**BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA**

**FACULTY OF MATHEMATICS AND COMPUTER SCIENCE SPECIALIZATION COMPUTER SCIENCE**

**DIPLOMA THESIS**

**SmartShoppingPlugin**

**Supervisor**

Profesor Doctor Ioan Badarinza

**Author**

Coman Vlad-Paul

**2020**

Table of Contents

1. Introduction 4
2. The problem 7

2.2 Existing applications 8

1. Theoretical aspects 12

3.1 Front-End Technologies Used 12

3.2 Back-End Technologies Used 14

1. Proposed solution 19
2. Implementation details 28

5.1 Frameworks and libraries used 28

1. Conclusion 31
2. References 33

List of Figures

1. ContentScriptMessageEx
2. Dealicious interface
3. The Camelizer
4. Honey
5. Pricescout
6. Microservices
7. Spring Boot logo
8. Maven logo
9. Dependency for mysql connector
10. MySql Logo
11. REST API
12. Web Crawler
13. Use Case diagram
14. Popup.html file
15. Plugin Architecture
16. Example ajax logic

Chapter 1. Introduction

**Shopping** is an activity in which a customer browses the available goods or services presented by one or more retailers with the potential intent to purchase a suitable selection of them.

A typology of shopper types has been developed by scholars which identifies one group of shoppers as recreational shoppers, that is, those who enjoy shopping and view it as a leisure activity.

Online shopping has become a major disruptor in the retail industryas consumers can now search for product information and place product orders across different regions. Online retailers deliver their products directly to the consumers' home, offices or wherever they want.

The B2C (business to consumer) process has made it easy for consumers to select any product online from a retailer's website and to have it delivered relatively quickly. Using online shopping methods, consumers do not need to consume energy by physically visiting physical stores. This way they save time and the cost of travelling. A retailer or a shop is a business that presents a selection of goods and offers to trade or sell them to customers for money or other goods.

Shoppers' shopping experiences may vary. They are based on a variety of factors including how the customer is treated, convenience, the type of goods being purchased, and mood.

According to a 2000 report, in New York State, women purchase 80% of all consumer goods.[15]

**History**

In antiquity, marketplaces and fairs were established to facilitate the exchange of goods and services. People would shop for goods at a regular market in nearby towns. However, the transient nature of stalls and stall-holders meant the consumers needed to make careful inspection of goods prior to purchase.

In ancient Greece, the agora served as a marketplace where merchants kept stalls or shops to sell their goods.

Ancient Rome utilized a similar marketplace known as the forum. Rome had two forums; the Forum Romanum and Trajan's Forum. Trajan's Market at Trajan's forum, built around 100-110CE, was a vast expanse, comprising multiple buildings with tabernae that served as retail shops, situated on four levels. The Roman forum was arguably the earliest example of a permanent retail shopfront.

 In the Roman world, the central market primarily served the local peasantry. Those who lived on the great estates were sufficiently attractive for merchants to call directly at their farm-gates, obviating their need to attend local markets.

Shopping lists are known to have been used by Romans. One such list was discovered near Hadrian's wall dated back to 75–125 CE and written for a soldier.

Archaeological evidence suggests that the British engaged in minimal shopping in the early Middle Ages.

Instead, they provided for their basic needs through subsistence farming practices and a system of localised personal exchanges. However, by the late Middle Ages, consumers turned to markets for the purchase of fresh produce, meat and fish and the periodic fairs where non-perishables and luxury goods could be obtained.

Women were responsible for everyday household purchases, but most of their purchasing was of a mundane nature. For the main part, shopping was seen as a chore rather than a pleasure.

Relatively few permanent shops were to be found outside the most populous cities. Instead customers walked into the tradesman's workshops where they discussed purchasing options directly with tradesmen. Itinerant vendors such as costermongers, hucksters and peddlers operated alongside markets, providing the convenience of home delivery to households, and especially to geographically isolated communities.

In the more populous European cities, a small number of shops were beginning to emerge by the 13th century. Specialist retailers such as mercers and haberdashers were known to exist in London, while grocers sold "miscellaneous small wares as well as spices and medicines." However, these shops were primitive. As late as the 16th century, London's shops were described as little more than "rude booths."

The Medieval shopper's experience was very different from that of the contemporary shopper. Interiors were dark and shoppers had relatively few opportunities to inspect the merchandise prior to consumption.

Glazed windows in retail environments, were virtually unknown during the medieval period. Goods were rarely out on display; instead retailers kept the merchandise at the rear of the store and would only bring out items on request.

The service counter was virtually unknown and instead, many stores had openings onto the street from which they served customers.

In Britain, medieval attitudes to retailing and shopping were negative. Retailers were no better than hucksters, because they simply resold goods, by buying cheaper and selling dearer, without adding value of national accounts.

Added to this were concerns about the self-interest of retailers and some of their more unethical practices. Attitudes to spending on luxury goods also attracted criticism, since it involved importing goods which did little to stimulate national accounts, and interfered with the growth of worthy local manufacturers.

The needs of better and smarter shopping keep to adapt even form a month to another, the main idea of the application is that to keep the user informed about the different prices and websites, and a various gamma of sources form where the user can get the best offer for the searched product, in instant time using the SmartShopping extension.

The second chapter called “The Problem” we will describe a real life problem that this application is trying to solve.

The third chapter will describe the theoretical aspects in this application manly the technologies used for developing it form scratch and understanding the algorithms used. The following tools were used: SpringBoot, Scrapy(Web Scrapping Tool), MySql , ChromePluginDevTools.

Fourth chapter “Proposed Solution” describes the solution provided for “The Problem” in this application and also the software development used for this approach.

In the fifth chapter we will describe the implementation of the application, this chapter contains a guide of how the API of the server can be used, how the database is populated and the connections made for the application workflow.

Sixth chapter will contain the final conclusion.

Chapter 2. The problem

In this paper we will describe a real-time system for helping the user to perform a better and a smarter shopping. This application will easily adapt to a very wide gamma of products form various site sources.

A close up of a logo

Description automatically generated

This application aims to keep the user informed about a specific products from different sources(different web sites)in real time. Very often it can happen to find the same product on a different site and with a cheaper price than the initially found. So with this SmartShopping plugin the user will be always informed, we will store and display data about few products but this can easily be adapted to all the products available on the internet for the only shopping.

This research paper aims at design efficient software for storing data form different sites in a database with the help of web scrapping application written in python wich will store data in a mysql database about a very wide gama of products, then we have a Springboot server that sends data trought several API’s in to a chrome plugin so that data will be displayed in real time on the web page.

The interface of the plugin is as most friendly user as can be, why? The user of the SmartShopping plugin will be just one click away from getting the data about the specific product from different sites.

**2.1Existing applications**

The following ChromePlugins have similar purpose with my application, all of them designed for a more efficient online shopping, having several things in common but each one has some features that makes them unique.

1.Dealicious [1]

Dealicious[1] (Verified) is a browser add-on that acts just like a personal deal finder, offering you awesome recommendations for lower prices from reliable merchants, while you browse through various online shops.When you look ata.

Product online, Dealicious (Verified) analyzes the product page in the background and checks its ever-growing database, in order to show you a list of stores where you can find that same product, but at a more affordable price.(Figure 1)

A screenshot of a cell phone

Description automatically generated

*Figure 1 – Dealicious interface [1]*

1.*The Camelizer [2]*

Most savvy shoppers know that shopping at certain times can snag you the best deals. [The Camelizer](http://camelcamelcamel.com/camelizer" \t "_blank) is an extension that shows you the price history of an item you're browsing on Amazon to help you decide whether to buy now or wait for a better deal.

It will also send you price drop alerts via email or Twitter, and you can import your Amazon wish list so that you know when your heart's desire goes on sale.[2](Figure 2)

A screenshot of a computer

Description automatically generated

*Figure 2 – The Camelizer[2]*

2.*Honey[2]*

[Honey](https://chrome.google.com/webstore/detail/honey/bmnlcjabgnpnenekpadlanbbkooimhnj?hl=en-US) is one of the most popular money-saving Chrome extensions, and for good reason. Whenever you check out online, Honey will search for applicable coupons and apply them to your purchase by automatically filling in the coupon code for you. No more wondering if there's an amazing coupon you're missing out on.

Plus, if you're shopping at one of the supported online stores, such as Target, eBay, Walmart or Macy's, a click of the Honey button in your browser will show you all of the sales and coupons that the site has to offer.

Spending at these stores also earns you HoneyGold, points that can be redeemed for gift cards or real money[2].

A screenshot of a social media post

Description automatically generated

*Figure 3 – Honey[2]*

*Pricescout[2]*

[Pricescout](http://pricescout.io/) can find coupon codes for you, just like Honey, but it also comparison shops. While you're shopping, it scans the sites of over 21,000 retailers and will pop up with better prices. I like to combine Honey and Pricescout to find the best deal.

A screenshot of a cell phone

Description automatically generated

*Figure – Pricescout [2]*

Chapter 3. Theoretical aspects

This project is a Chrome Plugin application based on a Client-Server architecture, in this application are three main programs that are running at different times and all have the same database: the scrapper (Python program) that goes over the sites and crawl all the data needed from the markets pages and seeds all the data in to mysql database, then the code form the server (springboot) that responds to HTTP requests (backend) and the code that lives in the chrome plugin that takes user requests (frontend) and communicates with the server.

Backend part is split into independent Micro Services, each handling parts of the system that allows easy maintenance and very adaptability for changes also development and deployment of each service individuality. For example if we want to change some attributes of the “mobiles” products we can change in serves site just the related endpoints, a hole refactoring not to be needed. Service communicate with each other through REST Template wich will get the connection to API endpoints.

The server site is build with Spring boot technologie wich is an open source Java-Based framework used to develop a stand-alone application, a very easy to understand and use wich will reduce the developing time and increase the productivity. Spring Boot [4] automatically builds the application based on the dependencies added in the project wich are managed by Maven[5].

**3.1 Front-End Technologies**

Choosing the right technologies for the Front-End part of the application is essential because the user interacts directly with this part. The option to present the hypertext are limited, being maintained by World Wide Web Consortium (W3C), the standard language is displayed with the help of the HTML along with the CSS(the one that defines the style of the page in most cases). The JavaScript and Jquery are also essential for the user interaction with the page (plugin in our case).

The Chrome browser knows to read an html file specified in the manifest.js file so that the browser can upload the associated code in your plugin extension

Extensions are made of different, but cohesive, components. Components can include background scripts, content scripts, an option page, Ui elements and various logic files. Extension components are created with web development technologies: HTML, CSS, and JavaScript. An extension's components will depend on its functionality and may not require every option.[5]

Extensions are zipped bundles of HTML, CSS, JavaScript, images, and other files used in the web platform, that customize the Google Chrome browsing experience. Extensions are built using web technology and can use the same APIs the browser provides to the open web.

Extensions have a wide range of functional possibilities. They can modify web content users see and interact with or extend and change the behavior of the browser itself.

Consider extensions the gateway to making the Chrome browser the most personalized browser.[6]

HTML (HyperText Markup Language) is the language used to create webpages and also in our case ChromePlugins, alongside with CSS JavaScript and Jquerry wich are base technologies used for web pages and user interfaces. Its in a continuous developing with the help of W3C wich are the ones responsible for maintaining and improving it.

Is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages.

With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as

Other tags such as

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.[7]

**3.2 Back-End Technologies Used**

Microservices also know as the microservice architecture is an architectural style that structures an application as a collection of services that are : highly maintainable and testable,

loosely coupled, independently deployable, organized around business. Having its own connection to the database. (Figure 3) They can handle many tasks and also since the services don’t know about each other implementation it a lot easier to maintain test and make changes.

A screenshot of a cell phone

Description automatically generated

*Figure 3 – Microservices[9]*

A close up of a logo

Description automatically generated

*Figure 4- Spring Boot logo [9]*

*Spring Boot[9]*

The server is build around Spring Boot, an open-source Java-based framework used to

build application with minimum configurations in order to run. The main important feature of springboot is to avoid XML configuration completely. Rod Johnson crated this framework in 2002, being ready for the market a year later, the application received updates in every year or at two years.Spring 1.0 was launched in 2013 and along side other innovations was the support for Java Standard Edition 8. The most important features provided by SpringBoot are :

* Authentification, wich can be very quick configurated and has the support of Spring Security
* JDBC Java Database Conectivity allowing us to work with relational databases
* OOP programming

Even today with all the updates and the nice and easy to use features the SpringBoot is still Open Source and accessible for everyone.

A close up of a logo

Description automatically generated

*Figure 5- Maven logo[4]*

*Maven[4]*

Maven is a open-source operating system to build and administrate projects write in java code. Maven takes two aspects to build software: the first is the one that describes the way that the software is build, the second part describes the dependencies used. Unlike the previous instruments like Apache Ant, maven uses the convention for building procedures and only the exceptions need to be writed. An XML file describes the build project, the dependencies used for this and external components, necessary plugins. Maven has predefined objectives for assembling and compiling code, knows how to dynamically download Java libraries and maven plugins and stores them in a local cache.

In Maven we don’t need to manage jar files manualy. Instead of downloading and placing the items correctly of jar type, we can simply use Maven whit dependencies. All we need to do is to add these dependencies in “pom.xml” file to benefit form this ones maven needs internet connection so that he can download the Jars[4].

For example to connect our SpringBoot project with MySql database we only need 5 lines of code in pom.xml file(figure 7).

A picture containing drawing, food

Description automatically generated

*Figure 6- Dependency for mysql connector*



*Figure 7- MySql Logo*

*MySql[8]*

MySql is a management system for relational data bases. The name origins are form “My” the name of the co-founder daughter Michael Widenius, and “SQL” from “Structure Language Query”.This is a free and open source software with a lot of licensed versions.

*MySQL* is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

*MySQL*Server works in client/server or embedded systems.

The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

We also provide MySQL Server as an embedded multithreaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

I’ve choosen to use mysql because it’s very easy to use and understand and also is interacting pretty well with the springboot site from the back-end part.

A lot of world wide organizations work with mySql databases storages such as : Facebook , Google, Adobe, Alcatel Lucent and Zappos, they are using mysql to save time and money, also for filling up Web pages that requires a big number of data[8].

*JDBC[10]*

This is the central class in the JDBC core package**.**

It simplifies the use of JDBC and helps to avoid common errors. It executes core JDBC workflow, leaving application code to provide SQL and extract results.

This class executes SQL queries or updates, initiating iteration over ResultSets and catching JDBC exceptions and translating them to the generic, more informative exception hierarchy defined in the org.springframework.dao package.

Code using this class need only implement callback interfaces, giving them a clearly defined contract. The PreparedStatementCreator callback interface creates a prepared statement given a Connection, providing SQL and any necessary parameters.

The ResultSetExtractor interface extracts values from a ResultSet. See also PreparedStatementSetter and RowMapper for two popular alternative callback interfaces.

Can be used within a service implementation via direct instantiation with a DataSource reference, or get prepared in an application context and given to services as bean reference.

The DataSource should always be configured as a bean in the application context, in the first case given to the service directly, in the second case to the prepared template.

Because this class is parameterizable by the callback interfaces and the SQLExceptionTranslator interface, there should be no need to subclass it.

All SQL operations performed by this class are logged at debug level, using "org.springframework.jdbc.core.JdbcTemplate" as log category.

Unlike the usual Java implementation in wich case closing/opening the database connection should be taken in consideration, with JDBC template we don’t need to worry about that anymore, also the SQLException the described variant form above it’s much more simpler.

For interrogating the MySql database with JDBC we cand simply use the query method an alternative for queryForObject or queryForList etc depending on the type of the problem.

The query method accepts the parameters being sql text (the actual query made) and also the RowMapper parameter for translating all the affected rows in to an Object.

The method queryForObject returns basically the same result as query method, the difference between the two is that the queryForObject method returns only one object and the query method returns a list of objects[10].

*REST API[10]*

**Representational state transfer** (**REST**) is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called *RESTful* Web services, provide interoperability between computer systems on the Internet.

RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. Other kinds of Web services, such as SOAP Web services, expose their own arbitrary sets of operations.

"Web resources" were first defined on the World Wide Web as documents or files identified by their URLs. However, today they have a much more generic and abstract definition that encompasses every thing, entity, or action that can be identified, named, addressed, handled, or performed, in any way whatsoever, on the Web.

In a RESTful Web service, requests made to a resource's URI will elicit a response with a payload formatted in HTML, XML, JSON, or some other format. The response can confirm that some alteration has been made to the resource state, and the response can provide hypertext links to other related resources.

When HTTP is used, as is most common, the operations (HTTP methods) available are GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS and TRACE[10].

A close up of a map

Description automatically generated

*Figure 8- REST API*

Chapter 4. Proposed Solution

A ChromeExtension wich will keep informed the user about all the products over the

Internet and keep the data inside a MySql database, the presented solution is just a sample of what can be made with these technologies.

The solution is composed by three applications:

* A Python scrapper which will crawl over many pages and bring data about different products.
* A SpringBoot server which will contain several API’s and be connected with the database.
* A ChromeExtension that will make some ajax requests so that we populate the SmartShoppingPlugin

The application can very easily adapt to a web page that use data from another pages,

But I fond more revealed and user friendly implemented like this.

At first I will like to begin with the architecture of the application and describe a little few concepts introduced here.

The first and the essential application is written in python and with the help of the Scrapy framework used and Web Crawling technologie, i’ve managed to store data in a local mysql database for a wide gamma of products for example: Phones, Tablets, TVs and the application can be easily adapt to different type of product like books or products from different domains.

*Web Crawling*[12]

A **Web crawler**, sometimes called a **spider** or **spiderbot** and often shortened to **crawler**, is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing (*web spidering*).

Web search engines and some other sites use Web crawling or spidering software to update their web content or indices of others sites' web content. Web crawlers copy pages for processing by a search engine which indexes the downloaded pages so users can search more efficiently.

Crawlers consume resources on visited systems and often visit sites without approval. Issues of schedule, load, and "politeness" come into play when large collections of pages are accessed. Mechanisms exist for public sites not wishing to be crawled to make this known to the crawling agent. For example, including a robots.txt file can request bots to index only parts of a website, or nothing at all.

The number of Internet pages is extremely large; even the largest crawlers fall short of making a complete index. For this reason, search engines struggled to give relevant search results in the early years of the World Wide Web, before 2000. Today, relevant results are given almost instantly.

A close up of a logo

Description automatically generated*Figure 9 Web Crawler*

So after this application crawls data about our products from different site sources for example (emag, evomag, pcGarage in our case) the data, more specific the image of the product the price and the site source from where we can find it will be stored in our database.

After that we have a SpringBoot server with the help of the maven dependencies is connected with the same database. The SpringBoot server can be easily adapt and changed even to send data to different types of applications.

The SpringBoot server will communicate with the ChromePlugin and throught several API’s we will have our desired data almost instantly in the right side of the page for our product, the user just need to click on the SmartShoppingPlugin icon and the data will be displayed.

A close up of a map

Description automatically generated

*Figure 10*-*Use Case diagram*

*ChromeExtensions*[13]

Extensions are zipped bundles of HTML, CSS, JavaScript, images, and other files used in the web platform, that customize the Google Chrome browsing experience. Extensions are built using web technology and can use the same APIs the browser provides to the open web.

Extensions have a wide range of functional possibilities. They can modify web content users see and interact with or extend and change the behavior of the browser itself.

Consider extensions the gateway to making the Chrome browser the most personalized browser.

**Architecture of the ChromeExtension**

An extension’s architecture will depend on its functionality, but many robust extensions will include multiple components:

* Manifest
* BackgroundScript
* UI Elements
* ContentScript
* Option Page

Background Script

The background script is the extension's event handler; it contains listeners for browser events that are important to the extension. It lies dormant until an event is fired then performs the instructed logic. An effective background script is only loaded when it is needed and unloaded when it goes idle.

UI Elements

**UI Elements**

An extension's user interface should be purposeful and minimal. The UI should customize or enhance the browsing experience without distracting from it. Most extensions have a browser action or page action, but can contain other forms of UI, such as context menus, use of the omnibox, or creation of a keyboard shortcut.

Extension UI pages, such as a popup, can contain ordinary HTML pages with JavaScript logic. Extensions can also call tabs.create or window.open() to display additional HTML files present in the extension.

An extension using a page action and a popup can use the declarative content API to set rules in the background script for when the popup is available to users. When the conditions are met, the background script communicates with the popup to make it’s icon clickable to users.

A screenshot of a cell phone

Description automatically generated

*Figure 11*-*Popup.html file*

**Content scripts**

Extensions that read or write to web pages utilize a content script. The content script contains JavaScript that executes in the contexts of a page that has been loaded into the browser.

Content scripts read and modify the DOM of web pages the browser visits.

Content scripts can communicate with their parent extension by exchanging messages and storing values using the storage API.

A close up of a map

Description automatically generated

*Figure 12*-*Plugin Architecture*

So this is how we have acces to the title of the page with the help of the ChromeExtension design, for our case in the background.js file we gave acess to all the pages so that the extension can inject the content.js file wich contains JavaScript code.

We do not have access to all the data from HTML pages because the Extension designed is to work in “different worlds” so we don’t have acess directly to our page because the content script is injected in the webpage that we navigate and the background.js file are inside the ExtensionPage for example the popup.html code will be displayed in our SmartShoppingPlugin but here intervenes **Message Passing** so that we can take whatever content we would like from the WebPages and manipulate in our applications.

**Message Passing[13]**

Since content scripts run in the context of a web page and not the extension, they often need some way of communicating with the rest of the extension. For example, an RSS reader extension might use content scripts to detect the presence of an RSS feed on a page, then notify the background page in order to display a page action icon for that page.

Communication between extensions and their content scripts works by using message passing. Either side can listen for messages sent from the other end, and respond on the same channel. A message can contain any valid JSON object (null, boolean, number, string, array, or object). There is a simple API for one-time requests and a more complex API that allows you to have long-lived connections for exchanging multiple messages with a shared context. It is also possible to send a message to another extension if you know its ID.

Sometimes it's useful to have a conversation that lasts longer than a single request and response. In this case, you can open a long-lived channel from your content script to an extension page , or vice versa, using runtime.connect or tabs.connect, respectively . The channel can optionally have a name, allowing you to distinguish between different types of connections.

One use case might be an automatic form fill extension. The content script could open a channel to the extension page for a particular login, and send a message to the extension for each input element on the page to request the form data to fill in.

The shared connection allows the extension to keep shared state linking the several

messages coming from the content script.When establishing a connection, each end is given a [runtime.Port](https://developer.chrome.com/extensions/runtime#type-Port) object which is used for sending and receiving messages through that connection[13].

In our case the data that we use is the title of the product this is obtained with the help of the content script file figure-12



*Figure 13-ContentScriptMessageEx*

Here is one example of how we have acess to the title of the specific product, of course this is not a general one, for each WebPage the title of the product could be named differently but as we can see there are only 2 lines of code in wich we extract the title and already send it to background script so that our plugin will be aware of this title.

In the background script we add a listener where will espect our value and we wait for the content script to respond at our listener and set the value of the title figure13.

A screenshot of a cell phone

Description automatically generated

*Figure 14-Example call from background Script*

From the figure above we can see that the background script waits for the content script to send the value of the title in a json format, we set the value of the title variable so we can send it into our popup.js file where we make the calls to the SpringBoot server as we can see in the example figure-14.

A picture containing food

Description automatically generated

*Figure 15-Example call from background Script*

After this message is send we have the desired data in the popup.js file where we check if the title is different from null because our content.js file will inject the javascript in every WebPage over the internet even If the user will not click the chromeExtension.

Basically we make a get method to the server in which we deliver as data the title obtained so we can perform our search in back-end site on the mysql database.

We also have an HTML container where we populate the data received and also the image that we scrape in to the database

Ajax Requests is a set of web development techniques using many web technologies on the client side to create asynchronous [web applications](https://en.wikipedia.org/wiki/Web_application).

With Ajax, web applications can send and retrieve data from a [server](https://en.wikipedia.org/wiki/Web_server) asynchronously (in the background) without interfering with the display and behavior of the existing page. By decoupling the data interchange layer from the presentation layer, Ajax allows web pages and, by extension, web applications, to change content dynamically without the need to reload the entire page. In practice, modern implementations commonly utilize [JSON](https://en.wikipedia.org/wiki/JSON) instead of XML.

Ajax is not a single technology, but rather a group of technologies. HTML and CSS can be used in combination to mark up and style information. The webpage can then be modified by JavaScript to dynamically display—and allow the user to interact with—the new information. A screenshot of a cell phone

Description automatically generated

*Figure 16-Example ajax logic[14]*

The built-in [XMLHttpRequest](https://en.wikipedia.org/wiki/XMLHttpRequest) object, or since 2017 the new "fetch()" function within JavaScript, is commonly used to execute Ajax on webpages allowing websites to load content onto the screen without refreshing the page.[14]

Ajax is not a new technology, or different language, just existing technologies used in new ways.

.A screenshot of a cell phone

Description automatically generated

*Figure 17-Ajax Request*

After this call is made the data arrives in SpringBoot Controller on the 8080/productS port where we eliminate the keywords and we make sure that the query will be run with optimized parameters then the server make the search with the help of JDBC Repository figure16.

A screenshot of a cell phone

Description automatically generated

*Figure 17-PartOfSpringBootController*

In the figures below we will present an example of the UI interface looks like and how an user should use the SmartShopping Extension on steps:

* On the first step the user should go on the website where he found the desired product
* At the second step the user should click the SmartShopping plugin icon on the right-side of the page

A screenshot of a social media post

Description automatically generated

*Figure 18-User Steps[19]*

In the figure above we can see where our extension is located in a normal webpage, and the title of the product is underlined because that is the title which we send to the springboot server that will make the search on the database, after that we can se we get a list with the same product but from different site sources and here we can notice that the price offered by the SmartShopping extension is lower than the initially found one.

A screenshot of a cell phone

Description automatically generated

*Figure 19-SmartShoppingUI*

Chapter 5. Implementation Details

**5.1 Frameworks and libraries used**

**Python** is an interpreted, high-level, general-purpose [programming language](https://en.wikipedia.org/wiki/Programming_language). Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.5.x[[33]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-33) and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains [CPython](https://en.wikipedia.org/wiki/CPython), an [open source](https://en.wikipedia.org/wiki/Open-source_software)[34] reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.[16]

The scrapper is build in Python and we use Scrapy as framework so that can make our “spider” and as the interpreter we us pycharm from JetBrains.

Web scraping has become an effective way of extracting information from the web for decision making and analysis. It has become an essential part of the data science toolkit. Data scientists should know how to gather data from web pages and store that data in different formats for further analysis.

Any web page you see on the internet can be crawled for information and **anything visible on a web page can be extracted** . Every web page has its own structure and web elements that because of which you need to write your web crawlers/spiders according to the web page being extracted.

Scrapy provides a powerful framework for extracting the data*, processing it and then save it.*

Scrapy uses spiders, which are self-contained crawlers that are given a set of instructions . In Scrapy it is easier to build and scale large crawling projects by allowing developers to reuse their code.

A screenshot of a cell phone

Description automatically generated

*Figure 20-Scrapy logic*

On the following figure i will explain how I’ve used scrapy for my case so that I can bring the desired data and populate my database, with the help of another chrome extension I’ve managed to find the elements from where I want the data.

*Selector Gadget[18]*

SelectorGadget is an open source tool that makes CSS selector generation and discovery on complicated sites a breeze. Just install the Chrome Extension or drag the bookmarklet to your bookmark bar, then go to any page and launch it. A box will open in the bottom right of the website. Click on a page element that you would like your selector to match (it will turn green).

SelectorGadget will then generate a minimal CSS selector for that element, and will highlight (yellow) everything that is matched by the selector. Now click on a highlighted element to remove it from the selector (red), or click on an unhighlighted element to add it to the selector.

Through this process of selection and rejection, SelectorGadget helps you come up with the perfect CSS selector for your needs[18].

A screenshot of a social media post

Description automatically generated

*Figure 21-Selector Gadget Ui[18]*

Chapter 6.Conclusion

Chapter 7.References

[1]*Dealcious(Verified)* : accesed on 13:05:2020

<https://chrome.google.com/webstore/detail/dealiciousverified/pplphlbogdadpkmdhbjdebhfhjifbfka?hl=en>

[2][Bradford] Bradford A: *8* *Chrome extensions that can save you hundreds of dollars,* accesed on 13:05:2020

<https://www.cnet.com/pictures/chrome-extensions-that-can-save-you-hundreds-of-dollars/>

[3]*The Camelizer* : accesed on 13:05:2020

<https://chrome.google.com/webstore/detail/thecamelizer/ghnomdcacenbmilgjigehppbamfndblo?hl=en>

[4]*Maven:*

<https://jira.moreover.com/browse/ND-20723>

[5]*Getting Started Turorial* : accesed on 13:05:2020

<https://developer.chrome.com/extensions/getstarted>

[6]*Overview:* accesed on 13:05:2020

<https://developer.chrome.com/extensions/overview>

[7]*jQuery:* accesed on 13:05:2020

<https://jquery.com/>

[8]*MySql:* accesed on 13:05:2020

<https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>

[9]*Spring Framework* : accesed on 14:05:2020

<https://en.wikipedia.org/wiki/Spring_Framework>

[10]*JDBC Template:accesed on 14:05:2020*

<https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/jdbc/core/JdbcTemplate.html>

[11]*REST API:* accesed on 14:05:2020

<https://en.wikipedia.org/wiki/Representational_state_transfer>

[12]Web crawling: accesed on 14:05:2020

<https://en.wikipedia.org/wiki/Web_crawler>

[13]Chrome Extensions: accesed on 14:05:2020

<https://developer.chrome.com/extensions/overview>

[14]Ajax Requests: accesed on 14:05:2020

<https://en.wikipedia.org/wiki/Ajax_(programming)>

[15]Shopping: accesed on 15:05:2020

<https://en.wikipedia.org/wiki/Shopping>

[16]Python: accesed on 15:05:2020

<https://en.wikipedia.org/wiki/Python_(programming_language)>

[17]Making Web Crawlers Using Scrapy for Python: accesed on 20:05:2020

<https://www.datacamp.com/community/tutorials/making-web-crawlers-scrapy-python?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=278443377092&utm_targetid=dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=20897&gclid=EAIaIQobChMI6pCWjLXv6QIVyIGyCh2AsQdaEAAYAiAAEgLwEPD_BwE#overview>

[18]Selector Gadget: point and select CSS selectors: accesed on 15:05:2020

<https://selectorgadget.com/>

[19]Emag product: accesed on 15:05:2020

<https://www.emag.ro/telefon-mobil-apple-iphone-xs-64gb-space-grey-mt9e2rm-a/pd/D5SY9VBBM/?X-Search-Id=b9868662c6cc2e2c6bb6&X-Product-Id=37248908&X-Search-Page=1&X-Search-Position=0&X-Section=search&X-MB=0&X-Search-Action=view>