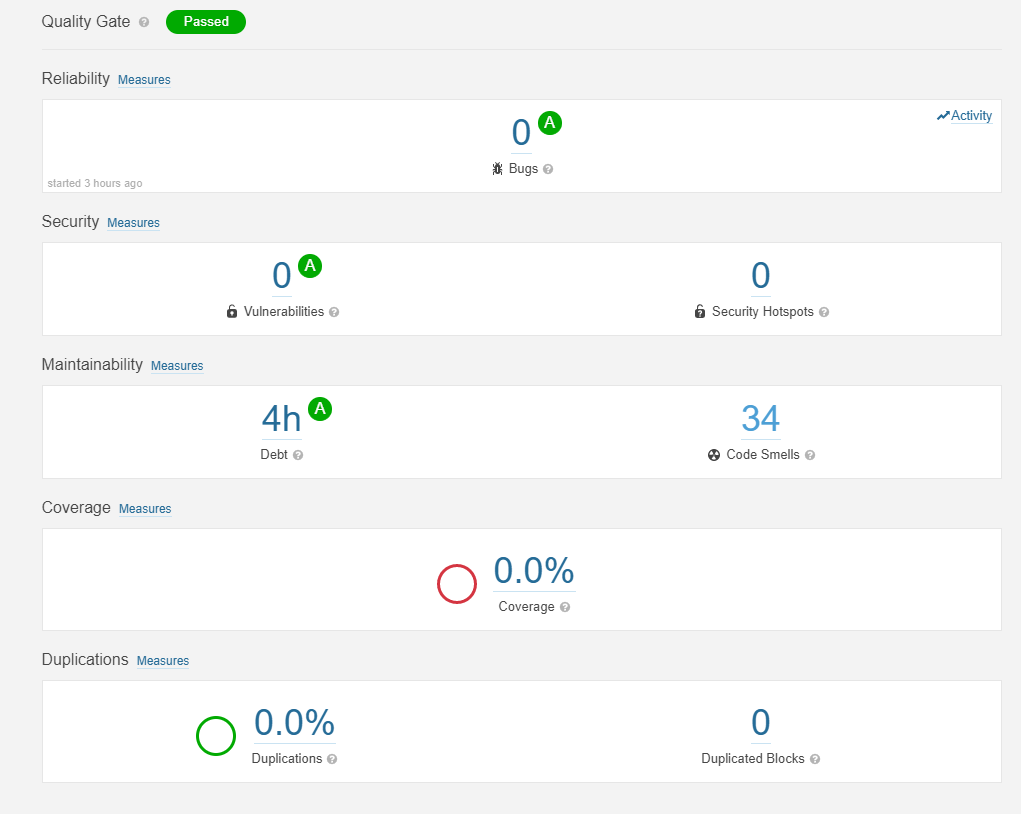
Código analisado com a API Sonarqube em 29/10/2019

**Análise geral do código**

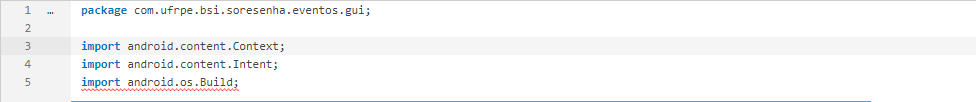


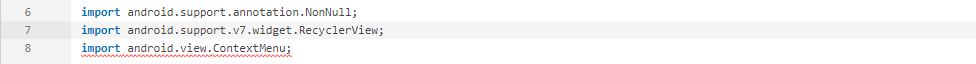
**Code Smells presentes no código**

**Erros sublinhados de vermelho**

A parte de importações de um arquivo deve ser manipulada pelo IDE, não manualmente.

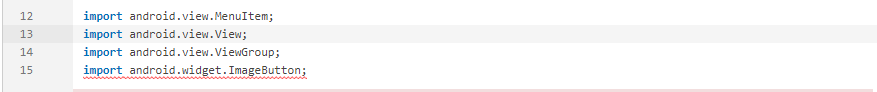
Importações não utilizadas e inúteis não devem ocorrer, se for o caso. Deixá-los reduz a legibilidade do código, pois a presença deles pode ser confusa.





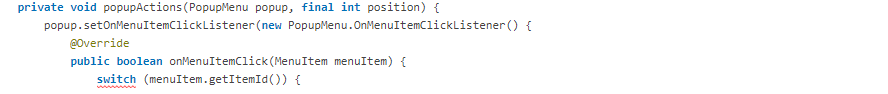






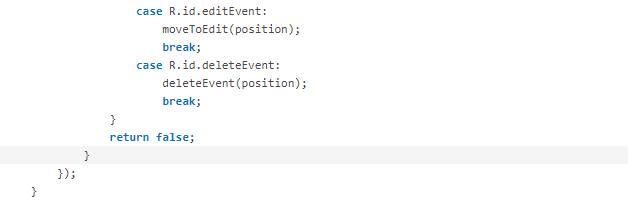


O requisito para um default é a programação defensiva. O método deve tomar as medidas apropriadas ou conter um comentário adequado sobre o motivo pelo qual nenhuma ação foi tomada.



As instruções switch são úteis quando existem muitos casos diferentes, dependendo do valor da mesma expressão.

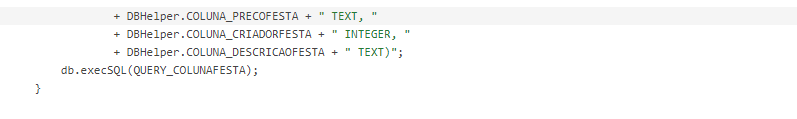
Porém, apenas para um ou dois casos, o código ficará mais legível com as instruções if.



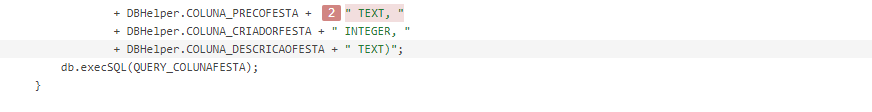
Strings literais duplicadas tornam o processo de refatoração propenso a erros, pois se deve atualizar todas as ocorrências.

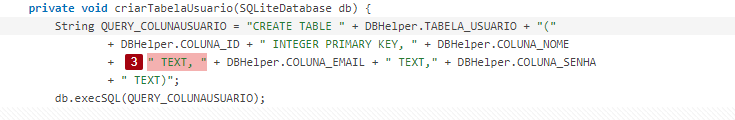
Por outro lado, as constantes podem ser referenciadas de muitos lugares, mas precisam ser atualizadas apenas em um único local.



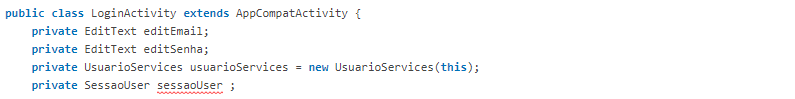




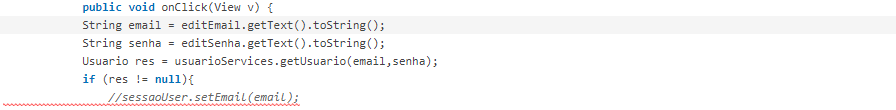




Se um campo privado for declarado, mas não usado no programa, ele poderá ser considerado código morto e, portanto, deverá ser removido. Isso melhora a capacidade de manutenção porque os desenvolvedores não se perguntarão para que serve a variável.



Os programadores não devem comentar o código, pois ele incha os programas e reduz a legibilidade.O código não utilizado deve ser excluído e pode ser recuperado do histórico de controle de origem, se necessário.

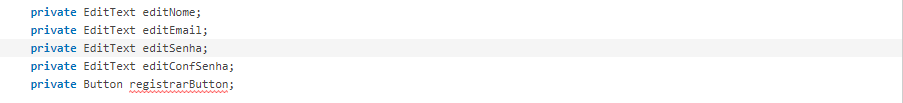


Se um campo privado for declarado, mas não usado no programa, ele poderá ser considerado código morto e, portanto, deverá ser removido. Isso melhora a capacidade de manutenção porque os desenvolvedores não se perguntarão para que serve a variável.

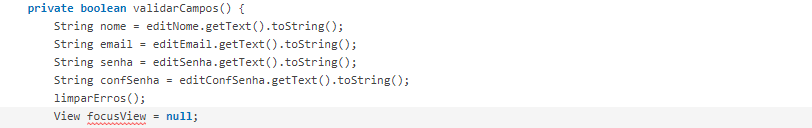
Quando o valor de um campo privado é sempre atribuído aos métodos de uma classe antes de ser lido, ele não está sendo usado para armazenar informações da classe. Portanto, ela deve se tornar uma variável local nos métodos relevantes para evitar qualquer mal-entendido.







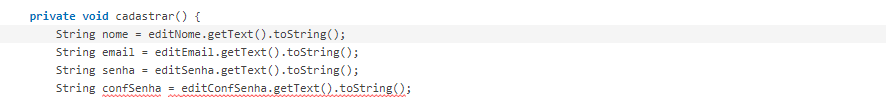
Se uma variável local for declarada mas não usada, será um código morto e deve ser removida. Isso aumentará a capacidade de manutenção, pois os desenvolvedores não se perguntarão para que serve a variável.







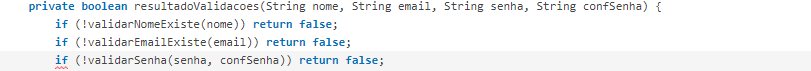




O retorno de instruções literais booleanas agrupadas em instruções if-then-else deve ser simplificado.

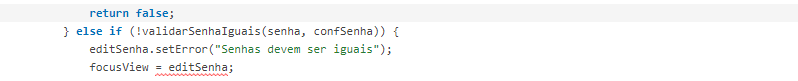
Da mesma forma, as invocações de métodos agrupadas em if-then-else diferentes apenas de literais booleanos devem ser simplificadas em uma única invocação.

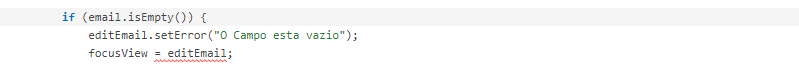


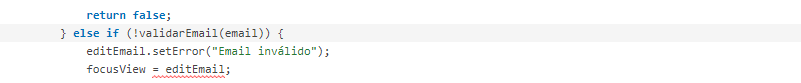


Um armazenamento morto ocorre quando uma variável local recebe um valor que não é lido por nenhuma instrução subsequente. Calcular ou recuperar um valor apenas para substituí-lo ou jogá-lo fora pode indicar um erro grave no código. Mesmo que não seja um erro, é na melhor das hipóteses um desperdício de recursos. Portanto, todos os valores calculados devem ser utilizados.

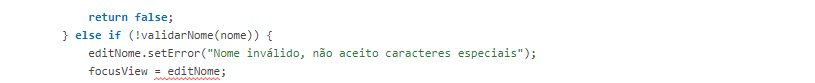












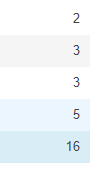
**Complexidade Ciclomática**

|  |  |
| --- | --- |
| **Complexidade** | **Avaliação** |
| 1-10 | Método simples. **Baixo risco**. |
| 11-20 | Método razoavelmente complexo. **Moderado risco**. |
| 21-50 | Método muito complexo. **Elevado risco**. |
| 51-N | Método de **altíssimo risco** e bastante instável. |

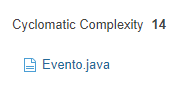
Eventos

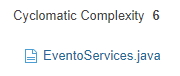
Gui de eventos

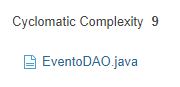
dominio de eventos

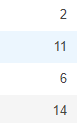
negócio de eventos

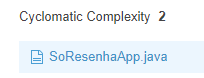
persistência de eventos

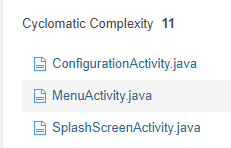
Infra

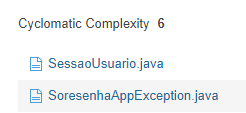
Infra app

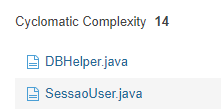
Infra gui

Infra negócio

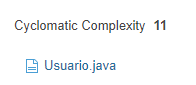
Infra persistência

Usuário

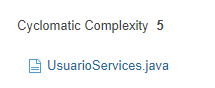
Usuário domínio



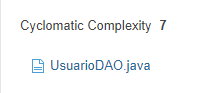
Usuário gui

Usuário negócio

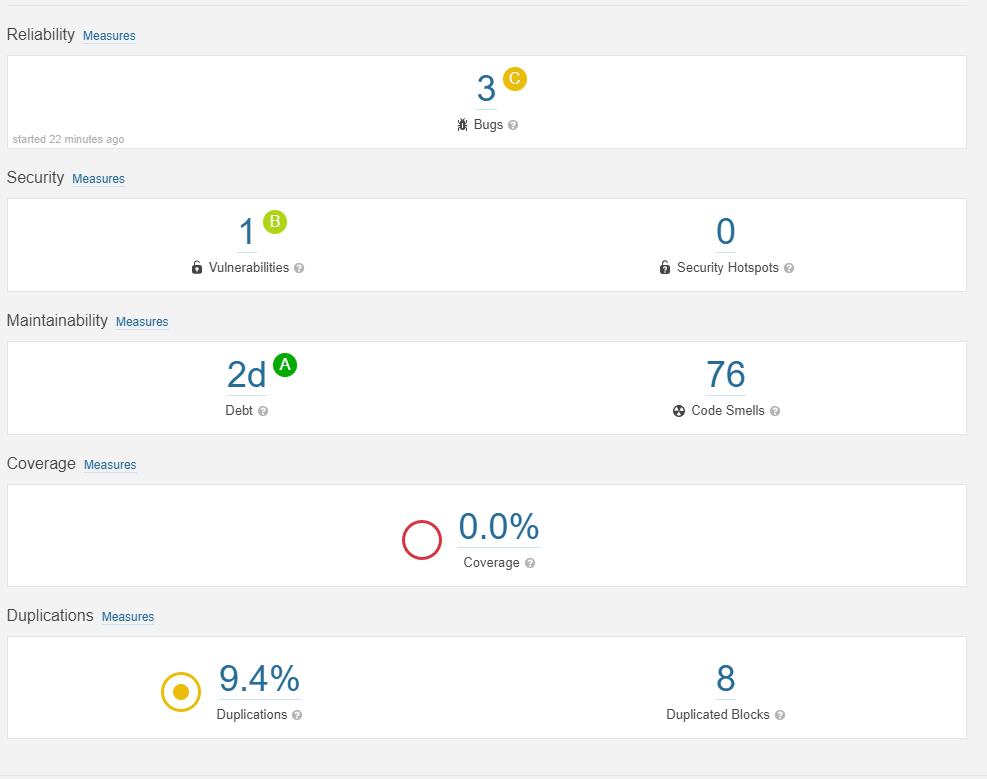
 

Usuário persistência

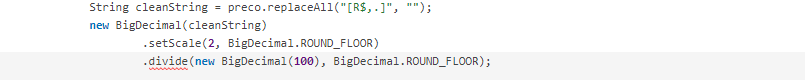
 

Código analisado com a API Sonarqube em 04/11/2019

**Análise geral do código 2.0**



**BUGS**

****

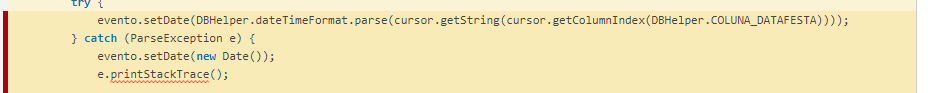
Quando a chamada para uma função não tem efeitos , qual é o sentido de fazer a chamada se os resultados são ignorados? Nesse caso, a chamada da função é inútil e deve ser descartada ou o código-fonte não se comporta conforme o esperado.

Esse mesmo erro foi notificado duas vezes

**Code Smells presentes no código**

**Erros sublinhados de vermelho**

Throwable.printStackTrace (...) imprime um Throwable e seu rastreamento de pilha em algum fluxo. Por padrão, esse fluxo System.Erro, que inadvertidamente pode expor informações confidenciais.



O requisito para uma cláusula padrão é a programação defensiva. A cláusula deve tomar as medidas apropriadas ou conter um comentário adequado sobre o motivo pelo qual nenhuma ação foi tomada.

**Mesmo erro notificado três vezes**



Strings literais duplicadas tornam o processo de refatoração propenso a erros, pois você deve atualizar todas as ocorrências.







A atualização correta de um campo estático a partir de um método não estático é difícil de corrigir e pode facilmente levar a erros se houver várias instâncias de classe e / ou vários threads em execução. Idealmente, os campos estáticos são atualizados apenas a partir de métodos estáticos sincronizados.

Esta regra levanta um problema sempre que um campo estático é atualizado a partir de um método não estático.

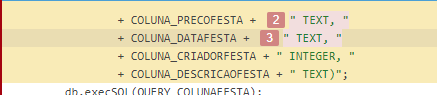


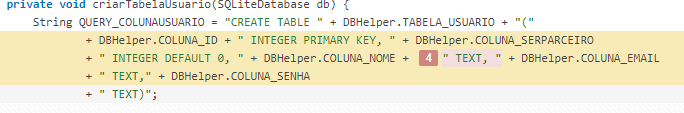




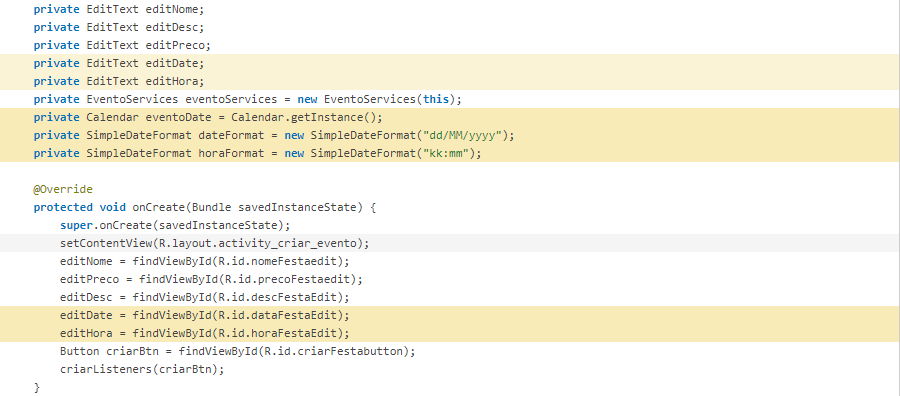
Defina uma constante em vez de duplicar esse "TEXT", 4 vezes

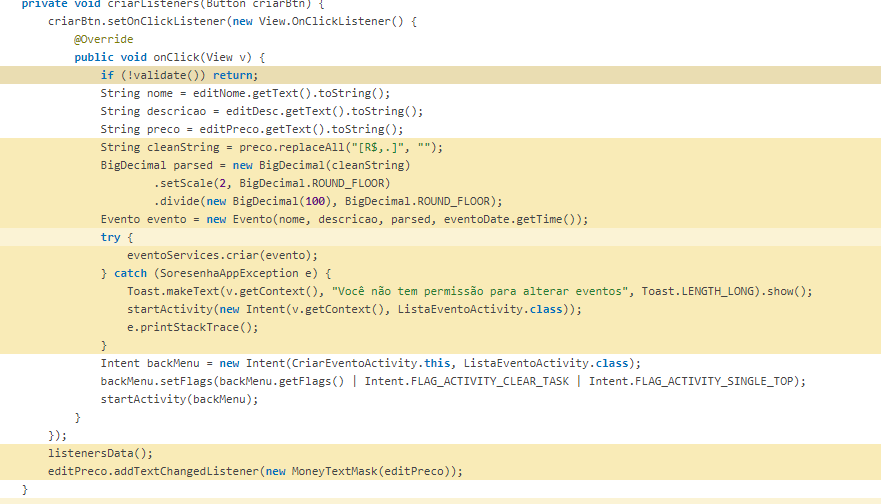




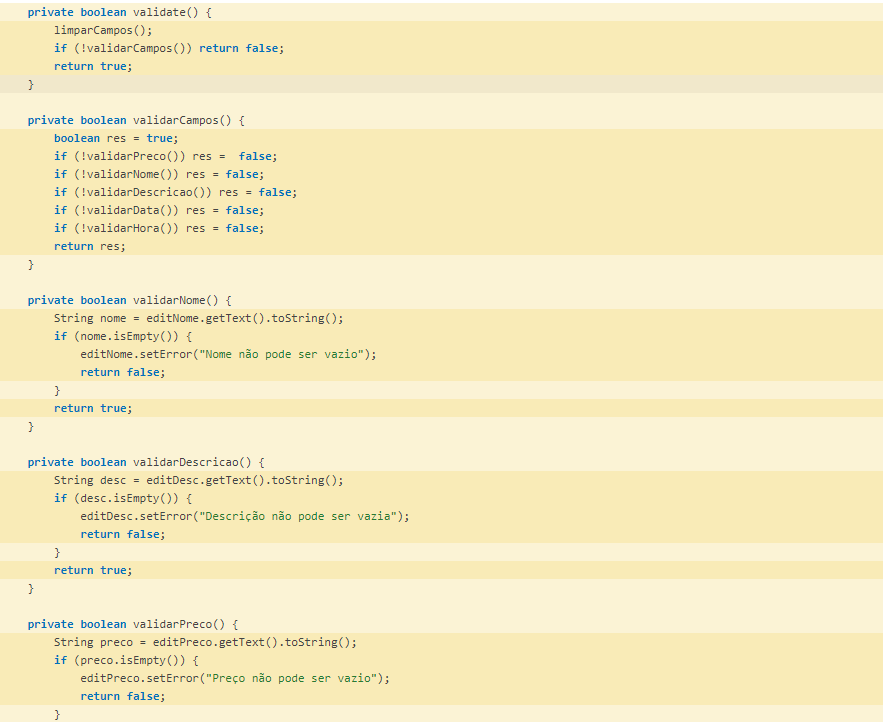


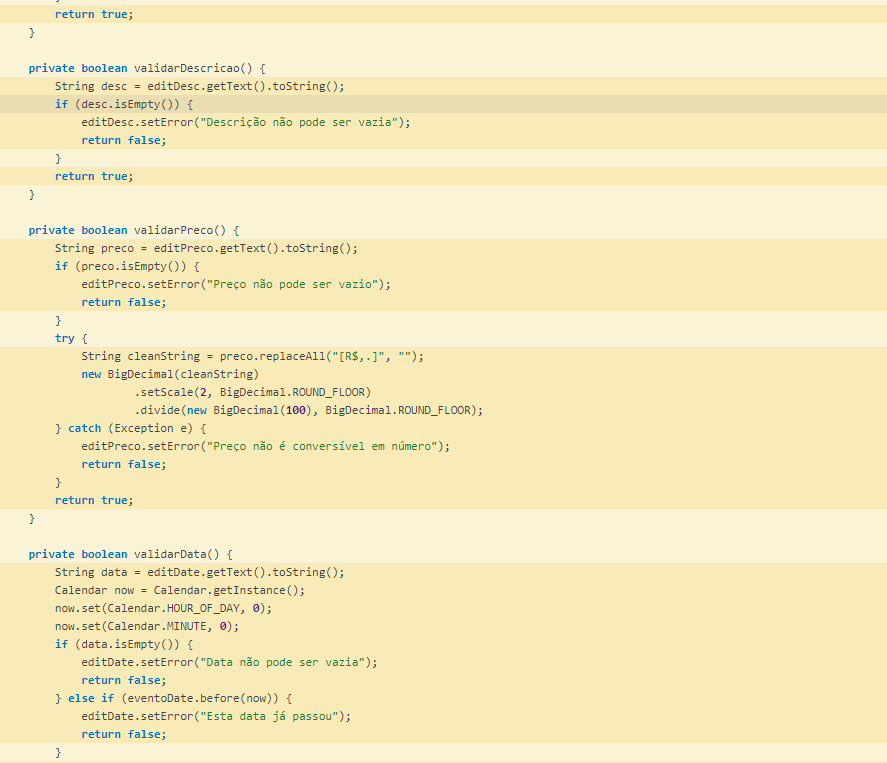
Blocos de código duplicados 2 classes iguais que poderia facilmente ser uma

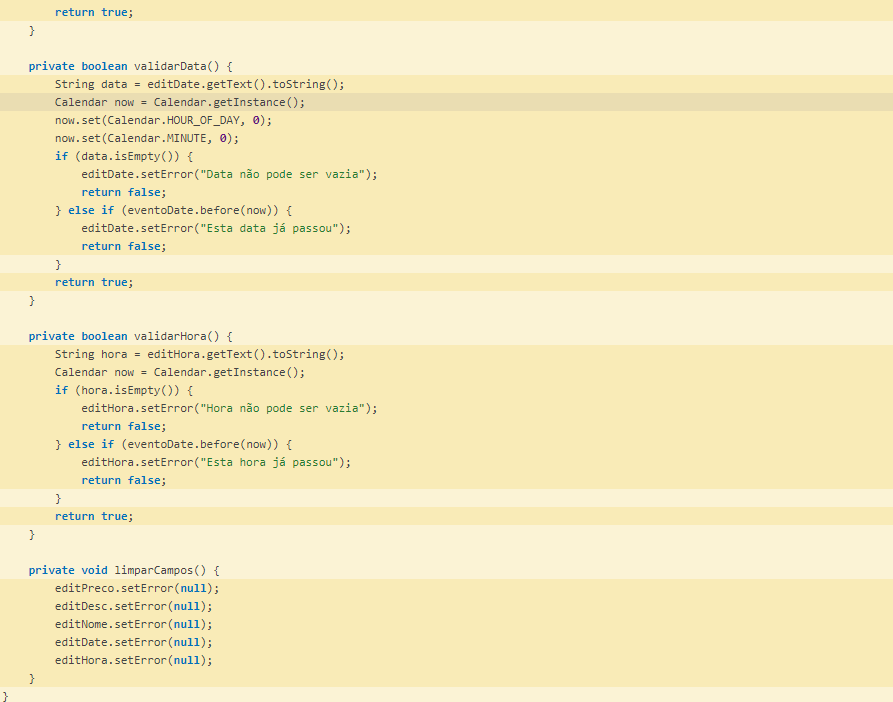












Concatenação desnecessária e que pode impactar no desempenho



Se um campo privado for declarado, mas não usado no programa, ele poderá ser considerado código morto e, portanto, deverá ser removido. Isso melhora a manutenção, pois os desenvolvedores não se perguntarão para que serve a variável.



Parâmetros não utilizados são enganosos. Quaisquer que sejam os valores passados para esses parâmetros, o comportamento será o mesmo.



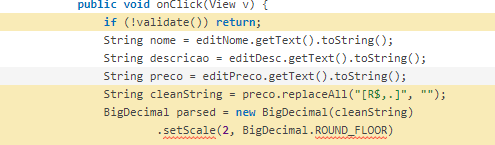
Os programadores não devem comentar o código, pois ele incha os programas e reduz a legibilidade.

O código não utilizado deve ser excluído e pode ser recuperado do histórico de controle de origem, se necessário.





Uma vez obsoleto, classes e interfaces e os seus membros devem ser evitados, ao invés de usado, herdada ou estendida. A descontinuação é um aviso de que a classe ou interface foi substituída e, eventualmente, será removida. O período de suspensão permite que você faça uma transição suave para longe do envelhecimento, logo-a-ser-aposentado tecnologia.





Imports não utilizados







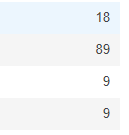
Classe vazia



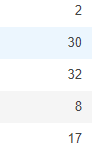
**Complexidade Ciclomática 2.0**

|  |  |
| --- | --- |
| **Complexidade** | **Avaliação** |
| 1-10 | Método simples. **Baixo risco**. |
| 11-20 | Método razoavelmente complexo. **Moderado risco**. |
| 21-50 | Método muito complexo. **Elevado risco**. |
| 51-N | Método de **altíssimo risco** e bastante instável. |

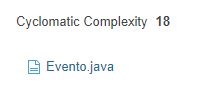
Eventos



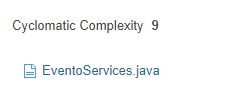
Gui de eventos

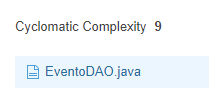
domínio de eventos

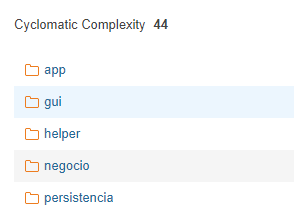
negócio de eventos

persistência de eventos

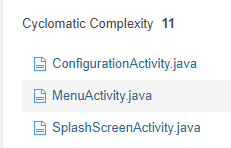
Infra

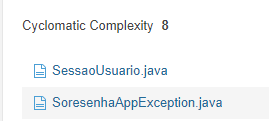
Infra app

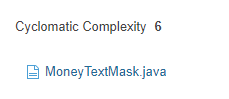
Infra gui

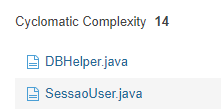
Infra negócio

Infra Helper



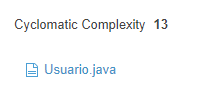
Infra persistência

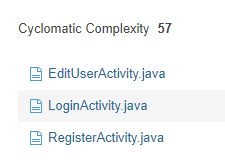
Usuário

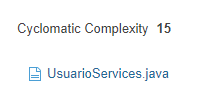
Usuário domínio



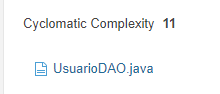
Usuário gui

Usuário negócio

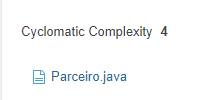
Usuário persistência

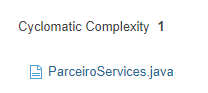
Parceiro



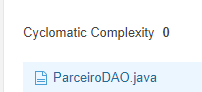
Parceiro domínio



Parceiro Negócio

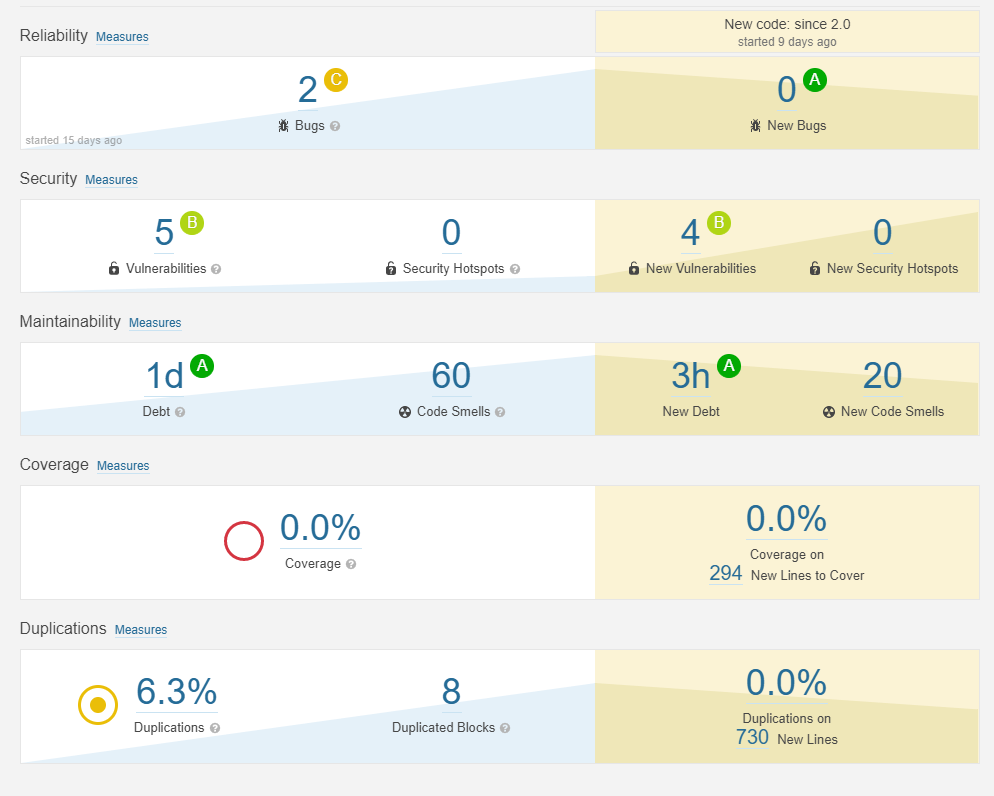


Parceiro persistência



Código analisado com a API Sonarqube em 11/11/2019

**Análise geral do código 3.0**



**Code Smells presentes no código**

**Erros sublinhados de vermelho**

Os code smells anteriores não foram resolvidos

Os valores estão sujeitos a alterações em qualquer parte do código



eventos/dominio/Avaliacao.java os 3 ids e o tipo Avaliação

Campo privado não usado



eventos/negocio/EventoServices.java

Parâmetro não usado



/eventos/persistencia/EventoDAO.java

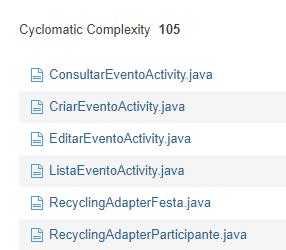
**Complexidade Ciclomática 3.0**

|  |  |
| --- | --- |
| **Complexidade** | **Avaliação** |
| 1-10 | Método simples. **Baixo risco**. |
| 11-20 | Método razoavelmente complexo. **Moderado risco**. |
| 21-50 | Método muito complexo. **Elevado risco**. |
| 51-N | Método de **altíssimo risco** e bastante instável. |

Eventos



Gui de eventos

domínio de eventos

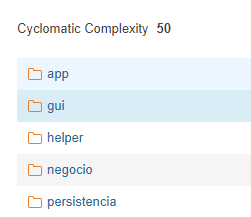
negócio de eventos

persistência de eventos

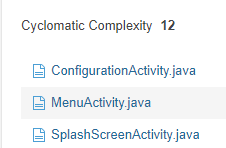
Infra

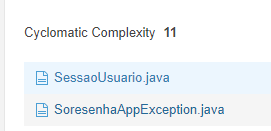
Infra app

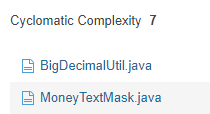
Infra gui

Infra negócio

Infra Helper



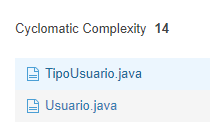
Infra persistência

Usuário

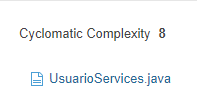
Usuário domínio



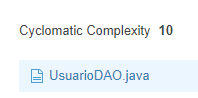
Usuário gui

Usuário negócio

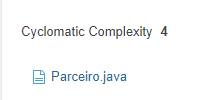
Usuário persistência

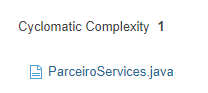
Parceiro



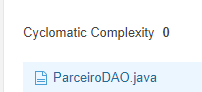
Parceiro domínio



Parceiro Negócio

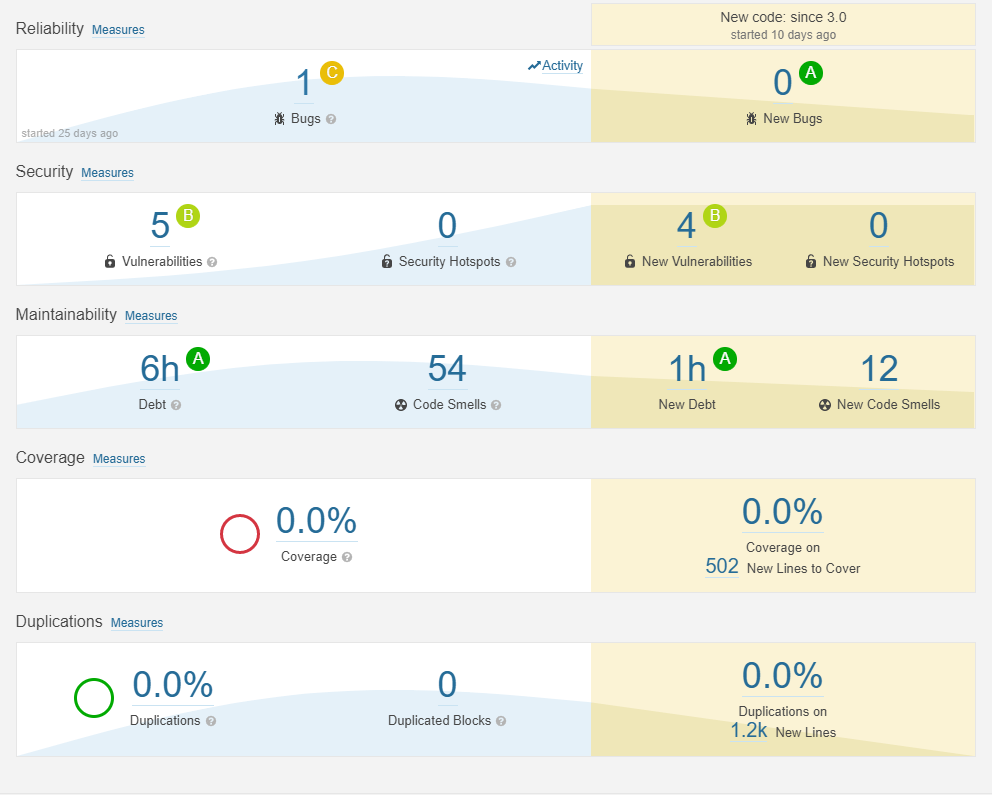


Parceiro persistência



Em suma a gui de eventos está sobrecarregada e deveria ser mais otimizada.

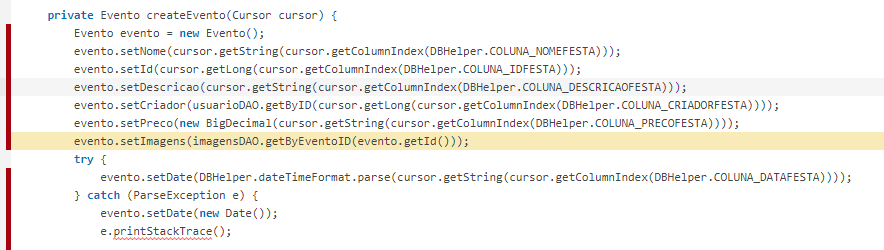
Código analisado com a API Sonarqube em 18/11/2019

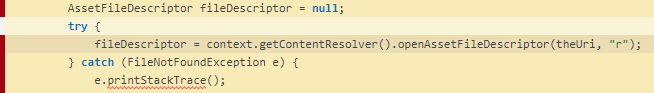


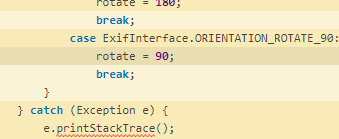
**Code Smells presentes no código**

**Erros sublinhados de vermelho**

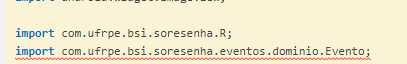
Foram encontrados 3 prints dentro do código esse erro ja vem sendo cometido a 2 versões







Import não utilizado



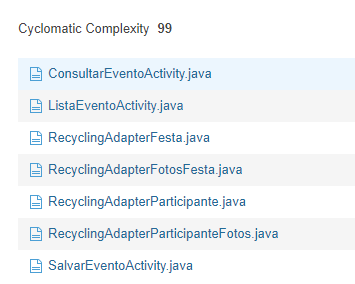
**Complexidade Ciclomática 4.0**

|  |  |
| --- | --- |
| **Complexidade** | **Avaliação** |
| 1-10 | Método simples. **Baixo risco**. |
| 11-20 | Método razoavelmente complexo. **Moderado risco**. |
| 21-50 | Método muito complexo. **Elevado risco**. |
| 51-N | Método de **altíssimo risco** e bastante instável. |

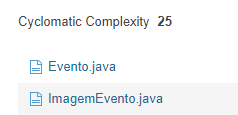
Eventos



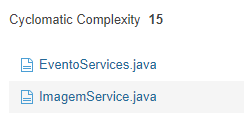
Gui de eventos

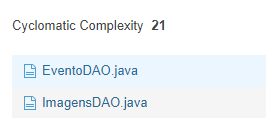
domínio de eventos

negócio de eventos

persistência de eventos

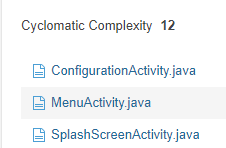
Infra

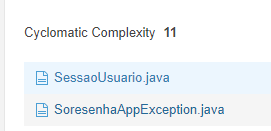
Infra app

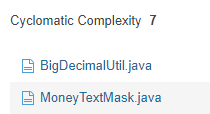
Infra gui

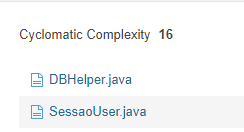
Infra negócio

Infra Helper



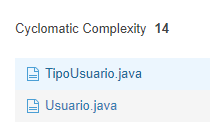
Infra persistência

Usuário

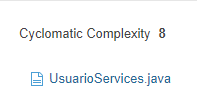
Usuário domínio



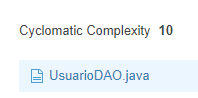
Usuário gui

Usuário negócio

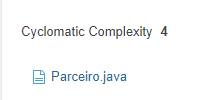
Usuário persistência

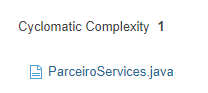
Parceiro



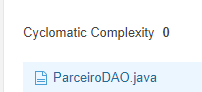
Parceiro domínio



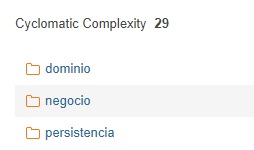
Parceiro Negócio



Parceiro persistência



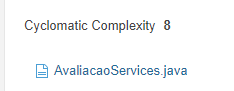
Avaliação



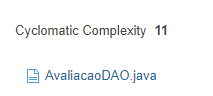
Avaliação domínio



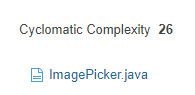
Avaliação negocio



Avaliação persistência



External Image Picker



Por mais que a complexidade tenha caído ainda assim a evento se mantem muito carregada.