|  |  |  |
| --- | --- | --- |
| [Politechnika ¦l±ska](http://www.polsl.pl/) | Politechnika Śląska  Wydział Inżynierii Materiałowej  i Metalurgii | |
| Temat:  Projekt: SmartShop  Dokumentacja | | |
| Komentarze: | | |
| Data:  Podpis: | | Aleksandra Miękina,  Tomasz Szostak,  Damian Kotulski,  Krzysztof Kurkiewicz |
| IPP-30PP |

Table of Contents

[1. Założenia projektu//Krzysztof 4](#_Toc485571274)

[2. Wykorzystane technologie//Krzysztof 4](#_Toc485571275)

[a. Spis i opis technologii//Krzysztof 4](#_Toc485571276)

[b. Wtyczki wspomagające //Krzysztof 4](#_Toc485571277)

[3. Diagramy 4](#_Toc485571278)

[a. Diagram przypadków użycia//Krzysztof 4](#_Toc485571279)

[b. Diagram klas programu//Aleksandra 4](#_Toc485571280)

[c. Diagram encji bazy danych//Tomasz 4](#_Toc485571281)

[4. Baza Danych//Tomasz 4](#_Toc485571282)

[a. Opis//Tomasz 4](#_Toc485571283)

[b. Opis procedur//Tomasz 4](#_Toc485571284)

[5. PluginLogIn//Krzysztof 4](#_Toc485571285)

[a. Opis projektu//Krzysztof 4](#_Toc485571286)

[b. Najważniejsze kody wraz z komentarzem//Krzysztof 4](#_Toc485571287)

[6. SmartShop.CommunicateToWebService//Damian 4](#_Toc485571288)

[a. Opis projektu//Damian 4](#_Toc485571289)

[b. Najważniejsze kody wraz z komentarzem//Damian 5](#_Toc485571290)

[7. SmartShop.Models//Krzysztof 5](#_Toc485571291)

[a. Opis projektu//Krzysztof 5](#_Toc485571292)

[b. Najważniejsze kody wraz z komentarzem//Krzysztof 5](#_Toc485571293)

[8. SmartShop.Tests//Krzysztof 5](#_Toc485571294)

[a. Opis projektu//Krzysztof 5](#_Toc485571295)

[b. Najważniejsze kody wraz z komentarzem//Krzysztof 5](#_Toc485571296)

[9. SmartShopWebApp//Damian 5](#_Toc485571297)

[a. Opis projektu//Damian 5](#_Toc485571298)

[b. Najważniejsze kody wraz z komentarzem//Damian 5](#_Toc485571299)

[10. SmartShopWpf//Krzysztof + Ola 5](#_Toc485571300)

[a. Opis projektu//Krzysztof 5](#_Toc485571301)

[b. Najważniejsze kody wraz z komentarzem//Krzysztofa+Ola 5](#_Toc485571302)

[11. Podział obowiązków//Sami sie dopisujemy 5](#_Toc485571303)

[a. Tomasz Szostak 5](#_Toc485571304)

[b. Krzysztof Kurkiewicz 5](#_Toc485571305)

[c. Aleksandra Miękina 6](#_Toc485571306)

[d. Damian Kotulski 6](#_Toc485571307)

# Założenia projektu//Krzysztof

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:

1. możliwość logowania się na własne, spersonalizowane konto
2. możliwość wybrania produktu z listy wszystkich produktów
3. możliwość wpisania kodu danego produktu(imitacja skanowania kodów kreskowych)
4. możliwość wpisywania i edycji ilości sztuk zakupionego przedmiotu
5. możliwość usuwania przedmiotów z listy zakupionych
6. możliwość dokonywania zwrotów
7. możliwość drukawania paragonów
8. możliwość drukowania raportów w wybranym odstępie czasowym
9. możliwość nadawania zniżek całościowych oraz pojedyńczych, złotówkowych oraz procentowych
10. możliwość zdjęcia vat z produktu
11. możliwość przeglądania wszystkich transakcji

# Wykorzystane technologie//Krzysztof

# Spis i opis technologii//Krzysztof

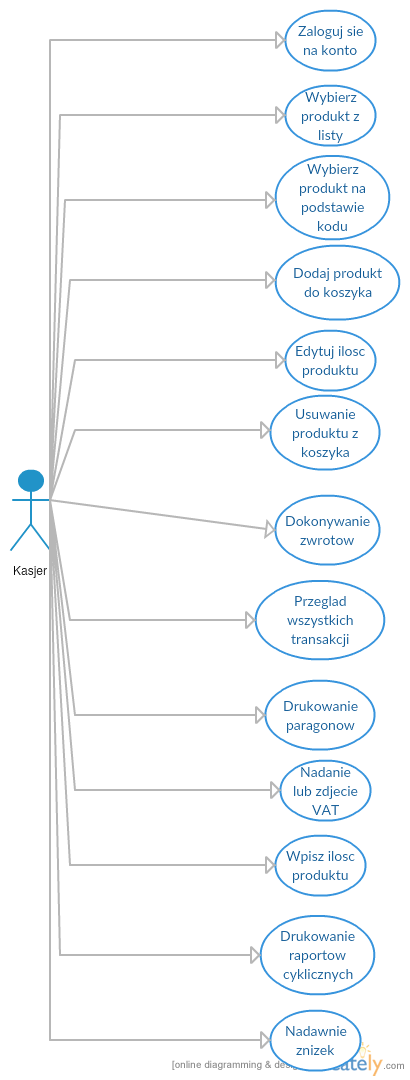
* + 1. Baza danych – MariaDB postawiona na zewnętrznych serwerze
    2. Środowisko do połączenią się z bazą - HeidiSQL
    3. Aplikacja kliencka – Windows Presentation Foundation w języku C#
    4. Webservice – w architekturze REST, napisany w języku C# w aplikacji typu ASP.NET MVC, użyty ORM to Entity Framework
    5. Środowisko – Microsoft Visual Studio 2015

# Wtyczki wspomagające //Krzysztof

* + 1. MySQL.Data oraz MySql.Data.Entity.EF6 – wtyczki pozwalające na połączenie technologii .NET z bazą napisaną na silniku MySQL
    2. Newtonsoft.Json – wtyczka pozwalająca na serializacje i deserializacje informacji w formacie Json
    3. NUnit 3 – wtyczka pozwalające na pisanie testów jednostkowych
    4. Moq – wtyczka pozwalająca na mokowanie zewnętrznych zależności do testów
    5. RestSharp – wtyczka ułatwiająca połączenie z webservicem o architekturze REST
    6. Owin – wtyczka do wymuszenia autoryzacji na bazie tokenu w kontrolerach
    7. CodeMaid – wtyczka pozwalająca zachować czystość i zwięzłość w kodzie
    8. ReSharper – wtyczka rozszerzająca możliwości Visual Studio

# Diagramy

# Diagram przypadków użycia//Krzysztof



# Diagram klas programu//Aleksandra

# Diagram encji bazy danych//Tomasz

# Baza Danych//Tomasz

# Opis//Tomasz

# Opis procedur//Tomasz

# PluginLogIn//Krzysztof

# Opis projektu//Krzysztof

# Najważniejsze kody wraz z komentarzem//Krzysztof

# SmartShop.CommunicateToWebService//Damian

# Opis projektu//Damian

# Najważniejsze kody wraz z komentarzem//Damian

# SmartShop.Models//Krzysztof

# Opis projektu//Krzysztof

# Najważniejsze kody wraz z komentarzem//Krzysztof

# SmartShop.Tests//Krzysztof

# Opis projektu//Krzysztof

# Najważniejsze kody wraz z komentarzem//Krzysztof

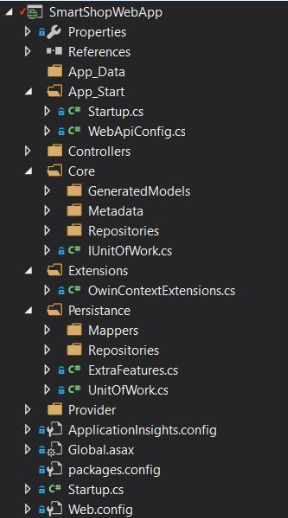
# SmartShopWebApp

# 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.

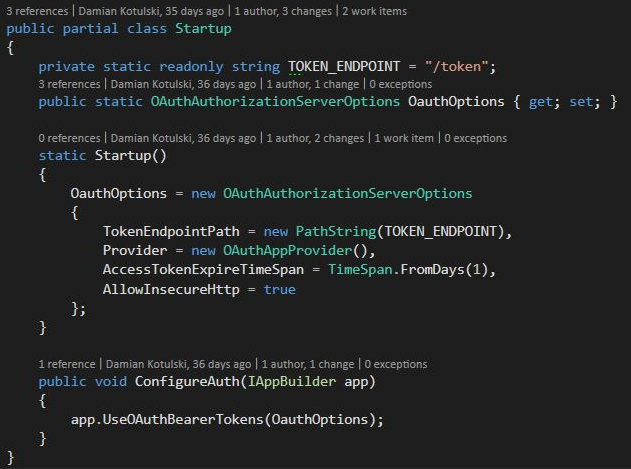
# Najważniejsze kody wraz z komentarzem

* Struktura projektu



* Omówienie najważniejszych komponentów

**App\_Start:** klasa Startup.cs



- 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:

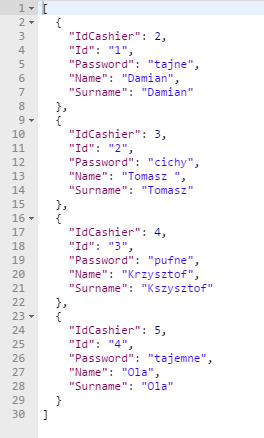
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Pobieranie  GET | Dodawanie  POST | Modyfikacja  PUT | Usuwanie  DELETE |
| Shop | X  (w Cashboxes) | X | X | X |
| Cashboxes | **TAK** | X | X | X |
| Cashiers | TAK | X | TAK  (hasło) | X |
| Transactions | TAK | TAK | TAK | X |
| Orders | TAK |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

Cashiers

* GET

**ENDPOINT**: /api/cashiers

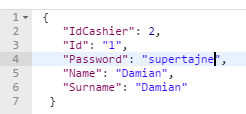
PRZYKŁADOWA ODPOWIEDŹ:



* PUT

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

PRZYKŁADOWE BODY:



ODPOWIEDŹ:

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

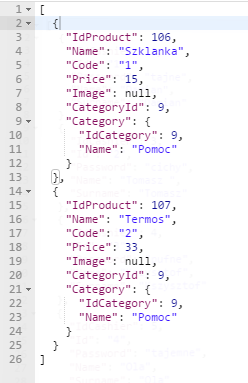


Products

* GET

**ENDPOINT**: /api/products

PRZYKŁADOWA ODPOWIEDŹ:



Cashboxes

* GET

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

PRZYKŁADOWA ODPOWIEDŹ:



Orders

* GET

**ENDPOINT**: /api/orders

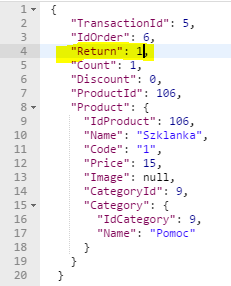
PRZYKŁADOWA ODPOWIEDŹ:



* PUT

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

PRZYKŁADOWE BODY: (zwracamy produkt)



ODPOWIEDŹ:

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

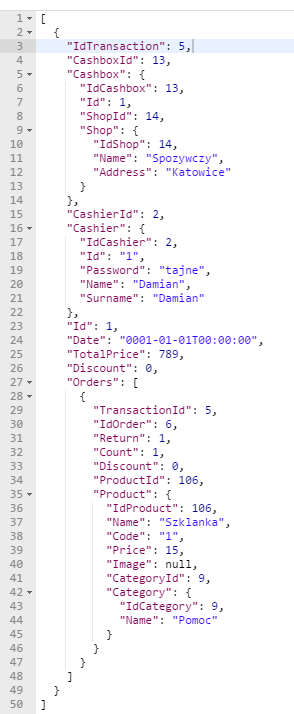


Transactions

* GET

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

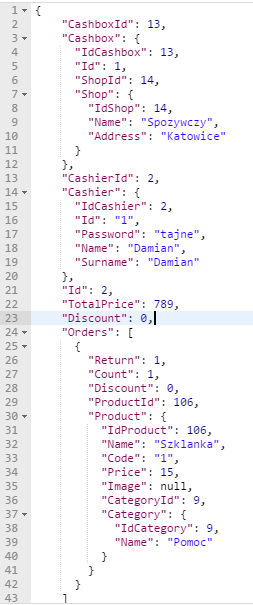
PRZYKŁADOWA ODPOWIEDŹ:



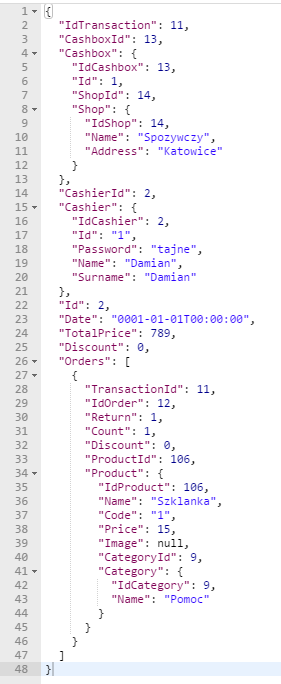
* POST

**ENDPOINT**: /api/transactions

PRZYKŁADOWE BODY: (nie ma idtransaction i idorder)



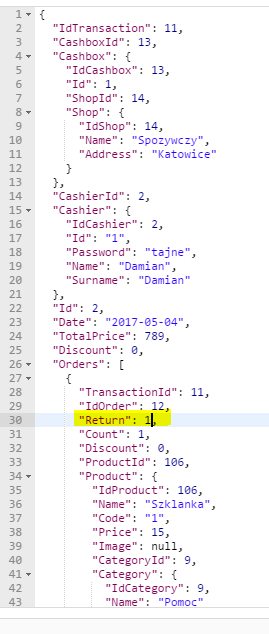
PRZYKŁADOWA ODPOWIEDŹ: (dostajemy odpowiedź z wygenerowanym idTransaction i idOrder dla każdego z zamówienia)



* PUT

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

PRZYKŁADOWE BODY: (zwracamy produkt)



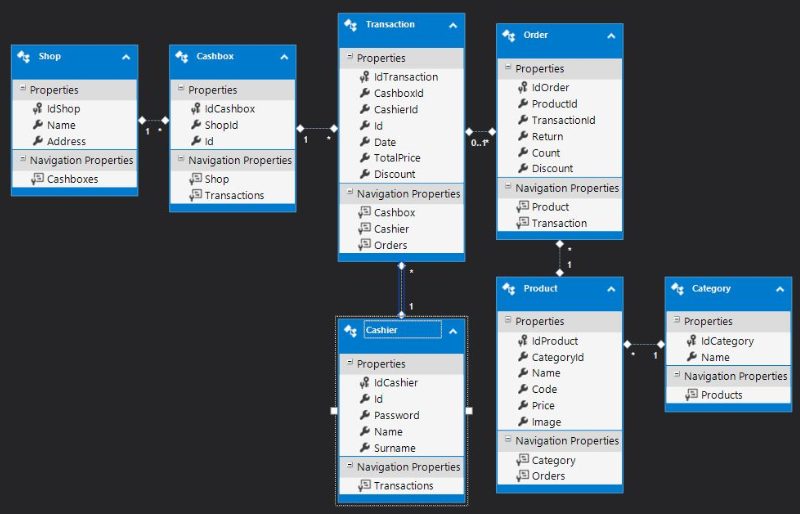
ODPOWIEDŹ:

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



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

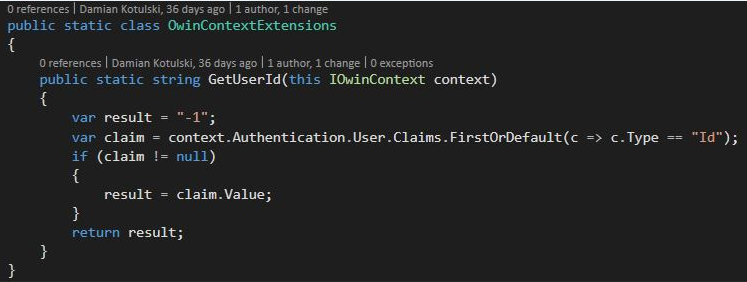
Diagram klas modeli

****

**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:** klasaOwinContextExtensions

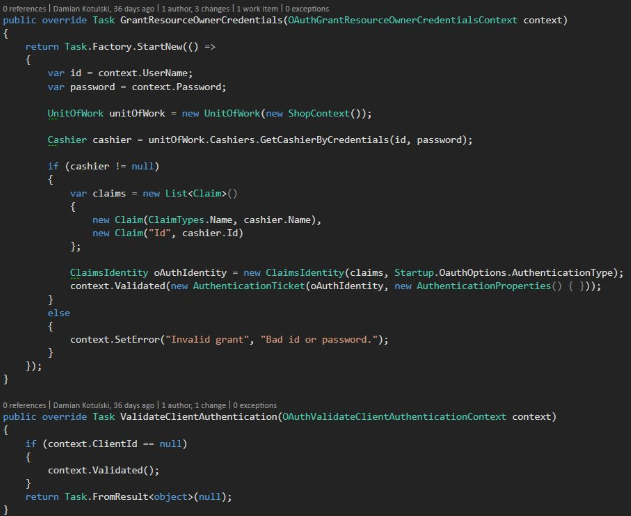


**-** klasa zawierająca metodę rozszerzającą, która obsługuje mechanizm przyznawania uprawnień dostępu do usług WebService’u dla konkretnych użytkowników (kasjerów)

**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



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

# SmartShopWpf//Krzysztof + Ola

# Opis projektu//Krzysztof

# Najważniejsze kody wraz z komentarzem//Krzysztofa+Ola

# Podział obowiązków//Sami sie dopisujemy

# Tomasz Szostak

# Krzysztof Kurkiewicz

# Aleksandra Miękina

# 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