KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

T120B162 Programų sistemų testavimas

Lab3. Statinė kodo analizė.

Komanda

Atliko:

Aleksas Juzukonis IFF-1/8

Lukas Kuzmickas IFF-1/6

Mildaras Karvelis IFF-1/4

Priėmė:

doc. prakt. Guogis Evaldas

Turinys

Įvadas

Atliksime statinę kodo analizę mūsų programėlei. Pradžiai atliksime visos programėlės pasirinktų kodo dalių statinę analizę su pasirinktinais įrankiais, tada susikursime kelias statinės analizės taisykles ir pritaikysime įrankius.

Pasirinkti įrankiai

Pasirinktas “Lint” statinės kodo analizės įrankis, dėl lengvo pritaikymo su mūsų Typescript kodu (1pav.).

1 pav. Javascript karkasas statinei kodo analizei „Lint“.

Kodo peržiūra

Svarbiausia kodo dalis yra mūsų duombazės kodas ir komponentai, šis kodas privalo būti logiškas ir gerai parašytas. Todėl pasirinkome analizuoti būtent tik šitas kodo dalis, dėl programėlės kodo eilučių dydžio.

```
export default {
  files: ["src/hooks/**/*.{js,jsx,ts,tsx}", "src/components/**/*.{js,jsx,ts,tsx}"],
  languageOptions: {
    parserOptions: {
      ecmaVersion: 2020,
      sourceType: "module",
      ecmaFeatures: {
        jsx: true,
      },
    },
    parser: tsEslintParser, // TypeScript parser
    globals: globals.browser, // Globals setup
  },
```

2 pav. „Lint“ įrankio analizuojamas kodas.

Statinė kodo analizė

Atliekame statinę kodo analizę „hooks“ (duombazės) ir „components“ (komponentų) dalims. Iš rezultatų matome, kad problemų yra nemažai (žr. 3 pav.) tarp jų 232 klaidos pranešimai ir 105 įspėjimai.

```
✖ 337 problems (232 errors, 105 warnings)
  0 errors and 4 warnings potentially fixable with the `--fix` option.
```

3 pav. „Lint“ įrankio statinės analizės rezultatai.

Mūsų „Lint“ įrankis buvo sukonfigūruotas dirbti su „React-Native“ aplikacijos kodu. Dažniausiai pasikartojančios klaidos ir įspėjimai:

Tipų apsirašymas ir grąžinimas

```
62:31  warning  Missing return type on function

                                        @typescript-eslint/explicit-function-return-type
```

Nepanaudoti kintamieji

```
C:\Users\Vartotojas\Desktop\3 kursas\unifit_test\src\hooks\getMissionComplete.tsx
  4:3   warning  'addDoc' is defined but never used                @typescript-eslint/no-unused-vars
  5:3   warning  'serverTimestamp' is defined but never used       @typescript-eslint/no-unused-vars
  8:3   warning  'deleteDoc' is defined but never used             @typescript-eslint/no-unused-vars
 15:71  warning  Missing return type on function                   @typescript-eslint/explicit-function-return-type
 29:13  warning  'missionDoc' is assigned a value but never used   @typescript-eslint/no-unused-vars
 51:69  warning  Missing return type on function                   @typescript-eslint/explicit-function-return-type
 69:13  warning  'missionData' is assigned a value but never used  @typescript-eslint/no-unused-vars
```

Jest klaidos (tačiau jos nėra aktualios)

```
C:\Users\Vartotojas\Desktop\3 kursas\unifit_test\src\components\Main\Home\FillerModal.test.tsx
  5:1   error  'describe' is not defined  no-undef
  6:3   error  'it' is not defined        no-undef
  9:5   error  'expect' is not defined    no-undef
 12:3   error  'it' is not defined        no-undef
 15:5   error  'expect' is not defined    no-undef
```

Statinės kodo analizės taisyklės

Apsirašome kelias paprastas statinės analizės taisykles:

- *Neleisti kelių tuščių eilučių* - ši taisyklė užtikrina, kad kode nebūtų per daug tuščių eilučių, kas gali padaryti kodą per daug išsklaidytą ir sunkiai skaitomą ;
- *Neleisti naudoti var* - ši taisyklė užtikrina, kad kode nebūtų naudojamas var kintamųjų deklaravimui, vietoj jo rekomenduojama naudoti let arba const ;
- *Užtikrinti, kad kintamieji ir funkcijos būtų pavadinti pagal camelCase taisykles;*
- *Užtikrinti nuoseklią indentaciją;*
- *Naudoti tik viengubas kabutes:*
- *Privalomi kabliataškiai;*
- *Riboti eilutės ilgį iki 100 simbolių;*
- *Neleisti naudoti console.log (tik console.warn ir console.error leidžiami).*
- *Kintamieji turi pradidėti iš mažosios raidės - ši taisyklė užtikrina, kad visi kintamieji prasidėtu iš mažosios raidės ir juos būtų galima lengvai atskirti*
- *Nenaudojami importai - ši taisyklė užtikrina, kad nebūtų nereikalingu importu, kas gali padaryti kodą švaresnį ir labiau įskaitomą*

Rezultatai po taisyklių paleidimo su statinės analizės įrankiu (žr. 4 pav.)

```
✖ 789 problems (765 errors, 24 warnings)
  665 errors and 3 warnings potentially fixable with the `--fix` option.
```

4 pav. Statinės analizės taisyklių rezultatai.

Išvados

Atliekant statinę kodo analizę pastebėjome, kad pats programėlės kodas nėra gerai aprašytas. Pačiam kodui trūksta tvarkos. Reikėtų gero kodo pertvarkymo, pagal statinės analizės pateiktas kodo klaidas.

Šaltiniai

- ESLint. *ESLint: The pluggable linting utility for JavaScript and JSX*. Available at: https://eslint.org [Accessed 1 Dec. 2024].