



PLAN DE GESTIÓN DE MANTENIMIENTO



Javier Santana Delgado

Jose Antonio Santacruz Gallego

Tamara Redondo Soto

Julio Sánchez de las Heras Martín Consuegra

Alejandro Riquelme Castaño

José Antonio Oliver González-Ortega



PLAN DE GESTIÓN DE MANTENIMIENTO

| | |
|---|----------|
| INTRODUCCIÓN AL MANTENIMIENTO | 3 |
| INFORME DEL ESTADO DEL SISTEMA TRAS LA PRIMERA RELEASE | 3 |
| PMP | 3 |
| FINDBUGS | 3 |
| CHECKSTYLE..... | 4 |
| JXR..... | 5 |
| JAVADOC | 5 |
| MEJORAS DEL PROYECTO | 6 |
| FUTURAS MEJORAS | 8 |

PLAN DE GESTIÓN DE MANTENIMIENTO

INTRODUCCIÓN AL MANTENIMIENTO

En este documento se tratarán los **errores detectados** después de lanzar nuestro proyecto Fritura al mercado. Estos se detallan gracias a la creación de informes a través de plugins como "Pmd", "Findbugs", "JXR", "JavaDoc" o "Checkstyle" los cuales son agrupados en un index.html gracias al plugin de Surefire y ejecutando mvn site:site.

INFORME DEL ESTADO DEL SISTEMA TRAS LA PRIMERA RELEASE

Gracias a los plugins previamente comentados, se ha analizado el **estado de nuestro proyecto** tras haber realizado la primera release, este sería:

PMD



Escanea el código fuente Java y localiza problemas potenciales como posibles bugs, código duplicado, etc.

Como podemos comprobar en las capturas insertadas a continuación se muestran las variables en la clase **DTOReserva** que no son utilizadas y al lado la línea a la que corresponde.





Violations By Priority

Priority 3

Dominio/DTOMesa.java

| Rule | Violation | Line |
|---|--|---------|
| CollapsibleIfStatements  | These nested if statements could be combined | 142-144 |
| UnusedLocalVariable  | Avoid unused local variables such as 'consulta'. | 157 |

Dominio/DTOReserva.java

| Rule | Violation | Line |
|---|---|------|
| UnusedLocalVariable  | Avoid unused local variables such as 'res2'. | 94 |
| UnusedLocalVariable  | Avoid unused local variables such as 'subConsulta'. | 141 |
| UnusedLocalVariable  | Avoid unused local variables such as 'rs'. | 144 |
| UnusedLocalVariable  | Avoid unused local variables such as 'turno'. | 265 |

FINGBUGS

Este plugin se encarga de **buscar bugs en programas java**, principalmente sobre el concepto de antipatrones así como las características del lenguaje difíciles, invariantes mal interpretadas, etc.

Así por ejemplo, en la siguiente captura, el informe realizado por el plugin nos informa que al generar una **consulta SQL** para mandarla a la base de datos no deberíamos haber

utilizado el símbolo "+" para concatenar las variables en la variable String que representa dicha consulta.

Summary

| Classes | Bugs | Errors | Missing Classes |
|---------|------|--------|-----------------|
| 8 | 15 | 0 | 19 |

Files

| Class | Bugs |
|-------------------------------------|------|
| Dominio.DTOCamarero | 1 |
| Dominio.DTOMesa | 4 |
| Dominio.DTOReserva | 10 |

Dominio.DTOCamarero

| Bug | Category | Details | Line | Priority |
|---|----------|--|------|----------|
| A prepared statement is generated from a nonconstant String in Dominio.DTOCamarero.leerAvisos(String, LinkedList) | SECURITY | SQL_PREPARED_STATEMENT_GENERATED_FROM_NONCONSTANT_STRING | 73 | High |

CHECKSTYLE

Genera un informe sobre la adherencia a un estilo de codificación preestablecido al que se deben adherir todos los desarrolladores.

En nuestro caso la captura mostrada abajo nos resume que tenemos un total de **821 errores** del tipo del estilo de codificación y también los errores que tiene cada clase individualmente. Se explicará más en detalle más adelante.

Dominio_it1

Parent Project

Fritura

Project Documentation

Project Information

Project Reports

Surefire Report

JaCoCo

CPD

PMD

FindBugs

Checkstyle

Source Xref

Test Source Xref

Javadoc

Test Javadoc

Built by maven

Last Published: 2020-12-21 | Version: 0.0.1-SNAPSHOT

Dominio_it1

Checkstyle Results

The following document contains the results of Checkstyle 8.29 with sun_checks.xml ruleset. [xml](#)

Summary

| Files | Info | Warnings | Errors |
|-------|------|----------|--------|
| 8 | 0 | 0 | 821 |

Files

| File | I | W | E |
|--|---|---|-----|
| Dominio/Camarero.java | 0 | 0 | 34 |
| Dominio/DTOCamarero.java | 0 | 0 | 58 |
| Dominio/DTOMesa.java | 0 | 0 | 163 |
| Dominio/DTOReserva.java | 0 | 0 | 452 |
| Dominio/Estado.java | 0 | 0 | 16 |
| Dominio/Mesa.java | 0 | 0 | 23 |
| Dominio/Reserva.java | 0 | 0 | 41 |
| Dominio/Turnos.java | 0 | 0 | 34 |

JXR

Este plugin genera referencias cursadas entre las fuentes de un proyecto.

Es usado por otros plugins como **PMD** para enlazar la detección de bugs y línea de código. En las imágenes de a continuación se mostrarán todas las clases que se encuentren en esa iteración y si nos metemos dentro de ellas se mostrará la clase completa.

Package Dominio

| Class Summary |
|---------------|
| Class |
| Camarero |
| DTOCamarero |
| DTOMesa |
| DTOReserva |
| Estado |
| Mesa |
| Reserva |
| Turnos |

[View Javadoc](#)

```
1  package Dominio;
2
3  public class Reserva implements Estado {
4      /**
5       * id de la reserva.
6       */
7      private int id;
8      /**
9       * numero comensales reserva.
10     */
11     private int numComensales;
12     /**
13      * nombre reserva.
14      */
15     private String nombre;
16     /**
17      * Constructor clase Reserva
18      * @param pId
19      * @param pNumComensales
20      * @param pNombre
21      */
22     public Reserva(final int pId, final int pNumComensales, final String pNombre) {
23         this.setId(pId);
24         this.setNumComensales(pNumComensales);
25         this.setNombre(pNombre);
26     }
```

JAVADOC

Genera la documentación html del código basado en la herramienta Javadoc de Java.

Así, en nuestro proyecto tendríamos un informe como el fragmento que se muestra en la imagen siguiente donde tendríamos los métodos de los que consta junto con su comentario de **Javadoc**. Esto facilita mucho el mantenimiento a futuro al simplemente tener que consultar este informe para poder saber la utilidad de cada elemento del código.

| Constructors | | |
|---------------|-------------|--|
| Constructor | Description | |
| DTOCamarero() | | |

Method Summary

| All Methods | | |
|-------------------|---|------------------------|
| Static Methods | | |
| Concrete Methods | | |
| Modifier and Type | Method | Description |
| static void | leerAvisos(java.lang.String camarero, java.util.LinkedList<java.lang.String> lista) | Metodo leerAvisos. |
| static void | leerCamareroId(int id, java.util.LinkedList<Camarero> lista) | Metodo leerCamareroId. |
| static void | leerCamareros(java.util.LinkedList<Camarero> lista) | Metodo leerCamareros. |

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

MEJORAS DEL PROYECTO

Nos centraremos específicamente en los errores que nos muestra el plugin del **checkstyle**. El motivo de sólo corregir los errores del checkstyle es debido a que si en un futuro, nuestro programa fuera reutilizado por otro equipo se encontraría todo el código en condiciones correctas para comprenderlo y poder desarrollar sobre él.

Decidimos corregir los errores correspondientes al módulo del **dominio de la iteración 1** debido a que creemos que es la base respecto a todo el proyecto. En la imagen mostrada abajo, se reflejarán los **errores de dicho módulo antes de corregirlos**, no mostraremos todos los que son porque ocuparían demasiado, pero mostraremos el total al principio y al final y cómo van decrementando cuando los corrijamos.

Summary

| Files | Info | Warnings | Errors |
|-------|------|----------|--------|
| 8 | 0 | 0 | 821 |

Files

| File | I | W | E |
|--------------------------|---|---|-----|
| Dominio/Camarero.java | 0 | 0 | 34 |
| Dominio/DTOCamarero.java | 0 | 0 | 58 |
| Dominio/DTOMesa.java | 0 | 0 | 163 |
| Dominio/DTOReserva.java | 0 | 0 | 452 |
| Dominio/Estado.java | 0 | 0 | 16 |
| Dominio/Mesa.java | 0 | 0 | 23 |
| Dominio/Reserva.java | 0 | 0 | 41 |
| Dominio/Turnos.java | 0 | 0 | 34 |

Profundizando un poco más en algunos errores que serán corregidos, vemos de gran necesidad **renombrar algunas variables** cuyos nombres anteriores podrían llevar a confusión debida a conflictos con algunos atributos de clase los cuales tenían el mismo nombre, así hemos decidido, por convenio que las variables que se reciban por parámetro en métodos, constructores, etc., tendrán como nombre el suyo propio además de estar precedido por una p minúscula, señalando que esa variable tiene su origen como parámetro. Además, estos se han declarado como **final** ya que lo que llega como parámetro es una copia de la dirección de memoria.

También, se ha decidido quitar algunos **imports** que finalmente no han sido utilizados con el fin de hacer el programa algo más ligero al no tener que cargarlos.

Por último, también se proveerá a cada método y variable de un pequeño **comentario en Javadoc** con el mismo objetivo de facilitar la comprensión a futuros empleados que tengan que desarrollar sobre el mismo código.

| | | | | |
|-------|---------|----------------------|-----------------------------------|---|
| Error | javadoc | MissingJavadocMethod | Missing a Javadoc comment. | 8 |
| Error | misc | FinalParameters | Parameter codigo should be final. | 8 |
| Error | coding | HiddenField | 'codigo' hides a field. | 8 |

Así, por ejemplo, una parte del código la cual hemos mejorado sería la **clase Reserva**, a continuación se muestra una imagen de ejemplo de las mejoras acometidas. Como podemos apreciar, se han aplicado las técnicas antes comentadas, como la **inclusión de Javadoc**, el **cambio de nombre al parámetro**, añadiendo una **p** al inicio o declarando el parámetro como **final**.

```
/**
 * Modificar numero de comensales de la reserva.
 * @param pNumComensales
 */
public void setNumComensales(final int pNumComensales) {
    this.numComensales = pNumComensales;
}
```

Todas estas mejoras y muchas más han sido llevadas a cabo sobre el módulo antes comentado, y volviendo a realizar los **reports** gracias a los **plugins**, podemos ver una gran mejora.

Dominio_it1

Last Published: 2020-12-22 | Version: 0.0.1-SNAPSHOT Dominio_it1

Parent Project
Fritura
Project Documentation
Project Information
Project Reports
Surefire Report
JaCoCo
CPD
PMD
FindBugs
Checkstyle
Source Xref
Test Source Xref
Javadoc
Test Javadoc
Built by:

Checkstyle Results

The following document contains the results of Checkstyle 8.29 with sun_checks.xml ruleset. [xml](#)

Summary

| Files | Info | Warnings | Errors |
|-------|------|----------|--------|
| 8 | 0 | 0 | 180 |

Files










| File | I | W | E |
|--------------------------|---|---|-----|
| Dominio/Camarero.java | 0 | 0 | 16 |
| Dominio/DTOCamarero.java | 0 | 0 | 3 |
| Dominio/DTOMesa.java | 0 | 0 | 37 |
| Dominio/DTOReserva.java | 0 | 0 | 106 |
| Dominio/Estado.java | 0 | 0 | 5 |
| Dominio/Mesa.java | 0 | 0 | 2 |
| Dominio/Reserva.java | 0 | 0 | 7 |
| Dominio/Turnos.java | 0 | 0 | 4 |

Como podemos ver, el total de errores del módulo ha decrecido de una manera abrupta, teniendo ahora un total de **180 errores** comparados con los **previos 821**. Además, a nivel de archivo, vemos como por ejemplo el **DTO de Reserva** ha pasado de **452 errores a 106**.

FUTURAS MEJORAS

Nuestro equipo ha decidido que en un futuro, cuando no tengamos que realizar otros proyectos a este, nos dispondremos a mejorar nuestro proyecto Fritura, ya que hemos comprobado los errores que quedan y no nos supondría una **gran complicación resolverlos**, como por ejemplo en esta clase **DTOReserva** la cual solo tendríamos que quitar los import's innecesarios y las variables que no están usadas.

Dominio/DTOReserva.java

| Rule | Violation | Priority | Line |
|---|---|----------|------|
| UnusedImports  | Avoid unused imports such as 'java.sql.Date' | 4 | 3 |
| UnusedImports  | Avoid unused imports such as 'java.sql.Time' | 4 | 6 |
| UnusedImports  | Avoid unused imports such as 'java.sql.Timestamp' | 4 | 7 |
| UnusedImports  | Avoid unused imports such as 'java.util.Timer' | 4 | 15 |
| UnusedImports  | Avoid unused imports such as 'java.util.TimerTask' | 4 | 16 |
| UnusedLocalVariable  | Avoid unused local variables such as 'res2'. | 3 | 94 |
| UnusedLocalVariable  | Avoid unused local variables such as 'subConsulta'. | 3 | 141 |
| UnusedLocalVariable  | Avoid unused local variables such as 'rs'. | 3 | 144 |
| UnusedLocalVariable  | Avoid unused local variables such as 'turno'. | 3 | 265 |

- En cuanto al **mantenimiento correctivo**, nuestro objetivo es obtener un servidor para la base de datos más potente debido a que en algunas operaciones resulta algo lento, lo cual no se puede permitir ya que el entorno donde se va a utilizar el producto software requiere de rapidez.
- En cuanto al **mantenimiento perfectivo**, un requisito nuevo que nos ha pedido la empresa una vez desplegada la primera release, es la opción de despedir a un empleado que esté trabajando en este momento.
- En cuanto al **mantenimiento preventivo**, como se ha explicado antes se desea eliminar variables e import's no usados para mejorar su posterior mantenimiento, mejorar su legibilidad, etc.
- En cuanto al **mantenimiento adaptativo**, una acción importante para nuestro proyecto será intentar hacer que nuestra aplicación se adecúe en entornos distribuidos de red para que sea un sistemas escalable y tolerante a fallos, que será importante en cuanto la empresa que nos contrató comience a crecer.