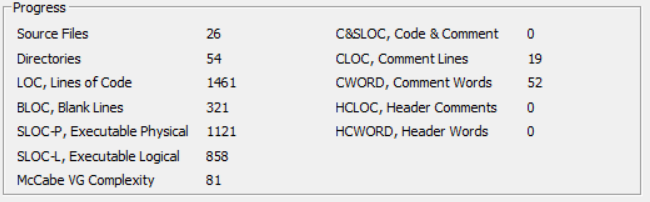
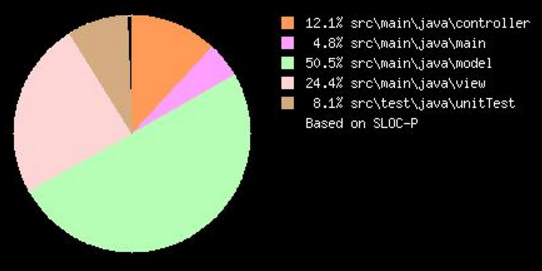
Plan de Remoción de defectos

Utilizando la herramienta LocMetrics, determinamos que el trabajo final de Ingeniería de Software tiene un total de 1461 líneas de código.



Del total, existen 321 líneas que están en blanco y 19 se corresponden a líneas de comentarios puros, lo que deja un total de 1121 líneas de código ejecutables.

Estas se encuentran distribuidas dentro del proyecto de la siguiente manera:



Utilizando la definición de Capers Jones, para estimar la cantidad de defectos esperables en el proyecto: “1 error por cada 10 oportunidades”. Podemos decir que de las 1121 de código ejecutable se espera encontrar al menos 112 errores.

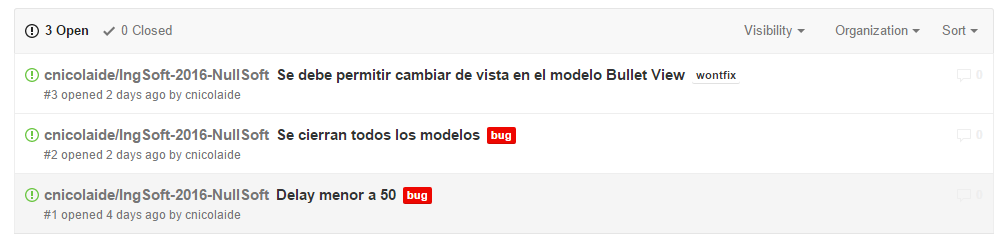
Lo cual podría resultar un poco exagerado, para mejorar la estimación utilizamos valores históricos de tamaños similares de inspecciones de código e ingeniería inversa de los defectos reales luego del release, determinando un total de 47 defectos esperables dentro del proyecto.

La Cessi, Cámara de la Industria Argentina del Software, reveló los resultados de su encuesta, realizada a 150 empresas de todo el país y con el único objetivo de conocer cuánto pagan a sus más de 12.000 desarrolladores de software.[[1]](#footnote-1)

Los datos obtenidos, arrojan un salario bruto mensual promedio de $13.100 para la categoría de programadores sin experiencia previa (Junior), $17.598 con alguna experiencia (Semisenior) y $23.500 para los programadores con experiencia (Senior).

Teniendo en cuenta que el proyecto fue desarrollado por estudiantes, es decir, programadores sin experiencia procedemos a calcular el costo por hora de trabajo. Teniendo en cuenta una jornada laboral completa de 8hs, 5 días a la semana, 4 semanas al mes. Obtenemos un total de 160 hs/mes. Por lo tanto $13.100/160hs arroja un precio por hora de $82 la hora de trabajo.

Comparando este valor con la cantidad de errores encontrados al momento de la entregar el release en la materia ingeniería de software (apenas 3), podemos percibir que probablemente existan una gran cantidad de errores sin descubrir y/o corregir.



Esto con seguridad se traducirá en mayores costos a la hora de corregir los defectos.

Revisión de los requerimientos

Revisión del diseño

Revisión de código

Análisis estático de código

Pruebas Unitarias

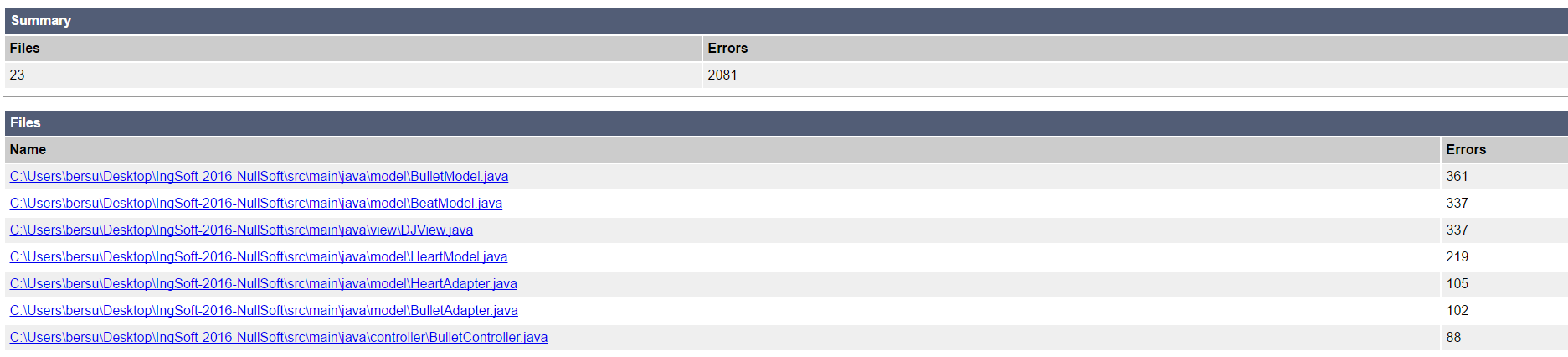
9. Integración continúa

Como herramienta para realizar integración continua, se decidió utilizar TravisCI, al cual se le instalaron los siguientes plugins.

**Check Style**

Descripción: Este plugin es una herramienta de desarrollo para ayudar a los programadores escribir código Java que se adhiere a un estándar de codificación. Automatiza el proceso de verificación de código, y nos libera de esta aburrida (pero importante) tarea. Es ideal para proyectos que quieren hacer cumplir un estándar de codificación.

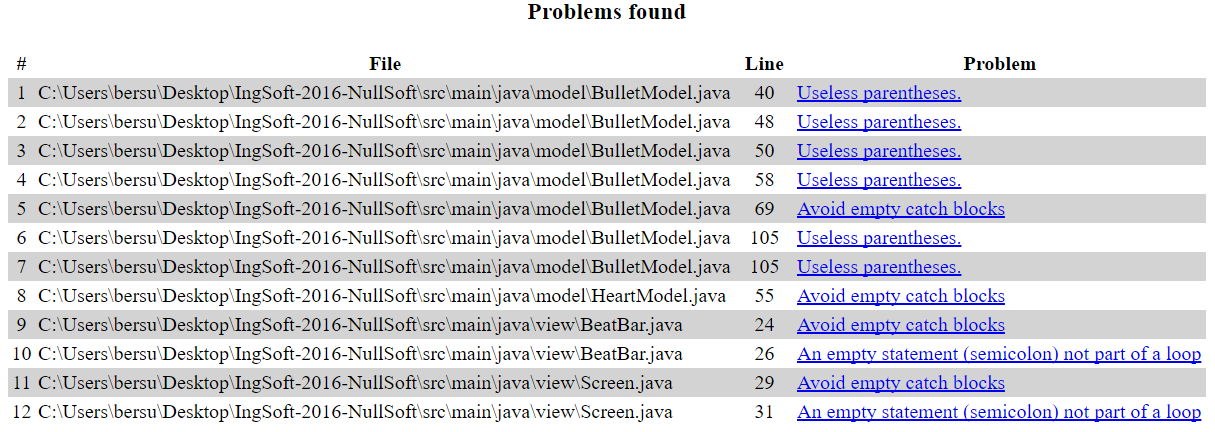
Reporte:



**PMD**

Descripción: Este plugin es un analizador de código fuente. A través del cual se encuentra defectos habituales de programación como las variables inutilizadas, bloques catch vacíos, la creación de objetos innecesarios, y así sucesivamente. Es compatible con Java, JavaScript, Salesforce.com Apex, PLSQL, Apache Velocity, XML, XSL.

Reporte:



**CPD**

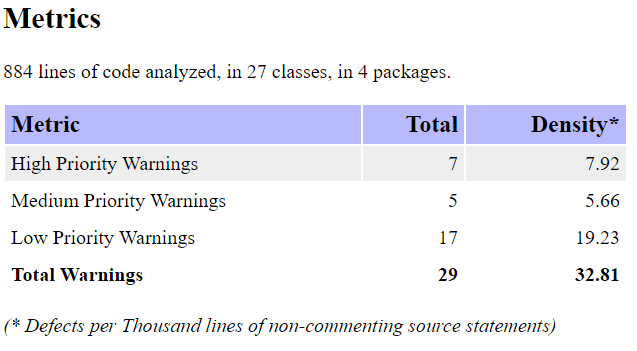
Descripción: Este plugin ofrece un mecanismo de detección automática de copy/paste de líneas de código dentro del proyecto. Es ideal para mantener un control automático sobre esta práctica habitual pero no recomendada (reutilizar bloques de código).

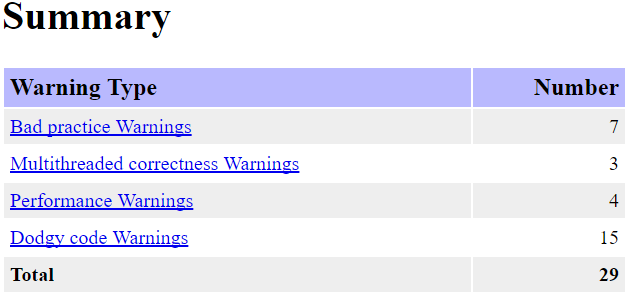
Reporte:

**FindBugs**

Descripción: Este plugin de gradle utiliza el análisis estático para buscar errores en el código Java.

Reporte:





**Jacoco**

Descripción:

Reporte:

En todos los plugins el build falla si la cantidad de errores supera la cantidad que había en el build anterior. Además de esto si hay problemas en la compilación, también fallara el build.

Para hacer el chequeo se debe correr el comando gradlew check

1. http://www.infobae.com/2016/04/14/1804380-cuanto-gana-un-programador-la-argentina/ [↑](#footnote-ref-1)