TIN2009-11005
DAMASK

Data-Mining Algorithms with Semantic Knowledge

PROYECTO DE INVESTIGACIÓN
PROGRAMA NACIONAL DE INVESTIGACIÓN FUNDAMENTAL,
PLAN NACIONAL DE I+D+i 2008-2011
ÁREA TEMÁTICA DE GESTIÓN: Tecnologías informáticas

Deliverable D6
User-oriented recommender system

Authored by
Ferran Mata
Antonio Moreno
Aïda Valls
Carlos Vicient

Document information

project name:	DAMASK	
Project reference:	TIN2009-11005	
type of document:	Deliverable	
file name:	D6.pdf	
version:	1.0	
authored by:	F. Mata, A. Moreno, A. Valls, C.Vicient	26/07/2012
co-authored by		
released by:	A. Valls	30/07/2012
approved by:	Co-ordinator	Antonio Moreno

Document history

version	date	reason of modification
1.0	27.July.2012	User-oriented recommender system.

Table of Contents

1 Introduction 4

2 The architecture 6

2.1 Tools used 6

2.1.1 JAVA + JavaServer Faces 6

2.1.2 MySQL + Hibernate 7

2.1.3 Spring security 8

2.1.4 jQuery 9

2.2 Structure of the system 9

3 A Web recommender system for touristic destinations 11

3.1 Parts of the Web page 12

3.2 Sections of the Web 13

3.2.1 City list 13

3.2.2 Clusters 13

3.2.3 Profile 15

3.2.3.1 Structure of the user profile 16

3.2.3.2 Definition of the profile in the Web page 17

3.2.4 Recommendation 20

3.2.4.1 The recommendation algorithm 21

3.2.4.2 Recommendation of touristic city destinations in the DAMASK Web application 21

1 Introduction

This document corresponds to Task 3 of the DAMASK project. In task T3 the goal is to evaluate the deployment of the methods designed in the previous tasks in a particular case study: a personalized recommendation system of touristic destinations. A Web application has been designed to offer this kind of recommendations to any user. The tool is focused on searching touristic destinations in the different types of touristic resources available in Internet using the tools developed in task T1. The clustering method defined in T2.4 has been integrated in this prototype to obtain a classification of touristic destinations based on the domain knowledge and the user preferences, in order to be able to recommend the set of places that match better with the user's interests. In particular, the recommender system is built using the the data matrix explained in the DAMASK internal report 3.2, and the clusters built using the clustering technique explained in Deliverable D5.

This deliverable is the result of the task T3.1 and T3.2 as shown in the schedule of the tasks in the DAMASK project is given in Figure 1.

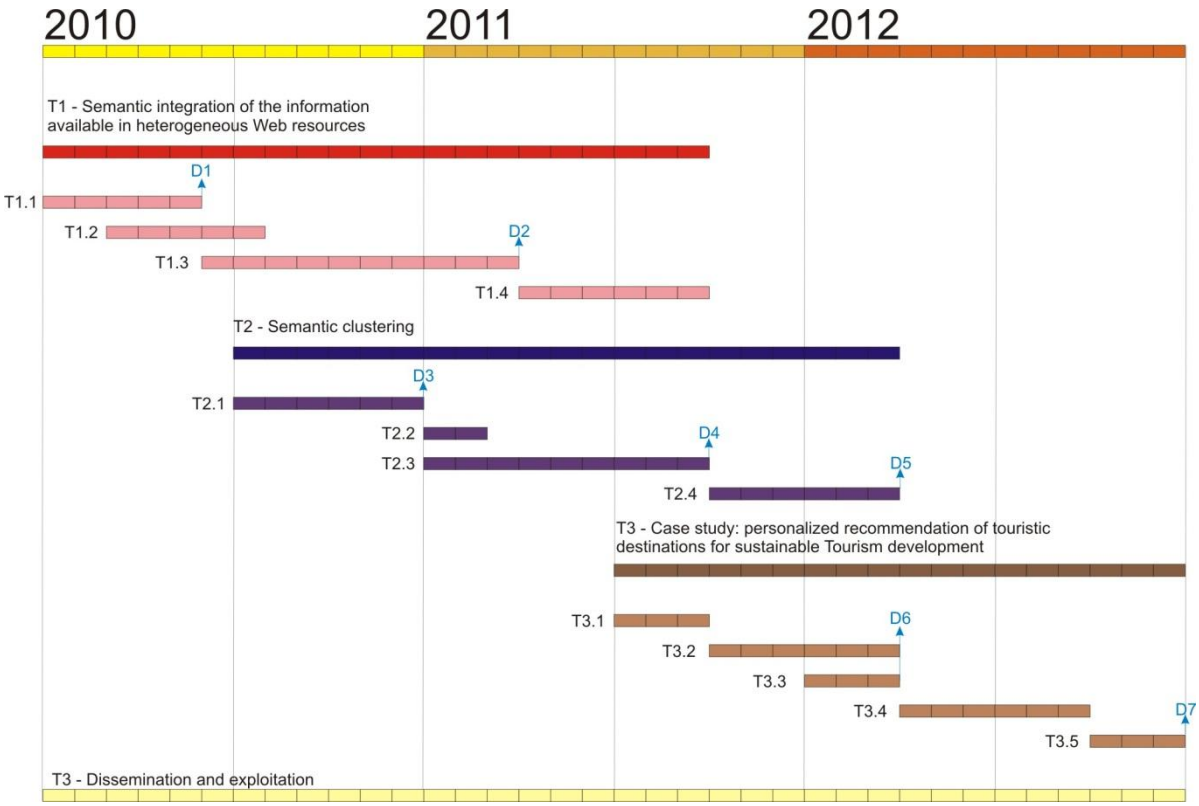

Figure 1: Tasks of DAMASK

The DAMASK recommender system is intended to be used for people who want to search the most appropriate city for making some Tourism. The system provides a Web page

that is connected with a server application to permit to the user to receive a list of touristic destinations that fits with his/her interests and preferences. The systems allows to have multiple users registered, each one linked to a personal profile that stores the preferred values on a set of criteria.

Briefly, each user defines a set of requirements for the desired type of city using only semantic attributes. The recommender system selects the cluster that is able to fulfill to a highest degree the preferences given by the user. Then, from the cities that belong to this cluster, a selection can be made using some filters defined upon the numerical attributes (population and elevation) and the categorical attributes (continent and climate).

This document is structured in two main sections. The first explains in detail the design of the Web application, the tools used for the implementation and its architecture. The second gives details about the functionalities provided by the application, the interface, as well as the algorithms behind the recommendation process.

The DAMASK System is available at: http://10.30.102.154:8080/Damask/

2 The architecture

The implementation tools have been selected taking into account that Java is the programming language adopted at the beginning of the project. Java facilitates an easy portability to different operating systems and platforms, even to different types of devices.

To deploy the recommender system at the Web, a computer has been configured as Web server. This server has been installed using the tool WAMP (www.wampserver.com), which includes: Windows, Apache, MySQL and PHP. Since the project is just a prototype, without the intention to build a more robust and ad-hoc server to host a system for real exploitation, the WAMP application sufficiently fulfills the needs of the prototype. Apache is used as an HTTP server, MySQL is the database and PHP is no used (since the system is implemented in Java). WAMP also provides a phpMyAdmin to manage the MySQL database. It is worth to note that the Apache server does not work with Java applications per se. For that purpose, Apache Tomcat is used as the main HTTP server to support the development with Java Web applications.

2.1 Tools used

2.1.1 JAVA + JavaServer Faces

The base of the system is built using Java 7 for an important reason. The previous applications of the DAMASK project were build using JAVA. Hence, the portability of some algorithms to the Web system is much easier if the same language is used. But Java *per se* does not provide tools to program a Web system. For that purpose, the JavaServer Faces technology is used.

JavaServer Faces (JSF) is a Java-based technology or Web application framework designed to simplify and provide tools for the development of Web-based user interfaces.JSF follows the model-view-controller (MVC) model, which is a standard in Web development. Basically, MVC separates the representation of information from the user's interaction with it. The *model* consists of application data and the *controller* is responsible of manage the input to process and command it to the *model* or the *view* (see Figure 2).

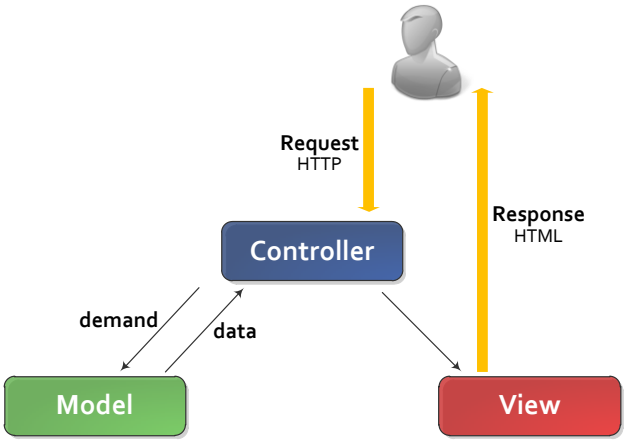

Figure 2: MVC architecture

JavaServer Faces uses JavaServer Pages (JSP) as display technology in its first specification. JSP is a technology used to create dynamically generated Web pages based on HTML, XML, or other document types. But the 2nd specification of the JSF uses Facelets, its own view handler technology. Facelets requires valid input XML documents to work. JSF 2 can also work with JSP, but its use is discouraged.

Facelets provides templating features. This is ideal to build a Web site that reuses at each page the same elements such as the header, the footer, menus… Hence, a master template can be built indicating which source to use as permanent elements of the Web.

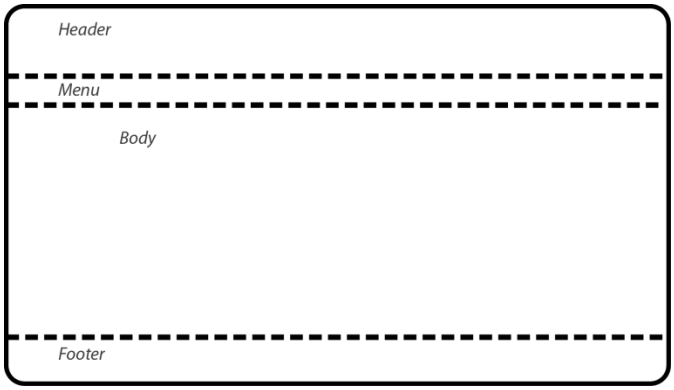

Figure 3: Representation of a master template

When a new page is developed, only the body needs to be written, indicating in the file that the document uses the template *X* and its content replaces the placeholder *Y* ("body", for instance).

2.1.2 MySQL + Hibernate

The WAMP server includes a MySQL database. MySQL is a relational database management system (RDBMS) that provides multi-user access to a number of databases. As its name suggests, MySQL uses the Structured Query Language (SQL).

MySQL databases can be managed and administered using phpMyAdmin that WAMP also includes. Some of the phpMyAdmin feature includes the creation, modification and

deletion of databases, tables, fields and rows, execute SQL statements, or manage users and its permissions.

It is not trivial to use a MySQL database with Java. There exists some implementation to access to it in the Java core, but it is much better to use a framework like Hibernate to access the database.

Hibernate works as an Object-Relational Mapping (ORM). This kind of frameworks provides mapping for object-oriented domain model to a relational database. Hibernate solves *object-relational impedance mismatch*(a set of conceptual and technical difficulties) problems by replacing direct persistence-related database accesses with high-level object handling functions. In other words, what Hibernate does is mapping from Java classes to database tables and vice versa.

Hibernate provides data query and retrieval facilities, and implements its own query language (*Hibernate Query Language*, HQL) in order to provide an abstraction layer for different databases, which means that working with hibernate makes the type of database transparent for the developer. Only in the configuration file must be specified the type of database that Hibernate is going to communicate with.

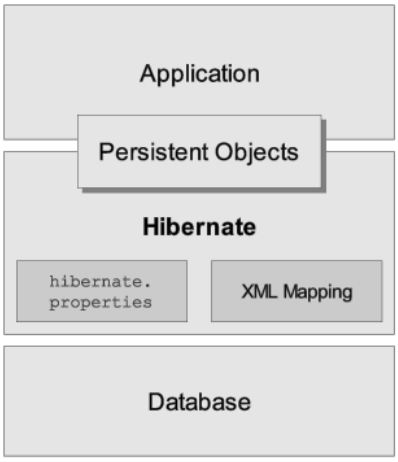

Figure 4: Schema of the Hibernate interaction

2.1.3 Spring security

The system implements a module of the Spring framework, which is a popular application development framework for enterprise Java. Spring framework has several modules that provide multiple functionalities, but the module used in the DAMASK project is the one that provides *authentication* and *authorization*.

The module is useful to manage the users. Actually, is not just a module but a sub-project. **Spring security** provides authentication, authorization and other security features for enterprise applications. In the DAMASK project is used to maintain the user sessions and to protect their information, mainly its password, which is encrypted using an *md5* hash.

In the spring security configuration, certain *url* patterns are defined to intercept and check for user authorization and authentication. For instance, if a user tries to access to *"/faces/profile***" spring security will intercept the request and check if the user is valid before send the request to the other controllers of the request. If the user is not valid, spring security would redirect the user to the login page or to an error page.

2.1.4 jQuery

jQuery is a JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid Web development. In the DAMASK project it is used for its improvement on the Javascript management and for the animations and effects, which adds a nice feel to the site.

2.2 Structure of the system

As explained in the above sections, the system uses a model-view-controller user interface. This strictly separates the parts of the system represented in the following schema:

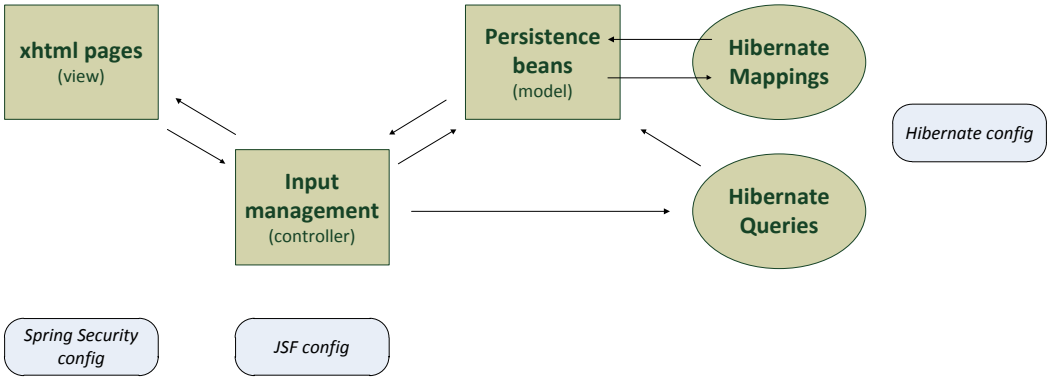

Figure 5: Schema of the system

- **xhtml pages:** JSF2 specifies that Facelets use the Extensible HyperText Markup Language (XHTML) format. Hence, the pages are a combination of xhmtl, JSF2 tags, JavaServer Pages Standard Tag Library (JSTL) and Javascript. This represents the view of the system along with the Java Beans that directly interact with the pages. These Javas beans are responsible of provide and receive information to the pages.

- **Input management:** Once an event is triggered in the view, the view manages the event and asks the controller to do something with the data gathered in the view. This usually implies two kinds of procedures (or both at once):

    - *Save data:* The user inputs data in the view that is passed to the controller to be saved in the database as a model.

    - *Retrieve data:* The view asks the controller for same data. The controller access the database to get the requested data.

In both cases, the controller can use the data to get the desired result. For instance, the user may ask through the view about the nearest cluster to the user profile. Then the controller will get from the view the attributes percentages, checked concepts and the filter values to build the user profile. Afterwards the controller will retrieve from the database all the clusters data and check which cluster is the one that its centroid is the nearest to the user profile. Then the controller filters the selected cluster and sends the result to the view, so that the user can see the resulting cities.

- **Persistence beans:** The Hibernate framework establishes relations between rows in tables in a database and Java beans through the hibernate mappings. So, the Java beans are, in some way, the representation of the database. When the controller wants to save data into the database, it has to create persistence beans first with that data. The Hibernate queries are used to retrieve these beans (usually lists of them) with the help of HQL for filtering purposes.

2.3 Installation Steps

1. Install java 7. The computer must have java 7 runtime environment installed.
2. Install wampserver. Wampserver is a windows Web development environment.
3. Install tomcat. Tomcat is is an open source software implementation of the Java Servlet and JavaServer Pages technologies.
4. Copy the file damask.war into the folder: "C:\Archivos de programa\Apache Software Foundation\Tomcat 7.0\webapps"
5. Tomcat deploys the .war file when it is copied into the folder.
6. Open PhpMyAdmin from the wamp icon.
7. Go to the "Databases" tab and create a database named "damask"
8. Select the "damask" database.
9. Go to the SQL tab and copy the content of damask.sql file. Click on execute to load the database content.
10. Go to http://localhost:8080/Damask to test if the system works properly.
11. Remember to open the port 8080 on the windows firewall to access from outside of localhost.

At the moment the DAMASK System is available at: http://10.30.102.154:8080/Damask/

3 A Web recommender system for touristic destinations

The DAMASK Web recommender system has been designed and implemented to be easy to use. Actually, it is prepared to be used by any kind of user, not requiring any knowledge about using other recommender systems or any specialized computer program. Hence, the Web follows the typical pattern of Web applications at Internet: a header, a menu, the body, and footer, as displayed in Figure 5.

The Web application is structured into 4 sections: a city list, a clusters list (each with its own list of cities), a profile where the user creates his personal profile and finally a page where the recommendation to the user is presented. The Web also provides a user login system in order for each user to maintain its own profile.

Figure 6: View of the main page

3.1 Parts of the Web page

The Web page presented to the user is divided into four main parts:

- The header only shows the logo of the project and an image representing the skyline of a city with a plane. The logo is used as a return-to-index link. The skyline image has been designed for decoration purposes of this prototype, devoted to the recommendation of touristic city destinations.

Figure 7: Header of the Web page in the DAMASK recommender system

- The menu part is represented as a simple bar below the header. The links to the different sections of the site (*city list*, *clusters*, *profile* and *recommendation*) are simple text links on the left, with no images that change their colors when the mouse moves over. On the right one can find the user management related links.

  When not logged in, a user will see the links for either **log in** or, if it has no registered user yet, **register** a new one. When the user is logged in, the links change to a text showing the **user name** and a link to **logout**. Also, when not logged in, the available links to the sections are just the ones to access the *city list* and the *clusters list*. This is done this way because it an unregistered user cannot have a saved profile, and, without a profile, the system cannot make recommendations.

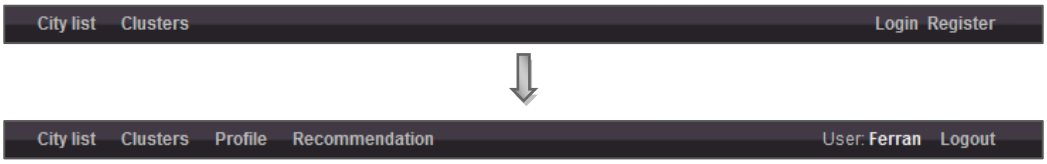

Figure 8: Menu differences when the user is not looged in and when it is.

- The content (or body) is the part of the Web that changes on every section of the menu. Hence each section will be explained in the following sections. The content for the main page is a photo of the Tower Bridge in London, just for decoration purposes. This image is licensed under Creative *Commons Attribution*.
- The footer is used to put the rights of the DAMASK project and the logos of the university and the research group where it has been developed.

Figure 9: Footer of the Web

3.2 Sections of the Web

The application has four sections corresponding to the functionalities provided by the system. These are found in the menu bar. Each section has a menu item that opens a new Web page.

3.2.1 City list

In this page, a user can see the entire list of cities that have been included in the system. The cities are the 148 leading and most dynamic cities in terms of tourist arrivals, to the ranking made by Euromonitor International. For further information, see DAMASK internal report 3.2. Actually, the cities were 150, but two were almost without information and were discarded.

Each city is represented as a table. The header of the table has the name of the city in bigger font size and different colours. Then in the content of the table, each row is each one of the semantic attributes with the list of concepts that the city has. The last row represents the numerical (population and elevation) and categorical (continent and climate) values of the city.

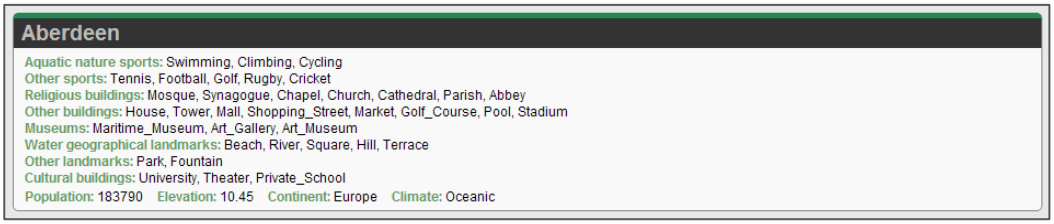

Figure 10: Representation of a city in the DAMASK Web

3.2.2 Clusters

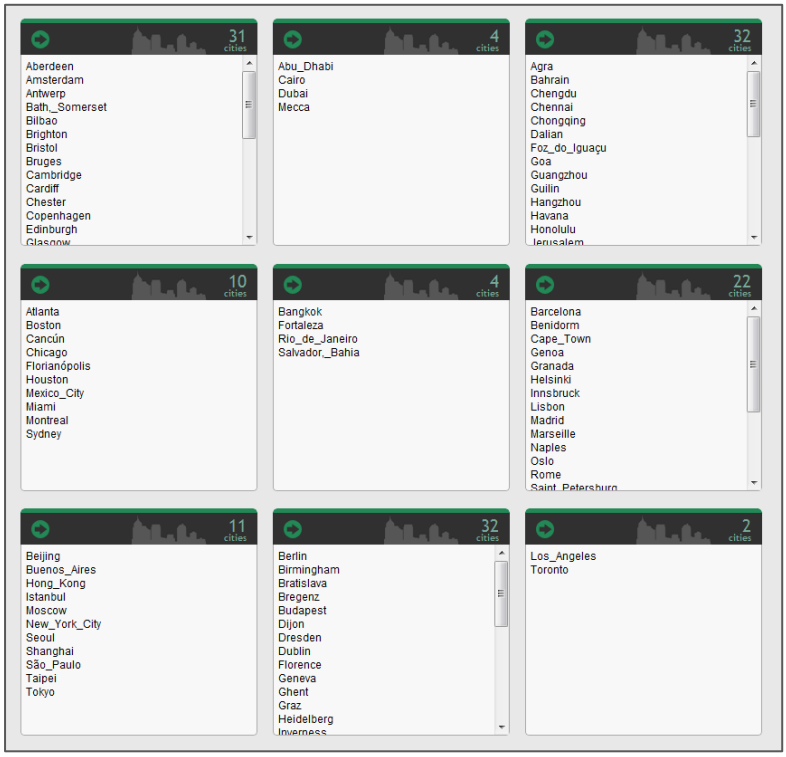

Figure 11: Representation of the clusters in the Web

In the cluster section, the user can see 9 blocks representing the 9 clusters of the system. Each cluster has the list of their cities (just the name, not the explicit table like the one seen above). The number of cities that the cluster has is also represented (Figure 11).

In the clusters with a large list of cities, a scroll bar appears. This is done to maintain the same size in all the blocks, which results in a nicer view than having blocks of different sizes of cropped lists of cities.

One important thing here is that if the user wants to see the cities of a cluster at detail, he can click the header of a cluster (that works as a link) to access to a different page that lists the cities of the cluster in a similar way that the *city list* section does.

This cluster's city list page is no different than the city list except for two things: (1) the main difference is that the cities are limited, obviously, to the ones that belong to the selected clusters, (2) the other difference is that at the top of the page appears the centroid of the cluster. This centroid is represented as the rest of cities, with a table with the semantic attributes in rows and the numerical and categorical attributes at the bottom, but with the difference that the semantic attributes indicate also the weight assigned to them. The weights are presented without the normalization, so that they correspond to the frequency of appearance of the concepts in the cluster (see Figure 11).

In order to facilitate the identification of the centroid, the colour is different than the one used for the list of cities (see Figure 12).

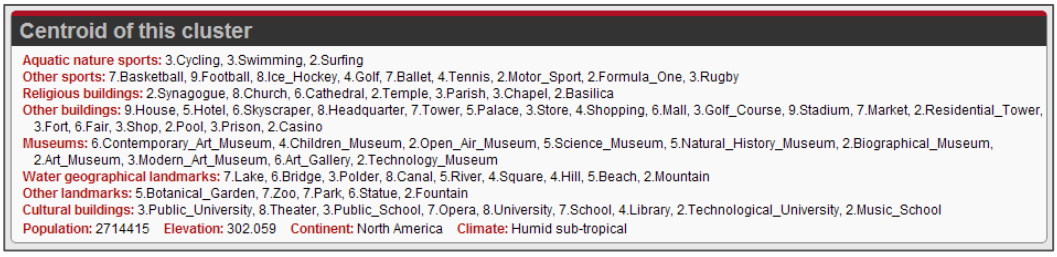

Figure 12: Representation of the centroid of a cluster.

The next image represents the view of an entire cluster. The idea is that the user can compare the concepts on each of the cities and the centroid that represents the cluster. Notice that at the top of the page, a number appears telling the user the number of cities that the cluster has.

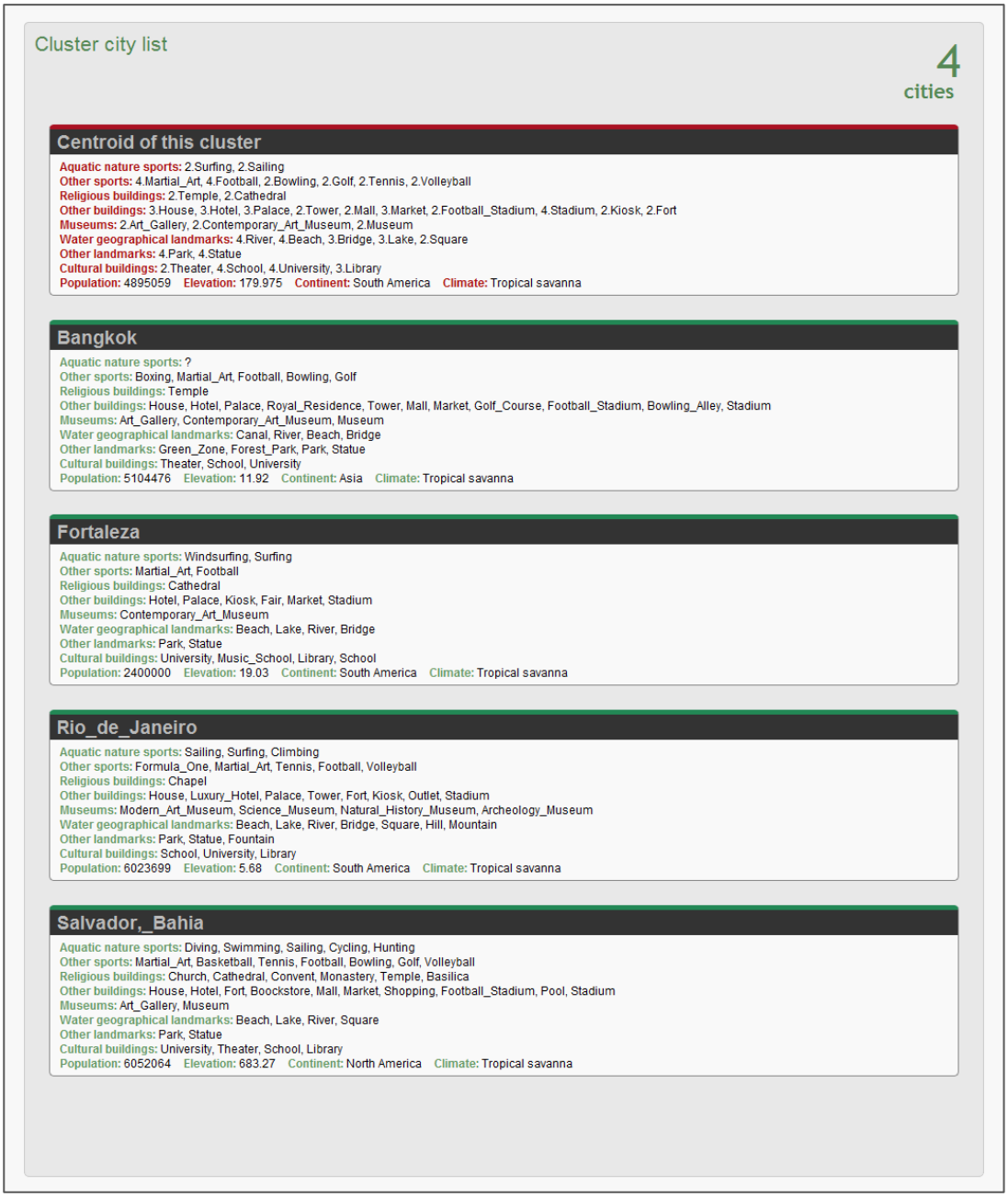

Figure 13: Detail view of a cluster with the centroid and its list of cities.

3.2.3 Profile

This section is only available for registered users that are logged in. This page provides some options for a user to build a personal profile, which will be used to make recommendations to the user based on the similarity with the centroid of the cluster.

In this section we first present the details about the structure and contents of the user profile that have been defined in the project. Secondly, the interface through the Web page is detailed.

3.2.3.1 Structure of the user profile

The different alternatives available in the system, the objects, are defined using three types of data: numerical, categorical and semantic attributes. Therefore, the user must be able to express some kind of preference information regarding each of those types of attributes. The requirements can be given in two forms:

a) Mandatory constraints or filters: the user indicates the subset of values he is interested on. Any value outside this subset is discarded.

b) Preference requirements: the user indicates which values are the ones he prefers, but without discarding similar values.

Each application domain will require a different type of requirement for each attribute. In the case study in the DAMASK project about selection of touristic destinations, the semantic attributes are the criteria that define the type of destination desired by the user, while the numerical and categorical attributes refer to contextual information, so they can be better used as filtering criteria. The following table summarizes the criteria and its role in the user profile. For each of the attributes, the range of possible values is also indicated.

Table 1. Attributes by types and its values for the user profile.

Preference attributes	Type	Values
Aquatic nature sports	S	Swimming, Rafting, Surfing, Diving, Kayaking, Skiing, Snowboarding, Climbing, Mountain_Biking, Cycling, Water_Polo, Windsurfing, Waterskiing, Hunting, Sailing
Other sports	S	Football, Squash, Roller_Hockey, Formula_One, Skateboarding, Hockey, Boxing, Silat, Basketball, Rally, Popular_Running, Judo, Golf, Bowling, Swimming_Race, Stickball, Motor_Sport, Badminton, Volleyball, Martial_Art, Ballet, Handball, Street_Hockey, Rugby, Tennis, Karate, Table_Tennis, Cricket, Ice_Hockey, Paddle
Religious buildings	S	Synagogue, Mosque, Chapel, Sanctuary, Temple, Cathedral, Basilica, Religious_Building, Abbey, Church, Parish, Convent, Monastery
Other buildings	S	Casino_Resort, Stadium, Flea_Market, Mall, Cotton_Mill, Residential_District, Trade_Fair, Royal_Residence, Formula_One_Circuit, Golf_Course, Skyscraper, Football_Stadium, Velodrome, Headquarter, House, Prison, Shopping_Centre, Kiosk, Pool, Souvenir_Shop, Shop, Sport_Complex, Residential_Building, Hockey_Stadium, Townhouse, Market, Palace, Residential_Tower, Store, Outlet, Luxury_Hotel, Shophouse, Fair, Tower, Boockstore, Industrial_Building, Food_Market, Fashion_Shop, Aquatic_Centre, Fort, Shopping_Street, Supermarket, Shopping, Squash_Centre, Camping, Baseball_Stadium, Casino, Hotel, Hypermarket, Convenience_Store, Garden_Apartments, Bowling_Alley, Antiquarian_Shop, Shopping_Area
Museums	S	Modern_Art_Museum, Erotic_Museum, Egyptian_Museum, Toy_Museum, Open_Air_Museum, Railway_Museum, Fishing_Museum, Music_Museum, Wax_Museum, Biographical_Museum, Folk_Art_Museum, Aviation_Museum, Maritime_Museum, Military_Museum, Technology_Museum, Sex_Museum, Art_Gallery, Archeology_Museum, Art_Museum, Science_Museum, Contemporary_Art_Museum, Museum, Industrial_Museum, Astronomy_Museum, Children_Museum, Natural_History_Museum, Woman_Museum, Computer_Museum
Water geographical landmarks	S	Hill, Stone_Bridge, Square, Polder, Bridge, Cave, Terrace, Pedestrian_Bridge, Canal, Gorge, Beach, Lake, River, Mountain
Other landmarks	S	Tomb, Sepulchre, Obelisk, Park, Urban_Park, Statue, Garden_Park, Nature_Reserve, Ionic_Column, Zoo, Pyramid, Ancient_Obelisk, Historic_Park, Column, Fountain, Megalithic, Green_Zone, Refuge,

		Forest_Park, Crypt, Egyptian_Obelisk, Mausoleum, Suburban_Park, Botanical_Garden
Cultural buildings	S	Private_School, Opera, Ancient_Greek_Theatre, Business_School, Forum, Public_School, Music_School, University, Private_University, Public_University, Library, Roman_Amphitheatre, Technological_University, Theater, School, Art_School, Amphitheatre
Filter attributes		
Population	N	0 - 15.000.000
Elevation	N	0 - 2.500
Continent	C	Europe, Asia, Africa, North America, South America, Oceania
Climate	C	Alpine, Desert, Humid continental, Humid sub-tropical, Mediterranean, Oceanic, Polar, Semi-arid, Subarctic, Tropical monsoon, Tropical rainforest, Tropical savanna

For each of the filtering criteria, the user will determine the range of accepted values. For example to indicate low values for the Population if one wants to be in a small city.

For the preference criteria, two types of information must be given:

- The *weight*, interpreted as the relative preference of that attribute with respect to the other.
- The *preferred values*, a list of terms (semantic concepts) that fit with the user's interests.

3.2.3.2 Definition of the profile in the Web page

The Web page has two differentiated sections: one with the semantic attributes to indicate the user's preference profile, and one with the numerical and categorical attributes that work as a filter for the recommendation result.

The part with the preference semantic criteria is presented with sliders, one for each one of the criteria, which are used to indicate the degree of importance that the user wants to give to the attribute (Figure 14). In other words, the user will set a preference between 0% and 100% for each semantic attribute. This weight is used when the similarity with the clusters' centroid is computed.

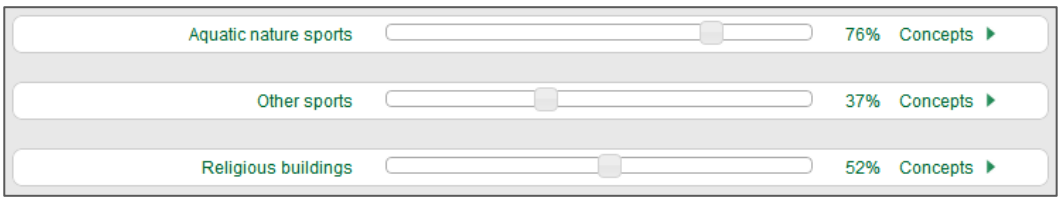

Figure 14: Respresentation of the semantic attributes preference

In addition, the user has to introduce which are the values that we is looking for in the city. The user can click the "concepts ►" link in order to unfold the corresponding section and see the list of concepts available to check. The system builds a list with the selected concepts, which is compared against the list in clusters' centroids. This process is explained in the next

section. An extra checkbox is provided as a *select all* / *unselect all* toggle mechanism, as shown in Figure 15.

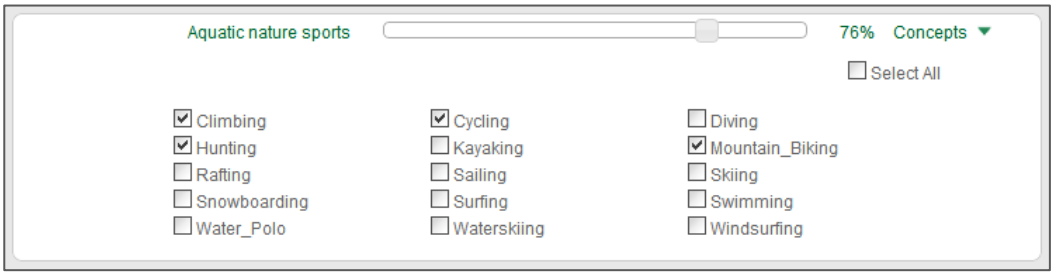

Figure 15: Representation of a semantic attribut list of concepts

If some criterion is set to 0%, it indicates that the user will not give any preference information regarding this attribute, so it is not used in the comparison with the centroids of the clusters.

The other part of the same page is the filtering section, used to reduce the resulting list of recommended cities, just keeping the cities that fulfill the constraints indicated by these criteria.

The numerical filters are implemented with two range sliders, whereas two selection lists elements are used for categorical attributes (Figure 16).

Figure 16: Representation of the filter in the profile page.

There is one interesting thing here, the values of the sliders do not increase in an arithmetic progression way, but they do it following a geometric progression instead. Notice that the city with the lowest population has only 20.000 people, and the biggest has about 15.000.000. The difference is enormous, and because of that, the change of the value of the slider for just 1 pixel is also big. This caused, in a first implementation, that the range of the majority of cities was too small. The values rapidly passed from 0 to 50.000 in just one pixel or two, what is horrible when 60 cities, the 40% of the total cities, are in that range. The same occurs with the Elevation since the majority of the most touristic cities are placed in the coast. Remember the graphics of frequency distribution surveyed at DAMASK report 3.2:

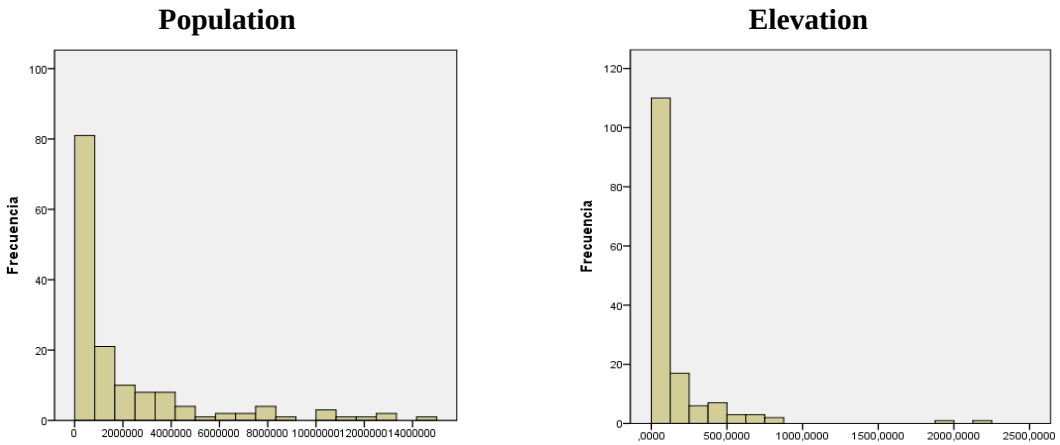

Figure 17: Frequency distribution of the numerical variables

Hence, to solve this issue, we have implemented a function to transform the arithmetic progression of the sliders provided by jQuery into a more suitable geometric progression. With this, the values of the slider are better distributed. Starting with a change of just 1000 and ending with changes of about 500.000.

The select elements for the continent and the climate show all possible values plus the "*Any*" option, that is selected as default. The selection of "*Any*" indicates that the used does not want to use the *filter* option, so that all the values are accepted.

Finally, there are two buttons at the bottom of the page, "*Save*" and "*Save and recommend*". The first one is just to save the profile of the user in the database. The second does the same, but redirects to the recommendation page automatically (which is the expected flow of events).

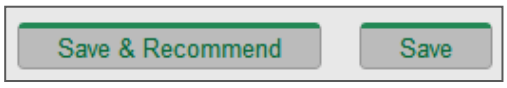

Figure 18: Buttons of the profile page.

The following image is an entire view of the profile page. An attribute was unfolded to see its concepts. Some of these concepts were checked. The filters were also changed in a manner that the recommendation will filter for European cities with population above 45.000 and elevation below 166. The list of cities in the recommendation will have any kind of climate since "*Any*" is the selected option.

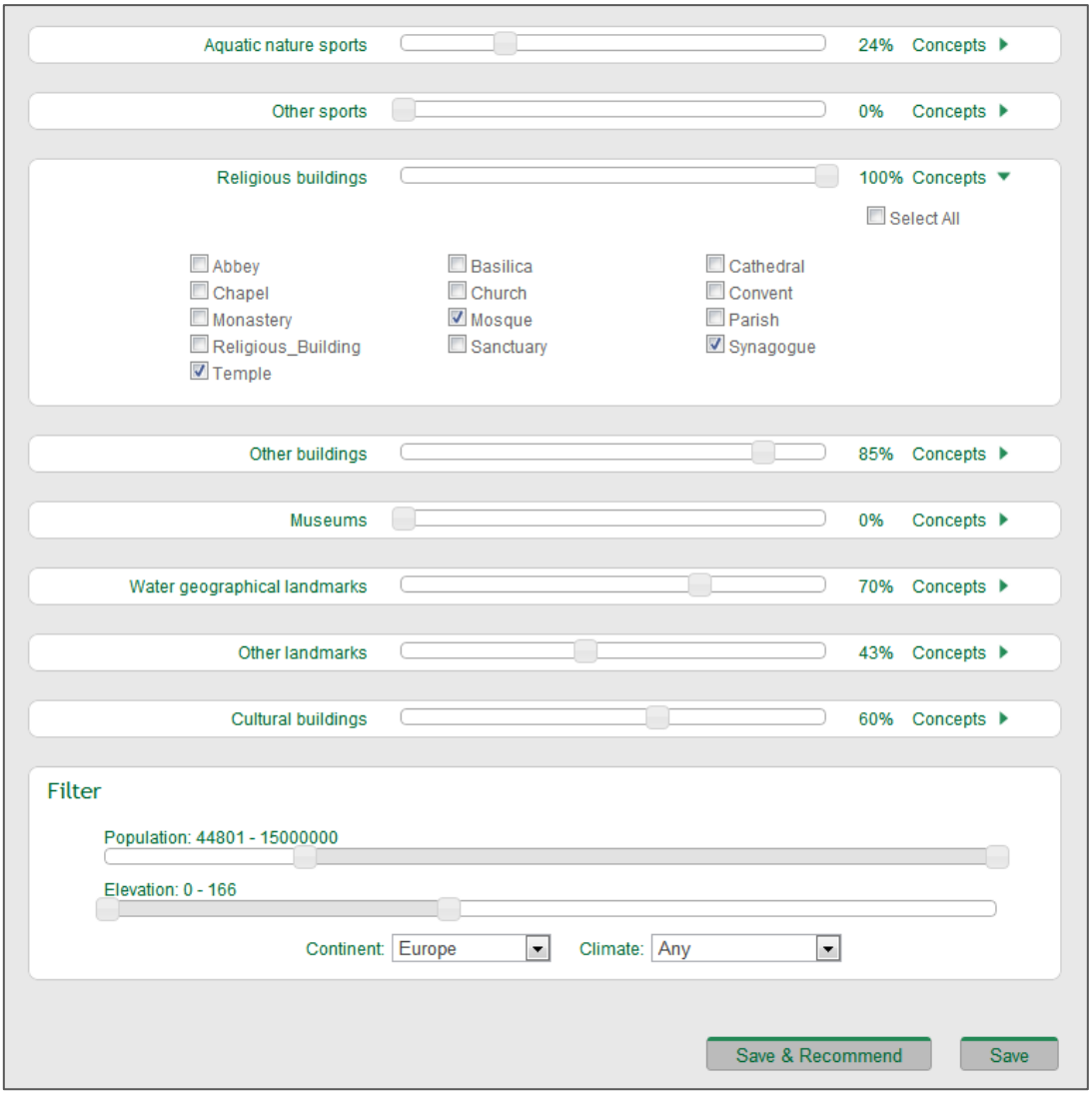

Figure 19: View of the profile page with an unfold attribute and some concepts checked.

3.2.4 Recommendation

Once the user has completed its profile both with the preference information and the filtering criteria, the system starts the recommendation process. If a user tries to access the recommendation page without completing his/her personal profile, the system will redirect the user automatically to the profile page.

3.2.4.1 The recommendation algorithm

Retrieve from database the **user profile**
Retrieve from database all the **clusters centroids**
For each centroid {
    For each attribute of the centroid {
        Compute the **distance** from the centroid value to the user profile value multypling the distance by the weight of the attribute in the user profile
    }
    Sum all the weighted distances to obtain the **absolute distance** between the user profile and the centroid
}
Select the centroid with the lowest distance and **retrieve** its **cluster** from DB
Filter the cities of this cluster according to the ranges given in the user's profile
The clusters of the centroids whose distance is **below a threshold** are also filtered and saved as "**additional cities**" set
Show the results to the user

The recommendation process starts creating the prototype of a city from the concepts selected in the profile page. This prototype only has semantic attributes since the numerical and categorical are reserved to filtering purposes.

Then, the distance between the prototype and the centroids of the different clusters is computed in a similar way that it has been done in the clustering application. The prototype works as a plain city in this procedure. When the algorithm is at the point where the distance is computed for each semantic attribute, each of these distances is multiplied by the factor that the user specified in his profile. After that, the semantic distance is computed. This distance works here as the definitive distance, because the numerical and categorical distances are not computed.

The following step is devoted to check which centroid is the most similar to the prototype. The selected centroid is used then to get the cluster that represents. Finally, the list of cities of the cluster is filtered using the user's profile indications.

3.2.4.2 Recommendation of touristic city destinations in the DAMASK Web application

The results are displayed in a similar way than the list of cities with their description (option City List). However, we find some particularities.

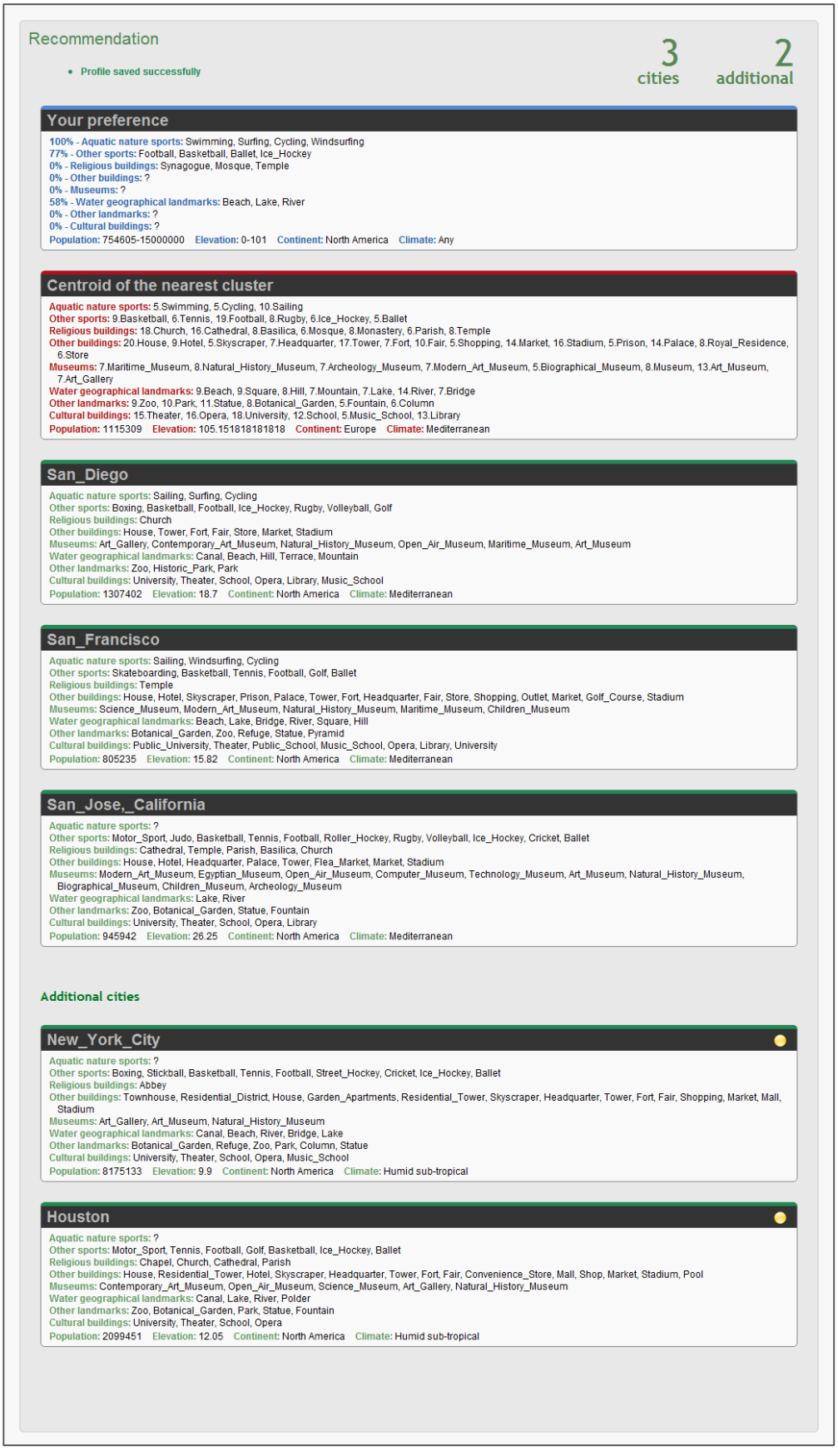

Figure 20: View a the recommendation page, with some cities of the recommended cluster and some others additional.

The first *block* shown represents the user profile introduced in the system (Figure 21). For each preference semantic attribute, the relative weight is indicated in percentage, together with the selected concepts. Filters are also displayed at the bottom. Different colors are used to differentiate it from the list of recommended cities.

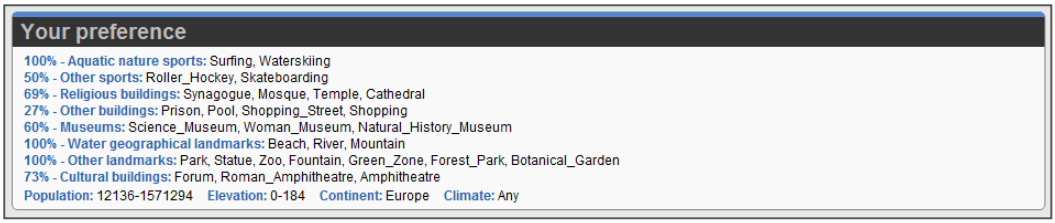

Figure 21: Representation of the prototype

After the user profile representation we can find the centroid of the most similar cluster (displayed as in the *clusters* option). Below we can find the list of recommended touristic cities. Only the ones that have passed the filters are displayed.

When the filters are too strict (allowing a small interval of possibilities) the list of destinations recommended may be too short. To allow the user to find some other alternatives, it has been implemented a mechanism to show additional cities, that are also similar to the user's profile. All the clusters within a distance to the profile smaller than a given threshold are considered. All the cities that belong to those clusters are filtered according to the mandatory criteria and a secondary list of alternatives is obtained.

At first, the additional cities are hidden in the Web page, and they will only be showed if the user clicks the "*Additional cities*" link at the bottom of the page. To differentiate the additional cities from the main ones, a yellow ball icon is added at the top right of the *city box*.

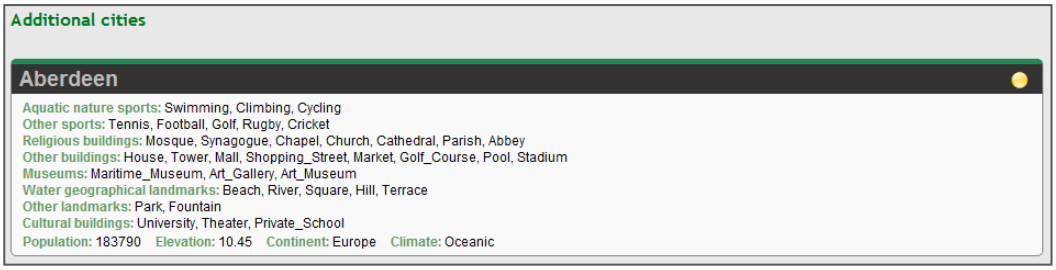

Figure 22: Representation of an additional city and the "*Additional cities*" link.