**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

# *Analiza możliwości adaptacji środowisk uruchomieniowych dla logiki biznesowej na platformach mobilnych.*

# *Analysis of possible adaptation or business logic runtimes for mobile platforms.*

Autor: *Tomasz Landowski*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr hab. Grzegorz J. Nalepa*

Kraków, 2014

*Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.*

*Serdecznie dziękuję żonie i córce za cierpliwość.*

# Spis treści

# 1. Wstęp

...

## 1.1. Cel pracy

Celem poniższej pracy jest ...

## 1.2. Struktura pracy

W rodziale 2 ...

# 2. Analiza problemu modelowania procesów biznesowych

Modelowanie procesów biznesowych jest pojęciem bardzo ogólnym, nie zawiera w sobie żadnych szczegółów technicznych, określa jedynie pewne specyficzne podejście do rozwiązywania problemów informatycznych. W podejściu tym podczas analizy, główny nacisk kładziony jest na wyłonienie procesów występujących w analizowanym problemie. Proces jest tutaj rozumiany jako zbiór następujących w określonej kolejności operacji prowadzących do osiągnięcia konkretnego celu. Najczęściej modelowanie procesów biznesowych jest rozważane w kontekście działania konkretnego przedsiębiorstwa, w którym wyłonienie procesów oraz odpowiednie nimi zarządzenie ma kluczowe znaczenie dla osiągnięcia sukcesu.

## 2.1. BPM

W literaturze pojęcie modelowania oraz zarządzanie procesami biznesowymi w kontekście przedsiębiorstw występuje pod skrótem BPM, którego angielskie rozwinięcie to Buisness Process Management (pl. Zarządzanie Procesami Biznesowymi). Idea BPM jest bardzo popularna i szeroko rozwijana w środowisku IT, powstała nawet organizacja pod nazwą *European Association of Buisness Process Managment* zajmująca się rozwojem i promocją BPM. Na stronie internetowej organizacji znaleźć możemy oficjalną definicje BPM, która mówi, że w skład Zarządzania Procesami Biznesowymi wychodzi:

– projektowanie,

– wykonywanie,

– dokumentacja,

– pomiar,

– monitorowanie,

– kontrola

zautomatyzowanych oraz niezautomatyzowanych procesów biznesowych. [1]

W środowisku IT zdarza się, że BPM traktowany jest jako osobna klasa systemów informatycznych, obok ERP, MES, CRM itd. Z punktu widzenia przedstawionej wyżej definicji trudno zgodzić się z takim

podejściem, można jednak zauważyć że BPM de facto może zostać wykorzystany do realizacji każdego z wymienionych rodzajów oprogramowania, lub posłużyć jako narzędzie integrujące wyżej wymienione klasy systemów, w celu stworzenia globalnego systemu zarządzania przedsiębiorstwem. [2]

## 2.2. Projektowanie BPM

Zdecydowanie jednym z najistotniejszych etapów w tworzeniu systemu opartego o procesy biznesowe jest etap projektowania. Jednym z najbardziej popularnych narzędzi do tego celu jest BPMN (*Buisness Process Modeling Notation*), która doczekała się już dwóch wersji. W kontekście niniejszej pracy magisterskiej notacja BPMN nie jest szczególnie ważna jednak została wspomniana ze względu na jej wykorzystanie w przykładach. Warto zwrócić uwagę na fakt, że BPMN nie jest w żadnym stopniu implementacją, jest to jedynie sposób wizualizacji procesu za pomocą diagramu w postaci obrazka.

## 2.3. Wykonywanie BPM

Jak zostało wspomniane na samym początku Modelowanie Procesów Biznesowych jest pojęciem ogólnym nie wskazującym konkretnej technologii, która ma posłużyć do wykonania procesów biznesowych. Z definicji można jednak wnioskować, że środowiskiem do wykonywania procesów biznesowych w przedsiębiorstwach na pewno nie powinna być prosta aplikacja desktopowa czy mobilna. Biorąc pod uwagę wachlarz zastosowań BPM nie trudno dojść do wniosku, że najbardziej odpowiednim środowiskiem dla procesów biznesowych są środowiska aplikacji rozproszonych. Powyższe stwierdzenie potwierdza przegląd istniejących systemów uruchomieniowych dla procesów biznesowych. W ogromnej większości są to aplikacje webowe.

## 2.4. Podsumowanie

Z punktu widzenia niniejszej pracy magisterskiej, głównym wnioskiem płynącym z powyższego opisu BPM jest skala problemów rozwiązywanych przez to podejście oraz ich usytuowanie w środowisku aplikacji rozproszonych.

# 3. Analiza aplikacji na platformy mobilne

Wzrost popularności smartphonów oraz urządzeń przenośnych takich jak tablet wpłynął na powstanie specyficznej klasy systemów informatycznych zwanych aplikacjami mobilnymi. Aplikacje mobilne cechuje ukierunkowanie na rozwiązywanie wąskiego rodzaju problemów przy pomocy urządzenia dostępnego dla użytkownika niemal 24h na dobę. Szczególną popularność zyskały aplikacje mobilne ukierunkowane na użytkowników (TODO jak nazwać użytkownika typu powszechny Jan Kowalski). Coraz większą popularność zyskują również aplikacje biznesowe ukierunkowane np. na wymianę danych pomiędzy pracownikami przedsiębiorstwa w celu rozwiązania konkretnego zadania.

## 3.1. Rodzaje aplikacji mobilnych

Ze względu na rodzaj zastosowania aplikacje mobilne można sklasyfikować w następujący sposób:

– Samodzielne aplikacje – aplikacje wykorzystujące tylko i wyłącznie lokalne zasoby urządzenia, korzystające z lokalnej bazy danych bez połączenia z systemami zewnętrznym, często nie wymagające połączenie do internetu. Dobrym przykładem takiej aplikacji może być prosty notatki.

– Aplikacje klienckie – są to aplikacje bazujące na komunikacji z systemami zewnętrznymi przy pomocy jakiegokolwiek interfejsu komunikacyjnego, najczęściej połączenia HTTP. Zazwyczaj aplikacje te wykorzystują również lokalną bazę danych umożliwiając użytkownikom pracę z aplikacji w przypadku braku połączenia z systemem zewnętrznym.

– Internetowe – najczęściej strony www, aplikacje te nie wykorzystują lokalnych zasobów urządzeń mobilnych, ich rolą jest udostępnienie prostego interfejsu użytkownika systemu dla użytkowników aplikacji mobilnych.

– Gry – szczególny przypadek samodzielnych aplikacji ukierunkowanych głównie na wykorzystanie lokalnych zasobów urządzenie w celach rozrywkowych, szczególnie eksploatowana w tego typu aplikacjach jest karta graficzna urządzenia.

## 3.2. Tryby online/offline

Najbardziej docenianym rodzajem aplikacji mobilnych z punktu widzenia biznesu są aplikacje klienckie. W przypadku tych aplikacji szalenie istotny jest problem połączenia z systemami zewnętrznymi. Wyróżniamy tutaj dwa tryby pracy aplikacji:

– Tryb online – urządzenie udostępnia przynajmniej jeden typ komunikacji bezprzewodowej (Wi-fi, Bluetooth, łącza podczerwieni (IrDa), GPRS). W trybie online aplikacje mobilne mają bezpośredni dostęp do zewnętrznych i zdalnych źródeł danych, innych urządzeń mobilnych lub stacjonarnych systemów komputerowych.

– Tryb offline – urządzenie ma bezpośredni dostęp tylko do lokalnie przechowywanych informacji. Dane te mogą być jednak synchronizowane z innymi urządzeniami w czasie krótkich sesji komunikacyjnych. Wymiana danych podczas synchronizacji może następować w obu kierunkach.

## 3.3. Cechy specyficzne dla aplikacji mobilnych

Opisując aplikacje mobilnie nie można zapomnieć o cechach specyficznych, które są szalenie ważne z punktu widzenia projektanta i programisty aplikacji mobilnych. To właśnie dzięki tym cechom aplikacje mobilne posiadają tę a nie inną specyfikę:

– Ograniczone zasoby sprzętowe – urządzenie przenośnie pod względem zasobów sprzętowych nigdy nie zastąpią komputerów stacjonarnych a tym bardziej urządzeń serwerowych. Właśnie ze względu na tę cechę aplikacje mobilne ukierunkowane są na rozwiązywanie małych - jednostkowych problemów. Projektant aplikacji mobilnych powinien szczególnie zwrócić uwagę na ten aspekt w przypadku projektowania interfejsu do komunikacji z systemami zewnętrznymi, aplikacje mobilne powinny wykorzystywać lekkie interfejsy komunikacyjne np. REST, aby jak najmniej obciążać kartę sieciową urządzenia.

– Uproszczony interfejs użytkownika dostosowany do ekranów dotykowych – cecha ta jest bardzo ważna głównie dla projektanta interfejsu użytkownika, ale ma również znaczący wpływ na rodzaj zadań rozwiązywanych za pomocą tych aplikacji.

– Przerwy w działaniu aplikacji – aplikacje mobilne w przeciwieństwie do innych rodzajów systemów są szczególnie narażone na przerwy w działaniu, najprostszym przykładem może być rozmowa telefoniczna przychodząca w trakcie pracy z aplikacją. Najczęściej w takich przypadkach aplikacje przechodzą w tryb uśpienia i nie ma pewności czy praca zostanie wznowiona. Programiści powinny pamiętać o zapamiętywaniu poszczególnych stanów aplikacji aby użytkownik nie tracił wykonanej pracy.

– Dostęp do GEO lokalizacji – aplikacje mobilne oprócz ograniczeń posiadają również wiele cech dodatkowych wyróżniających je wśród innych systemów, jedną z nich jest dostęp do GEO lokalizacji. Funkcjonalność ta bardzo szeroko wykorzystywana jest przez systemu zwane Context Aware Middleware.

– Dostęp do aparatu/kamery – jest to kolejna cecha rozszerzająca możliwości aplikacji mobilnych, może być wykorzystana np. jako narzędzie do skanowania kodów kreskowych, albo sposób dokumentacji pracy wykonanej przez użytkownika systemów mobilnych.

– Specyficzna architektura dla różnych systemów operacyjnych – bardzo wiele cech aplikacji mobilnych jest wspólna w przypadku wszystkich systemów operacyjnych dostępnych na rynku, trzeba mieć jednak świadomość, że jest również wiele cech specyficznych dla różnych platform mobilnych.

## 3.4. Komunikaty Push

Ze względu na zmienne warunki w dostępie do mediów komunikacyjnych, aplikacji mobilne wykorzystują w głównej mierze komunikacje pull, (czyli pobieranie informacji na żądanie) z systemami zewnętrznymi. Najpopularniejsze platformy mobilne jak Android, iOS oraz Windows Phone udostępniają również alternatywny sposób komunikacji - push.

Komunikacja push odbywa się jednak zawsze za pośrednictwem serwisów udostępnianych przez platformy mobilne do tego celu. Komunikacja ta posiada wiele ograniczeń a już na pewno najważniejszym z nich jest brak pewności że komunikat zostanie w ogóle dostarczony. Komunikacja push powinna być zatem wykorzystywana jedynie do notyfikacji użytkownika o jakimś zdarzeniu a nie do przesyłania informacji kluczowych dla działania aplikacji.

# 4. BPEL „inteligentne” narzędzie integracyjne

BPEL (*Business Process Execution Language*) a właściwie pełna nazwa WS-BPEL, której rozwinięcie to *Web Services Business Process Execution Language* jest językiem opartym o składnię XML, służącym do wykonywania procesów biznesowych w środowisku usług sieciowych. Język BPEL jest ustandaryzowany przez oranizacje OASIS (*Organization for the Advancement of Structured Information Standards*).

Organizacja OASIS definiuje język BPEL jako: *Umożliwiający użytkownikom opis aktywności procesów biznesowych jako usługi sieciowe oraz definicję w jaki sposób usługi te mogę być połączone między sobą w celu wykonania zadania.* [3]

Jak argumentuje OASIS, język BPEL powstał z konieczności stworzenia dodatkowej warstwy integracyjnej, która wykorzysta potencjał dostarczony przez usługi sieciowe oraz procesy biznesowe. Warto tutaj wspomnieć o historii powstania BPEL. Microsoft i IBM dostrzegając potrzebę stworzenia technologii pozwalającej definiować przepływ wywołań usług sieciowych stworzyli osobne, ale bardzo podobne języki, służące do tego celu, nazwane WSFL (*Web Service Flow Language*) oraz Xlang. W miare wzrostu popularności BPM zdecydowali się połączyć siły aby stworzyć wspólny standard (opisanych za pomocą WSDL. Zarówno Microsoft jak i IBM (zresztą nie tylko oni) dostrzegli, że architektura SOA (*Service Oriented Architecture*) wymaga wprowadzenia czegoś co pozwoli składać klocki jakimi są usługi sieciowe w jedną zgrabną całość i właściwie to było genezą powstania BPEL'a.

Dotychczasowy model komunikacji udostępniany przez usługi sieciowe był niewystarczający, głównie z powodu braku zachowania stanu podczas prostej komunikacji żądanie-odpowiedź. Komunikacja ponadto była głównie jednokierunkowa. Model komunikacji dla biznesu wymagał natomiast sekwencyjnej wymiany wiadomości pomiędzy wieloma węzłami, operacje wykonywane na węzłach mogły trwać bardzo długo, dlatego istnieje również konieczność obsługi tego typu procesów przy pomocy zapamiętywania stanu, oraz asynchronicznego wywoływania serwisów. [4]

## 4.1. Cechy języka BPEL

– Definiowane procesy biznesowe komunikują się z usługami sieciowymi przy pomocy WSDL 1.1 i same są usługami sieciowymi opisanymi przez WSDL 1.1.

– Do komunikacji pomiędzy usługami wykorzystywany jest protokół SOAP.

– Procesy zdefiniowane są w języku bazującym na XML.

– BPEL dostarcza funkcji do wykonywania prostych manipulacji na danych.

– Posiada wsparcie do identyfikowania instancji procesów.

– Posiada wsparcie dla podstawowego cyklu życia procesu.

– Wspiera transakcyjność przy wykorzystaniu podstawowych technik takich jak akcje kompensacyjne.

## 4.2. Orkiestracja czy choreografia?

BPEL jest językiem orkiestracji usług sieciowych, a nie choreografii. Podstawowa różnica polega na tym, że w choreografii usługi współpracują ze sobą bezpośrednio. W orkiestracji usługa komunikuje się wyłącznie z menedżerem orkiestracji (w tym przypadku: maszyna BPEL), który wie, gdzie i jak przekazać dalej komunikaty. Nie musi tym samym znać innych usług biorących udział w procesie.

Korzyści płynące z tego, że BPEL jest warstwa orkiestracji to [5]:

– podejście integracyjne do komunikacji pomiędzy aplikacjami, które uniezależnia aplikacje od siebie,

– umożliwia sterowanie przepływem, zarządzanie bezpieczeństwem oraz niezawodność komunikacji,

– co najważniejsze sposób zarządzanie i monitorowania procesów jest scentralizowany.

## 4.3. Silniki uruchomieniowe WS-BPEL

WS-BPLE dostarcza jedynie opisu w jaki sposób proces ma przebiegać oraz z jakimi usługami ma się komunikować, potrafi również zdefiniować proste operacje proceduralne jak tworzenie zmiennych, operacje przypisania, warunki czy pętle. Język sam w sobie jednak nie ma zbyt wielkiej wartości bez odpowiedniego środowiska uruchomieniowego. Procesy BPEL można traktować jako skrypty interpretowane, wykonywane przez maszyny procesów biznesowych w środowisku zgodnym ze standardem BPEL.

Najbardziej popularne środowiska uruchomieniowe BPEL to:

– ActiveVOS

– Apache ODE

– ExpressBPEL BPM

– BizTalk Server

– Open ESB

– Oracle BPEL Process Manager

– OW2 Orchestra

– Parasoft BPEL Maestro

– Petals BPEL Engine

– WebSphere Process Server

Do celów niniejszej pracy magisterskiej zostanie wykorzystany Apache ODE (*Apache Orchestration Director Engine*). Jest to narzędzie rozwijane przez społeczność Open Source, zgodne ze standardem WS-BPEL. Silnik Apache ODE został zaimplementowany przy wykorzystaniu języka Java, jest on w rzeczywistości standardową aplikacją webową uruchamianą na serwerach aplikacyjnych Java np. na serwerze Tomcat.

## 4.4. Podsumowanie

Należy pamiętać, że język BPEL nie jest jedynym słusznym rozwiązaniem służącym do implementacji procesów biznesowych. Bardzo często zdarza się, że projektanci systemów IT, którzy preferują podejście top-down design (pl. projektowanie od góry do dołu), zaczynając projektować system IT przy pomocy notacji BPMN. Następnie zaś próbują przełożyć modele procesów biznesowych na kod wykonywalny w postaci BPEL. Nie jest to jednak słuszne podejście. Procesy BPMN zawierają często bardzo wiele szczegółów opisujących zachowanie samych operacji jednostkowych, natomiast pomijają szczegóły techniczne dotyczące usług sieciowych. Zazwyczaj okazuje się, że usługa sieciowa która została by wygenerowana z diagramu BPMN w ogóle nie istnieje, a jej stworzenie może być mało sensowne.

Niestety nazwa BPEL jest w tym przypadku nie do końca trafna, ponieważ sugeruje, że BPEL jest językiem służącym do implementacji procesów biznesowych. W rzeczywistości jednak BPEL należy traktować jako narzędzie integracyjne. Narzędzie to ma służyć do definicji przebiegu komunikacji pomiędzy usługami sieciowymi, przy pomocy idei procesów biznesowych. Jest bardzo ważnym aby BPEL był stosowany właśnie w takich przypadkach, pozwoli to uniknąć poważnych błędów projektowych wynikających z zastosowania nieodpowiednich narzędzi.

# 5. BPEL vs Aplikacje mobilne

W rozdziałach poprzednich sporo uwagi zostało skupione na temacie modelowania procesów biznesowych. Został również przybliżony temat aplikacji mobilnych, nadszedł czas aby odnieść się do tematu niniejszej pracy magisterskiej i spojrzeć na zestawienie tych dwóch technologii.

W postawionym problemie jak zwykle kluczowym jest odpowiednie podejście do tematu, uświadomienie sobie ograniczeń i odpowiednie dopasowanie technologii. Ograniczeń jest tutaj bardzo wiele głównie od strony aplikacji mobilnych.

Przykładów na wykorzystanie modelowania biznesowego w środowisku aplikacji mobilnych można wymyślić bardzo wiele, tym bardziej w przypadku aplikacji klienckich przeznaczonych dla biznesu. W tym ostatnim, środowisko mimo, że ukierunkowane na niewielkie aplikacje jest środowiskiem rozproszonym, w którym do tej pory procesy biznesowe sprawdzały się doskonale.

## 5.1. Adaptacja środowisk uruchomieniowych na platformach mobilnych

Jednym z możliwych sposobów podejścia do tematu może być próba bezpośredniej adaptacji środowisk uruchomieniowych dla procesów biznesowych na platformach mobilnych. Bezpośrednia adaptacja w tym przypadku oznacza próbę uruchomienia jakiegoś lekkiego środowiska uruchomieniowego na smartphonie. Po krótkim zastanowieniu można powiedzieć, że pomysł może być realny w realizacji - zasoby sprzętowe smartphonów są coraz większe, języki programowania wykorzystywane do tworzenia aplikacji mobilnych są identyczne jak języki do tworzenia środowisk uruchomieniowych (Android - język Java, Windows Phone - język `C#`). Gdy jednak kontynuujemy przemyślenia bardzo szybko natrafiamy na mur. Jak wspomniano w rozdziale 3, komunikacja aplikacji mobilnych odbywa się przede wszystkim metodą pull, głównie ze względu na zmienne warunki dostępu do mediów komunikacyjnych. W jaki sposób zatem środowisko uruchomieniowe miałby odbierać komunikację z zewnątrz? Jest przecież jeszcze metoda push, dostarczana przez dostawców platform mobilnych. Skorzystanie z niej jest jednak kiepskim pomysłem, głównie ze względu na brak gwarancji dostarczenia komunikatu. Podejście adaptacyjne zatem nie jest dobrym pomysłem.

## 5.2. Wykorzystanie procesów biznesowych w systemach back end'owych

Jak wspomniano wyżej najwięcej zastosowań dla procesów biznesowych w kontekście aplikacji mobilnych można dostrzec w aplikacjach klienckich. Należy się zatem zastanowić w jaki sposób działają typowe aplikacje klienckie. Z powodu ograniczeń opisanych w poprzednim akapicie komunikacja zazwyczaj nie odbywa sie na zasadzie Peer-to-Peer, użytkownicy aplikacji klienckich jednak w jakiś sposób wymieniają między sobą informację. Jest to możliwe dzięki aplikacjom centralnym, zazwyczaj webowym, zwanych systemami back'endowymi. Aplikacje mobilne w takich przypadkach komunikują się jedynie z systemem back end'owym nie posiadając informacji o sobie nawzajem. W tego rodzaju sposobie komunikacji można dostrzec bardzo wiele cech wspólnych z warstwą orkiestracji opisaną w rozdziale 4. Można zatem wyciągnąć wniosek, że procesy biznesowe w kontekście aplikacji mobilnych zostaną najefektywniej wykorzystane właśnie po stronie systemu back end'owego.

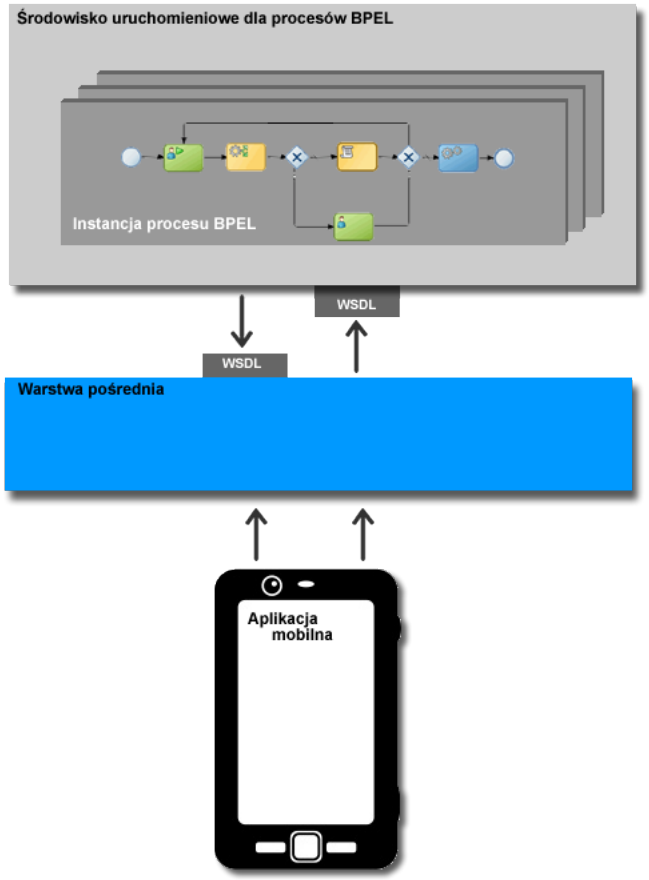
Podejście to można nazwać podejściem integracyjnym, w przypadku skorzystania z języka BPEL można sobie wyobrazić, że zrealizowany w ten sposób system back end'owy oprócz obsługi żądań aplikacji mobilnych mógłby komunikować się z innymi systemami przy wykorzystaniu usług sieciowych. Stworzony w ten sposób proces mógłby kontrolować przebieg komunikacji między aplikacjami mobilnymi traktując je jednocześnie na równi z innymi systemami. Niestety kolejny raz mimo sensownego pomysłu występują problemy w realizacji, mało tego ograniczenie kolejny raz jest takie samo. Proces BPEL jak opisano w rozdziale 4, do komunikacji poszczególnych węzłów wykorzystuje usługi sieciowe udostępniające odpowiednie interfejsy WSDL. Aplikacje mobilne nie są w stanie jednak udostępnić takich usług.

W tym przypadku istnieje sposób na poradzenie sobie z tym problemem, można stworzyć pewnego rodzaju adapter w postaci aplikacji webowej będącej warstwą pośrednią między procesami BPEL a aplikacjami mobilnymi. Taka warstwa pośrednia miałby za zadanie z jednej strony udostępnienie usług sieciowych dla procesów biznesowych a z drugiej strony realizowała by komunikacje z aplikacjami mobilnymi w celu przekazania żądań pochodzących od procesów biznesowych.

Opisanemu powyżej sposóbowi wykorzystania procesów biznesowych w aplikacjach mobilnych została poświęcona niniejsza praca, zostanie on dokładniej opisany w kolejnych rozdziałach.

# 6. Propozycja wykorzystania warstwy pośredniej

W rozdziale poprzednim, podczas analizy sposobu działania klienckich aplikacji mobilnych, wyciągnięto wniosek, że najbardziej odpowiednim miejscem do wykorzystania procesów biznesowych, będą aplikacje back end'owe. Stwierdzono również, że w celu realizacji takiego rozwiązania konieczne jest zastosowanie dodatkowej warstwy pośredniej, realizującej komunikacje aplikacji mobilnych z procesami biznesowymi. Głównym powodem stosowania warstwy pośredniej jest brak możliwości udostępnienie przez aplikacje mobilne, usług sieciowych z którymi proces mógłby się komunikować.




Rysunek 6.1: Poglądowy schemat działania warstwy pośredniej

Na rysunku 6.1. przedstawiono poglądowy schemat działania opisanego powyżej rozwiązania, ma on

na celu lepsze zobrazowanie problemu i wyłonienie funkcjonalności, które warstwa pośrednia musi realizować. Na schemacie można dostrzec trzy rodzaje systemów, środowisko uruchomieniowe dla procesów biznesowych, warstwę pośrednią oraz aplikację mobilną. Strzałki umieszczone na schemacie przedstawiają kierunki komunikacji pomiędzy poszczególnymi systemami. Komunikacja między aplikacją mobilną a warstwą pośrednią zawsze odbywa się z inicjatywy aplikacji mobilnej, jest to związane ze specyfiką dostępu aplikacji do mediów komunikacyjnych. Wynika z tego, że jedną z funkcjonalności, które musi realizować warstwa pośrednia jest buforowanie żądań pochodzących z procesów biznesowych do momentu aż aplikacja mobilna się po nie zgłosi sama. Aplikacje mobilne są specyficzne pod jeszcze jednym względem - poszczególne instancje aplikacji uruchamiane są zawsze na urządzeniach personalnych. Bardzo mocny akcent zatem kładziony jest tutaj na samych użytkowników. Gdy przyglądniemy się procesowi BPEL to możemy dostrzec, że wątek ludzki jest tutaj zupełnie pomijany. Proces komunikuje się z usługą sieciową jedynie w celu rozwiązanie jakiegoś konkretnego zadania i zazwyczaj oczekuje na rezultat

## 6.1. BPEL4People

Jak wspomniano powyżej procesy WS-BPEL traktują użytkowników po macoszemu, skupiają się przede wszystkim na sekwencji wykonywania kolejnych operacji. Problem dostrzegli projektanci z zespołów IBM oraz SAP, a proponowaną przez nich odpowiedzią jest powstanie rozszerzenia do BPEL nazwanego BPEL4People.

Twórcy standardu argumentują powstanie kolejnej specyfikacji koniecznością uwzględnienia aspektu ludzkiego w procesach BPEL. Podają szereg przykładów, w których interakcja procesu z człowiekiem jest konieczna. Jednym z przykładów jest konieczność dokonywania decyzji akceptacji lub odrzucenia wyniku jakiejś operacji. Innym przykładem jest wprowadzanie dodatkowych danych niezbędnych dla procesu za pomocą formularzy. [6, str. 4]

Specyfikacja opiewa w szereg bardzo ciekawych pomysłów integracji interakcji użytkowników z procesami BPEL. Porusza problem przejścia pomiędzy serwisami dostosowanymi dla ludzi a serwisami zautomatyzowanymi, zwraca uwagę na problem monitorowania zadań przeznaczonych dla ludzi, opisuje problem terminów ostatecznych wykonania zadań oraz eskalacji w przypadku ich przekroczenia.[6, str. 6]

### 6.1.1. WS-HumanTask

Bezpośrednio powiązany z rozszerzeniem BPEL4people jest temat usług sieciowych dla ludzi.

// TODO:

### 6.1.2. Implementacje

Z punku widzenia proponowanego w niniejszej pracy magisterskiej rozwiązanie zastosowanie BPEL4People rozwiązuje szereg problemów związanych ze spersonalizowaną specyfiką aplikacji mobilnych. Kolejne instancje aplikacji można utożsami z użytkownikami. Warstwa pośrednia udostępniałaby w takim podejściu WS-HT, z którymi komunikowałby się proces BPEL4People. Niestety rzeczywistość w tym przypadku okazuje się brutalna. Rozszerzenie do BPEL jeśli nie zostaje zaimplementowane jest bezwartościowe. W przypadku BPEL4People mimo tego że standard istnieje już 9 lat, nie został jeszcze wdrożony w żadnym z popularnych środowisk uruchomieniowych. Warstwa pośrednia powinna więc, samodzielnie uwzględniać aspekt ludzki pomijany przez procesy WS-BPEL.

Mimo braku implementacji można dostrzec pewne korzyści płynące z powstania rozszerzenia BPEL4People. Jest nimi z pewnością szereg pomysłów dotyczących rozwiązania problemu dystrybucji żądań pochodzących z procesów WS-BPEL między różnymi użytkownikami. W niniejszej pracy magisterskiej niejednokrotnie skorzystamy z tej wiedzy.

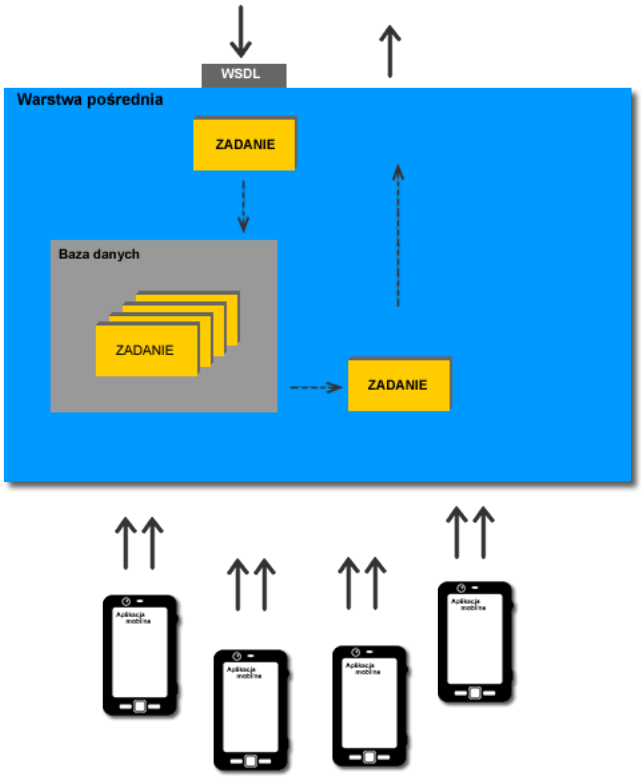
## 6.2. Zadania

Pierwszym z pomysłów zaczerpniętych z BPEL4People jest pomysł na traktowanie kolejnych żądań pochodzących z procesów WS-BPEL jako zadań, a aplikacji mobilnych jako zespołu specjalistów potrafiących rozwiązać te zadania. Podejście to rozwiązuje problem konieczności buforowania żądań procesu ze względu na komunikacje pull od strony aplikacji mobilnej.

Na rysunku 6.2. przedstawiono poglądowy schemat wykorzystania zadań w warstwie pośredniej. Strzałkami przerywanymi został oznaczony podstawowy cykl życia zadania. Po otrzymaniu żądanie od procesu BPEL warstwa pośrednia utworzy nowe zadanie i zapisze go w bazie danych. Aplikacje mobilne korzystając z lekkiego interfejsu komunikacyjnego będą potrafiły wyciągać zadania z bazy danych w celu ich rozwiązania. Po rozwiązaniu zadania, następować będzie wysłanie odpowiedzi do procesu biznesowego.

### 6.2.1. Dystrybucja zadań

Przedstawiony powyżej schemat nie jest jednak kompletny. Oczywistym jest, że aby proces biznesowy miał sens powinien wykonywać więcej niż jedną operację, dla każdej operacji w proponowanym rozwiązaniu istnieć będzie zupełnie inna klasa zadań. Nie trudno dojść do wniosku, że każda z tych klas obsługiwana może być przez zupełnie różne grupy użytkowników. Na przykład niemal w każdym przedsiębiorstwie zadania dyrektora są kompletnie inne od zadań zwykłego pracownika, ale są one między sobą w jakiś sposób powiązane na przykład występują w tym samym procesie. Wnioskiem z tego płynącym jest konieczność wprowadzenia w warstwie pośredniej mechanizmu przydzielania zadań poszczególnym użytkownikom. W przypadku tym ponownie rozszerzenie BPEL4People przewidziało ten




Rysunek 6.2: Poglądowy schemat wykorzystania zadań w warstwie pośredniej.

problem i zaproponowało rozwiązanie. Wprowadza ono do każdej klasy zadań, definicje grupy docelowych adresatów.

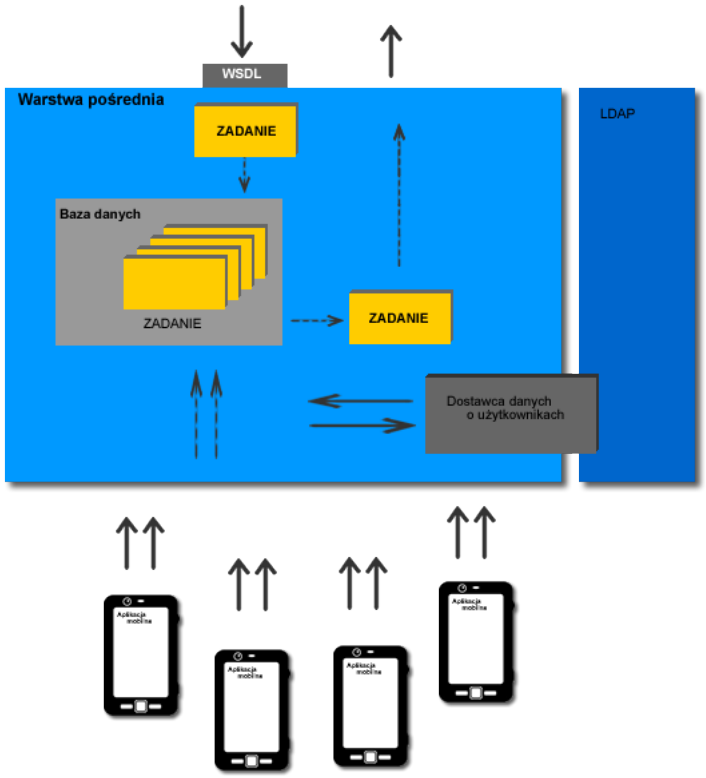
Na rysunku 6.3. przedstawiono schemat warstwy pośredniej uzupełniony o zarządzanie użytkownikami. Aplikacje mobilne podczas rozwiązywania zadań będą przedstawiać się przy pomocy nazwy użytkownika. Przed pobraniem zadań z bazy danych dostawca danych o użytkownikach dostarczy informacji niezbędnych do określenia zadań specyficznych dla danego użytkownika. Sposób ten zapewni, że zadania należące do dyrektora przedsiębiorstwa nie trafią do zwykłych pracowników.

Zastosowanie dodatkowej warstwy nazwanej Dostawcą danych o użytkownikach jest tutaj celowe, i będzie wymagać osobnej implementacji dla różnych rodzajów metod przechowywania danych o użytkownikach. Nie koniecznie użytkownicy musza pochodzi z systemu katalogowego takiego jak LDAP. Możliwości przechowywania danych o użytkownikach jest bardzo wiele, dla prostych systemów lista użytkowników może być nawet predefiniowana.

#### 6.2.1.1. Context aware middleware

Dodatkowym atutem udostępnienia możliwości implementacji dostawcy jest możliwość integracji adaptera z systemami zwanymi Context aware middleware.

Context aware middleware jest oprogramowaniem które za pomocą danych dostarczanych z możliwie jak największej ilości źródł, stara się rozwiązać zadany problem. Rozwiązanie to bardzo szerokie




Rysunek 6.3: Schemat warstwy pośredniej uzupełniony o zarządzanie użytkownikami.

zastosowanie znajduje głównie w aplikacjach mobilnych, dlatego że kontekst w przypadku tych aplikacji zmienia się bardzo dynamicznie. Architektura skupia się przede wszystkim na dostarczeniu informacji opisujących takie wymiary jak: - miejsce, Gdzie ? - czas, Kiedy? - stan np. Jaka jest temperatura otoczenia? - podmiot, Kto?

Bazując na tych danych stara się rozwiązywać dany problem w najbardziej optymalny sposób, np. przydzielając zadania osobie która jest najbliżej, która jest najmniej obciążona, która jest aktualnie w pracy itd.

Aby zastosować Context aware middleware w warstwie pośredniej wystarczy odpowiednia implementacja dostawcy danych o użytkownikach. Uzyskamy dzięki takiemu rozwiązaniu potężne narzędzie potrafiące dystrybuować zadania w bardzo optymalny sposób zapewniając ich szybką realizację.

### 6.2.2. REST API

Interfejs komunikacji warstwy pośredniej od strony procesów biznesowych został wymuszony przez specyfikację BPEL. Aby skompletować projekt warstwy pośredniej konieczne jest zdefiniowanie interfejsu komunikacyjnego od strony aplikacji mobilnych.

Interfejs ten powinien przede wszystkim rozwiązywać podstawowy problem występujący podczas rozwiązywania zadań przez wielu użytkowników jakim są konflikty. Nie może on dopuścić do sytuacji w której jedno zadanie byłoby rozwiązywanie przez wielu użytkowników jednocześnie. W celu rozwią-

zanie tego problemu konieczne jest wprowadzenie statusów zadań. W proponowanej warstwie pośredniej będą występowały następujące statusy zadania:

– gotowe do wykonania (ang. ready) – zadanie zostanie oznaczone tym statusem tuż po utworzeniu zadania. Użytkownik aplikacji mobilnych poszukując zadań do wykonania otrzyma listę zadań oznaczonych właśnie tym statusem.

– przypisane (ang. climed) – status ten oznacza, że któryś z użytkowników zadeklarował chęć jego wykonania. Będzie ono od tej pory dostępne tylko dla tego użytkownika. Pozostali użytkownicy poszukujący zadań do wykonania go nie zobaczą.

– rozwiązane (ang. completed) – statusem tym oznaczone będą zadania zrealizowane z sukcesem, których rezultat został odesłany do procesu biznesowego.

– błąd podczas rozwiązywania (ang. failed) – statusem tym oznaczone zostaną wszystkie zadania, podczas których rozwiązania wystąpił problem techniczny.

Znając poszczególne statusy nadszedł czas na określenie interfejsu komunikacyjnego z aplikacjami mobilnymi. Ponieważ wśród aplikacji mobilnych najbardziej popularnym medium komunikacyjnym jest połączenie sieciowe na pewno najrozsądniejszym pomysłem będzie z niego skorzystanie. Jak wspomniano w rozdziale 3, podczas projektowania aplikacji mobilnych należy zwrócić uwagę na to aby wybierane technologie nie obciążały zbytnio ograniczonego pod względem zasobów urządzenia mobilnego. Dobrym wyborem w takim wypadku będzie REST.

#### 6.2.2.1. Usługi sieciowe RESTFul

// TODO

#### 6.2.2.2. Definicja interfejsu

Warstwa pośrednia do komunikacji z aplikacjami mobilnymi będzie udostępniać następujące funkcje:

– lista zadań do wykonania – metoda nie będzie przyjmować żadnych parametrów dodatkowych, kontekst użytkownika zostanie określony na podstawie nagłówków http wymaganych przez HTTP Basic Authentication.

– przypisz do zadania – metoda jako parametr będzie przyjmować unikalny identyfikator zadania

– zrezygnuj z zadania – metoda jako parametr będzie przyjmować unikalny identyfikator zadania

– rozwiąż zadanie – metoda jako parametr będzie przyjmować unikalny identyfikator zadania oraz wynik rozwiązania zadania.

## 6.3. Wybrane technologie

Niemal w każdym projekcie informatycznym kluczową sprawą jest dobór odpowiednich technologii. Do realizacji warstwy pośredniej potrzebna będzie technologia webowa potrafiąca udostępnić usługi sieciowe, potrafiąca komunikować się z bazą danych i posiadająca wsparcie dla technologii REST. Niemal wszystkie popularne języki programowania spełniają każdy z tych warunków. Należy zatem pójść o jeden krok dalej i poszukać gotowego frameworka (zestawu gotowych bibliotek i wzorców postępowania), który przy wykorzystaniu jednego z tych języków będzie potrafił zrealizować przynajmniej cześć zadań. Wybór w niniejszej pracy magisterskiej padł na Spring Framework zaimplementowany w języku Java.

### 6.3.1. Spring Framework

Spring Framework zapewnia kompleksowy model programistyczny i konfiguracyjny dla nowoczesnych aplikacji biznesowych opartych na języku Java. Kluczowym elementem jest wsparcie infrastrukturalne na poziomie aplikacji: Spring skupia się na "hydrauliceąplikacji dla przedsiębiorstw, tak, że zespoły mogą skupić się na logice biznesowej na poziomie aplikacji, bez konieczności określania środowisk programowania. [**?**]

– Spring Web Services – w przypadku warstwy pośredniej jedną z głównych funkcjonalności koniecznych do realizacji jest udostępnienie usług sieciowych do komunikacji z procesami biznesowymi. Spring Framework posiada specjalnie do tego celu przygotowany moduł nazwany Spring Web Services. Moduł ten koncentruje się tworzeniu usług sieciowych na podstawie dokumentów SOAP przy wykorzystaniu stylu, umowa-najpierw(ang. contract-first). Styl ten polega na definicji interfejsów w postaci plików WSDL, przed stworzeniem konkretnej implementacji usługi. Jak argumentuje Spring styl ten pozwala na tworzenie elastycznych usług sieciowych przy wykorzystaniu jednego z wielu sposobów na manipulacje zawartością XML. [**?**]

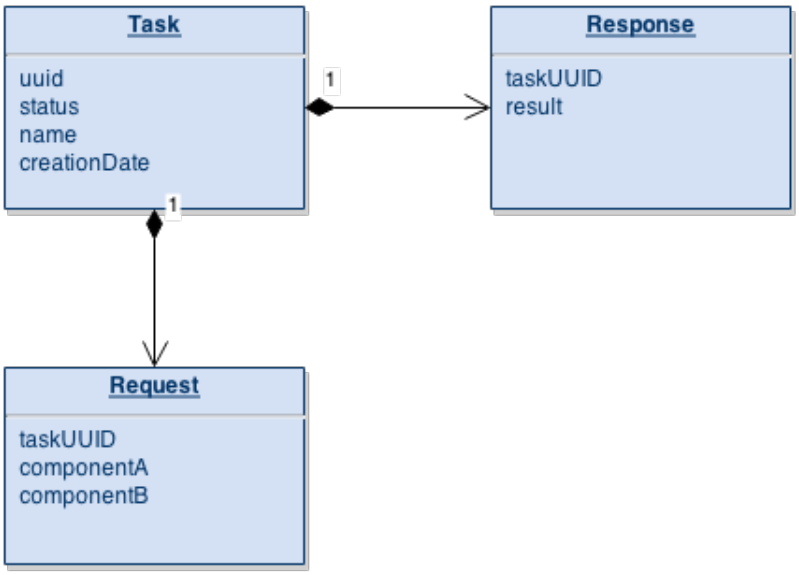
– Spring MVC – moduł MVC jak sama nazwa wskazuje jest przeznaczony do tworzenia aplikacji opartych o jeden z najbardziej popularnych wzorców projektowych Model Widok Kontroler. Spring MVC jest przeznaczony przede wszystkim do zastosowania wzorca w aplikacjach webowych gdzie do komunikacji wykorzystywany jest protokół HTTP. W przypadku aplikacji webowych widokiem zazwyczaj jest dynamicznie generowany kod HML, nie jest to jednak reguła. W przypadku warstwy pośredniej MVC zostanie wykorzystane do przygotowania REST API. Widokiem w takim wypadku będą dane opisane za pomocą dokumentów JSON.

– Spring Data – Oprócz wyżej wymienionych modułów, Spring Framework dostarcza również wsparcie do obsługi komunikacji z bazą danych, najczęściej do tego celu wykorzystując popularne ORM'y ( Biblioteki do mapowania obiektowo relacyjnego) jak Hibernate. W przypadku warstwy pośredniej istnieje konieczność przechowywania zadań w bazie danych dlatego Spring Data również zostanie wykorzystany.

– Spring Secutiry – W niniejszym rozwiązaniu uwierzytelnienia użytkowników REST API będzie odbywało się za pomocą HTTP Basic Authentication. Spring posiada również dodatkowe wsparcie do obsługi bezpieczeństwa przy pomocy HTTP Basic Authentication, wsparcie to udostępnione jest za pomocą modułu Spring Security.

### 6.3.2. MongoDB

Bardzo ważną decyzją jest również wybór odpowiedniej bazy danych. Wymagania dotyczące warstwy pośredniej jasno określają, że głównym modelem przechowywanym w bazie danych będzie zadanie. Zadanie oprócz podstawowych informacji takich jak identyfikator, status, data utworzenia itd. będzie posiadał wszelkie informacje potrzebne do jego utworzenia, oraz jeśli zadanie zostanie wykonane posiadać będzie również informacje o rozwiązaniu. W przypadku skorzystania z relacyjnej bazy danych klaruje nam się przykładowy schemat bazy danych przedstawiony na rysunku 6.4.




Rysunek 6.4: Diagram ERD przykładowego schematu bazy danych relacyjnej.

W przypadku pojedynczego zadania schemat może okazać się słuszny, problem powstaje gdy warstwa pośrednia obsługuje więcej niż jedno zadanie. W takim przypadku każdy kolejny rodzaj danych wejściowych poszczególnych zadań powinien znajdować się w osobnej tabeli. Mało tego dane wejściowe zadań nie muszą ograniczać się jedynie do płaskiej struktury danych. Mogą one zawierać argumenty będące osobnymi strukturami danych. W takich przypadkach, z punktu widzenia normalizacji bazy danych, każda z tych struktur powinna również znajdować się w osobnych tabelach. Taka postać rzeczy ma bardzo negatywny wpływ na poziom komplikacji zapytań wyciągających zadania dla poszczególnych użytkowników. Mogą istnieć przypadki w których na podstawie danych wejściowych będzie potrzeba ograniczenia adresatów zadania, np. wybrania użytkowników w obrębie kilku kilometrów od adresu znajdującego się w danych wejściowych.

Sposobem na rozwiązanie problemu nie koniecznie płaskiej struktury danych wejściowych może być zastosowanie typu XML do przechowywania tych danych. W takich przypadkach dane wejściowe zostałyby serializowane do dokumentów xml i w takiej postaci zapisane w bazie danych. Kolejny raz problemem są jednak skomplikowane zapytania do bazy danych.

Odpowiedzią na postawione wymagania są dokumentowe bazy danych. Przechowują one dane w postaci kolekcji dokumentów. Dokumenty przechowywane w bazie danych mogą posiadać dowolnie skomplikowaną strukturę danych, a odpowiednio przygotowane zapytania będą potrafiły filtrować listę tych dokumentów na podstawie wszystkich argumentów znajdujących się w tej strukturze.

Przykładowy dokument bazy danych MongoDB może przedstawiać się następująco:

// TODO

Zapytanie wyciągające wszystkie dokumenty dla których wartość argumentu ... jest równa ... przedstawia się następująco:

// TODO

W przypadku relacyjnej bazy danych powyższa struktura wymagała by stworzenie ... tabel a zapytanie które dałby by dokładnie taki sam rezultat wyglądało by następująco:

// TODO

### 6.3.3. Maven

Kolejna technologią wykorzystaną do realizacji idei przestawionej w niniejszej pracy magisterskiej jest Maven. Wykorzystanie tej technologii wynika z decyzji wykorzystania języka Java. Maven jest narzędziem wspomagającym budowanie projektów stworzonych w języku Java. Odpowiada on za skompletowania wszystkich wymaganych bibliotek zdefiniowanych w odpowiednim pliku konfiguracyjnym - pom.xml. Biblioteki te pobierane są ze specjalnie przygotowanych do tego celu repozytoriów dostępnych w sieci, na lokalną maszynę. Biblioteki te są następnie wykorzystywane w procesie budowania plików wynikowych.

## 6.4. Sprawienie rozwiązania generycznym

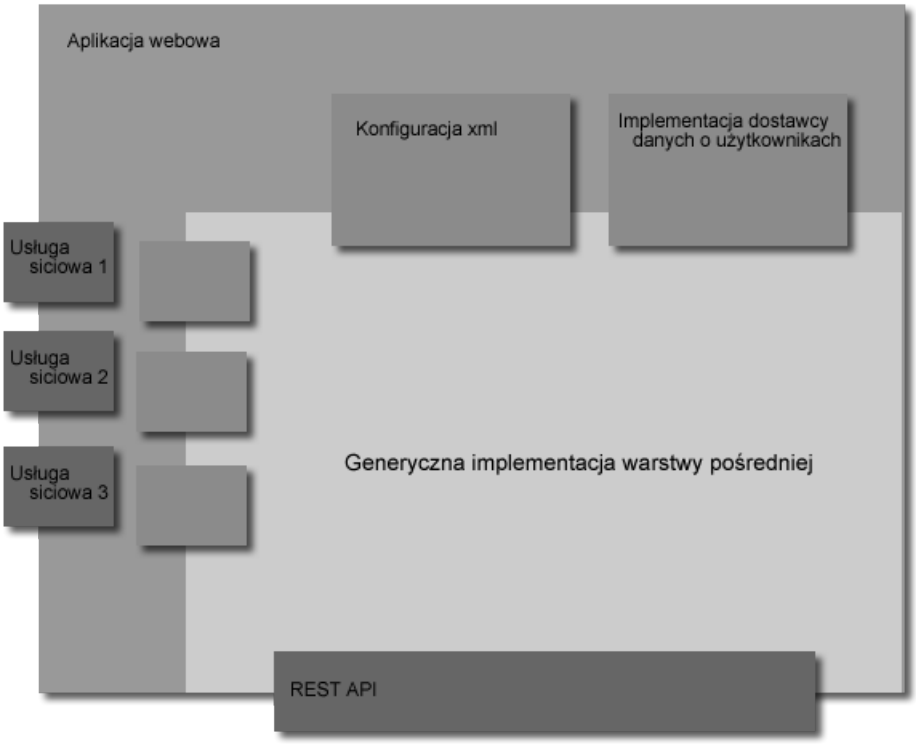
Przedstawiony do tej pory model warstwy pośredniej skupia się przede wszystkim na problemie komunikacji pomiędzy procesem biznesowym oraz aplikacją mobilną, opisuje również sposób przechowywania i dystrybucji zadań. Model jest przy okazji bardzo ogólny i możliwy do zastosowania bez względu, na rozwiązywany problem. Nie ma w nim mowy o konkretnych przypadkach zastosowań.

Dobrym pomysłem będzie więc stworzenie rozwiązania na tyle elastycznego aby mogło być wykorzystywane bez względu na rozwiązywany problem. W środowisku IT taki model postępowanie nazywany jest generycznym. W przypadku warstwy pośredniej sposób zarządzania zadaniami jest taki sam dla wszystkich rodzajów zadań - utworzenie zadania po otrzymaniu żądania od procesu, udostępnienie zadania aplikacją mobilnym, odpowiedź do procesu. Główną różnicą są dane wejściowe różnych żądań oraz sposób dystrybucji zadań pomiędzy aplikacjami na podstawie tych danych. Problem różnych

struktur danych wejściowych dla różnych zadań rozwiązuje nam wybór bazy danych MognoDB, który będzie potrafił zapisać te struktury bez dodatkowych konfiguracji, oraz będzie potrafił je przeszukiwać za pomocą prostych zapytań.

Problem różnego sposobu dystrybucji danych rozwiążemy za pomocą pliku konfiguracyjnego, który będzie opisywał poszczególne zadania, oraz definiował jakie grupy użytkowników są uprawnione do ich rozwiązywania.

Opisany model można zobrazować za pomocą schematu widocznego na rysunku 6.5.



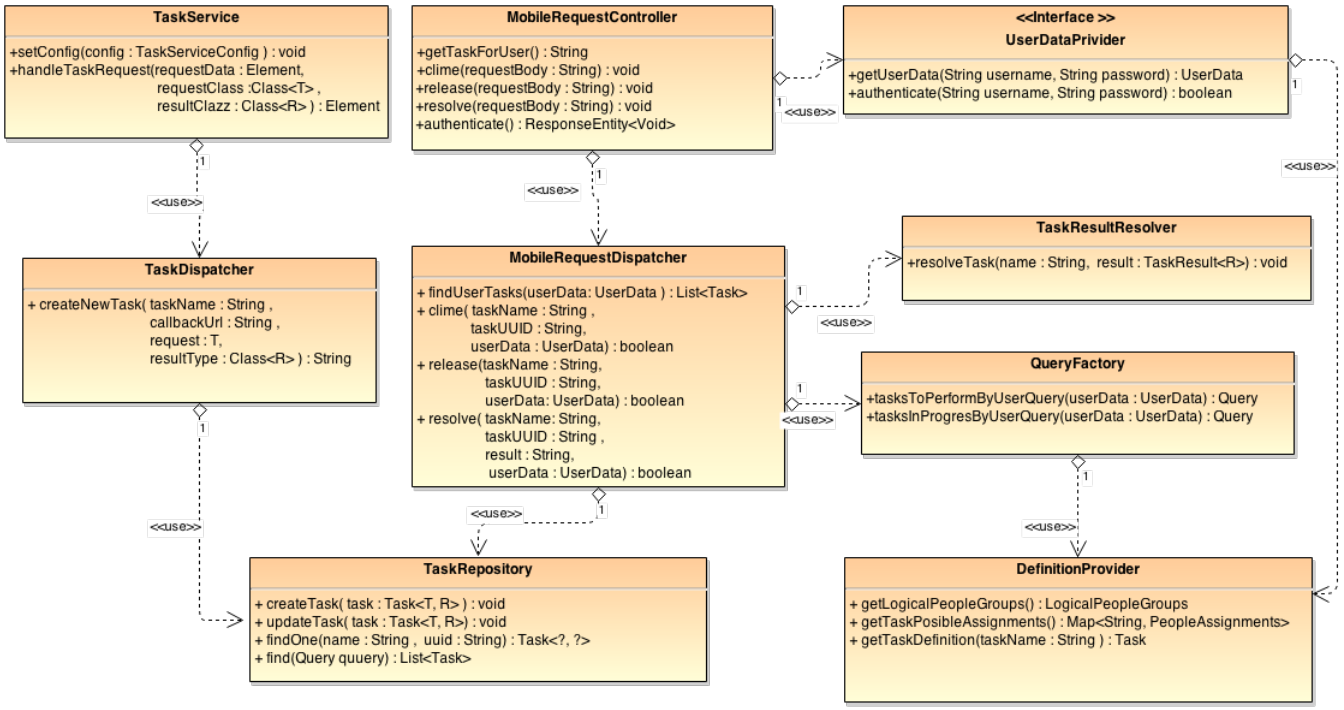

Rysunek 6.5: Poglądowy schemat wydzielenia biblioteki odpowiedzialnej za obsługę warstwy pośredniej.

Podsumowując, opisane w niniejszej pracy magisterskiej rozwiązanie będzie generyczną biblioteka służącą do tworzenie aplikacji webowej. Aplikacja ta będzie warstwa pośrednią pomiędzy procesem biznesowym a aplikacjami mobilnymi.

## 6.5. Implementacja

Implementacja biblioteki zostanie przedstawiona w sposób schematyczny, zostanie przedstawiony projekt klas występujących w rozwiązaniu wraz z ich opisem. W następnej część zostaną przedstawione diagramy sekwencji najważniejszych funkcjonalności rozwiązania, opisujące przebieg wywołań kolejnych funkcji zdefiniowanych klas.

### 6.5.1. Diagram klas




Rysunek 6.6: Diagram klas implementacji warstwy pośredniej.

Diagram klas przedstawiony na rysunku 6.6. przedstawia zestaw klas, które wymagają dodatkowego objaśnienia:

– TaskService – jest to klasa odpowiedzialna za obsługę żądań pochodzących z procesów biznesowych. Każdy rodzaj zadania powinien posiadać własną instancję tej klasy, która z kolei powinna być wykorzystywana przez usługę sieciową odpowiedzialną za jego obsługę. Klasa posiada dwie metody, setConfig odpowiedzialną za ustawienie podstawowych informacji o zadaniu, które będzie obsługiwać instancja tej klasy, takich jak nazwa, lista klas definiujących dane wejściowe i wyjściowe zadania. Kolejną metodą jest metoda handleTaskRequest metoda ta przyjmuje jako parametr obiekt typu Element zawierający dane wejściowe w postaci XML. Metoda przetwarza te dane na odpowiednie klasy i zapisuje zadanie w bazie danych do późniejszej realizacji. Zwraca ona obiekt typu Element zawierający informacje o unikalnym identyfikatorze przypisanym zadaniu, obiekt ten jest plikiem xml który powinna zwrócić usługa sieciowa. Metoda handleTaskRequest powinna zostać wywołana przez usługę sieciową odbierającą żądanie od procesu biznesowego.

– TaskDispatcher – Klasa odpowiedzialna za przetwarzanie, przyjętych przez klasę TaskService zadań. Posiada ona jedną metoda służącą do zapisu zadania przetworzonego przez TaskService, nazwaną createNewTask.

– TaskRepository – repozytorium zadań odpowiedzialne za obsługę bazy danych. Posiada metody służące do zapisu i edycji zadań oraz do wyszukiwania zarówno pojedynczych zadań, jak i ich listy.

– MobileRequestController – klasa przeznaczona do udostępnienia REST API. Posiada ona metody udostępniane przez interfejs przeznaczony do aplikacji mobilnych. Klasa ta korzystając ze Spring Security, odnajduje kontekst aktualnego użytkownika, pobiera informacje o nim za pomocą UserDataProvidera przekazując następnie kontrole MobileRequestPrivider'owi.

– UserDataProvider – interfejs służący do określenia metod które musi implementować dostawca informacji o użytkowniku. Implementacje UserDataProvidera będą wykorzystywana między innymi przez Spring Security.

– DefinitionProvider – klasa to odpowiedzialna jest za deserializację pliku xml zawierającego konfiguracje poszczególnych zadań, oraz dostarczenie tej konfiguracji do pozostałych klas warstwy pośredniej.

– MobileRequestDispatcher – klasa odpowiedzialna za obsługę żądaniami pochodzącymi z aplikacji mobilnych. Współpracuje z klasami przeznaczonymi do generowania zapytań, wyciągania danych z bazy oraz odpowiedzialnymi za odesłanie rozwiązania do procesów biznesowych.

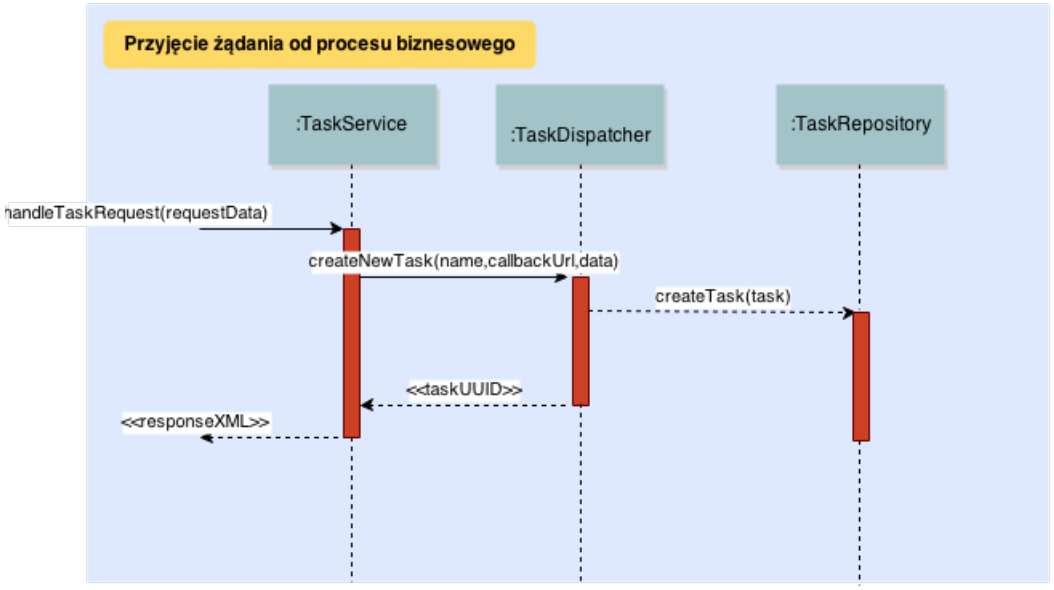
– QueryFactory – głównym zadaniem tej klasy jest przygotowanie zapytań pozwalających odnaleźć zadania dostępne dla danego użytkownika. Zapytania generowane są na podstawie konfiguracji xml oraz danych o użytkowniku.

– TaskResultResolver – klasa posiada tylko jedną metodę nazwaną resolveTask służąca do wysłania odpowiedzi do procesu biznesowego zawierającej rezultat wykonania zadania.

### 6.5.2. Diagramy sekwencji

Sposób działania powyżej opisanych klas zostanie przedstawiony za pomocą diagramów sekwencji. Diagramy przedstawiają sekwencje kolejnych wywołań funkcji w celu realizacji funkcjonalności dostarczanych przez warstwę pośrednią.

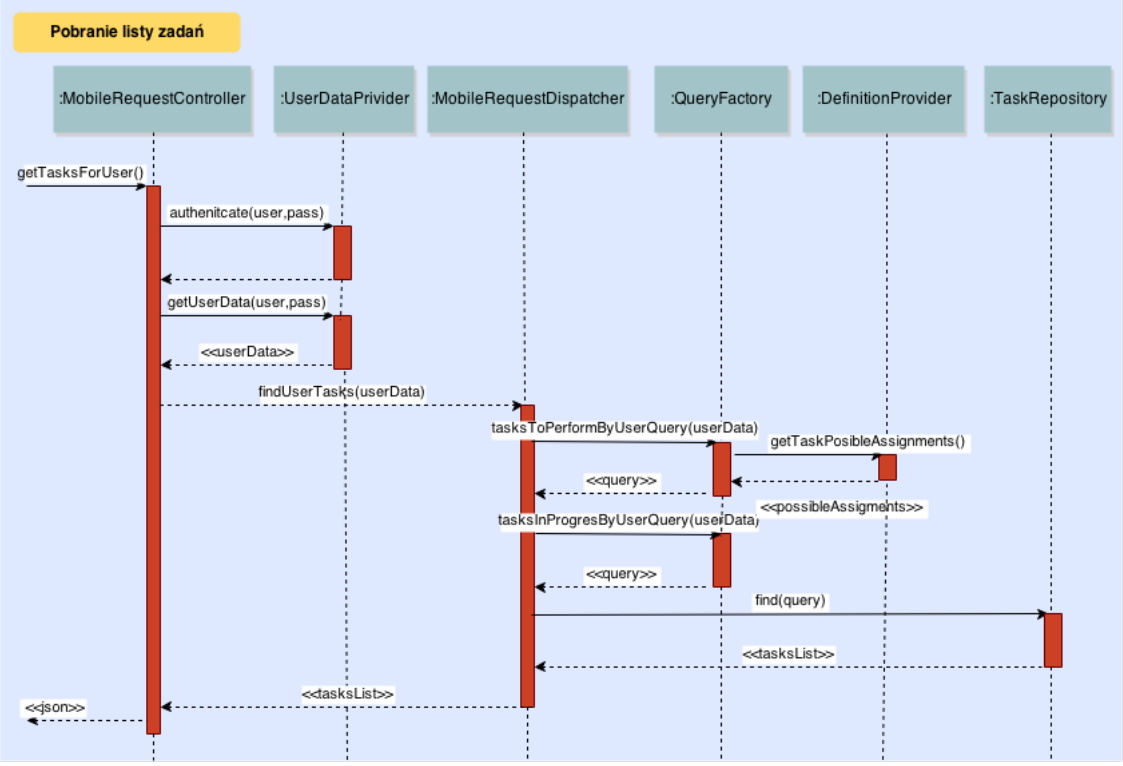
Na rysunku 6.7 zobaczyć można diagram sekwencji funkcjonalności -przyjęcie żądania od procesu biznesowego. Usługa sieciowa korzystając z konkretnej instancji klasy TaskService uruchamia metodę handleTaskRequest przekazując do niej obiekt zawierający dokument XML z danymi wejściowymi. Wewnątrz metody, dokument XML zostaje zdeserializowany do postaci obiektów podanych w konfiguracji klasy TaskService. Zostaje również wydobyty adres url usługi przyjmującej ostateczny rezultat zadania. Sterowanie przekazywane jest następnie do klasy TaskDispatcher w celu utworzenie nowego zadania w bazie danych. Metoda createNewTask klasy TaskDispatcher oprócz stworzenia nowego dokumentu w kolekcji w bazie danych MongoDB za pomocą repozytorium TaskRepository, odpowiedzialna jest za




Rysunek 6.7: Diagram sekwencji obsługi żądania pochodzącego z procesu BPEL.

generację unikalnego identyfikatora w postaci klucza GUID. Unikalny identyfikator jest następnie zwracany ponownie do TaskService'u gdzie zostaje opakowany w dokument XML, który zostanie zwrócony do procesu biznesowego. Wysłanie unikalnego klucza do procesu jest w tym przypadku krytyczne, po stronie silnika uruchomieniowego procesów biznesowych BPEL będzie istniało bardzo wiele instancji procesów w różnym stadium wykonania. Komunikacja procesu z warstwą pośrednią odbywa się w sposób asynchroniczny, poprzez kolejne wywołania aktywności request (wysłanie żądania), by następnie przejść do aktywności recieve (oczekiwać na jego rezultat). W między czasie proces może również wykonać inne operacje. Aby skojarzyć aktywność wysłania żądania z otrzymaniem żądania w przypadku wielu instancji procesu konieczna jest identyfikacja tych żądań za pomocą tego samego unikalnego klucza. W procesach biznesowych BPEL mechanizm ten został nazwanych mechanizmem korelacji. Aby skorzystać zatem z mechanizmy korelacji, podczas tworzenia nowego zadania proces biznesowy musi otrzymać unikalny klucz tego zadania by następnie podczas otrzymania rezultatu zadania był w stanie skojarzyć ten rezultat z konkretną instancją procesu. Rezultat w takim wypadku również musi zostać podpisany unikalnym identyfikatorem zadania.

Rysunek 6.8 przedstawia diagram sekwencji pobierania listy zadań przez aplikację mobilną. Sekwencja w tym przypadku zaczyna się od klasy MobileRequestController, która jest odpowiedzialna za obsługę żądań odpowiednich żądań http. Kontroler ten rozpoczyna wykonanie metody od autentykacji użytkownika którym przedstawia się żądanie http (z wykorzystaniem http basic authentication). Po poprawnej autentykacji zostają pobrane informacje o aktualnym użytkowniku, informacje te zostaną wykorzystane do wygenerowania listy zadań dla niego dostępnych. Sterowanie zostaje następnie przekazane do klasy MobileRequestDispatcher, która za pomocą QueryFactory zbuduje dwa zapytania, jedno do wyciągnięcia nowych zadań do których użytkownika ma dostęp. Drugie zapytanie pokryje wszystkie zadania w trakcie realizacji przypisane do użytkownika. Zapytania te posłużą następnie do wyciągnięcia

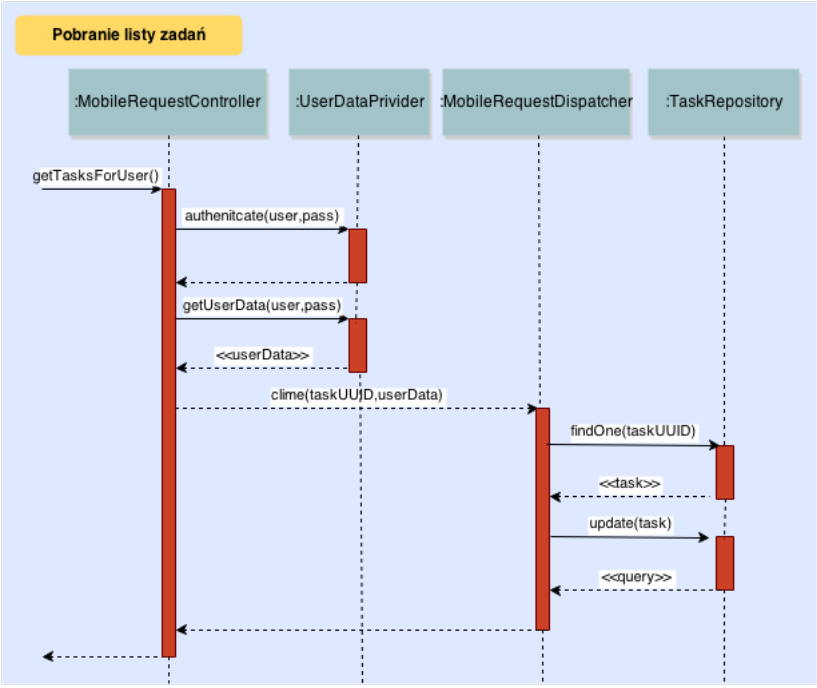



Rysunek 6.8: Diagram sekwencji pobrania listy zadań dostępnych dla użytkownika.

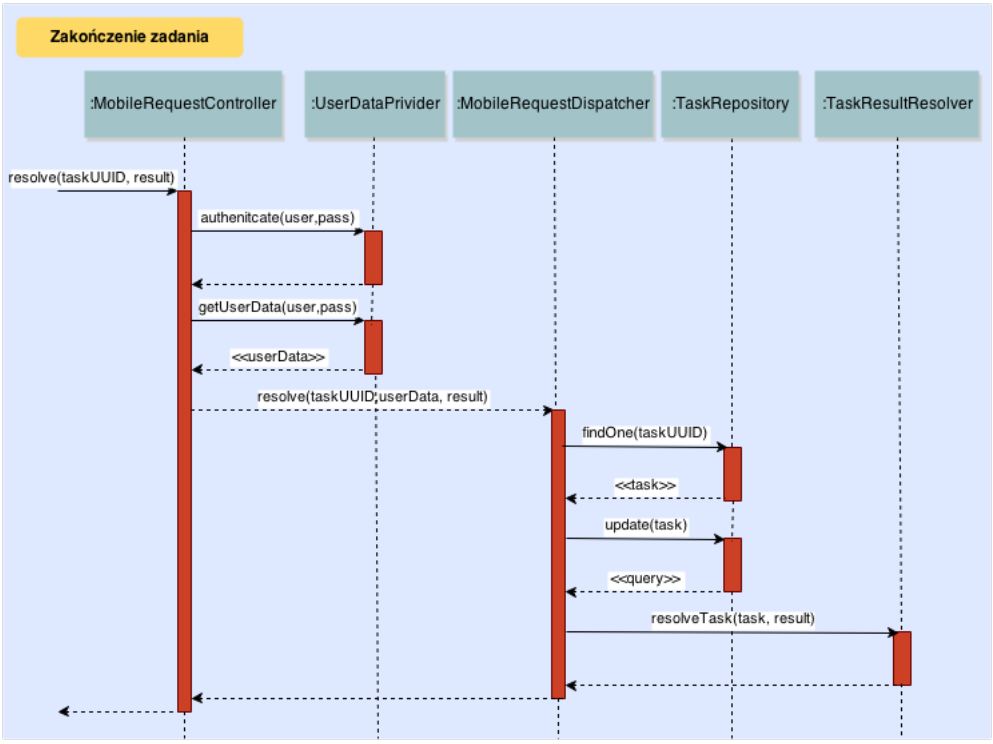
listy zadań z bazy danych przy pomocy repozytorium TaskRepository. Ostatnim krokiem będzie serializacja wydobytych zadań do postaci dokumentu JSON.

Rysunek 6.9 przedstawia diagram przypisania konkretnego zadania do użytkownika. Przypisanie odbywa się za pomocą REST API. Odpowiednie żądanie HTTP zostaje przyjęte przez MobileRequestController, ponownie jak w przypadku pobierania listy zadań, dwie pierwsze operacje wykonywane przez kontroler to autentykacja i pobranie informacji o użytkowniku. Sterowanie zostaje następnie przekazane do MobileRquestDispatcher'a, który znajduje odpowiednie zadanie w repozytorium, zmienia jego status uzupełniając informacje o przypisanym użytkowniku by następnie zapisać te zmiany. Operacją kompensacyjną do przypisania jest operacja zwolnienia zadania. Diagram sekwencji w przypadku tej drugiej wygląda niemal identycznie jak diagram przypisania z tym że MobileRequrestDispatcher przywraca zadaniu status do realizacji.

Na rysunku 6.10 przedstawiono ostatni rodzaj funkcjonalności jakim jest obsługa rozwiązania zadania. Decyzja o rozwiązaniu również pochodzi od strony aplikacji mobilnej i a funkcjonalność ją obsługująca udostępniona jest za pomocą REST API. Tak jak w poprzednich przypadkach przed rozpoczęciem przetwarzania żądania następuje autentykacja oraz pobranie informacji o użytkowniku. Następnie pobierane zostaje zadanie z repozytorium, ustawiany rezultat oraz zmieniany status. Rezultat następnie zostaje odesłany do procesu biznesowego przez klasę TaskResultResolver. Jeśli odesłanie rezultatu zakończy się sukcesem zadanie z odpowiednio ustawionym statusem i rezultatem zostaje zapisane do bazy danych.




Rysunek 6.9: Diagram sekwencji przypisania użytkownika do zadania.




Rysunek 6.10: Diagram sekwencji rozwiązania zadania.

# Bibliografia


[1] European Association of Business Process Management. *Organization home website*. `http://www.eabpm.org/`.

[2] Antipodes. *What is a Business Process Management system?*. `http://www.antipodes.bg/en/cubes/what_is_bpm/`.

[3] OASIS. *OASIS Web Services Business Process Execution Language (WSBPEL) TC* `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`

[4] OASIS *Web Services Business Process Execution Language Version 2.0*. OASIS Standard 11 April 2007

[5] Mulesoft. *What is application orchestration?* `http://www.mulesoft.org/what-application-orchestration`

[6] IBM and SAP *WS-BPEL Extension for People – BPEL4People* A Joint White Paper by IBM and SAP July 2005