

Informatikos fakultetas

**T120B162 Programų sistemų testavimas**

**3 laboratorinis darbas**

|  |  |  |
| --- | --- | --- |
|  | Studentas: | Ignas Jasonas IFF-6/6 |
| Dėstytojas: | doc. Šarūnas Packevičius |

Kaunas 2019

Turinys

[1. Testuojama sistema 3](#_Toc27531478)

[2. Programos kodo peržiūra 3](#_Toc27531479)

[3. Kodo peržiūra 3](#_Toc27531480)

[4. Statinė analizė 7](#_Toc27531481)

[5. Programos kodo statinė analizė 7](#_Toc27531482)

[6. Nauja statinės analizės taisyklė 9](#_Toc27531483)

[7. Išvados 11](#_Toc27531484)

# Testuojama sistema

Testuojama sistema pasirinka pirmąjame laboratoriniame darbe aprašyta sistema GraffitiTour. Šiame laboratoriniame darbe bus panaudotas statinės kodo analizės įrankis eslint, kad galėtume aptikti klaidas prieš kompiliuojant ir vykdant kodą.

**Darbo uždaviniai:**

1. Pritaikyti statinę projekto kodo analizę be jokio įrankio – peržiūrėti programos kodą ir rasti potencialias klaidas;
2. Susirasti tinkamus statinės kodo analizės įrankius kuriamam projektui;
3. Pritaikyti statinės kodo analizės įrankius programos kodui ir identifikuoti programos kodo klaidas;
4. Parašyti naują taisyklę, kuri būti pritaikoma statiškai nagrinėjant kodą;
5. Statinės kodo analizės rezultatus pateikti laboratorinio darbo ataskaitoje.

# Programos kodo peržiūra

Pirmiausia yra peržiūrimas programos kodas nenaudojant jokių įrankių. Šis procesas vadinamas kodo peržiūra. Tam, kad kodo peržiūra būtų nuosekli, vadovaujamasi šiais kodo peržiūros punktais:

* 1. Kodo formatavimas – ar kodas rašomas nuosekliai, ar tinkami kodo eilučių atitraukimai nuo kairiojo krašto, ar yra taikomos teisingos kodo rašymo taisyklės
  2. Kodo architektūra – ar kodas yra išskaidytas į komponentus arba failus
  3. Kodo gerosios praktikos – ar nėra hard-coded kinatmųjų, ar naudojamos konstantos vietoj įkoduotų bereikšmių skaičių (pvz., if (role == 1)), ar vengiama didelių if/else blokų, ar tinkamai išnaudojamos projekte naudojamų karkasų galimybės ir pan;
  4. Kodo saugumas – ar kodas yra apsaugotas nuo SQL injekcijų, cross-side scripting, ar tie programos duomenys/rezultatai, kurie neturėtų būti matomi, nėra išvedami į konsolę/naršyklės langą, ar tikrinamos įvedimo laukų reikšmės (ar yra taikoma laukų validacija).

# Kodo peržiūra

Buvo peržiūrėta keletas failų ir iš jų išrinkti pastebėti neatitikimai ar klaidos:

|  |  |
| --- | --- |
|  | |
| Failo kodo eilutė(s) | 10 |
| Klaidos kategorija (iš sąrašo) | Gerosios praktikos |
| Klaidos apibūdinimas | Konstantos iškėlimo galimybė |
| Galimas sprendimas/komentaras | Kad kodą geriau būtų galima skaityti, reikia kiek galima daugiau išsikelti hardcoded reikšmes į konstantas ir suteikti prasmingus vardus |

|  |  |
| --- | --- |
|  | |
| Failo kodo eilutė(s) | 29-30 |
| Klaidos kategorija (iš sąrašo) | Gerosios praktikos |
| Klaidos apibūdinimas | Nepanaudoti kintamieji |
| Galimas sprendimas/komentaras | Jei nereikia kintamųjų, reikia juos išrinti |

|  |  |
| --- | --- |
|  | |
| Failo kodo eilutė(s) | 8-17 |
| Klaidos kategorija (iš sąrašo) | Kodo saugumas |
| Klaidos apibūdinimas | Nesunaikinamas sukurtas listener‘is |
| Galimas sprendimas/komentaras | Kad nebūtų atminties nutekėjimo, reikia sunaikinti listener‘į, kai jo nebereikia |

|  |  |
| --- | --- |
|  | |
| Failo kodo eilutė(s) | 18-23 |
| Klaidos kategorija (iš sąrašo) | Gerosios praktikos |
| Klaidos apibūdinimas | Nereikalingas komentaras |
| Galimas sprendimas/komentaras | Komentarai, kurie nieko nereiškia ir yra nereikalingi, turėtų būti ištrinti |

|  |  |
| --- | --- |
|  | |
| Failo kodo eilutė(s) | 44-60 |
| Klaidos kategorija (iš sąrašo) | Kodo formatavimas |
| Klaidos apibūdinimas | Nesuformatuotas css kodas |
| Galimas sprendimas/komentaras | Reikėtų pagal standartus suformatuoti kodą |

|  |  |
| --- | --- |
|  | |
| Failo kodo eilutė(s) | 13 |
| Klaidos kategorija (iš sąrašo) | Kodo saugumas |
| Klaidos apibūdinimas | Nenurodyti kintamųjų tipai |
| Galimas sprendimas/komentaras | Reikėtų nuyrodyti tipus, kurių gali būti kintamieji |

# Statinė analizė

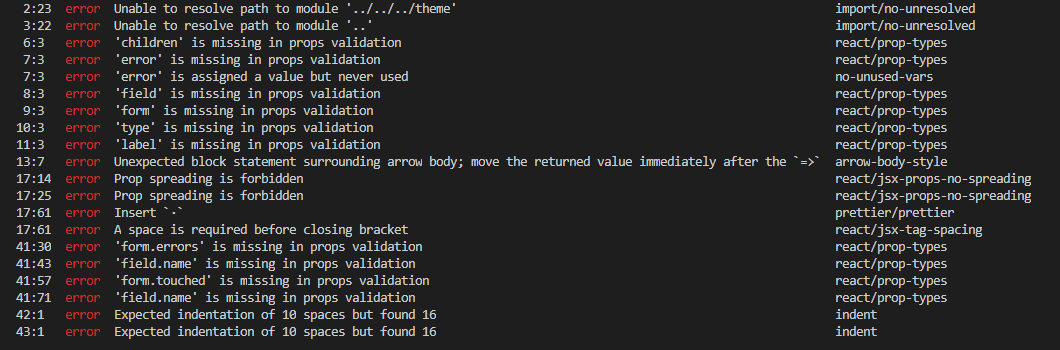
Statinei analizei vykdyti naudojamas ESLint įrankis. Šis įrankis nesudėtingai kofigūruojamas, lengvai išplečiamas naujomis taisyklėmis, turi daug vartototų ir nuolat pildomas.

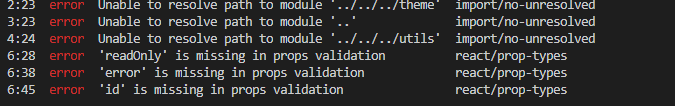
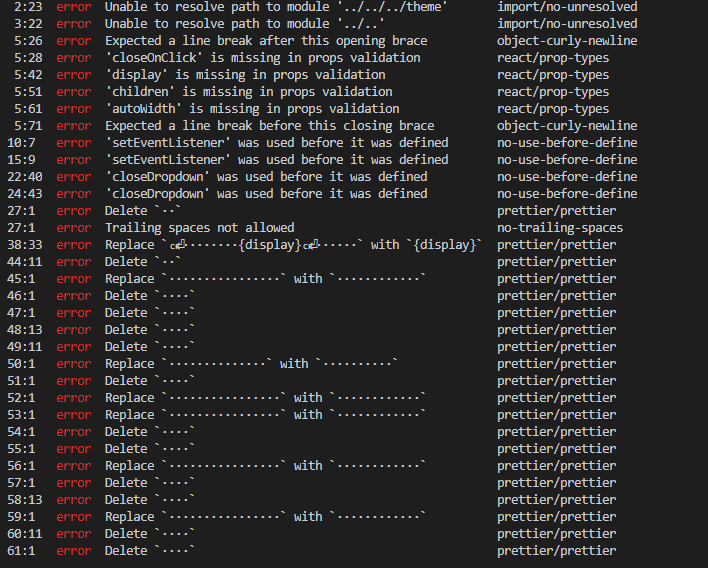
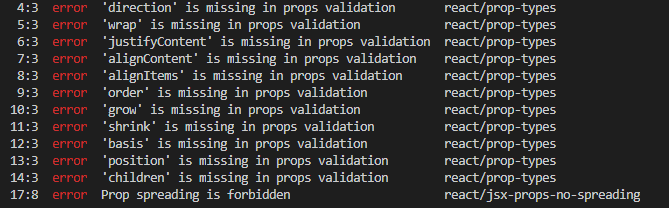
# Programos kodo statinė analizė

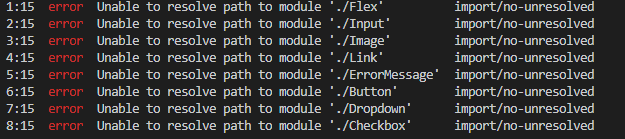
Paleidus analizės įrankį ir patikrinus visus failus buvo surasta daug klaidų.



Daugumą klaidų sudarė nepanaudoti kintamieji, nenurodyti naudojamų kintamųjų tipai, blogai importuojami failai. Kai kur blogai suformatuotas kodas.



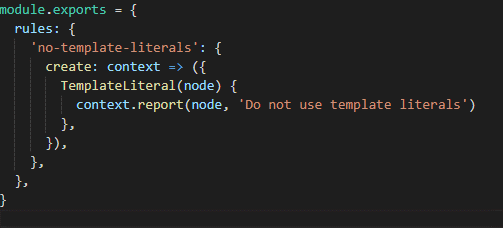


# Nauja statinės analizės taisyklė

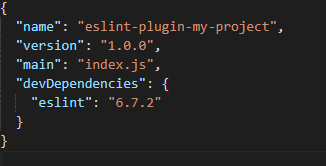
Kad galėtum sukurti naują statinės analizės taisyklę, reikia sukurti atskirą įskiepį ESLint įrankiui ir aprašyti pačią taisyklę. Naujame folderyje sukūriau 2 failus.



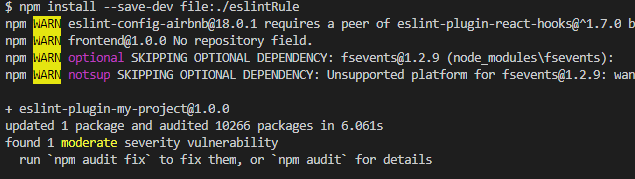
Index.js faile aprašyta pati taisyklė:



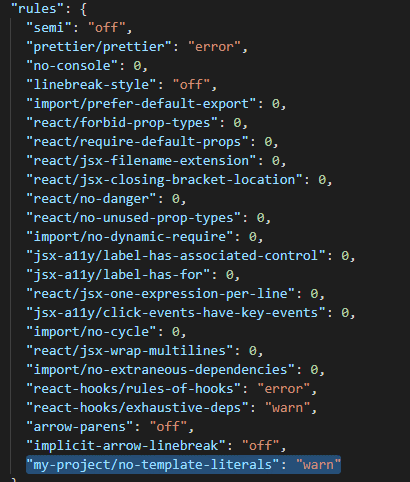
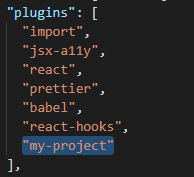
O package.json faile aprašyta bendra informacija apie šią taisyklę:



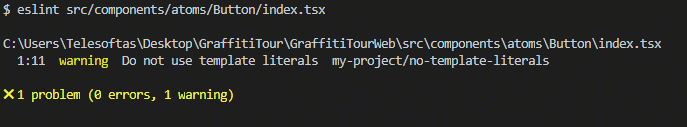
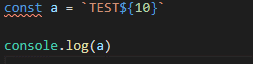
Šiuo atveju taisyklė neleidžia naudoti string literal tipo kintamųjų. Kad ją įjungtume, turime įrašyti į projektą šį įskiepį:



Ir sukonfigūruoti ESLint įrankį, kad ją galėtų naudoti:



Tuomet, paleidus įrankį faile, kuriame panaudotas šiai taisyklei prieštaraujantis kodas, gauname rezultatą:



# Išvados

Laboratorinio darbo metu buvo išmoktos galimos klaidų kategorijos, susipažinta su kodo peržiūros procesu ir statine analize. Statinei analizei buvo panaudotas ESLint įrankis, juo įvykdyti projekto statinė analizė ir pridėta nuosava nauja taisyklė, bei ji išbandyta.