STUDIA INFORMATICA 2014
Volume 35 Number 2 (116)

Robert BRZESKI, Paweł WRÓBEL
Politechnika Śląska, Instytut Informatyki

# MOŻLIWOŚĆ ODCZYTU I UMIESZCZANIA WIADOMOŚCI SMS NA STRONIE WWW

**Streszczenie**. Artykuł przedstawia opis aplikacji i zastosowanych technologii umożliwiających przesył wiadomości SMS pomiędzy urządzeniem GSM a komputerem oraz zapis odczytanych wiadomości do bazy danych i udostępnianie ich w aplikacjach internetowych.

**Słowa kluczowe**: SMS, Hayes, PDU, GSM, WWW, wiadomość tekstowa, GSMComm, SerialPort, SOAP, HTTP POST, baza danych, XML

# THE POSSIBILITY OF THE READING AND PLACING THE SMS MESSAGE ON THE WEB PAGE

**Summary**. The article presents the description of technologies enabling the sending of the SMS message between computer and the GSM device, and also the record of read messages to the database and making accessible of massages for internet applications.

**Keywords**: SMS, Hayes, PDU, GSM, WWW, text message, GSMComm, SerialPort, SOAP, HTTP POST, database, XML

## 1. Wstęp

Ostatnie kilkanaście lat to wciąż rosnąca popularność takich dwu technologii jak internetowe strony WWW oraz przesyłanie wiadomości SMS (ang. *Short Message Service*) [1]. Rozpowszechnienie obu tych technologii spowodowało, że zaczęto je ze sobą łączyć. W wyniku tego powstały i zyskały dużą popularność m.in. serwisy WWW umożliwiające z poziomu strony internetowej wysłanie wiadomości na telefon komórkowy. Zdecydowanie mniejsza popularność dotyczy przesyłu wiadomości w odwrotnym kierunku, dlatego powsta-

ło zagadnienie, w jaki sposób wysłać wiadomość SMS na stronę WWW. Należy mieć na uwadze zalety przesyłania wiadomości SMS za pomocą telefonów komórkowych. Wysrępują tutaj niskie koszty przesyłu wiadomości SMS, ogromny zasięg usługi, dostępność telefonów komórkowych oraz łatwość obsługi i popularność przesyłu informacji przy wykorzystaniu wiadomości SMS [2, 3].

Zagadnienie przesyłu wiadomości SMS na stronę WWW podzielono na 2 etapy:

- odczyt wiadomości z urządzenia (modem GSM, telefon komórkowy).
- umieszczenie wiadomości na stronie WWW.

## 2. Narzędzia do odczytu wiadomości SMS

W sieci WWW można znaleźć aplikacje do odczytu wiadomości SMS bezpośrednio z modemu GSM czy telefonu komórkowego. Najbardziej popularne projekty to: Gnokii [4], Gammu, Wammu [5], Kannel [6], SMPPSim [7], SMSLib [8], Inetlab.SMPP [9], GSMComm [10]. Ponieważ większa część przedstawionych narzędzi wymaga wykupienia dostępu do centrum SMS lub oferuje niewygodny sposób udostępnienia odczytanej wiadomości do dalszego jej przetwarzania, zdecydowano się na stworzenie od podstaw aplikacji „SMSHost". Zrealizowana aplikacja realizuje odczyt wiadomości SMS na kilka różnych sposobów, w zależności od zawartości pliku konfiguracyjnego. Aplikacja magazynuje także wiadomości w taki sposób, aby było możliwe odczytanie ich za pomocą strony WWW lub klasycznej aplikacji typu „desktop".

Aplikacja została podzielona na 3 główne przestrzenie nazw:

- SMSDataSource – zawiera m.in. klasy odpowiedzialne za odczyt wiadomości z różnych źródeł,
- WcfServiceLibrary – zawiera klasy odpowiedzialne za udostępnianie wiadomości za pośrednictwem technologii WCF (ang. *Windows Communication Foundation*) [11], komunikację z serwerem bazodanowym MS SQL Server, klasy odpowiedzialne za zapis i odczyt wiadomości oraz klasy zapisu logów aplikacji i klasy odpowiadające za odczyt wartości zapisanych w pliku konfiguracyjnym.
- SMSLib – biblioteka zawierająca klasy pomocnicze służące do konwersji wiadomości w formie łańcuchów znakowych do obiektów typu „SMSMessage" (obiekt klasy przechowującej informacje o przesłanej wiadomości SMS).

Ogólny schemat działania aplikacji „SMSHost" jest zobrazowany na diagramie czynności (rys. 1). Przedstawia on najważniejsze czynności wykonywane w pętli głównej programu.


Rys. 1. Pętla główna aplikacji SMSHost – ogólny schemat działania
Fig. 1. Main loop of the application SMSHost – the general chart of working

Po uruchomieniu aplikacji realizuje ona zadania w sposób określony w pliku konfiguracyjnym. Pierwszym działaniem jest włączenie samodzielnego hostingu (ang. s*elf hosting*) WCF [12] – procesu odpowiedzialnego za udostępnianie aplikacjom zewnętrznym odpowiednich usług (ang. *services*). Następnie realizowane są: odczyt wiadomości (rozdział 3) z wybranego źródła i konwersja łańcuchów znakowych na obiekty wiadomości typu SMSMessage. Ostatnią czynnością jest zapisanie wiadomości (rozdział 4) w magazynie (do bazy danych lub do pliku). Aplikacja SMSHost, oprócz odczytu wiadomości, zapewnia również funkcjonalność udostępnienia ich innym aplikacjom.

## 3. Odczyt danych – przesłanych wiadomości SMS

Wiadomości SMS można pobrać z urządzenia GSM (telefonu komórkowego) podłączonego do komputera przez port szeregowy USB (rozdziały 3.1 i 3.2), z wykorzystaniem protokołu „Hayes" [13] i jego „komend AT" [14, 15], lub przez sieć internetową od dostawcy takiej usługi (płatnej) (rozdział 3.3).

### 3.1. Odczyt danych z wykorzystaniem składowych biblioteki GSMComm

Pierwszą z dostępnych strategii odczytu wiadomości jest odczyt (listing 1) przy wykorzystaniu darmowej biblioteki .NET [16] o nazwie GSMComm [10]. Po połączeniu z modemem GSM przez utworzenie obiektu typu GSMCommMain w celu uzyskania dostępu do metod umożliwiających przesyłanie komend AT konieczne jest utworzenie obiektu typu IProtocol przez wywołanie metody GSMCommMain.GetProtocol(). Następnie z wykorzystaniem me-

tody IProtocol.ExecAndReceiveAnything, wysyłane są kolejno komendy protokołu Hayes: „AT”, „AT+CMGF=1”, AT+CMGR=1” oraz „AT+CMGD=1”. Celami tych komend są: sprawdzenie połączenia z urządzeniem, przełączenie urządzenia w tryb tekstowy, odczytanie wiadomości zapisanej w miejscu pamięci o numerze pierwszym i usunięcie tej wiadomości z pamięci telefonu. Ta czynność jest wykonywana tyle razy, ile jest komórek pamięci (w przykładzie: 300 razy). Jeżeli w odpowiedzi uzyskamy wyjątek typu MessageServiceErrorException, najprawdopodobniej oznacza to, że miejsce w pamięci, z którego próbowano odczytać wiadomość, jest puste. Wówczas jako wartość odczytywaną uznaje się kod błędu nr 321, który jest następnie wskazówką dla funkcji odpowiedzialnej za konwersję łańcucha tekstowego do obiektu, że ów łańcuch podczas konwersji powinien być pominięty.

Należy mieć tutaj na uwadze, że aby przedstawiony sposób odczytu działał, urządzenie GSM musi obsługiwać standardowe polecenia protokołu Hayes obsługujące przesył wiadomości SMS (niektórzy producenci wprowadzają własne wersje komend AT).

```
using GsmComm.GsmCommunication;
namespace SMSClient
{
    class GsmCommDataSource:IDataSource
    {
public List<string> Read()
        {
            List<string> messages = new List<string>();
            var comm =
      new GsmCommMain(ConfigReaderSingleton.Instance.GetPort(), 19200, 300);
            comm.Open();
            IProtocol protocolLevel = comm.GetProtocol();
            for (int i = 1; i < 300; i++)
            {
                string input=string.Empty;
                try
                {
                    input =   protocolLevel.ExecAndReceiveAnything("AT","");
                    input+=   protocolLevel.ExecAndReceiveAnything("AT+CMGF=1",
                              "");
                    Thread.Sleep(200);
                    input +=
                 protocolLevel.ExecAndReceiveAnything("AT+CMGR=" + i, "");
                    protocolLevel.ExecAndReceiveAnything("AT+CMGD=" + i, "");
                    Console.WriteLine(input);
                }
                catch (MessageServiceErrorException)
                {
                    input += ErrorConstants.CMS_ERR321;
                }

                messages.Add(input);
            }
            return messages;
        }
```

Listing 1. Odczyt wiadomości przy użyciu biblioteki GSMComm

### 3.2. Odczyt wiadomości przy użyciu klasy SerialPort i protokołu Hayes

Drugą metodą odczytu wiadomości SMS (listing 2) jest wykorzystanie klasy SerialPort [17] dostępnej w środowisku .NET i reprezentującej w aplikacji port szeregowy komputera, do którego podłączone są urządzenia, np. telefon lub modem GSM.

```
private string ReadOneMessage(int i)
        {
            try
            {
                if (port.IsOpen)
                {
          port.DiscardOutBuffer();
                    port.DiscardInBuffer();
                    port.WriteLine("AT\r");
                    Thread.Sleep(200);
                    port.WriteLine("AT+CMGF=1\r");
                    Thread.Sleep(200);
                    Console.WriteLine
                    (string.Format("Sending command: AT+CMGR={0}\r", i));
                    port.Write(string.Format("AT+CMGR={0}\r", i));
                    Thread.Sleep(200);
                    string msg = port.ReadExisting();
                    Console.WriteLine(msg);
                    Console.WriteLine
                    (string.Format("Sending command: AT+CMGD={0}\r", i));

                    port.Write(string.Format("AT+CMGD={0}\r", i));
                    Thread.Sleep(200);
                    string delMsg = port.ReadExisting();
                    Console.WriteLine("DELETE RESPONSE: "+delMsg);


                    return msg;
                }
                else return string.Empty;
            }
            catch (Exception ex)
            {
    [...]
            }
        }
```

Listing 2. Odczyt wiadomości z telefonu komórkowego za pomocą klasy SerialPort i komend standardu Hayes

Podobnie jak w przypadku biblioteki GSMComm (rozdział 3.1) tutaj również wysyłane są kolejno komendy „AT”, „AT+CMGF”, „AT+CMGR” oraz „AT+CMGD”. Po wysłaniu każdego polecenia konieczne jest uśpienie wątku (metoda Thread.Sleep), aby dać czas urządzeniu na przetworzenie polecenia i wygenerowanie odpowiedzi.

Inną zaimplementowaną metodą odczytu wiadomości opartą na klasie SerialPort jest strategia zawarta w klasie SerialPortPDUDataSource. Pozwala ona na odczyt wiadomości w formacie PDU (ang. *Protocol Discription Unit*) [18] (różnica w implementacji względem listingu 2: argument polecenia „AT+CMGF” dla trybu PDU powinien być równy 0).

### 3.3. Odczyt z serwerów dostawcy usługi odbioru wiadomości

Opisane w tym podrozdziale metody odczytu wiadomości wiążą się z koniecznością wykupienia odpowiedniej usługi w zewnętrznej firmie. Wiadomości po odebraniu przez zewnętrzny serwer mogą być również w nim magazynowane, a sama firma dostarcza odpowiednie mechanizmy udostępniania wiadomości. W ten sposób można udostępniać wiadomości SMS wprost na stronie WWW, bez udziału aplikacji SMSHost.

#### *3.3.1. Odczyt za pośrednictwem SOAP*

Odczyt za pośrednictwem protokołu SOAP [19] (ang. *Simple Object Access Protocol* – protokół prostego dostępu do obiektów). Protokół umożliwia przesyłanie obiektów pomiędzy różnymi platformami programistycznymi za pomocą dokumentów XML (ang. *Extensible Markup Language*) [20, 21] i protokołu HTTP [22]. Odczyt wiadomości SMS polega na wywoływaniu odpowiednich metod usługi sieciowej udostępnianej przez dostawcę usługi.

#### *3.3.2. Odczyt za pośrednictwem protokołu HTTP*

Odczyt za pośrednictwem HTTP jest realizowany przez wywołanie odpowiedniego adresu URL (ang. *Uniform Resource Locator*) dostawcy serwisu, tj. „promosms.pl”. W adresie należy przekazać dane autentykacyjne. W rezultacie otrzymywany jest dokument XML zawierający wiadomości SMS.

#### *3.3.3. Odczyt danych metodą HTTP POST*

Inną możliwością odczytu danych, jaką oferuje serwis smsapi.pl, jest użycie metody HTTP POST [23]. Aby odczytać wiadomości, należy na stronie WWW dostawcy zdefiniować adres URL własnego skryptu umieszczonego na swoim serwerze, do którego wiadomość SMS zostanie przekazana jako tablica POST języka PHP [24]. Skrypt ten będzie wywoływany po każdym odebraniu wiadomości. Po zakończeniu działania skrypt musi zwrócić łańcuch „OK”, co jest sygnałem dla serwisu dostawcy, że wiadomość została odebrana (listing 3).

```
<?php
$nadawca = $_POST['sms_from'];
$tekst = $_POST['sms_text'];
[...]
echo "OK"
>
```

Listing 3. Przykładowy skrypt PHP wywoływany przez serwis smsapi.pl

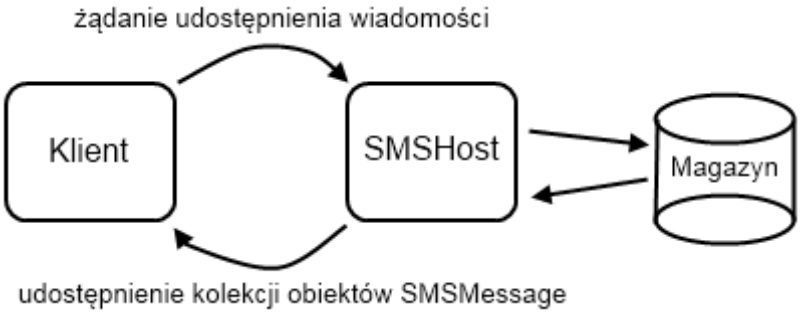
## 4. Składowanie i udostępnianie zapisanych wiadomości SMS

Kolejnym zadaniem po odczytaniu wiadomości SMS było ich składowanie i udostępnianie innym aplikacjom. Rozwiązaniem może być zapisywanie wiadomości do bazy danych i udostępnianie ich bezpośrednio z niej. Pomimo iż jest to podejście najprostsze, niesie ze sobą spore zagrożenia i ograniczenia.

- W przypadku użycia statycznych stron internetowych, opartych wyłącznie na dokumentach HTML i języku JavaScript [25], bezpośrednie połączenie do bazy danych jest rozwiązaniem nieodpowiednim ze względu bezpieczeństwa dostępu do danych. Mianowicie skrypty języka JavaScript są zagnieżdżone w dokumencie HTML i ich wykonanie odbywa się po stronie przeglądarki internetowej klienta i wtedy dane potrzebne do połączenia się z bazą danych również znajdą się w jawnej dostępnej formie [26].
- W przypadku wykorzystania stron w technologii Adobe Flash [27] czy Microsoft Silverlight [28] bezpośredni dostęp jest albo niebezpieczny z podobnych względów co w przypadku skryptów JavaScript bądź jest niedostępny z powodu ograniczeń wynikających z działania aplikacji w zamkniętym środowisku „piaskownicy" (ang. *sandbox*).

Dlatego aby wyeliminować przedstawione ograniczenia, udostępnianie wiadomości zrealizowano w architekturze SOA (ang. *Service-Oriented Architecture*) [29] oraz technologii WCF [11]. Proces odczytu wiadomości został przedstawiony na rysunku 2. Klient wywołuje metody programu SMSHost, prosząc w ten sposób o pobranie wiadomości z magazynu (odczyt i rodzaj magazynu są zdefiniowane w pliku konfiguracyjnym). Lista obiektów zawierających wiadomości SMS (SMSMessage) jest następnie zwracana klientowi.

Magazynem może być plik zapisany na dysku twardym komputera, na którym pracuje aplikacja SMSHost lub serwer bazy danych.




Rys. 2. Udostępnianie wiadomości
Fig. 2. Making accessible of the messages

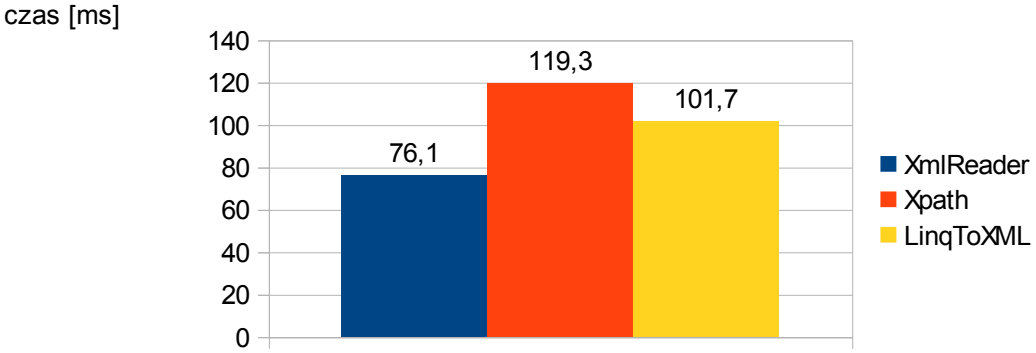
### 4.1. Składowanie i udostępnianie wiadomości z pliku XML

Zastosowanie technologii WCF stworzyło możliwość składowania wiadomości na dysku twardym w pliku XML. Zaletą takiego rozwiązania jest uproszczenie systemu przez wyłączenie z niego serwera bazy danych. W tym rozwiązaniu kolekcja obiektów typu SMSMessage (obiekty zawierające wiadomości SMS) jest dopisywana do pliku messages.xml.

Aby wiadomości mogły być prezentowane na stronie WWW, należy wybrać sposób, w jaki plik XML będzie odczytywany [33, 37]:

- odczyt za pomocą klasy XmlReader [30] – jest to podejście strumieniowe, w którym zawartość pliku jest odczytywana element po elemencie;
- odczyt za pomocą języka XPath (ang. *XML Path Language*) [31] – język XPath pozwala na poruszanie się po drzewie dokumentu XML poprzez nazwy węzłów i atrybutów;
- odczyt za pomocą LinqToXML (ang. *Language Integrated Query To XML*) [32] – LinqToXML jest językiem zapytań wykonywanym na kolekcjach obiektów platformy .NET. Jako kolekcje obiektów wykorzystano kolekcje węzłów dokumentu XML.

Średnie wyniki z dziesięciokrotnego uruchomienia odczytywania wiadomości z pliku XML przedstawiono na rys. 3. Wyników tych nie można traktować jako w pełni miarodajnych testów wydajnościowych. Takie badania wymagałyby m.in. odpowiednio dużych zbiorów danych oraz odpowiedniej liczby powtórzeń. Do ich przeprowadzenia byłby przydatny rzeczywisty działający system z odpowiednio dużym zbiorem danych. Obecnie uzyskane wyniki mogą być podstawą wstępnego oszacowania, dając informacje, jak mogą się kształtować wyniki pełnych badań wydajnościowych.




Rys. 3. Porównanie czasów odczytu danych z pliku XML
Fig. 3. Comparison of times of reading data from XML file

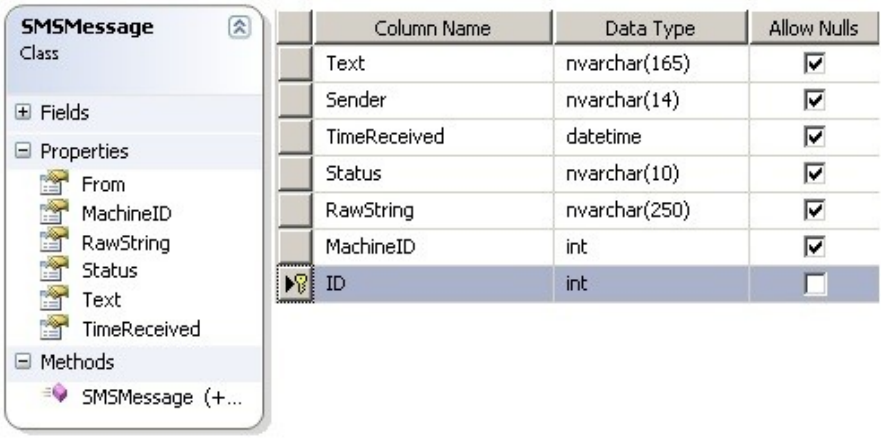
Wyniki wstępnego porównania sugerują, że najszybszym sposobem odczytu wiadomości XML jest wykorzystanie klasy XMLReader. Jej dodatkową zaletą jest fakt, że jako jedyna nie wymaga zapisania całego dokumentu do pamięci podręcznej. Za jej wadę można uznać to, że jej użycie wymusza zastosowanie struktury XML, w której większa część danych jest przechowywana w formie atrybutów, a nie węzłów. Jest to konsekwencją przetwarzania

strumieniowego – analizując dany element, nie mamy informacji, w którym miejscu drzewa dokumentu on się znajduje.

### 4.2. Wykorzystanie bazy danych do składowania i udostępniania wiadomości SMS

Drugą metodą może być składowanie obiektów wiadomości w bazie danych, przy czym mamy tutaj 2 możliwości.

Dokonanie dekompozycji na obiekcie wiadomości – polega na przechowywaniu ich w tabeli „Messages", której kolumny odpowiadają poszczególnym właściwościom obiektu (rys. 4).

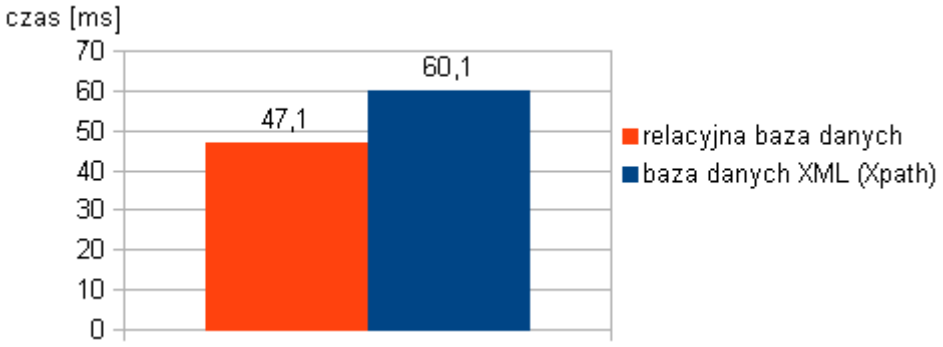



Rys. 4. Klasa „SMSMessage" oraz tabela „Messages"
Fig. 4. Class "SMSMessage" and table "Messages"

Takie rozwiązanie pozwala na „wyszukiwanie w bazie danych wiadomości spełniających dowolne kryteria, jednak przy zmianie struktury obiektu SMSMessage potrzebne będzie zmodyfikowanie zarówno struktury tabeli, jak i kodu programu odpowiadającego za dodanie obiektu do bazy danych.

Druga możliwość to przechowywanie wiadomości w postaci dokumentów XML [34, 35, 36]. Serwer baz danych „MS SQL Server 2008" pozwala na przechowywanie dokumentów XML jako typu danych o tej samej nazwie. Wykorzystując mechanizm serializacji obiektu, można przechowywać wiadomości jako pojedyncze dokumenty XML w bazie danych. Dzięki takiemu rozwiązaniu struktura bazy nie ulegnie zmianie, nawet jeżeli zostanie zmieniona struktura obiektu SMSMessage. Zapytanie SQL powinno wówczas zawierać wyrażenie XPath, które pozwoli określić warunki, jakie musi spełnić dokument.

Średnie wyniki z dziesięciokrotnego uruchomienia wyszukania wiadomości w bazie danych przedstawiono na rys. 5. Ponownie wyników tych nie można traktować jako w pełni miarodajnych testów wydajnościowych z powodu zdecydowanie zbyt małej ilości danych i przeprowadzonych powtórzeń. Można potraktować je jako wstępne oszacowanie dające informacje, jak mogą się kształtować wyniki pełnych badań wydajnościowych.




Rys. 5. Porównanie czasów wykonania wyszukania wiadomości w bazie danych
Fig. 5. Comparison of times of the selecting the messages in database

Otrzymane wyniki pokazują, że wyszukiwanie wiadomości SMS w relacyjnej bazie danych w przeprowadzonym teście jest trochę szybsze w porównaniu z wyszukiwaniem dokumentów XML. Jednak ze względu na niedużą różnicę w czasie wykonania obydwóch zapytań, w połączeniu z opisaną korzyścią wykorzystania typu danych XML, warto zastanowić się nad zastosowaniem drugiej opcji przechowywania wiadomości w bazie danych.

## 5. Podsumowanie – wnioski

Mając możliwość składowania wiadomości SMS w bazie danych oraz udostępniania ich innym aplikacjom, możemy ostatecznie umieszczać je na stronach WWW. W artykule przedstawiono kilka sposobów przesyłania wiadomości SMS na stronę WWW. Całe zadanie zostało podzielone na dwa osobne zagadnienia – odbioru wiadomości SMS (rozdział 3) przez komputer PC oraz składowanie i udostępnianie wiadomości (rozdział 4), np. na internetowych stronach WWW.

W celu wykonania tych zadań została zrealizowana aplikacja „SMSHost”. W ramach pierwszego zagadnienia zaimplementowano funkcjonalność przy użyciu darmowej biblioteki GSMComm (listing 1), a także wersję z własnymi funkcjami z wykorzystaniem standardowych klas platformy .NET (listing 2). W ramach drugiego zagadnienia zrealizowano funkcjonalność udostępniającą aplikacjom klienckim za pośrednictwem technologii WCF metody umożliwiające pobranie listy wiadomości z pliku XML. Zaprezentowano także dwa sposoby zapisywania wiadomości SMS w bazie danych (rys. 5).

Aplikacje pozwalające na odbiór SMS na komputerze PC mogą mieć powszechne zastosowanie. Możliwość wysłania wiadomości SMS jest szeroko dostępna ze względu na niski koszt usługi, dostępny zasięg sieci GSM oraz popularność telefonów komórkowych. Daje to możliwość niemalże natychmiastowego udostępnienia przesłanej wiadomości nieograniczonej lub wybranej grupie odbiorców. Wysłanie wiadomości o odpowiedniej treści pod wska-

zany numer można wykorzystać w celach marketingowych – przez dopisanie numeru telefonu do bazy danych system może być wykorzystany jako platforma do składania zamówień czy potwierdzeń. Dzięki napisaniu aplikacji SMSHost uzyskano wygodny sposób gromadzenia wiadomości i wykorzystywania ich np. w aplikacjach internetowych.

Przedstawiony w artykule opis poszczególnych elementów technologii umożliwiającej dostęp z poziomu komputera PC do wiadomości SMS może stać się dla innych osób nie tylko dużym ułatwieniem w realizacji takiego zadania, lecz także może stanowić inspirację do tworzenia własnych pomysłów zawierających przedstawione rozwiązania.

## BIBLIOGRAFIA


1. SMS: http://en.wikipedia.org/wiki/Short_Message_Service
2. U.S. Wireless Quick Facts: http://www.ctia.org/advocacy/research/index.cfm/aid/10323
3. The end of SMS debate: http://businesstech.co.za/news/mobile/11473/the-end-of-sms-debate/
4. Gnokii: http://gnokii.org/
5. Gammu and Wammu: http://wammu.eu/
6. Kannel: Open source WAP and SMS Gateway: http://www.kannel.org/
7. SMPPSim: http://www.seleniumsoftware.com/downloads.html
8. SMSLib: http://smslib.org/
9. Inetlab.SMPP: http://www.inetlab.com/Products/Inetlab.SMPP.aspx
10. GSMComm: http://www.scampers.org/steve/sms/libraries.htm
11. WCF: http://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx
12. Self Hosting: http://msdn.microsoft.com/en-us/library/ee939340.aspx
13. Hayes AT command set: http://www.activexperts.com/mobile-messaging-component/at/
14. AT Command Set: http://www.olitec.com/pub/commandes_at_usbgprs.pdf
15. Hayes AT command dla GSM http://www.activexperts.com/mobile-messaging-component/at/etsi/
16. .NET: http://msdn.microsoft.com/en-us/library/w0x726c2%28v=vs.110%29.aspx
17. Klasa SerialPort: http://msdn.microsoft.com/pl-pl/library/system.io.ports.serialport%28v=vs.110%29.aspx
18. SMS and PDU format: http://www.smartposition.nl/resources/sms_pdu.html
19. SOAP: http://www.w3.org/TR/soap/
20. Wikipedia – wolna encyklopedia – XML: http://pl.wikipedia.org/wiki/XML
21. XML: http://www.w3.org/XML/
22. HTTP: http://www.w3.org/Protocols/

23. HTTP POST: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.5
24. Tablica POST: http://4programmers.net/PHP/Tablice_w_PHP
25. JavaScript: https://developer.mozilla.org/pl/docs/JavaScript
26. How to connect to SQL server database from javascript?: http://stackoverflow.com/questions/857670/how-to-connect-to-sql-server-database-from-javascript
27. Adobe Flash: http://www.adobe.com/pl/products/flash.html
28. Microsoft Silverlight: http://www.microsoft.com/silverlight/
29. Michlmayr A., Rosenberg F., Platzer C., Treiber M., Dustdar S.: Towards Recovering the Broken SOA Triangle – A Software Engineering Perspective. IW-SOSWE '07 2nd International Workshop on Service Oriented Software Engineering: in conjunction with the 6th ESEC/FSE joint meeting. ACM, New York, NY, USA 2007, s. 22÷28.
30. XMLReader Class: http://msdn.microsoft.com/en-us/library/system.xml.xmlreader.aspx
31. XPath: http://www.w3.org/TR/xpath20/
32. LinqToXML: http://msdn.microsoft.com/pl-pl/library/bb387098%28v=vs.110%29.aspx
33. Kozielski M.: Wykorzystanie grupowania do przyspieszania zapytań ilościowych na dokumentach XML. [w:] Kozielski S., Małysiak B., Kasprowski P., Mrozek D. (red.): Bazy Danych – Rozwój Metod i Technologii. WKŁ, Warszawa 2008.
34. Duszeńko A., Kozielski M.: Bazy danych XML. [w:] Kozielski S., Małysiak B., Kasprowski P., Mrozek D. (red.): Bazy Danych – Modele, Technologie, Narzędzia. WKŁ, Warszawa 2005.
35. Nurzyńska K., Duszeńko A.: Hybrydowe bazy danych XML. [w:] Kozielski S., Małysiak B., Kasprowski P., Mrozek D. (red.): Bazy Danych – Struktury, Algorytmy, Metody. WKŁ, Warszawa 2006.
36. Drzewiecki Ł., Tuzinkiewicz K.: Rola testowania w projektowaniu XML-owych baz danych. [w:] Kozielski S., Małysiak B., Kasprowski P., Mrozek D. (red.): Bazy Danych – Struktury, Algorytmy, Metody. WKŁ, Warszawa 2006.
37. Cavalieri F., Guerrini G., Mesiti M.: XSPath: Navigation on XML Schemas Made Easy. IEEE Transactions on Knowledge and Data Engineering, Vol. 26, 2014, s. 485÷499.

Wpłynęło do Redakcji 7 lutego 2014 r.

**Abstract**

Recent a dozen of years is continually growing popularity of such two technologies as the internet WWW pages and sending the SMS messages [1]. It is why arose the question: in what way to place on the WWW page the SMS message, sent by mobile phone.

The task of sending SMS messages on WWW pages was divided on 2 stages:

- Reading of message from device (GSM modem, mobile phone).
- Placing of the message on the WWW page.

For the realization of these tasks the application 'SMSHost' was realized (Fig. 1). This application can read the SMS message in several different ways:

- Reading of data with use of 'GSMComm' library (Listing 1).
- Reading of the message with use of the class 'SerialPort' and protocol 'Hayes' (Listing 2).

The application stores messages in the way that make possible of their reading by WWW pages as well as by other type application (Fig. 2). As data store can be use XML file or database (Fig. 5).

Making accessible the message from XML file (Fig. 3) were realized on 3 ways: reading with use of the class XmlReader [30], with use of the language XPath [31], or reading with use of LinqToXML [32]. The database for storing and making accessible the SMS message was realized as relational database (Fig. 4) and also as XML database.

Realized SMSHost application make possible to record and use SMS messages in comfortable way. Presented in the article description of technology that allows access from PC computer to SMS messages can be for many person both, large help in realization of such task and inspiration to the creation of own ideas based on the introduced solution.

**Adresy**

Robert BRZESKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, robert.brzeski@polsl.pl.
Paweł WRÓBEL: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, wrobelpj@gmail.com.