INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Explicação Automática de Conceitos

RAFAEL REIS FONTES
Outubro de 2023

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Explicação Automática de Conceitos

Rafael Fontes

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Software Engineering

Supervisor: Ricardo Almeida, PhD
Co-Supervisor: Nuno Escudeiro, PhD

Evaluation Committee:
President:

Members:

Porto, October 12, 2023

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore the work presented in this document is original and authored by me, having not previously been used for any other end.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, October 12, 2023

Abstract

Technical terms are an essential part of all technical and scientific documentation, whether directed to education, to research or to labour. In education settings, there is a need to provide clear definitions of terms; using a glossary explaining the meaning of each technical term is paramount to enter a new study field. It is exactly at this last point that equity and inclusion issues originate. If we provide these definitions, or explanations using spoken languages in writing we leave apart all those who cannot fluently read them. Deaf people cannot read fluently. Sign languages and spoken languages are distinct languages. One cannot expect that a deaf person used to communicate via sign language can understand spoken language. When forcing deaf students to study via written spoken languages, we are putting them at a clear disadvantage and compromising equity. There is a need for a tool that can introduce and explain to deaf students technical or scientific concepts from specific areas of knowledge in sign language.

This project aims to assist deaf students gaining access to education in their first language thus enhancing their learning experience by developing an online platform providing explanations of technical terms in sign language.

We expect that Text Mining and Information Extraction techniques can help generate explanations of concepts that do not exist in the sign language lexicon that can then be translated and played in sign language. The purpose of this project is to develop the information extraction component that will generate the explanations in spoken language. These explanations will be translated to sign language using the VirtualSign Application Programming Interface (API).

The developed solution involves an API designed to produce explanations for a word or concept, either by scraping online dictionaries or exploring text summarization. The API provides this information to a Web Application exposed to the users, and is responsible for receiving their input and displaying the information, along with the sign language translation using the VirtualSign avatar plugin.

The solution was tested and surveyed by end-users being deemed a valuable step toward inclusive education.

Keywords: Sign language, Automatic concept explanation, Web scrapping, Text summarization, API, Web application

Resumo

Disponibilizar uma explicação precisa para um conceito é crucial no que toca a linguagens com um vocabulário reduzido, tal como a língua gestual, pois esta mesma não é capaz de representar todas as palavras representadas em línguas orais e textuais, como a Língua Portuguesa.

Pessoas que utilizam língua gestual diariamente, quando confrontados com palavras ou conceitos específicos, como por exemplo "nanotecnologia", recorrem a ferramentas criadas para línguas verbais e textuais, como dicionários, porque ferramentas direcionadas a língua gestual são escassas ou pouco apropriadas. Estas soluções, como intérpretes de língua gestual, não possuem rigor nas traduções ou são impráticas.

Para abordar este problema, este projeto foi continuado para avaliar a hipótese de utilizar técnicas de *Text Mining* e *Information Extraction* para gerar explicações dadas palavras ou conceitos que não existem, ou não são possíveis de traduzir diretamente na língua gestual.

A solução desenvolvida involve uma *Application Programming Interface* (API) com o objetivo de produzir explicações para palavras e conceitos, via *Scraping* de dicionários online ou sumarização de texto. A API disponibiliza esta informação a uma *Web Application* exposta ao público, e é responsável por receber os *inputs* do utilizador e disponibilizar as explicações, junto com a tradução em língua gestual recorrendo a um Avatar.

A solução foi testada e, posteriormente, sujeita a feedback das partes interessadas do projeto, em relação à sua utilização e foi possível afirmar que a solução é capaz de gerar explicações dada uma palavra ou conceito. Com base no feedback, foi possível concluir também que a utilização da aplicação é satisfatória, no entanto, há espaço para melhoria no futuro.

Contents

List of Figures **xiii**

List of Tables **xv**

List of Source Code **xvii**

1 Introduction **1**
1.1 Problem . . . 1
1.2 Context . . . 1
1.3 Objectives . . . 1
1.4 Structure . . . 2

2 State of the Art **3**
2.1 Sign Language . . . 3
2.2 Information Extraction . . . 3
2.3 Text Mining . . . 4
2.4 Online Dictionaries . . . 5
2.5 Readability Metrics . . . 5
2.6 Technology . . . 6
2.6.1 Libraries and Frameworks . . . 7
Natural Language Toolkit . . . 7
Scrapy . . . 7
Beautiful Soup 4 . . . 7
Flask . . . 8
React . . . 9
Analysis . . . 9
Alternatives . . . 9
2.6.2 Databases . . . 10
MySQL . . . 10
Apache Cassandra . . . 10
Comparison . . . 10
2.6.3 Others . . . 10
Docker . . . 10
Geolocation DB . . . 11
PM2 . . . 11
NGINX . . . 11
Analysis . . . 12
Alternatives . . . 12
2.7 Related Work . . . 12

3 Value Analysis **15**

3.1 New Concept Development Model . . . 15
3.2 Value . . . 16
3.3 Value Proposition . . . 16
3.4 Business Model Canvas . . . 17
3.5 Analytic Hierarchy Process . . . 19
3.5.1 Stage 1 - Defining the goal . . . 19
3.5.2 Stage 2 - Defining the Hierarchy Tree . . . 19
3.5.3 Stage 3 - Pair wise comparison . . . 20
3.5.4 Stage 4 - Relative priority of each criteria . . . 21
3.5.5 Stage 4 - Consistency evaluation of relative priorities . . . 21
3.5.6 Stage 5 - Constructing the parity comparison matrix per criteria . . . 22
3.5.7 Stage 6 - Obtain the composite property for alternatives . . . 24
3.5.8 Stage 7 - Choosing alternative . . . 24

4 Analysis and Design **25**
4.1 Requirements Engineering . . . 25
4.1.1 Functional Requirements . . . 25
UC1: Obtain Explanations . . . 26
UC2: Obtain Summarization . . . 26
UC3: Display Sign Language Translation . . . 26
UC4: Change Language . . . 26
UC5: Rate Explanation . . . 27
4.1.2 Non-functional Requirements . . . 27
Functionality . . . 27
Usability . . . 27
Reliability . . . 27
Performance . . . 27
Supportability . . . 28
Plus (+) . . . 28
4.2 Logical View . . . 28
4.3 Deployment View . . . 29
4.4 Process View . . . 29
4.5 Functional Requirements Design . . . 33
4.5.1 UC1: Obtain Explanations . . . 33
4.5.2 UC2: Obtain Summarization . . . 34
4.5.3 UC3: Display Sign Language Translation . . . 35
4.5.4 UC4: Change Language . . . 36
4.5.5 UC5: Rate Explanation . . . 37

5 Implementation **39**
5.1 Major Changes . . . 39
5.1.1 Support Other Languages . . . 39
5.1.2 Rating Explanations . . . 39
5.1.3 Added Database . . . 39
5.1.4 Help Page . . . 40
5.2 EAC API - Backend . . . 40
5.2.1 Improvement Changes . . . 40
Find explanations functionality . . . 40
Summarization functionality . . . 42

5.2.2 New functionalities Changes . . . 43
Support Other Languages . . . 43
Rating explanations . . . 45
Added Database . . . 45
5.3 EAC APP - Frontend . . . 47
5.3.1 Improvement Changes . . . 47
Avatar . . . 47
Language support . . . 49
Board with Context . . . 49
5.3.2 New functionalities Changes . . . 50
Translate Input Word . . . 50
Rating explanations . . . 50
Help page . . . 51
5.4 EAC DB - Database . . . 52
5.5 Deployment . . . 53
5.5.1 EAC API - Backend . . . 54
5.5.2 EAC APP - Frontend . . . 54
5.5.3 Cassandra Database . . . 54
5.5.4 Exposing the solution . . . 55

6 Evaluation **57**
6.1 Software tests . . . 57
6.2 Hypothesis . . . 57
6.3 Methodology . . . 57
6.4 Survey Results Analysis . . . 58

7 Conclusion **61**
7.1 Developments . . . 61
7.2 Achievements . . . 62
7.3 Future Work . . . 62
7.4 Final Appreciation . . . 63

Bibliography **65**

A Solution Survey **69**

List of Figures

3.1 Value Proposition Canvas 16
3.2 Business Model Canvas 18
3.3 Hierarchical Decision Tree 19
3.4 Hierarchical decision tree with criteria parity comparison 24

4.1 Use Case Diagram 26
4.2 Component Diagram 28
4.3 Deployment Diagram 29
4.4 Activity Diagram - Possible Solution 30
4.5 Activity Diagram - 1. User input 31
4.6 Activity Diagram - 2. Find Explanations 31
4.7 Activity Diagram - 3. Provide Summarization 32
4.8 Activity Diagram - 4. Provide Explanations with Scores 32
4.9 Activity Diagram - 5. Translate Explanation 33
4.10 UC1 34
4.11 UC2 35
4.12 UC3 36
4.13 UC4 36
4.14 UC5 37

5.1 Avatar Plugin in Display 48
5.2 Context Board Component Example 49
5.3 Translate Input Button 50
5.4 Help Page 52
5.5 EAC APP in PM2's Running Applications List 54

List of Tables

2.1 Parsers libraries (Beautiful Soup 4). 8

3.1 Perceived Value 16
3.2 Fundamental scale 20
3.3 Comparison Criteria 20
3.4 Comparison Criteria Normalized 21
3.5 Relative priority 21
3.6 Index Table [64]. 22
3.7 Usability Parity Comparison Matrix 22
3.8 Usability Relative Priority 23
3.9 Speed Parity Comparison Matrix 23
3.10 Speed Relative Priority 23
3.11 Supportability Parity Comparison Matrix 23
3.12 Supportability Relative Priority 23

6.1 Survey Scale. 58
6.2 Survey Answers 59

List of Source Code

5.1 Improved Priberam scraping method. 41
5.2 Sample response for searching "Prótese" with PT language. 42
5.3 Similarity calculation code. 43
5.4 Find explanations endpoint. 44
5.5 Fran scraping method. 44
5.6 Rate explanation code. 45
5.7 Database helper code. 46
5.8 HTML including the virtualsign plugin by GILT. 48
5.9 Send translate text signal to avatar. 48
5.10 i18next Implementation. 49
5.11 Submit search with geolocation query. 51
5.12 Cassandra database schema. 53
5.13 EAC API Systemd Service Definition. 54
5.14 Cassandra database's docker compose definition. 55
5.15 NGINX HTTP Configuration. 56

Chapter 1

Introduction

This chapter provides the context of the dissertation, a description of the problem, the objectives and how this document is structured.

1.1 Problem

According to the World Health Organization, as of March 2022, 1.5 billion people live with some degree of hearing loss, which leads to some having to resort to sign language to communicate [1].

Sign languages do not have a one-to-one correspondence between signs and words or concepts in spoken language, and sign language interpreters must have a deep understanding of both the sign language and the spoken language in order to accurately convey meaning. This problem is recurrent when it comes to scientific domains, for example, to understand a concept such as *nanotechnology*, for which there are no signs for, a sign language user has to access tools with content made for oral language users.

1.2 Context

This project is the continuation of the work documented in "Explicação automática de conceitos" [2].

As it was already stated in the previous work, sign languages have a shorter vocabulary than oral languages because they rely on hand gestures, facial expressions and body language rather than sounds. This limited vocabulary is due to the physical constraints of conveying meaning through movement. Thus, concepts are the key aspect for meaningful learning [3], and the explanation of said concepts remains at the core of this thesis.

Differences from the previous work include: fixing, polishing and implementing new features on the base provided. As well as deploying to production where it will be exposed to the target audience.

1.3 Objectives

Taking into account the problem previously identified, the solution is to ease the process of understanding such concepts by explaining/defining them first and then providing a friendlier translation.

The core of the solution is the same as the previous work [2]. That being, a web application featuring an Application Programming Interface (API) that utilizes Text Mining, Information Scraping, and Information Retrieval techniques to produce explanations of words. The application allows users to search for a word and displays the generated explanation in plain text.

The previous work never went to production and needed some fine-tuning on its features and design, as well as new features that bring value to the end solution, which will serve as tasks for this development:

- Communication between services improved
- Scraping algorithm's complexity reduced (improved performance)
- Introduced like and dislike system
- Introduced database to improve quality of service
- Audit system introduced with geolocation information
- Adding multi-language possibilities for the sign languages of Portugal, Germany, Greece and Slovenia

Details on these previous topics are given in Chapter 5.

This thesis aims to improve the previous work and complement it in such a way that it becomes useful for the deaf, that is, displaying the explanations in sign language instead of spoken language in Portuguese. On top of this, the new application will also gather usage data so it can provide explanations in local dialects.

1.4 Structure

This dissertation is divided in 7 chapters:

- Chapter 1 - Introduction: Includes the definition of the problem, the context of the work, and its objectives.
- Chapter 2 - State of the Art: Provides information about the topics involved in this dissertation's theme and the technologies used for its development.
- Chapter 3 - Value Analysis: Provides an analysis relating this work's value with its cost.
- Chapter 4 - Analysis and Design: Provides an analysis on the requirements of this project accompanied by the adequate design diagrams.
- Chapter 5 - Implementation: Focus on the implementation that consists in this project as a whole, as well as on its actual deployment
- Chapter 6 - Evaluation: Describes the verification and validation process as well as the evaluation methodology from the point of view of end-users
- Chapter 7 - Conclusion: Present main takeaways obtained with the development of this work and present future work.

Chapter 2

State of the Art

In this chapter, several subjects will be presented and related to the core of this thesis, including Sign Language, Information Extraction, and Text Mining. Additionally, information about readability metrics, related work and various technologies mentioned in this document are tackled. All these topics compose the state of the art of this document.

2.1 Sign Language

Sign language is a visual language that is used primarily by deaf and hearing impaired persons, to communicate with each other and with hearing individuals. According to the World Federation of the Deaf, there are an estimated 72 million deaf people worldwide, and roughly 80% of them live in developing countries. Of these individuals, it is estimated that approximately 60% use sign language as their primary means of communication [4]. In addition to being used by the deaf community, sign language is also used by hearing individuals who work with deaf or hard-of-hearing individuals, such as interpreters, educators, and healthcare professionals.

The use of sign language has many benefits for deaf individuals. Research has shown that sign language can improve cognitive, social, and emotional development. Sign language has also been found to be an effective means of communication for deaf individuals in a variety of settings, such as at home, at school, and in the workplace [5] [6].

Despite the many benefits of sign language, there are still many challenges faced by the deaf community. One major challenge is the lack of access to sign language interpretation in many parts of the world. This can make it difficult for deaf individuals to communicate with hearing individuals and to access important services such as healthcare and education. However, efforts are being made to promote the use of sign language and to increase access to sign language interpretation and other resources for the deaf community.

2.2 Information Extraction

Information extraction (IE) is the process of automatically extracting structured information from unstructured or semi-structured sources, such as natural language text, web pages, or social media posts. The goal of information extraction is to transform unstructured data into a more structured format that can be easily analyzed and interpreted by machines.

The outcome of an information extraction procedure varies depending on the specific case, as it can be customized to meet the requirements of the intended use. Presently, these applications have the capability to address personal, scientific, and business needs.

There are many different approaches to information extraction, including rule-based systems, machine learning, and hybrid systems that combine several approaches [7].

In recent years, deep learning approaches, such as neural networks, have shown to be promising in improving the performance of information extraction systems. One example is a recent study by Shubham Chatterjee where an entity-oriented neural network is used to answer questions [8].

2.3 Text Mining

Text Mining, also known as text data mining, is a process of extracting valuable insights and knowledge from large collections of unstructured textual data. It falls within the intersection of several fields, including Data Mining, Information Retrieval, Information Extraction, Machine Learning, Knowledge Discovery, and Natural Language Processing. It involves uncovering implicit, previously unknown, and potentially valuable information from a corpus, for example, a collection of text documents [9].

Text Mining is similar to Information Extraction, the main difference being the latter involves the extraction of specific structured information and relations, while the former focuses on discovering general unsuspected information and new relations[10].

The goal of Text Mining is to combine a human's linguistic capabilities with the processing power of a computer[11]. To achieve its goals, it utilizes numerous algorithms and techniques that are commonly employed in related fields.

Some of the most used Text Mining techniques[12][13][14] are:

- **Summarization** - This technique involves condensing a longer text by removing unnecessary details, while retaining the main points and overall meaning. There are two types of text summarization methods: extractive and abstractive. Extractive methods involve selecting important sentences or paragraphs based on statistical and linguistic features, while abstractive methods aim to create a human-like interpretation of the document and express those concepts in natural language with linguistic methods. The end result is a shorter text that contains the most important information from the original text.
- **Categorization** - This technique involves placing a document into a pre-established category, in order to determine its main theme. To achieve this, a computer program will count how many times each word appears in the document. This count is then used to identify the primary topics addressed in the document. Typically, tools that employ this technique have a mechanism for ranking the documents based on the amount of content related to a particular topic.
- **Clustering** - This technique is used to sort and group similar documents into specific clusters. Unlike categorization, the clusters are not pre-defined and are determined during the execution of the technique. One advantage of clustering is that a document can be assigned to more than one cluster.
- **Concept linkage** - This technique involves linking related documents by recognizing the common concepts between them. Its main objective is to offer an information browsing experience rather than a search-based approach, as seen in Information Retrieval.

- **Question Answering** - This technique involves using natural language queries to identify the most appropriate answer to a specific question. If a website has the capability of performing question answering, it can enable users to pose questions to a computer and receive either an exact or a related answer.
- **Information Visualization** - This technique enables users to visually browse through text sources in a hierarchical or map format, in addition to simple searching. The process of visualization involves three essential steps: data preparation, data analysis and extraction, and visualization mapping. By resorting to this technique, it is possible to narrow down a vast array of documents and investigate related topics.

2.4 Online Dictionaries

Online dictionaries are a convenient and readily accessible resource for anyone seeking to define a word, check spelling, or learn about word usage. Online dictionaries offer a range of features such as audio pronunciations, synonyms, antonyms, and examples of word usage in context.

Some online dictionaries, such as Merriam-Webster [15] and the Oxford English Dictionary [16], have a long history and are highly regarded as authoritative sources of information on the English language. There is also Priberam [17] and Infopédia [18] which provide some of the same useful features but for the Portuguese language. Other online dictionaries, such as Dictionary.com [19] and Urban Dictionary [20], are more modern and offer a more casual approach to language.

Overall, online dictionaries have become a tool for anyone seeking to improve their language knowledge knowledge and usage, as well as vocabulary range, or simply satisfy their curiosity about words.

2.5 Readability Metrics

Readability metrics are tools that help measure the ease of understanding of a piece of text by analyzing its language features and structure according to a score [21]. By using these metrics, it is possible to understand if a certain text is accessible and comprehensible to a target audience.

Some common readability metrics [22] for the English language are:

- **Flesch Reading Ease Score** - This metric outputs a score ranging from 0 to 100, with higher values indicating better legibility and readability. This metric was initially designed to assess the legibility of educational texts and takes into account the total number of words, sentences, and syllables in a text to calculate its score. The score is then mapped to corresponding readability levels, such as "very difficult" or "difficult".
- **Flesch-Kincaid Grade Level** - This metric recalculates the Flesch Reading Ease Score. Unlike the mapped values of the Reading Ease Score, the Grade Level produces a result equivalent to the minimum grade level of education required to understand the text. For instance, a score of 8 would correspond to at least an eighth-grade education level.
- **Gunning Fog Index** - This metric produces scores ranging from 0 to 20, which correspond to the education grade level required for the reader to understand the text on the first reading. Unlike the Flesch-Kincaid Grade Level, this metric considers the

presence of complex words in the text. Complex words are words with three or more syllables, with a few exceptions.

- **Automated Readability Index** - This readability metric generates a score corresponding to the education grade level of the reader as well. However, unlike other metrics, it takes into account the word length, which is calculated based on the number of characters rather than the number of syllables.
- **Coleman-Liau Index** - This metric generates a score approximating the minimum education grade level required to comprehend a particular text in the United States. Like the Automated Readability Index, this metric uses the number of characters to calculate the word length. However, it differs in that it only takes into account a sample of 100 words.
- **Dale-Chall and New Dale-Chall** - These metrics were developed based on the Flesch Reading Ease to determine the grade level necessary to comprehend a particular text. What sets this metric apart from others is that it considers the number of "hard" words, which are words that do not appear in a predefined list of familiar and easy-to-read words in the English language. By having a defined list of words, this metric can adjust the difficulty level for different contexts.
- **Simple Measure of Gobbledygook** - This is another readability metric that maps the resulting score to the grade level. What sets this metric apart is that it only considers the number of polysyllabic words in a text. Polysyllabic words are those with three or more syllables and are calculated from a sample of 30 lines, comprising the first 10, middle 10, and last 10 lines.
- **Fry Graph** - This metric estimates the grade level required for a reader to comprehend a text based on a graph created by the author. The graph was developed under the assumption that texts with shorter sentences and words with fewer syllables are more readable. To calculate its score, this metric considers the number of sentences and syllables in a sample of 100 words. However, this metric is bound to words with less than six characters of length.
- **Raygor Estimate Graph** - This is a readability metric that, like the Fry Graph, estimates the required grade level for a reader using a graph. However, unlike the previous one, this metric also considers the number of words with six or more characters in the text.
- **FORCAST** - This metric can indicate either the required grade level or the age needed to comprehend a text. Unlike the other readability metrics, FORCAST is commonly used to evaluate multiple-choice quizzes and forms rather than text.
- **SPACHE** - This metric, similar to Dale-Chall, considers the grade level required for comprehension based on sentence length and the number of unfamiliar words. However, unlike Dale-Chall, this metric is specifically designed to assess the readability of texts intended for readers at the third-grade level or below.

2.6 Technology

This section will tackle the technologies that were investigated to use in the application development, as well as their comparison with other technologies when needed, and why one was chosen over the other(s).

2.6.1 Libraries and Frameworks

This subsection contains the libraries and frameworks investigated for the application.

Natural Language Toolkit

Natural Language Toolkit (NLTK) is a widely-used Python library for working with human language data. It provides a variety of tools and resources for tasks such as tokenization, stemming, tagging, parsing, and sentiment analysis. It also includes a variety of corpora and datasets for research and experimentation, as well as an extensive documentation and a large community for support. NLTK is often used in the fields of natural language processing (NLP) and computational linguistics to develop and test models, and to process and analyze textual data.

This library was built with four goals in mind:

- **Simplicity** - To provide users with useful insights into NLP, while avoiding the cumbersome processes typically involved in language data processing.
- **Consistency** - To ensure uniformity in interfaces and data structures, as well as to use straightforward method names.
- **Extensibility** - The structure is designed in such a way that it can easily and seamlessly integrate new software modules.
- **Modularity** - To provide individual components that can function independently of one another.

These goals are stated in a book written by the authors of this library [23] which is a guide to using NLTK, with practical examples, suitable for advanced users and beginners alike.

Scrapy

Scrapy [24] is a widely-used open source Python framework designed for web scraping and web crawling. It provides a flexible and powerful toolset for extracting data from websites and APIs, and can also be used for more general-purpose web crawling. A web crawler is a program that automatically downloads and indexes website content.

Web crawling is the first step in building any web information retrieval system [25]. Scrapy's primary component is its self-contained crawler, or "Spider", which executes a set of instructions based on the goal it is trying to achieve. These instructions can be as simple as a starting page for a basic crawler or as complex as using XPaths and CSS expressions to extract targeted information and store it in a database.

Scrapy also provides the added benefit of being able to target specific parts of a document, rather than the entire page, using the built-in Selectors mechanism. This can be particularly useful for crawling websites that are filled with hyperlinks, such as Wikipedia. Additionally, Scrapy offers the option to deploy spiders in the cloud, although this feature typically requires payment for unlimited usage, like many other cloud solutions.

Beautiful Soup 4

Beautiful Soup 4 (BS4) [26] is a Python library that allows parsing and extracting data from HTML and XML documents. It provides a simple and intuitive way to navigate, search,

and modify the parse tree, making it easier to extract the desired data from an HTML document. It works by creating a parse tree from the HTML document and then allowing to navigate and manipulate the tree using intuitive syntax. This makes it a great tool for web scraping, data mining, and other applications where data needs to be extracted from HTML documents.

When it comes to parse trees, they are generated using dedicated parser libraries. The table 2.1 below outlines the available parsers for Beautiful Soup 4 along with their pros and cons [27].

Table 2.1: Parsers libraries (Beautiful Soup 4).

Parser	**Advantages**	**Disadvantages**
Python's html.parser	Decent speed Lenient	Not as fast as lxml, less lenient than html5lib
lxml's parser HTML	Very fast Lenient	External C dependency
lxml's parser XML	Very fast The only supported XML parser	External C dependency
html5lib	Extremely lenient Parses pages like a web browser Creates valid HTML5	Very slow External Python dependency

When opting to use an external parsing library, lxml [28] is considered the most secure choice as it is both the fastest and can handle both HTML and XML parsing. On the other hand, html5lib [29] is a suitable choice when parsing web pages that may contain errors in their HTML structure. This library adheres to the HTML5 standards, which means it can correct incomplete or mismatched tags, as well as add any necessary missing tags.

Flask

Flask is a Python microframework that is open source and suitable for both web applications and microservices. The term "micro" in microframework refers to its minimal dependencies on external libraries, relying only on the Jinja template engine and the Werkzeug WSGI toolkit [30].

Compared to other Python frameworks like Django [31], Flask is more flexible and customizable, allowing developers to choose and integrate only the specific tools and libraries they need. Flask also has a smaller learning curve than Django, which comes with a lot of built-in functionality and can be overwhelming for beginners. However, because Flask is more minimalistic, it requires more manual configuration and can be less suitable for larger, more complex applications. Other popular Python frameworks like Pyramid [32] and Bottle [33] fall somewhere in between Flask and Django in terms of complexity and functionality.

Flask can integrate with various types of databases, including SQL and NoSQL databases, which simplify database integration and management in Flask applications [34].

React

React is a popular JavaScript library for building user interfaces [35]. It was developed by Facebook and is now widely used to create fast and responsive web applications.

It works by breaking down user interfaces into components, which are reusable and self-contained modules that can be easily combined to create complex interfaces.

One of the main advantages of React is its ability to update and render only the necessary components, which can greatly improve the performance of web applications. React also has a large and active community that has contributed a wide range of tools and libraries to extend its functionality.

React also stands in the top 3 web frameworks used as of 2022, with a total of 42,62% developers using it [36].

Analysis

All of these libraries and frameworks (NLTK, Scrapy, BS4, Flask and React) were already being used in the previous work [2]. Their alternatives are mentioned in the subsection below.

Alternatives

Regarding NLTK there are alternatives such as OpenNLP[1] and Google Cloud NLP[2]. However, it was proven in the previous work's[2] value analysis that NLTK is the best option for this project.

The library responsible for scraping used in the project is Scrapy together with BS4 for parsing. There are alternatives such as ZenRows[3] and Selenium[4]. But Scrapy, despite being open-source instead of having a price like ZenRows, it's also not as resource-intensive as Selenium. With response time being an important factor, the Scrapy and BS4 combination proved more adequate.

There are many alternatives to Flask, such as Django[5], Falcon[6], and Eve[7]. Since Flask was already in use and is a well documented framework, no alternative was integrated in the project.

For the web application, React is used, however there are alternatives, such as AngularJS[8] and Vue.js[9]. Angular is a good contender due to its robustness and ease of use, even so, much like Flask, React was already in use and it proved a good opportunity to learn more about this unused library.

[1] https://opennlp.apache.org/
[2] https://cloud.google.com/natural-language
[3] https://www.zenrows.com/
[4] https://selenium-python.readthedocs.io/
[5] https://www.djangoproject.com/
[6] https://falcon.readthedocs.io/en/stable/
[7] https://docs.python-eve.org/en/stable/
[8] https://angularjs.org/
[9] https://vuejs.org/

2.6.2 Databases

This subsection describes the databases investigated for this project.

MySQL

MySQL is a well-known open-source relational database management system currently maintained and developed by Oracle Corporation. It is a popular choice to store and manage data due to its ease of use, speed, and scalability. MySQL offers support for several programming languages and platforms, including PHP, Java, Python, and Node.js. Its robust features and strong security measures make it an excellent choice for businesses of all sizes.

One of MySQL's significant characteristics is its ability to handle large amounts of data efficiently. It uses a client/server architecture, where the server manages the database and multiple clients access it. This enables several users to access and update the database concurrently without causing conflicts or slowing down the system. Furthermore, MySQL has various built-in features that help optimize performance, such as indexing and caching. Overall, MySQL is a powerful and dependable database management system that has been widely adopted around the world [37].

Apache Cassandra

Apache Cassandra [38] is an open-source, distributed NoSQL database originally developed by Facebook. It is designed for handling large amounts of data across many commodity servers while maintaining high availability and no single point of failure. Cassandra's flexible data model makes it a popular choice for use cases such as real-time analytics, IoT applications, and online recommendation systems.

One of the key features of Apache Cassandra over traditional relational databases like MySQL is its ability to handle large volumes of data at scale with high throughput and low latency. Cassandra's distributed architecture and its use of a peer-to-peer gossip protocol make it optimized for read-heavy workloads and able to quickly process queries across large data sets. Additionally, Cassandra's distributed and fault-tolerant architecture provides high availability and reliability in the face of node failures, making it a popular choice for use cases where uptime and resiliency are critical [39].

Comparison

Since the data stored was mostly for auditing purposes and information which did not require relations, Apache Cassandra was the database of choice.

2.6.3 Others

This subsection tackles the technologies that do not fit in the previous categories.

Docker

Docker is a containerization platform that allows developers to package applications and their dependencies into a portable, self-contained unit. This makes it easy to share and deploy applications across different environments, such as development, testing, and production [40].

Docker containers can be easily shared and re-used on Docker Hub [41], a central repository of public and private containers. Developers can search for and download pre-built containers for a wide range of applications, libraries, and tools, saving time and effort in setting up development environments.

Its ease of use and portability has made it a popular choice among developers and DevOps teams. Docker's simplicity makes it easy for developers to get started with containerization, and its large and active community provides a wealth of resources and support [42]. In addition, Docker's flexibility and scalability make it a valuable tool for deploying applications in a wide range of environments, from small startups to large enterprises [43].

Geolocation DB

Geolocation DB is a service that provides IP geolocation data for businesses and individuals who need to identify the physical location of an IP address [44]. This data can be used for a variety of purposes, such as marketing, security, and fraud prevention.

PM2

PM2 is a process manager for Node.js applications that provides a simple and efficient way to manage multiple Node.js processes on a single server. PM2 can be used to start, stop, and monitor applications, as well as manage their logs and environment variables. It is designed to keep applications running continuously, ensuring that they are always available to handle requests [45].

PM2 provides a number of powerful features that make it a popular choice for managing Node.js applications in production environments. One of the key features of PM2 is its built-in load balancer, which distributes incoming traffic across multiple instances of a Node.js application to ensure that requests are handled efficiently. It also provides detailed monitoring and logging capabilities, allowing you to easily track the performance and health of running applications [46].

NGINX

NGINX is an open-source web server software that is known for its high performance, reliability, and scalability. It can handle large volumes of concurrent connections and is particularly efficient at serving static content. NGINX can also be used as a load balancer to distribute incoming traffic across multiple servers, making it a popular choice for websites that need to serve large amounts of data quickly [47].

NGINX's ability to handle a wide range of protocols, including HTTP, HTTPS, SMTP, and POP3, makes it a versatile web server software. In addition to serving content, NGINX can act as a reverse proxy, allowing it to direct traffic to multiple servers based on different criteria such as load balancing or geographic location [48]. This makes it a powerful tool for high-traffic websites that require improved performance and reliability.

This web server is widely used, according to usage statistics it is used in 34,2% of websites [49].

Analysis

NGINX and PM2 were already being used in the previous work [2] and there was no need for alternatives.

Alternatives

For Docker there were no alternatives since Docker remains as the best container solution. The alternative in this use case would be to install the service contained directly in the server provided.

For GeolocationDB there are a few alternatives such as IP2Location[10] and DBIP[11]. However, GeolocationDB is free and it is the easiest to setup. It's security mechanism is also rather simple since it simply consists of an app-key as a request header.

2.7 Related Work

This section presents related work that tried to achieve similar results through different approaches.

Noraset Thanapon et al. [50] pursued a different method to accomplish a comparable objective. Thanapon employed a Deep Learning strategy, which employed a recurrent neural network (RNN) model with distributed representations of words, known as word embeddings, to produce dictionary representations. These models were trained on a pre-existing dataset.

Ni Ke et al. [51] also opted for a Deep Learning method that involved a RNN model to produce an interpretation of a given word or expression from "tweets", which are posts created by Twitter[52] users. The objective was to analyze non-standard expressions, including slang, featured in these posts. To train the RNN model, an online, user-contributed dictionary called Urban Dictionary [20] was utilized.

Giorgina Dinu et al. [53] attempted to create a sentence that accurately conveyed the meaning in a distributional vector. The method was evaluated in two test scenarios. The first scenario involved generating the phrase in English within a monolingual setting, while the second scenario was cross-lingual, in which the vector was initially used to generate the English phrase and then translate it to Italian.

William Dolan et al. [54] presented an automated approach to generate a lexical knowledge base using online dictionaries. The approach was utilized to construct a directed graph that established semantic connections between words found in the Longman Dictionary of Contemporary English [55]. According to the authors, using a knowledge base offers more comprehensive information about a word's meaning compared to a typical lexical lookup.

Alan Akbik et al. [56] created an algorithm named Wanderlust, which automatically extracts semantic relationships from natural language text. The English Wikipedia [57] corpus was utilized to apply this algorithm and semantic relationships were obtained to populate a semantic wiki.

Atin Das et al. [58] proposed a theoretical model that resorts to Neural Networks to extract from an article what the authors refer to as "featured words". These "featured words" are

[10] https://lite.ip2location.com/
[11] https://db-ip.com/

the words that most accurately describe a particular article. As this is a theoretical study, there are no tests or specific use cases provided.

Amirata Ghorbani et al. [59] developed an algorithm that automatically extracts visual concepts and provides the data to a Machine Learning model, enhancing the model with concept-based explanations instead of feature-based explanations which are commonly used. One of the main drawbacks is it's functionality is restrained to only images.

Liana Ermakova et al. [60] proposed a system to simplify scientific literature. It is a three steps process that consists of: content selection, complexity spotting and text simplification. The study aims to make scientific papers easier to understand for everyone, including people not versed in scientific language.

In sum, the majority of works related to explaining a concept or word use Deep Learning and train neural networks to achieve a result. However, no works resort to Text Mining and Information Retrieval and Extraction to achieve the same goal.

Chapter 3

Value Analysis

This chapter will provide an analysis on the value this work provides and relate it with the cost.

3.1 New Concept Development Model

The New Concept Development Model (NCD) is a systematic approach that provides a framework for converting an opportunity into a concept by following specific stages. By doing so, it enables a clear and structured evaluation of the value of the opportunity [61].

The five key stages that make up this model were used to define the opportunity this project has:

- **Opportunity identification** - A member of the deaf community identified an opportunity where there were no existing solutions to aid the process of explaining a concept using sign language. This is because sign language has a limited vocabulary and many words do not have a direct translation, necessitating the use of available gestures to convey meaning.
- **Opportunity analysis** - During analysis, it was found that the only tools with some similarity to the solution were sign language dictionaries. However, these dictionaries had certain limitations that made them impractical. For example, in order to add a new translation, a person must be recorded performing the necessary signs. Additionally, these dictionaries did not have translations for scientific terms.
- **Idea creation** - During this stage, several potential solutions were generated to address the identified opportunity, with the best option chosen from the other alternatives analyzed. One such idea was to develop a solution that automates the process of explaining a concept. This could be achieved by utilizing information extraction and text mining techniques to extract an explanation from an online source. To make the explanation accessible to the deaf community, it would be translated into sign language using an existing GILT application.
- **Idea selection** - During this phase, the objective was to merge all potential ideas and approaches into a single one that satisfies the necessary requirements. The chosen idea involves creating an automatic interpretation system that is tailored to the needs of the deaf community. This system will employ information extraction and text mining techniques to generate an explanation for a given word or expression. Additionally, it will translate the explanation into sign language.

- **Concept definition** - Once the idea was selected, the subsequent step was to establish the necessary objectives to accomplish it. This work aims to create an API capable of generating an explanation for a particular concept, as well as a web application that will employ this API. Additionally, an existing GILT application will be utilized in the web application to assist the deaf community in the process of concept explanation.

3.2 Value

The perceived value depends on the benefits that are recognized and the sacrifices that are identified in Table 3.1 below.

Table 3.1: Perceived Value

	Product	**Service**	**Relationship**
Benefit	Knowledge	Utility	Trust
Sacrifice		Search costs	

3.3 Value Proposition

The aim of this project is to improve the process of explaining concepts, with a particular focus on making it inclusive for the deaf community. The approach involves utilizing information extraction and text mining techniques to create a system that can generate an explanation of a given word or expression and also provide its sign language translation.

To further describe the value proposition, the Osterwalder's Value Proposition Canvas is presented in Figure 3.1. This model has two sides, the Customer Profile in which the customer's understanding is clarified and the Value Map that describes how value will be created for that customer. When one side meets the other, Fit is achieved, which means customers get excited about the value proposition created [62].

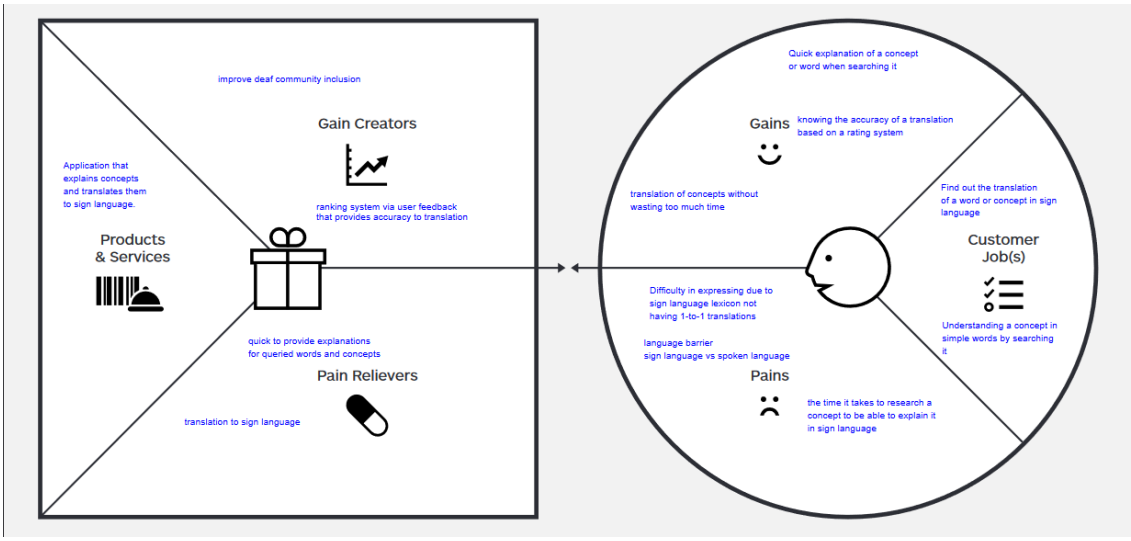


Figure 3.1: Value Proposition Canvas

Starting with the Customer Profile, the Customer Job(s) section aims to describe actions the customers try to get done on their daily routine. In this case, understanding a concept by searching it as well as finding a translation for a concept in sign language.

The Customer Pains section describes anything that can be an inconvenience during the jobs mentioned previously. For this project, the topics tackled are the difficulty in translating words since sign language doesn't have a one-to-one relation with spoken languages and how time consuming searching for a concept to understand it, and then translating it, can be.

The Customer Gains section describes the outcomes and benefits the customers want, expected or desirable. In this case this section contains items such as quickly knowing how to explain a concept and having it's translation in sign language, as well as knowing the accuracy of an explanation.

Moving to the Value Map, the Products & Services section is a list of what can be offered, in this case it's simply an application that explains concepts and translates them to sign language.

The Pain Relievers section describes how the products and services defined previously can alleviate customer's pains. For this project they are the quickness in providing an explanation for searched concepts and being able to translate explanations to sign language to alleviate the language barrier.

Lastly, the Gain Creators section describes how the products and services mentioned previously create customer gains. In this case, the topics on this section are the application improves the deaf community's inclusion and a ranking system based on user feedback that helps in establishing the accuracy of an explanation.

3.4 Business Model Canvas

Zooming out of the model presented in the previous section 3.3 there is another tool that can help in strategic management planning, the Business Model Canvas. It provides 9 key blocks to aid in the process of planning the business concepts [63].

- **Key Partners** - Refers to the group of suppliers and collaborators who play a crucial role in enabling the business model to function effectively.
- **Key Activities** - Outlines the essential tasks or operations that a company needs to perform in order to ensure the successful implementation of its business model.
- **Key Resources** - Refers to the critical assets or resources that a business model relies upon to operate effectively.
- **Value Propositions** - Refers to the combination of goods and services offered by a business that delivers significant value to its targeted Customer Segments.
- **Customer Relationships** - Outlines the various kinds of interactions and connections that a company builds with its targeted Customer Segments.
- **Channels** - Refers to the methods or mediums through which a company communicates with and reaches its Customer Segments to deliver its Value Proposition effectively.
- **Customer Segments** - Refers to the distinct categories of individuals or entities that a company intends to target and provide its products or services to.

- **Cost Structure** - Encompasses all the expenses involved in operating a business model.
- **Revenue Streams** - Refers to the income or monetary value that a company generates from each of its targeted Customer Segments.

The Business Model Canvas for this work is illustrated in Figure 3.2 below.

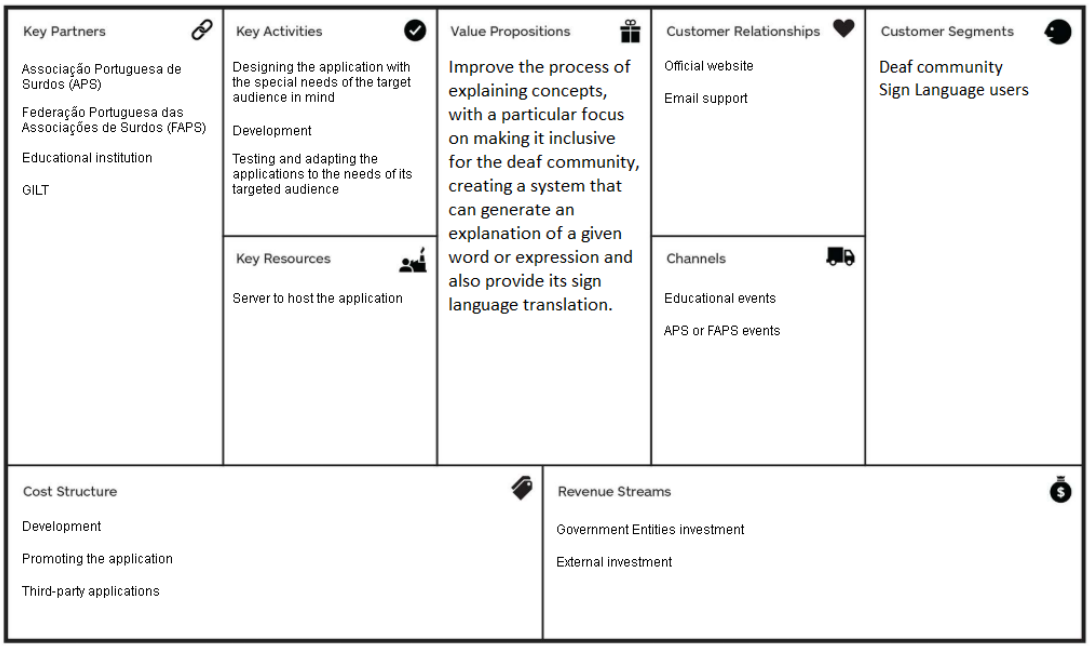


Figure 3.2: Business Model Canvas

Firstly, the solution created targets two customer segments: the deaf community and Sign Language users.

For the product provided to these customers is an application that helps users of Sign Language learn concepts, promoting equal opportunities and social inclusion for people with disabilities.

To reach the target audience, the solution should be presented at educational and association events for the Portuguese Deaf Association or the Portuguese Federation of Deaf Associations.

The solution will be available on a website, offering all the necessary information and email support to address any concerns the customers may have.

In regard to the revenue for the solution, it will come from investments made by external companies or government entities.

The server to host the application is the most critical resource required for this solution.

It is crucial to design and develop the application with the special needs of the target audience in mind, test and adapt the solution to meet their needs, and prioritize the key partners such as education institutions, the Portuguese Deaf Association, the Portuguese Federation of Deaf Associations, and GILT.

Lastly, the costs for developing and promoting the application, along with any third-party applications used to scale the solution, will be incurred.

3.5 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) is a useful tool for navigating complex decision-making processes. It enables individuals to establish priorities by considering both qualitative and quantitative criteria, thereby breaking down complex decisions into a set of paired comparisons. Utilizing this approach can also help mitigate any biases that may arise during the decision-making process [64].

3.5.1 Stage 1 - Defining the goal

The goal of this analysis is to find which Database is best suited for this project. To make this analysis 3 criteria will be used to compare database systems and in the end decide which one is better suited.

3.5.2 Stage 2 - Defining the Hierarchy Tree

The databases to be analysed will be Apache Cassandra and MySQL, according to the criteria of Usability, Efficiency and Supportability.

The hierarchical decision tree is presented in figure 3.3.

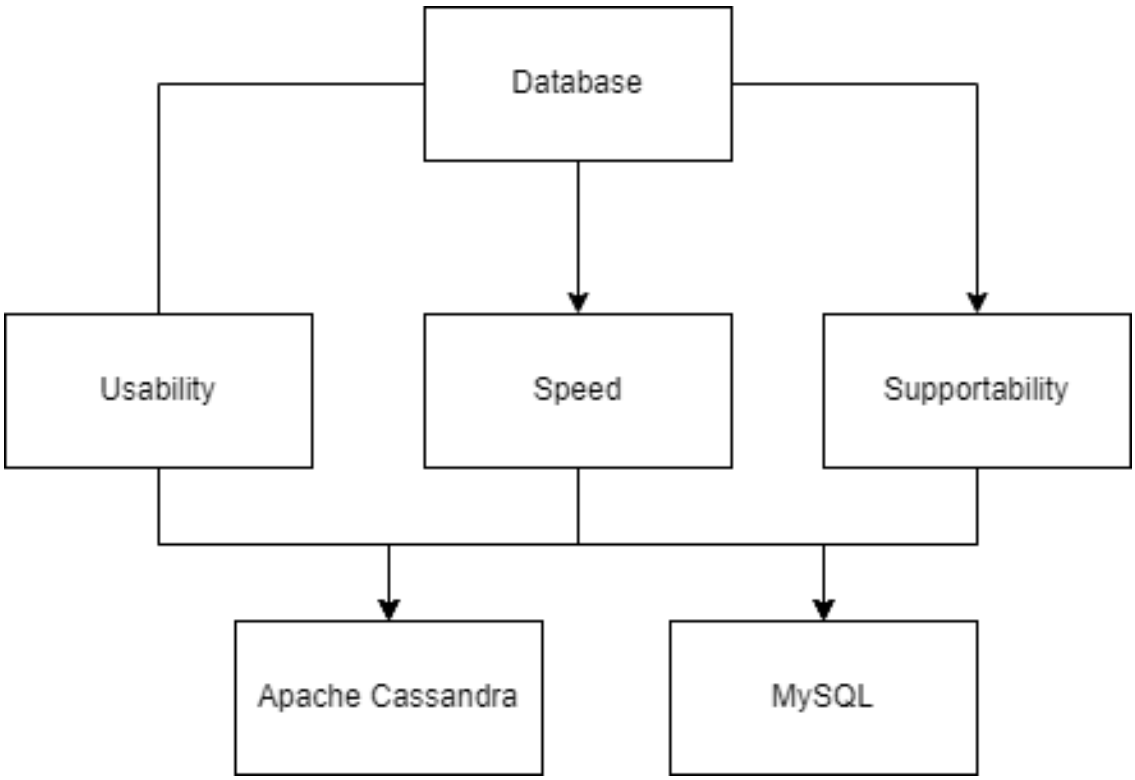


Figure 3.3: Hierarchical Decision Tree

In regard to the criteria, each of them had the following logic for being picked:

- **Usability** - How easy it is to utilize each of the tools.
- **Speed** - How fast each tool is when searching for information
- **Supportability** - The number and quality of reliable sources of information about each tool (e.g. Books, Documentation, Forums, etc.).

3.5.3 Stage 3 - Pair wise comparison

This stage involves using the Fundamental scale [64] to define priorities between each criteria. The Fundamental scale is present in Table 3.2 below.

Table 3.2: Fundamental scale

Intensity of importance	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective
2	Weak	
3	Moderate importance of one over another	Experience and judgment strongly favor one activity over another
4	Moderate plus	
5	Essential or strong importance	Experience and judgment strongly favor one activity over another
6	Strong plus	
7	Very strong importance	An activity is strongly favored and its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation

Values 2,4,6 and 8 don't have an explanation since they are intermediate values between judgments so when they are selected a compromise is needed.

After considering their respective significance, the values from the table mentioned above were used to define each comparison between two criteria. The outcome of these comparisons concerning the decision tree's criteria is presented in Table 3.3.

Table 3.3: Comparison Criteria

Criteria	Usability	Speed	Supportability
Usability	1	2	1/3
Speed	1/2	1	1/3
Supportability	3	3	1
Sum	9/2	6	5/3

After analysing the table it's possible to conclude that Supportability is of moderate importance (3) over Usability and slightly more important (2) than Speed. It's also possible to conclude that Usability is slightly more important (2) than Speed.

3.5.4 Stage 4 - Relative priority of each criteria

This stage involves in calculating the normalized criteria values and its relative priorities. This can be done by diving each value by the sum of it's respective column. The normalized values can be seen in Table 3.4

Table 3.4: Comparison Criteria Normalized

Criteria	**Usability**	**Speed**	**Supportability**
Usability	2/9	1/3	1/5
Speed	1/9	1/6	1/5
Supportability	2/3	1/2	3/5

The next step is to calculate the relative priority of each criteria. This is achieved by calculating the arithmetic mean of each criteria. The results can be seen in Table 3.5. All values will be rounded to the third decimal case.

Table 3.5: Relative priority

Criteria	**Relative priority**
Usability	0.252
Speed	0.159
Supportability	0.589

With these calculations it's possible to conclude that the priority of criteria, from most to least important, is Supportability, followed by Usability and lastly Speed.

3.5.5 Stage 4 - Consistency evaluation of relative priorities

This stage's purpose is to calculate the Consistency Ratio (CR) to assess the consistency of the priorities defined in Table 3.3. To calculate it the Equation 3.1 is needed.

$$CR = \frac{CI}{RI} \tag{3.1}$$

To calculate the Consistency Index (CI) the Equation 3.2 is required.

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{3.2}$$

In the previous equation, n is the number of criteria and λ_{max} is the largest eigenvalue of the matrix.

It's possible to then use Equation 3.3 to multiply the comparison matrix with it's priority vector.

$$\begin{bmatrix} 1 & 2 & 1/3 \\ 1/2 & 1 & 1/3 \\ 3 & 3 & 1 \end{bmatrix} * \begin{bmatrix} 0.252 \\ 0.159 \\ 0.589 \end{bmatrix} = \begin{bmatrix} 0.766 \\ 0.481 \\ 1.822 \end{bmatrix} \tag{3.3}$$

As well as Equation 3.4 to calculate λ_{max}.

$$\lambda_{max} = \frac{\frac{0.766}{0.252} + \frac{0.481}{0.159} + \frac{1.822}{0.589}}{3} = 3.05 \tag{3.4}$$

It's possible to calculate the CI with the equation below.

$$CI = \frac{3.05 - 3}{3 - 1} = 0.025 \tag{3.5}$$

The Random Consistency Index (RI) is shown in Table 3.6 below.

Table 3.6: Index Table [64].

N	**1**	**2**	**3**	**4**	**5**
RI	0	0	0.58	0.90	1.12

Since n is 3 the RI used will be 0.58 and it's possible to obtain the Consistency Ratio by doing

$$CR = \frac{0.025}{0.58} = 0.043 \tag{3.6}$$

As the value obtained in Equation 3.6 is 0.043 and less than 0.1 (1%) it can be concluded that the values attributed to the properties are consistent.

3.5.6 Stage 5 - Constructing the parity comparison matrix per criteria

This stage aims to create a parity comparison matrix for each of the alternatives in the hierarchy tree, per criteria. The relative priorities are calculated in the same way as the criteria in Stage 4, subsection 3.5.4.

The calculations can be seen through the following tables:

Table 3.7: Usability Parity Comparison Matrix

Usability	**Apache Cassandra**	**MySQL**
Apache Cassandra	1	1/2
MySQL	2	1
Sum	3/2	3

Table 3.8: Usability Relative Priority

Usability	**Relative Priority**
Apache Cassandra	0.333
MySQL	0.667

Table 3.9: Speed Parity Comparison Matrix

Speed	**Apache Cassandra**	**MySQL**
Apache Cassandra	1	7
MySQL	1/7	1
Sum	8/7	8

Table 3.10: Speed Relative Priority

Speed	**Relative Priority**
Apache Cassandra	0.875
MySQL	0.125

Table 3.11: Supportability Parity Comparison Matrix

Supportability	**Apache Cassandra**	**MySQL**
Apache Cassandra	1	1
MySQL	1	1
Sum	2	2

Table 3.12: Supportability Relative Priority

Supportability	**Relative Priority**
Apache Cassandra	0.500
MySQL	0.500

To finalize this step, the values calculated are added to the hierarchy tree previously defined, as shown in Figure 3.4.

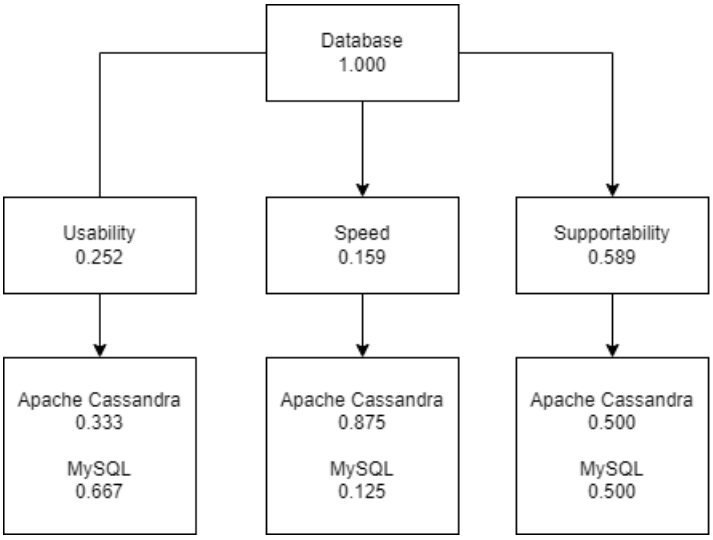


Figure 3.4: Hierarchical decision tree with criteria parity comparison

3.5.7 Stage 6 - Obtain the composite property for alternatives

This stage aims to calculate the composite property for alternatives. To obtain it, the matrix that has each criteria relative priority is multiplied by the criteria weight, as it's demonstrated in the Equation 3.7 below.

$$\begin{bmatrix} 0.333 & 0.875 & 0.500 \\ 0.667 & 0.125 & 0.500 \end{bmatrix} * \begin{bmatrix} 0.252 \\ 0.159 \\ 0.589 \end{bmatrix} = \begin{bmatrix} \mathbf{0.518} \\ 0.482 \end{bmatrix} \tag{3.7}$$

3.5.8 Stage 7 - Choosing alternative

During this last stage, the acquired values are scrutinized and evaluated to identify the optimal alternative. Based on the selected criteria and their calculated significance, It can be confidently concluded that Apache Cassandra is the most favorable option as it has the highest value (0.518). The findings from this procedure indicate that the database used in the project will be Apache Cassandra.

Chapter 4

Analysis and Design

This chapter starts by introducing Requirements Engineering, which involves the design of software requirements in the form of both functional and non-functional requirements. Subsequently, the logical view, process view, and the design of functional requirements are discussed. Finally, the deployment view is presented.

4.1 Requirements Engineering

This section will analyze the functional and non-functional requirements for this project. The model used is FURPS+ [65] which classifies requirements representing them in categories. It is depicted by the following acronym:

- **Functionality** - Pertains to attributes related to the primary features of the project.
- **Usability** - Focuses on qualities such as visual appeal and ensuring a consistent user interface experience.
- **Reliability** - Encompasses attributes such as system availability, accuracy of calculations, and the ability to recover from failures.
- **Performance** - Involves aspects such as throughput, response time, recovery time, start-up time, and shutdown time.
- **Supportability** - Encompasses qualities such as testability, adaptability, maintainability, compatibility, configurability, installability, scalability, and localizability.
- **"+"** - Tackles design, implementation and physical requirements.

4.1.1 Functional Requirements

This subsection provides information concerning the functional requirements of the system, represented in the form of Use Cases. These requirements serve to encapsulate the principal feature of the product and are depicted in Figure 4.1

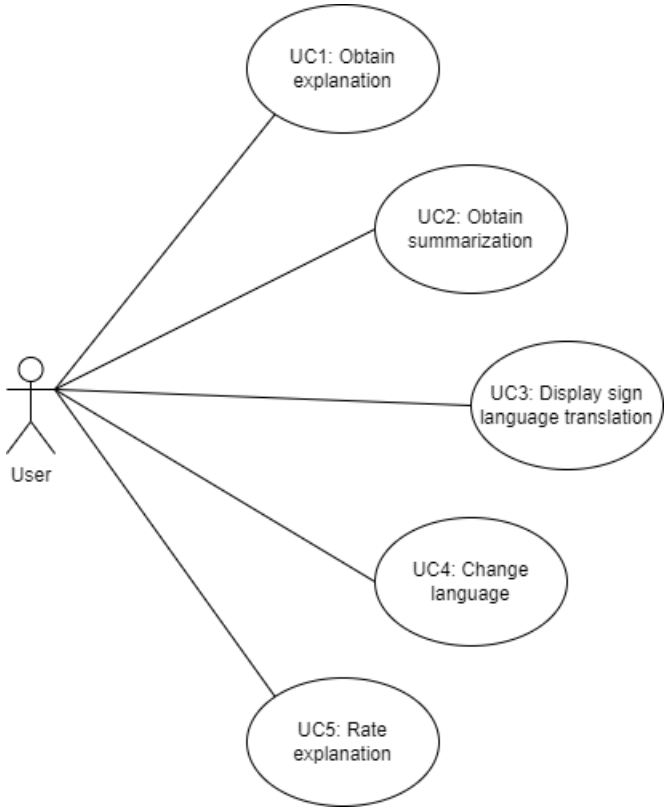


Figure 4.1: Use Case Diagram

UC1: Obtain Explanations

The user requests the system for explanations of a queried word or concept. The system should be able to provide explanations by scraping an online dictionary or retrieving them from the database and presenting them to the user.

UC2: Obtain Summarization

After Use Case 1, the user can request a Summarization to the system for a specific explanation. The system should find similarities between the explanation and the queried word's Wikipedia page, in hope of obtaining a different set of explanations to clarify the meaning of the explanation.

UC3: Display Sign Language Translation

The user requests a translation to sign language of the word or concept being queried or, after Use Case 1, the user requests the sign language translation for an explanation. The request is sent to the avatar API which in turns returns the animation representing the sign language translation.

UC4: Change Language

The user requests the system to change languages, to which the system responds with the available languages and, upon selecting one, the system changes accordingly.

UC5: Rate Explanation

The user gives a positive (thumbs-up) or negative (thumbs-down) review to an explanation. The system saves this information to show the explanations with greater positive feedback first as they are more likely to be in context for the user.

4.1.2 Non-functional Requirements

The non-functional requirements are represented by the functionalities that are transverse across the use cases mentioned in the previous subsection, as well as other requirements, usually architecturally significant.

Functionality

- **Auditing** - Actions that affect the experience for other users should be audited, in this specific case, through rating explanations.
- **Security** - The system should be secure, not allowing the user to freely interact with the services that handle the main logic behind the project.

Usability

- **User Interaction** - The user interaction should be straightforward and highly intuitive, considering the specific requirements of the target audience.
- **Help** - The system is required to offer appropriate and context-sensitive assistance to users while they are performing tasks.
- **Interface** - The interface should be visually appealing and easily readable, considering the specific requirements of the target audience.

Reliability

- **Availability** - The system is expected to maintain a high rate of availability. Ideally above 90% which means a maximum downtime of 36.5 days per year.
- **Predictability** - The system should exhibit reliability by being free from technical errors. This means the system should be able to prevent internal errors being shown to the user, catching them and treating each failure as a possible outcome and show a meaningful message, allowing the user to know what to expect.
- **Fault Tolerance** - The system should demonstrate the ability to tolerate errors in order to safeguard users from unintended mistakes.

Performance

- **Response Time** - The system should deliver fast response times to ensure swift access to data. Ideally below the timeout defined below, so below 5 seconds should be the response time for normal calls. Summarization calls however, need performance tests to know what is expected and what values are acceptable and possible to achieve.
- **Timeouts** - The system should have timeouts to avoid users having to wait in an exception where the item above is not verified. It was defined a timeout of 5 seconds

for normal dictionary calls and no timeout for summarization calls since they require IO operations and need performance tests before a value is agreed upon.

Supportability

- **Testability** - The system should possess ease of testability to instill high confidence in its correctness.
- **Maintainability** - The system should exhibit a high level of maintainability to accommodate future requirements and/or repairs.
- **Configurability** - The system should facilitate effortless configurability to enable the addition of new features.
- **Localizability** - The system should provide support for multiple languages.

Plus (+)

- **Physical** - The solution should be implemented in provided servers (DEI). No matter how many services, they should all function in one machine. Solution scalability would be addressed when the solution is shown capable of operation in current conditions.
- **Implementation** - The solution should be hosted on a system with a Linux distribution.

4.2 Logical View

The logic view illustrates the primary components of the application and their interactions which constitute the system. The core solution implemented in the previous work [2] was not changed. However there were some additions to accommodate new functionalities.

The proposed architecture is represented by the Component Diagram in Figure 4.2.

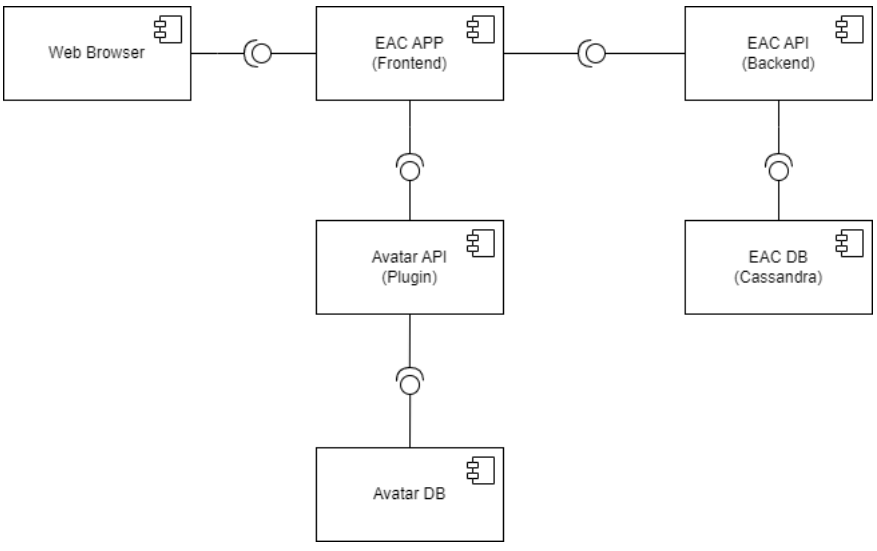


Figure 4.2: Component Diagram

In detail, the main components are:

- **EAC APP (Frontend)** - Tasked with handling user input, processing it, and converting the output from APIs to be compatible with the web browser.
- **EAC API (Backend)** - Responsible for providing explanations, summarizations, and handling feedback about explanations.
- **EAC DB (Cassandra)** - Responsible for storing explanations (in case sources are not available), as well as auditing user feedback.
- **Avatar API (Plugin)** - Tasked with providing the animations when translating text to sign language.
- **Avatar DB** - Responsible for storing information about the animations needed per language for translations.

The components made and/or improved from previous work are **EAC APP**, **EAC API** and **EAC DB**.

4.3 Deployment View

The deployment view is used to observe and describe the arrangement of physical components within a system. This view takes into consideration certain non-functional requirements of the system, such as performance, scalability, availability, and reliability. Figure 4.3 showcases the deployment diagram specifically created for this system.

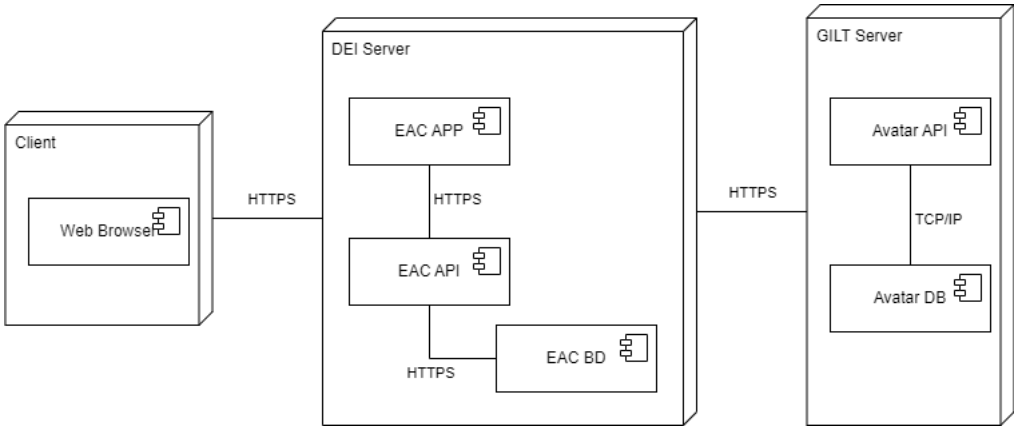


Figure 4.3: Deployment Diagram

As it is visible from the image above, **EAC APP** and **EAC API** will be improved and new functionality will be developed, whilst **EAC BD** is a new component to deploy and be available for EAC APP to interact with. These 3 components will be hosted in DEI which is ISEP's computer engineering department.

The avatar-related components are hosted in GILT's servers, which only EAC APP will communicate with via the plugin made available by GILT.

4.4 Process View

Before devising a possible solution, the previous work's [2] design was analysed and taken into consideration, so it would be possible to be closer to the optimal solution's design.

The initial task involved outlining a broad perspective of the core functionalities that the application should possess. Figure 4.4 illustrates the primary activities of the intended application in the form of an activity diagram.

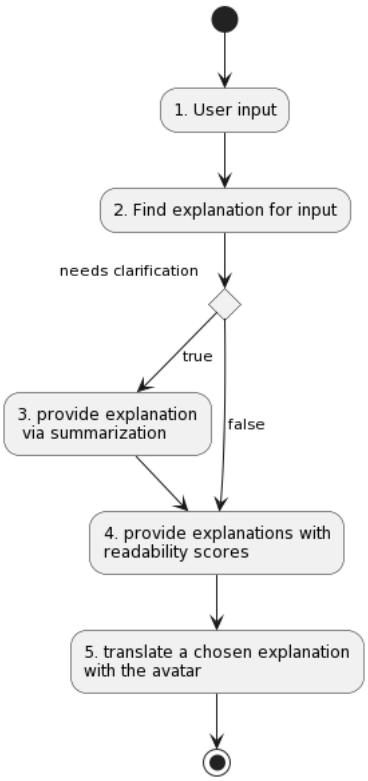


Figure 4.4: Activity Diagram - Possible Solution

The activities are ordered as the normal flow occurs.

The input received in 1 is used by the second activity, which finds explanations by scraping an online dictionary. In case more explanations are needed, activity 3 comes into play and provides a summarization of a chosen explanation based on the Wikipedia page of the user input. Then, with activity 4, the solution provides the explanations with their associated readability scores. Lastly, any of the provided explanations can be translated with the sign language avatar.

Each of the activities can be detailed with their own activity diagram, which will be illustrated below. This will provide a better understanding of each activity previously mentioned, starting with the first activity, as represented in Figure 4.5.

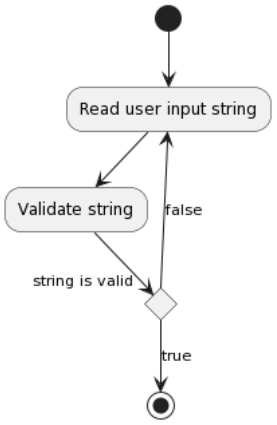


Figure 4.5: Activity Diagram - 1. User input

This activity is simple and contains a loop. It will read a user's input string and validate it; if the string is not valid, it will read another input. The string's validity, for now, is related to containing spaces or not, because currently the system only supports words and concepts without spaces in between.

After the first activity, there is the activity 2, illustrated in Figure 4.6.

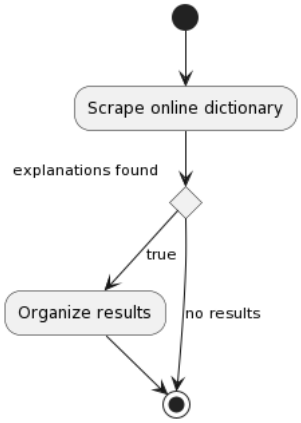


Figure 4.6: Activity Diagram - 2. Find Explanations

Activity 2 will apply Text Mining techniques such as web scraping to obtain explanations from an online dictionary. This information, if found successfully, is organized, otherwise no explanations are provided. Then, this result can be provided to activity 3 or 4, depending on the clarity of the explanations. If the user wants to, he can request a summarization triggering activity 3, but the normal flow of events is to advance to activity 4.

However, assuming the user requested a summarization and following the ordering of the activities, below is Figure 4.7 representing activity 3.

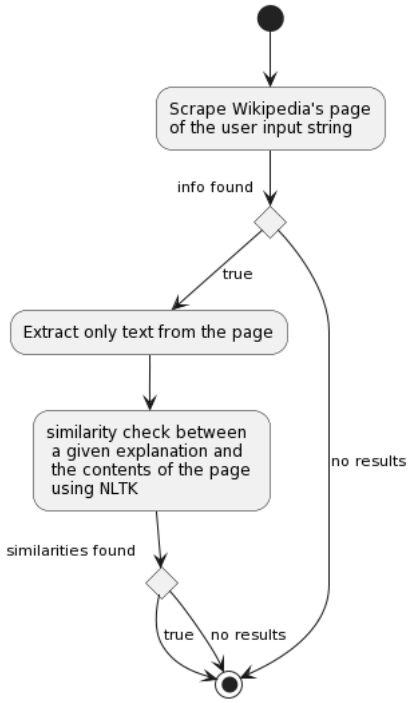


Figure 4.7: Activity Diagram - 3. Provide Summarization

Activity 3 is where the solution has more processing weight. It will perform web scraping on the Wikipedia's page for the user input string and if there is information the solution will extract the text and then check the similarities between a given explanation and the contents of the page that was obtained previously. If similarities were found, those similarities (which are now explanations) will be provided into activity 4. If no information is found while web scraping or when no similarities are found, activity 4 will proceed with no results.

As previously mentioned, activity 4 is represented by the Activity Diagram in Figure 4.8.

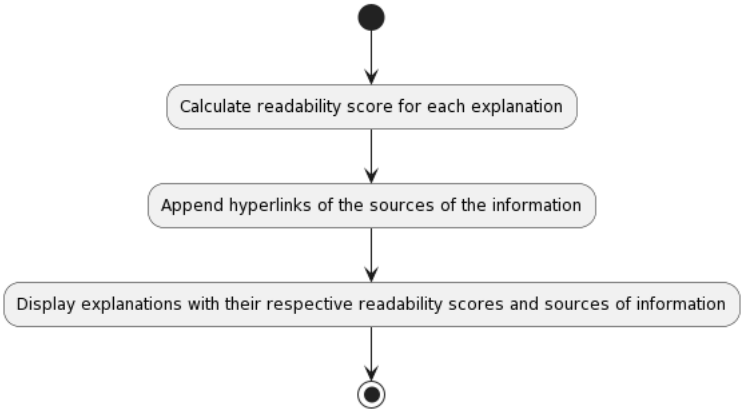


Figure 4.8: Activity Diagram - 4. Provide Explanations with Scores

Activity 4 simply takes the explanations from the previous activities, calculates their readability scores (in sign language), and appends the hyperlinks of the source of information, which can be the Wikipedia page of the user input, as well as the online dictionary entry

for the user input. After appending those to the response, the solution displays all that information to the user.

Lastly, Figure 4.9 which represents the final activity represented in the first Activity Diagram of the solution as a whole.

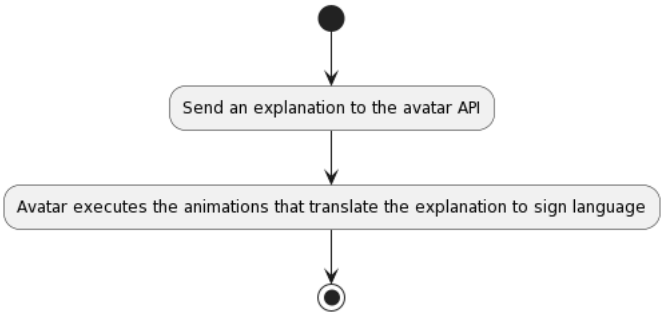


Figure 4.9: Activity Diagram - 5. Translate Explanation

Activity 5 will send the user's chosen explanation to the Avatar API, which in turn will execute the translation of the explanation to sign language and display it to the user.

4.5 Functional Requirements Design

This section is specifically dedicated to showcasing the sequence diagrams of the use cases that were briefly introduced in section 4.1. A sequence diagram is employed to depict the involved objects and their interactions within a functionality, arranged in a chronological sequence.

Throughout this section any reference to "Backend", "Frontend", and "Cassandra" will mean: "EAC API", "EAC APP", and "EAC DB".

4.5.1 UC1: Obtain Explanations

Below in Figure 4.10 is the design of obtaining an explanation.

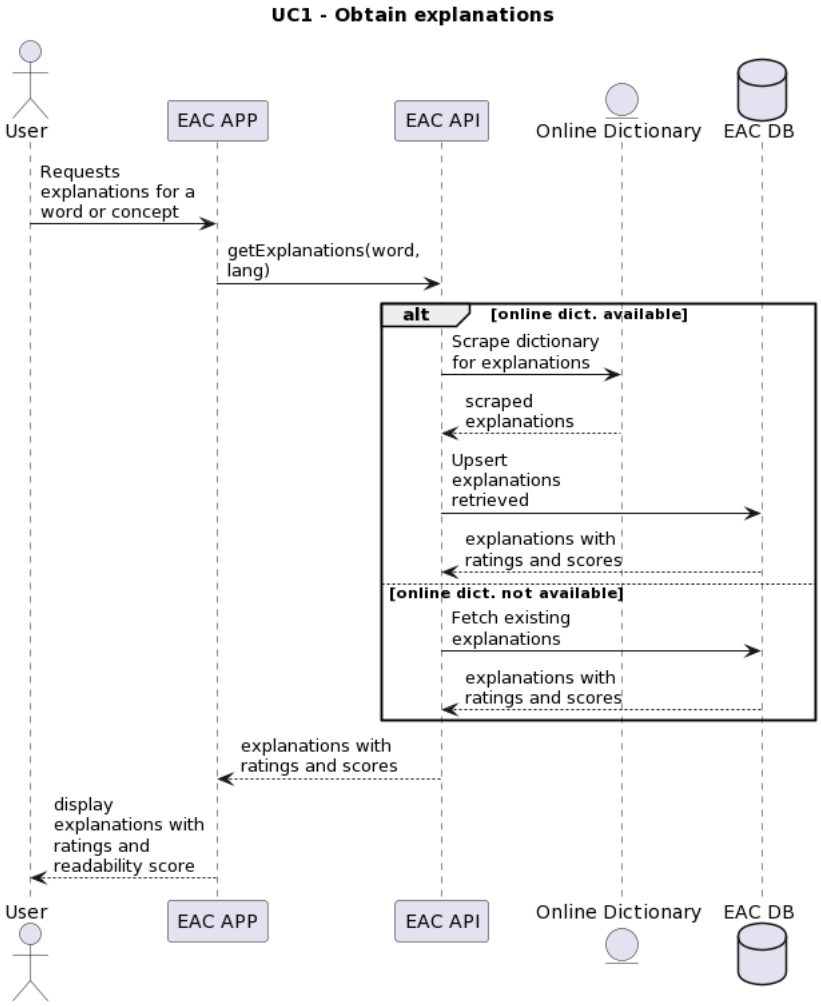


Figure 4.10: UC1: Obtain Explanation

Firstly, the user requests explanations by querying a word or concept in the Frontend. Secondly, the Frontend will request the Backend sending the word or concept as a parameter, as well as the language. The language is sent so the Backend knows which Online Dictionary to scrape. The Backend scrapes the online dictionary for all the explanations and saves them in the database with their calculated readability scores, later returning them to the Frontend. In case the online dictionary is not available, the explanations stored in Cassandra will be returned to the Frontend instead. Lastly, the Frontend displays all the explanations, ratings, and readability scores.

4.5.2 UC2: Obtain Summarization

Displayed below in Figure 4.11 is the design of the obtain summarization use case.

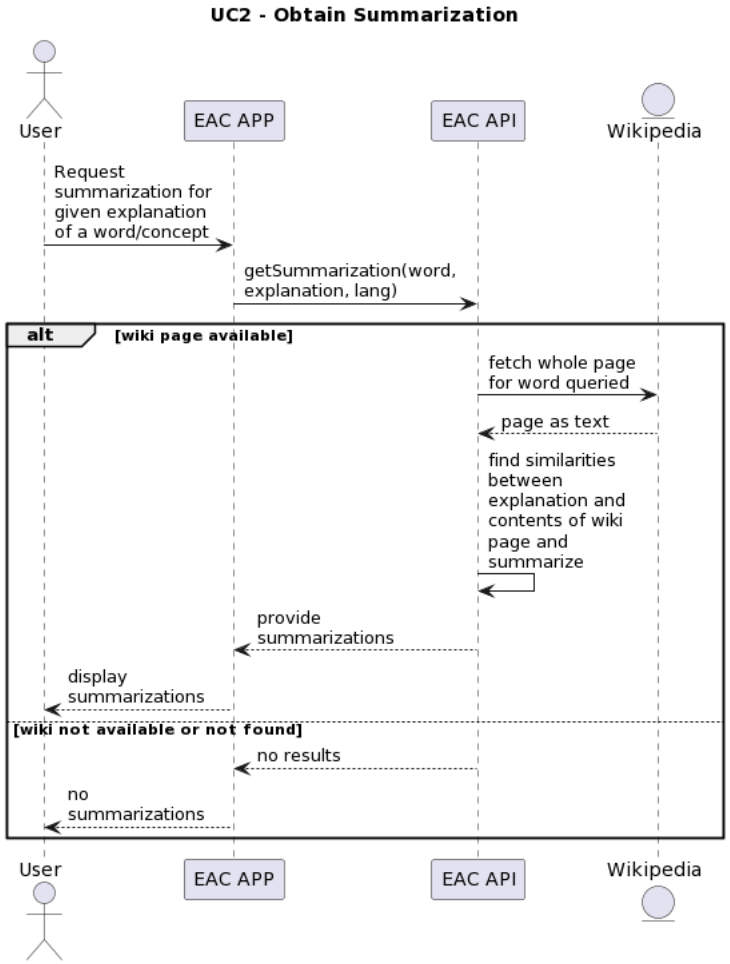


Figure 4.11: UC2: Obtain Summarization

When looking at an explanation, the user starts by requesting a summarization to the Frontend. This results in a request to the Backend with the word or concept that was queried in the first place, the explanation in which the user requested a summarization and the language. The language is once again important so the Backend can fetch the Wikipedia page in the correct language. The Backend, as mentioned, fetches the whole Wikipedia page for the word or concept received. Later, the Backend obtains all the relevant text from the page and runs a summarization algorithm using NLTK with the text in the page and the explanation received. This results in one or multiple summarizations which the Backend returns to the Frontend, together with their readability scores. Finally, the Frontend displays the summarizations to the user. If there is not a Wikipedia page for the word, no summarizations will be obtained and the information is conveyed to the user.

4.5.3 UC3: Display Sign Language Translation

Shown in Figure 4.12 below, is the sequence diagram of displaying the sign language translation use case.

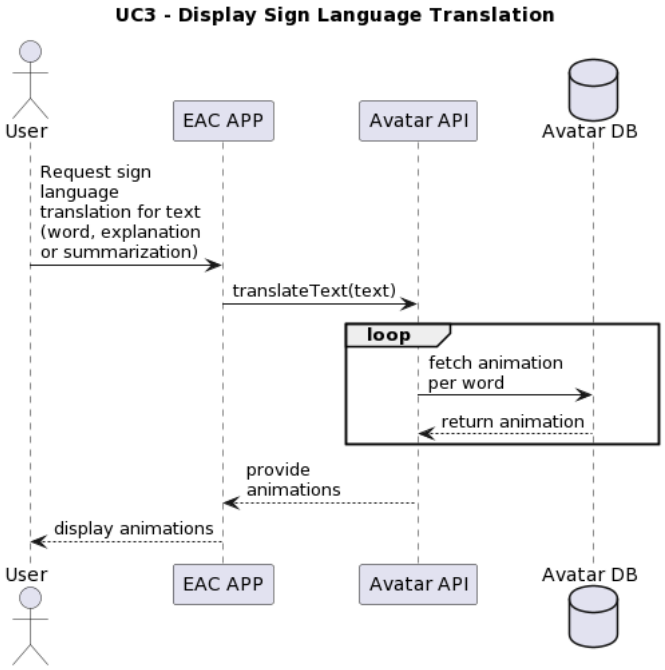


Figure 4.12: UC3: Display Sign Language Translation

The user starts by requesting a translation to the word queried, to an explanation or to a summarization to the Frontend. The Frontend has a plugin installed to communicate with the Avatar API, which promptly requests a translation of the text requested by the user. The Avatar API then requests the multiple animations needed for translating the expression given. When completed the Avatar API returns the complete set of animations translating the expression, and in the end, Frontend displays them to the user.

4.5.4 UC4: Change Language

The Figure 4.13 shows the sequence diagram related to the change language use case.

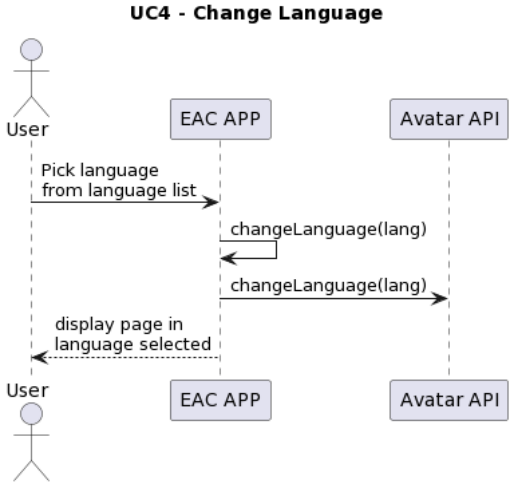


Figure 4.13: UC4: Change Language

The user begins by selecting a language from the possible languages that the Frontend provides. After that, the Frontend itself changes the web application's language, displaying it to the user, as well as requesting a change in language to the Avatar API via the plugin installed, which will make any translations translate to the correct sign language.

4.5.5 UC5: Rate Explanation

Lastly, below in Figure 4.14 is the sequence diagram of how rating an explanation happens.

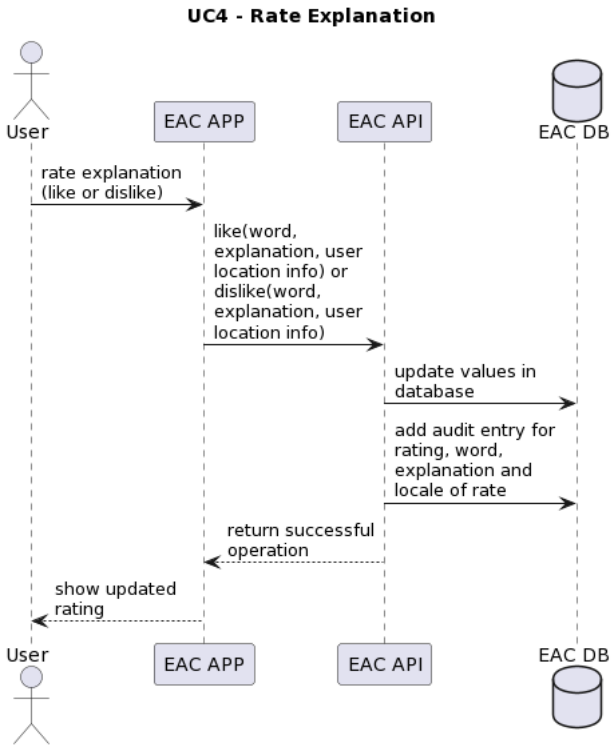


Figure 4.14: UC5: Rate Explanation

Starting with having an explanation displayed, the user decides to like or dislike the explanation and makes this request to the Frontend. The Frontend carries this request to the Backend providing the word or concept queried in the first place, the explanation the user interacted with and the user's location information. This information is obtained by querying GeolocationDB[44] before the request, storing it in the browser memory, then, whenever a user rates an explanation that information is sent as together with the action (positive or negative rating).

After this, the Backend will update the values in Cassandra (increasing the number of likes or dislikes), as well as adding an audit entry containing what explanation for what word was liked or disliked and the location information of the user. Next, the Backend returns the operation was a success and the Frontend will display to the user the updated rating.

Chapter 5

Implementation

This chapter focuses on explaining the implementation details of the developed solution. It begins by mentioning the major changes since the previous work [2], and then explaining the two main services: EAC API and EAC APP, the backend and frontend services respectively, as well as the Cassandra database. Both of the services and their functionalities are described together with how all the components are deployed.

Details on the new functionalities will be added for each section of the service as well as changes that were made to improve current functionalities.

5.1 Major Changes

As mentioned previously, there were changes to the solution as a whole to increase its efficiency as well as support new functionalities.

5.1.1 Support Other Languages

Previously the only supported language was Portuguese. Although it had English, which translated the website, the API was not ready for English. This was changed and support was added to other languages. Currently the languages Portuguese, Slovene, Greek and German are fully implemented and English is partially supported as, by the time of writing, it is missing the online reference dictionary needed to provide explanations.

5.1.2 Rating Explanations

A mechanism for rating explanations was added so the user can visualize if a certain explanation is trustworthy or not, as well as there is a high probability that the higher the rating, the more likely an explanation is to be correct. This was paired with ordering the explanations based on ratings to enhance the user experience by visualizing the most probable correct explanations closer to the search bar.

5.1.3 Added Database

A Cassandra database, explained in detail in Chapter 2, was used to provide resiliency and a stable response time. It stores explanations that were previously queried for and saves the number of likes and dislikes each explanation has. It was also added for auditing the rating mechanism which currently is only serving as storage but can be used for some interesting data since the action (like or dislike) is registered for an explanation together with the location of the user that rated it.

5.1.4 Help Page

A support page was added which at the moment of writing, contains an image and a tutorial of how to use the application, which later should contain help in video format to clarify how to use this solution.

5.2 EAC API - Backend

The EAC API was already present in the previous work [2]. However, it was improved and extended to enhance current functionalities and support new ones.

This API was developed using Flask, a lightweight open-source Python framework, explained in detail previously in Chapter 2.

The two main functionalities of this API are:

- **Find explanations** - Scrapes online dictionaries to provide explanations for an input string. This functionality is fast.
- **Summarization** - Scrapes Wikipedia page for an input string and finds similarities between the text on the page and an input explanation, returning a new explanation. This functionality is slower and heavier since it handles files and uses NLTK to handle finding similarities.

There is a third functionality which is calculating readability score, which calculates how complicated an explanation is in LGP (Língua Gestual Portuguesa/Portuguese Sign Language). This functionality is only available for Portuguese and was not changed.

These functionalities remain from the previous work [2], but as mentioned previously they suffered changes to improve their efficiency. In the subsection below global changes made in the EAC API will be presented, followed by detailed changes regarding the main functionalities.

5.2.1 Improvement Changes

This subsection will detail changes that improved the working solution.

Find explanations functionality

To improve how fast the API could scrape the Portuguese online dictionary, Priberam, a new method was created and is visible on Listing 5.1.

```python
def minimal_scrap(word, timeout):
    results = []
    path = Priberam.URL + word
    try:
        page = requests.get(path, headers=Priberam.HEADERS, timeout=timeout)
    except requests.exceptions.RequestException:
        print("priberam call timed out!")
        results.append(Priberam.ERROR_MESSAGE)
        return results

    soup = BeautifulSoup(page.content, "lxml")
    # if it has the alert it's not a word
    if soup.find('div', {'class': 'mb-12 alert alert-info'}):
        results.append(Priberam.NOT_FOUND_MESSAGE)
        return results

    main_content = soup.find('div', {'id': 'resultados'})
    all_definition_p = main_content.find_all('p', {'class': 'py-4 dp-definicao-linha'}) \
                       + main_content.find_all('p', {'class': 'ml-12 py-4 dp-definicao-linha'})

    for defs in all_definition_p:
        context = None
        meaning = None
        context_html = defs.find(name='span', attrs={'class': 'varpt'}, recursive=False)
        if context_html is not None:
            context = context_html.get_text().strip().replace('[', '').replace(']', '')

        meaning_html = defs.find(name='span', attrs={'class': 'def'})
        if meaning_html is not None:
            meaning = meaning_html.get_text().strip()

        if meaning is not None and meaning != '':
            results.append((context, meaning))

    return results
```

Listing 5.1: Improved Priberam scraping method.

This method now supports a timeout to avoid hanging the request, as well as search the HTML tags for any warning so the error can be properly propagated to the user. Then, to find the explanations, it targets specific HTML tags that were analyzed to contain the explanation and context of an explanation, aggregate them, and return them. A sample response is present in Listing 5.2 below.

```
{
    "additionalInfo": [
        "https://www.infopedia.pt/dicionarios/lingua-portuguesa/protese",
        "https://www.lexico.pt/protese",
        "https://pt.wikipedia.org/wiki/protese"
    ],
    "definition": [
        {
            "context": "Cirurgia",
            "sentence": "Dispositivo ou aparelho que tem por fim substituir um orgao de que se faz ablacao ou amputacao parcial ou total ou melhorar uma funcao (ex.: protese auditiva, protese mamaria).",
            "score": 4.053,
            "likes": 5,
            "dislikes": 0
        },
        {
            "context": "Linguistica",
            "sentence": "Figura que consiste em juntar uma letra ou uma silaba no principio de uma palavra (como em arruido por ruido).",
            "score": 3.828,
            "likes": 1,
            "dislikes": 2
        }
    ],
    "expression": "Protese",
    "source": "https://dicionario.priberam.org/protese"
}
```


Listing 5.2: Sample response for searching "Prótese" with PT language.

It is visible that the most liked explanation will be on top of the list and there is a context associated to each explanation.

Summarization functionality

Previously, the API did the following in order:

1. Clean workspace
2. Scrape Wikipedia page for input string and save it to a file in workspace
3. Extract from the HTML in the file the text content into another file
4. Run similarity between the input explanation and the text in the file
5. Calculate readability score for each explanation and order them using it
6. Return the explanations and readability scores

There was a clear improvement to be made. The file writes and reads were reduced as they are costly and instead work on application memory until returning the results. However, there is a limitation in using Scrapy's spiders, detailed in Chapter 2, which saves to an HTML file and cannot save to memory, so the improvement was done on extracting text content from the HTML file. This change made the summarization process faster.

The similarity calculation was also improved by rewriting the similarity calculation code. This improvement was not performance-related but readability-related, as the code was too complex which could affect how well the solution is maintained in the future. The resulting code is present in Listing 5.3.

```
nltk_supported_languages = [Language.PT.name, Language.EN.name]

def sim(prev_sentence, summarization, lang):
    if lang in Language._member_names_ and lang in
    nltk_supported_languages:
        language = Language[lang].value
        sim_array = file_sentence_similarity(prev_sentence,
    summarization, language)
        return format_result(sim_array)
    return 'Language not supported'


def file_sentence_similarity(prev_sentence, sentences, language):
    sim_array = {}
    for sentence in sentences:
        if len(sentence) > MIN_CHAR:
            if "." in sentence:
                vecSentence = sentence.split(".")
                for newSent in vecSentence:
                    newSent = newSent.strip()
                    if len(newSent) > MIN_CHAR:
                        sim_array[newSent] = calculate_similarity(
    prev_sentence, newSent, language)
            else:
                sim_array[sentence] = calculate_similarity(
    prev_sentence, sentence, language)

    return sim_array
```

Listing 5.3: Similarity calculation code.

5.2.2 New functionalities Changes

The changes in this subsection are related to new functionalities that were added to the API.

Support Other Languages

To support other languages in the two main functionalities the API suffered changes that also simplified and eased code readability. Taking the first functionality as an example, finding explanations by scraping an online dictionary, the endpoint stayed as visible in Listing 5.4 and each language logic was extracted to a separate file.

```
@app.route('/search')
def search():
    lang = request.headers[LANGUAGE_HEADER].upper()
    print(lang)

    if lang in Language._member_names_:
        match Language[lang]:
            case Language.PT:
                return pt_logic.search(request, timeout)
            case Language.EN:
                return en_logic.search(request, timeout)
            case Language.SL:
                return sl_logic.search(request, timeout)
            case _:
                return 'Language not implemented.'
    else:
        return 'Invalid language.'
```

Listing 5.4: Find explanations endpoint.

Then, to implement Slovene language support the online dictionary Fran [66] was used as reference. The Listing 5.5 displays how the scraping mechanism was implemented, in a much similar way to the implementation of the Priberam one in Listing 5.1.

```
def minimal_scrap(word, timeout):
    results = []
    path = Fran.URL + word

    try:
        page = requests.get(path, headers=Fran.HEADERS, timeout=
    timeout)
    except requests.exceptions.RequestException:
        print("fran call timed out!")
        results.append(Fran.ERROR_MESSAGE)
        return results

    soup = BeautifulSoup(page.content, 'lxml')

    # if it has a button requesting new word it doesn't exist in dict
    if soup.find('a', {'class': 'btn btn-primary btn-lg new-word'})
    or soup.find(string=re.compile('Va e iskanje ni bilo uspe no'))
    :
        results.append(Fran.NOT_FOUND_MESSAGE)
        return results

    main_content = soup.find('div', {'class': 'entry-content'})
    all_definitions = main_content.find_all('span', {'data-group': '
    explanation '}, recursive=False)

    for defs in all_definitions:
        results.append((None, defs.get_text().strip()[:-1]))

    return results
```

Listing 5.5: Fran scraping method.

Rating explanations

To accommodate this new change mentioned in the beginning of the chapter, new endpoints in the API were developed, liking and disliking an explanation. In Listing 5.6 there is the code to add a positive rating to a definition.

```
@app.route('/like-definition', methods=['POST'])
def like_definition():
    word = request.json['word'].lower()
    definition = request.json['definition']
    lang = request.headers[LANGUAGE_HEADER].upper()

    CassandraConn.add_like_dislike_cassandra(
        word,
        definition,
        request.json['ipv4'],
        request.json['country_code'],
        request.json['country_name'],
        request.json['location_state'],
        request.json['location_city'],
        "LIKE"
    )
    new_def = common_logic.calculate_readability_score([CassandraConn
    .find_word_def_cassandra(word, definition)],
                                                        Language[lang
    ])[0]
    return jsonify(new_def)
```

Listing 5.6: Rate explanation code.

The API takes as argument the input word of a user, an explanation and some additional geolocation information for auditing, and increments the like value of that explanation in the database.

Added Database

As previously mentioned there is a database involved and the API is dependent on its connection to provide the expected behaviour. A timeout of 2 seconds for the find explanations functionality was added: if the web scraping process takes longer than 2 seconds the API will fetch the explanations from the database, if they are present.

The API has a dedicated class to handle database operation, present in Listing 5.7.

```
class CassandraConn:

    def write_exp_def_to_cassandra(expression, definitions, source):
        for meaning in definitions:
            print("adding " + expression + " -> " + meaning[1])
            session.execute(write_def_stmt,
                            [expression, meaning[1], meaning[0],
    datetime.datetime.now(), source])

    def find_word_cassandra(word):
        definitions = []
        rows = session.execute(find_word_stmt, [word])
        for def_row in rows:
            print(f"fetched from cassandra -> {def_row.explanation}")
            definitions.append((def_row.context,
                                def_row.explanation,
                                def_row.total_likes if def_row.
    total_likes is not None else 0,
                                def_row.total_dislikes if def_row.
    total_dislikes is not None else 0))

        return sorted(
            definitions,
            key=lambda t: t[2],
            reverse=True
        )

    def find_word_def_cassandra(word, explanation):
        row = session.execute(find_word_def_stmt, [word, explanation
    ]).one()
        print(row)
        return (row.context, row.explanation, row.total_likes, row.
    total_dislikes) if row else None

    def add_like_dislike_cassandra(expression, definition,
                                   ipv4, country_code, country_name,
    location_state, location_city, action):
        rows = session.execute(find_word_def_stmt, [expression,
    definition])
        for row in rows:
            if action.upper() == 'LIKE':
                if row.total_likes is not None:
                    total = row.total_likes + 1
                else:
                    total = 1
                session.execute(write_like_stmt, [expression,
    definition, total])
            if action.upper() == 'DISLIKE':
                if row.total_dislikes is not None:
                    total = row.total_dislikes + 1
                else:
                    total = 1
                session.execute(write_dislike_stmt, [expression,
    definition, total])
        session.execute_async(write_audit_stmt,
                              [expression, definition, ipv4,
    country_code, country_name, location_state, location_city,
                               action.upper(), datetime.datetime.now
    ()])
```

Listing 5.7: Database helper code.

Each of these methods simplify how to write to the database, extract values from it and change values, which in this case happens mostly on the new rating functionality.

5.3 EAC APP - Frontend

Like the API in the previous section, the EAC APP was also present in the previous work [2]. It was also enhanced with some fixes and new functionalities already mentioned.

The EAC APP provides a user interface to interact with the EAC API. It is developed using React, an open source Javascript framework, detailed further in Chapter 2.

The React web application can be divided in components which include:

- **App** - Main component of the web application, responsible for setting the position, order, and condition in which the next components are displayed. It is also responsible for handling requests to the Backend.
- **Board with Context** - Small component that represents each card with an explanation and its info. The user will mainly interact with this component.
- **Display** - Component which will show a list of Boards mentioned above and additional information. This component will use the App component to make queries to the Backend and build Boards according to the results.
- **Navbar** - Presents the logo and title of the application, together with the language menu and a link to the help page, which due to its simplicity is not a new component but a Modal (pop-up) inside the Navbar.
- **Search** - The search bar allows users to input a word or conceptfor which they want to receive explanations and translate to sign language. It sends the request to the API, via App component, and the Display component handles the information display as mentioned above.

The components mentioned above already describe some changes that were made to them but like the previous section, the next subsections will detail what changes were made with the purpose of improvement and what changes were made to support new functionalities.

5.3.1 Improvement Changes

This subsection will detail what changes were made to improve the solution.

Avatar

Previously the Avatar was a component and mocked in the website possibly due to time constraints. However, in this iteration the Avatar was added and provided by GILT via a plugin that is referenced in the HTML as present in Listing 5.8 and is not defined as a component.

```
<body>
    <noscript>You need to enable JavaScript to run this app.</
    noscript>

    <div id="virtualsign" autostart="false" color="#021e22"></div>
    <div id="root"></div>
    <script defer src="https://virtualsigngilt.s3-eu-west-1.amazonaws
    .com/Build/vsloader.js" type="text/javascript"></script>
    <script defer src="https://virtualsigngilt.s3-eu-west-1.amazonaws
    .com/Build/virtualsign.js" type="text/javascript"></script>
  </body>
```

Listing 5.8: HTML including the virtualsign plugin by GILT.

This made implementation faster and easier, since it is a script referenced in the HTML code. All that was needed to use the plugin, was call the function present in the script like in Listing 5.9.

```
handleChangeAvatarValue = () => {
        this.setState({
            showSummarization: false
        })
        window.translateByText(this.state.explanationObject.sentence)
    }
```

Listing 5.9: Send translate text signal to avatar.

The Avatar plugin can be seen in Figure 5.1, performing a translation, along with some of the Display component containing two Boards with Context.

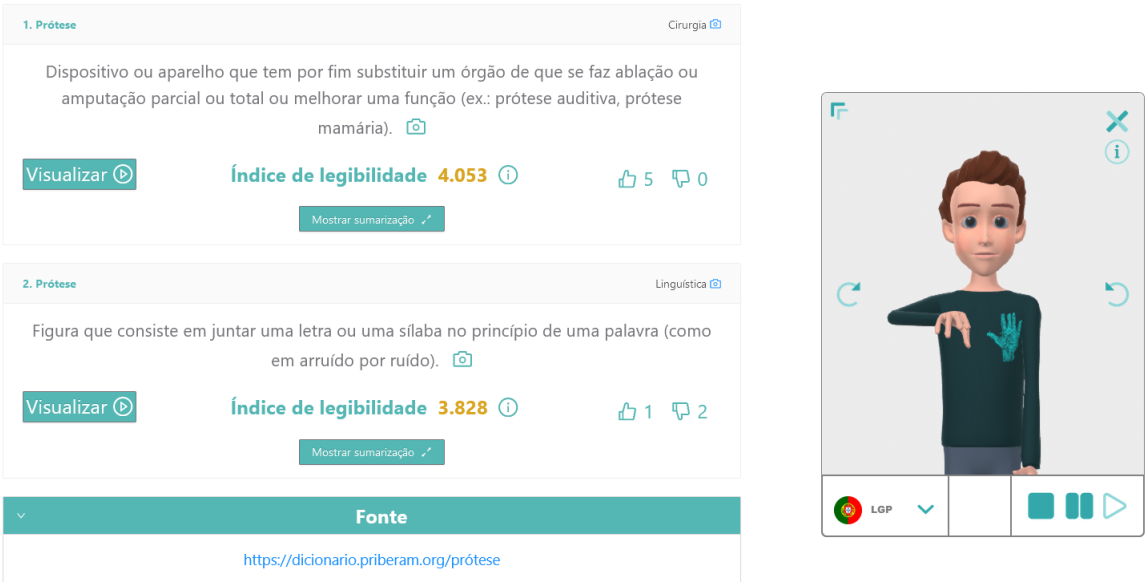


Figure 5.1: Avatar Plugin in Display

Language support

To show the website in multiple languages i18next[1] was used. However, the JSON definition was hard-coded in the Javascript file and only Portuguese and English were present, the latter being only partially implemented. This was improved by separating each language into a JSON file, isolating each language, adding full translation for all languages, and adding Slovene, Greek and German as a possible languages. This simplified the code that imports and uses the translations, as it is seen in Listing 5.10.

```
import i18next from 'i18next';
import pt from './locales/pt.json'
import en from './locales/en.json'
import sl from './locales/sl.json'

i18next
    .init({
        interpolation: {
            // React already does escaping
            escapeValue: false,
        },
        fallbackLng: 'pt',
        resources: {
            pt, en, sl, gr, de
        },
    })

export default i18next
```

Listing 5.10: i18next Implementation.

Board with Context

This component was mentioned previously and is a mix between improvement and new functionality. This new component operates similarly as the previous iteration, but has a new "context" option added which, based on the API response, if an explanation has a context it will provide a link to a Google Images search for that context. This behaviour was present for the explanation text but it was added for the context text. The functionality of showing a summarization of a word given an explanation can be triggered by a new button exposed on the component. An example of the new component can be seen in Figure 5.2.

Figure 5.2: Context Board Component Example

[1] https://www.i18next.com/

5.3.2 New functionalities Changes

This subsection will detail changes that add value to the solution.

Translate Input Word

Previously, only the explanations could be translated to sign language. But a new button below the search bar, as illustrated in Figure 5.3 was added.

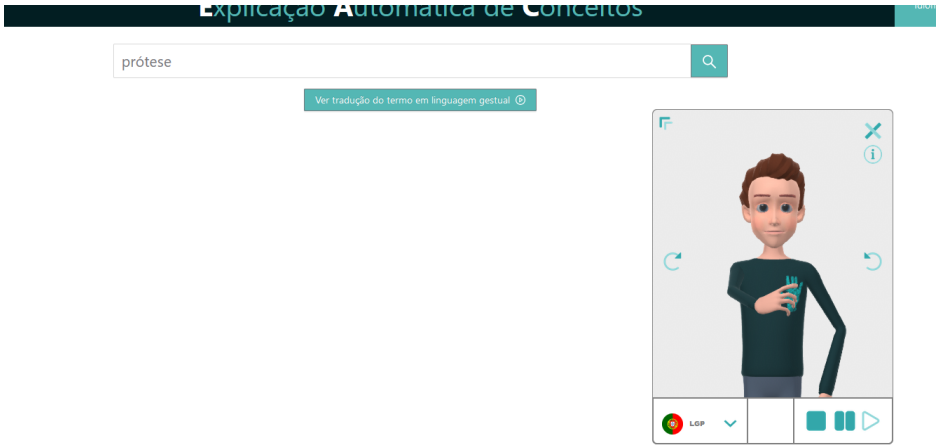


Figure 5.3: Translate Input Button

The button follows the same rules as translating explanations, much like Listing 5.9, but instead takes the user's input string and sends the text to the Avatar, so it can perform the sign language translation.

Rating explanations

The solution is now capable of storing user's feedback to a given explanation. As illustrated in Figure 5.2, on the bottom right corner the values for positive and negative feedback are shown.

These values are stored by the API in the Cassandra database. Also, as mentioned before, user's rating action is audited and appended with geolocation information. This is obtained using Geolocation DB [2], which is used as detailed in Listing 5.11 below.

[2] https://geolocation-db.com/

```
handleSubmit = async (conceptFromSearch) => {
    this.setState({
        loading: true
    })
    const response = await explanation.get('/search', {
        headers: {
            'Accept-Language': this.state.language
        },
        params: {
            query: conceptFromSearch
        }
    })
    const res = await axios.get('https://geolocation-db.com/json/7
    a9b1b60-6cd6-11ed-a5c7-1104687560a9')
    .catch(function(error) {
        console.log(error);
    });

    console.log(response.data)
    console.log(res ? res.data : 'could not find geolocation info')
    this.setState({
        explanations: response.data,
        localData: res ? res.data : this.state.localData,
        loading: false,
    })
};
```

Listing 5.11: Submit search with geolocation query.

After querying the API for results, geolocation information is queried as well and stored in the web application's state. This information is then used to populate the necessary fields to trigger a positive or negative feedback in the API to an explanation chosen by the user.

Help page

As mentioned in the major changes at the beginning of this chapter, a Help page was added to support application users and clarify any doubts regarding the application's functionalities.

The help page is not a component but a simple modal that pops up, as illustrated in Figure 5.4.

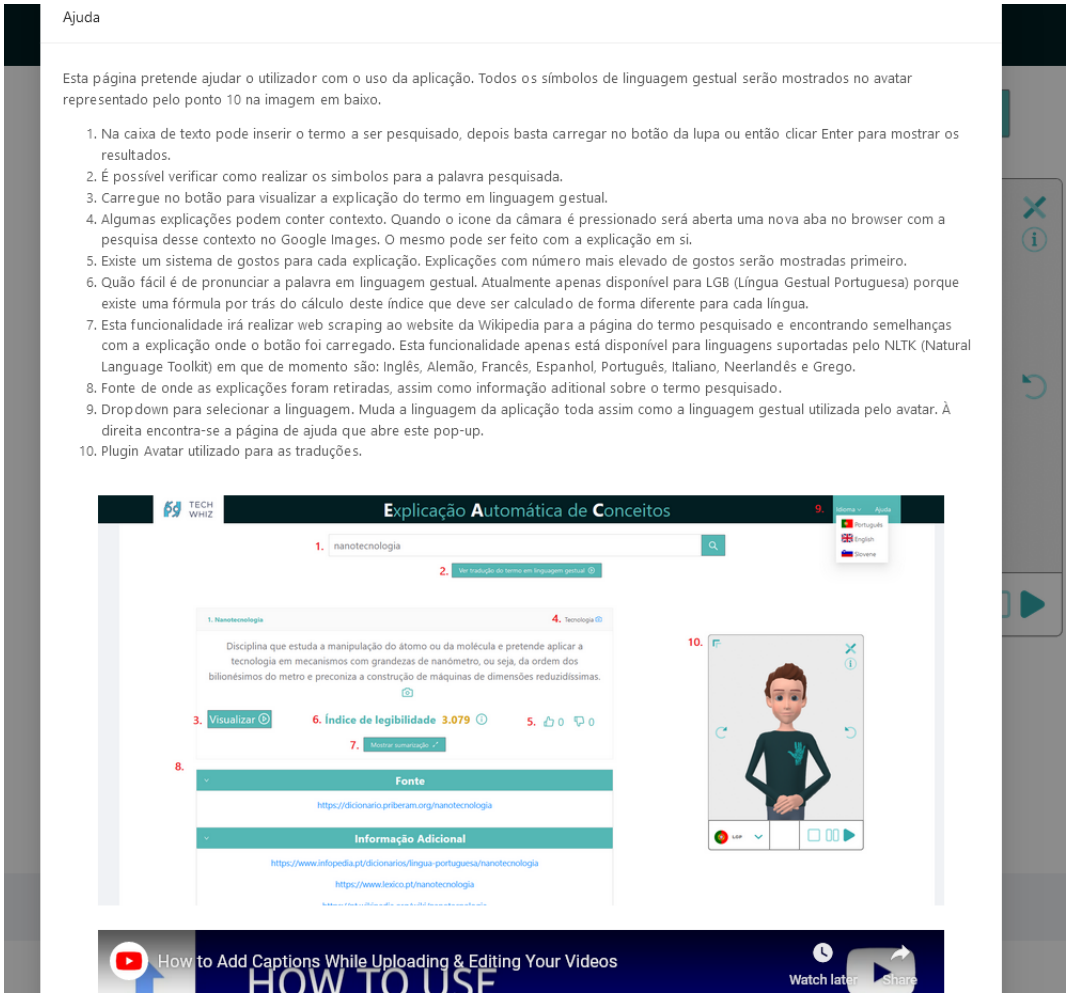


Figure 5.4: Help Page

The text has full translations and is intended to have video tutorials to further increase the user experience. For now, at the time of writing, it contains an image of the application after a search, and numbers associated with each functionality, every number having a description explaining the functionality. The help page also contains a placeholder video at the bottom, in which the language chosen by the user will affect the video's captions.

5.4 EAC DB - Database

The database is a new component of the solution, as previously mentioned in Section 5.2.2. As stated, the database chosen is Cassandra [3]. More details on Cassandra and why it was chosen over other databases are presented in Chapter 2.

The schema created to store all the information required for the solution is represented in Listing 5.12 below.

[3]https://cassandra.apache.org/

```
create keyspace eac WITH replication = {'class': 'SimpleStrategy', '
    replication_factor' : 1};

create table eac.wordsExpl(
    word text,
    explanation text,
    context text,
    source text,
    total_likes int,
    total_dislikes int,
    last_modified timestamp,
    PRIMARY KEY (word, explanation)
);

create table eac.audit(
    word text,
    explanation text,
    ipv4 text,
    country_code text,
    country_name text,
    location_state text,
    location_city text,
    action_submitted text,
    action_time timestamp,
    PRIMARY KEY (word, explanation, action_time)
);
```

Listing 5.12: Cassandra database schema.

The schema is composed of two tables, one for storing explanations and their ratings, and another for auditing.

The first table stores the input word, and explanation as a compound primary key. It also stores additional information, such as the number of positive and negative feedbacks, the source of the explanation (online dictionary from where it was scraped from) and a timestamp indicating when a given explanation was last added or edited.

The second table, much like the first, stores audit entries with a compound primary key composed of the input word, the explanation and the timestamp the rating was given. Apart from those three, it stores the public IPv4 of the device the user performed the rating from, the country the rating originated from, the state (if available) and city of the request, and lastly what rating was given, positive or negative.

5.5 Deployment

This section will provide more detail on how each component was deployed. The deployment strategy for the solution is the same as in the previous work [2].

DEI provided a machine that could be accessed remotely running Ubuntu 20.04.6 LTS[4] which is a Linux distribution. The project was cloned from its repository and then each component was deployed.

[4] https://releases.ubuntu.com/focal/

5.5.1 EAC API - Backend

The API was deployed by creating a Systemd service, as follows in Listing 5.13.

```
[Unit]
Description=Explanation API
After=network.target

[Service]
User=utechw
WorkingDirectory=/home/utechw/thesis-eac/code/api
ExecStart=/home/utechw/thesis-eac/code/api/venv311/bin/python app.py
Restart=always
RemainAfterExit=yes
Environment="PATH=/home/utechw/.local/bin:/usr/local/sbin:/usr/local/
    bin:/usr/sbin:/usr/bin"

[Install]
WantedBy=multi-user.target
```

Listing 5.13: EAC API Systemd Service Definition.

With this definition, Systemd will manage the API and automatically restart without manual intervention, if there are any issues.

5.5.2 EAC APP - Frontend

PM2[5] was used to deploy the web application. It is a daemon process manager that facilitates the deployment and management of Node.js applications.

After deploying successfully, the currently running applications can be visualized, as seen in Figure 5.5.

```
utechw@techwhiz:~/cassandra-docker$ sudo pm2 ls
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	eac-fe	default	5.2.2	fork	1183370	9m	0	online	0%	23.9mb	root	disabled

Figure 5.5: EAC APP in PM2's Running Applications List

PM2's Running Application List also provides additional information such as uptime and how many resources each application is consuming.

5.5.3 Cassandra Database

To deploy the Cassandra Database on the host DEI provided, Docker Compose[6] was utilized. It is fast and easy to setup, as well as easy to migrate as it is run in a container. The Docker Compose file definition is present in Listing 5.14.

[5] https://pm2.keymetrics.io/docs/usage/quick-start/
[6] https://docs.docker.com/compose/

```
version: '3.8'

services:
  cassandra:
    image: cassandra:3.11.14
    ports:
      - "9042:9042" # native protocol clients
    volumes:
      - ./data/cassandra:/var/lib/cassandra
```

Listing 5.14: Cassandra database's docker compose definition.

The only peculiarity is the presence of a volume entry in the definition file. This is to guarantee that, even if the Docker process is shut down, all entries in the database are not lost and, when the container is revived, the information is the same as before the Docker process was shut down.

5.5.4 Exposing the solution

To expose the components to be accessible via HTTP, a reverse proxy using NGINX[7] was set up. A reverse proxy is a server that stays in front of web servers and forwards requests to those web servers. In this case, NGINX will sit in front of EAC API and EAC APP, forwarding user requests to EAC APP and the responses to the user's web browser.

In order to improve security, HTTPS was enabled in the reverse proxy server. This was done by using Certbot[8]. Certbot is an open source tool that automatically uses Let's Encrypt[9] certificates to enable HTTPS protocol.

Listing 5.15 details the configuration of http in NGINX's server configuration file.

[7] https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/
[8] https://certbot.eff.org/instructions?ws=nginxos=ubuntufocaltab=standard
[9] https://letsencrypt.org/

```
http {
  server {
    listen 80;
    server_name techwhiz.dei.isep.ipp.pt;

    location / {
        proxy_pass http://localhost:3000;
      }
      location /search {
        proxy_pass http://localhost:5000;
      }
      location /summarizationNew {
        proxy_pass http://localhost:5000;
      }
      location /like-definition {
      proxy_pass http://localhost:5000;
      }
      location /dislike-definition {
        proxy_pass http://localhost:5000;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/techwhiz.dei.isep.ipp.pt/
    fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/techwhiz.dei.isep.ipp.
    pt/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
    Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
    Certbot
  }
}
```

Listing 5.15: NGINX HTTP Configuration.

There are multiple entries for each endpoint which allow every request to interact with the endpoints depicted above. This is to have some comfort and access the API without using the web application. In the future, only the first entry will be needed to block direct access to the API, further improving the solution's security and integrity.

Chapter 6

Evaluation

This chapter presents the tests performed to ensure the quality of the solution. It will also present the hypothesis of the project and the methodology used to evaluate it.

6.1 Software tests

There are multiple types tests, manual or automated, that can be used. However, for this solution only 3 of them were implemented:

- **Unit Tests** - Performed at low level, in the application's code.
- **Integration Tests** - Verify that different modules interact correctly with each other, for example, testing the interaction between the API and the Database.
- **End-to-end Tests** - Attempt to replicate the behaviour of a user, and verifies that multiple flows work as intended.

Since there is no CI/CD tool or process implemented for this solution, each suite of tests are run manually.

Unit tests are, ideally, manually run before a commit to the development branch. The same applies to the Integration tests, which are mainly run to ensure the connectivity between the API and the Cassandra Database is successful. Lastly, the End-to-end tests are performed by interacting with the deployed solution, covering various scenarios within each functionality supported.

6.2 Hypothesis

The hypothesis for this project remains the same as the previous work [2], which is to verify the validity of a solution that uses Text Mining[9] and Information Extraction[7] techniques to generate explanations for a given word and translate it to sign language, given the word does not exist in the sign language's lexicon.

6.3 Methodology

After defining the hypothesis mentioned in the previous section, evaluation methodologies will be used to test it and assess its validity. Given the hypothesis' direct connection to the core functionality of the developed solution, a survey methodology was employed that focuses on evaluating its usability, performance, and reliability from a user's experience perspective.

The survey mentioned is displayed in Appendix A. To confirm the validity of the solution, the overall rating from the survey's answers needs to be above average, following the scale in Table 6.1 below:

Table 6.1: Survey Scale.

Scale	**Description**
1	Completely Disagree
2	Disagree
3	Agree
4	Completely Agree

The target audience for this survey are LGP users. So the survey was intended to be distributed, with the help of the co-supervisor of this project, to students of an LGP school. However, that was not possible and was introduced to the stakeholders of this project. This setback originated from the absence of translations in the Avatar's Database.

Feedback was gathered by asking the audience to associate each of the following statements with a value in the previous scale:

- The information is presented in an organized matter.
- The graphics of the application are pleasant.
- The functionalities are well explained in the "Help" page.
- The functionalities are intuitive and easy to use.
- The explanations are accurate.
- The response time of the application is adequate.
- The ability to display explanation and context images are helpful in understanding their meaning.
- The explanation rating system is helpful.
- I would like to use this application in the future.
- I will recommend this application to my peers.

6.4 Survey Results Analysis

The answers to the survey are displayed in Table 6.2, followed by an analysis of the results.

Table 6.2: Survey Answers

Question	1 (%)	2 (%)	3 (%)	4 (%)	Average
Q1	5	10	55	30	3.1
Q2	0	5	80	15	3.1
Q3	0	10	60	30	3.2
Q4	0	15	50	35	3.2
Q5	0	10	80	10	3
Q6	0	25	70	5	2.8
Q7	0	10	55	35	3.25
Q8	5	30	50	15	2.75
Q9	0	15	55	30	3.15
Q10	0	10	45	45	3.35

Although there were few answers and the target audience was not as originally planned, the total average of all questions' average is 3.09 which is positive since it's slightly above the "Agree" scale.

The lowest average is regarding the utility of the explanation rating system, followed by the application's response time. This concludes that there's a clear need in improving the response time of the application and adding some enhanced purpose to the rating system, or remove it altogether in the future when a similar survey to this is presented to deaf people and the answers remain in the negative side.

However, the highest averages are regarding recommending this application to other users and the utility of presenting images to further understand an explanation. This observation leads to the conclusion that this functionality should remain and be improved in the future, for example, presenting images in the app directly instead of opening a new tab in the browser.

Chapter 7

Conclusion

In this final section the conclusion for this project is presented. Firstly, the development itself is mentioned in general as well as how it progressed. Secondly, the achievements from the defined objectives are detailed, followed by the future developments that are divided in developments that were not able to be accomplished in this project and development ideas for this project and context for future developers. Lastly, a final appreciation is presented, including the author's thoughts and opinions regarding the project.

7.1 Developments

As stated in previous chapters, the main focus of this project is developing a solution that helps explaining a word or concept and translate that explanation to sign language. And to achieve that core functionality the solution must use Text Mining and Information Extraction techniques.

The state of the art was presented, detailing concepts crucial to this solution - Sign Language and the techniques used in the solution. It was also detailed more on Online Dictionaries since they are the main source of explanations for the solution. Lastly, the technologies evaluated for usage were contextualized.

After that, a value analysis was done where a value proposition and business model CANVAS was presented. Followed by using AHP to aid in the decision between the database to use in the solution, which resulted in, according to the solution's needs, Cassandra was the best option.

Followed by the value analysis the design process started. The functional and non-functional requirements were established according to the solution's core functionalities and expectations. Next, the solution's architecture was defined illustrating the logic, deployment and process view. Finally, sequence diagrams for each use case were presented and detailed.

Then, with the architecture defined and requirements designed, the functionalities were implemented and documented. Little to no setbacks were experienced during implementation due to good communication with stakeholders and great contextualization before the solution analysis started.

As soon as the solution was more stable, tests were introduced. At the same time, a survey was created and distributed by the supervisor of the project. Unfortunately, the target audience of this survey changed from LGP users to stakeholders of the project due to some issues with the Avatar's Database. To evaluate the solution, according to the hypothesis defined, the answers to the survey were analysed. It is concluded that the solution is

satisfactory, however, as all things, there is room for improvement. Thanks to the feedback gathered, there is a clear path to the future work of this project.

7.2 Achievements

Since the main goal was somewhat tackled in the previous work[2], this project aimed to, above all, improve and fix every functionality the previous work had, as well as introduce new ones. The main functionality was improved and all the defined objectives in Chapter 1 were achieved.

The communication between services was improved by making the payloads open for extension in case new functionalities are added, as well as being human readable which helps future work by other developers.

The scraping algorithm's performance was improved by reducing complexity as well as being documented in the code to help future developers change it since changes in scraping are to be expected because online dictionaries often change their user interface. The algorithm to present a summarization was also improved by reducing file access operations and treating everything in memory for a faster response.

The rating system was introduced to be able to distinguish between good and bad explanations depending on the context they are queried for.

A database was introduced in the solution to store the rating previously mentioned paired with the explanations themselves. In case of inability to access the online dictionary for any reason, the database is there to provide the explanations improving the quality of service and availability.

The audit system with geolocation was introduced as well. The audit is stored in the database and currenty there isn't any functionality attached to it, but it can provide insight on what explanation is more suitable according to the country or area of the country a user is in.

7.3 Future Work

To improve the solution's value there is work that can be done in the future. For example, using the audit with geolocation implementation to extract some valuable data relating explanation rating with the region of the user.

There is also the need for a more robust solution. Currently everything is hosted in a single machine in DEI's servers. These servers have regular maintenance which requires manual intervention to keep the solution up. A dedicated host for each service and component and a strategy to make them automatically recover.

Along with more robustness, CI/CD would be a good addition and help with keeping the solution up due to regular deployments. Paired with this CI/CD process, a well established testing phase to ensure the quality of the solution with automatic tests would bring a lot of value and trust to the solution.

Lastly, as analysed via the survey results, there is a clear need to improve the solution's response times.

7.4 Final Appreciation

This project appeared to be interesting and met the author's expectations.

It provided knowledge in languages, frameworks and tools never used before. It also made the author improve the writing skills and time management.

In sum, the appreciation for the project as a whole is positive and it leaves a feeling of accomplishment and happiness due to the solution implemented directly helping the deaf community.

Bibliography

[1] *Deafness*. url: https://www.who.int/news-room/facts-in-pictures/detail/deafness (visited on 02/11/2023).

[2] André Filipe Teixeira Dias. "Explicação automática de conceitos". Accepted: 2020-12-14T11:47:26Z. masterThesis. 2020. url: https://recipp.ipp.pt/handle/10400.22/16560 (visited on 01/15/2023).

[3] Joseph D Novak. *Learning, creating, and using knowledge: Concept maps as facilitative tools in schools and corporations*. Routledge, 2010.

[4] *Our Work*. WFD. url: https://wfdeaf.org/our-work/ (visited on 02/19/2023).

[5] Francois Grosjean. "The Right of the Deaf Child to Grow Up Bilingual". In: *Sign Language Studies* 1.2 (2001), pp. 110–114. issn: 1533-6263. doi: 10.1353/sls.2001.0003. url: http://muse.jhu.edu/content/crossref/journals/sign_language_studies/v001/1.2grosjean.html (visited on 02/19/2023).

[6] Harry Knoors and Marc Marschark. "Language Development". In: *Teaching Deaf Learners: Psychological and Developmental Foundations*. Ed. by Harry Knoors and Marc Marschark. Oxford University Press, Jan. 22, 2014, p. 0. isbn: 978-0-19-979202-3. doi: 10.1093/acprof:oso/9780199792023.003.0004. url: https://doi.org/10.1093/acprof:oso/9780199792023.003.0004 (visited on 02/19/2023).

[7] Christopher Manning, Prabhakar Raghavan, and Hinrich Schuetze. "Introduction to Information Retrieval". In: (2009).

[8] Shubham Chatterjee. "Answering Topical Information Needs Using Neural Entity-Oriented Information Retrieval and Extraction". In: *ACM SIGIR Forum* 56.2 (Jan. 31, 2023), 20:1–20:2. issn: 0163-5840. doi: 10.1145/3582900.3582926. url: https://doi.org/10.1145/3582900.3582926 (visited on 02/14/2023).

[9] *Marti Hearst: What Is Text Mining?* url: https://people.ischool.berkeley.edu/~hearst/text-mining.html (visited on 02/25/2023).

[10] Matt Mulins. "Information extraction in text mining". In: (2008). Publisher: Western Washington University.

[11] Weiguo Fan et al. "Tapping the power of text mining". In: *Communications of the ACM* 49.9 (2006). Publisher: ACM New York, NY, USA, pp. 76–82.

[12] Vishal Gupta, Gurpreet S Lehal, et al. "A survey of text mining techniques and applications". In: *Journal of emerging technologies in web intelligence* 1.1 (2009), pp. 60–76.

[13] Yuen-Hsien Tseng, Chi-Jen Lin, and Yu-I Lin. "Text mining techniques for patent analysis". In: *Information processing & management* 43.5 (2007). Publisher: Elsevier, pp. 1216–1247.

[14] Falguni N Patel and Neha R Soni. "Text mining: A Brief survey". In: *International Journal of Advanced Computer Research* 2.4 (2012). Publisher: Citeseer, p. 243.

[15] *Dictionary by Merriam-Webster: America's most-trusted online dictionary*. url: https://www.merriam-webster.com/ (visited on 02/15/2023).

[16] *Home : Oxford English Dictionary*. url: https://www.oed.com/ (visited on 02/15/2023).

[17] Priberam Informática S.A. *Dicionário Priberam Online de Português Contemporâneo*. Dicionário Priberam. url: https://dicionario.priberam.org/ (visited on 02/15/2023).

[18] Infopédia. *infopedia.pt - Dicionários Porto Editora*. infopedia.pt - Porto Editora. url: https://www.infopedia.pt/dicionarios/lingua-portuguesa (visited on 02/15/2023).

[19] *Dictionary.com | Meanings & Definitions of English Words*. Dictionary.com. url: https://www.dictionary.com/ (visited on 02/15/2023).

[20] *Urban Dictionary*. Urban Dictionary. url: https://www.urbandictionary.com/ (visited on 02/15/2023).

[21] Bonnie JF Meyer. "Text coherence and readability". In: *Topics in Language Disorders* 23.3 (2003). Publisher: LWW, pp. 204–224.

[22] Luís Miguel Cardoso Lopes Correia. "Evaluation Metrics for Text and Creation of Writing Tool for Sports Journalism". In: (2020).

[23] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[24] *Scrapy | A Fast and Powerful Scraping and Web Crawling Framework*. url: https://scrapy.org/ (visited on 02/19/2023).

[25] Chandni Saini and Vinay Arora. "Information retrieval in web crawling: A survey". In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2016, pp. 2635–2643.

[26] *Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation*. url: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (visited on 02/19/2023).

[27] *Installing a parser*. Publication Title: Beautiful Soup Documentation. url: https://www.crummy.com/software/BeautifulSoup/bs4/doc/#installing-a-parser.

[28] *lxml - Processing XML and HTML with Python*. url: https://lxml.de/ (visited on 02/19/2023).

[29] *Overview — html5lib 1.2-dev documentation*. url: https://html5lib.readthedocs.io/en/latest/ (visited on 02/19/2023).

[30] *Welcome to Flask — Flask Documentation (2.2.x)*. url: https://flask.palletsprojects.com/en/2.2.x/ (visited on 02/19/2023).

[31] *Django*. Django Project. url: https://www.djangoproject.com/ (visited on 02/19/2023).

[32] *Welcome to Pyramid, a Python Web Framework*. url: https://trypyramid.com/ (visited on 02/19/2023).

[33] *Bottle: Python Web Framework — Bottle 0.13-dev documentation*. url: https://bottlepy.org/docs/dev/ (visited on 02/19/2023).

[34] *Define and Access the Database — Flask Documentation (2.2.x)*. url: https://flask.palletsprojects.com/en/2.2.x/tutorial/database/ (visited on 02/19/2023).

[35] *Getting Started – React*. url: https://reactjs.org/docs/getting-started.html (visited on 02/19/2023).

[36] *Most used web frameworks among developers 2022*. Statista. url: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (visited on 02/19/2023).

[37] Thomas M. Connolly and Carolyn E. Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson, 2014. 135-136.

[38] *Apache Cassandra | Apache Cassandra Documentation*. url: https://cassandra.apache.org/_/index.html (visited on 02/19/2023).

[39] *Cassandra: The Definitive Guide, 3rd Edition [Book]*. ISBN: 9781098115166. url: https://www.oreilly.com/library/view/cassandra-the-definitive/9781098115159/ (visited on 02/19/2023).
[40] *Docker Deep Dive [Book]*. ISBN: 9781800565135. url: https://www.oreilly.com/library/view/docker-deep-dive/9781800565135/ (visited on 02/19/2023).
[41] *Docker Hub Container Image Library | App Containerization*. url: https://hub.docker.com/ (visited on 02/19/2023).
[42] Aater Suleman. *Council Post: Docker And Kubernetes: Furthering The Goals Of DevOps Automation*. Forbes. Section: Innovation. url: https://www.forbes.com/sites/forbestechcouncil/2018/10/10/docker-and-kubernetes-furthering-the-goals-of-devops-automation/ (visited on 02/19/2023).
[43] Chris Tankersley. *Docker for Developers: php[architect] print edition*. php[architect], July 28, 2016. 66 pp. isbn: 978-1-940111-36-0.
[44] *Geolocation DB - Geographic location By IP Address*. url: https://geolocation-db.com/ (visited on 01/15/2023).
[45] *PM2 - Home*. url: https://pm2.keymetrics.io/ (visited on 02/19/2023).
[46] *PM2 - Monitoring*. url: https://pm2.keymetrics.io/docs/usage/monitoring/ (visited on 02/19/2023).
[47] *Advanced Load Balancer, Web Server, & Reverse Proxy*. NGINX. url: https://www.nginx.com/ (visited on 02/19/2023).
[48] *NGINX Reverse Proxy | NGINX Plus*. url: https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/ (visited on 02/19/2023).
[49] *Usage Statistics and Market Share of Nginx, February 2023*. url: https://w3techs.com/technologies/details/ws-nginx (visited on 02/19/2023).
[50] Thanapon Noraset et al. "Definition modeling: Learning to define word embeddings in natural language". In: *arXiv preprint arXiv:1612.00394* (2016).
[51] Ke Ni and William Yang Wang. "Learning to explain non-standard english words and phrases". In: *arXiv preprint arXiv:1709.09254* (2017).
[52] *Explore*. Twitter. url: https://twitter.com/explore (visited on 02/19/2023).
[53] Georgiana Dinu and Marco Baroni. "How to make words with vectors: Phrase generation in distributional semantics". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2014, pp. 624–633.
[54] William Dolan, Lucy Vanderwende, and Stephen D Richardson. "Automatically deriving structured knowledge bases from on-line dictionaries". In: *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*. Pacific Association for Computational Linguistics Vancouver, 1993, pp. 5–14.
[55] *Longman Dictionary of Contemporary English | LDOCE*. url: https://www.ldoceonline.com/ (visited on 02/19/2023).
[56] Alan Akbik and Jürgen Broß. "Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns". In: *www workshop*. Vol. 48. 2009.
[57] *Main Page*. In: *Wikipedia, the free encyclopedia*. Page Version ID: 1114291180. Oct. 5, 2022. url: https://en.wikipedia.org/w/index.php?title=Main_Page&oldid=1114291180 (visited on 02/19/2023).
[58] Atin Das et al. "Neural net model for featured word extraction". In: *Unifying Themes in Complex Systems IV*. Springer, 2008, pp. 353–361.
[59] Amirata Ghorbani et al. "Towards Automatic Concept-based Explanations". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.,

2019. url: https://proceedings.neurips.cc/paper/2019/hash/77d2afcb31f6493e350fca61764efb9Abstract.html (visited on 10/06/2023).

[60] Liana Ermakova et al. "Automatic Simplification of Scientific Texts: SimpleText Lab at CLEF-2022". In: *Advances in Information Retrieval*. Ed. by Matthias Hagen et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 364–373. isbn: 978-3-030-99739-7. doi: 10.1007/978-3-030-99739-7_46.

[61] Peter Koen et al. "Providing clarity and a common language to the "fuzzy front end"". In: *Research-Technology Management* 44.2 (2001). Publisher: Taylor & Francis, pp. 46–55.

[62] Alexander Osterwalder et al. *Value Proposition Design: How to Create Products and Services Customers Want*. 1st edition. Hoboken: Wiley, Oct. 20, 2014. 320 pp. isbn: 978-1-118-96805-5.

[63] Alexander Osterwalder and Yves Pigneur. *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons, 2010.

[64] Roseanna W Saaty. "The analytic hierarchy process—what it is and how it is used". In: *Mathematical modelling* 9.3 (1987). Publisher: Elsevier, pp. 161–176.

[65] Peter Eeles. "Capturing Architectural Requirements". In: (Nov. 1, 2001).

[66] *Fran/iskanje*. Fran. url: https://fran.si/ (visited on 07/29/2023).

Appendix A

Solution Survey

4. Os gráficos da aplicação são agradáveis. *

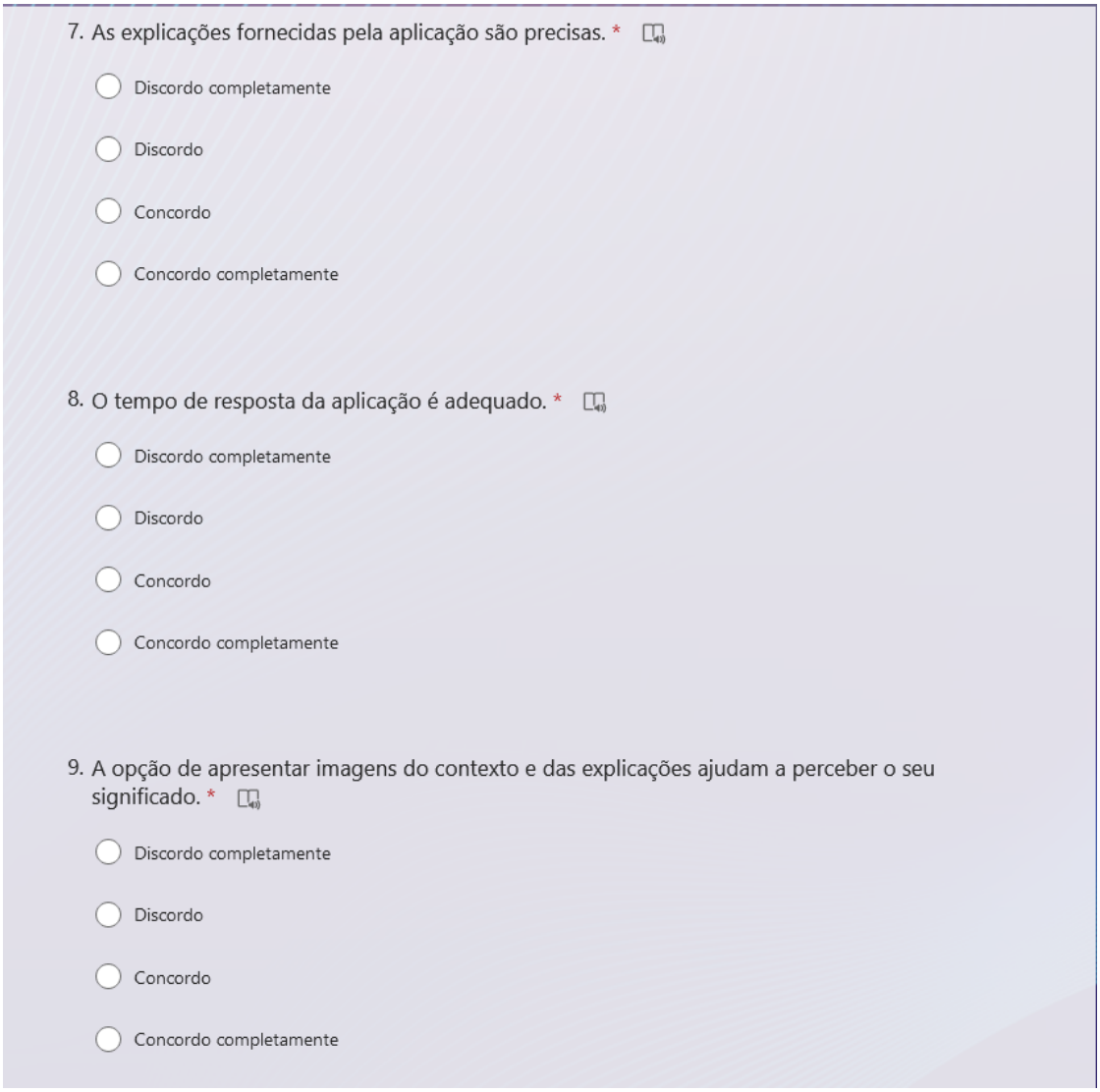
- ○ Discordo completamente
- ○ Discordo
- ○ Concordo
- ○ Concordo completamente

5. As funcionalidades são bem explicadas na página de "Ajuda". *

- ○ Discordo completamente
- ○ Discordo
- ○ Concordo
- ○ Concordo completamente

6. A aplicação é intuitiva e fácil de utilizar. *

- ○ Discordo completamente
- ○ Discordo
- ○ Concordo
- ○ Concordo completamente



7. As explicações fornecidas pela aplicação são precisas. *

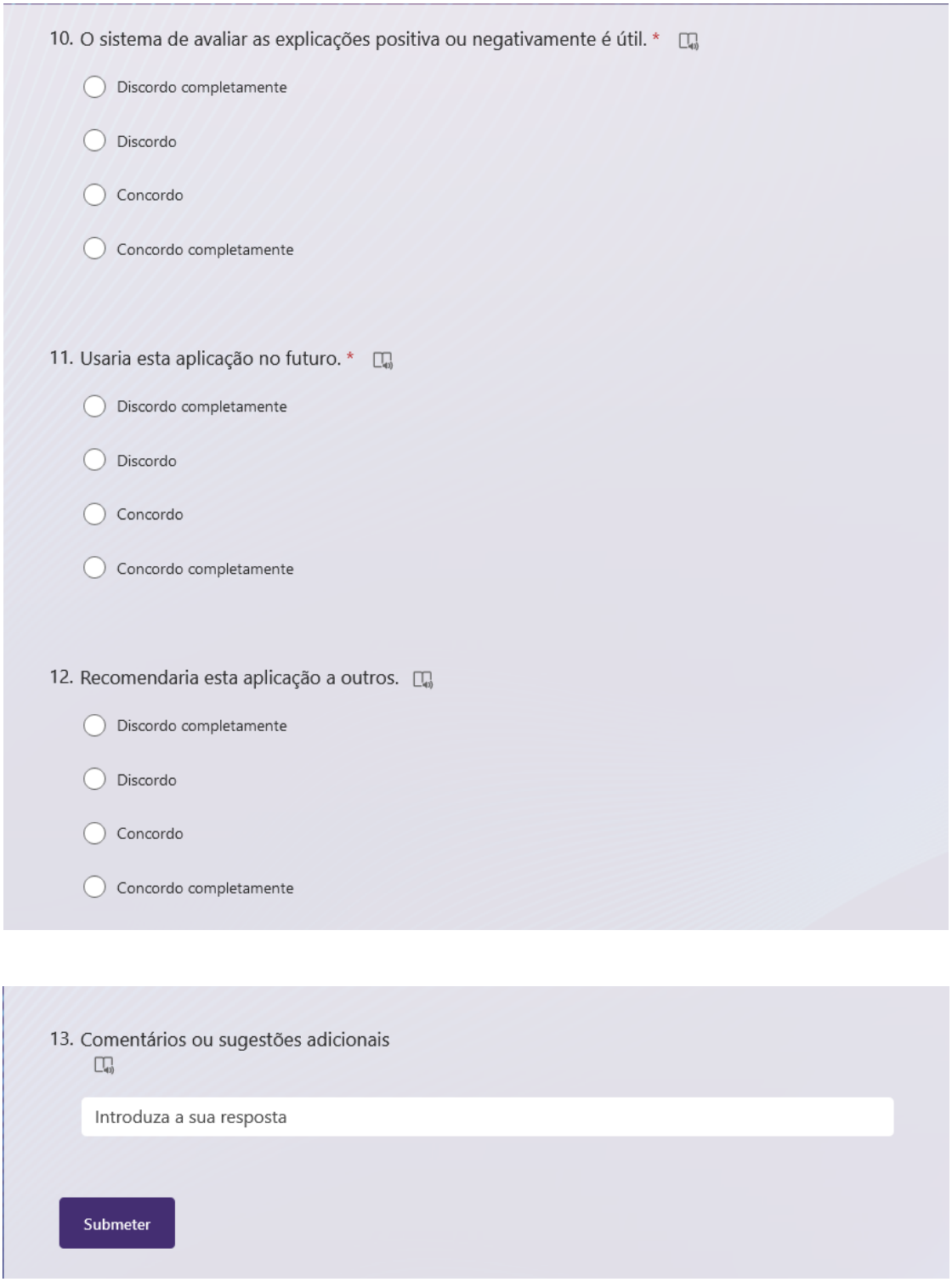
- Discordo completamente
- Discordo
- Concordo
- Concordo completamente

8. O tempo de resposta da aplicação é adequado. *

- Discordo completamente
- Discordo
- Concordo
- Concordo completamente

9. A opção de apresentar imagens do contexto e das explicações ajudam a perceber o seu significado. *

- Discordo completamente
- Discordo
- Concordo
- Concordo completamente


10. O sistema de avaliar as explicações positiva ou negativamente é útil. *
Discordo completamente
Discordo
Concordo
Concordo completamente
11. Usaria esta aplicação no futuro. *
Discordo completamente
Discordo
Concordo
Concordo completamente
12. Recomendaria esta aplicação a outros.
Discordo completamente
Discordo
Concordo
Concordo completamente
13. Comentários ou sugestões adicionais
Introduza a sua resposta
Submeter