Static Code Analysis & Code Review

UC14 - Publicar e Testar Aplicações Web

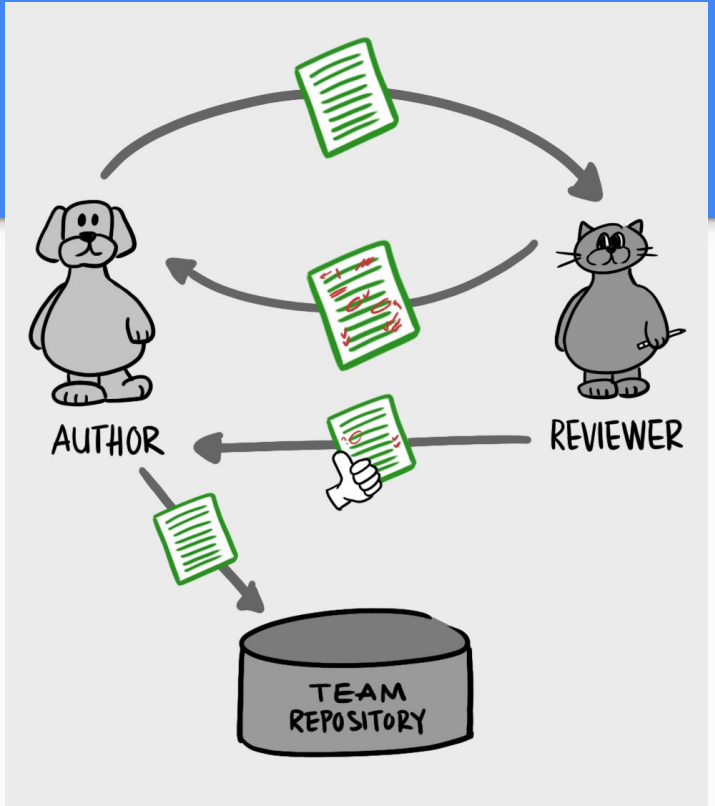
Técnico em Desenvolvimento de Sistemas Bilingue
Prof. Daniel Lopes Ferreira

Agenda

1. Revisão da aula passada
2. Exercício - Static Code Analysis


OK. BACK TO
"WORK"

Code Review



Static Analysis ou Static Code Analysis

A análise estática, também chamada de análise de código estático, é um método de depuração feito examinando o código sem a execução do programa.

O processo fornece uma compreensão da estrutura do código e pode ajudar a garantir e validar que o código adere aos padrões da empresa e do projeto ou aplicação.

Ferramentas automatizadas podem nos ajudar na realização das análises estáticas.

O processo irá analisar todo o código do projeto conforme configurado para verificar se há vulnerabilidades enquanto valida o código.

Static Analysis ou Static Code Analysis

A análise estática é geralmente boa para encontrar problemas de codificação, tais como:

- Erros de programação ou boas práticas: por exemplo, const vs let
- Violações de padrão de codificação: por exemplo, variáveis que não seguem padrão de código definido - camelCase vs snake_case
- Valores indefinidos: por exemplo, sugestão de melhorias de tipagem
- Violações de sintaxe
- Vulnerabilidades de segurança

Static Analysis ou Static Code Analysis

- Pode abordar no código-fonte problemas que podem levar a **estouros de buffer** ou **loops infinitos**
- Pode verificar condições que jamais acontecerão como um if que sempre será true ou false

Static Analysis ou Static Code Analysis

Como é feita a análise estática?

- Geralmente é automatizado
- Realizado antes do teste e durante o desenvolvimento
- Uma vez que o código é escrito, um analisador de código estático deve ser executado para examiná-lo
- Vai gerar um relatório de erros e avisos (warnings)
- É possível que o software sinalize falsos positivos, por isso é importante que alguém revise e descarte essas possibilidades

Static Analysis ou Static Code Analysis

Como é feita a análise estática?

- Após descartados os falsos positivos os desenvolvedores devem atuar para corrigir os apontamentos do relatório
- Ao final da revisão, o código pode seguir o fluxo de desenvolvimento (code review, teste, implantação)
- Sem a utilização de ferramentas, a análise estática terá muito trabalho, já que os humanos terão que revisar o código e descobrir como ele se comportará sem executá-lo

```
$ gulp js:lint
(node:22881) fs: re-evaluating native module sources is not supported.
[09:22:39] Using gulpfile ~/Projects/xwingman/gulpfile.js
[09:22:39] Starting 'js:lint:server'...
[09:22:39] Starting 'js:lint:client'...
[09:22:40]
/Users/justin/Projects/xwingman/server.js
  9:3  error  Unexpected console statement  no-console

✖ 1 problem (1 error, 0 warnings)

[09:22:40] 'js:lint:server' errored after 843 ms
[09:22:40] ESLintError in plugin 'gulp-eslint'
Message:
    Failed with 1 error
[09:22:40] Finished 'js:lint:client' after 931 ms
```

Static Code Analysis - Exercício eslint

Instalar e configurar o eslint

```
npm install --save-dev eslint @eslint/js globals
```

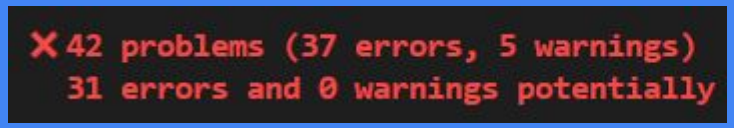
Para executar a validação altere o package.json para incluir um script de lint

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node index.js",
  "lint": "eslint ."
},
```

https://github.com/daniellferreira/aula-publicacao-teste-apps-web

Exercício - eslint

ex1.js



ex2.js

Total

https://github.com/daniellferreira/aula-publicacao-teste-apps-web

Entrega: relatório e repositório

Data: próxima aula

1. Configurar o arquivo eslint.config.js para encontrar esses problemas e executar o comando **npm run lint**
2. Criar script **lint:fix** no package.json para corrigir automaticamente
3. Configurar o editor para corrigir automaticamente ao salvar
4. Desabilitar o linter um arquivo inteiro de cada vez para ter os resultados por arquivo
5. Desabilitar por linha os comandos console.log
6. DESAFIO: criar um arquivo ex-challenge.js e um arquivo de configs com o maior número de regras e problemas para consertar exemplificados, quem configurar e exemplificar a maior quantidade de regras irá ganhar um prêmio