

SonarCloud

IS2

2023/2024

Proyecto Bets

Autores:

Unai Artano

Asier Contreras

Martin Ian Horsfield

eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

Introducción

Una vez el proyecto estaba subido a sonar y comprobamos el informe, vimos que el proyecto base no estaba tan mal a lo esperado y tuvimos directamente el “Passed” de Sonar. En cuanto a correcciones, el proyecto no tenía tantos bugs y vulnerabilidades como para llegar a las 5 por persona que solicitaba el proyecto por lo que hemos hecho un reparto de tareas diferentes entre los 3 participantes del grupo. El reparto de tareas ha sido el siguiente:

Asier Contreras:

3 bugs (Reliability - Sonarcloud) + 5 Code smells

Martin Ian Horsfield

2 vulnerabilidades (Security - Sonarcloud) + 5 Code Smells

Unai Artano

11 fallos de seguridad (Security Review - Sonarcloud) + 5 Code Smells

Enlaces

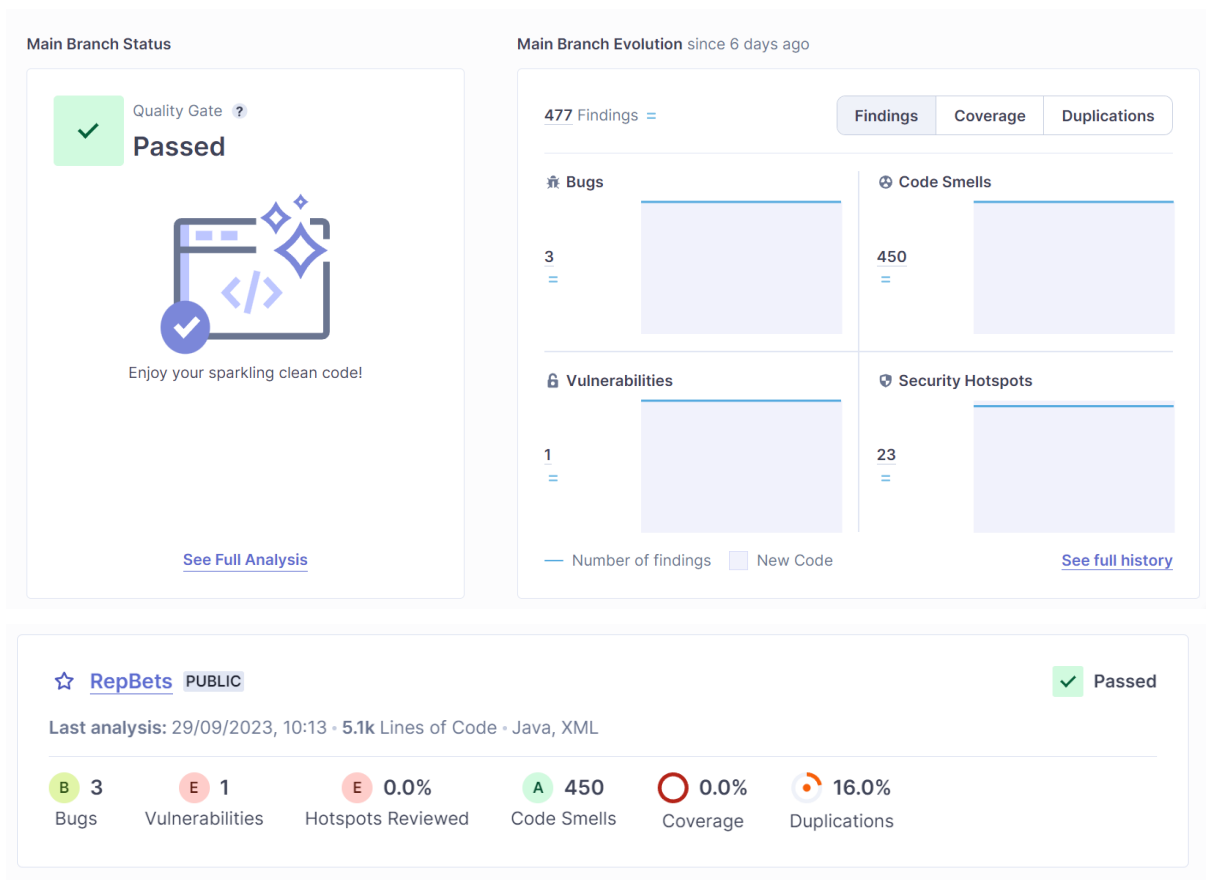
Github:

<https://github.com/asiercontreras/IS2-RepBets>

SonnarCloud:

https://sonarcloud.io/project/overview?id=proyectois2-bets_repbets

Primer informe



Asier Contreras:

Bug 1:

En la clase Bet.java se ha tenido que crear el método equals(), y según la especificación de java, cuando se modifica un equals también hay que modificar el hashCode. Por lo que la solución a este bug ha sido, simplemente, crear el método hashCode() necesario para la clase Bet.

src/main/java/domain/Bet.java

☐ Intentionality issue

[This class overrides "equals\(\)" and should therefore also override "hashCode\(\)".](#) cert cwe +

☐ Open ☐ Not assigned ☐ Reliability ☐ Bug ☐ Minor 15min effort • 12 days ago

Solución:

```
@Override
public int hashCode() {
    return Objects.hash(bet, fr, id, usr);
}
```

Bug 2:

Este bug era igual al anterior pero en la clase Forecast.

src/main/java/domain/Forecast.java

☐ Intentionality issue

[This class overrides "equals\(\)" and should therefore also override "hashCode\(\)".](#) cert cwe +

☐ Open ▾ ☐ Not assigned ▾ Reliability ▾ 🚩 Bug ⬇ Minor 15min effort • 12 days ago

Solución:

```
@Override
public int hashCode() {
    return Objects.hash(bets, description, frNum, minimumBet, question, winrate);
}
```

Bug 3:

Al igual que los otros 2, este tercer y último bug es lo mismo que los anteriores, por lo que con crear el hashCode() es suficiente.

src/main/java/domain/Question.java

☐ Intentionality issue

[This class overrides "equals\(\)" and should therefore also override "hashCode\(\)".](#) cert cwe +

☐ Open ▾ ☐ Not assigned ▾ Reliability ▾ 🚩 Bug ⬇ Minor 15min effort • 12 days ago

Solución:

```
@Override
public int hashCode() {
    return Objects.hash(betMinimum, event, forecasts, question, questionNumber, result);
}
```

Una vez los 3 bugs han sido solucionados, comprobamos los cambios en Sonar y como se puede observar en la siguiente imagen, han sido solucionados los 3 y el proyecto ya no sufre de bugs indeseados.

Bugs

0
↓ -3



Code Smells

Code Smell 1:

☐ Intentionality issue

[This block of commented-out lines of code should be removed.](#)

unused +

☐ Open ▾ ☐ Not assigned ▾ Maintainability ⚠️

Code Smell ⚠️ Major

5min effort • 13 days ago

Este code smells nos indica que hay varias líneas de código comentadas que deberían de ser eliminadas ya que no son comentarios si no código antiguo que se ha comentado. La solución como es normal ha sido eliminar esas líneas de código comentadas.

Code Smell 2:

☐ Adaptability issue

[Define a constant instead of duplicating this literal "initialize" 3 times.](#)

design +

☐ Open ▾ ☐ Not assigned ▾ Maintainability ⚠️

Code Smell ⚠️ Critical

8min effort • 13 days ago

Ya que se usaba el mismo string en dos métodos diferentes 3 veces, lo más adecuado era generar una constante tipo String y sustituirlo en los campos necesarios. La solución ha quedado así:

```
String init = "initialize";

public BLFacadeImplementation() {

    System.out.println("Creating BLFacadeImplementation instance");
    ConfigXML c = ConfigXML.getInstance();

    if (c.getDataBaseOpenMode().equals(init)) {
        dbManager = new DataAccess(c.getDataBaseOpenMode().equals(init));
        dbManager.initializeDB();
    } else
        dbManager = new DataAccess();
    dbManager.close();

}
```

Code Smell 3:

☐ Intentionality issue

[Remove the declaration of thrown exception 'java.lang.Exception', as it cannot be thrown from method's body.](#)
clumsy error-handling ... +

☐ Open ▾
 ☒ Asier Contreras ▾
 Maintainability ▾
⚠ Code Smell ▾ Minor

5min effort • 3 days ago

Este code smell nos indica que tenemos un método con un **throws** Exception que nunca va a ocurrir ya que no puede ser lanzado por ese método.

Code Smell 4:

☐ Intentionality issue

[Remove this unused "f1" local variable.](#)
unused +

☐ Open ▾
 ☒ Not assigned ▾
 Maintainability ▾
⚠ Code Smell ▾ Minor

5min effort • 16 days ago

Tenemos variables creadas que no se usan, por lo que es mejor si esas variables “unused” las eliminamos ya que no aportan nada.

Code Smell 5:

☐ Intentionality issue

[Add logic to this catch clause or eliminate it and rethrow the exception automatically.](#)
clumsy error-handling ... +

☐ Open ▾
 ☒ Not assigned ▾
 Maintainability ▾
⚠ Code Smell ▾ Minor

5min effort • 16 days ago

```

@WebMethod
public Card addCard(long cardNum, int cvv) throws ObjectAlreadyExistException, NoSuchAlgorithmException {
    User usr = this.getCurrentUser();
    Card card = null;

    String num = String.valueOf(cardNum);
    String last3Digits = num.substring(num.length() - 3, num.length() - 0);
    String hashedNum = hashPass(num);
    System.out.println("hash: " + hashedNum);

    dbManager.open(false);
    try {
        card = dbManager.addCard(usr, hashedNum, cvv, last3Digits);
    } catch (ObjectAlreadyExistException e) {
        throw e;
    } finally {
        dbManager.close();
    }

    return card;
}

```

El catch no tiene ningún sentido en este método, por lo que lo recomendable es eliminarlo ya que el método que use este método se encargará de cazar el error en caso de haberlo y lanzar la excepción.

Unai Artano:

Los 11 fallos de seguridad que se nos han creado han sido en los *try and catch*; pero especialmente en los *catch*-s, al hacer *e.printStackTrace()*. Para solucionarlo, hemos decidido hacer loggers. Un logger es una herramienta que se utiliza para registrar información relevante sobre la ejecución de un programa. Los loggers se utilizan para registrar eventos, mensajes, errores y otros datos importantes en tiempo de ejecución, lo que facilita el diagnóstico y la resolución de problemas en el software.

Otra cosa que hemos tenido en cuenta y que esté más organizado es qué a la hora de hacer los loggers en las clases que se necesiten (a continuación las mencionaremos) crear un archivo *.log* con cada clase donde apliquemos el logger.

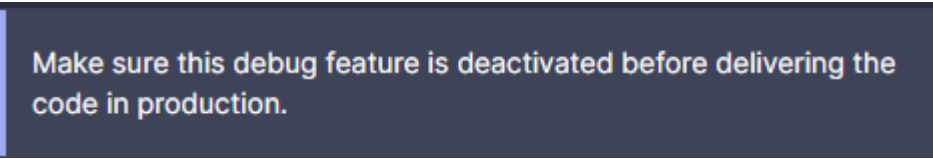
Uno de los problemas es que a la hora de estar programando el logger, cada acción que hacíamos de una clase, se nos escribía en el fichero inicial pero luego se iban creando más ficheros. Por tanto, para solucionar ese fallo que nos causaba, hemos añadido que si no existe el archivo, que no cree otro archivo y que lo modifique en el único archivo que haya de esa clase.

Para poder hacer un logger, tenemos que importar la librerías *logging* de *java.util*:

```
import java.util.logging.*;
```

1. fallo de seguridad:

En la clase *ConfigXML.java* en la línea 128.



Make sure this debug feature is deactivated before delivering the code in production.


```
src/main/java/configuration/ConfigXML.java
```

```

123         System.out.print("\t databaseLocal="+databaseLocal);
124         System.out.println("\t dataBaseOpenMode="+dataBaseOpenMode);
125
126     } catch (Exception e) {
127         System.out.println("Error in ConfigXML.java: problems with " + configFile);
128         e.printStackTrace();
129
130     }
131
132     private static String getTagValue(String sTag, Element eElement)
133

```

Make sure this debug feature is deactivated before delivering the code in production.

Solución:

```

76     // Crear logger
77     private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
78
79     private ConfigXML() {
80
81         try {
82
83             // Crear el nombre del que va a tener el archivo
84             String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.txt";
85
86             // Crear un fichero para saber si se ha creado o no
87             File archivo = new File(nombreArchivo);
88
89             // Comprobar si ya existe el archivo para no crear otro
90             if (!archivo.exists()) {
91                 // Crear el fichero
92                 FileHandler fileHandler = new FileHandler(nombreArchivo);
93
94                 // El formato que vamos a querer darle al logger
95                 fileHandler.setFormatter(new SimpleFormatter());
96
97                 fileHandler.setLevel(Level.INFO);
98
99                 logger.addHandler(fileHandler);
100             }
101
102             // El mensaje que queremos poner cuando el programa se ejecute correctamente
103             logger.log(Level.INFO, ">>>>>> Read from config.xml: \n");
104             logger.log(Level.INFO, ">>>>>> businessLogicLocal=" + businessLogicLocal + "\n");
105             logger.log(Level.INFO, ">>>>>> databaseLocal=" + databaseLocal + "\n");
106             logger.log(Level.INFO, ">>>>>> dataBaseOpenMode=" + dataBaseOpenMode + "\n");
107
108             } catch (SecurityException e) {
109                 // El mensaje que queremos poner cuando el programa salte un error
110                 logger.log(Level.INFO, ">>>>>> ERROR en el logger\n");
111             } catch (Exception e) {
112                 // El mensaje que queremos poner cuando el programa salte un error
113                 logger.log(Level.INFO, ">>>>>> Error in ConfigXML.java: problems with " + configFile + "\n");
114             }
115
116     }
117

```

2. fallo de seguridad:

En la clase DataAccess.java en la línea 162.

Make sure this debug feature is deactivated before delivering the code in production.

```
src/main/java/dataAccess/DataAccess.java
↑ Show 156 more lines
157         this.createForecast("Fernando 'el niño' Torres", 2.2f, q2);
158
159         System.out.println("Db initialized");
160     }
161     catch (Exception e){
162         e.printStackTrace();
163
164     }
165
166     /**
167     * This method creates a question for an event, with a question text and the minimum bet
168     */
169     ↓ Show 323 more lines
```

Solución:

```
35     // Crear logger
36     private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
37
38
62     // Crear el nombre del que va a tener el archivo
63     String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.log";
64
65     // Crear un fichero para saber si se ha creado o no
66     File archivo = new File(nombreArchivo);
67
68     // Comprobar si ya existe el archivo para no crear otro
69     if (!archivo.exists()) {
70         // Crear el fichero
71         Handler fileHandler = new FileHandler(nombreArchivo);
72
73         // El formato que vamos a querer darle al logger
74         fileHandler.setFormatter(new SimpleFormatter());
75
76         fileHandler.setLevel(Level.INFO);
77
78         logger.addHandler(fileHandler);
79     }
80
81
82     // El mensaje que queremos poner cuando el programa se ejecute correctamente
83     logger.log(Level.INFO, ">>>>>> BD inicializada\n");
84     } catch (SecurityException e) {
85         // El mensaje que queremos poner cuando el programa salte un error
86         logger.log(Level.INFO, ">>>>>> ERROR en el logger\n");
87     } catch (Exception E) {
88         logger.log(Level.INFO, ">>>>>> ERROR al inicializar la BD\n");
89     }
90 }
```

3. fallo de seguridad:

En la clase ObjectdbManagerServer.java en la línea 48.

Make sure this debug feature is deactivated before delivering the code in production.

```
src/main/java/dataAccess/ObjectdbManagerServer.java
↑ Show 42 more lines
43
44     ObjectdbManagerServer dialog = new ObjectdbManagerServer();
45     dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
46     dialog.setVisible(true);
47 } catch (Exception e) {
48     e.printStackTrace();
49
50 }
51
52
53 public ObjectdbManagerServer() {
↓ Show 80 more lines
```

Solución:

```
44 //Crear logger
45 Logger logger = Logger.getLogger(ObjectdbManagerServer.class.getSimpleName());
46
47 try {
48
49     //Crear el nombre del que ya a tener el archivo
50     String nombreArchivo = ObjectdbManagerServer.class.getSimpleName() + "LOGGER.txt";
51
52     //Crear un fichero para saber si se ha creado o no
53     File archivo = new File(nombreArchivo);
54
55     //Comprobar si ya existe el archivo para no crear otro
56     if(!archivo.exists()) {
57         // crear el fichero
58         FileHandler fileHandler = new FileHandler(nombreArchivo);
59
60         // El formato que vamos a guardar darle al logger
61         fileHandler.setFormatter(new SimpleFormatter());
62
63         fileHandler.setLevel(Level.INFO);
64
65         logger.addHandler(fileHandler);
66     }
67
68
69 //El mensaje que queremos poner cuando el programa se ejecuta correctamente
70 logger.log(Level.INFO, ">>>>>>> "+ObjectdbManagerServer.class.getSimpleName()+" ejecutando correctamente\n");
71 } catch (SecurityException e) {
72     // El mensaje que queremos poner cuando el programa salta un error
73     logger.log(Level.INFO, ">>>>>>> ERROR en el logger\n");
74 } catch (Exception e) {
75     //El mensaje que queremos poner cuando el programa salta un error
76     logger.log(Level.INFO, ">>>>>>> ERROR al ejecutar en "+ObjectdbManagerServer.class.getSimpleName()+"\n");
77 }
78
79 }
80
81 }
```

4. fallo de seguridad:

En la clase CloseEventGUI.java en la línea 59.

Make sure this debug feature is deactivated before delivering the code in production.

```
src/main/java/gui/CloseEventGUI.java

54
55     public CloseEventGUI() {
56         try {
57             jbInit();
58         } catch (Exception e) {
59             e.printStackTrace();
60         }
61     }
62
63     private void jbInit() throws Exception {
64
```


Solución:

```
59     // Crear logger
60     private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
61
62     public CloseEventGUI() {
63
64         try {
65
66             // Crear el nombre del que va a tener el archivo
67             String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.txt";
68
69             // Crear un fichero para saber si se ha creado o no
70             File archivo = new File(nombreArchivo);
71
72             // Comprobar si ya existe el archivo para no crear otro
73             if (!archivo.exists()) {
74                 // Crear el fichero
75                 FileHandler fileHandler = new FileHandler(nombreArchivo);
76
77                 // El formato que queremos poner cuando se ejecute correctamente
78                 fileHandler.setFormatter(new SimpleFormatter());
79
80                 fileHandler.setLevel(Level.INFO);
81
82                 logger.addHandler(fileHandler);
83             }
84
85             jbInit();
86
87             // El mensaje que queremos poner cuando el programa se ejecute correctamente
88             logger.log(Level.INFO, ">>>>>> " + this.getClass().getSimpleName() + " ejecutando correctamente\n");
89         } catch (SecurityException e) {
90             // El mensaje que queremos poner cuando el programa salta un error
91             logger.log(Level.INFO, ">>>>>> ERROR en el logger\n");
92         } catch (Exception e) {
93             // El mensaje que queremos poner cuando el programa salta un error
94             logger.log(Level.INFO, ">>>>>> ERROR al ejecutar en " + this.getClass().getSimpleName() + "\n");
95         }
96     }
97
```

5. fallo de seguridad:

En la clase CreateBetGUI.java en la línea 62.

Make sure this debug feature is deactivated before delivering the code in production.



The screenshot shows a code editor with the file path `src/main/java/gui/CreateBetGUI.java`. Line 62 contains the code `e.printStackTrace();`. A red squiggly line underlines this code, and a callout box points to it with the message: "Make sure this debug feature is deactivated before delivering the code in production." The code around line 62 is as follows:

```
57
58     public CreateBetGUI() {
59         try {
60             jbInit();
61         } catch (Exception e) {
62             e.printStackTrace();
63         }
64     }
65
66     private void jbInit() throws Exception {
67
```

Solución:

```
60 // crear logger
61 private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
62
63 public CreateBetGUI() {
64
65     try {
66
67         // crear el nombre del que va a tener el archivo
68         String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.txt";
69
70         // crear un fichero para saber si se ha creado o no
71         File archivo = new File(nombreArchivo);
72
73         // comprobar si ya existe el archivo para no crear otro
74         if (!archivo.exists()) {
75             // crear el fichero
76             FileHandler fileHandler = new FileHandler(nombreArchivo);
77
78             // El formato que vamos a querer darle al logger
79             fileHandler.setFormatter(new SimpleFormatter());
80
81             fileHandler.setLevel(Level.INFO);
82
83             logger.addHandler(fileHandler);
84         }
85
86         jbInit();
87
88         // El mensaje que queremos poner cuando el programa se ejecute correctamente
89         logger.log(Level.INFO, ">>>>>>> " + this.getClass().getSimpleName() + " ejecutando correctamente\n");
90     } catch (SecurityException e) {
91         // El mensaje que queremos poner cuando el programa salte un error
92         logger.log(Level.INFO, ">>>>>>> ERROR en el logger\n");
93     } catch (Exception e) {
94         // El mensaje que queremos poner cuando el programa salte un error
95         logger.log(Level.INFO, ">>>>>>> ERROR al ejecutar en " + this.getClass().getSimpleName() + "\n");
96     }
97 }
98
```

6. fallo de seguridad:

En la clase CreateEventGUI.java en la línea 33.

Make sure this debug feature is deactivated before delivering the code in production.

```
src/main/java/gui/CreateEventGUI.java

28
29     public CreateEventGUI() {
30         try {
31             jbInit();
32         } catch (Exception e) {
33             e.printStackTrace();
34
35         }
36
37         private void jbInit() throws Exception {
38             setResizable(false);
39
40         }
41     }
42 }
```

Solución:

```
31 // Crear logger
32 private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
33
34 public CreateEventGUI() {
35
36     try {
37
38         // Crear el nombre del que va a tener el archivo
39         String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.txt";
40
41         // Crear un fichero para saber si se ha creado o no
42         File archivo = new File(nombreArchivo);
43
44         // Comprobar si ya existe el archivo para no crear otro
45         if (!archivo.exists()) {
46             // Crear el fichero
47             FileHandler fileHandler = new FileHandler(nombreArchivo);
48
49             // El formato que vamos a querer darle al logger
50             fileHandler.setFormatter(new SimpleFormatter());
51
52             fileHandler.setLevel(Level.INFO);
53
54             logger.addHandler(fileHandler);
55         }
56
57         jbInit();
58
59         // El mensaje que queremos poner cuando el programa se ejecute correctamente
60         logger.log(Level.INFO, ">>>>>>> " + this.getClass().getSimpleName() + " ejecutando correctamente\n");
61     } catch (SecurityException e) {
62         // El mensaje que queremos poner cuando el programa salte un error
63         logger.log(Level.INFO, ">>>>>>> ERROR en el logger\n");
64     } catch (Exception e) {
65         // El mensaje que queremos poner cuando el programa salte un error
66         logger.log(Level.INFO, ">>>>>>> ERROR al ejecutar en " + this.getClass().getSimpleName() + "\n");
67     }
68 }
69 }
```

7. fallo de seguridad:

En la clase CreateForecastGUI.java en la línea 64.

Make sure this debug feature is deactivated before delivering the code in production.

```
src/main/java/gui/CreateForecastGUI.java
... Show 58 more lines
59
60     public CreateForecastGUI() {
61         try {
62             jbInit();
63         } catch (Exception e) {
64             e.printStackTrace();
65
66         }
67     }
68     private void jbInit() throws Exception {
69
... Show 279 more lines
```

Solución:

```
65     // Crear logger
66     private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
67
68     public CreateForecastGUI() {
69
70         try {
71
72             // Crear el nombre del que va a tener el archivo
73             String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.txt";
74
75             // Crear un fichero para saber si se ha creado o no
76             File archivo = new File(nombreArchivo);
77
78             // Comprobar si ya existe el archivo para no crear otro
79             if (!archivo.exists()) {
80                 // Crear el fichero
81                 FileHandler fileHandler = new FileHandler(nombreArchivo);
82
83                 // El formato que vamos a usar para el logger
84                 fileHandler.setFormatter(new SimpleFormatter());
85
86                 fileHandler.setLevel(Level.INFO);
87
88                 logger.addHandler(fileHandler);
89             }
90
91             jbInit();
92
93             // El mensaje que queremos poner cuando el programa se ejecute correctamente
94             logger.log(Level.INFO, ">>>>>>>" + this.getClass().getSimpleName() + " ejecutando correctamente\n");
95         } catch (SecurityException e) {
96             // El mensaje que queremos poner cuando el programa salte un error
97             logger.log(Level.INFO, ">>>>>>> ERROR en el logger\n");
98         } catch (Exception e) {
99             // El mensaje que queremos poner cuando el programa salte un error
100             logger.log(Level.INFO, ">>>>>>> ERROR al ejecutar en " + this.getClass().getSimpleName() + "\n");
101         }
102     }
103
```

8. fallo de seguridad:

En la clase CreateQuestionGUI.java en la línea 51.

Make sure this debug feature is deactivated before delivering the code in production.

```
src/main/java/gui/CreateQuestionGUI.java

... Show 45 more lines

46
47     public CreateQuestionGUI(Vector<domain.Event> v) {
48         try {
49             jbInit(v);
50         } catch (Exception e) {
51             e.printStackTrace();
52         }
53     }
54
55     private void jbInit(Vector<domain.Event> v) throws Exception {
56
57     ... Show 258 more lines
```

Solución:

```
51     private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
52
53     public CreateQuestionGUI(Vector<domain.Event> v) {
54
55         try {
56
57             // Crear el nombre del que va a tener el archivo
58             String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.txt";
59
60             // Crear un fichero para saber si se ha creado o no
61             File archivo = new File(nombreArchivo);
62
63             // Comprobar si ya existe el archivo para no crear otro
64             if (!archivo.exists()) {
65                 // Crear el fichero
66                 FileHandler fileHandler = new FileHandler(nombreArchivo);
67
68                 // El formato que vamos a querer darle al logger
69                 fileHandler.setFormatter(new SimpleFormatter());
70
71                 fileHandler.setLevel(Level.INFO);
72
73                 logger.addHandler(fileHandler);
74             }
75
76             jbInit(v);
77
78             // El mensaje que queremos poner cuando el programa se ejecuta correctamente
79             logger.log(Level.INFO, ">>>>>> " + this.getClass().getSimpleName() + " ejecutando correctamente\n");
80         } catch (SecurityException e) {
81             // El mensaje que queremos poner cuando el programa salta un error
82             logger.log(Level.INFO, ">>>>>> ERROR en el logger\n");
83         } catch (Exception e) {
84             // El mensaje que queremos poner cuando el programa salta un error
85             logger.log(Level.INFO, ">>>>>> ERROR al ejecutar en " + this.getClass().getSimpleName() + "\n");
86         }
87     }
88
```


9. fallo de seguridad:

En la clase CreateQuestionGUI.java en la línea 305.

Make sure this debug feature is deactivated before delivering the code in production.

```
src/main/java/gui/CreateQuestionGUI.java
Show 299 more lines
300     jLabelMsg.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorQueryAlreadyExist"));
301   } catch (java.lang.NumberFormatException e1) {
302     jLabelError.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorNumber"));
303   } catch (Exception e1) {
304
305     e1.printStackTrace();
306
307   }
308 }
309
310 private void jButtonClose_actionPerformed(ActionEvent e) {
Show 4 more lines
```

Solución:

```
298 //Crear el nombre del que va a tener el archivo
299 String nombreArchivo1 = this.getClass().getSimpleName() + "_BotonCreate_LOGGER.txt";
300
301 //Crear un fichero para saber si se ha creado o no
302 File archivo1 = new File(nombreArchivo1);
303
304 //Comprobar si ya existe el archivo para no crear otro
305 if(!archivo1.exists()) {
306   // Crear el fichero
307   FileHandler fileHandler1 = new FileHandler(nombreArchivo1);
308
309   // El formato que vamos a guardar darle al logger
310   fileHandler1.setFormatter(new SimpleFormatter());
311
312   fileHandler1.setLevel(Level.INFO);
313
314   logger.addHandler(fileHandler1);
315 }
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339 logger.log(Level.INFO, ">>>>>> TODO CORRECTO en el boton de crear la pregunta en "
340 + this.getClass().getSimpleName() + ".\n");
341
342
343
344
345
346
347
348
349
350
351 } catch (SecurityException s) {
352   // El mensaje que queremos poner cuando el programa salte un error
353   logger.log(Level.INFO, ">>>>>> ERROR en el logger\n");
354 } catch (Exception e1) {
355   // El mensaje que queremos poner cuando el programa salte un error
356   logger.log(Level.INFO,
357 ">>>>>> ERROR en el boton de crear la pregunta en " + this.getClass().getSimpleName()+".\n");
358 }
```

10. fallo de seguridad:

En la clase FindQuestionGUI.java en la línea 62.

Make sure this debug feature is deactivated before delivering the code in production.

```
src/main/java/gui/FindQuestionsGUI.java
... Show 56 more lines
57     {
58         jbInit();
59     }
60     catch(Exception e)
61     {
62         e.printStackTrace();
63     }
64 }
65
66
67 private void jbInit() throws Exception
... Show 176 more lines
```

Make sure this debug feature is deactivated before delivering the code in production.

Solución:

```
53 // Crear logger
54 private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
55
56 public FindQuestionsGUI() {
57     try {
58
59         // Crear el nombre del que va a tener el archivo
60         String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.txt";
61
62         // Crear un fichero para saber si se ha creado o no
63         File archivo = new File(nombreArchivo);
64
65         // Comprobar si ya existe el archivo para no crear otro
66         if (!archivo.exists()) {
67             // Crear el fichero
68             FileHandler fileHandler = new FileHandler(nombreArchivo);
69
70             // El formato que queremos a que nos de al logger
71             fileHandler.setFormatter(new SimpleFormatter());
72
73             fileHandler.setLevel(Level.INFO);
74
75             logger.addHandler(fileHandler);
76         }
77
78         jbInit();
79
80         // El mensaje que queremos poner cuando el programa se ejecute correctamente
81         logger.log(Level.INFO, ">>>>>>>" + this.getClass().getSimpleName() + "ejecutando correctamente\n");
82     } catch (SecurityException e) {
83         // El mensaje que queremos poner cuando el programa salte un error
84         logger.log(Level.INFO, ">>>>>>> ERROR en el logger\n");
85     } catch (Exception e) {
86         // El mensaje que queremos poner cuando el programa salte un error
87         logger.log(Level.INFO, ">>>>>>> ERROR al ejecutar en " + this.getClass().getSimpleName() + "\n");
88     }
89 }
90 }
```

11. fallo de seguridad:

En la clase LoginGUI.java en la línea 120.

Make sure this debug feature is deactivated before delivering the code in production.

```
120 public LoginGUI() {
121
122     try {
123
124         //Crear el nombre del que va a tener el archivo
125         String nombreArchivo = this.getClass().getSimpleName() + "LOGGER.txt";
126
127         //Crear un fichero para saber si se ha creado o no
128         File archivo = new File(nombreArchivo);
129
130         //Comprobar si ya existe el archivo para no crear otro
131         if(!archivo.exists()) {
132             // Crear el fichero
133             FileHandler fileHandler = new FileHandler(nombreArchivo);
134
135             // El formato que vamos a poner darle al logger
136             fileHandler.setFormatter(new SimpleFormatter());
137
138             fileHandler.setLevel(Level.INFO);
139
140             logger.addHandler(fileHandler);
141         }
142
143         getContentPane().add(getPanel());
144         jbInit();
145
146         // El mensaje que queremos poner cuando el programa se ejecuta correctamente
147         logger.log(Level.INFO, ">>>>>>> " + this.getClass().getSimpleName() + " ejecutando correctamente\n");
148     } catch (SecurityException e) {
149         // El mensaje que queremos poner cuando el programa salta un error
150         logger.log(Level.INFO, ">>>>>>> ERROR en el logger\n");
151     } catch (Exception e) {
152         // El mensaje que queremos poner cuando el programa salta un error
153         logger.log(Level.INFO, ">>>>>>> ERROR al ejecutar en " + this.getClass().getSimpleName() + "\n");
154     }
155 }
```

src/main/java/gui/LoginGUI.java

Show 114 more lines

```
115 public LoginGUI() {
116     try {
117         getContentPane().add(getPanel());
118         jbInit();
119     } catch (Exception e) {
120         e.printStackTrace();
121     }
122 }
123
124 private void jbInit() throws Exception {
125
```

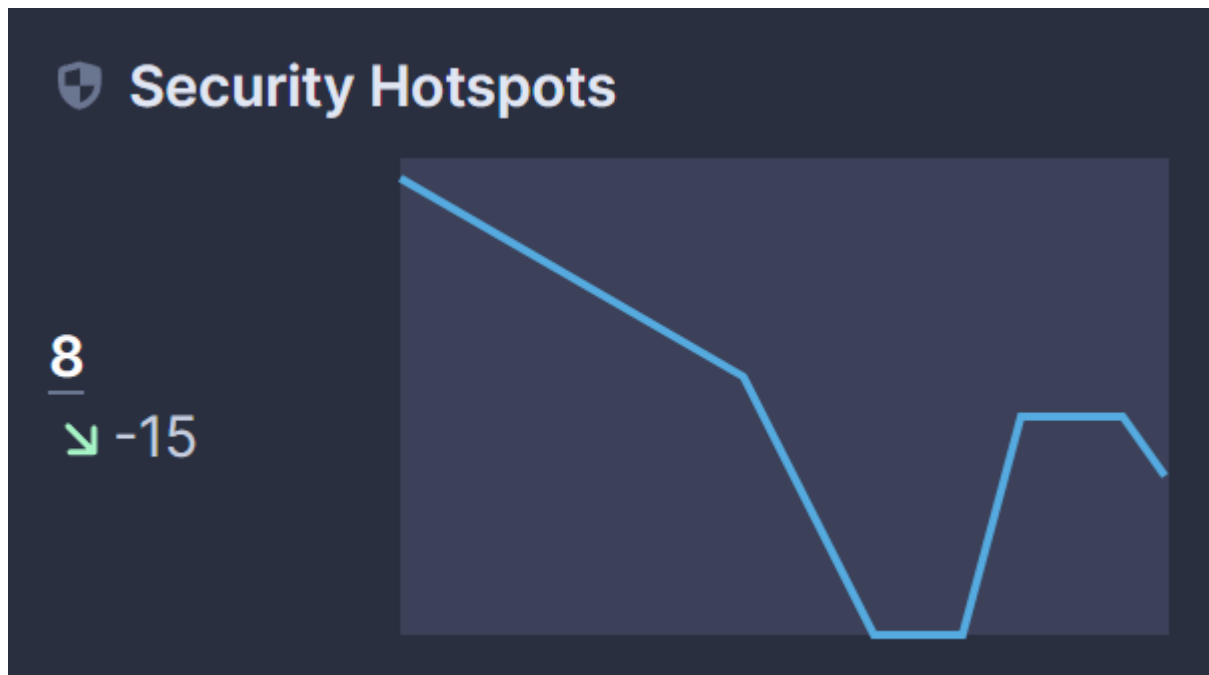
Make sure this debug feature is deactivated before delivering the code in production.

Show 124 more lines

Solución:

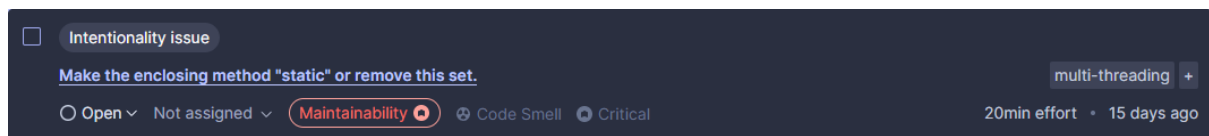
```
37 // Crear logger
38 private Logger logger = Logger.getLogger(this.getClass().getSimpleName());
39
```

Una vez los 11 fallos de seguridad han sido solucionados, comprobamos los cambios en Sonar y como se puede observar en la siguiente imagen, han sido solucionados los 11 y el proyecto ya no sufre de fallos de seguridad.



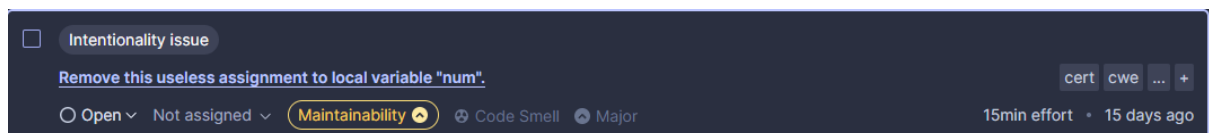
Code Smells

Code Smell 1:



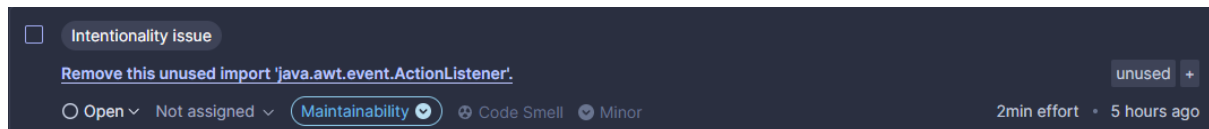
Este Code Smell nos indica que los métodos de instancias no deben de escribirse en campos "static".

Code Smell 2:



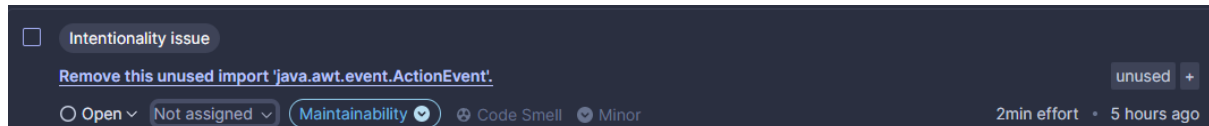
Este Code Smell nos dice que teníamos una variable *num* que no se estaba utilizando. Para ello, hemos quitado esa variable ya que no aportaba nada al método.

Code Smell 3:



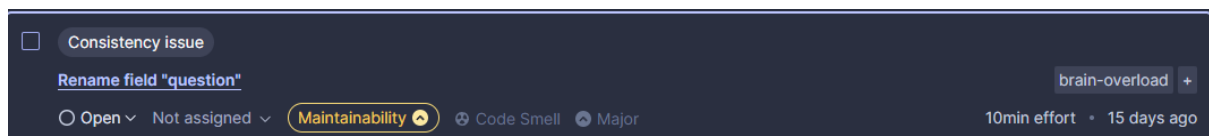
Este Code Smell nos indica que hay un import; exactamente el *import java.awt.event.ActionListener* que no se está usando.

Code Smell 4:



Este Code Smell nos indica que hay un import; exactamente el *import java.awt.event.ActionEvent* que no se está usando.

Code Smell 5:



Este Code Smell nos indica que un campo no debe duplicar el nombre de la clase que lo contiene. Por tanto, es cambiar de *question* a *pregunta*, por ejemplo.

Martín Horsfield:

Vulnerabilidades en el código:

Primera vulnerabilidad

secrets:S6702

Make sure this SonarQube token gets revoked, changed, and removed from the code.

```

  65         <sonar.organization>proyectois2-bets</sonar.organization>
  66         <sonar.host.url>https://sonarcloud.io</sonar.host.url>
  67         <sonar.projectKey>proyectois2-bets_rep bets</sonar.projectKey>
  68         <sonar.login>3fc9c5fa96837f1b974ee0d2c77b26f14a4a05b4</sonar.login>
  69     </properties>
  70
  71     <!-- https://mvnrepository.com/artifact/junit/junit -->
```

Este script está situado en el pom.xml. Es código necesario para que sonarcloud pueda subir el análisis del código al repositorio en sonarcloud cuando termine. El problema surge cuando el pom.xml es visible en nuestro proyecto de GitHub. Al ser visible, el “token” también es visible y cualquier persona puede ver nuestro “token”.

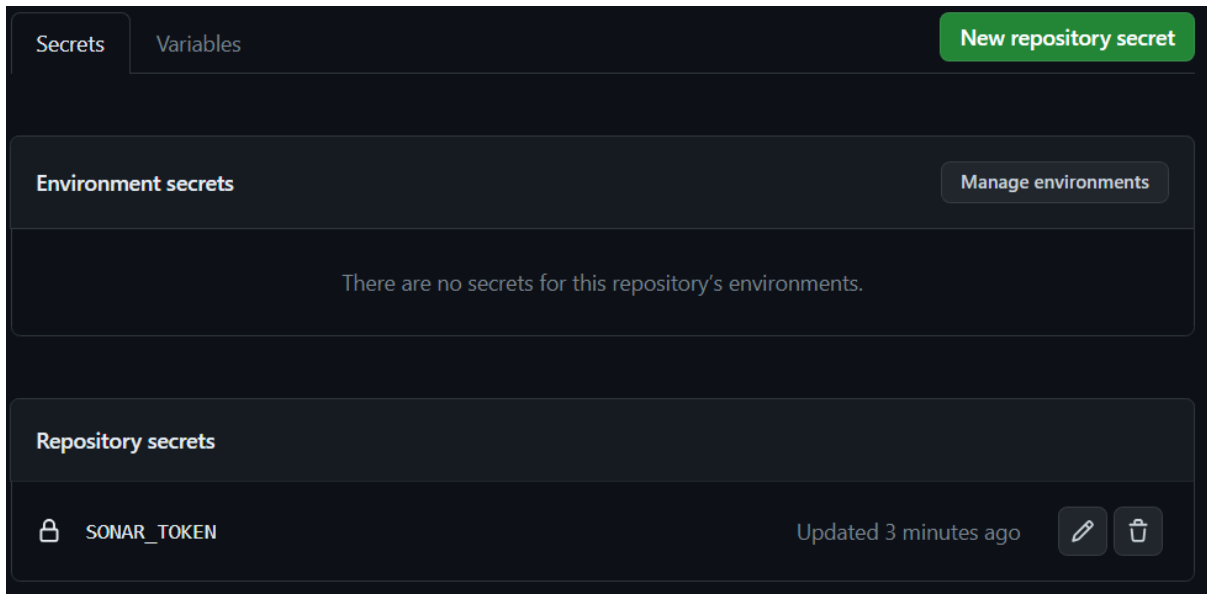
Para solucionarlo debemos de conocer el build.yml, que se creó para poder usar GitHub actions.

Esta es la sección necesaria:

```

33     - name: Build and analyze
34       env:
35         GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to
36         SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
37       run: mvn clean org.jacoco:jacoco-maven-plugin:prepare-ag
```

En “env” tenemos nuestro “secrets.SONAR_TOKEN” que recibe el token de un secreto declarado en GitHub de la siguiente manera.



En GitHub podemos crear este “secreto” para que, aunque sea visible el código, no sea visible el token.

En el pom.xml, quedaría de la siguiente manera:

```
65         <sonar.organization>proyectois2-bets</sonar.organization>
66         <sonar.host.url>https://sonarcloud.io</sonar.host.url>
67         <sonar.projectKey>proyectois2-bets_repbets</sonar.projectKey>
68 +       <sonar.login>${env.SONAR_TOKEN}</sonar.login>
69     </properties>
70
71     <!-- https://mvnrepository.com/artifact/junit/junit -->
```

“env.SONAR_TOKEN” llama al build.yml que llama a GitHub para recibir el token.

Segunda vulnerabilidad

java:S2755

Disable access to external entities in XML parsing.

```
src/main/java/configuration/ConfigXML.java

@@ -97,6 +97,7 @@ private ConfigXML() {
    }
    97
    98
    99         DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

    100         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
    101         Document doc = dBuilder.parse(new File(configFile));
    102         doc.getDocumentElement().normalize();
```

Cuando se usa “parsing” con entidades externas en XML puede surgir una vulnerabilidad a ataques XXE. Para ello, SonarCloud recomienda añadir la siguiente opción:

```
    97         }
    98
    99         DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
    100 +         dbFactory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
    101         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
    102         Document doc = dBuilder.parse(new File(configFile));
    103         doc.getDocumentElement().normalize();
```

Code smell 1:

Code smell: Replace the type specification in this constructor call with the diamond operator("<>")

Maintainability: Low

```
src/main/java/dataAccess/DataAccess.java

@@ -358,7 +358,7 @@ public Vector<Event> getEvents(Date date) {
    */
    358
    359     public Vector<Date> getEventsMonth(Date date) {
    360         System.out.println(">> DataAccess: getEventsMonth");
    361 -         Vector<Date> res = new Vector<Date>();
    362
    363         Date firstDayMonthDate = UtilDate.firstDayMonth(date);
    364         Date lastDayMonthDate = UtilDate.lastDayMonth(date);
```

java.S2293

Previo a Java 7, el compilador de Java necesitaba que se declarase explícitamente el tipo de argumento para un tipo genérico, como "List" o "Array". Java 7 introdujo el operador diamante ("<>") y para evitar redundancias, no se debería de declarar en la llamada a la constructora.

```
358         */
359         public Vector<Date> getEventsMonth(Date date) {
360             System.out.println(">> DataAccess: getEventsMonth");
361 +         Vector<Date> res = new Vector<>();
362
363             Date firstDayMonthDate = UtilDate.firstDayMonth(date);
364             Date lastDayMonthDate = UtilDate.lastDayMonth(date);
```

Code smell 2:

Code smell: Make this anonymous inner class a lambda

Maintainability: Medium

```
src/main/java/businessLogic/BusinessLogicServer.java
@@ -71,14 +71,12 @@ public void windowClosed(WindowEvent arg0) {
    getContentPane().add(buttonPane, BorderLayout.SOUTH);
    {
        JButton okButton = new JButton("OK");
        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                textArea.append("\n\n\nClosing the server... ");

                //server.close();

                System.exit(1);
            }
        });
        okButton.setActionCommand("OK");
        buttonPane.add(okButton);
    }
}
```

java:S1604

Con Java 8 ya no se debería de definir funciones dentro de un parámetro de un método. En cambio se deberían de usar funciones lambda.

```
71      getContentPane().add(buttonPane, BorderLayout.SOUTH);
72      {
73          JButton okButton = new JButton("OK");
74      +      okButton.addActionListener((e) -> {
75      +          textArea.append("\n\n\nClosing the server... ");
76      +
77      +          //server.close();
78      +
79      +          System.exit(1);
80      +
81      +      });
82      +      okButton.setActionCommand("OK");
83      +      buttonPane.add(okButton);
84      +    }
85  }
```

Code smell 3:

Code smell: Use static access with "javax.swing.WindowConstants" for "DISPOSE_ON_CLOSE"

Maintainability: High

```
src/main/java/businessLogic/BusinessLogicServer.java

@@ -15,6 +15,7 @@
15  import configuration.ConfigXML;
16
17  import javax.swing.JTextArea;

18  import javax.xml.ws.Endpoint;
19
20

@@ -38,7 +39,7 @@ public class BusinessLogicServer extends JDialog {
38      public static void main(String[] args) {
39          try {
40              BusinessLogicServer dialog = new BusinessLogicServer();
41  -          dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
42              dialog.setVisible(true);
43          } catch (Exception e) {
44              ///e.printStackTrace();
```

java:S3252

Para mantener claridad y evitar la apariencia de que existen dos tipos estáticos distintos, se debería de usar el atributo de la clase base, y no de la derivada.

```
15  import configuration.ConfigXML;
16
17  import javax.swing.JTextArea;
18  + import javax.swing.WindowConstants;
19  import javax.xml.ws.Endpoint;
20
21

39      public static void main(String[] args) {
40          try {
41              BusinessLogicServer dialog = new BusinessLogicServer();
42  +          dialog.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
43              dialog.setVisible(true);
44          } catch (Exception e) {
45              ///e.printStackTrace();
```

Code smell 4:

Code smell: Remove this empty statement

Maintainability: Low

```
src/main/java/businessLogic/BLFacadeImplementation.java

@@ -111,13 +111,10 @@ public Question createQuestion(Event event, String question, float betMinimum)
111
112 +      dbManager.open(false);
113
114 -      try {
115 -          qry = dbManager.createQuestion(event, question, betMinimum);
116 -      } catch (QuestionAlreadyExist e) {
117 -          throw e;
118 -      } finally {
119 -          dbManager.close();
120 -      }
121
122      return qry;
123  };
```

java:S1116

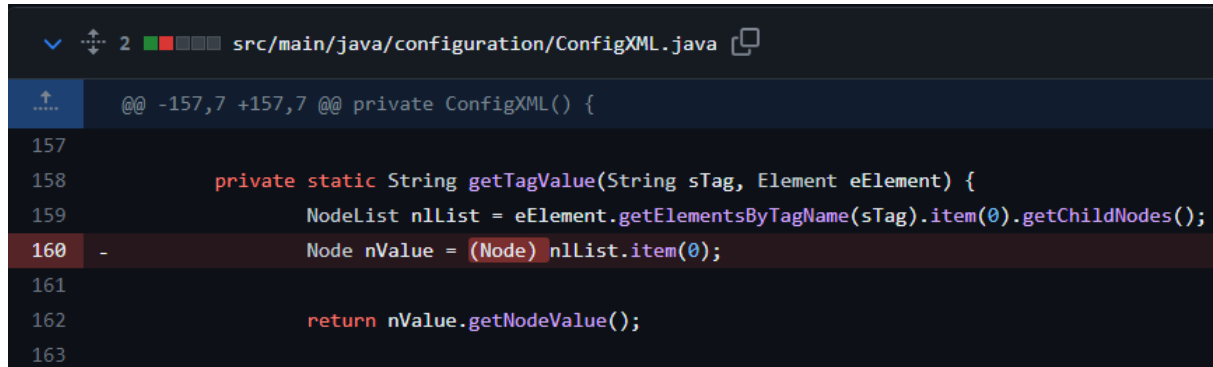
Un code smell con código innecesario.

```
111
112      dbManager.open(false);
113
114 +      qry = dbManager.createQuestion(event, question, betMinimum);
115 +
116 +      dbManager.close();
117 +
118
119      return qry;
120  };
```

Code smell 5:

Code smell: Remove this unnecessary cast to "Node"

Maintainability: Low

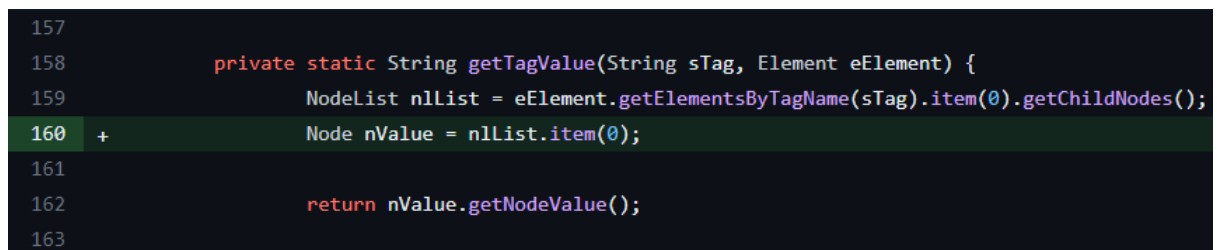


```
src/main/java/configuration/ConfigXML.java

@@ -157,7 +157,7 @@ private ConfigXML() {
157
158     private static String getTagValue(String sTag, Element eElement) {
159         NodeList nList = eElement.getElementsByTagName(sTag).item(0).getChildNodes();
160 -        Node nValue = (Node) nList.item(0);
161
162         return nValue.getNodeValue();
163
```

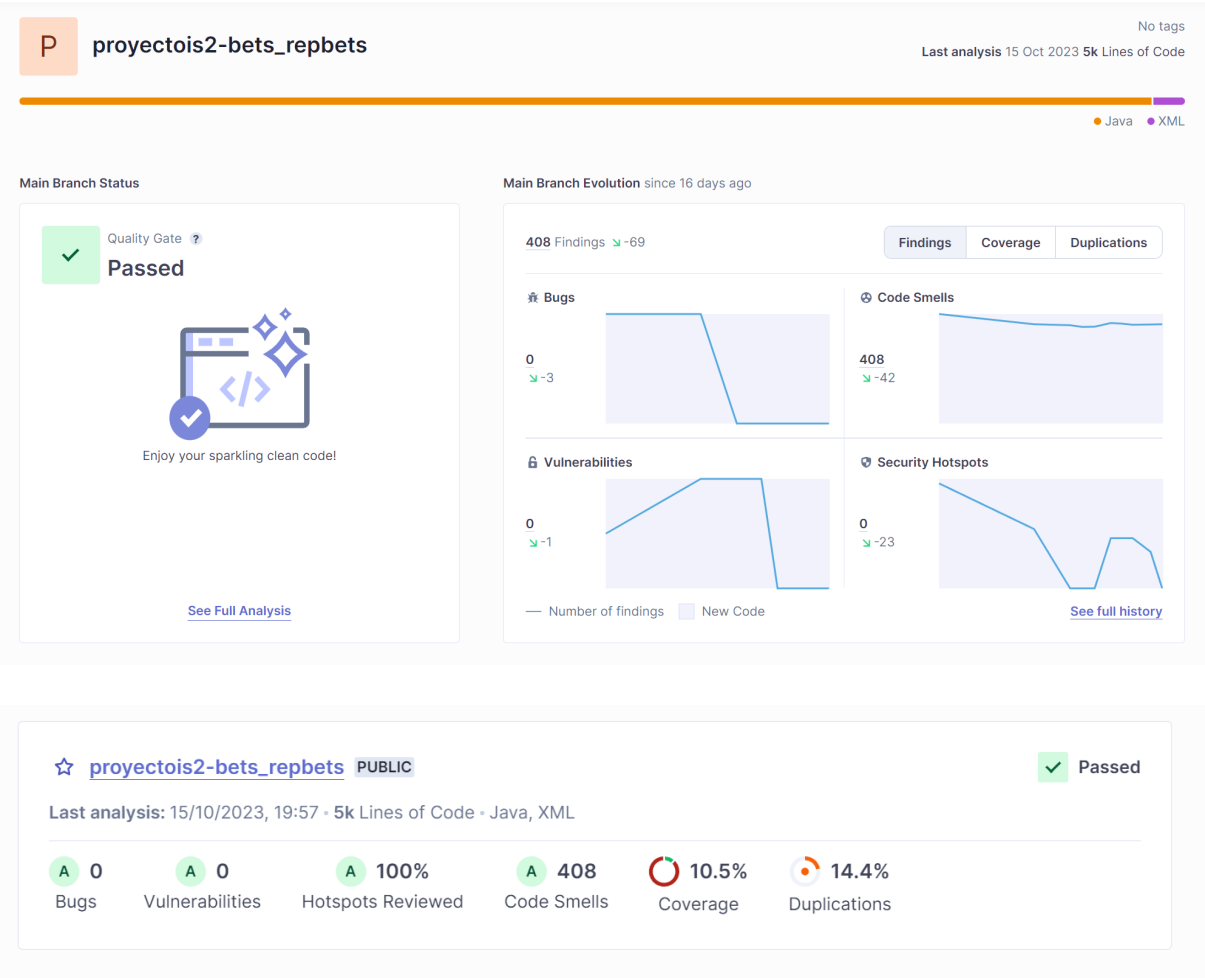
java:S1905

No se deberían de usar 'casts' innecesarios para evitar redundancias y para que el código sea más legible.



```
157
158     private static String getTagValue(String sTag, Element eElement) {
159         NodeList nList = eElement.getElementsByTagName(sTag).item(0).getChildNodes();
160 +        Node nValue = nList.item(0);
161
162         return nValue.getNodeValue();
163
```

Informe final



Una vez todos los fallos de seguridad, bugs, code smells... han sido solucionados, hemos terminado con la evolución que se puede ver en las imágenes. Hemos corregido los 3 bugs, la única vulnerabilidad que detectaba SonarCloud y hemos aumentado el Coverage del código de 0% inicial al 10.5% tras las clases de testing. Además, hemos reducido el código duplicado del 16% al 14.4%.

Reparto de horas

Nombre	Horas
Unai Artano	3
Asier Contreras	2
Martin Ian Horsfield	2