STUDIA INFORMATICA 2013
Volume 34 Number 2B (112)

Aleksandra PRZYBYŁO, Zdzisław ONDERKA
Akademia Górniczo-Hutnicza, Zakład Geoinformatyki i Informatyki Stosowanej

PROJEKT SYSTEMU BARTERU WIELOSTRONNEGO[1]

Streszczenie. Artykuł dotyczy projektu sieciowego systemu barterowego. Przedstawiono w nim szczegółową analizę zapotrzebowań, popartą diagramami przypadków użycia, oraz projekt logiczny systemu (modele statyczny i dynamiczny). Zaprojektowano również szczegółowo bazę danych. Wykonano testy wydajnościowe oraz poprawności kodu za pomocą walidatora organizacji W3C.

Słowa kluczowe: system barterowy, projekt obiektowy, relacyjna baza danych

BARTER SYSTEM PROJECT

Summary. The article considers a network barter system. A detailed demand analysis, supported by use-cases diagrams as well as logical system project (static and dynamic models) were presented here. A database was designed in detail. Correctness tests was carried out by means of Validator W3C.

Keywords: barter system, object oriented project, relational database

1. Barter wielostronny

Barter jest pierwotną formą wymiany bezgotówkowej, czyli towar za towar bądź usługa za towar [1]. Strony uzgadniają między sobą wartość towarów/usług i dążą do tego, aby bilans między nimi był zerowy. Z reguły jest to dość trudne. Nowoczesną formą wymiany towarowej jest barter wielostronny [1]. Działa on na tej samej zasadzie co jego pierwotna wersja. Ponadto, umożlwia on wymianę towarów w ramach więcej niż jednej transakcji. Ostatecznie wartości wymienianych dóbr nie muszą być identyczne, co zdecydowanie ułatwia znalezienie potencjalnego klienta do wymiany.

[1] Praca wykonywana w ramach prac statutowych.

Idea ta została użyta do stworzenia tak zwanych platform systemu barterowego. Zrzeszają one grupy przedsiębiorców, umożliwiając im dokonanie wzajemnych transakcji. Transakcje te rozliczane są na podstawie umownej jednostki płatniczej (UJP), a więc bezgotówkowo. Każdy użytkownik platformy ma własne konto, gdzie zapisywany jest jego aktualny stan UJP.

Pierwsza platforma barteru wielostronnego WIR (niem. *Wirtschaftsring* – ang. *business circle)* powstała już w 1934 roku w Szwajcarii [1]. Następnie pomysł ten pojawił się w Stanach Zjednoczonych, gdzie do dzisiaj działa najwięcej platform wymiany barterowej.

W Polsce jako pierwsza wdrożeniem projektu barterowego zajęła się spółka Barter System Polska [2]. W systemie tym mogą rejestrować się małe oraz średnie przedsiębiorstwa.

Innym popularnym serwisem jest Barter Online [3]. Podobnie jak w poprzednim serwisie, każda firma będąca członkiem za dokonaną sprzedaż może otrzymać wirtualne pieniądze, nazywane złotówkami barterowymi (e-PLN). Wadą serwisu jest długi czas oczekiwania na założenie konta, co wynika z przyznania kredytów w e-PLN dla nowych użytkowników. Zaletą natomiast jest bezpieczeństwo podczas prowadzenia transakcji.

2. Wymagania funkcjonalne

Poniżej zdefiniowano skrótowo podstawowe wymagania funkcjonalne zaprojektowanego i zaimplementowanego systemu barterowego:

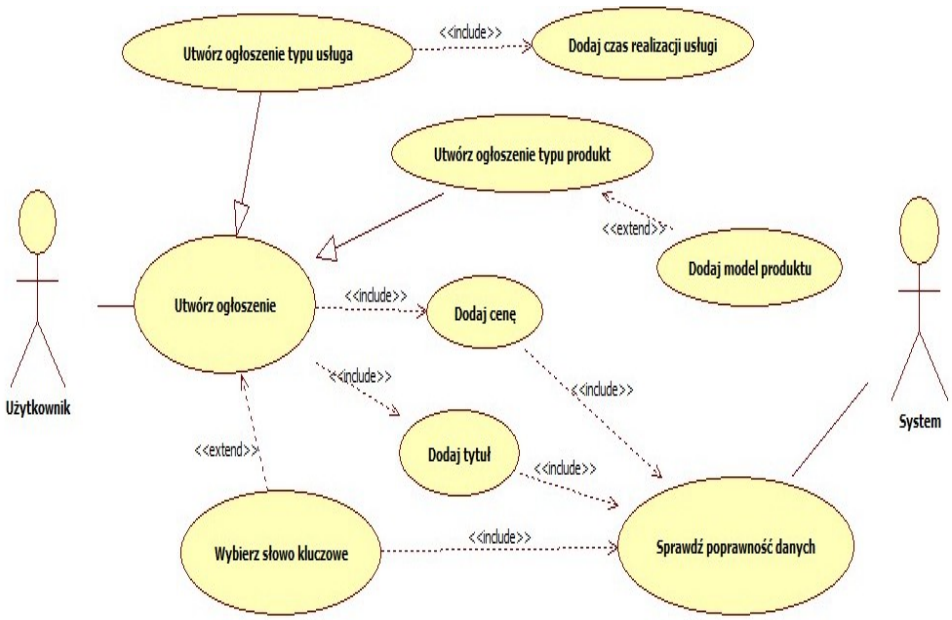
- **Rejestracja użytkowników** – w celu zarejestrowania w serwisie użytkownik zobowiązany jest do wypełnienia kilkunastu pól dotyczących danych osobowych, takich jak: imię, nazwisko, e-mail czy adres zamieszkania. Zanim informacje zostaną zapisane w bazie danych muszą zostać zweryfikowane pod względem poprawności.
- **Konta użytkowników** – po zalogowaniu do systemu możliwe jest zarządzanie: danymi osobowymi, ogłoszeniami, ofertami złożonymi innym użytkownikom, wiadomościami, transakcjami, obserwowanymi towarami/usługami oraz przyznawaniem punktów po zrealizowaniu transakcji.
- **Wystawianie ogłoszeń** – dodanie ogłoszenia dotyczącego produktu/usługi następuje po poprawnym uzupełnieniu formularza. Użytkownik ma także możliwość zarządzania ogłoszeniami.
- **Wysyłanie wiadomości** – realizacja transakcji wymaga sprawnej komunikacji między użytkownikami, dlatego system musi mieć własny mechanizm wysyłania i odbioru wiadomości.
- **Proponowanie wymiany** – Składanie propozycji wymiany można znacznie uprościć, tworząc odpowiedni formularz. Zawierałby on tylko najważniejsze, pola np. numer

przedmiotu czy rodzaj zapłaty (barter lub/i wirtualna waluta). W porównaniu do wiadomości tekstowej, tak złożone oferty są bardziej przejrzyste.

- **Dokonywanie transakcji** – transakcje można podzielić na cztery główne etapy:
  - wybieranie użytkownika, z którym chce się dokonać wymiany,
  - wybór towarów/usług użytkownika, wchodzących w skład transakcji,
  - dodawanie własnych przedmiotów i usług. W przypadku wymiany „towar za wirtualną gotówkę" ten krok może być pominięty.
  - czwartym etapem jest ewentualna dopłata w wirtualnej walucie (gdy wartość przedmiotów/usług nie jest równoważna).

  Ponadto, każdy z użytkowników musi mieć możliwość anulowania, podglądu oraz potwierdzenia transakcji.
- **Wystawianie ocen użytkownika** – ocena użytkownika ma na celu potwierdzenie jego wiarygodności. W najprostszej postaci są to punkty pozytywne, negatywne lub neutralne. W wersji bardziej rozwiniętej może to być system komentarzy.
- **Wyszukiwarka** – narzędzie pozwalające na szybkie przeszukanie bazy aktualnych ogłoszeń, zgodnie z odpowiednim filtrem.
- **Śledzenie ogłoszeń** – podgląd ogłoszeń, którymi użytkownik jest zainteresowany.

3. Projekt systemu



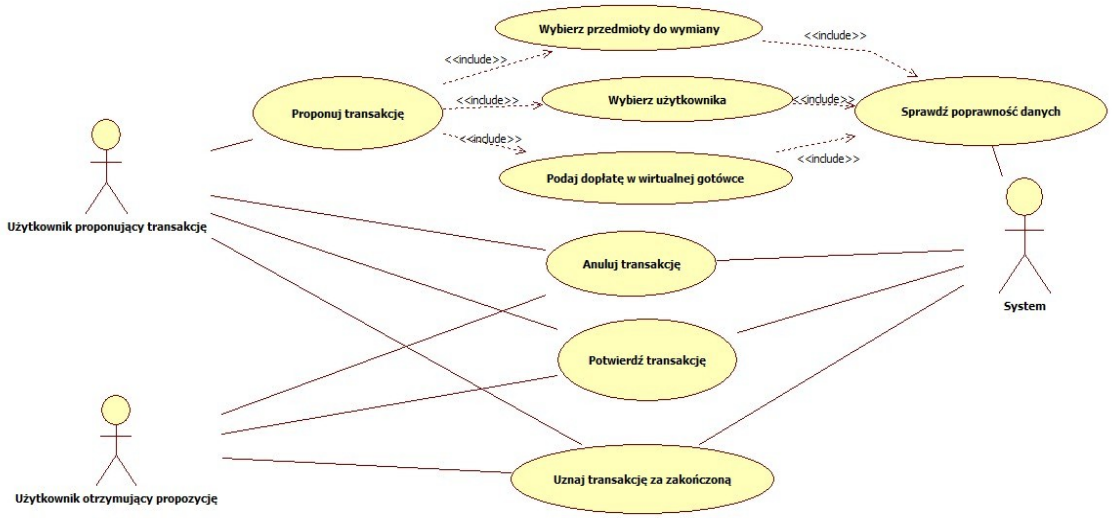
Rys. 1. Przypadki użycia dodawania ogłoszenia wymiany
Fig. 1. Add announcement use case

Ze względu na dużą złożoność projektu, poniżej przedstawiono jedynie kilka przypadków użycia: dla funkcjonalności „dodawania ogłoszenia” (rys. 1) i dla „transakcji” (rys. 2), modele statyczne dla funkcjonalności „wystawiania ogłoszenia” (rys. 3) i „dodawania transakcji” (rys. 4) oraz modele dynamiczne dla funkcjonalności „wystawiania ogłoszenia” (rys. 5) i dla „wystawiania oferty” (rys. 6).

Tabela 1

Opis diagramu przypadków użycia dla dodawania ogłoszenia

Cel	Dodawanie nowego ogłoszenia do baz już istniejących.
Aktorzy	Zalogowany użytkownik oraz system bazodanowy.
Warunki początkowe	Użytkownik jest w posiadaniu produktów, które chce oddać w barterze.
Warunki końcowe	Użytkownikowi udało się dodać produkt do bazy danych.
Przebieg główny	1. Użytkownik podaje wymagane dane, takie jak tytuł czy cena, oraz ewentualnie słowa kluczowe. 2. Użytkownik wybiera typ ogłoszenia, czy będzie to produkt czy usługa. 3. Użytkownik uzupełnia dodatkowe pola. W przypadku produktu są to model oraz stan, a w przypadku usługi czas jej realizacji. 4. System weryfikuje dane. 5. Jeżeli dane były poprawne, to nowe ogłoszenie zostało dodane do bazy danych.
Przebiegi alternatywne	Użytkownik podał niepoprawne dane.
Sytuacje wyjątkowe	1. Nastąpił błąd bazy danych. 2. Nastąpił błąd z sesją użytkownika.
Reguły	Użytkownik musi być zalogowany.



Rys. 2. Przypadki użycia procesu wymiany
Fig. 2. Exchange Process use case

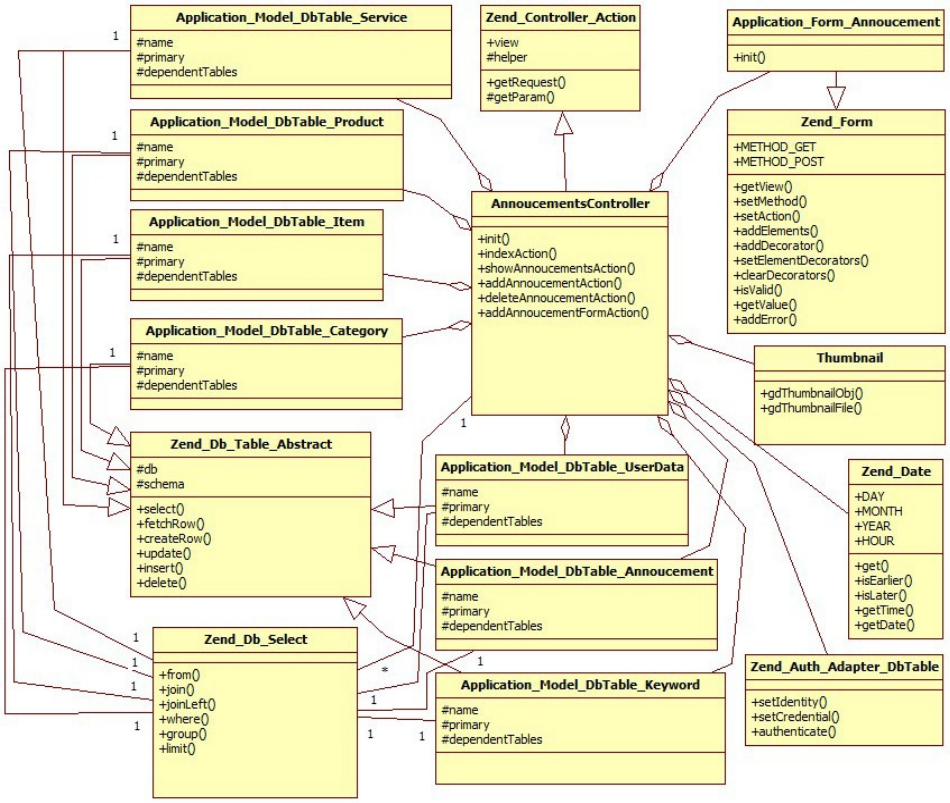
Tabela 2

Opis diagramu przypadków użycia dla procesu wymiany

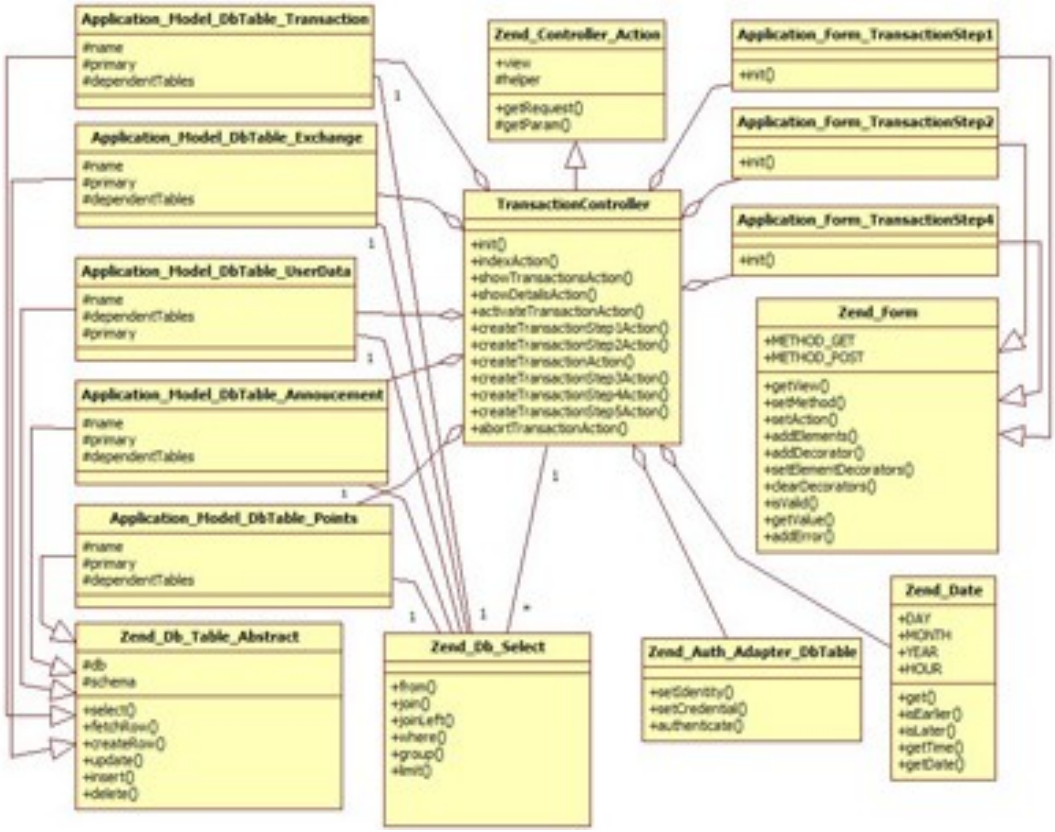
Cel	Wymiana towarów/usług z lub bez użycia wirtualnego pieniądza.
Aktorzy	Użytkownik proponujący transakcję, użytkownik otrzymujący propozycję oraz system bazodanowy.
Warunki początk.	Obustronna potrzeba wymiany
Warunki końcowe	Obydwaj użytkownicy dokonają wymiany towarów/usługi oraz uznają transakcję za zakończoną.
Przebieg główny	1. Użytkownik, proponujący transakcję, podaje login osoby, z którą chce dokonać wymiany. 2. Dokonuje on wyboru swoich przedmiotów lub usług, biorących udział w wymianie. 3. Wybiera towary lub usługi kontrahenta. 4. Wprowadza kwotę dopłaty w wirtualnej gotówce (jeżeli istnieje taka potrzeba). 5. Weryfikacja danych. 6. Dodanie transakcji do bazy danych. 7. Potwierdzenie transakcji przez proponującego. 8. Potwierdzenie transakcji przez otrzymującego propozycję. 9. Wymiana towarów/usług. 10. Potwierdzenie zakończenia transakcji przez proponującego. 11. Potwierdzenie transakcji przez otrzymującego propozycję. 12. Przejście wcześniej ustalonych środków z konta na konto.
Przebiegi alternatywne	1. Anulowanie transakcji przez jednego z użytkowników. 2. Wprowadzenie przez proponującego błędnych danych. 3. Jeden z użytkowników nie potwierdził transakcji. 4. Jeden z użytkowników nie uznał transakcji za zakończoną.
Sytuacje wyjątkowe	1. Powstał błąd związany z sesją użytkownika. 2. Powstał błąd z zapisem do bazy.
Reguły	Użytkownicy muszą być zalogowani.

Ważniejsze klasy to:

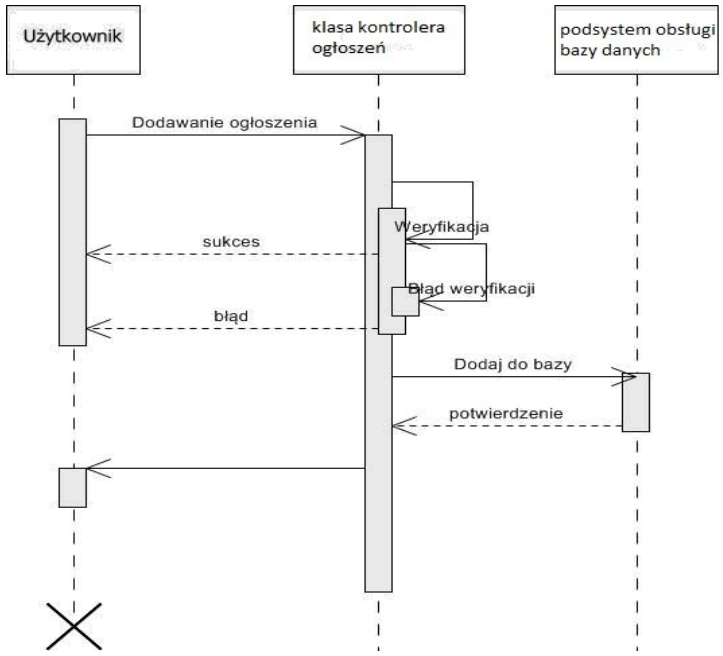
- **Zend_Form –** klasa obsługi formularzy,
- **Zend_Db_Table_Abstract, Zend_Db_Select –** klasy obsługi baz danych,
- **AnnouncementController –** klasa reprezentująca kontroler ogłoszeń,
- **TransactionController –** klasa reprezentująca kontroler transakcji,
- **Application_Model_DbTable_Category, Application_Model_DbTable_Announcement, Application_Model_DbTable_Exchange, Application_Model_DbTable_Transaction –** klasy reprezentujące tabele w bazie danych,
- **Application_Form_TransactionStep1, Application_Form_TransactionStep2, Applcation_Form_TransactionStep4 –** klasy reprezentujące kolejne kroki transakcji.



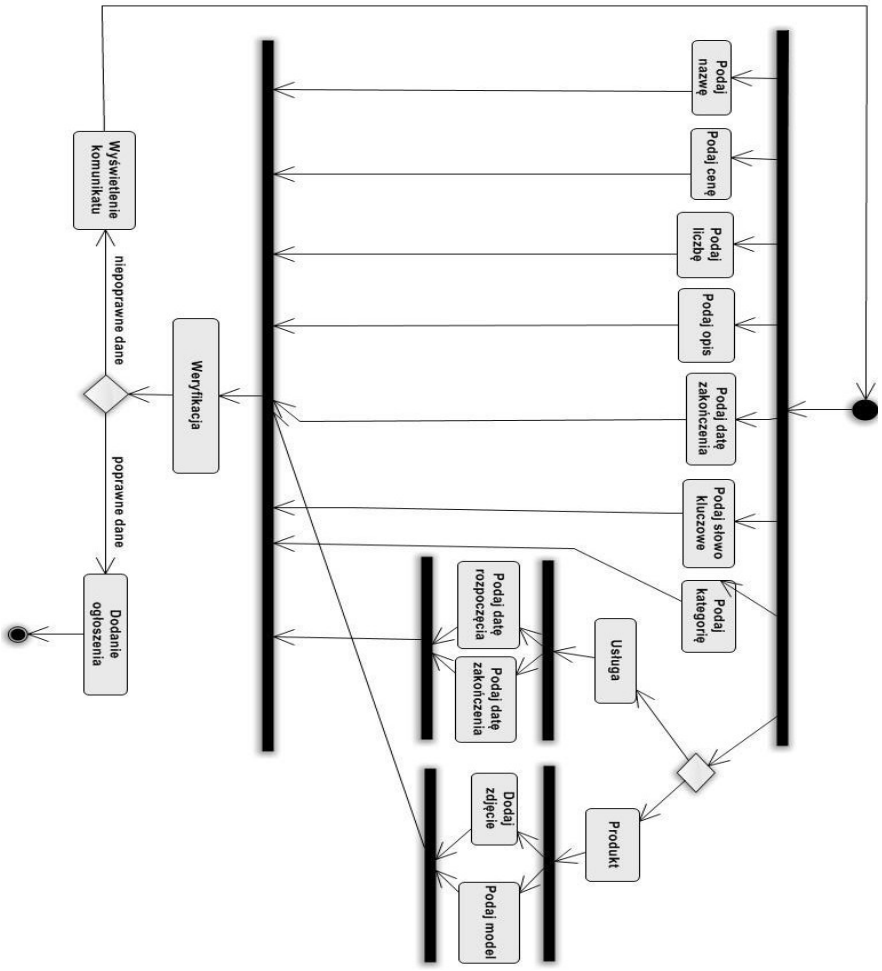
Rys. 3. Model obiektowy wystawiania ogłoszenia
Fig. 3. Object oriented model of announcement adding



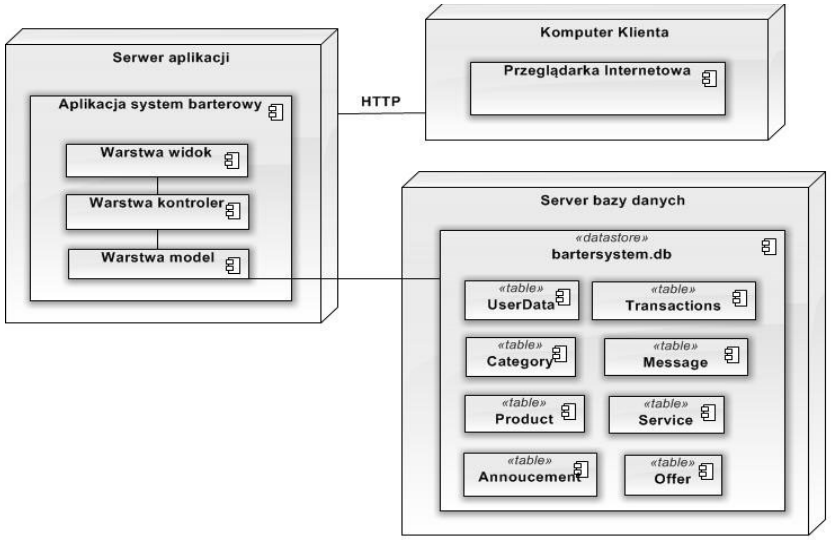
Rys. 4. Model obiektowy dodawania transakcji
Fig. 4. Object oriented model of transaction adding



Rys. 5. Diagram sekwencji dla procesu dodawania ogłoszenia
Fig. 5. Sequence diagram for announcement adding process



Rys. 6. Diagram aktywności dla procesu dodawania ogłoszenia
Fig. 6. Activity diagram of announcement adding process



Rys. 7. Diagram wdrożenia
Fig. 7. Deployment diagram

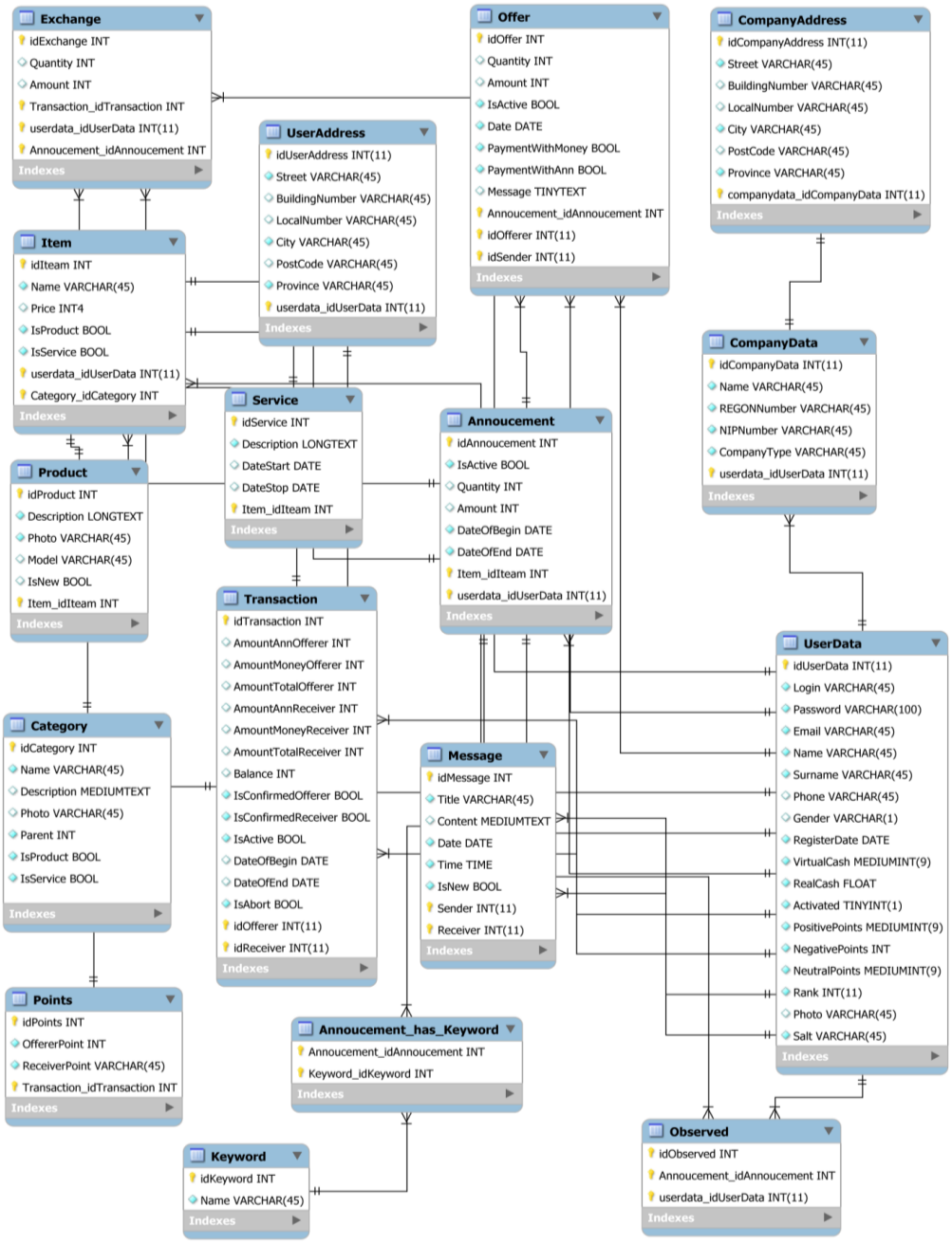
4. Wdrożenie

System barterowy składa się z trzech warstw. Warstwa widok odpowiada za prezentacje aplikacji, warstwa kontroler za obsługę żądań użytkownika, a warstwa model za połączenie z serwerem bazodanowym MySQL. Użytkownik systemu, korzystając z przeglądarki, wysyła żądanie wyświetlenia strony WWW do serwera aplikacji. Warstwa kontroler je obsługuje. Następnie komunikuje się ona z warstwą model, w celu uzyskania danych z bazy. Jest to realizowane poprzez mapowanie obiektowo-relacyjne (ang. *Object-Relational Mapping ORM*). Ostatecznie warstwa kontroler przekazuje te dane do części odpowiedzialnej za prezentacje strony. Tak powstały widok aplikacji zostaje wyświetlony w przeglądarce użytkownika.

5. Projekt bazy danych

Projekt bazy danych składa się z 17 tabel połączonych między sobą relacjami (rys. 8). Przechowują one: **UserData** – dane użytkownika, **UserAddress** – adres użytkownika, **CompanyData** – informacje o firmie, **CompanyAddress** – adres firmy, **Item** – podstawowe wiadomości o produkcie/usłudze, **Category** – informacje o kategorii, **Product** – dane produktu, **Service** – dane usługi, **Announcement** – informacje o ogłoszeniu, **Offer** – oferty, **Exchange** – oferty wymieniane w transakcjach, **Message** – wiadomości, **Transaction** – dane transakcji, **Observed** – obserwowane oferty, **Points** – punkty, **Keyword** – słowa kluczowe, **Announcement_has_Keyword** – związki słów kluczowych z ogłoszeniami.

6. Implementacja i testy



Rys. 8. Relacyjna baza danych
Fig. 8. Relational database

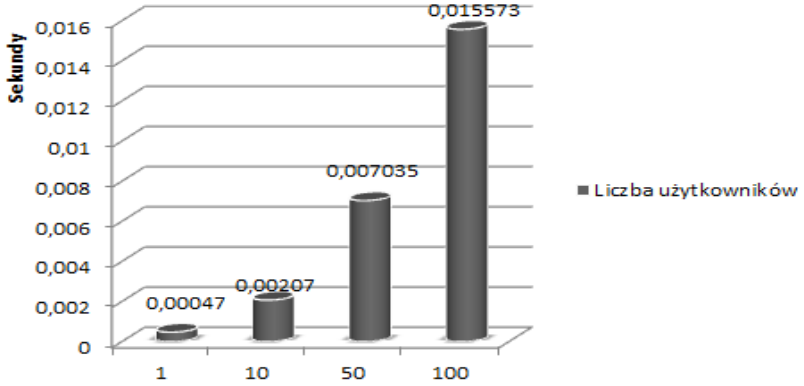
Do implementacji systemu użyto technologii PHP (Zend Framework) [4], HTML 5, CSS, jQuery (biblioteka JavaScript) oraz silnik bazodanowy MySQL. Zastał zastosowany wzorzec projektowy Model-Widok-Kontroler, co miało na calu oddzielenie logiki biznesowej aplikacji od prezentacji danych.[4].

Ze względu na wielkość systemu, w pierwszym rzędzie zaimplementowano moduły: kont użytkowników, wystawiania oraz wyszukiwania ogłoszeń, a także tworzenia transakcji.

W późniejszej fazie dodano możliwość rejestracji, wysyłania wiadomości, proponowania wymiany, wystawiania ocen użytkownikom oraz śledzenia ogłoszeń.

W ramach testowania aplikacji w pierwszym rzędzie sprawdzono wpływ prędkości ładowania strony na jej wydajność. Całkowity czas od momentu wpisania adresu strony do pełnego wczytania i wyświetlenia w przeglądarce bywa bardzo zróżnicowany. Jest on zależny od liczby osób korzystających jednocześnie z serwisu internetowego, prędkości łącza, a także w dużej mierze od samej jego budowy. Optymalizacja strony ma więc wpływ na jej szybkość ładowania oraz np. na wyższą pozycję w wyszukiwarce Google.

Wykres na rys. 9 obrazuje długość ładowania strony w zależności od liczby użytkowników jednocześnie z niej korzystających. Wynik dotyczy jedynie wczytania wszystkich zasobów przez serwer. Nie jest natomiast wliczony czas generowania strony przez przeglądarkę. Pomiar został dokonany za pomocą programu ApacheBench. Zasymulowano 10, 50 oraz 100 jednoczesnych żądań HTTP na serwerze.



Rys. 9. Czas ładowania strony w zależności od liczby użytkowników
Fig. 9. Relationship between page load time and number of visitors

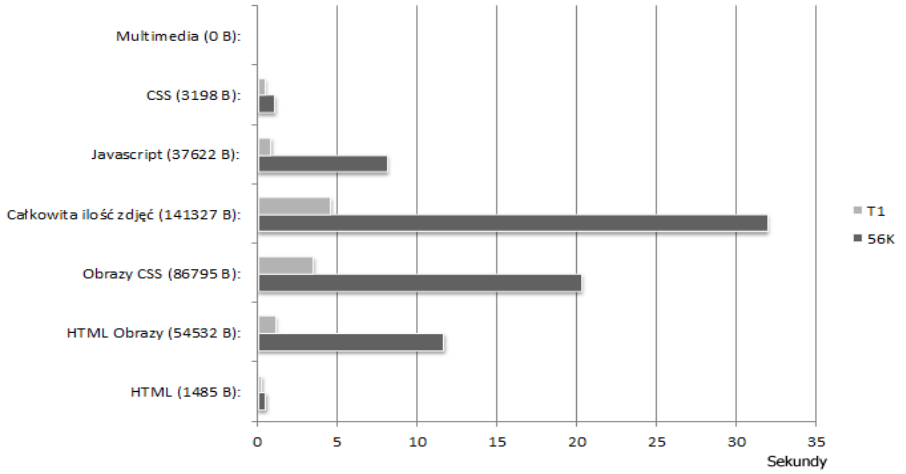
Jak można było się spodziewać, im więcej osób jednocześnie korzysta z serwisu, tym większe występuje opóźnienie we wczytywaniu strony. Projektujący aplikację nie ma wpływu na liczbę użytkowników odwiedzających stronę. Może on jedynie tak ją zoptymalizować, aby została jak najszybciej wczytana.

Wykres na rys. 10 obrazuje czas wczytywania poszczególnych elementów aplikacji przez przeglądarkę, w zależności od prędkości pobierania danych (56 kbit/s i 1536 kb/s). Pomiar został przeprowadzony przy użyciu funkcji php time(). Jak można zauważyć, duże znaczenie ma rozmiar grafiki. Należy zatem skompresować grafikę tak, by zachować dostateczną jakość, a jednocześnie jak najbardziej zmniejszyć jej rozmiar. W zaprojektowanym serwisie barterowym liczba grafiki została maksymalnie zredukowana oraz zapisana w formacie PNG. Format ten obsługuje stopniowaną przezroczystość oraz 48-bitową głębię kolorów.

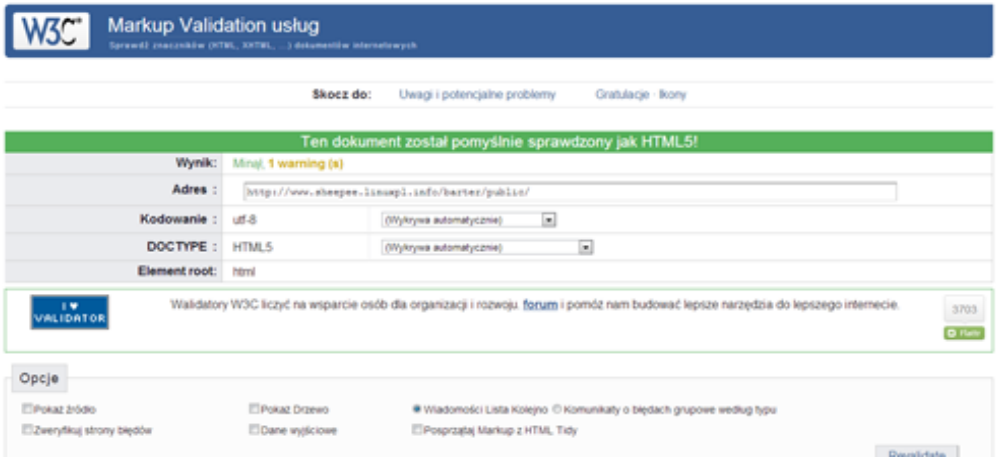
Bardzo ważnym aspektem jest również ograniczenie zapytań do serwera, powstających na skutek łączenia plików z arkuszami stylów czy kodem JavaScript. Im mniejsza ich liczba, tym kod zostanie szybciej wczytany.

Kolejnym ważnym elementem jest poprawność budowy aplikacji internetowej. Pozbawiony błędów kod ma nie tylko pozytywny wpływ na roboty wyszukiwarek, ale także na szybkość ładowania strony. Niezbędne jest więc testowanie jej za pomocą walidatora organizacji W3C (rys. 11).

Dla opisywanego serwisu warto także zoptymalizować stronę pod względem zapytań do bazy danych, ponieważ częste ich wykonywanie znacząco opóźnia ładowanie strony. Także nadużywanie symbolu '*' w zapytaniach znacznie obciąża serwer. Z bazy danych powinny być pobierane tylko te informacje, które są potrzebne [8]. W aplikacji system barterowy liczba zapytań do bazy została możliwie jak najbardziej zmniejszona. Pobierane są tylko niezbędne informacje, co wpływa pozytywnie na czas ładowania strony.



Rys. 10. Prędkość wczytywania poszczególnych elementów (strona główna)
Fig. 10. Page load time (main page)



Rys. 11. Wynik walidacji – strona startowa
Fig. 11. W3C Validator – main page

W pracy przedstawiono najważniejsze elementy projektu inżynierskiego dla realizacji portalu społecznościowego „Barter wielostronny". Jest to typowy przykład zastosowania baz danych, podobnie jak np. w [7]. Opisaną aplikację można by dodatkowo urozmaicić, stosując trójwymiarowy interfejs graficzny [6], oraz zastosować mechanizmy bazowodanowe dla implementacji w chmurze [5].

BIBLIOGRAFIA

1. Kent D.: Healthy Money Healthy Planet. Craig Potton, 2005.
2. Barter System Polska, http://bartersystem.pl/serwis/.
3. Barter Online, http://barteronline.pl/.
4. Lo N., Brown S., Allen R.: Zend Framework in Action. Manning Publications Co., Greenwich 2009.
5. Bajerski P., Augustyn D. R., Bach M., Brzeski R., Duszeńko A., Werner A.: Bazy danych a chmury obliczeniowe. Studia Informatica, Vol. 33, No. 2A (105), Gliwice 2012.
6. Augustyn D. R., Warchał Ł., Żelaźnicki P.: Tworzenie nowoczesnych aplikacji z trójwymiarowym graficznym interfejsem użytkownika, opartym na technologii CUBE. Studia Informatica, Vol. 32, No. 2B (97), Gliwice 2011.
7. Haręźlak K., Dziurdzia D., Stelmach M.: Społecznościowy serwis dla rowerzystów. Studia Informatica, Vol. 33, No. 2B (106), Gliwice 2012.
8. Barczak A., Sacharczuk D.: Typowe problemy optymalizacji zapytań SQL przy tworzeniu średnich i dużych serwisów aplikacji WWW. Studia Informatica, Vol. 33, No. 2B (106), Gliwice 2012.

Wpłynęło do Redakcji 9 stycznia 2013 r.

Abstract

The work was carried out a full object-oriented design for the web-based system of the longest-running trade barter scheme (WIR – *Wirtschaftsring*) The concept of the barter was defined and a detailed analysis of needs was performed i.e. it was formulated functional and nonfunctional requirements supported by the use-case diagrams. Static (object-oriented diagram) and dynamic models were implemented (sequence diagram, activity diagram) and a diagram of the implementation of the system was discussed. For the purposes of the system

a relational database was designed in detail. As part of testing phase the system efficiency tests were carried out and correctness of the code was checked with the W3C validator. Application was implemented using the Model-View-Controller design pattern to separate the business logic of the data presentation. Tests showed the dependence of the effectiveness of the portal on the number of users and the size of the drawing used on the portal.

Adresy

Aleksandra PRZYBYŁO: Akademia Górniczo-Hutnicza, Zakład Geoinformatyki i Informatyki Stosowanej, Al. Mickiewicza 30, 30-059 Kraków, aleksandraprzybylo@gmail.com.
Zdzisław ONDERKA: Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, Al. Mickiewicza 30, 30-059, Kraków, zonderka@agh.edu.pl.