

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 zagadnienia związane z integracją systemów klasy enterprise i urządzeń mobilnych. Przedstawione zostały rodzaje integracji, problemy, które należy rozważyć oraz istniejące podejścia do tworzenia aplikacji mobilnych. W części implementacyjnej pracy zaproponowany został szablon integracyjny dla urządzeń mobilnych wraz z przykładową implementacją.

Słowa kluczowe: *integracja, urządzenia mobilne, enterprise, webserwisy, wiadomości, jms, kuix, blackberry, symbian, opera mini, internet explorer mobile*

Abstract

Title: *Mobile devices to enterprise-class systems integration*

This thesis presents issues connected with mobile devices to enterprise integration. Different kinds of integration have been shown, along with integration challenges that ought to be considered. Furthermore, various approaches of mobile applications development are described. Finally, in the implementation part, a sample integration framework and its appliance is presented.

Key words: *integration, mobile devices, enterprise, webservices, messages, jms, kuix, blackberry, symbian, opera mini, internet explorer mobile*

Spis treści

1. Integracja w środowiskach Enterprise	1
1.1. Potrzeba integracji	2
1.1.1. Pełnowartościowe środowiska informatyczne	3
1.1.2. Systemy mobilne	4
1.1.3. Miejsce systemów mobilnych w środowisku Enterprise	5
1.2. Klasyczne wyzwania integracji	6
1.2.1. Wyzwania techniczne	6
1.2.2. Wyzwania biznesowe	8
1.3. Problemy integracji mobilnej	9
1.3.1. Różnorodność platform	10
1.3.2. Wysokie koszty	11
1.3.3. Kanały komunikacyjne	12
1.3.4. Ograniczone zasoby	14
1.3.5. Bezpieczeństwo	16
1.3.6. Systemy off-line	18
1.4. Istniejące podejścia do integracji	19
1.4.1. Wybór technologii do integracji mobilnej	21
2. Metody realizacji aplikacji mobilnej zorientowanej na środowisko enterprise	29
2.1. Przeglądarka mobilna	31
2.1.1. Rodzaje przeglądarek mobilnych	31
2.1.2. Zagadnienia związane z projektowaniem stron	36
2.2. Generatory aplikacji	39
2.3. Rozwiązania dedykowane	46
3. Szablon komunikacyjny	48

3.1. Możliwości i zastosowania	49
3.2. Architektura	51
3.2.1. Aplikacja serwerowa	53
3.2.2. Aplikacja mobilna	56
3.3. Przykładowa implementacja: Mobilny system informacyjny	63
3.3.1. Założenia	63
3.3.2. Wymagania systemu	64
3.3.3. Wstępna analiza projektu	64
3.3.4. Interfejs mobilny	65
4. Zakończenie	74
Bibliografia	75
Indeks	76

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, na przykład kontakty z klientami lub księgowość. Rzadko się jednak zdarza, by istniał jeden system, który potrafiłby spełnić wymagania klienta, a nawet jeśli istniał by taki program, to firma, z różnych przyczyn, może nie chcieć go wdrożyć. Tak więc, nawet w obrębie jednego przedsiębiorstwa często można spotkać się 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 niedawna mówiąc o integracji, mieliśmy na myśli 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 na nowe problemy, wynikające ze specyfiki środowiska mobilnego.

1.1. Potrzeba integracji

Potrzeba integracji pojawia się, gdy występuje różnorodność, wymagająca 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
 - rozdzielenie 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, ale 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, przygotowanych pod platformy, które często nie mogą liczyć już na wsparcie producentów. Wielokrotnie

można natrafić na kombinacje tego typu aplikacji, które działają w kilku 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. Budowanie aplikacji, posiadającej wszystkie niezbędne funkcje aplikacji, która byłaby w stanie w pełni zaspokoić potrzeby dużego przedsiębiorstwa, stanowi duże wyzwanie. 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.

Następnie należy zwrócić uwagę na to, że rozwiązania, które składają się z wielu małych zintegrowanych podsystemów pozwalają na pewną elastyczność w doborze najlepszych rozwiązań w zakresie danej dziedziny. Może zostać wybrane takie rozwiązanie, które w danej chwili jest najlepsze i stosunkowo najtańsze na rynku. Nie wymusza to inwestowania w całościowe rozwiązania, które są 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. Potencjalnie, w przypadku jednego dużego systemu, 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, stanowi on jedną całość. Natomiast w rzeczywistości może składać się z wielu małych, zintegrowanych ze sobą systemów. 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 i autoryzacji użytkownika oraz sprawdzić obciążenie w celu wybrania optymalnego serwera 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 przechowywanych w nim informacji. Typowy użytkownik takiego systemu chce mieć możliwość połączenia się z nim o każdej porze dnia i nocy w celu pobrania konkretnych informacji. Wraz z pojawieniem się nowej generacji urządzeń mobilnych możliwym stało się zrealizowanie tej potrzeby. Nowoczesne platformy oferują niespotykaną wcześniej moc oraz zasięg, pozwalając tym samym 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. Integracja mobilna 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.

1.1.3. Miejsce systemów mobilnych w środowisku Enterprise

Tak jak zostało wspomniane w poprzednim rozdziale, głównym celem istnienia mobilnych systemów w środowisku enterprise jest umożliwienie uzyskania szybkiego i łatwego dostępu do potrzebnych w danej chwili informacji, znajdujących się w systemach bazodanowych firmy. Takie podejście sugeruje, że systemy mobilne są jedynie klientami, które udostępniają wygodny w użyciu interfejs dostępu do danych. Możemy jednak znaleźć liczne przykłady na to, że urządzenia mobilne w obecnych czasach są czymś więcej, ponieważ stają się pełnoprawnymi systemami, biorącymi czynny udział w wymianie danych. Potrafią nawet gromadzić i przetwarzać dane. Dobrym przykładem takich rozwiązań są systemy integrujące urządzenia GPS z danymi zawartymi w sieci korporacyjnej. Urządzenia GPS mogą istnieć jako niezależne środowiska, służące do nawigacji. Po wprowadzeniu integracji z systemem korporacyjnym uzyskujemy pewną wartość dodaną, która nie istnieje w żadnym z tych systemów oddzielnie. Możemy sobie na przykład wyobrazić system, który będzie wyświetlał na mapie GPS informacje o kontrahentach, znajdujących się w pobliżu osoby, będącej na stanowisku handlowca. Informacje te mogą zostać zaczerpnięte z bazy CRM firmy, dla której on pracuje. Co więcej aplikacja

taka może pozwolić na odsyłanie do bazy informacji, dotyczącej pozyskania nowego klienta oraz o powstaniu nowej relacji ze starym. I wszystko to może się odbyć natychmiast po zaistnieniu danego zdarzenia.

1.2. Klasyczne wyzwania integracji

Integracja aplikacji w przedsiębiorstwach często nie jest prostym zadaniem. Mimo, że w niemal każdej dużej firmie zachodzi potrzeba integracji i powstało już wiele opracowań na ten temat, nikt nie przedstawił kompletnej listy problemów, które należałoby rozważyć podczas integrowania systemów. Dodatkowo, wyzwania, które stawia integracja, wykraczają nie tylko poza techniczne rozważania, ale także rozciągają się na procedury biznesowe. Tak jak już wspominaliśmy, 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, 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, że problemy z komunikacją przy integracji danych znane są od wielu lat, duża część 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 wsparcia dla ich obsługi możemy oczekiwać jedynie w przypadku stosunkowo nowych aplikacji. 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 zdarza się, by była ona całkowicie scentralizowana i nie wymagała ingerencji lub chociaż konfiguracji integrowanych aplikacji. W związku z powyższym, osoby, które odpowiadają za utrzymanie takiej platformy, muszą posiadać wiedzę na ich temat. Dodatkowo, ciężko jest tę wiedzę utrzymać, zwłaszcza gdy pracownicy często się zmieniają. Natura platformy integracyjnej wymusza jej automatyzację, co powoduje, że może ona bezproblemowo działać bardzo długo, wręcz przezroczyście dla wszystkich aż do wystąpienia pierwszego problemu lub potrzeby wprowadzenia modyfikacji. Wtedy okazuje się, że osoby, od których wymaga się

tej wiedzy są tymi, które 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, a czasami bardzo nietypowych źródeł. Przykład może stanowić wypowiedź pewnego biznesmena, który przedstawiał sytuację w momencie, gdy pojawiło się zarządzenie, wymagające gromadzenia w jednym miejscu wszystkich informacji, związanych z rozwojem projektu. 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 funkcjonowania 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, gdyż inaczej może to doprowadzić do awarii wszystkich integrowanych systemów.

Różnice semantyczne w pojęciach. Choć XML rozwiązuje wiele problemów technicznych, to nie stanowi on odpowiedzi na różną semantykę w ramach obu systemów. Weźmy za przykład słowo "konto". W jednym systemie może oznaczać numer konta bankowego. Natomiast w innym 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 ze 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 oraz technologii .NET Microsoftu. Natomiast po stronie mobilnej mamy do wyboru wersje mobilne wymienionych platform oraz dodatkowo Blackberry, Symbiana, Google Android, Palm OS oraz kilku mniejszych dostawców. Oprócz tego, Apple promuje swój produkt 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, umożliwiających automatyzację często powtarzanych 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, co i tak będzie się wiązało z użyciem dodatkowych narzędzi, rozwiązujących dawno już zbadane problemy znane z klasycznych systemów.

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 języka JavaScript, na którym opiera się wiele stron internetowych. Istnieją oczywiście wersje stron, które zostały przygotowane dla platform mobilnych, ale wiele brakuje im do interaktywności oferowanej przez bogate i przypominające biurowe aplikacje strony internetowe spotykane we współczesnych przeglądarkach na systemach stacjonarnych. Nie oznacza to, że wykorzystanie przeglądarki mobilnej jest zawsze skazane na porażkę, ale, że

wiąże się z dodatkowymi ograniczeniami i może być stosowane tylko w przypadku prostych systemów.

1.3.2. Wysokie koszty

Koszty tworzenia aplikacji mobilnych (i wynikające z nich koszty integracji) różnią się w przypadku każdej z platform. Istnieją jednak czynniki, które podnoszą koszt w przypadku każdej z nich. Bez względu na metodologię tworzenia oprogramowania, możemy wyróżnić trzy fazy: analizę i projektowanie, programowanie oraz testowanie. Mogą one co prawda występować w różnej kolejności lub pojawiać się wielokrotnie, jednak żaden większy projekt informatyczny nie może się bez nich obejść. Jeżeli przyjrzymy się każdej z nich, będziemy mogli znaleźć potencjalne powody wyższych kosztów tworzenia aplikacji mobilnych.

- Analiza i projektowanie - duże, rozbudowane aplikacje mobilne, będące częścią informatycznego systemu korporacji są ciągle jeszcze rzadkością. Powoduje to, że osoby odpowiedzialne za zebranie wymagań, przeprowadzenie analizy i studium wykonalności mają mniej przykładów działających już systemów, na których mogą się oprzeć w swoich oszacowaniach.
- Programowanie - tworzenie aplikacji mobilnych wymaga najczęściej uwzględnienia wielu platform, na których mogą one działać. Podczas programowania najczęściej korzysta się z emulatorów, które jednak ostatecznie nie zawsze w pełni odpowiadają docelowym urządzeniom. Niekiedy musimy zaakceptować nieco niepoprawne działanie aplikacji na emulatorze, aby działała ona bez zarzutu na docelowym urządzeniu. W przypadku, gdy uruchamiamy aplikację na urządzeniu mobilnym, mamy ograniczone możliwości debugowania i logowania.

Warto też zwrócić uwagę na to, że społeczność programistów tworzących aplikacje mobilne jest niewielka i źle zorganizowana. Oznacza to, że szeroko stosowane postępowanie, polegające na poszukiwaniu podobnych problemów, z którymi spotkał się ktoś inny, nie zawsze

przynosi równie dobre rezultaty. Zwiększa to istotnie czas, który jest potrzebny na rozwiązanie problemu.

- Testowanie - w przypadku aplikacji mobilnych samo uruchomienie stanowiska testowego jest bardziej skomplikowane, czy to ze względu na emulator, czy docelowe urządzenie.

1.3.3. Kanały komunikacyjne

Możliwości połączenia z urządzeniami mobilnymi różnią się zasadniczo od tych, oferowanych przez klasyczne systemy. Z oferowanych przez urządzenia znajdujące się na rynku możemy wyróżnić:

- Bluetooth - protokół komunikacyjny pozwalający na łatwe łączenie urządzeń bezprzewodowych. Ma zasięg od 1 do 100 metrów, w zależności od wersji. jednak w najbardziej typowym przypadku jest to zasięg do około 10 metrów. Ze względu na dość niską prędkość (2.1Mbps w najszybszym, rzadko stosowanym wariancie) i ograniczoną mobilność, może być stosowany jako protokół synchronizacyjny, przy założeniu, że synchronizacja odbywa się w ustalonym miejscu dostępowym. Bluetooth jest obecnie dostępny w prawie wszystkich oferowanych telefonach komórkowych, smartfonach i PDA.
- Wifi - sieć bezprzewodowa, zgodna ze standardem 802.11a/b/g. Oferuje o wiele większą przepustowość niż Bluetooth (do 100Mbps), pozwalając na normalny dostęp do sieci komputerowej. Niestety, moduł Wifi nie jest powszechnie dostępny we wszystkich urządzeniach. Podobnie jak Bluetooth, pozwala na połączenie tylko w zasięgu punktu dostępowego.
- Sieć komórkowa - dostępna w przypadku wszystkich telefonów komórkowych i dużej części mobilnych organizatorów oraz palmtopów. Pozwala na dostęp z dość dobrą przepustowością w obrębie dużych miast (do 7,2Mbps w technologii HSDPA, przy teoretycznie oferowanej

14,4Mbps). Ze względu na najlepszą mobilność jest właściwie jedynym wyborem, gdy chcemy mieć możliwość nieprzerwanej komunikacji między urządzeniem mobilnym i siecią firmową.

- SMS - choć umieszczenie wiadomości tekstowych obok tak zaawansowanych protokołów, jak wymienione wcześniej, może się wydawać mało przekonujące, to musimy zwrócić uwagę na ich zaletę: działają nawet wtedy, gdy dostęp do sieci komórkowej jest mocno ograniczony. W oparciu o wiadomości tekstowe można przysyłać małe porcje informacji, które w wybranych zastosowaniach mogą się okazać wystarczające. Zostało to uwzględnione przez producentów oprogramowania, dzięki czemu z poziomu API możemy wysyłać i odbierać wiadomości tekstowe (na przykład na Windows Mobile), jak i korzystać z tej technologii w bardziej przezroczysty sposób, wykorzystując wiadomości, o których mowa w następnym rozdziale
- Stacje dokujące - wiele urządzeń mobilnych można podłączyć do komputera za pomocą kabla lub specjalnej stacji dokującej. Jest to, oczywiście, rozwiązanie mało wygodne. Jednak jego zaletą jest największa z pośród przedstawionych technologii prędkość. W związku z tym, że wiele urządzeń mobilnych posiada obecnie znaczną pamięć, sięgającą kilku gigabajtów, którą często można rozbudowywać za pomocą dodatkowych kart, można oczekiwać, że użytkownik będzie chciał pobrać dużą ilość danych drogą bezprzewodową, co może być bardzo czasochłonne. W tym momencie dobrym rozwiązaniem może być początkowe ładowanie danych do urządzenia za pomocą stacji dokującej i uzupełnianie ich w zależności od potrzeb np. przez Internet, dostęp do którego oferują nam operatorzy sieci komórkowych.

Problemem, który się pojawia podczas integrowania urządzeń mobilnych, jest konieczność dostosowania sposobów przysyłania danych do możliwości, oferowanych przez każdy z kanałów komunikacyjnych. Warto zwrócić uwagę, że żaden z nich nie oferuje niezawodności równej sieci komputerowej, opartej o tradycyjne media (poza stacją dokującą, jednak

nie jest to rozwiązanie, które pozwalałoby na prawdziwą mobilność).

Możemy dopuścić synchronizację przy użyciu każdej z dostępnych metod. Jednak to sieć komórkowa oferuje największą mobilność, wobec czego szcharakteryzujemy ją dokładniej. Projektując aplikację, która będzie się komunikowała za jej pomocą, musimy pamiętać o:

- Możliwych utratach połączenia - tak jak już mówiliśmy wcześniej, utraty połączenia są sytuacją, której możemy się spodziewać podczas komunikacji z urządzeniem mobilnym.
- Asymetryczności komunikacji - łącza komórkowe są najczęściej asymetryczne, co oznacza, że dane pobierane są wielokrotnie szybciej niż wysyłane (np. dla HSDPA przy prędkości pobierania 7,2Mbps wysyłanie odbywa się jedynie z prędkością 384kb/s). Oznacza to, że przesyłanie danych należy tak zorganizować, aby pozornie dość wydajne łącze nie stało się wąskim gardłem systemu.
- Kosztach połączenia - przesyłanie danych w sieciach komórkowych często jest obciążone opłatą zależną od ich ilości. Powoduje to, że należy znaleźć złoty środek między utrzymywaniem aktualności danych, a częstością ich pobierania. Może się bowiem okazać, że przy bardzo często przesyłanych, ale niewielkich paczkach danych, będziemy płacili głównie za przesyłanie sieciowych nagłówków i stałych elementów protokołu, z którego korzystamy. Sprawa nabiera jeszcze większego znaczenia, gdy korzystamy z Web serwisów, które wykorzystują XML, który jest formatem mało efektywnym. Z drugiej strony, przysyłając bardzo duże paczki danych narażamy się na konieczność powtórzenia całej transmisji w razie utraty połączenia.

1.3.4. Ograniczone zasoby

Urządzenia mobilne są zaprojektowane z uwzględnieniem kompromisu, który trzeba zawrzeć pomiędzy ich rozmiarami, wydajnością oraz czasem pracy na baterii. Różnicę widać już pomiędzy komputerem stacjonarnym, a laptopem. W tej samej cenie otrzymujemy laptopa, który jest

znacznie wolniejszy od zwykłego komputera, ale oferuje mobilność, której tamten nie posiada. Zasobami, o których trzeba pamiętać w przypadku urządzenia mobilnego są:

- **Możliwości przetwarzania.** Choć procesory, montowane w przenośnych urządzeniach kilka lat temu, z powodzeniem mogłyby być stosowane w komputerach stacjonarnych, to musimy pamiętać, że ciągle ich wydajność ustępuje temu do czego jesteśmy przyzwyczajeni w współcześnie używanych komputerach stacjonarnych. Powoduje to, że tworząc rozwiązanie integracyjne, powinniśmy brać pod uwagę, że niektóre operacje są wymagające pamięciowo (jak np. parsowanie XML-a i budowanie drzewa DOM), czy obliczeniowo (np. szyfrowanie).
- **Bateria.** Choć wiele urządzeń mobilnych umożliwia nawet 2-3 dniową pracę bez ładowania, to jednak nie możemy zapominać, że pewne operacje są znacznie bardziej energochłonne od pozostałych. Szczególnie obciążające jest przesyłanie danych - łączenie się z siecią WiFi, Bluetooth lub komórkową. I chociaż nawet wysokie koszty korzystania z tej ostatniej nie są dla nas przeszkodą, to już wysokie zużycie energii na pewno nią będzie. Oznacza to, że musimy projektować komunikację w taki sposób, by ograniczyć czas połączeń.
- **Ekran.** Chociaż nie jest to może problem ściśle związany z integracją, a raczej z tworzeniem aplikacji mobilnych, to warto zauważyć, że urządzenia mobilne dysponują o wiele mniejszym ekranem niż stacjonarne komputery. Oznacza to, że możemy na nim jednorazowo wyświetlić znacznie mniej informacji, co powoduje, że musimy je przedstawić w odpowiedni sposób.
- **Mała skalowalność.** Urządzenia mobilne rzadko dają się rozbudowywać (wyjątkiem są karty pamięci). Powoduje to, że tworząc aplikację mobilną musimy się ograniczyć do istniejących już zasobów. W przypadku, gdyby okazało się, że nasza aplikacja działa zbyt wolno na urządzeniach, działających w obrębie firmy, to użytkownicy będą mogli swoje telefony czy organizery wyposażyć w szybszy procesor czy

więcej pamięci operacyjnej. Filozofia tworzenia oprogramowania przypomina w tym momencie nieco tworzenie gier dla konsol - programista ma do dyspozycji sprzęt o znanych możliwościach i jego zadaniem jest wykorzystać je jak najlepiej.

1.3.5. Bezpieczeństwo

Bezpieczeństwo w urządzeniach mobilnych jest zagadnieniem bardzo szerokim i, niewątpliwie, mogłoby stanowić temat odrębnej pracy. Integracja urządzeń mobilnych z systemami klasy enterprise przynosi nowe zagrożenia, z których należy zdawać sobie sprawę. W celu ich dokładniejszej analizy, przedstawimy je w kolejności od najbardziej ogólnych do najbardziej szczegółowych.

Polityka firmy Większość dużych firm posiada zasady bezpieczeństwa, do przestrzegania których zobowiązani są jej pracownicy. Mogą one obejmować, między innymi, zakaz wynoszenia firmowych, poufnych danych poza miejsce pracy, stosowanie złożonych haseł, czy też zakaz instalacji własnego oprogramowania. Jednakże, wraz z integracją urządzeń mobilnych, zasady bezpieczeństwa muszą być przeanalizowane pod kątem ich stosowalności w nowym kontekście. Może się okazać, że zakaz wynoszenia danych straci sens z powodu zewnętrznego dostępu, który uzyskają pracownicy. Przedstawienie listy reguł, które należałoby dołączyć do stosowanej obecnie polityki, nie jest możliwe, bowiem w znacznym stopniu zależy ona od funkcjonalności, które chcemy udostępnić mobilnym użytkownikom, jak też od stosowanych w tym celu technologii. Ważna jest jednak świadomość potrzeby przeanalizowania istniejącej polityki pod tym kątem.

Czynnik ludzki Pamiętajmy, że nawet najlepsza polityka bezpieczeństwa nie przyniesie spodziewanych efektów, jeżeli nie będzie realizowana przez ludzi. Stosowanie najbardziej skomplikowanych haseł zakończy się niepowodzeniem, jeżeli będą one zapisane w widocznym miejscu. W związku

z tym, konieczne jest kształtowanie świadomości użytkowników, którzy będą mieli dostęp do systemu informatycznego z poziomu urządzenia mobilnego. Należy pamiętać, że każde z takich urządzeń staje się, potencjalnie, miejscem dostępu do poufnych, firmowych danych. Dodatkowo, w przypadku urządzeń mobilnych, istnieje większe ryzyko kradzieży, niż komputera stacjonarnego.

Warto zwrócić uwagę na fakt, że użytkownicy traktują bardziej osobiście urządzenia mobilne, takie jak smartfon czy PDA, niż firmowe komputery. Oprócz możliwości przechowywania na nich prywatnych danych, mogą one także instalować nieautoryzowane, potencjalnie niebezpieczne aplikacje. Jeżeli dodatkowo będą to urządzenia, pochodzące od różnych producentów, z których część będzie własnością pracowników, to wówczas, kontrola bezpieczeństwa staje się prawdziwym wyzwaniem.

Aplikacje W zależności od tego, czy integrujemy istniejące już aplikacje, czy też dopiero piszemy aplikację mobilną, musimy rozważyć bezpieczeństwo integracji z istniejącym już systemem. Istnieje zagrożenie, że integrowana aplikacja mobilna posiada luki w bezpieczeństwie (przykładowo zapis danych w jawnej postaci na lokalnym nośniku), które powodują, że dołączenie jej do działającego systemu informatycznego zwiększy ryzyko ujawnienia poufnych danych.

Systemy operacyjne Systemy operacyjne, stosowane w urządzeniach mobilnych, stają się coraz bardziej skomplikowane wraz ze wzrostem funkcjonalności, które należy obsłużyć. Pierwsze telefony służyły właściwie wyłącznie do wykonywania połączeń, dlatego też nie rozważano możliwości przenoszenia się pomiędzy nimi wirusów. Współczesne smartfony posiadają wielowątkowe systemy operacyjne, procesory, które są szybsze od tych, stosowanych parę lat temu w komputerach stacjonarnych, a dodatkowo, obsługują kilka sposobów komunikacji bezprzewodowej. W związku z tym, kwestią czasu było powstanie pierwszych wirusów, atakujących tego typu urządzenia. Poza złośliwymi programami, głównym

zadaniem których było unieruchomienie zainfekowanych telefonów (np. wirus Skulls dla systemu Symbian), o wiele większe zagrożenie stanowią będą wirusy, stworzone w celu wykradania danych z urządzenia mobilnego. W związku z powyższym, duże zróżnicowanie mobilnych platform, będących przeszkodą dla twórców aplikacji, utrudniających tworzenie uniwersalnych wirusów, może stanowić swego rodzaju pocieszenie. Warto jednak zwrócić uwagę na to zagrożenie, gdyż ataki na urządzenia mobilne nie są już tylko niegroźną ciekawostką, lecz faktem.

Łacza komunikacyjne Choć wybór sposobu komunikacji jest tematem rozdziału 1.4, to warto zwrócić uwagę na niższą warstwę komunikacyjną, która odpowiada za transmisję danych i nie jest bezpośrednio widoczna z poziomu aplikacji. O ile wszystkie, stosowane algorytmy szyfrowania danych zakładają transmisję po otwartym kanale komunikacyjnym, o tyle wybór odpowiedniej technologii, spośród przedstawionych wcześniej, może dodatkowo zwiększyć bezpieczeństwo. Tak, jak już wspominaliśmy, w związku z tym, iż wybór sieci komórkowej pozwala na największą mobilność, to należy dodać, że jest to najbezpieczniejszy sposób przesyłania danych. Warto również zwrócić uwagę na to, że możliwe jest uzyskanie bardzo dużego bezpieczeństwa poprzez wybór odpowiedniego dostawcy urządzeń mobilnych. Takim dostawcą jest na przykład firma Blackberry, która oferuje bardzo rozbudowany model bezpieczeństwa danych, do których mają dostęp użytkownicy mobilni (więcej informacji w rozdziale 2.2).

1.3.6. Systemy off-line

Głównym celem urządzeń mobilnych, zorientowanych na systemy enterprise, jest umożliwienie stałego dostępu do danych korporacyjnych. W związku tym, że nie w każdym miejscu mamy dostęp do sieci, to aby urządzenia te mogły spełniać swoje zadanie, ważnym jest istnienie możliwości tymczasowego przechowywania części potrzebnych danych w pamięci

trwałej. Pozwala to na dostęp do nich off-line. Niestety, funkcjonalność ta generuje bardzo dużo różnych problemów technicznych, związanych z synchronizacją. Co więcej, urządzenia mobilne mają bardzo ograniczone możliwości przechowywania danych w stosunku do normalnych systemów. Pojawia się więc problem wyboru. Musimy zautomatyzować proces podejmowania decyzji o tym, które dane kopiować na czas przebywania poza siecią. Jeśli dodatkowo użytkownik, nie mając połączenia z bazą, wprowadził do urządzenia jakieś nowe dane lub też zmodyfikował istniejące to pojawia się problem związany z zapewnieniem spójności danych znajdujących się na urządzeniu oraz w bazie. Synchronizacja tych danych stanowi duże wyzwanie. Często zdarza się, że nie jest możliwa pełna automatyzacja tego procesu. Dotyczy to głównie sytuacji, w których mamy do czynienia z dwustronną wymianą danych. W momencie gdy taka synchronizacja doprowadziła by do utraty danych, które zostały wprowadzone przez różnych użytkowników w tym samym okresie czasu, stosuje się replikację z konfliktami. Polega ona na tworzeniu dodatkowych wpisów w bazie danych, które informują o powstaniu konfliktu. Pozwalają one na dokonanie wyboru, o tym które dane chcemy pozostawić, a które są zbędne.

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. W momencie, gdy obie korzystają z tych samych danych, to nie ma potrzeby ich duplikacji. Dodatkowo, przez ograniczenia wymuszone przez bazę, musimy je zapisywać w ściśle określonym formacie, przestrzegając typów i więzów integralności. Niestety, 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 wprowadzenia 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

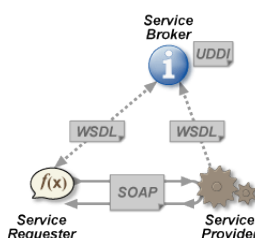
ją tak zaprojektować, aby przerwy w komunikacji były jak najmniej odczuwalne. Systemy komunikacyjne, takie jak RPC, RMI czy CORBA, projektowane były dla warunków, w których problemy komunikacyjne są sytuacją nadzwyczajną. Tymczasem, w przypadku łączności bezprzewodowej, utrata połączenia jest czymś, 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 (choć 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. Jednakże, nawet gdyby udało się stworzyć sterowniki, które pozwoliłyby na dostęp do bazy danych z poziomu urządzenia mobilnego, ponownie spotkalibyśmy się z problemem zawodnej komunikacji.

Okazuje się, że dwie, godne rozważenia metody stanowią, z jednej strony, wiadomości, a z drugiej - 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 powszechnych technologii internetowych, takich jak protokół HTTP czy język XML. Daje to nam gwarancję na to, 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 technologii można zobaczyć na poniższym rysunku.



Rysunek 1.1. Trzy podstawowe elementy związane z Web Services

Wspomniany wcześniej SOAP, czyli Simple Object Access Protocol, jest formą elektronicznej koperty, która zawiera podstawowe informacje o rzę-daniu oraz o odpowiedzi. Jest ona zbudowana w oparciu o język XML. Drugą kluczową technologią jest język opisu usługi - Web Service Description Language (WSDL). Z jego pomocą jesteśmy w stanie poinformować, jakie usługi oferuje nasz serwer oraz możemy wskazać dokładny sposób ich wywoływania. W wypadku urządzeń o ograniczonej mocy obliczenio-wej spotyka się go tylko przy automatycznym generowaniu kodu wywołu-jącego 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 zoptymalizowanej 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 wiadomości 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 generuje on dla całej organizacji. Z myślą o bezpiecznej komunikacji przy użyciu technologii Web Services, 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 nagłówek, związanych z szyfrowaniem danych. Wspiera także 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 są tu 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 maszynę Java. Pozostałe platformy, takie jak Blackberry czy WMD (Windows Mobile Device), również posiadają stosowne biblioteki, które pozwalają w bardzo prosty sposób podłączyć się do istniejącej infrastruktury. 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 dostarczenia, to jest to technologia, której użycie jest zalecane przy integracji mobilnej, czy też jakiegokolwiek 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 on samodzielnie rozwiązać. Jest to, przede wszystkim, brak gwarancji dostarczenia komunikatu. Dodatkowo, samodzielnie należy obsłużyć błędy transmisji, możliwe przerwy i konieczność ponownego nawiązania połączenia. nie implikuje to 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. Z jednej strony, 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 deklaracyjny 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. To, co musimy wziąć pod uwagę, to 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. Zwiększa on jednak 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. Warto zwrócić uwagę na 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 wiadomości tekstowych. 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. Wszystko zależy 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, wynikających 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 nieprzewidzianych 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 chwilę obecną (pierwszy kwartał 2009 roku) na rynku dostępne są urządzenia, działające w oparciu o 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 przez firmę Research in Motion, oferuje urządzenia z preinstalowanym 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, bardzo ryzykowne jest 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. Ciągłe jednak nie zapewniają one wygody porównywalnej z przeglądaniem stron na tradycyjnym komputerze. Jednakże 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 używana 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).

Uprozczone przeglądarki mobilne

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



Rysunek 2.1. Przykładowa strona wyświetlona w przeglądarce Opera Mini

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 przy próbie uzyskania dostępu do wrażliwych danych. Strach przed ich kradzieżą może być dużo większy niż korzyści, jakie można odnieść z używania tego programu.



Rysunek 2.2. Przykładowa strona wyświetlona w przeglądarce BlackBerry

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. Tunelowaniem zajmuje się działający w obrębie firmowej sieci serwer BES (Blackberry Enterprise Server). Jest on jedynym elementem wystawionym na zewnątrz i, poprzez szyfrowane

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, jedyną gwarancją bezpieczeństwa jest https, przy czym nie ma żadnego zapewnienia bezpieczeństwa danych, które przechodzą przez serwery proxy (tak, jak w wypadku przeglądarki Opera Mini). Tak więc w wypadku, gdy głównym czynnikiem decydującym o wyborze jest bezpieczeństwo, warto rozważyć zastosowanie rozwiązań, które je gwarantują.

Pełne przeglądarki mobilne



Rysunek 2.3. Przykładowa strona wyświetlona w przeglądarce Opera

Opera Mobile Opera Mobile jest jedną z najpopularniejszych pełnowartościowych przeglądarek mobilnych. Została zbudowana na bazie silnika przeglądarki stacjonarnej. Zachowuje prawie całkowitą zgodność z pełną wersją, co oznacza, że w środowisku opartym tylko i wyłącznie o tę markę, możliwe jest używanie stron internetowych bez żadnych zmian, przystosowujących je do wersji mobilnej. Niestety, ze względu na niewielką popularność przeglądarki stacjonarnej oraz jej całkowity brak zgodności z

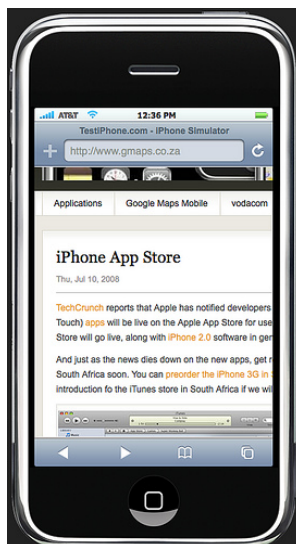
dwoma dominującymi przeglądarkami (wynikający z bardzo restrykcyjnego pilnowania standardów przez deweloperów Opery), zwykle niezbędne są zmiany w konstrukcji strony. Działa na urządzeniach korzystających z platform Symbian S60 oraz Windows Mobile.



Rysunek 2.4. Przykładowa strona wyświetlona w przeglądarce Internet Explorer Mobile

Internet Explorer Mobile W związku z tym, że 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 przeglądarki Internet Explorer. Może to zwiększyć nakład pracy niezbędny do uzyskania mobilnej wersji naszej strony.



Rysunek 2.5. Przykładowa strona wyświetlona w przeglądarce Safari Mobile

Safari Przeglądarka firmy Apple używana jest głównie w jej produktach. Jej mobilna wersja pojawiła się na rynku w momencie wprowadzenia telefonu iPhone. Charakteryzuje się bardzo dobrą obsługą języka JavaScript oraz pełną zgodnością z jej wersją stacjonarną. W odróżnieniu od produktów Opera Software na urządzeniach stacjonarnych firmy Apple, jest ona 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 przeglądarek 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. W związku z tym, że różnią się one głównie brakiem wsparcia dla niektórych elementów języka Javascript, to praca, związana z ich umobilnianiem, wiąże się głównie z uproszczeniem pewnych elementów lub przepisywaniem ich tak, by były kompatybilne. Zwykle trzeba zrezygnować też z pewnych zaawansowanych funkcjonalności, które działają w oparciu o technologię języka Javascript. Jeśli projektujemy portal od początku, to możemy w bardzo prosty sposób, przestrzegając kilku podstawowych reguł, umożliwić dostęp do niego z urządzeń mobilnych. Najważniejsze aspekty, które należy mieć na uwadze to:

- Należy pamiętać o budowaniu prostych i funkcjonalnych układów stron. Dotyczy to w szczególności form, które powinny mieścić się na jednym ekranie. Co więcej, należy wykorzystywać listy opcji tam, gdzie tylko się da, tak, by użytkownik musiał w minimalnym stopniu używać klawiatury.
- Nie można zakładać, że do nawigacji będzie wykorzystywana wirtualna myszka. Należy tak projektować strony, by można było je obsługiwać innymi rodzajami kontrolerów (na przykład przy pomocy technologii trackwheel). Wprowadzanie danych z klawiatury urządzenia mobilnego jest bardzo niewygodne, więc należy zadbać o to, by użytkownik nie musiał wprowadzać dużych ilości danych (wspomniane wcześniej listy opcji rozwiązują w pewnym stopniu ten problem)
- Urządzenia mobilne dysponują innymi, często znacznie mniejszymi rozdzielczościami niż systemy stacjonarne. Niektóre przeglądarki, takie jak Opera Mobile, oferują funkcje szybkiej nawigacji w obrębie dużych stron, dzięki czemu problem ten jest w pewien sposób ograniczany. Jeśli jednak chcemy dostarczyć najlepszą możliwą obsługę stron, to należy wykorzystać pewne elementy CSS, takie jak zapytania o typ urządzenia, aby dostosować stronę do konkretnej rozdzielczości.

Dobrym przykładem, jak ważne jest uwzględnienie tych rozbieżności, jest Opera Desktop oraz Opera Mini. Na rysunku 2.6 widać przykładową realizację strony z wykorzystaniem specjalnych stylów dla przeglądarki mobilnej.



Rysunek 2.6. Ta sama strona otwarta w Opera Desktop 9.5 oraz Opera Mini 4.1

— Należy również wziąć pod uwagę różnice pomiędzy przeglądarkami mobilnymi - w szczególności oferowanymi przez różnych producentów. Różnice w silnikach tych przeglądarek bywają tak duże, że często projektant jest zmuszony do przygotowywania różnych wersji tej samej strony dla różnych urządzeń mobilnych. Niezgodności te wynikają głównie z nieprzestrzegania standardów przez producentów oraz z wprowadzania do swoich rozwiązań dodatkowych funkcjonalności, niespotykanych na innych platformach. Dobrym przykładem takich

działań jest wprowadzenie w najnowszej wersji mobilnej przeglądarki Safari na urządzeniu iPhone możliwości budowania animacji poklatkowych przy pomocy samych stylów CSS. Jest to bardzo ciekawa i przydatna funkcjonalność, ale całkowicie nie przenośna na inne przeglądarki.

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 :

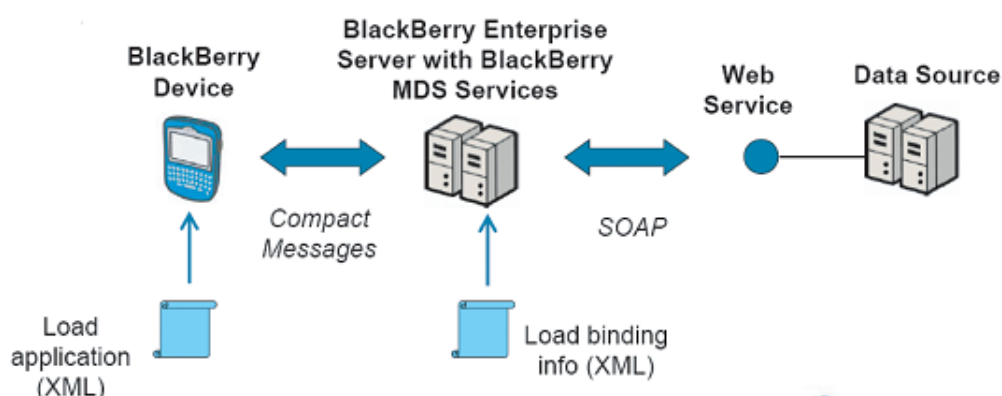
- Pozwalać na łatwe i szybkie tworzenie aplikacji dostępowych - tak jak mobilne przeglądarki
- Dostarczać elastyczności i rozbudowanych interfejsów użytkownika - tak jak aplikacje mobilne

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 użyciu standardowego drag

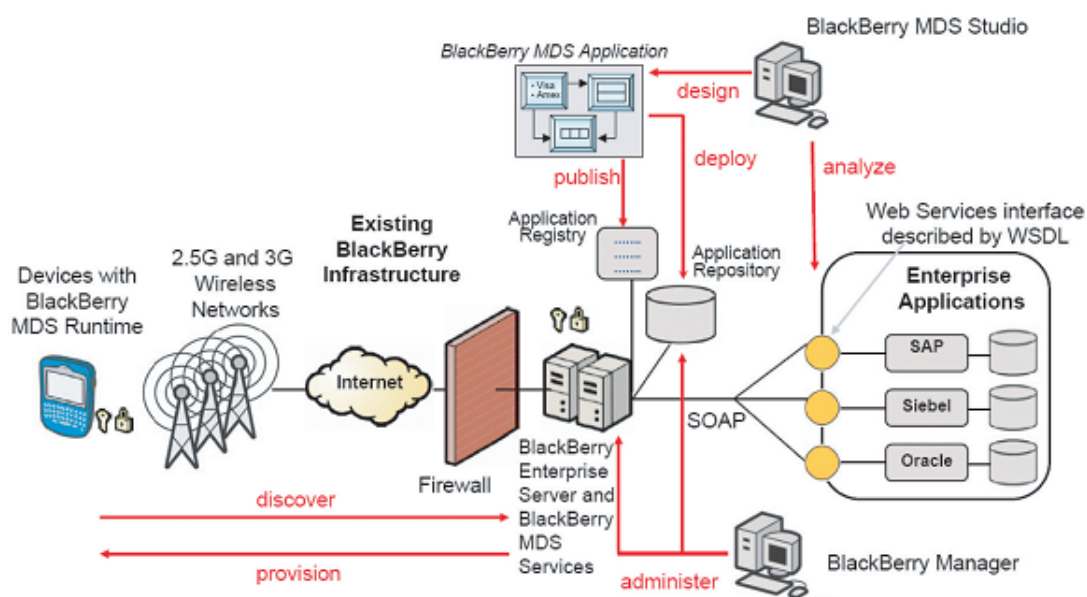
and drop. Następnie, możemy połączyć tak zbudowane komponenty przy użyciu różnych przewodników(wizards), czy też edytorów. Co więcej, aplikacja MDS daje nam pełną kontrolę nad przepływem sterowania - przy pomocy języka JavaScript jesteśmy w stanie zaimplementować niemal dowolną logikę aplikacji. Aplikacje typu enterprise oraz źródła danych dostępne są dla aplikacji mobilnej poprzez standardową technologię Web Services.



Rysunek 2.7. Architektura aplikacji opartej o BlackBerry MDS

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 scentralizowanej metodzie 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:



Rysunek 2.8. Sposób zarządzania aplikacjami opartymi o MDS

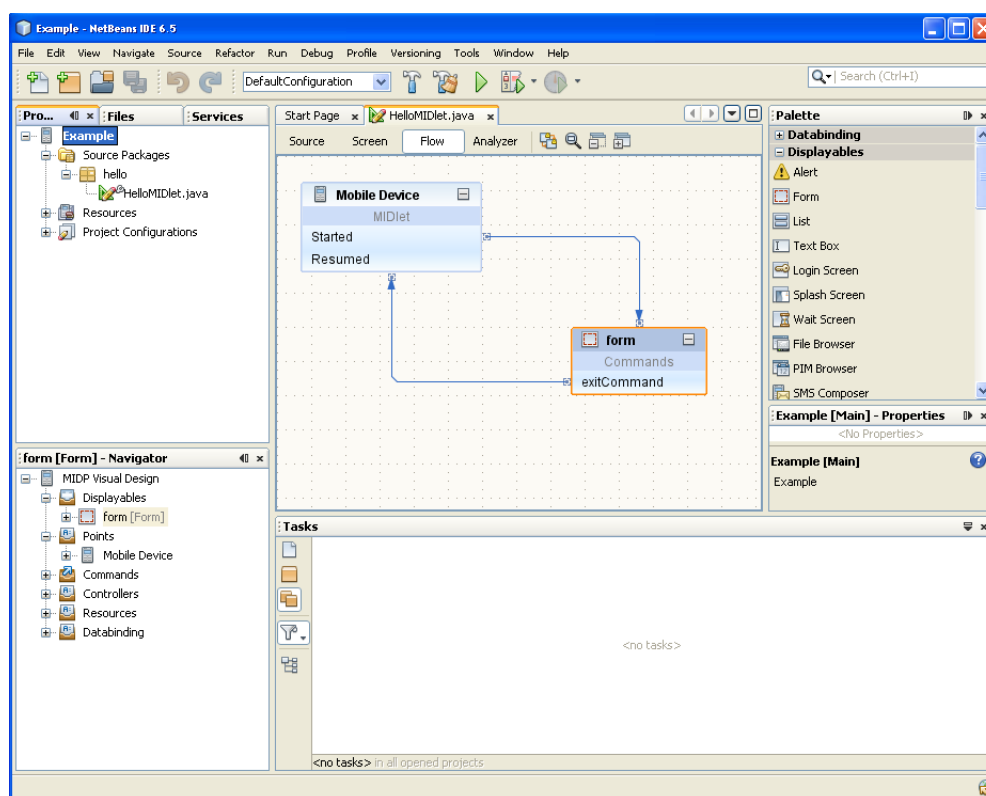
- Izolację platformy od szczegółów, związanych z integracją źródeł danych
- Standard asynchronicznej wymiany komunikatów
- Przezroczyste dla użytkownika i developera protokoły bezpieczeństwa
- Transformacje wiadomości
- Wspomnianą wcześniej scentralizowaną administrację przy pomocy technologii push

Blackberry MDS runtime Po stronie urządzenia mobilnego zainstalowany jest kontener aplikacji MDS. Zajmuje się on instalacją, wyszukiwaniem 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

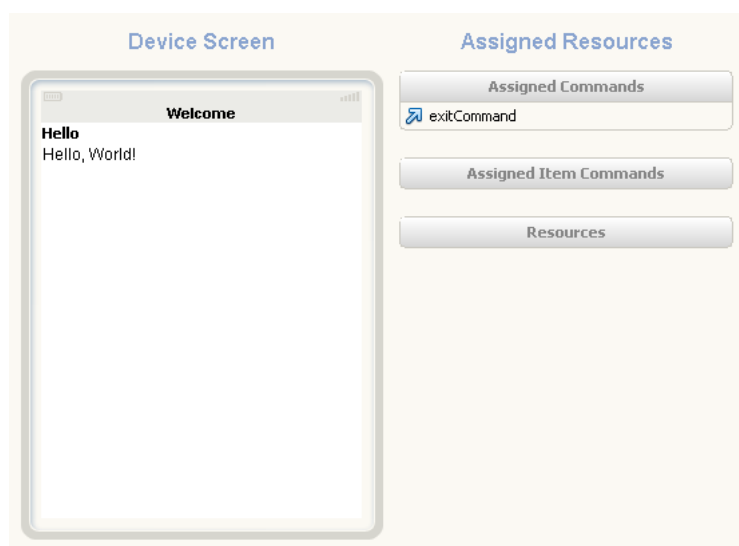
przeciwieństwie do poprzedniego przykładu, to środowisko pozwala budować uniwersalne aplikacje, które działają na niemalże dowolnej platformie mobilnej, wspierającej J2ME. Rozpoczynając tworzenie aplikacji, natychmiast natykamy się na udogodnienia przygotowane przez twórców Netbeans. Zamiast pisać setki 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.



Rysunek 2.9. Środowisko NetBeans z diagramem przepływu

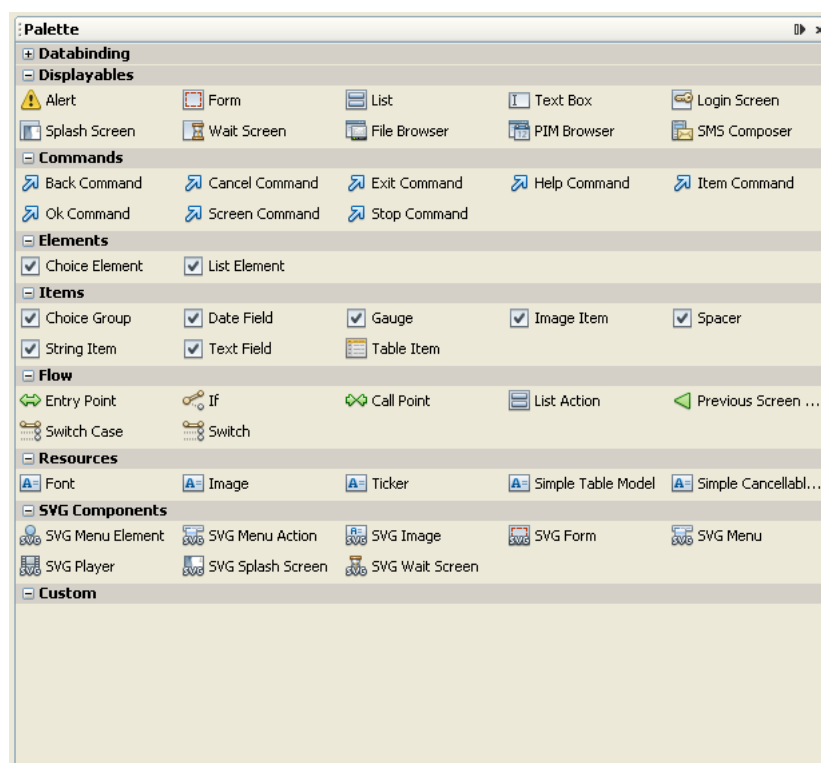
Przykładowy diagram przepływu składa się z dwóch elementów - bloku, reprezentującego stan wyjściowy aplikacji mobilnej (a jednocześnie stan końcowy) oraz bloku, reprezentującego główny i jedyny ekran.

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 drugą stronę, podpięta jest do komendy exitCommand. Wyzwala ona zamknięcie aplikacji (jej drugi koniec wskazuje blok wyjściowy). Ekran główny został zaprojektowany tak, by wyświetlić napis oraz komendę, pozwalającą na opuszczenie aplikacji.



Rysunek 2.10. Tworzenie interfejsu użytkownika w środowisku NetBeans

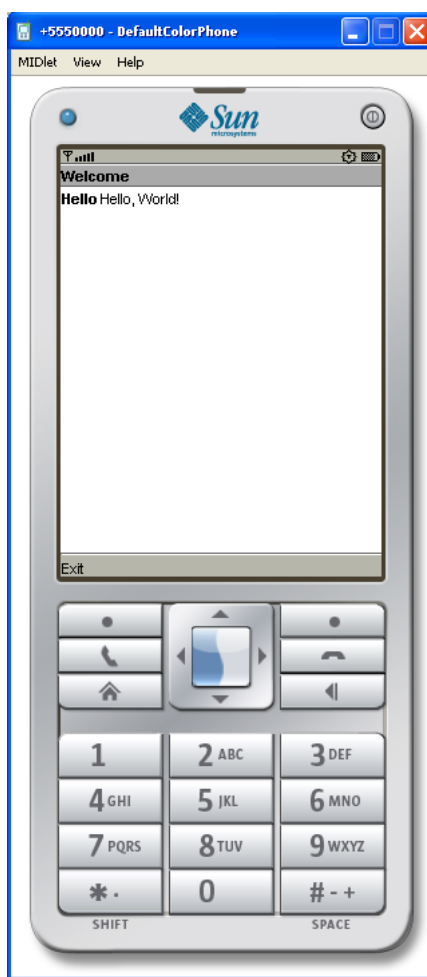
Tak jak wspomniano wcześniej tego typu ekrany buduje się z gotowych komponentów. Środowisko Netbeans udostępnia zestaw predefiniowanych komponentów o podstawowej funkcjonalności. Użytkownik może odnaleźć je na palecie bocznej programu. W wersji 6.5 paleta ta wygląda tak, jak na poniższym rysunku.



Rysunek 2.11. Paleta komponentów w środowisku NetBeans

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 języku Java, 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 językiem Java. Dostęp do kodu źródłowego automatycznie generowanej aplikacji pozwala na elastyczne modyfikowanie jej zachowania w

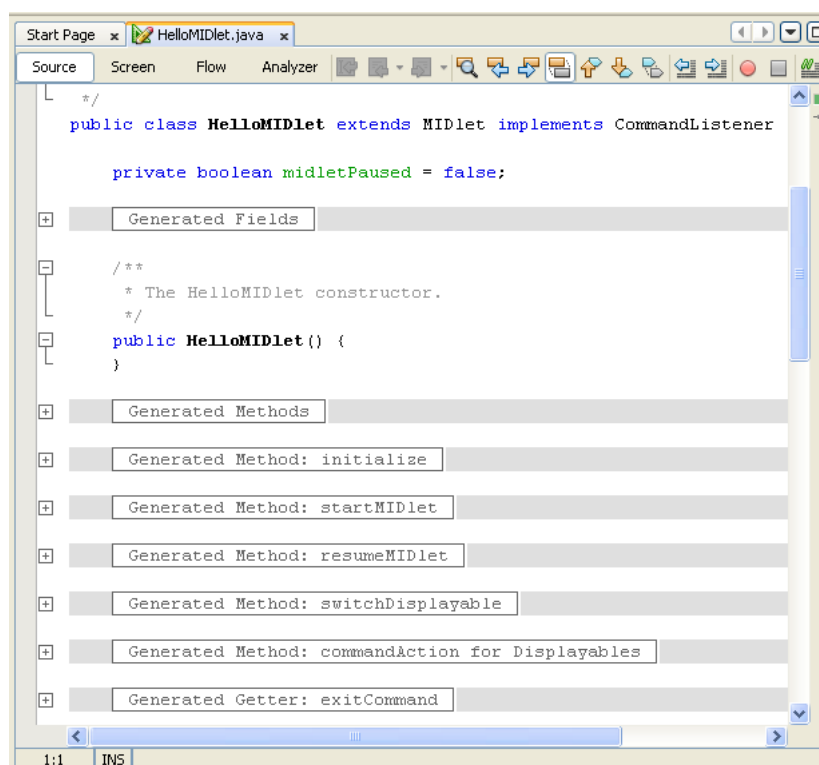


Rysunek 2.12. Emulator telefonu dostarczany przez Suna

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 wygenerowanym kodem w języku 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 w języku Java, który może kontrolować dowolne aspekty zachowania się aplikacji mobilnej.

Przedstawione środowisko programistyczne jest jednym z popularniejszych podejść do problemu szybkiego konstruowania aplikacji mobilnych.



Rysunek 2.13. Edycja kodu źródłowego w NetBeans

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, nawet jeśli byłyby 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, opisana tu sytuacja wskazuje na problem, który generuje stworzona z myślą 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

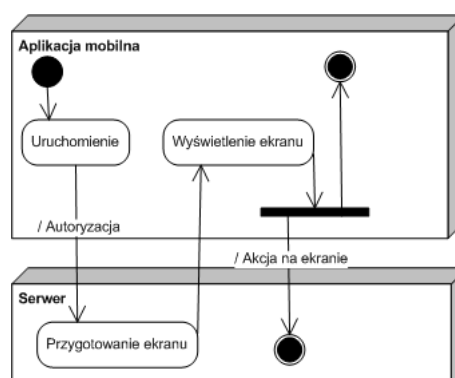
Do zademonstrowania możliwości komunikacyjnych pomiędzy systemem stacjonarnym i mobilnym stworzyliśmy szablon, ułatwiający tworzenie aplikacji, posiadających mobilny interfejs, z poziomu, z którego można sterować aplikacją serwerową. Rozwiązanie, które przyjęliśmy jest na tyle ogólne, że umożliwia rozwiązanie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



Rysunek 3.1. Diagram przepływu sterowania w szablonie integracyjnym

- 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. W związku z tym, że to właśnie serwer jest w całości odpowiedzialny za przygotowanie ekranu, to pojawia się możliwość kontrolowania ekranu i jego zachowania po stronie klienta w dowolny sposób.

3.1. Możliwości i zastosowania

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 byłoby 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 na komputerach stacjonarnych, na urządzenie mobilne. Nie zawsze jest to podejście słuszne - użytkownik, korzystający z urządzenia mobilnego - 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. Wobec tego 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.

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.

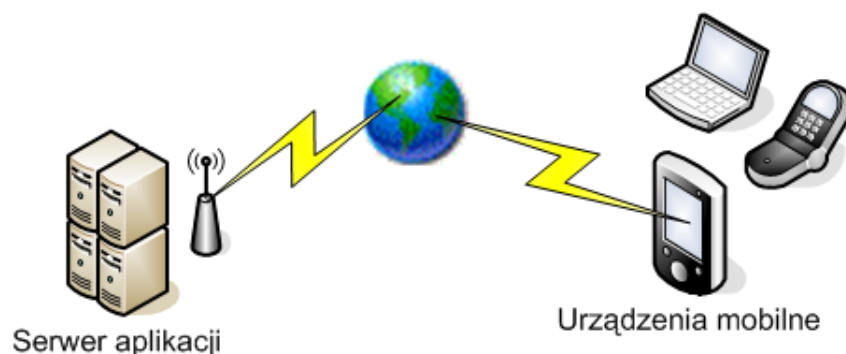
— Ankietywanie. Jednym z zastosowań, które możemy sobie wyobrazić, jest zdalne ankietywanie użytkowników. Stworzenie aplikacji, która

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. Interakcja 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.



Rysunek 3.2. Komunikacja między urządzeniami mobilnymi i systemem enterprise

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 zdejmuję 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.

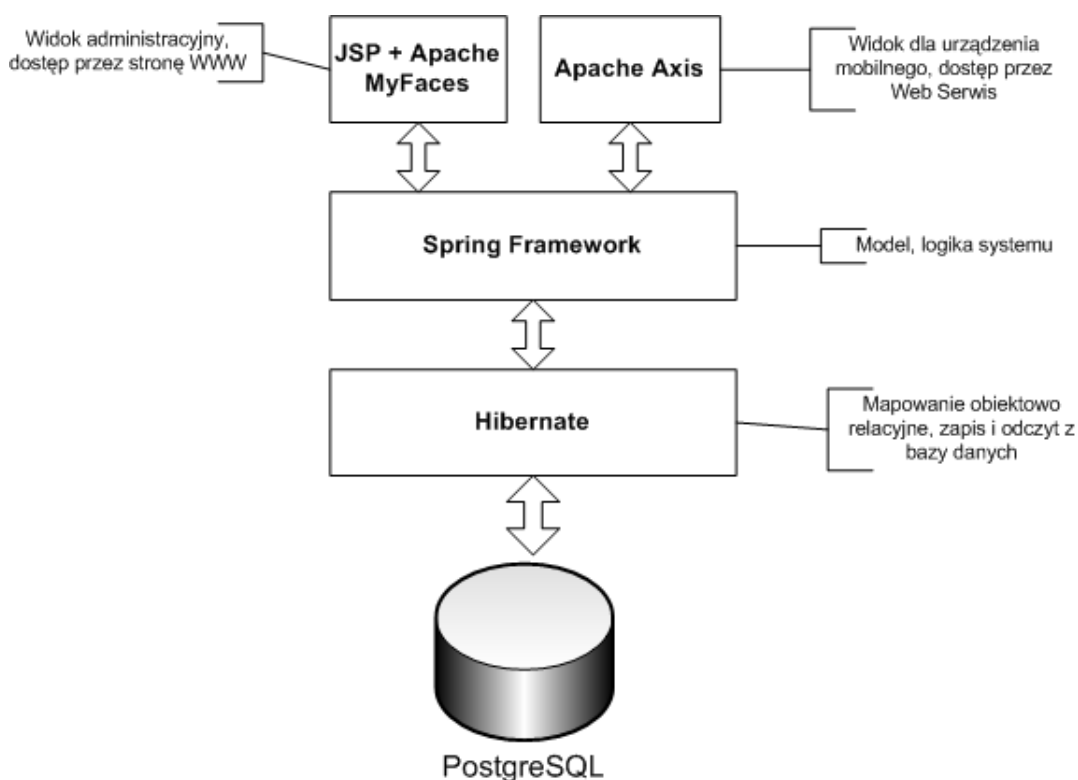
3.2.1. Aplikacja serwerowa

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.

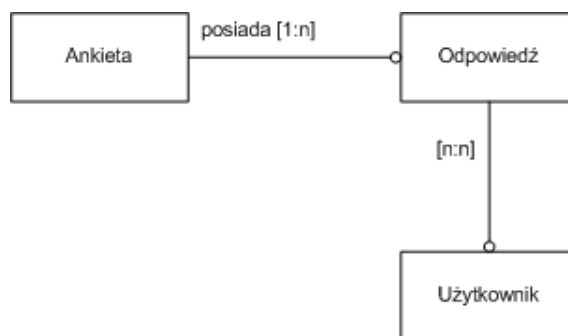


Rysunek 3.3. Architektura części serwerowej szablonu integracyjnego

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 przedstawione wcześniej założenia stworzony został prosty schemat bazy danych widoczny na rysunku 3.4.



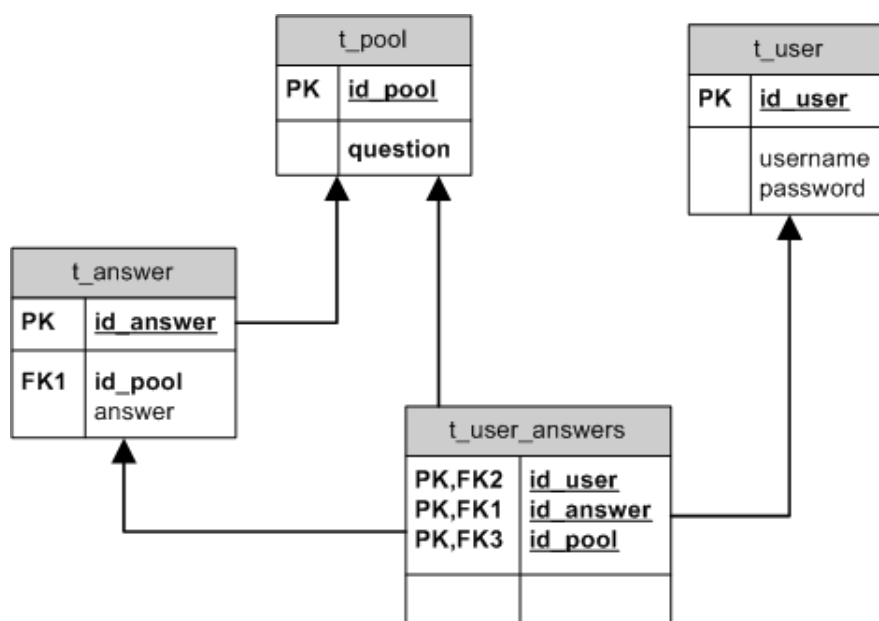
Rysunek 3.4. Schemat koncepcyjny bazy danych części serwerowej szablonu integracyjnego

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 podstawie schematu koncepcyjnego stworzony został schemat fizyczny bazy danych widoczny na rysunku 3.5.

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 cyklem życia obiektów oraz zwalnia programistę z konieczności pisania często powtarzanego kodu przez dostarczenie



Rysunek 3.5. Schemat fizyczny bazy danych części serwerowej szablonu integracyjnego

odpowiednich szablonów. Dodatkowo, wymusza on pewien model programowania i sprawia, że pisane aplikacje są bardziej przejrzyste i łatwiejsze w utrzymaniu. Jak już zostało 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 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 zdefiniowanym przez programistę układem oraz wyglądem interfejsu graficznego. Wiąże się to bezpośrednio ze sposobem pisania aplikacji, w którym programista ma do dyspozycji gotowe komponenty UI, które muszą zostać odpowiednio oprogramowane i wkompiłowane w końcową wersję aplikacji mobilnej. To powoduje, że jakiegokolwiek 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 cienkiego klienta w postaci mobilnej przeglądarki internetowej. Zapewnia ono centralizację aplikacji, przez co wdrożenie nowych wersji staje się 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 użytkownika. W odróżnieniu od rozwiązań opartych 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, zarówno 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 dodatkowe 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, obrazujący podstawowe idee.

HelloKuix Na rysunku 3.6 widoczna jest struktura bardzo prostego programu, posiadającego wszystkie cechy dostarczane przez szkielet Kuix.



Rysunek 3.6. Struktura prostego projektu wykorzystującego 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 method
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 w pewnym sposób przypomina tworzenie interfejsu użytkownika dla stron wyświetlanych w przeglądarkach internetowych.

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;
```

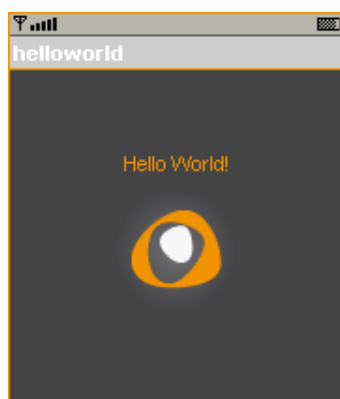
```
        font-style: normal;
        color: #f19300;
    }
    screenTopbar text {
        color: white;
        padding: 1 2 1 2;
    }
    screenTopbar {
        font-style: bold;
        bg-color: #cccccc;
        border: 0 0 1 0;
        border-color: #f19300;
    }
    desktop {
        bg-color: #444447;
    }
```

Powyższy kod CSS praktycznie niczym się nie 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, znajdujący się w zasobach projektu, należy użyć statycznej metody loadCSS klasy Kuix:

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

Końcowy efekt połączenia pliku XML ze stylami CSS widoczny jest na rysunku 3.7.



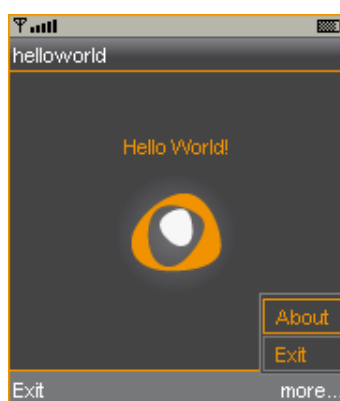
Rysunek 3.7. Wygląd omawianej aplikacji

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) widzimy na rysunku 3.8

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



Rysunek 3.8. Wygląd omawianej aplikacji

spotkać na urządzeniach mobilnych firmy Nokia. Błędy te nie utrudniały pracy, a jedynie pozostawiały wrażenie ogólnego niedopracowania obecnej wersji szkieletu. Należy spodziewać się, że zostaną usunięte w następnych wersjach.

Kolejnym poważnym problemem związanym ze szkieletem jest uboga dokumentacja techniczna. Dobrze i szczegółowo wykonana jest jedynie dokumentacja w formie javadoc oraz kursy, wprowadzające do tematyki. 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 nieudokumentowanymi funkcjonalnościami oraz ewentualne wprowadzenie własnych poprawek.

Bardzo uciążliwym problemem jest rozmiar skompilowanych bibliotek środowiska 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, jaki może 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 mobilnych dla biznesu ograniczenia są dużo wyższe lub możliwe

jest dzielenie programu na biblioteki (takie rozwiązanie można znaleźć na 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 na krótkie, podane w ciekawej formie informacje	Obawa ludzi przed korzystaniem z internetu mobilnego, jako ciągle zbyt drogiego

Podobne systemy Do stworzenia tego rozwiązania zainspirował nas 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 aplikacja 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 rysunku 3.9 widzimy główne menu aplikacji. Została ona uruchomiona w domyślnym środowisku symulacyjnym, dostarczonym razem z pakietem Netbeans Mobility Pack.

Zakładka Settings W zakładce settings (rysunek 3.10) 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'.

Zakładkę 'Settings' po wprowadzeniu nowego adresu serwera widzimy na rysunku 3.11.

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 widzimy na rysunku 3.12.

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 (rysunek 3.13, 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.

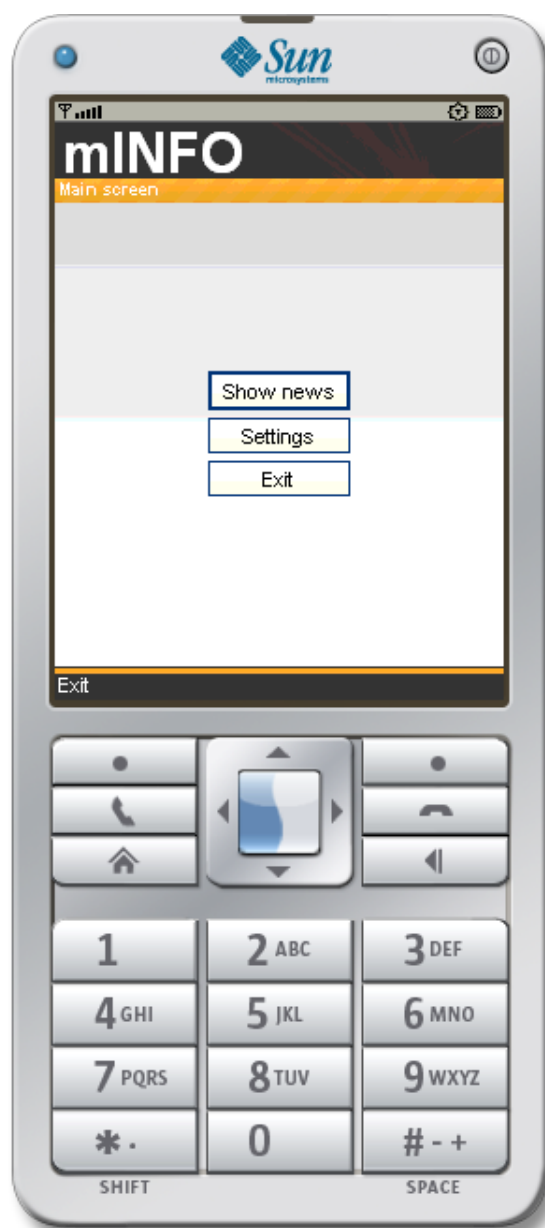
Druga część aplikacji działa w kontenerze serwletów. Instalacja odbywa się poprzez umieszczenie przygotowanego archiwum w katalogu dla aplikacji. Po przejściu na stronę serwletu (domyślnie: `http://serwer:port/mINFO`) Na górze mamy dostęp do edycji wiadomości, użytkowników oraz 'tagów'. Po uruchomieniu widzimy listę wiadomości (rysunek 3.14), znajdujących się w systemie. Możemy dodawać wiadomości ręcznie, jak i pobrać je automatycznie z serwera Wirtualnej Polski (dodanego głównie dla testów systemu).

Każda z wiadomości jest w rzeczywistości 'ankieta', o której wcześniej pisaliśmy w opisie szablonu integracyjnego. Dostępne odpowiedzi to TAK

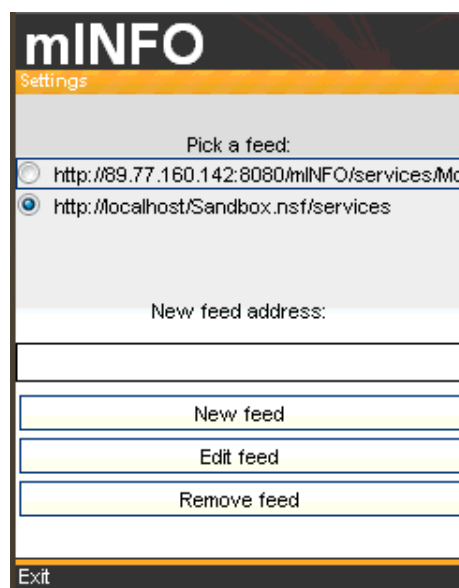
i NIE, określające, czy dana wiadomość jest dla użytkownika interesująca (istnieje oczywiście możliwość dodania własnych odpowiedzi, jednak aplikacja, którą stworzyliśmy, uwzględnia tylko te dwie w dalszym przetwarzaniu).

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.

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 bieżącą strukturę przygotowywanych dla niego informacji. System na tej podstawie selekcjonuje wiadomości oraz przygotowuje kolejne ekrany informacyjne dla klienta mobilnego.



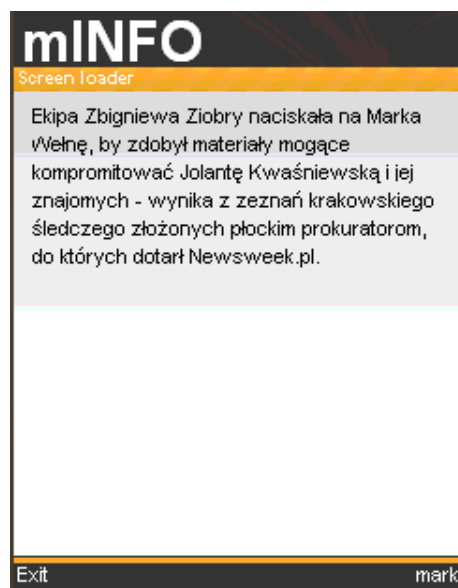
Rysunek 3.9. Ekran startowy aplikacji



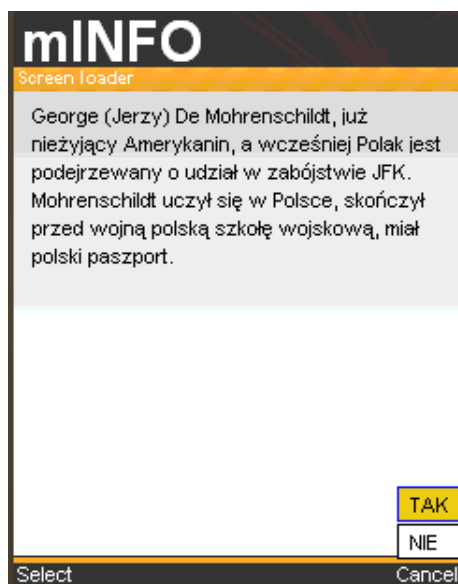
Rysunek 3.10. Ekran wyboru serwera wiadomości



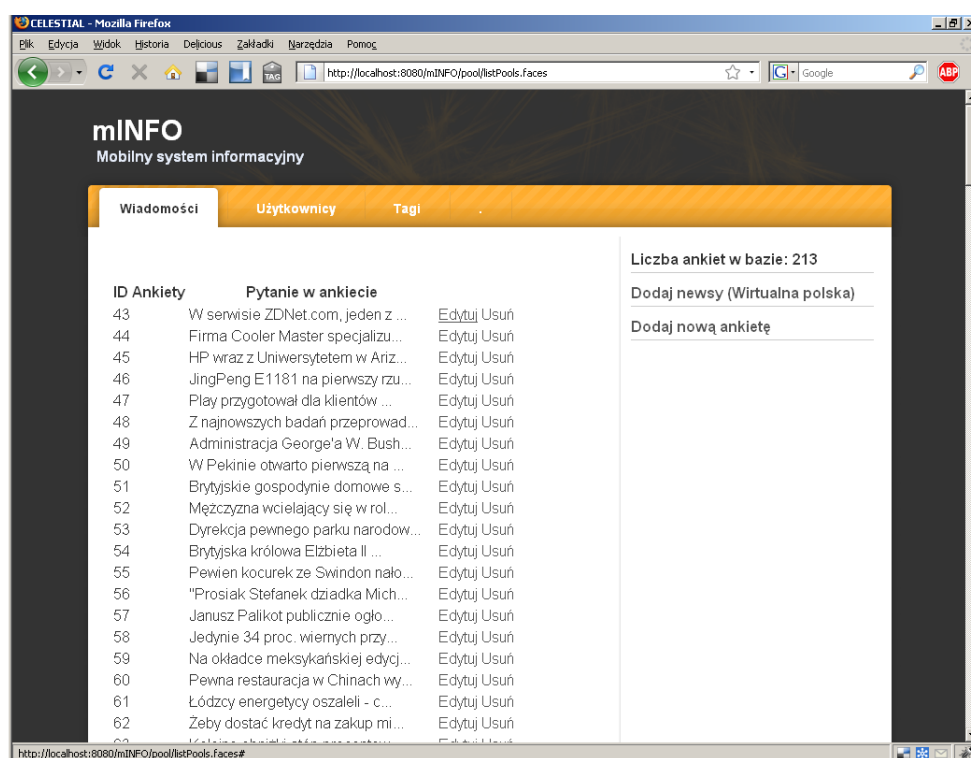
Rysunek 3.11. Ekran wyboru po dodaniu nowego serwera wiadomości



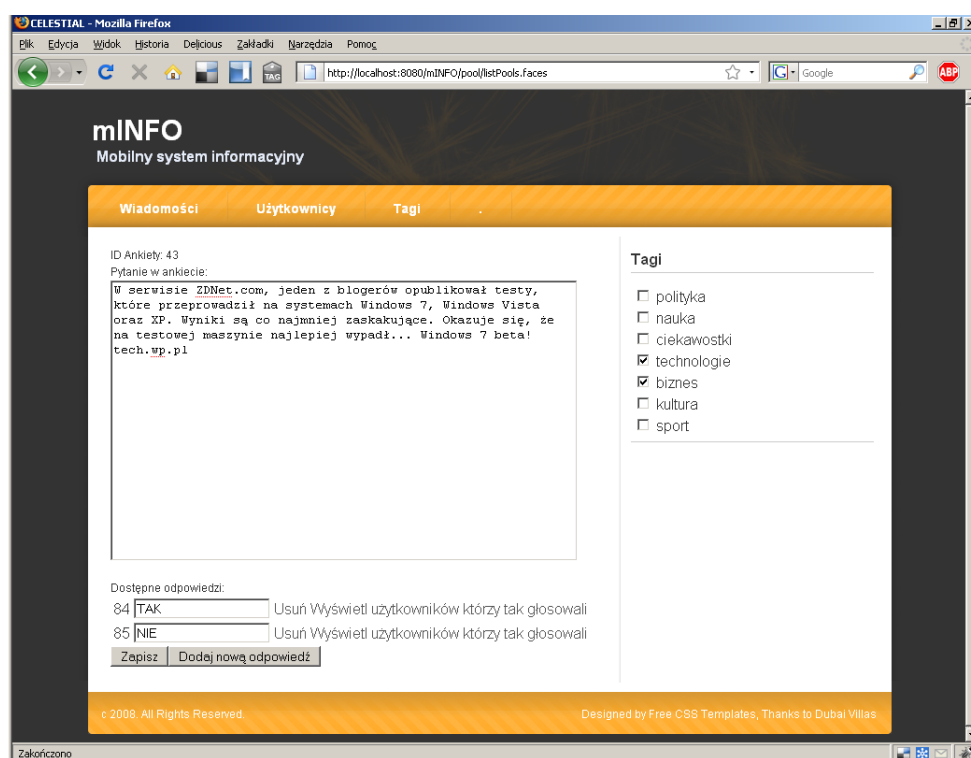
Rysunek 3.12. Ekran z wiadomością



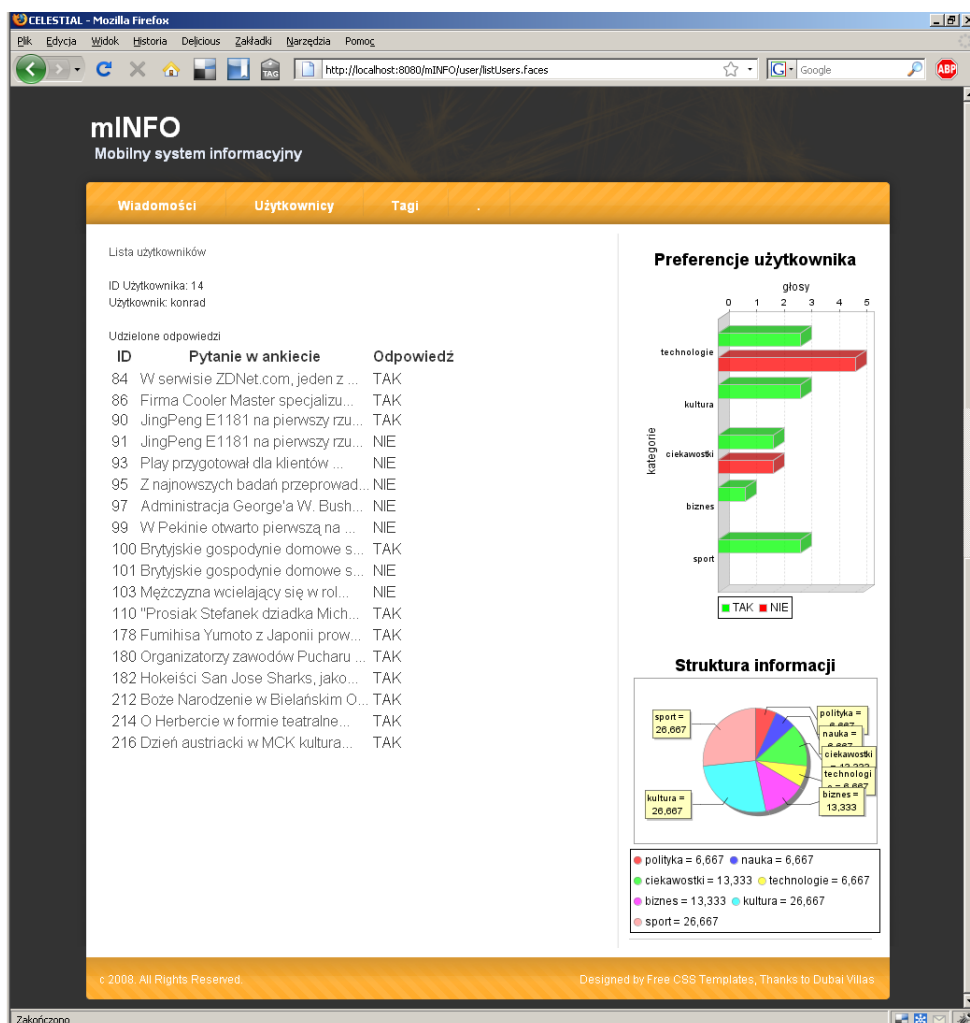
Rysunek 3.13. Określenie preferencji użytkownika



Rysunek 3.14. Lista wiadomości w systemie



Rysunek 3.15. Edycja wiadomości



Rysunek 3.16. Ustawienia i preferencje informacyjne użytkownika

4. Zakończenie

Rozwój urządzeń mobilnych oraz coraz szybszych łączów komunikacyjnych przyczynia się do powstawania możliwości uniezależnienia się od jednego miejsca pracy. Stale powiększa się grono osób, które doceniających możliwości oferowane przez organizery, PDA i smartfony. Próba przeniesienia funkcjonalności, wcześniej zarezerwowanych dla klasycznych komputerów na urządzenia mobilne, rodzi nowe wyzwania, które staraliśmy się opisać w naszej pracy.

Przedstawienie wszystkich zagadnień, związanych z tworzeniem i integracją mobilnych aplikacji, nie byłoby możliwe. W związku z tym, podjęliśmy próbę przedstawienia najbardziej typowych problemów, a także ich uznanych rozwiązań - sposobów tworzenia mobilnych aplikacji.

Różnorodność mobilnych platform rodzi konieczność skoncentrowania się na jednej z nich, aby być w stanie tworzyć na niej aplikacje. W naszej pracy zdecydowaliśmy się na wykorzystanie Javy w celu stworzenia szablonu integracyjnego, przedstawionego w trzecim rozdziale. Pragnęliśmy także zaprezentować nieco odmienne podejście do tworzenia aplikacji mobilnych, niż to, obecnie powszechnie stosowane. Przedstawiona idea może zostać wykorzystana na innych platformach, co otworzyłoby możliwości tworzenia bardziej interaktywnych, dostosowanych do urządzeń mobilnych, aplikacji.

Bibliografia

- [1] Michael Juntao Yuan: *Enterprise J2ME Developing Mobile Applications*, Prentice Hall Pennsylvania 2004
- [2] Gregor Hoppe, Bobby Woolf: *Enterprise Integration Patterns*, The Addison-Wesley Signature Series 2003
- [3] : *Spring Tutorial*, <http://www.techfaq360.com/tutorial/spring/>
- [4] : *Which WSDL*, <http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/>

Indeks

źródło danych, 8

analiza, 11

Apache Axis, 52

baza danych, 17

bezpieczeństwo, 16

Blackberry, 30, 36

Bluetooth, 12

CSS, 56

emulator, 11

Generatory aplikacji, 36

Hibernate, 51

integracja, 2

Internet Explorer Mobile, 32

JMS, 24

JSF, 52

koszty, 14

Kuix, 54

Netbeans, 38

off-line, 16

Opera Mini, 29

Opera Mobile, 31

platforma integracyjna, 7

pliki, 17

programowanie, 11

projektowanie, 11

projektowanie stron, 33

RPC, 17

Safari, 33

Sieć komórkowa, 12

SMS, 13

Spring, 51

Stacja dokująca, 13

Symbian, 10

testowanie, 12

trójwarstwowa architektura, 10

webserwisy, 20, 52

Wiadomości, 18

wiadomości, 22

Wifi, 12

XSLT, 52