Automated web scraping and data visualisation for tourism based on popular accommodation platforms

Sérgio Rosa Pereira

UMinho | 2022

Universidade do Minho
Escola de Letras, Artes e Ciências Humanas

Sérgio Rosa Pereira

Automated web scraping and data visualisation for tourism based on popular accommodation platforms

October 2022

Universidade do Minho
Escola de Letras, Artes e Ciências Humanas

Sérgio Rosa Pereira

Automated web scraping and data visualisation for tourism based on popular accommodation platforms

Master dissertation
Master's in Digital Humanities

Dissertation supervised by:
Professor Doutor Sérgio Adriano Fernandes Lopes
Professora Doutora Sílvia Lima Gonçalves Araújo

October 2022

ACKNOWLEDGEMENTS

Firstly, I would like to thank my parents for all the support given during my academic journey.

I also wish to express my sincere thanks and appreciation to my supervisors Dr. Sílvia and Dr. Sérgio, for the opportunity to participate in this project and for all the help and feedback given during its course.

Lastly, I also want to thank all my friends and colleagues, for accompanying and assisting me throughout all these years.

STATEMENT OF INTEGRITY

ABSTRACT

AUTOMATED WEB SCRAPING AND DATA VISUALISATION FOR TOURISM BASED ON POPULAR ACCOMMODATION PLATFORMS

The project developed is part of "Programa INTERREG V A España – Portugal (POCTEP)", on which several entities collaborate in cross-border projects, with the main goal of securing the sustainability, innovation and efficient management of tourism resources in Portugal and Spain, while also harmonising the use of technology in the tourism sector.

Through web scraping and data visualisation techniques, information regarding tourists and their destinations was extracted from online platforms, being then organised and interpreted, in order to obtain useful insights. With the Python programming language as this project's main pillar, an automated web scraping tool was designed, with a custom user interface to facilitate access. Then, after the cleaning of data using regular expressions and text replacement, several graphs were conceived, followed by a data visualisation dashboard which also allows interaction with those graphs. In the end, the whole process was automated, allowing this method to periodically monitor the targeted tourism areas with efficiency.

Thus, through this self-sufficient competitive vigilance system, an effective management of the tourism sector resources can be ensured.

KEYWORDS: Web Scraping; Tourism; Python; Data Visualisation; Data Science.

RESUMO

WEB SCRAPING E VISUALIZAÇÃO DE DADOS DE TURISMO AUTOMATIZADOS, COM BASE EM PLATAFORMAS POPULARES DE ALOJAMENTO

O projeto desenvolvido faz parte do "Programa INTERREG V A España – Portugal (POCTEP)", no qual diversas entidades colaboram em projetos transfronteiriços, com o principal objetivo de assegurar a sustentabilidade, inovação e gestão eficiente dos recursos turísticos em Portugal e Espanha, harmonizando também o uso da tecnologia no setor turístico.

Através de técnicas de web scraping e de visualização dados, foi extraída de plataformas turísticas informação relativa aos turistas e aos seus destinos turísticos, sendo então organizada e interpretada, de forma a obter as suas perceções. Com a linguagem de programação Python como o principal pilar deste projeto, uma ferramenta de web scraping automatizada foi criada, com uma interface de utilizador customizada, para facilitar o acesso. Então, após a limpeza dos dados usando expressões regulares e substituição de texto, vários gráficos foram concebidos, seguidos de uma dashboard de visualização de dados que também permite interação com esses dados. No fim, o processo todo foi automatizado, permitindo que este método analise periodicamente as áreas-alvo de turismo com eficácia.

Assim, através deste sistema de vigilância competitiva autossuficiente, uma gestão eficiente dos recursos do sector turístico pode ser assegurada.

PALAVRAS-CHAVE: Web Scraping; Turismo; Python; Visualização de Dados; Ciência de Dados.

CONTENTS

LIST OF ACRONYMS

API - Application Programming Interface

CSS - Cascading Style Sheets

CSV - Comma-separated values

DDoS - Distributed Denial-of-Service

GPS - Global Positioning System

GUI – Graphical User Interface

HTML - Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

ICT - Information and Communications Technology

IDE - Integrated development environment

IP - Internet Protocol

Regex - Regular expression

UI - User Interface

URL - Uniform Resource Locators

WWW - World Wide Web

Wi-Fi – Wireless Fidelity

XML - Extensible Markup Language

LIST OF FIGURES

LIST OF LISTINGS

1. INTRODUCTION

In this first chapter, my Master's dissertation is introduced, with a description of the funded program on which this project was carried out, while also focusing on the objectives that motivate this project's development.

1.1. CONTEXT - POCTEP

This report presents a description, analysis and conclusion of the tasks I performed during my curricular internship, which took place between March 2021 and February 2022, as part of my 2nd year in the Master's Degree in Digital Humanities.

This project is part of "Programa INTERREG V A España – Portugal (POCTEP)", a program coordinated by AMTEGA (Agencia para la Modernización Tecnológica de Galicia) in collaboration with "Agencia de Turismo de Galicia", "Turismo do Porto e Norte de Portugal", "Centro de Computação Gráfica da Universidade do Minho", Instituto Politécnico de Viana do Castelo", among others. With the support of the European Union, it promotes cross-border projects between Portugal and Spain, with a high focus on tourism, sustainability and innovation.

1.2. MOTIVATION

Tourism is one of the biggest financial sectors in Portugal and Spain, and, due to that substantial economic impact, it is of high importance to monitor the sector and ensure its development. As its success depends on the tourists, it is necessary to collect data, in order to obtain insights and find efficient management strategies for the tourism sector.

A huge part of the data related to tourists has an enormous value, as in the tourism sector there is a lot of competitiveness and it is up to each competitor to analyse that data, as a means to determine which factors influence the tourists the most, when deciding their travelling destinations. Economically speaking, acquiring data is expensive. However, the Internet contains, available for anyone, large sources of data which only need to be extracted and treated to be used.

1.3. OBJECTIVES

With a focus on Galicia and the Northern and Center regions of Portugal, this project aims to use these strategies of free data collection with the main objective of conceiving a system of competitive vigilance. It should be able to capture disparities between the several touristic destinations, based on the prices, accommodation scores, user comments, and so on. This gives us the understanding of which factors make certain areas be more successful than the others.

My personal objective for this project is to conceive a system which can, almost entirely autonomously, extract data from where I want, export that data into datasets and create visual representations of that data. For that to be possible, I will create a method with which I need to achieve the following goals:

1. Extract data from tourism platforms;
2. Export extracted data into datasets;
3. Create an intuitive user interface on which my web scraping code will run;
4. Clean the data from the datasets, making them “readable” for the next step;
5. Create graphs to represent the extracted data visually;
6. Create an interactive dashboard that presents the graphs previously created;
7. Automate the process of extraction in order to obtain data every day.

1.4. RESEARCH APPROACH

To accomplish this project’s goals, I carried out a research methodology based on literature revision.

Firstly, the target areas were chosen, keeping in mind the cross-borders context of this project’s program. Based on the accommodation availability on those areas, the tourism platforms to extract data from were then selected.

Secondly, bibliographic research on tourism was performed, focusing specifically on tourism in Portugal, while also tackling the role of technology in this sector and briefly correlating it to the Covid-19 pandemic. Then, the definition and studying of indicators relevant for extraction was executed, paving the way for the extraction of data.

Subsequently, I made bibliographic research of the state of the art in web scraping, covering all types of scraping, software tools available and problems in the field, ending with an analysis on web scraping in tourism, based on the methodologies of previous works accomplished by other researchers.

To finish the state of the art research, I tackled data visualisation, focusing on representation of tourism data.

Then, the method was executed, beginning with the scraping of web data, followed by the creation of datasets and the creation of a GUI (Graphical User Interface) to run my program. Afterwards, the cleaning of data was performed, setting the path for the data visualization portion of this process, on which I created graphs and an interactive dashboard to display them. This whole method is finalised with the implementation of automation, allowing me to schedule and perform extractions without needing user interaction.

After discussing the problems encountered, conclusions are drawn, thus completing this project.

1.5. DOCUMENT STRUCTURE

This report is structured in five chapters. In the first chapter, I introduced the research subject, giving context on this project's nature and its main objectives.

The second chapter presents an overview on the state of the art in both tourism and web scraping.

Chapter 3 contains the methodology proposed, while the fourth chapter implements that methodology, conceiving a system that ranges from data extraction to data visualisation, also including the automation of the whole process.

This document ends with chapter 5, on which I reveal the conclusions of this project.

2. STATE OF THE ART

2.1. TOURISM

Firstly, it is essential to define the concept of "tourism". Mathieson and Wall (1990) define tourism as "the temporary movement of people to destinations outside their normal places of work and residence, the activities undertaken during their stay in those destinations, and the facilities created to cater to their needs". It is one of the world's most important economic activities, impacting every single place on the globe. Given this impact and popularity, there is a necessity to discover strategies that ensure this field's development and sustainability. In this first section, I will tackle the role of tourism in Portugal, followed by a brief analysis of technology in the field, ending with a dissection of the global pandemic's effects on the tourism sector.

2.1.1. TOURISM IN PORTUGAL

In Portugal, there is no doubt that tourism is a major influencer in economic growth, so there have been more and more initiatives and funding for the evolution and effective management of the tourism sector. That said, what makes Portugal such a successful tourism destination? In 2017, 2018 and 2019, Portugal was elected "Best Destination in the World" at the World Travel Awards. Furthermore, the country is known for its sun, beaches, gastronomy, culture and high territorial diversity, thus covering all types of tourism, including sun and sea tourism (due to the long coastline), residential tourism (Portugal is a country with a high social level), religious tourism (which is quite appealing, as it covers all ages, sex and social classes) and rural tourism (Portugal has a large number of mountains, villages, natural parks, among others).

With the high demand in the Portuguese tourism sector, there is also a huge amount of tourism data which must be studied in order to understand in detail the reason for this success, thus being able to invest in more ways to contribute to the economic growth of this sector. By learning what influences tourists when it is time to decide their travel destinations and acting accordingly, we can try to prevent Portugal from losing tourists to other countries. There is no doubt that the countries that compete with Portugal in the tourism sector also invest in data collection and analysis, in order to create predictive analytics models. That way, they can find out what makes tourists choose Portugal instead of any other destination. The tourism sector is a competition and only those who study their opponents win.

2.1.2. TECHNOLOGY IN TOURISM

According to Buhalis and O'Connor (2005), Information and Communication Technologies (ICTs) have been transforming tourism globally. The developments made in technology in the last two decades allowed several fields to exponentially grow, and tourism was no exception.

Xiang (2018) divides the knowledge development on IT and tourism in two different eras: the era of "digitization" (1997-2006) and the era of "acceleration" (2007-2016). The first era was defined by the appearance and development of the Internet as a commercial tool, while the second era highlights the appearance of numerous technologies, such as Wi-Fi, search engines, the smartphone, machine learning and artificial intelligence, all of which contribute to the increasing role of Internet on tourism. Defined as "e-tourism", Buhalis (2003) states that it refers to a phenomenon and research area in which the adoption of ICTs transforms the processes and the value chains in the tourism industry.

> ICTs have profound implications for tourism and e-tourism reflects the digitization of all processes and value chains in the tourism, travel, hospitality and catering industries. Tactically, e-tourism enables organizations to manage their operations and undertake e-commerce. Strategically, e-tourism revolutionizes business processes, the entire value chain as well as strategic relationships with stakeholders. E-tourism determines the competitiveness of organizations by taking advantage of intranets for reorganising internal processes, extranets for developing transactions with trusted partners and the Internet for interacting with all stakeholders. (Buhalis 2003)

While the use of ICTs vastly improves the development and management of tourism resources, it also leads to more data being generated, which, not only in tourism across-the-board, but also in the context of this project, means more information to be analysed, which subsequently leads to more valuable insights to be gained.

> The wide use of ICT by tourism businesses and tourists generates a large amount of data from information searches, transactions, and spatial movement. Today's tourists will likely carry many technology gadgets and use them to interact with ICT resources. A tourist will generate and contribute a tremendous amount of data, including data points in a tourism

> website's analytics data, a hotel mobile app's log data, call center logs, the amount of traffic at a destination, the sales records of tourism services, search engine query volumes, social media mentions, location data from cell phones, GPS and photos, etc. All of these are potential indicators of a tourist's likes and dislikes, motivations, planning behavior, and actual stay experiences. (Pan, 2015)

Pan (2015) also states that innovative and predictive research is needed to direct the tourism industry into the right direction, using the insights obtained in order to defined future marketing and management strategies, which is a notion that is supported by several studies I have encountered during my bibliographic research. That same notion is also defended by myself, in concordance with the techniques used and results obtained with this project.

2.1.3. COVID AND TOURISM

The global pandemic, that started in 2019, revolutionised the world's economy. With citizens confined to their respective homes, tourism as a whole took a huge hit, with a large number of hotels and touristic attractions having to shut down for months/years, due to the governments' anti-covid regulations. Several businesses went bankrupt, while others saw this global lockdown as an opportunity to embrace technology and distance-based activities. This led to an increase in the popularity of technology, especially in those which allow people to connect with each other without having to leave their houses. While the mandatory lockdown directly opposes the notion of tourism, which technically implies the movement of people to destinations outside their normal places of residence, there was still an attempt to counteract the negative effects brought by Covid. For instance, Airbnb, in response to the global pandemic, introduced "online experiences" on their website, which consist of an alternative to actual travelling, on which users experience cultural activities without having to leave their houses. Martina (2021) performed a study on which he determined whether Airbnb's new online experiences were a real long-term business opportunity or just a trend influenced by the pandemic. After using multiple methods of research, it was determined that these online experiences not only proved to be a valuable alternative to travelling for people who seek unique and authentic experiences, but will also be a precious complement to travelling, especially to try the cultural activities of a specific destination before actually travelling there.

2.2. WEB SCRAPING

With data science being, nowadays, one of the most prolific scientific fields in the sector of technology, all techniques that work with data are bound to be extremely useful, and web scraping is no exception. In a world where information is everywhere, having a way of collecting that information in a cheap and efficient way is of incalculable value. With the increasing accessibility of the Internet, the amount of data created is directly proportional, giving web scraping a very important role when dealing with collection of data. In this subchapter, I will demonstrate the importance of data, leading to an explanation of big data and its role in predictive analytics, including in Portugal. After defining web scraping as a term and tackling some of the available software tools, I will mention some of the problems that might occur with web scraping, ending with a brief analysis on other works who also tackled web scraping within the tourism field.

2.2.1. THE IMPORTANCE OF DATA

We live in a time where personal data is becoming increasingly valuable, so companies use it in order to gain competitive advantage. According to Steve Todd (2015), by 2024 we will have a fully established Information Economy where data are critical to businesses. Joel Grus (2019) defends that over the next 10 years, we'll need billions and billions more data scientists than we currently have. Given the growing importance of data in a competitive world where all companies strive for information based on patterns, it is of the utmost priority to find ways to acquire data cheaply and efficiently. Thus, as an alternative to the direct purchase of data, web scraping appears, which is a data mining technique applied on websites.

> Data mining refers to extracting knowledge from a large amount of observational data, in order to discover the unsuspected relations and patterns hidden in the data, presenting the same in innovative, comprehensive and useful ways for users (Adeniyi, Wei, & Yongquan, 2016).

The main advantage of web scraping consists in the simple fact of it being a very cheap and effective process, thus reducing the need to acquire data from external sources. The Internet is full of unusable information, which just needs to be extracted and processed to be usable. With these methods, the companies can analyse the information obtained and try to predict what behaviour the tourists will have, thus adapting the tourism sector according to these observed patterns. It is very important to point out that predictive analytics is not a method of predicting the future, it is simply the calculation and interpretation of information already obtained, in order to try to determine which things are most likely to happen again. In the past, companies made their decisions based on past experiences and on the knowledge they already had, but with technological advances, they can now shape their sales models based on the information acquired through the extraction and analysis of data.

2.2.2. BIG DATA AND PREDICTIVE ANALYTICS

According to data from a survey by Domo (2017), 2.5 quintillion data are produced every second. This phenomenon, called Big Data, is one of the reasons for the growing popularity of these predictive techniques, together with the constant evolution of computers with increasingly high processing capacity.

According to Gartner (2012), Big Data is defined as assets of high-volume, high-velocity and/or high-variety information that demand cost-effective, innovative ways of processing information that enable enhanced insight, decision making, and process automation.

With the growing popularity of information and communication technologies, companies increasingly rely on them to develop marketing strategies for financial success. Today, we are constantly surrounded by sources of information, and companies know how to take advantage of this. It is no coincidence that Digital Marketing is considered one of the "jobs of the future". Over the years, technology will become more and more relevant, and, with that, data will be become even more valuable for companies.

In the end, it all boils down to the following: the customer decides whether or not to purchase a service; the supplier's success depends on the customer's decision; therefore, to achieve success, the company has to do everything in their power to positively influence the customer's decision. The best way to do this is to study all the customers they have had, and then, define the profiles of a satisfied buyer and a dissatisfied buyer. Based on these two profiles, the only thing left to do is to determine which key characteristics are present in easy profile, thus allowing the company to find ways to decrease the rate of unhappy clients. For instance, let's say a Portuguese company sells an app whose services are paid monthly: after studying customers, they discover that 80% of users who have not renewed their monthly subscription are of foreign origins. With this information, the company can decide to translate its app into several languages, thus leading to a gradual decrease in the dropout rate. The company needs to know what the customer wants, and from there it only has to adapt its sales models in accordance with the insights obtained from the customers studied.

The same notion applies to the program this project is a part of: with the data extraction carried out, information related to tourists and tourist destinations was acquired, which then can be used to trace and compare tourist profiles and determine what makes a tourist satisfied or dissatisfied. The conclusions drawn would be of high informative value and would contribute to shaping tourism resource management models.

2.2.3. PREDICTIVE ANALYTICS IN PORTUGUESE TOURISM

After researching on academic platforms, such as Google Scholar and ResearchGate, I was able to conclude that, in Portugal, there is a still a huge shortage of literature regarding web scraping and predictive analytics aimed at tourism.

> There is a need to promote the creation of specific literature on the relation between predictive analytics, the study of behaviour and its application to the tourism sector in Portugal as a decisive factor in the creation of innovation and competitiveness, through the support it provides to decision-making (Galinha, 2017).

Data are essential material for the study of tourist behaviour, and given the impact that tourism has on the financial sector in Portugal, I believe that companies should invest in these technological fields, as it is a promising industry with a lot of work to be done, in line with the increasing amount of information conceived daily.

It is expected that both the tourism sector and the data extraction and analysis sector will evolve a lot in Portugal, in the coming years, given the current explosion in the volume of data available. In this country, there are more and more centralised platforms for booking accommodation and tourism attractions, such as TripAdvisor, Booking.com and Airbnb, whose popularity has increased circumstantially in the recent years. The access to them has also been increasingly facilitated, with the evolution and standardisation of ICT. Combining this growth trend, which facilitates access to more data to be extracted and analysed, with the extreme scarcity of literature on web scraping in this country's tourism, I am confident that any investment made in this area will have very fruitful results for Portugal, in the years to come.

2.2.4. WEB SCRAPING DEFINITION

Zhao (2017) defines web scraping as a technique to extract data from the World Wide Web (WWW) and save it to a file system or database for later retrieval or analysis. Due to the large amount of data constantly generated on the Internet, web scraping is considered a very efficient technique to collect big data (Mooney et al., 2015). Zhao also states that scraping data from the Internet is divided into two main steps: acquiring web resources and then extraction information from the data we acquired:

> Specifically, a web scraping program starts by composing a HTTP request to acquire resources from a targeted website. This request can be formatted in either a URL containing a GET query or a piece of HTTP message containing a POST query. Once the request is successfully received and processed by the targeted website, the requested resource will be retrieved from the website and then sent back to the given web scraping program. (Zhao, 2017)

A web scraping program is composed by two essential modules: the first performs the HTTP request, which, on this project, was Selenium, while the second parses and extracts information from the HTML code. For this second part, I used Beautiful Soup. In my particular case, Selenium automatically opened a temporary browser to access the tourism platforms, while Beautiful Soup scraped the information on those platforms, through their HTML source codes.

Data extracted can then be exported to a file or a database, for further analysis. In this case, the data obtained were transformed into Excel datasets.

2.2.5. WEB SCRAPING SOFTWARE TOOLS

While there is a fair amount of web scraping tools available, I only considered two possible options before beginning my project: using a web scraping software tool or building my own. Initially, through some research, I tried to come up with an easy way of extracting data from tourism platforms. After having read some literature regarding web scraping tools, I concluded that

Octoparse[1] would be a good choice. According to Matta et al. (2020), Octoparse is a tool that lets people who don't want to code use web scraping at its best use. Although coding was not a problem for me, I surely would not oppose to adopting a different methodology, if it saved me time and proved to be efficient. After experimenting with that program, I concluded Octoparse was a very good web scraping tool, confirming the statements by Almaqbali et al. (2019), which defend that Octoparse stands out due to its user-friendly interface, allied to a very powerful performance. However, there was a big problem with this program: its free version had some limitations which would not allow me to perform everything I wanted. Therefore, I decided to build my own web scraper, which had the potential to be better than a general web scraping software tool, as it was going to be custom made by myself, meaning I could focus only on the functionalities I really need, directing them specifically to the tasks at hand. Evidently, the programming language of choice was Python, which is not only one of the most popular programming languages nowadays, but also the best for data science, which is what this whole project revolves around.

2.2.6. PROBLEMS IN WEB SCRAPING

Despite the evident advantages associated with web scraping, there are a few drawbacks that might compromise the integrity of obtained information. One of the main problems consists of the fact that the quality of results depends on the quality and availability of collected data. Even if the person extracting the data is extremely skilled and uses professional methods, the results will always depend on the origin from which that data was extracted. Any inconsistent data must be identified and corrected before the analysis. According to Galinha (2017), it is estimated that 75% of the resources used in a project of data extraction are focused on the preparation of said data. Another problem, which is very relevant given this current project, is that fact that the automated extraction of data does not guarantee results that are 100% accurate, meaning there is always the possibility of needing human intervention. It is up to the researcher to evaluate which processes are more or less automatable and autonomous, in order to create a viable system. Sometimes, some random fluctuations occur, resulting in non-observable patterns in the extraction of data, making this one of the possible problems in web scraping. It is also important to know, being this project not only connected to informatics but also tourism, that having only informatics skills is not enough, as the intervention of someone qualified in the field of tourism is necessary, to evaluate the importance

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

of each extractable element and interpret the results obtained. Thus, the current project counts on the collaboration with several people working in the tourism field, allowing us to join forces and obtain results with a high degree of information and reliability.

Another problem associated with web scraping consists in the resulting legal and ethical issues, such as copyright implications (O'Reilly 2006). Since it is a very powerful technique, controversy around that topic is inevitable. Technically, there are no legal issues regarding the scraping of web data, as the data that is collected is free and available for everyone to use. While there may be certain terms of services in websites that condemn such practices, one can just state that he or she never officially agreed to those terms of service (Zhao, 2017). Therefore, instead of focusing on legal punishments, websites try to adopt systems to stop people from abusing these powerful scraping techniques. For instance, the websites check the HTML headers when they receive a request from the scraper, in order to determine if that request is coming from a normal user browsing their website or from an automated tool. To counter that counteract, I used a "decoy" header that tricks the target website into thinking I am a normal user accessing their website normally. My code can also include a pause time between each request, avoiding an overload of requests, which would be considered a denial-of-service attack (DDoS).

2.2.7. WEB SCRAPING IN TOURISM

Oliveira (2017) defends web scraping as a viable method for defining relevant tourism indicators, which allow the companies to obtain valuable insights, thus developing the sector and contributing to better business models. In his study, he extracted data from the TripAdvisor tourism platform using Import.io[2], obtaining user information regarding the Brazilian city of Minas Gerais. His methodology allowed him to conclude that the extracted data not only portrayed the behaviour of tourists on that zone, but also confirmed the veracity of that information, by comparison with what was observed in the tourism market related to that city.

Choong (2019) also performed a study on web scraping, creating an automated web scraping tool for Malaysia tourism. In his project, he creates his own web scraper tool, which, just like mine, uses Selenium and Beautiful Soup, while also being supported by PySimpleGUI to build a Graphical User Interface (GUI). With his work, he concluded that there is too much public tourism data

[2] https://www.import.io/

available on the Internet, which is essentially being wasted, instead of being used for its potential value.

With their work regarding "Development of online travel Web scraping for tourism statistics in Indonesia", Adhinugroho et al. (2020) conduct a web scraping methodology on Indonesian online travel agent websites, Agode and Pegipegi. Their method also included Python, using Beautiful Soup and the Requests modules to extract the chosen data. They displayed a fair amount of graphs and tables, which led to the conclusion that the results obtained from their extractions coincided with the official statistics, further emphasizing the idea that web scraping is a very useful and powerful technique that can lead to valuable results.

2.3. DATA VISUALISATION

Ajibade and Adediran (2016) define data visualisation as the act of data presentation in a graphical or pictorial layout. According to them, data visualisation allows one to visually observe analytics and easily understand complex ideas.

As stated by Khedikar (2021), the visual presentation of data is understood easily by the human brain and it allows us to extract useful insights. Thus, we can easily identify patterns and outliers in large datasets. Due to the large amount of data obtained from web scraping, it is necessary to use powerful tools that can receive and represent data in extensive ways. Initially, I experimented with Tableau, as it is a very good tool for visualising and analysing extensive volumes of data. Despite its quality, I discovered that Python includes a wide array of libraries that can perform data visualisation techniques very effectively, thus making it my choice.

One of the biggest challenges on this project was knowing what types of graphs to use and which tourism indicators I should cross visually. To cite Ajibade and Adediran:

> You must, first of all, understand the data want to visualize with its size and cardinality (the uniqueness of data value contained in a column). You also should determine what you are trying to visualize and the type of information to be communicated; you are also supposed to have a good knowledge of your audience and understand how it processes the vital information, and, lastly, you should make use of a visual that carried the information in the best and easiest way to your audience or end users. (Ajibade and Adediran, 2016)

Thanks to Plotly and Dash libraries, I was able to not only turn my extracted data into graphs, but also add interactivity to those graphs, which allowed me to take this whole notion of web scraping and data analysis one step further, obtaining even more information and details from my data visualisations.

3. METHODOLOGY

In this section, I will list and describe the tourism platforms selected, the tourism indicators that I considered relevant for collection, and the software tools I used to perform the extractions.

3.1. TOURISM PLATFORMS

Given the available tourism platforms available in Portugal and Galicia, these were the ones chosen, due to their popularity and amount of data accessible for extraction:

- Booking.com
- Tripadvisor
- Airbnb

3.1.1. BOOKING.COM

Booking.com is an international website which allows the user to book accommodation for vacation or travelling. It was founded in 1996 by Geert-Jan Bruinsma, in the Netherlands.

3.1.2. TRIPADVISOR

Tripadvisor is a travelling website that includes information and reviews on several fields related to tourism, such as accommodation, transportation, activities, and restaurants. It was founded in the United States in 2000, by Langley Steinert, Stephen Kaufer, and others.

3.1.3. AIRBNB

Airbnb is an online marketplace for lodging, on which users can either announce or discover and book accommodations. They do not own any of the listed properties, they simply provide the means for people to rent those properties, while receiving a commission from each booking.

3.2. TOURISM INDICATORS

Within the context of this project, the following zones were chosen as the most relevant to be observed:

- Galicia
- North of Portugal
- Center of Portugal

In collaboration with IPDT (Instituto de Planeamento e Desenvolvimento do Turismo), the following indicators were defined as relevant for extraction:

- Accommodation name
- Zone
- Price
- Rating
- Number of comments
- Spots remaining at the presented price
- URL

The check-in date settings were also defined, allowing for comparison of extracted results:

- In 1 day
- In 30 days

- In 60 days

3.3. SOFTWARE TOOLS USED

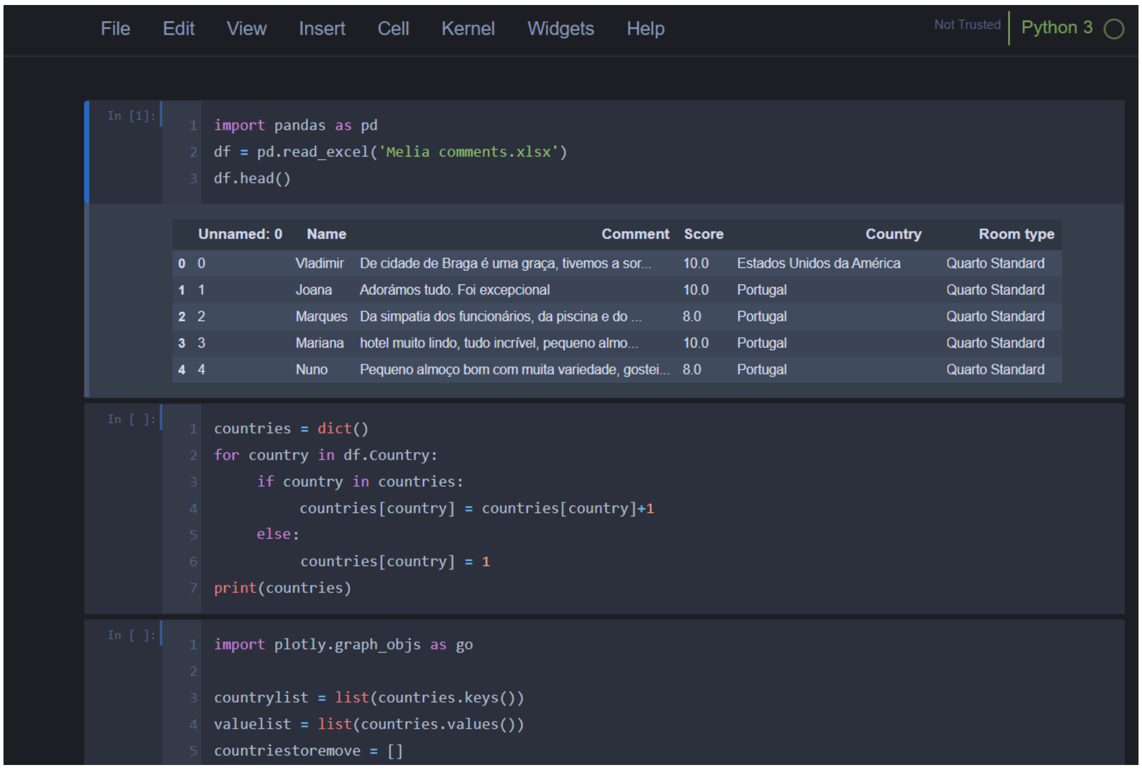
3.3.1. PYTHON

Python is a high-level object-oriented programming language, created by Guido van Rossum in 1991. It is considered the one of best programming languages to automate tasks and conduct data analysis. Python is extremely popular nowadays, due to its versatility and user-friendliness. Given this easiness, Python is used by many non-programmers to perform daily tasks that would otherwise require manual work. It is a very powerful tool in the fields of data science, web development and automation. Almost every step of the method described in this document relies on Python.

3.3.2. JUPYTER NOTEBOOK

While programming, it is essential to have a good IDE (integrated development environment). Having tried many, I noticed that my code would require a lot of testing of single features within the program. Therefore, it was essential to use a Notebook type of IDE, which, as the name indicates, functions as a sort of notebook on which a sequential "story" will be written. As with every story, it is divided in parts, on which they follow a sequential order. With a notebook, we have the option of dividing our code in blocks of code and running them separately. If previous blocks are run before our current code, then the laws applied by those previous blocks will be applied to our current code. Unlike normal IDEs, we do not need to run the whole code every time we modify it. For instance, with this program, I started with the first block that included the codes for opening the website and extracting all the data available. Then, to focus on a specific element that I want to extract, I could just create another block and write my code for that specific element there. If I ran that block, I did not need to re-run the whole program, it would just run that specific block and I would get the desired result. This makes a notebook IDE the perfect tool for this type of project on which there are a lot of trial and error attempts.

After some research, I concluded that Jupyter Notebook[3] was the best IDE for my task. After installing it on my computer, I just needed to run it, and it would automatically open on my browser. Although it runs on my browser, it works locally, requiring no internet access.



In [1]:
```python
import pandas as pd
df = pd.read_excel('Melia comments.xlsx')
df.head()
```

	Unnamed: 0	Name	Comment	Score	Country	Room type
0	0	Vladimir	De cidade de Braga é uma graça, tivemos a sor...	10.0	Estados Unidos da América	Quarto Standard
1	1	Joana	Adorámos tudo. Foi excepcional	10.0	Portugal	Quarto Standard
2	2	Marques	Da simpatia dos funcionários, da piscina e do ...	8.0	Portugal	Quarto Standard
3	3	Mariana	hotel muito lindo, tudo incrível, pequeno almo...	10.0	Portugal	Quarto Standard
4	4	Nuno	Pequeno almoço bom com muita variedade, gostei...	8.0	Portugal	Quarto Standard

In [ ]:
```python
countries = dict()
for country in df.Country:
    if country in countries:
        countries[country] = countries[country]+1
    else:
        countries[country] = 1
print(countries)
```

In [ ]:
```python
import plotly.graph_objs as go

countrylist = list(countries.keys())
valuelist = list(countries.values())
countriestoremove = []
```

Figure 1 - Jupyter Notebook interface.

4. IMPLEMENTATION

4.1. DATA EXTRACTION

There are several tools available which make web scraping a much easier task. While they are very user-friendly and intuitive, the user is restricted to the capabilities of each individual program. This makes dedicated web scraping software very good for casual use. However, it is not the best when you have very specific goals, which was my case. I wanted very specific extractions from three different online tourism platforms, and I wanted those extractions to occur automatically, while also wanting that whole process to be done in an easy and intuitive way, with an also intuitive user interface that was solely focused on the goals of this project. For me to achieve this, I knew I could not count on external web scraping tools to perform these tasks. I had to create a web scraper from scratch, using programming language. That way, everything works the way I want, with every

[3] https://jupyter.org/

little “cogwheel” rotating in the directions I want. I did not need a web scraper with countless features that could perform several different tasks. I needed a web scraper that went straight to the point and extracted everything I instructed it to, converting all the extracted data into datasets. My web scraper was not written in a way that it could extract anything from any website. My web scraper was fully and solely dedicated to the subject of this project (tourism in Galicia and North/Center of Portugal), it was coded in a way that it already knew what it had to extract, as it was built based on the source codes of Booking, TripAdvisor and Airbnb pages. Regular web scraping software programs require you to insert the URL from which you want to extract data. My program does not need that because it already has pre-set functions that extract data from the platforms previously selected during this project. When performing the extractions, I did not have to select the URLs and elements I wanted to extract, as all of that had already been specified in the code I had written. The only interaction between the user and the program is choosing between the three tourism platforms and choosing whether he wants to extract the hotels’ data to coincide with check-in dates of “In 1 day”, “In 30 days” or “In 60 days” from today (the day of extraction).

The main goal of this project was to design a system that collects data from the tourism platforms in Galicia and North/Center of Portugal, which subsequently gets analysed in order to obtain valuable insights. My program did that almost automatically.

4.1.1. LIBRARIES AND PACKAGES

In order to create a web scraper fully capable of extracting and managing data, I required several Python libraries and packages, which aided me in different situations.

Beautiful Soup

Beautiful Soup[4] is a Python library for extracting data from HTML and XML files. It does not support JavaScript, which is included in the pages I wanted to extract data from, so I needed another library to help Beautiful Soup extract data.

[4] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Selenium

Selenium WebDriver[5] runs a browser natively, just like a normal use would, automatically. In this context, it automatically opens a temporary browser which opens the tourism platforms, allowing Beautiful Soup to extract data from the HTML source code obtained by Selenium. While BS4 can't interpret JavaScript, Selenium can, which is why this was necessary for the whole process.

csv

A Python module that allowed me to instruct my program to read and write the extracted data into a dataset in Excel format.

Time

This module can perform several time-related functions. I used it to force my code to wait a few seconds between opening the webpages and actually extracting the data, in order to avoid losing data. This concept will be explained in a later section.

Datetime

This module allows me to manipulate dates and times. In this case, I used it to allow the program to distinguish different dates, in relation to the date on which the extraction is performed. Before implementing these functions, my program would open the URL with listings from the current day. If I ran the same program a few days later, it would still open the URL from three days earlier (which would result in an error). With this module, the program always functions at the date on which it is being used. It allowed me to define the date for the search results, in relation to the date on which the extraction was performed. For instance, my program can search results corresponding to 60 days after the current day. If I run that program five days later, that function will present results for 60 days after the day on which I am performing the extraction, instead of 55.

[5] https://www.selenium.dev/documentation/webdriver/

Pandas

Pandas is a powerful Python library used to analyse data. In this case, I only used two specific functions that allowed me to transform the extracted data into a data frame, which then got transformed into an Excel file.

re (Regular expression operations)

This module allows me to use regular expressions on my code. For this particular case, regular expressions helped me obtain cleaner data during the extractions. Sometimes, some of the elements extracted came with extra text that was not necessary, and with this module I was able to only receive the information I want.

math (Mathematical functions)

The math module allows access to mathematical functions. I've used some of these functions to assist me with the lines of code related to pagination, on which I tried to discover the number of the last page of results (explained on a later section).

PySimpleGUI

PySimpleGUI[6] is a Python package that allowed me to create a GUI (Graphical User Interface) that runs my code, making the whole program intuitive and usable by any person. Before adopting this package, all the actions performed relied on coded lines, which was difficult to follow, from an outsider's perspective. With a visual interface, not only is my program more aesthetically pleasing, but it also facilitates the whole process of debugging my code, as I'm no longer only looking at endless lines of code.

[6] https://www.pysimplegui.org/

4.1.2. WEB SCRAPING METHOD

There were two possible ways on which I could have approached this situation. I could have either extracted data directly from the websites, using Python functions that access the website, read and extract all the information available, or I could have used an API (application programming interface), created by the companies to allow external third-party developers to access their data and functionalities. The API is considered the recommended method, as it is created by the tourism platforms with a defined set of rules, transmitting only relevant data, whereas extracting data directly makes it so we receive everything, including useless details that will need to be deleted later. Unfortunately, it is often difficult to receive access to their API keys, and this was the case, so I was forced to use the other method of data extraction.

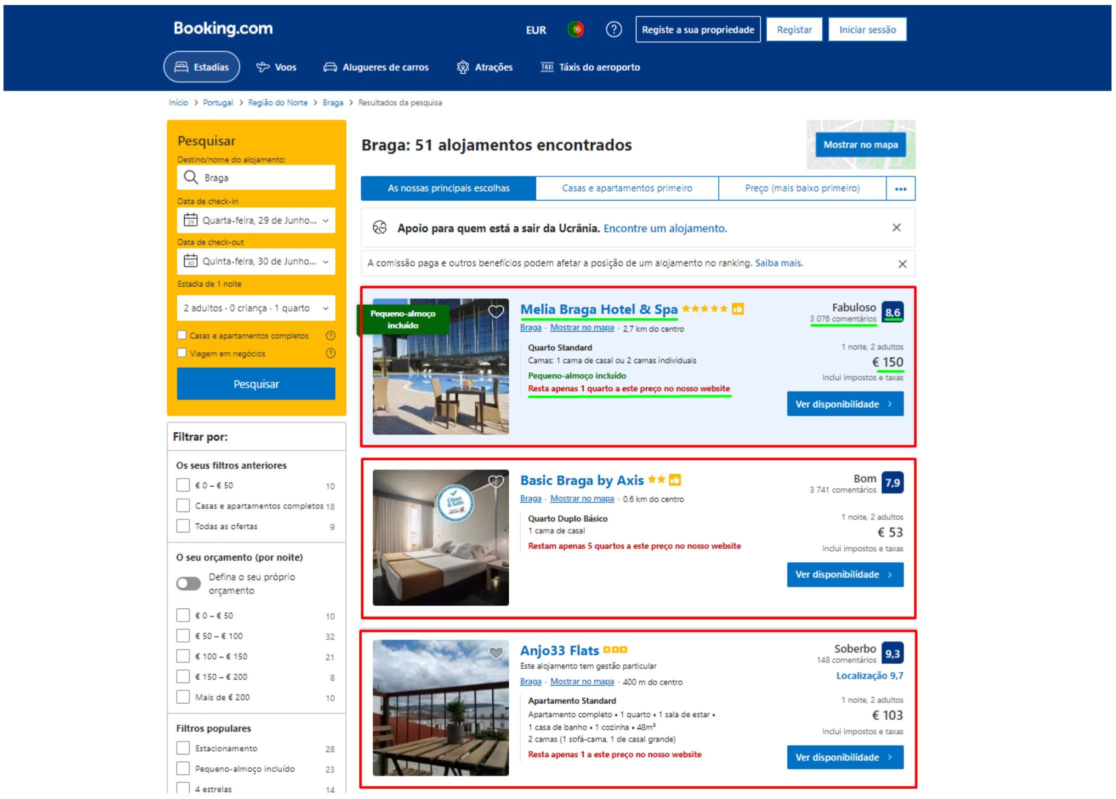
As I had to extract everything directly from the website, I had to come up with a method on which it was easy to define from which zone and on which date I wanted my results to be from. In this case, my web scraping method was based on the target URL link. Since the respective URLs of each of the three tourism platforms contain zone and check-in/check-out parameters, this method was extremely viable. Besides that, they also contain information regarding the pagination, which is very important, as while my program needs to extract data from every single page, it also needs to be able to recognize when it reaches the last page, otherwise it will be stuck in an infinite loop.

So basically, I had to come up with a code that performed the following tasks:

1. Receive and modify the target URL with the search parameters I choose
2. Access the URL that was modified previously and extract the data from that page
3. Jump to the next page, extract the data, and repeat for the subsequent pages
4. Stop the loop once it reaches the final page
5. Compile all the data extracted into a data frame, which is then transformed into a dataset in Excel format.

4.1.2.1. FIRST EXTRACTION: GENERAL RESULTS

To portray this method, I used the Booking.com part of the program as an example, although the same method was used for all three tourism platforms. All results obtained, including prices, correspond to a one-day booking, for a group of two adults.


Booking.com EUR Registe a sua propriedade Registar Iniciar sessão

Estadias Voos Alugueres de carros Atrações Táxis do aeroporto

Início > Portugal > Região do Norte > Braga > Resultados da pesquisa

Pesquisar
Destino/nome do alojamento: Braga
Data de check-in: Quarta-feira, 29 de Junho...
Data de check-out: Quinta-feira, 30 de Junho...
Estadia de 1 noite
2 adultos · 0 criança · 1 quarto
Casas e apartamentos completos
Viagem em negócios
Pesquisar

Filtrar por:

Os seus filtros anteriores
€ 0 – € 50 10
Casas e apartamentos completos 18
Todas as ofertas 9

O seu orçamento (por noite)
Defina o seu próprio orçamento
€ 0 – € 50 10
€ 50 – € 100 32
€ 100 – € 150 21
€ 150 – € 200 8
Mais de € 200 10

Filtros populares
Estacionamento 28
Pequeno-almoço incluído 23
4 estrelas 14

Braga: 51 alojamentos encontrados

Mostrar no mapa

As nossas principais escolhas | Casas e apartamentos primeiro | Preço (mais baixo primeiro) | •••

Apoio para quem está a sair da Ucrânia. Encontre um alojamento.

A comissão paga e outros benefícios podem afetar a posição de um alojamento no ranking. Saiba mais.

Pequeno-almoço incluído
Melia Braga Hotel & Spa ★★★★★
Braga · Mostrar no mapa · 2,7 km do centro
Quarto Standard
Camas: 1 cama de casal ou 2 camas individuais
Pequeno-almoço incluído
Resta apenas 1 quarto a este preço no nosso website
Fabuloso 8,6
3 076 comentários
1 noite, 2 adultos
€ 150
Inclui impostos e taxas
Ver disponibilidade >

Basic Braga by Axis ★★
Braga · Mostrar no mapa · 0,6 km do centro
Quarto Duplo Básico
1 cama de casal
Restam apenas 5 quartos a este preço no nosso website
Bom 7,9
3 741 comentários
1 noite, 2 adultos
€ 53
Inclui impostos e taxas
Ver disponibilidade >

Anjo33 Flats
Este alojamento tem gestão particular
Braga · Mostrar no mapa · 400 m do centro
Apartamento Standard
Apartamento completo • 1 quarto • 1 sala de estar • 1 casa de banho • 1 cozinha • 48m²
2 camas (1 sofá-cama, 1 de casal grande)
Resta apenas 1 a este preço no nosso website
Soberbo 9,3
148 comentários
Localização 9,7
1 noite, 2 adultos
€ 103
Inclui impostos e taxas
Ver disponibilidade >

Figure 2 - Booking.com general results for Braga city.

Before beginning, I had to import all the necessary libraries and packages, which would be used to scrape data, create data frames, read and create Excel files, find and replace text, calculate and define dates, and create a visual interface. I also added an empty list called hotelslistBooking, placed in the beginning of the code, as it should be outside our main function. This list will contain all the data extracted, which will then be converted into a data frame, and, subsequently, into a dataset.

```
from bs4 import BeautifulSoup
import requests
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from webdriver_manager.chrome import ChromeDriverManager
from csv import writer
import time
import pandas as pd
import PySimpleGUI as sg
import datetime
import re
import math

hotelslistBooking = []
```

Listing 1 - Importing necessary libraries and creating my list.

When scraping data, my program is essentially sending a request to the target website that is not only saying “I want to extract your data”, but also revealing information about who is trying to extract that information and what type of information is being extracted. However, websites do not want people to use those requests to extract data from them automatically. If the target website knows that my requests are coming from an automated program purely intended to scrape their data, it will try to blacklist my address. Therefore, it is necessary to use a “header”, which contains information regarding the browser being used, fooling the target website into thinking these requests are coming from a normal user accessing their website through a browser.

```
headers = {'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82
Safari/537.36'}
```

Listing 2 - Headers section of the code, which avoids me being blacklisted by the target website.

To begin, I defined a function, which I called getHotelsBooking, that will execute a large amount of tasks: it will receive the modifiable URL, access it using Beautiful Soup and Selenium, then search for the elements I want (identified by their CSS classes), extract them, add them to a dictionary called hotelbooking, which then will be added to a list which I earlier named HotelsListBooking. Then, with the assistance of a “for loop”, repeat the same process for the following pages, constantly adding the extracted data into the growing HotelsListBooking list. After reaching the final page, the concluded HotelsListBooking is then turned into a dataframe, which is converted into an Excel dataset.

Here is an example of a standard Booking.com URL, which I must modify:

```
https://www.booking.com/searchresults.pt-pt.html?label=gen173nr-
1DCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXyAEM2AED6AEBiAIBqAIDuAK05-
GUBsACAdICJDA4ZmNhMjM0LWQyNzctNDYzMC04MGU2LTEyMzJlMTg5Mjc4MdgCBOACAQ
&sid=06ca9cc81011f571efff0bf1217d614b&sb=1&sb_lp=1&src=index&src_ele
m=sb&error_url=https%3A%2F%2Fwww.booking.com%2Findex.pt-
pt.html%3Flabel%3Dgen173nr-
1DCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXyAEM2AED6AEBiAIBqAIDuAK05-
GUBsACAdICJDA4ZmNhMjM0LWQyNzctNDYzMC04MGU2LTEyMzJlMTg5Mjc4MdgCBOACAQ
%26sid%3D06ca9cc81011f571efff0bf1217d614b%26sb_price_type%3Dtotal%26
%26&ss=Braga%2C+Regi%C3%A3o+do+Norte%2C+Portugal&is_ski_area=&checki
n_year=2022&checkin_month=6&checkin_monthday=2&checkout_year=2022&ch
eckout_month=6&checkout_monthday=3&group_adults=2&group_children=0&n
o_rooms=1&b_h4u_keep_filters=&from_sf=1&ss_raw=braga&ac_position=0&a
c_langcode=pt&ac_click_type=b&dest_id=-
2160205&dest_type=city&place_id_lat=41.551094&place_id_lon=-
8.42827&search_pageview_id=958d3b1a46680009&search_selected=true&sea
rch_pageview_id=958d3b1a46680009&ac_suggestion_list_length=5&ac_sugg
estion_theme_list_length=0&offset=0
```

Listing 3 - Standard Booking.com URL.

Although this is a very long URL, amid all these random characters I can find relevant information, such as the zone, the check-in/check-out dates and the page number, highlighted in red. If I replace those relevant elements (highlighted in bold red) with variables, my code will essentially open the URL with the zone and dates I choose, as long as I teach my program to do so beforehand. I will name my variables page, zone, and year1, month1, day1 for the check-in date, and year2, month2 and day2 for the check-out date. For instance, that portion of the URL will become the following:

```
%26&ss={zone}&is_ski_area=&checkin_year={year1}&checkin_month={month
1}&checkin_monthday={day1}&checkout_year={year2}&checkout_month={mon
th2}&checkout_monthday={day2}&group_adults=2&group_children=0&no_roo
ms=1&b_h4u_keep_filters=&from_sf=1&ss_raw=braga&ac_position=0&ac_lan
gcode=pt&ac_click_type=b&dest_id={zone}&dest_type=city
```

Listing 4 - Portion of the URL that contains the date and zone parameters.

Thus, I can start defining my function, which will receive those parameters, turning my URL into an accessible one.

```
def getHotelsBooking(page, zone, day1, month1, year1, day2, month2,
year2):
url = f'https://www.booking.com/searchresults.pt-
pt.html?label=gen173nr-
1DCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXyAEM2AED6AEBiAIBqAIDuAK05-
GUBsACAdICJDA4ZmNhMjM0LWQyNzctNDYzMC04MGU2LTEyMzJlMTg5Mjc4MdgCBOACAQ
&sid=06ca9cc81011f571efff0bf1217d614b&sb=1&sb_lp=1&src=index&src_ele
m=sb&error_url=https%3A%2F%2Fwww.booking.com%2Findex.pt-
pt.html%3Flabel%3Dgen173nr-
1DCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXyAEM2AED6AEBiAIBqAIDuAK05-
GUBsACAdICJDA4ZmNhMjM0LWQyNzctNDYzMC04MGU2LTEyMzJlMTg5Mjc4MdgCBOACAQ
%26sid%3D06ca9cc81011f571efff0bf1217d614b%26sb_price_type%3Dtotal%26
%26&ss=Braga%2C+Regi%C3%A3o+do+Norte%2C+Portugal&is_ski_area=&checki
n_year={year1}&checkin_month={month1}&checkin_monthday={day1}&checko
ut_year={year2}&checkout_month={month2}&checkout_monthday={day3}&gro
up_adults=2&group_children=0&no_rooms=1&b_h4u_keep_filters=&from_sf=
1&ss_raw=braga&ac_position=0&ac_langcode=pt&ac_click_type=b&dest_id=
{zone}&dest_type=city&place_id_lat=41.551094&place_id_lon=-
8.42827&search_pageview_id=958d3b1a46680009&search_selected=true&sea
rch_pageview_id=958d3b1a46680009&ac_suggestion_list_length=5&ac_sugg
estion_theme_list_length=0&offset={page}'
```

Listing 5 - Main function getHotelsBooking, that modifies the parameters on the URL.

Now that I have a function that accesses our URL based on the parameters it receives, I need to write a code that adds those parameters to the URL.

Let's say I want my program to give me results from Braga city, with a check-in date of today (not today as in the day I am writing this, but today as in the day the search is performed, which can be any day), and a check-out on the following day. If today is June 3rd, I could just say that "day1 = 3" and "month1 = 6", and when I run my function, the URL will give me today's results. However, this is a terrible solution, because if I come back tomorrow and run the program again, it will not give me tomorrow's results, it will give me June 3rd's results, as day1 and month1 are defined as constants. This means I must define a variable for today's date, and for tomorrow's date, that is always updated. I can easily achieve this by using the Datetime module, mentioned previously. Let's begin by defining a variable that always gives me the current date and hour:

```
datetoday = datetime.datetime.now()
```

Listing 6 - Function to obtain current date and hour.

As I am writing this, it is now 06:12 of June 3rd. Printing datetoday will give me the following output:

```
2022-06-03 06:12:12.236867
```

Listing 7 - datetoday output.

However, by observing the link, I can see that I do not need the entire date and time, as the parameters seen on the URL are in a number format, so I need to define variables for today's day, month and year:

```
daytoday = datetoday.strftime("%d")
monthtoday= datetoday.strftime("%m")
yeartoday = datetoday.strftime("%Y")
```

Listing 8 - Defining variables for today's day, month and year.

Printing those 3 variables will give me the following output:

```
03
06
2022
```

Listing 9 - daytoday, monthtoday and yeartoday's outputs.

I also need to use the same method for tomorrow's day, which will be my check-out date. I can re-use datetoday and use "datetime.timedelta", which will allow me to jump forward 1 day.

```
datetomorrow = datetime.date.today() + datetime.timedelta(days=1)
daytomorrow = datetomorrow.strftime("%d")
monthtomorrow = datetomorrow.strftime("%m")
yeartomorrow = datetomorrow.strftime("%Y")
```

Listing 10 – Advancing 1 day on our previous functions.

Printing them gives me this output:

```
2022-06-04 06:12:12.236867
04
07
2022
```

Listing 11 - datetomorrow, daytomorrow, monthtomorrow and yeartomorrow's outputs.

Now that my time variables are ready, I just have to define the parameter for the zone I want. In this case, for demonstration purposes, I will want Braga city. If I go to Booking.com and search for hotels in Braga, I can see the portion of the link that includes the zone, which is **"**-2160205**".**

As I now know almost every parameter that will be added to the URL, I can begin by defining the parameters of my "getHotelsBooking(page, zone, day1, month1, year1, day2, month2, year2)" function.

```
zone = -2160205
day1 = daytoday
month1 = monthtoday
year1 = yeartoday
day2 = daytomorrow
month2 = monthtomorrow
year2 = yeartomorrow
```

Listing 12 - Zone and time parameters defined.

However, there is still one variable remaining, which is "page". Without "page", my program would open the link, which would be on the first page of results, but then it would not continue its task on the following pages. Therefore, I need the page variable and I need to instruct my code to start at the first page, and then jump to the next one, repeat the same process, and then jump to the next page, and so on. Besides that, it must also know when it has reached the final page, so it stops the program. I can do that with a "for loop", but before jumping into that subject, we need to understand the process of pagination.

Usually, websites include the page number. If I analyse Booking.com's URL, I can observe that the link includes information on the page, but instead of containing the page's exact number, it does so in leaps of 25. So, the URL on the first page includes a 0, the second page's includes a 25, the third page's includes a 50, and so on. Having this information, I now know that I just have to instruct my program to start at "page = 0", extract the data and then jump to the second page, which would be "page = 25", and then to the third page, which would be "page = 50". Essentially, I need my program to be executed in a loop, in leaps of 25 between each loop, until the last page. With the first part of the pagination process complete, I now need to make it so my program is able to tell when it has reached the final page. There are two main methods of doing this:

1. Search the HTML element that includes the total number of listings found, extract that piece of text, transform it into an "int", which allows me to divide it by 25, giving me

the approximate number of total pages, which would also have to be rounded up afterwards.

2. Use BeautifulSoup's soup.find to look for the HTML element that represents the "Next page" arrow, which takes me to the following page when clicked. This element is always present when I am not in the last page, but always missing when I am on the last page, as it is impossible to jump to the next page when there are no pages remaining.

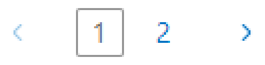


Figure 3 - Page buttons at the first page.

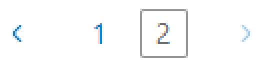
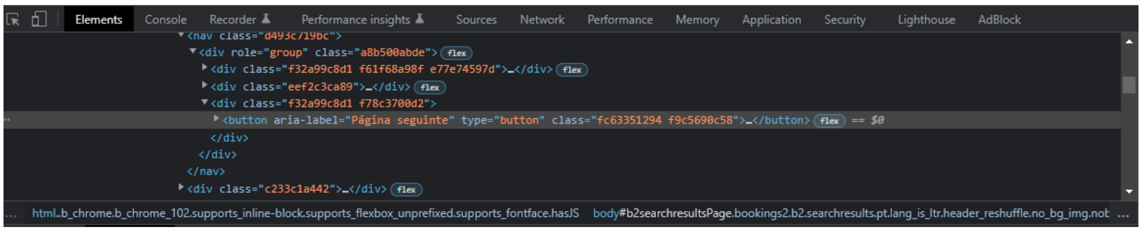


Figure 4 - Page buttons at the first page.

Since the latter is a much simpler and faster method, I chose that one. However, later I will also perform the first method as an alternative, in case the second one stops working due to updates on the source code.

In order to find that HTML element, I need to open the page on my browser, right-click the next page arrow button and select "Inspect".



```
▼<nav class="d493c719bc">
  ▼<div role="group" class="a8b500abde"> flex
    ▶<div class="f32a99c8d1 f61f68a98f e77e74597d">…</div> flex
    ▶<div class="eef2c3ca89">…</div> flex
    ▼<div class="f32a99c8d1 f78c3700d2">
      ▶<button aria-label="Página seguinte" type="button" class="fc63351294 f9c5690c58">…</button> flex == $0
     </div>
   </div>
 </nav>
▶<div class="c233c1a442">…</div> flex
… html..b_chrome.b_chrome_102.supports_inline-block.supports_flexbox_unprefixed.supports_fontface.hasJS   body#b2searchresultsPage.bookings2.b2.searchresults.pt.lang_is_ltr.header_reshuffle.no_bg_img.not …
```

Figure 5 - Inspecting the "next page" element to discover its HTML tag and CSS class.

This way, I can see that the arrow button's type is "button", while its CSS class is "fc63351294 f9c5690c58". I can use "soup.find" to extract the source code corresponding to that arrow button. I will name this variable NotLastPage, and since it will be included inside getHotelsBooking, it will look like the following:

```
getHotelsBooking.NotLastPage = soup.find('button', {'class':
'fc63351294 f9c5690c58'})
```

Listing 13 - Function to find the arrow button.

If I print getHotelsBooking.NotLastPage, its output will be:

```
<button aria-label="Página anterior" class="fc63351294 f9c5690c58"
type="button"><span aria-hidden="true" class="b6dc9a9e69
e25355d3ee"><svg data-rtl-flip="true" viewbox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path d="M14.55 18a.74.74 0 0 1-.53-.22l-5-5A1.08 1.08 0
0 1 8.7 12a1.1 1.1 0 0 1 .3-.78l5-5a.75.75 0 0 1 1.06 0 .74.74 0 0 1
0 1.06L10.36 12l4.72 4.72a.74.74 0 0 1 0 1.06.73.73 0 0 1-.53.22zm-
4.47-5.72zm0-.57z"></path></svg></span></button>
```

Listing 14 - Printing my function that gives me the arrow's element.

Now that I can extract this element, I will need to instruct my program to check if that element exists, which will tell me if I am on the last page. Before doing that, I need to setup my "for loop" statement, which will run my program in a loop, stopping only when it reaches the final page (otherwise it will run forever).

This "for loop" will essentially run my main function, with the "page" number varying in iteration of the loop. So, in this case, getHotelsBooking will be run in a way so that day1, month1, year1, day2, month2 and year2 will always be constant during our loop. The only thing that will change between loops is the page, which I will call "x". When creating my "for loop", I will have 3 parameters:

```
for x in range(a, b, c):
```

Listing 15 - Example of a "for loop".

"a" is the starting number of x on my loop. In this case, on the first page, the "page" number is 0, so "a" will be 0.

"b" is the ending number of x on my loop. In this case, it is not important because I will force it to stop when it doesn't find the arrow element. Nevertheless, I will add 1000, which is the maximum number of results that Booking.com displays.

"c" defines how big is the "leap" between each loop. Since on the first page, "zone" is 0 and on the second page, "zone" is 25, I know that between each loop, x will have to increase by 25. Therefore, my c = 25.

This gives me:

```
for x in range(0, 1000, 25):
      getHotelsBooking(x, zone, day1, month1, year1, day2, month2,
year2)
      if hasattr(getHotelsBooking.NotLastPage, '__len__') == True:
            continue
      else:
            break
```

Listing 16 - "For loop" that cycles through every page until it reaches the last one.

Here's what I am telling my program: run getHotelsBooking for every instance of x (which corresponds to the page), starting with x = 0 and jumping in intervals of 25. In every loop, if the next page button exists, the loop should continue normally (it checks if it has the attribute "__len__", which only does not occur if there is no text). If the button does not exist, it does not have the "__len__" attribute, because no text was extracted, since there was no element to extract it from. If that is the case, then the loop should break, because it means it has reached the last page.

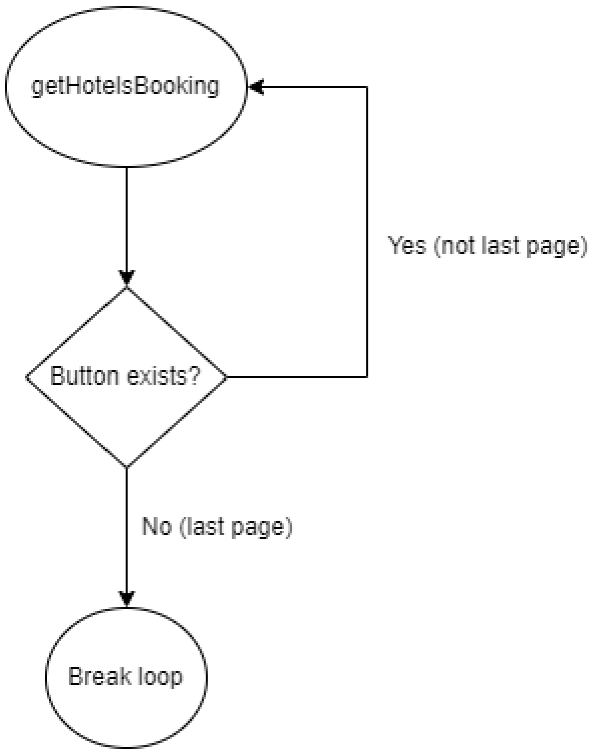
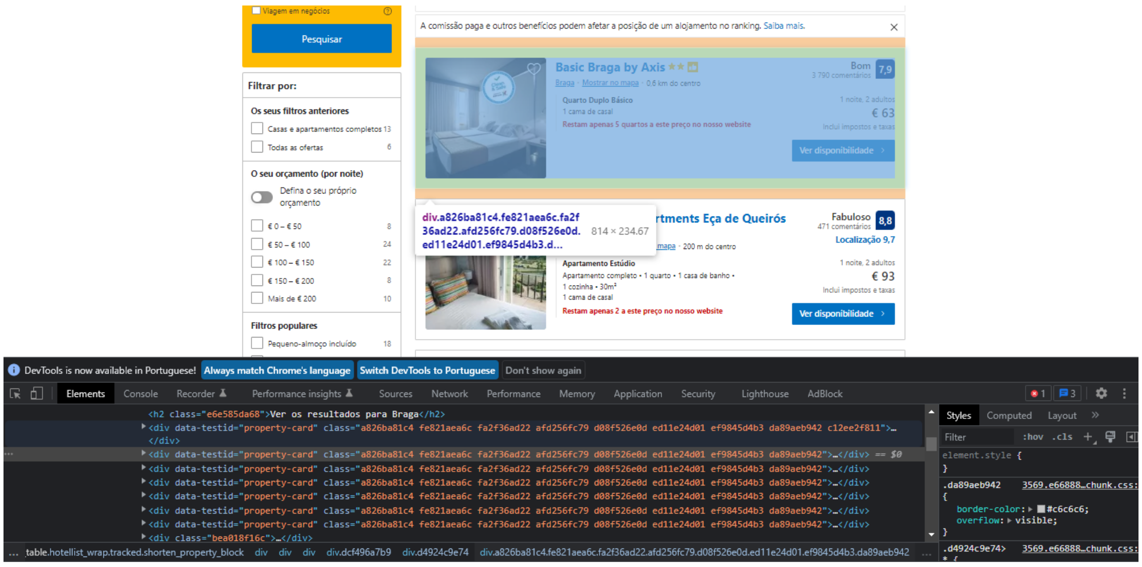


Figure 6 - Main function's flowchart (getHotelsBooking).

With the loop functioning perfectly, I can now move on to the extraction part. My function can open the URL and perform its tasks in each page, until it reaches the last page. Now I must define what those tasks are, inside each page. Therefore, I need to select the elements I want to extract, using the same method I previously used to locate the "next page" arrow button.

Firstly, I need to examine the first page to understand how the HTML and CSS elements are structured. By inspecting the elements, I can observe that there are multiple blocks with the same class, each corresponding to each listing.



```
<h2 class="e6e585da68">Ver os resultados para Braga</h2>
▸<div data-testid="property-card" class="a826ba81c4 fe821aea6c fa2f36ad22 afd256fc79 d08f526e0d ed11e24d01 ef9845d4b3 da89aeb942 c12ee2f811">…
</div>
▸<div data-testid="property-card" class="a826ba81c4 fe821aea6c fa2f36ad22 afd256fc79 d08f526e0d ed11e24d01 ef9845d4b3 da89aeb942">…</div> == $0
▸<div data-testid="property-card" class="a826ba81c4 fe821aea6c fa2f36ad22 afd256fc79 d08f526e0d ed11e24d01 ef9845d4b3 da89aeb942">…</div>
▸<div data-testid="property-card" class="a826ba81c4 fe821aea6c fa2f36ad22 afd256fc79 d08f526e0d ed11e24d01 ef9845d4b3 da89aeb942">…</div>
▸<div data-testid="property-card" class="a826ba81c4 fe821aea6c fa2f36ad22 afd256fc79 d08f526e0d ed11e24d01 ef9845d4b3 da89aeb942">…</div>
▸<div data-testid="property-card" class="a826ba81c4 fe821aea6c fa2f36ad22 afd256fc79 d08f526e0d ed11e24d01 ef9845d4b3 da89aeb942">…</div>
▸<div data-testid="property-card" class="a826ba81c4 fe821aea6c fa2f36ad22 afd256fc79 d08f526e0d ed11e24d01 ef9845d4b3 da89aeb942">…</div>
▸<div class="bea018f16c">…</div>
```

DevTools is now available in Portuguese!
Always match Chrome's language
Switch DevTools to Portuguese
Don't show again
Elements
Console
Recorder
Performance insights
Sources
Network
Performance
Memory
Application
Security
Lighthouse
AdBlock
div.a826ba81c4.fe821aea6c.fa2f36ad22.afd256fc79.d08f526e0d.ed11e24d01.ef9845d4b3.da89aeb942
Basic Braga by Axis
Ver disponibilidade

Figure 7 - Inspecting each hotel's HTML "block" to discover their CSS class.

Since the information I want to extract is contained inside that block, and since I want to do that for every block available, I am now faced with another situation where a "for loop" will be very useful, as after I define the elements I want to extract, that extraction will occur for every block the program encounters. I can define a variable called BookingHotel, on which I will use Beautiful Soup to find and extract every block, with "soup.find_all".

```
BookingHotel = soup.find_all('div', {'class': 'a826ba81c4 fe821aea6c
fa2f36ad22 afd256fc79 d08f526e0d ed11e24d01 ef9845d4b3 da89aeb942'}
```

Listing 17 - BookingHotel function that finds all block elements that contain each accommodation.

If I print BookingHotel, I will receive the html code for all 25 listings, which itself will contain every bit of information I wish to extract.

So I want to create a “for loop” that, for each individual listing (which I will name “item”), extracts the elements we want, and then adds all the extracted information into a dictionary, which I will call BookingHotelDict.

```
for item in BookingHotel:
      try:
            name = item.find('div', {'class': 'fcab3ed991
a23c043802'}).text
      except:
            name = ‘None’
      try:
            zone = item.find('span', attrs={"data-testid":
      "address"}).text
      except:
            zone = ‘None’
      try:
            price = item.find('span', {'class': 'fcab3ed991
      bd73d13072'}).text
      except:
            price = ‘None’
      try:
            rating = item.find('div', {'class': 'b5cd09854e
d10a6220b4'}).text
      except:
            rating = 'None'
      try:
            commentsaux = item.find('div', {'class': 'd8eab2cf7f
      c90c0a70d3 db63693c62'}).text
            comments = re.sub("[^0-9]", "", commentsaux)
      except:
            comments = 'None'
      try:
            remainingaux = item.find('div', {'class':
      'cb1f9edcd4'}).text
            remaining = re.sub("[^0-9]", "", remainingaux)
      except:
            remaining = '10 ou mais'
            link = item.find('a', {'class': 'e13098a59f'})['href']

      BookingHotelDict = {
      'Name': name,
      'Zone': zone,
      'Price': price,
      'Rating': rating,
      'Comments': comments,
      'Remaining': remaining,
      'Link': link
      }
```

Listing 18 - "For loop" that extracts selected elements on each iteration.

On each item, the program tries to find and extract all these elements. However, if it searches for a specific element and does not find it, the program would normally stop and give me an error, because I only gave it instructions for a scenario where the element is found. Therefore, it is mandatory to add a “try” and “except” on each element I search, that way the program knows what to do in both scenarios. If it finds the element, it extracts it. If the element does not exist, I receive a “None”.

It is also important to mention that “soup.find” gives me the element it found, including html and CSS tags. I do not want that, as I am only looking for the text. Therefore, I must add a “.text” at the end of my variables, which will remove all the HTML/CSS tags and return only the text.

On the comments and remaining spots’ elements, I would receive a line of text that included the numbers I wanted to extract. Since I am only interested in the numbers, I can use regular expressions to remove the non-numeric characters. With re.sub, I can replace the whole text element with only the number characters, leaving me only with the numbers I want.

When creating my BookingHotelDict dictionary, I am getting a glimpse of what my future dataset will look like. With every item, the info extracted is added to this dictionary, which is then appended to my main list HotelsListBooking. With each loop, the list grows larger. In the end, that list will be transformed into a data frame, which will then be converted into an Excel file.

So far, I have defined my main function getHotelsBooking, which grabs and modifies the URL based on our parameters, accesses it, extracts the useful elements from the first page, adds them into a dictionary which is then progressively appended into our main list HotelsListBooking**.** Then, it verifies if it has reached the last page of results. If that is the case, the program stops. If it is not, the program jumps to the second page of results and repeats the whole process, until it reaches the final page and ends the loop.

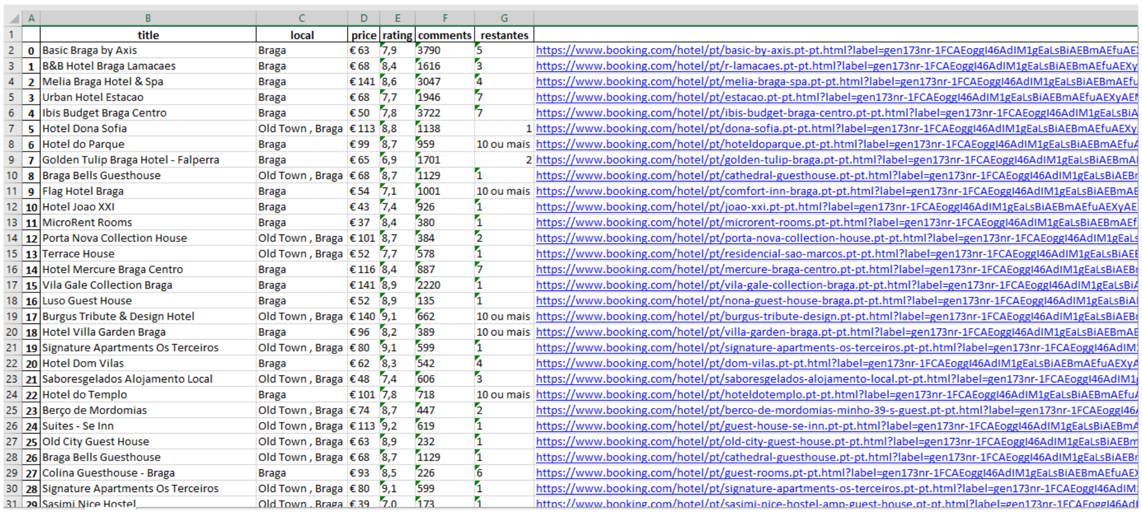
Once the loop is over, my list is finalized, and I can now convert it into a data frame.

To do so, I will use “pd.DataFrame”, a function from Pandas’ library that transforms a list into a data frame. In this case, I will name it “df”. Then, I convert that data frame into an Excel file, which will contain the exact date and time on which the dataset was created, ensuring there never exist files with the same name, otherwise the older file would be overwritten.

```
df = pd.DataFrame(HotelsListBooking)
df.to_excel('BOOKING
{}.xlsx'.format(pd.datetime.now().strftime('%m_%d_%Y_%H_%M_%S')))
```

Listing 19 - Creating a data frame and converting it to .xlsx.

I now have an Excel file called "BOOKING 06_04_2022_11_31_46" on my target folder, which includes all the data I have extracted from Booking.



	A	B	C	D	E	F	G	
1		title	local	price	rating	comments	restantes	
2	0	Basic Braga by Axis	Braga	€ 63	7,9	3790	5	https://www.booking.com/hotel/pt/basic-by-axis.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfuAE
3	1	B&B Hotel Braga Lamacaes	Braga	€ 68	8,4	1616	3	https://www.booking.com/hotel/pt/r-lamacaes.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXy
4	2	Melia Braga Hotel & Spa	Braga	€ 141	8,6	3047	4	https://www.booking.com/hotel/pt/melia-braga-spa.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfu
5	3	Urban Hotel Estacao	Braga	€ 68	7,7	1946	7	https://www.booking.com/hotel/pt/estacao.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXyAE
6	4	Ibis Budget Braga Centro	Braga	€ 50	7,8	3722	7	https://www.booking.com/hotel/pt/ibis-budget-braga-centro.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiA
7	5	Hotel Dona Sofia	Old Town , Braga	€ 113	8,8	1138	1	https://www.booking.com/hotel/pt/dona-sofia.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXy
8	6	Hotel do Parque	Braga	€ 99	8,7	959	10 ou mais	https://www.booking.com/hotel/pt/hoteldoparque.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfu
9	7	Golden Tulip Braga Hotel - Falperra	Braga	€ 65	6,9	1701	2	https://www.booking.com/hotel/pt/golden-tulip-braga.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmA
10	8	Braga Bells Guesthouse	Old Town , Braga	€ 68	8,7	1129	1	https://www.booking.com/hotel/pt/cathedral-guesthouse.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEB
11	9	Flag Hotel Braga	Braga	€ 54	7,1	1001	10 ou mais	https://www.booking.com/hotel/pt/comfort-inn-braga.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAE
12	10	Hotel Joao XXI	Braga	€ 43	7,4	926	1	https://www.booking.com/hotel/pt/joao-xxi.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXyAE
13	11	MicroRent Rooms	Braga	€ 37	8,4	380	1	https://www.booking.com/hotel/pt/microrent-rooms.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEf
14	12	Porta Nova Collection House	Old Town , Braga	€ 101	8,7	384	2	https://www.booking.com/hotel/pt/porta-nova-collection-house.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaL
15	13	Terrace House	Old Town , Braga	€ 52	7,7	578	1	https://www.booking.com/hotel/pt/residencial-sao-marcos.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAE
16	14	Hotel Mercure Braga Centro	Braga	€ 116	8,4	887	7	https://www.booking.com/hotel/pt/mercure-braga-centro.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEB
17	15	Vila Gale Collection Braga	Braga	€ 141	8,9	2220	1	https://www.booking.com/hotel/pt/vila-gale-collection-braga.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBi
18	16	Luso Guest House	Braga	€ 52	8,9	135	1	https://www.booking.com/hotel/pt/nona-guest-house-braga.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiA
19	17	Burgus Tribute & Design Hotel	Old Town , Braga	€ 140	9,1	662	10 ou mais	https://www.booking.com/hotel/pt/burgus-tribute-design.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEB
20	18	Hotel Villa Garden Braga	Braga	€ 96	8,2	389	10 ou mais	https://www.booking.com/hotel/pt/villa-garden-braga.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAE
21	19	Signature Apartments Os Terceiros	Old Town , Braga	€ 80	9,1	599	1	https://www.booking.com/hotel/pt/signature-apartments-os-terceiros.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM
22	20	Hotel Dom Vilas	Braga	€ 62	8,3	542	4	https://www.booking.com/hotel/pt/dom-vilas.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfuAEXyA
23	21	Saboresgelados Alojamento Local	Old Town , Braga	€ 48	7,4	606	3	https://www.booking.com/hotel/pt/saboresgelados-alojamento-local.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1
24	22	Hotel do Templo	Braga	€ 101	7,8	718	10 ou mais	https://www.booking.com/hotel/pt/hoteldotemplo.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfu
25	23	Berço de Mordomias	Old Town , Braga	€ 74	8,7	447	2	https://www.booking.com/hotel/pt/berco-de-mordomias-minho-39-s-guest.pt-pt.html?label=gen173nr-1FCAEoggI46
26	24	Suites - Se Inn	Old Town , Braga	€ 113	9,2	619	1	https://www.booking.com/hotel/pt/guest-house-se-inn.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmA
27	25	Old City Guest House	Old Town , Braga	€ 63	8,9	232	1	https://www.booking.com/hotel/pt/old-city-guest-house.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBm
28	26	Braga Bells Guesthouse	Old Town , Braga	€ 68	8,7	1129	1	https://www.booking.com/hotel/pt/cathedral-guesthouse.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEB
29	27	Colina Guesthouse - Braga	Braga	€ 93	8,5	226	6	https://www.booking.com/hotel/pt/guest-rooms.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM1gEaLsBiAEBmAEfuAE
30	28	Signature Apartments Os Terceiros	Old Town , Braga	€ 80	9,1	599	1	https://www.booking.com/hotel/pt/signature-apartments-os-terceiros.pt-pt.html?label=gen173nr-1FCAEoggI46AdIM
31	29	Sasimi Nice Hostel	Old Town , Braga	€ 39	7,0	173	1	https://www.booking.com/hotel/pt/sasimi-nice-hostel-amp-guest-house.pt-pt.html?label=gen173nr-1FCAEoggI46Ad

Figure 8 - Dataset with information from my first Booking.com extraction, on Excel (general results).

However, I had only accomplished this for Braga city. Within the context of this project, I had to do the same whole process for Galicia, North of Portugal and Central Portugal. I will not demonstrate such cases, as it consists of simple replacements of search parameters. Besides that, I also had to perform all those searches on three different date presets: in 1 day, in 30 days and in 60 days. This led to a very extensive code that will need a visual interface to simplify it, but more on that later.

For now, I have extracted every page available. However, I needed to find a way to extract only a specific number of pages, especially for testing purposes. While I was troubleshooting, it was not very practical to have to wait until all the pages are scraped to see if the extraction was indeed successful. Therefore, I will now demonstrate how I made my program allow me to choose the amount of pages I want to extract. Even ignoring the testing purposes that led to this modification on the code, it could still be useful, as one might want only to see the first pages of results, since they usually contain the best and most relevant results.

Going back to the “for loop”, choosing the amount of pages is actually simpler than having the loop end when it finds the last page. When I choose the number of pages I want to extract, I am already telling my program when to stop the loop. Therefore:

```
for x in range(0, page, 25):
      getHotelsBooking(x, zone, day1, month1, year1, day2, month2,
year2)
break
```

Listing 20 - "For loop" that receives the amount of pages I want to extract.

My loop starts at page 0, increases in increments of 25 (which equals one page) and ends at “page”, which will be the final page. Then, I only had to define that “page” will be the number that I choose. Technically, I could just say “page = 2”, but that would require me to change the code every time I want a different amount of pages. Therefore, I had to make it so the program receives user input with the number of pages they want.

Firstly, I had to keep in mind that the pages’ numbers on the URL are multiplied by 25, so the program has to do the same with the input received from the user.

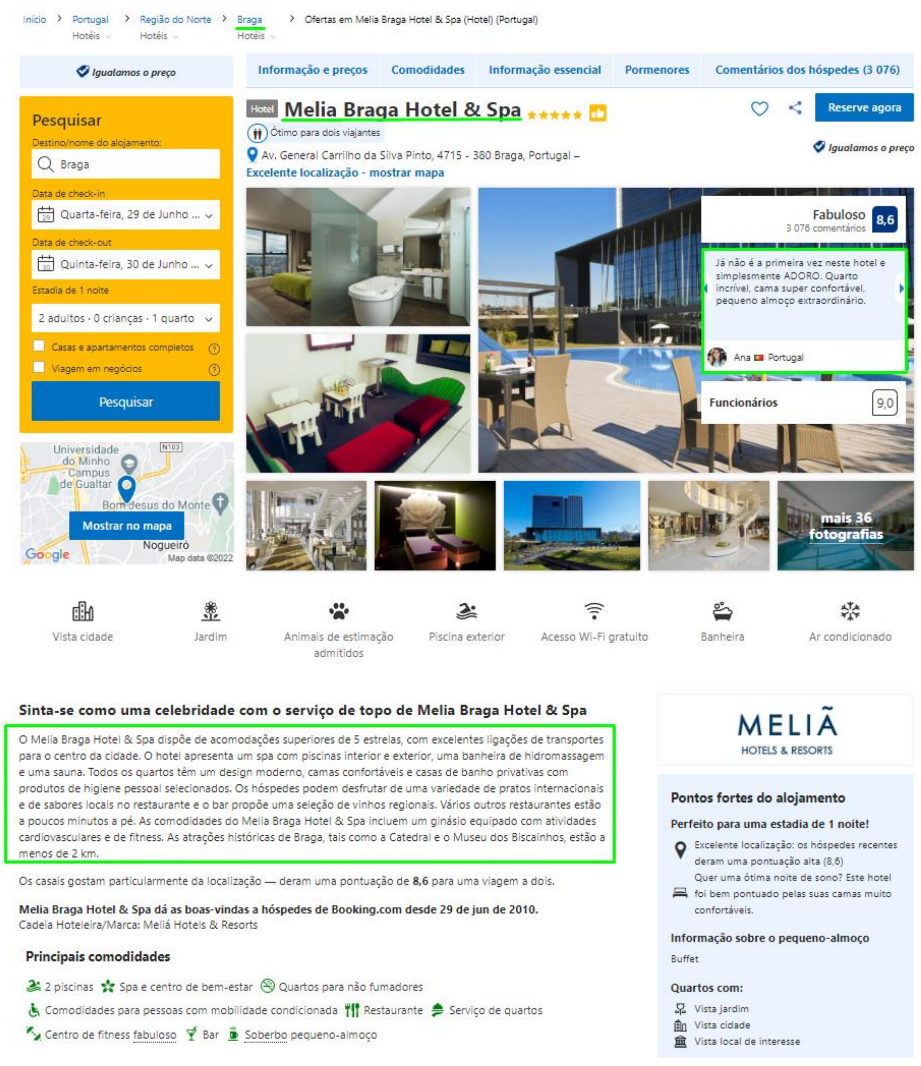
```
page = int(values['npages']) * 25
```

Listing 21 - Defining the number of pages I want to extract.

I defined page as an integer that multiplies the user input by 25. I named the user input “npages”, which will be explained later, as it is part of the GUI (Graphical User Interface) I added later, to make the program more intuitive and responsive, and to allow the user to interact with it.

4.1.2.2. SECOND EXTRACTION: INDIVIDUAL RESULTS

After analysing the datasets I have obtained by using my program, I concluded that they did not contain as much information as I wanted, since the search result pages only include a few details about each accommodation. Therefore, I decided to also extract information from each individual result, using the URLs previously extracted to be able to obtain even more data.


Início › Portugal Hotéis › Região do Norte Hotéis › Braga Hotéis › Ofertas em Melia Braga Hotel & Spa (Hotel) (Portugal)

Informação e preços | Comodidades | Informação essencial | Pormenores | Comentários dos hóspedes (3 076)

Igualamos o preço

Pesquisar
Destino/nome do alojamento: Braga
Data de check-in: Quarta-feira, 29 de Junho ...
Data de check-out: Quinta-feira, 30 de Junho ...
Estadia de 1 noite
2 adultos · 0 crianças · 1 quarto
Casas e apartamentos completos
Viagem em negócios
Pesquisar

Hotel **Melia Braga Hotel & Spa** ★★★★★
Ótimo para dois viajantes
Av. General Carrilho da Silva Pinto, 4715 - 380 Braga, Portugal –
Excelente localização - mostrar mapa

Reserve agora
Igualamos o preço

Fabuloso 8,6
3 076 comentários

Já não é a primeira vez neste hotel e simplesmente ADORO. Quarto incrível, cama super confortável, pequeno almoço extraordinário.

Ana Portugal

Funcionários 9,0

mais 36 fotografias

Mostrar no mapa

Vista cidade · Jardim · Animais de estimação admitidos · Piscina exterior · Acesso Wi-Fi gratuito · Banheira · Ar condicionado

Sinta-se como uma celebridade com o serviço de topo de Melia Braga Hotel & Spa

O Melia Braga Hotel & Spa dispõe de acomodações superiores de 5 estrelas, com excelentes ligações de transportes para o centro da cidade. O hotel apresenta um spa com piscinas interior e exterior, uma banheira de hidromassagem e uma sauna. Todos os quartos têm um design moderno, camas confortáveis e casas de banho privativas com produtos de higiene pessoal selecionados. Os hóspedes podem desfrutar de uma variedade de pratos internacionais e de sabores locais no restaurante e o bar propõe uma seleção de vinhos regionais. Vários outros restaurantes estão a poucos minutos a pé. As comodidades do Melia Braga Hotel & Spa incluem um ginásio equipado com atividades cardiovasculares e de fitness. As atrações históricas de Braga, tais como a Catedral e o Museu dos Biscainhos, estão a menos de 2 km.

Os casais gostam particularmente da localização — deram uma pontuação de **8,6** para uma viagem a dois.

Melia Braga Hotel & Spa dá as boas-vindas a hóspedes de Booking.com desde 29 de jun de 2010.
Cadeia Hoteleira/Marca: Meliá Hotels & Resorts

Principais comodidades

2 piscinas · Spa e centro de bem-estar · Quartos para não fumadores · Comodidades para pessoas com mobilidade condicionada · Restaurante · Serviço de quartos · Centro de fitness fabuloso · Bar · Soberbo pequeno-almoço

MELIÃ HOTELS & RESORTS

Pontos fortes do alojamento

Perfeito para uma estadia de 1 noite!

Excelente localização: os hóspedes recentes deram uma pontuação alta (8,6)

Quer uma ótima noite de sono? Este hotel foi bem pontuado pelas suas camas muito confortáveis.

Informação sobre o pequeno-almoço

Buffet

Quartos com:

Vista jardim
Vista cidade
Vista local de interesse

Figure 9 - Booking.com individual results for Braga city (first result).

When performing our global extractions, each Booking.com extraction from a single zone took about 10 minutes, as it had to open 40 different URLs (1000 results, 25 results in each page). For this next process, I had to open 1000 different URLs for each individual zone, which means that each single extraction took roughly 4 hours and 10 minutes. Following the same method described earlier, I inspected the page to find the CSS classes for the elements I wanted to extract, which were: name, zone, score, description and top 10 comments. Theoretically, it is possible to extract every single comment from every single result, but given the fact that some hotels contain over 1000 comments, simple math tells me that extracting comments from every result, in every zone and in all 3 time periods would take roughly 250 days of continuous extractions. Therefore, I extracted only the top 10 comments (which Booking highlights above the others).

Since each extraction was going to take approximately 4 hours, I had to make sure my code worked flawlessly, otherwise I would have to start the extraction from the top if any errors occurred during the process. In order to avoid one of the most common errors in web scraping, I added a "try" and "except" in every element I want to extract. For instance:

```
name = soup.find('a', {'id': 'hp_hotel_name_reviews'}).text
```

Listing 22 - Extracting the names of accommodations without adding exceptions.

If I only included the line listed above in my code, in each URL, the program would look for the hotel name and extract it normally. However, if it comes across a page that does not contain that hotel name element, the program would not know what to do, as it was only given instructions for a scenario where said element is found. This leads to an error that forces the code to stop. Therefore, I needed to do the following:

```
    try:
        name = soup.find('a', {'id': 'hp_hotel_name_reviews'}).text
    except:
        name = 'None'
```

Listing 23 - Extracting the names of accommodations with exceptions for when a name isn't found.

Now, the program knows that if the name element is found, it must extract the text of that element. If that element does not exist, then it adds a "None" in its place, thus continuing the program's normal course without disruptions.

The method of extraction is almost the same as the one used previously, but it varies in one main thing: instead of looping through a modifiable URL that constantly changes throughout the loops based on my input, it loops through a list of URLs I obtained previously, making it a much simpler process.

In order to obtain that list of URLs, I need to transform the column that contains the URLs in my extracted dataset into a list. The easiest way to do so is to read that Excel file into a data frame, and then convert the URL column into a list:

```
df = pd.read_excel(Booking North 30 days.xlsx', sheet_name="Sheet1")
links_list = df['link'].tolist()
```

Listing 24 - Converting my dataset into a data frame and transforming the column containing the links into a list.

Now that I have my list, named links_list, my main function getHotels will use the values of that list to access the URLs.

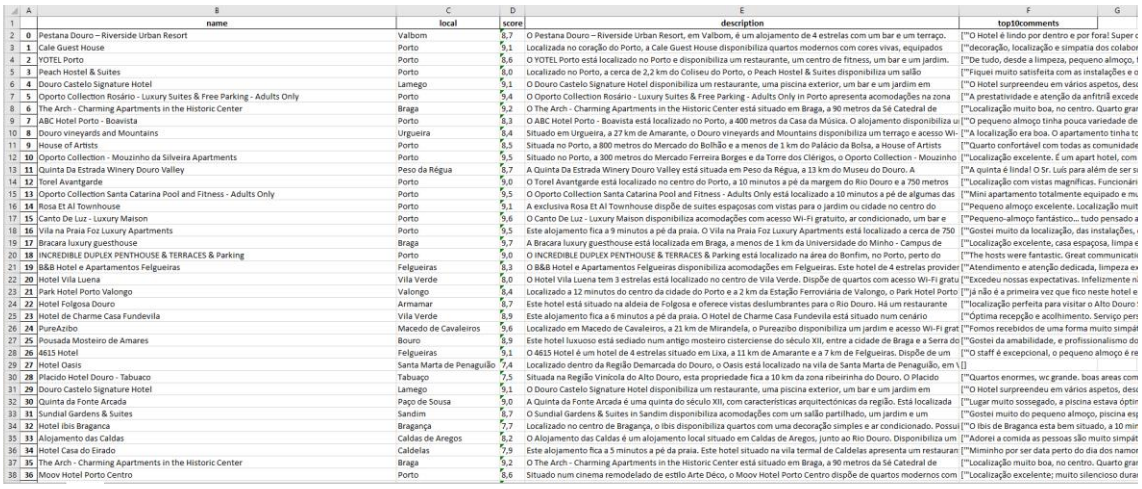
```
def getHotels(link):
      url = f'{link}'
```

Listing 25 - Defining a main function that will receive URLs.

With a simple loop, I can make getHotels loop through the entire list, thus extracting the data I want from every single URL.

```
for x in links_list:
      getHotels(x)
```

Listing 26 - "For loop" that cycles through the list of URLs.



	A	B	C	D	E	F	G
1		name	local	score	description	top10comments	
2	0	Pestana Douro – Riverside Urban Resort	Valbom	8,7	O Pestana Douro – Riverside Urban Resort, em Valbom, é um alojamento de 4 estrelas com um bar e um terraço.	['"O Hotel é lindo por dentro e por fora! Super c	
3	1	Cale Guest House	Porto	9,1	Localizada no coração do Porto, a Cale Guest House disponibiliza quartos modernos com cores vivas, equipados	['"decoração, localização e simpatia dos colabor	
4	2	YOTEL Porto	Porto	8,6	O YOTEL Porto está localizado no Porto e disponibiliza um restaurante, um centro de fitness, um bar e um jardim.	['"De tudo, desde a limpeza, pequeno almoço, t	
5	3	Peach Hostel & Suites	Porto	8,0	Localizado no Porto, a cerca de 2,2 km do Coliseu do Porto, o Peach Hostel & Suites disponibiliza um salão	['"Fiquei muito satisfeita com as instalações e o	
6	4	Douro Castelo Signature Hotel	Lamego	9,1	O Douro Castelo Signature Hotel disponibiliza um restaurante, uma piscina exterior, um bar e um jardim em	['"O Hotel surpreendeu em vários aspetos, desc	
7	5	Oporto Collection Rosário - Luxury Suites & Free Parking - Adults Only	Porto	9,4	O Oporto Collection Rosário - Luxury Suites & Free Parking - Adults Only in Porto apresenta acomodações na zona	['"A prestatividade e atenção da anfitriã excede	
8	6	The Arch - Charming Apartments in the Historic Center	Braga	9,2	O The Arch - Charming Apartments in the Historic Center está situado em Braga, a 90 metros da Sé Catedral de	['"Localização muito boa, no centro. Quarto gran	
9	7	ABC Hotel Porto - Boavista	Porto	8,3	O ABC Hotel Porto - Boavista está localizado no Porto, a 400 metros da Casa da Música. O alojamento disponibiliza u	['"O pequeno almoço tinha pouca variedade de	
10	8	Douro vineyards and Mountains	Urgueira	8,4	Situado em Urgueira, a 27 km de Amarante, o Douro vineyards and Mountains disponibiliza um terraço e acesso Wi-	['"A localização era boa. O apartamento tinha to	
11	9	House of Artists	Porto	8,5	Situada no Porto, a 800 metros do Mercado do Bolhão e a menos de 1 km do Palácio da Bolsa, a House of Artists	['"Quarto confortável com todas as comunidade	
12	10	Oporto Collection - Mouzinho da Silveira Apartments	Porto	9,5	Situado no Porto, a 300 metros do Mercado Ferreira Borges e da Torre dos Clérigos, o Oporto Collection - Mouzinho	['"Localização excelente. É um apart hotel, com	
13	11	Quinta Da Estrada Winery Douro Valley	Peso da Régua	8,7	A Quinta Da Estrada Winery Douro Valley está situada em Peso da Régua, a 13 km do Museu do Douro. A	['"A quinta é linda! O Sr. Luís para além de ser s	
14	12	Torel Avantgarde	Porto	9,0	O Torel Avantgarde está localizado no centro do Porto, a 10 minutos a pé da margem do Rio Douro e a 750 metros	['"Localização com vistas magnificas. Funcionári	
15	13	Oporto Collection Santa Catarina Pool and Fitness - Adults Only	Porto	9,5	O Oporto Collection Santa Catarina Pool and Fitness - Adults Only está localizado a 10 minutos a pé de algumas das	['"Mini apartamento totalmente equipado e mu	
16	14	Rosa Et Al Townhouse	Porto	9,1	A exclusiva Rosa Et Al Townhouse dispõe de suites espaçosas com vistas para o jardim ou cidade no centro do	['"Pequeno almoço excelente. Localização muit	
17	15	Canto De Luz - Luxury Maison	Porto	9,6	O Canto De Luz - Luxury Maison disponibiliza acomodações com acesso Wi-Fi gratuito, ar condicionado, um bar e	['"Pequeno-almoço fantástico... tudo pensado a	
18	16	Vila na Praia Foz Luxury Apartments	Porto	9,5	Este alojamento fica a 9 minutos a pé da praia. O Vila na Praia Foz Luxury Apartments está localizado a cerca de 750	['"Gostei muito da localização, das instalações,	
19	17	Bracara luxury guesthouse	Braga	9,7	A Bracara luxury guesthouse está localizada em Braga, a menos de 1 km da Universidade do Minho - Campus de	['"Localização excelente, casa espaçosa, limpa e	
20	18	INCREDIBLE DUPLEX PENTHOUSE & TERRACES & Parking	Porto	9,0	O INCREDIBLE DUPLEX PENTHOUSE & TERRACES & Parking está localizado na área do Bonfim, no Porto, perto do	['"The hosts were fantastic. Great communicati	
21	19	B&B Hotel e Apartamentos Felgueiras	Felgueiras	8,3	O B&B Hotel e Apartamentos Felgueiras disponibiliza acomodações em Felgueiras. Este hotel de 4 estrelas provider	['"Atendimento e atenção dedicada, limpeza ex	
22	20	Hotel Vila Luena	Vila Verde	8,0	O Hotel Vila Luena tem 3 estrelas está localizado no centro de Vila Verde. Dispõe de quartos com acesso Wi-Fi gratu	['"Excedeu nossas expectativas. Infelizmente n	
23	21	Park Hotel Porto Valongo	Valongo	8,4	Localizado a 12 minutos do centro da cidade do Porto e a 2 km da Estação Ferroviária de Valongo, o Park Hotel Porto	['"já não é a primeira vez que fico neste hotel e	
24	22	Hotel Folgosa Douro	Armamar	8,7	Este hotel está situado na aldeia de Folgosa e oferece vistas deslumbrantes para o Rio Douro. Há um restaurante	['"localização perfeita para visitar o Alto Douro	
25	23	Hotel de Charme Casa Fundevila	Vila Verde	8,9	Este alojamento fica a 6 minutos a pé da praia. O Hotel de Charme Casa Fundevila está situado num cenário	['"Óptima recepção e acolhimento. Serviço pers	
26	24	PureAzibo	Macedo de Cavaleiros	9,6	Localizado em Macedo de Cavaleiros, a 21 km de Mirandela, o Pureazibo disponibiliza um jardim e acesso Wi-Fi grat	['"Fomos recebidos de uma forma muito simpát	
27	25	Pousada Mosteiro de Amares	Bouro	8,9	Este hotel luxuoso está sediado num antigo mosteiro cisterciense do século XII, entre a cidade de Braga e a Serra do	['"Gostei da amabilidade, e profissionalismo do	
28	26	4615 Hotel	Felgueiras	9,1	O 4615 Hotel é um hotel de 4 estrelas situado em Lixa, a 11 km de Amarante e a 7 km de Felgueiras. Dispõe de um	['"O staff é excepcional, o pequeno almoço é re	
29	27	Hotel Oasis	Santa Marta de Penaguião	7,4	Localizado dentro da Região Demarcada do Douro, o Oasis está localizado na vila de Santa Marta de Penaguião, em V	[]	
30	28	Placido Hotel Douro - Tabuaco	Tabuaço	7,5	Situada na Região Vinícola do Alto Douro, esta propriedade fica a 10 km da zona ribeirinha do Douro. O Placido	['"Quartos enormes, wc grande. boas areas com	
31	29	Douro Castelo Signature Hotel	Lamego	9,1	O Douro Castelo Signature Hotel disponibiliza um restaurante, uma piscina exterior, um bar e um jardim em	['"O Hotel surpreendeu em vários aspetos, desc	
32	30	Quinta da Fonte Arcada	Paço de Sousa	9,0	A Quinta da Fonte Arcada é uma quinta do século XII, com características arquitectónicas da região. Está localizada	['"Lugar muito sossegado, a piscina estava óptin	
33	31	Sundial Gardens & Suites	Sandim	8,7	O Sundial Gardens & Suites in Sandim disponibiliza acomodações com um salão partilhado, um jardim e um	['"Gostei muito do pequeno almoço, piscina esp	
34	32	Hotel ibis Braganca	Bragança	7,7	Localizado no centro de Bragança, o ibis disponibiliza quartos com uma decoração simples e ar condicionado. Possui	['"O ibis de Braganca esta bem situado, a 10 min	
35	33	Alojamento das Caldas	Caldas de Aregos	8,2	O Alojamento das Caldas é um alojamento local situado em Caldas de Aregos, junto ao Rio Douro. Disponibiliza um	['"Adorei a comida as pessoas são muito simpát	
36	34	Hotel Casa do Eirado	Caldelas	7,9	Este alojamento fica a 5 minutos a pé da praia. Este hotel situado na vila termal de Caldelas apresenta um restauran	['"Miminho por ser data perto do dia dos namor	
37	35	The Arch - Charming Apartments in the Historic Center	Braga	9,2	O The Arch - Charming Apartments in the Historic Center está situado em Braga, a 90 metros da Sé Catedral de	['"Localização muito boa, no centro. Quarto gran	
38	36	Moov Hotel Porto Centro	Porto	8,6	Situado num cinema remodelado de estilo Arte Déco, o Moov Hotel Porto Centro dispõe de quartos modernos com	['"Localização excelente; muito silencioso dura	

Figure 10 - Dataset with information from my second Booking.com extraction, on Excel (individual results).

After this, all that was left to do was repeating the process for every zone and every time period, which took me a few days. Having finished those extractions, I then had 9 datasets for the global search results (3 for each zone in each time period) and 9 datasets for the individual URLs (3 for each zone in each time period).

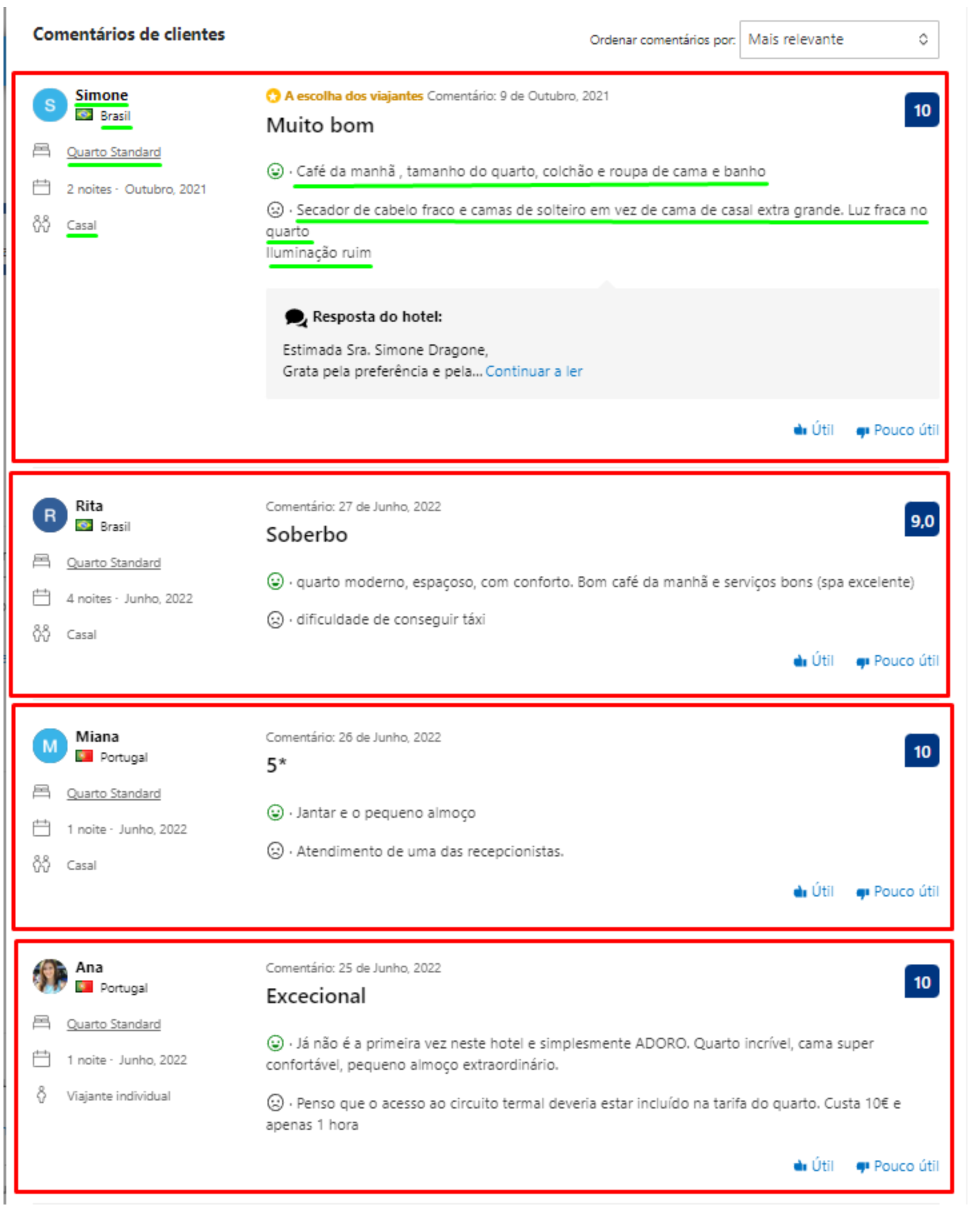
Theoretically speaking, it would be possible to merge those 2 groups of 9 datasets into a single group of 9 datasets, as the second group is technically a better version of the first, but since these extractions occurred during the course of a few days, it led to a situation where the global search results for a specific zone did not include the exact same 1000 results as the individual URLs, as those extractions needed to occur at the exact same moment to ensure the page does not get updated in the meantime.

Thus, the new datasets had the advantage of allowing me to create additional graphs based on the accommodation descriptions and top 10 comments, while the old datasets had the advantage of being extremely faster to extract, although restricting me to analysing only general data.

4.1.2.3. THIRD EXTRACTION: USER COMMENTS

While the user comments may contain tons of potentially valuable information, there is one large con in extracting that type of data: it takes way too much time due to the large volume of comments on each accommodation. If I were to extract the comments from every result of every zone, it would take me over a month. Despite that, there is no reason for me not to, at least, construct the code that extracts and analyses the comments of each specific hotel. In the future, if necessary, one simple “for in” loop can apply this extraction to every single hotel in a list of URLs, just like it was described in the previous subchapter of this document.

For demonstration purposes, I extracted 3050 comments from “Melia Braga Hotel & Spa”. I chose to collect the names, nationalities, comments, scores given and type of room.


Comentários de clientes

Ordenar comentários por: Mais relevante

Simone
Brasil
Quarto Standard
2 noites · Outubro, 2021
Casal

A escolha dos viajantes Comentário: 9 de Outubro, 2021
Muito bom
10
· Café da manhã , tamanho do quarto, colchão e roupa de cama e banho
· Secador de cabelo fraco e camas de solteiro em vez de cama de casal extra grande. Luz fraca no quarto
Iluminação ruim

Resposta do hotel:
Estimada Sra. Simone Dragone,
Grata pela preferência e pela... Continuar a ler

Útil Pouco útil

Rita
Brasil
Quarto Standard
4 noites · Junho, 2022
Casal

Comentário: 27 de Junho, 2022
Soberbo
9,0
· quarto moderno, espaçoso, com conforto. Bom café da manhã e serviços bons (spa excelente)
· dificuldade de conseguir táxi

Útil Pouco útil

Miana
Portugal
Quarto Standard
1 noite · Junho, 2022
Casal

Comentário: 26 de Junho, 2022
5*
10
· Jantar e o pequeno almoço
· Atendimento de uma das recepcionistas.

Útil Pouco útil

Ana
Portugal
Quarto Standard
1 noite · Junho, 2022
Viajante individual

Comentário: 25 de Junho, 2022
Excecional
10
· Já não é a primeira vez neste hotel e simplesmente ADORO. Quarto incrível, cama super confortável, pequeno almoço extraordinário.
· Penso que o acesso ao circuito termal deveria estar incluído na tarifa do quarto. Custa 10€ e apenas 1 hora

Útil Pouco útil

Figure 11 - Comments section on Melia Braga Hotel & Spa.

I used the same method as the one in the two previous subsections: I defined a function getComments that modifies the {page} variable of the URL, based on the for loop sequence.

```
def getComments(page):
      url= f'https://www.booking.com/reviewlist.pt-pt.html?aid=356980&label=gog235jc-1DCAsouwFCD21lbGlhLWJyYWdhLXNwYUgfWANouwGIAQGYAR-4ARfIAQzYAQPoAQH4AQKIAgGoAgO4AsqjwZUGwAIB0gIkY2NjMjhiMjUtY2U4NC00ODY3LTgyZGQtMmJiYTJlODU3NmI32AIE4AIB&sid=10eb916e05932c73e6abd20218be0222&cc1=pt;dist=1;length_of_stay=1;pagename=melia-braga-spa;type=total&&offset={page};rows=10'
```

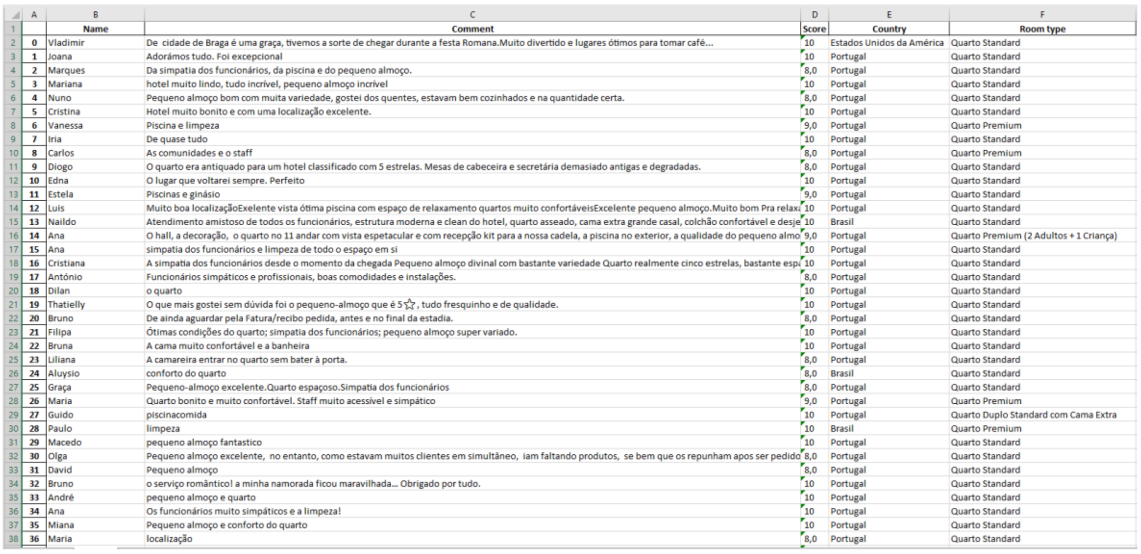
Listing 27 - Defining a function that modifies the "page" variable on the URL.

In this case, I set the "page" value to 3050 so that my loop, which cycles in leaps of 10, does not stop until it reaches the 306th page, that being the last page.

```
page = 3050
for x in range(0, page, 10):
    getComments(x)
```

Listing 28 - "For loop" that will cycle through all pages until it reaches the last one.

When the extraction had finished, I obtained the following dataset:



	A	B	C	D	E	F
1		Name	Comment	Score	Country	Room type
2	0	Vladimir	De cidade de Braga é uma graça, tivemos a sorte de chegar durante a festa Romana.Muito divertido e lugares ótimos para tomar café...	10	Estados Unidos da América	Quarto Standard
3	1	Joana	Adorámos tudo. Foi excepcional	10	Portugal	Quarto Standard
4	2	Marques	Da simpatia dos funcionários, da piscina e do pequeno almoço.	8,0	Portugal	Quarto Standard
5	3	Mariana	hotel muito lindo, tudo incrível, pequeno almoço incrível	10	Portugal	Quarto Standard
6	4	Nuno	Pequeno almoço bom com muita variedade, gostei dos quentes, estavam bem cozinhados e na quantidade certa.	8,0	Portugal	Quarto Standard
7	5	Cristina	Hotel muito bonito e com uma localização excelente.	10	Portugal	Quarto Standard
8	6	Vanessa	Piscina e limpeza	9,0	Portugal	Quarto Premium
9	7	Iria	De quase tudo	10	Portugal	Quarto Standard
10	8	Carlos	As comunidades e o staff	8,0	Portugal	Quarto Premium
11	9	Diogo	O quarto era antiquado para um hotel classificado com 5 estrelas. Mesas de cabeceira e secretária demasiado antigas e degradadas.	8,0	Portugal	Quarto Standard
12	10	Edna	O lugar que voltarei sempre. Perfeito	10	Portugal	Quarto Standard
13	11	Estela	Piscinas e ginásio	9,0	Portugal	Quarto Standard
14	12	Luis	Muito boa localizaçãoExelente vista ótima piscina com espaço de relaxamento quartos muito confortáveisExcelente pequeno almoço.Muito bom Pra relaxa	10	Portugal	Quarto Standard
15	13	Naildo	Atendimento amistoso de todos os funcionários, estrutura moderna e clean do hotel, quarto asseado, cama extra grande casal, colchão confortável e desje	10	Brasil	Quarto Standard
16	14	Ana	O hall, a decoração, o quarto no 11 andar com vista espetacular e com recepção kit para a nossa cadela, a piscina no exterior, a qualidade do pequeno almo	9,0	Portugal	Quarto Premium (2 Adultos + 1 Criança)
17	15	Ana	simpatia dos funcionários e limpeza de todo o espaço em si	10	Portugal	Quarto Standard
18	16	Cristiana	A simpatia dos funcionários desde o momento da chegada Pequeno almoço divinal com bastante variedade Quarto realmente cinco estrelas, bastante espa	10	Portugal	Quarto Standard
19	17	António	Funcionários simpáticos e profissionais, boas comodidades e instalações.	8,0	Portugal	Quarto Standard
20	18	Dilan	o quarto	10	Portugal	Quarto Standard
21	19	Thatielly	O que mais gostei sem dúvida foi o pequeno-almoço que é 5☆, tudo fresquinho e de qualidade.	10	Portugal	Quarto Standard
22	20	Bruno	De ainda aguardar pela Fatura/recibo pedida, antes e no final da estadia.	8,0	Portugal	Quarto Standard
23	21	Filipa	Ótimas condições do quarto; simpatia dos funcionários; pequeno almoço super variado.	10	Portugal	Quarto Standard
24	22	Bruna	A cama muito confortável e a banheira	10	Portugal	Quarto Standard
25	23	Liliana	A camareira entrar no quarto sem bater à porta.	8,0	Portugal	Quarto Standard
26	24	Aluysio	conforto do quarto	8,0	Brasil	Quarto Standard
27	25	Graça	Pequeno-almoço excelente.Quarto espaçoso.Simpatia dos funcionários	8,0	Portugal	Quarto Standard
28	26	Maria	Quarto bonito e muito confortável. Staff muito acessível e simpático	9,0	Portugal	Quarto Premium
29	27	Guido	piscinacomida	10	Portugal	Quarto Duplo Standard com Cama Extra
30	28	Paulo	limpeza	10	Brasil	Quarto Premium
31	29	Macedo	pequeno almoço fantastico	10	Portugal	Quarto Standard
32	30	Olga	Pequeno almoço excelente, no entanto, como estavam muitos clientes em simultâneo, iam faltando produtos, se bem que os repunham apos ser pedido	8,0	Portugal	Quarto Standard
33	31	David	Pequeno almoço	8,0	Portugal	Quarto Standard
34	32	Bruno	o serviço romântico! a minha namorada ficou maravilhada... Obrigado por tudo.	10	Portugal	Quarto Standard
35	33	André	pequeno almoço e quarto	10	Portugal	Quarto Standard
36	34	Ana	Os funcionários muito simpáticos e a limpeza!	10	Portugal	Quarto Standard
37	35	Miana	Pequeno almoço e conforto do quarto	10	Portugal	Quarto Standard
38	36	Maria	localização	8,0	Portugal	Quarto Standard

Figure 12 - Dataset with information from my third Booking.com extraction, on Excel (user comments).

With my extractions successfully performed, the web scraping portion of this methodology was then concluded.

4.2. GRAPHICAL USER INTERFACE

While creating my program, I have noticed that it was getting increasingly harder to test the various functions I was designing, as every time I wanted to run something, I had to go on Jupyter Notebook and type the function I wanted to run, which is not very practical. There was a point where I had multiple functions, all very similar, only differentiating on their search parameters. So I figured: "why not build a user interface with a simple but intuitive design, that will allow me to run my code without the hassle of typing the name of the functions?". That is when I came across PySimpleGUI, a Python package that allowed me to create a GUI for my program. After testing for a bit, I realised that it was a very easy-to-use tool that facilitates not only the task of creating a working user interface, but also the process of troubleshooting during my tests. It comes with many templates and colour schemes, and it gives me the freedom to place the UI elements wherever I want, given the right commands.

To begin this process, I had to create a new function, which I named booking_window. I started with a default layout to see how the package looks, and based on that default window, I drew a quick sketch of what I wanted my window to look like:

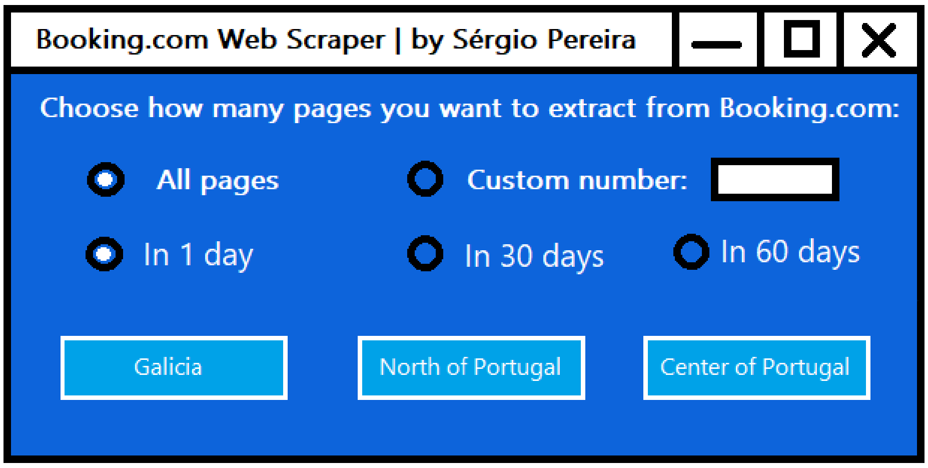

Booking.com Web Scraper | by Sérgio Pereira
Choose how many pages you want to extract from Booking.com:
All pages
Custom number:
In 1 day
In 30 days
In 60 days
Galicia
North of Portugal
Center of Portugal

Figure 13 - Sketch of a potential GUI for my program.

Since I had made functions for extracting every page and for extracting only a specified number of pages, I added radio buttons, which make it so you can only choose one of them at a time, otherwise my program would crash, as it can't perform both actions at the same time. "Custom number" includes an input field, on which I can enter the number of pages, which is the npages mentioned earlier. I also added buttons for each time-period.

Transforming the sketch above into a real window was easier than I expected: I used sg.Text to insert a line of text, sg.Radio to add the radio buttons, using "font" and "size" to customise the text to my liking, and sg.Button to add the time-period buttons. I used the theme "DarkTeal12" as its colour scheme resembles the Booking.com logo.

```
def booking_window():
    sg.theme('DarkTeal12')
    layout = [
        [sg.Text('Choose how many pages you want to extract from 
Booking.com:',  font='Verdana')],
        [sg.Radio('All pages', "NumberOfPages", key="-Q1-", 
default=True, font='Verdana', size=(15,1)), sg.Radio('Custom 
number:', "NumberOfPages", key="-Q2-",  font='Verdana'), 
sg.InputText(key='pagina', size=(8,1))],
[sg.T(""), sg.T(""), sg.Radio('In 1 day', "Data", key="-Q3-", 
default=True, font='Verdana', size=(15,1)), sg.Radio('In 30 days', 
"Data", key="-Q4-", default=False, font='Verdana', size=(15,1)), 
sg.Radio('In 60 days', "Data", key="-Q5-", default=False, 
font='Verdana', size=(10,1)) ],
[sg.Button('Galicia', size=(17,2), font='Verdana'), sg.Button(North 
of Portugal', size=(17,2), font='Verdana'), sg.Button('Center of 
Portugal', size=(17,2), font='Verdana')],
    ]
    window = sg.Window('Booking.com Web Scraper | by Sérgio 
Pereira', layout)
```

Listing 29 - Defining the interface for my Booking.com window.

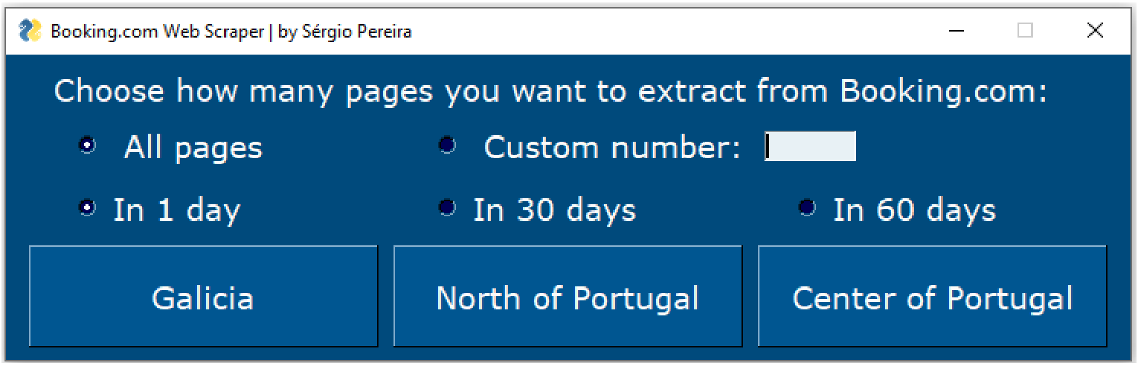

Booking.com Web Scraper | by Sérgio Pereira
Choose how many pages you want to extract from Booking.com:
All pages
Custom number:
In 1 day
In 30 days
In 60 days
Galicia
North of Portugal
Center of Portugal

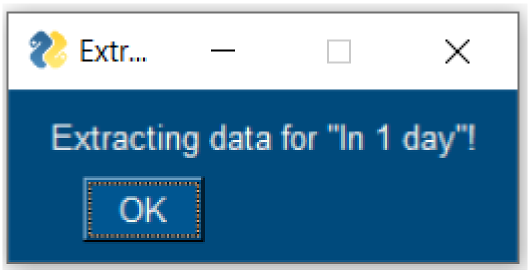
Figure 14 - Web scraping tool user interface for Booking.com.

For the sake of demonstrating this in a clearer manner, I will keep this current code limited to results of “in 1 day”, as the rest of the options will be exactly in the same code structure, but with different parameters.

Now that I have my main UI window, I figured it would be useful to add two pop-up windows: one that shows up when you start extracting data, and one that shows up when all the data has been successfully extracted. Firstly, I will create both windows, and then, in the end, I will place them in the correct position, after merging my main scraping code into the PySimpleGUI code. To create the pop-up windows, one must only use the sg.popup function, adding the desired message between parenthesis:

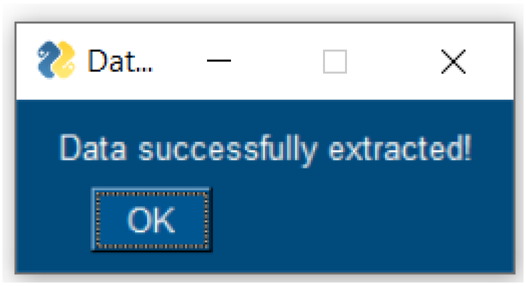
```
sg.popup('Extracting data for "In 1 day"!')
sg.popup('Data successfully extracted!')
```

Listing 30 - Adding pop-ups for the beginning and end of the extraction.


Extr...

Extracting data for "In 1 day"!

Figure 15 - Pop-up window when extraction begins.


Dat...

Data successfully extracted!

Figure 16 - Pop-up window when extraction ends.

With my GUI created, all that is left to do is adding my whole web scraping code into the GUI code. So going back into booking_window, right after my window code, I need to create a "while" loop with "if" statements, that will dictate what happens when I click the elements of my interface.

```
while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED:
                break
```

Listing 31 - "While" loop that stops when the window is closed.

Firstly, I am creating an event loop that processes the events and receives the values of the inputs I enter. I am also telling my program to stop that loop when the window is closed.

Secondly, I need to create another event, which will be the one where I select the "Galicia" button, while also choosing to extract all pages and choosing "In 1 day" as the check-in date. Therefore, I need to use "Galicia" as the event's name, while also giving the values of "All pages" and "In 1 day” keys, which are “-Q1-“ and “-Q3-“. Besides that, I need to add the variables I created earlier that define my search parameters as “today” and “tomorrow”, while also defining Galicia’s code that appears on the URL. Afterwards, I need the “for loop” that runs the code until it finds the last page.

```
if (event == 'Galicia') and (values ["-Q1-"] == True) and (values
["-Q3-"] == True):
    day1 = daytoday
    month1 = monthtoday
    year1 = yeartoday
    day2 = daytomorrow
    month2 = monthtomorrow
    year2 = yeartomorrow
    zone = '735'
    sg.popup('Extracting data for "In 1 day", in Galicia!')
    for x in range(0, 10000, 25):
        getHotelsBooking(x, day1, month1, year1, day2, month2,
year2)
        if hasattr(getHotelsBooking.NotLastPage, '__len__') == True:
            continue
        else:
            break
    break
```

Listing 32 - Defining what happens when selecting Galicia, while extracting all pages with "In 1 day" selected.

Then, I created another event possibility, on which I extract Galicia’s results from “In 1 day”, while selecting a specific amount of pages. Following the same procedure, I name it the same as the “Galicia” button, but this time I give it the value of the “Custom number” radio button, which is “-Q2-“, while still invoking “In 1 day”’s key, which is “-Q3”. Following that, I add my code that extracts a specified number of pages, on which I will match “page” with the number received on the input.

```
if (event == 'Galicia') and (values ["-Q2-"] == True) and (values
["-Q3-"] == True):
    page = int(values['npages']) * 25
    day1 = daytoday
    month1 = monthtoday
    year1 = yeartoday
    day2 = daytomorrow
    month2 = monthtomorrow
    year2 = yeartomorrow
    zone = '735'
    sg.popup('Extracting data for "In 1 day", in Galicia!')
    for x in range(0, page, 25):
        getHotelsBooking(x, day1, month1, year1, day2, month2,
year2)
    break
```

Listing 33 - Defining what happens when selecting Galicia, while extracting a specific amount of pages, with "In 1 day" selected.

With the same procedure, I had to repeat the same steps for the other 2 zones, while also adding the events for the other 2 time periods. This resulted in 18 different possible events, which would have been much harder to comprehend if I had not implemented a GUI to execute them. After finishing the codes for the remaining zones and time options, this GUI's flowchart looked like the following:

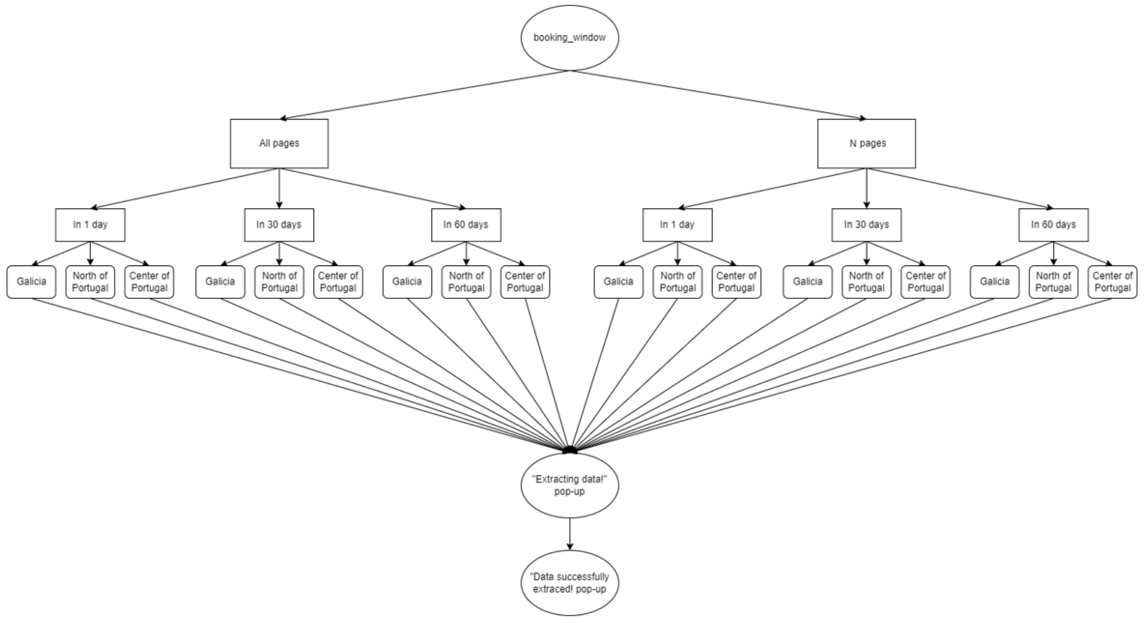


Figure 17 - Flowchart for Booking.com's portion of my web scraping tool's GUI.

With my program performing everything I want, it is now time to go back to the beginning and repeat the whole process with the remaining two websites, Tripadvisor and Airbnb. Since the methods are the same, I will skip the coding process and jump straight to the GUI part.

Starting with the window itself, I figured it would be better if there was a specific window for each online platform, otherwise the main platform would include too many elements to choose and manipulate, which could lead to an overload of information for the user.

When creating the Tripadvisor window, whose function I named tripadvisor_window, I decided that I should follow my previous choices and go with a color scheme similar to the company's logo, that being the "DarkTeal13" theme.

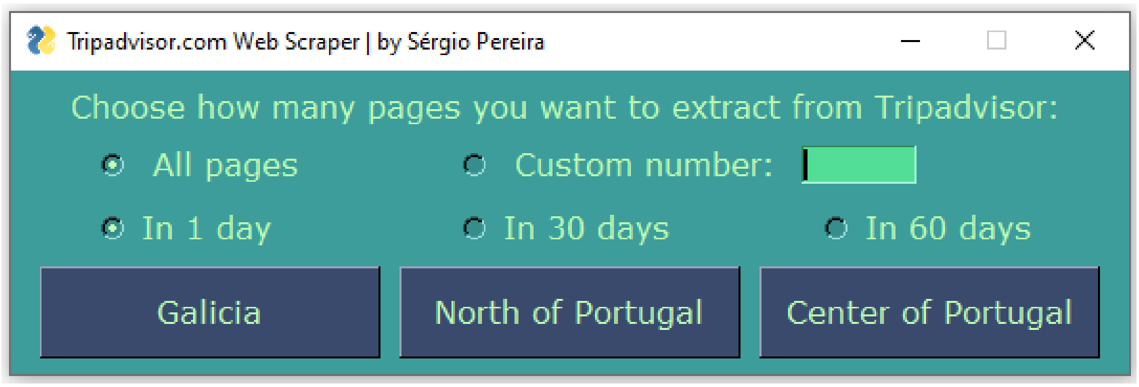

Tripadvisor.com Web Scraper | by Sérgio Pereira
Choose how many pages you want to extract from Tripadvisor:
All pages
Custom number:
In 1 day
In 30 days
In 60 days
Galicia
North of Portugal
Center of Portugal

Figure 18 - Web scraping tool user interface for TripAdvisor.

With Airbnb, and following the same ideas of my 2 previous cases, the theme used was “DarkRed1”. Its function was named airbnb_window.

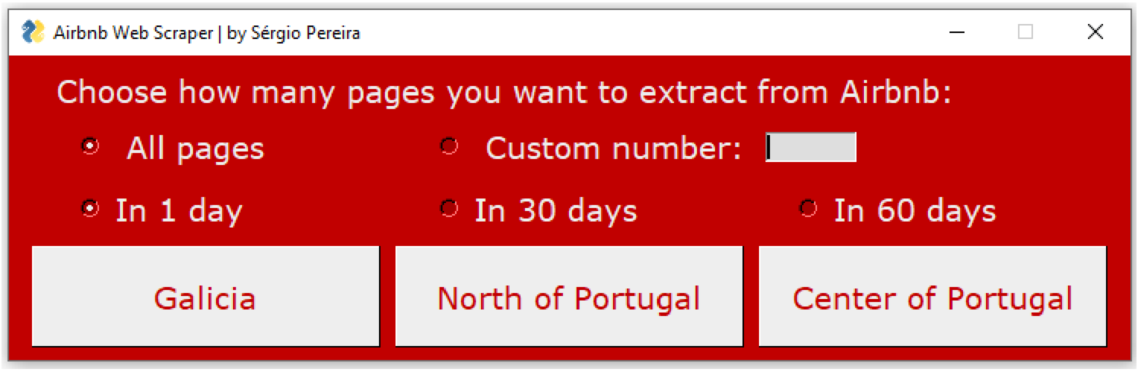


Figure 19 - Web scraping tool user interface for Airbnb.

As I now have three different windows for three different platforms, it is imperative that I create a new main window, on which I can choose from which of the three platforms I want to extract data. Following the same procedure:

```
sg.theme('BrownBlue')
layout = [
    [sg.Text('Choose the online tourism platform to extract data
from:',  font='Verdana')],
    [sg.Button('Booking.com', size=(14,2), font='Verdana'),
sg.Button('Tripadvisor', size=(14,2), font='Verdana'),
sg.Button('Airbnb', size=(14,2), font='Verdana')],
]
window = sg.Window('Tourism platform Web Scraper | by Sérgio
Pereira', layout)
```

Listing 34 - Defining the interface of my main window, on which I can select the online platform to extract data from.

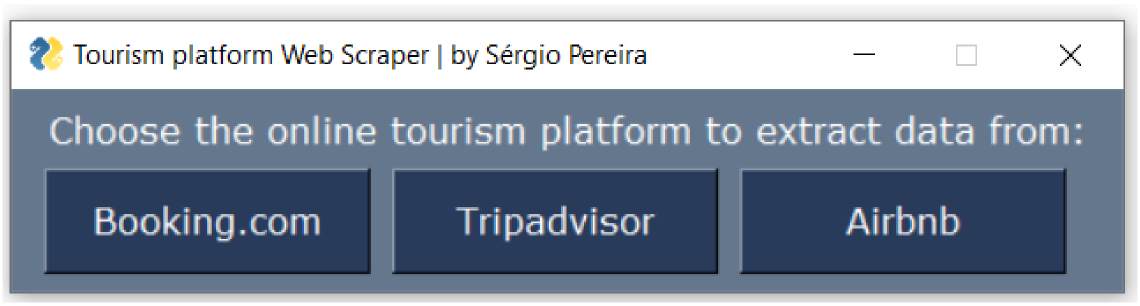

Tourism platform Web Scraper | by Sérgio Pereira
Choose the online tourism platform to extract data from:
Booking.com
Tripadvisor
Airbnb

Figure 20 - Web scraping tool user interface for the main window.

In order to make it work, I created another set of conditions based on the three possible options to choose from.

```
while True:
    event, values = window.read()
    if event == sg.WIN_CLOSED:
        break
    if event == 'Booking.com':
        booking_window()
    if event == 'Tripadvisor':
        tripadvisor_window()
    if event == 'Airbnb':
        airbnb_window()
window.close()
```

Listing 35 - Defining what happens when each of the buttons is clicked.

I now have a program that allows me to choose any of the three tourism platforms, and within each of the three, I can choose the amount of pages to extract, the zone to extract from and the time period from which the results will be. The following flowchart illustrates every possible event (excluding the variability of the amount of pages chosen for extraction):

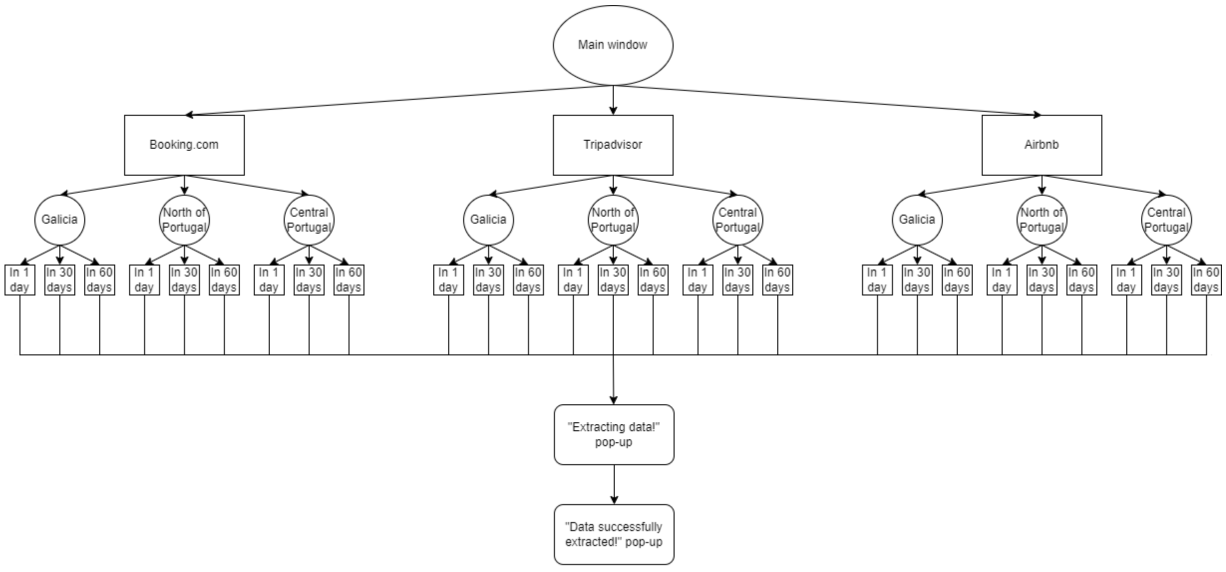


Figure 21 - Web scraping tool user interface flowchart.

4.3. DATA CLEANING

While my data may have been extracted successfully, it might not have come exactly the way I wanted. Sometimes, it might have extra words that I do not want, or additional characters that are not part of the element I wanted to extract. Very frequently, it will also have duplicate results, which need to be erased. That means I will have to open Excel and clean the dataset in order to make it "readable". For this task, regular expressions and/or simple substitutions come in handy.
In computer science, regular expressions are patterns used to select combinations of characters in a string, used to match, find and manage text. This will allow me to replace the elements that have extra characters with the correct ones, without having to do it manually.

As I wanted this phase to be short and quick, I tried to clean the extracted data as much as possible before actually extracting it, with the assistance of Beautiful Soup and regular expressions in the original extraction code. For instance, if I extract the element that contains the number comments and the element that contains the number of rooms remaining, at the current price, I will get the following, without using regular expressions:

"1322 comments" instead of "1322".

"Only 3 rooms remaining at the current price" instead of "3".

In order to fix this, I had to use regular expressions:

```
comments_aux = item.find('div', {'class': 'd8eab2cf7f c90c0a70d3
db63693c62'}).text
comments = re.sub("[^0-9]", "", comments_aux)
remaining_aux = item.find('div', {'class': 'cb1f9edcd4'}).text
remaining = re.sub("[^0-9]", "", remaining_aux)
```

Listing 36 - Using regular expressions to clean data.

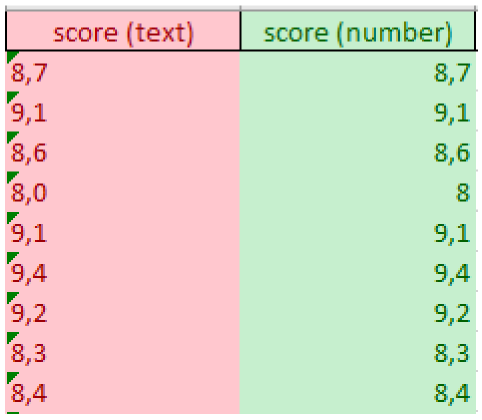
This way, I only receive the number, as with this regular expression I am instructing my program to replace anything that is not a digit from 0 to 9 with “”(literally nothing, so I am essentially deleting everything that is not a number).

Fortunately, after extracting my data, there is only one column that does need to be cleaned, which is the price column, that includes the “€” symbol. Even though it looks better that way, as it is easier to interpret the information when I can see the currency symbol, I needed to remove it because the price it is a numeric element. If I decide to make any type of graph based on the price, I need to be able to use those numbers in calculations, but if it comes with a symbol, it is no longer considered a number and the programs will not be able to interpret its numeric value.

In order to clean this column, a simple “find and replace” on Excel should do the trick. I just had to select the column I wanted to clean, press “Ctrl” + “L”, select the “Replace tab” and replace “€ “ by “” (empty field, which means I was replacing the text I did not want with nothing, leaving only the price).

Another common occurrence with web scraping is the extraction of numbers that do not get stored as actual numbers. When numbers get stored as text values instead of numeric values, they are not usable for statistical purposes, as the programs cannot interpret the value of those numbers. On Excel it is very easy to tell if the numbers on a column are stored as text or numeric values: if they are being interpreted as text, the numbers are aligned on the left side of the column, just like other text columns; if they are being interpreted as numbers, they will be aligned on the right side of the column.

In order to transform numbers stored as text into actual numbers, I selected the column containing those numbers, then selected “Text to Columns”, on the “Data” tab. After clicking “Next” on the three windows that appearing, selecting the default choices, the numbers gained their numeric values.



score (text)	score (number)
8,7	8,7
9,1	9,1
8,6	8,6
8,0	8
9,1	9,1
9,4	9,4
9,2	9,2
8,3	8,3
8,4	8,4

Figure 22 - Example of numbers stored as text (left column) and as numbers (right column).

With all data clean and ready to be analysed, there was still one step left: checking for duplicates. It is very frequent to encounter duplicate data in our extractions: sometimes the same result appears on subsequent pages, as the last result of the previous page and the first result of the following page; sometimes, there are hotels that are added to the website during the extraction, pushing every hotel out of its original position, and sometimes, the same hotel simply appears in more than one page, always leading to a webpage with the exact same number of comments, rating and price, even if it is through a different URL.

To remove duplicated results, I used the “Remove Duplicates” option on Excel’s Data tab. I simply had to select the “title”, “zone” and “price” columns; if there were any results with the same title, zone and price, Excel would remove the duplicated ones. Doing this removed approximately 200 duplicated results.

With all the datasets clean, they were then ready to be analysed, with the assistance of data visualisation tools.

4.4. DATA VISUALISATION

Throughout my year in this project, I have experimented with several data visualisation methods. I started with the R programming language, which proved to be very efficient and useful, although very hard to get into. As this project was almost entirely based on the Python programming language, for me it did not make sense to use another language for visualising the extracted data, so I sticked with Python, using several libraries, such as Pandas, Plotly, Dash, NLTK, among others. Despite also having experimented with Tableau, which gave me excellent results, I wanted to make the whole process (from extraction to analysis) achievable using only Python, and so I did.

As I had many datasets from three different zones and different time periods, I will use my dataset with Booking.com results from North of Portugal, 30 days after the extraction, for demonstration purposes. All the following data visualisation examples will be applied to all datasets of every zone and check-in date.

4.4.1. LIBRARIES AND PACKAGES

Plotly

Plotly is a Python graphing library that creates a wide range of interactive graphs. It is compatible with Dash, which was used to display those graphs.

Dash

It is an open source framework used for building data visualisation interfaces. It was used to create a dashboard that displays all the graphs created with Plotly. It is very useful, as it allowed me to create a data visualisation platform without requiring extensive web development knowledge.

NLTK

It is one of the most used libraries for natural language processing. It was used to analyse textual data from the datasets I have obtained, such as the user comments, for instance.

Matplotlib

Matplotlib is another library for Python that can create both static and interactive graphs.

4.4.2. CREATING GRAPHS

Before beginning, I tried to analyse all the elements available in my datasets to figure out which type of graphs made sense. Having done that, I decided to represent the following data:

1. Most frequent zones;
2. Average price per zone, in Euros;
3. Top 10 most expensive results;
4. Top 10 cheapest results;
5. Average score per zone;
6. Most frequent nationalities in user's comments;
7. Positive comments' word cloud;
8. Negative comments' word cloud.

The first 5 graphs were created using the datasets obtained from my first extraction. The last 3 graphs were based on my third extraction, which originated a dataset with comments from "Melia Braga Hotel & Spa", for demonstration purposes.

After importing the necessary libraries and packages, this process starts by creating a data frame of our dataset. Using pd.read_excel, from Pandas, I transformed my "Booking North of Portugal - 30 days" dataset into a data frame named dfnorth, which will be used to create graphs.

```
dfnorth = pd.read_excel('NORTE 30 DIASlol.xlsx',
sheet_name="Sheet1")
```

Listing 37 - Converting my dataset into a data frame.

1. Most frequent zones

To represent the most frequent zones, I used a pie graph. I started by creating a new dataframe using the column that includes the zone information.

```
dfnorthcounts_aux = dfnorth['zone'].value_counts()
dfnorthcounts = pd.DataFrame(dfnorthcounts_aux)
```

Listing 38 - Creating a new data frame that includes "zone".

With ".value_counts()", I have obtained the amount of times each zone appears in our dataset. Instead of having multiple lines with the same zone, my data frame now had one line for each zone, each with the corresponding number of occurrences.

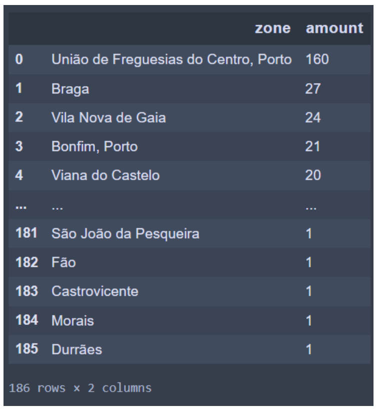
As the new data frame included a new column that had no header, I renamed it to "amount", in order the make this code cleaner and easier to understand.

```
dfnorthcounts = dfnorthcounts.reset_index()
dfnorthcounts.rename(columns = {'zone':'amount', 'index':'zone'},
inplace = True)
```

Listing 39 - Renaming the columns of my data frame.

Initially, I used ".dfnorthcounts_top10.loc[dfnorthcounts_top10.amount < 10, 'amount'] = 'other'" to make it so every zone on this data frame with less than 10 results was listed as "Other", instead of its name. Otherwise, the pie graph would have approximately 180 slices. However, this led to a final graph that showed "Other" as the most popular result with an overwhelming lead over the top zones, having hundreds of occurrences. This did not seem like a good way of presenting this data, so instead I kept it the way it was, and then when it was time to create the graph, I would restrict its data to the first 10 rows, giving me a pie graph with only 10 slices, instead of 186.

With this, I had a data frame that listed the number of occurrences of each zone, sorted in descending order. All I had to do was transform it into a pie graph.



	zone	amount
0	União de Freguesias do Centro, Porto	160
1	Braga	27
2	Vila Nova de Gaia	24
3	Bonfim, Porto	21
4	Viana do Castelo	20
...	...	...
181	São João da Pesqueira	1
182	Fão	1
183	Castrovicente	1
184	Morais	1
185	Durrães	1

186 rows × 2 columns

Figure 23 - Data frame with the number of occurences of each zone (dfnorthcounts).

With px.pie I created my pie chart, using the "amount" column of my data frame as the values and "zone" for the name of each element. With "[:10]", I restricted my data frame to only the first 10 rows. With ".update_traces(textinfo='value'), I added the amount of occurrences on each slice, and with ".update_layout" I added a title to my graph.

```
fig1_north = px.pie(dfnorthcounts[:10], values='amount',
names='zone')
fig1_north.update_traces(textinfo='value')
fig1_north.update_layout(title_text="Most frequent zones: North of
Portugal")
```

Listing 40 - Creating a pie chart (fig1_north).

Using "fig1_north.show()", the resulting graph appeared:

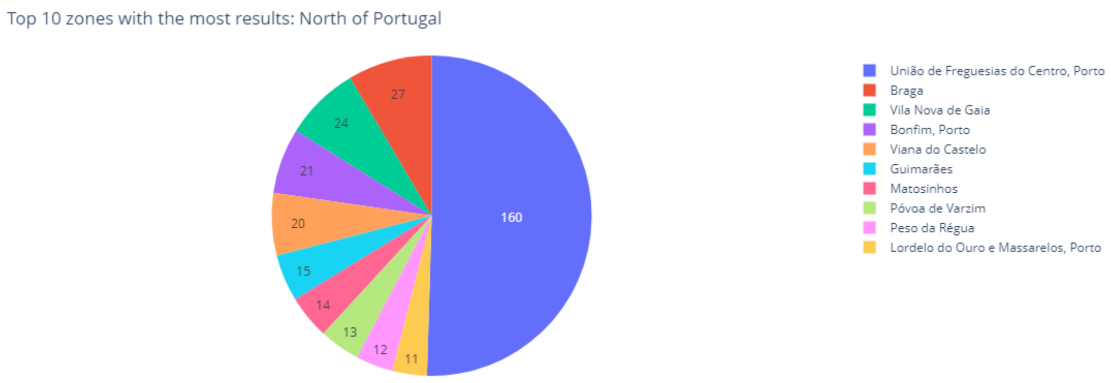


Figure 24 - Pie chart with the top 10 zones with the most results in the North of Portugal (fig1_north).

2. Average price per zone, in Euros

To represent the average price per zone, I used a bar graph. Since average values can be misleading when the sample size is too low, my objective was to present only the average values for the zones with the highest amount of results, which we observed on the first graph. In order to obtain the average price, I used the ".mean()" function that calculates the mean value for each zone, creating then a new data frame dfnorth_avg with contains a column with the average price in each zone.

```
dfnorth_avgg = dfnorth.groupby('zone')['price'].mean()
dfnorth_avg = pd.DataFrame(dfnorth_avgg)
dfnorth_avg = dfnorth_avg.reset_index()
```

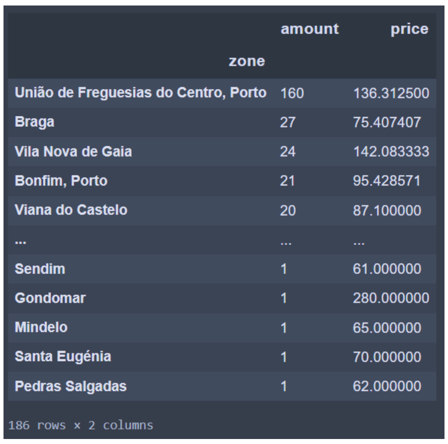
Listing 41 - Creating a new data frame that includes a column with the average price of each zone.

However, since this new data frame dfnorth_avg only contained the zone and average price columns, there was no way of relating that information to the column with the number of occurrences of each zone, as it was located in a different data frame. Therefore, I had to merge this new data frame with the other one created earlier (dfnorthcounts).

```
newdfnorth =
dfnorthcounts.set_index('zone').join(dfnorth_avg.set_index('zone'))
newdfnorth.sort_values(by=['amount'], inplace=True, ascending=False)
```

Listing 42 - Merging two dataframes.

By using ".sort_values", I obtained a data frame of average prices in descending order based on the number of occurrences.



	amount	price
zone		
União de Freguesias do Centro, Porto	160	136.312500
Braga	27	75.407407
Vila Nova de Gaia	24	142.083333
Bonfim, Porto	21	95.428571
Viana do Castelo	20	87.100000
...	...	...
Sendim	1	61.000000
Gondomar	1	280.000000
Mindelo	1	65.000000
Santa Eugénia	1	70.000000
Pedras Salgadas	1	62.000000

```
186 rows × 2 columns
```

Figure 25 - Data frame with the average prices of each zone, in descending order (newdfnorth).

With px.bar, I created the bar graph, restricting my data frame to zones with 10 or more occurrences, placing those zones on the x axis and the average price on the y axis:

```
fig2_north = px.bar(newdfnorth.loc[newdfnorth['amount'] > 9], x =
newdfnorth.loc[newdfnorth['amount'] > 9].index, y = 'price')
fig2_north.update_layout(title_text="Average price per zone (€):
North of Portugal")
```

Listing 43 - Creating a bar chart (fig2_north).

With fig2_north.show(), I obtained the following graph:

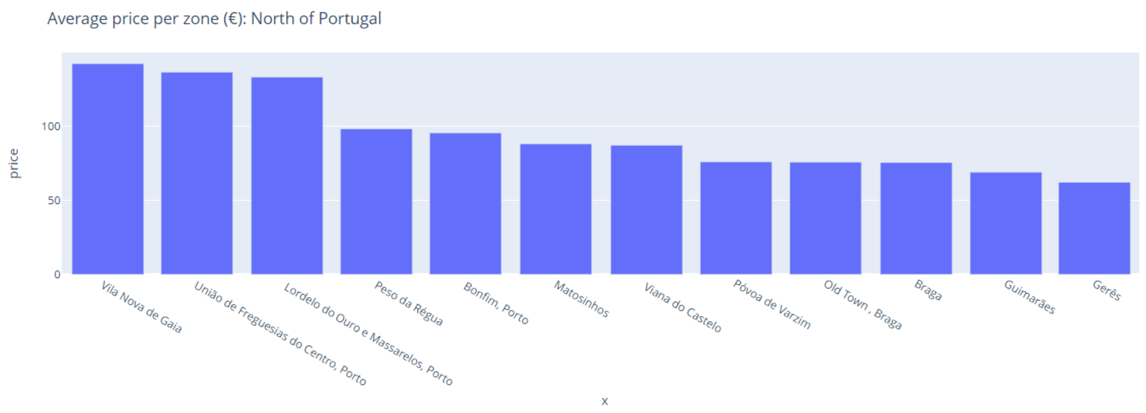


Figure 26 - Bar chart with the average price (€) per zone in the North of Portugal (fig2_north).

3. Top 10 most expensive results

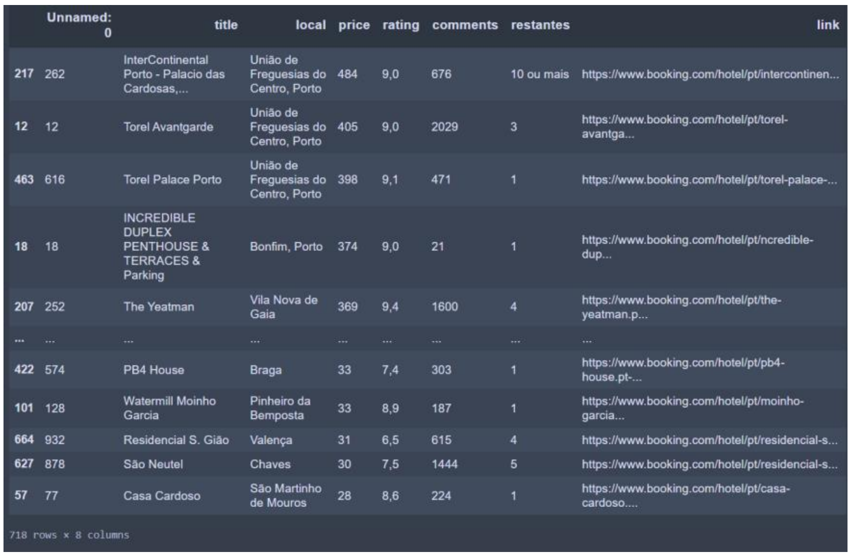
To represent the most expensive results, I figured a table would be the most informative way of presenting it visually, as I could also add more columns with more relevant information, such as the score and the number of comments. Those elements could be very relevant, as while sometimes there may be hotels that have ridiculously high prices, one cannot assume it is worth the price without proper feedback from previous users.

Beforehand, I created a copy of my original data frame, which I named dfnorth_aux.

```
dfnorth_aux = dfnorth.copy()
dfnorth_aux.sort_values(by=['price'], inplace=True, ascending=False)
```

Listing 44 - Creating a new data frame and sorting it by "price".

By sorting the values, based on the "price" column, in descending order, the following data frame was obtained:



	Unnamed: 0	title	local	price	rating	comments	restantes	link
217	262	InterContinental Porto - Palacio das Cardosas,...	União de Freguesias do Centro, Porto	484	9,0	676	10 ou mais	https://www.booking.com/hotel/pt/intercontinen...
12	12	Torel Avantgarde	União de Freguesias do Centro, Porto	405	9,0	2029	3	https://www.booking.com/hotel/pt/torel-avantga...
463	616	Torel Palace Porto	União de Freguesias do Centro, Porto	398	9,1	471	1	https://www.booking.com/hotel/pt/torel-palace-...
18	18	INCREDIBLE DUPLEX PENTHOUSE & TERRACES & Parking	Bonfim, Porto	374	9,0	21	1	https://www.booking.com/hotel/pt/ncredible-dup...
207	252	The Yeatman	Vila Nova de Gaia	369	9,4	1600	4	https://www.booking.com/hotel/pt/the-yeatman.p...
...	...	...	...	...	...	...	...	...
422	574	PB4 House	Braga	33	7,4	303	1	https://www.booking.com/hotel/pt/pb4-house.pt-...
101	128	Watermill Moinho Garcia	Pinheiro da Bemposta	33	8,9	187	1	https://www.booking.com/hotel/pt/moinho-garcia...
664	932	Residencial S. Gião	Valença	31	6,5	615	4	https://www.booking.com/hotel/pt/residencial-s...
627	878	São Neutel	Chaves	30	7,5	1444	5	https://www.booking.com/hotel/pt/residencial-s...
57	77	Casa Cardoso	São Martinho de Mouros	28	8,6	224	1	https://www.booking.com/hotel/pt/casa-cardoso....

```
718 rows × 8 columns
```

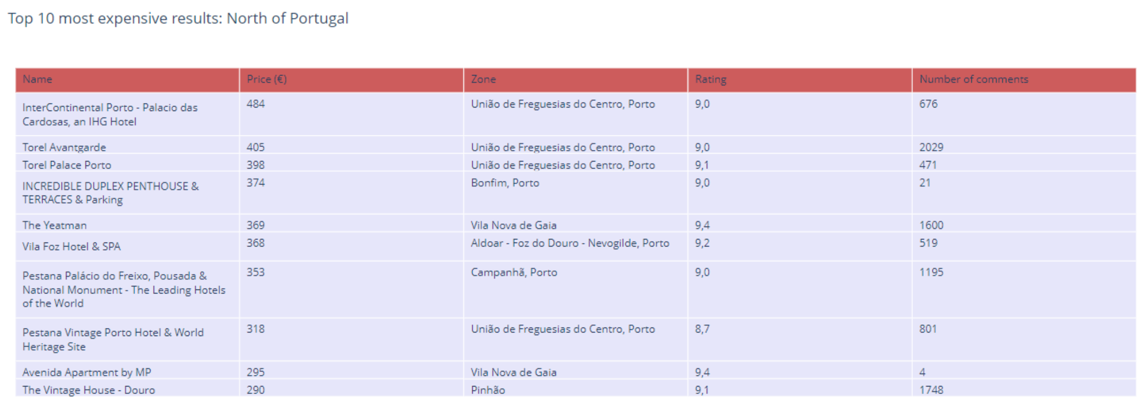
Figure 27 - Data frame for the North of Portugal, sorted by price, in descending order (dfnorth_aux).

Since the data frame is already in descending order, all I had to do was create the table using only the first 10 rows, with “[:10]”. Inside my “header”, I can define the title of each column, while inside “cells” I define the elements that go into each column, in the order I enter them.

```
fig3_north = go.Figure(data=[go.Table(
    header=dict(values=['Name', 'Price (€)', 'Zone', 'Rating',
'Number of comments'],
                fill_color='indianred',
                align='left'),
    cells=dict(values=[dfnorth_aux[:10].title,
dfnorth_aux[:10].price, dfnorth_aux[:10].local,
dfnorth_aux[:10].rating, dfnorth_aux[:10].comments],
               fill_color='lavender',
               align='left'))
])
fig3_north.update_layout(title_text="Top 10 most expensive results:
North of Portugal")
```

Listing 45 - Creating a table (fig3_north).

With “fig3_north.show()”, I was given the table for the top 10 most expensive results:


Top 10 most expensive results: North of Portugal

Name	Price (€)	Zone	Rating	Number of comments
InterContinental Porto - Palacio das Cardosas, an IHG Hotel	484	União de Freguesias do Centro, Porto	9,0	676
Torel Avantgarde	405	União de Freguesias do Centro, Porto	9,0	2029
Torel Palace Porto	398	União de Freguesias do Centro, Porto	9,1	471
INCREDIBLE DUPLEX PENTHOUSE & TERRACES & Parking	374	Bonfim, Porto	9,0	21
The Yeatman	369	Vila Nova de Gaia	9,4	1600
Vila Foz Hotel & SPA	368	Aldoar - Foz do Douro - Nevogilde, Porto	9,2	519
Pestana Palácio do Freixo, Pousada & National Monument - The Leading Hotels of the World	353	Campanhã, Porto	9,0	1195
Pestana Vintage Porto Hotel & World Heritage Site	318	União de Freguesias do Centro, Porto	8,7	801
Avenida Apartment by MP	295	Vila Nova de Gaia	9,4	4
The Vintage House - Douro	290	Pinhão	9,1	1748

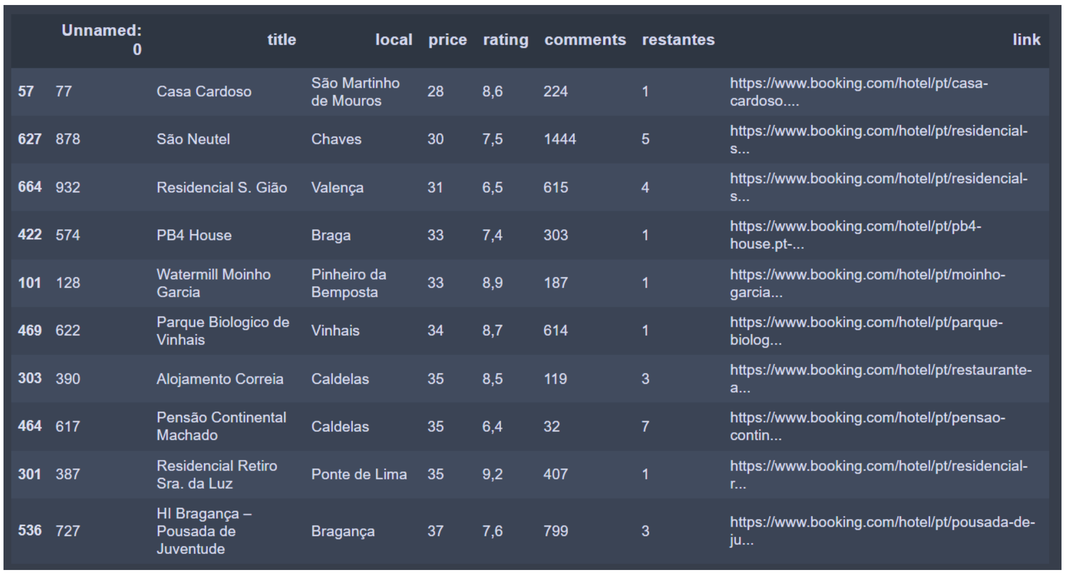
Figure 28 - Table with the top 10 most expensive results in the North of Portugal (fig3_north).

4. Top 10 cheapest results

Following the same method as before, I created a table for the cheapest results. Initially, I used the same data frame created for the most expensive results, and instead of capturing the first 10 rows, I captured the last 10. However, I encountered a problem: since the rows were in descending order, the first element of my new data frame was, in fact, the 10th cheapest result, instead of the cheapest. This could have been easily fixed by altering the "ascending" attribute of the dfnorth_aux data frame from "False" to "True", but since that data frame was also being used for the previous table, I did not change it, as it would change the results of my previous table. Instead, the solution was simple, I only had to create a new copy of the original data frame dfnorth and sort it by price values again, but this time in ascending order. However, simple is boring, so instead I created a copy of my previous copy, captured the last 10 results with "[-10:]" and inverted their order, thus obtaining the correct results in the correct order.

```
dfnorth_auxx = dfnorth_aux[-10:]
dfnorth_auxx.sort_values(by=['price'], inplace=True, ascending=True)
```

Listing 46 - Creating a copy of my previous data frame, capturing only the last 10 rows.



	Unnamed: 0	title	local	price	rating	comments	restantes	link
57	77	Casa Cardoso	São Martinho de Mouros	28	8,6	224	1	https://www.booking.com/hotel/pt/casa-cardoso....
627	878	São Neutel	Chaves	30	7,5	1444	5	https://www.booking.com/hotel/pt/residencial-s...
664	932	Residencial S. Gião	Valença	31	6,5	615	4	https://www.booking.com/hotel/pt/residencial-s...
422	574	PB4 House	Braga	33	7,4	303	1	https://www.booking.com/hotel/pt/pb4-house.pt-...
101	128	Watermill Moinho Garcia	Pinheiro da Bemposta	33	8,9	187	1	https://www.booking.com/hotel/pt/moinho-garcia...
469	622	Parque Biologico de Vinhais	Vinhais	34	8,7	614	1	https://www.booking.com/hotel/pt/parque-biolog...
303	390	Alojamento Correia	Caldelas	35	8,5	119	3	https://www.booking.com/hotel/pt/restaurante-a...
464	617	Pensão Continental Machado	Caldelas	35	6,4	32	7	https://www.booking.com/hotel/pt/pensao-contin...
301	387	Residencial Retiro Sra. da Luz	Ponte de Lima	35	9,2	407	1	https://www.booking.com/hotel/pt/residencial-r...
536	727	HI Bragança – Pousada de Juventude	Bragança	37	7,6	799	3	https://www.booking.com/hotel/pt/pousada-de-ju...

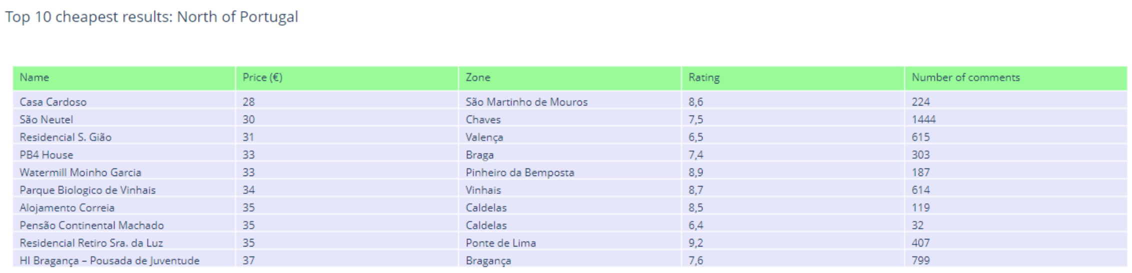
Figure 29 - Data frame for the North of Portugal with only the first 10 rows, sorted by price, in ascending order (dfnorth_auxx).

Once again, I used “header” to define the title of the columns and “cells” to define their content.

```
fig4_north = go.Figure(data=[go.Table(
    header=dict(values=['Name', 'Price (€)', 'Zone', 'Rating',
'Number of comments'],
                fill_color='palegreen',
                align='left'),
    cells=dict(values=[dfnorth_auxx.title, dfnorth_auxx.price,
dfnorth_auxx.local, dfnorth_auxx.rating, dfnorth_auxx.comments],
               fill_color='lavender',
               align='left'))
])
fig4_north.update_layout(title_text="Top 10 cheapest results: North
of Portugal")
```

Listing 47 - Creating a table (fig4_north).

With “fig4_north.show()”, I finally obtained the table for the top 10 cheapest results:



Top 10 cheapest results: North of Portugal

Name	Price (€)	Zone	Rating	Number of comments
Casa Cardoso	28	São Martinho de Mouros	8,6	224
São Neutel	30	Chaves	7,5	1444
Residencial S. Gião	31	Valença	6,5	615
PB4 House	33	Braga	7,4	303
Watermill Moinho Garcia	33	Pinheiro da Bemposta	8,9	187
Parque Biologico de Vinhais	34	Vinhais	8,7	614
Alojamento Correia	35	Caldelas	8,5	119
Pensão Continental Machado	35	Caldelas	6,4	32
Residencial Retiro Sra. da Luz	35	Ponte de Lima	9,2	407
HI Bragança – Pousada de Juventude	37	Bragança	7,6	799

Figure 30 - Table with the top 10 cheapest results in the North of Portugal (fig4_north).

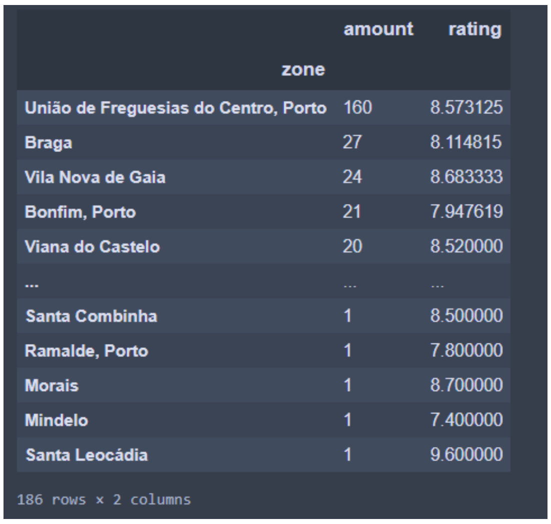
5. Average score per zone

To obtain the average score per zone, the method is similar to the one used in graph 2. Instead of getting the average price, I am getting the average score, using ".mean()" while grouping the results by zone.

```
dfnorth_scoreavgg = dfnorth.groupby('zone')['rating'].mean()
dfnorth_scoreavg = pd.DataFrame(dfnorth_scoreavgg)
dfnorth_scoreavg = dfnorth_scoreavg.reset_index()
newdfnorthavg =
dfnorthcounts.set_index('zone').join(dfnorth_scoreavg.set_index('zon
e'))
newdfnorthavg.sort_values(by=['amount'], inplace=True,
ascending=False)
```

Listing 48 - Obtaining a new data frame with the average score of each zone, and then merging it into another data frame.

Using ".join" to merge my average score data frame into "dfnorthcounts" (created on graph 2), which includes the amount of occurrences on each zone, gives me the following data frame:



	amount	rating
zone		
União de Freguesias do Centro, Porto	160	8.573125
Braga	27	8.114815
Vila Nova de Gaia	24	8.683333
Bonfim, Porto	21	7.947619
Viana do Castelo	20	8.520000
...	...	...
Santa Combinha	1	8.500000
Ramalde, Porto	1	7.800000
Morais	1	8.700000
Mindelo	1	7.400000
Santa Leocádia	1	9.600000

```
186 rows × 2 columns
```

Figure 31 - Merged data frame that includes the amount of occurrences of each zone and their respective ratings (newdfnorthavg).

Once again, I create our bar chart using "px.bar", restricting my results to only zones with 10 or more results. With "text_auto", the value of each zone's average score appears over their respective bars, while "orientation='h'" changes this bar chart's orientation from vertical to horizontal.

```
fig5_north = px.bar(newdfnorthavg.loc[newdfnorthavg['amount'] > 9],
y = newdfnorthavg.loc[newdfnorthavg['amount'] > 9].index, x =
'rating', text_auto=True, orientation='h')
fig5_north.update_layout(title_text="Average score por zone: North
of Portugal")
```

Listing 49 - Creating a horizontal bar chart (fig5_north).

Using “fig5_north.show()” gives me the horizontal bar chart with the average score per zone:

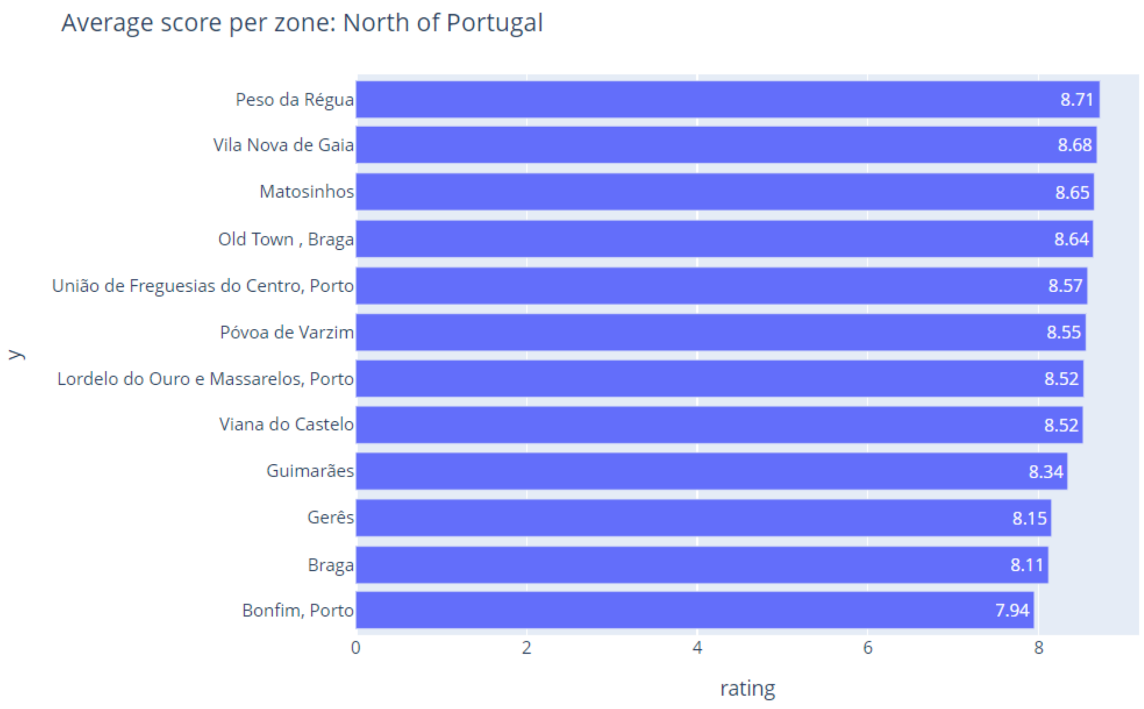


Figure 32 - Horizontal bar chart with the average scores per zone in the North of Portugal (fig5_north).

With all graphs created for this specific dataset, I then proceeded to repeat the same process for every other dataset of other zones and time periods. There is no need to describe that process, as it was simply an extensive sequence of copying, pasting and renaming codes.

Afterwards, I switched to the dataset obtained from my third extraction, that collected data from the users’ comments on Booking’s “Melia Braga Hotel & Spa” page.

As before, I started by creating my main data frame with the info from the dataset:

```
import pandas as pd
df = pd.read_excel('Melia comments.xlsx')
```

Listing 50 - Transforming my comments' dataset into a data frame.

6. Most frequent nationalities in user's comments;

For this graph, I made a pie graph to present the most frequent nationalities. Just like in the first graph, I had the problem of having many elements with a low number of occurrences, so I restricted this graph to countries with 20 or more comments. In order to avoid repetitiveness, I used a different method than the one used on the first graph: instead of restricting the data frame appearing on the graph to its first rows, I created a dictionary that includes the number of comments for each country, and then used a "for loop" to remove the countries that do not have more than 20 occurrences.

Firstly, I created a dictionary of countries, which I named "countries".

```
countries = dict()
for country in df.Country:
     if country in countries:
          countries[country] = countries[country]+1
     else:
          countries[country] = 1
```

Listing 51 - Creating a dictionary of countries.

Here, I was instructing my program to fetch the elements in the "Country" column of my data frame. For each element, if that country was already in my dictionary, it would add 1 to that country's count. If it was not, then it would set that country's count as 1, as it was the first occurrence. When the loop ended, the dictionary would have the exact count of occurrences for each country.

Afterwards, I created three lists, one with countries, another with their count, and the last one for the countries to remove from my dictionary, starting as an empty list. Firstly, I used a "for loop" that adds the countries to my third list if their count is lower than 20. Then, with another "for loop", I deleted each country with less than 20 occurrences from my initial dictionary.

```
import plotly.graph_objs as go
countrylist = list(countries.keys())
valuelist = list(countries.values())
countriestoremove = []

for x in countrylist:
      if countries[x] < 20:
             countriestoremove.append(str(x))

for y in countriestoremove:
      del countries[y]
```

Listing 52 - Creating two lists and two "for loops" that remove countries with less than 20 occurrences.

Then, using my newly created lists that no longer include the countries with a low number of occurrences, I created my pie chart, using “update_traces” to include the actual values instead of percentages, as percentages can be misleading, since I had removed part of my results.

```
fig6 = go.Figure(data=[go.Pie(labels=countrylist, values=valuelist,
hole=.3)])
fig6.update_traces(textinfo='value')
```

Listing 53 - Creating a pie chart (fig6).

With “fig6.show()”, I obtained graph for the most frequent nationalities:

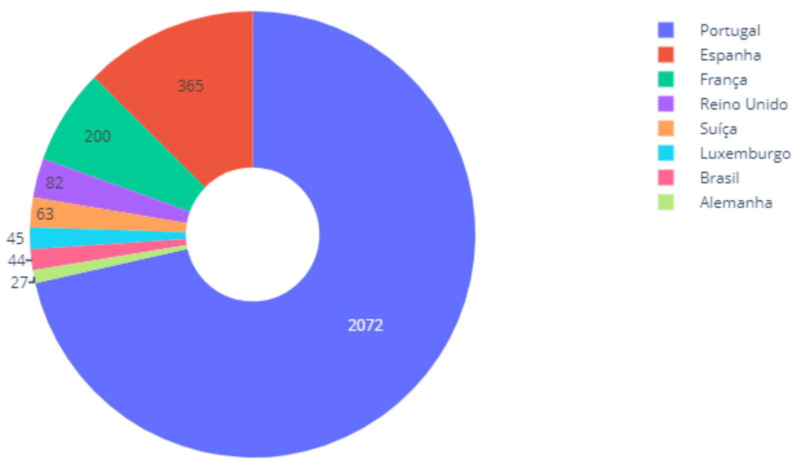


Figure 33 - Most frequent nationalities in Melia Braga Hotel & Spa comments section.

7. Positive comments' word cloud

Before delving into the word clouds, I had to divide the comments based on their score. On a scale from 0 to 10, I considered comments with a 5 or higher as positive, and comments lower than 5 as negative. In order to do so, I created two copies of my original data frame, using ".loc" to divide them based on the score. "dfpositive" will only have comments with a score of 5 or higher, while "dfnegative" will only have comments with scores lower than 5.

```
dfpositive = df.copy()
dfnegative = df.copy()
dfpositive.loc[dfpositive['Score'] > 4.9]
dfnegative.loc[dfnegative['Score'] < 5]
```

Listing 54 - Dividing my main data frame into two, one for positive comments and the other for negative comments.

To create a word cloud, I needed to remove the stop words, which are the most common words that do not describe the content of the phrase. They are usually determiners, coordinating conjunctions and prepositions, such as “the”, “a”, “for”, “but”, “in”, among many others. Since Portugal, Spain, France and the UK were the most frequent countries in the comments, I removed the Portuguese, Spanish, French and English stop words, with the assistance of the stopwords package in NLTK.

```
import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud
import matplotlib.pyplot as plt

stopwords = set(nltk.corpus.stopwords.words('portuguese')) |
set(nltk.corpus.stopwords.words('spanish')) |
set(nltk.corpus.stopwords.words('french')) |
set(nltk.corpus.stopwords.words('english'))
```

Listing 55 - Removing stopwords from my results.

Then, I created a string called “text1”, to which each positive comment will be added, using “.join”.

```
text1 = " ".join(str(review) for review in dfpositive.Comment)
```

Listing 56 - Creating a string which will contain all positive comments.

To create my wordcloud, I used “WordCloud”, with “.generate” receiving the text1 string created earlier. With “plt.subplots”, I resized my figure, while “plt.axis(“off”)” removed the x and y axes.

```
positivewordcloud = WordCloud(width = 800, height = 600,
background_color = 'white', stopwords=stopwords).generate(text1)
plt.subplots(figsize = (8,8))
plt.imshow(wordcloudpos, interpolation='bilinear')
plt.axis("off")
```

Listing 57 - Creating a word cloud with positive comments.

Using “plt.show()” gave me the word cloud for the most frequent words in positive comments:

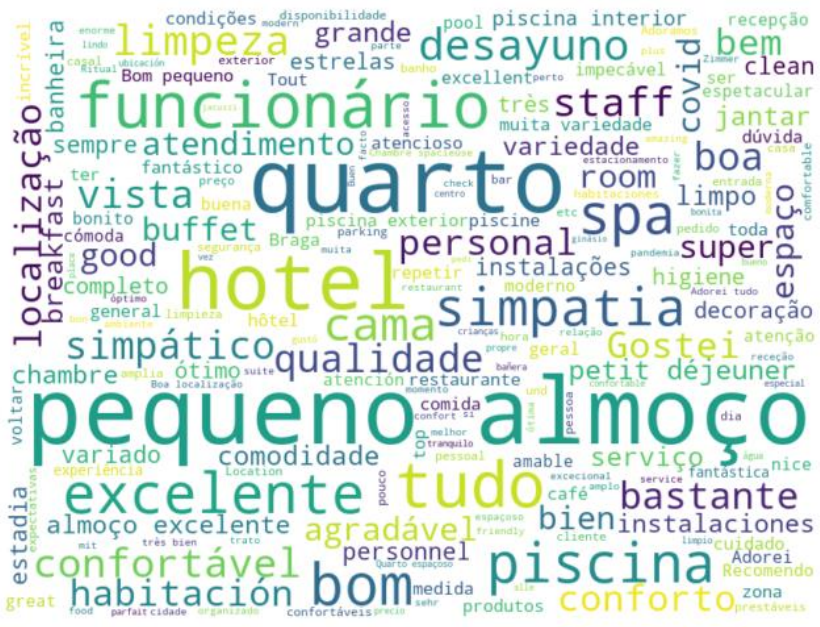


Figure 34 - Word cloud for positive comments in Melia Braga Hotel & Spa comments section.

8. Negative comments' word cloud

To create the word cloud for the negative comments, the process was exactly the same as the previous graph. This time, instead, I used the data frame for the comments classified as negative.

```
Text2 = " ".join(str(review) for review in dfpositive.Comment)
```

Listing 58 - Creating a string that will contain negative comments.

To differentiate this graph from the previous one, I used "colormap" to change the colour scheme, giving it warmer colours that resemble negativity, as opposed to the green colour scheme.

```
negativewordcloud = WordCloud(width = 800, height = 600, colormap =
'inferno', background_color = 'white',
stopwords=stopwords).generate(text2)
plt.subplots(figsize = (8,8))
plt.imshow(negativewordcloud, interpolation='bilinear')
plt.axis("off")
```

Listing 59 - Creating a word cloud with negative comments.

With "plt.show()", I obtained the word cloud for the most frequent words in negative comments:

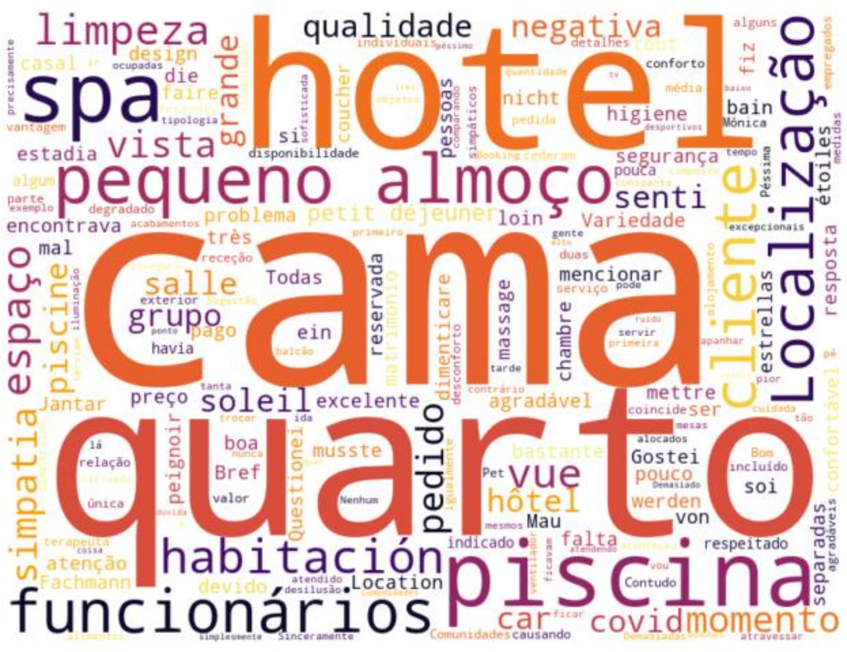


Figure 35 - Word cloud for negative comments in Melia Braga Hotel & Spa comments section.

Having finished that, I was approaching the end of my tasks regarding data visualisation, with only one last step remaining: creating a platform on which I could present those graphs, without having to use a Python IDE.

4.4.3. INTERACTIVE DASHBOARD

In order to make my graphs easily accessible, I decided to create a platform on which they could not only be viewed, but also manipulated, thanks to their dynamic nature (brought by the Plotly library). After some research, I decided that Dash would be the best option, as it is even developed and recommended by Plotly itself. With Dash[7], I can create an interface that is then accessible through my browser, within a local offline server.

After familiarising myself with the framework, I realised that with my HTML and CSS skills, I could create a very decent data interface on Dash. My goal was to be able to present graphs from every zone, with a dropdown menu that allowed me to choose which zone's data I wanted to see.

[7] https://dash.plotly.com/

Firstly, I imported the necessary libraries and packages.

```
import plotly.graph_objects as go
import dash
from dash import dcc
from dash import html
```

Listing 60 - Importing libraries and packages.

Before continuing, it is worth noting that I created this interface on the same Jupyter Notebook .ipynb file as the one used previously to create the first 5 graphs, which will be included in this dashboard. This means that all data frames created are still accessible, and all the libraries imported are still active. This includes Pandas, which is important for Dash, as Plotly depends on it to create graphs.

I start creating my dashboard by creating my app, which will also invoke an external CSS file that I made to improve the interface's appearance. With "app.title", I defined the title of the page, which will appear on the browser tab.

```
external_stylesheets = [
    {
        "href": "https://fonts.googleapis.com/css2?"
        "family=Lato:wght@400;700&display=swap",
        "rel": "stylesheet",
    },
]
app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
app.title = "Booking.com: data visualisation of extracted data"
```

Listing 61 - Creating my app and adding an external CSS file.

Within "app.layout" I inserted the code that will define the structure of my page. Although the code is structured in Python, anyone with HTML experience can understand what each line means. I began by creating a title and a sub-title for this page (bear in mind that from now on, every piece of coding presented is included inside "app.layout".

```
app.layout = html.Div(children=[
html.H1(children='BOOKING.COM: NORTH OF PORTUGAL, GALICIA AND CENTER
OF PORTUGAL', style={'text-align': 'center'}, className="header"),
html.Div(children='Data visualisation of extracted data',
style={'text-align': 'center'}, className="header-description"),
```

Listing 62 - Creating the main layout of my app.

I also added CSS attributes and a class for each element (that would be customised by the CSS external file). Before adding my graphs, I created a new html.Div for a dropdown menu, on which I could select between North, Galicia and Center of Portugal. Firstly, I had to add a unique ID (‘dropdown’) that will be used to invoke this element later, when I create the function that makes the graphs switch based on the zone selected. Inside “options”, I used “label” to define the text appearing in each option, while “value” defines its actual value, which will also be invoked later. After “options”, “value” sets the default position for the dropdown menu. In this case, I want my North graphs to appear by default, so I chose “north1”.

```
html.Label(['Choose your zone:'],style={'font-weight': 'bold',
'text-align': 'center'}, className="choose"),
        dcc.Dropdown(
            id='dropdown1',
            options=[
                {'label': 'North', 'value': 'north1'},
                {'label': 'Galicia', 'value': 'galicia1'},
                {'label': 'Center', 'value': 'center1'},
                    ],
            value='north1',
            style={"width": "60%", 'display': 'block', 'margin-
left': 'auto', 'margin-right': 'auto'}, className="drop"),
```

Listing 63 - Creating the layout for my first dropdown menu.

Afterwards, I proceeded to add my graphs. Normally, one would create the graphs at this moment, using the same libraries we have used before. However, since my graphs had already been created before, I could just invoke them using the names I had defined for them (fig1_north, fig2_north, fig3_north, fig4_north and fig5_north). With “dcc.Graph”, inside another html div, I created the element on which my first graph will appear. Instead of using the name of the first graph right away, I instead gave it a specific ID. That ID will be invoked later in a conditional statement, within a function that will define from which zone the presented graph should be. Once again, I used "style" to define its appearance.

```
html.Div([dcc.Graph(
            id='g1',
            className="card",
            style={"width": "75%", 'display': 'block', 'margin-
left': 'auto', 'margin-right': 'auto'})
         ])
```

Listing 64 - Creating the layout for my first graph.

As I had 5 unique graphs, I repeated the code above 4 more times, giving unique IDs to each graph (g2, g3, g4 and g5) and to each dropdown menu (dropdown2, dropdown3, dropdown4 and dropdown5). I repeated the dropdown menus because I wanted to be able to compare graphs of different zones, instead of using one dropdown to change every graph at the same time. This concludes the layout portion of this section.

Since I wanted my graphs to change depending on the options I chose, I needed two things:

1. Five callback functions, one for each graph and dropdown menu.
2. A function that will define the value of my graph, with "if" statements that will cover every possible graph from every zone.

Whenever an input property changes, which is the case when we select a dropdown menu option, the callback function will provide Dash with the new property value, updating the output element. For instance, the callback function for the first graph and dropdown menu will be the following:

```
@app.callback(
    Output('g1', 'figure'),
    Input('dropdown1', 'value'))
```

Listing 65 - Creating a callback function.

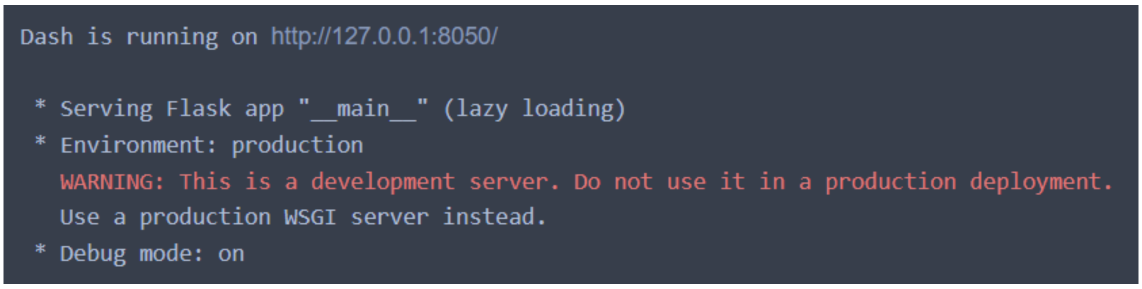
Its input will be the value of "dropdown1". "dropdown1" is the ID I set for the first dropdown menu, while its value will either be "north1", "galicia1" or "center1". The output will be "g1", which is the unique ID I set for the first graph. This function applies our dropdown selection to the first graph. Following that, I created four more callback functions, each invoking the corresponding remaining four graphs and dropdown menus.

Afterwards, I finally defined the function that updates the graph shown on each "dcc.Graph" element. I named it "update_figure", and its outcome depended on the value it received, which was the value defined on the dropdown menus (either North, Galicia or Center).

```
def update_figure(value):
    if value == 'north1':
        figure=fig1_north
        return figure
    if value == 'galicia1':
        figure=fig1_galicia
        return figure
    if value == 'center1':
        figure=fig1_center
        return figure
```

Listing 66 - Function that will change the graphs shown based on the input given to the dropdown menu.

Here, I recalled the graphs I created earlier, using "if" statements that defined which graph will appear, based on the input entered on the dropdown menu. However, this code altered only the first graph, and my page structure included five graphs. Therefore, I repeated the "if" statements above four more times, using the proper IDs and graph names. After finishing my function, the Dash code was concluded with "app.run_server(debug=True, use_reloader=False)", which creates the local server on which my Dash platform will run. I entered "use_reloader=False" because I was using Jupyter Notebook, and Dash does not work without that option disabled.

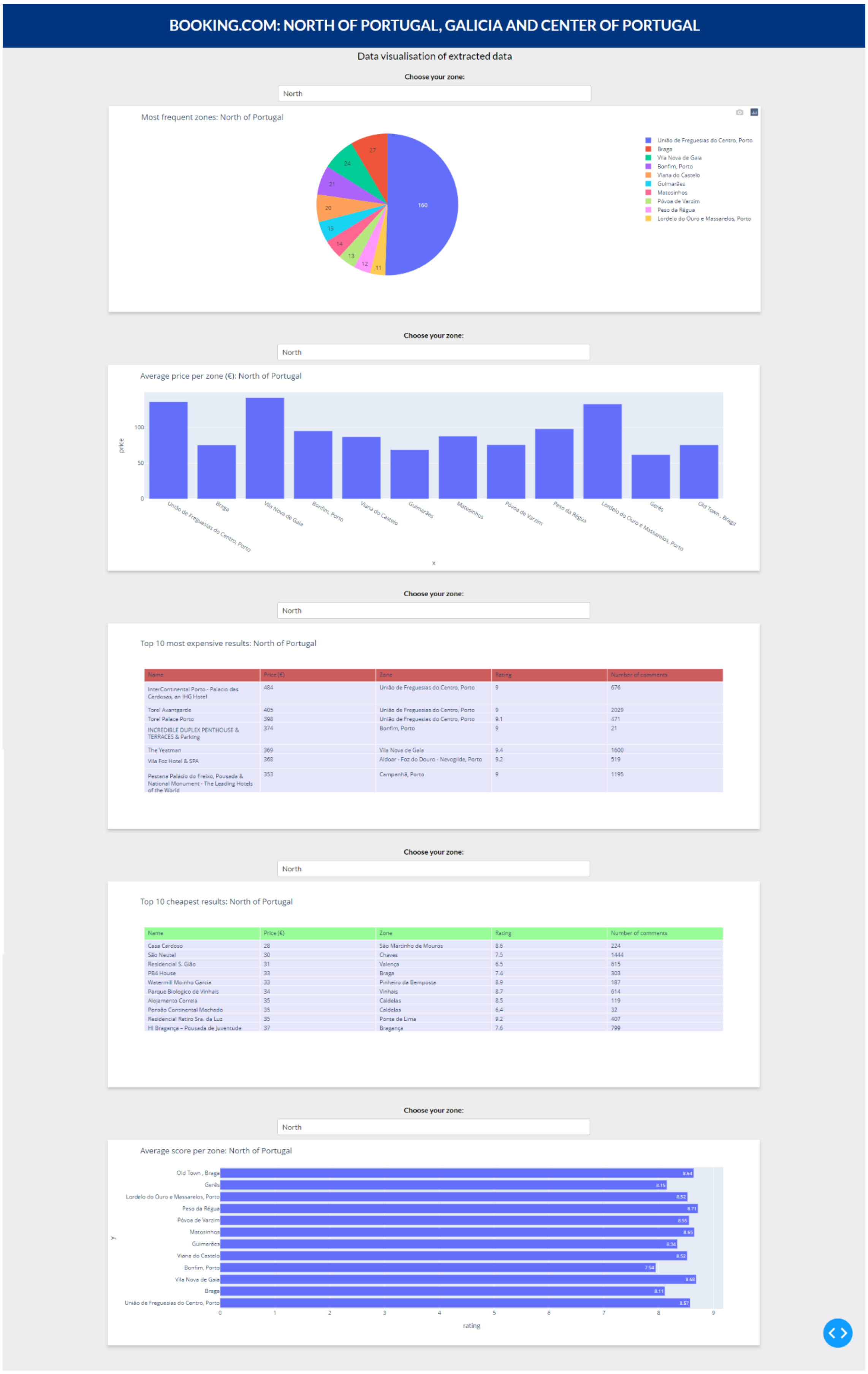


```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
```

Figure 36 - Dash output on Jupyter Notebook after running the code. The URL takes me to the local server on which the dashboard runs.

Clicking on the URL opened my default browser, taking me to the platform I had created.


BOOKING.COM: NORTH OF PORTUGAL, GALICIA AND CENTER OF PORTUGAL
Data visualisation of extracted data
Choose your zone:
North
Most frequent zones: North of Portugal
160
27
24
21
20
15
14
13
12
11
União de Freguesias do Centro, Porto
Braga
Vila Nova de Gaia
Bonfim, Porto
Viana do Castelo
Guimarães
Matosinhos
Póvoa de Varzim
Peso da Régua
Lordelo do Ouro e Massarelos, Porto
Choose your zone:
North
Average price per zone (€): North of Portugal
price
0
50
100
União de Freguesias do Centro, Porto
Braga
Vila Nova de Gaia
Bonfim, Porto
Viana do Castelo
Guimarães
Matosinhos
Póvoa de Varzim
Peso da Régua
Lordelo do Ouro e Massarelos, Porto
Gerês
Old Town , Braga
x
Choose your zone:
North
Top 10 most expensive results: North of Portugal

Name	Price (€)	Zone	Rating	Number of comments
InterContinental Porto - Palacio das Cardosas, an IHG Hotel	484	União de Freguesias do Centro, Porto	9	676
Torel Avantgarde	405	União de Freguesias do Centro, Porto	9	2029
Torel Palace Porto	398	União de Freguesias do Centro, Porto	9.1	471
INCREDIBLE DUPLEX PENTHOUSE & TERRACES & Parking	374	Bonfim, Porto	9	21
The Yeatman	369	Vila Nova de Gaia	9.4	1600
Vila Foz Hotel & SPA	368	Aldoar - Foz do Douro - Nevogilde, Porto	9.2	519
Pestana Palácio do Freixo, Pousada & National Monument - The Leading Hotels of the World	353	Campanhã, Porto	9	1195

Name	Price (€)	Zone	Rating	Number of comments
Casa Cardoso	28	São Martinho de Mouros	8.6	224
São Neutel	30	Chaves	7.5	1444
Residencial S. Gião	31	Valença	6.5	615
PB4 House	33	Braga	7.4	303
Watermill Moinho Garcia	33	Pinheiro da Bemposta	8.9	187
Parque Biologico de Vinhais	34	Vinhais	8.7	614
Alojamento Correia	35	Caldelas	8.5	119
Pensão Continental Machado	35	Caldelas	6.4	32
Residencial Retiro Sra. da Luz	35	Ponte de Lima	9.2	407
HI Bragança – Pousada de Juventude	37	Bragança	7.6	799

Figure 37 - Dashboard that presents all created graphs.

By default, the platform shows data for the North datasets. However, by using the dropdown menus, the graphs change based on my input.

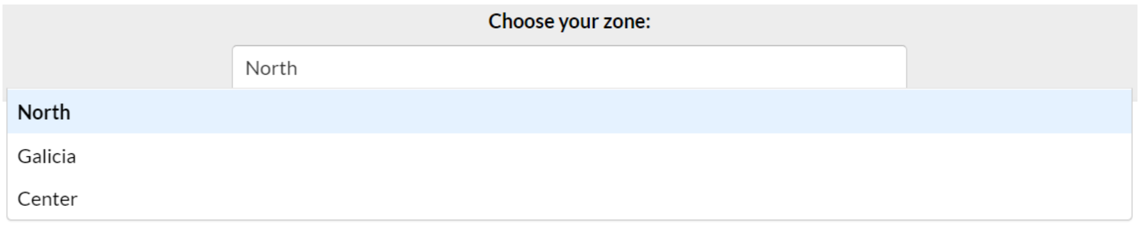


Figure 38 - Dropdown menu that allows selection of zone for each graph.

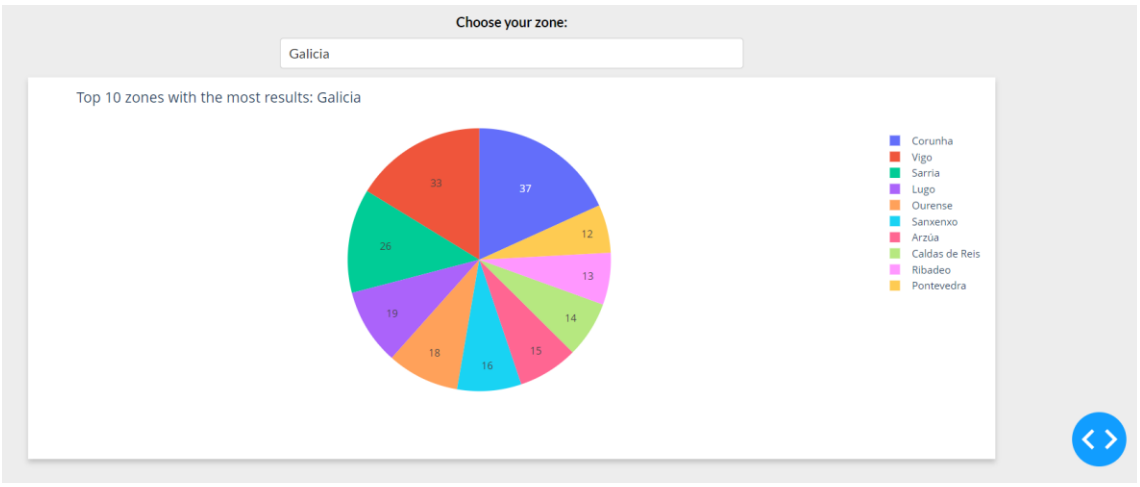


Figure 39 - Top 10 zones with the most results in Galicia, selected on the dropdown menu.

Besides being able to observe all graphs, I could also interact with them. For instance, on pie graphs I could present only the elements that I wanted to see, by double clicking on the names on the right side of the screen to enter "selection mode", and then choosing the zones that I want to compare. For instance, I selected Corunha, Vigo, Sanxenxo and Pontevedra and obtained the following result:

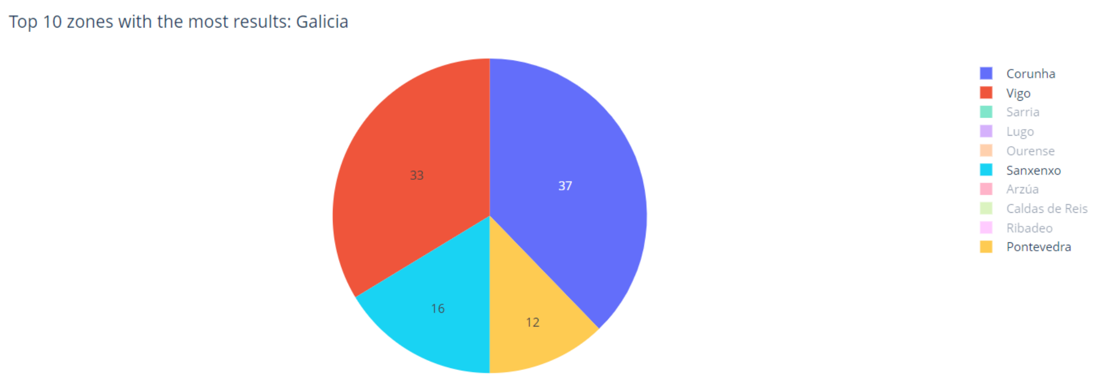


Figure 40 - Top 10 zones with the most results in Galicia, with filtered selections, in Dash.

On bar graphs, I could mouseover each bar to see the exact value of that specific bar. Besides that, I could also click and select a certain area of the graph to zoom. For example, by zooming on the first three bars, we obtain the following bar chart:

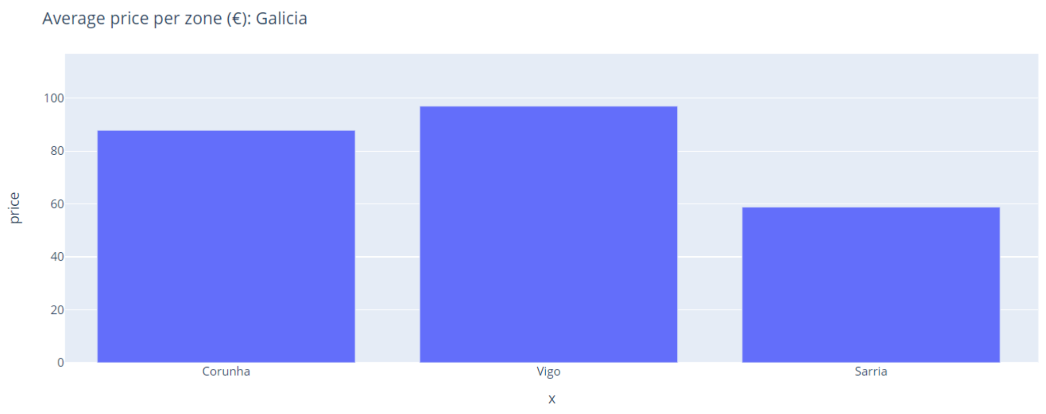


Figure 41 - Average prices per zone in Galicia, with filtered selections, in Dash.

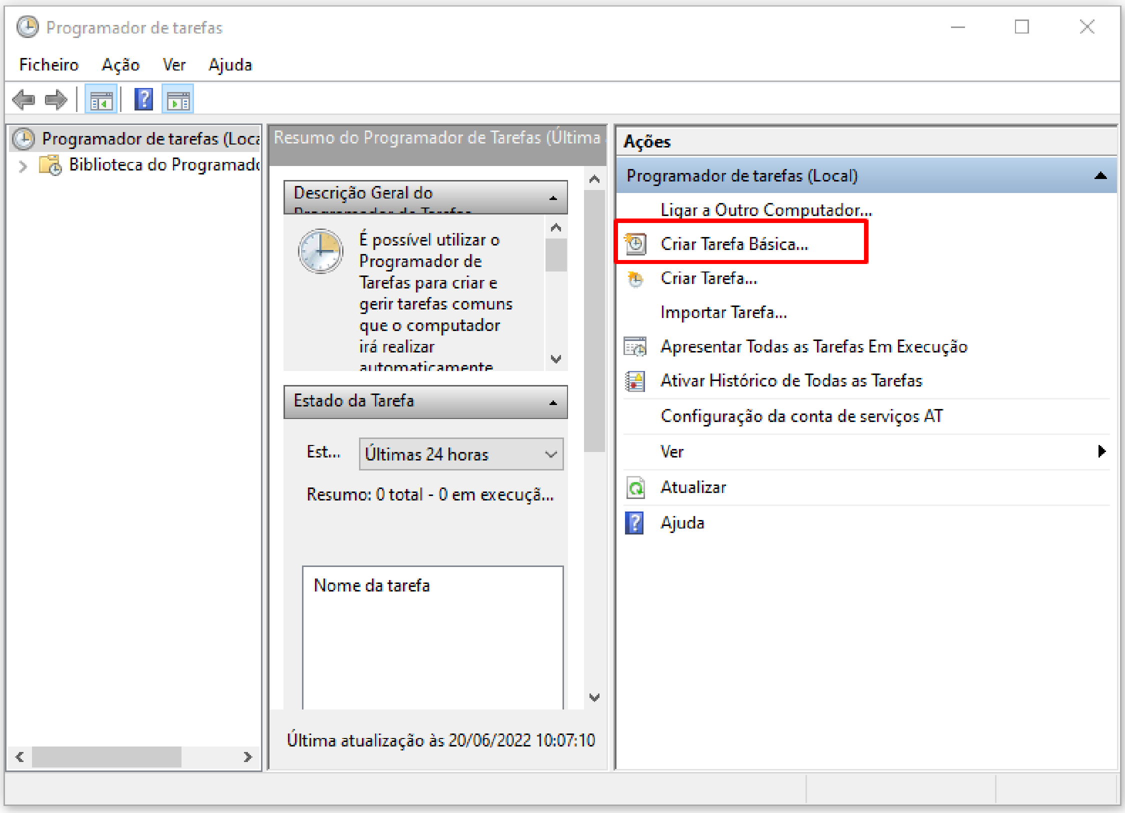
Although not very useful in this case, with tables I could also select and drag each column to move it to any position I wanted.

Moreover, every graph included a button in the upper right corner that allowed me to instantly save that graph on my computer, in PNG format.

4.5. AUTOMATION

I had a fully working script that extracted everything I wanted, from every zone and time period. I also had an interface to select those extractions manually. Besides that, I had a script that received a dataset and created graphs with the received data, and a script for a platform that displayed those graphs in a dynamic and interactive interface. Then, I just needed to automate this process, scheduling my code, for instance, to run daily, at 8am. I wanted to be able to, without having to lift a finger, extract data from the three zones and obtain the corresponding datasets, and then, from those datasets, create graphs that would be automatically saved on my folder. When I started this project, I thought that this was going to be the hardest of all the tasks, but, in fact, it was the easiest. All I had to do was save my programs in .py format, and then use Windows Task Scheduler to define when to run those .py programs. The hardest part, which was creating the programs, had already been done.

After opening Windows Task Scheduler, one needs only to click on "Create basic task".



Programador de tarefas
Ficheiro
Ação
Ver
Ajuda
Programador de tarefas (Loca
Biblioteca do Programado
Resumo do Programador de Tarefas (Última
Descrição Geral do
Estado da Tarefa
Est...
Últimas 24 horas
Nome da tarefa
Ações
Programador de tarefas (Local)
Ligar a Outro Computador...
Criar Tarefa Básica...
Criar Tarefa...
Importar Tarefa...
Apresentar Todas as Tarefas Em Execução
Ativar Histórico de Todas as Tarefas
Configuração da conta de serviços AT
Ver
Atualizar
Ajuda

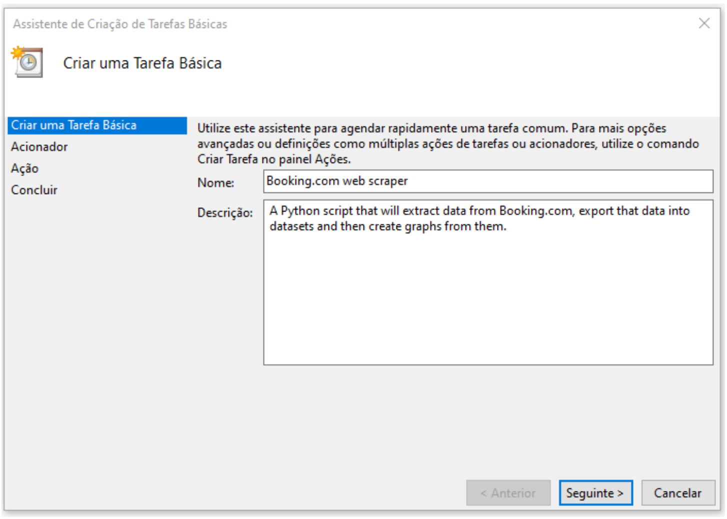
É possível utilizar o Programador de Tarefas para criar e gerir tarefas comuns que o computador irá realizar automaticamente

Resumo: 0 total - 0 em execuçã...

Última atualização às 20/06/2022 10:07:10

Figure 42- Windows Task Scheduler interface

After giving it a suitable title and description, I clicked "Next", allowing me then to select the periodicity on which my program would run.


Assistente de Criação de Tarefas Básicas

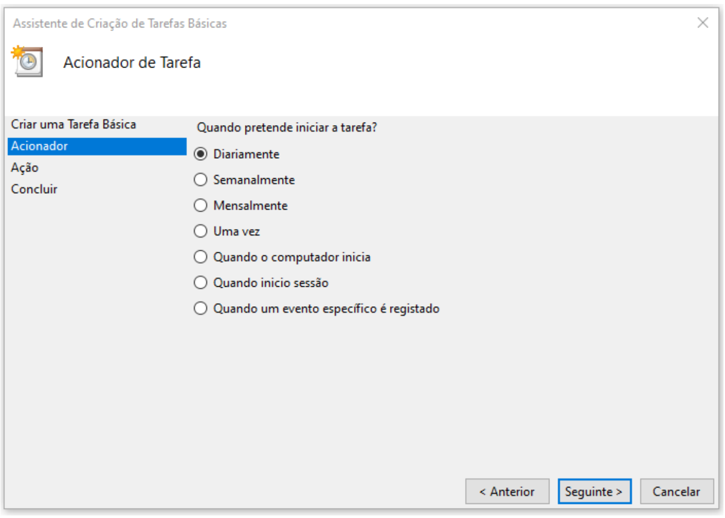
Criar uma Tarefa Básica

Utilize este assistente para agendar rapidamente uma tarefa comum. Para mais opções avançadas ou definições como múltiplas ações de tarefas ou acionadores, utilize o comando Criar Tarefa no painel Ações.

Nome: Booking.com web scraper

Descrição: A Python script that will extract data from Booking.com, export that data into datasets and then create graphs from them.

Figure 43 - Windows Task Scheduler: naming my task.


Assistente de Criação de Tarefas Básicas

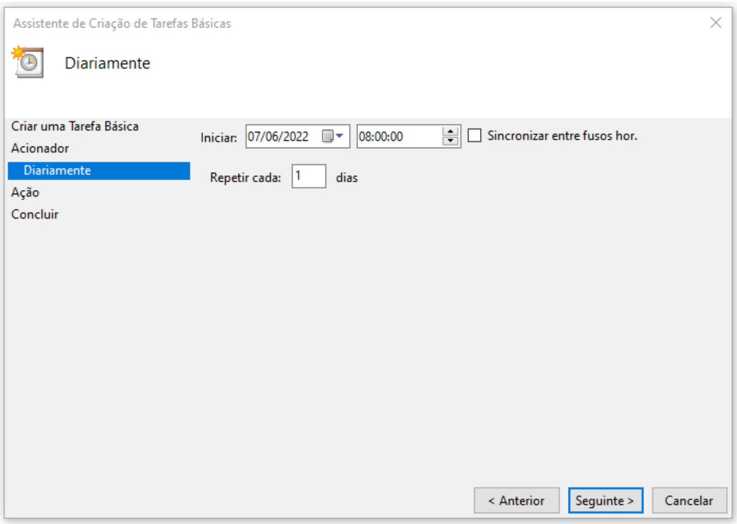
Acionador de Tarefa

Quando pretende iniciar a tarefa?

- [✓ Diariamente]
- Semanalmente
- Mensalmente
- Uma vez
- Quando o computador inicia
- Quando inicio sessão
- Quando um evento específico é registado

Figure 44 - Windows Task Scheduler: selecting the periodicity of my task.

I then selected the time of the day for the program to run, setting it to repeat the process every single day.


Assistente de Criação de Tarefas Básicas

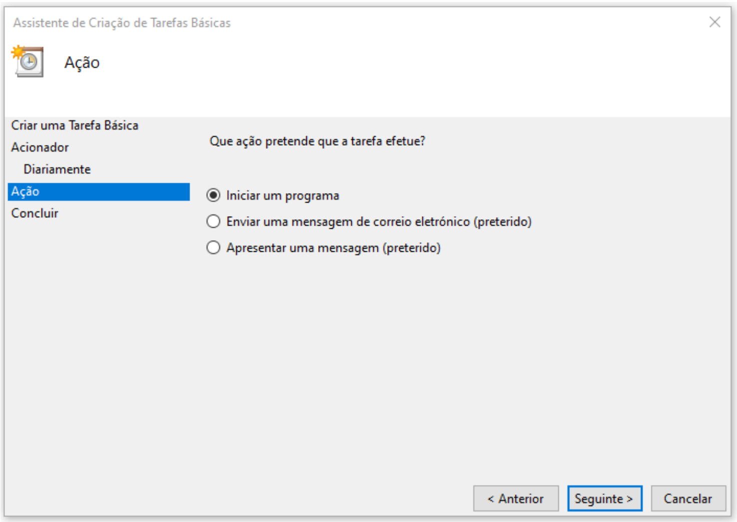
Diariamente

Criar uma Tarefa Básica
Acionador
Diariamente
Ação
Concluir

Iniciar: 07/06/2022 08:00:00 ☐ Sincronizar entre fusos hor.

Repetir cada: 1 dias

Figure 45 - Windows Task Scheduler: choosing the time and date to start the task.


Assistente de Criação de Tarefas Básicas

Ação

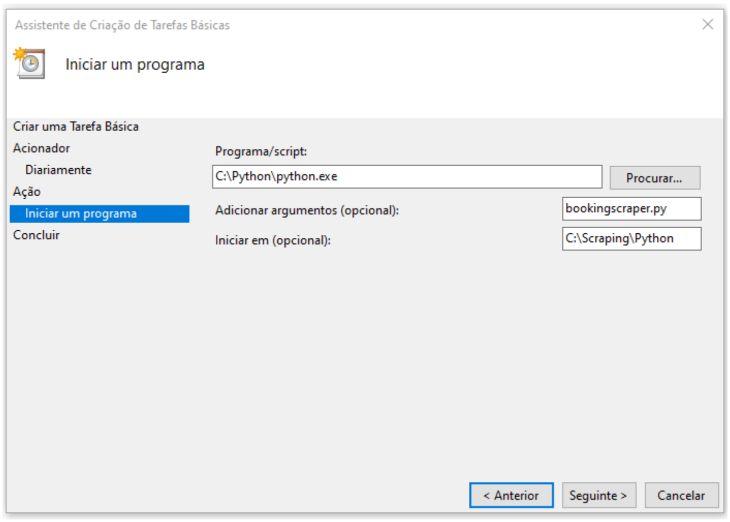
Criar uma Tarefa Básica
Acionador
Diariamente
Ação
Concluir

Que ação pretende que a tarefa efetue?

● Iniciar um programa
○ Enviar uma mensagem de correio eletrónico (preterido)
○ Apresentar uma mensagem (preterido)

Figure 46 - Windows Task Scheduler: instructing my task to run a program.

And lastly, I entered the location of two files: the location on which my Python.exe is installed, entered on "Program/script:", and the location of my .py file containing my scraping code on "Start in (optional)", while also typing the name of the .py script on "Add arguments (optional)".


Assistente de Criação de Tarefas Básicas

Iniciar um programa

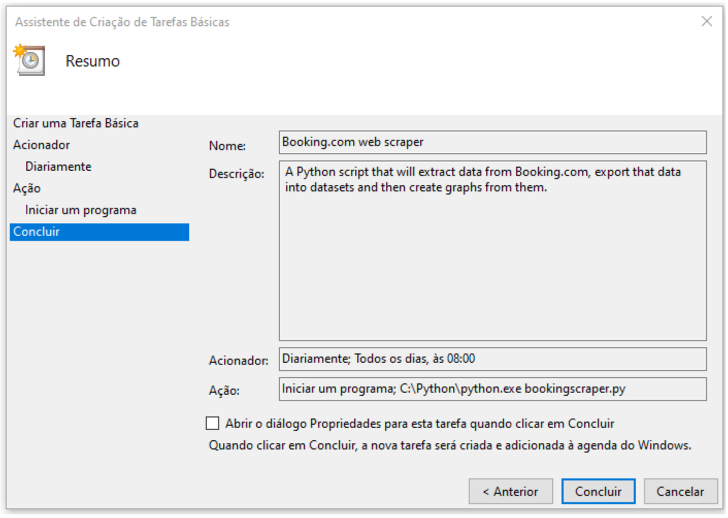
Criar uma Tarefa Básica
Acionador
Diariamente
Ação
Iniciar um programa
Concluir

Programa/script: C:\Python\python.exe

Adicionar argumentos (opcional): bookingscraper.py

Iniciar em (opcional): C:\Scraping\Python

Figure 47 - Windows Task Scheduler: selecting Python.exe installation and my program's folders.


Assistente de Criação de Tarefas Básicas

Resumo

Criar uma Tarefa Básica
Acionador
Diariamente
Ação
Iniciar um programa
Concluir

Nome: Booking.com web scraper

Descrição: A Python script that will extract data from Booking.com, export that data into datasets and then create graphs from them.

Acionador: Diariamente; Todos os dias, às 08:00

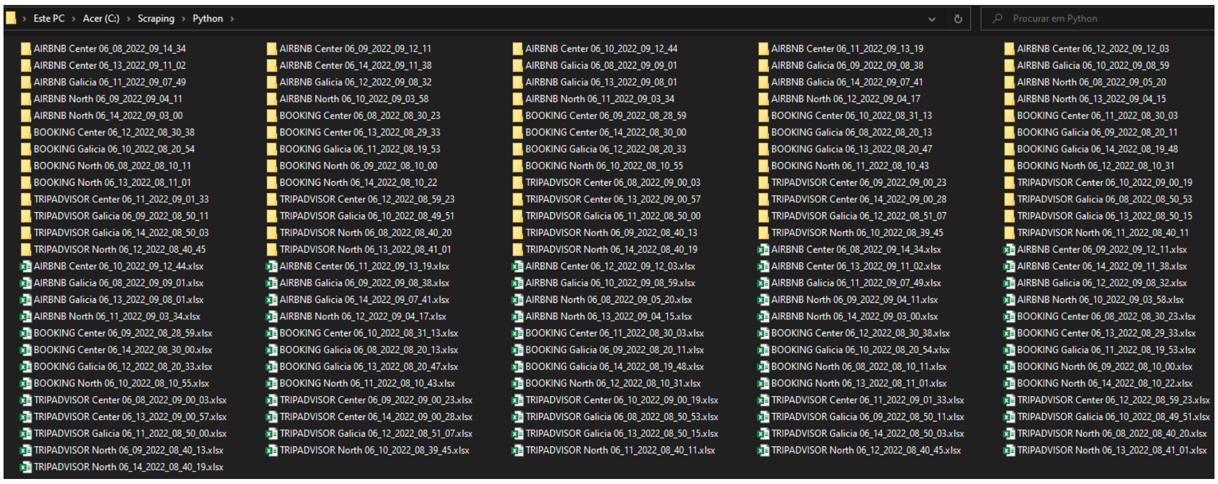
Ação: Iniciar um programa; C:\Python\python.exe bookingscraper.py

☐ Abrir o diálogo Propriedades para esta tarefa quando clicar em Concluir

Quando clicar em Concluir, a nova tarefa será criada e adicionada à agenda do Windows.

Figure 48 - Windows Task Scheduler: confirming my scheduled task.

My Python script was then scheduled to run every day, at 8 am. I let it run for one week, and every extraction was successful.



```
Este PC > Acer (C:) > Scraping > Python >
Procurar em Python

AIRBNB Center 06_08_2022_09_14_34
AIRBNB Center 06_13_2022_09_11_02
AIRBNB Galicia 06_11_2022_09_07_49
AIRBNB North 06_09_2022_09_04_11
AIRBNB North 06_14_2022_09_03_00
BOOKING Center 06_12_2022_08_30_38
BOOKING Galicia 06_10_2022_08_20_54
BOOKING North 06_08_2022_08_10_11
BOOKING North 06_13_2022_08_11_01
TRIPADVISOR Center 06_11_2022_09_01_33
TRIPADVISOR Galicia 06_09_2022_08_50_11
TRIPADVISOR Galicia 06_14_2022_08_50_03
TRIPADVISOR North 06_12_2022_08_40_45
AIRBNB Center 06_10_2022_09_12_44.xlsx
AIRBNB Galicia 06_08_2022_09_09_01.xlsx
AIRBNB Galicia 06_13_2022_09_08_01.xlsx
AIRBNB North 06_11_2022_09_03_34.xlsx
BOOKING Center 06_09_2022_08_28_59.xlsx
BOOKING Center 06_14_2022_08_30_00.xlsx
BOOKING Galicia 06_12_2022_08_20_33.xlsx
BOOKING North 06_10_2022_08_10_55.xlsx
TRIPADVISOR Center 06_08_2022_09_00_03.xlsx
TRIPADVISOR Center 06_13_2022_09_00_57.xlsx
TRIPADVISOR Galicia 06_11_2022_08_50_00.xlsx
TRIPADVISOR North 06_09_2022_08_40_13.xlsx
TRIPADVISOR North 06_14_2022_08_40_19.xlsx

AIRBNB Center 06_09_2022_09_12_11
AIRBNB Center 06_14_2022_09_11_38
AIRBNB Galicia 06_12_2022_09_08_32
AIRBNB North 06_10_2022_09_03_58
BOOKING Center 06_08_2022_08_30_23
BOOKING Center 06_13_2022_08_29_33
BOOKING Galicia 06_11_2022_08_19_53
BOOKING North 06_09_2022_08_10_00
BOOKING North 06_14_2022_08_10_22
TRIPADVISOR Center 06_12_2022_08_59_23
TRIPADVISOR Galicia 06_10_2022_08_49_51
TRIPADVISOR North 06_08_2022_08_40_20
TRIPADVISOR North 06_13_2022_08_41_01
AIRBNB Center 06_11_2022_09_13_19.xlsx
AIRBNB Galicia 06_09_2022_09_08_38.xlsx
AIRBNB Galicia 06_14_2022_09_07_41.xlsx
AIRBNB North 06_12_2022_09_04_17.xlsx
BOOKING Center 06_10_2022_08_31_13.xlsx
BOOKING Galicia 06_08_2022_08_20_13.xlsx
BOOKING Galicia 06_13_2022_08_20_47.xlsx
BOOKING North 06_11_2022_08_10_43.xlsx
TRIPADVISOR Center 06_09_2022_09_00_23.xlsx
TRIPADVISOR Center 06_14_2022_09_00_28.xlsx
TRIPADVISOR Galicia 06_12_2022_08_51_07.xlsx
TRIPADVISOR North 06_10_2022_08_39_45.xlsx

AIRBNB Center 06_10_2022_09_12_44
AIRBNB Galicia 06_08_2022_09_09_01
AIRBNB Galicia 06_13_2022_09_08_01
AIRBNB North 06_11_2022_09_03_34
BOOKING Center 06_09_2022_08_28_59
BOOKING Center 06_14_2022_08_30_00
BOOKING Galicia 06_12_2022_08_20_33
BOOKING North 06_10_2022_08_10_55
TRIPADVISOR Center 06_08_2022_09_00_03
TRIPADVISOR Center 06_13_2022_09_00_57
TRIPADVISOR Galicia 06_11_2022_08_50_00
TRIPADVISOR North 06_09_2022_08_40_13
TRIPADVISOR North 06_14_2022_08_40_19
AIRBNB Center 06_12_2022_09_12_03.xlsx
AIRBNB Galicia 06_10_2022_09_08_59.xlsx
AIRBNB North 06_08_2022_09_05_20.xlsx
AIRBNB North 06_13_2022_09_04_15.xlsx
BOOKING Center 06_11_2022_08_30_03.xlsx
BOOKING Galicia 06_09_2022_08_20_11.xlsx
BOOKING Galicia 06_14_2022_08_19_48.xlsx
BOOKING North 06_12_2022_08_10_31.xlsx
TRIPADVISOR Center 06_10_2022_09_00_19.xlsx
TRIPADVISOR Galicia 06_08_2022_08_50_53.xlsx
TRIPADVISOR Galicia 06_13_2022_08_50_15.xlsx
TRIPADVISOR North 06_11_2022_08_40_11.xlsx

AIRBNB Center 06_11_2022_09_13_19
AIRBNB Galicia 06_09_2022_09_08_38
AIRBNB Galicia 06_14_2022_09_07_41
AIRBNB North 06_12_2022_09_04_17
BOOKING Center 06_10_2022_08_31_13
BOOKING Galicia 06_08_2022_08_20_13
BOOKING Galicia 06_13_2022_08_20_47
BOOKING North 06_11_2022_08_10_43
TRIPADVISOR Center 06_09_2022_09_00_23
TRIPADVISOR Center 06_14_2022_09_00_28
TRIPADVISOR Galicia 06_12_2022_08_51_07
TRIPADVISOR North 06_10_2022_08_39_45
AIRBNB Center 06_08_2022_09_14_34.xlsx
AIRBNB Center 06_13_2022_09_11_02.xlsx
AIRBNB Galicia 06_11_2022_09_07_49.xlsx
AIRBNB North 06_09_2022_09_04_11.xlsx
AIRBNB North 06_14_2022_09_03_00.xlsx
BOOKING Center 06_12_2022_08_30_38.xlsx
BOOKING Galicia 06_10_2022_08_20_54.xlsx
BOOKING North 06_08_2022_08_10_11.xlsx
BOOKING North 06_13_2022_08_11_01.xlsx
TRIPADVISOR Center 06_11_2022_09_01_33.xlsx
TRIPADVISOR Galicia 06_09_2022_08_50_11.xlsx
TRIPADVISOR Galicia 06_14_2022_08_50_03.xlsx
TRIPADVISOR North 06_12_2022_08_40_45.xlsx

AIRBNB Center 06_12_2022_09_12_03
AIRBNB Galicia 06_10_2022_09_08_59
AIRBNB North 06_08_2022_09_05_20
AIRBNB North 06_13_2022_09_04_15
BOOKING Center 06_11_2022_08_30_03
BOOKING Galicia 06_09_2022_08_20_11
BOOKING Galicia 06_14_2022_08_19_48
BOOKING North 06_12_2022_08_10_31
TRIPADVISOR Center 06_10_2022_09_00_19
TRIPADVISOR Galicia 06_08_2022_08_50_53
TRIPADVISOR Galicia 06_13_2022_08_50_15
TRIPADVISOR North 06_11_2022_08_40_11
AIRBNB Center 06_09_2022_09_12_11.xlsx
AIRBNB Center 06_14_2022_09_11_38.xlsx
AIRBNB Galicia 06_12_2022_09_08_32.xlsx
AIRBNB North 06_10_2022_09_03_58.xlsx
BOOKING Center 06_08_2022_08_30_23.xlsx
BOOKING Center 06_13_2022_08_29_33.xlsx
BOOKING Galicia 06_11_2022_08_19_53.xlsx
BOOKING North 06_09_2022_08_10_00.xlsx
BOOKING North 06_14_2022_08_10_22.xlsx
TRIPADVISOR Center 06_12_2022_08_59_23.xlsx
TRIPADVISOR Galicia 06_10_2022_08_49_51.xlsx
TRIPADVISOR North 06_08_2022_08_40_20.xlsx
TRIPADVISOR North 06_13_2022_08_41_01.xlsx
```

Figure 49 - Results obtained after 7 days of automated extractions.

Each extraction ran for approximately 1 hour and 10 minutes, giving me 9 datasets and the respective graphs, per day.

4.6. PROBLEMS ENCOUNTERED

1. This web scraping method requires regular maintenance

The online tourism platforms update their pages' source codes very often, which makes it necessary to review the code and update the HTML/CSS elements' tags. In the span of 5 months, I had to update my code 6 times due to sudden alterations on the online platforms. It is a relatively fast process, but still needed, as it depends on the tourism platforms and is beyond our control. Most of the times, the websites only alter their CSS tags, without modifying the structure of their page, which can be fixed rather quickly. However, occasionally, their updates might be more drastic, requiring a full inspection of my code.

2. The API keys for Booking.com, Tripadvisor and Airbnb were unavailable

Initially, the plan was to request Booking, Airbnb and Tripadvisor's APIs, in order to collect their data, as with APIs we receive information in the way they intend us to. Unfortunately, these tourism platforms have not been accepting requests for new API keys for months, which made it so I had to collect their data using the method described in this dissertation. This led to the maintenance requirements previously mentioned, among other web scraping difficulties that would otherwise not occur.

3. The amount of comments to extract was too large

Initially, my plan was to extract every single user comment from every single accommodation result available. Then, I realised that, taking only Booking.com into consideration, I was going to extract 3000 unique results (adding all zones and ignoring time periods, as the comments will always be the same). It is fair to assume that each hotel might have at least 1000 comments, as was the case for many of the single results I have analysed. Each page of comments contains 10 comments, which means that on each hotel, I would have to scrape through 100 different pages. Multiplying that number by those 3000 unique results gives me 300 000 unique pages to scrape through. On my computer, scraping one single page takes approximately 10 seconds, which means that to scrape every single comment, my computer would have to spend 3 million seconds extracting data. This equals to almost 35 days of continuous scraping, which is impossible for me to accomplish alone. Although it is doable with large companies capable of dealing with big data, I settled with extracting only the top 10 comments of each single result, which was enough to obtain some valuable information.

4. Some platforms do not show every listing available

In this particular case, Booking only presented 1000 results, even if there were 3000 listed as available. On Airbnb, only the first 300 results were presented, despite the number of listings available. It would be possible to fix this by filtering the searches even further. However, especially with Booking, I figured that the first 1000 results were enough to obtain valuable information, taking into consideration the fact that while approaching the later pages of results, the number of hotels with very few information and reviews became higher and higher. Therefore, there was no need to include even more results that would possibly be almost empty.

5. Sometimes there might be loss of data while scraping the web platforms.

There are two situations on which there may be loss of data. The first occurs when the code is being executed too quickly, which makes it so the program tries to extract the data before the page has had time to fully load, resulting in the extraction of empty data. This was easily fixed by instructing the code to wait a few seconds before extracting the data.

The second reason is the fact that I was extracting data from my personal computer, using my personal IP address. Although it was a rare occurrence, after a while the online platforms would blacklist my IP address, leading to blank data. This is fixable by using online proxies, which rotate through several IP addresses, tricking the online platforms into thinking all these requests are not coming from the same person.

6. Tripadvisor does not include information on check-in/check-out dates on their URLs

Since I could not use APIs, my method of extraction depended on modifying the URL to define the target zone and the check-in and check-out data. It usually works pretty well, but in this case, Tripadvisor does not include information on the check-in and check-on on their link, which made it very difficult to automatically extract data from Tripadvisor selecting a date other than the current day. Although rather unpractical with this method of extraction, there was a solution, which was the "action" class on Selenium to emulate the process of using a mouse to click on the calendar and select the desired dates.

7. Excel does not support Regular Expressions

While regular expressions are not always needed to clean, I realised that they were not working in Excel, requiring instead the usage of Excel's functions to define the rules for finding and replacing text. Instead of doing that, I used Google Sheets, which fully supports regular expressions.

8. Online platforms try to counter web scrapers

Sometimes, while not always on purpose, the tourism platforms implement measures that counter web scrapers. From my personal experience, besides the regular change of CSS codes, they also tried to disable the HTML element that includes the "next" button, making it so I can't invoke it when using Beautiful Soup, which forced me to find an alternative to find the total number of pages to extract. There was also an instance where if the number of results for a specific zone was too low, it would show results from a different zone after the last result from the desired zone, forcing me to teach the code to distinguish the correct and the wrong results.

5. CONCLUSION

This project made use of the Python programming language to develop a system for the extraction and analysis of tourism data from the main online tourism platforms. Initially, a web scraping tool was created, accompanied by an intuitive user interface. Then, after cleaning the extracted data, graphs were created, followed by the creation of an interactive dashboard to assist in the visualisation of data. Lastly, all previous steps were automated, thus creating an automated web scraping and data visualisation tool that fulfilled this project's needs.

The main conclusion drawn from this project is that there is an enormous amount of available web data that are not being used, despite their potential value. Companies spend thousands of dollars on the acquisition of user data to improve their systems of predictive analytics, while overlooking data that are available for free on the Internet. Sometimes, only minimal effort is required to collect and process data with extremely high value. Therefore, I would classify web scraping as an excellent data collection technique, whose main goals lie in its low cost and high efficiency.

It is also evident that, based on the results obtained from my extractions, Portugal is a good tourism destination, with most user reviews being classified as positive. There is a clear preference of bigger cities, especially Porto, which has over 15 times the number of listings available in the region of Gerês, despite being 15 times smaller in size. Coastal cities are clearly favoured by the tourists, which makes sense, considering Portugal has beaches as one of its main touristic strengths. Thus, we should keep investing on the development and sustainability of this sector (especially in the non-coastal zones), as it is a major influencer on this country's economic growth.

Finally, I have concluded that there is a lot of potential in the field of Data Science, as we are constantly surrounded by information, which will only increase in the future. It is no coincidence that "data scientist" is considered one of the jobs of the future, given the expected development of ICTs. Following that scientific field, Python appears as one of the best programming languages, not only for data analysis and visualisation, but also machine learning and artificial intelligence. Thanks to its high versatility, I was able to create a very effective tool that performed all the tasks necessary. Thus, it serves as a powerful foundation for the future collection and analysis of tourism data in Portugal and Galicia, within the context of this project's associated program.

BIBLIOGRAPHIC REFERENCES

Adeniyi, D.A., Wei, Z., & Yongquan, Y. (2016). Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. Applied Computing and Informatics, 12(1), 90-108. https://doi.org/10.1016/j.aci.2014.10.001

Adhinugroho, Y., Putra, A. P., Luqman, M., Ermawan, G. Y., Takdir, Mariyah, Siti, & Pramana, Setia. (2020). Development of online travel Web scraping for tourism statistics in Indonesia. Information Research, 25(4), 885. https://doi.org/10.47989/irpaper885

Ajibade, S.S., & Adediran, A. (2016). An overview of big data visualization techniques in data mining. International Journal of Computer Science and Information Technology Research, 4(3), 105-113. https://www.researchgate.net/publication/305905594_An_Overview_of_Big_Data_Visualization_Techniques_in_Data_Mining

Almaqbali, I. S. H., Al Khufairi, F. M. A., Khan, M. S., Bhat, A. Z., & Ahmed, I. (2020). Web Scraping: Data Extraction from Websites. Journal of Student Research. https://doi.org/10.47611/jsr.vi.942

Buhalis, D. (2003). eTourism: Information Technology for Strategic Tourism Management. Pearson Education Limited, Harlow. https://doi.org/10.1080/02508281.2005.11081482

Buhalis, D. & O'Connor, P. (2005). Information Communication Technology Revolutionizing Tourism. Tourism Recreation Research, 30:3, 7-16. https://www.researchgate.net/publication/30930596_Information_Communication_Technology_Revolutionizing_Tourism

Choong, W. J. (2019). An automated web scraping tool for Malaysia tourism. Diss, UTAR. http://eprints.utar.edu.my/3493/

Galinha, P.F.S.L. (2017). DATA MINING NO TURISMO EM PORTUGAL: Análise Preditiva no Suporte à Tomada de Decisão. https://run.unl.pt/handle/10362/34382

Grus, J. (2019). Data science from scratch: first principles with python. O'Reilly Media.

Khedikar, K. A. (2021) Data Analytics for Business Using Tableau. Proceedings of the International Conference on Innovative Computing & Communication 2021. http://dx.doi.org/10.2139/ssrn.3835030

Martina, B. (2021). The disruptive reaction of the travel and tourism industry to the Covid-19 pandemic: the case of Airbnb online experiences. http://hdl.handle.net/10362/140262

Mathieson, A & Wall, G. (1990). Tourism: Economic, Physical and Social Impacts. New York: John Wiley & Sons

Matta, P., Sharma, N., Sharma, D., Pant, B. & Sharma, S. (2020). Web Scraping: Applications and Scraping Tools. International Journal of Advanced Trends in Computer Science and Engineering, 9(5). https://doi.org/10.30534/ijatcse/2020/185952020

Mooney, S. J., Westreich, D. J., & El-Sayed, A. M. (2015). Epidemiology in the era of big data. Epidemiology, 26(3), 390. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4385465/

Oliveira, R.A. (2016). Extração de dados do site TripAdvisor como suporte na elaboração de indicadores do turismo de Minas Gerais: uma iniciativa em Big Data. https://repositorio.ufmg.br/handle/1843/ECIP-AN2PRB

O'Reilly, S. (2006). Nominative fair use and Internet aggregators: Copyright and trademark challenges posed by bots, web crawlers and screen-scraping technologies. Loyola Consumer Law Review, 19, 273. https://lawecommons.luc.edu/cgi/viewcontent.cgi?article=1163&context=lclr

Pan, B. (2015). E-Tourism. https://www.researchgate.net/publication/270273782_E-Tourism

Todd, S. (2015). O valor dos dados em um mundo impulsionado por informações. Accessed in 23/02/2021, available at: https://canaltech.com.br/big-data/o-valor-dos-dados-em-um-mundo-impulsionado-por-informacoes-51425/

Xiang, Z. (2018). From digitization to the age of acceleration: On information technology and tourism. Tourism Management Perspectives, 25, 147-150. https://doi.org/10.1016/j.tmp.2017.11.023

Zhao, B. (2017). Web scraping. Encyclopedia of big data, 1-3. https://doi.org/10.13140/2.1.3121.5681