UNIVERSIDADE DA BEIRA INTERIOR
Faculdade de Engenharia

A Micro-Interaction Tool for Online Text Analysis

Rita Pessoa Correia

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Bruno M. C. Silva
Co-orientador: Prof. Doutor João Correia

Covilhã, outubro de 2021

Dedication

Dedicated to my family, especially to my parents, brother, and grandmother who always believed in my potential to keep overcoming myself.

Acknowledgements

Reaching the end of this journey, I would like to express my sincerest gratitude for those that made this possible.
Firstly, I would like to thank all the staff from the IT department of the University of Beira Interior for giving me the chance to learn and to improve myself personally and academically. This institution really taught me a lot of values, objectives and gave me the pillars to keep growing.
Secondly, as this work was initially supported by National Founding from the FCT (*Fundação para a Ciência e a Tecnologia*), I would like to thank Labcom for the opportunity to work in the Re/media.Lab project. It was my first job in my area of study and it really helped me to understand better how the market works and how to overcome my difficulties. I would like to make a special thanks to Prof. Dr. Pedro Jerónimo and Prof. Dr. João Correia for having followed my work more closely and supported me in every moment of this journey.
Thirdly, I would like to express my gratitude to my dissertation advisor Prof. Dr. Bruno Silva for believing in me and in my potential since day one, for his continuous guidance, patience, support, and expertise, which encouraged me so much during the whole process of this project. Also, to Prof. Dr. Nuno Garcia, for being constantly available to help me in any situation of my entire academic path and, especially, in this dissertation and its future perspectives.
To my parents, I express my greatest gratitude. Their support was unconditional, not only to make this chapter of my life possible but for every other moment. I want to give them a big thank you for not letting me forget who I am and what I intend to do with my life. They taught me that I am the only one responsible for my future according to my own choices, and regardless of those, they are going to be there to support me, no matter what.
Also, I want to thank my boyfriend, Rui Salgado, for teaching me the value of little things and tiny achievements. A thank you isn't enough for all the encouragement, strength, and hope that he gave me during this path. Thank you for just being there and motivate me to be the best version of myself, even when I wanted to give it all up. You made me want to live again and believe I was worthy of being happy, and for that, I'll be eternally grateful. Thank you for everything, I love you.
Last, but not least, I want to thank all of my friends who were always with me and supporting me on this path. A special appreciation to Andreia Cardona, João Clarinha, João Pacheco, Júlio Mendonça, Rita Pinho and Tiago Sena who always gave me the best advices. Thank you from the bottom of my heart for being the best friends someone could possibly have.

Resumo

Os dispositivos móveis permitem que os utilizadores permaneçam conectados ao Mundo de forma ubíqua, criando novos contextos para o uso dos mídia. Diante as mudanças estruturais no mercado jornalístico, as organizações de mídia estão a tentar liderar esta transição digital, (re)ganhando a atenção do público [WS15]. Esta evolução digital pode trazer tanto muitas vantagens ou abrir a porta para o jornalismo apressado, como a publicação de notícias falsas e conteúdo malicioso, que pode ter efeitos críticos sobre os indivíduos e a sociedade como um todo. Por esse motivo, está a tornar-se cada vez mais importante verificar os factos das fontes de informação.

A desinformação é informação incorreta ou enganosa, que pode levar à distorção das opiniões das pessoas sobre diversos assuntos e a consequências indesejadas. Portanto, a verificação de factos com informações de fontes confiáveis é talvez a melhor maneira de combater a disseminação de informações incorretas. É portanto muito importante utilizar fontes confiáveis para verificar os factos, caso contrário, corremos o risco de perpetuar o ciclo [Ohi].

Para ajudar a combater este problema global, podemos utilizar a interação dos utilizadores de Internet com os produtores/jornalistas de conteúdo, para que esses utilizadores possam interagir com o conteúdo da Web, validando, comentando ou expressando emoções sobre este, de forma a diminuir a percentagem de conteúdo falso, malicioso ou questionável, bem como simultaneamente criar um perfil desses mesmos utilizadores e produtores de conteúdo, através da aplicação de regras de reputação. Com esta estratégia, os produtores de conteúdo online podem obter uma interação dinâmica e *feedback* do público sobre o conteúdo publicado, para que possam verificar os factos e ter um maior grau de veracidade.

Esta dissertação de mestrado apresenta uma ferramenta *Web* que permite aos utilizadores realizar uma verificação rápida de factos, interagindo com os mídia responsáveis por uma determinada notícia ou texto. Este trabalho começa por apresentar um estudo sobre as principais ferramentas e técnicas que estão a ser utilizadas no jornalismo para a verificação de factos. Em seguida, descreve detalhadamente o processo de implementação da ferramenta desenvolvida, que consiste numa extensão *Web* para auxiliar neste domínio de verificação de factos. Por fim, a dissertação apresenta alguns testes que foram realizados para avaliar a viabilidade da solução.

Palavras-chave

notícias falsas, verificação de factos, extensões do Chrome, jornalismo, dispositivos móveis

Abstract

Mobile devices allow users to remain connected to the World in a ubiquitous way, creating new contexts of media use. Considering the structural changes in the journalistic market, media organizations are trying to lead this digital transition, (re)gaining the attention of the public [WS15]. This digital evolution can bring either many advantages or open the door to rushed journalism, such as the publication of fake news and malicious content, which can have critical effects on both individuals and society as a whole. For this reason, it's becoming really important to fact-check the sources of information.

Misinformation is incorrect or misleading information, which can lead to the distortion of people's opinions on several matters and unintended consequences. Thus, fact-checking claims with reliable information from credible sources is perhaps the best way to fight the spread of misinformation. By double-checking a claim, you can verify whether or not it's true. However, it's important to use verifiable and reputable sources to fact-check that information, otherwise, you risk perpetuating the cycle [Ohi].

In order to help to fight this global issue, we can use the interaction from Internet users with the content producers/journalists, so those users can interact with Web content, validating, commenting, or expressing emotions about it to decrease the percentage of false, malicious or questionable content, as well as simultaneously create a profile of these same users and content producers, through the application of reputation rules. With this strategy, online content producers can get dynamic interaction and feedback from the public about the published content, so they can fact-check it and have a greater degree of truthfulness.

This Master's dissertation presents a Web tool that enables users to perform a fast fact-checking, interacting with the media responsible for the news or text. This work, starts by presenting a study on the main tools and techniques that are being used in journalism for fact-check information. Then, it describes in detail the implementation process of the developed tool, that consists on a Web extension to help in this fact-checking domain. Finally, the dissertation presents an assessment and tests that were conducted to evaluate the feasibility of the solution.

Keywords

fake news, fact-checking, chrome extensions, journalism, mobile devices

Contents

List of Figures

List of Tables

Listings

Acronyms List

AI	Artificial Intelligence
API	Application Programming Interface
CSS	Cascading Style Sheets
DB	Database
ERD	Entity-Relationship Diagram
FR	Functional Requirements
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IT	Information Technology
ML	Machine Learning
NFR	Non-Functional Requirements
OS	Operating System
PF	Pseudo-Feedback
PHP	PHP: Hypertext Preprocessor
PWA	Progressive Web Apps
UI	User Interface
URIs	Uniform Resource Identifiers
URL	Uniform Resource Locator
UX	User Experience

Chapter 1

Introduction

1.1 Contextualization

This Master's dissertation was developed in the scope of the Curricular Unit of Dissertation, present in the second year of the Master's Degree in Computer Engineering, in the University of Beira Interior. The project was mainly developed in the NetGNA laboratory of the *Instituto de Telecomunicações* Research group, and it emerged from a research grant with the reference *Bolsa Licenciado_Remedia.Lab_Inf*, which took place in Labcom and was financed by *Fundação para a Ciência e a Tecnologia*.

The circulation of fake news and malicious content is not new, but its visibility and impact increased with the emergence of social networks and online social media. There is so much misinformation spread around, that people no longer know what to believe. The line between journalism and other content has blurred, making it really important for all writers, regardless of their platform, to check their facts.

Unlike the conventional verification process, which requires checking information before it is published, fact-checking is dedicated to post-hoc checking, or in other words, verifying statements and alleged facts after they have been published.

Many techniques and mechanisms have been studied for fake news detection and fact-checking, most of them using the synergy between Machine Learning (ML) and Artificial Intelligence (AI) algorithms with Human sensibility, sense and emotion.

1.2 Problem Statement

Textual compositions are one of the most common forms of sharing content in the Web. However, the interaction with non-textual content, such as multimedia and hybrid content, represents a more challenging task, that is often done through ranking evaluations such as thumbs up or thumbs down. Those interactions can get really elaborate as the content complexity grows.

With the constant increase of new technologies, also more misinformation, fake news and questionable content have been published in many different way and forms all over the Internet. This is undoubtedly a current problem because the public's knowledge and the respective opinions on several global matters are being easily persuaded and distorted. For this reason it's important for the journalists to fact-check the information by analyzing sources, claims and rating them as true or false, in order to provide accurate content.

1.3 Objectives

The main goal of this Master's dissertation project is to construct and evaluate a tool that allows the users to interact at a micro-level with the online text in a regional Journal, sending feedback and identifying fake and questionable content, so those content producers can receive and fact-check that information properly according to a set of rules and approaches.
In order to achieve the objective of this project, several partial objectives were defined:

1. Technological background study;
2. Detailed review of the related work;
3. Requirements analysis;
4. System design and implementation;
5. Testing.

1.4 Main Contributions

One of the main contributions of this Master's Dissertation was the writing of an article named ”Fact-checking going mobile: new tools and methods to help journalism” in the scope of the *Jornalismo e Dispositivos Móveis* (JDM) international congress, held at the University of Beira Interior. The article presents several tools and techniques that are being used in mobile journalism in order to verify facts. Some reflections on future tools and techniques are also approached.
Moreover, a patent application of the project idea was also submitted and ongoing.

1.5 Document Organization

In order to reflect the work that has been done, this document is structured as follows:

1. The first chapter – **Introduction** – presents the contextualization of the project, the problem statement of it, its objectives, the main contributions and the respective organization of the document.
2. The second chapter – **Related Work** – analyzes several Web technologies and concepts in the scope of the project, as well as some already existent similar projects.
3. The third chapter – **Conceptual Design** – describes the project functional and non-functional requirements, the use-cases, the Database (DB) schema, the system architecture and the used technologies.
4. The fourth chapter – **Implementation** – walks through all the development and implementation processes of both the extension and the platform.

5. The fifth chapter – **Tests** – shows all the tests performed on the system and the respective error messages.
6. The sixth chapter – **Conclusion** – refers the main conclusions of this Master's dissertation project and the future work.

Chapter 2

Related Work

2.1 Introduction

This chapter intends to review the main concepts and characteristics of some of the most important Web enabling technologies in the frontend development field, as well as the challenges and some already existing and identified solutions in the scope of journalism for mobile devices integrated with the detection of fake news and fact-checking mechanisms.
This chapter is organized into the following sections:

1. The first section – **Introduction** – intends to make a short introduction about what will be discussed in this chapter.

2. The second section – **Web Enabling Technologies** – details some of the enabling frontend technologies and their main characteristics and concepts, for a better understanding of the developed tool.

3. The third section – **Fake News and Fact Checking on the Web: The Road So Far** – describes some existing projects for detecting fake news and for fact-checking purposes with the interaction between ML and AI algorithms and Human sensibility.

4. The fourth section – **Conclusion** – refers the main conclusions of the chapter.

2.2 Web Enabling Technologies

2.2.1 Native, Web and Progressive Web Apps

Mobile applications (or apps) have been reaching a huge success in the mobile market. This opportunity attracted a lot of interested companies to have their optimized mobile apps, especially for the main mobile operating systems, such as Android and iOS. However, these developments are usually expensive and require more development time when building natively for each mobile Operating System (OS) [FB18].
New improvements on the Web technologies, allowed more features and capabilities that were only possible on apps developed in a native form. This started new possibilities on consolidating most developments on Web Apps, which are apps that run on Web browsers [FB18].
As we move into a ”mobile-first world”, the question of whether to develop Native or Web Apps is increasing. As so, companies and developers have three different possibilities:

build Native, Web, or Progressive Web Applications. The answer will always depend on the developer entity priorities and many factors such as the speed of development, the objectives and the budget for the development of the app, and the necessary features.

2.2.1.1 Native Apps

A Native or Mobile App is built specifically for an OS and is installed directly on the phone's storage, for example, Android apps are written in Java and iOS apps are written in Objective-C [Gig19]. These are the apps that are often pre-packaged into the device itself and they only work on the platform for which they are designed [Dan]. As so, Native Apps can take full advantage of the device features — they can work much faster by harnessing the power of the processor and can access specific hardware like GPS. [Gra].
Thus, we can say that the main advantage of Native applications is that they offer the highest level of functionality, the most fluid performance, and work offline. Since the app is written in the native language of the device, it can access all of its utilities and features such as the microphone, camera, contacts lists, etc, making a faster, more reliable, and more enjoyable User Experience (UX)[Dan]. Also, they can incorporate gestures and use the device's notification system [Gra].
On the other hand, the big disadvantage of Native Apps is that if we want them to offer experience on multiple platforms and OS's, we will have to develop the app for each platform and get the approval of the specific Marketplace (e.g. App Store). This increases the cost of development and updating [Dan].

2.2.1.2 Web Apps

Web Apps are websites with interactivity that feels similar to a mobile app. They are built to work and function on any OS, so they are not directly installed onto the phone's storage, being thus browser-based. Web applications run in multiple browsers — such as Safari or Chrome — and are written in HTML5 and/or JavaScript [Gig19].
The users first access them as they would access any Web page: they navigate to a specific Uniform Resource Locator (URL) and then have the option of "installing" them on their home screen by creating a bookmark to that page. Web Apps require Internet access and their operating speeds are dependent on the quality of smartphone signal or the speed of the Wi-Fi broadband they are connected to [Gra].
The major advantage of developing a Web App is that it is the most cost-effective option. As it works on the browser of the device, it offers cross-platform support without having to build the app for multiple OS's [Dan]. Also, these applications have easy development and maintenance, not being necessary the specific Marketplace approval.
The downside is that Web Apps can be slower, less intuitive, and inaccessible through app stores. Additionally, the users won't have the Web App's icon automatically downloaded to their home screens, so they won't be constantly reminded to use the app [Gig19]. Besides, Web applications cannot access all of the features of the device. This means that they can't perform the same range of complex functions that would be available with a Native App [Dan].

2.2.1.3 Progressive Web Apps

PWA are Web applications with responsiveness and ability to provide some content of the Web application even when the user is offline. Thus, they are browser-driven applications that focus on the progressive and independent development of the platform that will run the application [FB18].
PWA have been designed so they are capable, reliable, and installable. These three pillars transform them into an experience that feels like a platform-specific application:

- **Capable**: while some capabilities are still out of the Web's reach, new and upcoming Application Programming Interface (API) are looking to change that, expanding what the Web can do with features like file system access, media controls, app badging, and full clipboard support. All of these capabilities are built with the Web's secure, user-centric permission model, ensuring that going to a website is never a scary proposition for users;
- **Reliable**: a reliable PWA feels fast and dependable regardless of the network;
- **Installable**: a PWA run in a standalone window instead of a browser tab, being launchable from the user's home screen, dock, taskbar, or shelf [Sam20].

In figure 2.1, we can see the capabilities versus the reach of Native Apps, Web Apps, and PWA. As we can see, platform-specific apps represent the best of capabilities whereas Web Apps represent the best of reach. As so, PWA are built and enhanced with modern APIs to deliver improved capabilities, reliability, and installability while reaching anyone, anywhere, on any device with a single codebase [Sam20].

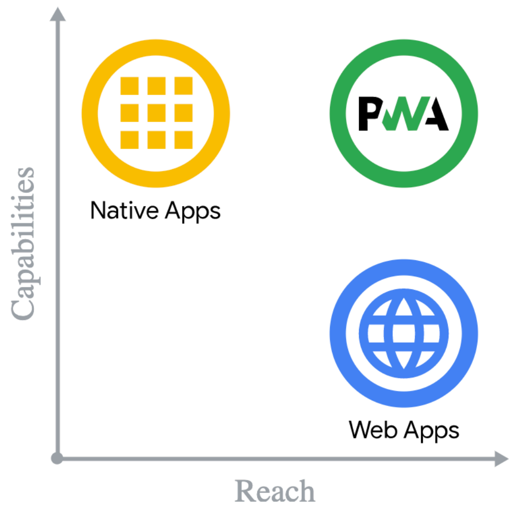


Figure 2.1: Capabilities vs. reach of platform-specific apps, Web App, and Progressive Web Apps [Sam20].

Technically, a PWA does not use a specific technology. It is not a new development framework, just as it is not a new programming language. In fact, PWA are a set of strategies, techniques, and APIs that provide the user with a UX very similar to the experience in a Native application [FB18].

In a streaming media context, a PWA enables a universal video experience in one codebase that performs just as well on mobile devices as it does on more powerful desktop computers. Some current PWA features that support an enhanced and adaptable UX are as follows:

- **MediaSession API**: this feature enables content that plays out of the context of the Web application, such as a video or audio stream controlled from the locked screen overlay or notifications area;
- **Offline/cached content**: service workers act as a proxy between the Web page and the site, allowing caching of images, code, and media that can be used while offline;
- **Background Fetch**: still in development, this API will allow media files to download in the background while users navigate to other areas of the application;
- **Full-screen mode**: HTML5's full-screen capabilities can be leveraged to allow easier takeovers of the entire screen of the device, giving the PWA the look and feel of a Native application;
- **Media Capabilities API**: while not yet available in current browser stacks, development is underway to enable configurations that determine the best media source to play for a given device's processing power, bandwidth, and power consumption [Rei17].

There are several frameworks largely used for Web development and PWA, such as React [Rea], Angular [Ang], Vuejs [Vue] and Ionic [Ion].
Briefly, the main characteristics of PWA are that they are platform-independent, app-like, installable, and linkable.
Based on this short analysis, we can conclude that the main advantages in the development of a PWA concerning the Native applications are as follows:

- Cheaper and faster development;
- Management of development teams easier;
- Centralized implementation always updated;
- Technology with great evolution and easier debugging;
- It can be run on any device;
- It doesn't need the approval of the respective Marketplace [Rei17].

There are also a few challenges that PWA face, such as:

- Limited functionalities in terms of hardware;
- Less UX than in Native Apps;

- They can't be promoted in Marketplace.

In table 2.1, we can see the comparison between Native Apps, PWA and standard Web Apps on several parameters.

	Native App	**PWA**	**Standard Web App**
Installation	Need to go to the Marketplace and click download.	Just click a button to add them to the phone home screen.	Installation not required.
Updates	Need to be submitted to the Marketplace and then downloaded by the user.	Updates are instant.	Updates are instant.
Size	Mostly heavy in size. They can take time for downloading on a users' device.	Small and fast.	Small and fast.
Offline access	Available.	Need to use the app once online and then should be able to access the cached content offline.	Not required.
UX	Excellent when the application is well designed.	It can become a little confusing because of the double menus (app menu and browser menu).	Same as PWA.
Push notification	Yes.	Yes.	Yes (Only possible with third party services).
Discoverability	Not good – need to work hard on app store optimization.	Good – to make appear in search results need to be optimized for Search Engine Optimization (SEO).	Not required.

Table 2.1: Comparison between Native App, PWA and Standard Web App [TJ18].

Conclusively, in order to make the Web application responsive to any device, we can use backend frameworks like Django combined with the Model, View, Template (MTV) model. Chrome browser also has Dev Tools to help to adjust this need of **responsibility**.

Then, it is necessary to make the Web App **work offline**. For this, we have the **Service Worker**, which is a JavaScript file that is executed on a different thread than the browser with the ability to intercept network requests, by listening to events such as the fetch event (which is triggered when resources are being consumed through the Hypertext Transfer Protocol (HTTP) protocol) and handling them appropriately [KABAK] [RPS19]. It uses a cache API that enables the storage and retrieval of content. It functions in conjunction with the Service Worker to enable it to cache network requests so that they can provide appropriate responses while offline. Finally, it also enables the application to synchronize with the server when the network conditions allow it [RPS19].

2.2.2 Client-side Technologies

The main client-side technologies, at the base of frontend development, are divided into: Markup (HTML), Style (Cascading Style Sheets (CSS)) and Behaving (JavaScript) [RPS19].

2.2.2.1 HTML

The first Web sites that appeared on the Internet, contained only static content, that is, they consisted of HTML pages exposing only texts, images, and links, without much interaction with the user. This markup language is therefore responsible for defining what each content represents on a page and creating links to other content [Devc].
Currently, HTML is in version 5 and is standardized by the W3C (World Wide Web Consortium), an international organization responsible for setting standards for the Internet [Deva].

2.2.2.2 CSS

The CSS is a style sheet used to define the presentation of HTML documents. Its main benefit is to separate the appearance of the content of a document. Thus, instead of the formatting being inside the HTML document, there will only be a link to the CSS file, so that several pages can use the same formatting style [Devc].
CSS can specify a document's style in terms of colors, fonts, page layout, the spacing between paragraphs, background images, variations in display for different devices and screen sizes, as well as a variety of other effects [Tutc].
The main advantages of CSS are that it saves time (we can write the CSS once and then reuse the same sheet in multiple HTML pages), has easy maintenance, has a much wider array of attributes than HTML, has multiple device compatibility (allows content to be optimized for more than one type of device), presents global Web standards and also the pages load faster (if we are using CSS, we don't need to write HTML tag attributes in every occurrence) [Tutc].

2.2.2.3 JavaScript

The need for more dynamism on the Web pages culminated in the creation of JavaScript. With this technology, more interaction on the pages was possible, since it has features

such as handling content, elements, and HTML events [Devc].
JavaScript is an interpreted programming language. This language was originally implemented as a part of Web browsers so that scripts could be executed on the client-side and interact with the user, without the need for this script to pass through the server, controlling the browser, performing asynchronous communication, and changing the content of the displayed document. Thus, it is currently the main language for client-side programming in Web browsers. It is also starting to be widely used on the server-side through environments such as node.js [Devb].
JavaScript codes can be inserted into the HTML pages in two ways: directly inside the HTML file structure with the `<script> </script>` tags or in a separated file with ".js" extension, referencing it on the HTML document [Devb].
The core of JavaScript language consists of several programming common benefits such as store useful content in variables, run the code in response to certain events that occur on a Web page, String operations, loading new content or data onto the page without reloading the page, among others [Moz].

2.2.3 Chrome Extensions

A common characteristic of modern Web browsers is that their functionality can be extended via third-party add-ons. In this section, we focus on Chrome extensions, to which the Chrome browser exports an API: extensions can make network requests, access the local file system, get low-level information about running processes, etc [BCJ+14].
Google Chrome Developers define these extensions as small software programs that customize the browsing experience, enabling users to tailor Chrome functionality and behavior to individual needs or preferences. They are built on Web technologies such as HTML, JavaScript, and CSS [Chr13].
The User Interface (UI) can range from a simple icon to overriding an entire page. They are zipped into a single *.crx* package, that the user downloads and installs. Thus, they don't depend on the content from the Web, unlike common Web Apps [Chr13].
Extensions are distributed to the public through the Chrome Developer Dashboard and published to the Chrome Web Store [Chr13].
Once installed, Chrome extensions make themselves visible by placing a logo icon on your navigation bar. These shortcut icons can be managed and organized according to your specifications [Mic17].
In Chrome's Web Store, you can find a broad range of extension categories, such as productivity, organization, security, communications, and more. Most of them are centered on saving you time, keeping you organized, increasing efficiency, and making your collaborations more engaging [Mic17].

2.2.3.1 Manifest

The Manifest.json is the most important required file in a Chrome extension. Here is stored the main information about the extension that the Web browser will firstly read.

This file will also define all the proprieties, rules, and permissions about the *software* [Wil19].
There are required, recommended and optional properties (or attributes). The "***background***" property must contain the background script that will be used in the extension. This script is used based on the Chrome App Lifecycle, which is Chrome's life cycle on extensions [Wil19].

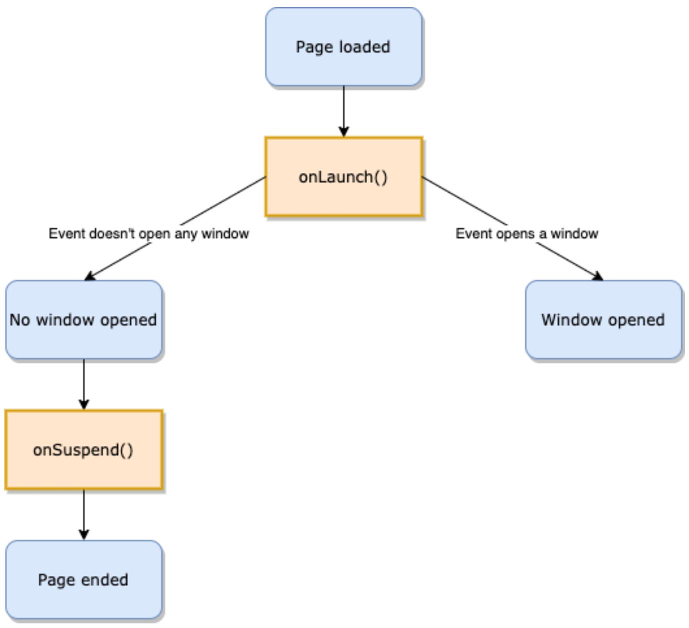


Figure 2.2: Chrome's app life cycle.

As we can see in figure 2.2, there are two events that can be triggered: `onLaunch()` and `onSuspend()`. Inside the first one, there are two possibilities: a window is open or no window is open. The second event runs just before the page is ended, due to the fact that no windows were previously opened [Wil19].
We can also find the property "***manifest_version***" that specifies the version of the manifest file that we are using in the extension. There are two possibilities: versions 1 or 2. The first one works with Chrome versions above and below 17, however, we can only use the functionalities below version 17. On the other hand, on version 2, we have access to all new and old functionalities, despite that Chrome browsers with versions 17 and below won't be able to read the new ones [Wil19].
Another property that may be in the manifest.json file is the "***default_locale***". As we can see in figure 2.3, this is an attribute that specifies the sub-directory of the languages folder, if exists. This field is mandatory in extensions that have the "_locales" folder, otherwise, it should be absent [Wil19].

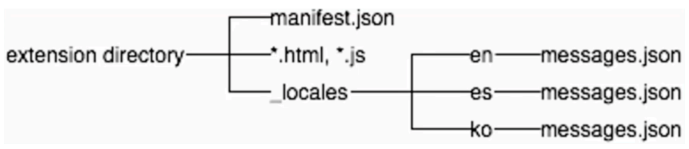
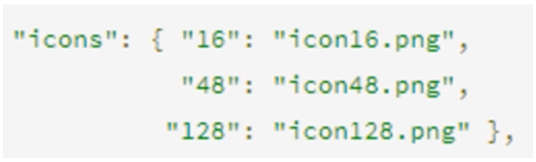


Figure 2.3: "_locales" folder [Wil19].

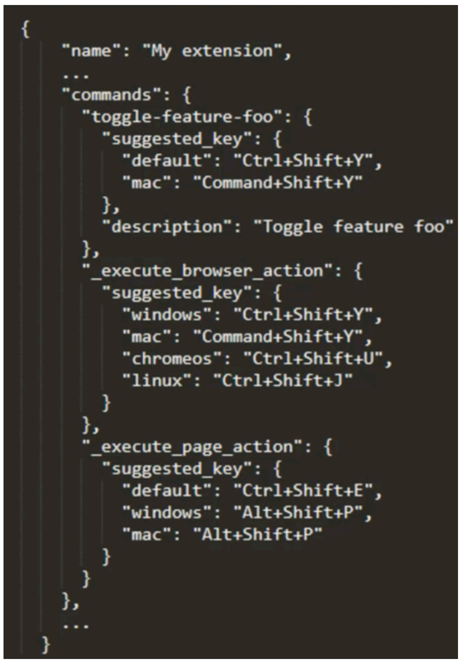
We can also have of other properties such as ”***description***” and ”***icons***”. The ”*description*” attribute is where the extension description is written, and it appears directly on the extension’s developer dashboard. The ”*icons*” attribute allows you to define several icon sizes (these sizes must be a perfect square such as 16x16 or 48x48) that will appear in Chrome’s navigation bar, as we can see in figure 2.4 [Wil19].



```
"icons": { "16": "icon16.png",
           "48": "icon48.png",
          "128": "icon128.png" },
```

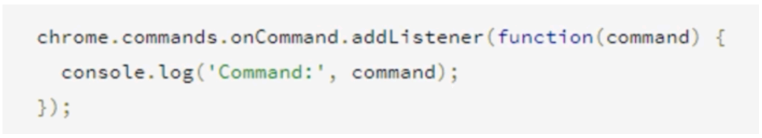
Figure 2.4: Example of icon sizes [Wil19].

The property ”***action_handlers***” declares which user actions or intentions the extension supports. These can also serve as alternative initialization points for the app [Wil19]. As the name suggest, the ”***author***” attribute defines the author’s name and the ”***bluetooth***” property declares which permissions will be available to the Chrome’s Bluetooth API usage [Wil19].
There is also the property ”***commands***”, where you can define specific keyboard shortcuts for the interaction between the user and the extension, as we can see in figures 2.5 and 2.6 [Wil19].



```
{
    "name": "My extension",
    ...
    "commands": {
      "toggle-feature-foo": {
        "suggested_key": {
          "default": "Ctrl+Shift+Y",
          "mac": "Command+Shift+Y"
        },
        "description": "Toggle feature foo"
      },
      "_execute_browser_action": {
        "suggested_key": {
          "windows": "Ctrl+Shift+Y",
          "mac": "Command+Shift+Y",
          "chromeos": "Ctrl+Shift+U",
          "linux": "Ctrl+Shift+J"
        }
      },
      "_execute_page_action": {
        "suggested_key": {
          "default": "Ctrl+Shift+E",
          "windows": "Alt+Shift+P",
          "mac": "Alt+Shift+P"
        }
      }
    },
    ...
  }
```

Figure 2.5: Defining commands [Wil19].



```
chrome.commands.onCommand.addListener(function(command) {
  console.log('Command:', command);
});
```

Figure 2.6: Triggering commands actions [Wil19].

The ”***permissions***” property is one of the most important ones in the manifest file. Here is where is declared the required permissions array for running the extension [Wil19].

There are plenty other properties that can be present in the manifest.json file such as: ”***sockets***, ”***storage***”, ”***version_name***”, ”***update_url***”, ”***offline_enabled***”, ”***requirements***”, ”***event_rules***”, among others [Wil19].

2.2.3.2 Background Script

This script will manipulate all events triggered by Chrome. It contains listeners to the browser's events that are important to the extension. It will remain inactive until an event is triggered and executes the instructed logic. Thus, an effective background script is loaded only when needed and unloaded when it goes idle. Some examples of background script events include:

- The extension is first installed or updated to a new version;
- The background page was listening for an event, and the event is dispatched;
- A content script or other extension sends a message;
- Another view in the extension, such as a popup, calls `runtime.getBackgroundPage` [Chr12b].

Once it has been loaded, a background page will stay running as long as it is performing an action, such as calling a Chrome API or issuing a network request. Additionally, the background page will not unload until all visible views and all message ports are closed. It is important to notice that opening a view doesn't cause the event page to load, but only prevents it from closing once loaded [Chr12b].

2.2.3.3 UI

An extension's UI should be clean, purposeful, and minimalist. It should customize or enhance the browsing experience without distracting from it [Chr12a].

Most extensions have a browser action or page action, but they can contain other forms of UI, such as context menus, use of the omnibox, or creation of a keyboard shortcut. Extension UI pages, such as a popup, can contain ordinary HTML pages with JavaScript logic [Chr12a].

An extension using a page action and a popup can use the declarative content API to set rules in the background script for when the popup is available to users. When the conditions are met, the background script communicates with the popup to make its icon clickable to users, as we can observe in figure 2.7 [Chr12a].

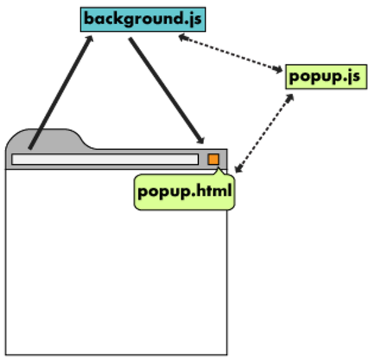


Figure 2.7: Interaction between background script and popup page [Chr12a].

2.2.3.4 Content Script

Extensions that read or write to Web pages use a content script. As we can see in figure 2.8, it contains JavaScript that executes in the contexts of a page that has been loaded into the browser. Content scripts read and modify the DOM of Web pages the browser visits [Chr12a].

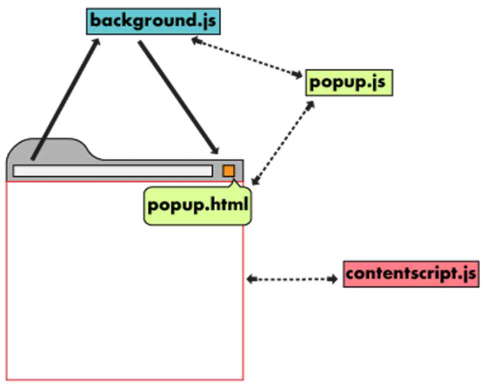


Figure 2.8: Content script working [Chr12a].

They don't have direct communication with the popup or the background, however, they can communicate with their parent extension by exchanging asynchronous messages and storing values using the storage API, as we can see in figure 2.9 [Chr12a].

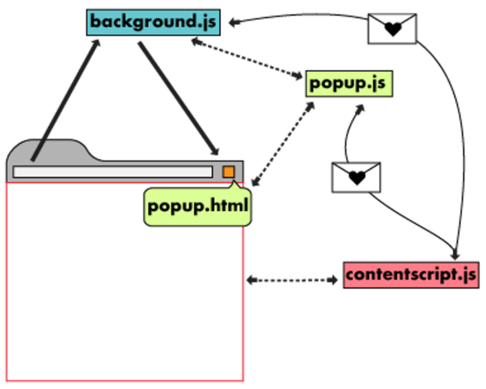
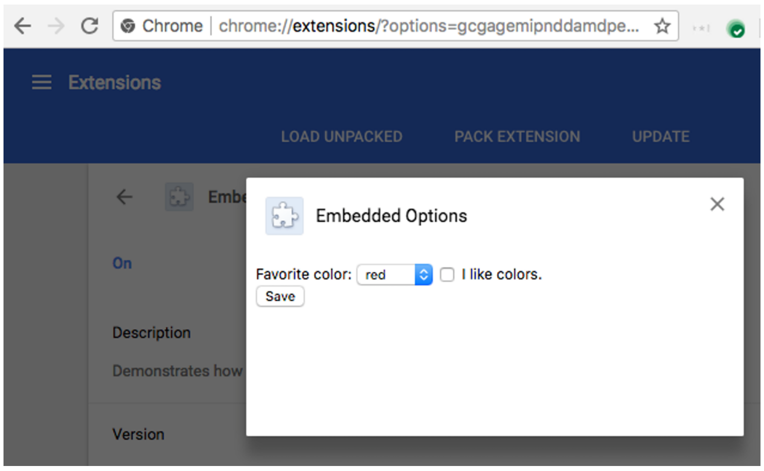


Figure 2.9: Content script asynchronous messages [Chr12a].

2.2.3.5 Options Page

Just as extensions allow users to customize the Chrome browser, the options page enables customization of the extension. Options can be used to enable features and allow users to choose what functionality is relevant to their needs as we can see in figure 2.10 [Chr12a].
A user can view an extension's options by right-clicking the extension icon in the toolbar and then selecting options or by navigating to the extension management page at `chrome://extensions`, locating the desired extension, clicking Details, and then selecting the options link [Chr12a].



Favorite color: red ☐ I like colors.

Figure 2.10: Options page [Chr12a].

2.2.3.6 Chrome API's

The Chrome API's are a fundamental part of an extension architecture as they offer access to a huge library of interaction possibilities between the user and the browser [Wil19].
Those API's have asynchronous and synchronous methods. Most of them are asynchronous: they return immediately without waiting for the operation to finish. If an extension needs to know the outcome of an asynchronous operation it can pass a callback function into the method. The callback is executed later, potentially much later, after the method returns. On the other hand, synchronous methods have no callback option and return a type of String. For example, `string chrome.runtime.getURL()` method synchronously returns the URL as a string and performs no other asynchronous work [Chr12a].

2.2.3.7 Advantages vs Disadvantages

As with all software, Chrome extensions present clear advantages for the users but can also have some serious disadvantages and vulnerabilities.
Some of the **advantages** of Chrome extensions are that they include the ability to choose programs to make you more productive and stay organized [Mic17]. It improves business productivity by saving time, keeping a strong check, and control over the business operations and resources [Cod18].

There are also many Chrome extensions that can be used on multiple devices, improving employees' performance while making them accessible anywhere and at any time [Cod18]. Chrome extensions can additionally help you to manage social media tasks. You can use these extensions to improve the social media marketing workflows [Cod18].
Shortly, the main advantages of Chrome extensions are as follows:

- Support offline functionality and features that are limited while using HTTP protocol;
- A few extensions pave a way for shortcuts that are extremely useful and truly uncomplicated;
- The development of Chrome extensions would envelop a huge spectrum of individuals hence, the marketing of products and services will subsequently amplify;
- Chrome extensions add great functionality in terms of privacy, security, etc. These functionalities are not accessible otherwise and unless having a website;
- Assurance of work in an organized environment;
- Reaching targets within seconds and in the most effective manner;
- Obtaining unique business exposure [Cod18].

On the other hand, extensions can also have **disadvantages**, as they can create an entry point for malware or adware.
Since an extension is browser-based and operated, any security threats would include the extensions having access to your online behavior [Mic17].
Through Web browsers, users can conveniently access a wide range of services such as email, cloud-based file sharing, banking, health care, shopping, etc. While a user is interacting with such services, his sensitive, personal information, such as bank account numbers and passwords, are exposed to the browser and scripts running on Web pages. These powerful add-ons to the browser, combined with already fragile browser security, further expand browsers' attack surfaces. As a result, malicious add-ons can enable attackers to gain access to a wide range of private, sensitive data, and computer resources [BCJ$^+$14].
Google usually does a good job of controlling permissions and making you aware of what resources the extension will require, however, you can minimize the risks by choosing only extensions that come from trustworthy sources and are well-reviewed [Mic17].
In 2009, Google Chrome introduced a new extension platform with several features intended to prevent and mitigate extension vulnerabilities: strong isolation between websites and extensions, privilege separation within an extension, and an extension permission system [CFW12].
It can also be good to limit the number of extensions you install, so they won't affect the performance of your system. It's important to keep in mind that every extension comes with its own code and each one is pulling on system resources [Mic17].

2.2.4 Fact-checking Going Mobile

Due to the increasing popularity of online social media, the Internet has become a fertile breeding ground for spreading fake news, such as misleading information, fake reviews, fake advertisements, rumors, fake political statements, and so on. In addition, a large amount of real-time information is created, commented on, and shared via online social media every day, which makes online real-time fake news detection even more difficult [ZG20a].
As the technology is growing, the use of smartphones for live reporting and journalism has questioned the authenticity of the content being published on the Internet. The use of photo and video sharing sites and social platforms has maximized the unauthenticity of content. The immediate transmission has increased the competition between professional reporters and amateur eyewitnesses capturing events using their personal devices. The videos and photos may be tempered or manipulated, or the content may be provided in such a way that it conveys half or wrong information [Uma16].
Some of the methods that can help to authenticate information in mobile journalism can be:

- The existence of a large number of people reporting about the same information, in order to be more reliable;
- The inclusion of GPS coordinates to determine the location and then the authenticity of the content.

The growing use of mobile devices has dramatically changed how information is spread. This reality may come from the fact that fewer people read newspapers, listen to the news on the radio or watch it on the television, means that served as an "anchor" or starting point for the propagation of the most reliable and credible news. Nowadays, with the majority use of mobile devices, people tend to consume without context, in most cases not directly entering pages of reference media, but on social networks, absorbing everything that comes from them. Therefore, it can be said that there is a tremendous proliferation of pseudo-media that assumes the appearance of reliable media to disseminate disinformation, which often makes the work of detecting those fake news quite difficult [BTCG19].
It is easier to distribute fake news via social media than through traditional mainstream media that tend to check and validate news before distribution. So, fake news production from religious, commercial, and political propaganda units has become active, to influence, persuade, alienate and brainwash the public through social media [VVY19].
Even though the definition of fake news has changed with time, they still tend to be intrusive and diverse in terms of topics, style, and platforms, and it's not easy to construct a generally accepted definition for it.
As we can see in figure 2.11, **fake news have 3 basic characteristics**. The **volume** of the fake news refers to the massive amount of fake content that is distributed through the Internet, and the ease of it. The **veracity** of the fake news means that with the increasing popularity of social media, fake news can dominate the public's opinions, interests, and decisions, changing the way people interact with the real news. The **velocity** of the fake

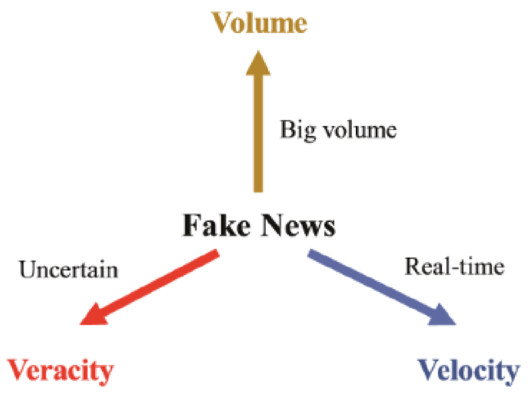


Figure 2.11: The volume, velocity, and veracity of fake news.

new applies to the fact that it is usually short-lived, and only exists in a certain moment of a current event, in order to avoid detection by the detection systems [ZG20b].
To clearly understand the scope of online fake information, it will be analyzed some important aspects for defining fake news, such as its four major components:

- **Creator/Spreader**: can be either human (essential sources for fake news diffusion) or not human. As non-human creators, we can have, for example, Social Bots and Cyborgs. Social Bots are computer algorithms that are designed to exhibit human-like behaviors, automatically produce content, and interact with humans on social media. Cyborgs refer to either bot-assisted humans or human-assisted bots. After being registered by a human, the Cyborg account can post tweets and participate with the social community. Malicious Cyborg accounts mislead and exploit online social users by propagating fake information and messages;

- **Target Victims**: the main targets of online fake news. They can be users from online social media or other online news platforms. Based on the purposes of the dishonest information, the targets usually are vulnerable or easily influenced people (e.g. students, voters, parents, etc);

- **News Content**: refers to the body of the news. It contains both physical content (e.g. title, body text, videos) and non-physical content (e.g. purpose, topics, opinions). Non-physical content is the main core of fake news since it contains all the important ideas, feelings, and views that the authors want to pass to the readers. To make their news persuasive, authors often express strong positive or negative feelings in the text body;

- **Social Context**: indicates how the news is distributed through the Internet. Includes user network analysis (how online users are involved in the news) and broadcast pattern of the dissemination. Online users can not only learn about trending events, but they can also share their stories and promote problems and issues as well. From friends to followers, online social users share experiences and interactions within certain social groups. If a group of like-minded people post, share and forward certain information, the influence of that message may be amplified. This effect (echo chamber) can facilitate the spread of fake news since online users may be exposed to the social community in an exaggeratedly distorted way [ZG20b].

We can also analyze the impact of fake news on Ad Tech Firms. Major of these companies usually tolerate fake news, because the advertising ecosystem is built around the idea of targeting individual users based on diverse forms of behavioral profiling (user-based approach). One of these advertising strategies is "retargeting", which is the practice of displaying ads to users for products they have previously viewed. This focus on the users incentivizes advertisers to accept the placement of their ads on less credible and dishonest sites, particularly when users visiting these sites probably have no problem with them and it means getting the ad space more cheaply than on the site of a reliable publisher like a mainstream news outlet [BE19].
The modern Web has become so synonymous with mobile access that websites are now increasingly designed mobile-first. Mobile devices are usually never far from our sides, providing us instant access to the Web at large. Yet, their small screens, constrained interfaces where we are unlikely to open large numbers of tabs and on-the-go usage, promote the very behaviors that help fake news, disinformation, and foreign influence spread, by discouraging us from researching the things we read [Kal19].
Fake news diffusion patterns on social media have often been studied to identify the characteristics of fake news that can help differentiate it from true news. Fake news detection essentially relies on exploiting these differences to classify information based on its veracity [SQJ+19].
Simply speaking, we can define fake news detection as the task of assessing the truthfulness of a certain piece of news [VR14]. Nowadays, false information is spreading over social media at an increasing speed, and the detection of it is becoming even more challenging. However, fake news detection is a non-trivial task, which requires multi-source information such as news content, social context, and dynamic information. One of the challenges for fake news detection is the lack of a labeled benchmark dataset with reliable ground truth labels and comprehensive information space, based on which we can capture effective features and build models [SMW+18].
According to Karishma et al. [SQJ+19], some of the basic relevant characteristics for fake news detection are:

- **The verification of the source/promoters of the information**: the use of misleading domain names is a particular characteristic of fake news;
- **The information content**: the content of the information being spread is essentially what needs to be classified as true or fake. Horne & Adali [HA17] identified some characteristics that can differentiate between real and fake news by studying textual content both in real and fake news websites. Those researches suggest that titles of fake news are usually longer, have more capitalized words, have a lot of negative statements and the body content is shorter and repetitive, having fewer technical vocabulary. Also, the text tends to be focused on the present and future, instead of being more objective and factual;

- **User responses on social media**: user responses on social media provide helpful information that is significantly beneficial for detecting fake news. They usually provide more effective signals for fake news detection than the information content itself because user responses and propagation patterns are harder to manipulate than the information content [SMW$^+$18].

It is easier to spread fake news via social media than through traditional mainstream media that tend to check and validate news before distribution. So, fake news production from religious, commercial, or political propaganda units has become active, in order to influence and brainwash the public through social media. Hence, automatic fake news detection is one of the emerging research fields, which can rely on assessing the credibility of text news. However, has been also researched the detection of distorted images in fake news [VVY19].

2.3 Fake News and Fact Checking on the Web: The Road So Far

The credibility of fake news is usually getting boosted by several factors, such as the imitation of well-known authors writing style (style-based approach) or the expression of opinions with a vocabulary commonly used in real news. With the exponent increasing of fake news, several detection methods have been studied. These methods may be grouped into two different classes: linguistic-based and network-based methods [ZGK$^+$19]. Network-based methods apply network properties as a supporting component for diverse linguistic-based approaches. These properties can be websites information, authors information, timestamps, etc.
A study from Venkatesan et al. [VJ13] proposes a model that focuses on investigating the accuracy of the posts (binary variable) in online crowd-sourced health, based on the clarity of the thread question, the information richness (amount of detail contained in a thread) and the users' potential for making useful contributions (user rating).
Zhang et al. [ZGK$^+$19] approach aim to clutch and understanding news through complete comparisons of news content. This approach uses topic and event-level analysis to understand patterns that are deeply embedded within the news for improved detection accuracy. To achieve this detection system, they used in-depth semantic analysis of the sentences by incorporating open information extraction coupled with other techniques to extract knowledge from news articles.
To help distinguish between real and fake news, Vishwakarma et al. [VVY19] also implemented a new way to automate fake news detection, that consists of a four-step approach:

1. **Text extraction**: a maximally stable external region in the image is identified using an algorithm to extract the text of different sizes and fonts;
2. **Refinement**: the extracted text is refined, removing multiple occurrences of the same word, spelling errors, and extracting unique entities;

3. **Searching**: the refined text is used for search through Google and the related Web links are ordered and classified into reliable or unreliable sources;

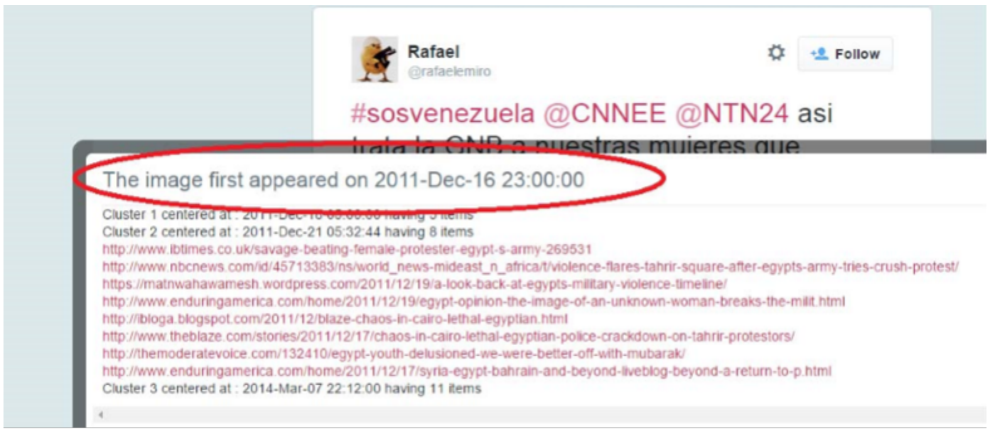
4. **Calculation of the reality parameter**: this parameter has to be more than 40 for news to be considered as real.

A researcher team from *Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência* in Oporto is developing a new system for analyzing and detecting automatically fake news and misinformation in social networks. The system also intends to help users filtering the most relevant content in social networks. This solution uses more than 100 indicators, such as psycholinguistic as statistical ones, and it is based on a ML model, where it classifies within a certain probability if a news publication is real or fake. According to Álvaro Figueira, one of the researchers, the main challenge that the system faces, is the domain change and the temporal context in which fake news can arise [Lus20].
Ultimately, the Covid-19 pandemic has been increasing the amount of fake news and disinformation sharing through social networks and mobile applications, such as WhatsApp. This fake news gained a lot of strength and visibility, especially in mid-March 2020, when the disease began to reach worrying numbers, in Portugal. A study made by researchers from *Instituto Universitário de Lisboa* refers that the sharing of false information and manipulated content were mostly made through audio clips (approximately 70%) in WhatsApp, being those messages forwarded without any concern or content verification, most of the time [CG20]. According to Cardoso et al., the number of Facebook groups about Covid-19 exponentially increased, and the number of members is also growing, and with them the spread of misinformation. The analysis of posts and messages shared about the "Coronavirus" in social networks shows that in the beginning, when the search and information sharing about that topic was most intense, there was a greater tendency for misleading messages to appear on the Internet [CG20].

2.3.1 Fake News Detection on Images

Few studies have been done in fake news detection on images however, for understanding the truthfulness of social news events, it is important to exploit visual content on social media. Images often reveal vivid descriptions of the news events and are largely used in social networks, mainly on Twitter or Instagram, because it is easier and more efficient to tell stories or to spread some (fake) news through them, due to their eye-catching nature.
Images also have different distribution patterns for real and fake news. Usually, people tend to report the news with images taken by themselves at the event scene. If the event is real, then a lot of images are often posted by different witnesses on social media or networks. On the other hand, if the event is fake, there are very few different images posted [JJZ+17]. For news verification, Jin et al. [JJZ+17] focus on images features extracted from the visual characteristics (clarity, diversity, and coherence in a news event) and the overall statistics of images in news event (image number, multi-image number, image-to-tweet ratio, resolution of the image, popularity, etc).

Elkasrawi et al. [SDAB16] approach is based on analyzing the images attached to online news reports or social media stories and trying to find occurrences of those images online. To do so, their algorithm uses Google Image Search, which returns a list of similar images based on the given image, and applies K-means clustering based on the publishing day, to get the general overview of how images were used over time, as we can see in figure 2.12.



```
The image first appeared on 2011-Dec-16 23:00:00
Cluster 1 centered at : 2011-Dec-16 [illegible] having [illegible] items
Cluster 2 centered at : 2011-Dec-21 05:32:44 having 8 items
http://www.ibtimes.co.uk/savage-beating-female-protester-egypt-s-army-269531
http://www.nbcnews.com/id/45713383/ns/world_news-mideast_n_africa/t/violence-flares-tahrir-square-after-egypts-army-tries-crush-protest/
https://matnwahawamesh.wordpress.com/2011/12/19/a-look-back-at-egypts-military-violence-timeline/
http://www.enduringamerica.com/home/2011/12/19/egypt-opinion-the-image-of-an-unknown-woman-breaks-the-milit.html
http://ibloga.blogspot.com/2011/12/blaze-chaos-in-cairo-lethal-egyptian.html
http://www.theblaze.com/stories/2011/12/17/chaos-in-cairo-lethal-egyptian-police-crackdown-on-tahrir-protestors/
http://themoderatevoice.com/132410/egypt-youth-delusioned-we-were-better-off-with-mubarak/
http://www.enduringamerica.com/home/2011/12/17/syria-egypt-bahrain-and-beyond-liveblog-beyond-a-return-to-p.html
Cluster 3 centered at : 2014-Mar-07 22:12:00 having 11 items
```

Figure 2.12: Example output for image authenticity check [SDAB16].

To detect the alteration in the image (and believing that the real one is on the Internet), it also uses techniques like edge detection, scaling, and color conversion. Based on the alteration in the image, the difference in publishing date, and the first occurrence of the image on the Web, it labels the news events as fake or real [SDAB16] [Kom17].

2.3.2 Fake News Detection on Text

Many authors and researchers have worked on the veracity classification problem on fake news body text.
The system proposed by Alrubaian et al. [AAQHA18] represents an iterative process that combines an automated-based methodology for achieving better credibility or trustworthiness results with sophisticated accuracy. As so, it consists of five general procedures:

1. **Tweet collecting and repository** (a network-based approach): using a streaming API (to collect datasets on given events) and an API for searching tweets (to collect users' tweets histories simultaneously);

2. **Credibility scoring technique**: uses a credibility classifier engine, and relies on ML methods that are based on training with an established ground truth;

3. **Reputation scoring technique**: a reputation-based technique that measures the level of dependability of the user who posted the content. It does not consider aspects such as message content features but does consider factors such as the structure of the network in its model;

4. **UX measuring technique**: a user-based approach that requires user expertise method applying both previous techniques in order to establish the reliability of users;

5. **Trustworthiness value**: is where all of the scores obtained using the three techniques are combined, to obtain the trustworthiness value of a given tweet.

To increase the instantaneous detection performance of unconfirmed information, Qin et al. [QWLT16] introduced novelty-based features for rumors detection and dubbed Pseudo-Feedback (PF) features to boost its performance. The PF feature describes the maximum similarity between a news document and those previously considered as rumors. They also applied features based on the presence or numbers of URLs, hashtags and usernames, punctuation characters as well as eight different categories of sentiment and emotions.
Sarcasm is also one of the main problems with social media. Bouazizi & Ohtsuki [BO16] treated this sarcasm problem on Twitter using a pattern-based approach and introduced four sets of features that cover the types of sarcasm (sarcasm as a wit, as a whimper, and as evasion) and classified tweets as sarcastic or non-sarcastic, using ML algorithms to the classification.

2.3.3 Online Fact-checking Resources

Fact-checking resources are a practical-based detection approach, which works from the perspective of Internet users, commonly made by mainstream media organizations. Real-time news is always the mixture of information and due to this, sometimes a binary classification result cannot fully describe the overall problem. Many evaluation criteria are used to define the truthful level of the news in current fact-checking resources, by telling people what is true or fake [ZG20a].
However, there is no homogeneous definition of these organizations or some common essential characteristics. Some of them differ according to subjects such as political claims, rumors, online lies, or controversies.
All of these fact-checking platforms played a significant role mainly in the context of the Covid-19 crisis. Due to global reach, the pandemic has spread to these specialized platforms and has become a topic that monopolizes a large number of checks on verification platforms [SBLP+20].
Some of the popular fact-checking online resources are:

- ***Factmata.com*** [Fac]: used for statistical fact-checking and claim detection. The claim checking is depending entirely on AI and ML algorithms. The main goal of this platform is to detect and verify misinformation providing a better-informed opinion on the world;

- ***Hoaxy.iuni.iu.edu*** [Hoa]: a framework for collection, detection and analysis of online misinformation and its related fact-checking efforts;

- ***Snopes.com*** [Sno]: a fact-checking website for validating and debunking stories. Includes a large range of topics such as business, computers, crime, fraud, scams, and so on, assigning a truth rating to that printed resources;
- ***TruthOrFiction.com***[Tru]: an online website where people can quickly get information about e-rumors, warnings, hoaxes, viruses, and stories. This website mainly focuses on misleading information that is popular via emails;
- ***Newtral.es*** [New]: an audiovisual content startup that has three business areas: production of television programs and new narratives on social networks, innovation in journalism through fact verification (fact-checking), and also launched a line of investigation based on AI protocols;
- ***Maldita.es*** [Mal]: an independent journalistic platform focused on the control of disinformation and public discourse through fact-checking and data journalism techniques.

2.3.4 Machine Learning Contribution in Fake News Detection

Initially, and before the discussion about machine-based approaches, we can identify the human detection approach as being the humans deciding, without the help of any tool, if a news article is real or fake. This is usually a fast method of classification but have also a lot of limitations, such as:

- **Time to compute information**: humans cannot fact-check the huge information volume that is always being generated;
- **Lack of relevant information available**: the processing of persuasive information (e.g. fake news) involves assimilating, interpreting, and evaluating a lot of content;
- **Comprehension**: humans often lack the required literacy skills to discern correctly if a news article is real or fake;
- **Virality**: most of the time, the more visibility a news article has, the more is believing (illusory-truth) [OAI+19].

There are two methods in which humans can process persuasive information: systematic processing, which involves careful consideration and analysis of the information content and the idea behind it, and heuristic processing which relies on the use of mental shortcuts and common sense to discern the validity of the information [OAA+18].
On the other hand, and to help solve the drawbacks of the human-based approach, ML techniques are quite used. These are simply patterns detectors, where you can present a large set of labelled input and output data and train the algorithms to perceive the underlying patterns. This approach broadly uses linguistic methods and the network analysis approach, as presented in previous sections.

However, both of these methods depend on a pre-existing knowledge base and have their limitations. The ability to transfer our intuitive sense of context to the algorithms (satire, humor, exaggeration, etc) is one of the limitations of the linguistic-based approach. On the other hand, the network-based approach has some limitations too such as the non-existence of standardized fake news DBs and large-scale benchmark datasets. Also, there is no definitive list of fake news websites for URL-based solutions [Kom17] [OAA$^+$18] [Kni17].

Hence, when we try to develop an overarching fact-checker, we soon realize that our current ML tools are wonderful pattern-matches and pattern-detectors, but not full-fledged reasoning machines [Kom17].

The fake news detection fight is not a problem to be solved by technology alone, and with the limitations of the existing approaches, it emerged the need for a hybrid one, that pulls together the strengths of the human and machine methods for fake news detection [Kom17] [OAA$^+$18].

For example, on Facebook, users now can flag content as fake news, and based on a threshold level of flagging, such articles are sent for review by professional fact-checkers [Kom17].

During her keynote in EmTech MIT 2017, Daniela Rus talked about a study by researchers from Harvard where they compared the ability of doctors and AI/ML software to diagnose cancer. The results were that the doctors perform is much better than the software one but, however, doctors together with software were even better [Kni17].

As we can see, the human-plus-machine approach is being leveraged, rather than the machine's algorithms themselves. Thus, in the fact-checking domain, humans should lead, and machine learning algorithms should play the supporting role [Kom17].

2.3.5 Fact-checker API

This tool [Api21] is a powerful high-level API Design Stack, that allows access to the DB of fact-checked statements. This API was created by a company called Apiary [Ora21] and was later acquired by Oracle. Some Fact-checker API endpoints don't require authentication while others do: the query facts endpoint should not require authentication, since it will be used by the browser extension and the website widget; all other endpoints will require authentication since they will be used to build tools to enable easier association of facts to Uniform Resource Identifiers (URIs) [Api21].

The priority, for now, is the unauthenticated endpoints. All the others are just ideas at the moment to allow the development of tools for easier updating and association of URIs with already existing facts [Api21].

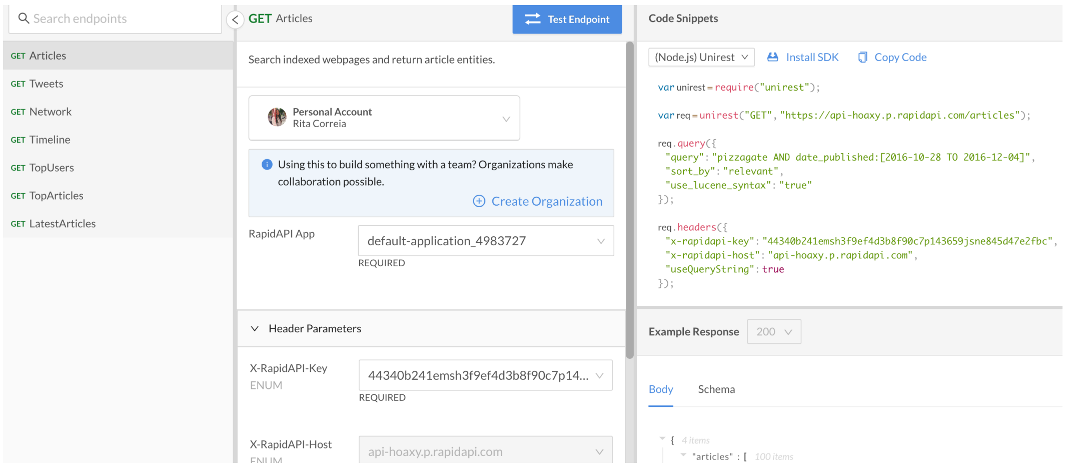
A statement is a claim made by politician (or organization) that has been fact-checked. There are several attributes that can be associated with statements, such as: `id`, `text`, `person_name`, `rating`, `explanation`, `factchecker_uri` and `sources` [Api21].

This endpoint will return all the statements that are associated with the specified URIs. If no URI parameter is set, all the statements in the DB will be returned [Api21].

The sources of the statements that were fact-checked can include news articles or governmental websites, but also Youtube videos and other online sources [Api21].

2.3.6 Hoaxy API (RapidAPI)

As it was mentioned in sub-section 2.3.3, Hoaxy [Hoa] visualizes the spread of claims and related fact checking online. A claim may be a fake news article, hoax, rumor, conspiracy theory, satire, or even an accurate report. Anyone can use Hoaxy to explore how claims spread across social media, and it is possible to select any matching fact-checking articles to observe how those spreads as well, as we can see in figure 2.13 [tru21].



Search indexed webpages and return article entities.

Personal Account
Rita Correia

Using this to build something with a team? Organizations make collaboration possible.

Create Organization

RapidAPI App: default-application_4983727 (REQUIRED)

X-RapidAPI-Key (ENUM): 44340b241emsh3f9ef4d3b8f90c7p14... (REQUIRED)

X-RapidAPI-Host (ENUM): api-hoaxy.p.rapidapi.com

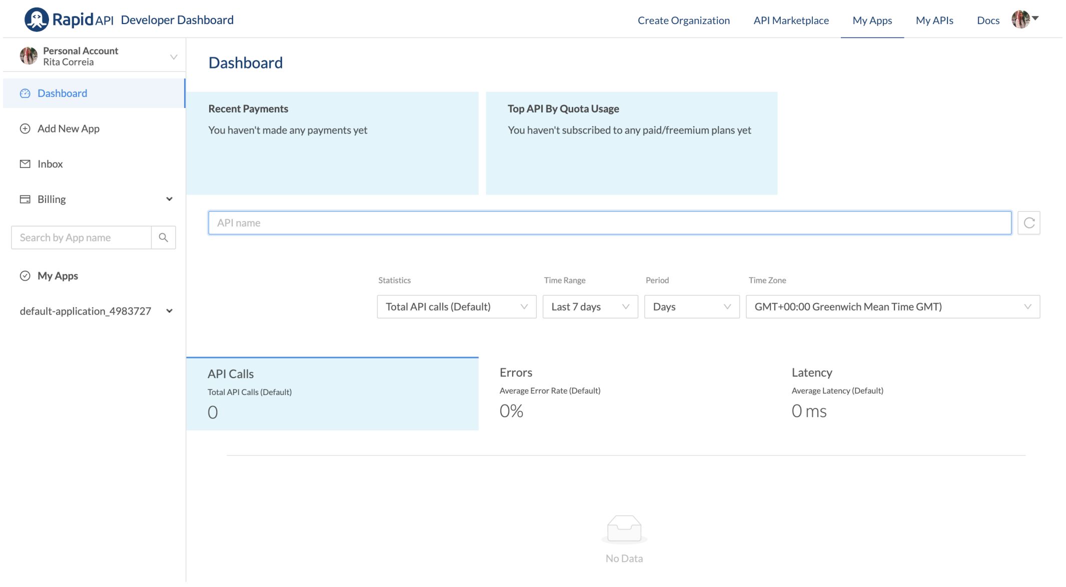
(Node.js) Unirest · Install SDK · Copy Code

```
var unirest = require("unirest");

var req = unirest("GET", "https://api-hoaxy.p.rapidapi.com/articles");

req.query({
  "query": "pizzagate AND date_published:[2016-10-28 TO 2016-12-04]",
  "sort_by": "relevant",
  "use_lucene_syntax": "true"
});

req.headers({
  "x-rapidapi-key": "44340b241emsh3f9ef4d3b8f90c7p143659jsne845d47e2fbc",
  "x-rapidapi-host": "api-hoaxy.p.rapidapi.com",
  "useQueryString": true
});
```

```
{ 4 items
  "articles" : [ 100 items
```

Figure 2.13: Testing endpoints in Hoaxy.

As we can see in figure 2.14, this API provides clear visibility as it allows to monitor call volumes and corresponding billing charges for all APIs in one dashboard, it ensures the app's uptime by keeping track of API errors and trends in latency and it also debugs faster by searching and viewing logs for the API calls.



Personal Account
Rita Correia

default-application_4983727

Dashboard

Recent Payments: You haven't made any payments yet

Top API By Quota Usage: You haven't subscribed to any paid/freemium plans yet

Statistics: Total API calls (Default) · Time Range: Last 7 days · Period: Days · Time Zone: GMT+00:00 Greenwich Mean Time GMT)

API Calls — Total API Calls (Default): 0

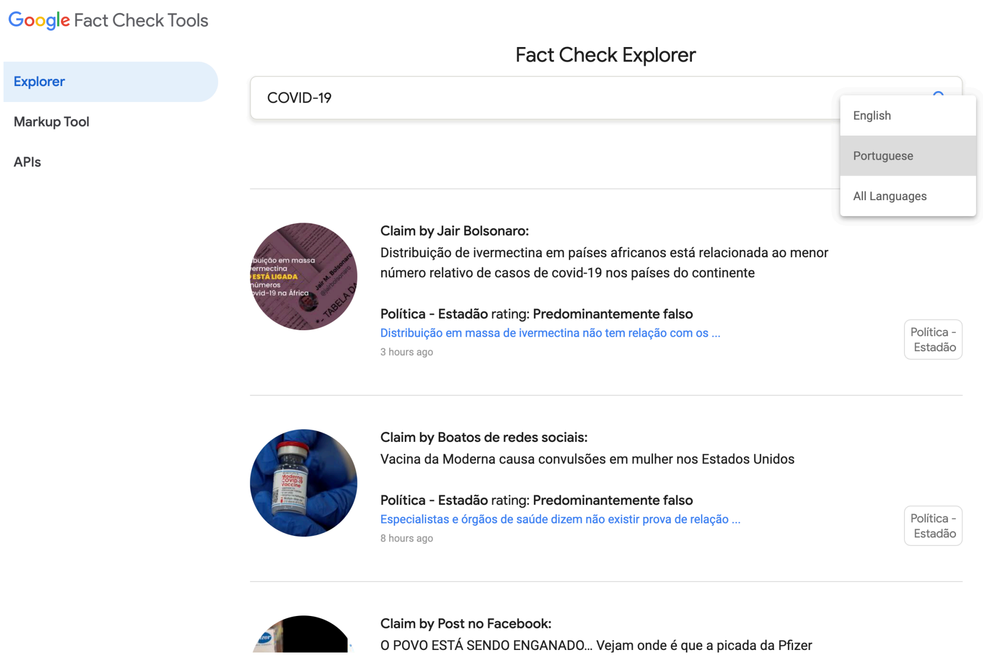
Errors — Average Error Rate (Default): 0%

Latency — Average Latency (Default): 0 ms

Figure 2.14: RapidAPI dashboard.

2.3.7 Fact Check Tools API

Google Fact Check Tools consist of two tools: Fact Check Explorer and Fact Check Markup Tool. Both tools aim to facilitate the work of fact-checkers, journalists and researchers. The Fact Check Explorer allows people to easily browse and search for fact checks [Goo21c]. As we can see in figure 2.15, we can search for a politician's statement or a specific topic. It is also possible to restrict results to a specific publisher.



COVID-19

Claim by Jair Bolsonaro:
Distribuição de ivermectina em países africanos está relacionada ao menor número relativo de casos de covid-19 nos países do continente

Política - Estadão rating: Predominantemente falso
Distribuição em massa de ivermectina não tem relação com os ...
3 hours ago
Política - Estadão

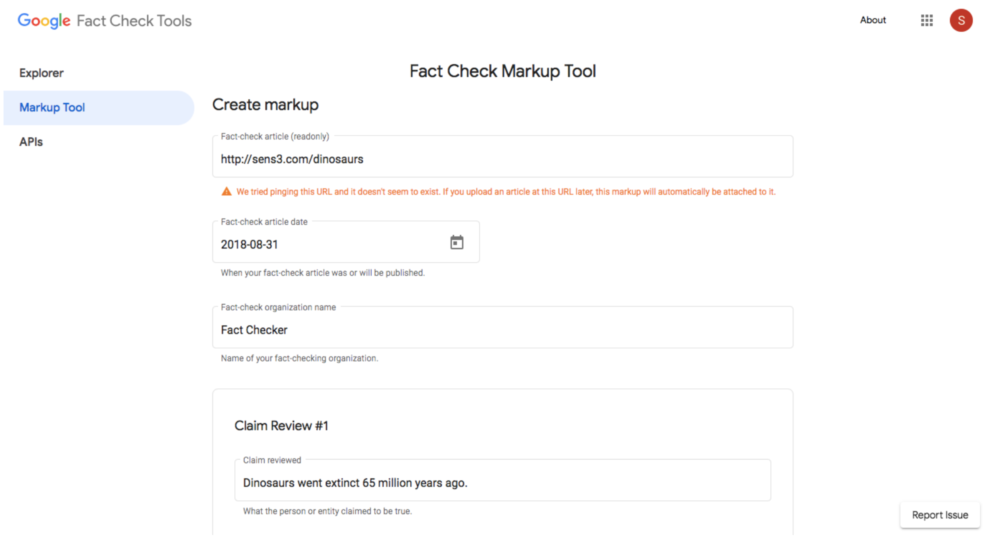
Claim by Boatos de redes sociais:
Vacina da Moderna causa convulsões em mulher nos Estados Unidos

Política - Estadão rating: Predominantemente falso
Especialistas e órgãos de saúde dizem não existir prova de relação ...
8 hours ago
Política - Estadão

Claim by Post no Facebook:
O POVO ESTÁ SENDO ENGANADO... Vejam onde é que a picada da Pfizer

Figure 2.15: Google Fact Check Explorer [Goo21b].

The Fact Check Markup Tool makes the process of creating `ClaimReview` markup easier, by allowing markup to be submitted via a simple Web form without the need to add anything to the article itself [Goo21c], as we can see in figure 2.16.



Create markup

Fact-check article (readonly)
http://sens3.com/dinosaurs
We tried pinging this URL and it doesn't seem to exist. If you upload an article at this URL later, this markup will automatically be attached to it.

Fact-check article date
2018-08-31
When your fact-check article was or will be published.

Fact-check organization name
Fact Checker
Name of your fact-checking organization.

Claim Review #1

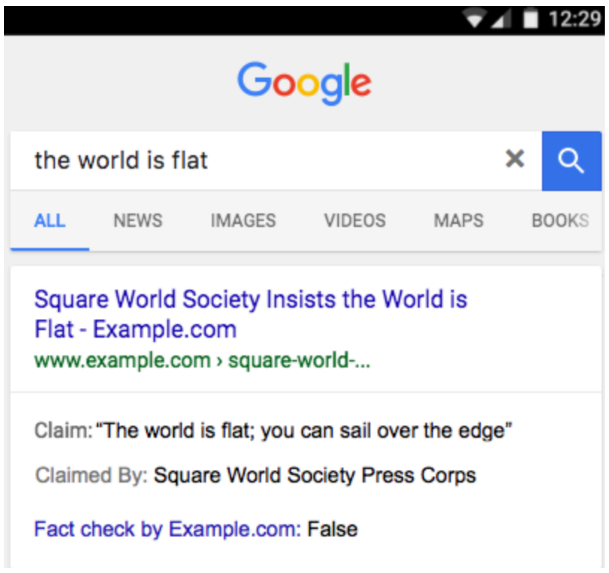
Claim reviewed
Dinosaurs went extinct 65 million years ago.
What the person or entity claimed to be true.

Figure 2.16: Google Fact Check Markup Tool - Web form [Goo21c].

There are some typical use cases for Google FactCheck ClaimReview Read/Write API such as adding `ClaimReview` markup, editing `ClaimReview` Markup and deleting `ClaimReview` Markup for a site's fact checking articles URL [Goo21d].

In a Web page that reviews a claim made by others, it is possible to include `ClaimReview` structured data on the Web page. `ClaimReview` structured data can enable a summarized version of the fact check to display in Google Search results when the page appears in search results for that claim [Goo21a].

Let's consider a page that evaluates the claim that the Earth is flat. The figure 2.17 shows what a search for "the world is flat" might look like in Google Search results, if the page provides a `ClaimReview` element [Goo21a].



Square World Society Insists the World is Flat - Example.com
www.example.com › square-world-...

Claim: "The world is flat; you can sail over the edge"

Claimed By: Square World Society Press Corps

Fact check by Example.com: False

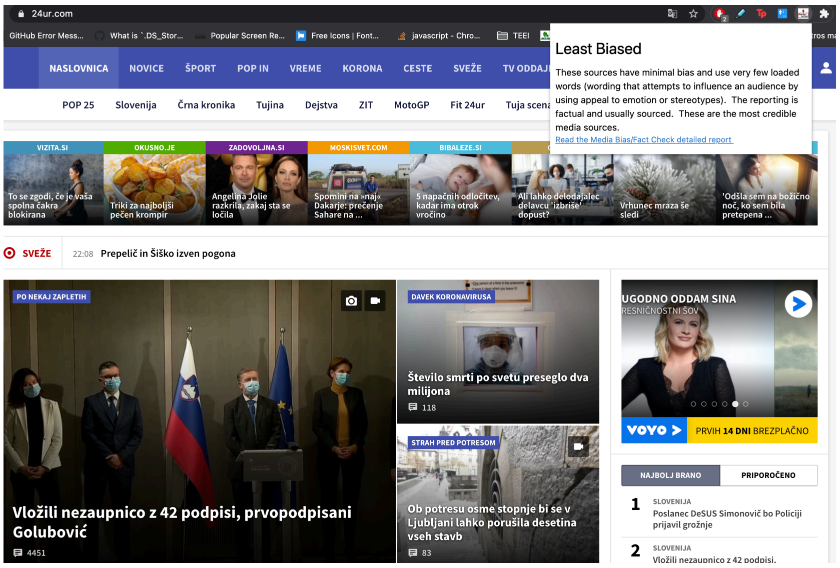
Figure 2.17: Example of a Google search using `ClaimReview` element [Goo21a].

2.3.8 Media Bias/Fact Check Extension

This Chrome extension shows Media Bias/Fact Check ratings for Facebook, Twitter, and news websites as we browse [Mik21]. There are currently more than 3500 media sources listed in their DB and growing every day [Med21].

With this extension, the rated sites can be analyzed within 9 categories:

- **Least Biased**: these sources have minimal bias and use very few loaded words (wording that attempts to influence an audience by using appeal to emotion or stereotypes). The reporting is factual and usually sourced. These are the most credible media sources [Med21]. The figure 2.18 is an example of a page rated in the Least Biased.


Least Biased

These sources have minimal bias and use very few loaded words (wording that attempts to influence an audience by using appeal to emotion or stereotypes). The reporting is factual and usually sourced. These are the most credible media sources.

Read the Media Bias/Fact Check detailed report

24ur.com
NASLOVNICA
NOVICE
ŠPORT
POP IN
VREME
KORONA
CESTE
SVEŽE
SVEŽE 22:08 Prepelič in Šiško izven pogona
Vložili nezaupnico z 42 podpisi, prvopodpisani Golubović
Število smrti po svetu preseglo dva milijona
Ob potresu osme stopnje bi se v Ljubljani lahko porušila desetina vseh stavb
NAJBOLJ BRANO
PRIPOROČENO

Figure 2.18: Example of a Least Biased rated page.

- **Left Bias**: these media sources are moderately to strongly biased toward liberal causes through story selection and/or political affiliation. They may utilize strong loaded words, publish misleading reports and omit reporting of information that may damage liberal causes. Some sources in this category may be untrustworthy [Med21].

- **Left-Center Bias**: these media sources have a slight to moderate liberal bias. They often publish factual information that utilizes loaded words to favor liberal causes. These sources are generally trustworthy for information but may require further investigation [Med21].

- **Right-Center Bias**: these media sources are slightly to moderately conservative in bias. They often publish factual information that utilizes loaded words to favor conservative causes. These sources are generally trustworthy for information but may require further investigation [Med21].

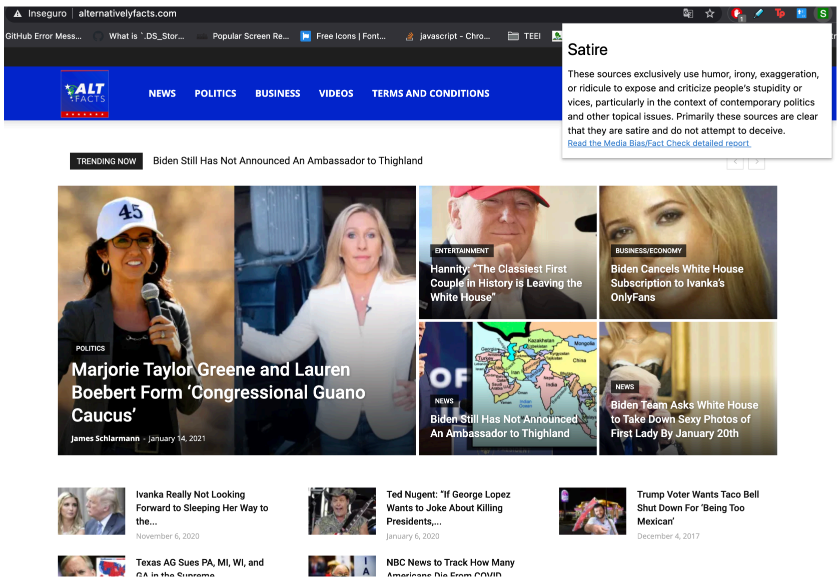
- **Right Bias**: these media sources are moderately to strongly biased toward conservative causes through story selection and/or political affiliation. They may utilize strong loaded words, publish misleading reports and omit reporting of information that may damage conservative causes. Some sources in this category may be untrustworthy.

- **Conspiracy-Pseudoscience**: sources in this category may publish unverifiable information that is not always supported by evidence. These sources may be untrustworthy for credible/verifiable information, therefore fact checking and further investigation is recommended on a per article basis when obtaining information from these sources [Med21].

- **Questionable Sources**: a questionable source exhibits one or more of the following: extreme bias, consistent promotion of propaganda/conspiracies, poor or no sourcing to credible information, a complete lack of transparency and/or is fake news. Sources listed in this category may be very untrustworthy and should be fact checked on a per article basis. Please note sources on this list are not considered fake news unless specifically written in the reasoning section for that source [Med21].

- **Pro-Science**: these sources consist of legitimate science or are evidence based through the use of credible scientific sourcing. Legitimate science follows the scientific method, is unbiased and does not use emotional words. These sources also respect the consensus of experts in the given scientific field and strive to publish peer reviewed science. Some sources in this category may have a slight political bias, but adhere to scientific principles [Med21].

- **Satire**: these sources exclusively use humor, irony, exaggeration, or ridicule to expose and criticize people's stupidity or vices, particularly in the context of contemporary politics and other topical issues. Primarily these sources are clear that they are satire and do not attempt to deceive [Med21]. In the figure 2.19 is a example of a page rated in Satire.


Inseguro | alternativelyfacts.com
GitHub Error Mess...
What is `.DS_Stor...
Popular Screen Re...
Free Icons | Font...
javascript - Chro...
TEEI
ALT FACTS
NEWS
POLITICS
BUSINESS
VIDEOS
TERMS AND CONDITIONS
TRENDING NOW
Biden Still Has Not Announced An Ambassador to Thighland
POLITICS
Marjorie Taylor Greene and Lauren Boebert Form 'Congressional Guano Caucus'
James Schlarmann - January 14, 2021
ENTERTAINMENT
Hannity: "The Classiest First Couple in History is Leaving the White House"
BUSINESS/ECONOMY
Biden Cancels White House Subscription to Ivanka's OnlyFans
NEWS
Biden Still Has Not Announced An Ambassador to Thighland
NEWS
Biden Team Asks White House to Take Down Sexy Photos of First Lady By January 20th
Ivanka Really Not Looking Forward to Sleeping Her Way to the...
November 6, 2020
Ted Nugent: "If George Lopez Wants to Joke About Killing Presidents,...
January 6, 2020
Trump Voter Wants Taco Bell Shut Down For 'Being Too Mexican'
December 4, 2017
Texas AG Sues PA, MI, WI, and
NBC News to Track How Many

Satire

These sources exclusively use humor, irony, exaggeration, or ridicule to expose and criticize people's stupidity or vices, particularly in the context of contemporary politics and other topical issues. Primarily these sources are clear that they are satire and do not attempt to deceive.
Read the Media Bias/Fact Check detailed report

Figure 2.19: Example of a Satire rated page.

2.3.9 NewsCracker Extension

The NewsCracker Chrome extension allows receiving the accuracy and spin ratings for an article or news, on a scale from 0-10. According to the developer, by checking an article for potential inaccuracy or bias, we can stay away from fake news and remain a savvy, informed news consumers while browsing an internet increasingly dominated by misleading content [jch21].
Plus, if a user disagrees with the rating of an article he is viewing, he can "flag" the rating, which will prompt the developer to look into the rating and see what went wrong [jch21]. The extension scores are generated by an algorithm, so they are not absolute indicators of

accuracy or inaccuracy, just as they are not absolute indicators of subjectivity or objectivity, according to the developer [jch21].
It should be noted that this extension was tested in portuguese (*Público* and *Jornal de Notícias*), french (*Le Monde*), spanish (*El Mundo* and *El País*), british (*BBC News* and *The Guardian*) and american (*The New York Times* and *USA Today*) online newspapers, and it only worked in 2 of them, these being the *The New York Times* and the *The Guardian*, as we can see in figures 2.20 and 2.21, respectively.

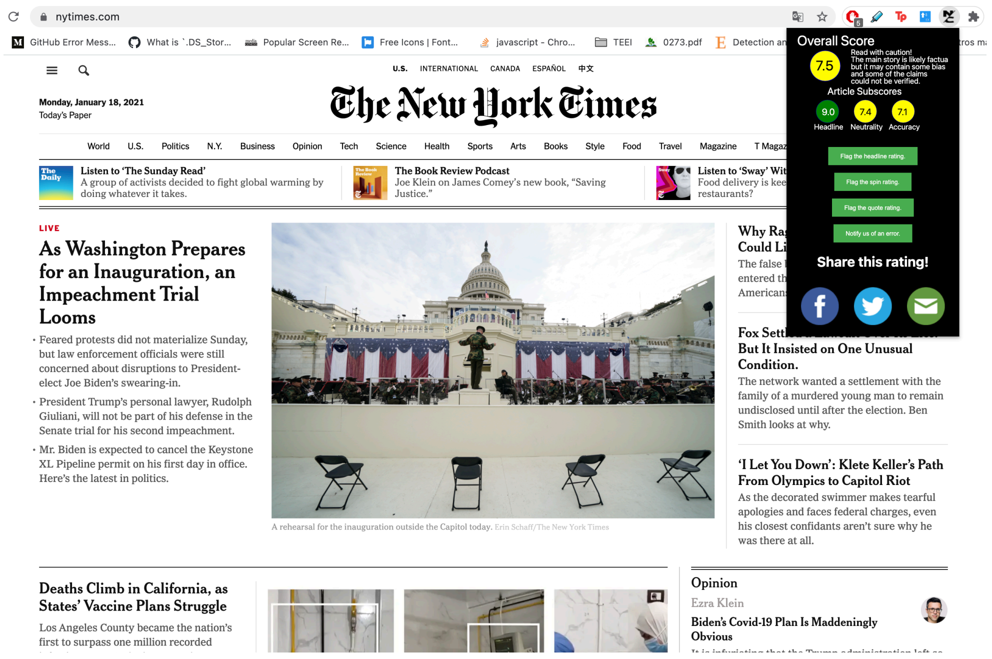

Overall Score
7.5
Read with caution!
The main story is likely factua
but it may contain some bias
and some of the claims
could not be verified.
Article Subscores
9.0 Headline
7.4 Neutrality
7.1 Accuracy

Figure 2.20: NewsCracker extension scoring the *The New York Times* newspaper.

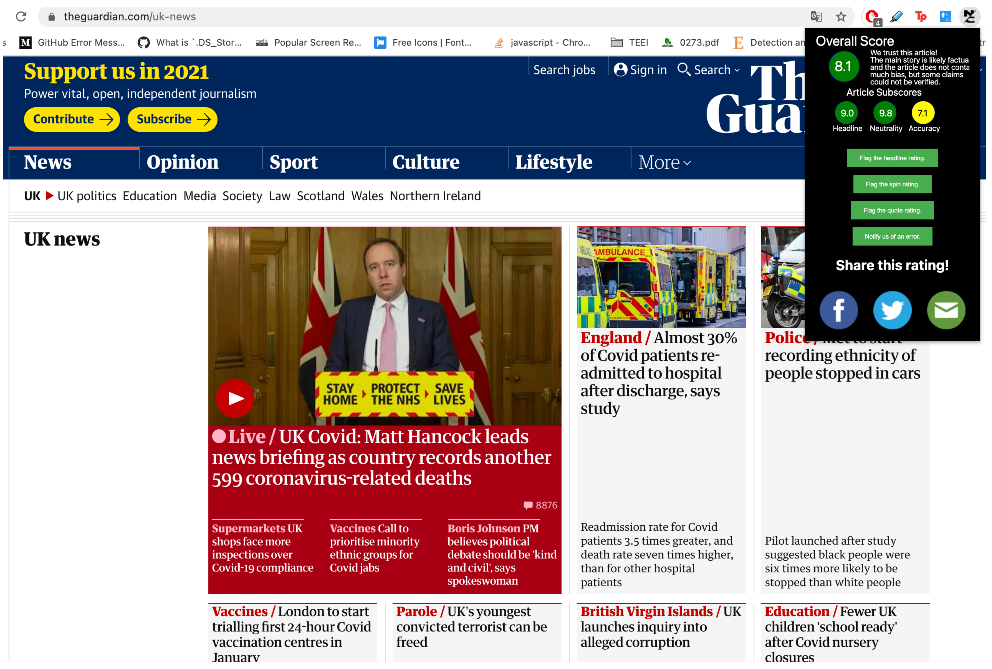

Overall Score
8.1
We trust this article!
The main story is likely factua
and the article does not conta
much bias, but some claims
could not be verified.
Article Subscores
9.0 Headline
9.8 Neutrality
7.1 Accuracy

Figure 2.21: NewsCracker extension scoring the *The Guardian* newspaper.

As we can see, the extension evaluates an article or news according to 3 parameters:

Headline, Neutrality, and Accuracy, giving an overall score at the end. In this case, *The Guardian* newspaper obtained a better overall score compared to the *The New York Times*.

2.4 Conclusion

In this chapter, we first discussed the differences between Native Apps, Web Apps, and PWA, presenting some pros and cons for each one of them. About this topic, we concluded that with the use of progressive enhancement, new capabilities are enabled in modern browsers, making the application more reliable and installable by anyone, anywhere, on any device with a single codebase.

Mobile devices are transforming the ways that journalism is practiced, with the rise of a new participant, the prosumer. The Internet and various types of mobile devices have already reorganized the newsroom by introducing new means to connect with the audience and to communicate with other journalists within the same place. This fake news occurrence is due to this huge information sharing through the network, where a lot of misinformation and distorted content is emerging as one of the most threatening harms on social media. Fake news can be used by malicious entities to manipulate users' options and decisions in a large range of things such as online shopping, health-care options, presidential elections, etc. As so, fake news detection mechanisms are a very important but challenging task. In this chapter, we discussed several relevant topics in the fake news world, such as its main characteristics and components. We also analyzed the impact of fake news in Ad Tech Firms, and why they are not considered such a bad thing to most of them. We additionally compared different approaches and existing methods for fake news and distorted image detection. As discussed, ML and AI algorithms are not enough to detect fake news and misinformation, being the human lead in this matter a significant piece.

Besides, it was approached, in this chapter, some relevant APIs, such as Fact-checker API, from Oracle, Hoaxy API, and Fact Check Tools API, from Google.

Finally, 2 extensions for Chrome were presented, demonstrating its way of use through screenshots. The first one (Media Bias/Fact Check) proved to be more wide and precise in terms of evaluation aspects of the article or news compared to the second (NewsCracker), which appears to work for a smaller range of pages and with fewer evaluation parameters, however, with a more direct and objective classification to the user.

Chapter 3

Conceptual Design

3.1 Introduction

In this chapter, it will be presented the conceptual design of the system and the mechanisms used in the domain of Software Engineering, mainly the functional and non-functional requirements of the tool, the use-case and activity diagrams, the relational DB model, the system architecture, and the technologies that were used to develop and implement the project.

Thus, the chapter is divided as follows:

1. The first section – **Introduction** – makes the introduction and division of the chapter and subjects that will be approached.
2. The second section – **Requirements** – defines all the functional and non-functional requirements of both the extension and platform.
3. The third section – **Use-Cases** – specifies the concept and shows the use-cases in the field of the application and the adjacent platform.
4. The fourth section – **Activity Diagram** – represents the activity diagram of the extension and its concept.
5. The fifth section – **Database** – introduces the DB, as well as the concept of ERD and its implementation on the system.
6. The sixth section – **System's Architecture** – shows and explains the system architecture and the definition of RESTful API.
7. The seventh section – **Used Technologies** – approaches the technologies that were used in the development and implementation of the project.
8. The seventh section – **Conclusion** – reflects on the main conclusions of the chapter.

3.2 Requirements

This section will address the functional and non-functional requirements that were considered during the development of the software.
The Functional Requirements (FR) explicitly describe the features and services of the system, documenting how the system should react to specific inputs, how it should behave

in certain situations and what should not happen. They should not have contradictory definitions.
On the other hand, the Non-Functional Requirements (NFR) defines the system properties and restrictions such as security, performance, and disk space. These can refer to the entire system or individual parts of it. If the non-functional requirements are not met, the system becomes useless [Edu].
All these requirements were met both in the project proposal and with the help of my project coordinator.

3.2.1 Functional Requirements

The requirements presented in this subsection are related to the extension itself and to the informative platform, so they contain the complete, clear, and numbered specifications of all the software needs for its correct functioning, allowing the graphical representation of the project to be the as correct as possible and the one idealized by the user.

3.2.1.1 Extension

Here it will be presented the FR of the extension that is connected to the browser and with which the users will directly interact.
The table 3.1 describes the order in which these requirements were met and will be covered.
As the extension has three different interfaces, we can describe the FR of the login page, where the user authenticates, the register page, where the user can create an account in order to use the developed extension, and the main page, which is the page that the user sees after his login on the extension and use to interact and express opinions on some online text.

Requirement ID	Requirement Name	Requirement Description
FR01	Login	Contains all the requirements associated with FR01.X. The user does not have to have an account to access this view.
FR02	Register	Contains all the requirements associated with FR02.X. The user does not have to have an account to access this view.
FR03	Main page	Contains all the requirements associated with FR03.X. The user must have an account to access this view.

Table 3.1: Organization of the functional requirements of the extension.

The following table describes particularly the FR of the login page:

ID FR01.X	Requirement Name	Requirement Description
FR01.1	Title and subtitle	At the top of the form referred to in FR01.4, must be the name of the extension, centered on the page and with the color black, along with a message below in gray and with a lower size letter, saying ”Please enter your account details:”.
FR01.2	User’s email	The fulfillment of this field by the user is mandatory. If this field is left blank when clicking the login button, it becomes red with a message inside saying ”Enter an email.”. If the user enters a badly formatted email (without the ”@”), an error message should be displayed indicating the insertion of the ”@” in the email address. If the email does not have any characters after the ”@”, an error message for that purpose should be displayed. If the email does not exist, an authentication failure message should be displayed indicating that the email or password is incorrect.
FR01.3	User’s password	The fulfillment of this field by the user is mandatory. If this field is left blank when clicking the login button, it becomes red with a message inside saying ”Enter a password.”. It includes alphanumeric coding, which can contain capitalized or not capitalized letters. The use of special characters is allowed. If the password does not exist, an authentication failure message should be displayed indicating that the email or password is incorrect.

FR01.4	Form	The data referring to FR01.2 and FR01.3 must be entered in a blank white form, centered on the extension page. It should be the first thing that the user sees when clicking on the extension icon. Right after the form, there should be a hyperlink to the user registration page referred to in FR02.X., with the word ”Register”.
FR01.5	Login button	It must be centered on the page, immediately after the form referred in FR01.4, and lead to the main page of the extension if the user correctly authenticates.

Table 3.2: Functional requirements of the extension login page.

The following table describes particularly the FR of the register page:

ID FR02.X	**Requirement Name**	**Requirement Description**
FR02.1	Title and subtitle	At the top of the form referred to in FR02.6, must be the name of the extension, centered on the page and with the color black, along with a message below in gray and with a lower size letter, saying ”Create an account:”.
FR02.2	User’s name	The fulfillment of this field by the user is mandatory. If this field is left blank when clicking the register button, it becomes red with a message inside saying ”Enter a name.”. Covers alphanumeric coding, which may contain capitalized letters or not.

FR02.3	User's email	The fulfillment of this field by the user is mandatory. If this field is left blank when clicking the login button, it becomes red with a message inside saying ”Enter an email.”. If the user enters a badly formatted email (without the ”@”), an error message should be displayed indicating the insertion of the ”@” in the email address. If the email does not have any characters after the ”@”, an error message for that purpose should be displayed. If the email already exists in the DB, a failure message should be displayed indicating that the email is already in use.
FR02.4	User's password	The fulfillment of this requirement by the user is mandatory. If this field is left blank when clicking the login button, it becomes red with a message inside saying ”Enter a password.”. It covers alphanumeric coding, which can contain capitalized or uncapitalized letters. The use of special characters is allowed. There must be a confirmation field for the password (FR02.5). When submitting the form, both fields (password and password confirmation) must match. If the passwords entered are different, then an error message should appear with the message ”Passwords do not match!”.
FR02.5	User's password confirmation	The fulfillment of this requirement by the user is mandatory. If this field is left blank when clicking the login button, it becomes red with a message inside saying ”Confirm your password.”. It covers alphanumeric coding, which can contain capitalized or uncapitalized letters. The use of special characters is allowed.

FR02.6	Form	The data referring to FR02.2, FR02.3, FR02.4, and FR02.5, must be entered in a blank white form, centered on the extension register page. It should be the first thing that the user sees when clicking on the ”Register” hyperlink on the login page. Right after the form, there should be a hyperlink to the user login page, referred to in FR01.X, with the word ”Back”.
FR02.7	Register button	It must be centered on the page, immediately after the form referred to in FR02.6, and lead to the login page of the extension if the user correctly registers.

Table 3.3: Functional requirements of the extension register page.

The following table describes particularly the FR of the main page:

ID FR03.X	Requirement Name	Requirement Description
FR03.1	Title and subtitle	At the top of the main extension page must be the name of the extension, centered on the page and with the color black, along with a message below in gray and with a lower size letter, saying ”Please check the content that you consider to be fake or questionable, by first selecting it on the website and then clicking on the respective extension button.”.
FR03.2	Fake content button	It should be below the FR03.1 and must have the color red with a logo inside which explicitly refers to false content. When hovering the mouse in the button, should appear the message ”Fake”. After the user selects the respective content in the webpage and clicking this button, that text should become highlighted in red.

FR03.3	Questionable content button	It should be below the FR03.1 alongside the button referred to in FR03.2 and must have the color yellow with a logo inside which explicitly refers to questionable content. When hovering the mouse in the button, should appear the message ”Questionable”. After the user selects the respective content in the webpage and clicking this button, that text should become highlighted in yellow.
FR03.4	Body	In the main page body, below the buttons referred to in FR03.2 and FR03.3, there should be two text areas, one below the other, with the titles ”Fake content:” and ”Questionable content:”, with the fake and questionable text, respectively, that the user previously highlighted.

Table 3.4: Functional requirements of the extension main page.

3.2.1.2 Platform

Here will be presented the FR of the developed platform, which is connected to the Chrome extension DB.
The table 3.5 describes all the FR of the platform page, which is the website that shows the extension insights for the content producers to see in order to fact-check the appropriate content.

ID FR01.X	**Requirement Name**	**Requirement Description**
FR01.1	Title and subtitle	At the top of the page must be the name of the extension, centered on the page and with the color black, along with a message below in gray and with a lower size letter, saying ”Check your insights.”.

FR01.2	Nav-tab	The body of the page should consist of a nav-tab, with at least two tabs with different content, referring to the fake and questionable text, respectively. Each tab must contain a table (FR01.3) with the extension relevant information.
FR01.3	Table	Depending on the tab, the table should have a title referring to the content we are seeing (e.g. Fake content or Questionable content). The word ”Fake” should be in red and the word ”Questionable” should be in yellow, while the others should be black. The table body in each tab should present the ID (#) of the content in the DB, the content itself that the users selected as fake or questionable (depending on the tab), alongside the name and email of the respective user and the active URL of the page that was used. The tables should be paginated and have the options to choose how many entries we want to see at the same time, to search by words, to order up or down the information of each column, the ”First”, ”Previous”, ”Next” and ”Last” buttons to easily navigate through the table and the page number of the table that we currently are seeing.

Table 3.5: Functional requirements of the platform main page.

3.2.2 Non-Functional Requirements

These types of requirements are responsible for informing about the performance, usability, reliability, security, availability, maintenance, and technologies involved in the system. They also deal with external factors that affect the system and development process.
In this section, it will be approached the NFR of both the browser extension and the platform for the content producers.

3.2.2.1 Extension

Here it will be presented the NFR of the extension, that is connected to the browser, and with which the extension users will directly interact, highlighting the content that they consider to be either fake or questionable on several websites.

The table 3.6 describes some of the extension NFR found.

NFR ID	Requirement Name	Requirement Description
NFR01	Working hours	The extension must be available every day and all the time, except when it is in maintenance periods, which must be short.
NFR02	Search and input time	The extension should take no more than 1 second to perform a search or data entry in the DB.
NFR03	Availability	The extension should be available in a computer or mobile device.
NFR04	Usability	The system must be simple and intuitive. It should not be necessary to have training for its correct use. The user should be able to use the software after a brief explanation of how it works.
NFR05	Connection	A Internet connection is required for the use of the extension.
NFR06	OS	The operating system must be indifferent.
NFR07	Browser	The browser should be a recent stable version of Google Chrome.

Table 3.6: Non-functional requirements of the extension.

3.2.2.2 Platform

Here it will be presented the main NFR of the platform, that is directly connected to the Chrome extension DB, to which the allowed content producers will access in order to fact-check the previous highlighted information by the extension users.

The table 3.7 describes some of the platform NFR found.

NFR ID	Requirement Name	Requirement Description
NFR01	Working hours	The platform must be available every day and all the time, except when it is in maintenance periods, which must be short.
NFR02	Search time	The platform should take no more than 1 second to perform a search in the DB.
NFR03	Availability	The extension should be available in a computer, tablet or mobile device.
NFR04	Usability	The system must be simple and intuitive. It should not be necessary to have training for its correct use. The user should be able to use the software after a brief explanation of how it works.
NFR05	Connection	A Internet connection is required for the use of the platform.
NFR06	OS	The operating system must be indifferent.
NFR07	Browser	To access the platform you need a modern browser, such as: Microsoft Internet Explorer version 11 or higher; Apple Safari version 9 or higher; Google Chrome version 46 or higher; Firefox version 45 or higher.

Table 3.7: Non-functional requirements of the platform.

3.3 Use-Cases

In this section, the use-case diagrams considered during the extension and platform development will be presented. The use-case diagrams represent a possible use of the system by an actor, which can be a person, physical device, mechanism, or subsystem that interacts with the target system, using some of its services, and can be identified based on the requirements. In these kinds of diagrams, it can be two types of actors: the main actor, who interacts directly with the system, and the secondary actor who interacts with other actors.

As we can see in the figure 3.1, the regular extension user, which is the system actor, after register and correctly authenticate into his account, can highlight content/text excerpts on several websites that he considers to be either fake or questionable.

The content is then stored in a DB, along with the user name, email, and URL of the

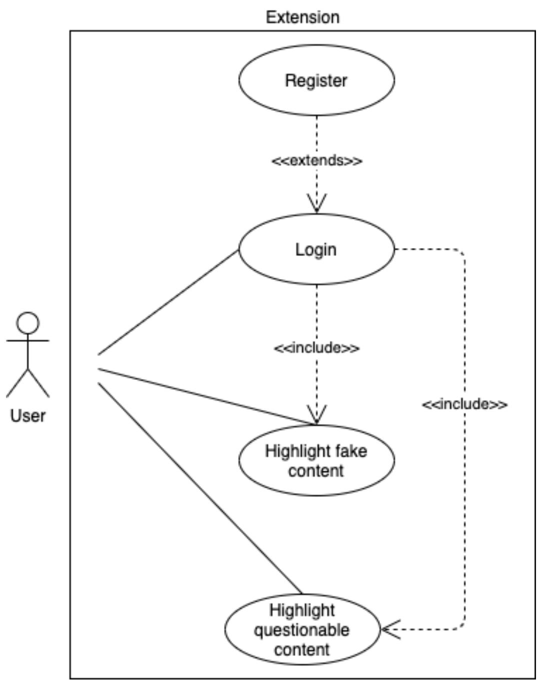


Figure 3.1: Use-case diagram for the extension.

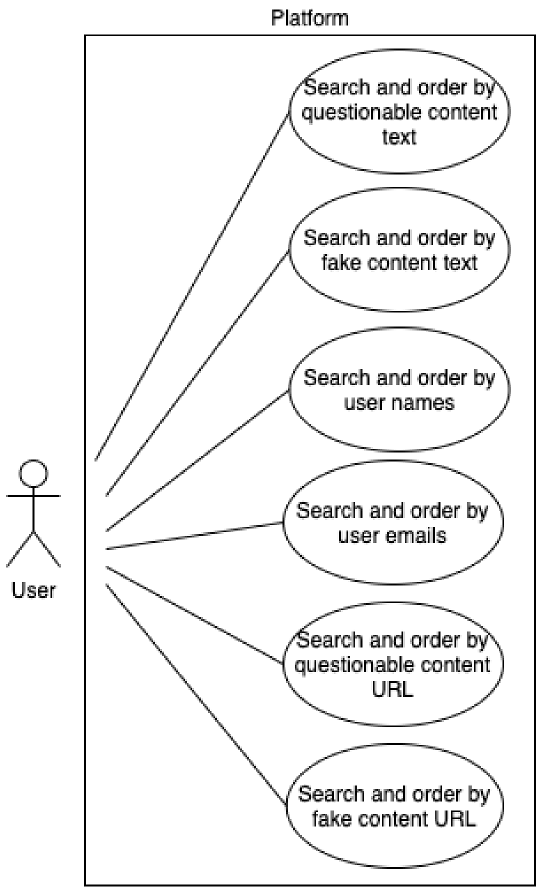

Platform
Search and order by questionable content text
Search and order by fake content text
Search and order by user names
Search and order by user emails
Search and order by questionable content URL
Search and order by fake content URL
User

Figure 3.2: Use-case diagram for the platform.

respective page. That information can then be then visualized in a user-friendly platform for the content producers, which in this case are the system actors, so they can analyze it, fact-check the information and proceed most adequately (figure 3.2).

3.4 Activity Diagram

An activity diagram is an important **behavioral diagram** in Unified Modeling Language (UML). It is essentially an advanced version of a flow chart that modeling the flow from one activity to another, so it focuses on the condition of flow and the sequence in which it happens. This type of diagram portrays the control flow from a start point to a finish point, showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modeling where their primary use is to depict the dynamic aspects of a system.

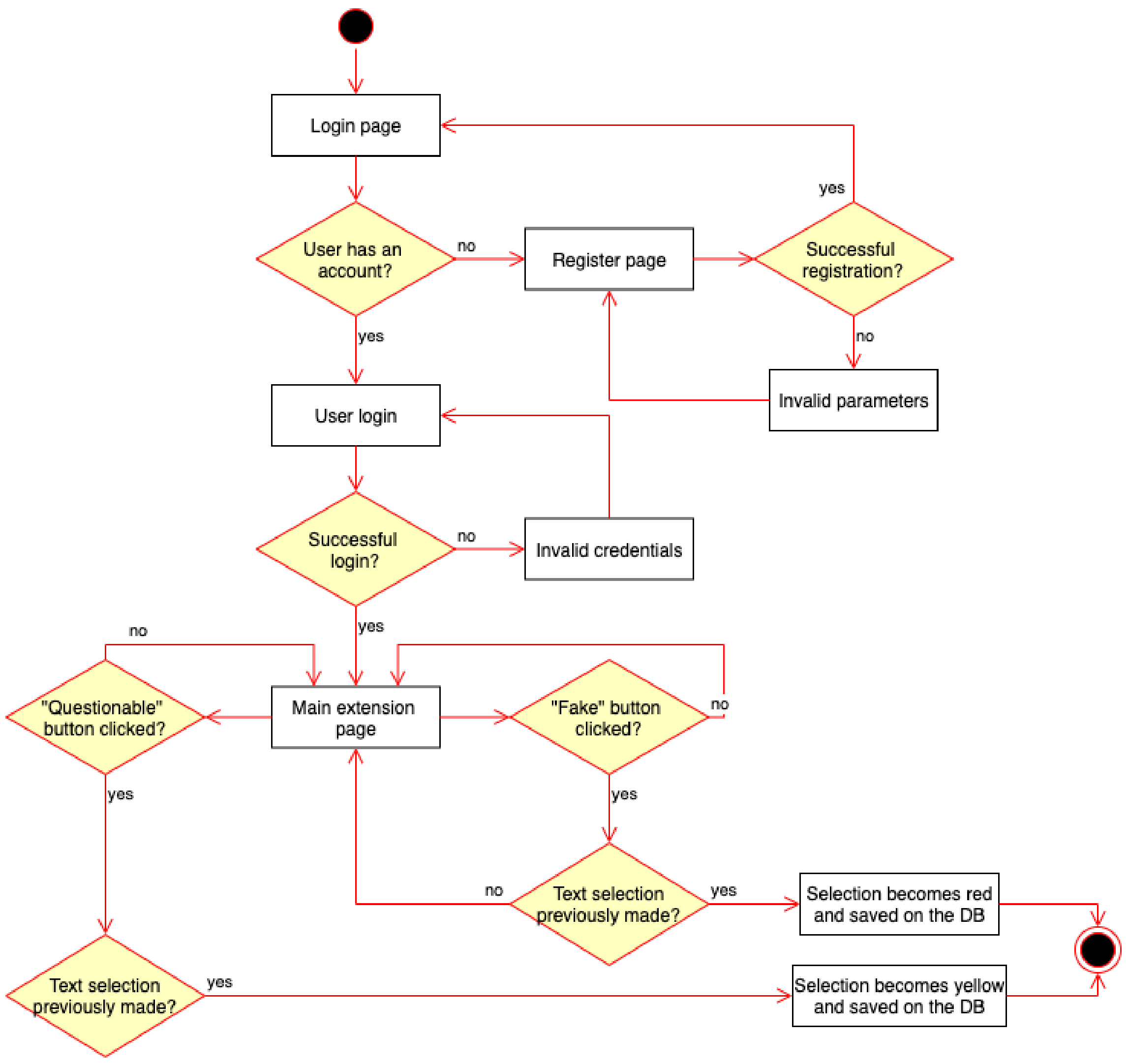


Figure 3.3: Extension activity diagram.

In figure 3.3, is represented the activity diagram of the extension system. As we can see, the first page that the user reaches after installing the extension is the login page. If no account has been created so far, the user must access the registration page. If the registration process is done successfully, the user returns to the login page where he can authenticate, otherwise, he remains on the same page. If the login credentials are correct, the user can finally view the main page of the extension, otherwise, he remains on the login page. In order for the user to classify some content as fake or questionable, he must first do that text selection on the browser and then click on the respective extension button. If a button is pressed without a previous selection, an alert appears and the user remains on the main page, otherwise, the selection becomes red, if the user considers it to be fake, or yellow if he considers it to be questionable.

3.5 Database

The DB is one of the most important parts of the development of any project, as it stores all the data related to it. The implementation of this storage structure was a simple but continuous process, as it was changed a few times throughout the project's development. An ERD shows the logical structure of a DB, including the relationships and constraints that determine how data should be stored and accessed. In an ERD, the concept "Entity" can be defined as a real-world object, which is described by a set of attributes. Each attribute has an associated domain of possible values. Thus, a set of entities can be defined as a collection of similar entities, which can share attributes and associations. A key is the minimum set of attributes whose values uniquely identify each entity in the set. On the other hand, the concept of "Association" is defined as a relationship between two or more entities. The association can have descriptive attributes, where is information about this association. We can call an "instance" a set of associations [Alb]. Some advantages of this type of diagram are that it reveals possible inconsistencies, it has a phased construction "Top-Down" and the ease of communication that is created between Information Technology (IT) and users.

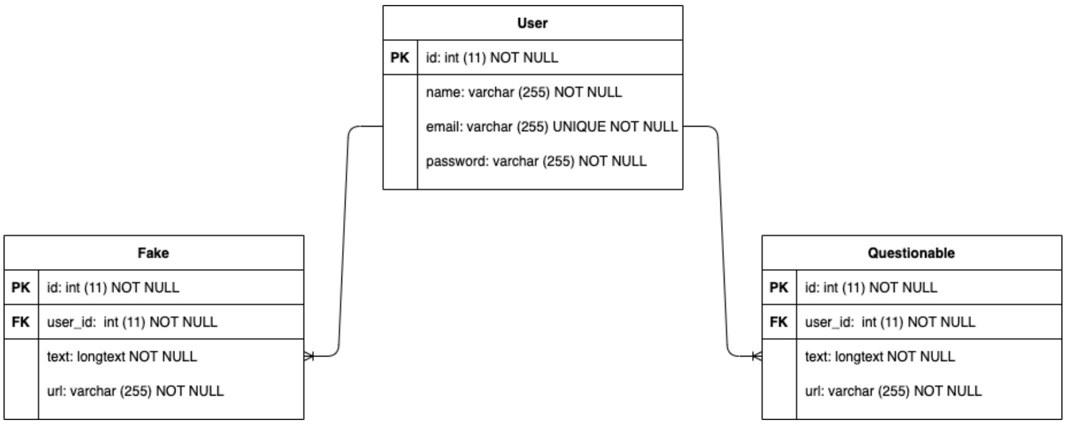

User
PK
id: int (11) NOT NULL
name: varchar (255) NOT NULL
email: varchar (255) UNIQUE NOT NULL
password: varchar (255) NOT NULL
Fake
PK
id: int (11) NOT NULL
FK
user_id: int (11) NOT NULL
text: longtext NOT NULL
url: varchar (255) NOT NULL
Questionable
PK
id: int (11) NOT NULL
FK
user_id: int (11) NOT NULL
text: longtext NOT NULL
url: varchar (255) NOT NULL

Figure 3.4: ERD of the system.

The schema represented in the figure 3.4, describes the final data implemented. As we can see, we have three tables: "Fake", "Questionable" and "User". The "Fake" one stores all the information that the users consider to be false as well as the user unique identifier and the respective page URL. The "Questionable" table is the same as the previous one but related to the content that the users consider to be questionable. The "User" table stores all the names, emails, and encrypted passwords of the users registered in the extension.

3.6 System's Architecture

The system's architecture style of this Master's Dissertation project is based on a **RESTful API** (more information on sub-section 3.6.1).
Here we can distinguish three different sides (see figure 3.5) - Extension, Server, and Journal.

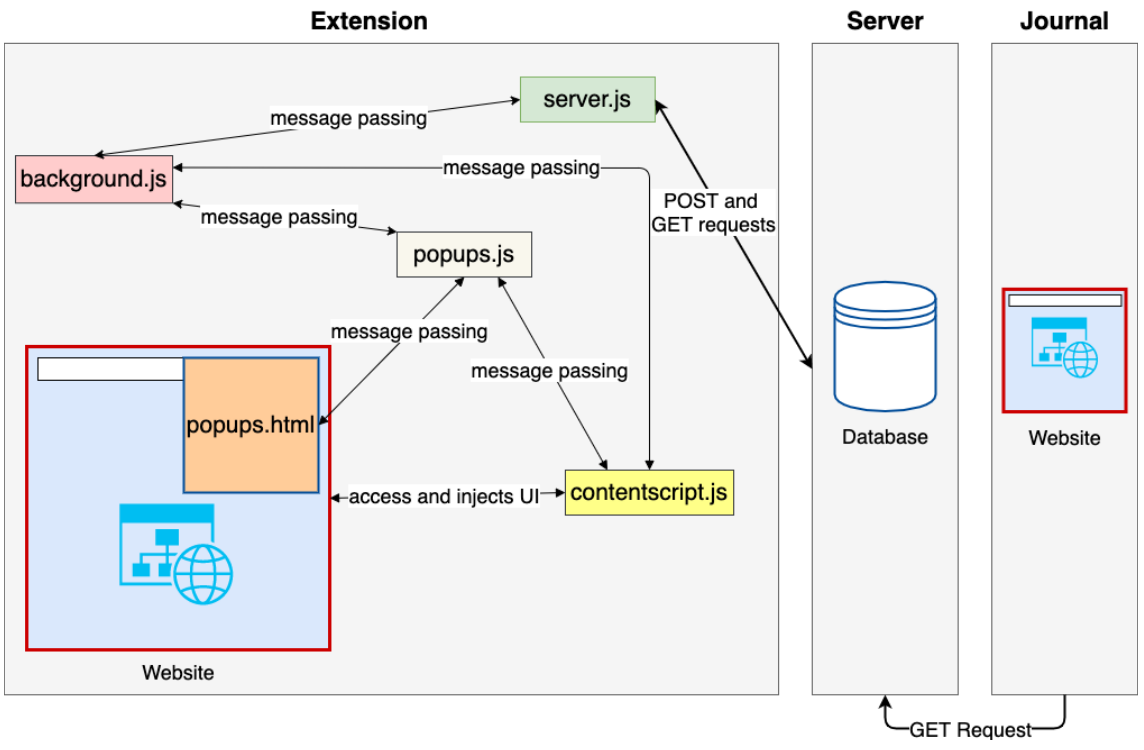


Figure 3.5: System's architecture.

The **Extension** side is where the Chrome extension itself is, operates, and interacts with the users. Here we can see three different popups interfaces (login, register, and main page) that interact with the respective popup scripts to make the pages dynamic. The popup script of the main page interacts directly with the background to get the previously stored user selections and print them on the extension, while the other two popup scripts (from login and register pages) interact with the background in order to send the user registration information and the authentication credentials. The authentication and registration process happens on the server.js due to a *fetch()* method from the background, which then will create endpoints that send `GET` and `POST` requests to the DB on the **Server** side. The popup script of the main page also interacts with the content script, sending the action codes from the buttons listeners (fake and questionable buttons). The content script will act from there as it accesses and injects code on the browser page. If the action

received from the popup script of the main page comes by clicking the fake button, the selection previously made by the user on the browser becomes red, otherwise, it becomes yellow. After that, the content script sends that selection to the background script which will communicate with the server.js through a *XMLHttpRequest* object in order to store that selected content on the DB, as well as the respective user name, email, and page URL. This happens through endpoints that send `GET` and `POST` requests from the server.js to the Database on the Server side.
Then, after the DB stores all the data, an endpoint from the **Journal** side to the DB is created, through a `GET` request. This last side will contain a website that will display all the relevant information for the journalist to fact-check and handle the appropriate content.

3.6.1 RESTful API

A REST API, also known as RESTful API, is an API that conforms to the constraints of REST architectural style and allows for interaction with RESTful Web services [Red20].
An API is a set of definitions and protocols for building and integrating application software. It spells out the proper way for a developer to write a program requesting services from an operating system or other application [Red20] [Ale20].
A RESTful API uses HTTP requests in order to access and use data. That data can be used to `GET`, `PUT`, `POST` and `DELETE` data types, which refers to the reading, updating, creating and deleting of operations concerning resources [Ale20]. When a client request is made via a REST API, it **transfers a representation of the state** of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text [Red20].
In order to be a RESTful API, a Web service must adhere to the following six REST architectural constraints:

- **Use of a uniform interface**: Resources should be uniquely identifiable through a single URL, and only by using the underlying methods of the network protocol, such as `DELETE`, `PUT` and `GET` with HTTP, should it be possible to manipulate a resource [Ale20]. They shouldn't be too large but should contain every piece of information that the client might need [IBM21];

- **Client-server based**: There should be a clear delineation between the client and server. UI and request-gathering concerns are the client's domain. Data access, workload management, and security are the server's domain. This loose coupling of the client and server enables each to be developed and enhanced independently of the other [Ale20];

- **Stateless operations**: All client-server operations should be stateless, and any state management that is required should take place on the client, not the server [Ale20];

- **RESTful resource caching**: When possible, resources should be cacheable on the client or server-side. Server responses also need to contain information about whether caching is allowed for the delivered resource. The goal is to improve performance on the client-side, while increasing scalability on the server-side [IBM21];
- **Layered system**: REST APIs allow for an architecture composed of multiple layers of servers [Ale20]. They need to be designed so that neither the client nor the server can tell whether it communicates with the end application or an intermediary [IBM21];
- **Code on demand (optional)**: REST APIs usually send static resources, but in certain cases, responses can also contain executable code (such as Java applets). In these cases, the code should only run on-demand [IBM21].

Though the REST API has these criteria to conform to, it is still considered easier to use than a prescribed protocol like SOAP (Simple Object Access Protocol), which has specific requirements like XML messaging, and built-in security and transaction compliance that make it slower and heavier [IBM21]. On the other hand, REST is a set of guidelines that can be implemented as needed, making REST APIs faster and more lightweight, with increased scalability—perfect for Internet of Things (IoT) and mobile app development [IBM21].

3.7 Used Technologies

In order to develop and correctly implement the system represented in the previous section, it was necessary several back-end and front-end technologies such as HTML,CSS, Bootstrap, PHP, JavaScript, jQuery, Node.js and its dependencies and packages, and MAMP.

3.7.1 HTML and CSS

As already referred in 2.2.2, HTML is a markup and interpretation language used in the construction of Webpages, which had its first implementation published by Tim Berners-Lee in late 1991. HTML together with CSS and JavaScript form the three main technologies of the Web. The browsers executed on the client-side receive the HTML documents from a Web server and interpret these documents through a compiler already implemented in the browser.
The CSS is a mechanism used to style HTML documents, which was developed by W3C in 1996, and arose from the need for a mechanism for formatting the webpage using tags, which HTML did not have. Thus, CSS separates the content from the visual representation of the site being possible, for example, to change the text and background color, the font, and the spacing between paragraphs. With this tool, it is also possible to create tables, use variations of layouts, adjust images, and so on [Ari21].
These two technologies were fundamental for the implementation of the things that the users see such as the extension popup and the platform pages on the Journal side.

3.7.2 Bootstrap

Bootstrap is an open-source tool, developed by designer Mark Otto and the developer Jacob Thornton, as a front-end framework that helps consistency in style for building responsive, mobile-first sites, with Bootstrap CDN and a template starter page [Booa] [Boob]. Through version 4.5.3 of this framework, it was possible to have some already formatted CSS files to customize several extension and platform components, such as buttons, forms, navigation bars, tables, among others.

3.7.3 PHP

PHP, is a server-side scripting language that is normally embedded in HTML. It is used to manage dynamic content, databases, session tracking, among others. PHP is integrated with many popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server [Tutb]. With this language it is possible to generate dynamic page content, create, open, read, write, delete, and close files on the server, collect form data, add, delete, modify data in your database, encrypt data, etc [W3Sb].
In the field of the project, the PHP was embedded in the **platform** main HTML file to directly access the DB, in order to show its content on a table on the Journal side page.

3.7.4 JavaScript

As also approached in 2.2.2, JavaScript is a dynamic computer programming language, which was used for both the client and the server-side of mainly the **extension** development, as it is an interpreted programming language with object-oriented capabilities [Tuta].
Thus, JavaScript is the core of the project. It was used to implement all the parts that a Chrome extension needs, such as the popup.js for the buttons events of the popup.html itself and the respective visual output for the user, the content-script.js, for the direct interaction with the popup.js, which handle the actions from those buttons, the popup-register.js that treats the user's registration process, the popup-sign-in.js so the users can properly login into the extension, the background.js page that directly interacts with the server.js, and the server itself which communicates with the DB.

3.7.5 jQuery

jQuery is a lightweight, ”write less, do more”, JavaScript library. The purpose of it is to make it much easier to use JavaScript. As so, jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish and wraps them into methods that you can call with a single line of code. It also simplifies a lot of the complicated things from JavaScript, as AJAX calls, DOM manipulation, HTML event methods, effects, and animations, etc [W3Sa].
In the extension code, it was used for the exact purpose referred to above, making some DOM manipulations and event methods more easily implemented.

3.7.6 Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment that uses **asynchronous programming** [Nod].
A Node.js app runs in a single process, without creating a new thread for every request. It provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm [Nod].
When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back [Nod].
Node.js has a unique advantage because millions of front-end developers that write JavaScript for the browser are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language [Nod].
NPM is a package manager for Node.js packages (or modules), that is automatically installed with Node.js. With this package manager, it is possible to install several important modules through the command line simply with the "`npm install x`" line, being the "x" the name of the module. You can also see all the modules installed with npm writing "`npm list`" in the command line.
In the project context, Node.js was used as the extension server in port 3000, and with the npm module "`mysql2`" the connection to the DB was made. Other npm modules that were used are the `body-parser`, `bcrypt`, `nodemon`, etc.

3.7.7 MAMP

Standing for "Mac OS X, Apache, MySQL, and PHP.", MAMP is an application that allows us to create a local server in a few simple steps, so we can develop our Web applications locally before they are finished and working on a Web server. MAMP is available in two versions, one free and one paid [Iva]. The free version is very complete, and it was the one that was used to store the project DB. By default, it uses the port 8888 for Apache, so if we want to access the DB we simply go to `http://localhost:8888/phpMyAdmin`. In the Node.js server it was necessary to establish a `socketPath` pointing to MAMP so it could access the DB, with the following code line in the connection: `socketPath:"/Applications/MAMP/tmp/mysql/mysql.sock"`.
Besides the DB storage, MAMP is also being used as the localhost for the **platform** of the Journal side.

3.8 Conclusion

In this chapter were presented all of the project functional and non-functional requirements, as being, respectively, the needs underlying the project functionality and the ones fundamental to good functioning. Then, it was represented the use-cases related to the extension and the platform, as well as the activity diagram of the extension. Taking into

account all the requirements, use-cases, and the activity diagram of the system, it is possible to idealize the functioning of the software in an efficient, effective, and safe way.
After that, it was approached the DB and the respective ERD of the system.
The system's architecture was also graphically explained in three sides perspective: the Extension side, the Web Server side, and the Journal side.
For last, all the tools used throughout the development and implementation of the project were reviewed and defined. All of them were essential to fulfilling the proposed objectives for this Master's dissertation.
The next chapter will address how both the extension and the platform were implemented at the user and the server level.

Chapter 4

Implementation

4.1 Introduction

This chapter will explain in detail every step of the development of both the extension and platform. For that purpose, it will be also shown some useful code snippets of the project. After all the implementation process is explained, it will be analyzed the obtained results.

This chapter is organized within the following sections:

1. The first section – **Introduction** – introduces and makes the division of the chapter.
2. The second section – **Browser Extension: A tool for fact checking** – intends to analyze all the extension implementation process.
3. The third section – **Web Platform for Content Producers** – explains all the platform implementation process.
4. The fourth section – **Conclusion** – refers the main conclusions of the chapter.

4.2 Browser Extension: A tool for fact checking

In this section, it will be approached all the steps and main functions that were followed and created through the extension development, including the dependencies needed and all the adjacent files that exist in the scope of this technology.
The initial goal of this extension was for it to work in a specific regional Journal, however, it was extended for all websites in order to perform a deepest content analysis in the future.

4.2.1 Dependencies

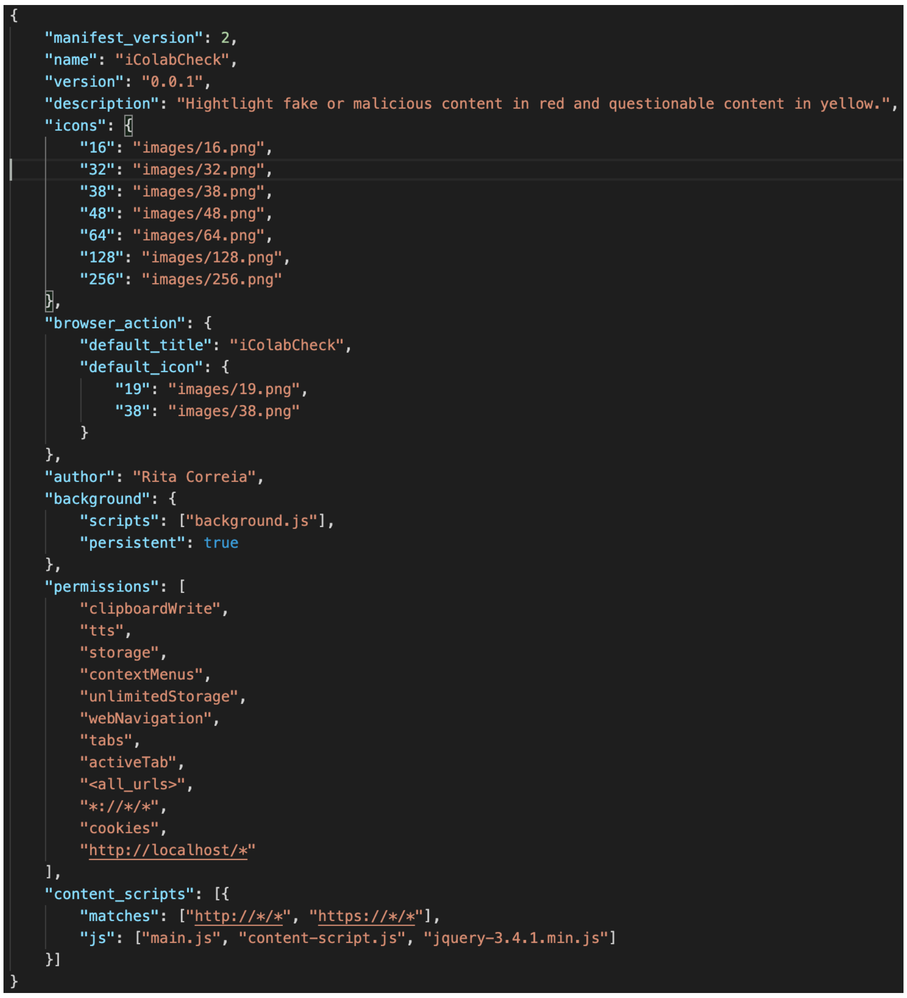
There are some dependencies and modules that were needed to install through the extension development. The first one was obviously **node.js**, as already explained in the previous chapter. The installation process of this software is very easy and clean. Also, it was immediately installed the **MAMP** software as it has the DB stored.
After these two main dependencies are set up and ready to use, I had to install some **node modules** with npm within the development process, being those:

- `express`: it's a fast, unopinionated, minimalist Web framework for node [npmb] that I only used at the beginning for node test purposes;

- `mysql2`: it's the MySQL client for Node.js with focus on performance [npmc]. Thus, it was needed for making the connection between node.js and the DB in the server;
- `body-parser`: it's the node.js body parsing middleware. It was used to parse incoming request bodies in the middleware before the handlers, and it's available under the `req.body` property [npma];
- `nodemon`: it's a tool that helps develop node.js-based applications by automatically restarting the node application when file changes in the directory are detected [npmd]. This was a very useful module, as I didn't need to worry about having to restart the server every time I did some file change;
- `bcrypt`: this module includes a library to help you hash and compare passwords in Node [Cor17]. This was very useful in the users' authentication process.

4.2.2 Manifest

As already discussed and analyzed in 2.2.3.1, the Manifest is the most important file in an extension, as it stores all the information, proprieties, rules, and permissions about the extension itself.



```
{
    "manifest_version": 2,
    "name": "iColabCheck",
    "version": "0.0.1",
    "description": "Hightlight fake or malicious content in red and questionable content in yellow.",
    "icons": {
        "16": "images/16.png",
        "32": "images/32.png",
        "38": "images/38.png",
        "48": "images/48.png",
        "64": "images/64.png",
        "128": "images/128.png",
        "256": "images/256.png"
    },
    "browser_action": {
        "default_title": "iColabCheck",
        "default_icon": {
            "19": "images/19.png",
            "38": "images/38.png"
        }
    },
    "author": "Rita Correia",
    "background": {
        "scripts": ["background.js"],
        "persistent": true
    },
    "permissions": [
        "clipboardWrite",
        "tts",
        "storage",
        "contextMenus",
        "unlimitedStorage",
        "webNavigation",
        "tabs",
        "activeTab",
        "<all_urls>",
        "*://*/*",
        "cookies",
        "http://localhost/*"
    ],
    "content_scripts": [{
        "matches": ["http://*/*", "https://*/*"],
        "js": ["main.js", "content-script.js", "jquery-3.4.1.min.js"]
    }]
}
```

Figure 4.1: Extension Manifest file.

In the figure 4.1, we can see the print-screen of the extension's Manifest file. As shown, it is first described the manifest version and the extension name on Chrome, in this case, "iColabCheck". After that is the extension version and description for the user to see in the extension installation process. I also defined some icons sizes to be shown depending on the device that the extension is operating. After defining the default title and icon, we have the author's name and the background script file that it should point to with the persistent value as "true". Then, we can see all the permission that the extension has, as well as the scripts that the background file should have direct access to as the content script.

4.2.3 Popup

The popup is the user interface of the extension. It should be minimalist, user-friendly, and efficient. In this case, we can distinguish three different popups: the **login** page, the **register** page and the **main** page. The first page that the user sees after installing the extension is the login page (figure 4.2), where he has to insert his email and password. If the user doesn't have an account, he should click on the option "Register" below on that page, which will redirect him to the register interface (figure 4.3), where he has to insert some parameters like his name, email, and password or return to the login page. If some registration field is left blank, it should become red and not proceed until the user fills it. Also, if the user inserts some already existing email, it should appear an error message and the user must register again. As we can see, there are two different password fields (password and confirm password). The user must insert the same password on both fields, or the registration process won't proceed, showing an error message.

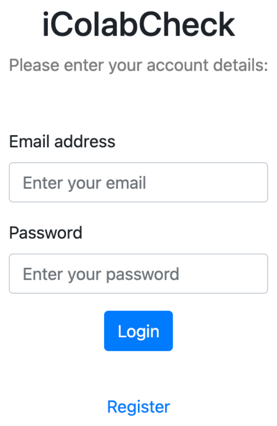


Figure 4.2: Extension login page.

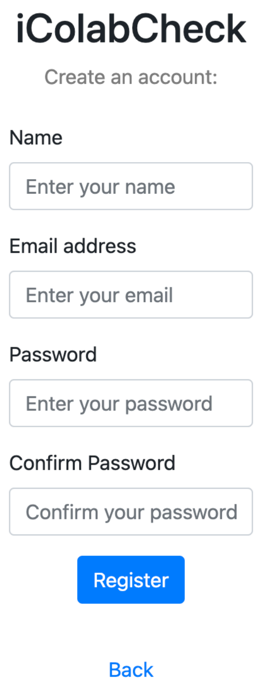
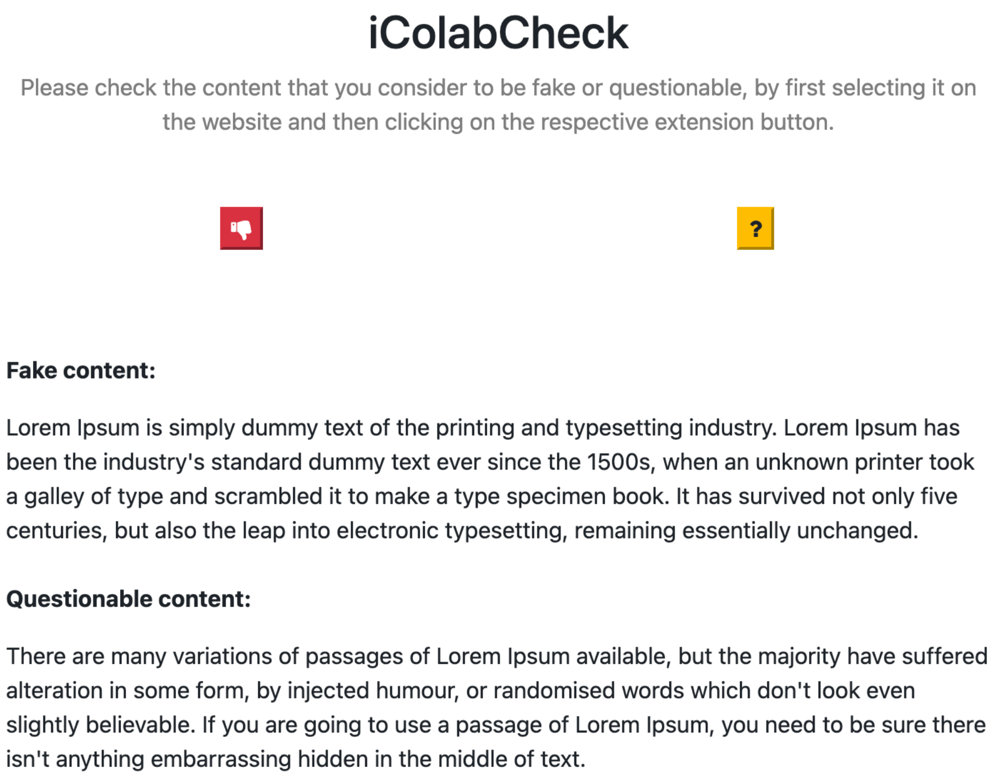


Figure 4.3: Extension register page.

After the user is logged in, it will appear the main extension page (figure 4.4), which has two buttons, one for the fake content and another one for the questionable one. For the correct usage of those buttons, the user must firstly select with the mouse or computer track-pad some content that he considers to be fake or questionable and then open the extension and click on the respective button. After this action, the selected content will become highlighted in red or yellow on the chosen website, depending on if the user selected it as fake or questionable, respectively. That selected content will also appear in the extension main popup, as we can see with an example in the figure 4.4.
When the content is selected by the user and one of the buttons is pressed, that content is stored on the DB along with the user name, email, and page URL, so it can be presented to the content producers on the platform, as we will see in the section 4.3.


iColabCheck

Please check the content that you consider to be fake or questionable, by first selecting it on the website and then clicking on the respective extension button.

?

Fake content:

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged.

Questionable content:

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text.

Figure 4.4: Extension main page.

4.2.4 Popup Script

The popup script interacts directly with the popup.html file. It has the functions of what to do when a button is clicked and what to output to the extension popup. In this case, beyond the popup.html, this file also interacts directly with the content script and the background script. The first one contains the highlight functions and the second one interacts mainly with the server to get or store data.

```javascript
document.getElementById('btn').addEventListener('click', function() {
    chrome.tabs.query({ active: true, currentWindow: true }, function(
        activeTabs) {
        chrome.tabs.sendMessage(activeTabs[0].id, { action: 'executeCode' })
            ;
    });
});

document.getElementById('btn2').addEventListener('click', function() {
    chrome.tabs.query({ active: true, currentWindow: true }, function(
        activeTabs) {
        chrome.tabs.sendMessage(activeTabs[0].id, { action: 'executeCode2'
            });
    });
});
```

Listing 4.1: Request from popup script to content script.

The code snippet 4.1 represents the actions of clicking in the fake (btn) and questionable (btn2) buttons. As we can see, if the user clicks on the first one, it will be sent a query to the content script with the request action ”executeCode”, otherwise the request action message is ”executeCode2”.
Also, in the popup script, we have the popup output functions, which will allow the user to see the fake and questionable content that he previously selected on the extension main page.

```javascript
function onSelection(selection1) {
    document.getElementById("output").innerHTML = selection1; /* outputs the
         fake text selections */
}

function onSelection2(selection2) {
    document.getElementById("output2").innerHTML = selection2; /* outputs
        the questionable text selections */
}

var gettingPage = chrome.extension.getBackgroundPage();
const selection1 = gettingPage.Fake();
onSelection(selection1);

const selection2 = gettingPage.Questionable();
onSelection2(selection2);
```

Listing 4.2: Popup output functions.

As we can see in the code snippet 4.2, the fake and questionable contents are being retrieved from the background script as two different arrays. Then, there are two functions (onSelection and onSelection2) to output both kinds of contents to the extension main page.

4.2.5 Content Script

The content script interacts directly with the Web page and it contains the browser highlight functions that result from clicking the popup buttons. Beyond those functions, it also contains a function that counts the number of the same words in the console, for future purposes, as we can see in the code snippet 4.3.

```
function count(str) {
    var obj = {};
    str.split(/[ ,.]+/).forEach(function(el) {
        obj[el] = obj[el] ? ++obj[el] : 1;
    });
    return obj;
}
```

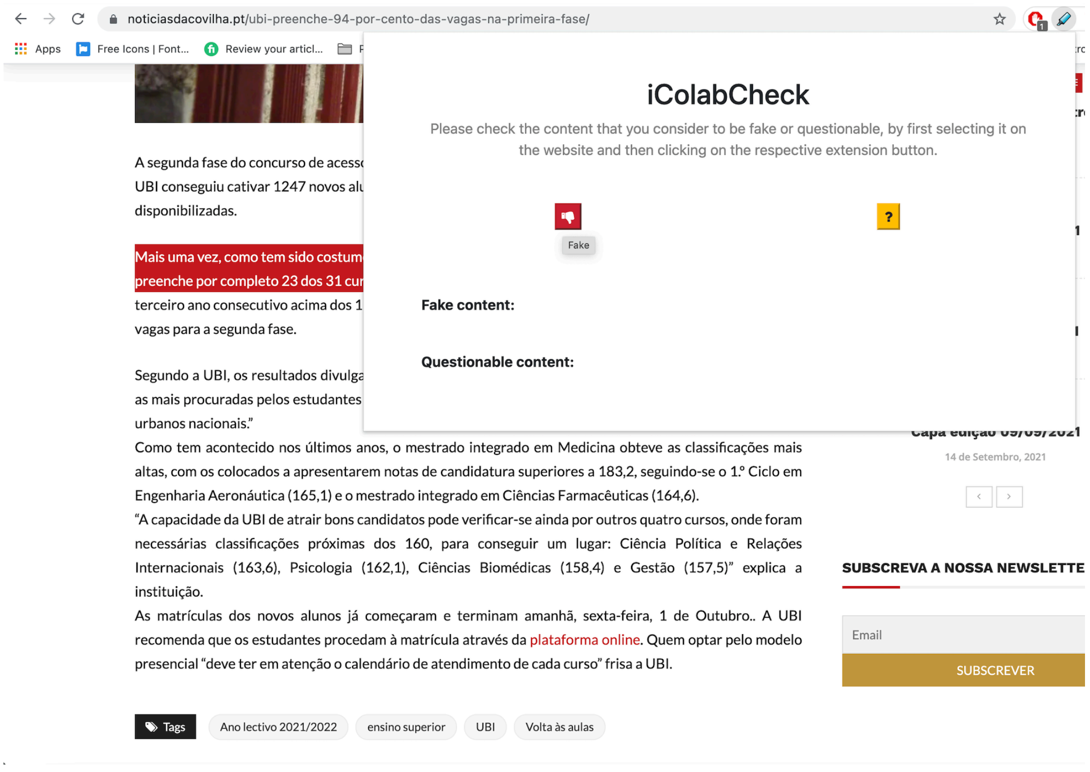
Listing 4.3: Word count function.

```
chrome.runtime.onMessage.addListener(function(request) {
    if (request.action == 'executeCode') { //fake button clicked
        var sel = window.getSelection().toString();
        if(sel.length)
            chrome.extension.sendRequest({'message':'setTextFake','data':
                sel},function(response){console.log(response.resp)})
        arrFake.push(sel);
        console.log(count(arrFake.toString()));
        highlightSelection();
        return;
    }
    else if (request.action == 'executeCode2') { //questionable button
        clicked
        var sel = window.getSelection().toString();
        if(sel.length)
            chrome.extension.sendRequest({'message':'setTextQuestionable','
                data': sel},function(response){console.log(response.resp)})
        arrQuestionable.push(sel);
        console.log(count(arrQuestionable.toString()));
        highlightSelection2();
        return;
    }
});
```

Listing 4.4: Buttons listener.

As we can see in the code snippet 4.4, it contains a listener for when the user presses the fake or questionable buttons. If the request action from the popup script is the "executeCode" one, it means that the user clicked on the fake content button, similarly to what is happening on the representative example on figure 4.5. As result, the content script will get that text selection from the `window.getSelection()` JavaScript function as a String. Then, if the selection is valid, the extension will send a request to the background script with the message "setTextFake" and the respective text selection and wait for a response.

After that, that text selection is stored in an array, the word count function is executed and finally, the selected text from the chosen Web page is highlighted in red through the `highlightSelection()` function, which we will explore later.



noticiasdacovilha.pt/ubi-preenche-94-por-cento-das-vagas-na-primeira-fase/

Apps · Free Icons | Font... · Review your articl...

iColabCheck

Please check the content that you consider to be fake or questionable, by first selecting it on the website and then clicking on the respective extension button.

Fake content:

Questionable content:

A segunda fase do concurso de acess
UBI conseguiu cativar 1247 novos alu
disponibilizadas.

Mais uma vez, como tem sido costum
preenche por completo 23 dos 31 cur
terceiro ano consecutivo acima dos 1
vagas para a segunda fase.

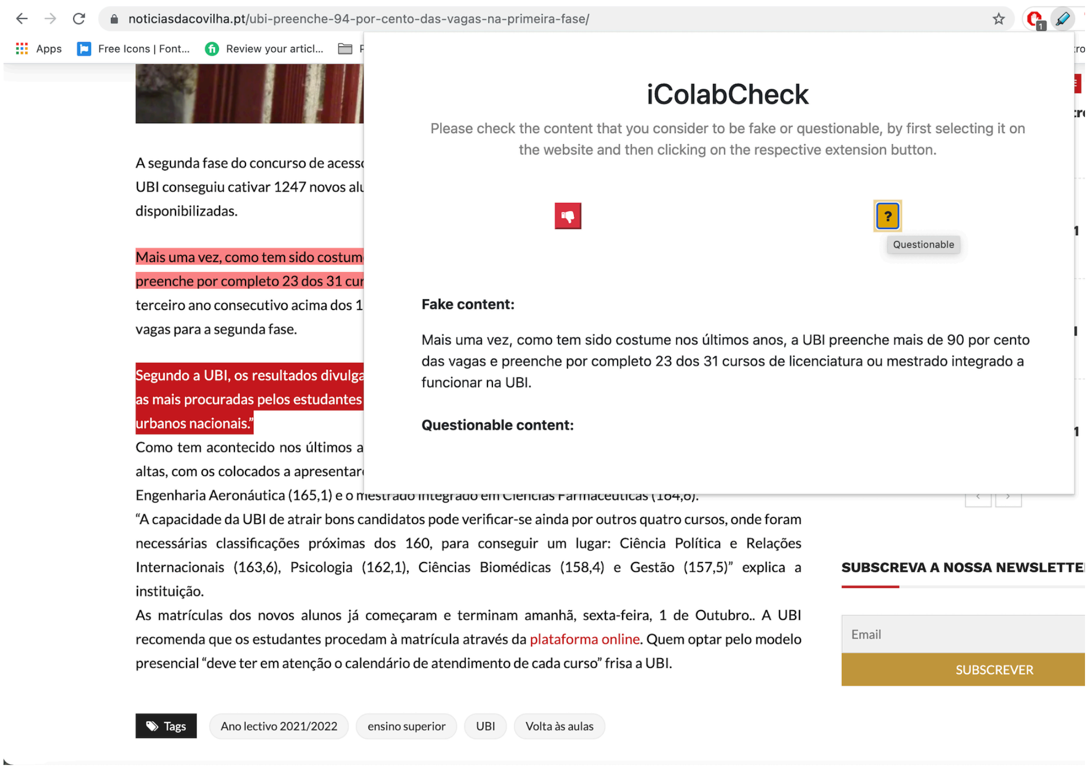
Segundo a UBI, os resultados divulga
as mais procuradas pelos estudantes
urbanos nacionais."
Como tem acontecido nos últimos anos, o mestrado integrado em Medicina obteve as classificações mais altas, com os colocados a apresentarem notas de candidatura superiores a 183,2, seguindo-se o 1.º Ciclo em Engenharia Aeronáutica (165,1) e o mestrado integrado em Ciências Farmacêuticas (164,6).
"A capacidade da UBI de atrair bons candidatos pode verificar-se ainda por outros quatro cursos, onde foram necessárias classificações próximas dos 160, para conseguir um lugar: Ciência Política e Relações Internacionais (163,6), Psicologia (162,1), Ciências Biomédicas (158,4) e Gestão (157,5)" explica a instituição.
As matrículas dos novos alunos já começaram e terminam amanhã, sexta-feira, 1 de Outubro.. A UBI recomenda que os estudantes procedam à matrícula através da plataforma online. Quem optar pelo modelo presencial "deve ter em atenção o calendário de atendimento de cada curso" frisa a UBI.

Tags · Ano lectivo 2021/2022 · ensino superior · UBI · Volta às aulas

Capa edição 09/09/2021
14 de Setembro, 2021

SUBSCREVA A NOSSA NEWSLETTE
Email
SUBSCREVER

Figure 4.5: "Fake" button clicked on selection example.



noticiasdacovilha.pt/ubi-preenche-94-por-cento-das-vagas-na-primeira-fase/

Apps · Free Icons | Font... · Review your articl...

iColabCheck

Please check the content that you consider to be fake or questionable, by first selecting it on the website and then clicking on the respective extension button.

Fake content:

Mais uma vez, como tem sido costume nos últimos anos, a UBI preenche mais de 90 por cento das vagas e preenche por completo 23 dos 31 cursos de licenciatura ou mestrado integrado a funcionar na UBI.

Questionable content:

A segunda fase do concurso de acess
UBI conseguiu cativar 1247 novos alu
disponibilizadas.

Mais uma vez, como tem sido costum
preenche por completo 23 dos 31 cur
terceiro ano consecutivo acima dos 1
vagas para a segunda fase.

Segundo a UBI, os resultados divulga
as mais procuradas pelos estudantes
urbanos nacionais."
Como tem acontecido nos últimos a
altas, com os colocados a apresentar
Engenharia Aeronáutica (165,1) e o mestrado integrado em Ciências Farmacêuticas (164,6).
"A capacidade da UBI de atrair bons candidatos pode verificar-se ainda por outros quatro cursos, onde foram necessárias classificações próximas dos 160, para conseguir um lugar: Ciência Política e Relações Internacionais (163,6), Psicologia (162,1), Ciências Biomédicas (158,4) e Gestão (157,5)" explica a instituição.
As matrículas dos novos alunos já começaram e terminam amanhã, sexta-feira, 1 de Outubro.. A UBI recomenda que os estudantes procedam à matrícula através da plataforma online. Quem optar pelo modelo presencial "deve ter em atenção o calendário de atendimento de cada curso" frisa a UBI.

Tags · Ano lectivo 2021/2022 · ensino superior · UBI · Volta às aulas

SUBSCREVA A NOSSA NEWSLETTE
Email
SUBSCREVER

Figure 4.6: "Questionable" button clicked on selection example.

On the other hand, if the request action from the popup script is ”executeCode2”, it means that the user clicked on the ”Questionable” button, likewise at what is happening on the representative example on figure 4.6, and the process occurs in a similar way for the questionable content and the selected text is highlighted in yellow.

```
// it gets the fake content text and calls the function to highlight the
    selection in red
function highlightSelection() {
    var safeRanges = getSafeRanges(window.getSelection().getRangeAt(0));
    for (var i = 0; i < safeRanges.length; i++) {
        highlightRange(safeRanges[i]);
    }
}
// it gets the questionable content text and calls the function to highlight
     the selection in yellow
function highlightSelection2() {
    var safeRanges = getSafeRanges(window.getSelection().getRangeAt(0));
    for (var i = 0; i < safeRanges.length; i++) {
        highlightRange2(safeRanges[i]);
    }
}
```

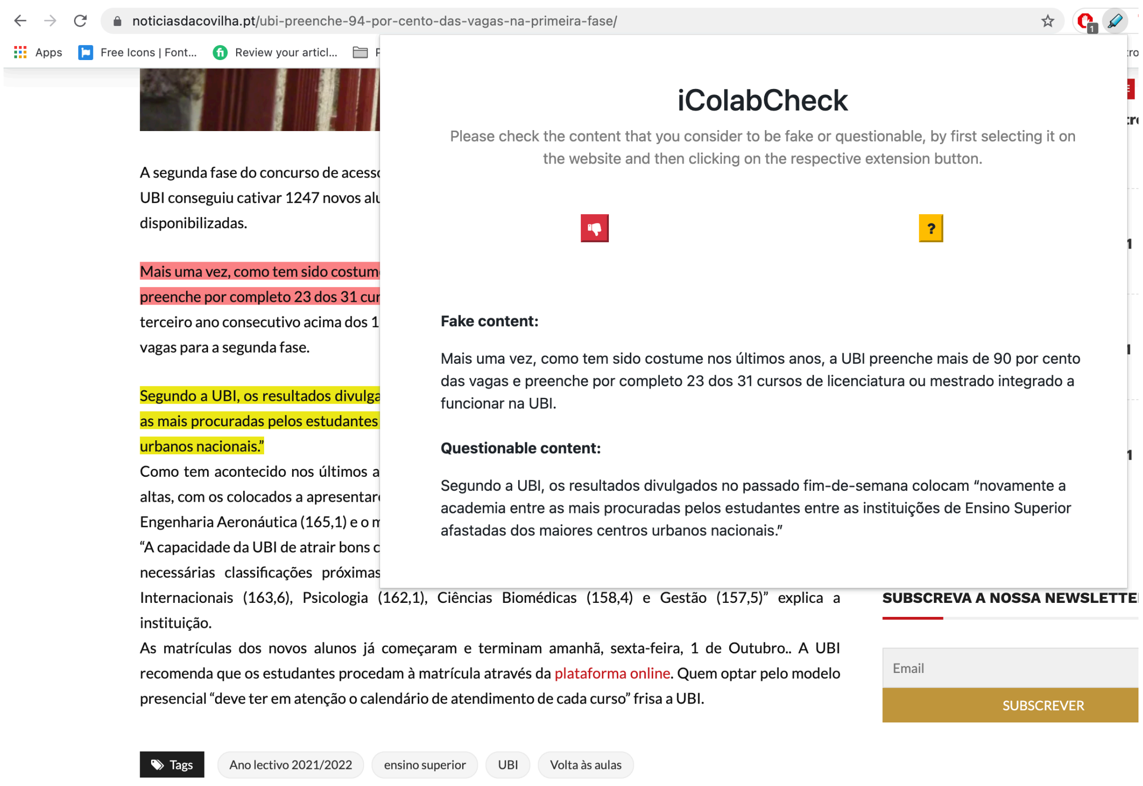
Listing 4.5: Functions that get the safe ranges and make the call to highlight the selections.

In the code snippet 4.5, are the functions that check the safe ranges of the selected content and call the one to highlight the text in red or yellow. The `getSafeRanges(t)` function that is called in the beginning works to get through all HTML tags in the body of the selected text. Then, for all the safe ranges, the content will be highlighted in the browser with the functions `highlightRange(x)` for the fake content and `highlightRange2(x)` for the questionable one.

```
//function that highlights in red the selection in the Web page
function highlightRange(range) {
    var newNode = document.createElement("div");
    newNode.setAttribute(
        "style",
        "background-color: #ff8585; display: inline;");
    range.surroundContents(newNode);
}
//function that highlights in yellow the selection in the Web page
function highlightRange2(range) {
    var newNode = document.createElement("div");
    newNode.setAttribute(
        "style",
        "background-color: #ebeb1e; display: inline;");
    range.surroundContents(newNode);

}
```

Listing 4.6: Highlight functions.

In the code snippet 4.6, are the functions that highlight the selected content in the Web page in red if the user marks it as fake and in yellow if the user sees it as questionable, as we can see in figure 4.8. For this purpose, it was instantiated a variable `newNode` that creates a `div` element in the page body which will have the appropriate background color depending on the content. Then, the `Range.surroundContents()` method is called, which is a method that moves the content of the range into a new node, placing the new node at the start of the specified range.



noticiasdacovilha.pt/ubi-preenche-94-por-cento-das-vagas-na-primeira-fase/
Apps
Free Icons | Font...
Review your articl...

iColabCheck

Please check the content that you consider to be fake or questionable, by first selecting it on the website and then clicking on the respective extension button.

Fake content:

Mais uma vez, como tem sido costume nos últimos anos, a UBI preenche mais de 90 por cento das vagas e preenche por completo 23 dos 31 cursos de licenciatura ou mestrado integrado a funcionar na UBI.

Questionable content:

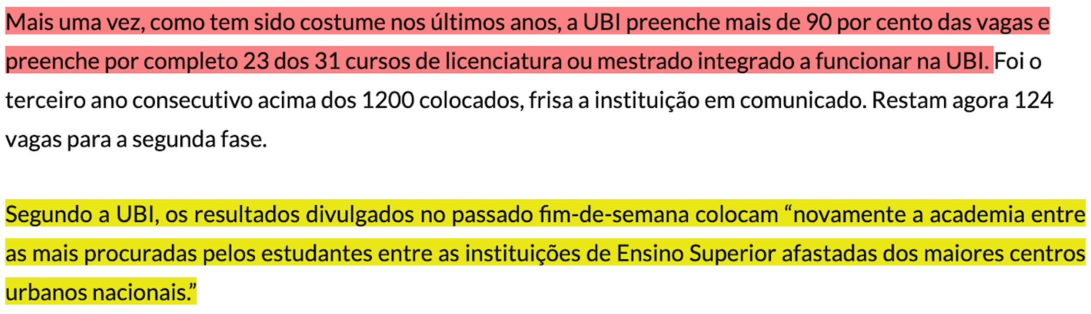
Segundo a UBI, os resultados divulgados no passado fim-de-semana colocam “novamente a academia entre as mais procuradas pelos estudantes entre as instituições de Ensino Superior afastadas dos maiores centros urbanos nacionais.”

A segunda fase do concurso de acess
UBI conseguiu cativar 1247 novos alu
disponibilizadas.

Mais uma vez, como tem sido costum
preenche por completo 23 dos 31 cur
terceiro ano consecutivo acima dos 1
vagas para a segunda fase.

Segundo a UBI, os resultados divulga
as mais procuradas pelos estudantes
urbanos nacionais.”
Como tem acontecido nos últimos a
altas, com os colocados a apresentar
Engenharia Aeronáutica (165,1) e o m
“A capacidade da UBI de atrair bons c
necessárias classificações próximas
Internacionais (163,6), Psicologia (162,1), Ciências Biomédicas (158,4) e Gestão (157,5)” explica a instituição.
As matrículas dos novos alunos já começaram e terminam amanhã, sexta-feira, 1 de Outubro.. A UBI recomenda que os estudantes procedam à matrícula através da plataforma online. Quem optar pelo modelo presencial “deve ter em atenção o calendário de atendimento de cada curso” frisa a UBI.

Figure 4.7: Selection results on the extension.



Mais uma vez, como tem sido costume nos últimos anos, a UBI preenche mais de 90 por cento das vagas e preenche por completo 23 dos 31 cursos de licenciatura ou mestrado integrado a funcionar na UBI. Foi o terceiro ano consecutivo acima dos 1200 colocados, frisa a instituição em comunicado. Restam agora 124 vagas para a segunda fase.

Segundo a UBI, os resultados divulgados no passado fim-de-semana colocam “novamente a academia entre as mais procuradas pelos estudantes entre as instituições de Ensino Superior afastadas dos maiores centros urbanos nacionais.”

Figure 4.8: Selection results on the browser.

In the figure 4.7, are the results on the extension from clicking the fake and questionable buttons on an **example selection**. As we can see, both of the selected text excerpts are printed on the respective box area on the extension. At the same time, as it is emphasized in figure 4.8, the selections became highlighted in the browser page in red and yellow, being these, content that the user considers to be fake and questionable, respectively.

4.2.6 Background Script

The background script is basically something that runs in the background and listens for triggers while the user interacts with the Chrome browser. As so, it has a listener so it can make actions according to the message received.

```
chrome.extension.onRequest.addListener(function(request, sender,
    sendResponse) {
    switch(request.message) {
        case 'login':
            creds = request.payload.email;
            flip_user_status(true, request.payload)
                .then(res => sendResponse(res)) //send the final response
                    and close the msg line.
                .catch(err => console.log(err));
            return true; //keeps the message line open between the popup-
                sign-in script and the background script.

        case 'logout':
            flip_user_status(true, null)
                .then(res => sendResponse(res))
                .catch(err => console.log(err));
            return true;

        case 'register':
            register_user(request.payload)
                .then(res => sendResponse(res)) //send the final response
                    and close the msg line.
                .catch(err => console.log(err));
            break;
```

Listing 4.7: Background script listener - part one.

As we can see in the code snippet 4.7, we have a `switch-case` statement inside the listener, which will define which condition will be executed according to the request message received. If that message says ”login”, the first condition will proceed, firstly by getting to a variable the payload data, which is the data that was passed to the background script (in this case the user login info such as the email and password), and then by running the `flip_user_status (x,y)` function that we will analyze later. After the final response is sent, we will return true at the end of that condition in order to keep the message line open between the popup-sign-in script, which is the script that sent the user credentials, and the background script. This is an asynchronous process. The ”logout” condition will work similarly, although it is not implemented in this version of the project.
If the message received by the background script is ”register”, the user registration function, which we will analyze later, will be called with the user information as parameters.

```
        case 'setTextFake':
            chrome.tabs.query({active: true, lastFocusedWindow: true}, tabs
                => {
                let url = tabs[0].url;
                window.seltext = request.data;
                arrFake.push(request.data);
                savefake(request.data, creds, url);
                sendResponse({resp: "Fake content sent to bg script!"});
            });
            break;

        case 'setTextQuestionable':
            chrome.tabs.query({active: true, lastFocusedWindow: true}, tabs
                => {
                let url = tabs[0].url;
                window.seltext = request.data;
                arrQuestionable.push(request.data);
                savequestionable(request.data, creds, url);
                sendResponse({resp: "Questionable content sent to bg script!
                    "});
            });
            break;

        default:
            sendResponse({data: 'Invalid arguments'});
        break;
    }
});
```

Listing 4.8: Background script listener - part two.

As we can see in the code snippet 4.8, still in the background script listener, if the message received is ”setTextFake” the action of storing the fake content selected by the user in the DB will proceed by calling the function `savefake(x,y,z)` which has as parameters the selected text, the user email, and the Web page URL. The process for the questionable content is done similarly through the clause ”setTextQuestionable” which is associated with the function `savequestionable(x,y,z)`.

```
function flip_user_status(signIn, user_info){
    if(signIn){
        return fetch ('http://localhost:3000/login', {
            method: 'GET',
            headers: {
                'Authorization': 'Basic ' + btoa(`${user_info.email}:${
                    user_info.pass}`)
            }})
            .then(res => {
                return new Promise(resolve => {
                    if (res.status !== 200){
                        alert ("Email or password incorrect!");
                        resolve('fail');
                    }
                    chrome.storage.local.set({userStatus: signIn, user_info
                        }, function(response){
                        if (chrome.runtime.lastError) resolve ('fail');
                        user_signed_in = signIn;
                        resolve('success');
                    });
                })
            })
            .catch(err => console.log(err));
    }
```

Listing 4.9: Flip user status function - part one.

The function presented in the code snippet 4.9 flips the user status and returns a Promise. If the `signIn` variable is true, the user is signed in. We firstly return a fetch call to the server API, hitting the localhost:3000/login route with the method GET, and we send the email and the password through the authorization header, parsing that information with the `btoa` method which creates a Base64-encoded ASCII string from a binary string (i.e., a String object in which each character in the string is treated as a byte of binary data). If the response is not 200, we will resolve a fail. If not, we know we've successfully passed the backend test, and we saved that status in the local storage of Chrome. After the user status is saved, which is true or false (in this case true), we save the email and the encrypted password and got a response. If there is an error, it will resolve a fail. If not, the user has successfully signed in, so we change the variable to true, and we resolve the "success". That's what the case "login" response will expect.

```
else if(!signIn){
        return new Promise (resolve => {
            chrome.storage.local.get(['userStatus', user_info], function(
                response){
                if (browser.runtime.lastError) resolve ('fail');
                if (response.userStatus === undefined) resolve ('fail');
                fetch ('http://localhost:3000/logout', {
                method: 'GET',
                headers: {
                    'Authorization': 'Basic ' + btoa(`${response.user_info.
                        email}:${response.user_info.pass}`)
                }})
                .then(res => {
                    if (res.status !== 200){
                        resolve('fail');
                    }
                    chrome.storage.local.set({userStatus: signIn, user_info:
                            { }}, function(response){
                                if (chrome.runtime.lastError) resolve ('fail');
                                user_signed_in = signIn;
                                resolve('success');
                            });
                        })
                        .catch(err => console.log(err));
                         });
        });
    }
}
```

Listing 4.10: Flip user status function - part two.

On the other hand, being the code snippet 4.10 still in the ”flip_user_status” function, if the `signIn` variable is false, it means that the user wants to sign out. Again, we return a Promise because this function is supposed to return a Promise.
As the `chrome.storage.local.get` is not a Promise, we need to wrap it. So we do a new Promise, get the resolve object and open up the chrome local storage to see if the user has the proper credentials by getting the user status and info. If there is an error in getting that data, we send back ”fail”. If the status is undefined (it’s neither true nor false) we also send back a ”fail”. Otherwise, we know that’s a true or false and fetch to the logout route through the GET method and send in the header the authorization with the user email and password with the `btoa` method. The back-end verifies if that user is actually on the system. If he is, it sends a 200 code. If not, we resolve a ’fail’. If the received code is 200, we know the user has proper credentials, so we open the `storage.local.set`, flip the `userStatus` variable to false and clear his information. If there is an error doing that, we resolve a ’fail’. If not, we know the whole process was made correctly so the `user_signed_in` variable will be false and we resolve with a success.

```
function register_user(user_info){
    return fetch ('http://localhost:3000/register', {
            method: 'GET',
            headers: {
                'Authorization': 'Basic ' + btoa(`${user_info.name}:${
                    user_info.email}:${user_info.pass}:${user_info.
                    confirmpass}`)
            }})
            .then(res => {
                return new Promise(resolve => {
                    if (res.status === 400){
                        alert ("Email already in use!");
                        resolve('fail');
                    }
                    if (res.status === 401){
                        alert ("Passwords do not match!");
                        resolve('fail');
                    }
                    else if (res.status !== 200 ){
                        resolve('fail');
                    }
                    else {resolve('success');}
                })
            })
            .catch(err => console.log(err));
}
```

Listing 4.11: User registration function.

The code snippet 4.11 represents the user registration function in the background script. This function works similarly to the one presented before. We start with a fetch, which is a Promise, and hit the localhost:3000/register route through the GET method passing in the authorization header the user name, email, password, and password confirmation with the `btoa` method. Then, we make a new Promise, get the resolve object and if the status response is 400, it means that the inserted email is already in use and resolve a ”fail”, giving an alert to the user. If the status is 401, it means that both passwords don’t match and resolve a ”fail” also giving an alert to the user. If the status is anything but 200 it means that another error occurred and we also resolve a ”fail”. Otherwise, we resolve a ”success”.

```
function savefake(text, creds, url) {
    var jax = new XMLHttpRequest();
    var params = "text="+text+"&creds="+creds+"&url="+url;
    jax.open("POST","http://localhost:3000/InsertFake", true);
    jax.setRequestHeader("Content-Type","application/x-www-form-urlencoded")
        ;
    jax.send(params);
    jax.onreadystatechange = function() { if(jax.readyState==4) { alert(jax.
        responseText);  }}
}

function savequestionable(text, creds, url) {
    var jax = new XMLHttpRequest();
    var params = "text="+text+"&creds="+creds+"&url="+url;
    jax.open("POST","http://localhost:3000/InsertQuestionable", true);
    jax.setRequestHeader("Content-Type","application/x-www-form-urlencoded")
        ;
    jax.send(params);
    jax.onreadystatechange = function() { if(jax.readyState==4) { alert(jax.
        responseText);  }}
}
```

Listing 4.12: Save content functions.

The code snippet 4.12 has the functions that called the specific server route to save the fake or questionable content that the user selected on the DB. As we can see, in the `savefake(x,y,z)` function, we used the `XMLHttpRequest` which is an object that provides functionality to the client to transfer data between a client and a server, without having to do a full-page refresh. In order to do that, we firstly created a new instance of that object. Then, in the variable `params`, we stored the necessary parameters such as the content itself, the user email, and the URL of the Web page in a String mode. After that, we open a request with an asynchronous call through the POST method to the localhost:3000/InsertFake route, add the header and send that request with the defined parameters. The function `savequestionable(x,y,z)` works in a similar way for the content that the user finds questionable.

4.2.7 Server

The server interacts directly with the DB and with the background script. In this file, we firstly define the dependencies and make the mysql connection to the DB through the npm `mysql2` module. After that, we began to implement the routing process. Routing refers to how an application's endpoints (URIs) respond to client requests. You define routing using methods of the Express app object that correspond to HTTP methods; for example, app.get() to handle GET requests and app.post() to handle POST requests. These routing methods specify a callback function called when the application receives a request to the specified route (endpoint) and HTTP method [Exp].

```
app.post('/InsertFake', (req, res) => {
    var text = req.body.text;
    var email = req.body.creds;
    var user_id = 0;
    var url = req.body.url;

   // handling apostrophe in sql query
  if (text.includes("'")){
    var s_aux = "";
    for (let i = 0; i<text.length; i++){
        if (text[i] == "'"){
            s_aux += text[i];
        }
        s_aux += text[i];
    }
    text = s_aux;
  }

  function get_info(callback){
    var sql = "SELECT `id` FROM `User` WHERE `email` = '"+email+"'";
    connection.query(sql, function(err, result){
        if(err) throw err;
        user_id = result[0];
        return callback(result[0]);
    })
  }
  get_info(function(result){
    user_id = result.id;
    var sql = "INSERT INTO `Fake` (`text`, `user_id`, `url`) VALUES ('"+text
       +"', '"+user_id+"', '"+url+"')";
    connection.query(sql, (err,result)=>{
      if(err) throw err;
      res.send(result);
    });
  });
});
```

Listing 4.13: Store fake content route.

In the code snippet 4.13, we can see the route when it comes to adding content that the user considers being **fake** to the respective DB table. For this purpose, we need to get the text itself, the user email and ID, and the page URL. All of these parameters, except for the user id, are sent to the server through the background script, as we previously saw. Then, we needed to create an `If` statement in order to handle the apostrophe in sql queries. Without this statement, every time the user selected some text that contained an apostrophe, the query would stop there. After that, it was implemented a function (`get_info(c)`) that makes a query to the DB so we can get the ID of the user that selected the content, returning that value. When that function is called we can finally execute the query that will add the ”fake” content to the DB as we already have all the necessary parameters.
The process of adding the content that the user considers being **questionable** to the respective DB table is similar.

```
function authenticate_user(req, res, next) {
  var user_pass = 0;
  let creds = req.get('Authorization');
  creds = creds.substr(creds.indexOf(' ') + 1); // will start at the begin
      of the credentials
  creds = Buffer.from(creds, 'base64').toString('binary'); //convert it back
       to binary string
  creds = creds.split(':'); //give us an array with the email and the pass
  var email = creds[0];
  var pass = creds[1];

  /* Here is the DB check of credentials */

  function get_auth(callback){
  var sql = "SELECT `password` from User WHERE email = '"+email+"' ";
  connection.query(sql, function(err, result){
    if (err) throw err;
    user_pass = result[0];
    return callback(result[0]);
  })
  }

  get_auth(function(result){
    try{
      user_pass = result.password;
      bcrypt.compare(pass, user_pass, (err, result) => {
        if (err || !result) {
          res.status(401).end() //user is unauthorized
          return false;
        }
        else {
          res.status(200).end(); //user is authorized
          next();
        }
      });
    }

    catch{
      res.status(404).end() //user not existent
      return false;
    }
  });
}
```

Listing 4.14: Login user route function.

In the code snippet 4.14, we have the middleware function to do the authorization/check of the user credentials when he tries to authenticate through the login route.
The user credentials (email and password) are passed to the server from the background script through the headers basic authorization of the `fetch` API. When getting that request to a ”creds” variable, the content of that variable would be something like ”Basic encodedemailandpass”. As we don't need the ”Basic” word and the space after, we have to remove it, in this case, through the `substr()` method. Now that we have just the credentials their-selves, we need to convert them back from base64 to binary String. At this point, if we printed the creds variable, it would look something like ”email:password”. Here, we just need to split up the email and the password that are divided through the ”:” character, and attribute the values to the respective variables.
After this process, we have to do a DB check of credentials. For this purpose, I created a function `get_auth(c)` that performs a query to the DB that, as result, returns the encrypted password stored on the DB of the user that is trying to login, according to his email. When that function is called we just need to compare the password that the user typed in the login form with the one stored in the DB through the `bcrypt.compare()` method in a `try/catch` statement. If there are no correspondences, the user is unauthorized and can't login, otherwise, if the response status is 200, the user is authorized and able to login. If the statement gets to the catch clause, the user does not exist at all and also can't login into the system.

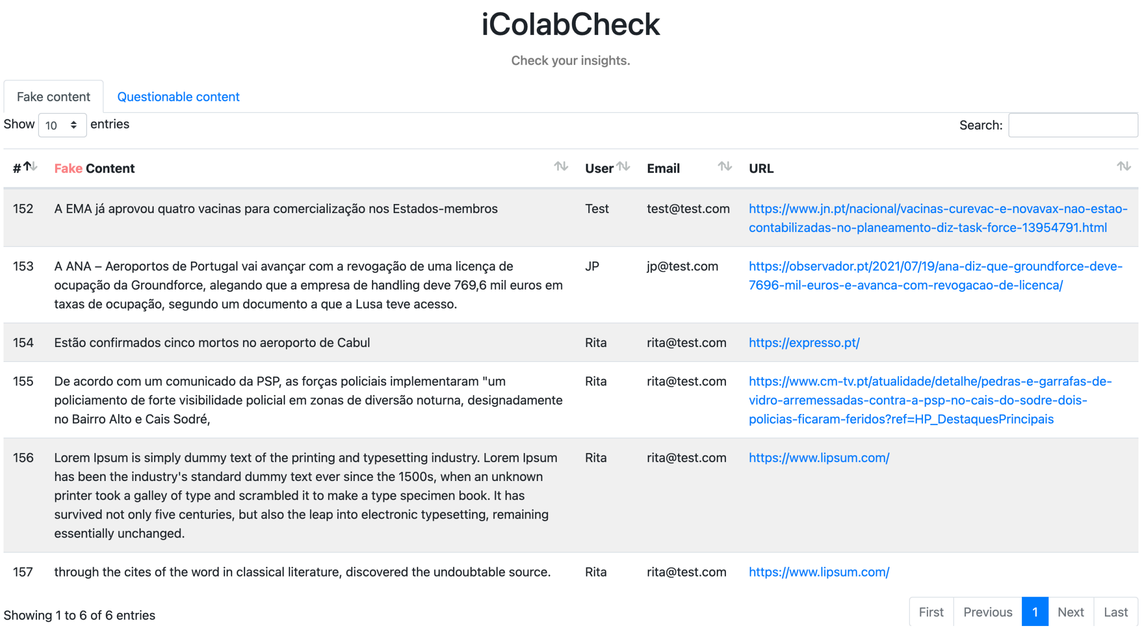
```javascript
app.get('/register', (req, res) => {

  let creds = req.get('Authorization');
  creds = creds.substr(creds.indexOf(' ') + 1);
  creds = Buffer.from(creds, 'base64').toString('binary');
  creds = creds.split(':');

  var name = creds[0]
  var email = creds[1];
  var pass = creds[2];
  var confirmpass = creds[3];

  connection.query('SELECT email FROM User WHERE email = ?', [email], async
      (error, results) => {
    if (error) {
      console.log(error);
      res.status(500).end();
    }

    if (results.length > 0){ //email already exists
      res.status(400).send("That email is already in use!").end();
    }

    else if (pass === confirmpass) { // passwords are the same
      let hashedPassword = await bcrypt.hash(pass, 8);

      connection.query('INSERT INTO `User` SET ?', {name: name, email: email
          , password:hashedPassword}, (error, results ) =>{
          if (error) throw err;
          else {
           res.status(200).send("User registered.").end();
          }
      });
    }

    else { // passwords are not the same
      res.status(401).end();
    }
  });
});
```

Listing 4.15: User registration route.

In the code snippet 4.15, we can see the route for registering a user on the extension. The first part of getting and decoding the credentials is similar to the login route previously analyzed. The only difference is that here we have more parameters (name, email, password and confirm password). So, after storing those credentials in the proper variables, we have a query that will search if the email given by the user already exists in the DB. If there are any results, it means that the email already exists and the registration process won't proceed with code 400. We also have a statement that confirms if both passwords written by the user in the registration form are the same. If they are, we will encrypt the password through the `bcrypt.hash()` method and make a query to the DB adding the new user to the respective table and end the process with code 200. Otherwise, the passwords provided don't match and the connection will end with code 401.

4.3 Web Platform for Content Producers

This section will approach the steps and the main components that constitute the platform for the content producers on the Journal side.
The server that is under the platform is MAMP localhost (see 3.7.7). It is running on Apache port 8888 and the PHP version is 7.4.12.



iColabCheck

Check your insights.

Fake content | Questionable content

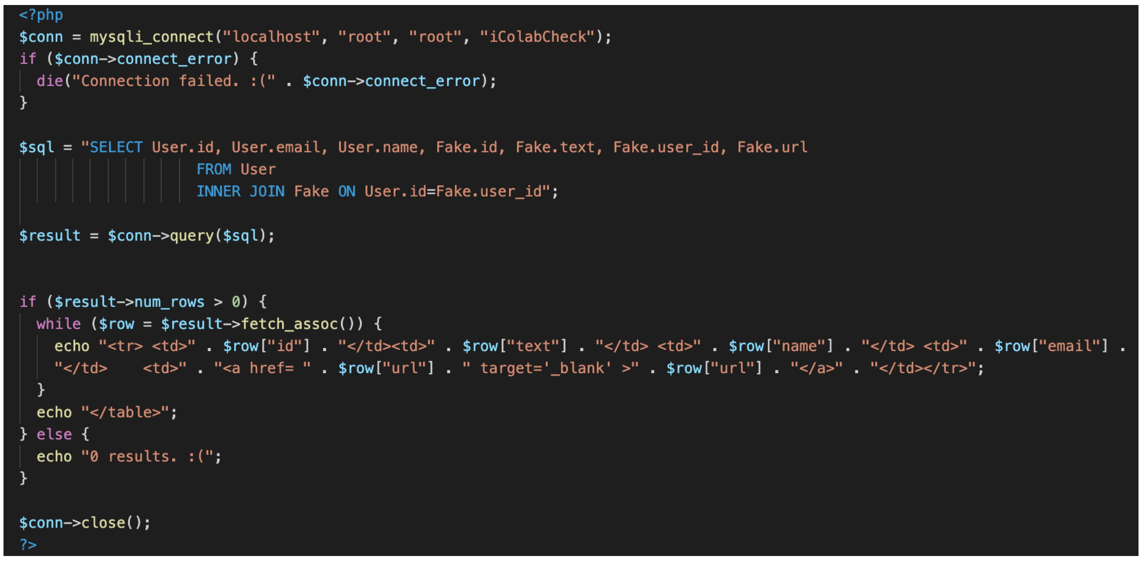
Show 10 entries | Search:

#	Fake Content	User	Email	URL
152	A EMA já aprovou quatro vacinas para comercialização nos Estados-membros	Test	test@test.com	https://www.jn.pt/nacional/vacinas-curevac-e-novavax-nao-estao-contabilizadas-no-planeamento-diz-task-force-13954791.html
153	A ANA – Aeroportos de Portugal vai avançar com a revogação de uma licença de ocupação da Groundforce, alegando que a empresa de handling deve 769,6 mil euros em taxas de ocupação, segundo um documento a que a Lusa teve acesso.	JP	jp@test.com	https://observador.pt/2021/07/19/ana-diz-que-groundforce-deve-7696-mil-euros-e-avanca-com-revogacao-de-licenca/
154	Estão confirmados cinco mortos no aeroporto de Cabul	Rita	rita@test.com	https://expresso.pt/
155	De acordo com um comunicado da PSP, as forças policiais implementaram "um policiamento de forte visibilidade policial em zonas de diversão noturna, designadamente no Bairro Alto e Cais Sodré,	Rita	rita@test.com	https://www.cm-tv.pt/atualidade/detalhe/pedras-e-garrafas-de-vidro-arremessadas-contra-a-psp-no-cais-do-sodre-dois-policias-ficaram-feridos?ref=HP_DestaquesPrincipais
156	Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged.	Rita	rita@test.com	https://www.lipsum.com/
157	through the cites of the word in classical literature, discovered the undoubtable source.	Rita	rita@test.com	https://www.lipsum.com/

Showing 1 to 6 of 6 entries

Figure 4.9: Platform page.

As we can see in figure 4.9, on the platform main page we have a nav-tab so we can distinguish the fake content stats from the questionable ones. Inside each tab, and with the help of Bootstrap (see 3.7.2), we created a table so the content producer can see the organized content and the respective information such as the user who highlight it, his email, and the page URL. It is also possible to arrange those parameters ascending or descending, show a certain number of entrances at once, search for keywords and see the first, previous, next, and last table pages. The table in the questionable content tab works similarly. The content in each tab is being retrieved from the same DB that interacts with the extension. This process is done by injecting some PHP on the HTML file, as we can see in figure 4.10.
In the figure 4.10, after the SQL connection to the DB is correctly made through the `mysqli_connect()` method, a query is performed so we can get all the needed informa-



```php
<?php
$conn = mysqli_connect("localhost", "root", "root", "iColabCheck");
if ($conn->connect_error) {
  die("Connection failed. :(" . $conn->connect_error);
}

$sql = "SELECT User.id, User.email, User.name, Fake.id, Fake.text, Fake.user_id, Fake.url
                    FROM User
                    INNER JOIN Fake ON User.id=Fake.user_id";

$result = $conn->query($sql);


if ($result->num_rows > 0) {
  while ($row = $result->fetch_assoc()) {
    echo "<tr> <td>" . $row["id"] . "</td><td>" . $row["text"] . "</td> <td>" . $row["name"] . "</td> <td>" . $row["email"] .
    "</td>    <td>" . "<a href= " . $row["url"] . " target='_blank' >" . $row["url"] . "</a>" . "</td></tr>";
  }
  echo "</table>";
} else {
  echo "0 results. :(";
}

$conn->close();
?>
```

Figure 4.10: PHP injected in the HTML of the platform page.

tion to fill the tables of both tabs (in this case, the ”Fake Content” tab). After the query is executed, a `while` loop is done so it fills the table line by line until there are no more results. The code injected for the questionable content is similar.

4.4 Conclusion

In this chapter, was approached and explained all the implementation details and the main code functions of both the Chrome extension and the platform development process. As it became clear, it was a very long process that suffered a lot of changes in the meantime, but in the end, I was able to learn a lot more about JavaScript, its complex syntax, and specificities.
In the next chapter, it will be detailed some manual tests performed on both the extension and platform, as well as their visual results to the user.

Chapter 5

Tests

5.1 Introduction

During the development of the project, several manual tests were carried out on the extension, in order to ensure that the user experience is good and that he receives the respective notifications or alerts when a problem occurs. In the case of the platform, as it is still in the first stage of development, the tests were simple and carried out with Chrome Dev Tools, which performed a global overview of the page within some different aspects.

This chapter is organized into the following sections:

1. The first section – **Introduction** – introduces and makes the division of the chapter.
2. The second section – **Extension Tests** – intends to analyze the manual tests performed on the extension.
3. The third section – **Platform Tests** – demonstrates the tests performed on the platform with Chrome Dev Tools.
4. The fourth section – **Conclusion** – refers the main conclusions of the chapter.

5.2 Extension Tests

The manual tests for the extension were carried out throughout the development process, with several constant improvements and better implementation solutions having been created for some functionalities. During the development of the extension, several error messages and alerts were implemented for the user, in case something goes wrong.

5.2.1 Error Messages on Login Page

The figure 5.1 represents the login form when the user doesn't fill the email or password fields (in this case, both). Thus, the respective box turns red and with a message to the user to enter the respective missing field.
When the email and/or the password are incorrect and not registered in the DB, an alert appears on the page with that information to the user, as we can see in the figure 5.2.
The figure 5.3 shows the error message that appears to the user when the email is badly formatted and missing an ”@”. Also, when the email is incomplete and there is nothing after the ”@” character, another message shows with that information to the user, as we

can see in the figure 5.4. The language of both of these messages depends on the browser language of the user.

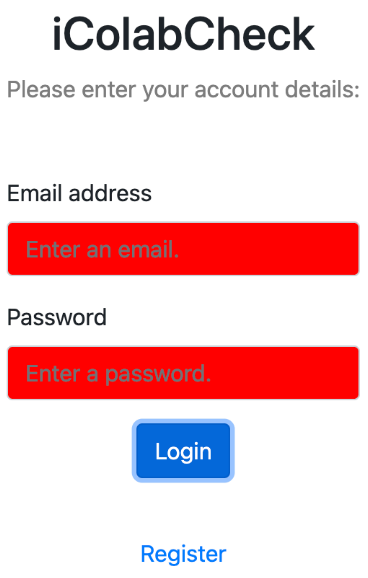
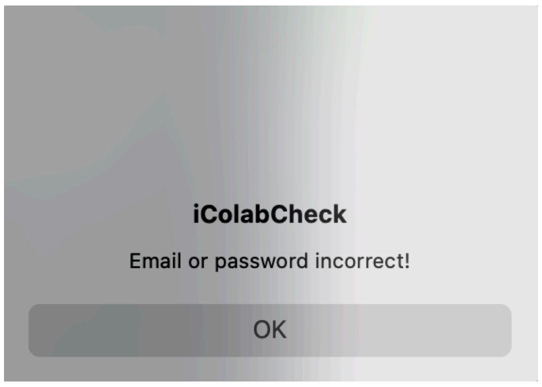
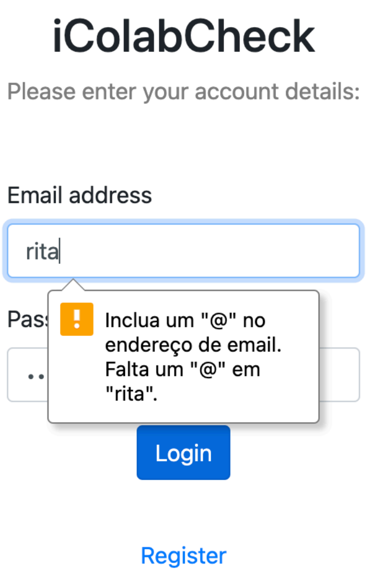


Figure 5.1: Unfilled fields in login.


iColabCheck

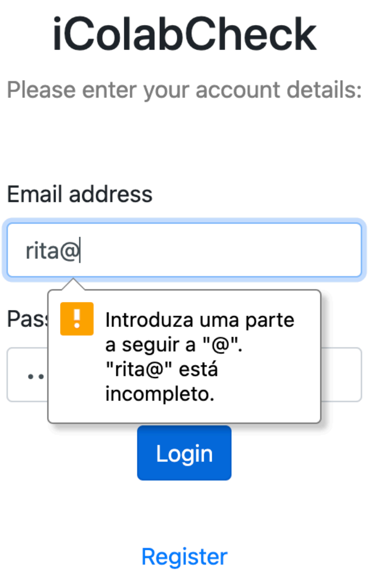
Email or password incorrect!

Figure 5.2: Wrong email or password alert.



Inclua um "@" no endereço de email. Falta um "@" em "rita".

Figure 5.3: Incorrect email formatting.

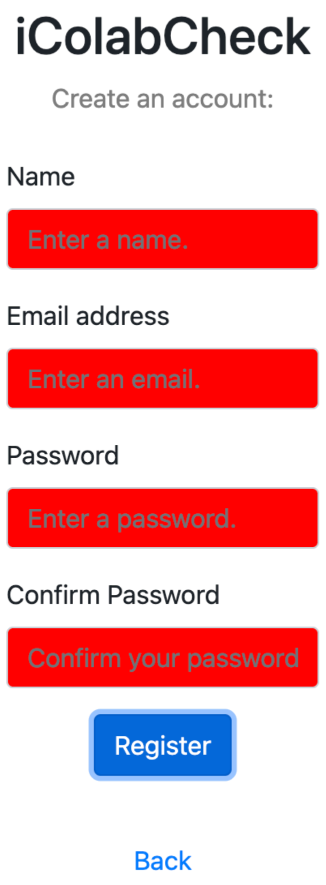


Introduza uma parte a seguir a "@". "rita@" está incompleto.

Figure 5.4: Incomplete email address.

5.2.2 Error Messages on Register Page

In the register form, if the email is badly formatted or incomplete, the same error messages from figures 5.3 and 5.4 also appear. Besides, as we can see in figure 5.5, if the user doesn't fill any of the required fields, those fields become red and with the message for the user to enter them. As we can see in figure 5.6, if the email that the user introduces is already in use, an alert will show with that information. Also, if the "password" and the "confirm password" fields don't match, a proper alert will appear, as we can see in figure 5.7.


iColabCheck

Create an account:

Name

Enter a name.

Email address

Enter an email.

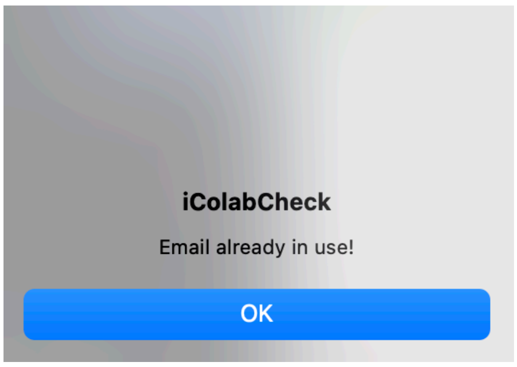
Password

Enter a password.

Confirm Password

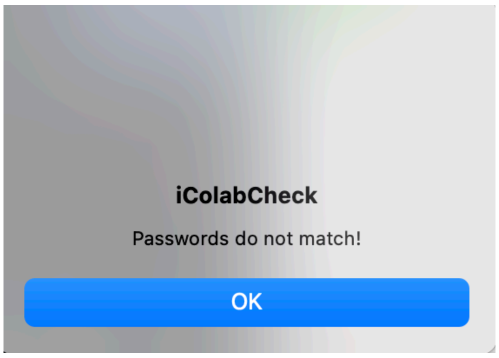
Confirm your password

Figure 5.5: Unfilled fields in register.


iColabCheck

Email already in use!

Figure 5.6: Email already in use alert.

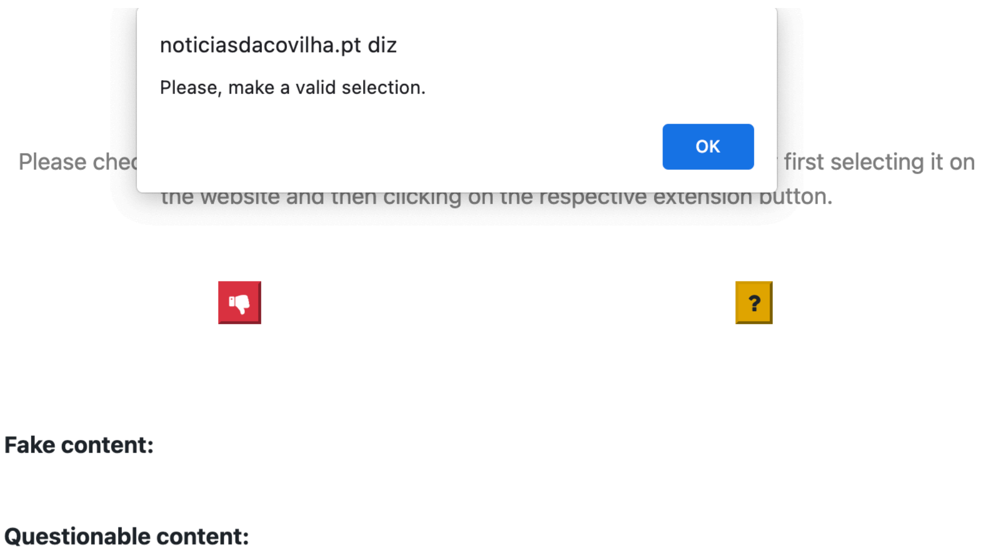

iColabCheck

Passwords do not match!

Figure 5.7: Passwords don't match alert.

5.2.3 Error Messages on Main Page

On the extension main page, after the user correctly authenticates, if he wants to rate some content as fake or questionable on a Web page, a valid selection must be made. So, if the user doesn't select anything at all and just clicks on the fake or questionable buttons, an alert appears with that information, as we can see in figure 5.8.


noticiasdacovilha.pt diz

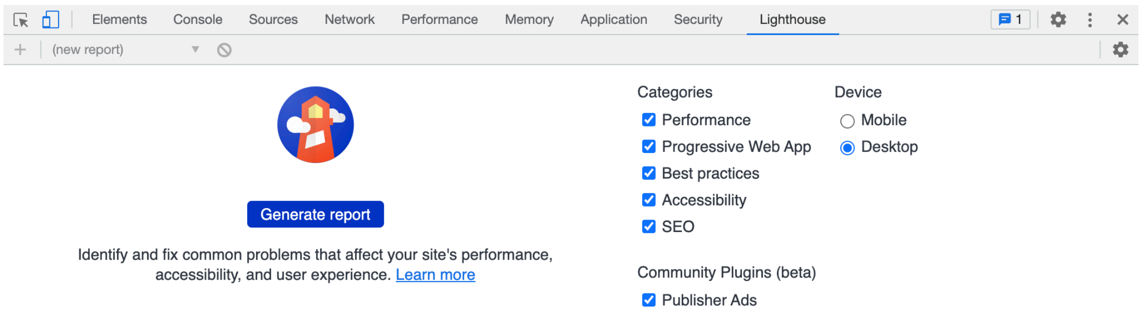
Please, make a valid selection.

Figure 5.8: Invalid selection alert.

5.3 Platform Tests

As the platform on the Journal side is currently a simple website for the content producers to check the respective insights from the extension, the testing phase was made through Chrome Dev Tools. It is a set of Web page inspection, debugging, and optimization tools that are built into the Google Chrome browser.
The *Lighthouse* is an open-source, automated tool for improving the quality of Web pages. You can run it against any Web page, public or requiring authentication. It has audits for performance, accessibility, progressive Web apps, SEO, and more, as we can see in figure 5.9. Thus, it runs a series of audits against the page, and then it generates a report on how well the page did. From there, you can use the failing audits as indicators of how to improve the page. Each audit has a reference doc explaining why the audit is important, as well as how to fix it. It can be run in Chrome Dev Tools, from the command line, or as a Node module [Goo].
In the figure 5.10 is the overview of the report generated through the *Lighthouse*. As we can see, the ”Performance” parameter carries the best score that the page got, and the ”Best Practices” the worst one. So, if we scroll through the complete report and go, for example, to the ”Best Practices” part, we have more detailed information about what can

be improved on this matter, as we can see in figure 5.11. In this case, we can see that some links to cross-origin destinations are unsafe and it has some front-end JavaScript libraries with vulnerabilities.



Identify and fix common problems that affect your site's performance, accessibility, and user experience. Learn more

Categories
- [✓] Performance
- [✓] Progressive Web App
- [✓] Best practices
- [✓] Accessibility
- [✓] SEO

Device
- Mobile
- [✓ Desktop]

Community Plugins (beta)
- [✓] Publisher Ads

Figure 5.9: Chrome Dev Tools - Lighthouse.

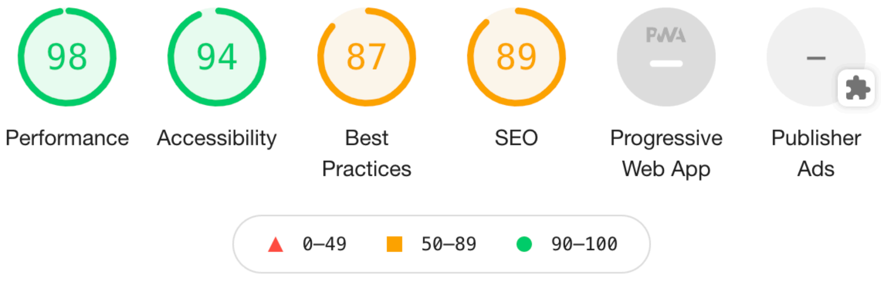
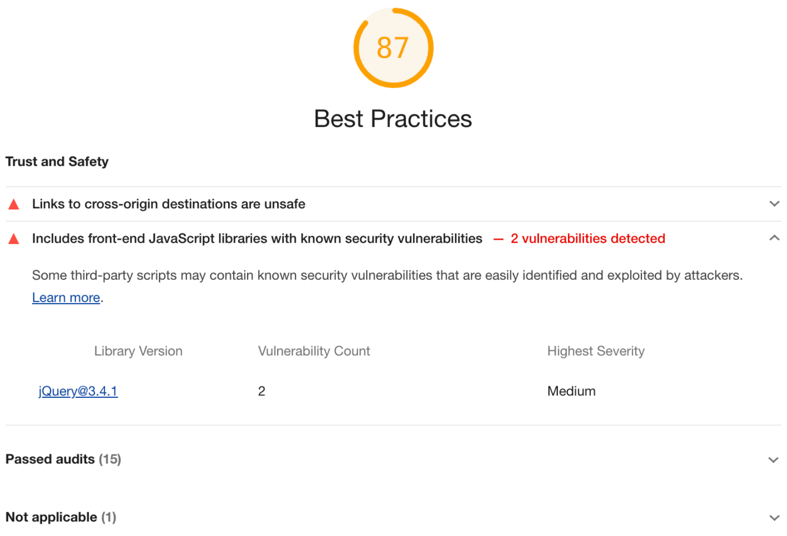


Figure 5.10: Lighthouse results.



Trust and Safety

▲ Links to cross-origin destinations are unsafe

▲ Includes front-end JavaScript libraries with known security vulnerabilities — 2 vulnerabilities detected

Some third-party scripts may contain known security vulnerabilities that are easily identified and exploited by attackers. Learn more.

Library Version	Vulnerability Count	Highest Severity
jQuery@3.4.1	2	Medium

Passed audits (15)

Not applicable (1)

Figure 5.11: Lighthouse results - best practices.

5.4 Conclusion

In this chapter was approached all the tests performed in both the extension and the Web platform. The extension tests were done manually through the implementation process and take into account the user experience on this system usage. From those tests, we saw some error messages directly on the extension incorrect fields and also some alerts of what went wrong. On the platform case, as it is a simple website at the moment, the tests were performed through the *Lighthouse*, which is a tool from Chrome Dev Tools that analyzes several parameters such as performance, accessibility, best practices, and SEO. The final results were good since the score ranged between 87 and 98.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this Master's dissertation project report, were covered in detail all the relevant topics for the construction of a Chrome extension capable of assisting journalists in verifying the veracity of published news, through users feedback and interaction with the tool.
Therefore, in chapter 2, we firstly analyzed the difference between Native, Web, and Progressive Web applications, having the PWA the best reach and capabilities compared to the other two options.
Then, we reviewed the main client-side technologies (HTML, CSS and JavaScript) and their importance and role in frontend development.
The Chrome extensions section exhaustively approaches the definition, concept, and architecture aspects of a Chrome extension, as well as its major vantages and possible vulnerabilities.
The last sub-section of section 2.1 in chapter 2, was all about how mobile devices are transforming the way that journalism is practiced and how fake news proliferation is increasing with the huge information sharing through the network.
The section 2.3 of the same chapter, detailed some projects and existing tools in the fact-checking domain and explained how fake news detection mechanisms are a very important but challenging task. Therefore, different approaches and existing methods for fake news and distorted image detection were presented.
Additionally, at the end of chapter 2, it was presented some relevant fact-checking APIs and two already existing Chrome extensions for the same purpose.
In the third chapter, were presented the functional and non-functional requirements of both the extension and the platform, as well as the use-cases. Still on this chapter, was presented the system's DB and architecture in a 3 sides perspective and the used technologies on this Master's dissertation project.
In chapter four - Implementation and Results - were presented all the implementation phases and main functions of the extension and the platform. For this purpose, some relevant code snippets were shown and explained as well as some print screens of the respective visual results.
The fifth chapter was all about the tests that were done in both the extension and the Web platform. For the extension, it was performed manual tests in order to see the error messages and alerts presented to the user in the case that something goes wrong. In the case of the platform, the testing was done through the Chrome Dev Tools using the *Lighthouse* which generates reports to analyze de Web page performance, accessibility, best practices, and SEO.

I can conclude that all the initial project objectives were accomplished and some more were created and started through this development and implementation time, such as the expansion of the extension to all the websites and not only for one regional Journal, as well as the creation of the platform for the content producers to check some insights.

6.2 Future Work

As future work I would like to improve the authentication mechanism, including putting a number of minimum characters for the password, an email confirmation, the ”forgot your password?” option, and also improve some error messages for the user.
On the extension main page, I would like to create some more buttons so the users can make a more precise evaluation of the selected content. Also, I want to improve the design of the extension itself, create a comment section for the users to explain why they have selected a certain piece of text as fake or questionable, optimize some functions related to the user visualization of the selected content on the extension and implement a logout button and system.
In what concerns the extension functioning, I would like to implement better the chrome local storage so the users don't automatically logout when they click somewhere outside the extension popup. Also, as future work, I want to put the extension online and available for all the users and find an online server for the DB to operates.
The platform is pretty trivial at this point. Thus, I would like to develop an authentication system, as well as more statistics about the data collected (e.g. most criticized website monthly, most questionable website weekly, user's profiling, most trusted user, etc).

Bibliography

[AAQHA18] M. Alrubaian, M. Al-Qurishi, M. M. Hassan, and A. Alamri. A credibility analysis system for assessing information on twitter. *IEEE Transactions on Dependable and Secure Computing*, 15(4):661–674, 2018. 23

[Alb] Alberto Sardinha. Modelo entidade-associação (ea) [online]. [Online]. Available from: `https://fenix.tecnico.ulisboa.pt/downloadFile/3779579572812/mod02-1-Modelo-EA.pdf` [cited August 16th 2021]. 47

[Ale20] Alexander S. Gillis. What is rest api (restful api)? [online]. September 2020. [Online]. Available from: `https://searchapparchitecture.techtarget.com/definition/RESTful-API` [cited October 3rd 2021]. 49, 50

[Ang] Angular. Angular [online]. [Online]. Available from: `https://angular.io/` [cited January 19th 2021]. 8

[Api21] Apiary. Fact-checker api · apiary [online]. 2021. [Online]. Available from: `https://factualkm.docs.apiary.io/#` [cited January 15th 2021]. 26, 27

[Ari21] Ariane G. O que é css? guia básico para iniciantes [online]. July 2021. [Online]. Available from: `https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/` [cited August 17th 2021]. 50

[BCJ$^+$14] Lujo Bauer, Shaoying Cai, Limin Jia, Timothy Passaro, and Yuan Tian. Analyzing the dangers posed by chrome extensions. 10 2014. 11, 17

[BE19] Joshua Braun and Jessica Eklund. Fake news, real money: Ad tech platforms, profit-driven hoaxes, and the business of journalism. *Digital Journalism*, 7:1–21, 01 2019. 20

[BO16] M. Bouazizi and T. Otsuki. A pattern-based approach for sarcasm detection on twitter. *IEEE Access*, 4:5477–5488, 2016. 24

[Booa] Bootstrap. History [online]. [Online]. Available from: `https://v4-alpha.getbootstrap.com/about/history/` [cited August 17th 2021]. 51

[Boob] Bootstrap. Introduction [online]. [Online]. Available from: `https://getbootstrap.com/docs/4.1/getting-started/introduction/` [cited August 17th 2021]. 51

[BTCG19] Ana Bernal-Triviño and Judith Clares-Gavilán. Uso del movil y las redes sociales como canales de verificacion de fake news. el caso de maldita.es. *El Profesional de la Informacion*, 28:e280312, 05 2019. 18

[CFW12] Nicholas Carlini, Adrienne Felt, and David Wagner. An evaluation of the google chrome extension security architecture. 01 2012. 17

[CG20] Narciso Inês Moreno José Miguel Crespo Palma Nuno Sepúlveda Rita Cardoso Gustavo, Martinho Ana P. Information and misinformation on the coronavirus in portugal. *European Journal of Operational Research*, pages 1–43, April 2020. Available from: `https://medialab.iscte-iul.pt/wp-content/uploads/information-and-misinformation-on-the-coronavirus-in-portugal.pdf`. 22

[Chr12a] Chrome Developers. Architecture overview [online]. September 2012. [Online]. Available from: `https://developer.chrome.com/docs/extensions/mv3/architecture-overview/#pages` [cited February 3th 2021]. xiii, 14, 15, 16

[Chr12b] Chrome Developers. Manage events with background scripts [online]. September 2012. [Online]. Available from: `https://developer.chrome.com/docs/extensions/mv2/background_pages/` [cited February 3th 2021]. 14

[Chr13] Chrome Developers. What are extensions? [online]. February 2013. [Online]. Available from: `https://developer.chrome.com/docs/extensions/mv2/overview/` [cited February 2th 2021]. 11

[Cod18] CodeIT. The importance of chrome extension development [online]. April 2018. [Online]. Available from: `https://medium.com/@codeit_llc/the-importance-of-chrome-extension-development-b8cbb9405bf2` [cited February 3th 2021]. 16, 17

[Cor17] Cory LaViska. Hashing passwords with node.js and bcrypt [online]. February 2017. [Online]. Available from: `https://www.abeautifulsite.net/posts/hashing-passwords-with-nodejs-and-bcrypt` [cited August 19th 2021]. 56

[Dan] Daniel Boterhoven. Native, web and hybrid apps what's the difference? [online]. [Online]. Available from: `https://denimdev.com.au/blog/native-web-and-hybrid-apps-whats-the-difference/` [cited January 18th 2021]. 6

[Deva] DevMedia. Html básico - códigos html [online]. [Online]. Available from: `https://www.devmedia.com.br/html-basico-codigos-html/16596` [cited January 21th 2021]. 10

[Devb] DevMedia. Javascript tutorial [online]. [Online]. Available from: `https://www.devmedia.com.br/javascript-tutorial/37257` [cited January 21th 2021]. 11

[Devc] DevMedia. Primeiros passos no html5, javascript e css3 [online]. [Online]. Available from: `https://www.devmedia.com.br/primeiros-passos-no-html5-javascript-e-css3/25647` [cited January 21th 2021]. 10, 11

[Edu] Eduardo Figueiredo. Requisitos funcionais e requisitos não funcionais [online]. [Online]. Available from: `https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/req-funcional-rnf_v01.pdf` [cited August 3th 2021]. 36

[Exp] Express. Express routing [online]. [Online]. Available from: `https://expressjs.com/en/guide/routing.html` [cited September 9th 2021]. 69

[Fac] Factmata. Empowering everyone with a better understanding of content. [online]. [Online]. Available from: `https://factmata.com/` [cited January 15th 2021]. 24

[FB18] D. Fortunato and J. Bernardino. Progressive web apps: An alternative to the native mobile apps. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6, 2018. 5, 7

[Gig19] Gigvy. What's the difference between native vs. web vs. hybrid apps? [online]. 2019. [Online]. Available from: `https://getgist.com/difference-between-native-vs-web-vs-hybrid-apps/` [cited January 18th 2021]. 6

[Goo] Google Developers. Lighthouse [online]. [Online]. Available from: `https://developers.google.com/web/tools/lighthouse` [cited September 16th 2021]. 78

[Goo21a] Google. Fact check markup for search | google search central [online]. 2021. [Online]. Available from: `https://developers.google.com/search/docs/data-types/factcheck` [cited January 16th 2021]. xiii, 29

[Goo21b] Google. Fact check tools [online]. 2021. [Online]. Available from: `https://toolbox.google.com/factcheck/explorer` [cited January 16th 2021]. xiii, 28

[Goo21c] Google. Fact check tools [online]. 2021. [Online]. Available from: `https://toolbox.google.com/factcheck/about` [cited January 16th 2021]. xiii, 28

[Goo21d] Google. Fact check tools api | google developers [online]. 2021. [Online]. Available from: `https://developers.google.com/fact-check/tools/api` [cited January 16th 2021]. 29

[Gra] Grant Glas. Web app vs. native app [online]. [Online]. Available from: `https://www.app-press.com/blog/web-app-vs-native-app` [cited January 18th 2021]. 6

[HA17] Benjamin D. Horne and Sibel Adali. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *CoRR*, abs/1703.09398, 2017. Available from: `http://dblp.uni-trier.de/db/journals/corr/corr1703.html#HorneA17`. 20

[Hoa] Hoaxy. Hoaxy: How claims spread online [online]. [Online]. Available from: https://hoaxy.osome.iu.edu/ [cited January 15th 2021]. 24, 27

[IBM21] IBM Cloud Education. What is a rest api? | ibm [online]. April 2021. [Online]. Available from: https://www.ibm.com/cloud/learn/rest-apis [cited October 3rd 2021]. 49, 50

[Ion] Ionic. Ionic - cross-platform mobile app development [online]. [Online]. Available from: https://ionicframework.com/ [cited January 19th 2021]. 8

[Iva] Ivan Messina. Mamp: the definitive guide [online]. [Online]. Available from: https://supporthost.com/mamp/ [cited August 17th 2021]. 52

[jch21] jchiang. Newscracker [online]. 2021. [Online]. Available from: https://chrome.google.com/webstore/detail/newscracker/lmpfanpnpoaegbafkodbifallmfcncpb [cited January 18th 2021]. 31, 32

[JJZ+17] Z. Jin, Cao J., Y. Zhang, J. Zhou, and Q. Tian. Novel visual and statistical image features for microblogs news verification. *IEEE Transactions on Multimedia*, 19(3):598–608, 2017. 22

[KABAK] Asharul Islam Khan, Ali Al-Badi, and Mahmood Al-Kindi. Progressive web application assessment using ahp. *Procedia Computer Science*, 155:289–294. 10

[Kal19] Kalev Leetaru. Are smartphones making ”fake news” and disinformation worse? [online]. 5 2019. [Online]. Available from: https://www.forbes.com/sites/kalevleetaru/2019/05/01/are-smartphones-making-fake-news-and-disinformation-worse/?sh=7c68989b49b2 [cited January 13th 2021]. 20

[Kni17] Knight, Will. More evidence that humans and machines are better when they [online]. 2017. [Online]. Available from: https://www.technologyreview.com/2017/11/08/147809/more-evidence-that-humans-and-machines-are-better-when-they-team-up/ [cited January 15th 2021]. 26

[Kom17] Kashyap Kompella. Can Machine Learning Help Fight Fake News? *EContent*, 40(5):40–40, October 2017. Available from: http://www.econtentmag.com/Articles/Column/Cognito-Ergo-Sum/Can-Machine-Learning-Help-Fight-Fake-News-119979.htm. 23, 26

[Lus20] Lusa. Investigadores criam sistema para detetar informações falsas nas redes sociais [online]. 5 2020. [Online]. Available from: https://combatefakenews.lusa.pt/fake-news-investigadores-criam-sistema-para-detetar-informacoes-falsas-nas-redes-sociais-2/ [cited January 15th 2021]. 22

[Mal] Maldita. Maldita.es — periodismo para que no te la cuelen - creamos herramientas periodísticas para que no te la cuelen: Maldita. [online]. [Online]. Available from: `https://maldita.es/` [cited January 15th 2021]. 25

[Med21] Media Bias/Fact Check. Media bias/fact check - search and learn the bias of news media [online]. 2021. [Online]. Available from: `https://mediabiasfactcheck.com/` [cited January 16th 2021]. 29, 30, 31

[Mic17] Michael Guta. What is a google chrome extension? [online]. May 2017. [Online]. Available from: `https://smallbiztrends.com/2017/05/google-chrome-extension.html` [cited February 2th 2021]. 11, 16, 17

[Mik21] Mike Crowe. Media bias/fact check extension [online]. 2021. [Online]. Available from: `https://chrome.google.com/webstore/detail/media-biasfact-check-exte/ganicjnkcddicfioohdaegodjodcbkkh/related` [cited January 16th 2021]. 29

[Moz] Mozilla. O que é javascript? [online]. [Online]. Available from: `https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/O_que_e_JavaScript` [cited January 21th 2021]. 11

[New] Newtral. Newtral - periodismo, fact-cheking, tecnología y datos. [online]. [Online]. Available from: `https://www.newtral.es/` [cited January 15th 2021]. 25

[Nod] Nodejs. Introduction to node.js [online]. [Online]. Available from: `https://nodejs.dev/learn` [cited August 17th 2021]. 52

[npma] npmjs. body-parser [online]. [Online]. Available from: `https://www.npmjs.com/package/body-parser` [cited August 19th 2021]. 56

[npmb] npmjs. express [online]. [Online]. Available from: `https://www.npmjs.com/package/express` [cited August 19th 2021]. 55

[npmc] npmjs. mysql2 [online]. [Online]. Available from: `https://www.npmjs.com/package/mysql2` [cited August 19th 2021]. 56

[npmd] npmjs. nodemon [online]. [Online]. Available from: `https://www.npmjs.com/package/nodemon` [cited August 19th 2021]. 56

[OAA$^+$18] Efeosasere Okoro, Benjamin Abara, Umagba Alex, Anyalewa Ajonye, and Zayyad Isa. A hybrid approach to fake news detection on social media. *Nigerian Journal of Technology*, 37, 04 2018. 25, 26

[OAI$^+$19] Efeosasere Okoro, Benjamin Abara, Zayyad Isa, Umagba Alex, and Anyelewa Alan-Ajonye. Effects of human and human-machine fake news detection approaches on user detection performance. *International Journal of Advanced Computer Research*, 10:26–32, 02 2019. 25

[Ohi] Ohio University. Fake news, misinformation, fact-checking | ohio university mpa | ohio university [online]. [Online]. Available from: https://onlinemasters.ohio.edu/masters-public-administration/guide-to-misinformation-and-fact-checking/ [cited October 2nd 2021]. vii, ix

[Ora21] Oracle. Apiary | platform for api design, development documentation [online]. 2021. [Online]. Available from: https://apiary.io/ [cited January 15th 2021]. 26

[QWLT16] Yumeng Qin, Dominik Wurzer, V. Lavrenko, and Cunchen Tang. Spotting rumors via novelty detection. *ArXiv*, abs/1611.06322, 2016. 24

[Rea] React. React – a javascript library for building user interfaces [online]. [Online]. Available from: https://reactjs.org/ [cited January 19th 2021]. 8

[Red20] Red Hat. What is a rest api? [online]. May 2020. [Online]. Available from: https://www.redhat.com/en/topics/api/what-is-a-rest-api [cited October 3rd 2021]. 49

[Rei17] R. Reinhardt. Progressive web apps: A new hope. *Streaming Media*, 14:18–19, 2017. 8

[RPS19] Felipe Rêgo, Filipe Portela, and Manuel Filipe Santos. Towards pwa in healthcare. *Procedia Computer Science*, 160:678 – 683, 2019. Available from: http://www.sciencedirect.com/science/article/pii/S1877050919317284. 10

[Sam20] Sam Richard and Pete LePage. What are progressive web apps? [online]. 2020. [Online]. Available from: https://web.dev/what-are-pwas/ [cited January 19th 2021]. xiii, 7

[SBLP$^+$20] Ramón Salaverría, Nataly Buslón, Fernando López Pan, Bienvenido León, Ignacio López-Goñi, and María Erviti. Desinformación en tiempos de pandemia:tipología de los bulos sobre la covid-19. *El Profesional de la Informacion*, 29:e290315, 05 2020. 24

[SDAB16] S. S.Elkasrawi, A. Dengel, A. Abdelsamad, and S. S. Bukhari. What you see is what you get? automatic image verification for online news content. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 114–119, 2016. xiii, 23

[SMW$^+$18] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. preprint on webpage, 09 2018. Available from: https://www.researchgate.net/publication/327464821_FakeNewsNet_

A_Data_Repository_with_News_Content_Social_Context_and_Dynamic_Information_for_Studying_Fake_News_on_Social_Media. 20, 21

[Sno] Snopes. Snopes.com. [online]. [Online]. Available from: https://www.snopes.com/ [cited January 15th 2021]. 25

[SQJ+19] Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. Combating fake news: A survey on identification and mitigation techniques. *ACM Transactions on Intelligent Systems and Technology*, 10:1–42, 04 2019. 20

[TJ18] Sayali Tandel and Abhishek Jamadar. Impact of progressive web apps on web app development. *International Journal of Innovative Research in Science, Engineering and Technology*, 7, 09 2018. xv, 9

[Tru] TruthOrFiction. Truth or fiction? - seeking truth, exposing fiction. [online]. [Online]. Available from: https://www.truthorfiction.com/ [cited January 15th 2021]. 25

[tru21] truthy. Rapidapi [online]. 2021. [Online]. Available from: https://rapidapi.com/truthy/api/hoaxy?endpoint=57cf3befe4b0e451de067c69 [cited January 15th 2021]. 27

[Tuta] Tutorialspoint. Javascript - overview [online]. [Online]. Available from: https://www.tutorialspoint.com/javascript/javascript_overview.htm [cited August 17th 2021]. 51

[Tutb] Tutorialspoint. Php - introduction [online]. [Online]. Available from: https://www.tutorialspoint.com/php/php_introduction.htm [cited August 17th 2021]. 51

[Tutc] Tutorialspoint. What is css? [online]. [Online]. Available from: https://www.tutorialspoint.com/css/what_is_css.htm [cited January 21th 2021]. 10

[Uma16] Sajid Umair. Mobile reporting and journalism for media trends, news transmission and its authenticity. *Journal of Mass Communication & Journalism*, 6:323, 2016. 18

[VJ13] Wencui; Kisekka Victoria; Sharman Raj; Kudumula Vidyadhar; Venkatesan, Srikanth; Han and Hardeep Singh Jaswal. Misinformation in online health communities. december 2013. Available from: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1038&context=wisp2012. 21

[VR14] Andreas Vlachos and Sebastian Riedel. Fact checking: Task definition and dataset construction. pages 18–22, 01 2014. 20

[Vue] Vue. Vue.js [online]. [Online]. Available from: https://vuejs.org/ [cited January 19th 2021]. 8

[VVY19] Dinesh Kumar Vishwakarma, Deepika Varshney, and Ashima Yadav. Detection and veracity analysis of fake news via scrapping and authenticating the web search. *Cognitive Systems Research*, 58:217 – 229, 2019. Available from: `http://www.sciencedirect.com/science/article/pii/S1389041719301020`. 18, 21

[W3Sa] W3Schools. jquery introduction [online]. [Online]. Available from: `https://www.w3schools.com/jquery/jquery_intro.asp` [cited August 17th 2021]. 51

[W3Sb] W3Schools. Php introduction [online]. [Online]. Available from: `https://www.w3schools.com/php/php_intro.asp` [cited August 17th 2021]. 51

[Wil19] William Bittencourt Moraes. Criando extensões/apps para o chrome do básico ao avançado [online]. October 2019. [Online]. Available from: `https://www.udemy.com/course/chrome-extensions-do-basico-ao-avancado/` [cited February 3th 2021]. xiii, 12, 13, 14, 16

[WS15] Cornelia Wolf and Anna Schnauber. News consumption in the mobile era. *Digital Journalism*, 3(5):759–776, 2015. vii, ix

[ZG20a] Xichen Zhang and Ali A. Ghorbani. An overview of online fake news: Characterization, detection, and discussion. *Information Processing Management*, 57(2):102025, 2020. Available from: `https://www.sciencedirect.com/science/article/pii/S0306457318306794`. 18, 24

[ZG20b] Xichen Zhang and Ali A. Ghorbani. An overview of online fake news: Characterization, detection, and discussion. *Information Processing Management*, 57(2):102025, 2020. Available from: `http://www.sciencedirect.com/science/article/pii/S0306457318306794`. 19, 20

[ZGK+19] Chaowei Zhang, Ashish Gupta, Christian Kauten, Amit V. Deokar, and Xiao Qin. Detecting fake news for reducing misinformation risks using analytics approaches. *European Journal of Operational Research*, 279(3):1036 – 1052, 2019. Available from: `http://www.sciencedirect.com/science/article/pii/S0377221719304977`. 21