

A monastery website model used for entrepreneurial purposes

Mirela-Catrinel VOICU

Abstract— In this paper we present a website model on the presentation of Romanian monasteries. Religion, arts, history, culture and tourism are some of the reference points which constitute the attraction of this monastery study. Our website model can be used for tourists or also when we want to develop some kind of business. For example, if someone is interested on business development, including activities on school of painting on glass, painting of icons, religious music, etc., then our website model can be used as a helping tool.

Keywords— Websites, tourism, monasteries, Java, SQL Server.

I. INTRODUCTION

WITH more than five hundred orthodox monasteries and sketes, Romania seems to be the country with the highest number of orthodox monasteries of the world (see also [20]).

History, treasury, saints and wonders are the subjects displayed in many books or other presentation items, regarding the monasteries. In our study we are focused on building a website in order to present Romanian monasteries. We use [4] as basic information, where we find 305 of the most important Romanian monasteries. 177 of them are monasteries of monks and the others 128 are monasteries of nuns.

We find monasteries in Bucharest and in all the 41 counties of Romania, in the following way: Alba county - with 13 monasteries, Arad county - with 4 monasteries, Argeş county - with 17 monasteries, Bacău county - with 13 monasteries, Bihor county - with 4 monasteries, Bistriţa Năsăud county - with 5 monasteries, Botoşani county - with 9 monasteries, Brăila county - with 2 monasteries, Braşov county - with 7 monasteries, Bucharest - with 3 monasteries, Buzău county - with 5 monasteries, Călăraşi county - with 1 monasteries, Caraş - Severin county - with 7 monasteries, Cluj county - with 9 monasteries, Constanţa county - with 3 monasteries, Covasna county - with 2 monasteries, Dâmboviţa county - with 8 monasteries, Dolj county - with 5 monasteries, Galaţi county - with 5 monasteries, Giurgiu county - with 1 monasteries, Gorj county - with 10 monasteries, Harghita county - with 5 monasteries, Hunedoara county - with 2 monasteries, Ialomiţa county -

with 2 monasteries, Iaşi county - with 18 monasteries, Ilfov county - with 7 monasteries, Maramureş county - with 12 monasteries, Mehedinţi county - with 5 monasteries, Mureş county - with 3 monasteries, Neamţ county - with 19 monasteries, Olt county - with 4 monasteries, Prahova county - with 10 monasteries, Sălaj county - with 2 monasteries, Satu Mare county - with 6 monasteries, Sibiu county - with 3 monasteries, Suceava county - with 22 monasteries, Teleorman county - with 2 monasteries, Timiş county - with 5 monasteries, Tulcea county - with 5 monasteries, Vâlcea county - with 17 monasteries, Vaslui county - with 10 monasteries and Vrancea county - with 13 monasteries.

The monasteries were built in different centuries. Parts of some old monasteries have been destroyed over time. For this reason, in the case of the older monasteries, some parts were rebuilt in various centuries. Even if some monasteries are not old, they were built near the ancient religious sites, such as an old church. Now, considering the oldest part from each monastery, we find 1 monastery from the 12th century, 10 monasteries from the 13th century, 17 monasteries from the 14th century, 27 monasteries from the 15th century, 43 monasteries from the 16th century, 61 monasteries from the 17th century, 43 monasteries from the 18th century, 20 monasteries from the 19th century, 82 monasteries from the 20th century (55 monasteries were built after 1989) and 1 monastery from the 21th century.

Some from these monasteries are historical, UNESCO or architectural monuments.

For a monastery, we find between 1 and 5 holy patrons or religious celebration, depending also on the number of monastery churches. We find the *Dormition of the Virgin* holy patron in 52 monasteries, the *Birth of the Virgin* holy patron in 26 monasteries, the *Saint Nicholas* patron in 25 monasteries, the *Holy Trinity* holy patron in 23 monasteries, the *Saint George* patron in 19 monasteries, the *Apostles Peter And Paul* patron in 15 monasteries, the *Holy Virgin's Entering the Church* patron in 14 monasteries, the *Holy Archangels Michael and Gabriel* patron in 13 monasteries, the *Curing Spring* patron in 12 monasteries, the *Annunciation of the Blessed Virgin Mary* patron in 12 monasteries, the *Holy Virgin's Veil* patron in 11 monasteries, the *Prophet Elias* patron in 11 monasteries, the *St John the Baptizer* patron in 10 monasteries, the *Transfiguration of Jesus* patron in 10 monasteries, the *Ascension of Jesus* patron in 8 monasteries, the *Parascheva the Pious* patron in 7 monasteries, the *Saint Demetrius* patron in 7 monasteries, the *Beheading of St. John the Forerunner* patron in 7 monasteries, the *Saints Constantine & Helen* patron in 6 monasteries, the *Descent of*

Manuscript received January 27, 2011

Mirela-Catrinel Voicu is professor at the Faculty of Economics and Business Administration, West University of Timisoara, ROMANIA (phone: +40-256-592551; fax: +40-256-592500; e-mail: mirelavoicu@yahoo.fr).

the *Holy Ghost* patron in 6 monasteries, the *Sunday of all Saints* patron in 5 monasteries, the *Great Martyr Panteleimon* patron in 4 monasteries, the *Resurrection* patron in 3 monasteries, the *Ascension of the Holy Cross* patron in 3 monasteries, the *Saint Antony the Great* patron in 3 monasteries and the *Saint Andrew* patron in 2 monasteries. We find others 23 patrons, each of them as patrons in only one monastery.

In some monasteries, we can find saint relics or important personalities buried inside.

From the types of treasury objects of the Romanian monasteries, we recall (see also *Figure 2*, the *Significant objects* list):

- school, school of painting on glass, typography, school of priests, theological seminary, carpet weaving workroom, embroidery workroom, painting school, cultural center, center of famous painters on glass, Greek school, orthodox spirituality and culture courses, monastic theological seminary, tailor's workroom, ecclesiastical painting workroom, carpentry workroom, painting workroom, knitting workroom, elementary and secondary school, library, eparchy residence, school of religious painting, school of church singers;
- the miracle-making icon and holy spring;
- old books collection, ecclesiastical museum collection, wood icons, glass icons, ethnographic objects, religious items, historical monuments, sculpture, architecture, objects of medieval art, paintings, mural painting, heritage objects, folk museum collection, bibliophile property, collection of historical and cultural value wines, byzantine architecture, typography, state archives, artistically embroidery, chrim church, old documents, icons in relief, chronicle mural, manuscripts, tailoring for ecclesiastical vestments, painting with artistic and historical value, tapestries, ceramics, wax painting, engravings, rudimentary printing presses, portrait gallery, religious ornaments;
- exhibitions dedicated to the life and activities of bishops, museum, exhibition of icons, religious art exhibitions, ecclesiastical music concerts, memorial houses, famous cultural center (calligraphers, miniaturists, historians), printers, ecclesiastical old art collection, coin museum;
- bishops necropolis and princely necropolis;
- religious assistance, social assistance and asylum of elderly;
- resort, center of treatment, sanatorium center and sulphurous springs;
- vineyards;
- caves.

Starting from the information presented above, in the *Section II*, we built a database model, used to store the website data. In the *Section III*, we present the website.

II. DATABASE PRESENTATION

The website information is stored in the database presented in *Figure 1*. Now, we present each table from the database and the signification of fields.

In the *Monasteries* table we use the followings fields: *ID_monastery* – for the unique identification of the monastery

in database, *Name*, *County*, *Map_area* (for the map area where we can find the monastery), *Type* (the values in this field are monks or nuns), *MYear* (this field refers to the year when the oldest part of the monastery was built), *MCentury* (this field refers to the century when the oldest part of the monastery was built – we use this field because for many oldest monasteries we do not have the value for *MYear*; in the case of these monasteries only the century is specified), *Monument* (in this field we save value as: UNESCO, historical, architectural), *Address*, *Telephone*, *E_mail*, *Web_address* (in this field we save the web address of each monastery; we consider that, at this web address, the important information on the monastery's present-day activity is displayed) and *Abbot_prioress*.

In the *Patrons* table we use the fields *ID_patron*, *PDay*, *PMonth* (for many patrons, in the orthodox calendar we find a corresponding date, e.g. 6 December for the *Saint Nicholas* holy patron – in this case, we save the 6 value for *PDay* and the *December* value for *PMonth*) and *Observation* (in the case of certain patrons, we do not have a fixed date in the orthodox calendar – the event takes place on different dates, depending on the year; e.g. for the *Ascension of Jesus* patron, in the field *Observation* we save the 40 days after Easter value).

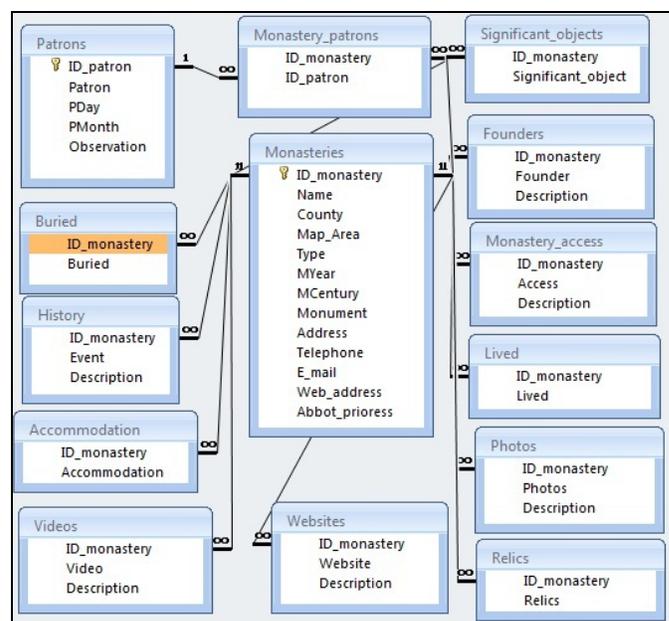


Fig. 1 Database structure

In the *Monastery_patrons* table, in the case of each monastery, we specify all patrons.

To describe the monastery history, we use the *History* table. Here we can find the building or rebuilding years and all the remarkable events that affected the monastery.

In the *Founders* table we save data on the founders. For different cases of monasteries, we find one or more founders, corresponding to different stages of the monastery's history. Also, we can find a person as founder in the case of more than one monastery (e.g. *Stephen the Great*).

In the case of different monasteries we can find saint relics. We save this information in the *Relics* table.

Also, in some monasteries, certain historical personalities have been buried and we save the corresponding data in the *Buried* table. Over time, remarkable personalities have lived for short or long term, inside of monasteries - we save these data in the *Lived* table.

Many websites present information on different monasteries. These websites can be dedicated to a monastery or to monasteries from a certain Romanian region or to Romanian orthodox life. Also, on *YouTube* we find many video files with wonderful movies presenting different monasteries (see, for example, [12]-[18]). In the *Websites* table we save the address of such websites and we provide a short description. However, in this table we do not consider the monastery's official web address (this web address is saved in the *Monasteries* table). In order to display, in our website, image and video files, we use the *Photos* and *Videos* tables.

For visiting Romanian monasteries, we have motorway or railway access. We specify this information in the *Monastery_access* table.

Many Romanian monasteries offer accommodation and we specify this information the *Accommodation* table.

III. WEBSITE PRESENTATION

In *Figure 2* we present the first page of our website. If for all components presented in *Figure 2*, we have selected the *all*

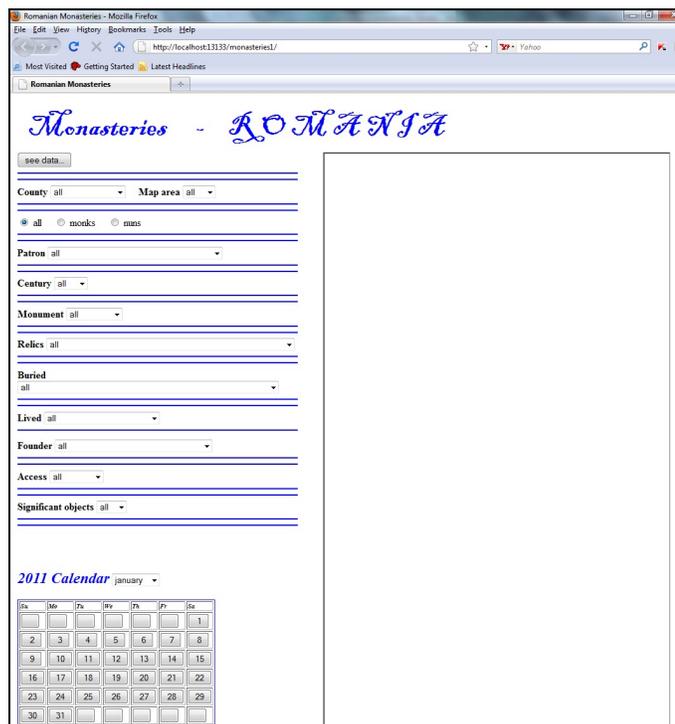


Fig. 2 Website – the first page

value, then we can view all monasteries ordered by county, as in *Figure 4*.

We can select a county or a map area, where we want to view the data on monastery. If we select a county, from the *map area* list, the application automatically selects the *all* value. Reversely, when we select a certain map area, from the *county* list, the application automatically selects the *all* value.

This operation excludes possibilities when the user chooses a county and a map area corresponding to the cases in which the chosen county is not in the chosen map area (such a situation means zero monasteries result records).

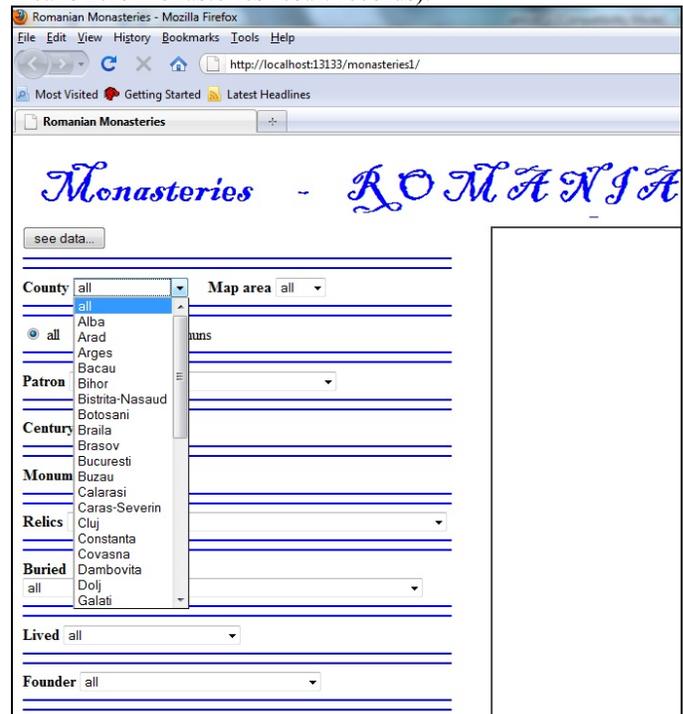


Fig. 3 Website – selecting the desire data

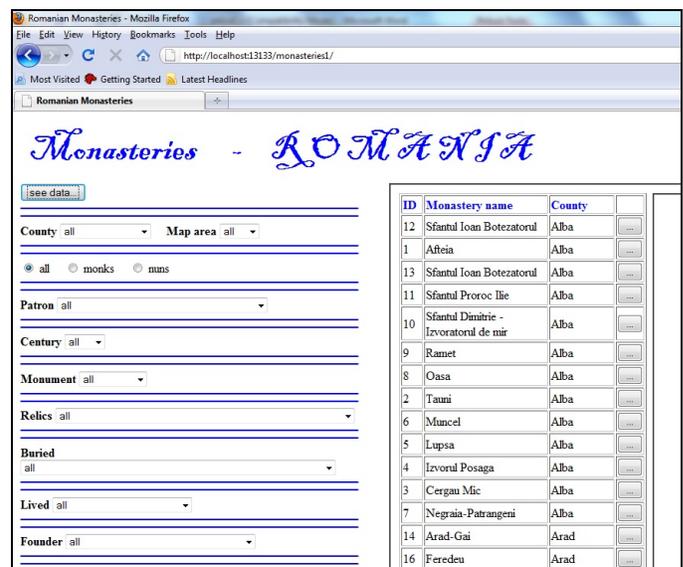


Fig. 4 Website – selecting a monastery

Using HTML list components, we can select a single county or a single map area. We also can use other HTML components for selection. For example, we can use for each county a checkbox component. But this means 42 checkbox components only for selecting the desired counties (41 counties and Bucharest). In this way, we consider that the HTML components used in order to build the website are at the discretion of the website builder.

To select the monastery type (monks, nuns or both), we use radio components. Out of all the three possibilities, the user selects only one radio component. For *patron*, *century*, *monument*, *relics*, *buried*, *lived*, *founders*, *access* and

significant objects (see Figure 2), we use list components. In each case we can view data for all elements or only for a single selected item (see also Figure 3, where we select a county).

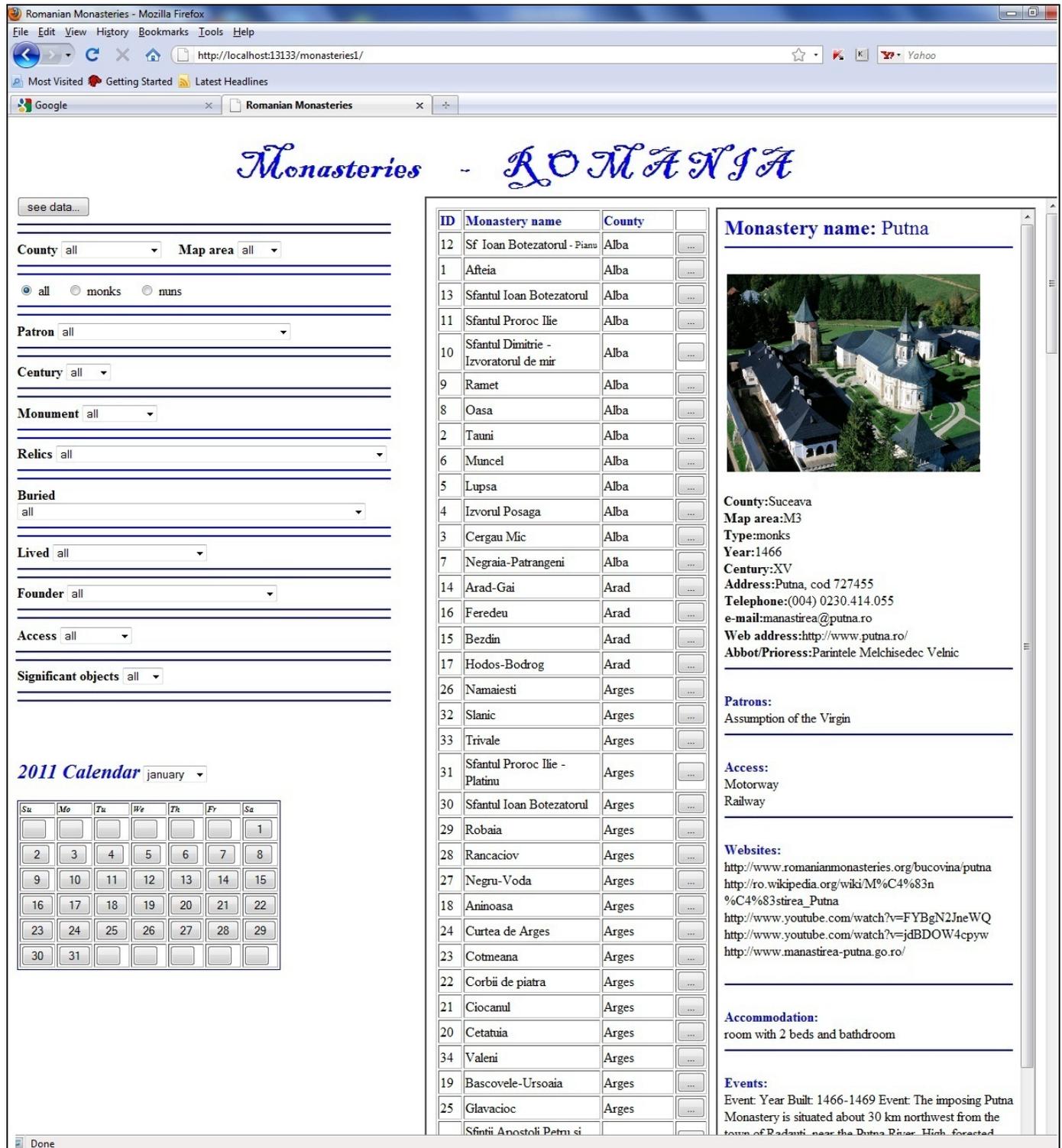


Fig. 5 Website – monastery data

With a click on the “see data...” button (see Figure 3) we obtain in Figure 4 all monasteries which correspond to our selection, from the left side of the webpage.

With a click on the monastery “...” button (see Figure 4), we can view data on the selected monastery (see Figure 5).

In *Figure 5*, using the scroll bar, we can view all the data corresponding to the selected monastery: county, map area, type, year and the century when it was built, if it is a monument, its address, telephone, the e-mail, the web address, the abbot or the prioress, the patrons, access, the websites, the information on accommodation possibilities, the historical events, the significant objects, information on relics, information on person buried in the monastery, information on the person who have lived at the monastery, information on founders and monastery image and video files. In *Figure 5*, Putna Monastery (which has the largest monastery museum from Romania) is presented.

We can use a calendar (see *Figure 5*) in order to present information from the Orthodox calendar. When we select a date, we can view the corresponding religious information.

IV. CODE PRESENTATION

We present our implementation using a *Java* code and a *Microsoft SQL Server* database.

A. The first page of the website

In order to obtain the first website page, which is presented in *Figure 2*, we use the following *JSP* file:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%> <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN ""http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><title>Romanian monasteries</title>
<!-- In Figure 2 we have a calendar. For the month selection, we use a drop-down list component and for the day selection, we use button components. When we select a month, in the HTML table of days, we can view the corresponding days. For this operation, we use the days function in the following script: --><Script Language="JavaScript">
function days() {
if(document.form1.month.selectedIndex==0) {
// The days of January – the year 2011
document.form1.b11.value=""; document.form1.b12.value="";
document.form1.b13.value=""; document.form1.b14.value="";
document.form1.b15.value=""; document.form1.b16.value="";
document.form1.b17.value="1";document.form1.b21.value="2";
document.form1.b22.value="3"; document.form1.b23.value=" 4";
document.form1.b24.value=" 5"; document.form1.b25.value=" 6";
document.form1.b26.value=" 7"; document.form1.b27.value=" 8";
document.form1.b31.value=" 9"; document.form1.b32.value="10";
document.form1.b33.value="11"; document.form1.b34.value="12";
document.form1.b35.value="13"; document.form1.b36.value="14";
document.form1.b37.value="15"; document.form1.b41.value="16";
document.form1.b42.value="17"; document.form1.b43.value="18";
document.form1.b44.value="19"; document.form1.b45.value="20";
document.form1.b46.value="21"; document.form1.b47.value="22";
document.form1.b51.value="23"; document.form1.b52.value="24";
document.form1.b53.value="25"; document.form1.b54.value="26";
document.form1.b55.value="27"; document.form1.b56.value="28";
document.form1.b57.value="29"; document.form1.b61.value="30";
document.form1.b62.value="31"; document.form1.b63.value=" ";
document.form1.b64.value=" "; document.form1.b65.value=" ";
document.form1.b66.value=" "; document.form1.b67.value=" ";}
```

// We use the same kind of code for the other months

```
...
}
</script></head>
<body><form name="form1" method=post target="frame1" action=" http://localhost:8080/monasteries2/monasteries">
<table><tr><td><center><b> <font color="blue" size="12" face=" Blackadder ITC"> Monasteries &nbsp; - &nbsp; &nbsp;R O M A N I A &nbsp; &nbsp;</font>
<input type="submit" value="VIEW DATA"></b></center>
</td></tr> <tr><td><iframe src ="p1.jsp" width="1250" height="500" name="frame1"></iframe></td></tr><tr><td>
<br><b><i>
```

<!-- The calendar from Figure 2 -->

```
<font size=5 color="blue">2011 Calendar</font></i></b>
<select name="month" onchange="days()"><option checked>
January </option><option>February</option><option>March
</option><option>April</option></select>
<br><br><table border=1 bordercolor="blue"
bordercolorlight="yellow" bordercolordark="yellow">
<tr><td><b><i><font size=1>Su</font></i></b></td>
<td><b><i><font size=1>Mo</font></i></b></td>
<td><b><i><font size=1>Tu</font></i></b></td><td>
<b><i><font size=1>We</font></i></b></td>
<td><b><i><font size=1>Th</font></i></b></td>
<td><b><i><font size=1>Fr</font></i></b></td>
<td><b><i><font size=1>Sa</font></i></b></td></tr>
```

<!-- For the days of month, we use command button components and, if we need, we can add in the script different functions associated with their click events. The days displayed in the form are the days of the January month. - ->

```
<tr><td><input type="button" name="b11" value=" "></td>
<td><input type="button" name="b12" value=" "></td>
<td><input type="button" name="b13" value=" "></td>
<td><input type="button" name="b14" value=" "></td>
<td><input type="button" name="b15" value=" "></td>
<td><input type="button" name="b16" value=" "></td>
<td><input type="button" name="b17" value=" 1"></td>
</tr><tr>
<td><input type="button" name="b21" value=" 2"></td>
<td><input type="button" name="b22" value=" 3"></td>
<td><input type="button" name="b23" value=" 4"></td>
<td><input type="button" name="b24" value=" 5"></td>
<td><input type="button" name="b25" value=" 6"></td>
<td><input type="button" name="b26" value=" 7"></td>
<td><input type="button" name="b27" value=" 8"></td>
</tr>
<!-- We use the same type of HTML tags for all the other days of month- ->
</table></td></tr></table></form></body></html>
```

The *JSP* source file of the *iframe* component from *Figure 2* has the following content:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

```
<html><head><meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
<title>JSP Page</title></head> <body> </body></html>
```

In many stages, our programming code refers to the database update. This code is the same in each situation, for this reason we define a function named *database_update* which will be called when we need it.

For the database connection we use a user, a password and a URL. For each of these items, we use variables which are declared and initialized in an encrypted servlet, which is extended to the other servlets of our project.

In the following parts of our source code presentation we use *servlets*. The header and the footer of the *HTML* files which will be created, have the same general structure. For this reason, we also use two functions (named *header* and *footer*) which contain the header or the footer of the web pages. These functions will be called, when we need them.

In order to define the functions and declare the variables presented above, we use a *servlet*, named *Servlet1*.

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

public class Servlet1 extends HttpServlet {
protected String user = ""; protected String password = "";
protected String dbURL = "jdbc:odbc:monastery";
protected void database_update(String sql_statement){
try {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection c = DriverManager.getConnection(dbURL, user,
password); Statement s = c.createStatement();
s.executeUpdate(sql_statement);s.close();
}
catch (ClassNotFoundException a) {}
catch (SQLException a) {}
}

public void header(String title, PrintWriter out) throws
IOException{
out.println("<html><head><title>"+title+"</title>");
}

public void footer( PrintWriter out) throws IOException{
out.println("</body></html>");
}
}
```

B. Selecting monasteries using specific different criteria

We can select monasteries, specifying different criteria in the web page which is presented in *Figure 3*. In order to obtain this web page, we use the following servlet:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

public class monasteries extends Servlet1 {
/* In Figure 3, we use some drop-down list components in
order to specify different criteria. The associated codes of
these components are the same in each case. For this reason,
we now define a function, which will be called when we need
it. */
void f_criteria(String o_criteria, String n_o_criteria, String
sql_statement, PrintWriter out){
String o="";
out.println("<b> "+o_criteria+"</b><select
name='"+n_o_criteria +
"'\> <option value='all'\>all</option>");
try {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
// The variables dbURL, user and password were declared in
Servlet1
Connection c = DriverManager.getConnection(dbURL, user,
password);Statement s = c.createStatement();
ResultSet r = s.executeQuery(sql_statement);
while (r.next()) {
o=r.getString(1);
out.println("<option value='"+o+"'\>"+o+"</option>");
}
r.close();s.close();
}
catch (ClassNotFoundException a) {}
catch (SQLException a) {}
out.println("</select>");
}

protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
header("Romanian monasteries",out);
out.println("</head><body><form name='form1' target='"+
"frame2" action='http://localhost:8080/monasteries2/' +
"select_monasteries\" method=post >");
out.println("<table> <tr><td valign='top'\>");
out.println("<input type='submit' value='see data...'\>");
out.println("<br><hr size=2 color='blue'\><hr size=2"+
"color='blue'\>");

// In a drop-down list component, we add the counties
sql_statement="select distinct county from monasteries order
by county"; f_criteria("County", "county", sql_statement, out);
out.println("<br><hr size=2 color='blue'\><hr size=2
color='blue'\>");
```

In order to display the map areas, we use the same kind of programming code as in the county case.

```
//Using radio components, we can select the monastery type
out.println("<input type=radio name='mtype\" value='"+
"all\" checked> all<input type=radio name='mtype\""+
```

```
"value=\"monks\"> monks <input type=radio name="+
"\mtype\" value=\"nuns\"> nuns <br>");
out.println("<br><hr size=2 color=\"blue\"><hr size=2"+
"color=\"blue\">");
```

In order to display patrons, we use the same kind of programming code as in the county case.

```
// In a drop-down list component, we add the centuries
out.println(" <b> Century </b> <select name=\"century\">");
out.println("<option value=\"all\">all</option><option"+
"value=\"XII\">XII </option><option value=\"XIII\">XIII"+
"</option><option value=\"XIV\"> XIV</option><option "+
"value=\"XV\">XV</option><option value=\"XVI\">"+
"</XVI</option><option value=\"XVII\">XVII</option> "+
"</option value=\"XVIII\">XVIII</option><option value "+
"=\"XIX\">XIX</option><option value=\"XX\">XX"+
"</option><option value=\"XXI\">XXI</option></select>");
out.println("<br><hr size=2 color=\"blue\"><hr size=2
color=\"blue\">");
```

In order to display the monuments (relics, the personalities buried at the monasteries, the personalities who lived at the monasteries, the founders, the access ways to the monasteries and the significant objects) we use the same kind of programming code as in the county case.

```
out.println("<td valign=\"top\"><iframe src =\"p1.jsp\" "+
"width=\"750\"height=\"1300\" name=frame2>"+
"</iframe></td></tr></table></form>");
footer(out);
} finally {
    out.close(); }
}}
```

C. Selecting a monastery

In *Figure 3* we specify the criteria needed to detect monasteries. In the web page from *Figure 4* we find all corresponding results. In *Figure 4* we can view the name and the county of each monastery result. With a click on its associated submit button, we can view all monastery data (see *Figure 5*). In order to obtain the web page presented in *Figure 4*, we use the following servlet:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
public class select_monasteries extends Servlet1 {
    protected void processRequest(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            header("Select monasteries",out);
            out.println("</head><body><table><tr><td valign=\"top\">");
```

```
out.println("<form name=\"form2\" target=\"frame3\""+
"action=\"http://localhost:8080 /monasteries2/monastery \"+
" method=post >");
String sql_statement=""; String county=""; ma="";
county=request.getParameter("county");
ma=request.getParameter("ma");
```

```
//We detect monasteries by county
if(county.equals("all")){
    sql_statement="select distinct ID_monastery into"+
"IDMmonasteries from monasteries";
}else{
    sql_statement="select distinct ID_monastery into"+
"IDM_monasteries from monasteries where
county=\""+county+"\"";
}
// The function database_update was defined in Servlet1
database_update(sql_statement);
```

We detect monasteries by map area using the same type of code as in the county case.

```
//We detect monasteries by monastery type
String mtype="";
mtype=request.getParameter("mtype");
if(mtype.equals("monks")){
    sql_statement="select distinct ID_monastery into"+
"IDM_mtype from monasteries where type='monks'";
}
if(mtype.equals("nuns")){
    sql_statement="select distinct ID_monastery into"+
"IDM_mtype from monasteries where type='nuns'";
}
if(mtype.equals("all")){
    sql_statement="select distinct ID_monastery into"+
"IDM_mtype from monasteries";
}
database_update(sql_statement);
```

In order to detect monasteries by holy patron (century, the monuments, relics, the personalities buried at the monasteries, the personalities who lived at the monasteries, the founders, the access ways to the monasteries and the significant objects) we use the same kind of programming code as in the county case.

```
// We detect the monasteries corresponding to all criteria
sql_statement="delete * from IDMonasteries where" +
"ID_monastery not in (select ID_monastery from IDM_ma)";
database_update(sql_statement);
```

In the same way, we delete from *IDMonasteries* table all records which have not the *ID_monastery* in the corresponding field from the *IDM_mtype* table (the *IDM_patron*, *IDM_century*, *IDM_monument*, *IDM_relics*, *IDM_buried*, *IDM_lived*, *IDM_founder*, *IDM_access* and *IDM_so* tables).

We delete from database the following temporary tables: *IDM_ma*, *IDM_mtype*, *IDM_patron*, *IDM_century*, *IDM_monument*, *IDM_relics*, *IDM_buried*, *IDM_lived*, *IDM_founder*, *IDM_access* and *IDM_so*.

```
// We detect the monasteries which correspond to all criteria
sql_statement="select * from Monasteries where
id_monastery in (select "+ "id_monastery from
IDMonasteries) order by county";
```

```
// We display the name and the county for each monastery
result
```

```
out.println("<table border=1><tr><td><b><font color="+
"<\/>IDfont<\/b><\/td><td><b><font color=\"blue\"+
\">Monastery name<\/font> <\/b><\/td><td><b><font\"+
\"color=\"blue\">County<\/font><\/b><\/td><td> <\/td><\/tr>");
int nb=0; int hh=0;
```

```
try {
rvlet1
Connection c = DriverManager.getConnection(dbURL,
user, password);Statement s = c.createStatement();
ResultSet r = s.executeQuery(sql_statement);
while (r.next()) {hh=r.getInt(1); nb++;
out.println("<tr><td><input type=\"hidden\" value="+
String.valueOf(hh)+
"name=hh"+nb+">"+String.valueOf(hh)+"<\/td> "+
"<td>"+r.getString(2)+
"<\/td><td>"+r.getString(3)+"<\/td>");
out.println("<td><input type=\"submit\" name=button"+
String.valueOf(nb) +
value=\"...\"
onclick=\"f1()\"><\/td><\/tr>");
}
r.close();
s.close();
}
catch (ClassNotFoundException a) {}
catch (SQLException a) {}
sql_statement="drop table IDMonasteries";
database_update(sql_statement);
```

```
// We save the number of monasteries, using a hidden
component
```

```
out.println("<\/table><td><input type=\"hidden\" value="+
nb+" name= "+ "\"h1\"> <\/td>");
out.println("<\/form> ");
out.println("<\/td><td valign=\"top\"><iframe src =\"p1.jsp\"
width= "+ "\"375\" height=\"1300\"
name=\"frame3\"><\/iframe><\/td><\/tr><\/table>");
footer(out);
} finally {
out.close();}
}
}
```

D. Data on a monastery

In the web page, which is presented in the *Figure 5*, we can observe the way in which the monastery data are displayed in website. There, we use the following servlet:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
```

```
public class monastery extends Servlet1 {
//In order to display data, we use the following function:
```

```
protected void monastery_data(String sql_s){
String a1= " ";
try {
Connection c = DriverManager.getConnection(dbURL, user,
password);
Statement s = c.createStatement();
ResultSet r = s.executeQuery(sql_s);
while (r.next()) {
```

```
// We read the monastery data
```

```
a1=r.getString(1);
if(a1!=null){
```

```
// We display the monastery data
```

```
out.println(a1+"<br>");
```

```
}
}
r.close();
s.close();
}
catch (ClassNotFoundException a) {}
catch (SQLException a) {}
out.println(" <hr size=2 color=\"blue\"><br>");
}
```

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
```

```
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
header("Monastery",out);
out.println("<\/head><body>");
String b=""; int nom=0; int idm=0; int i=0;
nom=Integer.parseInt(request.getParameter("h1"));
String a3="", a4="", a5="", a6="", a7="", a8="", a9="",
a10="", a11=""; String a12="", a13="";
```

```
/* We detect the selected submit button from the web page
presented in Figure 4 */
```

```
for(i=0;i<=nom;i++){
b=request.getParameter("button"+String.valueOf(i));
if(b!=null){
idm=Integer.parseInt(request.getParameter("hh"+String.v
alueOf(i)));
String sql_statement="";
```

```
// We detect the selected monastery
```

```
sql_statement="select * from monasteries where
id_monastery="+idm;
try {
Connection c = DriverManager.getConnection(dbURL,
user, password);
Statement s = c.createStatement();
ResultSet r = s.executeQuery(sql_statement);
```

```
while (r.next()) {  
  
// We read the monastery data  
a12=r.getString(12);a3=r.getString(3);a4=r.getString(4);  
a5=r.getString(5); a6=r.getString(6); a7=r.getString(7);  
a8=r.getString(8);a9=r.getString(9); a10=r.getString(10);  
a11=r.getString(11); a13=r.getString(13);  
  
// We display the monastery data  
out.println("<font size=5 color=\"blue\"><b>  
Monastery" + "</font><br><hr size=" +  
r.getString(2)+"</font><br><hr size=" +  
color=\"blue\"> br>");  
out.println("<table><tr><td><img src=\"photos/m"+ idm  
+ "p1.jpg\" + \"width= 300></td></tr></table>");  
if(a3!=null){  
out.println("<b> County: </b>"+a3+"<br>");  
}else{  
out.println("<b> County: </b><br>");  
}  
  
/*We display map area (monastery type, the year of  
building, the century of building, the monument type, the  
monastery address, telephone, e-mail, web address and  
the abbot/prioress) in the same way as one used for  
county. */  
}r.close();s.close();  
}  
catch (ClassNotFoundException a) {}  
catch (SQLException a) {}  
out.println("<hr size=2 color=\"blue\"><br>");  
  
// We detect the monastery patrons  
sql_statement="select patron from patrons where"+  
"id_patron in (select Id_patron from monastery_patrons  
where id_monastery="+idm+")";  
out.println("<font size=3 color=\"blue\"><b>Patrons: "+  
"< </font>>br>");  
monastery_data(sql_statement);  
/* In the same way (as one presented for holy patrons), we  
detect the monastery access ways, websites, accommodation  
conditions, descriptions, significant objects, relics, the  
personalities buried, the personalities who lived and the  
founders.*/  
}}  
footer(out);  
} finally {  
out.close();}}}
```

V. CONCLUSION

We can find many sources on Romanian monasteries (books, guides, maps, etc.). Also, many websites (from Romania or even abroad) present one or more monasteries.

Many monasteries have a very long history including very many events. Also, many of them have a very impressive treasury.

In our study we have presented a website model which helps the user to find monasteries according to some criteria. We

can find a lot of websites on monastery presentation. But many of them, even if they provide much information, they do not allow us to find monasteries in relation to some conditions.

For example, in *Figure 2*, at *Significant objects* if we select *library*, we can find all the monasteries that have library. Here, we recall that the *Neamt* monastery has a library with 18000 volumes, the *Agapia* monastery has a library with 15000 volumes, the *Cernica* monastery has a library with 14000 volumes, the *Hurez* monastery has a library with 4000 volumes etc. In *Section I*, we have presented the very large set of significant objects that can be found in Romanian monasteries. The selections from our websites can help the user in order to have a cultural image on monasteries.

Also, our website can be a wonderful guide for tourists. With good information from website, the user can create an itinerary for his visit.

In the same time, around of monastery life and traditions, we can observe many activities with business opportunities. Our website model provide very precious information and can be used as a helping tool.

REFERENCES

- [1] Ceballos Sierra, Fco. Javier – *Java 2* – Ra-Ma Publisher, Madrid, 2008
- [2] Chopra Vivek, Sing Li, Jeff Genender – *Apache Tomcat 6* – Anaya Multimedia Publisher, 2008, Madrid
- [3] Collection *Orthodox monasteries*- Ed. Deagostini, Romania, 2010
- [4] *Touristic and Road Map – Romanian Monasteries* – Ed.Niculescu, Romania, 2009
- [5] *Map of Romanian Monasteries* – Ed Amco Press, Romania, 2009
- [6] Marchis, Iustin – *Romanian monasteries and churches: Transilvania* – Ed. Noi Media Print, Romania, 2008
- [7] Voicu Mirela-Catrinel, Pirtea Marilen, Varga Roxana Ioana - *Website about Summer Offers at the Romanian Black Sea Resorts* - The 3rd WSEAS Conference on CULTURAL HERITAGE and TOURISM, (CUHT '10), [Corfu Island, Greece, July 22-24, 2010](http://www.crestinism-ortodox.com/orthodoxy/churches/romania/)
- [8] Voicu Mirela-Catrinel, Pirtea Marilen, Bazavan Sandra - *Travel guide. Websites-* The 3rd WSEAS Conference on CULTURAL HERITAGE and TOURISM (CUHT '10), [Corfu Island, Greece, July 22-24, 2010](http://www.crestinism-ortodox.com/orthodoxy/churches/romania/)
- [9] <http://www.crestinism-ortodox.com/orthodoxy/churches/romania/rommonasteries/>
- [10] <http://www.goarch.org/chapel/chapel/calendar>
- [11] <http://www.iconograms.org/>
- [12] <http://il.youtube.com/watch?v=eoJXWH1PwJ4>
- [13] http://il.youtube.com/watch?v=h2NMG4X_rV0
- [14] http://il.youtube.com/watch?v=9ACS0n_7zfg
- [15] <http://il.youtube.com/watch?v=gpcJSbeY870>
- [16] <http://il.youtube.com/watch?v=nOA3yoK9dW8&feature=related>
- [17] <http://il.youtube.com/watch?v=pyaiKfQn9w&feature=related>
- [18] <http://il.youtube.com/watch?v=8UwVhovvxEo&feature=related>
- [19] http://www.orthodoxphotos.com/Monasteries_and_Churches/Romanian/
- [20] <http://www.orthodox-monasteries.com/>
- [21] <http://www.ortodox.ro/manastiri/>
- [22] <http://www.romanian-monasteries.go.ro/>
- [23] <http://www.romanianmonasteries.org/>
- [24] <http://www.romaniatourism.com/>

M. C. Voicu was born in Romania. In 1995, she graduated from the Faculty of Mathematics and Computer Sciences, West University of Timisoara. She received the MSc degree in Applied Mathematics, Informatics in Economy and Computer Sciences from the West University of Timisoara. She followed a training course for PhD thesis at the National Institute for Statistics and Economic Studies, Paris, France. She received her PhD in 2001 from the University of Timisoara, Romania (with the "Cum laude" distinction) and from the University of Paris 13, France (with the "Très honorable avec

félicitations" distinction). Currently she is a Professor at the Department of Economic Informatics, within the Faculty of Economics and Business Administration, West University of Timisoara, Romania, where, since 1995, she has held several academic positions. Her activity includes Programming and Internet Programming, Informatics in Economy, Databases, OOP, Data structures. Through the collaboration program between the Faculty of Economics and Business Administration and the Faculty of Mathematics and Informatics of the West University, she has held classes with international participation, in postgraduate education, in the section of "Mathematic modeling in economics and applied sciences" Exchange Rate Evolution Models subject introduced within the program due to the original contributions in her PhD. During the collaboration between the Faculty of Economics and Business Administration, West University of Timisoara and CUOA Italy, she has taught a postgraduate class for the section of

"Management of Business and Public Administration", regarding the Internet. She has 79 papers in conference proceedings or refereed journals (from these papers, 26 have been presented or published abroad). She has published 8 books (1 book in France, 1 book in Germany and 6 books in Romania). She is a reviewer and a member in international program committee of various WSEAS conferences from abroad, reviewer at "Journal of Knowledge, Communications and Computing Technologies", member in the teams of 9 research projects (one of which is international) and project manager for one research project. Since 2002, she is a member of INFOREC (Romanian Association for Economic Informatics Training Promotion), since 2005, she is a member of WSEAS (World Scientific and Engineering Academy and Society) and since 2009, she is a member of SCT (Society for Computing Technologies).