Task-Aware Location-Based Services for Mobile Environments

FP7-SME-207-1-222292-TALOS

Annotation Tool
D3.2

Deliverable lead contractor: CTI

Alexandros Efentakis, CTI efedakis@cti.gr
Anna Stathaki, CTI stathaki@cti.gr
Dieter Pfoser, CTI pfoser@cti.gr
Tassos Arvanitis, IMIS anarv@imis.athena-innovation.gr
Stefan Pfennigschmidt, ISST stefan.pfennigschmidt@isst.fraunhofer.de

Due data: 31.3.2010
Actual submission date: 26.4.2010

Abstract

Presents the annotation tool allowing for Web-based annotation of content with (i) geocontent, (ii) tasks, and (iii) dynamic Web content (Web scraping).

Research Academic Computer Technology Institute, Greece
Fraunhofer Gesellschaft, Institute for Software and Systems Engineering, Germany
Institute for the Management of Information Systems / Athena Research and Innovation Center in Information, Communication and Knowledge Technologies, Greece
Katholieke Universiteit Leuven, Belgium
Michael Müller Verlag, Germany
Talent SA, Greece
WiGeoGIS, Austria

Table of Contents

1 Introduction

One of the prerequisites of the TALOS project is creating a content repository that will be fully linked with the task hierarchy. Still, the task annotation of content is only one of the various components of the TALOS project. Other important components are:

- The POI (Points of Interest) annotation of content (linking content with various points of interest)
- Creating a geocoding framework for batch and manual geocoding of points of interest
- A web scraping tool for enhancing existing content and POI information with web content

The annotation tools represents an effort to combine all those separate and individual tools and frameworks, into a simple, concise, easy to use web application. A web application was the obvious choice, since:

- It requires no installation on the client's machine, as it works within the browser
- It is easy to use and learn, since it is very similar to a normal web site
- Various free tools and frameworks exist for creating rich web applications

In the following sections, we will fully describe the various data components of this web application and their storage format. We will also describe the various tools and frameworks we used during the creation of the TALOS annotation web tool and consequently provide a detailed and comprehensive tutorial covering the most important aspects of the application.

2 The data components

2.1 Content

The raw data which will be the actual source of TALOS content originates from travel guides provided by TALOS partner Michael Müller Verlag (MMV).

MMV mainly deals with printed travel guides (although it features some electronic guides as well). Therefore its travel guides are mainly focused for printing purposes. From its beginning MMV used Microsoft Word for its DTP jobs. Consequently, the travel guides provided for the TALOS project are in Microsoft Word 2007 format (docx files). The docx format is the first effort by Microsoft to provide a purely XML based format for storing office documents. This format is relatively new, so very few developers have actually developed applications for docx creation and manipulation. Also, since it is created by a commercial company like Microsoft, it is faced with mixed emotions from the open source community which favours the ODF format (the Open Office XML based format for office documents). One of the main challenges of the TALOS project is to efficiently manipulate this rather new format. For the TALOS project, MMV agreed to provide the content of their travel guide for Brussels.

In order to make the travel guides more accessible to the TALOS framework and for internal use, MMV developed a custom 'ingesting' application for storing the contents of the docx files comprising a travel guide, into a Microsoft SQL Server 2008 database. SQL server 2008 was chosen by MMV, over other RDBMS (PostgtreSQL was initially proposed for the TALOS project), since MMV uses on their product line strictly Microsoft tools (Word, Visual Studio). Since, SQL server 2008 was already chosen for internal MMV use, the TALOS framework will also use SQL server 2008 (meaning its free version SQL Server 2008 Express edition) in order to avoid the overhead of maintaining and synchronising 2 separate databases stored in 2 different RDBMS. The choice of RDBMS had no important impact on the creation of the Talos annotation web application, since the tools and concepts used are RDBMS neutral.

As a result of the MMV ingesting process, each section of the Word document is stored as a separate record on a database table. The sequence of sections within the document was also stored as a separate field. For each of those sections, MMV stores the following information into the SQL server 2008 database:

- The OpenXML (the Microsoft's proprietary XML format used inside the docx file) representation of the section
- The Rich Text Format (RTF - another proprietary older format for DTP documents) representation of the section
- The plain text representation of the same section

Unfortunately, none of those representations can be used as it is, in a Web application, since:

- OpenXML and RTF are DTP formats and not web standards
- The plain text representation is extremely "poor", since it is devoid of any formatting instructions.

Preserving format is crucial for our purposes, since formatting styles are used in a DTP document, not just for appearance reasons, but also for showing the document structure (chapter - section titles, image legends, indexes). Our web representation of MMV content should be as close to its DTP counterpart in order to:

- preserve the original document's structure
- be easy for someone to easily identify the web version of a section with its DTP version
- be visually attractive

Therefore, CTI chose to convert the OpenXML representation of a section into XHTML format for use in TALOS annotation application. The OpenXML format was used as source, because it includes all the necessary formatting information, including paragraph and character styles, formatting instructions ("bold", "italics"), even hyperlinks and image paths.

Converting OpenXML to XHTML was a time consuming and tough process. Given that OpenXML is a relatively new format, no existing tools exist for such a conversion (at least at a mature level). On the other hand, since OpenXML is basically XML with a documented XML schema, converting it to XHTML (another XML format) was basically a process of using XQuery and XPath in order to "break down" the original XML to its subcomponents, strip unnecessary tags and convert existing tags to their XHTML equivalents. Fortunately, SQL server 2008 supports, not only the storage of XML documents, but also a full XQuery and XPath engine for querying them. Subsequently, in order to convert the OpenXML representation of a document section to its XHTML representation, CTI had to develop stored procedures inside the SQL server 2008, which make extensive use of XPath and XQuery for the required conversion.

The result of this process was a database table with the following structure:

Table "contents":

- id (Primary Key and unique identifier of the section)
- content_html (XHTML representation of the section)
- content_toc (XHTML representation of the section title – useful for showing a Table of contents)
- seq_indx (An integer showing the sequence of sections within a section)

Please keep in mind that the "id" and "seq_indx" fields were created during the MMV ingesting phase and were kept as they were, for synchronisation reasons.

The travel guide's content may be one important aspect of our data used in the TALOS annotation application, but it is not the only one. In the following section, we will talk about the other two data components of our application: Points of Interest (POIs) and Tasks.

2.2 Points of interest (POIs)

MMV provided a catalog of 33435 accommodation facilities (hotels, pensions, hostels, bed & breakfast facilities) covering the entire world. Information from those facilities was manually extracted and recorded from the MMV travel guides. Those facilities information was provided as a total of 880 Microsoft Excel files (as noted earlier MMV uses on their production line Microsoft products exclusively). A custom SQL server stored procedure was developed by CTI in order to ingest those accommodation facilities into our SQL server database. Although, only Brussels is the area of interest for our TALOS application, all hotels were ingested, so that the geocoding and web scraping portion of CTI's annotation tool should work for all available hotels.

Fortunately, during the MMV's ingesting phase of Brussels travel guide, content was annotated with the Brussels accommodation facilities (a total of 76 entries).

2.2.1 Identifying POI datasets

In order to enrich what was provided by MMV for Brussels, CTI searched for additional web sources that could provide additional POIs. Results of this effort are provided below:

2.2.1.1 Muselius effort

Muselius is a World museums directory. It also offers an API that lists museums per MBR. We used for Brussels the MBR provided by Yahoo geo services and we found a total of 11 museums. Each of those 11 museums has also a specific Muselius page devoted to them. Each of those web pages were web scraped in order to extract more information about a museum, and all extracted information was stored on our database. The information extracted was:

- address
- longitude
- latitude
- phone number
- website
- opening hours
- prices
- description

This process was automated, through a custom Java application developed by CTI, that:

- retrieves all museums from Muselius api (including the links leading to each museum specific web page on Muselius site) for a specific city MBR
- web scrape each museum's specific web page on Muselius and extract information (opening hours, description etc...)

2.2.1.2 ChefMoz effort

The ChefMoz Project (according to its creators) aims at developing the most comprehensive database of restaurants, coffee shops and bars worldwide (a Wikipedia for such establishments, including user reviews). Their restaurant database is freely available as an RDF file at http://chefmoz.org/rdf/20091110/chefmoz.rest.rdf.gz. It includes a total of 281827 restaurants with 114836 user reviews and includes many pieces of information like:

- restaurant's name
- restaurant's description
- address
- cross street
- neighbourhood
- city
- state
- country
- phone
- delivery phone
- fax
- URL
- delivery URL
- reservation URL
- menu URL
- type of cuisine
- working hours
- credit cards accepted
- alcohol served
- if smoking is allowed
- dress code
- parking facility
- additional features
- food rating
- service rating
- ambience rating
- overall rating
- recommended dishes
- accessibility
- accessibility notes
- if reservations are possible
- capacity

- largest party allowed
- clientele
- link resources

Documentation for each of those fields are provided in: http://chefmoz.org/rdf/elements/1.0/

All ChefMoz entries were inserted on our database, by using a custom Java application developed by CTI. For Brussels only it includes 909 restaurants, coffee shops and bars.

2.2.1.3 Booking.com effort

Booking.com is one of the largest websites dealing with booking hotels worldwide. It provides a wealth of hotels' information, such as many photos per hotel, detailed information about hotel features but most importantly, reviews from its many users. Although Booking.com offers a free XML API, this API is focused towards websites (that work as partners), since a part of the revenue from booking.com registrations originating from those sites, goes to those partner websites (something like Google Adsense). Therefore, CTI tried to extract the hotel information directly from the Booking.com website.

If one executes a query on the Booking.com website, like "get me all five star hotels with wireless", the query's result is a KML file (Google Earth custom format) showing the hotels that satisfy those criteria (up to a total of 50 hotels). As a result each Booking.com query is translated to a KML file. If we choose our queries carefully, like "give all hotels of a specific category" or "all hotels with a specific feature" (like wireless internet)" and so-on we can get a set of KML files (about 13 files per city) that can be later used for extracting information for the hotels in the city.

After getting the KML files (no more than 10 minutes of work per city), CTI created a custom Java application that extracts all information inside the KML files and also uses this extracted KML info to extract even more information directly from the Booking.com website (For example the URLs of photos for a specific hotel, that are not included in the KML file).

A sample of this process was done for Brussels: Information for 155 hotels was gathered (significantly more than what the 76 accommodation facilities MMV provided for Brussels). Each hotel entry is fully geocoded (another advantage lacking from MMV accommodation entries), so we know its exact latitude and longitude coordinates. The full information extracted is:

- hotel name
- category (one star - five stars)
- address
- latitude
- longitude
- short description (provided by the hotel owners)
- hotel features like spa, gym, wireless, disabled people support, allowing pets, airport shuttle, parking, rating from booking.com users
- total number of total user reviews
- total user rating

The URLs of the hotel photos are also stored on our db, so therefore for one hotel we have from 3 to 10 photos (as links to Booking.com website).

In order to avoid referencing the same hotel twice, the booking.com hotels were cross checked with the MMV accommodation entries (by name and address). In the end 46 hotels were common in those 2 catalogs.

2.2.1.4 DBpedia effort

The DBpedia data set is a large multi-domain ontology which has been derived from Wikipedia. The DBpedia data set currently describes 3.4 million "things" with over 1 billion "facts" (March 2010).

The DBpedia data set currently consists of over 1 billion RDF triples, which have been extracted from 92 language versions of Wikipedia.

The DBpedia knowledge base currently describes more than 3.4 million things, out of which 1.5 million are classified in a consistent Ontology, including 312,000 persons, 413,000 places (including 310,000 populated places), 94,000 music albums, 49,000 films, 15,000 video games, 140,000 organizations (including 31,000 companies and 31,000 educational institutions), 146,000 species and 4,600 diseases. The DBpedia data set features labels and abstracts for these 3.2 million things in up to 92 different languages; 1,460,000 links to images and 5,543,000 links to external web pages; 4,887,000 external links into other RDF datasets, 565,000 Wikipedia categories, and 75,000 YAGO categories. The DBpedia knowledge base altogether consists of over 1 billion pieces of information (RDF triples) out of which 257 million were extracted from the English edition of Wikipedia and 766 million were extracted from other language editions.

A sample URL like http://dbpedia.org/page/Brussels returns all available information in Wikipedia for the city of Brussels in a structured way. The RDF returned, has a "is dbpedia-owl:Place/location" element that describes all available landmarks (that have separate Wikipedia entries) within the city of Brussels and their subsequent URLs on the DBpedia ontology.

CTI developed a custom Java application that:

- reads the DBpedia entry for a city (Brussels)
- extracts the DBpedia URLs for the city's landmarks
- visits those distinct DBpedia URLs and extracts information for those landmarks
- stores the extracted information on our database

The information extracted for each landmark:

- dbpedia URL (The DBpedia URL for this specific landmark)
- thumbnail URL (The URL of a thumbnail photo – extracted from Wikipedia - for this specific landmark)
- photo collection URL (Usually one or more Flickr photo albums for this specific landmark
- wiki URL (The Wikipedia URL for this specific landmark)
- URLs (Does this landmark has a dedicated web site?)
- description (A short Wikipedia text summary for this landmark)

As a result a total of 23 landmarks were extracted for the city of Brussels.

2.2.2 Summary

Although a rather limited subset of POIs was provided for Brussels by MMV (76 accommodation facilities) extensive work has been done to enhance and expand the available POI information for Brussels. This process (although not directly linked to the annotation tool developed by CTI) had two primary objectives:

- Improve and expand the TALOS mobile prototype application
- Work as a test case for MMV partner, on how traditional travel guides should and may be expanded by available web sources

2.3 Tasks

The third data component of the annotation tool is Tasks. The task annotation process after all, requires not only content but sample tasks as well. Tasks were stored in a database table with the following simplified recursive structure:

- task_id1 (The parent task id)
- task_name1 (The parent task name)
- task_id2 (The child task id)
- task_name2 (The child task name)
- task_level2 (how deep is the child task within the task hierarchy – eg the root task is of level 0, children of the root task are level 1 and so-on)

It is implied that the root task has no parent task. We filled those tasks with a subset of tasks suggested by project partner ISST that will be also used for the TALOS mobile prototype application. Although, the proposed task hierarchy is rather limited, it is sufficient enough for our sampling purposes. After all, TALOS annotation tool may work with as many as tasks required, as long as the task hierarchy follows the previous storage format.

The suggested task hierarchy (used in both the mobile application prototype and the Talos annotation tool) is shown below.

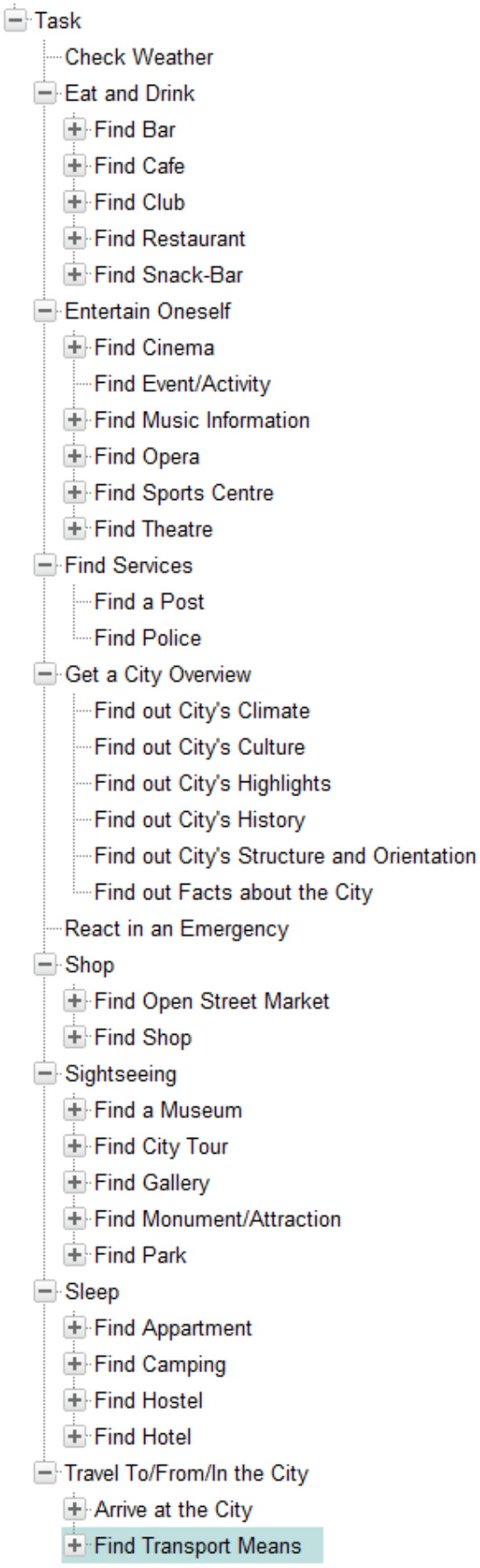


Figure 1: The task hierarchy (partially expanded)

3 Implementation of TALOS annotation tool

As noted earlier the backend database of TALOS annotation tool is SQL server 2008 Express edition. The web application was built using Ruby on Rails. Ruby on Rails is an open source web application framework for the Ruby programming language. It is intended to be used with an agile development methodology that is used by web developers for rapid development. Like many other web frameworks, Rails uses the Model-View-Controller (MVC) architecture pattern to organize application programming.

Ruby on Rails includes tools that make common development tasks easier "out of the box", such as scaffolding that can automatically construct some of the models and views needed for a basic website. Also included are WEBrick, a simple ruby web server, and Rake, a build system. Together with Rails these tools provide a basic development environment.

Ruby on Rails relies on a web server to run it. Mongrel may be used instead of WEBrick but it also may be run on Lighttpd, Abyss, Apache (either as a module - Passenger for example - or via CGI, FastCGI or mod_ruby), and many others.

We decided to use the JRuby implementation of the Ruby programming language, in order to overcome compatibility issues between Ruby and SQL server 2008. JRuby is a Java implementation of the Ruby programming language and it is also free software released under a three-way CPL/GPL/LGPL license. JRuby is tightly integrated with Java to allow the embedding of the interpreter into any Java application with full two-way access between the Java and the Ruby code and according to its creators support JRuby (in some cases) is even faster than the standard Ruby implementation.

JRuby supports Ruby on Rails since version 0.9 (May 2006), with the ability to execute RubyGems and WEBrick. JRuby version 1.0 successfully passed nearly all of Rails' own test cases.

Another advantage of the JRuby path we chose (apart from better UTF8 support and compatibility with SQL server 2008) was the fact that with JRuby, our Rail applications can run alongside Java Web applications on existing Java EE application servers, like Tomcat or Glassfish. These application servers have a strong technical infrastructure. The JVM has a much more sophisticated security model than Ruby's, giving JRuby on Rails tools for dealing with Web applications' typical security challenges, including control of Ruby scripts received from various sources. It also includes built-in support for internationalization. Simply by running on the JVM, these application servers gain the benefit of the immense optimization efforts put into the JVM over the last decade.

Therefore for development reasons, our TALOS annotation tool runs on a Windows 2003 virtual machine with Apache Tomcat and SQL server 2008 installed.

3.1 Javascript libraries used

Rails uses extensively the JavaScript libraries Prototype and Script.aculo.us for its Ajax purposes, out of the box. For additional purposes, like the use of sortable data grids, context menus and the tree view of our task hierarchy, we used the Yahoo! User Interface (YUI) JavaScript library.

YUI is an open-source JavaScript library for building rich, interactive web applications using techniques such as Ajax, DHTML and DOM scripting. YUI includes several core CSS resources and it is available under a BSD License and is free for all uses. Development on YUI began in 2005 and was released for public use in February 2006. It is actively developed by a core team of Yahoo! engineers. CTI used YUI2 version of the library, since version 3 is still under development.

The geocoding part of our TALOS Web annotation tool was done by utilizing Google Maps JavaScript API for embedding the necessary Google Maps in our application. CTI used the latest version of the Google Maps API (v3 at the time of production).

The web scraping part of the application utilizes a custom Firefox extension developed by CTI. As a result, in order to have maximum functionality the latest version of Mozilla Firefox is required. Firefox extensions are basically zip files suffixed by .xpi extension and include XML and Javascript files. In order to expand Firefox's user interface, XUL is used. XUL is Mozilla's XML-based language that may be used to build feature-rich cross platform applications on Mozilla platforms.

3.2 IDE used

In order to develop any kind of application a high-quality Integrated Development Environment is required. CTI chose to use Aptana Studio for many reasons:

- Aptana Studio is a complete web development environment that combines powerful authoring tools for Ruby, HTML, CSS, and JavaScript, along with thousands of additional plugins created by the community.
- Aptana Studio's editors provide Ruby, HTML, CSS, and JavaScript code completion, reference, and validation at your fingertips.
- Aptana Studio provides support for popular JavaScript libraries including jQuery, Prototype, YUI, dojo, Ext JS, MooTools, and others.
- Aptana Studio provides powerful plugins and ready-to-use runtimes for Ruby on Rails, Python and PHP.
- It is free, open source and cross platform (since it is based on Eclipse) and works on Windows, Mac, or Linux.
- It features multi-browser previews, SQL database tools, a JavaScript debugger and server tail views.

4 Usage of the TALOS annotation Web application

In order for someone to use the TALOS annotation web application, Firefox should point to the following URL:

http://web.imis.athena-innovation.gr:11290/jrails

The login screen appears:

Figure 2: The login screen

The user may login with the following credentials:

Username: **talosuser2**

Password: **12345678**

4.1 POI annotation

Once the user logins successfully, he is forwarded to the POI annotation fragment of the application:

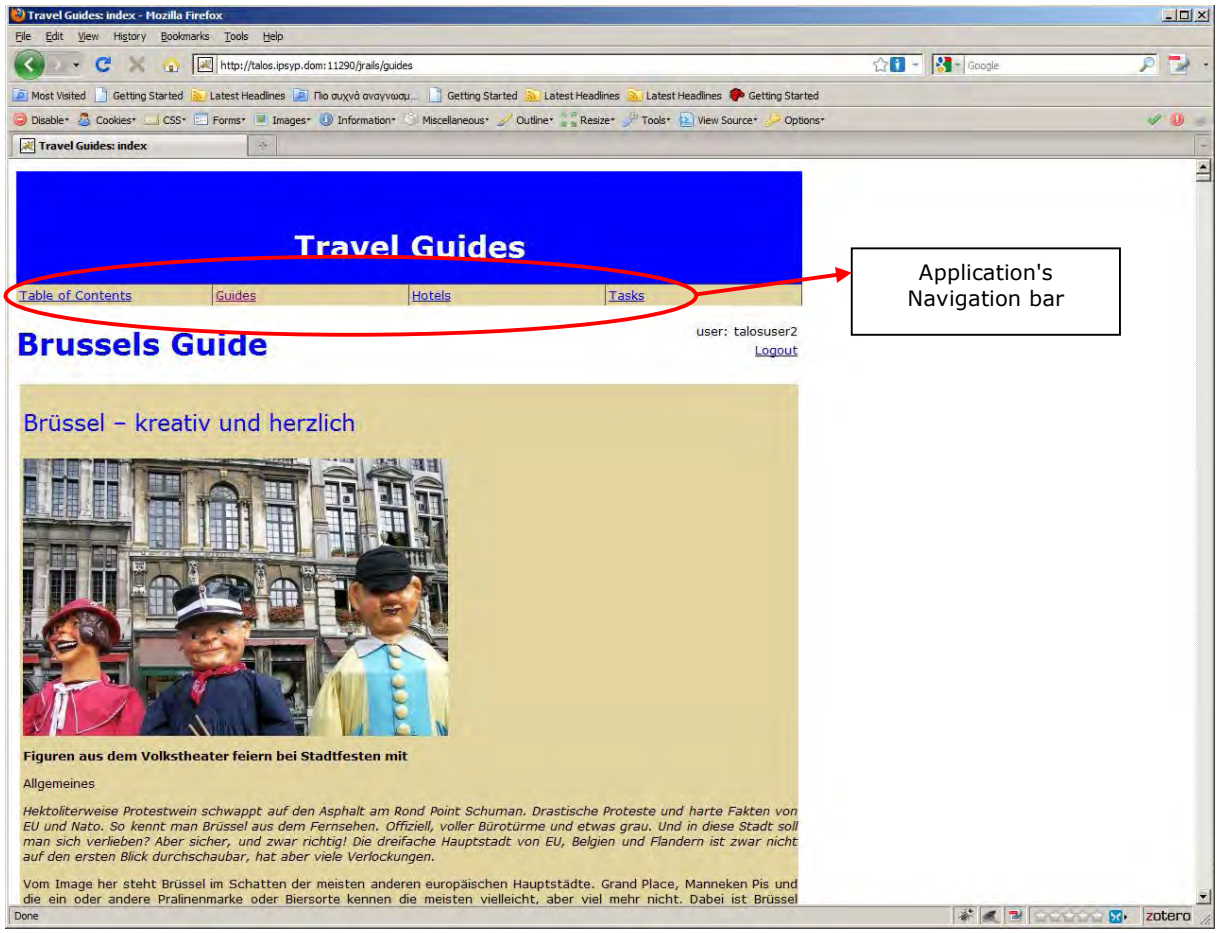


Figure 3: The travel guide's content in full colour, including images and hyperlinks

On the top of the window, the user sees the application's navigation bar, with the following options:

- Table of contents
- Guides
- Hotels
- Tasks
- Admin (This menu is visible, only if the user logged in as the 'admin' user)

By clicking on those menu choices, the user may use the individual subtools of CTI's TALOS annotation tool.

As seen on the previous figure, the user initially sees the contents of the travel guide (for Brussels). By using the browser's scrollbars on the right, one can navigate through the whole travel guide. As it is obvious, most paragraph and character styles are preserved and images and hyperlinks are fully supported.

Clicking on a travel guide's section, highlights the section and shows the accommodation facilities (POIs) linked with the selected section, in a new pop up window.

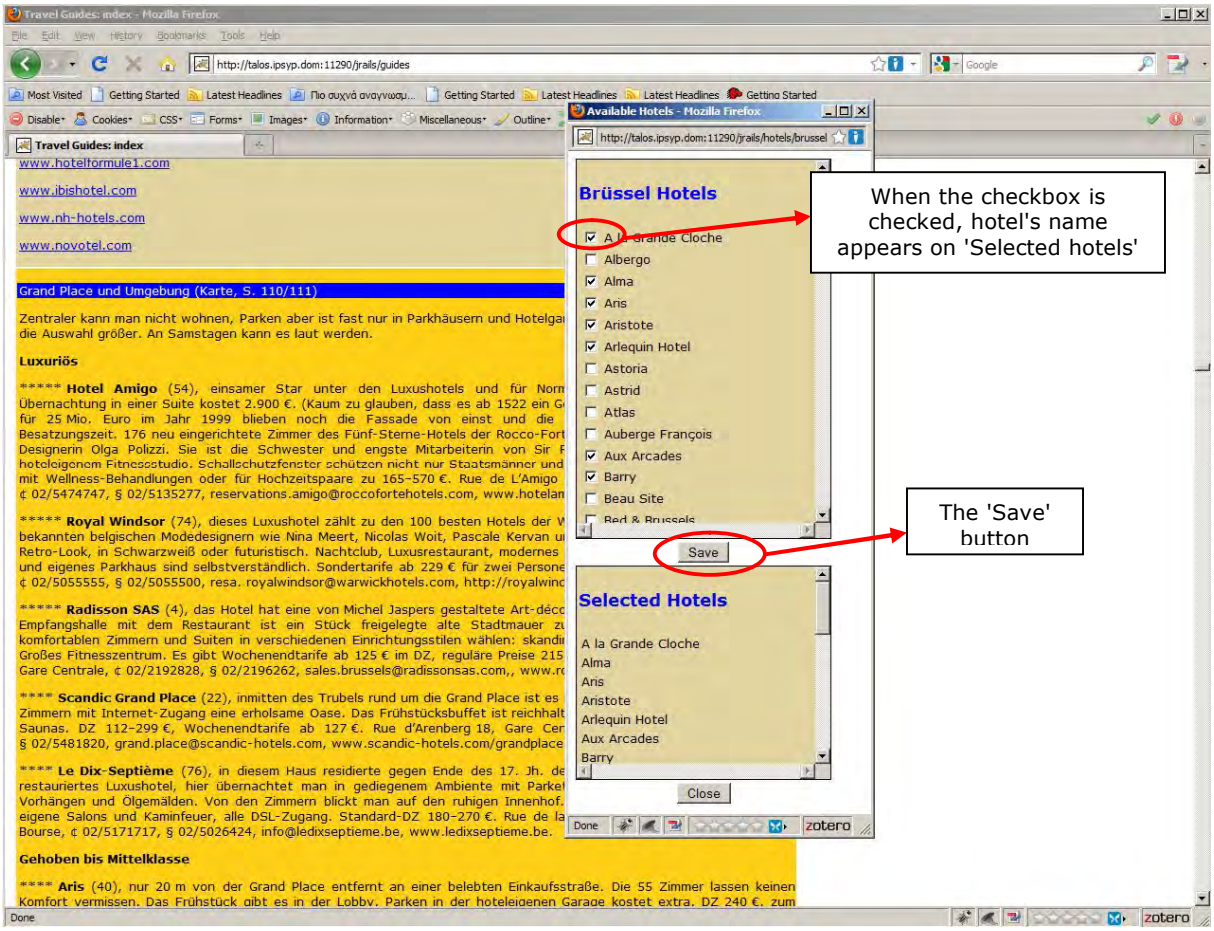


Figure 4: POI annotation of travel guide content

The popup window includes all accommodation facilities provided by MMV on the upper part. By clicking on a hotel's checkbox on the left of its name, the hotel name is added on the sorted list of hotels, linked with the highlighted section (shown in yellow) on the bottom of the popup window. Unchecking the checkbox, removes the hotel from the list of selected hotels.

Once the user selects all the POIs he wishes to associate with the highlighted section, he must press 'Save' in order to record its actions to the backend SQL server 2008 database.

At present, only the annotation of the POIs (accommodation facilities) provided by MMV is supported. On the other hand, supporting the additional POIs submitted by CTI (ChefMoz restaurants and coffee shops, DBpedia landmarks or Muselius museums) will be very easy to implement, with a similar interface.

If the user navigates to another menu option, he may return to POI annotation, by clicking the 'Guides' menu option on the application's navigation bar.

4.2 Task annotation

By pressing the 'Tasks' menu option on the application's navigation bar, the user navigates to the Task annotation part of the application.

Task annotation is pretty similar to POI annotation. The user initially sees the travel guide's content and may navigate by using the browser's scrollbars. As before, clicking on a section highlights the section and shows a new pop up window, with the suggested task hierarchy as a tree view and a list of already linked tasks with the specific section.

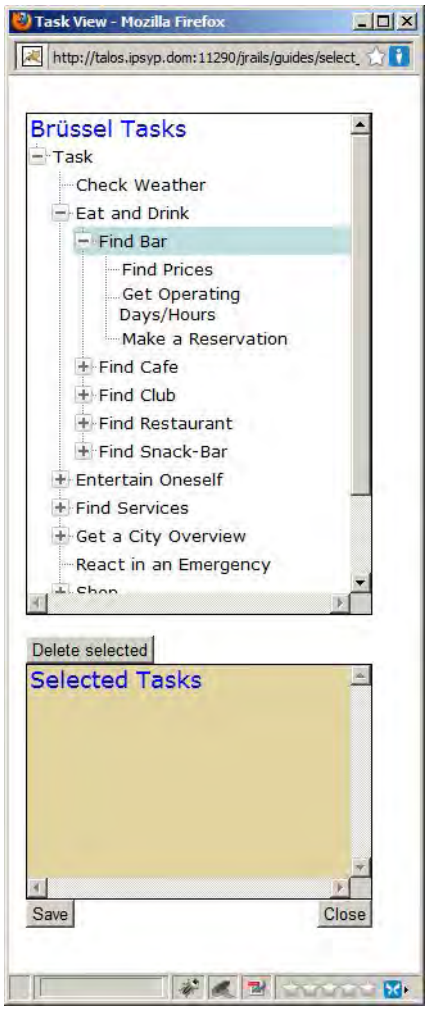


Figure 5: The task hierarchy

In order to expand the node of our task hierarchy tree, the user may click the [+] button on the left of the task and then the [+] button is converted to [-]. If the user clicks on a [-] button the children tasks of this particular tasks are hidden.

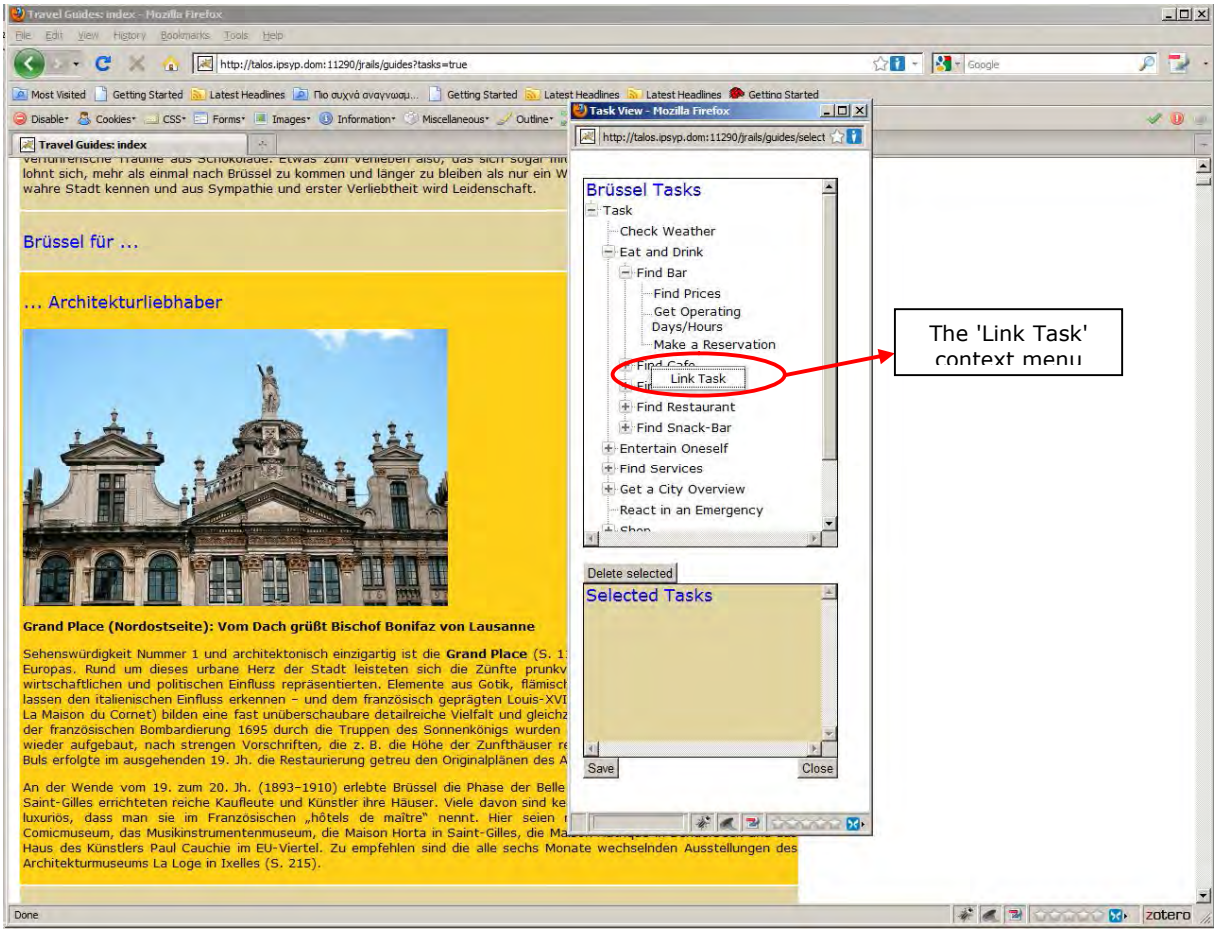


Figure 6: Task annotation of travel guide content

Right clicking on a task, shows a context menu 'Link Task'. Once the context menu is clicked, the selected task is added to the sorted list of selected tasks (associated with the specific section) at the bottom of the popup window.

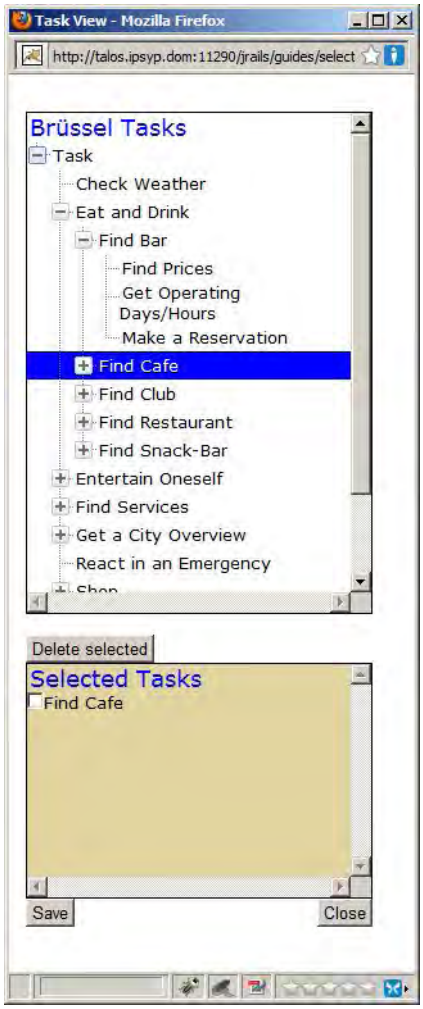


Figure 7: Selected task is added to the list of selected tasks (linked to a particular section)

By right clicking several tasks, the 'Selected Tasks' list at the bottom of the pop up window, gets filled with linked tasks. Every one of those tasks has a checkbox on its left. If the user wishes to 'unlink' one or more of those tasks, he has to check their checkboxes and press the 'Delete selected' button.

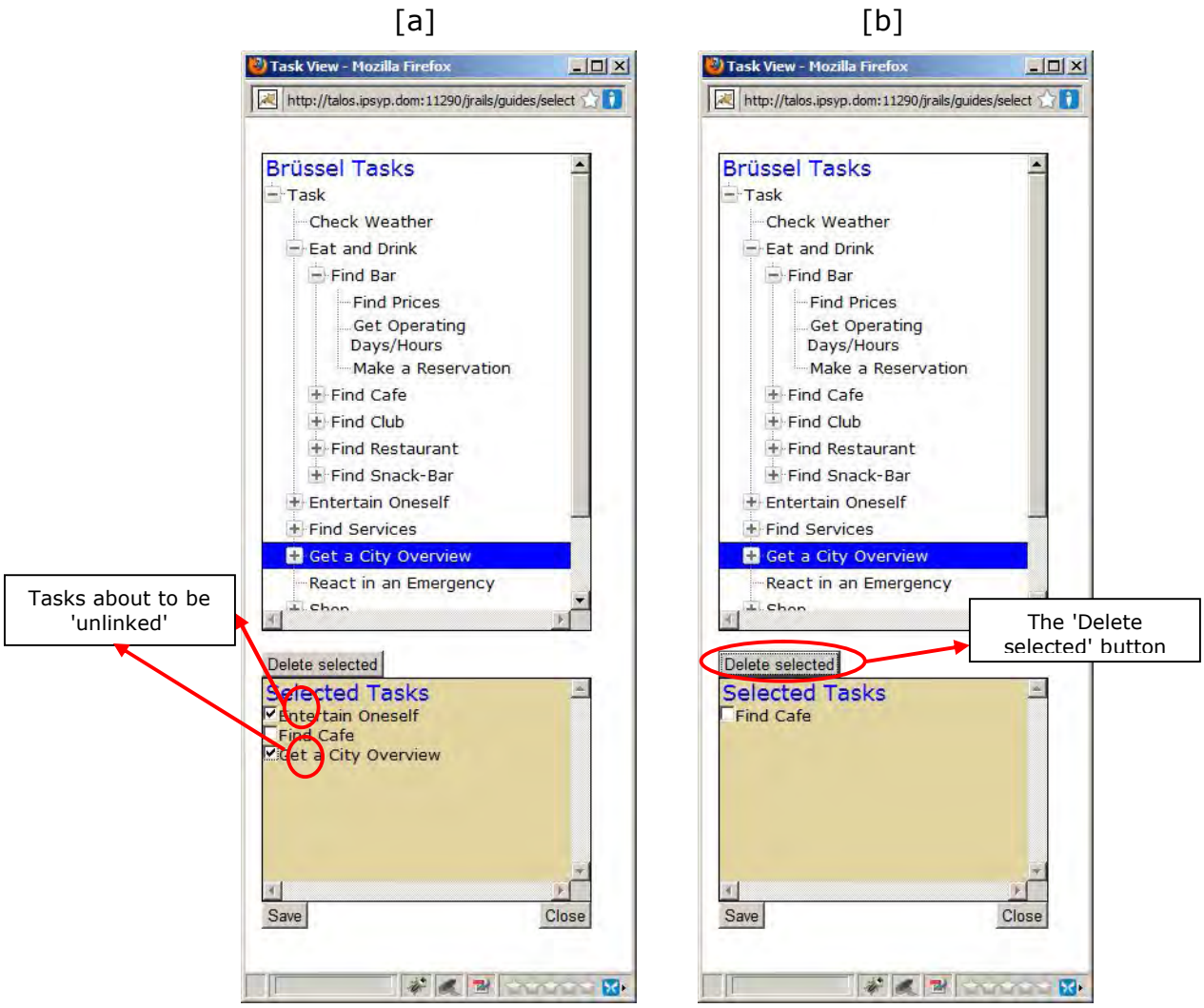


Figure 8: User selects the tasks he wishes to 'unlink' [a] and presses the 'Delete selected' button. Result is [b]

Once the user is done with all his changes (linking and unlinking tasks for a particular section), he should press 'Save' in order to record its actions to the backend SQL server 2008 database.

4.3 Table of contents browsing

Another useful subtool of the application is the table of contents browsing (Menu 'Table of Contents' on the navigation bar).

The Table of Contents for the Brussels travel guide appears:

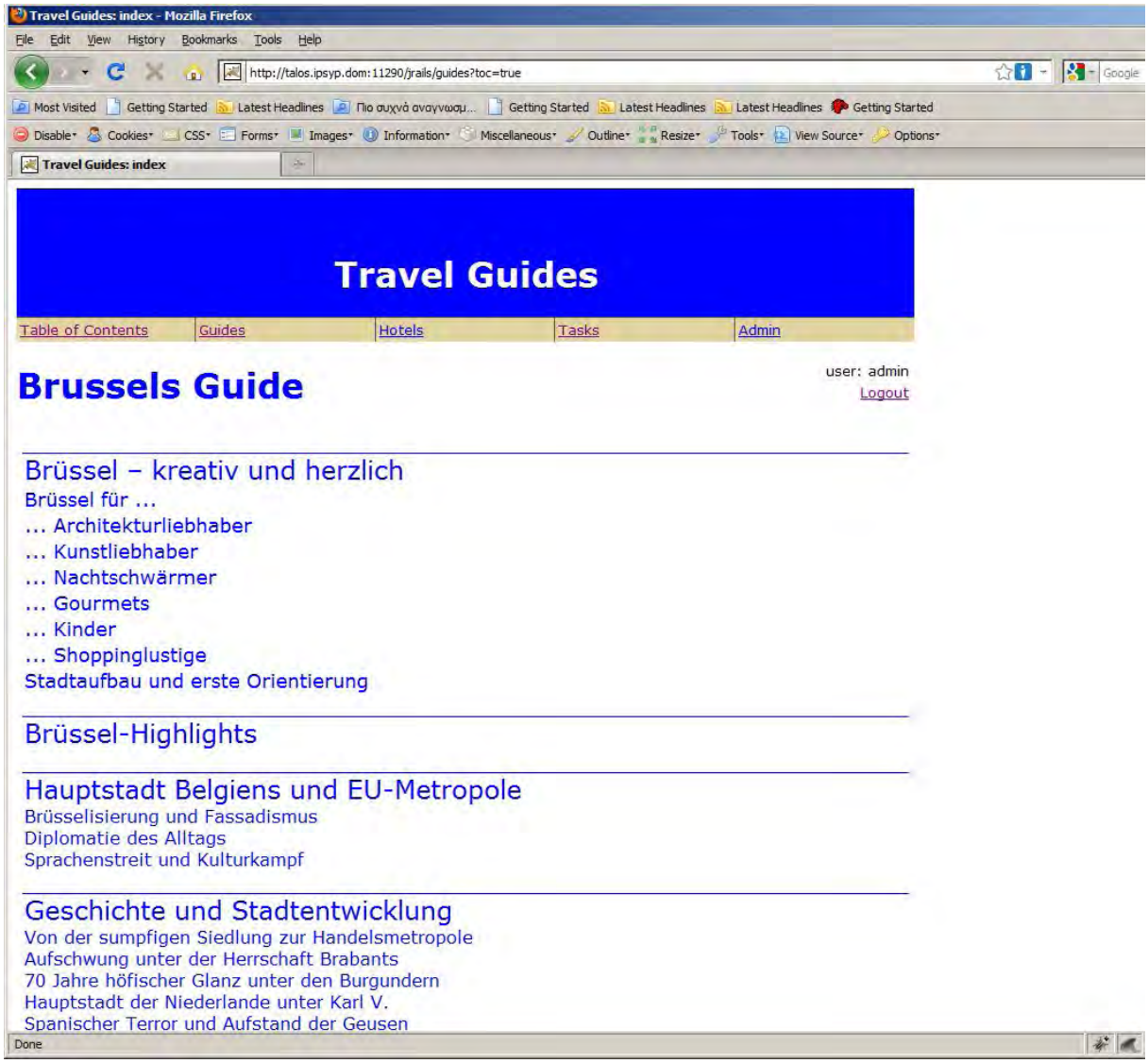


Figure 9: The Table of Contents view

Clicking on a Section title shows the content of the selected section.

Figure 10: Contents of a particular section

By clicking 'Back' the user returns to the Table of Contents view.

4.4 POI geocoding and web scraping

One of the most exciting features of CTI's Talos annotation tool is the ability to manually geocode and link information on the Web with existing POIs. As before, the POIs used are the MMV accommodation facilities (all 33435 entries – not just the Brussels ones). Those accommodation facilities are categorized by Land (usually the country), Region (usually the area of interest for a travel guide) and City (maybe a distinct city or a city subarea – neighbourhood).

By pressing the 'Hotels' menu option on the application's navigation bar, the user must then select the land, region and city of interest.

This process is done on three (3) steps described in the following images.

Figure 11: Land selection

Initially the user has to select from all the available 'Lands'. Once a 'Land' is clicked, only its 'Regions' are displayed:

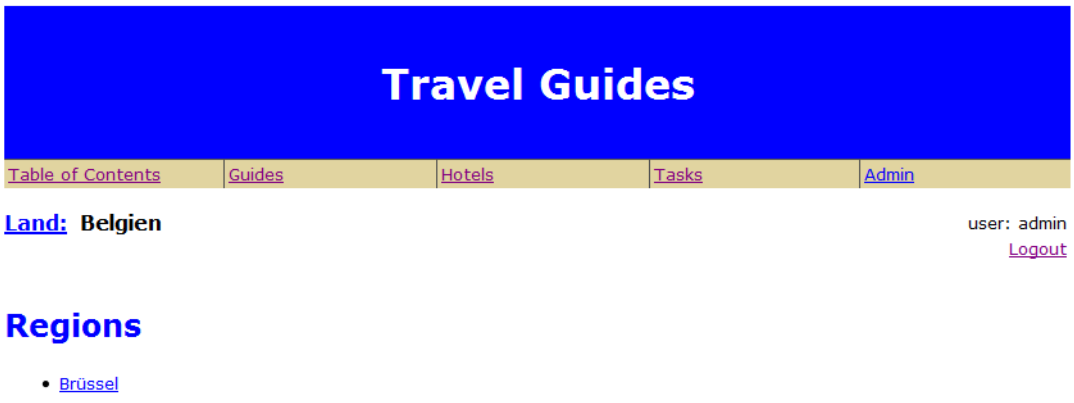


Figure 12: For Land = Belgien only the Region 'Brussels' is available

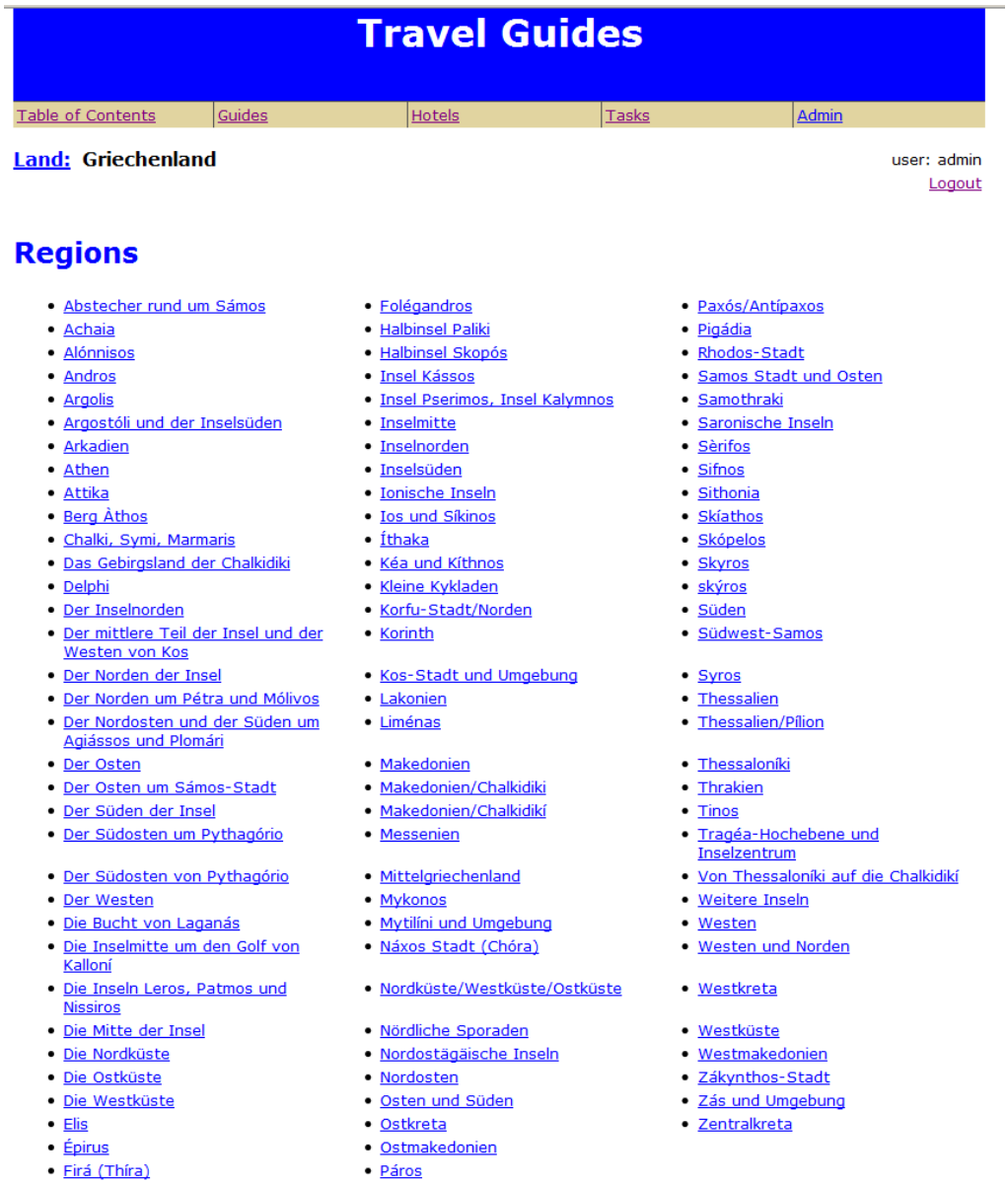


Figure 13: For Land = Griechenland many Regions are available

Once the user selects a Region as well, only its 'Cities' (its administrative subareas) are displayed:

Figure 14: Land: Belgien, Region: Brussel

Please keep in mind that if the user makes a wrong choice, he can switch back to any level (Land selection, Region selection, City selection) by clicking on the links 'Land', 'Region' or 'City' accordingly.

Once the user selects a city as well, then the user may select if he prefers to perform geocoding or web scraping operations for the hotels of the selected Land, Region and City by pressing the appropriate button that appears on the bottom of the page.

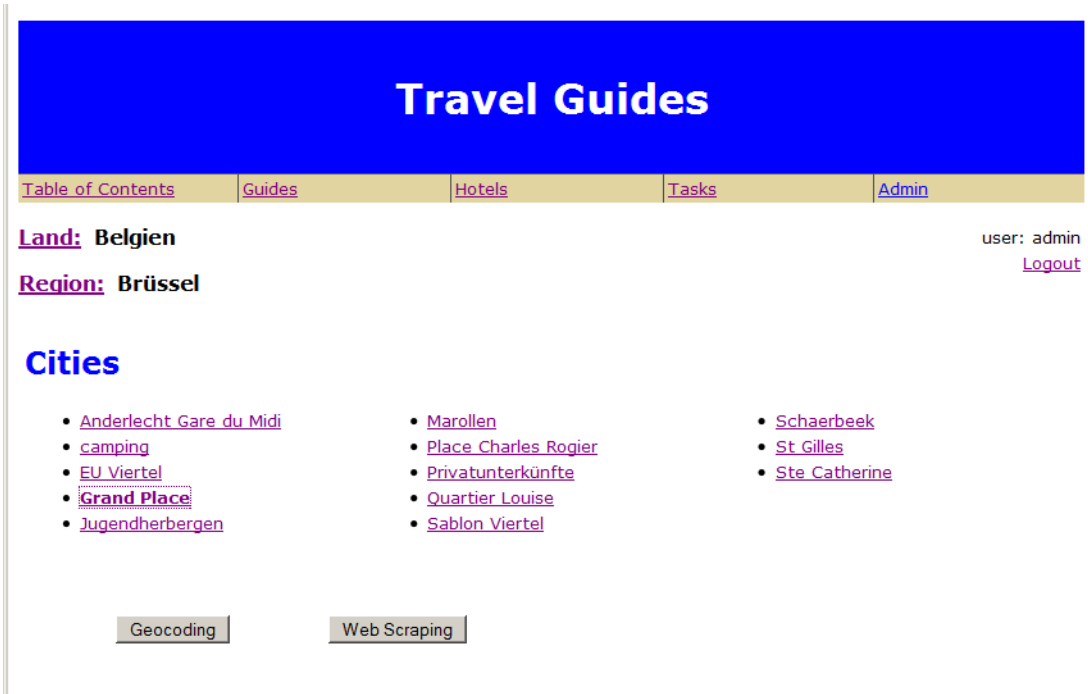


Figure 15: The user selected 'Grand Place' so now he must choose between 'Geocoding' or 'Web Scraping'

4.4.1 Web scraping

Once the user presses the 'Web Scraping' button, he is introduced to a data grid containing all MMV accommodation facilities that exist in the selected Land, Region and City. This data grid is fully sortable by Name and Address, by clicking on those column headers.

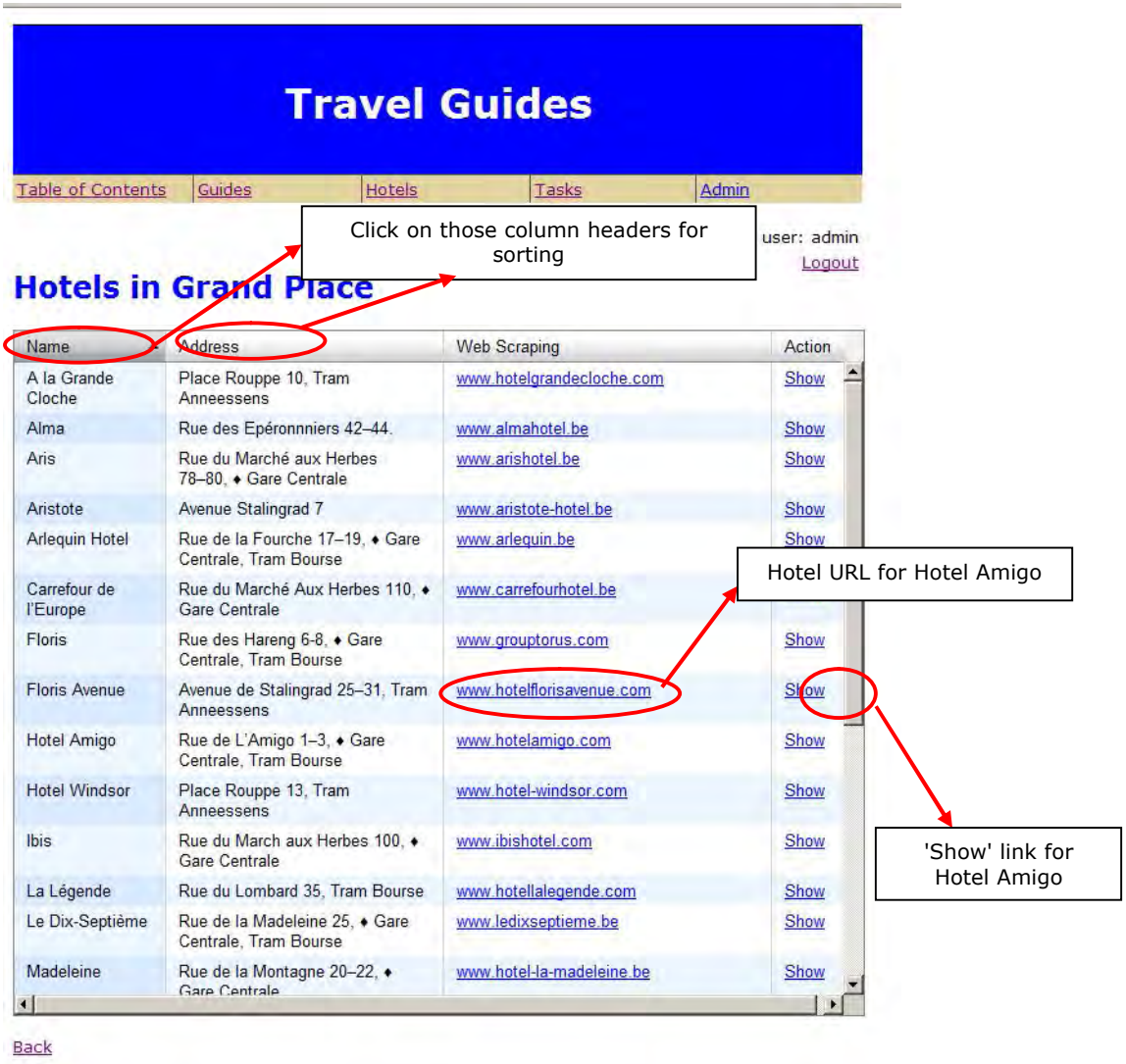


Figure 16: Accommodation facilities for Belgien, Brussels, Grand Place

As was noted earlier, Web scraping requires the use of a custom Firefox extension, developed by CTI. A Firefox extension is a set of JavaScript and XML files, packaged as a zip file with the .xpi extension. In order to install this xpi file, one has to right click on it and then select 'Open with' and select 'Mozilla Firefox.

Let us say that the user wants to add some web content for some hotel (Hotel Amigo for example). Initially, he must click on the hotel URL.

A new browser window opens. Web scraping is only possible within this new browser window. The user initially is taken to the hotel URL (provided by MMV) but he may navigate to any other site (we will talk about Web scraping limitations later on). He must then select the portion of the page, he wishes to 'collect', right click and select 'Send to database' (a new Firefox context menu added through our Firefox extension. This process is visualized in the following image.

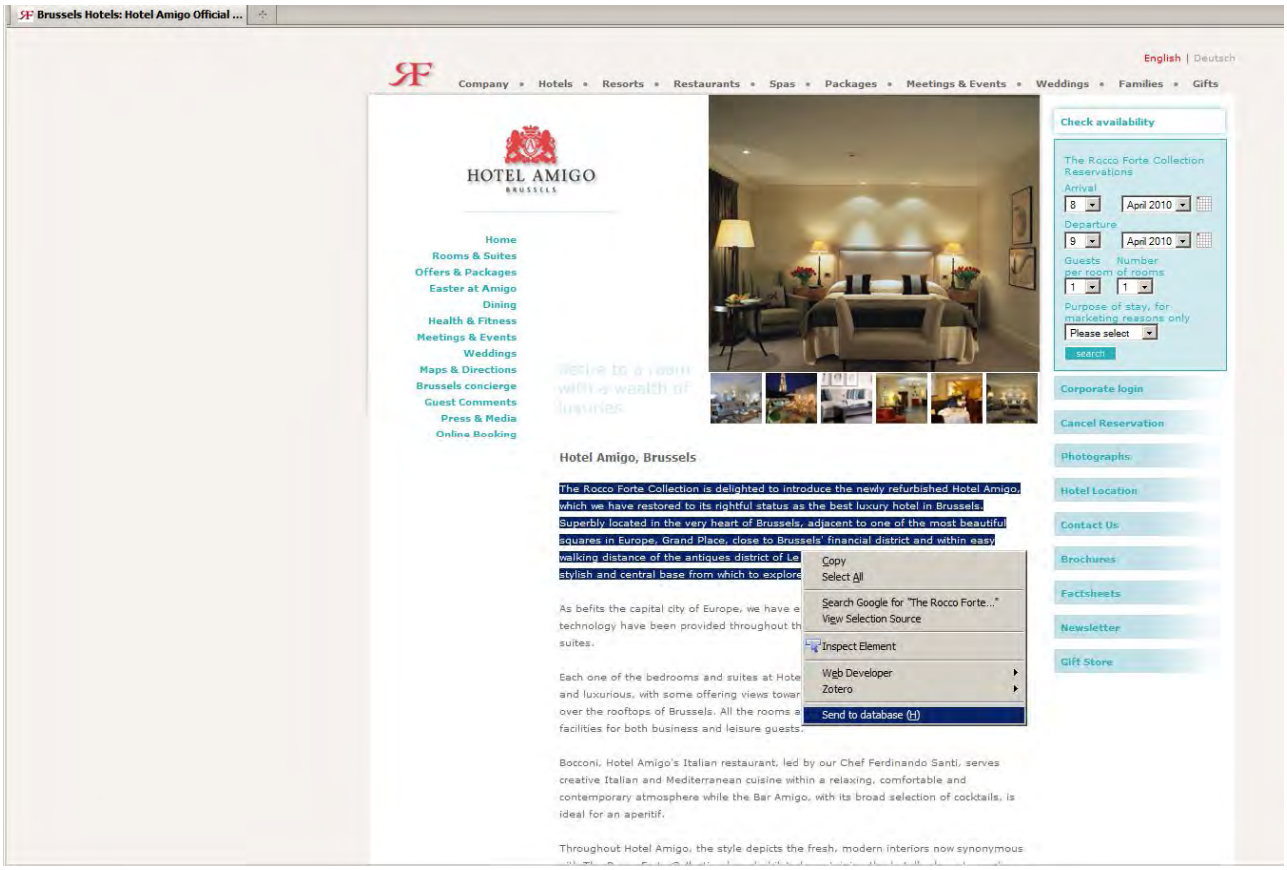


Figure 17: Selecting the desired portion of the page and 'Send to database'

Once the ''Send to database ' is selected the new browser window is closed and we return to our previous page. In order to see all Web scraped content for 'Hotel Amigo' the use should click on its 'Show' link. A new pop up window appears:

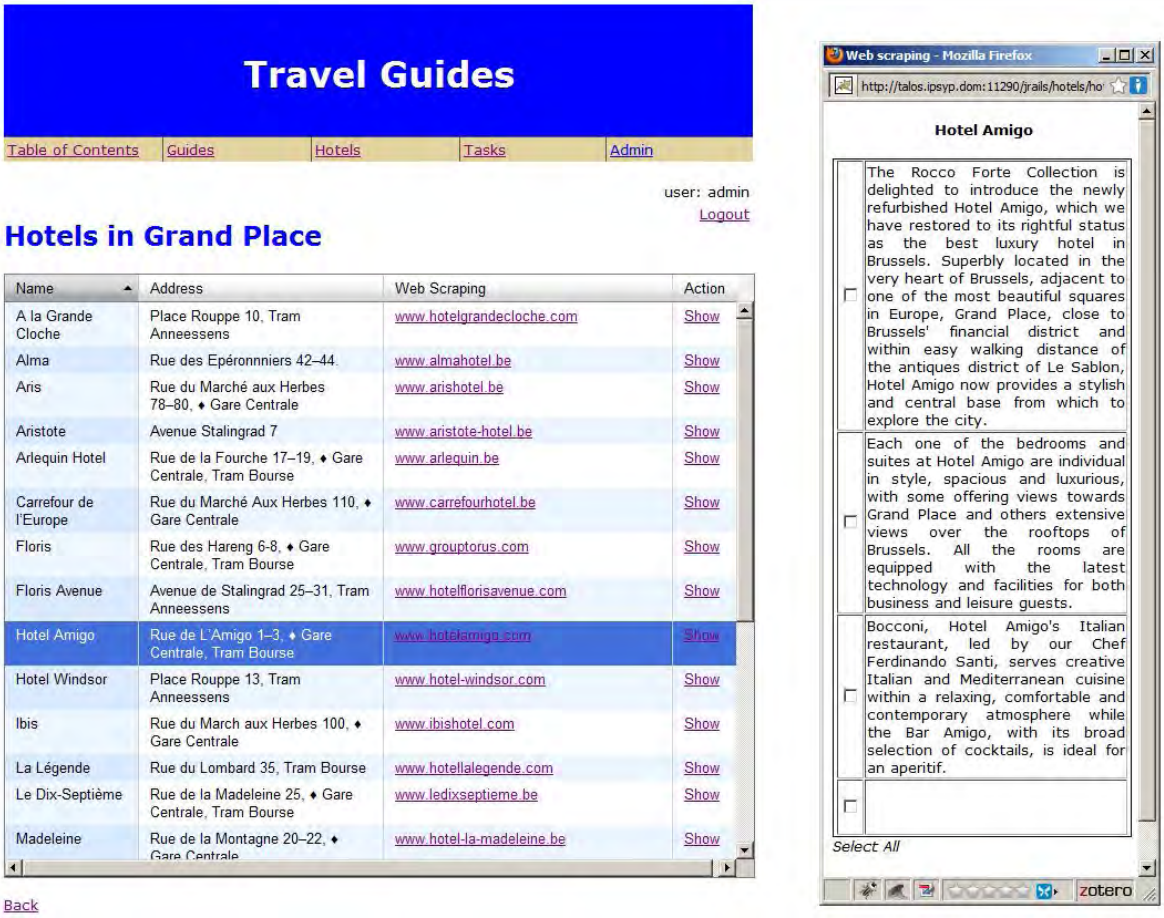


Figure 18: Web scraped content for 'Hotel Amigo'

Each of those web scraped entries has a checkbox on its left. If the user wishes to unlink some of those entries, he must check the corresponding checkboxes and then press the 'Delete selected' button at the bottom of the pop up window. He must then choose 'OK' on the confirmation box. This process is shown below:

[a] [b] [c]

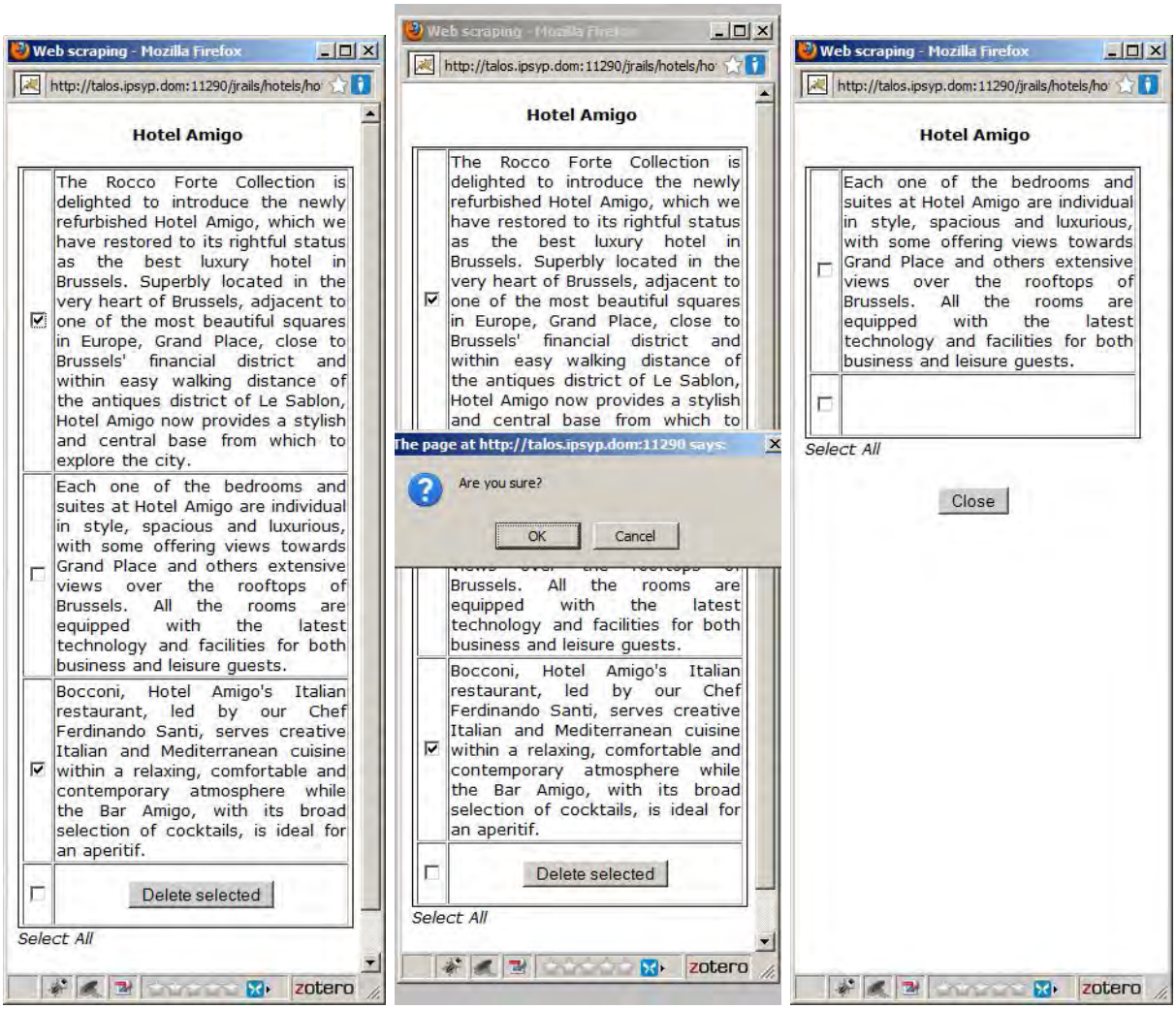


Figure 19: Removing Web scraped content for 'Hotel Amigo', steps [a], [b] and [c]

This way the user may easily add and remove Web scraped content for all available POI entries. The important thing is, that in the SQL server 2008 database it is not just the text of the outside web page that is stored, but also the external URL and the full hierarchy of the HTML node selected in the DOM hierarchy of the page. Therefore if the text of the paragraph changes, then it is possible to automatically update the corresponding entry on the database.\

4.4.1.1 Web scraping limitations

It is obvious that this Web scraping process can only apply to normal HTML text. If the text that the user wishes to add is embedded inside an Adobe's Flash movie, then the Firefox extension cannot work. There are also ways that by editing its 'robots.txt' file (http://en.wikipedia.org/wiki/Robots_exclusion_standard), an external web site can block access to its contents for our Firefox extension. On the other hand, this is not very likely to happen and our Web scraping approach should probably work in most cases and it is already tested with popular sites like Wikipedia.

4.4.2 Geocoding

Geocoding is divided into two separate components. A Java command line application used for batch geocoding and the online version incorporated on our TALOS annotation web application for manually rectifying the geocode results of the command line tool.

So the question that arises is why do we need 2 separate (but complimentary) tools for geocoding? The question is actually really simple, if we consider the total bulk of POI entries we have to geocode. MMV provided more than 33000 accommodation facilities that needed to be geocoded. ChefMoz includes more than 280000 entries. On the other hand, CTI also suggested other sources (Muselius, Booking.com, DBpedia) for POI extraction. When we have to deal with such huge load, it is obvious that constructing an online tool that geocodes one entry at a time (either manually or automatically) will not be very useful, since it will require enormous human labour and work hours. Therefore, CTI thought that this effort should be divided on two parts:

- A Java command line application) that bulk reads entries from the SQL server 2008 database and stores the geocoding results in the same database as well. This command line tool is mainly targeted towards MMV developers, so it can be tweaked in compliance to the format of the data stored on the database. Therefore it may give a huge number results fast, strictly based on automatic geocoding services, with no human interference on the results. This lack of human interference of course is open to errors and that is why it is crucial to also provide another tool for manually rectifying those results.
- An online tool that gives the user the possibility to manually and easily rectify the results provided by the Java offline tool or even geocode entries for which the offline tool gave no results at all. The online tool is mainly targeted towards travel guide authors (MMV authors) that know nothing about databases but know exactly where the POIs are on a map and therefore it is very easy for them to move a POI marker on the correct position.

Since, the online geocoding tool is an internal part of the TALOS annotation tool, we will initially describe this first. As said earlier, before using the online tools, POIs have to pre-geocoded using the Java application. CTI geocoded all 33435 MMV accommodation facilities entries (that was a good test case for the Java command line tool) and the Java tool geocoded 20806 of those (62%). From those statistics is also obvious that a batch geocoding tool is not sufficient and a manual complimentary tool is also needed.

4.4.2.1 Geocoding subtool of the TALOS annotation web application

As described earlier, once the user selects a Land a Region and a City he is presented with 2 choices: Either geocode or add web scraping content for the MMV accommodation facilities that exist on the selected Land, Region and City. Once he selects 'Geocoding' he sees the following screen:

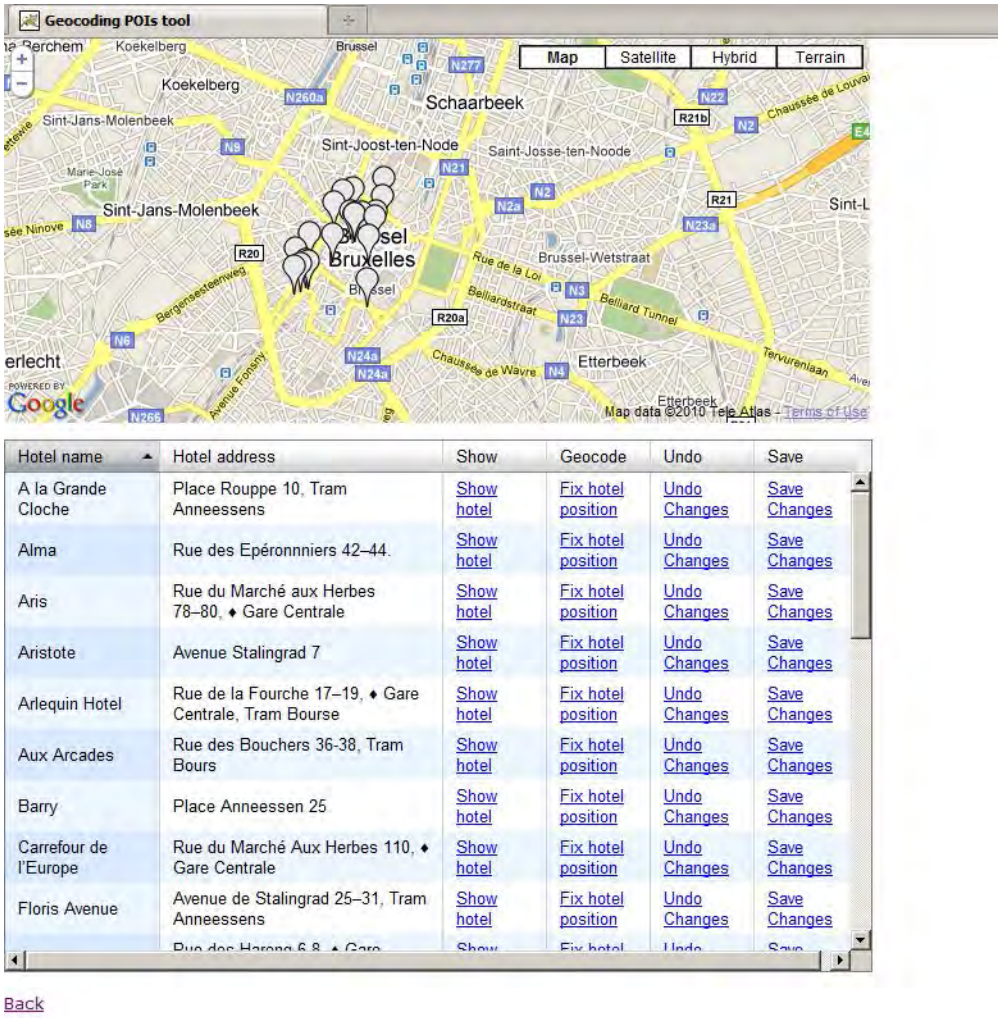


Figure 20: The Geocoding Google Maps interface

The list of all available accommodation facilities for the selected city are shown in a data grid, sortable by Hotel Name and Hotel address (the user has to click on those column headers for sorting) on the bottom of the web page. Showing the POI's address was necessary, so that the user knows where the hotel actually is before altering its location on the map.

The position of the accommodation facilities (results returned by the Java tool) are shown as markers on a Google Maps interface. Google Maps was chosen for its simplicity of use but the application could easily be converted to use Open Street Maps, if the final usage of the product contradicts the Google Maps terms of use (right now we are still at a development phase, so there is no such issue). It uses the latest version of Google Maps API (v3) in order to include its latest features. All the well known Google Maps features are there, including the ability to zoom in and zoom out on the draggable Map.

Since the POIs are in nearby locations, initially all POI markers are not draggable, so that the user may not accidentally move one or more markers. Only one marker may be draggable at a time, in order to avoid confusion and errors.

If the user clicks on a marker or selects the 'Show hotel' link for a particular hotel, the marker zooms in to this hotel location and shows the hotel information. That way:

- The user may see what marker belongs to which hotel
- If the hotel location is correct

The current POI marker is shown on purple, so it can be easily visible.

Hotel name	Hotel address	Show	Geocode	Undo	Save
A la Grande Cloche	Place Rouppe 10, Tram Anneessens	Show hotel	Fix hotel position	Undo Changes	Save Changes
Alma	Rue des Epéronnniers 42–44	Show hotel	Fix hotel position	Undo Changes	Save Changes
Aris	Rue du Marché aux Herbes 78–80, ♦ Gare Centrale	Show hotel	Fix hotel position	Undo Changes	Save Changes
Aristote	Avenue Stalingrad 7	Show hotel	Fix hotel position	Undo Changes	Save Changes
Arlequin Hotel	Rue de la Fourche 17–19, ♦ Gare Centrale, Tram Bourse	Show hotel	Fix hotel position	Undo Changes	Save Changes
Aux Arcades	Rue des Bouchers 36-38, Tram Bours	Show hotel	Fix hotel position	Undo Changes	Save Changes
Barry	Place Anneessen 25	Show hotel	Fix hotel position	Undo Changes	Save Changes
Carrefour de l'Europe	Rue du Marché Aux Herbes 110, ♦ Gare Centrale	Show hotel	Fix hotel position	Undo Changes	Save Changes
Floris Avenue	Avenue de Stalingrad 25–31, Tram Anneessens	Show hotel	Fix hotel position	Undo Changes	Save Changes

Back

Figure 21: Location of hotel 'Alma' shown on the map

In order for the user to be able to move the marker for a POI, he must click on the 'Fix hotel position' link for this particular POI. Once he does that, this POI's marker is draggable, so it can be placed anywhere on the map.

Hotel name	Hotel address	Show	Geocode	Undo	Save
A la Grande Cloche	Place Rouppe 10, Tram Anneessens	Show hotel	Fix hotel position	Undo Changes	Save Changes
Alma	Rue des Epéronnniers 42–44.	Show hotel	Fix hotel position	Undo Changes	Save Changes
Aris	Rue du Marché aux Herbes 78–80, ♦ Gare Centrale	Show hotel	Fix hotel position	Undo Changes	Save Changes
Aristote	Avenue Stalingrad 7	Show hotel	Fix hotel position	Undo Changes	Save Changes
Arlequin Hotel	Rue de la Fourche 17–19, ♦ Gare Centrale, Tram Bourse	Show hotel	Fix hotel position	Undo Changes	Save Changes
Aux Arcades	Rue des Bouchers 36-38, Tram Bours	Show hotel	Fix hotel position	Undo Changes	Save Changes
Barry	Place Anneessen 25	Show hotel	Fix hotel position	Undo Changes	Save Changes
Carrefour de l'Europe	Rue du Marché Aux Herbes 110, ♦ Gare Centrale	Show hotel	Fix hotel position	Undo Changes	Save Changes
Floris Avenue	Avenue de Stalingrad 25–31, Tram Anneessens	Show hotel	Fix hotel position	Undo Changes	Save Changes

Back

Figure 22: Hotel 'Alma' marker was moved by the user

If the user wishes to save the POI's new location to the backend SQL server 2008 database, he will have to click on the 'Save changes' link for this POI. If he wishes to retain the POI's original location (and therefore undo his changes) he should click on the 'Undo changes' link for this POI. Both of those links are 'dead' if the user has not changed this POI location on the map.

When one or more markers are moved on the map (but are not stored on the database), they are shown in yellow. This visual reminder is useful, in order for the user to:

- Remember to save or undo his changes
- Easily see for what POI he has made changes (he just has to click on the yellow marker to see its name)

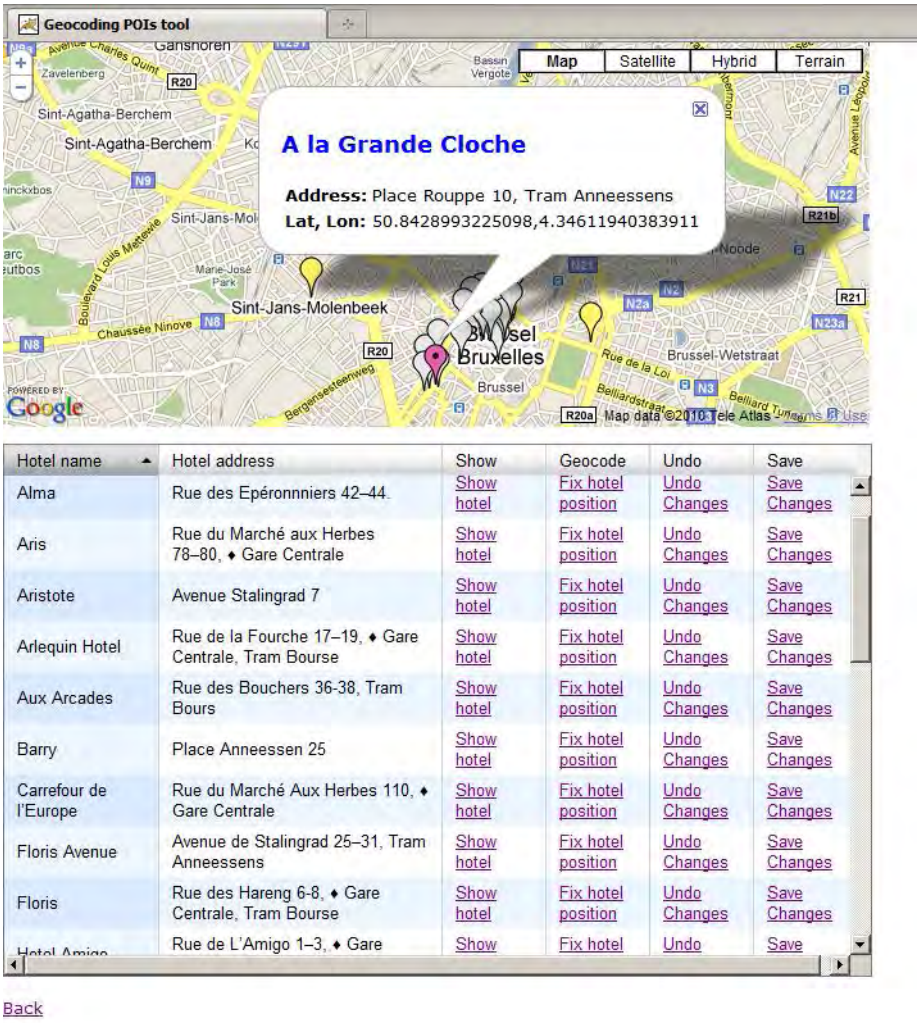
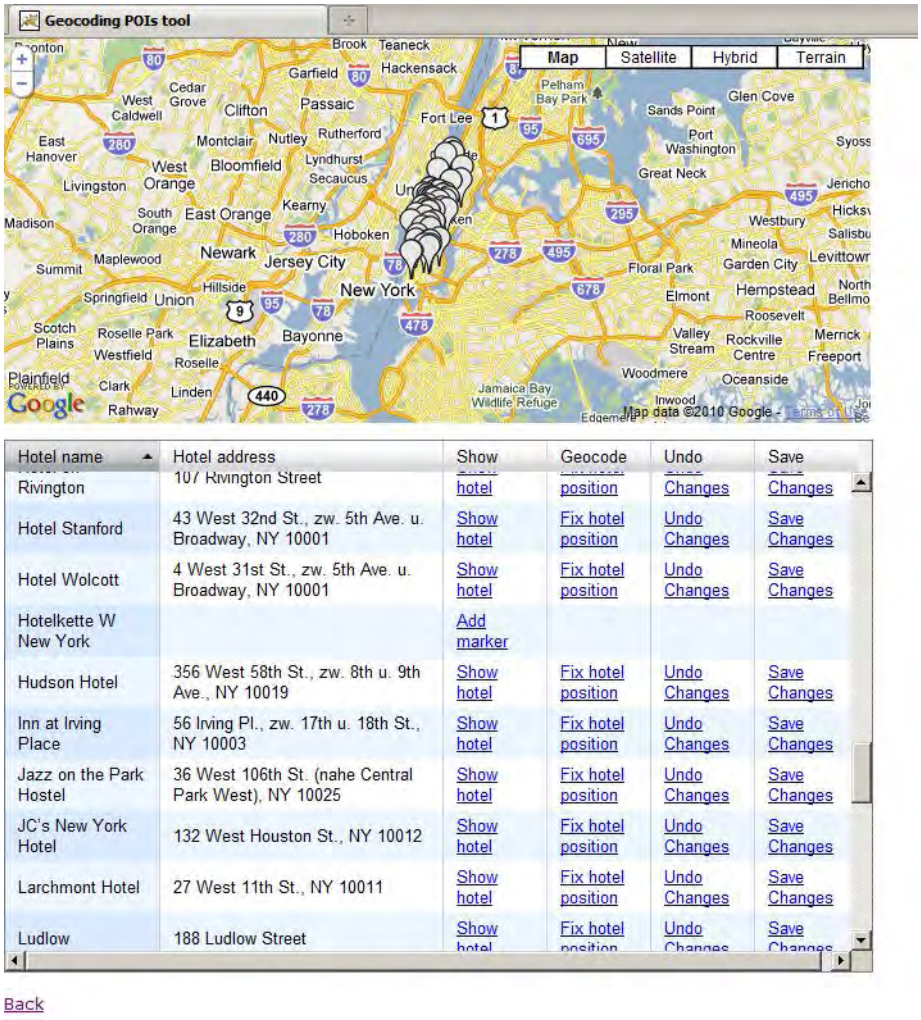


Figure 23: Two POIs' location has changed (yellow markers) and need to be stored on the database

In case that the offline tool has not found a result for a suggested POI, through the online tool the user may add a marker on the Map for the suggested POI, by clicking the 'Add marker' link. Here is an example from New York:



Hotel name	Hotel address	Show	Geocode	Undo	Save
Rivington	107 Rivington Street	hotel	position	Changes	Changes
Hotel Stanford	43 West 32nd St., zw. 5th Ave. u. Broadway, NY 10001	Show hotel	Fix hotel position	Undo Changes	Save Changes
Hotel Wolcott	4 West 31st St., zw. 5th Ave. u. Broadway, NY 10001	Show hotel	Fix hotel position	Undo Changes	Save Changes
Hotelkette W New York		Add marker			
Hudson Hotel	356 West 58th St., zw. 8th u. 9th Ave., NY 10019	Show hotel	Fix hotel position	Undo Changes	Save Changes
Inn at Irving Place	56 Irving Pl., zw. 17th u. 18th St., NY 10003	Show hotel	Fix hotel position	Undo Changes	Save Changes
Jazz on the Park Hostel	36 West 106th St. (nahe Central Park West), NY 10025	Show hotel	Fix hotel position	Undo Changes	Save Changes
JC's New York Hotel	132 West Houston St., NY 10012	Show hotel	Fix hotel position	Undo Changes	Save Changes
Larchmont Hotel	27 West 11th St., NY 10011	Show hotel	Fix hotel position	Undo Changes	Save Changes
Ludlow	188 Ludlow Street	Show hotel	Fix hotel position	Undo Changes	Save Changes

Back

Figure 24: Hotelkette W New York does not have not a marker. So the user may add a marker by pressing the 'Add marker' link

As you see, the geocoding component of our Talos annotation web application is very easy to use. Significant effort has also been done in order to be as error proof as possible.

4.4.2.2 Java Geocoding command line application for batch geocoding

The Geocoding command line application may use one of the following free online geocoding services:

The Google Geocoding Web Service

The OpenStreetMap Name Finder

Both of those services have their advantages. In short:

- Google's service gives better results when the city's data is available, since it takes into account the street name and the street number. The OSM service does not take into account the street number (eg the Rue del l'Amigo 5 would return the same result as Rue del l'Amigo 10, which will be a point in the middle of Rue del l'Amigo). In that sense if the street is several kms long, OSM results are inaccurate
- OSM namefinder has no limitations on its terms of use. Google service requires that the result of the geocoding service, should exclusively be displayed on a Google Maps (a term fulfilled by TALOS up till now). Also, Google does not allow more than 2,500 geocode requests in a 24 hour period from a single IP address or geocode requests from a single IP address at too fast a rate. The second part (geocode requests from a single IP address at too fast a rate) is taken care of, since the application waits for 1 second before executing a new request. Therefore we have a maximum of 1 geocoding request per second. The 2500 limit just imposes a limitation on the number of days it would require to perform batch geocoding of addresses that exceed this limit
- Since, OSM is community driven, it is ever expanding even in areas with no actual commercial interest (therefore in some areas OSM gives results and Google does not)

As a conclusion, Google results are better in most cases but its terms are restrictive, so therefore, the application may need adjustments (or a special license) in order to be used on a commercial environment.

The Java application originally worked with the older (v2) of the Google geocoding service, but it has been rewritten to use the latest version (v3)

In order for the Java Geocoding command line application to work:

- The user must have JVM (Java Virtual Machine) 1.6 installed.
- All addresses that need to be geocoded must reside on one SQL server 2008 db. Each such address is uniquely identified by a BIGINT id field.
- Results of the geocoding process are stored on 2 separate tables on the same database. In order to create those 2 tables in the correct format, the user may use the following SQL query:

```
/* Start of query */
USE [sample_geocoding_databa] /* This is the name of your database
*/
GO

/* The names of the 2 tables are not important. You can change them
anyway */
CREATE TABLE [dbo].[hotels_exp_geocode](
      [autoinc] [bigint] IDENTITY(1,1) NOT NULL,
      [id] [bigint] NULL,
      [longitude] [float] NULL,
      [latitude] [float] NULL,
      [google_address] [nvarchar](max) NULL,
      [types] [nvarchar](max) NULL,
      [db_table] [nvarchar](max) NULL,
 CONSTRAINT [PK_hotels_exp_geocode] PRIMARY KEY CLUSTERED
(
      [autoinc] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[hotels_exp_geocode_long](
      [autoinc] [bigint] IDENTITY(1,1) NOT NULL,
      [id] [bigint] NULL,
      [short_name] [nvarchar](max) NULL,
      [long_name] [nvarchar](max) NULL,
      [types] [nvarchar](max) NULL,
      [db_table] [nvarchar](max) NULL,
 CONSTRAINT [PK_hotels_exp_geocode2] PRIMARY KEY CLUSTERED
(
      [autoinc] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
/* End of query */
```

In order for the user to tweak the application to his needs, he must edit a properties file, provided with the application. Here the user may change his SQL server, the port the SQL server listens on, the db user and password and other small details. Tweaking the properties file is very easy to do, so the average user will not have any serious problems. After all, this tool is targeted towards developers and not travel guide authors.

A sample properties file is shown below:

```
# Your SQL server
db_server = localhost
# The port SQL server listens to. Default is 1433
db_port = 1433
# The user that connects to the SQL server. It is not required if
you use Windows authentication
db_user = postgres
# The password for the SQL server user. It is not required if you
use Windows authentication
db_user_pass = my_talos
# The database we will use
db = sample_geocoding_databa
# The db table that includes the addresses we want to geocode.
db_table_query= brussels_hotels
# Results of the geocoding process are stored on 2 tables. Here we
define the name of the first
db_table_insert= hotels_exp_geocode
# Results of the geocoding process are stored on 2 tables. Here we
define the name of the second
db_table_insert2= hotels_exp_geocode_long
# The query we use for querying the db table that includes the
addresses. Query must return only 2 fields. 'id' and 'address'. 2
sample queries are included below
db_select_query = SELECT TOP 1000
id,SUBSTRING(Address,0,CHARINDEX(',',Address)) +  ', ' +'Brussels'
AS address FROM brussels_hotels WHERE id NOT IN (SELECT id FROM
hotels_exp_geocode) AND CHARINDEX(',',Address)>0
# The language that Google will return results. Sample languages
are: En for English, DE for German
google_language = DE
```

4.5 Administration of users

If the user has logged in as admin (and has the necessary permissions), the new menu 'Admin' is added to the application. From this menu, the administrator of the web application may add new users, who may have access to the application.

Figure 25: Administration screen

In order to add a new user, the administrator has to click the 'New user' link. He then has to complete the following fields:

- Username
- Name
- Email
- Password

Figure 26: Adding a new user

In case the administrator forgets to complete some fields correctly, he gets detailed information about what he should correct. Therefore, it is almost impossible to add invalid entries for new users of the web application.

Figure 27: Adding a new user validation errors

So, it is obvious that the administration component of TALOS annotation tool follows the same philosophy of the rest of the application. It provides a simple and elegant user interface, that tries to protects the user from accidental errors. After all, the TALOS annotation tool is targeted towards MMV authors and not developers and therefore had to be as easy to use as possible

5 Conclusion

Creating the TALOS annotation tool was a very interesting and demanding challenge. The main obstacle was that there are too many diverse and complicated subtools and frameworks that had to embedded into a unified web application, that would still 'feel' easy to use for its user, despite its complicated nature. This effort required the use of many different frameworks and techniques in order to create a rich Web application that requires no installation and yet feels and offers all the advantages and functionalities of a desktop application.

An important factor that was initially underestimated in the creation of this tool was the preparation and pre-processing of all available data, in order to be fully incorporated on the application. Namely:

- Ingesting the contents of the travel guides (Microsoft Word format) inside the SQL server 2008 database
- Converting the DTP format of those previous entries into XHTML through the extensive use of stored procedures, SQL, XPath and XQuery
- Ingesting the POIs (accommodation facilities in more than 880 Excel files) into the SQL server 2008 database, through stored procedures
- Expanding the available POIs by custom Java applications extracting information from various web sources, like Booking.com, DBPedia, Muselius and ChefMoz
- Creating a fully tweakable and Java command line tool for geocoding addresses, using Google Geocoding service and OpenStreetMap Name Finder
- Using this previous tool for geocoding POIs that lack location coordinates (namely 33435 accommodation facilities provided by MMV and ChefMoz entries for Brussels)
- Automating all those processes, so they can be easily repeated for bigger bulk of documents and areas covered

After the preparation of data, which was a lengthy, time consuming and complicated process, we had to create the actual web application. Ruby On Rails (its JRuby port) was used for its simplicity, extensibility and overall features. The use of JRuby also facilitated the hosting of our application on any Java application server. We have chosen Tomcat because it is well tested, secure, easy to administer and relatively lightweight compared to other Java EE servers like Glassfish and JBoss. Therefore the combination of Java and Ruby on Rails provided the best of both worlds: the stability and security of Java Enterprise platform and simplicity of Ruby.

Although, Ruby on Rails features JavaScript libraries Prototype and Script.aculo.us for its Ajax purposes out of the box, we decided to incorporate YUI (another JavaScript library) for expanding even further the user experience and provide features, usually present on a desktop application, like sortable data grids, tasks presented as a tree view and context menus.

A custom Firefox extension (by the use of XUL and JavaScript) was developed for providing Web Scraping capabilities to our application. The Google Maps JavaScript API was used for building an interactive maps environment, for providing an straightforward and powerful environment for tweaking his geocoding needs.

Building a complicated web application like the TALOS annotation tool, requires the use of a high quality IDE. Therefore, Aptana Studio was chosen, for its Ruby on Rails support and its HTML, CSS and JavaScript editors. Best of all, it is free, open source and cross platform.

A full video tutorial for the use of TALOS web annotation tool is available at: http://talos.cti.gr/public/tool_video/talos.html