        }
        var discountCodes = new DiscountCode[]
        {
            new DiscountCode{ DiscountCode="123ABC", Percent=50 },
            new DiscountCode{ DiscountCode="234BCD", Percent=30 },
            new DiscountCode{ DiscountCode="123ABC", Percent=10 },
            new DiscountCode{ DiscountCode="123XYZ", Percent=20 },
            new DiscountCode{ DiscountCode="789KLM", Percent=70 }
        };

        foreach (DiscountCode discountCode in discountCodes)
        {
            context.DiscountCodes.Add(discountCode);
        }
        context.SaveChanges();
    }
}

```

Inicjalizujemy naszą bazę przy jej pierwszym stworzeniu w klasie *Program.cs*.

```

1 odwołanie
public class Program
{
    Odwołania:0
    public static void Main(string[] args)
    {
        var host = CreateHostBuilder(args).Build();

        CreateDbIfNotExists(host);

        host.Run();
    }

    1 odwołanie
    private static void CreateDbIfNotExists(IHost host)
    {
        using (var scope = host.Services.CreateScope())
        {
            var services = scope.ServiceProvider;
            try
            {
                var context = services.GetRequiredService<ShopContext>();
                DbInitializer.Initialize(context);
            }
            catch (Exception ex)
            {
                var logger = services.GetRequiredService<ILogger<Program>>();
                logger.LogError(ex, "An error occurred creating the DB.");
            }
        }
    }
}

```

Jak widać mapowanie w Entity Framework ułatwia tworzenie aplikacji. Dzięki niemu możemy szybciej tworzyć aplikacje bazodanowe.

Kod z pliku 20201118230938_migracja.cs:

```

using System;
using Microsoft.EntityFrameworkCore.Migrations;

namespace ProjektSklep.Migrations
{
    public partial class migracja : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Address",
                columns: table => new
                {
                    AddressID = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    CustomerID = table.Column<int>(nullable: false),
                    Country = table.Column<string>(nullable: true),
                    Town = table.Column<string>(nullable: true),
                    PostCode = table.Column<string>(nullable: true),

```

```

        Street = table.Column<string>(nullable: true),
        HouseNumber = table.Column<int>(nullable: false),
        ApartmentNumber = table.Column<int>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Address", x => x.AddressID);
    });
});

migrationBuilder.CreateTable(
    name: "Category",
    columns: table => new
    {
        CategoryID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        ParentCategoryID = table.Column<int>(nullable: true),
        Name = table.Column<string>(nullable: true),
        Visibility = table.Column<bool>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Category", x => x.CategoryID);
        table.ForeignKey(
            name: "FK_Category_Category_ParentCategoryID",
            column: x => x.ParentCategoryID,
            principalTable: "Category",
            principalColumn: "CategoryID",
            onDelete: ReferentialAction.Restrict);
    });
});

migrationBuilder.CreateTable(
    name: "DiscountCode",
    columns: table => new
    {
        DiscountCodeID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        DiscountCode = table.Column<string>(nullable: false),
        Percent = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_DiscountCode", x => x.DiscountCodeID);
    });
});

migrationBuilder.CreateTable(
    name: "Expert",
    columns: table => new
    {
        ExpertID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        FirstName = table.Column<string>(nullable: true),
        LastName = table.Column<string>(nullable: true),
        Email = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Expert", x => x.ExpertID);
    });
});

```

```

    });

migrationBuilder.CreateTable(
    name: "PageConfiguration",
    columns: table => new
    {
        PageConfigurationID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        CustomerID = table.Column<int>(nullable: false),
        SendingNewsletter = table.Column<bool>(nullable: false),
        ShowNetPrices = table.Column<bool>(nullable: false),
        ProductsPerPage = table.Column<int>(nullable: false),
        InterfaceSkin = table.Column<int>(nullable: false),
        Language = table.Column<int>(nullable: false),
        Currency = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_PageConfiguration", x =>
x.PageConfigurationID);
    });

migrationBuilder.CreateTable(
    name: "PaymentMethod",
    columns: table => new
    {
        PaymentMethodID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        Name = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_PaymentMethod", x => x.PaymentMethodID);
    });

migrationBuilder.CreateTable(
    name: "ShippingMethod",
    columns: table => new
    {
        ShippingMethodID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        Name = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_ShippingMethod", x => x.ShippingMethodID);
    });

migrationBuilder.CreateTable(
    name: "Product",
    columns: table => new
    {
        ProductID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        CategoryID = table.Column<int>(nullable: false),
        ExpertID = table.Column<int>(nullable: false),
        Name = table.Column<string>(nullable: true),

```



```

        ProductDescription = table.Column<string>(nullable: true),
        Image = table.Column<string>(nullable: true),
        DateAdded = table.Column<DateTime>(nullable: false),
        Promotion = table.Column<bool>(nullable: false),
        VAT = table.Column<int>(nullable: false),
        Price = table.Column<decimal>(type: "decimal(18,2)", nullable:
false),

        Amount = table.Column<int>(nullable: false),
        Visibility = table.Column<bool>(nullable: false),
        SoldProducts = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Product", x => x.ProductID);
        table.ForeignKey(
            name: "FK_Product_Category_CategoryID",
            column: x => x.CategoryID,
            principalTable: "Category",
            principalColumn: "CategoryID",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_Product_Expert_ExpertID",
            column: x => x.ExpertID,
            principalTable: "Expert",
            principalColumn: "ExpertID",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "Customer",
    columns: table => new
    {
        CustomerID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        AddressID = table.Column<int>(nullable: false),
        PageConfigurationID = table.Column<int>(nullable: false),
        FirstName = table.Column<string>(nullable: true),
        LastName = table.Column<string>(nullable: true),
        Login = table.Column<string>(nullable: true),
        Password = table.Column<string>(nullable: true),
        Email = table.Column<string>(nullable: true),
        AdminRights = table.Column<bool>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Customer", x => x.CustomerID);
        table.ForeignKey(
            name: "FK_Customer_Address_AddressID",
            column: x => x.AddressID,
            principalTable: "Address",
            principalColumn: "AddressID",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_Customer_PageConfiguration_PageConfigurationID",
            column: x => x.PageConfigurationID,
            principalTable: "PageConfiguration",
            principalColumn: "PageConfigurationID",

```

```

        onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "Attachment",
    columns: table => new
    {
        AttachmentID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        ProductID = table.Column<int>(nullable: false),
        Path = table.Column<string>(nullable: false),
        Description = table.Column<string>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Attachment", x => x.AttachmentID);
        table.ForeignKey(
            name: "FK_Attachment_Product_ProductID",
            column: x => x.ProductID,
            principalTable: "Product",
            principalColumn: "ProductID",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "Order",
    columns: table => new
    {
        OrderID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        CustomerID = table.Column<int>(nullable: false),
        ShippingMethodID = table.Column<int>(nullable: false),
        PaymentMethodID = table.Column<int>(nullable: false),
        OrderStatus = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_Order", x => x.OrderID);
        table.ForeignKey(
            name: "FK_Order_Customer_CustomerID",
            column: x => x.CustomerID,
            principalTable: "Customer",
            principalColumn: "CustomerID",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_Order_PaymentMethod_PaymentMethodID",
            column: x => x.PaymentMethodID,
            principalTable: "PaymentMethod",
            principalColumn: "PaymentMethodID",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_Order_ShippingMethod_ShippingMethodID",
            column: x => x.ShippingMethodID,
            principalTable: "ShippingMethod",
            principalColumn: "ShippingMethodID",
            onDelete: ReferentialAction.Cascade);
    });

```

```

migrationBuilder.CreateTable(
    name: "ProductOrder",
    columns: table => new
    {
        ProductOrderID = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        ProductID = table.Column<int>(nullable: false),
        OrderID = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_ProductOrder", x => x.ProductOrderID);
        table.ForeignKey(
            name: "FK_ProductOrder_Order_OrderID",
            column: x => x.OrderID,
            principalTable: "Order",
            principalColumn: "OrderID",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_ProductOrder_Product_ProductID",
            column: x => x.ProductID,
            principalTable: "Product",
            principalColumn: "ProductID",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateIndex(
    name: "IX_Attachment_ProductID",
    table: "Attachment",
    column: "ProductID");

migrationBuilder.CreateIndex(
    name: "IX_Category_ParentCategoryID",
    table: "Category",
    column: "ParentCategoryID");

migrationBuilder.CreateIndex(
    name: "IX_Customer_AddressID",
    table: "Customer",
    column: "AddressID");

migrationBuilder.CreateIndex(
    name: "IX_Customer_PageConfigurationID",
    table: "Customer",
    column: "PageConfigurationID");

migrationBuilder.CreateIndex(
    name: "IX_Order_CustomerID",
    table: "Order",
    column: "CustomerID");

migrationBuilder.CreateIndex(
    name: "IX_Order_PaymentMethodID",
    table: "Order",
    column: "PaymentMethodID");

```

```

migrationBuilder.CreateIndex(
    name: "IX_Order_ShippingMethodID",
    table: "Order",
    column: "ShippingMethodID");

migrationBuilder.CreateIndex(
    name: "IX_Product_CategoryID",
    table: "Product",
    column: "CategoryID");

migrationBuilder.CreateIndex(
    name: "IX_Product_ExpertID",
    table: "Product",
    column: "ExpertID");

migrationBuilder.CreateIndex(
    name: "IX_ProductOrder_OrderID",
    table: "ProductOrder",
    column: "OrderID");

migrationBuilder.CreateIndex(
    name: "IX_ProductOrder_ProductID",
    table: "ProductOrder",
    column: "ProductID");
}

protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.DropTable(
        name: "Attachment");

    migrationBuilder.DropTable(
        name: "DiscountCode");

    migrationBuilder.DropTable(
        name: "ProductOrder");

    migrationBuilder.DropTable(
        name: "Order");

    migrationBuilder.DropTable(
        name: "Product");

    migrationBuilder.DropTable(
        name: "Customer");

    migrationBuilder.DropTable(
        name: "PaymentMethod");

    migrationBuilder.DropTable(
        name: "ShippingMethod");

    migrationBuilder.DropTable(
        name: "Category");

    migrationBuilder.DropTable(
        name: "Expert");
}

```

```

        migrationBuilder.DropTable(
            name: "Address");

        migrationBuilder.DropTable(
            name: "PageConfiguration");
    }
}
}

```

Kod z pliku ShopContextModelSnapshot.cs:

```

// <auto-generated />
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Infrastructure;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
using ProjektSklep.Data;

namespace ProjektSklep.Migrations
{
    [DbContext(typeof(ShopContext))]
    partial class ShopContextModelSnapshot : ModelSnapshot
    {
        protected override void BuildModel(ModelBuilder modelBuilder)
        {
#pragma warning disable 612, 618
            modelBuilder
                .HasAnnotation("ProductVersion", "3.1.10")
                .HasAnnotation("Relational:MaxIdentifierLength", 128)
                .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

            modelBuilder.Entity("ProjektSklep.Models.Address", b =>
            {
                b.Property<int>("AddressID")
                    .ValueGeneratedOnAdd()
                    .HasColumnType("int")
                    .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                b.Property<int?>("ApartmentNumber")
                    .HasColumnType("int");

                b.Property<string>("Country")
                    .HasColumnType("nvarchar(max)");

                b.Property<int>("CustomerID")
                    .HasColumnType("int");

                b.Property<int>("HouseNumber")
                    .HasColumnType("int");
            }
        }
    }
}

```

```

        b.Property<string>("PostCode")
            .HasColumnType("nvarchar(max)");

        b.Property<string>("Street")
            .HasColumnType("nvarchar(max)");

        b.Property<string>("Town")
            .HasColumnType("nvarchar(max)");

        b.HasKey("AddressID");

        b.ToTable("Address");
    });

modelBuilder.Entity("ProjektSklep.Models.Attachment", b =>
{
    b.Property<int>("AttachmentID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("Description")
        .IsRequired()
        .HasColumnType("nvarchar(max)");

    b.Property<string>("Path")
        .IsRequired()
        .HasColumnType("nvarchar(max)");

    b.Property<int>("ProductID")
        .HasColumnType("int");

    b.HasKey("AttachmentID");

    b.HasIndex("ProductID");

    b.ToTable("Attachment");
});

modelBuilder.Entity("ProjektSklep.Models.Category", b =>
{
    b.Property<int>("CategoryID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("Name")
        .HasColumnType("nvarchar(max)");

    b.Property<int?>("ParentCategoryID")
        .HasColumnType("int");

    b.Property<bool>("Visibility")
        .HasColumnType("bit");

```

```

        b.HasKey("CategoryID");

        b.HasIndex("ParentCategoryID");

        b.ToTable("Category");
    });

modelBuilder.Entity("ProjektSklep.Models.Customer", b =>
{
    b.Property<int>("CustomerID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<int>("AddressID")
        .HasColumnType("int");

    b.Property<bool>("AdminRights")
        .HasColumnType("bit");

    b.Property<string>("Email")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("FirstName")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("LastName")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("Login")
        .HasColumnType("nvarchar(max)");

    b.Property<int>("PageConfigurationID")
        .HasColumnType("int");

    b.Property<string>("Password")
        .HasColumnType("nvarchar(max)");

    b.HasKey("CustomerID");

    b.HasIndex("AddressID");

    b.HasIndex("PageConfigurationID");

    b.ToTable("Customer");
});

modelBuilder.Entity("ProjektSklep.Models.DiscountCode", b =>
{
    b.Property<int>("DiscountCodeID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("DiscountCode")

```

```

        .IsRequired()
        .HasColumnType("nvarchar(max)");

    b.Property<int>("Percent")
        .HasColumnType("int");

    b.HasKey("DiscountCodeID");

    b.ToTable("DiscountCode");
});

modelBuilder.Entity("ProjektSklep.Models.Expert", b =>
{
    b.Property<int>("ExpertID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("Email")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("FirstName")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("LastName")
        .HasColumnType("nvarchar(max)");

    b.HasKey("ExpertID");

    b.ToTable("Expert");
});

modelBuilder.Entity("ProjektSklep.Models.Order", b =>
{
    b.Property<int>("OrderID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<int>("CustomerID")
        .HasColumnType("int");

    b.Property<int>("OrderStatus")
        .HasColumnType("int");

    b.Property<int>("PaymentMethodID")
        .HasColumnType("int");

    b.Property<int>("ShippingMethodID")
        .HasColumnType("int");

    b.HasKey("OrderID");

    b.HasIndex("CustomerID");

```



```

        b.HasIndex("PaymentMethodID");

        b.HasIndex("ShippingMethodID");

        b.ToTable("Order");
    });

modelBuilder.Entity("ProjektSklep.Models.PageConfiguration", b =>
{
    b.Property<int>("PageConfigurationID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<int>("Currency")
        .HasColumnType("int");

    b.Property<int>("CustomerID")
        .HasColumnType("int");

    b.Property<int>("InterfaceSkin")
        .HasColumnType("int");

    b.Property<int>("Language")
        .HasColumnType("int");

    b.Property<int>("ProductsPerPage")
        .HasColumnType("int");

    b.Property<bool>("SendingNewsletter")
        .HasColumnType("bit");

    b.Property<bool>("ShowNetPrices")
        .HasColumnType("bit");

    b.HasKey("PageConfigurationID");

    b.ToTable("PageConfiguration");
});

modelBuilder.Entity("ProjektSklep.Models.PaymentMethod", b =>
{
    b.Property<int>("PaymentMethodID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("Name")
        .HasColumnType("nvarchar(max)");

    b.HasKey("PaymentMethodID");

    b.ToTable("PaymentMethod");
});

```

```

modelBuilder.Entity("ProjektSklep.Models.Product", b =>
{
    b.Property<int>("ProductID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<int>("Amount")
        .HasColumnType("int");

    b.Property<int>("CategoryID")
        .HasColumnType("int");

    b.Property<DateTime>("DateAdded")
        .HasColumnType("datetime2");

    b.Property<int>("ExpertID")
        .HasColumnType("int");

    b.Property<string>("Image")
        .HasColumnType("nvarchar(max)");

    b.Property<string>("Name")
        .HasColumnType("nvarchar(max)");

    b.Property<decimal>("Price")
        .HasColumnType("decimal(18,2)");

    b.Property<string>("ProductDescription")
        .HasColumnType("nvarchar(max)");

    b.Property<bool>("Promotion")
        .HasColumnType("bit");

    b.Property<int>("SoldProducts")
        .HasColumnType("int");

    b.Property<int>("VAT")
        .HasColumnType("int");

    b.Property<bool>("Visibility")
        .HasColumnType("bit");

    b.HasKey("ProductID");

    b.HasIndex("CategoryID");

    b.HasIndex("ExpertID");

    b.ToTable("Product");
});

modelBuilder.Entity("ProjektSklep.Models.ProductOrder", b =>
{
    b.Property<int>("ProductOrderID")
        .ValueGeneratedOnAdd()

```

```

        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

        b.Property<int>("OrderID")
        .HasColumnType("int");

        b.Property<int>("ProductID")
        .HasColumnType("int");

        b.HasKey("ProductOrderID");

        b.HasIndex("OrderID");

        b.HasIndex("ProductID");

        b.ToTable("ProductOrder");
    });

modelBuilder.Entity("ProjektSklep.Models.ShippingMethod", b =>
{
    b.Property<int>("ShippingMethodID")
        .ValueGeneratedOnAdd()
        .HasColumnType("int")
        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("Name")
        .HasColumnType("nvarchar(max)");

    b.HasKey("ShippingMethodID");

    b.ToTable("ShippingMethod");
});

modelBuilder.Entity("ProjektSklep.Models.Attachment", b =>
{
    b.HasOne("ProjektSklep.Models.Product", "Product")
        .WithMany("Attachments")
        .HasForeignKey("ProductID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();
});

modelBuilder.Entity("ProjektSklep.Models.Category", b =>
{
    b.HasOne("ProjektSklep.Models.Category", "Parent")
        .WithMany("Children")
        .HasForeignKey("ParentCategoryID");
});

modelBuilder.Entity("ProjektSklep.Models.Customer", b =>
{
    b.HasOne("ProjektSklep.Models.Address", "Address")
        .WithMany()
        .HasForeignKey("AddressID")
        .OnDelete(DeleteBehavior.Cascade)

```

```

        .IsRequired();

        b.HasOne("ProjektSklep.Models.PageConfiguration",
"PageConfiguration")
        .WithMany()
        .HasForeignKey("PageConfigurationID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();
    });

modelBuilder.Entity("ProjektSklep.Models.Order", b =>
{
    b.HasOne("ProjektSklep.Models.Customer", "Customer")
        .WithMany("Orders")
        .HasForeignKey("CustomerID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();

    b.HasOne("ProjektSklep.Models.PaymentMethod", "PaymentMethod")
        .WithMany("Orders")
        .HasForeignKey("PaymentMethodID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();

    b.HasOne("ProjektSklep.Models.ShippingMethod", "ShippingMethod")
        .WithMany("Orders")
        .HasForeignKey("ShippingMethodID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();
});

modelBuilder.Entity("ProjektSklep.Models.Product", b =>
{
    b.HasOne("ProjektSklep.Models.Category", "Category")
        .WithMany("Products")
        .HasForeignKey("CategoryID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();

    b.HasOne("ProjektSklep.Models.Expert", "Expert")
        .WithMany("Products")
        .HasForeignKey("ExpertID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();
});

modelBuilder.Entity("ProjektSklep.Models.ProductOrder", b =>
{
    b.HasOne("ProjektSklep.Models.Order", "Order")
        .WithMany("ProductOrders")
        .HasForeignKey("OrderID")
        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();

    b.HasOne("ProjektSklep.Models.Product", "Product")
        .WithMany("ProductOrders")
        .HasForeignKey("ProductID")

```

```

        .OnDelete(DeleteBehavior.Cascade)
        .IsRequired();
    });
#pragma warning restore 612, 618
    }
}
}

```

5. FUNKCJONALNOŚĆ APLIKACJI

- Obsługa wirtualnego koszyka, do którego można dodawać produkty dostępne w sklepie. W koszyku klient zobaczy podsumowanie w postaci listy dodanych produktów oraz ile sztuk danego produktu jest w koszyku. Dodatkowo zostanie wyliczona łączna cena za produkty.
- Możliwość wybrania preferowanej metody płatności, typu dostawy oraz wprowadzenia kodu rabatowego. Po zatwierdzeniu zamówienia zostaną wyświetlone wszystkie istotne jego szczegóły i zostanie ono dodane do bazy danych.
- Możliwość przeglądania produktów (w tym również wg. kategorii) przez klienta oraz możliwość edycji i rozszerzania bazy danych poprzez dodawanie kolejnych rekordów w tabelach przez administratora.

Działanie funkcjonalności jest zaprezentowane na zrzutach ekranu w kolejnym akapicie, który dotyczy interfejsu użytkownika.

6. INTERFEJS UŻYTKOWNIKA

Ekran rejestracji nowego użytkownika:

Ekran logowania :

Log in - ProjektSklep

https://localhost:44368/Account/Login

Search Szukaj

Register Login Koszyk Ustawienia

Log in

Use a local account to log in.

Email
mejljk@gmail.com

Password

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

Use another service to log in.

There are no external authentication services configured. See [this article](#) for details on setting up this ASP.NET application to support logging in via external services.

© 2020 - ProjektSklep - [Privacy](#)

Strona główna zalogowanego klienta (niezalogowany klient widzi to samo co zalogowany, jednakże aby złożyć zamówienie musi wcześniej się zalogować):

Strona główna - ProjektSklep

https://localhost:44368

Search Szukaj

Hello mejljk@gmail.com! Logout Koszyk Ustawienia

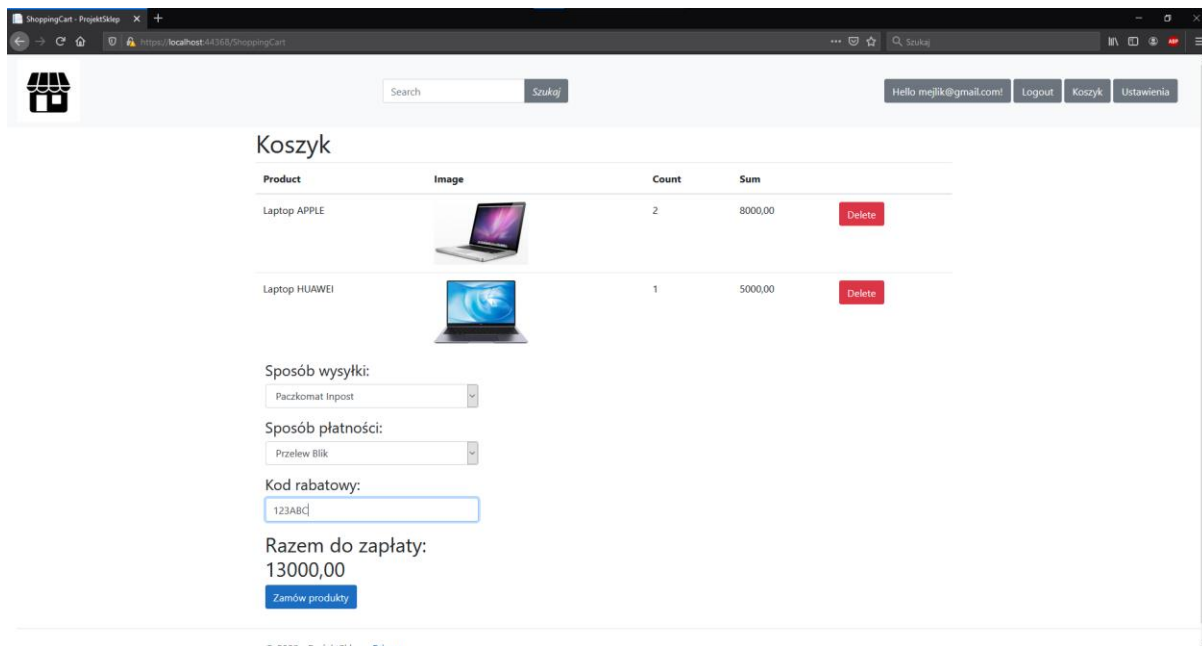
Kategorie

- Komputery
- Laptopy
- Smartfony
- Elektronika

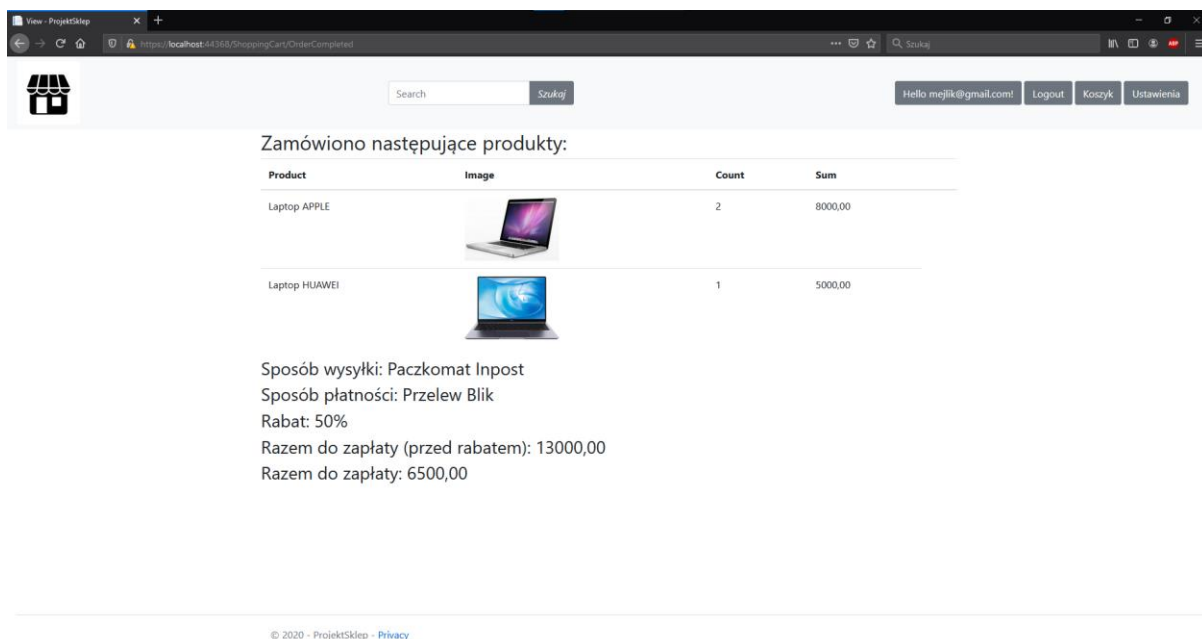
	Laptop APPLE Add To Cart Dobry laptop Kategoria: Laptopy	4000.00 PLN
	Laptop HUAWEI Add To Cart Dobry laptop Kategoria: Laptopy	5000.00 PLN
	Smartfon HUAWEI P30 Add To Cart Dobry smartfon Kategoria: Smartfony	2999.00 PLN
	Laptop LENOVO Add To Cart Dobry laptop Kategoria: Laptopy	4300.00 PLN

© 2020 - ProjektSklep - [Privacy](#)

Koszyk zalogowanego klienta z dodanymi już produktami (niezalogowany użytkownik nie będzie widział przycisku służącego do złożenia zamówienia):



Widok po dokonaniu zamówienia przez użytkownika:



Widok zamówień z perspektywy administratora (zamówienie o ID równym 8 zawiera wszystkie istotne informacje ze złożonego przed chwilą zamówienia.):

The screenshot shows the 'Index' page of the ProjektSklep administrator. The page has a sidebar with a 'Baza danych:' menu containing links to DiscountCodes, Addresses, Customers, PageConfigurations, Orders, ShippingMethods, PaymentMethods, Products, Categories, Attachments, Experts, and ProductOrders. The main content area is titled 'Index' and includes a 'Create New' link. Below this is a table of orders:

OrderID	OrderStatus	Customer	ShippingMethod	PaymentMethod	
1	OnTheWay	bartlomiejuminski1999@gmail.com	Kurier UPS	Przelew Tradycyjny	Edit Details Delete
2	Delivered	kacpersiegienczuk@gmail.com	Kurier DHL	Karta	Edit Details Delete
3	Delivered	michalkozikowski@gmail.com	Poczta	Szybki Przelew	Edit Details Delete
4	OnTheWay	jakubkozowski@gmail.com	Poczta	Przelew Tradycyjny	Edit Details Delete
5	Preparing	klientklientowski@gmail.com	Paczkomat Inpost	Przelew Blik	Edit Details Delete
8	Preparing	mejlik@gmail.com	Paczkomat Inpost	Przelew Blik	Edit Details Delete

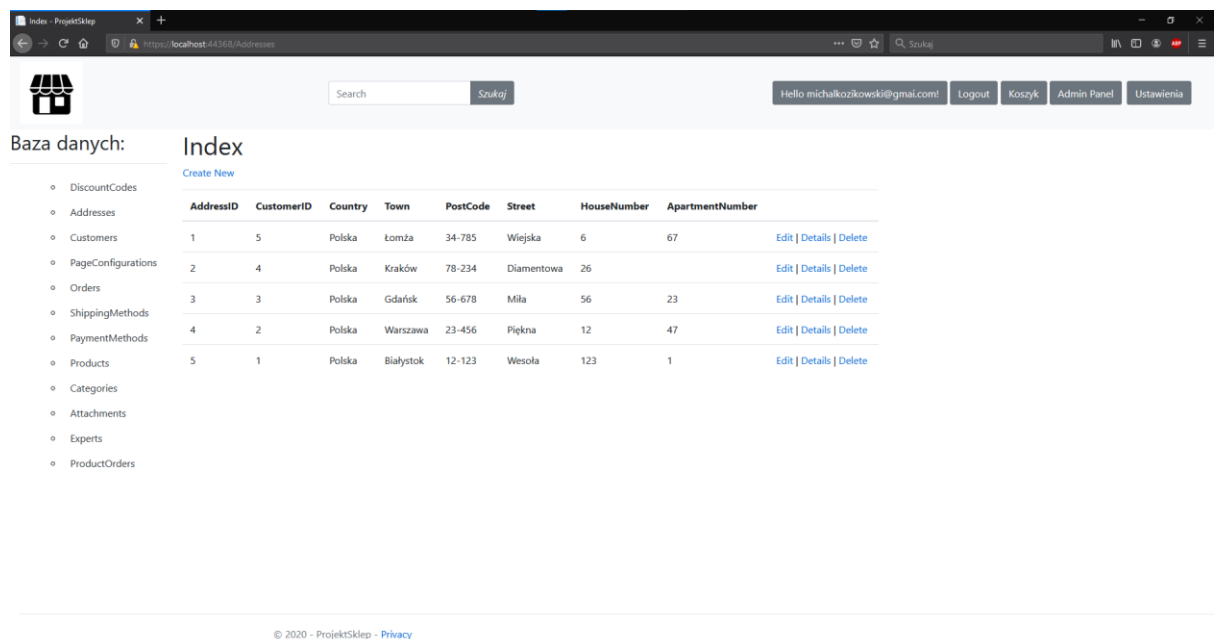
At the bottom of the page, there is a footer: © 2020 - ProjektSklep - [Privacy](#).

Panel administratora (Administrator również jest klientem sklepu i może robić zakupy, jednakże dodatkowo posiada panel administracyjny, w którym może edytować i przeglądać bazę danych sklepu):

The screenshot shows the 'Admin Panel' page of the ProjektSklep administrator. The page has a sidebar with a 'Baza danych:' menu containing links to DiscountCodes, Addresses, Customers, PageConfigurations, Orders, ShippingMethods, PaymentMethods, Products, Categories, Attachments, Experts, and ProductOrders. The main content area is titled 'Witamy w panelu admina'. The page layout is similar to the 'Index' page, with a search bar and user navigation links at the top.

At the bottom of the page, there is a footer: © 2020 - ProjektSklep - [Privacy](#).

Przykładowy podgląd na jedną z tabel (w tym przypadku na Adresy):



The screenshot shows the 'Index' page of the ProjektSklep application. The page has a header with a search bar, a 'Szukaj' button, and user information. A sidebar on the left lists various database tables. The main content area displays a table of addresses with columns for AddressID, CustomerID, Country, Town, PostCode, Street, HouseNumber, and ApartmentNumber. Each row includes links for 'Edit', 'Details', and 'Delete'.

Baza danych:

- DiscountCodes
- Addresses
- Customers
- PageConfigurations
- Orders
- ShippingMethods
- PaymentMethods
- Products
- Categories
- Attachments
- Experts
- ProductOrders

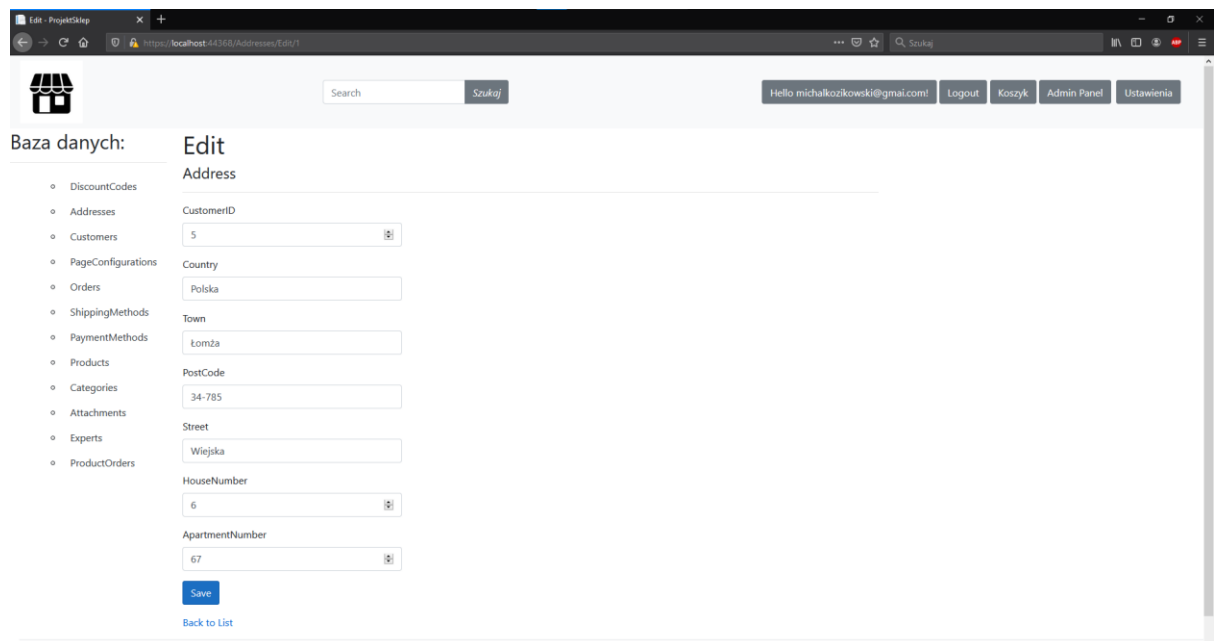
Index

[Create New](#)

AddressID	CustomerID	Country	Town	PostCode	Street	HouseNumber	ApartmentNumber	
1	5	Polska	Łomża	34-785	Wiejska	6	67	Edit Details Delete
2	4	Polska	Kraków	78-234	Diaamentowa	26		Edit Details Delete
3	3	Polska	Gdańsk	56-678	Miła	56	23	Edit Details Delete
4	2	Polska	Warszawa	23-456	Piękna	12	47	Edit Details Delete
5	1	Polska	Białystok	12-123	Wesoła	123	1	Edit Details Delete

© 2020 - ProjektSklep - [Privacy](#)

Edycja rekordu:



The screenshot shows the 'Edit' page of the ProjektSklep application. The page has a header with a search bar, a 'Szukaj' button, and user information. A sidebar on the left lists various database tables. The main content area displays a form for editing an address record, with fields for CustomerID, Country, Town, PostCode, Street, HouseNumber, and ApartmentNumber. A 'Save' button and a 'Back to List' link are at the bottom.

Baza danych:

- DiscountCodes
- Addresses
- Customers
- PageConfigurations
- Orders
- ShippingMethods
- PaymentMethods
- Products
- Categories
- Attachments
- Experts
- ProductOrders

Edit

Address

CustomerID:

Country:

Town:

PostCode:

Street:

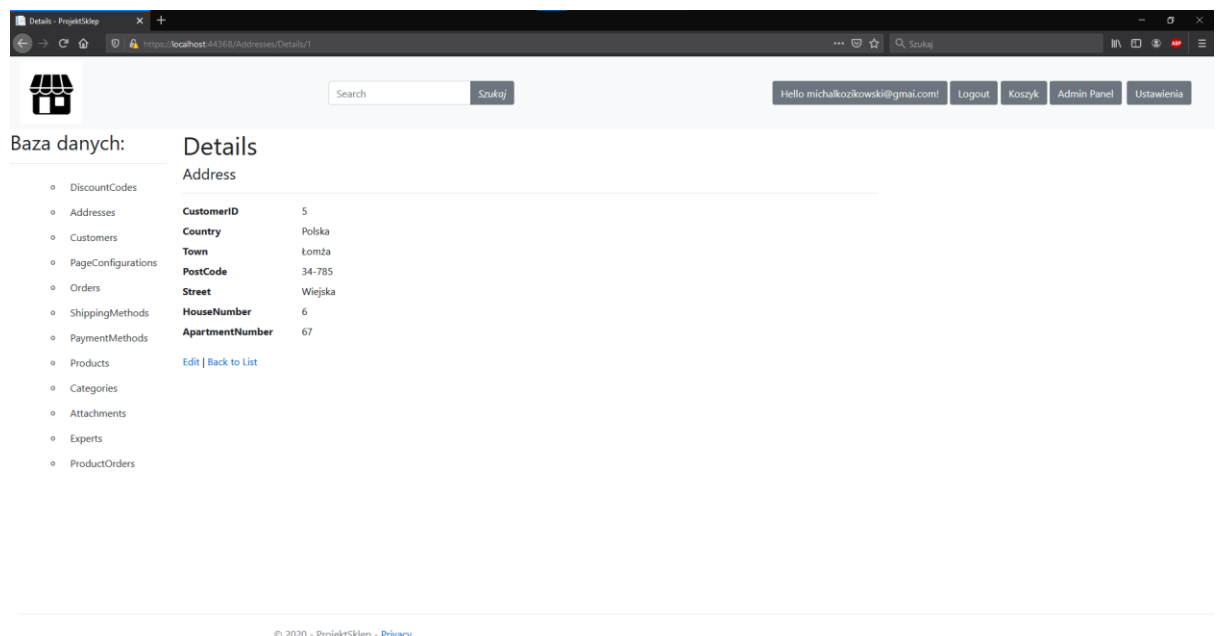
HouseNumber:

ApartmentNumber:

[Save](#)

[Back to List](#)

Szczegóły rekordu:



7. PODSUMOWANIE

Aplikacja jest oparta o technologię *.NET Core MVC*, oraz posiada podstawowe funkcjonalności niezbędne do prowadzenia internetowego sklepu. *Entity Framework* posłużył nam do mapowania bazy danych oraz stworzenia stron odpowiedzialnych za operacje CRUD.

8. ETAPY TWORZENIA APLIKACJI

- Zaprojektowano diagram klas UML,
- Zaprojektowano diagramy ER i relacji za pomocą programu *sqldeveloper*,
- Stworzono projekt aplikacji internetowej w *VisualStudio* w technologii *.NET Core MVC*
- Dodano modele klas służące do wygenerowania bazy danych za pomocą narzędzia *EntityFramework ORM*
- Wygenerowano bazę danych poprzez wykorzystanie mechanizmu migracji
- Zaimplementowanie podstawowych funkcjonalności aplikacji takich jak logowanie, panel administratora, możliwość wykonania zamówienia, możliwość przeglądania produktów dostępnych w sklepie.

9. DOKŁADNY OPIS TABEL BAZY DANYCH

- Customer - tabela przechowuje dane użytkowników systemu (klientów i administratorów)
 - Każdy rekord z tabeli Customer przechowuje ID, imię, nazwisko, login, email, hasło, zmienną boolowską sprawdzającą czy dany użytkownik ma prawa administracyjne, ID adresu oraz ID ustawionej konfiguracji strony.
- Order - tabela przechowuje pojedyncze zamówienie
 - Każdy rekord z tabeli Order przechowuje ID zamówienia, status zamówienia, ID zamawiającego, ID metody płatności oraz ID sposobu dostawy.
- Product - tabela przechowuje dane produktów dostępnych do zamówienia
 - Każdy rekord z tabeli Product przechowuje ID produktu, nazwę, opis, ścieżkę do obrazu, datę dodania, zmienną boolowską informującą czy produkt jest objęty promocją, podatek VAT, cenę, ilość, zmienną boolowską informującą czy produkt jest widoczny dla klientów, ilość sprzedanych sztuk, ID eksperta odpowiedzialnego za produkt oraz ID kategorii produktu.
- ProductOrder - tabela pośrednia pomiędzy produktami i zamówieniami (wyeliminowanie relacji wiele do wielu)
 - Każdy rekord z tabeli ProductOrder przechowuje ID pary [Product, Order], ID zamówienia oraz ID produktu.
- Category - tabela z kategoriami produktów
 - Każdy rekord z tabeli Category przechowuje ID kategorii, nazwę, zmienną boolowską informującą czy kategoria jest widzialna dla klientów oraz czy kategoria posiada nadrzędną kategorię (ParentCategoryID).
- Attachment - tabela ze ścieżkami do załączników produktów
 - Każdy rekord z tabeli Attachment przechowuje ID załącznika, ścieżkę do załącznika, opis oraz ID produktu.
- Expert - tabela z danymi ekspertów zajmujących się konkretnymi produktami
 - Każdy rekord z tabeli Expert przechowuje ID eksperta, imię, nazwisko oraz jego email.
- PageConfiguration - konfiguracja strony użytkownika
 - Każdy rekord z tabeli PageConfiguration przechowuje ID konfiguracji strony, zmienną boolowską informującą czy klient chce otrzymywać newsletter, zmienną boolowską informującą czy aplikacja ma wyświetlać ceny netto, ilość produktów wyświetlanych się na stronie, jakiej skórki interfejsu chce używać klient, język, w jakiej walucie wyświetlać mają się ceny oraz ID klienta.
- ShippingMethod - tabela z metodami dostawy dostępnymi w sklepie
 - Każdy rekord z tabeli ShippingMethod przechowuje ID sposobu wysyłki oraz nazwę.
- PaymentMethod - tabela z płatnościami dostępnymi w sklepie
 - Każdy rekord z tabeli PaymentMethod przechowuje ID metody płatności oraz nazwę.

- Address - tabela przechowująca adresy do przesyłek
 - Każdy rekord z tabeli Address przechowuje ID adresu, kraj, miasto, kod pocztowy, ulica, numer domu, numer mieszkania oraz ID klienta.
- DiscountCode - tabela z kodami zniżkowymi
 - Każdy rekord z tabeli DiscountCode przechowuje ID kodu zniżkowego, treść kodu zniżkowego oraz informację o ile procent cena zamówienia będzie niższa.

10. DODATEK A: SKRYPTY TWORZĄCE OBIEKTY BAZ DANYCH

Poniżej zamieszczono kod skryptu służącego do wygenerowania obiektów bazy danych w naszym projekcie.

```
-- Generated by Oracle SQL Developer Data Modeler 20.3.0.283.0710
-- at:          2020-12-04 23:35:07 CET
-- site:        Oracle Database 11g
-- type:        Oracle Database 11g

-- predefined type, no DDL - MDSYS.SDO_GEOMETRY

-- predefined type, no DDL - XMLTYPE

CREATE TABLE address (
  addressid          INTEGER NOT NULL,
  country            NVARCHAR2(2000) NOT NULL,
  town               NVARCHAR2(2000) NOT NULL,
  postcode           NVARCHAR2(2000) NOT NULL,
  street             NVARCHAR2(2000) NOT NULL,
  housenumber        INTEGER NOT NULL,
  apartmentnumber    INTEGER,
  customer_customerid INTEGER NOT NULL
);

CREATE UNIQUE INDEX address__idx ON
  address (
    customer_customerid
  ASC );

ALTER TABLE address ADD CONSTRAINT address_pk PRIMARY KEY ( addressid );

CREATE TABLE attachment (
  attachmentid       INTEGER NOT NULL,
  path               NVARCHAR2(2000) NOT NULL,
```

```

        description      NVARCHAR2(2000) NOT NULL,
        product_productid INTEGER NOT NULL
    );

```

```

ALTER TABLE attachment ADD CONSTRAINT attachment_pk PRIMARY KEY (
attachmentid );

```

```

CREATE TABLE category (
    categoryid      INTEGER NOT NULL,
    name            NVARCHAR2(2000) NOT NULL,
    visibility       CHAR(1) NOT NULL,
    category_categoryid INTEGER NOT NULL
);

```

```

ALTER TABLE category ADD CONSTRAINT category_pk PRIMARY KEY ( categoryid );

```

```

CREATE TABLE customer (
    customerid      INTEGER NOT NULL,
    firstname       NVARCHAR2(2000) NOT NULL,
    lastname        NVARCHAR2(2000) NOT NULL,
    login           NVARCHAR2(2000) NOT NULL,
    password        NVARCHAR2(2000) NOT NULL,
    email           NVARCHAR2(2000) NOT NULL,
    adminrights     CHAR(1) NOT NULL,
    address_addressid INTEGER NOT NULL,
    pageconfigurationid INTEGER NOT NULL
);

```

```

CREATE UNIQUE INDEX customer__idx ON
    customer (
        address_addressid
    ASC );

```

```

CREATE UNIQUE INDEX customer__idxv1 ON
    customer (
        pageconfigurationid
    ASC );

```

```

ALTER TABLE customer ADD CONSTRAINT customer_pk PRIMARY KEY ( customerid );

```

```

CREATE TABLE discountcode (
    discountcodeid  INTEGER NOT NULL,
    discountcode    NVARCHAR2(2000) NOT NULL,
    percent         INTEGER NOT NULL
);

```

```
ALTER TABLE discoundcode ADD CONSTRAINT discoundcode_pk PRIMARY KEY (
discountcodeid );
```

```
CREATE TABLE expert (
    expertid    INTEGER NOT NULL,
    firstname   NVARCHAR2(2000) NOT NULL,
    lastname    NVARCHAR2(2000) NOT NULL,
    email       NVARCHAR2(2000) NOT NULL
);
```

```
ALTER TABLE expert ADD CONSTRAINT expert_pk PRIMARY KEY ( expertid );
```

```
CREATE TABLE "Order" (
    orderid                INTEGER NOT NULL,
    orderstatus            INTEGER NOT NULL,
    customer_customerid    INTEGER NOT NULL,
    paymentmethod_paymentmethodid INTEGER NOT NULL,
    shippingmethodid       INTEGER NOT NULL
);
```

```
ALTER TABLE "Order" ADD CONSTRAINT order_pk PRIMARY KEY ( orderid );
```

```
CREATE TABLE pageconfiguration (
    pageconfigurationid  INTEGER NOT NULL,
    sendingnewsletter    CHAR(1) NOT NULL,
    shownetprices        CHAR(1) NOT NULL,
    productsperpage      INTEGER NOT NULL,
    interfaceskin        INTEGER NOT NULL,
    language             INTEGER NOT NULL,
    currency             INTEGER NOT NULL,
    customer_customerid  INTEGER NOT NULL
);
```

```
CREATE UNIQUE INDEX pageconfiguration__idx ON
pageconfiguration (
    customer_customerid
ASC );
```

```
ALTER TABLE pageconfiguration ADD CONSTRAINT pageconfiguration_pk PRIMARY
KEY ( pageconfigurationid );
```

```
CREATE TABLE paymentmethod (
    paymentmethodid  INTEGER NOT NULL,
    name            NVARCHAR2(2000) NOT NULL
);
```

```
ALTER TABLE paymentmethod ADD CONSTRAINT paymentmethod_pk PRIMARY KEY (
paymentmethodid );
```

```
CREATE TABLE product (
    productid          INTEGER NOT NULL,
    name               NVARCHAR2(2000) NOT NULL,
    productdescription NVARCHAR2(2000) NOT NULL,
    image              NVARCHAR2(2000) NOT NULL,
    dateadded          TIMESTAMP NOT NULL,
    promotion          CHAR(1) NOT NULL,
    vat                INTEGER NOT NULL,
    price              NUMBER(18) NOT NULL,
    amount             INTEGER NOT NULL,
    visibility         CHAR(1) NOT NULL,
    soldproducts       INTEGER NOT NULL,
    expert_expertid    INTEGER NOT NULL,
    category_categoryid INTEGER NOT NULL
);
```

```
ALTER TABLE product ADD CONSTRAINT product_pk PRIMARY KEY ( productid );
```

```
CREATE TABLE productorder (
    productorderid    INTEGER NOT NULL,
    order_orderid     INTEGER NOT NULL,
    product_productid INTEGER NOT NULL
);
```

```
ALTER TABLE productorder ADD CONSTRAINT productorder_pk PRIMARY KEY (
productorderid );
```

```
CREATE TABLE shippingmethod (
    shippingmethodid INTEGER NOT NULL,
    name              NVARCHAR2(2000) NOT NULL
);
```

```
ALTER TABLE shippingmethod ADD CONSTRAINT shippingmethod_pk PRIMARY KEY (
shippingmethodid );
```

```
ALTER TABLE address
    ADD CONSTRAINT address_customer_fk FOREIGN KEY ( customer_customerid )
    REFERENCES customer ( customerid );
```

```
ALTER TABLE attachment
    ADD CONSTRAINT attachment_product_fk FOREIGN KEY ( product_productid )
    REFERENCES product ( productid );
```

```
ALTER TABLE category
```

```

        ADD CONSTRAINT category_category_fk FOREIGN KEY ( category_categoryid )
            REFERENCES category ( categoryid );

ALTER TABLE customer
    ADD CONSTRAINT customer_address_fk FOREIGN KEY ( address_addressid )
        REFERENCES address ( addressid );

ALTER TABLE customer
    ADD CONSTRAINT customer_pageconfiguration_fk FOREIGN KEY (
pageconfigurationid )
        REFERENCES pageconfiguration ( pageconfigurationid );

ALTER TABLE "Order"
    ADD CONSTRAINT order_customer_fk FOREIGN KEY ( customer_customerid )
        REFERENCES customer ( customerid );

ALTER TABLE "Order"
    ADD CONSTRAINT order_paymentmethod_fk FOREIGN KEY (
paymentmethod_paymentmethodid )
        REFERENCES paymentmethod ( paymentmethodid );

ALTER TABLE "Order"
    ADD CONSTRAINT order_shippingmethod_fk FOREIGN KEY ( shippingmethodid )
        REFERENCES shippingmethod ( shippingmethodid );

ALTER TABLE pageconfiguration
    ADD CONSTRAINT pageconfiguration_customer_fk FOREIGN KEY (
customer_customerid )
        REFERENCES customer ( customerid );

ALTER TABLE product
    ADD CONSTRAINT product_category_fk FOREIGN KEY ( category_categoryid )
        REFERENCES category ( categoryid );

ALTER TABLE product
    ADD CONSTRAINT product_expert_fk FOREIGN KEY ( expert_expertid )
        REFERENCES expert ( expertid );

ALTER TABLE productorder
    ADD CONSTRAINT productorder_order_fk FOREIGN KEY ( order_orderid )
        REFERENCES "Order" ( orderid );

ALTER TABLE productorder
    ADD CONSTRAINT productorder_product_fk FOREIGN KEY ( product_productid )
        REFERENCES product ( productid );

```


-- Oracle SQL Developer Data Modeler Summary Report:

```
--  
-- CREATE TABLE 12  
-- CREATE INDEX 4  
-- ALTER TABLE 25  
-- CREATE VIEW 0  
-- ALTER VIEW 0  
-- CREATE PACKAGE 0  
-- CREATE PACKAGE BODY 0  
-- CREATE PROCEDURE 0  
-- CREATE FUNCTION 0  
-- CREATE TRIGGER 0  
-- ALTER TRIGGER 0  
-- CREATE COLLECTION TYPE 0  
-- CREATE STRUCTURED TYPE 0  
-- CREATE STRUCTURED TYPE BODY 0  
-- CREATE CLUSTER 0  
-- CREATE CONTEXT 0  
-- CREATE DATABASE 0  
-- CREATE DIMENSION 0  
-- CREATE DIRECTORY 0  
-- CREATE DISK GROUP 0  
-- CREATE ROLE 0  
-- CREATE ROLLBACK SEGMENT 0  
-- CREATE SEQUENCE 0  
-- CREATE MATERIALIZED VIEW 0  
-- CREATE MATERIALIZED VIEW LOG 0  
-- CREATE SYNONYM 0  
-- CREATE TABLESPACE 0  
-- CREATE USER 0  
--  
-- DROP TABLESPACE 0  
-- DROP DATABASE 0  
--  
-- REDACTION POLICY 0  
--  
-- ORDS DROP SCHEMA 0  
-- ORDS ENABLE SCHEMA 0  
-- ORDS ENABLE OBJECT 0  
--  
-- ERRORS 0  
-- WARNINGS 0
```