

DISTRIBUTED SYSTEMS
AN AUTOMATED AND PERSONAL TOURISTIC TOUR
APPLICATION USING GOOGLE MAPS API
FINAL DELIVERY

Quentin AUGRAIN
Florent MALLARD

June 2, 2015

Contents

1	Introduction	3
2	Application	4
2.1	Presentation	4
2.2	Technologies	7
3	Proof of testing	8
4	Conclusion	9

1 Introduction

In the context of the Distributed System Engineering project task, we designed a web-based application using the tools provided by the Google Maps API, that allows a tourist to personalize and optimize his route while visiting a city. The user may enter in the page the places he wants to visit, and the application will provide a optimized path to the several places the user set, with traveling indications. In this report, we will first present our application, its features and how it is used. Then, we will detail the technologies we used to create our application, with some illustrations. Finally, in the second part of our report, we will conclude on the performance testing of the application.

2 Application

2.1 Presentation

Here we will present the look of our application, and describe its fonctionnalities. In the FIGURE 1 we can see the welcoming screen of the guide. It has :

1. A title;
2. A data entering zone;
3. A welcoming map.

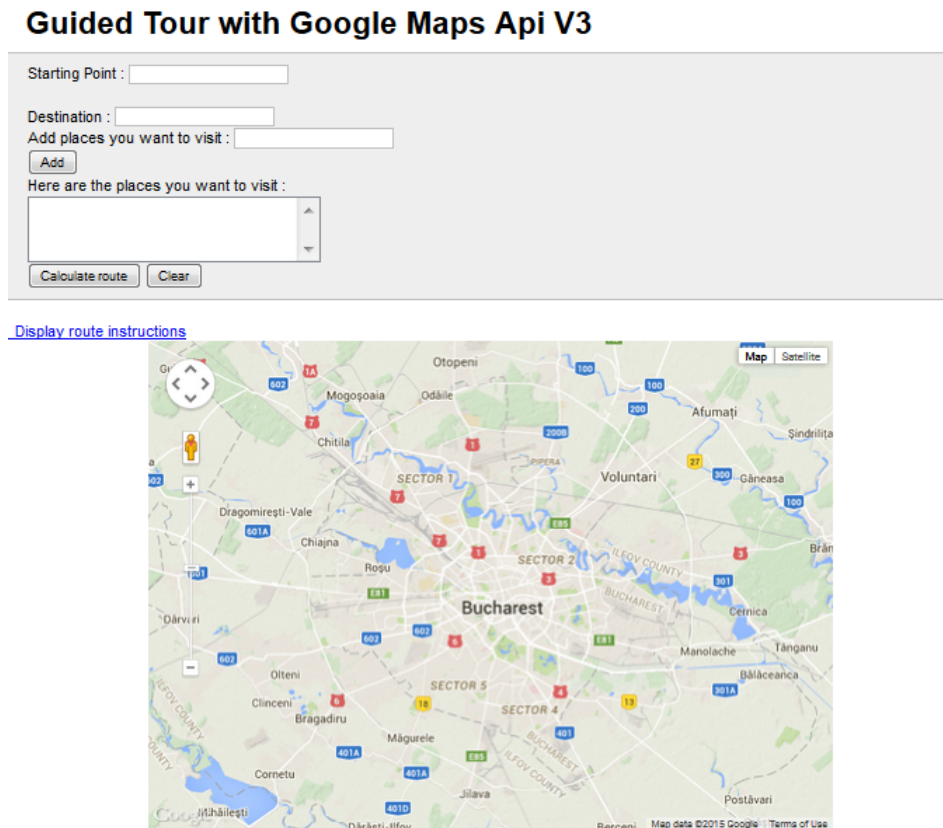


Figure 1: General look of the application.

The map provided by the Google Maps API is directly loaded when the page is open. It focuses by default on Bucharest.

If HTML5 geolocation is not enabled by default on your browser, or if you ask for each time a website wants to access it, you may see the pop-up in FIGURE 2, asking you to enable this feature.

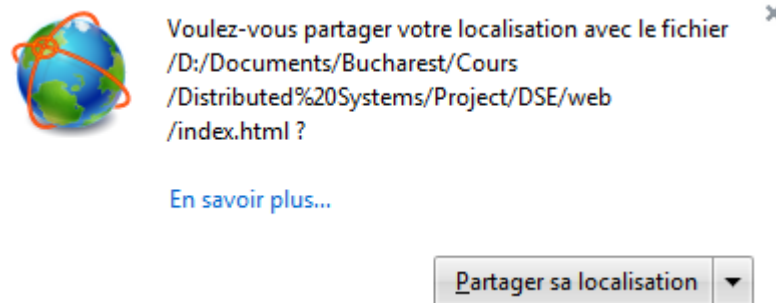


Figure 2: Pop-up asking for your location.

If you decide to enable it, the welcoming map will display your position, as shown in the FIGURE 3.

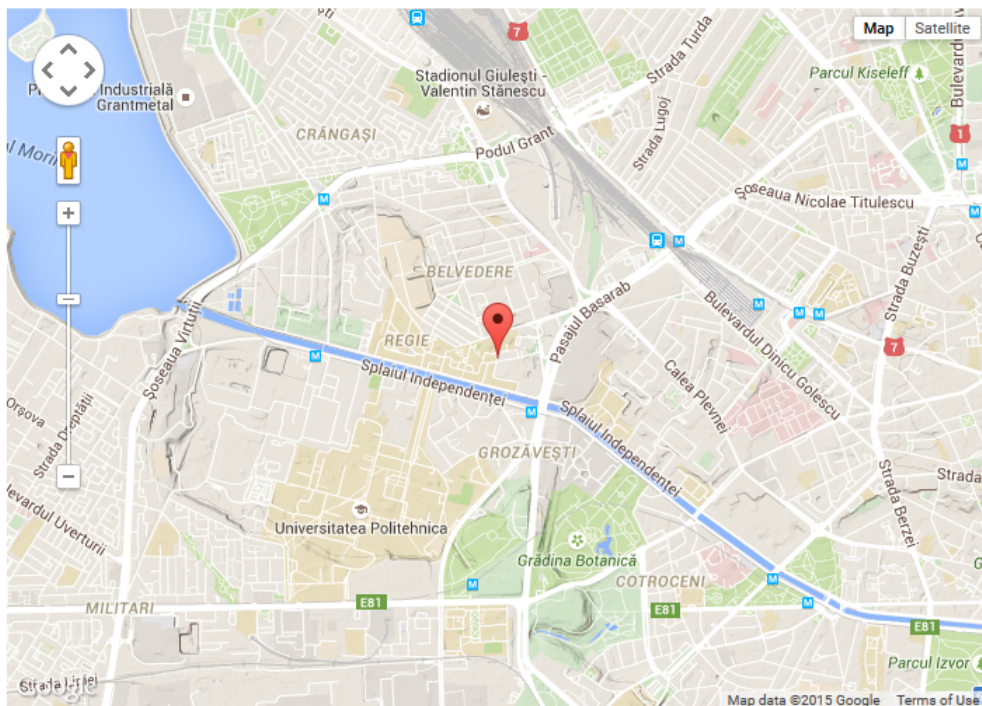


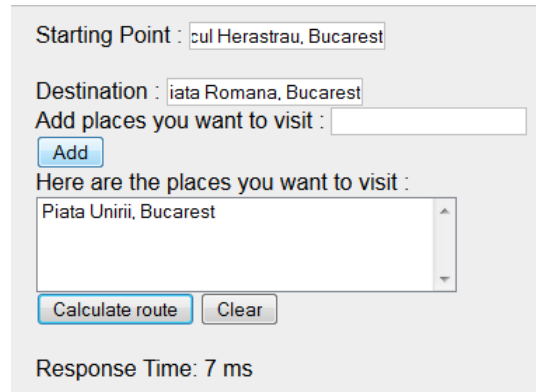
Figure 3: Map showing your position.

Once the welcoming page reached, you can start using the application. It is very simple, you should follow these steps :

1. Enter a starting point;
2. Adding an arrival point;
3. Add the intermediate points you want to reach by entering them one by one, and clicking the *Add* button.

4. Click the *Calculate button*

Note that the order in which you enter the intermediate points does not matter, the way will be optimized during the calculation phase. There is a *Clear* button to clear this list in case you made a mistake.



The screenshot shows a web application interface for route calculation. It has a 'Starting Point' field with the text 'cul Herastrau, Bucurest'. Below it is a 'Destination' field with the text 'Iata Romana, Bucurest'. There is an 'Add places you want to visit' section with an 'Add' button and a list box containing 'Piata Unirii, Bucurest'. At the bottom of this section are 'Calculate route' and 'Clear' buttons. Below the buttons, it says 'Response Time: 7 ms'.

Figure 4: How to use the application.

The route you should follow is displayed on the map after you clicked the *Calculate* button. Markers are also used to show the points you want to pass by.

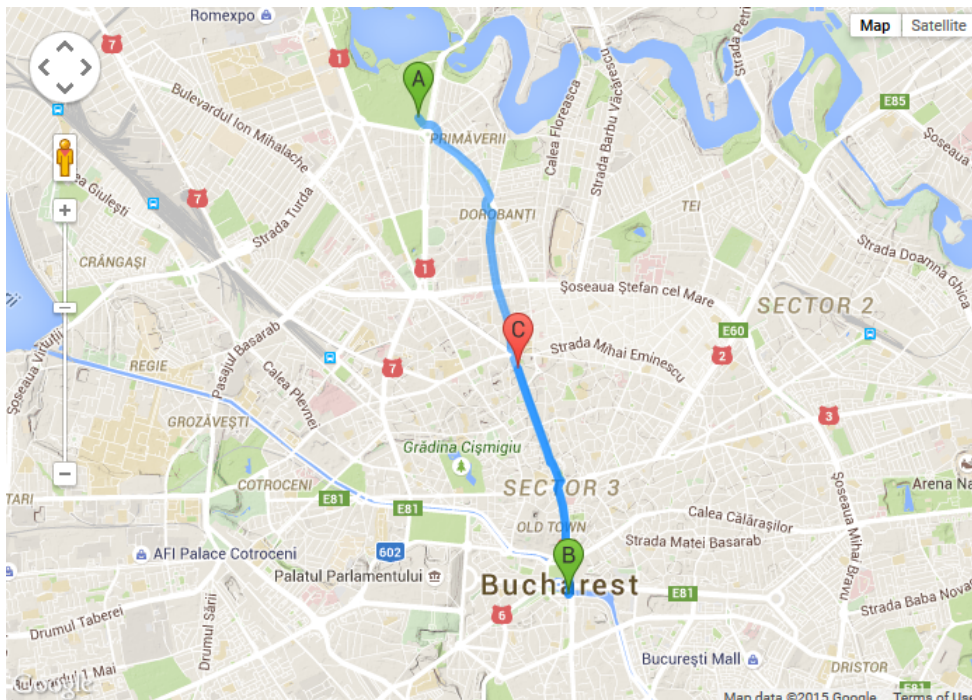


Figure 5: Map displaying your optimized way through the points wanted.

As you can see, the directions are not visible by default, since they can be very long and detailed. The *Display route instructions* link at the top of the map allows to display the route instructions for the tour. Since the application has been designed to visit a city, the directions will be for walking people, but you could eventually follow them with a bike. As you can see in FIGURE 6, the directions have the basic features of Google Maps, like :

1. The approximate time travel to the next step;
2. The distance between those steps;
3. The distance to make between each instruction.

[Display route instructions](#)















Walking directions are in beta. Use caution – This route may be missing sidewalks or pedestrian paths.	
 Parcul Herăstrău, București, Romania	
5.3 km - about 1 hour 6 mins	
1. Head northeast	5 m
 2. Turn right	89 m
 3. Turn left	0.2 km
 4. Turn left	0.8 km
 5. Turn right toward Strada Roma	39 m
 6. Continue straight	0.2 km
 7. Turn right onto Strada Roma	0.7 km
8. Continue onto Strada Căderea Bastiliei	0.6 km
 9. Turn left toward Bulevardul General Gheorghe Magheru/DN1/DN7/E60	85 m
10. Take the crosswalk	56 m
 11. Turn left onto Bulevardul General Gheorghe Magheru/DN1/DN7/E60 Continue to follow DN1/DN7/E60	1.1 km
 12. Turn left onto Strada Batiștei	12 m
13. Take the crosswalk	0.2 km
 14. Turn left toward Pasajul Nicolae Bălcescu	22 m
 15. Turn right onto Pasajul Nicolae Bălcescu Take the stairs	0.3 km
16. Take the crosswalk	0.3 km
 17. Slight left at Strada Șepcari	0.2 km
 18. Turn right	14 m

Figure 6: You can display the specific directions.

2.2 Technologies

Google Maps API and JavaScript We mostly used the tools provided by the Google Maps API in our application. The Google Maps API is composed of a wide array of specialized APIs, dedicated to web pages, web services, and mobile applications. While, on the second step of our project, we mainly focused on the mobile API, we decided to implement a web page rather than a complete mobile application. The API dedicated to this task in the Google Maps API is the JavaScript API v3, so most of the code we wrote was JavaScript. Also, the statement we made in our previous report regarding the UI controls and customization of the maps are entirely valid in our implementation. To illustrate a basic utilization of the API, we will detail the main steps to create a map in a web page:

- Load the API: the API is loaded in the line : `src="http://maps.googleapis.com/maps/api/js";`
- Set the properties of the map: the map is initialized at the beginning by this function:

```

7 initialize = function(){
8   var myOptions = {
9     zoom      : 14,
10    center    : pos,
11    mapTypeId : google.maps.MapTypeId.TERRAIN,
12    maxZoom   : 20
13  };
14  map        = new google.maps.Map(document.getElementById('map'), myOptions);
15  panel      = document.getElementById('panel');
16 }
17

```

Figure 7: Setting the map.

The Map object that represents our map in the API is created by the constructor `google.maps.Map(document.getElementById("map"), map)` that takes as parameters the container of the map in the HTML file, and the set of properties we set in the `initialize()` function. In our application, we also used the HTML5 geolocation integration in the API, and the calculation of directions. This calculation of the route and its drawing on the map is handled by this code:

```

86   var request = {
87     origin      : origin,
88     destination : destination,
89     waypoints   : waypts,
90     optimizeWaypoints: true,
91     travelMode  : google.maps.DirectionsTravelMode.WALKING
92   }
93
94   var directionsService = new google.maps.DirectionsService();
95   directionsService.route(request, function(response, status){
96     if(status == google.maps.DirectionsStatus.OK){
97       direction.setDirections(response);
98     }
99   })

```

Figure 8: Request to calculate the route, and draw it on the map.

In conclusion, all the tools regarding map generation, route calculation, time request calculation and directions computing are provided out-of-the-box by the Google Maps API in the form of JavaScript objects and functions.

3 Proof of testing

In order to test the performance of our application, we implemented a little feature that displays the response time of Google Maps servers to process our requests. We looked at this response time with a growing number of intermediate points to visit. FIGURE 9 shows the results of this little experiment.

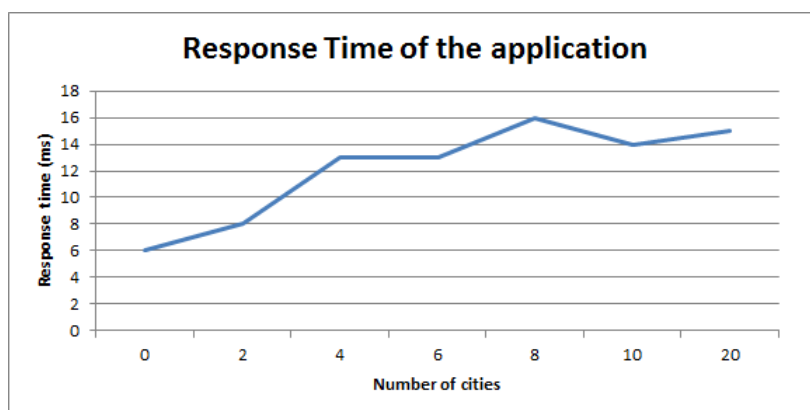


Figure 9: Response time according to the number of points to visit.

As you can see, the response time does not change much, going from 6 to 16ms, which is quick enough to assure a good user experience. We deduced from this experiment that the response time is not much influenced by the number of steps, allowing the user to organize a huge city tour.

4 Conclusion

Finally, we designed a web application that allows us to personalize a city tour using the Google Maps API and JavaScript language. This tour is organized and optimized, like a human would try to do. Its instructions and the map make the way clear and it's easy to follow the instructions to reach each place you want to visit.