Software Engineering Project

Donatas Kozlovskis, Enric Cornellà
Andrey Pak, Fernando Garcia
VIBOT-8

11/12/13

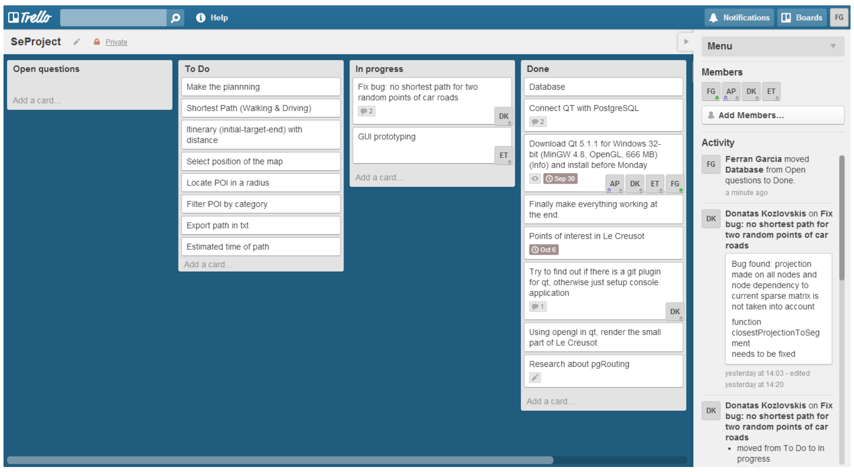
1 Introduction

The present document illustrates all decisions, procedures and methodology followed during the development of the application map. But also specific documentation regarding the implementation such as diagrams or remarkable algorithms specifically designed to solve the problems from the implementation.

This document has been divided in three different main sections; the first one shows the time line followed for carrying out the project and the common decision taken regarding the two other following sections, the second section focusses on the C++ implementation and finally, the last one, focusses on the Matlab part.

1.1 Tools and resources

This section shows which software has been used directly or indirectly during the whole phase of design as well as the phase of development. In order to manage the whole project and its tasks, we have chosen the on-line manager Trello that keeps track of everything, from the big picture to the minute details. In this way, all kind of tasks can be divided and assigned to the people involved in the project and also can be located in different boards according to the state of development of the task as in the following image:



Open questions
Add a card...

To Do
Make the plannning
Shortest Path (Walking & Driving)
Itinerary (initial-target-end) with distance
Select position of the map
Locate POI in a radius
Filter POI by category
Export path in txt
Estimated time of path
Add a card...

In progress
Fix bug: no shortest path for two random points of car roads 2 DK
GUI prototyping ET
Add a card...

Done
Database
Connect QT with PostgreSQL 2
Download Qt 5.1.1 for Windows 32-bit (MinGW 4.8, OpenGL, 666 MB) (Info) and install before Monday Sep 30 AP DK ET FG
Finally make everything working at the end.
Points of interest in Le Creusot Oct 6
Try to find out if there is a git plugin for qt, otherwise just setup console application 1 DK
Using opengl in qt, render the small part of Le Creusot
Research about pgRouting
Add a card...

FG **Ferran Garcia** moved **Database** from Open questions to Done.
a minute ago

DK **Donatas Kozlovskis** on **Fix bug: no shortest path for two random points of car roads**
Bug found: projection made on all nodes and node dependency to current sparse matrix is not taken into account
function closestProjectionToSegment
needs to be fixed
yesterday at 14:03 - edited yesterday at 14:20

DK **Donatas Kozlovskis** on **Fix bug: no shortest path for two random points of car roads**
- moved from To Do to In progress

Figure 1: Image of the task manager during the development process at early stages. The work is divided in four different boards with multiple drag and drop tasks and is assigned to one person

It is also important to establish the IDE and tools before starting to develop the project. On one hand, we assigned that Qt 5.0.1 OpenGL under Mingw compiler would be the appropriate platform because it is the last stable version of Qt and also because this open source developing environment allows us to create easier the graphical user interface and also render without adding by hard the OpenGL libraries. On the other hand, regarding to Matlab part, everything has been developed using the R2013a version of the software.

However, the previous software is not the only one used in the project. We supposed correctly the basic necessity to deal with data structures such as information related with roads, nodes, buildings or points of interest among other possible types (see section *Methodology*). The options were to deal with this data coming from one, *txt/xml* format files or similar or two, databases. In both cases, Matlab and C++ implementation, the decision was to work with databases because of the amount of data and specially because is the easiest and fastest way to get detailed information of the structures without going over the whole structure in such cases as getting nodes shared by two different roads (see the *Database* section).

Despite sharing the same EER schema, the database chosen for the Matlab part it is different from the one chosen for the C++ part. In this way, we decided to use PostgreSQL just because it is easier to make it work together with Matlab and using MySQL 5.6 with Qt which was more familiar to us. But also, the fact that we chose two different databases made us able to change between them if some problems of generation had been encountered. Note that we have used the correspondent browser of both platforms.

1.2 Related works

In order to be able to design a user-friendly and helpful application it is necessary to have a look to the current work, for instance GoogleMaps (https://maps.google.es/) or another non common but open-source solutions as MapGuide (http://mapguide.osgeo.org/) or OpenStreetMap (http://www.openstreetmap.org/). In fact checking the last solution we found the way to get most of the data required to render our map and to fulfil our databases. In addition, some of the information provided by this source was not used.

OpenStreetMap has been a good provider of data but at the same time it has been necessary to adapt our database tables to the information obtained and in some cases to add some additional parameters because we have to take into account that this platform is not able to provide a route between two points, at least not in a direct way.

2 Application Design

In this section is explained how the application was devised starting by the definition of the database as well as the analysis of the use cases taking into account the user and the design of the resultant graphical user interface prototype.

2.1 Use cases

In order to prototype the graphical user interface of both parts, we conceived the diagram of cases of use that the user could find. But, first of all, which use cases we have?

According to the statement provided we can identify the following cases:

- Finding the shortest path between two nodes picked up from the screen providing information about distance and time depending on the way of transport.
- Finding the shortest path between three points including the same information.

- Locating a point of interest.
- Finding all points of interest of a specific type around one place.
- Creating/Modifying/Deleting a point of interest.

That also can be summarized in the following diagram:

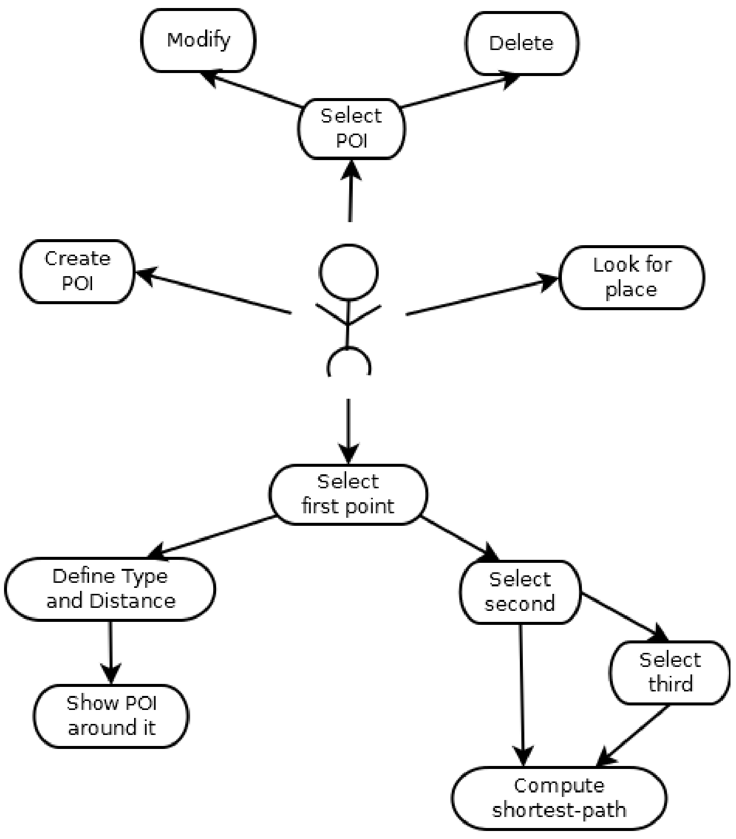


Figure 2: Diagram of the use cases provided to the user

The user has to be able to select a point of interest which already exists, modify it and deleted from the data base. But also has to be able to create a new point of interest and simply located in the map.

Once the user selects a point from the map, no matters if it is random or a point of interest, we have the option to establish the distance we want to locate a specific type of points of interest around it. But also, we have the option to select a second or still a third point of interest for getting the shortest path.

2.2 GUI prototype

After defining the use cases that users have to be able to deal with, we can start prototyping the graphical user interface that is also common between the two implementations. To accomplish this hard task that can be adjusted to solve future requirements, we decided to divide the screen in 5 different relevant points explained below:

THE BROWSER: This section is marked with the number one in the following figure, first GUI prototype. Basically, it brings to the user the possibility to look for a point of interest that is already present in the database.

THE MAP: The zone shows a map able to be scaled (zoom in and zoom out) and reactive to mouse events in the whole surface.

SHORTEST-PATH: Number Three allows the user to establish a route between two or three points taking into account if he or she is moving by car or by foot. But also brings the possibility using the buttons located on the left, to look for a type of point of interest around one position.

POINTS OF INTEREST: Both Buttons located in number four allows the user to create a new point of interest or modify an existing one through a form.

ROUTE: Finally, number five the distance and time that the user will spend having selected some locations previously. Showing the turns and the distances between them. Also brings the possibility to download all this text in a *txt* file.

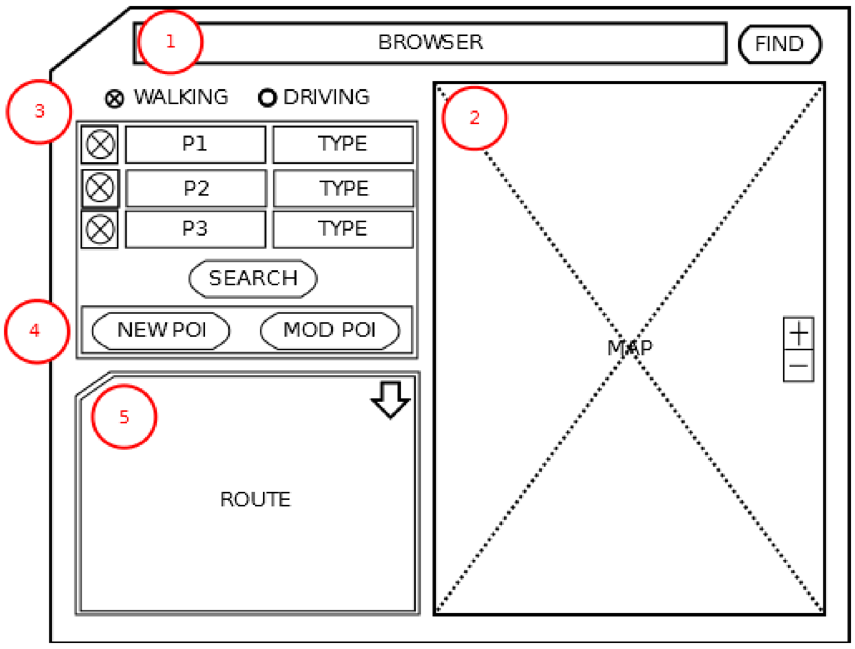


Figure 3: First GUI prototype conceived at early stages of the design process

2.3 Getting the data

With the aim of creating a robust database where solving the problems proposed by the statement and in order to the maintain the wholeness of the data inside we have to consider the following statement:

"One map is composed by ROADS which in turn are composed by different NODES. However, a node could belong to more than one road which identifies an intersection between two roads. The only attributes necessary for a node would be a unique ID and the position of the node in a map defined by LATITUDE and LONGITUDE. In the same way all roads have the following attributes: A unique ID that identifies it, its NAME, a TYPE that describes if the road can be used by cars and finally, if it can be walked in both WAYS."

Here, we can identify two tables, its attributes and the relation between them which can e defined as N:M because of the fact that one node can belong to multiple roads, and one road have multiple nodes. But also this relationship has to be defined taking into account the order the node has in the road that belongs to, giving an attribute of the relationship called ORDER.

On the other hand, the user has to be able to see the points of interest, create news, delete them and also modify the ones which already exist. We decided to create a table in the same schema without any relationship with the previous tables in order to maintain the isolation from the points updated all the time because it does not belong to any specific road. The number of attributes considered from the very beginning for the points of interest (POI) table are: The ID and the position in LATITUDE and LONGITUDE, the NAME of the point, as well as the ADDRESS and the TYPE.

Taking into account the previous steps, we can see the result of the database in the following schema:

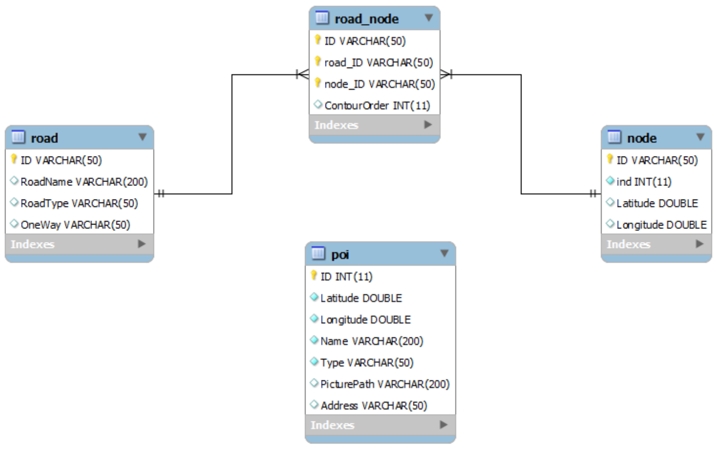


Figure 4: Diagram of the relational database model created during the design

Notice that part of the process of development and parsing of the data from OpenStreetMap into our databases was accomplished together with the rest of the groups.

3 Methodology

4 Procedure