TRACKD

Maurício Witter
Alexsander Paulo Horn

1 ANÁLISE ESTÁTICA DE CÓDIGO

Para o Projeto *Trackd*, é empregado como analisador estático do código a ferramenta *ESlint*. O ESlint é semelhante ao *SonarQube*, ademais, ele fornece um ambiente de extensão com plugins que permitem criar ou seguir padrões de projeto, *style guides* e boas práticas de codificação. Além disso, os plugins se integram com bibliotecas e fazem checagem sobre possíveis bugs e codificação duvidosa ou incorreta.

2 ESLINT

O *ESLint* faz análise estática do código para encontrar problemas rapidamente. Ele é incorporado à maioria dos editores de texto e permite executar o *ESLint* como parte do pipeline de integração contínua. Os problemas podem ser qualquer coisa, desde possíveis erros de tempo de execução, até não seguir as melhores práticas, até problemas de estilo.

2.2 REGRAS

As regras são o principal componente do *ESLint*. Uma regra válida se o código atende a uma certa expectativa e o que fazer se não atender a essa expectativa. As regras também podem conter opções de configuração adicionais específicas para essa regra.

2.3 ARQUIVO DE CONFIGURAÇÃO

Um arquivo de configuração do *ESLint* é um local em que se coloca a configuração do *ESLint* do projeto. Pode-se incluir regras internas, *plugins* com regras personalizadas, configurações compartilháveis, os arquivos desejados que as regras sejam aplicadas e mais.

2.4 PLUGINS

Um *plugin ESLint* é um módulo npm que pode conter um conjunto de regras, configurações, processadores e ambientes *ESLint*. Muitas vezes, os *plugins* incluem regras personalizadas. Os *plugins* podem ser usados para impor um guia de estilo e oferecer suporte a extensões *JavaScript* (*TypeScript*), bibliotecas como (*React*) e *frameworks* (*Angular*).

2.5 PARSER

Um analisador *ESLint* converte o código em uma árvore de sintaxe abstrata que o *ESLint* pode avaliar. Por padrão, o *ESLint* usa o analisador *Espree*, compatível com tempos de execução e versões padrão do *JavaScript*.

2.6 INSTALAÇÃO E CONFIGURAÇÃO

Para instalar o *ESlint* é bem simples, basta executar o comando:

$$npm\ init\ @eslint/config$$

A configuração é feita no arquivo na raiz do projeto, chamado de $.eslintrc.js$. Neste arquivo, pode-se escrever as regras de padronização e configurar *plugins* e *style-guides*. Para formatar o código nos padrões das regras e remover erros, basta executar o comando:

$$eslint\ 'lib/\ **\ /\ *.ts'\ --\ fix$$

fonte: autores.

3 SENTRY

> O Sentry é uma plataforma de rastreamento de erros e monitoramento de desempenho voltada para o desenvolvedor que ajuda os desenvolvedores a ver o que realmente importa, resolver mais rapidamente e aprender continuamente sobre seus aplicativos.

O Sentry auxilia os desenvolvedores e empresas a monitorar erros em tempo real em seus aplicativos e plataformas e dando feedback sobre desempenho. Desta

forma, o Sentry fornece informações valiosas do local específico que aconteceu um erro crítico e os desenvolvedores podem resolver o problema em menos tempo e evitar perda de clientes e dinheiro para as empresas.

3.1 INSTALAÇÃO

Para instalar o Sentry, é preciso escolher a plataforma de desenvolvido e instalar a biblioteca específica para a plataforma. No *Nodejs*, pode-se instalar o Sentry com via cli com o comando:

$$yarn\ add\ @sentry/node\ @sentry/tracing$$

3.2 CONFIGURAÇÃO

Para configurar, basta importar o Sentry e inicializar uma instância do Sentry. Para capturar um erro, em vez de adicionar para cada *try/catch*, pode-se criar um *error handling* e capturar o erro quando um *errorEvent* acontecer.

```
import * as Sentry from '@sentry/node';
import * as Tracing from '@sentry/tracing';
import type { FastifyInstance } from 'fastify';
import { Prisma } from './prisma';

const makeSentry = (app: FastifyInstance) => {
  Sentry.init({
    dsn: process.env.SENTRY_DSN,
    tracesSampleRate: 1.0,
    environment: process.env.NODE_ENV,
    release: 'trackd@1.2.3',
    integrations: [
      new Sentry.Integrations.Http({ tracing: true }),
      new Sentry.Integrations.RequestData(),
      new Tracing.Integrations.BrowserTracing(),
      new Tracing.Integrations.Prisma({ client: Prisma }),
    ],
  });

  app.addHook('onError', (request, reply, error, done) => {
    if (process.env.NODE_ENV === 'production') {
      Sentry.captureException(error);
    }
    done();
  });
};

export { makeSentry };
```

Fonte: autores.