

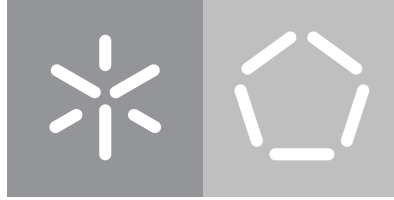


Universidade do Minho

Escola de Engenharia

João Miguel Pires Barreira

“HealthAdvisor” – Development of Central Services



Universidade do Minho

Escola de Engenharia

João Miguel Pires Barreira

“HealthAdvisor” – Development of Central Services

Master’s Dissertation

Integrated Master’s in Informatics Engineering

Work supervised by

Professor Doutor Cesar Analide

Professor Doutor João Ricardo Ramos

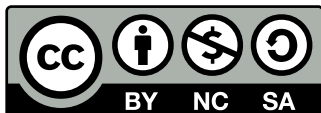
COPYRIGHT AND TERMS OF USE OF THIS WORK BY A THIRD PARTY

This is academic work that can be used by third parties as long as internationally accepted rules and good practices regarding copyright and related rights are respected.

Accordingly, this work may be used under the license provided below.

If the user needs permission to make use of the work under conditions not provided for in the indicated licensing, they should contact the author through the RepositóriUM of Universidade do Minho.

License granted to the users of this work



**Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International
CC BY-NC-SA 4.0**

<https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en>

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of Universidade do Minho.

Abstract

“HealthAdvisor” – Development of Central Services

Over the last years, there has been an increase in popularity of online platforms that enable users of the most varied economic activities (e.g. restaurants, bars, shops, shopping centers, nightlife, etc.) to evaluate the quality of the service received. Thus, based on the community’s aggregate of opinions, the user can then make a more informed decision when opting between a wide variety of competing services.

However, the field of medical services and health is one of the economic areas where this procedure isn’t widely adopted. Consequently, the goal of this work is to develop an intelligent technological solution centered on the user, that allows the registration, evaluation, and recommendation of healthcare institutions.

In this work, we present a platform called HealthAdvisor. This platform is comprised of a web User Interface, a Recommender System that generates recommendations for the user, and a backend service that supports both of these components by providing and exposing multiple REST APIs (Representational State Transfer Application Programming Interfaces). By using this solution, it is possible for a user to search for healthcare institutions, read the reviews of institutions made by other users, provide their own reviews, and receive recommendations based on their user profile as well as their past usage of the service.

Keywords: Recommender systems, user-generated content, healthcare, RESTful APIs

Resumo

“HealthAdvisor” – Desenvolvimento dos Sistemas Centrais

Nos últimos anos, têm vindo a ganhar popularidade várias plataformas *online* que possibilitam aos utilizadores de serviços das mais variadas áreas de atividade económica (e.g. restaurantes, bares, lojas, centros comerciais, diversão noturna) avaliar o serviço recebido. Desta forma, com base no agregado das opiniões disponibilizadas pela comunidade, o utilizador poderá tomar uma escolha mais informada no momento da decisão entre os vários serviços disponíveis.

No entanto, uma das áreas onde este processo não está a ser amplamente utilizado prende-se com a área dos serviços médicos e de saúde. Assim sendo, o objetivo deste trabalho é o desenvolvimento de uma solução inteligente centrada no utilizador, que permita o registo, avaliação e recomendação de instituições de saúde.

Neste trabalho, apresentamos uma plataforma chamada HealthAdvisor. Esta plataforma é composta por uma interface de utilizar *web*, um sistema de recomendação que gera recomendações para o utilizador, e um serviço de *backend* que suporta estes componentes através da disponibilização de REST APIs (Representational State Transfer Application Programming Interfaces). Através da utilização desta solução é possível ao utilizador procurar instituições de saúde, ler as avaliações a instituições feitas por outros utilizadores, submeter as suas próprias avaliações e obter recomendações baseadas no seu perfil de utilizar e histórico de utilização do sistema.

Palavras-chave: Sistemas de recomendação, Conteúdo gerado pelo utilizador, saúde, APIs RESTful

Contents

List of Figures	viii
List of Tables	ix
Listings	x
List of Algorithms	xi
1 Introduction	1
1.1 Context and Motivation	1
1.2 Objectives	2
1.3 Document structure	2
2 State of the Art	3
2.1 Recommender Systems Definition	3
2.2 Recommender System Paradigms	3
2.2.1 Collaborative Filtering	3
2.2.2 Content-based Filtering	5
2.2.3 Hybrid Recommendations	6
2.3 Recommender Systems Evaluation	7
2.4 Comparison Between Recommender Systems Paradigms	14
2.5 Real-world applications of Recommender Systems	15
2.5.1 Recommender Systems and User-Generated Content in Travel	15
2.5.2 Recommender Systems and User-Generated Content in Healthcare	15
3 Implementation	17
3.1 Modelling	17
3.1.1 Domain Model	17
3.1.2 Use Case Diagram	18
3.2 Architecture	19

3.3	HealthAdvisor Frontend	21
3.3.1	Mockups	22
3.3.2	Final Result	26
3.3.3	Tests	32
3.4	HealthAdvisor Backend	34
3.4.1	Data Storage	35
3.4.2	APIs	36
3.4.3	Tests	48
3.5	HealthAdvisor Recommendations	49
3.5.1	Recommendations API	50
3.5.2	Dataset Builder	50
3.5.3	Recommender System	51
3.6	Recommender System	51
3.6.1	Recommender System Architecture	52
3.6.2	Intermediate Content-Based Recommender Systems	55
3.6.3	Aggregator Recommender System	61
3.7	Usage of Learned Data	62
3.8	Flow of Data and Generation of New Recommendations	64
4	Conclusions and Future Work	66
4.1	Conclusions	66
4.2	Future Work	67
	Bibliography	69
	Appendices	72
A	Appendix 1 – Script that calculates the polarity of a text	72

List of Figures

1	Domain Model	18
2	Use Case Diagram	19
3	System architecture	21
4	Homepage mockup	23
5	Login page mockup	23
6	Register page mockup	24
7	Search results page mockup	25
8	Institution page mockup	26
9	Homepage – "Appointment" search	27
10	Homepage – "Institution" search	27
11	Login page	27
12	Register page	28
13	"Appointment" search results page (location)	29
14	"Institution" search results page (name and location)	29
15	Institution page	32
16	HealthAdvisor Backend Architecture	34
17	Logical schema	36
18	Institutions/Search API tests	49
19	HealthAdvisor Recommendations Architecture	50
20	Register page with a short questionnaire for medical specialties and subsystems	53
21	Recommender System Architecture	54
22	Recommendations presented to the user on the homepage	63
23	Recommendations presented to the user on the search results page	64
24	Cycle of data	65

List of Tables

1	Recommendation process using a user-based approach	4
2	Recommendation process using an item-based approach	5
3	Classification of the outcome of a system's item recommendation to a user	8
4	Comparison between Recommender System paradigms	14
5	<i>findByld</i> API request specification (Users API)	37
6	<i>findAll</i> API request specification (Users API)	37
7	<i>create</i> API request specification (Users API)	38
8	<i>update</i> API request specification (Users API)	38
9	<i>delete</i> API request specification (Users API)	39
10	<i>findByld</i> API request specification (Institutions API)	39
11	<i>getAllSpecialties</i> API request specification (Institutions API)	40
12	<i>getAllSubsystems</i> API request specification (Institutions API)	40
13	<i>getAllLocations</i> API request specification (Institutions API)	41
14	<i>findBySpecialtySubsystemLocation</i> API request specification (Institutions API)	42
15	<i>findByNameLocation</i> API request specification (Institutions API)	43
16	<i>addInstitutionReview</i> API request specification (Ratings API)	44
17	<i>addInstitutionPageVisit</i> API request specification (Analytics API)	45
18	<i>addTimeSpentOn</i> API request specification (Analytics API)	45
19	<i>getAllInstitutionsData</i> API request specification (Ratings API)	46
20	<i>getUserProfile</i> API request specification (Ratings API)	47
21	<i>getAllUsersProfileData</i> API request specification (Ratings API)	48

Listings

1	Example of a map tile with a marker using Leaflet and OpenStreetMap	30
2	Example of a snapshot file for a React component	33
3	Example of a snapshot test	34

List of Algorithms

1	Calculate similarity between Institutions: Specialties	56
2	Calculate similarity between Institutions: Agreements	56
3	Calculate similarity between Institutions: Locations	57
4	Calculate similarity between Institutions	57
5	Calculate similarity between User Profile: Sex	58
6	Calculate similarity between User Profile: Age	59
7	Calculate similarity between User Profiles: Institutions	60
8	Calculate similarity between User Profiles	61
9	Calculate similarity between Institutions (aggregator)	62

Introduction

1.1 Context and Motivation

Consumer protection is one of the concerns of today's societies. More specifically, one of the fundamental pillars of consumer protection is the consumer's right to information. This right is enshrined in law both in Portugal – through Article 60 of the Constitution of the Portuguese Republic ¹, as well as Lei n.º 24/96 ("*Lei da Defesa do Consumidor*"²) –, but also in the European Union itself ³.

However, nowadays, there are multiple technology platforms – such as *TripAdvisor* ⁴ or *Foursquare* ⁵ –, that have proved to be valuable tools for the consumer through the possibility of consulting and submitting reviews on services and institutions.

Through the application of the fundamental concepts of the so-called Web 2.0 (e.g. user-generated content, participatory culture, ease of access and use, etc.), this type of platform has made a positive contribution to increasing the amount of information available to consumers (despite the risk arising from fraudulent reviews).

However, there is still room for maneuver in order to extend the positive contribution of these types of platforms to other areas of economic activity – such as healthcare – with the help of emerging technologies such as Recommender Systems.

¹<https://www.parlamento.pt/Legislacao/Paginas/ConstituicaoRepublicaPortuguesa.aspx#art60>

²<https://dre.pt/pesquisa/-/search/406882/details/maximized>

³https://ec.europa.eu/info/law/law-topic/consumers/consumer-contract-law/consumer-rights-directive_en

⁴<https://www.tripadvisor.com/>

⁵<https://foursquare.com/city-guide>

1.2 Objectives

The main objective of this work is the development of a technological solution that allows the interaction of a community of users around the most diverse services in the health field. In this context, the main functionality delivered should be order to provide the user the possibility to review and evaluate the service received.

Through the application of techniques related to Recommender Systems, it is also intended to enrich the final product with more technologically advanced capabilities, such as, for example, the detection of behavior patterns in the use of the platform by the community.

Another objective is to create a technological solution that is designed and focused on the health area, but which cornerstone – based on the previously described concepts (such as *user-generated content* and *Web 2.0*) –, makes it possible to extend this solution to other areas of economic activity, in a facilitated manner.

1.3 Document structure

The present document is divided into four chapters. In Chapter 2 (p. 3) the study of the State of the Art is presented. This chapter provides a comprehensive view on the topic of Recommender Systems, as well as its real-world applications in the fields of healthcare and travel. Chapter 3 (p. 17) describes in detail the implementation of the HealthAdvisor platform. It begins by describing the models of the system, as well as its architecture. Then, it presents the different components that comprise the project – HealthAdvisor Frontend, Backend, and Recommendations –, taking a closer look at the recommender system in the following section. At the end of this chapter, the cycle from the collection of usage data to the generation of new recommendations is presented. Finally, Chapter 4 (p. 66) focuses on summarizing the work done, evaluating the objectives that were completed, as well as presenting ideas for future work on this project and topic.

State of the Art

2.1 Recommender Systems Definition

According to Ricci et al. [19], "Recommender Systems are software tools and techniques providing suggestions for items to be of use to a user. The suggestions relating to the various decision-making processes, such as what items to buy, what music to listen to, or what online news to read".

The main objective of a Recommender System (RS) is to predict the preference a user would give to a certain item, generating a recommendation.

In a general sense, a Recommender System (RS) can be used as a means to maximize sales and increase user engagement and satisfaction of its users. Such systems can benefit both the users – as well as the recommendation provider – by, for example, suggesting new items, identifying items users did not know about, and targeting users with relevant sales and promotions.

2.2 Recommender System Paradigms

There are multiple types of Recommender Systems. Some examples of these different paradigms are Collaborative Filtering (2.2.1), Content-based Filtering (2.2.2), and Hybrid Recommendations (2.2.3).

2.2.1 Collaborative Filtering

Collaborative filtering is one of the most common and successful recommender system techniques [19] [2] [13] [23].

By using this technique, recommender systems generate new recommendations for a given user by taking into account the items that other users with similar tastes have liked in the past.

The user-item interaction matrix is the name given to the data structure that is used to learn the users' preferences, predict new ratings, recommend items to users and evaluate the system's performance [13]. There are two distinct subsets of systems within Collaborative Filtering recommender systems that use this user-item interaction matrix differently. These two subtypes are known as Memory-based and Model-based systems.

2.2.1.1 Memory-based Collaborative Filtering

Memory-based algorithms use the whole user-item matrix to generate new predictions. First, it creates a neighbourhood for the active user that is composed of other users or items using a similarity function. There are multiple examples of such similarity functions, namely cosine, adjusted cosine, Pearson correlation coefficient, constrained Pearson correlation, Euclidean difference and mean-squared difference similarities [13].

The most used algorithm to compute neighbourhoods is k Nearest Neighbours (kNN). Moreover, two subtypes of this algorithm are typically used in Collaborative Filtering recommender systems: User-based Nearest Neighbour and Item-based Nearest Neighbour.

In the User-based Nearest Neighbour approach, a neighbourhood of size k is computed for the active user using the selected similarity measure. In order to receive a prediction using this neighbourhood, an aggregation operation (e.g. average, weighted sum or adjusted weight aggregation) needs to be applied. Finally, the top- N recommendations can be obtained by selecting the N best-ranked items [23]. Table 1 illustrates this recommendation process.

	Item 1	Item 2	...	Item Y
User A	10	9	...	10
User B	8	10	...	9
...	...	...	...	...
User C	9	10	...	r_{3C}

Table 1: Recommendation process using a user-based approach

In the Item-based Nearest Neighbour approach, the recommendations are produced by analyzing the similarity between items rather than users. First, the set of items in the system are compared to the items that the active user has rated in the past using the similarity measures already mentioned. It is then possible for the system to make a recommendation by discovering items unknown to the user that were liked or rated highly by users whose item ratings partially match some or all of the active user's past ratings. Table 2 illustrates this recommendation process.

	Item 1	Item 2	...	Item Y
User A	10	9	...	10
User B	8	10	...	9
...	...	...	...	...
User C	9	10	...	r_{3C}

Table 2: Recommendation process using an item-based approach

2.2.1.2 Model-based Collaborative Filtering

Model-based Collaborative Filtering systems make recommendations by using different data mining and machine learning techniques. Examples of such techniques include clustering models, association rules, probabilistic approaches, bayesian classifiers, singular value decomposition, probabilistic latent semantic analysis, linear regression, etc. [13]. The application of these techniques allows the system to recognize complex patterns based on the training data and then make intelligent predictions on test or real-world data. [22]

In comparison to the memory-based collaborative filtering paradigm, this type of system has better scalability and speed [2].

2.2.2 Content-based Filtering

Content-based recommender systems make recommendations by taking into account the items that the target user has liked in the past. In opposition to Collaborative Filtering, this type of system doesn't consider the ratings of other users. The similarity between items that the user has interacted with in the past and unknown items that can be recommended is based on the items' descriptions. In general, Content-based recommender systems rely on two sources of data. First, they rely on the content-centric description of the multiple items available in the system. Second, they rely on the user profile which is generated from user feedback about items they have interacted with in the past. This feedback can be implicit (i.e. user's ratings) or explicit (i.e. user's actions).

There are several approaches within this paradigm. In keyword-based approaches, the user indicates keywords reflecting their personal interests which will then be used as their user profile; then, recommendations are generated by analyzing the similarity of the items regarding the keywords of interest of the user. Semantic-based approaches focus on semantic description of the items that provide additional information about the items in an attempt to improve the system's knowledge regarding the user profile and the items available. [2]

These systems deal well with the cold start problem in regards to new items being added to the system as they rely solely on its content description. The cold start problem regarding new users is still present [1]. Additionally, content-based systems offer very low serendipity as this paradigm relies on the content of items rated by the user in the past [1].

2.2.3 Hybrid Recommendations

Hybrid recommender systems are systems that combine multiple recommender system paradigms such as Collaborative Filtering and Content-based Recommendations. The main motivation behind the construction of hybrid recommender systems is to leverage the complementary strengths of the various paradigms and better tackle the problems that each faces on their own.

The Netflix recommender system is a good example of a hybrid recommender [10]. This system makes recommendations based on past watched and liked items of similar users using collaborative filtering techniques, as well as based on the similarity between the content of items that the user has watched or liked and the content of undiscovered items, employing content-based techniques.

In [1], the author divided hybrid recommender systems into three subtypes: monolithic, ensemble, and mixed. In monolithic systems either existing recommenders are redesigned to implement some processes of other approaches or new recommenders that combine the features of multiple approaches are constructed. Ensemble systems are designed by using multiple recommenders in sequential or parallel arrangement. Finally, mixed systems are systems whose recommendations come from multiple approaches and are presented to the user simultaneously.

In turn, Burke [4] classified hybrid recommendation systems into seven different subtypes:

- **Weighted** – Combines the score of different recommendation components.
- **Switching** – Selects between different recommendation components and applies the one selected.
- **Mixed** – Presents recommendations from multiple recommenders simultaneously.
- **Feature Combination** – Augments the data set using collaborative information which is treated as an additional feature for each item. Then it applies content-based techniques over this augmented data set.
- **Cascade** – Utilizes a pipeline of different recommenders after one another. Recommenders with less priority are placed at a later stage on the pipeline, avoiding having to apply these lower priority algorithms to all items. Instead, these algorithms are used for more specific and restricted tasks like breaking classification ties that resulted from the components with a higher priority. Each of the processes of the components performs independently from one another and. In the final stage, their results are combined in a prioritized manner.
- **Feature Augmentation** – Computing a feature or set of features using one technique, which is then used as part of the input to the next technique. This technique differs from cascade in that components in later stages of the pipeline use the output of previous components as input for its classification instead of producing two different classifications independently from one another and then combining them.

- **Meta-level** – Applies a recommendation technique that produces a model that is used as the input of the next technique. This differs from feature augmentation in that the entire model is used as the input of the next component instead of using the features generated from this previous model as input.

2.3 Recommender Systems Evaluation

Evaluation is an important aspect to take into account so as to gauge how well a RS is performing. There are various metrics that can be used to accomplish this evaluation. Multiple works have focused on this topic.

One example is the work done in [19] where a review of the evaluation process of recommender systems is conducted. A range of properties that are commonly considered when deciding which recommendations are collected.

- **User Preference** – When selecting between multiple options of recommender systems, a perhaps natural approach would be to perform a user study where several participants are asked to choose between competing systems. While this evaluation is not centered on specific properties – but user satisfaction in general – breaking down satisfaction into smaller components is viewed to be helpful in order to understand the system and possible improvements. Nonetheless, additional good practices are outlined, such as weighing the votes by the importance of each individual user when applicable, as well as utilizing non-binary answers for the preference questions in order to better understand the distribution of the users' opinions (e.g. registering how much a user liked the system instead of just recording that they have liked it).
- **Prediction Accuracy** – This property measures how accurate the recommendations of a system are. The authors delineate three different classes of prediction accuracy: measuring the accuracy of rating predictions, measuring the accuracy of usage predictions, and measuring the accuracy of rankings of items.

Accuracy of ratings predictions. In application contexts where the recommender system predicts the ratings a user would give to an item, it is important to measure the accuracy of such predictions. For this purpose, the most popular metric used is Root Mean Squared Error (RMSE). Using this metric, for a set $\mathcal{T}$ of user-item pairs (u, i) , the system generates rating predictions $\hat{r}_{ui}$, while the true item ratings given by the user are hidden. Thus, RMSE between the predicted and true ratings is given by:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i) \in \mathcal{T}} (\hat{r}_{ui} - r_{ui})^2}$$

The authors also refer Mean Absolute Error (MAE) as an alternative measure that doesn't disproportionately penalize large errors. This metric can be obtained by:

$$\text{MAE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i) \in \mathcal{T}} |\hat{r}_{ui} - r_{ui}|}$$

Both these metrics can be normalized to the range of item ratings (i.e. $r_{\max} - r_{\min}$). These versions are called Normalized RMSE (NRMSE) and Normalized MAE (NMAE). Additionally, Average RMSE and Average MAE can be used to adjust for unbalanced test sets where RMSE and MAE would have been heavily influenced by errors on few but frequent items. Finally, in some contexts, it may also be desirable to use a distortion measure $d(\hat{r}, r)$ to adjust for situations where the absolute error of a predicted rating $\hat{r}$ is not only directly correlated to its magnitude of difference in relation to the actual user rating r . For example, in an application where recommending an item a user dislikes is worse than not recommending an item a user likes, a distortion measure $d(1, 0) = 4$, $d(0, 1) = 2$ (where 0 denotes a 'dislike' item rating and 1 denotes a 'like' item rating) can be used to penalize the first scenario twice as much as the second.

Accuracy of usage predictions. Alternatively, in many contexts of application, the system tries to recommend the user items that they may use. In these scenarios, the goal is not to measure whether the system correctly guesses the user's ratings of the items, but rather if the system correctly recommends items that the user ends up using (i.e. Usage Prediction). In order to measure this property, offline experiments can be conducted where a user's item selection is hidden and the system tries to recommend items that the user will use. The following table encompasses all possible results for each item recommendation:

	Recommended	Not recommended
Used	True Positive (tp)	False Negative (fn)
Not used	False Positive (fp)	True Negative (tn)

Table 3: Classification of the outcome of a system's item recommendation to a user

From this table, multiple values can be calculated, namely:

$$\text{Precision} = \frac{\#tp}{\#tp + \#fp}$$

$$\text{Recall} = \text{True Positive Rate} = \frac{\#tp}{\#tp + \#fn}$$

$$\text{False Positive Rate} = 1 - \text{Specificity} = \frac{\#fp}{\#fp + \#fn}$$

In an application where the number of recommendations that be presented to the user is fixed and predetermined, the authors suggest using Precision at N as a measure. However, in applications where this number is not fixed nor predetermined, it is preferable to evaluate algorithms over a range of recommendation list lengths. For this purpose, the computation of curves that measure the proportion of preferred items that are recommended is suggested. For example, one can compute curves that compare precision to recall (precision-recall curves) or true positive rate to false positive rate (Receiver Operating Characteristic or ROC curves). The former emphasizes the proportion of recommended items that are preferred while the latter emphasizes the proportion of items that are not preferred but are recommended. Therefore, selecting between one or the other should be done by taking this difference into account and choosing the one that better suits the application domain as well as its goal. While measures that summarize the precision-recall of ROC curves such as F-measure and Area Under the ROC Curve (AUC) are useful when comparing different algorithms, the authors suggest performing this selection based on a measure that reflects the application's needs and objective instead.

When evaluating accuracy for multiple users, a Customer ROC (CROC) curve can be used when there is a fixed number of recommendations for each user and a Global ROC (GROC) curve when this number varies. In contexts where the user is able to obtain a growing number of recommendations, the authors refer that the most common method of measure is computing a precision-recall (ROC) curve for each user and then averaging the resulting curves over all the users.

Accuracy of rankings of items.

- **Coverage** – This property most commonly refers to the Item Space Coverage which corresponds to the proportion of items the recommendation system can recommend (i.e. catalog coverage). This property can be measured directly by calculating the percentage of all items of an input data set that can ever be recommended using a specific algorithm. This measurement can also be done in either an offline, online, or user study experiment. Additionally, it is possible to weigh the items by another property (e.g. utility). By doing so, it may desirable to adjust the calculation of coverage to tolerate a lack of coverage over lower-valued items while enforcing coverage over high-valued ones. Another method to measure coverage is sales diversity which measures how "unequally different" items are chosen by users when using a specific system. This can be calculated using the Gini Index, where $i_1, \dots, i_n$ is the list of items ordered by the increasing value of $p(i)$ which represents the proportion item i accounts for among all the items available:

$$G = \frac{1}{n-1} \sum_{j=1}^n (2j - n - 1)p(i_j)$$

Using this formula, G is 1 when one item is always recommended and 0 when all of the items are recommended equally often. This can also be calculated with Shannon Entropy, using the same definition of $p(i)$:

$$H = - \sum_{i=1}^n p(i) \log p(i)$$

Similarly, H is 0 when one item is always recommended $\log n$ when n items are recommended equally often. Another definition for coverage is the User Space Coverage which corresponds to the "proportion of users for which the system can recommend items". Indeed, in some contexts, it may not be possible for a system to give recommendations for a certain user due to them having an empty or incomplete user profile (e.g. new users). Typically, this occurs because, as the system does not possess enough information about the user, prediction accuracy is negatively affected. Therefore, providers may desire to evaluate the trade-off between coverage and prediction accuracy when choosing between different recommender systems. A measure for this property can be determined by how complete the user profile must be in order for the system to be able to make a recommendation. This measurement can be computed in offline experiments.

- **Confidence** – This property can be defined as "the system's trust in its predictions". To evaluate a recommender system's confidence in a given recommendation, the probability that the predicted value is true can be used. In alternative, it is also possible to gauge confidence by determining the interval in which a predefined portion of the users' actual ratings match the system's predictions lie (e.g. 95% of the users' ratings lie within a -0.5 and +0.5 margin of a certain system's prediction). However, the most general method to assess confidence is to determine the complete distribution over possible outcomes (i.e. actual user ratings) for a given prediction. These values can be gathered in offline evaluations, similarly to the process of measuring prediction accuracy.
- **Trust** – While confidence represents the system's trust in its predictions, trust describes the user's trust in the recommendations that are given to them by the system. One way to increase a user's trust in a system is to limit the recommendations to items that are more familiar to the user or to set a higher confidence threshold in the system's predictions for that user. Another way to achieve this is for the system to explain to the user the recommendations that it provides (much in the same way as Explainable Recommender Systems function). While it is pointed out that this is a property whose evaluation is not suitable for offline environments, multiple methods for measuring trust are enumerated, such as asking users directly in user studies, measuring the number of recommendations followed by the users, as well as determining the number of users who return to the system. This last method, however, relies on the assumption that users who use the system repeatedly do so because they trust its recommendations. Indeed, it is also mentioned that these

measurements in general may exclude other factors that contribute to user satisfaction that are not related to the users' notion of credibility in the system's predictions.

- **Novelty** – This property represents the system's ability to recommend items that the user does not know about prior to the recommendation. When the history of items the users have used in past is available, a simple way to introduce novelty on the system's recommendations is to filter out these items. Nonetheless, in many cases, this information is not available. Furthermore, the authors also point out methods with which to evaluate novelty. One way to measure this property is to conduct user studies where users are asked whether they had known about the item being recommended to them. Additionally, these measurements can also be done in an offline fashion by performing experiments in which the dataset of user ratings is split into two subsets corresponding to the ratings that occurred before and after a selected point in time. Using this approach, it is possible to filter out some user ratings that occurred before and filter out all ratings that occurred after that time. By doing so, the system can then be rewarded during the offline exercise for each item that was recommended and rated by the user after the split time, while being punished for each item recommended that was rated by the user prior to the split time. Moreover, the authors also underline the importance of taking accuracy into consideration while measuring novelty, as "irrelevant recommendations may be new to the user, but still worthless". Finally, it is suggested that popularity can also be a factor to take into account in these measurements, under the assumption that popular items are less likely to be novel to the user.
- **Serendipity** – The authors define serendipity as being the measure of "how surprising the successful recommendations are". For example, if a user is recommended a new movie starring an actor that he likes (i.e. who appears several times in their movie history), the recommendation may be accurate and novel, but it is not surprising. Therefore, this recommendation is considered to have low serendipity. On the other hand, if the user is presented with a movie whose cast is completely unknown to them – and they enjoy the movie –, that recommendation is considered to be serendipitous. Several approaches to measure serendipity are outlined. One of each, put forward by [26] suggests to manually label pairs of items as being redundant. By doing so, one can measure serendipity by penalizing systems if they recommend redundant items. Furthermore, another method to measure serendipity that doesn't rely on manually labeling any items is suggested. Instead, it is proposed that a measurement between items based on content be designed. By this means, each successful recommendation can be ranked based on the distance from the recommended item to the set of items the user had already known. The furthest the recommendation is from the known set, the more serendipitous it is. Finally, it is referred that another method to measure this property can be conducting a user study that asks the users whether the received recommendations were unexpected. However, the importance of evaluating this property over time

is underlined, as the users may initially be curious regarding an unexpected recommendation but regret it once they delve into similar items. Indeed, this might prove to be detrimental to the system as the user may stop following future suggestions or stop using the recommender system altogether.

- **Diversity** – Diversity can be defined as being the opposite of similarity. In general, we can say that the more similar a recommended item is when compared to items already suggested, the less diverse is the recommendation. The most common method to determine diversity is to use item-to-item similarity measures typically based on the item's content. By doing so, one can then determine the similarity between pairs of items which, in turn, enables measuring the diversity gained by a new recommendation. Finally, it is pointed out that an increase in diversity may come at a cost in accuracy, a trade-off between these two properties may need to be analyzed.
- **Utility** – Utility is a property that is related to the value the recommendation system offers its provider. For example, it can be related to the revenue the system generates. In general, this property can be measured by the analysis of the utility functions that the provider wants to optimize. These utility functions may be of varied nature, varying from one provider to another, and may also be tied with other properties. For example, for a provider that wants their recommender system to provide diverse recommendations to its users, Diversity can be used as a utility function, whereas another provider more interested in novel recommendations may use Novelty instead. Defining and measuring utility from the user's point of view is typically more challenging as, many times, user utility is difficult to capture and model. At the same time, selecting a specific measure to gauge the utility the users receive from the system's recommendations may be misleading. For example, one may choose to use the minimization of user's cost as a utility function; however, considering that the same amount of money does not have the same utility for people with different income levels, for instance, this analysis may be flawed. Nonetheless, it is possible, for example, to use user ratings as a utility function by assuming that a higher rating denotes higher user utility. Negative recommendations may also be used to assign negative utilities. Another way to measure this type of utility is to conduct user studies where users are asked to assign a utility to a certain outcome that follows a recommendation from the system, or to say whether they prefer one outcome or another.
- **Risk** – In some contexts, it may be useful to measure the level of risk of a system's recommendations. For example, in a hypothetical recommender system that deals with financial assets, some of its users may be more risk-seeking while others may prefer low-risk items. Therefore, the system may choose not only to measure the value generated from its recommendations, but also to minimize their underlying risk. The most common way to measure risk is to introduce the utility variance when measuring the utility of a system.

- **Robustness** – The authors define robustness as the "stability of the recommendation in the presence of fake information". Indeed, in a widely-used recommender system, it may be desirable for an interested party to change the rating of a given item. For example, the owner of a restaurant may be interested in altering the rating of their restaurant in order to increase its profitability. Nonetheless, in general, it is considered that designing a system that is immune to any type of attack is not realistic. Therefore, a typically better approach is to measure the cost of an attack which is normally associated with the amount of injected information into the system. Following this approach, one can, for example, run an experiment where attacks that inject information into the system are simulated and their cost calculated. Additionally, in online experiments, it may be useful to, alternatively, analyze the system's data, identifying subsets of information that were a result of an attack, and calculating their cost. Finally, the authors explain that robustness may also refer to "the stability of the system under extreme conditions, such as a large number of requests". Therefore, this definition can also relate to the system's infrastructure and its scalability.
- **Privacy** – In recommender systems, users provide personal information regarding their item preference with the intention of getting useful item recommendations. Therefore, it is especially important in this context to protect the user's privacy regarding the information they provide. Not doing so may undermine the user's trust in the system. As a consequence, the user may feel reticent when using the system and be less likely to provide valuable information which, in turn, will lead to a worse user experience. Ultimately, disregard for a system's privacy may lead the user to leave the system altogether. Moreover, the authors delineate multiple methods to measure privacy, namely studying worst-case scenarios, comparing algorithms by evaluating the portion of users whose private information was compromised, or by defining different levels of privacy such as k-identity and comparing the propensity for data breaches for each. The need for a careful evaluation of the trade-off between privacy and accuracy is also underlined.
- **Adaptivity** – This property is defined by the speed at which the system's recommendations adapt to the insertion of additional information. Indeed, certain application contexts may inherently require this characteristic, such as a system that recommends news stories that may wish to recommend events that began trending a short time before. Adaptivity can be measured offline by calculating the amount of information needed before an item is recommended. However, increasing this property (i.e. decreasing the amount of information needed for new recommendations) may come at a cost of accuracy. Therefore, this trade-off may also be taken into consideration when comparing different systems of recommendation algorithms. Another form of adaptivity referred by the authors is the "rate by which the system adapts to a user's personal preferences or to changes in user profile". For example, if a user rates an item they have recently consumed, they may expect the system to update the recommendations it gives in a timely fashion. A method to measure this type of adaptivity

can be analyzing the difference between the recommendations after new information is added to the system. For this approach, the authors suggest using the Gini index and Shannon entropy to measure this variability in the recommendation item set.

- **Scalability** – This property defines how well a recommender system can deal with the growth of the dataset it is applied on. That is to say, given a dataset with N records, how the performance of the recommender system is affected when the number of records tends to a large enough value. Indeed, with this increase in the size of the collection of values, it is expected that a slow down is observed or a larger amount of computational resources consumed also increases. Therefore, one of the metrics used to assess scalability is the measurement of the consumption of additional system resources such as computational power or memory as the size of the dataset enlarges. It is also deemed important to Ricci2015 to measure how fast the system provides recommendations (i.e. variation of throughput) as well as the required time for making a recommendation online (i.e. variation of latency or response time) with growing datasets.

2.4 Comparison Between Recommender Systems Paradigms

Based on the previous study done in Section 2.2 and the work of [2], Table 4 summarizes the most relevant advantages and limitations of each of the Recommender System paradigms, drawing a comparison between each other.

Paradigm	Advantages	Limitations
Collaborative Filtering	<ul style="list-style-type: none"> • No knowledge needed about the items' content • Serendipity • Independent of any machine-readable representation of the items • Based on the users' evaluation of items 	<ul style="list-style-type: none"> • Coverage • Scalability • Cold-start • Sparsity • New user problem • New item problem
Content-based Filtering	<ul style="list-style-type: none"> • Minimize sparsity problem • Minimize cold-start problem • New items are handled more easily 	<ul style="list-style-type: none"> • New user problem • No serendipitous recommendations • Limited content analysis • Inability to evaluate the quality of an item
Hybrid Recommendations	<ul style="list-style-type: none"> • Overcome limitations of a specific paradigm • Improve prediction performance • Flexibility 	<ul style="list-style-type: none"> • Increased complexity • More expensive to implement

Table 4: Comparison between Recommender System paradigms

2.5 Real-world applications of Recommender Systems

2.5.1 Recommender Systems and User-Generated Content in Travel

In today's digital world, recommender systems and user-generated reviews play a great role in the economical choices of individuals. Indeed, they have an influential capacity and have changed the way an individual decides which products to acquire.

One of the reasons for this has been presented by Senecal and Nantel [21] in suggesting that, in order to evaluate a product or service, many consumers prefer to rely on the opinions of other users over information presented by the creators of those products. This notion that consumers tend to view the opinions of peers as more trustworthy is also shared by the work of Litvin et al. [15]. As a result, a growing number of travelers are choosing to utilize platforms on which other consumers share their experiences, at the expense of information put forward by service providers [11].

The influence of user-generated content in travel has been studied by Ye et al. [25], that highlight the importance of user-generated reviews in the field, by observing its concrete impact in the context of China. Their empirical findings show that a ten percent increase in review ratings was responsible for a rise in online bookings of more than five percent. They also make the connection to previous research in other fields, citing similar findings in retail [5] and in the movie industry [7].

In their own work, Ayeh et al. [3] stress the importance of the credibility of user-generated reviews, stating that the provision of personal information in the reviews, as well as the notion of homophily, contribute positively to an increase in perceived credibility. However, they also note that there isn't a correlation between this perceived credibility and the truthness reflected by the review itself. Therefore, this aspect is cited as having far-reaching implications for sales, profitability, and travel's brand reputation.

However, this general fear of lack of credibility is put into question by the work of O'Connor [16]. Using a sample of London hotels, the author notes that these fears are unfounded and finds no system compromise as a result of false reviews. Nonetheless, the study also shows that while *TripAdvisor* provides a reply mechanism, this feature is rarely used by property owners.

2.5.2 Recommender Systems and User-Generated Content in Healthcare

In the field of healthcare, the work done by Hardey [12] also highlights the importance of user-generated reviews as they provide a way of previewing a doctor or clinic before any interaction. Moreover, the use of these reviews is conjectured to be responsible for a change in the nature of the medical profession and the doctor-patient relationship. The pivotal role of credibility and trust is also referenced in this study, where the work of Preece [17] and Wellman is [24] are cited. Furthermore, the challenge posed by the act of 'self-reviewing' is mentioned by the citation of the work done by Shay and Pinch [6] regarding e-commerce websites.

Emmert et al. [9] also highlight the importance of physician-rating websites (PRW) by stating that they may lead to quality improvements as demonstrated by previous work [8]. However, by analyzing *jameda*, a German PRW, their main contributions are that, in that particular scenario, peer-to-peer communication between patient and physician has not been reached, as only a minority of medical practitioners respond to ratings and, altogether, are aware of the PRW existence.

However, in a different setting, work points towards a different direction. Indeed, the study of Lagu et al. [14] analysed the impact of *NHS Choices*, a government-led platform that enables users to review United Kingdom's National Health Service institutions and medical practitioners. First, they posit that the use of the platform is increasing. Second, they note that hospitals read and are aware of the importance of the reviews, even describing practical changes made in response to those reviews. Third, they stress the importance of mixing narrative reviews – such as the ones analyzed – with systematically collected patient experience data, proving the usefulness of both.

The usefulness of narrative reviews is also noted by the works of Benjamin et al. [18] and Rozenblum and Bates [20]. In summary, these studies corroborate the notion that user-generated reviews complement – but not supplant – traditional surveys. Indeed, it is stated that they help differentiate "poor care from outstanding care"[20] and contribute positively to the overall assessment of hospital quality [18].

Implementation

3.1 Modelling

In order to better understand the domain and the functionality the final application would have, a Domain Model was constructed (3.1.1) and the Use Cases were enumerated, described and put together to construct a Use Case Diagram (3.1.2).

3.1.1 Domain Model

The Domain Model (Figure 1) was constructed so as to better describe the system's entities as well as the relationships established between one another.

An *Institution* is perhaps the most central entity of the whole system. It has a name, a fiscal identification number (*Número de Identificação Fiscal* – NIF), a location, a phone number, an e-mail, and an average rating. For each of its *Specialties*, it may have *Agreements* set in place between said *Specialty* and health *Subsystems* (or insurance companies). These *Agreements* have an expiration date.

One of the other central entities is the *User* which, in turn, has an e-mail, name, age, and sex. It makes a *Review* about an *Institution*. That *Review* has a value (integer from 1 to 5), a comment containing the text of the review, and a timestamp indicating when that review was made.

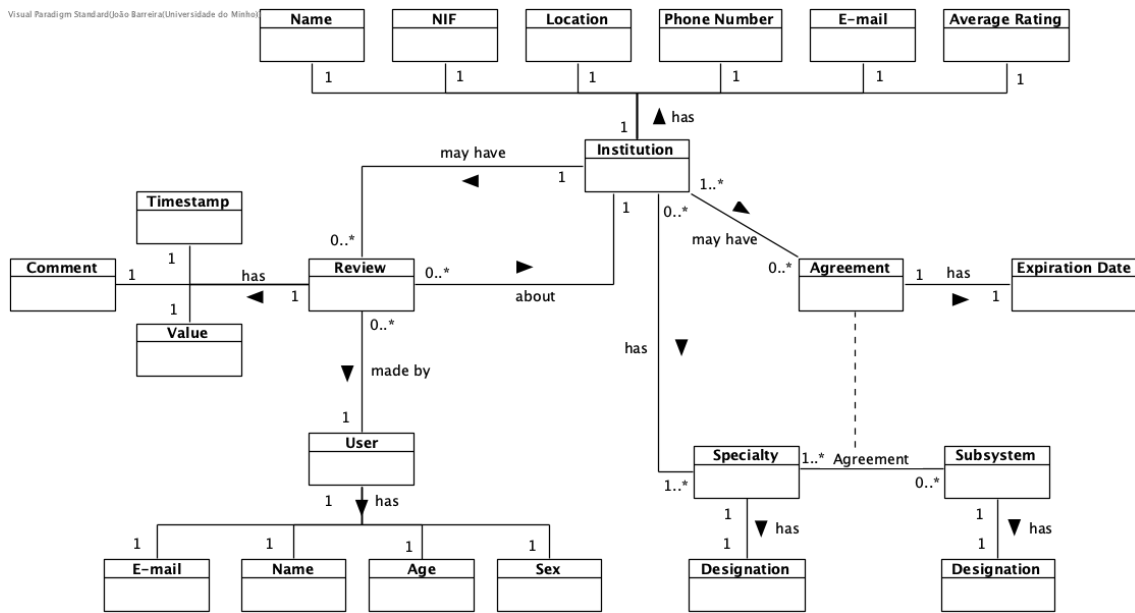


Figure 1: Domain Model

3.1.2 Use Case Diagram

The Use Case Diagram (Figure 2) is intended to be a representation of the various users' interactions with the system. In this case, there is only a single class of users. The list of use cases that must be supported is the following:

- **Use Case #1: Register** – The user must have the ability to register in the system (i.e. create an account).
- **Use Case #2: Login** – The user must be able to log in to the system using their account.
- **Use Case #3: Institution Search (specialty)** – The user must be able to search institutions by medical specialty.
- **Use Case #4: Institution Search (subsystem)** – The user must be able to search institutions by subsystem.
- **Use Case #4: Institution Search (name)** – The user must be able to search institutions by institution name.
- **Use Case #4: Institution Search (location)** – The user must be able to search institutions by location.
- **Use Case #5: Institution's details** – The user must be able to view the page of an institution that contains its details.

- **Use Case #6: Submit review** – The user must be able to submit a review for an institution.
- **Use Case #7: Other reviews** – The user must be able to view the reviews made by other users for all institutions in the system.
- **Use Case #8: Receive recommendations** – The user must receive recommendations featuring institutions the system thinks they will like.

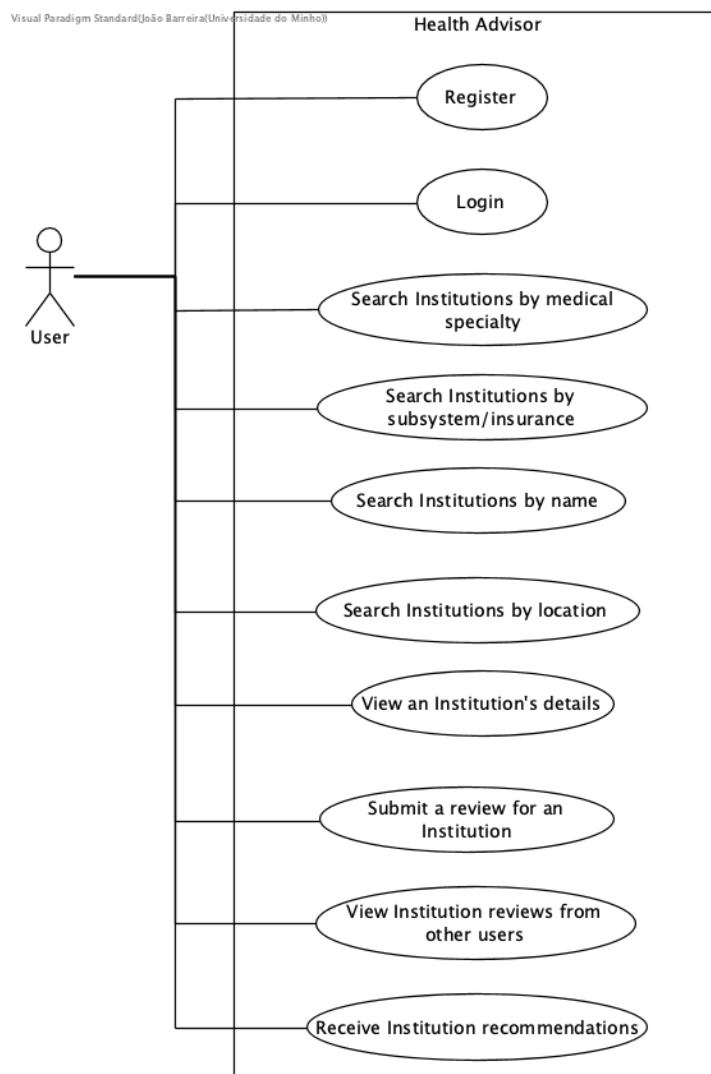


Figure 2: Use Case Diagram

3.2 Architecture

The general architecture of the implemented solution for HealthAdvisor is presented in Figure 3. This implementation comprises three different subsystems:

- **HealthAdvisor Frontend** – This subsystem is responsible for HealthAdvisor’s user interface. It includes multiples pages for the different functionalities available to the user: Login, Register, Homepage, Institution Search, and Institution details. This component is further detailed in Section [3.3](#).
- **HealthAdvisor Backend** – This subsystem is responsible for the provisioning of multiple APIs that serve both the Frontend as well as Recommendations. More specifically, it exposes APIs for User Authentication, Search, User details, and Institutions details. Additionally, it implements an Analytics API that deals with user analytic data. This component is further detailed in Section [3.4](#).
- **HealthAdvisor Recommendations** – This subsystem is responsible for the generation of new recommendations for the users. It comprises multiple scripts that run periodically to build the needed datasets ([3.5.2](#)) and generate the recommendations ([3.5.3](#)), along with an API that returns the list of pre-computed recommendations for a given user ([3.5.1](#)). This component is further detailed in Section [3.5](#).

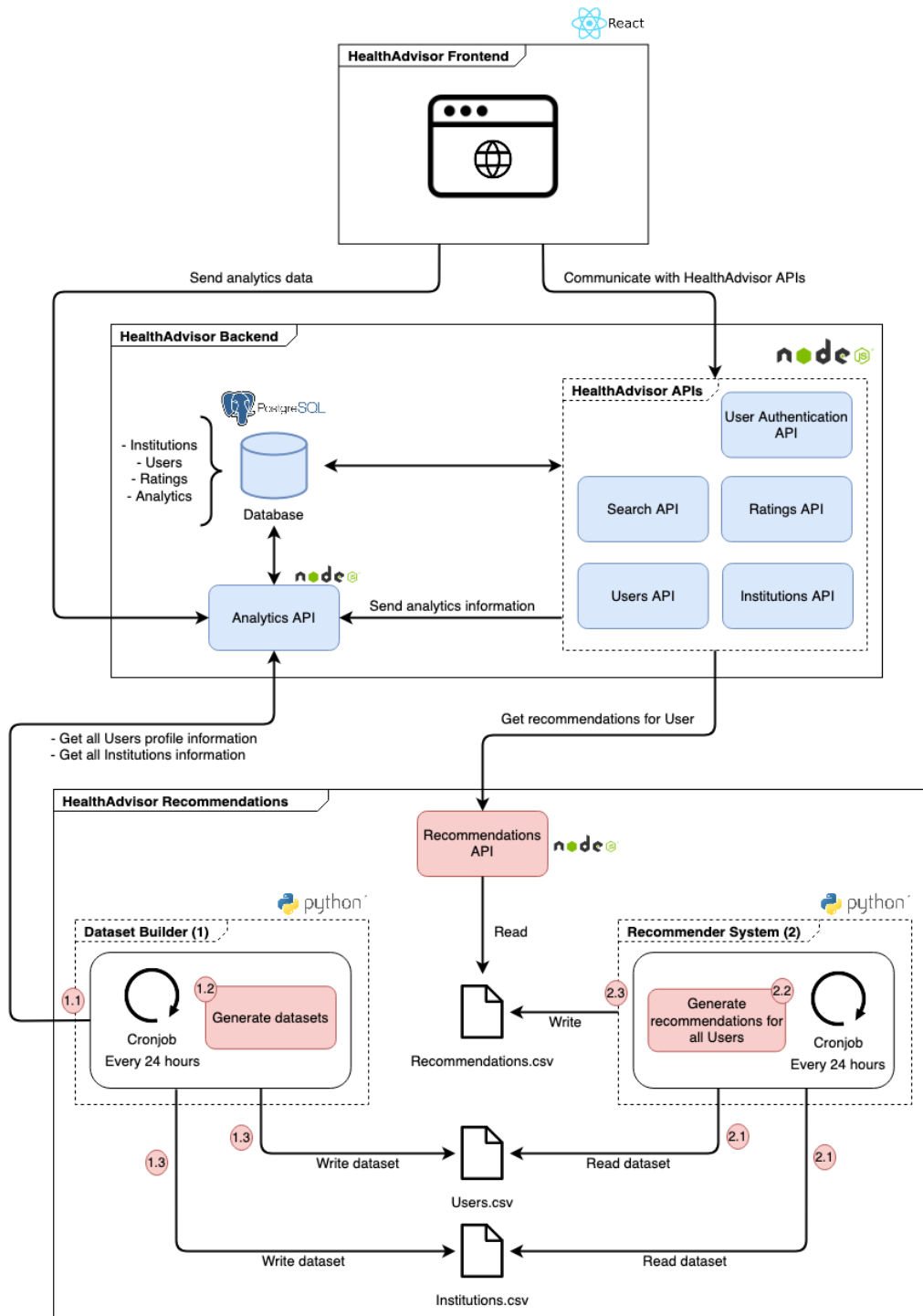


Figure 3: System architecture

3.3 HealthAdvisor Frontend

The HealthAdvisor Frontend is the component responsible for the implementation of the User Interface that the user interacts with. This corresponds to a React application that implements the multiple pages

that comprise the HealthAdvisor application.

React was chosen because it is a free and open-source Javascript library for building component-based user interfaces, that has great community support and is highly popular. By virtue of its virtual Document Object Model (DOM)¹ it also enables high performance. Using the virtual DOM, React creates an in-memory data structure that represents the object tree, updating the real DOM efficiently in a process called reconciliation. For each visual change on the page, this process occurs and updates only the elements that were actually modified, instead of the whole DOM. It also features functionality like JSX² and JSS³ that work as an easy-to-use replacement for HyperText Markup Language (HTML) and Cascading Style Sheets (CSS), respectively, on the React ecosystem.

3.3.1 Mockups

The first step in the development of the frontend, was the creation of mockups. This step allowed us to plan what the user interface of the platform would look like, making changes and iterating on the designs before the final product was developed.

3.3.1.1 Home Page

The first one, presented in Figure 4 represents the homepage.

At the top, there is a navigation bar with a reference to the name of the platform. Below that there is a form area where the user can search for an institution either by its name (and location) or by entering a medical specialty, a health subsystem of insurance company name, and/or a location. At the bottom of the page, there is also a simple footer area.

¹<https://reactjs.org/docs/faq-internals.html>

²<https://reactjs.org/docs/introducing-jsx.html>

³<https://cssinjs.org/>

Figure 4: Homepage mockup

Clicking on the Login button on the top right, takes the user to the login page (Figure 5).

Clicking on the search button on the right-hand side of the form takes the user to the search results page (Figure 7).

3.3.1.2 Login Page

The mockup for the login page (Figure 5) features a simple form with two fields for the user's email address and the password, with a submit button. A link to the registration can be seen below so that the user who doesn't have an account can sign up. Clicking on this link takes the user to the register page (Figure 6).

Figure 5: Login page mockup

3.3.1.3 Register Page

The register page is similar to the login page. However, it features additional fields so that the user can input their name, age, and sex. Furthermore, there is also a link below the form but it takes the user back to the login page (Figure 5).



The mockup shows a web page for 'HealthAdvisor'. At the top right is a close button (X). The title 'HealthAdvisor' is in red. Below it is the heading 'Registrar'. The form consists of several input fields: 'Nome próprio' and 'Apelido' (side-by-side), 'Endereço de e-mail', 'Palavra-passe', and 'Repetir palavra-passe' (stacked vertically), and 'Idade' and 'Sexo' (side-by-side, with a dropdown arrow for 'Sexo'). Below the form is a red button labeled 'Iniciar Sessão'. At the bottom, there is a link: 'Já tem conta? [Faça login](#)'.

Figure 6: Register page mockup

3.3.1.4 Search Results Page

The search results page (7) is the page that appears after the user searches for an institution on the homepage (Figure 4). On this page, the navigation bar is still present and, below, the results are displayed. To the right, there is an area that is reserved for the map of the institutions that result from the search. To the left, there is a list composed of cards for each of these institutions. These cards have the name of the institution, its address, its average rating, and a photo. On the very bottom of the page, the footer is still present as well.

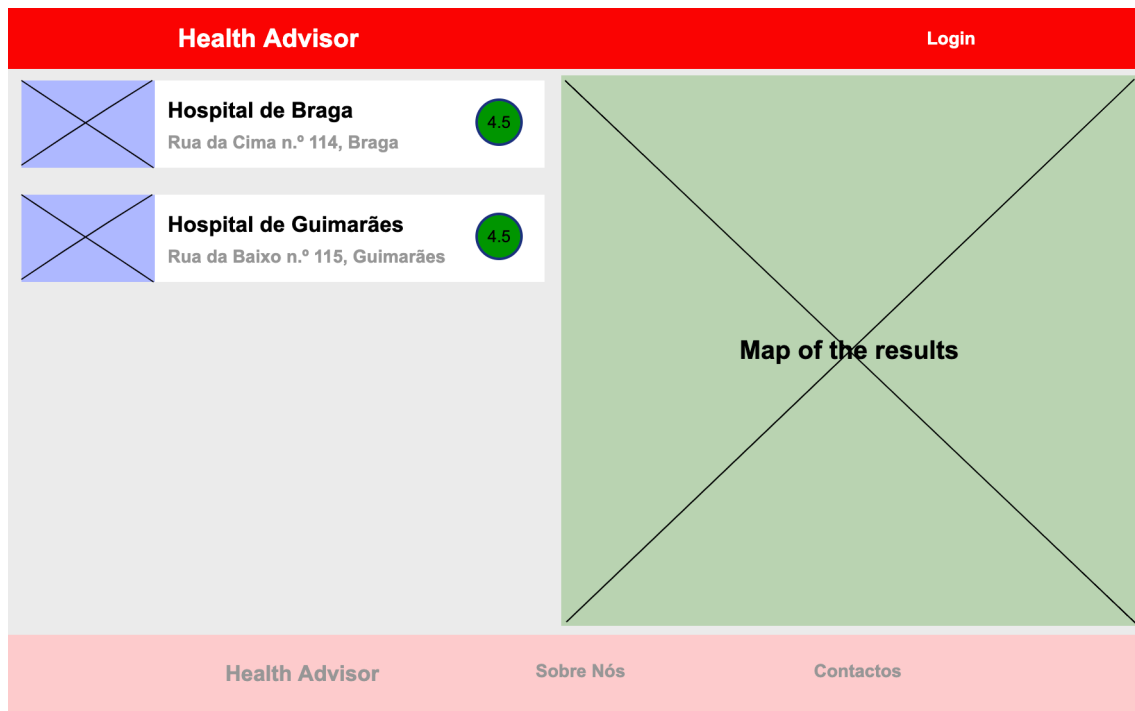


Figure 7: Search results page mockup

3.3.1.5 Institution Page

At the top, the institution's page (Figure 8) features a card similar to the one seen on the search results page (Figure 7). This card, however, is enlarged and features additional information like contacts and a bigger, more detailed average rating area. Below, two different sections appear. On the left, there is an area for a user to submit a review for that institution. Below, the reviews made by other users also appear. On the right, there is a section that comprises a map to the institution's address, as well as the specialties and subsystems the institution offers.

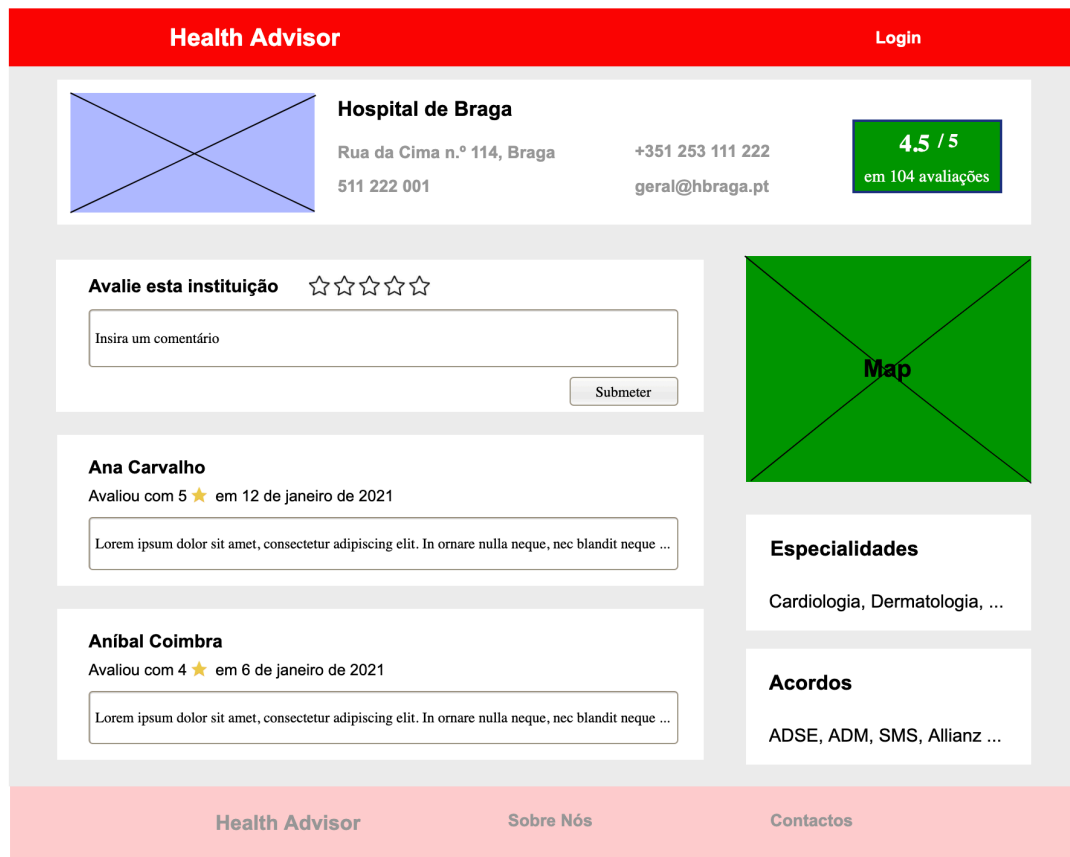


Figure 8: Institution page mockup

3.3.2 Final Result

In this section, the final result of the implementation of the User Interface is presented. This implementation was based on the mockups presented previously (Section 3.3.1).

3.3.2.1 Home Page

The upper part of the homepage consists of the navigation bar. On the left-hand side, there's a reference to the name of the platform as well as a logo. On the right-hand side, there is an outlined button reserved for the login of the user.

Directly below the navigation bar, there is the *search form*. In this form, the user can choose the type of search: either by "appointment"(Figure 9) or by "institution"(Figure 10). Selecting one or the other changes the form fields without reloading the page. This effectively combines the two forms presented in the mockup (4) in a simpler, smaller format.

The first type of search is displayed in Figure 9 and includes a field for the medical specialty, the health subsystem or insurance company, and the location. The second type of search is presented in Figure 10

and, in turn, includes only two fields: one for the name (complete or partial) of the institution and another for its location.

All the fields of both variations of this search form have a small line below each with examples of valid inputs. This, along with the accompanying icons, depictive of said fields, serve as an aid to better guide the user. To the right side, there is also a button that takes the user to the search results page, described in the following subsection.

 The screenshot shows the 'Appointment' search form on the Health Advisor homepage. At the top, there is a red header with the 'Health Advisor' logo and a 'LOGIN' button. Below the header, the text 'Pretendo pesquisar por' is followed by a dropdown menu set to 'Consulta'. There are three input fields: 'Especialidade' (with a plus icon and example 'e.g. Oftalmologia, Cardiologia'), 'Subsistema' (with a document icon and example 'e.g. ADSE, seguros privados'), and 'Localização' (with a location pin icon and example 'e.g. Distrito, cidade'). A green 'Procurar' button with a magnifying glass icon is on the right.

Figure 9: Homepage – "Appointment" search

 The screenshot shows the 'Institution' search form on the Health Advisor homepage. It has the same red header with the 'Health Advisor' logo and 'LOGIN' button. The text 'Pretendo pesquisar por' is followed by a dropdown menu set to 'Instituição'. There are two input fields: 'Instituição de Saúde' (with a plus icon and example 'e.g. Hospitais, Clínicas') and 'Localização' (with a location pin icon and example 'e.g. Distrito, Cidade'). A green 'Procurar' button with a magnifying glass icon is on the right.

Figure 10: Homepage – "Institution" search

3.3.2.2 Login Page

The final result that implements the mockup for the login page (Figure 5) can be seen in Figure 11.

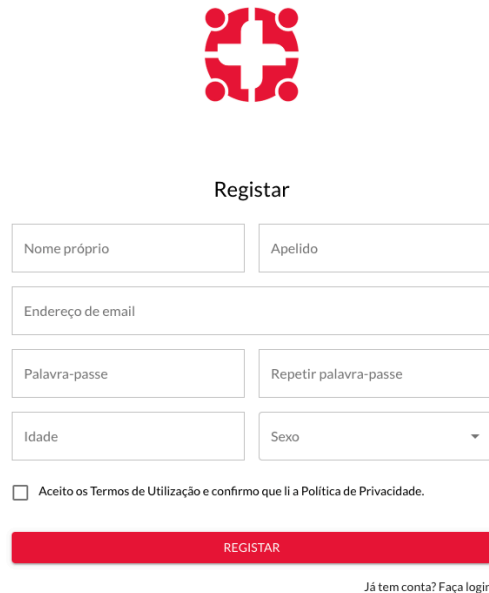
 The screenshot shows the login page on the Health Advisor website. At the top, there is a red header with the 'Health Advisor' logo. Below the header, the text 'Iniciar sessão' is centered. There are two input fields: 'Endereço de email' and 'Palavra-passe'. A red 'INICIAR SESSÃO' button is below the fields. At the bottom, there is a link that says 'Ainda não tem conta? Registe-se'.

Figure 11: Login page

3.3.2.3 Register Page

The final result that implements the mockup for the login page (Figure 6) can be seen in Figure 11.

×



The image shows a web form titled "Registrar" for user registration. At the top is a red logo consisting of four stylized human figures holding hands in a square. Below the logo, the form contains several input fields: "Nome próprio" and "Apelido" (first and last name), "Endereço de email" (email address), "Palavra-passe" (password), and "Repetir palavra-passe" (repeat password). There are also fields for "Idade" (age) and "Sexo" (gender) with a dropdown arrow. Below these fields is a checkbox labeled "Aceito os Termos de Utilização e confirmo que li a Política de Privacidade." At the bottom is a large red button labeled "REGISTAR". Below the button is a link that says "Já tem conta? Faça login".

Figure 12: Register page

3.3.2.4 Search Results Page

Upon clicking the search button on the homepage (Section 3.3.2.1), the search results page appears to the user.

This page also includes the same navigation bar as it was presented on the homepage. However, the search form is now condensed, to make room for the search results that follow below. Although condensed, the continued presence of this area facilitates the possibility of making another search, as the form fields can be edited and the search button clicked again in order to yield new results.

Below this, the results area takes most of the space of this page.

To the left-hand side, a list containing the institutions that matched the user's query appears. This list includes the details of said institutions (name, location, average rating, and an image) in a series of cards. As it may happen that not all the results can be rendered on the viewport (Figure 13), this list is scrollable. Additionally, the user can click on each of the cards to be redirected to a page containing more information about the institution (Figure 15).

On the right-hand side, there is a map displaying a marker for each of the institutions. Clicking a marker displays the name of the institution it refers to. The zoom level of the map can be augmented or diminished using the plus and minus icons, respectively, on the top left corner. The user can also browse

the map freely. However, scrolling on the list of results to the left doesn't affect the display of the map whose dimensions don't surpass the limits of the viewport.

Below, Figure 13 represents the search of type "appointment".

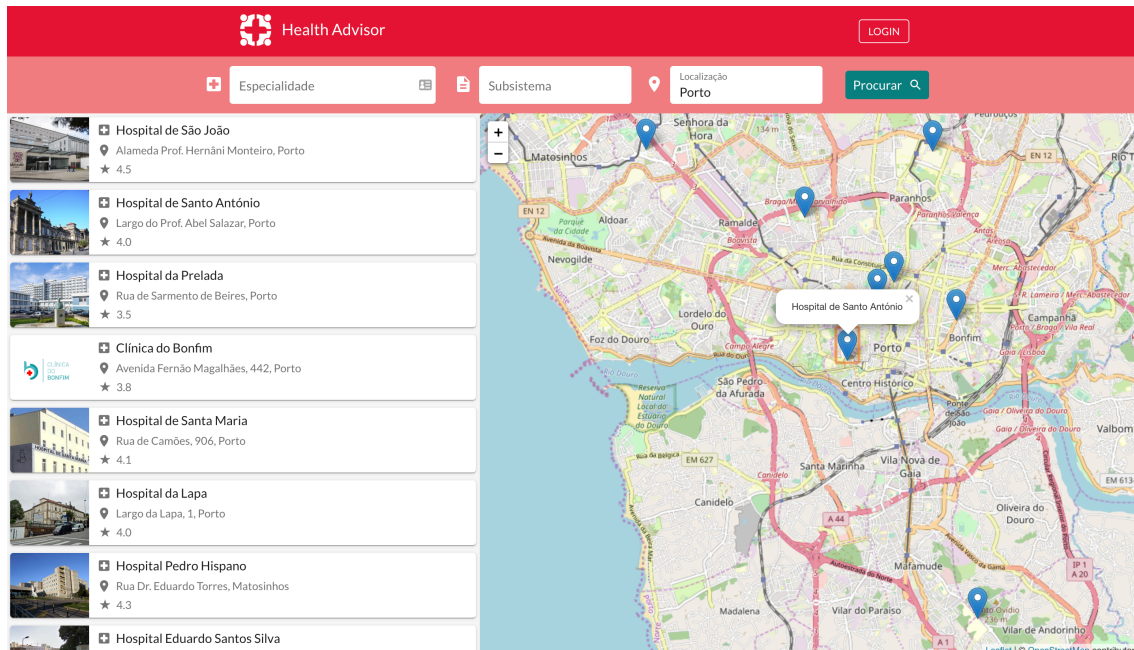


Figure 13: "Appointment" search results page (location)

In Figure 14, we can see the results of the "Institution" search.

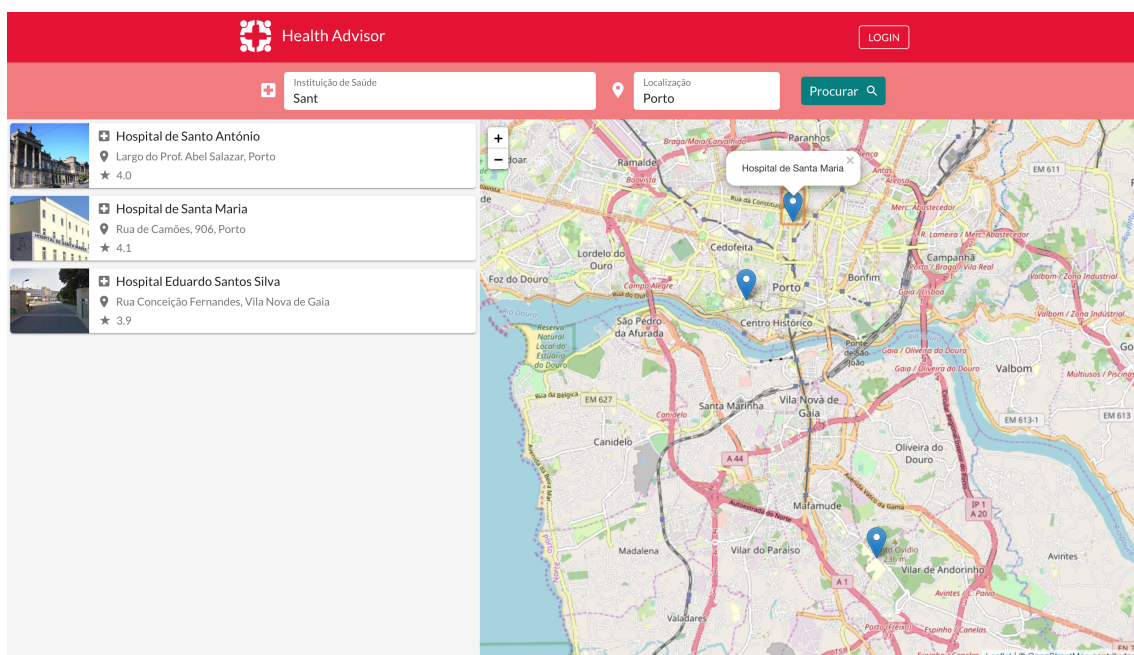


Figure 14: "Institution" search results page (name and location)

Obtaining and rendering map information For the display of the map, an external API was used. The chosen one was OpenStreetMap⁴⁵ because it is a Free and Open Source Software (FOSS) that fits the necessary requirements: displaying a map tile and point markers given a set of one or more coordinates. Additionally, OpenStreetMap is a project that is community-driven – a value shared with the vision behind this very project. Indeed, the project’s data is crowdsourced, and is collected in multiple ways by its users: Global Positioning System (GPS) devices, aerial photography, free sources, or simply from the users’ own knowledge of a certain area. This data is made available and freely distributed under the Open Database License (ODbL), a license that is published by Open Data Commons (part of the Open Knowledge Foundation).

More precisely, the Open Street Map’s Tiles API⁶ was used. This API provides 256 by 256 pixel Portable Network Graphics (PNG) files that correspond to a portion of the world map. It is available in [http://\[abc\].tile.openstreetmap.org/{zoom}/{x}/{y}.png](http://[abc].tile.openstreetmap.org/{zoom}/{x}/{y}.png), where *zoom* is the zoom level of the image, *x* and *y* are the tile numbers, and *a*, *b* or *c* are the names of the subdomains of the tile server. The tile numbers can be derived⁷ from GPS coordinates, according to the two equations below, where lon_{deg} and lat_{rad} are the longitude in degrees and the latitude in radians, respectively.

$$x_{tile} = 2^{zoom} * \frac{lon_{deg} + 180}{360}$$

$$y_{tile} = 2^{zoom} * \frac{1 - (\frac{\log(\tan(lat_{rad}) + \sec(lat_{rad}))}{\pi})}{2}$$

The Open Street Map’s Marker API⁸ was used to display the markers, given a set of coordinates.

Additionally, the Leaflet Javascript library⁹¹⁰ was used to display the interactive maps on the webpage. This library is also a FOSS under the Berkeley Source Distribution (BSD) open-source license that has support for multiple map providers, abstracting some of their inner functionality in order to be easy to use. Listing 1 shows an example of the code necessary to render a map tile with a marker on a page.

```
var map = Leaflet.map('map').setView([51.505, -0.09], 13);

Leaflet.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
    ↪ contributors'
}).addTo(map);
```

⁴<https://www.openstreetmap.org/>

⁵https://wiki.openstreetmap.org/wiki/Main_Page

⁶<https://wiki.openstreetmap.org/wiki/Tiles>

⁷https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames#Implementations

⁸https://wiki.openstreetmap.org/wiki/Marker_API

⁹<https://leafletjs.com/>

¹⁰<https://github.com/Leaflet/Leaflet>

```
Leaflet.marker([51.5, -0.09])  
  .addTo(map)  
  .bindPopup('A pretty CSS3 popup.<br> Easily customizable.')
```

Listing 1: Example of a map tile with a marker using Leaflet and OpenStreetMap

3.3.2.5 Institution Page

In Figure 15, the page of an institution is presented.

Here, we can see that below the navigation bar, the header of the page appears. This header contains some information about the institution: its name, location, and contact information. It also contains the average rating of the institution, as well as the total number of reviews submitted by the users.

To the right, we can see a more detailed view of the institution's location, the listings of the medical specialties the institution has, and the subsystems it has agreements with.

To the left, an area where the user can submit a new review for that institution appears.

Below that, the reviews of other users are presented. These contain the name of the user, the rating they gave, a comment, as well as the date on which the review was submitted to the system. Additionally, the user can scroll on the page to see more reviews made by other users.

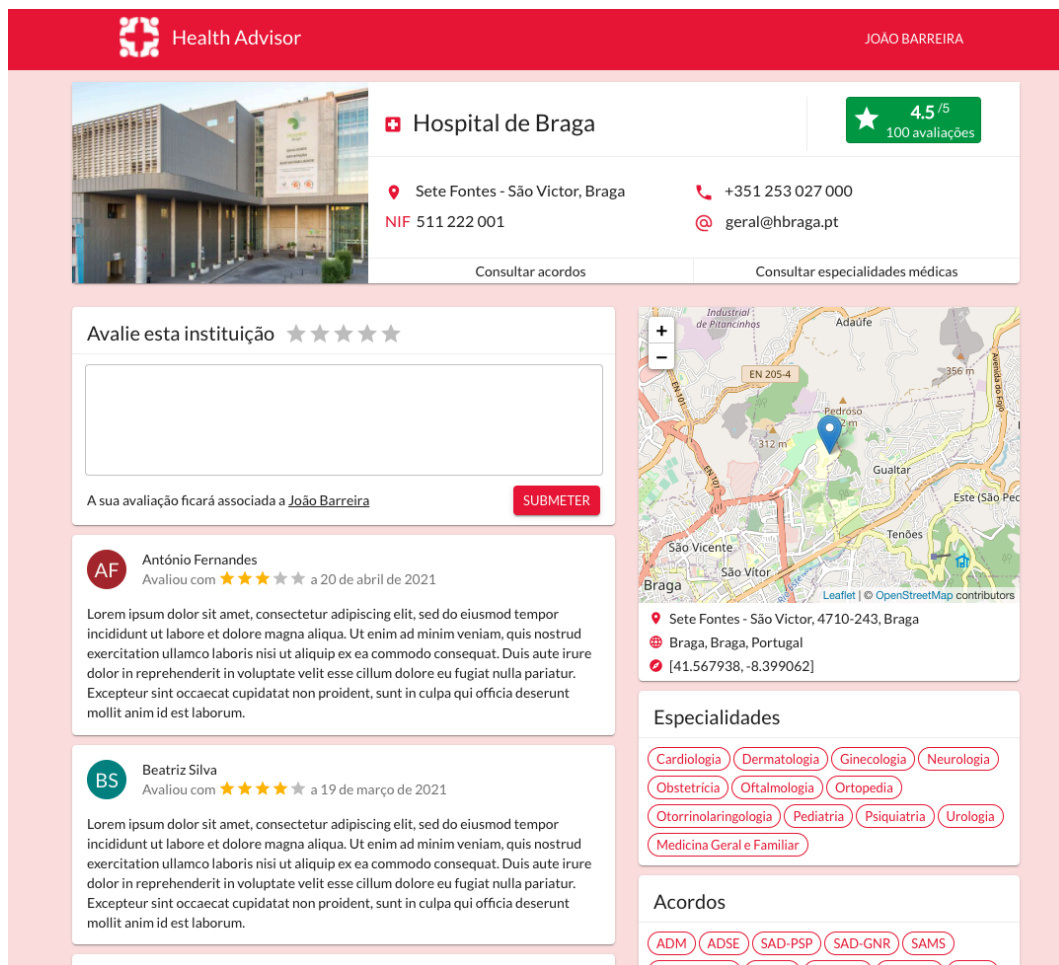


Figure 15: Institution page

For the display of the map, the same tools as those enunciated in Section 3.3.2.4 were used.

3.3.3 Tests

In order to test the visual rendering of the pages developed, a suite of tests was developed. The main goals behind this effort were to increase the reliability and the quality of the software delivered, as well as ensure that the user interface doesn't suffer unintended changes with further developments of the codebase.

In the case of the frontend, the test suite corresponds to various snapshot tests¹¹ for the React components that make up each of the pages. In these types of tests, a snapshot file that contains the representation of the Document Object Model (DOM)¹² for that particular component is generated (Listing 2).

¹¹<https://jestjs.io/docs/snapshot-testing>

¹²https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

```

exports[`Navbar matches the snapshot 1`] = `
  <WithStyles(ForwardRef(AppBar)) elevation={0} position="sticky">
    <WithStyles(ForwardRef(Container)) maxWidth="md">
      <WithStyles(ForwardRef(Toolbar))>
        <WithStyles(ForwardRef(Grid)) alignItems="center" container={true} direction="row"
          ↪ justify="center" spacing={2}>
          <WithStyles(ForwardRef(Grid)) item={true}>
            
          </WithStyles(ForwardRef(Grid))>
          <WithStyles(ForwardRef(Grid)) className="makeStyles-brand-1" item={true}>
            <WithStyles(ForwardRef(Typography)) variant="h6">
              Health Advisor
            </WithStyles(ForwardRef(Typography))>
          </WithStyles(ForwardRef(Grid))>
          <WithStyles(ForwardRef(Grid)) item={true}>
            <WithStyles(ForwardRef(Button)) color="inherit" size="small" variant="outlined">
              Login
            </WithStyles(ForwardRef(Button))>
          </WithStyles(ForwardRef(Grid))>
        </WithStyles(ForwardRef(Grid))>
      </WithStyles(ForwardRef(Toolbar))>
    </WithStyles(ForwardRef(Container))>
  </WithStyles(ForwardRef(AppBar))>
`;

```

Listing 2: Example of a snapshot file for a React component

Afterward, one can construct a test function that renders the component being tested and compares it with the previously generated snapshot. These types of tests are especially useful as a means of being certain that a new change in the component doesn't trigger unexpected visual changes on its user interface. Multiple tools provide this functionality. Among those is Enzyme¹³ that provides a shallow rendering¹⁴ functionality that is useful for testing a component as a unit, without triggering side effects on its children. An example of such a test is displayed in Listing 3.

¹³<https://enzymejs.github.io/enzyme/>

¹⁴<https://enzymejs.github.io/enzyme/docs/api/shallow.html>

```

import React from 'react';
import { shallow } from 'enzyme';
import toJson from 'enzyme-to-json';
import Navbar from './Navbar';

describe('Navbar', () => {
  it('matches the snapshot', () => {
    const wrapper = shallow(<Navbar />);
    expect(toJson(wrapper)).toMatchSnapshot();
  });
});

```

Listing 3: Example of a snapshot test

3.4 HealthAdvisor Backend

The HealthAdvisor Recommendations (Figure 16) is the system responsible for exposing the APIs (3.4.2) used by the Frontend and Recommendations as well as acting as the persistence layer of the application (3.4.1).

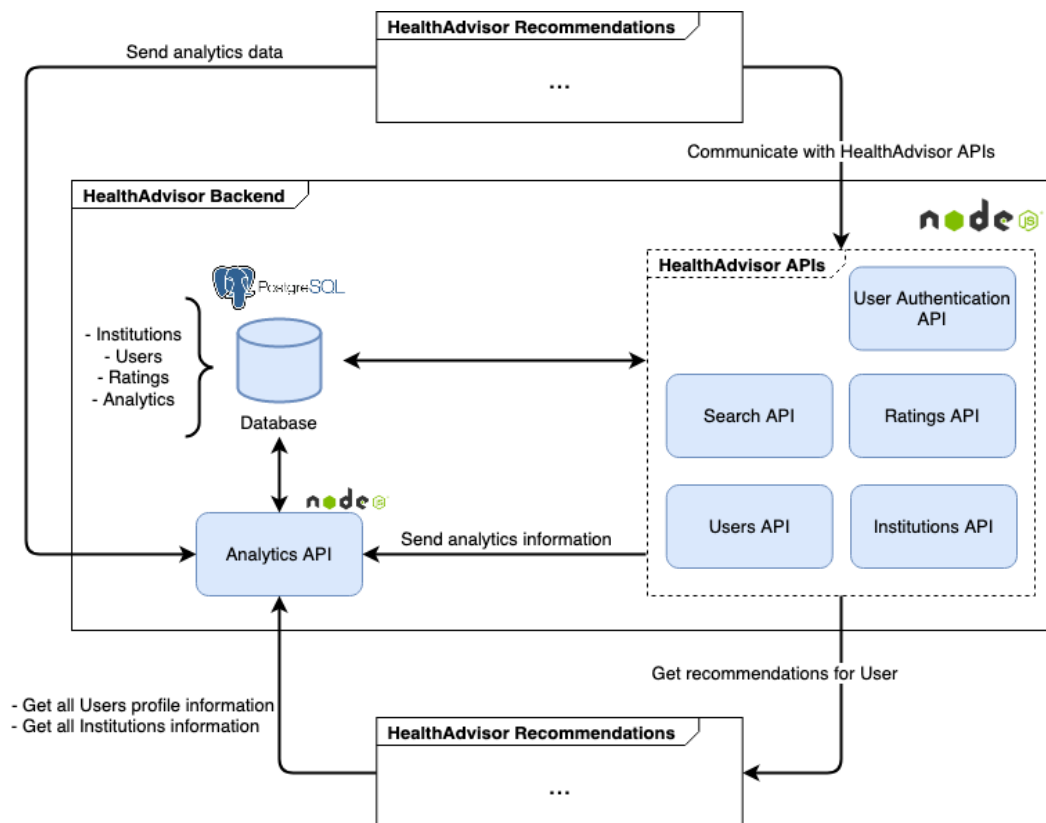


Figure 16: HealthAdvisor Backend Architecture

This system is built with NodeJS¹⁵. One of the advantages of choosing this language is the fact that it has a high performance and scalability due to its non-blocking input/output and asynchronous request handling which enables processing multiple requests without any delay and handling these requests asynchronously in a native manner through the utilization of callbacks. Another advantage is the uniformization of the technological tool stack used for the project, as the frontend also uses Javascript. It also has a rich ecosystem and a solid package manager, *npm*¹⁶, that features a high number of useful packages which are often community-driven and open source. One of the potential drawbacks of NodeJS is performance bottlenecks in CPU-heavy scenarios, which was not deemed particularly deterrent for this specific project, as its purpose would be to serve and expose different APIs, without any heavy computation.

3.4.1 Data Storage

The first step in the implementation of the HealthAdvisor Backend was the definition of the entities, the relationships between each other, as well as its attributes, in a logical schema (Figure 17). In order to achieve this, the entities of the domain model described in the previous section were translated into their own tables containing their attributes (as fields).

These tables were designed to be used both by the HealthAdvisor APIs (Figure 3.4, right) as well as the Analytics API (Figure 3.4, left).

¹⁵<https://nodejs.org/en/>

¹⁶<https://www.npmjs.com/>

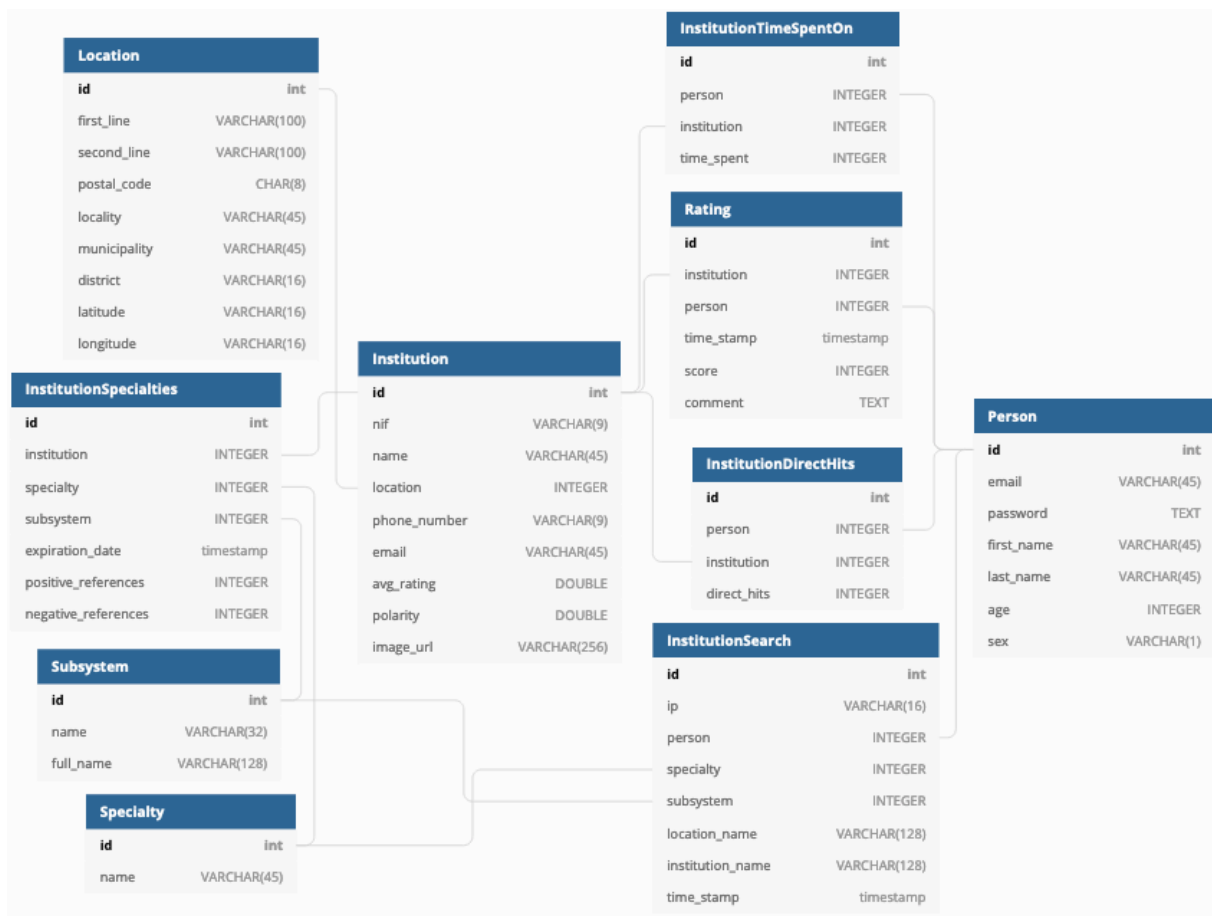


Figure 17: Logical schema

For this, a relational Postgres¹⁷ database was used. This Relational Database Management System (RDBMS) was chosen because it is a free, open-source, and proven solution with a lot of features like transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. Despite its great range of features it also has high extensibility support, which enables users to extend many of its functionalities. It also has high reliability and disaster recovery functionality via its Write-Ahead Logging (WAL), replication, and Point-In-Time-Recovery (PITR).

3.4.2 APIs

In the following sections, the APIs that the HealthAdvisor Backend exposes are enumerated and explained in better detail.

3.4.2.1 Users API

The Users API implements the functionality of creating, updating, deleting, and finding users.

¹⁷<https://www.postgresql.org/>

- *findById* – Finds a User by their ID (Table 5).
- *findAll* – Finds a User by their ID (Table 6).
- *create* – Creates a new User (Table 7).
- *update* – Updates the contents of an existing User (Table 8).
- *delete* – Deletes an existing User (Table 9).

HTTP Method	GET
Endpoint	/users/:id
Summary	Finds a User by their ID
Example body	None
Example response	<pre>{ "id": 1, "email": "a.antunes@email.com", "first_name": "Ana", "last_name": "Antunes", "sex": "F", "age": "32" }</pre>

Table 5: *findById* API request specification (Users API)

HTTP Method	GET
Endpoint	/users
Summary	Gets all Users
Example body	None
Example response	<pre>[{ "id": 1, "email": "a.antunes@email.com", "first_name": "Ana", "last_name": "Antunes", "sex": "F", "age": "32" }, ... { "id": 5, "email": "e.esteves@email.com", "first_name": "Elísio", "last_name": "Esteves", "sex": "M", "age": "51" }]</pre>

Table 6: *findAll* API request specification (Users API)

HTTP Method	POST
Endpoint	/users
Summary	Creates a new User
Example body	<pre>{ "email": "a.antunes@email.com", "first_name": "Ana", "last_name": "Antunes", "sex": "F", "age": "32" }</pre>
Example response	<pre>{ "id": 1, "email": "a.antunes@email.com", "first_name": "Ana", "last_name": "Antunes", "sex": "F", "age": "32" }</pre>

Table 7: *create* API request specification (Users API)

HTTP Method	PATCH
Endpoint	/users/:id
Summary	Updates the contents of a User
Example body	<pre>{ "email": "a.antunes@email.com", "first_name": "Anabela", "last_name": "Antunes", "sex": "F", "age": "32" }</pre>
Example response	<pre>{ "id": 1, "email": "a.antunes@email.com", "first_name": "Anabela", "last_name": "Antunes", "sex": "F", "age": "32" }</pre>

Table 8: *update* API request specification (Users API)

HTTP Method	DELETE
Endpoint	/users/:id
Summary	Deletes a User
Example body	None
Example response	<pre>{ "deletedUserId": "1" }</pre>

Table 9: *delete* API request specification (Users API)

3.4.2.2 Institutions API

The Institutions API implements the functionality for finding Institutions and getting the information regarding all the institutions' specialties, subsystems, and locations.

- *findById* – Finds an Institution by its ID (Table 10).
- *getAllSpecialties* – Returns a list of all existing Specialties (Table 11).
- *getAllSubsystems* – Returns a list of all existing Subsystems (Table 12).
- *getAllLocations* – Returns a list of all existing Locations (Table 13).

HTTP Method	GET
Endpoint	/institutions/:id
Summary	Finds an Institution by its ID
Example body	None
Example response	<pre>{ "id": 1, "nif": "511222001", "name": "Hospital de Braga", "location": { "id": 1, "first_line": "Sete Fontes - São Victor", "second_line": "", "postal_code": "4710-243", "locality": "Braga", "municipality": "Braga", "district": "Braga", "latitude": "41.567938", "longitude": "-8.399062" }, "phone_number": "253027000", "email": "geral@hbraga.pt", "avg_rating": 4.0 }</pre>

Table 10: *findById* API request specification (Institutions API)

HTTP Method	GET
Endpoint	/institutions/specialties
Summary	Gets all existing Specialties
Example body	None
Example response	<pre>[{ "id": 1, "name": "Cardiologia" }, ... { "id": 12, "name": "Medicina Geral e Familiar" }]</pre>

Table 11: *getAllSpecialties* API request specification (Institutions API)

HTTP Method	GET
Endpoint	/institutions/subsystems
Summary	Gets all existing Subsystems
Example body	None
Example response	<pre>[{ "id": 1, "name": "ADM", "full_name": "Assistência na Doença aos ↳ Militares das Forças Armadas" }, ... { "id": 7, "name": "Allianz", "full_name": "Allianz" }]</pre>

Table 12: *getAllSubsystems* API request specification (Institutions API)

HTTP Method	GET
Endpoint	/institutions/specialties
Summary	Gets all existing Specialties
Example body	None
Example response	<pre>[{ "locality": "Braga", "municipality": "Braga", "district": "Braga" }, ... { "locality": "Guimarães", "municipality": "Guimarães", "district": "Braga" }]</pre>

Table 13: *getAllLocations* API request specification (Institutions API)

3.4.2.3 Search API

The Search API implements the functionality to search for Institutions.

- *findBySpecialtySubsystemLocation* – Finds an Institution by a given specialty and/or subsystem and/or location (Table 14).
- *findByNameLocation* – Finds an Institution by its name and/or location (Table 15).

HTTP Method	GET
Endpoint &location=Braga	/institutions?specialty=Cardiologia&subsystem=Allianz
Summary	Finds an Institution by a given specialty and/or subsystem and/or location
Example body	None
Example response	<pre>[{ "id": 2, "nif": "511222002", "name": "Hospital Privado de Braga", "location": { "id": 2, "first_line": "Lugar da Igreja Nogueira", "second_line": "", "postal_code": "4715-196", "locality": "Braga", "municipality": "Braga", "district": "Braga", "latitude": "41.524688", "longitude": "-8.414187" }, "phone_number": "253680200", "email": "geral@hpbraga.pt", "avg_rating": 4.0 }]</pre>

Table 14: *findBySpecialtySubsystemLocation* API request specification (Institutions API)

HTTP Method	GET
Endpoint	/institutions?institutionName=Braga&location=Braga
Summary	Finds an Institution by a given name and/or location
Example body	None
Example response	<pre>[{ "id": 1, "nif": "511222001", "name": "Hospital de Braga", "location": { "id": 1, "first_line": "Sete Fontes - São Victor", "second_line": "", "postal_code": "4710-243", "locality": "Braga", "municipality": "Braga", "district": "Braga", "latitude": "41.567938", "longitude": "-8.399062" }, "phone_number": "253027000", "email": "geral@hbraga.pt", "avg_rating": 4.0 }, { "id": 2, "nif": "511222002", "name": "Hospital Privado de Braga", "location": { "id": 2, "first_line": "Lugar da Igreja Nogueira", "second_line": "", "postal_code": "4715-196", "locality": "Braga", "municipality": "Braga", "district": "Braga", "latitude": "41.524688", "longitude": "-8.414187" }, "phone_number": "253680200", "email": "geral@hpbraga.pt", "avg_rating": 4.0 }]</pre>

Table 15: *findByNameLocation* API request specification (Institutions API)

3.4.2.4 Ratings API

The Ratings API implements the functionality to add a new Institution review.

- *addInstitutionReview* – Adds a new Institution review (Table 16).

HTTP Method	POST
Endpoint	/institutions/:id/reviews
Summary	Adds a new Institution review
Example body	<pre>{ "userId": "3", "rating": "4.0", "comment": "I like this hospital." }</pre>
Example response	<pre>{ "id": "1", "institutionId": "10" "userId": "3", "rating": "4.0", "comment": "I like this hospital." }</pre>

Table 16: *addInstitutionReview* API request specification (Ratings API)

3.4.2.5 Analytics API

The Analytics API implements the functionality that is related to the persistence of new analytic data. It also serves as the endpoint that the HealthAdvisor Recommendations 3.5 uses in order to get the information needed to construct the Users and Institutions datasets.

- *addInstitutionPageVisit* – Adds a new occurrence of an institution’s page visit by a user (Table 17).
- *addTimeSpentOn* – Increases the time spent by a User on an Institution’s page (Table 18)
- *getAllInstitutionsData* – Gets all Institutions’ data to construct the Institutions dataset (Table 19).
- *getUserProfile* – Gets the user profile for a given user (Table 20). This includes: age, sex, name, search specialties, searched subsystems, and institutions the user is interested in. These institutions include institutions the user has rated positively, visited more times, or institutions in which pages they have spent more time.
- *getAllUsersProfileData* – Gets all users’ data to construct the Users dataset (Table 21). This corresponds to the user profiles of all the existing users.

HTTP Method	POST
Endpoint	/analytics/institutions/:id/visits
Summary	Adds a new institution page visit
Example body	<pre>{ "userId": "3", "institutionId": "2" }</pre>
Example response	<pre>{ "id": "4", "userId": "3", "institutionId": "2", "visits": "1" }</pre>

Table 17: *addInstitutionPageVisit* API request specification (Analytics API)

HTTP Method	POST
Endpoint	/analytics/institutions/:id/time
Summary	Increases the time spent by a User on an Institution's page
Example body	<pre>{ "userId": "3", "institutionId": "2", "timeSpent": "2", }</pre>
Example response	<pre>{ "id": "4", "userId": "3", "institutionId": "2", "timeSpent": "2" }</pre>

Table 18: *addTimeSpentOn* API request specification (Analytics API)

HTTP Method	GET
Endpoint	/analytics/institutions
Summary	Gets all Institutions' data
Example body	None
Example response	<pre>[{ "id": 1, "name": "Hospital de Braga", "location": { "id": 1, ... "latitude": "41.567938", "longitude": "-8.399062" }, "avg_rating": 4.5, "specialties": [{ "id": 3, "name": "Ginecologia" }, ... { "id": 9, "name": "Pediatria" }], "agreements": [{ "id": 3, "name": "SAD-PSP" }, ... { "id": 9, "name": "Multicare" }] }, ... { "id": 12, "name": "Hospital de Guimarães", "location": { "id": 3, ... "latitude": "41.441938", "longitude": "-8.305188" }, "avg_rating": 4.3, "specialties": [{ "id": 1, "name": "Cardiologia" }, ... { "id": 10, "name": "Psiquiatria" }], "agreements": [{ "id": 1, "name": "ADM" }, ... { "id": 10, "name": "Tranquilidade" }] }, ...]</pre>

Table 19: *getAllInstitutionsData* API request specification (Ratings API)

HTTP Method	GET
Endpoint	/analytics/users/:id
Summary	Gets a user profile of a user
Example body	None
Example response	<pre> { "id": 1, "age": 63, "sex": "F", "searchedSpecialties": [{ "id": 1, "name": "Cardiologia" }, { "id": 11, "name": "Ginecologia" }], "searchedSubsystems": [{ "id": 2, "name": "ADSE" }], "institutionsInterestedIn": [{ "id": 2, "name": "Hospital Privado de Braga" } { "id": 1, "name": "Hospital de Braga" },] } </pre>

Table 20: *getUserProfile* API request specification (Ratings API)

HTTP Method	GET
Endpoint	/analytics/users
Summary	Gets the user profile of all users
Example body	None
Example response	<pre>[{ "id": 1, "age": 63, "sex": "F", "searchedSpecialties": [{ "id": 1, "name": "Cardiologia" }, { "id": 11, "name": "Ginecologia" }], "searchedSubsystems": [{ "id": 2, "name": "ADSE" }], "institutionsInterestedIn": [{ "id": 2, "name": "Hospital Privado de Braga" }, { "id": 1, "name": "Hospital de Braga" },] }, { "id": 5, "age": 49, "sex": "F", "searchedSpecialties": [{ "id": 3, "name": "Dermatologia" }], "searchedSubsystems": [], "institutionsInterestedIn": [{ "id": 9, "name": "Hospital da Lapa" }, { "id": 10, "name": "Hospital Pedro Hispano" },] }]</pre>

Table 21: *getAllUsersProfileData* API request specification (Ratings API)

3.4.3 Tests

In order to test their functionality, some API tests were developed using Jest¹⁸ and Supertest¹⁹. Jest is an open-source javascript testing framework developed and maintained by Facebook. One of its main goals is to be simple to use and configure. It features a testing script that runs all the tests in the project and has support for various metrics such as code coverage. It also features an assertion library²⁰ that has support for many types of assertions and verifications. Supertest is an easy-to-use library that provides a high-level abstraction for testing HTTP functionality. In this particular case, it was used in the test files in

¹⁸<https://jestjs.io/>

¹⁹<https://github.com/visionmedia/supertest>

²⁰<https://jestjs.io/docs/expect>

order to be able to easily perform the HTTP requests to the different endpoints that were developed and that needed to be tested.

Figure 18 shows an example of the tests developed for some of the functionalities of Institutions API and Search API.



Figure 18: Institutions/Search API tests

3.5 HealthAdvisor Recommendations

The HealthAdvisor Recommendations (Figure 19) is the system responsible for generating item recommendations. It encompasses three different subsystems: the Recommendations API (3.5.1), the Dataset Builder (3.5.2), and the Recommender System (3.5.3 and 3.6).

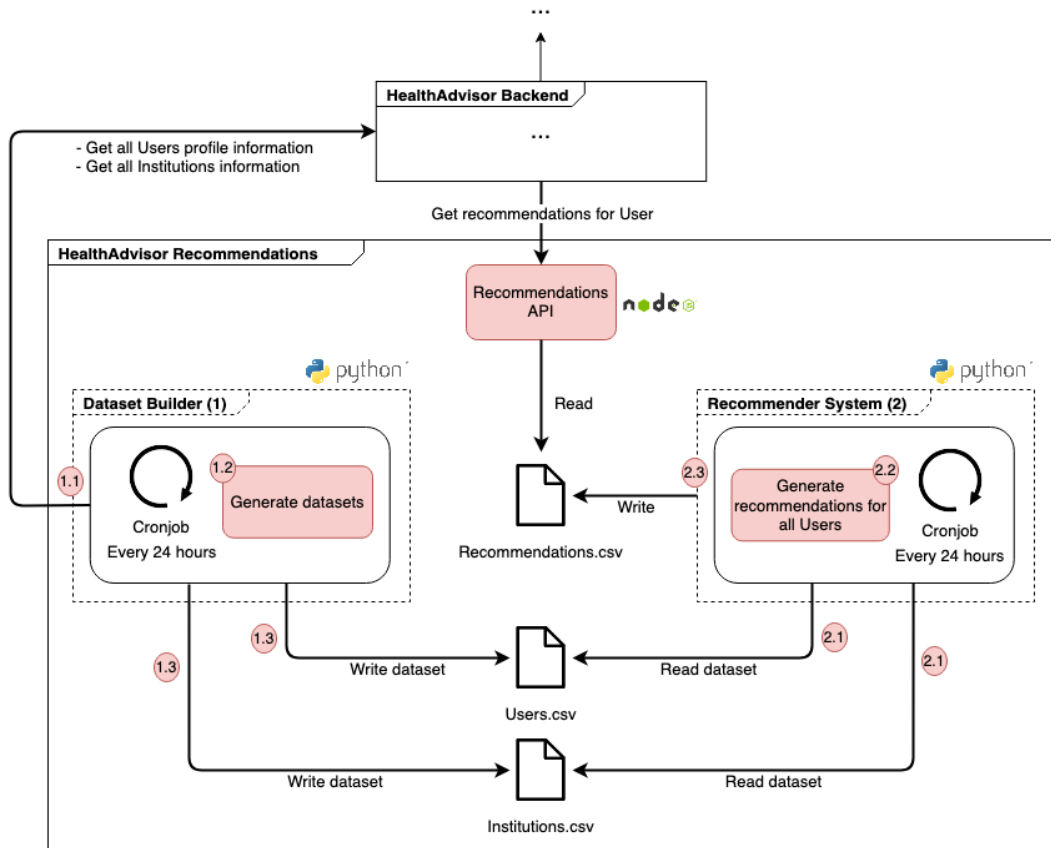


Figure 19: HealthAdvisor Recommendations Architecture

3.5.1 Recommendations API

The Recommendations API is the component that is exposed to the outside. It does this by means of an API that is responsible for receiving the ID of a given user registered in the system and returning the pre-computed list of recommendations the Recommender System (3.5.3) has generated for that specific user ahead of time. It does this by reading the *Recommendations.csv* file that was created by the Recommender System. This file contains all the generated recommendations for every user in the system. Therefore, it is only necessary to look for the recommendations for the user being queried and return them.

3.5.2 Dataset Builder

The Dataset Builder is the component responsible for the generations of the datasets that are used by the Recommender System (3.5.3). There are two of such datasets: the Institutions dataset and the Users dataset, containing various information regarding the system's institutions and users, respectively.

Institutions Dataset This dataset contains the information relating to the system's institutions that will be used by the Recommender System in order to generate new recommendations. In order to obtain this

information, it queries the HealthAdvisor Backend's Institutions API (3.4) via an HTTP GET request to the `/analytics/institutions` endpoint. By doing so, it obtains the information that relates to all the registered institutions in the system. More specifically, it obtains each institution's profile (e.g. id, name, location, average rating, etc.) along with its specialties and subsystems. Finally, it saves this information to disk in an *Institutions.csv* file.

Users Dataset This dataset contains the information relating to the system's users that will be used by the Recommender System in order to generate new recommendations. In order to obtain this information, it queries the HealthAdvisor Backend's Users API (3.4) via an HTTP GET request to the `/analytics/users` endpoint. Thereby, it receives the information relating to all the registered users in the system. This information includes both the user profile (e.g. id, age, sex, etc.) as well as their analytics information such as the institutions they are interested in and data relating to their past searches (e.g. specialties, subsystems, etc.). Then, all this information is saved to disk in a *Users.csv* file.

By default, this component runs every 24 hours. To accomplish this, a python script that includes the process described above was developed. This script is run via a *cronjob* – a time-based utility available across Unix-like operating systems – that queries the necessary HealthAdvisor Backend endpoints and saves the information to the respective files.

3.5.3 Recommender System

The Recommender System is the component responsible for generating the recommendations. In order to achieve this, it reads the datasets that were constructed by the Dataset Builder 3.5.2 and that contain information relating to the entirety of the system's users and institutions. Afterward, this information is used by the Recommender System to generate new recommendations for each of the users in the system. These recommendations will then be saved onto a *Recommendations.csv* file which will be read when a new request to get the recommendations for a given user is received.

By default, the Recommender System runs every 24 hours – after the execution of the Dataset Builder – by means of a python script that is executed via a *cronjob*.

The inner functionality of the Recommender System is documented in detail in Section 3.6.

3.6 Recommender System

As said earlier, the Recommender System is the component of HealthAdvisor Recommendations responsible for the generating new recommendations. In the following sections, its architecture and inner functionality are further detailed.

3.6.1 Recommender System Architecture

Choice of Recommender System Paradigm In order to choose the most appropriate recommender system paradigm, the study done in the State of the Art (Section 2) was consulted. More specifically, the comparison presented in Table 4 was taken into consideration. Another aspect that was crucial to this choice were the characteristics inherent to the domain of the application.

From all the limitations listed in the table for the different paradigms, we began by choosing the ones that were considered the most detrimental, given the domain of the application. More specifically, the cold start and the sparsity problems were chosen. The cold start problem was considered relevant because of the fact that the recommender system was to be built in the context of a new application that didn't yet have any real users. The sparsity problem was also considered pertinent because of the assumption that in the context of a recommender system built for healthcare institutions, the users would probably not have a tendency to visit many different healthcare institutions, at least when compared to other – perhaps more usual – fields of application such as a movie rating applications or book review platforms. In these contexts of application, it seemed safe to assume that the users would consume items at a much bigger rate. Additionally, healthcare services are, in general, concentrated in bigger hospitals, with small clinics being visited at a frequency many times lower. Under this assumption, a sparsity problem could arise from the fact that the majority of the users would interact with the bigger institutions, while the small clinics would have few to no users interacting with them.

In light of this, Content-based Filtering seemed to be the better choice. However, this paradigm also has other types of limitations, and, therefore, they needed to be addressed.

- *New user problem* – This problem could arise from the fact that when a new user joined the platform, the system would have no information about their item preference. Therefore, it would not be able to calculate the similarity between their preferred items, and the other existing items in the system, so as to generate new recommendations. In order to mitigate this problem, it was decided that during the user registration, a short questionnaire (Figure 20) could be shown to the user, asking for the medical specialties or subsystems they were most interested in. The location could be inferred from the IP information of the HTTP request (through *geolocation*).
- *No serendipitous recommendations* – In the current context of application, the lack of serendipity in the recommendations was not deemed as being a prohibitively serious problem. This was considered under the assumption that the healthcare field – and, as an example, the medical care a person receives – is not an area where a person would want to receive particularly surprising recommendations.
- *Limited content analysis* – Although this could be a problem, it was assumed that the contents of the system's items (healthcare institutions) were not something that was particularly difficult to

represent and compare by means of similarity functions. This could be done by analyzing the fields defined in the Domain Model (Figure 1), namely, the institution's list of specialties, agreements, and its location.

- *Inability to evaluate the quality of an item* – This problem seemed to be notably relevant, as solely recommending a new institution based on the similarity of its list of specialties, subsystems, and its location would not filter the institutions that were less liked by other users (i.e. had a lower average rating). Therefore, it was decided that the recommender system must take into consideration the ratings of the institutions in some form.

X



Registrar

Nome próprio *	Apelido *
Endereço de email *	
Palavra-passe *	Repetir palavra-passe *
Idade *	Sexo * ▼
Em que especialidades médicas está interessado? ▼	
Em que subsistemas de saúde está interessado? ▼	
☐ Aceito os Termos de Utilização e confirmo que li a Política de Privacidade.	
<div style="background-color: #e91e63; color: white; padding: 10px 20px; display: inline-block; border-radius: 5px;">REGISTAR</div>	

Já tem conta? Faça login

Figure 20: Register page with a short questionnaire for medical specialties and subsystems

Final Recommender System Architecture After this analysis, we arrived at the final architecture for the recommender system. This architecture (Figure 21) consists of three pieces that work together to perform the recommendations for each user. First, there is a Content-Based Recommender System that computes the similarity between Institutions in order to provide recommendations. Second, there is another Content-Based Recommender System that, alternatively, uses User Profile similarity to find the most similar uses and provide recommendations based on their preferred institutions. Finally, the recommendations that result from these two intermediate recommender systems are attributed weights

and combined in an Aggregator Recommender System based on the weights set. Lastly, a top N containing the best recommendations for the given user is computed and the final list is obtained.

Based on the analysis performed in the State of the Art (Chapter 2), we can say that this is a Hybrid Recommender System. More precisely, because the combination of the lists of recommendations from the intermediate systems is done by the use of weights, we can say that, according to Burke's classification [4], it belongs to the Weighted Recommender System subtype.

This architecture – together with the short questionnaire used to mitigate the new user problem (Figure 20) – accomplishes all the requirements that arose from the previous analysis. Additionally, it also takes advantage of the existing demographic information of the system's users (i.e. age and sex).

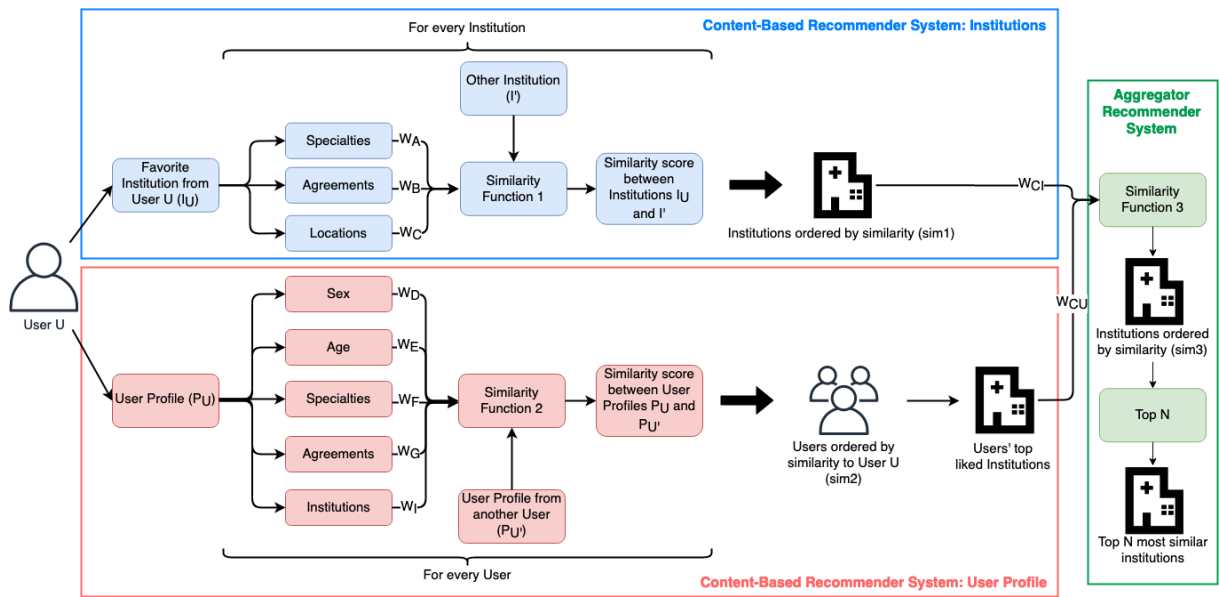


Figure 21: Recommender System Architecture

Types of Data Used The chosen architecture relies upon a specific set of data from the users. More specifically, it uses different types of explicit and implicit data.

In regards to the implicit data, we can enumerate the following:

- *Searches for a specific medical specialty* – When a user searches for a specific medical specialty, it is implicitly assumed that the user has an interest for that specialty.
- *Searches for a specific medical subsystem* – When a user searches for a specific subsystem, it is implicitly assumed that the user has an interest in that subsystem.
- *Searches for a specific location* – When a user searches for a specific location, it is implicitly assumed that the user has an interest in that location.
- *Searches for a specific institution* – When a user searches for a specific institution, it is implicitly assumed that the user has an interest in that institution.

- *Institution page visit* – When a user visits the page of an institution, it is implicitly assumed that the user has an interest for that institution.
- *Time spent in an institution's page* – When a user spends time on the page of an institution, it is implicitly assumed that the user has an interest for that institution. It is also assumed that the more time the user spends on the page, the more interest they have in that institution.

This data is used to compute the user profile of a specific user in the respective steps of Figure 21 (inputs of *Similarity Function 2*). The data that tells the system that a user is interested in a specific institution is also used to compute the favorite institution from the user (before *Similarity Function 1*), as well as to get the institutions that the users that resulted from the output of *Similarity Function 2* like the most.

In relation to the explicit data, we can point out the following:

- *Institution ratings submitted by the users* – When a user gives a good rating to an institution, this explicitly tells the system that the user is interested in that institution. It was defined that a good rating corresponds to a rating of at least 4 given by a user to an institution.

Similarly, taking a look at Figure 21, this data is used in the computation of the favorite institution from the user (before *Similarity Function 1*), as well as to get the institutions that the users that resulted from the output of *Similarity Function 2* like the most.

3.6.2 Intermediate Content-Based Recommender Systems

In the following sections, the functionality of the intermediate recommender systems – Institutions (3.6.2.1) and User Profile (3.6.2.2) – is detailed.

3.6.2.1 Content-Based Recommender System: Institutions

The first recommender system (Figure 21, blue) generates new recommendations by taking into account the user's item preference and comparing its contents with the other items in the system.

First, the system fetches the most highly rated institution from the user. The computation of the list of institutions the user is interested in considers both explicit and implicit data. In terms of explicit data, it takes into account the ratings the user has given in the past to the institutions, whereas, in terms of implicit data, it takes into account not only the searches the user has made – similarly to the list of specialties and agreements – but also other analytic information. However, the explicit data is given precedence when computing the list of institutions that the user is interested in. More precisely, the first step is to fetch the institutions that result from good past ratings of the user (i.e. at least 4 in 5). Only if there are less than 10 institutions that result from this process, does the implicit data play a role. Moreover, the institutions

that are obtained from implicit data can never appear in a higher position in the resulting list than those obtained by means of explicit data from the user. In this context, the calculation that uses implicit data considers past user searches of institutions as well as *institution page visits* and *time spent on*. *Institution page visits* is a record of how many times the user visits a certain institution's page. *Time spent on* is a record of how much time a user spends on a specific institution's page. Lastly, the system selects the institution with the highest score from the final list.

Second, this institution's fields are compared with every other institution in the system by means of three intermediate similarity functions. These similarity functions compute the similarity between two institutions' specialties, agreements, and locations.

Intermediate Similarity Function: Specialties The similarity function that measures the similarity between the specialties of two different institutions is given by Algorithm 1.

Algorithm 1 Calculate similarity between Institutions: Specialties

```

function jaccard_similarity( $set_1$ ,  $set_2$ )
    return  $\#(set_1 \cap set_2) / \#(set_1 \cup set_2)$ 
end function

```

Require: $specialties_{i_1}, specialties_{i_2} \in$ List of institutions' specialties
return $jaccard_similarity(specialties_{i_1}, specialties_{i_2})$

Intermediate Similarity Function: Agreements The similarity function that measures the similarity between the agreements of two different institutions is given by Algorithm 2.

Algorithm 2 Calculate similarity between Institutions: Agreements

```

function jaccard_similarity( $set_1$ ,  $set_2$ )
    return  $\#(set_1 \cap set_2) / \#(set_1 \cup set_2)$ 
end function

```

Require: $agreements_{i_1}, agreements_{i_2} \in$ List of institutions' agreements
return $jaccard_similarity(agreements_{i_1}, agreements_{i_2})$

Intermediate Similarity Function: Locations The similarity function that measures the similarity between the locations of two different institutions is given by Algorithm 3.

Algorithm 3 Calculate similarity between Institutions: Locations

```

function normalize(value, scale_min, scale_max)
    return (value - min_value)/(max_value - min_value)
end function
function distance_haversine(coords1, coords2)
     $\varphi_1 \leftarrow \text{coords}_1.\text{lat}$ 
     $\varphi_2 \leftarrow \text{coords}_2.\text{lat}$ 
     $\Delta\varphi \leftarrow |\varphi_1 - \varphi_2|$ 
     $\Delta\lambda \leftarrow |\text{coords}_1.\text{lng} - \text{coords}_2.\text{lng}|$ 
     $d \leftarrow 2R \arcsin \sqrt{\sin^2 \frac{\Delta\varphi}{2} + \cos \varphi_1 \cos \varphi_2 \sin^2 \frac{\Delta\lambda}{2}}$ 
    return d
end function
Require: coords1, coords2 = (lat, lng); lat, lng ∈ [0°, 90°]
    dist ← distance_haversine(coordsi1, coordsi2)
    return normalize(dist, 0, 578)

```

Then, these three intermediate results are used by an aggregator similarity function (Algorithm 4). This function attributes a weight to each of the intermediate results and outputs the final score by multiplying – and then summing – the intermediate result by its weight.

The normalization of the distance between the scale of 0 to 578 was decided based on the fact that 578km is approximately the biggest distance between two points in Continental Portugal.

Regarding the weights, it was decided that the specialties and the agreements should have a slightly higher contribution towards the final result than the location, under the assumption that two institutions that had a different set of specialties and agreements would be less similar to one another, than two other institutions whose specialties and agreements were identical but were farther away from each other.

Algorithm 4 Calculate similarity between Institutions

```

Require:  $\text{sim}_{\text{specialties}}, \text{sim}_{\text{agreements}}, \text{sim}_{\text{coordinates}} \in [0, 1]$ 
     $w_{\text{specialties}} \leftarrow 0.4$ 
     $w_{\text{agreements}} \leftarrow 0.4$ 
     $w_{\text{locations}} \leftarrow 0.2$ 
    return  $w_{\text{specialties}} * \text{sim}_{\text{specialties}} + w_{\text{agreements}} * \text{sim}_{\text{agreements}} + w_{\text{locations}} * \text{sim}_{\text{locations}}$ 

```

After reproducing the process above for every item in the system, the institutions can be then ordered by how similar their contents are to the input institution. Finally, the N-most similar institutions (e.g. top 10) are selected and outputted to the next recommender system, the aggregator (Section 3.6.3).

3.6.2.2 Content-Based Recommender System: User Profiles

User Profile The second recommender system (Figure 21, red) generates recommendations by taking into account the User Profile of the users. The User Profile contains various information regarding the user,

namely:

- *Sex* – The sex of the user;
- *Age* – The age of the user;
- *Specialties* – The list of specialties the user has searched;
- *Agreements* – The list of agreements the user has searched;
- *Institutions* – The list of institutions the user is interested in.

The computation of the list of institutions the user is interested in is similar to what was explained in Section 3.6.2.1.

Similarity between User Profiles Therefore, the first step of this recommender system is to compare the user profile of the user it wants to give recommendations to with the user profile of all the other users in the system. It does this by means of several intermediate similarity functions that measure the similarity between each of the fields of the user profile (i.e. sex, age, specialties, agreements, and institutions).

Intermediate Similarity Function: Sex The similarity function that measures the similarity between the sexes of two different users is given by Algorithm 5.

Algorithm 5 Calculate similarity between User Profile: Sex

Require: $s_1, s_2 \in \{ 'M', 'F' \}$

if $s_1 = 'M'$ **then**

$s_1 \leftarrow 1$

else

$s_1 \leftarrow 0$

end if

if $s_2 = 'M'$ **then**

$s_2 \leftarrow 1$

else

$s_2 \leftarrow 0$

end if

return $|s_1 - s_2|$

Intermediate Similarity Function: Age The similarity function that measures the similarity between the ages of two different users is given by Algorithm 6.

Algorithm 6 Calculate similarity between User Profile: Age

```

function normalize(value, scale_min, scale_max)
  return  $(value - min\_value) / (max\_value - min\_value)$ 
end function

```

Require: $a_1, a_2 \in [18, 118]$

```

 $age\_dif \leftarrow |a_1 - a_2|$ 
return  $normalize(age\_dif, 0, 100)$ 

```

Intermediate Similarity Function: Specialties The similarity function that measures the similarity between the specialties of two different institutions is given by Algorithm 1.

Intermediate Similarity Function: Agreements The similarity function that measures the similarity between the agreements of two different institutions is given by Algorithm 2.

Intermediate Similarity Function: Institutions The similarity function that measures the similarity between two different institutions is given by Algorithm 7. The same values as those in Algorithm 4 were utilized for the weighing of the specialties, agreements, and locations.

Algorithm 7 Calculate similarity between User Profiles: Institutions

```

function normalize(value, scale_min, scale_max)
    return (value – min_value)/(max_value – min_value)
end function
function jaccard_similarity(set1, set2)
    return #(set1 ∩ set2)/#(set1 ∪ set2)
end function
function distance_haversine(coords1, coords2)
    φ1 ← coords1.latitude
    φ2 ← coords2.latitude
    Δφ ← |φ1 – φ2|
    Δλ ← |coords1.longitude – coords2.longitude|
    d ← 2R arcsin √[sin2  $\frac{\Delta\phi}{2}$  + cos φ1 cos φ2 sin2  $\frac{\Delta\lambda}{2}$ ]
    return d
end function
function institutions_specialties_similarity(specialtiesi1, specialtiesi2)
    return jaccard_similarity(specialtiesi1, specialtiesi2)
end function
function institutions_agreements_similarity(agreementsi1, agreementsi2)
    return jaccard_similarity(agreementsi1, agreementsi2)
end function
function institutions_coordinates_similarity(coordsi1, coordsi2)
    dist ← distance_haversine(coordsi1, coordsi2)
    return normalize(dist, 0, 578)
end function
Require: i1, i2 ∈ List of institutions
    simspecialties ← institutions_specialties_similarity(i1.specialties, i2.specialties)
    simagreements ← institutions_agreements_similarity(i1.agreements, i2.agreements)
    simcoordinates ← institutions_coordinates_similarity(i1.coordinates, i2.coordinates)
    wS ← 0.4
    wA ← 0.4
    wC ← 0.2
    return wS * simspecialties + wA * simagreements + wC * simcoordinates

```

Aggregate Similarity Function Finally, the similarity scores from the previous intermediate similarity functions that calculate the similarity of each individual field within the User Profiles are combined (Algorithm 8). In order to do this, a weight is set for each of these intermediate similarity scores. Then, each of the intermediate similarity scores is multiplied by its corresponding weight. Finally, these results are summed and the final similarity score between the user profile of user u and the user profile of user u' is obtained. Regarding the weights, it was decided that the institutions and specialties would have the highest weight, followed by the sex of the user and, last, by the age and agreements.

Algorithm 8 Calculate similarity between User Profiles

Require: $sim_{sex}, sim_{age}, sim_{specialties}, sim_{agreements}, sim_{institutions} \in [0, 1]$
 $w_{sex} \leftarrow 0.2$
 $w_{age} \leftarrow 0.1$
 $w_{specialties} \leftarrow 0.3$
 $w_{agreements} \leftarrow 0.1$
 $w_{institutions} \leftarrow 0.3$
return $w_{sex} * sim_{sex} + w_{age} * sim_{age} + w_{specialties} * sim_{specialties} + w_{agreements} * sim_{agreements} + w_{institutions} * sim_{institutions}$

This similarity score can be translated to how similar the user profiles of user u and of the user it is being compared against are. Therefore, this process is done for every user in the system. After this, it is then possible to obtain the list of users most similar to the user u by ordering the other users by the similarity score that resulted.

Finally, for each of the most similar users (e.g. top 10), it is possible to obtain the list of institutions they are interested in. From this list, the system collects the most common institutions and uses them for the recommendations it outputs to the next recommender system, the aggregator (Section 3.6.3).

3.6.3 Aggregator Recommender System

The aggregator recommender system (Figure 21, green) is responsible for combining the results from the previous two recommender systems (Sections 3.6.2.1 and 3.6.2.2). To that end, two weights, W_{CI} and W_{CU} (Figure 21), are set and used by a similarity function. This function takes the institutions recommended from the previous systems and generates a new similarity score by multiplying the previous results by the weight set for each of the two components, W_{CI} and W_{CU} . This process is demonstrated by Algorithm 9. Regarding the weights, it was decided that the result from the recommender system that compares the user profiles should have a slightly lower importance towards the final similarity result than the recommender system that compares the institutions to the user's favorite institution.

Algorithm 9 Calculate similarity between Institutions (aggregator)

Require: $institutions_{CI}, institutions_{CU} = [(I, sim_I)], I \in \text{List of Institutions}, sim_I \in [0, 1]$
 $w_{CI} \leftarrow 0.6$
 $w_{CU} \leftarrow 0.4$
 $recommendations \leftarrow []$
for all I in $institutions_{CI}$ **do**
 $new_sim_I \leftarrow sim_I * w_{CI}$
 Insert (I, new_sim_I) into $recommendations$
end for
for all I in $institutions_{CU}$ **do**
 $new_sim_I \leftarrow sim_I * w_{CU}$
 Insert (I, new_sim_I) into $recommendations$
end for
 $sorted_recs \leftarrow \text{Sorted } recommendations \text{ by } new_sim_I \text{ desc}$
return $sorted_recs$

Upon completion of this process, the institutions are reordered according to the new similarity score that was calculated. Afterward, the top N institutions are extracted from this list and the final list of institution recommendations is obtained.

3.7 Usage of Learned Data

Naturally, after the generation of the recommendations by the Recommender System (Section 3.6), they have to be presented to the user. This is achieved in two different locations of the user interface.

The first one – presented in Figure 22 – is the homepage. Every time a user logs in onto the system, they are presented with the recommendations that have been generated for them, ahead of time. This was achieved by augmenting the user interface of the homepage that was initially displayed in Section 3.3.2.1 with a new section where this list of institutions appears.

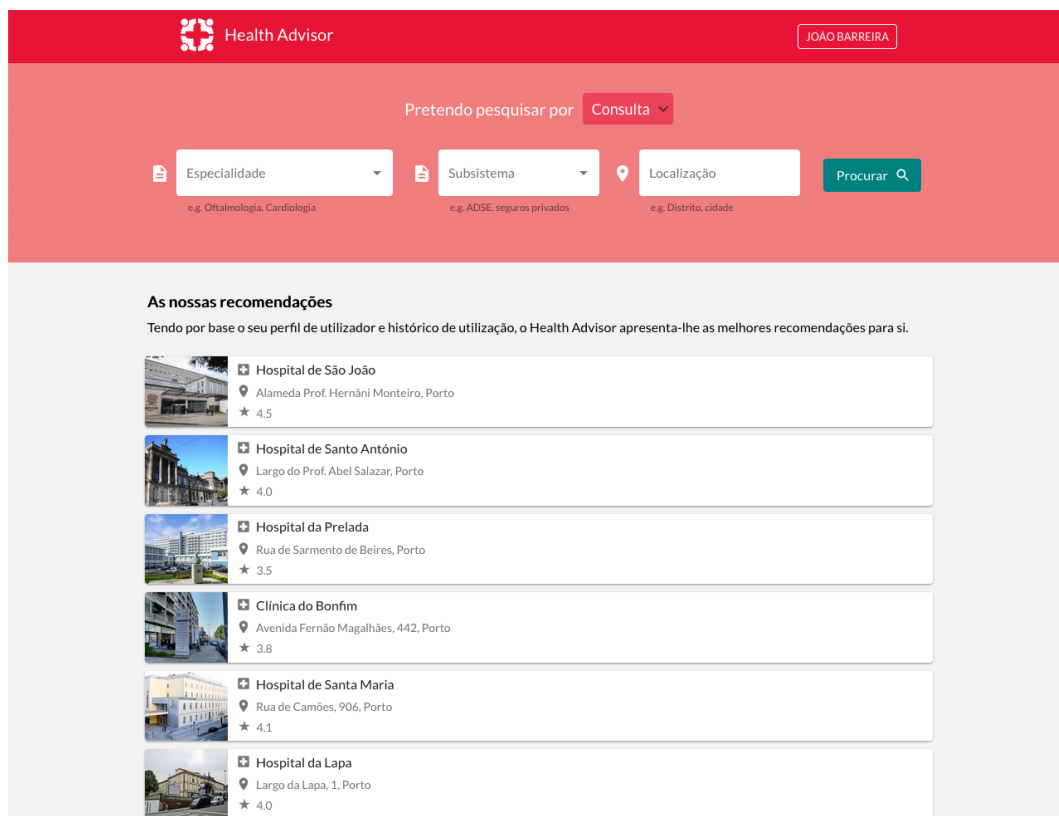


Figure 22: Recommendations presented to the user on the homepage

The second one – presented in Figure 23 – is the search results page. Every time a user performs a new search, a list of institutions that match the corresponding query is displayed. However, when the system has generated recommendations for that specific user, the user interface changes slightly. More specifically, if, amongst the list of institutions that result from the search, an institution that also appears in the recommendations for that user is included, that information is displayed. This is done by means of a new "Recommended by HealthAdvisor" bubble that appears for the matching institutions. Therefore, the resulting page corresponds to a slightly modified version of the user interface initially displayed in Section 3.3.2.4.

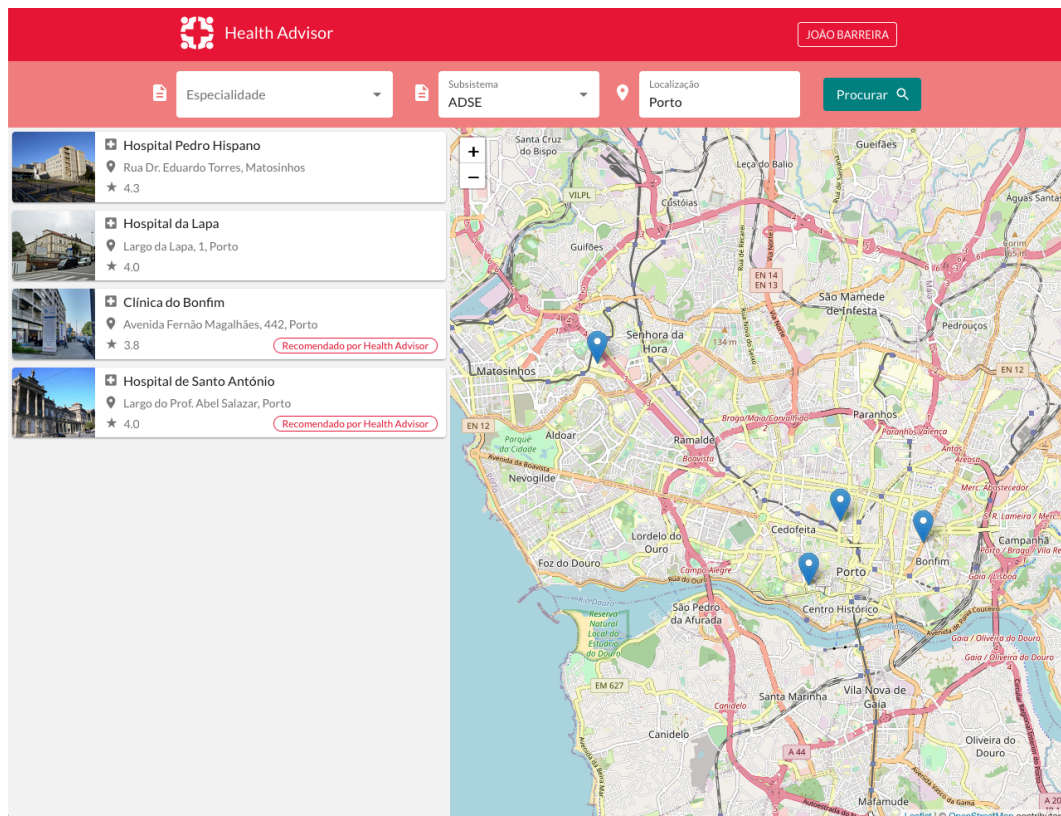


Figure 23: Recommendations presented to the user on the search results page

3.8 Flow of Data and Generation of New Recommendations

In summary, the data from the user that is used to generate new recommendations flows through the system in a cyclical manner (Figure 24). By using the system, the user generates new information: new searches provide information about specialties and subsystems the user may be interested in; new reviews provide information about how high a user rates a specific institution; institution page visits tells the system that a user may also be interested in an institution by the number of visits or the amount of time they have spent in its page. This information is collected in real-time by the system and sent to the backend via the Analytics API where it is persisted in the database (Section 3.4). Then, every 24 hours (by default), this information is fetched by HealthAdvisor Recommendations and is used by its Dataset Builder to update the datasets (Section 3.5.2). After this, these new datasets are used by the Recommender System to generate new recommendations (Section 3.5.3). Finally, the new recommendations are shown to the user every time they visit the homepage or perform a new search.

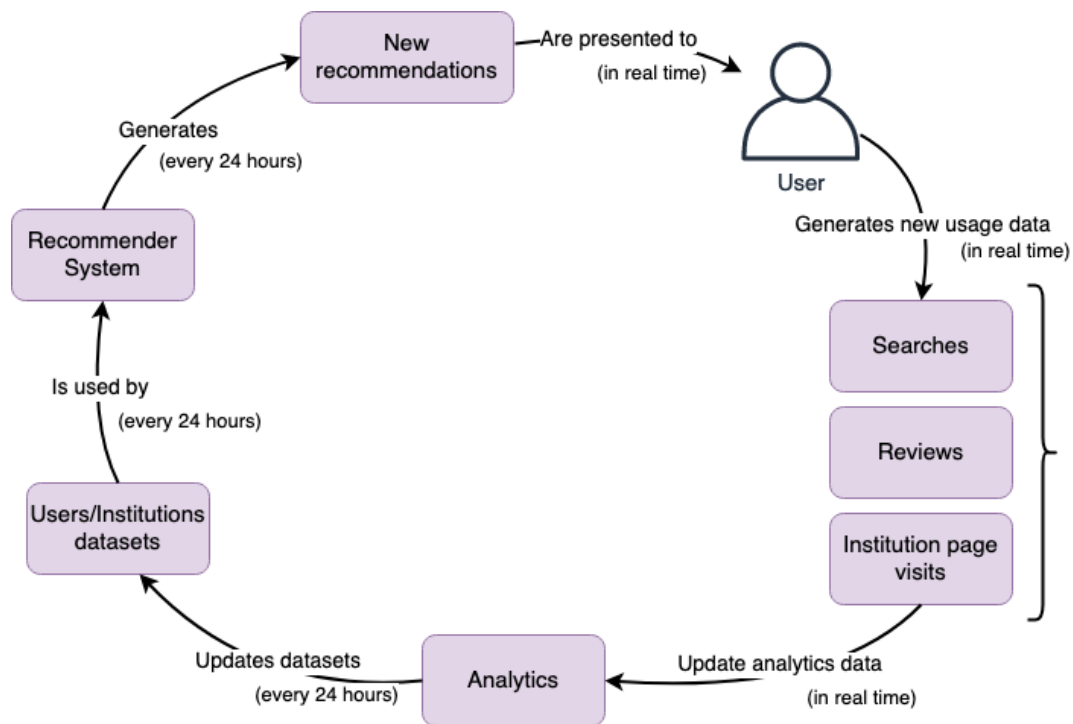


Figure 24: Cycle of data

Therefore, we can see that the system is dynamic. Every 24 hours, new recommendations are generated and presented to the user, altering and enriching their experience using the system. We can also say that the more a user interacts with the system, the more information will be available to the Recommender System to make better recommendations. This is also true for all the other users in the system. Indeed, the more active users there are in the system, the more information will be available to be used by the Recommender System to compare the current user to the other user profiles. Additionally, the more reviews, the more information will be available for the user to decide upon.

Conclusions and Future Work

In this section, we take a step back to analyze the work done in the context of this thesis, draw some conclusions, and present ideas for future extensions and improvements.

4.1 Conclusions

In terms of conclusions, we can say that the work that was planned for this thesis – laid out, initially, in the Objectives section (1.2) – was successfully completed.

In this work, we have accomplished the goal of providing a platform for the search, review, and recommendation of healthcare institutions. In summary, in the result of this thesis we have covered all the eight use cases enumerated in Section 3.1.2. Namely, the final product presented in this thesis delivers the functionality necessary for providing a user the ability to register on the system, log in, perform multiple types of healthcare institution searches, see an institution's details, submit a review of an institution, read the reviews of other users, and receive recommendations for unknown institutions.

We've done this by presenting a solution with an expansive development from a technical standpoint, as it not only encompasses a recommender system, but also a frontend application as well as the backend platform to support both of these systems.

In the frontend (Section 3.3), we have delivered a solution that is modern-looking, well tested, and easily extendable.

In the backend (Section 3.4), we have developed a system that provides several APIs that – though broad in scope – are well-segmented and modularized. More precisely, we have built APIs for user authentication, search, ratings, users, and institutions. Additionally, we have developed a separate API for analytics that receives the users' usage data from the frontend and exposes user profile and institution

information to the recommendations subsystem. We have also invested the time and the effort to develop several tests for the APIs that were constructed, as a means to increase the quality and the reliability of the developed system.

In the recommendations component (Section 3.5), we have presented a solution that not only comprises a recommender system, but also a dataset builder that constructs the datasets needed and an API that provides recommendations of healthcare institutions for a given user. Additionally, we have added the functionality to run the dataset builder and the generation of new recommendations periodically (every 24 hours, by default) through the utilization of specialized utilities (i.e. *cronjobs*). Moreover, in the development of the recommender system, we took into consideration the study performed on the State of the Art section (Section 2) and analyzed the final recommender, focusing on the limitations that had been enumerated for the type of system chosen. For each of these limitations, we either provided a direct solution to them (e.g. questionnaire to solve the "new user" problem) or argued that, given the context of application, they were acceptable (e.g. lack of serendipity for the generated recommendations).

4.2 Future Work

In terms of future work, an improvement to the work done in this thesis could perhaps be the utilization of a more complex a robust mechanism to optimize the choice of the weights that were used in the similarity functions of the recommender system. A shortcoming of the current version is that the definition of these values was made manually, in an attempt to give more importance to a given input instead of another. This method is perhaps not optimal and an effort to improve the selection of these values would most likely improve the quality of the recommender system and of its recommendations.

One important improvement to the system would be to start getting users utilizing the system as a way to gather feedback to improve the existing functionalities, as well as obtain user data. This way, it would be possible to test the current recommender system and its recommendations. With this information, it would be possible to treat the current recommender system as a baseline and try to improve it. This could be done by means of direct modifications to the developed system or by adding a new recommender and aggregating the recommendations from each one. Either way, it is critical that the system be tested against real-world data, so as to assess its performance.

A future line of work could be the development of a system that verifies the cases such as that of a user that attributes a high 1 to 5 rating to an institution while writing a particularly negative comment on that same review (or vice-versa). Indeed, it could perhaps be interesting to include sentiment analysis capabilities in the system. This way, it would be possible to, for example, evaluate the polarity of the reviews' comments and, additionally, calculate the total polarity of the comments of a given institution. This could then be used as another differentiating factor between institutions during the recommendation process. In fact, this approach was initiated – and a script that parses a text and calculates its polarity was developed

(see Appendix A) – but it was not finished and included in the final system.

Another interesting extension could be the inclusion of a mechanism that checks for the existence of references to any medical specialty or subsystem in the comment of a user's review which may signal the user's interest in them. This information could then be used to improve the recommendation process. A way of doing this could be in conjunction with the calculation of the comments' polarity enunciated above. By using both of these functionalities, the system could, first, gain knowledge of a medical specialty that the user may be interested in – apart from the current functionality of implicitly obtaining this information through their searches – and, second, recommend an institution by looking for positive references to that medical specialty in the comments of the reviews made to that institution.

Bibliography

- [1] C. C. Aggarwal. *Recommender Systems. The Textbook*. First Edition. Springer International Publishing, 2016. isbn: 978-3-319-29659-3. doi: [10.1007/978-3-319-29659-3](https://doi.org/10.1007/978-3-319-29659-3).
- [2] F. Alyari and N. J. Navimipour. "Recommender systems: A systematic review of the state of the art literature and suggestions for future research." In: *Kybernetes* (2018). doi: [10.1108/K-06-2017-0196](https://doi.org/10.1108/K-06-2017-0196).
- [3] J. K. Ayeh, N. Au, and R. Law. "'Do We Believe in TripAdvisor?' Examining Credibility Perceptions and Online Travelers' Attitude toward Using User-Generated Content." In: *Journal of Travel Research* 52.4 (2013), pp. 437–452. doi: [10.1177/0047287512475217](https://doi.org/10.1177/0047287512475217).
- [4] R. Burke. "Hybrid Recommender Systems: Survey and Experiments." In: *User Modeling and User-Adapted Interaction* 12.4 (2002). issn: 1573-1391. doi: [10.1023/A:1021240730564](https://doi.org/10.1023/A:1021240730564).
- [5] J. A. Chevalier and D. Mayzlin. "The Effect of Word of Mouth on Sales: Online Book Reviews." In: *Journal of Marketing Research* 43.3 (2006), pp. 345–354. doi: [10.1509/jmkr.43.3.345](https://doi.org/10.1509/jmkr.43.3.345).
- [6] S. David and T. J. Pinch. "Six degrees of reputation: The use and abuse of online review and recommendation systems." In: *SSRN Electronic Journal* 857505 (2005).
- [7] W. Duan, B. Gu, and A. B. Whinston. "Do online reviews matter? — An empirical investigation of panel data." In: *Decision Support Systems* 45.4 (2008). Information Technology and Systems in the Internet-Era, pp. 1007–1016. issn: 0167-9236. doi: <https://doi.org/10.1016/j.dss.2008.04.001>.
- [8] M. Emmert, U. Sander, and F. Pisch. "Eight Questions About Physician-Rating Websites: A Systematic Review." In: *J Med Internet Res* 15.2 (Feb. 2013), e24. issn: 1438-8871. doi: [10.2196/jmir.2360](https://doi.org/10.2196/jmir.2360).
- [9] M. Emmert, L. Sauter, L. Jablonski, U. Sander, and F. Taheri-Zadeh. "Do Physicians Respond to Web-Based Patient Ratings? An Analysis of Physicians' Responses to More Than One Million Web-Based Ratings Over a Six-Year Period." In: *J Med Internet Res* 19.7 (July 2017), e275. issn: 1438-8871. doi: [10.2196/jmir.7538](https://doi.org/10.2196/jmir.7538).

-
- [10] C. A. Gomez-Urbe and N. Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation." In: *ACM Transactions on Management Information Systems* 6 (4 2016). issn: 2158-656X. doi: [10.1145/2869770](https://doi.org/10.1145/2869770).
 - [11] U. Gretzel, K. H. Yoo, and M. Purifoy. "Online travel review study: Role and impact of online travel reviews." In: *Texas A & M University. Department of Recreation, Park and Tourism Sciences* (2007).
 - [12] M. Hardey. "Consuming Professions: User-review websites and health services." In: *Journal of Consumer Culture* 10.1 (2010), pp. 129–149. doi: [10.1177/1469540509355023](https://doi.org/10.1177/1469540509355023).
 - [13] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, and M. Salehi. "Evaluating Collaborative Filtering Recommender Algorithms: A Survey." In: *IEEE* 6 (2018). doi: [10.1109/ACCESS.2018.2883742](https://doi.org/10.1109/ACCESS.2018.2883742).
 - [14] T. Lagu, S. L. Goff, N. S. Hannon, A. Shatz, and P. K. Lindenauer. "A Mixed-Methods Analysis of Patient Reviews of Hospital Care in England: Implications for Public Reporting of Health Care Quality Data in the United States." In: *The Joint Commission Journal on Quality and Patient Safety* 39.1 (2013), 7–AP4. issn: 1553-7250. doi: [https://doi.org/10.1016/S1553-7250\(13\)39003-5](https://doi.org/10.1016/S1553-7250(13)39003-5). url: <https://www.sciencedirect.com/science/article/pii/S1553725013390035>.
 - [15] S. W. Litvin, R. E. Goldsmith, and B. Pan. "Electronic word-of-mouth in hospitality and tourism management." In: *Tourism Management* 29.3 (2008), pp. 458–468. issn: 0261-5177. doi: <https://doi.org/10.1016/j.tourman.2007.05.011>. url: <https://www.sciencedirect.com/science/article/pii/S0261517707001343>.
 - [16] P. O'Connor. "User-Generated Content and Travel: A Case Study on Tripadvisor.Com." In: *Information and Communication Technologies in Tourism 2008*. Springer Vienna, 2008, pp. 47–58. isbn: 978-3-211-77280-5.
 - [17] J. Preece. "Sociability and usability: Twenty years of chatting online." In: *Behavior and Information Technology Journal* 20.5 (2001), pp. 347–356.
 - [18] B. Ranard, R. Werner, T. Antanavicius, H. Schwartz, R. Smith, Z. Meisel, D. Asch, L. Ungar, and R. Merchant. "Yelp Reviews Of Hospital Care Can Supplement And Inform Traditional Surveys Of The Patient Experience Of Care." In: *Health Affairs* 35 (Apr. 2016), pp. 697–705. doi: [10.1377/hlthaff.2015.1030](https://doi.org/10.1377/hlthaff.2015.1030).
 - [19] F. Ricci, L. Rokach, and B. Shapira. *Recommender Systems Handbook*. Second Edition. Springer, 2015. isbn: 978-1-4899-7636-9. doi: [10.1007/978-1-4899-7637-6](https://doi.org/10.1007/978-1-4899-7637-6).
 - [20] R. Rozenblum and D. Bates. "Patient-centred healthcare, social media and the internet: The perfect storm?" In: *BMJ quality & safety* 22 (Feb. 2013). doi: [10.1136/bmjqs-2012-001744](https://doi.org/10.1136/bmjqs-2012-001744).

-
- [21] S. Senecal and J. Nantel. "The influence of online product recommendations on consumers' online choices." In: *Journal of Retailing* 80.2 (2004), pp. 159–169. issn: 0022-4359. doi: <https://doi.org/10.1016/j.jretai.2004.04.001>. url: <https://www.sciencedirect.com/science/article/pii/S0022435904000193>.
- [22] X. Su and T. M. Khoshgoftaar. "A Survey of Collaborative Filtering Techniques." In: *Advances in Artificial Intelligence* 2009 (2009). issn: 1687-7470. doi: [10.1155/2009/421425](https://doi.org/10.1155/2009/421425).
- [23] P. B. Thorat, R. M. Goudar, and S. Barve. "Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System." In: *International Journal of Computer Applications* 110 (2015). doi: [10.5120/19308-0760](https://doi.org/10.5120/19308-0760).
- [24] B. Wellman. *Networks in the global village: Life in contemporary communities*. Routledge, 2018.
- [25] Q. Ye, R. Law, B. Gu, and W. Chen. "The influence of user-generated content on traveler behavior: An empirical investigation on the effects of e-word-of-mouth to hotel online bookings." In: *Computers in Human Behavior* 27.2 (2011), pp. 634–639. issn: 0747-5632. doi: [10.1016/j.chb.2010.04.014](https://doi.org/10.1016/j.chb.2010.04.014).
- [26] Y. Zhang, J. Callan, and T. Minka. "Novelty and Redundancy Detection in Adaptive Filtering." In: *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '02. New York, NY, USA: Association for Computing Machinery, 2002, pp. 81–88. isbn: 1581135610. doi: [10.1145/564376.564393](https://doi.org/10.1145/564376.564393).



Appendix 1 – Script that calculates the polarity of a text

```
import argparse
from polyglot.text import Text

STOPWORDS_FILE_PATH = "stopwords_portuguese.txt"

def read_file(path_to_file):
    with open(path_to_file, 'r') as file:
        file_content = file.read().replace('\n', ' ')
    return file_content

def get_stopwords_from_file(path_to_stopwords_file):
    with open(path_to_stopwords_file, "r") as stopwords_file:
        stopwords = stopwords_file.read().splitlines()
    return stopwords

def remove_stopwords(text, path_to_stopwords_file):
    stopwords = get_stopwords_from_file(path_to_stopwords_file)
    words_in_text = text.split(' ')
    words_in_text_without_stopwords = [w for w in words_in_text if w not in stopwords]
    return " ".join(words_in_text_without_stopwords)

def print_words_polarity(text):
    polyglot_text = Text(text)
    print("{:<16}{}".format("Word", "Polarity") + "\n" + "-" * 30)
```

```
for w in polyglot_text.words:
    if w.polarity != 0:
        print("{:<16}{:>2}".format(w, w.polarity))

def calculate_text_polarity(text):
    polyglot_text = Text(text)
    return polyglot_text.polarity

if __name__ == '__main__':
    # Argument parser
    parser = argparse.ArgumentParser(description="Calculate the polarity of a file's text")
    parser.add_argument("-f", "--file", type=str, required=True, help="File containing the text
        ↪ ")
    parser.add_argument("-d", "--details", default=False, action="store_true", help="Print the
        ↪ polarity of all the words in the input text [default: %(default)s]")
    args = parser.parse_args()
    # Read and parse text
    text = read_file(args.file)
    text = remove_stopwords(text, STOPWORDS_FILE_PATH)
    # Print the polarity of all words in text if 'details' argument was received
    if args.details:
        print_words_polarity(text)
    # Calculate and print total polarity of text
    polarity = calculate_text_polarity(text)
    print(polarity)
```