Pannon Egyetem   
Műszaki Informatikai Kar  
Projekt Alapú Szoftverfejlesztés (VEMISAM233P/2020/21/2)  
Programtervező informatikus MSc

**Publikációs listák szinkronizálása**

2021

(RSZT 3. téma)

Témavezető: Szekér Szabolcs

Csapatattagok:

Szászfai Júlia (GO45XL)  
 Csizmazia Máté (XI32IS)  
 Pataki Miklós (BQPM6L)

**Tartalom**

[**1.** **MTMT és Google Scholar rendszerek ismertetése** 3](#_Toc66370844)

[**1.1.** **MTMT rendszer** 3](#_Toc66370845)

[**1.2.** **Google Scholar rendszer** 3](#_Toc66370846)

[**2.** **Felhasználóhoz tartozó publikációs lista lehívása MTMT és Google Scholar rendszerekből** 3](#_Toc66370847)

[**3.** **Google Scholar API vizsgálata, kiértékelése (MTMT nem biztosít API-t)** 3](#_Toc66370848)

[**3.1.** **Scraper API** 3](#_Toc66370849)

[**3.2.** **SERP API** 4](#_Toc66370850)

[**3.3.** **SerpWow** 4](#_Toc66370851)

[**3.4.** **Scale SERP** 4](#_Toc66370852)

[**3.5.** **Publish or Perish** 5](#_Toc66370853)

[**3.6.** **scholarly** 5](#_Toc66370854)

[**4.** **Web scraping technológia bemutatása** 6](#_Toc66370855)

[**4.1 Web scraping fogalma** 6](#_Toc66370856)

[**4.2 Web scraping története** 7](#_Toc66370857)

[**4.3 Technikák** 7](#_Toc66370858)

[**4.3.1 Emberi másolás-beillesztés** 7](#_Toc66370859)

[**4.3.2 Mintaszöveg egyezés** 7](#_Toc66370860)

[**4.3.3 HTTP programozás** 8](#_Toc66370861)

[**4.3.4 HTML elemzés (parsing)** 8](#_Toc66370862)

[**4.3.4 DOM elemzés** 8](#_Toc66370863)

[**4.3.5 Függőleges összesítés** 8](#_Toc66370864)

[**4.3.6 Szemantikai annotáció felismerése** 8](#_Toc66370865)

[**4.3.7 Számítógépes látás – weboldal analízis** 9](#_Toc66370866)

[**4.4 Selenium** 9](#_Toc66370867)

[**5.** **Azonos feltételű kereső megvalósítása MTMT és Google Scholar rendszerekben** 9](#_Toc66370868)

[**6.** **Adatok kigyűjtése, találatok megjelenítése** 10](#_Toc66370869)

[**7.** **Találatok összehasonlítása és különbségek keresése** 10](#_Toc66370870)

[**8.** **Automatikus/szoftverrel történő összehasonlítás előnye a manuálissal szemben** 10](#_Toc66370871)

[**9.** **Adatok összehasonlítása és kiértékelése, különbségek felfedezése** 10](#_Toc66370872)

[**10.** **Irodalomjegyzék** 11](#_Toc66370873)

1. **MTMT és Google Scholar rendszerek ismertetése**
   1. **MTMT rendszer**
   2. **Google Scholar rendszer**
2. **Felhasználóhoz tartozó publikációs lista lehívása MTMT és Google Scholar rendszerekből**
3. **Google Scholar API vizsgálata, kiértékelése (MTMT nem biztosít API-t)**

A Google Scholar (Tudós) a tudományos cikkek és kutatási folyóiratok milliárdjaival az egyik legjobb online forrás a magas színvonalú tudományos kutatáshoz. Mivel azonban a Google nem nyújt API-t, amely lehetővé teszi a szükséges adatok nagy mennyiségű megkeresését és kinyerését, más számos módon kell megszerezni a szükséges adatokat. Esetleg külső felek API-ján keresztül, vagy webscraping technikával, kimásoljuk az adatokat. A következő részben néhány lehetőséget tárgyalunk ezek közül, és megadjuk előnyeit és hátrányait.

* 1. [**Scraper API**](https://www.scraperapi.com/blog/best-google-scholar-apis-proxies/#scraperapi)

A Scraper API egy proxy API, amelyet arra terveztek, hogy az internetről az adatokat a lehető legnagyobb mértékben lementse.

A Google-hoz hasonlóan nehezen scrap-elhető webhelyekhez készült Scraper API csökkenti a saját proxy infrastruktúrájának kiépítésének és fenntartásának szükségességét. Egyszerűen elküldi a lescrapelni kívánt URL-t az API-nak, és ez kezeli a változó proxykat, az automatikus újrapróbálkozásokat, a CAPTCHA-kat és a blokkolásokat, a hibákat elrejti és csak sikeres eredményeket hozza. Innentől kezdve a parancsfájlnak csak elemeznie kell a HTML-válaszból szükséges adatokat.

Ha egy előre elkészített Google Scholar scrap-elő könyvtárral kombináljuk, mint például a Scholarly, akkor néhány órán belül könnyedén létrehozhatunk egy egyedi Google Scholar API-t, amely megfelel az adatigényeinek.

99 dollárért havonta 1 000 000 Google Scholar oldalt is kinyerhetünk. A Scraper API-kat rendelkezik ingyenes próbaverzióval, ami 5000 ingyenes kérést tud.

Előnyök: A legolcsóbb lehetőség azok számára, akik megbízhatóan szeretnék kinyerni a Google Scholar adatait kutatási projektjeikhez. Ráadásul nagyvonalú ingyenes próbaverzió is rendelkezésre áll.

Hátrányok: Alapvető ismeretekre van szüksége a webes scrapelésről.

* 1. [**SERP API**](https://www.scraperapi.com/blog/best-google-scholar-apis-proxies/#serpapi)

A következő lehetőség a SERP API, amely remek lehetőség azok számára, akik csak a Google Scholar adatait szeretnék lekérni, és nem akarják, hogy saját webscrapereket készítsenek.

A SERP API egy kifejezetten a Google Scholar számára létrehozott API, amely a tipikus keresési eredményekben található összes adatot visszaadja; beleértve: cím, link, töredék, idézetek, kiadvány , stb.

Ennek a nagyszerű API-nak az egyetlen hátránya a költség. A kötségmodell 50 dollárnál kezdődik 5000 keresés után, és 250 dollárra nő 30 000 API-hívás esetén, ez a Google Scholar API nagyon költséges lehetőség lehet, ha sok Google Scholar-adatra van szüksége.

Előnyök: Kiváló minőségű és könnyen használható Google Scholar API, amely visszaadja az összes szükséges információt.

Hátrányok: Drágább lehet nagyobb projekteknél, és nem testreszabható a saját projektje követelményeinek megfelelően.

* 1. [**SerpWow**](https://www.scraperapi.com/blog/best-google-scholar-apis-proxies/#serpwow)

Egy másik, harmadik féltől származó Google Scholar API-t biztosító vállalat a SerpWow. Bár nincs annyira dokumentálva, mint a SERP API Google Scholar API-ja, ez az API sok ugyanolyan funkcionalitással rendelkezik, valamivel olcsóbb áron.

Egyszerűen el lehet küldeni a keresési lekérdezést az API -juknak, és az összes Google Scholar keresési eredményt JSON formátumban adja vissza.

Az 5000 API-hívásért 45 dollárral kezdődő költségmodell nagyszerű megoldás, ha gyors és egyszerű módszert szeretnénk kinyerni a Google Scholar adataiból. Azonban a részletezett API megoldásokhoz hasonlóan ez is nagyon drága lehet, ha nagyobb mennyiségű adatra van szükség - 100 000 API hívás havonta 500 dollár.

Előnyök: Az adatokat JSON formátumban adja vissza, és valamivel olcsóbb, mint a SERP API.

Hátrányok: Nincs külön Google Scholar dokumentáció, és nagyon költséges.

* 1. [**Scale SERP**](https://www.scraperapi.com/blog/best-google-scholar-apis-proxies/#scaleserp)

A Scale SERP a következő Google Scholar API a listán. Bár a SerpWow-hoz hasonlóan néz ki, alacsonyabb áron kínálnak nagyon hasonló terméket.

Mivel a tervek havi 4 dollárért kezdődik 250 API-hívásért, és 10 millió API-hívássá növekednek 8500 dollárért, a Scale SERP mindenki számára kínál valamit.

A SerpWow-hoz és a SERP API-hoz hasonlóan ők is JSON formátumban adják vissza az adatokat, azonban adataik nem annyira részletesek. Csak a csupasz alapvető dolgokat részesíti előnyben, például a címet, a linket, a szerzőt és a részleteket. Bizonyos adatokat nem ad vissza, például az idézetek száma stb.

Előnyök: A legolcsóbb Google Scholar specifikus API, de még mindig legalább háromszor drágább, mint a Scraper API.

Hátrányok: Nem ad olyan részletes adatokat, mint a többi API, és nem testreszabható.

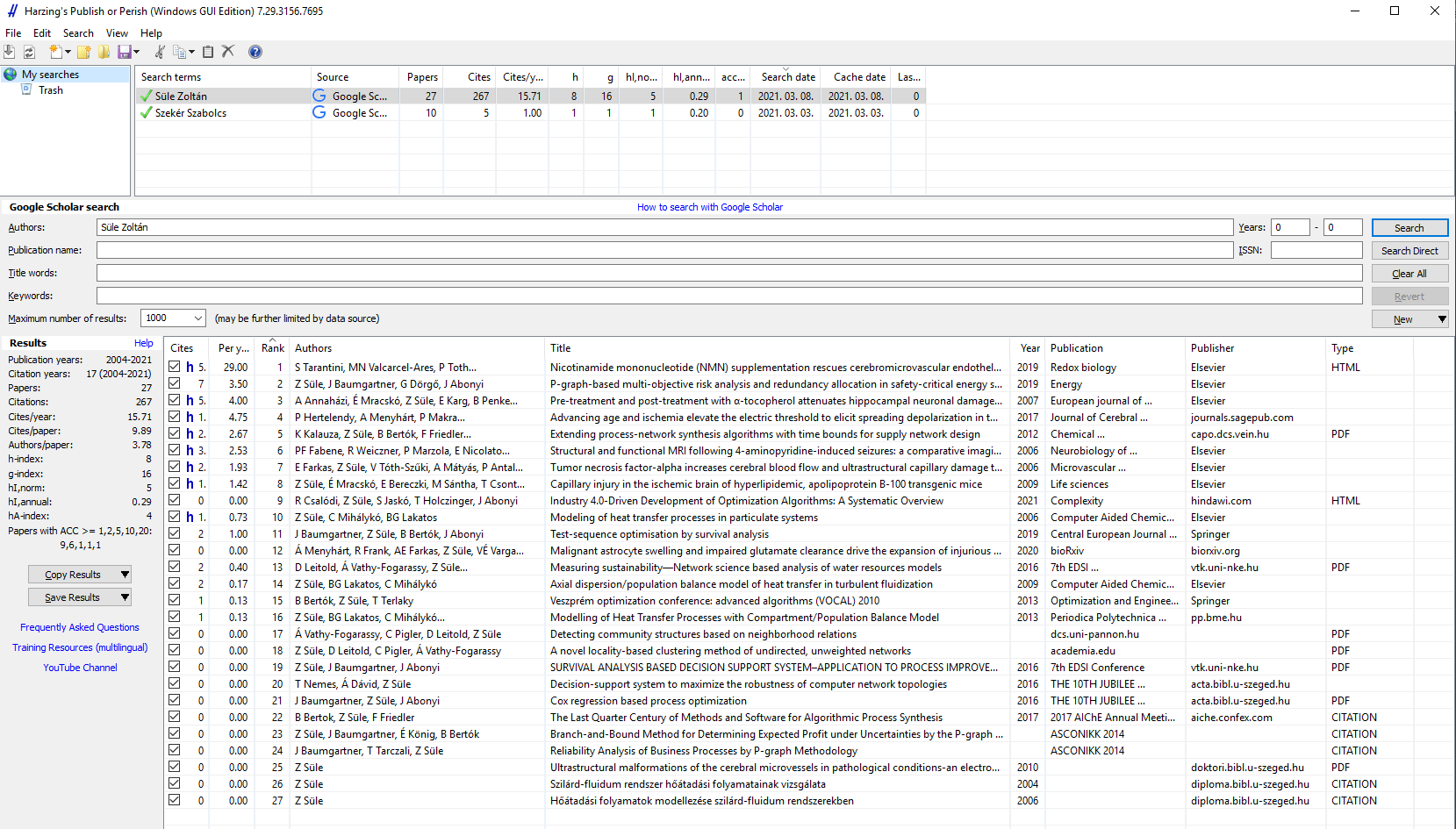
* 1. [**Publish or Perish**](https://www.scraperapi.com/blog/best-google-scholar-apis-proxies/#publishorperish)

A Publish or Perish kifejezetten a Google Scholar számára tervezett adatkivonó eszköz, amely lehetőséget nyújt saját Google Scholar API létrehozására.

Vigyázni kell azonban, mivel a szoftver kéréseket küld a Google Scholar-nak a számítógép IP-címének felhasználásával, így a Google letilthatja IP-címét, ha túl sok adatot próbálunk meg kinyerni ezzel az eszközzel. Tehát, ha több mint pár száz keresési eredményt kell kinyerniünk a Google Scholarból, akkor mindenképpen használnunk kell egy olyan proxy megoldást, mint pl. a Scraper API.

Előnyök: Teljesen ingyenes és könnyen használható.

Hátrányok: Fennáll annak a kockázata, hogy IP-címét letiltják, ha proxy használata nélkül használjuk.



* 1. **scholarly**

Előnyök:

Hátrányok:

1. **Web scraping technológia bemutatása**

**4.1 Web scraping fogalma**

Az angol szótár szerint a scraping jelentése kaparás. A szó szerinti jelentés nem is áll nagyon messze a technika lényegétől. A webscraping technológia az adatok kinyerését jelenti webhelyekről. A webscraper szoftver közvetlenül hozzáférhet a világhálóhoz a Hypertext Transfer Protocol vagy a webböngésző segítségével. Míg a webscraping-nek a felhasználója egy szoftver, webes adatok kinyerését manuálisan is elvégezhetünk. A webscarping kifejezés tipikusan robotok segítségével megvalósított automatizált folyamatokra utal. Ez egy olyan másolási forma, amelyben meghatározott adatokat gyűjtenek és másolnak az internetről, jellemzően egy központi vagy helyi adatbázisba vagy táblázatba későbbi visszakeresés vagy elemzés céljából.

A webscraping magában foglalja annak beolvasását és kibontását. A letöltés egy oldal letöltése, amelyet a böngésző akkor csinál, amikor a felhasználó megnéz egy oldalt. Ezért az internetes feltérképezés az internetes scraping egyik fő eleme, az oldalak későbbi feldolgozás céljából történő letöltéséhez. A letöltés után kivonás történhet. Az oldal tartalma elemezhető, kereshető, újra formázható, adatát táblázatba másolhatja, és így tovább. A webscraper-ek általában kivesznek valamit az oldalból, hogy más célra felhasználják valahol másutt. Példa lehet a nevek és telefonszámok, vagy a vállalatok és azok URL-jeinek megkeresése és egy listára másolása (névjegy-scraping).

A weboldalak szöveges jelölőnyelvek (HTML és XHTML) felhasználásával készülnek, és gyakran rengeteg hasznos adatot tartalmaznak szöveges formában. A legtöbb weboldal azonban az emberi végfelhasználók számára készült, és nem az automatizált használat megkönnyítése érdekében. Ennek eredményeként speciális eszközöket és szoftvereket fejlesztettek ki a weboldalak scrap-elésének megkönnyítésére.

A webscraping újabb formái magukban foglalják a webszerverekről érkező adatcsatornák meghallgatását. Például a JSON-t általában az ügyfél és a webkiszolgáló közötti szállítási tárolási mechanizmusként használják.

Vannak olyan módszerek, amelyeket egyes webhelyek megakadályoznak az internetes scrapingben, például észlelik és letiltják a robotok oldalainak feltérképezését. Válaszul vannak olyan webscraping rendszerek, amelyek a DOM-elemzés, a számítógépes látás és a természetes nyelv feldolgozásának technikáira támaszkodnak az emberi böngészés szimulálásához, hogy lehetővé tegyék a weboldal tartalmának összegyűjtését offline elemzéshez.

A webes scraping az a folyamat, amely során automatikusan bányásznak adatokat, vagy információkat gyűjtenek a világhálóról. Ez egy olyan terület, amelynek aktív fejlesztései közös célt szolgálnak a szemantikus webes jövőképpel, egy ambiciózus kezdeményezés, amelyhez még mindig áttörésekre van szükség a szövegfeldolgozásban, a szemantikai megértésben, a mesterséges intelligenciában és az ember-számítógép interakciókban. A jelenlegi webscraping megoldások az ad-hoc-tól kezdve, emberi erőfeszítéseket igényelnek, egészen a teljesen automatizált rendszerekig, amelyek képesek teljes weboldalakat strukturált információvá alakítani, korlátozásokkal.

**4.2 Web scraping története**

A világháló 1989-es születése után 1993 júniusában hozták létre az első webrobotot, a World Wide Web Wanderert, amelyet csak a web méretének mérésére szántak.

1993 decemberében elindult az első robot, amely webalapú keresőmotort, a JumpStation-t jelentette. Mivel nem volt olyan sok weboldal elérhető az interneten, a keresőmotorok abban az időben az emberi webhelyek rendszergazdáira támaszkodtak, hogy összegyűjtsék és szerkesszék a linkeket egy adott formátumban. Ehhez képest a JumpStation nagy előrelépést hozott, ez volt az első olyan WWW keresőmotor, amely egy webrobotra támaszkodott.

2000-ben jött az első webes API (Application Programming Interface) és API robot. Ez egy olyan felület, amely sokkal könnyebbé teszi a programfejlesztést építőelemek biztosításával. 2000-ben a Salesforce és az eBay elindította saját API-ját, amely lehetővé tette a programozók számára, hogy hozzáférjenek és letöltsék a nyilvánosság számára elérhető adatok egy részét. Azóta számos webhely kínál webes API-kat, hogy hozzáférjenek nyilvános adatbázisukhoz.

**4.3 Technikák**

**4.3.1 Emberi másolás-beillesztés**

A webscraping legegyszerűbb formája az adatok manuális másolása és beillesztése egy weboldalról egy szöveges fájlba vagy táblázatba. Néha még a legjobb webscraping technológia sem helyettesítheti az emberi vizsgálatot és a másolás és beillesztés funkciót, és néha ez lehet az egyetlen működőképes megoldás, amikor a kaparó weboldalak kifejezetten akadályokat állítanak a gépi automatizálás megakadályozása érdekében.

**4.3.2 Mintaszöveg egyezés**

Az információk weblapokról történő kinyerésének egyszerű, ugyanakkor hatékony megközelítése a UNIX grep parancson vagy a programozási nyelvek rendszeres kifejezés-illesztési szolgáltatásain (például Perl vagy Python) alapulhat.

**4.3.3 HTTP programozás**

A statikus és dinamikus weboldalakat úgy is le lehet kérni, hogy HTTP kéréseket küldünk a távoli webszerverre socket programozással.

**4.3.4 HTML elemzés (parsing)**

Sok webhely nagy oldalgyűjteménnyel rendelkezik, amelyek dinamikusan generálódnak egy mögöttes strukturált forrásból, például egy adatbázisból. Az azonos kategóriába tartozó adatokat tipikusan egy közös szkript vagy sablon kódolja hasonló oldalakra. Az adatbányászatban wrappernek (burkolónak) nevezzük azt a programot, amely egy adott információforrásban felismeri az ilyen sablonokat, kinyeri annak tartalmát és relációs formába fordítja. A wrappergeneráló algoritmusok feltételezik, hogy a wrapper indukciós rendszer bemeneti oldalai megfelelnek egy közös sablonnak, és könnyen azonosíthatók egy közös URL séma alapján. Sőt, néhány félig strukturált adatlekérdező nyelv, például az XQuery és a HTQL, is használható HTML-oldalak elemzésére, valamint az oldalak tartalmának lekérésére és átalakítására.

**4.3.4 DOM elemzés**

A teljes körű webböngésző, például az Internet Explorer vagy a Mozilla böngészőbe ágyazásával a programok lekérhetik az ügyféloldali szkriptek által generált dinamikus tartalmat. Ezek a böngészők vezérlik a weboldalakat DOM fává is, amelyek alapján a programok lekérhetik az oldalak egy részét. Olyan nyelvek, mint az Xpath, használhatók a kapott DOM-fa elemzésére.

**4.3.5 Függőleges összesítés**

Néhány vállalat kifejlesztett vertikum specifikus adatgyűjtési platformot. Ezek a platformok készítenek és figyelnek nagy mennyiségű botokat bizonyos különböző vertikumra ember nélkül és nem egy megjelölt céloldalra. Az előkészítés magában foglal egy tudásbázis felépítését az egész vertikumra és ezután a botok elkészülnek automatikusan. A platform robusztusságát a kinyert információk minőségével mérik (általában a mezők száma) és a skálázhatóságával (milyen gyorsan skálázható száz vagy ezernyi oldalakra). Ez a skálázhatóság gyakran az oldalak „hosszú farkát” (statisztikában az előfordulási grafikon domináló jobb oldalát) célozzák meg, a közös felhalmozást bonyolult vagy túl munkaigényes, hogy a tartalmat kibányásszák.

**4.3.6 Szemantikai annotáció felismerése**

A scrap-elt oldalak tartalmazhatnak metaadatokat, szemantikai jelöléseket és annotációkat, amik jelölhetnek bizonyos adattöredékeket. Ha az annotációk az oldalba vannak ágyazva (ahogy a Microformat-nál), akkor ez a technika a DOM elemzésnek egy különleges eseteként tekinthető. Más esetekben az annotációkat egy szemantikus rétegbe szervezik és ezeket a weboldalaktól külön tárolják és szervezik, tehát a scraper-ek le tudják kérdezni az adatok sémáját ebből a rétegből.

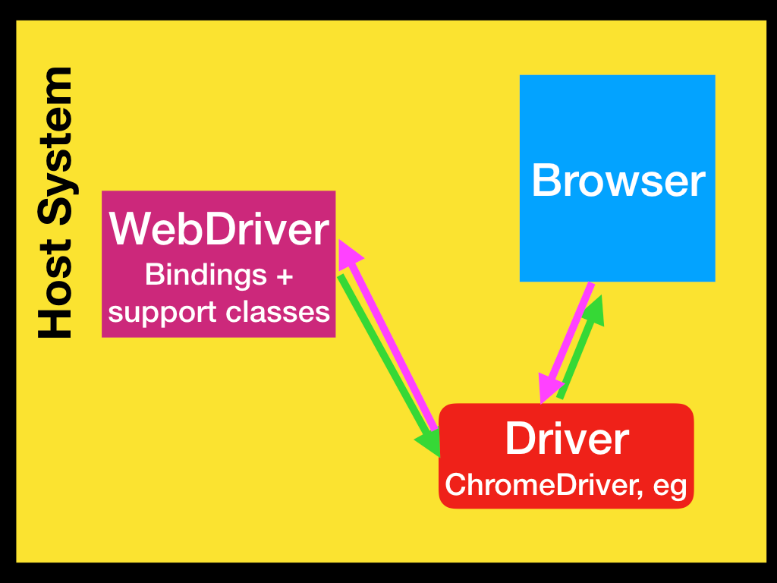
**4.3.7 Számítógépes látás – weboldal analízis**

A gépi tanulás és a számítógépes látás alkalmazásával olyan erőfeszítéseket tesznek, amelyek megkísérlik azonosítani és kinyerni az információkat a weboldalakról azáltal, hogy az oldalakat vizuálisan értelmezik, mint ember.

**4.4 Selenium**

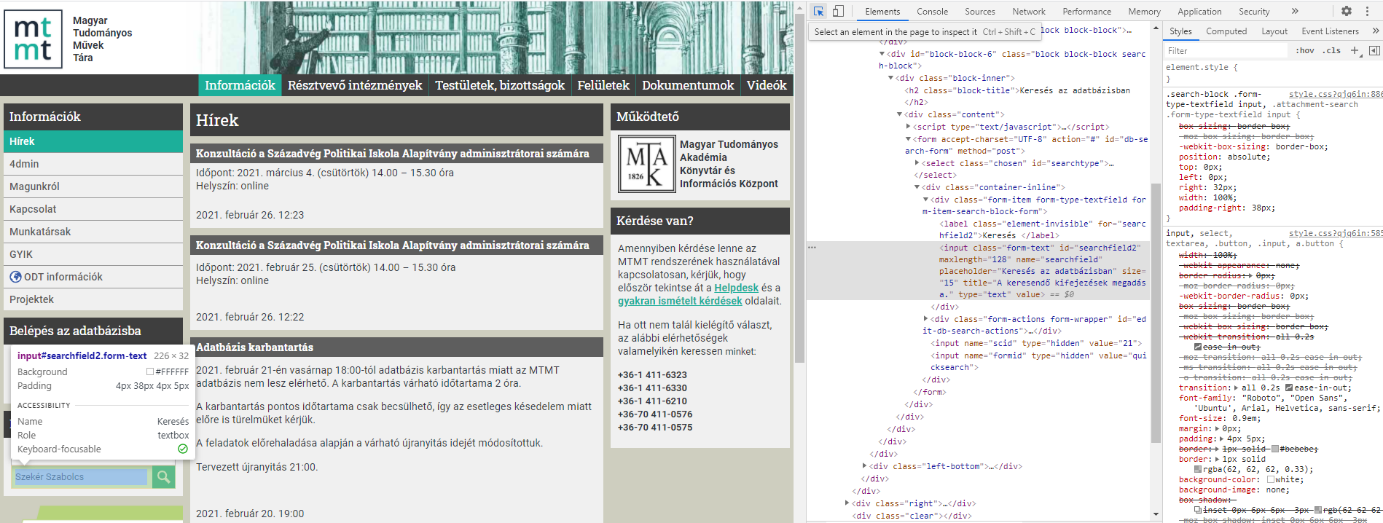
A Selenium egy összefoglaló „ernyő” projekt számos eszköz és könyvtár számára, amelyek lehetővé teszik és támogatják a webböngészők automatizálását. Bővítményeket kínál a felhasználói böngészőkkel való interakció utánzásához, amely lehetővé teszi kód írását az összes nagyobb webböngészőhöz. Ezt a projektet olyan önkéntes közreműködők teszik lehetővé, akik a nyílt forráskóddal mindenki számára szabadon hozzáférhetővé tették. A Selenium középpontjában a WebDriver található, amely egy utasításkészletek írására szolgáló felület, amely sok böngészőben felcserélhetően futtatható.

Felépítését a következő ábrán láthatjuk:



1. **Azonos feltételű kereső megvalósítása MTMT és Google Scholar rendszerekben**

Példa egy weboldal megfelelő részének megtalálásához a Chrome DevTools segítségével az MTMT oldalon:



1. **Adatok kigyűjtése, találatok megjelenítése**
2. **Találatok összehasonlítása és különbségek keresése**
3. **Automatikus/szoftverrel történő összehasonlítás előnye a manuálissal szemben**
4. **Adatok összehasonlítása és kiértékelése, különbségek felfedezése**
5. **Irodalomjegyzék**

<https://www.mtmt.hu/>

<https://scholar.google.com/>

<https://www.scraperapi.com/blog/best-google-scholar-apis-proxies/>

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

<https://www.selenium.dev/>