Website Model for Top 10 Travel Guides

MIRELA-CATRINEL VOICU, MARILEN PIRTEA, SANDRA BĂZĂVAN
Faculty of Economics and Business Administration
West University of Timisoara
ROMANIA
mirela.voicu@feaa.uvt.ro, marilen.pirtea@feaa.uvt.ro, Sandra.Bazavan@gmail.com, http://www.feaa.uvt.ro

Abstract: - *Top 10* is a famous series of travel guides in the world. Starting from the data presented in these travel guides, we propose a website model. Our model supposes a single implementation, which can be used for all city destinations. The website can be used easily and includes plenty and various information. The displayed data in the website pages are stored in a database. These data can be quickly modified or deleted. We present our implementation in *Java*.

Key-Words: - Travel guides, websites, Java.

1 Introduction

In order to increase the pleasure of each travel, people must be informed on the opportunities of destination. These opportunities refer to transport, accommodation, famous places for visiting, restaurants, events and festivals, shopping, etc.

Generally, for information, people use the Internet or travel guides.

Number one in UK, and famous in the world, *Top 10* from DK(Dorling Kindersley) has a small size format, easy to carry with you in tour and it contains information in a concentrated form, which can be studied quickly. This is what you need when you visit a new city.

For each important city area, a street map is presented in "Top 10" travel guide. For each street map, the most known places which you must view or visit are presented. In various situations, we find a price (or rate) estimation - for a museum ticket, for a hotel room, etc. In this way, when we arrived in a certain city area, we already know what we want to visit. It doesn't matter if people travel for the holidays or for business, time is still limited. For this reason, it is very important to have quickly and easily much information regarding the new places. *Top 10* travel guide is a perfect tool on this problem.

In this paper, we would like to present a website model, which can be used as a travel guide. Our model refers to the implementation way that can be used in order to be able to populate the site quickly with as many information as needed.

Firstly, in the *Section 2*, we present the information type which, generally, can be found in these websites and starting from this information we build the database. In the *Section 3* we present our website model.

2 Database presentation and website building

2.1 Database information presentation

In order to estimate the general structure of database, we start from a particular case. Here, we choose Paris.

The general information includes, but is not limited to: churches, cathedrals, chapels, museums, art galleries, monuments, palace, fortresses, historical monuments, modernist or famous buildings, markets, hotels, cafes, terraces, bars, restaurants, parks, shopping centers, festivals, events, panoramas, children attractions, transport, tourist tours, arrivals, information sources, etc.

In order to store this information, we need a database with various tables and different structures, as followings:

We can have a table named *Churches* with the fields: *ID_churches* - Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area*–Text, *Web_address* –Text, *Minimum_price* – Number, *Maximum_ price* – Number, *Price_currency* – Text, *Description* - Text and *Religion*-Text in which we save information on churches. The *ID_churches* field is the field used for the unique identification of the records. Its value is given by the program. In this case, the *Name* field refers to the church name. The *Map_Area* field refers to the squares from the street map (in this case, Paris street map). The *Web_address* field provides you the web address of the place that you want to visit (in this case, a church). For many places, the entry is not free, for this reason, we specify, here, two prices – minimum and maximum. Generally, for children, for students or other person categories we find a discount. In this way, we don't present all prices, but we present the interval of value for the ticket price. Here, we use the *Price_currency* field for specifying the currency used for payment. The *Description* field is used for a short description on the corresponding church and also, it can contain information like time table, a near metro station, etc. In the *Religion* field it will be specified the church's religion.

In the presentation of the following tables, we will specify only the significance of the fields with new names. For example, the *Price_currency* field has the same significance in each table. For this reason, we will do not recall its signification in each case.

For information on hotels, we can use a table, named *Hotels*, with the following fields: *ID_hotels*- Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area*–Text, *Type* – Text, *Description* - Text, *Telephone* – Text, *Web_address* –Text and *Rate_symbol* – Text. In the *Rate_symbol* field we will have values as €, €€, € €€, etc. In this paper, with "€"we represent a rate between 1-30 euro, with "€€"we represent a rate between 31-60 euro, etc. In case of another currency, we will use the corresponding symbols (as £, $, etc.).

In a table with data on *museums*, we can have the fields: *ID_museums* – Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area*–Text, *Web_address* –Text, *Minimum_price* – Number, *Maximum_ price* – Number, *Price_currency* -Text, *Description* - Text and *Museum_Type* – Text. In the *Museum_Type* field we can save data like: art, nature, history, etc.

For *palaces* and *historic buildings*, we can have fields like: *ID_palaces*– Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area*–Text, *Web_address* –Text, *Minimum_price* – Number, *Maximum_ price* – Number, *Price_currency* - Text, *Description* – Text and *Year* – Number. The *Year* field refers to the year when the building or palace was finished.

In a table regarding *markets*, we can use the following fields: *ID_markets*– Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area*– Text, *Description* - Text and *Type* – Text. In the *Type* field, we store data like vegetables, food, goods, flowers, birds, etc.

For *cafes and terraces* or for *bars and restaurants*, we can use tables with the following fields: *ID_cafes_terraces*– Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area*–Text, *Description* - Text, *Price_symbol* – Text and *Type* – Text.

For *parks and gardens*, we can use a table with the following fields: *ID_parks*– Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area*– Text, *Description* - Text, *Minimum_price* – Number, *Maximum_ price* – Number, *Price_currency* –Text and *Type*- Text. In the *Type* field we have values as: park, botanical garden, garden, etc.

For shopping centers, we can have a table with the following fields: *ID_shopping_centers*– Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area*–Text, *Description* – Text and *Type* – Text.

On *festivals and events,* we can have tables with the following fields: *ID_festival_events*- Number- Primary Key, *Name* – Text, *Address* – Text, *Map_Area* – Text, *Web_address* –Text, *Minimum_price* – Number, *Maximum_ price* – Number, *Price_currency* –Text and *Description* - Text.

For *panoramas*, we can use a table with the fields *ID_panoramas*- Number - Primary Key, *Name* – Text, *Map_Area*–Text and *Description* - Text.

On *children attractions*, we can use a table with the fields: *ID_Children_attractions* - Number - Primary Key, *Name* – Text, *Address* – Text, *Map_Area* – Text, *Web_address* – Text, *Minimum_price* – Number, *Maximum_price* – Number, *Price_currency* – Text and *Description* - Text.

For *transport, tourist tours, arrivals, information sources*, we can use tables with the following fields: *ID* - Number - Primary Key, *Name* – Text, *Web_address* –Text, *Map_Area*–Text and *Description* - Text.

Here, we have described a possible structure of some tables which can be used for a travel guide, but the list of the represented data is not closed with our example.

From the presented examples of tables, we want to observe the following common features:

- In every table we use the field named *Name*- specifying the name of the referred object. This field will help us to find a certain object, using his name.

- In every table we need the *Map_area* field – to specify the area from map, where we find the corresponding object. This field will help us to find all objects from a certain area map.
- In some tables, we use fields referring to prices, like: *Minimum_price*, *Maximum_price*, *Price_symbol* or in other cases we use the field *Rate_symbol*. These fields will help us to find places depending on budget.

In the followings steps of our study, we will see that these three observations are very important and these will help us in order to explore the database.

Now, we recall that all the presented tables above are only some examples. Our implementation supposes that the website has no information. The final user (in this case, the administrator of the website for a certain city) can populate the website in the desired way. These means that our application must permit to build any table, but only respecting the following three observations:

- each table has a field named *Name*;
- every table contains the field named *Map_area*;
- in order to specify prices(rates), the user must have fields with the name containing the word *price(rate)*, different of *Rate_symbol* or *Price_currency* (these fields will be used to specify the symbol of rate, respectively the currency of price).

In order to be able to explore the database, we need additional tables with information on our database. These tables and the whole database structure will be presented in the following subsection.

2.2 Database structure and website implementation

For our website implementation, we need to use in the database some additional tables.

For each table which will be created (see *Figures 1* and 2), we will store in the *Table1* table (see *Figure 3*) the following data: an *ID* – used for the unique identification of the table, table name, a short description and the number of fields. The application will create a table named *t_table_name* (for example, for the *Hotels* table we will obtain a new table named *t_Hotels*) in which, for each field from table (in this example, for each field from the *Hotels* table) we save the following data: an *ID* – given by application, the field name, the field data type and a symbol for the data type.

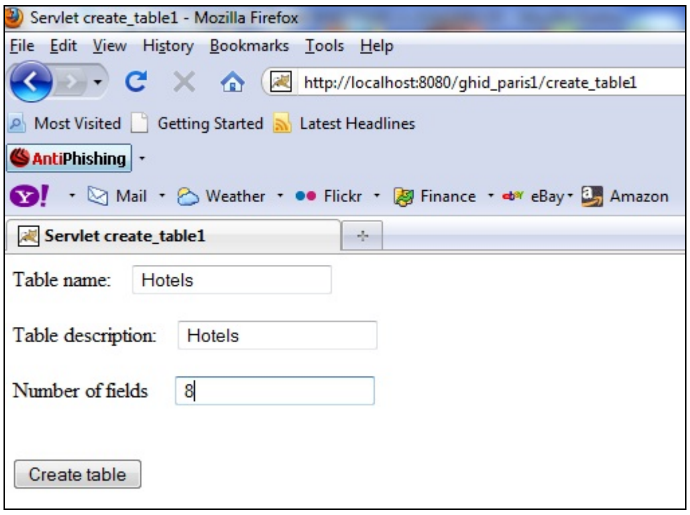

Servlet create_table1 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/ghid_paris1/create_table1
Most Visited
Getting Started
Latest Headlines
AntiPhishing
Mail
Weather
Flickr
Finance
eBay
Amazon
Servlet create_table1

Table name:	Hotels
Table description:	Hotels
Number of fields	8

Create table

Figure 1: Create a table – step 1

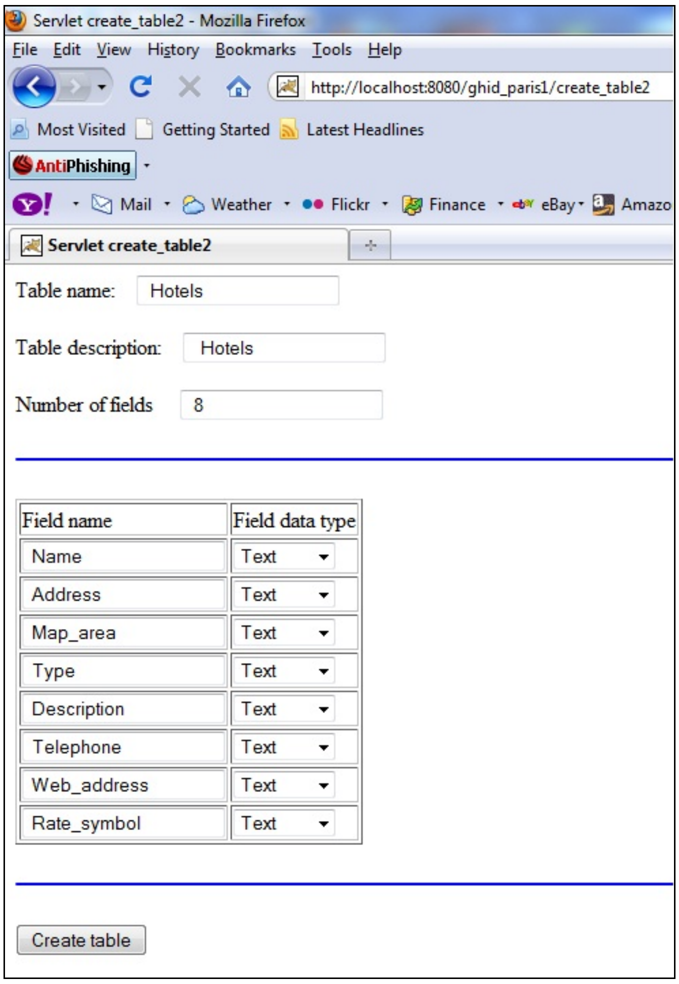

Servlet create_table2 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/ghid_paris1/create_table2
Most Visited
Getting Started
Latest Headlines
AntiPhishing
Mail
Weather
Flickr
Finance
eBay
Amazo
Servlet create_table2

Table name:	Hotels
Table description:	Hotels
Number of fields	8

Field name	Field data type
Name	Text
Address	Text
Map_area	Text
Type	Text
Description	Text
Telephone	Text
Web_address	Text
Rate_symbol	Text

Create table

Figure 2: Create a table – step 2

The tables presented in *Figures 3* and *4* are additional tables in database, which will help us in different steps of implementation.

To insert records in a table, we will select the table, like in *Figure 5*. The application will display a number of input texts (see *Figure 6*). This number is obtained from the table presented in *Figure 3*. For each field we will have presented the name and the field data type symbol.

In *Figure 7*, we present some records from the *Hotels* table.

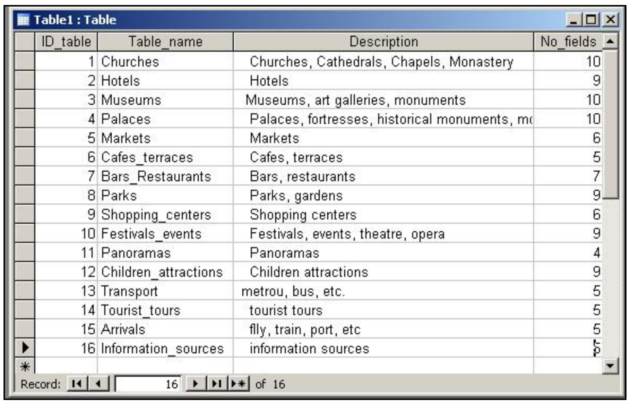
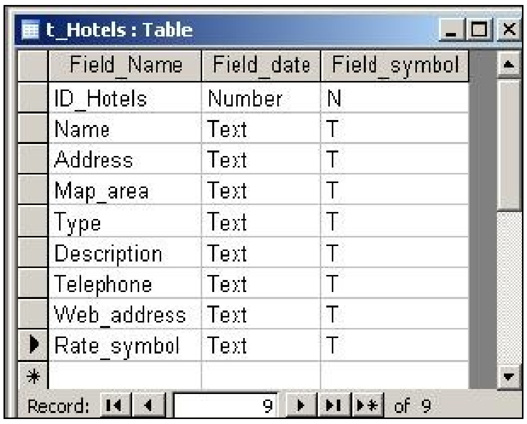


Table1 : Table

ID_table	Table_name	Description	No_fields
1	Churches	Churches, Cathedrals, Chapels, Monastery	10
2	Hotels	Hotels	9
3	Museums	Museums, art galleries, monuments	10
4	Palaces	Palaces, fortresses, historical monuments, mc	10
5	Markets	Markets	6
6	Cafes_terraces	Cafes, terraces	5
7	Bars_Restaurants	Bars, restaurants	7
8	Parks	Parks, gardens	9
9	Shopping_centers	Shopping centers	6
10	Festivals_events	Festivals, events, theatre, opera	9
11	Panoramas	Panoramas	4
12	Children_attractions	Children attractions	9
13	Transport	metrou, bus, etc.	5
14	Tourist_tours	tourist tours	5
15	Arrivals	flly, train, port, etc	5
16	Information_sources	information sources	5

Record: 16 of 16

Figure 3: Information on the working tables from database

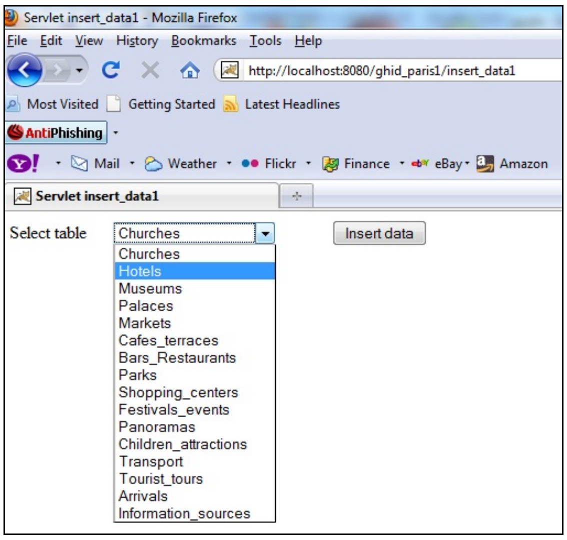


t_Hotels : Table

Field_Name	Field_date	Field_symbol
ID_Hotels	Number	N
Name	Text	T
Address	Text	T
Map_area	Text	T
Type	Text	T
Description	Text	T
Telephone	Text	T
Web_address	Text	T
Rate_symbol	Text	T

Record: 9 of 9

Figure 4: Information on the fields of a table



Servlet insert_data1 - Mozilla Firefox

http://localhost:8080/ghid_paris1/insert_data1

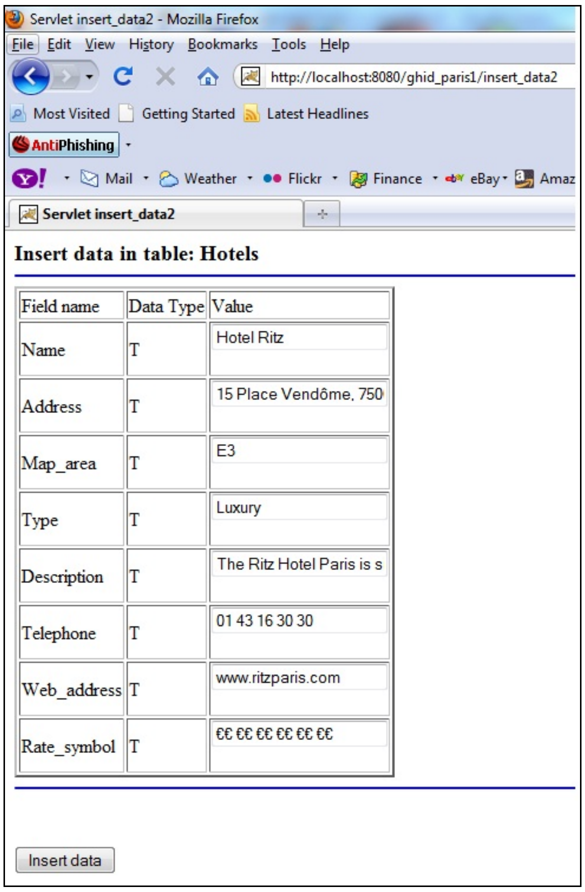
Select table: Churches

Churches
Hotels
Museums
Palaces
Markets
Cafes_terraces
Bars_Restaurants
Parks
Shopping_centers
Festivals_events
Panoramas
Children_attractions
Transport
Tourist_tours
Arrivals
Information_sources

File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines
AntiPhishing
Mail Weather Flickr Finance eBay Amazon
Servlet insert_data1
Insert data

Figure 5: Select a table for insert records

In many situations, we will store on server images or video files corresponding to the records from the working tables. For this reason, we will use other two additional tables: *Photos* and *Movies* (see *Figures 8* and *9*).



Servlet insert_data2 - Mozilla Firefox

http://localhost:8080/ghid_paris1/insert_data2

Insert data in table: Hotels

Field name	Data Type	Value
Name	T	Hotel Ritz
Address	T	15 Place Vendôme, 750
Map_area	T	E3
Type	T	Luxury
Description	T	The Ritz Hotel Paris is s
Telephone	T	01 43 16 30 30
Web_address	T	www.ritzparis.com
Rate_symbol	T	€€ €€ €€ €€ €€ €€

File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines
AntiPhishing
Mail Weather Flickr Finance eBay Amaz
Servlet insert_data2
Insert data

Figure 6: Insert records in a table

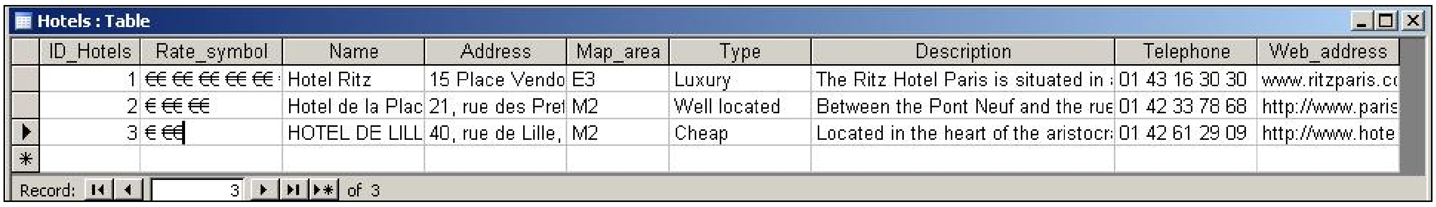
In the *Photos* table we will store data for the photos corresponding to all records, from all working tables of database. For this reason, in the *Photos* table we use the following fields:

- - *ID_table* – in order to know the table where we have the record for which we use the image file;
- - *ID_record* – for have information on the record corresponding to the image file;
- - *ID_photo* – for each record, from any table, this *ID* will start with the value 1;
- - *Photo_path* – the value of this field is given by the application and it is obtained using the first three fields presented above (see *Figure 8*). Here, we use *t* for table, *r* for record and *p* for photos. In this way, *t8r1p1.jpg* means the *jpg* file corresponding for the first image of the record with the *ID* 1 (see also *Figure 7*) from the table with the *ID* 8(see also *Figure 3*).

The *ID_table*, *ID_record* and *ID_photos* fields provide us an unique identification of records from the *Photos* table (see *Figure 8*). The same happens for video files (see *Figure 9*).

To upload an image, we will select a table as in *Figure 10* and the corresponding record, as in *Figure 11*.

-

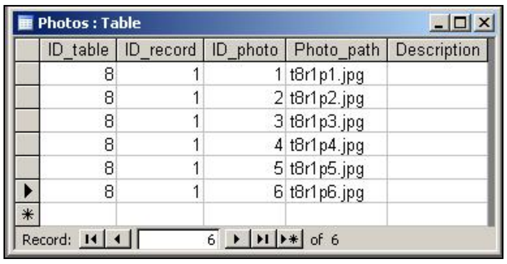


Hotels : Table

ID_Hotels	Rate_symbol	Name	Address	Map_area	Type	Description	Telephone	Web_address
1	€€ €€ €€ €€ €€	Hotel Ritz	15 Place Vendo	E3	Luxury	The Ritz Hotel Paris is situated in	01 43 16 30 30	www.ritzparis.c
2	€ €€ €€	Hotel de la Plac	21, rue des Pret	M2	Well located	Between the Pont Neuf and the rue	01 42 33 78 68	http://www.paris
3	€ €€	HOTEL DE LILL	40, rue de Lille,	M2	Cheap	Located in the heart of the aristocr	01 42 61 29 09	http://www.hote

Record: 3 of 3

Figure 7: Table and records

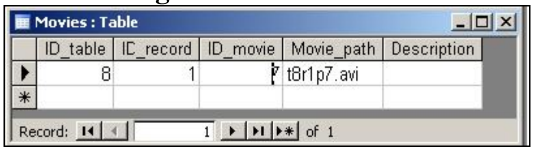


Photos : Table

ID_table	ID_record	ID_photo	Photo_path	Description
8	1	1	t8r1p1.jpg	
8	1	2	t8r1p2.jpg	
8	1	3	t8r1p3.jpg	
8	1	4	t8r1p4.jpg	
8	1	5	t8r1p5.jpg	
8	1	6	t8r1p6.jpg	

Record: 6 of 6

Figure 8: Table *Photos*

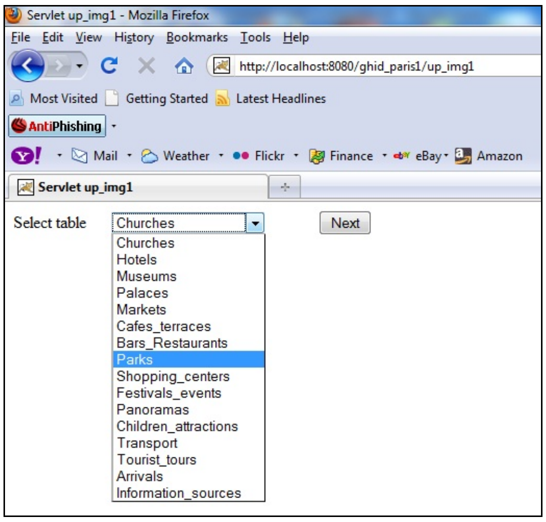


Movies : Table

ID_table	ID_record	ID_movie	Movie_path	Description
8	1	7	t8r1p7.avi	

Record: 1 of 1

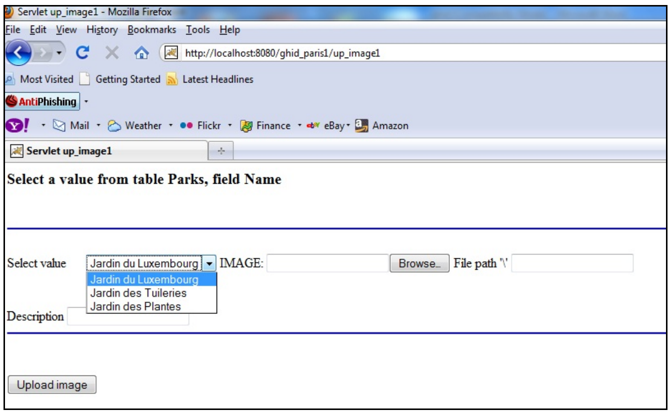
Figure 9: Table *Movies*


Servlet up_img1 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/ghid_paris1/up_img1
Servlet up_img1
Next

Select table: Churches

Churches
Hotels
Museums
Palaces
Markets
Cafes_terraces
Bars_Restaurants
Parks
Shopping_centers
Festivals_events
Panoramas
Children_attractions
Transport
Tourist_tours
Arrivals
Information_sources

Figure 10: Upload an image file – step 1


Servlet up_image1 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/ghid_paris1/up_image1
Servlet up_image1
Browse...
Upload image

Select a value from table Parks, field Name

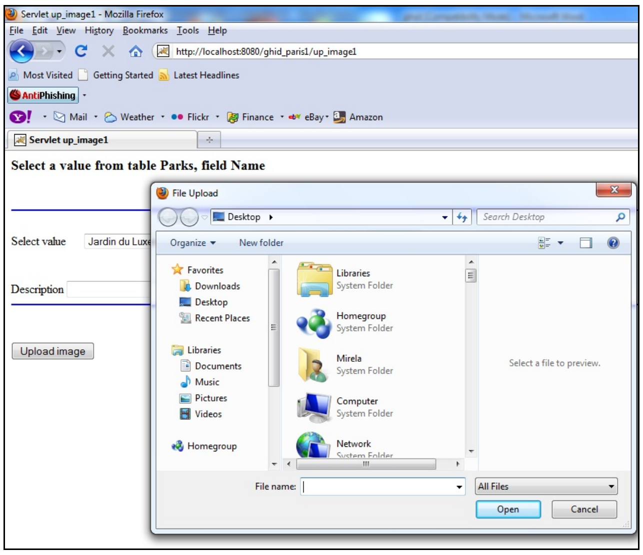
Select value: Jardin du Luxembourg IMAGE: File path '\'

Jardin du Luxembourg
Jardin des Tuileries
Jardin des Plantes

Description

Figure 11: Upload an image file – step 2

Using an *HTML* input element, where *type= "file"*, we will select an image file, as in *Figure 12*. When we will action on the *Upload image* submit control (from *Figures 11* and *12*), the application will provide the name of the image file (see *Figure 8*, the *Photo_path* field) and it will store this file in the web folder of the website. The same happens if we will upload a video file.



Select a value from table Parks, field Name

Select value: Jardin du Luxe

Description

Figure 12: Upload an image file – step 3

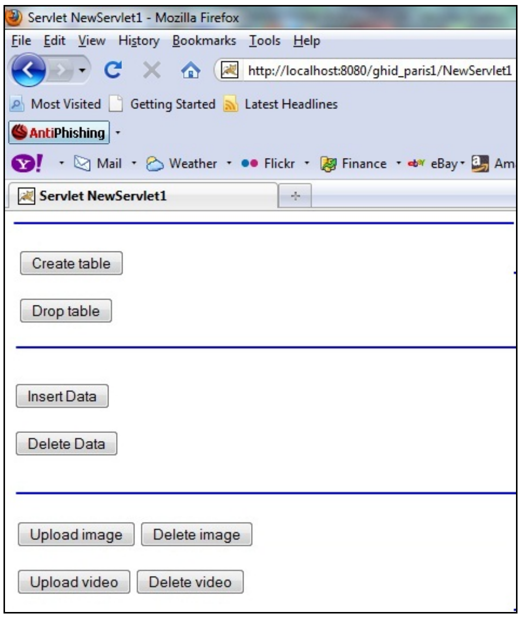
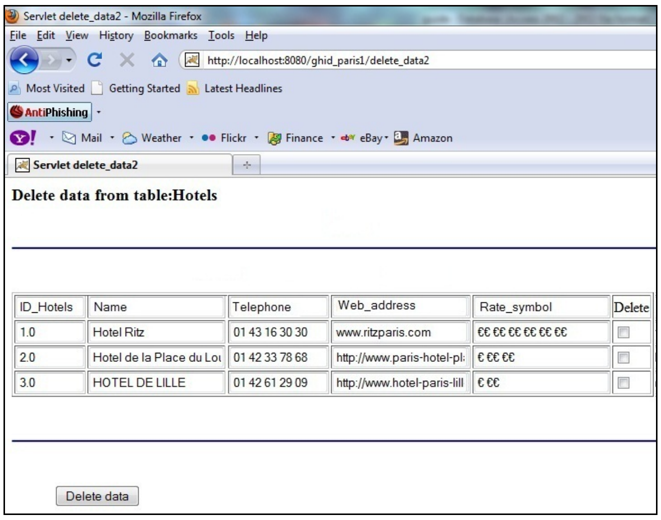

Servlet NewServlet1 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/ghid_paris1/NewServlet1
Servlet NewServlet1
Create table
Drop table
Insert Data
Delete Data
Upload image
Delete image
Upload video
Delete video

Figure 13: Database administration

Our application allows us to create tables, insert records and upload image or video files. But also, we can delete tables, records, image or video files.

For example, to delete records, we will select a table like in *Figure 10*. If we select the *Hotels* table, we will see all records, as in *Figure 14*. In *Figure 14* we only have a subset of fields from the *Hotels* table, but in web pages, we represent all these fields, that occurs a space with a large width. In *Figure 14* we select the records which will be deleted. In the moment in which we delete a record from the *Hotels* table, from *Photos* and *Videos* tables we will delete the corresponding data and also, we will delete the corresponding image or video files from the server.

Similar, we have web pages used to delete only photos or movies.


Servlet delete_data2 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/ghid_paris1/delete_data2
Most Visited Getting Started Latest Headlines
AntiPhishing
Mail Weather Flickr Finance eBay Amazon
Servlet delete_data2

Delete data from table:Hotels

ID_Hotels	Name	Telephone	Web_address	Rate_symbol	Delete
1.0	Hotel Ritz	01 43 16 30 30	www.ritzparis.com	€€ €€ €€ €€ €€ €€	☐
2.0	Hotel de la Place du Lou	01 42 33 78 68	http://www.paris-hotel-pl	€ €€ €€	☐
3.0	HOTEL DE LILLE	01 42 61 29 09	http://www.hotel-paris-lill	€ €€	☐

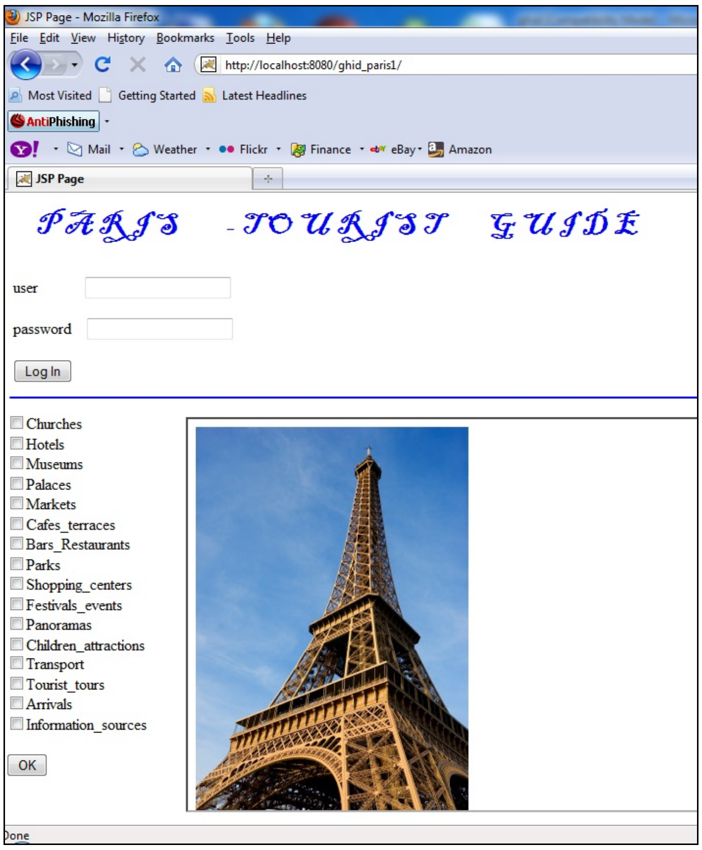
Delete data

Figure 14: Delete records

3 Website presentation

In *Figure 15* we will have a *checkbox* for each working table from the database. These tables are specified in *Figure 3*. This means that if we will add or delete tables from the database, then the number of *checkboxes* will be automatically changed. In *Figure 15*, we will select the *checkboxes* for which we want to view data. For example, if we select the *Churches, Museums* and *Parks* checkboxes, then in *Figure 16* we will view all records from these tables (here, only the *ID* and *Name* fields will be displayed). At this point, with a click on the corresponding *OK* button, we can view all information from database according to our selection. For example, if in *Figure 16*, we select to view data on *Jardin du Luxembourg* from the *Parks* table, then in *Figure 17* we will view the information from the *Parks* table and the image (video) files – according with the data from the *Photos* (*Movies*) table.

Now, using the *Map_area* field from all tables, we can select a certain value and in this way we can view data as in *Figure 16*, but only for the cases corresponding to the selected map area.


JSP Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/ghid_paris1/
Most Visited Getting Started Latest Headlines
AntiPhishing
Mail Weather Flickr Finance eBay Amazon
JSP Page
PARIS - TOURIST GUIDE
Done

user

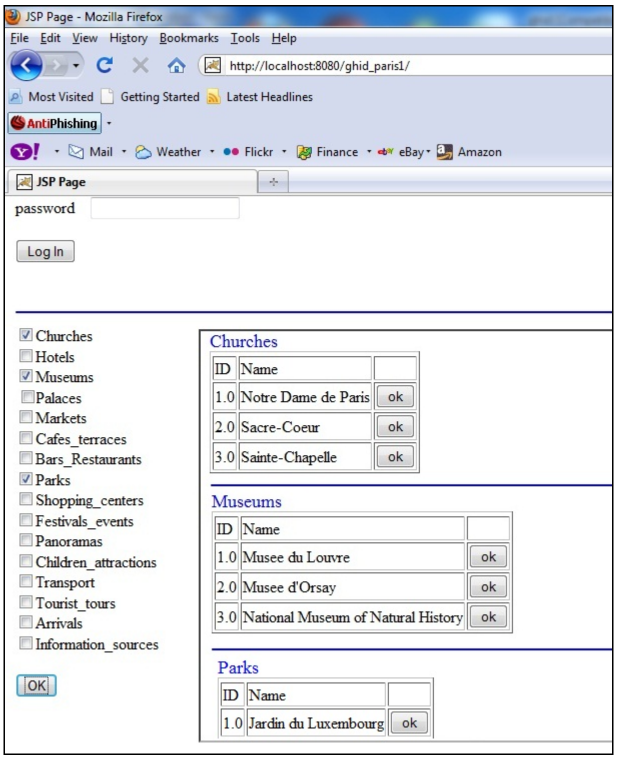
password

Log In

☐ Churches
☐ Hotels
☐ Museums
☐ Palaces
☐ Markets
☐ Cafes_terraces
☐ Bars_Restaurants
☐ Parks
☐ Shopping_centers
☐ Festivals_events
☐ Panoramas
☐ Children_attractions
☐ Transport
☐ Tourist_tours
☐ Arrivals
☐ Information_sources

OK

Figure 15: The first page from the site


JSP Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/ghid_paris1/
Most Visited Getting Started Latest Headlines
AntiPhishing
Mail Weather Flickr Finance eBay Amazon
JSP Page

password

Log In

☑ Churches
☐ Hotels
☑ Museums
☐ Palaces
☐ Markets
☐ Cafes_terraces
☐ Bars_Restaurants
☑ Parks
☐ Shopping_centers
☐ Festivals_events
☐ Panoramas
☐ Children_attractions
☐ Transport
☐ Tourist_tours
☐ Arrivals
☐ Information_sources

OK

Churches

ID	Name	
1.0	Notre Dame de Paris	ok
2.0	Sacre-Coeur	ok
3.0	Sainte-Chapelle	ok

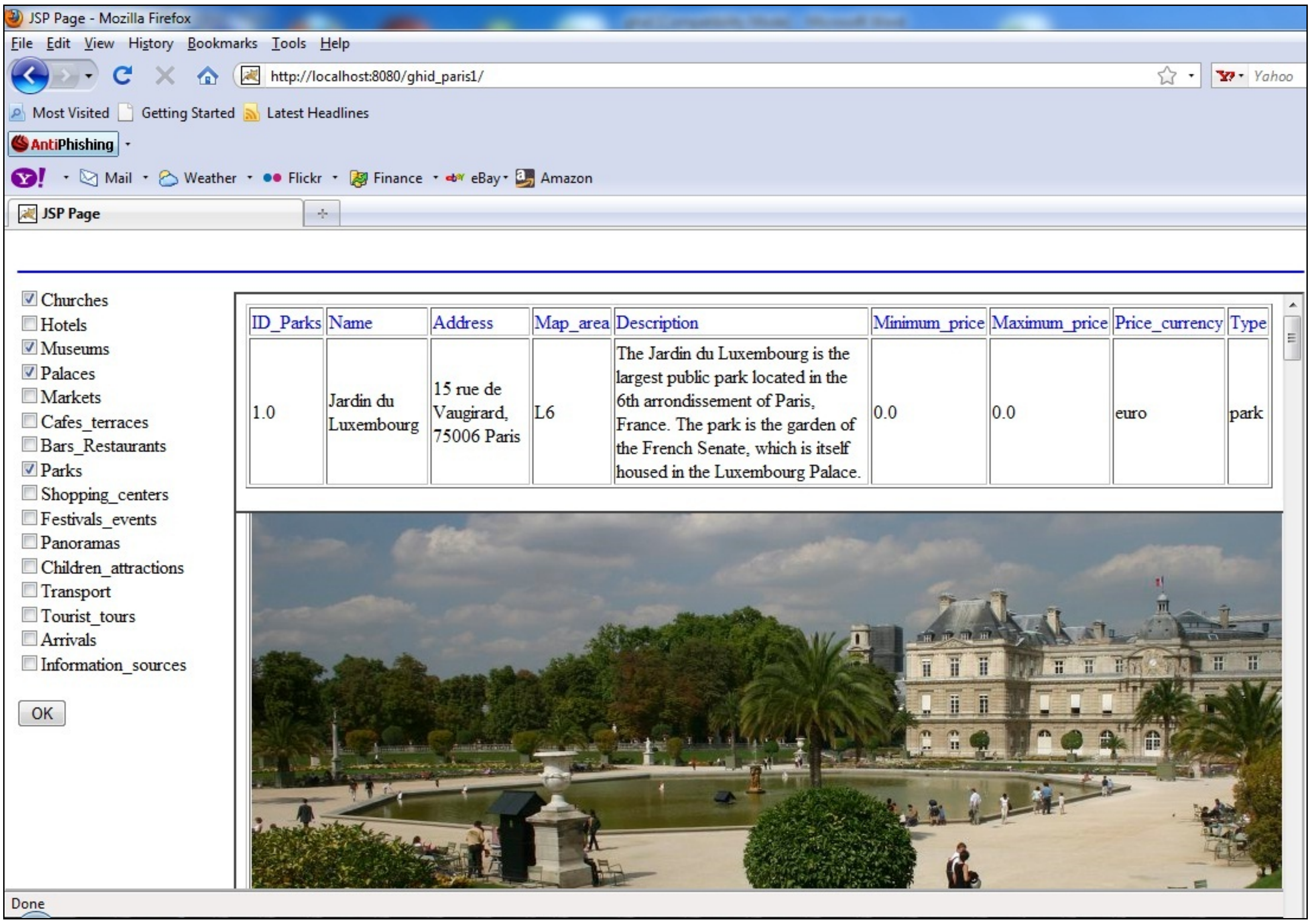
Museums

ID	Name	
1.0	Musee du Louvre	ok
2.0	Musee d'Orsay	ok
3.0	National Museum of Natural History	ok

Parks

ID	Name	
1.0	Jardin du Luxembourg	ok

Figure 16: Selection of the interest data



ID_Parks	Name	Address	Map_area	Description	Minimum_price	Maximum_price	Price_currency	Type
1.0	Jardin du Luxembourg	15 rue de Vaugirard, 75006 Paris	L6	The Jardin du Luxembourg is the largest public park located in the 6th arrondissement of Paris, France. The park is the garden of the French Senate, which is itself housed in the Luxembourg Palace.	0.0	0.0	euro	park

Figure 17: Data for a touristic objective

Also, using the fields which refer to rate or price, we can view in *Figure 16*, only the records corresponding at such criteria.

Moreover, using criteria for map area, rates and prices, we can view in *Figure 16*, only the records which respect these conditions.

Using the criteria from the three presented above, we can help the tourist to know what he will visit in a certain area map and we will inform him on a necessary amount of money.

4 Programming code presentation

In this section we present a part of our programming code.

In order to obtain the page presented in *Figure 1*, we use in the *create_table1* servlet the following function:

```
protected void processRequest(HttpServletRequest
request, HttpServletResponse response)
 throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();

try {
// html header
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet create_table1</title>");
out.println("</head>");

/* The JavaScript function used to the Create table
submit button. This way of implementation is
recommended especially when we have two or more
submit buttons on the form */
out.println(" <script language=\"javascript\">");
out.println("function create_table2(){
document.form1.action=\"http://localhost:8080/ghid_pa
ris1/create_table2\"; } ");
out.println("</script>");

/*the html body and the form*/
out.println("<body>");
out.println("<form name=\"form1\" action=\"\"
method=post> ");

/* Text boxes used to introduce the table name, its
description and the number of fields */
 out.println(" Table name:&nbsp;&nbsp;&nbsp;&nbsp;
<input name=\"text1\" type=\"text\" value=\" \"><BR>
<BR>");
out.println(" Table
description:&nbsp;&nbsp;&nbsp;&nbsp;<input
name=\"text2\" type=\"text\" value=\" \"><BR>
<BR>");
 out.println(" Number of fields &nbsp;&nbsp;
&nbsp;&nbsp;<input name=\"text3\" type=\" text\"
value=\" \"><BR><BR>");

// The Create table submit button
out.println(" <br><input type=\"submit\"
value=\"Create table\" onclick=\"create_table2()\">");
out.println("</form>"); out.println("</body>");
out.println("</html>");
} finally
{ out.close();}}
```

In order to introduce the name and data type of the new table fields, we use the page presented in *Figure 2*, which is created by the following programming code:

```
protected void processRequest(HttpServletRequest
request, HttpServletResponse response)
throws ServletException, IOException {
 response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet create_table2</title>");
out.println("</head>");

out.println(" <script language=\"javascript\">");
out.println("function create_table2(){
document.form2.action=\"http://localhost:8080/ghid_pa
ris1/create_table3\"; } ");
out.println("</script>");
out.println("<body>");

/*Reading the values from the text boxes presented in
Figure 1*/
String table_name=request.getParameter("text1");
String table_description=request.getParameter("text2");
String no_fields=request.getParameter("text3");
double nf=0.0;
Double dt=0.0;

/*We introduce the name and the data type for each
field of the new table*/
 if(no_fields!=null) {dt= Double.valueOf(no_fields);}
 nf=dt.doubleValue();
 int nof=(int)nf;
 int i=1;
out.println("<form name=\"form2\" action=\"\"
method=post> ");
out.println(" Table
name:&nbsp;&nbsp;&nbsp;&nbsp;<input
name=\"text1\" type=\"text\" value=\" "+table_name+"
\"><BR><BR>");
out.println(" Table
description:&nbsp;&nbsp;&nbsp;&nbsp;<input
name=\"text2\" type=\"text\" value=\"
"+table_description+" \"><BR><BR>");
out.println(" Number of fields
&nbsp;&nbsp;&nbsp;&nbsp;<input name=\"text3\"
type=\"text\" value=\" "+no_fields+"  \"><BR><BR>");

//The header of the table presented in Figure 2
out.println("<hr border=3
color=\"blue\"><br>");
out.println("<table border=1>");
out.println("<tr>");
out.println("<td> Field name </td>");
out.println("<td> Field data type </td>");
out.println("</tr>");

// The rows of the table presented in Figure 2
for(i=1;i<=nof;i++){
out.println("<tr>");
out.println("<td>");
out.println("<input name=\"t"+String.valueOf(i)+"\"
type=\"text\" value=\" \">");
out.println("</td>");
out.println("<td>");
out.println(" <select
name=\"combo"+String.valueOf(i)+"\">");
out.println("  <option>Text</option>    ");
out.println("  <option>Number</option>    ");
out.println("  <option>Date</option>    ");
out.println(" </select>");
out.println("</td>");
out.println("</tr>");
}
out.println("</table>");

out.println("<br><hr border=3 color=\"blue\">");
out.println(" <br><input type=\"submit\"
value=\"Create table\" onclick=\"create_table2()\">");
out.println("</form>");out.println("</body>");
out.println("</html>");
} finally { out.close(); }}
```

In order to create the new table specified in the page presented in *Figure 2* and for insert the corresponding additional information in tables presented in *Figures 3* and *4*, we use the following function:

```
protected void processRequest(HttpServletRequest
request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet create_table3</title>");
out.println("</head>");
out.println("<body>");

/*Reading the values from the text boxes presented in
Figure 2*/
String table_name=request.getParameter("text1");
String table_description=request.getParameter("text2");
String no_fields=request.getParameter("text3");
double nf=0.0;
Double dt=0.0;

 if(no_fields!=null)  {dt= Double.valueOf(no_fields);}
 nf=dt.doubleValue();
int nof=(int)nf+1;

/*We detect the ID of the new table. This ID is saved in
the table presented in Figure 3*/
String sql_statement="select max(ID_table) from
table1";
int ID_max=0;

try {
String user = "";
String password = "";
String dbURL = "jdbc:odbc:Driver={Microsoft Access
Driver (*.mdb)};DBQ=C:\\Program Files\\Apache
Software Foundation\\Tomcat
6.0\\webapps\\guide\\guide.mdb;DriverID=22;READO
NLY=false";
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection c = DriverManager.getConnection(dbURL,
user, password);
Statement s = c.createStatement();
ResultSet r = s.executeQuery(sql_statement);
while (r.next()) {
    ID_max=r.getInt(1);
   }
  r.close();
  s.close();
 }
 catch (ClassNotFoundException a) {
    // Could not find the driver
 }
 catch (SQLException a) {
      // Could not connect to the database
 }
ID_max+=1;

/* Inserting information (on the new table)  in the table
named  table1 and presented in Figure 3*/
sql_statement="Insert into table1 values("+
String.valueOf(ID_max)+", '"+table_name.trim()+"', '"+
table_description+"', "+String.valueOf(nof)+")";

/* The database connection and the SQL statement
execution */
try {
String user = "";
String password = "";
 String dbURL = "jdbc:odbc:Driver={Microsoft Access
Driver (*.mdb)};DBQ=C:\\Program Files\\Apache
Software Foundation\\Tomcat
6.0\\webapps\\guide\\guide.mdb;DriverID=22;READO
NLY=false";
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) {
}

// Creating the new table
sql_statement="Create table
"+table_name+"(ID_"+table_name.trim()+" Number
Primary Key, ";
int i=1;
String t_param="t";
String combo_param="combo";

String f_name="", f_type="",f_pk="";
for(i=1;i<=nof-1;i++){
t_param="t"+String.valueOf(i);
f_name=request.getParameter(t_param);
sql_statement=sql_statement+f_name+" ";
combo_param="combo"+String.valueOf(i);
f_type=request.getParameter(combo_param);

sql_statement=sql_statement+f_type;

if(i==nof-1)
{sql_statement=sql_statement+" )";}
else
  {sql_statement=sql_statement+",";}
}

/* The database connection and the SQL statement
execution - see the code presented above */

/* Creating the associated  t_ table (see the table
presented in Figure 4)*/
sql_statement="Create table
t_"+table_name.trim()+"(id_f Number Primary Key,
Field_Name Text, Field_date Text, Field_symbol
Text)";

/* The database connection and the SQL statement
execution - see the code presented above */

/* Inserting records in  the associated t_ table (see the
table presented in Figure 4)*/
sql_statement="Insert into t_"+table_name.trim()+"
values(1, 'ID_"+table_name.trim()+"', 'Number', 'N')";

/* The database connection and the SQL statement
execution. For database connection, see the code
presented above */
sql_statement="";
out.println("<BR>");
for(i=1;i<=nof-1;i++){
t_param="t"+String.valueOf(i);
f_name=request.getParameter(t_param).trim();
combo_param="combo"+String.valueOf(i);
f_type=request.getParameter(combo_param).trim();
sql_statement="Insert into t_"+table_name.trim()+"
values("+ String.valueOf(i+1)+", '"+f_name+"',
'"+f_type+"', ";

if(f_type.equals("Number"))
{sql_statement=sql_statement+" 'N')";}
if(f_type.equals("Text"))
{sql_statement=sql_statement+" 'T')";}
if(f_type.equals("Date"))
{sql_statement=sql_statement+" 'mm/dd/yyyy/')";}
out.println("<BR>");

/* The database connection and the SQL statement
execution - see the code presented above */

out.println("<h1>Tabela a fost creata</h1>");
out.println("</body>");
out.println("</html>");
} finally {
 out.close();}
}
```

In this section, we have presented the code used to create tables in our database. This operation can be used in any moment, even when we use the website. In this way, the number of tables increases (or, if necessary, it decreases), depending by the information that must be displayed in website at the certain moment.

The programming code presented above can provide a good idea on the kind of implementation. For this reason, we here stop function presentations. We use similarly programming code for the operations presented in *Figure 13*. For the page presented in *Figure 15* we use a *JSP* file and for the pages presented in *Figures 16* and *17* we use servlets.

5 Conclusion

Using the model of the *Top 10* series of travel guides, we can create websites which will help the tourist with a good documentation on the places which he will visit.

Before of his travel, the tourist can build an itinerary and estimate the necessary amount of money in order to be able to visit all proposed objectives.

In this paper we try to respect the idea of the *Top 10* series. For this reason we suppose that all modifications on website will be made by websites administrators.

However, this idea can be extended (if necessary) in the case in which, using users and passwords, different person can have access to populate database with data.

In this last case, we will find more additional records, but this idea will change the model proposed by Top 10.

For website implementation we have used Java servlets and *JSP* files. Our data have been stored in Access database, but for implementation, we can use any type of relational database (Oracle, SQL Server, MySQL, etc.) where we use the same programming code.

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] Mastorakis, N.E. - *Genetic algorithms with Nelder-Mead optimization in the variational methods of boundary value problems* March 2009, WSEAS Transactions on Mathematics, Volume 8, Issue 3

[4] Muntean M.C., Nistor R., Nistor C. - *Competitiveness of Developing Regions in Romania* - WSEAS TRANSACTIONS on BUSINESS and ECONOMICS Volume 7, 2010, ISSN: 1109-9526

[5] Scoarţă I., Bâră A., Constantinescu R., Zota R., Năstase F. - *Improvingorganizational efficiency and effectiveness in a Romanian Higher Education Institution*, WSEAS Transactions on Computers, Volume 8, Issue 10, pp. 1641 - 1650, october 2009, ISSN:1109-2750

[6] Top 10 Athens, Dorling Kindersley Limited, London, 2007

[7] Top 10 London, Dorling Kindersley Limited, London, 2007

[8] Top 10 Paris, Dorling Kindersley Limited, London, 2007

[9] Top 10 Rome, Dorling Kindersley Limited, London, 2007

[10]http://www.dorlingkindersley-uk.co.uk/static/html/uk/series/top10_index.html