**INFORME N° 002-2017/UGDS-LFRV\_RNPv5**

|  |  |  |
| --- | --- | --- |
| **A** | **:** | **ISABEL VEGA PALOMINO**  Jefa de la Unidad de Gestión de Desarrollo de Software (e) |
| **DE** | **:** | **LIZETTE RIVERA VASQUEZ**  Profesional de la Unidad de Gestión de Desarrollo de Software |
|  |  |  |
| **ASUNTO** | **:** | Observaciones técnicas detectadas en la revisión del Sistema RNP versión 5.0 |
|  |  |  |
| **REFERENCIA** | **:** | Contrato Nº 062-2013-OSCE |
| **FECHA** | **:** | Jesús María, 21 de Setiembre del 2017 |

Tengo el agrado de dirigirme a Usted en relación al documento de la referencia, a través del cual la Oficina de Tecnologías de la Información solicita un informe de estado técnico en lo referente al desarrollo de software del Contrato Nº 062-2013-OSCE, Servicio de consultoría de análisis, diseño, construcción, prueba e implantación del Sistema del Registro Nacional de Proveedores RNP Versión 5.0.

# ANTECEDENTES

* 1. Mediante el Contrato N° 062-2013-OSCE, de fecha 06 de noviembre del 2013, el Organismo Supervisor de las Contrataciones del Estado (OSCE) y el Consorcio Everis formalizaron la contratación del “Servicio de Consultoría de Análisis, Diseño, Construcción, Prueba e Implantación del Sistema del Registro Nacional de Proveedores, RNP, Versión 5.0” conforme a los términos de referencia del Concurso Público N° 004-2013-OSCE, por la suma de S/ 6’000,000.00, cuyo plazo de ejecución de la prestación es de veintiún (21) meses computados desde el día siguiente de suscrito el contrato, es decir desde el 07 de noviembre del 2013 hasta el 06 de agosto del 2015.
  2. Entregable aprobado por la Dirección del SEACE – Producto 01, 02 (conformidad técnica según Bases Integradas)

Mediante Memorando N° 676-2014/DSEACE, de fecha 17 de junio del 2014, la Dirección del SEACE otorga la **conformidad técnica** del “**Producto N° 01**: Modelo de Negocio, Diagnostico Situacional y Alcance de Alto Nivel Servicio de Diagnóstico de Procesos”.

Mediante Memorando N° 454-2014/DSEACE, de fecha 21 de abril del 2014, la Dirección del SEACE otorga la **conformidad técnica** del “**Producto N° 02**: Plan de Gestión y Calidad del proyecto”.

* 1. Entregable aprobado por la Dirección del SEACE – Producto 03 (conformidad técnica según Bases Integradas)

Mediante Memorando N° 677-2014/DSEACE, de fecha 17 de junio del 2014, la Dirección del SEACE otorga la **conformidad técnica** del **“Producto N° 03**: Especificación de Requerimientos, Diseño Funcional”.

* 1. Adenda N° 01 al Contrato N° 062-2013-OSCE

Con fecha 30 de junio del 2014, el Organismo Supervisor de las Contrataciones del Estado (OSCE) y el Consorcio Everis suscribieron la Adenda N° 01 al Contrato N° 062-2013-OSCE, mediante el cual se modifica la “Cláusula Cuarta: Del Pago” del Contrato N° 062-2013-OSCE, a fin de rectificar el porcentaje de pago.

* 1. Entregable aprobado por la Dirección del SEACE – Producto 04 (conformidad técnica según Bases Integradas)

Mediante Memorando N° 1140-2014/DSEACE, de fecha 23 de octubre del 2014, la Sub Dirección de Desarrollo de Proyectos otorga la **conformidad técnica** del “**Producto N° 04**:”Primera Iteración del Diseño Técnico”.

* 1. Entregable aprobado por la Dirección del SEACE – Producto 05 (conformidad técnica según Bases Integradas)

Mediante Memorando N° 1410-2014/DSEACE, de fecha 12 de diciembre del 2014, la Dirección del SEACE otorga la **conformidad técnica** del “**Producto N° 05**: Segunda Iteración del Diseño Funcional y Técnico”.

Mediante Memorando N° 1738-2014/DRNP, de fecha 15 de diciembre del 2014, la Dirección del RNP otorga la conformidad funcional al “Producto N° 05: Modelo de Negocio, Diseño Funcional y Diseño Técnico – Segunda Iteración.”.

* 1. Adenda N° 02 al Contrato N° 062-2013-OSCE

Conforme señala la adenda, *“Con Informe N°001-2015/LOG”, de fecha 05 de enero del 2014, la unidad de Logística propone la modificación del cronograma establecido en el Contrato N° 062-2013-OSCE, extendiendo los plazos de presentación de los Productos 07, 08, 09 y 10, considerando el avance significativo del servicio y que se requiere un mayor análisis y tiempo para determinar el impacto funcional y técnico como consecuencia de la nueva Ley de Contrataciones del Estado y el proyecto de su Reglamento, todo ello para alcanzar la finalidad del Contrato.”*

El 05 de enero del 2015, el Organismo Supervisor de las Contrataciones del Estado (OSCE) y el Consorcio Everis suscribieron la Adenda N° 02 al Contrato N° 062-2013-OSCE, mediante el cual se modifica la Cláusula Cuarta y Sexta del Contrato N° 062-2013-OSCE, con el fin de modificar las fechas de presentación de los entregables.

De acuerdo a lo señalado en las citadas Cláusulas Cuarta y Sexta de la Adenda N° 02 se modificó el cronograma establecido en el Contrato N° 062-2013-OSCE, el nuevo plazo de ejecución de la prestación quedó establecido desde el 07 de noviembre del 2013 hasta el 18 de noviembre del 2015.

* 1. Mediante correo electrónico de fecha 20/12/2016, personal del Consorcio Everis remite la última actualización del código fuente a personal de la Unidad de Gestión de Desarrollo de Software.
  2. El 08 de agosto del 2017, se culmina el análisis estático de software, en el cual se considera la última versión del código fuente (última actualización) remitida por el Consorcio Everis.
  3. El software analizador estático que se ha utilizado en agosto del presente año es “SONARQUBE”, el cual es un software de código libre.

# ANÁLISIS ESTÁTICO DE CÓDIGO FUENTE DEL RNP V.5.0 CON SONARQUBE

En el marco de la ejecución del Contrato N° 062-2013-OSCE, la Unidad de Gestión de Desarrollo de Software en su calidad de validador técnico en lo relacionado al desarrollo de software, conforme sus competencias descritas en el Reglamento Organizacional de Funciones del OSCE y en coordinación con la Oficina de Tecnologías de la Información ha tomado la iniciativa de realizar un análisis estático de software del Sistema RNP versión 5.0, para lo cual se utilizó la herramienta “SONARQUBE”.

# CONCEPTOS UTILIZADOS POR SONARQUBE

1. **Componente:** Pedazo de software (proyecto, módulo/paquete, archivo) o una vista o un desarrollador.
2. **Complejidad ciclomática;** Métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. El resultado obtenido en el cálculo de la complejidad ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

El Software Engineering Institute (SEI) presentó la siguiente tabla de riesgos en su artículo Software Technology Reference Guide[[1]](#footnote-1):

|  |  |
| --- | --- |
| **Complejidad Ciclomática** | **Evaluación de Riesgo** |
| 1-10 | Programa simple, sin mucho riesgo. |
| 11-20 | Más Complejo, **riesgo moderado**. |
| 21-50 | Complejo, **programa de alto riesgo**. |
| greater than 50 | Programa imposible de probar muy alto riesgo. |

1. **Defecto o Issue:** Cuando un componente no cumple con una regla de código, se registra un defecto (issue).
2. **“Deuda técnica” o Technical Debt:** Tiempo estimado requerido para corregir todos los defectos o issues de mantenimiento referidos a “código que huele mal” o Code Smells.
3. **Regla de codificación:** Una buena práctica de desarrollo de código fuente. No cumplir con reglas de codificación conlleva tener fallas de calidad y la creación de defectos en SonarQube.

En SonarQube cada **regla** de codificación entre otra información tiene asignada su tipo (bug,

vulnerabilidad, código que huele mal), severidad (bloqueante, crítica, mayor, menor, informativa) y

el tiempo estimado de resolución (utilizado para sumarizar la deuda técnica).

1. **Severidad**: El incumplimiento de las reglas de codificación de SonarQube generan **evidencias**, las cuales tienen diferentes tipos de severidad:

* **BLOQUEANTE:** Error con una alta probabilidad de impactar en el comportamiento de las aplicaciones en producción: pérdida de memoria, conexión JDBC no cerrada, ***....El código DEBE ser inmediatamente corregido.***
* **CRÍTICA:** Ya sea un error con una baja probabilidad de impacto en el comportamiento de la aplicación en producción o un tema que representa un fallo de seguridad: bloque catch vacío, inyección SQL*,* ***... El código debe ser revisada de inmediato.***
* **MAYOR:** Defecto de calidad que puede tener un alto impacto en la productividad de los desarrolladores: trozo de código no cubierto, bloques duplicados, parámetros no utilizados, ...
* **MENOR:** Defecto de calidad con un leve impacto en la productividad de los desarrolladores: líneas que no deberían ser tan largas, las sentencias "switch" deben tener al menos tres casos,...
* **INFORMATIVA:** Ni un error, ni un defecto de calidad, sólo un hallazgo.

1. **Tipos de Defectos (Issues):**
   1. **“Código que huele mal” o Code Smell:** Defecto relacionado a mantenibilidad del código fuente. Dejar el código tal como está significaría que en el mejor de los casos los desarrolladores que le dan mantenimiento tardarían más tiempo del que deberían haciendo cambios al código fuente. En el peor de los casos, los desarrolladores estarían tan confundidos por el estado del código fuente que ellos mismos introducirían errores adicionales mientras hacen cambios. Defecto que impide incluir cambios con la misma rapidez con que se hubiera modificado si se empezara el desarrollo desde cero. Este tipo de defecto afecta la mantenibilidad del sistema.
   2. **Bug:** Defecto que representa algo errado en el código fuente. Si este defecto aún no ha surgido, surgirá y probablemente en el peor momento posible. Esto necesita ser arreglado para “ayer”. Defecto que resalta un punto de falla real o potencial en el software. Este tipo de defecto afecta la confiabilidad del sistema.
   3. **“Vulnerabilidad” o Vulnerability:** Defecto relacionado a Seguridad que representa una “puerta trasera” o backdoor potencial para atacantes cibernéticos. Defecto que resalta un hueco de seguridad que puede ser utilizado para atacar el software. Este tipo de defecto afecta la seguridad del sistema.
   4. **Categorías de reglas de codificación**: El Modelo de Calidad de SonarQube divide las reglas en 3 categorías: Bugs, Vulnerabilidades de Seguridad y “Código que huele mal” o Code Smells.

# PERSONALIZACIÓN DE SONARQUBE

En función a las configuraciones que permite realizar SONARQUBE, a continuación se han realizado personalizaciones de ***Puerta de calidad y Perfiles de calidad*** que pueden ser utilizadas para el Sistema RNP versión 5.0:

* Puerta de calidad:

|  |  |
| --- | --- |
| **NOMBRE** | **COMENTARIO** |
| OSCE-RNP5 | Se extiende la configuración que por defecto proporciona SonarQube según se aprecia en la siguiente imagen: |

* Perfiles de calidad :

|  |  |
| --- | --- |
| **NOMBRE** | **COMENTARIO** |
| OSCE-Java | Usado para el lenguaje Java, contiene reglas de codificación de severidad: bloqueante, crítica y mayor. |
| OSCE-JavaScript | Usado para el lenguaje JavaScript, contiene reglas de codificación de severidad: bloqueante, crítica y mayor. |
| OSCE-XML | Usado para el lenguaje XML, contiene reglas de codificación de severidad: bloqueante, crítica y mayor. |

# PRUEBA DE RNP V 5.0 CON SONARQUBE

# Escenario de Prueba RNP v5

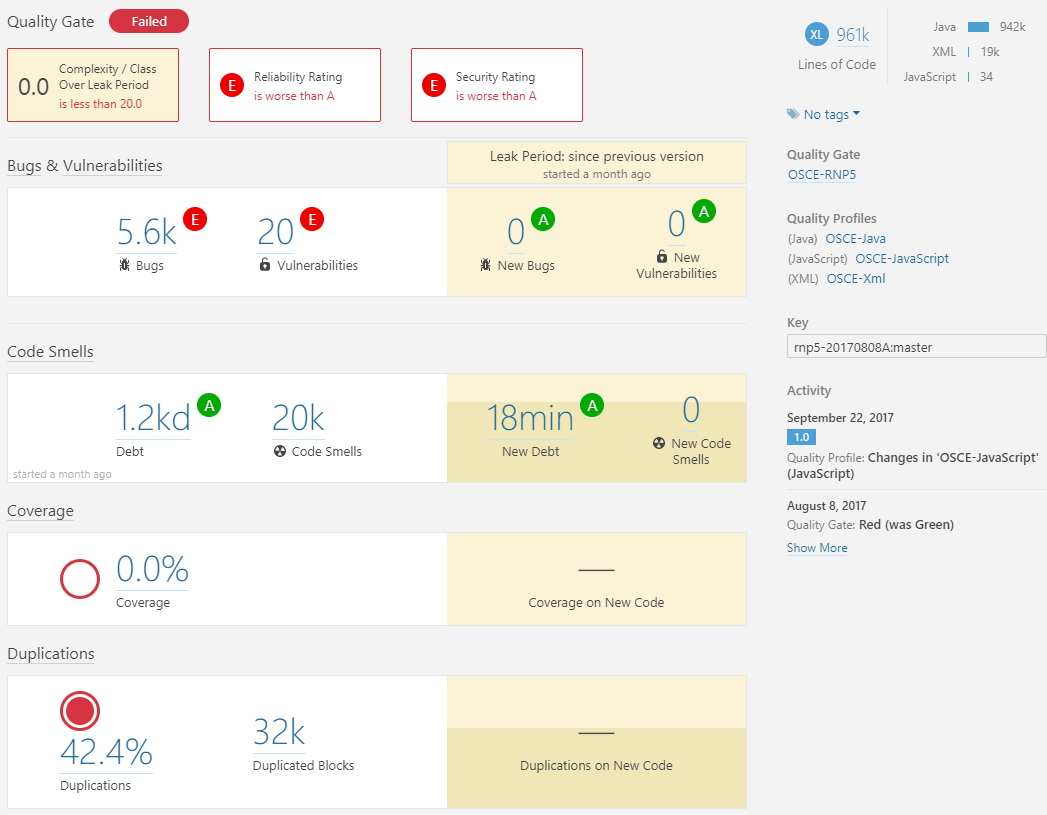
* La prueba se realizó sobre el código fuente que Everis actualizó al 20/12/2016.
* La prueba incluyo el análisis de ficheros del tipo \*.java, \*.javascript y \*.xml.
* Se utilizó la puerta de calidad OSCE-RNP5.
* Se utilizaron los perfiles de calidad: OSCE-Java, OSCE-JavaScript, OSCE-XML.
* La siguiente imagen de SonarQube resume estadística del código analizado durante la prueba:

# Umbrales para RNP v5

El siguiente cuadro contiene una propuesta de umbrales a considerar para el análisis de código estático utilizando SonarQube:

|  |  |  |
| --- | --- | --- |
| **MEDIDA** | **VALOR ESPERADO** | **COMENTARIO** |
| Complejidad ciclomática media por clase | **debe ser menor igual a 20** | Se acepta el riesgo moderado por la complejidad del RNP v5. |
| Deuda técnica permitida | La deuda técnica máxima permitida es de **180 días** | Se asume el uso de 06 meses de mesa de ayuda. |
| Cantidad permitida de defectos del tipo Bug | 0 | Este tipo de defectos afecta la confiabilidad del sistema |
| Cantidad permitida de defectos del tipo vulnerabilidad | 0 | Este tipo de defectos afecta la seguridad del sistema |

# Cuadro de Mandos del resultado



De los valores mostrados en el cuadro de mando se interpreta:

* Existen aproximadamente 5,600 evidencias de tipo bug.
* Existen 20 evidencias de tipo vulnerabilidades.
* Existen aproximadamente 20,000 evidencias de tipo “código que huele mal” que genera una **deuda técnica aproximada de 1,200 días**.
* Existen aproximadamente 32,000 bloques de líneas duplicadas.

Accediendo a los enlaces que proporciona el cuadro de mando se puede obtener el siguiente detalle:

* Las evidencias de tipo “bug” tienen la siguiente información

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TOTAL** **OCURRENCIAS:** | | 5,591 | **CALIFICACIÓN DE** **CONFIABLIDAD:** | E |
| **SEVERIDAD** | Bloqueante | 107 | **ESFUERZO PARA REMEDIAR CONFIABILIDAD:** | 164 días |
| Critica | 0 |
| Mayor | 5,484 |

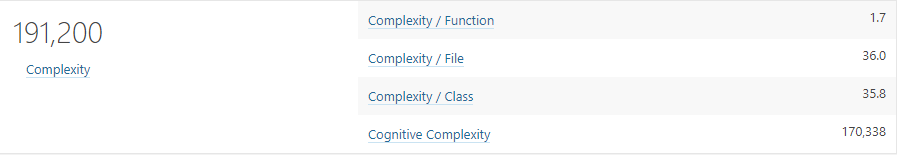
* Las evidencias de tipo “vulnerabilidad” tienen la siguiente información

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TOTAL OCURRENCIAS:** | | 20 | **CALIFICACIÓN DE SEGURIDAD:** | E |
| **SEVERIDAD** | Bloqueante | 10 | **ESFUERZO PARA REMEDIAR SEGURIDAD:** | 3 días |
| Critica | 10 |
| Mayor | 0 |

* Las evidencias de tipo “código que huele mal” tienen la siguiente información

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TOTAL OCURRENCIAS:** | | 19,666 | **CALIFICACIÓN DE MANTENIBILIDAD:** | B |
| **SEVERIDAD** | Bloqueante | 80 | **DEUDA TÉCNICA:** | 1,166 días |
| Critica | 8,040 |
| Mayor | 11,546 |

* Las evidencias de tipo “código duplicado” indica que el 42.4% de líneas es duplicado
* La complejidad ciclomática media por clase se encuentra por encima del umbral establecido.

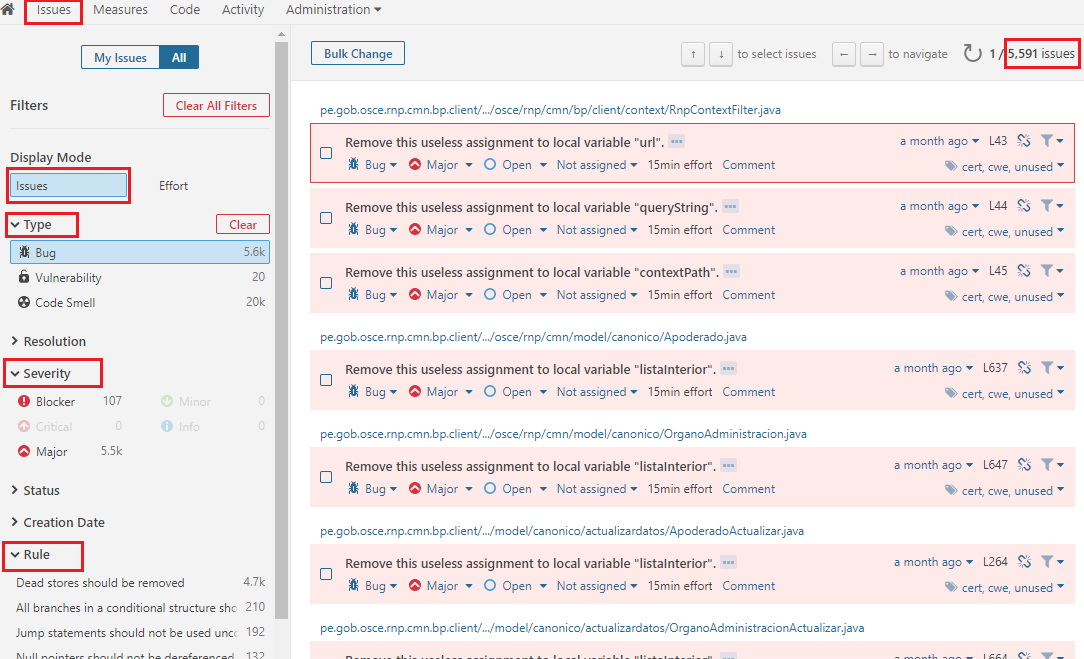


# Estadística de Evidencias en RNP v5

La información relacionada a las evidencias encontradas en los proyectos del RNP v5 han sido obtenidos al consultar la opción “Issues” del proyecto denominado para el análisis “rnp5-20170808A” y haciendo uso de los filtros de:

* **Type:** Bug, Vulnerability, Code Smell
* **Severity:** Blocker, Critical, Major, Minor, Info
* **Rule:** *lista de reglas*

La imagen siguiente contiene cuadros de borde rojo donde se indican alguna de las opciones utilizadas para visualizar la información de “evidencias”.



# BUG’s

|  |  |  |
| --- | --- | --- |
| **SEVERIDAD** | **REGLA** **INCUMPLIDA** | **N° DE OCURRENCIAS** |
| **bloqueante** | Recursos deben ser cerrados | 107 |
| **mayor** | Almacenamiento no usado debe ser removido | 4664 |
| **mayor** | Todas las ramas de una estructura condicional tienen exactamente la misma implementación | 210 |
| **mayor** | Salto de sentencia no de ser usada sin condiciones | 192 |
| **mayor** | Puntos “Null“ no deben ser referenciados | 132 |
| **mayor** | Atributos estáticos no debe ser modificados en constructor | 78 |
| **mayor** | Números con punto flotante no debe ser testeadas con igualdad | 66 |
| **mayor** | Bloques condicionalmente ejecutados deben ser alcanzables | 43 |
| **mayor** | Expresiones idénticas no debe ser utilizado en ambos lados de un operador binario | 27 |
| **mayor** | Bloques multilineas debe estar encerrado en corchetes | 27 |
| **mayor** | Clases no pueden ser comparada por nombres | 21 |
| **mayor** | “BigDecimal (double)” no debe ser usado | 8 |
| **mayor** | Sentencias “Id/else if” relacionadas no deben tener la misma condición | 7 |
| **mayor** | No se deben hacer verificaciones similares de igualdad | 3 |
| **mayor** | Variables no pueden autoasignarse | 2 |
| **mayor** | Los valores devueltos no se deben ignorar cuando las llamadas de función no tienen efectos secundarios | 2 |
| **TOTAL** | | **5589** |

# VULNERABILIDAD

|  |  |  |
| --- | --- | --- |
| **SEVERIDAD** | **REGLA INCUMPLIDA** | **N° DE OCURRENCIAS** |
| **bloqueante** | Restricciones de seguridad deben estar definidos | 6 |
| **bloqueante** | Credenciales no deben ser hardcore | 4 |
| **Crítica** | Pseudo generadores de números aleatorios (PRNGs) no debe ser usado en secure context | 6 |
| **Crítica** | Clases no deben ser cargadas dinámicamente | 4 |
| **TOTAL** | | **20** |

# CODIGO QUE HUELE MAL

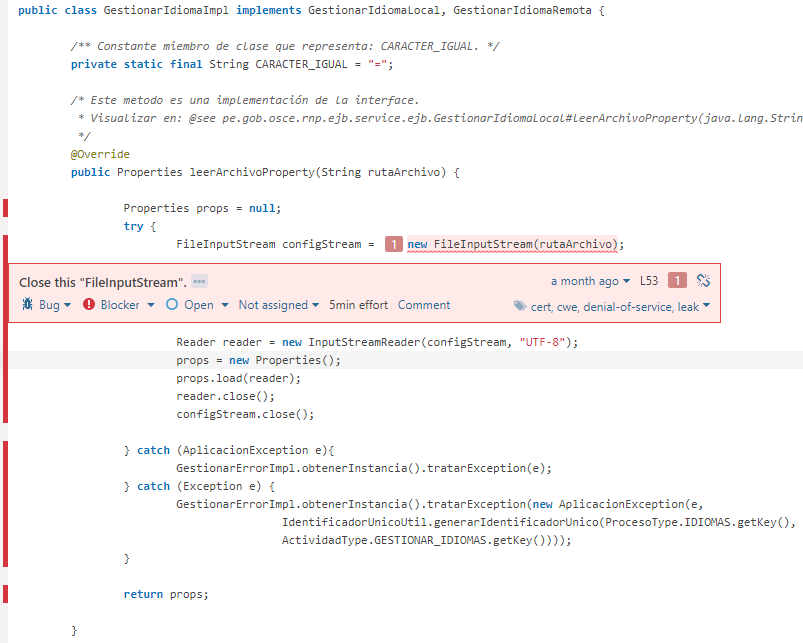
| **SEVERIDAD** | **REGLA INCUMPLIDA** | **N° DE OCURRENCIAS** |
| --- | --- | --- |
| **bloqueante** | Nombres de métodos y atributos no deben ser los mismos o diferentes solo por Capitalization | 60 |
| **bloqueante** | Switch cases debe terminar con un incondicional “break” | 11 |
| **bloqueante** | “Clone” no debe ser sobrescrito | 6 |
| **bloqueante** | Lógica de corto-circuito debe ser usado en contexto booleano | 3 |
| **Crítica** | Literal String no debe ser duplicado | 3047 |
| **Crítica** | Complejidad cognitiva posible de reducir | 2401 |
| **Crítica** | Métodos no deben estar vacíos | 1898 |
| **Crítica** | Nombre de constantes de cumplir con convención | 475 |
| **Crítica** | Métodos de instancia de clases no de escribir a atributos estáticos | 115 |
| **Crítica** | Sentencias “switch” debe finalizar con “default” | 99 |
| **Crítica** | No debe retornarse “Null“ en método donde se espera como salida Boolean | 2 |
| **Crítica** | Debe usarse Try-with-resources | 2 |
| **Crítica** | Constantes no debe ser definida en interfaces | 1 |
| **mayor** | Sección de código comentado inadecuado que no ha sido removido (la mayoría código descartar) | 3,193 |
| **mayor** | Bloques de código duplicado | **2299** |
| **mayor** | Bloques de condiciones que pueden unirse | 2226 |
| **mayor** | Variables locales con nombre de atributos | 688 |
| **mayor** | Constructores y métodos con demasiados parámetros | 471 |
| **mayor** | Condiciones que siempre son verdaderas | 409 |
| **mayor** | Excepciones genéricas no debe usarse para lanzar mensaje personalizado | 384 |
| **mayor** | Atributos de clases no utilizadas | 353 |
| **mayor** | Nombre de atributo con igual nombre de clase | 220 |
| **mayor** | Bloques de condicionales vacíos | 183 |
| **mayor** | Uso de constructos inadecuados para clases del tipo String, Boolean, Double, etc | 178 |
| **mayor** | Uso de clases sincronizadas | 155 |
| **mayor** | Dos ramas e un estructura condicional | 115 |
| **mayor** | Retorno de collections nulas | 92 |
| **mayor** | Parámetros de métodos no usados | 82 |
| **TOTAL** | | **19,168** |

# 2.3 EJEMPLOS DE EVIDENCIAS EN RNP V5

Los siguientes son algunos ejemplos de evidencias del incumplimiento de reglas de codificación de SonarQube encontradas en el proyecto de RNP v5.

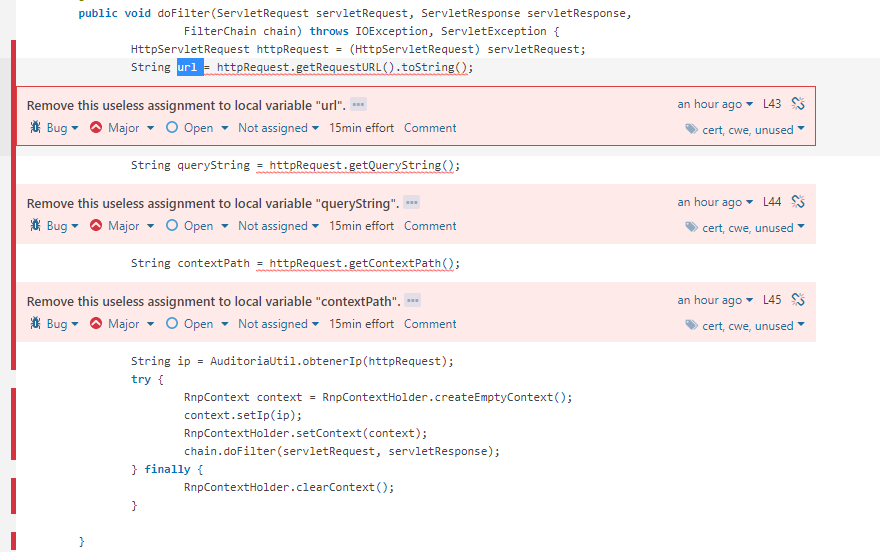
# BUG’s

* + **Regla incumplida:** Recursos deben ser cerrados



* + **Regla incumplida:** Almacenamiento no usado debe ser removido (Variables no usadas);

**public** **class** RnpContextFilter **implements** Filter {



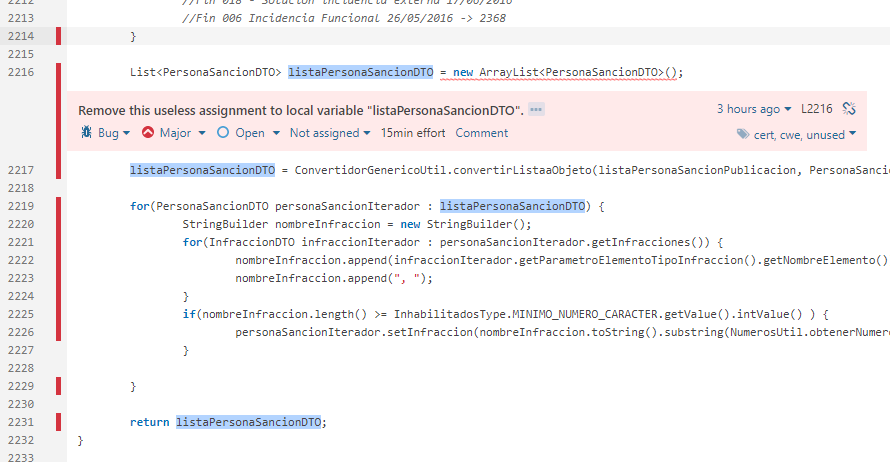
* + **Regla incumplida:** Atributos estáticos no debe ser modificados en constructor



* + **Regla incumplida:** Clases no pueden ser comparada por nombres

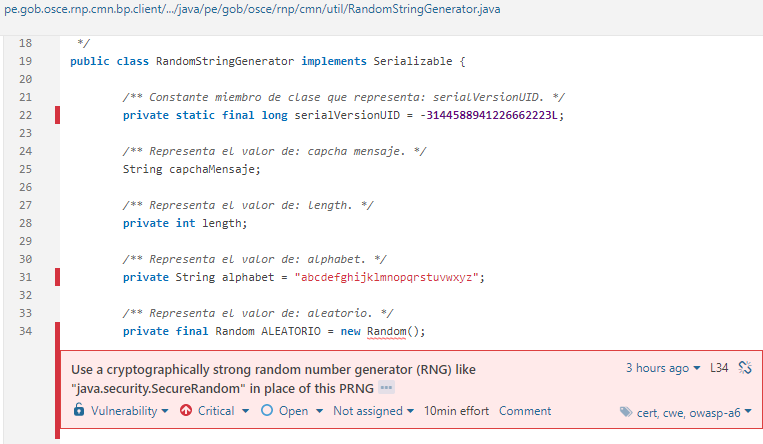


* + **Regla incumplida:** Almacenamiento no usado debe ser removido



# VULNERABILIDAD

* + **Regla incumplida:** Pseudo generadores de números aleatorios (PRNGs) no debe ser usado en secure context



* + **Regla incumplida:** Clases no deben ser cargadas dinámicamente

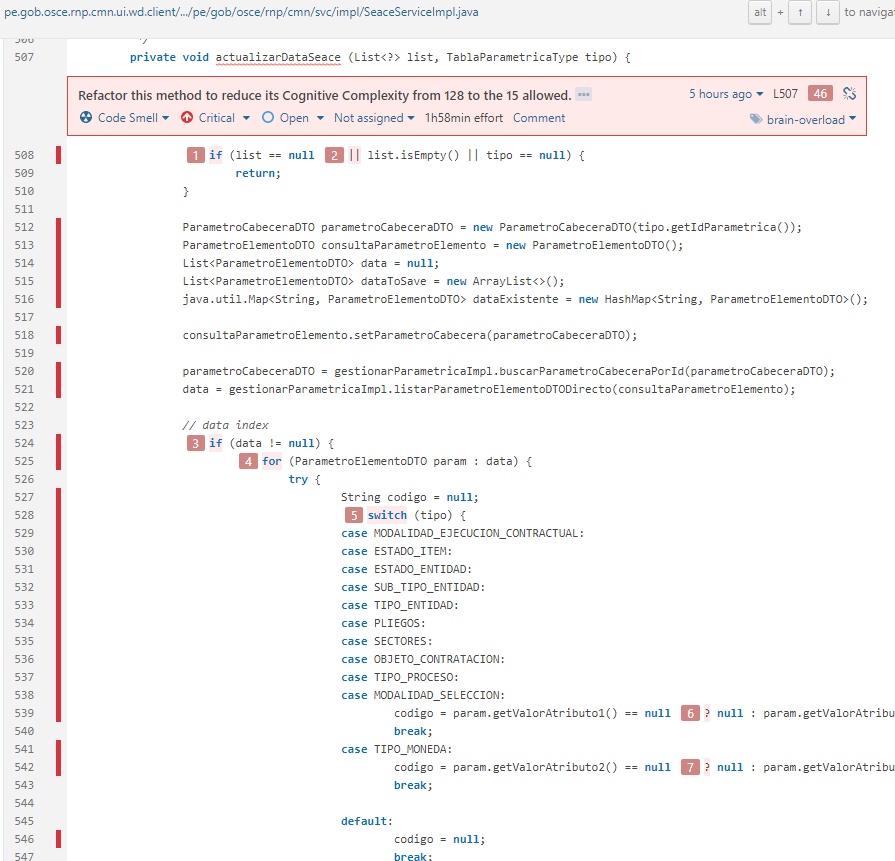


# CODIGO QUE HUELE MAL

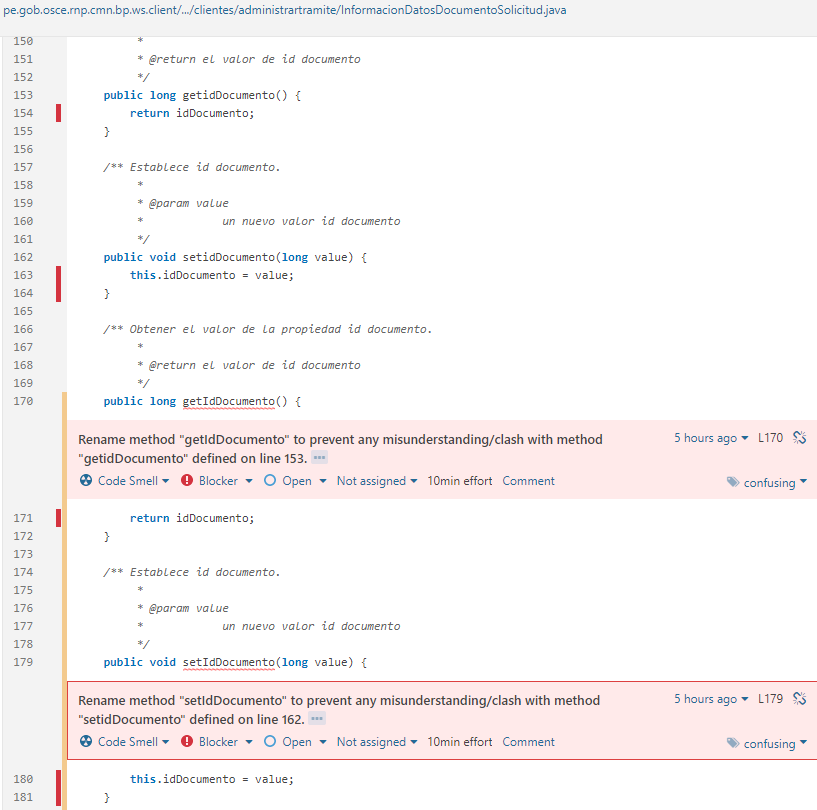
* + **Regla incumplida:** Nombre de constantes de cumplir con convención



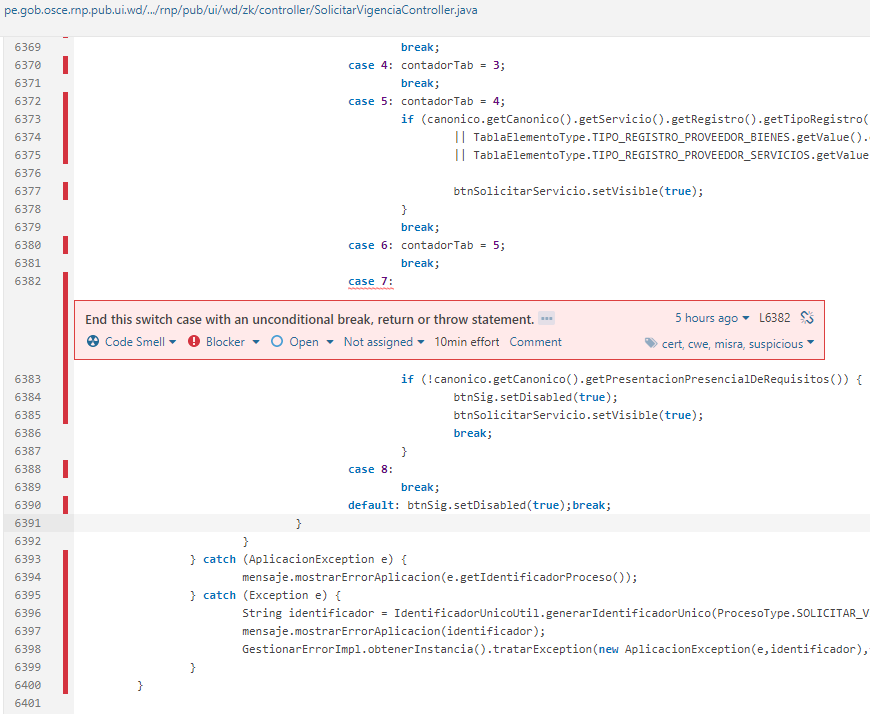
* + **Regla incumplida:** Complejidad cognitiva posible de reducir.



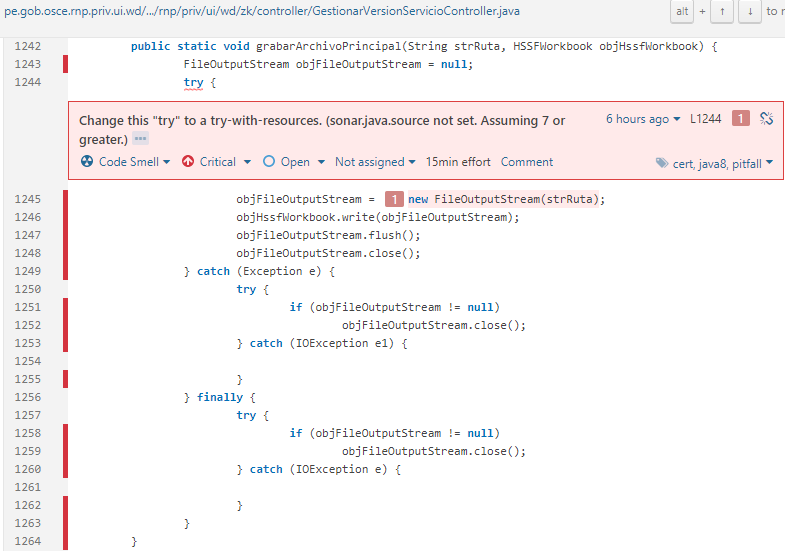
* + **Regla incumplida:** Nombres de métodos y atributos no deben ser los mismos o diferentes solo por Capitalization



* + **Regla incumplida:** Switch cases debe terminar con un incondicional “break”: Case con “salida” ambigua



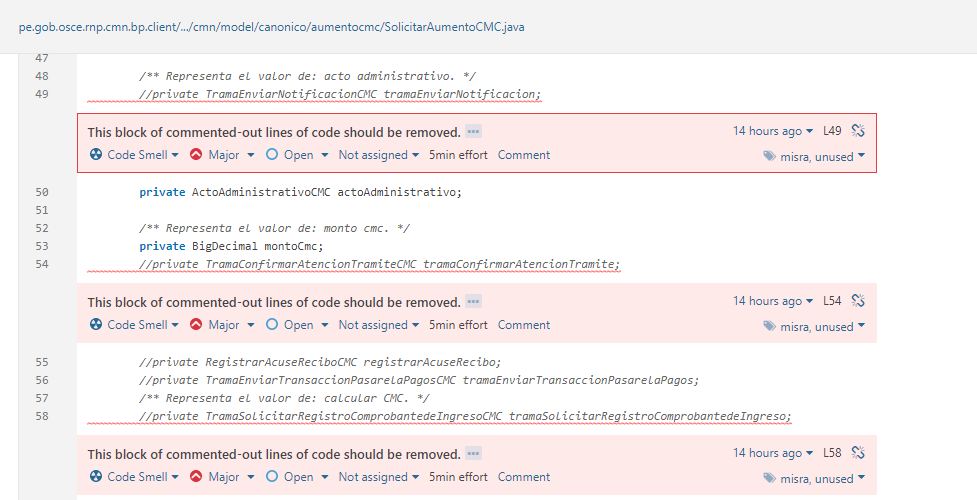
* + **Regla incumplida:** Debe usarse Try-with-resources

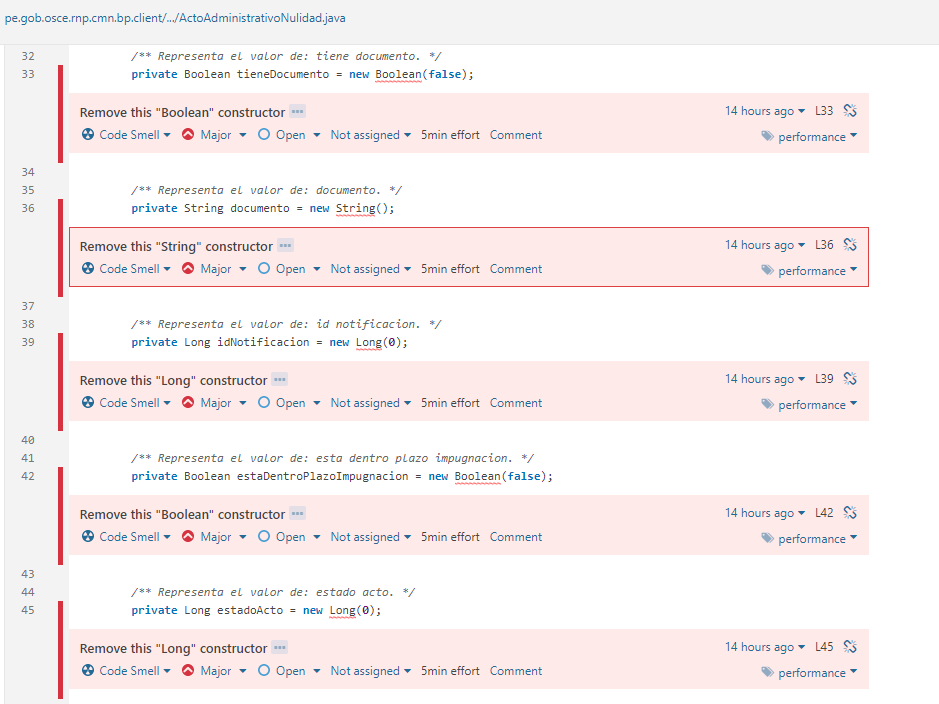


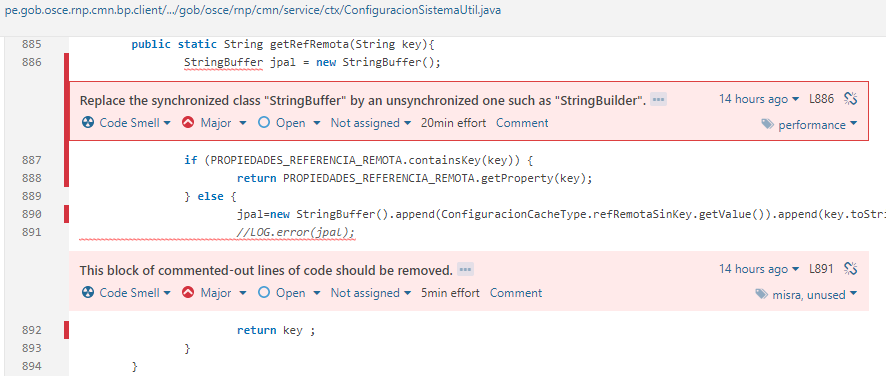
* + **Regla incumplida:** Literal String no debe ser duplicado



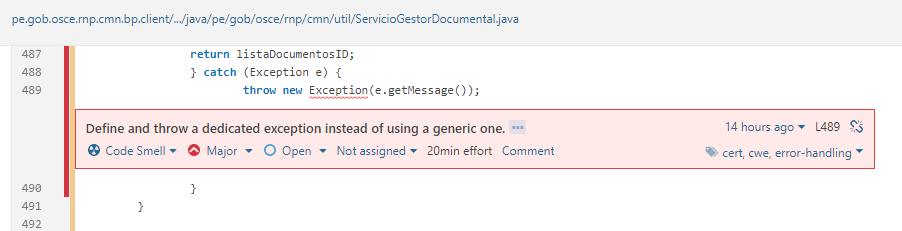
* + **Regla incumplida:** Sección de código comentado inadecuado que no ha sido removido



* + **Regla incumplida:** Uso de constructos inadecuados para clases del tipo String, Boolean, Double
  + **Regla incumplida:** Uso de clases sincronizadas



* + **Regla incumplida:** Excepciones genéricas no debe usarse para lanzar mensaje personalizado



* + **Regla incumplida:** Condiciones que siempre son verdaderas



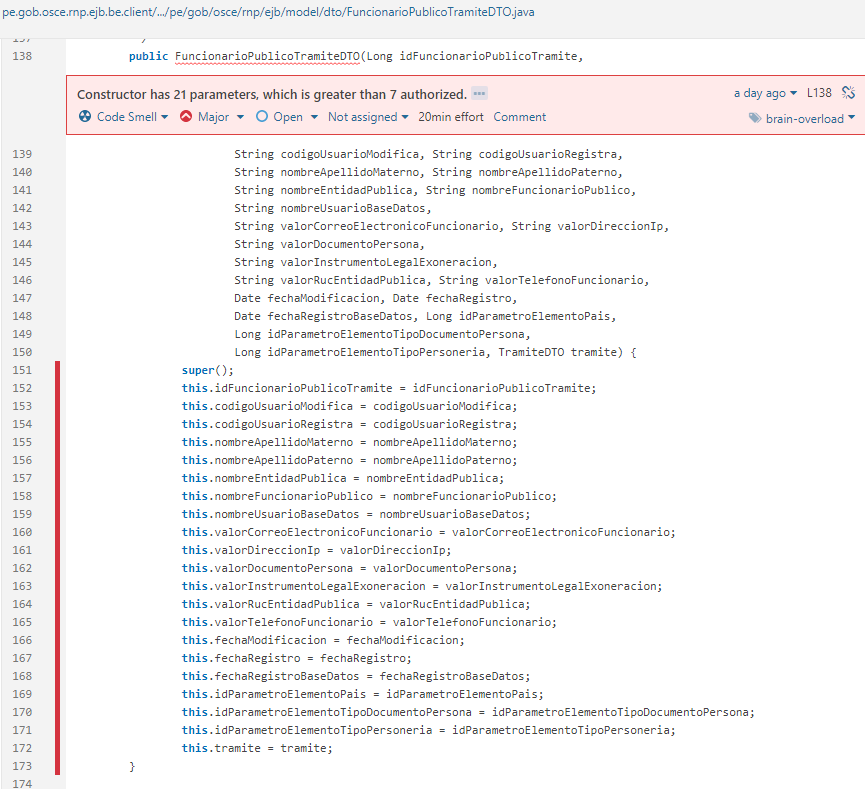
* + **Regla incumplida:** Nombres de métodos y atributos no deben ser los mismos o diferentes solo por Capitalization



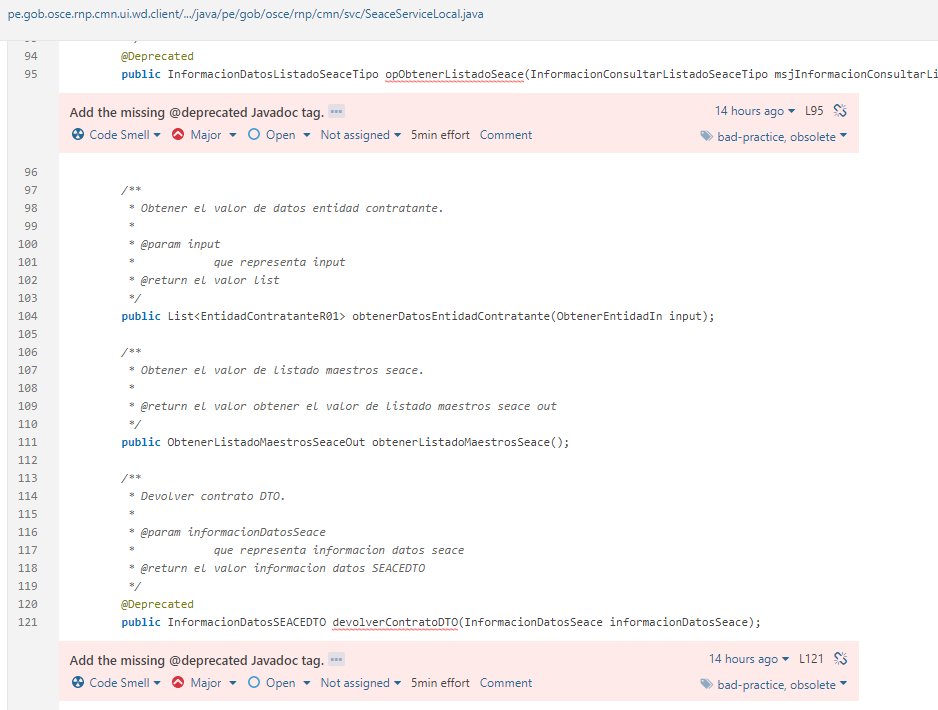
* + **Regla incumplida:** Bloques de condicionales vacíos*.*



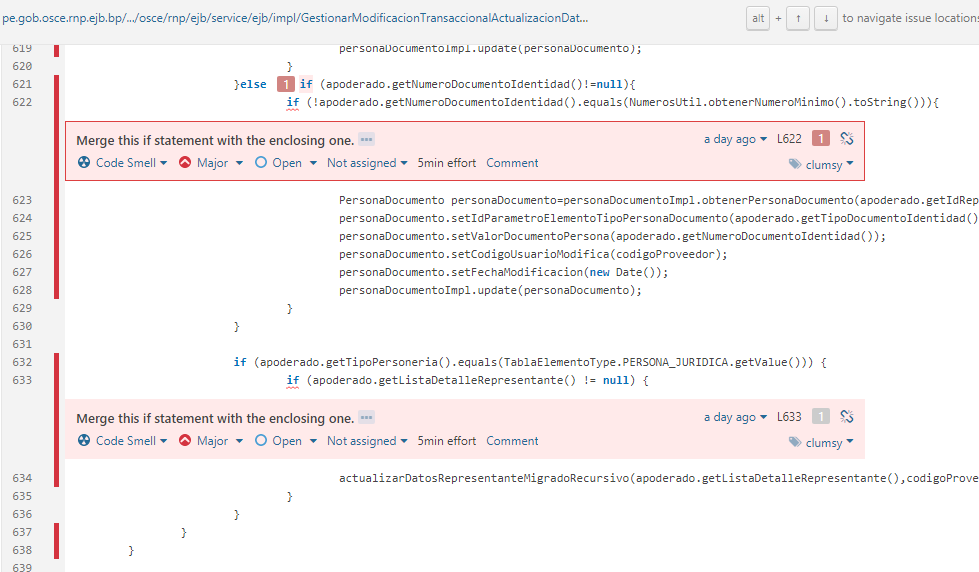
* + **Regla incumplida:** Constructores y métodos con demasiados parámetros



* + **Regla incumplida:** Existencia de métodos marcados con deprecados



* + **Regla incumplida:** Bloques de condiciones que pueden unirse.



# CONCLUSIONES

* La última versión de la actualización al código fuente remitida por el Consorcio es la que figura en el correo electrónico de fecha 20/12/2016, el cual fue remitido al personal de la Unidad de Gestión de Desarrollo de Software.
* El código del Sistema RNP v. 5.0 recibido y sus actualizaciones presentan deficiencias técnicas, las cuales han sido observadas y reportadas en su momento.
* De acuerdo al resultado del análisis estático de la última actualización del código fuente remitida por el Consorcio Everis, la deuda técnica máxima permitida es de 180 días, tiempo que indica en el Contrato N° 062-2013-OSCE para el período de Mesa de Ayuda.
* El Consorcio Everis no ha remitido la subsanación de las observaciones técnicas remitidas por la Oficina de Tecnologías de la Información a la fecha.

# RECOMENDACIONES

* Se recomienda realizar un estudio costo/beneficio de las acciones necesarias para cerrar la brecha a cubrir para contar con un Sistema RNP versión 5.0 listo para desplegar en el ambiente de producción para el inicio de su uso por parte de los funcionarios de OSCE, proveedores, entidades públicas y ciudadanía en general.

Se informa lo anteriormente señalado, para su conocimiento y fines.

Atentamente,

**LIZETTE FABIOLA RIVERA VASQUEZ**

**Supervisora de la Unidad de Gestión de Desarrollo de Software**

1. http://www.sei.cmu.edu/reports/97hb001.pdf [↑](#footnote-ref-1)