Kauno technologijos universitetas

Informatikos fakultetas

Žiniatinklio pakeitimo aptikimo metodas

Baigiamasis magistro studijų projektas

Lukas Antanavičius

Projekto autorius

vyr. lekt., dr. Gedeiminas Činčikas

Vadovas

Kaunas, 2024

Kauno technologijos universitetas

Informatikos fakultetas

Žiniatinklio pakeitimo aptikimo metodas

Baigiamasis magistro studijų projektas

Informacijos ir informacinių technologijų sauga (6211BX008)

Lukas Antanavičius

Projekto autorius

vyr. lekt., dr. Gedeiminas Činčikas

Vadovas

prof. Jevgenijus Toldinas

Recenzentas

Kaunas, 2024

Kauno technologijos universitetas

Informatikos fakultetas

Lukas Antanavičius

Žiniatinklio pakeitimo aptikimo metodas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Lukas Antanavičius

Patvirtinta elektroniniu būdu

Lukas Antanavičius. Žiniatinklio šablonų pakeitimo aptikimo metodas. Magistro studijų baigiamasis projektas vadovas vyr. lekt., dr. Gedeiminas Činčikas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): informatikos inžinerija.

Reikšminiai žodžiai: puslapio pakeitimas, puslapio pakeitimo aptikimas, šablono ir puslapio tikrinimas, puslapio šablonai.

Kaunas, 2024. 57 p.

Santrauka

Žiniatinklio pakeitimo atvejai niekur nedingsta ir ši problema lieka greitu metu niekur nepradings. Šiuo metu didžiausia dalis interneto puslapių yra dinaminiai, kurie generuoja puslapius įterpdami papildomus turinio komponentus. Šiame darbe siūlomas naujas metodas interneto puslapio pakeitimų aptikimui panaudojant šablonus, iš kurių pagalba turinys yra generuojamas. Metodas tikrina gautą puslapį su atitinkamai nurodytu šablonu ir pagal tai lyginamas turinys. Jei puslapis neatitinka bent vieno komponento, esančio šablone, reiškia puslapio turinys yra pakeistas. Toks sprendimas yra potencialiai turintis didžiausią taiklumą, kadangi tikrinamas pats dinaminio puslapio turinio šaltinis.

Lukas Antanavičius. A New Method for Website Alteration Detection. Master's Final Degree Project supervisor sen. lect., dr. Gedeiminas Činčikas; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Informatics Engineering.

Keywords: defacement, website defacement, change detection, content change detection, website change, website change detection.

Kaunas, 2024. 57 pages.

Summary

Website defacement is not disappearing as an attack vector and it is still a potential problem that websites can face. This proposed new method tries to tackle the elusive problem of dynamically generated website change detection. Some dynamic websites have pages generated using templates. These templates insert content dynamically. This paper proposes a new method for website page change detection with the use of these templates. The method checks for any deviations of the received web page from the template. If the page does not match with at least one component, that means the page most likely changed illegally. This solution potentially has a highest accuracy, because change detection is performed using the very source of the dynamically generated web page.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

LFi – Lokalaus failo įtraukimas (angl. *Local file inclusion*);

RFi – Nuotolinio failo įtraukimas (angl. *Remote file inclusion*);

SQLi – *SQL* sintaksės įskiepijimas (angl. *SQL injection*);

DOM – Dokumento objektų modelis (angl. *Document Object Model*).

Terminai:

Žetonas – mažiausias leksinis programavimo kalbos atominis vienetas (angl. *token*).

Leksinė analizė – pirminis programavimo kalbos simbolių atpažinimas ir grupavimas į žetonus.

Lekseris – programinė įranga, vykdanti leksinę analizę.

Įvadas

Žiniatinklis (angl. *World Wide Web, WWW*), pradėtas žymaus anglų mokslininko Tim Berners-Lee 1989 metais Europos branduolinių mokslinių tyrimų organizacijoje (*CERN*), kurio originali paskirtis buvo CERN informacijos, esančios kompiuteriuose, nuotolinė, paskirstyta ir paprasta prieiga [1]. Ši idėja tapo viešai prieinama 1991 metais ir įgavo populiarumo pagreitį po 1993 metais išleistos pirmosios interneto naršyklės su grafine sąsaja *Mosaic* [2]. Po *Mosaic* 1994 metais buvo sukurtas geriau žinomas *Netscape Navigator*, kuris suteikė galimybę naudotis *Java* ir *JavaScript* programavimo kalbomis žiniatinklyje ir pirmasis įgyvendino *HTTP* slapukų (angl. *cookies*) bei naršyklės kadrų (angl. *frames, HTML5* standarte žinomas kaip *iframe* elementas) funkcionalumą [3]. Tai yra pradžia pirmosios kartos saitynui (angl. *Web 1.0*), kuris buvo statinis ir nekeičiamas (tik skaitymui) [4].

Kadangi saitynas sukurtas kaip dokumentų ir kitų failų nuotolinė prieiga, buvo sukurta sistema šiuos duomenų failus pasiekti pavadinimu universalusis ištekliaus identifikatorius (angl. *Uniform Resource Identifier*, URI). Šie adresai pasiekiami per hiperteksto persiuntimo protokolą (angl. *HyperText Transfer Protocol, HTTP*) interneto tinkle [5].

Antrosios kartos žiniatinklis (angl. *Web 2.0*) laikomas žiniatinklio etapas nuo maždaug 2004. Tai saitynas, kuriame žmonės gali interaktyviai kelti turinį ir sąveikauti su interneto svetainėmis. Antroje kartoje interneto svetainės tapo dinaminės – tiek dizaino, tiek funkcionalumo prasme [4].

Trečioji saityno karta (angl. *Web 3.0*) šiuo metu siejasi su decentralizacija ir nuosavybe, grindžiama blokų grandinėmis bei kartu taip pat su semantiniu saitynu, kurį pasiūlė tas pats Tim Berners-Lee [4, 6].

Dar prieš pirmosios kartos saityną programišiai įvairiais motyvais įsilauždavo į sistemas išnaudojant saugumo spragą ar kitus metodus. Programišiai natūraliai nusitaikė į šią naują pirmos kartos saityno technologiją. Interneto svetainė buvo viešai pasiekiama vieta, kurią gali pamatyti potencialiai daug žmonių. Taip gimė piktavalė žiniatinklio pakeitimo ataka. Šios atakos metu programišius pakeičia interneto svetainės turinį išnaudodamas serverio saugumo spragą arba gaudamas administratoriaus prisijungimo duomenis. Tokia ataka dažniausiai ištrindavo buvusį turinį ir pakeisdavo jį programišiaus slapyvardžiu kartais pridedant kitą žinutę su politiniu, religiniu, finansiniu ar kitu motyvu [7].

Dėl programišių vykdomų atakų, reikėjo sugalvoti metodus ir įrankius, kurie apsaugotų ar bent praneštų, jog tokio tipo ataka įvyko ar yra vykdoma. Taip buvo pritaikytos įsilaužimo nustatymo sistemos (angl. *Intrusion Detection System, IDS*) ir įsilaužimo prevencijos sistemos (angl. *Intrusion Prevention System, IPS*). Statinės interneto svetaines lengva tikrinti ir atpažinti pakitimą, tačiau dinaminės interneto svetainės gali keisti savo turinį kas kartą ją atidarius (pavyzdžiui naudotojų komentarai, tinklaraščio įrašai, naujienų įrašai). Taip pat problema kyla automatizuojant turinio pakeitimo aptikimą [7]. Saityno turinio pakeitimas gali įvykti tiek dėl piktavalių programišių, tiek dėl žmogiškos administratoriaus klaidos.

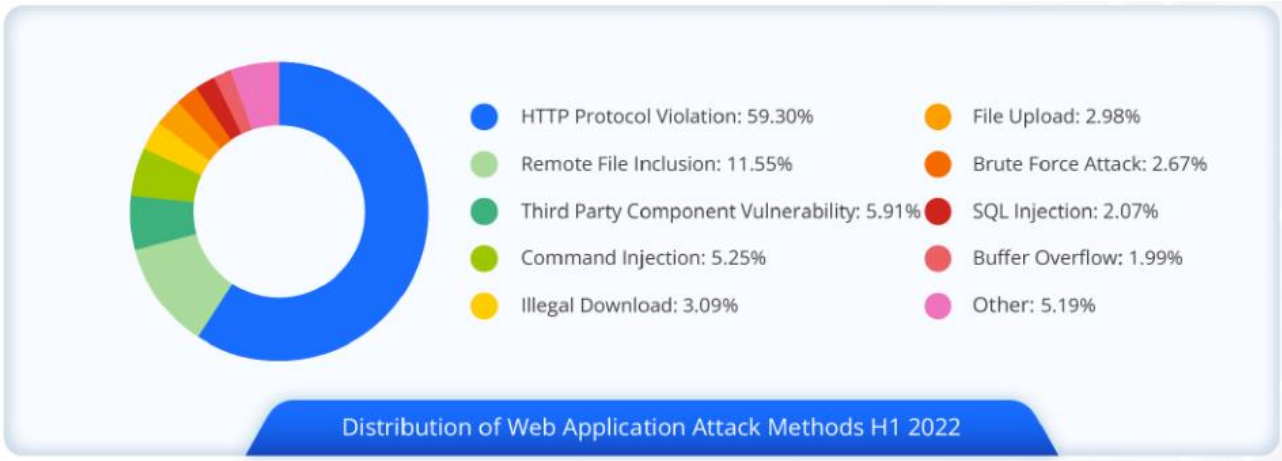
1. Žiniatinklio turinio pakeitimo ir aptikimo analizė

Atakos prieš saityno turinį serveriuose yra viešai matomas įsilaužimo faktas. Tai motyvuoja įmones užkirsti tokio tipo atakas. Dėl saityne naudojamų technologijų įvairovės atakų paviršius yra platus. Į daugumą interneto svetainių yra įsilaužiama dėl neatnaujintų sistemų, kurių saugumo spragos yra žinomos arba neapsaugoti duomenų įvesties metodai. Tam programišiai naudoja skenavimo įrankius, kviečiant kuo daugiau interneto svetainių ir bandant automatiškai išnaudoti šias spragas per *HTTP* užklausas [7]. Dažniausiai pasitaikančios atakos gali būti grupuojamos į vieną bendrinę „įterpimo“ grupę, kuriai priklauso:

- Tarppuslapinis skriptų panaudojimas (angl. Cross-site Scripting, *XSS*);
- SQL sintaksės įterpimas (angl. *SQL injection, SQLi*);
- *PHP* kodo įterpimas (*PHPi*);
- Komandų įterpimas (*CMDi*)
- Lokalių failų įtraukimas (LFi)
- Nuotolinių failų įtraukimas (RFi) [7].

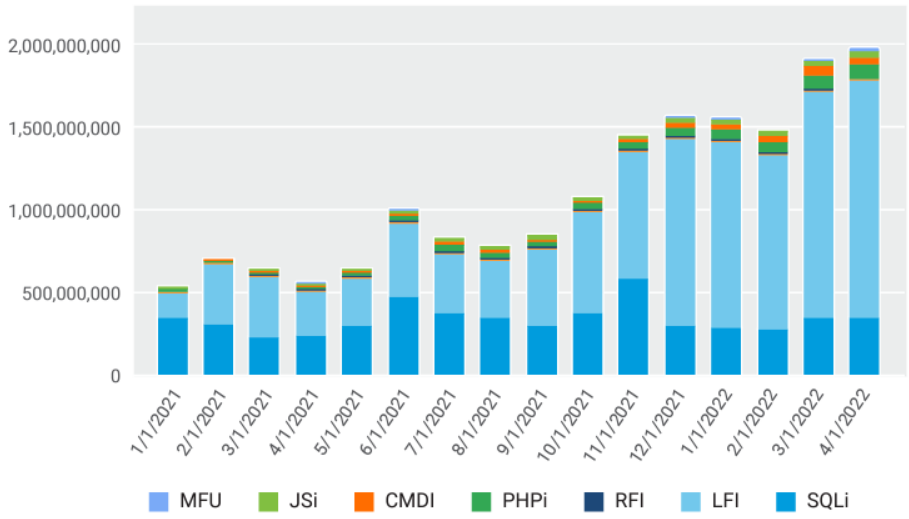
Taip pat piktavaliai programišiai gali pasinaudoti artimais kodo įskiepijimo metodais, tokiais kaip tarppuslapinės užklausos klastojimas (angl. *Cross-site request forgery, CSRF*). Jeigu įsilaužiama į patį serverį (pavyzdžiui, per *SSH*), programišius turi pilną prieigą ir gali keisti visą serverio turinį ir konfigūraciją.

Pagal *CDNetworks* paviešintą ataskaitą, 2022 metais *HTTP* protokolo nesilaikymo atakos išpopuliarėjo apie 25 % lyginant su 2021 metais, antroje vietoje nuotolinio failo įtraukimo ataka bei trečioje – trečiųjų šalių komponentų saugumo spragos (žr. 1 pav.). Čia į *HTTP* protokolo nesilaikymo atakas įeina *XSS, CSRF* ir kitos [8].

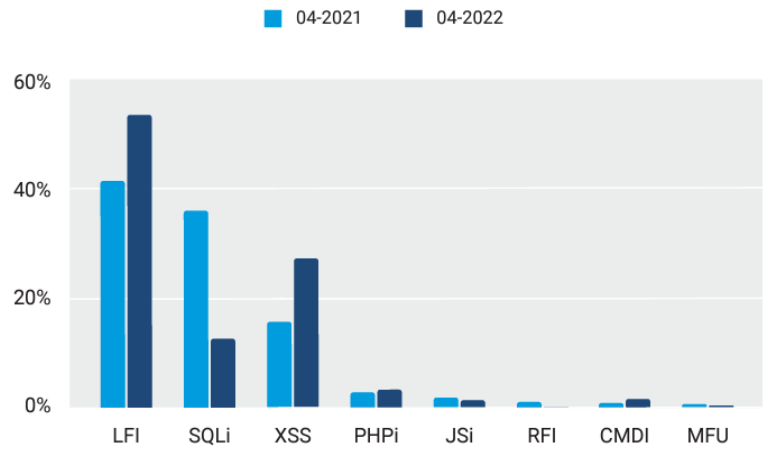


1 pav. Saityno atakų tipai 2022 metų pirmos pusės laikotarpiu pagal *CDNetworks* duomenis [8].

Pagal Akamai duomenis, 2022 metais populiariausias atakos vektorius – lokalaus failo įterpimo (LFi), kurios padidėjo trigubai nuo 2021 metų, antroje vietoje seka *SQL* sintaksės įterpimo atakos (SQLi), kurių populiarumas žymiai sumažėjo lyginant su 2021 metais. Trečioje vietoje – *XSS* atakos (žr. 2 pav. ir 3 pav.). Tokia tendencija rodo, jog piktavaliai programišiai teikia prioritetą ne duomenų išgavimui per *SQL* injekcijas, o pilnam įsilaužimui į serverį [9].



2 pav. Mėnesinės atakų vektorių populiarumo tendencijos 2021 ir 2022 metais pagal *Akamai* duomenis [9]



3 pav. Atakų vektoriai 2021 ir 2022 metais pagal *Akamai* duomenis [9]

Taip pat *Akamai* akcentuoja, jog saugumo spragos, žinomos kaip *Log4Shell* ir *Spring4Shell*, yra aktyviai išnaudojamos ir išlieka vienos populiariausių atakos metodų [9].

1.1. Žiniatinklio atakų tipai

Dauguma atakų bendruoju atveju naudojasi kodo įterpimo atakos metodu. Šis metodas gali būti įgyvendinamas pasinaudojant įvairiomis žiniatinklio technologijų saugumo spragomis.

1.1.1. *LFi* atakos

Lokalių failų įterpimo (angl. *local file inclusion, LFi*) atakos gali būti naudojamos dvejopai: pirma, galima atrasti sisteminius prisijungimo duomenis naršyklės pagalba pasiekiant, pavyzdžiui, */etc/passwd* failą ir antra, piktavalis, atradęs sisteminius failus sužino, jog sistema nėra tinkamai apsaugota ir supranta, kad kitos serverio vietos taip pat greičiausiai nebus tinkamai apsaugotos [9]. Pasinaudojant atitinkamos operacinės sistemos kelio parinkimo sintakse, galima pasiekti kitus nenumatytus failus. Tam naudojami ženklai `../` *Unix* ir `\..` *Windows* operacinėse sistemose, kurie reiškia „vienu katalogu aukščiau“. Pilnas tokios atakos pavyzdys atakos prieš serverį, naudojantį *Unix* operacinę sistemą:

```
https://insecure-website.com/loadImage?filename=../../../etc/passwd
```

Pilnas pavyzdys atakuojamo serverio, naudojančio *Windows* operacinę sistemą:

```
https://insecure-website.com/loadImage?filename=..\..\..\windows\win.ini
```

Netgi kai serveris tinkamai išfiltruoja tokios atakos ženklus `../` ir `..\` vis tiek įmanoma įvykdyti tokią ataką pakeičiant ženklus į *URL* koduotę: `%2e%2e%2f` arba `%252e%252e%252f`.

Norint apsisaugoti nuo tokių atakų, saityno serveriai, tokie kaip *Nginx*, *Apache* ar *Internet Information Services* (*IIS*) turi filtruoti užklausas griežtai reikalaujant, jog kiekvienos užklausos kelias nebūtų kitoks, nei toks, kurį naudoja interneto puslapis.

1.1.2. *SQLi* atakos

SQL įterpimo (SQLi) atakos visuomet buvo programišių akiratyje. Ši ataka gali būti įvykdoma ten, kur duomenų užklausa interneto svetainėje nėra tikrinamai apsaugota ir yra tiesiai perduodama vykdyti duomenų bazei.

Tokios atakos pavyzdys gali būti serverio pusėje įgyvendintas duomenų atrinkimas pagal naudotojo identifikacijos numerį, nefiltruojant duomenų:

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

Čia antroje eilutėje *txtSQL* tiesiog prijungia gautą naudotojo duomenų bazės identifikacijos numerį ir siunčia vykdyti duomenų bazės serveriui. Paprastam naudotojui interneto svetainė grąžins tik to naudotojo duomenis iš lentelės. Jei naudotojo ID yra 105, *SQL* užklausa atrodytų taip:

```
SELECT * FROM Users WHERE UserId = 105
```

Tačiau piktavalis, pateikęs užklausą su papildomais argumentas `105 OR 1=1` gali gauti visų naudotojų informaciją. Čia išpildomas pirmasis loginis patikrinimas `UserId = 105`, tuomet pateikiamas antras tikrinimas su raktažodžiu `OR` (arba), kuris bus visada teisingas `1=1`. Tuomet *SQL* užklausa atrodo taip:

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

Tokiu būdu, *SQL* apribojimas `WHERE` yra visuomet teisingas ir atrenkama visa lentelė *Users*.

Norint išvengti šios atakos, būtina serverio programiniame kode tikrinti gautus duomenis ir filtruoti netinkamas užklausas. Šiame pavyzdyje užtektų tikrinti, ar argumentas, esantis `WHERE` užklausoje yra skaičiaus tipo.

1.1.3. *XSS* atakos

Tarppuslapinis skriptų panaudojimo (angl. *Cross-site scripting*, *XSS*) ataka veikia iš interneto serverio pateikiant nuorodą į užkrėstą failą naudotojams. Programišius, turintis prieigą prie interneto puslapio failų, prideda *JavaScript* kodą, kuris bus vykdomas naudotojo kompiuteryje. Pavyzdžiui, interneto puslapyje yra paieškos funkcija, kurios nuoroda gali atrodyti taip:

```
https://www.insecure-website.com/search?q=search+term
```

Čia paieška vykdoma su parametro raktu `q`, kurio reikšmė yra `search term`. Naudotojui atidaromas paieškos langas su atrinktais duomenimis.

Jei interneto svetainė pažeista *XSS* ataka, toks pats paieškos kreipinys gali būti užkrėstas:

```
https://www.insecure-website.com/search?q=<script>x=new
Image;x.src=”http://attacker.website/getcookie.php?cookie=”+document.cookie;</script>
```

Kadangi *JavaScript* kodą galima vykdyti net naršyklėje esančioje nuorodos juostoje, šis kodas bus vykdomas nuosekliai: sukuriamas nuotraukos objektas, nuotraukos objektui priskiriamas šaltinis su programišiaus valdomu serveriu, kurio nuoroda sukonstruota taip, kad galėtų persiųsti naudotojo sesijos slapuką. Šiame pavyzdyje numanoma, jog piktavalio svetainė suprogramuota taip, kad galėtų priimti šį slapuką per parametrą `cookie` ir išsaugoti savo duomenų bazėje. Taip gali būti perimta sesija ir piktavalis gali naudotojo vardu vykdyti veiksmus originalioje interneto svetainėje.

Norint išvengti tokio tipo atakos serveris turi filtruoti ir atrinkti užklausas, kuriose yra nestandartiniai duomenys, tokie kaip šiame pavyzdyje `<script>` žymės.

1.1.4. *CSRF* atakos

Kryžminis svetainės užklausos klastojimo *(*angl. *Cross Site Request Forgery, CSRF)* ataka naudotojui nežinant pakeičia kreipinio į puslapį duomenis. Paprasčiausias šios atakos panaudojimas gali būti į nuotraukos žymę įkelta nuoroda į puslapį su norimu atlikti veiksmu. Pavyzdžiui, jei interneto puslapis `www.insecure-website.com` turi funkcionalumą per *HTTP GET* metodą pakeisti naudotojo el. paštą, galima sudaryti tokią nuotraukos žymę:

```
<img src="https://www.insecure-website.com/email/change?email=attacker@attacker.website">
```

Ši ataka, pavyzdžiui, atsiųsta el. laiške ar įkelta interneto forume, gali įvykdyti kvietimą iš naudotojo pusės vien tik taikomajai programai užkrovus nuotrauką. Sudėtingesni *CSRF* atakos atvejai naudoja formas bei *HTTP POST* metodą. Tam reikia naudotojui paspausti mygtuką, kuris įvykdo ataką.

Norint apsisaugoti nuo *CSRF* atakų, interneto svetainės turi įgyvendinti *CSRF* saugumo rakto (angl. *CSRF token*) funkcionalumą, kai norima atlikti neeilinį funkcijos kvietimą. Taip pat naršyklės pačios įgyvendina „tos pačios svetainės“ slapukų (angl. *SameSite cookies*) apsaugos mechanizmą, kai naršyklė tokio tipo nuorodos neatidaro esant skirtingam adresui nei šiuo metu esama.

1.1.5. RFi atakos

Nuotolinių failų įtraukimo (angl. *remote file inclusion, RFi*) atakos įterpia į interneto svetainę kodo failą iš kito adreso. Ši saugumo spraga galima tik svetainėse, kurios įgyvendintos naudojant interpretuojamą programavimo kalbą. Tai dažniausiai bus *PHP*. Pavyzdžiui, *PHP* kodo dalis serveryje, kuri pagal gautą parametrą atidaro naują failą vykdymui:

```
<?php
  $module = $_GET["module"];
  include $module;
?>
```

Programišius, išsiaiškinęs, jog serveryje esanti logika yra tokia, gali pateikti puslapio kvietimą su jo kontroliuojamame serveryje esančio failo nuoroda parametre:

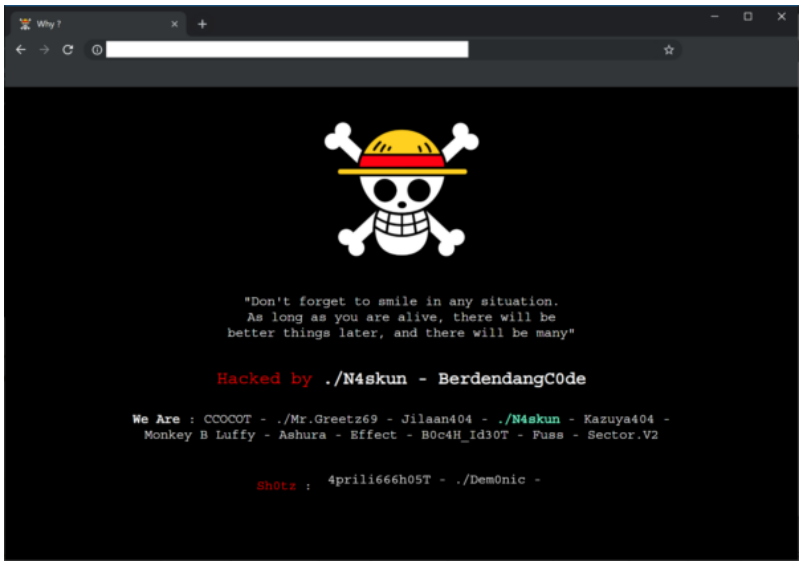
```
https://www.insecure-website.com/index.php?module= http://attacker.website/php-reverse-shell.php
```

Tuomet atakuojamame serveryje užkraunamas ir vykdomas gautas failas `php-reverse-shell.php`, kurio turinys ir naudojimas žinomas programišiui.

Norint išvengti tokios atakos *PHP* įgyvendintoje interneto svetainėje, būtina atnaujinti *PHP* versiją bent į *7.4.0* arba naujesnę. Taip pat konfigūraciniame faile *php.ini* nustatyti reikšmę `allow_url_include = Off`. Papildomai, programuojant reiktų vengti *include* kvietimo, kuris naudoja įvesties duomenis. Tačiau būna situacijų, kai toks sprendimas neišvengiamas, todėl rekomenduojama nustatyti leidžiamų failų sąrašą (angl. *whitelist*) [10].

1.2. Žiniatinklio turinio pakeitimas

Programišius, panaudojęs vieną ar kelis iš atakos metodų įsilaužti ar kitaip neigiamai paveikti interneto svetainę, dažniausiai pakeičia turinį į netinkamą. Puslapio nulaužimas gali būti dvejopas: apgauti ir išnaudoti naudotojus siekiant naudos atakuotojui arba pakeisti tekstą ir dizainą norint paskelbti kokią nors žinutę, kuris vadinamas svetainės suniokojimu (angl. *defacement*). Svetainės suniokojimas dažniausiai turi matomą programišiaus slapyvardį, svetainės fono spalva – juoda ir dizainas dažniausiai yra labai paprastas (žr. 4 pav. ir 5 pav.)



4 pav. Svetainės suniokojimo pavyzdys be akivaizdaus tikslo [11]

5 pav. Svetainės suniokojimo pavyzdys su perspėjimu saugumui [12]

Remiantis šiais pavyzdžiais, galima teigti, jog dažniausiai suniokotos interneto svetainės dažniausiai yra lengvos – neužimančios daug vietos ir nenaudojančios daug duomenų dėl jų paprasto dizaino, lyginant su moderniomis interneto svetainėmis.

1.3. Žiniatinklio turinio pakeitimo aptikimo metodai

Žiniatinklio turinio pakeitimo aptikimas gali būti skiriamas į dviejų tipų metodus: algoritminis ir mašininio mokymo. Tačiau taip pat svarbu ir tai, kaip interneto svetainė atrodo vizualiai. Tam naudojami vaizdo analizės metodai, kurie gali būti grįsti tiek algoritmu, tiek mašininiu mokymu.

Aptikimo metodai yra grindžiami periodiniu tikrinimu. Pirmoji ir pagrindinė problema, su kuria susiduria visi aptikimo metodai – nustatyti tikrinimo dažnumą. Kuo didesnis tikrinimo periodas, tuo didesnė apkrova bus tiek tikrinimo programos įrangai, tiek pačio interneto puslapio serveriui. Antroji problema – neteisingų perspėjimai. Jei įrankis grąžina klaidingą perspėjimą – ilgainiui juo besinaudojantis asmuo pradės į šiuos pranešimus kreipti mažiau dėmesio arba net ignoruoti. Tuo pačiu per dažnas tikrinimo dažnis padidina klaidingų pranešimų dažnumą.

1.3.1. Kontrolinės sumos

Pats paprasčiausias turinio pakitimo aptikimo metodas naudoja kontrolines sumas (angl. *checksum*). Šios kontrolinės sumos gali naudoti *MD5*, *SHA1* ar kitus algoritmus. Toks metodas lygina gautą ir prieš tai buvusią puslapių kontrolines sumas. Jei jos nesutampa – galimai įvyko nenumatytas pokytis. Tačiau šis metodas tinka tik statinių svetainių turiniui, kadangi bet koks bent vieno simbolio pokytis keičia kontrolinės sumos reikšmę [13, 7].

1.3.2. Diff algoritmai

Kitas paprastas interneto svetainių turinio pakeitimo aptikimo metodas grindžiamas *diff* tipo algoritmu, kuris plačiai naudojamas *git* įrankyje, skirtam programinio kodo versijų kontrolėje. Šis metodas lygina visą puslapio turinį (įskaitant *HTML* žymas) su ankstesne versija ir tikrina kiekvieno simbolio skirtumą. Šis metodas yra lankstesnis ir gali būti naudojamas tiek statinėse, tiek dinaminėse interneto svetainėse. Norint panaudoti šį metodą dinaminėse svetainėse, reikia nustatyti turinio pakitimo slenkstį. Pagrindinė užduotis tokiam metodui įgyvendinti yra šio slenksčio nustatymas. Jei slenkstis per žemas – nenumatytas turinio pakeitimas gali būti nenustatytas ir apie jį nepranešama. Jei slenkstis per aukštas – normalus pokytis dinaminiame puslapyje gali būti traktuojamas kaip problema ir tokių pranešimų kiekis bus per didelis [13, 7].

1.3.3. DOM analizė

Puslapio dokumentų objektų modelis (angl. *Document Object Model*, *DOM*), apibrėžiantis loginę *HTML* dokumento struktūrą, gali būti panaudotas remiantis jo programų sąsaja (*Application Programming Interface*, *API*) analizuojant pakeistą struktūrą. Lyginant su *diff* algoritmo grįstu metodu, kuris tikrina visus gautus simbolius, *DOM* tikrinimo metodas yra abstraktesnis. Šis metodas tikrina ar pasikeitė interneto svetainės struktūra vietoje viso turinio. Taip išvengiama dinaminėse svetainėse esančių pokyčių, kurie dažniausiai keičiasi tik matomu (tekstiniu) turiniu, bet ne *DOM* struktūra. Šis modelis sudaro puslapio modelį, kurį tikrinimo metu svetainė turi atitikti. Norint naudoti *DOM* struktūros analizės modelį, svetainė turi turėti stabilią ir nekintamą struktūrą. Matomo turinio pokyčiai nėra tikrinami [13].

1.3.4. Statistinė analizė

Yra apibūdinti ir sudėtingesni algoritmai turinio pokyčio aptikimui, kurie remiasi statistine analize. Pavyzdžiui, panaudojant *n-gram* (šiuo atveju *2-gram*) statistinį metodą, nustatantį dažniausiai pasikartojamas vietas puslapyje. Šis metodas pirma turi būti apmokytas ir turi būti sukurti puslapio modeliai, tuomet puslapius galima lyginti. Šio *2-gram* modelio pranašumas – sukurtus modelius galima pritaikyti dinamiškai besikeičiančioje interneto svetainėje. Tačiau toks dinaminis modelio keitimas yra ir trūkumas, kadangi tokiu būdu atsiranda daugiau neteisingų perspėjimų ir pastovus modelio perskaičiavimas reikalauja daugiau kompiuterinių skaičiavimo resursų. Taip pat šiuo metodu galima analizuoti ir teksto turinį. Jeigu piktavalis įkėlė savo žinutę nepakeitęs struktūros, tokį nežymų išdarkymą galima atlikti pasitelkiant šį statistinį metodą, kuris dažniausiai ir yra naudojamas pagal šią kalbos analizės paskirtį [7, 13, 14].

1.3.5. Vizuali analizė

Interneto svetainės vaizdo palyginimas gali būti įgyvendintas naudojant tiek algoritmus, tiek mašininį mokymą, tačiau šio metodo esmė ta pati – turint puslapio vaizdą palyginti su prieš tai užfiksuotu vaizdu. Vaizdo palyginimas turi pranašumą, kadangi į vizualią informaciją aukščiau pateiktuose metoduose nėra atsižvelgiama, kai pagrindinis faktorius yra pakeistas vaizdas, kadangi svetainės naudotojai tai ir mato. Tačiau toks metodas potencialiai gali išnaudoti daug resursų, kadangi būtina saugoti vaizdų failus bei vaizdo analizė reikalauja didelių skaičiavimo resursų. Algoritminis vaizdo palyginimas remiasi pikselių palyginimu. Kiekvienas pikselis yra lyginamas su ankstesnės versijos vaizdu ir taip randamas pokytis. Šį lyginimą galima optimizuoti sumažinant skaičiavimų kiekį konvertuojant pikselius tik į pilko atspalvio (angl. *grayscale*) atitikmenį. Vaizdo failai gali būti suglaudinti naudojant *Joint Photographic Experts Group* (*JPEG*) ar kitus formatus [15].

1.3.6. Mašininio mokymo ir dirbtinio intelekto puslapio analizė

Mašininiui mokymui reikalinga didelė duomenų imtis. Šiems duomenis apdoroti ir klasifikuoti dažnai naudojama *n-gram* teksto analizė, dažniausiai *2-gram* arba *3-gram* [13, 16]. Tai reiškia, kad modeliui sukurti būtina pirma gauti teksto statistinės analizės modelį, kuris naudojamas mašininio mokymo mokymui. Tai reikalauja didelių kompiuterinių resursų. Tokių metodų tikslumas priklauso nuo duomenų kokybės, tačiau pagal analizuotus straipsnius, teisingų aptikimų yra apie 90 proc.

1.4. Realizuoti žiniatinklio turinio pakeitimo aptikimo įrankiai

Žiniatinklio turinio pakeitimo aptikimo įrankiai yra automatizuota programinė įranga, skirta atrasti ir pranešti apie interneto puslapyje atsiradusį pakeitimą. Tokius įrankius gali naudoti administratoriai, norintys užtikrinti puslapio turinio ir struktūros vientisumą, tiek verslininkai ir kiti asmenys, norintys sekti konkurentų puslapius bei juose esantį turinį (pavyzdžiui, produktų kainas).

1.4.1. WebCQ

WebCQ – vienas pirmųjų pakeitimo aptikimo įrankių, paskelbtų 2000 m. metais. Šio įrankio unikalumas – tarpinio serverio (angl. *proxy*) panaudojimas [17]. *WebCQ* pokyčio aptikimas vyksta trimis etapais:

1. Objektų išgavimas – puslapio atsisiuntimas;
2. Objekto analizė – puslapio turinio ir struktūros pakeitimų tikrinimas;

3. Puslapio talpinimas į tarpinį serverį [17].

Tokia sistema, naudojanti tarpinius serverius, gali būti kartu ir lengviau plečiama. Objektų išgavimo etape agentas (straipsnyje vadinamas angliškuu terminu *sentinel*) aptinka duomenis, esančius *HTML* žymose: lentelės, sąrašo, paragrafo, nuorodos ir nuotraukos. Tekstinė informacija papildomai atrenkama ir tikrinama *regex* pagalba [17]. Šiame straipsnyje nėra minimas metodo tikslumas. Metodo idėja šiuo metu yra plačiai naudojama, pavyzdžiui, kaip pagrindinė įmonės *Cloudflare* apsaugos ir greitaveikos spartinimo priemonė.

1.4.2. Kiti aptikimo metodai akademinėje literatūroje

Yra sukurtas įrankis pavadinimu *Web Defacement and Intrusion Monitoring Tool* (*WDIMT*), kuris tikrina puslapyje esančius failus [18]. Šis metodas išsaugo puslapį ir jo kontrolės sumą su maišos funkcijos algoritmu. Turint čia informaciją, periodiškai yra tikrinamas nurodytas puslapis apskaičiuojant jo kontrolinę sumą. Atradus neatitikimą, interneto svetainės serveryje esantys HTML failai yra pakeičiami į išsaugotus įrankio failus, kurie, tikėtina, jog yra teisingi.

Dar vienas sukurtas metodas, įgyvendinantis prižiūrinčio serverio funkcionalumą, kuris tikrina ar administratorius perdavė kontrolės kintamojo reikšmę. Jei kontrolės kintamasis neperduotas kontrolės serveriui, inicijuojamas failų atstatymas arba trynimas [19].

Grafų analizės metodai gali būti pritaikyti analizuojant HTML DOM struktūrą. Panaudojami svoriai kiekvienoje grafo viršūnėje. Kadangi DOM struktūra iš savęs yra hierarchinė, grafas yra medžio struktūros [20]. Grafo medžių viršūnėse saugomi svoriai yra palyginami tarpusavyje ir taip nusprendžiama, ar turinys buvo pakeistas pakankamai, kad būtų traktuojamas, kaip piktavalis turinio pakeitimas.

Mašininio mokymo metodas, apmokytas nepakeisto ir nelegaliai pakeisto turinio interneto svetainėmis, siekia didesnį nei 93 proc. tikslumą, priklausomai nuo pasirinkto *n-gram* ir mašininio mokymo algoritmų (*Naïve Bayes* arba *J48*) [16].

Kombinuotas kelių metodų sujungimas tikrinimo ir klasifikavimo tikslais naudojimas didina metodo tikslumą [13]. Šiame metode nustatytas tikslumas yra 98,80 proc. Tačiau tokiam tikslumui turėjo būti kuriami modeliai, specifiškai taikomi Vietnamo interneto svetainėms. Tai reiškia, kad skirtingų pobūdžių interneto puslapiams turi būti taikomi ir kuriami skirtingi modeliai nežinant kuris bus tiksliausias.

Dar vienas kombinuotų modelių sprendimas naudoja vizualinę informaciją kartu su treniruotais neuroniniais tinklais [21]. Tiek vizualinei informacijai, tiek tekstinei, analizės metu vykdoma klasifikacija, pagal kurią yra nusprendžiama ar puslapis yra su netinkamais turinio pakeitimais. Šio metodo tikslumas yra 97,49 proc.

Kombinuotas vizualios analizės, vertinamos mašininio mokymo pagalba sprendimas *Meerkat* vizualiai analizuoja puslapio vaizdą neuroninių tinklų pagalba [22]. Šiai sistemai naudojamos interneto svetainės su žinomu turinio pakeitimu ir su nepakeistu turiniu, padaroma kiekvieno puslapio nuotrauka ir pagal tai neuroninis tinklas yra apmokomas. Šio modelio tikslumas yra tarp 97,422 proc. ir 98,375 proc.

1.4.3. Komerciniai sprendimai

Šiuo metu yra sukurta nemažai įrankių, kurie nebūtinai reklamuoja save kaip žiniatinklio sudarkymo ar kitokio turinio pakeitimo aptikimo produktais. Pavyzdžiui, įmonės, vystančios prekybos verslą ir turinčios savo el. parduotuvę, darbuotojai gali pasinaudoti panašiais įrankiais sekti konkurentų el. parduotuvės produktų asortimentą ir jų kainas. Toks įrankis formaliai patenka į pakeitimų aptikimo įrankių grupę, tačiau šiame magistriniame darbe problema apžvelgiama dėl piktavalių ar žmogiškų svetainės administratorių klaidų atsiradusių nenorimų pakeitimų. Tiriant įvairių įrankių funkcionalumą buvo atrinkti tik suniokojimui tiesiogiai ar netiesiogiai aptikti skirta įranga (žr. 1 lentelė).

1 lentelė. Įgyvendintų įrankių palyginimas

Įrankis	**Tikrinimas**				**Papildomos paslaugos**
	Vizualus	**Tekstinis**	**Struktūrinis**	**Šaltinio**	
Fluxguard[1]	✓	✓	✓	✓	Tinklo pokyčių analizė. Rezultatų pateikimas su dirbtiniu intelektu. Pateikia *Google Lighthouse* auditą. Tikrina atakų vektorius.
Site24x7 aptikimo įrankis[2]	✓	✓	✓	✓	Automatinis slenkstinių ribų nustatymas. Domeno, pasiekiamumo, API, sertifikatų, serverio paslaugų stebėjimas.
ManageEngine Applications Manager[3]	✓	✓	✓	✓	Tikrina realių naudotojų atsako laikus, pasitaikiusias programines klaidas, serverio, duomenų bazės tinkamą veikimą. Prižiūri serverio teikiamų paslaugų statusą bei sertifikatų galiojimą. Naudoja dirbtinį intelektą pranešimams. Tikrina atakų vektorius.
Hexowatch[4]	✓	✓	✓	✓	Domeno tikrinimas. Tikrinimas pasitelkia dirbtinį intelektą. API tikrinimas.
Visualping[5]	✓	Tik vizualiai	X	X	
ChangeTower[6]	✓	✓	✓	✓	

1.4.4. Automatizuoti testai

Įmanoma turinio pakeitimo aptikimą vykdyti panaudojant automatizuotų naudotojo sąsajos testų karkasą. Vienas populiariausių tokio tipo karkasų yra *Selenium*[7]. Tačiau tokio tipo sprendimas reikalautų atskiro projekto, kuriame turi būti nurodyti atskiri HTML elementai, kurie turi atitikti bet kokią suprogramuotą logiką. Tai yra geras sprendimas norint užtikrinti tinkamą puslapio veikimą ir

[1] https://fluxguard.com
[2] https://www.site24x7.com/monitor-webpage-defacement.html
[3] https://www.manageengine.com/products/applications_manager
[4] https://hexowatch.com
[5] https://visualping.io
[6] https://changetower.com
[7] https://www.selenium.dev

jo integralumą, tačiau reikalauja sąlyginai daug laiko automatizuotų testų kodo rašymui kiekvienam projektui ir jame esantiems puslapiams.

1.4.5. Turinio pakeitimo aptikimas naudojant šablonus

Atlikus literatūros apžvalgą nebuvo rasta sprendimų, kurie naudotų šabloną, pagal kurį būtų atliekama turinio pakeitimo analizė. Yra užsimenama apie pačių failų integralumo tikrinimą panaudojant maišos funkcijas ir failų struktūros tikrinimą. Tačiau nebuvo rasta nieko apie pačių šablonų panaudojimą turinio pakeitimo tikrinimui.

1.5. Puslapių šablonų veikimo modeliai ir sintaksė

Dalis vystomų interneto svetainių naudoja šablonus. Šablonai yra aprašomi specialiai sukurtose šablonų kalbose, kurios leidžia integruoti statinį turinį su dinaminiu. Šie šablonai yra valdomi šablono variklio, kuris atsakingas už standartinio HTML failo generavimą iš šabloninio failo. Šiuos HTML failus iš šablonų galima generuoti tiek naudotojo kompiuteryje (angl. *front-end*), tiek serverio pusėje (angl. *back-end*). Šiame metode yra naudojamas serverio pusės šablonų generavimas, kadangi šablonas ir gaunamas turinys skiriasi naudotojui pateikus užklausą. Jeigu šablonas generuojamas naudotojo kompiuteryje, serveris siunčia šabloną nieko negeneruodamas, todėl šablonas ir pirminis gautas failas yra vienodi. Skirtingos technologijos įgyvendina šablonų veikimą skirtingai. Jų sintaksės skiriasi atitinkamai.

1.5.1. Mustache sintaksė

Egzistuoja keli šablonų sintaksių tipai. Vienas populiariausių yra *Mustache* sintaksė, išvertus anglišką žodį būtų ūsai, dėl naudojamų skliaustelių „{“ ir „}“, kadangi pavertus juos ant šono primena ūsus, tačiau oficiali termino kilmė nežinoma. Šis sintaksės tipas pasitaiko dažniausiai tarp šablonų variklių. Nors *Mustache* kilo iš naudotojo pusėje generuojamo šablono modelio, ši sintaksė plačiai naudojama ir serverio pusės šablonuose [23, 24].

Mustache sintaksė [25]

```
1. Hello {{name}}
2. You have just won {{value}} dollars!
3. {{#in_ca}}
4. Well, {{taxed_value}} dollars, after taxes.
5. {{/in_ca}}
```

Šioje sintaksėje matome paprastą tekstą ir įterpiamas dinamines vietas tarp „{{“ ir „}}“ skliaustelių (dažnai žymima „{{ ... }}“). Ši sintaksė naudojama didžiojoje daugumoje *PHP* karkasų. Taip pat sintaksė papildomai naudoja „{%“ ir „%}“ bei kitokio tipo įterptinę logiką. Tokie PHP karkasų naudojami šablonų variklių pavyzdžiai yra *Django, Jinja, Twig, Flask* ir *Laravel Blade*.

Django[8]

```
1. My first name is {{ first_name }}. My last name is {{ last_name }}.
```

```
1. {% if user.is_authenticated %}Hello, {{ user.username }}.{% endif %}
```

[8] https://docs.djangoproject.com/en/5.0/topics/templates

Jinja[9]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>My Webpage</title>
</head>
<body>
    <ul id="navigation">
    {% for item in navigation %}
        <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
    {% endfor %}
    </ul>

    <h1>My Webpage</h1>
    {{ a_variable }}

    {# a comment #}
</body>
</html>
```

Twig[10]

```
{% for user in users %}
    * {{ user.name }}
{% else %}
    No users have been found.
{% endfor %}
```

Flask[11]

```
<!doctype html>
<title>{% block title %}{% endblock %} - Flaskr</title>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
<nav>
  <h1>Flaskr</h1>
  <ul>
    {% if g.user %}
      <li><span>{{ g.user['username'] }}</span>
      <li><a href="{{ url_for('auth.logout') }}">Log Out</a>
    {% else %}
      <li><a href="{{ url_for('auth.register') }}">Register</a>
      <li><a href="{{ url_for('auth.login') }}">Log In</a>
    {% endif %}
  </ul>
</nav>
<section class="content">
  <header>
    {% block header %}{% endblock %}
  </header>
  {% for message in get_flashed_messages() %}
    <div class="flash">{{ message }}</div>
  {% endfor %}
  {% block content %}{% endblock %}
</section>
```

[9] https://jinja.palletsprojects.com/en/3.1.x/templates

[10] https://twig.symfony.com

[11] https://flask.palletsprojects.com/en/3.0.x/tutorial/templates

Go Templates[12]

```
1. <p><strong>Pets:</strong> {{ . | len }}</p>
2. {{ range . }}
3. <hr />
4. <dl>
5.     <dt>Name</dt>
6.     <dd>{{ .Name }}</dd>
7.     <dt>Sex</dt>
8.     <dd>{{ .Sex }} ({{ if .Intact }}intact{{ else }}{{ if (eq .Sex "Female") }}spayed{{ else
}}neutered{{ end }}{{ end }})</dd>
9.     <dt>Age</dt>
10.    <dd>{{ .Age }}</dd>
11.    <dt>Breed</dt>
12.    <dd>{{ replace .Breed “/” “ & ” }}</dd>
13. </dl>
14. {{ end }}
```

1.5.2. Ruby on Rails, Embedded Ruby sintaksė

Skirtinga sintaksė naudojama Ruby on Rails karkase. Čia naudojamos į *PHP* panašios įterpiamos „<% … %>“ kodo dalys. Tačiau bendra struktūra išlieka tokia pati, kaip ir *Handlebars* atveju.

Embedded Ruby (*ERB, eRuby*)[13]

```
 1.   <html>
 2.     <head><title>Ruby Toys -- <%= @name %></title></head>
 3.     <body>
 4.
 5.       <h1><%= @name %> (<%= @code %>)</h1>
 6.       <p><%= @desc %></p>
 7.
 8.       <ul>
 9.         <% @features.each do |f| %>
10.           <li><b><%= f %></b></li>
11.         <% end %>
12.       </ul>
13.
14.       <p>
15.         <% if @cost < 10 %>
16.           <b>Only <%= @cost %>!!!</b>
17.         <% else %>
18.            Call for a price, today!
19.         <% end %>
20.       </p>
21.
22.     </body>
23.   </html>
```

1.5.3. Senesnė Laravel Blade sintaksė

Laravel karkaso naudojamas *Blade* šablonų variklis naudoja išplėstą *Mustache* sintaksę kartu su „@” ženklo įterpiamu kodu.

Blade komponentai (senesnėje versijoje)[14] naudoja „@” ženklus nurodyti įvairių komponentų pradžią ir pabaigą. Pavyzdžiui:

```
1. <div class="alert alert-danger">
2.     {{ $slot }}
3. </div>
```

[12] https://www.digitalocean.com/community/tutorials/how-to-use-templates-in-go
[13] https://docs.ruby-lang.org/en/2.3.0/ERB.html#class-ERB-label-Ruby+in+HTML
[14] https://laravel.com/docs/5.8/blade#components-and-slots

```
@component('alert')
    <strong>Whoops!</strong> Something went wrong!
@endcomponent
```

```
@component('alert')
    @slot('title')
        Forbidden
    @endslot

    You are not allowed to access this resource!
@endcomponent
```

Blade If sąlygos[15]

```
@if (count($records) === 1)
    I have one record!
@elseif (count($records) > 1)
    I have multiple records!
@else
    I don't have any records!
@endif
```

Blade sekcijų direktyvos

```
@hasSection('navigation')
    <div class="pull-right">
        @yield('navigation')
    </div>

    <div class="clearfix"></div>
@endif
```

1.5.4. Thymeleaf sintaksė

Visiškai kitokio pobūdžio sintaksė naudojama *Java* aplinkoje *Thymeleaf* šablonų variklis. Ši sintaksė talpinama į HTML objektų atributus, taip išlaikant originalią HTML struktūrą šablone, kuri yra dinamiškai keičiama šablonų variklio.

Thymeleaf[16]

```
<table>
  <tr>
    <th>NAME</th>
    <th>PRICE</th>
    <th>IN STOCK</th>
    <th>COMMENTS</th>
  </tr>
  <tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">
    <td th:text="${prod.name}">Onions</td>
    <td th:text="${prod.price}">2.41</td>
    <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
    <td>
      <span th:text="${#lists.size(prod.comments)}">2</span> comment/s
      <a href="comments.html"
         th:href="@{/product/comments(prodId=${prod.id})}"
         th:if="${not #lists.isEmpty(prod.comments)}">view</a>
    </td>
  </tr>
</table>
```

[15] https://laravel.com/docs/10.x/blade#if-statements

[16] https://www.thymeleaf.org/doc/tutorials/3.1/usingthymeleaf.html

1.5.5. ASP.NET Razor sintaksė

Kitokio tipo šablonų sintaksę naudoja Microsoft *ASP.NET* karkaso *Razor* šablonų variklis. *Razor* šablonų variklio failai turi savo plėtinius – *.cshtml C#* ir *.vbhtml VisualBasic* serverio pusės programavimo kalbai, o *.razor* – klientinės pusės vykdymo projektams. Pagrindinis ženklas šiame šablone yra „@“. Po šio ženklo sekančios reikšmės bus interpretuojamos *Razor* šablonų variklio pagalba [26, 27]. Jei iš karto seka atidarantis skliaustas „{“, tai reiškia, kad tai yra kodo blokas, kuriame gali būti ir HTML sintaksės. Panaši sintaksės logika yra ir *VBNET Razor* šablonuose su pačios programavimo kalbos semantiniais sintaksės skirtumais [28].

Razor sintaksės pavyzdžiai [28]:

```
<!–Single statement blocks  →
@{ var total = 7; }
@{ var myMessage = „Hello World“; }

<!–Inline expressions →
<p>The value of your account is: @total </p>
<p>The value of myMessage is: @myMessage</p>

<!–Multi-statement block →
@{
    var greeting = „Welcome to our site!“;
    var weekDay = DateTime.Now.DayOfWeek;
    var greetingMessage = greeting + „ Today is: „ + weekDay;
}
<p>The greeting is: @greetingMessage</p>
```

Razor If sąlygos [28]:

```
@if (value % 2 == 0)
{
    <p>The value was even.</p>
}
else if (value >= 1337)
{
    <p>The value is large.</p>
}
else
{
    <p>The value is odd and small.</p>
}
```

Šiuose pavyzdžiuose matome, jog tai yra *C#* ir *HTML*, o ši kombinacija yra *Razor* sintaksėje, kuri naudoja *.cshtml* failo plėtinį [27].

1.6. Esamų žiniatinklio turinio pakeitimo aptikimo metodų analizės išvados

Žiniatinklio atakų tipų, esamų įgyvendintų metodų ir puslapių šablonų analizės išvados:

1. Kontrolės sumos ir *diff* algoritmai naudingi tik statinių puslapių analizei, o statistinė bei vizuali analizė reikalauja daug kompiuterinės įrangos resursų. Analizė pagal DOM struktūrą netikrina pačio turinio;
2. metodai, kurie naudoja daugiau nei vieną tikrinimo metodą dažniausiai aptinka pakeitimus didesniu tikslumu;
3. visi apžvelgti metodai turi didesnį nei 90 proc. pakeitimo aptikimo tikslumą;
4. komerciniai sprendimai dažniausiai naudoja daugiau nei vieną metodą puslapio pakeitimo analizei atlikti;

5. visi šablonai turi atitinkamą sintaksę, žyminčią šablono įterptinę sritį;
6. šablonų varikliai interpretuoja šablono failą, dinamiškai pakeičia jo šablonines dalis ir generuoja HTML struktūrą.
7. atlikus analizę nustatyta, kad literatūros šaltiniuose nėra aptikta jokio metodo, kuris naudotų puslapio tikrinimą pagal puslapio šabloną.

Šio magistrinio baigiamojo darbo tikslas: Sukurti žiniatinklio pakeitimo aptikimo metodą, kuris tikrintų interneto puslapį pagal šablonus ir juose esančią HTML struktūrą bei teksto turinį.

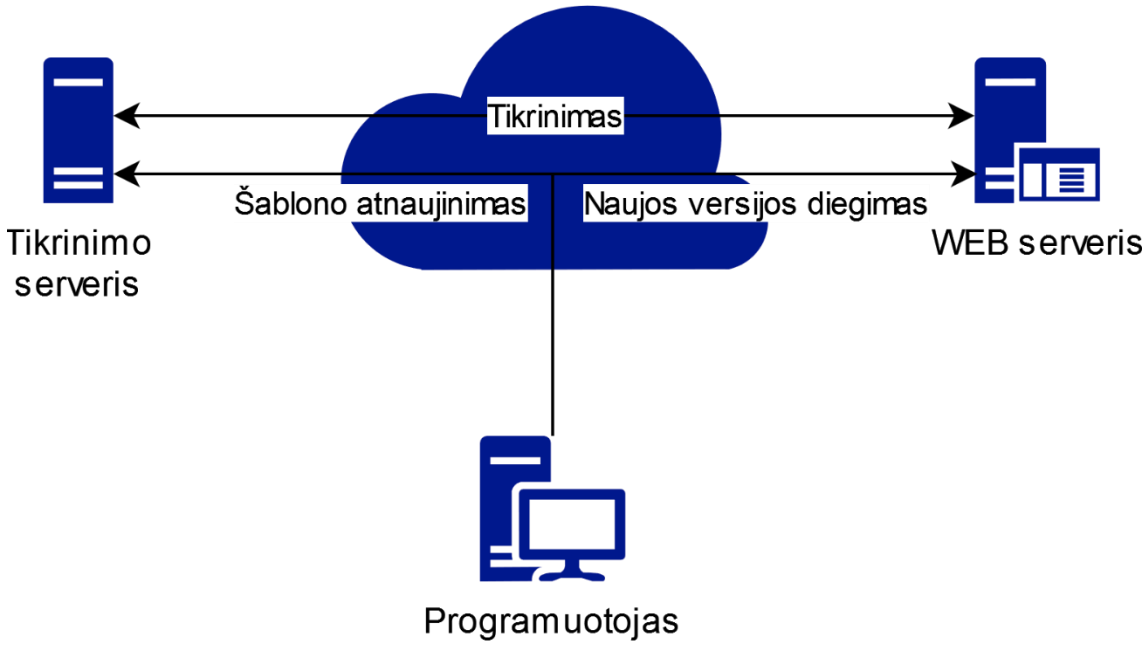
Uždaviniai:

1. detalizuoti šablonų žiniatinklio turinio pakeitimo tikrinimo pagal šablonus algoritmą;
2. realizuoti siūlomo metodo algoritmą programiniame kode;
3. sukurti siūlomo žiniatinklio turinio pakeitimo aptikimo metodo prototipą;
4. atlikti prototipo sintetinius bandymus tikslumui ir greitaveikai įvertinti.

2. Siūlomas žiniatinklio pakeitimo aptikimo metodas

Šabloniniai failai dažniausiai būna keičiami svetainės atnaujinimo metu, todėl normalaus veikimo metu šie failai neturėtų būti keičiami. Dėl šios priežasties tokie šablonų failai yra gera vieta tikrinimui. Teoriškai, sugeneruotas HTML puslapis pagal šabloną turi atitikti savo šabloną 100 proc. tikslumu.

Šablonas turi būti tikrinamas su gautu puslapiu (žr. 6 pav.). Sistemos įdiegimas susideda iš atskiro tikrinimo vietos – serverio. Į šį tikrinimo serverį būtina įkelti tuos pačius šablonus, kurie yra naudojami interneto svetainėje. Šablonai turi būti sujungiami su atitinkamomis puslapio nuorodomis. Vystant interneto puslapį – kiekvieną kartą keičiamas šablonas turi būti atnaujinamas taip pat ir tikrinimo serveryje. Šį šablonų įkėlimą galima automatizuoti naudojant nuolatinio diegimo (angl. *continuous delivery / deployment, CD*) metodus.



6 pav. Šablonų tikrinimo koncepcinis modelis

Metodą įgyvendinanti palyginimo programa gali būti tiek kitame, tiek tame pačiame serveryje arba net gali būti kviečiama pačioje interneto puslapio vykdomojoje programoje. Tačiau saugiausias pasirinkimas yra nuotolinis serveris, į kurį yra keliamas naujausias teisingas šablonas. O tikrinimo serveris periodiškai kviečia ir lygina gautus interneto svetainės puslapius su atitinkamu šablonu.

Šiam tikrinimui reikalinga papildoma programinė įranga tikrinimo dalyje. Jos veikimo eiga tokia:

1. Tikrinimui skirtame serveryje įdiegiama ir sukonfigūruojama šablonų tikinimo programinė įranga.
2. Interneto svetainės atnaujinimo metu tikrinimo serveryje esantys šablonai taip pat atnaujinami.
3. Tikrinimo programinė įranga kviečia atitinkamus puslapius ir juos lygina pagal šablonus.

Šio naujo pakeitimų aptikimo metodo principas – palyginti gautą puslapį su šablonu. Tačiau šablonas daugeliu atveju nebus toks pats, kaip statinis puslapis, kurį visą galima palyginti kontrolės suma ar *diff* algoritmu. Dėl to pirma reikia turėti šių dviejų dalių – HTML puslapio ir šablono – vienodai formuojamą modelį.

2.1. Tikrinimo pagal šablonus veikimas

Palyginimo funkcija turi grąžinti atsakymą, ar puslapis atitinka šabloną, ar ne. Kadangi metodas remiasi tik šabloniniu failu ir neturi visų duomenų, kuriais yra užpildomas realus puslapis, tokias duomenų įterpimo vietas reikia praleisti. Pavyzdžiui, esant loginei sąlygai *if* ar ciklui *foreach* šablone ir norint gauti 100 proc. atitinkantį atsakymą, reiktų sukompiliuoti šį failą su visais susijusiais modeliais ir duomenimis. Kadangi šio metodo tikslas – analizuoti puslapį tik pagal šablono failą, šie papildomai įterpiami duomenys turi būti ignoruojami. Toks modelio veikimas gali būti traktuojamas kaip atgalinės inžinerijos (angl. *reverse engineering*) metodo taikymas šablonų kompiliavimui. Tik šiuo atveju nėra mašininio kodo, o vietoje sukompiliuoto kodo yra HTML žymos ir tekstinis turinys.

Kai šablone yra tik HTML sintaksė – tai yra statinis puslapis ir lyginimas su interneto puslapiu įvyks bet kokiu tiesioginiu palyginimu – maišos funkcija, *diff* algoritmai ar tiesiogiai lyginant visus simbolius. Kitu atveju HTML yra generuojamas dinamiškai.

Konceptualiai, HTML sintaksė, įkelta į kintamąjį ar kviečiama per kitas funkcijas, yra tiesiogiai atitinkanti statinio puslapio HTML elementus. Pavyzdžiui, turint tokį PHP kodą:

```
1. <?php
2.     echo '<p>Teksto pastraipa</p>';
3. ?>
```

Jeigu šablone yra apibrėžtas kintamasis, galima bandyti rasti jo panaudojimo vietą. Iš esmės toks sprendimas jau būtų tam tikras šablono interpretatorius ar kompiliatorius. Pavyzdžiui, jeigu naudojama kintamoji reikšmė yra žinoma (yra šablone, ne duomenyse), galima būtų tikrinti sąlygą pagal jos reikšmę:

```
1. <?php
2. $amzius = 18;
3.
4. if ($amzius >= 18) {
5.     echo '<p>Jūs esate pilnametis.</p>';
6. } else {
7.     echo '<p>Jūs esate nepilnametis.</p>';
8. }
9. ?>
```

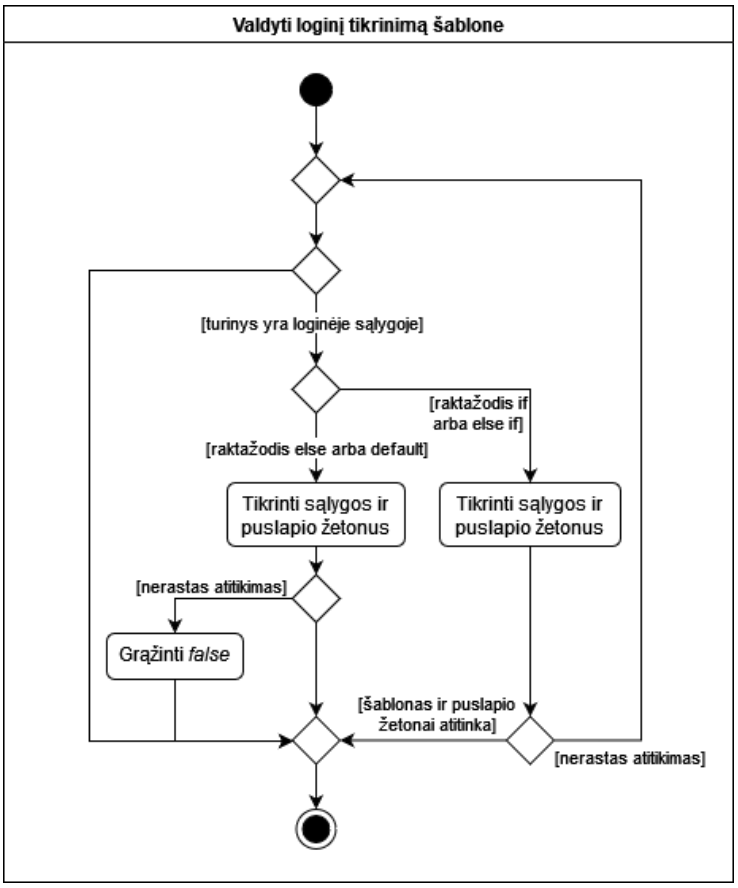
Dar vienas pavyzdys *Thymeleaf* sintaksės šablone:

```
1. <div th:if="${1 == 0}">
2.     Vienas lygu nuliui.
3. </div>
4. <div th:else-if="${0 == 1}">
5.     Nulis lygu vienam.
6. </div>
7. <div th:else>
8.     Nulis nelygu vienam.
9. </div>
```

Čia gautas sukompiliuotas rezultatas yra paskutinė *else* žyma. O ją galima būtų tikrinti, kadangi visi duomenys yra statiniai ir pasiekiami šablone, nėra gaunami iš duomenų bazės ar karkaso. Tačiau dažniausiu atveju šios reikšmės bus nežinomos ir nebus galima nuspėti kokia bus sukompiliuota reikšmė. Abejais minėtais atvejais būtų įmanoma pasitelkti dirbtinio intelekto pagalbą nuspėjamai fiktyvaus kompiliavimo rezultatui gauti ar patikrinti rezultatą su šablonu. Iš vienos pusės, tokiame sprendime atsirastų nenuspėjamumas, kai dirbtinio intelekto modelis ne visada teisingai nuspės koks rezultatas turėtų būti teisingas, ypač didelėse, sudėtingose sąlygose. Iš kitos pusės, šio modelio įgyvendinimas nereikalautų atskiro įgyvendinimo kiekvienai šablonų technologijai.

2.1.1. Loginės sąlygos šablone tikrinimas

Teoriškai, tokių loginių sąlygų, kaip *if*, *else if*, *switch* ir t.t., priklausomai nuo sąlygos rezultato, viduje esantis kodas gali būti vykdomos arba nevykdomas. Išimtis gali būti tik tokioje sąlygoje, kuri turi būtinai vykdomą kodo kontrolės srautą, pavyzdžiui, *else* ir *switch* naudojamas *default* raktažodis (žr. 7 pav.). Tai reiškia, kad jeigu tikrinimo metu sąlygos vykdymo bloke esantys HTML elementai nėra kviečiamame puslapyje – vadinasi, didelė tikimybė, jog sąlyga tiesiog nebuvo įvykdyta.



7 pav. Loginės sąlygos šablone tikrinimo veiklos diagrama

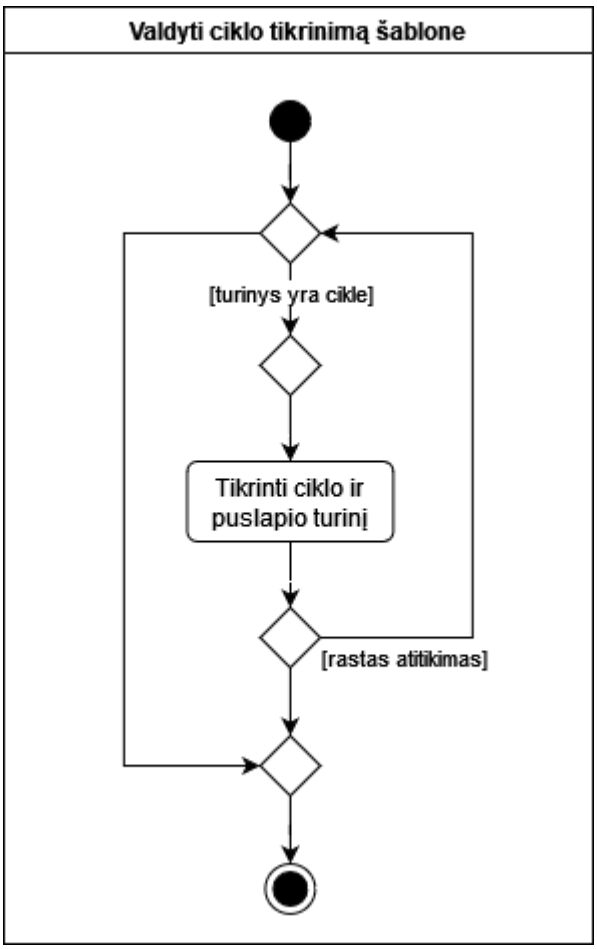
2.1.2. Turinį generuojančio ciklo šablone tikrinimas

Panašus principas taikomas ir ciklų logikoje. Teoriškai, ciklas gali būti vykdomas nulį arba begalybę kartų. Tai reiškia, kad visas cikle esantis HTML turinys gali ir neatsirasti interneto puslapyje arba gali atsirasti *n* kartų (žr. 8 pav.). Todėl tikrinimo metodas turi palyginti cikle esantį HTML turinį ir tikrinti puslapyje esančius atitinkamus elementus iki kol bus rastas neatitikimas. Jei šis neatitikimas įvyksta pirmojo tikrinimo metu, tai reiškia, kad ciklo turinys nebuvo įvykdytas ir toliau tikrinamas turinys už ciklo esantiems elementams pagal sekančio žetono palyginimą.

Cikle taip pat galima atrasti kiek kartų bus (ar nebus) ciklo vykdymų, jei visos reikiamos reikšmės yra šablone. Pavyzdžiui, turint tokį *Embedded Ruby* šabloną:

```
1. <% for skaicius in 1..5 %>
2.   <p>Skaičius: <%= skaicius %></p>
3. <% end %>
```

Galima nuspėti kiek kartų HTML paragrafo žyma bus kartojama ir kokia reikšmė atsiras kiekviename teksto turinio gale.



8 pav. Ciklo šablone tikrinimo veiklos diagrama

2.1.3. Tekstinio turinio palyginimas tarp puslapio ir šablono

Lyginimo metu tekstas turi visiškai atitikti tai, kas yra šablone. Šis tikrinimas sudėtingėja, kai HTML teksto turinyje atsiranda kintamieji. Turint tik šablono failą nėra galimybės sužinoti kokia reikšmė šioje vietoje gali būti, jei duomenys yra įkeliami iš duomenų modelio, todėl tokias vietas reikia praleisti iki kito žinomo teksto arba jo pabaigos. Tačiau tokiu atveju iškyla vienas neišsprendžiamas atvejis – kintamojo sukompiliuotos gautos reikšmės simbolyje bus toks pats pasikartojantis turinys, kaip ir po jo sekančioje tekstinėje dalyje. Pavyzdžiui, turime tokį *Razor* šablono failą:

```
1. @{
2.     string keblusPasisveikinimas = "labas!"; // tariama reikšmė iš duomenų modelio, o ne šablone
3. }
4. <p>O, @keblusPasisveikinimas!</p>
```

Šis šablonas susikompiliuoja į tokį HTML failą:

```
1. <p>O, labas!!</p>
```

Tikrinimo metu vietoje kintamojo reikšmės atsiradusiam turiniui bus ignoruojamas tekstas iki pirmojo po jo sekančio šauktuko simbolio „!“, tačiau sukompiliuotame variante šis simbolis jau yra kintamojo paskutinis simbolis. Vadinasi tikrinimo algoritmas ignoruos tik tekstą „labas“, o ne „labas!“ ir įvyks teksto neatitikimas – puslapis bus traktuojamas kaip nelygus šablonui.

Kitu atveju, kai šablone esančiame kintamajame yra ilgesnis tekstas ir yra panašių žodžių, atitiktis turi būti korektiška. Pavyzdžiui, turint tokį *Razor* šabloną:

```
1. @{
2.     string tekstas = "šis tekstas yra skirtas įkėlimui";
3. }
4. <p>Puslapyje @tekstas yra neskirtas įkėlimui.</p>
```

Šis šablonas yra sukompiliuojamas į tokį HTML failą:

```
1. <p>Puslapyje šis tekstas yra skirtas įkėlimui yra neskirtas įkėlimui.</p>
```

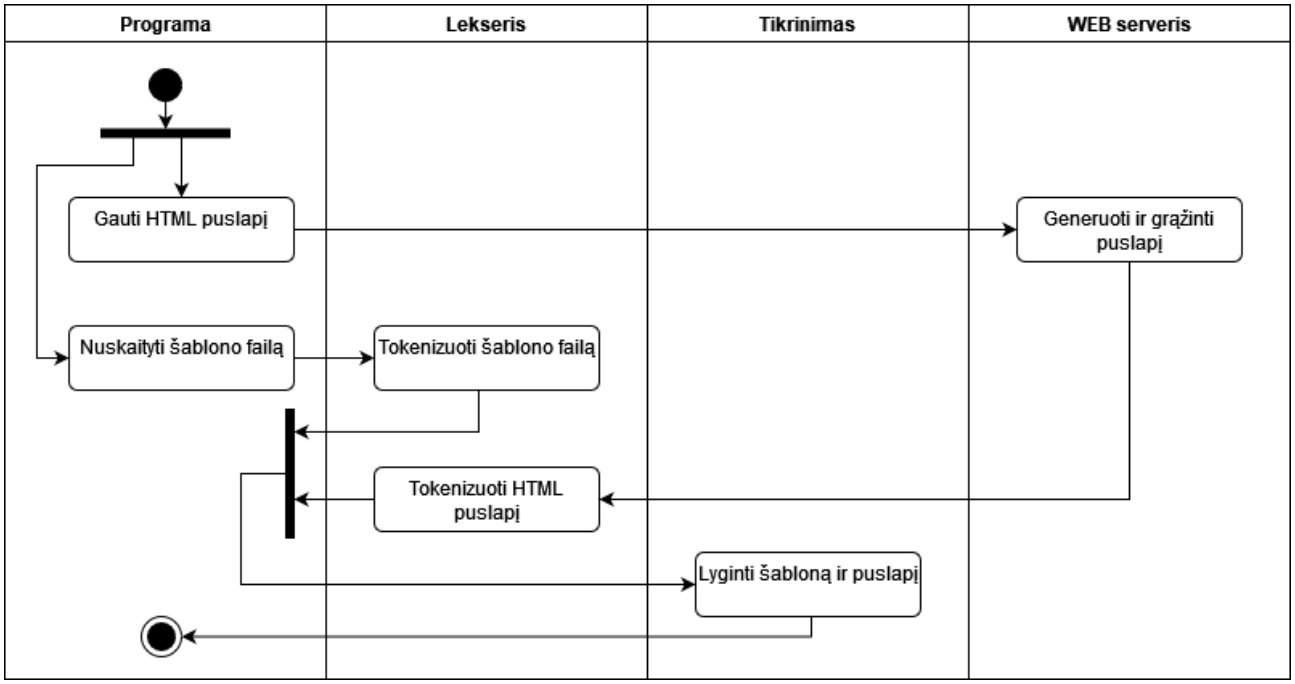
Čia teksto patikrinimas turi tokį eiliškumą: 1) tekstas „Puslapyje “; 2) bet koks tekstas pagal *Razor* žymą su nuoroda į kintamąjį; 3) tekstas „ yra neskirtas įkėlimui.“. Kadangi trečioji teksto dalis visiškai vienoda, tai puslapio simboliai iki šios teksto vietos bus praleisti ir tikrinami toliau.

2.2. Siūlomo pakeitimo aptikimo metodo išvados

1. Siūlomas pakeitimo aptikimo metodas atitikimo tikrinimui naudoja šablono atgalinės inžinerijos rezultatą;
2. siūlomas pakeitimo aptikimo metodas tikrina visas statines šablono atitiktis bei atranda atitinkamą dinamiškai generuojamą HTML turinį;
3. siūlomas pakeitimo aptikimo metodas teoriškai gali pasiekti iki 100 proc. pakeitimo aptikimo tikslumą, jei pakeitimas nėra įvykdytas panaudojant šabloną;
4. siūlomas pakeitimo aptikimo metodas nenaudoja jokio įgyvendinto modelio, todėl tikėtina, jog greitaveika bus spartesnė ir atminties sunaudojimas mažesnis už žinomus metodus.

3. Siūlomo žiniatinklio pakeitimo aptikimo metodo prototipas

Modelio prototipas įgyvendinamas ir aprašomas naudojant *.cshtml Razor* šablonus. Kadangi *Razor* sintaksė susideda iš HTML ir *C#* kodo su atitinkamomis žymėmis, toks prototipas įrodytų, jog yra įmanoma įgyvendinti ir paprastesnių šablonų veikimą, kurie nenaudoja programavimo kodo savyje. Modelio aprašyme minėtas vienodai traktuojamas modelis (žr. 2 aukščiau) čia yra specialiai sukurtas lekseris (žr. 3.1 žemiau), kurio analizės rezultatas yra leksinių žetonų masyvas vienam failui. Po abiejų failų leksinės analizės vykdomas šablono, esančio tikrinimo programos pasiekiamame diske (ar netgi pasiekiamo nuotoliu) ir sugeneruoto bei grąžinto tikrinamos interneto svetainės HTML failo, leksinės analizės rezultatų (žetonų masyvų) lyginimas (žr. 9 pav.). Visas metodas išskaidytas į du pagrindinius žingsnius – leksinę analizę (lekseris) ir tikrinimo algoritmą (tikrinimas), kuriuos kviečia pagrindinė programa (žr. 9 pav., 1 Priedas, 0 ir 5 Priedas).



9 pav. Siūlomo žiniatinklio pakeitimo aptikimo metodo tikrinimo programos proceso veiklos diagrama

Šablonai, naudojantys *Razor* sintaksę, yra interpretuojami ir kompiliuojami. Šiuose šablonuose yra tiek HTML, tiek C# kodo. Visą *.NET* technologijų karkasą kompiliuoja, interpretuoja ir valdo .NET kompiliavimo platforma (dar kitaip žinoma kaip *Roslyn*) [29]. Tačiau, pagrindinė problema šiame prototipe – *Roslyn* yra integruota su visu *.NET* karkasu ir dėl to, dažniausiai, puslapio kompiliavimui reikalingos visos naudojamos C# klasės ir modeliai, pagal kuriuos generuojamas ir kompiliuojamas HTML puslapis iš *Razor* šablono.

3.1. Specializuotas lekseris *Razor* sintaksei

Tam, kad tvarkingai sugeneruoti palyginimo modelį (žetonų masyvą), pagal kurį galima lyginti puslapį su šablonu, reikalingas *Razor* sintaksę palaikantis algoritmas.

Šis lekseris nėra visiškai grynas lekseris, kadangi HTML atributai yra iš karto priskiriami žymai žetono C# klasės struktūroje, o ne lekserio žetonų masyve. Didelė dalis C# funkcinių išraiškų nepalaikomos ir joms šiame projekte nėra daug prasmės palaikyti. Lekseris buvo suprogramuotas be

kitų įgyvendintų lekserių gilios analizės. Tokio kūrimo priežastys: 1) siekta sukurti paprastesnį sprendimą; 2) toks sprendimas būtų potencialiai greitesnis (kadangi neatsižvelgia į visus C# atvejus); 3) dėl mažesnės apimties turėtų išnaudoti mažiau atminties. Sukurtas lekseris pagal savo įgyvendinimą turi panašumų su, pavyzdžiui, *Roslyn* lekseriu, todėl galima teigti, jog yra nepriklausomai pataikyta įgyvendinti universalų ir labiausiai priimtiną lekserio sprendimą.

3.1.1. Roslyn problematika

Norint naudoti *Roslyn* ir jo *Razor* valdymo kodą, būtina turėti šablonui reikalingus modelius. Šis naujas siūlomas pakitimo aptikimo metodas turi būti nepriklausomas nuo modelių ir privalo analizuoti tik nepriklausomus šabloninius failus. Dėl to dauguma *Roslyn* generavimo ir kompiliavimo etapų nėra naudingi šiam metodo įgyvendinimui.

Taip pat *Roslyn* turi lekserį[17]. Tačiau šis lekseris yra skirtas C# kodo analizei[18], nors HTML ar *.cshtml* tipo failus taip pat galima naudoti. *Roslyn* lekseris atrenka simbolius iš C# kodo perspektyvos, todėl, pavyzdžiui, HTML žyma „<p>“ skaidoma į „< mažiau nei“, „turinys p“ ir „> daugiau nei“. Kai iš tiesų tai būtų tiesiog HTML „p“ žyma.

Roslyn yra visos .NET platformos valdymo įrankis, todėl jame yra daug įvairiausių funkcijų, išimčių ir kito kodo, galimai mažinančio našumą. Todėl naujai kuriamo algoritmo tikslas – analizuoti HTML ir tai, kas yra susiję su HTML. Dėl šių priežasčių *Roslyn* nebus naudojamas, nes jame yra našumo bei perteklinės analizės problemų naujam siūlomam metodui.

3.1.2. Neteisingas *Roslyn* žymėjimas

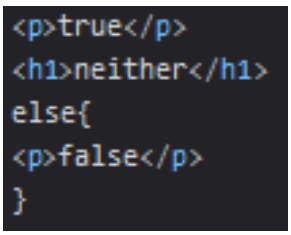
Įgyvendinant specialų *Razor* lekserį buvo atrasta, jog *.NET* naudojama *Roslyn* kompiliavimo platforma neteisingai žymi paprastą tekstą, parašytą kaip, pavyzdžiui, dalis prieš tai buvusios logikos (žr. 10 pav. ir 12 pav.).

```
@if(true){<p>true</p>}
<h1>neither</h1>
else{<p>false</p>}
```

10 pav. Neteisingas sintaksės žymėjimas *.cshtml* faile, *Visual Studio* programavimo aplinkoje

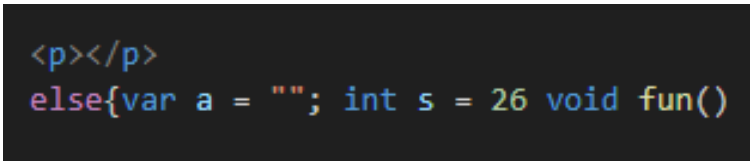
Tačiau, nepaisant nekorektiško žymėjimo iš .NET kompiliavimo platformos, pilnai sukompiliuotas puslapis yra teisingas – paskutinėje eilutėje yra tekstas su HTML „<p>“ žyma (žr. 11 pav.).

[17] https://github.com/dotnet/roslyn/blob/main/src/Compilers/CSharp/Portable/Parser/Lexer.cs
[18] https://github.com/dotnet/roslyn/blob/main/src/Compilers/CSharp/Portable/Syntax/CSharpSyntaxTree.cs



11 pav. Sukompiliuota HTML failo dalis naršyklėje, grąžinama iš serverio.

Šiame neteisingai žymimame pavyzdyje naudojama *Razor Statement* tipo sintaksė, kurioje panaudojamas *Razor* simbolis ir iš karto rašomas C# kalbos raktažodis. Šiame prototipe būtent tokia forma aprašytas kodas nėra palaikomas. Pateikto pavyzdžio atveju C# raktažodis yra *if*. Šiame pavyzdyje įrašyta visada vykdoma sąlyga *true*. Kodo bloke įrašyta HTML žyma. Už kodo bloko seka paprasta HTML žyma ir tuomet, pagal *Razor* sintaksę, seka tekstas „else{“, o po šios dalies seka dar viena HTML žyma, o už jos – tekstas iš vieno simbolio „}“. Šį neteisingą žymėjimą galima pastebėti bet kurioje *.cshtml* failo vietoje įrašant *else*. Tačiau, tik jei prie šio raktažodžio yra skliaustas arba kodo bloko atidaranti žyma („(“ arba „{“), kodo rašymo aplinkoje ši sintaksė žymima kaip kodas (žr. 12 pav.). Greičiausiai šioje vietoje nėra tikrinama, ar prieš *else* buvo *if* teiginys ir ar *else* iš viso yra *Razor* kodo bloke (po „@“ žymos).

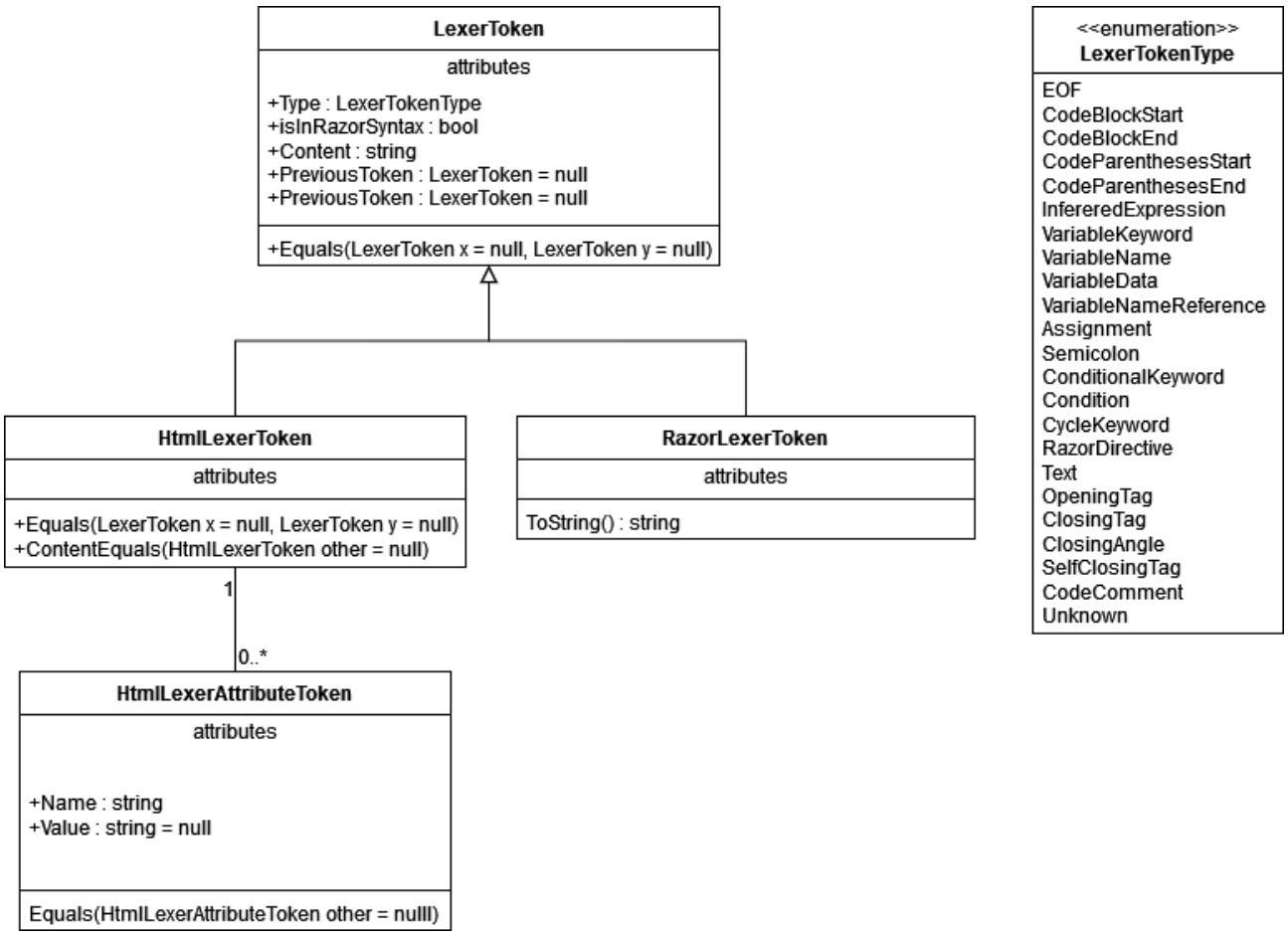


12 pav. Neteisingas sintaksės žymėjimas *.cshtml* faile naudojant *Visual Studio Code* kodo redaktorių

3.1.3. Specializuoto lekserio struktūra

Visas lekserio ir jo korektiškam veikimui užtikrinti skirtas vienetinių testų kodas pateiktas prieduose (žr. 1 Priedas ir 2 Priedas). Šis lekseris pilnai veikia su HTML standarto failais, o *Razor* ir C# sintaksės palaiko tiek, kiek reikia tolimesniam palyginimui vykdyti. Visa lekserio logika – atpažinti HTML sintaksę bei tam tikrus *C#* ir *Razor* sintaksės simbolius, kurie bus naudingi palyginimo algoritmui.

Lekserio klasės yra dvi – *HtmlLexerToken* ir *RazorLexerToken*. Šios dvi klasės yra paveldimos iš bendrinės klasės – *LexerToken* (žr. 13 pav.).



13 pav. Lekserio naudojamų žetonų klasių diagrama

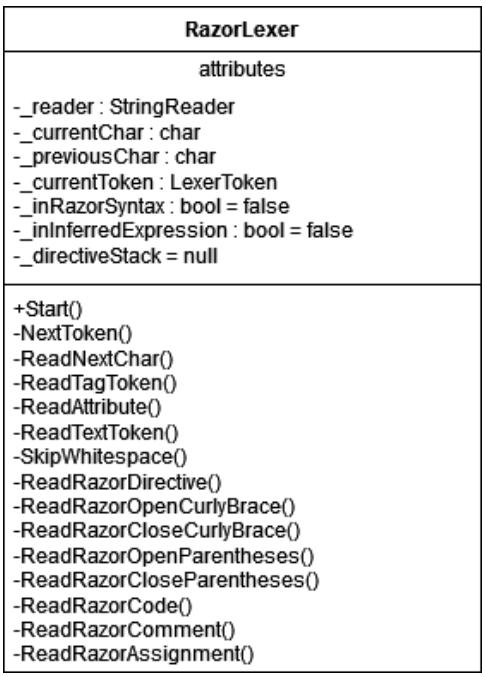
Lekseris turi aštuonis kintamuosius (žr. 14 pav.):

1. Žetonų masyvas: *ICollection<ILexerToken> Tokens*. Šis masyvas laiko ir prie jo yra pridedami analizuojamo failo žetonai;
2. Esamas simbolis: *char _currentChar*. Simbolis, kuris šiuo metu bus analizuojamas algoritme;
3. Praeitas simbolis: *char _previousChar*. Simbolis, kuris buvo analizuojamas prieš tai – buvusi *_currentChar* reikšmė;
4. Esamas žetonas: *ILexerToken _currentToken*. Žetonas, kuris buvo nustatytas leksinės analizės metu ir bus priskirtas prie žetonų masyvo;
5. Žetono, esančio *Razor* sintaksės viduje, režimo žyma: *bool _inRazorSyntax*. Kai *Razor* faile prieinama prie simbolio „@“, tuomet ši reikšmė pasikeičia į *true*. Pasibaigus šiam kodo blokui – nustatoma į *false*;
6. Žetono, esančio *Razor* sintaksės įterptoje išraiškoje, režimo žyma: *bool _inferredExpression*. Panašiai, kaip ir *Razor* sintaksės režime, čia žyma pasikeičia kai skaitomas simbolis yra viduje įterptos išraiškos „@(…)“ sintaksės;
7. Žetono, esančio kodo vykdymo režime, žyma: *bool _inCodeEvaluation*. Papildomas režimas *Razor* sintaksės režimui, žymintis tekstą, esantį viduje *Razor* sintaksės, kaip kodą;

8. HTML žymų ir kodo blokų dėklas: *Stack<ILexerToken> _directiveStack*. Ši duomenų struktūra saugo HTML žymų ir kodo blokų gylį.

Visas lekseris vykdomas per jame esančią funkciją *Start*, kurioje begalinis ciklas *while(true)* vykdo kito žetono nuskaitymą. Lekseryje esančios funkcijos yra tokios (žr. 14 pav.):

1. Start – pradinė funkcija (žr. 15 pav.);
2. NextToken – Start funkcijoje kviečiama funkcija, kuri pradeda esamo simbolio analizę ir kviečia kitas funkcijas žetonui suformuoti (žr. 16 pav.);
3. ReadNextChar – funkcija, nuskaitanti kitą simbolį;
4. ReadTagToken – funkcija, nuskaitanti HTML atidaromąją žymą;
5. ReadAttribute – funkcija, nuskaitanti HTML atidaromojoje žymoje esančius atributus;
6. ReadTextToken – funkcija, kuri kviečiama visais kitais atvejais NexToken esančioje logikoje. Iš čia nustatoma, ar sekantys simboliai yra HTML tekstas, ar *Razor* arba C# kodas (žr. 17 pav.);
7. SkipWhitespace – funkcija, skirta praleisti tuštumos simbolius, tokius kaip tarpai, tabuliacijos, naujos eilutės ir pan.;
8. ReadRazorDirective – funkcija, kuri nuskaito *Razor* žymą „@“ ir nustato žetoną, jeigu nevykdomos išimtys;
9. ReadRazorOpenCurlyBrace – funckija, nuskaitanti *Razor* atidaromąjį kodo simbolį „{“;
10. ReadRazorCloseCurlyBrace – funkcija, nuskaitanti *Razor* uždaromąjį kodo simbolį “}”;
11. ReadRazorOpenParentheses – funckija, nuskaitanti *Razor* atidaromąjį skliaustų simbolį „(“;
12. ReadRazorCloseParentheses – funkcija, nuskaitanti *Razor* uždaromąjį skliaustų simbolį “)”;
13. ReadRazorCode – funkcija, nuskaitanti *Razor* kodo režime esantį C# kodą (žr. 18 pav.). Čia nusprendžiama ar kodo žetonas yra priskiriamas pagal enumeratoriuje esančias reikšmes: priskyrimo (Assignment), kintamojo duomenys (VariableData), kabliataškis (Semicolon), kintamojo pavadinimas (VariableName), sąlyga (Condition), kintamojo kvietimas (VariableNameReference), kintamojo raktažodis (VariableKeyword), sąlygos raktažodis (ConditionalKeyword), ciklo raktažodis (CycleKeyword), arba nežinomas (Unknown).



14 pav. Specializuoto lekserio klasė

3.1.4. Specializuoto lekserio veikimas

Lekseris atskiria HTML ir *Razor* kodo dalis. Pavyzdžiui, jeigu *.cshtml* faile pasitaiko *Razor* kodo žyma „@“ – tai reiškia, kad greičiausiai prasideda kodas ir lekserio režimas pasikeičia į kodo atpažinimo režimą. Pavyzdžiui, turint tokią kodo eilutę su lekseriu kodo režime:

```
for (int i = 0; i < 5; i++)
```

atpažinami žetonai bus keturi (čia pavyzdyje lekserio kodo režimas būtų įjungtas):

1. *for* ciklo raktažodis;
2. skliaustų pradžia;
3. visa sąlyga, esant viduje skliaustų;
4. skliaustų pabaiga.

Tikrinimo metode nebus galima pasinaudoti išraiškomis (sąlygomis), esančiomis funkcijų ir kitų kodo srautų argumentuose, todėl ši dalis sudedama į vieną žetoną, kadangi: 1) tai yra kodas, o ne HTML; 2) tikrinimo metu nebus panaudojamos sąlygos; 3) net ir turint išskaidytą išraišką, negalima būtų panaudoti, nes, tikėtina, kad cikle bus kintamosios reikšmės iš modelio. Šie argumentai galioja visų kodo kontrolės srauto (angl. *control flow*) vietose. Tačiau čia naudinga turėti ciklo raktažodį, kadangi pagal šį raktažodį palyginimo logikoje galima panaudoti ciklo kontrolės srauto žetoną, kaip nulio arba daug HTML struktūros ir turinio, esančio cikle, patikrinimą. Panašiai ir loginio *if* patikrinimo atveju – priskiriamas sąlygos žetonas, pagal kurį galima tikrinti kodo kontrolės srauto galimybes.

Galioja vienintelė išimtis žetonams – HTML atributai, kurie yra nuskaitomi ir priskiriami atidaromo atributo klasės objektui. Tokia logika padaro šį specializuotą lekserį ne grynu lekseriu, nes kartu turi

šioje vietoje ir sintaksės analizės elementų – kuriama sintaksinio medžio struktūra HTML žymos atributams. Pavyzdžiui, turint tokią HTML sintaksę:

```
<a class=”link” href=”https://ktu.edu“>Nuoroda</a>
```

žetonai bus penki, o pirmasis žetonas turės struktūrą su dvejais atributais:

1. Atidaromoji žyma „a“;
    a. Atributo raktažodis „class“ su reikšme „link“;
    b. Atributo raktažodis „href“ su reikšme „https://ktu.edu“;
2. Uždaromasis simbolis „>“;
3. Tekstas „Nuoroda“;
4. Uždaromoji žyma „a“;
5. Uždaromasis simbolis „>“.

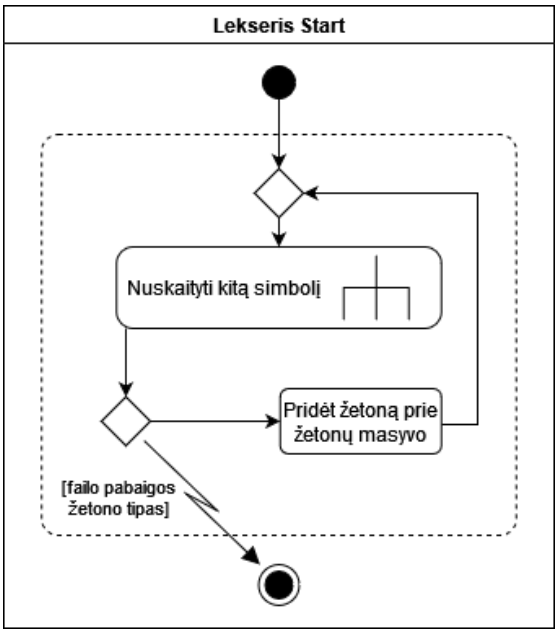
Kadangi lekseryje naudojamos žetonų klasės yra dvi: 1. HTML ir 2. *Razor* arba C# struktūros, pagal šias struktūras galima atpažinti kokio tipo yra žetonas. Tai palengvina pačio lekserio analizę ir jas panaudoti galima tikrinimo algoritme. Kiekvienas žetonas taip pat turi loginę (angl. *boolean*) reikšmę, kurios pagalba galima sekti kokiame režime yra lekserio analizuojama reikšmė – algoritmo logika atsižvelgia, ar simbolis arba žetonas yra viduje *Razor* sintaksės, ar išorėje. Ši reikšmė yra *tiesa* ir HTML sintaksėje, kai ji yra viduje *Razor* kodo bloko. Ir visada *tiesa,* kai *Razor* tipo žetonas yra *Razor* sintaksėje. Tikrinimo dalyje tai yra naudinga atskyrimui ar, pavyzdžiui, HTML žyma yra kode ir ciklo ar loginės sąlygos patikrinimo kodo bloke.

3.1.5. Specializuoto lekserio algoritmas

Šiame specializuotame lekseryje norėta išvengti rekursijos ir daugiau nei vieno žetono priskyrimo pagrindinio ciklo vykdymo metu. Visas lekserio algoritmas vykdomas *Start* funkcijoje esančiame begaliniame *while* cikle (žr. 15 pav.), kuris iškviečia funkciją tolimesnei analizei pagal esamą simbolį ir lekserio režimą. Šioje esamo simbolio analizės funkcijoje tikrinami devyni atvejai *switch* sąlygoje (žr. 16 pav.):

1. „failo pabaigos“ ar „srauto pabaigos“ reikšmė, kai pasibaigia simboliai duomenų sraute. Ši reikšmė gali būti *char* duomenų tipo reikšmės „\0“ arba „\uffff“;
2. HTML žymos atidaromasis simbolis „<“;
3. HTML žymos uždaromasis simbolis „>“;
4. *Razor* kodo žyma „@“;
5. Kodo bloko pradžia „{“;
6. Kodo bloko pabaiga „}“;
7. Skliaustų išraiškos pradžia „(“;
8. Skliaustų išraiškos pabaiga „)“;

9. Bet kokia kita reikšmė (tolimesnė analizė arba tekstas).



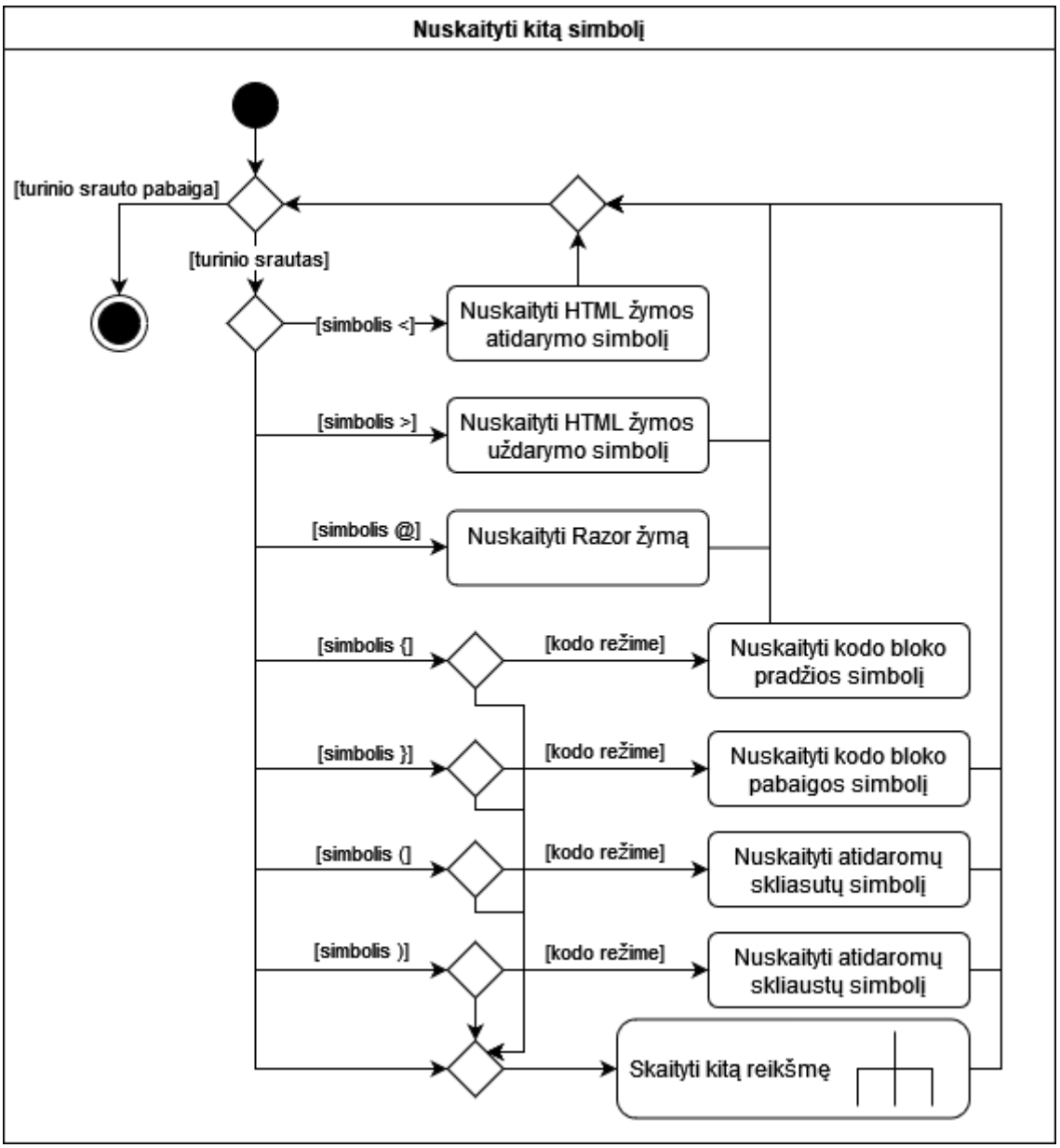
15 pav. Specializuoto lekserio algoritmo pagrindinio begalinio ciklo veiklos modelis

Vykdant kito simbolio nuskaitymo funkciją (žr. 16 pav.) ir jei pirmoji sąlyga atitinka – reiškia buvo pasiektas failo duomenų srauto galas ir daugiau nebėra ko nuskaityti. Vykdant šią sąlygos šaką sukuriamas žetonas su failo pabaigos (EOF, angl. *end-of-file*) tipu. Grįžus atgal į *Start* funkciją, tikrinama, ar esamas žetonas yra failo pabaigos tipo. Šiuo atveju tai yra *tiesa* ir begalinis ciklas nutraukiamas – lekserio algoritmas baigė darbą.

Jei žyma yra atidaromasis simbolis – vykdoma atidaromosios žymos nuskaitymo funkcija. Šioje funkcijoje turi būti atsižvelgta į šiuos atvejus: 1) žymos pavadinimas gali būti bet koks; 2) žymos viduje gali pasitaikyti tuščių ženklų; 3) žyma gali turėti atributų; 4) žyma gali būti save uždaranti. Jeigu HTML struktūra korektiška, simboliai skaitomi iki HTML uždaromosios žymos, esamas žetonas nustatomas į atitinkamą ir funkcija baigiasi su nustatytu esamu simboliu „>“ tolimesnei pagrindinio ciklo iteracijai apdoroti. Jei žyma nėra save uždaranti ir žetonas yra HTML žymos atidaromasis – į dėklą įdedama ši žetono reikšmė. Jei HTML žyma yra uždaromoji, tuomet iš dėklo yra panaikinama paskutinė reikšmė, kuri korektiškame HTML faile bus ta pati, kaip ir atidaromoji.

Jeigu simbolis yra HTML žymos uždarymas, esamas žetonas nustatomas į HTML uždaromąjį tipą ir nuskaitomas kitas simbolis.

Jeigu simbolis yra *Razor* žyma „@“, tuomet kviečiama Razor žymos apdorojimo funkcija. Šioje funkcijoje tikrinamos trys sąlygos. Pirmoji sąlyga – jeigu po šios žymos iš karto seka dar viena tokia pati žyma, vadinasi tai yra sintaksės „pabėgimo“ (angl. *escape*) simbolis, kuris turi būti traktuojama kaip dalis paprasto teksto (du ženklai „@@“ reiškia vieną „@“ teksto simbolį) ir toliau yra skaitomas tekstas. Antroji sąlyga – jeigu seka žvaigždutės simbolis „*“, kuris žymi *Razor* komentaro pradžią, toliau skaitomi simboliai iki komentaro galo simbolių „*@“. Jeigu kitas simbolis nepatenka į šias išimtis – sukuriamas *Razor* žymos tipo žetonas ir *Razor* lekserio kodo režimas nustatomas į reikšmę *true*.



16 pav. Specializuoto lekserio algoritmo pagrindinio simbolių atskyrimo veiklos diagrama

Jeigu simbolis yra kodo bloko atidarymas „{“ ir šiuo metu lekserio kodo režimas yra įjungtas, vykdoma *Razor* kodo pradžios simbolio nuskaitymo funkcija. Kitu atveju simbolis skaitomas kaip paprastas tekstas. Funkcija nustato esamą žetoną į kodo bloko pradžios tipą, į kodo dėklą įdedamas naujai priskirtas esamas žetonas ir kodo vykdymo režimas nustatomas į *true*.

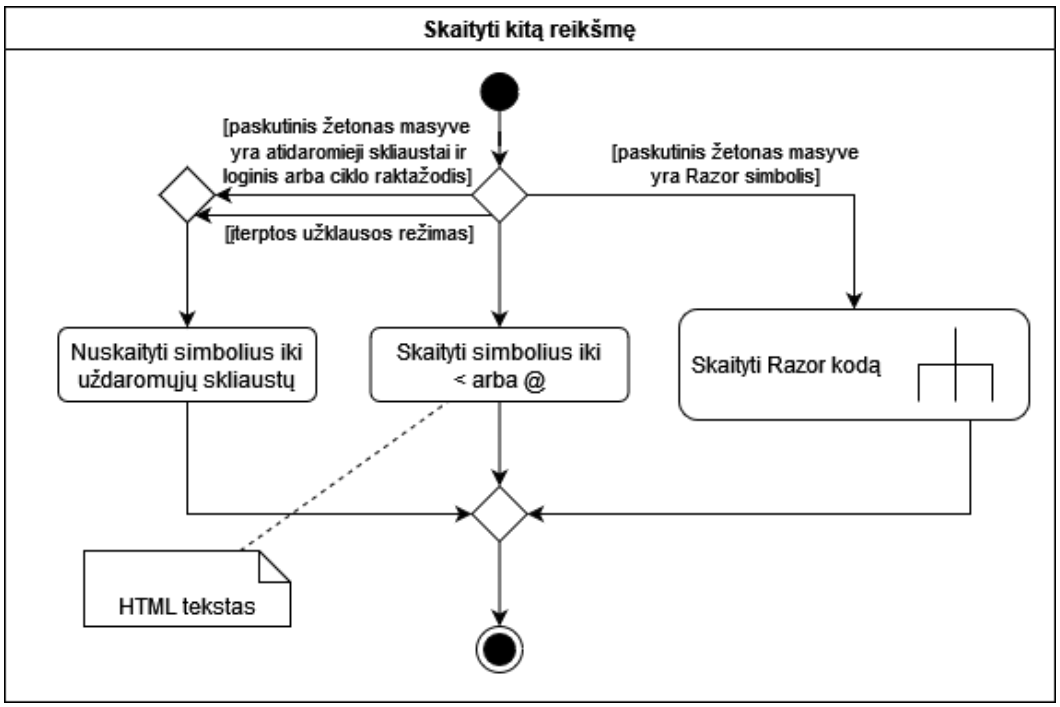
Jeigu simbolis yra kodo bloko uždarymas „}“ ir šiuo metu lekserio kodo režimas yra įjungtas, vykdoma *Razor* kodo pabaigos simbolio nuskaitymo funkcija. Kitu atveju, taip pat kaip ir su kodo atidarymo simboliu, nukreipiama į paprasto teksto apdorojimą. Šioje kodo bloko uždarymo funkcijoje žetonas priskiriamas į kodo uždarymo tipą. Dėkle esanti reikšmė yra panaikinama. Jeigu daugiau atidarytų kodo blokų nėra – lekserio kodo režimas ir kodo vykdymo režimas nustatomi į *false*.

Jeigu simbolis yra atsidarantys skliaustai „(“ ir šiuo metu lekserio kodo režimas yra įjungtas, vykdoma *Razor* skliaustų atsidarymo simbolio nuskaitymo funkcija. Šioje funkcijoje esamas žetonas

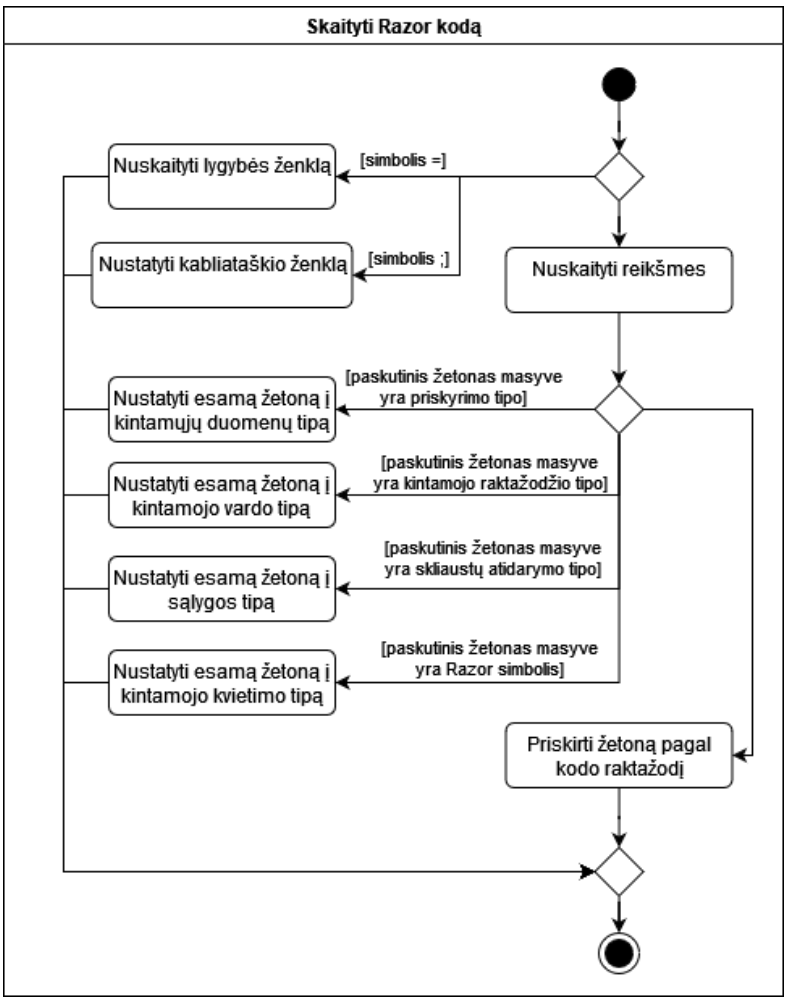
nustatomas į skliaustų atsidarymo tipą, į kodo dėklą įkeliamas šis naujai priskirtas žetonas ir, jeigu prieš tai buvęs žetonas buvo *Razor* žyma „@“ – nustatomas *Razor* įterptos funkcijos režimas į *true*.

Jeigu simbolis yra užsidarantys skliaustai „)“ ir šiuo metu lekserio kodo režimas yra įjungtas, vykdoma *Razor* skliaustų uždarymo simbolio nuskaitymo funkcija. Šioje funkcijoje esamas žetonas nustatomas į skliaustų uždarymo tipą, o iš kodo dėklo ištrinamas paskutinis žetonas, kuris korektiškai parašytame faile turėtų būti skliaustų atsidarymo žetonas. Jeigu kodas buvo įterptos funkcijos režime, šis režimas, kartu su *Razor* kodo režimu yra pakeičiamas į *false*.

Paskutinis atvejis, kai tikrinama bet kokia kita reikšmė, išsiplečia į visą likusį HTML ir *Razor* kodo valdymą. Šioje vietoje skaitomos reikšmės pagal prieš tai buvusius žetonus. Jei vienas jų atitinka kodo logiką – skaitomas Razor kodas, numatytuoju atveju reikšmės skaitomos kaip paprastas tekstas (žr. 17 pav. ir 18 pav.).



17 pav. Kitos reikšmės skaitymo detali veiklos diagrama



18 pav. Razor kodo skaitymo detali veiklos diagrama

3.1.6. Specializuoto lekserio pavyzdžiai

Lekseris kuriamas su jo panaudojimo tikrinimo etape tikslu. Dėl to tiek HTML, tiek *Razor* kode esanti HTML sintaksė turi būti skaitoma vienodai. Šį funkcionalumą geriausiai užtikrina vienetiniai testai (žr. 2 Priedas). Norint geriau suprasti kaip turi atrodyti galutinis leksinės analizės rezultatas, galima išanalizuoti kelis pavyzdžius.

Pirmasis pavyzdys – kodo dalyje nėra HTML sintaksės, o HTML sintaksėje nėra *Razor* kodo.

```
1. @{
2.     ViewData["Title"] = "Privacy Policy";
3. }
4. <p>
5.     Puslapio paragrafas su  skirtingais   tarpų      tipais
6. </p>
```

Čia Razor lekserio elementai: 1) *Razor* žyma; 2) kodo bloko pradžia; 3) nežinomo tipo žetonas „ViewData["Title"]“; 4) priskyrimo (lygybės) simbolis; 5) kintamojo duomenys; 6) kabliataškis; 7) kodo bloko pabaiga; 8) HTML atidaromoji žyma „p“ 9) uždaromasis simbolis „>“; 10) Tekstas „Puslapio paragrafas su skirtingais tarpų tipais“; 11) HTML uždaromoji žyma „p“; 12) uždaromasis simbolis „>“. Pagal HTML standartą – pertekliniai tarpai bet kurioje vietoje turi būti ignoruojami [30].

Antras pavyzdys – HTML turinyje esantys Razor kintamieji.

```
1. <p>Šiuo metu tekstą skaito @lankytojuSk žmonės. Iš viso lankytojų buvo @visoLankytojuSk</p>
```

Leksinės analizės metu reikia atskirti paprastą HTML tekstą nuo įterptų kintamųjų duomenų. Šiame pavyzdyje žetonai bus tokie: 1) atidaromoji HTML žyma „p“; 2) Uždaromasis HTML simbolis „>“; 3) Tekstas „Šiuo metu tekstą skaito “; 4) *Razor* žyma „@“; 5) Kintamasis pavadinimu „lankytojuSk“; 6) Tekstas „ žmonės. Iš viso lankytojų buvo “; 7) *Razor* žyma „@“; 8) Kintamasis pavadinimu „visoLankytojuSk“; 9) Uždaromoji HTML žyma „p“; 10) Uždaromasis HTML simbolis „>“. Čia reiktų atkreipti papildomą dėmesį į teksto žetonus, kuriuose gali būti tarpai pradžioje ir gale.

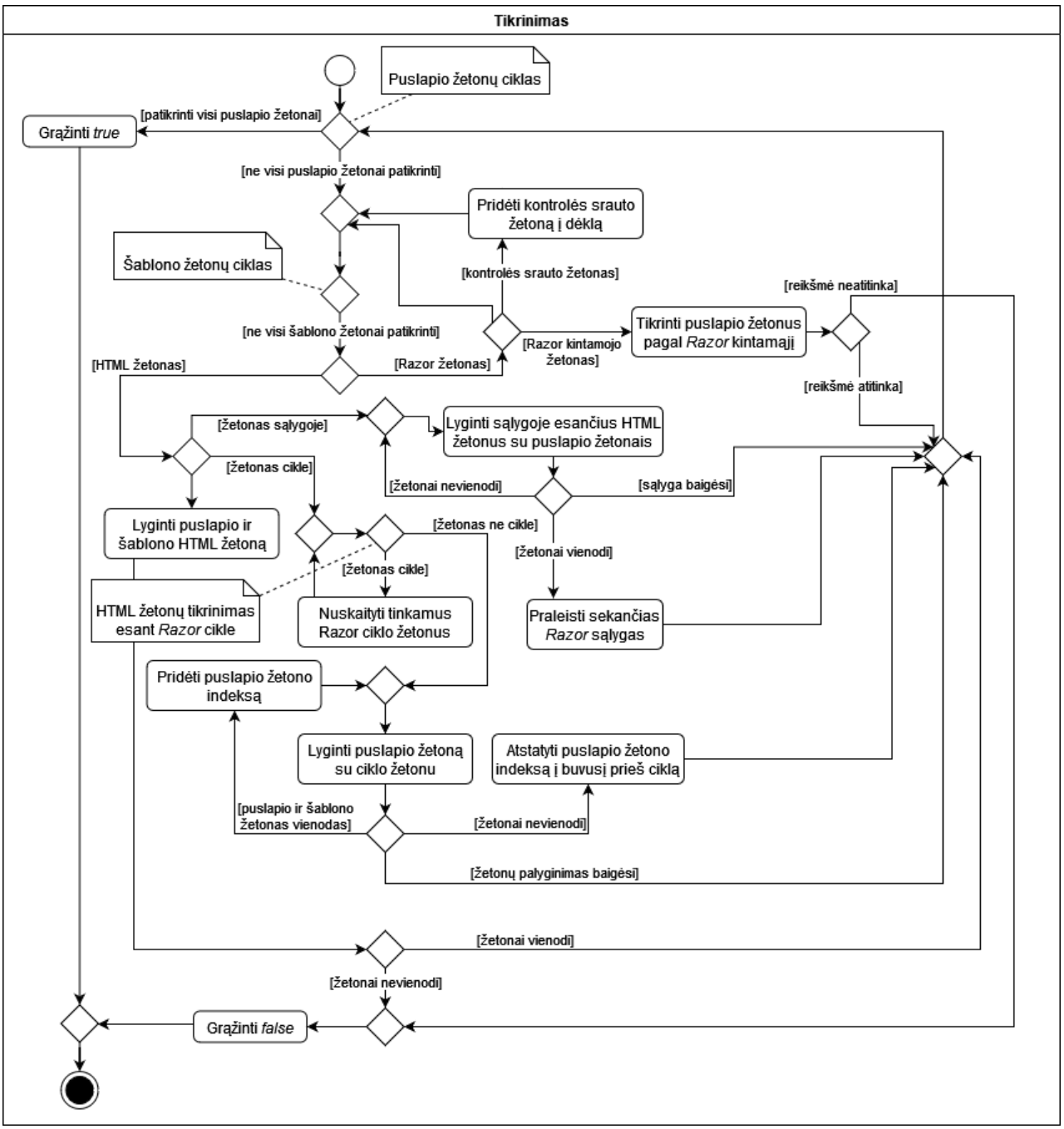
Trečiasis pavyzdys – loginė funkcija su HTML žymomis.

```
1. @{
2.     if (true)
3.     {
4.         <a href="https://ktu.edu">KTU</a>
5.     }
6.     else
7.     {
8.         <span>Nuorodos nėra</span>
9.     }
10. }
```

Sąlygos logika privalo būti viduje Razor sintaksės kodo bloko. Šiuo atveju žetonai turi būti tokie: 1) *Razor* žyma „@“; 2) kodo bloko pradžia; 3) sąlyginės logikos raktažodis *if*; 4) atidaromieji skliaustai; 5) sąlyga; 6) uždaromieji skliaustai; 7) atidaromieji kodo bloko skliaustai; 8) atidaromoji HTML žyma „a“ su parametro raktu „href“ ir verte „"https://ktu.edu"“; 9) tekstas „KTU“; 10) uždaromoji HTML žyma „a“; 11) uždaromasis HTML simbolis „>“; 11) uždaromieji kodo bloko skliaustai; 12) sąlyginės logikos raktažodis *else*; 13) atidaromieji kodo bloko skliaustai; 14) atidaromoji HTML žyma „span“; 15) uždaromasis HTML simbolis „>“; 16) Tekstas „Nuorodos nėra“; 16) uždaromoji HTML žyma „span“; 17) uždaromasis HTML simbolis „>“; 18) uždaromieji kodo bloko skliaustai; 19) uždaromieji kodo bloko skliaustai.

3.2. Šablono tikrinimas

Tikrinimas vyksta pagal lekserio analizės rezultato metu sukurtus žetonus. Šiame tikrinimo algoritme yra du pagrindiniai ciklai: pirmasis – atsiųsto puslapio, antrasis – šablono puslapio lekserio žetonų masyvams. Puslapio žetonų masyvas rodo į gauto puslapio žetono indeksą, pagal kurį antrame masyve esantis šablono žetonų masyvo indeksas tikrina esamą pirmojo ciklo indekso masyve reikšmę (žr. 19 pav.).



19 pav. Siūlomo žiniatinklio pakeitimo aptikimo tikrinimo algoritmo proceso diagrama

3.2.1. Tekstinio turinio tikrinimas

Teksto lyginimo metodas turi atsižvelgti ne tik pačio teksto atitikimą, bet kartu ir į įterptines kintamąsias reikšmes (žr. 20 pav.). Pagrindinės teksto, esančio HTML žymoje, kuriame yra kintamosios reikšmės, problemos yra keturios:

1. HTML žymoje yra tik kintamojo reikšmė;

```
<p>@i</p>
```

2. teksto turinys prasideda kintamuoju;

```
<p>@pasisveikinimas, Lukai!</p>
```

3. vienas ar keli kintamieji yra viduryje esančio teksto;

```
<p>Iš viso yra: @visoSk vienetų.</p>
```

4. kintamasis yra gale teksto.

```
<p>Iš viso: @suma</p>
```

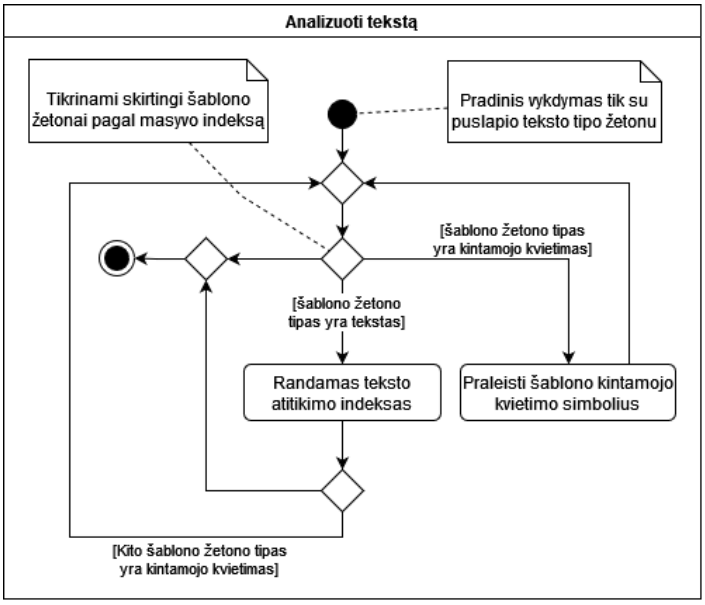
Po HTML užsidarymo atitikimo patikrinimo seka teksto tikrinimas. Pirma reikia patikrinti, ar reikšmė yra *Razor* tipo. Pirmuoju ir antruoju atveju pradinė HTML teksto srityje esanti reikšmė yra kintamasis. Kadangi pirmuoju atveju jokio teksto daugiau nėra, tai reiškia, kad visas turinys bus praleidžiamas. O antruoju atveju imamas tekstas „, Lukai!“ ir ieškomas šio teksto atitikimo indeksas pasinaudojant C# esančia funkcija *IndexOf(string)* ir bus palyginami simboliai. Trečiuoju atveju reikšmė yra viduryje teksto, reiškia iki jos ir po jos esantis tekstas turi atitikti. Ketvirtuoju atveju visas tekstas iki kintamojo turės atitikti. Tokia pati logika yra taikoma įterptinei *Razor* išraiškai:

```
1. <p>Suma: @(1 + 1)</p>
```

Kurios rezultatas yra toks:

```
1. <p>Suma: 2</p>
```

Šioje vietoje tekstas „Suma: “ sutaps, o intervalas nuo šio teksto pabaigos iki HTML uždaromosios žymos yra ignoruojamas.



20 pav. Siūlomo žiniatinklio pakeitimo aptikimo teksto tikrinimo proceso diagrama

3.2.2. HTML tikrinimas šablono loginėje sąlygoje

Tikrinimo algoritmui aptikus sąlygos raktažodį šablono lekserio žetonų masyve, žetonai nuskaitomi iki pirmo pasitaikiusio HTML žetono. Toliau laikinai išsaugoma puslapio žetonų masyvo indekso reikšmė. HTML turinys tikrinamas. Jeigu rastas atitikimas, visa likusi susijusi loginė sąlyga toliau yra praleidžiama. Jei pirmoje sąlygoje puslapio turinys neatitiko šablono, toliau laikinas indeksas nustatomas kaip pirminis naudojamas puslapio žetonų ciklo indeksas ir tikrinamas sekantis susijęs loginis tikrinimas (jeigu toks yra). Jeigu sekantis loginės sąlygos raktažodis yra *else*, atitikimas privalo būti, kitaip tikrinimo algoritmas stabdomas ir grąžinama reikšmė *false* (žr. 7 pav.).

3.2.3. HTML tikrinimas šablono cikle

Ciklas gali įvykti nežinomą kiekį kartų arba iš vis neįvykti. Tokiu atveju šablono cikle esančios HTML žymos yra tikrinamos pagal puslapį tol, kol nebus rastas atitikimas (žr. 8 pav.). Tai reiškia, kad tik prasidėjęs tikrinimo ciklas iš karto gali pasibaigti, todėl tikėtina, jog puslapio generavimo metu šis ciklas tiesiog nebuvo vykdomas.

Šis įgyvendinimas apverčia puslapio ir šablono žetonų masyvų tikrinimo ciklų eiliškumą (žr. 19 pav. esantis komentaro objektas „HTML žetonų tikrinimas esant Razor cikle“).

3.2.4. Tikrinimo algoritmo pavyzdžiai

Paprastas HTML puslapio ir šablone esančių paprastų HTML elementų palyginimas vykdomas kaip statiniame puslapyje – vykdomas tiesioginis lekserio simbolių palyginimas. Tačiau prasidėjus *Razor* sintaksei, loginis tikrinimas vykdomas atsižvelgiant į esminius *Razor* elementus (žr. 4 Priedas). Jei tikrinimo algoritmas prieina prie *Razor* išraiškos su loginiu *if* ir *else* patikrinimu, šiame pavyzdyje visada įvykdant pirmąją sąlygą:

```
@{
    if (true)
    {
        <a href="https://ktu.edu">KTU</a>
    }
    else
    {
        <span>Nuorodos nėra</span>
    }
}
```

Puslapio sukompiliuotas rezultatas yra:

```
<a href="https://ktu.edu">KTU</a>
```

Tikrinimas bus sėkmingas, kadangi pateiktame šablono pavyzdžio ketvirtoje eilutėje yra pateiktas tas pats HTML turinys, kaip ir galutiniame HTML puslapyje. Jeigu interneto puslapyje būtų HTML turinys, esantis *else* sąlygos dalyje, tokiu atveju tikrinimas taip pat būtų sėkmingas. Tačiau taip pat jei tikrinamame puslapyje nebūtų reikšmės, esančius loginės išraiškos kodo bloke, tai taip pat būtų teisingas variantas, kadangi sąlyga tiesiog galėjo būti nepriimta HTML generavimo metu.

Jeigu prieinama prie ciklo:

```
@{
    for (int i = 0; i < 3; i++){
        <p>@i</p>
    }
}
```

Sukompiliuotas puslapio HTML rezultatas atrodo taip:

```
<p>0</p>
<p>1</p>
<p>2</p>
```

Tikrinimo algoritmas turi atrasti, kad visi trys HTML paragrafo („<p>“) elementai yra generuoti *for* ciklo ir rezultatas yra teigiamas – puslapis atitinka šabloną.

Turint teksto turinį su įterptomis kintamosiomis reikšmėmis, tikrinimo metu atitikti privalo tik statinis tekstas:

```
<p>@(DateTime.Now.Hour > 19 ? "Labas vakaras" : "Laba diena"), sistemos naudotojau!</p>
```

Visa *Razor* įterptinės logikos reikšmė yra ignoruojama ir tikrinama, ar tekstas, prasidedantis po jos, yra vienodas. Vienas iš šios išraiškos rezultatų gali būti toks:

```
1. <p>Laba diena, sistemos naudotojau!</p>
```

o teksto atitiktis bus tikrinama tik pagal reikšmes „„ sistemos naudotojau!“, kurios yra galas žymimas pagal uždaromąjį HTML žymos simbolį. Čia galima būtų toliau tobulinti prototipą analizuojant išraišką ir jos grąžinamas reikšmes.

Šiame modelio prototipe nėra įgyvendintas daugiau nei vieno gylio kodo kontrolės srauto tikrinimas. Tai reiškia, kad loginė sąlyga, esanti viduje kitos sąlygos, cikle esančios loginės sąlygos ir t.t. nėra palaikomos, tačiau, teoriškai, tokia struktūra gali būti palaikoma, jei šis žiniatinklio pakeitimo aptikimo modelio prototipas būtų tobulinamas toliau.

3.2.5. Šablonų tikrinimo programos panaudojimas

Šio įgyvendinto metodo kvietimui reikia nurodyti tris parametrus:

1. tikrinamo puslapio nuorodą (*url*).
2. šablono failą (*template*);
3. šablono išdėstymo failą (nebūtinas, *layout*);

Parametrai turi būti nurodomi programai taip:

```
TemplateDiff.exe --url "https://lukasa.lt" --template "./failas.cshtml" --layout "./failas.cshtml"
```

Vykdymo metu rodomi atitikimo momentai (sutapę elementai). Pasibaigus algoritmo vykdymui pateikiamas rezultatas, nurodantis ar interneto puslapis atitinka, ar neatitinka nurodytam šablonui (žr. 21 pav.).

21 pav. Įgyvendinto siūlomo žiniatinklio pakeitimo aptikimo prototipo veikimo rezultatas

4. Žiniatinklio pakeitimo aptikimo metodo prototipo rezultatai

Prototipas realizuotas kaip .NET konsolės programa, kuri yra sudaryta iš dviejų pagrindinių dalių: specializuotas lekseris ir tikrinimo algoritmas. Tai yra viena nuo kitos priklausomos dalys, todėl jų abiejų veikimo greitaveika turi būti įvertinta. Tikrinimo algoritmui turi būti nustatomas pakeitimo aptikimo tikslumas.

4.1. Žiniatinklio pakeitimo aptikimo metodo tikslumas

Pakeitimo aptikimo tikslumui įvertinti naudojamos dvi reikšmės: teisingas aptikimas (angl. *true positive*, TP) ir teisingas netinkamo pakeitimo aptikimas (angl. *true negative*, TN). Teisingo aptikimo reikšmė yra teisingai nustatytas nepakeistas turinys (šablonas ir puslapis atitinka vienas kitą). Teisinga netinkamo pakeitimo aptikimo reikšmė apibūdina tikrinimo metu teisingai aptiktą nesankcionuotą pakeitimą puslapyje (puslapis pakeistas ir šablonas nebeatitinka HTML).

Teisingas aptikimas yra procentinė teisingų atpažinimų reikšmė, apskaičiuojama:

$$TP = \frac{n_t}{n} \times 100\ \% \tag{1}$$

Čia n_t yra teigiamų identifikacijų skaičius, o n – iš viso atliktų tikrinimų skaičius. Šių skaičių dalybos rezultatas yra santykinė reikšmė tarp 0 ir 1, kuri yra dauginama iš 100 proc. ir gaunama teisingo aptikimo tikslumo reikšmė procentais.

Atvirkščiai skaičiuojamas teisingas netinkamo pakeitimo aptikimas:

$$TN = \frac{n_{nt}}{n} \times 100\ \% \tag{2}$$

Čia n_{nt} yra teisingų netinkamo pakeitimo aptikimų skaičius.

Bendras tikslumas T gaunamas iš n_t ir n_{nt} reikšmių vidurkio:

$$T = \frac{n_t + n_{nt}}{2} \tag{3}$$

4.2. Sintetinio duomenų generavimo tikrinimas

Patikrinti realų veikimą yra sudėtinga, kadangi metodas reikalauja šablono failo, kuris pasiekiamas tik sistemos vystytojams. Tai gali būti skaitoma, kaip jautrios prieigos ar komercinės paslapties apsaugoti failai. Taip pat reiktų turėti didelį kiekį įvairių veikiančių puslapių ir jų šablonų. Todėl pasirinkta tyrimą atlikti naudojant sintetinį turinio generavimą ir jį patikrinti.

4.2.1. Sintetinių duomenų generavimas

Sintetinių duomenų generavimui sukurta atskira programa, kuri pagal *C#* esančią *Random* klasę generuoja atsitiktinį kiekį HTML ir *Razor* reikšmių (žr. 6 Priedas). Kiekvienas atskiras reikšmių generatorius turi ciklą, kuris gali pridėti arba nepridėti reikšmių prie pagrindinio turinio.

Generavimas ir tikrinimas vykdomas dvejais etapais. Pirmasis etapas – teisingo aptikimo atvejo reikšmių generavimas, kurio metu tiek HTML, tiek *.cshtml* reikšmės turi atitikti. Pavyzdžiui, jeigu *Razor* sintaksėje įkeliamas ciklas, tai taip pat atitinkamai HTML sintaksė turės nustatytą kiekį žymų

ir turinio pagal *Razor* ciklą. Antrasis etapas - teisingo netinkamo pakeitimo aptikimo generavimas. Čia abiejų sintaksių turinys generuojamas taip pat, kaip ir pirmojo etapo metu, tačiau generavimui pasibaigus visas HTML turinys yra ištrinamas ir įkeliamas naujas, paprastas, neatitinkantis šablono HTML turinys. Taip gaunama nesankcionuoto turinio pakeitimo simuliacija.

4.2.2. Sintetinių duomenų tikrinimo rezultatas

Turinys sintetiškai sugeneruotas po 100 failų abejiems atvejams. Visi 100 failų teisingo aptikimo atveju buvo atitinkantys, reiškia iš 100 bandymų teisingai nustatyta 100 HTML puslapių atitinkančių šabloną. gaunamas teisingo netinkamo pakeitimo aptikimo rezultatas yra taip pat iš 100 bandymų, aptikta 100 šablono neatitikimų. Įvedus rezultatų reikšmes į formules gaunama prototipo tikslumo procentinė reikšmė yra:

$$TP = \frac{100}{100} \times 100 = 100\ \% \tag{4}$$

$$TN = \frac{100}{100} \times 100 = 100\ \% \tag{5}$$

$$T = \frac{100\ \% + 100\ \%}{2} = 100\ \% \tag{6}$$

Tai reiškia, kad šio tikrinimo algoritmo tikslumas T yra 100 proc. tikslus pagal ištirtus sintetinius duomenis.

4.3. Įgyvendinto prototipo greitaveika

Prototipas susideda iš dvejų dalių: lekserio ir tikrinimo. Tam, kad patikrinti prototipo greitaveiką, reikia įvertinti šių dalių greitaveiką. Naudojant *BenchmarkDotNet* įrankį nustatytas prototipo vykdymo greitis.

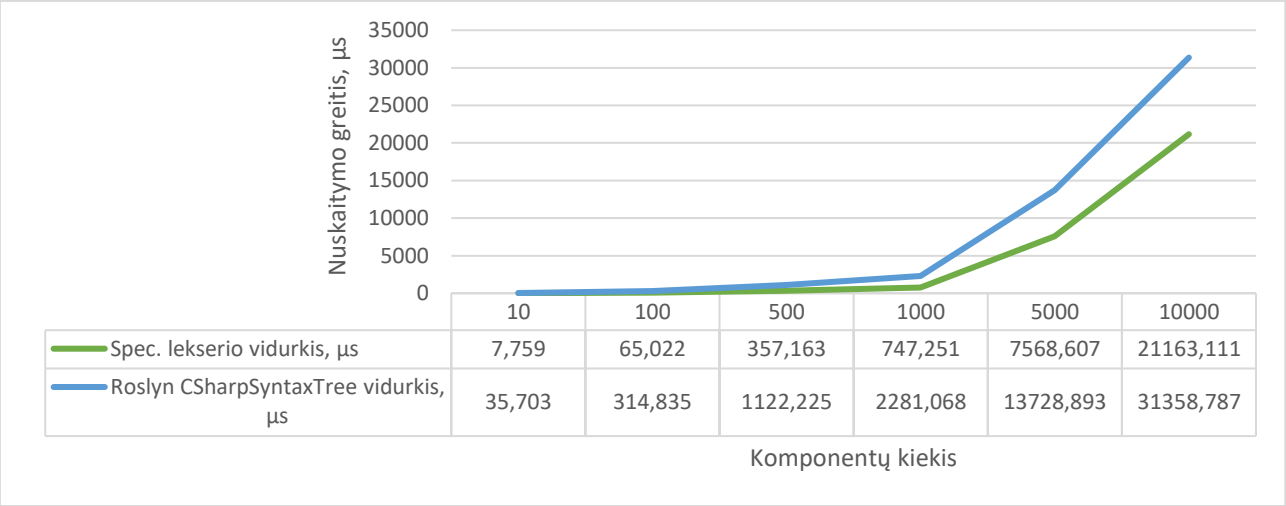
4.3.1. Specializuoto lekserio palyginimas su Roslyn

Atliktas greitaveikos palyginimo tyrimas tarp specializuoto lekserio ir *Roslyn CSharpSyntaxTree* funkcijos atskiriant HTML ir *.cshtml* failų sintaksę (žr. 22 pav. ir 23 pav.). Lyginant abiejų sprendimų HTML analizės greitį, pastebimas apie 9,8 karto spartesnis veikimas. O *.cshtml* failo analizės vykdymo laikas yra apie 5 kartus greitesnis. Sukurtas specializuotas lekseris yra spartesnis šio metodo vykdymo kontekste. Simbolių kiekiui didėjant santykinis skirtumas mažėja, tačiau vidutiniškai specializuoto lekserio greitaveika išlieka apie 3 kartus spartesnė (žr. 2 lentelė).

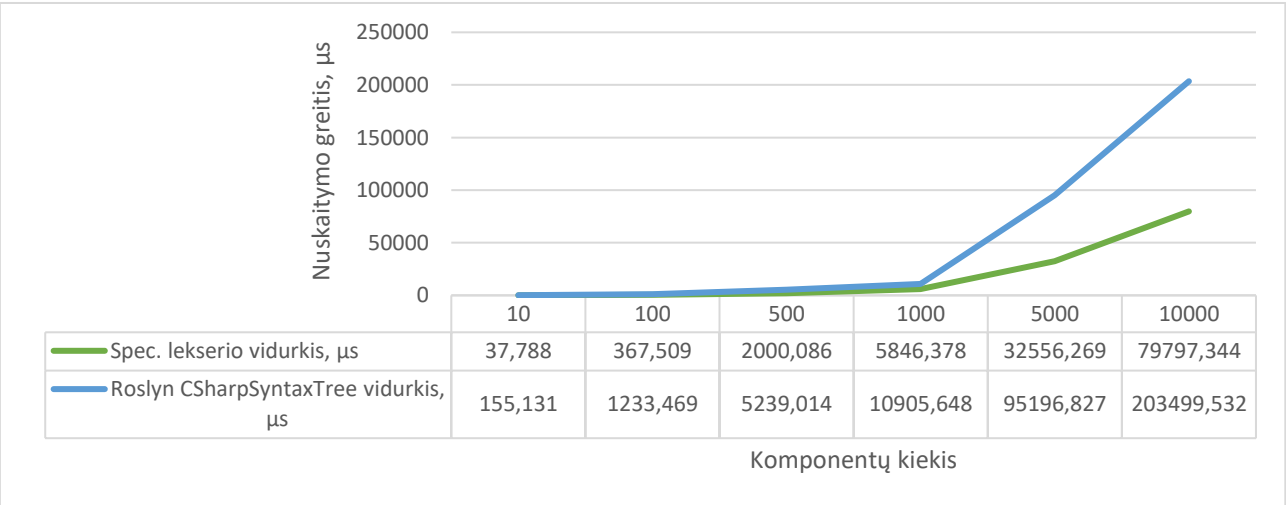
2 lentelė. Specializuoto lekserio ir *Roslyn CSharpSyntaxTree* greitaveikos palyginimo duomenys

Failo tipas	**Elementų kiekis**	**Specializuoto lekserio vykdymo greitaveika, µs**	***Roslyn CSharpSyntaxTree* vykdymo greitaveika, µs**	**Santykinė greitaveika, kartai**
HTML	10	7,759	35,703	4,6
	100	65,022	314,835	4,8
	500	357,163	1122,225	3,1
	1000	747,251	2281,068	3,1

	5000	7568,607	13728,893	1,8
	10000	21163,111	31358,787	1,9
CSHTML	10	37,788	155,131	4,1
	100	367,509	1233,469	3,4
	500	2000,086	5239,014	2,6
	1000	5846,378	10905,648	1,9
	5000	32556,269	95196,827	2,9
	10000	79797,344	203499,532	2,6
			Vidutinė reikšmė:	3



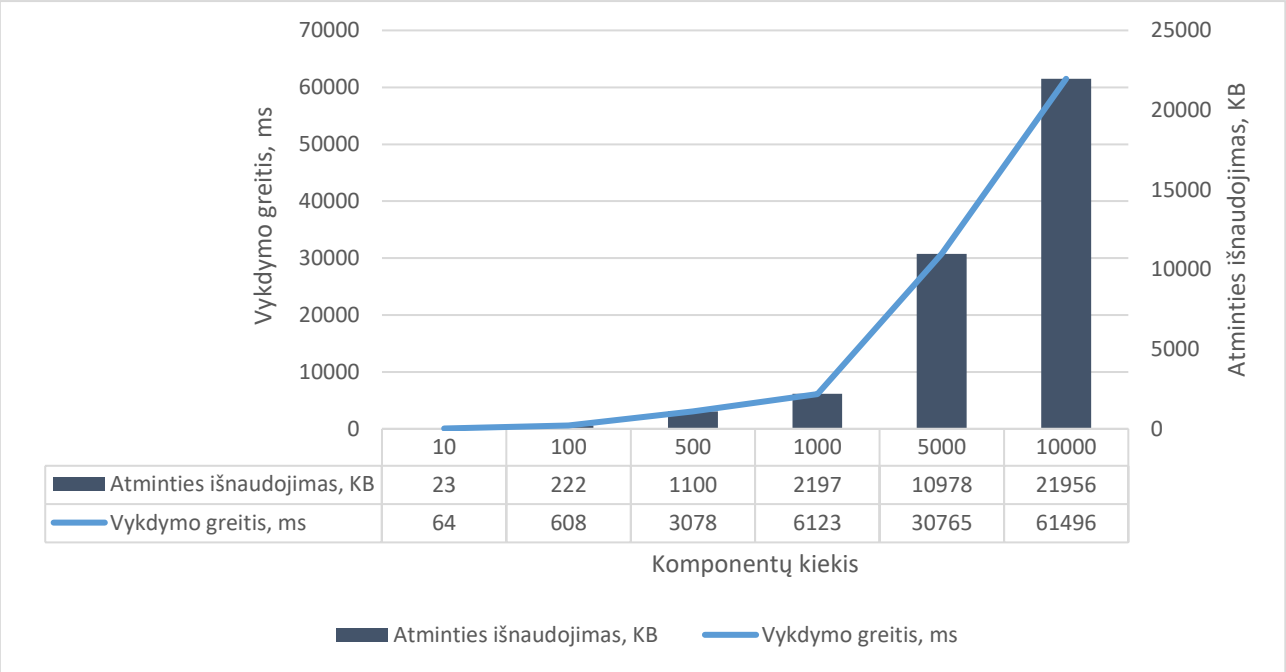
22 pav. Specializuoto lekserio ir *Roslyn CSharpSyntaxTree* greitaveikos palyginimas didėjant HTML simbolių kiekiui



23 pav. Specializuoto lekserio ir *Roslyn CSharpSyntaxTree* greitaveikos palyginimas didėjant *Razor* sintaksės simbolių kiekiui

4.3.2. Įgyvendinto tikrinimo algoritmo greitaveika

Tikrinimo algoritmo įgyvendintas prototipas kompiuterinę atmintį naudoja proporcingai pagal vykdymo greitį (žr. 24 pav.). Čia tyrimas atliktas ir su realybėje retai pasitaikančiu kiekiu komponentų. Realiuose projektuose šių komponentų kiekis dažniausiai svyruoja tarp 100 ir 1000. Naudojama atmintis, kai komponentų kiekis yra 10 000 siekia 22 MB, o vykdymo laikas – apie 1 min. Jei šis prototipas būtų naudojamas su modernia kompiuterine įranga, greitaveikos bei atminties problemos turėtų būti minimalios.



	10	100	500	1000	5000	10000
Atminties išnaudojimas, KB	23	222	1100	2197	10978	21956
Vykdymo greitis, ms	64	608	3078	6123	30765	61496

24 pav. Žiniatinklio pakeitimo aptikimo metodo prototipo greitaveika ir atminties išnaudojimas

4.4. Galimi metodo patobulinimai

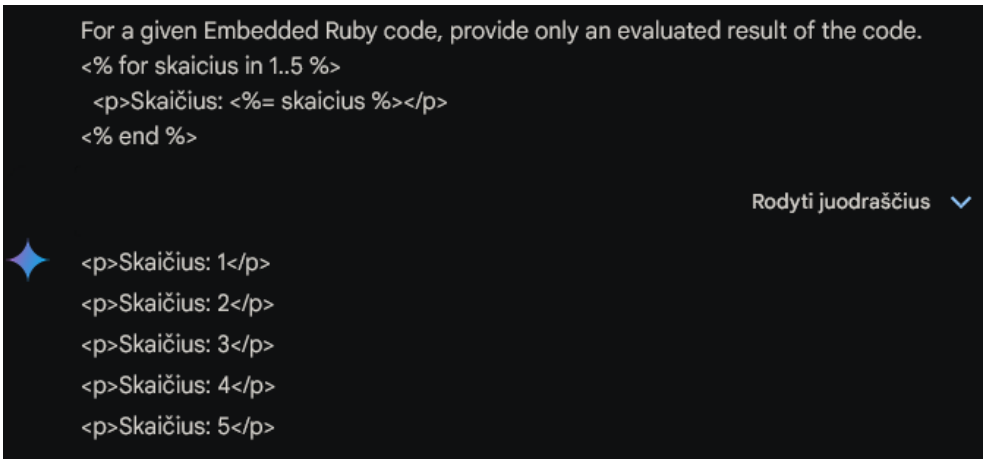
Remiantis atlikta esamų metodų analize, šį žiniatinklio turinio pakeitimo aptikimo metodą galima tobulinti panaudojant generatyvinio dirbtinio intelekto (angl. *generative artificial intelligence*), mašininio mokymo metodus ir šablonų failų integralumą.

4.4.1. Dalinis fiktyvus kompiliavimas dirbtinio intelekto pagalba

Vien tik lekseris negali suteikti pilno šablono atitikimo jo HTML rezultatui. Esami kompiliatoriai reikalauja klasių, metodų ir kitų vykdymo metu reikalingų dalių pilnam atitikimui. Tam galima būtų pasitelkti dirbtinį intelektą dinaminiam fiktyviam kompiliavimui. Pavyzdžiui, jei šablone yra ciklas, dirbtinio intelekto modelis galėtų nuspręsti kiek kartų su kokiomis reikšmėmis šis ciklas būtų vykdomas ir kokie ciklo rezultatai gautųsi. Taip galima būtų pagerinti metodo tikslumą, tačiau, tikėtina, jog tai kartu ir padarytų įtaką metodo greitaveikai ir tikslumui. Tokiu atveju būtų tikrinamos, o ne praleidžiamos kintamojo reikšmės. Pavyzdžiui, turint tokį *Embedded Ruby* programinį kodą:

```
1. <% for skaicius in 1..5 %>
2.   <p>Skaičius: <%= skaicius %></p>
3. <% end %>
```

galima pateikti užklausą dirbtinio intelekto programinei sąsajai ir gauti tikėtiną rezultatą (žr. 25 pav.). Taip, neturint programinės įrangos interpretatoriaus ar kompiliatoriaus, gaunama reikšmė, pagal kurią galima tikrinti ir kintamojo reikšmės rezultatą.



25 pav. *Google Gemini* platformoje pateikta užklausa ir jos rezultatas *Embedded Ruby* sintaksei

Tuo pačiu dirbtinio intelekto sprendimas turėtų potencialią saugumo spragą dėl įvesties įskiepijimo (angl. *prompt injection*) galimybės. Užkrėstas puslapis potencialiai galėtų turėti nuorodas dirbtiniam intelektui, kurios pakeistų originalią jo užduotį.

4.4.2. Atitikmens tikrinimas panaudojant mašininį mokymą

Sukurtas tikrinimo pagal šablonus metodas gali būti įgyvendintas pagal mašininio mokymo sprendimą, jeigu būtų apmokytas teisingais šablonais ir jų puslapių atitikmenimis. Tokiu atveju nereiktų įgyvendinti algoritmo kiekvienai šablonų sintaksei. Tačiau, dėl nedeterministinio mašininio mokymo metodo pobūdžio, tikslumas gali suprastėti.

4.4.3. *JavaScript* tikrinimas

JavaScript kodas HTML puslapyje nėra tikrinamas. Tai reiškia, jog į puslapį patekęs piktavalis kodas nebus atrastas, kadangi dinaminiai pokyčiai vyksta naršyklėje, o ne pradiniame sugeneruotame HTML faile. Tam galima būtų įgyvendinti *JavaScript* tikrinimo modulį. Šis modulis pirma taip pat tikrintų kodą šablonuose su gauta nuotoline kompiliuota puslapio versija.

4.4.4. Šablonų failų integralumo tikrinimas

Papildomai tikrinimo aplikacijos dalyje esančius šablonų failus galima sekti. Pavyzdžiui, apskaičiuoti santraukų reikšmes kiekvienam failui ir šią informaciją saugoti duomenų bazėje. Taip piktavaliui pakeitus failą ar nuorodą į jį diske bus aptinkamas neatitikimas.

Išvados

1. Atlikta analizė, kurios metu nustatyta, jog literatūroje nebuvo rastas metodas, naudojantis puslapio atitikimo tikrinimui pagal jo šabloną.
2. Detalizuotas šablonų žiniatinklio turinio pakeitimo pagal šablonus algoritmas.
3. Bendras pateiktas tikrinimo modelis turi vienodai tikrinti tiek HTML, tiek šablono failus, todėl atitikmens analizei realizuotas specializuotas lekseris, kuris vienodai analizuoja abu failus.
4. Pasiektas magistro baigiamojo darbo tikslas sukūrus žiniatinklio pakeitimo aptikimo metodą, tikrinantį interneto puslapius pagal jų šablonus ir juose esančią HTML struktūrą bei teksto turinį.
5. Siūlomas žiniatinklio pakeitimo aptikimo metodo algoritmas realizuotas programiniame kode.
6. Atlikti realizuoto žiniatinklio pakeitimo aptikimo metodo prototipo sintetiniai bandymai parodė, kad sukurtas specializuotas lekseris yra vidutiniškai apie 3 kartus greitesnis už *Roslyn* lekserį, o tikrinimo tikslumas yra iki 100 proc. tikslus.
7. Sukurtas žiniatinklio pakeitimo aptikimo metodas gali būti toliau tobulinamas pridedant *JavaScript* kodo analizę, šablonų failų integralumo tikrinimą bei dirbtinio intelekto fiktyvią, nuspėjamąją kompiliaciją bei lyginimą.

Literatūros sąrašas

1. BERNERS-LEE, T. [interaktyvus]. .1989. [žiūrėta 2023-01-12]. Prieiga per internetą: <https://www.w3.org/History/1989/proposal.html>.

2. ANDREESSEN, M. - BINA, E. NCSA Mosaic: A Global Hypermedia System. In *Internet Research* . 1994. Vol. 4, no. 1, p. 7–17. .

3. THE HISTORY OF DOMAINS [interaktyvus]. .Prieiga per internetą: <https://www.historyofdomains.com/netscape/>.

4. AGHAEI, S. Evolution of the World Wide Web : From Web 1.0 to Web 4.0. In *International journal of Web & Semantic Technology* . 2012. Vol. 3, no. 1, p. 1–10. .

5. W3C Architecture of the World Wide Web. In *W3C Recommendation* [interaktyvus]. 2004. [žiūrėta 2023-01-12]. Prieiga per internetą: <https://www.w3.org/TR/webarch/>.

6. FENWICK, M. - JURCYS, P. The Contested Meaning of Web3 and Why it Matters for (IP) Lawyers. In *SSRN Electronic Journal* [interaktyvus]. 2022. [žiūrėta 2023-01-12]. . Prieiga per internetą: <https://www.ssrn.com/abstract=4017790>.

7. ALBALAWI, M. ir kt. Website Defacement Detection and Monitoring Methods: A Review. In *Electronics* . 2022. Vol. 11, no. 21, p. 3573. .

8. CDNETWORKS INC. [interaktyvus]. .[s.l.]: CDNetworks Inc., 2022. [žiūrėta 2023-01-13]. Prieiga per internetą: <https://www.cdnetworks.com/wp-content/uploads/2022/11/CDNetworks-State-of-Web-Security-H1-2022.pdf>.

9. EMMONS, T. ir kt. [interaktyvus]. .[s.l.]: Akamai, 2022. [žiūrėta 2023-01-13]. Prieiga per internetą: <https://www.akamai.com/resources/research-paper/akamai-web-application-and-api-threat-report>.

10. NIDECKI, T.A. Remote file inclusion (RFI). In [interaktyvus]. Prieiga per internetą: <https://www.invicti.com/learn/remote-file-inclusion-rfi/>.

11. CHANNELL, J. What is a Website Defacement? In *Sucuri* [interaktyvus]. 2020. [žiūrėta 2023-01-14]. Prieiga per internetą: <https://blog.sucuri.net/2020/06/what-is-a-website-defacement.html>.

12. BEN MARTIN [interaktyvus]. 2015. [žiūrėta 2024-04-20]. Prieiga per internetą: <https://blog.sucuri.net/2015/01/website-hacks-defacements-2014.html>.

13. HOANG, X.D. - NGUYEN, N.T. A Multi-layer Model for Website Defacement Detection. In *Proceedings of the Tenth International Symposium on Information and Communication Technology - SoICT 2019* [interaktyvus]. Hanoi, Ha Long Bay, Viet Nam: ACM Press, 2019. p. 508–513. [žiūrėta 2022-12-14]. Prieiga per internetą: <http://dl.acm.org/citation.cfm?doid=3368926.3369730>.

14. REISENBERGER, B. *Enhanced Defacement Detection using Text Analysis* [interaktyvus]. Linz: Universität Linz, 2021. Prieiga per internetą: <https://epub.jku.at/obvulihs/download/pdf/5939150>.

15. KOTHARI, R. - VYAS, G. An Image Processing Based Approach for Monitoring Changes in Webpages. In SMYS, S. ir kt.Sud. *Computational Vision and Bio-Inspired Computing* [interaktyvus]. Cham: Springer International Publishing, 2020. p. 974–982. [žiūrėta 2022-12-14]. ISBN 978-3-030-37217-0Prieiga per internetą: <http://link.springer.com/10.1007/978-3-030-37218-7_103>.

16. HOANG, X.D. A Website Defacement Detection Method Based on Machine Learning Techniques. In *Proceedings of the Ninth International Symposium on Information and Communication Technology - SoICT 2018* [interaktyvus]. Danang City, Viet Nam: ACM Press, 2018. p. 443–448. [žiūrėta 2024-05-16]. Prieiga per internetą: <http://dl.acm.org/citation.cfm?doid=3287921.3287975>.

17. LIU, L. ir kt. *WebCQ* -detecting and delivering information changes on the web. In *Proceedings of the ninth international conference on Information and knowledge management - CIKM '00* [interaktyvus]. McLean, Virginia, United States: ACM Press, 2000. p. 512–519. [žiūrėta 2022-12-14]. Prieiga per internetą: <http://portal.acm.org/citation.cfm?doid=354756.354860>.

18. MASANGO, M. ir kt. An Approach for Detecting Web Defacement with Self-healing Capabilities. In GAVRILOVA, M.L. ir kt.Sud. *Transactions on Computational Science XXXII* [interaktyvus]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018. p. 29–42. [žiūrėta 2023-05-23]. ISBN 978-3-662-56671-8Prieiga per internetą: <http://link.springer.com/10.1007/978-3-662-56672-5_3>.

19. MAO, B.-M. - BAGOLIBE, K.D. A Contribution to Detect and Prevent a Website Defacement. In *2019 International Conference on Cyberworlds (CW)* [interaktyvus]. Kyoto, Japan: IEEE, 2019. p. 344–347. [žiūrėta 2023-05-21]. Prieiga per internetą: <https://ieeexplore.ieee.org/document/8919158/>.

20. AKHI, M. - GHAZIZADEH, N. [interaktyvus]. .[s.l.]: arXiv, 2023. [žiūrėta 2024-05-16]. arXiv:2310.03891 [cs]. Prieiga per internetą: <http://arxiv.org/abs/2310.03891>.

21. NGUYEN, T.H. ir kt. Detecting Website Defacement Attacks using Web-page Text and Image Features. In *International Journal of Advanced Computer Science and Applications* [interaktyvus]. 2021. Vol. 12, no. 7. [žiūrėta 2024-05-16]. . Prieiga per internetą: <http://thesai.org/Publications/ViewPaper?Volume=12&Issue=7&Code=IJACSA&SerialNo=25>.

22. BORGOLTE, K. ir kt. Meerkat: Detecting Website Defacements through Image-based Object Recognition. In *24th USENIX Security Symposium (USENIX Security 15)* [interaktyvus]. Washington, D.C.: USENIX Association, 2015. p. 595–610. Prieiga per internetą: <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/borgolte>.

23. CARVALHO, F.M. ir kt. Text Web Templates Considered Harmful. In BOZZON, A. ir kt.Sud. *Web Information Systems and Technologies* [interaktyvus]. Cham: Springer International Publishing, 2020. p. 69–95. [žiūrėta 2024-05-15]. ISBN 978-3-030-61749-3Prieiga per internetą: <http://link.springer.com/10.1007/978-3-030-61750-9_4>.

24. JAN LEHNARDT ir kt. mustache.js - Logic-less {{mustache}} templates with JavaScript. In *GitHub* [interaktyvus]. [žiūrėta 2024-05-15]. Prieiga per internetą: <https://github.com/janl/mustache.js>.

25. CHRIS WANSTRATH ir kt. Mustache. In *GitHub* [interaktyvus]. 2016. [žiūrėta 2024-04-21]. Prieiga per internetą: <https://github.com/mustache/mustache/blob/master/README.md>.

26. LUKE LATHAM ASP.NET Core Razor components. In *Microsoft Learn* [interaktyvus]. 2024. [žiūrėta 2024-04-29]. Prieiga per internetą: <https://learn.microsoft.com/en-us/aspnet/core/blazor/components/?view=aspnetcore-8.0>.

27. RICK ANDERSON ir kt. Razor syntax reference for ASP.NET Core. In [interaktyvus]. Prieiga per internetą: <https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-8.0>.

28. TOM FITZMACKEN Introduction to ASP.NET Web Programming Using the Razor Syntax. In *Microsoft Learn* [interaktyvus]. 2023. [žiūrėta 2024-04-21]. Prieiga per internetą: <https://learn.microsoft.com/en-us/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-c>.

29. BILL WAGNER The .NET Compiler Platform SDK. In *Microsoft Learn* [interaktyvus]. 2021. [žiūrėta 2024-04-29]. Prieiga per internetą: <https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/>.

30. WEB HYPERTEXT APPLICATION TECHNOLOGY WORKING GROUP (WHATWG) HTML Living Standard. In [interaktyvus]. 2024. [žiūrėta 2024-05-11]. Prieiga per internetą: <https://html.spec.whatwg.org>.

1 Priedas. Razor lekseris

LexerEnums.cs

```
 1. namespace TemplateDiff.Models.Lexer
 2. {
 3.     public static class LexerEnums
 4.     {
 5.         public enum LexerTokenType
 6.         {
 7.             EOF = 0,
 8.             /// <summary>
 9.             /// A C# open curly brace `{`
10.             /// </summary>
11.             CodeBlockStart,
12.
13.             /// <summary>
14.             /// A C# close curly brace `}`
15.             /// </summary>
16.             CodeBlockEnd,
17.
18.             /// <summary>
19.             /// A Razor or C# opening parentheses `(`
20.             /// </summary>
21.             CodeParenthesesStart,
22.
23.             /// <summary>
24.             /// A Razor or C# closing parentheses `)`
25.             /// </summary>
26.             CodeParenthesesEnd,
27.
28.             /// <summary>
29.             /// An inline Razor expression.
30.             /// <code>@(monthSum * 12)</code>
31.             /// </summary>
32.             InfereredExpression,
33.
34.             /// <summary>
35.             /// A C# variable keyword like `var`, `int, `string`, etc.
36.             /// </summary>
37.             VariableKeyword,
38.
39.             /// <summary>
40.             /// A C# variable name after `var` `int` etc. E.g. the `count` in `var count`
41.             /// </summary>
42.             VariableName,
43.
44.             /// <summary>
45.             /// Data that comes after the equals. E.g. the "strval" in string str = "strval"
46.             /// </summary>
47.             VariableData,
48.
49.             /// <summary>
50.             /// A C# variable name reference. E.g. the variable 'a' in
51.             /// <code>
52.             /// @var a = "";
53.             /// @a
54.             /// </code>
55.             /// </summary>
56.             VariableNameReference,
57.
58.             /// <summary>
59.             /// The equals symbol in C# code. Works with '++' self asignment. E.g. the = in var a
= 1 or i++
60.             /// </summary>
61.             Assignment,
62.
63.             /// <summary>
64.             /// The semicolon in C# ends the variable. E.g. the ; in var a = 1;
65.             /// </summary>
66.             Semicolon,
67.
68.             /// <summary>
```

```
69.             /// A C# conditional logic keyword like `if`, `else`, `switch`, etc.
70.             /// </summary>
71.             ConditionalKeyword,
72.
73.             /// <summary>
74.             /// The condition inside of an expression in <see cref="ConditionalKeyword"/> or <see
cref="CycleKeyword"/>.
75.             /// For example the 'true' in <code>if (true) { };</code>
76.             /// </summary>
77.             Condition,
78.
79.             /// <summary>
80.             /// A C# cycle keyword like `for`, `foreach`, `while`, etc.
81.             /// </summary>
82.             CycleKeyword,
83.
84.             /// <summary>
85.             /// The razor `@` symbol
86.             /// </summary>
87.             RazorDirective,
88.
89.             /// <summary>
90.             /// Not implemented / error parsing
91.             /// </summary>
92.             Unknown,
93.
94.             Text,
95.             OpeningTag,
96.             ClosingTag,
97.             ClosingAngle,
98.             SelfClosingTag,
99.             CodeComment,
100.         }
101.     }
102. }
```

ILexerToken.cs

```
1. namespace TemplateDiff.Models.Lexer
2. {
3.     using System.Diagnostics.CodeAnalysis;
4.     using TemplateDiff.Models.Lexer.Html;
5.     using static TemplateDiff.Models.Lexer.LexerEnums;
6.
7.     public interface ILexerToken
8.     {
9.         public LexerTokenType Type { get; set; }
10.
11.         public bool isInRazorSyntax { get; set; }
12.
13.         public string Content { get; set; }
14.
15.         public ILexerToken? PreviousToken { get; set; }
16.
17.         public ILexerToken? NextToken { get; set; }
18.     }
19.
20.     public class LexerTokenComparer : IEqualityComparer<ILexerToken>
21.     {
22.         public bool Equals(ILexerToken? x, ILexerToken? y)
23.         {
24.             if (x.GetType() != y.GetType())
25.             {
26.                 return false;
27.             }
28.
29.             if (x.GetType() == typeof(HtmlLexerToken))
30.             {
31.                 if (!(x as HtmlLexerToken)!.Equals(y as HtmlLexerToken))
32.                 {
33.                     return false;
34.                 }
```

```
35.             }
36.
37.             return x.Type == y.Type && x.isInRazorSyntax == y.isInRazorSyntax &&
x.Content.Equals(y.Content);
38.         }
39.
40.         public int GetHashCode([DisallowNull] ILexerToken obj)
41.         {
42.             return obj.GetType().GetHashCode() ^
43.                 obj.Type.GetHashCode() ^
44.                 obj.isInRazorSyntax.GetHashCode() ^
45.                 obj.Content.GetHashCode();
46.         }
47.     }
48. }
49.
```

LexerToken.cs

```
 1. namespace TemplateDiff.Models.Lexer
 2. {
 3.     using System.Text;
 4.     using static TemplateDiff.Models.Lexer.LexerEnums;
 5.
 6.     public class LexerToken : ILexerToken
 7.     {
 8.         public LexerTokenType Type { get; set; }
 9.
10.         public bool isInRazorSyntax { get; set; }
11.
12.         public string Content { get; set; }
13.
14.         public ILexerToken? PreviousToken { get; set; }
15.
16.         public ILexerToken? NextToken { get; set; }
17.
18.         public LexerToken (LexerTokenType type, bool isInRazor, string content)
19.         {
20.             Type = type;
21.             isInRazorSyntax = isInRazor;
22.             Content = content;
23.         }
24.     }
25. }
```

HtmlLexerToken.cs

```
 1. namespace TemplateDiff.Models.Lexer.Html
 2. {
 3.     using System;
 4.     using System.Text;
 5.     using static TemplateDiff.Models.Lexer.LexerEnums;
 6.
 7.     public class HtmlLexerToken(LexerTokenType type, bool inRazorSyntax, string content)
 8.         : LexerToken(type, inRazorSyntax, content),
 9.         IEquatable<HtmlLexerToken>
10.     {
11.         public ICollection<HtmlLexerAttributeToken>? Attributes { get; set; }
12.
13.         public bool Equals(HtmlLexerToken? other)
14.         {
15.             if (other is null)
16.             {
17.                 return false;
18.             }
19.
20.             return
21.                 Type == other.Type &&
22.                 isInRazorSyntax == other.isInRazorSyntax &&
23.                 Content.Equals(other.Content) &&
24.                     ((this.Attributes is null && other.Attributes is null) ||
```

```
                    (this.Attributes is not null && other.Attributes is not null &&
                    this.Attributes.SequenceEqual(other.Attributes)));
        }

        public bool ContentEquals(HtmlLexerToken? other)
        {
            if (other is null)
            {
                return false;
            }

            return Type == other.Type &&
                Content.Equals(other.Content) &&
                    ((this.Attributes is null && other.Attributes is null) ||
                    (this.Attributes is not null && other.Attributes is not null &&
                    this.Attributes.SequenceEqual(other.Attributes)));
        }

        public override int GetHashCode() => base.GetHashCode();

        bool IEquatable<HtmlLexerToken>.Equals(HtmlLexerToken? other) => Equals(other);

        public override string ToString()
        {
            if (Type == LexerTokenType.Text || Type == LexerTokenType.ClosingAngle)
            {
                return Content;
            }

            StringBuilder sb = new StringBuilder();
            if (Type == LexerTokenType.OpeningTag)
            {
                sb.Append('<');
                sb.Append(Content);
                if (Attributes is not null && Attributes.Count > 0)
                {
                    foreach (var attribute in Attributes)
                    {
                        sb.Append(' ');
                        sb.Append(@$"{attribute.Name}=""{attribute.Value}""");
                    }
                }
            }
            else if (Type == LexerTokenType.ClosingTag)
            {
                sb.Append('<');
                sb.Append('/');
                sb.Append(Content);
            }

            return sb.ToString();
        }
    }
}

```

HtmlLexerAttributeToken.cs

```
namespace TemplateDiff.Models.Lexer.Html
{
    public class HtmlLexerAttributeToken : IEquatable<HtmlLexerAttributeToken>
    {
        public string Name { get; }

        public string? Value { get; }

        public HtmlLexerAttributeToken(string name, string? value)
        {
            Name = name;
            Value = value;
        }

```

```
15.         public bool Equals (HtmlLexerAttributeToken? other)
16.         {
17.             if (other == null)
18.             {
19.                 return false;
20.             }
21.
22.             return (Name is null ? other.Name is null : Name.Equals(other.Name)) &&
23.                    (Value is null ? other.Value is null : Value.Equals(other.Value));
24.         }
25.
26.         bool IEquatable<HtmlLexerAttributeToken>.Equals(HtmlLexerAttributeToken? other) =>
Equals(other);
27.
28.         public override string ToString() => $"{Name}={Value}";
29.     }
30. }
```

RazorLexerToken.cs

```
 1. namespace TemplateDiff.Models.Lexer.Razor
 2. {
 3.     using System.Text;
 4.     using static TemplateDiff.Models.Lexer.LexerEnums;
 5.
 6.     public class RazorLexerToken : LexerToken
 7.     {
 8.         public RazorLexerToken(LexerTokenType type, bool inRazorSyntax, string? content) :
base(type, inRazorSyntax, content)
 9.         {
10.         }
11.
12.         public override string ToString()
13.         {
14.             return Content;
15.         }
16.     }
17. }
```

2 Priedas. Razor lekserio vienetiniai testai

Tik HTML sintaksės testai:

```
 1. namespace UnitTests.Lexer
 2. {
 3.     using System;
 4.     using System.Collections.Generic;
 5.     using System.Linq;
 6.     using static TemplateDiff.Models.Lexer.LexerEnums;
 7.     using TemplateDiff.Lexer;
 8.     using TemplateDiff.Models.Lexer.Html;
 9.     using TemplateDiff.Models.Lexer;
10.     using System.Text.Json;
11.
12.     public class RazorLexerHtmlTests
13.     {
14.         [Fact]
15.         public void Parses_Tag()
16.         {
17.             // Arrange
18.             var html = @"<html></html>";
19.             var expectedTokens = new List<ILexerToken>()
20.             {
21.                 new HtmlLexerToken(LexerTokenType.OpeningTag, false, "html"),
22.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
23.                 new HtmlLexerToken(LexerTokenType.ClosingTag, false, "html"),
24.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
25.             };
26.             var lexer = new RazorLexer(new StringReader(html));
27.
28.             // Act
29.             lexer.Start();
30.
31.             // Assert
32.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
33.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
34.         }
35.
36.         [Fact]
37.         public void Parses_Tag_Self_Closing()
38.         {
39.             // Arrange
40.             var html = @"<br />"; // space between 'br' and '/' is intentional.
41.             var expectedTokens = new List<ILexerToken>()
42.             {
43.                 new HtmlLexerToken(LexerTokenType.SelfClosingTag, false, "br"),
44.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
45.             };
46.             var lexer = new RazorLexer(new StringReader(html));
47.
48.             // Act
49.             lexer.Start();
50.
51.             // Assert
52.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
53.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
54.         }
55.
56.         [Fact]
57.         public void Parses_Tag_Self_Closing_With_Properties()
58.         {
59.             // Arrange
60.             var html = @"<img src=""image/circle"" alt=""image"" />"; // space between 'br' and
'/' is intentional.
61.             var expectedTokens = new List<ILexerToken>()
62.             {
63.                 new HtmlLexerToken(LexerTokenType.SelfClosingTag, false, "img")
64.                 {
65.                     Attributes =
66.                     [
67.                         new HtmlLexerAttributeToken("src", "image/circle"),
68.                         new HtmlLexerAttributeToken("alt", "image")
```

```
                    ]
                },
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
            };
            var lexer = new RazorLexer(new StringReader(html));

            // Act
            lexer.Start();

            // Assert
            Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
            Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
        }

        [Fact]
        public void Parses_Tag_Unclosed()
        {
            // Arrange
            var html = @"<meta><style type=""text/css""></style>";
            var expectedTokens = new List<ILexerToken>()
            {
                new HtmlLexerToken(LexerTokenType.OpeningTag, false, "meta"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.OpeningTag, false, "style")
                {
                    Attributes = [new HtmlLexerAttributeToken("type", "text/css")]
                },
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.ClosingTag, false, "style"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
            };
            var lexer = new RazorLexer(new StringReader(html));

            // Act
            lexer.Start();

            // Assert
            Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
            Assert.True(expectedTokens.SequenceEqual(lexer.Tokens.ToList(), new
LexerTokenComparer()));
        }

        [Fact]
        public void Parses_Tags_Nested()
        {
            // Arrange
            var html = @"<html><head></head><body><h1>hello</h1></body></html>";
            var expectedTokens = new List<ILexerToken>()
            {
                new HtmlLexerToken(LexerTokenType.OpeningTag, false, "html"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.OpeningTag, false, "head"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.ClosingTag, false, "head"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.OpeningTag, false, "body"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.OpeningTag, false, "h1"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.Text, false, "hello"),
                new HtmlLexerToken(LexerTokenType.ClosingTag, false, "h1"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.ClosingTag, false, "body"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.ClosingTag, false, "html"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
            };
            var lexer = new RazorLexer(new StringReader(html));

            // Act
            lexer.Start();

            // Assert
```

```
141.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
142.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens.ToList(), new
LexerTokenComparer()));
143.         }
144.
145.         [Fact]
146.         public void Parses_Tag_With_Attributes()
147.         {
148.             // Arrange
149.             var html = @"<h1 class=""center"" id=""header1""></h1>";
150.             var expectedTokens = new List<ILexerToken>()
151.             {
152.                 new HtmlLexerToken(LexerTokenType.OpeningTag, false, "h1")
153.                 {
154.                     Attributes =
155.                     [
156.                         new HtmlLexerAttributeToken("class", "center"),
157.                         new HtmlLexerAttributeToken("id", "header1")
158.                     ],
159.                 },
160.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
161.                 new HtmlLexerToken(LexerTokenType.ClosingTag, false, "h1"),
162.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
163.             };
164.             var lexer = new RazorLexer(new StringReader(html));
165.
166.             // Act
167.             lexer.Start();
168.
169.             // Assert
170.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
171.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens.ToList(), new
LexerTokenComparer()));
172.             for (int i = 0; i < expectedTokens.Count; i++)
173.             {
174.                 var expectedToken = expectedTokens[i] as HtmlLexerToken;
175.                 var actualToken = lexer.Tokens.ElementAt(i) as HtmlLexerToken;
176.                 Assert.Equal(expectedToken, actualToken);
177.             }
178.         }
179.     }
180. }
181.
```

Razor sintaksės testai:

```
 1. namespace UnitTests.Lexer
 2. {
 3.     using System;
 4.     using System.Collections.Generic;
 5.     using System.Linq;
 6.     using static TemplateDiff.Models.Lexer.LexerEnums;
 7.     using TemplateDiff.Lexer;
 8.     using TemplateDiff.Models.Lexer.Html;
 9.     using TemplateDiff.Models.Lexer;
10.     using TemplateDiff.Models.Lexer.Razor;
11.
12.     public class RazorLexerCshtmlTests
13.     {
14.         [Fact]
15.         public void Parses_Razor_Html_Tag()
16.         {
17.             // Arrange
18.             var html = @"@{<html></html>}";
19.             var expectedTokens = new List<ILexerToken>()
20.             {
21.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
22.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
23.                 new HtmlLexerToken(LexerTokenType.OpeningTag, true, "html"),
24.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
25.                 new HtmlLexerToken(LexerTokenType.ClosingTag, true, "html"),
26.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
```

```
27.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
28.             };
29.             var lexer = new RazorLexer(new StringReader(html));
30.
31.             // Act
32.             lexer.Start();
33.
34.             // Assert
35.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
36.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
37.         }
38.
39.         [Fact]
40.         public void Parses_Razor_Variable_Call_In_Tag_Only()
41.         {
42.             // Arrange
43.             var html = @"<p>@variable</p>";
44.             var expectedTokens = new List<ILexerToken>()
45.             {
46.                 new HtmlLexerToken(LexerTokenType.OpeningTag, false, "p"),
47.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
48.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
49.                 new RazorLexerToken(LexerTokenType.VariableNameReference, true, "variable"),
50.                 new HtmlLexerToken(LexerTokenType.ClosingTag, false, "p"),
51.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
52.             };
53.             var lexer = new RazorLexer(new StringReader(html));
54.
55.             // Act
56.             lexer.Start();
57.
58.             // Assert
59.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
60.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
61.         }
62.
63.         [Fact]
64.         public void Parses_Razor_Variable_String_Assign()
65.         {
66.             // Arrange
67.             var html = @"@{var a = ""A"";}";
68.             var expectedTokens = new List<ILexerToken>()
69.             {
70.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
71.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
72.                 new RazorLexerToken(LexerTokenType.VariableKeyword, true, "var"),
73.                 new RazorLexerToken(LexerTokenType.VariableName, true, "a"),
74.                 new RazorLexerToken(LexerTokenType.Assignment, true, "="),
75.                 new RazorLexerToken(LexerTokenType.VariableData, true, @"""A"""),
76.                 new RazorLexerToken(LexerTokenType.Semicolon, true, ";"),
77.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
78.             };
79.             var lexer = new RazorLexer(new StringReader(html));
80.
81.             // Act
82.             lexer.Start();
83.
84.             // Assert
85.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
86.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
87.         }
88.
89.         [Fact]
90.         public void Parses_Razor_String_With_Space_Assign()
91.         {
92.             // Arrange
93.             var html = @"@{string str = ""A B"";}";
94.             var expectedTokens = new List<ILexerToken>()
95.             {
96.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
97.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
98.                 new RazorLexerToken(LexerTokenType.VariableKeyword, true, "string"),
99.                 new RazorLexerToken(LexerTokenType.VariableName, true, "str"),
```

```
100.                 new RazorLexerToken(LexerTokenType.Assignment, true, "="),
101.                 new RazorLexerToken(LexerTokenType.VariableData, true, @"""A B"""),
102.                 new RazorLexerToken(LexerTokenType.Semicolon, true, ";"),
103.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
104.             };
105.             var lexer = new RazorLexer(new StringReader(html));
106. 
107.             // Act
108.             lexer.Start();
109. 
110.             // Assert
111.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
112.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
113.         }
114. 
115.         [Fact]
116.         public void Parses_Razor_Variable_Assign_Nospace()
117.         {
118.             // Arrange
119.             var html = @"@{var a=""A"";}";
120.             var expectedTokens = new List<ILexerToken>()
121.             {
122.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
123.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
124.                 new RazorLexerToken(LexerTokenType.VariableKeyword, true, "var"),
125.                 new RazorLexerToken(LexerTokenType.VariableName, true, "a"),
126.                 new RazorLexerToken(LexerTokenType.Assignment, true, "="),
127.                 new RazorLexerToken(LexerTokenType.VariableData, true, @"""A"""),
128.                 new RazorLexerToken(LexerTokenType.Semicolon, true, ";"),
129.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
130.             };
131.             var lexer = new RazorLexer(new StringReader(html));
132. 
133.             // Act
134.             lexer.Start();
135. 
136.             // Assert
137.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
138.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
139.         }
140. 
141.         [Fact]
142.         public void Parses_Razor_Variable_Assign_Newline()
143.         {
144.             // Arrange
145.             var html = @"@{var a =
146. ""A"";}";
147.             var expectedTokens = new List<ILexerToken>()
148.             {
149.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
150.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
151.                 new RazorLexerToken(LexerTokenType.VariableKeyword, true, "var"),
152.                 new RazorLexerToken(LexerTokenType.VariableName, true, "a"),
153.                 new RazorLexerToken(LexerTokenType.Assignment, true, "="),
154.                 new RazorLexerToken(LexerTokenType.VariableData, true, @"""A"""),
155.                 new RazorLexerToken(LexerTokenType.Semicolon, true, ";"),
156.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
157.             };
158.             var lexer = new RazorLexer(new StringReader(html));
159. 
160.             // Act
161.             lexer.Start();
162. 
163.             // Assert
164.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
165.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
166.         }
167. 
168.         [Fact]
169.         public void Parses_Razor_Variable_Assign_String_Array()
170.         {
171.             // Arrange
172.             var html = @"@{string[] strs = {""a"", ""b""};}";
```

```
173.             var expectedTokens = new List<ILexerToken>()
174.             {
175.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
176.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
177.                 new RazorLexerToken(LexerTokenType.VariableKeyword, true, "string[]"),
178.                 new RazorLexerToken(LexerTokenType.VariableName, true, "strs"),
179.                 new RazorLexerToken(LexerTokenType.Assignment, true, "="),
180.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
181.                 new RazorLexerToken(LexerTokenType.Unknown, true, @"""a"""),
182.                 new RazorLexerToken(LexerTokenType.Unknown, true, @","),
183.                 new RazorLexerToken(LexerTokenType.Unknown, true, @"""b"""),
184.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
185.                 new RazorLexerToken(LexerTokenType.Semicolon, true, ";"),
186.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
187.             };
188.             var lexer = new RazorLexer(new StringReader(html));
189. 
190.             // Act
191.             lexer.Start();
192. 
193.             // Assert
194.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
195.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
196.         }
197. 
198.         [Fact]
199.         public void Parses_Razor_Variable_Call_In_Tag_Beginning()
200.         {
201.             // Arrange
202.             var razor = @"<p>@greeting, friends!</p>";
203.             var expectedTokens = new List<ILexerToken>()
204.             {
205.                 new HtmlLexerToken(LexerTokenType.OpeningTag, false, "p"),
206.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
207.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
208.                 new RazorLexerToken(LexerTokenType.VariableNameReference, true, "greeting"),
209.                 new HtmlLexerToken(LexerTokenType.Text, false, ", friends!"),
210.                 new HtmlLexerToken(LexerTokenType.ClosingTag, false, "p"),
211.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
212.             };
213.             var lexer = new RazorLexer(new StringReader(razor));
214. 
215.             // Act
216.             lexer.Start();
217. 
218.             // Assert
219.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
220.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
221.         }
222. 
223.         [Fact]
224.         public void Parses_Razor_Variable_Call_In_Tag_Middle()
225.         {
226.             // Arrange
227.             var razor = @"<p>one @two three</p>";
228.             var expectedTokens = new List<ILexerToken>()
229.             {
230.                 new HtmlLexerToken(LexerTokenType.OpeningTag, false, "p"),
231.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
232.                 new HtmlLexerToken(LexerTokenType.Text, false, "one "),
233.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
234.                 new RazorLexerToken(LexerTokenType.VariableNameReference, true, "two"),
235.                 new HtmlLexerToken(LexerTokenType.Text, false, " three"),
236.                 new HtmlLexerToken(LexerTokenType.ClosingTag, false, "p"),
237.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
238.             };
239.             var lexer = new RazorLexer(new StringReader(razor));
240. 
241.             // Act
242.             lexer.Start();
243. 
244.             // Assert
245.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
```

```
246.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
247.         }
248.
249.         [Fact]
250.         public void Parses_Razor_Variable_Call_In_Tag_End()
251.         {
252.             // Arrange
253.             var razor = @"<p>one @two</p>";
254.             var expectedTokens = new List<ILexerToken>()
255.             {
256.                 new HtmlLexerToken(LexerTokenType.OpeningTag, false, "p"),
257.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
258.                 new HtmlLexerToken(LexerTokenType.Text, false, "one "),
259.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
260.                 new RazorLexerToken(LexerTokenType.VariableNameReference, true, "two"),
261.                 new HtmlLexerToken(LexerTokenType.ClosingTag, false, "p"),
262.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
263.             };
264.             var lexer = new RazorLexer(new StringReader(razor));
265.
266.             // Act
267.             lexer.Start();
268.
269.             // Assert
270.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
271.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
272.         }
273.
274.         [Fact]
275.         public void Parses_Razor_Inline_Expression_In_Html()
276.         {
277.             // Arrange
278.             var razor = @"<p>Amount: @( monthlyTotal * 12)</p>";
279.             var expectedTokens = new List<ILexerToken>()
280.             {
281.                 new HtmlLexerToken(LexerTokenType.OpeningTag, false, "p"),
282.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
283.                 new HtmlLexerToken(LexerTokenType.Text, false, "Amount: "),
284.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
285.                 new RazorLexerToken(LexerTokenType.CodeParenthesesStart, true, "("),
286.                 new RazorLexerToken(LexerTokenType.InfereredExpression, true, "monthlyTotal*12"),
287.                 new RazorLexerToken(LexerTokenType.CodeParenthesesEnd, true, ")"),
288.                 new HtmlLexerToken(LexerTokenType.ClosingTag, false, "p"),
289.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
290.             };
291.             var lexer = new RazorLexer(new StringReader(razor));
292.
293.             // Act
294.             lexer.Start();
295.
296.             // Assert
297.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
298.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
299.         }
300.
301.         [Fact]
302.         public void Parses_Razor_RazorDirective_Escape()
303.         {
304.             // Arrange
305.             var razor = @"<p>@@Username</p>";
306.             var expectedTokens = new List<ILexerToken>()
307.             {
308.                 new HtmlLexerToken(LexerTokenType.OpeningTag, false, "p"),
309.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
310.                 new HtmlLexerToken(LexerTokenType.Text, false, "@Username"),
311.                 new HtmlLexerToken(LexerTokenType.ClosingTag, false, "p"),
312.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
313.             };
314.             var lexer = new RazorLexer(new StringReader(razor));
315.
316.             // Act
317.             lexer.Start();
318.
```

```
319.             // Assert
320.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
321.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
322.         }
323.
324.         [Fact]
325.         public void Parses_Razor_Code_Comment()
326.         {
327.             // Arrange
328.             var razor = @"@*@{/* C# comment */// Another C# comment}<!-- HTML comment -->*@";
329.             var expectedTokens = new List<ILexerToken>()
330.             {
331.                 new RazorLexerToken(LexerTokenType.CodeComment, true, "@{/* C# comment *///
Another C# comment}<!-- HTML comment -->"),
332.             };
333.             var lexer = new RazorLexer(new StringReader(razor));
334.
335.             // Act
336.             lexer.Start();
337.
338.             // Assert
339.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
340.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
341.         }
342.
343.         [Fact]
344.         public void Parses_Razor_Code_Conditional()
345.         {
346.             // Arrange
347.             var razor = @"@{if(true){}else{}}";
348.             var expectedTokens = new List<ILexerToken>()
349.             {
350.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
351.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
352.                 new RazorLexerToken(LexerTokenType.ConditionalKeyword, true, "if"),
353.                 new RazorLexerToken(LexerTokenType.CodeParenthesesStart, true, "("),
354.                 new RazorLexerToken(LexerTokenType.Condition, true, "true"),
355.                 new RazorLexerToken(LexerTokenType.CodeParenthesesEnd, true, ")"),
356.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
357.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
358.                 new RazorLexerToken(LexerTokenType.ConditionalKeyword, true, "else"),
359.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
360.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
361.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}")
362.             };
363.             var lexer = new RazorLexer(new StringReader(razor));
364.
365.             // Act
366.             lexer.Start();
367.
368.             // Assert
369.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
370.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
371.         }
372.
373.         [Fact]
374.         public void Parses_Razor_Code_Cycle_While()
375.         {
376.             // Arrange
377.             var razor = @"@{while(true){}}";
378.             var expectedTokens = new List<ILexerToken>()
379.             {
380.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
381.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
382.                 new RazorLexerToken(LexerTokenType.CycleKeyword, true, "while"),
383.                 new RazorLexerToken(LexerTokenType.CodeParenthesesStart, true, "("),
384.                 new RazorLexerToken(LexerTokenType.Condition, true, "true"),
385.                 new RazorLexerToken(LexerTokenType.CodeParenthesesEnd, true, ")"),
386.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
387.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
388.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}")
389.             };
390.             var lexer = new RazorLexer(new StringReader(razor));
```

```
391.
392.             // Act
393.             lexer.Start();
394.
395.             // Assert
396.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
397.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
398.         }
399.
400.         [Fact]
401.         public void Parses_Razor_Code_Cycle_For()
402.         {
403.             // Arrange
404.             var razor = @"@{for(int i = 0; i < 2; i++){}}";
405.             var expectedTokens = new List<ILexerToken>()
406.             {
407.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
408.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
409.                 new RazorLexerToken(LexerTokenType.CycleKeyword, true, "for"),
410.                 new RazorLexerToken(LexerTokenType.CodeParenthesesStart, true, "("),
411.                 new RazorLexerToken(LexerTokenType.Condition, true, "int i = 0; i < 2; i++"),
412.                 new RazorLexerToken(LexerTokenType.CodeParenthesesEnd, true, ")"),
413.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
414.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
415.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}")
416.             };
417.             var lexer = new RazorLexer(new StringReader(razor));
418.
419.             // Act
420.             lexer.Start();
421.
422.             // Assert
423.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
424.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
425.         }
426.
427.         [Fact]
428.         public void Parses_Razor_Code_Cycle_Foreach_With_Html()
429.         {
430.             // Arrange
431.             var razor =
432. @"@{
433.     foreach (var item in items)
434.     {
435.         <p>@item</p>
436.     }
437. }";
438.             var expectedTokens = new List<ILexerToken>()
439.             {
440.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
441.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
442.                 new RazorLexerToken(LexerTokenType.CycleKeyword, true, "foreach"),
443.                 new RazorLexerToken(LexerTokenType.CodeParenthesesStart, true, "("),
444.                 new RazorLexerToken(LexerTokenType.Condition, true, "var item in items"),
445.                 new RazorLexerToken(LexerTokenType.CodeParenthesesEnd, true, ")"),
446.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
447.                 new HtmlLexerToken(LexerTokenType.OpeningTag, true, "p"),
448.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
449.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
450.                 new RazorLexerToken(LexerTokenType.VariableNameReference, true, "item"),
451.                 new HtmlLexerToken(LexerTokenType.ClosingTag, true, "p"),
452.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
453.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
454.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}")
455.             };
456.             var lexer = new RazorLexer(new StringReader(razor));
457.
458.             // Act
459.             lexer.Start();
460.
461.             // Assert
462.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
463.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
```

```
464.         }
465.
466.         [Fact]
467.         public void Parses_Razor_Code_Conditional_If_With_Html()
468.         {
469.             // Arrange
470.             var razor =
471. @"@{
472.     if (true)
473.     {
474.         <h1>Header 1</h1>
475.     }
476. }";
477.             var expectedTokens = new List<ILexerToken>()
478.             {
479.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
480.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
481.                 new RazorLexerToken(LexerTokenType.ConditionalKeyword, true, "if"),
482.                 new RazorLexerToken(LexerTokenType.CodeParenthesesStart, true, "("),
483.                 new RazorLexerToken(LexerTokenType.Condition, true, "true"),
484.                 new RazorLexerToken(LexerTokenType.CodeParenthesesEnd, true, ")"),
485.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
486.                 new HtmlLexerToken(LexerTokenType.OpeningTag, true, "h1"),
487.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
488.                 new HtmlLexerToken(LexerTokenType.Text, true, "Header 1"),
489.                 new HtmlLexerToken(LexerTokenType.ClosingTag, true, "h1"),
490.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
491.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
492.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}")
493.             };
494.             var lexer = new RazorLexer(new StringReader(razor));
495.
496.             // Act
497.             lexer.Start();
498.
499.             // Assert
500.             Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
501.             Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
502.         }
503.
504.         [Fact]
505.         public void Parses_Razor_Code_Conditional_If_Else_With_Html()
506.         {
507.             // Arrange
508.             var razor =
509. @"@{
510.     if (true)
511.     {
512.         <h1>Header 1</h1>
513.     }
514.     else
515.     {
516.         <h2>Header 2</h2>
517.     }
518. }";
519.             var expectedTokens = new List<ILexerToken>()
520.             {
521.                 new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
522.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
523.                 new RazorLexerToken(LexerTokenType.ConditionalKeyword, true, "if"),
524.                 new RazorLexerToken(LexerTokenType.CodeParenthesesStart, true, "("),
525.                 new RazorLexerToken(LexerTokenType.Condition, true, "true"),
526.                 new RazorLexerToken(LexerTokenType.CodeParenthesesEnd, true, ")"),
527.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
528.                 new HtmlLexerToken(LexerTokenType.OpeningTag, true, "h1"),
529.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
530.                 new HtmlLexerToken(LexerTokenType.Text, true, "Header 1"),
531.                 new HtmlLexerToken(LexerTokenType.ClosingTag, true, "h1"),
532.                 new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
533.                 new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
534.                 new RazorLexerToken(LexerTokenType.ConditionalKeyword, true, "else"),
535.                 new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
536.                 new HtmlLexerToken(LexerTokenType.OpeningTag, true, "h2"),
```

```
                new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
                new HtmlLexerToken(LexerTokenType.Text, true, "Header 2"),
                new HtmlLexerToken(LexerTokenType.ClosingTag, true, "h2"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, true, ">"),
                new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
                new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}")
            };
            var lexer = new RazorLexer(new StringReader(razor));

            // Act
            lexer.Start();

            // Assert
            Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
            Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
        }

        [Fact]
        public void Parses_Razor_Viewdata()
        {
            // Arrange
            var razor =
@"<html>
    <body>
        @{
            ViewData[""Title""] = ""Page Title"";
        }
    </body>
</html>";
            var expectedTokens = new List<ILexerToken>()
            {
                new HtmlLexerToken(LexerTokenType.OpeningTag, false, "html"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.OpeningTag, false, "body"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new RazorLexerToken(LexerTokenType.RazorDirective, true, "@"),
                new RazorLexerToken(LexerTokenType.CodeBlockStart, true, "{"),
                new RazorLexerToken(LexerTokenType.Unknown, true, @"ViewData[""Title""]"),
                new RazorLexerToken(LexerTokenType.Assignment, true, "="),
                new RazorLexerToken(LexerTokenType.VariableData, true, @"""Page Title"""),
                new RazorLexerToken(LexerTokenType.Semicolon, true, ";"),
                new RazorLexerToken(LexerTokenType.CodeBlockEnd, true, "}"),
                new HtmlLexerToken(LexerTokenType.ClosingTag, false, "body"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
                new HtmlLexerToken(LexerTokenType.ClosingTag, false, "html"),
                new HtmlLexerToken(LexerTokenType.ClosingAngle, false, ">"),
            };
            var lexer = new RazorLexer(new StringReader(razor));

            // Act
            lexer.Start();

            // Assert
            Assert.Equal(expectedTokens.Count, lexer.Tokens.Count);
            Assert.True(expectedTokens.SequenceEqual(lexer.Tokens, new LexerTokenComparer()));
        }
    }
}
```

3 Priedas. Palyginimo algoritmas

RazorComparer.cs

```
  1. namespace TemplateDiff.Comparer
  2. {
  3.     using System.Text;
  4.     using TemplateDiff.Models.Lexer;
  5.     using TemplateDiff.Models.Lexer.Html;
  6.     using TemplateDiff.Models.Lexer.Razor;
  7.     using static TemplateDiff.Models.Lexer.LexerEnums;
  8.
  9.     public static class RazorComparer
 10.     {
 11.         public static Stack<LexerTokenType> _flowStack = [];
 12.         public static Stack<HtmlLexerToken> _htmlStack = [];
 13.
 14.         public static bool RazorEquals(ICollection<ILexerToken> pageTokens,
ICollection<ILexerToken> templateTokens)
 15.         {
 16.             if (templateTokens == null || pageTokens == null)
 17.             {
 18.                 return false;
 19.             }
 20.
 21.             int templateIndex = 0;
 22.             for (int i = 0; i < pageTokens.Count; i++) // page iterator
 23.             {
 24.                 if (pageTokens.ElementAt(i) is null)
 25.                 {
 26.                     throw new ArgumentNullException("HTML page token is null.");
 27.                 }
 28.
 29.                 if (pageTokens.ElementAt(i) is not HtmlLexerToken) // Lexer returned non-HTML
value from HTML page
 30.                 {
 31.                     throw new ArgumentException("Non-HTML value in received HTML page.");
 32.                 }
 33.
 34.                 int textIndex = 0;
 35.                 var htmlPageToken = pageTokens.ElementAt(i) as HtmlLexerToken;
 36.                 if (htmlPageToken!.Type == LexerTokenType.OpeningTag)
 37.                 {
 38.                     _htmlStack.Push(htmlPageToken);
 39.                 }
 40.                 else if (htmlPageToken.Type == LexerTokenType.ClosingTag)
 41.                 {
 42.                     if (_htmlStack.Peek().Content.Equals(htmlPageToken.Content)) // closing the
same token
 43.                     {
 44.                         _htmlStack.Pop();
 45.                     }
 46.                 }
 47.
 48.                 for (int j = templateIndex; j < templateTokens.Count; j++) // template iterator
 49.                 {
 50.                     if (templateTokens.ElementAt(j) is null)
 51.                     {
 52.                         throw new ArgumentNullException("RazorLexer returned null token.");
 53.                     }
 54.
 55.                     if (templateTokens.ElementAt(j) is HtmlLexerToken)
 56.                     {
 57.                         var htmlTemplateToken = templateTokens.ElementAt(j) as HtmlLexerToken;
 58.                         Console.WriteLine("Html token '{0}' with content '{1}' and attributes
'{2}'",
 59.                             htmlTemplateToken!.Type, htmlTemplateToken!.Content.ToString(),
String.Join(", ", htmlTemplateToken.Attributes ?? []));
 60.                         bool conditionEnd = false;
 61.
 62.                         if (_flowStack.Count > 0) // in some code flow logic
 63.                         {
 64.                             if (_flowStack.Peek() == LexerTokenType.CycleKeyword) // in a cycle
```

```
 65.                             {
 66.                                 ICollection<ILexerToken> cycleTokens = [];
 67.                                 while (templateTokens.ElementAt(templateIndex).Type !=
LexerTokenType.CodeBlockEnd) // template cycle iterator
 68.                                 {
 69.                                     if (templateIndex >= templateTokens.Count - 1)
 70.                                     {
 71.                                         break;
 72.                                     }
 73.
 74.                                     if (templateTokens.ElementAt(templateIndex) is HtmlLexerToken
htmlTemplateCycleToken) //
 75.                                     {
 76.                                         cycleTokens.Add(htmlTemplateCycleToken);
 77.                                     }
 78.                                     else if (templateTokens.ElementAt(templateIndex).Type ==
LexerTokenType.RazorDirective &&
 79.                                         templateTokens.ElementAt(templateIndex).NextToken.Type ==
LexerTokenType.VariableNameReference)
 80.                                     {
 81.                                         cycleTokens.Add(templateTokens.ElementAt(templateIndex));
// razor directive
 82.
cycleTokens.Add(templateTokens.ElementAt(++templateIndex)); // variable name reference
 83.                                     }
 84.
 85.                                     templateIndex++;
 86.                                     j++;
 87.                                 }
 88.
 89.                                 int pageTokensAppearedCount = 0;
 90.                                 bool breakHtmlCyle = false;
 91.                                 while (i < pageTokens.Count) // template cycle for page values
 92.                                 {
 93.                                     if (breakHtmlCyle)
 94.                                     {
 95.                                         break;
 96.                                     }
 97.
 98.                                     for (int cycleTokenIndex = 0; cycleTokenIndex <
cycleTokens.Count; cycleTokenIndex++)
 99.                                     {
100.                                         // now iteration happens on HTML side here
101.                                         htmlPageToken = pageTokens.ElementAt(i) as HtmlLexerToken;
102.                                         if (htmlPageToken.Type != LexerTokenType.Text)
103.                                         {
104.                                             if
(!htmlPageToken.ContentEquals(cycleTokens.ElementAt(cycleTokenIndex) as HtmlLexerToken))
105.                                             {
106.                                                 breakHtmlCyle = true;
107.                                                 break; // cycle possibly not run
108.                                             }
109.                                         }
110.                                         else if (CheckTextContent(pageTokens.ElementAt(i) as
HtmlLexerToken, cycleTokens, ref cycleTokenIndex))
111.                                         {
112.                                             cycleTokenIndex--; // CheckTextContent() increments,
when this inner for does also.
113.                                         }
114.                                         else
115.                                         {
116.                                             throw new NotImplementedException();
117.                                         }
118.
119.                                         i++;
120.                                     }
121.
122.                                     if (!breakHtmlCyle)
123.                                     {
124.                                         pageTokensAppearedCount++;
125.                                     }
126.                                 }
127.
```

```
128.                                 Console.WriteLine("Page token '{0}' appeared {1} time(s)",
string.Join(string.Empty, cycleTokens), pageTokensAppearedCount);
129.                                 continue;
130.                             }
131.                             else if (_flowStack.Peek() == LexerTokenType.ConditionalKeyword) // in
a conditional
132.                             {
133.                                 StringBuilder sb = new ();
134.                                 bool oneConditionalCorrect = false;
135.                                 int htmlPageConditionalTokenIndex = i;
136.                                 while (j < templateTokens.Count && i < pageTokens.Count) //
iterate until conditional content end
137.                                 {
138.                                     if (templateTokens.ElementAt(j) is RazorLexerToken)
139.                                     {
140.                                         if (templateTokens.ElementAt(j).Type ==
LexerTokenType.CodeBlockEnd) // conditional ended
141.                                         {
142.                                             j++; // move by one template token since conditional
ended
143.                                             if (templateTokens.ElementAt(j).Type ==
LexerTokenType.ConditionalKeyword) // conditional chain
144.                                             {
145.                                                 if (oneConditionalCorrect) // skip chain if one
was found correct
146.                                                 {
147.                                                     Console.WriteLine("The same HTML found,
skipping conditional to end.");
148.                                                     while (j < templateTokens.Count)
149.                                                     {
150.                                                         var templateTokenConditional =
templateTokens.ElementAt(j);
151.                                                         if (templateTokenConditional.Type ==
LexerTokenType.ConditionalKeyword)
152.                                                         {
153.                                                             j++;
154.                                                             continue;
155.                                                         }
156.
157.                                                         if (templateTokenConditional.Type ==
LexerTokenType.CodeBlockEnd && // conditional chain end
158.
templateTokenConditional.NextToken.Type != LexerTokenType.ConditionalKeyword)
159.                                                         {
160.                                                             _flowStack.Pop();
161.                                                             //i--; // bring cursor to CodeBlockEnd
since next iteration will do i++
162.                                                             j--;
163.                                                             conditionEnd = true;
164.                                                             break;
165.                                                         }
166.
167.                                                         j++;
168.                                                     }
169.                                                 }
170.
171.                                                 j++;
172.                                                 continue;
173.                                             }
174.                                             else // conditional end
175.                                             {
176.                                                 templateIndex = j;
177.                                                 _flowStack.Pop();
178.                                                 conditionEnd = true;
179.                                                 break;
180.                                             }
181.
182.                                             i--;
183.                                         }
184.                                     }
185.                                     else if (templateTokens.ElementAt(j) is HtmlLexerToken
templateHtmlConditionalToken)
186.                                     {
```

```
187.                                     if
(templateHtmlConditionalToken.ContentEquals(pageTokens.ElementAt(i) as HtmlLexerToken))
188.                                     {
189.                                         // current value in conditional is the same
190.                                         i++; // move by one HTML token
191.                                         oneConditionalCorrect = true;
192.                                         sb.Append(templateHtmlConditionalToken.Content);
193.                                     }
194.                                     else if (CheckTextContent(pageTokens.ElementAt(i) as
HtmlLexerToken, templateTokens, ref j))
195.                                     {
196.                                         j--;
197.                                         i++;
198.                                         oneConditionalCorrect = true;
199.                                         sb.Append(pageTokens.ElementAt(i).Content);
200.                                     }
201.                                     else // this conditional is not correct, try again
202.                                     {
203.                                         i = htmlPageConditionalTokenIndex;
204.                                         oneConditionalCorrect = false;
205.                                         sb.Clear();
206.                                     }
207.                                 }
208.
209.                                 j++;
210.                             }
211.
212.                             Console.WriteLine($"conditional value: {sb}");
213.                         }
214.
215.                         if (templateTokens.ElementAt(templateIndex).Type !=
LexerTokenType.CodeBlockEnd &&
216.                                     (_flowStack.Peek() == LexerTokenType.ConditionalKeyword ||
_flowStack.Peek() == LexerTokenType.CycleKeyword))
217.                         {
218.                             _flowStack.Pop();
219.                             templateIndex++;
220.                         }
221.                     }
222.
223.                     if (conditionEnd)
224.                     {
225.                         if (templateTokens.ElementAt(j).Type == LexerTokenType.CodeBlockEnd &&
226.                             _flowStack.TryPeek(out var f) && f ==
LexerTokenType.RazorDirective)
227.                         {
228.                             _flowStack.Pop();
229.                             templateIndex++; // skip code block end in template cycle
230.                         }
231.
232.                         if (_htmlStack.TryPeek(out _))
233.                         {
234.                             _htmlStack.Pop();
235.                         }
236.
237.                         i--; // backtrack to token after closing code block
238.                         break;
239.                     }
240.
241.                     if (htmlPageToken!.ContentEquals(htmlTemplateToken)) // lexer entries are
the same
242.                     {
243.                         templateIndex = j + 1;
244.                         break;
245.                     }
246.
247.                     if (htmlPageToken.Type == LexerTokenType.Text &&
CheckTextContent(htmlPageToken, templateTokens, ref templateIndex)) // text is fine
248.                     {
249.                         break;
250.                     }
251.
252.                     else if (htmlPageToken.Type == LexerTokenType.Text) //
```

```
253.                         {
254.                             if (htmlPageToken.Content.StartsWith(htmlTemplateToken.Content))
255.                             {
256.                                 templateIndex = j + 1;
257.                                 textIndex = htmlTemplateToken.Content.Length - 1; // text literal
index
258.                                 string remainingHtmlText =
htmlPageToken.Content.Substring(textIndex + 1);
259.                                 ILexerToken? nextTemplateToken = null;
260.                                 int tempTemplateIndex = templateIndex;
261.                                 for (int k = j + 1; k < templateTokens.Count; k++)
262.                                 {
263.                                     if (templateTokens.ElementAt(k).GetType() ==
typeof(HtmlLexerToken))
264.                                     {
265.                                         tempTemplateIndex += k - j;
266.                                         nextTemplateToken = templateTokens.ElementAt(k);
267.                                         break;
268.                                     }
269.                                 }
270.
271.                                 if (nextTemplateToken != null)
272.                                 {
273.                                     // Find the index of the next token in the HTML
274.                                     int nextTokenIndex =
remainingHtmlText.LastIndexOf(nextTemplateToken.Content);
275.                                     if (nextTokenIndex != -1)
276.                                     {
277.                                         // Skip the text between the current token and the next
token in the HTML
278.                                         textIndex += nextTokenIndex +
nextTemplateToken.Content.Length;
279.                                         j += tempTemplateIndex - templateIndex;
280.                                         templateIndex = tempTemplateIndex;
281.                                         Console.WriteLine("Literal text '{0}' matched with
template '{1}' and variable content was ignored", htmlPageToken.Content, htmlTemplateToken.Content);
282.                                         break;
283.                                     }
284.                                 }
285.                                 else
286.                                 {
287.                                     // If there are no more tokens in the template, skip the
remaining text in the HTML
288.                                     textIndex = htmlPageToken.Content.Length - 1;
289.                                     Console.WriteLine("Literal text '{0}' matched with template
'{1}' and variable content was ignored", htmlPageToken.Content, htmlTemplateToken.Content);
290.                                     break;
291.                                 }
292.                             }
293.                             else
294.                             {
295.                                 Console.Error.WriteLine("Literal text '{0}' did match with
template '{1}'", htmlPageToken.Content, htmlTemplateToken.Content);
296.                                 return false; // literal text did not match with template
297.                             }
298.                         }
299.                         else
300.                         {
301.                             return false;
302.                         }
303.                     }
304.                     else if (templateTokens.ElementAt(j) is RazorLexerToken)
305.                     {
306.                         var razorTemplateToken = templateTokens.ElementAt(j) as RazorLexerToken;
307.                         if (razorTemplateToken!.Type == LexerTokenType.RazorDirective)
308.                         {
309.                             if (templateTokens.ElementAt(j).NextToken is RazorLexerToken) //
lookahead
310.                             {
311.                                 var nextToken = templateTokens.ElementAt(j).NextToken as
RazorLexerToken;
312.                                 if (nextToken!.Type == LexerTokenType.VariableNameReference) //
razor variable only - anything goes
```

```
313.                                 {
314.                                     if (CheckTextContent(htmlPageToken!, templateTokens, ref
templateIndex))
315.                                     {
316.                                         Console.WriteLine("Razor variable tokens '{0}' and '{1}'
with content '{2}{3}'",
317.                                             razorTemplateToken!.Type, nextToken.Type,
razorTemplateToken!.Content, nextToken.Content);
318.                                         break;
319.                                     }
320.                                     else
321.                                     {
322.                                         return false;
323.                                     }
324.                                 }
325.                                 else if (nextToken!.Type == LexerTokenType.CodeBlockStart)
326.                                 {
327.                                     _flowStack.Push(LexerTokenType.RazorDirective);
328.                                 }
329.                             }
330.                             else
331.                             {
332.                                 throw new NotImplementedException();
333.                             }
334.                         }
335.                         else if (razorTemplateToken.Type == LexerTokenType.ConditionalKeyword)
336.                         {
337.                             _flowStack.Push(LexerTokenType.ConditionalKeyword);
338.                         }
339.                         else if (razorTemplateToken.Type == LexerTokenType.CycleKeyword)
340.                         {
341.                             _flowStack.Push(LexerTokenType.CycleKeyword);
342.                         }
343.                         else if (razorTemplateToken.Type == LexerTokenType.CodeBlockEnd)
344.                         {
345.                             if (_flowStack.TryPeek(out var s) &&
346.                                 (s == LexerTokenType.CycleKeyword ||
347.                                 s == LexerTokenType.ConditionalKeyword ||
348.                                 s == LexerTokenType.RazorDirective))
349.                             {
350.                                 _flowStack.Pop();
351.                             }
352.                         }
353.
354.                         Console.WriteLine("Razor token '{0}' with content '{1}'",
razorTemplateToken!.Type, razorTemplateToken!.Content.ToString());
355.                         templateIndex++;
356.                     }
357.                     else
358.                     {
359.                         throw new ArgumentException("RazorLexer returned non HTML or Razor
token");
360.                     }
361.                 }
362.
363.                 Console.WriteLine("Page and Template token are the same: {0}\n",
htmlPageToken!.Content);
364.             }
365.
366.             return true;
367.         }
368.
369.         public static bool CheckTextContent(this HtmlLexerToken pageTextToken,
ICollection<ILexerToken> templateTokens, ref int templateIndex)
370.         {
371.             if (pageTextToken.Type != LexerTokenType.Text)
372.             {
373.                 return false;
374.             }
375.
376.             var txt = pageTextToken.Content;
377.             while (pageTextToken!.NextToken!.Type != templateTokens.ElementAt(templateIndex).Type)
378.             {
```

```
379.                 if (templateIndex >= templateTokens.Count)
380.                 {
381.                     return false;
382.                 }
383.
384.                 var templateToken = templateTokens.ElementAt(templateIndex);
385.
386.                 if (templateToken.Type == LexerTokenType.Text) // actual text comparison
387.                 {
388.                     var textPart = txt.IndexOf(templateToken.Content) +
templateToken.Content.Length;
389.                     int txtDiff = txt.Length - textPart;
390.                     if (txtDiff < 0) // text is definitely incorrect
391.                     {
392.                         return false;
393.                     }
394.
395.                     txt = txt.Substring(textPart, txtDiff);
396.                     if (pageTextToken.Content.Length == textPart) // reached end of HTML text
397.                     {
398.                         templateIndex++;
399.                         return true;
400.                     }
401.
402.                     templateIndex++;
403.                 }
404.                 else if (templateToken.Type == LexerTokenType.RazorDirective) // Razor @
405.                 {
406.                     templateToken = templateTokens.ElementAt(++templateIndex);
407.                     if (templateToken.Type == LexerTokenType.VariableNameReference) // variable
reference
408.                     {
409.                         templateIndex++;
410.                         continue;
411.                     }
412.                     // TODO Razor Statement Syntax (not inside @{} but directly @if(){} @while(){}
etc.
413.
414.                     templateIndex++;
415.                 }
416.                 else if (templateToken is RazorLexerToken) // other relevant syntax
417.                 {
418.                     templateIndex++;
419.                 }
420.             }
421.
422.             return true;
423.         }
424.     }
425. }
```

4 Priedas. Palyginimo algoritmo vienetiniai testai

ComparerTests.cs

```
1. namespace UnitTests.Comparer
2. {
3.     using TemplateDiff.Comparer;
4.     using TemplateDiff.Lexer;
5.
6.     public class ComparerTests
7.     {
8.         [Fact]
9.         public void Comparer_Html_Tag_Equals()
10.        {
11.            // Arrange
12.            var html = new StringReader(@"<html></html>");
13.            var lexer = new RazorLexer(html);
14.
15.            // Act
16.            lexer.Start();
17.            var equals = RazorComparer.RazorEquals(lexer.Tokens, lexer.Tokens);
18.
19.            // Assert
20.            Assert.True(equals);
21.        }
22.
23.        [Fact]
24.        public void Comparer_Html_Tag_With_Attribute_Equals()
25.        {
26.            // Arrange
27.            var html = new StringReader(@"<h1 class=""header""></h1>");
28.            var lexer = new RazorLexer(html);
29.
30.            // Act
31.            lexer.Start();
32.            var equals = RazorComparer.RazorEquals(lexer.Tokens, lexer.Tokens);
33.
34.            // Assert
35.            Assert.True(equals);
36.        }
37.
38.        [Fact]
39.        public void Comparer_Html_Tag_With_Content_Equals()
40.        {
41.            // Arrange
42.            var html = new StringReader(@"<h1>Header 1</h1>");
43.            var lexer = new RazorLexer(html);
44.
45.            // Act
46.            lexer.Start();
47.            var equals = RazorComparer.RazorEquals(lexer.Tokens, lexer.Tokens);
48.
49.            // Assert
50.            Assert.True(equals);
51.        }
52.
53.        [Fact]
54.        public void Comparer_Html_Page_Equals()
55.        {
56.            // Arrange
57.            var html = new StringReader(
58. @"<html>
59.     <head>
60.         <title>Page title</title>
61.     </head>
62.     <body>
63.         <h1>Header 1</h1>
64.     </body>
65. </html>");
66.            var lexer = new RazorLexer(html);
67.
68.            // Act
69.            lexer.Start();
```

```
70.             var equals = RazorComparer.RazorEquals(lexer.Tokens, lexer.Tokens);
71.
72.             // Assert
73.             Assert.True(equals);
74.         }
75.
76.         [Fact]
77.         public void Comparer_Page_With_Razor_Variable_Only_Equals()
78.         {
79.             // Arrange
80.             var html = new StringReader(
81. @"<html>
82.     <body>
83.         Hello world!
84.     </body>
85. </html>");
86.             var cshtml = new StringReader(
87. @"@{
88.     var hw = ""Hello world!"";
89. }
90. <html>
91.     <body>
92.         @hw
93.     </body>
94. </html>");
95.
96.             var htmlLexer = new RazorLexer(html);
97.             var cshtmlLexer = new RazorLexer(cshtml);
98.
99.             // Act
100.            htmlLexer.Start();
101.            cshtmlLexer.Start();
102.            var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
103.
104.            // Assert
105.            Assert.True(equals);
106.        }
107.
108.        [Fact]
109.        public void Comparer_Page_With_Razor_Variables_And_Conditional_Equals()
110.        {
111.            // Arrange
112.            var html = new StringReader(
113. @"<html><head></head><body>
114. <h>Lorem ipsum dolor sit amet</h>
115. <p>0,9586050737644802</p>
116. <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.0,13710348352414958</p>
117. </body></html>");
118.            var cshtml = new StringReader(
119. @"<html><head></head><body>
120. <h>Lorem ipsum dolor sit amet</h>
121. <p>@variableReference</p>
122. @{
123.     if(true)
124.     {
125.         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.@variableReference</p>
126.     }
127. }
128. </body></html>");
129.
130.            var htmlLexer = new RazorLexer(html);
131.            var cshtmlLexer = new RazorLexer(cshtml);
132.
133.            // Act
134.            htmlLexer.Start();
135.            cshtmlLexer.Start();
136.            var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
137.
138.            // Assert
139.            Assert.True(equals);
140.        }
141.
142.        [Fact]
```

```
143.         public void Comparer_Page_In_Razor_Codeblock_Equals()
144.         {
145.             // Arrange
146.             var html = new StringReader(@"<h1>Hi</h1>");
147.             var cshtml = new StringReader(@"@{<h1>Hi</h1>}");
148. 
149.             var htmlLexer = new RazorLexer(html);
150.             var cshtmlLexer = new RazorLexer(cshtml);
151. 
152.             // Act
153.             htmlLexer.Start();
154.             cshtmlLexer.Start();
155.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
156. 
157.             // Assert
158.             Assert.True(equals);
159.         }
160. 
161.         [Fact]
162.         public void Comparer_Page_Cycle_With_Tag_Equals()
163.         {
164.             // Arrange
165.             var html = new StringReader(
166. @"<p>1</p>
167. <p>2</p>
168. <p>3</p>");
169.             var cshtml = new StringReader(
170. @"@{
171.     for(int i = 0; i < 3; i++)
172.     {
173.         <p>@i</p>
174.     }
175. }");
176. 
177.             var htmlLexer = new RazorLexer(html);
178.             var cshtmlLexer = new RazorLexer(cshtml);
179. 
180.             // Act
181.             htmlLexer.Start();
182.             cshtmlLexer.Start();
183.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
184. 
185.             // Assert
186.             Assert.True(equals);
187.         }
188. 
189.         [Fact]
190.         public void Comparer_Page_Cycle_With_Tags_Equals()
191.         {
192.             // Arrange
193.             var html = new StringReader(
194. @"<p>1</p><br/>
195. <p>2</p><br/>
196. <p>3</p><br/>");
197.             var cshtml = new StringReader(
198. @"@{
199.     for(int i = 0; i < 3; i++)
200.     {
201.         <p>@i</p><br/>
202.     }
203. }");
204. 
205.             var htmlLexer = new RazorLexer(html);
206.             var cshtmlLexer = new RazorLexer(cshtml);
207. 
208.             // Act
209.             htmlLexer.Start();
210.             cshtmlLexer.Start();
211.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
212. 
213.             // Assert
214.             Assert.True(equals);
215.         }
```

```
216.
217.         [Fact]
218.         public void Comparer_Page_Cycle_Not_Run_With_Tags_Equals()
219.         {
220.             // Arrange
221.             var html = new StringReader(@"<h1>Hello</h1><p>End</p>");
222.             var cshtml = new StringReader(
223. @"<h1>Hello</h1>
224. @{
225.     while(false)
226.     {
227.         <span>I'm never!</span>
228.     }
229. }
230. <p>End</p>");
231.
232.             var htmlLexer = new RazorLexer(html);
233.             var cshtmlLexer = new RazorLexer(cshtml);
234.
235.             // Act
236.             htmlLexer.Start();
237.             cshtmlLexer.Start();
238.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
239.
240.             // Assert
241.             Assert.True(equals);
242.         }
243.
244.         [Fact]
245.         public void Comparer_Page_Conditional_If_With_Tags_Equals()
246.         {
247.             // Arrange
248.             var html = new StringReader(@"<p>Labas</p>");
249.             var cshtml = new StringReader(
250. @"@{
251.     if(true)
252.     {
253.         <p>Labas</p>
254.     }
255. }");
256.
257.             var htmlLexer = new RazorLexer(html);
258.             var cshtmlLexer = new RazorLexer(cshtml);
259.
260.             // Act
261.             htmlLexer.Start();
262.             cshtmlLexer.Start();
263.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
264.
265.             // Assert
266.             Assert.True(equals);
267.         }
268.
269.         [Fact]
270.         public void Comparer_Page_Conditional_If_Else_With_Tags_True_Equals()
271.         {
272.             // Arrange
273.             var html = new StringReader(
274. @"<p>Labas</p>");
275.             var cshtml = new StringReader(
276. @"@{
277.     if(true)
278.     {
279.         <p>Labas</p>
280.     }
281.     else
282.     {
283.         <p>Ate</p>
284.     }
285. }");
286.
287.             var htmlLexer = new RazorLexer(html);
288.             var cshtmlLexer = new RazorLexer(cshtml);
```

```
289.
290.             // Act
291.             htmlLexer.Start();
292.             cshtmlLexer.Start();
293.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
294.
295.             // Assert
296.             Assert.True(equals);
297.         }
298.
299.         [Fact]
300.         public void Comparer_Page_Conditional_If_Else_With_Tags_Attributes_True_Equals()
301.         {
302.             // Arrange
303.             var html = new StringReader(
304. @"<a href=""https://ktu.edu"">KTU</a>");
305.             var cshtml = new StringReader(
306. @"@{
307.     if (true){
308.         <a href=""https://ktu.edu"">KTU</a>
309.     }
310.     else{
311.         <span>:(</span>
312.     }
313. }");
314.
315.             var htmlLexer = new RazorLexer(html);
316.             var cshtmlLexer = new RazorLexer(cshtml);
317.
318.             // Act
319.             htmlLexer.Start();
320.             cshtmlLexer.Start();
321.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
322.
323.             // Assert
324.             Assert.True(equals);
325.         }
326.
327.         [Fact]
328.         public void Comparer_Page_Conditional_If_Else_With_Tag_False_Equals()
329.         {
330.             // Arrange
331.             var html = new StringReader(
332. @"<p>Ate</p>");
333.             var cshtml = new StringReader(
334. @"@{
335.     if(false)
336.     {
337.         <p>Labas</p>
338.     }
339.     else
340.     {
341.         <p>Ate</p>
342.     }
343. }");
344.
345.             var htmlLexer = new RazorLexer(html);
346.             var cshtmlLexer = new RazorLexer(cshtml);
347.
348.             // Act
349.             htmlLexer.Start();
350.             cshtmlLexer.Start();
351.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
352.
353.             // Assert
354.             Assert.True(equals);
355.         }
356.
357.         [Fact]
358.         public void Comparer_Page_Conditional_If_Else_With_Tag_With_Html_Above_Equals()
359.         {
360.             // Arrange
361.             var html = new StringReader(
```

```
@"<h1>Sveiki</h1>
<p>Labas</p>");
            var cshtml = new StringReader(
@"<h1>Sveiki</h1>
@{
    if(true)
    {
        <p>Labas</p>
    }
    else
    {
        <p>Ate</p>
    }
}");

            var htmlLexer = new RazorLexer(html);
            var cshtmlLexer = new RazorLexer(cshtml);

            // Act
            htmlLexer.Start();
            cshtmlLexer.Start();
            var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);

            // Assert
            Assert.True(equals);
        }

        [Fact]
        public void Comparer_Page_Conditional_If_Else_With_Tag_With_Html_Below_Equals()
        {
            // Arrange
            var html = new StringReader(
@"<p>Labas</p>
<h1>Sveiki</h1>");
            var cshtml = new StringReader(
@"@{
    if(true)
    {
        <p>Labas</p>
    }
    else
    {
        <p>Ate</p>
    }
}
<h1>Sveiki</h1>");

            var htmlLexer = new RazorLexer(html);
            var cshtmlLexer = new RazorLexer(cshtml);

            // Act
            htmlLexer.Start();
            cshtmlLexer.Start();
            var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);

            // Assert
            Assert.True(equals);
        }

        [Fact]
        public void Comparer_Page_Conditional_If_With_Tag_With_Html_Equals()
        {
            // Arrange
            var html = new StringReader(
@"<html><head></head><body>
<h>Lorem ipsum dolor sit amet</h><p></p>
<p></p>
</body></html>");
            var cshtml = new StringReader(
@"<html><head></head><body>
<h>Lorem ipsum dolor sit amet</h><p></p>
@{
    if(true)
```

```
435.     {<p></p>}
436. }</body></html>");
437. 
438.             var htmlLexer = new RazorLexer(html);
439.             var cshtmlLexer = new RazorLexer(cshtml);
440. 
441.             // Act
442.             htmlLexer.Start();
443.             cshtmlLexer.Start();
444.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
445. 
446.             // Assert
447.             Assert.True(equals);
448.         }
449. 
450.         [Fact]
451.         public void Comparer_Page_Conditional_If_Else_With_Tag_And_Variable_False_Equals()
452.         {
453.             // Arrange
454.             var html = new StringReader(@"<p>Ate, Lukas!</p>");
455.             var cshtml = new StringReader(
456. @"@{
457.     var name = ""Lukas"";
458.     if(false)
459.     {
460.         <p>Labas, @name</p>
461.     }
462.     else
463.     {
464.         <p>Ate, @name!</p>
465.     }
466. }");
467. 
468.             var htmlLexer = new RazorLexer(html);
469.             var cshtmlLexer = new RazorLexer(cshtml);
470. 
471.             // Act
472.             htmlLexer.Start();
473.             cshtmlLexer.Start();
474.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
475. 
476.             // Assert
477.             Assert.True(equals);
478.         }
479. 
480.         [Fact]
481.         public void Comparer_Page_Conditional_Multi_If_Else_With_Tag_Equals()
482.         {
483.             // Arrange
484.             var html = new StringReader(@"<p>Ate</p>");
485.             var cshtml = new StringReader(
486. @"@{
487.     if(false)
488.     {
489.         <p>Labas</p>
490.     }
491.     else if (true)
492.     {
493.         <p>Ate</p>
494.     }
495.     else
496.     {
497.         <b>Something went wrong</b>
498.     }
499. }");
500. 
501.             var htmlLexer = new RazorLexer(html);
502.             var cshtmlLexer = new RazorLexer(cshtml);
503. 
504.             // Act
505.             htmlLexer.Start();
506.             cshtmlLexer.Start();
507.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
```

```
508.
509.             // Assert
510.             Assert.True(equals);
511.         }
512.
513.         [Fact]
514.         public void Comparer_Page_With_Tags_And_Cycle_With_Tags_Equals()
515.         {
516.             // Arrange
517.             var html = new StringReader(
518. @"<h1>Header</h1>
519. <p>1</p><br/>
520. <p>2</p><br/>
521. <p>3</p><br/>
522. <footer>End</footer>");
523.             var cshtml = new StringReader(
524. @"<h1>Header</h1>
525. @{
526.     for(int i = 0; i < 3; i++)
527.     {
528.         <p>@i</p><br/>
529.     }
530. }
531. <footer>End</footer>");
532.
533.             var htmlLexer = new RazorLexer(html);
534.             var cshtmlLexer = new RazorLexer(cshtml);
535.
536.             // Act
537.             htmlLexer.Start();
538.             cshtmlLexer.Start();
539.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
540.
541.             // Assert
542.             Assert.True(equals);
543.         }
544.     }
545. }
```

CheckTextTests.cs

```
 1. namespace UnitTests.Comparer
 2. {
 3.     using TemplateDiff.Comparer;
 4.     using TemplateDiff.Lexer;
 5.     using TemplateDiff.Models.Lexer.Html;
 6.
 7.     public class CheckTextTests
 8.     {
 9.         [Fact]
10.         public void Comparer_CheckText_PlainText_Equals()
11.         {
12.             // Arrange
13.             var html = new StringReader(@"<p>Hello, Lukas!</p>");
14.             int templateIndex = 2;
15.             var htmlLexer = new RazorLexer(html);
16.
17.             // Act
18.             htmlLexer.Start();
19.             var equals = RazorComparer.CheckTextContent(htmlLexer.Tokens.ElementAt(2) as
HtmlLexerToken, htmlLexer.Tokens, ref templateIndex);
20.
21.             // Assert
22.             Assert.Equal(3, templateIndex);
23.             Assert.True(equals);
24.         }
25.
26.         [Fact]
27.         public void Comparer_CheckText_Variable_In_Beginning_Equals()
28.         {
29.             // Arrange
30.             var html = new StringReader(@"<h1>Hello, Lukas!</h1>");
```

```
 31.             var cshtml = new StringReader(@"@{var header = ""Hello"";}<h1>@greeting,
Lukas!</h1>");
 32.             int templateIndex = 10;
 33.             var htmlLexer = new RazorLexer(html);
 34.             var cshtmlLexer = new RazorLexer(cshtml);
 35.
 36.             // Act
 37.             htmlLexer.Start();
 38.             cshtmlLexer.Start();
 39.             var cmp = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
 40.             var equals = RazorComparer.CheckTextContent(htmlLexer.Tokens.ElementAt(2) as
HtmlLexerToken, cshtmlLexer.Tokens, ref templateIndex);
 41.
 42.             // Assert
 43.             Assert.Equal(13, templateIndex);
 44.             Assert.True(equals);
 45.             Assert.True(cmp);
 46.         }
 47.
 48.         [Fact]
 49.         public void Comparer_CheckText_Variable_In_End_Equals()
 50.         {
 51.             // Arrange
 52.             var html = new StringReader(@"<h1>Lukas, hello!</h1>");
 53.             //                              01 2    3    4     5     678 9  10   11   12   13 14
 54.             var cshtml = new StringReader(@"@{var header = ""hello!"";}<h1>Lukas,
@greeting</h1>");
 55.             int templateIndex = 10;
 56.             var htmlLexer = new RazorLexer(html);
 57.             var cshtmlLexer = new RazorLexer(cshtml);
 58.
 59.             // Act
 60.             htmlLexer.Start();
 61.             cshtmlLexer.Start();
 62.             var equals = RazorComparer.CheckTextContent(htmlLexer.Tokens.ElementAt(2) as
HtmlLexerToken, cshtmlLexer.Tokens, ref templateIndex);
 63.
 64.             // Assert
 65.             Assert.Equal(13, templateIndex);
 66.             Assert.True(equals);
 67.         }
 68.
 69.         [Fact]
 70.         public void Comparer_CheckText_Variable_In_Beginning_And_End_Equals()
 71.         {
 72.             // Arrange
 73.             var html = new StringReader(@"<h1>Lukas, hello!</h1>");
 74.             //
1516 1718  19   20
 75.             var cshtml = new StringReader(@"@{var name = ""Lukas"";var header =
""hello!"";}<h1>@name, @greeting</h1>");
 76.             int templateIndex = 15;
 77.             var htmlLexer = new RazorLexer(html);
 78.             var cshtmlLexer = new RazorLexer(cshtml);
 79.
 80.             // Act
 81.             htmlLexer.Start();
 82.             cshtmlLexer.Start();
 83.             var equals = RazorComparer.CheckTextContent(htmlLexer.Tokens.ElementAt(2) as
HtmlLexerToken, cshtmlLexer.Tokens, ref templateIndex);
 84.
 85.             // Assert
 86.             Assert.Equal(20, templateIndex);
 87.             Assert.True(equals);
 88.         }
 89.
 90.         [Fact]
 91.         public void Comparer_CheckText_Variable_In_Middle_Equals()
 92.         {
 93.             // Arrange
 94.             var html = new StringReader(@"<p>Hello, world and Lukas!</p>");
 95.             //                              01 2    3  4     5    678 9   10  11 12    13     14
15
```

```
 96.             var cshtml = new StringReader(@"@{var wStr = ""world"";}<p>Hello, @wStr and
Lukas!</p>");
 97.
 98.             int templateIndex = 10;
 99.
100.             var htmlLexer = new RazorLexer(html);
101.             var cshtmlLexer = new RazorLexer(cshtml);
102.
103.             // Act
104.             htmlLexer.Start();
105.             cshtmlLexer.Start();
106.             var equals = RazorComparer.CheckTextContent(htmlLexer.Tokens.ElementAt(2) as
HtmlLexerToken, cshtmlLexer.Tokens, ref templateIndex);
107.
108.             // Assert
109.             Assert.Equal(14, templateIndex);
110.             Assert.True(equals);
111.         }
112.
113.         [Fact]
114.         public void Comparer_CheckText_Variable_Multiple_In_Middle_Equals()
115.         {
116.             // Arrange
117.             var html = new StringReader(
118. @"<html>
119.     <body>
120.         <p>Hello Lukas, have a nice night!</p>
121.     </body>
122. </html>");
123.             var cshtml = new StringReader(
124. @"@{
125.     var name = ""Lukas"";
126.     var what = ""nice"";
127. }
128. <html>
129.     <body>
130.         <p>Hello @name, have a @what night!</p>
131.     </body>
132. </html>");
133.
134.             var htmlLexer = new RazorLexer(html);
135.             var cshtmlLexer = new RazorLexer(cshtml);
136.
137.             // Act
138.             htmlLexer.Start();
139.             cshtmlLexer.Start();
140.             var equals = RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens);
141.
142.             // Assert
143.             Assert.True(equals);
144.         }
145.     }
146. }
```

5 Priedas. Pagrindinės programos kodas

Program.cs

```
#if !DEBUG
using BenchmarkDotNet.Running;
#endif
using Microsoft.CodeAnalysis;
using Microsoft.CodeAnalysis.CSharp;
using Microsoft.Extensions.Configuration;
using TemplateDiff;
#if !DEBUG
using TemplateDiff.Benchmarks;
#endif
using TemplateDiff.Comparer;
using TemplateDiff.Lexer;
using TemplateDiff.Models.Lexer;
public static class Program
{
    private static async Task Main(string[] args)
    {
        if (args.Contains("--benchmark"))
        {
#if DEBUG
            Console.WriteLine("Benchmark must be run in Release build.");
            return;
#else
            RunBenchmarks();
            return;
#endif
        }

        IConfiguration cmdArgs = new ConfigurationBuilder()
            .AddCommandLine(args)
            .Build();

        if (!CheckArgs(cmdArgs, out string url, out string layout, out string template))
        {
            return;
        }

        var html = await Page.GetPage(url);

        Console.WriteLine("=========================");
        Console.WriteLine($"Received HTML from {url}");
        Console.WriteLine(html);
        Console.WriteLine("-------------------------");

        ICollection<ILexerToken> htmlTokens;
        ICollection<ILexerToken> templateTokens;

        using (var htmlSr = new StringReader(html))
        {
            using var fileStreamlayout = new StreamReader(layout, new FileStreamOptions()
            {
                Mode = FileMode.Open,
                Access = FileAccess.Read,
                Share = FileShare.Read,
                Options = FileOptions.SequentialScan
            });

            using var fileStreamTemplate = new StreamReader(template, new FileStreamOptions()
            {
                Mode = FileMode.Open,
                Access = FileAccess.Read,
                Share = FileShare.Read,
                Options = FileOptions.SequentialScan
            });

            var layoutStr = await fileStreamlayout.ReadToEndAsync();
            var templateStr = await fileStreamTemplate.ReadToEndAsync();

            Console.WriteLine($"Received template layout from {layout}");
```

```
70.             Console.WriteLine(layoutStr);
71.             Console.WriteLine("-------------------------");
72.             Console.WriteLine($"Received template from {template}");
73.             Console.WriteLine(templateStr);
74.
75.             layoutStr = Page.InsertToRenderBody(layoutStr, templateStr);
76.             templateStr = string.Empty; // free up memory
77.             using var templateSr = new StringReader(layoutStr);
78.
79.             var htmlLexer = new RazorLexer(htmlSr);
80.             var templateLexer = new RazorLexer(templateSr);
81.
82.             // TODO multithreadding. Performance check. Use Task.Run and Task.WhenAll
83.             Console.WriteLine("========================");
84.             Console.WriteLine("performing lexical tokenisation");
85.             htmlLexer.Start();
86.             htmlTokens = htmlLexer.Tokens;
87.
88.             templateLexer.Start();
89.             templateTokens = templateLexer.Tokens;
90.         }
91.
92.         Console.WriteLine("========================");
93.         Console.WriteLine("comparing");
94.         bool compareResult = RazorComparer.RazorEquals(htmlTokens, templateTokens);
95.
96.         Console.WriteLine("========================");
97.         Console.WriteLine("result");
98.         Console.WriteLine(compareResult);
99.
100.     }
101.
102.     private static bool CheckArgs(IConfiguration cmdArgs, out string url, out string layout, out
string template)
103.     {
104.
105.         url = cmdArgs["url"]!;
106.         layout = cmdArgs["layout"]!;
107.         template = cmdArgs["template"]!;
108.
109.         if (string.IsNullOrEmpty(url))
110.         {
111.             ConsoleOutput.NoUrl();
112.             return false;
113.         }
114.
115.         if (string.IsNullOrEmpty(layout))
116.         {
117.             ConsoleOutput.NoLayout();
118.             return false;
119.         }
120.
121.         if (string.IsNullOrEmpty(template))
122.         {
123.             ConsoleOutput.NoTemplate();
124.             return false;
125.         }
126.
127.         return true;
128.     }
129.
130.     private static void Print(this IEnumerable<SyntaxNode> nodes)
131.     {
132.         foreach (var item in nodes)
133.         {
134.             Console.WriteLine("Fullstring: {0} and kind: {1}", item.ToFullString(), item.Kind());
135.             if (item.ChildNodes().Any())
136.             {
137.                 Print(item.ChildNodes());
138.             }
139.         }
140.     }
141. #if !DEBUG
```

```
    private static void RunBenchmarks()
    {
        Console.WriteLine("Starting lexer benchmark...");
        BenchmarkRunner.Run<LexerBenchmark>();

        Console.WriteLine("Starting comparer benchmark...");
        BenchmarkRunner.Run<ComparerBenchmark>();

        Console.WriteLine("Starting lexer benchmark scaling...");
        BenchmarkRunner.Run<LexerBenchmarkScaling>();

        Console.WriteLine("Starting comparer benchmark scaling...");
        BenchmarkRunner.Run<ComparerBenchmarkScaling>();
    }
#endif
}

```

Page.cs

```
namespace TemplateDiff
{
    using System;
    using System.Threading.Tasks;

    public static class Page
    {
        /// <summary>
        /// TODO: auto retry
        /// </summary>
        /// <param name="url"></param>
        /// <returns>Response content.</returns>
        public static async Task<string> GetPage(string url)
        {
            try
            {
                Console.WriteLine($"Calling {url}");
                HttpClient httpClient = new HttpClient();
                using HttpResponseMessage response = await httpClient.GetAsync(url);

                if (response.IsSuccessStatusCode)
                {
                    return await response.Content.ReadAsStringAsync();
                }
                else
                {
                    //TODO auto retry
                    throw new ArgumentNullException(string.Format("Failed to retrie HTML: {0}",
response.StatusCode));
                }
            }
            catch (Exception ex)
            {
                //TODO auto retry
                Console.WriteLine("Error retrieving HTML: {0}", ex.Message);
                throw;
            }
        }

        /// <summary>
        /// Replaces `@RenderBody()` in a _Layout.cshtml or similar files with the template
content.
        /// </summary>
        /// <param name="layout">The _Layout.cshtml file.</param>
        /// <param name="template">Any .cshtml template file.</param>
        /// <returns></returns>
        public static string InsertToRenderBody(this string layout, string template)
        {
            // TODO handle if there is no '@RenderBody()' in layout.
            return layout.Replace("@RenderBody()", template);
        }
    }
}
```

ConsoleOutput.cs

```
 1. namespace TemplateDiff
 2. {
 3.     using System;
 4.     using System.Collections.Generic;
 5.     using System.Linq;
 6.     using System.Text;
 7.     using System.Threading.Tasks;
 8.
 9.     internal class ConsoleOutput
10.     {
11.         public static void Help() => Console.WriteLine("For help use --help.");
12.
13.         public static void ShowHelp()
14.         {
15.             Console.WriteLine("--url <link> \t\tFor add a url.");
16.             Console.WriteLine("--layout <path> \t\tTo add a template layout.");
17.             Console.WriteLine("--template <path> \t\tTo add a template.");
18.         }
19.
20.         public static void NoUrl()
21.         {
22.             Console.WriteLine("No provided URL.");
23.             ShowHelp();
24.         }
25.         public static void BadUrl()
26.         {
27.             Console.WriteLine("Provided URL is bad.");
28.             ShowHelp();
29.         }
30.
31.         public static void NoLayout()
32.         {
33.             Console.WriteLine("No layout provided.");
34.             ShowHelp();
35.         }
36.
37.         public static void NoTemplate()
38.         {
39.             Console.WriteLine("No template provided.");
40.             ShowHelp();
41.         }
42.     }
43. }
```

6 Priedas. Sintetiniai testai

Program.cs

```
using System.Text;
using TemplateDiff.Comparer;
using TemplateDiff.Lexer;

internal class Program
{
    const byte TestCount = 100;
    static byte CorrectTest;
    static byte IncorrectTest;
    static double TruePositive = 0;
    static double TrueNegative = 0;

    const string htmlStart = @"<html><head></head><body>";
    const string htmlEnd = @"</body></html>";

    private static void Main(string[] args)
    {

        Random rnd = new Random();
        StringBuilder html = new();
        StringBuilder cshtml = new();

        Test_Correct(html, cshtml, rnd);
        TruePositive = CorrectTest / TestCount;
        CorrectTest = 0;
        IncorrectTest = 0;

        Test_Incorrect(html, cshtml, rnd);
        TrueNegative = IncorrectTest / TestCount;

        Console.WriteLine($"Total tests: {TestCount * 2}");

        Console.WriteLine($"True Positive: {TruePositive * 100} %");
        Console.WriteLine($"True Negative: {TrueNegative * 100} %");
    }

    private static void Test_Correct(StringBuilder html, StringBuilder cshtml, Random rnd)
    {
        for (int i = 0; i < TestCount; i++)
        {
            html.Append(htmlStart);
            cshtml.Append(htmlStart);

            Add_Html_Heading(html, cshtml, rnd);
            Add_Html_Paragraph(html, cshtml, rnd);
            Add_Logical_Condition(html, cshtml, rnd);
            Add_Cycle(html, cshtml, rnd);

            html.Append(htmlEnd);
            cshtml.Append(htmlEnd);

            Perform_Check(html.ToString(), cshtml.ToString());

            html.Clear();
            cshtml.Clear();
        }
    }

    private static void Test_Incorrect(StringBuilder html, StringBuilder cshtml, Random rnd)
    {
        for (int i = 0; i < TestCount; i++)
        {
            html.Append(htmlStart);
            cshtml.Append(htmlStart);

            Add_Html_Heading(html, cshtml, rnd);
            Add_Html_Paragraph(html, cshtml, rnd);
            Add_Logical_Condition(html, cshtml, rnd);
            Add_Cycle(html, cshtml, rnd);
```

```
70.
71.             html.Append(htmlEnd);
72.             cshtml.Append(htmlEnd);
73.
74.             html.Clear();
75.             html.Append(
76. @"<html>
77.     <head></head>
78.     <body>
79.         <h1>WEBSITE HAS BEEN HACKED</h1>
80.     </body>
81. </html>");
82.
83.             Perform_Check(html.ToString(), cshtml.ToString());
84.
85.             html.Clear();
86.             cshtml.Clear();
87.         }
88.     }
89.
90.     private static void Perform_Check(string html, string cshtml)
91.     {
92.         RazorLexer htmlLexer = new RazorLexer(new StringReader(html));
93.         htmlLexer.Start();
94.         RazorLexer cshtmlLexer = new RazorLexer(new StringReader(cshtml));
95.         cshtmlLexer.Start();
96.
97.         if (RazorComparer.RazorEquals(htmlLexer.Tokens, cshtmlLexer.Tokens))
98.         {
99.             CorrectTest++;
100.        }
101.        else
102.        {
103.            IncorrectTest++;
104.        }
105.    }
106.
107.    private static void Add_Html_Heading(StringBuilder sbHtml, StringBuilder sbCshtml, Random rnd)
108.    {
109.        while (rnd.NextDouble() >= 0.5)
110.        {
111.            string htmlP = @$"<h>Lorem ipsum dolor sit amet</h>";
112.
113.            sbHtml.Append(htmlP);
114.            sbCshtml.Append(htmlP);
115.        }
116.    }
117.
118.    private static void Add_Html_Paragraph(StringBuilder sbHtml, StringBuilder sbCshtml, Random
rnd)
119.    {
120.        sbHtml.Append("<p>");
121.        sbCshtml.Append("<p>");
122.
123.        bool variable = false;
124.        if (rnd.NextDouble() >= 0.5)
125.        {
126.            variable = true;
127.            sbHtml.Append(rnd.NextDouble());
128.            sbCshtml.Append("@variableReference");
129.        }
130.
131.        bool txt = false;
132.        while (rnd.NextDouble() >= 0.5)
133.        {
134.            txt = true;
135.            if (variable)
136.            {
137.                sbHtml.Append(' ');
138.                sbCshtml.Append(' ');
139.            }
140.
141.            sbHtml.Append("Lorem ipsum dolor sit amet, consectetur adipiscing elit.");
```

```
142.             sbCshtml.Append("Lorem ipsum dolor sit amet, consectetur adipiscing elit.");
143.         }
144.
145.         if (txt && rnd.NextDouble() >= 0.5)
146.         {
147.             sbHtml.Append(rnd.NextDouble());
148.             sbCshtml.Append("@variableReference");
149.         }
150.
151.         sbHtml.Append("</p>");
152.         sbCshtml.Append("</p>");
153.     }
154.
155.     private static void Add_Logical_Condition(StringBuilder sbHtml, StringBuilder sbCshtml, Random
rnd)
156.     {
157.         if (rnd.NextDouble() >= 0.5)
158.         {
159.             sbCshtml.Append(@"
160. @{
161.     if(true)
162.     {");
163.             Add_Html_Paragraph(sbHtml, sbCshtml, rnd);
164.             sbCshtml.Append('}');
165.
166.             if (rnd.NextDouble() >= 0.5)
167.             {
168.                 sbCshtml.Append(@"
169.     else
170.     {");
171.                 sbCshtml.Append("<span>else value</span>");
172.                 sbCshtml.Append('}');
173.             }
174.
175.             sbCshtml.Append('}');
176.         }
177.
178.     }
179.
180.     private static void Add_Cycle(StringBuilder sbHtml, StringBuilder sbCshtml, Random rnd)
181.     {
182.         if (rnd.NextDouble() >= 0.5)
183.         {
184.             sbCshtml.Append(@"<ol>");
185.             sbCshtml.Append(@"
186. @{
187.     for (int i = 0; i < 5; i++)
188.     {
189.         <li>@i</li>
190.     }
191. }");
192.             sbCshtml.Append(@"</ol>");
193.
194.             sbHtml.Append(@"<ol>");
195.             sbHtml.Append(@"<li>0</li>");
196.             sbHtml.Append(@"<li>1</li>");
197.             sbHtml.Append(@"<li>2</li>");
198.             sbHtml.Append(@"<li>3</li>");
199.             sbHtml.Append(@"<li>4</li>");
200.             sbHtml.Append(@"</ol>");
201.         }
202.     }
203. }
```