International Journal of Differential Equations and Applications
Volume **20**, No. 2 (2021), pages: 157-168
ISSN (Print): 1311-2872; ISSN (Online): 1314-6084;
url: https://www.ijdea.eu

ijdea.eu

AUTOMATED PRODUCT INFORMATION RETRIEVAL IN E-COMMERCE

Stanislav Dakov[1], Anna Malinova[2]

[1,2]Faculty of Mathematics and Informatics
Paisii Hilendarski University of Plovdiv
236 Bulgaria Blvd., 4003 Plovdiv, BULGARIA

ABSTRACT: The recent paper presents automatic retrieval of publicly available information about products on the Internet, analysis and aggregation of this information and sending the relevant notifications to the users. A Telegram bot has been developed, which periodically checks the information for certain products and analyzes the retrieved data. This saves customers periodic visits to online stores. A number of commands have been developed and integrated in the bot that can be used by the users. We give examples of using the developed application for retrieving information about product availability, increasing or decreasing price, product reviews, posting new ads on sales sites. Due to the lack of a unified standard for presenting product information in e-commerce, it was necessary to develop adapters for each site. The application is based on techniques for web data extraction, microservices and cron jobs. Possibilities for further development of the presented application are also considered.

AMS Subject Classification: 68N01, 68N19
Key Words: automated retrieval of information, web scraping, e-commerce, telegram bot, data collection, product price

Received: August 23, 2021 **Revised:** November 3, 2021
Published: November 15, 2021 **doi:** 10.12732/ijdea.v20i2.2
Academic Publications, Ltd. https://acadpubl.eu

1. INTRODUCTION

Nowadays e-commerce has become anintegral part of our daily lives [17]. Increasingly,

customers prefer the Internet to buy goods [18]. Often e-shops provide better promotions of certain products than the corresponding physical stores. The same products can be quickly found on different sites, but at different prices or a product's price can change many times in the same site. This causes buyers to store links of products and check them periodically for promotions or other changes. Same goes when the product is out of stock. The buyer still has to check the shop for product availability. Some online shops have built-in functionality to notify the customer when the product gets in stock, but not all of them offer that. This paper provides a generic solution for automatically monitoring products and notifying the user when the product data has changed, or a new product has been added.

There are four main techniques to collect information about a product from an e-commerce website:

- Using a predefined file with products' details provided to the tracking system by online shops. One of the most famous platforms for monitoring products' prices in Bulgaria is pazaruvaj.com. It is preferred because it has been operating for a long time and already has a fairly large range of stores and products – over 4,000,000 products from 761 stores [7]. Adding products from an online shop to that system is done through a predefined XML or CSV file. These files can be generated automatically, but this depends on the e-commerce platform that has been used for creating the given online shop, otherwise they can be created manually. Once the product information has been provided by the online stores, pazaruvaj.com gives information about the best offer for a given product.

- Using RSS feed [19] – RSS stands for Really Simple Syndication, and it is a simple, standardized content distribution method that can help users stay up to date with their favorite newscasts, blogs, websites, and social media channels. The RSS of an online store contains a list of all products with current prices, thus other applications can retrieve the latest information about a product without the need of parsing HTML content.

- Rest API [5, 20] – some websites provide APIs to access parts of their data. But even though these sites provide APIs, there still exist some data fields that we cannot scrape or have no permission to access. For example, Amazon provides a Product Advertising API [22], but the API itself does not provide access to all of the information displayed on a product page. In this case, the only way to scrape more data, say the price data field, is to build our own scraper or use ready-made scraper tools.

- Using web scraping (also screen scraping, web data extraction, web harvesting, etc.) [1, 2] – this is a technique for automatically extracting data from websites and saving the extracted data to a local file or to a database. A web scraping software can crawl multiple pages within a website and automate the time-consuming task of manually copying and pasting the data displayed. Most often custom web scrapers are developed, although there exist ready-made systems, e.g., https://www.scraping-bot.io/, https://www.scrapingbee.com/, https://scrapingrobot.com/, etc. Web scraping technology is mainly used for:
  - Search engine bots: web crawlers sent out from the world's biggest web sites to index content for their search engines and social media platforms, e.g., Googlebot, Baiduspider, MSN Bot/Bingbot, Yandex Bot, etc.
  - Market research companies: scrape pricing and other information on products from e-commerce websites, vehicles on dealership sites, trips on travel sites or property information from real estate sites.
  - Scraping lead information from directories: either individual contact information or company information to populate CRMs. For example, scraping platforms such as Yelp or Yellow Pages.
  - Academic research: collect data from sites for various research purposes.
  - Build aggregators that collect blog posts, classified ads or jobs.
  - Scrape data from an old website to move the content over to a new website, where exporting or API are not available.
  - Scrape reviews and comments for sentiment analysis.

The rest of the paper describes the software application we have developed for automatically extracting product information from websites, aggregating this information, and sending notifications to the user. Some of the possible uses of the developed application are retrieving information about product availability, increasing or decreasing price, product reviews, new ads posted on sales sites.

2. BUILDING THE WEB SCRAPING BOT

The application that we have developed is a web scraping bot that uses Telegram messaging and consists of three separate microservices:

1) Web scraper

2) Command listener

3) Product scanner

The web scraper service provides functionality to get publicly available information for a product. The command listener is responsible for controlling the Telegram bot. It uses a cron job to periodically check and process commands submitted by the user. Based on what command is executed, the command listener gives a corresponding answer to the user. The product scanner is a service that takes care of updating the data for the monitored products. It periodically checks the current data of products such as price, availability, reviews, and rating. If there are differences with the previous product parameters, the service sends a message to the user.

Since the presented application uses web scraping to extract information about a product, one of the problems that arose was the lack of unified presentation of product metadata. Many sites do not use a single standard to present product data. There are couple of standards that can be used to represent microdata [11, 12, 13, 14]:

- OpenGraph – Facebook suggests the use of meta OG tags [10]. They contain summary of the product data.
- Schema.org [15] – a collaborative community activity with a mission to create, maintain, and promote schemas for structured data on web pages, in email messages, etc. It is supported by Google and other major search engines.
- JSON-LD [21] – a lightweight Linked Data format. It is based on the JSON format and provides a way to help JSON data interoperate at Web-scale.

If each store uses a standard, the process of retrieving information will become very easy and fast. The idea is to have standardized representation of the microdata for the product. But if we want to extract something more than basic information such as price, name, description, or feedback, we will still need a separate parser for each store. Since most online stores do not use standard representations, the application only supports certain platforms. Currently it supports Emag, Amazon and Ebay. We have developed a separate parser for each store. Emag uses the standard JSON-LD to represent the microdata for a product, so this parser can be used for any site that uses this standard.

Figure 1 presents the web scraper service. The communication with it is done by HTTP requests. It retrieves the product URL by a POST request and then gets the source code of the page using a GET request. Then the algorithm checks where the

page came from and based on the domain name, it selects a specific parser. To parse the source code, it uses different HTML parsers:

– DOMDocument and DOMXPath

– Symphony DomCrawler [9]

– PHP Html Parser

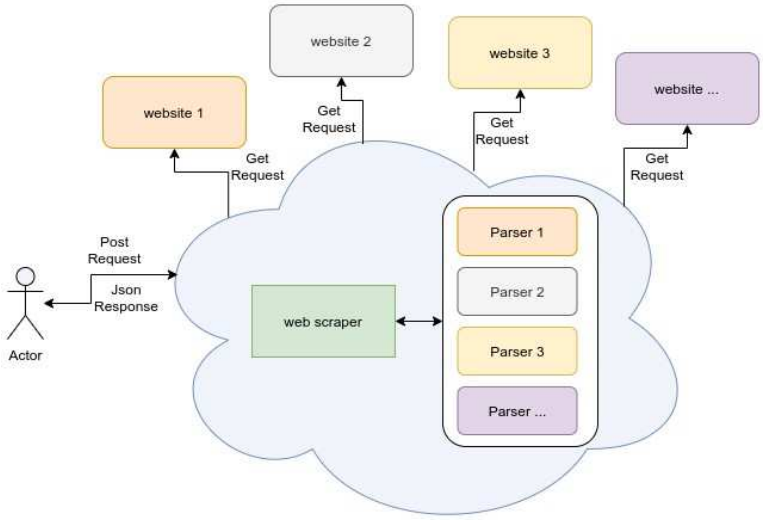

Figure 1. Web scraper service

After analyzing the source code, the web scraper service generates a Product object and returns it in JSON format. Figure 2 shows all of the product object properties.

Figure 2. Product object properties

The command listener, shown on Figure 3, and product scanner, shown on Figure 7, use the web scraper service to retrieve information about a product. When they get that information, they send custom notifications to the user using Telegram – a platform for messaging communication, similar to Facebook Messenger, WhatsApp, Signal and others. It is a cross platform mobile messaging app for smart phones and tablets for iOS and Android. It has desktop and web integration as well. Telegram allows users

to send and receive messages, photos, and have secure conversations. Telegram has a free REST API [5, 20], which provides a list of commands through which it controls or creates bots [8, 16]. It has easy integration with different programming languages. The integration is relatively easy [6]. A Telegram bot is created and managed by Telegram itself, using "BotFather" commands – Telegram's main bot, which controls all other bots. A new bot is created by using the command /newbot. It then asks for a name and once created, returns a token which is used for communication with the Telegram API. The application uses curl to make requests to the API, but there are many libraries that can be used:

- Telegram Bot API Base – clear and simple Telegram Bot API.
- PHP Telegram API – a complete async capable Telegram Bot API implementation for PHP7.
- PHP Telegram Bot – PHP Telegram Bot based on the official Telegram Bot API.
- Bot API PHP SDK – a Telegram Bot API PHP SDK. Supports Laravel out of the box.
- TeleBot – presents an easy way to create Telegram bots in PHP. Rich Laravel support out of the box.
- NovaGram – an Object-Oriented PHP library for Telegram Bots.
- PHP Telegram Bot.
- PHP Telegram Bot Api – native PHP Wrapper for Telegram BOT API.
- TuriBot – a simple way to communicate with Telegram APIs in PHP.
- TelegramBotApiBundle – a Symfony wrapper bundle for Telegram Bot API.

The whole process is controlled by the user through commands and for this purpose a command lister periodically checks for certain commands. Telegram allows the implementation of custom commands, too. When sending a command, the service periodically checks what command the user has sent and decides what action to perform. We have implemented the following eight commands:

- /help – shows basic information about the bot.
- /watch product [url] – add a new item to the watchlist, [url] is the product's url.

– /watch shop [url] – add a new item to the watchlist, [url] is the product's url.

– /list product – shows all watched products.

– /list shop – shows all watched shop lists.

– /remove product [productId] – remove product from the watched list.

– /remove shop [productId] – remove shop list from the watched list.

– /history [productId] – shows the price history of a specific product.

When sending a command to add a new product, the bot sends a POST request containing the product URL to the scraper service. Then the scraper service returns the product object. After that the object is saved in the database along with information about which user it was sent from. It then sends a message to the user which says that the product has been saved. It is possible for the user to send a link with a product that is not yet supported. The product is still stored in the database, but it is marked as not supported.

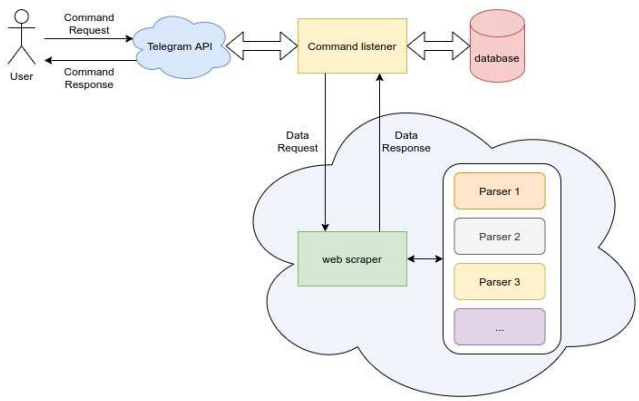

Figure 3. Command listener service

Telegram also allows the implementation of inline buttons. Adding a button is done by sending additional options to the command. These buttons improve user interaction. On Figure 4 is shown the result of using the /help command – the user can see all other commands and use the buttons that perform the same actions.

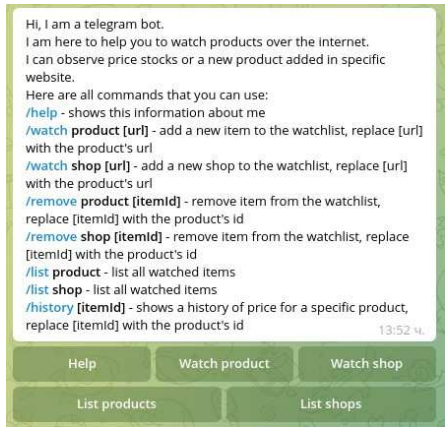

Figure 4. Simple bot information and commands

By pressing the "Help" button, the "/help" command is executed. When a user clicks the "Watch product" button, the command "/watch product" is executed, then the bot sends a message that it is waiting for the user to send back the URL of the product. When pressing the "Watch shop" button, the command "/watch shop" is executed, then the bot waits for the user to enter the URL for a list result of a product search in an e-commerce website. When the "List product" button is pressed, the "/list product" command is executed, and the bot returns a list of all added products by the user. Figure 5 shows a listed product. When the "List shop" button is pressed, the "/list shop" command is executed, whereby the bot returns a list of all added stores for the given user. Similar buttons are added to each product when it is being displayed. This makes it easier for the user to open a link to a product, delete it or see the price history.

Figure 5. Example of a listed product

When pressing the "Open link" button, the user is redirected to the product in the online store. When the "Remove" button is clicked, the "/remove product [productId]" command is executed, and the product is deleted. Then the bot sends a message back to the user about successfully deleting the product. When the user clicks the "History" button, the "/history [productId]" command is executed and returns the price history for a given product. Figure 6 shows the history of a product's price.

Figure 6. Example of a product price history

The product scanner service takes care of updating the data for each registered product. The current data for each product is checked. When starting the scanning process, the service takes all supported products from the database and calls the Web scraping service by sending a POST request with the product URL. The web scraping service returns information – then the algorithm compares the differences between the current data and that in the database.

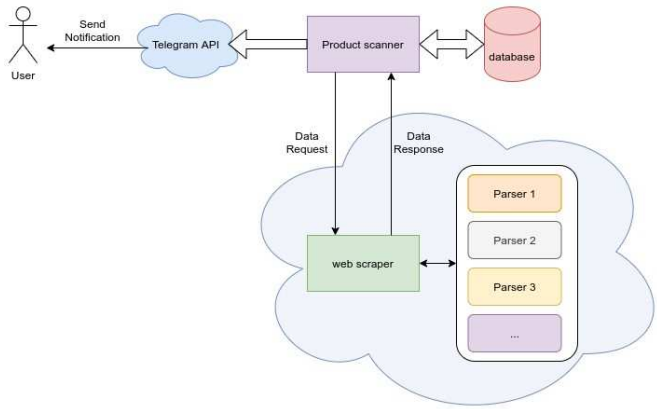

Figure 7. Product scanner service

If the service finds any difference, it sends notifications to the end user depending on what the difference is:

– Price change

– Change in availability

– New opinion

– New rating

– New product has been added to a shop

When changing product data, the scanner sends a message to the user with the new product data. Figure 8 shows an example of a notification when the price increases.

Figure 8. Example of product with increased price

When the scanner finds a new product in a list, it sends a notification to the user with the data for the new product (Figure 9). Then the user has the option, by pressing the buttons, to add the product directly to the watch list or if they decide – to delete the watchlist.

Figure 9. A new added product

3. CONCLUSIONS

It can be concluded that a Telegram bot is suitable to be used with a web scraping tool to perform repeated HTML parsing tasks and user notifications. The bot successfully tracks products in e-commerce websites, and the user gets notified when the price has been changed and when a new product has been added. In the future more e-commerce website parsers can be added, as well as a command listener which uses webhooks. Most online stores are developed using ready-made platforms. Each e-commerce framework has a specific structure by which it represents the product data. For future development, the possibility of generic parsers for the largest e-commerce platforms will be considered: Bigcommerce; Magento; Prestashop; Shift4Shop; Shopify; Squarespace; Volusion; Weebly; Wix; WooCommerce.

ACKNOWLEDGMENTS

This paper is supported by the project FP21-FMI-002 of the Scientific Fund of the Paisii Hilendarski University of Plovdiv, Bulgaria.

REFERENCES

[1] J. Hillen, Web scraping for food price research, British Food Journal, (2019), Vol. **121**, No. 12, 3350–3361, ISSN: 0007-070X, DOI: 10.1108/BFJ-02-2019-0081.

[2] Z. Bo, Web scraping, Encyclopedia of Big Data, Springer International Publishing, (2017), 1–3, DOI: 10.1007/978-3-319-32001-4_483-1.

[3] Telegram. 2021. *Bots: An introduction for developers.* Retrieved on September 5, (2021), from https://core.telegram.org/bots.

[4] R. Nufusula, Rancang Bangun Chat Bot Pada ServerPulsa Mengunakan Telegram Bot API, *Journal of information system*, (2018), 80–88, DOI: 10.33633/joins.v3i1.1884.

[5] S. Lyu, REST APIs, Practical Rust Web Projects, (2021), 55–102, DOI: 10.1007/978-1-4842-6589-5_3.

[6] N. Modrzyk, Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API, *Apress; 1st ed. edition*, (2019), DOI: 10.1007/978-1-4842-4197-4.

[7] Pazaruvaj, 2021, https://www.pazaruvaj.com/.

[8] D. Ismawati, The Development of Telegram BOT Through Short Story, *Proceedings of the Brawijaya International Conference on Multidisciplinary Sciences and Technology*, (2020), 209–212, DOI: 10.2991/assehr.k.201021.049.

[9] Symfony, *The DomCrawler Component*, Retrieved on September 5, (2021), from https://symfony.com/doc/current/components/dom_crawler.html.

[10] Facebook, *Product Object*, Retrieved on September 6, (2021), from https://developers.facebook.com/docs/payments/product/.

[11] P. Jimenez, On the synthesis of metadata tags for HTML files, *Software: Practice and Experience*, (2020), 2169–2192, DOI: 10.1002/spe.2886.

[12] R. Meusel, The WebDataCommons Microdata, RDFa and Microformat Dataset Series, *The Semantic Web – ISWC'2014*, (2014), 277–292, DOI: 10.1007/978-3-319-11964-9_18.

[13] P. Petrovski, Integrating product data from websites offering microdata markup, *Proceedings of the 23rd International Conference on World Wide Web*, (2014), 1299–1304, DOI: 10.1145/2567948.2579704.

[14] A. Kannan, Matching Unstructured Product Offers to Structured Product Descriptions, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2011), 404–412, DOI: 10.1145/2020408.2020474.

[15] J. Ronallo, HTML5 Microdata and Schema.org, *Code4Lib J*, (2012), Issue 16, ISSN: 1940-5758.

[16] M. Rosid, Integration Telegram Bot on E-Complaint Applications in College, *IOP Conference Series: Materials Science and Engineering*, (2018), 1757–8981, DOI: 10.1088/1757-899x/288/1/012159.

[17] V. Jain, An Overview of Electronic Commerce (e-Commerce), *Journal of Contemporary Issues in Business and Government*, (2021), **27**, 665–670, DOI: 10.47750/cibg.2021.27.03.090.

[18] T. Raval, Study of Effectiveness of Online Shopping, *Indian Journal of Applied Research*, (2014), 76–78, DOI: 10.36106/ijar.

[19] F. Getahun, Multi-Query Optimization on RSS Feeds, *J Data Semant*, (2018), **7**, 47–64, DOI: 10.1007/s13740-018-0085-3.

[20] B. Williams, REST API, *Professional WordPress Plugin Development*, (2020), 279–314, DOI: 10.1002/9781119666981.ch12.

[21] M. Lanthaler, On using JSON-LD to create evolvable RESTful services, *WS-REST '12: Proceedings of the Third International Workshop on RESTful Design*, (2012), 25–32, DOI: 10.1145/2307819.2307827

[22] Amazon's Product Advertising API 5.0, 2021, https://webservices.amazon.com/paapi5/documentation/.