Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Soporte para el estudio de la detección de defectos de diseño en proyectos de código abierto

Autor:

Mario Cartón González

Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Soporte para el estudio de la detección de defectos de diseño en proyectos de código abierto

Autor:

Mario Cartón González

Tutor:

Yania Crespo González-Carvajal

A mi madre…

Resumen

El objetivo de este trabajo de fin de grado es doble, por una parte, se realizará una herramienta de apoyo a tareas de investigación sobre detección de defectos de diseño. Con esta herramienta se consigue disponer automáticamente de un dataset para investigar sobre los defectos de diseño que se detectan en proyectos de código libre alojados en repositorios públicos. Por otra parte, esta herramienta también tendrá su utilidad de cara a los propios desarrolladores de los proyectos de código abierto. Se les proporcionará a estos desarrolladores los informes obtenidos de detección de defectos de diseño en sus proyectos. La herramienta proporcionará una forma fácil para que los desarrolladores aporten retroalimentación una vez analizados los informes de detección presentados. De esta manera, la herramienta puede obtener información subjetiva y validación de parte de los desarrolladores que le permita aprender en el futuro de estos casos. Con esto se pretende una simbiosis en la que ambas partes se benefician, los investigadores y los desarrolladores de los proyectos de código libre.

La herramienta tiene tres partes bien diferenciadas que deben estar apropiadamente integradas. La primera parte consiste en la automatización de las tareas necesarias para obtener de los repositorios públicos la información de proyecto y el proyecto en sí mismo, analizarlo de forma automática mediante herramientas de detección de defectos de diseño, almacenar y analizar resultados. La segunda parte consiste en el desarrollo de una aplicación web de cara a ser utilizada por los desarrolladores de los proyectos de código abierto de forma que estos puedan registrar sus proyectos y registrarse a sí mismos como los desarrolladores responsables de estos proyectos, enlazando la información tanto de desarrolladores como de proyectos con el repositorio público donde se alojan. Por último, como tercera parte, la herramienta deberá realizar la generación automática de los informes de detección que se incorporan a la aplicación web en forma de encuestas interactivas en las cuales los desarrolladores no solamente consultan los resultados de la detección, sino que pueden con facilidad aportar retroalimentación.

Para esta primera versión de la herramienta se han seleccionado tres herramientas de detección de defectos de diseño pmd, ptidej DÉCOR y JDeodorant. Con un enfoque de desarrollo evolutivo, comenzando por incorporar la herramienta pmd, se irán añadiendo las otras dos herramientas preseleccionadas, realizando un diseño que permitirá añadir en futuras versiones nuevas herramientas en este entorno integrado. De la misma forma, se realizará el trabajo de integración de la obtención de datos y código de proyectos de los repositorios públicos de código abierto. Se ha preseleccionado GitHub y SourceForge, comenzando por GitHub y generalizando a partir de ahí para facilitar la integración a futuro de otros repos.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

1.1 CONTEXTO

La calidad del software es una preocupación a la que se dedica mucho esfuerzo. Sin embargo, el software casi nunca es perfecto. Todo proyecto tiene como objetivo producir software de la mejor calidad posible que cumpla, y si puede, que supere las expectativas de los usuarios. Este proyecto se centra en la calidad del software respecto del diseño.

Con el trabajo en grandes proyectos software, la calidad en el diseño cobra una gran importancia para la lectura y la simplificación de código. Para asegurar esta calidad hay una serie de buenas prácticas, las cuales llamamos patrones de diseño. Por el contrario, existe también un conjunto de malas prácticas a las que llamamos defectos de diseño o *Design Smells*. En este proyecto nos centraremos en su detección.

Dentro del mundo de los *Design Smells*, se puede encontrar una gran diversidad de opiniones en cuanto a su detección y/o existencia dentro de un proyecto software. A la hora de realizar análisis con distintas herramientas de detección, no hay una amplia coincidencia entre los resultados de detección obtenidos. Esta diferencia de criterio no solo se encuentra en las herramientas de detección, también se encuentra en el ámbito profesional. Los profesionales del sector también discrepan en el criterio de elección para definir si ciertas prácticas son o no *Design Smell*. Se deduce de este modo, que en la detección *de Design Smells* influye un alto grado de subjetividad al que, en esta práctica, se intenta dar respuesta.

1.2 OBJETIVOS

En este proyecto se ha intentado afrontar el problema de *Design Smells* que tienen muchos proyectos de código abierto. En este caso, los proyectos de código abierto escritos en lenguaje Java alojados en repositorios públicos. Así que, para la detección de estos *Design Smells* se plantea el desarrollo de una aplicación web que tiene una doble finalidad. Por un lado, la detección de defectos de diseño mediante herramientas automatizadas y, por otro lado, la validación (o no) por parte de los desarrolladores de estos defectos de diseño detectados.

Gracias a estas dos acciones de detección y validación, en un futuro, se podrá realizar un trabajo de minería de datos en el que se pueda aprender la correlación que existe entre la detección por parte de las herramientas y por parte de los profesionales en el ámbito de la informática, en qué situaciones coinciden sus criterios y en cuales discrepan.

1.3 MOTIVACIÓN

Este proyecto surgió como una propuesta de mi tutora Yania para la realización de este Trabajo Fin de Grado. Se consultó a Yania la posibilidad de que fuera mi tutora de Trabajo Fin de Grado debido a que resulta interesante su campo de trabajo y las asignaturas que imparte en la Universidad. Es decir, todo lo referente al campo de la Ingeniería del Software y más concretamente al Diseño Software.

La aplicación que se va a desarrollar es una herramienta de ayuda a la investigación de defectos de diseño en proyectos de código abierto.

1.4 ESTRUCTURA DE LA MEMORIA

Esta memoria recoge la explicación del marco teórico que da pie al desarrollo de la aplicación y la documentación del desarrollo de la misma.

- **Design Smells, descripción y herramientas para su detección:** En este apartado se exponen los aspectos teóricos del trabajo y que herramientas se consideran representativas para la detección junto con su explicación.

- **Herramientas de detección frente a la detección humana y grado de acuerdo en la evaluación de Design Smells:** En este apartado se exponen algunas de las razones que motivan el desarrollo de este trabajo.

- **Proyecto software:** En este apartado se explica cómo se ha decidido desarrollar el proyecto, la planificación inicial, la toma de requisitos y como ha quedado el proyecto tras cada iteración y el resultado final del mismo.

- **Conclusiones:** En este apartado se tratan aspectos y temas referidos al final del proyecto.

- **Bibliografía:** Referencias bibliográficas del proyecto.

- **Anexos:** Documentos adicionales. Manual de usuario, manual de instalación y manual de mantenimiento y extensibilidad.

2 DESIGN SMELLS. DESCRIPCIÓN Y HERRAMIENTAS DE DETECCIÓN

2.1 INTRODUCCIÓN

En el ámbito de la ingeniería y desarrollo software, la calidad del software cada vez cobra una mayor importancia. Durante la evolución de un sistema software, su estructura y código crece y sufre modificaciones. Estas dos acciones pueden llevar a una degradación del código empeorando su mantenibilidad y reusabilidad. Estos problemas se conocen como defectos de código o *Code Smells*. Estos defectos en el código no son bugs, es decir, no son técnicamente incorrectos y no impiden que el programa deje de funcionar. En lugar de ello indican diferencias que pueden influir en el retraso del desarrollo e incrementan el riesgo de errores.

2.2 EL TÉRMINO SMELL

A continuación, se explica el término *smell* referido a problemas de diseño software. El término *smell* describe una situación donde hay indicios de que se puede tener un problema. Pero para determinar si existe realmente o no un problema hay que examinar la situación con más detalle. El término *smell* se puede utilizar para describir una mala estructura en un programa que es evidente y obvio que es perjudicial en sí mi misma. Al mismo tiempo también se puede emplear para describir una situación de diseño que no es tan obvia ni tan perjudicial, o cierta estructura no es tan desventajosa por sí misma, pero puede indicar la existencia de un problema en otro lugar del diseño. De cualquier manera, el término *smell* expresa bastante bien los distintos niveles en la escala de daño que la estructura del software asociado puede representar. Este término también expresa la idea de que el diseño debe ser cuidadosamente analizado para confirmar la ocurrencia de un posible problema.

2.3 CODE SMELL

Un *Code Smell* se define según Martin Fowler [4] como "*un indicador o un síntoma en el código fuente de un sistema, que posiblemente indica un problema más profundo en el sistema*". Continúa comentando "*En primer lugar un smell es por definición algo fácilmente detectable. Un método largo es un buen ejemplo (..) En segundo lugar, los smells no indican necesariamente un problema. Algunos métodos largos están bien. Hay que mirar más profundamente para ver si realmente hay algún problema subyacente ahí. Los smells no son inherentemente malos, pero frecuentemente son indicadores de un problema*".

Una estructura de código sospechosa puede que directamente no provoque daños (en términos de fallos y errores de ejecución), pero tiene un impacto negativo en la estructura global del sistema y, como consecuencia, sobre sus factores de calidad. La presencia de *Code Smells* es un indicador del desorden en el diseño de un sistema, dificultando su comprensión y mantenimiento. Además, su presencia puede avisar de problemas de desarrollo como errores en la elección de la arquitectura o incluso malas prácticas de gestión. Un buen ejemplo de *Code Smell* es la presencia de código duplicado. Esta situación aparece cuando un conjunto de instrucciones, una expresión, un algoritmo etc. es utilizado en muchos lugares del sistema. Este *Code Smell* hace que la estructura del sistema sea más difícil de mantener, convirtiéndose en un foco de futuros fallos de ejecución en el caso de tener que cambiar una copia del código duplicado.

2.4 DESIGN SMELL

El término *Design Smell* se puede referir como concepto similar al de *Code Smell* pero de una manera más general. Este término cubre el gran rango de problemas relacionados con la estructura del software en la parte del diseño. Una de las muchas definiciones de *Design Smell* puede ser: "*soluciones pobres de diseño para problemas software generalizados y recurrentes*" (N.Moha). También podemos hablar de *Design Smell* como estructuras en el diseño que indican violaciones de los principios fundamentales del diseño con un negativo impacto en la calidad del diseño. Las violaciones de estos principios hace que el diseño sea frágil y difícil de mantener. Si no se identifican estos defectos de diseño y se aplica una apropiada refactorización, se contribuye a la acumulación de una deuda técnica. Esta deuda técnica tiene un factor negativo en la calidad del software en desarrollo y compromete la viabilidad a largo plazo del proyecto.

La detección de los *smells* puede reducir sustancialmente el coste de las actividades posteriores de desarrollo y fases de mantenimiento. Pero las actividades de detección, sobre todo en sistemas grandes, tienen un elevado coste en tiempo y recursos, que además dejan un amplio espacio para la interpretación. Para la detección de los *Design Smells* normalmente se suelen emplear herramientas automáticas (algunas de estas herramientas se describirán más adelante). Estas herramientas suelen hacer detecciones, que en muchos casos entre las propias herramientas no coinciden con un alto grado de acuerdo, a no ser que el error detectado sea muy evidente. Pero es que incluso entre las propias herramientas de detección y los programadores y/o los encargados de mantenimiento del código hay discrepancias. Todo este tema de acuerdo y desacuerdo en la detección lo desarrollaremos en el siguiente capítulo.

Robert C. Martin [14] se refiere a los *Design Smells* como defectos de alto nivel que causan la decadencia de la estructura del sistema software, que puede ser detectada cuando el software empieza a mostrar los siguientes problemas:

- **Rigidez:** El diseño es difícil de cambiar porque cada cambio implica otros muchos cambios en otras partes del sistema.
- **Fragilidad:** El diseño es fácil de romper. Los cambios causan al sistema roturas en lugares que no tienen relación conceptual con la parte que fue cambiada.
- **Inmovilidad:** Es difícil descomponer el sistema en componentes que puedan ser utilizados en otros sistemas.
- **Viscosidad:** Hacer las cosas bien es más difícil que hacerlas mal. Es más fácil utilizar atajos rápidos.
- **Complejidad innecesaria:** El sistema está supra-diseñado, contiene infraestructuras que no añaden beneficio directo.
- **Repetición innecesaria:** El sistema contiene estructuras repetitivas que podrían ser unificadas en una única abstracción.
- **Opacidad:** El sistema es difícil de leer y comprender y no expresa correctamente su objetivo

Los diferentes *Design Smells* afectan al software en distintos niveles de granularidad, desde métodos a toda la arquitectura software. Esta puede ser una de las razones de la cantidad de terminología acerca de los *smell* que han acuñado los distintos autores. Por ejemplo:

- **Antipatterns:** Un antipatrón es una forma literaria que describe una solución recurrente que genera consecuencias negativas. Dicho de otra forma, se podría describir como un patrón de diseño que invariablemente conduce a una mala solución para un problema. Un antipatrón suele ser el resultado de una decisión equivocada sobre cómo resolver un determinado problema, o bien, la aplicación correcta de un patrón de diseño en un contexto equivocado. Catálogo de *antipatterns*:

Antipatterns		
Abstraction Inversion	Floating Point Currency	Project Mismanagement?
Accidental Complexity	Floating Point Fractions	Ransom Note Antipattern?
Accidental Inclusion	Fool Trap	Reinvent The Wheel
Acme Pattern	Functional Decomposition	Roll Your Own Database Requirements Tossed over the Wall
Alcohol Fueled Development	Game Over You Lose?	Rube Goldberg Machine
Ambiguous View point	Geographically Distributed Development	Scape Goat
Analogy Breakdown Antipattern	Give Me Estimates Now	Secret Society
Analysis Paralysis	Glass Wall	Shoot The Messenger
An Athena	Golden Hammer	Single Function Exit Point
Anchored Helper?	Ground Hog Day Project	Smoke and Mirrors
Architecture by Implication	Heir Apparent	Software Merger
Asynchronous Unit Testing	Hero Culture	Spaghetti Code
Autogenerated Stove pipe Antipattern	Hidden Requirements	Specify Nothing
Bear Trap	Idiot Proof Process.	Standing on the Shoulders of Midgets
Big Ball of Mud	IfItIs Working don't Change	Stovepipe Enterprise?
Blame Storming	Import Beast?	Stovepipe AntiPattern
Blow hard Jamboree	Implementation Inheritance	String without Length
Boat Anchor	Input Kludge	Sumo Marriage
Cargo Cult	Intellectual Violence	SweepIt under the Rug Antipattern
Cascading Dialog Boxes Antipattern	Internationalizationis Fully Supported?	Swiss Army Knife
Configuration Abomination?	Irrational Management?	That's not Really An Issue
Corn Cob	It's an Operator Problem	The Blob
Cover your Assets	Job Keeper	The Customers are Idiots
Creeping Featuritis	Jumble Antipattern	The Feud?

Tabla 1. Catálogo de antipatterns. Parte 1

Antipatterns		
Copy and Paste Programming	Junkyard Coding	The Grand old Duke Of York
Dead End	Kill Two Birds With One Stone	They Understood Me
Death by Planning	Kitchen Sink Design	Thrown over the Wall
Decision by Arithmetic	Lava Flow	Tower of Voodoo
Design by Committee	Leading Request	Train the Trainer
Design For The Sake Of Design	Magic Container Manager	Trusting Souls?
Discordant Reward Mechanisms	Controls Process	Untested but Finished
Doer and Knower	Mushroom Management	Vendor LockIn
Dry Waterhole	Naked Documents?	Viewgraph Engineering
Egalitarian Compensation	Nationalism	Walking Througha Mine Field
EmailIs Dangerous	Net Negative Producing Programmer	Warm Bodies
Emperors New Clothes	Not Invented Here	We are Idiots
Empire Building	Null Flag	Wolf Ticket
Exception Funne	Over Generalization of Business Logic	Yet Another Meeting will SolveIt
False Economy	Path of Least Resistance	Yet Another Programmer
False Surrogate Endpoint	Passing Nulls to Constructors	Zero Means Null
Fear of Success	Plug Compatible Interchangeable Engineers	
Fire Drill	Poltergeists	

Tabla 2. Catálogo de antipatterns. Parte 2

- **Bad Smells:** Los *Bad Smells* se definen como síntomas de errores de diseño que indican la necesidad de refactorización. La detección de *Bad Smells* puede llevarse a cabo, por ejemplo, mediante la comprobación de métricas que puede ayudar al programador a aplicar la refactorización conveniente. A continuación, se muestra una lista de *Bad Smells*:

Bad Smell	Grupo	Granularidad
Data Clumbs	Bloater	Class
Large Class	Bloater	Class
Long Method	Bloater	Method

Tabla 3. Catálogo de Bad Smells. Parte 1

Bad Smell	Grupo	Granularidad
Long Parameter List	Bloater	Method
Primitive Obsession	Bloater	Method
Alternative Classes with Different Interfaces	O-O Abusers	Method
Refused Bequest	O-O Abusers	Class
Switch Statements	O-O Abusers	Method
Temporary Field	O-O Abusers	Class
Divergent Change	Change Prevents	Class
Parallel Inheritance Hierarchies	Change Prevents	Class
Shotgun Surgery	Change Prevents	Class
Data Class	Dispensables	Class
Duplicated Code	Dispensables	Method
Lazy Class	Dispensables	Class
Speculative Generality	Dispensables	Class
Feature Envy	Couplers	Method
Inappropriate Intimacy	Couplers	Class
Message Chains	Couplers	Class
Middle Man	Couplers	Class
Comments	Not Defined	Class
Incomplete Library	Not Defined	Class

Tabla 4. Catálogo de Bad Smells. Parte 2

- **Disharmonies:** Las *disharmonies* son defectos de diseño que afectan a entidades individuales tales como clases y métodos. La particularidad de estas es que su efecto negativo sobre el diseño se puede considerar como un elemento aislado, es decir, son problemas que se encuentran en determinadas partes del código y únicamente afectan a estas partes, no viéndose alteradas de manera indirecta otras partes del código. Se pueden identificar tres aspectos distintos que contribuyen a la identidad de las *disharmonies*: su tamaño, su interfaz, y su aplicación. Para cada uno de ellos se define una regla:

    - Las operaciones y las clases deben tener un tamaño armonioso, es decir, se debe evitar el excesivo tamaño en las extremidades.
    - Cada clase debe presentar su identidad, es decir, su interfaz, por un conjunto de los servicios que tienen una responsabilidad única y que proporcionan un comportamiento único.
    - Los datos y las operaciones deben colaborar armónicamente en la clase a la que pertenecen semánticamente.

Disharmonies	
Identity Disharmonies	Collaboration Disharmonies
- Rules of Identity Harmony - Overview of identity Disharmonies - God Class - Feature Envy - Data Class - Brain Method - Brain Class - Significant Duplication - Recovering from Identity Disharmonies	- Collaboration Harmony Rule - Overview of Collaboration Disharmonies - Intensive Coupling - Dispersed Coupling - Shotgun Surgery - Recovering from Collaboration Disharmonies.

Tabla 5. Catálogo de Disharmonies

- **Bad Coding Style:** Son violaciones de las convenciones de estilo o estándares de código ya establecidos.
- Otros como **Bad Pattern Usage** (Referente a la mala aplicación de un patrón en el código a analizar) o **Metrics Warnings** (La herramienta detecta defectos a través del cálculo de métricas).

A pesar de la terminología diferente utilizada en la literatura, todos los términos mencionados tienen en común que describen problemas relacionados con malos diseños.

2.5 MODELADO DE DESIGN SMELLS

Antes de exponer las herramientas consideradas candidatas a incluir en el proyecto, se va a exponer las características que modelan a los *Design Smells* y que una herramienta puede abordar. Esta sección está extraída de [15]. Para representar el *Design Smell*, se divide en cuatro ramas de funciones agrupadas, y una función opcional que especifica si la relación de herencia es relevante para el *smell* o no.

- **Tipo de *smell***: Esta característica describe la naturaleza del *smell* a abordar. Puede abarcar desde simples advertencias de métricas a problemas de diseño descritos en detalle en los catálogos de *Design Smells*. Los problemas en el diseño del sistema se especifican en término de simples advertencias o violaciones cada vez que una métrica determinada excede un umbral determinado. Las herramientas que utilizan este tipo de definición de *smell* suelen producir listas de filtrado de las entidades que violen de manera relativa o absoluta el umbral de las métricas.

- **Granularidad:** Esta característica se utiliza para describir los distintos tipos de entidades que participan en los *smells* soportados. Por ejemplo, para el código fuente de java, los diferentes niveles de granularidad podrían ser sistema, subsistema, paquete, clase y método. Para otro tipo de código esta subcaracteristica no se tiene en cuenta ya que no sigue la misma jerarquización de Java y no se puede detallar un nivel de granularidad por desconocimiento del lenguaje.

- **Relación entre entidades:** Se refiere a si el *smell* se produce entre entidades del mismo tipo (inter) o entre entidades de distintos tipos (intra).

- **Herencia:** Las herramientas que soportan la herencia lo hacen como una característica opcional y tratan la gestión de *smells* que implican relaciones de herencia entre las entidades consideradas.

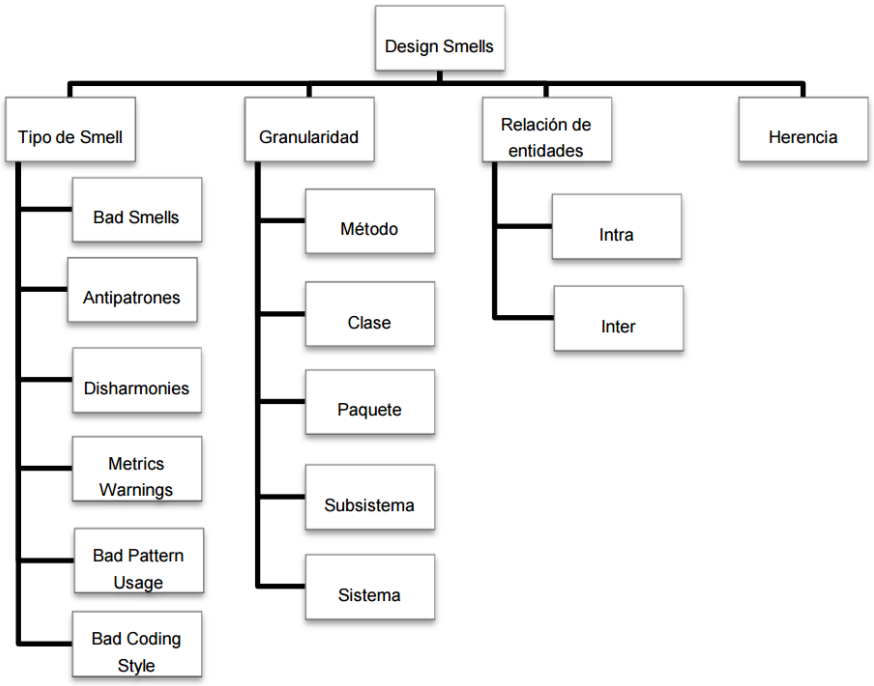


Figura 1. Modelado de Design Smell

2.6 HERRAMIENTAS DE DETECCIÓN

Una vez expuesto cómo se caracterizan y modelan los *Design Smells*, se va a proceder a presentar las herramientas de detección que se han considerado que eran buenas candidatas a incluir en la aplicación que se va a desarrollar. Estas herramientas son: PMD, jDeodorant, Together, iPlasma y DECOR. La elección de estas herramientas ha venido dada por mi tutora Yania. Como ha realizado la selección estas herramientas se explica en uno de sus artículos [2]:

"*Se ha utilizado como punto de partida un informe técnico [15] del grupo GIRO de la Universidad de Valladolid. En éste se evalúan un amplio conjunto de herramientas de detección y corrección de Design Smells. A partir de este informe, se prescinde de las herramientas que realizan corrección de Design Smells pero no detección, quedando 24 herramientas. Del estudio de sus principales características, seleccionamos aproximadamente el 50%, aquéllas que permiten detectar Design Smells en código Java. Posteriormente filtramos aquellas que tienen en común un grupo de Design Smells y que pueden procesar un proyecto desarrollado con la misma versión java. (...) se seleccionó iPlasma como representante del grupo de 3 herramientas (iPlasma, inFusion, inCode) del mismo origen (...) y se introdujeron las herramientas más citadas en el estado del arte actualmente: DECOR y PMD, quedando una selección de 5 herramientas formadas por: Together, JDeodorant, iPlasma, PMD y Decor*".

- **PMD:** PMD es una herramienta de código abierto (opensource). Es un analizador estático de código fuente, concretamente analiza código fuente java basado en unas normas de evaluación que se han habilitado durante la ejecución. La herramienta viene con un conjunto predeterminado de las reglas que pueden ser utilizadas para descubrir errores comunes de desarrollo tales como variables que nunca se utilizan, objetos que son innecesarios, etc. Además, PMD también permite a los usuarios ejecutar análisis personalizados que les permitan elaborar nuevas normas de evaluación de una forma cómoda. La herramienta viene con una interfaz de línea de comando relativamente fácil de usar y, al mismo tiempo, también se puede integrar con diversos entornos de desarrollo populares tales como eclipse. PMD está centrado en analizar código Java.

- **jDeodorant:** jDeodorant es un plugin de Eclipse que identifica problemas de diseño software conocidos como *Bad Smells* y los resuelve mediante la aplicación de refactorizaciones apropiadas. jDeodorant emplea las nuevas metodologías con el fin de identificar *Design Smells* y sugerir las refactorizaciones convenientes para resolverlos. En la actualidad, la herramienta identifica cuatro tipos de *Bad Smells*: *Feature envy, Type checking, Long Method* y *God Class*.

- **Together:** Together permite analizar, diseñar e implementar un software flexible y fácil capaz de mantener arquitecturas que deban ser modificadas cuando cambien algunos de los requerimientos. Together es capaz de reconocer tres tipos de *smells*. Dos de ellos son *bad coding style* y *bad smells/disharmonies*.

- **iPlasma:** "*iPlasma es un entorno integrado para el análisis de calidad de los sistemas software orientados a objetos que incluye soporte para todas las fases de análisis necesarias: desde la extracción de modelos (incluyendo el análisis escalable para C\++ y Java) hasta el análisis basado en métricas de alto nivel, o la detección de la duplicidad de código. iPlasma tiene tres grandes ventajas: extensibilidad del análisis asistido ('supported analysis'), integración con otras herramientas de análisis y escalabilidad, como fue usado en el pasado para analizar los proyectos a gran escala con millones de líneas de código (por ejemplo, Eclipse y Mozilla)* "[21].

- **DECOR:** Ptidej (Pattern Trace Identification, Detection and Enhancement in Java) es un equipo que desarrolla teorías, métodos y herramientas para comprender, evaluar y mejorar la calidad de los sistemas software promoviendo el uso de lenguajes, patrones de diseño y patrones arquitectónicos. El equipo de Ptidej fue uno de los primeros en perseguir la identificación y detección de *Code* y *Design Smells*. Este tema de investigación condujo al desarrollo de varios enfoques y herramientas novedosas. DECOR es un plug-in de la tool suite de Ptidej.

  La descripción de DECOR (Defects, dEtection, CORrection) según el grupo Ptidej en su página web es: "*DECOR es un procedimiento cuya instanciación son técnicas de detección de Code and Design Smells. DETEX es nuestra instanciación de DECOR, permite la especificación y la detección de defectos tales como Code Smells y Antipatterns usando un vocabulario y un lenguaje unificado. Ptidej es el front-end del conjunto de herramientas para evaluar y mejorar la calidad de los programas orientados a objetos, programas orientados a objetos de ingeniería inversa, y la promoción de patrones. Ptidej integra DECOR así como algoritmos de visualización para facilitar la comprensión de los defectos detectados*" [16].

Todas estas herramientas (y otras tantas no descritas) realizan, en cierta medida, las mismas tareas: analizar código en busca de *Design Smells*. Pero ¿Todas estas herramientas producen los mismos resultados en los mismos proyectos? Si no lo hacen ¿Mantienen un grado de acuerdo entre ellas?, los resultados obtenidos ¿Concuerdan con lo que realmente piensan los programadores? es decir ¿Los programadores o los diseñadores de la aplicación respaldan y validan el resultado de las herramientas analizadoras? Todas estas cuestiones se van a tratar en el siguiente capítulo.

3 HERRAMIENTAS DE DETECCIÓN FRENTE A DETECCIÓN HUMANA. GRADO DE ACUERDO EN LA EVALUACIÓN DE DESIGN SMELLS

3.1 INTRODUCCIÓN

Como se ha visto en el capítulo anterior, se pueden encontrar diversas herramientas que detectan *Design Smells*. Pero se plantean las siguientes cuestiones: ¿Los resultados que se obtienen son los mismos, o al menos, con cierto grado de acuerdo? ¿Los defectos detectados son realmente defectos o los diseñadores y programadores decidieron realmente hacerlo así? Estas preguntas son las que motivan este Trabajo Fin de Grado. Aunque la aplicación que se va desarrollar no de respuestas estas preguntas, la aplicación se va a encargar de recopilar información para conseguir dar una respuesta en un futuro a estas preguntas.

La detección de los *Design Smells* está cargada de un grado de subjetividad notorio. En ciertos proyectos, muchas decisiones de diseño vienen dadas por el contexto o simplemente son el mejor camino para atajar un problema de diseño. Distintas investigaciones intentan aportar un poco de objetividad a estos problemas, o al menos intentan encuadrar la situación en la que se encuentra la disparidad de opiniones respecto a la detección de *Design Smells* por parte de herramientas (automaticas o semiautomaticas) y también por parte de profesionales del ámbito de la informática. En este capítulo se comentarán alguna de estas investigaciones y los resultados obtenidos, y cómo, a raíz de esto, surgen las motivaciones que llevan a realizar este proyecto.

3.2 INFORMACIÓN PREVIA. GOD CLASS Y FEATURE ENVY

En los artículos que comentamos en esta sección 3 se realiza el estudio sobre dos *Design Smells* concretos: *God Class* y *Feature Envy*. Estos dos *Design Smells* son los más populares en la literatura de este tipo. Se trata de dos *Design Smells* de distinta naturaleza. *God Class* es un *Design Smell* intra clase, sólo se necesita observar el código de dicha clase para detectarlo. Por su parte *Feature Envy* es un *Design Smell* inter clase, para detectarlo es relevante observar la interacción de dicha clase con otras clases relacionadas.

En los buenos diseños orientados a objetos la lógica del sistema está distribuida uniformemente entre las clases de los niveles superiores. En [7] se define una *God Class* como un objeto que controla demasiados objetos en el sistema o que ha crecido más allá de toda lógica para convertirse en una clase que lo hace todo. En una *God Class* hay demasiadas variables instancia y demasiado código. Donde hay demasiado código hay peligro de que aparezca duplicidad de código. Este problema de diseño puede ser parcialmente asimilado con el defecto *Large Class* definido por Fowler [6]. También es conocido por antipatrón "*the blob*" en [5].

Feature Envy se recoge en el catálogo Fowler [6] y se clasifica por Wake [8] en la categoría que denomina "entre clases" y la subcategoría "de responsabilidad". Un método tiene el defecto Feature Envy sí parece estar más preocupado en manipular datos de otras clases que de la suya propia. Está relacionado con la responsabilidad de la clase, ya que contiene métodos que deberían estar en otra clase. Como excepciones, Fowler indica que existen varios patrones de diseño que rompen esta regla, como el Visitante y el Estrategia.

A continuación, se comentan dos artículos que desarrollan y prueban, mediante experimentación, las preguntas formuladas en la sección 3.1. El primer artículo [1] expone una comparación de las herramientas de detección de *Design Smells* y su grado de acuerdo en los resultados obtenidos. El segundo artículo [2] expone el grado de acuerdo entre los evaluadores en la detección de *Design Smells.* Tras ver el contenido de estos dos artículos se van a exponer una serie conclusiones que enganchan con la motivación para el desarrollo de la aplicación/herramienta propuesta en este trabajo. Se va a explicar, con los hechos expuestos, como puede

ayudar la aplicación desarrollada en este trabajo (por supuesto, en modesta medida) a intentar estandarizar en un futuro la detección de los *Design Smells*, entre otras cosas.

3.3 "COMPARACIÓN DE HERRAMIENTAS DE DETECCIÓN DE DESIGN SMELLS" [2]

Este artículo plantea la siguiente reflexión: ¿Por qué los métodos de detección de *Design Smell* no han sido adoptados de una manera tan contundente como lo han sido los métodos de refactorización? No se entiende que esto no sea posible si según Fowler los *Design Smells* "*son ciertas estructuras en el código que sugieren (a veces gritan) la posibilidad de refactorización*", es decir, que indican puntos donde se puede aplicar refactorización. Esto lleva a pensar que quizá la falta de adopción sea la falta de acuerdo entre las distintas herramientas existentes. El artículo quiere probar exactamente ese grado de acuerdo.

Para el estudio se eligieron los dos *Design Smells* anteriormente descritos en la sección 3.2, *God Class* y *Feature Envy*. Las herramientas elegidas en este trabajo son elegidas por los criterios de elección de este artículo y los cuales se han explicado ya en la sección 2. Se realizan dos experimentos en el artículo.

El primer experimento se realiza sobre el proyecto Apache Lucene 3.1.0 con las herramientas de detección inCode, inFusion, iPlasma, jDeodorant y Together. En este primer experimento se comprueba que solo hay un buen acuerdo entre las tres primeras. Este acuerdo simplemente es debido a que las tres están creadas prácticamente por el mismo equipo de desarrollo. En el resto de herramientas hay un acuerdo débil.

El segundo experimento se realiza sobre 24 proyectos de código abierto alojados en SourceForge. Las herramientas elegidas para este experimento son iPlasma, como representante del grupo detectado en el experimento anterior, jDeodorant, Together, además de añadir las ya conocidas DECOR y PMD. Tras el experimento se obtiene un grado de acuerdo pobre entre las cinco herramientas. La conclusión del artículo es la siguiente:

"A la vista de los resultados observados, detectamos que sería necesario, por una parte, disponer de herramientas capaces de detectar un amplio espectro de Design Smell. Por otra parte, sería necesario que dichas herramientas pudieran compararse fácilmente, de acuerdo a "patrones estándar" y se pudiera integrar en el proceso software. Todo ello facilitaría que la industria adoptara algunas de esas herramientas, en forma similar al conjunto de herramientas de refactoring. Se debe trabajar en la integración de estas herramientas con las tendencias actuales de automatización de proyectos, permitiendo obtener informes de detección de forma automática sin intervención humana" [2].

3.4 "SOBRE EL GRADO DE ACUERDO ENTRE EVALUADORES EN LA DETECCIÓN DE DESIGN SMELLS" [1]

Este artículo parte del estudio del artículo anterior. Si los estudios comparativos entre herramientas de detección de *Design Smells* concluyen que existe un grado de acuerdo pobre entre ellas ¿Pasará lo mismo en la detección humana? ¿Puede haber acuerdo entre las detecciones humanas y las detecciones de las herramientas? Parece que hay un grado de subjetividad en cuanto a la detección de *Design Smells*. El estudio del artículo está motivado por la subjetividad en la detección de *Design Smells* por evaluadores humanos. Para este estudio también se eligió los *Design Smells God Class* y *Feature Envy* para la detección por parte de evaluadores humanos.

En el experimento de este artículo se extraen del proyecto Apache Lucene 3.1.0 únicamente cinco clases y se les facilitan a los encuestados. Los resultados de este experimento son los siguientes.

El grado de acuerdo en la detección de *God Class* es pobre mientras que con *Feature Envy* directamente no existe. Los programadores inexpertos tienen un mayor grado de acuerdo en la detección de *Feature Envy*, mientras que los más experimentados tienen más desarrollados los aspectos relativos al manejo del tamaño y la complejidad (relacionado con *God Class*). Frente al acuerdo entre evaluadores y herramientas (las elegidas en el artículo anterior) también se concluye que no existe ningún grado de acuerdo, ninguna de las herramientas se acerca más a la detección realizada por los evaluadores. La conclusión del artículo es la siguiente:

"Se detecta que existe un perfil de evaluador en el que se dan los mejores casos de acuerdo entre sí y las herramientas. Este perfil nos indica que los desarrolladores experimentados coinciden mejor en cuestiones relativas al tamaño y complejidad, que los menos experimentados tienen más reciente el conocimiento sobre los principios y los patrones de diseño Orientado a Objetos y que se necesita una mayor formación en Design Smells.(...)Se debe incorporar formación sobre Design Smells en los estudios que preparan a los futuros desarrolladores de software e investigadores en Ingeniería del Software, se debe contar con benchmarks consensuados, con el fin de asegurar que las herramientas cumplen unos mínimos que aseguren su utilidad para detectar Design Dmells, se necesita contar con la opinión de profesionales expertos en aquellas actividades relacionadas con la detección de Design Smells (...)" [1].

3.5 CONCLUSIONES

Los dos artículos describen la situación actual en que se encuentra la detección de *Design Smells*, un alto grado de desacuerdo como en un principio se esperaba. Un alto grado de desacuerdo entre las distintas herramientas que quizá sea consecuencia del alto grado de desacuerdo en la detección de *Design Smells* por parte de evaluadores humanos. También se ha demostrado que la detección de Design Smells está cargado de un cierto nivel de subjetividad. Por ejemplo, esto se publica en el segundo artículo, no tienen la misma concepción de *God Class* los profesionales expertos que los inexpertos, siendo estos últimos más optimistas con el tamaño de las clases que pueden realmente manejar.

Estos hechos son la base sobre la que se asienta la motivación para la realización de la aplicación de este Trabajo Fin de Grado. La aplicación que se pretende desarrollar es una herramienta de ayuda a tareas realizadas en investigaciones y experimentos como los de estos dos artículos. En parte unifica las dos investigaciones de los artículos en uno solo. En relación con el primer artículo, se integran en la aplicación las herramientas de detección de Design Smells. En relación con el segundo artículo, se aporta un método de retroalimentación para los resultados de las herramientas de detección de Design Smells, con esto se consigue saber si es un falso positivo o es un positivo, pero se ha decidido realizarlo así.

Gracias a su uso por parte de usuarios que pueblan la base de datos, futuras investigaciones podrán realizar tareas de minería de datos sobre la misma para aprender un poco más del comportamiento y opiniones tanto de las herramientas como de los usuarios.

4 DESARROLLO DEL PROYECTO

4.1 INTRODUCCIÓN

Como ya se comentó en la sección 3.5, se propone la creación de una herramienta que incluya un conjunto de herramientas detectoras de *Design Smells* que ayude, por un lado, a los desarrolladores con la detección de defectos de diseño en sus proyectos, pero que por otro lado, ayude a investigadores a clarificar el mundo de los *Design Smells* y contribuir así al uso de herramientas de detección y refactoring de estos *Design Smells*.

El objetivo de este proyecto es la creación de una aplicación web en la que lanzar análisis de proyectos alojados en repositorios de código abierto, generar un informe de análisis y facilitar un informe de retroalimentación en la que los usuarios puedan expresar el grado de acuerdo con las detecciones.

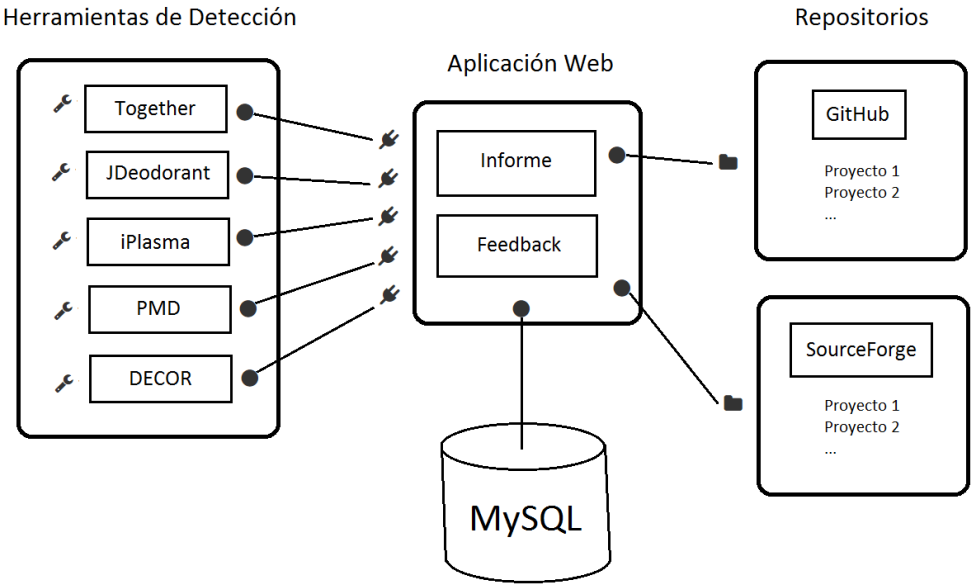


Figura 2. Estructura de la aplicación

La metodología que se ha elegido para el desarrollo de esta práctica es una metodología ágil, basada en iteraciones. Al inicio y al final de cada iteración se realizaba una reunión en la que se validaban los progresos realizados y se fijaban los objetivos para la siguiente iteración.

4.2 PLANIFICACIÓN INICIAL

Con la conclusión del primer cuatrimestre a mediados de febrero y sabiendo que había aprobado todas las asignaturas del primer cuatrimestre, se propuso a mi actual tutora, Yania, la realización del proyecto fin de grado con ella. Entre que la búsqueda de un tema para el trabajo fin de grado y un poco la ambientación por parte de Yania en el tema, y además, la búsqueda de las prácticas externas se pudo comenzar a trabajar en el proyecto en marzo. Así que se ha fijado como inicio del proyecto el día 1 de marzo de 2016.

Como la forma de trabajo fue orientada a un desarrollo ágil, las fases de trabajo se han dividido en sprints o iteraciones de una duración aproximada entre 3 y 4 semanas. Un poco más largas que las iteraciones estándar de los proyectos ágiles, debido a la alta carga de horas por parte del resto de las asignaturas, sobre todo por parte de las prácticas externas, cuyo horario era de 15:30 a 20:30 un total de 5 horas.

4.2.1 Tareas a realizar para el desarrollo del proyecto

- **Elección de un framework:** Hay que realizar una breve evaluación de los posibles frameworks en los que se puede desarrollar la aplicación.
- **Elección de las herramientas de detección:** Hay que evaluar las cinco posibles herramientas dadas por el usuario para establecer una preferencia a la hora de ir añadiéndolas en la aplicación.
- **Elección del o los repositorios:** Hay que realizar una evaluación de los posibles repositorios públicos para ver cuál o cuáles se ajusta mejor al desarrollo de la aplicación.
- **Recopilación de información y aprendizaje de uso de las anteriores elecciones:** Tras realizar las tres tareas anteriores hay que hacer un estudio de su funcionamiento y adaptación a las necesidades del desarrollo del proyecto.
- **Desarrollo iterativo incremental:** Una vez se tiene claro lo que se va a hacer, hay que iniciar un proceso de desarrollo iterativo incremental. Cada iteración empieza y acaba con una reunión con el cliente que propone lo que quiere para esa iteración y valida el trabajo realizado en la iteración anterior.
- **Documentación del trabajo realizado:** Esta tarea se realiza durante todo el desarrollo del proyecto.

4.2.2 Diagrama de Gantt y horas semanales

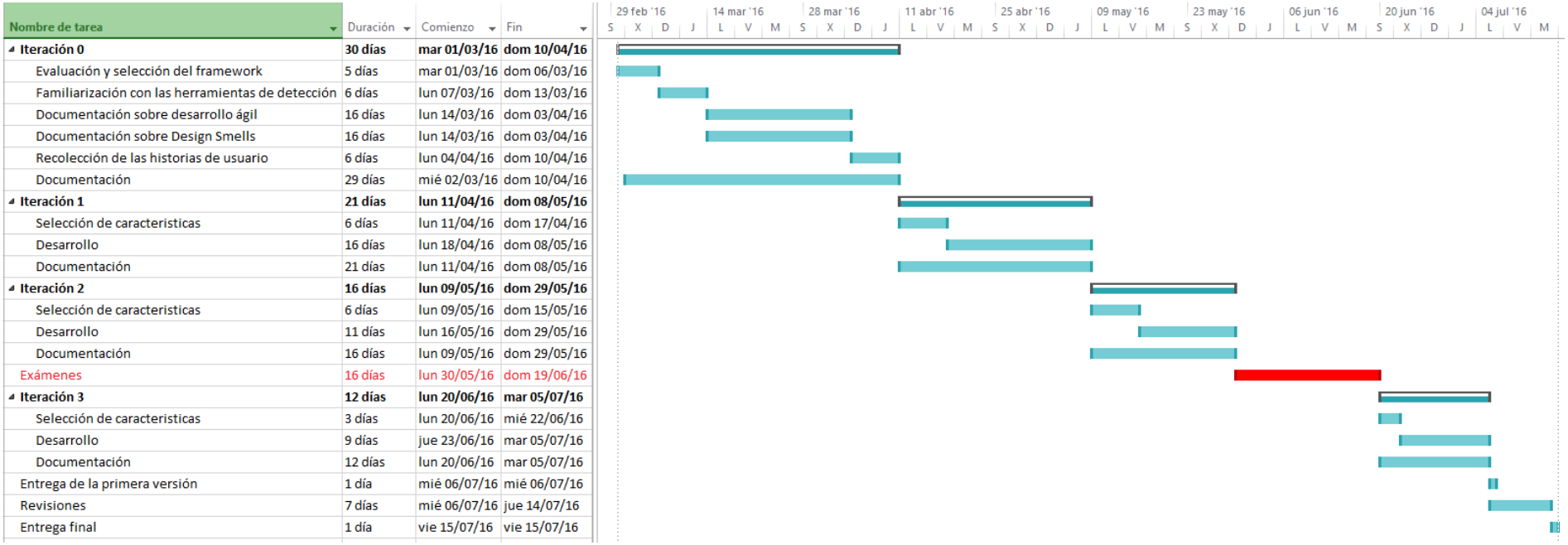
La estimación y distribución de horas ha sido la siguiente:

Del 1 de marzo al 30 de mayo, fecha de finalización de las prácticas externas, sólo podía trabajar en el proyecto por la mañana en las horas libres que no tenía clase o por las noches después de las prácticas externas. La estimación de horas semanales es de 10h/semana (a razón de 2h de lunes a viernes sin un horario fijo) aunque en parte variable en función de la carga de trabajo de las clases. Las tutorías se fijarán una o dos por iteración.

Entre el 30 de mayo hasta el 17 de junio no podré avanzar nada en el proyecto ya que se dedicará completa y exclusivamente a estudiar las asignaturas que tengo este segundo cuatrimestre, ya que son una prioridad y necesario aprobarlas para poder presentar este proyecto.

Del 20 de junio hasta el 6 de julio habrá una plena dedicación al desarrollo y conclusión de una primera versión de la memoria y en parte de la aplicación. La estimación de horas es de 40h/semanales (a razón de 8h de lunes a viernes sin un horario fijo) con la posibilidad de hacer más horas los fines de semana o alguna más entre semana en función del progreso del proyecto.

Del 6 de julio al 15 de julio se trabajará también plenamente con la misma estimación de horas que el tramo temporal anterior, se trabajará en la revisión de la documentación y últimos cambios en la aplicación de cara a una versión final para la presentación definitiva del proyecto.



Nombre de tarea	Duración	Comienzo	Fin
Iteración 0	**30 días**	**mar 01/03/16**	**dom 10/04/16**
Evaluación y selección del framework	5 días	mar 01/03/16	dom 06/03/16
Familiarización con las herramientas de detección	6 días	lun 07/03/16	dom 13/03/16
Documentación sobre desarrollo ágil	16 días	lun 14/03/16	dom 03/04/16
Documentación sobre Design Smells	16 días	lun 14/03/16	dom 03/04/16
Recolección de las historias de usuario	6 días	lun 04/04/16	dom 10/04/16
Documentación	29 días	mié 02/03/16	dom 10/04/16
Iteración 1	**21 días**	**lun 11/04/16**	**dom 08/05/16**
Selección de caracteristicas	6 días	lun 11/04/16	dom 17/04/16
Desarrollo	16 días	lun 18/04/16	dom 08/05/16
Documentación	21 días	lun 11/04/16	dom 08/05/16
Iteración 2	**16 días**	**lun 09/05/16**	**dom 29/05/16**
Selección de caracteristicas	6 días	lun 09/05/16	dom 15/05/16
Desarrollo	11 días	lun 16/05/16	dom 29/05/16
Documentación	16 días	lun 09/05/16	dom 29/05/16
Exámenes	16 días	lun 30/05/16	dom 19/06/16
Iteración 3	**12 días**	**lun 20/06/16**	**mar 05/07/16**
Selección de caracteristicas	3 días	lun 20/06/16	mié 22/06/16
Desarrollo	9 días	jue 23/06/16	mar 05/07/16
Documentación	12 días	lun 20/06/16	mar 05/07/16
Entrega de la primera versión	1 día	mié 06/07/16	mié 06/07/16
Revisiones	7 días	mié 06/07/16	jue 14/07/16
Entrega final	1 día	vie 15/07/16	vie 15/07/16

Figura 3. Diagrama de Gantt para la planificación inicial

4.2.3 Análisis de riesgos

A continuación, se pasan a detallar los riesgos previstos, explicando cada uno en qué consisten, definiendo una probabilidad de ocurrencia, y el tiempo estimado de retraso que se producirá en caso de presentarse dicho riesgo. Más adelante se ordenará la lista de riesgos en base a su magnitud, que es una cantidad calculada a partir del peso del riesgo y la probabilidad de ocurrencia. Esto nos permitirá evaluar cuáles son los riesgos más peligrosos y el plan de contingencia a seguir para tratar de amortiguar el efecto del riesgo sobre la planificación.

ID	Riesgo	Descripción	Probabilidad	Peso caso promedio (semanas)
RSK-01	Planificación optimista	No se han considerado márgenes de error o de indisponibilidad adecuados	0.4	4
RSK-02	Estimación de horas semanales no satisfecha	No se puede dedicar el tiempo semanal estimado.	0.5	2.5
RSK-03	Suspender exámenes	Se ha suspendido alguno de los exámenes por lo que hay que recuperarlo en periodo extraordinario	0.5	4
RSK-04	Falta de tiempo para tutorías	Motivos laborales o personales del tutor o del proyectado, juntas o reuniones en horas reservadas a tutorías, entregas de prácticas o exámenes.	0.25	2
RSK-05	Reducción de la productividad por inexperiencia	El desarrollo es más lento debido a la falta de experiencia con el lenguaje/bibliotecas/herramientas.	0.33	3
RSK-06	No aprobar todas las asignaturas	Se ha suspendido alguna asignatura en la convocatoria extraordinaria y hay que volver a matricularse.	0.20	26

Tabla 6. Listado detallado de riesgos

ID	Riesgo	Magnitud	Plan de contingencia
RSK-06	No aprobar todas las asignaturas	5.2	Retrasar el fin de proyecto hasta la convocatoria de enero.
RSK-03	Suspender exámenes	2	Aumentar el número de horas semanales, en caso de no ser posible plantear el retraso de la entrega hasta la siguiente convocatoria.
RSK-01	Planificación optimista	1.6	Aumentar el número de horas semanales. Ajustar la funcionalidad implementada en cada iteración.
RSK-02	Estimación de horas semanales no satisfechas	1.25	Recuperar fines de semana el tiempo no dedicado a lo largo de la semana.
RSK-05	Reducción de la productividad por inexperiencia	0.99	Aumentar el número de horas semanales. Ajustar la funcionalidad implementada en cada iteración.
RSK-04	Falta de tiempo para tutorías	0.5	Aumentar la frecuencia de las tutorías. Utilizar medios virtuales y documentos compartidos para las tutorías.

Tabla 7. Priorización de los riesgos

En ninguno de los casos es posible retrasar la entrega del proyecto, ya contemplamos que la fecha de entrega es lo más tarde que podemos, dentro de las fechas otorgadas por la escuela, el día 15 de julio de 2016. Así que los únicos planes de contingencia posibles son los de echar más horas de trabajo o sacrificar cierta funcionalidad.

4.3 TECNOLOGÍA EMPLEADA Y ARQUITECTURA DE LA APLICACIÓN

A continuación, se va explicar la tecnología empleada para el desarrollo de la aplicación y cómo se organiza el proyecto. El framework que se elige debe ser PHP ya que es un requisito del cliente, al igual que la base de datos debe ser MySQL.

4.3.1 Framework

Para la realización de la aplicación se valoraron distintos frameworks PHP: Yii Framework, Symfony y DooPHP. Como el aprendizaje de un nuevo framework retrasaría mucho el desarrollo del proyecto, se ha optado por DooPHP que, aunque sea menos conocido, ya existe un conocimiento y un manejo de él por mi parte.

Finalmente se decidió por DooPHP al ser un framework con el que ya se había trabajado anteriormen-te. Esta es la principal razón de su elección. Era muy arriesgado aprender un nuevo framework y tener la posibilidad de perder mucho tiempo en su aprendizaje.

DooPHP es un framework con una gran simplicidad a la hora de instalar y configurar. En él, la aplicación del modelo-vista-controlador resulta bastante sencilla. Una de sus principales características es su rapidez. Este framework además permite un sistema de enrutamiento para la web, tiene un acceso y escritura a la base de datos simplificado, mecanismos para soporte de caché, seguridad y acceso entre otras características de menor relevancia.

Además de DooPHP, para la parte de la vista emplearemos un gestor de plantillas llamado Twig. Uno de los contras que le podemos encontrar a DooPHP es que la última versión estable que sacaron fue la 1.5 el 5 de octubre de 2013. El dominio del proyecto expiró el día 7 de julio, durante la realización del trabajo, y no tiene pinta de renovación en un futuro.

4.3.2 Arquitectura de la aplicación

La arquitectura de la aplicación viene dada por el propio framework DooPHP. La arquitectura que a priori obliga a utilizar el framework es la siguiente:

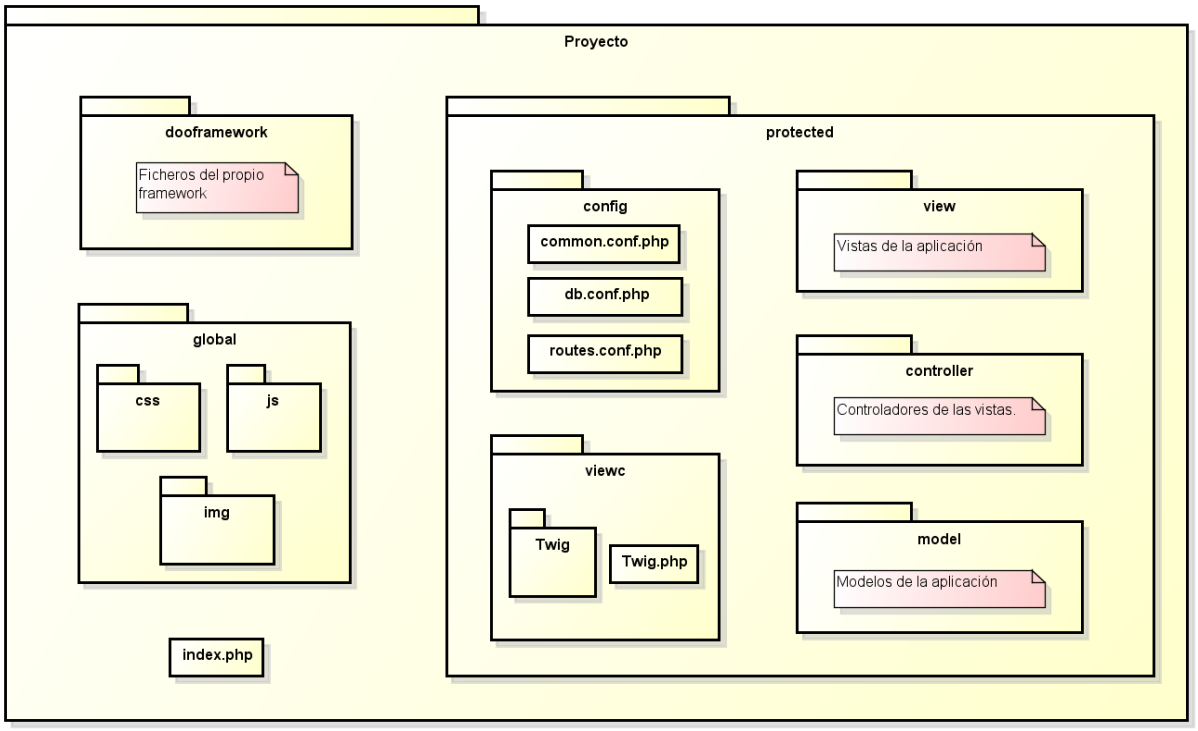


Figura 4. Arquitectura dada por el framework

El propio framework se encuentra en el directorio dooframework. En global se encuentran todos los ficheros javascript, css, imágenes y demás recursos de las vistas. Dentro de protected, en el directorio config se encuentran los ficheros de configuración del framework, base de datos y enrutamiento, dentro de viewc el gestor de plantillas Twig y finalmente las carpetas de los controladores, las vistas y los modelos.

Al final de todas las iteraciones se mostrará la arquitectura final de la aplicación. Ya con las vistas, los controladores y los modelos.

4.3.3 Patrón MVC

MVC o Modelo-Vista-Controlador (Model-View-Controller) es un patrón arquitectónico de Ingeniería del Software para sistemas que implementan interfaces de usuario. Divide el sistema en tres partes y separa la representación de la información de la lógica, de la información presentada y recibida por el usuario en su interfaz.

- Modelo: Esta capa es la que representa la información y los datos de la base de datos. La mayoría de veces nos encontramos una tabla y su correspondiente modelo en la aplicación. Depende del patrón de acceso a datos empleado, se implementa de una manera u otra. DooPHP emplea ORM (Object.Relational mapping) al que le dedicamos un apartado más adelante.

- Vista: La vista es la capa de presentación de nuestro sistema, es la responsable de generar y mostrar información del modelo. En DooPHP se ha integrado Twig, un gestor de plantillas, al que también le dedicaremos una sección más adelante.

- Controlador: El controlador es la capa encargada de conectar el modelo con la vista, actúa de intermediario y procesa los datos que vienen del modelo para pasárselos a la vista. No debe tener lógica de acceso a la base de datos, estas operaciones deberían ser realizadas por el modelo.

En la Figura 4 se puede ver que la arquitectura del framework realiza esta división.

4.3.4 ORM

ORM son las siglas de mapeo objeto-relacional (en inglés Object-Relational mapping). Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo). En DooPHP para traer un objeto de la base de datos:

```
$db_object = Doo::db()->find(<objeto>);
```

Ese <objeto> es un objeto que tiene alguno de los atributos que se quiere traer de la base de datos. Normalmente suele traer de la base de datos un array de objetos que cumplen esas características. Para traer únicamente uno, hay que introducir en <objeto> que se pasa a *find* el indentificador que representa a ese objeto en la base de datos. También se puede traer un único objeto de la siguiente manera:

```
$db_object = Doo::db()->find(<objeto>, array('limit'=> 1);
```

Trae el primer objeto que cumple esas características. Esta forma de acceso a la base de datos es la que principalmente se usa, pero también tiene otras formas de buscar en la base de datos. Se puede incluso ejecutar sentencias directamente:

```
$result = Doo::db()->query('<sentencia SQL');
```

4.3.5 Twig

Twig es un motor de plantillas para el lenguaje de programación PHP. La sintaxis se origina a partir de las plantillas de Jinja y Django. Es un producto de código abierto bajo una Licencia BSD y mantenido por Fabien Potencier. La versión inicial fue creada por Armin Ronacher. El framework Symfony2 viene con un soporte integrado para Twig como motor de plantillas por defecto. Un ejemplo de la sintaxis de Twig:

```
{% extends "base.html" %}
{% block navigation %}
    <ul id="navigation">
    {% for item in navigation %}
        <li>
            <a href="{{ item.href }}">
                {% if item.level == 2 %}  {% endif %}
                {{ item.caption|upper }}
            </a>
        </li>
    {% endfor %}
    </ul>
{% endblock navigation %}
```

Twig define tres tipos de delimitadores:

- **{% ... %}**, usado para ejecutar sentencias tales como bucles for.
- **{{ ... }}**, usado para imprimir el contenido de variables o resultados de la evaluación de una expresión.
- **{# ... #}**, usado para los comentarios en las plantillas. Estos comentarios no se incluyen en la renderización de la página.

4.4 METODOLOGÍA DE DESARROLLO

En esta sección se va a explicar la metodología empleada para el desarrollo del proyecto software. Los métodos y procesos de desarrollo utilizados entran dentro de lo que se conoce como métodos ágiles.

4.4.1 Introducción

Los métodos ágiles refieren a métodos iterativos e incrementales de ingeniería del software. Estos métodos hacen especial énfasis en la comunicación directa y continua con el cliente y hacer primar estas documentaciones sobre la documentación escrita. Las ideas en las que se basan estos métodos fueron resumidas en el Manifiesto Ágil [3]. En general estos métodos promueven:

- Planificación con capacidad de adaptación.
- Desarrollo evolutivo.
- Comunicación con el cliente por encima de la documentación.
- Entrega temprana de demos funcionales.
- Mejora continua.
- Respuesta flexible a los cambios.

4.4.2 Sprints o iteraciones

Las entregas se realizan después de cada iteración o periodo de desarrollo, este concepto se conoce como Sprints. En este trabajo se ha conservado el término iteración. Estos periodos suelen tener una duración de dos semanas en las que se desarrollan una serie de características (*features*) que se han planificado previamente. La idea es tener al final de cada iteración un prototipo funcional que integre los cambios introducidos por las *features*.

Al tener periodos de desarrollo tan cortos, se consigue minimizar el riesgo de desarrollar algo incorrectamente. Tras cada periodo de desarrollo se debe llevar a cabo una reunión con el cliente para verificar los cambios implementados y establecer nuevos requisitos mediante historias de usuario.

4.4.3 Historias de usuario

Las historias de usuario o *User Stories* son representaciones de requisitos expresados en un lenguaje común entendible por el usuario. Tienen la capacidad de responder rápidamente a requisitos cambiantes y evitar gran cantidad de documentos formales. Es el equivalente en metodología ágil al paso de toma de requisitos a casos de uso. Son un acuerdo que formaliza la funcionalidad descrita y esperada de parte de la aplicación.

La descripción o enunciado de la historia de usuario tiene una forma de ser construida, con la estructura "Como - Quiero - Para" (en inglés "As a - I want - So that"):

- **Como:** Se describe el rol que se toma en esta historia y/o las condiciones que se dan para ese rol.
- **Quiero:** Se describe expresamente qué se quiere que ocurra en la historia.
- **Para:** Aquí se pone el objetivo que se busca cumplir con la historia.

Las historias de usuario también deben tener una serie de características:

- **Independientes:** cada una tiene un resultado independiente del resto.
- **Negociables:** La discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.
- **Valoradas por los clientes:** Cada historia debe ser importante para los usuarios más que para el desarrollador
- **Estimables:** Un resultado de la discusión de una historia de usuario es la estimación de tiempo en completarla.
- **Short:** Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.
- **Testeable:** Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables.

4.4.4 Ventajas y desventajas

Ventajas	Desventajas
• Buena respuesta a los cambios en los requisitos. • Al tener trato continuo con el cliente no hay suposiciones de requisitos • El equipo no tiene que preocuparse de que en la entrega del producto los requisitos hayan cambiado	• Es complicado calcular el esfuerzo total requerido al principio del proyecto. • Depende de que el cliente exprese los requisitos correctamente. • Falta de énfasis en diseños y documentación, a veces necesarios.

Tabla 8. Ventajas y desventajas del desarrollo ágil

4.4.5 ¿Por qué usar desarrollo ágil?

Se ha elegido esta metodología porque no se sabía en qué grado se iba a poder abarcar el total de lo que el cliente quiere. No se podía saber si se iba a poder integrar el conjunto de cinco herramientas, anteriormente descritas, o el conjunto de repositorios públicos mencionados por el cliente. Así que, mediante el método iterativo que proporciona la metodología ágil, se podrá ir añadiendo funcionalidad progresivamente. Esta forma iterativa acompaña también a la gestión de riesgos, ya que al estar dividido en iteraciones los riesgos disminuyen en cada parte.

Para el desarrollo de forma ágil se ha tenido como referencia el libro Engineering Software as a Service: An Agile Approach Using Cloud Computing [9] y un Trabajo Fin de Grado [10] inspirado en este mismo libro.

4.5 DESCRIPCIÓN DE ITERACIONES

4.5.1 Introducción

En esta sección 4.5 se registra toda la documentación referente a cada iteración. Se reúne toda la documentación escrita durante todo el proceso iterativo, desde el inicio con el planteamiento inicial del cliente, hasta el final con el estado final de la aplicación en la finalización de la última iteración.

4.5.2 Iteración 0: Puesta en marcha y planteamiento inicial

4.5.2.1 Introducción

La iteración 0 es la iteración inicial, aquí se conocen las características que el cliente quiere, en un principio, que su aplicación tenga. En este caso el cliente es la tutora del proyecto. Además, en esta iteración se eligen las tecnologías que se van a emplear y que mejor se adapten a las necesidades del cliente. También se va a especificar el entorno de trabajo. En esta primera iteración se incluyen las primeras historias de usuario.

Esta primera iteración es una toma de contacto con el cliente, la idea de su proyecto y la inicialización y prueba de las tecnologías y herramientas que se van a emplear. El objetivo de esta primera iteración es la familiarización y la documentación del tema central del proyecto, que son los defectos de diseño software (*Design Smells*) y las herramientas que facilitan su detección.

4.5.2.2 Planteamiento inicial por parte del usuario

En esta sección se describe la idea inicial transmitida por el cliente en la primera reunión.

El objetivo es desarrollar una aplicación web que apoye en la investigación acerca de la detección de *Design Smells*. Esta aplicación se centra en los proyectos de código abierto alojados en repositorios públicos, por tanto, deberá poder descargar los proyectos de dichos repositorios guardando su estado de versión. La aplicación tendrá que integrar una serie de herramientas de detección de *Design Smells* y guardar los reportes en una base de datos MySQL. Una vez detectados los *Design Smells* es importante dar un soporte de retroalimentación a los usuarios, para que validen los datos presentados por las distintas herramientas.

4.5.2.3 Requisitos no funcionales, herramientas y tecnología

A continuación, se describen el conjunto de requisitos no funcionales. Condicionado por estos requisitos no funcionales se elige la tecnología para el desarrollo de la aplicación.

Uno de los primeros requisitos y más importantes es la base de datos. Primero debe ser obligatoriamente MySQL, y segundo, debe de tener un buen diseño ya que no solo se piensa utilizar para almacenar datos, sino que en un futuro se quiere utilizar para realizar tareas de minería de datos sobre ella.

Un segundo requisito es el lenguaje de programación empleado. El lenguaje que se emplee en el desarrollo del proyecto debe ser seguro y evitar ataques como por ejemplo inyecciones SQL. Se llegó a la conclusión junto con el cliente que el mejor lenguaje para el desarrollo del proyecto es PHP.

Un tercer requisito es que la aplicación que se desarrolla debe incluir una serie de herramientas de análisis y detección *Design Smells*. Estas herramientas son: PMD, iPlasma, JDeodorant, DECOR, Together.

Dados estos requisitos, la decisión fue el uso de un framework PHP. El framework elegido DooPHP, que usa el patrón MVC y el mapeo objeto-relacional para el acceso y escritura a base de datos. Para la implementación de la vista no se ha partido desde cero, sino que se ha elegido una plantilla de estilo de administración gratuita [23].

4.5.2.4 Primeras historias de usuario

A continuación, muestran las primeras historias de usuario extraídas de la primera reunión con el cliente. Estas historias de usuario explican los propósitos del proyecto. Las historias de usuario definidas aquí hacen la función de requisitos funcionales durante el desarrollo del proyecto.

Historia: Añadir un proyecto alojado en un repositorio.
Como usuario
Para poder analizar los proyectos almacenados en los repositorios
Quiero añadir proyectos alojados en los repositorios.

Historia: Análisis de un proyecto añadido.
Como usuario
Para poder ver los defectos de diseño
Quiero analizar los proyectos añadido

Historia: Almacenamiento de los datos obtenidos a partir de las herramientas.
Como usuario
Para poder revisar y realizar una retroalimentación de los errores detectados.
Quiero almacenar los datos obtenidos a partir del análisis de los proyectos con las herramientas.

Historia: Crear grupos de desarrollo.
Como usuario
Para que todos los participantes en el desarrollo del mismo puedan ver los análisis realizados sobre el mismo
Quiero poder crear un grupo de desarrollo alrededor de un proyecto.

Historia: Informar a los grupos de desarrollo de los reportes de un proyecto al que pertenecen.
Como usuario
Para que puedan ver los reportes de los proyectos en los que trabajan
Quiero que los usuarios sean notificados de los análisis que otros miembros del equipo de desarrollo al que pertenecen.

Historia: Mostrar los informes de los reportes de los análisis de los proyectos.

Como usuario
Para poder revisar los errores y realizar retroalimentaciones a la plataforma sobre los mismos.
Quiero que la plataforma muestre informes de los reportes obtenidos por las distintas herramientas.

Historia: Permitir la retroalimentación de los participantes del proyecto con los análisis y reportes de la herramienta fabricada.

Como usuario
Para poder extraer información, estadísticas, mejorar la plataforma y en un futuro poder realizar tareas sobre minería de datos.
Quiero que se pueda realizar una realimentación sobre los informes y los reportes otorgados por la plataforma.

```
Historia: Compartir informes de análisis.
      Como usuario
      Para que otros usuarios ya sean de la aplicación o no puedan ver mis análisis
      Quiero poder compartir los informes de análisis
```

4.5.3 Iteración 1: Primer prototipo

4.5.3.1 Introducción

Una vez concluida la parte de elección del framework, documentación, familiarización con las herramientas de detección y aclarado el propósito de la aplicación, en esta iteración se empieza a desarrollar la aplicación.

En esta iteración se va a construir un primer prototipo, que lo único que va a hacer, a parte del registro e inicio de sesión, es uno de los flujos principales de la plataforma. Este flujo consiste en añadir un proyecto a la plataforma y analizar su estado actual.

Construir esta funcionalidad con todas las herramientas de análisis y todos los repositorios de proyectos demandados por el usuario sería un trabajo demasiado grande para una sola iteración. Por esto, para esta primera iteración se ha elegido solamente el repositorio de proyectos GitHub y la herramienta de análisis PMD. Se ha elegido GitHub por una mayor proximidad y familiarización con su uso. Se ha elegido PMD por ser la herramienta más sencilla que se ha encontrado en el proceso de familiarización en la iteración anterior.

A continuación, se detallan un poco más las características que se van a implementar.

4.5.3.2 Características a implementar

Las siguientes características están descritas con la estructura que el libro Engineering Software as a Service: An Agile Approach Using Cloud Computing [9] hace para que funcionen en Cucumber. En este caso solo es la estructura de tal modo que se asemeje o aproxime un poco a lo que es la descripción de casos de uso en el desarrollo en el proceso unificado. Estas características que se muestran a continuación pretenden ser más detalladas que las historias de usuario descritas en la iteración inicial.

```
Característica: Registro del usuario introduciendo nombre de usuario, contraseña
y datos personales relevantes para la aplicación.

      Como usuario no registrado
      Para poder usar la aplicación
      Quiero poder registrarme.

Escenario: Registro como usuario de la página.
      Estoy en la página principal de la aplicación
      Pulso en el botón de registro
      Y aparece un formulario de registro
      Después introduzco los todos los datos de registro (nombre de usuario,
      contraseña…)
      Finalmente, el registro concluye.
Escenario: Registro fallido con un nombre de usuario en uso.
      Estoy en la página principal de la aplicación
      Pulso en el botón de registro
      Y aparece un formulario de registro
      Después introduzco los todos los datos de registro (nombre de usuario,
      contraseña…)
      Finalmente, vuelve al formulario de registro y se muestra un error indicando
      que el nombre de usuario introducido ya está en uso.
```

```
Escenario: Registro fallido al repetir las contraseñas una no corresponde con la
otra.
      Estoy en la página principal de la aplicación
      Pulso en el botón de registro
      Y aparece un formulario de registro
      Después introduzco los todos los datos de registro (nombre de usuario,
      contraseña…)
      Finalmente, vuelve al formulario de registro y se muestra un error indicando
      que las contraseñas introducidas no eran iguales.

Escenario: Registro fallido al repetir los correos electrónicos uno no corresponde
con el otro.
      Estoy en la página principal de la aplicación
      Pulso en el botón de registro
      Y aparece un formulario de registro
      Después introduzco los todos los datos de registro (nombre de usuario,
      contraseña…)
      Finalmente, vuelve al formulario de registro y se muestra un error indicando
      que los correos electrónicos no eran iguales.
```

```
Característica: Inicio de sesión del usuario con nombre de usuario y contraseña.

      Como usuario registrado
      Para poder usar la aplicación
      Quiero poder iniciar sesión.

Escenario: Inicio de sesión.
      Estoy en la página principal de la aplicación
      Pulso en el botón de iniciar sesión
      Introduzco nombre de usuario y contraseña
      Finalmente, veo el panel de control con mis proyectos.

Escenario: Inicio de sesión fallido, nombre de usuario incorrecto.
      Estoy en la página principal de la aplicación
      Pulso en el botón de iniciar sesión
      Introduzco nombre de usuario incorrecto y contraseña
      Finalmente, vuelvo al formulario de iniciar sesión y se muestra un error
      indicando que el nombre de usuario o contraseña no son correctos.

Escenario: Inicio de sesión fallido, contraseña incorrecta.
      Estoy en la página principal de la aplicación
      Pulso en el botón de iniciar sesión
      Introduzco nombre de usuario y contraseña incorrecta.
      Finalmente, vuelvo al formulario de iniciar sesión y se muestra un error
      indicando que el nombre de usuario o contraseña no son correctos.
```

```
Característica: Añadir un proyecto de GitHub a la aplicación para poder realizar
análisis.

      Como usuario
      Para poder analizar los proyectos almacenados en GitHub
      Quiero añadir proyectos alojados en GitHub.

Antecedentes: He iniciado sesión en el sistema.
```

```
Escenario: Añadir proyecto
      Estoy en el panel de control de mis proyectos
      Pulso en añadir nuevo proyecto
      Aparece un formulario en el que añado la dirección web de GitHub y el
      nombre que le quiero dar a mi proyecto dentro de la aplicación
      Finalmente, pulso aceptar y mi proyecto aparece en la lista de proyectos
      del panel de control.

Escenario: Cancelar añadir proyecto
      Estoy en el panel de control de mis proyectos
      Pulso en añadir nuevo proyecto
      Aparece un formulario para añadir la información para añadir el
      repositorio
Finalmente, pulso cancelar, vuelvo al panel de control y el repositorio no se
añade
```

```
Característica: Lanzar un análisis de proyecto con la herramienta PMD.

      Como usuario
      Para ver los defectos de diseño de un proyecto alojado en GitHub
      Quiero lanzar análisis con la herramienta PMD.

Antecedentes: He iniciado sesión en el sistema y he añadido un proyecto de
GitHub.

Escenario: Lanzar análisis.
      Estoy en el panel de control de mis proyectos
      Pulso en el proyecto que quiero analizar
      Se muestran una serie de análisis realizados anteriormente, en el caso que
      se haya realizado alguno
      Pulso en el botón de realizar nuevo análisis
      Aparece un formulario en el que está marcada la opción de PMD
      Finalmente, pulso aceptar para lanzar el análisis.

Escenario: Cancelar lanzar análisis.
      Estoy en el panel de control de mis proyectos
      Pulso en el proyecto que quiero analizar
      Se muestran una serie de análisis realizados anteriormente, en el caso que
      se haya realizado alguno
      Pulso en el botón de realizar nuevo análisis
      Aparece un formulario en el que está marcada la opción de PMD
      Finalmente, pulso el botón cancelar, vuelvo al panel de control y el
      análisis no se lanza.
```

```
Característica: Ver el resultado de un análisis de un proyecto con la
herramienta PMD.

      Como usuario
      Para saber los defectos de diseño de un proyecto alojado en GitHub
      Quiero ver los resultados del análisis de la herramienta PMD.

Antecedentes: He iniciado sesión en el sistema, he añadido un proyecto de GitHub
y he lanzado con anterioridad un análisis y este se ha completado.

Escenario: Ver resultado de un análisis
      Estoy en el panel principal de mis proyectos
```

```
Pulso en el proyecto que quiero ver el análisis planificado con
anterioridad
Aparecen una lista de análisis realizados, que en el caso de que sea el
primero solo aparecerá el que queremos
Pulso en el análisis que quiero ver
Finalmente aparece un resumen de los datos que PMD otorga tras
el   análisis.
```

4.5.3.3 Diagrama de clases y tablas

El modelo de dominio y el modelo de entidad relación (las tablas de la base de datos) son iguales. Esto es debido a que el framework empleado usa un mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational Mapping, ORM) para la relación entre el modelo y la base de datos. ORM es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional.

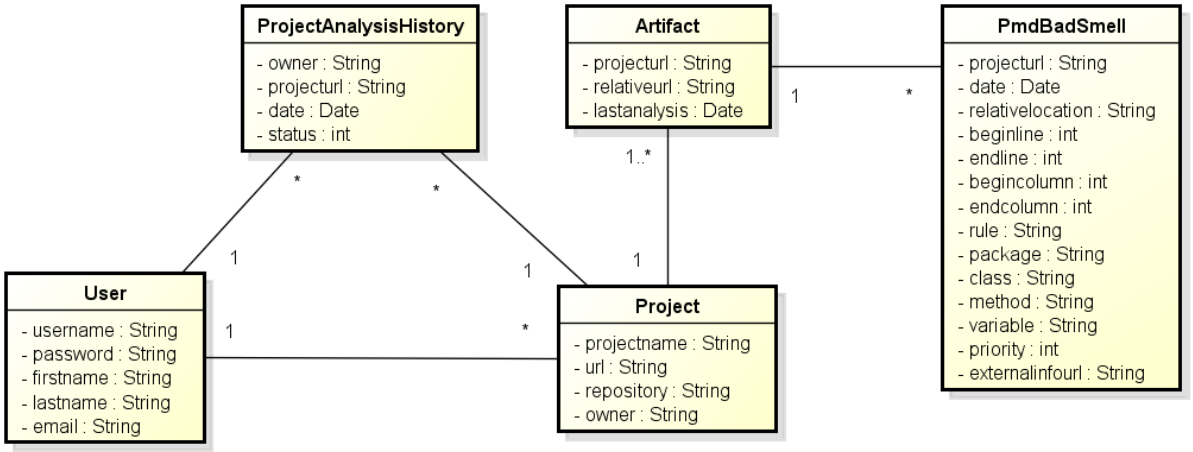


Figura 5. Diagrama de clases y tablas. Iteración 1

4.5.3.4 Capturas de la aplicación

En esta iteración conforme a las características que se especificaron, se desarrollaron las siguientes vistas: Página de registro con un formulario simple.

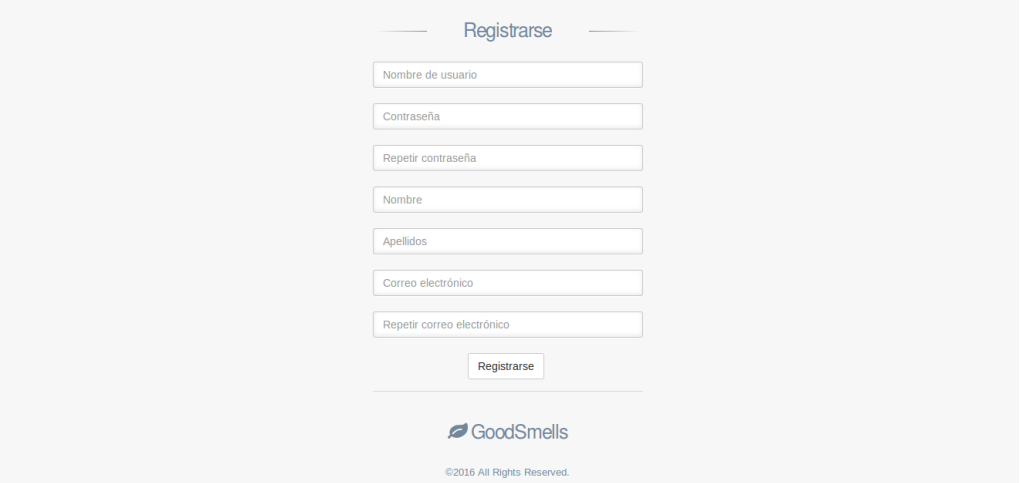


Figura 6. Vista para el registro de usuario. Iteración 1

Página de inicio de sesión con un formulario simple.

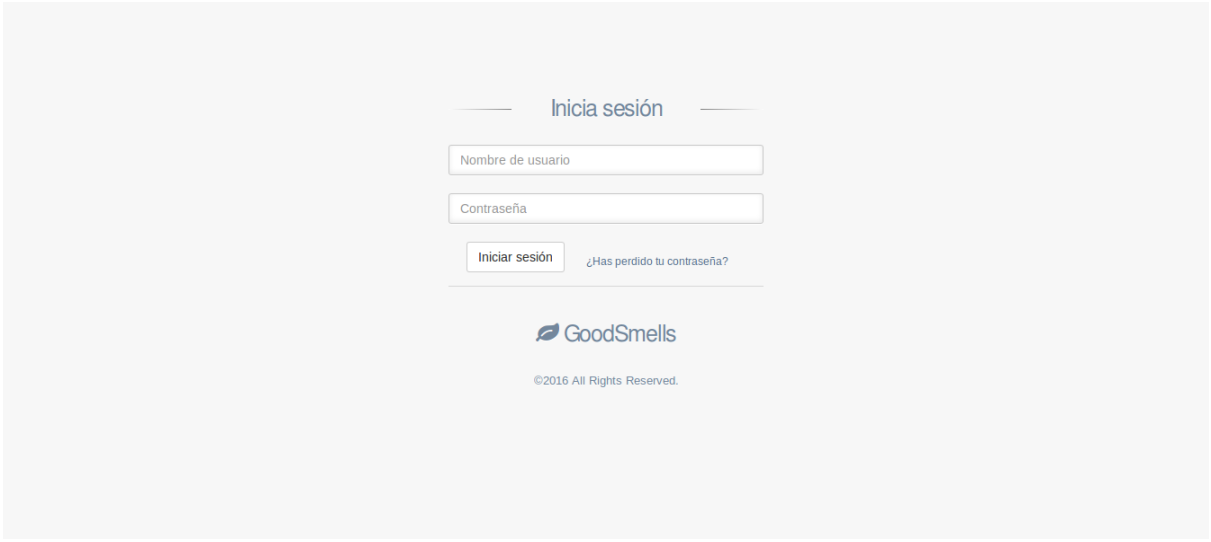


Figura 7. Vista para el inicio de sesión. Iteración 1

Tras el inicio de sesión se muestra una vista con todos los proyectos que el usuario ya tiene añadidos.

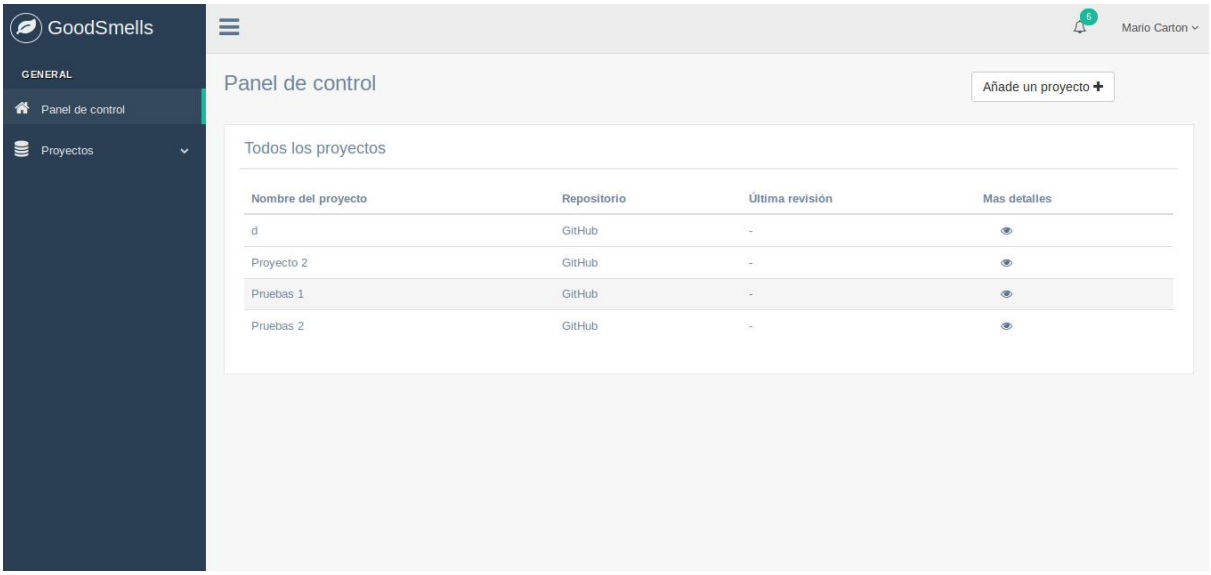


Figura 8. Vista principal del panel de control. Iteración 1

En esa misma vista se encuentra un botón para poder añadir nuevos proyectos a la cuenta del usuario para su posterior análisis. Se muestra un formulario para añadir el proyecto.

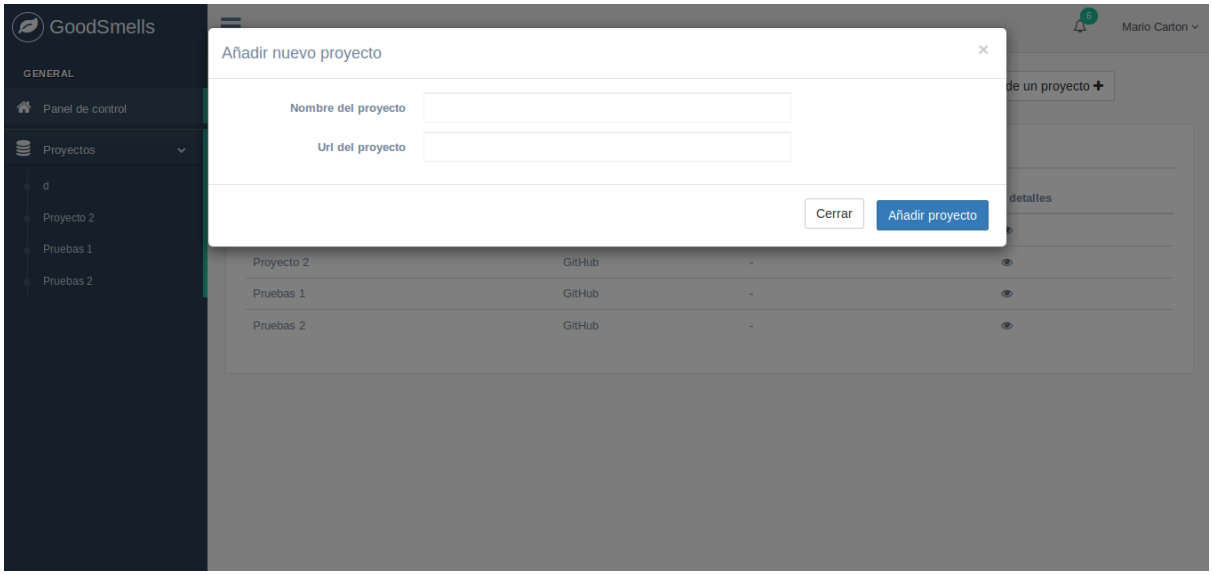


Figura 9. Vista añadir proyecto. Iteración 1

Una vez añadido el proyecto se puede lanzar un análisis. Se muestra un formulario para añadir un análisis.

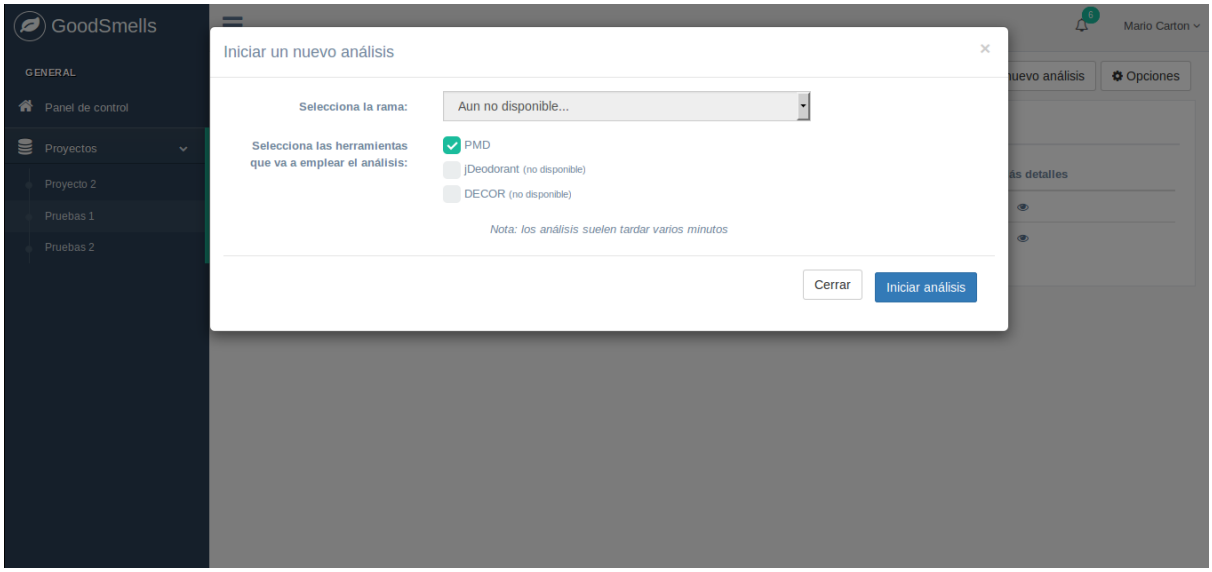


Figura 10. Vista lanzar análisis. Iteración 1

Una vez se han realizado uno o más análisis se pueden ver y la fecha en la siguiente vista. Desde esta vista es desde la que se lanzan también los nuevos análisis, en el botón de iniciar análisis.

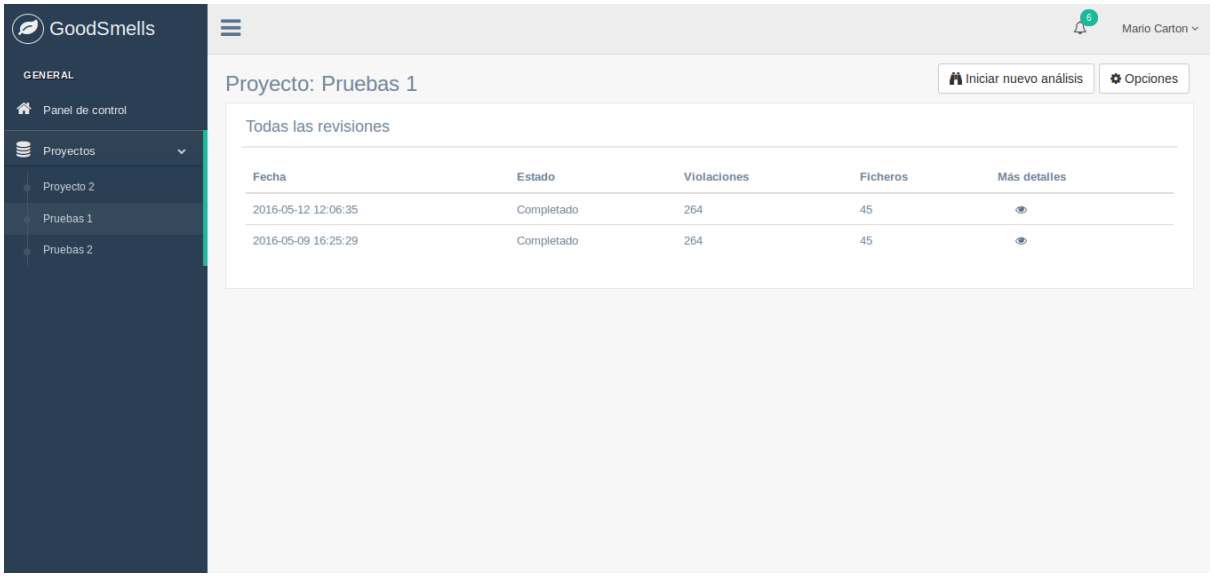


Figura 11. Vista resumen de proyecto. Iteración 1

Otra vista muestra los detalles del análisis.

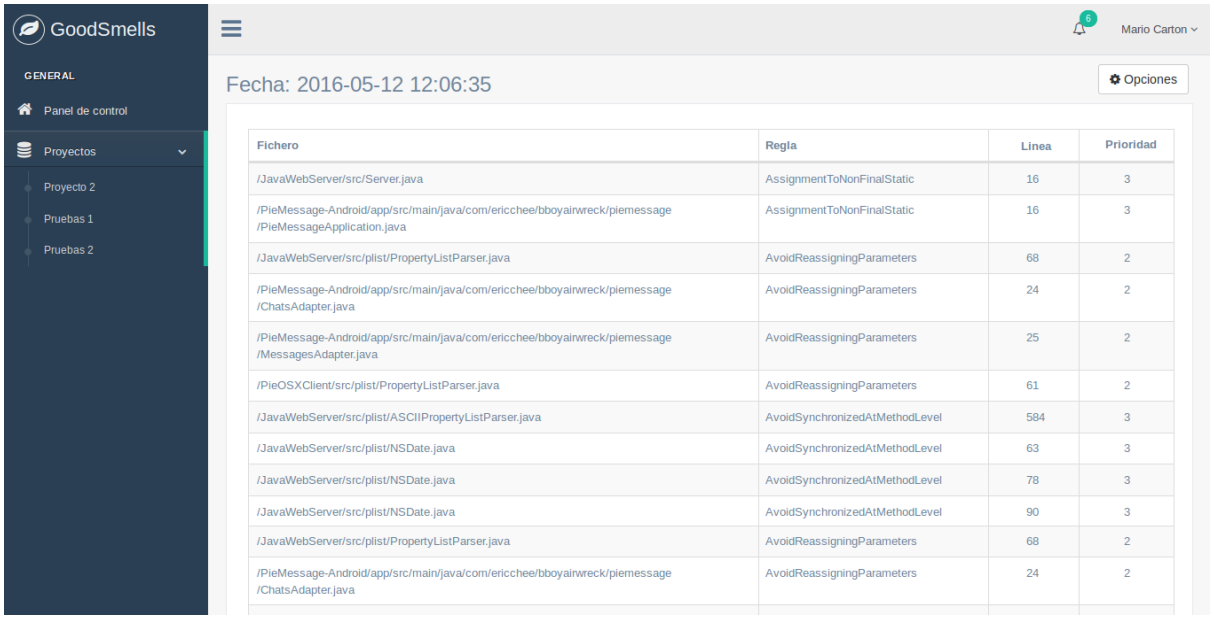


Figura 12. Vista resultado de análisis. Iteración 1

4.5.3.5 Pruebas realizadas durante esta iteración

CP_I1_01	Registro de usuario en la aplicación
Versión	1.0
Descripción	Un usuario se registra en la aplicación
Resultado esperado	El usuario quede registrado
Aplicación	1.0
Resultado	Correcto

Tabla 9. Descripción de CP_I1_01

CP_I1_02	Registro fallido con un nombre de usuario en uso
Versión	1.0
Descripción	El usuario intenta registrarse con un nombre de usuario que ya ha cogido otro usuario
Resultado esperado	El usuario no se registra y se le muestra un mensaje de error.
Aplicación	1.0
Resultado	Primera ejecución: Incorrecto Causa: No se muestra ningún mensaje Acciones realizadas: Corregir el código del controlador y la vista para que muestre el mensaje de aviso Segunda ejecución: Correcta

Tabla 10. Descripción CP_I1_02

CP_I1_03	Registro fallido al repetir las contraseñas una no corresponde con la otra.
Versión	1.0
Descripción	El usuario no introduce correctamente la contraseña y su confirmación
Resultado esperado	El usuario no se registra y se muestra un mensaje de error.
Aplicación	1.0
Resultado	Primera ejecución: Incorrecto Causa: No se muestra ningún mensaje Acciones realizadas: Corregir el código del controlador y la vista para que muestre el mensaje de aviso Segunda ejecución: Correcta

Tabla 11. Descripción CP_I1_03

CP_I1_04	Registro fallido al repetir los correos electrónicos uno no corresponde con el otro.
Versión	1.0
Descripción	El usuario no introduce correctamente el correo electrónico y su confirmación
Resultado esperado	El usuario no se registra y se muestra un mensaje de error
Aplicación	1.0
Resultado	Primera ejecución: Incorrecto Causa: No se muestra ningún mensaje Acciones realizadas: Corregir el código del controlador y la vista para que muestre el mensaje de aviso Segunda ejecución: Correcta

Tabla 12. Descripción CP_I1_04

CP_I1_05	Inicio de sesión
Versión	1.0
Descripción	El usuario registrado inicia sesión en la aplicación
Resultado esperado	El usuario entra en la aplicación y se le muestran los proyectos que tiene guardados (si tiene alguno)
Aplicación	1.0
Resultado	Correcto

Tabla 13. Descripción CP_I1_05

CP_I1_06	Inicio de sesión fallido, nombre de usuario incorrecto.
Versión	1.0
Descripción	El usuario introduce nombre de usuario incorrecto al iniciar sesión en el sistema
Resultado esperado	El usuario no inicia sesión y se le muestra un mensaje de error
Aplicación	1.0
Resultado	Primera ejecución: Incorrecto Causa: No se muestra ningún mensaje Acciones realizadas: Corregir el código del controlador y la vista para que muestre el mensaje de aviso Segunda ejecución: Correcta

Tabla 14. Descripción CP_I1_06

CP_I1_07	Inicio de sesión fallido,contraseña incorrecta.
Versión	1.0
Descripción	El usuario introduce contraseña incorrecta al iniciar sesion en el sistema
Resultado esperado	El usuario no inicia sesion y se le muestra un mensaje de error
Aplicación	1.0
Resultado	Primera ejecución: Incorrecto Causa: No se muestra ningún mensaje Acciones realizadas: Corregir el código del controlador y la vista para que muestre el mensaje de aviso Segunda ejecución: Correcta

Tabla 15. Descripción CP_I1_07

CP_I1_08	Añadir un proyecto de GitHub
Versión	1.0
Descripción	El usuario añade un proyecto GitHub a la aplicación
Resultado esperado	El proyecto queda añadido
Aplicación	1.0
Resultado	Correcto

Tabla 16. Descripción CP_I1_08

CP_I1_09	Cancelar añadir proyecto GitHub
Versión	1.0
Descripción	El usuario cancela añadir proyecto GitHub a la aplicación
Resultado esperado	No se añade el proyecto
Aplicación	1.0
Resultado	Correcto

Tabla 17. Descripción CP_I1_09

CP_I1_10	Lanzar análisis
Versión	1.0
Descripción	El usuario lanza un análisis de un proyecto ya añadido a la aplicación
Resultado esperado	El análisis se lance y al completarse guarde los resultados del análisis
Aplicación	1.0
Resultado	Correcto

Tabla 18. Descripción CP_I1_10

CP_I1_11	Cancelar lanzar análisis
Versión	1.0
Descripción	El usuario cancela lanzar un análisis de un proyecto ya añadido a la aplicación
Resultado esperado	No se lanza el análisis
Aplicación	1.0
Resultado	Correcto

Tabla 19. Descripción CP_I1_11

CP_I1_12	Ver resultado de un análisis
Versión	1.0
Descripción	Se muestra al usuario todas las detecciones que se han encontrado en el análisis
Resultado esperado	Mostrar todas las detecciones de un análisis
Aplicación	1.0
Resultado	Correcto

Tabla 20. Descripción CP_I1_12

4.5.3.6 Conclusiones, planes de futuro y sugerencias para la siguiente iteración

Una vez finalizada esta iteración, se tiene creada la estructura de la aplicación. A partir de aquí solo se tiene que iterar cuantas veces se necesite para ir refinando poco a poco. Esta iteración se ha centrado prácticamente en la parte de computación, con un repositorio y una herramienta, para construir una base sobre la que poder empezar a trabajar en la interfaz. Esta base de computación ha quedado un poco rudimentaria y en un futuro habría que pulirla. Por ejemplo, para la descarga del proyecto de GitHub, en vez de hacer un clone, realiza un

wget a un link (que normalmente es estándar) donde se puede descargar el código fuente del proyecto que se encuentra en la rama master. Tampoco, al realizar el análisis se contempla la versión en la que se encuentra el proyecto para poder revisar códigos pasados desde la propia aplicación. Esto último es un buen punto a tener en cuenta para el futuro.

De momento se va a conservar esa base computacional, con ella vamos a aumentar la funcionalidad de la página. Se podrán cambiar algunas formas visuales de ordenar la información. También ver más detalles de los análisis generados y una forma de generar retroalimentación sobre ellos por parte de los usuarios. De momento nuestra prioridad es acabar la parte web. Terminar la parte de la funcionalidad que quiere el cliente en su plataforma, es decir, la retroalimentación de por parte de los usuarios. Todo esto a pesar de que no incluya gran parte de las herramientas que le gustaría.

4.5.3.7 Reunión con el cliente tras la iteración 1

Al final de esta iteración, se ha acordado una reunión con el cliente y se le ha mostrado la primera aproximación a su idea de proyecto/aplicación. Esta reunión con el cliente marcará los pasos a seguir en la o las próximas iteraciones.

El cliente parece satisfecho con este primer paso en la aplicación en lo que a visualización se refiere, la plantilla elegida y los formularios creados para las distintas funcionalidades. A pesar de esto ha hecho distintas apreciaciones a tener en cuenta para la próxima iteración.

La primera apreciación es en los reportes de análisis, la prioridad son las reglas que incumplen los distintos ficheros, dando la opción al usuario a filtrar por reglas. Todo esto sin quitar la opción de poder ver los reportes ordenados y filtrados por ficheros.

La segunda apreciación sería poder compartir los reportes que realizan los usuarios con otros que puedan estar interesados en esa información, ya sea porque pertenecen directa o indirectamente al proyecto, o también porque sea un usuario externo el que ha realizado el análisis del proyecto y quiere compartir opiniones y valoraciones sobre los resultados obtenidos.

Finalmente ha visto conveniente seguir trabajando en la retroalimentación de los resultados obtenidos por los usuarios en los reportes de análisis. Todo esto marca una línea por la que seguir trabajando en la siguiente o siguientes iteraciones.

4.5.4 Iteración 2: Retroalimentación

4.5.4.1 Introducción

Esta iteración 2 comienza guiada por las apreciaciones dadas por el cliente al final de la iteración 1, y guiada un poco también, por las conclusiones y planes de futuro sacados de esa iteración 1. A continuación se resume las actividades y operaciones que se van a realizar durante esta iteración y que se van a detallar más adelante.

La iteración anterior se ha centrado en desarrollar la primera mitad del total de la aplicación, que es el análisis de proyectos y el almacenamiento de los reportes de las herramientas. Esta va a centrar en la segunda mitad, que es la retroalimentación de los reportes obtenidos por las herramientas de análisis por parte de los usuarios. Una vez completada esta funcionalidad simplificada, se va a pasar a implementar las dos apreciaciones del cliente: el filtrado de los reportes por ficheros y por reglas incumplidas, y la opción de poder compartir reportes con otros usuarios para intercambiar valoraciones e impresiones.

Con la finalización de esta iteración se completa la estructura principal de la aplicación. A partir de aquí las siguientes iteraciones serán retoques, tanto a nivel de interfaces e interacción con el usuario, como a nivel

computacional, añadiendo nuevos repositorios y herramientas. Las siguientes interacciones podrán incluir, también, futuras especificaciones que el cliente no haya solicitado hasta ahora, o pequeños cambios, tanto funcionales como visuales.

4.5.4.2 Características a implementar

A continuación, se presentan las características principales implementadas en esta iteración.

```
Característica: Filtrado de los reportes obtenidos.

      Como usuario
      Para ver con claridad los defectos de diseño
      Quiero poder filtrar los reportes obtenidos.

Antecedentes: He iniciado sesión en el sistema, he añadido un proyecto de GitHub
y he lanzado con anterioridad un análisis, este se ha completado y estoy viendo
el resultado del análisis.

Escenario: Ordenar los resultados alfabéticamente por columnas.
      Estoy en la vista donde se pueden ver los resultados del análisis programado.
      Pulso en el título de la columna por la que quiero ordenar.
      Finalmente, los resultados se ordenan alfabéticamente por la columna que he
      pulsado.

Escenario: Buscar por palabras.
      Estoy en la vista donde se pueden ver los resultados del análisis programado.
      Pulso en el cuadro de búsqueda y escribo la palabra clave que quiero buscar.
      Finalmente, se muestran los resultados asociados a entrada introducida en el
      cuadro de búsqueda.
```

```
Característica: Ver detalles de un reporte.

      Como usuario
      Para saber más detalles de los defectos de diseño
      Quiero ver todos los detalles de un reporte.

Antecedentes: He iniciado sesión en el sistema, he añadido un proyecto de GitHub
y he lanzado con anterioridad un análisis, este se ha completado y estoy viendo
el resultado del análisis.

Escenario: Ver la descripción más detallada de la detección de un Bad Smell.
      Estoy en la vista donde se pueden ver los resultados del análisis programado.
      Pulso en ver más detalles de la detección.
      Se muestran todos los datos aportados por la herramienta de análisis.
```

```
Característica: Aportar retroalimentación de un reporte.

      Como usuario
      Para dar mi opinión sobre la detección de un reporte
      Quiero aportar la retroalimentación de un reporte.

Antecedentes: He iniciado sesión en el sistema, he añadido un proyecto de GitHub
y he lanzado con anterioridad un análisis, este se ha completado, he visto los
detalles del análisis y he visto uno de los detalles del reporte.
```

Escenario: Aportar la retroalimentación del posible Bad Smell encontrado.
Estoy viendo los detalles de uno de los reportes obtenidos.
Pulso en el botón de aportar retroalimentación.
Aparece un formulario con tres preguntas referentes al Bad Smell.
Contesto a las tres preguntas y pulso aceptar.
Finalmente, las respuestas a las preguntas quedan guardadas.

Escenario: Cancelar aportar la retroalimentación del posible Bad Smell encontrado.
Estoy viendo los detalles de uno de los reportes obtenidos.
Pulso en el botón de aportar retroalimentación.
Aparece un formulario con tres preguntas referentes al Bad Smell.
Finalmente, pulso cancelar, vuelvo a ver los detalles de unos de los reportes obtenidos y las respuestas a la retroalimentación no se guardan.

Escenario: Modificar la retroalimentación de un reporte ya introducido.
Estoy viendo los detalles de unos de los reportes obtenidos.
Pulso el botón de aportar retroalimentación.
Aparece un formulario con tres preguntas referentes al Bad Smell en el que aparece ya he introducido un reporte referente a este Bad Smell.
Modifico las tres preguntas y pulso aceptar.
Finalmente, las respuestas a las preguntas se modifican y se guardan.

Escenario: Ver la retroalimentación aportada.
Estoy viendo los detalles de unos de los reportes obtenidos.
Pulso el botón de aportar retroalimentación.
Aparece el formulario con las tres preguntas referentes al Bad Smell contestadas.
Finalmente, pulso cerrar y salgo de la vista.

Característica: Compartir resultado de un análisis mediante un enlace.

Como usuario
Para para que otros usuarios vean el resultado de un análisis
Quiero compartir el resultado de un análisis.

Antecedentes: He iniciado sesión en el sistema, he añadido un proyecto de GitHub y he lanzado con anterioridad un análisis y este se ha completado.

Escenario: Compartir resultado.
Estoy en la vista donde se pueden ver los resultados del análisis lanzado.
Pulso en el botón de compartir.
Aparece un cuadro con una url que puedo copiar para que usuarios externos puedan ver la información aportada por la herramienta.
Finalmente, copio la url.

Escenario: Cancelar compartir resultado.
Estoy en la vista donde se pueden ver los resultados del análisis lanzado.
Pulso en el botón de compartir.
Aparece un cuadro con una url que puedo copiar para que usuarios externos puedan ver la información aportada por la herramienta.
Finalmente, pulso cancelar sin copiar la url.

Característica: Compartir un reporte específico mediante un enlace.

Como usuario

```
      Para que otro usuario pueda ver los detalles de un reporte especifico
      Quiero compartir un reporte especifico mediante un enlace.

Antecedentes: He iniciado sesión en el sistema, he añadido un proyecto de GitHub
y he lanzado con anterioridad un análisis, este se ha completado, he visto los
detalles del análisis y he visto uno de los detalles del reporte.
Escenario: Compartir los detalles de un Bad Smell específico.
     Estoy viendo los detalles de uno de los reportes obtenidos.
     Pulso en el botón de compartir.
     Aparece un cuadro con una url que puedo copiar para que usuarios externos
     puedan ver la información aportada por la herramienta.
     Finalmente, copio la url.

Escenario: Cancelar compartir los detalles de un Bad Smell específico.
     Estoy viendo los detalles de uno de los reportes obtenidos.
     Pulso en el botón de compartir.
     Aparece un cuadro con una url que puedo copiar para que usuarios externos
     puedan ver la información aportada por la herramienta.
     Finalmente, pulso cancelar sin copiar la url.
```

4.5.4.3 Diagrama de clases y tablas

A continuación, se presenta la modificación de las clases, en esta iteración es mínima. Simplemente se han añadido los campos para que se puedan guardar las respuestas a la retroalimentación aportada por el usuario.

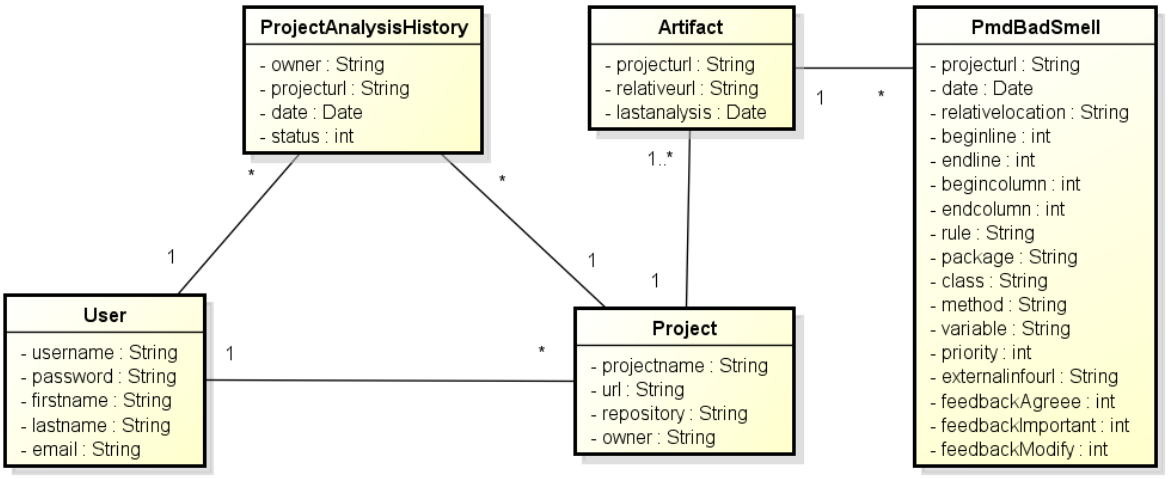


Figura 13. Diagrama de clases y tablas para la iteración 2

4.5.4.4 Capturas de la aplicación

En esta iteración conforme a las características que se especificaron, se desarrollaron las siguientes vistas:

Se ha realizado una modificación de la vista de detalles de reporte en la que se añade una tabla que permite ordenar por columnas. Así podrá ordenar por Regla, Fichero, o Prioridad. Además, hay un campo de búsqueda que permite buscar lo que quieres y se muestra en la tabla. Esta tabla es un Plugin de jquery [11].

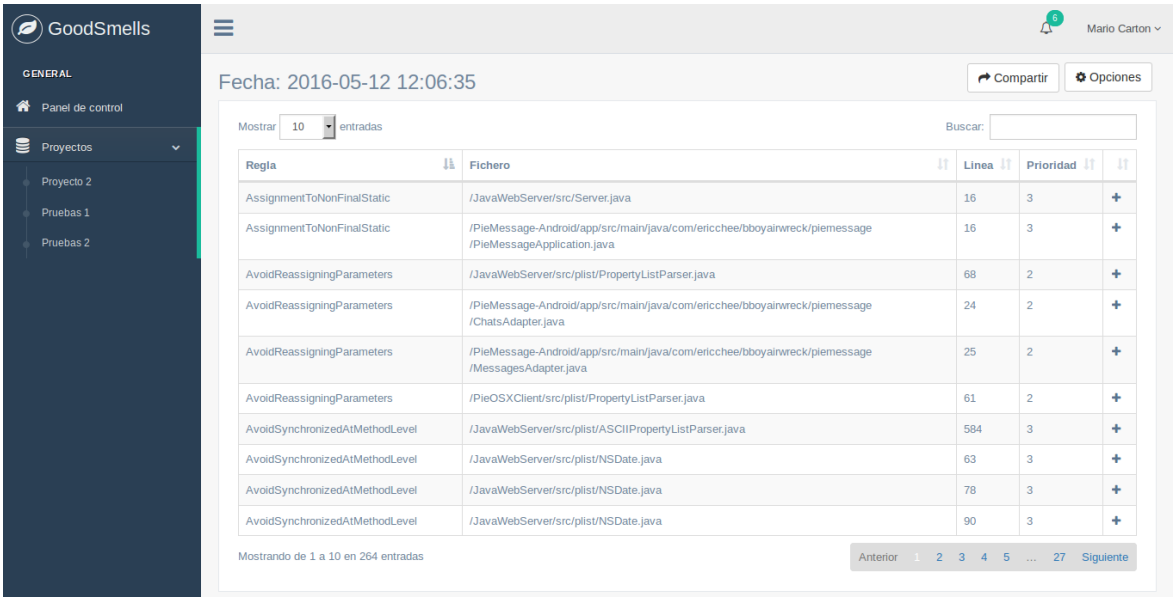


Figura 14. Vista resultado de análisis. Iteración 2

Además, en esta tabla se ha añadido un enlace que permite ver más detalles. La vista para ver más detalles es la siguiente:

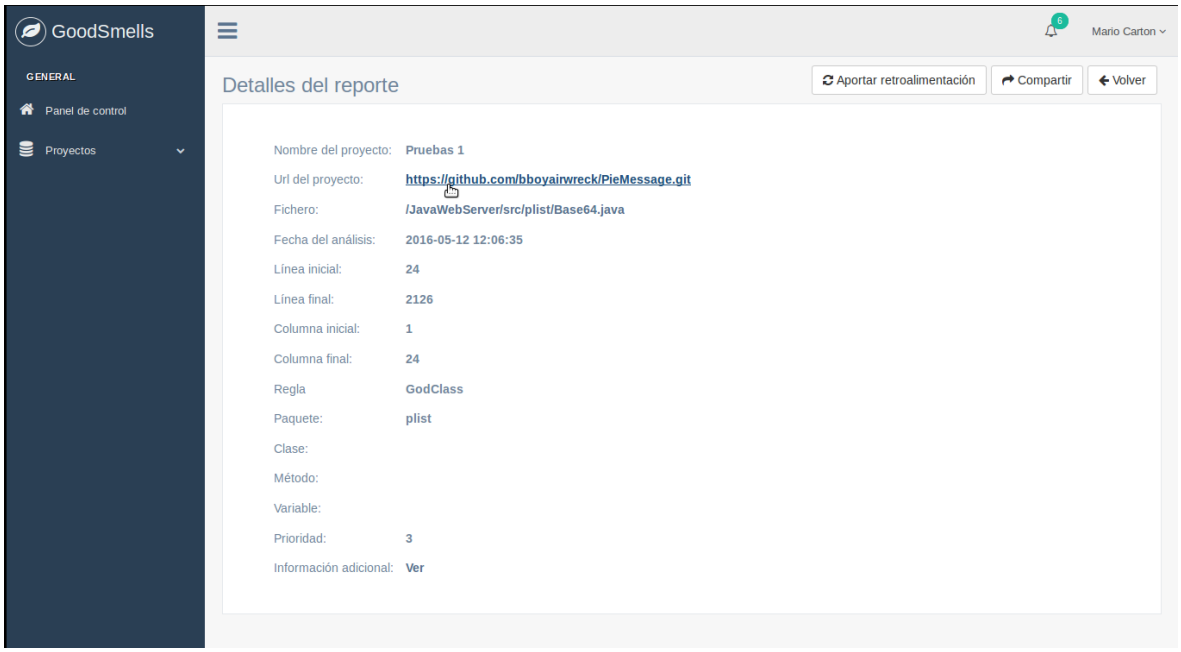


Figura 15. Vista detalles de detección. Iteración 2

Si se pulsa en los enlaces de url del proyecto lleva a la página principal en GitHub del proyecto, y si se pulsa en el enlace del fichero lleva al código de ese fichero en GitHub. En esta vista además se puede aportar la retroalimentación pulsando en el botón de retroalimentación. Muestra el siguiente formulario:

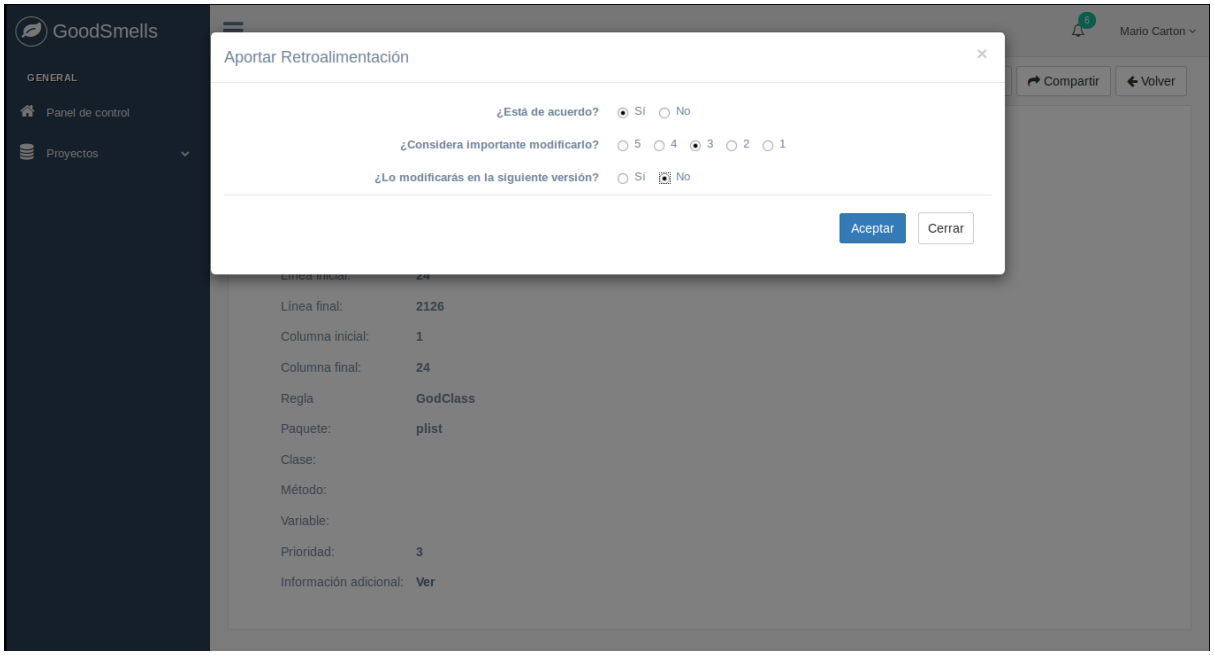


Figura 16. Vista aportar retroalimentación. Iteración 2

Tanto en la vista de resumen del análisis como en la vista de ver detalles hay un botón que permite compartir una cosa u otra apareciendo el siguiente cuadro de diálogo:

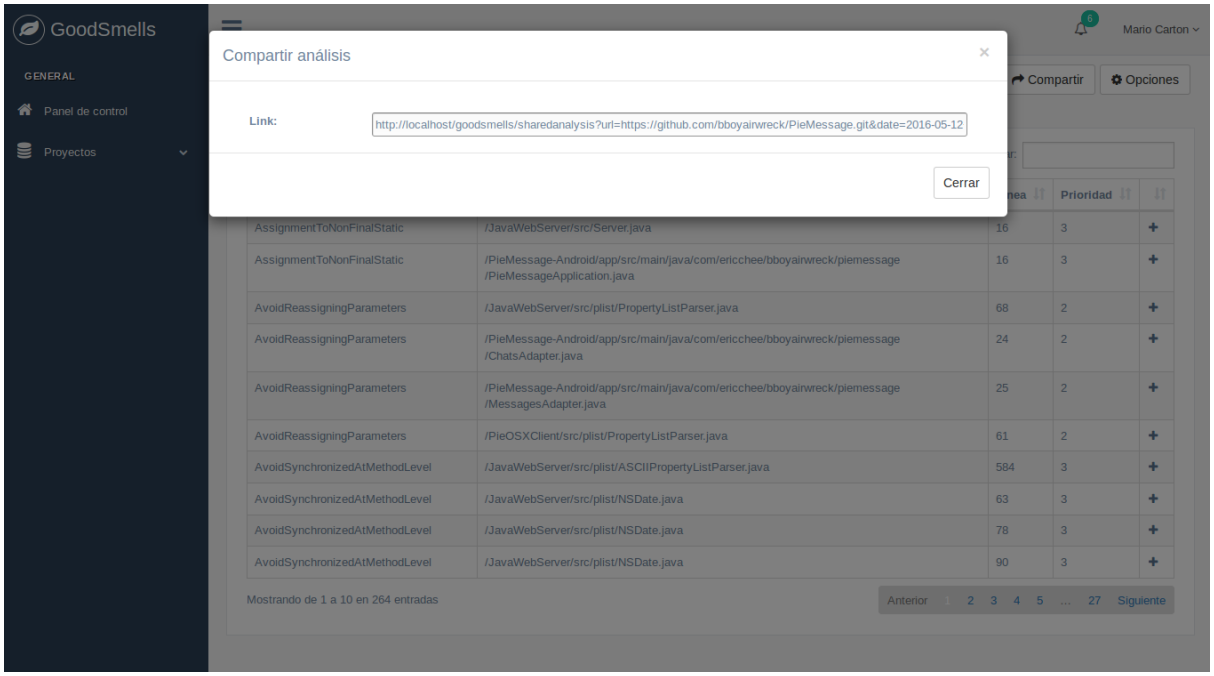
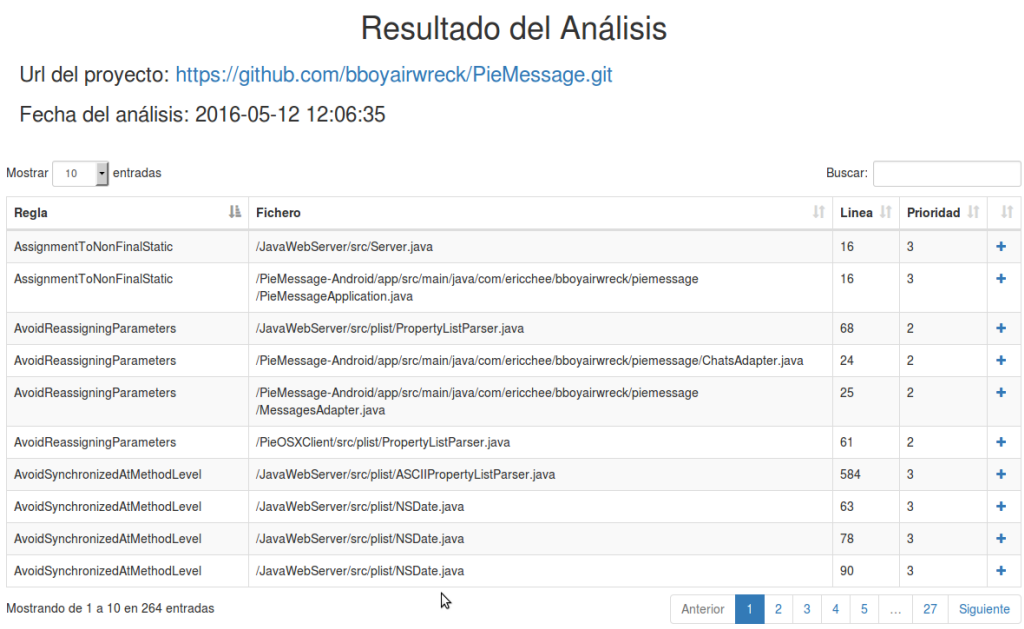


Figura 17. Vista compartir análisis. Iteración 2

La visualización en ese link de todo el resumen de análisis es la siguiente:



Resultado del Análisis

Url del proyecto: https://github.com/bboyairwreck/PieMessage.git

Fecha del análisis: 2016-05-12 12:06:35

Mostrar 10 entradas Buscar:

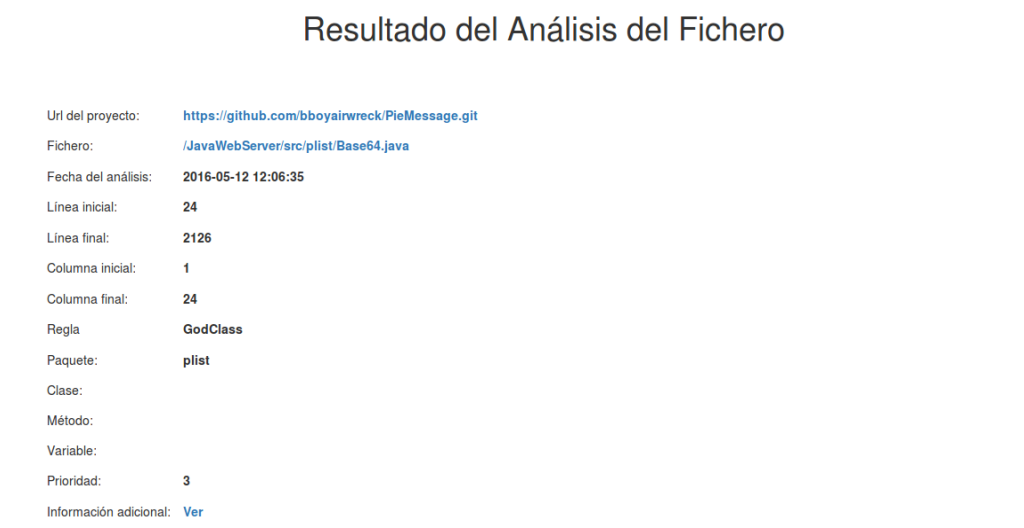
Regla	Fichero	Linea	Prioridad	
AssignmentToNonFinalStatic	/JavaWebServer/src/Server.java	16	3	+
AssignmentToNonFinalStatic	/PieMessage-Android/app/src/main/java/com/ericchee/bboyairwreck/piemessage /PieMessageApplication.java	16	3	+
AvoidReassigningParameters	/JavaWebServer/src/plist/PropertyListParser.java	68	2	+
AvoidReassigningParameters	/PieMessage-Android/app/src/main/java/com/ericchee/bboyairwreck/piemessage/ChatsAdapter.java	24	2	+
AvoidReassigningParameters	/PieMessage-Android/app/src/main/java/com/ericchee/bboyairwreck/piemessage /MessagesAdapter.java	25	2	+
AvoidReassigningParameters	/PieOSXClient/src/plist/PropertyListParser.java	61	2	+
AvoidSynchronizedAtMethodLevel	/JavaWebServer/src/plist/ASCIIPropertyListParser.java	584	3	+
AvoidSynchronizedAtMethodLevel	/JavaWebServer/src/plist/NSDate.java	63	3	+
AvoidSynchronizedAtMethodLevel	/JavaWebServer/src/plist/NSDate.java	78	3	+
AvoidSynchronizedAtMethodLevel	/JavaWebServer/src/plist/NSDate.java	90	3	+

Mostrando de 1 a 10 en 264 entradas

Anterior 1 2 3 4 5 ... 27 Siguiente

Figura 18. Vista resultado de análisis compartido. Iteración 2

La visualización de los detalles de una detección en concreto es la siguiente:



Resultado del Análisis del Fichero

Url del proyecto:	https://github.com/bboyairwreck/PieMessage.git
Fichero:	/JavaWebServer/src/plist/Base64.java
Fecha del análisis:	2016-05-12 12:06:35
Línea inicial:	24
Línea final:	2126
Columna inicial:	1
Columna final:	24
Regla	GodClass
Paquete:	plist
Clase:	
Método:	
Variable:	
Prioridad:	3
Información adicional:	Ver

Figura 19. Vista detalles de detección compartidos. Iteración 2

Estas dos vistas son muy similares a las que se muestran cuando se es un usuario registrado.

4.5.4.5 Pruebas realizadas durante esta iteración

A continuación, se muestran las pruebas realizadas para cada una de las características implementadas en esta iteración.

CP_I2_01	Ordenar los resultados alfabéticamente por columnas
Versión	1.0
Descripción	Los resultados del análisis se muestran en una columna que permite al usuario ordenar por columnas
Resultado esperado	Al pulsar en el título de una columna se ordene alfabéticamente por esa columna
Aplicación	1.0
Resultado	Correcto

Tabla 21. Descripción CP_I2_01

CP_I2_02	Buscar por palabras
Versión	1.0
Descripción	El usuario puede buscar un resultado esperado por palabras
Resultado esperado	Al introducir la palabra que se quiere buscar se muestran los resultados relacionados
Aplicación	1.0
Resultado	Correcto

Tabla 22. Descripción CP_I2_02

CP_I2_03	Ver la descripción detallada de un Bad Smell
Versión	1.0
Descripción	El usuario ve más detalles acerca de una detección encontrada
Resultado esperado	Al pulsar en ver más detalles se muestran todos.
Aplicación	1.0
Resultado	Correcto

Tabla 23. Descripción CP_I2_03

CP_I2_04	Aportar la retroalimentación de un posible Bad Smell encontrado
Versión	1.0
Descripción	El usuario aporta una retroalimentación de una detección resultante del análisis
Resultado esperado	La retroalimentación se guarda
Aplicación	1.0
Resultado	Primera ejecución: Incorrecto Causa: La retroalimentación no se guarda correctamente Acciones realizadas: Corregir el código del controlador y las clases implicadas en el controlador Segunda ejecución: Correcta

Tabla 24. Descripción CP_I2_04

CP_I2_05	El usuario cancela aportar retroalimentación de un posible Bad Smell
Versión	1.0
Descripción	El usuario cancela el intento de aportar una retroalimentación de un posible Bad Smell
Resultado esperado	La retroalimentación no se guarda
Aplicación	1.0
Resultado	Correcto

Tabla 25. Descripción CP_I2_05

CP_I2_06	Modificar la retroalimentación de un posible Bad Smell
Versión	1.0
Descripción	El usuario modifica una retroalimentación anteriormente realizada
Resultado esperado	Se guarda la última retroalimentación aportada
Aplicación	1.0
Resultado	Primera ejecución: Incorrecto Causa: La modificación de la retroalimentación no se guarda correctamente Acciones realizadas: Corregir el código del controlador y las clases implicadas en el controlador Segunda ejecución: Correcta

Tabla 26. Descripción CP_I2_06

CP_I2_07	Ver la retroalimentación aportada
Versión	1.0
Descripción	El usuario puede ver la retroalimentación que ha aportado anteriormente
Resultado esperado	Se muestra la retroalimentación aportada
Aplicación	1.0
Resultado	Correcto

Tabla 27. Descripción CP_I2_07

CP_I2_08	Compartir resultado
Versión	1.0
Descripción	El usuario comparte el resumen de análisis de un proyecto
Resultado esperado	Se muestra a un usuario externo el resumen de análisis de un proyecto
Aplicación	1.0
Resultado	Correcto

Tabla 28. Descripción CP_I2_08

CP_I2_09	Cancelar compartir resultado
Versión	1.0
Descripción	El usuario cancela compartir resumen de análisis de un proyecto
Resultado esperado	No se comparte el resumen de análisis
Aplicación	1.0
Resultado	Correcto

Tabla 29. Descripción CP_I2_09

CP_I2_10	Compartir detalles de un Bad Smell
Versión	1.0
Descripción	El usuario comparte los detalles de un Bad Smell
Resultado esperado	Se muestra a un usuario externo los detalles de un Bad Smell
Aplicación	1.0
Resultado	Correcto

Tabla 30. Descripción CP_I2_10

CP_I2_11	Cancelar compartir detalles de un Bad Smell
Versión	1.0
Descripción	El usuario cancela compartir detalles de un Bad Smell
Resultado esperado	No se comparten los detalles de un Bad Smell
Aplicación	1.0
Resultado	Correcto

Tabla 31. Descripción CP_I2_11

4.5.4.6 Conclusiones, planes de futuro y sugerencias para la siguiente iteración

Una vez finalizada esta iteración, se ha completado la estructura principal de la aplicación. A partir de ahora hay distintas partes en las que podemos trabajar: mejoras en la interfaz para mejorar la experiencia del usuario y facilitar su uso, mejoras en la funcionalidad añadiendo más herramientas de análisis y más repositorios sobre los que trabajar y, por último, mejoras en la computación y en la seguridad.

Uno de los cambios más interesante a realizar sería el mencionado en las conclusiones de la iteración 1, el cambio de wget por clone. De esta manera se podrá guardar la versión y la rama en la que se va a realizar el análisis para poder acceder al código en un futuro. De esa forma no tendremos problemas si hay nuevas actualizaciones de código sobre la rama master. De momento no se contempla añadir otro repositorio más hasta que no se haya modificado esta parte computacional de GitHub.

Otro de los cambios interesantes sería añadir otra herramienta (la candidata más posible es jdeodorant). Añadir esta herramienta supone dar un poco más de tonalidad a los Bad Smells detectados, con lo que podremos ver también las diferencias de criterio entre las dos.

También se puede modificar la característica de compartir los resultados y reportes. Esta modificación supondría poder distinguir entre compartir internamente, dentro de la aplicación, o externamente como está implementado ahora, para que cualquier usuario externo lo pueda ver.

4.5.4.7 Reunión con el cliente tras la iteración 2

Al concluir esta iteración se ha tenido una reunión con el cliente en la que se le han mostrado la aplicación ya con una estructura acabada. Tras esto se ha explorado las siguientes vías de desarrollo que puede tener la aplicación.

La primera idea del cliente ha sido añadir otro repositorio más de donde poder analizar los proyectos. El repositorio elegido era SourceForge. Al cliente le interesa esta vía de desarrollo porque quiere simular un experimento con proyectos que se encuentran allí alojados. Pero no se aconsejó esa vía de desarrollo, ya que se ha determinado que hasta que no esté bien implementada la parte de la descarga y recopilación de información de las versiones y el estado del proyecto alojado, no se va a añadir otro repositorio.

Por otro lado, hay un interés otra añadir otra herramienta más en la detección de Design Smells. La candidata seria jDeodorant, ya que PMD no reporta el *Design Smell Feature Envy* y jDeodorant sí. De esta manera se tendría cubierta la detección de estos dos *Design Smells* tan conocidos. Además, añadir una herramienta más a la aplicación, es una forma de dar una diversidad de opiniones en cuanto a detección, pudiendo así observar los distintos criterios que tienen estas dos herramientas.

Se valoraron las dos opciones, por un lado, resolver el problema con GitHub, guardar las versiones de los ficheros, de la aplicación, las ramas, los enlaces a los ficheros de esa versión etc. y al término de esto añadir el segundo repositorio, SourceForge, con las mismas características que se ha añadido a GitHub. Y por otro lado añadir la segunda herramienta, jDeodorant, que de una segunda opinión en la detección de *Design Smells* con todos los cambios, tanto en interfaz, como computacionalmente hablando.

Finalmente se optó por la segunda opción. Añadir una segunda herramienta y abarcar los casos de detección de *God Class* y *Feature Envy*. El principal motivo de esta elección ha sido que añadir esta característica ayudará a realizar el manual de mantenimiento y extensibilidad, sobre todo en la parte de extensibilidad.

4.5.5 Iteración 3: jDeodorant

4.5.5.1 Introducción

Esta iteración número 3 va a ser la última que se va a realizar. Consiste en añadir la herramienta de detección de *Design Smells*, jDeodorant. Esta última iteración implica un avance computacional, aunque quizá visualmente o para la experiencia de usuario no represente un gran avance.

jDeodorant es un plugin de eclipse con el que se ha adquirido una familiarización en el mismo eclipse, analizando proyectos descargados de GitHub. Esta familiarización se produjo durante la iteración 0. Su uso en eclipse es bastante sencillo. En el portal del proyecto jDeodorant se vio una forma de ejecutarlo por línea de comandos. Esta forma de ejecución es la que interesa para poder añadir herramientas de detección a la aplicación. Encontrar un tutorial en el que parece sencilla la ejecución por línea de comandos es lo que motivó a elegir jDeodorant en esta iteración. Para el resto de las otras 3 herramientas (iPlasma, DECOR, Together) habría que extraer de su código fuente la parte dedicada a la detección de *Design Smells* y ingeniárselas para ejecutarlo por línea de comandos y realizar una salida fácilmente filtrable.

4.5.5.2 Características a implementar

A continuación, se presentan las características principales implementadas en esta iteración.

```
Característica: Lanzar un análisis con la herramienta jDeodorant.
```

Como usuario
Para poder ver reportes de la herramienta jDeodorant
Quiero lanzar un análisis con la herramienta jDeodorant.

Antecedentes: He iniciado sesión en el sistema y he añadido un proyecto de GitHub.

Escenario: Lanzar análisis.
Estoy en el panel de control de mis proyectos
Pulso en el proyecto que quiero analizar
Se muestran una serie de análisis realizados anteriormente, en el caso que se haya realizado alguno
Pulso en el botón de realizar nuevo análisis
Aparece un formulario en el que aparecen las opciones de las herramientas disponibles.
Marco la opción de jDeodorant.
Finalmente, pulso aceptar para lanzar el análisis.

Escenario: Cancelar lanzar análisis.
Estoy en el panel de control de mis proyectos
Pulso en el proyecto que quiero analizar
Se muestran una serie de análisis realizados anteriormente, en el caso que se haya realizado alguno
Pulso en el botón de realizar nuevo análisis
Aparece un formulario en el que aparecen las opciones de las herramientas disponibles.
Finalmente, pulso cancelar y vuelvo al panel de control de mis proyectos.

Característica: Ver el resultado de un análisis de un proyecto con la herramienta jDeodorant.

Como usuario
Para saber los defectos de diseño de un proyecto alojado en GitHub
Quiero ver los resultados del análisis de la herramienta jDeodorant.

Antecedentes: He iniciado sesión en el sistema, he añadido un proyecto de GitHub y he lanzado con anterioridad un análisis y este se ha completado.

Escenario: Ver resultado de un análisis
Estoy en el panel principal de mis proyectos
Pulso en el proyecto que quiero ver el análisis planificado con anterioridad
Aparecen una lista de análisis realizados, que en el caso de que sea el primero solo aparecerá el que queremos
Pulso en el análisis que quiero ver
Finalmente aparecen un resumen de los datos que jDeodorant otorga tras el análisis.

Característica: Filtrado de los reportes obtenidos.

Como usuario
Para distinguir los distintos resultados entre herramientas
Quiero filtrar los reportes obtenidos.

```
Antecedentes: He iniciado sesión en el sistema, he añadido un proyecto de GitHub
y he lanzado con anterioridad un análisis, este se ha completado y estoy viendo
el resultado del análisis.

Escenario: Filtrar alfabéticamente por herramientas.
      Estoy en la vista donde se pueden ver los resultados del análisis programado.
      Pulso en el título de la columna de herramienta por la que ha sido detectada.
      Finalmente, los resultados se ordenan alfabéticamente por herramientas.

Escenario: Filtrar por ficheros.
      Estoy en la vista donde se pueden ver los resultados del análisis programado.
      Introduzco el nombre o parte del nombre del fichero.
      Finalmente, se muestra las detecciones de ese fichero por las distintas
      herramientas.
```

4.5.5.3 Diagramas de clases

Debido a distintos sucesos que se detallan en las siguientes secciones, se mantiene igual que en las iteraciones anteriores.

4.5.5.4 Capturas de la aplicación

Debido a distintos sucesos que se detallan en las siguientes secciones, se mantiene igual que en las iteraciones anteriores.

4.5.5.5 Problemas surgidos en esta iteración

Como se ha descrito anteriormente, jDeodorant es un plugin de eclipse, por lo tanto, en un principio no se puede ejecutar por línea de comandos. Para añadir una herramienta a la aplicación es necesario que se pueda ejecutar de esa forma. Pero los desarrolladores de jDeodorant en su propia web tienen un tutorial de como ejecutar jDeodorant por línea de comandos [16]. Este tutorial es el que se ha seguido para conseguir la ejecución por línea de comandos. Este tutorial explica cómo crear una aplicación eclipse que ejecuta un plugin, en este caso jDeodorant, por línea de comandos. Pero durante el transcurso de plugin eclipse a elemento ejecutable en shell, han sucedido una serie de problemas.

El primer problema encontrado ha sido durante la realización del tutorial. El comando de ejecución no es el que pone en el tutorial, o al menos, no funcionaba en este caso. La solución se encontró, tras mucha búsqueda, en el propio eclipse. Al ejecutar esa aplicación en el propio eclipse primeramente pone el comando que emplea. El comando que emplea eclipse sí que funciona en la ejecución en otra terminal linux y es el que se emplea aquí.

El segundo problema fue a la hora de introducir el ejecutable y el comando en la aplicación, más concretamente al ejecutar el comando en el script PHP que se ejecuta en segundo plano dentro de la aplicación (los detalles de funcionamiento de la aplicación se dan más adelante). El comando necesita ser ejecutado, al menos en esta situación, como sudo, porque necesita, en este caso, ciertos permisos de escritura en la ubicación del proyecto a analizar. El comando crea ciertos ficheros temporales en la ubicación del proyecto. El problema es que al ejecutar *shell_exec* en PHP no se puede introducir la contraseña del usuario, o al menos, no se quiere dejar ahí reflejada. La solución adoptada para este problema, ha sido ejecutar el comando *sudo visudo* y permitir la ejecución como sudo sin contraseña del shellscript jdeodorant.sh que se encuentra en tools/shellscripts.

A partir de este problema surgió un problema anidado: no se sabía a qué usuario dar el privilegio de ejecutar el shellscript sin contraseña. La forma de encontrar esta solución fue mandar ejecutar a la aplicación el comando *whoami* y redirigir la salida a un fichero. En ese fichero se encuentra que el usuario que ejecuta el comando es

daemon, tiene sentido pues todas las operaciones computacionales se ejecutan en segundo plano. Finalmente, esto es lo que se tenía que añadir en *sudo visudo* (debajo de *%sudo ALL=(ALL:ALL) ALL*)

```
daemon ALL=(ALL) NOPASSWD: <ruta al proyecto>/tools/shellscripts/jdeodorant.sh
```

Después de esto, otro problema que se ha tenido fue indicar correctamente la ruta del directorio a analizar. En este caso ha sido por despiste, pero es el típico fallo tonto que da un montón de dolores de cabeza. Ha sido muy importante durante todo el proyecto las rutas y los permisos a directorios y ficheros. Una vez solucionado todo hay que pasar al filtrado de la información y más tarde guardarla en la base de datos.

Una vez solucionado el problema de la ejecución se ha estudiado el comportamiento de jDeodorant ejecutado en línea de comandos y por la aplicación. Se ha observado distintos comportamientos por parte de jDeodorant en distintos casos. Los distintos comportamientos se describen a continuación.

- En ciertas ejecuciones por línea de comandos eclipse fallaba y no iniciaba.

- En ciertas ejecuciones por línea de comandos y en la aplicación, eclipse devolvía una gran cantidad de fallos, referidos a múltiples recursos necesarios por el plugin y por la aplicación eclipse creada para su ejecución por línea de comandos. No daba ningún resultado de jDeodorant.

- En ciertas ejecuciones por línea de comandos y en la aplicación, eclipse devolvía gran cantidad de fallos como los anteriores y, además, entre estos fallos, la salida de la ejecución de jDeodorant.

- En ciertas ejecuciones por línea de comandos y en la aplicación, eclipse, además de los fallos anteriormente descritos, no se detectaba bien los defectos de diseño, daba defectos de diseño en clases que no se encontraban en los directorios que estaba analizando.

- En ciertas ejecuciones por línea de comandos y en la aplicación, eclipse además de los fallos anteriormente descritos, no profundizaba bien en el árbol de directorios y no encontraba ningún defecto de diseño, por no encontrar ningún fichero. Esto le pasa sobre todo en proyectos grandes que tienen gran cantidad de directorios y subdirectorios y que tienen una organización arquitectónica compleja.

Todos estos comportamientos han llevado a que se descarte la integración de jDeodorant en esta iteración en el sistema. Se descarta por la imposibilidad y falta de tiempo de resolver todos estos errores y fallos en esta iteración. Sin embargo, se ha dejado añadido jDeodorant dentro de la aplicación, pero deshabilitado. Desde la aplicación no se puede ejecutar, pero se ha dejado preparado para que si se consiguen resolver estos problemas poder añadirlo.

A causa de todos estos problemas no se ha podido avanzar en la funcionalidad de la aplicación. Quedando cara al usuario la misma funcionalidad que en la iteración anterior.

4.5.5.6 Conclusiones y planes de futuro

Con esta iteración concluye el desarrollo de la aplicación. Esta iteración concluye con un desarrollo fallido de una funcionalidad. jDeodorant ha quedado integrado en la aplicación, pero anulada su funcionalidad debido a una baja fiabilidad, tanto en su ejecución como en los resultados otorgados. Con todos los fallos que se han encontrado ha decidido no continuar con el desarrollo para la integración de jDeodorant.

A partir de aquí, estaría bien mejorar la parte computacional de la descarga de Github. Sería recomendable realizar un clone del proyecto y emplear los distintos comandos que git proporciona para completar la información acerca del estado de la aplicación. Además de mejorar la parte de GitHub, también es interesante añadir el repositorio SourceForge al mismo nivel computacional que se complete GitHub. Estas dos tareas son las más sencillas y con las que se va a notar cambio en la funcionalidad de la aplicación.

Con respecto a añadir nuevas herramientas, se ha comprobado que es una tarea más costosa que lo que en un principio se pensaba. El primer paso para el desarrollo por esta parte computacional de las herramientas, sería intentar solucionar del todo el problema que hay con jDeodorant. Este paso incluiría ya revisar código. Otros pasos que también se pueden dar en este aspecto es añadir alguna otra herramienta. Añadir otra herramienta que no sea jDeodorant o PMD, implica extraer el código que se encarga de la detección en cualquiera de las tres herramientas que se elija y adaptarlo a una ejecución por línea de comandos. Esto hay que hacerlo porque las otras tres herramientas que quedan por añadir tienen interfaz gráfica y muestran sus resultados mediante la interfaz. Es decir, que añadir cualquier herramienta más de las propuestas es un trabajo muy laborioso.

4.6 ESTADO FINAL DE LA APLICACIÓN

4.6.1 Introducción

En esta sección se describe el estado en el que ha quedado la aplicación tras las tres iteraciones anteriormente descritas. Finalmente, en la aplicación se pueden añadir proyectos de GitHub y analizarlos únicamente con la herramienta PMD. Aunque la herramienta jDeodorant ha quedado añadida se ha deshabilitado su uso por las razones que se comentan en la última iteración. Después de los análisis se pueden ver y compartir los resúmenes de análisis y los detalles correspondientes a cada iteración. Una vez realizados los análisis, también se puede aportar una retroalimentación respecto a cada una de las detecciones que da la herramienta.

A continuación, se muestra un diagrama final de la aplicación que completa el diagrama inicial de la arquitectura descrito en la sección. También se muestran unos diagramas de actividades para ver el funcionamiento.

4.6.2 Diagrama final de la implementación

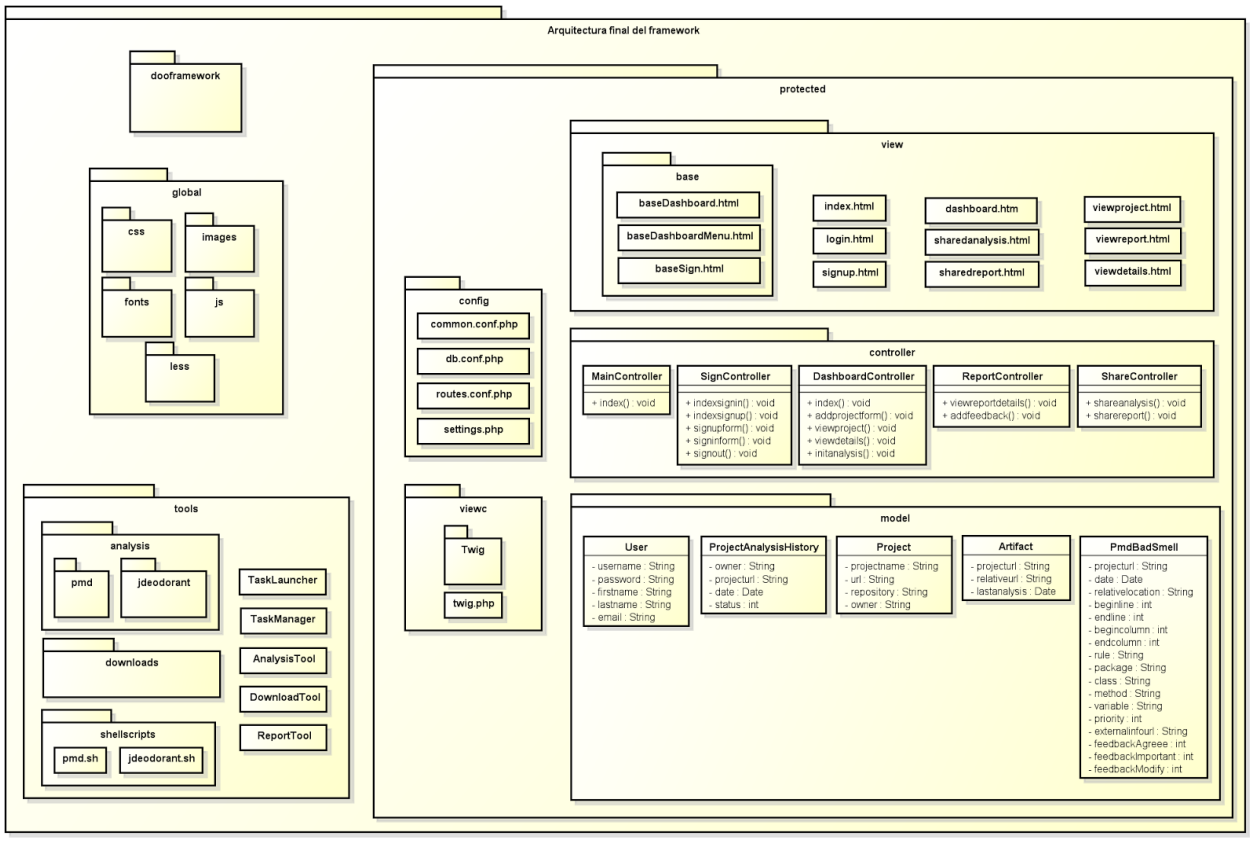


Figura 20. Arquitectura final de la aplicación

4.6.3 Estructura de la aplicación

Referente al diagrama anterior, en esta sección se explica estructura que tiene y que viene dada por el propio framework:

- **dooframework:** es el directorio que contiene el propio framework. No es necesario editar nada dentro de este directorio para cualquier acción de mantenimiento o extensibilidad.
- **global:** en este directorio se encuentran las imágenes, css, javascript y algún otro recurso de las vistas.
- **protected:** este es el directorio sobre el que se trabaja la mayoría del tiempo.
    - **config:** contiene los ficheros de configuración y enrutamiento.
        - **common.conf.php:** fichero de configuración general del framework.
        - **db.conf.php:** fichero que contiene la configuración para el acceso a la base de datos.
        - **routes.conf.php:** fichero de enrutamiento de las páginas o vistas de la aplicación. Aquí se indica que controlador se lanza en función de qué dirección introduzca el usuario.
        - **settings.php:** fichero en el que se encuentran algunas variables interesantes para el desarrollo de la aplicación.
    - **controller:** directorio que contiene los controladores de las vistas.
    - **model:** directorio que contiene los modelos de la aplicación. Estos modelos son especiales ya que utilizan una estructura propia del framework para el acceso a datos.

- **view:** directorio donde se encuentran las vistas de la aplicación. Estas vistas también son especiales, usan Twig. En el directorio "base" se encuentran las vistas de las cuales heredan el resto de las vistas que están fuera de ese subdirectorio.
- **viewc:** directorio en el que se encuentra alojado Twig. twig.php se encarga de integrar Twig en DooPHP.

- **tools:** aquí se encuentran las herramientas que utiliza la aplicación referente a la parte de computación. Aquí también se encuentra todo el código encargado de la descarga, análisis y posterior almacenamiento de los análisis en la base de datos.
  - **analysis:** en este directorio se encuentran las herramientas de análisis. Cuando se quiera añadir alguna herramienta más se debe alojar aquí.
  - **downloads:** directorio temporal donde se alojan los proyectos de los usuarios durante un análisis. Dentro de este directorio, hay subdirectorios para cada usuario.
  - **shellscripts:** este directorio contiene los scripts que lanzan las herramientas alojadas en el directorio analysis.
  - **TaskLauncher.php:** esta clase es la encargada de lanzar el análisis del proyecto. Esta clase es llamada desde el controlador DashboardController.php en el método initanalysis(). Esta clase prepara un comando, para la ejecución de un script, en función de las herramientas seleccionadas para lanzarlo en segundo plano.
  - **TaskManager.php:** este es el script que lanza TaskLauncher.php. Este script filtra las opciones que le pasa TaskManager.php y en función de las opciones realiza tres tareas: descarga, análisis y almacenamiento de resultados.
  - **DownloadTool.php:** clase encargada de descargar el proyecto.
  - **AnalysisTool.php:** clase que se encarga de lanzar los scripts de las herramientas almacenadas en el directorio shellscripts.
  - **ReportTool.php:** clase que se encarga de guardar en la base de datos los resultados de los análisis.
- **index.php:** es el fichero de arranque del framework.

4.6.4 Diagramas de actividades

A continuación, se muestran los diagramas de actividades de las características más importantes.

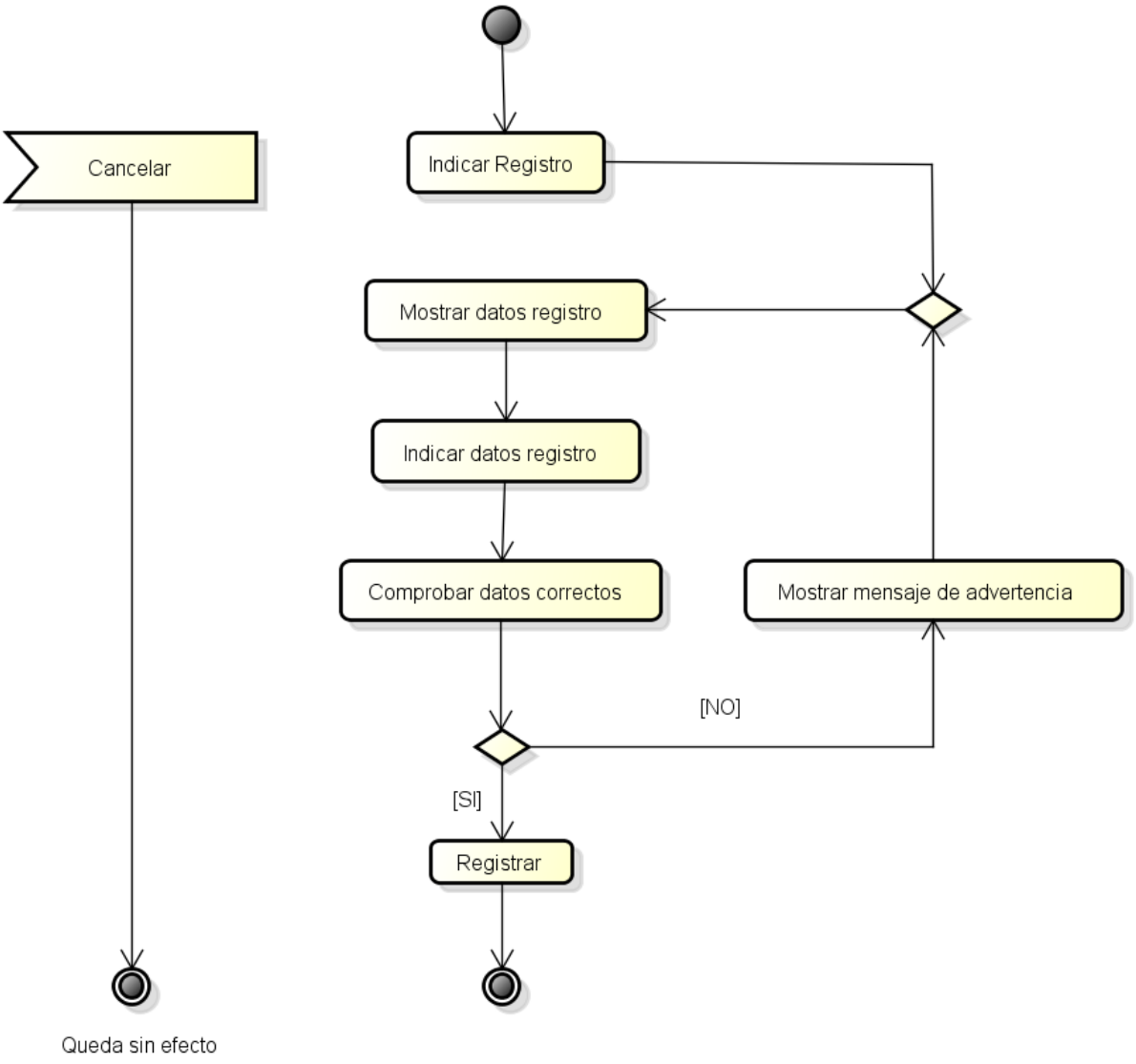


Figura 21. Diagrama de actividades. Registro de usuario

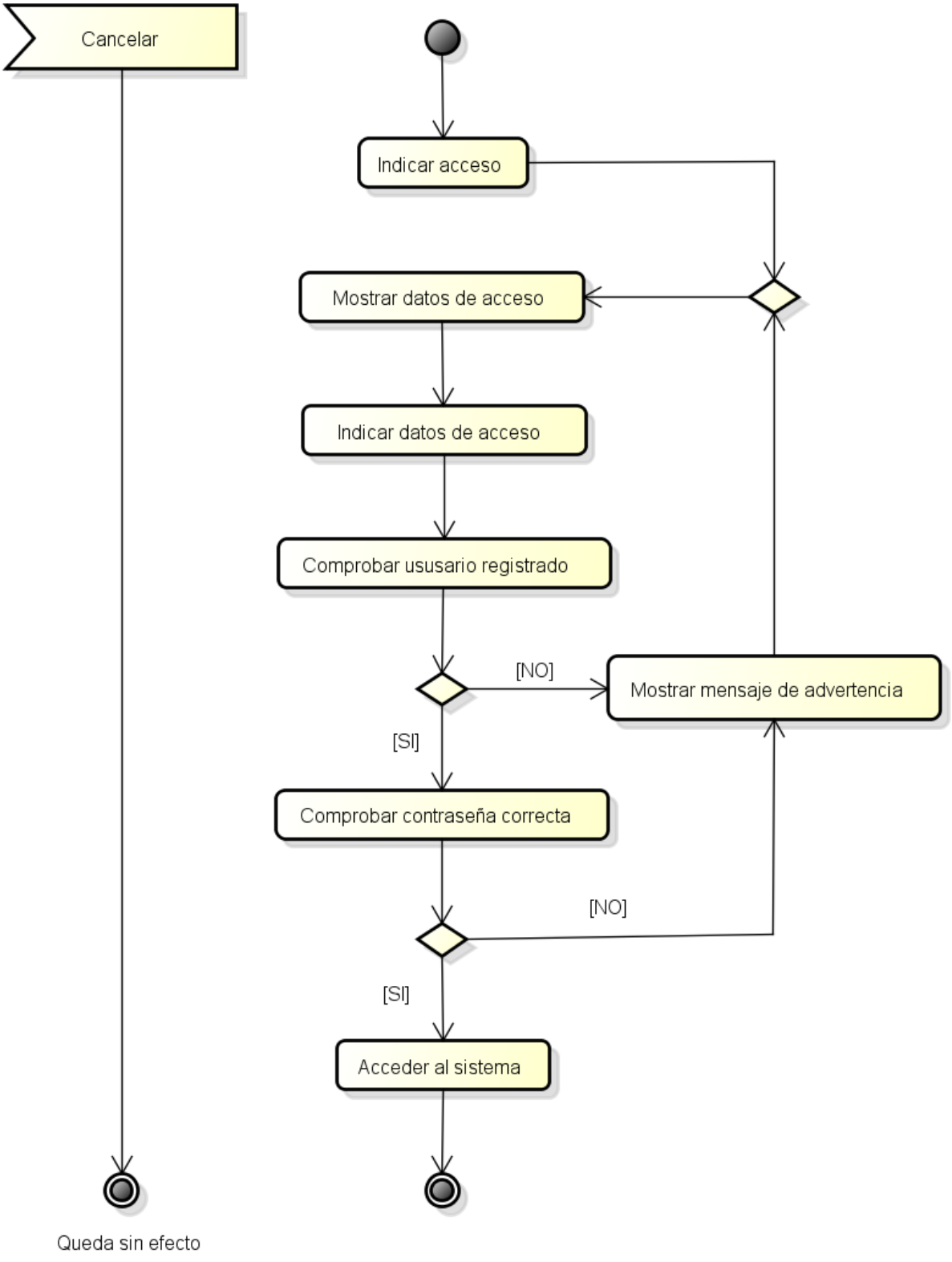


Figura 22. Diagrama de actividades. Inicio de sesión

<<antecedentes>> Se ha iniciado sesión en el sistema y he añadido un proyecto de GitHub.

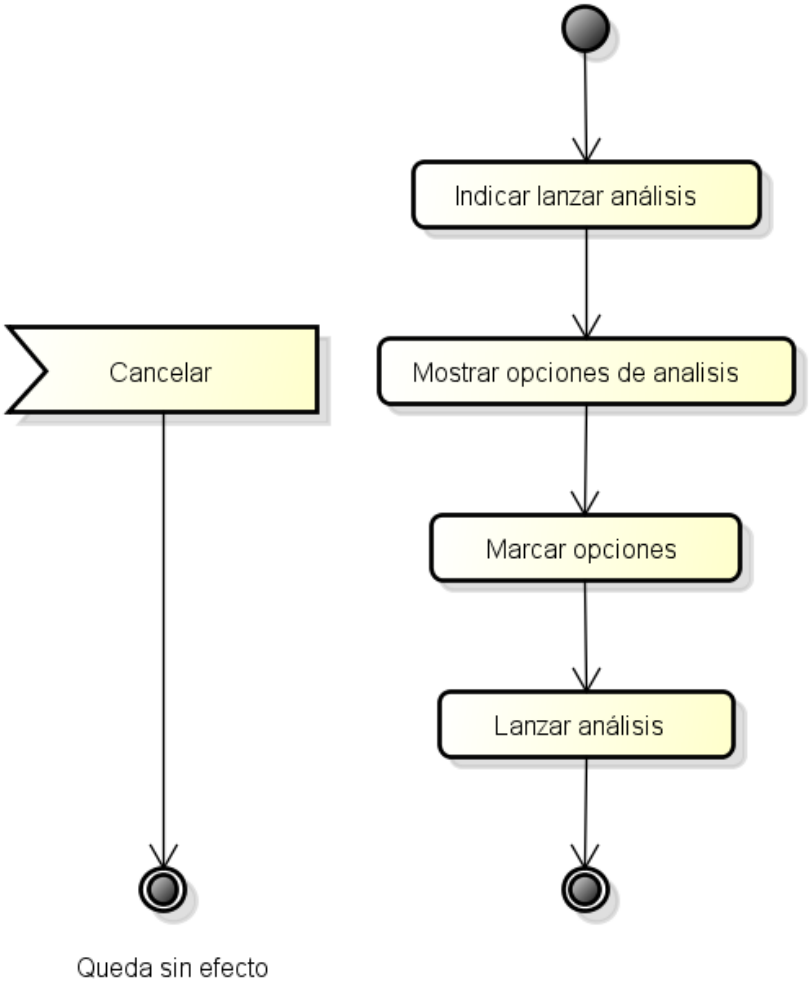


Figura 23. Diagrama de actividades. Lanzar análisis

<<antecedentes>> Se ha iniciado sesión en el sistema y se ha añadido un proyecto GitHub y se ha lanzado un analisis

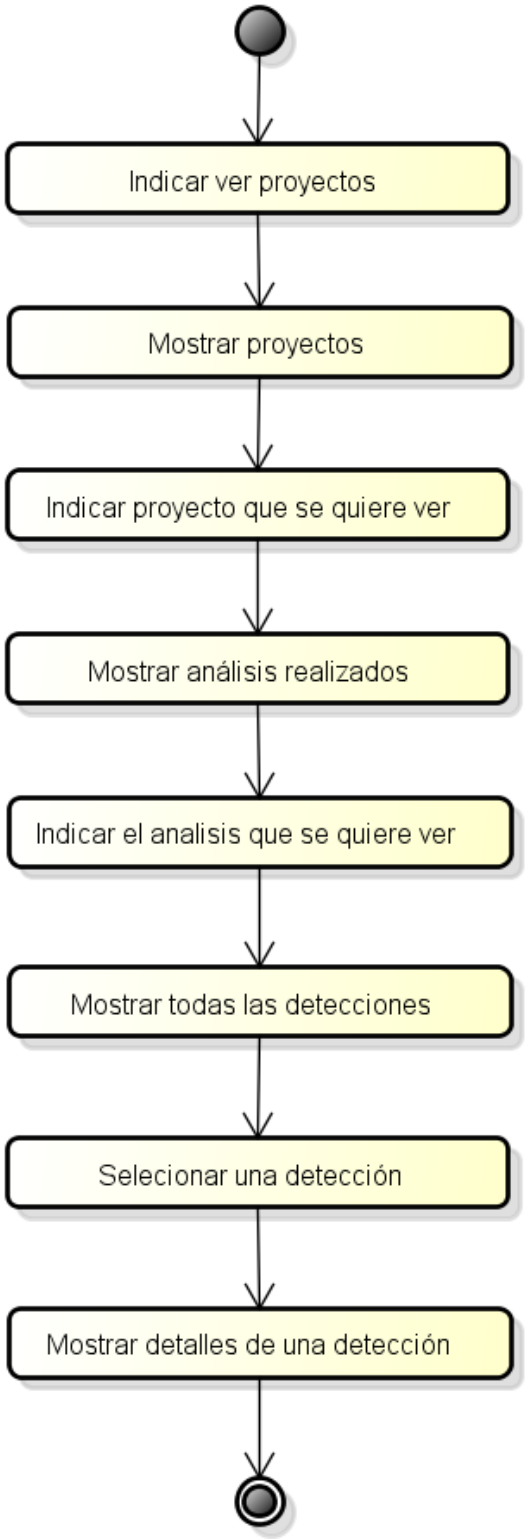


Figura 24. Diagrama de actividades. Ver detalles de análisis

<<antecedentes>> Se ha iniciado sesión en el sistema, se ha añadido un proyecto de GitHub y se ha lanzado un análisis, este se ha completado, se han visto los detalles de un reporte.

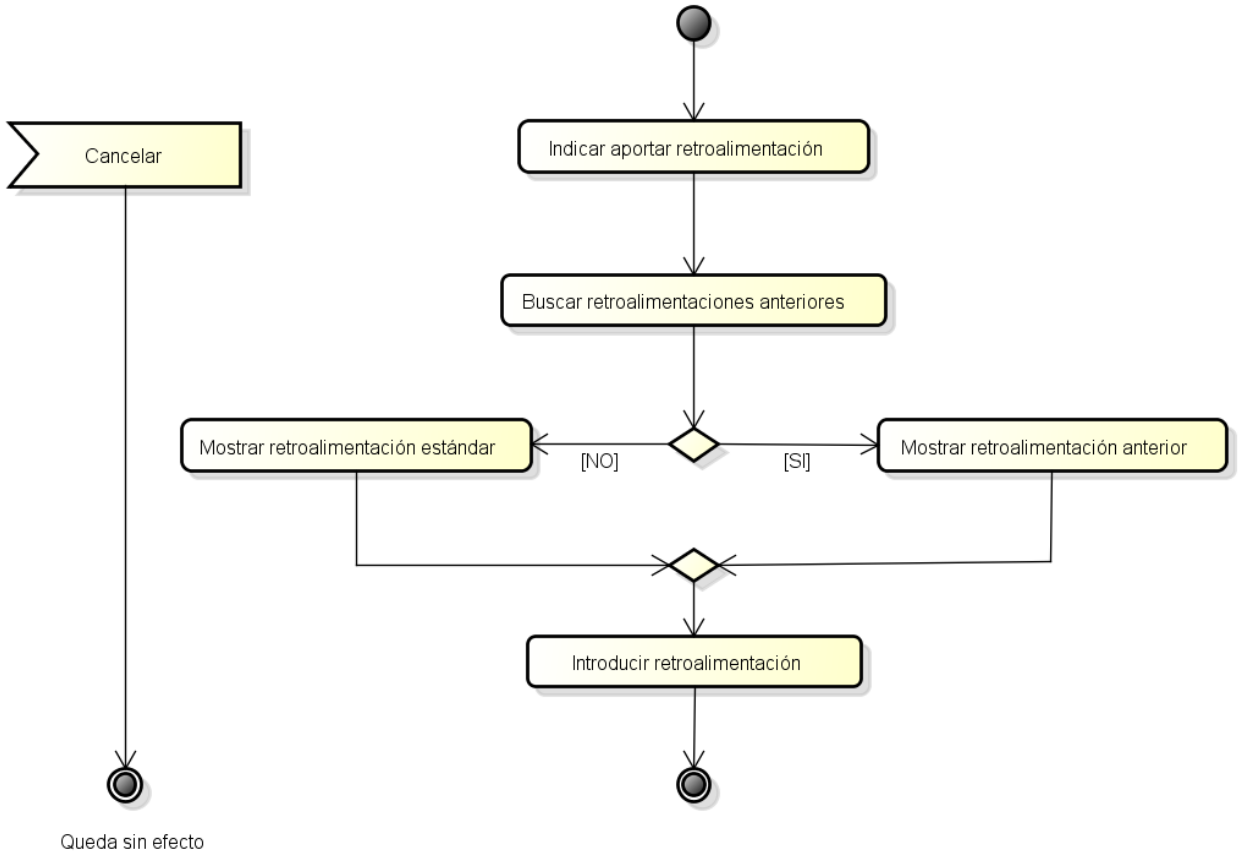


Figura 25. Diagrama de actividades. Aportar retroalimentación

4.6.5 Tablas y modelos

Este es el modelo y las tablas final de la aplicación:

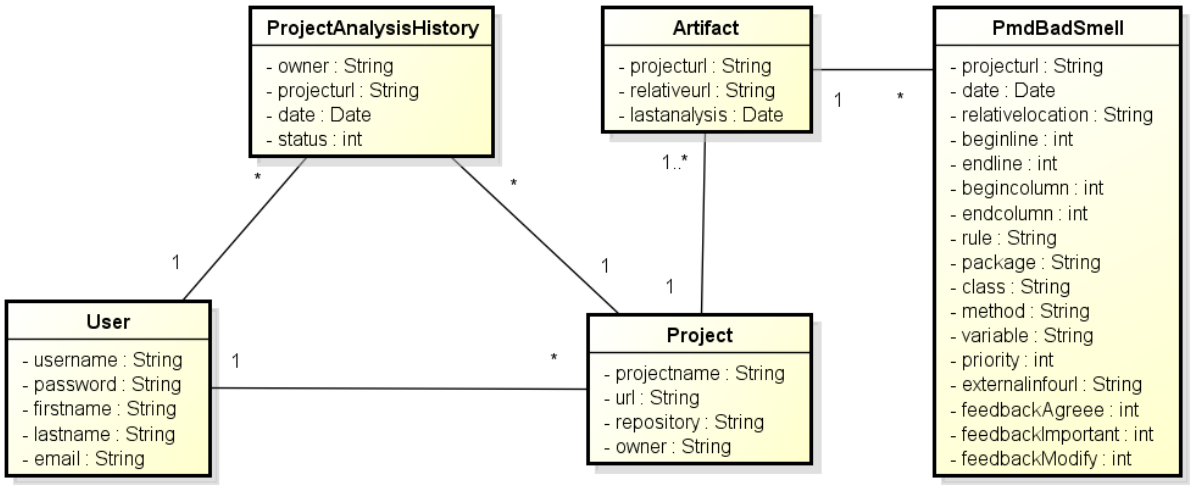


Figura 26. Diagrama final de clases y tablas

- **User:** Representa al modelo y a la tabla referente a los usuarios. Todos los campos son obligatorios, pero el más importante es *username,* ya que es único para cada usuario y es el que asocia los proyectos y análisis al usuario.

- **ProjectAnalysisHistory:** Representa un historial de análisis en el que se asocia a un usuario con un proyecto. Además, guarda la fecha en que se realiza el análisis y en qué estado se encuentra (En proceso o Completado).

- **Project:** Representa al modelo y a la tabla referente a los proyectos. Todos los campos son obligatorios. Guarda el nombre del proyecto, la url del repositorio en el que está guardado, que repositorio es y a quien pertenece.

- **Artifact:** Representa al modelo y la tabla de los artefactos que están en el repositorio, normalmente ficheros. Guarda la referencia de este artefacto dentro del proyecto.

- **PmdBadSmellReport:** Representa al modelo y la tabla de las detecciones de PMD. Además de recoger todos los datos que la herramienta aporta, guarda la retroalimentación de los usuarios con respecto a esa detección.

4.6.6 Funcionamiento de la parte computacional

En la sección 4.6.3 se ha hablado acerca de la estructura de la aplicación. En el directorio tools se encuentran todas las herramientas referidas a la parte computacional. A continuación, se habla acerca de esta parte.

En cada análisis que se lanza, la aplicación por debajo lanza un script PHP en segundo plano con la instrucción shell_exec. Este script es el que se ha denominado en la sección 4.6.4 con el nombre de TaskManager.php. Este script está prediseñado para que acepte parámetros, como son, la url del proyecto analizar, el usuario que lo lanza y las herramientas que van a analizar el proyecto. Tal y como se ejecuta en la aplicación:

```
sell_exec('<ubicacion de php> tools/TaskManager.php -user '.$this->userID.' -url
'.$this.projectUrl.' -options.' <opciones> &');
```

TaskManager se divide en tres partes: descarga del proyecto del repositorio, análisis con las herramientas de detección y almacenamiento de las detecciones en la base de datos. De la parte de descarga del proyecto se encarga la clase DownloadTool. Esta clase realiza un wget al repositorio y trae el proyecto a un directorio local de la aplicación. De la parte del análisis se encarga la clase AnalysisTool. Esta clase coge el proyecto de la ubicación en la que le ha dejado DownloadTool y lo analiza con las herramientas indicadas, en este caso final solo se ha podido realizar con PMD, aunque prácticamente se ha dado soporte a jDeodorant aunque no se ejecute. Los formatos de salida de las herramientas son diferentes:

- PMD: esta herramienta permite diferentes formas de salidas del análisis: XML, HTML, CSV, formato texto… entre otros. El elegido en este caso es XML, fácilmente de parsear. Ejemplo:

```
<?xml version="1.0"?><pmd>
<file name="/home/user/data/pmd/pmd/test-data/Unused1.java">
<violation line="5" rule="UnusedLocalVariable">
Avoid unused local variables such as 'fr'
</violation>
</file></pmd>
```

- jDeodorant: esta herramienta, al ser una modificación del plugin de eclipse, la salida no se da en un formato fácilmente parseable. JDeodorant aporta una salida con la siguiente estructura para cada directorio del proyecto:

```
Running JDeodorant on project <nombre del proyecto>
        Move Method Refactoring Opportunities: (Feature Envy)
              <Source Entity> -> <Target>
              ...
        Type Check Elimination Refactoring Opportunities: (Type Chek)
              <Type Cheking Method>
              ...
        Extract Method Refactoring Opportunities: (Long Method)
        <Source Method> <Variable Criterion> <Block-Based> Duplicated/Extracted>
              ...
        Extract Class Refactoring Opportunities: (God Class)
        <Source Class> <detalles> <Extractable Concept>
              …
```

  De la última parte de almacenamiento, se encarga ReportTool. Esta herramienta parsea los resultados obtenidos por la clase anterior (AnalysisTool) y los almacena correctamente en la base de datos.

4.7 CALENDARIZACIÓN FINAL Y ANÁLISIS DE RIESGOS A PROYECTO VENCIDO

4.7.1 Calendarización final

La calendarización final se ha cumplido ya que es una metodología ágil, al final de cada iteración, el desarrollo de la aplicación queda hasta donde se haya llegado. El único problema que ha habido es un retraso en la entrega de la documentación para la primera revisión de dos días. Sin contar el problema de la iteración 3 y la falta de funcionalidad en esa iteración.

4.7.2 Análisis de riesgos a proyecto vencido

Aquí se concretan los riesgos que han aparecido y el efecto que ha provocado su aparición. Para esto, se mostrarán en una tabla, ordenados por magnitud, los identificadores de los riesgos, diferenciando los que se

han presentado y los que no, y detallando si el plan de contingencia que se había estimado para cada uno se ha ajustado a la previsión propuesta.

ID	Riesgo	Magnitud	Aparición	Plan de contingencia	Contingencia Real
RSK-06	No aprobar todas las asignaturas	5.2	No	Retrasar el fin de proyecto hasta la convocatoria de enero.	Ninguna
RSK-03	Suspender exámenes	2	No	Aumentar el número de horas semanales, en caso de no ser posible plantear el retraso de la entrega hasta la siguiente convocatoria.	Ninguna
RSK-01	Planificación optimista	1.6	Sí	Aumentar el número de horas semanales. Ajustar las funcionalidades implementadas en cada iteración.	Se aumentó el número de horas semanales, pero en alguna iteración quedó funcionalidad incompleta
RSK-02	Estimación de horas semanales no satisfechas	1.25	Sí	Recuperar fines de semana el tiempo no dedicado a lo largo de la semana.	Debido a ciertas tareas de clase y las prácticas externas no se pudo satisfacer el número de hora recuperando gran parte el fin de semana.
RSK-05	Reducción de la productividad por inexperiencia	0.99	Sí	Aumentar el número de horas semanales. Ajustar las funcionalidades implementadas en cada iteración.	Surgió un gran problema jDeodorant obligando prácticamente a desechar una iteración entera. La elección fue dejar documentados los problemas en esa iteración.
RSK-04	Falta de tiempo para tutorías	0.5	No	Aumentar la frecuencia de las tutorías. Utilizar medios virtuales y documentos compartidos para las tutorías.	Ninguna.

Tabla 32. Lista detallada de riesgos a proyecto vencido

5 CONCUSIONES

En este Trabajo Fin de Grado, se ha expuesto uno de los problemas que existen dentro del campo de la calidad del software, que son los *Design Smell*. Se ha podido comprobar que el campo de estudio y trabajo de los *Design Smells* es complejo y es necesaria una estandarización. Esta estandarización es necesaria para la incentivar la adopción de herramientas por parte de la industria del software que ayuden a la detección y refactorización de los *Design Smell*. Este proceso de estandarización no parece, de momento, sencillo. Se ha comentado en este trabajo sobre ello. El proceso de estandarización no parece sencillo debido a una diferencia de opiniones a la hora de evaluar ciertos *Design Smell*. Esta diferencia de opinión provoca cierta subjetividad en cuanto a la detección de *Design Smells* se refiere. Esta diferencia de opinión también puede ser la causa de que las distintas herramientas de detección de *Design Smells* tengan un grado de acuerdo pobre a la hora de evaluar y detectar *Design Smells* sobre el mismo código fuente.

A raíz de todos estos problemas, se ha intentado desarrollar una aplicación que recoja información de los análisis realizados por distintas herramientas, y recoja también la opinión o el grado de acuerdo que tienen los usuarios que lanzaron el análisis con los resultados obtenidos por el mismo. Esta aplicación será en un futuro, con los ajustes necesarios que en este proyecto no se han podido realizar, una herramienta que dé soporte a la investigación de *Design S*mells tanto por su uso, como por el trabajo de minería de datos que se pueda realizar sobre su base de datos.

El desarrollo de este proyecto ha sido todo un reto para mí. No conocía apenas nada sobre el tema central de este trabajo, los *Design Smell*. He aprendido mucho sobre calidad y diseño software en el transcurso de este trabajo, aunque quizá no lo haya sabido plasmar bien sobre el papel o sobre el mismo código. La ejecución de este trabajo parecía, a priori, más sencilla de lo que luego ha resultado ser. Me he encontrado con múltiples problemas, sobre todo en la parte computacional. He encontrado múltiples problemas con la ejecución en segundo plano y la adaptación al proyecto de las distintas herramientas.

El trabajo futuro a realizar sobre este proyecto sería conseguir añadir las 5 herramientas propuestas, con los repositorios públicos que se han comentado en el transcurso de la práctica, GitHub y SourceForge. Mejorar el aspecto computacional, ya que se ha observado que al añadir más herramientas el tiempo que tarda en análisis tarda considerablemente. Otra de los desarrollos futuros sería añadir un soporte multilenguaje que facilite el uso por un conjunto global de usuarios, puesto que este trabajo está completamente en castellano.

6 BIBLIOGRAFÍA

[1] AlKharabsheh, K., Crespo, Y., Manso, E., Taboada, J.A.: Sobre el grado de acuerdo entre evaluadores en detección de Design Smells. Aceptado para publicación en Jornadas de Ingeniería del Software y Bases de Datos JISBD'2016, Salamanca 2016.

[2] AlKharabsheh, K., Crespo, Y., Manso, E., Taboada, J.A.: Comparación de herramientas de Detección de Design Smells. Aceptado para publicación en Jornadas de Ingeniería del Software y Bases de Datos JISBD'2016, Salamanca 2016.

[3] Mellor, S., Highsmith, J., and 15 others: The agile manifesto. http://www.agilemanifesto.org/, 2001. Última visita el 06/07/2016.

[4] Fowler, M.: Code Smell. (Febrero 2006) http://martinfowler.com/bliki/CodeSmell.html Ultima acceso: 08/07/2016

[5] Brown, W.J., Malveau, R.C., McCormick III, H.W., Mowbray, T.J.:AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. Jhon Wiley and Sons (March 1998), http://www.antipatterns.com

[6] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring Improving the Design of Existing Code. Object Technology Series, Addison-Wesley (June 1999).

[7] Riel, A.J.: Object Design Heuristics. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (April 1996).

[8] Wake, W.C.: Refactoring Workbook. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003).

[9] Patterson, D., Fox, A.: Engineering software as a service: An agile approach using cloud computing.

[10] Peralta, A.: Desarrollo SaaS con Ruby on Rails. Trabajo Fin de Grado. Universidad de Valladolid (Enero 2015).

[11] Pérez García, F.J.: Refactoring Planning for Design Smell Correction in Oject-Oriented Software. Tesis Doctoral. Universidad de Valladolid. http://www.giro.infor.uva.es/Publications/2011/Per11

[12] López Nozal, C.: Detección de defectos de diseño mediante métricas de código. Tesis Doctoral Universidad de Valladolid. http://uvadoc.uva.es/bitstream/10324/2719/1/TESIS281-130429.pdf

[13] Martin, R.C.: Agile Software Development: Principles, Patterns, and Practices. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.

[14] Mendo Alonso, A., Sobrino Fernandez, J., Pérez, J., Crespo, Y.: Evaluación de herramientas de detección y corrección de defectos de diseño. Tech. Rep. 2011/02, Grupo GIRO, Departamento de Informática, Universiad de Valaldolid (March 2011), http://www.giro.infor.uva.es/Publications/2011/MSPC11/

[15] jDeodorant Team: How i can execute JDeodorant in headless mode? http://users.en-cs.concordia.ca/~nikolaos/jdeodorant/index.php?option=com_content&view=article&id=87:how-can-i-execute-jdeodorant-in-batch-processing-mode&catid=27:installation-questions&Itemid=41. Último ac-ceso 28/06/2016.

[16] Ptidej Team: DECOR. http://www.ptidej.net/research/designsmells/. Último acceso 28/06/2016.

[17] SourceMaking: Smells. https://sourcemaking.com/refactoring/smells. Último acceso 27/06/2016.

[18] PMD. https://pmd.github.io/. Último acceso 28/06/2016.

[19] jDeodorant. https://www.jdeodorant.org. Último acceso 28/06/2016.

[20] Together. http://borland.com/es-ES/Products/Requirements-Management/Together. Últi-mo acceso 28/06/2016.

[21] iPlasma. http://loose.upt.ro/reengineering/research/iplasma. Último acceso 28/06/2016.

[22] SensioLabs: Twig. http://twig.sensiolabs.org. Último acceso 10/07/2016.

[23] Plantilla. Gentelella. https://github.com/puikinsh/gentelella. Último acceso 10/07/2016.

[24] DooPHP. http://doophp.com. Caducó el día 07/07/2016. Ubicación del proyecto en GitHub: https://github.com/darkredz/DooPHP. Último acceso 10/07/2016.

ANEXOS

A. MANUAL DE USUARIO

A.1 Registro

Para realizar un registro solo hay que introducir un nombre de usuario que no esté en uso, una contraseña, repetir la contraseña, un nombre, unos apellidos, un correo electrónico y confirmar el correo electrónico.

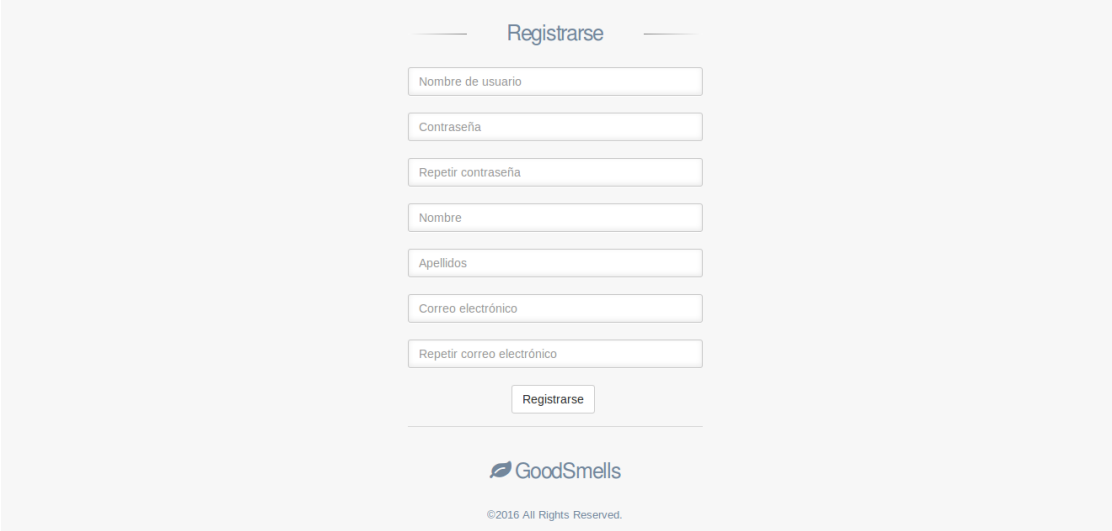


Figura 27. Registro de usuario

Una vez introducidos todos los datos pulsar registrarse y directamente se irá al panel de control. Nada más registrarse no se muestra ningún proyecto.

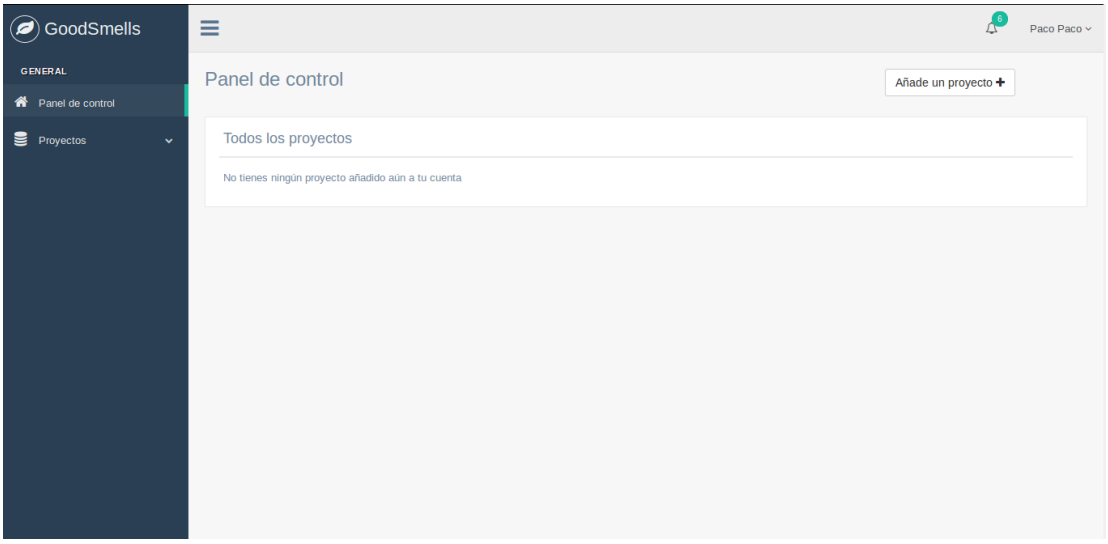


Figura 28. Panel de control vacío

A.2 Inicio de sesión

Para iniciar sesión solo hay que introducir el nombre de usuario y contraseña que fueron utilizados para el registro

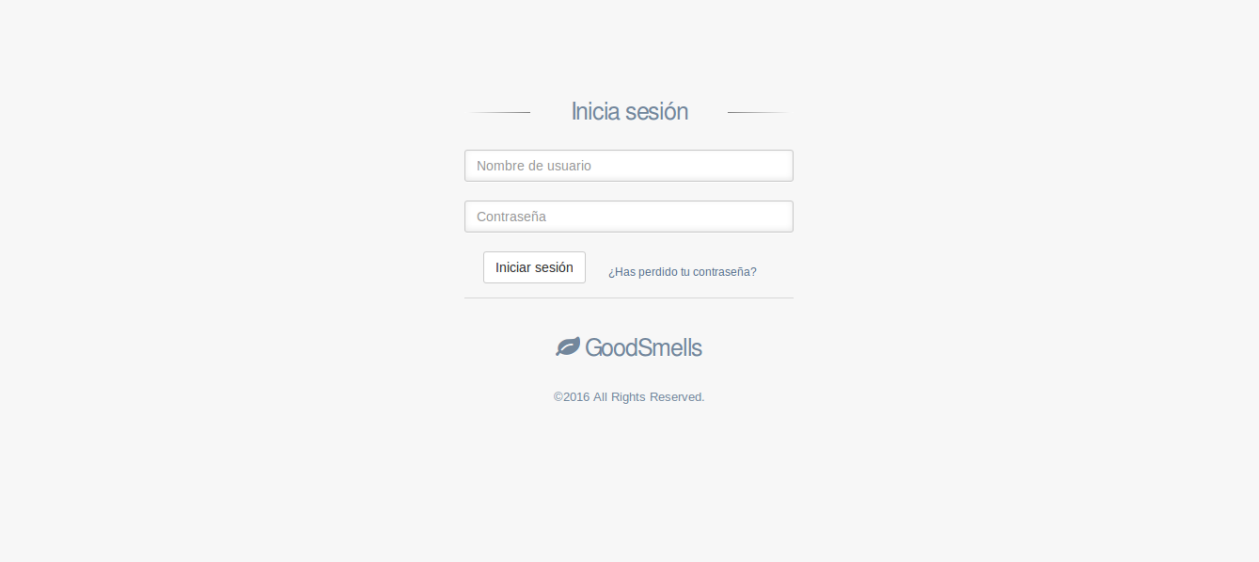


Figura 29. Inicio de sesión

Una vez introducidos los datos, se pulsa en iniciar sesión y se entra en el panel de control mostrando todos los proyectos que el usuario tiene guardados.

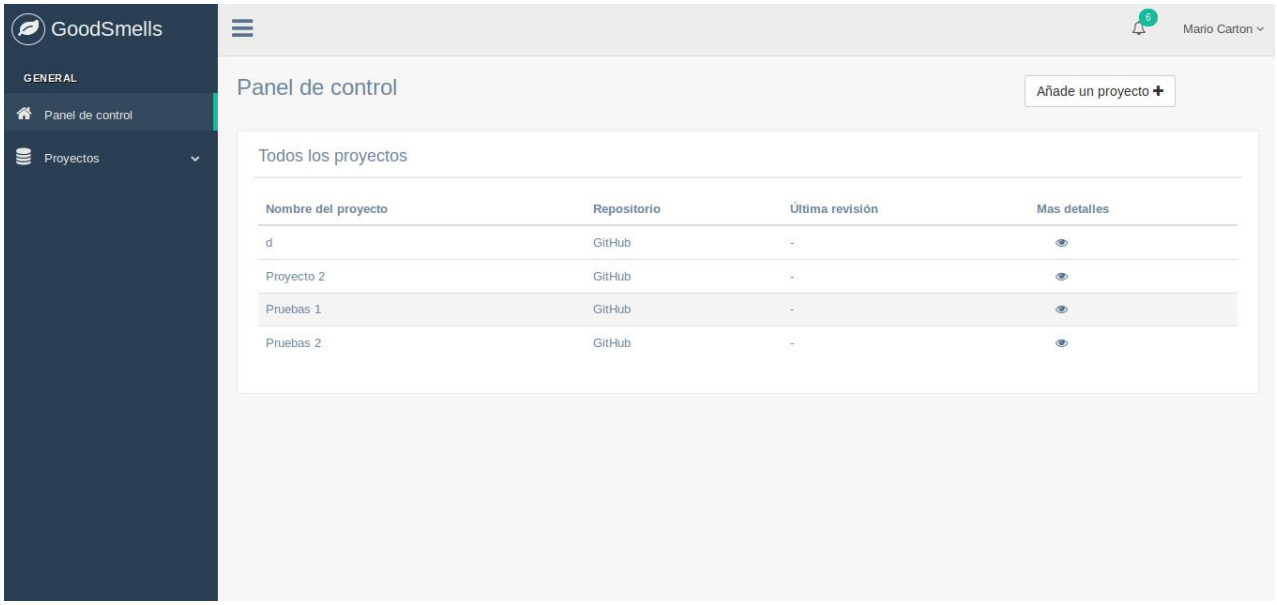


Figura 30. Resumen de proyectos

A.3 Añadir un repositorio

Para añadir un repositorio, hay que pulsar en el botón “añadir repositorio” que está situado en la parte superior del panel de control. Una vez pulsado aparece el siguiente cuadro de diálogo:

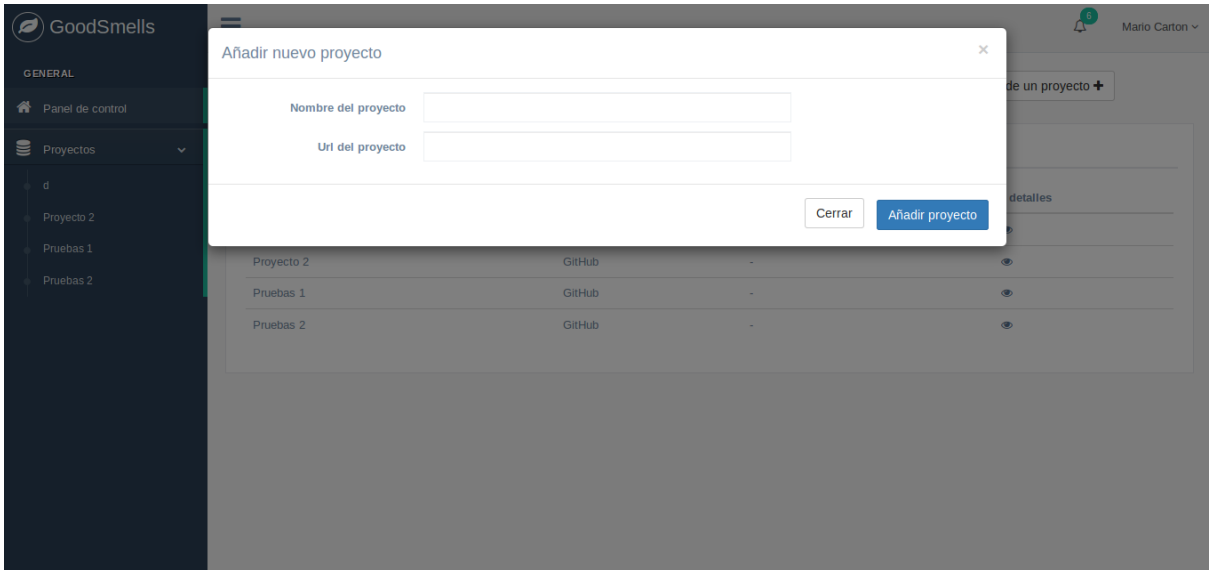


Figura 31. Añadir repositorio

Para añadir el proyecto GitHub, solo hay que introducir el nombre que se le da a dentro de la aplicación y el link al proyecto. IMPORTANTE: el link a GitHub debe ser la que tiene “.git”. Una vez introducidos estos datos, se pulsa el botón de “añadir proyecto” y el proyecto añadido se mostrará en el panel de control.

A.4 Lanzar un análisis

Desde el panel de control, se pulsa en el icono del ojo para ver más información sobre el proyecto. Se muestra una de las siguientes pantallas, dependiendo si se ha lanzado algún análisis o no:

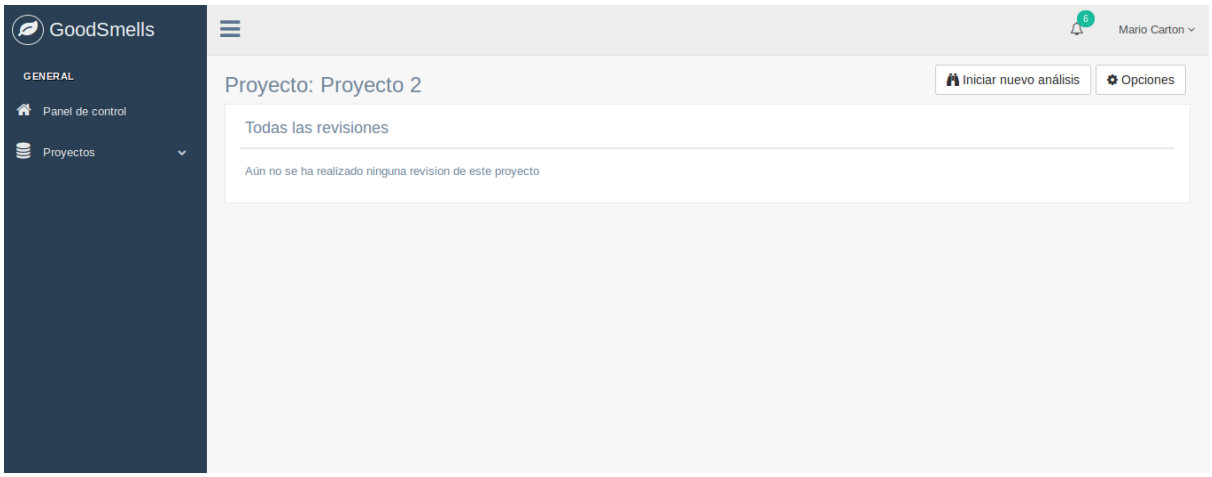


Figura 32. Resumen de un proyecto vacío

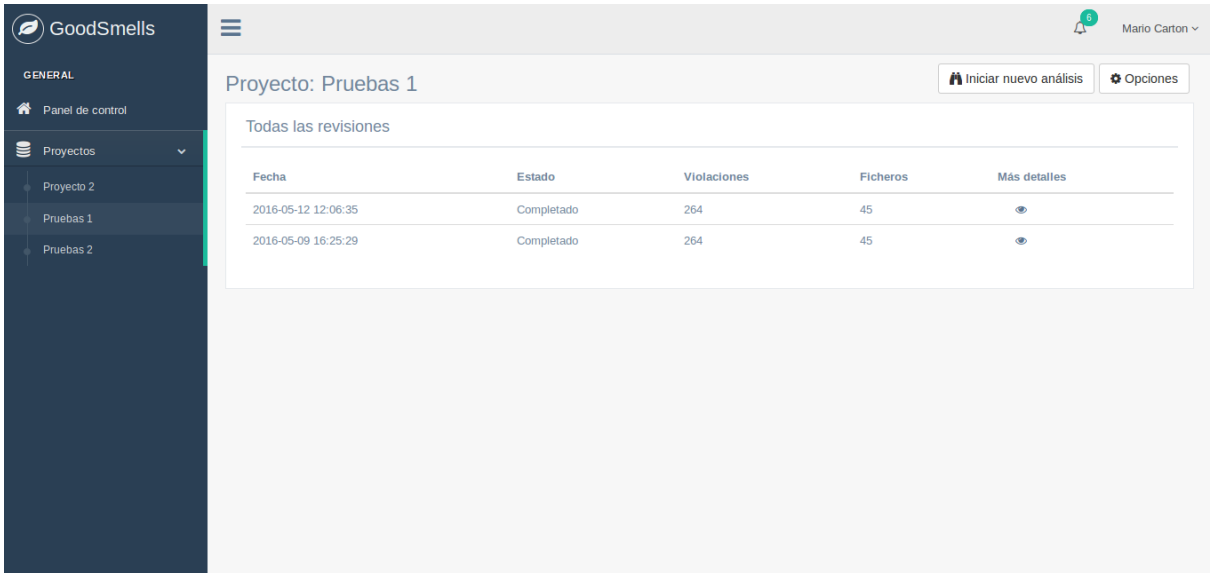


Figura 33. Resumen de un proyecto

Una vez estamos viendo los análisis lanzados (en el caso de que se haya lanzado alguno), se pulsa en el botón de lanzar análisis. Se muestra el siguiente cuadro de diálogo:

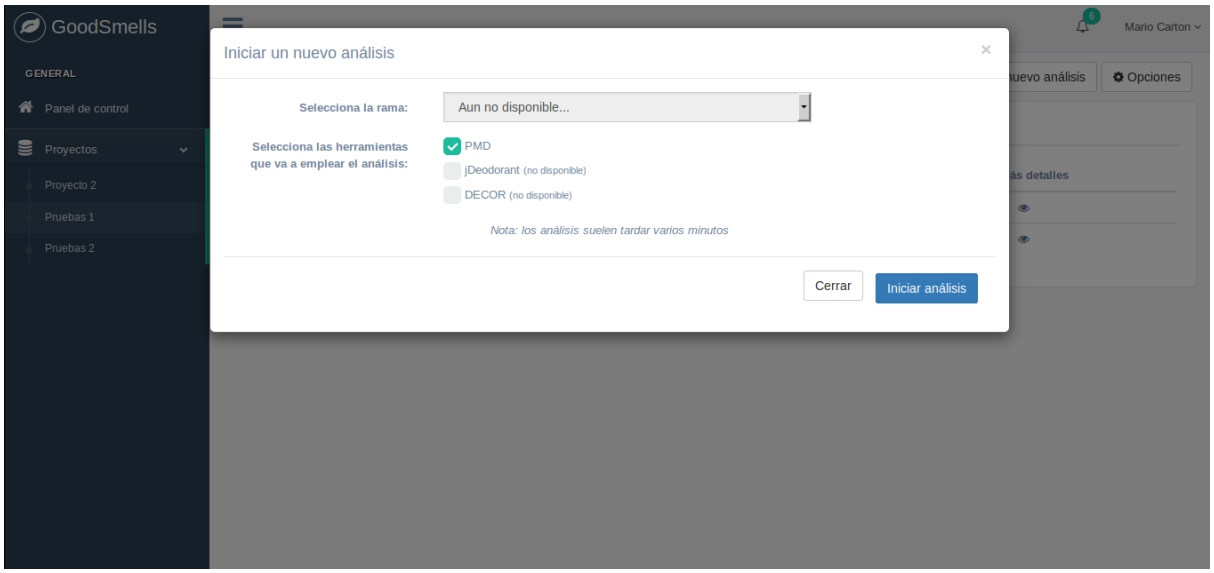


Figura 34. Lanzar análisis

Se marcan las opciones que se quieran realizar (en esta versión presentada solo se permite PMD) y el botón “lanzar análisis” para comenzar el análisis del proyecto.

A.5 Ver resultado de análisis y sus detalles

Una vez se ha lanzado un análisis, hay que esperar a que esté completo. Desde el panel de control se pulsa el icono del ojo para ver los análisis lanzados

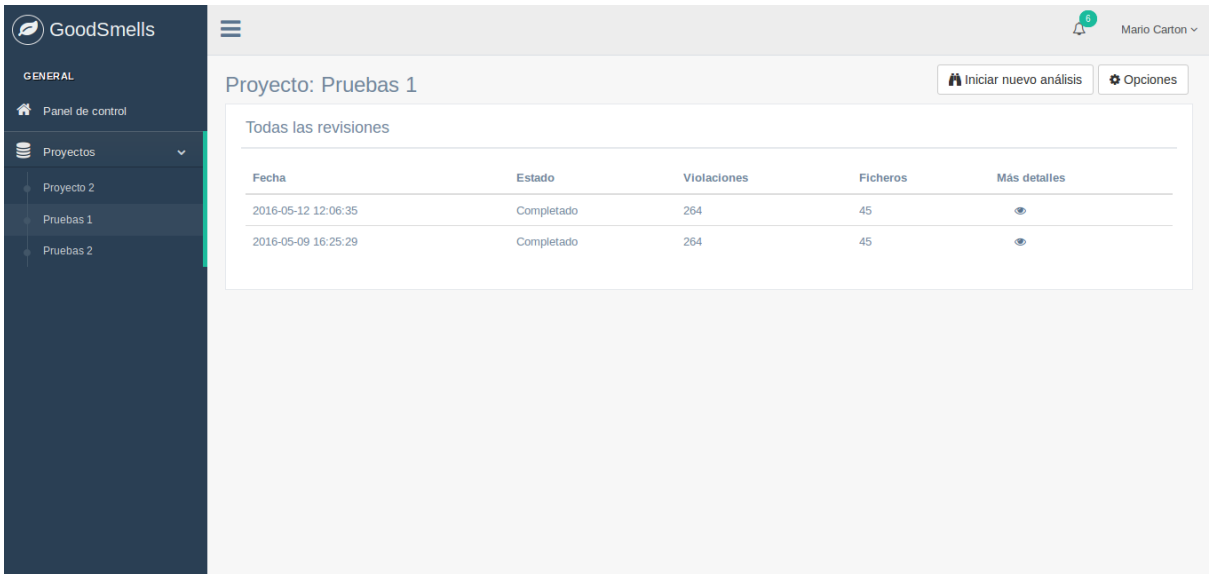


Figura 35. Resumen de un proyecto

Para ver los detalles de un proyecto completado, se pulsa en el icono del ojo para ver más detalles del análisis en concreto. Se muestra la siguiente pantalla:

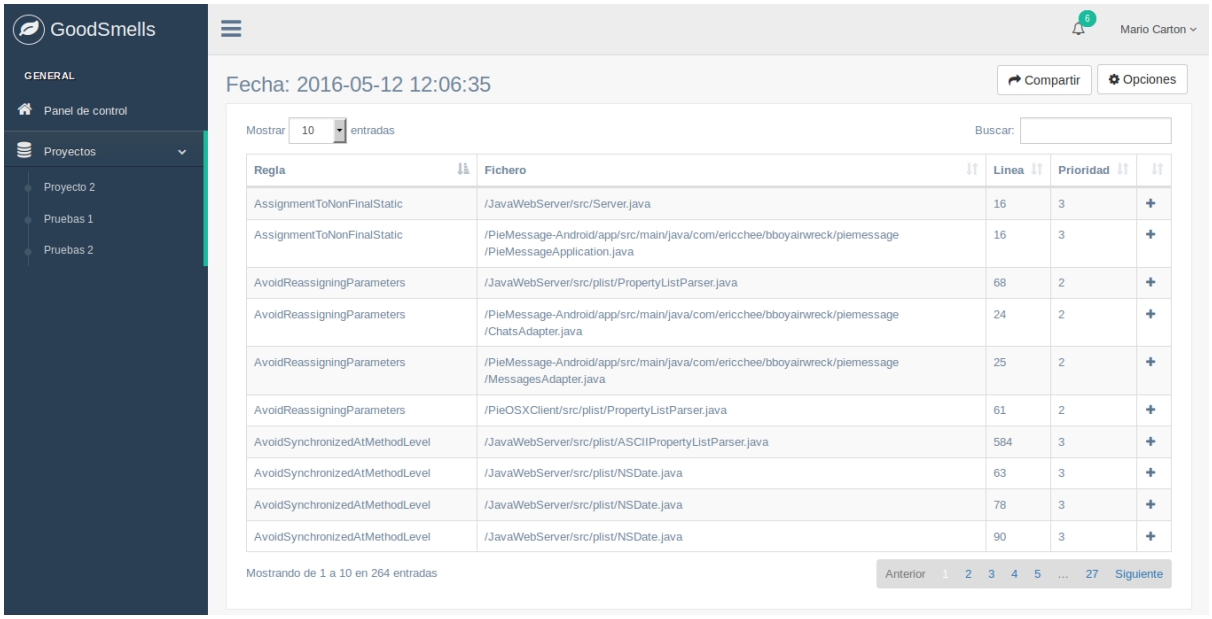


Figura 36. Resumen de análisis

En esta pantalla se muestran los resultados en una tabla. La tabla se puede ordenar alfabéticamente por columnas o se puede buscar por palabras en el cuadro de búsqueda. Para ver más detalles de una de las detecciones se pulsa el icono de un más y se muestra la siguiente pantalla:

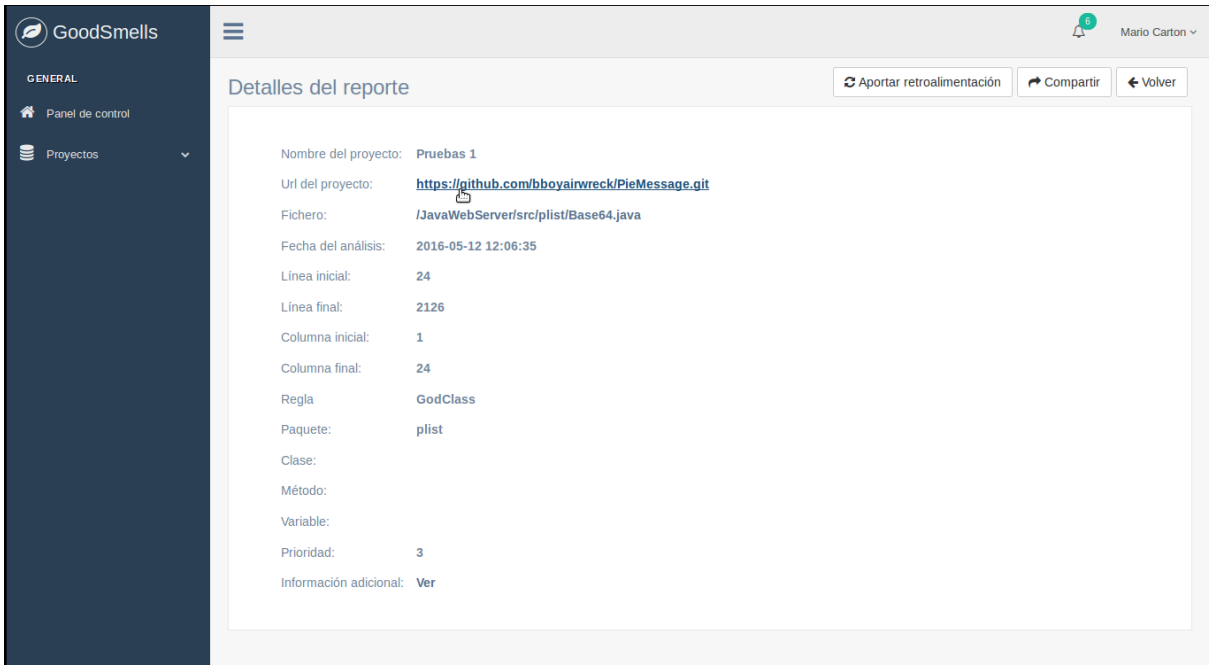


Figura 37. Detalles de una detección

Esta pantalla muestra todos los detalles de la detección. Si se pulsa en el enlace del fichero se puede ver código del fichero en el repositorio.

A.6 Aportar retroalimentación

Partiendo de la sección A.5, viendo los detalles de análisis. Se pulsa el botón “aportar retroalimentación” y se muestra el siguiente cuadro de dialogo: (Figura 38)

Se rellenan las preguntas y se pulsa aceptar para guardar las respuestas. Si se quiere modificar las respuestas solo hay que volver a realizar esta operación. Al intentar modificar las respuestas, aparece la respuesta anterior.

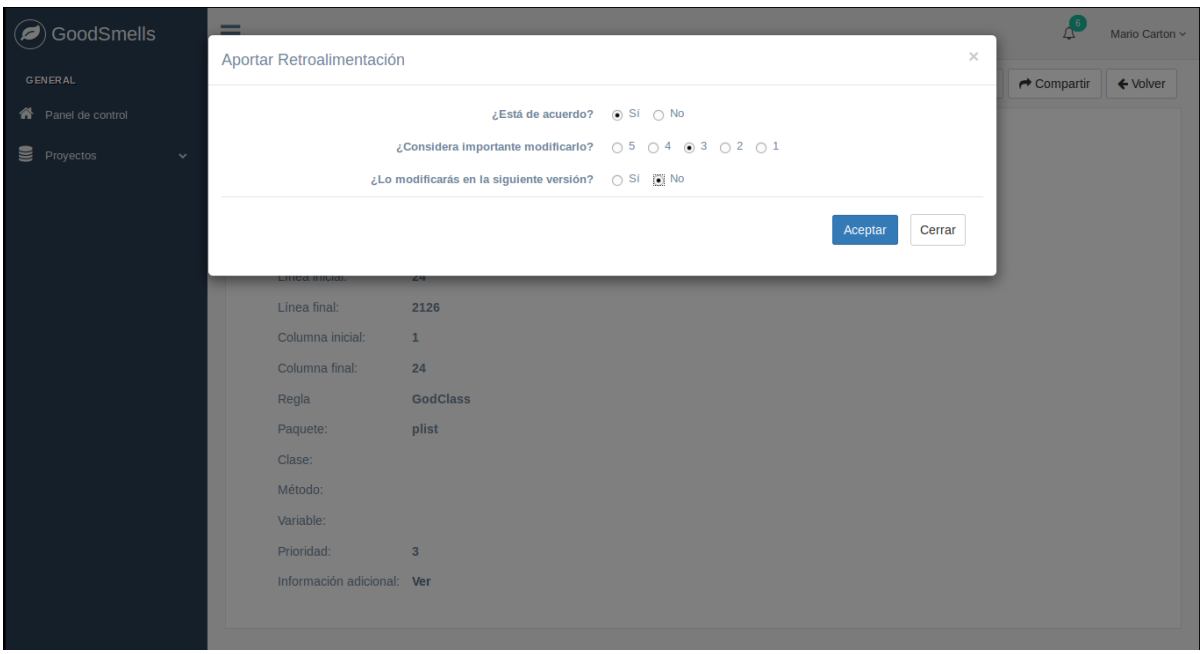


Figura 38. Aportar retroalimentación

B. MANUAL DE INSTALACIÓN

Este proyecto funciona en cualquier servidor que pueda ejecutar PHP5. En este caso se ha optado para el desarrollo un servidor independiente de la plataforma llamado LAMPP (de esta forma se llama para linux, el proyecto en sí se llama XAMPP). Se ha optado por este servidor porque integra todo lo que se necesita para la ejecución de este proyecto: Apache, PHP5 y MySQL. Para la gestión de la base de datos además trae phpmyadmin. A continuación, se explica cómo se ha realizado la instalación con LAMPP para linux, pero también se puede realizar con otros servidores que ejecuten PHP5 de forma similar.

Se descarga LAMPP de la página del proyecto (`https://www.apachefriends.org/index.html`) y se instala. Se inicia el servidor con el siguiente comando:

```
sudo /opt/lampp/lampp/ start
```

para parar el servidor:

```
sudo /opt/lampp/lampp/ stop
```

Se abre el navegador y se entra en *localhost* para comprobar que se ha iniciado correctamente. Ahora se entra en *localhost/phpmyadmin/*, por defecto suele venir sin contraseña. En este caso se ha optado por dejar así (es posible cambiar la contraseña si se desea). En *localhost/phpmyadmin/*, creamos una nueva base de datos con el nombre de *goodsmells* y se importa la base de datos con el script que se adjunta con el proyecto.

Una vez se tiene la base de datos, se copia el directorio del proyecto a /opt/lampp/htdocs/. Entramos en *localhost/goodsmells/* para comprobar que todo funciona bien.

La base de datos se proporciona vacía, así que, para empezar a usar la aplicación, primeramente, habrá que registrar a un usuario.

C. MANUAL DE MANTENIMIENTO Y EXTENSIBILIDAD

C.1 Estructura del proyecto

Lo primero que se necesita conocer para el mantenimiento y la extensibilidad es la estructura y organización del proyecto. Como ya se ha comentado en la sección 5.2 la tecnología empleada es un framework php, llamado DooPHP con Twig. La estructura del proyecto es la siguiente:

- **dooframework:** es el directorio que contiene el propio framework. No es necesario editar nada dentro de este directorio para cualquier acción de mantenimiento o extensibilidad.

- **global:** en este directorio se encuentran las imágenes, css, javascript y algún otro recurso de las vistas. En este directorio también se ha añadido dos directorios más, que son las plantillas originales utilizadas para el desarrollo del proyecto (indexTemplate, la plantilla de la página principal, y originalTemplate, la plantilla utilizada, en general, para todo el proyecto) por si son necesarias para alguna consulta para añadir alguna parte de la vista o añadir parte de la funcionalidad que estas aportan.

- **protected:** este es el directorio sobre el que se trabaja la mayoría del tiempo. Contiene los siguientes directorios relevantes dentro, aquellos no comentados aquí, no se han tocado.
    - **config:** contiene los ficheros de configuración y enrutamiento.
        - **common.conf.php:** fichero de configuración general del framework.
        - **db.conf.php:** fichero que contiene la configuración para el acceso a la base de datos. Se utiliza para indicar la base de datos que se emplea, dirección, nombre contraseña...
        - **routes.conf.php:** fichero de enrutamiento de las páginas o vistas de la aplicación. Aquí se indica que controlador se lanza en función de qué dirección introduzca el usuario.
        - **settings.php:** fichero en el que se encuentran algunas variables interesantes para el desarrollo de la aplicación.
    - **controller:** directorio que contiene los controladores de las vistas.
    - **model:** directorio que contiene los modelos de la aplicación. Estos modelos son especiales ya que utilizan una estructura propia del framework para el acceso a datos.
    - **view:** directorio donde se encuentran las vistas de la aplicación. Estas vistas también son especiales, usan Twig. En el directorio "base" se encuentran las vistas de las cuales heredan el resto de las vistas que están fuera de ese subdirectorio.
    - **viewc:** directorio en el que se encuentra alojado Twig. twig.php se encarga de integrar Twig en DooPHP.

- **tools:** aquí se encuentran las herramientas que utiliza la aplicación referente a la parte de computación. Aquí también se encuentra todo el código encargado de la descarga, análisis y posterior almacenamiento de los análisis en la base de datos.
    - **analysis:** en este directorio se encuentran las herramientas de análisis. Cuando se quiera añadir alguna herramienta más se debe alojar aquí.
    - **downloads:** directorio temporal donde se alojan los proyectos de los usuarios durante un análisis. Dentro de este directorio, hay subdirectorios para cada usuario.
    - **shellscripts:** este directorio contiene los scripts que lanzan las herramientas alojadas en el directorio analysis.
    - **TaskLauncher.php:** esta clase es la encargada de lanzar el análisis del proyecto. Esta clase es llamada desde el controlador DashboardController.php en el método initanalysis(). Esta clase

prepara un comando, para la ejecución de un script, en función de las herramientas seleccionadas para lanzarlo en segundo plano.

- **TaskManager.php:** este es el script que lanza TaskLauncher.php. Este script filtra las opciones que le pasa TaskManager.php y en función de las opciones realiza tres tareas: descarga, análisis y almacenamiento de resultados.
- **DownloadTool.php:** clase encargada de descargar el proyecto. De momento esta clase solo descarga el proyecto del master de la aplicación, pero en trabajo futuro también se tendría que encargar de ver la versión en la que se encuentra y almacenarla para en un futuro ver el estado del código en el momento del análisis.
- **AnalysisTool.php:** clase que se encarga de lanzar los scripts de las herramientas almacenados en el directorio shellscripts.
- **ReportTool.php:** clase que se encarga de guardar en la base de datos los resultados de los análisis.

- **index.php:** es el fichero de arranque del framework (no es necesario editarlo).

C.2 Añadir un nuevo repositorio

Esta tarea de extensibilidad es relativamente sencilla, porque apenas hay que realizar modificaciones en la interfaz. En este caso solo habría que modificar el fichero DownloadTool.php que es el encargado de la descarga de los proyectos. Simplemente habría que añadir un nuevo método dentro de ese fichero que se encargue de la descarga del proyecto. Una vez descargado, no importa de qué repositorio sea, porque la ejecución de las herramientas de análisis se realiza localmente.

C.3 Añadir una nueva herramienta de análisis

A diferencia de la sección C.2, esta tarea de extensibilidad es mucho más compleja, ya que implica tanto cambios en la parte computacional, como cambios en la parte de interfaz de usuario. Es recomendable ir mirando el código mientras se lee esta sección, debido que algunas explicaciones pueden ser un poco liosas.

Lo primero que se debe asegurar para añadir una herramienta es que se pueda ejecutar por terminal. Si no es ejecutable mediante terminal, hay que buscar algún método de transformación para que pueda ser ejecutable por línea de comandos, como pasa, por ejemplo, en el caso de jdeodorant. Una vez asegurado que funciona la ejecución por terminal, se almacena la herramienta el directorio tools/analysis y se crea un script shell que lo ejecute en el directorio tools/shellscripts. Por ejemplo, el shellscript de la herramienta PMD (partiendo que nos encontramos en el directorio sellscripts):

```
cd ..

cd analysis/pmd/bin/

./run.sh pmd -d ../../../../$1 -f xml -R java-design
```

Una vez se ha creado el shellscript y se ha asegurado su funcionamiento, el siguiente paso es añadir una nueva función en el fichero AnalysisTool.php en la que se ejecuta el comando, por ejemplo:

```
shell_exec('cd tools/sellscripts && ./pmdtool.sh '.$dir);
```

el resultado del comando se guarda en una variable, que es lo que se devuelve a TaskManager.php (habia invocado anteriormente a AnalysisTool.php). A contiuación, TaskManager delega en ReportTool para guardar el resultado del análisis en la base de datos. La forma de guardar los datos en la base de datos depende de la herramienta, simplemente hay que crear una tabla en la base de datos que obligatoriamente tenga los atributos

projecturl, date, relativelocation, y para el feedback, feedbackagree, feedbackimportant y feedbackmodify, el resto de la tabla se crea en función de la necesidad de la herramienta.

Una vez hecho esto, en TaskManager hay que filtrar las opciones, es decir, añadir la opción para que, en función de los parámetros que pase TaskLauncher, se ejecute el análisis con la nueva herramienta o no. En TaskLauncher hay que añadir una nueva función que permita añadir el parámetro de la nueva herramienta que se le pasa a TaskManager, como tiene por ejemplo PMD.

Una vez hecho esto se pasa a la parte de la vista y el controlador. Hay que añadir en la vista viewproject.html la opción de la nueva herramienta en el formulario. En el controlador y método referente a esa vista y acción de submit, DashboardController->initanalysis(), añadir también la opción de si se ha marcado la opción de la herramienta añadirlo a TaskLauncher.

Hasta aquí la parte de añadir la herramienta, ahora solo hay que editar las secciones visuales que se encargan de mostrar los datos.

C.4 Otros apuntes acerca del mantenimiento

La base de datos está por defecto sin contraseña. Si se quiere poner contraseña, solo hay que entrar en localhost/phpmyadmin y ponerle la contraseña. Una vez esto, modificar en db.conf.php (ubicación descrita en el apartado C.1) la siguiente línea:

```
$dbconfig['dev'] = array ('localhost','goodsmells', 'root', '<contraseña>', 'mysql', true);
```

Si la aplicación no se va a ejecutar sobre el sevidor lampp, para que funcione la ejecucion en segundo plano hay que modificar la siguiente linea (liena 34) de TaskLauncher.php para especificar la ubicación en la que se encuentra php:

```
sell_exec('<ubicacion de php> tools/TaskManager.php -user '.$this->userID.' -url '.$this.projectUrl.' -options.' &');
```

Por si acaso hay que revisar también las rutas de algunos ficheros que se hayan podido escapar a este apunte. Puede ser una de las causas por las que el funcionamiento no sea el correcto.

D. CONTENIDO DEL CD

db: directorio que contiene el script de la base de datos: *goodsmells.sql*

doc: directorio que contiene la memoria del TFG: *memoria.pdf*

src: directorio que contiene el código fuente de la aplicación. Este código fuente es el que hay que meter en el servidor.