Cebraspe
Centro Brasileiro de Pesquisa em Avaliação e Seleção e de Promoção de Eventos

CONCURSO PÚBLICO

GOVERNO DO DISTRITO FEDERAL
PROCURADORIA-GERAL DO DISTRITO FEDERAL

CARGO 2: ANALISTA JURÍDICO
ESPECIALIDADE: ANALISTA DE SISTEMA (DESENVOLVIMENTO DE SISTEMA)

PROVA DISCURSIVA

Aplicação: 29/8/2021

PADRÃO DE RESPOSTA DEFINITIVO

O SonarQube é um conjunto de analisadores estáticos que podem ser usados para identificar áreas de melhoria em um código. Ele permite a análise da dívida técnica em um projeto e seu acompanhamento no futuro. O SonarQube realiza diversas análises de *bugs*, *code smells*, *test coverage*, vulnerabilidades e blocos duplicados. Sendo assim, é essencial a identificação de problemas como, por exemplo, um WebClient que esqueceu-se de dar um *dispose*.
*Considerar como uma informação adicional relevante caso o candidato exemplifique as severidades dos problemas (*issues*) de acordo com a documentação *do sonarqube: blocker, critical, major, minor e info*
(https://docs.sonarqube.org/7.4/user-guide/issues/)

A principal característica dessa ferramenta é permitir a inspeção contínua de qualidade do código, que possibilita a realização de revisões automáticas com análise estática de código, detecção de erros, vulnerabilidades de segurança, entre outros recursos, visando-se à melhora contínua da qualidade do código produzido.

Em suma, o SonarQube é uma ferramenta automática de revisão de código para detecção de *bugs* (erros), vulnerabilidades e *code smells* (odores de código). Pode ser integrado ao fluxo de trabalho existente para permitir a inspeção contínua de código nas ramificações de um projeto e para receber solicitações.

No SonarQube, os analisadores contribuem com regras executadas no código-fonte para gerar problemas. Existem quatro tipos de regras: *code smells*, ou odores de código (domínio de manutenção); *bug* (domínio de confiabilidade); vulnerabilidade (domínio de segurança); e ponto de acesso de segurança (domínio de segurança).

Para odores e erros de código, espera-se nenhum falso positivo, para que os desenvolvedores não precisem se perguntar se uma correção é necessária. Para as vulnerabilidades, o objetivo é fazer com que mais de 80% dos problemas sejam verdadeiros positivos. As regras de ponto de acesso de segurança chamam a atenção para códigos sensíveis à segurança. Espera-se que mais de 80% dos problemas sejam resolvidos rapidamente como “revisados” após a revisão por um desenvolvedor.

Um servidor SonarQube possui três processos principais: servidor *web* para desenvolvedores, gerentes para procurar instantâneos de qualidade e configurar a instância do SonarQube; servidor de pesquisa com base no Elasticsearch, para retornar pesquisas da interface do usuário; e Compute Engine Server, encarregado de processar relatórios de análise de código e salvá-los no banco de dados SonarQube.

As funções dos *code smells* incluem:

- F1 — parâmetros em excesso: as funções devem ter número pequeno de parâmetros, sendo desejável não ter nenhum e evitável ter mais que três;
- F2 — parâmetros de saída: são os parâmetros inesperados. Os leitores esperam que sejam de entrada em vez de saída;
- F3 — parâmetros lógicos: parâmetros booleanos explicitamente declaram que a função faz mais de uma operação. Eles são confusos e devem ser eliminados;
- F4 — função morta: devem-se descartar os métodos que nunca sejam chamados.

O princípio da responsabilidade única e tamanho de classes dispõe que uma classe ou módulo deve ter apenas um único motivo para mudar. Esse princípio dá orientação para o tamanho da classe. Classes devem ser pequenas. Da mesma forma, nas funções, ser pequena também é uma regra principal quando o assunto for criar classes.
*Considerar o conceito de "Classes Should Be Small" Small de Clean Code: A Handbook of Agile Software Craftsmanship - Robert C. Martin (MARTIN, 2008, p. 136)
O adjetivo **pequena** utilizado para descrever como as classes devem ser elaboradas se dá pelo fato de elas deverem **ter poucas responsabilidades.**

Os nomes das classes devem ser formulados com substantivo, não devendo conter verbos. Por sua vez, os nomes dos métodos devem conter verbos. Devem-se nomear métodos de acesso, alteração e autenticação segundo seus valores e adicionar o prefixo *get*, *set* ou *is* de acordo com o padrão Javabean.

QUESITOS AVALIADOS

2.1
0 – Não abordou o objetivo do SonarQube nem sua relação com a inspeção contínua.
1 – Abordou, de forma insuficiente, o objetivo do SonarQube e não tratou da inspeção contínua.
2 – Abordou, de forma completa, o objetivo do SonarQube, mas tratou de forma insuficiente sobre sua relação com a inspeção contínua.
3 – Abordou, de forma completa, o objetivo do SonarQube e sua relação com a inspeção contínua.

2.2
0 – Não indicou nenhuma regra do SonarQube.
1 – Indicou somente uma regra do SonarQube.
2 – Indicou somente duas regras do SonarQube.
3 – Indicou somente três regras do SonarQube.
4 – Indicou as quatro regras do SonarQube.

2.3.1
0 – Não indicou nenhum processo do SonarQube.
1 – Indicou somente um processo do SonarQube.
2 – Indicou somente dois processos do SonarQube.
3 – Indicou três processos do SonarQube.

2.3.2
0 – Não abordou o objetivo de nenhum processo do SonarQube.
1 – Abordou o objetivo de apenas um processo do SonarQube.
2 – Abordou o objetivo de apenas dois processos do SonarQube.
3 – Abordou o objetivo de três processos do SonarQube.

2.4.1
0 – Não indicou nenhuma função dos *code smells*.
1 – Indicou somente uma função dos *code smells*.
2 – Indicou somente duas funções dos *code smells*.
3 – Indicou somente três funções dos *code smells*.
4 – Indicou as quatro funções dos *code smells*.

2.4.2
0 – Não explicou nenhuma função dos *code smells*.
1 – Explicou apenas uma função dos *code smells*.
2 – Explicou apenas duas funções dos *code smells*.
3 – Explicou apenas três funções dos *code smells*.
4 – Explicou as quatro funções dos *code smells*.

2.5
0 – Não abordou ou abordou incorretamente o aspecto.
1 – Abordou corretamente apenas a responsabilidade única ou o tamanho de classes.
2 – Abordou corretamente a responsabilidade única e o tamanho de classes.

2.6
0 – Não abordou ou abordou incorretamente o aspecto.
1 – Abordou apenas o nome de classes ou o nome de métodos, de forma insuficiente.
2 – Abordou corretamente o nome de classes, mas abordou de forma incompleta o nome de métodos.
3 – Abordou, corretamente e de forma completa, o nome de classes e o nome de métodos.