

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Automatinis duomenų surinkimas naršyklėmis be grafinės sąsajos

Web Scraping using Headless Browsers

Bakalauro baigiamasis darbas

Atliko:	Dominykas Ramanauskas	(parašas)
Darbo vadovas:	lekt. Aurimas Šimkus	(parašas)
Darbo recenzentas:	asist. dr. Tomas Plankis	(parašas)

Vilnius – 2023

Santrauka

Automatinis duomenų surinkimas tapo svarbia dalimi duomenimis pagrįstų sprendimų pasaulyje. Tačiau, dėl nuolat tobulėjančių internetinių technologijų, automatinis duomenų surinkimas tampa vis sunkesniu procesu: modernūs puslapiai duomenis krauna dinamiškai ar naudoja duomenų surinkimo prevencijos įrankius. Vienas galimų šios problemos sprendimo įrankių – naršyklės be grafinės sąsajos. Naudojantis naršyklėmis be grafinės sąsajomis, galima simuliuoti tikro vartotojo elgesį bei lengviau rinkti duomenis iš internetinių puslapių, naudojančių prevencijos įrankius. Tačiau, automatinės duomenų surinkimo sistemos, pagrįstos naršyklėmis be grafinės sąsajos turi ir didelį trūkumą – naudoti šias sistemas dideliu mastu yra brangu, nes tam reikia didelio kiekio resursų. Dėl šios priežasties, didžioji dalis tokių sistemų siūlomos kaip mokamos paslaugos.

Šiame darbe kuriama automatinio duomenų surinkimo sistema, pagrįsta būtent naršyklėmis be grafinės sąsajos. Kuriama sistema orientuota į lengvą naudojimą bei mažesnę blokavimų skaičių, gaunamų pagal naršyklės pirštų antspaudų tikrinimą. Darbo metu, apžvelgtas automatinio duomenų surinkimo procesas, išnagrinėti egzistuojantys naršyklių be grafinės sąsajos valdymo įrankiai bei suprojektuota kuriama automatinio duomenų surinkimo sistema. Naudojantis „Playwright“ naršyklių be grafinės sąsajos valdymo biblioteka, įgyvendinta suprojektuota sistema. Įvertinus sukurta sistemą, pastebėtas mažesnis gautų blokavimų bei CAPTCHA patikrinimų skaičius.

Raktiniai žodžiai: automatinis duomenų surinkimas, naršyklės be grafinės sąsajos, playwright

Summary

Web scraping has become an important part in the world of data-driven solutions. However, due to the continuous improvement of web technologies, web scraping becoming more and more difficult: modern pages load data dynamically or use web scraping prevention tools. One possible solution to this problem are headless browsers. The use of headless browsers makes it possible to simulate the behaviour of real users and to facilitate the collection of data from web pages that use prevention tools. However, web scrapers using headless browsers have a major drawback - they are costly to use on a large scale because of the large amount of resources required. For this reason, most of these types of systems are offered as paid services.

In this paper, a web scraping system based on headless browsers is being developed. The system developed focuses on ease of use and on reducing the number of blocking resulting from browser fingerprinting. In this work, the web scraping process is reviewed, existing tools for the management of headless browsers analysed, and a web scraping is designed. The designed system is then implemented using the headless browser management library called Playwright. The evaluation of the designed system showed a reduction in the number of received blocks and CAPTCHA checks.

Keywords: web scraping, headless browsers, playwright

TURINYS

IVADAS	6
1. AUTOMATINIO DUOMENŲ SURINKIMO APŽVALGA	8
1.1. Automatinio duomenų surinkimo metodai	8
1.1.1. Duomenų surinkimas HTTP užklausomis	8
1.1.2. Duomenų surinkimas naršyklėmis	9
1.1.3. Efektyvumo palyginimas tarp duomenų surinkimo HTTP užklausomis ir naršyk- lėmis	10
1.1.3.1. Palyginimo kriterijai	10
1.1.3.2. Palyginimo rezultatai	10
1.1.3.3. Palyginimo apibendrinimas	12
1.2. Automatinio duomenų surinkimo prevencijos būdai	12
1.2.1. Naršyklių pirštų antspaudai	12
1.2.2. Apribojimai IP adresams	14
1.2.2.1. Duomenų centrų tarpiniai serveriai	14
1.2.2.2. Gyventojų tarpiniai serveriai	14
1.2.3. CAPTCHA patikrinimai	14
1.3. Apibendrinimas	15
2. NARŠYKLIŲ BE GRAFINĖS SĄSAJOS VALDYMO ĮRANKIŲ APŽVALGA	16
2.1. „Selenium WebDriver“ apžvalga	16
2.2. „Playwright“ apžvalga	17
2.3. „Puppeteer“ apžvalga	17
2.4. Naršyklių be grafinės sąsajos valdymo įrankių palyginimas	18
2.4.1. Funkcionalumo palyginimas	18
2.4.2. Greičio palyginimas	19
2.4.3. Palyginimo apibendrinimas	20
2.5. Apibendrinimas	20
3. AUTOMATINIO DUOMENŲ SURINKIMO SISTEMOS PROTOTIPAS	21
3.1. Sistemos prototipo poreikis	21
3.1.1. Sistemos prototipo tikslas	21
3.2. Sistemos prototipo projektavimas	21
3.2.1. Sistemos prototipo veikimas	21
3.2.2. Sistemos funkciniai reikalavimai	22
3.2.3. Sistemos nefunkciniai reikalavimai	23
3.3. Sistemos prototipo įgyvendinimas	23
3.3.1. Aplikacijos programavimo sąsaja	24
3.3.2. Maskuojamų parametrų parinkimas	24
3.4. Apibendrinimas	25
4. SISTEMOS PROTOTIPO ĮVERTINIMAS	26
4.1. Vertinimui parinkti puslapiai	26
4.2. Vertinimo kriterijai	26
4.3. Duomenų surinkimo rezultatai	27
4.4. Apibendrinimas	28
REZULTATAI	29
IŠVADOS	30

ŠALTINIAI	31
SAVOKŲ APIBRĖŽIMAI	33
PRIEDAI	34
1 priedas. Nuorodos, naudotos sistemos prototipo vertinimui	34

Įvadas

Automatinis duomenų surinkimas (angl. *web scraping* arba *web crawling*) tapo svarbiu įrankiu duomenimis pagrįstų sprendimų pasaulyje, leidžiančiu iš internetinių puslapių išgauti vertingą informaciją. Šią svarbą lemia tai, kad internetas yra plati ir nuolat auganti įvairių duomenų saugykla. Automatinis duomenų surinkimas suteikia galimybę mokslininkams, rinkodaros specialistams, duomenų analitikams ir daugybei kitų specialistų naudotis naujausiais duomenimis rinkos tyrimams, tendencijų analizei, internetinių prekių kainų palyginimams, konkurentų stebėjimui ir dar daugiau [Khd21].

Internetinėms technologijoms nuolatos tobulėjant, automatinis duomenų surinkimas tampa vis sudėtingesniu procesu: modernūs puslapiai duomenis krauna dinamiškai ar naudoja duomenų surinkimo prevencijos įrankius. Vienas galimų šios problemos sprendimo įrankių – naršyklės be grafinės sąsajos (angl. *headless browser*). Automatinio duomenų surinkimo sistemos, pagrįstos naršyklėmis be grafinės sąsajos, išpopuliarėjo dėl galimybės išgauti duomenis iš šiuolaikinių dinamiškų svetainių, tačiau pasižymi ir kitu svarbiu bruožu – šie įrankiai leidžia imituoti tikro vartotojo sąveiką bei suteikia prieigą prie turinio, kuris paprastai pasiekiamas tik per interneto naršykles.

Tačiau, automatinės duomenų surinkimo sistemos, pagrįstos naršyklėmis be grafinės sąsajos turi ir didelį trūkumą – naudoti šias sistemas dideliu mastu yra brangu, dėl tam reikalaujamo didelio kiekio resursų. Dėl šios priežasties, tokių sistemų pasirinkimas rinkoje vis dar ribotas: dažniausiai šios sistemos siūlomos kaip mokamos paslaugos arba siūlomi tik įrankiai automatinio duomenų surinkimo kūrimui. Tai tampa didele problema, kai reikia paprastos automatinio duomenų surinkimo sistemos, neskiriant didelių resursų šių sistemų vystymui.

Šiame darbe kuriama automatinio duomenų surinkimo sistema, kuri siūlo pilnai paruoštą automatinio duomenų surinkimo naršyklėmis be grafinės sąsajos sprendimą, nereikalaujantį papildomų vystymo išlaidų. Kuriama sistema orientuota į lengvą naudojimą bei mažesnę blokažimų skaičių, gaunamų pagal naršyklės pirštų antspaudų tikrinimą. Šio darbo metu nagrinėjami automatinio duomenų surinkimo sistemų, naudojančių naršykles be grafinės sąsajos, privalumai bei trūkumai, lyginami egzistuojantys naršyklių be grafinės sąsajos valdymo įrankiai, projektuojamas bei įgyvendinamas automatinio duomenų surinkimo sistemos prototipas. Taip pat, siekiant įvertinti sistemos prototipo kokybę, vertinami automatinio duomenų surinkimo rezultatai, lyginant su įprasta automatinio duomenų surinkimo sistema.

Darbo tikslas – pasiūlyti ir įvertinti automatinio duomenų surinkimo sistemą, pagrįstą naršyklėmis be grafinės sąsajos.

Darbo uždaviniai:

1. Išanalizuoti skirtingus automatinio duomenų surinkimo būdus, siekiant įvertinti naršyklių be grafinės sąsajos pritaikymo privalumus bei trūkumus.
2. Apžvelgti pagrindinius automatinio duomenų surinkimo prevencijos būdus.

3. Išnagrinėti egzistuojančius naršyklių be grafinės sąsajos valdymo įrankius bei išrinkti vieną įrankį, kuris bus naudojamas duomenų surinkimo sistemos įgyvendinimui.
4. Suprojektuoti automatinio duomenų surinkimo sistemą, paremtą naršyklėmis be grafinės sąsajos.
5. Įgyvendinti sistemos prototipą bei įvertinti prototipo duomenų surinkimo galimybes.

1. Automatinio duomenų surinkimo apžvalga

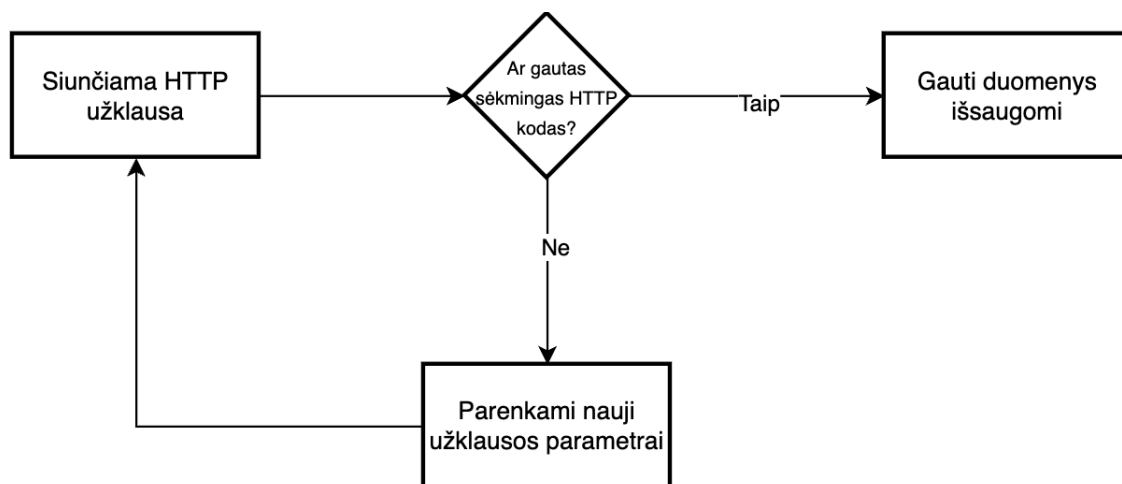
Šiame skyriuje apžvelgiamas automatinio duomenų surinkimo procesas bei palyginami du populiarius automatinio duomenų surinkimo būdai. Taip pat, šiame skyriuje susipažįstama su dažnai sutinkamais automatinio duomenų surinkimo prevencijos būdais. Skyrius skirtas išsiginėti į probleminę sritį.

1.1. Automatinio duomenų surinkimo metodai

Automatinis duomenų surinkimas – tai internetinių duomenų išgavimo procesas naudojant programinę įrangą. Dažniausiai, automatinis duomenų surinkimas vyksta naudojant hiperteksto perdavimo protokolą (angl. *Hypertext Transfer Protocol*) (HTTP) arba naudojant internetinę naršyklę [Zha17]. Dėl pastovaus kiekio nuolatos atsirandančių duomenų, automatinis duomenų surinkimas plačiai pripažįstamas kaip veiksmingas ir galingas būdas rinkti didelius kiekius duomenų. Nustatyta, kad naudojant automatinį duomenų surinkimą galima gauti daug išsamesnių, tikslesnių ir nuoseklesnių duomenų nei renkant juos rankiniu būdu [Khd21].

1.1.1. Duomenų surinkimas HTTP užklausomis

Vienas iš populiariausių automatinio duomenų surinkimo būdų – rezultatų traukimas tiesioginėmis užklausomis. Tai atliekama naudojant hiperteksto perdavimo protokolą, kurio pagalba siunčiamos užklauskos į tinklapio serverį bei gaunami atsakymai su duomenimis [Per19]. Iš esmės, tai yra tas pats metodas, kurį internetinės naršyklės naudoja puslapio turiniui pasiekti [Per19]. Tinklapiui sėkmingai gavus ir apdorojus užklausą, grąžinamas HTML, XML, JSON ar kito formato puslapio kodas, kuriame sudėta renkama informacija [Zha17].



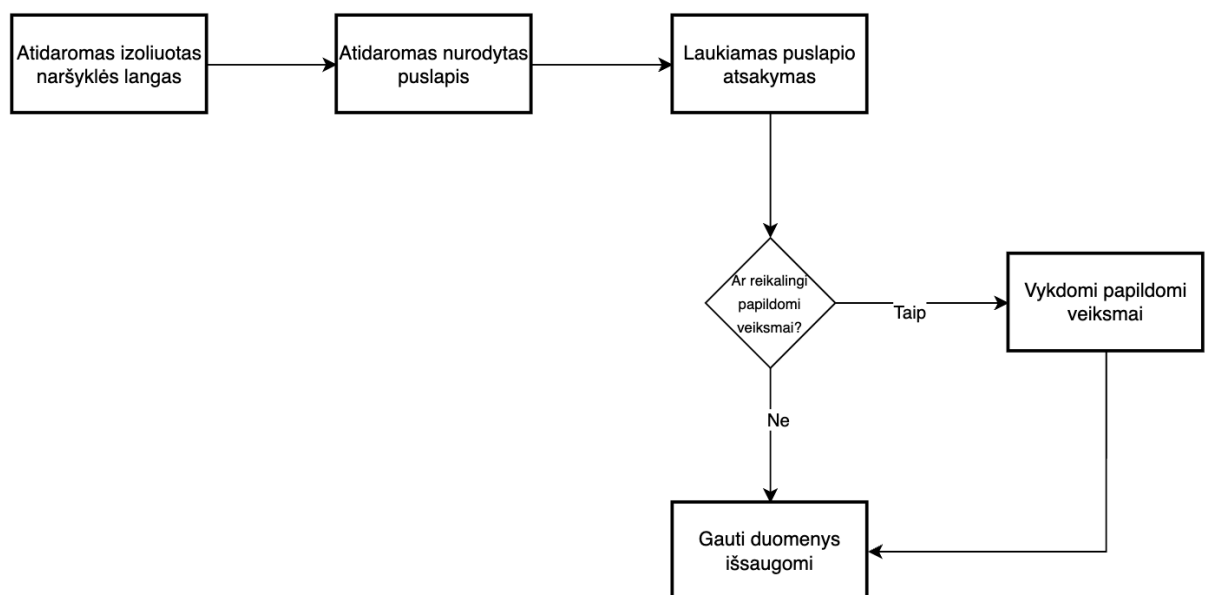
1 pav. Paprasto duomenų surinkimo HTTP užklausomis veikimo pavyzdys

Dėl tiesioginio kreipimosi į tinklapį, šis procesas yra ypač greitas ir nereikalauja didelių sąnaudų. Tačiau, sudėtingesniems tinklapiams, duomenis užkraunantiems vykdant JavaScript kodą kliento pusėje, šis metodas nėra tinkamas. Taip pat, nemažai modernių puslapių tikrina naršyklių

pirštų antspaudus (angl. *browser fingerprint*), kad apsisaugotų nuo automatinio duomenų rinkimo HTTP užklausomis [Zha17]. Tai reiškia, kad būtinas papildomų parametrų (pavyzdžiui, vartotojo agento ar slapukų) pateikimas kartu su užklausa. Nepateikus šių parametrų, siunčiamos užklausos dažniausiai yra blokuojamos ir grąžinami netinkami rezultatai ar CAPTCHA puslapiai.

1.1.2. Duomenų surinkimas naršyklėmis

Duomenų rinkimas naršyklėmis be grafinės sąsajos dažniausiai praverčia renkant duomenis iš internetinių puslapių, sukurtų naudojantis šiuolaikiniais JavaScript karkasais arba naudojančių papildomas apsaugos priemones, blokuojančias užklausas, kurios siunčiamos ne iš naršyklės [Scr22a]. Šio proceso metu, paleidžiama programinės įrangos valdoma internetinė naršyklė, tokia kaip „Google Chrome“ arba „Mozilla Firefox“, kuri leidžia vykdyti kodą kliento pusėje ir gauti tokius duomenis, kurie gaunami atidarius internetinį puslapį tikroje naršyklėje [Khd21]. Internetinės naršyklės naudojimas taip pat simuliuoja tikro vartotojo bruožus (vartotojo agentą, slapukų valdymą ir t. t.). Naudojant internetinę naršyklę galima atlikti ir papildomus veiksmus, pavyzdžiui, internetinėje parduotuvėje įdėti prekę į krepšelį ar nustatyti norimą vietos informaciją. Dažnu atveju, tokį veiksmą atlikti HTTP užklausomis neįmanoma, kadangi tai reikalauja daugiau nei paprasto užklausos siuntimo. Dėl šių priežasčių, internetiniams puslapiams yra sunkiau aptikti ar vartotojas yra tikras ir užklausos yra blokuojamos rečiau bei grąžinami kokybiškesni rezultatai.



2 pav. Paprasto duomenų surinkimo naršykle veikimo pavyzdys

Tačiau, vienas didžiausių automatinio duomenų surinkimo naršykle trūkumų yra tai, kad jas paleisti dideliu mastu yra brangu, kadangi tam reikia didelio kiekio resursų bei sunaudojama daug interneto srauto [Scr22a]. Ši problema tampa dar aktualesnė, kai tinklapiams randą būdą atskirti automatinio duomenų surinkimo sistemos siunčiamas užklausas nuo tikrų vartotojų siunčiamų užklausų. Tai leidžia tinklapiams apriboti prieigą duomenų surinkimo sistemoms, taip dar labiau pakeliant duomenų surinkimo kainą.

1.1.3. Efektyvumo palyginimas tarp duomenų surinkimo HTTP užklausomis ir naršyklėmis

Šio palyginimo tikslas yra išmatuoti praeitame poskyryje aptartų automatinio duomenų surinkimo metodų efektyvumą bei kokybę. Palyginimui pasirinkti du Vilniaus universiteto svetainės puslapiai:

- <https://mif.vu.lt> – pagrindinis Matematikos ir Informatikos fakulteto puslapis, kuriame skelbiamos fakulteto naujienos. Renkama informacija – straipsnių antraštės bei nuorodos.
- <https://tvarkarasciai.vu.lt>¹ – Vilniaus universiteto tvarkaraščių puslapis. Renkama informacija – paskaitų bei pratybų pavadinimai bei laikai.

Duomenų surinkimui naudojama Python programavimo kalba bei „requests“² biblioteka, skirta darbui su HTTP užklausomis. Antrajam metodui naudojama „Playwright“³ internetinių naršyklių valdymo biblioteka Python programavimo kalbai.

1.1.3.1. Palyginimo kriterijai

Palyginimui išskirti keli kriterijai:

- Vykdyimo laikas – duomenų surinkimo trukmė, nuo pradžios iki galutinio rezultato išsaugojimo į failą;
- Interneto srauto naudojimas – internetinio srauto sunaudojimas viso duomenų surinkimo metu;
- Gautų rezultatų kiekis – renkamų duomenų kiekis puslapyje (pvz. grąžintų knygų ar citatų kiekis), lyginant su esančių duomenų kiekiu.
- Blokavimų skaičius – skaičius gautų netinkamų atsakymų. Šių atsakymų rezultatai nebus įtraukiami į kitų kriterijų statistikas.

1.1.3.2. Palyginimo rezultatai

Rezultatų lentelėse palyginimui naudojamos šios nuorodos:

- Pagrindiniui MIF puslapiui – <https://mif.vu.lt>, rezultatų lentelėje trumpinama **MIF**.
- Tvarkaraščių puslapiui – <https://tvarkarasciai.vu.lt/mif/groups/programu-sistemas-4k-1gr-2022-2/?defaultDate=2022.12.05>, rezultatų lentelėje trumpinama **TVR**.

¹Palyginimui naudojamas ne pagrindinis puslapis, o tvarkaraščių puslapis įvedus studijų informaciją. Tikslu nuoroda pateikta prie palyginimo rezultatų.

²Šią biblioteką galima rasti <https://pypi.org/project/requests/>.

³„Playwright“ biblioteką Python programavimo kalbai galima rasti <https://playwright.dev/python/>.

1 lentelė. Duomenų surinkimo HTTP užklausomis rezultatai

Puslapis	Vykdyto laikas	Išnaudotas srautas	Duomenų kiekis	Blokavimai
MIF	0,238 sek.	68,88 KB	12/12 ⁴	0
TVR	0,18 sek.	27,11 KB	0/51	0

2 lentelė. Duomenų surinkimo naršykle rezultatai

Puslapis	Vykdyto laikas	Išnaudotas srautas	Duomenų kiekis	Blokavimai
MIF	2,13 sek.	2,05 MB	12/12	0
TVR	2,08 sek.	1,22 MB	51/51	0

Kaip matoma iš 1 ir 2 lentelių, duomenų surinkimas HTTP užklausomis užtruko apie dešimt kartų trumpiau ir sunaudojo mažiau interneto srauto. Tačiau, HTTP užklausomis nepavyko surinkti duomenų iš tvarkaraščių puslapio, kadangi tvarkaraščiai kraunami vykdant papildomą JavaScript kodą. Toliau pateiktuose 3 ir 4 paveiksluose pavaizduoti gauti HTML rezultatai po duomenų surinkimo.

Programų sistemos 4k. 1gr.

☐ Savaitgaliai

- [Spausdinti semestrą](#)
- [Atsiskaitymai](#)
- [Eksportuoti kalendorių](#)
-

[Dalis paskaitų yra paslėpta](#)



Rodyti pasirenkamuosius dalykus

[Vilniaus universitetas](#) | Tel. (8 5) 236 6200 arba el. p. pagalba@vu.lt | [ITPC](#) | [Privatumo politika](#) | [Prisijungti](#)

3 pav. Tvarkaraščio puslapio HTML rezultatas renkant HTTP užklausa

Kaip matoma 3 paveikslėlyje, reikalingi duomenys nebuvo užkrauti naudojantis HTTP užklausa, todėl šiam puslapiui būtinas duomenų rinkimas naršykle.

⁴Atidarius puslapį matoma tik 6 rezultatai, tačiau surinktuose rezultatuose galima rasti 12.

Programų sistemos 4k. 1gr.

[Bazė](#) | [Sąrašas](#) | [Diena](#)
— Savaitgaliai
Pagrindinės kalendoriaus funkcijos

- [Spausdinti savaitę](#)
- [Atidaryti](#)
- [Pasirinkti kalendorių](#)
- [Jei neturi sąrašo įrašų](#)

2022-12-05 – 09

Sav. 1 12-05 pr 12-06 an 12-07 tr 12-08 kt 12-09 pn

Vien
dienų

	08:30 - 10:00 Kodavimo teorija Pastrenkamasis 419 (MIF-DdL) kartu su IV k. INF studentais 08:30 - 10:00 Skaitiniai metodai Pastrenkamasis 213 (MIF-DdL) 10:15 - 12:00 Kodavimo teorija Pastrenkamasis 419 (MIF-DdL) 10:15 - 12:00 Matematinė analizė Pastrenkamasis 421 (MIF-DdL) 10:15 - 12:00				
08:30 - 10:00 Kombinatorika ir grafų teorija Pastrenkamasis 419 (MIF-DdL) 08:30 - 10:00 Transliavimo metodai Pastrenkamasis 310 (MIF-DdL) 10:15 - 12:00 Kombinatorika ir grafų teorija Pastrenkamasis 419 (MIF-DdL) 10:15 - 12:00 Loginis programavimas Pastrenkamasis 109B (MIF-DdL) 10:15 - 12:00 Transliavimo metodai Pastrenkamasis 310 (MIF-DdL) 12:00 - 14:00 Kombinatorika ir grafų teorija Pastrenkamasis 419 (MIF-DdL) 12:00 - 14:00 Loginis programavimas Pastrenkamasis 109B (MIF-DdL) 14:00 - 16:00 Programų kūrimo procesas 414 (MIF-DdL) 17:45 - 19:15 Blokų grandinių technologijos Pastrenkamasis Vyksta nuotoliniu būdu 19:15 - 20:45 Blokų grandinių technologijos Pastrenkamasis Vyksta nuotoliniu būdu	Matematinis modeliavimas Pastrenkamasis 311 (MIF-Naug.) 10:15 - 12:00 Optimizavimo metodai Pastrenkamasis 323 (MIF-DdL) 10:15 - 12:00 Skaitiniai metodai Pastrenkamasis 308 (MIF-DdL) 10:15 - 12:00 Statistinė duomenų analizė Pastrenkamasis 300 (MIF-Naug.) 12:00 - 14:00 Matematinė analizė Pastrenkamasis 421 (MIF-DdL) 12:00 - 14:00 Matematinis modeliavimas Pastrenkamasis S-103 (MIF-Salt) 12:00 - 14:00 Optimizavimo metodai Pastrenkamasis 323 (MIF-DdL) 12:00 - 14:00 Statistinė duomenų analizė Pastrenkamasis S-102 (MIF-Salt) 14:00 - 16:00 Kodavimo teorija Pastrenkamasis 419 (MIF-DdL) 16:00 - 18:00 Skaitmeninis intelektas ir sprendimų priėmimas Pastrenkamasis 102 (MIF-DdL) kartu su 3 kurso ISI, DM ir 4 kurso DM ir EKO studentais 16:00 - 17:30 Elektroninės komercijos technologijų pagrindai Pastrenkamasis 213 (MIF-DdL) 17:45 - 19:15 Elektroninės komercijos technologijų pagrindai Pastrenkamasis Pograpių: 2 109A - Iešvojimų erdvė (MIF-DdL) 18:00 - 20:00 Skaitmeninis intelektas ir sprendimų priėmimas Pastrenkamasis 308 (MIF-DdL) 18:00 - 19:30 Blokų grandinių technologijos Pastrenkamasis Vyksta nuotoliniu būdu	08:30 - 10:00 Lygiagretusis programavimas Pastrenkamasis S-206 (MIF-Salt) 08:30 - 10:00 Programavimas Windows API Pastrenkamasis 308 (MIF-DdL) 10:15 - 12:00 Informacinės sistemos Pastrenkamasis 310 (MIF-DdL) 08:30 - 10:00 Kompiuterinė grafika Pastrenkamasis 303 (MIF-DdL) 10:15 - 12:00 Informacinės sistemos Pastrenkamasis 310 (MIF-DdL) 10:15 - 12:00 Kompiuterinė grafika Pastrenkamasis 303 (MIF-DdL) 12:00 - 14:00 Optimizavimo metodai Pastrenkamasis 102 (MIF-DdL) 14:00 - 16:00 Dirbtinis intelektas Pastrenkamasis 419 (MIF-DdL) 16:00 - 18:00 Dirbtinis intelektas Pastrenkamasis 419 (MIF-DdL) 18:00 - 20:00 Dirbtinis intelektas Pastrenkamasis 419 (MIF-DdL)	08:30 - 10:00 Lygiagretusis programavimas Pastrenkamasis S-206 (MIF-Salt) 08:30 - 10:00 Programavimas Windows API Pastrenkamasis 308 (MIF-DdL) 10:15 - 12:00 Kompiuterinių žaidimų projektavimas ir kūrimas Pastrenkamasis 310 (MIF-DdL) 418 (MIF-DdL) 10:15 - 12:00 Lygiagretusis programavimas Pastrenkamasis S-206 (MIF-Salt) 10:15 - 12:00 Programavimas Windows API Pastrenkamasis 102 (MIF-DdL) 12:00 - 16:00 Kompiuterinių tinklų profesionalums II Pastrenkamasis 303 (MIF-DdL) 303 (MIF-DdL) 12:00 - 14:00 Kompiuterinių žaidimų projektavimas ir kūrimas Pastrenkamasis 109A - Iešvojimų erdvė (MIF-DdL) 12:00 - 14:00 Loginis programavimas Pastrenkamasis 103 (MIF-DdL) 12:00 - 14:00 Programavimas Windows API Pastrenkamasis 308 (MIF-DdL) 14:00 - 16:00 Loginis programavimas Pastrenkamasis 109B (MIF-DdL) 16:00 - 20:00 Bioinformatika Pastrenkamasis 303 (MIF-DdL) 16:00 - 18:00 Loginis programavimas Pastrenkamasis 109B (MIF-DdL)	08:30 - 10:00 Programų sistemų projektų ir kokybės valdymas 300 (MIF-Naug.) 10:15 - 12:00 Programų sistemų projektų ir kokybės valdymas 101 (MIF-Naug.) 12:00 - 14:00 Programų kūrimo procesas 101 (MIF-Naug.)	

4 pav. Tvarkaraščio puslapio HTML rezultatas renkant naršykle

1.1.3.3. Palyginimo apibendrinimas

Iš atlikto palyginimo, galima pastebėti, kad duomenų surinkimas HTTP užklausomis yra greitesnis bei pigesnis variantas, tačiau negarantuoja gražintų duomenų. Nors duomenų surinkimas naršykle ir reikalauja daugiau resursų, rezultatai yra labai panašesni į tuos, kurie gaunami atidarius puslapį savoje naršyklėje.

1.2. Automatinio duomenų surinkimo prevencijos būdai

Nemaža dalis modernių tinklapių naudoja vieną ar kelis automatinio duomenų surinkimo prevencijos įrankius, apsunkinančius duomenų surinkimą. Šiame poskyryje apžvelgiami populiariausi prevencijos būdai bei aptariami galimi jų vengimo būdai.

1.2.1. Naršyklių pirštų antspaudai

Automatinio duomenų surinkimo prevencija pagal naršyklės piršto antspaudą yra viena iš labiausiai paplitusių priemonių apsaugoti nuo duomenų surinkimo. Naršyklės pirštų antspaudų

tikrinimas – tai tinklapio lankytojų identifikavimo procesas, renkant tam tikrus duomenis iš įrenginių, HTTP jungčių ir programinės įrangos funkcijų [Zen23a].

Šio proceso metu, tikrinama tokia informacija kaip: [Zen23a]

- vartotojo agentas (angl. *user-agent*);
- operacinės sistemos tipas bei kalba;
- naršyklės tipas, versija ir papildiniai;
- laiko zona;
- naršyklės kalba bei šriftai;
- naršyklės navigatoriaus parametrai;
- ekrano rezoliucija;
- ir kita.

Nors aukščiau pateikta informacija yra bendra, labai retai pasitaiko, kad dviejų naudotojų duomenys sutaptų šimtu procentų [Zen23a]. Taipogi, pateikti šią informaciją renkant duomenis HTTP užklausa yra praktiškai neįmanoma, kadangi ši informacija dažniausiai tikrinama vykdant JavaScript kodą kliento pusėje. Tačiau, galima nesunkiai pateikti savo nurodytas HTTP antraštes (angl. *HTTP headers*) ir vartotojo agentų eilutes. Skirtingi HTTP užklausų siuntimo varikliai naudoja savo formato numatytas vartotojo agento eilutes. Pavyzdžiui, aplikacijų programavimo sąsajų testavimo aplikacija „Postman“ siunčia „PostmanRuntime/{versija}“ formato vartotojo agento eilutę, pagal kurią nesunkiai galima atskirti, kad užklausa siųsta ne iš internetinės naršyklės, o naudojantis būtent „Postman“ programa.

	Key	Value
☒	Postman-Token	<calculated when request is sent>
☒	Host	<calculated when request is sent>
☒	User-Agent	PostmanRuntime/7.32.2
☒	Accept	
☒	Accept-Encoding	
☒	Connection	
	Key	

We recommend using this header

The User-Agent header is added to help the server identify Postman as the HTTP requesting application or client.

It is recommended that this header be sent, but you can remove the header or enter a new one with a different value.

5 pav. „Postman“ aplikacijos siunčiamos HTTP antraštės (anglų k.)

5 paveikslėlyje pavaizduota „Postman“ aplikacijos siunčiamos numatytosios HTTP antraštės. Kaip matoma iš paveikslėlio, ši antraštė skirta HTTP užklausos identifikavimui, tačiau, ji gali būti pašalinta siunčiant užklausa. Neretai, antraščių pakeitimas suteikia galimybę apeiti duomenų

surinkimo prevenciją, jei internetinis puslapis nereikalauja papildomo JavaScript kodo vykdymo. Dažnas pasirinkimas duomenų surinkimo sistemose – populiarių naršyklių HTTP antraštės.

1.2.2. Apribojimai IP adresams

Interneto protokolo (IP) adresas – tai dažniausiai pasitaikantis būdas identifikuoti duomenų surinkimo sistemas [Scr22b]. IP adreso sekimas yra dažnai naudojamas siekiant apriboti arba užblokuoti automatinio duomenų surinkimo sistemas ar kitus nepageidaujamus lankytojus [Scr22b]. IP adresai gali būti apribojami pagal keletą kriterijų: IP adreso tipą (pvz. duomenų centro IP adresai), padarytų užklausų kiekį ar net IP adreso potinklį. Norint išvengti apribojimų, duomenų surinkimo sistemos privalo naudoti tarpinius serverius (angl. *proxy server*). Tarpiniai serveriai klasifikuojami į du pagrindinius tipus: duomenų centrų ir gyventojų [Scr22b].

1.2.2.1. Duomenų centrų tarpiniai serveriai

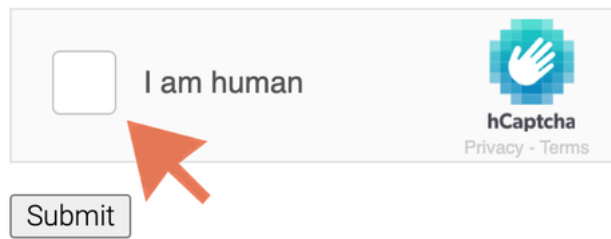
Duomenų centrų tarpiniai serveriai – tai tarpiniai serveriai, kurie nėra susieti su interneto paslaugų tiekėju [Mot21]. Duomenų centrų tarpiniai serveriai priklauso duomenų centrams ir pardodami keliems klientams vienu metu, todėl juos galima lengvai aptikti ir pritaikyti apribojimus.

1.2.2.2. Gyventojų tarpiniai serveriai

Gyventojų tarpiniai serveriai naudoja tikrų gyventojų IP adresus, kuriuos klientams suteikia interneto paslaugų teikėjai [Mot21]. Kadangi šie IP adresai prijungti prie realių įrenginių, juos naudojant nesunku simuliuoti tikrą vartotoją [Mot21]. Dažnu atveju, automatiniam duomenų surinkimui rekomenduojama naudoti gyventojų tarpinius serverius [Scr22b].

1.2.3. CAPTCHA patikrinimai

Dar vienas populiarius automatinio duomenų surinkimo prevencijos būdas – CAPTCHA (angl. *Completely Automated Public Turing test to tell Computers and Humans Apart*). CAPTCHA – tai užduotys, pateikiamos prieš patekimą į norimą puslapį. Šios užduotys gali būti įvairių formų: nuo paprasto mygtuko paspaudimo iki nuotraukų klasifikavimo ar teksto įvesties. Internetiniai puslapiai šias užduotis naudoja norėdami nustatyti ar ateinantis vartotojas yra žmogus ar robotas, vertindami galutinių užduočių rezultatų tikslumą bei užtrukusį laiką [Zen23b]. CAPTCHA patikrinimai turi mažiau nei 10 % nesėkmės atvejų bei kasmet tobulėja, taip tampanti viena patikimiausių kovos su robotais priemonių [Zen23b].



6 pav. CAPTCHA pavyzdys (anglų k.) [Zen23b]

CAPTCHA patikrinimai pateikiami pagrindė šiais trimis atvejais: [Zen23b]

- Neįprastas srauto padidėjimas iš vieno vartotojo trumpu laiko tarpu.
- Žmogui neįprasti arba robotams panašūs veiksmai, pavyzdžiui, visiškai nejudinama pelytė.
- Atsitiktinai, kai automatinio duomenų surinkimo prevencijos sistema arba ugniasienė tikrina dėl papildomo saugumo sumetimų.

Nors CAPTCHA užduotis galima bandyti spręsti naudojantis programine įranga, pats geriausias būdas susidoroti su CAPTCHA – jų vengimas. Kiekviena CAPTCHA patikrinimo sistema yra skirtinga ir jų sprendimas dažniausiai yra sudėtingas bei reikalauja daug laiko. Todėl, kuo didesnis tikro vartotojo bruožų simuliacijimas – pats geriausias būdas rinkti duomenis iš internetinio puslapio, naudojančio CAPTCHA.

1.3. Apibendrinimas

Apžvelgus bei palyginus du populiariausius automatinio duomenų surinkimo metodus, duomenų surinkimą HTTP užklausomis ir duomenų surinkimą naršyklėmis, matomi aiškūs panaudojimo atvejai abiem būdams. Automatinis duomenų surinkimas HTTP užklausomis turėtų būti naudojamas kaip pigus ir efektyvus duomenų surinkimo sprendimas, kuomet nėra tikimasi, kad šis būdas bus veiksmingas visiems atvejams. Kita vertus, automatinis duomenų surinkimas naršyklėmis yra brangesnis, tačiau patikimesnis bei kokybiškesnis būdas duomenims rinkti. Taip pat, išskirti trys esminiai automatinio duomenų surinkimo prevencijos būdai: naršyklės antspausdų tikrinimas, apribojimai IP adresams bei CAPTCHA patikrinimai.

2. Naršyklių be grafinės sąsajos valdymo įrankių apžvalga

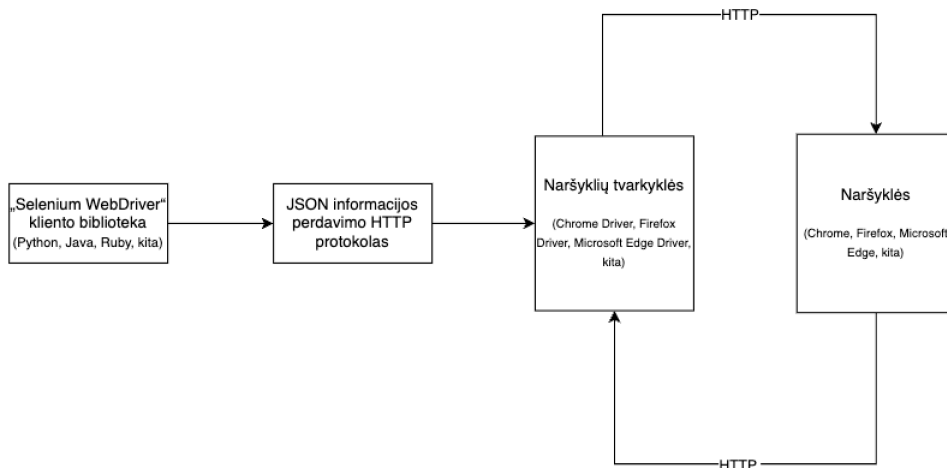
Šiame skyriuje apžvelgiami bei lyginami trys populiarius naršyklių be grafinės sąsajos valdymo programine įranga įrankiai. Šis skyrius skirtas išsiginčyti į kiekvieną įrankį ir išsirinkti vieną iš įrankių automatinio duomenų surinkimo sistemos prototipo kūrimui.

2.1. „Selenium WebDriver“ apžvalga

„Selenium WebDriver“ - tai biblioteka, leidžianti valdyti skirtingas naršyklės, naudojantis programine įranga. Šis įrankis sukurtas automatizuoti internetinių aplikacijų testavimą, siekiant patikrinti, ar jos veikia tinkamai. Naršyklių valdymui „Selenium WebDriver“ naudoja kūrėjų pateiktas naršyklės programavimo sąsajas (API), suteikiančias galimybę simuliuoti tikro vartotojo veiksmus [Sel22].

„Selenium WebDriver“ architektūrą sudaro keturi pagrindiniai komponentai: [Sin21]

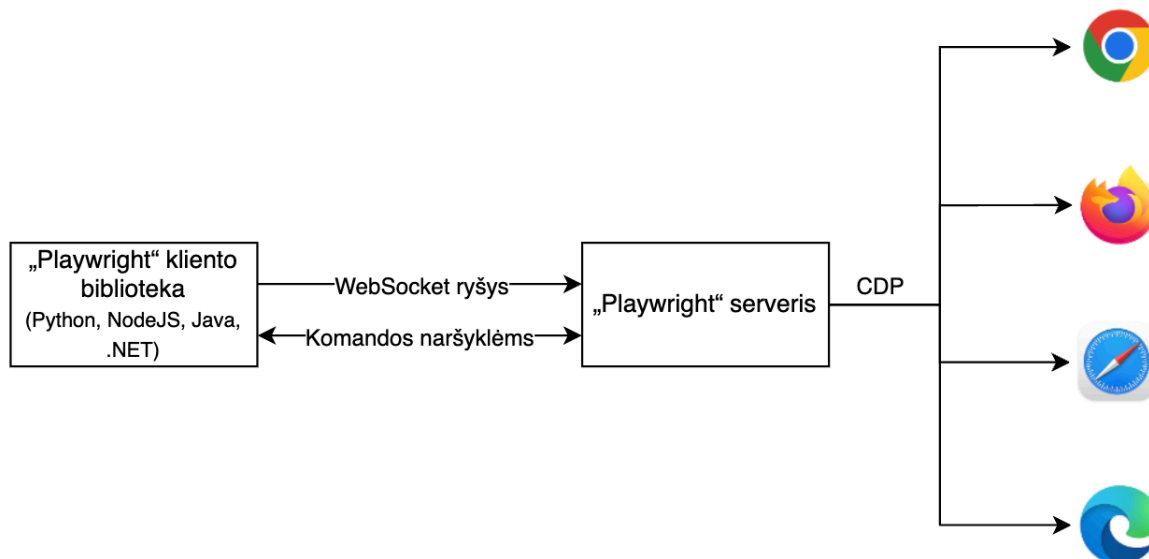
- „Selenium WebDriver“ kliento biblioteka / kalbų sąsajos – „Selenium WebDriver“ architektūra palaiko įvairias programavimo kalbas, pavyzdžiui, Java, C#, Python, Ruby, PHP ir kt.
- JSON informacijos siuntimo per HTTP protokolas (angl. *JSON Wire Protocol*) – tai duomenų perdavimo protokolas, naudojantis RESTful (angl. *Representational State Transfer*) programavimo sąsajas, kurio pagalba suteikiama galimybė dalintis JSON informacija HTTP protokolu.
- Naršyklių tvarkyklės (angl. *browser drivers*) – „Selenium“ kiekvienai naršyklei pateikia specifines W3C standarto tvarkyklės, neatskleidžiant vidinių naršyklių funkcijų logikos. Naršyklės tvarkyklė sąveikauja su atitinkama naršykle siunčiamą informaciją anksčiau minėtu JSON siuntimo per HTTP protokolu.
- Tikros naršyklės – vietiniame įrenginyje įdiegtos naršyklės, kuriomis „Selenium“ bendrauja per naršyklių tvarkyklės.



7 pav. „Selenium WebDriver“ veikimo diagrama [Sin21; Una23]

2.2. „Playwright“ apžvalga

„Playwright“ yra 2020 m. išleista atviro kodo internetinių aplikacijų testavimo biblioteka, vystoma įmonės „Microsoft“. Užklausas tarp kliento ir serverio „Playwright“ perduoda per vieną WebSocket ryšį, kuris išlieka tol, kol baigiamas programos vykdymas [Lam23; Pro22]. Tai leidžia greitai siųsti komandas per vieną ryšį ir sumažina nesėkmės taškų (angl. *point of failure*) kiekį [Lam23].



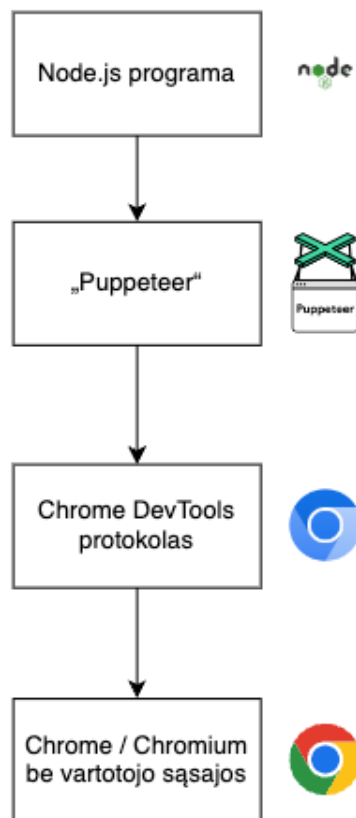
8 pav. „Playwright“ veikimo diagrama [Pro22]

„Playwright“ biblioteka naudoja Chrome DevTools protokolą⁵ (CDP), kuris suteikia galimybę valdyti naršyklę per aplikacijų programavimo sąsają. Taip pat, „Playwright“ biblioteka siūlo galingas naršyklės konteksto funkcijas, leidžiančias imituoti kelis įrenginius naudojant vieną egzempliorių [Lam23]. Kiekviename naršyklės kontekste galima valdyti kelis puslapius ir kiekvienas puslapis izoliuotas nuo kitų kontekstų, slapukų ir vietinės saugyklos požiūriu.

2.3. „Puppeteer“ apžvalga

„Puppeteer“ yra viena iš labiausiai naudojamų bibliotekų automatinio duomenų surinkimo rinkoje. Tai yra atvirojo kodo Node.js biblioteka, sukurta bei palaikoma įmonės „Google“. „Puppeteer“ suteikia aukšto lygio aplikacijų programavimo sąsają Chrome bei Chromium aplikacijoms valdyti [Pup23]. Panašiai kaip „Playwright“, „Puppeteer“ biblioteka naršyklių valdymui naudoja Chrome DevTools protokolą [Pup23].

⁵Daugiau informacijos apie Chrome DevTools galima rasti čia: <https://chromedevtools.github.io/devtools-protocol/>



9 pav. „Puppeteer“ veikimo diagrama [Sub21]

Priešingai nei „Selenium WebDriver“ ar „Playwright“, kurios veikia su dauguma naršyklių ir programavimo kalbų, „Puppeteer“ biblioteka susitelkia tik į Chrome / Chromium ir aktyviai vystoma tik JavaScript programavimo kalbai [Pup23].

2.4. Naršyklių be grafinės sąsajos valdymo įrankių palyginimas

Šiame poskyryje atliekami išnagrinėtų naršyklių be grafinės sąsajos įrankių funkcionalumo bei greičio palyginimai.

2.4.1. Funkcionalumo palyginimas

3 lentelėje pateikiamas funkcionalumo palyginimas tarp „Selenium WebDriver“, „Playwright“ bei „Puppeteer“ naršyklių be grafinės sąsajos valdymo įrankių. Lyginamas bendrinis įrankių funkcionalumas: palaikomos naršyklės, programavimo kalbos, architektūra ir t. t.

3 lentelė. Funkcionalumo palyginimas tarp „Selenium WebDriver“, „Playwright“ bei „Puppeteer“

Funkcionalumas	„Selenium WebDriver“	„Playwright“	„Puppeteer“
Naršyklių palaikymas	Chrome / Chromium, Firefox, Edge, Safari	Chrome / Chromium, Firefox, Edge, Safari	Chrome / Chromium ⁶
Programavimo kalbų palaikymas	JavaScript (Node.js), Python, Ruby, Java, Kotlin, C#	TypeScript, JavaScript (Node.js), Python, .NET, Java.	JavaScript (Node.js)
Naršyklių valdymas	Naudojamos naršyklių tvarkyklės	Naudojamas Chrome DevTools protokolas	Naudojamas Chrome DevTools protokolas
Architektūra	Komunikuojama HTTP užklausomis	Reaguojama į įvykius (angl. <i>event-driven architecture</i>)	Reaguojama į įvykius
Tikrų įrenginių naudojimas	Galima naudoti tikrus įrenginius	Tikri įrenginiai nepalaikomi, galima tik emuluoti	Tikri įrenginiai nepalaikomi, galima tik emuluoti

2.4.2. Greičio palyginimas

Greičio palyginimui naudojama automatinio duomenų surinkimo smėlio dėžė⁷ (angl. *sandbox*). Palyginimui naudojami trys smėlio dėžės puslapiai:

- <http://quotes.toscrape.com/js> – skirta duomenų, užkraudų JavaScript vykdomu kodu, rinkimui. Lentelėje žymima **JS**.
- <http://quotes.toscrape.com/login> – skirta prisijungimo puslapiui, kuriame reikia atlikti papildomus veiksmus, testavimui. Lentelėje žymima **LOGIN**.
- <http://quotes.toscrape.com/js-delayed> – skirta uždelstų duomenų, kraunamų JavaScript vykdomu kodu. Palyginime bus naudojamas dviejų sekundžių uždelsimas ir matuojama kaip greitai grąžinama informacija po atsiradimo. Lentelėje žymima **JS-DELAYED**.

⁶Kitų naršyklių palaikymas „beta“ stadijoje.

⁷Smėlio dėžę galima rasti čia: <https://toscrape.com>.

Kiekvienas puslapis bandomas rinkti 10 kartų, lentelėje surašytas rezultato gavimo greičio vidurkis. Visiems duomenims surinkti naudojama naujausios versijos Google Chrome internetinė naršyklė. „Selenium WebDriver“ bei „Playwright“ duomenys renkami Python programavimo kalba, „Puppeteer“ – JavaScript.

4 lentelė. Greičio palyginimas tarp „Selenium WebDriver“, „Playwright“ ir „Puppeteer“

Puslapis	„Selenium WebDriver“	„Playwright“	„Puppeteer“
JS	3,58 sek.	2,30 sek.	2,65 sek.
LOGIN	4,29 sek.	2,60 sek.	2,81 sek.
JS-DELAYED	5,86 sek.	4,29 sek.	4,51 sek.

2.4.3. Palyginimo apibendrinimas

Pagal 3 lentelę, galima pastebėti, jog „Puppeteer“ siūlo labai mažai pasirinkimo: specializuojasi tik į Chrome / Chromium vystymą bei palaiko tik JavaScript programavimo kalbą. „Selenium WebDriver“ bei „Playwright“ siūlo pakankamai didelį pasirinkimą programavimo kalbų ir palaiko beveik visas populiariausias naršykles. Iš 4 lentelėje pavaizduotų greičio rezultatų, matoma, kad, dėl panašios architektūros, „Playwright“ bei „Puppeteer“ atliko duomenų surinkimą mažiau nei pusės sekundės skirtumu, tačiau „Selenium WebDriver“ atsiliko visa sekunde ar daugiau. Taip pat, prisijungimo puslapyje „Selenium WebDriver“ sulėtėjo net dvigubai daugiau nei „Playwright“ ar „Puppeteer“.

2.5. Apibendrinimas

Išnagrinėjus bei palyginus tris populiarius naršyklių be grafinės sąsajos valdymo įrankius, „Selenium WebDriver“, „Playwright“ bei „Puppeteer“, matomas aiškus pasirinkimas prototipo kūrimui – „Playwright“. Priešingai nei „Puppeteer“, šis įrankis siūlo didelį naršyklių bei programavimo kalbų pasirinkimą, bei greičio palyginime lenkia „Selenium WebDriver“, atliekant tuos pačius automatinio duomenų surinkimo veiksmus.

3. Automatinio duomenų surinkimo sistemos prototipas

Šiame skyriuje aprašomas kuriamas automatinio duomenų surinkimo sistemos prototipas, sudaromi funkciniai ir nefunkciniai reikalavimai bei aptariamas įgyvendinimas.

3.1. Sistemos prototipo poreikis

Automatinio duomenų surinkimo įrankių pasirinkimas nėra mažas: paprasta internetinė paieška gali grąžinti kelis šimtus ar tūkstančius variantų, tinkančių visiems panaudojimo atvejams. Tačiau, su šiais įrankiais dažniausiai kyla viena problema – kaina. Didžioji dalis šių įrankių tiekėjų nemokamą naudojimą riboja arba siūlo tik duomenų surinkimo sistemų kūrimo įrankius. Todėl, šiame darbe kuriamas sistemos prototipas orientuotas į lengvą bei kokybišką automatinį duomenų surinkimą, nereikalaujantį papildomų išlaidų vystymui.

3.1.1. Sistemos prototipo tikslas

Vienas didžiausių iššūkių automatinio duomenų surinkimo sistemoms – duomenų surinkimo prevencijos vengimas. Remiantis anksčiau nagrinėta medžiaga, išskirti trys esminiai prevencijos būdai: naršyklės pirštų antspaudų tikrinimas, apribojimai IP adresams bei CAPTCHA patikrinimai. Šia sistema siekiama susitelkti kuo labiau į naršyklės pirštų antspaudų maskavimą.

Sistemos tikslas – sumažinti blokavimų, gaunamų pagal naršyklės pirštų antspaudų tikrinimą, skaičių⁸. Tai leis greičiau bei pigiau atlikti automatinį duomenų surinkimą.

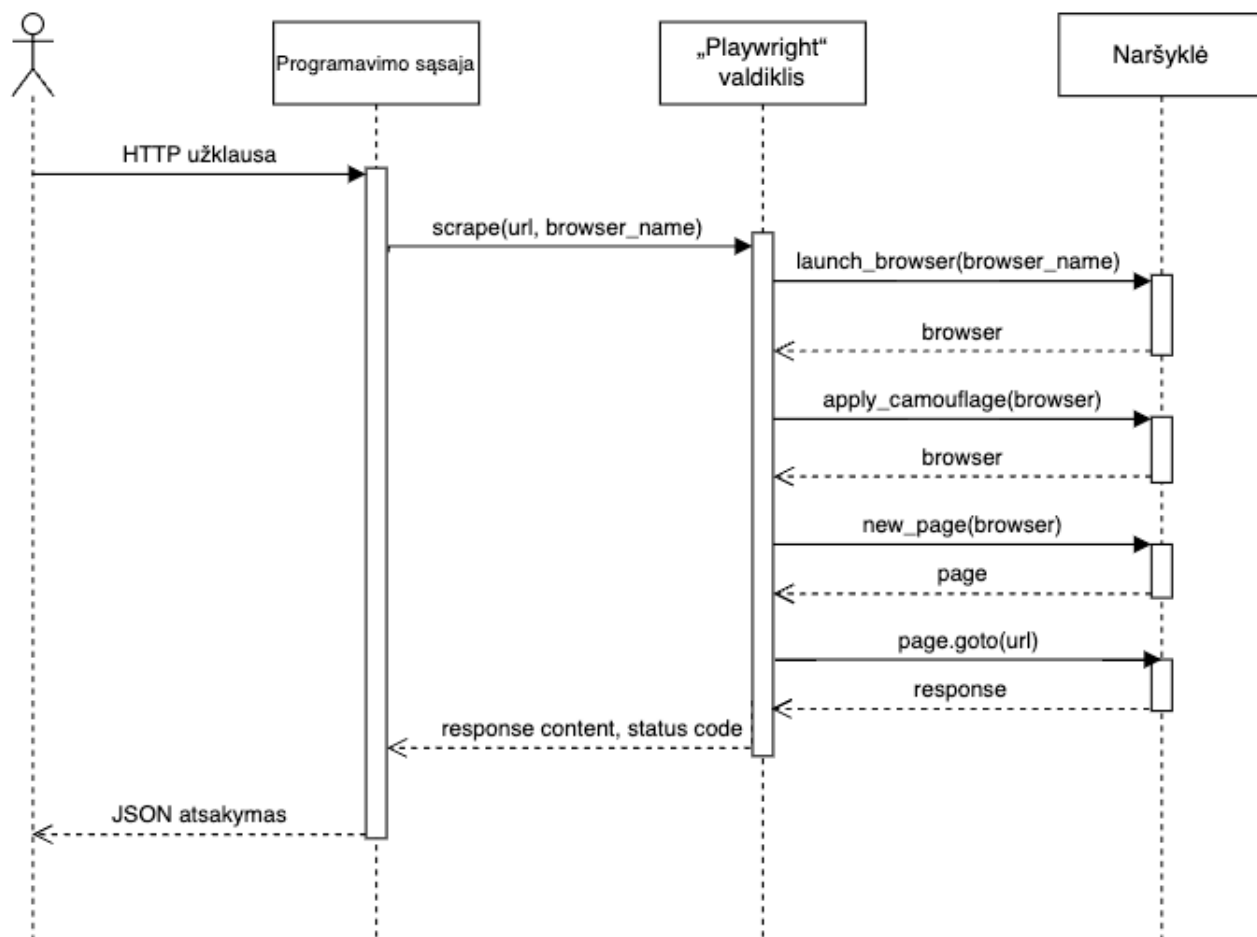
3.2. Sistemos prototipo projektavimas

Šiame poskyryje aprašomos sistemos prototipo techninės detalės, sudaromi funkciniai ir nefunkciniai reikalavimai bei apibrėžiamas visas veikimas.

3.2.1. Sistemos prototipo veikimas

Dažniausiai, informacijos rinkimas iš interneto nesustoja ties automatinio duomenų surinkimu: tai dalis didesnio proceso, kurio metu, renkami duomenys transformuojami bei analizuojami. Dėl šios priežasties, sistemos prototipas turėtų būti lengvai valdomas programine įranga. Lengviausias to įgyvendinimas – aplikacijų programavimo sąsaja (angl. *Application Programming Interface*). Tai suteiks galimybę kitoms programos sąveikauti su sistemos prototipu bei atlikti duomenų transformaciją ar kitus veiksmus nemodifikuojant automatinio duomenų surinkimo programos. Gavus HTTP užklausą per programavimo sąsają, sistemos prototipas turi atidaryti izoliuotą naršyklę, atsitiktinai parinkti vartotojo agentą, ekrano rezoliuciją ir kitus parametrus bei nueiti į užklausoje nurodytą puslapį. Baigus darbą, sistema turi grąžinti surinktą puslapio informaciją JSON formatu kaip pateiktos HTTP užklauskos atsakymą.

⁸Lyginant su paprasta automatinio duomenų surinkimo sistema, įgyvendinta su „Playwright“.



10 pav. Numatoma sistemos prototipo veikimo sekų diagrama

10 pav. matoma „apply_camouflage“ funkcija vykdo didžiąją dalį naršyklės antspaudo mas-kavimo: parenka navigatoriaus parametrus, papildinių skaičių, kalbą bei kita.

3.2.2. Sistemos funkciniai reikalavimai

Toliau pateiktoje 5 lentelėje matomi 9 funkciniai reikalavimai: pirmi 7 reikalavimai skirti automatinės duomenų sistemos naudojimo patogumui, likę yra skirti sistemos tikslo siekimui.

5 lentelė. Funkciniai reikalavimai prototipui

Reikalavimo nr.	Reikalavimo aprašymas
FR-1	Sistema turi priimti duomenų surinkimo užklausas per hiperteksto perdavimo protokolą (HTTP).
FR-2	Sistema atsakyme turi grąžinti puslapio pateiktą HTTP status kodą.
FR-3	Sėkmingo duomenų surinkimo atveju, sistema turi grąžinti JSON objektą su surinktais duomenimis.
FR-4	Sistema turi atmesti netinkamo formato nuorodas.

Reikalavimo nr.	Reikalavimo aprašymas
FR-5	Sistema turi galėti apdoroti daugiau nei vieną užklausą tuo pačiu metu.
FR-6	Sistema turi pranešti apie per ilgai užtrukusį puslapio krovimą.
FR-7	Sistema turi palaikyti duomenų rinkimą Chromium, Firefox bei Webkit (Safari) naršyklėmis.
FR-8	Sistema turi atsitiktinai parinkti vartotojo agentą prieš atlikdama automatinio duomenų surinkimą.
FR-9	Sistema turi užmaskuoti WebDriver objektą, atskleidžiantį apie naršyklės valdymą programine įranga.

3.2.3. Sistemos nefunkciniai reikalavimai

6 lentelėje pateikiami sistemos prototipo nefunkciniai reikalavimai, kuriais siekiama palengvinti sistemos prototipo naudojimą.

6 lentelė. Nefunkciniai reikalavimai prototipui

Reikalavimo nr.	Reikalavimo aprašymas
NFR-1	Sistema turi naudoti Python programavimo kalbą.
NFR-2	Sistema turi naudoti „FastAPI“ biblioteką aplikacijos programavimo sąsajai.
NFR-3	Sistema turi grąžinti atsakymą per 60 sekundžių ar mažiau nuo užklausos gavimo.
NFR-4	Sistema turi naudoti „Playwright“ biblioteką naršyklės be grafinės sąsajos valdymui.
NFR-5	Sistema turi palaukti bent 10 sekundžių po puslapio užsikrovimo, kad įsitikinti, jog visi duomenys užkrauti.

3.3. Sistemos prototipo įgyvendinimas

Šiame poskyryje aprašomos sistemos prototipo įgyvendinimo detalės bei procesas. Sistemos prototipo įgyvendinimo kodą galima rasti <https://github.com/darkixem/playwright-scraper>.

3.3.1. Aplikacijos programavimo sąsaja

Sistemos prototipo programavimo sąsajos implementacijai pasirinkta „FastAPI“ biblioteka. „FastAPI“ – tai moderni bei greita (didelio našumo) Python programavimo kalbos biblioteka, skirta kurti aplikacijų programavimo sąsajas [Fas23]. Tai viena greičiausių egzistuojančių bibliotekų Python programavimo kalbai [Fas23]. Šios bibliotekos pagalba, galima asinchroniškai apdoroti HTTP užklausas.

3.3.2. Maskuojamų parametrų parinkimas

Naršyklės pirštų antspaudas gali būti sudarytas iš daugybės parametrų, todėl svarbu pasirinkti tinkamus parametrus maskavimui. Parametrų maskavimo atrankai bei maskavimo kokybės įvertinimui naudojami du naršyklių analizės įrankiai: <https://bot.incolumitas.com> bei <https://bot.sannysoft.com>.

7 lentelė. Maskuojami naršyklės parametrai

Parametro pavadinimas	Maskavimo paaiškinimas
Vartotojo agentas	Vartotojo agentas atsitiktinai parenkamas iš populiariausių vartotojų agento sąrašo ⁹ .
Ekrano rezoliucija	Atsitiktinai parenkama viena iš populiarių kompiuterio ekrano rezoliucijų. Numatytoji „Playwright“ rezoliucija 1280x720.
WebDriver objektas	WebDriver objektas egzistuoja tik naršyklėse, valdomose programine įranga. Todėl, šis objektas perrašomas prieš pradedant rinkti duomenis.
Naršyklės papildiniai	Naršyklės be grafinės sąsajos dažniausiai neturi nei vieno papildinio, todėl reikia užmaskuoti naršyklės papildinius.
WebGL informacija	Pakeičiami „WebGL Renderer“ bei „WebGL Vendor“ parametrai. Numatytieji parametrai naudoja „Google Inc.“, kurie gali išduoti apie naršyklės be grafinės sąsajos naudojimą.

Šių parametrų maskavimas vyksta pasinaudojant „Playwright“ JavaScript vykdymo funkcionalumo, kai pateiktas kodas vykdomas su kiekvienu atidaromu puslapiu ar navigacijos pasikeitimu¹⁰. Taip pat, naudojama ir daugiau maskavimo priemonių, jau paplitusių automatinio duomenų surinkimo bendruomenėje, kurios nėra išvardintos 7 lentelėje.

⁹Populiariausių vartotojo agentų sąrašą galima rasti čia: <https://www.useragents.me>.

¹⁰Šios funkcijos dokumentaciją galima rasti čia: <https://playwright.dev/python/docs/api/class-page#page-add-init-script>.

Test Name		Result
User Agent	(Old)	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/113.0.5672.53 Safari/537.36
WebDriver	(New)	present (failed)
WebDriver Advanced		passed
Chrome	(New)	missing (failed)
Permissions	(New)	prompt
Plugins Length	(Old)	0
Plugins is of type PluginArray		failed
Languages	(Old)	en-US
WebGL Vendor		Google Inc. (Apple)
WebGL Renderer		ANGLE (Apple, Apple M1 Pro, OpenGL 4.1)
Hairline Feature		present
Broken Image Dimensions		16x16

11 pav. Naršyklės analizės testų pavyzdys (anglų k.)

Test Name		Result
User Agent	(Old)	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36
WebDriver	(New)	missing (passed)
WebDriver Advanced		passed
Chrome	(New)	present (passed)
Permissions	(New)	denied
Plugins Length	(Old)	3
Plugins is of type PluginArray		passed
Languages	(Old)	en-US
WebGL Vendor		Intel Inc.
WebGL Renderer		Intel Iris OpenGL Engine
Hairline Feature		present
Broken Image Dimensions		16x16

12 pav. Naršyklės analizės testų pavyzdys po maskavimo (anglų k.)

11 bei 12 pav. matomi sistemos prototipo naršyklės testų¹¹ rezultatai prieš ir po 7 lentelėje aprašytų parametrų maskavimo. Iš atliktų testų, maskuojami parametrai yra panašesni į paprastos naršyklės vartotojo, todėl puslapiams sunkiau atpažinti tikrą vartotoją nuo sistemos prototipo.

3.4. Apibendrinimas

Aptarus sistemos prototipo poreikį, galima teigti, kad rinkoje reikalinga kokybiška bei nemokama automatinio duomenų surinkimo sistema, nereikalaujanti papildomų išlaidų vystymui. Dėl poreikio lengvai valdyti sistemos prototipą programine įranga, nuspręsta prototipą pateikti kaip HTTP servisą, priimančią užklausas per programavimo sąsają. Išnagrinėjus naršyklių analizės įrankius, nuspręsta sistemos prototipe maskuoti pagrindinius parametrus: vartotojo agentą, ekrano rezoliuciją, WebDriver objektą bei kitus, taip sumažinant galimybę atskirti sistemos prototipą nuo tikro vartotojo.

¹¹ Pavyzdžiuose matomi testai iš puslapio: <https://bot.sannysoft.com>.

4. Sistemos prototipo įvertinimas

Šiame skyriuje aprašomas sistemos prototipo įvertinimo procesas bei rezultatai, lyginant su paprasta automatinio duomenų surinkimo sistema, naudojančią „Playwright“ biblioteką.

4.1. Vertinimui parinkti puslapiai

Automatinio duomenų surinkimo sistemos prototipo įvertinimui naudojami išrinkti atsitiktiniai puslapiai, atitinkantys šiuos reikalavimus:

1. Puslapio duomenys turi būti reiklūs automatiniam duomenų surinkimui. Pavyzdžiui, renkant duomenis iš internetinės parduotuvės, galima lyginti kainas su kitos parduotuvės turimų produktų kainomis.
2. Puslapis turi būti pakankamai didelis, kad turėtų bent menką apsaugą nuo duomenų surinkimo.
3. Renkami duomenys turi būti pasiekiami nereikalaujant registracijos ar prisijungimo.

Remiantis šiais reikalavimais, parinkti penki populiarius puslapiai:

- Internetinė parduotuvė **Amazon** – <https://amazon.com>. Renkami parduotuvės produktų puslapiai, turintys produktų aprašymus bei kainas;
- Paieškos variklis **Google** – <https://google.com>. Renkami paieškos rezultatai.
- Internetinė drabužių parduotuvė **Nordstrom** – <https://nordstrom.com>. Renkama parduodamų produktų informacija bei kainos.
- Kelionių planavimo įrankis **Tripadvisor** – <https://tripadvisor.com>. Renkami apsiųstojimų vietų informacija bei kainos.
- Socialinė tinklas **Twitter** – <https://twitter.com>. Renkama profilių informacija bei paskutiniai profilio pranešimai platformoje.

Taip pat, pasirinktas vienas prevencijos nuo robotų srauto ir automatinio duomenų surinkimo įrankių tiekėjas, **Datadome** – <https://datadome.co>. Šio puslapio duomenų rinkimo rezultatai nėra tokie svarbūs, vertinami taikyti prevencijos būdai prieš duomenų surinkimo sistemą.

4.2. Vertinimo kriterijai

Automatinio duomenų surinkimo sistemos prototipo įvertinimui užsibrėžti du pagrindiniai kriterijai: blokavimų skaičius bei gautų CAPTCHA patikrinimų skaičius. Pagal šiuos du kriterijus vertinami trečiame skyriuje aprašytos automatinio duomenų surinkimo sistemos rezultatai bei paprastos sistemos, naudojančios „Playwright“ biblioteką duomenų surinkimui.

4.3. Duomenų surinkimo rezultatai

Kiekvienam iš 4.1 poskyryje išvardintų puslapių atrinkta po 5 nuorodas bei atlikta po 5 bandymus kiekvienai iš nuorodų¹². Iš viso 25 automatinio duomenų surinkimo bandymai kiekvienam iš atrinktų puslapių. Šių bandymų metu buvo naudojamas vienas IP adresas, todėl parinktas nedidelis skaičius bandymų. Dėl agresyvaus blokavimo pagal naršyklės piršto antspaudą, 25 bandymai leis sėkmingai įvertinti sistemos prototipo veiksmingumą, nemaišant su IP adresais pagrįstos prevencijos būdų.

8 lentelė. Paprastos duomenų surinkimo sistemos rezultatai

Puslapis	Blokavimų skaičius	CAPTCHA patikrinimų skaičius	Sėkmingai surinktų duomenų procentas
Amazon	0/25	23/25	8 %
Google	0/25	0/25	100 %
Nordstrom	25/25	0/25	0 %
Tripadvisor	0/25	0/25	100 %
Twitter	25/25	0/25	0 %
Datadome	25/25	0/25	0 %

Kaip matoma iš 8 lentelės rezultatų, tik du atrinkti puslapiai negrąžino nei vieno blokavimo ar CAPTCHA patikrinimo. Didžioji dauguma duomenų surinkimo bandymų buvo nesėkmingi.

9 lentelė. Sistemos prototipo duomenų surinkimo rezultatai

Puslapis	Blokavimų skaičius	CAPTCHA patikrinimų skaičius	Sėkmingai surinktų duomenų procentas
Amazon	0/25	1/25	96 %
Google	0/25	0/25	100 %
Nordstrom	11/25	0/25	56 %
Tripadvisor	0/25	0/25	100 %
Twitter	0/25	0/25	100 %
Datadome	0/25	25/25	0 %

¹²Visas vertinimui naudotas nuorodas galima rasti pirmame priede.

Pagal 9 lentelėje pateiktus duomenų surinkimo rezultatus, matomas sėkmingų bandymų padidėjimas. Sistemos prototipas sugebėjo surinkti duomenis iš Nordstrom bei Twitter puslapių, taip pat pagerinant Amazon sėkmingų bandymų skaičių. Nors abiems sistemoms nepavyko surinkti duomenų iš Datadome puslapio, prototipas bandymai nebuvo automatiškai užblokuoti: gauti vien CAPTCHA patikrinimai. Tai leidžia teigti, kad sistemos prototipą sunkiau atskirti nuo įprastos automatinio duomenų surinkimo sistemos.

4.4. Apibendrinimas

Atlikus duomenų surinkimą paprasta duomenų surinkimo sistema, naudojanti „Playwright“ ir trečiame skyriuje aprašytu sistemos prototipu, galima teigti, kad įgyvendintas prototipas gali atlikti duomenų surinkimą patikimiau. Neskaitant Datadome puslapio rezultatų, visų puslapių duomenų surinkimą sistemos prototipui atlikti pavyko bent 50 % bandymų, kai paprastai sistemai visiškai nepavyko surinkti Nordstrom ar Twitter duomenų.

Rezultatai

Darbe pasiekti **rezultatai**:

1. Apžvelgtas automatinio duomenų surinkimo procesas:
 - Išskirti bei palyginti du populiarius automatinio duomenų surinkimo metodus: HTTP užklausomis bei internetine naršykle.
 - Išnagrinėti trys esminiai automatinio duomenų surinkimo prevencijos būdai: naršyklės antspaudų tikrinimas, apribojimai IP adresams bei CAPTCHA patikrinimai. Aptartas šių prevencijų vengimas.
2. Išnagrinėti trys egzistuojantys naršyklių be grafinės sąsajos valdymo įrankiai „Selenium WebDriver“, „Playwright“ bei „Puppeteer“:
 - Išnagrinėta kiekvieno iš šių įrankių veikimo architektūra.
 - Atliktas šių įrankių funkcionalumo palyginimas.
 - Atliktas duomenų surinkimo greičio palyginimas trimis atvejais:
 - Duomenis kraunant vykdant JavaScript kodą vartotojo pusėje.
 - Vykdant papildomus veiksmus (įrašant informaciją, mygtuko paspaudimą).
 - Laukiant specifinio elemento atsiradimo.
 - Pagal palyginimų rezultatus, išrinkta „Playwright“ biblioteka sistemos prototipo įgyvendinimui.
3. Suprojektuotas sistemos prototipo veikimas:
 - Užsibrėžtas sistemos prototipo tikslas.
 - Aprašytas sistemos prototipo veikimas.
 - Nubrėžta sistemos prototipo veikimo sekų diagrama.
 - Apibrėžti 9 funkciniai ir 5 nefunkciniai reikalavimai kuriamam automatinio duomenų surinkimo sistemos prototipui.
4. Įgyvendintas automatinio duomenų surinkimo prototipas:
 - Naudojantis naršyklės analizės įrankiais, atrinkti parametrai, kurie maskuojami sistemos prototipe.
 - Naudojantis sekų diagrama bei užsibrėžtais reikalavimais, suprogramuotas automatinio duomenų surinkimo sistemos prototipas.
5. Įvertintos sistemos prototipo duomenų surinkimo galimybės:
 - Atrinkti 5 populiarius puslapiai duomenų rinkimui, bei vienas apsisaugojimo nuo automatinio duomenų surinkimo įrankių tiekėjas.
 - Atliktas duomenų surinkimas sistemos prototipu bei įprasta automatinio duomenų surinkimo sistema, naudojančia „Playwright“ biblioteką.
 - Atliktas duomenų surinkimo rezultatų palyginimas, žiūrint į gautų blokavimų bei CAPTCHA patikrinimų skaičių.

Išvados

Gautos darbo **išvados**:

1. Automatinis duomenų surinkimas internetine naršykle yra brangesnis bei lėtesnis už automatinį duomenų surinkimą HTTP užklausomis: atlikto palyginimo rezultatuose, šis būdas užtruko apie 10 kartų ilgiau bei išnaudojo daugiau interneto srauto. Tačiau, nors duomenų surinkimas HTTP užklausomis ir yra greitesnis bei pigesnis, jis negali surinkti duomenų, kraunamų vykdant JavaScript programinį kodą kliento pusėje.
2. Atsižvelgiant į atliktus nagrinėtų naršyklių be grafinės sąsajos valdymo įrankių greičio bei funkcionalumo palyginimus, matoma, kad nors „Selenium WebDriver“ siūlo daugiausiai funkcionalumo, „Playwright“ architektūra leidžia duomenų surinkimą atlikti greičiausiai.
3. Pagal atliktą sistemos prototipo įvertinimą, atlikti parametrų maskavimai padėjo sistemai būti sunkiau atpažįstamai: gauta mažiau blokavimų ar CAPTCHA patikrinimų, lyginant su įprasta automatinio duomenų surinkimo sistema.
4. Sėkmingam naršyklės piršto antspaudų tikrinimo vengimui dažnai užtenka maskuoti svarbiausius parametrus, kaip vartotojo agentą, išduodančius, kad naudojama automatinio duomenų surinkimo sistema.

Šaltiniai

- [Fas23] FastAPI. *FastAPI*. 2023. [žiūrėta 2023-05-21]. Prieiga per internetą: <https://fastapi.tiangolo.com/lo/>.
- [Khd21] M. Khder. Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. *International Journal of Advances in Soft Computing and its Applications*. 2021-12, tomas 13, p. 145–168. Prieiga per internetą: <https://doi.org/10.15849/IJASCA.211128.11>.
- [Lam23] Lambdatest. *What Is Playwright? Playwright Testing Tutorial - A Guide With Examples*. 2023-03. [žiūrėta 2023-05-17]. Prieiga per internetą: <https://www.lambdatest.com/playwright>.
- [Mot21] L. Motiejunas. *Datacenter Proxies vs. Residential Proxies*. 2021-07. [žiūrėta 2023-05-16]. Prieiga per internetą: <https://oxylabs.io/blog/the-difference-between-data-center-and-residential-proxies>.
- [Moz23a] Mozilla. *HTTP*. 2023-05. [žiūrėta 2023-05-29]. Prieiga per internetą: <https://developer.mozilla.org/en-US/docs/Web/HTTP>.
- [Moz23b] Mozilla. *User-Agent*. 2023-04. [žiūrėta 2023-05-29]. Prieiga per internetą: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent>.
- [Per19] E. Persson. *Evaluating tools and techniques for web scraping*. 2019. Magistro darbas. KTH, School of Electrical Engineering ir Computer Science (EECS).
- [Pro22] ProgramsBuzz. *Playwright Architecture*. 2022-11. [žiūrėta 2023-05-17]. Prieiga per internetą: <https://www.programsbuzz.com/article/playwright-architecture>.
- [Pup23] Puppeteer. *Puppeteer Documentation*. 2023. [žiūrėta 2023-05-18]. Prieiga per internetą: <https://pptr.dev>.
- [Scr22a] ScraperHero. *How to make web scraping with browsers faster and cost effective*. 2022-11. [žiūrėta 2023-05-15]. Prieiga per internetą: <https://www.scrapehero.com/cheaper-and-faster-web-scraping-with-browsers/>.
- [Scr22b] ScrapFly. *Browser Fingerprinting: What It Is and How to Bypass It*. 2022-04. [žiūrėta 2023-05-16]. Prieiga per internetą: <https://scrapfly.io/blog/how-to-avoid-web-scraping-blocking-ip-addresses/>.
- [Sel22] Selenium. *Selenium Overview*. 2022-09. [žiūrėta 2023-05-16]. Prieiga per internetą: <https://www.selenium.dev/documentation/overview/>.
- [Sin21] R. Singh. *Selenium WebDriver Achitecture*. 2021-11. [žiūrėta 2023-05-17]. Prieiga per internetą: <https://www.toolsqa.com/selenium-webdriver/selenium-webdriver-architecture/>.
- [Sub21] D. Subramanian. *Puppeteer Basics*. 2021-06. [žiūrėta 2023-05-18]. Prieiga per internetą: <https://sdkdeepa.medium.com/puppeteer-basics-3be7f9f82a08>.

- [Una23] J. Unadkat. *Selenium WebDriver Tutorial : Getting Started with Test Automation*. 2023-04. [žiūrēta 2023-05-17]. Prieiga per internetą: <https://www.browserstack.com/guide/selenium-webdriver-tutorial>.
- [Zen23a] ZenRows. *Browser Fingerprinting: What It Is and How to Bypass It*. 2023-02. [žiūrēta 2023-05-16]. Prieiga per internetą: <https://www.zenrows.com/blog/browser-fingerprinting/>.
- [Zen23b] ZenRows. *How to Avoid CAPTCHA and reCAPTCHA: 7 Proven Methods*. 2023-03. [žiūrēta 2023-05-16]. Prieiga per internetą: <https://zenrows.com/blog/avoid-captcha>.
- [Zha17] B. Zhao. Web Scraping. Iš: 2017-05, p. 1–3. ISBN 978-3-319-32001-4. Prieiga per internetą: https://doi.org/10.1007/978-3-319-32001-4_483-1.

Sąvokų apibrėžimai

Aplikacijų programavimo sąsaja (angl. *Application Programming Interface*) – tai sąsaja, kuri suteikia programinei įrangai pasiekti kitos programos funkcionalumą ar apsikeistų su ja duomenimis.

Blokavimas – internetinio puslapio grąžintas atsakymas, skirtas pranešti apie nesuteiktą prieigą prie turimų resursų.

Hiperteksto perdavimo protokolas (HTTP) (angl. *HyperText Transfer Protocol*) – tai programos lygio protokolas, skirtas hipertekstiniais dokumentams, pavyzdžiui HTML, perduoti [Moz23a].

Naršyklė be grafinės sąsajos (angl. *headless browser*) – programine įranga valdoma naršyklė, neturi grafinės naudotojo sąsajos.

Naršyklės piršto antspaudas (angl. *browser fingerprint*) – tai informacija, surinkta apie naudojamą naršyklę bei jos parametrus.

Tarpinis serveris (angl. *proxy server*) – tai serverio programa, kuri veikia kaip tarpininkas tarp kliento ir kito serverio.

Vartotojo agentas (angl. *user-agent*) – HTTP užklauso antraštė, leidžianti internetiniams puslapiams identifikuoti programą, operacinę sistemą, tiekėją ir versiją [Moz23b].

Priedai

Priedas nr. 1

Nuorodos, naudotos sistemos prototipo vertinimui

Amazon:

1. <https://www.amazon.com/dp/B0B928B6BC>;
2. <https://www.amazon.com/dp/B01HID4Y7Q>;
3. <https://www.amazon.com/dp/B08C1W5N87>;
4. <https://www.amazon.com/dp/B07MFZXR1B>;
5. <https://www.amazon.com/dp/B08TG5X172>.

Google:

1. <https://google.com/search?q=vilniaus+universitetas>;
2. <https://google.com/search?q=vu+mif>;
3. <https://google.com/search?q=bakalaurinis+darbas>;
4. <https://google.com/search?q=vilnius+university>;
5. <https://google.com/search?q=vu>;

Nordstrom:

1. <https://www.nordstrom.com/s/court-borough-low-2-sneaker-baby-walker-toddler-little-kid-bi-5756069>;
2. <https://www.nordstrom.com/s/the-north-face-base-camp-large-travel-canister/6443281>;
3. <https://www.nordstrom.com/s/acton-iii-bluetooth-speaker/7096988>;
4. <https://www.nordstrom.com/s/smart-home-diffuser/7459906>;
5. <https://www.nordstrom.com/s/boll-and-branch-boll-branch-waffle-organic-cotton-blanket/5450544>.

Tripadvisor:

1. https://www.tripadvisor.com/VacationRentalReview-g274951-d17677537-Old_Town_Gate_Residence_in_the_heart_of_Vilnius_Maisonette_patio_parking-Vilnius_Viln.html;
2. https://www.tripadvisor.com/VacationRentalReview-g274951-d19933074-Paupio_sodyba-Vilnius_Vilnius_County.html;
3. https://www.tripadvisor.com/VacationRentalReview-g274951-d8335302-Bomb_Shelter_apartment_with_SPA_Old_town_free_parking-Vilnius_Vilnius_County.html;
4. https://www.tripadvisor.com/VacationRentalReview-g274951-d7803757-Villa_Aguona_3_bedrooms_villa-Vilnius_Vilnius_County.html;
5. https://www.tripadvisor.com/VacationRentalReview-g274951-d13560953-Villa_EverGreen_in_downtown_with_private_garden_Garage_parking_Terrace_Hot_tub-Vilniu.html.

Twitter:

1. https://twitter.com/VU_LT;
2. <https://twitter.com/Nike>;
3. <https://twitter.com/apple>;
4. https://twitter.com/dog_rates;
5. <https://twitter.com/samsung>.

Datadome:

1. <https://datadome.co/datadome-captcha/>;
2. <https://datadome.co/pricing/>;
3. <https://datadome.co/integrations/>;
4. <https://datadome.co/bot-protection-online-fraud-prevention/>;
5. <https://datadome.co/learning-center/>.