STUDIA INFORMATICA 2011
Volume 32 Number 2B (97)

Kamil WALCZAK, Katarzyna HARĘŻLAK
Politechnika Śląska, Instytut Informatyki

MOBILNY SYSTEM ZARZĄDZANIA NIERUCHOMOŚCIAMI

Streszczenie. Realizacja celów biznesowych różnorodnych firm i instytucji wspierana jest przez zastosowanie systemów informatycznych. Wydajne komputery, nowoczesne narzędzia i technologie dostępne na rynku umożliwiają tworzenie kompleksowych systemów, zaspokajających nawet bardzo wyszukane wymagania użytkowników. W artykule zaprezentowano należący do takiej klasy rozwiązań system, którego celem jest obsługa zadań biura nieruchomości.

Słowa kluczowe: baza danych, GPS, GoogleMaps, .NET, WebService

MOBILE REAL ESTATE MANAGEMENT SYSTEM

Summary. Realization of business goals in various companies and institutions is supported by using the IT systems. Efficient computers, modern tools and technologies make it possible to create complex systems, which are capable of satisfying sophisticated user needs. This article is presenting such a system for managing a real estate office.

Keywords: database, GPS, GoogleMaps, .NET, WebService

1. Wstęp

W obecnych czasach komputery stanowią niezastąpione narzędzie pracy. Moc obliczeniowa oraz możliwości znacząco zwiększyły się w przeciągu ostatniej dekady. Jednocześnie nastąpił znaczący spadek cen tych urządzeń, co wpłynęło na ich dużą popularyzację.

Współcześnie trudno wyobrazić sobie realizację celów biznesowych różnych instytucji bez wykorzystania systemu komputerowego. Dotyczy to także takich firm, jak biura nieruchomości. Celem artykułu jest prezentacja kompleksowego systemu, zbudowanego z użyciem najnowszych technologii, który umożliwia klientom przeglądanie oferty oraz kontakt z biurem, natomiast pracownikom agencji dostarcza narzędzi niezbędnych do wykony-

wania ich zadań. Chociaż na rynku znajduje się kilka komercyjnych rozwiązań [1, 2], zaprezentowany system ma szansę być dla nich sporą konkurencją. Został wyposażony w elementy, których brakuje w obecnych na rynku rozwiązaniach, a koszty jego wdrożenia są niskie.

2. Architektura systemu

Analizując omawianą dziedzinę przedmiotową, postanowiono zbudować system, który składać się będzie z trzech współpracujących ze sobą aplikacji:

- stacjonarnej – wykorzystywanej w siedzibie agencji nieruchomości, a która będzie umożliwiała zarządzanie wszystkimi danymi istniejącymi w systemie,
- mobilnej – służącej agentom biura nieruchomości pracującym w terenie do gromadzenia informacji o obiektach,
- serwisu internetowego (aplikacja WWW) – dodatkowego elementu systemu, który będzie prezentował ofertę agencji w Internecie.

Cały system oparto na jednej instancji bazy danych. Dzięki takiemu podejściu informacje wprowadzone lub zmodyfikowane w aplikacji stacjonarnej będą mogły od razu być udostępnione w serwisie internetowym. Ułatwione zadanie będzie miał również administrator infrastruktury, gdyż będzie musiał wykonywać kopie zapasowe tylko jednego źródła danych.

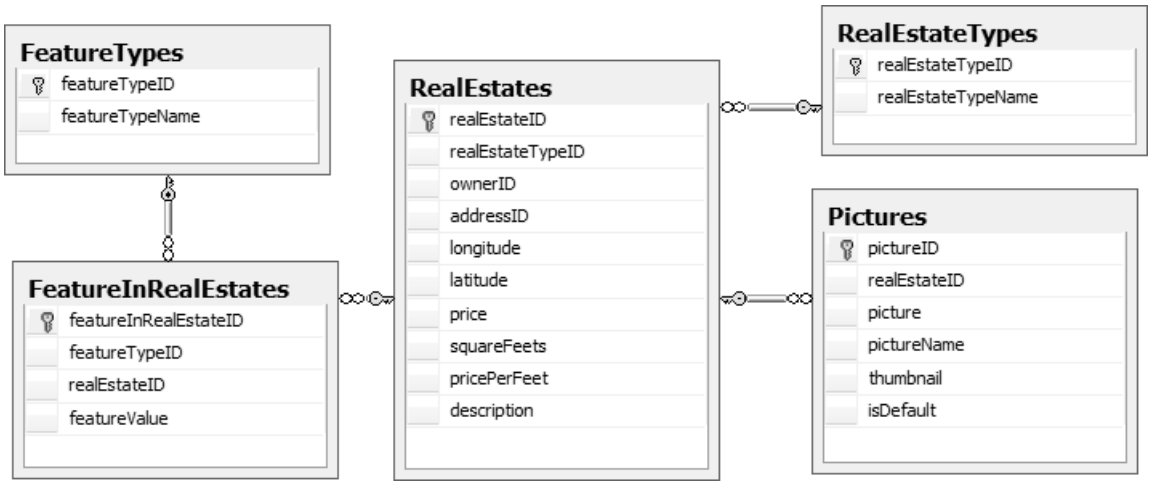
Część stacjonarna zbudowana została na podstawie architektury wielowarstwowej. Głównym jej założeniem jest rozdzielenie interfejsu użytkownika, przetwarzania oraz składowania danych na kilka niezależnych warstw, które mogą być osobno rozwijane. Takie rozbicie nie wpływa negatywnie na działanie całego systemu, a pozwala na jego łatwiejsze utrzymanie. Serwis internetowy oraz aplikacja przeznaczona na urządzenia mobilne zrealizowane zostały jako aplikacja typu klient-serwer.

Całość systemu zrealizowano na podstawie najnowszych technologii firmy Microsoft, do których należą platforma .NET Framework 4.0 wraz z językiem C# 4.0 [7], ASP.NET oraz, jako silnik bazodanowy, Microsoft SQL Server 2008 [8].

3. Baza danych systemu

Przestawiony na rys. 1 diagram bazy danych stanowi tylko fragment pełnej struktury przygotowanej na potrzeby systemu. Pełny schemat bazy danych składa się z 21 tabel gromadzących dane o nieruchomościach, związanych z nimi ofertami, przeprowadzonych transakcjach oraz klientach biura. Na rysunku przedstawiono tylko te tabele, które są istotne z punktu widzenia prezentowanych w artykule treści.

Tabelą przechowującą podstawowe informacje o nieruchomościach jest *RealEstates*. Zawiera ona przede wszystkim identyfikator typu nieruchomości, dla którego nazwa znajduje się w tabeli *RealEstateTypes* – jest to tabela słownikowa. Każda nieruchomość może posiadać zdjęcia. Są one umieszczone w tabeli *Pictures*, posiadającej, jako jedną ze swoich kolumn, identyfikator nieruchomości – dla jednego obiektu może istnieć wiele obrazów.



Rys. 1. Fragment schematu bazy danych systemu
Fig. 1. Part of the system database schema

Kolumny *picture* oraz *thumbnail* tabeli *Pictures* są typu varbinary(max). W projekcie rozważane było użycie kolumn typu *FileStream*[1] varbinary(max), jednak, ze względu na niewielki rozmiar plików graficznych, korzystniejsze okazało się wykorzystanie varbinary(max). Mechanizm *FileStream* wprowadza narzut czasowy związany z tworzeniem pliku oraz jego obsługą, z którym nie spotkamy się w przypadku typu varbinary(max), a to w przypadku obrazów o małej wielkości byłoby zauważalne.

Każda nieruchomość może posiadać wiele różnych cech. Chcąc umożliwić wprowadzanie pracownikom biura wszystkich niezbędnych informacji o opisywanym obiekcie, konieczne było zaprojektowanie elastycznej struktury, którą reprezentują tabele *FeatureTypes* oraz *FeatureInRealEstates*.

4. Wykorzystane mechanizmy

Zrealizowanie wszystkich wymagań stawianych przed prezentowanym systemem, zbudowanym z kilku aplikacji cechujących się rozbudowaną funkcjonalnością, wymagało zastosowania wielu niestandardowych mechanizmów. Dotyczą one takich obszarów działania systemu, jak: obsługa komunikacji i wymiany danych pomiędzy jego elementami, pozyskiwanie

[1] FileStream to nowość w SQL Server 2008, która umożliwia przechowywanie dużych obiektów binarnych zamiast w bazie danych.

i zarządzanie zdjęciami, pozyskiwanie danych o lokalizacji nieruchomości i wykorzystanie ich do prezentacji na mapie cyfrowej oraz ułatwienie komunikacji z klientami biura nieruchomości. Niniejszy artykuł poświęcony zostanie prezentacji wykorzystanych rozwiązań.

4.1. Komunikacja z bazą danych

W projekcie architektury warstwowej dla aplikacji desktopowej zdecydowano się na użycie mapowania obiektowo-relacyjnego. Służy ono jako warstwa dostępowa do bazy danych. Głównym celem aplikacji mobilnej jest gromadzenie oraz aktualizacja danych o nieruchomościach w terenie. Synchronizacja tak zmienionych informacji z centralną bazą danych odbywa się przez wykorzystanie usług udostępnionych w Internecie. Zadanie to spełnia *WebService* – usługa sieciowa niezależna od platformy i implementacji – której zadaniem jest dostarczenie żądanej funkcjonalności. Obiekt ten jest także pośrednikiem pomiędzy bazą danych i aplikacją WWW, zadaniem której jest prezentacja, w przejrzysty i atrakcyjny dla klientów sposób, witryny zgromadzonych ofert, bez możliwości dokonywania zmian.

Wykorzystanie obiektu *WebService* wymaga wcześniejszej jego rejestracji w projekcie (funkcja *AddWebReference*s). Sama komunikacja odbywa się przez utworzenie obiektu usługi, ustawienia jej adresu oraz podania danych niezbędnych do poprawnej autoryzacji. Akcje te wykonywane są przez stworzoną w aplikacji klasę *MobileWebService*, która pełni rolę obiektu dostępowego.

```
public class MobileWebService
{
   //obiekt usługi
   private readonly mobileWS.MobileWebService mobileWebService;
   //konstruktor bezparametryczny
   public MobileWebService()
   {
      //stworzenie obiektu usługi
      mobileWebService = new mobileWS.MobileWebService
      {
         //ustawienie danych logowania
         UserCredentialsValue = new UserCredentials
         {
            UserName = Consts.LoginUserName,
            Password = Consts.LoginUserPassword
         },
         //ustawienie adresu usługi
         Url = Consts.MobileWebServiceUrl
      };
   }
   //pozostałe metody
}
```

Zadaniem klasy jest przede wszystkim ulokowanie konfiguracji obiektu usługi w jednym miejscu oraz obsługi wyjątków, które mogą się pojawić w trakcie działania programu. W klasie tej dostępna jest także obsługa każdej, potrzebnej do prawidłowego funkcjonowania

aplikacji mobilnej, metody obiektu *WebService*. Funkcje obiektu *WebService* (*WebMethod*[2]), w celu wykonywania zapytań do relacyjnej bazy danych, wykorzystują technologię ADO.NET, która jest częścią .NET Framework. Ponieważ wywołanie metod z udostępnionej usługi wymaga procesu identyfikacji klienta, w przedstawionym wcześniej fragmencie kodu można zaobserwować ustawienie wartości *UserName* i *Password* w instancji klasy *UserCredentials*.

Firma Microsoft nie wspiera w swoich produktach plików konfiguracyjnych dla urządzeń przenośnych. W związku z tym, zdecydowano się użyć specjalnie przygotowanej klasy – *Consts* – zawierającej niezbędne ustawienia.

```
public class Consts
{
   public const string LoginUserName = "_G8Soj-DRJS-
   0AHDj5qI!!@hOeaUD)(0yX6^%hX";
   public const string LoginUserPassword =
   "!tgxVhuFa_+XPTsqh^^GxahceHYd8Kxf3k8j_!@#^%";
   public const string MobileWebServiceUrl =
   @"http://kwalczak:1080/MobileWebService.asmx";
}
```

Klasa zawiera trzy pola, z których każde jako wartość posiada łańcuch tekstowy:

- *LoginUserName* – nazwa użytkownika niezbędna do uwierzytelnienia żądania do wykorzystywanej usługi,
- *LoginUserPassword* – hasło użytkownika wykorzystywane wraz z *LoginUserName*,
- *MobileWebServiceUrl* – adres obiektu *WebService*, który zapewnia urządzeniu mobilnemu komunikację z bazą danych.

Z obiektem *WebService*, przez serwis internetowy, komunikuje się również aplikacja internetowa. Wymaga to uzupełnienia wpisu zawierającego adres udostępnionej usługi w pliku konfiguracyjnym tej aplikacji:

```
<applicationSettings>
   <REManager.WebSite.Properties.Settings>
      <setting name="REManager_WebSite_SiteWS_WebSiteService" seriali-
      zeAs="String">
         <value>http://localhost:31055/WebSiteService.asmx</value>
      </setting>
   </REManager.WebSite.Properties.Settings>
</applicationSettings>
```

W przedstawionej części pliku ‘web.config’ widoczny jest wpis nazwany *REManager_WebSite_SiteWS_WebSiteService*, którego wartość determinuje lokalizację obiektu *WebService*. Wpisy *LoginUserName* oraz *LoginUserPassword* w aplikacji WWW oraz w obiekcie WebService muszą być zgodne. W przeciwnym przypadku poprawna komunikacja pomiędzy nimi będzie niemożliwa.

[2] WebMethod – metoda udostępniana przez WebService.

4.2. Wymiana danych pomiędzy elementami aplikacji

Wymiana danych pomiędzy aplikacjami desktopową oraz mobilną realizowana jest z użyciem plików XML. *Compact Framework*, który jest uboższą wersją platformy .NET, dostarcza nam niezbędnych klas pozwalających na korzystanie ze standardu XML. Zastosowanie języka XML w stworzonym rozwiązaniu sprowadza się do dwóch akcji:

- serializacja – proces zamiany obiektu na plik XML,
- deserializacja – utworzenie obiektu z ustawionymi właściwościami na podstawie pliku XML.

Serializacji może zostać poddana każda klasa napisana w języku C#. Dalej zaprezentowane zostaną dwie funkcje, które realizują obsługę XML. Pierwsza z nich odpowiedzialna jest za zamienienie zawartości pliku XML na obiekt typu *RealEstateDetails*, z uzupełnionymi wartościami pól, klasa *XmlSerializer*, realizująca w tym przypadku proces deserializacji – metoda *Deserialize*.

```
public static RealEstateDetails Deserialize(FileInfo xmlPath)
{
   FileStream fs = null;
   RealEstateDetails details;
   try
   {
      //otwarcie strumienia do pliku z XML
      fs = xmlPath.OpenRead();
      //utworzenie obiektu serializera
      XmlSerializer xs = new XmlSerializer(typeof(RealEstateDetails));
      //deserializacja pliku do obiektu zawierającego szczegóły //nieruchomości
      details = xs.Deserialize(fs);
   }
   finally
   {
      //zamknięcie strumienia
      if (fs != null) fs.Close();
   }
   //wynikowy obiekt
   return details;
}
```

Druga z funkcji odpowiada za proces odwrotny – serializację. Jej zadaniem jest zamiana obiektu typu *RealEstateDetails* na plik XML. Podobnie do przypadku deserializacji, zapisywanie informacji o obiekcie do pliku XML odbywa się przez instancję klasy *XmlSerializer* – metoda *Serialize*. Zaimplementowane funkcje zapewniają możliwość zapisania stanu obiektu do pliku oraz jego odczytanie w zależności od żądanej akcji.

```
public static bool Serialize(String path, RealEstateDetails item)
{
   FileStream fileStream = null;
   try
   {
      //utworzenie docelowego pliku
      fileStream = new FileStream(path, FileMode.Create);
      //utworzenie obiektu serializera
      XmlSerializer xs = new XmlSerializer(typeof(RealEstateDetails));
```

```
            //utworzenie obiektu zapisującego XML
            //i ustawienie formatowania wynikowego pliku XML z wcięciami
            XmlTextWriter xmlTextWriter = new XmlTextWriter(fileStream, Encoding.UTF8)
            { Formatting = Formatting.Indented };
            //serializacja obiektu do wskazanego pliku
            xs.Serialize(xmlTextWriter, item, null);
            return true;
        }
        finally
        {
            //zamknięcie strumienia
            if (fileStream != null) fileStream.Close();
        }
    }
```

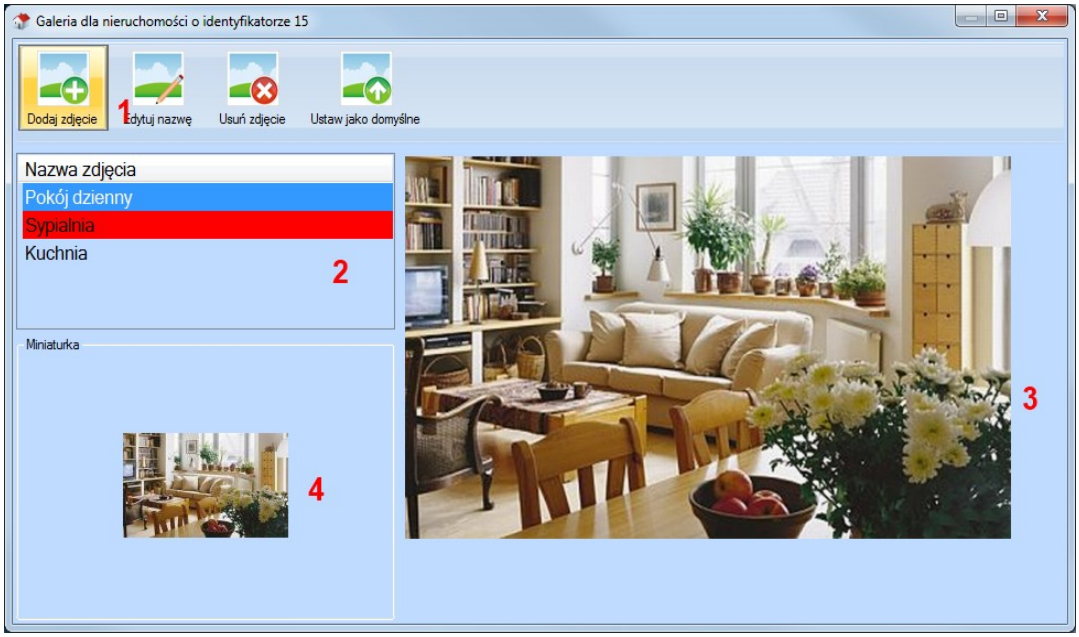
4.3. Obsługa zdjęć nieruchomości

Zadaniem aplikacji mobilnej jest aktualizacja danych o istniejącej w systemie nieruchomości. Główne okno aplikacji (rys. 2a) zawiera listę plików (1) z informacjami o zaimportowanych na urządzenie przenośne obiektach. Dodatkowo możliwe jest zrobienie zdjęć aktualizowanej nieruchomości, przez wykorzystanie wbudowanego w telefon aparatu cyfrowego (rys. 2b). Zakładka *Zdjęcia* zawiera listę zdjęć (1) oraz umożliwia zarządzanie nimi:

- Nowy (2) – wywołuje okno dialogowe robienia nowego zdjęcia (domyślne dla danego urządzenia przenośnego),
- Usuń (3) – usuwa wybrane w liście (1) zapisane zdjęcie.

Rys. 2. Aplikacja mobilna – edycja danych nieruchomości: a) opis i pozycja GPS, b) zdjęcia
Fig. 2. Mobile application real estate – data modification: a) description and GPS coordinates, b) photos

Po aktualizacji danych o nieruchomości w aplikacji mobilnej, konieczny jest eksport tych informacji do bazy danych. Do tego celu służy przycisk *Eksportuj*. Po poprawnym przesłaniu danych o obiekcie możemy je sprawdzić w aplikacji stacjonarnej (rys. 3).



Rys. 3. Aplikacja stacjonarna – galeria zdjęć dla nieruchomości
Fig. 3. Desktop application – real estate photo gallery

4.3.1. Przesyłanie zdjęć wraz z tworzeniem miniaturek

W momencie eksportu danych o nieruchomości z urządzenia mobilnego do bazy danych, oprócz informacji szczegółowych przesyłane są również zdjęcia obiektu. W momencie eksportu tworzone są automatycznie również ich miniaturki. Przekazywane są one w formie binarnej, dlatego w celu stworzenia miniaturki obrazu należy przekonwertować go na obiekt typu *Image*.

```
private static Image ByteArrayToImage(byte[] byteArray)
{
   //utworzenie strumienia w pamięci na podstawie tablicy bajtów
   MemoryStream ms = new MemoryStream(byteArray);
   //stworzenie obiektu Image na podstawie strumienia
   Image returnImage = Image.FromStream(ms);
   //zwrócenie wynikowego obiektu Image
   return returnImage;
}
```

Posiadając obiekt klasy *Image*, możemy, korzystając z metody *GetThumbnailImage*, w prosty sposób wygenerować jego miniaturkę. Funkcja ta tworzy pomniejszony obraz, o określonych rozmiarach, na podstawie zdjęcia w pełnej rozdzielczości.

```
//konwersja tablicy bajtów na obiekt typu Image
Image image = ByteArrayToImage(photoImage.Photo);
//wyliczenie skali na bazie wymaganej szerokości
double scale = image.Width / (double)150;
//obliczenie wysokości miniaturki, aby zachować proporcje
int newHeight = Convert.ToInt32(image.Height / scale);
//wygenerowanie miniaturki
Image thumbImage = image.GetThumbnailImage(150, newHeight, null, IntPtr.Zero);

//konwersja obiektu typu Image na tablicę bajtów
```

```
byte[] thumbArray = ImageToByteArray(thumbImage);
```

W pokazanym wcześniej fragmencie kodu użyta została metoda *ImageToByteArray*, która konwertuje obiekt *Image* na tablicę bajtów. Wynika to z faktu, że zdjęcia w bazie danych przechowywane są w formie binarnej.

4.3.2. Użycie aparatu fotograficznego

Dzięki wykorzystaniu technologii Compact Framework, obsługa aparatu wbudowanego w urządzenie przenośne sprowadziła się do stworzenia instancji oraz wywołania metod klasy *CameraCaptureDialog*. Przed wywołaniem okna dialogowego, które przeznaczone jest do robienia zdjęcia, należy ustawić wybrane parametry klasy *CameraCaptureDialo*.

```
//utworzenie obiektu
CameraCaptureDialog dlgCam = new CameraCaptureDialog
{
   //ustawienie tytułu
   Title = "Zrób zdjęcie",
   //ustawienie aparatu (możliwa jest jeszcze kamera)
   Mode = CameraCaptureMode.Still,
   //ustawienie jakości zdjęcia
   StillQuality = CameraCaptureStillQuality.High,
   //ustawienie rozdzielczości zdjęcia
   Resolution = new Size(1600, 1200)
};

//wywołanie okna dialogowego do robienia zdjęcia
if (dlgCam.ShowDialog() == DialogResult.OK)
{
   //utworzenie strumienia na podstawie pola FileName (lokalizacja //pliku w
   pamięci)
   FileStream image = new FileStream(dlgCam.FileName, FileMode.Open,
   FileAccess.Read, FileShare.Read);
   //wykorzystanie posiadanego strumienia danych
}
```

4.4. Technologia GPS

W celu określania pozycji nieruchomości oraz późniejszej jej prezentacji na mapie cyfrowej, konieczne jest ustalenie odpowiednich współrzędnych geograficznych. W tym celu można wykorzystać urządzenie przenośne wyposażone w odbiornik sygnału GPS-NAVSTAR [4]. Aplikacja mobilna wykorzystuje go do określania pozycji danego obiektu. Urządzenie, na którym testowana była aplikacja mobilna, nie posiadało wbudowanego odbiornika sygnału GPS, niezbędne więc było użycie zewnętrznego modułu, komunikującego się z wykorzystaniem technologii *Bluetooth*. W tym celu konieczne było włączenie w urządzeniu przenośnym funkcji *Bluetooth* oraz wyszukanie zewnętrznego odbiornika sygnału GPS. Następnie należało ustawić opcję w urządzeniu mobilnym, tak aby wykorzystywało ono zewnętrzny moduł do ustalania współrzędnych geograficznych. Do realizacji tej funkcjonalności opracowano klasę o nazwie *BluetoothGPS*.

```
//obiekt odpowiedzialny za pobieranie informacji z GPS
BluetoothGPS bluetoothGPS = new BluetoothGPS();
//strumień danych z GPS
StreamReader gpsStream;

//funkcja realizująca połączenie z modułem GPS
if (bluetoothGPS.ConnectToGPS())
{
   //strumień ramek z odbiornika GPS
   gpsStream = bluetoothGPS.GetStream();
}
else
{
   //informacja o błędzie podczas połączenia
   MessageBox.Show("Brak połączenia z modułem GPS");
}
```

Pobieranie współrzędnych wymaga znajomości formatu ramki GPS. Należy z niej wydobyć potrzebne informacje – współrzędne geograficzne aktualnej lokalizacji. Dostęp do danych z modułu GPS uzyskujemy przez ustawiony podczas połączenia strumień:

```
//pobranie ramki ze strumienia
string frame = gpsStream.ReadLine();
```

Natomiast proces przetwarzania ramki polega na uzyskaniu odpowiednich informacji z łańcucha tekstowego:

```
//przykładowa ramka odczytana z odbiornika GPS
const string frame =
"$GPRMC,040302.663,A,3939.7,N,10506.6,W,0.27,358.86,200804,,*1A";
//utworzenie tablicy z wartości rozdzielonych przecinkiem
string[] fields = frame.Split(',');
//odczytanie szerokości geograficznej
double latitude = Convert.ToDouble(fields[3].Replace(".", ","));
```

4.5. Prezentacja nieruchomości na mapie cyfrowej

System Informacji Geograficznej (GIS – Geographic Information System) [3, 5] jest to system informacyjny, którego zadaniem jest gromadzenie, obróbka, analiza i, przede wszystkim, wizualizacja danych geograficznych. W świecie nauki istnieje specjalna dziedzina – Geomatyka – która zajmuje się zastosowaniem systemów typu GIS. Za pomocą systemów typu GIS jesteśmy w stanie powiązać informację o cechach obiektów z danymi na temat ich współrzędnych geograficznych oraz topologii. Odpowiednie skrzyżowanie tych danych wraz z ich reprezentacją graficzną pozwalają uzyskać różne warstwy tematyczne (np. sieć rzeczna, stacje benzynowe, mapa adresowa miasta), które z kolei umożliwiają przeprowadzenie różnych analiz przestrzennych.

Podstawowym zadaniem aplikacji internetowej stworzonego systemu jest prezentacja katalogu agencji nieruchomości na mapie cyfrowej. Umożliwia to użytkownikom wizualne przeglądanie położenia poszczególnych ofert biura. W celu realizacji tego zadania zdecydowano się użyć komponentu *GoogleMaps.Subugrim.NET,* który bazuje na Mapach Google [6].

Wymaga to wprowadzenia ustawień do pliku konfiguracyjnego aplikacji:

```
<appSettings>
   <add key="googlemaps.subgurim.net" value="ABQIAAAAT0vLILe_5RN4rV-
   DTEAL2xQp2rNdjD9Dd-oq_0BgknHe6uVecRShrjVHG_oP7av5TrS8gs3559qStA" />
   <add key="LoginUserName" value="_G8Soj-DRJS-0AHDj5qI!!@hOeaUD)(0yX6^%hX"/>
   <add key="LoginUserPassword" val-
   ue="!tgxVhuFa_+XPTsqh^^GxahceHYd8Kxf3k8j_!@#^%"/>
</appSettings>
```

- *googlemaps.subgurim.net* – wartością tego klucza jest tzw. *Google Key*, który jest niezbędny do poprawnego działania mapy cyfrowej,
- *LoginUserName* – nazwa użytkownika, która musi być zgodna z analogicznym wpisem w usłudze wykorzystywanej do komunikacji z bazą danych,
- *LoginUserPassword* – hasło użytkownika potrzebne do identyfikacji (wraz z *LoginUserName*).

Uzyskany rezultat zaprezentowano na rys. 4. Kliknięcie na odnośnik Szczegóły powoduje przeniesienie na stronę przygotowaną do prezentacji dokładnych informacji o ofercie wraz z listą nieruchomości, które się w niej znajdują, oraz galerii miniaturek zdjęć obiektu, które po kliknięciu umożliwiają zobaczenie obrazu w pełnej rozdzielczości.

Do poprawnego wyświetlania obiektów na mapie cyfrowej w pełni wystarczyły trzy funkcje:

- wyśrodkowanie mapy, ustawienie jej typu oraz powiększenia:

```
//wywołanie funkcji setCenter, która jako parametry przyjmuje współrzędne
//geograficzne lntLong, powiększenie równe 10 oraz rodzaj mapy
this.GMap.setCenter(lntLong, 10, mapType);
```

- ustawienie opcji mapy:

```
GMapUIOptions options = new GMapUIOptions
{
    //możliwość przełączenia na mapę hybrdową
    maptypes_hybrid = false,
    //możliwość nawigacji używając klawiatury
    keyboard = false,
    ....
};
//przypisanie obiektowi mapy skonfigurowanego obiektu opcji
GMap.Add(new GMapUI(options));
```

- dodanie markera wraz z opisem:

```
//utworzenie obiektu markera z przypisanymi współrzędnymi geograficznymi lntLong
GMarker marker = new GMarker(lntLong);
//obiekt opisu w formie 'dymka'
GInfoWindow window = new GInfoWindow(marker, "Przykładowy opis", false, GListe-
ner.Event.mouseover);
//dodanie markera do mapy
GMap.addInfoWindow(window);
```

Rys. 4. Serwis internetowy – strona główna
Fig. 4. Internet application – main page

4.6. Wysyłanie wiadomości pocztą elektroniczną

Potencjalny klient agencji nieruchomości ma możliwość komunikacji z nią przez serwis internetowy. W tym celu wykorzystany został mechanizm poczty elektronicznej, który wymagał wprowadzenia odpowiednich ustawień w pliku konfiguracyjnym aplikacji:

```
<system.net>
   <mailSettings>
      <smtp from="userName@poczta.pl">
         <network host="poczta.pl" port="587" userName="userName" pass-
         word="p@ssw0rd"/>
      </smtp>
   </mailSettings>
</system.net>
```

Samo wysłanie poczty odbywa się na jeden z dwóch możliwych sposobów:

- wysłanie wiadomości do opiekuna oferty, w przypadku podania w linku jego identyfikatora:

```
http://www.biuronieruchomosci.pl/Sites/Contact.aspx?employeeID=1&mailTopic=Temat
```

- wysłanie wiadomości na adres ogólny, ustawienie którego jest możliwe przez plik konfiguracyjny.

5. Podsumowanie

W artykule przedstawiono system zarządzania biurem nieruchomości, składający się z trzech aplikacji dedykowanych różnym środowiskom pracy. Na system ten złożyły się aplikacje stacjonarna, internetowa oraz mobilna, które wzajemnie uzupełniając swoje funkcjonalności, pozwoliły na stworzenie kompleksowego rozwiązania.

Prezentacja zastosowanych w nim mechanizmów miała na celu potwierdzenie lub uświadomienie aktualności poszukiwań innowacyjnych rozwiązań w dziedzinach, które wydają się być od dłuższego skomputeryzowane. Wynika to z faktu, że ciągle rozwijane i udoskonalane technologie wytwarzania oprogramowania oraz metodyki projektowania systemów informacyjnych stwarzają nowe możliwości oraz nową jakość w zakresie udostępnianych usług.

Pomimo tego, że zaprezentowane elementy systemu dedykowane były konkretnej branży, z pewnością, ze względu na swą uniwersalność, mogą z powodzeniem zostać zaadoptowane do innych obszarów życia.

BIBLIOGRAFIA

1. Bluebell – Strony WWW dla biur nieruchomości, http://www.blue-bell.pl/aplikacje_biur_nieruchomosci.html, (04.2010).
2. Dega Software Development – oprogramowanie dla biur nieruchomości, http://www.dega.com.pl/, (04.2010).
3. Davis D. E.: GIS dla każdego. MIKOM, 2004.
4. Januszewski J.: Systemy satelitarne GPS, Galileo i inne. PWN, Warszawa 2006.
5. Litwin L., Myrda G.: Systemy Informacji Geograficznej. Zarządzanie danymi przestrzennymi w GIS, SIP, SIT, LIS. Helion, Gliwice 2005.
6. Strona z komponentem do ASP.NET, umożliwiającym wykorzystanie Goolge Maps, http://googlemaps.subgurim.net/.
7. Perry S. C.: Core C# i .NET. Helion, Gliwice 2006.
8. Stanek W. R.: Vademecum Administratora Microsoft SQL Server 2008, Microsoft Press, 2009.

Recenzent: Prof. dr hab. inż. Stanisław Wrycza

Wpłynęło do Redakcji 30 stycznia 2011 r.

Abstract

This article presents a system for real estate office. The system consists of three applications dedicated for various tasks. Desktop, web and mobile applications complement each others' capabilities and form a complex system, which has a chance to compete with other commercial solutions. The system has a few unique mechanisms, that other, similar, applications do not have, and the costs of deployment are low.

Realization of all features requires the use of many modern technologies. Communicating in system, exchanging of data between applications, making and managing photos, getting GPS localization, showing real estates on a digital map, sending e-mails and other functions require to use many custom solutions. Technologies, which were used in the sytem: C#, ASP.NET, GoogleMaps, GPS, WebService, XML, .NET Framework, SQL Server 2008.

The main purpose of showing some mechanisms of the system is confirmation, that looking for new solutions is still necessary even in a domain, that is using computers for years. New technologies provide opportunity to create better systems and increase the quality of services.

Although elements of system that were shown in this article were dedicated to real estate domain, they can be used in other branches, because of their versatility.

Adresy

Kamil WALCZAK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, kamilwalczak@o2.pl.
Katarzyna HARĘŻLAK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, katarzyna.harezlak@polsl.pl.