1. Magdalena Fuczko
2. Dawid Czarnecki
3. Michał Cziomer
4. Magda Chrostek
5. Artur Bogusiewicz

Tematem projektu jest rozpatrzenie oraz realizacja WebService'ów w języku Go[1].

Spis treści

1. Narzędzia
2. Analiza dostępności
3. Diagramy UML
4. Zasada działania scryptu
5. Wnioski
6. Źródła

1. Narzędzia

W Projekcie użyliśmy następujących narzędzi.

Narzędzia programistyczne:

- GOWSDL
- framework Martini

Narzędzia wspierające:

- Slack - aplikacja służąca do komunikacji
- Github - kontrola wersji
- Scrum - ramy postępowania
- GoDoc - narzędzie służące do tworzenia dokumentacji

2. Analiza dostępności

Istnieje dużo popularnych (jak na biblioteki GO) pakietów umożliwiających stworzenie WebService'ów w Go, lecz przeważnie są to pakiety REST-owe (np. Martini). Pakiety umożliwiające obsługę SOAP z reguły są przestarzałe, nierozwijane, bardzo ograniczone i problematyczne w skompilowaniu.

[1] Język Go - https://golang.org/

Pakiety są ograniczone najczęściej przez złożoność protokołu SOAP, gdzie większość pakietów powstały z małych jedno lub kilkuosobowych projektów. Związane to jest także z ciągłym poszerzaniem specyfikacji SOAP, która już jest bardzo rozbudowana.

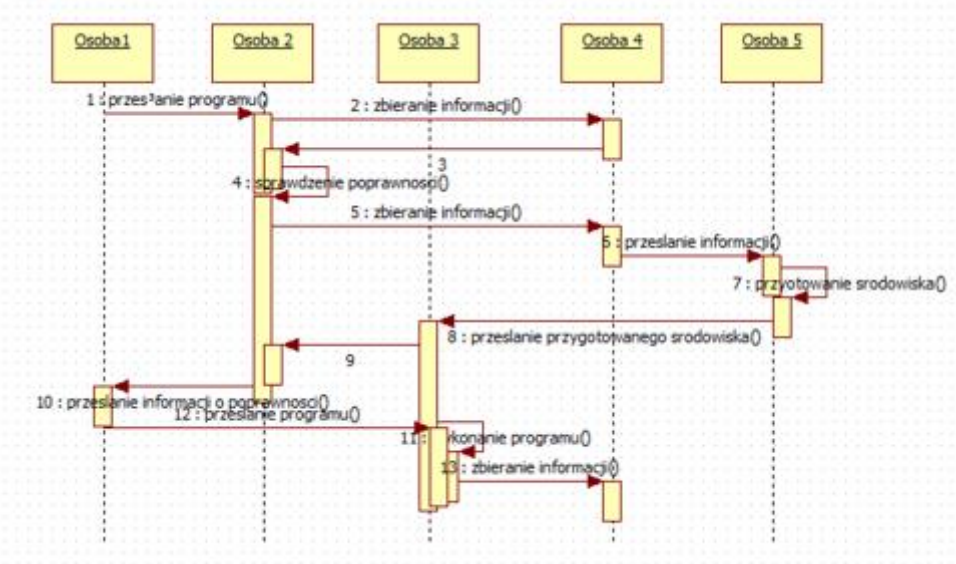
Framework **Martini** to solidny framework umożliwiający sprawne tworzenie modularnych aplikacji internetowych i usług sieciowych w języku Go. Jest on szeroko oceniany jako prosty w użyciu. Posiada nie wymagającą ingerencji budowę łatwo integrującą się z innymi pakietami w języku GO. Jego niewątpliwym plusem jest tu też dopasowywanie ścieżek i routingu, a także bogate zasoby handlerów i middleware'ów do wykorzystania. Jest w pełni kompatybilny z interfejsem http.HandlerFunc.[18]

WSDL, Web Services Description Language to język służący do definiowania i opisu korzystania z WebService'ów. Oparty jest na XML, serwisy zostały zinterpretowane jako kolekcja punktów końcowych - adekwatnych URI, do których wysyłane są żądania - mających możliwość przesyłania danych zawierających dokumenty lub parametry/wyniki zdalnych metod. Podobnie jak w przypadku CORBY celem jest definiowanie interfejsów i typów. WSDL ma jednak szersze możliwości np. przez traktowanie typów portów jako abstrakcyjne kolekcje operacji udostępnianych przez WebService. WSDL daje nam opis usług udostępnianych przez WebService, informacje jak wywoływać jego operacje i gdzie znaleźć WebService. WSDL definiuje od razu rozszerzenia dla wiązań z: m.in. SOAP 1.1. [17]

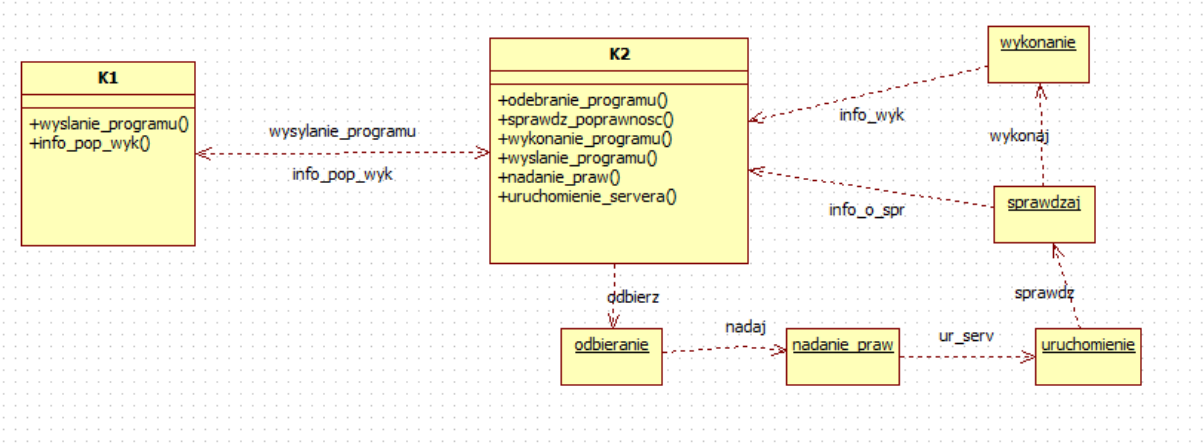
Ponieważ WSDL jest abstrakcyjnym opisem interfejsów Web Serwisu możliwe jest obecnie zarówno częściowe wygenerowanie kodu implementacji Web Serwisu o ile dane są definicje WSDL, jak i wygenerowanie definicji WSDL jeśli dany jest kod implementujący Web Serwis.
Podobno lepiej jest najpierw budować podstawowa, działająca implementacje , a następnie przy użyciu odpowiednich narzędzi wygenerować definicje WSDL. Przykładowe narzędzia, które udostępniają taką funkcjonalność to np. BEA's WebLogic Server 6.1.

Język GO oficjalnie nie wspiera technologii SOAP, lecz stosując biblioteki zbudowane przez społeczność jest możliwość stworzenia web serwisu. Konwersję kodu WSDL na kod GO umożliwiają zewnętrzne narzędzia takie jak WSDLGO. Udostępnia to metody i klasy tylko dla klienta. Po stronie serwera musieliśmy obsłużyć to samodzielnie na podstawie klas udostępnionych dla klienta.[5].

3. Diagramy UML



Rys.1. Sequence-Diagram



Rys.2.Dataflow-diagram

4. Zasada działania scryptu

Scrypt wyszukuje uruchamiany plik servera oraz przypisuje mu określone prawa. Za pomocą exec.Command() uruchamia zdalnie server na określonych wcześniej prawach oraz oczekuje na jego zamknięcie. Po zamknięciu się server kończy działanie aplikacji

5. Wnioski

W czasie wykonywania tego projektu napotkaliśmy na wiele trudności, które ostatecznie udało nam się pokonać. Na początku musieliśmy zapoznać się z językiem GO, który to nie jest zbytnio popularny. SOAP jest technologią, która przestała być popularna zanim powstał język GO - stąd też znikome wsparcie dla SOAP w GO. Zbudowaliśmy projekt, który funkcjonuje w sposób zgodny z założeniami. Potwierdzają to wykonane testy. Diagramy w sposób czytelny obrazują działanie naszego projektu. Ich celem było przedstawienie czasu życia wykonywanych przez nas modułów oraz przepływu danych między nimi. Pokazują również zaangażowanie każdego z uczestników. Dodatkowo wykonany Scrypt działa na zasadzie wyszukiwania uruchamianego pliku servera oraz przypisywania mu określonego prawa. Brak popularności tej technologii umożliwił zdobycie umiejętności przeszukiwania informacji, rozwiązywania problemów nierozwiązywalnych, zrozumienie idei tego przedmiotu, a także zapoznanie się z narzędziami pomocniczymi jak GoDoc i Slack.

6. Źródła

1. https://golang.org/doc/ - Oficjalna strona języka GO, ostatnia wprowadzona aktualizacja: 10.04.2017r.
2. https://golang.org/pkg/testing/ - Testy stosowane w języku GO, oficjalna strona języka GO
3. https://www.w3schools.com/xml/xml_soap.asp - “xml soap”, w3schools,strona aktualnie wspierana
4. https://www.w3schools.com/xml/xml_wsdl.asp - “xml wsdl”,w3schools, strona aktualnie wspierana
5. https://github.com/hooklift/gowsdl - “WSDL to GO”,Camilo Aguilar, ostatnia aktualizacja: 17.05.2017r., źródło sugerowane przez oficjalną stronę golanglibs
6. https://www.oxygenxml.com/demo/Create_New_WSDL.html -”WSDL”, oxygen, strona aktualnie wspierana
7. https://godoc.org/ - GoDoc, strona geenerująca dokumentacja dla jezyka GO
8. https://app.pluralsight.com/library/courses/go - Serwis Pluralsight, Tutoriale GO, aktualizacja: 9.07.2015r.

9. https://github.com/justwatchcom/goat - “Generate SOAP requests for Go at runtime”, Patrick Kohan, ostatnia aktualizacja: 7.04.2016r.
10. https://github.com/mfenniak/go-soap “Soap client for GO”, Mathieu Fenniak, ostatnia aktualizacja: 28.05.2013r,
11. https://github.com/achiku/sample-golang-soap-xml/blob/master/soap.go -”Golang soap”, Akira Chiku, ostatnia aktualizacja: 7.05.2016r.
12. http://rapidtrade.screenstepslive.com/s/standards/m/54261/l/560203-calling-a-soap-web-service, “Calling a SOAP web service”,aktualizacja: 29.06.2017r.
13. http://rapidtrade.screenstepslive.com/s/standards/m/54261/l/615902-creating-a-soap-server, “Creating a soap server”, aktualizacja: 24.01.2017r.
14. https://www.wsdl-analyzer.com/ - Analizer WSDL, analiza i porównanie
15. https://blog.smartbear.com/web-development/how-to-build-a-web-service-in-5-minutes-with-go/
16. https://golanglibs.com/ - GO libriaries
17. http://stencel.mimuw.edu.pl/abwi/20020514.WSDL/ “WSDL (Web Services Description Language)”,Krzysztof Stencel, materiały do przedmiotu
18. https://github.com/go-martini/martini - “Martini framework”, Miguel Molina, aktualizacja: 21.01.2017r.
19. https://commandwindows.com/command3.html „Command Line List and Reference” Vic Laurie 2012r
20. https://www.socketloop.com/tutorials “GO OS TUTORIAL”By Adam Ng aktualizacja 17 lipca 2015r.
21. https://nathanleclaire.com/blog/2014/12/29/shelled-out-commands-in-golang/ “Shelled-out Commands In Golang by Nathan Leclaire 29. Grudzień 2014r.