VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Problem-Based Project

Cheaper - Supermarket price aggregation website

Done by:
Ahmet Berk Koc
Dovydas Juodkazis
Paulius Mieščionaitis
Robertas Timukas

Supervisor:
dr. Linas Būtėnas

Vilnius
2022

Contents

Abstract

This report focuses on the development of a supermarket price aggregation website. The project is based on *pricer.lt, kaina24.lt, kainos.lt websites*. The main goal of this website is to list as many products from as many supermarkets as possible, so that people could check for discounts or prices of certain products before they go to go looking for that product. The project aims to make looking up best deals for any product as simple and convenient as possible.

Santrauka

Šiame dokumente yra aprašyta parduotuvių kainų palyginimo svetainės kūrimo procesas ir reikalavimai. Šis projektas yra paremtas *pricer.lt, kaina24.lt, kainos.lt* svetainėmis. Pagrindinė šios svetainės paskirtis yra leisti naudotojui surasti betkokią norimą prekę bei jos kainą kuo įmanoma paprasčiau ir greičiau.

Introduction

This report describes the process of making a supermarket price aggregation website. The website's design is compatible with pc and mobile. The document will discuss the game overview including vision of the project, analysis, diagrams, descriptions, motivating examples and case study.

Terminology

Often used abbreviations for this project:

- Scrapy - a similar word to scraping, after which a Python Framework dedicated for web scraping and crawling.
- spider - a spider is a script, specifically written to access, extract and parse and save the data from web page.

1 Requirements and Suggestions of the Supervisor

After a meeting with our supervisor our team was presented with a list of requirements for developing a website. Here are a few of major ones:

- The emphasis is on optimization;
- Use of correct HCI principles;
- Simplify the interface, it should be minimalistic and easy to navigate through.
- Make a web crawler which gathers data of products from supermarket websites.

2 Analysis of related work

Price comparison websites aren't something unheard of. Before beginning the "Cheaper" website project a couple of competitor websites were tested and analyzed. The reason behind this analysis was to find out what functionality is mandatory for such a website and how everything should be presented to the user in the most efficient way. Even though there are several websites already available for price comparison, and most of the e-commerce websites are present in them, the user interface is quite different in a few of them.

To begin with the analysis, 3 most known websites for price comparison in lithuanian e-commerce shops were chosen ('pricer.lt', 'kainos.lt' and 'kaina24.lt'). These websites are considered the main competitors.

The first website that was analyzed was 'pricer.lt'. The initial view of the website after first visiting it did not leave any special emotions. The front-end of the website was quite messy until visited with a smartphone, which shrinked it and made it look a little more high-end. This website seems to have other products scraper from e-commerce websites as well as manually uploaded products with picutres taken with a mobile phone. [1]

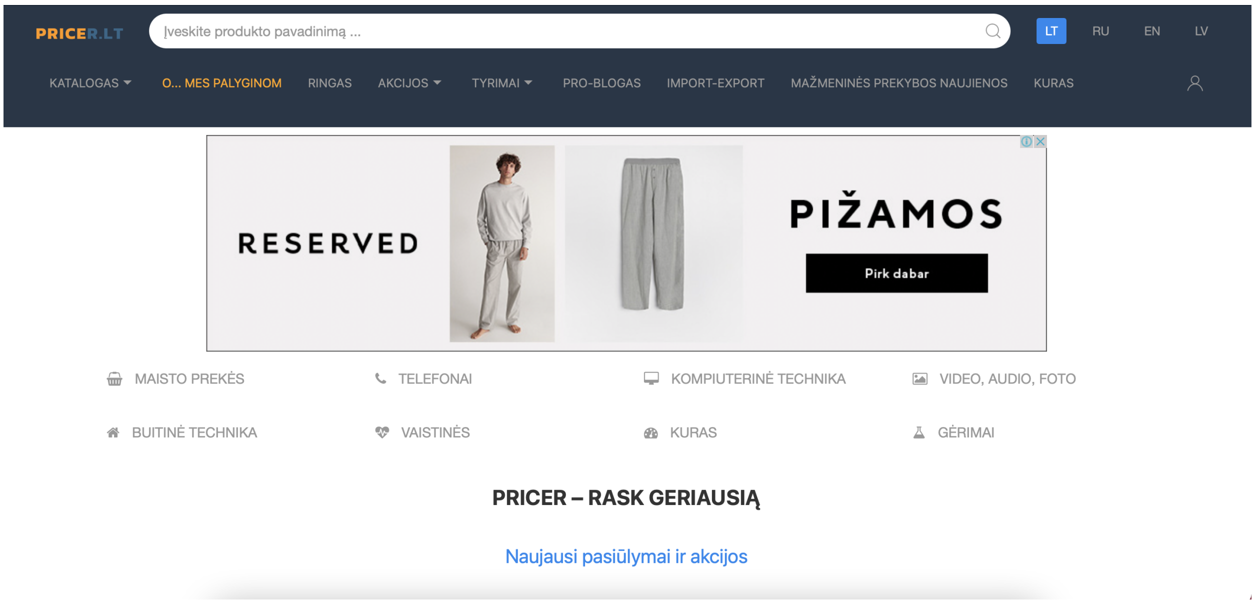

Figure 1. pricer.lt main page

The next website in our list was ’kainos.lt’. Since the first visit this website seemed like the most advanced one, it was perfectly finished, had categorising as well as search functionality presented to the user. It also had product template pages with several websites shown on where to find this product for different prices. [2]

The last website that analyzed was ’kaina24.lt’. It looks just like the last website from our analysis and seems to have just different styling. The functionality is the same one as the previous one too. The only unique thing noticed was that this website presented the user with a chart that showed price history of a chosen product. [3]

After analyzing the competitor websites it turned out that most of the price comparison websites are similarly structured and presents the user with similar user-interface sections beginning with categories, moving to products and eventually being redirected to the chosen product page.

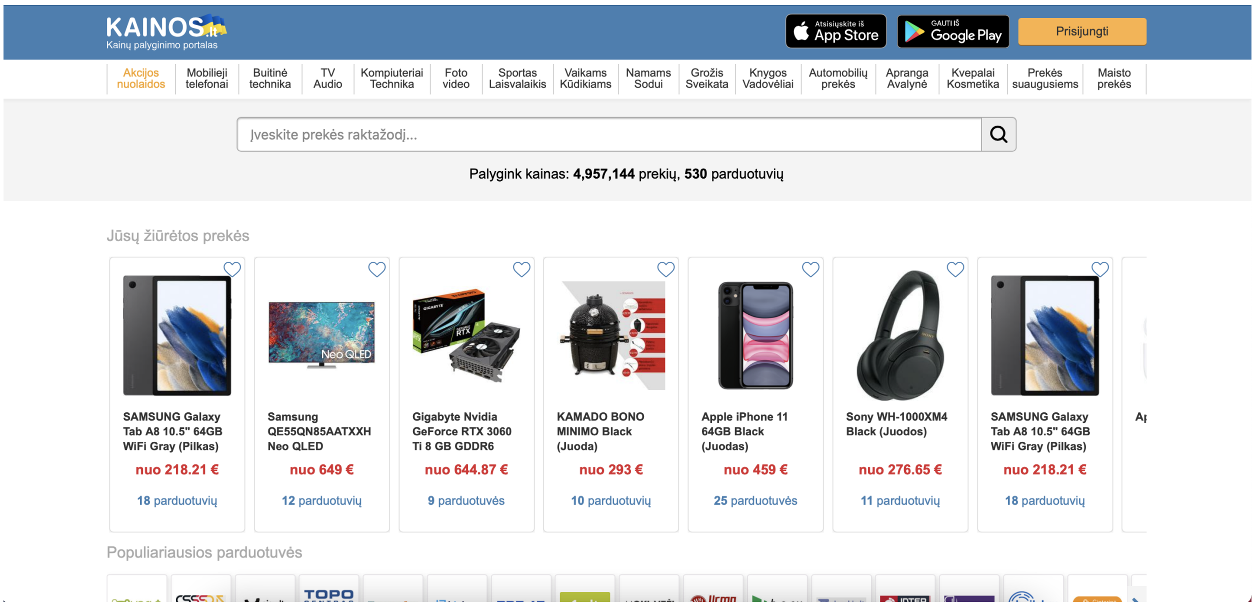

Figure 2. kainos.lt main page

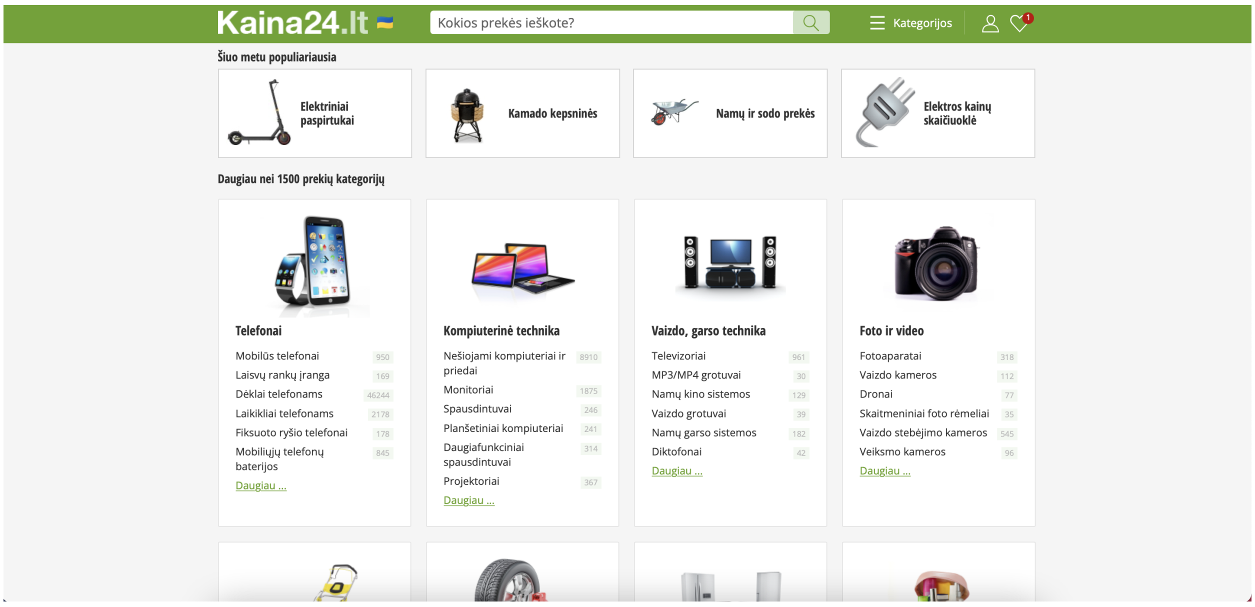

Figure 3. kaina24.lt main page

3 Architecture

3.1 Virtual Machine Configuration

This project mainly consists of three main parts: a web server which is the key part of serving products with their prices, a database for storing said product data and finally - a web crawler, the service that collects data from hand-picked shop websites. These services are described in [4].

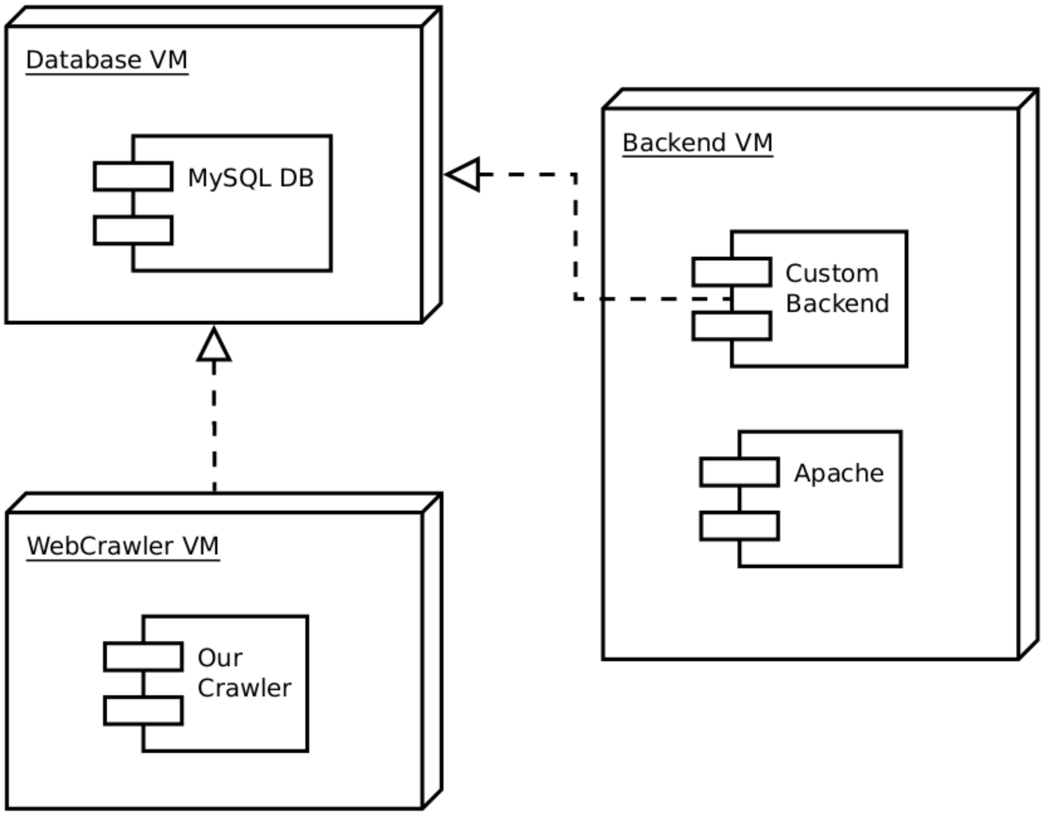

Figure 4. Deployment diagram

3.2 Web-Server Virtual Machine

It is by far the most complex configuration in this projects architecture. It houses two components: front-end app which is served by Apache2 HTTP daemon and custom back-end using ExpressJS with NodeJS as SystemD service daemon.

3.3 Web-Crawler Virtual Machine

This part of the project's component is mainly designed with Python language using the Scrapy framework. The framework brings existing functionality, written for the user to quickly develop a simple web-crawler to pool data from the internet. But with it comes a cost of developer having to fit their code inside a frame and API in order to work properly which was felt while trying to develop a solution for transferring the data to a remote database.

3.4 Database Virtual Machine

This is the last component of this project. By far the simplest of them all and used mainly to run MySQL database server, which on Debian systems by default is MariaDB, a derivative of MySQL. At first, it was decided to use MongoDB rather than MySQL or any other Relational Data Base Management System (RDBMS) due to the way it manages data which would make it easier to store and retrieve JSON format data. But due to complications of trying to run it on unsupported hardware, the decision was made to switch to RDBMS type of database that can be installed via the systems package manager.

4 Analysis

4.1 Description

This is a e-commerce product price aggregation website. The main goal of this website is to list as many products from as many supermarkets as possible so that people could check for discounts or prices of certain products before they go to go looking for that product. The project aims to make looking up best deals for any product as simple and convenient as possible.

4.2 Functional Requirements

This section lists website's functional requirements

1. User interface and User Experience
    - Easy to find and use search functionality;
    - Easy to compare price for same products in other supermarkets.
2. Back-end
    - Can manage data between front-end and database;
    - Can manage product loading;
    - Can manage filtering.
3. Web crawler
    - Can get product's URL;
    - Can get product's name;
    - Can get product's price;
    - Can get product's image's URL;
    - Can get superstore's name;
    - Can get superstore's logo URL;
    - Can get a date when product was last scraped;
    - Can refresh or add product's information to the database.

4.3 Non-Functional Requirements

This section lists website's non-functional requirements

1. User interface and User Experience

   - The purpose of pleasing and efficient User interface is to enable user to use website's functionality.
   - **User interface consists of:**
     - Navigation bar:
       * Website logo;
       * Search bar;
       * Filter selection.
     - Product cards:
       * Store logo;
       * Product image;
       * Product price;
       * Previous price;
       * Date when price was last updated;
       * Button which redirects to the product's store;
       * Button which shows price comparison popup dialog.
     - Price comparison popup dialog:
       * Store logo;
       * Product price;
       * Button which redirects to the product's store;
     - Button which loads more products;
     - Button which returns user to the top of the page.

2. Back-end

   - The purpose of back-end service is to manage data traffic between front-end and database securely.
   - **Main functionality for back-end:**
     - Manage search queries;
     - Manage product data nesting;
     - Manage extra filters;
     - Manage product order.

3. Web crawler

   - The purpose of web crawler is to gather information about products from various supermarket websites.
   - **Main functionality for web crawler:**
     - Get information about the product;

  - Check for product matches in the database;
  - Update information about product in the database if needed;
  - Add new product to the database

5 Data Collection

This project is based on comparing prices between different e-commerce websites in a wide variety of shops online. In order to have a big enough data set information about as many as possible items that are for sale online is needed. Looking for and saving such data manually is the most inefficient way for such a job. This project is using a self built data scrapers (also known as spiders) that collect the data from different websites. This way a big data set of information is collected quickly, with all the required information about the item. For this task Python alongside with Scrapy framework is used.

5.1 Scraping approaches

5.1.1 HTML parsing

The first way of extracting data from the website is parsing its HTML code. A request is sent to the provided WEB address, which returns an HTML file alongside its CSS information. The script (a spider) parses the HTML by provided instructions. For spider to know what parts of the code it needs to parse, we provide it selectors, either XPath or CSS. The selector work similarly as a file path, by guiding the script where to look for a specific tag with specific attributes in the HTML code. After the selector is defined, the script either extracts the information about an attribute (ex. <img> tag, src="/link/to/image" attribute, which contains a link to the picture that is shown) or extracts the text information in between the tags, like price (ex. <span>6.99 EUR</span>). The spiders are also able to visit other pages from found <a> tags and their 'href' attributes, which lets the spider to collect data from every existing page on the website.

5.1.2 API scraping

Sometimes websites have their information API's open to the public (it can be found by using Dev Tools within a web browser ['inspect element' in Chrome]). After opening up a website, you can view what fetch requests were sent to the server and what information was retrieved back to the users side of the wall while the page is loading. Later on, you can access these links which usually have information inside of them written in JSON format. Then all you have to do is retrieve the desired information from the given data sets. May it be links to different categories of products in this case, or the product set itself.

5.1.3 Which one to use

Both of these approaches have their own advantages and disadvantages which make the choosing on how to scrape a specific website a little more complicated. Firstly, API scraping is much faster approach than HTML parsing, because more data can be accessed instantly, than parsing the HTML (ex. you have to visit every single page in HTML, which requires sending a request to every single pagination link, to gather all the products, while accessing the JSON files through an API, you have a bigger data set provided to you faster). Unfortunately, in most of the cases such APIs are not accessible in the website and not visible to the user and we must parse the HTML. Parsing HTML is a simpler approach, even though it requires more computer resources, such spiders are faster to build if you have a structured way of writing the scripts. Despite that, some scraping frameworks, and in this case Scrapy, which we are using, are unable to pre-render javascript, which sometimes results in heavily javascript populated websites not to load the whole HTML code. At the end of the day, choosing the approach to scrape a website should be decided after reviewing the website and its structure.

5.2 Product information

A product in this case is like an object, which is a template for the information to be filled while scraping the websites. For CHEAPER price comparison website, we require to extract several points of data for our product to later on not only successfully display it in our website, but also let the visitor to follow the link where the item can be purchased. The information that is being scraped about the products:

- Link to the product
- Title
- Price
- Image URL

5.3 Saving scraped Data

Every single product that is scraped from the website, is appended to a JSON file. After the Scrapy spider process is finished, the JSON file is saved and ready to use to display the collected data in the Front-End of the application.

5.4 Website-friendly scraping

Scraping can be actually harmful for the website that is scraped. Scrapy is an asynchronous framework, that is able to send a very large number of requests at once to the same website. This can often result in overloading of the server and if the worst occurs - the website is crashed. That is why it should be evaluated and delays should be put on request sending time.
For this reason every website's robots.txt file must be checked. You can access it by adding '/robots.txt' extension at the end of the website's URL (ex. 'www.facebook.com/robots.txt'). This webpage will contain the urls that the crawler is allowed access.

5.5 Data Matching

This project requires a highly accurate matching of items so they can be grouped and later compared to each other. To achieve this functionality we used a technique called fuzzy matching.
Generally speaking, fuzzy searching (more formally known as approximate string matching) is the technique of finding strings that are approximately equal to a given pattern (rather than exactly).
Each product name is matched to all the other existing product names, they are grouped together if if 70 percent of the name matches.

6 Data Storage

6.1 Database structure

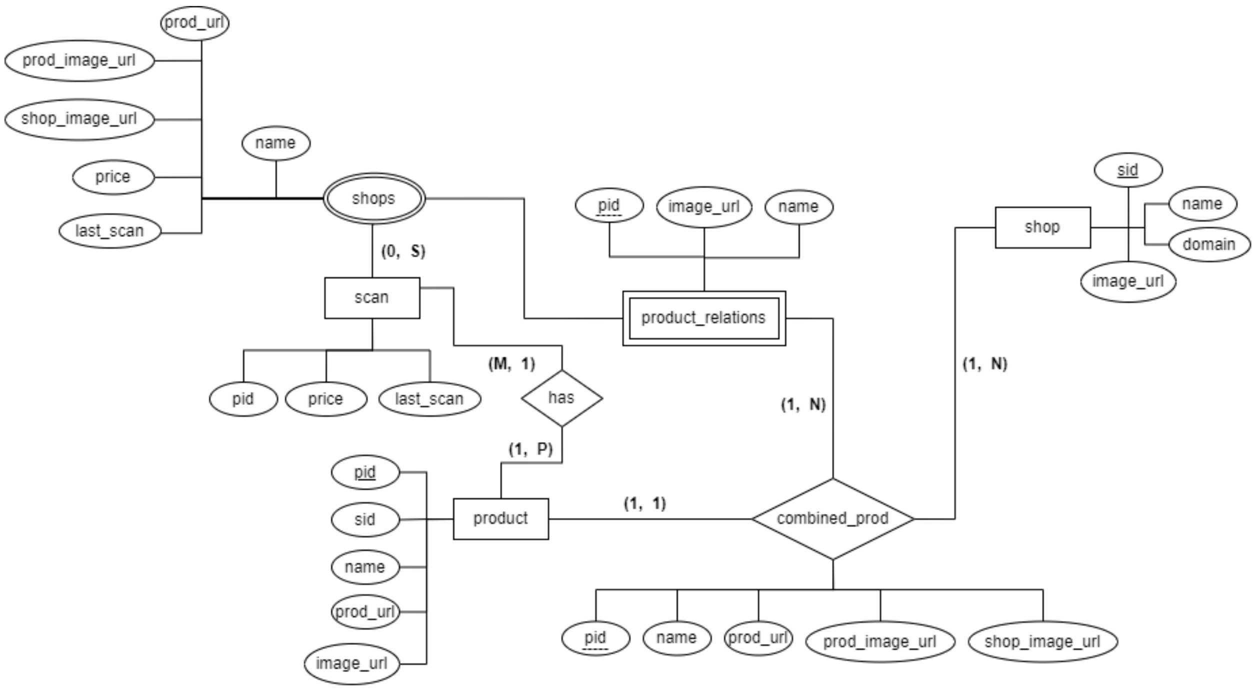

Figure 5. Database ER model

7 References

7.1 Competitor websites

- https://www.kainos.lt/
- https://www.kaina24.lt/
- https://pricer.lt/

7.2 Frameworks for backend

- https://www.sequelize.org/
- https://www.fusejs.io/
- https://github.com/sequelize/umzug