STUDIA INFORMATICA 2010
Volume 31 Number 2B (90)

Dariusz DOLIWA, Wojciech HORZELSKI
Uniwersytet Łódzki, Wydział Matematyki i Informatyki

ZARZĄDZANIE SPRZĘTEM I OPROGRAMOWANIEM NA ZDALNYCH KOMPUTERACH

Streszczenie. Aplikacja umożliwia inspekcję i inwentaryzację oprogramowania oraz sprzętu dla zdalnych komputerów. Program składa z modułu serwera i klienta. Klient gromadzi informacje o cechach danego komputera, zapisuje je i wysyła do serwera, który zbiera dane od agentów i przechowuje w bazie danych. Ponadto serwer pełni rolę warstwy prezentacyjnej pozwalając na przeglądanie ich poprzez przeglądarkę WWW.

Słowa kluczowe: audyt sprzętu, WMI, Hibernate, sesja TCP

HARDWARE AND SOFTWARE MANAGEMENT ON REMOTE HOSTS

Summary. The application helps to automate the inspection and inventory of hardware and software on remote hosts. The program consists of server and client modules. Client collects information about the characteristics of hardware and software configuration and send it to a server which collects data and stores them into database. Server also acts as a presentation layer for data accessible by WWW browser.

Keywords: hardware audit, WMI, Hibernate, TCP session

1. Funkcjonalność aplikacji

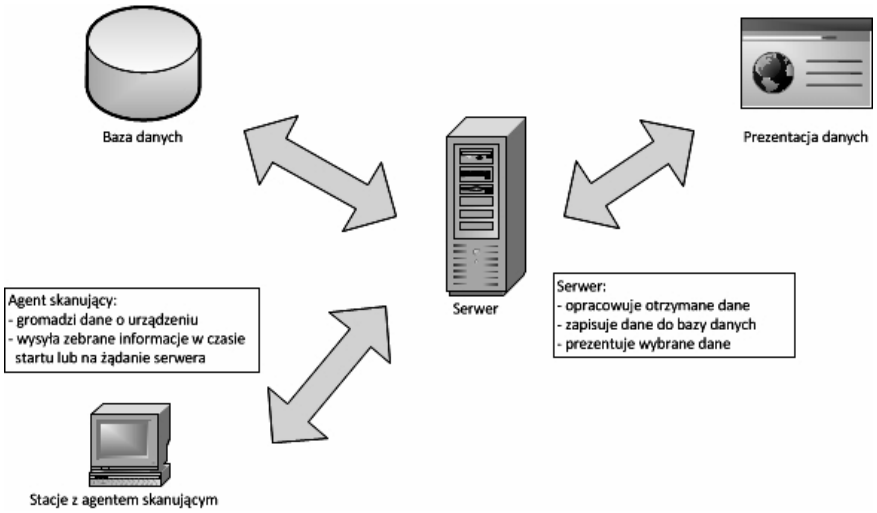
W dużych firmach z wieloma lokalizacjami często problemem jest zebranie i utrzymanie aktualnych informacji o sprzęcie komputerowym oraz zainstalowanych aplikacjach. Zmieniające się i aktualizowane konfiguracje, reorganizacje powodujące fizyczną relokację zasobów komputerowych, modyfikacje sprzętu sprawiają, że łatwo stracić kontrolę nad stanem posiadania. Potrzebny jest system, który zgromadzi wszystkie informacje w jednym miejscu (odpowiednio zaprojektowanej bazie danych), a następnie pozwoli na prezentację tych informacji w przejrzysty i prosty sposób. Zadanie to ma wypełniać aplikacja do przeprowadzania audytu sprzętu i oprogramowania. Celem aplikacji jest automatyczne i zdalne zbieranie informacji o komputerach

pracujących pod kontrolą systemu operacyjnego z rodziny Microsoft Windows połączonych siecią działającą w oparciu o stos protokołów TCP/IP. Aplikacja umożliwia uzyskanie informacji o zainstalowanym w tych komputerach sprzęcie (procesor, pamięć, karta grafiki, dyski, karty sieciowe, napędy CD/DVD) oraz oprogramowaniu (system operacyjny, aplikacje użytkowe). Ponadto pozwala ona na szybkie wyszukanie komputerów według ustalonych kryteriów (np. sprzętu, który spełnia warunki stawiane przez aplikację lub wymaga unowocześnienia). Dzięki aplikacji można również uzyskać aktualne parametry pracy monitorowanych urządzeń (np. zalogowany użytkownik, uruchomione procesy, ilość wolnych zasobów, liczniki wydajnościowe systemu).

2. Architektura systemu

Aplikacja działa poprzez zainstalowanych i uruchamianych automatycznie na komputerach agentów zbierających informacje przy wykorzystaniu mechanizmu WMI (*Windows Management Instrumentation*) i przekazujących je do centralnego serwera. Serwer zapisuje dane w bazie danych oraz pozwala na ich prezentację. Administrator serwera może również zażądać przesłania przez działającego klienta wartości wybranych parametrów.

System składa się komputerów, o których mają być zbierane informacje, serwera monitorującego, bazy danych i aplikacji do wyświetlania wyników.



Rys. 1. Elementy systemu
Fig. 1. System design

Agent skanujący uruchamiany jest automatycznie w momencie startu komputera jako działający w tle proces zbierający za pomocą mechanizmu WMI informacje o danym komputerze. Następnie nawiązuje on sesję z serwerem i przesyła do niego zgromadzone dane w postaci pliku XML. Jednocześnie nadsłuchuje on ewentualnego polecenia od serwera ponownego przeskanowania systemu i wysłania aktualnych wartości parametrów.

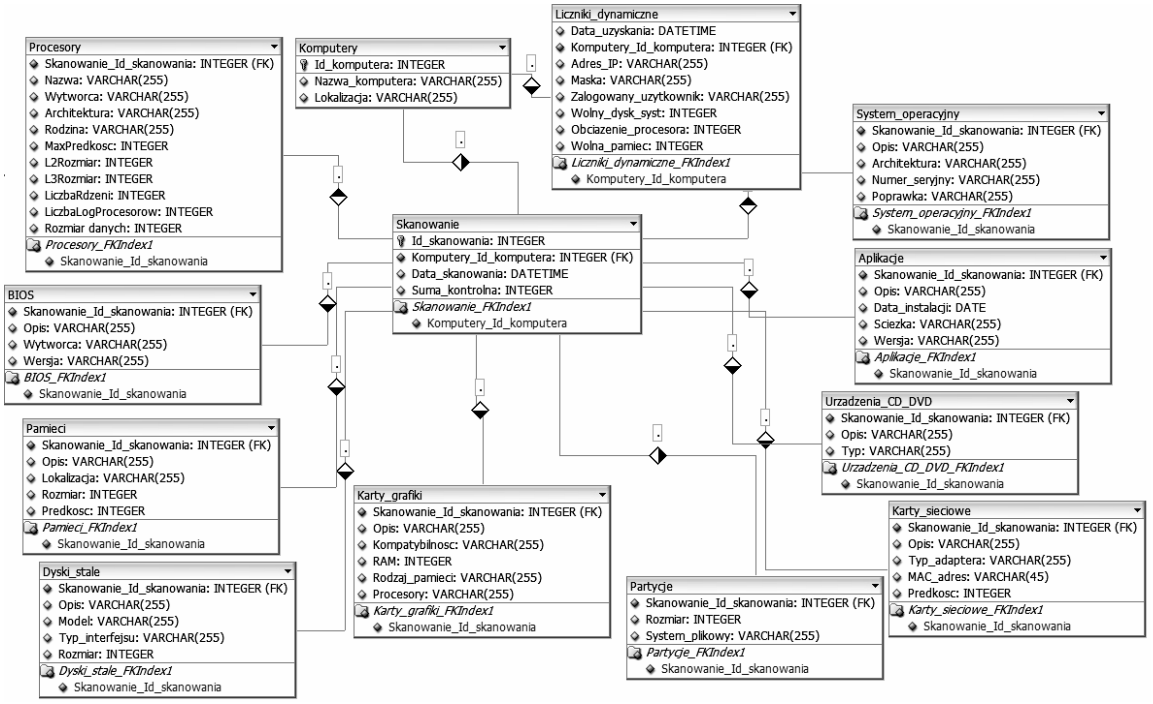
Serwer jest aplikacją wielowątkową (może jednocześnie obsługiwać komunikację wielu klientów). Serwer odbiera wszystkie te połączenia, zarządza nimi, sprawdza poprawność otrzymanych danych oraz je kolejkuje przed zapisem do bazy danych. Komunikacja serwera z bazą danych odbywa się za pomocą biblioteki *Nhibernate*. Zapewnia ona translację danych pomiędzy światem obiektowym (występującym w językach programowania) a relacyjną bazą danych (*ORM – Object-Relational Mapping*) [1 i 6]. Translacja ta opiera się na wykorzystaniu opisu struktury danych za pomocą języka XML (w ten sposób dane będą zapisywane na stacji, która podlega skanowaniu i przesyłane do serwera), dzięki czemu można "rzutować" obiekty na istniejące tabele bazy danych. Ważnym zadaniem modułu serwera jest również pełnienie roli warstwy prezentacyjnej zebranych danych. Z punktu widzenia użytkownika jest to najbardziej znacząca i najważniejsza część modułu. To tutaj bowiem dokonuje się fizycznej inspekcji zasobów nadzorowanych komputerów. To dzięki interakcji z interfejsem modułu serwera nadzorujący może w krótkim czasie przejrzeć zmiany, które dokonały się w konfiguracjach komputerów, zobaczyć, czy użytkownikom wystarcza zasobów takich, jak dyski twarde czy procesory, sprawdzić aktualność oprogramowania. Możliwy jest również dostęp do zgromadzonych w bazie zasobów dzięki klientowi WWW. Dzięki temu administrator może przeglądać zgromadzone zasoby z dowolnego miejsca w sieci.

Komunikacja pomiędzy klientem i serwerem odbywa się za pomocą specjalnie do tego stworzonego protokołu warstwy aplikacyjnej. Protokół ten definiuje sposób i format przesyłanych informacji od klienta do serwera oraz żądania generowane przez serwer do klienta, gdy chce on pobrać aktualną informację o jego zasobach. Na warstwie transportowej protokół ten używa protokołu TCP, który zapewnia mu pewny i stabilny transfer informacji. Po nawiązaniu sesji TCP pomiędzy klientem a serwerem (na serwerze do obsługi tej sesji tworzony jest dedykowany wątek) wysyłany jest komunikat HELLO z identyfikatorem komputera klienta oraz hasło dostępu. W odpowiedzi serwer odsyła komunikat STATUS z kodem odpowiedzi. Jeżeli kod odpowiedzi ma wartość różną od OK., to oznacza to, że serwer nie jest gotowy na odbiór pliku (np. w przypadku niezgodności haseł lub braku miejsca na dysku do składowania pliku), wtedy klient zapisuje informacje o błędzie do rejestru zdarzeń aplikacji. W sytuacji pozytywnej odpowiedzi od serwera klient umieszcza w strumieniu danych wysyłanych do serwera dane z pliku opisującego jego zasoby. Na koniec klient wysyła komunikat END i oczekuje od serwera potwierdzenia zakończenia komunikacji (komunikatu STATUS OK). W sytuacji, gdy serwer chce zainicjować ponowne przesłanie danych od klienta, zdefiniowany jest komunikat SCANREQUEST, który powoduje, że klient przeprowadza nowe skanowanie zasobów i jego wynik przesyła do serwera według schematu opisanego wyżej. Poprawność komunikacji (sekwencyjność przesyłanych danych, pewność ich dostarczenia poprzez mechanizmy potwierdzeń i retransmisji) jest zagwarantowana przez protokół TCP i nie musi być implementowana na warstwie aplikacyjnej [7].

Baza danych przechowuje zgromadzone dane otrzymane od serwera i pozwala administratorowi na wyszukiwanie i wyświetlanie zgromadzonych w niej informacji. W bazie danych będą również definiowane pewne wartości progowe (np. minimalna ilość wolnego miejsca na dysku) i jeżeli podczas skanowania systemu na określonym komputerze zostaną one przekroczone, będzie wygenerowane ostrzeżenie. W aplikacji bazę danych zrealizowano w oparciu o system bazodanowy MySQL. Ze względu jednak na użycie biblioteki *Nhibernate* może ona być łatwo zastąpiona innym systemem bazodanowym bez zmiany kodu samej aplikacji.

3. Struktura bazy danych

Baza danych składa się tabel zawierających dane o klientach (przeskanowanych komputerach). Tabele *Komputery*, *Procesory*, *Pamieci*, *Karty_grafiki*, *Karty_sieciowe*, *Dyski_stale*, *Urządzenia_CD_DVD* zawierają informacje o komponentach sprzętowych klientów, tabela *System_operacyjny* przechowuje dane o systemie operacyjnym (wersji, zainstalowanych poprawkach, języku oraz dacie instalacji), a tabela *Aplikacje* o programach użytkowych. Wartości w tych tabelach mają charakter statyczny w tym sensie, że zmieniają się rzadko, a ich zmiana wymusza ponowne uruchomienie agenta skanującego. Dane w tych tabelach są zapisywane tylko wtedy, gdy w momencie skanowania zmieniła się wartość ktoregoś z pól w nich zdefiniowanych (w tym celu z każdym skanowaniem związana jest suma kontrolna, która zmienia swoją wartość w przypadku zmiany któregoś z komponentów). Tabela *Liczniki_dynamiczne* definiuje parametry często zmieniające się, o których możemy chcieć uzyskać informacje od komputera klienta. Ponieważ każda zmiana w konfiguracji sprzętowej jest istotna, dane statyczne są zapisywane na stałe i nigdy nie są samoistnie usuwane. Zapis danych dynamicznych odbywa się przy każdym żądaniu. W przeciwieństwie do danych statycznych archiwizacja danych dynamicznych nie ma wiele sensu. Stąd też implementacja mechanizmu, który dla każdego klienta utrzymuje w bazie tylko najnowsze inspekcje. Dane starsze niż zdefiniowany w konfiguracji serwera przedział czasu są automatycznie usuwane.



Rys. 2. Struktura bazy danych
Fig. 2. Database structure

4. Aplikacja skanująca

Moduł klienta zbiera informacje na temat komputera, na którym go uruchomiono. Jest zainstalowany na każdym komputerze, który ma być objęty skanowaniem. Został on zaimplementowany w języku C# jako usługa systemu Windows, działająca w tle, startująca automatycznie podczas uruchamiania systemu operacyjnego. Nie generuje żadnych komunikatów dla użytkownika, wszystkie informacje zapisuje w stworzonym dla niego rejestrze zdarzeń. Podczas instalacji administrator wprowadza następujące dane: unikalną nazwę i lokalizację komputera, oraz wartość portu protokołu TCP, na którym program będzie nadsłuchiwał żądań od serwera (domyślny 8888), ponadto podawany adres i port, na którym nadsłuchuje serwer (domyślnie 8887) oraz port, na który klient ma wysyłać informacje o swoich zasobach podczas każdego uruchamiania. Generowany jest wtedy również za pomocą algorytmu UUID (*Universally Unique Identifier*) unikalny identyfikator komputera. Wszystkie te informacje są zapisywane do pliku konfiguracyjnego.

W czasie startu proces serwera przeprowadza wstępne skanowanie systemu korzystając usługi WMI [2]. Usługa WMI jest opracowaną przez firmę Microsoft implementacją standardu WBEM (*Web-Based Enterprised Management*) – zbioru technologii opracowanych w celu ujednolicenia zarządzania środowiskami IT. WMI pozwala na dostęp do zasobów systemowych jednocześnie ukrywając złożoność tego dostępu pod warstwami abstrakcji. Komunikacja opiera

się na schemacie dostawca – odbiorca, gdzie w tej pierwszej roli występują komponenty udostępniające informacje administracyjne, natomiast z drugiej strony odpowiednio skonstruowane aplikacje, w przypadku opisywanej aplikacji – agent skanujący. Język wybrany do implementacji klienta skanującego (C#) ma wbudowane klasy do obsługi WMI, dzięki czemu dostęp do informacji o sprzęcie i oprogramowaniu jest stosunkowo nieskomplikowany [3 i 4].

Przykładowa funkcja pozwalająca na uzyskanie danych o procesorze wygląda następująco:

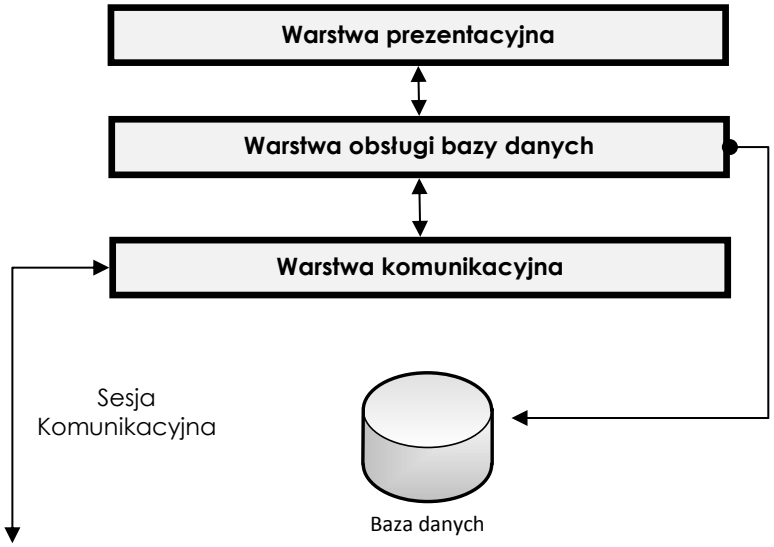
```
using System;
using System.Management;
using System.Windows.Forms;
using System.IO;
namespace WMISample
{
public class MyWMIQuery
{
public static void Main()
{
try
{
ManagementObjectSearcher searcher =
new ManagementObjectSearcher("root\\CIMV2","SELECT * FROM Win32_Processor");
FileInfo t = new FileInfo("C:\\Temp\\dane.xml");
StreamWriter plik =t.CreateText();
foreach (ManagementObject queryObj in searcher.Get())
{
plik.WriteLine("<Nazwa>{0}</Nazwa>", queryObj["Name"]);
plik.WriteLine("<Archit>{0}</Archit>", queryObj["Architecture"]);
plik.WriteLine("<Rodzina>{0}</Rodzina>", queryObj["Family"]);
plik.WriteLine("<Taktowanie>{0}</Taktowanie>", queryObj["MaxClockSpeed"]);
plik.WriteLine("<RozmiarL2>{0}</RozmiarL2>", queryObj["L2CacheSize"]);
plik.WriteLine("<Rozmiar_danych>{0}</Rozmiar_danych>", queryObj["DataWidth"]);
plik.Close();
}
}
catch (ManagementException e)
{
MessageBox.Show("An error occurred while querying for WMI data: " + e.Message);
}
}
}
}
```

W taki sposób w omawianej aplikacji dane będą zapisywane do pliku *dane.xml* opisującego dany komputer w formacie XML. Następnie klient ustanawia sesję TCP z serwerem i przesyła do niego ten plik. Gdy serwer potwierdzi otrzymanie pliku, klient zapisuje w pliku rejestru zdarzeń informację o sukcesie tej operacji.

Na kliencie uruchamiany jest również rezydentny proces nasłuchujący, w nieskończonej pętli, na skonfigurowanym porcie TCP żądań od serwera (komendy uzyskania danych opisanych w tabeli *Liczniki_dynamiczne*). Proces ten pozwala uzyskać administratorowi aktualne informacje o stanie systemu.

5. Aplikacja serwera

Komputer, na którym działa aplikacja serwera jest odpowiedzialny za wiele funkcji. Główną rolą serwera jest ciągły nasłuch i oczekiwanie na połączenia dokonywane ze strony modułów klienckich. Drugą istotną rolą tego programu jest trwałe zapisywanie danych przysyłanych przez klientów do bazy danych. Trzecim zadaniem modułu serwera jest pełnienie roli warstwy prezentacyjnej zebranych danych. Jest on więc pośrednikiem pomiędzy wszystkimi elementami systemu. Podobnie jak aplikacja klienta, został on napisany w języku C#.



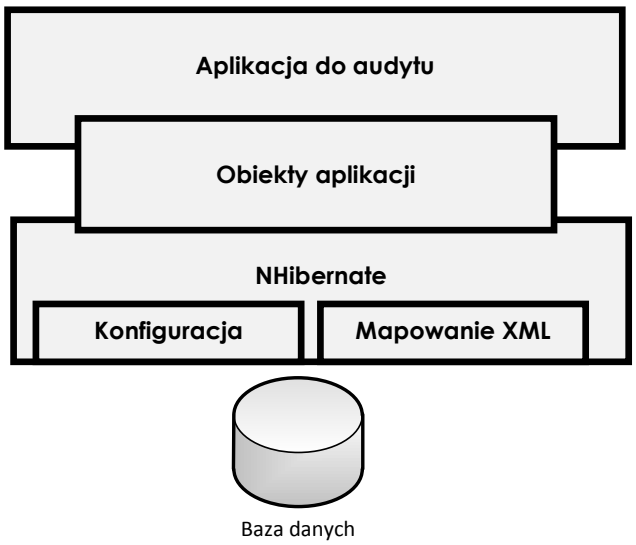
Rys. 3. Zadania aplikacji serwera
Fig. 3. Server tasks

5.1. Warstwa komunikacyjna

Komunikacja serwera z klientem odbywa się poprzez dwa oddzielne procesy. Pierwszy z nich jest wielowątkowym procesem nadsłuchującym na zdefiniowanym porcie TCP połączeń od klientów. Po nawiązaniu sesji komunikacyjnej przez klienta przesyła on do serwera dwa rodzaje informacji: o swojej aktualnej konfiguracji (adres IP, port, przez który serwer może kontaktować się z klientem) oraz plik *data.xml* opisujący zasoby klienta. Następnie przy użyciu specjalnej funkcji plik z danymi jest wczytywany i sprawdzany pod względem poprawności składniowej. Jeżeli nie zostanie zgłoszony błąd, funkcja wczytująca przypisze do odpowiednich obiektów pobrane z pliku wartości. W przypadku wykrycia błędu wysyłana jest komenda ponownego przeskanowania systemu i przesłania pliku z danymi. Drugi proces pozwala serwerowi na inicjowanie komunikacji z klientem w celu przesłania żądania aktualnej konfiguracji lub uzyskania wartości parametrów dynamicznych.

5.2. Warstwa obsługi bazy danych

Po otrzymaniu danych od klienta są one zapisywane do bazy danych. Do komunikacji z bazą MySQL wykorzystany jest obiektowy interfejs mapowania realizowany przez bibliotekę *NHibernate*. *NHibernate* jest przykładem oprogramowania typu „*object-to-relational mapping*" i przenosi na wyższy poziom abstrakcji relacyjny schemat danych przechowywanych w bazie, wprowadzając schemat koncepcyjny. Pozwala to programiście skupić się na samych danych jako modelu obiektowym. Dzięki temu podejściu aplikacja może traktować bazę danych i jej zasoby jako integralne obiekty [5 i 6]. W ten sposób obiekty wczytane z pliku przesłanego przez klienta będą zapisywane do bazy danych. Ustanawia powiązania pomiędzy danymi z bazy a obiektami zdefiniowanymi w środowisku .NET. W celu prawidłowej interpretacji danych *NHibernate* potrzebuje informacji, w jaki sposób dokonać skojarzenia stworzonych przez programowanie obiektów do danych z bazy. W omawianej aplikacji do mapowania wykorzystany jest specjalny plik w formacie XML definiujący związki między obiektami i tabelami. Dodatkowo plik konfiguracyjny zawiera ustawienia połączenia bazy danych, dialekt bazy itp. (zmiana tego pliku pozwala na prostą zmianę systemu bazodanowego używanego w aplikacji). Biblioteka *NHibernate* jest również używana do pobierania danych z tabel celu ich prezentacji.

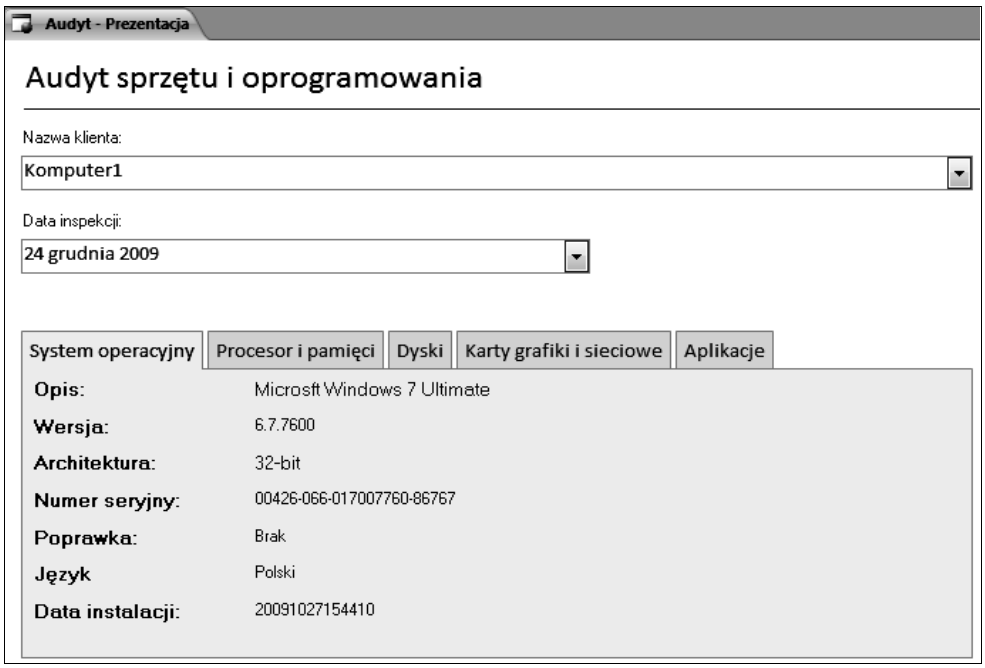


Rys. 4. Mechanizm zapisu do bazy
Fig. 4. Database access

5.3. Warstwa prezentacji danych

Wszystkie zgromadzone w bazie informacje w każdej chwili można odczytać wykorzystując do tego warstwę prezentacyjną aplikacji. Warstwa ta posiada specjalnie stworzony interfejs użytkownika, dzięki któremu w przejrzysty sposób można kontrolować, które dane będą odczytywane z bazy.

Dostęp do danych może odbywać się poprzez dedykowaną aplikację wyświetlającą informacje o wybranym komputerze. Pozwala ona na przeglądanie określonej inspekcji komputera. Informacje podzielone są na sekcje w zależności od rodzaju sprzętu, o którym chcemy uzyskać dane. Odpowiednie zakładki pozwalają wyświetlić dane o systemie operacyjnym, procesorze, pamięci i BIOS-ie, dyskach, kartach graficznych i sieciowych oraz o zainstalowanych aplikacjach.



Rys. 5. Interfejs dostępu do danych
Fig. 5. User interface

Istnieje również możliwość dotarcia do tych danych z poziomu przeglądarki WWW, co pozwala administratorowi na uzyskanie informacji z dowolnego miejsca w sieci.

Zaletą dedykowanej aplikacji jest też możliwość dostępu do danych dynamicznych lub wymuszenie uaktualnienia skanowania.

6. Wnioski

Przedstawiona w artykule aplikacja została wdrożona w wyższej uczelni do zarządzania konfiguracją sprzętową i programową około 300 stacji komputerowych (pracownie studenckie, komputery pracowników, sprzęt w zdalnych lokalizacjach). Zastąpiła stosowaną wcześniej aplikację o charakterze zamkniętym zbierającą i zapisującą dane w wewnętrznym formacie i mającą niewielką możliwość rozbudowy. Architektura, w której pracuje aplikacja, jest pozbawiona tych wad. Dzięki zastosowaniu biblioteki *nHibernate* aplikacja współpracuje bez konieczności zmian z dowolnym systemem bazodanowym, dobranym według potrzeb użytkownika aplikacji (w zależności od ilości zbieranej informacji). Poprzez zastosowanie systemu WMI i ukierunkowanie

aplikacji na gromadzenie informacji o komputerach opartych na systemie operacyjnym Windows – pozwala ona na pobieranie większej gamy informacji niż standardowe metody monitorowania urządzeń poprzez sieć. Sposób działania aplikacji nie obciąża w sposób ciągły stacji roboczych, na których jest ona zainstalowana, gdyż skanowanie systemu odbywa się na żądanie administratora. Ograniczony jest również ruch generowany w sieci, gdyż nie jest potrzebne stałe utrzymywania połączenia pomiędzy serwerem a skanowanymi klientami.

BIBLIOGRAFIA

1. O'Neil E.: Object/relational mapping 2008: hibernate and the entity data model (edm). Proceedings of the 2008 ACM SIGMOD international conference on Management of data, Vancouver 2008, s. 1351÷1356.
2. Lavy M., Meggitt A.: Windows Management Instrumentation, New Riders, 2001.
3. Tunstall C., Cole G.: Developing WMI Solutions: A Guide to Windows Management Instrumentation. Addison-Wesley Professional, 2002.
4. Blum R.: C# Network Programming. Sybex, 2003.
5. Kuaté P. H., Bauer Ch., King G., Harris T.: NHibernate in Action. Manning Publications, 2009.
6. NHibernate. 2008. NHibernate Reference Manual. http://hibernate.org .
7. Postel J. B.: Transmission Control Protocol (RFC 793), 1981.

Recenzenci: Dr inż. Aleksandra Werner
Dr inż. Hafed Zghidi

Wpłynęło do Redakcji 13 stycznia 2010 r.

Abstract

The application described in this paper helps to automate the inspection and inventory of hardware and software on remote hosts. It allows quick search computers for specific characteristics and monitor dynamic parameters. The program consists of two separate modules: the server module and client module (fig. 1), which form together a complete information exchange system. Client program is installed on each computer collects information about the characteristics of the host computer hardware and software configuration and send it to a central server. Server collects data from agents and stores them into the database (fig. 3). Additionally, the server acts as a

presentation layer allowing to browse the data collected from a dedicated application or generate dynamic web pages accessible via the World Wide Web browser (fig. 5).

Adresy

Dariusz DOLIWA: Uniwersytet Łódzki, Wydział Matematyki i Informatyki, ul. Banacha 22, 90-238 Łódź, Polska, doliwa@imul.uni.lodz.pl.
Wojciech HORZELSKI: Uniwersytet Łódzki, Wydział Matematyki i Informatyki, ul. Banacha 22, 90-238 Łódź, Polska, horzel@imul.uni.lodz.pl.