Krzysztof Rączka[*], *Marcin Kowalski*[*], *Stanisław Gąsiorek*[**]
[*]*DHL Aviation - Bruksela*
[**]*Katedra Inżynierii Rolniczej i Informatyki*
Akademia Rolnicza w Krakowie

TECHNOLOGIA IMPLEMENTACJI SYSTEMU OBROTU I ZARZĄDZANIA NIERUCHOMOŚCIAMI

Streszczenie

Praca opisuje technologię implementacji systemu obrotu i zarządzania nieruchomościami. Poświecony jest on zarówno zagadnieniom teoretycznym jak i praktycznym aspektom budowy systemów wielowarstwowych ze szczególnym naciskiem na dynamikę oraz rozszerzalność projektu uzyskaną poprzez przejrzystą modularną strukturę, szerokie stosowanie rozwiązań konfiguracyjnych oraz wysoką parametryzowalność systemu. Ważnym celem projektowym jest również przenaszalność rozwiązania uzyskana poprzez stosowanie uznanych standardów.

Słowa kluczowe: nieruchomość, system informatyczny, architektura wielowarstwowa, diagram encji, techniczny schemat bazodanowy, XML, J2EE, servlet, JSP, JS

Zarys analizy

System dedykowany jest do wspomagania zarządzania oraz pośrednictwa obrotu nieruchomościami. Cel ten wyznacza trzy zasadnicze encje (twory koncepcyjne) określające dziedzinę zagadnienia a mianowicie: nieruchomość, klienta oraz transakcję. Są to pojęcia bazowe dla systemu, z których dalsza analiza wyłania w przypadku nieruchomości rozróżnienie na budynki i lokale oraz grunty. Dalsze uszczegóławianie w obrębie budynków i lokali wprowadza podział na budynki niemieszkalne i budynki mieszkalne, oraz wewnątrz klasy lokali podział na lokale niemieszkalne i mieszkalne. Dokładniejszy opis klienta wprowadza wyodrębnienie osoby sprzedającej oraz kupującej. Pojęcie to nie jest zbyt interesujące z koncepcyjnego punku widzenia. Ciekawa jest natomiast relacja pomiędzy nieruchomością a klientem, która opisana jest poprzez osobną encję transakcja. Jak zalecane jest w teorii analizy systemów, relacja modelowana powinna być osobną encją,

w przypadku kiedy posiada jakieś nadające jej identyfikowalności atrybuty, co w badanym przykładzie wydaje się naturalne, warto wspomnieć choćby o dacie lub wartości transakcji. Z punktu widzenia modelowania, transakcja ciekawa jest dodatkowo z powodu swojej dynamiki w czasie. Jest ona przykładem bytu, który, jest zmienny w czasie, zachowując jednak swoją unikalną identyfikowalność. Mówić można o różnych stanach transakcji, gdzie niektóre z nich, przykładowo rezerwacja mogą wydawać się początkowo innym tworem koncepcyjnym, bliższe przyjrzenie się doprowadza jednak do wniosku, że dla potrzeb rozpatrywanej aplikacji jest to identyczny koncept w innym stanie przetwarzania.

Przykład ten ilustruje złożoną strukturę obiektów w otaczającym nas świecie, którego modelowanie już na poziomie koncepcyjnym może być przeprowadzone stosując metody uszczegółowiania zidentyfikowanego wcześniej pojęcia, zamiast poprzez próbę poszukiwania nowych pozornie niezależnych konstrukcji. Warto tu zauważyć różnicę pomiędzy encją nieruchomość, której uszczegółowienie odpowiada klasycznemu przypadkowi modelowania relacji hierarchicznej, zgodnie z zasadą, że jest to ten sam obiekt (na pewnym poziomie szczegółowości budynek jest nieruchomością), a encją transakcja, która opisuje ten sam obiekt, w różnym stanie przetwarzania (w innym okresie czasowym rezerwacja jest transakcją). Różnicę stanowi tu więc aspekt momentu czasu w którym następuje opis, a nie szczegółowość postrzegania, choć w obu przypadkach relację można opisać sformułowaniem: obiekt X jest obiektem Y.

Celem tego rozdziału nie jest dokładne przyjrzenie się wszystkim pojęciom koncepcyjnym wyodrębnionym podczas fazy analizy rozwiązywanego zagadnienia, jest natomiast zwrócenie uwagi na możliwość przyjrzenia się niebanalnemu przykładowi modelowanej rzeczywistości używając zaledwie trzech podstawowych pojęć. Oczywiście analiza szczegółowa prowadzi do wyłonienia znacznie większej ilości encji, odzwierciedlających pożądany poziom szczegółowości, natomiast istota relacji pomiędzy nimi pozostaje podobna jak w modelu ramowym.

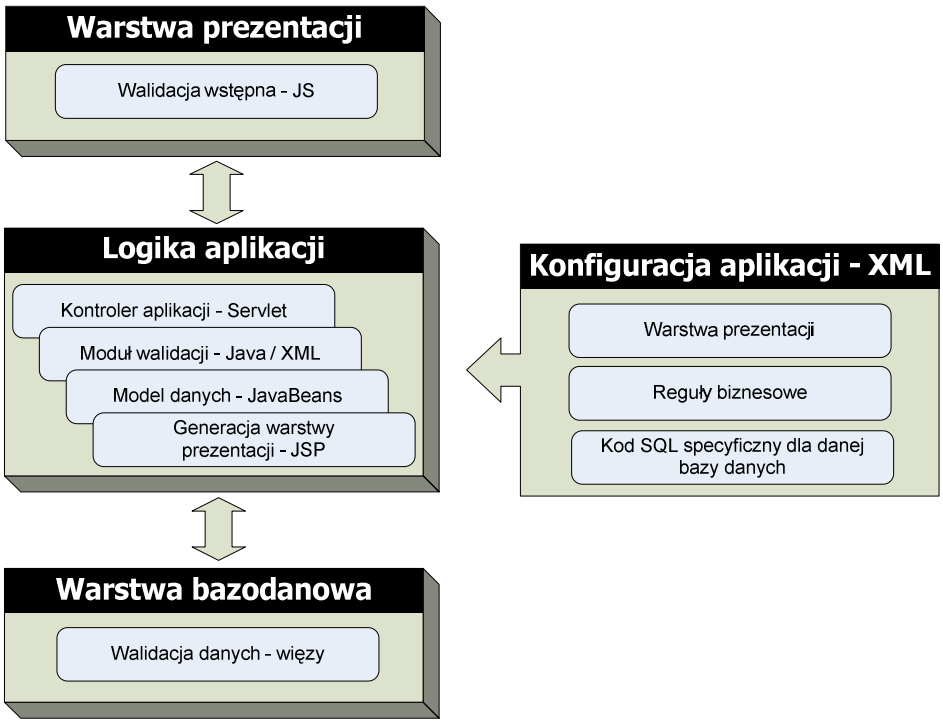
Architektura

Etapem projektu informatycznego następującym po fazie analizy systemu jest dobór stosownej architektury. Decyzja podjęta na tym etapie uzależniona jest od wymagań stawianych systemowi, oraz ogólnie zdefiniowanego sposobu użycia. W przypadku zagadnienia obrotu nieruchomościami naturalna wydaje się potrzeba dostępu do systemu poprzez użytkowników zlokalizowanych w różnorodnych lokalizacjach, oraz brak wiedzy o sprzęcie oraz systemach operacyjnych, z których

następuje żądanie realizacji usługi. Sytuacja taka niejako wymusza konieczność stosowania przeglądarki internetowej jako warstwy prezentacji systemu. Istnieją, co prawda inne możliwości uniwersalnej realizacji interfejsu użytkownika, jednak przeglądarki WWW z pewnością są najbardziej rozpowszechnionym medium dla przedstawionych powyżej założeń. Potrzeba dostarczenia dynamicznej treści wymusza z kolei stosowanie jednej z dostępnych technologii jej tworzenia. Obecnie jedynie dwie technologii pretendują do roli lidera w tej dziedzinie, są to rozwiązania oparte na języku Java oraz technologii Microsoft. Istnieje wiele opracowań starających się dowodzić wyższość jednego podejścia nad drugim, bezsporna jednak jest pełna przenaszalność rozwiązania serwerowego opartego na technologii Java, która zdecydowała o wyborze tego właśnie standardu. Rozpatrując warstwę najgłębszą, czyli składowanie danych, również przenaszalność była tu czynnikiem decydującym, powodującym wybór relacyjnej bazy danych jako technologii permanentnego składowania informacji.

Na rysunku 1 celowo nie podano nazwy konkretnej implementacji bazy danych stosowanej podczas fazy tworzenia systemu, gdyż całość odwołań odbywa się poprzez standard JDBC, który w połączeniu z językiem SQL zapewnia możliwość abstrakcji od konkretnego producenta systemu bazodanowego. Warto zaznaczyć, iż pomimo sięgającej już kilkunastu lat standaryzacji języka SQL nadal istnieją znaczące różnice pomiędzy dialektami tego języka wykorzystywanymi przez poszczególnych producentów, szczególnie widoczne podczas wykorzystania jego zaawansowanych aspektów, często niezbędnych do uzyskania pożądanej wydajności. Fakt ten spowodował zaprojektowanie mechanizmu umożliwiającego wydzielenie wszystkich zapytań SQL w postaci dedykowanego modułu, łatwego do podmiany lub modyfikacji w przypadku stosowania bazy danych o dużych rozbieżnościach od zalecanego standardu.

Istotną decyzją podjętą podczas prac nad architekturą systemu, było wprowadzenie dużej kontroli poprawności danych na każdym etapie przetwarzania, co odpowiada stosowaniu niezbędnej logiki niezależnie we wszystkich warstwach systemu. Dzięki stosowaniu takiego podejścia błędy wychwytywane są szybciej niż w przypadku stosowania rozwiązania klasycznego z badaniem poprawności jedynie w warstwie aplikacji lub w warstwie bazodanowej. Dodatkową zaletą jest możliwość poprawnej pracy niższej warstwy aplikacji w sposób wysoce niezależny od warstw wyższych, co ma szczególne znaczenie zwłaszcza w sytuacji konieczności zmiany odpowiednich warstw systemu.

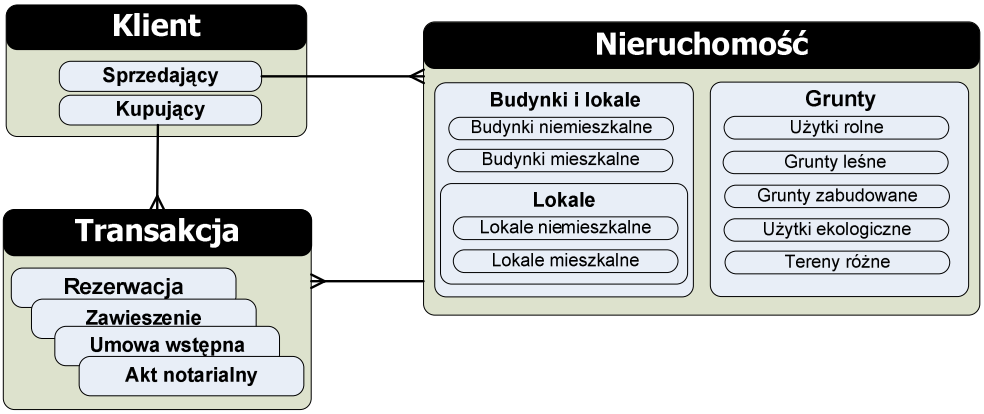


Rys. 1. Schemat architektury logicznej
Fig.1. Logical architecture diagram

Projekt bazy danych

Popularny dylemat, polegający na rozpoczęciu projektu systemu od warstwy aplikacji lub warstwy bazodanowej rozszczygnięto na korzyść bazy danych. Różne metodologie podają zalety wstępnego projektu obiektowego, uzupełnianego następnie o warstwę modelu danych, w przypadku opisywanego systemu zdecydowano się jednak na wybór metodologii bazującej na modelu bazodanowym dającym możliwość szybkiej weryfikacji poprawności poprzez implementację rozwiązań prototypowych. Metoda taka pozwala na szybsze rozpoczęcie fazy testów programistycznych, które ułatwiają eliminację błędów koncepcyjnych już na samym początku prac projektowych.

W celu reprezentacji modelu posłużono się diagramem encji, który w przejrzysty graficzny sposób obrazuje jego zawartość. Jak wspomniano już w zarysie analizy projekt rozpoczęto od definicji pojęć podstawowych dla systemu, został on następnie uszczegółowiony oraz rozszerzony o pojęcia referencyjne.



Rys. 2. Uproszczony diagram encji
Fig. 2. Simplified ERD diagram

Kolejnym etapem prac nad modelem bazodanowym było przekształcenie modelu encji w techniczny schemat bazodanowy, obrazujący już konkretną implementację tabel oraz zależności pomiędzy nimi. Zgodnie z założeniami zaprezentowanymi w przedstawionej architekturze systemu, schemat bazowy rozszerzono o więzy spójności danych zaimplementowane w warstwie bazodanowej. Typy reguł sprawdzane w ten sposób to: klucze podstawowe oraz unikalne, zależność referencyjna oraz więzy sprawdzające określone reguły poprawności dla każdego wiersza danej tabeli. Implementacja części logiki aplikacji bezpośrednio na bazie danych przybliża do interesującego dylematu określenia punktu rozgraniczenia pomiędzy pełnością logiki bazodanowej, a przenaszalnością rozwiązania pomiędzy bazami danych implementowanymi przez różnych producentów. O ile implementacja zarówno kluczy podstawowych, unikalnych oraz spójności referencyjnej jest bardzo naturalna dla każdej współczesnej bazy danych, to jednak w przypadku sprawdzania reguł bardziej zaawansowanych często zachodzi konieczność odwołania się do specyficznych rozszerzeń danej bazy danych, najczęściej w postaci języka proceduralnego rozszerzającego możliwości SQL, lub zaniechania realizacji tego typu reguł w warstwie najniższej. Zgodnie z nadrzędnym założeniem prezentowanej aplikacji o łatwości jej przenaszalności, ograniczono się do implementacji po stronie bazy danych jedynie reguł możliwych do zapisania w sposób zestandaryzowany i wykonywanie reszty sprawdzeń w warstwie logiki aplikacji.

Logika aplikacji

Warstwa logiki aplikacji wykonana została w technologii J2EE zapewniając jej bardzo wysoki stopień przenaszalności, oraz możliwość szybkiej realizacji systemu dzięki stosowaniu wielu standardowych usług dostarczanych razem z zastosowaną technologią.

Logika została zaprojektowana oraz zaimplementowana z uwzględnieniem założeń o klarownym rozdzieleniu modelu aplikacji, sterowania oraz tworzenia warstwy prezentacji. Dodatkowo dużą rolę położono na parametryzację różnorakich ustawień konfiguracyjnych, określających szczegóły pracy systemu, które wyeksponowane są poprzez technologię XML. Model aplikacji zbudowany został przy wykorzystaniu technologii Java Beans reprezentującej obiekty danych w warstwie logiki aplikacji. Podejście takie jest alternatywą architektury J2EE wykorzystującej technologię EJB (Enterprise Java Beans), która uznana została za nadmiarową, a tym samym niestosowną do rozwiązywanego problemu. Zastosowana technologia Java Beans, jest znacznie lżejsza z punktu widzenia przetwarzania, a obiekty w niej stworzone nadają się dobrze do przekazywania do warstwy budowy prezentacji. Wszystkie obiekty tego typu mają wbudowaną zdolność do zapisu oraz odczytu siebie samych z bazy danych, oraz zostały wyposażone w szereg metod określających wartości poszczególnych atrybutów logicznych. Szczegóły interakcji z warstwą bazy danych, zaimplementowane w języku SQL w całości składowane są w dedykowanym komponencie konfiguracyjnym, umożliwiającym jego łatwą modyfikację.

Sterowanie w warstwie logiki aplikacji zaimplementowane jest głównie przy użyciu technologii servlet, która ułatwia wykorzystanie języka Java do tworzenia dedykowanych serwisów, w tym konkretnym przypadku związanych z zarządzaniem logiką rozdzielenia oraz obsługi żądań przychodzących z warstwy prezentacji. Z kolei tworzenie warstwy prezentacji odbywa się stosując specyfikację JSP, opartą wewnętrznie na technologii servlet, dostosowaną jednak bardziej do szybkiego tworzenia interfejsu użytkownika, poprzez dostarczenie możliwości łączenia kodu HTML bezpośrednio z kodem Java, zapewniającym możliwość uzyskania dynamiki tak tworzonego rozwiązania.

Warstwa logiki aplikacji uzupełniona jest o moduł zarządzający oraz sprawdzający reguły zarówno techniczne jak i biznesowe, które muszą być spełnione przez dane w celu ich akceptacji. Moduł ten zapewnia szerokie możliwości sprawdzenia danych, a reguły w nim zawarte mogą częściowo być dublowane w logice zaimplementowanej w warstwie bazy danych. Nadmiarowość ta nie wpływa na pogorszenie efektywności aplikacji, gdyż w wielu przypadkach zapobiega niepotrzebnej,

kosztownej czasowo, komunikacji z bazą danych, która w przypadku danych naruszających reguły zakończyłaby się nieudaną próbą ich składowania. Wstępna walidacja danych ograniczona jest do reguł, których sprawdzanie w warstwie aplikacji jest naturalne, nie stara się natomiast implementować logiki typowej dla baz danych, takiej jak walidacja kluczy głównych lub unikalnych oraz zgodności referencyjnej, które są typowymi przykładami reguł, których wydajna walidacja na dużych zbiorach danych, wymaga wieloletnich doświadczeń w dziedzinie optymalizacji.

Warstwa prezentacji

Najbardziej zewnętrzną warstwą systemu, a zarazem jedyną, która dostępna jest bezpośrednio przez użytkownika końcowego jest warstwa prezentacji. Należy tutaj rozróżnić pomiędzy przygotowaniem warstwy prezentacji, która odbywa się w warstwie logiki aplikacji a samą warstwą prezentacji, która wykonywana jest na komputerze klienta, wewnątrz przeglądarki internetowej. Jak wspomniano w poprzednim rozdziale przygotowanie warstwy prezentacji odbywa się w technologii JSP, która umożliwia łatwe budowanie szablonów stron, które następnie, w momencie realizacji żądania dostępu do wybranej strony, zamieniane są na formę ostateczną i dostarczane są do komputera klienta w formie kodu HTML. Warto zauważyć, że nie jest to jedyna możliwa forma realizacji interfejsu użytkownika, innymi popularnymi metodami są dynamiczne generowanie kodu XML w warstwie logiki, a następnie lokalna transformacja tego formatu na formę HTML przy użyciu arkuszy stylu XLT, lub wysłanie postaci XML bezpośrednio do przeglądarki internetowej, która lokalnie zrealizuję transformatę na formę ostateczną. Każda z opisanych metod tworzenia interfejsu posiada pewne zalety i wady. Wybór najprostszej formy w postaci bezpośredniej generacji kodu HTML został dokonany ze względu na najszybszą do zaimplementowania możliwość realizacji wszystkich zamierzonych oczekiwań odnośnie interfejsu użytkownika.

Zgodnie z założeniem o realizacji kontroli jakości danych, na tak wczesnych etapach jak to możliwe, część walidacji danych realizowana jest bezpośrednio w warstwie prezentacji. Dynamika, oraz możliwość realizacji logiki wewnątrz przeglądarki internetowej uzyskana jest poprzez wykorzystanie języka JS (Java Script). Podobnie jak w przypadku interakcji pomiędzy warstwą logiki a warstwą bazy danych, również i tutaj granica walidacji wytyczona została w sposób praktyczny, a sprawdzeniu poddane są wszystkie reguły, które nie wymagają znajomości danych spoza zakresu lokalnego, czyli dostępnego bezpośrednio w bieżącej formatce interfejsu użytkownika.

Zagadnieniem związanym nierozerwalnie z warstwą prezentacji jest uwierzytelnienie oraz autoryzacja użytkowników systemu. Specyfika opisywanego systemu obrotu nieruchomościami wyróżnia część publiczną, dostępna do przeglądania oraz wyszukiwania użytkownikowi anonimowemu, oraz część prywatną, w której operator systemu dokonuje edycji zarówno dostępnych nieruchomości, ofert, transakcji, oraz pomocniczych elementów danych. Cześć publiczna dostępna jest bez stosowania mechanizmów zabezpieczających, natomiast część prywatna wprowadza zabezpieczenie zasobów zgodnie z zaleceniami specyfikacji J2EE.

Podsumowanie

System obsługi nieruchomości zaprojektowany jest w nowoczesnej, dynamicznej architekturze zapewniającej wysoką przenaszalność, modularność oraz łatwą rozbudowę. Wykorzystuje on zalety języka Java, starając się zarazem stosować wyłącznie niezbędne aspekty technologii J2EE, stosowne do rozwiązywanego zagadnienia, oraz nie wprowadzać niepotrzebnej nadmiarowości. Realizacja warstwy prezentacji wewnątrz przeglądarki internetowej, ma swoje zalety w postaci braku konieczności instalacji aplikacji po stronie klienta, wprowadza jednak pewne ograniczenia na możliwości graficzne, co jednak w przypadku omawianego systemu nie wpłynęło negatywnie na możliwości interfejsu użytkownika. System pozwala na zastosowanie praktycznie dowolnej relacyjnej bazy danych udostępniającej interfejs JDBC, poprzez pełne odseparowanie kodu SQL od rdzenia aplikacji, co zrealizowane jest w postaci niezależnego, wymiennego modułu konfiguracyjnego.

Bibliografia

Gruber M. 2000. SQL - wydanie drugie. Wydawnictwo Helion, Gliwice.

Hall M. 2002. Java Servlet i Java Servlet Pages. Wydawnictwo Helion, Gliwice.

http://jakarta.apache.org/tomcat/

http://java.sun.com/

http://java.sun.com/j2ee/

Hunter J., Crawford W. 2002. Java Servlet Programowanie. Wydawnictwo Helion i O`Reilly, Gliwice.

McLaughlin B. 2001. Java i XML. Wydawnictwo Helion i O`Reilly, Gliwice.

Roman E., Ambler S., Jewell T. 2002. Mastering Enterprise JavaBeans Second Editions.

Rozporządzenie Rady Ministrów z dnia 30 grudnia 1999 r. w sprawie Klasyfikacji Środków Trwałych (Dz. U. z dnia 31 grudnia 1999 r.) - na podstawie art. 40 ust. 2 ustawy z dnia 29 czerwca 1995 r. o statystyce publicznej.

Wiley Computer Publishing John Wiley & Sons, New York.

Wojciechowski M., Zakrzewisz M. 2002. Analiza porównawcza technologii tworzenia aplikacji internetowych dla baz danych Oracle. Politechnika Poznańska, Instytut Informatyki, Poznań.

ANALYSIS, ARCHITECTURE, TECHNICAL DESIGN AND IMPLEMENTATION TECHNOLOGY OF REAL ESTATE MANAGEMENT AND TURNOVER SYSTEM

Summary

The document describes architecture, analysis, technical design and implementation technology of the real estate management and turnover system. It is dedicated to both theoretical and practical aspects of the development of multilayer systems, putting an emphasis on the system's dynamics and extensibility, achieved by clear modular structure, wide usage of configuration mechanisms and high parameterization of the system. Another important design goal was to gain high portability of solution, achieved by usage of open standards.

Key words: real estate, computer system, multilayer architecture, server model diagram (SMD), entity relationship diagram (ERD), XML, J2EE, servlet, JSP, JS