L.Nayzabayeva, Zh.Orazbekov, G.Turken, G.Tleuberdiyeva

Designing of anylogic the imitation model for the incensement of enterprises competitiveness

Summary. The article discusses a new approach to assessment practices venture projects based on the creation of simulation models. The authors detail the agent-based simulation model of Internet startups in the sale of tickets: disclosed structure of the model, the logic of agents' behavior, analyzed the results. It is concluded that the high potential of simulation in the development strategy of the company and scenario planning.

ӘОЖ(378.016.02:004.032.6:574)

Жуманбаева А.М., Самбетбаева А.К., Мирзахмедова Г.А.
(Әл-Фараби атындағы Қазақ Ұлттық университеті
Алматы қ, Қазақстан Республикасы, gulban84@mail.ru)

C# ПРОГРАММАЛАУ ТІЛІНДЕ «ВКОНТАКТЕ» ӘЛЕУМЕТТІК ЖЕЛІСІНЕ АРНАЛҒАН DESKTOP ҚОСЫМШАСЫН ҚҰРУ

Аннотация. «ВКонтакте» әлеуметтік желісі күнделікті миллиондаған қолданушы пайдаланатын кең танымал жобалардың бірі болып табылады. ВКонтакте әлеуметтік желісі интернет браузер арқылы үздіксіз жұмыс істеумен қатар, өзіміздің дербес desktop-қосымшаларымызды құруға және іске қосуға мүмкіндік береді. «ВКонтакте» әлеуметтік желісінде қолданушыны авторландыруды жүзеге асыруға арналған браузерін қолдану ұсынылады, біз **WebBrowser** компонентін қолданамыз. Біз осы мақалада «ВКонтакте» әлеуметтік желісіне арналған қарапайым desktop-қосымшасын C# программалау тілінің көмегімен қалай жүзеге асыруға болатынын қарастырамыз.

Түйін сөздер: C# программалау тілі, XML тілі, .NET Framework 4.0 , WebBrowser, Интернет – браузер, API-әдісі.

C# программалау тілінде «ВКонтакте» әлеуметтік желісіне арналған desktop қосымша- сын құру үшін оны алдымен «ВКонтакте» қосу қажет. «ВКонтакте» әлеуметтік желісінде қолданушыны авторландыруды жүзеге асыруға арналған браузерін қолдану ұсынылады, біз **WebBrowser** компонентін қолданамыз. Бұл қолданушының қандай логин пароль енгізгенін қосымшада көрінбес үшін жасалынған. Шын мәнінде **WebBrowser**–ді пайдаланбай қолда-нушыдан оның логині мен түйін сөзін сұрау арқылы желіге тікелей қосылуға болады. Қолданушыны авторландыру рәсімін жүргізіп болғаннан соң API қосымшалар құру интерфейсімен жұмысты арнайы кілттің (**access_token**) көмегімен жүзеге асыруға болады. Қосымшаны жүзеге асыруға арналған сұранысты келесі адреске жібеуі қажет: https://api. vkontakte.ru/ method/ METHOD_NAME? PARAMETERS&access_token =ACCESS_TOKEN.

«ВКонтакте» әлеуметтік желісі сұраныс жүргізу нәтижесін **JSON**-форматтында немесе **XML** тілінде көресетеді. Біз сұранысты XML-мәліметі ретінде жүргіземіз. Ол қолданылатын әдістің атына (METHOD_NAME) xml (METHOD_NAME.xml) кеңейтілімін қосып жазамыз. «ВКонтакте» әлеуметтік желісіне арналған desktop қосымшасын **WebBrowser** арқылы авторландыру үшін жаңа **Windows Forms** қосымша формасынсын құрамыз. Осы формаға **WebBrowser** компонентін орналастырамыз. Содан соң формаларды жүктеу уақиғасында (**Form_Load**) қолданушыны авторландырушы **WebBrowser** парақшасын ашамыз, және біздің қосымшамыз үшін қажеттң құқықтарға сұраныс жасаймыз. **WebBrowser** парақшасының адресінде біздің қосымшамыздың сандық идентификаторын (**client_id**), қажетті рұқсатнамаларды (**scope**) және сұраныс типін(**display**) жіберу қажет. Қосымша идентификаторын ВКонтакте сайты қосымшасының баптауынан білуге болады. Қосымша идентификаторын оңай ауыстыру мүмкін болуы үшін жеке айнымалы ретінде сақтаған дұрыс, мысалы үшін , айнымалыны **appId** деп меншіктейік.

```
private int appId = 2419779;
```

Қолдануға ыңғайлы болу үшін біз сандың шамаларды қолданамыз, және құқықтар тізімін көрсетіп кетеміз.

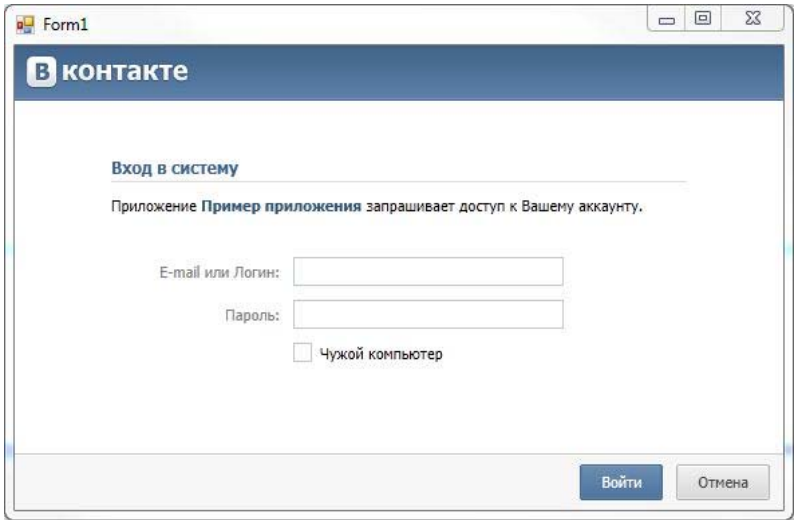
```
private enum VkontakteScopeList
{
notify = 1, friends = 2, photos = 4,
audio = 8, video = 16, offers = 32,
questions = 64, pages = 128, link = 256,
notes = 2048, messages = 4096, wall = 8192, docs = 131072;
}
```

Қажетті рұқсатнамаларды жеке айнымалығы сақтаймыз. Ол айнымалыны **scope** деп меншіктейік.

```
private int scope=(int)(VkontakteScopeList.audio|
VkontakteScopeList.docs|VkontakteScopeList.friends|
VkontakteScopeList.link|VkontakteScopeList.messages|
VkontakteScopeList.notes|VkontakteScopeList.notify|
VkontakteScopeList.offers|VkontakteScopeList.pages|
VkontakteScopeList.photos|VkontakteScopeList.questions|
VkontakteScopeList.video | VkontakteScopeList.wall);
```

ВКонтакте әлеуміттік желісін авторландыру парақшасына өту коды келесі түрде жазылады:

```
private void frmSignin_Load(object sender, EventArgs e)
{
webBrowser1.Navigate(String.Format("http://api.vkontakte.ru/oauth/authorize?client_id={0}&scope={1}&display=popup&response_type=token",appId, scope));
}
```



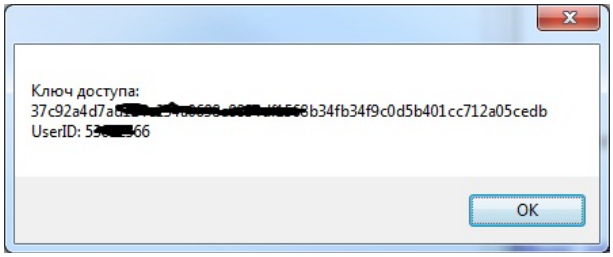
1-сурет. WebBrowser элементінде «ВКонтакте» сайтын авторландыру

Авторландыру және құқықтарға сұраныс жасау рәсімі қолжетімділік кілтін алу үшін қажет (**access_token**), осы кілттің көмегімен болашақта API интефесімен жұмыс жасауды жүзеге асырамыз. ВКонтакте әлеуметтік желісінде кілтті соңғы қадамда, яғни колданышу **WebBrowser** –дегі қажетті тексерістерді өтіп болғаннан кейін береді. **WebBrowser** –де қандай амалдар орындалып жатқанын **DocumentCompleted** оқиғасының көмегімен бақылап отыруға болады. Жүктелген **WebBrowser** адресінде **access_token** сөзі пайда болған соң сайттың **url** адресіне сұраныс жасап, одан қолданушының идентификаторы мен қол жетімділік кілтін алу қажет.

```
private void webBrowser1_DocumentCompleted(object sender,
WebBrowserDocumentCompletedEventArgs e)
{
if (e.Url.ToString().IndexOf("access_token") != -1)
{
string accessToken = "";
```

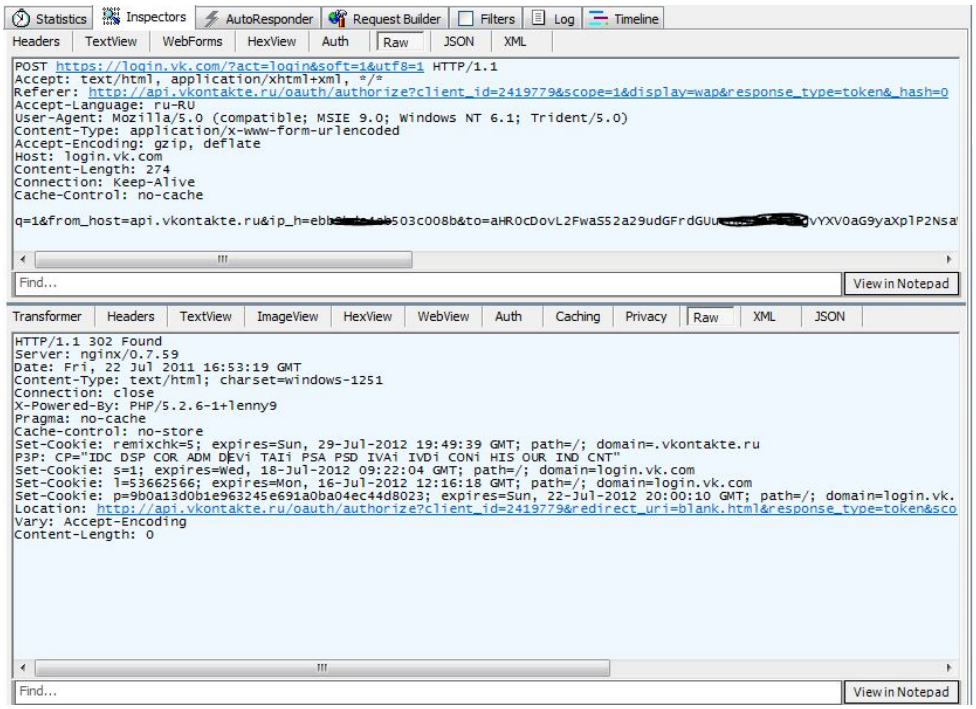
```
    int userId = 0;
    Regex myReg = new Regex (@"(?<name>[\w\d\x5f]+) =
(?<value>[^\x26\s]+)",RegexOptions.IgnoreCase| RegexOptions.Singleline);
    foreach (Match m in myReg.Matches(e.Url.ToString()))
    {
    if (m.Groups["name"].Value == "access_token"){
    accessToken = m.Groups["value"].Value;}
     else if (m.Groups["name"].Value == "user_id"){
     userId = Convert.ToInt32(m.Groups["value"].Value);}
    }
    MessageBox.Show(String.Format("Ключ     доступа:{0}\nUserID:     {1}",
accessToken, userId));
    }
```

Қолданушыны авторландыру және құқықтарға сұраныс жасау рәсімін жасап болған соң программаны іске қоссақ, бізге қолданушының идентификаторы мен қол жетімділік кілті көрсетілген терезе пайда болады



2-сурет. Қолданушының кілті мен ID

Алғашқы сұраныс біз браузер адрес қатарына жазылған адрес бойынша парақшаға барады. Ол жерге біз өзіміздің логиниміз бен түйін сөзімізді көрсеттік. «**Войти**» батырмасын шерткеннен соң формадағы мәліметтер парақшаға жіберіледі: ***https://login.vk.com/?act = login&soft=1& utf8=1***



3-сурет. «ВКонтакте» авторландыру барысында алынған(жоғары) және жіберілген(төменгі) HTTP-тақырыпшалары.

Авторландыру жұмыстары жүргізіліп болған соң, сервер біз құрып отырған қосымша және авторландыру ақпараттары на арналған сұраныс парақшасына қайта бағыттайтын 302-ші HTTP-кодты экранға шығарады. Қолданушы формадасындағы «**Разрешить**» батырмасын шерткеннен соң бізге келесі типтегі парақша ашылады: ***http://api.vkontakte.ru/ oauth/grant_access?hash= 68af114762009eaf68&client_id = 2419779&settings = 1&redirect_uri = blank.html&response_type=token&state.*** Осы команда бойынша қол жетімділік кілті және осы кілттің мерзімі, қолданушының идентификаторы көрсетілген парақшаға қайта бағыттайтын 302-ші HTTP-кодын аламыз.

Енді жоғарыда атап кеткен амалдардың барлығын программаға салу үшін, біз **WebBrowser** орнына **HttpWebRequest** және **HttpWebResponse** кластарын қолданамыз. «Қауіпсіз авторландыру» жұмысы үшін **appId**, **scope** екі айнымалысын, және **VkontakteScopeList** санағын (перечисление) қолданамыз. Сонымен қатар, екі тексттік өріс және бір батырмасы бар форма құрамыз. Бірінші текстік өріске **tbLogin**, екінші тексттік өріске **tbPassword**, ал батырмаға **btnSignin** деген атаулар береміз. Келесі кодтардың барлығы ақпараттарды өңдеуші (**btnSignin_Click**) батырмасын шерту арқылы жазылады.

```
    HttpWebRequest myReq = (HttpWebRequest)HttpWebRequest.Create
(String.Format("http://api.vkontakte.ru/oauth/authorize?client_id={0}&sco
pe={1}&display=wap&response_type=token", appId, scope));
    HttpWebResponse myResp = (HttpWebResponse)myReq.GetResponse();
    StreamReader myStream = new StreamReader (myResp.
GetResponseStream(),Encoding.UTF8);
     string html = myStream.ReadToEnd();
```

Парақшадағы ақпараттырдың барлығы **html** айнымалысына көшіріледі. Формадағы іздеу жұмысы көп қолданылатын айнымалылар бойынша жүргізіледі. Біз осы парақшадағы авторизациялау формасына сұраныс жасағанда қолданушының түйін сөзі мен логинін корсетіп оның барлық элементтерін көшіруіміз қажет.

```
    Regex myReg = new Regex("<form(.*?)>(?<form_body>.*?)</form>",
RegexOptions.IgnoreCase|RegexOptions.Multiline|RegexOptions.Singleline);
    if (!myReg.IsMatch(html) || (html = myReg. Match(html). Groups
["form_body"].Value) == "")
    {
    MessageBox.Show("Не удалось получить форму авторизации. Проверьте
шаблон регулярного выражения.");
    return; }
    myReg   = new Regex   ("<input(.*?)name=\   "(?<name>[^\x22]+)\"(.*?)
((value=\"(?<value>[^\x22]*)\"(.*?))|(.?))>",RegexOptions.IgnoreCase    |
RegexOptions.Multiline | RegexOptions.Singleline);
    NameValueCollection qs = new NameValueCollection();
    foreach (Match m in myReg.Matches(html))
    {
    string val = m.Groups["value"].Value;
    if (m.Groups["name"].Value == "email")
    {
     val = tbLogin.Text;
     }
    else if (m.Groups["name"].Value == "pass")
    {
    val = tbPassword.Text;
     }
    qs.Add(m.Groups["name"].Value, HttpUtility.UrlEncode(val));
```

Формадағы элементтердің барлығы тұтастай **qs** топтамасына көшіріледі. Логин (**email**) және парольдің (**pass**) орнына **tbLogin** және **tbPassword**. Текстің өрістегі ағымдағы қолданушының деректерімен алмастырылады. Содан соң ақпараттарды **POST** әдісі арқылы https://login.vk.com/?act=login&soft= 1&utf8=1 парақшасына сілтеме жасалады. **ContentType** элементін дұрыс беріп, Міндетті түрде **AllowAutoRedirect** автоматты қайта бағыттау элементін сөндіру мажет. Веб сайт туралы ақпараттармен (Cookies) жұмыс жасау үшін бізге сұраныста **CookieContainer** топтамасын құру қажет.

```
    byte[]b=System.Text.Encoding.UTF8.GetBytes(String.Join("&",from
item in qs.AllKeys select item + "=" + qs[item]));
    myReq
=(HttpWebRequest)HttpWebRequest.Create("https://login.vk.com/?ac        =
login&soft=1&utf8=1");
    myReq.CookieContainer = new CookieContainer();
    myReq.Method = "POST";
    myReq.ContentType="application/x-www-form-urlencoded";
    myReq.ContentLength = b.Length;
    myReq.GetRequestStream().Write(b, 0, b.Length);
    myReq.AllowAutoRedirect = false;
    myResp = (HttpWebResponse)myReq.GetResponse();
```

Серверде html-құрамы туралы жауап болмайды. Авторландыру жұмысы сәтті аяқталғаннан соң сервер веб-сатй туралы ақпаратты (Cookies) қайтарады. болашақта программада қолдану үшін бөлек айнымалыға сақтаймыз.

```
    CookieContainer cc = new CookieContainer();
    foreach (Cookie c in myResp.Cookies)
    {
    cc.Add(c);
    }
```

Сонымен сервер біздің қосымша үшін қолданушының сұраныс жасалған сілтемені парақшаға қайтарады. Парақша адресі **Location** HTTP-тақырыпшасында орналасуы қажет. Біз **GET** әдісінің көмегімен қолдан қайта бағыттау жұмысын жүргіземіз, сонымен қатар, алынған веб-сайт туралы ақпараттарды ұмытпаған жөн, кері жағдайда сервер бізге авторландыру парақшасын қайта шығарады.

```
    if (!String.IsNullOrEmpty(myResp.Headers["Location"]))
    {
    myReq = (HttpWebRequest) HttpWebRequest.Create(myResp. Headers
    ["Location"]);
    myReq.CookieContainer = cc;
    myReq.Method = "GET";
    myReq.ContentType = "text/html";
    myResp = (HttpWebResponse)myReq.GetResponse();
    myStream = new StreamReader(myResp.GetResponseStream(),
    Encoding.UTF8);
    html = myStream.ReadToEnd();}
    else
    { MessageBox.Show("Ошибка. Ожидался редирект.");
    return;
    }
```

html айнымалысында қосымшаға арналған рұқсат сұранысы парақшасының мазмұны болуы шарт. **html** айнымалысында осы ақпараттардың бар немесе жоқ екендігін тексері үшін біз теле парақшадан қолданушының аты мен сілтемесін табуымыз қажет. Ал егер қолданушының есімі болмаса, онда авторландыру жұмысының сәтсіз болғанын білдіреді.

```
myReg=new Regex("Вы авторизованы как <b> <a href=\"
(?<url>[^\\x22]+)\">(?<user>[^\\x3c]+)</a></b>",RegexOptions.IgnoreCase|R
egexOptions.Multiline|RegexOptions.Singleline); if (!myReg.IsMatch(html))
{
MessageBox.Show("Ошибка: Авторизация не прошла.");
return; }
else
MessageBox.Show(String.Format("Авторизация успешно прошла. \n
Пользователь {0}", myReg.Match(html).Groups["user"].Value));
```

Сонымен, біз «қауіпсіз авторландыру» әрекетінде алған нәтижелерге ие болдық, тек мұндағы айырмашылық қолданушы бұл авторландыру әрекетіне қатыспайды.

Содан соң қайта түзеп жіберу үшін форманы алу қажет. Бірақ алғашқы формаға (авторландыру формасы) қарағанда қосымшадағы сұраныс формасында бізді ең алдымен қателіктер түзетіліп қайта жіберу қажет болатын адрес қызықтыратын болады.

```
myReg= new Regex("<form(.*?)action=\"(?<post_url>[^\\x22]+)\"(.*?)>(?<
form_body>.*?)</form>", RegexOptions.IgnoreCase | RegexOptions.Multiline |
RegexOptions.Singleline);
  if (!myReg.IsMatch(html)) {
MessageBox.Show("Не удалось получить форму. Проверьте шаблон
регулярного выражения.");
string url = myReg.Match(html).Groups["post_url"].Value;
if (!url.ToLower().StartsWith("http://")) { url = String. Format
("http://api.vkontakte.ru{0}", url); }
Соңғы сұраныс жұмысын аяқтау үшін келесі командалық кодты теремiз:
myReq = (HttpWebRequest)HttpWebRequest.Create(url);
myReq.CookieContainer = cc;
myReq.Method = "POST";
myReq.ContentType = "application/x-www-form-urlencoded";
myReq.AllowAutoRedirect = false;
myResp = (HttpWebResponse)myReq.GetResponse();
Сервер қайта бығыттаған адрестен қолжетімділік кілтін аламыз.
 if (!String.IsNullOrEmpty(myResp.Headers["Location"]))
myReg= new Regex(@"(?<name>[\w\d\x5f]+)=(?<value>[^\x26\s]+)",
RegexOptions.IgnoreCase | RegexOptions.Singleline);
foreach (Match m in myReg.Matches(myResp.Headers["Location"]))
{
if (m.Groups["name"].Value == "access_token")
{ accessToken = m.Groups["value"].Value; }
else if (m.Groups["name"].Value == "user_id")
{userId = m.Groups["value"].Value; }
MessageBox.Show(String.Format("Ключ доступа: {0}\nUserID: {1}",
accessToken, userId));
else { MessageBox.Show("Ошибка. Ожидался редирект.");
}
```

Қолданушы туралы ақпаратты алу, құрамына келесі параметрлер енетін **getProfiles** api-әдісі арқылы жүзеге асырылады:

- Uids – үтір арқылы жазылған қолданушылардың ID-і;
- Domains - үтір арқылы жазылған қолданушылардың адрестері(uids-тің орнына қолднылады);
- Fields - үтір арқылы жазылған міндетті түрде толтырылатын анкеталар өрісі;
- name_case – қолданушының аты немесе тегі жіктелетін септік.

Біз қолданушының идентификаторына сұраныс жасайтын қарапайым функцияны құрамыз, бірақ сонымен қатар функция қолданушы туралы толық ақпаратты жинақтайтын болады.

```
public XmlDocument GetProfile(int uid)
{ NameValueCollection qs = new NameValueCollection();
```

```
    qs["uid"] = uid.ToString();
    qs["fields"]="uid,first_name,last_name,nickname,domain,sex,bdate,"
 +"city,country,timezone,photo,has_mobile,rate,contacts,education,online";
     return ExecuteCommand("getProfiles", qs);
    }
```

Программа дұрыс жұмыс жасаған жағдайда **GetProfile** функциясы бізге **response/user** тармағында орналасқан қолданушының Xml – ақпараттарын береді. Кері жағдайда программады ерекшелік пайда болады. Қолданушыны авторландыру нәтижесінде алған қол жетімділік кілтін беретін, **VKAPI** класының көшірмесін жасау арқылы функцияны іске қосайық.

```
VKAPI myVK = new VKAPI(accessToken);
```

Содан соң, **XmlDocument** типіндегі **profile** айнымалысын құрамыз.

```
XmlDocument profile;
```

Сәйкесінше осы айнымалыны **GetProfile** функциясының көмегімен алынған қолданушының профиліне орналастырамыз.

```
profile = myVK.GetProfile(103639273);
```

XmlDocument (**profile**) көшірмесіндегі **SelectSingleNode** әдісін қолдану арқылы қолданушы туралы ақпаратты қосымшаға шығаруға болады.

```
    MessageBox.Show(String.Format("Имя: {0}\r\nФамилия: {1}\r\nПол:
{2}",profile.SelectSingleNode("response/user/first_name").InnerText,
profile.SelectSingleNode("response/user/last_name").InnerText,
profile.SelectSingleNode("response/user/sex").InnerText));
```

Бірақ қолданушының профилінде ақпараттар толық болмауы мүмкін, сондықтан алынған xml-құжатта қажет етілген өрістер дұрыс толтырылғандығын тексеру қажет. Ол үшін жеке көмекші-функция құрамыз.

```
    public string GetDataFromXmlNode(XmlNode input)
    {
    if (input == null || String.IsNullOrEmpty(input.InnerText))
    { return "нет данных"; }
    else { return input.InnerText; }
    }
```

Көмекші функцияның көмегімен ерекшеліктерден қорықпай қолданушы туралы ақпаратты алуға болады.

```
    MessageBox.Show(String.Format("Имя: {0}\r\nФамилия: {1}\r\nПол:
{2}",GetDataFromXmlNode(profile.SelectSingleNode("response/user/
first_name")),GetDataFromXmlNode(profile.SelectSingleNode
    ("response/user/last_name")),GetDataFromXmlNode(profile.SelectSingleNo
de("response/user/sex"))));
```

Қолданушының суреттерін енгізу үшін xml-құжатта осы суреттер орналасқан адреске сілтеме ғана жасай аламыз. Осы сілтеменің көмегімен өолданушының суретін PictureBox –қа орналастырамыз. Сонымен қатар, суреттерді **WebClient** класындағы **DownloadData** функциясының көмегімен жүктеп алуға болады.

```
    if (profile.SelectSingleNode("response/user/photo")        != null &&
!String.IsNullOrEmpty(profile.SelectSingleNode("response/user/photo").Inn
erText))
    { PictureBox picPhoto = new picPhoto();
    WebClient wc = new WebClient();
    byte[]   b   =   wc.DownloadData(profile.SelectSingleNode("response/
user/photo").InnerText);
    MemoryStream m = new MemoryStream(b);
    picPhoto.Image = Image.FromStream(m);
    picPhoto.Width = picPhoto.Height = 50;
    picPhoto.Visible = true;
    this.Controls.Add(picPhoto);
    }
```

Қалалар мен елдердің атауларын алу үшін, класс профильде елді мекендердің сандық иеднтификаторын қайтаратын болғандықтан VKAPI функциясын қолданамыз. Функция елді – мекен атуаларын код ретінде қабылдап, экранға осы елдер мен қалалардың атауларын текс түрінде шығарады.

```
    public string GetCity(int id)
    {
    if (id <= 0) { return "нет данных"; }
    NameValueCollection qs = new NameValueCollection();
    qs["api_id"] = Program.appId.ToString();
    qs["cids"] = id.ToString();
    XmlDocument city = ExecuteCommand("getCities", qs);
return city.SelectSingleNode("response/city/name").InnerText;
    }
     public string GetCountry(int id) {
    if (id <= 0) { return "нет данных"; }
    NameValueCollection qs = new NameValueCollection();
    qs["api_id"] = Program.appId.ToString();
    qs["cids"] = id.ToString();
    XmlDocument country = ExecuteCommand("getCountries", qs);
    return country.SelectSingleNode("response/country/name").InnerText;
}
```

«ВКонтакте» әлеуметтік желісі қолдануға өте ыңғайлы болып келетін ауқымды арі-әдістерге ие. Бұл мақалады біз осы әдістердің бірнеше бөлігін ғана қарастырдық.

ӘДЕБИЕТ

1. Фахад Г. C# и наука: применение языковых средств C# в проектах для научных вычислений [Электронный ресурс] // MSDN Magazine. Электрон. дан. URL: http://msdn.microsoft.com/ru-ru/library/dd353137.aspx, свободный. Яз. рус. (дата обращения 12.12.2014).

2. Шилдт Г. Полный справочник по C#/ Г. Шилдт : пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 752 с.:ил.

3. Постолит А.В. Visual Studio.Net: Разработка приложение базы данных. // БХВ-Петербург, 2003г.

4. Агуров П.В. C#. Разработка компонентов в MS Visual Studio 2005/2008. // БХВ-Петербург, 2008г.

5. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .Net Framework 4.5 на языке C# // 4-ое изд.– Питер, - 2013г.

REFERENCES:

1. Fakhad G. C# i nauka: primeneniye yazykovykh sredstv C# v proyektakh dlya nauchnykh vychisleniy [Elektronnyy resurs] // MSDN Magazine. Elektron. dan. URL: http://msdn.microsoft.com/ru-ru/library/dd353137.aspx, svobodnyy. YAz. rus. (data obrashcheniya 12.12.2014).

2. Shildt G. Polnyy spravochnik po C#/ G. Shildt : per. s angl. – M.: Izdatel'skiy dom «Vil'yams», 2004. – 752 s.:il.

3. Postolit A.V. Visual Studio.Net: Razrabotka prilozheniye bazy dannykh. // BKHV-Peterburg, 2003g.

4. Agurov P.V. C#. Razrabotka komponentov v MS Visual Studio 2005/2008. // BKHV-Peterburg, 2008g.

5. Rikhter Dzh. CLR via C#. Programmirovaniye na platforme Microsoft .Net Framework 4.5 na yazyke S# // 4-oye izd.– Piter, - 2013g.

Жуманбаева А.М., Самбетбаева А.К., Мирзахмедова Г.А.

C# программалау тілінде «ВКонтакте» әлеуметтік желісіне арналған desktop- қосымшасын құру

Түйіндеме. Мақалада C# программалау тілінің көмегімен қарапайым «ВКонтакте» әлеуметтік желісінде арналған desktop-қосымшасын құру тәсілдері қарастырылған. desktop-қосымшасын құру үшін .NET Framework 4.0 және C# программалау тілі қолданылды. «ВКонтакте» әлеуметтік желісі қолдануға өте ыңғайлы болып келетін ауқымды арі-әдістерге ие. Бұл мақалады біз осы әдістердің бірнеше бөлігін ғана қарастырдық.

Түйін сөздер: C# программалау тілі, XML тілі, .NET Framework 4.0 , WebBrowser,.

Жуманбаева А.М., Самбетбаева А.К., Мирзахмедова Г.А.

Разработка desktop-приложения для социальной сети «ВКонтакте» на языке программирование C#

Резюме. В статье рассмотрены реализация простого desktop-приложения на C#. Для разработки desktop–приложения использовалась .NET Framework 4.0 и язык программирова- ния C#. Социальная сеть «ВКонтакте»