
Politechnika Śląska
Wydział Inżynierii Materiałowej
i Metalurgii

Temat:

# Projekt: SmartShop
# Dokumentacja

Komentarze:

Data:

Podpis:

Aleksandra Miękina,
Tomasz Szostak,
Damian Kotulski,
Krzysztof Kurkiewicz

IPP-30PP

# Spis treści

# 1. Założenia projektu

Założeniem projektu było wykonać nowoczesną aplikację, która potrafiłaby zastąpić kasę fiskalną dla sklepu. Aplikacja ma mieć pełną fukncjonalność pozwalającą na obsłużenie klienta przez sprzedawcę, jednocześnie mając prosty i intuuicyjny wygląd, aby mogła być obsłużona przez niewykwalifikowany w sposób techniczny personel. W tych funkcjonalnościach powinno mieścić się minimum:

a) możliwość logowania się na własne, spersonalizowane konto
b) możliwość wybrania produktu z listy wszystkich produktów
c) możliwość wpisania kodu danego produktu(imitacja skanowania kodów kreskowych)
d) możliwość wpisywania i edycji ilości sztuk zakupionego przedmiotu
e) możliwość usuwania przedmiotów z listy zakupionych
f) możliwość dokonywania zwrotów
g) możliwość drukawania paragonów
h) możliwość drukowania raportów w wybranym odstępie czasowym
i) możliwość nadawania zniżek całościowych oraz pojedyńczych, złotówkowych oraz procentowych
j) możliwość zdjęcia vat z produktu
k) możliwość przeglądania wszystkich transakcji

# 2. Wykorzystane technologie

## a. Spis i opis technologii

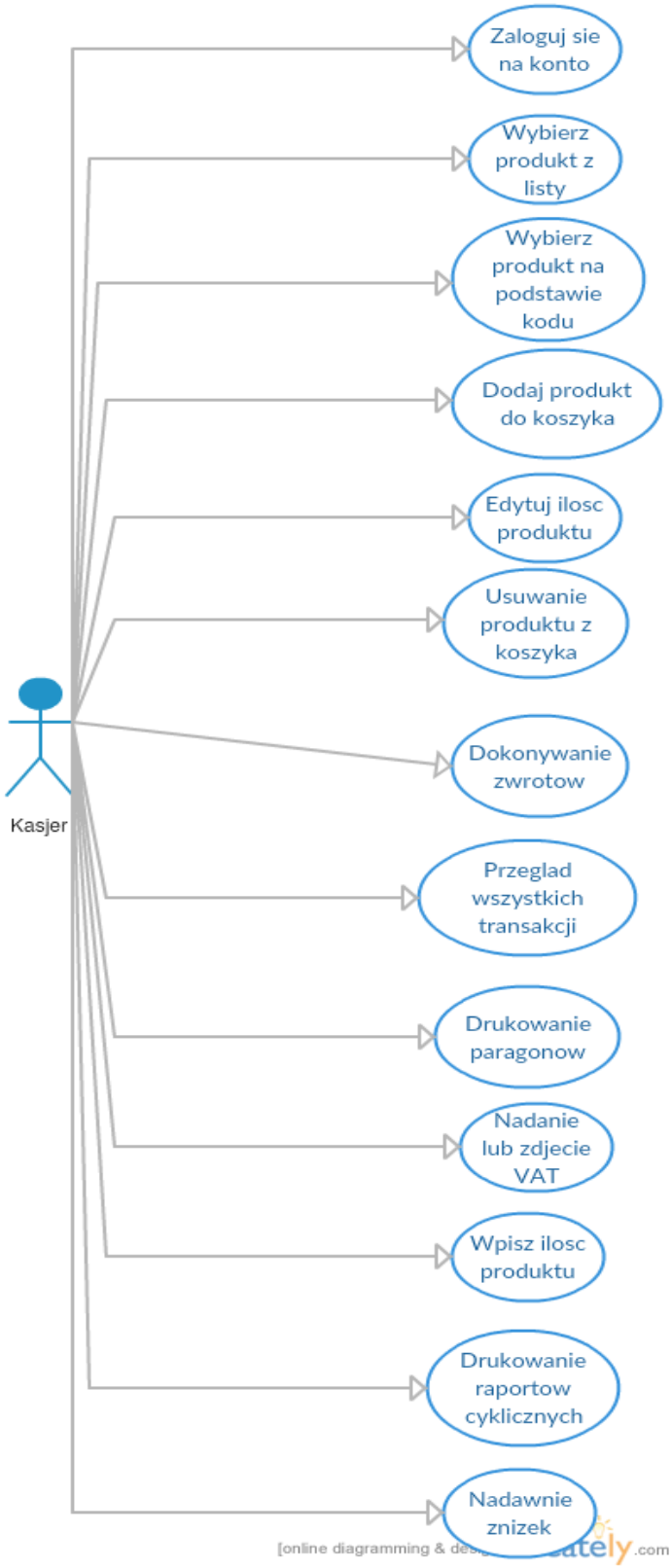
a) Baza danych – MariaDB postawiona na zewnętrznych serwerze
b) Środowisko do połączenią się z bazą - HeidiSQL
c) Aplikacja kliencka – Windows Presentation Foundation w języku C#
d) Webservice – w architekturze REST, napisany w języku C# w aplikacji typu ASP.NET, użyty ORM to Entity Framework
e) Środowisko – Microsoft Visual Studio 2015

## b. Wtyczki wspomagające

a) MySQL.Data oraz MySql.Data.Entity.EF6 – wtyczki pozwalające na połączenie technologii .NET z bazą napisaną na silniku MySQL
b) Newtonsoft.Json – wtyczka pozwalająca na serializacje i deserializacje informacji w formacie Json
c) NUnit 3 – wtyczka pozwalające na pisanie testów jednostkowych
d) Moq – wtyczka pozwalająca na mokowanie zewnętrznych zależności do testów
e) RestSharp – wtyczka ułatwiająca połączenie z webservicem o architekturze REST
f) Owin – wtyczka do wymuszenia autoryzacji na bazie tokenu w kontrolerach
g) CodeMaid – wtyczka pozwalająca zachować czystość i zwięzłość w kodzie
h) ReSharper – wtyczka rozszerzająca możliwości Visual Studio
i) Spire.PDF – do drukowania paragonów oraz raportów w postaci PDF

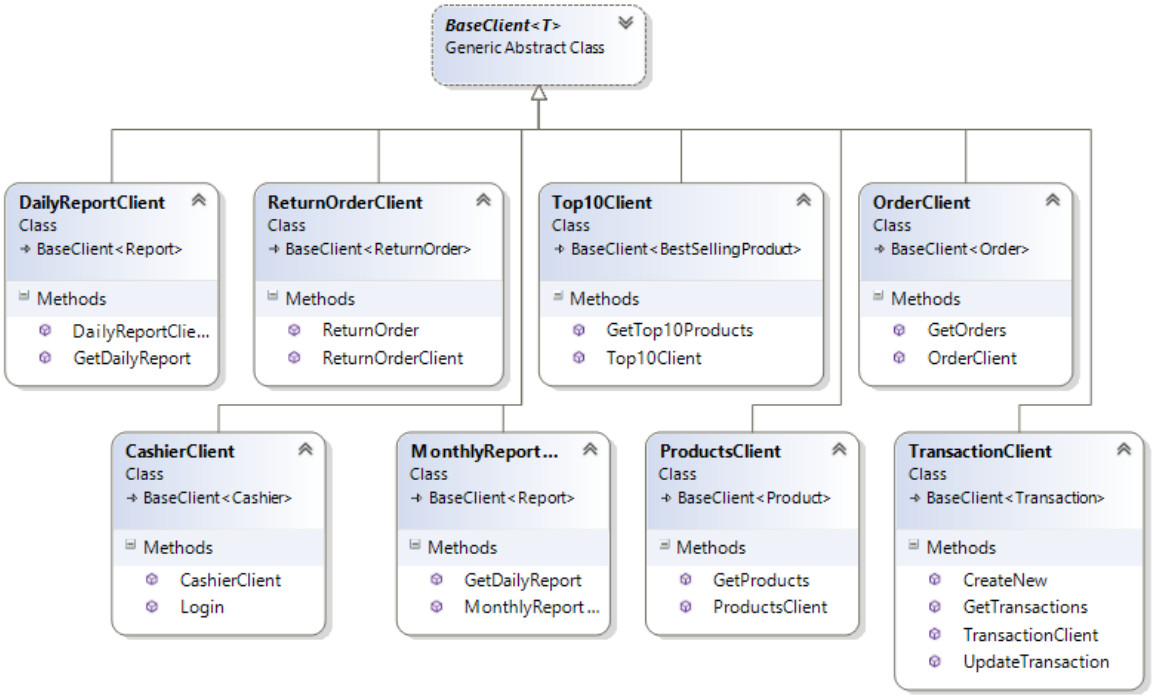
# 3. Diagramy

## a. Diagram przypadków użycia

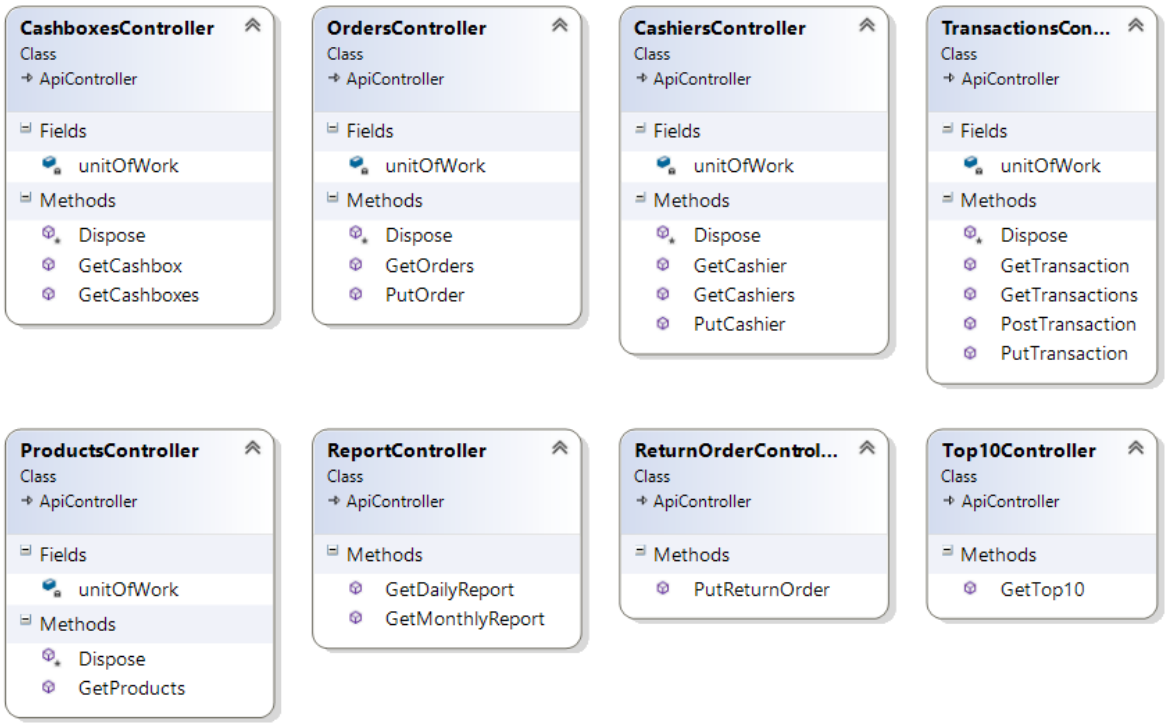


## b. Diagramy klas programu

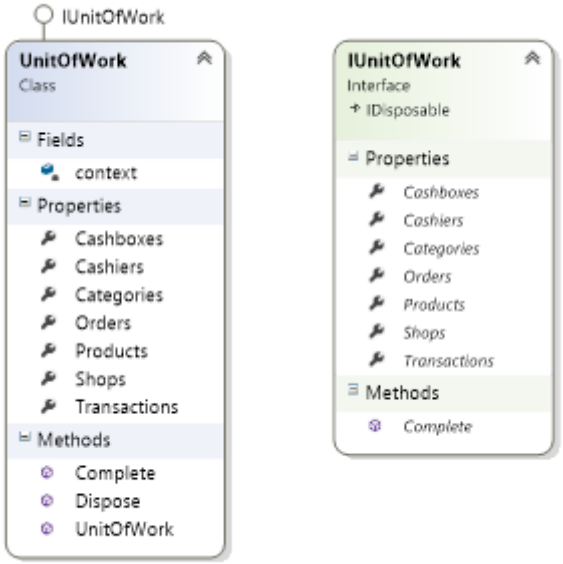
❖ Klasy odpowiedzialne za komunikację z WebService'm



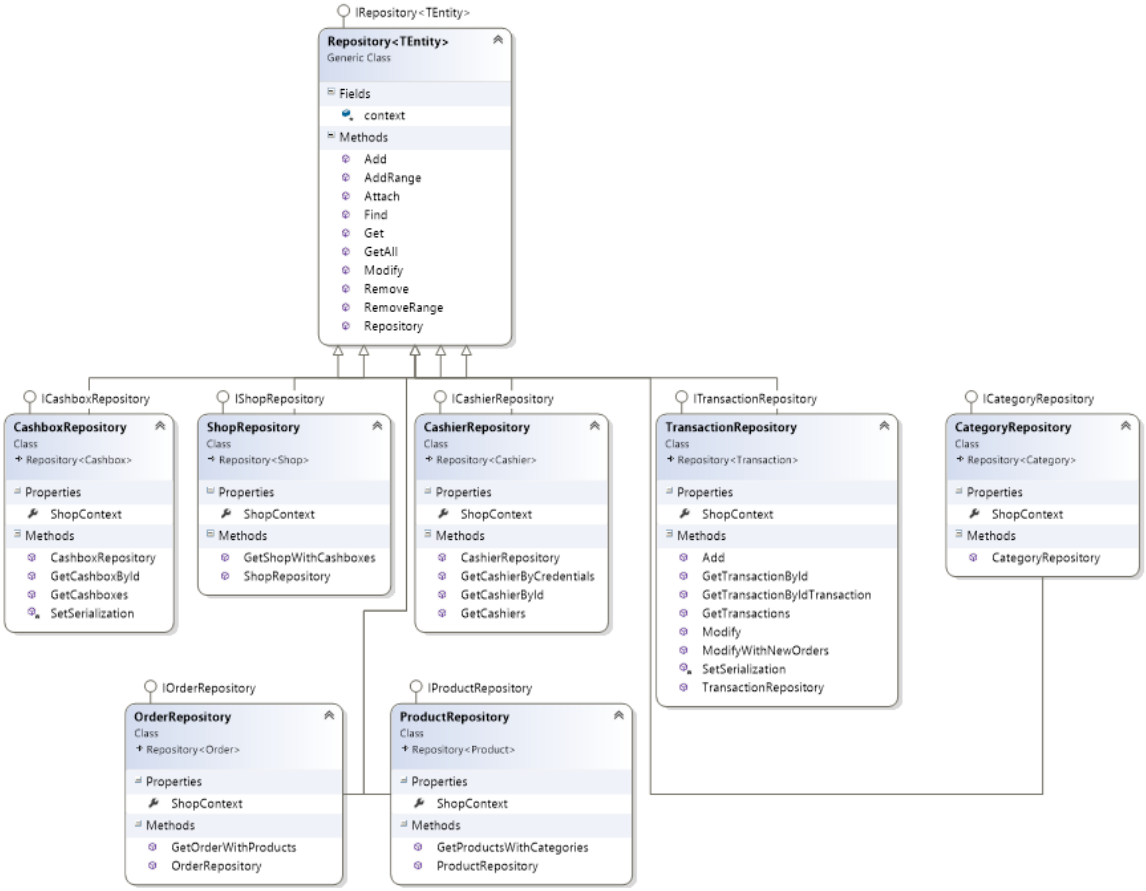

❖ Klasy kontrolerów



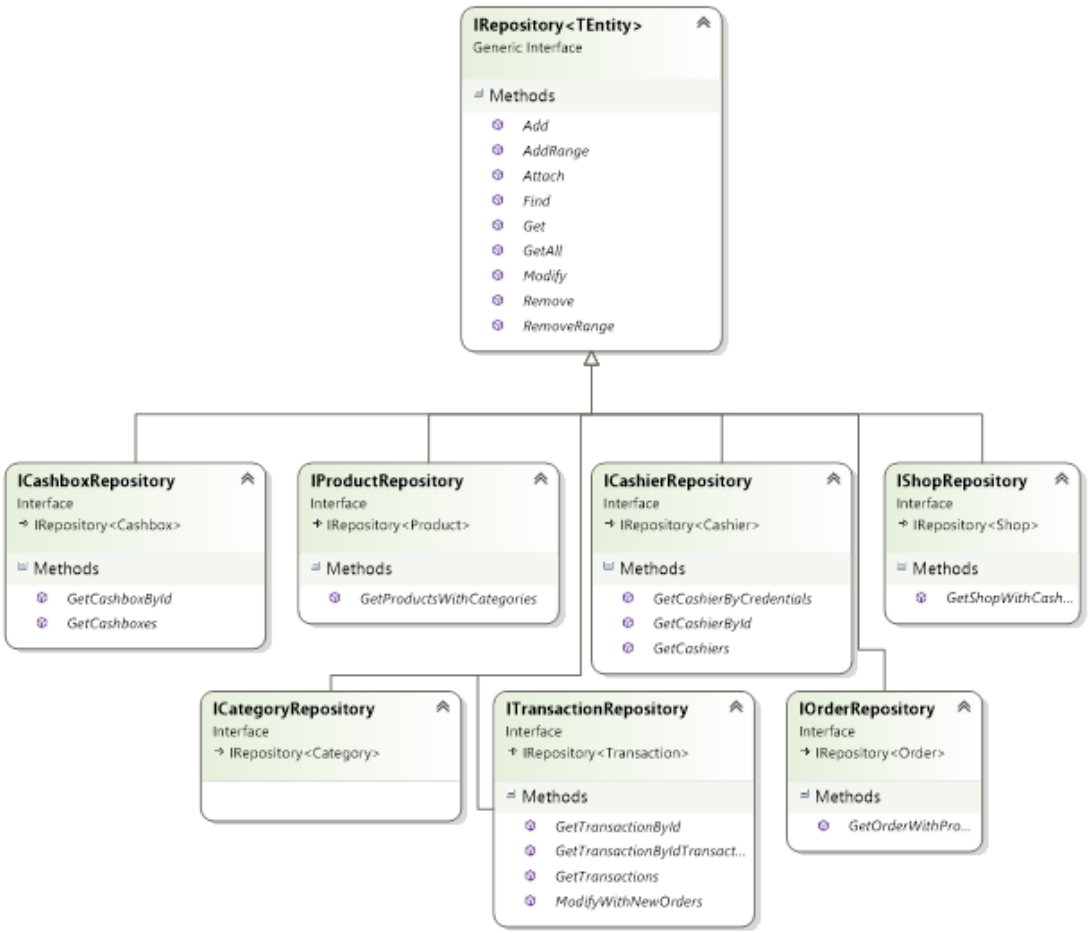
❖ Klasy wzorca Unit Of Work



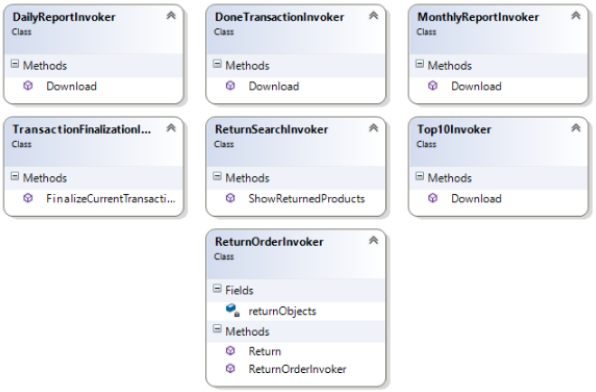

❖ Klasy wzorca Repozytorium



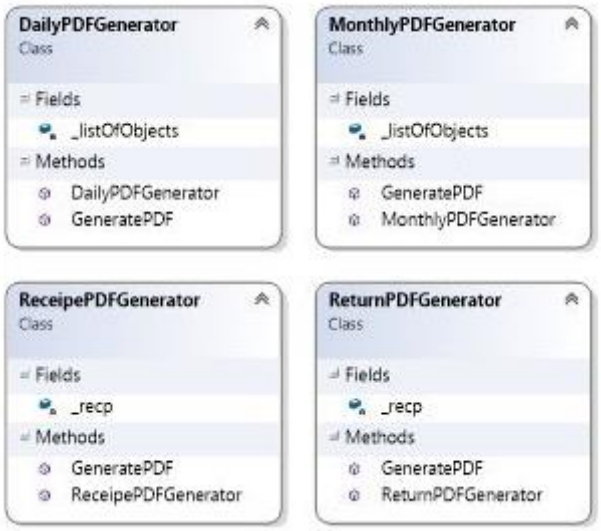
❖ Interfejsy wzorca Repozytorium



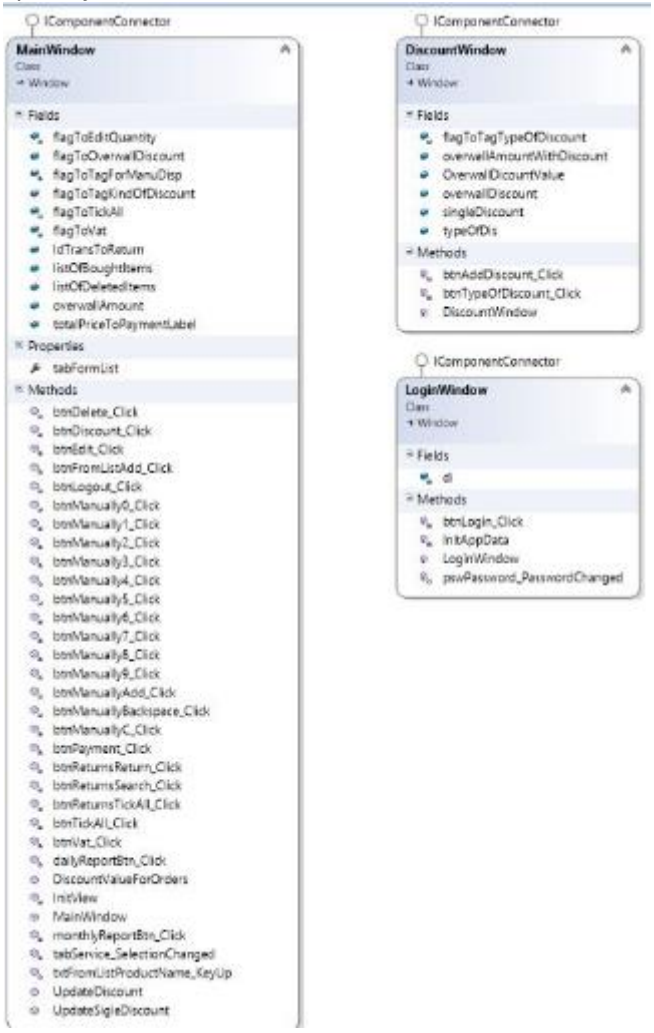

❖ Klasy odpowiadające za asynchroniczne przetwarzanie długotrwałych operacji



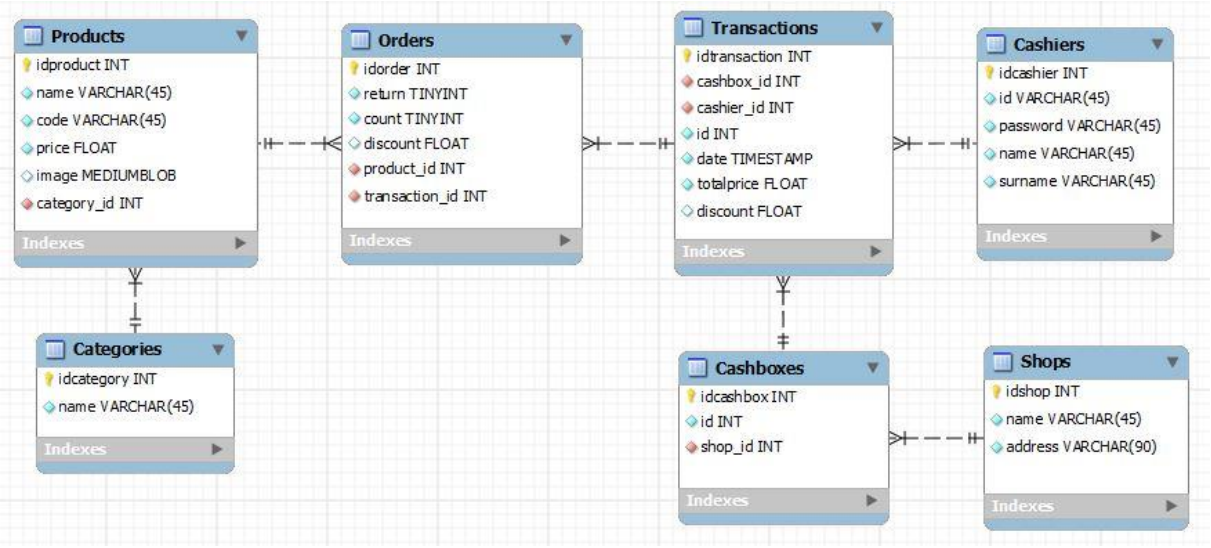
❖ Klasy odpowiedzialne za generowanie plików pdf




❖ Klasy widoków aplikacji



### c. Diagram encji bazy danych




## 4. Baza Danych

### a. Opis

Jako środowisko bazodanowe wybraliśmy MariaDB. Głównymi czynnikami wypływającymi na nasz wybór było to, że jest to darmowe środowisko, bazuje na MySQL z którym pracowaliśmy już wcześniej. Dodatkowymi atutami było to, że ów środowisko jest aktualnie jednym z najlepiej ocenianych środowisko bazodanowych, posiada bardzo dobre wsparcie społeczności, która je tworzy.

### b. Opis procedur

a) procedura logCashier

```
BEGIN
IF EXISTS (select Id from Cashiers where Id = Id1) THEN
BEGIN
select Password into @dbPassword from Cashiers where Id = Id1; -- and Password = AES_ENCRYPT(Password1,SHA2(Id1,512));
IF (@dbPassword = AES_ENCRYPT(Password1,SHA2(Id1,512))) THEN
BEGIN
select * from Cashiers where Id = Id1 and Password = AES_ENCRYPT(Password1,SHA2(Id1,512));
END;
ELSE
BEGIN
select null;
END;
END IF;
END;
ELSE
BEGIN
select null;
END;
END IF;
END
```

Celem procedury jest przeprowadzenie autoryzacji logującego się użytkownika. Korzystamy w tym przypadku z autorskiego mechanizmu. Polega on na tym, że jako parametr procedury podajemy ID kasjera, oraz jego hasło. Następnie sprawdzamy czy istnieje w bazie danych dany kasjer, jeśli tak to wyciągamy jego zaszyfrowane hasło z bazy danych. W kolejnym kroku szyfrujemy hasło, które podał

użytkownik, a następnie porównujemy je z tym z bazy danych. Jeśli hasła pasują do siebie to jako element zwracany odsyłamy rekord z danymi kasjera.

b) procedura regCashier

```
BEGIN
insert into Cashiers (Id, Password, Name, Surname) values (Id1, AES_ENCRYPT(Password1,SHA2(Id1,512)), Name1, Surname1);
END
```

Efektem działania tej procedury jest stworzenie w bazie danych kasjera, wraz z jego imieniem, nazwiskiem, loginem oraz hasłem. Hasło jest szyfrowane w celu zabezpieczenia danych w bazie danych. Jako parametr w procedurze podajemy kolejno login, hasło, imię, nazwisko.

c) procedura reportDay

```
BEGIN
select p.Name, SUM(o.Count) as 'Sum', (p.Price * SUM(o.Count)) as 'TotalPrice'
from Orders o join Transactions t join Products p
where o.TransactionId = t.IdTransaction and o.ProductId = p.IdProduct and o.`Return` = 0
and day(t.Date) = day(curdate()) group by Name with rollup;
END
```

Procedura bezparametrowa. Efektem jej wywołania jest zestawienie dziennego raportu sprzedanych produktów wraz z podliczeniem ilości produktów i łącznej ceny sprzedanych rzeczy danego dnia.

d) procedura reportMonth

```
BEGIN
select p.Name, SUM(o.Count) as 'Sum', (p.Price * SUM(o.Count)) as 'TotalPrice'
from Orders o join Transactions t join Products p
where o.TransactionId = t.IdTransaction and o.ProductId = p.IdProduct and o.`Return` = 0
and month(t.Date) = month(curdate()) group by Name with rollup;
END
```

Procedura bezparametrowa. Efektem jej wywołania jest zestawienie miesięcznego raportu sprzedanych produktów wraz z podliczeniem ilości produktów i łącznej ceny sprzedanych rzeczy danego miesiąca.

e) procedura returnProduct

```
BEGIN
IF EXISTS (select IdOrder from Orders where IdOrder = IdOrder1 AND Count >= Count1 AND Count1 >= 1 AND `Return` = 0) THEN
BEGIN
select Count into @Count from Orders where IdOrder = IdOrder1;
select Price into @Price from Products p join Orders o
where p.IdProduct = o.ProductId and o.IdOrder = IdOrder1;
select TransactionId into @TransactionId from Orders where IdOrder = IdOrder1;
IF (@Count = Count1) THEN
BEGIN
update Orders set `Return` = 1 where IdOrder = IdOrder1;
update Transactions set TotalPrice = TotalPrice - (@Price * Count1)
where IdTransaction = @TransactionId;
END;
ELSE
BEGIN
insert into Orders (ProductId, TransactionId, `Return`, Count)
select ProductId, TransactionId, `Return`, (Count - Count1) from Orders
where IdOrder = IdOrder1;
update Orders set `Return` = 1, Count = Count1 where IdOrder = IdOrder1;
update Transactions set TotalPrice = TotalPrice - (@Price * Count1)
where IdTransaction = @TransactionId;
END;
END IF;
END;
END IF;
END
```

Procedura, której zadaniem jest poprawne przeprowadzenie zwrotu produktów w bazie danych. Jako parametr przyjmuje ona id zwracanego produktu oraz ilość, którą zwracamy. Procedura sprawdza czy jest możliwe, aby zaistniał zwrot danego produktu (sprawdza czy taki produkt istnieje, czy liczba kupionych sztuk jest większa lub równa liczbie zwracanej, czy zwracamy jedną lub więcej sztuk oraz czy dany produkt nie był już wcześniej zwrócony). Jeśli zwracamy tyle sztuk produktu, ile zakupiliśmy to zmieniana jest flaga w danym produkcie Return na 1 oraz aktualizujemy cenę transakcji do której należał dany produkt poprzez odjęcie od niej ceny zwracanych produktów. W przeciwnym wypadku kopiujemy nasz rekord w tablicy Orders z pozostałymi nie zwróconymi sztukami towaru, a stary wpis aktualizujemy tak, aby zostawić w nim tylko zwrócone produkty, oraz aktualizujemy cenę transakcji do której należał dany produkt poprzez odjęcie od niej ceny zwracanych produktów.

f) procedura topProducts

```
BEGIN
select @number:=@number + 1 as 'Ordinal', Name, ordercount as 'TotalSales'
from (
select Name, SUM(Count) as ordercount from
Orders o join Products p
where o.ProductId = p.IdProduct group by Name order by SUM(Count)
desc limit 10
) t1, (select @number:=0) t2;
END
```

Procedura bezparametrowa, której zadaniem jest wyświetlić top 10 najlepiej sprzedających się produktów.

g) trigger addTransaction

```
BEGIN
SELECT `auto_increment` into @last_id FROM INFORMATION_SCHEMA.TABLES
WHERE table_name = 'Transactions';
set new.Id = @last_id+1000;
END
```

Trigger odpowiedzialny za dodatkowe autoinkrementowanie kolumny ID w tabeli Transactions.

# 5. PluginLogIn

## a. Opis projektu

Plugin ma za zadanie umożliwić zalogowanie do aplikacji. Pluginy stosuję się w celu umożliwienia rozszerzenia aplikacji, bez zmieniania kodu. Na samym początku Plugin sprawdzał tylko poprawność loginu i hasła dla jednego zahardkorowanego przypadku. Następnie ten plugin zamieniliśmy pluginem w obecnej postaci, bez zmiany kodu, a dodając rzeczywiste logowanie.

## b. Najważniejsze kody wraz z komentarzem

a) Interface, jego zadaniem jest dostarczenie metody dla klasy implementującej. Duża liczba nadpisywanych zmiennych wynika z faktu zapotrzebowania na nie w kodzie.

```
10  namespace PluginLogIn
11  {
        4 references
12      public interface ILogIn
13      {
            2 references
14          bool CheckLoginData(string id, string password,ref ProductsClient productsClient,
15              ref CashierClient cashierClient,ref Cashier cashier,ref List<Product> products, ref string token);
16      }
17  }
```

b) Implementacja interfejsu wykorzystująca do logowania token generowany przez odpowiedni projekt.

```
11  namespace PluginLogIn
12  {
        0 references
13      public class LogIn : ILogIn
14      {
            2 references
15          public bool CheckLoginData(string id, string password, ref ProductsClient productsClient,
16              ref CashierClient cashierClient, ref Cashier cashier, ref List<Product> products, ref string token)
17          {
18              token = TokenRequester.ReuqestToken(id, password);
19              bool result = false;
20              if (token != null)
21              {
22                  cashierClient = new CashierClient(token);
23                  productsClient = new ProductsClient(token);
24
25                  cashier = cashierClient.Login(id);
26                  products = productsClient.GetProducts();
27                  result = true;
28              }else { result = false; }
29
30              if (cashier != null && products.Count > 0 && result)
31              {
32                  result = true;
33              }else { result = false; }
34              return result;
35          }
36      }
37  }
```

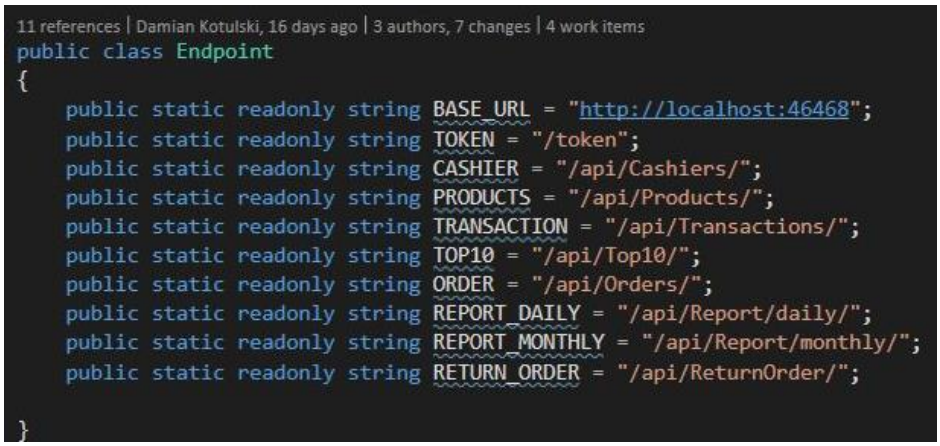
# 6. SmartShop.CommunicateToWebService

## a. Opis projektu

Projekt zawiera klasy odpowiedzialne za komunikację aplikacji z WebService'm.

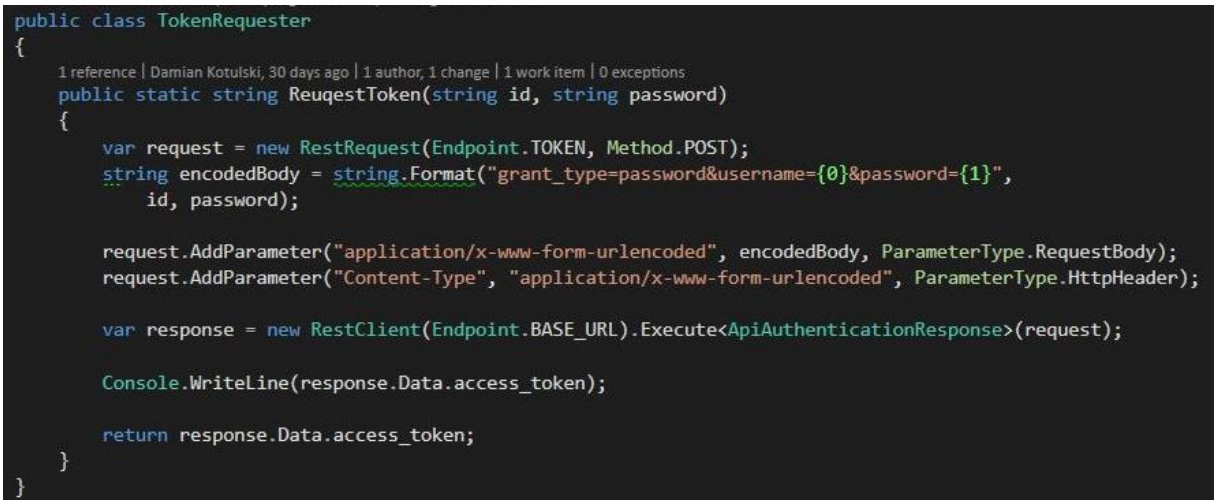
## b. Najważniejsze kody wraz z komentarzem

Klasa Enpoint – zawiera statyczne pola, będące adresami kontrolerów usług



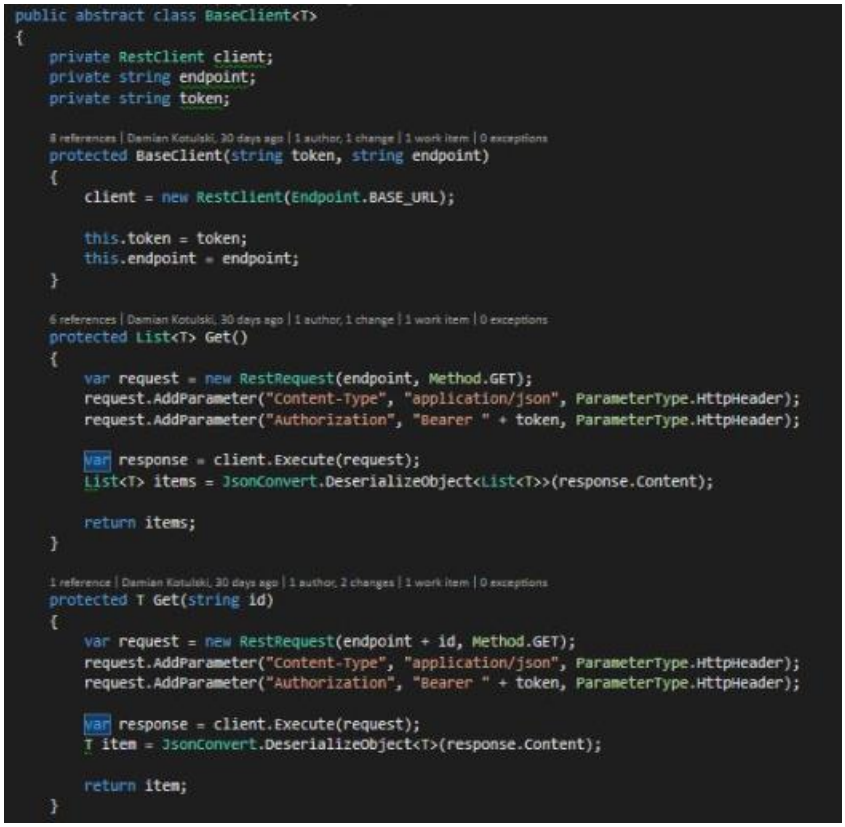
```
11 references | Damian Kotulski, 16 days ago | 3 authors, 7 changes | 4 work items
public class Endpoint
{
    public static readonly string BASE_URL = "http://localhost:46468";
    public static readonly string TOKEN = "/token";
    public static readonly string CASHIER = "/api/Cashiers/";
    public static readonly string PRODUCTS = "/api/Products/";
    public static readonly string TRANSACTION = "/api/Transactions/";
    public static readonly string TOP10 = "/api/Top10/";
    public static readonly string ORDER = "/api/Orders/";
    public static readonly string REPORT_DAILY = "/api/Report/daily/";
    public static readonly string REPORT_MONTHLY = "/api/Report/monthly/";
    public static readonly string RETURN_ORDER = "/api/ReturnOrder/";

}
```

Klasa TokenRequester – zawiera metodę odpowiedzialną za uzyskanie tokenu, niezbędnego do dalszej komunikacji

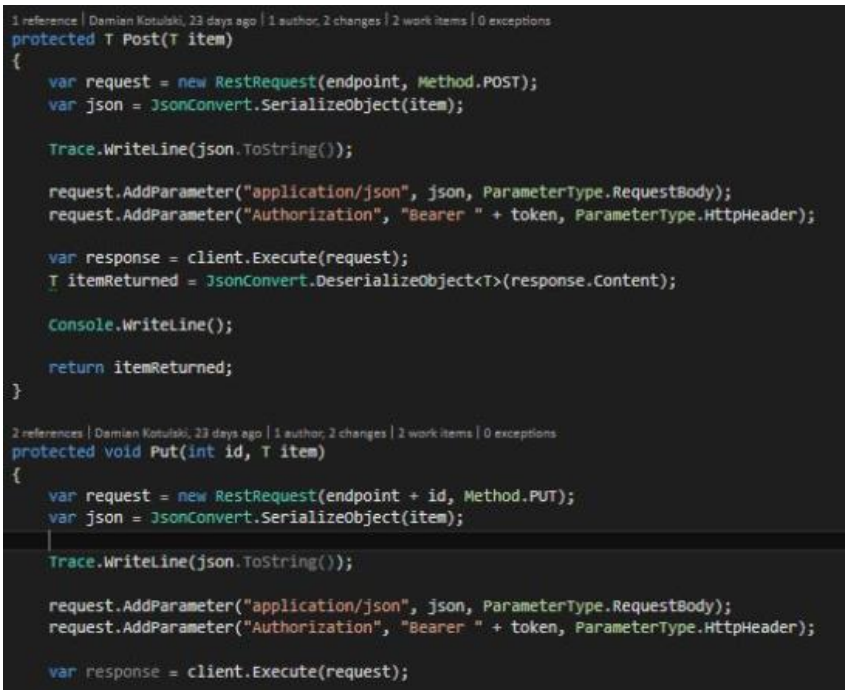


```
public class TokenRequester
{
    1 reference | Damian Kotulski, 30 days ago | 1 author, 1 change | 1 work item | 0 exceptions
    public static string ReuqestToken(string id, string password)
    {
        var request = new RestRequest(Endpoint.TOKEN, Method.POST);
        string encodedBody = string.Format("grant_type=password&username={0}&password={1}",
            id, password);

        request.AddParameter("application/x-www-form-urlencoded", encodedBody, ParameterType.RequestBody);
        request.AddParameter("Content-Type", "application/x-www-form-urlencoded", ParameterType.HttpHeader);

        var response = new RestClient(Endpoint.BASE_URL).Execute<ApiAuthenticationResponse>(request);

        Console.WriteLine(response.Data.access_token);

        return response.Data.access_token;
    }
}
```

Klasa BaseClient – bazowy klient odpowiedzialny za komunikację, klasa abstrakcyjna zawierająca generyczne metody do wykonywania zapytań http



```
public abstract class BaseClient<T>
{
    private RestClient client;
    private string endpoint;
    private string token;

    8 references | Damian Kotulski, 30 days ago | 1 author, 1 change | 1 work item | 0 exceptions
    protected BaseClient(string token, string endpoint)
    {
        client = new RestClient(Endpoint.BASE_URL);

        this.token = token;
        this.endpoint = endpoint;
    }

    6 references | Damian Kotulski, 30 days ago | 1 author, 1 change | 1 work item | 0 exceptions
    protected List<T> Get()
    {
        var request = new RestRequest(endpoint, Method.GET);
        request.AddParameter("Content-Type", "application/json", ParameterType.HttpHeader);
        request.AddParameter("Authorization", "Bearer " + token, ParameterType.HttpHeader);

        var response = client.Execute(request);
        List<T> items = JsonConvert.DeserializeObject<List<T>>(response.Content);

        return items;
    }

    1 reference | Damian Kotulski, 30 days ago | 1 author, 2 changes | 1 work item | 0 exceptions
    protected T Get(string id)
    {
        var request = new RestRequest(endpoint + id, Method.GET);
        request.AddParameter("Content-Type", "application/json", ParameterType.HttpHeader);
        request.AddParameter("Authorization", "Bearer " + token, ParameterType.HttpHeader);

        var response = client.Execute(request);
        T item = JsonConvert.DeserializeObject<T>(response.Content);

        return item;
    }
```



```
1 reference | Damian Kotulski, 23 days ago | 1 author, 2 changes | 2 work items | 0 exceptions
protected T Post(T item)
{
    var request = new RestRequest(endpoint, Method.POST);
    var json = JsonConvert.SerializeObject(item);

    Trace.WriteLine(json.ToString());

    request.AddParameter("application/json", json, ParameterType.RequestBody);
    request.AddParameter("Authorization", "Bearer " + token, ParameterType.HttpHeader);

    var response = client.Execute(request);
    T itemReturned = JsonConvert.DeserializeObject<T>(response.Content);

    Console.WriteLine();

    return itemReturned;
}

2 references | Damian Kotulski, 23 days ago | 1 author, 2 changes | 2 work items | 0 exceptions
protected void Put(int id, T item)
{
    var request = new RestRequest(endpoint + id, Method.PUT);
    var json = JsonConvert.SerializeObject(item);

    Trace.WriteLine(json.ToString());

    request.AddParameter("application/json", json, ParameterType.RequestBody);
    request.AddParameter("Authorization", "Bearer " + token, ParameterType.HttpHeader);

    var response = client.Execute(request);
```

Pozostałe klasy stanowią rozszerzenia klasy BaseClient i służą do obsługi dostępu do usług konkretnych kontrolerów WebService'u. Przykład:

Klasa TransactionClient – obsługuje utworzenie, aktualizację i pobranie transakcji:

```
6 references | Damian Kotulski, 27 days ago | 1 author, 4 changes | 1 work item
public class TransactionClient : BaseClient<Transaction>
{
    3 references | Damian Kotulski, 30 days ago | 1 author, 1 change | 1 work item | 0 exceptions
    public TransactionClient(string token) : base(token, Endpoint.TRANSACTION) { }

    1 reference | Damian Kotulski, 30 days ago | 1 author, 1 change | 1 work item | 0 exceptions
    public Transaction CreateNew(Transaction transaction)
    {
        return base.Post(transaction);
    }

    1 reference | Damian Kotulski, 29 days ago | 1 author, 1 change | 1 work item | 0 exceptions
    public void UpdateTransaction(Transaction transaction)
    {
        base.Put(transaction.IdTransaction, transaction);
    }

    1 reference | Damian Kotulski, 27 days ago | 1 author, 1 change | 0 exceptions
    public List<Transaction> GetTransactions()
    {
        return base.Get();
    }
}
```

## 7. SmartShop.Models

### a. Opis projektu

Stworzenie tego projektu miało na celu wyodrębnienie modeli klas z których będziemy korzystać w pozostałych projektach, aby nie tworzyć każdego z nich wieloktrotnie.

### b. Najważniejsze kody wraz z komentarzem

Projekt składa się z pojedyńczych klas pełniących role pojedyńczych modeli. Zostały one stworzone na podstawie wygenerowanych klas EntityFramework w webserviscie.

Kody poszczególnych modeli:

```
4 references
public class BestSellingProduct
{
    0 references
    public int Ordinal { get; set; }
    0 references
    public String Name { get; set; }
    0 references
    public int TotalSales { get; set; }
}
```

```
8 references
public class Cashbox
{
    2 references | 0/1 passing
    public Cashbox()
    {
        this.Transactions = new HashSet<Transaction>();
    }
    3 references | 0/1 passing
    public int IdCashbox { get; set; }
    1 reference
    public int ShopId { get; set; }
    2 references
    public int Id { get; set; }
    0 references
    public virtual Shop Shop { get; set; }
    1 reference
    public virtual ICollection<Transaction> Transactions { get; set; }

}
```

```
10 references
public class Cashier
{
    1 reference | 0/1 passing
    public Cashier()
    {
        this.Transactions = new HashSet<Transaction>();
    }
    2 references | 0/1 passing
    public int IdCashier { get; set; }
    1 reference
    public string Id { get; set; }
    0 references
    public string Password { get; set; }
    0 references
    public string Name { get; set; }
    0 references
    public string Surname { get; set; }
    1 reference
    public virtual ICollection<Transaction> Transactions { get; set; }

}
```

```
2 references
public class Category
{
    0 references
    public Category()
    {
        this.Products = new HashSet<Product>();
    }

    0 references
    public int IdCategory { get; set; }
    0 references
    public string Name { get; set; }
    1 reference
    public virtual ICollection<Product> Products { get; set; }

}

13 references
public class Order
{
    1 reference
    public int IdOrder { get; set; }
    1 reference
    public int ProductId { get; set; }
    1 reference
    public Nullable<int> TransactionId { get; set; }
    1 reference
    public sbyte Return { get; set; }
    2 references
    public sbyte Count { get; set; }
    2 references
    public Nullable<float> Discount { get; set; }

    4 references
    public virtual Product Product { get; set; }
    0 references
    public virtual Transaction Transaction { get; set; }
}
```

```
33 references
public class Product
{
    5 references | 0/3 passing
    public Product()
    {
        this.Orders = new HashSet<Order>();
    }

    2 references
    public int IdProduct { get; set; }
    0 references
    public int CategoryId { get; set; }
    5 references
    public string Name { get; set; }
    5 references | 0/3 passing
    public string Code { get; set; }
    4 references
    public float Price { get; set; }
    4 references
    public byte[] Image { get; set; }

    0 references
    public virtual Category Category { get; set; }
    1 reference
    public virtual ICollection<Order> Orders { get; set; }

}
```

```
2 references
public class Shop
{
    0 references
    public Shop()
    {
        this.Cashboxes = new HashSet<Cashbox>();
    }

    0 references
    public int IdShop { get; set; }
    0 references
    public string Name { get; set; }
    0 references
    public string Address { get; set; }
    1 reference
    public virtual ICollection<Cashbox> Cashboxes { get; set; }

}
```

```
20 references
public class Transaction
{
    1 reference
    public Transaction()
    {
        this.Orders = new HashSet<Order>();
    }

    2 references
    public int IdTransaction { get; set; }
    2 references
    public int CashboxId { get; set; }
    2 references
    public int CashierId { get; set; }
    5 references
    public int Id { get; set; }
    2 references
    public System.DateTime Date { get; set; }
    3 references
    public float TotalPrice { get; set; }
    0 references
    public Nullable<float> Discount { get; set; }
    0 references
    public virtual Cashbox Cashbox { get; set; }
    0 references
    public virtual Cashier Cashier { get; set; }
    3 references
    public virtual ICollection<Order> Orders { get; set; }

}
```

## 8. SmartShop.Tests

### a. Opis projektu

Projekt został wykonany, z powodu potrzeby zrobienia testów jednostkowych dla całej solucji. Zostały w nich wykonanych siedem testów jednostkowych, wszystkie przechodzą. Do ich utworzenia wykorzystano bibliotekę NUnit w wersji 3.

### b. Najważniejsze kody wraz z komentarzem

Przetestowane zostały dwie klasy z solucji, wybrane z nich metody.

a) ManuallyCodeTests

```
[Test]
0 references
public void CheckTheCodeIfResultIsGood()
{
    //ARRANGE
    ManuallyCode mC = ManuallyCode.GetInstance();
    List<Product> exampleList = new List<Product>();
    Product prod1 = new Product();
    prod1.Code = "333";
    exampleList.Add(prod1);
    //ACT
    bool result = mC.CheckTheCode("333", exampleList);
    bool expected = true;
    //ASSERT
    Assert.AreEqual(expected, result);
}
```

Powyższy test sprawdza czy metoda zwraca odpowiednią wartość jeśli spełni swoje zadanie, tzn. znajdzie kod produktu w liście.

```
[Test]
0 references
public void CheckTheCodeIfResultIsWrong()
{
    //ARRANGE
    ManuallyCode mC = ManuallyCode.GetInstance();
    List<Product> exampleList = new List<Product>();
    Product prod1 = new Product();
    prod1.Code = "333";
    exampleList.Add(prod1);
    //ACT
    bool result = mC.CheckTheCode("444", exampleList);
    bool expected = false;
    //ASSERT
    Assert.AreEqual(expected, result);
}
```

Powyższy test sprawdza czy metoda zwraca odpowiednią wartość jeśli spełni swoje zadanie, tzn. nie znajdzie kodu produktu w liście.

```
[Test]
0 references
public void CheckTheCodeIfOneOfParametersIsNull()
{
    //ARRANGE
    ManuallyCode mC = ManuallyCode.GetInstance();
    List<Product> exampleList = new List<Product>();
    Product prod1 = new Product();
    prod1.Code = "333";
    exampleList.Add(prod1);
    //ACT
    bool result = mC.CheckTheCode(null, exampleList);
    bool secondResult = mC.CheckTheCode("333", null);
    bool expected = false;
    //ASSERT
    Assert.AreEqual(expected, result);
    Assert.AreEqual(expected, secondResult);
}
```

Powyższy test sprawdza czy metoda zwraca odpowiednią wartość jeśli spełni swoje zadanie, tzn. nie jeden z parametrów wchodzącym do metody jest nullem.

```
[Test]
[TestCase(1,2)]
0 references
public void AddToBasketListIfReturnGoodBasket(int x, int y)
{
    //ARRANGE
    byte[] image = new byte[] { 0x20 };
    Product product = new Product() { Name = "ex", Image = new byte[] { 0x20 }, Price = 20 };
    ManuallyCode mC = ManuallyCode.GetInstance();
    mC.checkedProduct = product;
    //ACT
    Basket basketExpected = new Basket() { Name = product.Name, Image = product.Image,
        SigleWithoutVatPrice = product.Price, Number = y, Count = x };
    Basket basketResult =  mC.AddToBasketList(x, y);
    //ASSERT
    Assert.IsTrue(basketExpected.Count == basketResult.Count &&
        basketExpected.Number == basketResult.Number &&
        basketExpected.Name == basketResult.Name &&
        basketExpected.Image == basketResult.Image &&
        basketExpected.SigleWithoutVatPrice == basketResult.SigleWithoutVatPrice);
}
```

Powyższy test sprawdza czy metoda zwraca odpowiednią wartość jeśli spełni swoje zadanie, tzn. zwraca obiekt klasy Basket z odpowiednimi polami. Wykorzystuje także TestCasy.

```
[Test]
0 references
public void AddToBasketListIfGetCountOrCounterAreLessThenOneAndThrowArgumentException()
{
    //ARRANGE
    ManuallyCode mC = ManuallyCode.GetInstance();
    //ACT

    //ASSERT
    Assert.Throws(typeof(ArgumentException), () => mC.AddToBasketList(-1, -1));
}
```

Powyższy test sprawdza czy metoda zwraca odpowiednią wartość jeśli spełni swoje zadanie, tzn. zwraca błąd jeśli wprowadzone dane są nieprawidłowe, mniejsze niż zero.

b) TransactionManagerTests

```
public class TransactionManagerTests
{
    private DataHandler data;
    TransactionManager tM;
    [OneTimeSetUp]
    0 references
    public void Init()
    {
        data = DataHandler.GetInstance();
        tM = new TransactionManager();
    }

    [Test]
    0 references
    public void PrepareNewTrasactionIfHappend()
    {
        //ARRANGE
        data.Cashbox = new Cashbox() { IdCashbox = 5 };
        data.Cashier = new Cashier() { IdCashier = 5 };
        data.Token = "opo";
        //ACT
        tM.PrepareNewTransaction();
        //ASSERT
        Assert.AreNotEqual(null, data.Transaction);
    }
```

Powyższy test sprawdza czy metoda zwraca odpowiednią wartość jeśli spełni swoje zadanie, tzn. sprawdza czy transakcja została utworzona. Co więcej, test przechodzi tylko jako Run Selected, z powodu nie błędu testu, a zależności wystepującej w źle przeprojektowanej klasie.

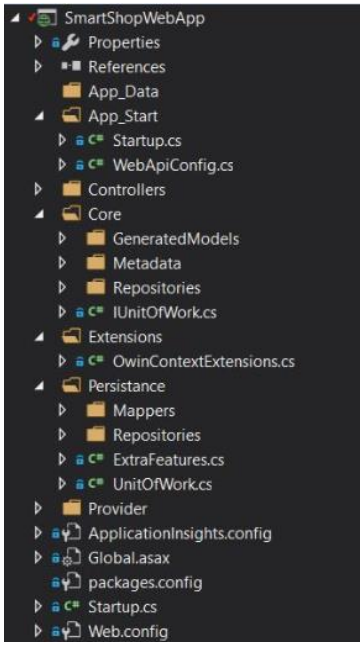
# 9. SmartShopWebApp

## a. Opis projektu

Projekt zawiera warstwę WebService'u, zaimplementowaną na bazie wzorca REST, zabezpieczoną mechanizmem autoryzacji opartym o standard OAuth 2.0. Wykorzystano EntityFramework i podejście database first oraz wzorce Repozytorium i UnitOfWork w celu łatwiejszego i ujednoliconego dostępu do danych z modeli podczas wykorzystywania ich w kontrolerach.

## b. Najważniejsze kody wraz z komentarzem

- Struktura projektu




- Omówienie najważniejszych komponentów

**App_Start:** klasa Startup.cs

```
3 references | Damian Kotulski, 35 days ago | 1 author, 3 changes | 2 work items
public partial class Startup
{
    private static readonly string TOKEN_ENDPOINT = "/token";
    3 references | Damian Kotulski, 36 days ago | 1 author, 1 change | 0 exceptions
    public static OAuthAuthorizationServerOptions OauthOptions { get; set; }

    0 references | Damian Kotulski, 36 days ago | 1 author, 2 changes | 1 work item | 0 exceptions
    static Startup()
    {
        OauthOptions = new OAuthAuthorizationServerOptions
        {
            TokenEndpointPath = new PathString(TOKEN_ENDPOINT),
            Provider = new OAuthAppProvider(),
            AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
            AllowInsecureHttp = true
        };
    }

    1 reference | Damian Kotulski, 36 days ago | 1 author, 1 change | 0 exceptions
    public void ConfigureAuth(IAppBuilder app)
    {
        app.UseOAuthBearerTokens(OauthOptions);
    }
}
```

- konfiguracja mechanizmu uzyskiwania tokena (ścieżka dostępu, czas wygaśnięcia) do komunikacji zgodnej ze standarem OAuth 2.0

**App_Start:** klasa WebApiConfig.cs
- konfigurację usług i reguły routingu

**Controllers:** klasy będące kontrolerami usług, reagują na zapytania http i zwracają odpowiedzi w formacie json

Specyfikacja najważniejszych dostępnych usług:

| | Pobieranie GET | Dodawanie POST | Modyfikacja PUT | Usuwanie DELETE |
|---|---|---|---|---|
| Shop | X (w Cashboxes) | X | X | X |
| Cashboxes | **TAK** | X | X | X |
| Cashiers | TAK | X | TAK (hasło) | X |
| Transactions | TAK | TAK | TAK | X |
| Orders | TAK | X (w Transactions) | TAK | X |
| Products | TAK | X | X | X |
| Categories | X (w Products) | X | X | X |

## Cashiers

- GET

**ENDPOINT**: /api/cashiers

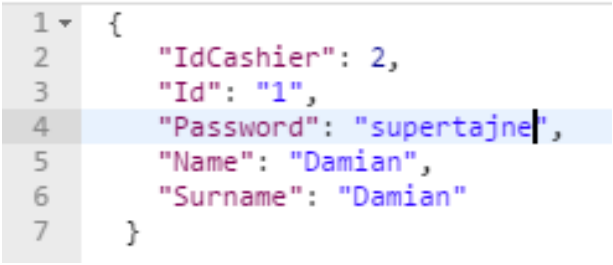
PRZYKŁADOWA ODPOWIEDŹ:


```
[
  {
    "IdCashier": 2,
    "Id": "1",
    "Password": "tajne",
    "Name": "Damian",
    "Surname": "Damian"
  },
  {
    "IdCashier": 3,
    "Id": "2",
    "Password": "cichy",
    "Name": "Tomasz ",
    "Surname": "Tomasz"
  },
  {
    "IdCashier": 4,
    "Id": "3",
    "Password": "pufne",
    "Name": "Krzysztof",
    "Surname": "Kszysztof"
  },
  {
    "IdCashier": 5,
    "Id": "4",
    "Password": "tajemne",
    "Name": "Ola",
    "Surname": "Ola"
  }
]
```

- PUT

**ENDPOINT**: /api/transactions/{idCashier}

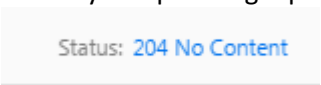
PRZYKŁADOWE BODY:



```
{
    "IdCashier": 2,
    "Id": "1",
    "Password": "supertajne",
    "Name": "Damian",
    "Surname": "Damian"
}
```

ODPOWIEDŹ:

Bez odpowiedzi, ale status informuje, że wszystko przebiegło pomyślnie:

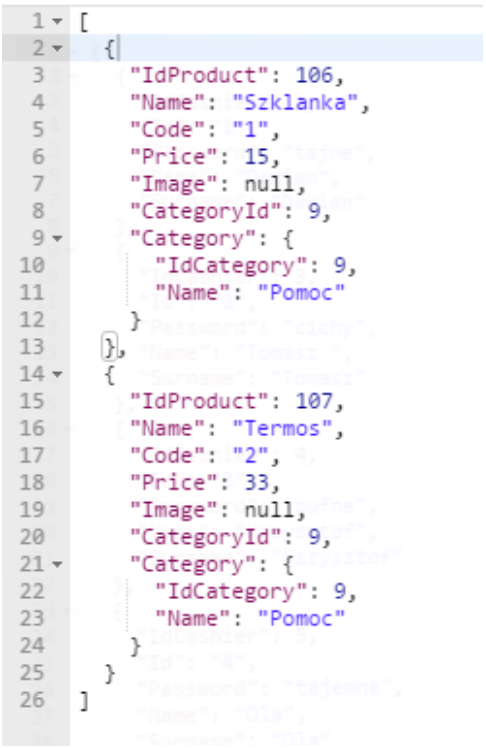



## Products

- GET

**ENDPOINT**: /api/products

PRZYKŁADOWA ODPOWIEDŹ:



```
[
  {
    "IdProduct": 106,
    "Name": "Szklanka",
    "Code": "1",
    "Price": 15,
    "Image": null,
    "CategoryId": 9,
    "Category": {
      "IdCategory": 9,
      "Name": "Pomoc"
    }
  },
  {
    "IdProduct": 107,
    "Name": "Termos",
    "Code": "2",
    "Price": 33,
    "Image": null,
    "CategoryId": 9,
    "Category": {
      "IdCategory": 9,
      "Name": "Pomoc"
    }
  }
]
```

# Cashboxes

- GET

**ENDPOINT**: /api/cashboxes I /api/ cashboxes /{id} dla konkretnego idCashbox

PRZYKŁADOWA ODPOWIEDŹ:


```
[
  {
    "IdCashbox": 13,
    "Id": 1,
    "ShopId": 14,
    "Shop": {
      "IdShop": 14,
      "Name": "Spozywczy",
      "Address": "Katowice"
    }
  },
  {
    "IdCashbox": 15,
    "Id": 2,
    "ShopId": 15,
    "Shop": {
      "IdShop": 15,
      "Name": "Przemyslowy",
      "Address": "Bytom"
    }
  }
]
```


# Orders

- GET

**ENDPOINT**: /api/orders

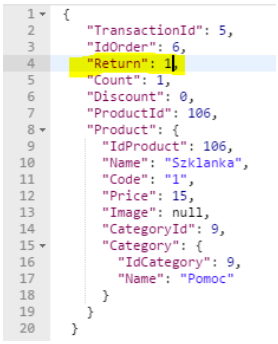
PRZYKŁADOWA ODPOWIEDŹ:


```
[
  {
    "TransactionId": 5,
    "IdOrder": 6,
    "Return": 1,
    "Count": 1,
    "Discount": 0,
    "ProductId": 106,
    "Product": {
      "IdProduct": 106,
      "Name": "Szklanka",
      "Code": "1",
      "Price": 15,
      "Image": null,
      "CategoryId": 9,
      "Category": {
        "IdCategory": 9,
        "Name": "Pomoc"
      }
    }
  },
  {
    "TransactionId": 11,
    "IdOrder": 12,
    "Return": 0,
    "Count": 1,
    "Discount": 0,
    "ProductId": 106,
    "Product": {
      "IdProduct": 106,
      "Name": "Szklanka",
      "Code": "1",
      "Price": 15,
      "Image": null,
      "CategoryId": 9,
      "Category": {
        "IdCategory": 9,
        "Name": "Pomoc"
      }
    }
  }
]
```

- PUT

**ENDPOINT**: /api/transactions/{idTransaction}

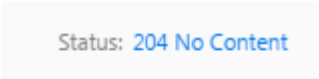
PRZYKŁADOWE BODY: (zwracamy produkt)



```
{
  "TransactionId": 5,
  "IdOrder": 6,
  "Return": 1,
  "Count": 1,
  "Discount": 0,
  "ProductId": 106,
  "Product": {
    "IdProduct": 106,
    "Name": "Szklanka",
    "Code": "1",
    "Price": 15,
    "Image": null,
    "CategoryId": 9,
    "Category": {
      "IdCategory": 9,
      "Name": "Pomoc"
    }
  }
}
```


ODPOWIEDŹ:

Bez odpowiedzi, ale status informuje, że wszystko przebiegło pomyślnie:




# Transactions

- GET

**ENDPOINT**: /api/transactions I /api/transactions/{id} dla konkretnego idTransaction

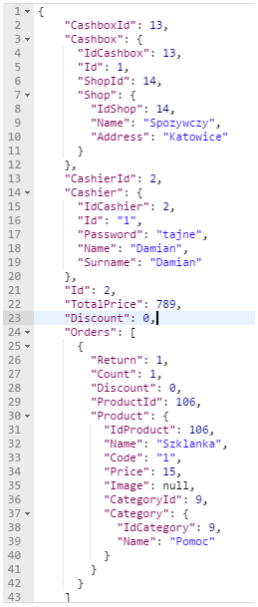
PRZYKŁADOWA ODPOWIEDŹ:

```
[
  {
    "IdTransaction": 5,
    "CashboxId": 13,
    "Cashbox": {
      "IdCashbox": 13,
      "Id": 1,
      "ShopId": 14,
      "Shop": {
        "IdShop": 14,
        "Name": "Spozywczy",
        "Address": "Katowice"
      }
    },
    "CashierId": 2,
    "Cashier": {
      "IdCashier": 2,
      "Id": "1",
      "Password": "tajne",
      "Name": "Damian",
      "Surname": "Damian"
    },
    "Id": 1,
    "Date": "0001-01-01T00:00:00",
    "TotalPrice": 789,
    "Discount": 0,
    "Orders": [
      {
        "TransactionId": 5,
        "IdOrder": 6,
        "Return": 1,
        "Count": 1,
        "Discount": 0,
        "ProductId": 106,
        "Product": {
          "IdProduct": 106,
          "Name": "Szklanka",
          "Code": "1",
          "Price": 15,
          "Image": null,
          "CategoryId": 9,
          "Category": {
            "IdCategory": 9,
            "Name": "Pomoc"
          }
        }
      }
    ]
  }
]
```

- POST

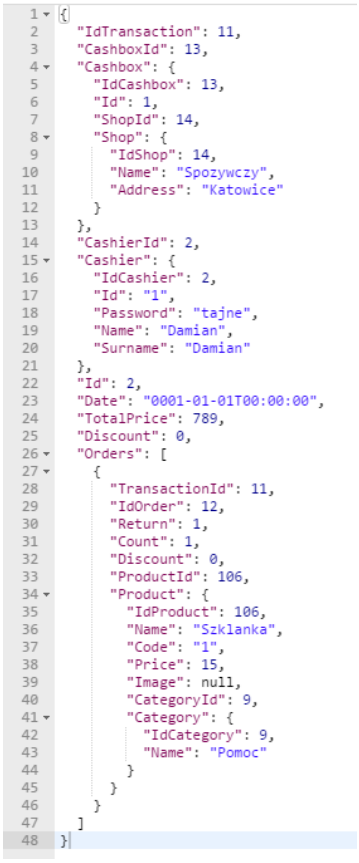
**ENDPOINT**: /api/transactions

PRZYKŁADOWE BODY: (nie ma idtransaction i idorder)



```
{
    "CashboxId": 13,
    "Cashbox": {
      "IdCashbox": 13,
      "Id": 1,
      "ShopId": 14,
      "Shop": {
        "IdShop": 14,
        "Name": "Spozywczy",
        "Address": "Katowice"
      }
    },
    "CashierId": 2,
    "Cashier": {
      "IdCashier": 2,
      "Id": "1",
      "Password": "tajne",
      "Name": "Damian",
      "Surname": "Damian"
    },
    "Id": 2,
    "TotalPrice": 789,
    "Discount": 0,
    "Orders": [
      {
        "Return": 1,
        "Count": 1,
        "Discount": 0,
        "ProductId": 106,
        "Product": {
          "IdProduct": 106,
          "Name": "Szklanka",
          "Code": "1",
          "Price": 15,
          "Image": null,
          "CategoryId": 9,
          "Category": {
            "IdCategory": 9,
            "Name": "Pomoc"
          }
        }
      }
    ]
```

PRZYKŁADOWA ODPOWIEDŹ: (dostajemy odpowiedź z wygenerowanym idTransaction i idOrder dla każdego z zamówienia)



```
{
  "IdTransaction": 11,
  "CashboxId": 13,
  "Cashbox": {
    "IdCashbox": 13,
    "Id": 1,
    "ShopId": 14,
    "Shop": {
      "IdShop": 14,
      "Name": "Spozywczy",
      "Address": "Katowice"
    }
  },
  "CashierId": 2,
  "Cashier": {
    "IdCashier": 2,
    "Id": "1",
    "Password": "tajne",
    "Name": "Damian",
    "Surname": "Damian"
  },
  "Id": 2,
  "Date": "0001-01-01T00:00:00",
  "TotalPrice": 789,
  "Discount": 0,
  "Orders": [
    {
      "TransactionId": 11,
      "IdOrder": 12,
      "Return": 1,
      "Count": 1,
      "Discount": 0,
      "ProductId": 106,
      "Product": {
        "IdProduct": 106,
        "Name": "Szklanka",
        "Code": "1",
        "Price": 15,
        "Image": null,
        "CategoryId": 9,
        "Category": {
          "IdCategory": 9,
          "Name": "Pomoc"
        }
      }
    }
  ]
}
```

- PUT

**ENDPOINT**: /api/transactions/{idTransaction}

PRZYKŁADOWE BODY: (zwracamy produkt)

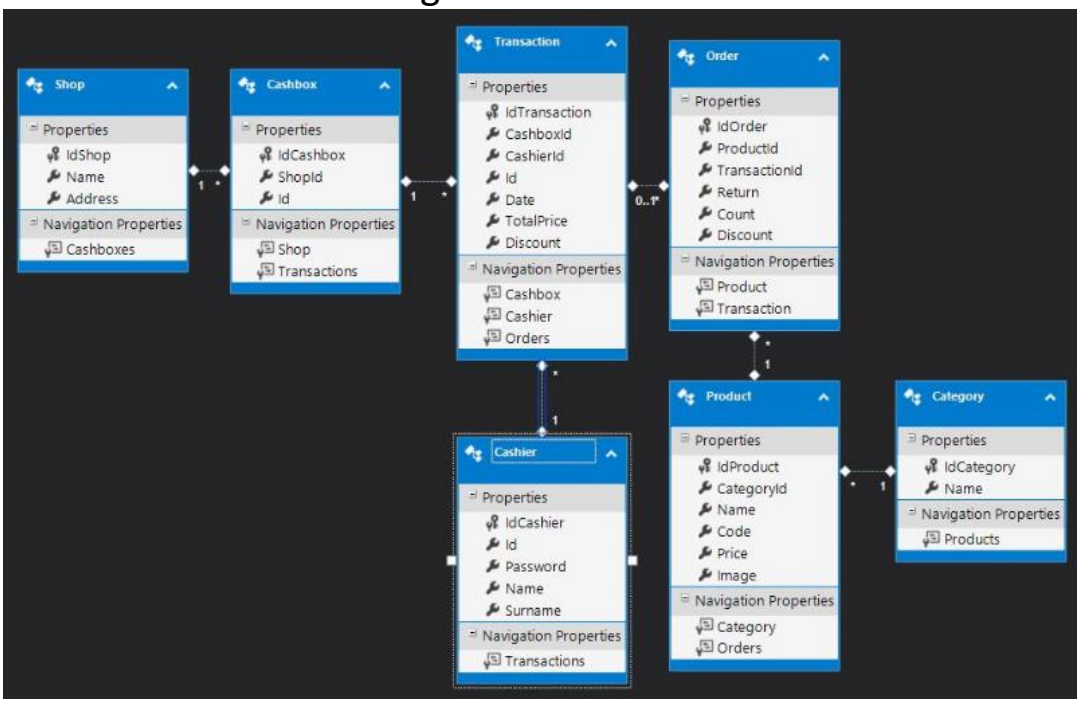
```
{
  "IdTransaction": 11,
  "CashboxId": 13,
  "Cashbox": {
    "IdCashbox": 13,
    "Id": 1,
    "ShopId": 14,
    "Shop": {
      "IdShop": 14,
      "Name": "Spozywczy",
      "Address": "Katowice"
    }
  },
  "CashierId": 2,
  "Cashier": {
    "IdCashier": 2,
    "Id": "1",
    "Password": "tajne",
    "Name": "Damian",
    "Surname": "Damian"
  },
  "Id": 2,
  "Date": "2017-05-04",
  "TotalPrice": 789,
  "Discount": 0,
  "Orders": [
    {
      "TransactionId": 11,
      "IdOrder": 12,
      "Return": 1,
      "Count": 1,
      "Discount": 0,
      "ProductId": 106,
      "Product": {
        "IdProduct": 106,
        "Name": "Szklanka",
        "Code": "1",
        "Price": 15,
        "Image": null,
        "CategoryId": 9,
        "Category": {
          "IdCategory": 9,
          "Name": "Pomoc"
```

ODPOWIEDŹ:

Bez odpowiedzi, ale status informuje, że wszystko przebiegło pomyślnie:

**Core/GeneratedModels:** klasy będące modelami danych, wygenerowane przez Entity Framework na podstawie istniejących modeli w bazie danych

Diagram klas modeli




**Core/Metadata:** klasy partial dla modeli, będące klasami typu Metadata (przechowują dane o danych), zawierają atrybuty określające kolejność serializacji i metody pozwalające wyłączyć serializację właściwości nawigacyjnych. Wszelkie zmiany na modelach muszą być dokonywane w tych klasach, ponieważ klasy wygenerowane przez Entity Framework mogą zostać nadpisane podczas wykonania update modelu w przypadku gdy zmianie ulegnie struktura bazy.

**Core/Repositories:** warstwa abstrakcyjna (interfejsy) wzorca Repozytorium i UnitOfWork

**Extensions:** klasa OwinContextExtensions

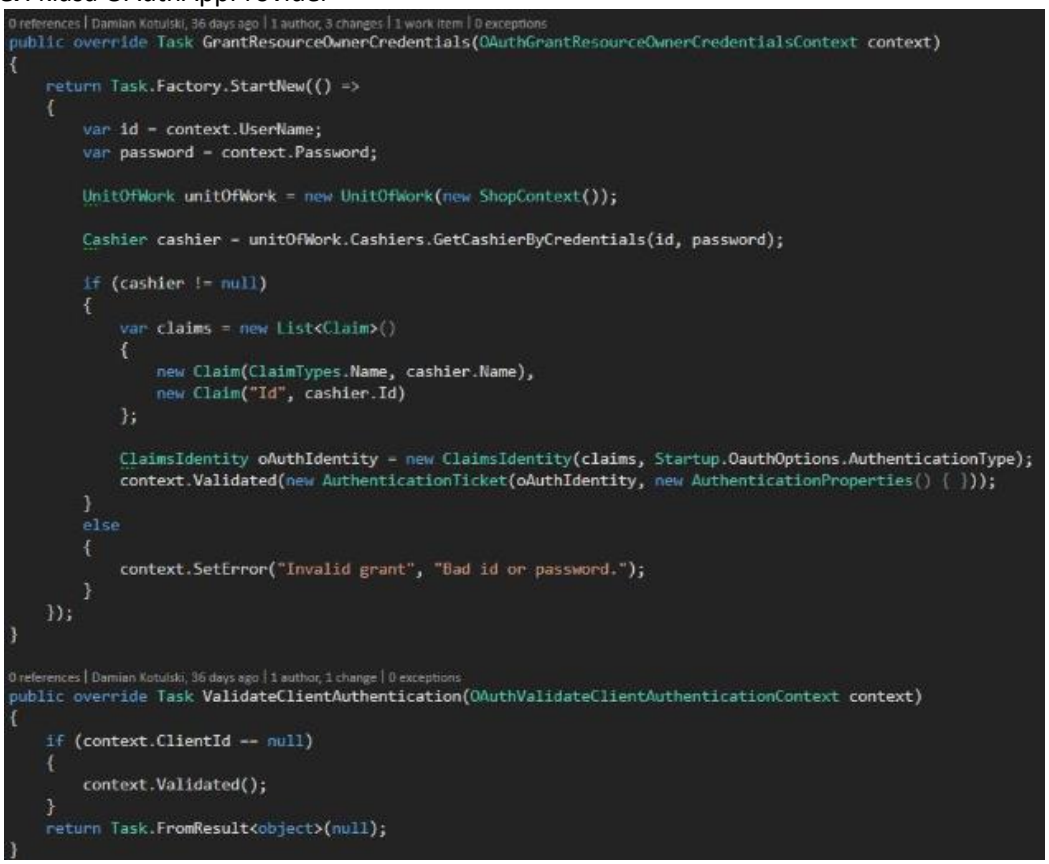
```
0 references | Damian Kotulski, 36 days ago | 1 author, 1 change
public static class OwinContextExtensions
{
    0 references | Damian Kotulski, 36 days ago | 1 author, 1 change | 0 exceptions
    public static string GetUserId(this IOwinContext context)
    {
        var result = "-1";
        var claim = context.Authentication.User.Claims.FirstOrDefault(c => c.Type == "Id");
        if (claim != null)
        {
            result = claim.Value;
        }
        return result;
    }
}
```

- klasa zawierająca metodę rozszerzającą, która obsługuje mechanizm przyznawania uprawnień dostępu do usług WebService'u dla konkretnych użytkowników (kasjerów)

**Persistance/Mappers:** klasy będące modelami danych dla komponentów wymaganych do obługi procedur składowanych z bazy danych

**Persistance/Repositories:** implementacja interfejsów (warstwy abstrakcyjnej) wzorca Repozytorium i UnitOfWork

**Provider**: klasa OAuthAppProvider



```
0 references | Damian Kotulski, 36 days ago | 1 author, 3 changes | 1 work item | 0 exceptions
public override Task GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext context)
{
    return Task.Factory.StartNew(() =>
    {
        var id = context.UserName;
        var password = context.Password;

        UnitOfWork unitOfWork = new UnitOfWork(new ShopContext());

        Cashier cashier = unitOfWork.Cashiers.GetCashierByCredentials(id, password);

        if (cashier != null)
        {
            var claims = new List<Claim>()
            {
                new Claim(ClaimTypes.Name, cashier.Name),
                new Claim("Id", cashier.Id)
            };

            ClaimsIdentity oAuthIdentity = new ClaimsIdentity(claims, Startup.OauthOptions.AuthenticationType);
            context.Validated(new AuthenticationTicket(oAuthIdentity, new AuthenticationProperties() { }));
        }
        else
        {
            context.SetError("Invalid grant", "Bad id or password.");
        }
    });
}

0 references | Damian Kotulski, 36 days ago | 1 author, 1 change | 0 exceptions
public override Task ValidateClientAuthentication(OAuthValidateClientAuthenticationContext context)
{
    if (context.ClientId == null)
    {
        context.Validated();
    }
    return Task.FromResult<object>(null);
}
```

- asynchroniczna obsługa mechanizmy dawania uprawnień do dostępu do usług WebService'u dla kasjera posiadającego aktualny i poprawny token

# 10. SmartShopWpf

## a. Opis projektu

Główny projekt zawierający nasz program. Należą do niego widoki oraz część backendowa porozmieszczana w osobnych klasach. To ten projekt odpowiada za działanie aplikacji.

## b. Najważniejsze kody wraz z komentarzem

a) LoginWindow

```
1 reference
private void btnLogin_Click(object sender, RoutedEventArgs e)
{
    string id = txtLogin.Text;
    string password = pswPassword.Password;
    ProductsClient productsClient = null;
    CashierClient cashierClient = null;
    Cashier cashier = null;
    List<Product> products = new List<Product>();
    string token = "";
    btnLogin.IsEnabled = false;
    Task.Factory.StartNew((Action)delegate { |
    foreach (FileInfo fi in di.GetFiles("PluginLogIn.dll"))
    {
        Assembly pluginAssembly = Assembly.LoadFrom(fi.FullName);
        foreach (Type pluginType in pluginAssembly.GetExportedTypes())
        {
            if (pluginType.GetInterface(typeof(ILogIn).Name) != null)
            {
                ILogIn TypeLoadedFromPlugin = (ILogIn)Activator.CreateInstance(pluginType);
                if (TypeLoadedFromPlugin.CheckLoginData(id, password, ref productsClient, ref cashierClient, ref cashier, ref products, ref token))
                {

                        InitAppData(cashier, products, token);
                        Dispatcher.BeginInvoke(new Action(delegate
                        {

                            MainWindow mW = new MainWindow(false);
                            mW.Show();
                            this.Close();
                        }));
                    }
                else
                {
                    MessageBox.Show("Invalid login or password. Please check the data", "Error", MessageBoxButton.OK, MessageBoxImage.Exclamation);
                }
            }
        }
```

Powyższy event odpowiedzialny jest za poprawne zalogowanie się na użytkownika. Działanie logowania wykonuje sie w pluginie, który zwraca true/false, zależnie od opcji event reaguje przejściem do następnego widoku, błąd błedem. Wczytywanie pluginu jest zrobione asynchroniczne, dla szybszego działania.

```
1 reference
private void pswPassword_PasswordChanged(object sender, RoutedEventArgs e)
{
    if (String.IsNullOrWhiteSpace(pswPassword.Password))
        pswPassword.Tag = "Hasło";
    else
        pswPassword.Tag = "";
}
```

Powyższy event odpowiedzialny jest za poprawne oznaczenie tagiem passwordboxa, zależnie od jego zawartości.

```
1 reference
private void InitAppData(Cashier cashier, List<Product> products, string token)
{
    DataHandler data = DataHandler.GetInstance();
    data.Token = token;

    Cashbox cashbox = new Cashbox();
    cashbox.IdCashbox = 13;
    cashbox.ShopId = 14;
    cashbox.Id = 1;

    data.Cashier = cashier;
    data.Products = products;
    data.Cashbox = cashbox;
}
```

Powyższa metoda służy do inicjalizacji danych początkowych.

b) MainWindow

```
1 reference
private void InitView()
{
    DateTime dateTime = DateTime.UtcNow.Date;
    lblDate.Content = dateTime.ToString("dd/MM/yyyy");

    DataHandler data = DataHandler.GetInstance();

    listVFromListListOfProducts.ItemsSource = data.Products;

    lblCashierNumber.Content = data.Cashier.Id;
    lblCashRegisterNumber.Content = data.Cashbox.Id;

    new Top10Invoker().Download(listVTop10ListTop10);
    new DoneTransactionInvoker().Download(listVTransactions);
}
```

Powyższa metoda służy do inicjalizacji danych  w oknie.

```
1 reference
private void btnEdit_Click(object sender, RoutedEventArgs e)
{
    if (listVBasket.SelectedItems.Count != 1)
    {
        MessageBox.Show("Musisz zaznaczyć jeden produkt");
    }

    else
    {
        Basket basketToEditHisQuantity = (Basket)listVBasket.SelectedItem;
        string tagForManuDisplCode = "Kod produktu";
        string tagForManuDisplQuan = "Ilość";
        bool ifFound = false;
        if (flagToEditQuantity == false)
        {
            lblManuallyTagOfCode.Content = tagForManuDisplQuan;
            btnEdit.Content = "Potwierdz";
            flagToEditQuantity = true;
            btnManuallyAdd.IsEnabled = false;
        }
        else
        {
            if (txtManuallyCodeEntry.Text == "" || txtManuallyCodeEntry.Text == "0")
            {
                MessageBox.Show("Ilosc musi byc wieksza niż 0");
            }
            else
            {
                foreach (var v in listOfBoughtItems)
                {
                    if (v.Number == basketToEditHisQuantity.Number)
                    {
                        //Przekazujemy całą kwotę ze zniżkami
                        float Sum = float.Parse(lblAmount.Content.ToString());

                        //Od sumy odejmujemy cenę Orderu przed zniżką
```

```
            //Od sumy odejmujemy cenę Orderu przed zniżką
                Sum -= v.BeforeDiscount;
                float SinglePrice = 0;

                SinglePrice  = v.BeforeDiscount / v.Count;

            //Ilość z Ręcznej klawiatury
                v.Count = Convert.ToInt32(txtManuallyCodeEntry.Text);

                basketToEditHisQuantity.Count = Convert.ToInt32(txtManuallyCodeEntry.Text);
                v.BeforeDiscount = SinglePrice * v.Count;


               //Do sumy pdodajemy cenę Orderu przed zniżką
               Sum += v.BeforeDiscount;

            //Cena pojdeycznego Orderu = Cenie przed Zniżką
                v.ChoseOptionPrice = v.BeforeDiscount;
                basketToEditHisQuantity.ChoseOptionPrice = basketToEditHisQuantity.BeforeDiscount;
                lblAmount.Content = Math.Round(Convert.ToDouble(Sum), 2);
                lblAmountWithoutDiscount.Content = Math.Round(Convert.ToDouble(Sum), 2);


            ifFound = true;
            flagToEditQuantity = false;
            lblManuallyTagOfCode.Content = tagForManuDisplCode;
            btnEdit.Content = "Edytuj";
            btnManuallyAdd.IsEnabled = true;
            txtManuallyCodeEntry.Text = "";
        }
    }
}
```

Powyższy przycisk służy do edycji ilości przedmiotów w koszyku.

```
1 reference
private void btnDelete_Click(object sender, RoutedEventArgs e)
{
    List<Basket> newList = new List<Basket>();
    foreach (var v in listVBasket.SelectedItems)
    {
        newList.Add((Basket)v);
    }
    float newSuma = float.Parse(lblAmount.Content.ToString());
    float newSumaWithoutDiscounts = float.Parse(lblAmountWithoutDiscount.Content.ToString());
    foreach (var v in newList)
    {
        if (v.OverwallDiscountName != null && v.OverwallDiscountName.Contains("%"))
        {
            float OverwallDiscountName = float.Parse(v.OverwallDiscountName.Remove(v.OverwallDiscountName.Length - 1));
            newSuma -= v.ChoseOptionPrice * (1 - (OverwallDiscountName / 100));
            newSumaWithoutDiscounts -= v.BeforeDiscount;
        }
        else if (v.OverwallDiscountName != null && v.OverwallDiscountName.Contains("zł"))
        {
            float OverwallDiscountName = float.Parse(v.OverwallDiscountName.Remove(v.OverwallDiscountName.Length - 2));
            newSuma = newSuma - v.ChoseOptionPrice;
            newSumaWithoutDiscounts -= v.BeforeDiscount;
        }
        else
        {
            newSuma -= v.ChoseOptionPrice;
            newSumaWithoutDiscounts -= v.BeforeDiscount;
        }
        listOfBoughtItems.Remove(v);
        listVBasket.Items.Remove(v);
        listOfDeletedItems.Add(v);
    }
    lblAmount.Content = Math.Round(Convert.ToDouble(newSuma), 2);
    lblAmountWithoutDiscount.Content = Math.Round(Convert.ToDouble(newSumaWithoutDiscounts), 2);
}
```

Powyższy przycisk służy do usuwania przedmitów z koszyka.

```
1 reference
private void btnLogout_Click(object sender, RoutedEventArgs e)
{
    LoginWindow oL = new LoginWindow();
    oL.Show();
    this.Close();
}
```

Powyższy przycisk służy do wylogowania się i przejścia do LoginWindow

```
1 reference
private void btnFromListAdd_Click(object sender, RoutedEventArgs e)
{
    Product p = (Product)listVFromListListOfProducts.SelectedItem;
    txtManuallyCodeEntry.Text = p.Code.ToString();
    tabService.SelectedItem = tabManually;
}
```

Powyższy przycisk służy do wybierania przedmiotu z listy przedmiotów.

```
1 reference
private void dailyReportBtn_Click(object sender, RoutedEventArgs e)
{
    new DailyReportInvoker().Download();
}

1 reference
private void monthlyReportBtn_Click(object sender, RoutedEventArgs e)
{
    new MonthlyReportInvoker().Download();
}
```

Powyższe metody służą do drukowania raportów.

c) DiscountWindow

```
1 reference | Ola, 15 days ago | 1 author, 2 changes | 2 work items | 0 exceptions
private void btnTypeOfDiscount_Click(object sender, RoutedEventArgs e)
{
    if (flagToTagTypeOfDiscount)
    {
        btnTypeOfDiscount.Content = "zł";
        flagToTagTypeOfDiscount = false;
    }
    else
    {
        btnTypeOfDiscount.Content = "%";
        flagToTagTypeOfDiscount = true;
    }
}
```

Przycisk służy do zmiany flagi, w zależności od tego, jaki typ zniżki wybieramy(procentowy czy kwotowy).

```
private void btnAddDiscount_Click(object sender, RoutedEventArgs e)
{
    double discountValueConverter = 0;

    try { discountValueConverter = Convert.ToDouble(txtDiscount.Text.Trim().ToString()); }
    catch { MessageBox.Show("Zły Format Zniżki!"); }

    float discountValue = (float)discountValueConverter;

    if (discountValue <= 0)
    {
        MessageBox.Show("Zniżka musi być większa niż 0!");
    }
    else
    {
        MainWindow mW = Owner as MainWindow;

        if (MainWindow.flagToTagKindOfDiscount)
        {
            foreach (Basket b in mW.listVBasket.Items)
            {
                b.OverwallDiscountName = discountValue.ToString() + btnTypeOfDiscount.Content;

                MainWindow.flagToOverwallDiscount = true;
            }
            overwallDiscount = discountValue.ToString() + btnTypeOfDiscount.Content;

            if (btnTypeOfDiscount.Content.ToString().Trim() == "%")
            {
                typeOfDis = "%";
                double percent = 100 - discountValue;
                OverwallDicountValue = percent;
                overwallAmountWithDiscount = (float)Math.Round((MainWindow.overwallAmount * percent * 0.01), 2);
                mW.UpdateDiscount();
            }
            else
            {
                typeOfDis = "zl";
                OverwallDicountValue = discountValue;
                overwallAmountWithDiscount = (float)Math.Round((MainWindow.overwallAmount - discountValue), 2);
                mW.UpdateDiscount();
            }
        }
        else
        {
            foreach (Basket b in mW.listVBasket.SelectedItems)
            {
                b.SigleDiscountName = discountValue.ToString() + btnTypeOfDiscount.Content;

                if (btnTypeOfDiscount.Content.ToString().Trim() == "%")
                {
                    double percent = 100 - discountValue;
                    if (b.BeforeDiscount != 0)
                    {
                        b.ChoseOptionPrice = (float)Math.Round((b.BeforeDiscount * percent * 0.01), 2);
                    }
                    else
                    {
                        b.BeforeDiscount = b.ChoseOptionPrice;
                        b.ChoseOptionPrice = (float)Math.Round((b.ChoseOptionPrice * percent * 0.01), 2);
                    }
                }
                else
                {
                    if (b.BeforeDiscount != 0)
                    {
                        b.ChoseOptionPrice = b.BeforeDiscount - (discountValue * b.Count);
                    }
                    else
                    {
                        b.ChoseOptionPrice = b.ChoseOptionPrice - (discountValue * b.Count);
                    }
                }
            }

            mW.UpdateSigleDiscount();
        }

        mW.listVBasket.Items.Refresh();
        this.Close();
    }
}
```

Przycisk odpowiada za dodawanie zniżki, w zależności czy jest to zniżka całościowa czy dla pojedynczego Orderu.

d) PaymentWindow

```
private void btnPay_Click(object sender, RoutedEventArgs e)
{
    MainWindow mW = Owner as MainWindow;

    float price = float.Parse(mW.lblAmount.Content.ToString(), CultureInfo.InvariantCulture.NumberFormat);

    //dataTotalPrice = price / 100;
    dataTotalPrice = (float)Convert.ToDouble(MainWindow.totalPriceToPaymentLabel);

    Receipe recp = new Receipe();
    //recp.TransactionNumber = data.Transaction.Id;
    recp.TransactionNumber = dataId;
    recp.Data = DateTime.Now;
    recp.listOfBoughtProducts = MainWindow.listOfBoughtItems;
    recp.PriceSum = dataTotalPrice;
    if (flagToKindOfPayment)
    {
        recp.kindOfPayment = "Gotowka";
    }
    else
    {
        recp.kindOfPayment = "Karta";
    }

    recp.PriceSum = dataTotalPrice;
    recp.CashNumber = Convert.ToInt32(mW.lblCashRegisterNumber.Content);
    recp.CashierNumber = Convert.ToInt32(mW.lblCashierNumber.Content);
    if (MainWindow.listOfDeletedItems.Count > 0)
        recp.listOfDeletedProducts = MainWindow.listOfDeletedItems;
    ReceipePDFGenerator rPDFGen = new ReceipePDFGenerator(recp);
    rPDFGen.GeneratePDF();

    mW.listVBasket.Items.Clear();
    MainWindow.listOfBoughtItems.Clear();
    MainWindow.listOfDeletedItems.Clear();
    mW.lblAmount.Content = 0;
    mW.lblAmountWithoutDiscount.Content = 0;
    mW.lblTransactionNumber.Content = "";
    MainWindow.flagToOverwallDiscount = false;
    mW.btnEdit.IsEnabled = true;
    mW.btnVat.IsEnabled = true;

    this.Close();
}
```

Przycisk podsumowuje transakcje, drukując paragon .pdf. Po wydrukowaniu paragonu, zostaje wyczyszczona lista zakupów, zniżki, dzięki czemu można swobodnie rozpocząć kolejną transakcje

```
private void btnKinfOfPayment_Click(object sender, RoutedEventArgs e)
{
    if (flagToKindOfPayment)
    {
        btnKinfOfPayment.Content = "Karta";
        flagToKindOfPayment = false;
    }
    else
    {
        btnKinfOfPayment.Content = "Gotówka";
        flagToKindOfPayment = true;
    }
}
```

Przycisk służy do zmiany flagi, w zależności od tego, jaki typ płatności wybieramy.

```
1 reference | Ola, 7 days ago | 1 author, 1 change | 0 exceptions
private void PaymentWindow_Loaded(object sender, RoutedEventArgs e)
{
    var hwnd = new WindowInteropHelper(this).Handle;
    SetWindowLong(hwnd, GWL_STYLE, GetWindowLong(hwnd, GWL_STYLE) & ~WS_SYSMENU);
}
```

Usunięcie możliwości zamknięcia okna Payment, bez podsumowania transakcji. Zapobiega błędom aplikacji.

### e) Asynchronous folder

Folder zawiera klasy odpowiedzialne za asynchroniczne przetwarzanie długotrwałych operacji. Wykorzystaliśmy bibliotekę zadań równoległych (TPL), aby wyeliminować przerwy w możliwości korzystania z graficznego interfejsu. Wszystkie klasy opierają się na schemacie utworzenia zadania (Task) poprzez wbudowaną w system fabrykę, wykonaniu go oraz zwróceniu lub wyświetleniu rezultatu w aplikacji. Przykład kodu:

```
2 references | Damian Kotulski, 17 days ago | 1 author, 2 changes | 2 work items
class Top10Invoker
{
    2 references | Damian Kotulski, 17 days ago | 1 author, 2 changes | 2 work items | 0 exceptions
    public void Download(ListView top10ListView)
    {
        Task.Factory.StartNew<List<BestSellingProduct>>(() =>
        {
            Top10Client top10Client = new Top10Client(DataHandler.GetInstance().Token);
            List<BestSellingProduct> top10 = top10Client.GetTop10Products();
            return top10;

        }).ContinueWith((asct) =>
        {
            top10ListView.Dispatcher.BeginInvoke(new Action(() =>
            {
                top10ListView.ItemsSource = asct.Result;

                Trace.WriteLine("POBRANO TOP10 SPRZEDANYCH PRODUKTÓW");
            }));
        });
    }
}
```

Asynchroniczne pobranie i wyświetlenie 10 najczęściej sprzedawanych produktów.

f) Data folder

```
16 references
public class Receipe
{
    public const string NAME = "SMARTSHOP";
    public const string ADDRESS = "Katowice, ul. Mariusza Cebuli 8";
    6 references
    public string kindOfPayment { get; set; }
    6 references
    public int TransactionNumber { get; set; }
    4 references
    public DateTime Data { get; set; }
    2 references
    public List<Basket> listOfBoughtProducts { get; set; }
    4 references
    public List<Basket> listOfDeletedProducts { get; set; }
    2 references
    public List<ReturnObject> listOfAllOrdersInTransactionToReturn { get; set; }
    5 references
    public List<ReturnObject> listOfReturnsOrders { get; set; }
    11 references
    public float PriceSum { get; set; }
    5 references
    public float PriceToReturn { get; set; }
    0 references
    public string OverDiscount { get; set; }
    2 references
    public int CashNumber { get; set; }
    2 references
    public int CashierNumber { get; set; }
}
```

Powyższa klasa służy do stworzenia rachunku, zbiera wszystkie potrzebne informacje z przedmiotów w koszyku, które są potrzebne do wydrukowania paragonu.

g) ReceipeMethods folder

Folder składa się z czterech klas odpowiedzialnych za drukowanie: Paragonu, Zwrotu, Dniowego oraz Miesiecznego raportu. Klasy te między sobą różnią się tylko drukowanymi informacjami. Ze względu na obszerność kodu nie umieszczamy go tutaj. Pdf'y tworzy sie jako pustą stronę. Nastepnie po określeniu rozmiaru strony, możemy drukować przy pomocy ustawiania współrzędnych X oraz Y. Strony są dodawane dynamicznie, jeśli zajdzie taka potrzeba, czyli spełni się warunek, że współrzędna Y przekroczyła limit.

# 11. Podział obowiązków

## a. Tomasz Szostak

1. Administracja serwera VPS.
2. Instalacja, konfiguracja i administracja serwera baz danych MariaDB.
3. Stworzenie modelu bazy danych.
4. Opracowanie procedur na potrzeby programów.

## b. Krzysztof Kurkiewicz

1. Obsługa przycisków Edytuj/Usuń/Wyloguj w MainWindow
2. Wyszukiwanie produktow z listy w MainWindow
3. Generowanie paragonów/raportów w formacie .pdf
4. Plugin MockLogowania, a następnie przypisanie nowego pluginu PlugLogIn
5. Użycie pluginów logowania w oknie logowania
6. Zapewnienie asynchronicznego działania pewnych funkcji
7. Wykonanie UnitTestow
8. Wykonanie logo dla projektu
9. Wyodrębnienie modeli do SmartShop.Models

## c. Aleksandra Miękina

1. Wykonanie całego widoku aplikacji. (Okna, kontrolki, hinty, triggery, taby).
2. Dodawanie produktów do koszyka i określanie ich ilości, za pomocą kodu produktu. Obsługa przycisków 0-9, „Dodaj”, „Cofnij”, „Wyczyść”.
3. Wyświetlanie dodanych do koszyka produktów, na głównej liście.
4. Wyświetlenie łącznej kwoty, za produkty które dodaliśmy do koszyka. Kwota aktualizuje się na bieżąco.
5. Obsługa przycisku Vat.
6. Obsługa przycisku Zniżka.
7. Obsługa zwrotów wraz z wydrukiem poprawionego paragonu w formacie .pdf

## d. Damian Kotulski

1. Organizowanie pracy zespołu, tworzenie zadań i nadzorowanie rozwoju projektu na platformie github
2. Konfiguracja, zarządzanie i wdrożenie zespołu w system kontroli wersji git
3. Utworzenie WebService’u:
    - stworzenie modeli danych odpowiadających tabelą w bazie danych
    - implementacja wzorców Repozytorium i Unit of Work
    - stworzenie kontrolerów obsługujących metody http spełniające założenia aplikacji
    - obsługa procedur składowanych w bazie danych
    - implementacja standardu autoryzacji OAuth 2.0
4. Utworzenie modułu odpowiedzialnego za komunikację aplikacji głównej z WebService’m

5. Wyświetlanie listy dostępnych produktów w aplikacji głównej
6. Implementacja mechanizmu odpowiedzialnego za obsługę transakcji - tworzenie, aktualizowanie, dodawanie do bazy
7. Wyświetlanie wszystkich przeprowadzonych transakcji w aplikacji głównej
8. Wyświetlanie top10 produktów w aplikacji głównej
9. Implementacja asynchronicznej komunikacji aplikacji głównej z WebService'm