STUDIA INFORMATICA 2003
Volume 24 Number 1 (52)

Paweł KASPROWSKI
Politechnika Śląska, Instytut Informatyki

ANALIZA WYBRANYCH MOŻLIWOŚCI UDOSTĘPNIANIA DANYCH ZAWARTYCH W REPOZYTORIUM DANYCH UŻYTKOWNIKOM WWW

Streszczenie. W artykule przedstawiono analizę możliwości udostępniania danych użytkownikom sieci WWW. Zostały omówione możliwe do zastosowania standardy, ich wady i zalety oraz przedstawiono rozwiązanie zastosowane podczas projektowania Regionalnego Systemu Informacji Przestrzennej.

Słowa kluczowe: repozytorium danych, WWW.

THE ANALYSIS OF CHOSEN POSSIBILITIES OF PRESENTING DATA FROM THE DATA REPOSITORY TO WWW USERS

Summary. The paper describes possibilities of presenting data to WWW users. Different methods and architectures were introduced with notification about all pros and cons. RSIP (Regional System of Geographic Information) was presented as an example of such architecture.

Keywords: data repository, WWW.

1. Wstęp

Regionalny System Informacji Przestrzennej (RSIP) był wspólnym projektem Instytutu Systemów Przestrzennych i Katastralnych oraz Politechniki Śląskiej w Gliwicach. Zadaniem projektu było zgromadzenie i opracowanie danych geograficznych dotyczących województwa śląskiego. Dane przechowywane są w repozytorium danych opartym na serwerze bazodanowym firmy Oracle.

Aby repozytorium danych spełniało swoje zadania, konieczne było zapewnienie dostępu do niego użytkownikom. Użytkownicy musieli mieć możliwość przeglądania danych w

postaci dla nich czytelnej, wyszukiwania informacji w repozytoriach i generowania raportów na temat interesujących ich obiektów. Aby zapewnić taką funkcjonalność, konieczne było stworzenie oprogramowania umożliwiającego realizację dostępu do danych. W związku z tym, jednym z elementów projektu było zaprojektowanie mechanizmu udostępniania zgromadzonych danych użytkownikom sieci Internet (tak zwanego portalu RSIP). Niniejsze opracowanie powstało na podstawie prac związanych z tą częścią projektu.

Istnieje wiele możliwych sposobów realizacji tego typu oprogramowania. W poniższym opracowaniu przedstawiono pokrótce te możliwości oraz zarysowano właściwości rozwiązania, które zostało ostatecznie wybrane jako docelowe, uzasadniając wybór.

2. Przegląd i ocena możliwych do zastosowania standardów

2.1. Dedykowana aplikacja kliencka

Najprostszym i najbardziej oczywistym rozwiązaniem jest stworzenie dedykowanej aplikacji umożliwiającej, po zainstalowaniu na komputerze użytkownika, łączenie się z repozytoriami danych. Takie rozwiązanie ma jednak szereg niedogodności:

- Konieczność ograniczenia się do jednej platformy systemowej po stronie użytkownika.

Można oczywiście próbować skompilować aplikacje pod różne systemy, ale na dłuższą metę takie rozwiązanie jest bardzo czasochłonne i trudne do utrzymania.

- Problemy z dystrybucją oprogramowania.

Użytkownik chcąc korzystać z repozytorium musi otrzymać oprogramowanie i zainstalować je na swoim komputerze. Może się przy tym okazać, że pojawią się jakieś konflikty z innym zainstalowanym oprogramowaniem lub z samym systemem operacyjnym. Konieczne jest więc wcześniejsze przetestowanie oprogramowania w jak największej ilości możliwych konfiguracji.

- Problemy z aktualizacją oprogramowania.

Każda poprawka lub zmiana dokonana w oprogramowaniu musi nieść za sobą konieczność aktualizacji wszystkich działających kopii oprogramowania u wszystkich użytkowników. Oprogramowanie to mogłoby być automatycznie ściągane przez sieć, lecz wtedy znów pojawiają się kłopoty omówione w poprzednim punkcie.

- Duże wymagania sprzętowe dla użytkownika.

Komputer, na którym pracuje użytkownik, powinien mieć wydajność wystarczającą do zainstalowania i uruchomienia całości oprogramowania. Zważywszy, że w grę wchodzi

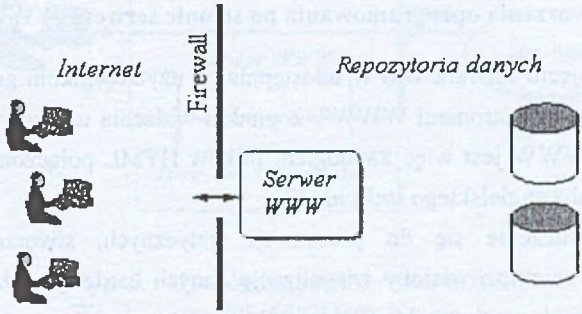
komunikacja z odległą bazą danych i wizualizacja graficzna jej zawartości, może to się stać problemem.

- Problemy z bezpieczeństwem repozytorium.

Dedykowana aplikacja kontaktuje się bezpośrednio z komputerem będącym serwerem repozytorium. Konieczne jest więc otwarcie go na świat i udostępnienie jego zasobów. Sytuacja taka sprzyja ewentualnym próbom niepowołanego dostępu do danych.

2.2. Wykorzystanie technologii internetowych

Uwzględniając problemy przedstawione w poprzednim rozdziale, lepszym rozwiązaniem wydaje się być wykorzystanie możliwości przeglądarki internetowej typu Internet Explorer czy Netscape Navigator. W rozwiązaniu takim klasyczna architektura klient-serwer rozszerzona zostaje o ogniwo pośrednie – serwer WWW [1]. Serwer ten znajduje się najczęściej na osobnym komputerze i tylko za jego pośrednictwem możliwy jest dostęp z sieci Internet do zawartości repozytorium (rys. 1).



Rys. 1. Sposób użycia serwera WWW
Fig. 1. The method of WWW server usage

Zalety takiego rozwiązania zostały przedstawione poniżej.

- Wieloplatformowość.

Jedynym warunkiem dla systemu użytkownika jest umożliwienie uruchomienia w tym systemie standardowej przeglądarki internetowej, jak Netscape Navigator czy Internet Explorer. Jeśli chodzi o produkt firmy Netscape, to jest on w tej chwili dostępny na praktycznie wszystkich liczących się platformach systemowych [2].

- Otwartość.

Aby korzystać z informacji zawartych w repozytorium, użytkownik nie musi w żaden sposób rekonfigurować swojego systemu. Wszystko realizowane jest przez oprogramowanie przeglądarki internetowej.

- Brak problemów z aktualizacją.

W momencie zmiany oprogramowania konieczna jest jedynie aktualizacja oprogramowania serwera WWW. Użytkownicy nie muszą nawet wiedzieć, że coś się zmieniło.

- Standardowe wymagania sprzętowe.

Komputer użytkownika musi umożliwiać uruchomienie przeglądarki internetowej. Jeśli to wymaganie jest spełnione – może on pracować z repozytorium. Dzięki temu twórca oprogramowania unika mozolnego testowania swojego oprogramowania na różnych platformach sprzętowych w celu określenia niezbędnego minimum.

- Podwyższone bezpieczeństwo danych w repozytorium.

Użytkownik w trakcie pracy komunikuje się jedynie z serwerem WWW (patrz rys. 1), który udostępnia mu różne usługi związane z danymi. Użytkownik nie ma i nie musi mieć żadnego dostępu do samego serwera repozytorium danych. Nie wie nawet, gdzie ten serwer się znajduje i jakiego oprogramowania systemowego używa. Rozwiązanie takie w istotny sposób podwyższa bezpieczeństwo całego systemu.

2.3. Standardy tworzenia oprogramowania po stronie serwera WWW

W klasycznym pojęciu serwera WWW udostępnia on użytkownikom gotowe dokumenty w formacie HTML zwane „stronami WWW". Z punktu widzenia użytkownika przeglądarki internetowej serwer WWW jest więc katalogiem plików HTML połączonych między sobą odsyłaczami, zwanymi z angielskiego *linkami* [3].

Oczywiście, ograniczenie się do prezentacji statycznych, stworzonych wcześniej dokumentów HTML uniemożliwiałoby wizualizację danych bardziej złożonych, w sposób zgodny z oczekiwaniami użytkownika. W związku z tym na serwerach WWW istnieje możliwość podstawienia w miejsce wywoływanej przez użytkownika strony HTML oprogramowania generującego taką stronę zgodnie z zapotrzebowaniem.

W chwili obecnej istnieje wiele standardów realizacji takiego rozwiązania. Klasycznym, obecnie rzadko już spotykanym w „czystej" formie rozwiązaniem jest standard CGI (Common Gateway Interface) [4]. W rozwiązaniu tym użytkownik podając adres WWW. podaje bezpośrednio nazwę programu, który chce wykonać, na przykład wpisanie w pole adresu:

```
http://moj.serwer.www/skrypty/strona.exe
```

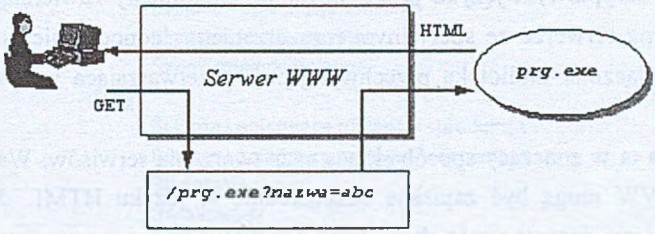
spowoduje uruchomienie na serwerze programu `strona.exe`. Program wywoływany jest przez oprogramowanie serwera WWW i na swoim wyjściu generuje kod HTML

przechwytywany przez serwer (rys. 2). Serwer dodaje do kodu konieczne nagłówki i wysyła go do użytkownika jako gotową stronę WWW.

Aby program mógł reagować na zapotrzebowanie użytkownika przesłaniem mu odpowiednio skonfigurowanego dokumentu, konieczne jest umożliwienie przesłania przez użytkownika informacji zwrotnej do serwera. Do tego celu wykorzystywane są formularze HTML [3]. Na formularzu użytkownik ma (często niejawną) możliwość podania wartości parametrów żądanej strony. Parametry te wysyłane są do serwera wraz z żądaniem przesłania następnej strony. Parametry mogą być wysłane bezpośrednio w adresie WWW (metoda GET), jak np.:

```
http://moj.serwer.www/skrypty/strona.exe?nazwa=abc
```

lub wewnątrz pakietu (metoda POST). Aplikacja odbiera parametry jako swoje parametry wywołania.



Rys. 2. Uproszczona zasada działania CGI
Fig. 2. Simplified CGI introduction

Architektura CGI niesie z sobą podstawową niedogodność. Gdy z serwerem pracuje wielu użytkowników jednocześnie, dla każdego z nich musi być niezależnie uruchamiana aplikacja. Ponieważ, z założenia, serwer WWW może być udostępniany bardzo dużej liczbie użytkowników, wymaga to dużej ilości pamięci i obciąża dość znacznie system.

W związku z tym wielu producentów serwerów WWW rozszerzyło swoje oprogramowanie o specjalne API, za pomocą których można rozszerzać samo jądro serwera. Rozwiązanie takie podnosi znacząco wydajność systemu – to samo oprogramowanie raz uruchomione obsługuje wszystkich chcących z niego skorzystać użytkowników. Oczywiście, minusem takiego rozwiązania jest przywiązanie swojej aplikacji do konkretnego serwera WWW bez możliwości prostej zmiany.

Programy CGI mogły być tworzone w dowolnym języku umożliwiającym pisanie na standardowe wyjście. API serwerów WWW są z reguły tworzone pod konkretny język (zwykle jest to C/C++). Nie da się ukryć, że jest to dosyć poważne ograniczenie.

Niektóre z API stworzonych początkowo dla konkretnych serwerów stało się dzięki swoim dużym możliwościom, a także dzięki popularności produktów danej firmy, rodzajem standardu używanego także w innych serwerach WWW. Przykładem takiego rozwiązania jest ISAPI (Internet Server API) [5] zaprojektowane przez firmę Microsoft. Innym przykładem jest technologia servletów (Servlet API) [6]. Stworzona przez firmę Sun początkowo tylko dla serwera Java Web Server, technologia ta rozpowszechniła się dzięki stworzeniu specjalnych bibliotek dla innych serwerów internetowych.

2.4. Aktywne strony WWW

Innym, często stosowanym, rozwiązaniem jest użycie aktywnych stron WWW. Projektant serwisu internetowego wzbogaca dokumenty HTML o specjalne wstawki w jakimś, zwykle dedykowanym, skryptowym języku programowania. Dokumenty zawierające takie wstawki zapisywane są na serwerze ze specjalnym rozszerzeniem. Jednocześnie do serwera zostaje dynamicznie dołączona biblioteka przechwytująca i przetwarzająca wszystkie pliki z tym rozszerzeniem.

Technologia ta w znaczący sposób skraca czas tworzenia serwisów. Wszystkie elementy stałe stron WWW mogą być zapisane bezpośrednio w języku HTML. Jedynie elementy strony wymagające dostosowania do potrzeb użytkownika są generowane na bieżąco, w momencie ściągania przez niego strony.

W klasycznym rozwiązaniu przedstawionym w poprzednim rozdziale punktem wyjścia jest stworzenie programu. Program zawierać musi odpowiednie biblioteki, pliki nagłówkowe i wszystkie elementy konieczne dla prawidłowej kompilacji. Dopiero za pomocą tak przygotowanego programu można wygenerować stronę – choćby była to strona najprostsza z możliwych.

Zastosowanie aktywnych stron WWW odwraca niejako kolejność projektowania. Bazą aplikacji jest kod HTML i tylko w razie potrzeby można go rozbudowywać o wstawki w języku programowania. Takie „zorientowane na HTML" rozwiązanie jest znacznie wygodniejsze z punktu widzenia projektanta, choć posiada pewne ograniczenia.

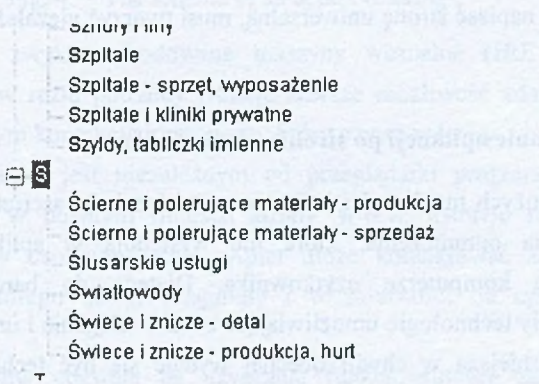
Najpopularniejszymi obecnie standardami umożliwiającymi generowanie aktywnych stron są: ASP (Active Server Pages) firmy Microsoft [7] oraz PHP (PHP – Hypertext Preprocessor) [8].

Należy podkreślić, że oba rozwiązania (także ASP!) są niezależne sprzętowo i mogą być uruchamiane na różnych serwerach WWW działających pod kontrolą różnych systemów operacyjnych.

2.5. Przenoszenie obciążenia na stronę użytkownika – język Javascript

Komunikacja z użytkownikiem za pomocą budowanych na serwerze i przesyłanych w formie gotowych dokumentów HTML ogranicza w istotny sposób możliwości tworzenia wygodnego interfejsu. Na stronach WWW istnieje co prawda możliwość umieszczania elementów aktywnych, jak pola tekstowe, pola wyboru czy przyciski, nie jest jednak możliwe nadanie stronie prawdziwej dynamiki – zmieniania się i dostosowywania do działań użytkownika.

Rozważmy typowy przykład rozwijalnego drzewa jak na rysunku 3.



Rys. 3. Przykładowe drzewo
Fig. 3. An example of the tree

Na przedstawionej liście branż wygodne byłoby umożliwienie użytkownikowi zwijania i rozwijania poszczególnych gałęzi drzewa (w tym przypadku kolejnych liter), tak aby mógł widzieć jedynie interesujące go pozycje.

Korzystając tylko z narzędzi po stronie serwera, jedynym rozwiązaniem jest wygenerowanie dokumentu tak, aby po każdym naciśnięciu przez użytkownika na węzeł wysyłane zostało do serwera żądanie następnej strony. Nowa strona zawierałaby elementy w konfiguracji zmienionej o zażądaną przez użytkownika (na przykład z rozwiniętym lub zwiniętym drzewem, na które nacisnął). Oczywiście, rozwiązanie takie jest mało efektywne, a przy wolnym łączu wręcz niemożliwe do użytkowania.

Jedynym sposobem rozwiązania tego problemu efektywnie jest przeniesienie części przetwarzania na komputer użytkownika. W takim rozwiązaniu przeglądarka użytkownika

otrzymałaby pełną listę elementów drzewa i kody funkcji realizujących obsługę drzewa i reakcje na działania użytkownika.

Najpopularniejszym sposobem realizacji przerzucenia obciążenia na komputer użytkownika jest wzbogacanie przesyłanych stron WWW o kod oparty na języku Javascript [9]. Za pomocą tego kodu można w dość szeroki sposób zdefiniować sposób pracy przeglądarki internetowej. Istnieje możliwość tworzenia na stronie elementów aktywnych, pojawiających się i znikających, przesuwających się itp. Podstawowym minusem tego rozwiązania są jednak ograniczenia przeglądarek internetowych. W chwili obecnej dwie najpopularniejsze na rynku i – co ważne – darmowe przeglądarki internetowe Netscape Navigator i Internet Explorer poprawnie obsługują Javascript. Problemem jest jednak ich wzajemna niezgodność w obsłudze bardziej zaawansowanych skryptów. Istotne różnice występują także pomiędzy poszczególnymi wersjami tego samego produktu (na przykład Netscape Navigator 4.7 i Netscape Navigator 6). Często prowadzi więc to do sytuacji, że programista, chcąc napisać stronę uniwersalną, musi tworzyć niezależnie skrypty dla różnych przeglądarek.

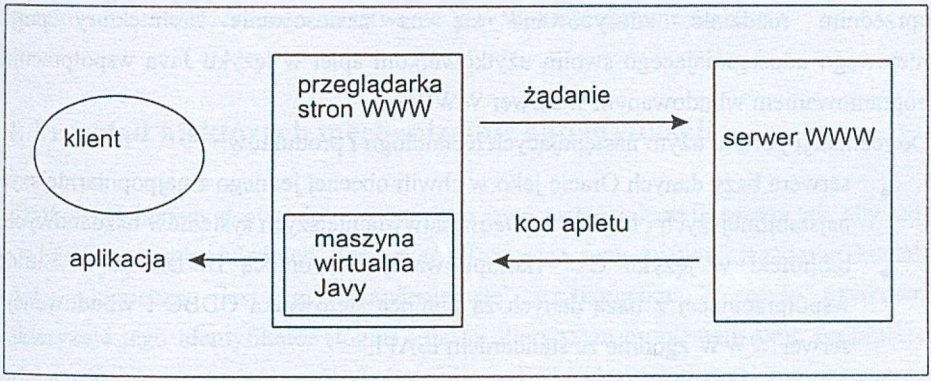
2.6. Uruchamianie aplikacji po stronie użytkownika

Mimo swych dużych możliwości i elastyczności język Javascript posiada jednak pewne trudne do obejścia ograniczenia, które nie występują w aplikacjach dedykowanych uruchamianych na komputerze użytkownika. Dlatego do bardziej zaawansowanych zastosowań powstały technologie umożliwiające zdalne ściąganie i uruchamianie programów z sieci. Najpopularniejsza w chwili obecnej wydaje się być technologia ActiveX firmy Microsoft [10]. Technologia ta umożliwia tworzenie i dystrybucję tak zwanych kontrolek ActiveX. Kontrolki mogą być tworzone w zasadzie w dowolnym języku programowania.

ActiveX jest rozwiązaniem „totalnym", umożliwiającym proste rozszerzanie funkcjonalności różnego typu aplikacji – w tym przeglądarki internetowej. Ta zaleta technologii jest jednocześnie jej największą wadą. Tworząc ją firma Microsoft, chcąc dać użytkownikom jak największe możliwości, zaniedbała jednocześnie zagadnienia związane z bezpieczeństwem. W chwili obecnej wraz z poprawkami i nowymi wersjami oprogramowania sytuacja się poprawia, ale trudno wciąż nazwać ActiveX technologią całkowicie bezpieczną.

Alternatywnym rozwiązaniem jest wykorzystanie technologii apletów tworzonych w języku Java [11]. Język Java został z założenia stworzony jako język niezależny sprzętowo i dlatego nie ma tak dużych możliwości ingerencji w system użytkownika. Każdy program napisany w Javie uruchamiany jest w specjalnie tworzonym środowisku „maszyny wirtualnej" (rys. 4). Całą komunikacją pomiędzy programem a środowiskiem komputera

zajmuje się oprogramowanie maszyny wirtualnej – łatwo więc zabezpieczyć się przed jego niedozwolonymi działaniami.



Rys. 4. Zasada działania apletów
Fig. 4. The schema of an applet working

Przeglądarki mają zwykle wbudowane maszyny wirtualne (JRE – Java Runtime Environment), jednak w razie potrzeby istnieje zawsze możliwość zdalnej instalacji JRE przez sieć przy pierwszym kontakcie z serwisem, który używa apletów.

Po uruchomieniu aplet jest niezależnym od przeglądarki programem, który zwykle pojawia się w oknie w pewnym miejscu strony WWW. Istnieje również możliwość uruchomienia apletu w osobnym oknie. Aplet może kontaktować się bezpośrednio z serwerem WWW, z którego został ściągnięty i w zależności od uprawnień z innymi komputerami w sieci.

Zastosowanie apletów pozwala na uzyskanie funkcjonalności bardzo zbliżonej do dedykowanej aplikacji przedstawionej w punkcie 2.1, pomijając jednocześnie opisane tam problemy.

3. Prototypowane rozwiązania architektoniczne

Projektując narzędzia dostępu do odczytu danych zawartych w repozytoriach oparto się na następujących założeniach:

- oprogramowanie musi mieć intuicyjny interfejs użytkownika, tak aby mogli go używać także użytkownicy, dla których komputer dopiero staje się narzędziem pracy,
- oprogramowanie musi być łatwo dostępne dla wszystkich chętnych,
- musi istnieć prosty sposób aktualizacji oprogramowania w przypadku dokonania zmian w strukturze repozytorium,

- należy zminimalizować koszty utrzymania oprogramowania.

Biorąc pod uwagę powyższe założenia i opierając się na rozważaniach zaprezentowanych w poprzednim rozdziale, zdecydowano się na zastosowanie architektury portalu internetowego udostępniającego swoim użytkownikom aplet w języku Java współpracujący z oprogramowaniem wbudowanym w serwer WWW.

Do realizacji portalu użyto następujących technologii i produktów:

- serwera bazy danych Oracle jako w chwili obecnej jednego z najpopularniejszych, najstabilniejszych i *last but not least* najwydajniejszych systemów bazdanowych,
- biblioteki w języku C++ (kompilowanej za pomocą Borland C++ Builder) współpracującej z bazą danych za pomocą sterownika ODBC i wbudowanej w serwer WWW zgodnie ze standardem ISAPI,
- serwera WWW zgodnego ze standardem ISAPI (na przykład Microsoft IIS),
- apletu w języku Java wysyłanego do użytkownika przy każdorazowym nawiązaniu połączenia.

Zastosowanie takiej architektury pozwoliło na rozdzielenie logiki systemu na trzy, współpracujące ze sobą, warstwy.

Pierwsza z nich – serwer bazy danych – odpowiada za udostępnianie danych zgodnie z zapotrzebowaniem zgłaszanym przez użytkownika. Przyjmuje zgłoszenia w postaci zapytań w języku SQL i odpowiada na nie ciągami danych. Oprogramowanie na tej warstwie zajmuje się także indeksowaniem i przechowywaniem wszelkich informacji na temat aktualnej pracy użytkowników, parametrów przez nich zdefiniowanych, zaznaczonych obiektów itd.

Warstwa druga – serwer WWW rozbudowany o bibliotekę DLL w języku C++ - stanowi ogniwo pośrednie pomiędzy repozytorium a apletem. Oprogramowanie przyjmuje pytania o dane zgłaszane przez aplet, przetwarza je na postać zapytań SQL do bazy danych i wysyła do serwera bazy danych. Następnie przetwarza zwrócone wyniki na postać zrozumiałą dla apletu i przesyła je do niego za pomocą protokołu HTTP.

Wreszcie warstwa trzecia – aplet uruchamiany na komputerze użytkownika – stanowi interfejs graficzny, z którym współpracuje użytkownik. Interfejs ten pozwala na dokonywanie przez użytkownika konfiguracji mapy, którą widzi, wybór obiektów na mapie różnymi metodami (przez zaznaczenie bezpośrednie, zaznaczenie wielokątem czy zaznaczenie przez podanie warunków dla atrybutów obiektu), uzyskiwanie szczegółowej informacji na temat obiektów, generowanie raportów. Jeśli aplet nie posiada informacji, której żąda użytkownik, wysyła zgłoszenie do serwera WWW i przetwarza otrzymane dane, aby w sposób graficzny zaprezentować je użytkownikowi.

Zastosowany podział oprogramowania na warstwy w istotny sposób upraszcza projektowanie całego systemu, a także jego ewentualne przyszłe aktualizacje.

Oprogramowanie apletu nie musi posiadać żadnej informacji na temat struktury bazy danych przechowującej repozytorium. Jeśli potrzebuje jakiejś informacji, zwraca się do oprogramowania umieszczonego na serwerze WWW, które zajmuje się resztą.

4. Przegląd niektórych mechanizmów autentyfikacji

Użytkownik mający dostęp do repozytorium musi zostać zidentyfikowany tak, aby można było udzielić mu odpowiednich uprawnień. Tak więc przy swoim pierwszym kontakcie z portalem konieczne jest „przedstawienie się" użytkownika. Z reguły użytkownika charakteryzuje jego identyfikator (login) i hasło. Pierwszym problemem staje się przesłanie tej informacji do serwera w sposób, który uniemożliwi (lub przynajmniej znacząco utrudni) przejęcie jej innym użytkownikom sieci.

4.1. Bezpieczeństwo danych identyfikacyjnych

Problem bezpieczeństwa danych udostępnianych w sieci Internet jest w chwili obecnej jednym z najważniejszych wyzwań informatyki. Bez zapewnienia takiego bezpieczeństwa nie można myśleć o poważnych zastosowaniach związanych z finansami czy bezpieczeństwem państwowym. Podstawowym problemem jest zapewnienie pewnej i rzetelnej autentyfikacji użytkowników sieci korzystających z jej zasobów [12].

Istnieje szereg gotowych technologii umożliwiających poprawną identyfikację użytkowników. Technologie te charakteryzuje różny poziom bezpieczeństwa, jak również różny poziom łatwości użytkowania. Najczęściej rozwiązania bezpieczniejsze wymagają tak od projektantów oprogramowania, jak i od użytkowników znacznie więcej zachodu przy konfiguracji, utrzymaniu i bieżącej pracy. Z kolei rozwiązania łatwe do zastosowania są z reguły mniej bezpieczne.

Przesłanie informacji o identyfikatorze i haśle z komputera użytkownika do serwera WWW musi siłą rzeczy odbywać się po ogólnodostępnych łączach. Z tego powodu przesłanie tej informacji w postaci zwykłego pakietu HTTP w sposób oczywisty naraża ją na przechwycenie przez innych [13]. Ktoś, kto przechwycił informacje identyfikacyjne uprawnionego użytkownika, może, podając się za niego, bez problemu korzystać z usług serwera.

Aby temu zapobiec, można przywiązać login użytkownika do konkretnego komputera (a właściwie numeru IP) w sieci. W takim przypadku użytkownik mógłby pracować z systemem tylko ze swojego własnego komputera. Rozwiązanie to nie zawsze jest jednak satysfakcjonujące, ponieważ ogranicza mobilność użytkowników. Poza tym wszelkie zmiany

konfiguracji sieci po stronie użytkownika musiałyby być natychmiast zgłaszane serwerowi. Innym problemem jest możliwość podszywania się intruzów pod przyznany adres IP. Sam komputer użytkownika może zresztą także fizycznie stać w miejscu dostępnym dla innych.

Inna metoda zabezpieczania konta użytkownika to zasada częstej zmiany hasła (nawet przy każdym połączeniu). Jest to jednak metoda wysoce niewygodna dla użytkowników. Spotyka się także na przykład zabezpieczenie przez przesyłanie za jednym razem tylko niektórych liter hasła. W ten sposób przechwycenie pakietu nie daje snifferowi (osobie wyłapującej przesyłane siecią pakiety) pełnej informacji.

Wszystkie te metody mogą utrudnić przejęcie hasła, jednak nie są w stanie przed nim powstrzymać. Jedyną prawdziwą metodą zabezpieczenia danych identyfikacyjnych jest ich zaszyfrowanie na czas przesyłu siecią. Jednak aby szyfrowanie odniosło pożądany skutek, zarówno nadawca, jak i odbiorca muszą znać klucz, według którego zaszyfrowano dane. Jeśli w grę wchodzi przeglądarka internetowa i serwer WWW, oba znajdujące się „gdzieś w Internecie", przed szyfrowaniem konieczna jest wymiana informacji na temat klucza. Ta informacja może zostać przejęta przez sniffera, co powoduje, że może on przechwycić i rozszyfrować przesyłane pakiety – a więc całe szyfrowanie traci sens.

Z tego powodu do szyfrowania informacji w Internecie najczęściej stosuje się tak zwane klucze asymetryczne. Klucze asymetryczne to para kluczy, z których jeden służy do szyfrowania danych, a drugi do ich odszyfrowania. Najważniejszą cechą kluczy asymetrycznych jest to, że posiadając tylko jeden z tych kluczy (na przykład klucz szyfrujący) nie można odgadnąć drugiego klucza z tej pary.

Dzięki zastosowaniu kluczy asymetrycznych możliwa jest bezpieczna szyfrowana komunikacja pomiędzy przeglądarką internetową a serwerem WWW. Najpierw zarówno serwer, jak i przeglądarka generują swoje pary kluczy szyfrujących i deszyfrujących. Następnie wymieniają się kluczami szyfrującymi, tak zwanymi kluczami publicznymi. Od tej chwili wszelka komunikacja pomiędzy obu stronami jest zawsze szyfrowana kluczem publicznym partnera. Przechwycenie przez intruza klucza publicznego nie jest groźne, ponieważ klucz ten nie pozwala na odszyfrowanie danych. Odszyfrować je można jedynie za pomocą klucza prywatnego, który nigdy nie jest przesyłany siecią i zawsze pozostaje w miejscu wygenerowania.

Opisana powyżej strategia jest istotą działania szyfrowanego protokołu HTTPS – rozszerzenia HTTP realizującego automatyczne szyfrowanie przesyłanych siecią danych. Jego zastosowanie w znaczący sposób zwiększa bezpieczeństwo danych.

4.2. Utrzymywanie połączenia z serwerem WWW

Komunikacja z wykorzystaniem standardowego protokołu HTTP (lub HTTPS) nie pozwala użytkownikowi na utrzymanie stałego połączenia z serwerem. Każde kolejne zapytanie użytkownika jest przez serwer traktowane i obsługiwane niezależnie.

W takiej sytuacji, aby utrzymać swoje uprawnienia, użytkownik musiałby z każdym zgłoszeniem przesyłać do serwera po raz kolejny swój identyfikator i hasło. Aby tego uniknąć, w momencie pierwszego pojawienia się użytkownika na serwerze jest dla niego tworzony zapis zwany *sesją*. W zapisie sesji serwer WWW przechowuje dane identyfikacyjne użytkownika oraz adres komputera, z którego dokonał połączenia. Oprócz tego przechowywane mogą być także dodatkowe dane, jak na przykład rodzaj przeglądarki czy system operacyjny użytkownika.

Każda sesja otrzymuje swój unikalny identyfikator. Po uwierzytelnieniu się na serwerze użytkownik otrzymuje identyfikator sesji, pod którym zostały na serwerze zapamiętane jego dane. Od tej chwili w każdym zgłoszeniu użytkownika musi się znaleźć informacja, jakiej sesji zgłoszenie dotyczy. Jeśli identyfikator sesji nie pojawi się jako parametr zgłoszenia, serwer automatycznie przechodzi do autentyfikacji użytkownika.

W trakcie komunikacji z użyciem protokołu HTTP użytkownik może w każdej chwili przerwać połączenie nie powiadamiając o tym serwera. Z tego powodu należy wprowadzić na serwerze parametr określający, po ilu minutach od ostatniego połączenia sesja przestaje być aktywna. Często stosowaną domyślną wartością jest 20 minut.

5. Podsumowanie

W powyższym opracowaniu starano się przeanalizować możliwe schematy współpracy użytkowników z repozytoriami danych. Bazując na tej analizie, przedstawiono proponowaną architekturę portalu RSIP. Wybór każdego rozwiązania poparty został uzasadnieniem i porównaniem go z innymi możliwymi architekturami. Starano się wykazać, że wybrana prototypowa architektura portalu RSIP jest najoptymalniejsza dla określonych warunków jego użytkowania.

LITERATURA

1. Jamsa K., Lalani S.: Programowanie WWW, MIKOM, Warszawa 1997.
2. Castro E.: Po prostu Netscape, Helion, Gliwice 1998.

3. Taylor D.: HTML. Tworzenie stron WWW, Oficyna Wydawnicza READ ME, 1996.
4. The CGI Specification, http://hoohoo.ncsa.uiuc.edu/cgi/interface.html.
5. Genusa S. i inni: „Using ISAPI", QUE, 1997.
6. Kasprowski P.: Zastosowanie mechanizmu Java Servlets do udostępniania zasobów w sieci Internet, ZN Politechniki Śląskiej, seria Informatyka z. 34, 1998.
7. Active Server Pages Guide, http://msdn.microsoft.com/library/ psdk/iisref/aspguide.htm
8. PHP: Hypertext Preporcessor, http://www.php.net.
9. Walter S.J., Weiss A.: Język JavaScript, Intersoftland, Warszawa 1999.
10. ActiveX Controls, http://www.microsoft.com/com/tech/ActiveX.asp.
11. Hoff A., Shaio S., Starbuck O.: Java, Helion, Gliwice 1996.
12. Garfinkel S., Spafford G.: WWW – bezpieczeństwo i handel, Helion, Gliwice 1999.
13. Kasprowski P.: Niezależny sprzętowo dostęp do baz danych z sieci Internet – omówienie możliwości i prezentacja zastosowań, ZN Politechniki Śląskiej, seria Informatyka z. 36. 1999.

Recenzent: Dr inż. Arkadiusz Sochan

Wpłynęło do Redakcji 10 grudnia 2002 r.

Abstract

The paper analyzes different possibilities of presenting data from a data repository to the end-user's machines. This work contains conclusions from RSIP (Regional System of Geographic Information) project which was realized by Silesian University of Technology and ISPIK company.

The simple not-WWW based client application suffers from several disadvantages (architecture dependency, difficulties with updating, lower security level). In contrast a WWW-based application is architecture independent and requires updating only on the server side. Moreover it secures data the repository much better than the previous one (see Fig.1.).

WWW-based applications can have different architectures. The simplest one is a CGI [4] (Common Gateway Interface) application which creates HTML pages as output (Fig. 2.). So called 'active' HTML pages are another example – these are pages with fragments of code written in a programming language. The most popular technologies are ASP [7] and PHP [8].

Programmers can also use Javascript scripts embedded in HTML code. The difference from ASP or PHP is that the Javascript code is run by a browser on a client's machine.

The most sophisticated way of realizing data access from WWW is creating an application which is run in the client's browser. The most popular solutions for this are ActiveX components and Java applets.

The communication with a WWW server is very simple – the user requests for a page and the server sends it. How to maintain a continuous connection between the client and the server using http (or https) protocol is one of the main problems of WWW-based applications. This problem is closely related with an user authentification issue. There are several methods to maintain connections and to authentificate users. It's described in sections 4.1 and 4.2 of the paper.

Adres:

Paweł KASPROWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, kasprowski@polsl.pl.