

SZAKDOLGOZAT

Radó Bence

2021

Pannon Egyetem

Műszaki Informatikai Kar

Rendszer- és Számítástudományi Tanszék

Programtervező informatikus BSc

SZAKDOLGOZAT

Kotlin web scraping fejlesztése

Radó Bence

Témavezető: Machalik Károly

2021



PANNON EGYETEM

MŰSZAKI INFORMATIKAI KAR

Programtervező informatikus BSc szak

Veszprém, 2021. március 10.

SZAKDOLGOZAT TÉMAKIÍRÁS

Radó Bence

Programtervező informatikus BSc szakos hallgató részére

Kotlin web scraping fejlesztése

Témavezető: Machalik Károly

A feladat leírása:

Napjainkban jellemző, hogy az számunkra értékes adatokat az internetről, weboldalakról szeretnénk elérni. A legtöbb weboldal nem csak a hasznos információt jeleníti meg, hanem design elemeket, reklámokat, nem releváns adatokat is. A hallgató feladata egy olyan alkalmazás tervezése, valamint megvalósítása, amely hatékonyan képes kiolvasni és feldolgozni a weboldalakon található adatokat. A rendszer biztosítsa, hogy a felhasználók vagy a fejlesztők egyszerűen tudják paraméterezni, hogy milyen adatokat szeretnének elérni.

Feladatkiírás:

- A hallgató ismerje meg a témakör elméleti hátterét, és tekintse át a piac jelenlegi állapotát!
- Tervezzon meg és hozzon létre olyan alkalmazást, amely hatékonyan támogatja az adatok lekérdezését!
- Implementáljon olyan felhasználói interfész modulokat, amelyek az alkalmazás által szolgáltatott adatok a mai kor elvárásainak megfelelően érhetők el!
- Mintapéldákon keresztül mutassa be az elkészült alkalmazás felhasználási lehetőségeit!
- Tesztelje a kifejlesztett rendszert és értékelje az elért eredményeket!

Dr. Süle Zoltán
egyetemi docens
szakfelelős

Nyilatkozat

Alulírott Radó Bence hallgató, kijelentem, hogy a dolgozatot a Pannon Egyetem Rendszer- és Számítástudományi Tanszék tanszékén készítettem a programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Veszprém, 2021. május 6.

.....

Aláírás

Alulírott Machalik Károly témavezető kijelentem, hogy a dolgozatot Radó Bence a Pannon Egyetem Rendszer- és Számítástudományi Tanszék tanszékén készítette programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védeésre bocsátását engedélyezem.

Veszprém, 2021. május 6.

.....

Aláírás

Köszönetnyilvánítás

Köszönetet szeretnék nyilvánítani Machalik Károlynak, aki végig segítette munkámat, illetve szakmai tanácsokkal látott el.

Köszönetet mondok a Pannon Egyetem Tanárainak, akiktől a megfelelő tudást elsajátíthattam az évek során.

Külön köszönettel tartozom Édesanyámnak és Édesapámnak, akik támogattak tanulmányaim során és bíztak bennem.

TARTALMI ÖSSZEFOGLALÓ

Dolgozatom témája: Kotlin web scraping fejlesztése. A web scraping egy gyors és könnyű módja különböző adatok kinyerésének többféle weboldalról. Nagyon sokféle web scraping program létezik, viszont mindegyiknek meg van az előnye, illetve a hátránya is. Témám során a Kotlin nyelvet használtam, és arra törekedtem, hogy a legkönnyebben használható és testreszabható programot készítsek el, ami mindenki számára könnyen használható.

Az alkalmazás amit készítettem, képes képek, linkek, email címek, forráskód és táblázatok kinyerésére. Képek esetében kiterjesztés alapján, email címek esetében pedig domain-ek alapján további szűrőfeltételek állíthatók be. Táblázatok esetében megadható, hogy hány darab táblázatra és ezekből a táblázatokból hány darab sorra, illetve oszlopra van szükségünk.

Kulcsszavak: web scraping, kotlin, jsoup, adatok kinyerése, táblázatok

Abstract

The topic of my thesis is: Kotlin web scraping development. Web scraping is a quick and easy way to extract different data from different websites. There are many types of web scraping programs, but each has its advantages and disadvantages. In my topic, I used the Kotlin language and sought to create the easiest-to-use and customizable program that was easy for everyone to use.

The application I created is capable of extracting images, links, email addresses, source code and spreadsheets. Additional filter criteria can be set based on extensions for images and domains for email addresses. In the case of tables, we can specify how many tables and how many rows or columns are needed from these tables.

Keywords: web scraping, kotlin, jsoup, data extraction, spreadsheets

Tartalomjegyzék

1. Bevezetés.....	1
2. Kutatás	3
2.1 Import.io	3
2.2 OutWit Hub	5
2.3 Beautiful Soup	6
2.4 Scrapy	7
3. Tervezés	9
4. Fejlesztés	11
4.1 JSoup	11
4.2 Felépítés.....	13
4.3 ignoreContentType	13
4.4 userAgent.....	13
4.5 referrer	14
4.6 timeout.....	14
4.7 maxBodySize.....	14
4.8 execute	15
4.9 Email címek kinyerése	15
4.10 Képek kimentése	16
4.11 Linkek kinyerése	17
4.12 Forráskód kinyerése	18
4.13 Táblázatok kinyerése	18
5. Tesztelés.....	20
5.1 Email címek kinyerése	20
5.2 Képek kimentése	23
5.3 Linkek kinyerése	25
5.4 Forráskód kinyerése	27
5.5 Táblázatok kinyerése	28
5.6 Összes funkció.....	31
6. Felhasználás	33
6.1 Maga az alkalmazásról	33
6.2 Email címek kinyerése	34

6.3 Képek kimentése	35
6.4 Linkek kinyerése	36
6.5 Forráskód kinyerése	37
6.6 Táblázatok kinyerése	38
7. Továbbfejlesztési lehetőségek.....	40
7.1 Táblázatok kinyerése funkció kibővítése	40
7.2 További funkciók implementálása	40
8. Előnyök	41
9. Összefoglalás.....	42

BEVEZETÉS

1. Bevezetés

A feladat címe, amit választottam a következő: Kotlin web scraping fejlesztése. Ez annyit jelent, hogy Kotlin nyelven kell megvalósítani az adott programot, ami jelen esetben egy web scraping alkalmazás. Először a Kotlin nyelvről írnék. A Kotlin erősen típusos, objektumorientált programozási nyelv. A Szentpétervár közelében lévő Kotlin-szigetről nevezték el. 2011-ben hozták nyilvánosságra a nyelv létezését. A Java nyelvvel szokták összehasonlítani, azonban több nyelv is hatással volt rá a kifejlesztése során, mint például a Scala, Groovy, C# illetve még a Gosu is. Több platformra is lefordítható maga a kód, többek között Java Virtual Machine-re (JVM) is, ebből kifolyólag a Java-val teljes mértékben kompatibilis a Kotlin, továbbá könnyebben tanulható, mivel egyszerűbb a szintaxisa.

A web scraping-et tulajdonképpen arra használják, hogy különféle weboldalakról különböző adatokat nyernek/gyűjtenek ki, hogy ezeket rendszerezetten egy helyen tárolják, majd később fel tudják használni különböző tevékenységekhez. Nevezik még web harvesting illetve web data extraction-nek is. Ezt a folyamatot úgy kell elképzelni, mintha egy személy felmenne egy weboldalra, kézzel kimásolgatna különböző adatokat és lementené magának. A web scraping-gel rengeteg időt és sok fáradságot lehet megspórolni az automatizáció miatt. Léteznek olyan web scraping programok, amelyekhez szükséges külső interakció, vagyis a felhasználónak különböző paramétereket kell megadnia az alkalmazásnak, például, hogy milyen típusú adatokat keressen a rendszer az adott weboldalon. Azonban léteznek olyanok is, amelyek teljesen automatizáltan gyűjtik ki az adatokat és a háttérben futnak. Továbbá léteznek ingyenesen használhatóak, illetve olyanok is amelyekért fizetni kell.

Egy keveset arról is írnék, hogy miért választottam ezt a témát. A mobil programozás mindig is foglalkoztatott, ennek kapcsán elkezdett érdekelni a Java nyelv, mivel az Android alkalmazások jelentős részében ezt használták a fejlesztők. Pár éve a Google a Kotlin nyelv mellé állt, és az elsődlegesen támogatott nyelv az Android operációs rendszerek esetében így már nem a Java. Ez volt a fő oka annak,

BEVEZETÉS

hogyan ezt a témát választottam. A másik pedig, hogy ki szerettem volna próbálni magam egy tőlem elég távol álló területen, mint például a webbel kapcsolatos problémák megoldása. A szakdolgozatom végére azt szeretném elérni, hogy egy olyan programot készítsék, ami ingyenesen elérhető bárki számára és könnyen használható. Egy nagyon letisztult GUI-t képzelek el a program számára, hogy minél könnyebben bele tudjanak jönni a felhasználók a használatába. A legnagyobb probléma amivel találkoztam a különböző web scraping programok tanulmányozása során, az az volt, hogy nem voltak testreszabhatók, ezt úgy értem, hogy nagyrészt csak egy adott funkciót tudott. Tovább jelentős probléma, hogy az általam vizsgált rendszerek egy adott ID-vel rendelkező mezőnek az azonosítóját gyűjtötték ki és használták a későbbiekben, azonban egy dinamikusan változó rendszerben ez nem konstans, ezért nem is használható. Az én programomban úgy próbálom kiküszöbölni ezeket a problémákat, hogy testreszabható legyen a program. Ezt úgy értem, meg lehet majd adni, hogy milyen típusú, azonosítójú mezőket, illetve milyen reguláris kifejezésekre illeszkedő elemekre keressen rá a program és azok alapján gyűjtsön adatokat. Olyan funkciót is szeretnék a programba, amelynek segítségével exportálni lehet a kigyűjtött adatokat egy Excel fájlba, hogy azokat rendszerezve egy helyen lássa a felhasználó, hogy majd később könnyebben fel lehessen használni azokat.

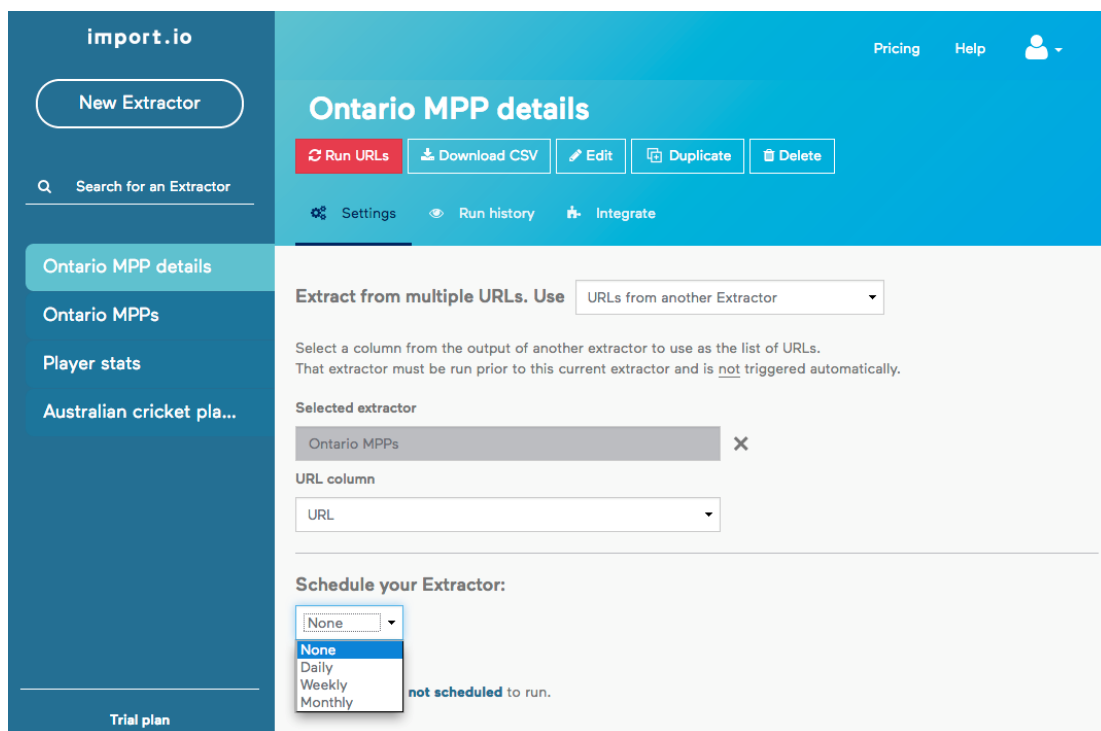
2. Kutatás

Említést tennék kutatásaimról, amelyeket szakdolgozatom elkészítése során végeztem. Ebben a részben említést teszek 4 különböző web scraping programról, amelyekről bővebben is írok. Ennek során megismerhetővé válnak az egyes alkalmazások előnyei, illetve hátrányai, hogy melyik miben jobb, illetve rosszabb a másikinál. Ennek a fejezetnek a végén kifejtem, hogy miben jobb szerintem az én web scraping alkalmazásom. A 4 program, amelyeket be fogok mutatni részletesebben a következők: Import.io, OutWit Hub, Beautiful Soup és a Scrapy.

2.1 Import.io

Az céget 2012-ben alapította egy három főből álló kis csapat, amelynek tagjai: Matthew Painter, Andrew Fogg és David White. Az Import.io egy Software as a Service (SaaS) Web Data Integration (WDI) platform, amely lehetővé teszi a felhasználók számára, hogy strukturáltan webes adatokat konvertáljanak egy strukturált formátumba, amelyet később a piacon az elemző szektor tud felhasználni. Legtöbbször üzleti, marketing, esetleg értékesítéssel foglalkozó alkalmazások szokták használni az Import.io által nyújtott szolgáltatásokat. Az Import.io egy vizuális felületet biztosít, amely megkönnyíti az adatok kinyerését, illetve átformálását. Az alábbi képen ez a felhasználói felület szerepel, ezáltal könnyebben elképzelhető, hogy valójában hogyan is néz ki egy ilyen GUI.

KUTATÁS



1. ábra – Import.io felhasználói felülete

Ezen a képen véleményem szerint egyből szembetűnő, hogy nem igazán egyszerű, letisztult az alkalmazás. Lehet, hogy funkcionalitást tekintve gazdag, viszont nem átlátható. A saját alkalmazásomban ez egy fontos szempont volt a fejlesztés során.

Szeretném megközelíteni az alkalmazást anyagi szempontból is. Ezeket az adatokat nem a hivatalos honlapon találtam, mivel ahhoz szükséges egy előzetes konzultáció, amely során felméri a megrendelő igényeit, azaz, hogy milyen adatokat és hány weblapról szeretné az információkat kinyerni. Ennek alapján ajánlanak egy csomagot a leendő felhasználónak. Viszont amit nem hivatalos forrásokból találtam, az a következő: egy kezdő számára a legkedvezőbb „csomag” egy próbaverzió. Ez korlátozott adatmennyiség kinyerésére teljesen megfelel, viszont egy nagyvállalati cég számára nem elegendő, ennek megfelelően azonban ingyenes. Ezen felül léteznek további csomagok is. Lehet választani 299\$-ért havi előfizetést, illetve éves tagságot is 1999\$-ért. Ez jelenlegi árfolyamon 91.000 Ft, illetve 609.000 Ft. Személyes véleményem szerint egy kisebb funkcionalitással bíró

KUTATÁS

alkalmazás többet ér, ha nincsen használati díja, mint egy magas költségű, amely több funkcióval rendelkezik.

2.2 OutWit Hub

A szoftver első verziója 2010-ben jelent meg. A jelenleg is frissnek számító verziója a 9-es, 2020 nyarán jelent meg. Az OutWit Hub egy Web Data Extraction szoftver, amely szintén online weboldalakról való adatok kinyerését szolgálja. Ez a program azonban képes linkek, dokumentumok, képek, kontaktok, ismétlődő szavak, RSS hírcsatornák, illetve adott kifejezések gyűjtésére is. Ezeket a kinyert adatokat formázott táblákba tárolja el, majd ezek tovább exportálhatók például Excel táblázatba, vagy akár adatbázisokba is. Ez az alkalmazás két féle formában érhető el bárki számára: létezik egy önálló alkalmazás, ami maga a program, illetve egy Mozilla Firefox bővítmény, amellyel ugyanazok a funkciók érhetők el, mintha letöltöttük volna a programot.

A program további tulajdonságai az alábbiak. Képes email címeket felismerni, automatikus lekérdezések, illetve URL címeket generálni, adott időközönként különböző feladatokat végrehajtani, és ami szerintem a legpozitívabb az OutWit Hub-bal kapcsolatban az az, hogy létezik egy „Custom Scrapers” szolgáltatás, ami annyit jelent, hogy mi összegyűjtjük, hogy milyen funkciókra van szükségünk, majd ezt egy konzultáció során ismertetjük a céggel. Ezek után elkészítik a számunkra optimális „Scraper”-t, majd ezt mi le tudjuk tesztelni, hogy valóban azokat a funkciókat tudja-e, amit mi kértünk. Amennyiben nem vagyunk elégedettek a validáció során, nem vagyunk kötelesek megtéríteni a szolgáltatásuk árát. Ez különben az általunk kért feladat komplexitásának függvényében változik. Kétszáz dollártól indul ez a szolgáltatás és akár több ezer dollárba is kerülhet.

Ennél a programnál is elérhető egy ingyenes verzió, amely a tulajdonos honlapjáról letölthető, azonban ennél a verziónál csak minimális funkciók érhetőek el. Létezik továbbá „Pro”, „Expert” és „Enterprise” előfizetés is. Ebben a sorrendben egyre gazdagabb funkcionalitást kapunk, egyre magasabb áron. A legdrágább előfizetés, amelyet a program kínál az „Enterprise” csomag, nem kevesebb, mint nettó 89000€, ami azt jelenti, hogy ehhez az összeghez még hozzá

KUTATÁS

kell adnunk a különféle adónemeket is. A végösszeg több mint hárommillió forint. Véleményem szerint ez már csak valóban az IT óriásoknak, illetve a piacvezető cégeknek érheti meg.

2.3 Beautiful Soup

Először itt is általános információkról írok. A BeautifulSoup egy package, amely Python nyelven készült. 2004-ben vált elérhetővé, Leonard Richardson volt az eredeti megalkotó. Azóta számos verzió jelent meg, a jelenleg is stabil a 4.9.1-es verzió, amely 2020. májusában jelent meg. XML, illetve HTML elemzésére használják, ennek a menete a következő: elemző fákat készít elemzett oldalaknak, ezáltal adatokat lehet kinyerni a HTML oldalról.

A technológiáról is említénék néhány szót. Ahhoz, hogy élvezhessük ennek a csomagnak az előnyeit, legalább minimálisan ismernünk kell a Python nyelvet, vagy képből kell lennünk a programozás világában. Nagy hátránynak tartom, hogy nincs külön grafikus felhasználói felület, amelyet a laikus felhasználók is könnyen, illetve hamar el tudnának sajátítani. A következő paranccsal telepíthető a BeautifulSoup legújabb verziója: „pip install beautifulsoup4”. Alább látható egy példakód a használatára:

```
from bs4 import BeautifulSoup
import requests
import os, os.path, csv

listingurl = "http://www.espn.com/college-sports/football/recruiting/databaseresults/_/sportid/24/class/2006/sc

response = requests.get(listingurl)
soup = BeautifulSoup(response.text, "html.parser")

listings = []
for rows in soup.find_all("tr"):
    if ("oddrow" in rows["class"]) or ("evenrow" in rows["class"]):
        name = rows.find("div", class_="name").a.get_text()
        hometown = rows.find_all("td")[1].get_text()
        school = hometown[hometown.find(",")+4:]
        city = hometown[:hometown.find(",")+4]
        position = rows.find_all("td")[2].get_text()
        grade = rows.find_all("td")[4].get_text()

        listings.append([name, school, city, position, grade])

with open("footballers.csv", 'a', encoding='utf-8') as toWrite:
    writer = csv.writer(toWrite)
    writer.writerows(listings)

print("ESPN College Football listings fetched.")
```

2. ábra –Beautiful Soup használata

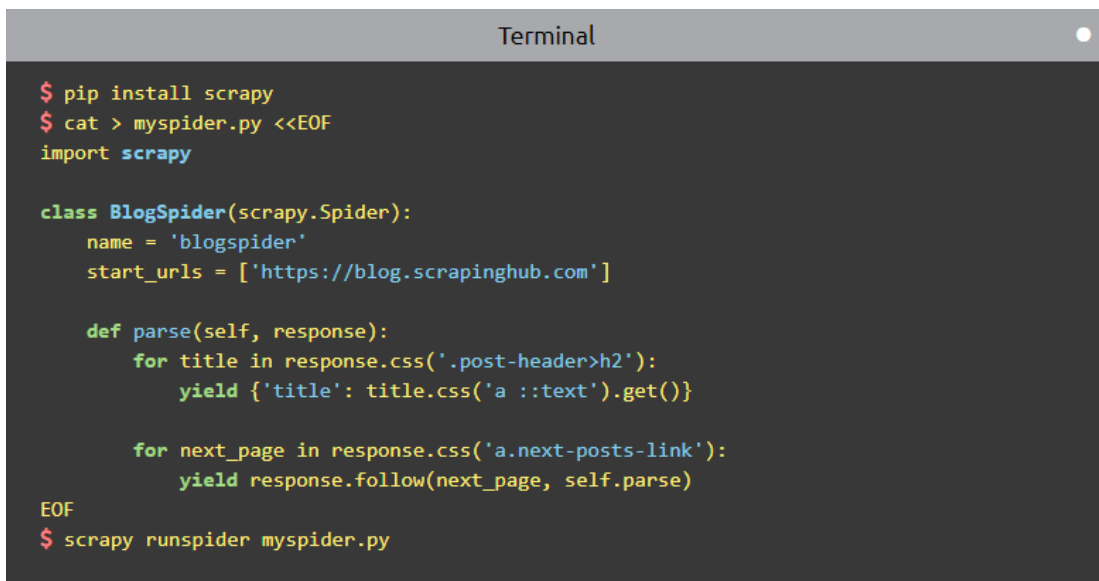
KUTATÁS

Azonban pont a grafikus felhasználói felület hiánya miatt, nem kell fizetünk semmit, a könyvtár használatáért. Ez nagyon fontos szempont lehet, ha egy, akár pár fős csapatról beszélünk felhasználó alatt, akiknek jelentősen kisebb az anyagi háttere, mint a nagyobb cégeknek. Amennyiben valaki nincs jelen az IT világában és szeretne valamilyen adatkinyerő módszert találni és nem riad meg a programozástól, annak jó választás lehet a Beautiful Soup. Aki picit jártasabb a kódolásban, de nem ismeri a Python nyelvet, annak sem kell megijednie, hiszen a Python nyelv elég könnyen elsajátítható.

2.4 Scrappy

A Scrappy egy nyílt forráskódú, ingyenes web scraping keretrendszer, amely Python nyelven készült. 2008. június 26-án jelent meg az első kiadása. Jelenleg a Scrapinghub Ltd., ami egy web scraping és szolgáltató cég, tartja karban a Scrappy-t, számos más közreműködő segítségével. Számos jól ismert cég használja ezt a szolgáltatást, mint például a Lyst, Parse.ly, Sayone Technologies, és a Science Po Medialab is. Az következő paranccsal telepíthető: „pip install scrapy”. A használata nagyon sokban hasonlít a Beautiful Soup-hoz – ebben az esetben sem tartozik grafikus felhasználói felület, amely segítené a használatot. Előnyeit, illetve hátrányait szintén a Beautiful Soup-hoz tudom hasonlítani, amelyeket kifejtettem az előző fejezetben. Viszonylag széles körben elterjedt a használata, például GitHub-on ezernyolcszázan figyelik, illetve Twitteren ötezer-egyszáz követője van a hivatalos oldalnak. Az alábbi kódrészlet egy használati példát mutat be.

KUTATÁS

A terminal window titled "Terminal" with a dark background and light-colored text. It shows the commands to install Scrapy, create a new spider file, and run it. The code for the spider is written in Python, defining a class BlogSpider that inherits from scrapy.Spider. It sets the name to 'blogspider' and the start_urls to a list containing 'https://blog.scrapinghub.com'. The parse method uses CSS selectors to extract titles and follow next-page links.

```
$ pip install scrapy
$ cat > myspider.py <<EOF
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://blog.scrapinghub.com']

    def parse(self, response):
        for title in response.css('.post-header>h2'):
            yield {'title': title.css('a ::text').get()}

        for next_page in response.css('a.next-posts-link'):
            yield response.follow(next_page, self.parse)
EOF
$ scrapy runspider myspider.py
```

3. ábra – Scrapy használata

TERVEZÉS

3. Tervezés

Alkalmazásom fejlesztése során sok szempontot figyelembe vettem, illetve számos személytől kértem tanácsot, mind a funkcionalitással kapcsolatban, mind a grafikus felhasználói felületet tekintve. Voltak közöttük szakmabeliek és laikusak egyaránt. Ezen személyek tanácsai nagymértékben befolyásolták döntéseimet a tervezés során. Kaptam negatívumokat és pozitívumokat is egyaránt. A negatívumokat építő kritikaként fogtam fel, megfogadtam őket és tanultam belőlük, ez egy szélesebb rálátást adott az egész projektre és nagyban megkönnyítette a feladatomat minden egyes fázisában. Végül úgy döntöttem, hogy a főbb szempontok, amelyeket figyelembe fogok venni a fejlesztés során a következők lesznek:

- Letisztult, könnyen kezelhető grafikus felhasználói felület. (I.)
- Díjtalan felhasználás bárki számára. (II.)
- Alacsony funkcionalitás a minimális komplexitásra törekedve. (III.)

Röviden kifejteném mind a 3 szempontot, hogy mit is értek alattuk és miért választottam ezeket a legfőbb pilléreknek.

- I. Számtalan alkalommal találkoztam olyan alkalmazással, amelynek nagyon komplex és bonyolult volt a felhasználói felülete, ezáltal több órába, esetleg napokba tellett a program megismerése. Szerintem ez nagyban befolyásolja a felhasználókat abban, hogy hamar megunják a program használatát és keressenek egy olyat, amelynek a funkcionalitása hasonló, de könnyebben megtanulható annak használata. Az én alkalmazásom egyszerű, jól átlátható grafikus felhasználói felülettel fog rendelkezni, pont azért, hogy ezt a problémát kiküszöbölje.
- II. Véleményem szerint a legbefolyásolóbb tényező bármilyen alkalmazásnál annak anyagi vonzata. Szerintem rajtam kívül nagyon sokan gondolják így, hogy inkább használnak egy egyszerűbb alkalmazást ingyen, mint egy sokkal magasabb funkcionalitással bíró programot, amelynek rendkívül magas használati díja van. A legtöbb

TERVEZÉS

program esetén nem elég az egyszeri megtérítés, sok olyan alkalmazást ismerek amelyekért havonta, vagy esetleg évente használati díjat számolnak fel. Az én programomban ezzel nem lesz probléma, hiszen ingyenesen elérhető lesz bárki számára.

- III. Végül a harmadik szempontom az alacsony komplexitás. Ahogy a mondás is tartja: „a kevesebb néha több”, ezt úgy kell érteni ebben az esetben, hogy nem feltétlen a magas funkcionalitású, összetett alkalmazások a jobbak, hiszen ezekben rengetek hibalehetőség merülhet fel. Egy bonyolult rendszert nagyon sok idő hibamentessé tenni, a teszteléséről nem is beszélve. Amennyiben egy nagyon komplex rendszert vizsgálunk, abban az esetben nem is lehetséges a kimerítő tesztelés. Az én programomba inkább minimális funkcionalitást képzelek el, viszont azokat nagyon robusztus módon szeretném implementálni. Szélsőséges esetekben is szeretném tesztelni, ezzel kiküszöbölve az esetlegesen előjövő hibákat a végfelhasználónál.

Sok mérlegelés után arra jutottam, hogy speciális web scraping alkalmazást fogok fejleszteni, nem pedig általános felhasználhatóságút. Ami miatt erre az elhatározásra jutottam az az volt, hogy egy általános felhasználhatóságú program nagyon sokszínű és elképesztően komplex a felépítése. Rengeteg dolgot kellene figyelembe venni az implementálás során, amihez nem lett volna elegendő tudásom, illetve elegendő időm sem. Végül arra a döntésre jutottam, hogy a következő dolgokhoz szeretnék web scraping alkalmazást fejleszteni: email címek, képek, linkek, táblázatok és végül, de nem utolsó sorban az adott weboldal forráskódjának a kinyerésére. Legnagyobb kihívásnak a táblázatok adatainak kinyerését tartottam. Nagyon sok olyan esettel lehet találkozni amikor, egy adott weboldalon rengeteg információ van felsorolva táblázat formájában. Esetenként ezek a táblázatok óriási méretűek és sok időbe telik az adatokat kimásolgatni belőlük. Az alkalmazásom egyik funkciója erre a problémára fog nyújtani megoldást, hiszen az URL megadása, illetve egy gomb megnyomása után az összes táblázat összes adata az adott oldalon kinyerésre kerül. Ezek után az adatok elérhetőek lesznek egy szöveges dokumentum formájában, ahonnan könnyen felhasználhatók lesznek az adatok.

4. Fejlesztés

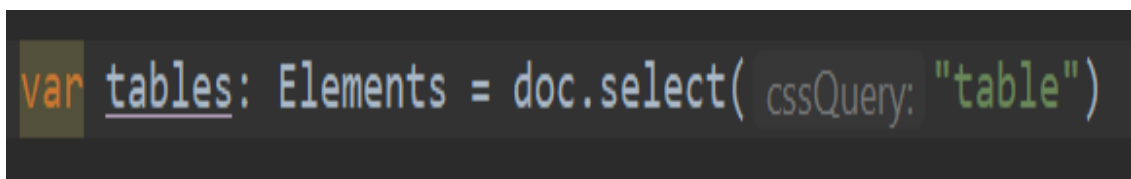
Az alkalmazás fejlesztésének megkezdése előtt nagyok sokat gondolkoztam azon, hogyan is fogjak neki az implementálásnak. Sok tanácsot kértem általam sokra tartott, nagy szakmai tudással rendelkező szaktársaimtól, illetve más egyetemekre járó hallgatóktól is. Továbbá kikértem véleményét a Kollégáimnak és nagyon sok segítséget kaptam témavezetőmtől is, hogy milyen úton érdemes elindulni a fejlesztéssel kapcsolatban. Rengeteget olvastam különféle fórumokon, hogy milyen technológiák érhetők el, amelyeket fel tudnék használni az alkalmazásomhoz. Sokféle módon próbáltam megközelíteni az előttem álló akadályokat. Legnehezebb feladatom a fejlesztés során az az volt, hogy a táblázatokat milyen módon fogja felismerni a program, amely alapján kigyűjthetők azok tartalmai. Azonban ehhez a problémához gondtalanul tudtam használni a JSoup könyvtárat, illetve annak minden előnyét. A továbbiakban bővebben írok erről a könyvtárról.

4.1 JSoup

A JSoup egy nyílt forráskódú Java könyvtár, amelyet HTML dokumentumokban tárolt adatok elemzésére, kinyerésére és kezelésére terveztek. 2009-ben írta egy Jonathan Hedley nevű szoftverfejlesztési ügyvezető. A fejlesztés célja elméletileg az volt, hogy "megbirkózzon a vadonban lévő HTML összes fajtájával". Több jelenlegi projektben is használják a könyvtárat, példának okáért, a Google OpenRefine adatkezelő eszköze is. Használata igazán egyszerű azok számára, akiknek nem idegen az informatika világa. A JSoup eredeti honlapjáról ingyenesen letölthető a könyvtár legújabb verziója, ami jelenleg az 1.13.1-es verzió: <https://jsoup.org/>. A .jar kiterjesztésű állomány letöltése után az adott projektbe könnyedén beimportálható. IntelliJ-ben a következő lépéseket kell ehhez megtenni: kívánt projekt megnyitása, amelyben szeretnénk használni a JSoup-ot. Az állapotsorból ki kell választanunk a File menüpontot, majd a baloldali panelen a Modules menüt. Ezek után a Dependencies fülön kattintsunk a "+" ikonra. Végül pedig az adott lehetőség közül válasszuk az elsőt, aminek a neve: 1 JARs or directories, itt nincs más dolgunk, mint a korábban letöltött .jar állományt

FEJLESZTÉS

betallózzuk. Ezzel a néhány egyszerű lépéssel már használhatjuk is a JSoup minden előnyét. A lépések leírása során az angol kifejezéseket használtam, mivel nekem az IntelliJ fejlesztőkörnyezet angolra van állítva és ezeket a kulcsszavakat ismerem. A mai felgyorsult és előrehaladott világban nem szabad, hogy gondot okozzon senkinek ennek a pár kifejezésnek a lefordítása magyarra, aki esetleg nem ismeretes az informatikai szaknyelvben. Az eredeti honlapon számos dolog található maga a könyvtárán kívül is. A News lapon különféle hírek olvashatók a korábban kiadott verziókkal kapcsolatban, például, hogy milyen fejlesztések történtek és milyen hibákat küszöböltek ki az előző verzióhoz képest. A Discussion fülön egy Stack Overflow hivatkozáson keresztül egy oldalra jutunk, ahol az összes kérdés megtalálható amelyben használták a jsoup címkét. Ez nagyon hasznos lehet bárki számára, aki valami miatt elakadt a könyvtár használata során, hiszen itt tapasztalt és olyan felhasználók válaszait olvashatjuk akik már használták korábban. Legfőképp azon felhasználók számára hasznos ez az oldal, akik még sohasem használták a JSoup-ot és nem tudják hogyan álljanak neki a használatának, többek között számomra is nagyon hasznos volt. Először itt olvastam különféle bejegyzéseket a könyvtárról. Nagyon sok hasznos komment volt a felhasználással kapcsolatban, hogy mire és pontosan hogyan használható. Tulajdonképpen bármilyen kérdésre kerestem az oldalon, találtam rá választ, megoldást és ekkor döntöttem el, hogy én is a JSoup könyvtárat fogom használni a fejlesztésem során, hiszen nagyban megkönnyítheti a dolgom. Elkezdtem használni, majd elég hamar kiderült, hogy jó döntés volt, hiszen amilyen akadályokkal szembe kerültem korábban, most könnyedén találtam rájuk megoldást. Úgy működik, hogy megadunk egy URL címet, majd a JSoup átkonvertálja HTML tartalommal. Ezután egy változóban megadjuk azt a HTML címkét a relációjelek nélkül, ami közre zárja azokat az adatokat, amelyre szükségünk van. Ekkor az adott változóba az összes olyan adatot letárolja, amely a megadott nyitó és záró címke között szerepel. Az alábbi képen látható, hogyan használtam a JSoup-ot a táblázatok kigyűjtésére:

A screenshot of a code editor showing a line of Java code. The code is: `var tables: Elements = doc.select(cssQuery: "table")`. The word 'var' is in orange, 'tables:' is in blue, 'Elements' is in blue, 'doc.select(' is in blue, 'cssQuery:' is in green, and '"table"' is in green. The code is set against a dark background.

4. ábra – Példa a Jsoup használatára

FEJLESZTÉS

A "tables" változó típusa Elements, ami lényegében egy lista, amely szabadon indexelhető. A fenti példát alapul véve, minden egyes eleme a "tables" változónak, egy-egy nyitó és záró table címke között lévő összes adat, ami jelen esetben egy konkrét táblázat az adott weboldalon.

4.2 Felépítés

Alkalmazásom alapját a már korábban is említett JSoup, illetve a Java Swing grafikus elemei adták. Először maga a kinézetéről írnék bővebben. Az alkalmazás mérete fix 1400 x 250 pixel méretű, azonban ez az indítás után szabadon változtatható. Találhatók a felületen JLabel, JCheckBox, JTextField és mindeössze egy darab JButton elem is. Maga az elrendezésnél GridBagLayout-ot használtam, ami nagyban elősegítette a táblázatszerű elrendezését az elemeknek. Meg kell adnunk az összes adatot amire a programnak szüksége van (bővebben a felhasználás fejezetben lesz kifejtve), ezek után pedig rá kell kattintanunk a Scrape! gombra. Amikor megnyomtuk a gombot, akkor kezd el igazán dolgozni a program. Először is egy, a JSoup által biztosított Connection osztályból hoztam létre egy változót a felhasználó által előre megadott URL címből. Nagyon sok tulajdonságot állítottam be a változó létrehozásakor, hogy a lehető legtöbb hibát kiküszöböljem vele. A legfontosabbak a következők voltak:

4.3 ignoreContentType

Igaz-ra van állítva ez a tulajdonság, és ebben az esetben a válasz elemzésekor figyelmen kívül hagyja a dokumentum Tartalom-típusát. Ezzel tulajdonképpen bármilyen típusú dokumentumhoz tudunk csatlakozni.

4.4 userAgent

Ezzel a tulajdonsággal meg lehet adni, hogy milyen típusú böngészőként kezelje a kérést a szerver. Ez azért fontos, mert előfordulhat olyan eset is, amikor két különböző böngészőben ugyanaz a weboldal kétféleképpen jelenik meg, illetve az is előfordulhat, hogy az adott weblap egyáltalán nem jelenik meg, mivel az adott

FEJLESZTÉS

böngésző nem támogatott. Ezt a tulajdonságot én Mozilla Firefox webböngészőre állítottam, hiszen ez eléggé elterjedt az egész világon és a legtöbb weboldal támogatja is.

4.5 referrer

Ezzel a tulajdonsággal azt lehet megadni, hogy a kérés melyik oldalról érkezett. Ezt a tulajdonságot a Google hivatalos honlapjára állítottam, mivel ez a weboldal elég széles körben ismert és nagyon kicsi rá az esélye, hogy valamelyik oldal ne engedélyezné ennek az oldalnak a hozzáférést.

4.6 timeout

Ez a tulajdonság arra használható, hogy megadjuk a timeout értéket és amennyiben a paraméterben megadott értéknél tovább tart az oldal lekérése, akkor megszakad a lekérés és nem a teljes html oldalt kapjuk meg. A paraméterben megadott érték milliszekundumban értendő. Ezt a tulajdonságot 0 paraméterrel állítottam be, ami azt jelenti, hogy nincsen timeout érték, vagyis bármilyen nagy lehet az adott oldal, mindenképpen megkapjuk a teljes forrást. Személyes tapasztalatom miatt adtam hozzá, mivel volt egy olyan eset az alkalmazás tesztelése során, hogy próbáltam kigyűjteni az összes táblázatot az oldalról, viszont néhány sor mindig lemaradt a végéről és nem értettem miért. Végül kiderült, hogy túl nagy volt az oldal és timeout hiba miatt nem jutottam hozzá az teljes méretéhez.

4.7 maxBodySize

Ezzel a tulajdonsággal megadható, hogy maximum milyen nagy weboldalhoz tudjunk hozzáférni. Alapbeállításként ez az érték egy megabájt. Ugyancsak az alkalmazás tesztelése kapcsán jött elő a probléma, hogy nem a teljes html forrást sikerült kinyerni. Éppen ezért ezt a tulajdonságot 0 paraméterrel állítottam be, ebben az esetben nincsen limitálva a méret és bármekkora oldalhoz hozzá lehet férni.

FEJLESZTÉS

4.8 execute

Ez a tulajdonság az utolsó, hiszen ezzel tudjuk végrehajtani az általunk indított kérést a fentebb beállított tulajdonságokkal együtt.

Egy másik, szöveges változóban eltárolom a felhasználó által megadott elérési útvonalat, ám előtte a ”\” jelet ki kell cserélnem ”\\” jelre, hiszen így tudja helyesen értelmezni a kódot. Ez egy úgynevezett feloldójel, ami egy karakterláncban azt jelöli, hogy a következő karaktert másképp kell értelmezni mint alapesetben. Az utolsó közös programrész, amelyet az összes funkció használ az alkalmazásban az az, hogy egy Document típusú változóba át kell adni a korábban létrehozott html kapcsolatot, hogy megfelelő típusban legyen eltárolva és később felhasználható legyen.

A következőkben a programkódszinten való megvalósításról írnék részletesebben, leírást adni külön-külön minden egyes funkciót tekintve, hogy melyiknél milyen eljárást használtam, illetve milyen nehézségekkel kerültem szembe a fejlesztés során.

4.9 Email címek kinyerése

Az első funkció, ami implementálásra került az az email címek kinyerése volt. Sokat keresgéltem az interneten, hogy hogyan is lehetne felismerni az emailcímeket egy egyszerű szövegben. Nem igazán találtam teljesen megfelelő megoldást jó darabig, míg egyszer eszembe jutott, hogy talán mintaillesztéssel megvalósítható lehet. Sokat olvastam a reguláris kifejezésekről, hogyan is működnek pontosan a gyakorlatban. Végül kiderült, hogy nem is olyan bonyolult a használata amilyennek tűnt először. Sokféle mintát próbáltam ki nagyon sok weboldalon, míg végül megtaláltam az ideális mintát. Ez lett az a minta: ”\\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,4}\\b”. Ezt a mintát felhasználva elmentettem egy listába az összes találatot, amit a reguláris kifejezéssel megtalált a program. Azonban ez nem adott teljesen jó megoldást, mivel nagyon nehéz olyan mintát létrehozni, ami kimondottan csak az email címeket találja meg, hiszen nagyon sok kikötés létezik az email címek szintaxisára vonatkozóan, hogy mi is számít ”valódi” címnek. Éppen ezért létrehoztam egy másik listát, amelyben eltároltam a már kiszűrt email

FEJLESZTÉS

címeket. Ezt egy for ciklussal oldottam meg, ami végigmegy a nyers listán és többféle szempont szerint vizsgálja a címeket. Először is megvizsgálja, hogy az adott cím az előre megadott végződéssel rendelkezik-e vagy sem. Az is figyelembe van véve, hogy a felhasználó melyik végződésű email címeket szeretné kigyűjteni és a program csak azokat adja hozzá a szűrt listához. Még három feltételt ellenőriz. Az egyik legalapvetőbb, hogy még nem létezik az éppen vizsgált cím a szűrt listában, ezzel elkerülve a redundanciát. Továbbá azt, hogy az éppen vizsgált cím nem tartalmaz-e ”.” karaktert, hiszen ebben az esetben nem valós az email cím. Végül pedig azt is megvizsgálja, hogy a ”@” jel előtt maximum hatvannégy karakter hosszúságú szöveg állhat, hiszen ha ennél hosszabb, akkor érvénytelen az email cím. Amint ez a rész lefutott, az összes általunk kívánt cím egy változóban elérhető. Ezek után nincs más hátra, mint szöveges állományba kiírni az adatokat. Itt egy feltételhez van kötve a kiírás. Mégpedig ahhoz, hogy a szűrt listánk, tartalmaz-e bármilyen szöveget. Hiszen, ha üres, vagy azért mert nem található egyetlen email cím sem az általunk megadott oldalon, vagy azért mert a felhasználó nem választotta ki az email címekre vonatkozó lehetőségeket, vagyis nincs szüksége rájuk, ezekben az esetekben nem kell létrehoznunk fölöslegesen üres állományokat. Viszont, ha ez a feltétel teljesül, és tartalmaz akár egy darab címet is a változó, ebben az esetben létrehoz a program egy Email.txt nevű állományt arra a helyre, amelyet a felhasználó a programban megadott. Az alkalmazás ebbe a fájlba minden egyes email címet külön sorba helyez el, ezáltal jól kezelhetők lesznek, esetleges további folyamatok számára. Az egész kódrész, ami ehhez a funkcióhoz tartozik, csak abban az esetben fut le, ha a felhasználó kiválasztotta bármelyik opciót az email címekhez tartozó opciók közül, hiszen, ha nincs szüksége egyetlen email címre sem, abban az esetben a fent említett kódrész le sem fut, ezzel is növelve a gyors futási időt, illetve a felesleges erőforrás használatot is elkerüli.

4.10 Képek kimentése

Ennek a funkciónak a fejlesztése nem okozott hatalmas megpróbáltatásokat. Könnyedén tudtam alkalmazni a JSoup nyújtotta előnyöket. Az tagek forrásaira keres rá az alkalmazás, majd ezeket egy lista változóba le is menti. Ezután létrehoztam egy másik lista változót ami a szűrt elemeket fogja tárolni. A

FEJLESZTÉS

szűrést ebben az esetben egy for ciklussal oldottam meg, ami végighalad a szüretlen listán. A ciklusban különböző feltételeket alkalmaztam a szűrésre. Az első feltétel azt vizsgálja, hogy a lista eleme "h" karakterrel kezdődik-e, ez azért fontos, mert a legtöbb esetben ezeken a weboldalakon a képek egy beágyazott linkként találhatók meg. Az is vizsgálva van, hogy a filterezett lista már tartalmazza-e az éppen vizsgált elemet. Ezekben a beágyazott linkekben majdnem minden esetben megtalálható a kép kiterjesztése egy karaktersorozat formájában. Éppen ezért azt is vizsgálja a program, hogy a vizsgált elem tartalmazza-e a "jpg", "jpeg", "png" vagy esetleg a "gif" karaktersorozatot, továbbá azt is ellenőrzésre kerül, hogy a felhasználó egyáltalán szeretné-e bármelyik képformátumú képeket kigyűjteni, amennyiben nem, az alkalmazás ezen része le sem fut. Amint a fent említett rész lefutott és találtunk legalább egyetlen linket, a következő lépés a linkek átkonvertálása, majd kiírása. Az alkalmazás ezen részét is egy for ciklussal oldottam meg. Ebben a ciklusban a már szűrt linkeken megy végig az iterátor. Első lépésként minden egyes link átkonvertálásra kerül egy URL típusú változóba, ezáltal később könnyebben lesznek kezelhetők a kiírás során. When kifejezéssel oldottam meg a linkek beazonosítását megfelelő kategóriába. A when kifejezés nagyon hasonlít a C-hez hasonló nyelvekben használatos switch kifejezéshez. Ebben a részben kerülnek a képek szortírozásra az alapján, hogy melyiknek mi a kiterjesztése. Azonos kiterjesztésű képek azonos nevű mappába kerülnek növekvő sorszámmal ellátva. Minden egyes típushoz tartozik egy-egy változó, amelyekben számon vannak tartva a képek darabszáma, ezek vannak segítségül hívva a kép nevének megadásakor. Ezzel a lépéssel ennek a funkció implementálásának a végére is értünk. Ez az egész kódrészlet csak abban az esetben fut le, ha a felhasználó kijelölte valamelyik képformátumot az alkalmazás használata során, ezzel is rövidítve a futási időt, illetve csökkentve az erőforráshasználatot.

4.11 Linkek kinyerése

Ennél a funkciónál is próbálkoztam reguláris kifejezések használatával, viszont ebben az esetben nem jártam sikerrel. Ezért a JSoup könyvtár előnyeit hívtam segítségül. A forráskódból először kigyűjtöttem azokat a szövegrészeket, amelyekre mutat valamilyen hivatkozás. Több oldalon való tesztelés során kiderült,

FEJLESZTÉS

hogy itt is akadnak hibák és további feltételek vizsgálatára van szükség a megfelelő működéshez. Éppen ezért itt is alkalmaztam egyfajta szűrést amelyre pontosabb megoldást ad az alkalmazás. A következő dolgokat figyeli a program. Először is azt, hogy az éppen vizsgált linket tartalmazza-e már a szűrt lista, hiszen kétszer ugyanazt a linket felesleges kigyűjteni. Vizsgálja még továbbá a link teljes hosszát, itt 4 karakter hosszúságot állítottam be, aminél hosszabbnak kell lennie, mivel elég ritka az ennél rövidebb link. Végül pedig azt vizsgálja meg az alkalmazás, hogy az adott link "http" karaktersorozattal kezdődik-e. Amennyiben az összes feltételnek megfelel egy link, abban az esetben hozzáadódik a szűrt listához. Ezek a lépések után már csak egy van hátra, mégpedig a fájlba való írás. Itt is egy for ciklus fut végig a szűrt listán, és a felhasználó által előre definiált helyre egy Links.txt fájlba írja ki az összes linket. Természetesen minden linket külön sorba, a jobb átláthatóság kedvéért. Itt is használtam a fent említett kódrészletre egy feltételt, ami azt vizsgálja, hogy az alkalmazásban ki lett-e választva a linkekhez kapcsolódó opció, amennyiben nem, abban az esetben egyik kódrészlet sem fut le, hiszen feleslegessé válik. Továbbá az üres szöveges állomány sem jön létre.

4.12 Forráskód kinyerése

Ezt a funkciót volt talán az egyik legegyszerűbb megvalósítani a fejlesztés során. Egy feltételben történik annak vizsgálata, hogy a felhasználónak szüksége van-e az adott weboldal forráskódjára. Amennyiben kiválasztotta ezt az opciót, a már korábban létrehozott Document típusú változót átkonvertálja sima szöveg típusúvá majd ezt a felhasználó által megadott mappába egy Source.txt nevű állományba ki is írja. Az alkalmazás ezen funkciója hasznos lehet azon személyek számára, akik kíváncsiak, hogy egy-egy általuk ismert weboldal hogyan és milyen elemekből épül fel, illetve milyen technológiákkal készült. Esetleg példaként is szolgálhat egy komolyabb oldalnak a forráskódja, melyből később ötletet lehet meríteni egy saját oldal létrehozásakor.

4.13 Táblázatok kinyerése

Ennek a funkciónak a kivitelezése volt számomra a legnehezebb. Sokkal több eshetőséget kellett számításba venni, mint a többi funkció fejlesztés során. A

FEJLESZTÉS

táblázatoknál ugyanis azt is le kell kezelni, hogy az adott oldalról hány darab táblázatot, illetve ezekből a táblázatokból hány darab sorra és oszlopra van szüksége a felhasználónak. Azt is kezelnem kellett valahogy, ha a felhasználó nem megfelelő karaktert ad meg, vagyis nem számot, akkor mi történjen az alkalmazásban. Továbbá azokat az eseteket is figyelembe kellett vennem, amikor a felhasználó nagyobb vagy esetleg negatív számot ad meg a valós méretek paramétereiként az alkalmazásban. Pár nap próbálkozás után rátaláltam a megfelelő megoldásra. Első körben vizsgálatra kerül a három beviteli mező, azaz a táblázatok számára, sorok, illetve oszlopok számára vonatkozó adatok. Amennyiben az összes mezőbe pozitív numerikus értéket ad meg a felhasználó, akkor futnak le a következő kódrészletek. Ellenkező esetben a Scrape! gomb megnyomása után ezen mezők alaphelyzetbe kerülnek, azaz az értékük "0" karaktert vesznek fel. Viszont ha minden adat megfelelő, akkor folytatódhat a folyamat és a táblázatok kezelése. Létrehoztam egy változót a táblázatok számának nyilvántartására, illetve a három különböző változóba el van tárolva a felhasználó által megadott három érték. A JSoup segítségével kigyűjtöttem az összes adatot a táblázat nyitó és záró tag-jei között. Ennek a változónak a típusa Elements, amit szintén a JSoup biztosít számunkra. Ezt a típust használva egy ciklus során egy konkrét táblázat adatain tudunk végigmenni. Pont emiatt én is így jártam el, egy for ciklussal végigmentem az összes táblázaton amely a felhasználó által megadott weboldalon található. Ezen a cikluson belül egy másik változót is létrehoztam a táblázat sorainak a számolására. Egy másik változóban letároltam az összes <tr> tag-ek között lévő adatokat, aminek szintén Elements a típusa. Ezután ezen a változón is végigmegy egy for ciklus. A cikluson belül az oszlopok számolására szintén létrehoztam egy Elements változót. Ezeken a ciklusokon belül hajtódnak végre a táblázatok szöveges állományba való írása a felhasználó által előre megadott paraméterek alapján. A táblázat elrendezése pont olyan formában jelenik meg az állományban, mint a weblapon. Továbbá egy feltétel vizsgálja azt is, hogy ha a felhasználó a táblázatok számát a valóban létező táblázatok számánál nagyobbra állítja, abban az esetben az összes táblázat feldolgozásra, illetve kigyűjtésre kerül.

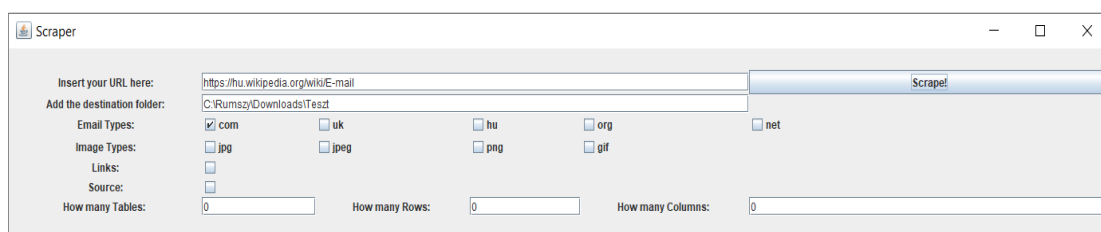
TESZTELÉS

5. Tesztelés

Amint már korábban is kifejtettem, a tesztelés nagyon fontos egy alkalmazás fejlesztése során. A legfőbb okát egy hétköznapi példán keresztül nagyon könnyen beláthatja bárki, hiszen senki nem szeret egy olyan alkalmazást használni, amely csak "félig" vagy még rosszabb esetben egyáltalán nem működik. Éppen ezért fordítottam nagy hangsúlyt a tesztelésre, hogy minden funkció a lehető legmegbízhatóbban és a leghatékonyabban működjön. Az összes funkciót amit az alkalmazás tud, teszteltem külön-külön és a legtöbb kombinációban is, hogy minimálisra csökkenjen a nem várt hibák száma. Mindegyik funkcióhoz ki szeretném fejteni a tesztelési folyamatot, hogy pontosan hogyan is zajlott és miket is teszteltem, illetve milyen weboldalakon.

5.1 Email címek kinyerése

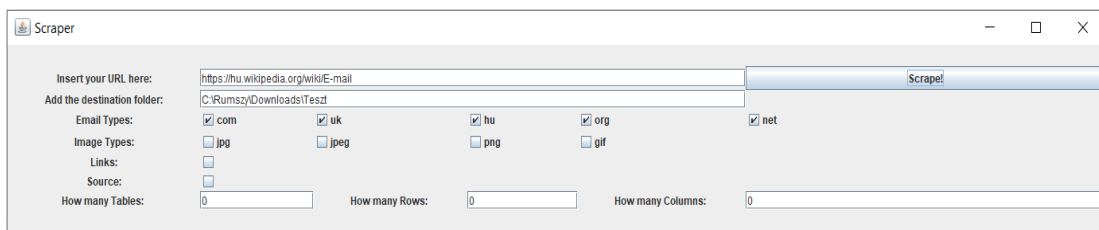
Első körben azt teszteltem, hogy egyáltalán működik-e az alkalmazásom, illetve az elvárt feltételeknek megfelelően. A következő weboldallal kezdtem a tesztelést: <https://hu.wikipedia.org/wiki/E-mail>. Semmilyen más funkciót nem kapcsoltam be, mint az email címek közül a .com végződésűeket, hogy minél rövidebb legyen a futási idő, ezáltal felgyorsítva a tesztelést. Az alábbi képen látható is a konfiguráció.



5. ábra – Egy domain kiválasztva az alkalmazásban

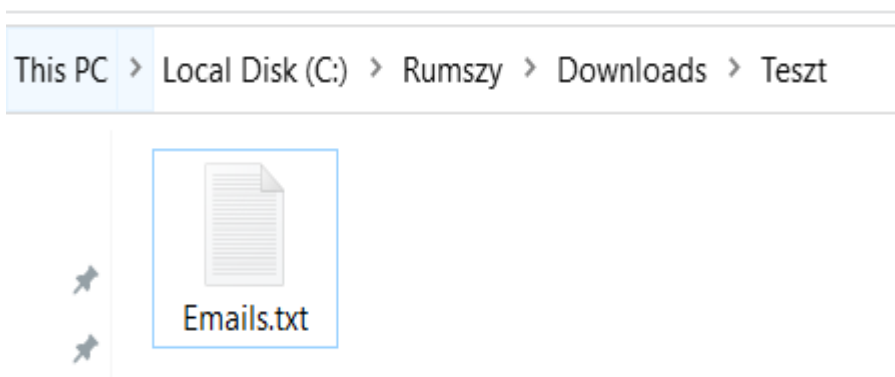
A program sikeresen befejezte a futását nagyon rövid időn belül. Sajnos egyetlen egy email címet sem talált, mivel a weboldal nem tartalmazott .com végződésű címet. Ezen teszt után átállítottam a konfigurációt oly módon, hogy az összes email címet találja meg és mentse ki egy szöveges állományba.

TESZTELÉS



6. ábra – Összes domain kiválasztva az alkalmazásban

Ezen konfiguráció lefuttatása után az alkalmazás sikeresen létrehozott egy Emails.txt nevű fájlt a megfelelő helyre, amelyet korábban megadtam az alkalmazásban.



7. ábra – Emails.txt állomány

Megnyitottam az állományt és csodálkozva láttam, hogy csak kettő darab email címet tartalmaz, még hozzá a következőket:

frau.mustermann@example.org

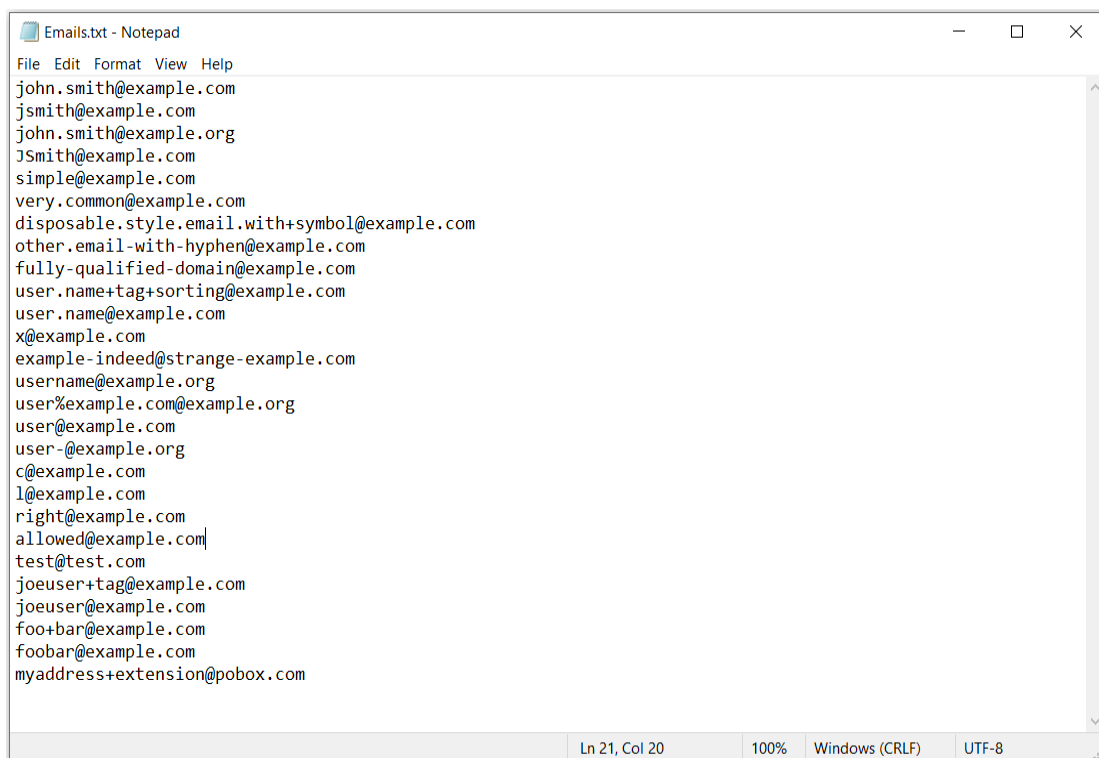
jkovacs@cegneve.hu.

Aggodalmam hamar csökkenni kezdett, amikor ellenőriztem a honlapot, hogy valóban csak kettő darab email címet tartalmaz. Ezt úgy hajtottam végre, hogy a weboldalon a Ctrl + f billentyűkombináció lenyomása után rákerestem a ”@” karakterre, hiszen minden valós email címnek tartalmaznia kell ezt a karaktert. Ez azonban három darab találatot eredményezett, viszont az egyik találat egy önálló ”@” karakter volt egy email címtől eltérő szöveggörnyezetben. Ezzel sikeresen teszteltem azt is, hogy nem azokat a karakterláncokat keresi és gyűjti ki az alkalmazásom amelyek tartalmazzák ezt a bizonyos karaktert. Ezen teszt lefutása után már birtokába jutottam az összes címnek amit az oldal tartalmaz. Leteszteltem

TESZTELÉS

azt is, hogy csak a .hu végződést állítottam be az alkalmazásba. Sikeresen vette az akadályt, hiszen a lefutás után csak a jkovacs@cegneve.hu címet tartalmazta a létrehozott file.

A következő teszthez egy olyan oldalra volt szükségem ahol sok email cím fordul elő. Rövidebb idő után rá is találtam a megfelelő oldalra, ami a következő volt: https://en.wikipedia.org/wiki/Email_address. Ez számos címet tartalmazott, kíváncsian vártam, hogy az alkalmazásom hogyan teljesít. Megadtam az alkalmazásnak a weblapot, illetve a korábban használt elérési útvonalat. A lefutás után ezt az eredményt kaptam. Jól látható, hogy minden cím külön sorban található, amely nagyban megkönnyíti az átláthatóságot.



```
File Edit Format View Help
john.smith@example.com
jsmith@example.com
john.smith@example.org
JSmith@example.com
simple@example.com
very.common@example.com
disposable.style.email.with+symbol@example.com
other.email-with-hyphen@example.com
fully-qualified-domain@example.com
user.name+tag+sorting@example.com
user.name@example.com
x@example.com
example-indeed@strange-example.com
username@example.org
user%example.com@example.org
user@example.com
user-@example.org
c@example.com
l@example.com
right@example.com
allowed@example.com
test@test.com
joeuser+tag@example.com
joeuser@example.com
foo+bar@example.com
foobar@example.com
myaddress+extension@pobox.com
```

8. ábra – Emails.txt állomány tartalma

Rákerestem az összes ”@” jelre a weboldalon és számos olyan címet kiszűrt az alkalmazás, amely nem valós email cím. Az egyik legérdekesebb talán az volt, hogy kiszűrte ezt a címet: 123456789012345678901234567890123456789012345678901234+x @example.com. Ez azért nem valós, mert a lokális rész hossza, azaz ami a ”@” karakter előtt szerepel, nem lehet nagyobb 64 karakternél. Továbbá azt a címet sem

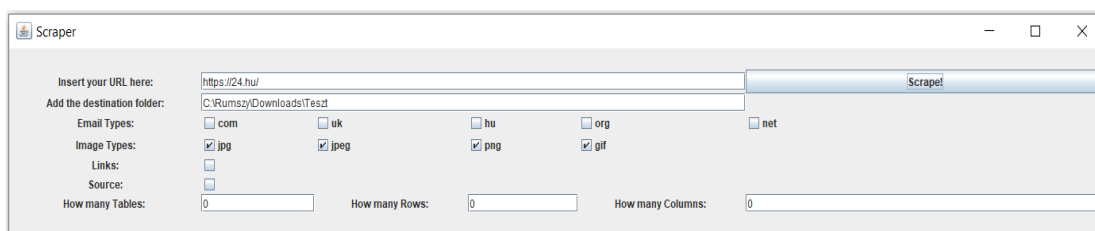
TESZTELÉS

mentette le, ami egymás után tartalmazott két darab ”.” karaktert, azonban a tesztelésre felhasznált oldal tartalmazott ilyen címet is.

Volt még rengeteg weboldal amelyeket a monotonitás, illetve az idő hiánya miatt nem sorolnék fel. A fent említett kettő tesztet nagy valószínűséggel lefedi a teljes funkcionalitását az alkalmazásnak, az email címek kinyerését tekintve.

5.2 Képek kimentése

A képek kimentésénél is azt a stratégiát alkalmaztam a tesztelés során, mint az email címek kinyerésénél, hogy a lehető legtöbb kombinációt kipróbáljam a képek kiterjesztését tekintve, illetve, hogy a legtöbb weboldalon használjam. Először egy igen jól ismert magyar hírportálon próbáltam ki, mégpedig a <https://24.hu/> weboldalon. Az alábbi beállításokkal futtattam az alkalmazást.



9. ábra – Összes képformátum kiválasztva az alkalmazásban

Ebben az esetben tovább tartott, amíg sikeresen lefutott az alkalmazás, de minden hiba nélkül letöltötte az általunk előre kiválasztott formátumú képeket. A képek mérete miatt volt pár másodperc a futási idő, a korábbi pár pillanathoz képest az email címek esetében. Összesen 17 megabájt volt a lementett adatok mérete, amely tartalmazott egyetlen „JPEG”, negyvenhárom „JPG” és hét darab „PNG” kiterjesztésű képet. Ezeket a képeket kiterjesztéstől függően külön mappákba rendezve helyezte el az alkalmazás.

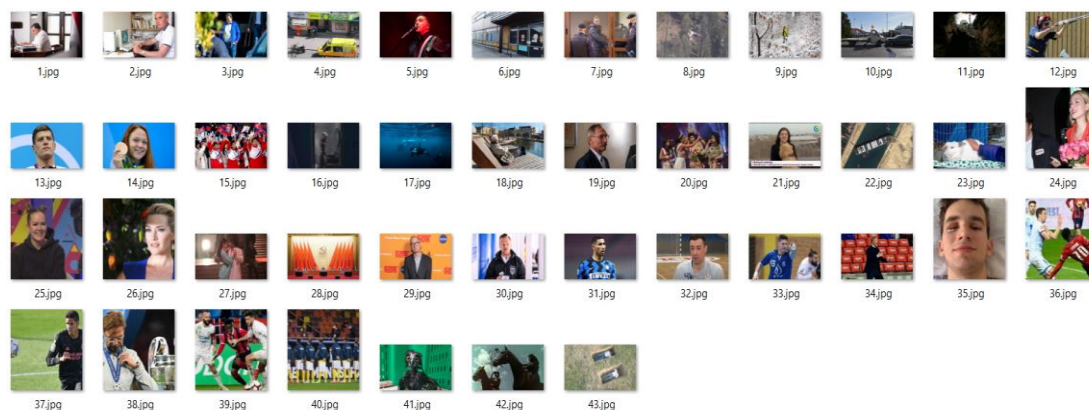
TESZTELÉS

This PC > Local Disk (C:) > Rumszy > Downloads > Teszt



10. ábra – Képek mappákba rendezve

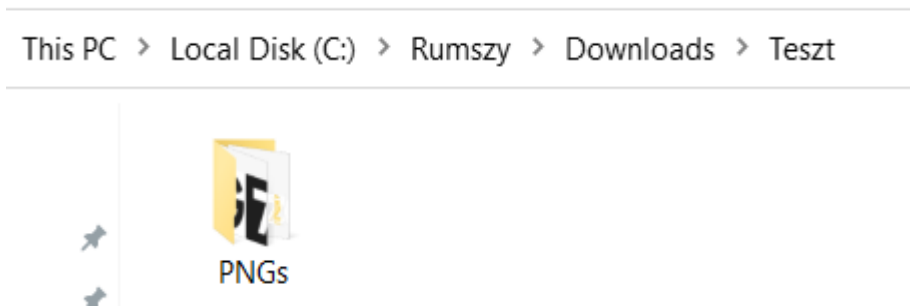
Minden egyes mappába 1-től kezdődően vannak sorszámozva a képek az oldalon való megjelenésük sorrendjében. Az átvizsgálás során azt is tudtam tesztelni, hogy kétszer ugyan azt a képet nem menti le az alkalmazás. Ez a funkció hasznos a redundancia elkerülésére és ezáltal számos tárhelyet kímélhetünk meg a felesleges felhasználástól. Mivel egy hírportálon futott ez a teszt és a nap bármely időszakában változhatnak a hírek, ezáltal a képek is eltérőek lehetek az itt bemutatott képektől. Ez a teszt 2021. április 6-án készült 20:42 perckor.



11. ábra – JPGs mappa tartalma

Kipróbáltam ugyanezen a weboldalon, azt hogy csak a „PNG” kiterjesztésű képeket szeretném lementeni. Beállítva ezeket a paramétereket az alkalmazásban, majd futtatva azt, a következő eredményt kaptam.

TESZTELÉS



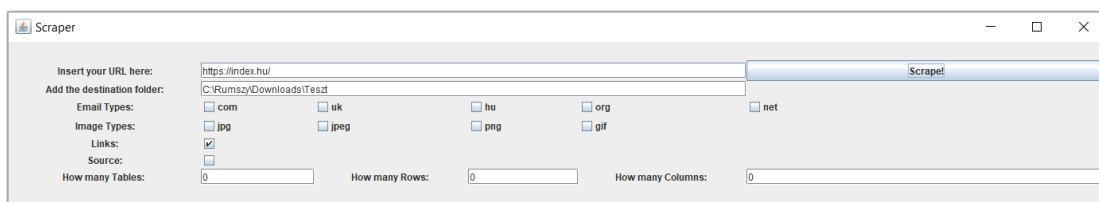
12. ábra – PNGs mappa

Ebből a tesztből is látszik, hogy ebben az esetben csak a „PNG” kiterjesztésű képeket gyűjtötte ki az alkalmazás.

További weboldalakon is teszteltem ezt a funkciót, viszont a fent említett kettő volt az a teszt eset, amely kielégítette a funkcionalitás képekre vonatkozó részét.

5.3 Linkek kinyerése

Ezt a funkciót is számtalan honlapon terveztem, míg az tudtam mondani, hogy megfelelően működik az alkalmazás ezen része. Lényegében azt kellett tesztelnem, hogy a linkeket megtalálja, illetve kigyűjti-e a program, vagyis rendeltetésszerűen működik. Ennél a tesztnél egy hazánkban jól ismert hírportált használtam fel a következő beállításokkal.



13. ábra – Linkek kiválasztva az alkalmazásban

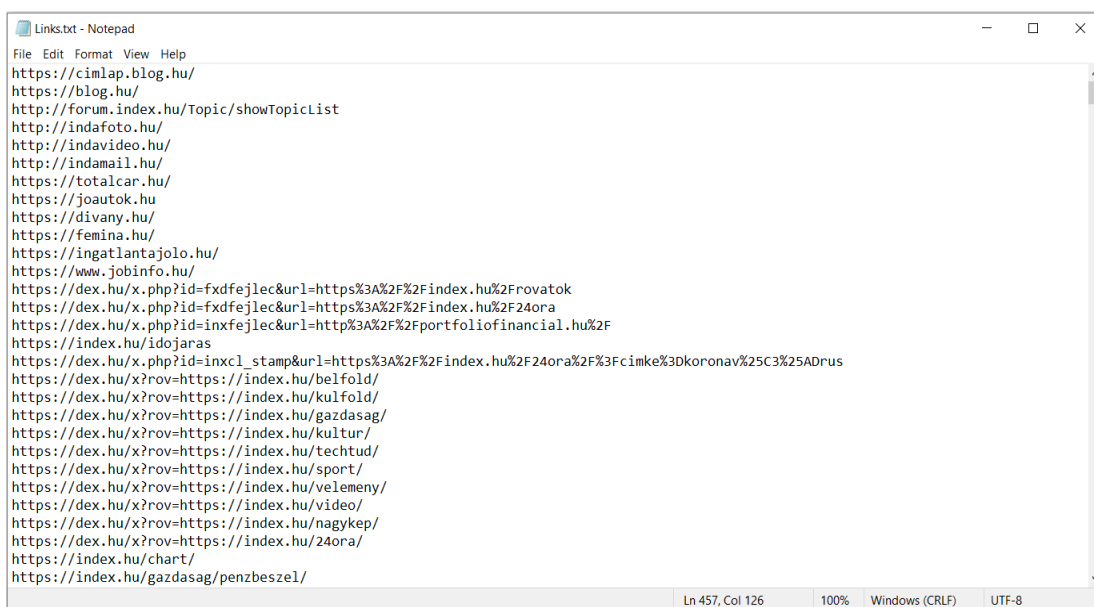
A Scrape! gomb megnyomása után pár pillanattal létrejött a megadott helyre a Links.txt állomány, ami tartalmazza az általunk megadott oldal összes hivatkozását, külön sorokba tagolva, a jobb átláthatóság kedvéért. A létrehozott állomány megnyitása után az is jól látható volt, hogy kétszer nem szerepel ugyanaz a link, vagyis a szűrés jól működik az alkalmazásban.

TESZTELÉS

This PC > Local Disk (C:) > Rumszy > Downloads > Teszt



14. ábra – Links.txt állomány



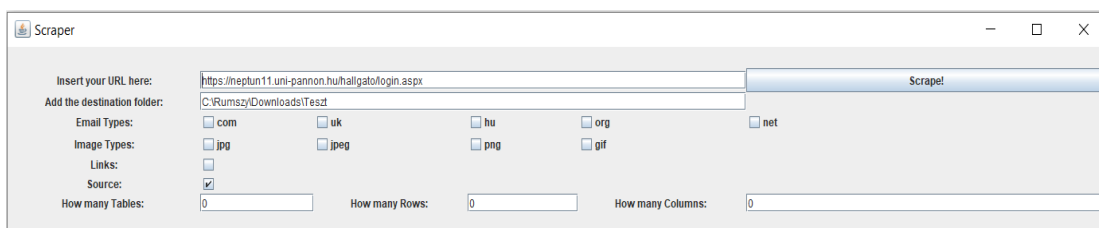
15. ábra – Links.txt állomány tartalma

A fent ábrázolt képeken is látszik, hogy rendben működik az alkalmazás, illetve az is, hogy összesen 457 darab linket talált a hivatkozás. Időm hiánya miatt nem az összes linket próbáltam beilleszteni a böngészőbe, hogy működnek-e, viszont szűrőpróbaszerűen körülbelül 70-80 címet kipróbáltam, és mindegyikük valós volt. Más weboldalakon is kipróbáltam ennek a funkciónak a működését és azokon is megfelelően működött. Azt is kipróbáltam, hogy nem volt kiválasztva az alkalmazásban ez a funkció, ebben az esetben nem is jött létre egyetlen állomány sem, ezáltal az elvártak megfelelt.

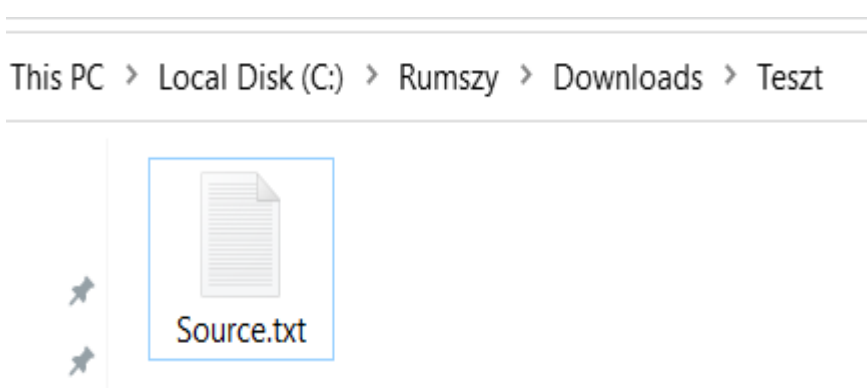
TESZTELÉS

5.4 Forráskód kinyerése

Ezen funkció tesztelésére nem fordítottam hatalmas hangsúlyt, hiszen itt nem fordulhattak elő komplex esetek, amelyeket többféleképpen kellett volna tesztelni. Lényegében ez a funkció vagy működik, vagy nem. Természetesen ez nem azt jelenti, hogy csak pár weboldalon teszteltem a funkciót. Ebben az esetben is sok oldalon kipróbáltam az alkalmazást és szerencsére nem is találkoztam olyannal ahol ne működött volna. Azt is leteszteltem, hogy ha a felhasználó nem szeretné kinyerni a forráskódot, akkor mi történik. Ezt az akadályt is sikeresen vette a program, hiszen amikor nincsen bekapcsolva ez a funkció, akkor nem jön létre semmilyen üres állomány. Amikor viszont bekapcsoltam a forráskód kinyerése funkciót, minden a legnagyobb rendben ment és egy Source.txt nevű fájlba íródott ki az adat, az általunk megadott helyre. A következő adatokkal teszteltem az alkalmazást. A weboldal amit választottam az egyetemi körökben jól ismert Neptun tanulmányi rendszer volt: <https://neptun11.uni-pannon.hu/hallgato/login.aspx>.



16. ábra – Forráskód kiválasztva az alkalmazásban



17. ábra – Source.txt állomány

Alig pár pillanatot vett igénybe az egész művelet. Sikeresen kigyűjtésre került a weboldal forráskódja. A következő képen is látható, hogy szépen tagolja az alkalmazás és könnyen beazonosíthatók a különböző egységekbe tartozó részek.

TESZTELÉS



```
Source.txt - Notepad
File Edit Format View Help
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+ARIA 1.0//EN" "http://www.w3.org/WAI/ARIA/schemata/xhtml-aria-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="hu">
<head>
<title>
Neptun.Net PE_HW_N11
</title>
<meta http-equiv="X-UA-Compatible" content="IE=10">
<meta content="Microsoft Visual Studio .NET 7.1" name="GENERATOR">
<meta content="C#" name="CODE_LANGUAGE">
<meta content="JavaScript" name="vs_defaultClientScript">
<meta content="http://schemas.microsoft.com/intellisense/ie5" name="vs_targetSchema">
<link id="Link1" rel="shortcut icon" href="favicon.ico" type="image/x-icon">
<link id="Link2" rel="icon" href="favicon.ico" type="image/ico">
<script>
document.msCapsLockWarningOff = true;
</script>
<link href="App_Themes/Skin_Neptun_Login/Custom/Hallgato/_Style_Neptun_Login_Custom.css" type="text/css" rel="stylesheet">
<link href="App_Themes/Skin_Neptun_Login/Custom/Oktato/_Style_Neptun_Login_Custom.css" type="text/css" rel="stylesheet">
<link href="App_Themes/Skin_Neptun_Login/jquery.widget.css" type="text/css" rel="stylesheet">
<link href="App_Themes/Skin_Neptun_Login/jquery-ui.css" type="text/css" rel="stylesheet">
<link href="App_Themes/Skin_Neptun_Login/Style_Neptun_Login.css" type="text/css" rel="stylesheet">
<link href="/hallgato/WebResource.axd?d=XuBEY3uS3cdfTmuvucADk-
R_Nsbr_juTk88hjhh016tuegwvf8neaH8h5xgbdwS6nN7q348EbmWAXF0d0SNRF9qM775dP5WEQWbc773SldSp7KytmeNPDMGe4-
XHBEs_snjjzHyWRWfWfHKLx58DgJIVv7Jc3IfEFRYnM7cWbo1&amp;t=637522476312968750" type="text/css" rel="stylesheet">
<link href="/hallgato/WebResource.axd?
d=ZDQKxb5x5xp_wmcjcxwHIpXamMgyvSX19czNMHJX8rIvVbnSXbtfwkdkfN8Sqmwt02kZzj6N1toA5LXeTyOPTM0o1RCUu-q0bAd2qLHLkgh55KSF1VxIplW4uNBd-
_CRU8nXGWMvzz55sinZHcuJHj3gmLPqr6B5tLP9bhcq65FCYx-
6ZXK902fbyJ5MTQNFZ9mxjNeucFNTELdwiXSpf8541NymDi0bbqNwS8g1&amp;t=637522476624375000" type="text/css" rel="stylesheet">
</head>
Ln 896, Col 8 100% Unix (LF) UTF-8
```

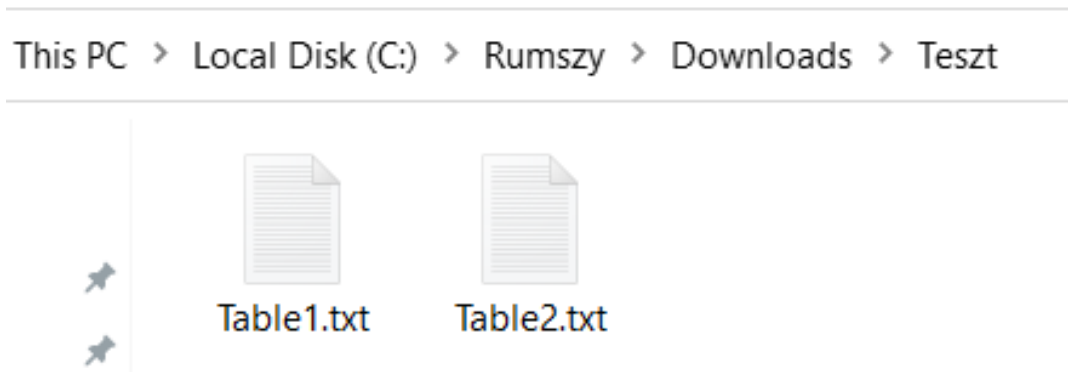
18. ábra – Source.txt állomány tartalma

Ezen a példán szerettem volna bemutatni a funkció tesztelését és maga a létrehozott állományokat, hogy hogyan is néznek ki a gyakorlatban.

5.5 Táblázatok kinyerése

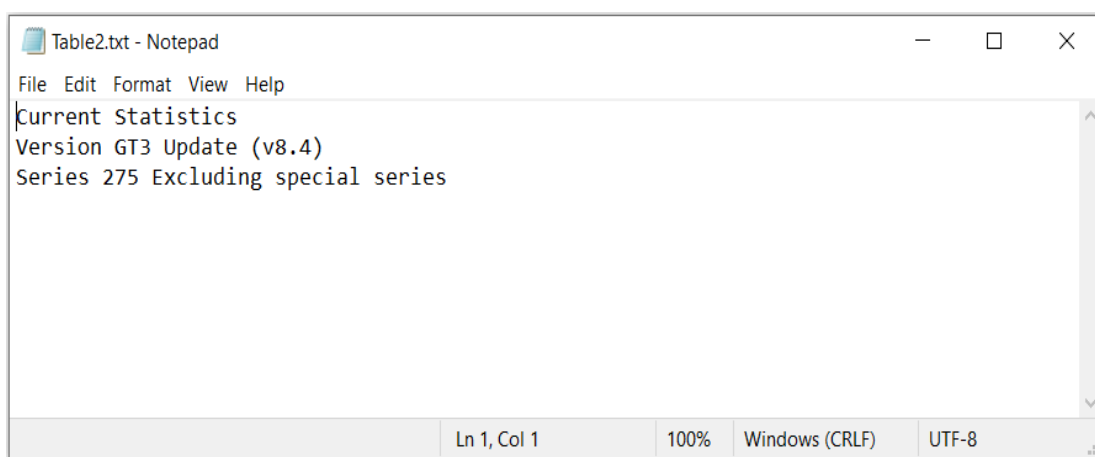
Ennek a funkciónak a tesztelésére fordítottam a legnagyobb hangsúlyt és ezzel töltöttem a legtöbb időt, hiszen ennek a funkciónak az implementálása tartott a leghosszabb ideig és elég bonyolult megoldásokat tartalmaz amelyekkel napokig szenvedtem a megvalósítás során, hogy valóban működőképes legyen az alkalmazás ezen része is. Több oldalon teszteltem ezt is mind az összes többi funkciót is. Témavezetőm javaslatára egy autóversenyzős játéknak a rajongók által létrehozott oldalt használtam a tesztelésre, mivel rengeteg verseny, autó és egyéb dolog adatait tartalmazza táblázatok formájában. Ennek az oldalnak a címe: https://rr3.fandom.com/wiki/RR3_Wiki:All_Series?fbclid=IwAR14-plliKHBdi466zby7SXkqw8_s9Xp8dw1vP-f9StB4aNRXLjvnndR1y0. Először azt teszteltem, hogy a táblázatok adatainál minden mező "0" értékkel van kitöltve. Ebben az esetben egy állomány sem jött létre, vagyis a funkció ezen része működött. Következő lépésként az alábbi paraméterekkel mutatom be az alkalmazást: táblázatok száma: 2, sorok száma: 3, oszlopok száma: 4.

TESZTELÉS



19. ábra – Table1.txt és Table2.txt állományok

A fent látható képen jól látszik, hogy az elvártaknak megfelelően kettő darab állomány jött létre, hiszen ezt adtuk meg az alkalmazásban. Minden egyes táblázat tartalma külön állományba kerül lementésre külön sorszámmal, abban a sorrendben amilyen sorrendben megjelenik a weboldalon. Példaként megnyitottam a második táblázatot, hogy lássam jól működött-e az implementáció többi része is.



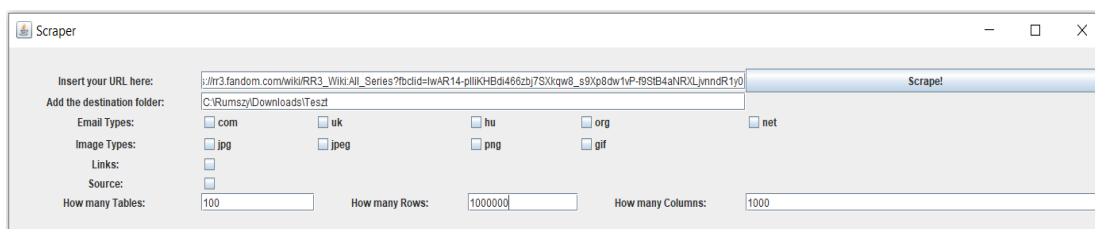
20. ábra – Table2.txt állomány tartalma

Current Statistics		
Version	GT3 Update (v8.4)	
Series	275	Excluding special series
Special Events	0	
Events	12,511	Excluding special events
Last Updated	May 25th 2020	

21. ábra – A teljes táblázat a weboldalon

TESZTELÉS

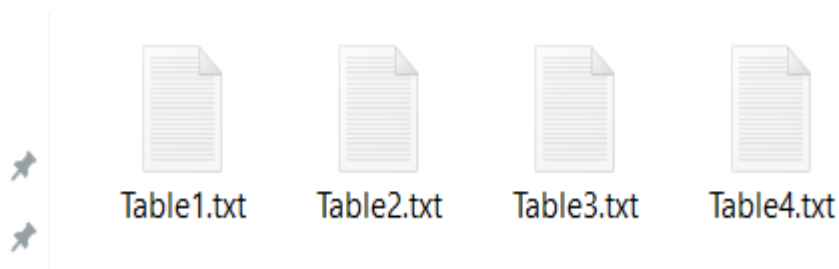
A fenti két képernyőfelvételtől is jól látszik, hogy az alkalmazás jól működött, hiszen a táblázat három sora, és ennek a három sornak mind a három oszlopa kimentésre került a létrehozott állományba. Ugyan mi azt adtuk meg, hogy négy darab oszlopra vagyunk kíváncsiak, viszont az alkalmazás úgy lett megírva, hogy ha nagyobb értéket adunk meg a valós méretnél, abban az esetben az összes érték kigyűjtésre kerül. Ez a teszt minden szempontból sikeres volt. A következő lépésben arra voltam kíváncsi, hogy vajon az összes táblázat összes adatára működik-e az alkalmazás. Ennek a kipróbálásához irreálisan nagy számokat adtam meg paraméterekként. Számszerűen: táblázatok száma: 100, sorok száma: 1000000, oszlopok száma: 1000.



22. ábra – Táblázatok kimentése nagy számokkal

A futtatás után nagyon rövid időn belül meg is kaptam az eredményt, ami a következő volt. Négy darab állományt hozott létre az alkalmazás, ami azt jelenti, hogy négy táblázatot tartalmaz az általunk választott weboldal. Egy rövid ellenőrzés után kiderült, hogy valóban annyit tartalmaz.

This PC > Local Disk (C:) > Rumszy > Downloads > Teszt

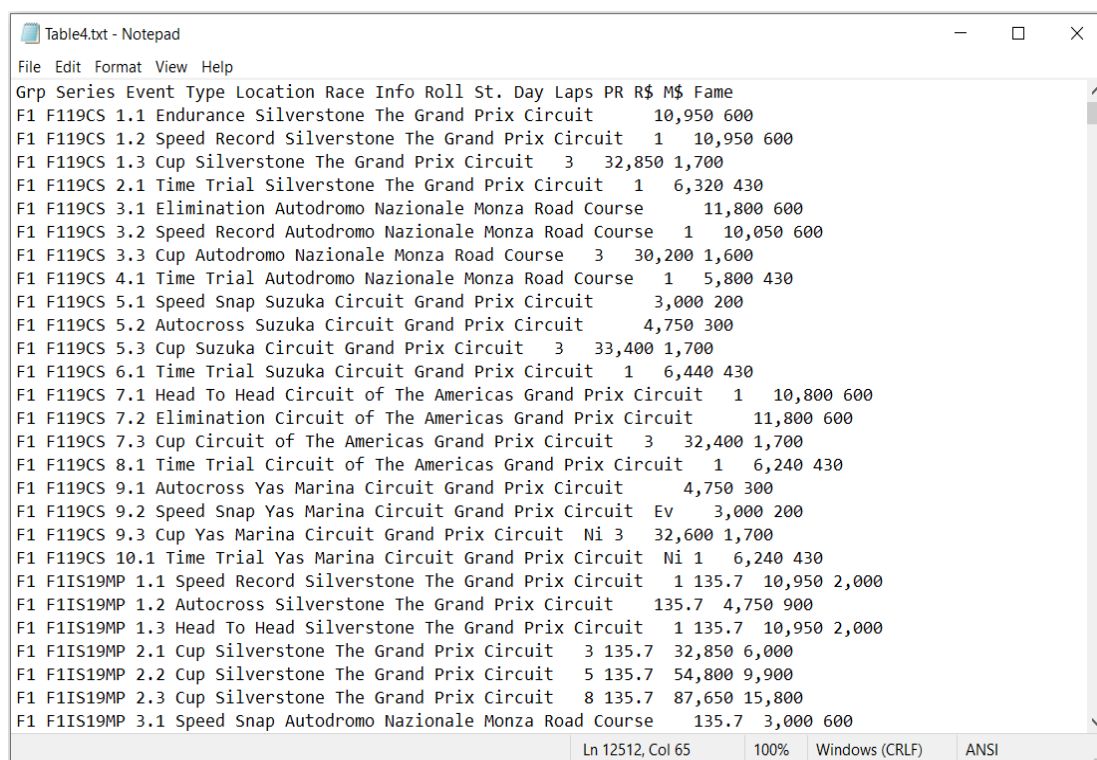


23. ábra – A négy táblázathoz tartozó állomány

Az állományok megnyitása után kiderült, hogy a negyedik táblázat tartalmazza a legtöbb adatot. Az táblázat teljességét úgy ellenőriztem, hogy a fájl

TESZTELÉS

utolsó adatára rákerestem a weboldalon és valóban a teljes táblázat kimentésre került.



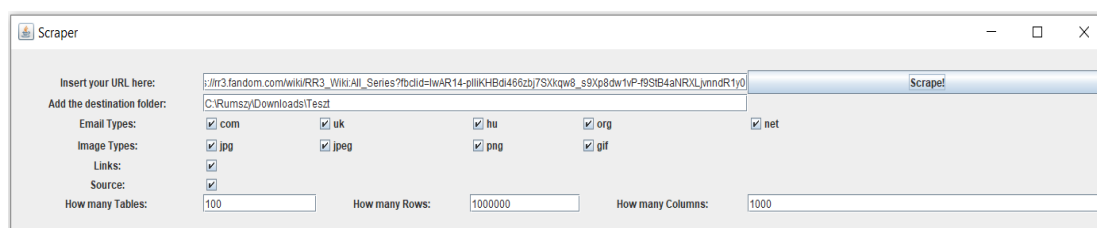
Grp	Series	Event	Type	Location	Race	Info	Roll	St.	Day	Laps	PR	R\$	M\$	Fame
F1	F119CS	1.1	Endurance	Silverstone	The Grand Prix Circuit					10,950	600			
F1	F119CS	1.2	Speed Record	Silverstone	The Grand Prix Circuit		1			10,950	600			
F1	F119CS	1.3	Cup	Silverstone	The Grand Prix Circuit		3			32,850	1,700			
F1	F119CS	2.1	Time Trial	Silverstone	The Grand Prix Circuit		1			6,320	430			
F1	F119CS	3.1	Elimination	Autodromo Nazionale	Monza Road Course					11,800	600			
F1	F119CS	3.2	Speed Record	Autodromo Nazionale	Monza Road Course		1			10,050	600			
F1	F119CS	3.3	Cup	Autodromo Nazionale	Monza Road Course		3			30,200	1,600			
F1	F119CS	4.1	Time Trial	Autodromo Nazionale	Monza Road Course		1			5,800	430			
F1	F119CS	5.1	Speed Snap	Suzuka Circuit	Grand Prix Circuit					3,000	200			
F1	F119CS	5.2	Autocross	Suzuka Circuit	Grand Prix Circuit					4,750	300			
F1	F119CS	5.3	Cup	Suzuka Circuit	Grand Prix Circuit		3			33,400	1,700			
F1	F119CS	6.1	Time Trial	Suzuka Circuit	Grand Prix Circuit		1			6,440	430			
F1	F119CS	7.1	Head To Head	Circuit of The Americas	Grand Prix Circuit		1			10,800	600			
F1	F119CS	7.2	Elimination	Circuit of The Americas	Grand Prix Circuit					11,800	600			
F1	F119CS	7.3	Cup	Circuit of The Americas	Grand Prix Circuit		3			32,400	1,700			
F1	F119CS	8.1	Time Trial	Circuit of The Americas	Grand Prix Circuit		1			6,240	430			
F1	F119CS	9.1	Autocross	Yas Marina Circuit	Grand Prix Circuit					4,750	300			
F1	F119CS	9.2	Speed Snap	Yas Marina Circuit	Grand Prix Circuit		Ev			3,000	200			
F1	F119CS	9.3	Cup	Yas Marina Circuit	Grand Prix Circuit		Ni	3		32,600	1,700			
F1	F119CS	10.1	Time Trial	Yas Marina Circuit	Grand Prix Circuit		Ni	1		6,240	430			
F1	F11S19MP	1.1	Speed Record	Silverstone	The Grand Prix Circuit		1	135.7		10,950	2,000			
F1	F11S19MP	1.2	Autocross	Silverstone	The Grand Prix Circuit		135.7			4,750	900			
F1	F11S19MP	1.3	Head To Head	Silverstone	The Grand Prix Circuit		1	135.7		10,950	2,000			
F1	F11S19MP	2.1	Cup	Silverstone	The Grand Prix Circuit		3	135.7		32,850	6,000			
F1	F11S19MP	2.2	Cup	Silverstone	The Grand Prix Circuit		5	135.7		54,800	9,900			
F1	F11S19MP	2.3	Cup	Silverstone	The Grand Prix Circuit		8	135.7		87,650	15,800			
F1	F11S19MP	3.1	Speed Snap	Autodromo Nazionale	Monza Road Course		135.7			3,000	600			

24. ábra – Table4.txt állomány tartalma

A fenti ábrán jól látható, hogy egy 12512 sorból álló táblázatról van szó, ami korántsem mondható kicsinek. Nagyon ritkán fordulnak elő az ilyen nagy terjedelmű táblázatok, azonban az alkalmazás gond nélkül megbirkózott az akadállyal. Az a szempont sem elhanyagolható, hogy alig pár másodperc alatt végzett is a program a futással.

5.6 Összes funkció

Az összes funkciót egyben is teszteltem, hogy így is működnek-e, és nem csak külön-külön. Mindent bekapcsoltam az alkalmazásban, amit csak lehetett.



Insert your URL here:

Add the destination folder:

Email Types: ☒ com ☒ uk ☒ hu ☒ org ☒ net

Image Types: ☒ jpg ☒ jpeg ☒ png ☒ gif

Links: ☒

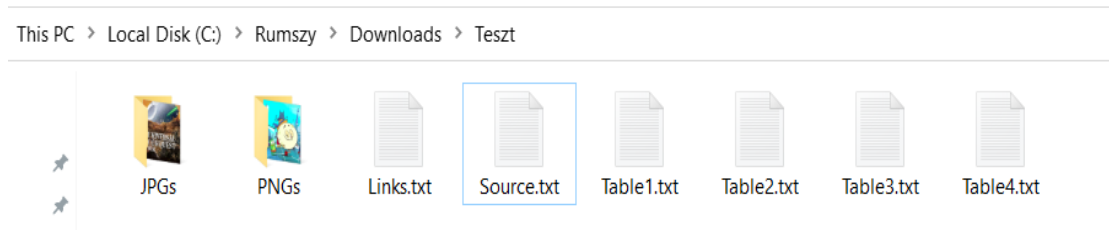
Source: ☒

How many Tables: How many Rows: How many Columns:

25. ábra – Minden funkció kiválasztása az alkalmazásban

TESZTELÉS

Ezzel a beállítással megnőtt a futási idő, hiszen több dolgot kellett elvégeznie a programnak, mint amikor a funkciókat külön-külön teszteltem. Azonban még ez az idő is elhanyagolható. Nagyon pozitív volt látni, hogy mennyire gyors is maga a Kotlin nyelv. A futás után ezek az állományok jöttek létre.



26. ábra – Az alkalmazás által létrehozott állományok

Nem láthatunk Emails.txt állományt, először kicsit megijedtem, hogy nem működik ez a funkció, azonban egy gyors ellenőrzés után kiderült, hogy az oldal nem tartalmaz egyetlen email címet sem.

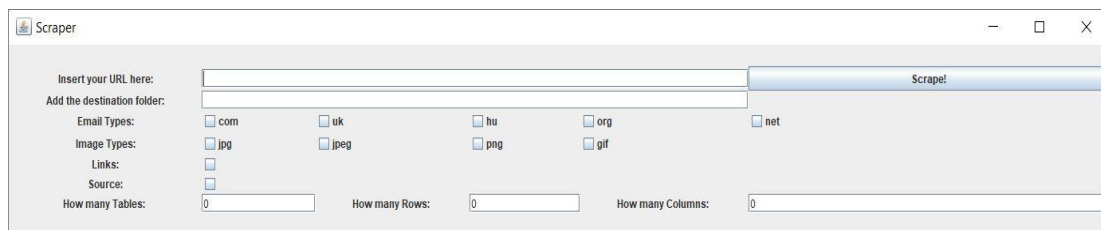
6. Felhasználás

Mindenekelőtt bemutatom a programot egy általános szinten, hogy hogyan is lehet hozzájutni és hogyan használható. Utána kifejtem az összes funkcióját az alkalmazásnak, hogy konkrétan mire is képes. 5 főbb funkcióval rendelkezik, amelyek a következők: email címek, képek, linkek, táblázatok, illetve a megadott honlap forráskódjának kinyerése.

6.1 Maga az alkalmazásról

Első lépésként el kell látogatnunk erre a weboldalra, ahol a program végső verziója, illetve a forráskódja is megtalálható: <https://github.com/Rumszy/Scraper>. Ezután el kell navigálnunk a „Jar” nevű mappába, azon belül fogjuk megtalálni maga az alkalmazás futtatható verzióját, aminek a neve Scraper.jar. Innen ingyenesen beszerezhető és szabadon felhasználható bárki számára. Az alkalmazás bárhová elhelyezhető, mindenhol kiválóan működik. Azonban mielőtt használni tudnánk az alkalmazást, szükségünk van Java futási környezetre (JRE – Java Runtime Environment). Erről az oldalról ingyenesen letölthető: <https://java.com/en/download/manual.jsp>. Ajánlom mindig a legfrissebb verzió beszerzését, ebben az esetben megakadályozhatjuk az esetleges hibákat. Ezen az oldalon ki kell választanunk a számukra megfelelő verziót, ebben pedig segítségünkre lesz az oldalon található számtalan információ ezzel kapcsolatban. Amit még fontos kiemelni, hogy a Java egy multiplatform programozási nyelv, hiszen egy Java compiler fordítja le a forráskódot JVM bytecode-ra, és ezt a kódot tudja futtatni a Java virtuális gép. Ugyan az alkalmazás Kotlin nyelven van írva, viszont oda-vissza száz százalékban kompatibilis a Java nyelvvel, vagyis ez előnyünkre válik. Ezáltal a java futási környezet elérhető Windows, Linux és még Mac OS rendszerekre is. Miután kiválasztottuk a megfelelő verziót, le kell töltenünk, és pár kattintás után fel is települt a számítógépünkre a JRE. Amint ezzel megvagyunk, máris elindítható a Scraper nevű alkalmazás. Az indítás után ez grafikus felhasználói felület fogad bennünket.

FELHASZNÁLÁS



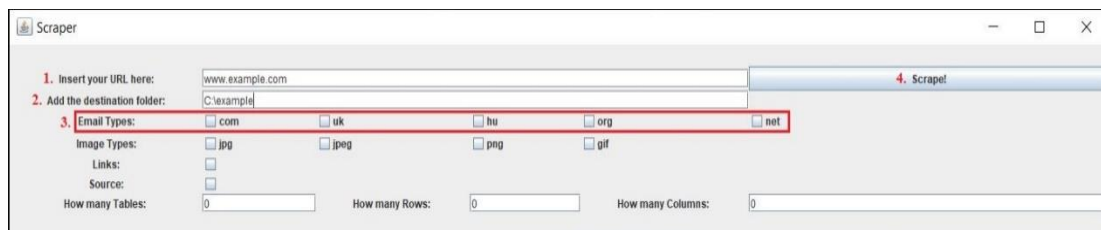
27. ábra – Az alkalmazás grafikus felhasználói felülete

Jó néhányszor kiemeltem, hogy az egyik legfontosabb szempontnak a letisztult grafikus felhasználói felületet tartom. Hiszen nagyon sokan vannak olyan felhasználók, akik pont a bonyolult rendszerek láttán futamodnak meg egy-egy alkalmazás használata előtt. Ennek kiküszöbölésére alkottam meg ezt az egyszerű, letisztult dizájnt. A program összesen kettő darab szövegmezőt tartalmaz ahová a kívánt oldal URL címét kell megadnunk, illetve azt a mappát ahová szeretnénk az adatokat lementeni. Továbbá jelölőnégyzeteket tartalmaz, amik segítségével kiválaszthatók azok a funkciók, amelyeket szeretnénk igénybe venni. Legalul pedig 3 mező látható ahová a táblázatokkal kapcsolatos információkat kell megadnunk. Amennyiben minden információt megadtunk, nincs más dolgunk, mint rákattintani a jobb felül található gombra és a program elvégzi a feladatot.

6.2 Email címek kinyerése

Ezt a funkciót több okból kifolyólag is szerettem volna leimplementálni. A legfőbb ok az volt, hogy rengeteg olyan weboldal létezik, amelyen sok elérhetőség van feltüntetve különféle formákban. Nagyon gyakran email cím formájában is megadják az elérhetőséget. Például fel van sorolva több száz ember egy honlapon és minden egyes személynek el szeretnénk küldeni egy levelet email címükön keresztül. Ebben az esetben, nem kell percekben keresztül, vagy akár órákon át pásztáznunk a szemünkkel a monitoron email címek után kutatva. Helyette használhatjuk a Scraper nevű alkalmazást. Rengeteg időt és fáradságot tudunk a használatával megspórolni magunknak. Miután letöltöttük az alkalmazást, futtassuk és az alább látható felhasználói felület fogad bennünket.

FELHASZNÁLÁS



28. ábra – Email címekhez tartozó funkció lépései

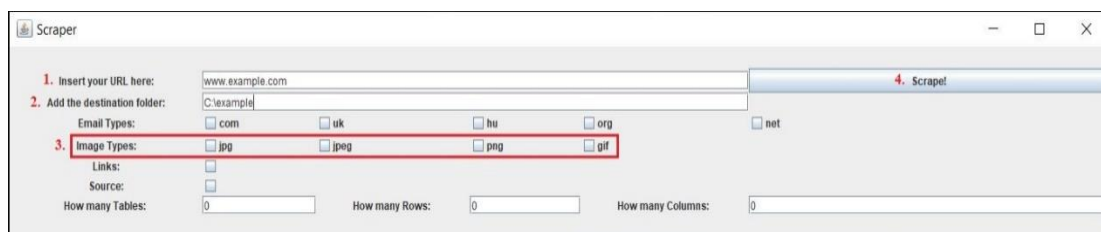
Mindössze pár lépés elvégzése után már hozzá is jutottunk az összes email címhez, amely az adott honlapon fellelhető volt. Ezek a lépések a következők: először is adjuk meg a weboldal URL címét, ahonnan szeretnénk az email címeket kigyűjteni és másoljuk be a legfelső mezőbe. Ez a mező a példa képen a következő szöveget tartalmazza: `www.example.com` és 1-es számmal jelölve is van. Következő lépésként meg kell adnunk, hogy a számítógépünk tárhelyén hová szeretnénk, hogy az email címek kigyűjtésére kerüljenek. Ezt az eggyel lejjebb lévő mezőbe kell beillesztenünk, ahogy a példánkból is jól látszik, 2-es számmal. Ha ezekkel a lépésekkel megvagyunk, akkor ki kell választanunk, hogy milyen tartománynevű email címek érdekelnek bennünket, ez 3-as számmal van feltüntetve a képen. A fejlesztés során próbáltam a leggyakoribbakat felsorolni, hogy a legtöbb email címet kezelni tudja a program. Ezek pedig a következő tartománynevek: `.com`, `.uk`, `.hu`, `.org` és a `.net`. Utolsó lépésként nincs más dolgunk, mint rákattintani a Scrape! feliratú gombra, ami fentebb a képen is jól látható, 4-es számmal. Készen is vagyunk! Amennyiben elnavigálunk a saját gépünkön abba a mappába, amit az alkalmazásban megadtunk, ott találni fogunk egy `Email.txt` nevű szöveges állományt, amelyben kilistázva megtalálható az összes email cím, külön sorban, ami fellelhető volt a megadott weblapon.

6.3 Képek kimentése

A következő funkció, amelynek a használatáról bővebben írok, az a képek mentése előre megadott weboldalról. Ez a funkció azok számára lehet hasznos, akik szeretnének rengeteg képet lementeni egy oldalról, anélkül, hogy egyesével kellene minden egyes képet letölteni. Ennek a funkciónak köszönhetően, egyetlen kattintással sok képhez juthatunk. A Scraper letöltése után igénybe vehető ez a

FELHASZNÁLÁS

hasznos funkció. Az alábbi képet alapul véve végig megyünk azon, hogy hogyan is használjuk ezt a funkciót.



29. ábra – Képekhez tartozó funkció lépései

Az első kettő lépés teljes mértékben megegyezik azokkal a lépésekkel amelyeket korábban kifejtettem az "Email címek kinyerése" részben. A következő lépés is hasonló, viszont ebben az esetben az általunk kiválasztott képformátumokat tudjuk kiválasztani. Azokat a formátumokat válasszuk ki, amelyekre nekünk szükségünk van. Ez a lépés piros kerettel lett kiemelve a képernyőképen, és 3-as ponttal lett jelölve. A 4. egyben az utolsó lépés, hogy megnyomjuk a Scrape! feliratú gombot. Ezek után az általunk megadott helyen létrejönnek a mappák annak megfelelően, hogy milyen kiterjesztésű képeket választottunk ki az alkalmazásban. Ha például a „jpg” és a „png” formátumot választottuk ki, ebben az esetben az általuk megadott helyre egy „JPGs” és egy „PNGs” nevű mappa jön létre és bennük megtalálhatóak lesznek a letöltött képek.

6.4 Linkek kinyerése

Ez a funkció arra szolgál, hogy az általunk megadott honlapon található összes hivatkozást ki tudjuk gyűjteni. Hasznos lehet például azok számára, akik kíváncsiak, hogy az általuk kiválasztott oldal milyen más oldalakkal van összekötve különböző hivatkozásokkal. Sok érdekesség fellelhető egy ilyen kutatás során. Olykor meglepő tud lenni, hogy egy oldal milyen, vele szöges ellentétben álló típusú oldalakra mutató hivatkozásokat tartalmaz. Ennek a funkciónak a használata a következő képen illusztrálva is látható.

FELHASZNÁLÁS



30. ábra – Linkekhez tartozó funkció lépései

Az első kettő lépés a megszokott módon végrehajtandó. A 3. lépés igazán egyszerű: a képen pirossal jelzett területen ki kell választanunk, hogy szeretnénk-e, hogy a program kigyűjtse számunkra a hivatkozások címét egy szöveges állományba, amelyeket később könnyedén fel tudunk használni, hiszen egy helyen lesz megtalálható az összes. A Scrape! gomb megnyomása után ez a file el is érhető Links.txt néven, minden egyes link külön sorban található meg.

6.5 Forráskód kinyerése

A forráskód kinyerése funkció azok számára lehet hasznos, akik jártasok az informatika világában. Esetleg szeretne valaki egy hasonló weboldalt készíteni egy általa jól ismert weblaphoz, ez a funkció nagy segítség lehet, hiszen ezáltal kinyerhető a forrása az adott oldalnak és megtekinthetővé válik a felépítése. Így könnyedén megtanulható, illetve elsajátítható minden egyes funkció, amit egy honlapon használni lehet. Lentebb, az alkalmazáson illusztrálva is látható ennek a funkciónak a használata.



31. ábra – Forráskódhoz tartozó funkció lépései

Az első két lépés ugyanaz, mint a korábbi esetekben. Amint ezeket megtettük, be kell jelölnünk, hogy szeretnénk-e az adott oldal forrásának a kinyerését. Amint igen, abban az esetben jelöljük be a 3. pontnál látható opciót. Végül pedig kattintsunk a Scrape! gombra. Amint ezt megtettük, az általunk előre megadott helyre létrejön egy Source.txt amely tartalmazza a teljes forráskódot.

FELHASZNÁLÁS

6.6 Táblázatok kinyerése

Az utolsó funkció, ám talán az egyik leglényegesebb a táblázatok kinyerése amire az alkalmazás képes. Hasznos lehet azon személyek számára, akiknek rengeteg adatot kell feldolgozniuk, ezek az információk legtöbbször táblázatok formájában vannak reprezentálva az oldalakon, hiszen egy jól átlátható és rendezett formát reprezentálnak a táblázatok. Az alkalmazás lehetővé teszi a felhasználók számára, hogy az összes táblázat összes adatát kinyerjük saját gépükre, egy előre meghatározott helyre. Azonban, ha nincs szükségünk az összes táblázatra, megadhatjuk azt is, hogy hány táblázatnak az adatait szeretnénk kigyűjteni. Tovább haladva, nem feltétlen van szükségünk a kívánt táblázatok összes sorára. Ezt a paramétert is megadhatjuk az alkalmazásban, hogy hány sorra vagyunk kíváncsiak. Hogy teljes legyen a paletta, az oszlopok számát is előre definiálhatjuk, ha nincsen szükségünk az összes oszlopra. Amennyiben nem tudjuk, hogy az adott oldalon hány táblázat, a legnagyobb táblázat hány sorból és hány oszlopból áll, de nekünk az összes táblázat teljes terjedelmére szükségünk van, nincsen semmi gond. Ugyanis, ha egy irreálisan nagy számot adunk meg mind a három paraméternek, abban az esetben nagy eséllyel ezek a számok nagyobbak lesznek, mint a táblázatok valós adatai. Ilyenkor az összes táblázat összes adatát ki tudjuk nyerni a weboldalról. Ebben az esetben is kép formájában szemléltetem ennek a funkciónak a használatát.

32. ábra –Táblázatokhoz tartozó funkció lépései

Az első két lépést a korábbiakhoz megszokott módon kell végrehajtani. Meg kell adnunk a kívánt oldalnak az URL címét, illetve a helyet ahová szeretnénk a táblázatok lementését. Ezek után a harmadik, negyedik és ötödik lépés nagyon hasonlít egymáshoz. Meg kell adnunk számszerűen, hogy hány darab táblázatot, hány sort, illetve oszlopot kívánunk lementeni az oldalról. Az utolsó lépés itt is, mint az eddigi összes esetben, a Scrape! gomb megnyomása. Ezután az általunk

FELHASZNÁLÁS

megadott helyen létrejön annyi TableX.txt szöveges állomány, amennyi táblázatot le kívántunk tölteni az oldalról. Az állomány nevében az X a különböző sorszámokat jelöli 1-től kezdődően.

7. Továbbfejlesztési lehetőségek

Néhány gondolat a továbbfejlesztés lehetőségeiről, amelyeket idő hiányában sajnos nem tudtam megvalósítani a munkám során. Sok potenciált látok az alkalmazásban, amit ki lehetne bővíteni további funkciók implementálásával. A következőkre gondoltam: táblázatok kinyerése funkció kibővítése, további funkciók implementálása.

7.1 Táblázatok kinyerése funkció kibővítése

A továbbfejlesztést úgy képzelem el, hogy a táblázatoknál meg lehetne adni plusz feltételeket. Nevezetesen, hogy konkrétan melyik sortól, illetve oszloptól melyik sorig, illetve oszlopig van szükségünk, ezáltal nem csak a táblázat első x sorát és oszlopát tudnánk kinyerni. Jó bővítés lehet még az is, hogy a felhasználói felületen megjelenne egy gomb, amit ha kiválasztunk, akkor az összes táblázat, összes adatát lementené számunkra az alkalmazás.

7.2 További funkciók implementálása

További funkciók implementálását úgy értem, hogy más konkrét adatokat is meg tudna találni a program és lementené külön állományokba. Ilyen adatok lehetnek például a nevek, telefonszámok és a címek.

A jövőben ezeket a funkciókat szeretném továbbfejlesztetni, illetve leimplementálni őket.

ELŐNYÖK

8. Előnyök

Az előnyökről pár gondolat, amelyeket tapasztaltam a fejlesztés, illetve a funkciók hosszas tesztelése során.

Egyik nagy előnye az alkalmazásomnak, ami az egyik fő célom is volt a tervezés során - az a letisztult grafikus felhasználói felület. Ezáltal minden felhasználó számára könnyedén használható a program, hiszen nem kell hosszas órákat azzal tölteni, amíg beletanulunk a használatába a bonyolultsága miatt. Pár kattintás után az általunk kívánt adatok már a számítógépünkön is elérhetők.

Másik fontos cél volt a fejlesztés során, hogy ingyenes alkalmazás jöjjön létre. Kutatásom során a legtöbb hasonló alkalmazás használatáért valamilyen díj ellenében lehet csak hozzáférni. Ezt szerettem volna kiküszöbölni az alkalmazásommal, hogy minél több felhasználóhoz elérjen.

Utolsó előny amit kiemelnék, az az alkalmazás gyorsasága. Hosszas tesztelés közben lettem figyelmes arra, hogy milyen gyorsan képes végezni az alkalmazás, még rengeteg adat lementése közben is jól teljesített. Egy konkrét példát említenék, amit a táblázatok tesztelése során vettem észre. Erről a weboldalról mentettem le az összes adatot a teszt során: https://rr3.fandom.com/wiki/RR3_Wiki:All_Series?fbclid=IwAR14-plliKHBdi466zbj7SXkqw8_s9Xp8dw1vP-f9StB4aNRXLjvnndR1y0. Ezen a weboldalon összesen 4 darab táblázat található. Az első háromnak a mérete elhanyagolható, viszont az utolsó táblázat több, mint tizenkétezer sort tartalmaz, ami korántsem mondható kevésnek. Az összes adat lementésével mindössze 5 másodperc alatt végzett a program, ami elég gyorsnak mondható. Elsősorban ezt a sebességet a Kotlin nyelvnek köszönhetem.

9. Összefoglalás

Szakedolgozatom témája Kotlin a web scraping fejlesztése volt. Nagy kihívásként tekintettem a feladatra, mint később kiderült nem is hiába, hiszen voltak nehézségek a megvalósítás során. Azonban ezeket kitartó próbálkozással, illetve tájékozódással leküzdöttem. Három fő alappillért tartottam szem előtt a fejlesztés során. Ezek a következők voltak. Az első amit figyelembe vettem, hogy jelenleg a piacon elég nagy a szórás az ingyenesen, illetve valamilyen díj ellenében használható alkalmazások elérése terén. Éppen ezért az alkalmazásomat ingyenessé tettem, bárki számára elérhető. Rengeteg hasonló programmal találkoztam a kutatásom során, amelyeknek a kezelése, felhasználhatósága kifogásolható volt számomra, vagyis túl bonyolult volt azok igénybe vétele és nagy valószínűséggel a laikusok számára használhatatlanok. Ennek kiküszöbölése volt a második cél. Az alkalmazásomhoz nem kell semmilyen előzetes szakmabeli tudás, hiszen pár kattintás után könnyen használható. Végül a harmadik pont, amit figyelembe vettem, hogy a legtöbb hasonló alkalmazás nagyon magas funkcionalitással rendelkezik, ezáltal növelve a hibalehetőségek számát. Ezt el szerettem volna kerülni, minél jobban működő alkalmazást kívántam fejleszteni, ezért 5 funkciót terveztem meg, majd később meg is valósítottam ezeket. Személy szerint kisebb-nagyobb sikerrel sikerült is ezeket megvalósítanom. Összességében nagyon élveztem a feladat elvégzését és az alkalmazás fejlesztését. Úgy érzem sokat fejlődtem szakmailag, amit a későbbiekben kamatoztathatok munkám során is.

Irodalomjegyzék

[1] <https://jsoup.org/>. *JSoup Hivatalos Weboldala*

[2] <https://hu.wikipedia.org/wiki/E-mail>. *Wikipédia E-mail*

[3] https://en.wikipedia.org/wiki/Email_address. *Wikipédia E-mail cím*

[4] <https://24.hu/>. *24.hu Hírportál*

[5] <https://neptun11.uni-pannon.hu/hallgato/login.aspx>. *Neptun Tanulmányi Rendszer*

[6] https://rr3.fandom.com/wiki/RR3_Wiki:All_Series?fbclid=IwAR14-plliKHBdi466zby7SXkqw8_s9Xp8dw1vP-f9StB4aNRXLjvnndR1y0. *Real Racing Wikipédia*

[7] <https://github.com/Rumszy/Scraper>. *GitHub Scraper*

[8] <https://java.com/en/download/manual.jsp>. *Java Hivatalos Weboldala*