
**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
TÎRGU-MUREŞ
SPECIALIZAREA CALCULATOARE**

Price Monitor

PROIECT DE DIPLOMĂ

**Coordonator științific:
Dr. Szántó Zoltán**

**Absolvent:
Palfi Szabolcs**

2021

LUCRARE DE DIPLOMĂ

Coordonator științific: DR. SZÁNTÓ ZOLTÁN | Candidat: PALFI SZabolcs
Anul absolvirii: 2021

a) Tema lucrării de licență: Price Monitor: Dezvoltarea unei aplicații pentru monitorizarea prețurilor unor produse ale diferitelor magazine online.

b) Problemele principale tratate:

- Dezvoltarea aplicațiilor mobile, a extensiilor pentru browser
- Achiziționarea datelor publice de pe paginile web ale magazinelor
- Prelucrarea datelor obținute

c) Desene obligatorii:

- Schema de bloc a aplicației
- Diagrame UML privind software-ul realizat

d) Softuri obligatorii:

- Webscraper implementat în Python folosind librăria BeautifulSoup
- Extensie Chrome
- Aplicație mobila realizată în Flutter
- Software care realizează colectarea, stocarea și actualizarea periodică a datelor

e) Bibliografia recomandată:

- Mahto, Deepak Kumar, and Lisha Singh. "A dive into the Web Scraper world." 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACOM). IEEE, 2016.
- Ujwal, B. V. S., et al. "Classification-Based Adaptive Web Scraper." 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2017.
- Upadhyay, Shreya, et al. "Articulating the construction of a web scraper for massive data extraction." 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE, 2017.
- Ullah, Habib, et al. "Web Scraper Revealing Trends of Target Products and New Insights in Online Shopping Websites." INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS 9.6 (2018): 427-432.
- Chaulagain, Ram Sharan, et al. "Cloud based web scraping for big data applications." 2017 IEEE International Conference on Smart Cloud (SmartCloud). IEEE, 2017.
- Noor, Mohd, and Nisa Asila. Price comparison website using web scraping/Nisa Asila Mohd Noor. Diss. Universiti Teknologi MARA, 2016.
- Android Development with Kotlin by M. Moskala, I. Wojda, 2017
- Practical Flutter: Improve Your Mobile Development with Google's Latest Open-Source SDK by Frank W. Zammetti, 2019

f) Termene obligatorii de consultații: săptămânal

g) Locul și durata practicii: Universitatea Sapientia,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

Primit tema la data de: 05.05.2020

Termen de predare: 06.07.2021

Semnătura Director Departament

Semnătura coordonatorului

**Semnătura responsabilului
programului de studiu**

Semnătura candidatului

Declarație

Subsemnatul PALFI SZabolcs, absolvent al specializării CALCULATOARE, promoția 2021 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Târgu Mureș

Data: 22.06.2021

Absolvent

Semnătura.....



Extras

În zilele noastre, majoritatea cumpărăturilor se fac online. Sunt foarte multe magazine online, care îți promit diferite promoții, indiferent de perioada anului. Aceste promoții de multe ori sunt doar de fațadă, în spatele lor de fapt nu este nici o scădere reală de preț. Aceste aspecte se pot observa doar în cazul în care urmărim prețul unui produs zilnic, ceea ce este foarte repetitiv și solicitant în raport cu timpul nostru. Toate aceste aspecte sunt valabile și în cazul în care dorim să cumpărăm produsele la prețuri corecte și cu adevărat aflate la promoție.

In această lucrare este prezentat un sistem pentru monitorizarea prețurilor care automatizează procesul descris mai devreme. Sistemul se trezește în momente predefinite pentru a mina date publice de pe Internet, care apoi vor fi salvate. Utilizatorul are la dispoziție o extensie de browser și o aplicație pentru telefon realizată în Flutter, prin care poate adăuga produse noi în listă și poate urmări pe un grafic, evoluția prețurilor. Așa zisa inimă a sistemului o reprezintă un script realizat în Python, care este responsabil pentru adunarea datelor, actualizarea periodică a prețurilor și a adăugării unor noi produse în lista utilizatorului. Datele adunate sunt stocate într-o bază de date furnizată de Firebase, care s-a dovedit a fi ideală pentru utilizarea noastră.

După luni de adunare a datelor, folosind sistemul implementat, am analizat datele, după care am făcut câteva observații interesante. Au fost mai multe modele după care se schimbau prețurile, însă nu a fost niciunul concluziv și valabil pentru toate produsele. Au fost mai multe cazuri unde am observat manipulări ale prețurilor, mai ales în perioada de Black Friday, dar pe de altă parte am observat și prețuri cu promoții reale și semnificative, așa că este destul de greu de generalizat în acest sens. După folosirea îndelungată a software-ului putem concluziona că acesta poate fi de folos unui potențial cumpărător, prin furnizarea unor informații utile și într-o formă ușor de citit, astfel economisind și timp.

Cuvinte cheie: webscraping, extensie de browser, aplicație Flutter

**SAPIENTIA ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK**

Price Monitor

DIPLOMADOLGOZAT

Témavezető:
Dr. Szántó Zoltán

Végzős hallgató:
Palfi Szabolcs

2021

Kivonat

Napjainkban az emberek nagy többsége a vásárlásaikat az online térben bonyolítják le. Rengeteg webáruház létezik, szinte állandóan vannak kedvezmenyék vagy ajánlatok. Viszont sok esetben a feltüntetett árak ingadoznak, vagy a kedvezmény csak látszólagos. Ezekre akkor lehet felfigyelni, ha napi szinten követjük az árak alakulását, viszont ez időigényes és repetitív folyamat. Ugyanez elmondható akkor is, ha egy terméket megszeretnénk vásárolni kedvező áron. Egy másik népszerű online áruházzal kapcsolatos jelenség a Black Friday napján történő drasztikus árcsökkentés, vagy legalábbis annak a látszata, hiszen sokszor hallottuk vagy gondoltuk, hogy ezek igazából hamisak vagy a csökkenés mértéke jóval kisebb, mint az valójában fel van tüntetve. Ezen gondolatok szolgálták a rendszer megvalósításához szükséges inspirációt.

A dolgozatban bemutatunk egy árakat követő rendszert, mely az előbb említett napi ellenőrzés folyamatát automatizálja. A rendszer meghatározott idő pillanatokban felébred, az online web áruházakon elérhető publikus adatokat bányássza, és az eredményeket elmenti. A rendszer biztosít egy böngésző kiegészítőt a termékek hozzáadására, követésére, illetve egy egyéb műveletekre. A mobil alkalmazás ugyanazt a funkcionálitást biztosítja, mint a böngésző kiegészítő, ez Flutter segítségével lett megvalósítva. A szoftver szívét úgymond egy Python script biztosítja, amely az adatok bányászását végzi a követni kívánt weboldalakról, valamint a periodikus frissítésért is felelős. Természetesen ezek az modszerek minden morális, illetve jogi feltételnek eleget tesznek a bányászás során. A bányászott adatok a Firebase által szolgáltatott Realtime Database segítségével vannak eltárolva, ami rendkívül jó és megbízható platformnak bizonyult a jelen felhasználásra.

A hónapokon keresztül gyűjtött adatokat kielemezve le tudtunk vonni pár következtetést éspedig azt, hogy valóban valamilyen szinten manipulálva vannak az árak, hiszen több érdekes mintára is felfigyeltünk, viszont ezek nem minden esetben a vásárló becsapását jelentik, volt példa reális és igencsak jelentős árcsökkenésre is, ezért elégé nehéz általánosítani. A rendszer hosszabb idejű használata során az a következtetés vonódott le, hogy igencsak hasznos lehet egy ilyen jellegű szoftver a vásárlók számára, hiszen valamilyen mértékben könnyebbé teheti a döntési folyamatot, az információk összesítése és ábrázolása által.

Kulcsszavak: webscraping, böngésző kiegészítő, Flutter alkalmazás

Abstract

These days the majority of shopping is done online. There are tons of online shopping websites, with some kind of big sale going on almost all the time, however these sales are most of the time made up. These false sales can be spotted only if you follow a products price daily, for a longer period of time which is a repetitive and time-consuming process. The same can be said when you're interested in purchasing some products but you're not quite decided yet, so you want to wait for the price to drop. These gave the inspiration for developing our system.

In this paper we'll present a price monitoring system which automates the process described earlier. At predefined moments the system wakes up and scrapes publicly available data from the Internet, which are then saved. The user has the opportunity to add products to his list, for them to be followed, and can view the price change on a chart. All this can be done from a Chrome browser extension or a mobile application built with Flutter. The so-called heart of the system is a python script which is responsible for the mining of the data, periodically updating the prices and pushing new products into the database. All this is done while respecting all moral and legal boundaries. All of our data is stored in a Realtime Database provided by Firebase, which turned out to be perfect for our application.

After months of gathering data, while using the system, we analyzed it and made a few conclusions. We saw multiple patterns emerging from the price changes but there is not one that matches every product. There were many scams regarding prices, especially during Black Friday but there were also real and significant price drops as well. After using the system for a longer period of time we can say that a software like this helps the buyers in making a purchase decision, by providing useful data in an easy to read form.

Keywords: webscraping, browser extension, Flutter app

Tartalomjegyzék

1. Bevezető	1
2. Célkitűzések	3
3. Szakirodalom áttekintése	4
3.1. Web bányászat	4
3.2. Web Struktúra bányászat	6
3.2.1. Személyre szabott információ lekérése példával illusztrálva	6
3.3. Web Scraping gyakori alkalmazásai	7
3.4. Web Scraping módszerek	8
3.5. Web Scraping szoftverek	9
3.6. Legális és Etikai keretek	10
3.6.1. Felhasználói feltételek	11
3.6.2. Szerzői jogok	11
3.6.3. GDPR	11
3.6.4. Weboldal károsítása	12
4. Rendszer specifikációi	14
4.1. Felhasználói Követelmények	15
4.2. Rendszer Követelmények	15
4.2.1. Funkcionális követelmények	15
4.2.2. Nem-Funkcionális követelmények	17

5. Rendszer architektúrája	19
5.1. A modulok megvalósítása	20
5.1.1. Chrome Extension	20
5.1.2. Telefonos alkalmazás	27
5.1.3. Backend	33
5.1.4. Web áruházak	38
5.1.5. Adatbázis	40
6. Esettanulmány	41
6.1. Black Friday	42
7. Összefoglalás	45
7.1. Megvalósítások	46
Irodalomjegyzék	46

Ábrák jegyzéke

3.1. A Web Bányászat rendszertana [1]	5
4.1. Use Case diagram	14
5.1. A rendszer architektúrája	19
5.2. Kiegészítő bejelentkezási/regisztrálási felülete	23
5.3. Főoldal, adminisztrációs rész	24
5.4. Termék árváltozása	25
5.5. Telefonos alkalmazás bejelentkezási/regisztrálási felülete	29
5.6. Főoldal, adminisztrációs rész	30
5.7. Termék hozzáadása	31
5.8. Termék árváltozása	32
5.9. Informáló üzenetek	33
5.10. Python script Update módban való indítása	34
5.11. Backend rész adminisztrációs felülete	36
5.12. Információ lokalizálása a weboldal strukturájában - Emag	38
5.13. Támogatott weboldalak különböző strukturája	39
5.14. Adatok tárolása az adatbázisba	40
6.1. Megfigyelt időszakos árnövekedések különböző termékek esetében	42
6.2. Példa az ár növekedésére már Black Friday előtt	43
6.3. Többszörös árcsökkentés Black Friday napján	44
6.4. Laptop mesterséges árleszállítása	44

1. fejezet

Bevezető

A mai rohanó világban a bevásárlások egyre növekvő százaléka történik Interneten, minden lehetőséget nyújtva a vásárlóknak, hogy egy bizonyos terméket több, akár hazai akár külföldi, oldalról is megvásárolhasson. Az e-commerce-el foglalkozó cégek rohamos fejlődésnek indultak az utóbbi évtizedben mely maga után vonja az érdekesebbnél érdekesebb marketing fogásokat, melyekkel a célközönséget próbálják vásárlásra bírni.

Valószínűleg mindenki hallott már a “Black Friday” az-az „Fekete Péntek” -nek nevezett jelenegről amely inspirációként szolgált az alkalmazás megvalósításához. Ez a kifejezés legelőször az 1800-as években fogalmazódott meg, amikor is Jay Gould és James Fisk az amerikai arany árak manipulálása által 20%-os esést okoztak a részvénypiacon melynek következtében az árucikkek értéke felére csökkent¹. A 20. század közepe fele ez már egészen más jelentéssel bírt, ugyanis a Hálaadás ünnepét követő napon, az-az pénteken vette kezdetét a karácsonyi árleszállítás, mely sok cég esetében életmentő volt, hiszen ekkor kerültek át a veszteséges állapotból melyet pirossal jelöltek, a jövedelmezőbe, amit már fekete írószerrel jegyeztek fel². Ebben az időszakban a megszokottnál jóval nagyobb és több árleszállítással vonzották az embereket.

Mint azt sokan tudjuk, országunkban is nagy népszerűségnek örvend ez a jelenség, habár elégé távol áll az eredeti koncepciótól. Nagyon sok mesterséges árleszállítással próbálják becsapni az em-

¹[https://en.wikipedia.org/wiki/Black_Friday_\(1869\)](https://en.wikipedia.org/wiki/Black_Friday_(1869))

²[https://en.wikipedia.org/wiki/Black_Friday_\(shopping\)](https://en.wikipedia.org/wiki/Black_Friday_(shopping))

bert, melyet legtöbb esetben jól kitervelt áringadozással oldanak meg³. Ugyanakkor, nem kizárálag ebben a periódusban lehet észrevenni az úgymond „hamis” kedvezményeket ezért szükségét láttuk egy olyan alkalmazás kifejlesztésének, amely nyomon tudja követni egy megadott termék árat, illetve annak ingadozását.

Mivel az Interneten publikus adatok találhatók, ezek felhasználásával semmiféle kár nem keletkezik az adott weboldalak számára. Egy internetes oldal betöltése során, mi, mint felhasználók, egy kérést intézünk egy szerver fele a böngészőnkön keresztül, ami majd a kapott válasz alapján felépít és megjeleníti számunkra a megtekinteni kívánt oldalt. Ezt a műveletet természetesen legtöbbször ahogyan előbbiekbén is említettem, böngészőn keresztül végezzük, viszont ez nem egy szükséglet, inkább egy eszköz, számos más módon is intézhetünk kéréseket egy adott szerver fele. Az általunk megvalósítani kívánt alkalmazás ezt a tulajdonságot hivatott kihasználni, ezáltal nyilvánosan elérhető adatok begyűjtésével, feldolgozásával és elemzésével szeretne foglalkozni.

³<https://cavaleria.ro/tepele-de-black-friday-2020/>

2. fejezet

Célkitűzések

Az alábbi fejezetben összefoglaljuk az alkalmazás fontosabb célkitűzéseit. A fő cél, mely érdekében létrejön a szoftver, az, hogy segítsen egy potenciális vásárlónak követni egy adott termék árának időbeli változását. Ennek megfelelően, a következő célok fogalmazódtak meg:

- Rövid használati útmutató, szöveges formában + támogatott oldalak listázása
- Regisztrálási, bejelentkezési lehetőség biztosítása, hogy különböző eszközökön is, mint például mobilos vagy webes, elérhetőek legyenek a követett termékek információi
- Felhasználói fiók jelszavának változtatási lehetősége, felhasználói fiókból való ki jelentkezés valamint annak törlése
- A termékek ábrázolása listaszerűen történjen
- Az árak változását a felhasználó számára grafikus formában szeretnénk ábrázolni, könnyen átlátható vonal diagram segítségével, a könnyebb átláthatóság érdekében
- Egyszerű átirányítás a termék oldalára
- Telefonos alkalmazás mely segítségével követni tudja az árakat, új termékeket tud hozzáadni
- Kiegészítő, Chrome alapú böngészőkre, mely segítségével követni tudja az árakat, új termékeket tud hozzáadni

3. fejezet

Szakirodalom áttekintése

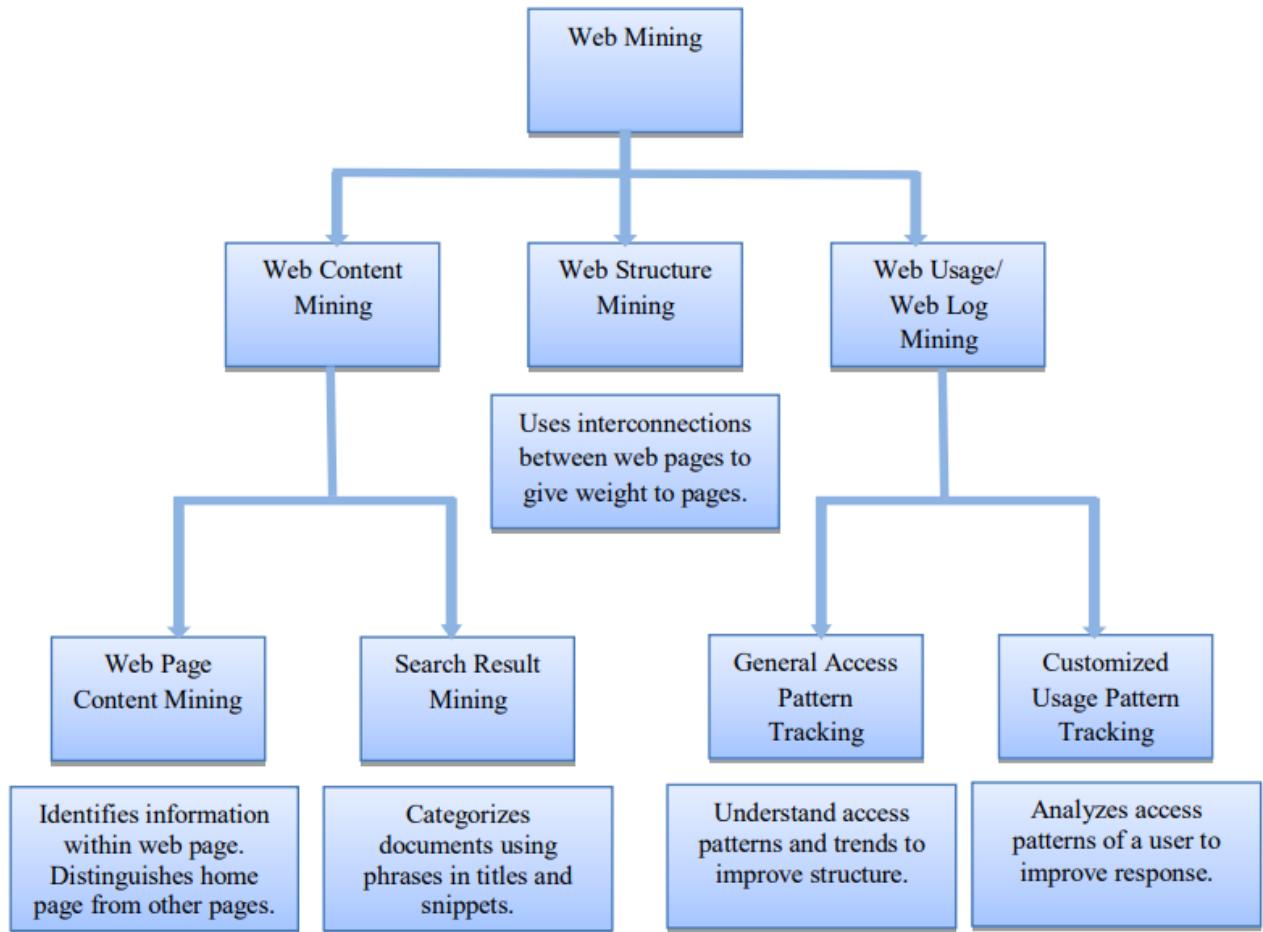
3.1. Web bányászat

Web bányászat, vagy angol kifejezésben Web mining, alatt azt a folyamatot értjük, amely által új, eddig nem ismert, de hasznos információt fedezünk fel az Interneten található adatok között. Mindezért a cégek arra használják, hogy új értékes információkat gyűjtsenek, ezeket feldolgozzák, majd ezek által a felhasználókat vagy fogyasztókat jobban megismерjék. Ez a folyamat az úgynyevezett adat bányászat technikát alkalmazza ahhoz, hogy automatikusan kinyerje az adatokat az Internetről [2].

Számos más technikát is alkalmaztak már új információk kinyerésére, az így is hatalmas és egyre növekvő adat mennyiségből, mint például az Information retrieval, Information extraction illetve gépi tanulás. Az Information retrieval működési elve az, hogy a szöveg indexelésé után nyeri ki a hasznos információt. Az Information Extraction arra fokuszál, hogy csak a lényeges információt nyerje ki, míg az előbb említett inkább hasznos dokumentumokat jelöl meg. A gépi tanulásos módszer nem kötődik direkt módon a Web Scraping-el viszont segítséget nyújt a szövegek osztályozási folyamatában. A web bányászatot három fő kategóriába soroljuk, ahogy az látható az 3.1 ábrán is.

A Web Content Mining vagy magyarul webes tartalom bányászat, olyan tartalmakra fekteti a hangsúlyt, mint például szöveg, kép, audió, videó, metaadatok, hiperlinkek. Ezek tanulmányozása, segít nekünk megérteni a felhasználók, vásárlók viselkedését mely által a weboldalak teljesítményét növelni lehet a későbbiekben, hogy azok jobban, illetve hatékonyabban működjenek.

Mivel a webes tartalom bányászat megvizsgálja úgy a kereséseket, mint azoknak a konkrét tartal-



3.1. ábra. A Web Bányászat rendszertana [1]

mát ezért ezt is tovább lehet osztani két kategóriába, Keresési Eredmény, illetve Weboldal Tartalom bányászat. Nevükből adódóan, ezek kiegészítik egymást, mivel a tartalom bányászat azon oldalakon történik, melyeket korábban megvizsgált és ígéretesnek talált a Keresési eredmény általi elemzés. A Web Structure Mining egy olyan ágazat, mely struktúrák bányászásával foglalkozik, mint például HTML vagy XML címkék, amely által weboldalak közötti kapcsolatot tud felismerni, ezáltal súlyokat rendelve azokhoz. A Web Usage Mining által lehet megérteni a különböző használati mintákat, amelyeket a felhasználók követnek, ezeket főként naplázás, felhasználók profiljai, sülik, könyvjelzők által, de ugyanide tartoznak a különböző egér mozdulatok vagy görgetési adatok is.

3.2. Web Struktúra bányászat

Rengeteg új adat generálódik az Interneten, napról napra az adat mennyisége exponenciálisán növekszik. Bőségesen állnak rendelkezésünkre szolgáltatások, illetve információk, elektronikus áruházak, elektronikus újságok, közösségi oldalak formájában, hogy csak párát említsünk. Habár ezen adatok fogyasztás céljára lettek szánva, sok időt el lehet tölteni az adatok kinyerésével és elemzésével. Továbbá, a weboldalak adatai HTML, illetve más webre szánt formátumban vannak jelen, amely megnehezíti az automata feldolgozást. Ez lett a mozgató rugója az ezen a téren zajló kutatásnak, mint például a Web Scraping.

A Web Structure Mining, vagy ismertebb nevén Web Scraping, az a folyamat, amely által hasznos információkat nyerünk ki egy weboldal HTML kódjából, amely az Internet fő formázási eszköze [3]. Az egyik metodológia megállapította, a rendezett annotációk, melyek nem mások, mint gépek számára is értelmezhető leíró információk az adott oldalra tekintve, egy külön szemantikus rétegben vannak tárolva, elválasztva a weboldaltól, ezáltal megkönnyítve és felgyorsítva a bányászási folyamatot, mivel először ezeket a fontos metaadatokat dolgozza fel, majd csak ezt követően magát a weboldalt.

A HTML struktúráját tekintve, két fő adat típust lehet bányászni belőle. Az egyik a felhasználó által generált - a másik pedig a metaadat. A felhasználó által generált adat bármi olyan típusú adatra vonatkozik, amelyet a felhasználó hozott létre vagy adott hozzá a weboldalhoz, akár személyesen akár egy közösségi platform hozzácsatolásával. A metaadat definíció szerint olyan adat, amely leír egy másik adatot [4], ezek általában minden weboldalon megtalálhatók, fontosabb leíró adatokat tartalmazva, mint például szerző, cím, cikkek esetén megjelenés időpontja, kulcsszavak. Ezek általában nem láthatóak a felhasználó számára, viszont kiolvashatóak az adott oldal HTML kódjából [4].

3.2.1. Személyre szabott információ lekérése példával illusztrálva

Tegyük fel, hogy egy személy fárasztónak találja, hogy a nap végen a fontos vagy számára értékes hírek után keressen, vagy előkeresse a kedvenc sport csapata elért eredményeit. Egy intelligens Web Scraper a tökéletes eszköz ebben az esetben, mivel az időközönként végig tudja böngészni az Internetet a felhasználó által megszabott témakörökben található információk után kutatva, vagy akár specifikus kulcsszavakat tartalmazó híreket előkerítve. Ez egy előre weboldalakat tartalmazó listán

menne végig, melyet a felhasználó határoz meg, hogy a számára hiteles információt kapja meg [3].

Egy másik példa egy Web Scraper felhasználására az, amikor például egy vásárló több terméket is kinézett magának, több különböző weboldalon. Ha esetleg nem szeretné azonnal megvásárolni a terméket, hanem csak követni annak árát, akkor esettől függően, több oldalra is be kell jelentkeznie, több felhasználóval oldalanként, de legjobb esetben is számos oldalt kellene naponta megfigyeljen és lejegyezzen. Egy Web Scraper abban könnyítené meg a felhasználó dolgát, hogy például egy böngészős kiegészítő keretein belül, a vásárló hozzá tudja adni egy listához a terméket és attól a pillanattól, a program naponta akár többször is le tudja kérni a termék árát, anélkül, hogy a felhasználó bármit is tenne. Ezáltal egy helyen lenne a vásárló több terméke, és pár kattintással tisztább képet alkothat a termékek árára vonatkozóan.

3.3. Web Scraping gyakori alkalmazásai

A web scrapinget, adatgyűjtő jellege miatt számos területen fel lehet használni, illetve igénynek megfelelően alakítani. Ezekre pár példa:

- Online ár összehasonlítás – ugyanazon termék árának összehasonlítása több weboldalon, pl. <https://www.compari.ro/>, <https://www.price.ro/>
- Contact Scraping – általában email címeket gyűjtenek, marketing céljából
- Időjárással kapcsolatos adatok gyűjtése
- Weboldal változásainak figyelése
- Több forrásból származó adat egyesítése
- Kedvezmény kuponok pl. pouch – chrome extension
- Álláshirdetések összesítése
- Brand monitoring – egy bizonyos márkhöz tartozó adatokat gyűjtik, általában az ahhoz társított véleményre kíváncsiak

- Piac tanulmány – egy adott termék piacon való elhelyezkedését, illetve potenciális sikerességet próbálják megjósolni a bányászott adatok átvizsgálásával.

3.4. Web Scraping módszerek

Számos technika áll rendelkezésünkre melyekkel az adatgyűjtést végezhetjük, ezeket mindig az adott helyzetnek megfelelően kell kiválasztani, főként a hatékonyságot tartva szem előt. Ebben a részben a Web Scraping egy pár technikája kerül röviden ismertetésre.

- Copy-paste - Időközönként valaki kézzel történő adatgyűjtést, valamint vizsgálatot végez. Adott helyzetekben ez a leghatékonyabb módszer, viszont nagyon hajlamos a hibákra, sok időt és fáradságot vesz igénybe az ember részéről, amíg a nagy adathalmazokat feldolgozza.
- Reguláris kifejezések - Ez egy egyszerű és erőteljes megközelítése az információ gyűjtésnek. A UNIX vagy más programozási nyelv által használt reguláris kifejezés illesztésen alapszik.
- HTML Parsing – Félig strukturált lekérdező nyelvek segítségével elemezni, illetve módosítani a weboldalak tartalmát.
- DOM Parsing – A böngészőkbe beépített kezelő programok segítségével, az erre a célra fejlesztett alkalmazások lekérhetik a kliens oldalon dinamikusan létrejött tartalmakat is, mellyel utána fel tudják építeni a DOM fát, ebben pedig könnyebben lehet specifikus adatok után keresni.
- Web Scraping Szoftver – Számos szoftver áll rendelkezésünkre, amelyeket személyre szabott keresésre lehet használni.
- Mesterséges Intelligencia – Több helyen is kísérleteznek gépi tanulásos adatbányászattal, melynek az a célja, hogy a gépek megtanulják úgy értelmezni a weboldalakat, ahogy azt az emberek tennék.

3.5. Web Scraping szoftverek

Több szoftver is rendelkezésünkre áll a piacon, úgy ingyenes mint fizetett formában is, mindegyiknek megvan az erőssége illetve, gyengesége, ezek közül egy párat mutat be a jelen fejezet.

- A Web Scraping szoftverek rendkívül fontos szerepet játszanak ezen a téren, mivel automatizálják és rendkívül felgyorsítják az adat-gyűjtő, valamint feldolgozó folyamatot. Számos ilyen szoftver található a piacon, a maga előnyeivel és hátrányaival. Ezeknek az ára a funkcionalitásuk függvényében, valamint a támogatás és frissítési időszakok függvényében változik.
- Visual Web Ripper¹ – Az egyik legfejlettebb web scraping szoftver, melyet a Sequentum csoport fejlesztett 2006-tól kezdődően. Weboldalakról gyűjtött információk bányászására használják, úgy egyszerű weboldalak, mint e-commerce oldalak esetében, mint például eBay, Amazon, magento, azonban titkosított tartalmak esetében is segítségünkre lehet. A bányászott adatokat kimenthetjük adatbázisba vagy CSV, illetve XML formátumban is. Előnye, hogy vizuális felülettel rendelkezik, ezért rendkívül egyszerűen ki lehet választani, hogy pontosan mit is szeretnénk. Egyszeri fizetéssel lehet megvásárolni a szoftvert, \$349.00 áron², viszont fontos szempont, hogy ezen alkalmazás elveszti a gyártó általi támogatottságát 2021-ig kezdődően, kivéteket képzéve azon esetek, ahol a vásárlóval karbantartási szerződés van érvényben, mely túlhaladja ezt az időpontot.
- Web Content Extractor³ – Nagyon jó automatizálási lehetőséget nyújt, rendkívül egyszerűen használható, pár kattintással meg lehet adni a kívánt mintát, ami szerint majd adatokat fog gyűjteni. A program rugalmas, abból a szempontból, hogy nem próbálja túlokoskodni a felhasználót, hanem egy előlnézetet ad az eredményről, majd a felhasználó maga végezheti el a szükséges módosításokat, amennyiben szükség van erre. Ezt a szoftvert, bérlet alapú előfizetéssel lehet megszerezni, több változat is elérhető, az árak \$30 - \$150 / hónap² között mozognak.
- Mozenda⁴ – Az egyik legegyszerűbben használható szoftver ezen a téren, ami lehetővé teszi

¹<http://visualwebripper.com/>

²Az árak aktualitásáért lásd a szolgáltató weboldalát

³<https://www.webcontentextractor.com/>

⁴<https://www.mozenda.com/>

a kevésbé technikailag hozzáértő személyeknek is, az egyszerű bányászásokat. Fő különbsége más alkalmazásokhoz kepést, hogy maga az adatbányászati folyamat a felhőben történik és nem a felhasználó erőforrásait felhasználva, amely hatalmas előnyt jelenthet. Elérhető egy 30 napos próba csomag is, ami után \$250/hónap-tól² kezdődően változnak az árak, a választott csomag függvényében.

- Screen-Scraper⁵ – Nagyon fejlett web scraping alkalmazás, amelyet több változatban is el lehet érni. Az alapszintű verzió ingyenes, ezzel egyszerűbb adatok után lehet bányászni, könnyen kezelhető, nem kell sok technikai tudás hozzá. Más változatok, mint például a profi vagy vállalkozás szintű verziók már sokkal komplexebbek, több lehetőséget nyújtanak. Nagy előnyé, hogy más rendszerekkel könnyen összeférhető pl. Java, ezért fel lehet használni más, nagyobb szintű programokban is.

Természetesen sok más program is rendelkezésünkre áll, melyek hasonló funkcionálisokkal rendelkeznek, minden esetben az adott alkalmazásnak megfelelőt és legjobban illőt érdemes választani a nagyobb hatékonyság érdekében. Egyéb web scraper szoftverek [5]: WebHarvy, Easy Web Extract, WebSunDew, FMiner, Scrapy, import io.

3.6. Legális és Etikai keretek

Ebben a fejezetben a legális valamit etikai kérdésekről lesz szó, amely elég megosztó, illetve nem teljesen egyértelmű terület. Egyenesen a Web Scraping-et nem szabályozza semmilyen törvény, de több területen is problémába lehet ütközni, ilyen például a védett tartalom, az úgynevezett szerződésszegés vagy GDPR. Attól függően, hogy a bányászat milyen országhoz tartozó területen zajlik, figyelembe kell venni az ottani törvénykezést is, ezért is nehéz konkrétan jellemzni legalitási szempontból. A legfontosabb jellemzés talán az lenne, hogy van, ami igen és van, ami nem.

⁵<https://www.screen-scraping.com/>

3.6.1. Felhasználói feltételek

Ugyanúgy, mint egy szoftver vagy szolgáltatás esetén, amikor egy weboldalt használunk el kell fogadnunk bizonyos felhasználói feltételeket, melyek leggyakrabban kis felugró ablakkent jelennek meg, amikor először látogatunk a weboldalra, vagy ezeket a regisztrálás folyamán tudatosítják velünk. Amennyiben valamilyen módon megszegjük ezeket a feltételeket, érvénybe lép a fentebb említett szerződésszegés. Mivel ez csak akkor érvényes, ha a felhasználó explicit módon elfogadja a feltételeket, ezért jogi szempontból nehéz kizárnai a Web Scrapinget⁶.

3.6.2. Szerzői jogok

Bányászni vagy újra publikálni olyan adatokat vagy információkat, amelyeknek a szerzője explicit módon fenntartja a szerzői jogokat, legális szempontból szerzői jogosításnak minősül. Ugyanakkor egy weboldal nem minden esetben rendelkezik a felhasználói által generált adatokkal, vegyük például egy film értékelő oldalt, ahol a felhasználók kifejtik a véleményüket [6]. Más ebbe a kategóriába tartozó tartalmak például a videók, képek, zenék, adatbázisok, cikkek. Maga a bányászása ezeknek az adatoknak nem teszi illegálissá, a felhasználási módjuk határozza meg azt, hogy milyen kategóriába soroljuk azt. Az hogy milyen adatokra hogyan vonatkoznak ezek a szabályok országonként változik, van ami egyes országokban megengedett másokban viszont nem.

3.6.3. GDPR

Az Európai Unió által érvénybe léptetett Általános Adatvédelmi Rendelet⁷ alapján nem tiltott az adatbányászat, kivételt kepezve, ha ez a tevékenység nem tartalmaz személyes adatokat. Ilyennek minősül a név, lakcím, email cím, telefonszám, bankkártya adatok, banki adatok, IP cím, születési dátum, foglalkozási információk, orvosi adatok, személyes fotók vagy videók.

Ugyanakkor, mint minden más esetben itt is figyelembe kell venni az országok törvényei és szabályozásait ilyen értelemben. Mint sok más országban, Romániában is csak homályos úgy a törvénykezés mint az alkalmazási metodológia is, például nem világos hogy egy román ip címről mit

⁶<https://www.todaysoftmag.ro/article/3246/consideratii-legale-si-etice-in-contextul-scalarii-procesului-de-web-scraping>

⁷<https://gdpr-info.eu/>

bányászhatunk egy kanadai oldalról és így tovább.

3.6.4. Weboldal károsítása

Ha bármilyen tevekénység által, amelyet a Web Scraper végez, túlterheljük az adott weboldal szervereit vagy bármilyen módon sértjük, gátoljuk annak működését bűncselekménynek számít és legális következményei lehetnek. Ehhez azonban a kárnak anyaginak kell lennie, valamint könnyen bizonyíthatónak bíróság előtt, ahhoz, hogy bármiféle kártérítési kérelemre legyen jogosult a weboldal [6].

A fentieket figyelembe véve tehát, nem lehet egyértelmű választ adni arra, hogy a Web Scraping legális-e vagy sem. Helyette a válasz az, hogy helyzettől függ. Maga a Web Scraping nem illegális, viszont akár az is lehet a következő három dolog függvényében [7].

- Hogyan lett bányászva az adat?
- A bányászott adat típusa
- Hogyan lesz felhasználva a bányászott adat?

Ahhoz, hogy eldöntsük az esetünk legalitását meg kell vizsgálni, hogy az adat, amit szeretnénk bányászni publikusan elérhető-e vagy sem. Ha az adat eléréséhez nem szükséges bejelentkezni egy adott oldalra akkor a felhasználói feltételek nem érvényesek ezért legálisan lehet bányászni, mivel ez az adat publikusnak számít. Ha bejelentkezésre van szükség a bányászni kívánt adat eléréséhez, akkor a felhasználói feltételek tanulmányozásával el kell dönteni, hogy legális-e vagy sem, mivel azok elfogadásával legálisan alkalmazhatóvá vált számunkra, azaz a megszegésük jogi következményeket vonhat maga után.

A bányászott adat típusa szerint két formájára kell nagyon odafigyelni, Személyes Adatok és Szerzői Jogokkal rendelkező adatok, ezek fentebb részletesebben is tárgyalva voltak.

Az adatok felhasználása során figyelembe kell venni, hogy az illegális módon vagy csalás által történő adatgyűjtést minden állam bünteti, tehát ez bűncselekménynek minősül. Amennyiben olyan tartalmakat bányászunk melyek bizalmas vagy bármi féle módon védett információt tartalmaznak, ezeknek felhasználása ugyancsak törvényszegést jelent és az erényben levő büntető eljárások érvé-

nyesek. Természetesen, legtöbb esetben már maga az törvénysértő lehet, hogy ezekhez az adatokhoz jogosultság nélkül fértünk hozzá, ezért egyértelműen nem felhasználhatóak ezek az adatok.

Leegyszerűsítve, három alapvető kérdésre kell választ adni, ahhoz, hogy eldöntsük legális-e az adat bányászat, amelyet végre szeretnénk hajtani:

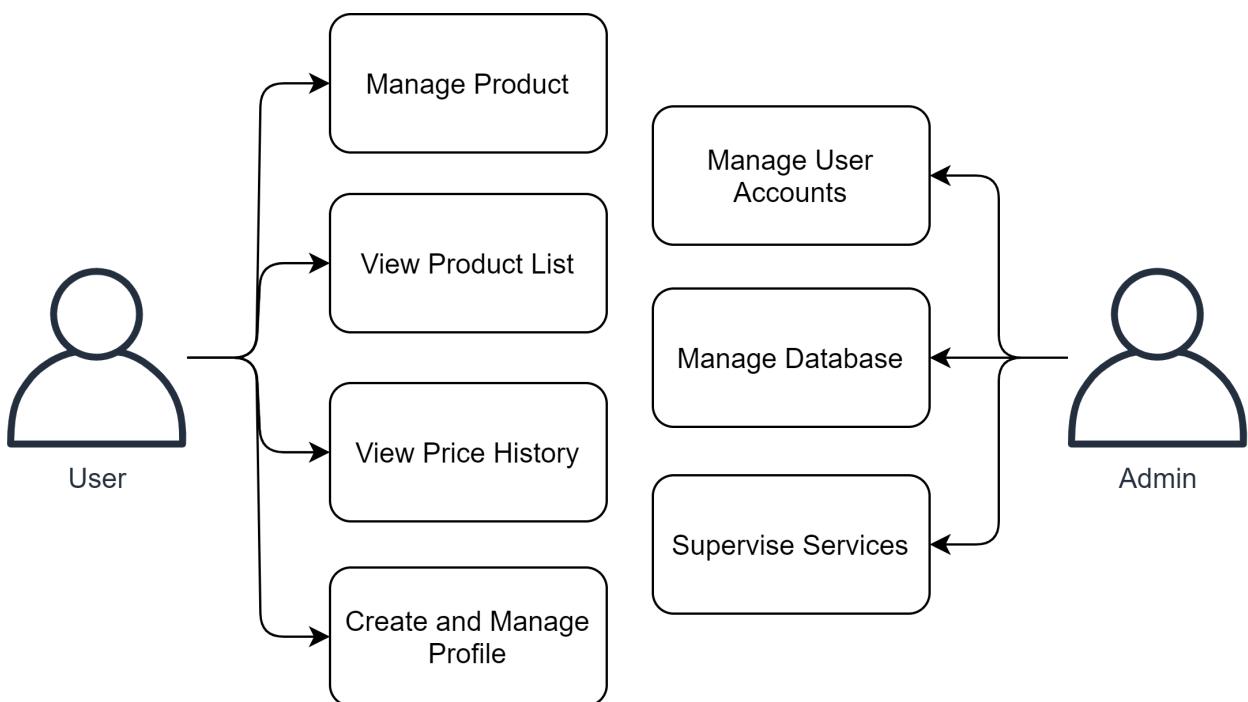
- Bejelentkezés által elérhető adatot bányászok?
- Személyes adatot bányászok?
- Szerzői jogokkal rendelkező adatot bányászok?

Amennyiben mindenhangot kérdésre nemleges a válasz, legálisnak bizonyul adott esetben a Web Scraping. Azonban, ha bármelyik kérdésre pozitív választ adunk nagy valószínűsséggel újra kell gondolni és figyelmesebben megvizsgálni az eset legális mivoltát. Ugyanakkor egyik esetben sem szabad megfeledkezni arról, hogy minden ország másképp kezelheti ezen kérdéseket, ezért ilyen értelemben is vizsgálódnunk kell.

4. fejezet

Rendszer specifikációi

A jelen dolgozatban egy árakat figyelő alkalmazás készítéséről van szó, mellyel különböző termékek árát lehet követni, bizonyos webshopokon. Az alkalmazás több platformon is elérhető, hogy minél könnyebben eljusson a kívánt információ a felhasználóig. A következőkben tárgyalva lesznek a felhasználói, valamint rendszer követelmények is.



4.1. ábra. Use Case diagram

4.1. Felhasználói Követelmények

Amint a 4.1 ábra mutatja, a rendszer használata során két alapvető szerepet lehet elkülöníteni, az egyik a tulajdonképpen felhasználó, aki használja a rendszert, igénybe veszi a szolgáltatást, a másik pedig egy adminisztrátor szerepkört betöltő személy. A következőkben e két szerepet betöltő személy követelményei lesznek bemutatva.

Felhasználó:

- Menedzselni tudja a profilját, vagyis regisztrálni, bejelentkezni tud, illetve, lehetősége van a jelszavának módosítására vagy adott esetben a profil törlésére
- Megtekintheti a termékeket tartalmazó listáját
- Kezelheti a termékeket tartalmazó listáját, vagyis új termékeket adhat hozzá, esetleg törölhet belőle
- Részletes reprezentálást kaphat a követett termékek árának változásáról, a követes pillanatától kezdődően
- Weboldal melyen követni szeretné a termékeket: Emag

Adminisztrátor:

- Kezeli a felhasználók fiókjait, az azokkal közbejövő problémákat
- Kezeli az adatbázist
- Felügyeli a rendszerek helyes működését, karbantartja a rendszert

4.2. Rendszer Követelmények

4.2.1. Funkcionális követelmények

A rendszernek mindenek előtt, egy bejelentkezési, illetve, regisztráció felülettel kell rendelkeznie. Regisztrálás után, a felhasználónak egy ellenőrző email-t kell kapnia, amivel igazolja, hogy ő a cím

tulajdonosa. A bejelentkezés nem lehetséges, abban az esetben, ha a felhasználó nem igazolta vissza az előbb említett email-ben a címét. A cím igazolása egy linkre való kattintással történik.

A felhasználónak lehetősége van a jelszavának módosítására melyet a bejelentkezési felületről ér el. Miután a felhasználó beírta az email címét, egy levelet fog kapni az adott címre, amelyen keresztül új jelszót tud beállítani.

Bejelentkezést követően, a felhasználó egy felületet lát, melyen bizonyos műveleteket végezhet. Megtekintheti a profiljához tartozó email címét, valamint törölheti a felhasználóját. Ugyanakkor, lehetősége van kijelentkezni az alkalmazásából melynek hatására újra a bejelentkezési oldalra kerül.

Ugynacsak a főoldalról a felhasználónak lehetősége van az alkalmazás használatával kapcsolatos információk megtekintésére mely tartalmaz egy listát is. A lista bizonyos weboldalakat tartalmaz, melyeket kiválasztva, az alkalmazás átirányít az adott elem oldalára.

A felhasználónak lehetősége van termékeket hozzáadni és kitörölni a listájából, valamint görgetni a lista tartalmában. Amennyiben frissíteni szeretné a lista tartalmát, ez úgy lehetséges, hogy a lista tetején tartózkodva, annak tartalmát megpróbálja görgetni (swipe down). A terméklistában egy elemet kiválasztva, részletes reprezentációt kap az adott elem tárolt adatairól, mint például aktuális ár, annak időbeli változása, hozzáadás időpontja, termék megnevezése.

Egy listaelem tartalmazza a termék megnevezését, aktuális árát, illetve egy képet róla. A termék ára mellett egy nyíl található, mely azt jelzi a felhasználó számára, hogy az aktuális ár hogyan változott a korábbi ellenőrzéshez képest. A termék árának csökkenését, egy lefele irányuló, a növekedését egy felfele irányuló, amennyiben pedig az nem változott, egy vízszintesen irányított nyíl jelzi. Ezeket a változásokat színekkel is kell jelezni, mely az árra és az előbb említett nyílra hat ki. Csökkenés esetén zöld, növekvés esetén piros, valamint, ha változatlan, akkor fehér vagy fekete színnel kell jelölni ezeket.

Amennyiben egy új termék került hozzáadásra vagy a termék törlése következett be, a rendszernek ezt fel kell ismernie és elvégeznie a szükséges lépéseket annak érdekében, hogy a felhasználó számára minden a legfrissebb adatok legyenek láthatóak. Amikor egy új terméket adott hozzá a felhasználó, a rendszer azonnal lekéri az adott webcím kódját, amiből kiveszi a szükséges adatokat, megnevezés, fénykép, aktuális ár, majd ezeket feltolti az adatbázisba az adott felhasználóhoz csatolva.

A rendszer back-end részének, mely a termékek hozzáadásáért és az adatbázis periodikus frissíté-

séért felelős, autonóm módon kell működni, minimális beavatkozással. Amikor egy felhasználó hozzá szeretne adni egy terméket, a rendszer azonnal reagáljon erre a kérésre.

4.2.2. Nem-Funkcionális követelmények

A rendszer backend része, mely a felhasználók által hozzáadott teremkek árainak ellenőrzését, frissítését, hozzáadását végzi, megszakítás nélkül kell működnie, napszaktól függetlenül. A rendszerre fel kell legyen telepítve a Python 3.8.0 vagy ennél frissebb verzió. Létfontosságú, hogy egy stabil Internet kapcsolattal rendelkezzen, vagyis a kapcsolatban ne legyenek megszakítások, valamint a fel-letöltési sebesség ne csökkenjen 10 Mb/s alá, annak érdekében, hogy megfelelő sebességgel lehessen az adatfeldolgozást, valamint az adatok adatbázisba való fel-, letöltését elvégezni. Amennyiben a kapcsolat megszakad, a rendszer azonnal próbáljon újrakapcsolódni, mindaddig amíg ez a művelet nem sikeres.

A rendszer vizuális felülete két alapvető platformon kell elérhető legyen. Az egyik egy böngésző kiegészítő, mely bármilyen Chromium alapú böngészőre telepíthető, asztali, valamint hordozható számítógépek esetében is, legalább 58-as verziójú Chrome-ot tamogatva. Természetesen ebben az esetben is elengedhetetlen az Internethez való csatlakozás.

A másik egy telefonos alkalmazás, mely Android operációs rendszerrel felszerelt készülékeken legyen elérhető. Az alkalmazásnak legalább 7.0 verziójú Androiddal felszerelt telefonokon kell működnie, mely rendelkezik stabil Internetkapcsolattal. Rendelkeznie kell Light illetve Dark móddal is, melyek közötti váltást automatikusan végzi, az operációs rendszer beállításaihoz igazodva. A felhasználó számára, a követett termékeit egy görgethető listában kell megjeleníteni.

Úgy a telefonos alkalmazás, mint a böngésző kiegészítő esetében szükség van Internet használati jogosultságra. Továbbá, a kiegészítő esetében még szükségesek a következő jogosultságok: "storage", "unlimitedStorage", "tabs", "activeTab", "<all_urls>".

A rendszer a Firebase által biztosított Realtime Database nevű adatbázist kell használnia az adatok tárolására. A felhasználó bejelentkezését és a fiókjához tartozó műveleteket szintén a Firebase által biztosított Authentication szolgáltatás végezze, mivel ez biztonságos módon tárolja a szükséges információkat, ugyanakkor, a jelszavakat titkosítva kezeli. Továbbá, ezen keresztül lehessen új jelszót beállítani, a felhasználót törlni, ugyanakkor a regisztrálás után szükséges email visszaigazolása is

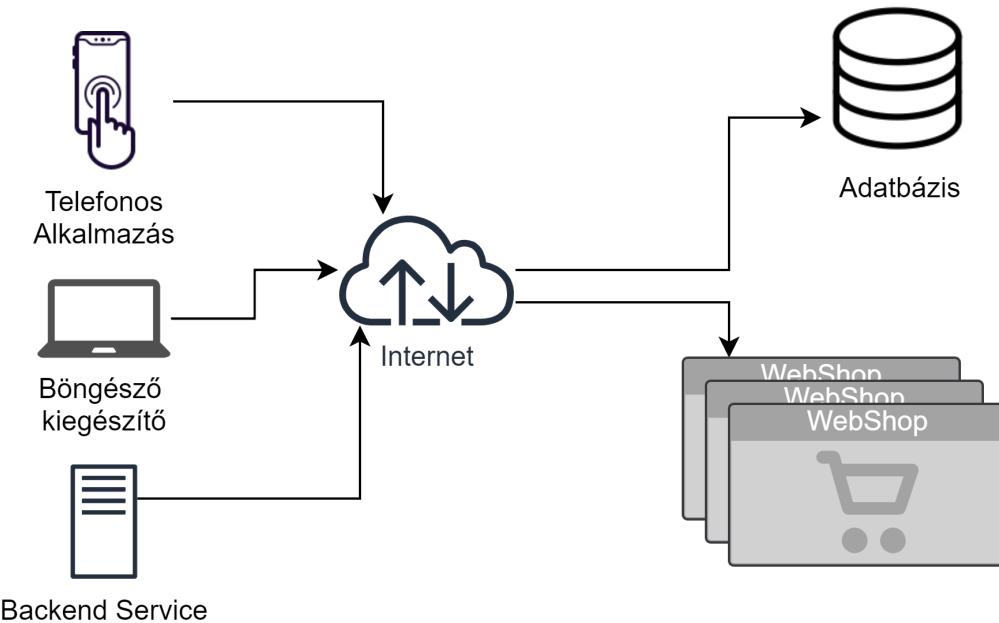
ezen a szolgáltatáson keresztül történjen, mivel ezekre egyszerűen használható, ugyanakkor hatékony és biztonságos megoldást nyújt.

Az alkalmazás könnyen használható kell legyen, a felhasználónak minden funkciót három kattintáson belül el kell érnie. Amikor a felhasználó egy új terméket szeretne hozzáadni a listájához, ne keljen több mint 5 másodpercet várnia ahhoz, hogy az új termék megjelenjen a listájában.

A rendszernek meg kell felelnie az érvényben lévő Európai, valamint Romániai törvényeknek melyek az adatok kinyerésére érvényesek. Továbbá, eleget kell tegyenek a bányászott weboldalak előírásainak is, melyek az adatok felhasználására és kinyerésére vonatkoznak.

5. fejezet

Rendszer architektúrája



5.1. ábra. A rendszer architektúrája

A rendszer alapvető részét képezi az adatbázis, amely az adatok tárolását, illetve az azokat elérő API-t szolgáltatja. Ehhez az adatbázishoz két alapvető típusú készülék csatlakozik. A felhasználó oldali, amely lehet akár böngésző kiegészítő vagy telefonos alkalmazás, illetve a szerver vagy logika oldali, amely az adatok feldolgozását biztosító logikát és erőforrást tartalmazza, utóbbi a 5.1 ábrán Backend Service-ként van jelölve.

A böngésző kiegészítő szükséges, mivel sokkal gyorsabban el lehet érni a megtekinteni kívánt adatokat, valamint sokkal egyszerűbbé teszi a termékek hozzáadását azzal, hogy a követni kívánt termék oldalát egyáltalán nem kell elhagyni. Továbbá fontos, hogy az alkalmazás tudja azt, hogy a felhasználó épp milyen oldalon tartózkodik, ahoz, hogy a megfelelő URL-t kapja meg, mindezt a kiegészítő könnyen és megbízhatóan el tudja végezni.

Mivel azonban nem minden vagyunk laptop vagy asztali gép közelben, ezért lehetőséget nyújtunk arra, hogy telefonos applikáció segítségével is el lehessen érni a követett termékeket, valamint minden ehhez tartozó műveletet, akárcsak az előbb említett kiegészítő esetén. Egy termék hozzáadása ezúttal az operációs rendszer megosztási menüjén keresztül történik, ami által az alkalmazás megkapja a követni kívánt termék elérhetőségét.

Ahhoz, hogy az eszközök kommunikálni tudjanak egymással, egyértelmű, hogy szükség van valamiféle összeköttetésre, amely a mi esetünkben az Internet lesz. Ez a funkció létfontosságú, mivel minden adatot fel, illetve le kell tölteni az adatbázisból, függetlenül az eszközök tartózkodási helyétől, ugyanakkor a szolgáltatást biztosító rendszer is ezen keresztül éri el a termékek weboldalát.

Az architektúra fontos részét képezik a webshop-ok is, mivel ezeket úgy a termék hozzáadásakor, mint azoknak periodikus ellenőrzésekor el kell érni, az adatok begyűjtése érdekében. A támogatott webshop-ok főként népszerűségüket tekintve lettek kiválasztva. Mivel minden weboldal másképp épül fel, mindegyik oldal struktúrájából másképp kell kinyerni az adatokat, ezért ezt a folyamatot személyre szabottan kell végezni. Ugyanakkor változhatnak is idővel ezek a struktúrák, ezeket folyamatosan figyelni kell.

5.1. A modulok megvalósítása

5.1.1. Chrome Extension

A böngésző kiegészítő vagy angolul browser extension, egy a böngésző környezetén belül futó alkalmazás, ami olyan funkciókat hivatott hozzáadni a felhasználói felülethez, melyek megkönnyítik vagy jobbá teszik a felhasználói élményt. Előnye, hogy alkalmazások ezrei állnak a felhasználok rendelkezésére, melyeket pár kattintással telepíthetnek is. Ezek már szinte minden böngészőn megtalálhatók

valamilyen formában, a legnépszerűbbek esetében, mint például, Google Chrome, Firefox ez a funkció régóta jelen van.

Ezen kiegészítők működése valószínűleg sokak számára ismert, amolyan lenyíló ablakként jelennek meg a böngészőben, anélkül, hogy hatással lennének az éppen megjelenített tartalomra. Ennek tudatában, ez a megközelítés tűnt a legmegfelelőbbnek a dolgozatban tárgyalt szoftver felhasználói felületének elkészítése során. Mivel ezek a funkciók csak asztali gépeken érhetőek el, ezért szükséges volt egy telefonos interface kifejlesztése is, mely a későbbiekben kerül bemutatásra.

A kiegészítő megvalósítása során, JavaScript, HTML, CSS programozási nyelvek voltak felhasználva. Mivel korábban nem volt tapasztalatom böngészőhöz való kiegészítők fejlesztésében, ezért az implementálási folyamat információ gyűjtéssel kezdődött.

Első lépésben szükséges egy manifest.json file létrehozása, mely a kiegészítő alapvető információt tartalmazza, mint például verziószám, név, rövid leírás, felhasznált függőségek, engedélyezett műveletek stb... . Ezek után, a fejlesztés hasonló egy hagyományos weboldal elkészítéséhez.

Listing 5.1. manifest.json file

```
1 {
2     "manifest_version": 2,
3     "name": "Price Monitor",
4     "description": "Price monitoring interface",
5     "version": "0.9",
6     "icons": {"128": "icons/icon128.png"},
7     "browser_action" : {
8         "default_icon": "icons/icon19.png",
9         "default_popup": "login.html"
10    },
11    "content_scripts": [
12        "matches": ["<all_urls>"],
13        "js": ["content.js"]
14    ],
15    "background": {
16        "page": "background.html"
17    },
18    "permissions": ["storage", "unlimitedStorage", "tabs", "activeTab", "<all_urls
```

```

18      >"] ,
19      "content_security_policy": "script-src 'self' https://cdn.amcharts.com/lib/4/
20      core.js https://cdn.amcharts.com/lib/4/charts.js https://cdn.amcharts.com/lib/
21      /4/themes/dataviz.js https://cdn.amcharts.com/lib/4/themes/animated.js https/
22      ://www.gstatic.com/ https://*.firebaseio.com https://apis.google.com https://
23      www.googleapis.com https://securetoken.googleapis.com; object-src 'self' ;
24      connect-src 'self' https://securetoken.googleapis.com https://apis.google.com
25      https://www.googleapis.com wss://*.firebaseio.com"
26  }

```

A fejlesztés során több könyvtár került felhasználásra, különböző funkciók ellátására, ezek a Bootstrap¹, sweetalert2², firebase³, amcharts⁴. A bootstrap egy ingyenes, nyílt forráskódú CSS framework, mely segítségével interaktívabbá, szebbé tehetjük a weboldalunkat, előre definiált mintákat biztosít nekünk, gombok, navigáció vagy más komponensek esetében. A sweetalert2 egy úgynevezett riasztásokért felelős könyvtár, olyan esetekben került használatra, amikor egy felugró ablak segítségével szeretnénk megerősítést kérni a felhasználótól egy bizonyos művelet elvégzésére, mint például termékek törlése vagy kijelentkezés során, viszont egyéb, információ közlési célokra is felhasználva lett, mint például egy művelet sikeres elvégzésének visszajelzése. A firebase könyvtárakat bejelentkezési, valamint adatbázis kezelő funkciók miatt volt szükséges használni. Az árak időbeli változását ábrázoló diagramok esetében több könyvtárat is kipróbáltam (Chart, Highcharts, ApexCharts), viszont a végső választás az amCharts nevűre esett, mivel ez volt a leg megfelelőbb az adott esetben, főleg az úgynevezett „panning” funkció miatt, mely lehetővé teszi hogy görgessünk a diagramon, illetve az idő skálát is változtatni tudjuk, mindez valós időben. Ezen funkciók segítségével pontosabban és hatékonyabban tudjuk követni az árak változását, ugyanakkor vizuális szempontból is kellemes élményt nyújt.

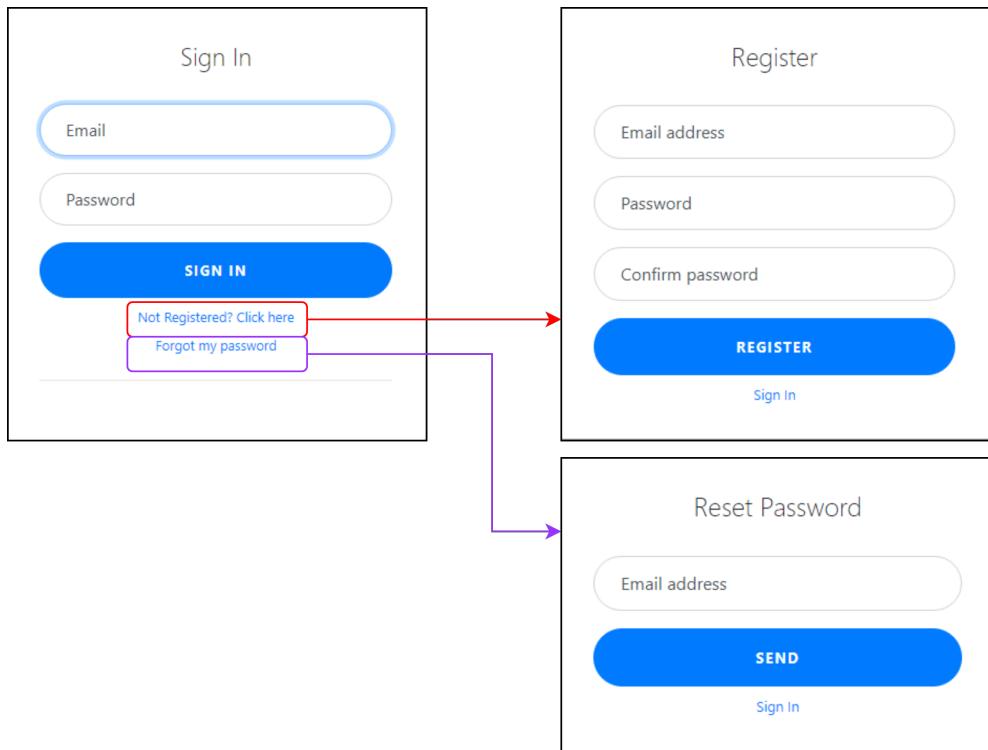
Amikor a felhasználó először használja a kiegészítőt, egy login oldal jelenik meg számára, melyen be tud jelentkezni, valamint regisztrálni tud (5.2 ábra).

¹<https://getbootstrap.com/>

²<https://sweetalert2.github.io/>

³<https://firebase.google.com/>

⁴<https://www.amcharts.com/>



5.2. ábra. Kiegészítő bejelentkezési/regisztrálási felülete

Bejelentkezés után a felhasználó a főoldalon találja magát, ahol megtekintheti a követett termékeit, ugyanakkor az alkalmazás használatával kapcsolatos információkat is megtalálja, a fejlécben található „i” szimbólummal jelölt gombra kattintva, valamint ugyanitt vannak felsorolva a támogatott oldalak, melyekre kattintva, az adott weboldalra navigálhatunk. A jelenleg támogatott webshopok az Emag⁵, Flanco⁶ és QuickMobile⁷. Továbbá, szintén a fejlécben található a felhasználó fiókjához tartozó információkat elérő gomb, melynek hatására egy felrúgó ablakban tekintheti meg az email címet, amivel bejelentkezett, illetve itt tud adminisztrációs műveleteket is végezni. Az előbb tárgyaltokat 5.3 ábrán láthatjuk.

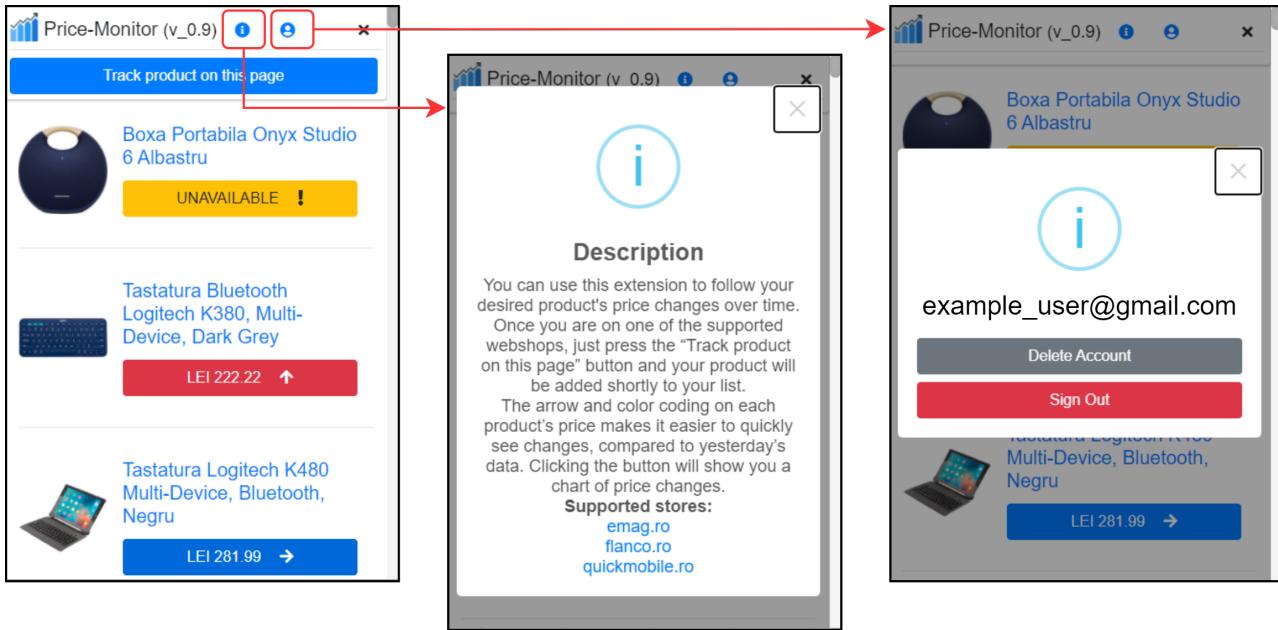
Új termék hozzáadásakor, az ablak tetején található „Track product on this page” gombra kattintva (5.3 ábra) lehetjük ezt meg, melyek után pillanatokon belül látható majd a listában az adott termék. Amennyiben nem az alkalmazás által nem támogatott oldalon tartózkodunk, ez az opció nem elérhető,

⁵<https://www.emag.ro/>

⁶<https://www.flanco.ro/>

⁷<https://www.quickmobile.ro/>

a gomb egyszerűen nem jelenik meg. Amikor követni szeretnénk egy terméket, az alkalmazás beszúrja a termék URL-jét az adatbázisba, ami után majd az szoftver backend része végzi majd az információk lekérését, erről később részletesen is szó lesz.

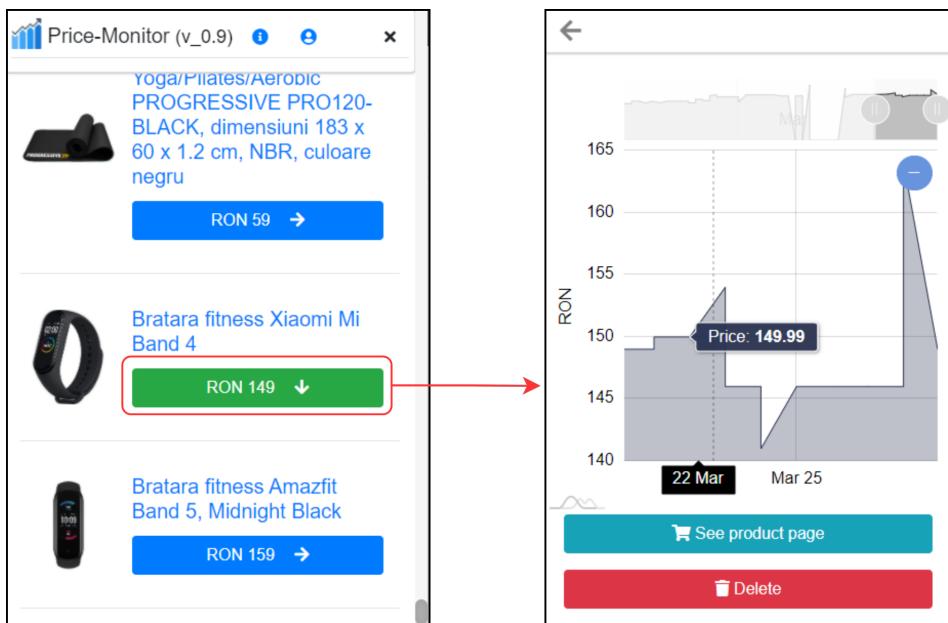


5.3. ábra. Főoldal, adminisztrációs rész

Amint a 5.3 ábrán is láthatjuk, a termékek gombjai tartalmazzák a termék aktuális árát, tőlük jobbra egy nyilat, valamint ezek különböző színűek. A nyilak, illetve, színkódolás azért van, hogy a felhasználónak visszajelzést adjuk arról, hogy a termék drágult, olcsóbb lett, nem változott az ára, esetleg jelenleg nem elérhető. Amennyiben a termék drágul, ezt egy felfele mutató nyíl és piros szín jelzi, csökkenés esetén a nyíl lefelé mutat és a gomb színe zöldre vált, az utóbbi az 5.4 ábrán látható. Amikor egy termék ára nem változik, ezt egy vízszintesen irányuló nyíl és a kék szín jelzi, valamint, ha a termék nem elérhető, akkor ezt sárga szín és az „UNAVAILABLE” szöveg mutatja.

Ha egy termék gombjára kattintunk, akkor megnézhetjük az adott termék árának változását egy diagramon, a követés pillanatától az aktuális dátumig. Amint a 5.4 ábra mutatja, a diagramon szépen látható az arák változása, ami sok esetben naponta akár többször is változhat. Mivel az adatmennyiség idővel nagyon nagy lehet, ezért a diagramot mozgatni lehet, hogy egy adott intervallumot vizsgáljunk, vagy az időskálát növelhetjük, illetve csökkenthetjük, igénynek megfelelően, a diagram tetején talál-

ható csúszka segítségével. Láthatjuk továbbá azt is, hogy ha az egeret a diagram egy adott pontján tartjuk, akkor az levetíti nekünk az adott dátumon a termék árát. Az ablak alján található a törlés gomb, ezzel törölhetjük a terméket a listánkból, ha esetleg már nem vagyunk érdekeltek az adott árucikk követésében. Az előbb említett törlés gomb fölött helyezkedik el még egy gomb, mely megnyomásával az adott termék weboldala nyílik meg számunkra a böngésző egy új ablakában.



5.4. ábra. Termék árváltozása

A diagram megvalósítása az amcharts nevű könyvtárral történt, mely a mi felhasználásunkra tökéletesnek minősült, minden funkciót hozva, amely megkönnyíti az adatok megtekintését, elemzését. Első lépésben az adathalmaz került létrehozásra és formázásra, annak megfelelően, ahogy az adott diagram elvárja azt (Listing 5.2). Következő lépésben maga a diagram lett létrehozva a különböző beállításokkal együtt, mint például időskála formázása, tengelyek elnevezése, a diagram kezdeti állapota, ezt a 5.3 kód részlet valósítja meg.

Listing 5.2. Adathalmaz létrehozása

```

1 var data = [];
2 for(let id of Object.keys(checks)){
3   data.push({date: new Date(cheks[id].date), price: checks[id].price})
4 }

```

Listing 5.3. Diagram létrehozása, beállításai

```
1 am4core.ready(function() {
2     am4core.useTheme(am4themes_dataviz);
3     am4core.useTheme(am4themes_animated);
4
5     var chart = am4core.create("chartContainer", am4charts.XYChart);
6     chart.data = data;
7
8     var dateAxis = chart.xAxes.push(new am4charts.DateAxis());
9     dateAxis.baseInterval = {
10         "timeUnit": "day",
11         "count": 0.5
12     };
13     dateAxis.tooltipDateFormat = "d MMM";
14
15     var valueAxis = chart.yAxes.push(new am4charts.ValueAxis());
16     valueAxis.tooltip.disabled = true;
17     valueAxis.title.text = product.currency;
18
19     var series = chart.series.push(new am4charts.LineSeries());
20     series.dataFields.dateX = "date";
21     series.dataFields.valueY = "price";
22     series.tooltipText = "Price: [bold]{ valueY }[/]";
23     series.fillOpacity = 0.3;
24
25     chart.cursor = new am4charts.XYCursor();
26     chart.cursor.lineY.opacity = 0;
27     chart.scrollbarX = new am4charts.XYChartScrollbar();
28     chart.scrollbarX.series.push(series);
29
30     dateAxis.start = 0.8;
31     dateAxis.keepSelection = true;
32 });

});
```

5.1.2. Telefonos alkalmazás

A telefonos alkalmazás Flutterben került megvalósításra. A Flutter⁸ egy nyílt forráskódú szoftver fejlesztési csomag, melyet a Google hozott a piacra 2018-ban és főként felhasználói felületek fejlesztésére alkalmazzák. Az utóbbi időben rendkívül nagy népszerűségnek örvend a platform, mivel széleskörű felhasználása van. Segítségével Android, iOS, Windows, Mac, Linux valamint a kevésbé ismert Google Fuchsia alatt futó alkalmazásokat lehet fejleszteni, amelyeket egyetlen kódállományból állít elő, azaz nem szükséges minden egyes platformra külön-külön megírni ezeket.

A Flutter alkalmazások Dart nyelven íródnak, mely szintén a Google által fejlesztett objektum orientált, osztály alapú programozási nyelv, ami 2013 debütált, viszont csak évekkel később terjedt el szélesebb körben.

A Flutter sajátossága az úgynevezett “Hot Reload”, mely lehetővé teszi a futás közbeni változtatások elvégzését, ami nagyban felgyorsítja a fejlesztési folyamatot, hiszen nem kerül az egész alkalmazás újra kompilálásra. Ezt a “just-in-time” kompilálás teszi lehetővé, melyet más nevén dinamikus fordításnak is neveznek, lényege hogy a kód futás közben kerül kompilálásra és nem pedig az alkalmazás indítása előtt. A platform egy Widget alapú megközelítést alkalmaz, azaz több definiált komponenst biztosít ahhoz, hogy felépítsük az általunk megvalósítani kívánt alkalmazást.

A telefonos alkalmazás fejlesztése Android Studio integrált fejlesztői környezetben történt, felhasználva az általa biztosított Android Emulator programot is, mely segítségével egy telefont lehet emulálni, annak érdekében, hogy az alkalmazást minél valósabb környezetben lehessen fejleszteni, minél változatosabb specifikációjú telefonon lehessen kipróbálni, természetesen ez fizikai telefonon is végezhető. A fejlesztés során úgy virtuális, mint fizikai környezetben is periodikusan ellenőrizve lett az alkalmazás.

Első lépéseben az alkalmazás bejelentkezési funkciója lett megvalósítva, amely a flutter_login⁹ nevű könyvtár segítségével történt, mivel ez nagyban felgyorsította a fejlesztést, ugyanakkor komplexebb funkcionalitásokat illetve vizuális hatásokat is tartalmaz, melyek kellemesebbé teszik a felhasználói élményt. Ilyenek például az error üzenetek megjelenítése melyek leírják milyen hiba történt, animációk. Ugyanúgy, mint az előbbiekben tárgyalt böngészős kiegészítő esetében, az alkalmazás

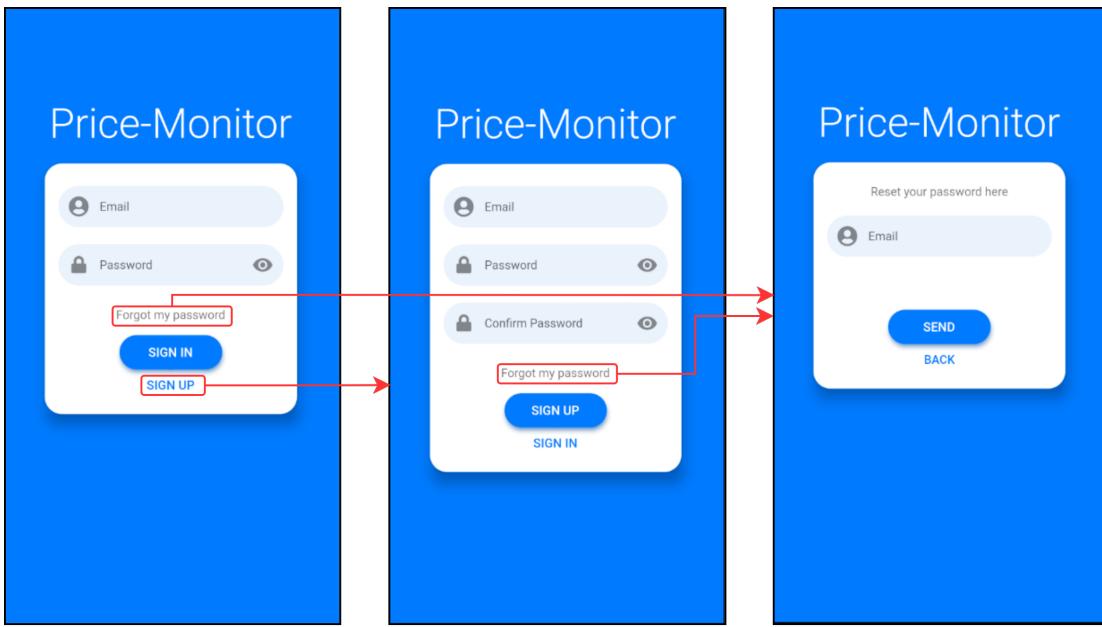
⁸<https://flutter.dev/>

⁹https://pub.dev/packages/flutter_login

telepítése után az alkalmazás első használatakor a bejelentkező felületet látjuk. Itt is lehetőségünk van regiszálni, ha még nem rendelkezünk felhasználói fiókkal, vagy amennyiben esetleg elfelejtettük a jelszavunkat, újat állíthatunk be. A bejelentkező felületet megvalósító kódrészlet valamint a tulajdonképpeni vizuális felület a 5.4 kódrészleten valamint a 5.5 ábrán lathatjuk.

Listing 5.4. Bejelentkező felületet megvalósító kódrészlet

```
1  Widget build(BuildContext context) {
2      var title = "";
3      if(MediaQuery.of(context).size.width > MediaQuery.of(context).size.height){
4          } else{
5              title = "Price Monitor";
6          }
7      return FlutterLogin(
8          title: title ,
9          theme: LoginTheme(
10             primaryColor: colorPrimaryBlue ,
11         ),
12         messages: LoginMessages(
13             loginButton: "SIGN IN",
14             signupButton: "SIGN UP",
15             forgotPasswordButton: "Forgot my password",
16             recoverPasswordButton: "SEND",
17             recoverPasswordDescription: ""
18         ),
19         onLogin: _authUser ,
20         onSignup: _registerUser ,
21         onRecoverPassword: _recoverPassword ,
22         onSubmitAnimationCompleted: () {
23             Navigator.pushAndRemoveUntil(context , MaterialPageRoute(builder: (context
24 ) => HomeScreen()), (r) => false);
25         },
26     );
}
```



5.5. ábra. Telefonos alkalmazás bejelentkezési/regisztrálási felülete

A bejelentkezés, regisztrálás a Firebase segítségével lett megvalósítva, melyek nagyon egyszerűvé és biztonságossá teszik ezeket a funkciókat. A jelszavak titkosítva tárolódnak az adatbázisban, ezáltal ezek ismeretlenek az adatbázist kezelő személyzetnek is. A könyvtár csomag biztosítja számunkra továbbá a jelszavak cseréjét, amennyiben elfelejtettük a régit vagy csak egyszerűen cserélni szeretnénk. Az ezeket megvalósító függvényeket és használatukat a 5.5 kód részlet illusztrálja.

Listing 5.5. Bejelentkező/regisztráló/jelszó csere funkciót megvalósító függvények

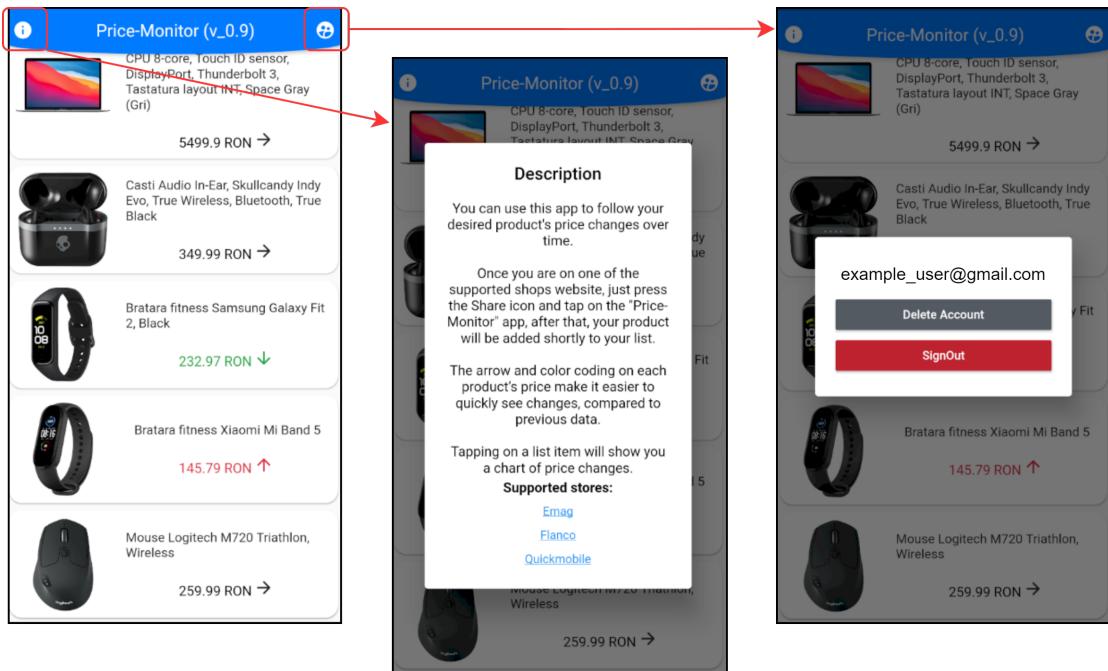
```

1 // Sign in
2 await _auth.signInWithEmailAndPassword(email: data.name, password: data.
3
4 // Register
5 await _auth.createUserWithEmailAndPassword(email: data.name, password: data.
6
7 // Reset Password
8 await _auth.sendPasswordResetEmail(email: email);

```

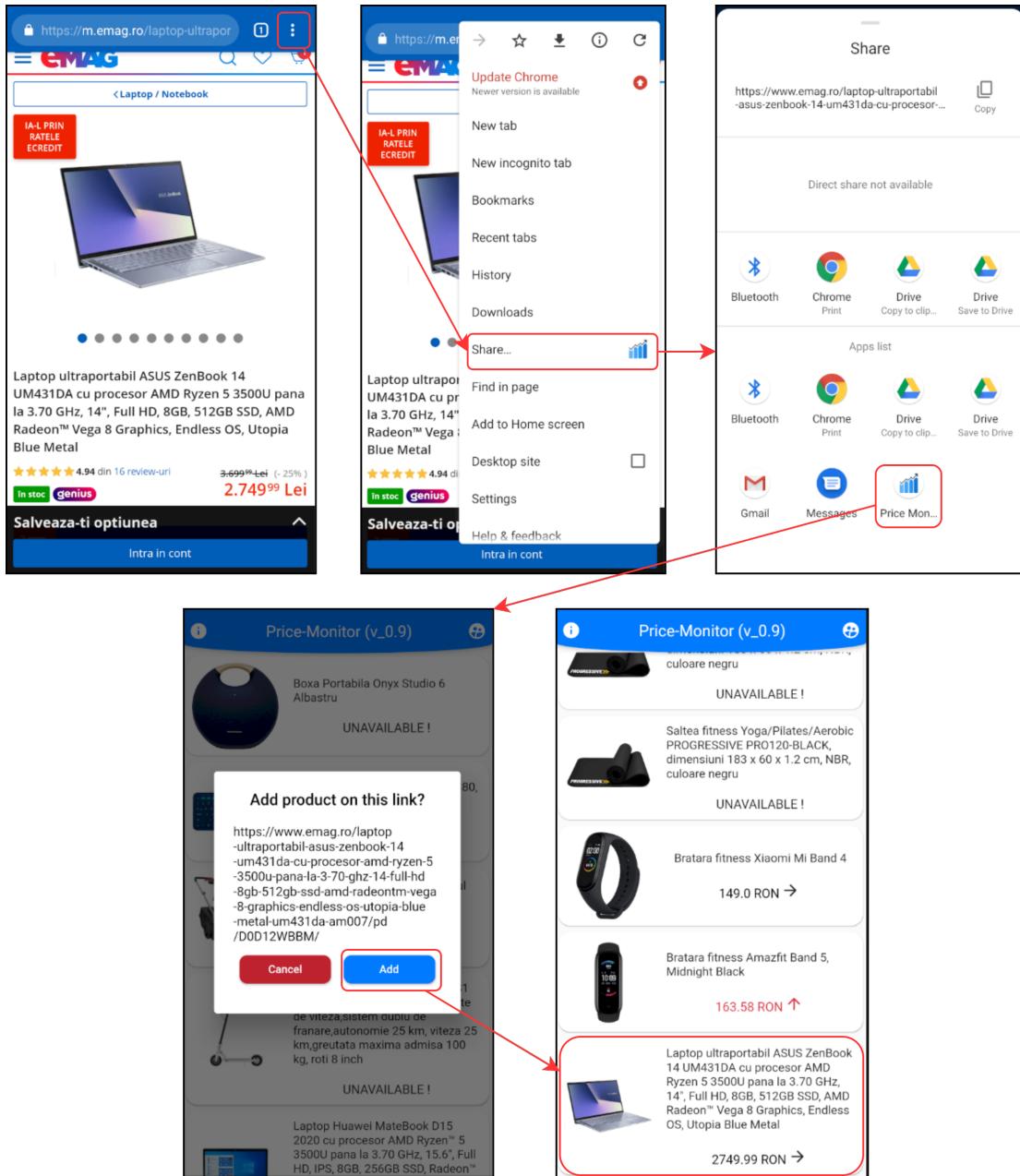
A felület készítésekor az egységesség játszott nagy szerepet, ezért túlnyomó részben ugyanazt a

design nyelvet követtük, mint a már bemutatott böngészős kiegészítő esetében, de telefonos formára igazítva. Ennek megfelelően, a bejelentkezés után szintén egy görgethető listában láthatja a felhasználó a követett termékeit, mint azt a 5.6 ábra is mutatja. A felhasználó számára ugyanúgy rendelkezésére áll egy információs, valamint egy a felhasználói fiókjához kapcsolódó menüpont melyek a fejlécben kaptak helyet. Funkciójuk az alkalmazás rövid leírása, támogatott weboldalak listázása, valamint a felhasználó fiókjához tartozó műveletek elérése (5.6 ábra).



5.6. ábra. Főoldal, adminisztrációs rész

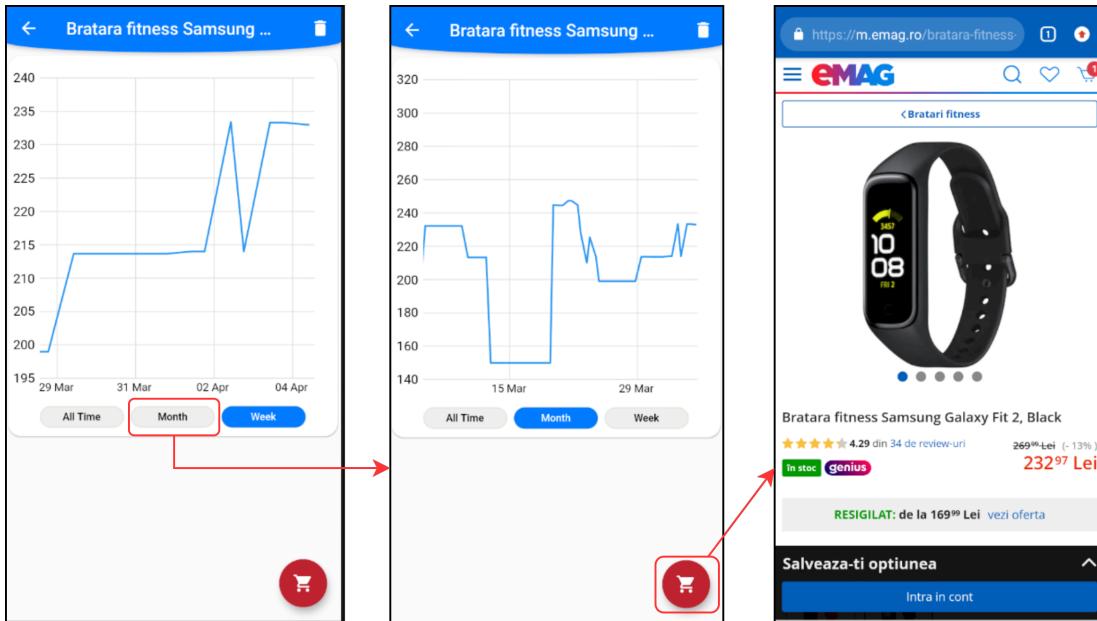
Egy termék hozzáadása a telefon megosztási menüpjból történik. Ehhez, az adott webshop oldalán kell lennünk, vagy amennyiben az adott oldal rendelkezik telefonos alkalmazással, mint például az Emag esetében, akkor ezt a funkciót onnan is el lehet érni. Ha megtaláltuk a követni kívánt terméket, a telefon megosztási menüpjbén ki kell választani az alkalmazás ikonját, ahogy ezt a 5.7 ábra mutatja. Miután ez megtörtént, át is kerülünk az alkalmazásba, ahol egy felrúgó ablakban jelenik meg számunkra, hogy biztosan hozzá szeretnénk-e adni az adott terméket. Amennyiben ezt jóváhagytuk az “Add” gomb megnyomásával, pillanatokon belül láthatjuk is a termékünket a lista alján (5.7).



5.7. ábra. Termék hozzáadása

Ebben az esetben is, a termékek árainak színe, illetve a mellettük található nyíl jelképezi, hogy az ár hogyan változott az utolsó ellenőrzés óta, a zöld szín, valamint lefele irányuló nyíl csökkenést, a piros szín, felfele irányuló nyíllal drágulást, az egyenesen irányított nyíllal pedig az ár változatlanságát jelképezni. Amennyiben a termék nem elérhető, az ár helyét az „UNAVAILABLE” szöveg veszi át. Amikor egy elemet kiválasztunk a listából, az alkalmazás átvisz minket egy másik oldalra, ahol az árak

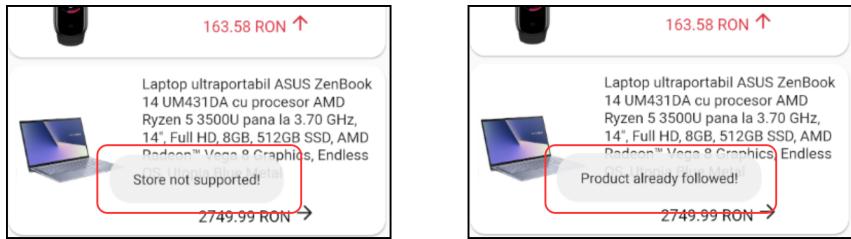
változását láthatjuk egy diagram formájában. A diagram alapértelmezetten az elmúlt heti változásokat mutatja, viszont a diagram alatt található gombok segítségével változtathatjuk az idő skálát, a heti szint mellett hónapos vagy a teljes adathalmaz ábrázolása érdekében, melyek működését a 5.8 ábrán lathatjuk.



5.8. ábra. Termék árváltozása

Az ablak jobb alsó sarkában található egy bevásárló kosárral jelölt gomb, melyet, ha megnyomunk az alkalmazás átirányít minket és a telefon alapértelmezett böngészőjében megnyitja nekünk a termék weboldalát (5.8). Amennyiben már nem vagyunk érdekeltek a termék követésesben, a jobb felső sarokban található, kukával jelölt gombbal törölhetjük azt a követési listánkból.

Ha egy nem támogatott oldal termékét próbáljuk meg hozzáadni, akkor egy felrúgó úgynevezett „toast” üzenetben az alkalmazás közli a felhasználóval hogy nem támogatott oldalról van szó a "Store not supported!" szöveggel, ugyanakkor ha már követi az adott terméket, akkor annak megfelelő „Product already followed!” szöveget ír ki, ahogy az a 5.9 ábrán látható.



5.9. ábra. Informáló üzenetek

5.1.3. Backend

A rendszer backend részéért egy Python script valósítja meg. Ez Python3-as nyelven íródott, felhasználva több, a célnak megfelelő függvény könyvtárat, mint például request: a http kérések intézéséhez, BeautifulSoup: a HTML kódban való kereséshez, különböző Firebase függvény könyvtárak: az egyes adatbázissal kapcsolatos műveletek megvalósítására, Schedule: az időszakos ellenőrzések automatizálására, valamint logging: az események naplózására, valamint terminálra való egységes rendezett kiírására.

A rendszernek ez a része felelős a termékek hozzáadásáért, azok árainak frissítéséért. Ez úgy történik, hogy lekéri az adatbázisból a termékek URL-jét, majd ezekhez egy HTTP kérést intéz. Az adott weboldal visszaküldi a megjelenítéshez szükséges HTML kódot, melyet aztán a BeautifulSoup függvény könyvtár segítségével, egy elemenként kereshető formátumra alakítja. Ebben már tulajdonképpen HTML tag-ekre kereshetünk, megadva bizonyos tulajdonságaikat, ezekből kinyerjük a számunkra szükséges információkat, amik a termék megnevezése, ára, valamint egy hozzá tartozó kép. A rendszer ezekhez társítja az ellenőrzés időpontját, majd ez feltöltésre kerül az adatbázisba, a megfelelő felhasználóhoz kötve.

A script terminálban fut, indítás után kiválaszthatjuk milyen futási módban szeretnénk indítani (5.10 ábra). Ezek a Continuous vagy folytonos, illetve az Update. A folytonos mód kiválasztása esetén a rendszer elindítja az új termékek hozzáadását figyelő és az ütemezett ellenőrzésekért felelős modulokat. Az új termékek hozzáadásáért felelős modul folyamatosan figyeli, hogy az adatbázis részéről érkezett-e valamiféle jelzés. Amennyiben egy új terméket akarunk hozzáadni az alkalmazásokon keresztül, ahogy azt az előbbi modulok tárgyalásánál láthattuk, a követni kívánt termék hozzáadásra kerül az adatbázisba, ami ezek után jelzést küld a Python scriptnek. Amint megérkezett a

jelzés, azonnal elkezdődik az adatok kinyerése és hozzáadása a felhasználó listájához, az előzőekben tárgyalt módon. Az ütemezett ellenőrzésekért felelős modul segítségével, előre meghatározott időpillanatokban elindul az árak frissítése, amely az adatbázisban található összes termék árát frissíti. Ezen a modulon belül találunk egy az internetkapcsolatot figyelő részt is, amely egy esetleges kapcsolat megszűnése esetén azonnal próbál újrakapcsolódni, amint ez sikerül, újraindítja a rendszer funkcióit, amelyek esetleg leálltak az internetkapcsolat megszűnésekor. Az Update módot kiválasztva, a rendszer azonnal elkezdi az összes termék frissítését, lásd 5.10 ábra, majd miután végzett, automatikusan folytatja a futást folytonos módban. Az ehhez tartozó megvalósítás a 5.6 kódrészleten látható.

```

Choose running mode:
 1 - Continuous (Will run scheduled tasks and start all listeners)
 2 - Update (Will update all products then resume in Continuous mode)

Mode: 2
INFO | pricetracker.py | 2021-05-17 12:55:57,760 | Starting in UPDATE mode |
INFO | pricetracker.py | 2021-05-17 12:55:59,526 | Updating prices |
INFO | pricetracker.py | 2021-05-17 12:56:35,342 | Updated | Boxa Portabila Onyx
INFO | pricetracker.py | 2021-05-17 12:56:46,947 | Updated | Tastatura Bluetooth
INFO | pricetracker.py | 2021-05-17 12:56:58,697 | Updated | Tastatura Logitech
INFO | pricetracker.py | 2021-05-17 12:57:10,212 | Updated | Masina electrica de
INFO | pricetracker.py | 2021-05-17 12:57:22,257 | Updated - No stock | Coasa de
INFO | pricetracker.py | 2021-05-17 12:57:34,167 | Updated | Radiator (calorifer)
INFO | pricetracker.py | 2021-05-17 12:57:45,982 | Updated - No stock | Gratar
INFO | pricetracker.py | 2021-05-17 12:57:57,777 | Updated - No stock | Gratar
INFO | pricetracker.py | 2021-05-17 12:58:09,467 | Updated | Anvelopa vară Michel
INFO | pricetracker.py | 2021-05-17 12:58:21,092 | Updated - No stock | Trotinet
INFO | pricetracker.py | 2021-05-17 12:58:32,640 | Updated - No stock | Trotinet

```

5.10. ábra. Python script Update módban való indítása

Listing 5.6. Backend szolgáltatás indítása

```

1 def run(mode):
2     try:
3         if mode is None:
4             print ("\nChoose running mode:\n"
5                   "    1 - Continuous (Will run scheduled tasks and start all
6                   listeners)\n"
7                   "    2 - Update (Will update all products then resume in
8                   Continuous mode)\n")

```

```

7     mode = int(input("Mode: "))
8
9     if (mode != 1) and (mode != 2):
10         print(" Invalid option , please select again !")
11         return run(None)
12
13 else :
14     if mode == 1:
15         logger.info(" Starting in CONTINUOUS mode ")
16         run_new_products_listener()
17         run_scheduled_checks()
18     if mode == 2:
19         logger.info(" Starting in UPDATE mode ")
20         update_prices()
21         logger.info(" All products updated , starting CONTINUOUS mode !")
22         run_new_products_listener()
23         run_scheduled_checks()
24
25 except Exception as error:
26     logging.critical(" Error while starting service !" + str(error))
27     for i in range(5, 0, -1):
28         logger.info(" Retrying in ... " + str(i) + " seconds ")
29         time.sleep(1)
30
31 return run(mode)

```

A funkciók könnyebb kezelése végett készült egy felhasználói felület az admin szerepet betöltő felhasználó számára is, ez PyQt5¹⁰ segítségével lett megvalósítva. A felület az előbbiekben említett funkcionálitásokat képés elindítani, leállítani, valamint lehetőséget nyújt a konfigurációs file módosítására, amely tartalmazza a program működéséhez szükséges információkat, legfontosabbak ezek közül az egyes weboldalak által használt tag-ekkel kapcsolatos információk, amelyek alapján történik az egyes adatok bányászása. Ezek módosítását teszi lehetővé a 5.11 ábrán látható fogaskerékkel jelölt gomb.

Az árak napi kétszeri frissítésekor, a rendszer a 5.7 pszeudokód alapján végzi az aktualizálást.

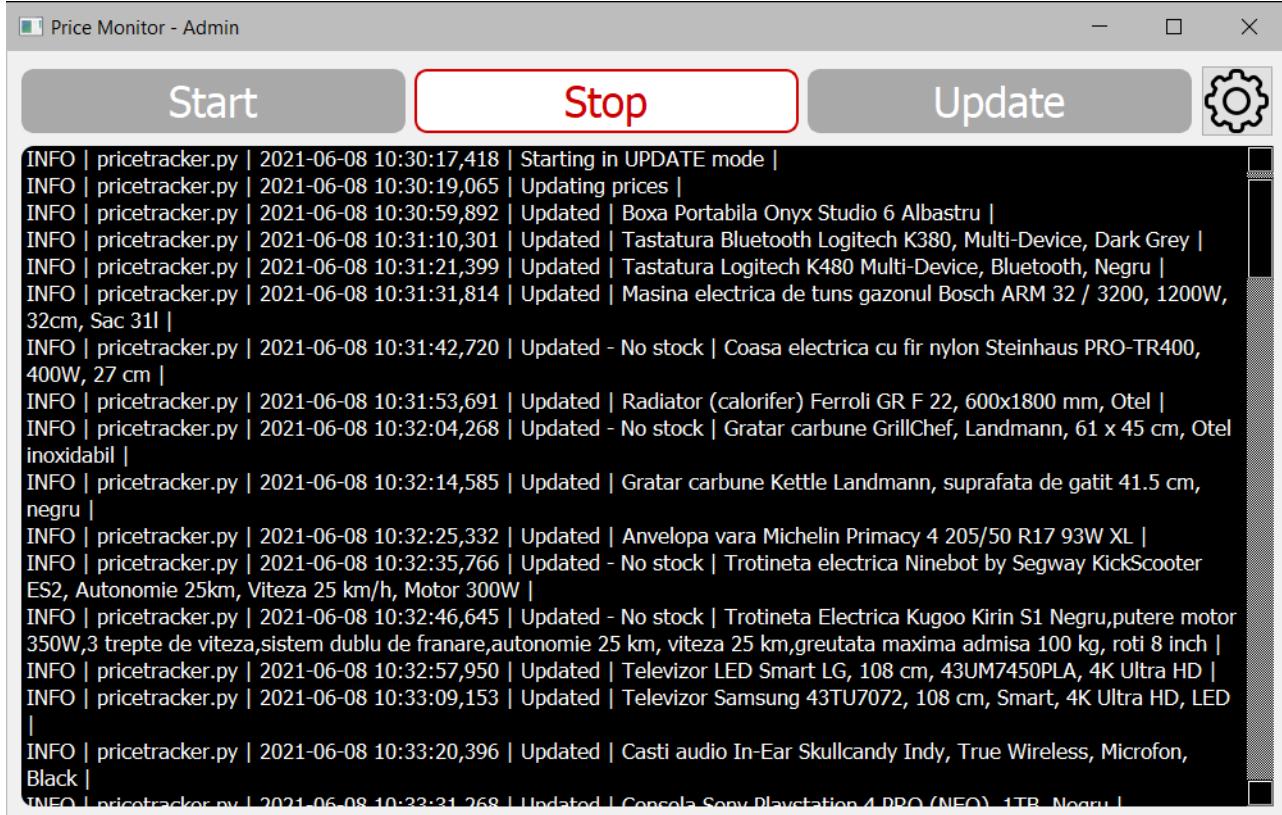
¹⁰<https://pypi.org/project/PyQt5/>

Listing 5.7. Termékek árának frissítését leíró pszeudokód

```

1 Update Prices
2     Get users list from database
3     for user in users_list:
4         Get product list of user
5         for product in product_list:
6             get product price from url
7             check for potential errors
8             if error then
9                 upload error to database
10            else
11                upload new price to database

```



5.11. ábra. Backend rész adminisztrációs felülete

Amint az előbbiekben szó volt, a BeautifulSoup nevű függvénykönyvtár segítségével lesznek ki-nyerve a szükséges információk, a weboldalak HTML kódjából. Ez a művelet a „find” nevű függ-

vénnyel történik, melynek meg kell adni két paramétert. Az egyik, hogy milyen típusú tag-eket szeretnénk keresni, mint például, div, span, h1, p, a másik pedig hogy ezek milyen tulajdonságokkal rendelkeznek, például a tag egy bizonyos osztálynével van ellátva. A 5.8 kódrészlet tartalmazza egy termék adatainak feldolgozását, amennyiben az, az Emag webáruhásról származik.

Listing 5.8. Emag linket feldolgozó függvény

```

1  def get_and_parse_emag(soup):
2      title = find_title(soup, "h1", "page-title")
3      out_of_stock = soup.find("span", attrs={"class": "label-out-of-stock"})
4      if out_of_stock or (title is constants.error):
5          price = constants.error
6      else:
7          try:
8              form = soup.find("form", attrs={"class": "main-product-form"})
9              price = form.find("p", attrs={"class": "product-new-price"}).text.
10             strip()
11             s = list(price)
12             s.insert(-6, ",")
13             price = "".join(s)
14             price = re.sub("Lei", "", price)
15             price = re.sub("\.", "", price)
16             price = re.sub(",", ".", price)
17             try:
18                 price = float(price.strip())
19             except ValueError:
20                 price = re.sub("de la", "", price)
21                 try:
22                     price = float(price.strip())
23                 except ValueError:
24                     price = constants.error
25             except AttributeError:
26                 price = constants.error
27             prod_currency = currency.ron
28             try:

```

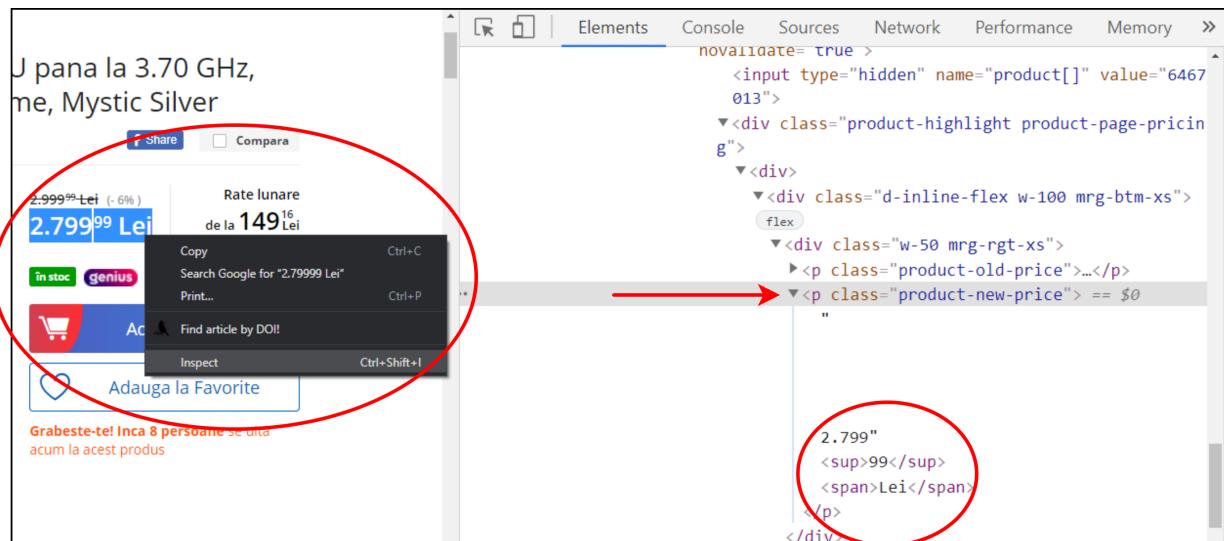
```

28     image = soup.find("div", attrs={"id": "product-gallery"}).img["src"]
29 except AttributeError:
30     image = "error"
31
32 product_data = class_ProductData.ProductData(title, price, prod_currency,
image)
33
34 return product_data

```

5.1.4. Web áruházak

A rendszer fontos részét képezik maguk a web áruházak is, hiszen ezek biztosítják számunkra a szükséges adatokat. A weboldalak HTML nyelven felépített elemekből állnak, melyeket a böngészőnk megjelenít számunkra. Ezt a struktúrát bárki megtekintheti, ha egy weboldalon tartózkodva megnyomja az F12 billentyűt, vagy egy jobb klikket követően az Inspect opciót kiválasztva. Amint azt korábban is említettük, az alkalmazás ezek között a struktúrák között keresi a számunkra értékes információt. Hogy ezt hogyan is találjuk meg, azt a 5.12 ábrán láthatjuk is. Ezen egy termék ára lett kijelölve, majd az előbb említett Inspect opció lett kiválasztva. A böngészőnk ki is adta, hogy a kijelölt információ egy `<p>` tag alatt található, amely egy „product new price” osztálynévvel van ellátva. Az alkalmazás pontosan ezeket az információkat bányássza ki egy weboldal strukturájából, legyen az az ár, a megnevezés vagy akár a társított kép elérhetősége.



5.12. ábra. Információ lokalizálása a weboldal strukturájában - Emag

Mivel minden weboldal más struktúrával rendelkezik, ezért az ezeken található információ is más-képp van elrendezve. Bármilyen weboldalon szeretnénk keresni, nagy valószínűséggel mindegyiknek más, egyedi struktúrája lesz (lásd 5.13 ábra), ezért ezeket üzletre szabottan kell kezelni, viszont ami közös, az az, hogy alapvetően minden információ valamelyik klasszikus HTML tag között lesz eltárolva. Ugyanakkor, előfordulnak változások is a struktúrában, ezért ezeket folyamatosan figyelni kell és el kell végezni a szükséges változtatásokat, annak érdekében, hogy a szoftver helyesen működjön.

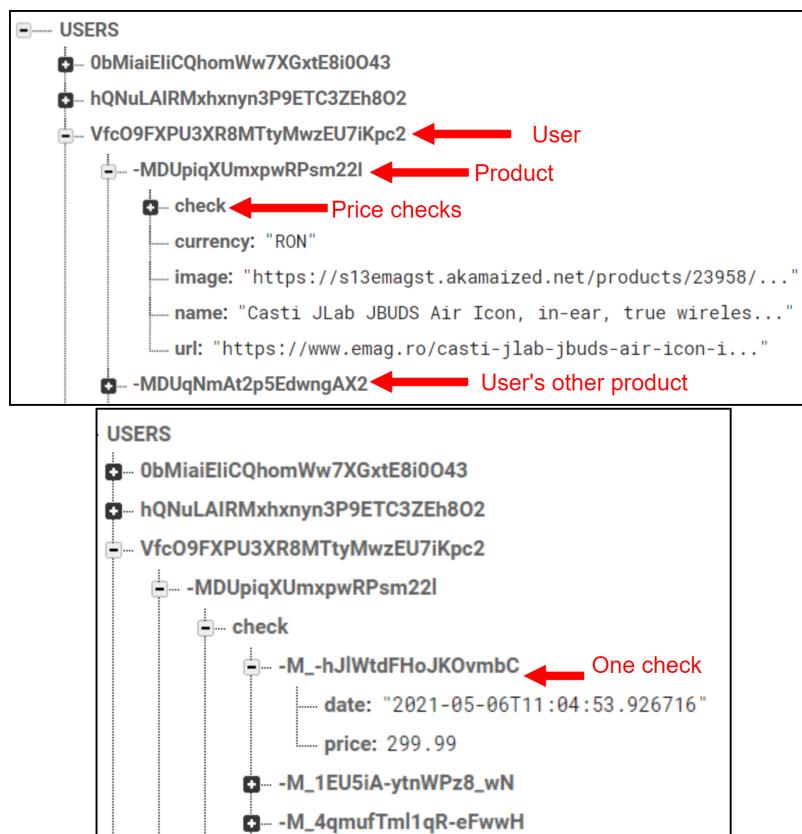
The figure consists of three side-by-side examples of e-commerce website pricing structures, each with its HTML source code displayed on the right.

- Emag:** A screenshot of an Emag product page showing a price of 1.799 Lei. A red circle highlights the price "1.799 Lei". To the right, the HTML code shows the price is stored in a `⁹⁹` element within a `Lei`.
- Flanco:** A screenshot of a Flanco product page showing a price of 1.749,99 lei. A red circle highlights the price "1.749,99 lei". To the right, the HTML code shows the price is stored in a `1.749,"` element with a `content="1749.99"` attribute.
- QuickMobile:** A screenshot of a QuickMobile product page showing a price of 166,90 Lei. A red circle highlights the price "166,90 Lei". To the right, the HTML code shows the price is stored in a `<div>166.89988</div>` element with a `content="166.89988"` attribute.

5.13. ábra. Támogatott weboldalak különböző strukturája

5.1.5. Adatbázis

Az rendszer adatbázisként a Firebase Realtime Database-t használja. Ez az adatbázis a Google platform által biztosított valós idejű adatbázis, amely számos funkcióval rendelkezik. Egy NoSQL adatbázisról van szó, amely az adatokat egy JSON formátumhoz hasonló struktúrában tárolja, ezáltal megkönnyítve a használatát, valamint az integrációt az egyes programozási nyelvek esetében. A választás egy NoSQL adatbázisra továbbá azért esett, mivel ez a formátum sokkal rugalmasabb, mint a hagyományos SQL adatbázisoké, egy esetleges változtatás vagy kiegészítés során jóval könnyebb módosítani az egyes struktúrákon. A rendszer valós idejűsége miatt, az adatok azonnal szinkronizálódnak az összes felhasználó között, így mindenki a legaktuálisabb adatokat látja. Az adatok ábrázolása az adatbázisban 5.14 ábrán látható.



5.14. ábra. Adatok tárolása az adatbázisba

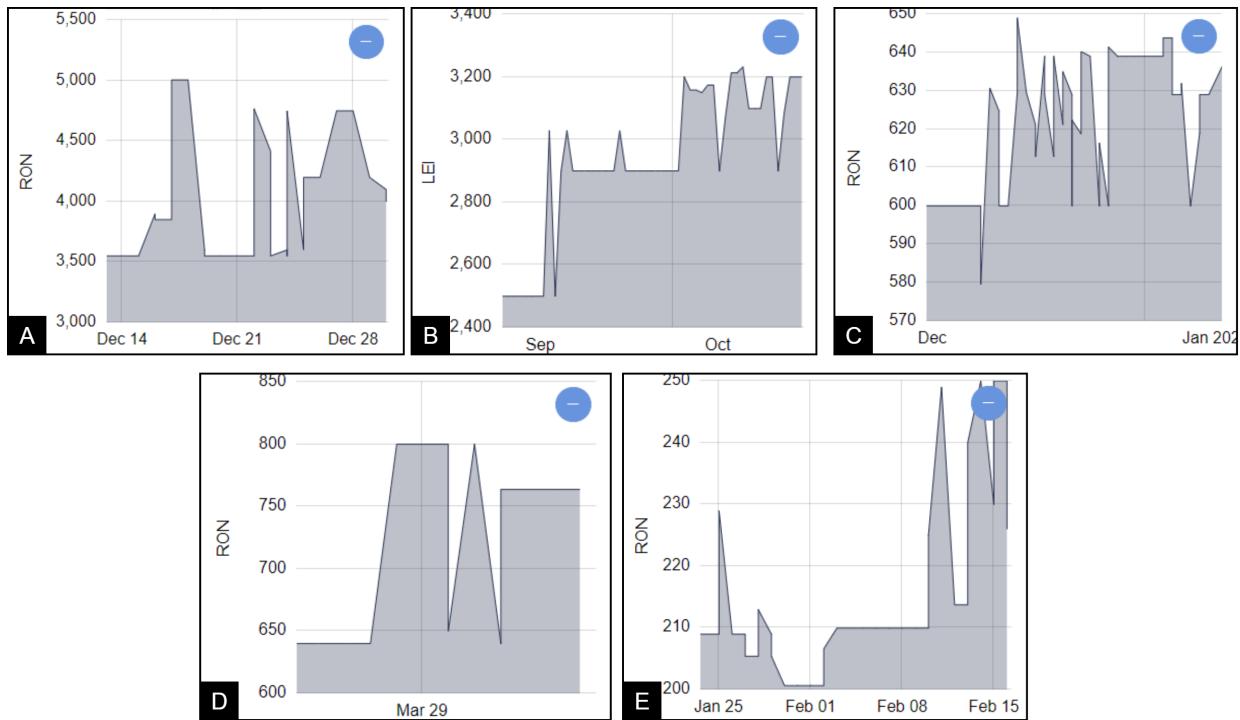
6. fejezet

Esettanulmány

Mivel a dolgozatban bemutatott szoftver azt a célt szolgálja, hogy egy esetleges vásárló reális képet alkossan egy termék árának változásáról, ezért ebben a fejezetben a használat során történt észrevételek lesznek bemutatva, tanulmányozva. A tanulmány a legnagyobb Romániai online áruházra fog koncentrálni, mely jelen pillanatban az Emag, ugyanakkor nagyító alá kerülnek a 2020-as Black Friday alatt történt észrevételek és események is.

Mivel nagyon sokszor hallottunk olyan történeteket, melyekben az emberek hamis árakról vagy promóciókról panaszkoztak, ezért a fejlesztett szoftvert felhasználva, követtünk pár népszerű terméket, melyeket egy átlag felhasználó megvásárolna, ilyenek például pár mobil telefon, úgy felső, mint közép kategóriából, az árukat tekintve, laptopok és egyéb elektronikai eszközök, de pár parfüm is helyet kapott. Az észrevételek esetenként több hónapnyi adatókból lettek megfogalmazva, viszont ezek közül néhány kifejezetten a népszerű Black Friday periódusra koncentrálódik.

A legsűrűbbén észrevett minta, amely szinte minden termék esetében megjelenik, az a hirtelen árnövekedés, mely egy napig vagy sok esetben csak fél napig van jelen. Ilyenkor úgy tűnik mintha a kereskedő megpróbálná úgymond drágábban eladni a terméket, vagyis teszteli a piacot. Megfigyelhető az is, hogy egy pár ilyen hirtelen növekedés után, egy kis idő elteltével stabilizálódik a termék ára sokszor egy magasabb értéken, majd onnan megint következik az újabb időnkénti növekedés. Mindez a 6.1 ábrán látható is.



6.1. ábra. Megfigyelt időszakos árnövekedések különböző termékek esetében

A - iPhone 11 - <https://tinyurl.com/5xa8y7u7>

B - Huawei Matebook laptop - <https://tinyurl.com/yw99mcdp>

C - Samsung Galaxy A20e - <https://tinyurl.com/2v9hdj68>

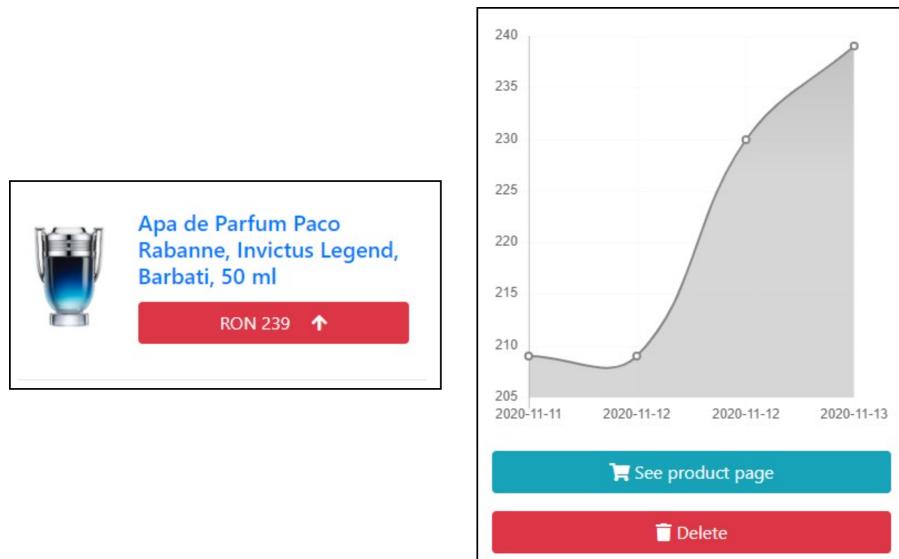
D - Xiaomi Poco M3 - <https://tinyurl.com/2kp3ezxk>

E - Samsung Galaxy Fit 2 smartband - <https://tinyurl.com/dnz6b7pc>

6.1. Black Friday

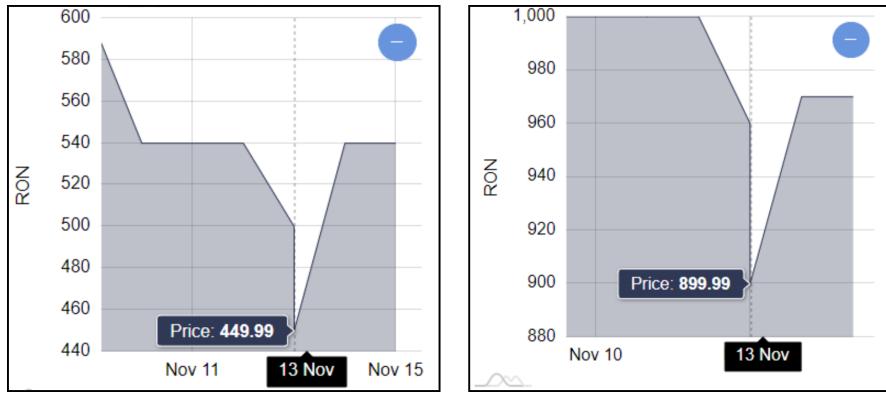
Mivel a szoftver létrejöttet nagyban inspirálta a Black Friday-kor történő manipulálások, ezért egy pár észrevétel ehhez kötődően is bemutatásra kerül. Romániában a Black Friday 2020-ban November 13 rendeződött meg. Annak köszönhetően, hogy rengeteg termék található az Emag online áruházában, ezért nehéz általánosítani és azt mondani, hogy minden termék hamis leszállítással vagy promócióval rendelkezik, viszont ezek elfőfordulnak. Ezért a következőkben egy pár általunk észlelt jelentős és érdekes eset lesz bemutatva.

Első esetben egy parfüm árának figyelésekor tettük meg azt az észrevételt, hogy a termék ára egy nappal a Black Friday időpontja előtt, megnőtt az előző napokhoz képest. Eddig semmi meglepő nem történt, hiszen erre számítottunk is sok termék esetében, viszont ebben az esetben az érdekességet az képviselte, hogy a termék drágább lett, de az esemény napján várt esés nem történt meg, pedig a szóban forgó parfüm akciósként volt megjelölve (6.2 ábra). Egy másik termék esetében, mely szintén egy parfüm volt, ugyanez a jelenség játszódott le, a termék drágább volt az esemény napján, az azelőtti árához képest, viszont, ugyanez a termék ugyanaz nap egy esti órájában még a reggeli árat is meghaladta, mindeközben egyre nagyobb árleszállítást reklámozva. Tehát az árleszállítás mértéke nagyobb lett, viszont a termék drágult.



6.2. ábra. Példa az ár növekedésére már Black Friday előtt

Egy másik érdekes észrevétel ugyanilyen formában fogalmazódott meg, több termék árának többszöri változásával a nap folyamán, de ezekben az esetekben, ezek reális csökkenések voltak. Ezek a bizonyos észrevételek telefonok követésekor voltak megfigyelve, amik esetében tényleg jóval alacsonyabb volt az akciósként megjelölt ár, az előző időszakhoz képest. Mivel az előző esetben tárgyalt többszöri változást láttuk már, ezekben az esetekben a leírt jelenség fordítottja történt. Vagyis az esemény napján, a termék ára kétszer is csökkent, mindenkor reálisan (6.3 ábra).



6.3. ábra. Többszörös árcsökkentés Black Friday napján

Mivel a laptopok az egyik legnépszerűbb termékek voltak idén ősszel, a kialakult vírus helyzet miatt otthonról dolgozók, illetve tanulók köreiben, természetesen ezek sem maradhattak el a várt kedvezményektől legalábbis a látszólagosaktól. Az akkoriban rendelkezésre álló adathalmazban csupán egyetlen laptop volt jelen, amire érvényes volt az esemény előtt pár nappal való ár növekedés (6.4).



6.4. ábra. Laptop mesterséges árleszállítása

7. fejezet

Összefoglalás

Röviden összefoglalva a fentebb tárgyaltakat, el lehet mondani, hogy sok esetben megalapozottan ítélijük meg az egyes webáruházakat az árak tudatos manipulálása miatt, viszont mint mindig, ebben az esetben sem lehet általánosítani, hiszen az adatok elemzése során találkoztunk valós kedvezményekkel is, nem csak gyanúsan változó árakkal, melyek félrevezetik vagy egyenesen becsapják a vásárlót. Ezeket figyelembevéve, érdemes minden több ideig figyelni valamit, ami iránt esetleg vásárlási szándékunk van és nem szabad pusztán az eladóra hagyatkozni, amikor kiírja a termék régi árát, egy nagyobb árleszállítás keretein belül. Erre a célról lett megtervezve és megvalósítva a dolgozatban bemutatott szoftver mely lehetőséget nyújt a felhasználóknak vagy potenciális vásárlóknak automatizálni az esetleges kívánt termékek napi ellenőrzését és az árak összehasonlítását. Mindezt megteheti bárhol is tartózkodik, a megvalósított telefonos alkalmazás segítségével, de egy egyszerűen használható böngészős kiegészítőből is, ha netán inkább az áll közelebb hozzá.

Természetesen egy ilyen típusú alkalmazás fejlesztése folyamatos munkát igényel. Van helye további fejlődésnek, változásnak, optimalizálásnak a jövőre nézve. Mivel a webáruházak struktúrája folyamatosan változik, ezért a háttérben működő szoftvert folyamatosan igazítani kell. Továbbá, egy lehetséges fejlesztés az új webshopok támogatása a jövőben, úgy hazai mint külföldi oldalakat figyelembe véve. Felhasználói felület szintjén is van hely fejlődésnek vagy új akár funkciók hozzáadásának, melyet majd idővel meg is lehet tenni.

7.1. Megvalósítások

A dolgozatban leírtak alapján a következők lettek megvalósítva:

- Web Scraping tanulmányozása szakirodalom által
- Web Scraping jogi és etikai hátttereinek tanulmányozása
- Web áruházak struktúrájának tanulmányozása
- Az elkészítendő rendszer specifikálása
- Firebase adatbázis be üzemelése
- Web Scraper implementálása Python script által
- Böngésző kiegészítő megvalósítása
- Telefonos alkalmazás megvalósítása Flutter segítségével
- Admin felület megvalósítása PyQt5 segítségével
- Esettanulmány az árak változásáról, külön kitérve a 2020-as Black Friday periódusra

Irodalomjegyzék

- [1] F. Johnson and S. K. Gupta, „Web content mining techniques: a survey,” *International Journal of Computer Applications*, vol. 47, no. 11, 2012.
- [2] F. Gorunescu, *Data Mining: Concepts, models and techniques*, vol. 12. Springer Science & Business Media, 2011.
- [3] B. G. Dastidar, D. Banerjee, and S. Sengupta, „An intelligent survey of personalized information retrieval using web scraper,” *International Journal of Education and Management Engineering*, vol. 6, no. 5, pp. 24–31, 2016.
- [4] R. N. Landers, R. C. Brusso, K. J. Cavanaugh, and A. B. Collmus, „A primer on theory-driven web scraping: Automatic extraction of big data from the internet for use in psychological research.” *Psychological methods*, vol. 21, no. 4, p. 475, 2016.
- [5] D. S. Sirisuriya *et al.*, „A comparative study on web scraping,” 2015.
- [6] V. Krotov and L. Silva, „Legality and ethics of web scraping,” 2018.
- [7] D. Ni, „Is web scraping legal?”