ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1999
Seria: INFORMATYKA z. 36 Nr kol. 1414

Paweł KASPROWSKI
Politechnika Śląska, Instytut Informatyki

NIEZALEŻNY SPRZĘTOWO DOSTĘP DO BAZ DANYCH Z SIECI INTERNET - OMÓWIENIE MOŻLIWOŚCI I PREZENTACJA ZASTOSOWAŃ

Streszczenie. Artykuł omawia metody i możliwości realizacji dostępu do baz danych z sieci Internet. Zaprezentowane są pokrótce specyfikacje CGI oraz JDBC oraz przykładowa metoda realizacji.

HARDWARE INDEPENDENT DATABASES ACCESS FROM INTERNET – POSSIBILITIES AND EXAMPLES

Summary. This article describes methods and possibilities of realizing hardware independent access to any database server. There are CGI and JDBC specifications and a simple example of realization presented in the paper.

1. Geneza problemu

Zagadnienie umożliwienia dostępu do danych dla użytkowników sieci Internet staje się w obecnym czasie jednym z kluczowych wyzwań informatyki. Internet od zarania został stworzony w celu udostępniania informacji. Początkowo była to najczęściej informacja statyczna w postaci stron WWW. Jednak rosnąca ilość informacji, a co za tym idzie, nadmierne rozrastanie się serwisów internetowych, wymusiła konieczność skorzystania z narzędzi do dynamicznej generacji stron WWW na żądanie użytkownika. W ten sposób powstała powszechnie obecnie używana technologia CGI (Common Gateway Interface). Pierwsze programy generujące strony WWW tworzone były przez specjalistów od sieci i najczęściej pobierały dane z rozbudowanych plików tekstowych. Wkrótce rozwiązanie to przestało wystarczać i do przygotowania danych zaczęto używać oprogramowania do tego przeznaczonego – baz danych.

W chwili obecnej zagadnienie umożliwienia dostępu do baz danych z sieci Internet jest jedną z najszybciej rozwijających się dziedzin informatyki. Idea Intranetu – wykorzystania technologii internetowych w sieci lokalnej zwiększyła jeszcze zapotrzebowanie na gotowe rozwiązania. Nic więc dziwnego, że większość liczących się firm produkujących serwery baz danych dysponuje już własnymi rozwiązaniami realizacji dostępu do nich przez Internet. Rozwiązania te są jednak najczęściej mało elastyczne – zakładają pewną filozofię współpracy z bazą i nie pozwalają na zbyt daleko idące modyfikacje proponowanych gotowych formularzy. W wielu przypadkach poważną barierą są bardzo wysokie wymagania narzucane komputerowi klienckiemu (tak jest na przykład w rozwiązaniu firm Magic czy Oracle). Rozwiązania dedykowane są także najczęściej dla konkretnego serwera.

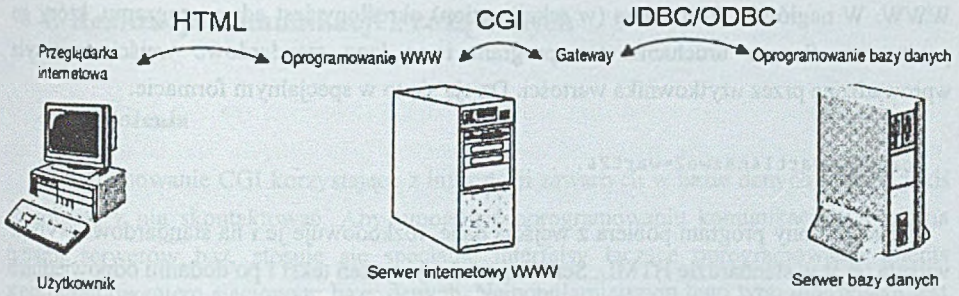
Rynek jest niezwykle dynamiczny i narzędzia podlegają nieustannym modyfikacjom. Celem pracy nie jest więc omówienie istniejących rozwiązań, gdyż takie podejście do problemu spowodowałoby całkowitą nieaktualność materiału w przeciągu kilku miesięcy. W tej sytuacji w niniejszej pracy starano się omówić zasadę działania tego typu rozwiązań w miejsce zagłębiania się w opis konkretnych gotowych narzędzi. Jako przykład zaproponowano własne, niezależne sprzętowo i programowo rozwiązanie oraz omówiono użyte podczas jego projektowania standardy.

2. Zadanie i sposób realizacji

Zadaniem pracy było stworzenie biblioteki klas pozwalających na budowę pośredniego ogniwa (gateway'a) umożliwiającego dostęp do dowolnego serwera bazy danych przez dowolnego użytkownika Internetu korzystającego ze standardowego oprogramowania. Przez standardowe oprogramowanie rozumie się zaimplementowany protokół TCP/IP oraz dowolną przeglądarkę internetową.

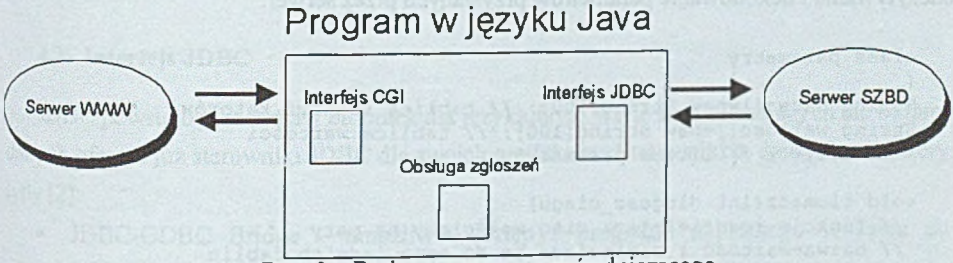
Ze względu na otwarte założenia dotyczące systemu operacyjnego i rodzaju serwera zdecydowano się na zastosowanie języka Java. Zagwarantowało to pełną niezależność aplikacji od konkretnych możliwości systemowych.

Klasyczny system sieciowego dostępu do serwera bazy danych rozbudowany został o ogniwo pośrednie składające się z serwera internetowego świadczącego usługi WWW wraz z zainstalowanym na nim oprogramowaniem.



Rys. 1. Architektura systemu
Fig. 1. System architecture

Oprogramowanie zainstalowane na serwerze WWW pełni rolę gateway'a przyjmującego zgłoszenia od użytkowników i po przekształceniu ich na zapytania przekazującego je do serwera bazy danych. Jak widać z rysunku 1, aplikacja ta komunikuje się ze stroną internetową (serwerem WWW) za pomocą protokołu CGI, a ze stroną danych (serwer bazy danych) za pomocą JDBC. Strukturę aplikacji przedstawia rysunek 2.



Rys. 2. Budowa programu pośredniczącego
Fig. 2. The structure of gateway program

Poniżej opisano pokrótce oba standardy oraz sposób realizacji połączenia.

3. Realizacja komunikacji z użytkownikiem

W chwili obecnej najpopularniejszym standardem komunikacji pomiędzy użytkownikiem a oprogramowaniem zainstalowanym na serwerze WWW jest tak zwany Common Gateway Interface (CGI) [3].

CGI wykorzystuje dane z formularzy tworzonych w języku HTML [7]. Gdy użytkownik wypełni formularz i naciśnie przycisk „Submit" – jego zawartość przesyłana jest do serwera

WWW. W nagłówku formularza (w sekcji action) określony jest adres programu, który go przetwarza. Serwer uruchamia ten program i na jego standardowe wejście przesyła wprowadzone przez użytkownika wartości. Dzieje się to w specjalnym formacie:

```
nazwa1=wart1&nazwa2=wart2&...
```

Uruchomiony program pobiera z wejścia dane, rozkodowuje je i na standardowe wyjście wysyła tekst w standardzie HTML. Serwer przechwytuje ten tekst i po dodaniu odpowiednich nagłówków wysyła go użytkownikowi jako gotową stronę WWW.

Program przetwarzający zgłoszenia użytkownika (zwany także często ze względów historycznych skryptem CGI) może być napisany w dowolnym języku programowania pozwalającym odczytywać dane ze standardowego wejścia i pisać na standardowe wyjście.

Ze względu na wzmiankowane na wstępie założenie niezależności od systemu zdecydowano się na implementację programu w języku Java [5].

Alternatywą wobec skryptów CGI są tak zwane servlety (Java servlets), które działają jako komponenty dołączane bezpośrednio do oprogramowania serwera. Bardziej szczegółowy opis tej technologii można znaleźć w [1].

Poniżej przedstawiono definicję klasy `parametry` odpowiedzialnej w programie za odczytywanie i dekodowanie parametrów przysłanych przez serwer.

```
class parametry
{
String nazwa[]=new String[100]; // tablica identyfikatorów
String wartosc[]=new String[100]; // tablica wartości
int ilosc=0; // liczba parametrów

void tlumacz(int dlugosc_ciagu)
 // funkcja rozdzielająca ciąg wejściowy na pary
 // nazwa-wartość i zapisująca je do odpowiednich tablic

String podajWartosc(String nazwa)
  // funkcja zwracająca wartość parametru o podanej nazwie
```

Jak widać, klasa ta udostępnia dwie funkcje: `tlumacz()` i `podajWartosc()`. Obiekt tworzony jest na początku pracy programu. Następnie wywoływana jest funkcja `tlumacz()`, która wypełnia obiekt danymi na podstawie wartości wejściowych. Podczas dalszej pracy programu wartości parametrów sprawdzane są za pomocą funkcji `podajWartosc()`.

4. Realizacja komunikacji z bazą danych

4.1. Założenia

Oprogramowanie CGI korzystające z informacji zawartych w bazie danych musi w jakiś sposób się z nią skontaktować. Aby umożliwić oprogramowaniu komunikację z wieloma typami serwerów baz, stosuje się specjalne interfejsy łączące oprogramowanie klienta z oprogramowaniem sieciowym bazy danych. Najpopularniejszym tego typu interfejsem jest ODBC firmy Microsoft.

Aby użyć ODBC, sieciowe oprogramowanie dostępu do serwera bazy musi posiadać możliwość rejestracji jako sterownik ODBC. Jednocześnie narzędzie do tworzenia oprogramowania CGI musi posiadać możliwość wysyłania zapytań do takiego sterownika.

Tak więc skrypt CGI musi po odebraniu zgłoszenia od użytkownika przekształcić je do postaci zapytania SQL i przekazać do serwera bazy danych. Do realizacji takiej komunikacji z serwerem bazy wykorzystano promowaną przez firmę SUN i zaimplementowaną w języku Java technologię JDBC. Nazwa JDBC wprost sugeruje konkurencyjność tego rozwiązania do proponowanego przez Microsoft standardu ODBC. W rzeczywistości JDBC i ODBC mogą się w prosty sposób uzupełniać, co wykazano w dalszej części pracy.

4.2. Interfejs JDBC

JDBC powstało jako zestaw bibliotek dla języka Java. Wiele firm zajmujących się bazami danych oferuje już sterowniki JDBC dla swoich produktów. Sterowniki te dzieli się na cztery typy [2]:

- JDBC-ODBC Bridge – umożliwia dostęp z programu napisanego w Javie do serwerów baz danych za pośrednictwem ODBC.
- Native-API partly-Java driver – konwertuje wywołania z programu w Javie na API konkretnego producenta bazy. W ten sposób program taki zachowuje się z punktu widzenia serwera jak zwykła aplikacja kliencka napisana pod ten system.
- Net-protocol all-Java driver – wywołania z Javy tłumaczone są na niezależny protokół sieciowy. Protokół ten jest następnie tłumaczony przez serwer bazy. Wprowadzenie pośredniego protokołu sieciowego niezależnego od rodzaju bazy jest krokiem naprzód w stosunku do ODBC, gdzie tłumaczenie na protokół specjalizowany odbywa się jeszcze u klienta.
- Native-protocol all-Java driver – konwertuje wywołania z programu w Javie bezpośrednio na protokół sieciowy konkretnego producenta. Jest to realizowane przez

rozszerzenie o tę możliwość klienckiej części oprogramowania serwera bazy danych i zależy od dobrej woli producenta.

Z zaprezentowanych typów bez wątpienia najciekawszy jest typ trzeci. Umożliwia on całkowite uniezależnienie klienta od rodzaju serwera bazy danych. Jednak sterownik taki wymaga wprowadzenia zrozumiałego dla serwerów baz niezależnego protokołu komunikacji sieciowej. W chwili obecnej istnieje kilka takich protokołów, ale żaden z nich nie jest na tyle rozpowszechniony, aby można było mówić o standardzie. Tak więc pozycja ODBC jako uniwersalnego protokołu dostępu do bazy danych pozostaje na razie niezagrożona.

4.3. Realizacja

W tej sytuacji, aby można było mówić o rzeczywistej przenośności tworzonego rozwiązania, zdecydowano się na skorzystanie ze sterownika typu JDBC-ODBC Bridge. Sterownik taki umożliwia programom w Javie korzystanie z zarejestrowanych w ODBC serwerów bazodanowych.

Użycie ODBC narzuca konieczność instalacji sterownika ODBC producenta bazy danych na komputerze klienta. W tym konkretnym przypadku ograniczenie to nie jest jednak poważne, ponieważ klientem jest komputer, na którym znajduje się serwer WWW, a nie komputer użytkownika. Tak więc użycie ODBC na etapie pośrednim komunikacji z bazą jest dla użytkownika niezauważalne i nie zmusza go do instalacji jakiegokolwiek dodatkowego oprogramowania.

Oczywiście, rozwiązanie to nie jest optymalne i w toku dalszych prac przewiduje się całkowite odejście od ODBC.

Wszelkie odwołanie do bazy danych realizowane są przez obiekt `baza`. Korzysta on ze standardowych klas z biblioteki `java.sql`, jak:

`Connection` – przechowuje uchwyt do bazy.

Przykładowa metoda: `createStatement()`

`Statement` – do definiowania i wykonywania zapytań SQL.

Przykładowa metoda: `executeQuery(String zapytanie)`

`ResultSet` – obiekty przechowujące tablice odpowiedzi na zapytania SQL.

Przykładowa metoda: `getString(String nazwa_atrybutu)`

5. Realizacja obsługi zgłoszeń

Po przygotowaniu interfejsów komunikacyjnych dla obu stron pozostaje zbudowanie obiektu odpowiedzialnego za obsługę zgłoszeń użytkownika. Podane przez niego parametry

są pobierane za pomocą metod interfejsu CGI, a komunikację z bazą zapewnia interfejs JDBC.

5.1. Założenia

Od strony użytkownika program zachowuje się jak zwykły program do obsługi bazy danych. Umożliwia przechodzenie pomiędzy oknami, wyszukiwanie elementów w bazie, modyfikację bazy. Użytkownik na każde swoje działanie uzyskuje reakcję w postaci pojawienia się kolejnego okna – formularza. Tak więc od strony oprogramowania musi powstać szereg procedur obsługi generujących kolejne 'okna' programu.

Program jest uruchamiany za każdym razem, gdy serwer przekaże mu zgłoszenie, a po zakończeniu przetwarzania kończy swe działanie. W momencie otrzymania zgłoszenia program musi posiadać informację, jakim oknem odpowiedzieć i jakie dane w nim zaprezentować.

Oczywiście, część parametrów wprowadzanych jest jawnie przez użytkownika. Jeśli chodzi na przykład o zaprezentowanie tablicy rekordów wg filtra, parametry filtra użytkownik podaje w odpowiednich polach formularza. Jednak oprócz tych jawnych parametrów istnieje także potrzeba przekazania programowi innych parametrów niewidocznych dla użytkownika.

Przede wszystkim musi to być identyfikator okna, jakie ma być pokazane. Jak już bowiem stwierdzono, moduł obsługi składa się z pewnej ilości procedur generujących okna (są to strony WWW zawierające formularze). Każde okno ma pewien predefiniowany wygląd i istnieje możliwość przejścia z niego do innych okien przez naciśnięcie któregoś z przycisków znajdujących się na nim.

Przykładowo, na ekranie znajduje się okno z dwoma przyciskami - Wybór Filtra i Edycja Rekordu. Po naciśnięciu przycisku Wybór Filtra użytkownik powinien przejść do okna Filtr, a po naciśnięciu przycisku Edycja Rekordu do okna Edycja.

W tym celu zastosowano pole z atrybutem HIDDEN [8]. Pole o takim atrybucie jest obiektem formularza, lecz nie jest widoczne dla użytkownika. Użytkownik nie ma też możliwości bezpośredniej modyfikacji jego zawartości. Wszelkie modyfikacje mogą być dokonywane jedynie programowo.

W polu typu HIDDEN o nazwie okno zamieszczony został identyfikator okna, które ma być wywołane w następnej kolejności.

```
<input type=hidden name=okno value='brak'>
```

Po naciśnięciu któregoś z przycisków najpierw zmieniana jest programowo zawartość pola okno.

```
<input type=submit value='Edycja Rekordu'
onClick="document.forms[0].okno.value='edycja'">

<input type=submit value='Wybór Filtra'
onClick="document.forms[0].okno.value='filtr'">
```

Ponieważ przyciski są typu SUBMIT, po ich naciśnięciu następuje wysłanie parametrów do serwera WWW i pobranie kolejnej strony. Strona generowana jest programowo przez procedurę, której identyfikator określa pole `okno`.

W powyższym przykładzie korzysta się z języka JavaScript [6] zaimplementowanego w praktycznie wszystkich używanych dziś przeglądarkach internetowych.

Oprócz pola okno można wprowadzić na stronę inne - statyczne - parametry przenoszące dane pomiędzy kolejnymi instancjami programu. Może to być na przykład pole `użytkownik` z nazwą lub kodem użytkownika. Przydatne także być może umieszczenie pola `poprzedni` określającego, z którego okna nastąpiło wywołanie okna bieżącego.

5.2. Budowa programu

Ze względu na wymagania otoczenia program składa się z pewnej ilości funkcji obsługujących poszczególne okna (formularze) wysyłane do użytkownika. Po uruchomieniu program odczytuje parametry i identyfikuje, jakie okno powinno być zwrócone. Następnie uruchamiana jest odpowiednia funkcja.

Typowa funkcja generująca okno odczytuje parametry i na ich podstawie generuje tekst zapytania SQL. Następnie łączy się z bazą, wysyła do niej zapytanie i pobiera jego wyniki.

Kolejny etap to budowa okna – a więc strony WWW zawierającej pobrane wyniki. Aby strona zachowała możliwość aktywnej współpracy z programem, konieczne jest, aby znalazł się na niej formularz. Do formularza dołączane są w postaci atrybutów HIDDEN wszelkie dodatkowe parametry.

6. Bezpieczeństwo udostępniania danych

W omawianych w poprzednim rozdziale zmiennych HIDDEN nie powinny być przechowywane żadne informacje tajne - należy pamiętać, że nie są one w żaden sposób zabezpieczane przed niepowołanym dostępem.

W przypadku zaistnienia konieczności zablokowania dostępu do niektórych danych przez nieupoważnionych użytkowników nieodzowne staje się wprowadzenie mechanizmu identyfikacji użytkownika.

W klasycznej realizacji serwera internetowego użytkownik korzystający z jego zasobów jest zupełnie anonimowy. Aby zidentyfikować użytkownika, należy na formularzu WWW zapytać go o jego login i hasło. Podstawowym problem jest jednak to, że hasło wysyłane w postaci jawnej siecią może być odczytane przez kogoś nieuprawnionego. Jedyną możliwością zaradzenia temu problemowi jest zakodowanie hasła po stronie klienta.

Przy założeniu że do pełnego użytkowania systemu klientowi potrzebna jest dowolna przeglądarka internetowa, problem można rozwiązać tylko poprzez skorzystanie z możliwości przeglądarki lub przez użycie własnego apletu.

Aplet [4] jest to mały program w języku Java ściągany siecią wraz ze stroną WWW. Można wyobrazić sobie aplet, który wyświetla okno logowania, pobiera login i hasło i następnie koduje je (np.: za pomocą PGP) i wpisuje w którąś ze zmiennych formularza.

Drugim problemem jest utrzymanie połączenia z użytkownikiem po jego identyfikacji. Jak już wspomniano, skrypt CGI jest uruchamiany od nowa dla każdego zgłoszenia użytkownika. Bezsensowne byłoby żądanie podania przez użytkownika hasła na każdym wysyłanym formularzu.

Utrzymanie połączenia zostało zrealizowane w następujący sposób. W momencie zalogowania (przesłania zakodowanego identyfikatora i hasła) użytkownik zostaje zarejestrowany przez system. Zapamiętywana jest jego identyfikacja i kilka dodatkowych danych na temat połączenia (np.: adres IP jego komputera). Jednocześnie system nadaje mu identyfikator liczbowy i na formularzu do niego wysyłanym umieszcza go w polu HIDDEN. W momencie gdy użytkownik wypełni formularz i prześle go z powrotem do serwera, program porównuje identyfikator i pozostałe dane z zapisanymi wcześniej. Jeśli identyfikacja przebiegnie prawidłowo, następuje obsługa zgłoszenia. Ponieważ identyfikatory nie są w żaden sposób kodowane, dla zwiększenia bezpieczeństwa systemu program generuje inny identyfikator. Tak więc wraz z każdą odpowiedzią użytkownik otrzymuje w polu HIDDEN nowy identyfikator.

Ponieważ w komunikacji przez Internet nie istnieje stałe połączenie klienta z serwerem, a wszelkie odwołania są od siebie niezależne – serwer nigdy nie posiada informacji, czy dany użytkownik zakończył już pracę. W tej sytuacji konieczne jest wprowadzenie timeout'u – jeśli zarejestrowany użytkownik przez 10 minut nie połączy się z serwerem, jego identyfikator ulega przedawnieniu i konieczne będzie ponowne logowanie.

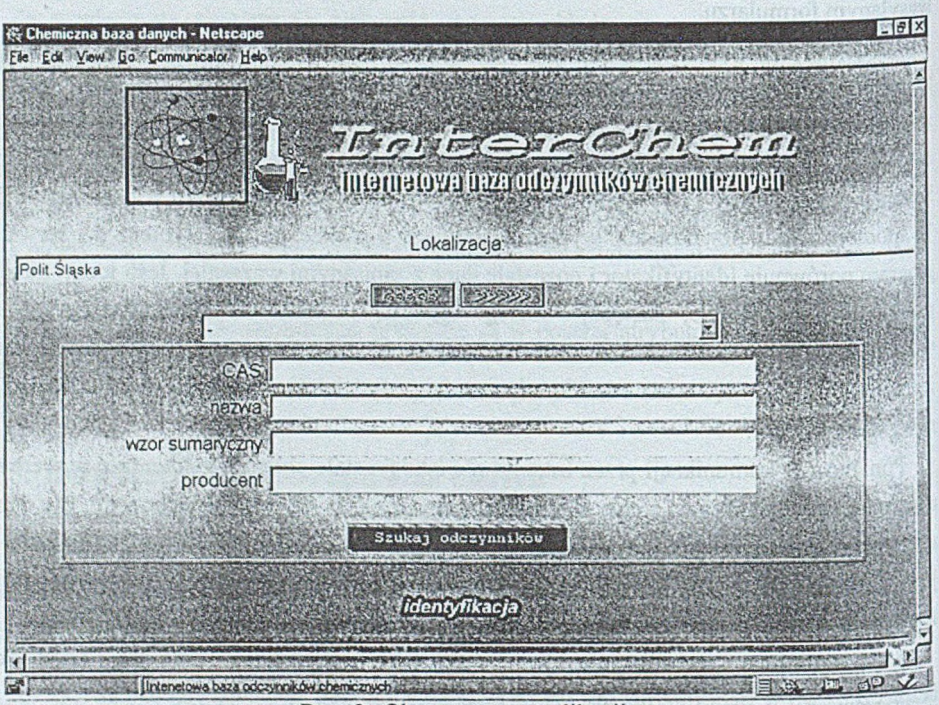
Zaprezentowana metoda nie jest całkowicie bezpieczna, ale jak wiadomo, bezpieczeństwo w sieci Internet jest jak na razie przedmiotem nieustających badań.

7. Przykładowy projekt sieciowej aplikacji udostępniającej dane

Większość opisanych powyżej rozwiązań wykorzystana została w trakcie projektowania aplikacji pozwalającej na uzyskanie dostępu do bazy odczynników chemicznych dla Wydziału Chemicznego Politechniki Śląskiej.

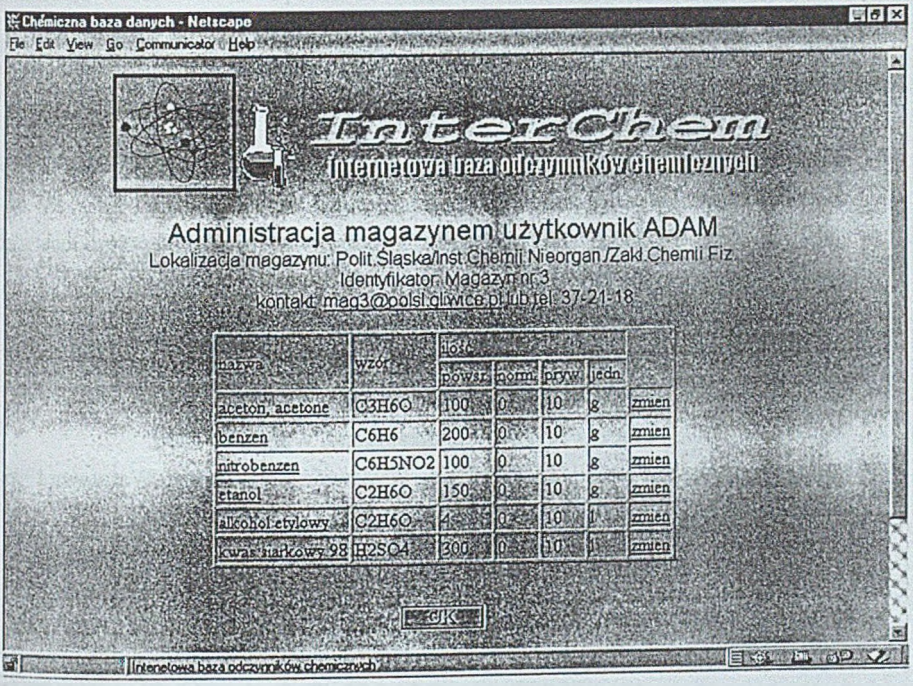
Baza przechowuje dane dwojakiego rodzaju: informacje na temat właściwości fizykochemicznych odczynników i dane o ich stanach w danej jednostce. Dostęp do danych na temat stanów powinien być limitowany – użytkownik anonimowy nie powinien mieć dostępu do danych na temat odczynników niebezpiecznych – na przykład o działaniu narkotycznym lub wybuchowym. Jednak każdy użytkownik Internetu powinien mieć dostęp do danych dotyczących odczynników niegroźnych. W związku z tym konieczne było zastosowanie identyfikacji na żądanie.

Rysunek 3 pokazuje okno startowe programu. Jak widać, użytkownik może za jego pomocą zdefiniować parametry wyszukiwania i przejść bezpośrednio do listy znalezionych odczynników. Może także za pomocą przycisku identyfikacja przejść do okna logowania i „przedstawić się" systemowi. Jak już stwierdzono, użytkownik zalogowany ma dostęp do większej ilości danych.



Rys. 3. Okno startowe aplikacji
Fig. 3. Application start window

Aplikacja pozwala na wyszukiwanie w bazie interesujących nas odczynników. Oprócz tego można za jej pomocą prowadzić za pośrednictwem Internetu pełną gospodarkę magazynową (przychody, rozchody, stany) zasobów odczynników.



Rys. 4. Okno administracyjne
Fig. 4. Administration window

Także administrator bazy ma do niej dostęp całkowicie przez Internet. Może za pomocą zwykłej przeglądarki zakładać użytkowników, magazyny, przeglądać i modyfikować uprawnienia.

Zapewnienie dostępu przez Internet bardzo upraszcza przeprowadzenie wdrożenia – jedyną czynnością techniczną jest postawienie podłączonego do Internetu odpowiednio skonfigurowanego serwera i poinformowanie potencjalnych użytkowników o jego adresie.

8. Podsumowanie

Dostęp do systemów baz danych za pomocą Internetu wydaje się dziś być ciekawą innowacją lub modą. Jednocześnie zagadnienia bezpieczeństwa powodują dużą niechęć właścicieli baz do ich udostępniania w Sieci. Wkrótce jednak taki sposób dostępu do danych

stanie się codziennością oraz podstawowym wymaganiem dla każdego systemu. Podobnie jak dziś użytkownik nie chce słyszeć o interfejsie tekstowym, preferując graficzny, tak w niedługiej przyszłości brak współpracy z Internetem może być uważany za duży brak każdego systemu.

LITERATURA

1. Kasprowski P.: Zastosowanie mechanizmu Java Servlets do udostępniania zasobów w sieci Internet. ZN Politechniki Śląskiej s. Informatyka z. 34, Gliwice 1997.
2. The JDBC(tm) Universal Data Access API (http://www.java.sun.com/products/jdbc), Sun Microsystems Inc., 1995-1999.
3. Jamsa K., Lalani S.: Programowanie WWW. MIKOM, Warszawa 1997.
4. Hoff A., Shaio S., Starbuck O.: Java. Helion, Gliwice 1996.
5. Jamsa K.: Java. MIKOM, Warszawa 1996.
6. Walter S.J., Weiss A.: Język JavaScript. Intersoftland, Warszawa 1996.
7. Taylor D.: Tworzenie stron HTML. Oficyna Wydawnicza README, Warszawa 1996.
8. Castro E.: Po prostu HTML. Helion, Gliwice 1996.

Recenzent: Dr inż. Leszek Płonka

Wpłynęło do Redakcji 31 marca 1999 r.

Abstract

This article describes possibilities of presenting database information via Internet.

This topic becomes more and more important nowadays. There are plenty of solutions presented and the market of Internet is growing very fast. This article doesn't try to present ready-to-use tools but instead of this presents a philosophy of Internet access and a simple example of how to do it. Java platform was used to implement an example program.

There is architecture of the system (Fig. 1) and a structure of the program (Fig. 2) presented in section 2.

The following section 3 tells about CGI Interface and section 4 discusses JDBC database access interface. Section 5 describes the program structure and a way of creating user interface.

Section 6 discusses some problems of user identification and secure matters.

Last section 7 shows some screenshots (Fig. 3 and 4) from an example application. This application presents chemistry database in Internet.