

Do poczytania o technologii WCF (Windows Communication Foundation):

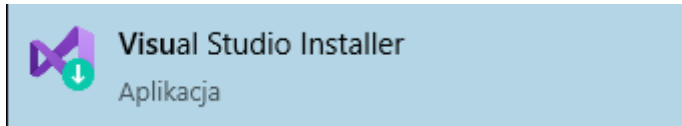
<https://cezarywalenciuk.pl/blog/programing/category/wcf>

<https://docs.microsoft.com/pl-pl/dotnet/framework/wcf/whats-wcf>

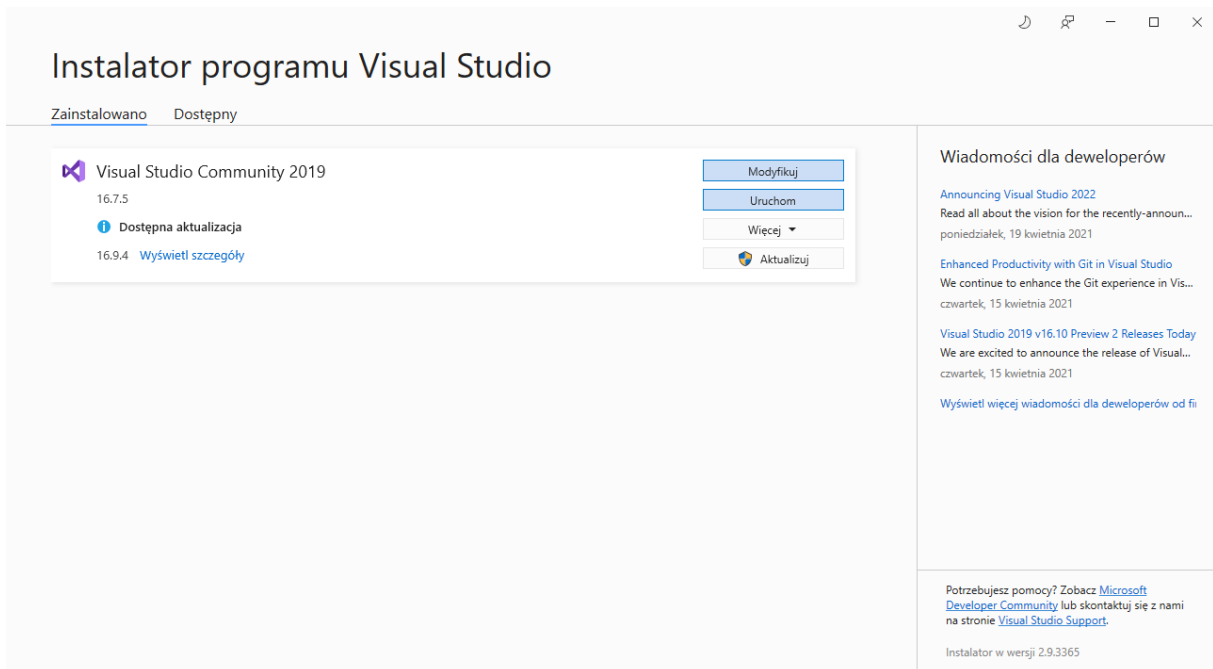
Aplikacja Postman:

<https://www.postman.com/downloads/>

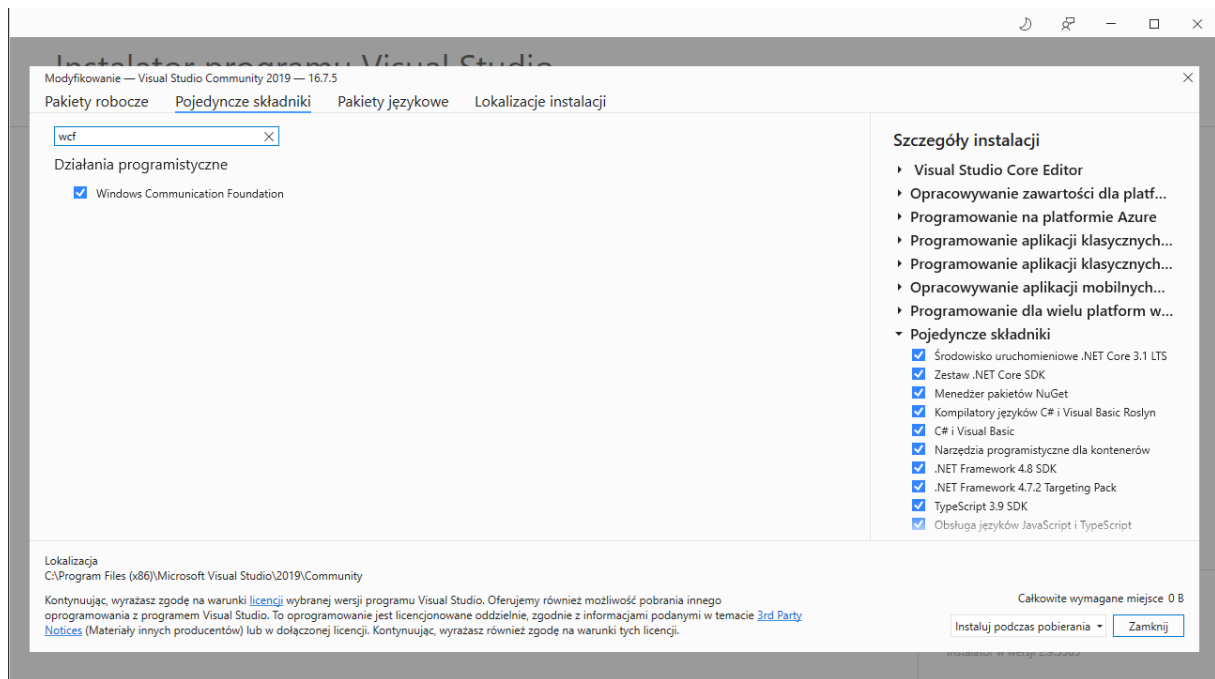
Uruchamiamy Visual Studio Installer:



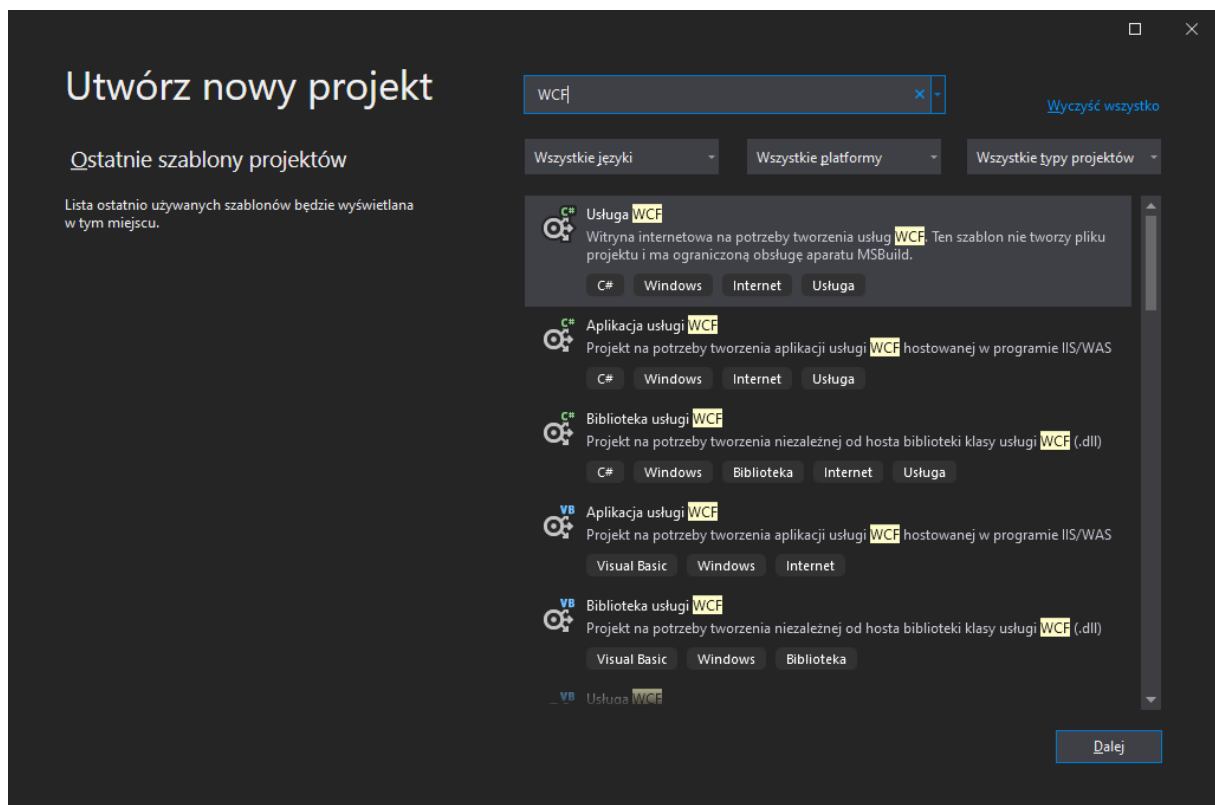
Klikamy przycisk „Modyfikuj”:



Następnie z zakładki „Pojedyncze składniki” instalujemy „WCF”:



Tworzenie projektu WCF:



Wprowadzamy nazwę projektu i rozwiązania.

Konfiguruj nowy projekt

Usługa WCF C# Windows Internet Usługa

Nazwa projektu

UsługaWCF

Lokalizacja

C:\WCF

Rozwiązanie

Utwórz nowe rozwiązanie

Nazwa rozwiązania ⓘ

UsługaWCF

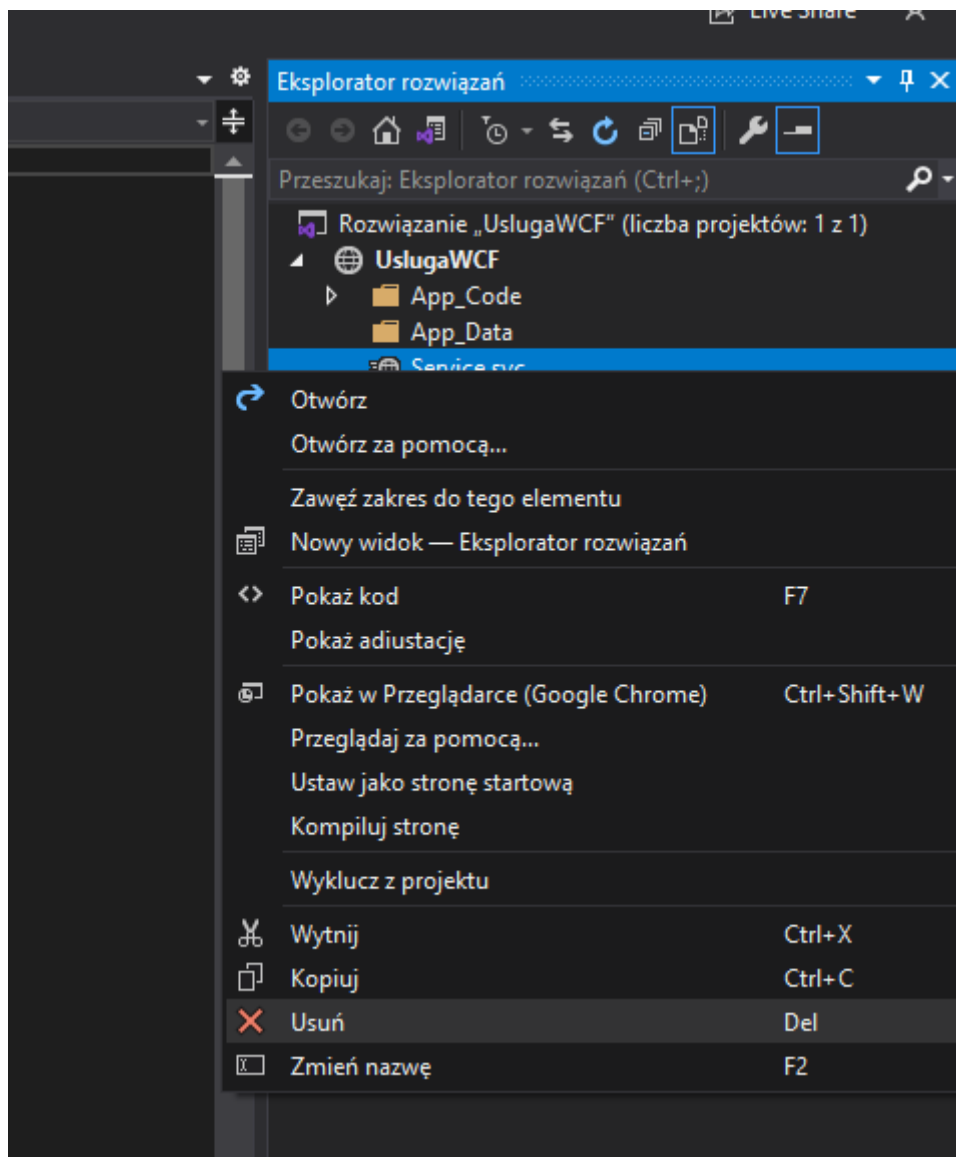
☐ Umieść rozwiązanie i projekt w tym samym katalogu

Platforma

.NET Framework 4.7.2

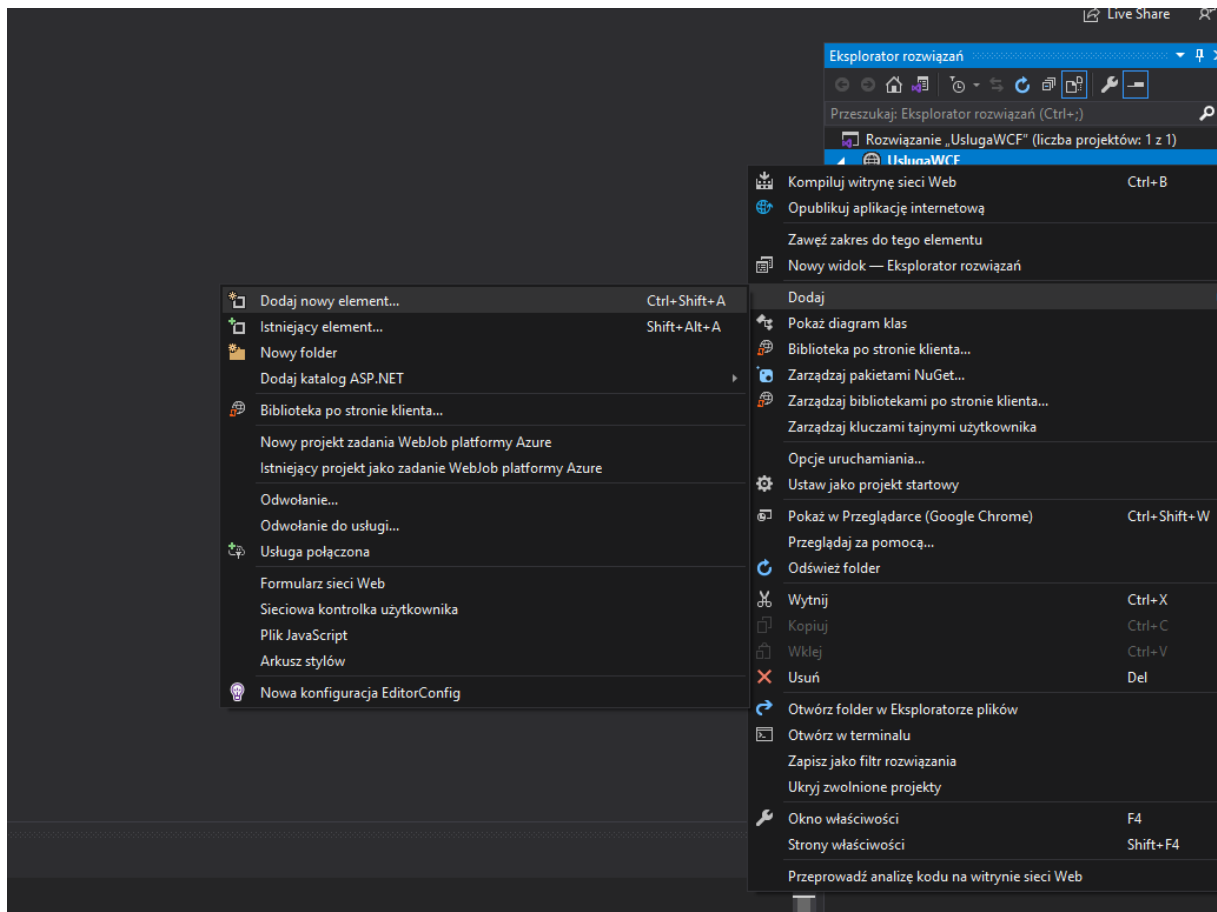
Wstecz Utwórz

Usuujemy domyślną usługę (oraz oba pliki w App\_Code)

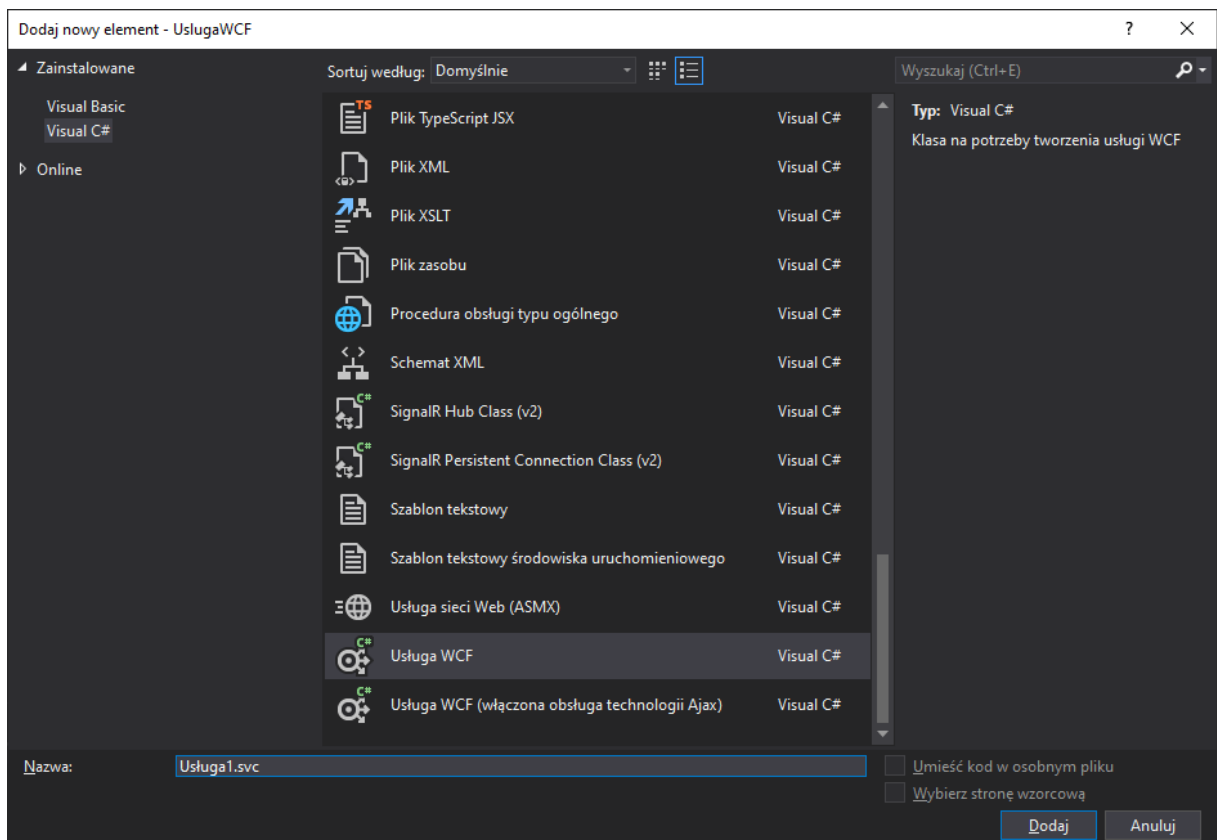


Dodajemy nową usługę:

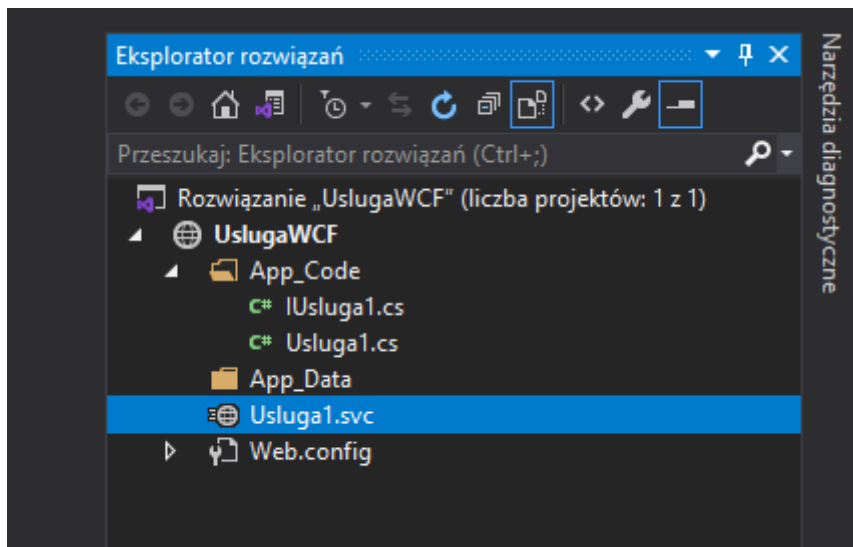
Klikamy PPM na nazwie projektu i wybieramy Dodaj > Dodaj nowy element...



Z listy wybieramy Usługa WCF a następnie nadajemy jej nazwę poniżej:



Widok po dodaniu usługi:



Interesują nas pliki CS. Plik Usługa1.cs to plik z klasą, w której implementować będziemy nasze funkcje. Plik IUsluga1.cs to interfejs, który musi spełniać klasa Usługi1. Uproszczając interfejs ten zawiera listę nagłówków publicznych funkcji z klasy Usługa1.

Interfejsów używa się z dziedziczeniem. Pełnią one zazwyczaj rolę swego rodzaju umowy, którą klasy muszą spełniać. Programista tworząc interfejs wprowadza tam listę metod które muszą zostać zaimplementowane przez klasę. (Do poczytania: dziedziczenie, polimorfizm, dependenci inversion, dependenci injection).

Przyjmuje się konwencje nazywanie interfejsów zaczynając od dużej litery I.

Przykład:

```
10 public interface IUsluga1
11 {
12     1 odwołanie
13     int Dodawanie(int a, int b);
14     1 odwołanie
15     int Mnozenie(int a, int b);
16 }
```

Powyższy interfejs wymusza na każdej klasie, która z niego dziedziczy by ta implementowała te dwie publiczne(!) metody.

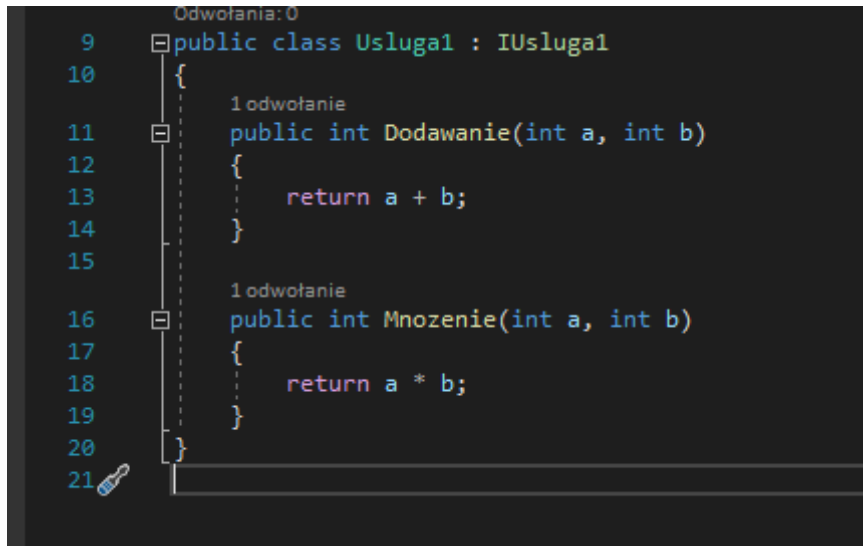
Dziedziczymy z interfejsu:

```
9 public class Usluga1 : IUsluga1
10 {
11 }
12
13
```

Jak widać nazwa interfejsu jest podkreślona na czerwono gdyż nie jest on zaimplementowany.

Zaimplementowanie tylko części interfejsu nie powoduje, że błąd znika.

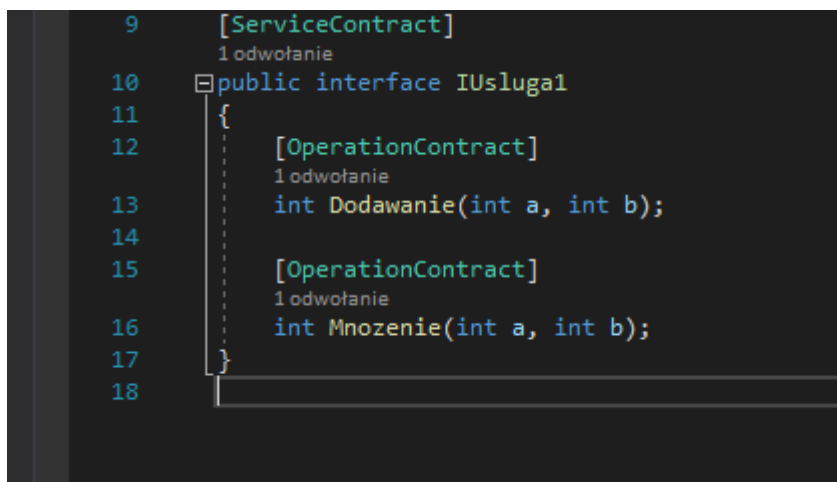
Po implementacji wszystko jest OK.:



```
9      public class Usluga1 : IUsluga1
10     {
11         1 odwołanie
12         public int Dodawanie(int a, int b)
13         {
14             return a + b;
15         }
16
17         1 odwołanie
18         public int Mnozenie(int a, int b)
19         {
20             return a * b;
21         }
22     }
```

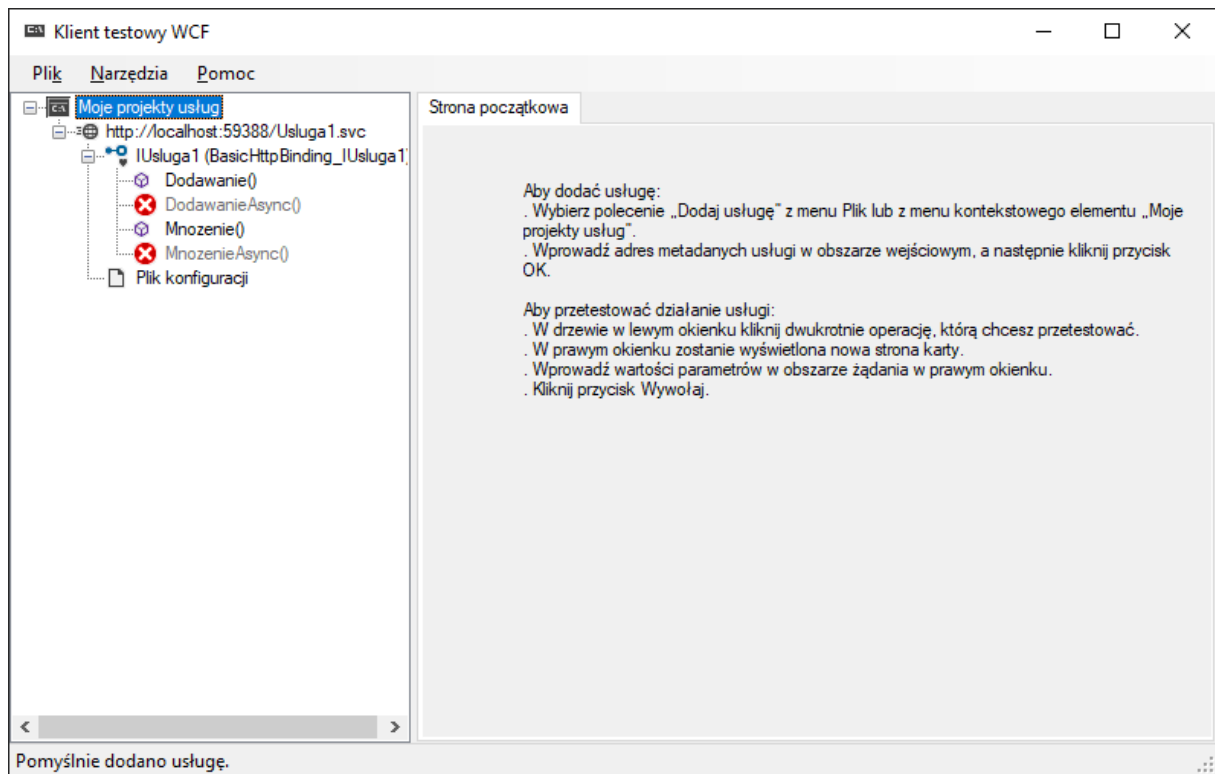
Dodatkowo WCF do działania wymaga nadania metodą wystawionym publicznie atrybutu: [OperationContract].

Możemy to zrobić w interfejsie.

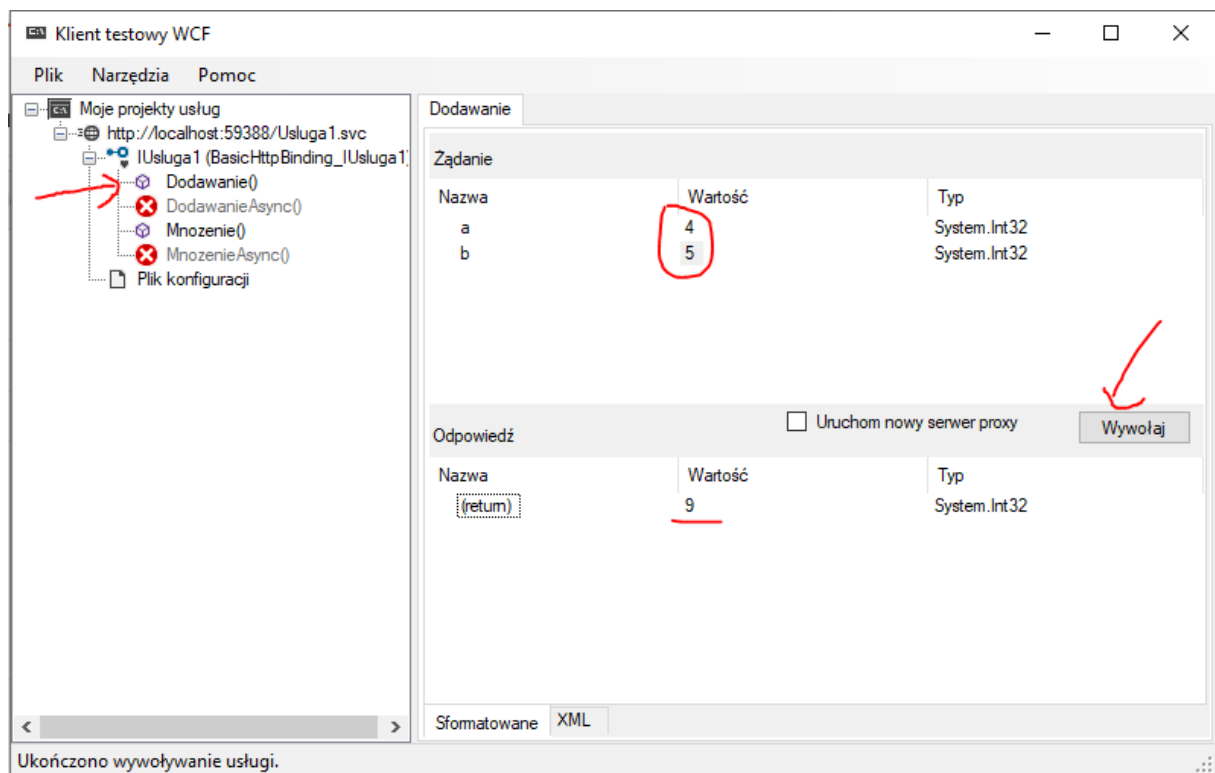


```
9      [ServiceContract]
10      1 odwołanie
11      public interface IUsluga1
12      {
13          [OperationContract]
14          1 odwołanie
15          int Dodawanie(int a, int b);
16
17          [OperationContract]
18          1 odwołanie
19          int Mnozenie(int a, int b);
20      }
```

Zapisujemy, zamykamy oba pliki i klikamy F5. Dostajemy takie okienko:



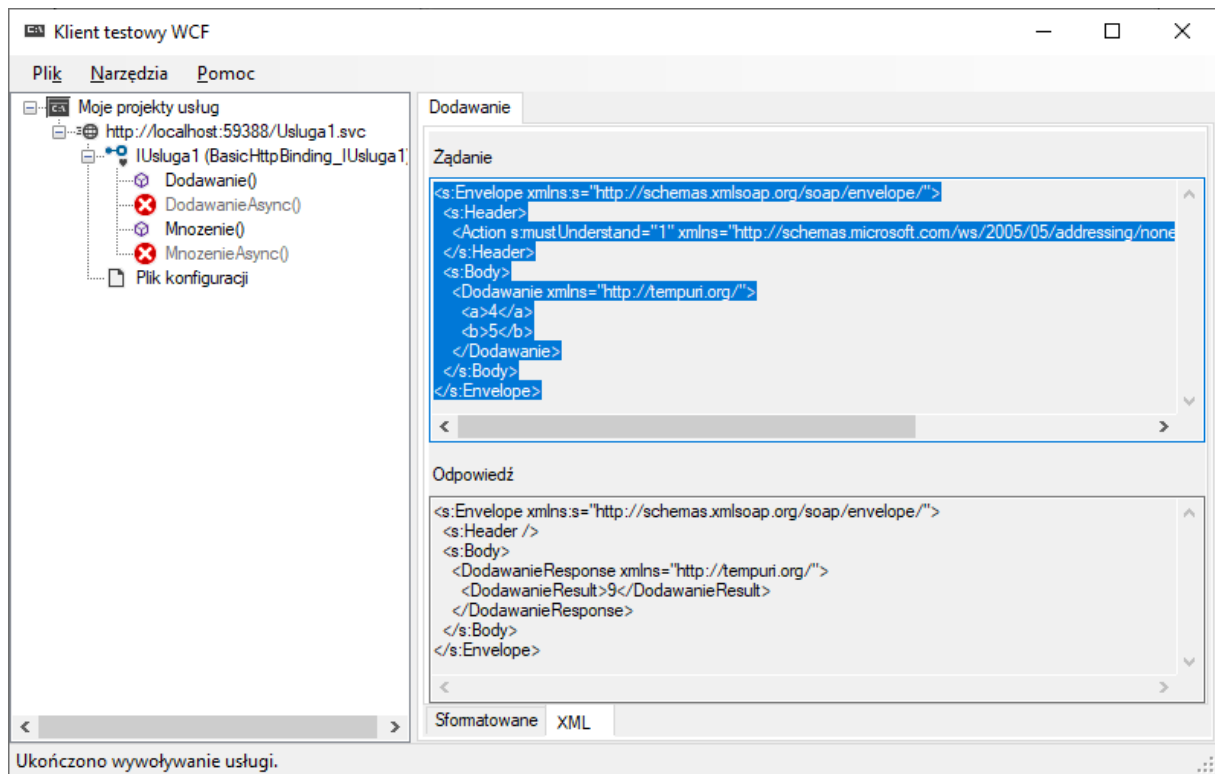
Test:



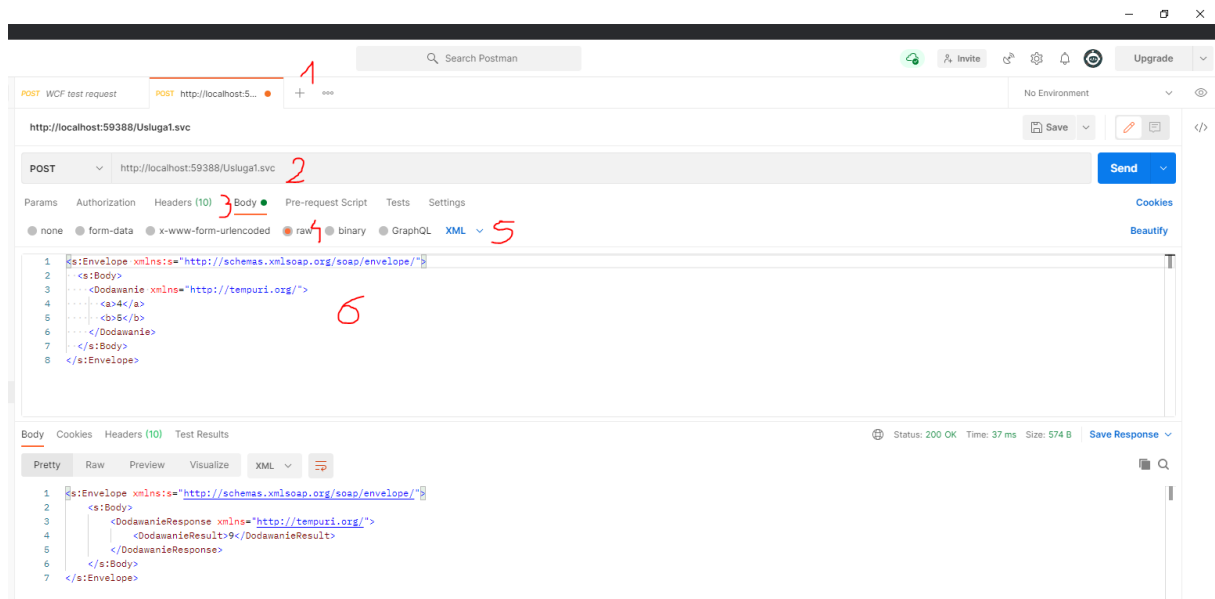
Testowanie w programie POSTMAN:

W powyższym kliencie w zakładce XML mamy zapytanie xml, które wykorzystamy w POSTMANie:

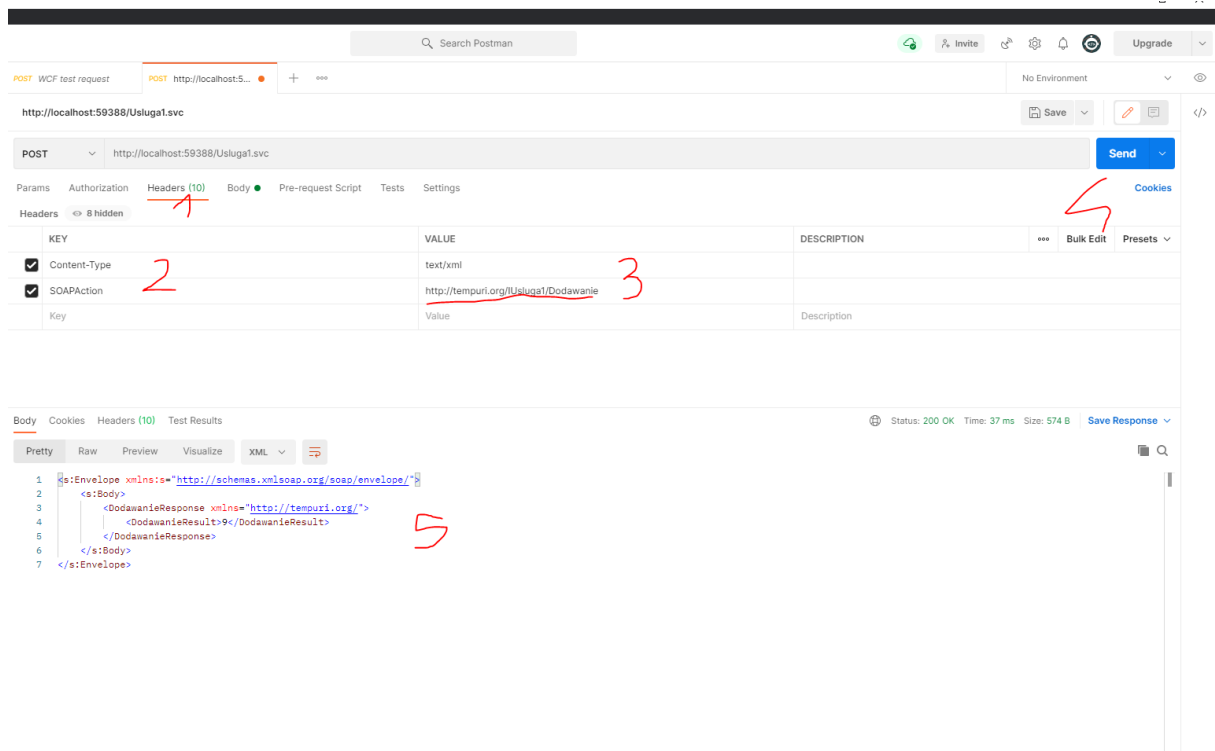




Uruchamiamy POSTMAN.



W pasek adresu wpisujemy adres z powyższego klienta. Wybieramy odpowiednie ciało żądania i wprowadzamy takie samo jak w kliencie bez sekcji header.



Przechodzimy do nagłówków i dodajemy dwa jak na obrazu z tym, że adres w drugim nagłówku to ten sam adres z XML z sekcji HEADER.

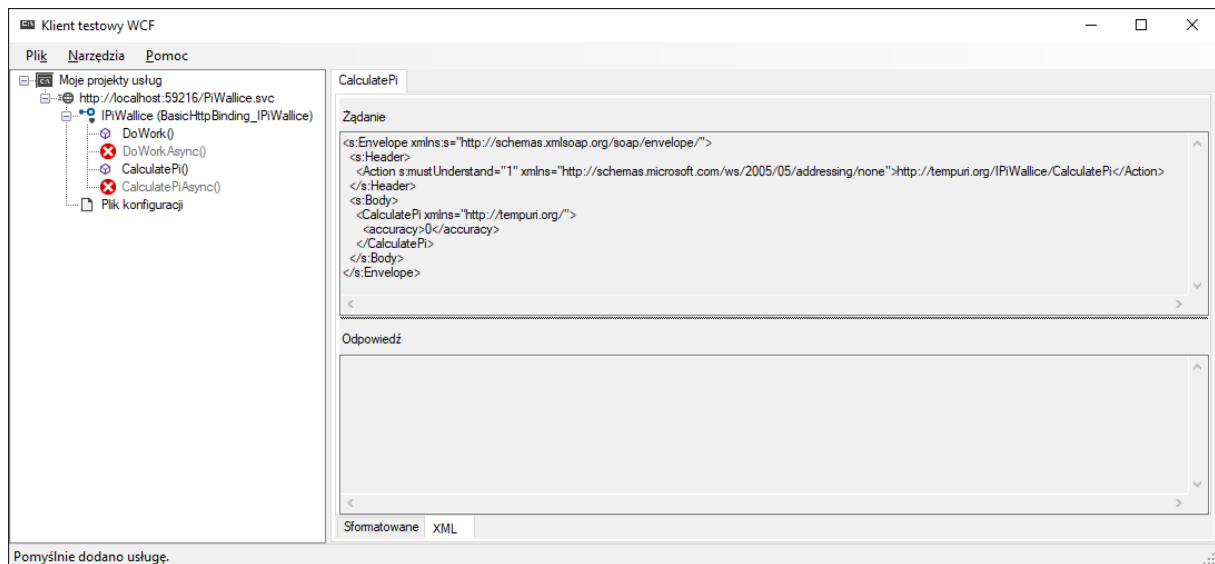
Na końcu wysyłamy żądanie i otrzymujemy odpowiedź w xmlu.

Zadanie 1:

Kod źródłowy z powyższych ćwiczeń umieść na repozytorium wraz z pozostałymi zadaniami.

Zadanie 2:

Zmodyfikuj otrzymany kod klienta by uzyskać dostęp do endpointa utworzonego przez SampleServices . Dodaj także możliwość dynamicznego pobierania i ustawiania parametru accuracy.



```
10 public interface IPiWallice
11 {
12     [OperationContract]
13     double CalculatePi(int accuracy);
14 }
15
```

```
Konsola debugowania programu Microsoft Visual Studio
Wywołuje
Wynik: 3.1415826535897198
Koniec
```