


# Recommendation System for Sightseeing Tours

Ricardo Claudino Valadas, Instituto Superior Técnico de Lisboa, Lisboa, Portugal

Elizabeth Simão Carvalho, CIAC/UAb, University Aberta, Portugal

 <https://orcid.org/0000-0001-7036-5628>

## ABSTRACT

This research proposes a model of a recommendation system (RS) for tourist itineraries. The RS suggests tips of what to visit in a city, based on the available time, personal preferences, current geo-location, and the user's context awareness. These suggestions are calculated based on the treatment of collected data in real time by external application programming interfaces, through a list of points of interest located within a radius that can be reached by the user. Preliminary tests validated the model's goals and its potential in the tourism sector. The RS for tourist itineraries proposed is based on four essential points, in order to make the experience different and well as possible: end-user's personal tastes, the time available, end-user's current location, and context awareness. The performance tests that were carried out brought very positive results and showed that the RS presented a number of requisitions proportional to the server response times and algorithm. The functionality tests were quite positive, with percentages of experience of using the RS between 62.5% and 100%.

## KEYWORDS

Context Awareness, Google Places API, Mobile Application, Point-of-Interest, Recommendation Systems, Route Calculation Algorithms, Route Planners, Smart Tourism

## INTRODUCTION

Nowadays, the ever increasing importance role of the tourism in the global economy is evident. From 2020, it is expected that this sector will increase an average annual rate of about 4,3% (Holjevac, 2003). International tourism represents 7% of the world's exports in goods and services. Tourism has grown faster than world trade for the past five years (Vanhove, 2017).

However not everything is perfect in this sector. One of the biggest issues that tourists are facing when visiting new places is to decide what to do, visit and how to get a geographic point, according to his preferences. The working hours of the establishments around is also an important information, there is no reason to give a suggestion about visiting a certain place that is currently closed. This concept is called context awareness (CA) that basically means the detection of environment changes that may arise during the system usage (Pascoe, Ryan & Morse, 2000).

This paper proposes a recommendation system model to help the tourists to create their touristic itinerary within a certain time frame and according their preferences, present geo-location and user's context awareness. Section 2 presents an overview of the area, referring the most up-to-date projects similar in the area. Section 3 introduces the conceptual model of the recommendation system, showing its proposed architecture and algorithms to calculate and produce the tour itinerary. Section 4 presents

DOI: 10.4018/IJTHMDA.2020070104

This article, originally published under IGI Global's copyright on July 2, 2020 will proceed with publication as an Open Access article starting on January 20, 2021 in the gold Open Access journal, International Journal of Tourism and Hospitality Management in the Digital Age (converted to gold Open Access January 1, 2021), and will be distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

the prototype that was implemented and the tests that were conducted. Section 5 discusses how the model was evaluated. Finally, section 6 points out the main results and potential future work.

## BACKGROUND

PAT-Planner is a project that aims to personalize a route based on points of interest (POI) nearby, available time and budget of the user (Lu et al. 2016). Its concept aims to address the planning of a trip combining existing tourist packages and created by travel agencies, with tourist attractions included. It demonstrates the creation of a new RS that allows to generate a tourist itinerary in a zone based on the user's personal interest, available time and maximum budget. Several users' tests have been conducted and presented proving the consistency and functioning of the PAT-Planner.

Multi Request Route Planning (MRRP) is based on the Dijkstra algorithm (Dijkstra 1959). The authors of this article want to suggest to the users a walking route through a different types of places in urban environments (Lu et al. 2016). CA is not a concept mentioned by the authors, but this RS ends up using its essence (figure I). The problem is that is not taken advantage of, for example: The user knows which direction should take and which stores can go to, but at no point is it mentioned in the study that the RS takes into account stores that are currently open or closed.

Figure 1. Multi request route planning (Source: Lu et al. 2016)



Trip planning route main goal is to suggest tourist routes according to the working hours and the popular time at each place to visit (Chia et al. 2016). According to the coordinates of a certain point to visit (latitude and longitude), hours of operation and recommended time of these same sites, this RS for tourist itineraries suggests to the user what to visit on the island of Penang, Malaysia.

Xenia is the name of the recommendation system for tourist itineraries created by Korakakis, Mylonas & Spyrou (2016). It aims to receive the data that come from the social network Flickr, which inform the system about POI in the vicinity, also taking into account the context-of-awareness (CA) concept. Information on the activity of the user in this social network is considered as the point of the origin of the information.

Trip-Mine wants (Lu et al. 2011) to improve the recommendations in the tourism sector according to the user's location. The biggest challenge is to deal with the user's available time. The authors

studied and departed from the Travelling Salesman Problem (TSP) algorithm. This is a positive point because this algorithm is fully suited to performance problems in the search for the quickest way to go.

Touch Map (Ying et al. 2013) is a framework developed with two modules: 1) A cloud system dedicated to search for points of interest. 2) A cloud system to create a touristic route based on user inputs, like available time and budget limit. The ultimate goal of Touch Map is to customize a tour itinerary based on the two modules indicated, taking into account the performance of the RS. The problems to be solved were well specified by the authors of the article. The fact that recommending a well-designed new tool demonstrates the preparation that existed before they started the development. The authors started from Trip-Mine, a system they had previously made, and used it as the basis of the second module. This system also belongs to the present state of the art and is present in the previous point. Usage tests were performed at very different locations, both geographically as social, North America and Asia.

Personalized Sightseeing Planning System (PSiS) is a recommendation system to create and suggest touristic routes based on user parameters such as preferences, budget limit and available time (Anacleto et al. 2014). This RS has as its starting point the problem of algorithms TSP, which tries to find the shortest path in a given route if a list of cities to visit are considered, with the conditioning of visiting each of these cities only once throughout the route.

Google Trips (Google Trips, 2019) is (figure 2) a mobile application that has a lot of features for its users. One of the main features is the possibility to create custom touristic walking routes around the world, according to the user preferences. Despite being a very complete and widely used mobile application, it contains many features and confuses a user who is interested in just one. Taking into account the functionality of the tourist itineraries, it can only be completed after a high number of clicks and actions, which makes the process take longer.

Table 1 summarizes the strengths and weaknesses of each analyzed similar research.

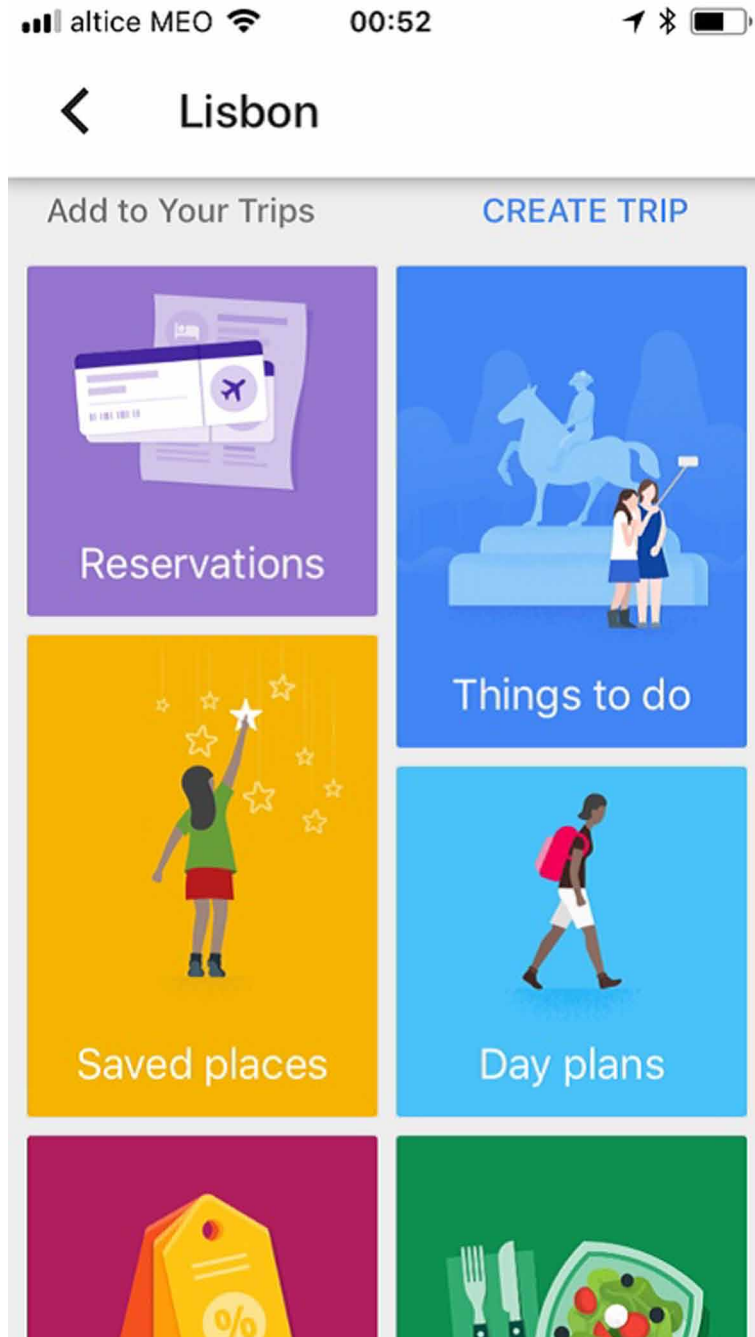
According to Cohen et al (2014), consumers use technology for many consumption-related tasks such as searching for information, buying, sharing opinions and experiences and for entertainment purposes. Social media, for example, has developed into one of the most important influences on tourism. Tourist organizations can benefit from a greater understanding. From a methodological point of view, tourists' growing use of (mobile) technologies can expand our knowledge on travel behavior by providing researchers the opportunity to use different methodologies (e.g. mobile ethnography) and data collection methods.

Ukpabi & Karjaluo (2017) say that the deployment of mobile technology has fundamentally transformed tourism offerings. Mobile technology enhances transactions through mobile devices and provides guides, even when the tourist is at the destination. Mobile apps are reportedly the 7th most downloaded apps, with 60% of global smartphone users downloading travel apps onto their devices and 45% in this group using these apps regularly to plan travel. Information and communications technology (ICT) has permeated virtually every sector. However, consumers adopt these emerging technologies in different ways.

Tourist independent (Tripathy et al 2018) mobility in many countries is a paradigmatic example for a smart city. Independent mobility of tourists is limited in many countries. Some of the major reasons include: (1) lack of trust in the ongoing tourism services, (2) lack of security support, and (3) fraud prone tourist places. Smart tourism is an example of the important components of smart cities.

To date (Gretzel et al 2016) the concept of smartness, which characterizes everything that is embedded or enhanced by Information and Communication Technologies (ICT), mainly emphasizes how interoperable systems can integrate functions that have the ability to manage big data and generate value for all stakeholders. The relevance of technology in discussing smart cities and smart tourism destinations should not be underestimated.

Figure 2. Google Trips main menu (Source: Google Trips, 2019)



## CONCEPTUAL MODEL

The proposed model work as a recommendation system for sightseeing tours, according to user's current location, available time, personal preferences and availability in real time of touristic

**Table 1. Strengths and weaknesses of similar researches**

Project	Strengths	Weaknesses	Author
PAT-Planner	Several user evaluations were made. Main goals were achieved and also the intention of implement geolocation for user tracking. The proof of concept of PAT-Planner is directly related of what is wanted to develop in this work	Lack of information about the failed tests during the evaluation with users.	Lu, E. H. C., Fang, S. H., & Tseng, V. S. (2016)
Multi Request Route Planning	Architecture presented by perceptible diagrams. Web services culture that can be useful to this project. Usage of Google Places API to get places according to user's location.	The Context awareness term is not mentioned by the authors of this article and it is a very important area to explore.	Lu, E. H. C., Chen, H. S., & Tseng, V. S. (2016)
Trip planning route	The time factor is the biggest challenge of this article. The results of the route generator is the combination of two methods: distance between two points and another one using Google Maps API LUT.	No tests were applied to proof this model of concept.	Chia, W. C., Yeong, L. S., Lee, F. J. X., & Ch'ng, S. I. (2016)
Xenia	Context awareness it is the biggest challenge of this article. Project's architecture presented by perceptible diagrams. Web services culture that can be useful to this project.	No tests were applied to proof this model of concept.	Korakakis, Mylonas & Spyrou (2016)
Trip-Mine	The authors developed optimization mechanisms to improve the efficiency of the place search. They analyzed and adopted the Travelling Salesman Problem Algorithm to start the logic of their own algorithm.	Due to the great theory around this article, the authors forgot about giving examples of their concept in real life.	Lu, E. H. C., Lin, C. Y., & Tseng, V. S. (2011)
Touch-map	Problems were well defined. The authors also developed a very well defined framework to solve their problems. This project was developed based on Trip-Mine that was also analyzed in this state of the art. The evaluations were made in several places around the world, like USA and Asia.		Ying, J. C., Lu, E. H. C., Huang, C. M., Kuo, K. C., Hsiao, Y. H., & Tseng, V. S. (2013)
Personalized Sightseeing Planning System	The authors analyzed and adopted the Travelling Salesman Problem Algorithm to start the logic of their own algorithm. They save user data that can be useful in the future.	There is no future work vision that can explain to where they want to grow.	Anacleto, R., Figueiredo, L., Almeida, A., & Novais, P. (2014)
Google Trips	Google Trips uses Google Places API. The biggest advantage is to use this in-house API. This mobile application is much known around the world and has millions of active users. This is a positive point because it turns information more trustable.	The quantity of the available features can make the user experience worst. The process to create a new touristic walking route is very long and the user has to select a lot of options to finish.	Google Trips (2019)

attractions (context awareness). It retrieves points of interest from external API's that holds this kind of information.

To develop it, the authors ran a survey with 12 questions in order to properly understand which tasks are being done currently by the users. One way to get this information is ask them what tasks they prefer or avoid to do while using similar recommendation systems and also, what tasks they want to see available in the near future.

Firstly, the questionnaire presented in this study was structured to obtain an answer to “11 Questions of Task Analysis”, because getting these answers gives a more clear insight of what users really need to solve their model problems of current systems (Lewis and Rieman 1993).

The survey was prepared and sent to potential end-users of this type of systems. It allowed us to know some personal information about the standard profile of the potential end-users, such as their age, gender, current and future wished tasks to be supported by the recommendation system. With regard to its dissemination, the social network Facebook was chosen where, through contacts and public publications in specific groups of tourist itineraries, it was possible to reach at a considerable number of people who were really interested in answering and knowing more about the theme in question.

As a result of the survey, the following list represents the most suggested end-user’s needs during the interview process, while table 2 shows the relationship between the interveners and the requirements:

1. List only sightseeing tours with touristic attractions open now (context awareness);
2. List only sightseeing tours based on user’s preferences;
3. List only sightseeing tours based on user’s available time.

Starting with the analysis of the results of the questionnaire, regarding age, 13.5% of respondents were between 35 and 44 years old, 17.3% were between 18 and 24 years old and 42.3% were between 25 and 34 years old, which makes these three age groups combined have a representation of 73.1% in the sample. Because the Internet has been used as the primary means of collecting responses, it may justify the greater prevalence of younger age groups.

**Table 2. Intervenors and their requirements (Source: Authors)**

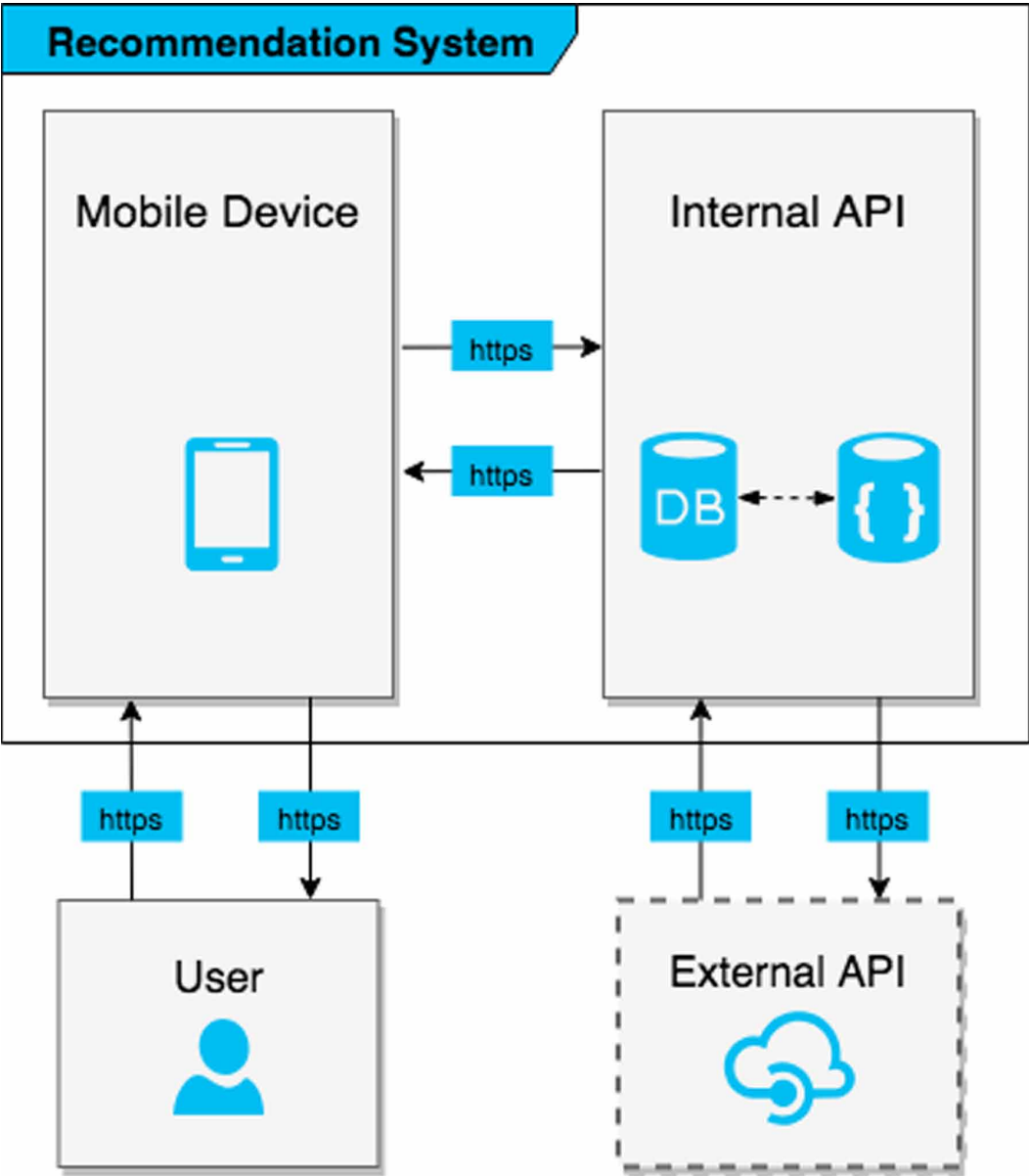
Intervenors	Requirements
<b>End-user</b>	<p>The system user must:</p> <ul style="list-style-type: none"> <li>Be able to select the category of the tour to be suggested.</li> <li>Be able to select the available time to perform a tour.</li> <li>Re-generate a new tour if the current one is not to your liking.</li> <li>Follow the tour directions through a guided view oriented to georeferencing.</li> </ul>
<b>RS</b>	<p>The RS should:</p> <ul style="list-style-type: none"> <li>Show tourist itineraries according to the entrance parameters of the end-user and CA.</li> <li>Request end-user permission to access your global current position (GPS).</li> <li>Assist the user in following the different POIs that make up the selected roadmap.</li> </ul>

## Architecture

The recommendation system model is based on an online Architecture, which means that all included features must be accessed through a device connected to the Internet. This is an important requirement since it is necessary to know, in real time, information about places around the user. Figure 3 illustrates the architecture proposed to solve the problem.

In the internal API layer is where all data is handled. As it can be seen, the basis of the Business Logic data and Source Code are in this layer and communicate with each other. To complement the information stored locally by this database, the source of information about the tourist itineraries

Figure 3. General RS architecture (Source: Authors)



to create, come from external APIs, the last layer in the diagram. It should also be noted that all communications between all layers are made through HyperText Transfer Protocol Secure (HTTPS), to ensure more efficiency on the security and protection of the data circulating between layers and which, without the applicability of this protocol, would be easily captured by third parties.

### The Tour Itinerary

After analyzing the model architecture showed in figure 3, it is possible to understand that the most critical and challenging part is the connection between internal and external API's. One of the end-user's inputs is his available time at the moment of the sightseeing tour creating process, so it is

important to get places from the external API according to this input. The greater the available time is, the highest is the number of places that should be returned.

Regarding the layer of external APIs, although they have only one action during the process, this action is the most important and critical. If for some reason the communication fails, the creation of the tourist itinerary is compromised because the information about the POIs that will constitute these itineraries do not reach the application.

However, the problem mentioned in the previous paragraph is not the only one that needs to be solved. Within the repetitive cycle, which allows to obtain the POIs that will constitute a certain tour itinerary, when the POI is received from the external API and before it is saved in a `listPOIs []` array, it should be checked whether it was already in the list to be returned at the end or not.

It is extremely important to pay attention to the execution time and complexity that this problem involves. Figures 4 and 5 show two proposed algorithms to solve this complexity.

As Figure 4 shows, the repetition statement is only executed while total time is less than the limit time. After that condition, the filling of the `POIsList` should be finished and will represent every suggested stop during the sightseeing tour.

Figure 4. Get POIs based on location and category (Source: Authors)

---

**Algorithm 1** Otencao de POIs com base na localizacao e categoria

---

```

1: tempoLimite ← Input
2: tempoTotal ← 0
3: listaPOIs ← []
4: while tempoTotal < tempoLimite do
5:   categoriaRoteiro ← Input
6:   coordenadasUtilizador ← Input
7:   locaisAbertos ← ContextAwareness           ▷ Faz set do valor a "true"
8:   argumentos[] ← categoriaRoteiro, coordenadasUtilizador, locaisAbertos
9:   POI ← ApiExterna(argumentos)
10:  if POI is not in listaPOIs[] then
11:    listaPOIs[] ← POI
12:  end if
13: end while

```

---

Figure 5. Calculate limit time of the tour (Source: Authors)

---

**Algorithm 2** Calculo do tempo limite do roteiro

---

```

1: tempoLimite ← Input
2: tempoTotal ← 0
3: while tempoTotal < tempoLimite do
4:   (...)
5:   tempoTotal ← tempoTotal + ApiExterna[POI][tempo]
6: end while

```

---



Sometimes a duplicated POI should be returned by the external API. To avoid this situation, this algorithm demonstration has also a condition to do not let POIsList be filled by duplicated data.

Finally, figure 5 that demonstrates how to end with the repetition statement. The total time must be always incremented by the time between current and POI's locations.

## PROTOTYPE IMPLEMENTATION

In order to validate and test our model a prototype was implemented. It is a mobile application to allow the end-user insert his inputs and obtain suggestions in real time, supported by a convenient graphical tool.

### Internal API

Symfony 3.4 framework (Symfony, 2019) was the chosen technology to implement the Internal API. The main reasons for this decision were existing professional and academic experience already gained in the use of these technologies and excellent official documentation of Symfony available online. Symfony 3.4 version is more minimalist than previous versions, which means that becomes a cleaner solution, besides using Doctrine ORM (Doctrine ORM, 2019) for data layer abstraction which is easier. The selected database management system was MySQL. It is easier to map with Doctrine ORM and grants reliability and data security.

Figure 6 represents the Symfony framework architecture assumed to this work. This framework is based on the Model View Controller (MVC) architecture standard and the code organization encourages the developer to adopt good practices when compared to with the other PHP frameworks. Figure 7 illustrates how Doctrine ORM deals with the database layer. In this case the table contains data that will be transformed to a JSON Object.

In order to structure the internal API, the implementation process began by setting up the Symfony to return JSON responses whenever a new request reaches the controllers of the application. To do this, it was installed via Composer FOSRestBundle, which is a component that has as a strong point bringing many basic configurations to API's Representational State Transfer (REST).

In order to structure the internal API, the implementation process began by setting up the Symfony to return JSON responses whenever a new request reaches the controllers of the application. To do this, it was installed via Composer FOSRestBundle, which is a component that has as a strong point bringing many basic configurations to API's Representational State Transfer (REST).

### External API

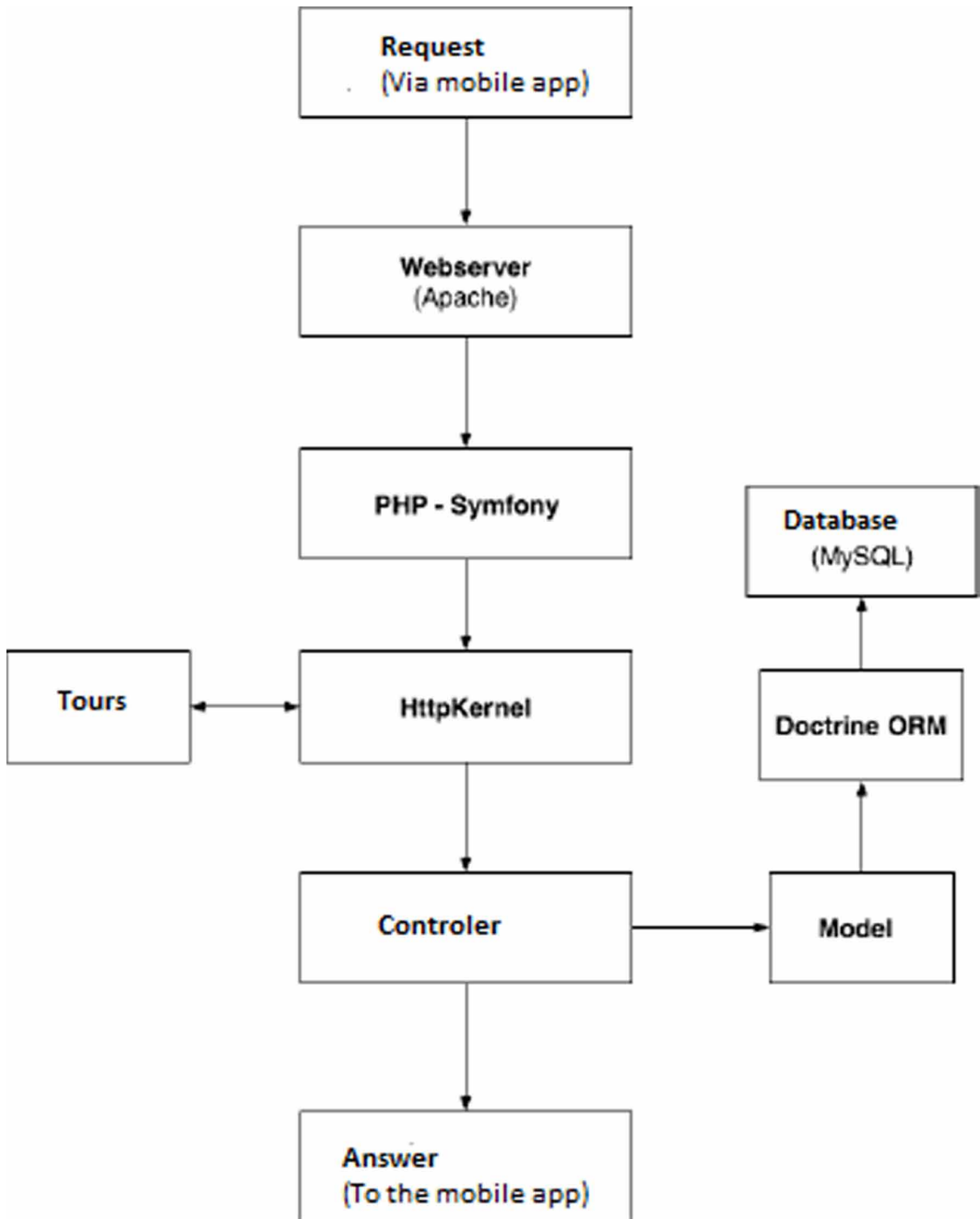
For the external API it was decided to use Google API's because one of the most important requirements was to get updated data from POI's in real time and with no geographical restrictions. It was necessary to obtain quality and trusted data, since the main goal of this recommendation system for sightseeing tours is to turn the user experience as best as possible.

The most projects listed in section 2, used Google external API's when needing to access data about location and POI's. Google has several API's with different proposes. In this case the authors opted for the following ones: Google Places API and Google Distance Matrix API.

Google Places API (Google Places, 2019) is a service with information about places, being those establishments, geographic locations or points of interest. With this API it is possible to get places in the neighborhood, by specific coordinates, types (categories) and according to a Boolean parameter – indicates if they are open now or not (context awareness). These 3 parameters are directly related with the goals of this work, which means that it will be possible to get all POIs according to them.

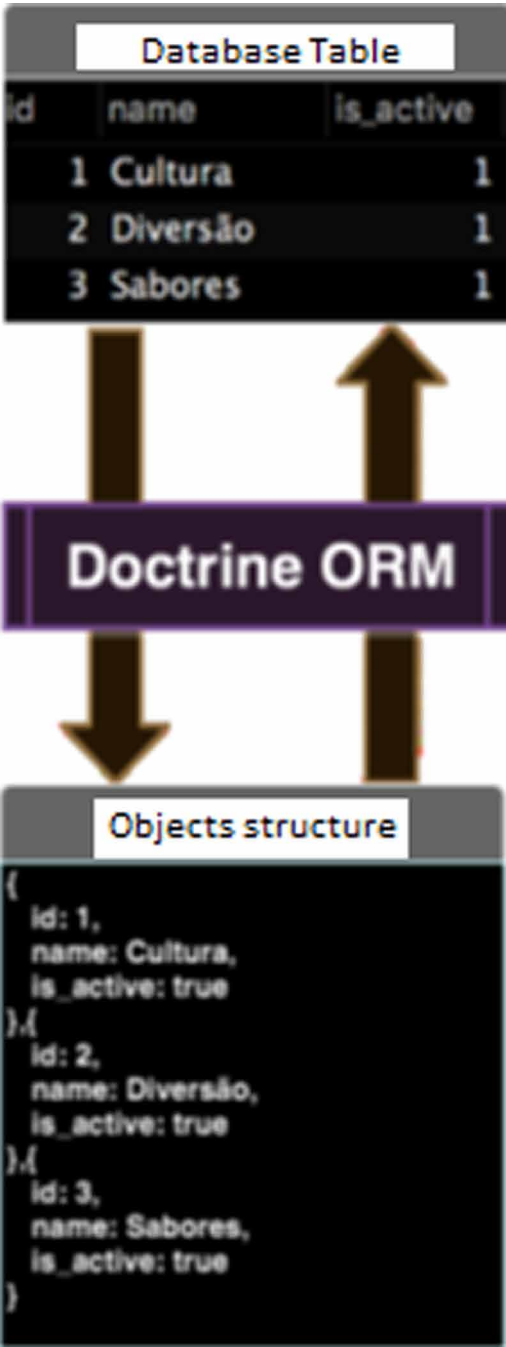
On the other hand there is the Google Distance Matrix API (Google Distance Matrix, 2019). Thanks to this service is possible to calculate how long will take to get from Point A to Point B when hiking. The distance between these two points is also possible to get in the same service response object.

Figure 6. Symphony architecture considering the API development (Source: Authors)



Within the context of this external API, there are several services with different types of information, the one that was most interesting to implement was the Place Search. The Place Search has also several sublevels of requests that can be executed and with different types of information. Although all types of requests can be used, a decision has been taken to use the NearBy Search only. With NearBy Search you can get:

Figure 7. Doctrine ORM (Source: Authors)



1. A list of places with summary information about them.
2. Within a specifically defined geographical area.

## Database

As defined above, the database that is supporting the internal API was developed using the database management system MySQL. Considering all features to be implemented in the prototype of this work, it was only needed to create 4 tables represented in Figure 8.

The first decision made was to create a mapping between Category and GoogleType entities. The first one holds information about the three category types to be selected by the end-user and it is only used internally, while the second one has several sub types defined by Google Places API. Figures 9 and 10 to show the content of each described table:

1. For Category entity “Cultura”, the three sub types are mapped in GoogleType entity: “art\_gallery”, “church” and “museum”.
2. For Category entity “Diversão”, the three sub types are mapped in GoogleType entity: “amusement\_park”, “night\_club” and “park”.
3. For Category entity “Sabores”, the three sub types are mapped in GoogleType entity: “bakery”, “cafe ” and “restaurant”.

Thanks to this mapping, it is possible to suggest a more varied tour to the user.

## Mobile Application

The development of the mobile application considered the use of the NativeScript framework because of the already existing professional experience and in addition, this framework turns the development faster and easy to connect with the internal API.

The design of the interface has taken into consideration its simplicity and minimalist look. This was done taking into account that an app interface must be easy and fast to use, besides intuitive (Lu et al, 2012). Special care was considered in its design as it was actually the “face” of the test object to be used by the end users. The app has only four screens, all managed by the app user, listed in the following points in sequential order: choice of category (tour theme), choice of the time available for the tour, acceptance of the sharing of the access of the current location of the end-user, and a map with the entire route calculated and ready to be performed.

Figures 11 and 12 illustrate the mobile application running.

After the end-user inputs the category and available time, the mobile application asks him to allow it access his location information and finally generates the sightseeing tour, as can be seen in figure 11.

User location is updated in real time to help him found the next POI. Also, these POIs are ordered by proximity and numbered when user clicks on any of them.

Requests to Google’s external APIs to obtain within-feature locations selected by the user are performed right after clicking the “Ok” button to the question “Allows the app accesses your location?”. So if the end-user wants to cancel and not accept this request, requests to external APIs are avoided, saving runtime and number of orders placed, which, as already mentioned, has a daily limit to be met.

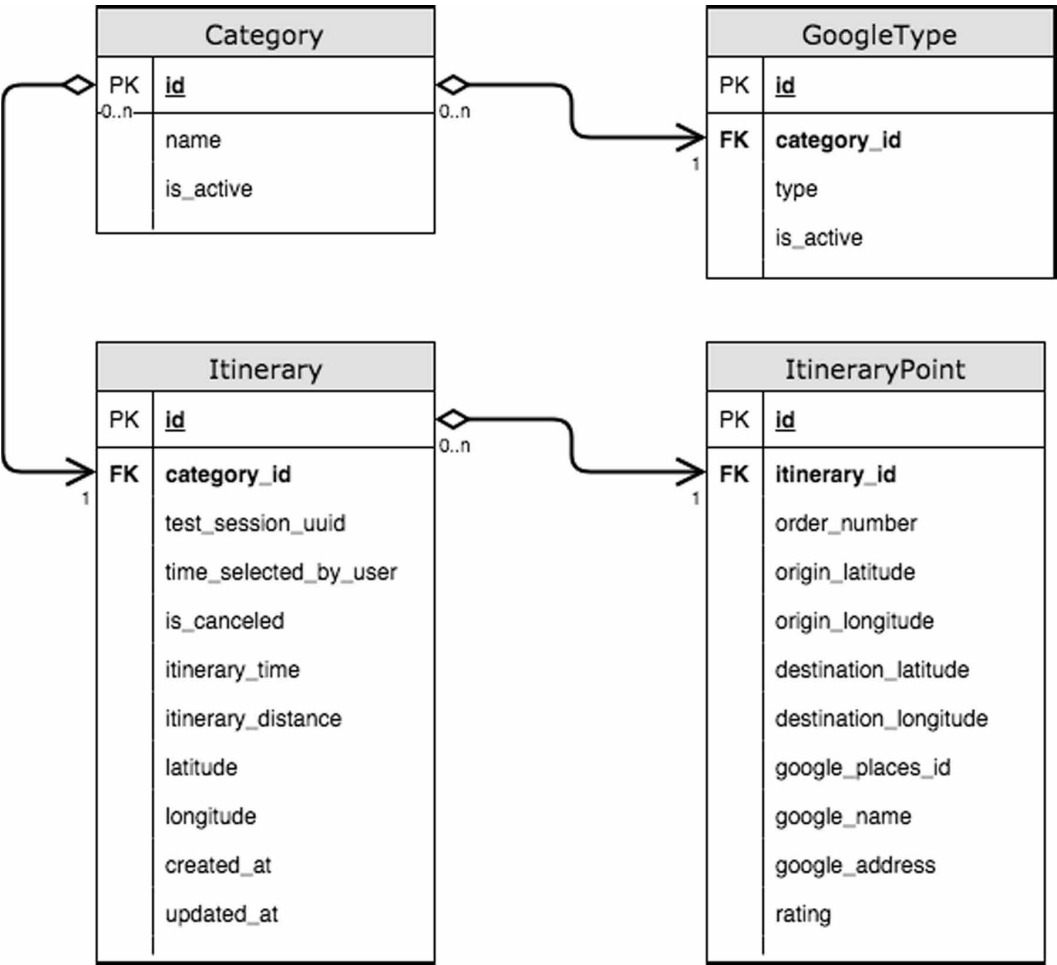
## Evaluation

Two types of tests were performed: performance and functionality. With the first one it was expected to obtain results about the performance of the API and the Server where it was hosted. The functionality tests wanted to validate what users think about the proposed recommendation system.

## Performance Tests

About the test environment is important to refer that the internal API is located at DigitalOcean, a hosting provider, with the following technical characteristics: 512MB of RAM, 20GB of disk space, operating system Linux Ubuntu 16.04 located at Amsterdam, The Netherlands.

Figure 8. E-R (Entity-Relationship) Diagram (Source: Authors)



To run the performance tests, it was used a tool called JMeter (JMeter, 2019). This work tool allows run tests simulating different number of simultaneous end-users using different endpoints of the API.

To test many requests to the GET category endpoint it was first defined 5 simultaneous end-users. After that, the same endpoint was requested by 10 end-users. This is an operation with a very low level of complexity as whereas at this time there are only three main categories to list in the mobile app. What this performance test tells us is that, given these conditions, there will be no performance issues to report. Figure 13 represents a chart with the final results of this test.

Then, the most complex operation of the RS model was tested, the creation of a new touristic itinerary. This is the operation that, after knowing all the input parameters that the end-user has chosen, will create a new tour also taking into account the various calls to Google’s external APIs. Five simultaneous requests were first taken into account considering that the available time of the end-user was 7200 seconds (2 hours).

As a comment to the results, considering that this is an operation sensitive to database writing and external API calls, the tests eventually ran above the expectations initially estimated. The next objective was to increase to the double the number of requests and analyze the results and then, to increase the available time value from 2 hours to 8 hours (28800 seconds).

Figure 9. Static data from Category table (Source: Authors)

id	name	is_active
1	Cultura	1
2	Diversão	1
3	Sabores	1

Figure 10. Static data from GoogleType table (Source: Authors)

id	category_id	type	is_active
1	2 →	amusement_park	1
2	1 →	art_gallery	1
3	3 →	bakery	1
4	3 →	cafe	1
5	1 →	church	1
6	1 →	museum	1
7	2 →	night_club	1
8	2 →	park	1
9	3 →	restaurant	1

Both results output were very positive, although it was conclusive that the increase in the time available for the tour by the end-user naturally impacted in the final calculation of the creation of new tours. Figure 14 represents the final results considering requests to POST create tour.

The last performance test, POST Itinerary with 10 requests to 8 hours of end-user's available time, considered the increase in the number of requests to double, compared to the previous tests. A load was expected proportionally direct and in a way that's what happened again. The server failed for the first time, one of the ten requests was not made successfully due to an error 500 (internal error in the server). This explains the overload, which was not a surprise on the other hand. It was a very heavy test and wrote in several tables of the database.

## FUNCTIONALITY TESTS

The protocol followed in the tests was based on empirical or end-user evaluation as it is a method of evaluating a recommendation system tested with real users. In this way, it was possible to collect the data in a controlled environment on how an end-user interacts with the system and what problems he faces, which are typically unpredictable. During the evaluation process the authors had to ensure that there were no factors outside the system that could cause errors or induce the end-user that the problem was originated from the application, such as no internet or server issues.

Figure 11. Tour category selector (Source: authors)

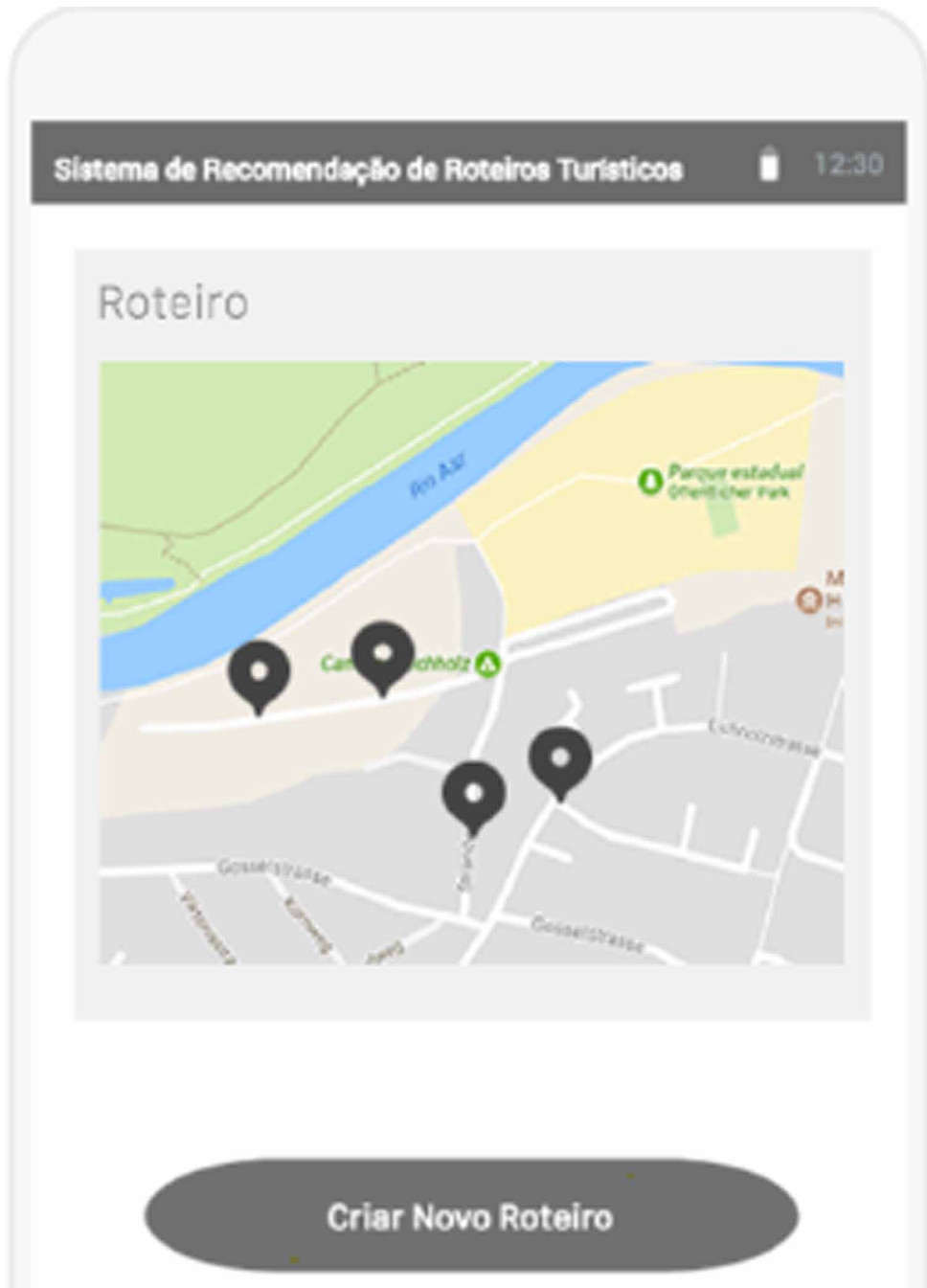


In order to ensure the success and transparency of the functionality tests, the end-users' objectives were further explained with the following guidelines:

1. The purpose of this test was to evaluate the system, not the user.
2. The user could talk freely.
3. The results obtained have the sole purpose of improving the interface.

The tests were carried out in the greater Lisbon area. All tests were tried outdoors, as this is the essence of any RS of tourist itineraries, because to get to know and visit the sites, they have to be reached, in this case, walking. In addition, this is a system that relies on the proper functioning of GPS in order to obtain more accuracy, so physical obstacles that may reduce the signal strength of this frequency, such as ceilings or building walls, should be avoided. This is a common practice that users accustomed to use these kind of systems, have in mind.

Figure 12. Sightseeing tour created (Source: authors)



To validate the prototype functionality, the tests were performed with 16 end-users: 10 males and 6 females. They were between 18 and 40 years old. The subjects were chosen considering people living within Lisbon and where it is more probable to get relevant POI's results.



Figure 13. GET Category (requisitions x milliseconds) (Source: authors)

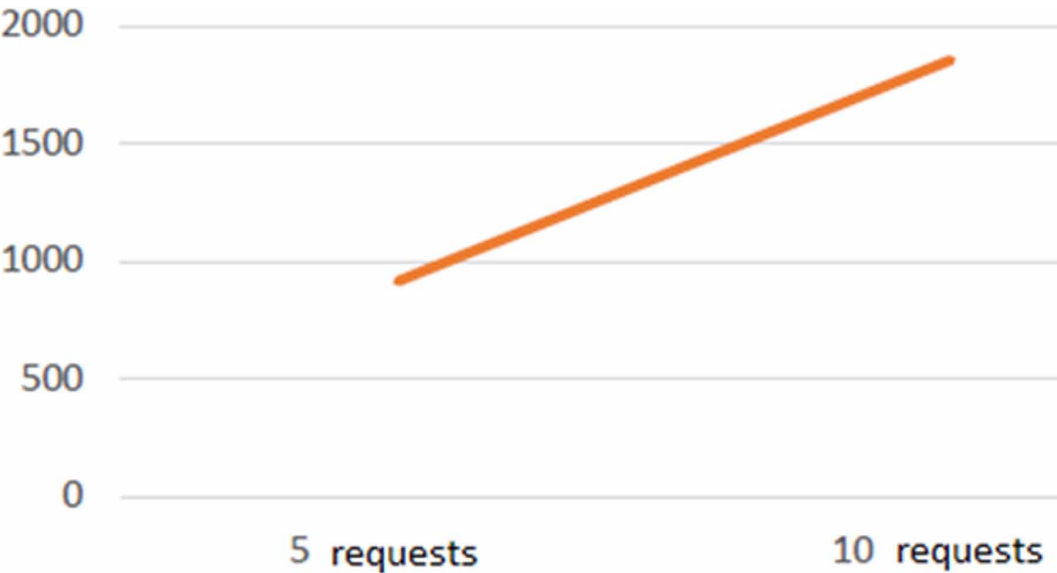


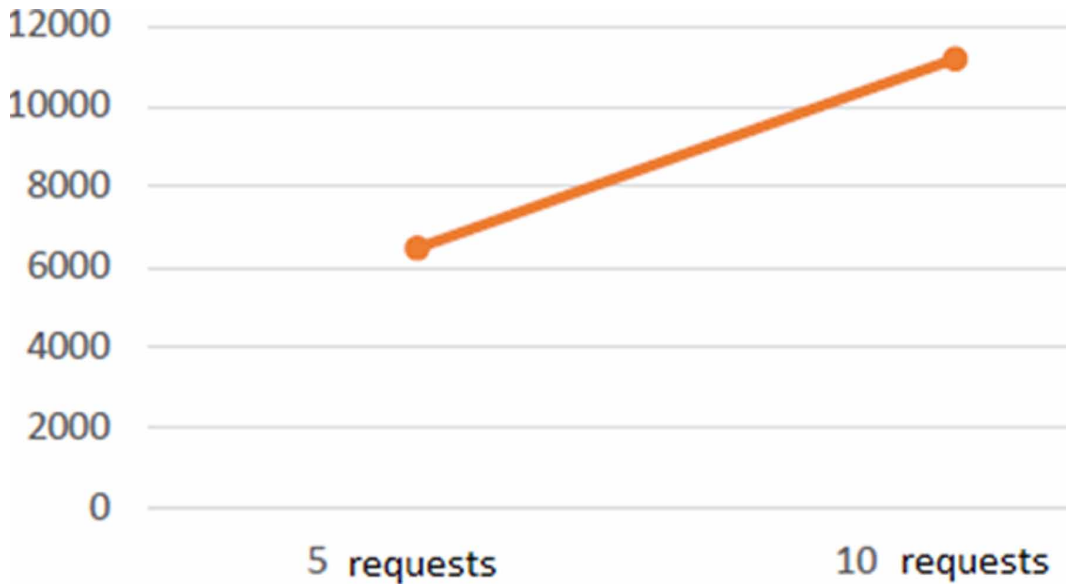
Figure 14. POST Create tour for 2 hours (requisitions x milliseconds) (Source: authors)



To test all functionalities, it was installed the mobile application on every end-user's cell phones. They followed a test script to know what to do during it, with the following steps:

1. Go to the recommendation system application;
2. Create a sightseeing tour answering all questions;
3. If you are satisfied with the suggest tour, start the journey using the help provided by the application;

Figure 15. POST Create tour for 8 hours (requisitions x milliseconds) (Source: authors)



4. At the end, before turning off the application, call the supervisor of the test;
5. Turn off the application.

After the 16 tests were accomplished, 4 requirements were identified to be evaluate:

1. R1: User was able to select the tour category;
2. R2: User was able to select the available time wanted;
3. R3 (optional): User does not like the sightseeing tour suggested and was able to return back to create a new one;
4. R4: User was able to follow the tour indications supported by a real time map.

Table 3 details the functionality tests results.

In general terms, the following strengths were noticed:

1. The vast majority of end-users did not need help completing the task to the end.
2. 100% of users completed the task.

Table 3. Evaluation results (Source: Authors)

Requirement	Success (%)	Failure (%)	Not done (%)	Usage experience	
				Positive (%)	Negative (%)
R1	100	0	0	100	0
R2	100	0	0	100	0
R3	50	18,75	31,25	62,5	37,5
R4	93,75	6,25	0	87,5	12,5

3. There were no problems that did not have a solution.
4. Most users have shown focus on the testing.

As a negative point, there were a considerable number of wrong actions, performed by the end-users.

## CONCLUSION AND FUTURE WORK

In this paper the authors proposed a recommendation system for sightseeing tours based on 4 essentials requirements: personal preferences, available time, current location and context awareness variables.

Performance and functionality tests revealed positive results. In terms of performance, it was concluded that, the more complexity is requested to the different endpoints, the more time the server needs to take processing the data, which is a normal behavior, but the fact that it is a proportional result, proves that the server and the internal API responded with success.

In functionality tests it is important to refer that the success rate for all requirements was expressly positive, considering percentages of usage experience between 62, 5% and 100%, which means that this model can be very useful in this scientific area. R3 is the requirement that must have improved in the future, because has a considerable percentage of “Not done” and “Negative” usage experience.

As future work there are some interesting features that can improve this recommendation system model, such as the average visit times to spend in each POI: With this feature it is possible to give to end-users a more accurate estimated time of a sightseeing tour. This feature can offer a considerable algorithmic complexity that must be studied before the development phase.

Another potentiality to explore is to include the possibility of adding social media networks to the RS. In this way, the end-user would have a suggested tour also complemented by others tips, if he allows. Like TripAdvisor and similar websites that use the comments given by the users to tune the quotation about a hotel, the app would also use this information to design a more interesting tour, for instance.

Another interesting aspects are meteorology and the sightseeing tours list. A total of 21, 2% of the survey's respondents indicated that the app should include some meteorology information also. With this feature the context awareness will be improved in this recommendation system. In terms of sightseeing tours list, at this moment the only graphic visualization method is to show the suggested tour in a map. But there is no option to select a tour from a list of several tours. Some end-users that performed the functionality tests referred about the lack of this feature.

One major limitation faced by the architecture of this prototype is the connection between internal and external API's due to the need of knowing the end-user's available time for the tour in order to get places from the external API. This may impact in terms of overall performance. A minor limitation was the total number of subjects who participated in the prototype tests. A greater number and more varied profile would be richer in terms of evaluation results, but due to time and subjects availability, it was not possible.

Finally, in terms of managerial implications, this work introduces a mobile app that can be simply downloaded and used. Any tourist arriving in a city and having a short or longer time available to visit it, can create a fast, personalized and effortlessly tour, within the available time. This tour will be presented in an interactive map visual approach, and lead the tourist to the interesting points of the city, according his preferences, and mostly, the available time to perform successfully the proposed sightseeing. This is greatly possible due to its innovative methods of calculation and evaluation of the tour.

## REFERENCES

- Anacleto, R., Figueiredo, L., Almeida, A., & Novais, P. (2014). Mobile application to provide personalized sightseeing tours. *Journal of Network and Computer Applications*, 41, 56–64. doi:10.1016/j.jnca.2013.10.005
- Chia, W. C., Yeong, L. S., Lee, F. J. X., & Ch'ng, S. I. (2016, August). Trip planning route optimization with operating hour and duration of stay constraints. In *2016 11th International Conference on Computer Science & Education (ICCSE)* (pp. 395-400). IEEE. doi:10.1109/ICCSE.2016.7581613
- Cohen, S. A., Prayag, G., & Moital, M. (2014). Consumer behaviour in tourism: Concepts, influences and opportunities. *Current Issues in Tourism*, 17(10), 872–909. doi:10.1080/13683500.2013.850064
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271. doi:10.1007/BF01386390
- DoctrineO. R. M. (2019). Available at: <http://doctrine-project.org>
- Google Distance MatrixA. P. I. (2019). Available at: <https://developers.google.com/maps/documentation/distance-matrix/>
- Google PlacesA. P. I. (2019). Available at: <https://developers.google.com/places/>
- Google Trips. (2019). Available at: <https://get.google.com/trips/>
- Gretzel, U., Zhong, L., Koo, C., Boes, K., Buhalis, D., & Inversini, A. (2016). Smart tourism destinations: ecosystems for tourism destination competitiveness. *International Journal of Tourism Cities*.
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2008). Design science in information systems research. *Management Information Systems Quarterly*, 28(1), 6.
- Holjevac, I. A. (2003). A vision of tourism and the hotel industry in the 21st century. *International Journal of Hospitality Management*, 22(2), 129–134. doi:10.1016/S0278-4319(03)00021-5
- JMeter. (2019). Available at: <https://jmeter.apache.org>
- Korakakis, M., Mylonas, P., & Spyrou, E. (2016, October). Xenia: A context aware tour recommendation system based on social network metadata information. In *2016 11th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)* (pp. 59-64). IEEE doi:10.1109/SMAP.2016.7753385
- Lewis, C., & Rieman, J. (1993). *Task-centered user interface design. A practical introduction*. Academic Press.
- Lu, E. H. C., Chen, H. S., & Tseng, V. S. (2016). An efficient framework for multirequest route planning in urban environments. *IEEE Transactions on Intelligent Transportation Systems*, 18(4), 869–879. doi:10.1109/TITS.2016.2593707
- Lu, E. H. C., Fang, S. H., & Tseng, V. S. (2016). Integrating tourist packages and tourist attractions for personalized trip planning based on travel constraints. *Geoinformatica*, 20(4), 741–763. doi:10.1007/s10707-016-0262-1
- Lu, E. H. C., Lin, C. Y., & Tseng, V. S. (2011, June). Trip-mine: An efficient trip planning approach with travel time constraints. In *2011 IEEE 12th International Conference on Mobile Data Management* (Vol. 1, pp. 152-161). IEEE.
- Lu, J., Chen, Q., & Chen, X. (2012, July). App interface study on how to improve user experience. In *2012 7th International Conference on Computer Science & Education (ICCSE)* (pp. 726-729). IEEE. doi:10.1109/ICCSE.2012.6295176
- Pascoe, J., Ryan, N., & Morse, D. (2000). Using while moving: HCI issues in fieldwork environments. *ACM Transactions on Computer-Human Interaction*, 7(3), 417–437. doi:10.1145/355324.355329
- Symfony. (2019). Available at: <http://symfony.com>
- Tripathy, A. K., Tripathy, P. K., Ray, N. K., & Mohanty, S. P. (2018). iTour: The future of smart tourism: An IoT framework for the independent mobility of tourists in smart cities. *IEEE Consumer Electronics Magazine*, 7(3), 32-37.

Ukpabi, D. C., & Karjaluoto, H. (2017). Consumers' acceptance of information and communications technology in tourism: A review. *Telematics and Informatics*, 34(5), 618–644. doi:10.1016/j.tele.2016.12.002

Vanhove, N. (2017). *The economics of tourism destinations: Theory and practice*. Routledge. doi:10.4324/9781351263801

Ying, J. C., Lu, E. H. C., Huang, C. M., Kuo, K. C., Hsiao, Y. H., & Tseng, V. S. (2013, March). A framework for cloud-based POI search and trip planning systems. In *2013 1st International Conference on Orange Technologies (ICOT)* (pp. 274-277). IEEE.

*Ricardo Valadas has a Master degree in Management Information Systems from Instituto Superior Técnico and a bachelor degree in Computer Science from Instituto Politécnico de Beja. Presently, he is a Senior Software Engineer at Edge, Lisbon, Portugal, being the head of Technology & Senior PHP Consultant.*

*Elizabeth Simão Carvalho received her PhD in 2008 in Systems Technology and Programming and Master in Computer Science in 1994, both from University of Minho. She has a post-graduation in Systems Analysis (1990) and a bachelor degree in Electrics Engineering (1984), both from Universidade Veiga de Almeida, Rio de Janeiro, Brazil. Her research goals are Information and Scientific Visualization, besides Mixed Reality. Presently she is working as Professor Auxiliar in Portuguese Open University and coordinator of a bachelor degree. She has author or co-author more than 30 scientific articles in the field and is a regular reviewer and supervisor in the area.*