1. IDENTIFICAÇÃO

Padrão	**Inspeção automatizada de código-fonte**
Segmento	**Arquitetura de Soluções**
Código	**P05.009**
Revisão	**Dezembro de 2020**

2. PUBLICAÇÃO

Versão	Data para adoção	**Publicação**
v. 2017	**29 de maio de 2017**	**PORTARIA "N" Nº 29 de maio de 2017.**
v.2020	**14 de dezembro de 2020**	**Iplanet** http://www.rio.rj.gov.br/web/epingrio/catalogo

3. PROPÓSITO DOPADRÃO

A padronização da **Inspeção Automatizada de Código-fonte** visa monitorar e melhorar a qualidade do código das aplicações, com uso de ferramentas de análise e indicadores de qualidade. Entende-se por qualidade do código-fonte um código simples e flexível.

4. RESPONSÁVEL PELO PADRÃO

Órgão	**IplanRio**
Diretoria	**Diretoria de Sistemas**
Setor	**Arquitetura de Software**
Responsável	**Arquiteto de Software**

5. DESCRIÇÃO DO PADRÃO

A inspeção automatizada de código-fonte tem o objetivo de analisar o código e gerenciar sua conformidade e qualidade.

A análise do código-fonte pode ser configurada para apurar a qualidade do código a partir de métricas e regras, como por exemplo: arquitetura e desenho, duplicações de código, testes unitários, comentários de código, complexidade de código, bugs potenciais.

Como referência de qualidade do código e estratégia para viabilizar a evolução da qualidade dos códigos na Prefeitura do Rio de Janeiro optou-se pelos critérios (métricas e indicadores) apresentados a seguir:

Indicador/Métrica	Descrição
Block Issues	É um tipo de Indicador que evidencia defeitos com alta probabilidade de afetar o comportamento do aplicativo em produção. https://docs.sonarqube.org/latest/user-guide/issues/
Critical Issues	É um tipo de Indicador que evidencia defeitos com baixa probabilidade de afetar o comportamento do aplicativo em produção ou um problema que representa uma falha de segurança. https://docs.sonarqube.org/latest/user-guide/issues/
Coverage	Indicador que evidencia o percentual de cobertura dos testes unitários. Quanto menor o resultado indica a necessidade de aumentar a cobertura os testes unitários.
Duplicated lines	Indicador que evidencia o percentual de linhas do código-fonte duplicadas.
Vulnerabilities	Indicador que evidencia número de problemas relacionados a vulnerabilidades de código, necessitando de revisão do código nos aspectos de segurança.
Rules / Cognitive Complexity of methods should not be too high (*)	Indicador que evidencia complexidade de cada método. Quanto maior o valor mais complexo o método.

(*) cálculo da métrica - https://www.sonarsource.com/docs/CognitiveComplexity.pdf

6. POLÍTICA E NORMATIZAÇÃO DE USO

6.1. Fica estabelecido como padrão tecnológico para **Inspeção Automatizada de Código-fonte**, os itens relacionados como adotado na especificação técnica deste documento.

6.2. Todos os projetos de sistemas e aplicações desenvolvidos para a Prefeitura da Cidade do Rio de Janeiro deverão ter seus códigos-fontes inspecionados de acordo com a especificação técnica adotada neste padrão.

6.3. A inspeção do código-fonte será realizada com base nos alertas de acompanhamento da medição de qualidade, conforme limites definidos abaixo:

Indicador/Métrica	Tipo de limite	Limite para rejeição	Disparo de Indicação
Block Issues	**maior que**	**0**	se o total de problemas do tipo *Blocker* for superior a 0.
Critical Issues	**maior que**	**0**	se o total de problemas do tipo *Critical* for superior a 0.
Coverage	**menor que**	**10%**	se os testes unitários forem inferiores a 10% do código fonte.
Duplicated lines	**maior que**	**8%**	se da quantidade total de linhas do código fonte mais de 8% for de linhas duplicadas.
Vulnerabilities	**maior que**	**0**	se a quantidade de vulnerabilidades for superior a 0.
Rules / Cognitive Complexity of methods should not be too high(**)	**Maior**	**25**	se a medida encontrada for maior a 25.

(**) Na análise de complexidade ciclomática da Rule **Cognitive Complexity of methods should not be too high** não existe uma forma padrão e nativa do Sonar para ignorar métodos que não sejam de negócio como ***equals*** e ***hashcode***. Existem formas para minimizar essa questão, mas todas envolvem modificações no código da aplicação:
1 - https://stackoverflow.com/questions/41873772/how-to-override-equals-without-increasingcyclomatic-complexity
2 - https://stackoverflow.com/questions/47923297/how-to-make-sonaqube-ignore-the-equals-and-hashcode

6.4. Por este padrão tratar-se de uma ferramenta para o aprimoramento das entregas da equipe de desenvolvimento cabe a ela a análise das métricas e indicadores gerados, para buscar uma melhor qualidade do código-fonte: livre de erros, expressivo, simples e flexível.

a. A decisão quanto a aceitação ou não do código-fonte é de responsabilidade do líder de projeto ou da equipe, conforme estratégia estabelecida pelo responsável do serviço de desenvolvimento e de manutenção em acordo com o líder de projeto.

6.5. Cabe ao Responsável pelo Padrão junto a Diretoria de Operações da IplanRio avaliar e estabelecer níveis de serviço para a realização de backups do repositório/banco de dados e atualização das versões das ferramentas relacionadas como adotadas e recomendadas no quadro do **item 7.1** deste documento;

6.6. Todo acesso ao sistema de gerenciamento de código-fonte deverá ser autenticado conforme políticas de acesso estabelecida pelo **Responsável do Padrão** junto a Diretoria de Operações da IplanRio;

6.7. Todas as exceções e dúvidas relacionadas a este documento devem ser tratadas com o **Responsável pelo Padrão;**

6.8. Com o objetivo de atualização, modernização e capacidade de melhor atender as demandas, os componentes do padrão tecnológico **Inspeção Automatizada de Código-fonte** serão avaliados pela **Diretoria de Sistemas da IplanRio** com periodicidade de no máximo 365 dias a contar da data de publicação da portaria que o regulamenta.

7. ESPECIFICAÇÃO TÉCNICA

7.1. Especificação dos componentes:

Componente	Especificação	Situação
Ferramenta de inspeção de código-fonte	SONAR : 7.9 LTS* (http://sonar.rio.rj.gov.br/sonar)	**Adotado**

***Long Term Support**

7.2. Informações sobre os componentes adotados e recomendados

a. **SONAR**: é uma plataforma de código aberto que possibilita gerenciar a qualidade do código, oferecendo indicadores e configuração de métricas.

i. Ambiente: No ambiente de desenvolvimento, o SONAR é acessado na Prefeitura do Rio, no endereço: http://sonar.apps.rio.gov.br/. Tendo seu repositório de dados centralizado no banco de dados MySQL.

ii. Perfil IPLANRIO: A ferramenta permite múltiplos perfis para medição (profile). Sendo que a configuração a ser utilizada deverá ser o perfil personalizado com o nome IplanRio, seja para o desenvolvimento interno ou por aquisição.

iii. Configuração: A seguir a tabela de propriedades a ser configurada para cada projeto:

- Propriedades Relativas à Identificação do Projeto

Propriedades Obrigatórias

Propriedades	Exemplo	Descrição
sonar.projectKey	silfae	A chave do projeto que é único para cada projeto.
sonar.projectName	SILFAE	Nome do projeto que será exibido na interface web.
sonar.projectVersion	0.10.1	A versão do projeto.
sonar.projectDescription	Sistema Integrado de Licenciamento e Fiscalização	Descrição do projeto.

Propriedades Opcionais

Propriedades	Exemplo	Descrição
sonar.sources	. , src/main,src/hot	Diretório dos códigos-fontes. Caso não seja configurado o sonar.sources ou o sonar.tests o sonarqube buscará o código-fonte no diretório de base do projeto.
sonar.sourceEncoding	UTF-8	Tipo de codificação do código-fonte. Se não for configurado o padrão adotado será o do sistema.
sonar.projectBaseDir	jenkins/jobs/myjob/workspace	Use esta propriedade quando precisar que a análise ocorra em um diretório diferente daquele a partir do qual foi iniciada. O diretório de workspace do jenkins é um exemplo de mudança de diretório.
sonar.java.libraries	lib/*.jar	Diretório de bibliotecas de dependência do projeto para código-fonte escrito em java.
sonar.java.binaries	exploded-archives/silfae.ear/silfae_jar	Diretório do arquivo executável para código-fonte escrito em java.

- Propriedades Relativas à Cobertura de Testes

Propriedades Obrigatórias

As propriedades relativas à cobertura de testes devem ser usadas de acordo com a linguagem de programação. Seguem abaixo os links, que exibem as propriedades de diversas linguagens de programação na documentação oficial do Sonarqube.
https://docs.sonarqube.org/7.9/analysis/coverage/
https://docs.sonarqube.org/7.9/analysis/analysis-parameters/

8. DEFINIÇÕES E ABREVIAÇÕES

---

9. REFERÊNCIAS

SonarQube 7.9 – Documentação.
https://docs.sonarqube.org/7.9/

Complexidade Cognitiva
https://artesoftware.com.br/2019/02/10/complexidade-cognitiva/

Issues
https://docs.sonarqube.org/latest/user-guide/issues/

Como adotar a análise estática de código.
https://www.devmedia.com.br/como-adotar-a-analise-estatica-de-codigo/32727

Sonar Java: avaliando o código através de métricas.
http://www.devmedia.com.br/sonar-java-avaliando-o-codigo-atraves-de-metricas/31278#ixzz3KxpM836b

Sondando qualidade de código com o sonar : conheça uma ferramenta que te ajuda a monitorar a qualidade de seu código
https://www.devmedia.com.br/sondando-qualidade-de-codigo-com-o-sonar/24239

Código Limpo e seu Mapeamento para Métricas de Código Fonte.
http://www.ime.usp.br/~cef/mac499-10/monografias/lucianna-joao/arquivos/monografia.pdf

10. GRUPO TÉCNICO RESPONSÁVEL PELA ELABORAÇÃO DO PADRÃO

Diretoria de Sistemas da IplanRio

Terson Rigaud