

UNIVERSIDADE DE LISBOA  
FACULDADE DE CIÊNCIAS  
DEPARTAMENTO DE INFORMÁTICA



## **Healthy Track: Healthy Route Recommendation**

Ana Sofia Pereira Nunes

**Mestrado em Informática**

Dissertação orientada por:  
Ana Paula Afonso, Ana Luísa Respício



## Acknowledgements

I would like to acknowledge the continued support of my friends, family and colleagues.

I start by thanking my professors, Dr. Ana Paula Afonso and Dr. Ana Luísa Respício, who throughout this thesis have always been available to help and provided me with much needed advice, which has been crucial throughout the writing process. I couldn't have asked for better guidance.

I also thank Professor Pedro Pinho of the Centre for Ecology, Evolution and Environmental Changes (CE3C) investigation unit and Professor Cristina Catita, who have offered their resources and time to collaborate on this thesis.

Then, can't forget to say thank you to my parents, who are always there for me for the good and bad, the best brother in the world, my many cousins, my godfather and godmother, my amazing grandmother, who has taught me a lot, and all the other members of this very tight-knit family, who have always welcomed me in their hearts and have nurtured and supported me throughout my life in many ways.

I must also thank my long-time friends, Sofia, Carolina, Rita, Miguel and Mariana, who throughout my life have been a constant positive force, a source of laughter and forever cherished memories, and who recently have sent me their messages of support, providing me with strength to finish this thesis.

To all my fellow classmates and friends from college, especially Pedro, Ricardo Jorge, Ricardo Subtil and Catarina, I am grateful for all the great times we have shared throughout college, in our projects and hangouts. I thank your tips and help throughout our studies, and of course, your continued friendship.

Finally, I must give a special thanks to Stefan, who has been my rock throughout this thesis, with his advice, helping hand, patience, unconditional love and friendship. I couldn't have done it without you.



## Resumo

Existem atualmente vários sistemas de recomendação de caminhos (Waze, Google Maps, etc.) que oferecem ao utilizador sugestões de caminhos entre duas localizações, com suporte para diversos meios de deslocação.

Estes sistemas sugerem um ou mais caminhos tipicamente com base em critérios de distância e tempo, i.e., recomendam o(s) caminho(s) mais curto(s)/mais económico(s), por norma não integrando outros critérios da preferência e relevância para o utilizador.

Nos percursos pedestres em ambientes urbanos é frequente que a escolha do utilizador não seja baseada no caminho mais curto ou mais rápido, privilegiando outros critérios, como por exemplo, níveis baixos de poluição, jardins e vegetação, segurança e iluminação.

O objetivo do presente trabalho é desenvolver um sistema de recomendação de percursos pedestres, entre dois pontos num mapa, considerando múltiplos critérios relevantes e as preferências do utilizador.

As contribuições deste trabalho incluem: uma revisão do estado da arte dos sistemas de recomendação, especificamente dos sistemas de recomendação de caminhos e da literatura existente sobre a navegação pedestre, com foco nas preferências dos pedestres nos seus caminhos e com uma discussão de sistemas de recomendação de caminhos pedestres existentes; a base de dados HealthyTrack, que armazena dados geográficos correspondentes a diferentes aspetos da cidade de Lisboa utilizados no cálculo de caminhos saudáveis, e que contém a funcionalidade de recomendação de caminhos com uso da extensão PgRouting; a aplicação *web* HealthyTrack, que fornece uma interface com mapas interativos, com utilização da biblioteca LeafletJS (uma biblioteca Javascript ou JS de mapas interativos), e permite a interação com o serviço de recomendação de caminhos, para sugerir caminhos saudáveis aos utilizadores.

Inicialmente, foi realizada uma pesquisa sobre a taxonomia dos sistemas de recomendação e uma revisão do estado da arte nesta área. Os sistemas de recomendação dividem-se em 3 tipos, de acordo com a sua abordagem de recomendação: recomendações baseadas em conteúdo, onde itens recomendados ao utilizador são similares aos que preferiu no passado; recomendações baseadas em colaboração, onde os itens recomendados são aqueles que utilizadores com preferências semelhantes preferiram no

passado; recomendações baseadas numa abordagem híbrida, que combina ambos os métodos anteriores.

Todas estas abordagens têm as suas vantagens e desvantagens, derivadas da sua natureza. Cada uma das abordagens pode, inclusive, ser progressivamente dividida em dois tipos de técnicas de recomendação para estimativa de classificação, baseando-se em heurísticas ou modelos, e por tipo de avaliação, baseado em previsão de *ranking* (classificações implícitas, como cliques em *links*) ou regressão de classificações (classificações explícitas, como escalas de classificação de 1 a 5 para filmes, por exemplo).

Depois foi efetuada uma revisão da literatura existente na área dos sistemas de navegação pedestre, para perceber as necessidades e preferências dos utilizadores nos seus caminhos.

No estudo de (Golledge, 1994) reuniram-se 32 adultos que, na primeira fase em laboratório, traçaram o que achavam ser o melhor caminho no mapa de uma zona conhecida desde um ponto A a B, e na segunda fase, no terreno, percorreram um caminho de ida e volta nessa mesma zona sem recurso do mapa. Este estudo tem como objetivo perceber como os pedestres mapeiam e desenham caminhos sem recurso a guias (mapas, GPS) e quais os critérios que os pedestres consideram quando percorrem um caminho, para além da distância.

O estudo realizado pelos autores (Borst *et al.*, 2009) analisa a influência do ambiente nos caminhos de pessoas idosas, feito com um grupo de idosos residentes de 3 distritos urbanos da mesma cidade, com criação de um modelo sobre esta influência. Era-lhes pedido que traçassem o caminho que normalmente percorrem nas suas viagens para 3 tarefas diferentes. Cada caminho traçado foi comparado com outros caminhos alternativos para o mesmo destino, construídos baseados em diferentes qualidades ao longo dos caminhos.

No estudo de (Rodríguez *et al.*, 2015) é feita uma análise sobre a influência do ambiente nos caminhos de raparigas adolescentes, dividida em dois grupos, um de uma área menos suburbana, com maior densidade populacional e mais pobreza, e outro local com condições inversas. Primeiro, os autores definiram “variáveis de qualidade” com base na literatura prévia, as quais caracterizam um caminho, dependentes de métricas que correspondem a diferentes aspetos dos caminhos. Por fim, cada segmento dos caminhos percorridos foi comparado com segmentos alternativos dos caminhos automaticamente gerados com uso de um algoritmo de procura.

Em (Czogalla & Herrman, 2017), similarmente aos dois estudos anteriores, foi criado um modelo sobre o processo de decisão de viagem pedestre, com uma introdução

do conceito de “atributo de qualidade pedestre”, ou PQA (*pedestrian quality attribute*), e a sua avaliação por fatores para análise das necessidades de qualidade de viagens pedestres. Este PQA corresponde à soma dos subatributos de qualidade de um caminho.

De acordo com estes estudos, pode-se concluir que existe uma preferência dos pedestres por caminhos mais curtos, mais esteticamente atrativos, próximos a zonas verdes, com maior segurança e menor complexidade (menor número de cruzamentos).

Seguidamente, foram analisados sistemas de recomendação de caminhos semelhantes àquele planeado para este projeto. O projeto documentado em (Fernandes, Carvalho, & Rito Lima, 2019) descreve um protótipo de um sistema de recomendação para cidades inteligentes, dependente de dados de sensores obtidos em tempo-real, com uso de uma rede de sensores sem fio (RSSF) simulada na plataforma Cupcarbon, uma plataforma *open-source* de simulação IoT, e para o qual reuniram um número de atributos do caminho que consideraram importantes para o sistema.

Em (Ramos, Trilles, Muñoz, & Huerta, 2018), é descrita uma metodologia *technology-agnostic* para recomendar caminhos livres de poluição, também constituída por uma RSSF, neste caso, de qualidade de ar. Esta metodologia envolve uma análise de vários índices de qualidade de ar, a criação de um novo índice específico ao problema, e a seleção de um modelo de interpolação para criar um conjunto de “obstáculos” que representam as áreas com pior qualidade de ar da cidade de Madrid, que o sistema ignora, desenhando caminhos que evitam estas zonas

Outro sistema apresentado em (Novack, Wang, & Zipf, 2018) foi implementado com o intuito de gerar caminhos pedestres “agradáveis” e personalizáveis pelos utilizadores. Este sistema utiliza dados do OpenStreetMap (OSM) para caracterizar cada segmento e um algoritmo de procura para gerar caminhos multiobjectivo, ao unir os atributos do caminho numa única fórmula de custo que considera os pesos de cada atributo, definidos pelo utilizador. Estes projetos também ajudaram a reunir os requisitos dos sistemas sobre as preferências dos pedestres, tal como mencionado anteriormente, nomeadamente, a iluminação, menor complexidade dos caminhos e fatores ambientais (como a qualidade de ar e temperatura).

Paralelamente, foram analisadas APIs e bibliotecas de mapas interativos para JS, já que um dos objetivos deste projeto é construir uma interface *web* para o sistema HealthyTrack. Com uso da API dos Google Maps, foram criadas umas páginas *web* com mapas, para testar o uso de ferramentas de mapas interativos na *web*. Por fim, foi decidido que seria melhor usar Leaflet para implementar as funcionalidades de mapas interativos da aplicação.

Depois disto, foi implementada a base de dados PostgreSQL HealthyTrack, com instalação da extensão PostGIS (para a gestão de dados geográficos). Selecionou-se esta base de dados por ser a solução mais popular para armazenar dados geográficos (com possibilidade de extensão com outras funcionalidades GIS). Esta base de dados contém as tabelas com os dados necessários para armazenar aspetos dos caminhos que são essenciais para as recomendações, tais como a tabela de dados de qualidade de ar e a tabela com as localizações das zonas verdes de Lisboa. Contém também a rede de estradas utilizada para criar os caminhos, que serão criados com uso dos segmentos de rua desta tabela.

O sistema de recomendação de caminhos tem por base um algoritmo de *pathfinding* (procura de caminhos), utilizado para a construção de caminhos com uso da extensão PgRouting, na base de dados. Com esta extensão, a rede de estradas pode ser transformada num grafo, para depois se fazerem *queries* de modo a obter caminhos mais curtos. Para visualizar os resultados das *queries* de caminhos o PgRouting foi instalado também no QGIS, que foi utilizado, em geral, para facilitar a gestão de dados do sistema.

Finalmente, foi implementada a aplicação *web* para o sistema HealthyTrack, baseada em NodeJS, com uso de ExpressJS que fornece o serviço de recomendação de caminhos. Com uso do Leaflet no lado cliente, os mapas interativos e resultados de pedidos de recomendação de caminhos são apresentados.

Por fim, foram realizadas umas experiências onde a funcionalidade de recomendação de caminhos pelo sistema HealthyTrack foi validada, sendo possível verificar que os caminhos mais verdes e mais limpos são normalmente distintos dos caminhos mais rápidos, priorizando os critérios de qualidade respetivos.

Para concluir e refletir sobre o possível trabalho futuro nesta área, possíveis avanços incluem melhorar as fórmulas de custo, já que estas são algo rudimentares, explorar outros fatores de qualidade de caminho, como por exemplo, níveis de iluminação das ruas (abrangido pelo critério de segurança), estética dos caminhos, fatores tais como meteorologia e hora do dia (que poderiam ser utilizados, por exemplo, de dia, para evitar exposição excessiva a raios UV, ou para evitar a chuva nos caminhos, e para priorizar níveis altos de iluminação a noite), complexidade dos caminhos (número de cruzamentos), qualidade e declive do piso (especialmente em subidas) e níveis de poluição sonora. Para além disto, também se poderia explorar a possibilidade de o utilizador adicionar múltiplas paragens a um caminho. Outro aspeto a investigar no futuro poderia ser a combinação de vários critérios numa só fórmula de custo, como foi feito no sistema implementado por (Novack *et al.*, 2018), com possibilidade de personalização por parte do utilizador dos seus caminhos. Por fim, aspetos a melhorar na interface da aplicação web ou a adicionar a futuras interfaces incluem a adição de um tutorial para uso



das funcionalidades e para contexto sobre cada caminho, e mostrar informação sobre cada caminho quando se faz *hover* ou clique por cima deste, por exemplo (informação sobre o tempo de viagem, complexidade do caminho, etc.), dando especial importância em informar sobre a vantagem de escolher esse caminho (mais limpo, mais verde, etc.).

**Palavras-chave:** sistemas de recomendação; caminhos pedestres; mapas; saúde e bem-estar; sustentabilidade



## Abstract

Currently, there exist many route recommendation systems, e.g., Waze, Google Maps, among others, which offer the user suggestions for routes between two locations, supporting several means of transportation (e.g., car, foot and public transportation).

These systems propose one or more routes often based on distance, cost and duration criteria, i.e., they recommend the fastest and/or most economic route(s) (where gas cost, toll charging and/or ticket purchasing is considered). Additionally, these systems may rely on other route characteristics, such as real-time traffic information, to improve the quality of recommendation provided to the user.

These general use route recommender systems, which are very focused on public and private transportation, are, however, limited to economic criteria, disregarding other types of preference criteria for the user.

For pedestrian courses in urban surroundings, it's common that the user's choice of route isn't the fastest/most direct/most economic, as they often prioritize other criteria, favoring routes where they are exposed to lower levels of pollution, more fresh air, vegetation and aesthetic scenery and, especially at night, less safety hazards and more lighting.

The goal of this project is to develop a route recommendation system for pedestrian courses, between two points on a map, considering multiple relevant criteria and users' preferences.

The developed system, called HealthyTrack, allows the user to request route recommendations from two points selected on the map, presenting the user with a faster route and two alternatives, a route that prioritizes air quality and another that prioritizes proximity to green spaces. This solution is focused on promoting the health of pedestrians and is based on their preferences and needs.

**Keywords:** recommendation systems; pedestrian routes; maps; health and well-being; sustainability



# Contents

List of Figures .....	xiv
Chapter 1 Introduction .....	1
1.1 Motivation .....	1
1.2 Goals .....	2
1.3 Contributions .....	2
1.4 Document Structure .....	3
Chapter 2 Concepts and Related Work .....	4
2.1 Recommender Systems .....	4
2.2 Pedestrian Navigation .....	9
2.2.1 Route Quality Criteria .....	9
2.2.2 Pedestrian Navigation Systems .....	16
2.3 Summary .....	26
Chapter 3 HealthyTrack .....	28
3.1 Data Analysis and Gathering .....	28
3.1.1 Definition of Route Quality Criteria .....	28
3.1.2 Data Gathering .....	29
3.2 Architecture .....	40
3.3 HealthyTrack Database .....	41
3.4 Routing with PgRouting .....	49
3.5 Route Recommendation Algorithm .....	52
3.6 Visualization of Data with QGIS .....	55
3.7 Web Application .....	59
3.7.1 Features .....	60
3.7.2 NodeJS Server .....	62
3.7.3 Interactive Maps with LeafletJS .....	62
3.7.4 Business Logic .....	63

3.8	Experiments .....	69
3.9	Summary.....	72
Chapter 4	Conclusion and Future Work .....	73
References	.....	76



# List of Figures

Figure 1 – The main categories of recommendation algorithms (Liu et al., 2013). .	6
Figure 2 – Round trip routes from point A to B (Golledge, 1994). .....	11
Figure 3 – Round trip routes from point X to Y (Golledge, 1994). .....	12
Figure 4 – Three examples of different walking routes (Borst et al., 2009). .....	13
Figure 5 – A representation of the technology-agnostic methodology to trace pollution-free routes. (Ramos et al., 2018). .....	17
Figure 6 – Madrid Local Air Quality Index definition (Ramos et al., 2018). .....	18
Figure 7 – Spatial distribution of air quality stations with MLAQI (Ramos et al., 2018). .....	19
Figure 8 – IDW interpolation output according to MLAQI (Ramos et al., 2018). .	20
Figure 9 – Screenshot of the developed application in (Ramos et al., 2018). .....	21
Figure 10 – A route that avoids the most polluted areas (Ramos et al., 2018). .....	21
Figure 11 – Main components of the proposed pedestrian navigation system (Novack et al., 2018). .....	23
Figure 12 – Street segment with social zones and a 50 meter buffer around it (Novack et al., 2018). .....	24
Figure 13 – Street segment with green zones within 100 m visibility radius of each observation point (Novack et al., 2018). .....	24
Figure 14 – Comparison between shortest route and alternative routes (Novack et al., 2018). .....	25
Figure 15 – Real-time air quality API services compared. ....	31
Figure 16 – Original model of air quality. ....	32
Figure 17 – Interpolated raster layer of air quality (Heatmap) in QGIS. ....	32
Figure 18 – First step of the OSM data exportation process in the CML geodata repository. ....	34
Figure 19 – First step of the OSM data exportation process with the HOT export tool. ....	35
Figure 20 – Second step of the OSM data exportation process. ....	35



Figure 21 – Third step of the OSM data exportation process. ....	36
Figure 22 – Fourth step of the OSM data exportation process. ....	36
Figure 23 – Fifth and final step of the OSM data exportation process. ....	37
Figure 24 – Original roads dataset, imported from the CML public spatial data repository, in QGIS. ....	38
Figure 25 – Green areas dataset page in the CML public spatial data repository...	38
Figure 26 – Parks and gardens dataset page in the CML public spatial data repository..	39
Figure 27 – Trees and shrubbery dataset page in the CML public spatial data repository. ....	39
Figure 28 – Green areas dataset, imported from the CML public spatial data repository, in QGIS. ....	40
Figure 29 – Architecture of the Healthy Track system. ....	41
Figure 30 – PostGIS shp2pgsql tool. ....	43
Figure 31 – Data model of HealthyTrack DB. ....	45
Figure 32 – Example of a buffer of air quality around an edge in QGIS. ....	46
Figure 33 – Example of a buffer for the green areas around an edge in QGIS. ....	48
Figure 34 – “Routable” roads (estradas_noded) table represented in QGIS. ....	51
Figure 35 – Graph with edges from a to f. The steps of the optimum path are marked in blue (Bitesize Data Science, 2019). ....	53
Figure 36 – Representation of the HealthyTrack spatial data in QGIS. ....	55
Figure 37 – Customization of the air quality layer in QGIS, with the inverted color range. ....	56
Figure 38 – Outline of the boundaries (in red) for the selected boundaries in QGIS. ....	57
Figure 39 – Zoomed in display of the selected Lisbon area for the proof-of-concept, in QGIS. ....	58
Figure 40 – Fast route in QGIS. ....	58
Figure 41 – Green route in QGIS. ....	59
Figure 42 – Clean route in QGIS. ....	59
Figure 43 – Healthy Track web application interface in the browser. ....	60

Figure 44 – Healthy Track web application interface, with the fastest route hidden under the cleanest.....	61
Figure 45 – Healthy Track web application interface, now showing the fastest route instead of the cleanest. ....	62
Figure 46 – Normal route without the previous issue, viewable in the web app. ...	64
Figure 47 – Graphical representation of a route. ....	65
Figure 48 – Graphical representation of the issue to be resolved. ....	65
Figure 49 – Route drawn in the web app. The line before the start of the route must be removed.....	66
Figure 50 – Graphical representation of the 4 Dijkstras solution. ....	66
Figure 51 – Graphical representation of the TRSP solution. ....	67
Figure 52 – First experiment results in HealthyTrack. ....	69
Figure 53 – First experiment results in Google Maps.....	70
Figure 54 – First experiment results in OSM, using OSRM.....	70
Figure 49 – Fastest to cleanest (from left to right) routes from the first experiment on HealthyTrack.....	71
Figure 56 – Second experiment results in HealthyTrack .....	71
Figure 57 – Fastest to cleanest (from left to right) routes from the second experiment on HealthyTrack.....	71
Figure 58 – Second experiment results in Google Maps. ....	72
Figure 59 – Second experiment results in OSM, using OSRM. ....	72



# Chapter 1

## Introduction

### 1.1 Motivation

Nowadays, when we look at the market of route recommender systems (for pedestrians and more), most of the dominant systems we see, such as Google Maps, base the cost of a trip on economic criteria (such as distance and time of travel).

These systems are, however, limited when it comes to factors of the real world and how they influence users' potential preferences, especially when it comes to pedestrians. Part of this is due to subjective preferences that the users have, which are based on perceived characteristics of the world around them (e.g.: aesthetic aspects of their surroundings, air quality, etc.) and also on their own physical and psychological condition (Alfonzo, 2005; Armeni & Chorianopoulos, 2013; Borst *et al.*, 2009; Giles-Corti & Donovan, 2003; Golledge, 1994; Guo & Loo, 2013; Hahm, Yoon, Jung, & Kwon, 2017; Rodríguez *et al.*, 2015; Seneviratne & Morrall, 1985; Weinstein Agrawal, Schlossberg, & Irvin, 2008).

On the other hand, there are growing concerns about the individual's effect on the environment and how it affects us, which leads to a growing focus on sustainability and personal health. This is due to the collective conscience of the sedentary lifestyle that most of the world's societies lead, which is spreading as the world becomes more dominated by technology, and global warming, which is beginning to bring possibly irreversible changes to the world as we know it. Thereby, the focus of this work will be on pedestrian recommendation systems, in the sense of evolving them to be more directed at the user and more centered on their health, and in this way, to possibly motivate the public to travel more on foot.

According to the literature, at least up until mid-2012, most of the studies on recommender systems, which include practical implementation projects of functional platforms, are focused on movies and shopping (Park, Kim, Choi, & Kim, 2012). It's easy to grasp the reasons for the popularity and investigation of these kinds of systems, with the many examples we see of these on the market, like Netflix and Amazon.

In addition to that, the technological advances made until today facilitate the gathering and processing of information, which allows for more precise recommendations than ever before, with increasing processing power and better hardware to measure and compute more data in less time. This puts us in a context where route recommender systems can, more and more, offer its users options that better match their necessities and expectations. Right now, route recommender systems that are based on these considerations are being further proposed and implemented (Chakraborty, 2012; Fernandes *et al.*, 2019; Novack *et al.*, 2018; Su *et al.*, 2014; Tsai & Chung, 2012).

## 1.2 Goals

In this project, the aim is to develop and evaluate a route recommendation system for pedestrian routes, named HealthyTrack, that, in addition to the distance and travel time between two points, takes other characteristics (according to the needs and desires of the user) of the surroundings into account, to generate routes that match other criteria of preference and relevance to the user.

Among these criteria, we are especially focused, at this initial stage, on aspects such as the proximity to green areas and air quality on the routes. The goal is to promote the users' psychological and physical health, so that they feel more capable of moving in a more sustainable way that respects the environment.

## 1.3 Contributions

The main contributions of this thesis include:

A review of investigation and state of the art of recommender systems, in specific, route recommender systems, and of the existing literature on pedestrian navigation, allowing us to understand the concepts in these areas, a background on the preferences of pedestrians for the development of criteria for route recommendation and a discussion of different implementations of route recommender / navigation systems presented in articles.

The HealthyTrack database, a PostgreSQL DB for storing the spatial data necessary to measure different aspects of the routes, values which are then used in the calculation of a recommended route, with basis on the route criteria selected, such as road length or air quality of the surroundings. This DB contains, above all, the road network, average air quality values per each 5 meters and green areas of the city of Lisbon, in addition to other data.

The HealthyTrack web app, a NodeJS based web application that is connected to the PostgreSQL DB previously mentioned and provides an interface with an interactive map,

which uses the LeafletJS <sup>1</sup> (interactive maps library for Javascript) service, where the user can input the origin and destination of a route and then get the output recommended routes for each criterion (fastest, cleanest and greenest) drawn on the map.

## 1.4 Document Structure

This document is divided in 5 chapters. In the present chapter I present the motivations behind the project of this dissertation and the main goals it aims to achieve.

In the next chapter, I analyze the context of the project, including an introduction on the taxonomy of recommender systems and a review of the state of the art, an overview of literature surrounding pedestrian navigation, including the preferences of pedestrians, and an analysis of developed route recommenders that intend to solve similar problems.

In the third chapter, the system developed is presented in further detail, The features of the system and the architecture including the web application that was developed to allow for interaction with the system, and other aspects, such as the selection of criteria that classify a route for this project, are discussed.

This chapter is followed by the conclusions chapter which contains a discussion of what was achieved in this project and possible improvements and additions for future work.

---

<sup>1</sup> <https://leafletjs.com/>

# Chapter 2

## Concepts and Related Work

In this chapter, in section 2.1 , I explore the current context of the recommender systems area and its taxonomy. Then, in 2.2 , the focus is on pedestrian navigation, where I first look at the pedestrians' preferences for their routes, and then analyze systems that are similar to the one intended to be developed in this project.

### 2.1 Recommender Systems

In this section I introduce the area of recommendation systems, listing the types of systems that exist and the current research.

According to (Adomavicius & Tuzhilin, 2005), the concept of recommendation system, in terms of area of investigation, emerged in the mid-1990s, when researchers began to focus on analyzing recommendation problems that depend on classification structures. These problems boil down to the task of estimating ratings for items (products, services) that haven't yet been viewed by an individual.

This estimation task is based on the concept of item utility for that individual. Systems may differ in if utility is user-defined, for example, there are systems where users can rank items, and others where ranking is system-defined with the use of an arbitrary function (e.g., a profit-utility function).

Once the ratings for other items have been estimated, based on the ratings of items already viewed by the customer, they can be recommended based on that rating, which can be done in a number of different ways.

Recommendation systems are divided into three different types, according to their recommendation approach: Content-based recommendations, where the items recommended to the user are like the ones that the user preferred in the past; Collaborative recommendations, where the items recommended to the user are the same that people with similar preferences have preferred in the past; Hybrid approach, which is the combination of both collaborative and content-based methods.

Each of these approaches can also be divided into two general classes of techniques for estimating rankings: heuristic-based and model-based, and, according to (Liu, Ma, Chen, & Xiong, 2013), they can also be divided by evaluation type: ranking prediction or ranking regression.

The main categories of recommendation algorithms, in regard to their solution type, evaluation type and the information that is exploited are displayed on a diagram in Figure 1.

To elaborate on the types of evaluation, essentially, the one based on ratings regression involves explicit ratings (e.g.: 1-5 scale in film applications), and the one based on ranking prediction involves implicit ratings (e.g., click on the recommended page link).

Of these systems, those based on ranking regression have had extensive investigation done on them, due to the extent of existing applications that are ranking-based, public datasets, etc. Their quality is evaluated by the approximation of their predicted ratings to the actual ones. To make this assessment, metrics such as mean absolute error (MAE), which measures the mean absolute deviation between predicted and actual classification, and root mean square error (RMSE), for possible larger margins of error, are used. As for those based on ranking prediction, the focus is on figuring out where the user will want to click and sorting those items in order of importance to be recommended.

As for the limitations of each, starting with ratings-based ones, these systems usually assume that users will be exposed to all content, when in fact this rarely happens. On the other hand, the way to deal with ranking predictions could be alternative, that is, instead of focusing on recommending only what ranks best, content could be presented more attractively to avoid low ratings. For ranking-based systems, the biggest limitation is the assumption that a person's click on an item means that similar items should be recommended to them, which is not always true.

Returning to (Adomavicius & Tuzhilin, 2005), for all the recommendation approaches mentioned, there is a set of characteristics and limitations that distinguish them from each other, that make them more suitable for certain problems and situations, and that make them limited in other contexts. A more detailed analysis of each of these approaches follows.

First, content-based recommendation systems are characterized by the focus on the recommendation of items that contain textual information, such as documents, websites and Unix User Network (Usenet) messages, due to the significant and anticipated advances made by the information gathering and filtering communities, and the importance of text-based applications, as indicated in the referenced article.

As already mentioned, in these systems, the utility  $u(c,s)$  of item  $s$  to user  $c$  is estimated based on the utilities  $u(c,s_i)$  assigned by user  $c$ , to items  $s_i$ , which are like item  $s$  in terms of content.



To characterize a particular item, its content, that is, the set of attributes that characterize that item, must be selected. This content usually exists in the form of keywords, when considering text-based items. The importance of a word is defined according to a weighing measure, which can be defined using various methods, one of the best-known examples of these being the Term Frequency/Inverse document frequency (TF-IDF) heuristic method.

After characterizing the items, they are compared with those associated with the user's profile, which the user preferred, of which the most similar are recommended. Items preferred by the user are stored in the user profile, which thus characterizes their preferences.

Both the user profile as well as the content of an item can be represented as vectors of keyword weights. Furthermore, the utility function  $u(c,s)$  is usually represented by a heuristic defined using vectors, one corresponding to the user profile and another to item content.

In addition to traditional heuristics, according to (Adomavicius & Tuzhilin, 2005), there are also techniques built on models that are based on statistical data, such as Bayesian classifiers and machine learning techniques which include clustering, decision trees and neural networks.

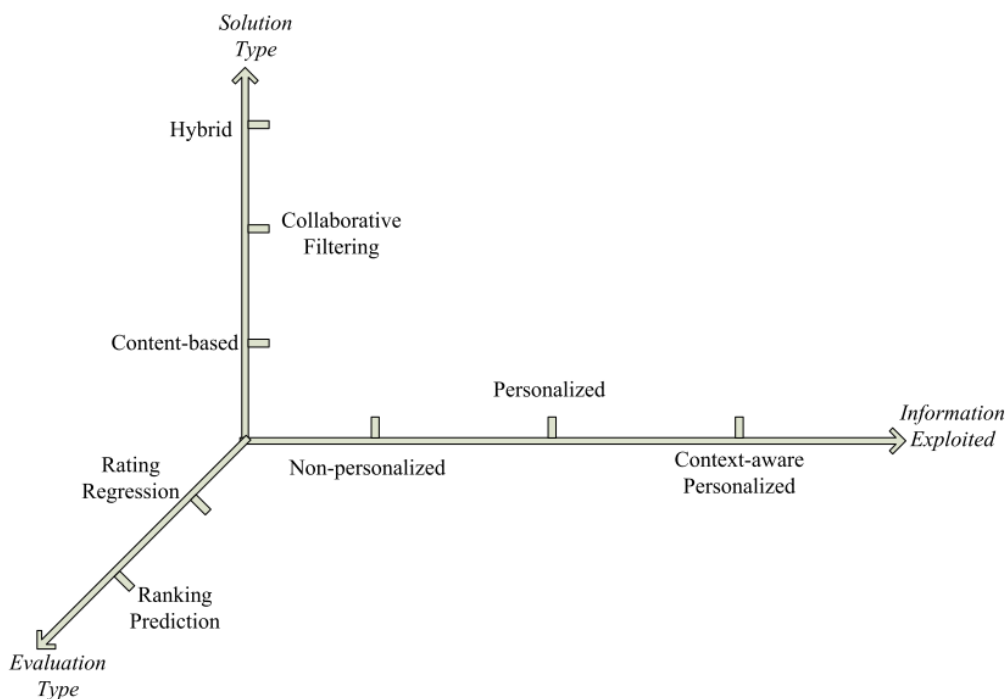


Figure 1 – The main categories of recommendation algorithms (Liu et al., 2013).

Content-based recommendation systems have three main limitations.

First, content analysis is limited, so since a lot of content must be processed automatically, it must be easily processable, usually in text format. Thus, content-based recommendations are not the most suitable for multimedia data, and on the other hand, different articles with the same characteristics are indistinguishable.

Second, like mentioned in (Adomavicius & Tuzhilin, 2005), there is the possibility of over-specialization, that is, that the recommendations are limited to articles with similar characteristics to those previously liked. This can be solved by introducing randomization into the recommendation process. Proposals to solve this problem include the introduction of genetic algorithms for filtering information. Some recommendation systems also contain a filtering process for articles that are too similar to what the user liked.

Finally, there is also the problem of new users, because before the system understands the user's preferences to provide accurate recommendations, it must rate enough items.

As for collaborative recommendation systems, to elaborate in their definition, the utility  $u(c,s)$  of item  $s$  for user  $c$  is estimated based on the utilities  $u(c_j,s)$  defined by other users like user  $c$ . A relevant example of one of these systems is Amazon's book recommendation system.

For this type of recommendation, regarding the techniques based on heuristics, the classification of item  $s$  by user  $c$  is calculated as the aggregate of classifications that the  $N$  most similar users to  $c$  gave to that same item. The aggregation function can take several forms, and it can be as simple as a simple averaging function, but it usually takes the form of a weighted sum, with weights obtained through a heuristic function, that vary depending on the similarity that users share.

As for the methods to calculate this similarity, the most popular are the nearest neighbor (correlation and cosine-based). Typically, these methods use a list of all items co-ranked by both users in question and check the intersection between both users in terms of equal ranked items.

Moving on to model-based methods, an example of a proposal that applies probabilities to collaborative filtering suggests a formula for calculating rankings that contains a probability expression to calculate the probability that user  $c$  will give a certain rank to item  $s$ , depending on the ratings that they have given to previous items.

To estimate this probability, two alternative probabilistic models can be used: cluster models and Bayesian networks.

In the first model, similar users are grouped (clustered) into classes. User classifications are independent of their class, i.e., the model structure is based on the naive Bayesian model. The number of classes and parameters are taken from the data.

The second model represents each item in the domain as a node in a Bayesian network, where the states of each node correspond to the possible classifications for each item. The network structure and conditional probabilities are obtained from the data. A limitation of this model is that each user can be grouped into a single cluster, and some recommendation applications benefit from grouping each user into more than one cluster.

Other widely used techniques include neural networks, linear regression and probabilistic models.

Collaborative recommendation systems don't share some of the shortcomings of content-based ones. By dealing only with user recommendations, they are not dependent on the type of content, being able to recommend any type of items.

A limitation of these systems is also the problem of the new user, since at the beginning the system does not know what the user's interests are due to the lack of classifications. Proposals to solve this problem involve hybrid approaches, and others propose the use of various techniques to determine the best items for the user to rank, based on popularity, entropy, user profile, etc.

Another problem occurs when introducing new items, because until an item has been rated by a significant number of users, the system cannot recommend it. This problem can also be solved using hybrid approaches.

A further limitation of these systems is due to the scarcity of ratings given versus the number of ratings that must be predicted. Therefore, the forecast must be as accurate as possible, despite the scarcity of material. In addition to this, the number of users must also be significant for the system to succeed. What can happen is that certain items that have not been rated as often are less recommended, and users with more particular preferences may not be satisfied with predictions that apply to the majority.

One way to solve this is to apply an extension of filtering techniques called "demographic filtering", where user profiles are added, to assess similarities more accurately. Other solutions involve using algorithms to verify similarities and methods and matrix factorization to reduce scarcity.

Finally, as for hybrid recommendation systems, they involve a combination of content-based and collaborative approaches, in order to resolve the limitations of both.

There are 4 different ways to combine these two approaches: implementing them separately and combining predictions (using a linear combination of predicted rankings

or a voting scheme); incorporating content-based features in a collaborative approach; incorporating collaborative features in a content-based approach; and building a unifying model that incorporates features of both.

These systems can also be enhanced by knowledge-based techniques to improve the accuracy of recommendations and resolve limitations of traditional systems. For example, the Entree system, presented in (Burke, 2013), uses domain knowledge about restaurants, cuisine, and foods to recommend restaurants to users (for example, it knows that seafood is not vegetarian). The problem with these systems is that they require previous knowledge, and so they are built only for domains where it is vast and available in a readable format.

To conclude this section, several articles compare hybrid to traditional methods, showing that hybrid methods produce more accurate recommendations.

In the case of the HealthyTrack system developed for this project, it falls into the category of content-based recommendations since it recommends routes to the user according to their preferences.

## **2.2 Pedestrian Navigation**

In this section, some works from the existing literature in the field of pedestrian navigation are presented where I explored the concepts of pedestrian route quality, that is, the characteristics that constitute a quality route. I also study some implementations of existing navigation systems, where additional quality criteria for route recommendation is considered. These implementations resemble what is intended to be the final product of this dissertation project.

### **2.2.1 Route Quality Criteria**

There is extensive research in this area that demonstrates and justifies the search for better recommendations of walking routes, as pedestrians are much more exposed to the outdoors and suffer greater restrictions of time, effort, etc. than they would in vehicle transport.

(Golledge, 1994) argues that the route selection problem is not limited to finding the lowest cost route. He reviews the existing literature in the field of cognitive mapping and cognitive distance and comes up with new criteria for selecting a pedestrian route in addition to the traditional ones, and to confirm his theory, he presents a study where participants traced routes on maps and also completed routes on foot.

In this study, the sample of participants consisted of 32 adults (16 females and 16 males), mostly students, aged between 20 and 35, of which 50% had good knowledge of geography.

In the first experiment of this study, as a first task, participants had to choose one of three routes drawn on a rectangular grid for a series of maps, imagining that they were making the journey from home to work or vice versa. In this test they were shown three types of routes, one in L-shape, with the longest part at the beginning of the route, the other as well, but with the longest part at the end, and another one approximating a diagonal route to the destination (to simulate a more direct route). All routes were the same distance.

The second task involved trip chaining (route with multiple stops), and for this one the grid was changed to allow for some diagonal linkages. The possibility of trip chaining isn't explored on this thesis but could be a pursuit in future works.

The third task of this experiment involved changing the rectangular grid to include curved roads and blocks at non-orthogonal intersections. Polygons representing objects with positive or negative attraction (e.g.: parks or dumps) were also placed on the map.

After each session, suggestions were collected to find out what criteria the study subjects considered when choosing their route, and if these applied in real life.

According to the results of the first experiment, the criteria most used by the participants were sorted from most used to least used, in the following order: shortest distance, less time, fewest turns, most scenic/aesthetic, first noticed, longest leg first, many curves, many turns, different from the previous route, shortest leg first.

In the second experiment, the study was conducted in the field. Participants were asked to choose a round-trip route between two points, with a different origin and destination each time. All participants knew the area well. Routes that conformed to the criteria most used in the first phase of the study were created, and these were compared to the actual routes travelled. Each participant was taken to the starting point for each route and given the destination, and each round trip was recorded. After this, the participants answered a questionnaire to find out which criteria they considered when selecting their route, among other aspects.

On average, the route criteria considered the most important were "shortest route" (as in perceived shortest route), "route taking the least time" and "route proceeding in the direction of destination", followed by "fewest turns", "first noticed", and "usual route".

Finally, when comparing the results of the first (laboratory) and second (real environment) experiments of the study, contrary to what most participants stated in the first phase, the issue of time minimization in a route choice is very important, according to the results of the second experiment. On route A to B (Figure 2) 62.9% of the participants made the same route on the return trip, while on route X to Y (Figure 3), only 15.6% repeated the route on the way back. According to (Golledge, 1994), it is easy to observe that in the first route, the fact that many have repeated the route is to minimize travel time/distance/effort. As for the second, he believes that additional criteria for choosing the route might have been significantly relevant in the change of route, as well as the fact that the participants have a different perspective on the route when they turn back.

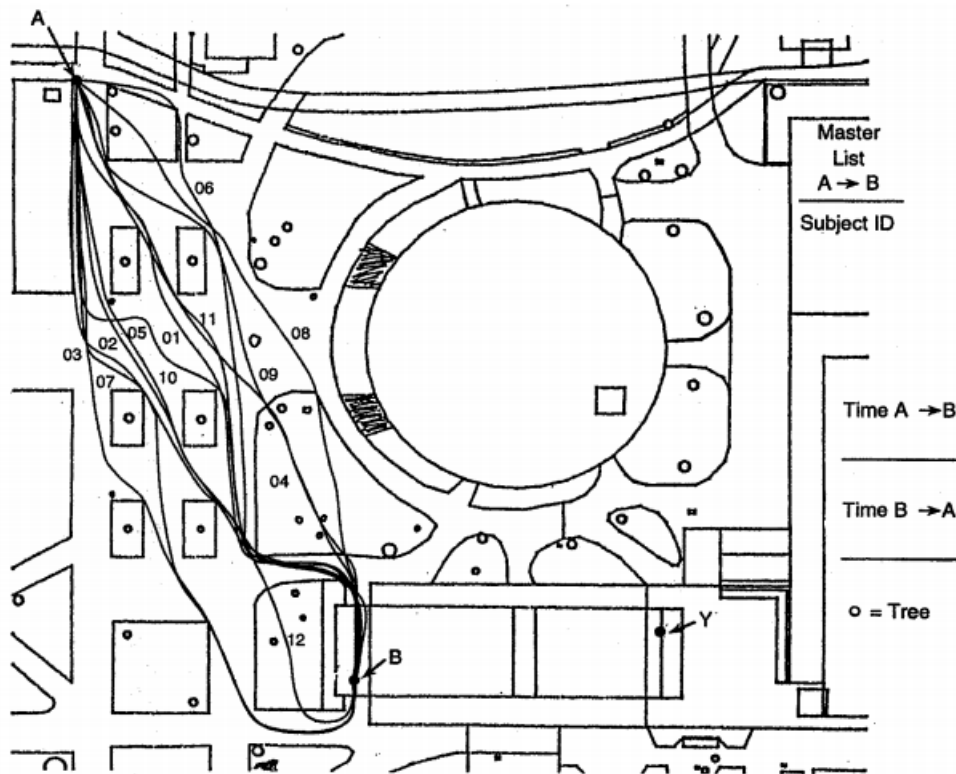


Figure 2 – Round trip routes from point A to B (Golledge, 1994).

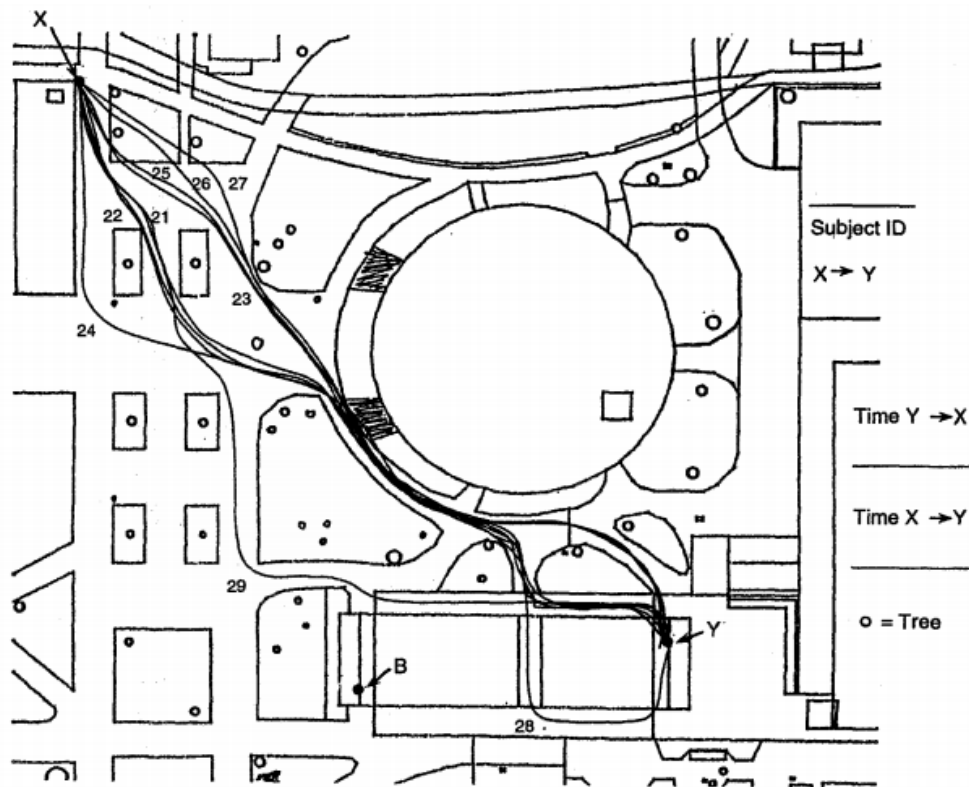


Figure 3 – Round trip routes from point X to Y (Golledge, 1994).

Another study was carried out on the influence of the environment on the routes of elderly people (Borst *et al.*, 2009). The authors of this article have also created a model of this influence.

Study participants were 364 independent seniors between the ages of 55 and 80 from three urban districts of the city of Schiedam in the Netherlands, all similar areas (with good accessibility for older people). After being selected, they received a diary with three copies of their district on a detailed map and were asked to draw the route they normally take, and the reasons for doing so, in the following situations: 1) going shopping, 2) going to the hospital or equivalent, and 3) visiting family and/or acquaintances.

Using digital city maps, a network of all possible routes was built into a GIS DB and then (valid) routes drawn by the elderly were also entered into it.

Data about characteristics (total of 23, including, for example, existence of sidewalks, declines, litter, place of passage, density of buildings, etc.) of each segment was also added to the DB (some obtained through field analysis by experts, others through external DBs with prior information about these). In addition to this, data on the number of robberies per 100 residents of each neighborhood were also used.

In Figure 4, we can see three examples of the travelled routes. In each example, the reported route is shown in panel ‘a’, the shortest route between the origin and destination is shown in panel ‘b’, and the route predicted by the route choice model (developed by the authors to predict the best route for the participants based on findings) is shown in panel ‘c’. The predicted and reported routes are very similar.

This study showed that the elderly rarely select the route with the shortest distance to destination and prefer characteristics such as: fewer turns and the existence of sidewalks, gardens, houses and shops. A curious result demonstrates that they prefer busy streets with lots of traffic, perhaps due to other reasons particular to this specific area (busy streets in this area are usually also the streets that contain other characteristics that seniors value), the easy recognition of these areas and less isolation. On the other hand, they avoid going through areas with high amounts of litter, areas that are partially surrounded by walls with no windows (according to the authors, it may be because urban areas portraying a sense of enclosure or concealment are perceived as unsafe) and areas that have stairs and/or slopes.



Figure 4 – Three examples of different walking routes (Borst *et al.*, 2009).

In (Rodríguez *et al.*, 2015), we have a study on the influence of the environment on the routes of adolescent girls.



This study was divided into two, with identical methodologies, so they are discussed as one in this report. In the first part, the routes of female students in San Diego were studied, and in the second, it was the routes of female students in Minnesota. Whatever differences constituted between these two groups have to do with the type of area studied. The specific areas of San Diego studied are generally less suburban with higher population density and a higher proportion of poverty than the areas studied in Minnesota, which are essentially the opposite of the areas of San Diego, with low population density and low levels of poverty.

First, the authors identified 4 key conceptual built environment domains: Aesthetics, Destinations, Functionality and Safety. Each of these domains depended on different metrics that correspond to aspects of the areas where the routes are traversed, metrics that were also defined by the authors based on previous literature and their expectations.

During the study, GPS devices, accelerometers and travel diaries (where participants report the details of their routes, such as name, address, type of destination, e.g.: home, school, etc.) were used. For a trip, it is assumed that each individual selects the route that maximizes its utility. This utility is expressed as a function of the observable characteristics of the route (represented by variables such as safety, security, aesthetics, etc.) and a random component. The authors base this notion on random utility theory, which proposes that people generally make choices according to their preferences, and the situations where they don't are caused by random factors.

The authors considered that a valid trip corresponds, among other details, to at least 5 minutes of activity. To check if the trip is valid, the accelerometer, GPS and trip diaries are used together. A trip is finished after 10 minutes of no activity.

When identified, the routes were taken from the GPS data and linked by hand to the city's road network on the maps. Alternative routes were generated, using a heuristic branch and bound algorithm proposed in (Prato & Bekhor, 2006). These routes were analyzed segment by segment, to verify if alternative segments as well as the total route were viable routes (that is, they are not too long, they do not make too many detours, they do not lead to dead ends, etc.).

Then, the routes travelled by the participants were compared with those automatically generated, in terms of route characteristics. For all valid segments on the routes (travelled and generated) these characteristics were examined with an audit instrument. Pairs of trained raters examined all segments twice in the same day, at

different times, during daylight hours and good weather, and then discussed the results. The raters searched for the existence of attributes that compose a measure for a domain (e.g., the existence or not of sidewalks in the segment influences the functionality of the route, as well as the existence of green routes and the quality of the soil of these routes).

As for the conclusions of this study: although the shorter distance is the decisive factor in choosing a route, it was demonstrated that the existence of green routes and sidewalks (associated with functionality), greater safety (which the authors consider to be influenced by the existence of green routes, traffic lights, lighting, pedestrian crossings etc.), and the existence of points of interest (e.g.: known restaurants, cultural sites) along the route were all positively associated with the choice of route, being factors that significantly influenced the routes of the participants in this study.

(Czogalla & Herrman, 2017) introduced a model of the pedestrian travel decision process, introducing the concept of pedestrian quality attribute (PQA) and its assessment by factors (safety, accessibility, sidewalk width, etc.) for analysis of the quality needs of pedestrian trips.

Essentially, the PQA is defined as the weighted sum of the quality sub-attributes. These are measured in the categories of safety, (defined by the existence of areas of crossings, traffic lights, density and average traffic speed), accessibility (defined by the existence and width of sidewalks, according to the limits defined by the authors, and inclination of slopes), attractiveness (how open the space is and how well-lit it is) and comfort (noise pollution and density of vegetation and shadows). Each quality sub-attribute (of values between -1 and 1) has its relative weight in this sum (weight coefficient of values between 0 and 1). All factors of these attributes are directly measured.

The following is the PQA formula:

$$PQA = q_P = a_S q_S + a_{Ac} q_{Ac} + a_{At} q_{At} + a_C q_C$$

Where  $q_P$  corresponds to pedestrian quality,  $q_S$  to safety,  $q_{Ac}$  to Accessibility,  $q_{At}$  to Attractiveness, and  $q_C$  to comfort, and all the rest correspond to the weight coefficients of each quality attribute.

In addition to these, an extra attribute was defined, for when there were significant deviations from the route not justified by any of the other attributes. This is the walkability attribute, with a value between -1 and 1, which is measured using the social factor (safe zone or high crime risk zone) and added to the PQA.

After defining these pre-conditions, route choices are made as follows: (1) Given the map network, the virtual distance is measured according to the PQA for each link or segment; (2) Given the virtual distance of each network connection, the cost of walking is associated depending on the virtual or spatial distance, depending on time constraints and other individual preferences; (3) The route is determined as a result of utility maximization, by minimizing costs across all routes, using Dijkstra's algorithm.

### **2.2.2 Pedestrian Navigation Systems**

(Fernandes *et al.*, 2019) present a pedestrian route recommendation system for smart cities, which considers objective functions dependent on user preferences. The system takes advantage of data collected through a wireless sensor network or WSN (since it is implemented for the urban context of smart cities), which was simulated using the CupCarbon simulation platform.

In this project, the authors consider that regular navigation systems are incomplete when it comes to considering user's preferences and characteristics of the environment to recommend routes, citing the existing literature, which they studied to learn about which other attributes pedestrians value when travelling.

In addition, the objective was to build a system that would work in smart cities, and as such, the authors envisioned an environment with a network of sensors that measured aspects of the terrain in real time.

After having gathered environment attributes that are considered important to pedestrians, according to the existing literature, they chose which attributes they could implement, as requirements of the system, and divided them into dynamic attributes (attributes that vary in real time), which include Wi-Fi coverage (a WSN requirement), temperature and lighting and static attributes, including route length, number of sensors (also a WSN requirement), if a route is circular or not (possibility to start and finish the route at the same spot), existence of open spaces and tourist POIs (points of interest).

The authors also document the use of the open-source platform CupCarbon, an IoT simulator, which allows you to visualize, compile and validate monitoring algorithms based on WSNs, as well as the creation of environmental scenarios and the storage of the results obtained for further analysis, presenting an interface for the simulation of maps and for interacting with them, for the placement of sensors and the creation of routes.

The system developed, as already mentioned, benefits from data obtained by sensors from the WSN, as well as user data. Using these, it automatically selects the route

that, according to a variety of attributes (e.g.: lighting, number of green areas, Wi-Fi access and quality, etc.) and considering what the user values (and the score they give to each valued attribute), most matches them.

In (Ramos *et al.*, 2018) a technology-agnostic methodology to recommend pollution-free routes, based on an air quality WSN, similarly to the solution in (Fernandes *et al.*, 2019), is presented. Their methodology works on two environments, physical (realm of the air quality sensors) and cyber (data, service and application layers). They used the ArcGIS services for most of the features of their system and the ArcGIS API for JS for their web application.

To elaborate on each of the cyber environment layers, as indicated in Figure 5, according to (Ramos *et al.*, 2018), the data layer connects with the physical environment to obtain the observations from an air quality network, using an ad hoc connector for each station. These connectors are used to format a set of measurements to be used by the service layer; the service layer has multiple components, such as the Air Quality Index (AQI) model component, which builds an index, defined by some pollutants, to provide for the Spatial Interpolation Model component, which, in turn, is responsible for generating a 2D surface by estimating the values of unsampled points from the values of sampled points. Then, on the top of the service layer, the map layer and the routing service do the routing operations and, for instance, generate alerts when entering areas of high pollution.

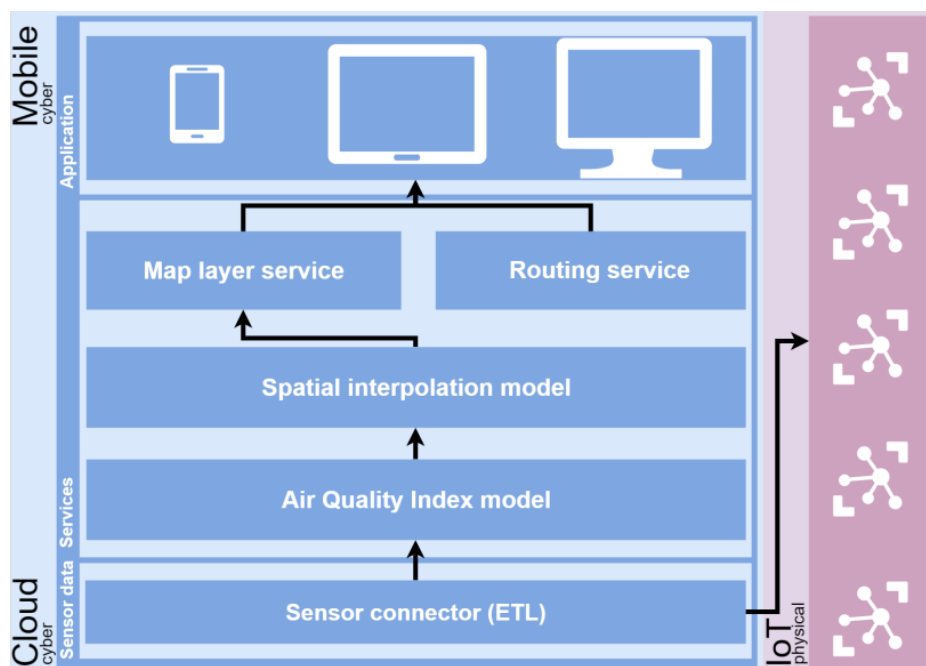


Figure 5 – A representation of the technology-agnostic methodology to trace pollution-free routes. (Ramos *et al.*, 2018).

The authors developed a proof-of-concept for this methodology, selecting the city of Madrid as a scenario for testing. Because they would be receiving hourly (air quality) data in the form of pollutant concentrations in the air, obtained from multiple air quality sensors in Madrid (distributed throughout the city as shown in Figure 7), they needed to utilize an AQI to categorize and normalize the air quality observations they obtained, which could be interpolated for all of Madrid (to create a map service with different zones based on air quality), so they start by conducting a study on the multiple AQIs developed by governmental agencies and by independent researchers (comparing their features), and finally end up developing a modified AQI, called Madrid Local AQI (MLAQI), for which its definition (with the color coding for the respective categories of air quality and detailed pollutant limit values) can be viewed in the table presented in Figure 6.

Index Range	Index Category	Color	Auxiliary Pollutant			
			Core Pollutant	NO <sub>2</sub>	O <sub>3</sub>	PM10
0–50	Good	Green	0–100	0–90	0–50	0–30
51–100	Acceptable	Yellow	101–200	91–180	51–90	31–55
101–150	Poor	Orange	201–300	181–240	91–150	56–90
>150	Very Poor	Red	>300	>240	>150	>90

Figure 6 – Madrid Local Air Quality Index definition (Ramos et al., 2018).

This index is based on the Madrid (MAQI) and Common (CAQI) AQIs and applied to the local situation of Madrid, allowing for the data extraction from most of the quality sensors in the city and the calculation of an index for each, being that these indexes are then interpolated to generate the final map layer. The pollutants considered in this index are  $NO_2$ ,  $O_3$ , PM10 and PM2.5.

The authors then explored the available data for errors distribution and outliers, by selecting the historical data from all the sensor stations in Madrid (Figure 7) for a given year (2017) and testing it for diurnal air quality consistency, comparing the air quality values collected at 7:00, 13:00 and 19:00 for a specific day, being that the 12 or 13 of that month, over several months, and seasonal consistency, selecting the middle months of every season (January, April, July and October), before selecting an interpolation method.

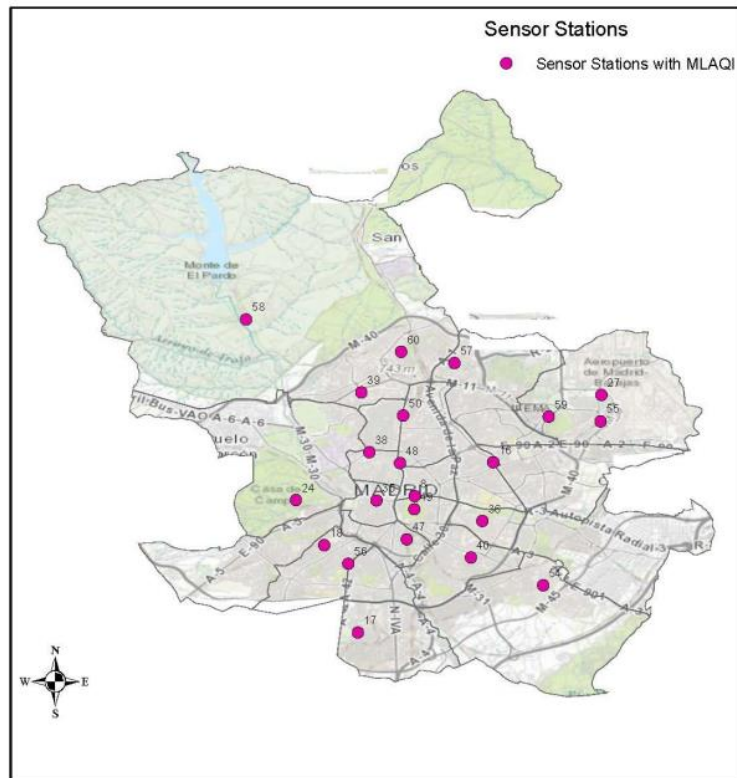


Figure 7 – Spatial distribution of air quality stations with MLAQI (Ramos et al., 2018).

After this, they analyzed the data structure for modelling with both the interpolation methods of inverse distance weighted (IDW) structural modelling and Variogram structural modelling. They ended up deciding that the IDW method fit their defined use case better, based on the consistency of results in producing the highest frequency in obtaining the lowest magnitude root mean square prediction error, which means that the air quality predictions this method provided were the most accurate out of the two options. So, this ended up being selected as the method to be applied to produce the predicted air quality values for the system as new data arrives. The output of the IDW interpolation (according to the MLAQI definition in Figure 6) can be seen in Figure 8.

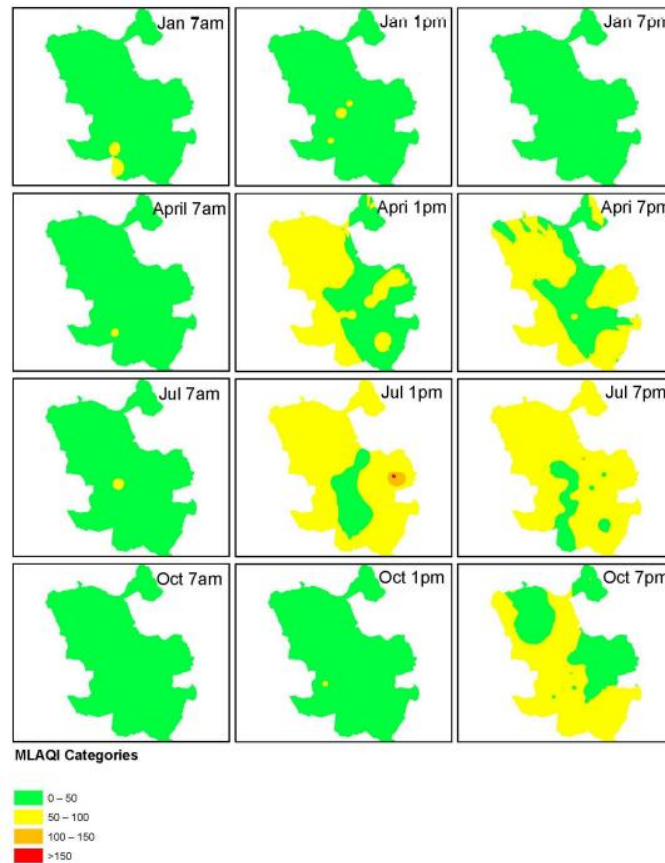


Figure 8 – IDW interpolation output according to MLAQI (Ramos et al., 2018).

Then, the map processing service takes the results of the IDW interpolation and creates a polygon map layer based on these, generating 4 air quality polygon feature classes, based on the MLAQI index, for all of Madrid, dividing the city in regions by air quality class. After publishing this web map layer, these polygon areas can be used as “barriers” in the generation of routes across the city. When a highly polluted area is defined as a barrier, all intersecting streets with areas (polygons) of the same class are excluded from the routes, with the guarantee of a pollution-free route. They characterize areas with poor to very poor air quality as highly polluted areas. As for the routing task, it uses the routing layer and defines route parameters like stops, polygon barriers and impedance (kilometers). A snippet of the resulting application is shown in Figure 9 and an example of a recommended route by the system, which avoids polluted areas, can be viewed in Figure 10.



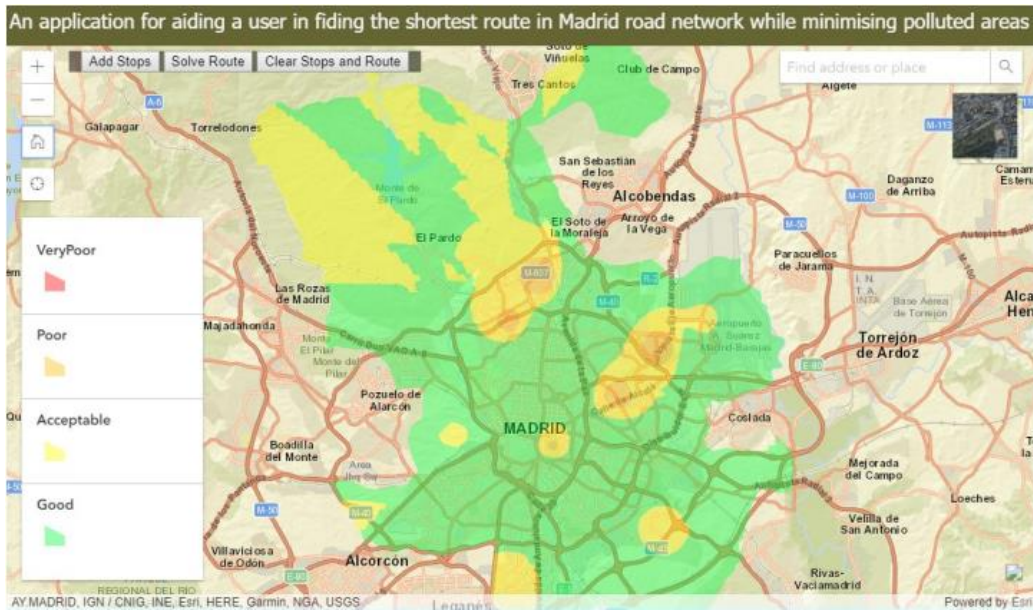


Figure 9 – Screenshot of the developed application in (Ramos et al., 2018).

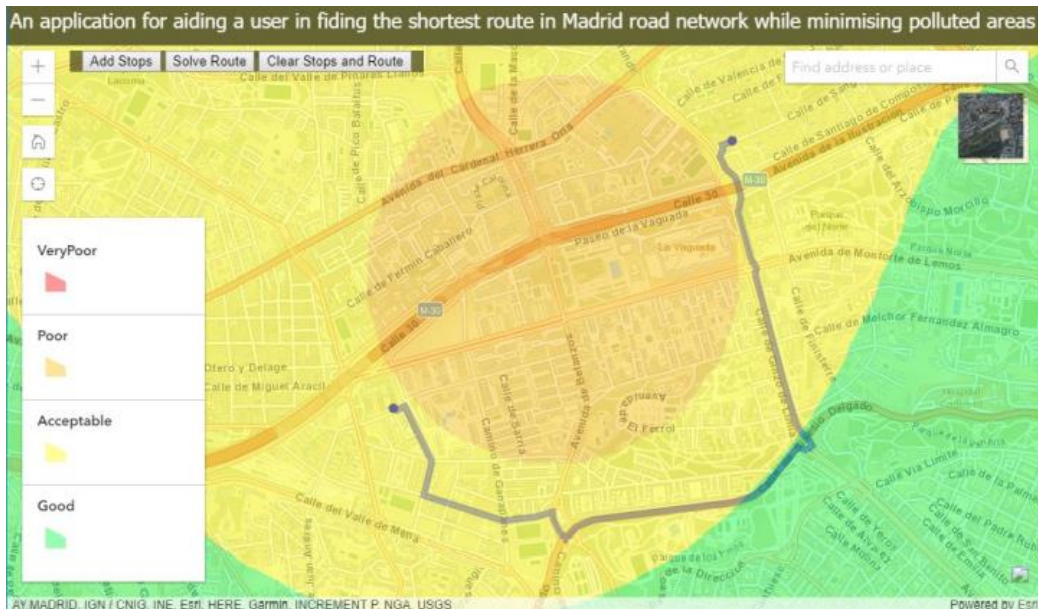


Figure 10 – A route that avoids the most polluted areas (Ramos et al., 2018).

In (Novack et al., 2018), a system that generates pleasant pedestrian routes customized to the user's preferences (with the use of open-source data from OSM<sup>2</sup>) is presented and documented.

This system allows its users to define how much they want to travel through green areas, social places and/or quieter streets. The levels of each of the three criteria that customize a route, greenness (corresponding to green areas), sociability (corresponding

<sup>2</sup> <https://www.openstreetmap.org>



to social spaces) and quietness (corresponding to quiet places) were defined and extracted from OSM, and later integrated into a cost function.

Before coming up with these route customization factors, the authors studied the existing literature around pedestrian navigation, on the criteria that influence pedestrian route quality. They paid special attention to the following criteria: safety, affected by factors such as lighting and sidewalk widths, etc.; ease of recognition, influenced by the complexity of the route, by aspects such as the existence of easily recognizable places (points of interest); how pleasant the route is, based especially on social factors (level of crime, the type of points of interest, aesthetics of the landscape, etc.); they also noted the influence of environmental factors on pedestrian routes.

The authors justify their choice of greenness, sociability and quietness factors based on the literature. They explain that the existence of green areas encourages physical activity, promotes physical and psychological comfort, and mitigates air and noise pollution. Social spaces are also associated with physical and psychological benefits, and there is a positive association to the proximity of points of interest, like services (bars, restaurants, etc.). As for quietness, this, like the contact with green areas, also promotes physical and psychological comfort, as in general it will also lead to routes that go through green areas or areas with less traffic.

They emphasize, however, that the three factors are often interdependent, namely: streets with more traffic generally have more social destinations; green areas such as parks and squares are also, at certain times of the day, social zones; safety and aesthetics of structures is generally associated with streets where there is an abundance of social and service points of interest.

After that, the authors explain how navigation systems operate, being based on graphs with nodes that represent intersections and extremities (edges) that represent street segments. Edges are often associated with “weights” that represent the street segment length and speed limits.

In the proposed system, the weight of each edge  $i$  is a function of the  $x$  variables, each one representing one of the 4 factors that can influence the route (length, greenness, sociability and quietness of the segment). Each of these is normalized by dividing each of its values by the maximum value observed in the field. Then the normalized value  $\hat{x}$  is multiplied by a weight  $\omega_s$  (between 0 and 10) defined by the user through a visual interface with selectors. These 4 terms are finally added to check the final weight of each

street segment  $i$ . In Figure 11, a preview of the application interface with the main features of the system is displayed.

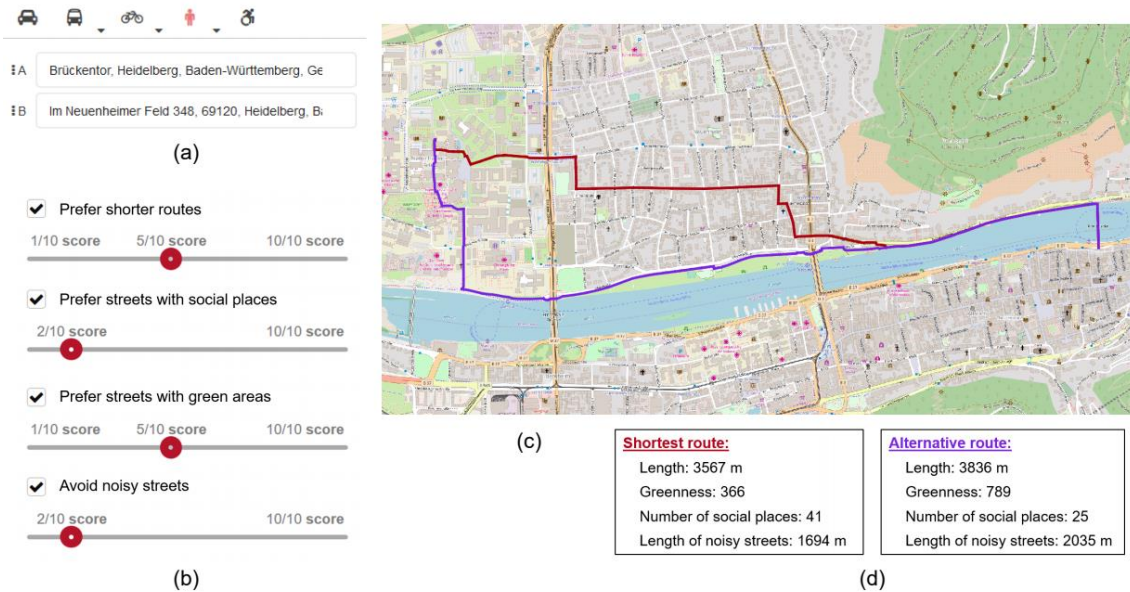


Figure 11 – Main components of the proposed pedestrian navigation system (Novack et al., 2018).

The authors also discuss the use of OSM data in the implementation of their project, concluding that it is both a shortcoming and an advantage, because, despite the data on there sometimes being incomplete with respect to certain areas of the world and in regards to other aspects, for example, in terms of accessibility information for people with disabilities, it is quickly becoming more established as a source for GIS applications, and as such, applications built with the use of these maps are expected to be sustainable long-term, especially since OSM has a well-established data structure that makes it easy to extend by multiple researchers from all parts of the world.

The authors explain how they extracted real-world factors to integrate them into the cost function. For sociability, the authors considered, in short, the existence and number of areas for leisure activities, commerce and restaurants in each segment, in a buffer (computed *polygon* or *multipolygon* type around a geometry, which represents all points whose distance from a geometry/geography is less than or equal to a given distance) of 50 meters around the segment (Figure 12).

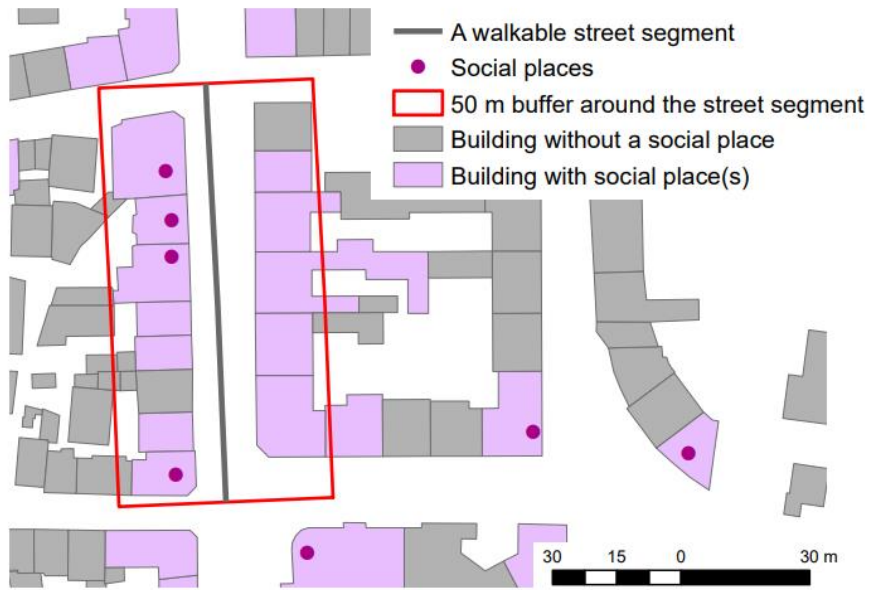


Figure 12 – Street segment with social zones and a 50 meter buffer around it (Novack et al., 2018).

For greenness, the authors essentially considered the existence and visibility of green places of any kind. Observation points were established along the segment, depending on the length of each segment (Figure 13).

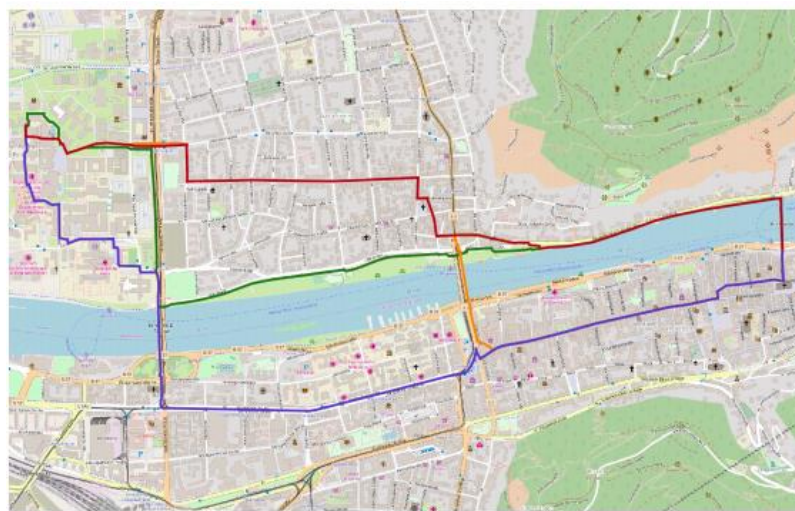


Figure 13 – Street segment with green zones within 100 m visibility radius of each observation point (Novack et al., 2018).

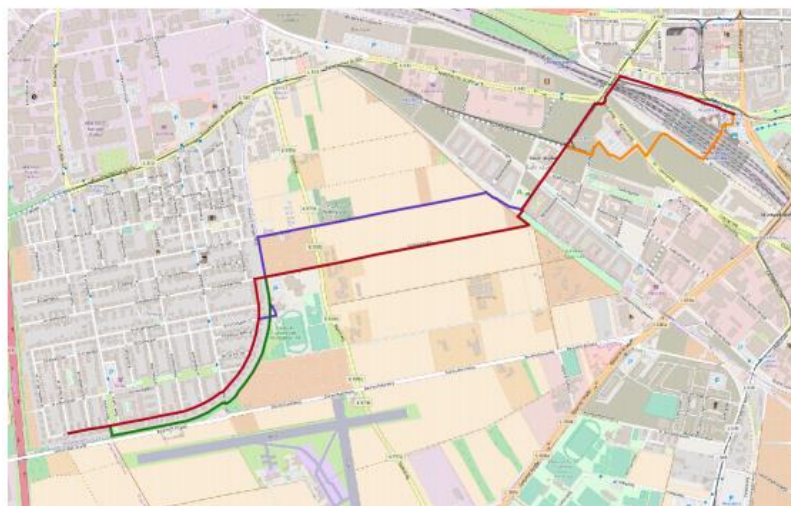
For the quietness factor, the OSM street segments were grouped into 2 categories that change according to the existing street/road type: noisy and less noisy. For example, a street segment with a road tag which indicates that this is a motorway is considered noisy, depending also on its length.

Finally, the cost function is equal to the sum of all the weighted factors, including the route length, with the weight of each factor defined by the user.

After this, the authors evaluated customized routes of 1000 trips, concluding that the system generates generally longer routes, which meet the requirements of what is requested. For instance, Figure 14 demonstrates alternative routes (green route, in green, social route, in purple and quiet route, in yellow) to the shortest route (route in red), with a weight of 2 being alternatively applied to the route length and 8 to the other factors, separately (to demonstrate how the factors affect each route). The route is of similar length, but it goes through a residential area, and there are no more interesting routes in this area that don't stray too far from the shorter route.



(a)



(b)

— Shortest route    — Green route    — Social route    — Quiet route

Figure 14 – Comparison between shortest route and alternative routes (Novack et al., 2018).

In addition, through a questionnaire of 156 participants, they assessed the usefulness, usability, controllability and transparency of the system. This questionnaire was divided into 4 parts. The first part had questions about the potential usefulness of the system (as to whether there is a demand, but no questions about the system); in the second part, the system and its operation were presented to users; in the third part, questions about the specific system were asked; finally, in the fourth part, the opinions of the participants about limitations and potential improvements were collected.

Most participants found the system useful, easy to use and felt that it gave them more control of their routes. These interviews also provide more insight into user requirements for custom route recommendation tools, such as the importance of informing them on the length of the route as well as the levels of each factor (in this case greenness, quietness, and number of social places) of the shortest and alternative routes, and the necessity of letting the users interactively define the weight they prefer for each factor. Almost 50% of the users also suggested the implementation of safe routes.

## 2.3 Summary

In this chapter I explored the current context of the recommendation systems area and its taxonomy, the concepts of pedestrian navigation and preferences of pedestrians for their routes and analyzed systems like the one that is intended to be developed in this project.

These works provided inspiration for the system developed in this project, in terms of the data to gather and use for measuring route quality and sources of that data, the tools to utilize for the system and application, and the interface and features of the system. Just like all the systems mentioned, the HealthyTrack system recommends routes for pedestrians based on the route's aspects and user's preferences, and although it shares similar goals and characteristics to these systems, it varies in some respects.

The HealthyTrack system is stored on a spatial DB and its features are accessible through a web application, like the systems discussed in (Novack *et al.*, 2018) and (Ramos *et al.*, 2018). It allows the user to select two points on a map and recommends three different types of routes: faster (shortest distance), greener (closer to green areas) and cleaner (better air quality). In the DB, with the use of the PgRouting extension, the routes are calculated, which distinguishes this system from the ones mentioned.

For the measurement of the proximity to green areas and of average air quality around a segment, a method similar to the one described in (Novack *et al.*, 2018), displayed in Figure 12, was used, with buffers established around the segments to measure

these aspects. This method is explained in more depth in the 3.3 section. Also, just like the system presented in that article, the system developed in this project uses OSM data, namely the roads of the city of Lisbon, and displays the route recommendations to the users in a similar way (alternatives with distinct colors), with a similar interface to that system, for the client. However, unlike that system, this one doesn't recommend customizable multi-criteria routes.

Also, just like in (Ramos *et al.*, 2018), as shown in Figure 9, an air quality heatmap for the city was created to allow for visualization of its distribution throughout the selected area for the system, but unlike in that solution, more polluted areas aren't simply avoided but just given a lower score, so these can still be travelled.



# Chapter 3

## HealthyTrack

In this chapter, I introduce the HealthyTrack system. In section 3.1 , I go over the gathering of data process for the system. Then, in 3.2 , you can find an overview of the architecture of the system. After that, in section 3.3 , the implementation of the HealthyTrack database with the gathered data is covered. In section 3.4 , the PgRouting extension for PostGIS is discussed, going over the steps to build the recommender feature, and then, in 3.5 , the subject of route recommendation algorithms is explored, with a focus on the solution selected for this system. I then explore the theme of visualization of spatial data for this system on the QGIS <sup>3</sup> application in section 3.6 , and after that, in 3.7 , the system's features and implementation are discussed in more detail, with a focus on the technologies used. Finally, in section 3.8 , some experiments for validation of this system are made.

### 3.1 Data Analysis and Gathering

In the following sections the decision-making process of the design behind the HealthyTrack system is documented.

#### 3.1.1 Definition of Route Quality Criteria

In this thesis, route quality criteria are defined as the collections of characteristics of the environment that pedestrians value in their travels, which influence the routes that they make, such as choosing between route  $x$  or alternative  $y$ . An example of a possible route quality criterion would be, for instance, meteorology, as it has an influence on whether, for instance, a pedestrian chooses to walk or take a taxi. Just like meteorology, as has already been discussed in this thesis, other factors have an influence on the routes of pedestrians, such as aesthetics along the route, air quality, and more.

As suggested before, the main goal of this thesis was to recommend routes that would benefit the health of the pedestrian. For this project, the criteria considered for measuring the “healthy” quality of a route includes proximity to green areas and air quality. According to the (World Health Organization, 2016), closer proximity to green areas can promote mental and physical health, and reduce morbidity and mortality in urban

---

<sup>3</sup> <https://www.qgis.org/en/site/>

residents by providing psychological relaxation and stress alleviation, stimulating social cohesion, supporting physical activity, and reducing exposure to air pollutants, noise and excessive heat. As for air quality, it has a more obvious impact on the pedestrian's health and quality of life, including effects on allergies. There are certainly more criteria for the quality of a route focused on the health of the pedestrian, but this thesis focuses on these two factors.

### 3.1.2 Data Gathering

In this section, I address the process of the gathering of spatial data for the HealthyTrack PostgreSQL DB, which includes the roads, air quality and green areas' locations in Lisbon city.

This specific data is essential for calculating the routes: the road network of the city is used to build the routes segment by segment, and their length, along with the air quality and green areas locations are used for measuring the quality of those same routes.

I first did some research on the sources for spatial data about the air quality and roads of Lisbon, and as for the green areas, an ESRI *shapefile* (archive format that contains spatial data in vector form) containing their locations and disposition in Lisbon was downloaded from the Câmara de Lisboa or Lisbon municipality (CML) spatial data public repository <sup>4</sup>.

#### Air quality

First, to obtain data about the air quality of Lisbon, I researched the web for sources of air quality data by location for the city of Lisbon.

I first looked for datasets with the yearly average air quality values for every location in Lisbon, without much luck. The main source I found was QualAr <sup>5</sup>, a project coined by the Agência Portuguesa do Ambiente (APA) or Portuguese Environment Agency, to provide a public source of air quality data. Here, I could find the hourly quantities of common air pollutants, for any specified number of days, by monitoring station. For this raw data to be useful for this project, I could either calculate an AQI dataset for the average AQI of each station based on the concentrations of each pollutant in historic data and generate a dataset with these values or fetch the updated data hourly and calculate the AQI for each station based on new values (which means I would have realistic real-time values). I could then use interpolation to assign AQIs to more points in the city, disposed equidistantly in a grid over the city, each with their AQI, and save these in a PostgreSQL

---

<sup>4</sup> <https://geodados-cml.hub.arcgis.com/>

<sup>5</sup> <https://qualar.apambiente.pt/>



table. This would be quite a complex task (especially if it was a real-time task), I didn't have experience with interpolation at the time, and didn't have this knowledge, having since come across more research papers which explore this problem in more depth such as (Ramos *et al.*, 2018). Also, since this work would present a proof of concept, there wasn't a particular interest in fetching real-time data, since the system can rely on a static dataset. Because of this, at the time, we tried to look for alternatives.

At this time, I also came across sources of real-time predictions (of air quality) which allow me to access this data through an API, and even though I didn't end up using any of these services, I thought it would be useful to list the ones I found. All the services I found require pay for access to their API, and most offer at least a free base package. Overall, any of these API services seemed a good choice to get air quality data from, even though they seem lacking in data about the city of Lisbon (having vastly more sources of data for other big cities). In Figure 15, the main comparisons made between the air quality APIs presented here are summarized.

The first one to note is Breezometer <sup>6</sup>. Through the API I can obtain the air quality for any instant, calculated as an index (which can be the Portuguese AQI) for any location in Lisbon. The data is collected through a collection of sources, such as local monitoring stations, satellites, local meteorology and traffic information. This service offers a free trial of 14 days. This seemed to be one of the best real time air quality API sources offered I found, since their method of joining the data collected seemed to be more complex, perhaps leading to more precise predictions. It also allows for the AQI to be calculated using the Portuguese index, which might be a plus because the HealthyTrack system operates on the city of Lisbon.

Another source of air quality data is AccuWeather <sup>7</sup>, which gets its data from Plume Labs <sup>8</sup>. They collect the data from the same sources as above. For Portugal, they specify that the AQI (based on the standards of the US Environmental Protection Agency, or EPA, and World Health Organization, or WHO) is calculated through data collected from local monitoring stations, belonging to the APA agency, and satellite data. The index is applied segment to segment (based on a set of roads) and can be accessed through the API for any instant in any location. There is a free limited base package which allows for at most 50 calls a day. AccuWeather is a nice source since it provides the aforementioned free package. Plume Labs, the provider of AccuWeather, also provides its own API, which is paid, however, they require you to establish contact with them to learn of what packages they offer.

---

<sup>6</sup> <https://www.breezometer.com/products/air-quality-api>

<sup>7</sup> <https://developer.accuweather.com/>

<sup>8</sup> <https://plumelabs.com/en/forecast-api/>

As for the AirVisual<sup>9</sup> (from the IQAir company) source, this API, just like the other ones, obtains their data from both governmental (APA, in the case of Portugal) and individual monitoring stations. They are also quite transparent about the estimation of the AQI (based on US guidelines). Just like the previous API, for this one there is a free limited access base package, but in this case, it allows for 10,000 calls per month, which is a more appealing offer than the previous one.

OpenWeatherMap<sup>10</sup> is yet another source of air quality data for API calls, which also offers a free base package. The index guidelines followed for the estimation of the AQI are not specified. They explain the way they calculate the AQI in greater detail than the other API tools mentioned here, with use of statistic error metrics (MAE and RMSE) to estimate it as accurately as possible. Once again, there is a free package, which in this case offers up to 1000000 calls a month, with at most 60 calls per minute.

Real-time air quality APIs	Air quality format	API access plans	Data sources	Resolution	Accuracy guarantee	Updates
Breezometer	Portuguese AQI (for the Portugal area), BAQI, individual pollutant concentrations ( $PM_{2.5}$ , $NO_2$ , etc.)	Paid (pricing plans by request), 14-day free trial	Local monitoring stations, satellites, traffic data, weather and meteorology, fires, land cover, algorithms and ML models	Down to 5 meter resolution	Comparison of air quality calculations with those from local governmental agencies with MAE	Hourly
Accuweather	Plume AQI (based on EPA and WHO standards)	Free (limited to 50 calls/day, limit of 1 key/developer), paid plans starting from \$25/mo	Local governmental monitoring stations (APA), satellite data	Down to 10 meter resolution	Cross validation (using 80% of the data for prediction and remaining 20% for evaluation) with MAE	Daily
Airvisual (IqAir)	AQI (based on US guidelines)	Free (limited to 10,000 calls/month), paid plans pricing by request	Local monitoring stations, satellites, weather and climate data	Not disclosed	Cross-checking of spikes with nearby measurements and against historical patterns and other parameters	Hourly
Openweathermap	CAQI (European AQI)	Free (60 calls/minute) and 1,000,000 calls/month), paid starting from 35 €/month	Local monitoring stations, satellites, weather data	Down to 500 meter resolution	Usage of the following metrics: MAE, RMSE, reliability and inaccuracy	Hourly

Figure 15 – Real-time air quality API services compared.

We ended up being provided a *csv* dataset from Professor Pedro Pinho of the CE3C investigation unit at Faculdade de Ciências of Universidade de Lisboa, which contains the average air quality values for Lisbon (with a resolution of 5 meters), converted from an air quality raster, which was the result of the work in (Matos, Vieira, Rocha, Branquinho, & Pinho, 2019), where the air quality of Lisbon was modelled through interpolation with the use of lichens as an ecological indicator.

In Figure 16, you can visualize the graphic of the original model of the air quality in this dataset, and, in Figure 17, the interpolated air quality heatmap of this data, after being imported to the DB. You can learn more about this data and how it is stored in section 3.3

<sup>9</sup> <https://www.iqair.com/us/commercial/air-quality-monitors/airvisual-platform/api>

<sup>10</sup> <https://openweathermap.org/api/air-pollution>

, and why the colors of the air quality heatmap, in Figure 17, are inverted in relation to the ones in Figure 16, in section 3.6 .

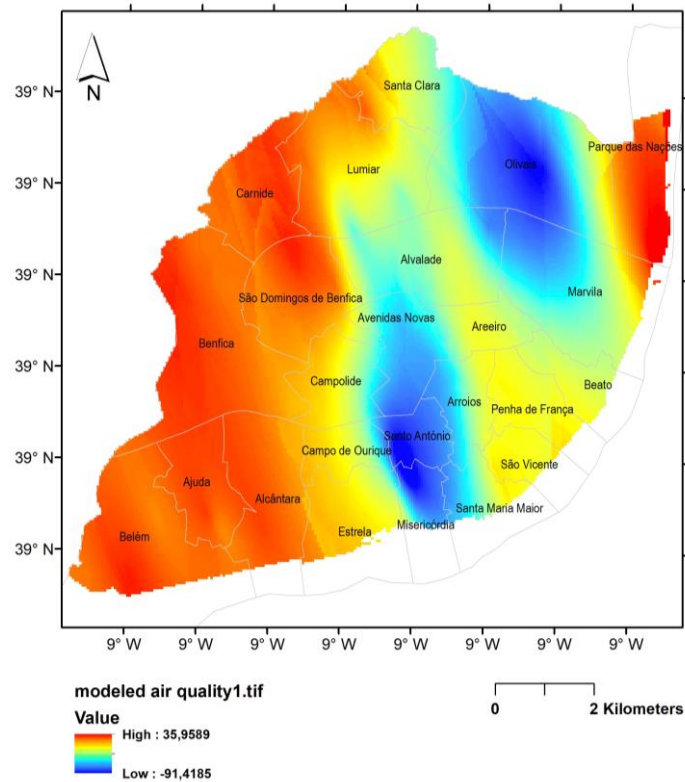


Figure 16 – Original model of air quality.

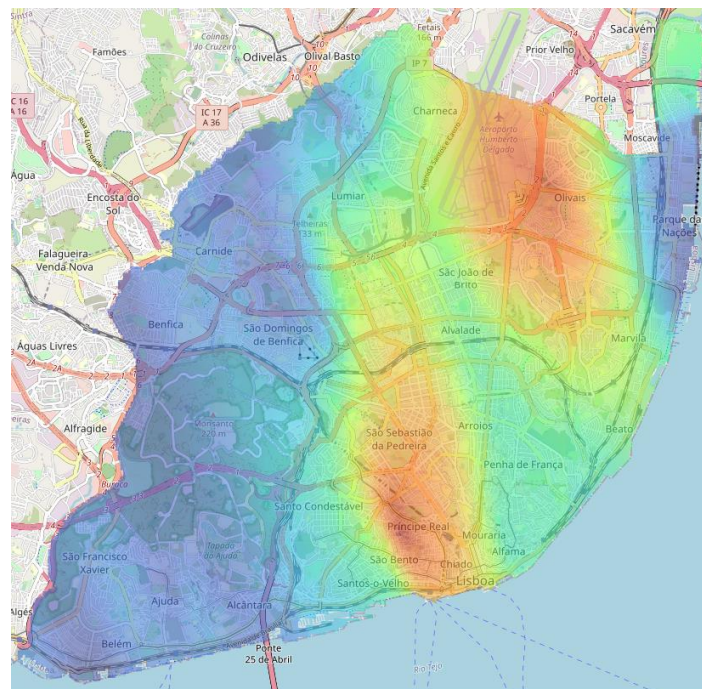


Figure 17 – Interpolated raster layer of air quality (Heatmap) in QGIS.

## Roads

To build routes, I needed to obtain spatial data with the road network of Lisbon for the purpose of this proof of concept. One of the best sources for this was OSM, as it is the main open-source collaborative project which rivals Google Maps and gets updated regularly. Once again, for the purposes of the proof-of-concept developed in this project, a static source of data was all that was necessary for the functionality of the system.

To obtain the data specific to the city of Lisbon, I went through the OSM wiki, which contains information about all the reputable sources of OSM spatial data. During this research I discovered, among other platforms, an open-source tool created by the Humanitarian OSM Team <sup>11</sup>, a nonprofit that is geared towards generating spatial data to support humanitarian aid and economic development. This tool, called HOT export tool<sup>12</sup>, allows me to export custom extracts of OSM. This means I can export only the selected data (provided the export size is not too large) inside a given shape, by drawing on the map or providing a *GeoJSON* file (no larger than 5 mb) with the shape (in coordinates according to the WGS 84 reference system). I can also edit and filter the results further by selecting what data I would like to be included in the export, including the table attributes (columns). Finally, I can export this data in most common spatial data formats, such as the OSM *pbf* or the *shapefile* format. Before this whole process, I am required to create an OSM account.

This tool proved to be very useful, as the other OSM sources I found only allowed, at most, for me to download the spatial data for all of Portugal, which then not only had to be reduced to the area of Lisbon, but also reduced to only the roads of Lisbon either through the PostgreSQL interactive terminal (psql) or Windows Powershell, with OSM commands. I could also have done that by downloading data from the Geofabrik website (another reputable OSM source, which is the main source of smaller OSM exports, containing the main countries and some main cities), however, this tool simplified my work in that regard.

For the export, I first got a *GeoJSON* file with the spatial data of the administrative boundaries of Lisbon (containing the city of Lisbon) from the CML spatial data public repository (Figure 18), using it to export only the data inside these boundaries. In my case, I only wanted to export the roads of Lisbon, so I selected only the roads, which also include the footpaths. I export all the roads and remove the ones where the *highway* tag equals *motorway*, since generally most roads are traversable alongside by pedestrians, except for motorways (commonly known as highways), which are in the minority.

---

<sup>11</sup> <https://www.hotosm.org/>

<sup>12</sup> <https://export.hotosm.org/en/v3/>

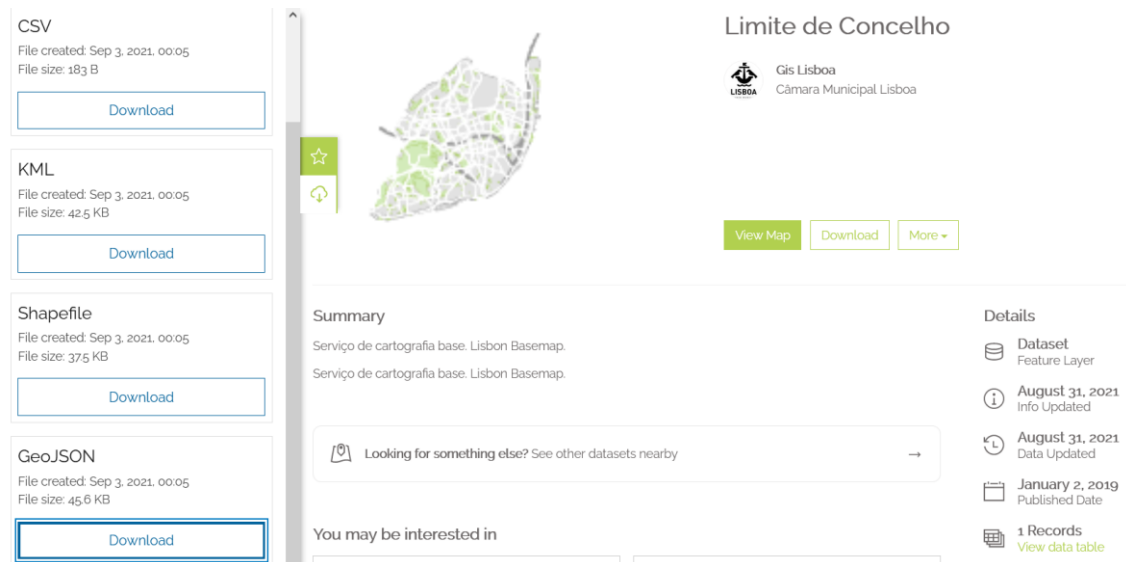


Figure 18 – First step of the OSM data exportation process in the CML geodata repository.

To elaborate on the decision mentioned in the last paragraph, beforehand, I learned about OSM tags in the OSM wiki, which describe features of map elements or a DB changeset. These consist of a key and value. For roads, streets or paths, the *highway* key (attribute) corresponds to the type of road. For instance, for a motorway road, *highway* equals *motorway*. So most or all motorway type roads inside the DB will be identified (correctly) with the *motorway* value inside the *highway* column. Of course, each tag has their most commonly used values (since OSM is a purely collaborative project, there is not necessarily a formal agreed upon standard, but there is a general respect for establishing a common ground). In that same page, I learned about the most common values for the *highway* key, which include *motorway*, *trunk* (the most important roads in a country's system that aren't motorways), *primary* (primary roads), *secondary* (secondary roads) etc.

To export the roads of Lisbon using this tool, I go through a series of steps, which I go through here. On the first step (Figure 19) you must name your export, give it a description and assign it to one of your projects (both optional). At any step you can select the area of export on the map.

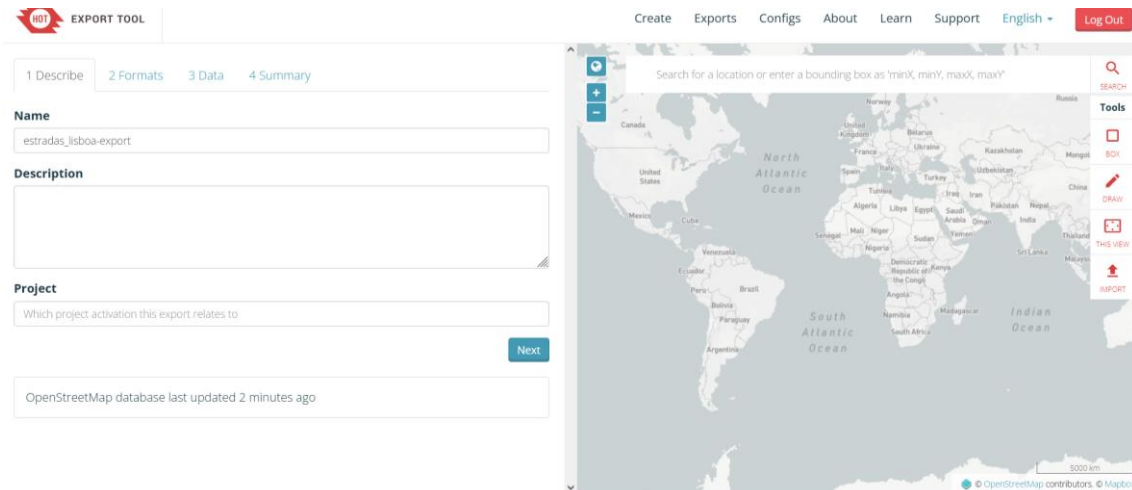


Figure 19 – First step of the OSM data exportation process with the HOT export tool.

On the second step (Figure 20) you can select the spatial data format of your export.

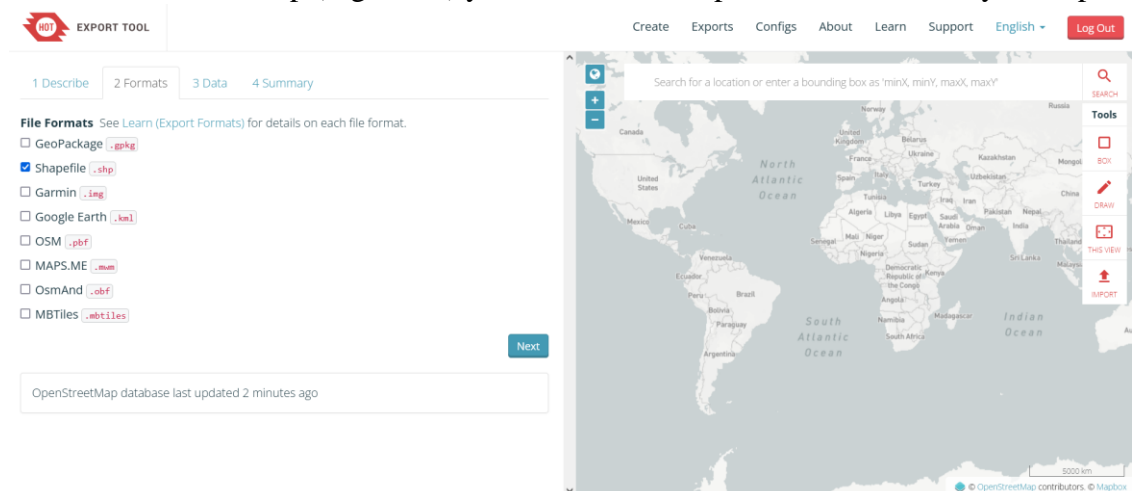


Figure 20 – Second step of the OSM data exportation process.

For the third step (Figure 21) you can select the geometries you want to export and, with the use of the YAML Ain't Markup Language (YAML), you can customize the export in further detail.



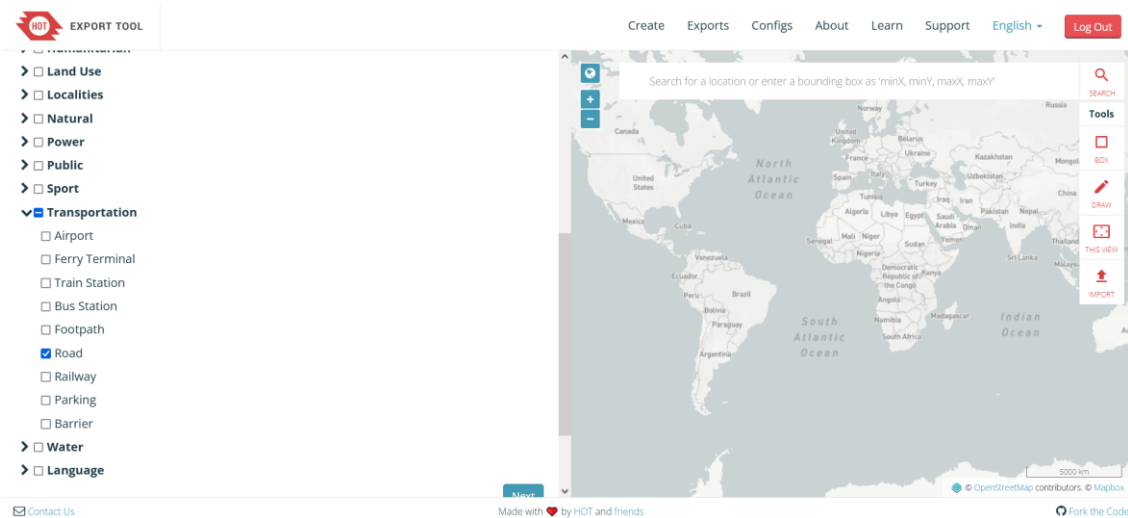


Figure 21 – Third step of the OSM data exportation process.

At the fourth step (Figure 22) you have some more advanced options to choose from for the publication of the export and such. The area of export (inside Lisbon boundaries from CML) is already defined at this stage.

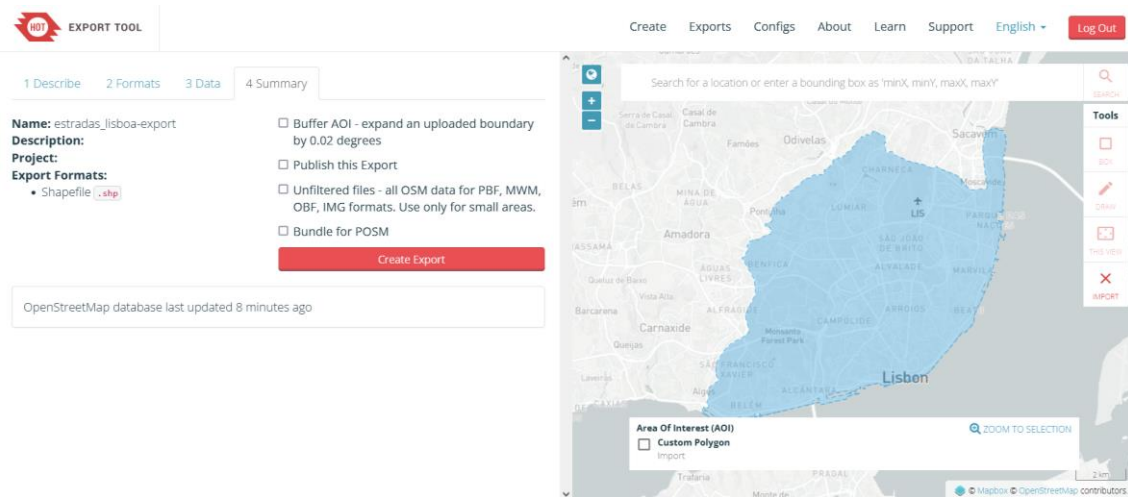


Figure 22 – Fourth step of the OSM data exportation process.

Finally, in the fifth and final step of this process (Figure 23) you can review what you will/have exported, delete it, among other options. In this step you can see what data the export will contain (including a visual preview of the data projected on the map).

The export area polygon is automatically simplified at this stage if it's a complex geometry (composed of more than 500 points), as is the case here, and enlarged to fit everything inside the scope you want. Because of this, I retrieved data outside the scope that I wanted. To fix this and keep only the information inside that scope, after importing the roads to the DB, with the use of the same boundaries of Lisbon, which were imported to the same DB, I used the `pgsql2shp` command, which returns a *shapefile* with the data inside the boundaries. I could also have achieved the same goal through pgAdmin

(graphical management tool for PostgreSQL), through psql with a query or, if exporting an OSM file, through OSM commands in the terminal.

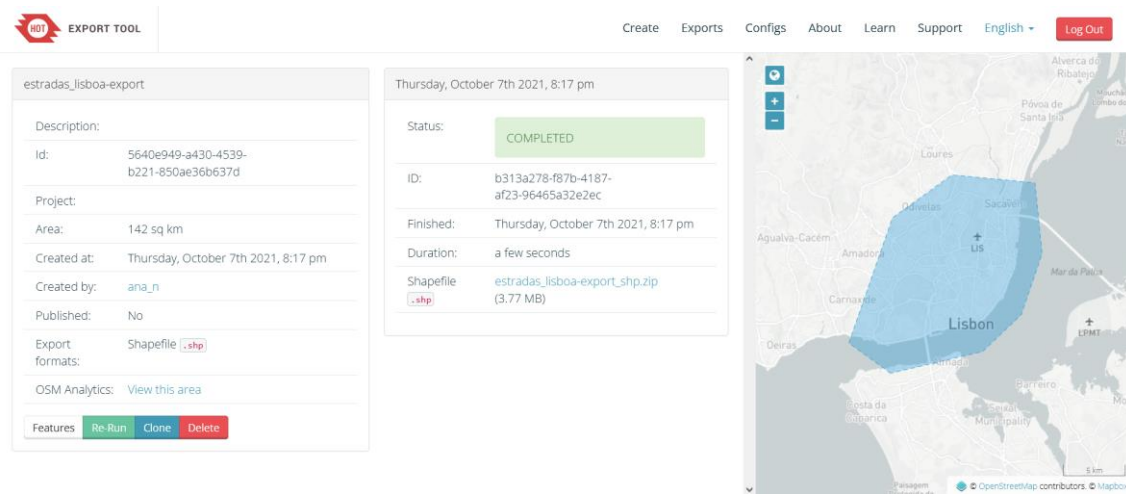


Figure 23 – Fifth and final step of the OSM data exportation process.

The following is the command I ran through the Windows PowerShell to filter the spatial data for only the roads inside Lisbon’s boundaries. This command saves a *shapefile* with all the roads inside the boundaries to any location specified in the pc file system.

```
pgsql2shp -f [location of geometry file] -h [hostname] -u [user] -P
[user password] [DB name] "SELECT poly.* FROM
lisboa.boundariestable b JOIN lisboa.roadtable r ON
(ST_Within(r.geom,b.geom) );"
```

A representation of the spatial dataset with the roads of Lisbon in QGIS can be visualized in Figure 24.





Figure 24 – Original roads dataset, imported from the CML public spatial data repository, in QGIS.

## Green Areas

As for the spatial data on green areas, I downloaded the dataset containing the green areas' locations and disposition in Lisbon (Figure 28), in the *shapefile* format, from the CML public spatial data repository.

**Espaços Verdes**

 Gis Lisboa  
Câmara Municipal Lisboa

[View Map](#) [Download](#) [More](#)

---

**Summary**

Serviço de dados geográfico com indicação do arvoredo, elementos de água, corredor verde e hortas urbanas existentes na cidade de Lisboa. Map service locating trees, fountains and kitchen gardens in Lisbon.

Serviço de dados geográfico com indicação do arvoredo, elementos de água, corredor verde e hortas urbanas existentes na cidade de Lisboa. Map service locating trees, fountains and kitchen gardens in Lisbon.

**Details**




-  Dataset  
Feature Layer
-  February 15, 2022  
Info Updated
-  February 15, 2022  
Data Updated
-  December 28, 2018

Figure 25 – Green areas dataset page in the CML public spatial data repository.

In this repository, I found not only that source, which contains all the areas that are considered “green” in Lisbon (meaning, spaces that contain grass, trees or other vegetation), but also another source specified on the parks and gardens of Lisbon (Figure 26) and yet another (not included in the green areas dataset) with the locations of trees and shrubbery (Figure 27).

Figure 26 – Parks and gardens dataset page in the CML public spatial data repository.

Figure 27 – Trees and shrubbery dataset page in the CML public spatial data repository.

At the end it was decided to keep only the green areas dataset, as this covers the most ground out of all these datasets, which is good for testing purposes. In Figure 28, you can visualize a representation of this dataset in QGIS, after being imported into the DB.

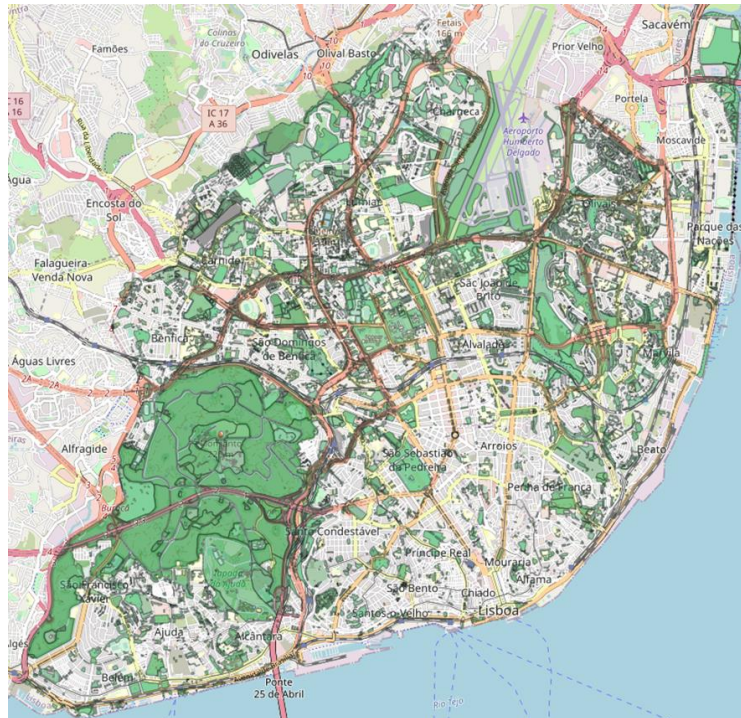


Figure 28 – Green areas dataset, imported from the CML public spatial data repository, in QGIS.

## 3.2 Architecture

In this section, the HealthyTrack system architecture, pictured in Figure 29, is presented.

This system is composed by a PostgreSQL spatial DB (which contains the roads, green areas and air quality tables), a NodeJS server and a JS client, composing the HealthyTrack route recommender web application.

The web services were implemented with the use of ExpressJS, for the server-side, and on the client side, the Leaflet library is used for displaying the maps and interacting with them for authorized operations (requesting route recommendations). In simple terms, the client can request the server for information about the origin and destination edges (through the *getEdge* web service) of the route and request a recommended route (through the *getRecommendedRoute* web service).

A user can get a recommended route through the web application by providing a point of origin and destination for that route, which can be done in multiple ways (discussed in section 3.7.1 ). The server then queries the DB for this information. The routes are generated with the use of the PgRouting extension installed on the PostgreSQL DB.

The Nominatim <sup>13</sup> API servers can get requests for coordinates from a written address through the client in case the user inputs a written address when asking for the recommended route.

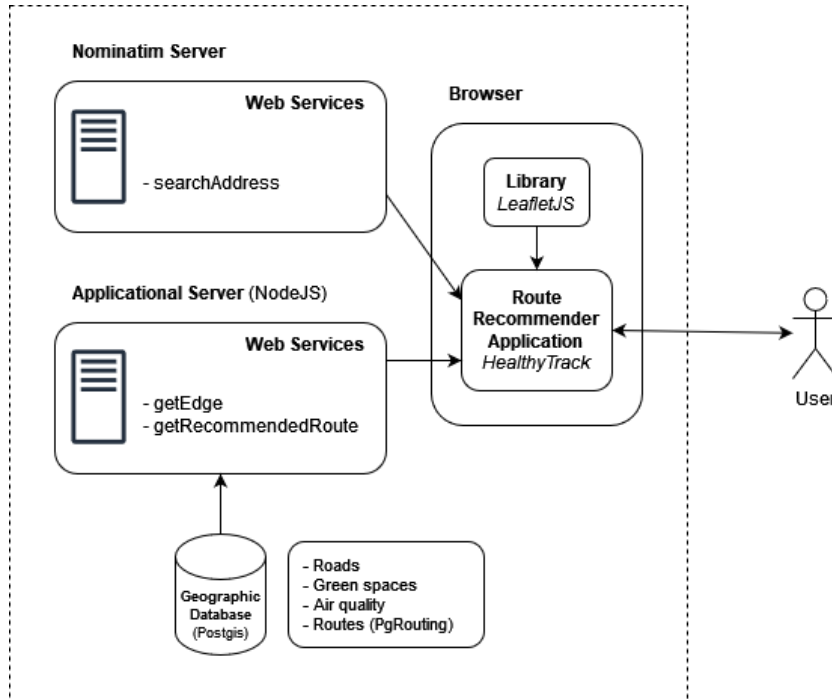


Figure 29 – Architecture of the Healthy Track system.

### 3.3 HealthyTrack Database

The HealthyTrack DB contains the road network and other datasets with the spatial features necessary to create “healthy” routes, as they influence the disposition of the routes.

Before implementing the DB, the software to handle this project needed to be selected. The main goal was to find a DB that could handle spatial data, prioritizing free, open-source software and DBs managed with the SQL language, easy to set up but extensible.

For these reasons, the PostgreSQL database management system (with the PostGIS spatial extension), one of the most widely used solutions for handling spatial data, was selected. PostgreSQL is a highly regarded free and open-source object-relational DB which uses a variation of the SQL language. It is also known for its powerful ability to work with spatial data with the use of the PostGIS extension, which adds the ability to handle and visualize spatial data. QGIS, an open-source spatial data visualizer and editor

<sup>13</sup> <https://nominatim.org/>



which rivals ArcGIS (a commercial GIS software by ESRI mostly used in professional settings, one of the most well-known GIS software companies), was installed for better visualization and edition of the spatial data. I will get into the QGIS functionality in the 3.6 section.

To implement this DB, after installation of the PostgreSQL software, the HealthyTrack DB was created and extended with the PostGIS and PgRouting (plugin for the routes, addressed in 3.4 tools. After this, a new schema, called *lisboa*, (inside which the tables are) was created inside the DB. A connection was then established, in QGIS, to the same DB.

The selected spatial reference system to be working on was the WGS 84 Coordinate Reference system (CRS) standard, identified by the 4326 spatial reference identifier (SRID), which is the one used by GPS systems. This standard was selected because most of the datasets had this format, however, the air quality data was initially based in the ETRS89 / Portugal-TM06 CRS and it ended up being converted to WGS 84 to match the rest.

Most of the datasets, like the green areas and roads, were in the *shapefile* format. To import this format, the PostGIS Shapefile Import/Export (*shp2pgsql*) tool, which has a GUI (Figure 30) that makes the process easier, was used. This interface allows me to select the *shapefile* on the file system, choose to create a new table or fill an existing one, and indicate the SRID of the EPSG (EPSG Geodetic Parameter Dataset, a public spatial format registry) format to be stored and the name of the column with the geometry code.

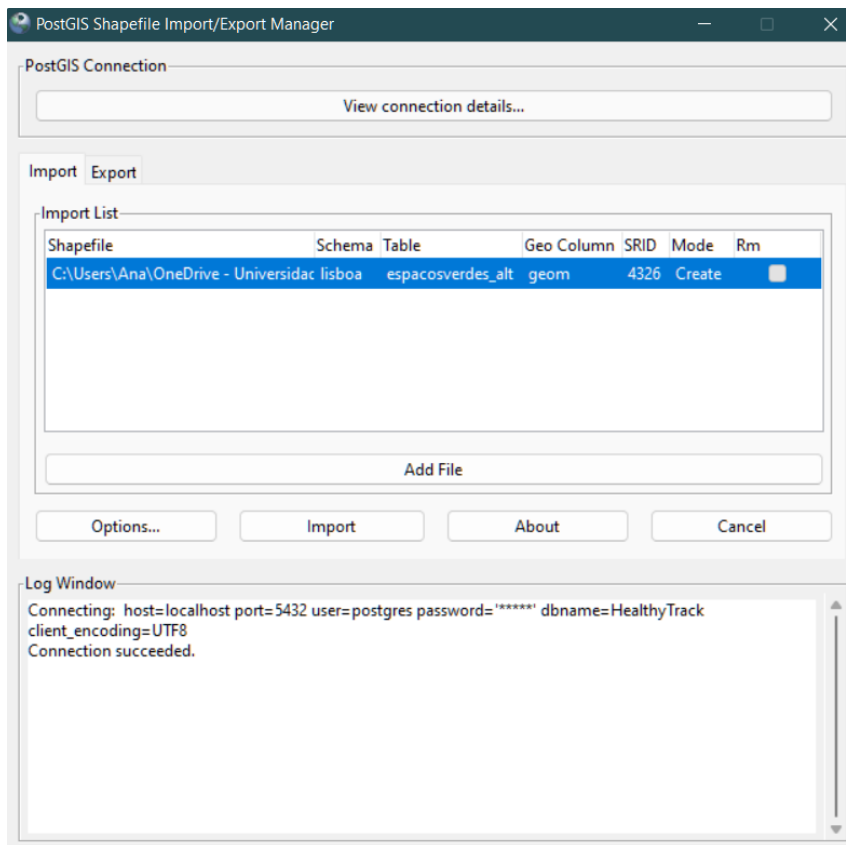


Figure 30 – PostGIS shp2pgsql tool.

For the air quality, in *csv* format, I first created the columns inside the air quality table in the DB (Figure 31) to match the ones in the *csv* dataset, with the type matching the contents of each attribute too, and finally, I used the following *copy psql* command to import the data:

```
psql -U postgres -d 'HealthyTrack' -c "\copy lisboa.qualidadeair
(column attributes) FROM 'C:\(path) \airqualityvalues.csv' WITH DE-
LIMITER ';' CSV HEADER"
```

After this, inside the *lisboa* schema we have the *espacosverdes* (green areas), *qualidadeair* (air quality) and *estradas* (roads) tables for the Lisbon city (Figure 31).

Each of the geometries is identified by an id, called *gid* for the *espacosverdes* and *estradas* table and *id* for the *qualidadeair* table (depending on where they were imported from). Another attribute common to all the tables is the geometry code, which is decoded by the spatial DB (according to the SRID and type of the geometry) to a geometry displayed on the map. The types of geometry for each table are distinct. I don't list the other attributes of the tables here, just the essential ones for the HealthyTrack system to work, but for instance, for the *espacosverdes* table there are other attributes about these geometries, such as their address and name.

Another detail to point out is the PostgreSQL geometry types for each table. The geometries stored in the air quality table are of the *point* type, each of them with their

own air quality value, named *valor\_poluicao*. A *point* in PostgreSQL is essentially just a location on the world, generally with a dimension of 2, meaning *x* and *y* (latitude and longitude), represented as text in the following syntax: *POINT(y x)*. As for the *roads*, each segment is of the type *linestring*, which is a sequence of *points*, represented as text in the following syntax: *LINESTRING(y x)*.

Each green area is of the type *multipolygon*, which is technically a collection of *polygons* (a *polygon* is a sequence of *points* that closes in on itself, with the same first *point* as the last), even though some of the *multipolygons* in the table are actually just *polygons* (only one *polygon*), since when importing, the *shp2pgsql* by default turns all geometries into *multi* geometries. The *roads* geometries were also turned into a *multi* type because of this, but I converted them back into *linestrings* for the routing operation, which I will explain in more detail in the 3.5 section.

After this, I used the PgRouting extension to turn the road network into a routable graph (detailed process in section 3.4 , over which I can use a pathfinding algorithm to generate routes. With the use of this extension, I divided the road network table into a table of nodes, which are the points where edges meet, called *estradas\_noded\_vertices\_pgr* and a table of edges, called *estradas\_noded*.

I also created new columns in *estradas\_noded* (see Figure 31), which provided additional information on each edge, based on the route criteria for this system, and essential for the recommendations. These columns are *airquality*, *airquality\_pos*, *length*, *fast\_cost*, *clean\_cost* and *green\_cost*.

The *airquality* column corresponds to the measured air quality on the edge, the average value of pollution of all points inside an area of 50 meters around the segment, from all sides. *airquality\_pos* corresponds to the *airquality* values converted to positive values (since the original air quality values ranged from negative to positive, as can be viewed in Figure 16, and both Dijkstra and TRSP, the algorithms used throughout this project to recommend routes, discussed in section 3.5 and 3.7.4 respectively, ignore edges with a negative cost) and inverted since, for both the algorithms, lower cost means better air quality and vice-versa. *length* is the measured length of the segment in meters, which is used as the cost for the fast route. As for *greenmeter*, it is the level of proximity to green areas, which can be 1, which means the edge intersects the green area, 2, which means the edge is closer than 25 meters to the green area, or 3, which means it's farther than 25 meters to the green area from all sides. *fast\_cost* is the cost based on route length. *clean\_cost* is the cost based on air quality, and finally, *green\_cost* corresponds to the cost based on proximity to green areas. I go into detail on the formulas for the formulas of each cost in section 3.5 .

In the *roads* table, *source* and *target* correspond to the *id* in *estradas\_noded\_vertices\_pgr* of the node which is the source or target of that edge. The same happens with the *source* and *target* columns of the *estradas\_noded* table; also, when an edge is split into several edges, a *sub\_id* identifies the number of that segment inside the original edge from *source* to *target* of the original (for instance, if it is the first after *source*, it's number 1, if last it's *x* depending for *x* broken down segments); *old\_id* simply identifies their original *id* in the *roads* table, before they were split.

The data model for the final version of the HealthyTrack database can be viewed in Figure 31.

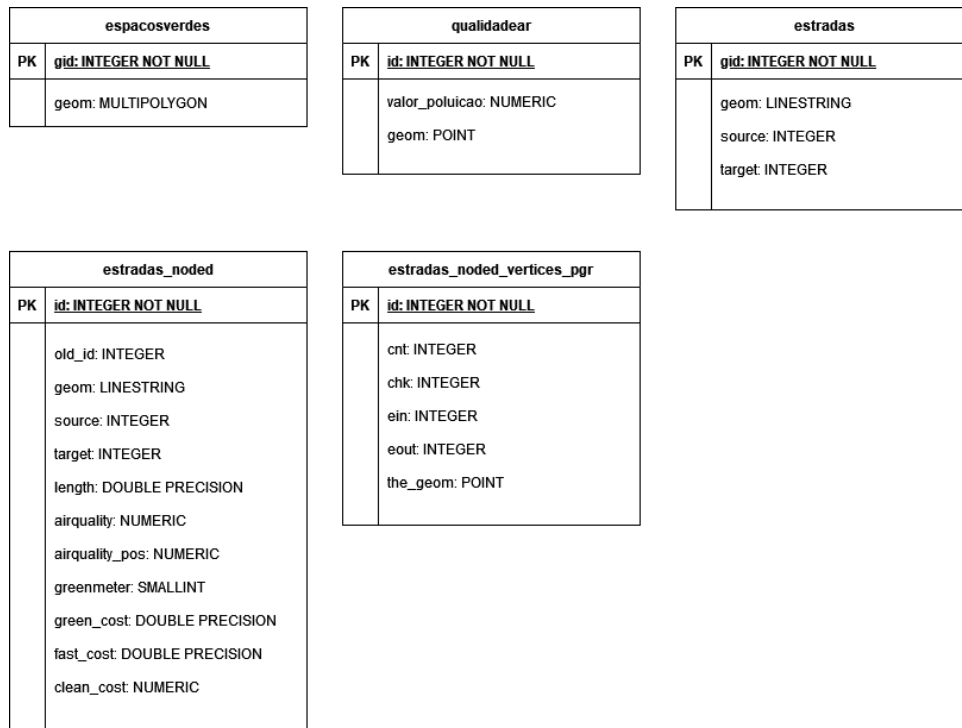


Figure 31 – Data model of HealthyTrack DB.

To calculate the *airquality* value, first, a view with a 50-meter buffer for each edge was created and, after that, another view with each edge's average pollution, measured from the pollution values inside that buffer's scope, was created and then the *airquality* column was updated with the corresponding air quality values for each edge. You can view an example of the buffer structure for measuring the air quality in Figure 32.



```

-- create view with 50 meter buffers for every edge
CREATE VIEW lisboa.estradas_buffer_pol AS
SELECT id, ST_Buffer(st_transform(geom,3763),50) AS buffer
FROM lisboa.estradas_alt_noded
ORDER BY id;

-- create view of edge average pollution within 50 meter buffer
CREATE VIEW lisboa.avg_pol_e_bff AS
SELECT eb.id as id, AVG(value_poluicao) as poluicao_med
FROM lisboa.qualidadear_alt qa
JOIN lisboa.estradas_buffer_pol eb
ON ST_Within(qa.geom, eb.buffer)
GROUP BY eb.id
ORDER BY eb.id;

-- set average pollution value for each edge
UPDATE lisboa.estradas_alt_noded e
SET airquality=avp.poluicao_med
FROM lisboa.avg_pol_e_bff avp
WHERE avp.id=e.id;

```



Figure 32 – Example of a buffer of air quality around an edge in QGIS.

After this, this data was converted to *airquality\_pos*, through the general steps mentioned in an earlier paragraph.

To calculate the level of proximity to green areas for each edge, a view with 50-meter buffers for every edge at all sides was created, and then separate views were created for the three levels of proximity: within the 25 meters and intersecting, within 25 meters but not intersecting and outside of the 25-meter range. After that the *greenmeter* rows were updated with their corresponding levels. You can view an example of the buffer

structure for measuring the level of proximity to green areas, with 25 meters extending from each side of the edge, in Figure 33.

```
-- 1. create view with 25 meter buffers for every edge
CREATE VIEW lisboa.estradas_buffer_g AS
SELECT id, st_transform(ST_Buffer(st_transform(geom,3763),25),4326)
AS buffer
FROM lisboa.estradas_alt_noded
ORDER BY id;

-- 2. create a view with edges that have polygons within 25 meter
buffer
CREATE VIEW lisboa.e_iswithin AS
SELECT DISTINCT eb.id AS edge_id
FROM lisboa.estradas_buffer_g eb
JOIN lisboa.espacosverdes_alt ev
ON ST_Intersects(eb.buffer, ev.geom);

-- 3. create a view with edges that dont intersect with any green
area
CREATE VIEW lisboa.e_notintersects AS
SELECT DISTINCT edge.id AS edge_id
FROM lisboa.estradas_alt_noded edge
LEFT JOIN lisboa.espacosverdes_alt ev
ON ST_Intersects(edge.geom, ev.geom)
WHERE ev.gid IS NULL;

-- 4. create a view with edges that dont intersect with any green
area but have green areas within 25 meter buffer
CREATE VIEW lisboa.closeto_g_areas_edges_bff AS
SELECT wi.edge_id
FROM lisboa.e_iswithin wi
JOIN lisboa.e_notintersects ni
ON wi.edge_id=ni.edge_id;

-- 5. create a view with edges that have polygons intersecting
CREATE VIEW lisboa.g_areas_e AS
SELECT DISTINCT id as edge_id
FROM lisboa.estradas_alt_noded edge
JOIN lisboa.espacosverdes_alt ev
ON ST_Intersects(edge.geom, ev.geom);
```

```

-- 6. update table with edges that dont intersect with any green
area but have green areas within 25 meter buffer
UPDATE lisboa.estradas_alt_noded e
SET greenmeter = 2
FROM lisboa.closeto_g_areas_edges_bff ctg
WHERE e.id = ctg.edge_id;

-- 7. update table with edges that are intersecting
UPDATE lisboa.estradas_alt_noded e
SET greenmeter = 1
FROM lisboa.g_areas_e g
WHERE e.id = g.edge_id;

-- 8. finally update table with polygons farther than 100 mt
UPDATE lisboa.estradas_alt_noded
SET greenmeter=3
WHERE greenmeter IS NULL;

```

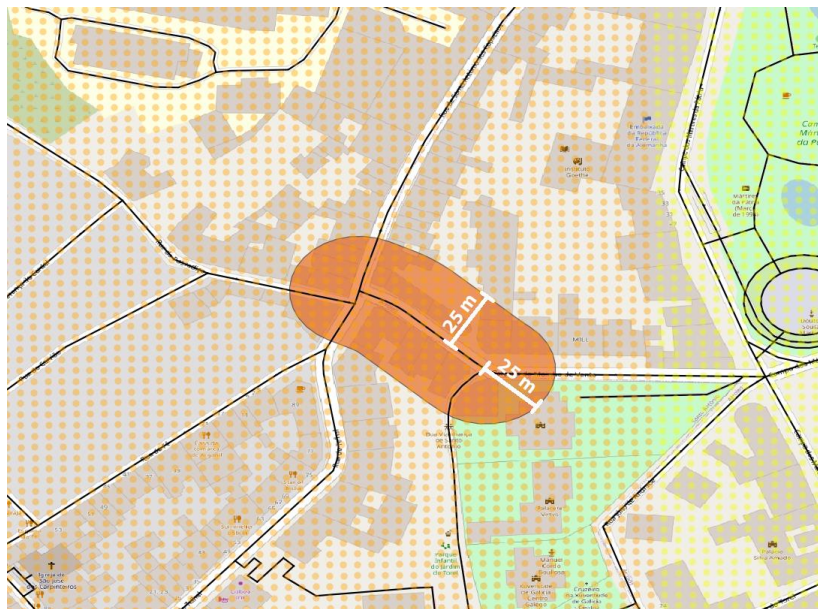


Figure 33 – Example of a buffer for the green areas around an edge in QGIS.

A 50-meter value was selected as the distance around the edge to measure for air quality and a 25-meter value to measure for proximity to green areas since we wanted to keep distances as short as possible, because Lisbon, as a city, is densely populated, so distances between places are shorter, and a walking distance of more than 50 meters, depending on the person, might already be a challenging distance. The air quality buffer is larger to catch a broader area because in case the air quality in a certain area is low, it has a negative effect on surrounding areas (unlike being close to a green space which always affects it positively), so by increasing this range, this will, in theory, aid in avoiding areas of lower air quality even more.

### 3.4 Routing with PgRouting

While configuring the DB, I investigated routing solutions to apply pathfinding algorithms for recommending routes. To implement a route recommender similarly to the way a service like Google Maps does, the recommendations should be based on a pathfinding algorithm, such as Dijkstra.

I first discovered a routing service called Open-Source Routing Machine <sup>14</sup> (OSRM), which can be used through the Leaflet library, the library I will be using to create my interactive web application which displays the map of Lisbon and recommended routes. This was limited from the aspect of customizing the routes quality criteria. After this, I discovered the PgRouting extension, an extension for PostGIS, which allows for creating a routable DB and, subsequently, to create routes, which uses most of the pathfinding algorithms (Dijkstra, A\*, etc.). With this extension, I can not only generate routes with the use of many algorithms, as well as turn a road network table (with the *linestring* geometries of road segments) into a graph that is routable (from which I can generate routes).

After installing the PgRouting extension, I first converted the *roads* from the *multilinestring* type to the *linestring* type. The reason I did this is that the PgRouting algorithm functions better with *linestrings*, as when converting the road network to a routable graph, what happens is that big segments, that make one *linestring*, are broken down into smaller segments, making multiple *linestrings*, generally where there is an intersection (where nonparallel *linestrings* cross). In the older versions of PgRouting, *multilinestrings* were not supported because it couldn't define their source and target point. But even though they are supported now, it is still safer to use the *linestring* format.

Most of the *linestrings* in the *roads* table weren't in fact, real *multilinestrings* (they were *multilinestrings* with only one element) and were just converted to a *multi* type by the *shapefile* importer, which does that by omission. As such, they were easily converted back to the *linestring* format, and as for the actual *multilinestrings*, of which there were only a few, they were removed from the database with a query, separated into singular *linestrings* and inserted as *linestrings* with new ids into the DB.

Then, a table with the identification of all nodes and a topology (creating vertices and connecting edges with respective vertices) was created. Before this, on the original *roads* table, the columns source and target were created, which are to be filled by the ids of that edge's source and target vertex respectively, when the function *pgr\_createTopology* is run on the pgAdmin query terminal, creating the vertices table with each node referenced on the source and target columns of the edges table.

---

<sup>14</sup> <http://project-osrm.org/>

Following this, with *pgr\_analyzeGraph*, the edges table is checked for dead ends, faults in the segments, isolated edges, “ring geometries” and intersections. This function adds stats to the vertices table about the number of times a vertex is referenced (to check for isolated edges, for instance). It fills the *cnt* (number of edges in *roads* table that reference this vertex) and *chk* (indicator of a possible issue on this vertex) columns.

Then, with *pgr\_analyzeOneway* I check for the existence of one-way streets, filling the *ein* (number of vertices in the edges table that reference this vertex as incoming) and *eout* (number of vertices in the edges table that reference this vertex as outgoing) columns of the vertices table. This function verifies if the edges of source or sink nodes which are one-way only constitute a dead-end path (if the edge direction is towards the sink node, which makes it impossible to turn back). These issues are automatically resolved (one-way edges that go towards a sink node are flipped backwards). In our case, since, as I mentioned earlier, pedestrians are allowed to travel in both directions, all *roads* are defined as two-way, bypassing this check. It’s important to mention that there is also the case of pedestrian dead ends in the real world. In this work, this is not taken into consideration.

After this, using *pgr\_nodeNetwork*, the noded edges table is created, according to what was found in the previous analysis. What happens is that, for instance, the segments which weren’t cut at intersections are now broken into several different segments and connected through the newly created vertices at those intersections, which now constitute nodes that sometimes connect multiple edges. A new table with the prefix *\_noded* is created, which contains some of the edges of the original *roads* table and some new ones. We get the columns *id*, which corresponds to this edge’s id, *old\_id*, corresponding to their previous id and *sub\_id*, a number for those edges that have been cut in various segments, indicating their place in the original edge in relation to the source, which ascends the further from source.

We must, then again, run the *pgr\_createtopology* function for the *\_noded* table, to create and fill the new vertices table with the updated information and with *pgr\_analyzeGraph*, analyze the table to check for existing issues and fill *cnt* and *chk*.

In Figure 34, the *estradas\_noded* table (the final roads table or the table of edges, after being turned into a “routable” graph through the process documented in this section) can be viewed in QGIS. The representation of this dataset is very similar to that of the original roads dataset, represented in Figure 24.





Figure 34 – “Routable” roads (*estradas\_noded*) table represented in QGIS.

Finally, a pathfinding algorithm to calculate the route with the use of the *\_noded* table must be chosen. For *pgr\_dijkstra*, I need to provide the *id* of the origin and destination edge, their *source* and *target* and either a column with the calculated cost or a formula to calculate the cost of each edge and indicate if the graph is directed or undirected. In this project, once again because pedestrians can travel the roads in any direction, the graph was defined as undirected. Here is an example of a query to run the Dijkstra algorithm in PostgreSQL.

```
SELECT s.path_seq, s.node, s.edge, s.cost, ST_astext(b.geom) as
edge_geom_as_text, b.geom as edge_geom, ST_astext(v.the_geom) as
node_geom_as_text, v.the_geom as node_geom
FROM (SELECT seq , path_seq , node, edge, cost , agg_cost
FROM pgr_dijkstra('SELECT id, source, target, length as cost FROM
lisboa.estradas_alt_noded', origin_node, destination_node, FALSE))
as s
LEFT JOIN lisboa.estradas_alt_noded b
ON (id = s.edge)
LEFT JOIN lisboa.estradas_alt_noded_vertices_pgr v
ON (v.id = s.node)
ORDER BY s.path_seq;
```

This was the query originally used to recommend routes for the web application. Dijkstra was later switched for another algorithm (which as the same performance as Dijkstra but allows for additional features that were required for the web application) to

calculate the routes for the web application, going into more detail on the use of this algorithm in section 3.7.4 .

Also, it's possible that some routes created in this system might not always correspond to routes that pedestrians could travel in the real world, because sometimes either the data might be wrong or outdated, or some of the roads selected might not have sidewalks (as the data used for this project, as mentioned, is taken from OSM, a collaborative project that's always being updated and changed) but data accuracy is outside the scope of this project, since it constitutes a proof-of-concept.

### **3.5 Route Recommendation Algorithm**

To recommend routes based on the criteria of proximity to green areas and air quality, I first looked at conventional navigation systems, such as Google Maps and Waze. These systems use pathfinding algorithms which optimize the search for the best path from multiple alternatives, and one of the most used is the Dijkstra algorithm, which guarantees the shortest path (the best among all the possible paths, by minimizing the cost). Cost is the value the algorithm must minimize, and this cost can be based on different criteria, for instance, we can define the length of a path as the cost of that path, and as Dijkstra minimizes the cost, the resulting path will be the one with the least length. According to (Bitesize Data Science, 2019), navigation systems use a more complex variation of this to feed the Dijkstra or A\* algorithm (more optimized version of Dijkstra), with other factors/criteria added, like for instance, traffic conditions, estimated travel cost, etc., which are economist criteria, focused on the time to destination.

For a brief explanation of how Dijkstra works and how it can be used for the HealthyTrack system, (pictured in Figure 35), there is a graph with edges (lines identified by their weights or individual costs) and nodes (the points identified with letters). In the context of a map, the edges represent roads or road segments, and the nodes represent intersections on those roads. Graphs can be undirected (can be crossed in both directions) or directed (only one direction). In my case, my road network is an undirected graph, which means the roads can be crossed both ways, since, for pedestrians, the rules that apply to cars don't generally apply and they can cross any route in any direction.

To select a path from A to F, as was mentioned, all the edges have a weight that gets added to the total cost as nodes are expanded by the algorithm. The chosen algorithm is the one that minimizes the total sum of the edge weights from origin to destination the most. In this case, Dijkstra would select the following path: a-c-e-f (each edge identified in blue, in Figure 35), the shortest path or the one which edge weight sum has the least cost of all the other options, with a total cost of 5.

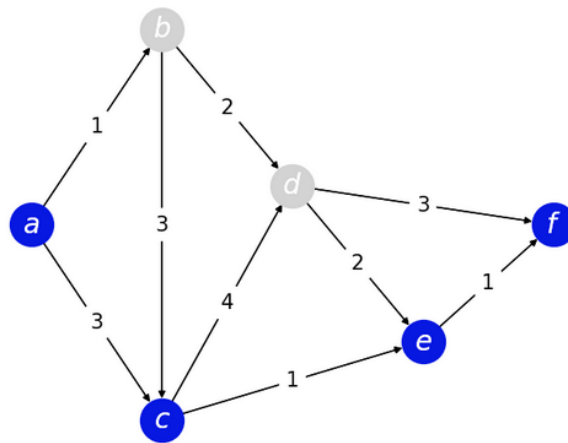


Figure 35 – Graph with edges from a to f. The steps of the optimum path are marked in blue (Bitesize Data Science, 2019).

Other algorithms include A\*, as mentioned, an optimized version of Dijkstra which makes finding the shortest route a faster process, and bidirectional algorithms, which calculate the route starting from both origin and destination node and meet halfway. I could have also produced a solution combining one or more of these algorithms (equally or more efficient, in the case of A\*, than Dijkstra) and/or methods, such as contraction hierarchies, which, according to (Bitesize Data Science, 2019) modify the road network by uniting certain nodes to accelerate the search for a route, and pre-processing of maps (for instance, with identification of key nodes with the use of POIs, facilitating the algorithm’s navigation where there is a high number of those), improving the route and route complexity calculation with user data.

To calculate the route, the current navigation systems use some of these methods, and to consider the route quality, they don’t just take into consideration the length of the segment but also complexity, in terms of node quantity (intersections) that every route has, in the case of routes for drivers and traffic. For this, many services, such as Google Maps, use user data, for instance, about the speed at which they are driving etc., to deduce the traffic, by comparing the car speed to the speed limits of the roads.

The Dijkstra algorithm was selected to calculate routes in this project, without any of the additional methods mentioned, because it is sufficiently effective, since the system focuses on pedestrians (not drivers) and performance isn’t a concern in this case. In future iterations, it might be worth considering the complexity of the route (defined by a higher quantity of intersections) to recommend routes, since in multiple articles referenced (Borst *et al.*, 2009; Golledge, 1994; Novack *et al.*, 2018) it is argued that there is a preference among pedestrians for less complex routes, with fewer turns and intersections.

To achieve the goal of this thesis, the focus was, as mentioned before, placed on the proximity to green areas and air quality as factors that influence the route. As such, the cost will be calculated for these two situations and also for route travel distance (as in



route length). Due to the large amount of data for the whole city of Lisbon (which made measuring the levels of proximity to green areas and air quality impossible, since some calculations would take many days), a specific area of Lisbon was selected for this proof of concept. This area is inside a rectangular shape with the following (WGS 84) coordinates  $(x, y)$ :  $(38.7155420413569828, -9.1544352592029252)$  and  $(38.7387513919086572, -9.1289049735960841)$ , pictured in Figure 38. This matter is discussed in more detail in the 3.6 section.

Currently, there are three different types of routes that are calculated, according to the route criteria that influences them. These are the fast route (based on the length of the segments, which are the edges of the graph), the clean route (based on the length and the air quality along the route) and the green route (based on the length and the proximity of green areas along the route).

For each route  $r$  the cost  $c$  is defined ahead. We consider the segment length  $sL$  for all the routes.

The fast route cost  $c_f$  for route  $r$  is equal to

$$c_f(r) = \sum_{i=0}^n sL_i$$

Where  $n$  = number of edges in route  $r$ .

For the green route cost  $c_g$ , the level of proximity of the segment to the green area  $pG$  is considered. The green route cost  $c_g$  for route  $r$  is equal to

$$c_g(r) = \sum_{i=0}^n sL_i \times pG_i$$

Where  $n$  = number of edges in route  $r$ .

For the clean route cost  $c_{AQ}$ , the mean air quality around the segment  $mAQ$  is considered. The clean route cost  $c_{AQ}$  for route  $r$  is equal to

$$c_{AQ}(r) = \sum_{i=0}^n sL_i \times mAQ_i$$

Where  $n$  = number of edges in route  $r$ .

$sL$  corresponds to the values in the column *length*,  $pG$  corresponds to the values inside the column *greenmeter* and  $mAQ$  corresponds to the values inside the column *airquality\_pos* of the table *estradas\_noded* in the HealthyTrack DB and computed as defined in section 3.3 .

### 3.6 Visualization of Data with QGIS

While building the HealthyTrack system, QGIS was used to better visualize the data, especially how the air quality data was distributed on the area. Then, after the implementation of a routable DB, with the use of the PgRouting plugin for QGIS, which allows for the selection of nodes and request of routes to be drawn on the map, the system and the route recommendation functionality were tested before implementing the web application.

Figure 36 displays the representation of the HealthyTrack geometries on QGIS, having on display the Lisbon boundary layer with all the areas of Lisbon over the air quality, green areas and road network of Lisbon layers, on top of the OSM layer. Figures Figure 17, Figure 24, and Figure 28, in section 3.1.2 , already present each of the isolated data layers reunited here.

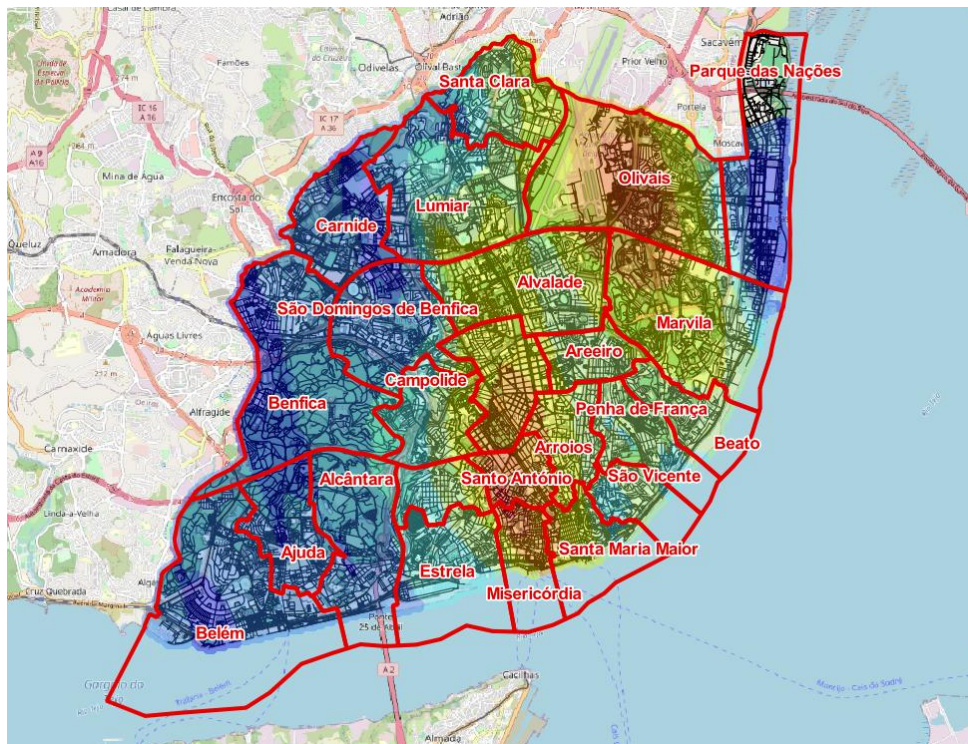


Figure 36 – Representation of the HealthyTrack spatial data in QGIS.

Here, you can see that the air quality data doesn't cover the whole city of Lisbon, missing some part of Parque das Nações. It also isn't visible unless zoomed in, but what looks like a cohesive gradient, or a "heatmap" representing the air quality of Lisbon is formed by small points which are in a grid like format, equidistant, at 5 meters from one another.

The air quality data representation was configured to simulate a heatmap by defining a color for each level of pollution, in 9 level scale (with a range of pollution values divided equally) which ranges from low (-91,4185) to high air quality (35,9589), as is visible in

Figure 37, in the QGIS layer properties edition screen, according to the original model of this air quality dataset, which tells us how to interpret this data (Figure 16). The colors were also inverted (as you can see in Figure 17) for a more intuitive visualization, but still based on the original scale (Figure 16), as the colors in the original scale went from red (cleanest) to blue (least clean).

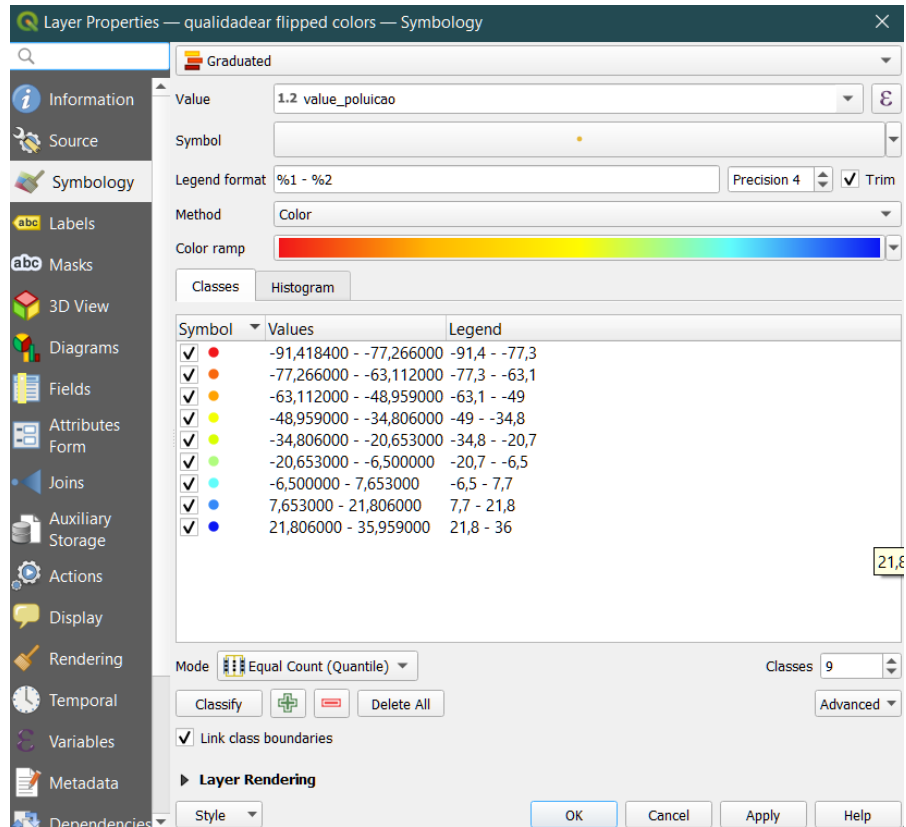


Figure 37 – Customization of the air quality layer in QGIS, with the inverted color range.

A better method to create a heatmap of air quality than the one described previously was then discovered and used, via an interpolated layer of the air quality values, by which you can generate a raster layer.

In Figure 17 you can visualize the interpolated air quality raster displayed over the city. The advantages from this method as opposed to the previous one is that the raster takes significantly less time and performance to load, when compared with the point layer, as you zoom in and out (since you don't have to load many *point* geometries, only one picture) and, unlike the point layer, where you might need to resize the points to maintain the same representation when you zoom in, the raster layer, as an image, doesn't have to be resized and can always be recreated with a higher number of pixels to achieve a smoother appearance.

As mentioned in section 3.5 , originally, the system was to accommodate the whole city of Lisbon, however, during the measurements of the average air quality, which involved joining two tables (*air quality* and *edges* tables) to calculate the average

pollution around 50 meters to every side of each edge, the volume of air quality data was too vast for measuring an average around all edges.

Among other solutions, one was to limit the data volume so PostgreSQL wouldn't have to process so many geometries to calculate the average pollution for every edge and thus run the query in less time, which could be done in some ways, like, for instance, reduce the "resolution" of the pollution values, interpolating them to make them more spaced out, to have a resolution of 10 or more meters, instead of 5. The other, simplest solution was to simply reduce the scope for the proof-of-concept of this project, which was the solution selected.

In Figure 38 you can see the new scope in context, and zoomed in, on Figure 39. It covers mostly the Arroios and Santo António areas of Lisbon, with a significant portion of Avenidas Novas and some of the other surrounding areas. This place was selected since it is in the center of Lisbon, where there are significant visible variations in the air quality and a significant number of green areas exists.

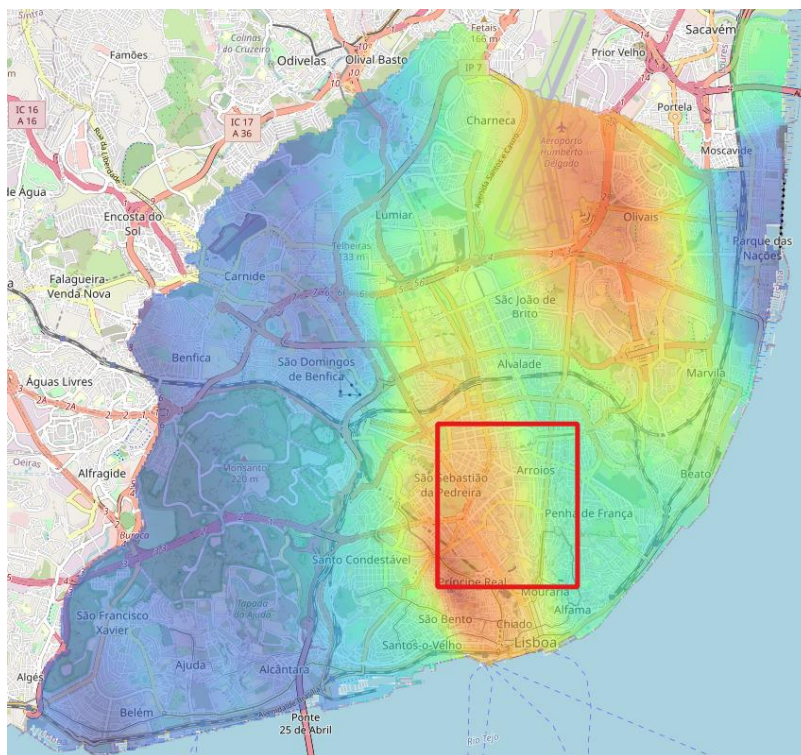


Figure 38 – Outline of the boundaries (in red) for the selected boundaries in QGIS.



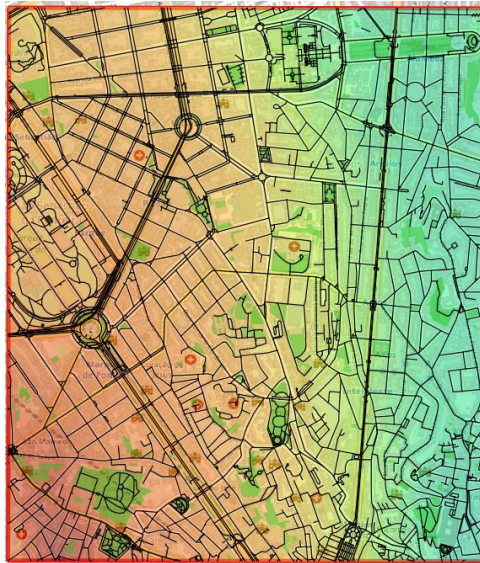


Figure 39 – Zoomed in display of the selected Lisbon area for the proof-of-concept, in QGIS.

After reducing the proof-of-concept area to work on, the average levels of air quality were able to be calculated in working time. After calculating the green, clean and fast cost for each edge, and in pgAdmin, building the routing functionality, the PgRouting plugin was installed in QGIS, allowing for the generation and visualization of routes.

Below are examples of the three types of route of the HealthyTrack system, for the same origin-destination pair, generated on QGIS, with a caption added to each figure that contains the air quality color scale, for easy analysis: the fast route (Figure 40), the green route (Figure 41), and the clean route (Figure 42).



Figure 40 – Fast route in QGIS.



Figure 41 – Green route in QGIS.



Figure 42 – Clean route in QGIS.

### 3.7 Web Application

After building the HealthyTrack system, there needed to be a way for users to interact with it. For this a web application was built, which provides interactive maps for the end user, where they can mark their desired route origin-destination pairs and get recommendations, based on the different criteria already discussed, with seamless visual feedback.

As mentioned in section 3.2 , the route recommender web service was built with ExpressJS for the server and Leaflet on the client, for map interaction.

To make this application work seamlessly, without the need to load a new page on requests, AJAX was used. Its use was simplified with the native JS API called *fetch*, introduced in ECMAScript 2015 (ES6), the most recent JS revision. This API is bound to replace the former *XMLHttpRequest* and is not supported in old browsers such as Internet

Explorer. It is based on *Promises*, which enable a simpler and cleaner API, reducing the lines of code to predict callbacks.

A *Promise* is an object which can be returned synchronously from an asynchronous function. It can have three statuses: fulfilled, rejected or pending. Callbacks can be created to deal with either the status of fulfillment or rejection. The promise guarantees that a task begins being completed even if the *promise's* status is pending. The advantage of this is allowing the management of multiple asynchronous operations at the same time in JS.

I will be going through the features of this web application, the research behind the decision for the server-side software, its advantages and characteristics, and behind the decision to use the Leaflet library for the interactive map interface. Then, I will be documenting the main aspects of the business logic, the main logic that had to be implemented in order to display the recommended routes to the main user.

### 3.7.1 Features

The main feature of the web application (Figure 43) is the route recommender. The input for this can be made in three ways: inserting points (origin, destination) through clicks on the map; inserting the origin and destination coordinates in the text input next to the map (in the indicated format); or by inserting addresses on that same text input (as shown in Figure 43). After insertion, the coordinates for that exact location are obtained through an API call to the OSM servers and, in case the address is ambiguous, which will lead to more than one location being found in the servers, the first of the results for that same input is selected and the full address is shown under the text input.

Healthy Track - Route recommender

Select the origin and destination (lat and lon or address) of your trip inside the boundaries.

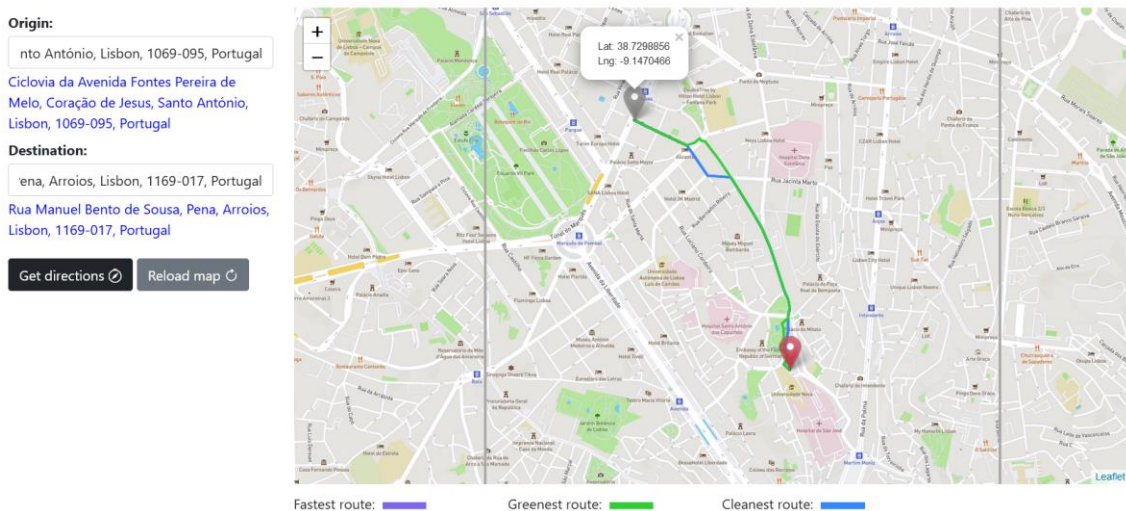


Figure 43 – Healthy Track web application interface in the browser.

After selecting the coordinates and pressing the “Get directions” button, the output is presented on the map, where three variations of the route are displayed, in different colors: green for greenest route, blue for cleanest route and purple for fastest route; you can also press the “Reload map” button to reload the map so you can request another route.

The user may also hover the mouse on the markers placed on the map to visualize the coordinates of each and can click on the lines that identify each route in the map caption to reveal them on the map in case another route overlaps that one. An example of this is pictured in Figure 44, where the “greenest route” in green and the “cleanest route” route in blue are visible, from origin (grey marker) to destination (red marker). The “fastest route”, in purple, isn’t visible since it is hidden under the cleanest route. By clicking on the line, displayed on the caption, that represents a route, the user can bring that route layer to the top, making it visible, as in Figure 45.

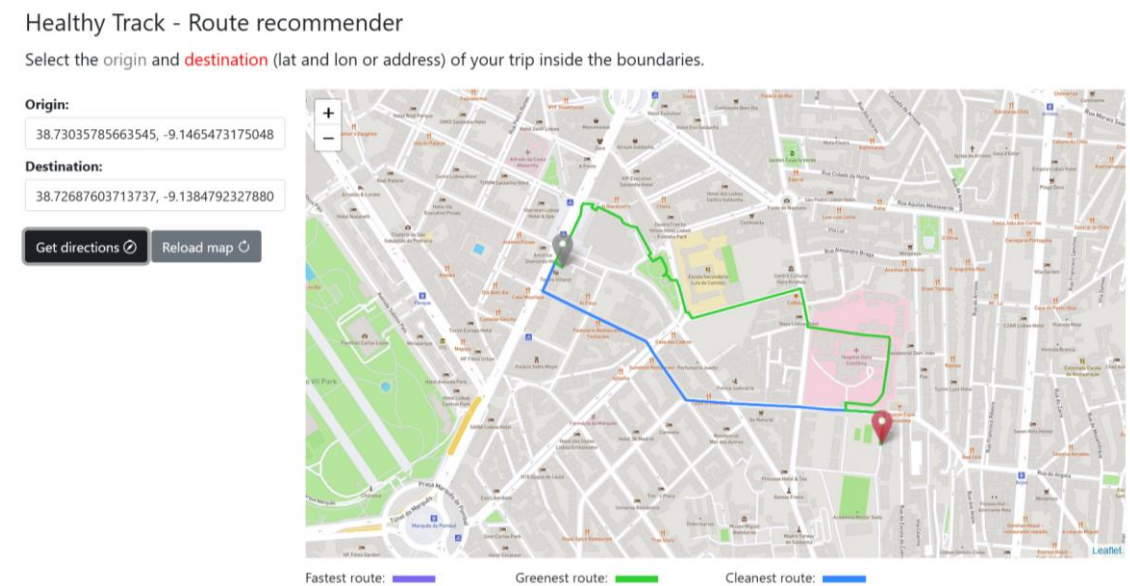


Figure 44 – Healthy Track web application interface, with the fastest route hidden under the cleanest route.



## Healthy Track - Route recommender

Select the origin and destination (lat and lon or address) of your trip inside the boundaries.

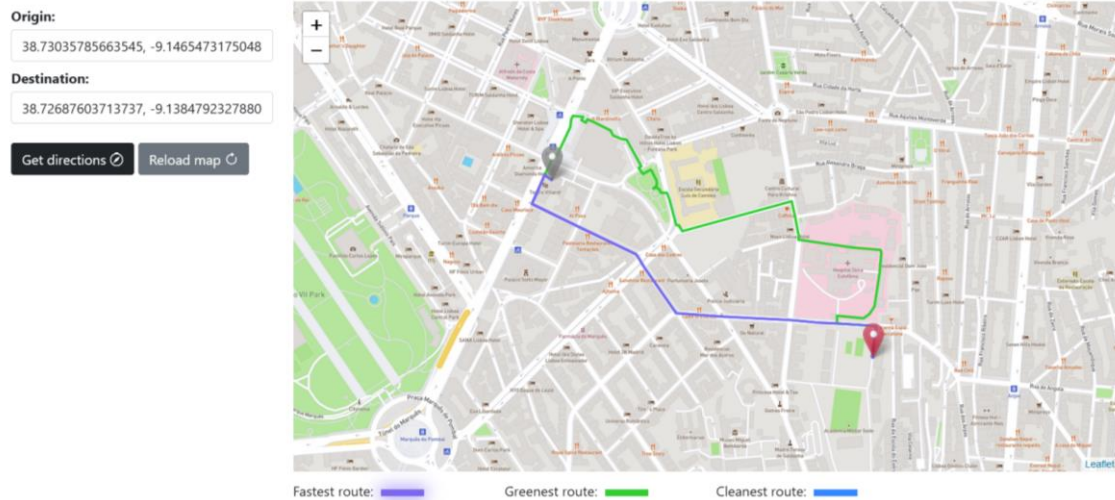


Figure 45 – Healthy Track web application interface, now showing the fastest route instead of the cleanest.

### 3.7.2 NodeJS Server

To build the web application, a backend language to run the server-side application of the route recommender needed to be selected.

Some of the most popular backend frameworks that exist are based on the following languages: Python, such as Django; JS based, such as ExpressJS; PHP based, like Laravel; Ruby on Rails, which is based on Ruby; and Phoenix, which is based on Elixir.

For this project, ExpressJS, one of the most widely used open-source frameworks for the NodeJS run-time environment, was selected. NodeJS is asynchronous and event-driven, single-threaded with event-looping, making it highly scalable and very fast because it's built on Google Chrome's v8 JS engine. It's good for simple, lightweight applications. The ExpressJS framework simplifies setting up web applications quickly, is very lightweight when installed but easily extensible and future-proof. The *node-postgres* plugin was used on the server side to allow for queries to be made on the HealthyTrack DB.

### 3.7.3 Interactive Maps with LeafletJS

For the client-side, adding features for interacting with maps also involved a process of exploration and analysis of the tools available, to select the best one for the project.

The Google Maps API was the first tool for interactive maps on the browser to be considered for this project's implementation. This API offers more advantages for drivers, with the most complete traffic and transit information of all the available options.

Other interactive map APIs include the Bing Maps and Here Maps, which provide similar functionality.

The most popular open-source alternative is Leaflet, a lightweight library for JS mobile-friendly interactive maps, which gets further added functionality from third party services. For instance, you can use map tile layers from sources like OSM, Mapbox <sup>15</sup> (for instance, in this project, the streets tile layer from Mapbox is being used as the base layer), etc. and, for basic routing functionality, there exist multiple routing services such as the Leaflet Routing Machine <sup>16</sup> (LRM), based on OSRM and GraphHopper <sup>17</sup>.

Other alternatives for interactive map functionality also include: Mapbox.js (built on top of Leaflet) which uses raster tiles for display; Mapbox GL JS, a standalone library which will be replacing Mapbox.js, displays vector tiles, uses JS and WebGL for better performance when drawing data, but is a relatively new technology as opposed to Leaflet, requires WebGL to work and doesn't support all browsers; MapLibre GL JS <sup>18</sup>, which is an open-source version of Mapbox; another not very mentioned, but still great choice is OpenLayers<sup>19</sup>, which is a more mature, fully featured library for mobile-ready interactive maps with the use of OSM that natively supports vector tiles unlike LeafletJS, is used by those who are generally more knowledgeable of GIS, is not as lightweight as Leaflet, though it has become more lightweight over the years, coming very close to the small size of Leaflet, is less high-level and has a more complex and verbose documentation, so it requires more time and effort to learn, but is more flexible and still extensible, with a generally helpful and more experienced (although smaller) community when compared to Leaflet. All these options, including Mapbox GL JS up to v2.0, are open-source.

Leaflet ended up being selected for the client-side interactive maps, because it has one of the most comprehensible and organized documentations of all the options presented. It is also lightweight, fast and responsive, open-source, widely supported, and extensible through the use of third-party services, unlike Google Maps.

### 3.7.4 Business Logic

In this section I go over the main logic of the application, focused on the recommender feature. I will be listing the main operations behind the process of the recommendation of a route, from the moment a user requests directions to when the

---

<sup>15</sup> <https://www.mapbox.com/>

<sup>16</sup> <https://www.liedman.net/leaflet-routing-machine/>

<sup>17</sup> <https://www.graphhopper.com/>

<sup>18</sup> <https://maplibre.org/>

<sup>19</sup> <https://openlayers.org/>

application outputs the results. I will also be identifying the main issues that had to be resolved in the context of the web application.

After the origin and destination are picked and the user presses “Get directions”, the insertion is validated, and the type of insertion (address, coordinates) is verified through pattern matching. In case the user inserted a readable address, that address gets turned into coordinates through a call to the Nominatim server, which allows to fetch the coordinates of a given address (only the closest result is returned so the input address shouldn’t be ambiguous), and as soon as the coordinates are returned, the operation proceeds.

Following this, the client requests the server for the *getEdge* web service, to obtain the closest edges to the origin and destination, along with other elements of each (id, source and target node ids and geometries, geometry of the closest point to the origin/destination point inside the closest edge and total distance from this point to the origin/destination).

To draw a route from two selected points on the map (such as in Figure 46), we have two issues: the issue of which node from the closest edge to select as the origin/destination node and how to “connect the dots” from selected point to edge to node to route (for the origin and destination). In Figure 47, we can have a look at the graphical representation of a route for the web application, from origin to destination, and in Figure 48, we have a more detailed view of the issue we want resolved in the same representation.

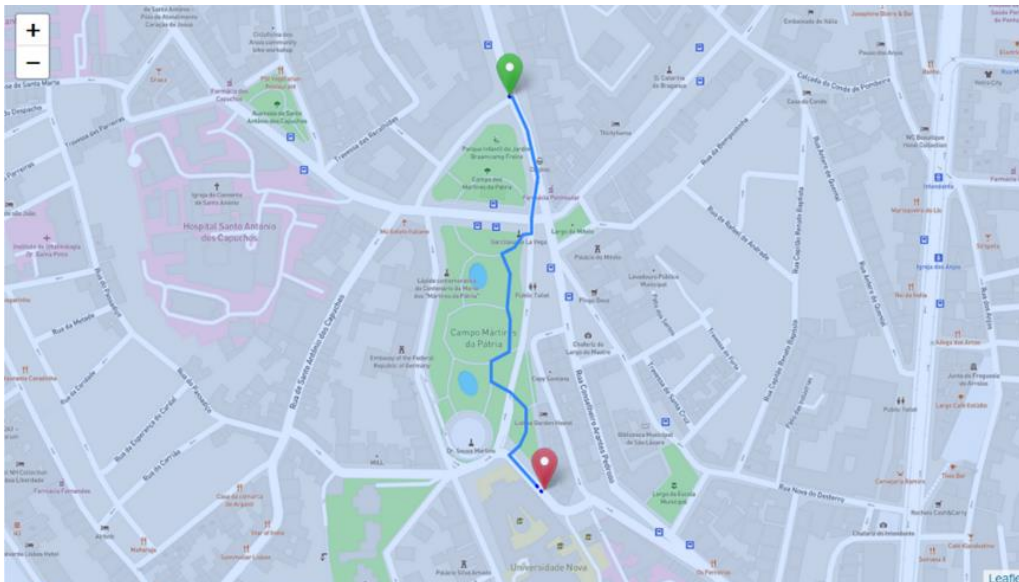


Figure 46 – Normal route without the previous issue, viewable in the web app.

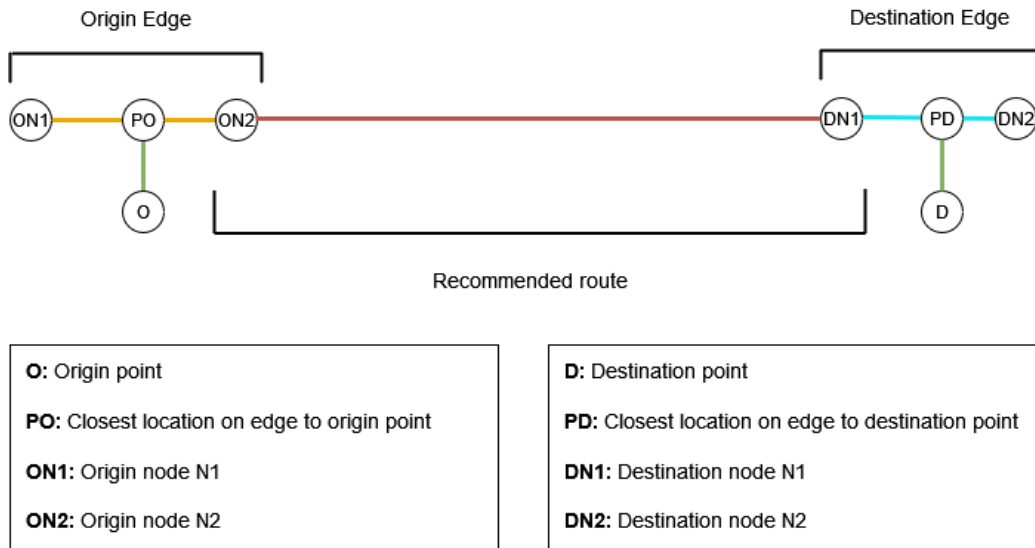


Figure 47 – Graphical representation of a route.

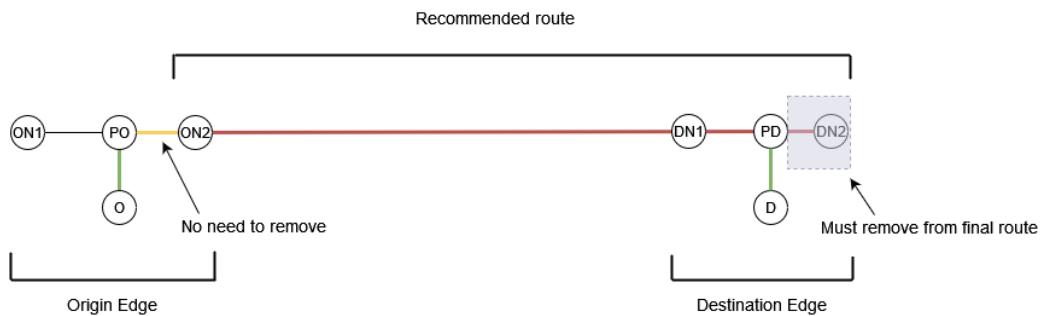


Figure 48 – Graphical representation of the issue to be resolved.

The solution for this involved selecting the closest node (as the origin/destination node of the route), and then to draw the whole route from origin to destination points, discarding the end of the edge outside the route (example of the issue in Figure 49 and a route drawn without the issue in Figure 46) by removing the part further away to each end node than the closest point to origin/destination. This is a flawed solution because it might happen that the closest point on the edge to the extremity is further away from the route origin/destination node than a part of the origin/destination edge that you want to remove.

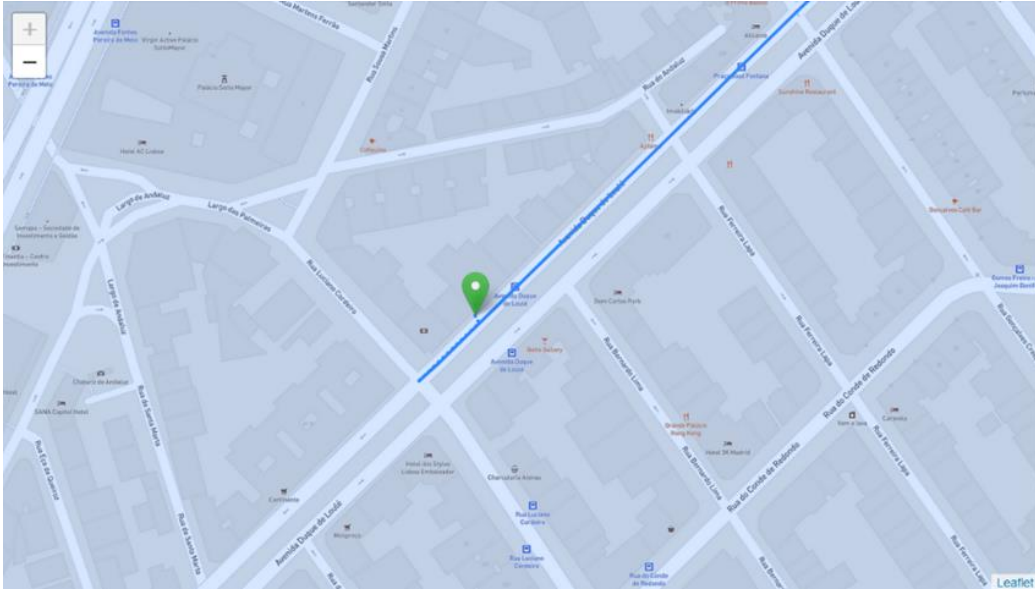


Figure 49 – Route drawn in the web app. The line before the start of the route must be removed.

Another solution (pictured in Figure 50) involves calculating 4 routes with Dijkstra in PgRouting for the 4 different combinations of origin and destination nodes (for the same edge), which include ON1-DN1, ON1-DN2, ON2-DN1 and ON2-DN2 (according to the caption in Figure 47). The aggregated costs between the 4 different combinations are compared, and the one with the shortest cost is selected, which solves this issue when it comes to the fast route, because the one selected will always be the shortest, in terms of distance. When it comes to the green and clean routes, since the cost isn't entirely based on the distance/travel time, the issue might still happen.

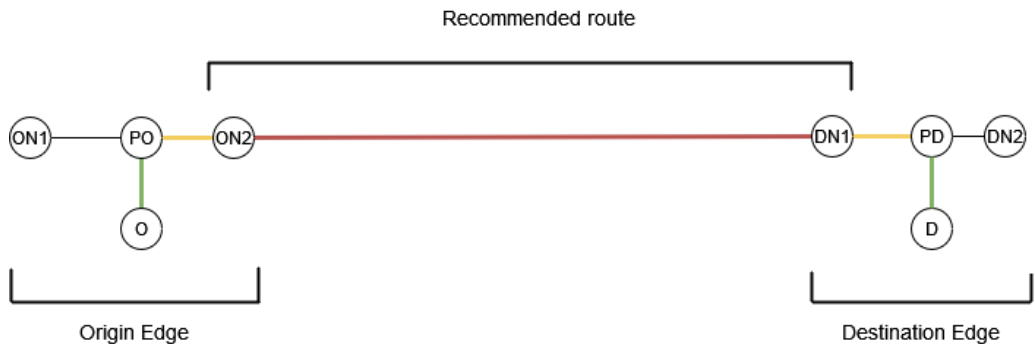


Figure 50 – Graphical representation of the 4 Dijkstras solution.

After some further research, it was found that some existing tools already help fix this issue. One of them is a pathfinding algorithm for PgRouting, called Turn Restriction Shortest Path (TRSP), which is nearly as optimal as the A\* algorithm, performance-wise, and has many additional features such as allowing to define turn restrictions, as the name suggests. It also allows to recommend a route with an input of origin-destination edge pair instead of nodes, selecting the best starting and ending node by itself, and it also allows to set the fraction of the extremity edge that is closest to the selected point on the map, with a percentage that goes from 0 to 1 starting from the beginning of the *linestring*

(source of an edge), which means that the closer to 1, the closer to the target of that edge that location is on that same edge, and vice-versa.

However, to create the actual route starting and ending from the point closest to the selected points on the map (as TRSP, like any of these algorithms, merely returns the references to the input tables, meaning the ids of edges/nodes of the route, and not a route starting from the points we want), you need a more complex query, or a set of smaller queries where you cut the extremities based on those fractions (which can be interpolated into a location that can then be used to cut a substring of the original edge, effectively removing the superfluous part of the route). After calculating the route, you can then connect the end substrings to that route, after removing the edge from each end of the route. A diagram of this solution is displayed in Figure 51.

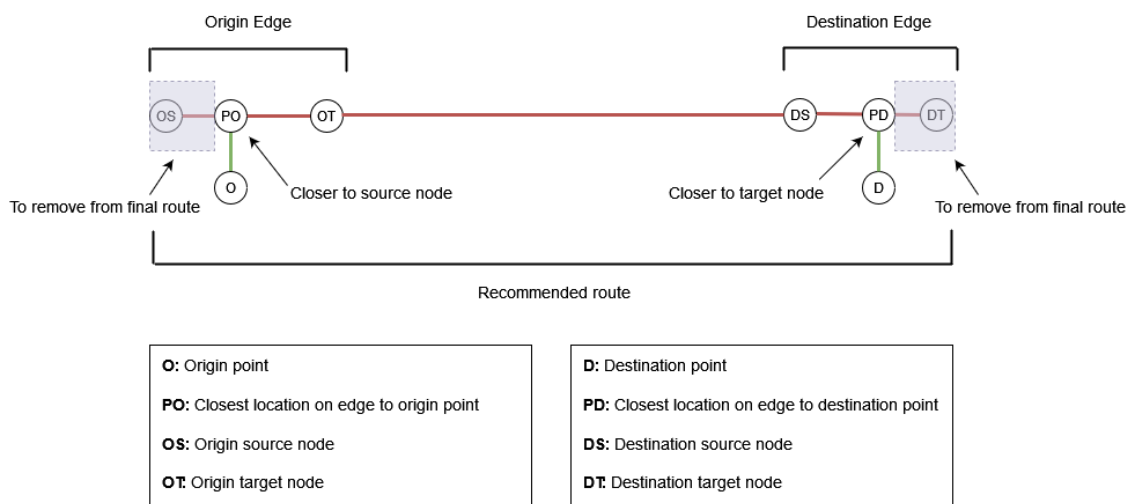


Figure 51 – Graphical representation of the TRSP solution.

As such, a solution using this algorithm was created, with a query calculating a route with TRSP, and another to create a substring of each extremity, and then these were connected together. To conclude, with this solution you don't have to check which edge is closest to the selected points on the map, and the shortest path is guaranteed.

Below is an example of the query used to provide the recommended route results in the web application, with the use of TRSP, and the separate query used to cut a substring for each end of the route.



```

-- trsp
SELECT s.seq, s.id1 as node, s.id2 as edge, s.cost, b.source,
b.target, ST_astext(b.geom) as edge_geom_as_text, b.geom as
edge_geom, ST_astext(v.the_geom) as node_geom_as_text
FROM pgr_trsp(
  'SELECT id::INTEGER, source::INTEGER, target::INTEGER,
length::DOUBLE PRECISION AS cost FROM lisboa.estradas_alt_noded',
  999,
  (SELECT ST_LineLocatePoint(geom, st_GeomFromText('POINT(-
9.151113241919122 38.73199952547669)',4326)) FROM
lisboa.estradas_alt_noded WHERE id=999)::DOUBLE PRECISION,
  999,
  (SELECT ST_LineLocatePoint(geom, st_GeomFromText('POINT(-
9.151097148665032 38.731991156114084)',4326)) FROM
lisboa.estradas_alt_noded WHERE id=999)::DOUBLE PRECISION,
  FALSE,
  FALSE) AS s
LEFT JOIN lisboa.estradas_alt_noded b
ON (b.id = s.id2)
LEFT JOIN lisboa.estradas_alt_noded_vertices_pgr v
ON (v.id = s.id1)
ORDER BY s.seq;

-- query for the edge substring
SELECT e.geom as e_geom,
ST_LineSubstring(e.geom, start_fraction, end_fraction) as
edge_substring_geom, st_astext(e.geom) as e_geom_as_text,
st_astext(ST_LineSubstring(e.geom, start_fraction, end_fraction)) as
edge_substring_as_text,
st_astext(ST_LineInterpolatePoint(e.geom, fraction)) as
closestpoint_as_text
FROM lisboa.estradas_alt_noded e
WHERE e.id=3036;

```

Another solution found, which wasn't implemented, is based on a yet unofficial family of functions from PgRouting, called *withPoints*, which allows you to route between arbitrary points located outside the original graph, meaning that, just like TRSP, you can provide it with the fractions in starting and ending edge which are closest to the locations selected on the map.

However, you must provide the algorithm with a row of results of a table including those fractions, which maybe can be solved by either creating a permanent table for this purpose, inserting the new locations as users click the “get directions” button, or a temporary table for one session, which means you would have to make two queries per session, one to save those locations and another to call the *pgr\_withPoints* function, which is another challenge that would require some changes to query functions.

The advantage of the *withPoints* solution is that, unlike all these other solutions, it doesn't require complex code or queries, cutting and joining the ends of the route, returning a formed route from origin to destination points (which only has to be connected to respective selected points in map) simplifying the business logic side.

All in all, considering the solutions presented, the TRSP algorithm provided the best working solution that effectively solves the issue discussed, considering the *withPoints* solution is yet unofficial.

### 3.8 Experiments

After building the HealthyTrack system, experiments were run to analyze the project's green and clean route recommendations against the routes recommended by the Google Maps and OSM navigation systems. We show two experiments, which showcase the difference between the alternative routes to the fastest route. For each of these experiments, the origin-destination pair was the same.

After comparing the results in the first experiment, we can see that the fastest route in the HealthyTrack web app (Figure 52) is similar to the ones recommended by the other navigation systems (Figure 53 and Figure 54), since it is focused on minimizing distance/time of travel.

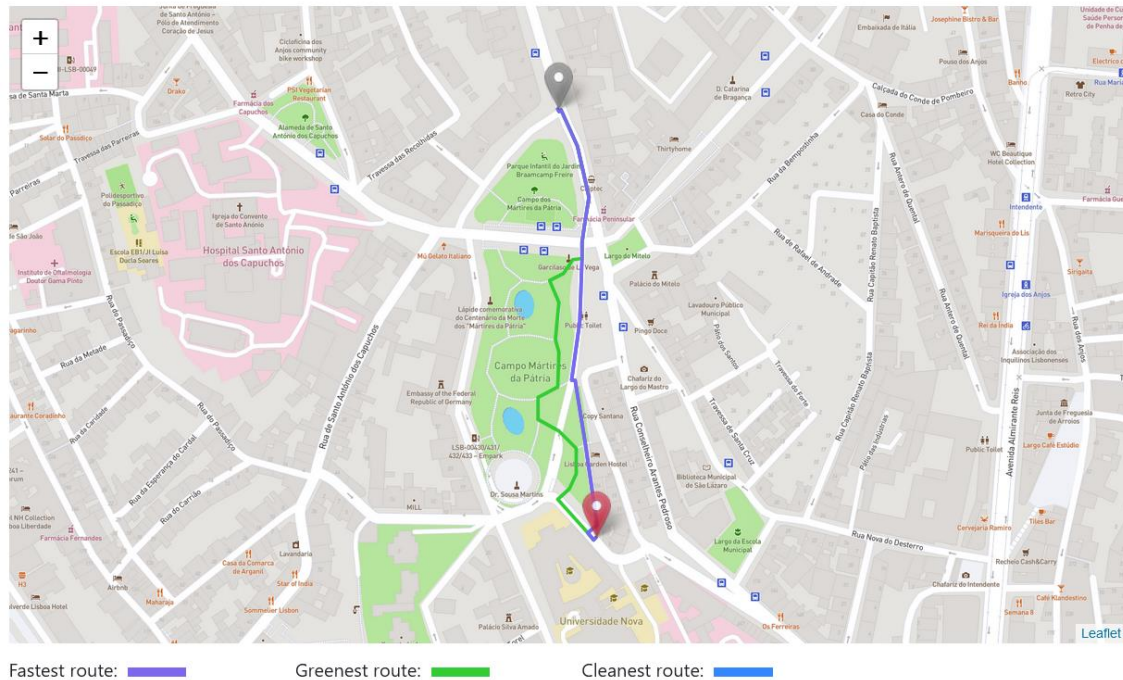


Figure 52 – First experiment results in HealthyTrack.



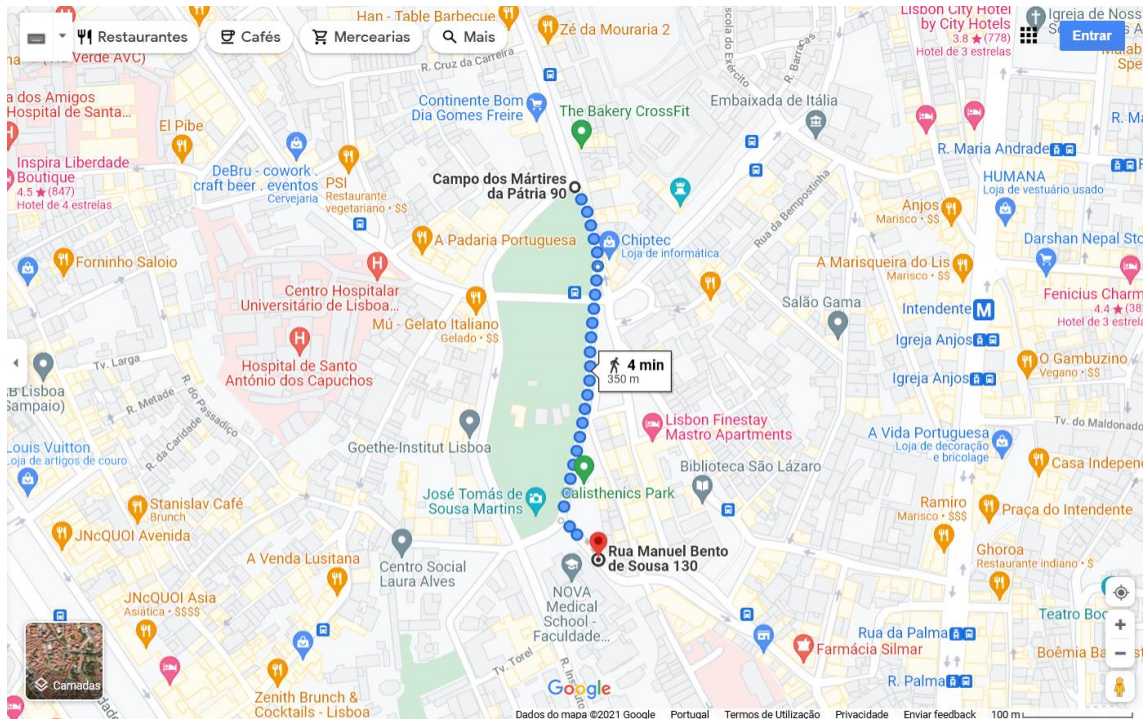


Figure 53 – First experiment results in Google Maps.

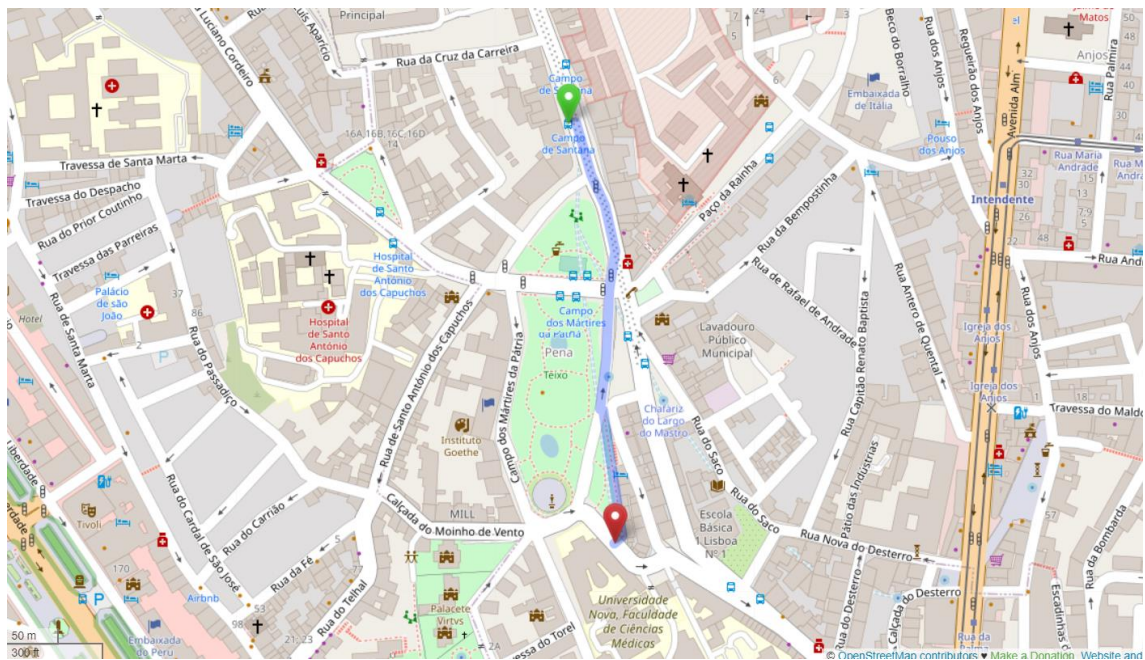


Figure 54 – First experiment results in OSM, using OSRM.

In contrast, the green route, by prioritizing proximity to green areas, goes through the green area pictured. In this case, there is no apparent variation in the clean route from the fast route, because, as we can see in Figure 55, the area travelled already minimizes low air quality, but a second experiment was also run, where visible changes to the route travelled, based on air quality, are more visible.



Figure 55 – Fastest to cleanest (from left to right) routes from the first experiment on HealthyTrack.

In the second experiment, when comparing the clean route to the fastest route (Figure 56 and Figure 57), it is possible to conclude that this route minimizes lower air quality.

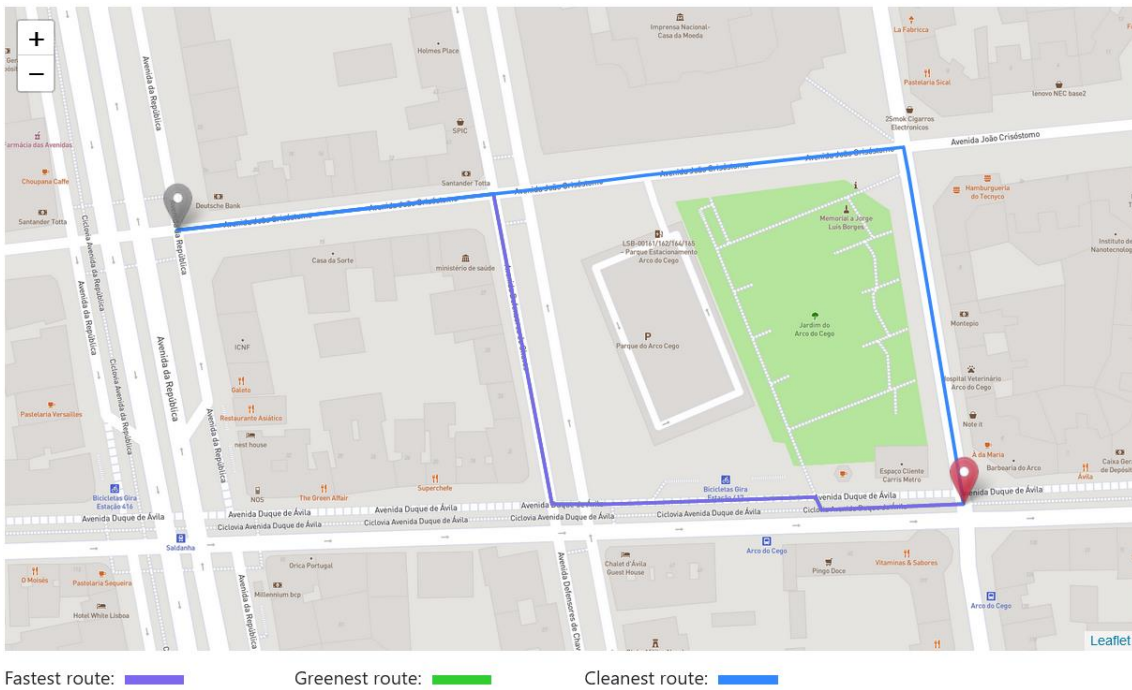


Figure 56 – Second experiment results in HealthyTrack .



Figure 57 – Fastest to cleanest (from left to right) routes from the second experiment on HealthyTrack.

The fastest route, by minimizing the distance/time of travel is once again very similar to the directions provided in common navigation systems (the route recommended in Google Maps, visible in Figure 58, is slightly different in this case because our system, which is based on OSM, lacks the edge that travels through the park in Google Maps, as you can confirm in Figure 59, since the result is the same in OSM through a different routing service).



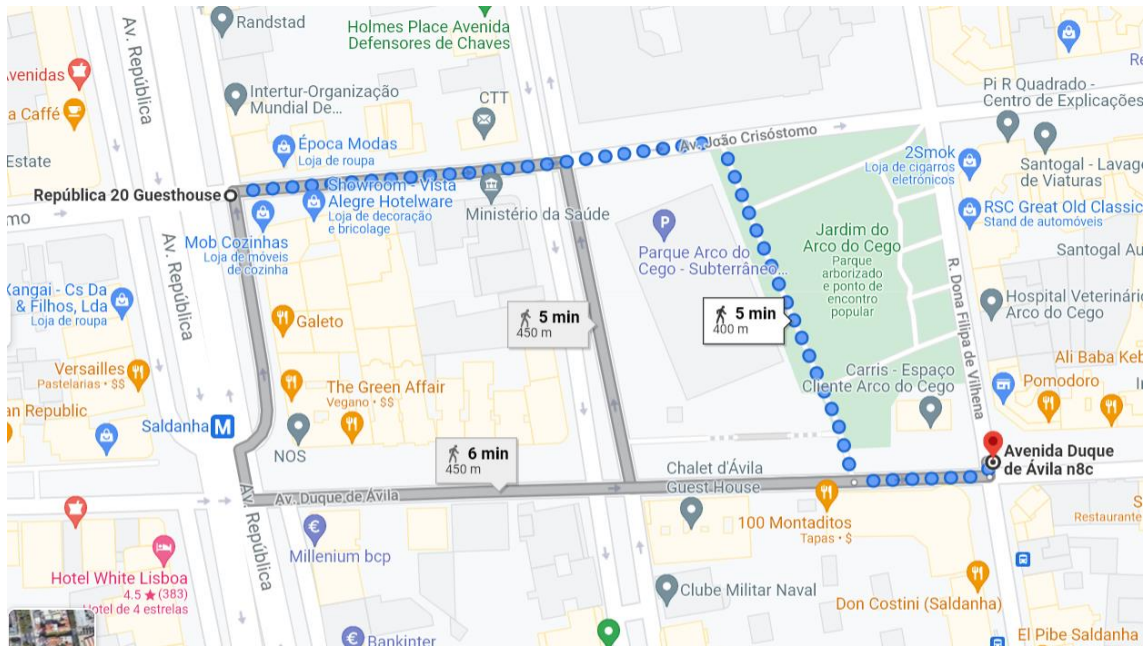


Figure 58 – Second experiment results in Google Maps.

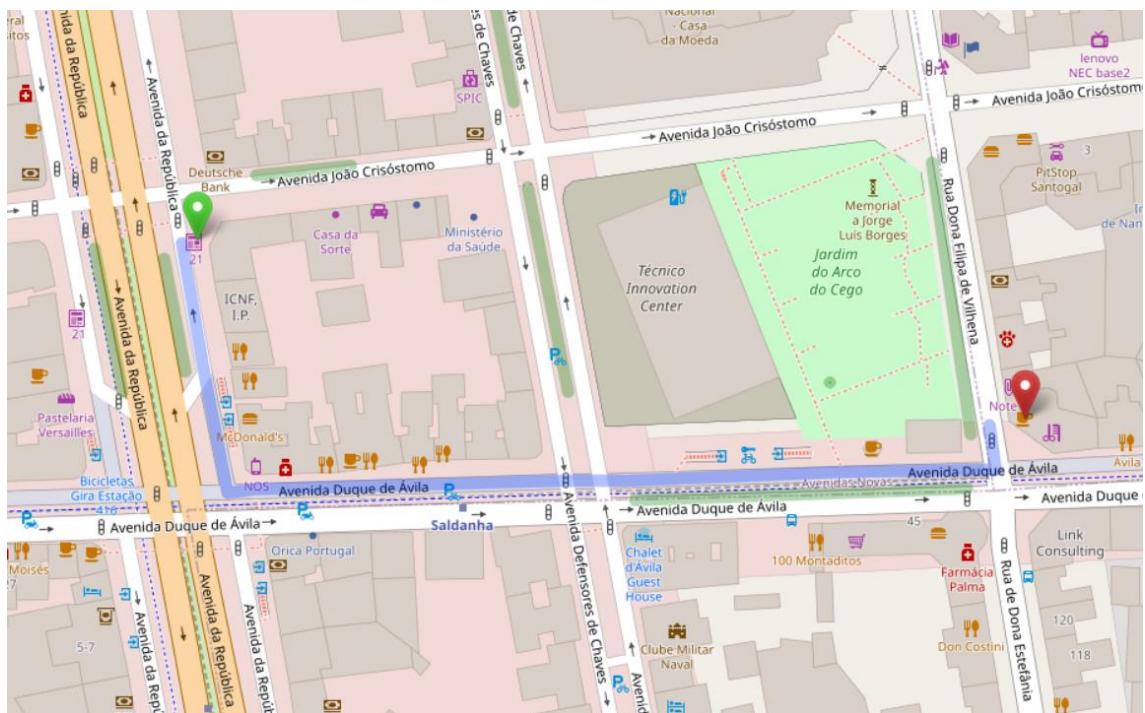


Figure 59 – Second experiment results in OSM, using OSRM.

### 3.9 Summary

In this chapter, I presented the HealthyTrack system, detailing its architecture, listing its features and exploring the core facets of its functionality. I then ran some experiments to analyze the system’s route recommender.

# Chapter 4

## Conclusion and Future Work

As the world gets more connected, it is becoming the perfect environment for innovation in many areas which rely on connectivity and data, such as the route recommender systems area. As it stands, most commercial navigation systems nowadays, apart from having a focus on drivers, transit and traffic, offer pedestrians route recommendations that prioritize shorter lengths of travel and less time from point A to B, when additional aspects of the routes could also be prioritized.

This work proposed a solution of a route recommender system for pedestrians which is focused on the individual (specifically on their health) and promotes walking, by selecting routes which pass through green areas and cleaner air, aspects that are generally linked with the promotion of people's health.

The main contributions of this work include: a review of the recommender systems state of the art and taxonomy, of the pedestrian navigation literature with focus on the user preferences and a discussion of current pedestrian route recommender systems; the HealthyTrack DB, which stores spatial data to measure different aspects of Lisbon's routes (air quality, green areas, road network), used to calculate healthy routes, with the use of the PgRouting extension on the same DB; the HealthyTrack web app connected to the previously mentioned DB, and providing an interactive web map interface with Leaflet, which allows for interaction with the route recommender system, so that users can get their recommended routes.

I first reviewed the state of the art of recommender systems, from which the main takeaway is that they are divided into three types, according to their recommendation approach: content-based, where the recommendations are based on similar previously liked items; collaborative, where recommended items are those that users with similar tastes have liked; and hybrid, which is a mix of the previous two.

I then went over the existing literature on pedestrian navigation systems, allowing me an understanding of the general preferences of pedestrians, to develop a notion of what makes them prioritize a route over another. I found that, generally, in their routes, users prioritized shorter lengths, better aesthetics, more greenery, security and less route complexity (less intersections).

After this, I researched about existing implementations of route recommender/navigation systems with similar goals in mind. I discovered two articles,

one which documents the development of a prototype, on the CupCarbon platform, for a pedestrian route recommender system that is meant to work in a smart city, which implies good network connection, based on real-time data (on illumination, green areas, access to and wi-fi quality etc.) collected from a network of sensors spread across that same city; in another the authors developed a technology-agnostic methodology, with a web app prototype that allows the user to get recommended routes which avoid pollution, they do this by obtaining air quality sensor data, which allowed for the creation of “barriers” for the areas of lowest air quality of the city, which the system avoids, in that way generating routes that are pollution-free; the other article documents a similar system, fully based on open-source data from OSM, which provides “pleasant” routes based on the user’s preferences, offering an interface that allows users to define their priorities on a route and generating a customized route.

I then moved on to implement the HealthyTrack system, which is a PostgreSQL DB that stores the spatial data necessary to measure different aspects of the routes, values which are then used to recommend routes, with basis on the route criteria selected, such as road length or air quality of the surroundings. This DB contains the road network, average air quality values and green areas of the city of Lisbon, in addition to other data, and allows me to recommend faster, greener or cleaner routes. I visualized and edited extra information through the QGIS software.

Finally, I implemented a web app for the HealthyTrack system, a NodeJS based web application that is connected to the PostgreSQL DB previously mentioned and provides an interface with an interactive map, which uses the Leaflet library, where the user can input the origin and destination of a route and then get the output recommended routes for each criterion (fastest, cleanest and greenest) drawn on the map.

I then made some experiments where I compared the performance of the HealthyTrack system to other navigation systems in terms of the routes recommended. I ended up concluding that the greener and cleaner routes, when compared to the fastest route, as planned, recommend distinct routes that prioritize their respective route quality criteria.

Along the project, I prioritized the use of free and open-source software. For the DB, as mentioned, I chose PostgreSQL software along with the PostGIS and PgRouting extensions for spatial data management, and QGIS for visualization and other advanced options, which are all open-source and maintained by volunteer work. For the web app, I chose JS for the server by using NodeJS, licensed under an open-source license. The data I used for this project, ranging from spatial datasets from OSM to the ones from the CML public spatial data repository, apart from the air-quality, is free and open-source, being accessible online to everyone. It was interesting to find that most of the software out there

that deals with GIS, apart from ArcGIS, which is one of the most used systems, is all open-source, and the open-source options are, if not better in certain areas, on par with the paid and closed-source systems. In sum, I had an easy time using these systems, apart from the fact that my machine wasn't enough for certain DB calculations, and I might have benefitted from cloud computing.

Apart from these other conclusions, for future work, the cost formulas for the greener and cleaner routes need to be improved for better recommendations and additional route quality criteria could be added, such as security aspects of the route, like levels of illumination. General weather and time of day might also be route criteria to have in consideration (for instance, avoiding exposure to too much UV light while also avoiding the rain, and at night higher levels of street illumination prioritized), along with aesthetics of the route, complexity of the routes (general number of intersections), quality of pavement, levels of inclination of the path along the route and levels of noise pollution. Other important additions to the system also include the possibility of trip chaining (addition of multiple stops) and the possibility to combine multiple quality criteria in the same formula for multi-criteria routes, with the ability for the user to customize these routes, according to their preferences. Furthermore, aspects to fix on the interface of the web application and to add to future implementations are, perhaps, the addition of a tutorial for use of the tool, for beginners, for context on the features and on what each route prioritizes, and information about each route on hover or click (for instance, length of route, complexity, etc.).

## References

- Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749. <https://doi.org/10.1109/TKDE.2005.99>
- Alfonzo, M. A. (2005). To Walk or Not to Walk? The Hierarchy of Walking Needs. *Environment and Behavior*, 37(6), 808–836. <https://doi.org/10.1177/0013916504274016>
- Armeni, I., & Chorianopoulos, K. (2013). Pedestrian navigation and shortest path: Preference versus distance. *Workshop Proceedings of the 9th International Conference on Intelligent Environments IE'13, July 16-19, 2013, Athens, Greece*, 647–652. <https://doi.org/10.3233/978-1-61499-286-8-647>
- Bitesize Data Science. (2019). Navigation Apps: The Algorithms that Get You from Here to There. Retrieved December 28, 2021, from <https://bitesizedatascience.com/2019/01/31/navigation-apps-the-algorithms-that-get-you-from-here-to-there/>
- Borst, H. C., de Vries, S. I., Graham, J. M. A., van Dongen, J. E. F., Bakker, I., & Miedema, H. M. E. (2009). Influence of environmental street characteristics on walking route choice of elderly people. *Journal of Environmental Psychology*, 29(4), 477–484. <https://doi.org/10.1016/j.jenvp.2009.08.002>
- Burke, R. (2013). *Knowledge-Based Recommender Systems Knowledge-based recommender systems*. (August), 167–197.
- Chakraborty, B. (2012). Integrating awareness in user oriented route recommendation system. *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 1–5. <https://doi.org/10.1109/IJCNN.2012.6252543>
- Czogalla, O., & Herrman, A. (2017). Parameters Determining Route Choice in Pedestrian Networks. *17th ITS World Congress*, 1–12.
- Fernandes, S., Carvalho, P., & Rito Lima, S. (2019). Pedestrian Route Recommendation System in Smart Cities. *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6. <https://doi.org/10.23919/CISTI.2019.8760875>
- Giles-Corti, B., & Donovan, R. J. (2003). Relative Influences of Individual, Social Environmental, and Physical Environmental Correlates of Walking. *American Journal of Public Health*, 93(9), 1583–1589. <https://doi.org/10.2105/AJPH.93.9.1583>
- Golledge, R. G. (1994). Path Selection and Route Preference in Human Navigation : A Progress Report. *International Conference on Spatial Information Theory*, (COSIT 1995), 207–222.
- Guo, Z., & Loo, B. P. Y. (2013). Pedestrian environment and route choice: evidence from New York City and Hong Kong. *Journal of Transport Geography*, 28, 124–136. <https://doi.org/10.1016/j.jtrangeo.2012.11.013>
- Hahm, Y., Yoon, H., Jung, D., & Kwon, H. (2017). Do built environments affect pedestrians' choices of walking routes in retail districts? A study with GPS experiments in Hongdae retail district in Seoul, South Korea. *Habitat International*, 70(July), 50–60. <https://doi.org/10.1016/j.habitatint.2017.10.002>
- Liu, Q., Ma, H., Chen, E., & Xiong, H. (2013). A survey of context-aware mobile recommendations. *International Journal of Information Technology & Decision Making*, 12(01), 139–172. <https://doi.org/10.1142/S0219622013500077>

- Matos, P., Vieira, J., Rocha, B., Branquinho, C., & Pinho, P. (2019). Modeling the provision of air-quality regulation ecosystem service provided by urban green spaces using lichens as ecological indicators. *Science of The Total Environment*, 665, 521–530. <https://doi.org/10.1016/J.SCITOTENV.2019.02.023>
- Novack, T., Wang, Z., & Zipf, A. (2018). A System for Generating Customized Pleasant Pedestrian Routes Based on OpenStreetMap Data. *Sensors*, 18(11), 3794. <https://doi.org/10.3390/s18113794>
- Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11), 10059–10072. <https://doi.org/10.1016/j.eswa.2012.02.038>
- Prato, C. G., & Bekhor, S. (2006). Applying Branch-and-Bound Technique to Route Choice Set Generation. *Transportation Research Record: Journal of the Transportation Research Board*, 1985(1), 19–28. <https://doi.org/10.1177/0361198106198500103>
- Ramos, F., Trilles, S., Muñoz, A., & Huerta, J. (2018). Promoting pollution-free routes in smart cities using air quality sensor networks. *Sensors (Switzerland)*, 18(8). <https://doi.org/10.3390/s18082507>
- Rodríguez, D. A., Merlin, L., Prato, C. G., Conway, T. L., Cohen, D., Elder, J. P., ... Veblen-Mortenson, S. (2015). Influence of the Built Environment on Pedestrian Route Choices of Adolescent Girls. In *Environment and Behavior* (Vol. 47). <https://doi.org/10.1177/0013916513520004>
- Seneviratne, P. N., & Morrall, J. F. (1985). Analysis of factors affecting the choice of route of pedestrians. *Transportation Planning and Technology*, 10(2), 147–159. <https://doi.org/10.1080/03081068508717309>
- Su, H., Zheng, K., Huang, J., Jeung, H., Chen, L., & Zhou, X. (2014). CrowdPlanner: A crowd-based route recommendation system. *Proceedings - International Conference on Data Engineering*, 1, 1144–1155. <https://doi.org/10.1109/ICDE.2014.6816730>
- Tsai, C.-Y., & Chung, S.-H. (2012). A personalized route recommendation service for theme parks using RFID information and tourist behavior. *Decision Support Systems*, 52(2), 514–527. <https://doi.org/10.1016/j.dss.2011.10.013>
- Weinstein Agrawal, A., Schlossberg, M., & Irvin, K. (2008). How Far, by Which Route and Why? A Spatial Analysis of Pedestrian Preference. *Journal of Urban Design*, 13(1), 81–98. <https://doi.org/10.1080/13574800701804074>
- World Health Organization. (2016). Urban Green Spaces and Health: a Review of Evidence. *WHO Regional Office for Europe*, 80.