Politechnika Warszawska
Wydział, Elektroniki i Technik Informacyjnych
Instytut Informatyki

Rok akademicki 2008/2009

Praca dyplomowa magisterska

Wojciech Klicki
Konrad Starzyk

Integracja technologii mobilnych i systemów klasy Enterprise

Opiekun pracy:
mgr inż. Piotr Salata

Ocena .....

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Specjalność: Informatyka –
Inżynieria oprogramowania
i systemy informacyjne

Data urodzenia: 1 stycznia 1980 r.

Data rozpoczęcia studiów: 1 października 2004 r.

Życiorys

Nazywam się ....

.....
podpis studenta

Egzamin dyplomowy

Złożył egzamin dyplomowy w dn. .....

Z wynikiem .....

Ogólny wynik studiów .....

Dodatkowe wnioski i uwagi Komisji .....

.....

Specjalność: Informatyka –
Inżynieria oprogramowania
i systemy informacyjne

Data urodzenia: 1 stycznia 1980 r.

Data rozpoczęcia studiów: 1 października 2004 r.

Życiorys

Nazywam się ....

.....
podpis studenta

Egzamin dyplomowy

Złożył egzamin dyplomowy w dn. .....

Z wynikiem .....

Ogólny wynik studiów .....

Dodatkowe wnioski i uwagi Komisji .....

.....

Streszczenie

Praca ta prezentuje . . .

Słowa kluczowe: *słowa kluczowe.*

Abstract

Title: *Thesis title.*

This thesis describes . . .

Key words: *key words.*

Contents

1. Integracja w środowiskach Enterprise

Oprogramowanie Enterprise jest pojęciem dość szerokim, opisującym systemy przeznaczone dla przedsiębiorstw, których działanie odwzorowuje zachodzące procesy biznesowe. Niekiedy pojęcie to odnosi się do oprogramowania pisanego na zamówienie lub też do rozbudowanych pakietów wspierających określone czynności, np. kontakty z klientami lub księgowość. Rzadko się jednak zdarza, by istniał jeden system który potrafiłby spełnić wymagania klienta, a nawet jeżeli, z różnych przyczyn firma może nie chcieć go wdrożyć. Tak więc nawet w jednym przedsiębiorstwie często można się spotkać z sytuacją w której działa wiele niezależnych aplikacji, nierzadko pisanych przez różne firmy, które muszą się ze sobą komunikować w celu wymiany danych.

Jeszcze do nie dawna mówiąc o integracji mieliśmy na myśli właściwie wyłącznie serwery aplikacyjne. Obecnie obowiązującym trendem jest postępująca miniaturyzacja i wzrost mocy obliczeniowej urządzeń mobilnych, które udostępniają klientom usługi dostępne do tej pory wyłącznie za pośrednictwem komputera stacjonarnego. Tworząc aplikację na urządzenia przenośne, która będzie współpracowała z istniejącymi już systemami, napotykamy jednak na nowe problemy, wynikające ze specyfiki środowiska mobilnego.

1.1. Potrzeba integracji

Potrzeba integracji pojawia się, gdy pojawia się różnorodność, która wymaga dodatkowego nakładu pracy sprowadzającego ją do wspólnego poziomu. Warto zauważyć, że istnieją różne poziomy różnorodności, a rozwiązania integracyjne mogą albo usuwać tę różnorodność na poziomie technicznym albo odpowiadać na potrzeby wynikające z różnorodności na wyższym poziomie, na przykład organizacyjnym. Możemy więc wyróżnić następujące poziomy różnorodności:

— Dyscyplinarna

  - oddzielne działy wewnątrz firmy lub różne firmy

— Geograficzna

  - w ramach jednego kraju

  - w ramach wielu krajów/stref czasowych

— Zawartości

  - rozdzielenie działów operacyjnych, np. obsługa klienta, logistyka, księgowość

  - podział pracy pomiędzy działy, np. procesu który jest obsługiwany przez każdy z nich

— Infrastrukturalna

  - użycie różnych narzędzi i architektur

  - użycie różnych standardów i procesów

W niniejszej pracy nie będziemy rozważać przyczyn różnorodności, przyjmujemy ją raczej jako problem, który należy rozwiązać. Jednak samo zwrócenie uwagi na listę możliwych przyczyn, sprawia, że możemy sobie wyobrazić, iż systemy jakie spotykamy w przedsiębiorstwach dużej skali często składają się z setek, jeśli nie tysięcy aplikacji wykonanych na zamówienie przez zewnętrzne firmy lub przez wewnętrzne działy informatyczne. Część z tych aplikacji jest tak zwanymi 'legacy systems' napisanymi w zapomnianych już językach, przygotowanymi pod platformy, które często nie mogą liczyć już na

wsparcie producentów. Wielokrotnie można natrafić na kombinacje tego typu aplikacji działające w różnych warstwach w obrębie różnych systemów operacyjnych.

1.1.1. Pełnowartościowe środowiska informatyczne

Przyczyn leżących u podstaw tak dużego skomplikowania sytuacji w środowiskach informatycznych typu Enterprise należy doszukiwać się w skomplikowaniu zagadnienia jakim jest projektowanie dużych aplikacji. Zbudowaniem jednej, posiadającej wszystkie niezbędne funkcje aplikacji, która byłaby w stanie w pełni zaspokoić potrzeby dużego przedsiębiorstwa. Firmy działające na rynku ERP (Enterprise Resource Planning) od lat nie ustają w wysiłkach by zbudować tego typu aplikację. Jednak nawet największe z nich wytworzyły produkty, które pokrywają tylko część niezbędnych funkcji. Można to zaobserwować na przykładzie systemów, które stanowią punkty integracyjne we współczesnych rozwiązaniach. Kolejnym powodem jest to, że rozwiązania, które składają się z wielu małych zintegrowanych podsystemów pozwalają na pewną elastyczność w doborze najlepszych rozwiązań w zakresie danej dziedziny. Mogą wybrać takie rozwiązanie, które jest w danej chwili najlepsze i stosunkowo najtańsze na rynku. Nie są zmuszeni do inwestowania w całościowe rozwiązania dostarczane przez tylko jednego dostawcę. Co więcej opieranie się na wielu podsystemach wykonywanych przez różne firmy może pozwolić na pewne zrównoleglenie prac nad wdrożeniem odrębnych części systemu. W przypadku jednego dużego systemu potencjalnie prace nad nim mogłyby być blokowane przez braki w zasobach przedsiębiorstwa realizującego zlecenie wdrożenia.

Niestety takie podejście przynosi najlepsze efekty przy ścisłym podziale funkcjonalności pomiędzy produktami różnych producentów. W rzeczywistości pokusa budowania systemów, które zabiorą elementarną funkcjonalność innej części z zupełnie innego pola, jest zbyt duża

i prowadzi do powstawania rozwiązań, które nie mają wyraźnego podziału na dziedziny.

Przy rozwiązaniach tego typu należy zwrócić uwagę, że z punktu widzenia użytkownika system enterprise, pomimo że w rzeczywistości może składać się z wielu małych, zintegrowanych ze sobą systemów, stanowi jedną całość. Na przykład użytkownik może wysłać zapytanie o stan konta oraz adres zamieszkania klienta. W sytuacji gdy mamy do czynienia z systemem korporacyjnym takie zapytanie może wymagać odwołania się do dwóch zupełnie różnych podsystemów (bilingowego oraz kontaktu z klientem). System często musi dokonać autentykacji oraz autoryzacji użytkownika, sprawdzić obciążenie serwerów w celu wybrania optymalnego do realizacji zlecenia. Tego typu proces może w bardzo prosty sposób zaangażować kilka systemów. Z punktu widzenia użytkownika jest to pojedyncza transakcja.

W celu zapewnienia poprawności działania systemów, umożliwienia wydajnej wymiany danych oraz zapewnienia przezroczystego funkcjonowania procesów biznesowych, opisane powyżej aplikacje muszą zostać zintegrowane. Wraz z integracją pojawiają się inne potrzeby takie jak zapewnienie bezpiecznej wymiany danych czy też umożliwienie dostępu do systemu z różnych platform zewnętrznych.

1.1.2. Systemy mobilne

Jedną z podstawowych cech jakie musi posiadać system enterprise jest łatwość dostępu do przechowywanym w nim informacji. Typowy użytkownik takiego systemu chce mieć możliwość połączenia się z nim i pobrania konkretnych informacji o każdej porze dnia czy nocy. Wraz z pojawieniem się nowej generacji urządzeń mobilnych możliwym stało się zrealizowanie tej potrzeby. Nowoczesne platformy oferują niespotykaną wcześniej moc oraz zasięg, które pozwalają na stały dostęp do sieci firmowej z dowolnego niemal miejsca na ziemi. Usługi synchronizujące pocztę korporacyjną, oferującą pojedynczą skrzynkę poczty

przychodzącej, są obecnie niezbędnym minimum w każdej dużej firmie. Następnym etapem w rozwoju jest umożliwianie szybkiego i interaktywnego dostępu do działających w obrębie intranetu usług wewnętrznych. Ten nowy rodzaj potrzeby integracji - integracja mobilna, stanowi zupełnie nowy rodzaj wyzwania dla firm oferujących narzędzia dla biznesu. Niesie ona ze sobą zupełnie nowe zagadnienia oraz problemy, które nie były spotykane przy klasycznej integracji. Wymusza to wypracowanie zupełnie nowej metodologii przemysłowego wytwarzania oprogramowania zapewniającego zaspokojenie tej potrzeby.

1.2. Klasyczne wyzwania integracji

Integracja aplikacji w przedsiębiorstwach rzadko jest prostym zadaniem. Mimo, że w niemal każdej dużej firmie zachodzi potrzeba integracji i powstało już wiele opracowań na ten temat, nikt nie przedstawi kompletnej listy problemów, które należałoby rozważyć podczas integrowania systemów. Dodatkowo, wyzwania które stawia integracja wykraczają poza tylko techniczne rozważania, a rozciągają się także na procedury biznesowe. Tak jak już wspomnieliśmy, przedstawienie kompletnej listy takich problemów nie jest możliwe, ze względu na konieczność dokonania analizy każdego z przypadków. Mimo to, spróbujemy przedstawić te najbardziej typowe, zauważone i opisane jako te, które koniecznie trzeba wziąć pod uwagę.

1.2.1. Wyzwania techniczne

Fizyczna i logiczna odrębność systemów, z której tak naprawdę wynika potrzeba integracji. Wprowadza to jednak konieczne do rozważenia kwestie, takie jak:

— wybór sposobu komunikacji między systemami (o których więcej w rozdziale 1.4)

— wpływ tego wyboru na bezpieczeństwo i zgodność takiego rozwiązania z polityką bezpieczeństwa firmy

Ograniczony dostęp do integrowanych aplikacji. Często okazuje się, że integrowane aplikacje to istniejące od dawna systemy, które nie dość, że nie dają możliwości zmian w źródłach, to dodatkowo udostępniają mocno ograniczony interfejs zewnętrzny. W takiej sytuacji programiści muszą niekiedy uciekać się do stosowania niskopoziomowych modyfikacji (np. na bazie danych, której używa aplikacja) czy w ostateczności do dekompilacji jej modułów.

Brak uniwersalnego standardu wymiany danych pomiędzy aplikacjami. Pomimo często istniejącej potrzeby integracji, wiele aplikacji nie wspiera żadnego uniwersalnego formatu danych, co powoduje konieczność specjalizowania platform integracyjnych pod kątem podłączanych do niej aplikacji. Jako rozwiązanie tego problemu często stosuje się XML i Web serwisy, jednak ich obsługi możemy oczekiwać jedynie w przypadku stosunkowo nowych aplikacji. Jednak nawet te technologie nie dają gwarancji pełnej uniwersalności ze względu na dodatkowe rozszerzenia, które się wykształciły w ramach Web serwisów (o których więcej informacji w dalszej części pracy). Warto zauważyć, że ten sam problem - brak współpracy pomiędzy aplikacjami oficjalnie wspierającymi pewien standard - był główną przeszkodą w stosowaniu CORBY-y, zaawansowanego protokołu umożliwiającego współpracę aplikacji stworzonych w różnych językach i działających na różnych platformach.

Utrzymanie platformy integracyjnej. O ile uruchomienie platformy integracyjnej - systemu komunikującego ze sobą wiele różnorodnych aplikacji jest trudnym zadaniem, o tyle utrzymanie go może być jeszcze trudniejsze. Rzadko się zdarza, by była ona całkowicie scentralizowana

i nie wymagała najmniejszej ingerencji, lub chociaż konfiguracji integrowanych aplikacji. W związku z tym, często osoby, które odpowiadają za utrzymanie takiej platformy muszą posiadać wiedzę na ich temat. Dodatkowo, ciężko jest tę wiedzę utrzymać, szczególnie gdy pracownicy się zmieniają. Natura platformy integracyjnej wymusza jej automatyzację - powoduje to,że może ona bezproblemowo działać bardzo długo, wręcz przezroczyście dla wszystkich, aż do wystąpienia pierwszego problemu, lub potrzeby wprowadzenia modyfikacji. Wtedy okazuje się, że osobami od których wymaga się tej wiedzy są właśnie ci, którzy tę platformę tworzyli, co z ich punktu widzenia zdecydowanie zwiększa nakład pracy (którego by nie ponieśli, gdyby systemy były integrowane bez punktu centralnego, ponieważ wtedy osoby odpowiadające za nie byłyby również odpowiedzialne za ich integrację). Ostatecznie okazuje się, że istnienie platformy integracyjnej jest utrudnieniem, a nie ułatwieniem dla działu IT.

Różnorodność źródeł danych. Niekiedy pojawia się potrzeba integrowania danych pochodzących z różnych, czasami bardzo nietypowych źródeł. Przykładem jest wypowiedź pewnego biznesmena, który przedstawiał sytuację, gdy pojawiło się zarządzenie wymagające gromadzenia wszystkich informacji związanych z rozwojem projektu w jednym miejscu. Naturalnym sposobem komunikacji w firmach jest poczta elektroniczna oraz spotkania na których powstają notatki. W tym momencie, przeniesienie wymienianych tą drogą ustaleń, było czasochłonnym zadaniem, którego efekt był ciężki w ocenie. Może się wręcz okazać, że jest to wymaganie niemożliwe do spełnienia, jeżeli format komunikacji nie zostanie wcześniej ustalony.

1.2.2. Wyzwania biznesowe

Zmiana polityki wewnętrznej przedsiębiorstwa. Poprawna integracja oznacza nie tylko doprowadzenie do współpracy systemów informatycznych, ale również departamentów w firmie. Wynika to z faktu, że integrowane aplikacje najczęściej komunikują automatycznie, co powoduje, że wprowadzenie zmian do systemu przez jeden departament, może mieć konsekwencje (np. w postaci realizacji zlecenia) w drugim. Konieczne jest uświadomienie użytkowników o tym fakcie, tak by integracja przynosiła korzyści w postaci uproszczenia przepływu danych i zaoszczędzenia czasu, zamiast prób ręcznego kontrolowania niezależnych do tej pory aplikacji.

Wpływ integracji na przedsiębiorstwo. Rozwiązania integracyjne łączą ze sobą wiele aplikacji, których działanie jest kluczowe dla działania przedsiębiorstwa. Powoduje to, że błędy w działaniu lub niedostępność platformy integracyjnej mają bardzo poważne konsekwencje, także finansowe. Z drugiej strony, poprawnie funkcjonująca integracja przyspiesza przepływ informacji i powoduje, że każdy scenariusz współpracy przebiega w dających się przewidzieć krokach, co gwarantuje powtarzalność i wiedzę o stanie współpracy (np. stanie zlecenia, które wpłynęło i jest obsługiwane). Należy pamiętać, że utrzymywanie platformy integracyjnej wymaga kontaktu pomiędzy osobami opiekującymi się integrowanymi systemami, a działem utrzymującym platformę. Nie można dopuścić do niekonsultowanych wcześniej zmian w którymkolwiek z systemów, bo inaczej może to doprowadzić do awarii wszystkich integrowanych systemów.

Różnice semantyczne w pojęciach. Choć XML rozwiązuje wiele problemów technicznych, nie stanowi on odpowiedzi na różną semantykę w ramach obu systemów. Przykładowo, tym co w jednym systemie rozumiemy pod pojęciem "konto" może oznaczać numer konta bankowego,

w innym natomiast może to być wewnętrzny numer konta w korporacji, czy też konto użytkownika. Dlatego poza znalezieniem wspólnych pojęć, konieczne jest dokładne zdefiniowanie ich znaczenia, aby uniknąć nieporozumień.

1.3. Problemy integracji mobilnej

Integracja mobilna niesie za sobą wyzwania podobne do przedstawionych w poprzednim rozdziale. Niektóre z nich nie były jednak, aż tak dużym problemem, jakim stają się w kontekście mobilnym. Inne wynikają z natury urządzeń mobilnych i komunikacji między nimi. Wszystkie te problemy należy wziąć pod uwagę podczas tworzenia rozwiązania integracyjnego.

1.3.1. Różnorodność platform

Pierwszym problemem, który możemy zauważyć, jest różnorodność mobilnych platform. Istnieje wiele niekompatybilnych ze sobą rozwiązań pochodzących od różnych producentów. Jeżeli spojrzymy na platformy, na których budowane są klasyczne systemy korporacyjne, możemy ograniczyć je do Javy Suna i .NET Microsoftu. Po stronie mobilnej natomiast, mamy do wyboru wersje mobilne wymienionych platform oraz dodatkowo , Symbiana, Android, Palm OS oraz kilku mniejszych dostawców. Oprócz tego Apple promuje swojego iPhone, który mimo, że jest platformą bardzo zamkniętą, cieszy się wielkim zainteresowaniem wśród zwykłych użytkowników.
Dodatkowym problemem jest to, że nie istnieje tak naprawdę standard tworzenia korporacyjnych aplikacji dla urządzeń mobilnych, który dostarczałby narzędzi ułatwiających często powtarzane czynności, podobnie jak ma to miejsce w przypadku Javy Enterprise. Tworząc lub integrując istniejącą aplikację dla urządzenia mobilnego musimy więc wziąć pod uwagę platformę na której działa, a i tak nie

obędzie się bez użycia dodatkowych narzędzi rozwiązujących dawno już rozwiązane w klasycznych systemach problemy.
W przypadku urządzeń mobilnych nie możemy też skorzystać ze sprawdzonej trójwarstwowej architektury, używając palmtopa lub telefonu komórkowego, tak jak cienkiego klienta. Jest to spowodowane niezgodnością przeglądarek mobilnych i brakiem pełnej obsługi JavaScript, na którym opartych jest wiele stron internetowych. Istnieją oczywiście wersje stron przygotowane dla platform mobilnych, ale daleko im do interaktywności oferowanej przez bogate i przypominające biurkowe aplikacje strony internetowe.

1.3.2. Wysokie koszty

Koszty tworzenia aplikacji mobilnych (i wynikające stąd koszty integracji) różnią się w przypadku każdej z platform, jednak tym co podnosi koszt w przypadku każdej z nich jest:

— Mniejsza społeczność programistów tworzących aplikacje dla danej platformy. Oznacza to, że szeroko stosowane postępowanie, polegające na poszukiwaniu podobnych problemów, z którymi starł się ktoś inny, nie zawsze przynosi równie dobre rezultaty. Zwiększa to istotnie czas potrzebny na rozwiązanie takiego problemu.

1.3.3. Kanały komunikacyjne

1.3.4. Sposoby interakcji

1.3.5. Systemy off-line

1.4. Istniejące podejścia do integracji

Istnieje kilka powszechnie znanych sposobów integracji, które mają swoje mocne i słabe strony. Można zauważyć, że pojawiały się one wraz z rozwojem systemów informatycznych i każdy z nich odpowiada

jakiemuś etapowi rozwoju (np. istnienie plików jako podstawowych struktur przechowujących dane, na długo przed pojawieniem się baz danych). Mimo to, każdy z nich ciągle znajduje swoje zastosowanie, gdy okazuje się, że nie wszystkie mechanizmy są jednakowo dostępne lub gdy bardziej skomplikowane technologie wprowadzają zbyt duży narzut.

Transfer plików - jedna aplikacja zapisuje plik w określonym formacie w ustalonym miejscu, następnie druga go odczytuje i przetwarza. Zaletą takiego rozwiązania jest jego dostępność - prawie zawsze będziemy mieli dostęp do systemu plików. Poza tym nie musimy wiedzieć, jak działa aplikacja, jedyne co powinniśmy zrobić to dostarczyć plik we właściwym formacie. Z drugiej strony, nie istnieje żaden sposób wymuszenia formatu zapisywanego pliku, możemy zapisać na dysku dowolny plik, a aplikacja docelowa będzie go mogła ewentualnie odrzucić.

Wspólna baza danych - dwie aplikacje działają na jednej bazie danych. Ponieważ obie korzystają z tych samych danych, nie ma potrzeby ich duplikacji. Dodatkowo, przez ograniczenia wymuszone przez bazę musimy je zapisywać w ściśle określonym formacie, przestrzegając typów i więzów integralności. Z drugiej strony, trudno jest taką bazę zaprojektować. Gdybyśmy nawet stworzyli dwie takie aplikacje, które współpracują z jedną bazą danych, to w wypadku gdybyśmy chcieli dołączyć kolejną, współpracującą aplikację, mogłoby się to wiązać z koniecznością przemodelowania bazy danych. Dodatkowo, zmiana danych w bazie nie zawsze wystarczy. Możemy sobie wyobrazić sytuację, w której zmiana wysokości pensji niesie za sobą konieczność wykonania dodatkowych czynności, np. zmiany wysokości składki w systemie ubezpieczeniowym.

Zdalne wywoływanie procedur - może się odbywać z wykorzystaniem mechanizmów typowych dla danej technologii (RPC,RMI czy WebServices). Jedna aplikacja udostępnia funkcjonalność, która może zostać wywołana z poziomu drugiej. Komunikacja odbywa się synchronicznie, aplikacja wywołująca oczekuje na wynik przetwarzania. Zaletą takiego rozwiązania jest dobra enkapsulacja wywołania - nie istnieje możliwość wywołania nieistniejącej funkcjonalności z przekazaniem parametrów nie spełniających określonych założeń. Wywołanie zdalnej procedury po odpowiednim skonfigurowaniu całego systemu, nie powinno się różnić od wywołania procedury w obrębie tej samej aplikacji. Pomimo wygody takiego rozwiązania, stwarza ono pewne niebezpieczeństwo. Programista, nieświadomy narzutu stwarzanego przez zdalne wywołanie procedur, może wywoływać je z równą bezstroską niczym procedury lokalne.

Wiadomości - jedna aplikacja wysyła komunikat do wspólnego kanału komunikacyjnego, który jest następnie przetwarzany przez drugą aplikację. Komunikacja odbywa się asynchronicznie, po umieszczeniu komunikatu w kanale, aplikacja kontynuuje działanie. Odbiorca komunikatu może go odebrać w dogodnym dla siebie momencie. Rozwiązanie to nie wprowadza ścisłych zależności pomiędzy systemami, podobnie jak przesyłanie plików. Pewną niedogodnością takiego rozwiązania jest jednak większe skomplikowanie systemu, wynikające z wprowadzenie dodatkowego elementu - brokera wiadomości.

1.4.1. Wybór technologii do integracji mobilnej

W kontekście przedstawionych informacji musimy dokonać wyboru metody komunikacji urządzenia mobilnego z klasycznym systemem informatycznym. W tym celu należy rozważyć następujące zagadnienia:

— Jakie mechanizmy są dostępne w obu środowiskach?

— Jakie są oczekiwania użytkowników co do zintegrowanych systemów i jak wybór metody integracji wpływa na ich spełnienie?

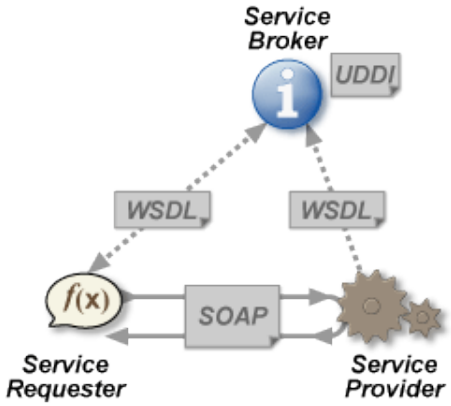
Rozważając te kwestie możemy stwierdzić, że użytkownik, szczególnie gdy jest przyzwyczajony do pracy w klasycznym środowisku informatycznym, oczekuje, że komunikacja będzie się odbywała jak najszybciej, bez uciążliwych okresów oczekiwania. Dodatkowo, oczywistymi wymaganiami są bezpieczeństwo i niezawodność komunikacji. Z drugiej strony musimy pamiętać, że warunki w jakich możemy komunikować się z urządzeniem przenośnym są dalekie od doskonałych. To sprawia, że należy ją tak zaprojektować, aby przerwy w komunikacji były jak najmniej odczuwalne. Systemy komunikacyjne takie jak RPC, RMI czy CORBA projektowane były dla warunków w których problemy komunikacyjne są sytuacją nadzwyczajną. Tymczasem w przypadku łączności bezprzewodowej utrata połączenia jest czymś czego możemy się spodziewać.

Zacznijmy od ograniczenia wyboru do technologii, z których możemy skorzystać w środowisku mobilnym. Na samym początku musimy wyeliminować transfer plików, jako metodę, która wymaga spełnienia jednego z warunków: dostęp do wspólnego systemu plików lub obsługa prostego protokołu przesyłania plików (np. FTP). Pierwsza możliwość jest trudna do zagwarantowania ze względu na brak wsparcia powszechnie stosowanych systemów plików we wszystkich urządzeniach mobilnych (chociaż są wyjątki, np. FAT), a także możliwe przerwy w komunikacji, o których wspominaliśmy wcześniej. Drugie rozwiązanie - obsługa uniwersalnego protokołu przesyłania plików nie jest powszechnie wspierane, poza tym także tu pojawia się problem zawodnej komunikacji.

Wspólna baza danych również jest rozwiązaniem, którego nie możemy wziąć pod uwagę. Główną przeszkodą jest brak uniwersalnego sposobu podłączenia do bazy działającej w klasycznym systemie.

Duże ograniczenia platformy mobilnej nie pozwalają na uruchomienie sterowników do baz danych przeznaczonych dla klasycznych systemów, dodatkowo ich rozmiar nierzadko przekracza rozmiar mobilnych aplikacji. Gdyby jednak nawet udało się stworzyć sterowniki, które pozwoliłyby na dostęp do bazy danych z poziomu urządzenia mobilnego, ponownie starlibyśmy się z problemem zawodnej komunikacji. Okazuje się, że dwoma godnymi rozważenia metodami są wiadomości oraz Web serwisy, będące jednym ze sposobów zdalnego wywoływania procedur.

Zdalne wywoływanie procedur - Web serwisy Pomimo, iż istnieje kilka różnych standardów RPC, Web serwisy są jedynym praktycznie stosowanym w przypadku integracji mobilnej. Jest to bezpośrednio związane z dużą niezależnością od platformy jaką gwarantuje zastosowanie protokołu SOAP oraz popularnością i powszechną akceptacją jako standardu komunikacji odrębnych systemów. Ich wyjątkowa siła leży w zastosowaniu kilku powszechny technologii internetowych, takich jak protokół HTTP czy język XML. To gwarantuje, że nie ma znaczenia w jakiej technologii i w jakim systemie pracuje dana aplikacja komunikująca się za pomocą Web Services. Trzy fundamentalne elementy wchodzące w skład tej techlogii można zobaczyć na poniższym rysunku.



Wspomniany wcześniej SOAP, czyli Simple Object Access Protocol jest formą elektronicznej koperty, która zawiera podstawowe informacje o rządaniu oraz o odpowiedzi. Jest zbudowana w oparciu o język XML. Drugą kluczową technologią jest język opisu usługi - Web Service Description Language (WSDL). Za jego pomocą jesteśmy w stanie poinformować jakie usługi oferuje nasz serwer oraz możemy wskazać dokładny sposób ich wywoływania. W wypadku urządzeń o ograniczonej mocy obliczeniowej spotyka się go tylko przy automatycznym generowaniu kodu wywołującego Web Services. Ostatnim fundamentem tej technologii jest usługa UDDI (Universal Description, Discovery, and Integration). Pozwala ona na dynamiczne odnajdowanie innych usług Web.

Największą wadą tego rozwiązania jest pewien narzut wydajnościowy oraz pojemnościowy związany z budową samych wiadomości. W innych metodach treść wiadomości przesyłana jest w zoptymalizownej postaci kodu binarnego i nie posiada zwykle żadnych metadanych(co zwykle prowadzi do problemów z komunikacją pomiędzy różnymi platformami). W przypadku Web Services wiadomości przesyłane są w kopercie zbudowanej na bazie XML, która posiada dużo nadmiernych informacji. Powoduje to, że przy przesyłaniu wiadomści łącze dodatkowo zostaje obciążone tymi nadmiarowymi danymi. Co więcej, po stronie odbiorcy należy przeprowadzić rozbiór gramatyczny otrzymanego komunikatu. Zwiększa to zapotrzebowanie na moc obliczeniową. Może to mieć kluczowe znaczenie w przypadku urządzeń mobilnych, które często dysponują bardzo ograniczonymi zasobami mocy obliczeniowej oraz przepustowości. Szczególnym przypadkiem są użytkownicy, którzy posiadają niezryczałtowany dostęp do sieci i płacą różne stawki w zależności od ilości przesłanych danych. W tym wypadku narzut pojemnościowy Web Services można w sposób bezpośredni przeliczyć na koszty jakie on generuje dla całej organizacji.

Z myślą o bezpiecznej komunikacji przy użyciu technologii Web Service wprowadzono protokół komunikacyjny WS-Security (Web Service Security), pozwalający na zaaplikowanie podstawowych standardów bezpieczeństwa. Protokół ten zawiera specyfikację opisującą metody wymuszania integralności oraz poufności przesyłanych danych. Pozwala na dołączanie do wiadomości SOAP podpisów oraz nagłówków związanych z szyfrowaniem danych. Wspiera taże formaty certyfikacyjne takie jak X.509. Protokół działa w warstwie aplikacyjnej i został tak zaprojektowany by zapewnić bezpieczeństwo od końca do końca (end-to-end security). Niestety w obecnej chwili protokół ten nie został zaimplementowany na urządzeniach mobilnych i na pewno w najbliższym czasie nie stanie się na nich standardem. Ograniczeniem tu są typowe dla tych urządzeń problemy z wydajnością. Alternatywnym rozwiązaniem problemu bezpieczeństwa jest posłużenie się zabezpieczeniami na poziomie warstwy transportowej (TLS) takimi jak https. Pozwala to na zapewnienie bezpieczeństwa punkt-punkt.
Technologia Web Services jest godna rozważenia przy wyborze sposobu komunikacji w integracji mobilnej. Istnieją bardzo dobre implementacje takie jak kSoap, które działają praktycznie na każdej platformie wspierającej wirtualną maszyne Java. Pozostałe platformy taki jak Blackberry czy WMD (Windows Mobile Device) również posiadają stosowne biblioteki, które pozwalają w bardzo prosty sposób podłączyć się do istniejącej infrastuktury. Znane są komercyjne rozwiązania integracyjne takie jak Mobile Data Service na platformie Blackberry, które potrafią automatycznie generować kod wywołujący procedury na podstawie podanego im wcześniej schematu WSDL. Jeśli kluczowym nie jest optymalne wykorzystanie łącza czy też gwarancja dostaczenia, to jest to technologia, której użycie jest zalecane przy integracji mobilnej, czy też jakiejkolwiek innej integracji w środowisku Enterprise.

Wiadomości w środowisku mobilnym Integracja za pomocą Web serwisów jest dość prosta do zaimplementowania, stawia jednak przed programistą szereg problemów, które musi samodzielnie rozwiązać. Jest to przede wszystkim brak gwarancji dostarczenia komunikatu. Dodatkowo jednak samodzielnie należy obsłużyć błędy transmisji, możliwe przerwy i konieczność ponownego nawiązania połączenia. Z drugiej jednak strony nie ma potrzeby uruchamiania brokera wiadomości, który jest dodatkowym elementem zwiększającym skomplikowanie systemu. Jednak w sytuacji, gdy w przedsiębiorstwie działa już broker obsługujący kilka aplikacji, dołączenie komponentu mobilnego może być dobrym rozwiązaniem. Zastosowanie wiadomości do integracji urządzeń mobilnych niesie ze sobą wszelkie konsekwencje tej metody integracji znane z integracji klasycznych systemów informatycznych:

— Oddzielenie nadawcy wiadomości od jej odbiorcy. Pozwala to na wysyłanie wiadomości nawet wtedy, gdy odbiorca nie jest w stanie jej aktualnie odebrać (na przykład ze względu na brak połączenia).
— Gwarancja dostarczenia. Umieszczenie wiadomości w kanale komunikacyjnym gwarantuje, że odbiorca odbierze ją wtedy gdy będzie pobierał przeznaczone dla niego wiadomości. Z drugiej strony, w przypadku niepowodzenia, jest o tym fakcie od razu informowany.
— Skalowalność. Wiadomości pozwalają na lepsze wykorzystanie zasobów serwerów. Aplikacje będące odbiorcami wiadomości mogą je odbierać w tempie, w którym są w stanie je przetwarzać. Wyklucza to więc możliwość ich przeciążenia (tutaj oczywistym wymaganiem jest odpowiednio duża pojemność kolejki wiadomości).

— Nadawanie priorytetów. Dzięki wiadomościom, możemy przetwarzać żądania zgodnie z ich priorytetem, szczególnie w momentach wysokiego obciążenia. Pozwala to na obsłużenie najpilniejszych wiadomości przed innymi (co nie jest możliwe bez dodatkowych ingerencji w przypadku Web serwisów).

— Możliwość stosowania wzorców integracyjnych. Droga wiadomości pomiędzy nadawcą i odbiorcą może być zmieniana w deklaratywny sposób (pomagają tu specjalne pakiety integracyjne, np. Apache Camel), co pozwala na dodatkowe przetwarzanie, konwertowanie, zmienianie kolejności i inne modyfikacje przesyłanych wiadomości. Podłączenie do kanału publish-subscribe pozwala na łatwe wysyłanie wiadomości do wielu odbiorców.

Nie jest to, oczywiście, technologia pozbawiona wad. Tym co musimy wziąć pod uwagę jest istnienie wspomnianego już brokera wiadomości. Dodatkowo, wszystkie wspomniane wcześniej możliwości mają swój koszt, jakim jest większy rozmiar aplikacji mobilnej wynikający z konieczności skorzystania z dodatkowych komponentów. Nie jest to duży narzut w przypadku klasycznych aplikacji, jednak zwiększa rozmiary aplikacji mobilnej w stopniu, który może nie pozwolić na jej uruchomienie na prostszych urządzeniach.

W przeciwieństwie do Web serwisów, nie wykształcił się jeden, uniwersalny standard przesyłania wiadomości. W ramach platformy J2ME naturalnym rozwiązaniem jest JMS pochodzący z Javy Enterprise. Jednak inne platformy, jak Windows Mobile mają swoje własne rozwiązania. Tym na co warto by zwrócić uwagę, jest fakt, że stosowanie wiadomości umożliwia zmianę najniższej warstwy komunikacyjnej pomiędzy urządzeniami. O ile w przypadku Web serwisów oczywistym wyborem jest połączenie z internetem, o tyle wiadomości mogą być przesyłane zarówno w ten sposób jak i za pomocą SMS-ów. Pozwala to na

funkcjonowanie aplikacji bez utrzymywania stałego połączenia z internetem, co w przypadku ograniczonej łączności jest mocno utrudnione.

Na pytanie, która metoda, wiadomości czy Web serwisy, jest lepsza, nie ma jednoznacznej odpowiedzi. Zależy to od oczekiwań użytkownika od integrowanego systemu, jego natury oraz aktualnie istniejących komponentów. Wydaje się, że przed podjęciem decyzji warto zadać sobie następujące pytania:

— Czy potrzebujemy gwarancji dostarczenia komunikatu, czy też jest to system w którym nie jest to kluczowe? Web serwisy nie zapewniają tej funkcjonalności i w razie potrzeby musimy ją sami zaimplementować.

— Czy integrujemy system mobilny z istniejącymi już systemami, które porozumiewają się ze sobą za pomocą brokera wiadomości? W takim wypadku wykorzystanie wiadomości jest naturalnym rozwiązaniem pozwalającym na dointegrowanie urządzeń mobilnych do sieci korporacyjnej.

— Czy bierzemy pod uwagę znaczne wydatki, które wynikają z wykorzystania kosztownych, komercyjnych brokerów wiadomości? Wykorzystanie Web serwisów pozwala na znaczne obniżenie kosztów, wynikające z obecności wielu bezpłatnych platform opartych na otwartych standardach.

— Czy szybki czas dostarczenia jest kluczowy dla działania naszego systemu? Wiadomości wprowadzają pewien narzut czasowy, który jednak nie musi być istotny z naszego punktu widzenia.

Po przeanalizowaniu obu metod komunikacji,w części implementacyjnej naszej pracy zdecydowaliśmy się na wykorzystanie Web serwisów z kilku powodów:

— Wykorzystanie komercyjnego oprogramowania nie było możliwe,a dodatkowo wymagałoby uruchomienia skomplikowanej części po

stronie serwerowej. Naszym celem nie było konfigurowanie jednej skomplikowanej platformy, ale raczej poznanie problemów, z którymi należy się liczyć integrując aplikację mobilną.

— Wykorzystanie powszechnie dostępnego pakietu kJORAM (bezpłatnej implementacji JMS dostępnej także dla urządzeń mobilnych) oznaczało tak naprawdę wykorzystanie Web serwisów, które zostały odpowiednio obudowane przez twórców szablonu.

— Tworzony przez nas szablon komunikacyjny musiał działać z jak najmniejszym opóźnieniem, tak aby użytkownik nie odczuwał dyskomfortu związanego z oczekiwaniem na kolejne ekrany. Zastosowanie wiadomości wprowadza dodatkowe opóźnienie, które nie sprzyja responsywności systemu.

2. Metody realizacji aplikacji mobilnej zorientowanej na środowisko enterprise

Na wstępie dyskusji o realizacji mobilnej części platformy integracyjnej zwróćmy uwagę na dostępne na rynku urządzenia, które posiadają preinstalowane systemy operacyjne pozwalające na dodawanie nie przewidzianych przez twórców zewnętrznych rozwiązań. Tego typu systemy operacyjne muszą się charakteryzować udostępnieniem na zewnątrz, do instalowanych przez nas aplikacji, pewnego API pozwalającego na realizację podstawowych usług, takich jak transfer danych czy ich składowanie w pamięci urządzenia. Na chwile obecną (pierwszy kwartał 2009 roku) na rynku dostępne są urządzenia działające w oparciu wymienione poniżej platformy mobilne :

— WMD - Windows Media Devices oparte o system operacyjny Windows Mobile opracowany przez firmę Microsoft
— Symbian - platforma spotykana głównie na smartfonach oferowanych przez firmy takie jak Nokia czy Samsung
— Blackberry - marka wypromowana prze firmę Research in Motion, oferuje urządzenia z preistalowanym systemem operacyjnym opartym na wirtualnej maszynie Java
— Android - system operacyjny oparty na Linuxie, możliwy do rozbudowywania głównie przez aplikacje J2ME (w obecnej chwili system jest w fazie testów)
— pozostałe autorskie systemy operacyjne, udostępniające standardową maszynę wirtualną Java

Po dokładnym przyjrzeniu się powyższym platformom nasuwa się jeden stanowczy wniosek - poza platformami opartymi na J2ME, wykonanie jednej, spójnej aplikacji spełniającej założenia interoperacyjności jest praktycznie niemożliwe. Jedynym rozwiązaniem jest przygotowanie kilku instancji tej samej aplikacji, które niestety nie będą miały ze sobą niczego wspólnego poza przyjętymi założeniami dotyczącymi podstawowej funkcjonalności. Wynika to między innymi z tego, że każda platforma oferuje inne możliwości i przy okazji wprowadza inne ograniczenia. Co więcej tego typu różnicowanie występuje również w obrębie samych platform. Na przykład J2ME posiada profile związane na stałe z modelami urządzeń, które decydują o tym, jakie funkcjonalności dostępne są dla programów. Wskazane powyżej ograniczenia techniczne wymuszają przy próbie wejścia na rynek aplikacji mobilnych, już na etapie planowania produktu, wybranie konkretnej platformy docelowej, na jakiej oferowane będzie nasze rozwiązanie. Wraz z sukcesem rozwiązania, możliwa będzie dalsza ekspansja na pozostałe platformy. Przed tą ostateczną ewaluacją wartości pomysłu na produkt, jest bardzo ryzykownym podjęcie próby równoległego wprowadzenia go na różnych, niekompatybilnych platformach. Zwiększa to znacznie próg wejścia oraz utrudnia poprawną kontrolę nad stopniem dojrzałości produktu. Co więcej przy słabym zarządzaniu funkcjonalnościami może wystąpić problem powstania różnic funkcjonalnych pomiędzy różnymi wersjami naszej aplikacji. Tego typu problemy wpłyną w sposób bezpośredni na postrzeganie naszego produktu przez klientów, w szczególności gdy będziemy mieli do czynienia z korporacjami, w obrębie których stosowane są różne rozwiązania mobilne.

Uzbrojeni w wiedzę o pewnych możliwych do napotkania problemach skupimy teraz naszą uwagę na doborze metodologii, według której będzie powstawała nasza aplikacja. Wyboru dokonuje się zwykle spośród trzech klasycznych podejść. Są to:

— Budowa aplikacji w oparciu o przeglądarkę mobilną
— Budowa aplikacji przy użyciu generatora aplikacji
— Budowa aplikacji od podstaw (przy ewentualnym wsparciu szkieletów aplikacyjnych)

Każde z tych podejść ma swoje wady i zalety, które powodują, że można do nich przypisać różne zastosowania. W następnych rozdziałach przyjrzymy się każdemu z tych podejść wyróżniając najważniejsze cechy każdego z nich oraz przedstawiając pewne nierozłącznie związane z nimi środowiska uruchomieniowe oraz programistyczne.

2.1. Przeglądarka mobilna

Urządzenia mobilne, w tym telefony komórkowe, mają wbudowane przeglądarki internetowe, za pomocą których coraz częściej można przeglądać wszystkie dostępne w Internecie strony (w przeciwieństwie do sytuacji sprzed kilku lat, gdy jedynie część z nich, posiadająca wersję WAP, była dostępna). W przeciągu bardzo krótkiego czasu dokonał się duży postęp w rozwoju mobilnych przeglądarek. Ciągle jednak nie zapewniają one wygody porównywalnej z przeglądaniem stron na tradycyjnym komputerze. Z drugiej strony, korzystanie z aplikacji zbudowanych w oparciu o model trójwarstwowy (z użyciem tak zwanego cienkiego klienta - czyli przeglądarki WWW) niesie za sobą szereg korzyści, takich jak brak potrzeby instalowania aplikacji, łatwość wdrożenia czy wreszcie duża przenośność pomiędzy różnymi platformami.

2.1.1. Rodzaje przeglądarek mobilnych

Omawiając temat przeglądarek mobilnych należy zwrócić uwagę na to, że na rynku znajduje się obecnie wiele tego typu produktów. Nie wszystkie są bezpośrednią konkurencją. Część z nich może działać

tylko w obrębie platformy na którą zostały zaprojektowane. Szczególnie widoczne jest to w wypadku przeglądarki Safari, która jest używa praktycznie tylko na urządzeniach mobilnych firmy Apple. Należy brać to pod uwagę przy projektowaniu strony mobilnej. Może się okazać, że można ograniczyć wysiłki zapewnienia pełnej zgodności na wszystkich możliwych przeglądarkach, ponieważ w obrębie organizacji funkcjonują urządzenia tylko jednej marki (na przykład Blackberry).

Uproszczone przeglądarki mobilne

Pierwszym rodzajem przeglądarek spotykanych na urządzeniach mobilnych są przeglądarki wyświetlające uproszczoną wersję stron internetowych. Dlatego przeniesienie istniejących portali korporacyjnych pod platformę, która ich używa wiąże się z dodatkową pracą wynikającą z konieczności przystosowania sposobu wyświetlania portali do możliwości przeglądarki.

Opera Mini Jest to najpopularniejsza przeglądarka mobilna działająca w oparciu o platformę Java 2 Micro Edition. Charakterystyczną cechą tej przeglądarki jest serwer proxy, który optymalizuje zawartość stron przed załadowaniem. Wadą tego rozwiązania może być niechęć do używania go w przedsiębiorstwach w celu uzyskiwania dostępu do wrażliwych danych. Strach przed ich utratą może być dużo większy niż korzyści jakie można odnieść z używania tego programu.

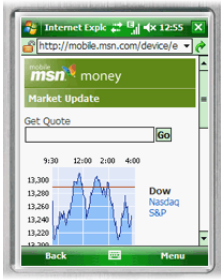
Blackberry Browser Blackberry Browser jest domyślną aplikacją pozwalającą na przeglądanie zasobów intranetu lub internetu. Podstawowym trybem działania tej przeglądarki jest umożliwianie dostępu do wewnętrznych zasobów firmy. Wiąże się to bezpośrednio ze sposobem działania urządzeń Blackberry opartym o sieć wirtualną. Każde urządzenie mobilne jest tunelowane przez powszechnie dostępną sieć telekomunikacyjną do sieci wewnętrznej firmy. Tunelowanie zajmuje się działający w obrębie firmowej sieci serwer BES (Blackberry Enterprise Server). Jest on jedynym elementem wystawionym na zewnątrz i poprzez szyfrowanie połączenie umożliwia przekierowywanie pakietów z i do urządzeń mobilnych. Dzięki temu każdy użytkownik ma zapewniony bezpieczny dostęp do informacji firmowych. Ma to ogromne znaczenie przy wyborze sposobu umobilniania. W przypadku pozostałych przeglądarek mobilnych jedną gwarancją bezpieczeństwa jest http przy czym nie ma żadnego zapewnienia bezpieczeństwa danych,

które przechodzą przez serwery proxy (tak jak w wypadku przeglądarki Opera Mini). Dlatego w wypadku, gdy głównym czynnikiem decydującym o wyborze jest bezpieczeństwo warto rozważyć zastosowanie rozwiązań, które je gwarantują.

Pełne przeglądarki mobilne

Opera Mobile Opera Mobile jest jedną z najpopularniejszych pełnowartościowych przeglądarek mobilnych. Została zbudowana na bazie silnika przeglądarki desktopowej. Zachowuje prawie całkowitą zgodność z pełną wersją. Co oznacza, że w środowisku opartym tylko i wyłącznie o tą markę możliwe jest używanie stron internetowych bez żadnych zmian przystosowujących je do wersji mobilnej. Niestety ze względu na niewielką popularność przeglądarki desktopowej oraz jej całkowity brak zgodności z dwoma dominującymi przeglądarkami (wynikający z bardzo restrykcyjnego pilnowania standardów przez deweloperów Opery), zwykle niezbędne są zmiany w konstrukcji strony. Działa na urządzeniach korzytających z platform Symbian S60 oraz Windows Mobile.

Internet Explorer Mobile Ponieważ firma Microsoft posiada od dawna systemy operacyjne przeznaczone na platformy mobilne (Windows Mobile oraz starszy Windows CE) razem z nimi zawsze dostarczana jest przeglądarka internetowa. Jest sporo wolniejsza od konkurencji, ale ze względu na fakt, że jest preinstalowana na platformach cieszy się dużym powodzeniem wśród mobilnych użytkowników. Przy wdrażaniu rozwiązań mobilnych w firmie, w której używane są urządzenia typu Pocket PC, należy się spodziewać, że zostanie ona narzucona z góry jako klient naszej mobilnej strony internetowej. Pozwala na działanie zarówno w trybie mobilnym, odczytującym uproszczony zapis html, jak i na pracę z pełnymi wersjami stron internetowych. Niestety silnik tej przeglądarki został napisany zupełnie od nowa przez co nie jest w pełni zgodny z tym, który jest używany w pełnej wersji Internet Explorera. Może to zwiększyć nakład pracy niezbędny do uzyskania mobilnej wersji naszej strony.



Safari Przeglądarka firmy Apple używana głównie w jej produktach. Jej mobilna wersja pojawiła się na rynku w momencie wprowadzenia telefonu iPhone. Charakteryzuje się bardzo dobrą obsługą JavaScripta oraz pełną zgodnością z jej wersją desktopową. W odróżnieniu od

produktów Opera Software na urządzeniach stacjonarnych firmy Apple jest one przeglądarką dominującą, więc dzięki pełnej zgodności nakłady pracy potrzebne na przystosowanie oraz wdrożenie są znacznie mniejsze niż w przypadku konkurencji.

2.1.2. Zagadnienia związane z projektowaniem stron

W poprzednich rozdziałach zostało przedstawionych kilka najpopularniejszych przeglądarke mobilnych. Wyróżnione zostały pewne elementy, które wymuszają zastosowanie specjalnej metodologii przy przystosowywaniu stron internetowych. Podział na przeglądarki uproszczone i pełne przeglądarki mobilne sugeruje dwa różne podejścia do tego zagadnienia. Pierwszy sposób przystosowania stron internetowych zakłada, że mamy do czynienia z pełnymi przeglądarkami. Ponieważ różnią się one głównie brakiem wsparcia dla niektórych elementów Javascript, to praca związana z ich umobilnianiem wiąże się głównie z upraszczeniem pewnych elementów lub przepisywaniem ich tak by były kompatybilne. Zwykle trzeba zrezygnować też z pewnych zaawansowanych funkcjonalności, które działają w oparciu o technologię Javascript.

2.2. Generatory aplikacji

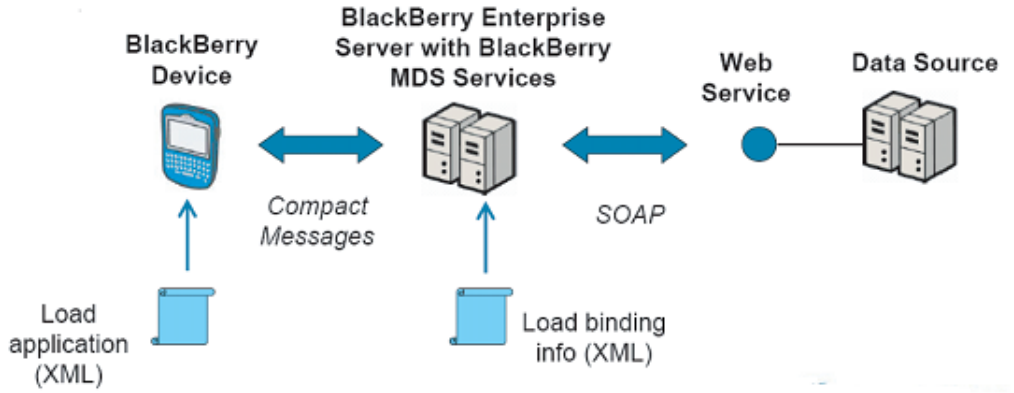
Wielu producentów podejmuje próby zbudowania środowisk umożliwiających szybkie tworzenie aplikacji mobilnych (tak zwane RAD - Rapid Application Development). Mają one łączyć zalety obu przedstawionych wcześniej światów :

— Pozwalać na łatwe i szybkie tworzenie aplikacji dostępowych - tak jak mobilne przeglądarki
— Dostarczać elastyczności i rozbudowanych interfejsów użytkownika - tak jak aplikacje mobilne

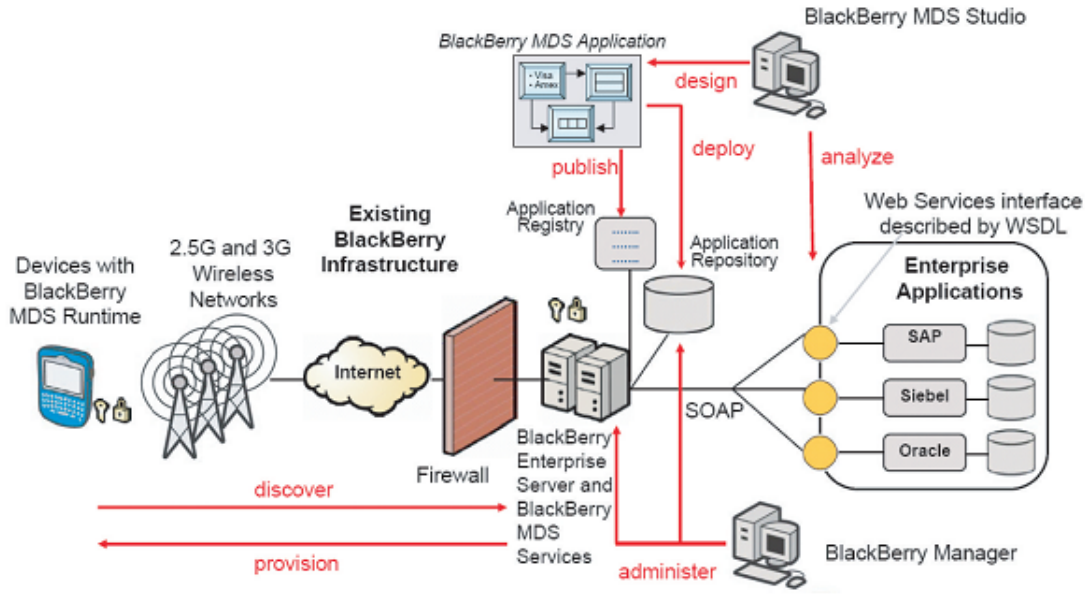
W dalszej części niniejszej pracy przedstawione zostaną dwa tego typu rozwiązania, pozwalające na oszczędzenie znacznej ilości pracy przy budowaniu nowych aplikacji.

Generator dedykowany - Blackberry MDS

Blackberry MDS jest przykładem narzędzia wyspecjalizowanego do tworzenia aplikacji tylko i wyłącznie na platformę jednego producenta. Rozwiązanie to opiera się na integracji małej aplikacji zainstalowanej na urządzeniach klienckich (MDS Runtime) z usługą typu Web Service działającą na serwerze. Dzięki niemu możliwe jest tworzenie aplikacji używając podejścia opisowego. Zamiast pisać całą aplikację używając języka programowania wraz z bibliotekami oferującymi interfejs użytkownika, możemy przy użyciu graficznego edytora projektować takie elementy jak ekrany czy pola z danymi. Wszystko to przy użycia standardowego drag and drop. Następnie możemy połączyć tak zbudowane komponenty przy użyciu różnych przewodników(wizards) czy też edytorów. Co więcej aplikacja MDS daje nam pełną kontrolę nad przepływem sterowania - przy pomocy języka JavaScript jesteśmy w stanie zaimplementować niemal dowolną logikę aplikacji. Aplikacje typu enterprise oraz źródłą danych dostępne są dla aplikacji mobilnej poprzez standardową technologię Web Service.



Aplikacja MDS dla Blackberry jest to aplikacja typu rich-client zbudowana na podstawie metadanych zdefiniowanych w języku XML. Te metadane wraz z zasobami takimi jak obrazki gromadzone są w pakietach, które następnie publikowane są na serwerze BES. Serwer ten przy użyciu technologii push rozsyła aplikacje do terminali mobilnych Blackberry. Dzięki zcentralizowanemu sposobowi dystrybucji oraz zarządzania aplikacjami mobilnymi koszty utrzymania mobilnej infrastruktury są znacznie niższe.



Usługa Blackberry MDS Usługa MDS Blackberry została zaprojektowana tak by umożliwić wymianę danych pomiędzy urządzeniami Blackberry a aplikacjami typu enterprise. Zapewnia ona:

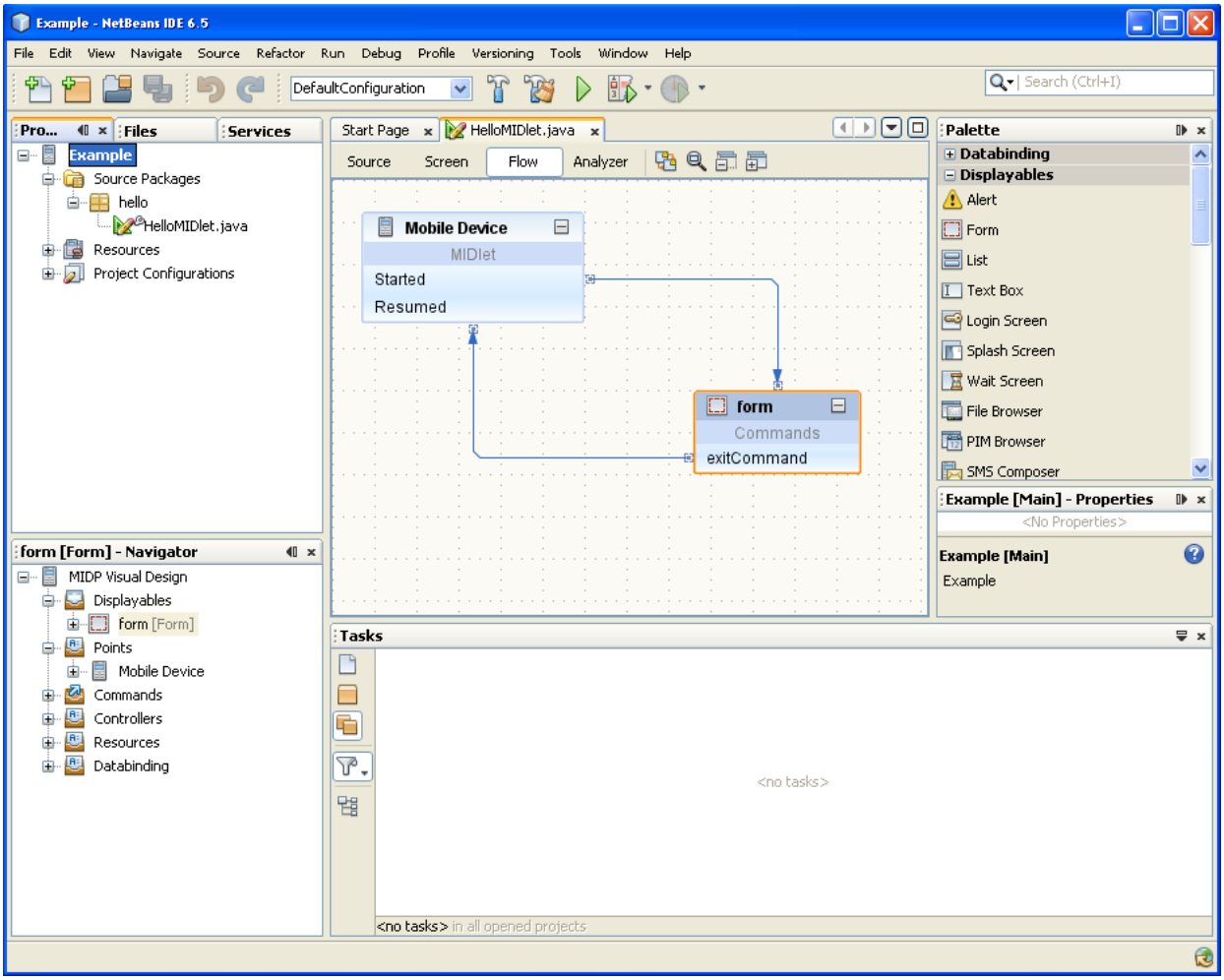
— Izolację platformy od szczegółów związanych z integracją źródeł danych
— Standard asynchronicznej wymiany komunikatów
— Przezroczyste dla użytkownika i developera protokoły bezpieczeństwa
— Transformacje wiadomości
— Wspomnianą wcześniej zcentralizowaną administrację przy pomocy technologii push

Blackberry MDS runtime Po stronie urządzenia mobilnego zainstalowany jest kontener aplikacji MDS. Zajmuje się on instalacją, wyszukiwanie oraz przeglądaniem aplikacji MDS. Dostarcza również niezbędne funkcjonalności do tych aplikacji, takie jak interfejs użytkownika, kontener danych oraz usługi komunikacyjne klient-server.

Generator uniwersalny - Netbeans Mobile IDE

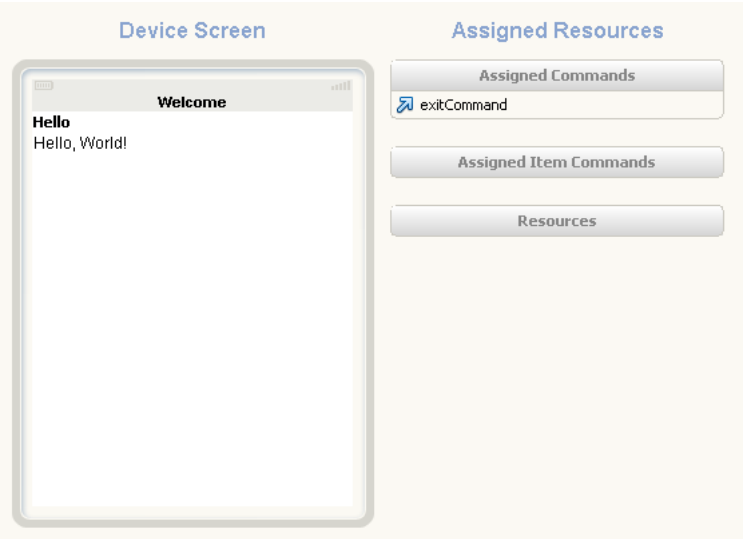
Drugim przykładem generatora aplikacji mobilnych jest Netbeans Mobility Pack oraz związane z nim środowisko programistyczne Netbeans. W przeciwieństwie do poprzedniego przykładu, to środowisko pozwala budować uniwersalne aplikacje, które działają na niemalże dowolnej platformie mobilnej wspierającej J2ME. Rozpoczynając tworzenie aplikacji natychmiast natykamy się na udogodnienia przygotowanie przez twórców Netbeans. Zamiast pisać setki linijek kodu Java możemy użyć specjalnych narzędzi generujących go dla nas. Do konfigurowania preferowanego przez nas zachowania aplikacji służą interaktywne diagramy przepływu sterowania pomiędzy ekranami aplikacji oraz graficzne edytory interfejsu użytkownika

pozwalające na budowanie, z gotowych komponentów, ekranów naszej mobilnej aplikacji. Poniżej przedstawiony jest przykład diagramu przepływu sterowania aplikacji typu Hello World.

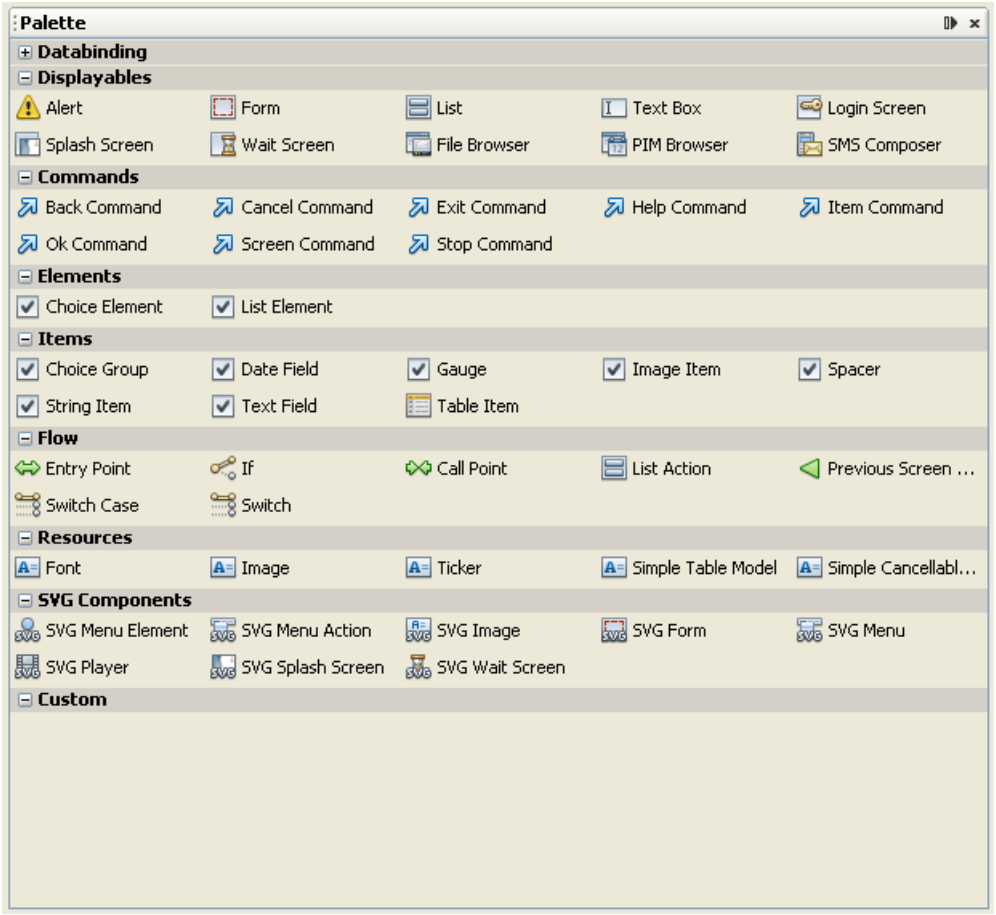


Przykładowy diagram przepływu składa się z dwóch elementów - bloku reprezentującego stan wyjściowy aplikacji mobilnej (a jednocześnie stan końcowy) oraz bloku reprezentującego główny i jedyny ekran. Elementy te połączone są dwiema strzałkami, które przedstawiają przepływ seterowania. Strzałka skierowna w stronę formy ma swój początek w zdarzeniu Started. Jest to zdarzenie wywoływane w momencie uruchomienia aplikacji. Dzięki takiej konfiguracji natychmiast po inicjalizacji aplikacja przechodzi do ekranu głównego. Strzałka w drógą stronę podpięta jest do komendy exitCommand. Wyzwala ona

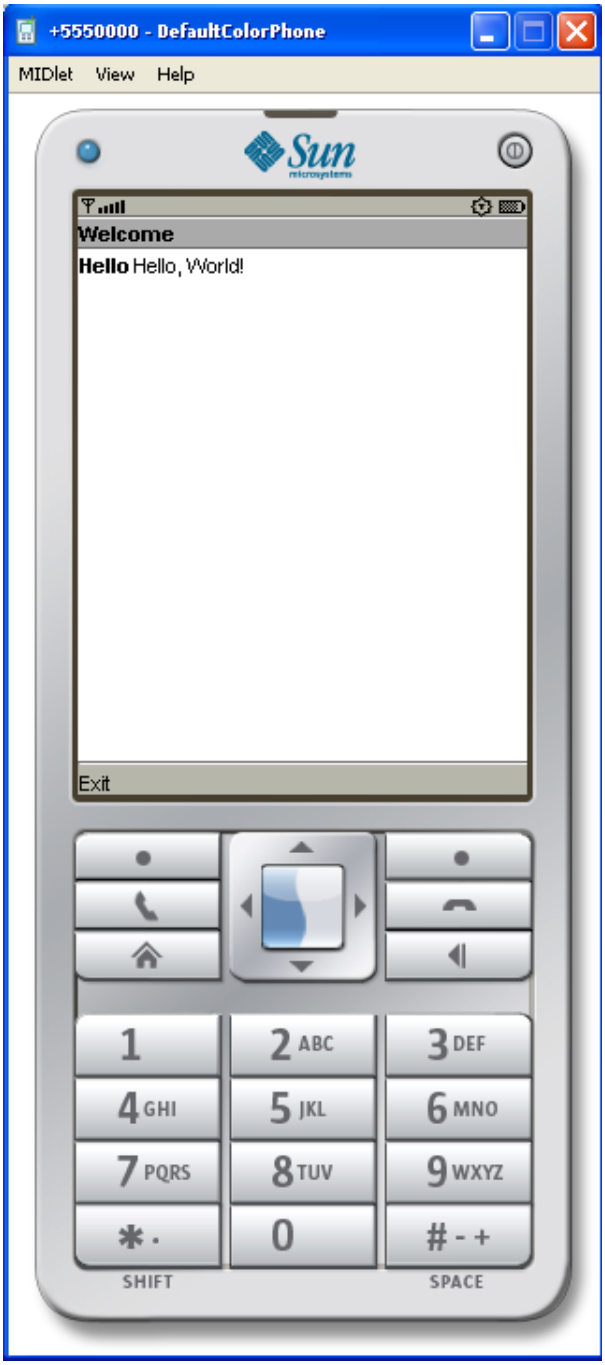
zamknięcie aplikacji (jej drugi koniec wskazuje blok wyjściowy). Ekran główny został zaprojektowany tak by wyświetlic napis oraz komendę pozwalajacą na opuszczenie aplikacji.



Tak jak wspomniano wcześniej tego typu ekrany buduje się z gotowych komponentów. Środowisko netbeans udostępnia zestaw predefiniowanych komponentów o podstawowej funkcjonalności. Użytkownik może odnaleźć je na palecie bocznej programu. W wersji 6.5 paleta ta wygląda tak jak na poniższym rysunku.

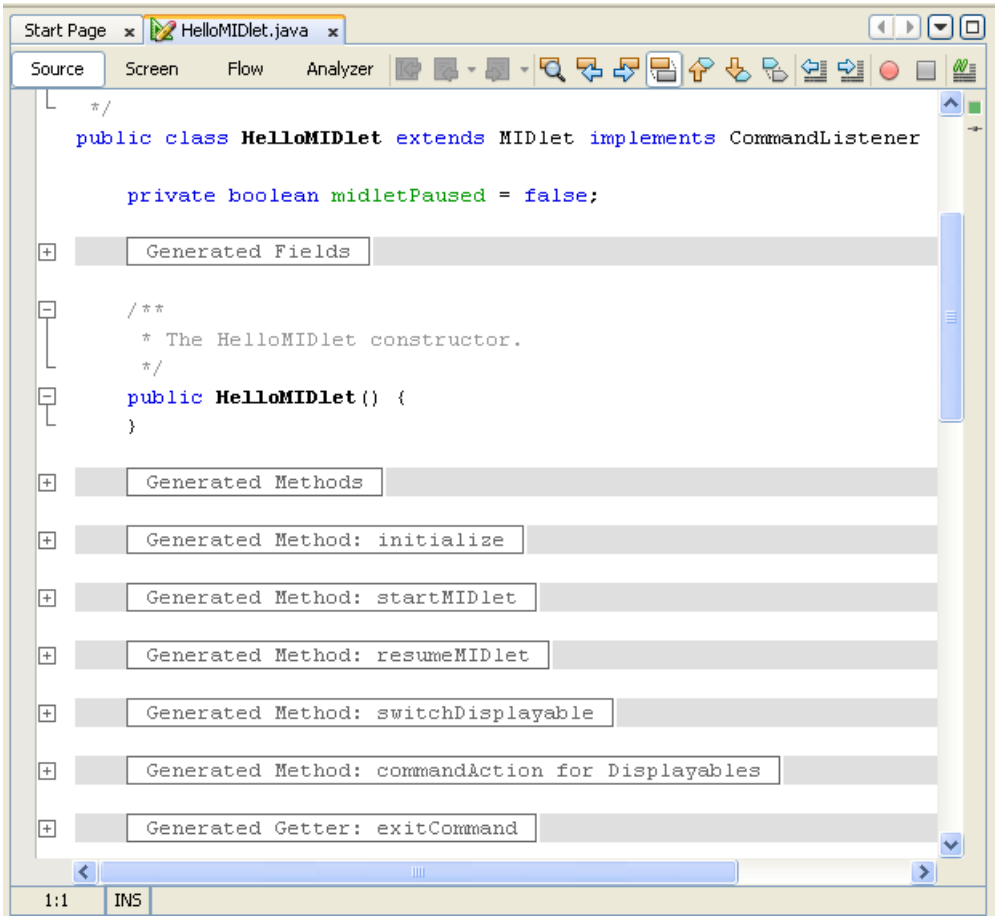


Komponentowe budowanie aplikacji znacznie przyspiesza proces tworzenia. Co więcej w razie potrzeby istnieje możliwość definiowania własnych komponentów, które później można wykorzystywać w wielu aplikacjach mobilnych. Miejsce na własne komponenty można zobaczyć w zakładce custom na standardowej palecie komponentów. Dzięki zastosowaniu przedstawionych wcześniej edytorów udało się zbudować bardzo prostą aplikację zdolną do wyświetlenia napisu. Zostało to zrobione bez napisania linijki kodu w javie, używając jedynie narzędzi graficznych. Tak zbudowana aplikacja działa na praktycznie każdej konfiguracji mobilnej i gwarantuje jednakowe zachowanie pomiędzy różnymi platformami.



Do konstruowania bardziej zaawansowanych aplikacji należy posłużyć się edytorem javy. Dostęp do kodu źródłowego automatycznie generowanej aplikacji pozwala na elastyczne modyfikowanie jej zachowania w zakresie dużo szerszym niż pozwala na to jakakolwiek graficzna konfiguracja. Co więcej każdy element, który został przez nas skonfigurowany w jednym ze wspomnianych wcześniej edytorów

jest powiązany z kodem java, z którym można się zapoznać w zakładce source.



Wyszarzone obszary to kod, który został automatycznie wygenerowany przez środowisko netbeans. W pozostałych obszarach można wprowadzać własny kod java, który może kontrolować dowolne aspekty zachowania się aplikacji mobilnej.

Przedstawione środowisko programistyczne jest jednym z popularniejszych podejść do probelmu szybkiego konstruowania aplikacji mobilnych. Jest elastyczne i efektywne dlatego posłużymy się nim przy budowie przykładowego systemu.

2.3. Rozwiązania dedykowane

Innym podejściem, przypominającym aplikacje zbudowane w oparciu o model dwuwarstwowy, jest stworzenie aplikacji mobilnej, która samodzielnie łączy się z Internetem i umożliwia wyświetlanie i modyfikację danych w sposób przewidziany przez jej twórców. Ogranicza to oczywiście funkcjonalność takiej aplikacji do zakresu przewidzianego w danej wersji, a wszelkie zmiany wymagają jej aktualizacji. W zamian za to otrzymujemy większą szybkość działania, możliwość walidacji danych po stronie klienta oraz elastyczność w tworzeniu interfejsu (który w tym przypadku nie jest ograniczony do kontrolek zapewnianych przez przeglądarkę WWW).

Wadą, z której niewiele osób zdaje sobie sprawę, jest mniejsza przenośność aplikacji, choćby były one napisane w języku projektowanym z myślą o niezależności od platformy, takim jak Java. Wynika ona z różnorodności urządzeń dostępnych na rynku. Całkowita przenośność możliwa jest do uzyskania tylko dla prostych aplikacji, nie korzystających z możliwości oferowanych przez konkretne modele urządzeń mobilnych. Przy próbie zbudowania czegokolwiek większego natychmiast natkniemy się na niewidzialny mur generowany przez ogromne różnice pomiędzy dostępnością profili na poszczególnych platformach oraz przez różnice w implementacji pewnych części samych maszyn wirtualnych. Co więcej tu opisana sytuacja wskazuje na problem, który generuje stworzona z myśla o przenośności platforma J2ME. Jeszcze większe problemy pojawiają się gdy chcemy rozszerzyć zasięg naszej aplikacji o kolejne segmenty rynku, takie jak użytkownicy systemów Symbian czy Windows Mobile. W tym momencie okazuje się, że zostaniemy zmuszeni do napisania nie jednej aplikacji, ale wielu, które często mogą nie mieć prawie żadnych części wspólnych.

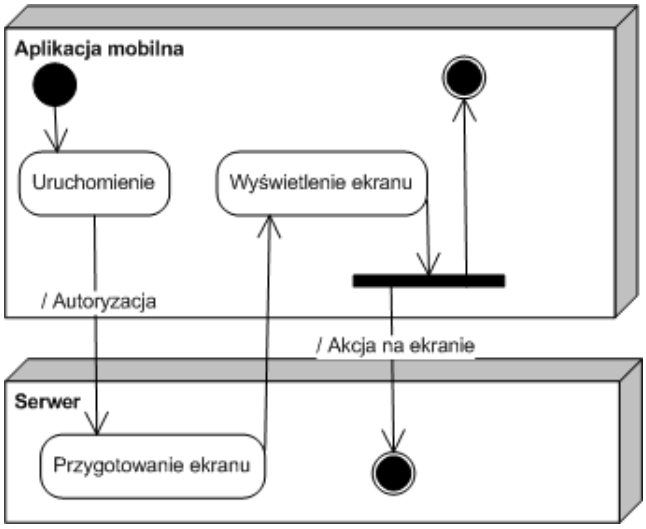
3. Szablon komunikacyjny

Do zademonstrowania możliwości komunikacyjnych pomiędzy systemem stacjonarnym i mobilnym stworzyliśmy szablon ułatwiający tworzenie aplikacji posiadających mobilny interfejs, z poziomu którego można sterować aplikacją serwerową. Rozwiązanie, które przyjęliśmy jest na tyle ogólne, że umożliwia rozwiązanie kilku podobnych problemów, które przedstawimy dalej.

Ideą stojącą za stworzeniem szablonu było połączenie zalet zapewnianych przez aplikację zbudowaną w oparciu o model trójwarstwowy z tymi typowymi dla samodzielnej aplikacji. Stworzone rozwiązanie stanowi raczej pomysł, który można rozwijać i wykorzystywać do budowy bardziej skomplikowanych systemów. Dlatego też możliwości takiego podejścia nie są ograniczone przez stworzony szablon.

Szablon składa się z dwóch części - mobilnej i serwerowej. Użytkownik może określić wygląd ekranu przesyłanego do urządzenia mobilnego, który po stronie mobilnej interpretowany jest do postaci elementów interfejsu ciężkiego klienta, na przykład pól tekstowych, wyboru czy rozwijanych list. Rozszerza to możliwości aplikacji w stosunku do zwykłej strony internetowej o elementy niedostępne w przypadku statycznej strony WWW jak rysunki wektorowe,obsługa przycisków urządzenia czy możliwość wyświetlania strony przez jakiś czas.

W typowym przypadku scenariusz użycia aplikacji będzie wyglądał następująco:



— Tworzymy nowy 'ekran' na serwerze. Oznacza to określenie wyglądu i zachowania aplikacji w formacie XML. Określamy dostępne dla danego ekranu akcje.
— Aplikacja mobilna łączy się z serwerem i pobiera przygotowany dla niej ekran. Użytkownik wykonuje jedną z dostępnych na nim akcji, a informacja o tym jest przesyłana do serwera.
— Aplikacja mobilna pobiera kolejny ekran przygotowany przez serwer i dalej postępuje według powyższego schematu. Ponieważ to serwer jest w całości odpowiedzialny na przygotowanie ekranu, pojawia się możliwość kontrolowania ekranu i jego zachowania po stronie klienta w dowolny sposób.

3.1. Możliwości

Aplikacja oparta o tak zaprojektowany model posiadałaby cechy typowe dla modelu dwuwarstwowego - w tym szybki czas reakcji, niezależność wyglądu aplikacji od stosowanego klienta oraz możliwość przeprowadzenia pewnych dodatkowych operacji (jak walidacja) po stronie klienta. W idealnym przypadku mógłby być dynamicznie przesyłany kod wykonywany po stronie klienta (odpowiednik JavaScript w przypadku przeglądarki internetowej). Pozbawiona

byłaby też typowych wad obu rozwiązań - brak konieczności aktualizowania aplikacji przy drobnych poprawkach (poza błędami w zainstalowanym oprogramowaniu). Nie byłaby również ograniczona do mechanizmów oferowanych przez statyczne strony WWW - można by było wyświetlać dowolne ekrany, a na nich wykonywać dowolne akcje. Dodatkowo, ponieważ stan aplikacji przechowywany byłby na serwerze, można w dowolnym momencie zmienić urządzenie mobilne, wyłączyć aplikację, a potem powrócić do zapamiętanego wcześniej miejsca.

Nowe podejście W przypadku tworzenia aplikacji mobilnych naturalna jest chęć przeniesienia funkcjonalności dostępnych w aplikacjach 'biurkowych' na urządzenie mobilne. Nie zawsze jest to podejście słuszne - użytkownik korzystający z urządzenia mobilngo - palmtopa czy telefonu komórkowego ma znacznie ograniczone możliwości działania i tym co się w takiej sytuacji liczy jest szybkość i łatwość obsługi aplikacji, którą można postawić przed jej dużymi możliwościami. Dlatego naszym zdaniem, celem tworzenia aplikacji mobilnych nie powinno być dostarczenie wszystkich możliwych funkcjonalności, ale raczej maksymalna ergonomia i prostota użycia, które przyświecały nam w trakcie tworzenia szablonu.
Ponadto, techniczne rozwiązania przyjęte w szablonie nie były - zgodnie z naszą wiedzą - do tej pory stosowane. Nasz szablon opiera się na bibliotece KUIX, która pozwala na wykorzystanie formatu XML do stworzenia interfejsu użytkownika. Pozwala to na dynamiczne ładowanie komponentów wcześniej przygotowywanych na serwerze. Naturalną drogą rozwoju byłoby rozbudowanie szablonu o możliwość dynamicznego ładowania kodu, mechanizmów walidacji czy tworzenia grafiki.

3.1.1. Zastosowania

Opisywana idea ułatwia tworzenie określonego typu aplikacji i tak jak każdy szablon nie pokrywa wszystkich istniejących rozwiązań. Nasz szablon ułatwi pisanie aplikacji, które nie wymagają przesyłania dużych ilości danych i od której oczekujemy interaktywności większej niż w przypadku strony internetowej pozbawionej JavaScript.

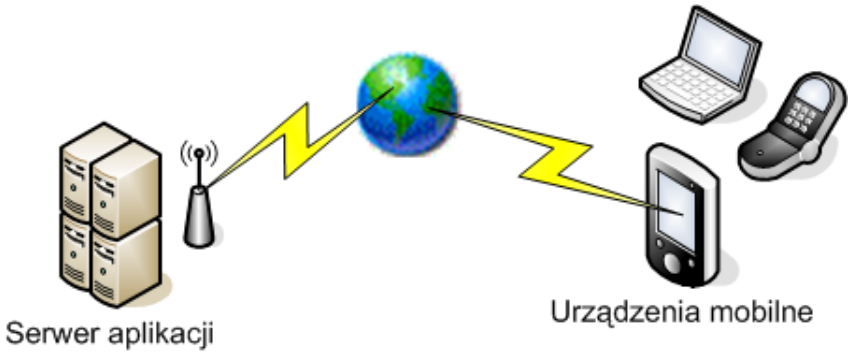
Ankietowanie Jednym z zastosowań które możemy sobie wyobrazić jest zdalne ankietowanie użytkowników. Stworzenie aplikacji która na ekranie wyświetla listę dostępnych odpowiedzi i pozwala na wybranie jednej z nich, jest przy użyciu przedstawianego szablonu prostym zadaniem.

Zdalne podejmowanie decyzji Podobnie jak powyżej, użytkownicy mogą wybrać jedną z możliwych decyzji, w takim przypadku potrzebne będzie rozbudowanie szablonu o zaawansowane mechanizmy bezpieczeństwa, jednak idea pozostaje podobna.

Zdalne wyświetlanie treści Urządzenie mobilne może być też jedynie miejscem które wyświetla dostarczane do niego informacje. Interakacja z użytkownikiem może być w tym momencie wyłączona, a urządzenie mobilne może pełnić rolę zdalnie kontrolowanego ekranu.

3.2. Architektura

Tak jak zostało to już wspomniane szablon składa się z dwóch głównych części - serwerowej oraz mobilnej. Przed przejściem do szczegółów implementacyjnych zaprezentujemy koncepcję całego systemu.



Jak widać, urządzenia mobilne będą się łączyły przez Internet do uruchomionego serwera aplikacyjnego. Każde z nich wysyła żądanie pobrania ekranu, który jest dynamicznie przygotowywany w zależności od stanu aplikacji dla danego użytkownika. Warto tu zwrócić uwagę, że szablon jest pomyślany właściwie wyłącznie dla urządzeń mobilnych - stosowane rozwiązania, są albo niedostępne, albo mało przydatne w przypadku komputerów stacjonarnych, o czym napiszemy w dalszej części pracy.

Podczas projektowania szablonu pojawiły się wątpliwości co do sposobu komunikacji pomiędzy elementami systemu. Początkowa i bardzo przekonująca koncepcja zakładała wykorzystanie JMS - Java Messaging System, specyfikacji umożliwiającej przesyłanie wiadomości w Javie. Przemawiał za tym szereg argumentów:

— Niezawodność - gwarancja dostarczenia wiadomości w warunkach łączności bezprzewodowej, która jest szczególnie narażona na przerwy w transmisji.

— Niezależność systemów - użycie wiadomości znacznie ułatwiłoby ewentualne zmiany po stronie serwera aplikacyjnego, ze względu na dodatkowy element pośredniczący jakim jest serwer wiadomości.

— Łatwość implementacji - przesyłane wiadomości mogą zawierać zserializowane obiekty, co zdejmuje z programisty obowiązek samodzielnej serializacji i deserializacji danych.

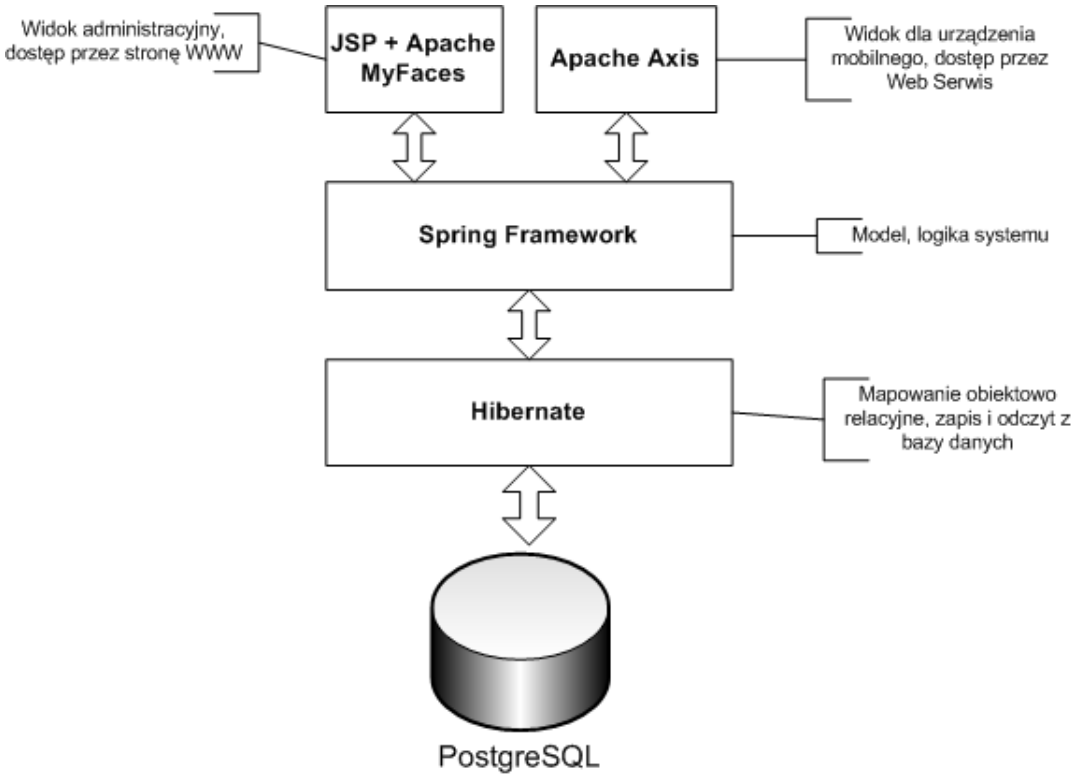
Jak się jednak okazało,istotne ograniczenia techniczne po stronie urządzeń mobilnych uniemożliwiły skorzystanie z JMS. Jest to technologia wymagająca znacznych zasobów, niedostępnych w specyfikacji Java Micro Edition - platformy na której stworzyliśmy nasz szablon. Nie daje ona możliwości bezpośredniego podłączenia do kanału komunikacyjnego, co wymusza korzystanie z zewnętrznych bibliotek. Jednak nawet wtedy, okazuje się, że takie biblioteki korzystają z Web Serwisów jako technologii pośredniczącej w komunikacji. Podobnie, automatyczna serializacja i deserializacja w takiej sytuacji nie jest jeszcze wspierana.
Z tych powodów zdecydowaliśmy się na wykorzystanie Web serwisów jako warstwy komunikacyjnej i samodzielne rozwiązanie problemów, które się z tym wiążą.

3.2.1. Aplikacja serwerowa

Przedstawimy teraz architekturę części serwerowej, komponenty z których się składa i technologie, które wykorzystują.
Aby zrozumieć projekt całego systemu wprowadźmy pewne pojęcia w naszym systemie, z których wynikają dalsze rozwiązania:

— *Ankieta* - treść, dla której istnieją dostępne zdefiniowane odpowiedzi.
— *Odpowiedź* - jedna z możliwości wyboru dostępnych dla danej ankiety
— *Użytkownik* - konto z którym związane są uprawnienia i udzielone w ankietach odpowiedzi. Jeden użytkownik może udzielić wielu odpowiedzi w trakcie działania całego systemu.

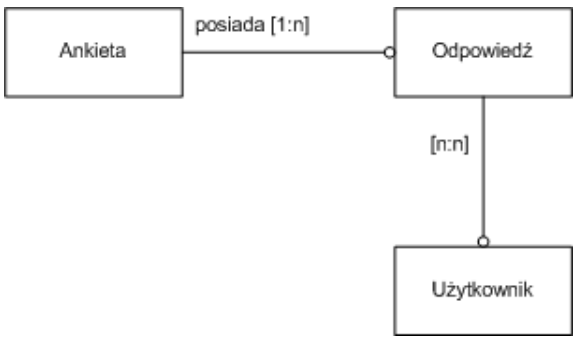
Aplikacja serwerowa zbudowana została w oparciu o model trójwarstwowy, z podwójnym dostępem (widokiem) do danych.



Przedstawmy teraz poszczególne warstwy systemu, wraz z przyjętymi w nich założeniami.

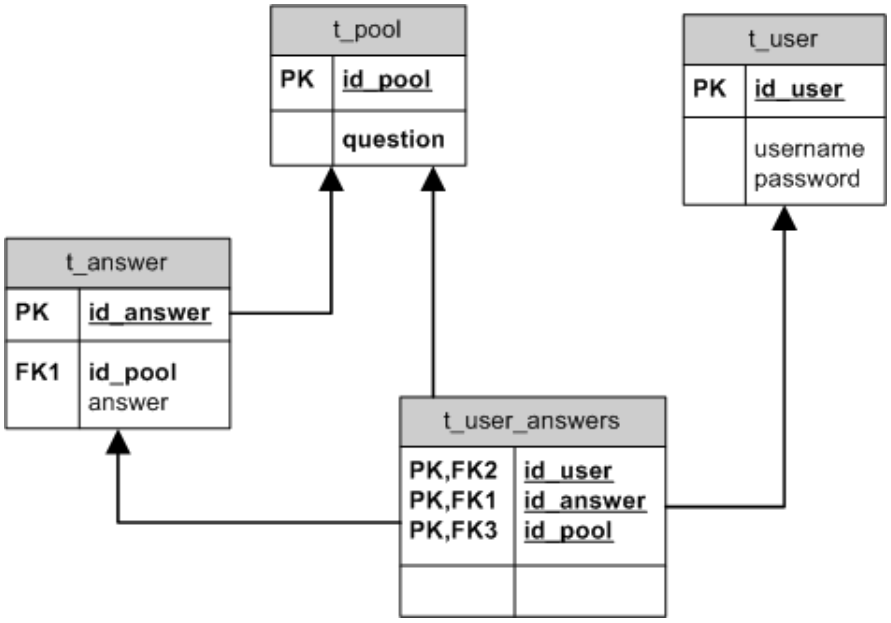
Serwer Aplikacja serwerowa jest uruchamiana na kontenerze Apache Tomcat 6.0.

Baza danych W oparciu o przedtawione wcześniej założenia stworzony został prosty schemat bazy danych:



Choć wydaje się, że ogranicza to działanie systemu do możliwości wyboru odpowiedzi z dostępnej listy, to jednak schemat bez problemu

można rozbudować o dodatkowe pola, które pozwolą na podawanie wolnych (nieograniczonych dostępną listą wyboru) odpowiedzi.
Na postawie schematu koncepcyjnego stworzony został schemat fizyczny bazy danych:



Hibernate Dostęp do danych realizowany jest za pomocą mapowania relacyjno-obiektowego Hibernate. Teoretycznie pozwala to, a na pewno znacznie ułatwia ewentualną zmianę dostawcy bazy danych. Dodatkowo uwalnia nas z konieczności ręcznego pisania zapytań SQL.

Spring Framework Spring jest zestawem narzędzi ułatwiającym tworzenie aplikacji. Pozwala on, między innymi, na deklaratywne zarządzanie transakcjami, zarządzanie czasem cyklem życia obiektów oraz zwalnia programistę z konieczności pisania często powtarzanego kodu przez dostarczenie odpowiednich szablonów. Dodatkowo wymusza on pewien model programowania i sprawia, że pisane aplikacje są bardziej przejrzyste i łatwiejsze w utrzymaniu. Jak już zostało to wspomniane aplikacja serwerowa jest aplikacją typu Web, z którą to framework Spring doskonale się integruje.

Apache Myfaces Myfaces jest implementacją specyfikacji Java Server Faces, która pozwala na wygodne tworzenie widoku aplikacji. W naszym przypadku interfejsem będzie dynamicznie generowana strona WWW, choć specyfikacja JSF nie wprowadza takiego ograniczenia. Warstwa ta będzie służyła do administrowania systemem, przeglądania podjętych decyzji i zarządzania ankietami i użytkownikami.

Apache Axis2 Axis2 to kontener webserwisów,który odpowiada za odbieranie zdalnych wywołań procedur i przekazywanie ich do odpowiednich obiektów które je wykonują. Zwalnia też programistę z konieczności samodzielnego przetwarzania XML w żądaniach, umożliwia też automatyczną serializację i deserializację obiektów Javowych.

Transformata XSLT Elementem wspomagającym w systemie jest transformata XSLT wykonywana na odpowiedziach przesyłanych do urządzenia mobilnego. Jej celem jest utworzenie widoku dla urządzenia mobilnego w sposób niezależny od generowanej odpowiedzi. W wyniku wywołania procedury generującej ekran, tworzony jest XML, zawierający podstawowe informacje, a następnie przetwarzany jest on w opisany sposób, dzięki czemu odpowiedź można dowolnie modyfikować bez zmiany kodu aplikacji.

3.2.2. Aplikacja mobilna

Cechą wyróżniającą podejście prezentowane w niniejszej pracy od typowych rozwiązań problemu integracji systemów enterprise jest sposób generowania interfejsu użytkownika. W typowym podejściu po stronie aplikacji mobilnej mamy do czynienia z raz zdefinowanym przez programistę układzie oraz wyglądzie interfejsu graficznego. Wiąże się to bezpośredno ze sposobem pisania aplikacji, w którym programista ma do dyspozycji gotowe komponenty ui, które muszą zostać odpowiednio oprogramowane i wkompilowane w końcową wersję aplikacji mobilnej.

To powoduje, że jakiekolwiek zmiany w interfejsie są możliwe tylko i wyłącznie poprzez przebudowę aplikacji. A co za tym idzie także i ponowne rozprowadzenie tak zmienionego programu. Takie podejście generuje dodatkowe koszty wynikające z potrzeby zapewnienia dostarczenia aktualnej wersji aplikacji do wszystkich klientów posiadających jej starą wersję. Częściowo problem ten rozwiązuje podejście angażujące cinkiego klienta w postaci mobilnej przeglądarki internetowej. Zapewnia ono centralizację aplikacji przez co wdrożenie nowych wersji staje sie niemal automatyczne. Niestety rozwiązanie tego typu jest często niewystarczająco elastyczne oraz posiada słabo rozbudowany interfejs użytkownika. W niniejszej pracy wykorzystane zostało podejście pośrednie, będące czymś pomiędzy interfejsem generowanym przez przeglądarki internetowe, a pełnowartościowym interfejsem aplikacji mobilnych.

Dynamicznie generowany interfejs użytkownika

W odróżnieniu od typowej aplikacji dostępnej na urządzeniach mobilnych, nasza aplikacja nie posiada zdefiniowanego z góry interfejsu użytkownika. Za wyjątkiem głównego ekranu, wszystkie pozostałe są generowane na podstawie danych przesyłanych z serwera. System zachowuje się analogicznie do przeglądarki internetowej - redeneruje znaczniki interfejsu użytkowniku. W odróżnieniu od rozwiązań oprtych na przeglądarce mamy możliwość zmian w kodzie źródłowym aplikacji redenerującej, dzięki czemu w zależności od potrzeby możemy dostosowywać zachowanie programu do wymagań użytkownika. Nasza aplikacja nie jest ograniczona przez niemodyfikowalną przeglądarkę internetową.

Kuix

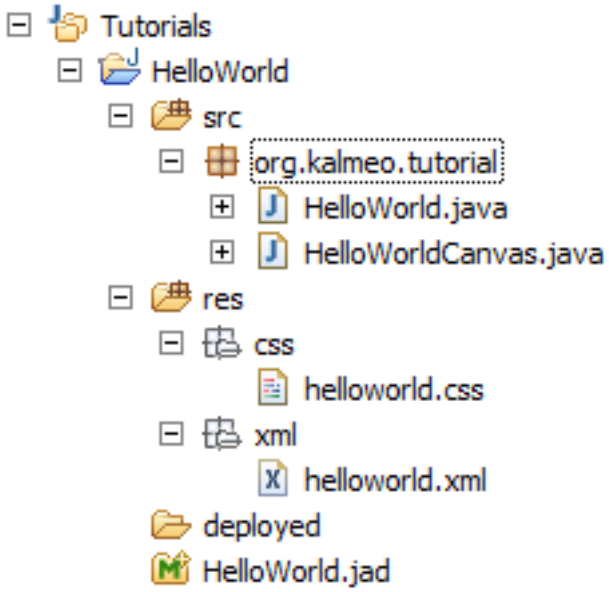
Opisany w poprzednim rozdziale dynamicznie generowany interfejs jest możliwy do osiągnięcia w środowisku j2me przy użyciu pewnego

danego z góry uniwersalnego sposobu opisu wyglądu poszczególnych elementów aplikacji. Ze względu na popularność XML oraz wsparcie dla tego języka za równo po stronie urządzeń jak i serwerów, można przyjąć, że właśnie w nim opisany zostanie interfejs użytkownika. Po stronie serwera, na podstawie informacji z bazy danych budowany będzie dokument XML łączący dane z ich sposobem wizualizacji. Po stronie urządzenia mobilnego dokument ten zostanie przetworzony przez parser XML, a następnie jego poszczególne elementy zostaną zmapowane na odpowiadające im elementy interfejsu j2me (odpowiednio rozszerzone o dodakowe komponenty, niedostępne w standardowej wersji tego środowiska). Opisana powyżej procedura została zaimplementowana w szkielecie programistycznym kuix.

Cechy szkieletu Kuix

Model programistyczny oferowany przez szkielet Kuix znacznie różni się od innych rozwiązań spotykanych na platformach mobilnych. Posiada on wiele cech charakterystycznych dla lekkiego modelu tworzenia oprogramowania mobilnego - opartego o mobilną przeglądarkę internetową. Najlepszą ilustracją zasady działania tego szkieletu, jest przykładowa aplikacja, więc w dalszej części pracy zostanie przedstawiony krótki przykładowy program obracujący podstawowe idee.

HelloKuix Na poniższym obrazku widoczną jest struktura bardzo prostgo programu posiadającego wszystkie cechy dostarczane przez szkielet Kuix.



Poza standardowymi plikami *.java zawierającymi kod aplikacji należy tu zwrócić uwagę na dodatkowe pliki zasobów :

— helloworld.css

— helloworld.xml

Plik helloworld.xml to dokument xml we wspomnianym wcześniej formacie reprezentującym dane wraz ze sposobem ich wizualizacji.

```
<screen title="helloworld">
    <container style="layout:inlinelayout(false,fill); align: center"
        <text text="Hello World!" />
        <picture src="logo_community.png" />
    </container>
</screen>
```

Tag screen to pojedynczy ekran na urządzeniu mobilnym mogący posiadać jeden, bądź więcej tak zwanych widgetów - elementów interfejsu użytkownika. Powyższy przykład zawiera kontener (container), czyli pojemnik widgetów pozwalający na grupowanie ich w odrębne

układy. W pojemniku znajduje się tekst (tag text) oraz obazek (tag picture). Informacje o zawartości oraz cechach poszczególnych elementów przekazywane są za pomocą atrybutów.

Powyższy plik xml jest wczytywany na początku działania programu. Służy do tego kod widoczny poniżej.

```
// Load the content from the XML file with Kuix.loadScreen static met
Screen screen = Kuix.loadScreen("helloworld.xml", null);


// Set the application current screen
screen.setCurrent();
```

Jak widać z załączonego przykładu budowanie elementów interfejsu, oraz ich wczytywanie jest bardzo proste i pewnien sposób przypomina tworzenie interfejsu użytkownika dla stron wyświetlanych w przeglądarkach inernetowych.

CSS, a Kuix Charakterystycznym elementem zapożyczonym przez Kuix z technologii internetowych jest sposób nadawania pożądanego wyglądu elementom aplikacji. Służą do tego kaskadowe arkusze stylów (Cascading Style Sheets). W przykładzie z poprzedniego paragrafu plik helloworld.css wygląda następująco :

```
text {
    align: center;
    font-style: normal;
    color: #f19300;
}
screenTopbar text {
    color: white;
    padding: 1 2 1 2;
}
screenTopbar {
```

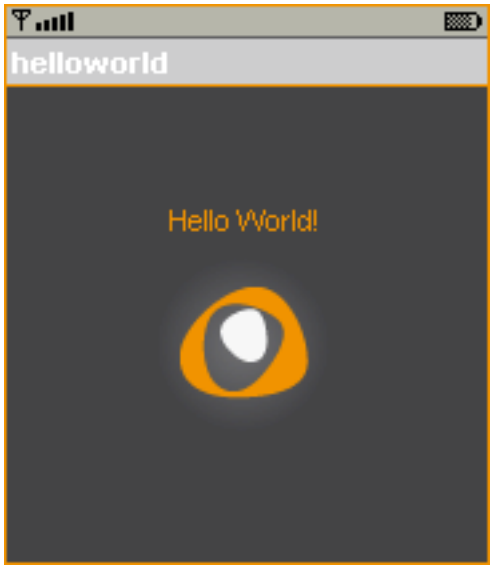
```
    font-style: bold;
    bg-color: #cccccc;
    border: 0 0 1 0;
    border-color: #f19300;
}
desktop {
    bg-color: #444447;
}
```

Powyższy kod css praktycznie niczym się nei różni od tego spotykanego w projektach pisanych pod standardowe przeglądarki internetowe. Do każdego elementu interfejsu, takiego jak ekran, czy pulpit (screen, desktop) mamy przypisane odpowiednie selektory. Na przykład by nadać styl paskowi tytułowemu ekranu używamy selektora screenTopbar.

By załadować do systemu plik css znajudjący się w zasobach projektu należy użyć statycznej metody loadCSS klasy Kuix:

```
// Load the stylesheet from the CSS-like file with Kuix.loadCss stati
//  note: a stylesheet is not associated with a screen but with the m
//  note 2 : by default '/css/' folder is use to find the 'helloworld
Kuix.loadCss("helloworld.css");
```

Końcowy efekt połączenia pliku xml ze stylami css widoczny jest na poniższym obrazku:



W analogiczny sposób można równie łatwo dodać podręczne menu do aplikacji :

```
<screenFirstMenu>Exit</screenFirstMenu>
<screenSecondMenu>
    more...
    <menuPopup>
        <menuItem>
            About
        </menuItem>
        <menuItem>
            Exit
        </menuItem>
    </menuPopup>
</screenSecondMenu>
```

Efekt końcowy wraz z menu dostępnym pod prawym przyciskiem (tak zwane Second Menu).

Problemy W chwili pisania niniejszej pracy środowisko Kuix było nowością na rynku. Najbardziej aktualna wersja 1.01 posiada nadal dużo wad i błędów. Z wykonanych przez nas testów wynika, że największe problemy związane są z pewnymi przekłamaniami graficznymi, które można spotkać na urządzeniach mobilnych firmy Nokia. Błędy te nie uniemożliwiały pracy, jedynie pozostawiały wrażenie ogólnego niedopracowania obecnej wersji szkieletu. Należy spodziewać się, że zostaną usunięte w następnych wersjach.

Kolejnym poważnym problem związanym ze szkieletem jest uboga dokumentacja techniczna. Dobrze i szczegółowo wykonana jest jedynie dokumentacja w formie javadoc oraz kursy wprowadzające do temetyki. Niestety na ich podstawie można opanować jedynie elementarne zasady posługiwania się środowiskiem. Jednak dzięki dostępności kodu źródłowego (Kuix jest w pełni otwarty) możliwe jest zapoznanie się z nieudokumenowanymi funkcjonalnościami oraz ewentualne wprowadzenie własnych poprawek.

Bardzo uciążliwym problemem jest rozmiar skompilowanych bibliotek środowiksa Kuix. Na chwilę obecną jest to około 250 kilobajtów. Wiele środowisk Javy spotykanych na platformach mobilnych posiada stałe nadane przez producenta ograniczenie rozmiaru pliku jar jak moża

załadować. Najczęściej ograniczenie to oscyluje w okolicach 100-150 kilobajtów przez co niemożliwe jest wykorzystanie Kuix na tych platformach. Problem ten dotyczy głównie rozwiązań konsumenckich, gdyż w rozwiązaniach moblnych dla biznesu ograniczenia są dużo wyższe lub możliwe jest dzielenie programu na biblioteki(taki rozwiązanie można znaleźć n platformie Blackberry).

3.3. Przykładowa implementacja: Mobilny system informacyjny

Przykładem integracji systemu klasy Enterprise z platformą mobilną jest aplikacja dostarczająca spersonalizowane informacje. Ideą stojącą za stworzeniem takiego systemu jest zagospodarowanie czasu użytkowników nie mogących aktywnie wyszukiwać interesujących ich wiadomości. Z takimi sytuacjami mamy do czynienia podczas oczekiwania na środki komunikacji miejskiej, a także w czasie utrudnień w ruchu samochodowym. W takim momencie można zaproponować lekturę najnowszych informacji w skróconej formie. Dodatkowo czytelnik w prosty sposób może określić, czy aktualnie pokazywane wiadomości interesują go czy też nie. W ten sposób uzyskamy możliwość analizy zainteresowań użytkowników, tak aby w przyszłości przekazywać im jedynie informacje które mogą ich zainteresować. Cała aplikacja oparta jest na przedstawionym wcześniej szablonie, do którego stworzony został interfejs administracyjny zgodny z charakterem systemu.

3.3.1. Założenia

Funkcjonalność systemu będzie oparta na następujących założeniach:

— Aplikacja na urządzeniu mobilnym wyświetla krótkie, dostosowane do użytkownika informacje

— Informacje zmieniają się automatycznie, tak aby ograniczyć potrzebę samodzielnego ich wyszukiwania
— Istnieje możliwość określenia czy aktualnie pokazywana wiadomość jest dla nas interesująca, co będzie miało wpływ na dobór kolejnych wiadomości

3.3.2. Wymagania systemu

Infrastruktura Potrzebne będzie połączenie z Internetem, dostępne z urządzenia mobilnego jak i z serwera dostarczającego informacje. Dodatkowo dla komfortowego przeglądania wiadomości, urządzenie mobilne powinno posiadać ekran o odpowiednio dużej rozdzielczości (co najmniej 320x240).

Oprogramowanie Urządzenie mobilne będzie musiało posiadać maszynę wirtualną Java. Aplikacja na serwerze będzie działała pod kontrolą kontenera Apache Tomcat.

3.3.3. Wstępna analiza projektu

Przypuszczamy, że system który planujemy stworzyć może znaleźć realne zastosowanie i odnieść sukces rynkowy.

	Zalety	Wady
Wewnętrzne	Wykorzystanie popularnej platformy Niski koszt rozwiązania	Powolna lub przerywana transmisja Zbyt mały ekran i mała ergonomia użytkowania
Zewnętrzne	Długi czas spędzany w korkach lub w środkach komunikacji w dużych miastach Podatność ludzi krótkie, podane w ciekawej formie informacje	Obawa ludzi przed korzystaniem z internetu mobilnego, jako ciągle zbyt drogiego

Podobne systemy Inspiracją, która stoi za naszym pomysłem jest system wyświetlania wiadomości na stacjach i w wagonach metra. Tym co chcielibyśmy w nim udoskonalić to możliwość dostarczenia spersonalizowanych, dostosowanych do każdego czytelnika informacji.

3.3.4. Interfejs mobilny

Mobilny klient naszej aplikacji jest midletem zbudowanym w oparciu o szkielet Kuix. Wersja instalacyjna składa się z dwóch plików :

— Archiwum jar zawierającego midlet

— Deskryptora jad zawierającego dane niezbędne do instalacji aplikacji w trybie OTA (Over-The-Air)

Menu główne Po instalacji aplikacji w zakładce zawierającej listę zainstalowanych na telefonie midletów powinna pojawić się nowa aplikacji o nazwie mINFO. Po jej uruchomieniu zostaniemy przeniesieni do głównego ekranu zawierającego trzy opcje :

— Show news - rozpoczyna ładowanie danych ze wskazanego serwera

— Settings - pozwala na skonfigurowanie źródła danych (serwera)

— Exit - kończy działanie midletu

Na poniższym obrazku widzimy główne menu aplikacji. Została ona uruchomiona w domyślnym środowisku symulacyjnym dostarczonym razem z pakietem Netbeans Mobility Pack.

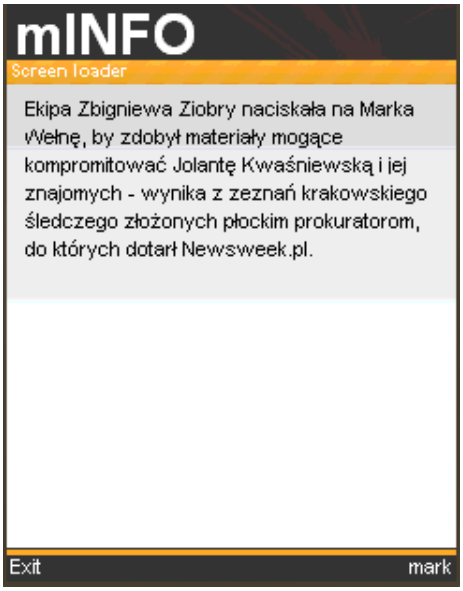
Zakładka Settings W zakładce settings mamy możliwość wybrania adresu, pod którym znajduje się serwer dostarczający dane dla naszej aplikacji. Aplikacja potrafi zapamiętać kilka takich adresów. By dodać nowy adres należy w polu 'New feed address' wpisać URL do nowego serwera, a następnie wybrać przycisk 'New feed'. Jeśli chcemy pozbyć się niepotrzebnych wpisów znajdujących się na liście, to zaznaczamy

wpis do usunięcia, a następnie wybieramy przycisk 'Remove feed'. Istnieje także możliwość modyfikacji adresu URL serwera. Służy do tego przycisk 'Edit feed'.

Zakładka 'Settings' po wprowadzeniu nowego adresu serwera:

Dynamicznie generowane ekrany wiadomości Po wybraniu z głównego menu opcji 'Show news' zostajemy zalogowani jako nowy użytkownik - pod warunkiem, że pierwszy raz podłączamy się do danego serwera. W przeciwnym wypadku z pamięci urządzenia zostaje

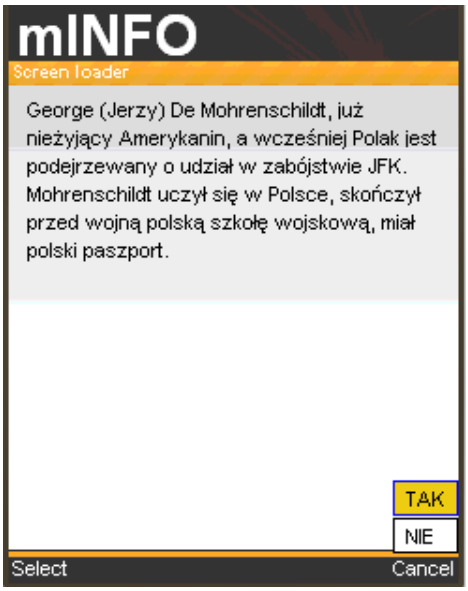
odczytana uprzednio wygenerowana nazwa użytkownika oraz hasło. Dzięki temu serwer jest w stanie śledzić działania wykonywane na danym urządzeniu przez użytkownika, poddawać je analizie, a następnie zmieniać zawartość wysyłanych treści. Ekrany z wiadomościami są całkowicie generowane po stronie serwera (nie tylko sama ich treść ale także elementy interfejsu użytkownika). Przykładowy ekran informacyjny :



Ekrany są automatycznie przeładowywane co pewien określony z góry przedział czasu. Dzięki temu jeśli użytkownik chce, to może biernie wykorzystywać naszą aplikację (w trybie nie wymagającym interakcji).

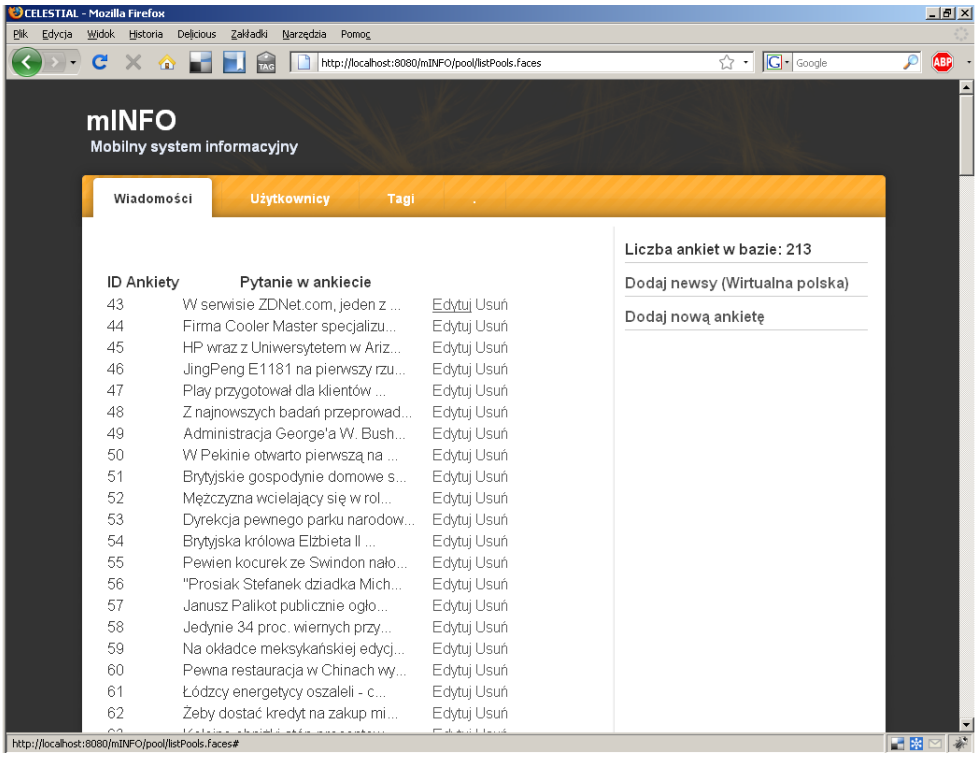
Wybór preferencji odnośnie treści wiadomości Jeśli użytkownik chce wykorzystywać aplikację w sposób interaktywny, to ma do dyspozycji podręczne menu określenia preferencji dotyczących aktualnie oglądanej wiadomości. Może poinformować system, że dana wiadomość go szczególnie zainteresowała, lub w przeciwnym wypadku, że nie chce więcej dostawać tego typu wiadomości. Na podstawie udzielonych odpowiedzi system personalizuje listę wiadomości, które

zostaną wysłane do danego użytkownika. Po wybraniu preferencji zostaje natychmiast załadowany ekran z następną wiadomością.



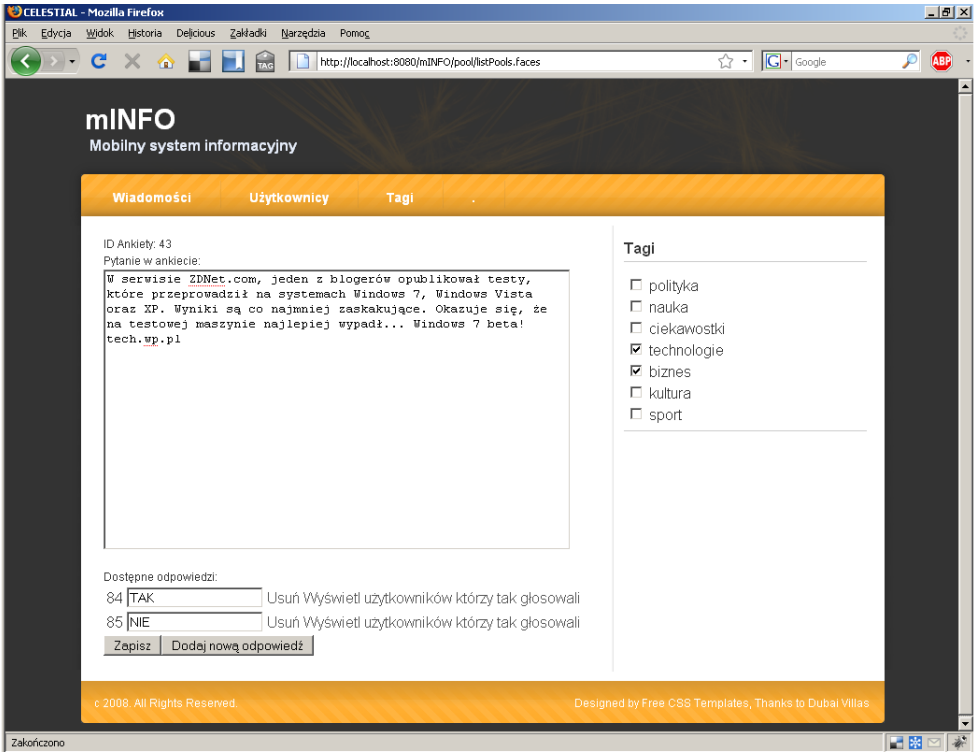
3.3.5. Interfejs administracyjny

Druga część aplikacja działa w kontenerze serwletów. Instalacja odbywa się poprzez umieszczenie przygotowanego archiwum w katalogu dla aplikacji. Po przejściu na stronę serwletu (domyślnie: http://serwer:port/mINFO) Na górze mamy dostęp do edycji wiadomości, użytkowników oraz 'tagów'. Po uruchomieniu widzimy listę wiadomości znajdujących się w systemie. Możemy dodawać wiadomości ręcznie, jak i pobrać je automatycznie z serwera Wirtualnej Polski (dodanego głownie dla testów systemu).



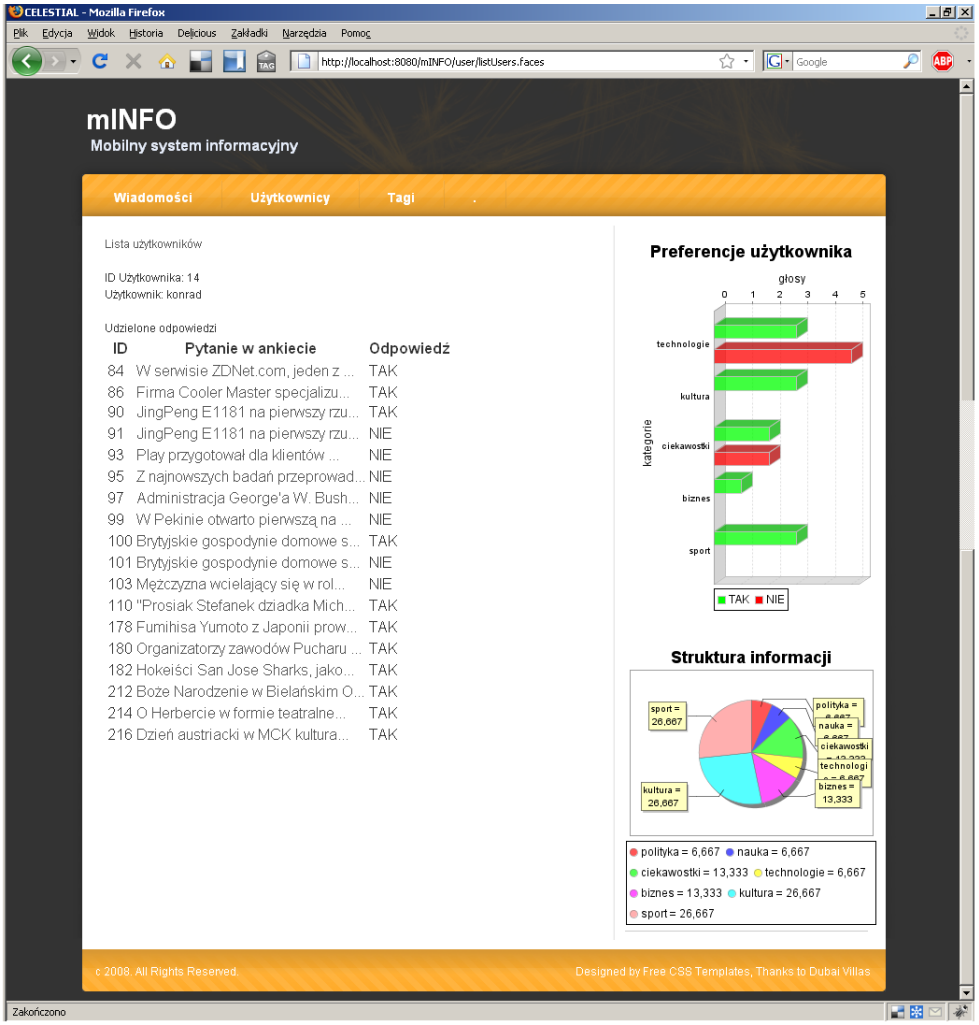
Lista wiadomości w systemie

Każda z wiadomości jest w rzeczywistości 'ankietą' o której wcześniej pisaliśmy w opisie szablonu integracyjnego. Dostępne odpowiedzi to TAK i NIE, określające czy dana wiadomość jest dla użytkownika interesująca (istnieje oczywiście możliwość dodania własnych odpowiedzi, jednak aplikacja którą stworzyliśmy, uwzględnia tylko te dwie w dalszym przetwarzaniu).



Edycja wiadomości

Dodatkowo, każdą wiadomość możemy 'otagować', to znaczy przypisać ją do pewnych kategorii tematycznych. Dzięki temu preferencje użytkownika zostają uogólnione na kategorie. Możemy też obejrzeć użytkowników, którzy głosowali w określony sposób w naszej ankiecie.



Ustawienia i preferencje informacyjne użytkownika

Na podstawie głosów użytkownika system tworzy jego profil informacyjny, aby serwować mu informacje coraz bardziej odpowiadające jego zainteresowaniom. Widzimy głosy, które oddał użytkownik na określone kategorie oraz bierzącą strukturę przygotowywanych dla niego informacji. System na tej podstawie selekcjonuje wiadomości oraz przygotowuje kolejne ekrany informacyjne dla klienta mobilnego.

4. Zakończenie

Bibliography

[1] Michael Juntao Yuan: *Enterprise J2ME Developing Mobile Applcations*, Prentice Hall Pennsylvania 2004

[2] Gregor Hophe,Bobby Woolf: *Enterprise Integration Patterns*, The Addison-Wesley Signaure Series 2003

[3] : *Spring Tutorial*, http://www.techfaq360.com/tutorial/spring/

[4] : *Which WSDL*, http://www.ibm.com/developerworks/webservices/library/ws-whichwsd