Budowa i Orkiestracja mikroserwisów z wykorzystaniem nowoczesnych technik wytwarzania oprogramowania

Spis treści

1. **Wstęp**

1.1 Cel I zakres pracy

1.2 Architektura mikroserwisów

1.3 Orkiestracja mikroserwisów

**2. Protokoły komunikacyjne mikroserwisów**

2.1 Protokół komunikacyjny SOAP

2.2 Protokół komunikacyjny REST

2.3 Porównianie protokołów komunikacyjnych

**3. Formaty wymiany danych**

3.1 XML jako format wymiany danych

3.2 JSON jako format wymiany danych

2.3 Zalety i wady formatów XML oraz JSON w komunikacji między serwisami

**4. Definicja oraz opis Web Serwisów**

4.1 XML Schema definition

3.2 Web Services Description Language

2.3 Generowanie WSDL za pomocą XSD

**5. Budowa Web Serwisu**

4.1 Spring-ws

3.2 Generowanie modelu z wykorzystaniem JAXB

2.3 Architektura wewnetrzna Web serwisu

**6. Narzędzia wspomagające budowę aplikacji**

4.1 Automatyzacja budowy oprogramowania

3.2 Systemy kontroli wersji

2.3 Narzędzia wspomagające wdrożenie serwisów

**6. Wnioski**

**7. Bibliografia**

1. Pojęcie mikroserwisu jest stosunkowo nowe w świecie informatyki, wywodzi się ono z architektury zorientowanej na usługi SOA (ang. Service-Oriented Architekture) której koncepcja została zaprezentowana już wiele lat temu i z pomyślnocią była stosowana w wielu zaawansowanych systemach i projektach informatycznych. Pojęcie mikroserwisów zwiększa granulację usług, według tej koncepcji pojedyncze serwisy powinny mieć minimalny zakres odpowiedzialności co pociąga za sobą posiadanie przez serwis tylko jednej funkconalności. Architektura systemu komputerowego oparta na miktoserwisach pociąga za sobą szereg pozytywnych aspektów. Jednym znich jest niewątpliwie skalowalność aplikacji. Dzięki zastosowaniu małych serwisów system jest budowany w sposób niezależny. Projektant całościowej aplikacji ma swobode doboru poszczególnych elementów będącymi mikroserwisami, może wybrać który serwis najlepiej pasuje do charakterystyki aplikacji i który zwraca oraz przetważa w pożadany sposób dane niezbędne do zbudowania końcowego produktu. Programiści tworzący i rozwijający różne części systemu w postaci mikroserwisów skupiają się jedynie na swojej części, nie wnikając w logikę, sposób działania ani nawet w jezyk i technologie w jakich są napisane pozostałe części systemu. Wykorzystanie mikroserwisów w budowie dużych systemów komputerowych ma także wiele plusów w stosunku do wielkich monolitycznych systemów w którym zmiana w jednej części aplikacji często powoduje konieczność zmian wiekszej ilości elementów. Zmiana w obrębie małego serwisu nie wpływa na sposób działania pozostałych. Wszystkie mikroserwisy są releasowane niezależnie a co za tym idzie po wprowadzniunawet niewielkiej zmiany w systemie nie jest konieczne kompilowanie oraz uruchamianie całego systemu na nowo. W przypadku wprowadzania nowej funkcjonalności czy też nawet naprawie błędów jakie zostały znalezione w poprzedniej wersji aplikacji monolityczne systemy wymagają całego cyklu testów aplikacji aby mieć pewność ze zmiany lub poprawki wprowadzone w systemie nie miały wpływu na pozostałe części aplikacji oraz nie powodują niepożądanych skutków w działaniu oprogramowania. W przypadków mikroserwisów po wprowadzonych zmianach konieczne jest jedynie przeprowanienie testów dotyczących jedynie tego jednago serwisu którego zmiany dotyczyły oraz testów integracyjnych całego systemu związanych w jakikolwiek sposób ze zmieniamym serwisem. Dodatkowo zmiany wprowadzane w mikroserwisach mogą być wprowadzane o wiele szybciej a proces inwestygacji błędów w aplikacji jest o wiele szybszy niż w złożonych monolitycznych systemach.

2. W ramach pracy implementowany był system wyszukiwania ofert pokojowych w obiekcie hotelowym. Zgodnie z tematem pracy architekturą zastosowaną w budowaniu aplikacji jest architektura mikroserwisów. Aby wyodrębnić odpowiednie serwisy na początkowym etapie modelowania system został podzielony według poszczególnych funkcjonalności.

Pierwszą funkcjonalnością jaką udało się wyodrebić jest zarządzanie sesją. Mikroserwisy mają różny przydział uprawnień a co za tym idzie nie każdy z nich powinien mieć dostęp do zasobów innych serwisów. Podczas jednej tranzakcji miedzy serwisami sesja powinna zostać stworzona, następnie zweryfikowana podczas próby dostępu do niej a następnie kiedy nie jest już poptrzebna usunięta z puli dostępnych sesji. Aby obsłużyć taką funkcjonalnosć powstał mikroserwis ManageSession który posiada trzy operacje. Pierwszą z operacji serwisu ManageSession jest operacja createSession za pośrednictwem której może zostać stworzona nowa sesja. Drugą oferowaną usługą jest verifySession. Za jej pośrednictwem weryfikowane jest czy dany serwis ma uprawnienia do skorzystania z niej. Ostatnią funkcją mikroserwisu jest endSession która kończy sesje i usuwa ją z puli dostepnych sesji.

Drugim wyodrębnionym mikroserwisem jest serwis o nazwie CurrencyService który za pomocą operacji getCurrencyExchangeRateAction zwraca kurs wymiany między dwoma walutami. Zapytanie zawiera parę walut, bazową, oraz docelową, pomiędzy którymi zostanie zwrócony współczynnik wymiany. Źródłem danych odnośnie kursu walut jest Europejski Bank Centralny. Instytucja ta udostępnia REST api umożliwiające pobranie najbardziej aktualnych kursów walut. Api to jest dostępne do swobodnego użytku przez zewnętrznych użytkowników. System został stworzony jako swego rodzaju Proxy dla danych zwracanych przez serwis udostępniony za pomocą api stworzonego przez Europejski Bank Centralny. Filtruje on jedynie te dane dla jakich wysłano zapytanie do serwisu CurrencyService oraz zwraca w formie pasującej do specyfiki całościowego systemu.

Trzecim małym serwisem jest kolejny serwis który zajmuje sie konwersją walut jest to serwis ExchangeSystemService. Mikroserwis ten przelicza kwoty między dwomia zadanymi walutami. Klient serwisu w zapytaniu wysyła kwotę oraz bazową walutę w jakiej ta kwota została podana, natomiast serwis zwraca kwotę przeliczoną na walutę docelową o którą pyta użytkownik. Do wyznaczania kursu między obiema walutami został użyty poprzednio wymieniony serwis CurrencyService. Funckjonalności jakie dostarczają mikroserwisy CurrencyService oraz ExchangeService są bardzo miałe i mogły by być skonsolidowane w postaci jednego serwisu, ale jest to doskonały przykład jak uzyskać minimalny poziom granulowości dla mikroserwisów który ułatwi późniejsze ich niezależne modyfikacje i dlatego w takiej formie zostały stworzone na potrzeby pracy.

Kolejnym serwisem jest serwis RoomFinderService udostępniający operację findAvailableRoomsAction. Operacja ta odpowiada po pierwsze za wyszukanie wszystkich pokoi które spełniają zadane w zapytaniu kryteria jakie sprecyzował klient. Dane odnośnie pokoi hotelowych serwis pobiera ze źródła danych jakim jest w tej sytuacji baza danych SQL. Drugą bardzo ważną funkcją jaką zajmuje się serwis RoomFinderService jest sprawdzenie dostępności pokoju w podanym przez klienta w zapytaniu terminie. Do tego celu serwis komunikuje się zkolejnym serwisem RoomReservationService (wymieniony w kolejnym podpunkcie) który zawiera informację odnośnie rezerwacji. Serwis RoomFinderService filtruje odpowiedz na podstawie odpowiedzi z serwisu RoomReservationService i w swojej odpowiedzi zwraca jedynie informacje o tych pokojach dla których możlowa jest rezerwacja w zadanym przedziale czasowym.

Następnym wspomnianym już w poprzednim podpunkcie serwisem jest RoomReservationService który zawiera operację checkRoomAvailabilityAction. Operacja ta służy do sprawdzenia czy dany pokój jest dostępny do rezerwacji w zadanym przedziale czasowym. Dane odnośnie rezerwacji pokoi hotelowych składowane są w bazie danych, natomiast mikroserwis RoomReservationService po przeanaliziwaniu tych danych zwraca informację o dostępności pokoju.

Serwisem odpowiedzialnym za zwrócenie ceny za rezerwacjię danego pokoju hotelowego jest serwis PriceCalculationService. Zródłem danych serwisu jest baza danych zawierająca informację na temat cen poszczególnych pokoi hotelowych oraz regół na podstawie jakich takie ceny są nakładane na ofertę rezerwacji. Akcja o nazwie priceRoomAction poprzez wykorzystanie danych zawartych w zapytaniu oraz informacji z bazy danych przelicza cenę rezerwacji na całość requestowanego pobytu w obiekcie oraz zwraca te dane w odpowiedzi. W przypadku cen zdefiniowanych w innej walucie niż waluta docelowa jaka została wysłąna w zapytaniu, serwis korzysta z mikroserwisu ExcangeSystemSercice aby przeliczyć informację o cenie pokoi na odpowiedną walutę.

Ostatnim większym serwisem jest serwis o nazwie RoomOffersService. Serwis ten jest odpwoeidzialny za orkiestrację serwisów niższego poziomu czyli RoomFinderService oraz PricingService. Akcją udostępnioną przez serwis jest getRoomOffersAction. Odpowiada ona po pierwsze za znalezienie za pośrednictwem mikroserwisu RoomFinderService ofert dla pokoi hotelowych które są dostępne w zadanym terminie. Po drugie serwis dzięki mikroserwisowi PriceCalculationService przydziela ceny do poszczególnych ofert jakie zostaną zwrócone w odpowiedzi.

3.Orkiestracja mikroserwisów jest procesem łączenia ich przy pomocy głównego systemu, który w odpowiedniej kolejności je wywołuje aby otrzymać niezbędne dane lub przy zaich pomoc wywołać niezbędne akcje lub zdarzenia umożliwiające działanie systemu jako całości. Taki główny serwis ma za zadanie przetworzyć dane aby dawały on logiczny rezultat z punktu widzenia biznesowego oraz zwracał wartość biznesową dla klienta końcowegokorzystającego z serwisu orkiestrującego. Klient korzystający z serwisu nie musi wiedzieć co dzieje się w jego wnętrzu, jakie kolejne mikroserwisy i serwisy są wywoływane, interesuje go jedynie wartość danych jakie zostaną zwrócone za jago pomocą oraz lub też akcję jakie dzięki użytkowaniu serwisu zostały wykonane. Systemem zarządzającym czyli orkiestratorem może być koleny serwis który składa wywołania kilku innych lub system docelowy z którego korzysta końcowy użytkownik aplikacji. Pomimo wielu zalet jakie niesie za sobą niewątpliwie budowa systemu informatycznego opartego na mikroserwisach, architektura ta posiada także kilka wad które w przypadku niektórych systemów mogą przeważyć na korzyść budowy oprogramowania bardziej monolitycznego. Jednym z problemów jakie niesie za sobą orkiestracja mikroserwisów jest narzut czasu związany z komunikacją miedzy nimi. Komunikacja zarówno protokołem SOAP jak i REST jest zdecydowanie wolniejsza niż lokalne wywołanie metod. W aplikacjach w których krytyczną rzeczą jest czas reakcji systemu może być to bardzo duża przeszkoda do wdrożenia architektury opartej na mikroserwisach. Aby częściowo zapobiec dużemu narzutowi na komunikacje miedzy serwisami, powszechną i często spotykaną ideą jest wykorzystanie lokalnego cache dla odpowiedzi z wywoływanych serwisów. Stosuje się to zwłaszcza w sytuacjach w których zapytania często są takie same i dotyczą identycznych danych. W przypadkach kiedy dane zwracane w odpowiedzi z wywoływanych serwisów są dynamiczne, często się zmianiają i nie koniecznie muszą zależeć od przesłanego zapytania, zastosowanie cache może być utrudnione a w niektórych przypadkach nawet niemożliwe. W momencie przetrzymywania danch z odpowiedzi mikroserwisu w lokalnej pamięci cache, niezbędne jest okresowy, lub zainicjowany pewnym zdarzeniem mechanizm który odświeża taki cache. Polega to na usunięciu całej lub żądanej jego części a w momencie kiedy takie dane są niezbędne dla systemu mechanizm wywoła zapytanie do mikroserwisu a jego nową świerzą odpowiedz zaladuje do cache. Kolejnym problemetycznym elementem w architekturze aplikacji opartej na mikroserwisach jest zarządzanie ich wersjami. W Systemach które cały czas się rozwijają oraz posiadają wiele różnych wersji powstaje problem ich zmiany na nowszą. Często w takich przypadkach taka zmiana powoduje konieczność modyfikacji systemy orkiestrującego. Zmiana wersji może pociągać za sobą także zmianę struktury zapytania do serwisu oraz odpowiedzi z niego. W takich przypadkach niezbędna jest modyfikacja sposobu budowania zapytania oraz przetwarzania danych zwróconych w odpwiedzi z takiego serwisu. W takich sytuacjach konieczna jest ścisła współpraca zespołów które tworzą system zarówno orkiestrujący jak i mikroserwisy w celu ustalenia sposobu i struktury kontraktu jakim komunikują się serwisy.

4.Architektura mikroserwisów wywodzi się z założeń architektury zorientowanej na usługi. Każdy serwis jak i mikroserwis jest tak naprawdę usługą dostarczającą pewną wartość. Podstawowym wymaganiem stawianym przed serwisami jest to aby dostarczały one pewną konieczną do realizacji systemu wartość biznesową. Klientem takiej usługu którą można nazwać komponentem nie musi być tylko i wyłącznie jeden system. Serwis taki powinien być zrealizowana w taki sposób aby dostarczać poprawną odpwiedz niezależnie od jego odbiory. Usłyga internetowa powinna być komponentem niezależnym od systemu który ją wywołał, niemniej jednak stanowić jego integralną część konieczną w realizacji projektu. Serwis powinien być jak najbardziej uniwersalny aby w łatwy spsoób mógł zostać wykorzystany w różnych niezależnych od siebie systemach komputerowych.