|  |
| --- |
| WYDZIAŁ INFORMATYKI POLITECHNIKI BIAŁOSTOCKIEJ    ROZPROSZONE SYSTEMY INTERNETOWE |

**Projekt z użyciem SOAP WS**

**Temat: System rezerwacji biletów kolejowych**

Białystok, 4.05.2020

|  |  |
| --- | --- |
| Wykonujący:  Adam Bajguz  Michał Kierzkowski  Grupa PS3 | Prowadzący:  dr hab. inż. Jacek Grekow |

Spis treści

[1.Wprowadzanie 3](#_Toc39325989)

[1.1.Zakres projektowy 3](#_Toc39325990)

[1.2.Funkcjonalności 3](#_Toc39325991)

[1.3. Definicje, Akronimy, Skróty 3](#_Toc39325992)

[1.4. Użyte technologie 4](#_Toc39325993)

[1.4.1 Po stronie serwera 4](#_Toc39325994)

[1.4.2 Po stronie klienta 4](#_Toc39325995)

[2. Specyfikacja usługi Web Service 4](#_Toc39325996)

[2.1 Standardy sieciowe 4](#_Toc39325997)

[2.2. Specyfikacja WSDL 5](#_Toc39325998)

[2.3. Dostępne usługi 5](#_Toc39325999)

[2.4. Opis wybranych usług 5](#_Toc39326000)

[2.4.1. Operacja ... 5](#_Toc39326001)

[2.4.2. Operacja … 5](#_Toc39326002)

[2.4.3. Operacja … 5](#_Toc39326003)

[2.4.4. Operacja … 6](#_Toc39326004)

[3. Specyfikacja usługi aplikacji klienckiej 6](#_Toc39326005)

[4.Instrukcja użytkownika 6](#_Toc39326006)

[4.1 Instrukcja do Web Service 6](#_Toc39326007)

[4.2 Instrukcja do Aplikacji klienckiej 6](#_Toc39326008)

[4.3 Ograniczenia dla usługi WS 6](#_Toc39326009)

# 1.Wprowadzanie

Celem tego dokumentu jest przedstawienie funkcjonalności, technologii oraz zasad działania(w tym wymiany informacji) w usłudze sieciowej systemu rezerwacji biletów kolejowych oraz interfejsowym systemie klienckim.

## 1.1.Zakres projektowy

Zakresem projektu było stworzenie systemy, który pozwoli na przechowywanie bazy stacji kolejowych i tras, na jakich przemieszczają się pociągi, a także umożliwi zakup biletów na daną trasę oraz pobranie go w formacie PDF. W tym celu należało stworzyć dwie aplikacje jedną, jako Web Service dostarczający wszystkie funkcjonalności w API, zaś drugą, jako aplikacje kliencką umożliwiającą łatwy dostęp do zasobów serwisowych.

## 1.2.Funkcjonalności

System posiada funkcjonalności podstawowe takie jak:

* rejestracja klienta;
* logowanie na serwer;
* wyświetlanie bazy tras kolejowych (Miasto od , Miasto do, dzień, godzina);
* wyszukiwarkę tras;
* kupno biletu;
* odbiór potwierdzenia kupna w formacie PDF;
* sprawdzenie rezerwacji na podstawie podanego numeru biletu;
* dodawanie, usuwanie, modyfikowanie tras oraz stacji (funkcjonalność dostępna na swagger tylko dla użytkowników z prawami administratora)

a także funkcjonalności dodatkowe:

* przypominanie hasła na podany podczas rejestracji adres email;
* szczegóły danej stacji kolejowej;
* prezentowanie na mapie położenia stacji;
* autoryzacja oparta na JSON Web Token.

## 1.3. Definicje, Akronimy, Skróty

**SOAP** - (ang. Simple Object Access Protocol) jest protokołem bazującym na standardzie XML. Protokół SOAP pozwala aplikacjom na komunikację przez Internet. Standard SOAP jest opracowywany przez W3C

**W3C** – (ang. The World Wide Web Consortium) - organizacja zajmująca się ustanawianiem standardów dla stron WWW. Publikowane przez W3C rekomendacje nie mają mocy prawnej, nakazującej ich użycie, lecz wskazują standardy dla rozwiązań technologicznych.

**WS, Web Service** – (ang. usługa sieciowa). Podstawowa technologia wykorzystywana w architekturze SOA. Jest to usługa systemu informatycznego, która może być używana przez inne komponenty programowe, również komponenty zewnętrzne projektowanego systemu. Web Service w paradygmacie SOA stanowi najistotniejszy interfejs (ang. API – application programming interface) projektowanego systemu. Web Service jest również wykorzystywany wewnętrznie do komunikacji pomiędzy komponentami systemu.

**WSDL** - (ang. Web Services Description Language). Plik WSDL to dokument XML, opisujący zbiór komunikatów SOAP.

**XML** - (ang. Extensible Markup Language, - Rozszerzalny Język Znaczników) to uniwersalny język formalny przeznaczony do reprezentowania różnych danych w ustrukturalizowany sposób. XML jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami i rekomendowany oraz specyfikowany przez organizację W3C.

**SSL** - (Secure Sockets Layer) jest protokołem do zabezpieczania komunikacji odbywającej się w Internecie. Dzięki niemu, wszystkie informacje wysyłane pomiędzy przeglądarką internetową a serwerem są szyfrowane. Aby z niego skorzystać, wymagana jest instalacja certyfikatu SSL.

**HTTP Strict Transport Security** (**HSTS**) – mechanizm bezpieczeństwa sieci, który chroni strony przed atakami takimi, jak wymuszone zmniejszenie poziomu protokołu oraz przechwytywanie sesji. Dzięki niemu do serwerów można połączyć się tylko za pomocą przeglądarek, korzystających z bezpiecznych połączeń HTTPS, natomiast nigdy nie dopuszcza on połączeń na bazie niezabezpieczonego protokołu HTTP. HSTS jest uznawany za standardowy protokół przez IETF; został opisany w dokumencie RFC 6797 .

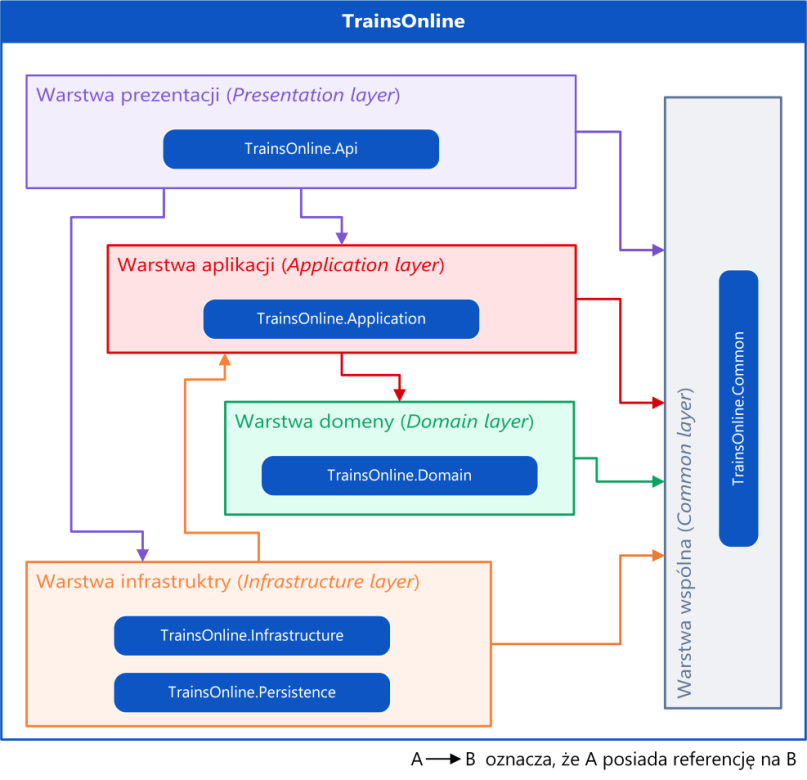
**JSON Web Token** - to internetowy standard do tworzenia, tokenów dostępu opartych na JSON. Tokeny są podpisywane przy użyciu prywatnego tajnego klucza lub klucza publicznego / prywatnego. Na przykład serwer może wygenerować token z roszczeniem „zalogowany, jako administrator” i przekazać go klientowi. Klient może następnie użyć tego tokena, aby udowodnić, że jest zalogowany, jako administrator.

## 1.4. Użyte technologie

### Po stronie serwera

Moduł TrainsOnline został zaimplementowany z użyciem podejścia określnego jako Clean Architecture, którego celem jest zapewnienie wysokiej skalowalności aplikacji i uniknięcie tworzenia monolitycznej architektury. Podejście to zrealizowano przy pomocy wzorca architektonicznego Domain Driven Design (DDD). Do stworzenia serwera aplikacji użyto: C# 8.0, NET Core 3.1.0, ASP.NET Core MVC, Microsoft SQL Server 2017. W implementacji zastosowano także następujące biblioteki:

|  |  |
| --- | --- |
| **Nazwa** | **Wersja** |
| AutoMapper.Extensions.Microsoft.DependencyInjection | 7.0.0 |
| coverlet.msbuild | 2.7.0 |
| FluentValidation.AspNetCore | 8.6.0 |
| MediatR | 7.0.0 |
| MediatR.Extensions.Microsoft.DependencyInjection | 7.0.0 |
| Microsoft.AspNetCore | 2.2.0 |
| Microsoft.AspNetCore.Authentication.JwtBearer | 3.1.0 |
| Microsoft.AspNetCore.Mvc | 2.2.0 |
| Microsoft.EntityFrameworkCore | 3.1.0 |
| Microsoft.EntityFrameworkCore.Design | 3.1.0 |
| Microsoft.EntityFrameworkCore.InMemory | 3.1.0 |
| Microsoft.EntityFrameworkCore.SqlServer | 3.1.0 |
| Microsoft.EntityFrameworkCore.Tools | 3.1.0 |
| Microsoft.Extensions.CommandLineUtils | 1.1.1 |
| Microsoft.Extensions.Configuration.EnvironmentVariables | 3.1.0 |
| Microsoft.Extensions.Configuration.Json | 3.1.0 |
| Microsoft.NET.Test.Sdk | 16.4.0 |
| Sentry.Serilog | 1.2.0 |
| Serilog.AspNetCore | 3.2.0 |
| Serilog.Enrichers.Memory | 1.0.4 |
| Serilog.Enrichers.Process | 2.0.1 |
| Serilog.Enrichers.Thread | 3.1.0 |
| Serilog.Sinks.Async | 1.4.0 |
| Serilog.Sinks.Console | 3.1.1 |
| Serilog.Sinks.File | 4.1.0 |
| Shouldly | 3.0.2 |
| Swashbuckle.AspNetCore | 5.0.0-rc4 |
| Swashbuckle.AspNetCore.Annotations | 5.0.0-rc4 |
| System.IdentityModel.Tokens.Jwt | 5.6.0 |
| xunit | 2.4.1 |
| xunit.runner.visualstudio | 2.4.1 |



Wzorzec DDD wymaga również podziału projektu na następujące warstwy:

1. wspólną – obejmuje wszystkie kwestie przekrojowe, czyli klasy i interfejsy wspólne dla różnych warstw;
2. prezentacji – zawiera kontrolery, punkty dostępowe do REST API oraz aplikację Swagger;
3. aplikacji – zawiera całą logikę. Warstwa jest zależna wyłącznie od warstwy domeny, są w niej zdefiniowane interfejsy, np. repozytorium lub jednostki pracy (Unit of Work), których implementacja znajduje się w innych warstwach. Oznacza to, że DDD wykorzystuje wzorzec odwrócenia sterowania (inversion of control). Przykładowo, jeśli aplikacja musi uzyskać dostęp do usługi powiadomień, do warstwy aplikacji zostanie dodany nowy interfejs, a implementacja zostanie utworzona w ramach warstwy infrastruktury;
4. domeny – zawiera dane, wyliczenia, wyjątki i logikę specyficzną dla domeny;
5. infrastruktury – zawiera klasy dostępu do zasobów zewnętrznych, takich jak: baza danych, systemy plików i poczta elektroniczna. Klasy te opierają się na interfejsach zdefiniowanych w warstwie aplikacji. W warstwie znajduje się również podwarstwa Persistence, w której zawarta jest cała konfiguracja bazy danych.

W projekcie z uwagi na małą liczbę obiektów domenowych, a tym samym tabel w bazie danych, zastosowano podejście grupowania klas w:

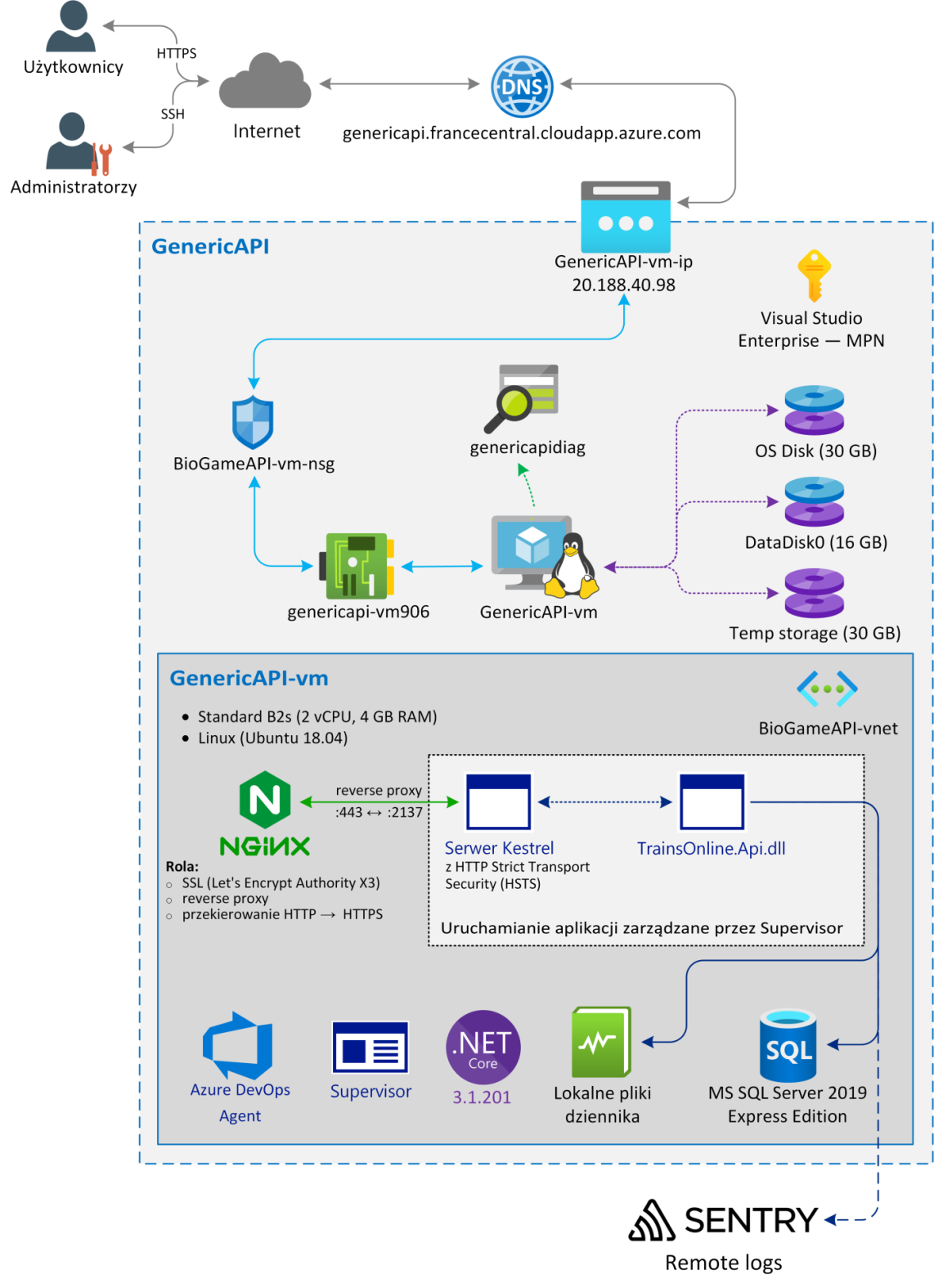
1. projekty (biblioteki) w oparciu o warstwy określone przez DDD (package by layer),
2. funkcjonalności (package by feature) na poziomie folderów i przestrzeni nazw w poszczególnych warstwach.

Rozwiązanie te jest wystarczające do zapewnienia czystego kodu oraz architektury w przypadku tego modułu. Ponadto nie wymaga implementowania m.in. koncepcji ograniczonych kontekstów (bounded contexts).

Moduł TrainsOnline został wdrożony na platformie Azure na maszynie wirtualnej typu Standard B2s, wyposażonej w dwa wirtualne procesory oraz 4 GB pamięci operacyjnej. Do maszyny podpięte są dwa dyski SSD typu Premium SSD, charakteryzujące się maksymalną liczbą operacji na sekundę (IOPS) równą 120 i przepływnością maksymalną na poziomie 25 MB/s, oraz jeden dysk tymczasowy o maksymalnej wartości IOPS równej 1600 i maksymalnej przepływności 15 MB/s :

* + OS Disk – dysk systemowy o pojemności 30 GB z systemem operacyjnym Ubuntu w wersji 18.04 oraz aplikacjami: .NET Core 3.1.0 Runtime, Microsoft SQL Server 2017 Express Edition, nginx, Azure DevOps Agent, supervizor;
  + DataDisk0 – dysk o pojemności 16 GB zawierający aplikację ‘’’oraz pliki dziennika aplikacji, tzw. logi;
  + Temp storage – dysk o pojemności 30 GB służący do przechowywania danych tymczasowych, np. plików tymczasowych serwera SQL, który ulega wyczyszczeniu m.in. po ponownym uruchomieniu maszyny wirtualnej.

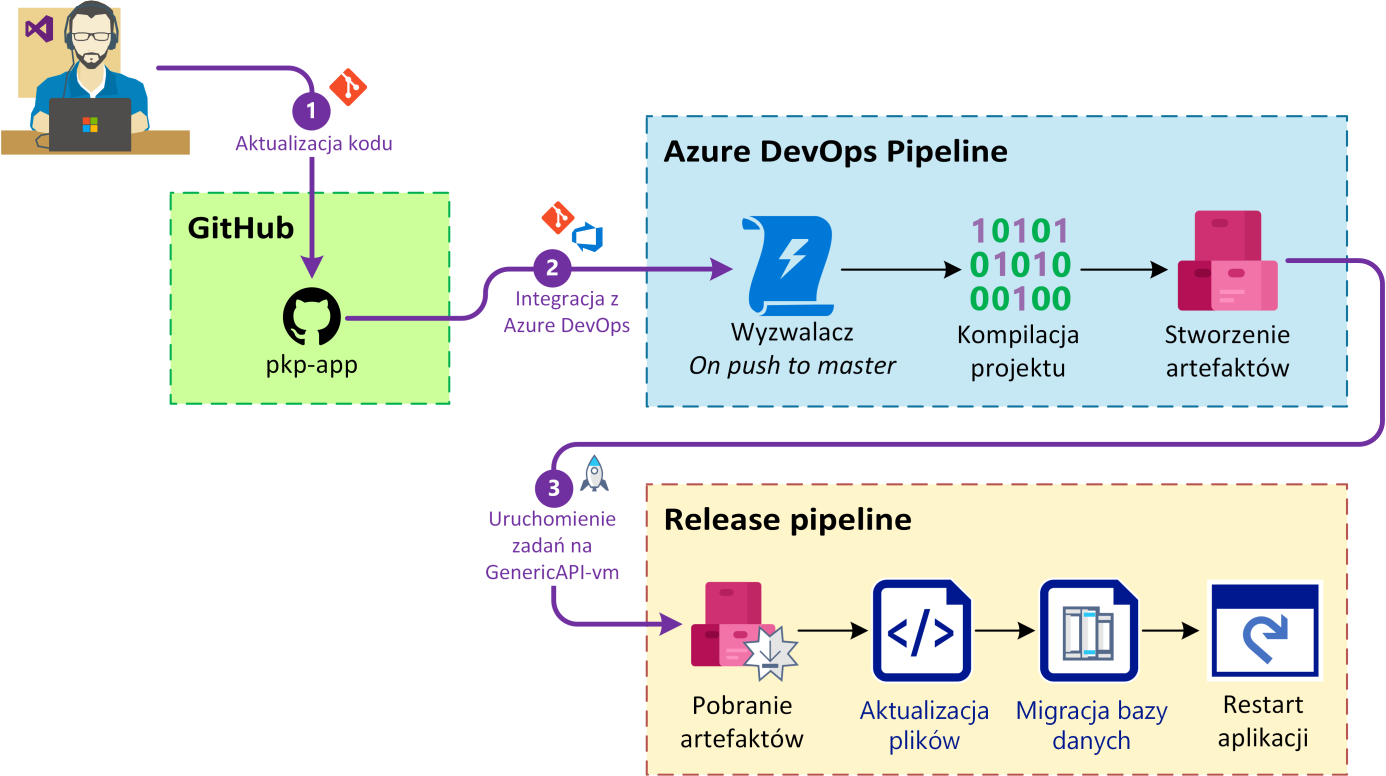
Zdalny dostęp do maszyny wirtualnej odbywa się poprzez protokół Secure Shell (SSH) z wykorzystaniem klucza RSA jako metody uwierzytelniania. Do połączenia się z maszyną wirtualną wykorzystano program Termius z pakietu GitHub Student Developer Pack. Dostarcza on wieloplatformowy terminal ze zintegrowanym klientem SSH. Dostęp przez użytkowników (klientów) do API odbywa się za pośrednictwem protokołu Hypertext Transfer Protocol Secure (HTTPS).Działanie aplikacji można sprawdzić pod adresem https://genericapi.francecentral.cloudapp.azure.com/soap-api. Certyfikat na potrzeby protokołu HTTPS uzyskano za pomocą programu Certbot, urzędem certyfikacji jest Let's Encrypt Authority X3. Aplikacja TrainsOnline uruchomiona jest na porcie 2137 za pomocą serwera Kestrel, który został użyty jako serwer graniczny. Komunikacja aplikacji z Internetem odbywa się za pomocą serwera nginx skonfigurowanego jako reverse proxy pomiędzy portami o numerach 443 oraz 2137. Aplikacja komunikuje się również z platformą Sentry, na której zapisywane są wszelkie zdarzenia błędów.



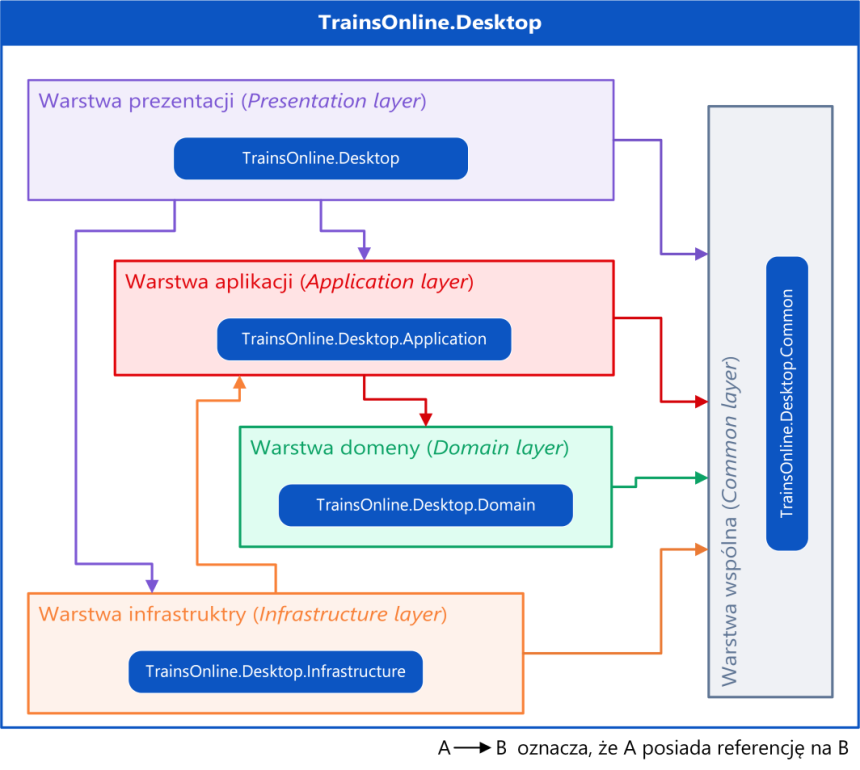
Zastosowanie ciągłej integracji (continuous integration) oraz ciągłego dostarczania (continuous delivery) umożliwiło łatwiejszą obsługę środowiska produkcyjnego TrainsOnline. Aktualizacje plików wykonywalnych, a także modyfikacje struktury bazy danych (migracje) odbywają się przy pomocy dwóch potoków:

1. uruchamianego na Azure DevOps w momencie pojawienia się zmian w gałęzi master repozytorium pkp-app, który buduje aplikację oraz przygotowuje paczkę z artefaktami;

2. uruchamianego na maszynie wirtualnej z poziomu Azure DevOps (z udziałem programu Azure DevOps Agent), którego zadaniem jest m.in. pobranie paczki artefaktów i aktualizacja plików.



### 1.4.2 Po stronie klienta

 Aplikacja TrainsOnline.Desktop jest aplikacją **UWP** zawierającą całą logikę niezbędną do komunikacji z serwerem rezerwacji biletów kolejowych. Technologia UWP została zastosowana w celu implementacji łatwego i przyjaznego dla użytkownika interfejsu okienkowego o spójnym wyglądzie. Architektura aplikacji wykorzystuje wzorzec **MVVM** w celu zapewnienia oddzielenia interfejsu użytkownika od logiki biznesowej. Do implementacji MVVM zastosowano bibliotekę Caliburn.Micro. Ponadto w implementacji zastosowano podejście DDD – podobnie jak w aplikacji serwerowej. Jedyną różnicą jest brak warstwy Persistence.

# 2. Specyfikacja usługi Web Service

## https://media.discordapp.net/attachments/682660849017552947/706230909598433350/unknown.png?width=735&height=6362.1 Standardy sieciowe

Usługa „TrainsOnline” zaimplementowana została jako usługa sieciowa (Web Service) z użyciem protokołu SOAP. Usługa dostępna jest poprzez protokół HTTPS . Usługa dostępna jest pod adresem: https://genericapi.francecentral.cloudapp.azure.com/soap-api , który prowadzi do strony zawierającej wszelkie dostępne na serwerze adresy, w tym adresy serwisów i ich opisów za pomocą WSDL.

## 2.2. Specyfikacja WSDL

WSDL serwisu obsługującego autoryzację ma 145 linii, serwisu tras kolejowych - 263, serwisu stacji kolejowych - 259, serwisu biletów - 256, a serwisu użytkowników – 314. Z tego powodu w niniejszym dokumencie przedstawiono wyłącznie przykłądy okrojonych okrojone WSDL'i zawieracjące tylko jedną metodę na raz.

* Przykładowy WSDL „Tworzenia biletu”:



* Przykładowy WSDL „Pobrania listy biletów”:



* Przykładowy WSDL „Szczegóły trasy”:



* Przykładowy WSDL „Zmiana hasła”:



* Przykładowy WSDL „Usuwanie stacji”:



## 2.3. Dostępne usługi

Usługi zostały podzielone na 5 grup użytkowych, reprezentowanych przez odpowiednie serwisy.

|  |  |
| --- | --- |
| Grupa użytkowa | Dostępne usługi |
| Autoryzacja | * Logowanie * Reset Hasła (krok 1 i 2) |
| Trasy kolejowe | * Utwórz trasę (A) * Szczegóły trasy (U) (A) * Modyfikacja trasy (A) * Usuwanie trasy (A) * Wypisanie wszystkich tras |
| Stacje kolejowe | * Utwórz stacji (A) * Szczegóły stacji (U) (A) * Modyfikacja stacji (A) * Usuwanie stacji (A) * Wypisanie wszystkich stacji |
| Bilety | * Tworzenie biletu (U) (A) * Szczegóły biletu (U) (A) * Tworzenie PDF biletu (U) (A) * Modyfikacja biletu (A) * Usuwanie biletu (A) * Wypisywanie wszystkich biletów danego użytkownika (U) (A) * Wypisywanie wszystkich biletów utworzonych w serwisie (A) |
| Użytkownik | * Tworzenie użytkownika * Szczegóły użytkownika (U) * Modyfikacja użytkownika (U) * Usuwanie użytkownika (U) * Zmiana hasła (U) * Wypisani wszystkich użytkowników (A) |

U - Usługa dostępna dla użytkownika serwisu

A - Usługa dostępna dla administratora serwisu

## 2.4. Opis wybranych usług

### 2.4.1. Operacja ...

Opis szczegółowy:

Dane wejściowe:

Dane wyjściowe:

Komunikat wysyłany do usługi:

(przykładowe przesyłane komunikaty SOAP- opis)

Komunikat zwrócony z usługi:

### 2.4.2. Operacja …

Opis szczegółowy:

Dane wejściowe:

Dane wyjściowe:

Komunikat wysyłany do usługi:

Komunikat zwrócony z usługi:

### 2.4.3. Operacja …

Opis szczegółowy:

Dane wejściowe:

Dane wyjściowe:

Komunikat wysyłany do usługi:

Komunikat zwrócony z usługi:

### 2.4.4. Operacja …

Opis szczegółowy:

Dane wejściowe:

Dane wyjściowe:

Komunikat wysyłany do usługi:

Komunikat zwrócony z usługi:

# 3.Instrukcja użytkownika

## 3.1 Instrukcja do Web Service

## 3.2 Instrukcja do Aplikacji klienckiej

## 3.3 Ograniczenia dla usługi WS

W celu zapewnienia dostępności Web Service dla wszystkich zainteresowanych podmiotów, twórcy zastrzega sobie prawo do czasowego/trwałego ograniczania dostępności usługi dla podmiotów nieprzestrzegających poniższych zasad korzystania z usługi. Podmioty korzystające z usługi Web Service “Systemu rezerwacji biletów kolejowych” zobowiązane są do zachowania następujących zasad/ograniczeń:

* przestrzeganie opisanych w specyfikacji standardów sieciowych, w szczególności konstruowanie komunikatów przesyłanych do usługi zgodnie z opublikowaną specyfikacją wejścia-wyjścia oraz opisem WSDL.
* ograniczenie ilości przesyłanych zapytań do maksymalnie 10 w ustalonym przedziale czasowym (1 sekunda);
* ograniczenie ilości przesyłanych zapytań z jednego adresu źródłowego (IP,domena);
* ograniczenie ilości przesyłanych zapytań dot. Jednej trasy czy też stacji (IP, domena);

Liczba zapytań do usługi może być ograniczana czasowo/trwale dla użytkowników nieprzestrzegających powyższych reguł. Użytkownik korzystając z usług Web Service zobowiązany jest do przestrzegania przepisów dotyczących bezpieczeństwa przetwarzania danych.