Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DEL SOFTWARE

Smellswisdom: una aplicación web colaborativa para compartir y mejorar el conocimiento sobre Design Smells

---

Alumno: Gino Jesús Quintana Angulo

Tutor: Yania Crespo González Carvajal

Dedico este trabajo a mis padres, por su apoyo incondicional y valores que me han inculcado para salir adelante en todo momento, por sus consejos, por la perseverancia y constancia que les caracterizan y que me han infundido siempre.

Agradecimientos

A mi familia, por haber sido el pilar fundamental para lograr los objetivos propuestos.

A todos los docentes de la carrera de Ingeniería Informática de la Escuela de Ingeniería Informática de Valladolid, en especial a mi tutora de trabajo de fin de grado, Yania Crespo González-Carvajal, cuyos conocimientos y gran trayectoria me han guiado en este trabajo.

A mis compañeros de carrera y amigos, que de una u otra manera se involucraron en este proyecto y me ayudaron de una manera desinteresada.

Por último, agradecer a todos los que contribuyeron con un granito de arena para terminar con éxito la meta propuesta.

Resumen

El objetivo de este proyecto es el desarrollo de una aplicación web basada en un estudio denominado “Software Design Smell Detection: a Systematic Mapping Study” [1], cuyo propósito es hacer una revisión sistemática de la literatura de un conjunto de 395 artículos, analizando todos los términos referentes a la detección de Design Smells e incluirlos en el estudio. Tras la obtención del conocimiento y plasmarlo en un modelo de dominio conceptual, se pretende proporcionar a la comunidad el acceso al conocimiento organizado sobre la detección de los ya mencionados defectos en el diseño y permitir a los usuarios colaborar aportando nuevos datos al conocimiento obtenido a través de la web colaborativa, que tiene como base un modelo lógico de datos relacional basado en este modelo conceptual.

Smellwisdom es una aplicación para visualizar la información sobre la detección de Design Smells tanto en texto plano como en gráficas, así como poder añadir nuevos datos al conocimiento y está enfocada principalmente a profesionales e investigadores del desarrollo software. Se trata de un proyecto desarrollado en Java, utilizando ZK framework, Spring e Hibernate(JPA). Siguiendo una metodología ágil basada en SCRUM adaptada para este contexto de trabajo de fin de grado.

Abstract

The objective of this project is the development of a web application based on a study called “Software Design Smell Detection: a systematic mapping study” [1], whose purpose is to make a systematic review of the literature of a set of 395 articles, analyzing all terms referring to the detection of Design Smells and include them in the study. After obtaining the knowledge and translating it into a conceptual domain model, it’s intended to provide the community with access to organized knowledge on the detection of Design Smells and allow users to collaborate by providing new data to the knowledge obtained through the collaborative website. The website is based on a logical model of relational data based on this conceptual model.

Smellwisdom is an application to visualize the information about the detection of Design Smells in both plain text and graphics, as well as to be able to add new data to the knowledge and is mainly focused on professionals and researchers of software development. This is a project developed in Java, using ZK framework, Spring and Hibernate (JPA). Following an agile methodology based on SCRUM adapted for this context of Final Degree Project.

Índice general

Agradecimientos **III**

Resumen **V**

Abstract **VII**

Lista de figuras **XV**

Lista de tablas **XIX**

1. Introducción **1**

1.1. Introducción 1

1.1.1. Objetivos 1

1.1.2. Contexto y Motivación 2

2. Antecedentes y explicación del dominio **3**

2.1. Design Smell 3

2.2. Revisiones sistemáticas y mapeos sistemáticos de la literatura 5

2.2.1. Revisión Sistemática de la literatura 5

2.2.2. Mapeo sistemático de la literatura 6

2.3. Antecedentes 6

2.4. El dominio del conocimiento sobre Design Smells 7

3. Planificación **11**

3.1. Planificación 11

3.1.1. Roles y responsabilidades 12

3.1.2. Historias de usuario 12

3.1.3. Product Backlog 13

3.2. Gestión de riesgos 14

3.3. Calendarización 17

3.4. Análisis de coste inicial 18

3.4.1. Coste hardware y software 18

3.4.2. Coste recursos humanos 19

3.4.3. Coste total inicial 19

4. Herramientas y Tecnologías **21**

4.1. Herramientas y Tecnologías 21

4.1.1. Herramientas de planificación y gestión 21

4.1.2. Herramientas de Diseño y Desarrollo 23

4.1.3. Backend y Frontend 26

4.1.4. Pruebas 30

5. Seguimiento **31**

5.1. Sprint 1 31

5.1.1. Sprint planning 31

5.1.2. Sprint backlog 32

5.1.3. Sprint review y Sprint retrospective 32

5.2. Sprint 2 33

5.2.1. Sprint Planning 33

5.2.2. Sprint backlog 33

5.2.3. Sprint review y Sprint retrospective 34

5.3. Sprint 3 34

5.3.1. Sprint Planning 34

5.3.2. Spring backlog 35

5.3.3. Sprint review y Sprint retrospective 35

5.4. Sprint 4 36

5.4.1. Sprint Planning 36

5.4.2. Spring backlog 36

5.4.3. Sprint review y Sprint retrospective 37

5.5. Sprint 5 37

5.5.1. Sprint Planning 37

5.5.2. Spring backlog 38

5.5.3. Sprint review y Sprint retrospective 38

5.6. Sprint 6 39

5.6.1. Sprint Planning 39
5.6.2. Spring backlog 39
5.6.3. Sprint review y Sprint retrospective 40
5.7. Sprint 7 40
5.7.1. Sprint Planning 40
5.7.2. Spring backlog 41
5.7.3. Sprint review y Sprint retrospective 41
5.8. Sprint 8 42
5.8.1. Sprint Planning 42
5.8.2. Spring backlog 42
5.8.3. Sprint review y Sprint retrospective 42
5.9. Resultados 43
5.9.1. Costes reales 43
5.9.2. Product backlog final 45

6. Diseño **47**

6.1. Modelo conceptual 47
6.2. Diseño lógico 48
6.2.1. Detección de Design Smells 48
6.2.2. Usuarios 57
6.2.3. Conclusión 58
6.3. Arquitectura 59
6.3.1. Visión general 59

6.3.2. ZK Framework 61
6.3.3. Spring Framework 62
6.3.4. JPA (Hibernate) 63
6.3.5. Patrón MVC 64
6.3.6. Patrón DAO 65
6.4. Descomposición modular y dependencias 66
6.5. Modelos Dinámicos 67
6.6. Diagrama de despliegue 71
6.7. Diseño interfaz Gráfica 71
6.8. Diseño de historias de usuario 75
6.8.1. Identificación de usuario 75
6.8.2. Registro de usuario 76
6.8.3. Cerrar sesión 76
6.8.4. Consultar conocimiento sobre la detección de Design smells 77
6.8.5. Visualizar estadísticas sobre la detección de Design smells. 79
6.8.6. Contribuir añadiendo conocimiento sobre la detección de Design smells. 81

7. Implementación y pruebas **83**

7.1. Construcción del entorno de desarrollo 83
7.2. Usando el patrón método factoría 85
7.3. Pruebas 85

8. Conclusiones y Trabajo futuro **87**

8.1. Conclusiones 87
8.2. Líneas de trabajo futuro 88

A. Manual de mantenimiento **91**

A.1. Requisitos 91

A.2. Ejecutar la aplicación 94

B. Manual de despliegue **97**

B.1. Máquina desarrollo 97

B.2. Heroku 98

B.3. Servidor base de datos 99

C. Manual de usuario **101**

D. Contenidos del CD-ROM **109**

Bibliografía **111**

Índice de figuras

2.1. Modelo conceptual del conocimiento sobre la detección de Design smell. Tomada de [1]. 9

2.2. Enumerados obtenidos durante la extracción de datos. Tomado de [1]. 10

6.1. Modelo conceptual para usuarios. 47

6.2. Modelo lógico del conocimiento sobre la detección de Design smell. 49

6.3. Modelo lógico para usuarios. 57

6.4. Modelo arquitectónico. Tomado de Wikipedia [55]. 59

6.5. Esquema arquitectura en 3 capas de la aplicación. 59

6.6. Pila de aplicaciones de Smellwisdom. Tomado de zkoss [56]. 60

6.7. Arquitectura de ZK. Tomado de zkoss [57] 61

6.8. Arquitectura de Spring. Tomado de docs.spring.io [58] 62

6.9. Arquitectura de JPA/Hibernate. Tomado de docs.jboss.org [59] 63

6.10. Patrón MVC. Tomado de zkoss [60] 64

6.11. Patrón DAO, de rinconprogramadorcito.blogspot [61] 65

6.12. Estructura de paquetes. 66

6.13. Diagrama de secuencia CU “Registro de usuario”. 67

6.14. Subdiagrama “Validar registro”. 68

6.15. Subdiagrama “Guardar usuario”. 68

6.16. Diagrama secuencia búsqueda de artículos. 69

6.17. Subdiagrama búsqueda por caso. 70

6.18. Modelo de despliegue. 71

6.19. Boceto Login. 72

6.20. Boceto Registro. 72

6.21. Boceto Búsqueda. 73

6.22. Boceto Estadísticas. 74

6.23. Boceto Nueva contribución. 74

7.1. Estructura directorios. 84

7.2. Patrón método factoría. 85

8.1. Icebox de Pivotal Tracker. 89

A.1. ZK Studio en Eclipse Marketplace. 92

A.2. Importar proyecto Maven. 93

A.3. Añadir carpeta de recursos al validador de JPA. 94

A.4. Pestaña Main. 95

A.5. Pestaña Arguments 95

A.6. Pestaña Environment. 96

B.1. Añadir variable de entorno en Heroku. 99

C.1. Página identificación de usuario. 101

C.2. Página modal de registro. 102

C.3. Página búsqueda artículos. 102

C.4. Página búsqueda artículos, con un artículo seleccionado. 103

C.5. Página detalle general. 104

C.6. Página detalle smells. 104

C.7. Página detalle evidencias. 105

C.8. Página detalle herramientas. 106

C.9. Página estadísticas publicación. 106

C.10.Página nueva contribución. 107

C.11.Página modal autoría. 107

C.12.Página nueva contribución añadir smells. 108

Índice de tablas

2.1. Ejemplos de Bad Smells. 4

2.2. Ejemplos de Disharmonies. 5

2.3. Ejemplos de Antipatterns. 5

3.1. Estimación en horas por iteración. 11

3.2. Icebox 13

3.3. Product Backlog 14

3.4. Matriz de riesgos. 15

3.5. Riesgo-01 15

3.6. Riesgo-02 16

3.7. Riesgo-03 16

3.8. Riesgo-04 16

3.9. Riesgo-05 16

3.10. Riesgo-06 16

3.11. Riesgo-07 17

3.12. Estimación de entrega por iteración. 17

3.13. Calendarización. 18

5.1. Tareas Sprint 1 32

5.2. Tareas Sprint 2 33

5.3. Tareas Sprint 3 35

5.4. Tareas Sprint 4 36

5.5. Tareas Sprint 5 38

5.6. Tareas Sprint 6 39

5.7. Tareas Sprint 7 41

5.8. Tareas Sprint 8 42

5.9. Horas estimadas vs horas reales. 43

5.10. Costes hardware. 43

5.11. Product backlog final. 45

6.1. El usuario podrá identificarse en la aplicación. 75

6.2. El usuario podrá registrarse en la aplicación. 76

6.3. El usuario podrá cerrar sesión en la aplicación. 76

6.4. El usuario podrá buscar artículos en la aplicación. 77

6.5. El usuario podrá el detalle un artículo (publicación, autoría y caracterización). 77

6.6. El usuario podrá ver el detalle un artículo (smells, evidencias y herramientas). 78

6.7. El usuario podrá ver el detalle un artículo (proyectos) 78

6.8. El usuario podrá visualizar estadísticas sobre publicaciones. 79

6.9. El usuario podrá visualizar estadísticas sobre Design smells. 79

6.10. El usuario podrá visualizar estadísticas sobre enfoques. 79

6.11. El usuario podrá visualizar estadísticas sobre herramientas. 80

6.12. El usuario podrá visualizar estadísticas sobre evidencias de validación. 80

6.13. El usuario podrá visualizar estadísticas sobre factores de calidad. 80

6.14. El usuario podrá contribuir añadiendo un nuevo elemento (publicación,autoría y caracterización). 81

6.15. El usuario podrá contribuir añadiendo un nuevo elemento (smells, evidencias y herramientas). 82

Capítulo 1

Introducción

1.1. Introducción

1.1.1. Objetivos

El objetivo principal que se persigue con la realización del proyecto es desarrollar una aplicación web colaborativa en la que los usuarios puedan estar informados sobre la detección de Design Smells y a la vez contribuir para mejorar el conocimiento sobre este tema. Esta idea de trabajo de fin de grado ha sido escogida entre las ofertas propuestas por la tutora del trabajo.

Los objetivos que se persiguen para definir el alcance de la aplicación son los siguientes:

- Facilitar a los usuarios el acceso a información completa y actualizada acerca de la detección de Design Smells.
- Permitir a los usuarios colaborar, ya sea aportando comentarios, sugerencias de corrección, nuevos elementos o detalles que aún no se han cubierto sobre un elemento.

La aplicación tiene dos conjuntos de funcionalidades distintas, dependiendo del tipo de rol con el que se acceda a la aplicación. De esta manera se diferencian:

- ROL USER: accede a la información contenida en el servidor y colabora mejorando el acceso a la información.
- ROL ADMIN: posee idénticas funciones al ROL USER, a las que se añade, además, una de las funciones primordiales en la gestión de la aplicación, la capacidad de aprobar o no las distintas aportaciones de los usuarios colaboradores.

1.1.2. Contexto y Motivación

Los Design Smells son indicadores que afectan negativamente a la calidad del software, además de la comprensibilidad, la capacidad de prueba, la extensibilidad, la reutilización y la capacidad de mantenimiento en general. La calidad es muy importante en la ingeniería del software, por lo que es de interés para investigadores y profesionales, que ponen una gran atención en aplicar estrategias para paliar este problema.

La detección de estos elementos software que sufren de una programación deficiente, un mal diseño u otros problemas en el código fuente o el diseño, se le denomina Design Smell detection.

Comprender esta área de conocimiento es importante, debido a que hoy en día numerosos proyectos software tienen grandes dimensiones, de manera que la detección manual no es realista. Generalmente la detección ocurre muy tarde y las soluciones son complejas. Como consecuencia, la calidad del software se ve degradada y la deuda técnica aumenta, por lo que volver a desarrollar el software aparece como la mejor opción.

Ante la necesidad de una revisión sistemáticas de la literatura al respecto, los autores de "Software Design Smell Detection: a systematic mapping study (2019)" [1], hacen un estudio integral de la literatura para abordar el estado actual del dominio sobre la detección de Design Smells, por lo que se enfoca en todos los tipos de dichas imperfecciones (Bad Smell, Antipatterns, Disharmonies, etc, usando como término unificador el de Design smell) y por otro lado la actividad de detección, así como la especificación, la corrección (refactorización) y la priorización. El objetivo principal, es el mapeo sistemático para obtener y organizar el mayor conocimiento posible sobre su detección en general (enfoques, herramientas, técnicas, conjunto de datos, factores de calidad, etc.). Todo este conocimiento es organizado de tal forma que los autores aplican técnicas de modelado conceptual para describir el área de conocimiento en un modelo de dominio.

El modelo de dominio aportado por los autores es de gran ayuda para la comunidad, puesto que se pretende tener una aplicación que pueda ayudar a tener organizada y accesible toda la información sobre la detección de Design smells obtenida en el estudio; así ayudará a los desarrolladores e investigadores a tomar decisiones para mejorar los procesos de evolución de software.

Como estudiante de ingeniería informática, resulta de gran interés desarrollar una aplicación que pueda ayudar a la comunidad del software, tanto para tener información de cómo los Design smells pueden afectar a la calidad del software (de manera positiva o negativa según el caso) como para mejorar a la hora de la toma de decisiones para afrontar estos problemas recurrentes en el proceso de desarrollo.

Capítulo 2

Antecedentes y explicación del dominio

2.1. Design Smell

Adoptaremos el término de Design Smell como un concepto similar a Code Smell o Bad Smell, pero en un sentido más general, abarca toda la variedad de problemas relacionados con la estructura del software, es decir, la parte de diseño, desde el diseño arquitectónico, de micro-arquitecturas hasta el diseño de grano más fino.

La definicion de Design Smell puede ser variada, algunas definiciones conocidas son las siguientes:

- N.Moha y col.: "Los Code y Design Smells son soluciones deficientes para problemas recurrentes de implementación y diseño." [2]
- RC Martin: "Los Design Smells son los olores del software podrido." [3]
- Fowler: "Los Smells son ciertas estructuras en el código que sugieren la posibilidad de refactorización." [4]
- S. Hassaine y col.: "Sugerimos que, como cualquier criatura viviente, los diseños de sistemas están sujetos a enfermedades, que son los Design smells (Code Smells y Anti-patterns). Los Design Smells se conjeturan en la literatura para impactar la calidad y la vida de los sistemas." [5]

Podemos decir que un Design Smell es una estructura software que no produce errores de compilación o de ejecución, pero produce violaciones de los principios fundamentales del

diseño que afecta negativamente a los factores de calidad del software. Los Design Smell provocan el aumento de la deuda técnica, ya que estos fragmentos de software hacen que el diseño sea frágil y difícil de mantener, lo que obliga a hacer una refactorización para evitar que la deuda técnica aumente y pueda comprometer el desarrollo de un sistema software. En este caso, Design Smell es un concepto unificador que agrupa a términos como Code Smells, Bad Smells, Disharmonies, Antipatterns, Design flaws, Technical debt, entre otros términos conocidos actualmente.

La detección de Design Smells ayuda a reducir notablemente el coste de actividades de mantenimiento del software, ya que permite a los investigadores y desarrolladores de software tomar decisiones para mejorar los procesos de desarrollo y evolución del software. Para la detección se utilizan todo tipo de enfoques, estrategias, técnicas, métodos y herramientas.

Los Design smells se pueden detectar en diferentes artefactos de software, de grano fino a grano grueso, incluidas variables, instrucciones, operaciones, métodos, clases, paquetes, subsistemas, capas y sus dependencias. Podemos encontrar varios ejemplos de Design smells que afectan a diferentes niveles de granularidad, desde métodos hasta la arquitectura del sistema completo. Esta podría ser una de las razones por la cual hay una abundante terminología acerca de los Design smells, como por ejemplo:

- Bad Smell: Los Bad Smells se definen como síntomas en el código fuente que indican la necesidad de refactorización. A continuación mostraremos algunos ejemplos de Bad Smells:

Bad Smell	Grupo	Granularidad
Large Class	Bloater	Class
Long Method	Bloater	Method
Refused Bequest	O-O Abusers	Class
Temporary Field	O-O Abusers	Class
Parallel Inheritance Hierarchies	Change Prevents	Class
Divergent Change	Change Prevents	Class
Data Class	Dispensables	Class
Duplicated Code	Dispensables	Method
Feature Envy	Couplers	Class
Message Chains	Couplers	Class

Tabla 2.1: Ejemplos de Bad Smells.

- Disharmonies: Las Disharmonies son defectos de diseño que afectan a entidades individuales tales como clases y métodos. Tienen un efecto negativo sobre partes del código y solo afectan a esas partes, sin alterar indirectamente. A continuación mostraremos algunos ejemplos de Disharmonies:

Disharmonies		
Identity Disharmonies	Collaboration Disharmonies	Classification Disharmonies
God Class Brain Class Feature Envy Brain Method	Dispersed Coupling Intensive Coupling Shotgun Surgery	Refused Parent Bequest Tradition Breaker

Tabla 2.2: Ejemplos de Disharmonies.

- Antipatterns: Los Antipatterns son ciertos patrones en el desarrollo de software que se consideran malas prácticas de programación. A continuación mostraremos algunos ejemplos de Antipatterns:

AntiPatterns		
Development	Architecture	Project Management
The Blob	Autogenerated Stovepipe	Blowhard Jamboree
Continuous Obsolescence	Stovepipe Enterprise	Analysis Paralysis
Lava Flow	Jumble	Viewgraph Engineering
Ambiguous Viewpoint	Stovepipe System	Death by Planning

Tabla 2.3: Ejemplos de Antipatterns.

Como hemos podido observar, la diferente terminología utilizada en la literatura describe problemas en el diseño del software.

2.2. Revisiones sistemáticas y mapeos sistemáticos de la literatura

2.2.1. Revisión Sistemática de la literatura

La definición de una Revisión sistemática de la literatura (SLR) puede ser variada, algunas definiciones conocidas son:

- García-Peñalvo: “Es un tipo de revisión de la literatura que recopila y analiza críticamente múltiples estudios o trabajos de investigación a través de un proceso sistemático.” [6]
- Sáenz: “Una revisión sistemática es aquella en la que existe una búsqueda exhaustiva de estudios relevantes sobre un tema. Una vez identificados y obtenidos los estudios, los resultados son sintetizados de acuerdo con un método preestablecido y explicito.” [7]

2.2.2. Mapeo sistemático de la literatura

Cascade Project: "Las técnicas de mapeo en la revisión de la literatura (Literature Mapping) son útiles al principio de una revisión sistemática de la literatura como una herramienta de tormenta de ideas y contextualización." [8]. Laś técnicas y resultados obtenidos del mapeo de la literatura son múltiples y dependen del propósito:

- Resumir hallazgos clave de revistas, libros y documentos de trabajo relacionados con un tema para crear mapas conceptuales.
- Presentar un resumen de las publicaciones, revistas, conferencias, años de publicación, autores, instituciones, etc., que se encuentran en la SLR.
- Otros propósitos.

2.3. Antecedentes

Zhang y col. (2011) [9], publicaron una revisión de la literatura que identifica el conocimiento actual sobre 22 Bad Smells descritos por Beck y Fowler. Revisaron 319 artículos publicados entre 2000 y 2009. Analizaron en detalle 39 artículos relacionados con los Bad Smells. De este estudio descubrieron que Bad Smells como el código duplicado (Duplicated Code), tienen mayor atención por parte de los investigadores que otros como la cadena de mensajes (Message Chain).

Rattan y col. (2013) [10], publicaron una revisión sistemática de la literatura sobre la detección de clones de software. Revisaron 213 artículos publicados entre 1998 y 2011. Se centraron únicamente en la de detección de clones de software (otro nombre relacionado con el código duplicado (Duplicated Code)).

Rasool y Arshad (2015) [11], publicaron una revisión de las herramientas y técnicas utilizadas en lo que denominan extracción de Code Smells. Revisaron 42 artículos en detalle publicados entre 1999 y 2015. Se centraron en los 22 Bad smells descritos por Fowler y clasificaron las herramientas y técnicas según métodos de detección.

Fernandes y col. (2016) [12], realizaron una revisión que resume y compara las herramientas de detección de Design Smells. Revisaron y analizaron 107 documentos relevantes publicados entre 2000 y 2015, referentes a las herramientas de detección. Encontraron 84 herramientas capaces de detectar diferentes Design Smells y admitir diferentes lenguajes de programación como Java, C , etc.

Singh and Kaur (2017) [13], realizaron una revisión sistemática de la refactorización sobre Code Smells y Antipatterns. Revisaron y analizaron 238 artículos publicados entre 1990 y

2015. Se centraron en identificar el estado actual de la refactorización, los tipos de enfoques y las herramientas.

Gupta y col. (2017) [14], publicaron una revisión de la literatura para estudiar los Bad Smells en código Java. Revisaron y analizaron en detalle 60 artículos publicados entre 1999 y 2016. Se centraron en las técnicas de detección y la correlación entre ellas y en la identificación de Bad Smells que requieren más investigación por parte de la comunidad.

En el artículo en que se basa este trabajo de fin de grado denominado: "Software Design Smell Detection: a systematic mapping study" [1], se realiza un estudio de mapeo sistemático integral de la literatura sobre la la detección de Design Smells, que difiere de las investigaciones mencionadas anteriormente. Se realizó un estudio más amplio y profundo para abordar el estado actual del dominio sobre la detección de Design Smells.

2.4. El dominio del conocimiento sobre Design Smells

El trabajo de los investigadores es obtener y organizar el conocimiento sobre la detección de Design Smells en general (enfoques, herramientas, técnicas, conjuntos de datos, factores de calidad, etc.). Para la obtención del conocimiento se realizan búsquedas avanzadas en base a datos con estudios y publicaciones relevantes. Se hacen tanto búsquedas automatizadas como manuales, en las que se comparan cadenas de búsqueda, tomando aquellas que tienen como fecha de publicación el periodo comprendido entre los años 2000 y 2018. De la búsqueda sistemática se extraen datos esenciales para fabricar un modelo de datos conceptual, que es una base para el modelo de datos relacional.

Los métodos utilizados para el mapeo del conocimiento integral son: definir y responder preguntas sobre el tema, determinar estrategias de investigación, clasificar trabajos incluidos y sintetizar los resultados.

Las preguntas que se plantean en el estudio son las siguientes:

- ¿Qué tipos de Design Smell se detectan?
- ¿Qué enfoques se han propuesto para detectar Design Smells en el software?
- ¿Qué herramientas (prototipo) se han utilizado para detectar Design Smells en el software?
- ¿La herramienta (prototipo) está validada por un experto o punto de referencia o en comparación con otras herramientas? ¿La herramienta o estrategia evalúa los resultados y mide el rendimiento, la precisión y la exhaustividad?
- ¿La detección de Design Smell está relacionada con los atributos de calidad de un modelo de calidad?

La estrategia de investigación aborda una cadena de búsqueda, el alcance de la investigación, el período de búsqueda, el método de búsqueda y los criterios de inclusión/exclusión que identifican los autores.

- La cadena de búsqueda final elegida por los autores es la siguiente: ("Design Smell" or "design-smell" or "bad smell" or "bad-smell" or "code smell" or "codesmell" or "design defect" or "design-defect" or "design flaw" or "design-flaw" or "antipattern" or "anti-pattern" or "disharmony" or "disharmonies") and ("detection" or "detecting" or "identification" or "identifying" or "finding" or "empirical")
- Para obtener una gran cobertura de búsqueda de estudios y publicaciones relevantes, se incluyen seis bases de datos electrónicas más populares y eficientes para realizar estudios sistemáticos para temas relacionados con la ingeniería y reingeniería del software. Estas bases de datos son las siguientes: Science Direct (www.sciencedirect.com), Scopus (`www.scopus.com`), Web of Science (`www.isiknowledge.com`), IEEE Xplore (`https://ieeexplore.ieee.org/Xplore/home.jsp`), Biblioteca digital ACM (`www.acm.org`) y Springer (`www.springerlink.com`) y también se incluye Google Scholar (`https://scholar.google.com/`).
- El período de búsqueda comprende desde enero de 2000 hasta diciembre de 2018.
- Para el método de búsqueda se utilizan búsquedas automáticas y manuales.
- Los criterios de inclusión elegidos son: documentos publicados dentro del periodo de búsqueda, documento completo publicado en conferencia, revista, etc. y documentos en los que la cadena de búsqueda aparezca en el titulo, resumen y palabras clave. Por otro lado, los criterios de exclusión son: documentos que no tengan relación con la cadena de búsqueda; informes, documentos de posición, tesis doctorales, propuestas de investigación, proyectos; documentos duplicados de un mismo estudio; documentos que no tienen su texto completo disponible y documentos no escritos en inglés.

Para clasificar los trabajos incluidos en el estudio se utilizan facetas definidas mediante diferentes palabras clave, que sirven para responder preguntas de investigación. Por ejemplo la faceta "Enfoque principal"tiene relacionadas palabras clave como: Design Smell Definition, Design Smell Detection y otras actividades relacionadas (refactoring, prioritization, etc.); La faceta "Tipo de Design Smell"tiene como palabras clave: Disharmony, Code Smell, Antipattern, Bad Smell, Design Defect, Design Flaw.

Para sintetizar los resultados se extraen los datos de los documentos relevantes una vez se ha completado el proceso de búsqueda. Se extrae la información relacionada con las preguntas de investigación y con las facetas utilizadas para clasificar los trabajos. Además, se extrae información extra para mejorar la organización del conocimiento como pueden ser las técnicas utilizadas (algoritmos, heurísticas, etc.), el grado de automatización de herramientas y enfoques, etc.

Una vez realizados todos los pasos para el mapeo del conocimiento integral, se analizan los datos extraídos y se aplican técnicas de modelado conceptual para crear un modelo de dominio sobre la detección de Design Smells.

El Modelo de dominio en el que se basa la aplicación web es el proporcionado por los autores del artículo [1], en el se ven todos los datos referentes a la detección de Design smell, como se puede observar en la Figura 2.1 y los tipos enumerados obtenidos después de la extracción de datos como se puede observar en la figura 2.2.

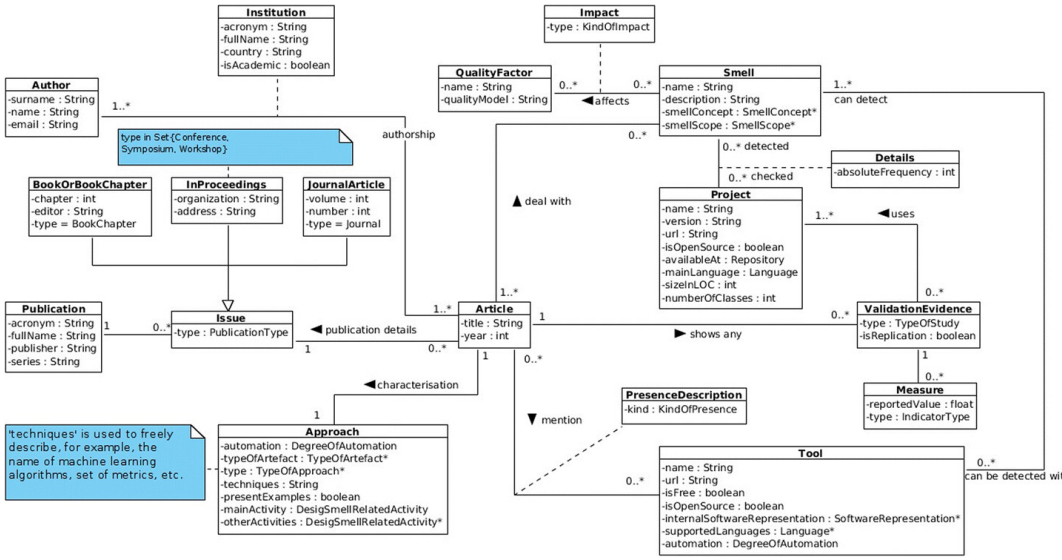

Figura 2.1: Modelo conceptual del conocimiento sobre la detección de Design smell. Tomada de [1].

En el modelo de dominio, se muestra por una parte a los autores, instituciones y artículos, que se modelan teniendo en cuenta que un mismo autor puede cambiar, durante un periodo de tiempo, de una institución a otra. Cada artículo tiene asociado un enfoque en el que se describen los métodos utilizados para detectar los Design Smells. Los atributos marcados con '*', pueden tomar más de un valor, por lo que puede haber combinaciones de tipos de enfoque o tratar con varios artefactos.

Un artículo también pueden mencionar herramientas y se pueden caracterizar según el tipo de presencia como presentar una herramienta por primera vez, mejorar una ya existente, comparar unas con otras o revisarlas en profundidad. Las herramientas se asocian con los Design Smell que pueden detectar.

Los Design smells tratados en un artículo tienen atributos que determinan el alcance que tiene (como de clase, método, sistema, etc.), así como el concepto que usan los autores para referirse a un Design Smell (como Code smell, Bad Smell, Design Flaw, etc.).

Las evidencias de validación asociados a los artículos, indican si los autores realizaron un estudio de caso, una encuesta, un experimento, etc. Además, hay proyectos asociados a las evidencias. Los Design Smells se asocian a los factores de calidad a los que afectan,

indicando el impacto que tienen sobre ellos. También existe la posibilidad de que los Design Smells puedan afectar a más de un factor de calidad, ya sea positiva o negativamente. Los proyectos permiten conocer qué Design smells se detectan en la validación y por otro lado qué indicadores se presentan como resultado de la validación.

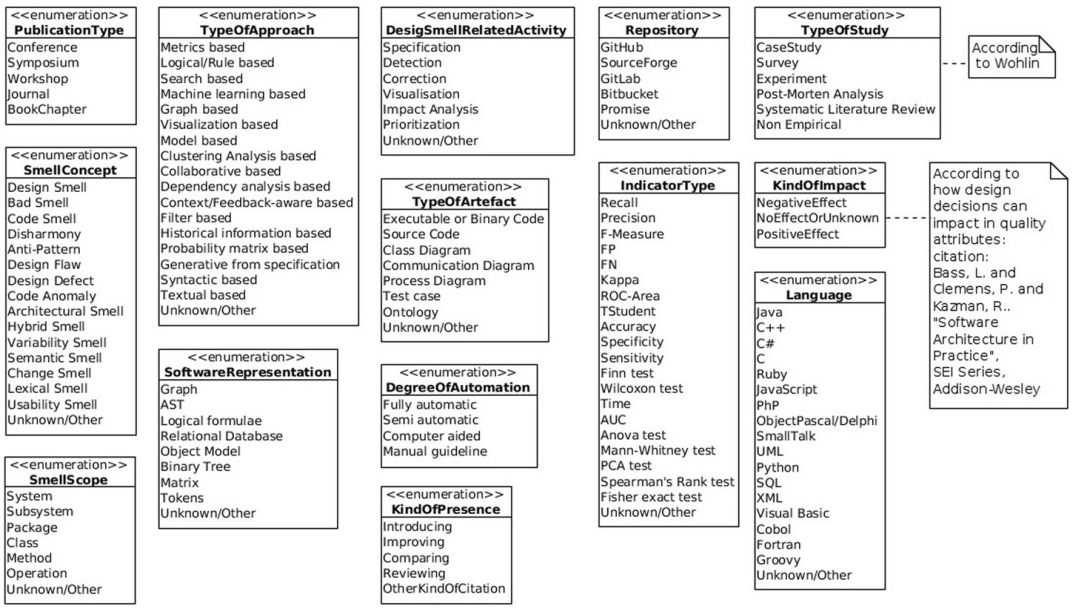

Figura 2.2: Enumerados obtenidos durante la extracción de datos. Tomado de [1].

Una vez que el modelo fue considerado definitivo, se realizó un diseño de la base de datos en MYSQL, el cual posteriormente se comparte a la comunidad.

Capítulo 3

Planificación

3.1. Planificación

La realización del proyecto está determinada por una metodología ágil basada en SCRUM [15, 16] con ciertas modificaciones para adaptarlas a las limitaciones del equipo y la propia asignatura. Por lo tanto en este marco de trabajo, los requisitos de usuario pasan a ser historias de usuario. Dichas historias se repartirán a lo largo del proyecto, en intervalos de tiempo denominados sprints.

El comienzo del proyecto se ha fijado para el 25 de Febrero de 2019, teniendo en cuenta el calendario laboral. Se establecen 7 iteraciones de 2 semanas de duración por cada iteración. La fecha prevista de finalización del proyecto será el 17 de Junio de 2019.

El proyecto se desarrollará a lo largo de 14 semanas, con un número número de horas estimadas para la realización de cada sprint como se muestra en la tabla inferior.

Sprints	Horas requeridas
Sprint 1	60h
Sprint 2	50h
Sprint 3	30h
Sprint 4	36h
Sprint 5	44h
Sprint 6	70h
Sprint 7	70h
Total	360h

Tabla 3.1: Estimación en horas por iteración.

Trabajaremos en un marco de trabajo iterativo e incremental, donde los requisitos evolucionan en función de las necesidades del proyecto, se trabaja colaborativamente en equipo y se mantiene una relación continua con el cliente. Se realizarán entregas parciales y regulares del producto final, donde al comienzo y al final de cada sprint se realizarán reuniones para mejorar la organización y aumentar la comunicación entre el equipo.

Por lo tanto las reuniones denominadas dailies scrum meeting, se realizarán una vez a la semana, ya que no es posible una reunión diaria debido a la incompatibilidad de horarios entre el tutor y el alumno. Por otro lado, el sprint review y el sprint retrospective se hacen al finalizar cada sprint, tras presentarle la funcionalidad planificada en el sprint al tutor del proyecto.

3.1.1. Roles y responsabilidades

- Product Owner: Será el nexo de unión entre equipo de desarrollo y responsables de negocio. Controla la evolución del proyecto, revisando el producto que se va desarrollando. En este caso se trata de la tutora del proyecto.

- Scrum Master: Es la persona responsable de apoyar al equipo de desarrollo , eliminar las barreras organizativas y mantener la consistencia del proyecto ágil. Según la metodología SCRUM es preferible que lo ocupe una persona a tiempo completo. En este caso se trata del alumno que realiza el proyecto.

- Team: Es el grupo de personas que de manera conjunta desarrollan el producto del proyecto. Tienen un objetivo común, comparten la responsabilidad del trabajo que realizan (así como de su calidad) en cada iteración y en el proyecto. En este caso se trata del alumno que realiza el proyecto.

3.1.2. Historias de usuario

Dado que el proceso de desarrollo está basado en Scrum, los requisitos del proyecto quedarán determinados en forma de historias de usuario. Estas historias representan las necesidades del cliente y estarán distribuidas en intervalos de tiempo denominados sprints.

Todas la historias de usuario que se prevén ser implementadas para dotar de funcionalidad completa a la aplicación, se van añadiendo en el Icebox. Por lo tanto, en este apartado se van añadiendo historias de usuario que se quieren implementar en un futuro y pueden ir incrementándose debido a las necesidades que vaya teniendo el cliente durante el desarrollo del proyecto.

Historia de usuario
Como un usuario registrado quiero iniciar una sesión para poder utilizar la aplicación.
Como un usuario no registrado quiero registrarme en el sistema para poder utilizar la aplicación.
Como un usuario identificado quiero cerrar sesión para salir de la aplicación.
Como un usuario quiero consultar los conocimientos sobre la detección de Design Smells.
Como un usuario quiero visualizar las estadísticas sobre la detección de Design Smells.
Como un usuario quiero contribuir con un nuevo elemento al conocimiento ya obtenido.
Como un usuario quiero realizar un comentario sobre el detalle de un elemento.
Como un usuario quiero realizar una sugerencia de corrección sobre el detalle de un elemento.
Como un usuario quiero contribuir dando información de un elemento que no lo tiene.
Como un Administrador quiero aprobar una nueva contribución hecha por un usuario.
Como un Administrador quiero aprobar un comentario de un elemento hecho por un usuario.
Como un Administrador quiero aprobar una sugerencia de corrección de un elemento hecha por un usuario.
Como un Administrador quiero aprobar la nueva información proporcionada de un elemento que no tenia hecha por un usuario.

Tabla 3.2: Icebox

Para este proyecto las historias de usuario fueron especificadas en el Icebox de Pivotal tracker, en ella están los requisitos que desea obtener el cliente en el desarrollo del proyecto, así como los nuevos requisitos que se van proporcionando.

Una vez analizados los requisitos a través de historias de usuario seleccionamos las que desarrollaremos en este trabajo de fin de grado, definiremos el Product Backlog donde las historias de usuario quedarán priorizadas y ordenadas para su implementación en los distintos sprints.

3.1.3. Product Backlog

El Product backlog es una lista priorizada de objetivos a realizar en el proyecto. Estos objetivos representan la visión y expectativas del proyecto. A continuación mostramos los requisitos que se realizarán ordenados por prioridad y su esfuerzo estimado en puntos-historia.

Estas historias de usuario se estimarán con una puntuación de puntos-historia que determinan el esfuerzo necesario para su realización y llevarán asociada una prioridad con la que serán implementadas. Esta estimación sigue una tendencia lineal (0,1,2,3), dando '0' a las historias de usuario que requieran poco esfuerzo para su implementación y '3' las que requieran más esfuerzo para implementarlas.

Historia de usuario	Descripción	Prioridad	Estimación
US-001	Como un usuario registrado, quiero iniciar una sesión para poder utilizar la aplicación.	1	3
US-002	Como un usuario no registrado, quiero registrarme en el sistema para poder utilizar la aplicación.	2	2
US-003	Como un usuario identificado, quiero cerrar sesión para salir de la aplicación.	3	1
US-004	Como un usuario, quiero consultar los conocimientos sobre la detección de Design Smells.	4	3
US-005	Como un usuario, quiero visualizar las estadísticas sobre la detección de Design Smells.	5	3
US-006	Como un usuario, quiero contribuir con un nuevo elemento al conocimiento ya obtenido.	6	3

Tabla 3.3: Product Backlog

Las historias de usuario que se abordarán en el desarrollo del proyecto se describirán con más detalle en el Capítulo 6, ya que aquí sólo se muestra una descripción simple de la funcionalidad a implementar.

3.2. Gestión de riesgos

Al ser un proyecto software se quiere obtener un producto con la mayor calidad y minimizar el impacto de los riesgos [17]. Por ello, se han identificado los principales riesgos que puede presentar el proyecto a lo largo del desarrollo separándolos en tres categorías [17]:

- Riesgos de proyecto: Afectan negativamente al plan de proyecto. Si estos ocurren el esfuerzo necesario y el coste aumentarán considerablemente.
- Riesgos técnicos: Afectan negativamente a la calidad. Si estos ocurren el proyecto tendrá una complejidad excesiva.
- Riesgos de negocio: Afectan negativamente la viabilidad del producto a construir. Si estos se cumplen el proyecto se podría cancelar. Debido al carácter educativo del proyecto, no se puede dar el caso de que el sistema no sea viable en el mercado.

Dado que a la hora de identificar los riesgos es necesario determinar una probabilidad de que ocurra el riesgo, así como el impacto de los mismos que afectan negativamente a lograr los objetivos del proyecto, se elabora una matriz en la que se prestará más atención a los riesgos de carácter alto en los que se precise una planificación de respuesta frente al riesgo.

Impacto/Probabilidad	Alto	Medio	Bajo
Alto	Planificar	Planificar	Considerar
Medio	Planificar	Considerar	Nada pero monitorizar
Bajo	Considerar	Nada pero monitorizar	Nada pero monitorizar

Tabla 3.4: Matriz de riesgos.

Ante la posibilidad de que un riesgo se haga realidad y pueda involucrar el fracaso del proyecto, tomaremos decisiones para afrontarlos y tenemos tres opciones para hacerlo [17]:

- Impedir el riesgo.
- Transferir el riesgo.
- Aceptar el riesgo, en este caso encontramos dos situaciones:
  - Mitigarlo, tomando decisiones rápidas para reducir la probabilidad o impacto.
  - Definir un plan de contingencia. Determinar qué se debe realizar en el caso en el que el riesgo llegue a ser un problema. Dicho de otra forma, crear un ”plan B”.

A continuación, se plantean algunos de los posibles riesgos que pueden ocurrir a lo largo del desarrollo del proyecto, así como un plan de acción para cada uno de ellos.

Formulario de gestión de riesgos		
Identificador: R-01	Título: Retrasos en la planificación	
Tipo: Riesgo de proyecto	Probabilidad: Alta	Impacto: Alto
Descripción: Retrasos en la planificación debido a la estimación incorrecta de las actividades a realizar.		
Plan de acción: Se realizará una nueva planificación aumentando las horas requeridas para la realización de las actividades que se prevén para alcanzar los objetivos.		

Tabla 3.5: Riesgo-01

Formulario de gestión de riesgos		
Identificador: R-02	Título: Baja médica del equipo de desarrollo	
Tipo: Riesgo de proyecto	Probabilidad: Media	Impacto: Alto
Descripción: En este caso el proyecto es realizado por una sola persona, por lo que una baja médica conllevaría al retraso de entrega del mismo.		
Plan de acción: Evaluar el tiempo perdido, y replanificar. Aumentar las horas dedicadas a la realización de las actividades.		

Tabla 3.6: Riesgo-02

Formulario de gestión de riesgos		
Identificador: R-03	Título: Problemas con software de terceros	
Tipo: Riesgo de proyecto	Probabilidad: Media	Impacto: Medio
Descripción: La utilización de software de terceros, puede que contenga fallos y no sea el adecuado, provocando un retraso en la realización del proyecto.		
Plan de acción: Ante todo se utilizará primordialmente software de páginas oficiales y si no es el caso se hará un estudio de lo que ofrece y carece el software seleccionado.		

Tabla 3.7: Riesgo-03

Formulario de gestión de riesgos		
Identificador: R-04	Título: Mala comunicación entre cliente y equipo	
Tipo: Riesgo de proyecto	Probabilidad: Baja	Impacto: Alto
Descripción: La falta de comunicación con el tutor del proyecto, afectará en el desarrollo correcto del proyecto, dando lugar a un trabajo incompleto.		
Plan de acción: Se establecen diferentes canales de comunicación, así como reuniones para la resolución de dudas.		

Tabla 3.8: Riesgo-04

Formulario de gestión de riesgos		
Identificador: R-05	Título: Falta de experiencia en el equipo	
Tipo: Riesgo Técnico	Probabilidad: Medio	Impacto: Alto
Descripción: Falta de experiencia y de conocimientos técnicos en las herramientas, procedimientos, tecnologías, etc, que puedan provocar un retraso en la realización del proyecto así como afectar a la calidad del producto.		
Plan de acción: Aumentar las horas dedicadas a la formación del personal.		

Tabla 3.9: Riesgo-05

Formulario de gestión de riesgos		
Identificador: R-06	Título: Cambio en los requisitos	
Tipo: Riesgo Técnico	Probabilidad: Medio	Impacto: Alto
Descripción: Cambios considerables en los requisitos durante el desarrollo del proyecto.		
Plan de acción: Evaluar si los cambios son o no viables para aprobar o rechazar el cambio.		

Tabla 3.10: Riesgo-06

Formulario de gestión de riesgos		
Identificador: R-07	Título: Diseño inadecuado	
Tipo: Riesgo Técnico	Probabilidad: Bajo	Impacto: Alto
Descripción: Partir de un diseño inadecuado puede provocar problemas graves en el desarrollo del proyecto.		
Plan de acción: Revisión del diseño durante el desarrollo.		

Tabla 3.11: Riesgo-07

Los riesgos anteriormente mencionados, ocurren a lo largo del desarrollo del proyecto. Esto se ve reflejado en el seguimiento del mismo en el capítulo 5.

3.3. Calendarización

Para el desarrollo completo del proyecto, se han estimado siete iteraciones(sprints), cada uno con un número de horas estimadas de duración. Dado que partimos de un modelo de dominio inicial la mayor parte del desarrollo ha sido de implementación. Se establecen los plazos estimados de entrega de cada iteración, como se muestra a continuación.

Sprints	Fecha finalización
Sprint 1	12 de Marzo de 2019
Sprint 2	26 de Marzo de 2019
Sprint 3	9 de Abril de 2019
Sprint 4	30 de Abril de 2019
Sprint 5	14 de Mayo de 2019
Sprint 6	28 de Mayo de 2019
Sprint 7	17 de Junio de 2019

Tabla 3.12: Estimación de entrega por iteración.

En la siguiente tabla se muestra el desglose semanal estimado desde el inicio del proyecto el 25 de Febrero con su sprint o iteración correspondiente, así como el final previsto del proyecto.

Semana	Fechas	Sprint
1	25 de Febrero - 4 de Marzo	1
2	5 de Marzo - 11 de Marzo	
3	12 de Marzo - 18 de Marzo	2
4	19 de Marzo - 25 de Marzo	
5	26 de Marzo - 1 de Abril	3
6	2 de Abril - 8 de Abril	
7	9 de Abril - 15 de Abril	4
8	23 de Abril - 29 de Abril	
9	30 de Abril - 6 de Mayo	5
10	7 de Mayo - 13 de Mayo	
11	14 de Mayo - 20 de Mayo	6
12	21 de Mayo - 27 de Mayo	
13	4 de Junio - 10 de Junio	7
14	10 de Junio - 17 de Junio	
-	17 de Junio	Entrega

Tabla 3.13: Calendarización.

3.4. Análisis de coste inicial

Para el coste inicial se tienen en cuenta tanto el hardware y software utilizado, así como el coste de horas trabajadas en el proyecto.

3.4.1. Coste hardware y software

Para el desarrollo del proyecto se ha utilizado el siguiente hardware:

- Ordenador portátil con un coste de 0 €, ya que ha sido adquirido a modo de préstamo a través de la universidad.
- Máquina virtual para el alojamiento de la base de datos que ha sido proporcionada por la escuela con un coste de 0 €.
- Para el despliegue de la aplicación se ha utilizado Heroku, que nos brinda unidades de cómputo denominadas dynos con un coste de 0 €.

No tenemos gastos por parte del hardware. El software utilizado es gratuito, no requiere un coste a la hora de utilizarlo, por lo tanto los gastos originados por el software también son nulos.

3.4.2. Coste recursos humanos

En cuanto a los recursos humanos, ha sido necesario buscar información sobre cuál es el salario medio de un desarrollador web en España y basándonos en ello hacemos una estimación inicial de cuánto va a ser el coste, tomando como unidad de medida horas/hombre.

Según el portal de empleo indeed.es, el salario anual estimado de un desarrollador web es de 22.941 € a fecha de julio de 2019. Esta estimación la hace tomando en cuenta varias fuentes.

Por lo tanto, un desarrollador web cobra por hora 11,95 €. Dado que la estimación inicial tiene como objetivo hacer 360 horas, en total de recursos humanos tenemos un coste de 360 horas * 11,95 € = 4302 €.

3.4.3. Coste total inicial

En total los costes del proyecto son 4302 € ya que sólo se producen gastos en los recursos humanos. Se trata de una estimación por lo que el coste real del proyecto puede variar durante su realización.

Capítulo 4

Herramientas y Tecnologías

4.1. Herramientas y Tecnologías

En este apartado se describirán las herramientas utilizadas en el desarrollo del proyecto y se englobarán en distintas categorías dependiendo de su uso.

4.1.1. Herramientas de planificación y gestión

En este grupo de herramientas se encontrarán las que sirven para la planificación del proyecto así como su seguimiento, las encargados del control de versiones y la utilizada como repositorio remoto.

Pivotal tracket

Pivotal tracker [18] es una herramienta simple para la gestión de proyectos ágiles que permite el seguimiento de tareas. La elección de esta herramienta viene determinada por el consejo de la tutora del proyecto, además de por su facilidad de uso y la gran cantidad de funcionalidades que ofrece.

Es de gran utilidad a la hora de administrar múltiples proyectos. Incluye espacios de trabajo que permiten ver proyectos en paralelo, además de otras funcionalidades.

Tiene más de 140 extensiones, incluyendo GitHub, Slack y Zendeck, de manera que podamos extender la funcionalidad en cuanto a la productividad, administración de proyectos, seguimiento de problemas, análisis, rastreo de tiempo, etc.

Algunas de sus ventajas son:

- Multitud de funcionalidades avanzadas como planificación automática de sprints, espacios de trabajo multiproyecto, más de 100 integraciones y analíticas profundas.
- Las aplicaciones nativas garantizan que Pivotal Tracker funcione de manera fluida en cualquier dispositivo, aunque está limitada a dos proyectos simultáneos.

Git

Git [19] es un sistema de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es permitir el seguimiento el código de la aplicación.

La elección de esta herramienta viene determinado por el intenso uso que le hemos dado en nuestro desarrollo profesional hasta el momento, además de las grandes ventajas que nos proporciona:

- Velocidad.
- Seguridad: los controladores de versiones cuentan con sistemas de cifrado y otros tipos de medidas de seguridad que se aplicarán para que nuestros datos permanezcan de forma íntegra. Git usa sistemas de árbol SHA1, lo que asegura que hasta que no se realice la comprobación del cifrado, los cambios no se escribirán en el servidor.
- Gratuidad de la herramienta.
- Compartición selectiva: posibilidad de compartir las partes que queramos del proyecto en cuestión y con quién queramos, decidiendo si sólo podrá, que tenga la posibilidad de añadir notas, comentarios o que pueda añadir cambios, en función de las necesidades.
- Convergencia: en el caso de que al crear una rama del proyecto localicemos que uno de los cambios incluidos se integra tal y como queremos, tenemos la opción de incluir dicha ramificación con el desarrollo principal de forma sencilla y segura.
- Sandbox: gracias a ello, los cambios realizados en alguna de las ramas del proyecto no tendrán consecuencias para los usuarios que en el momento estén usando la versión sin modificar (versión principal).
- Flujo de trabajo adaptable: Git nos ofrece la posibilidad de gestionar el flujo de desarrollo de la aplicación de diferentes maneras, destacando los modelos centralizados y los modelos de libre configuración.
- Ramificación: Git ofrece muchas posibilidades a la hora de realizar cambios en la estructura principal, pudiendo crear diferentes ramas sobre las que aplicar nuestras modificaciones.

GitLab

Gitlab [20] es un servicio web de control de versiones y gestor de repositorios basado en Git. Similar a su principal competidor, GitHub.

La elección de esta herramienta viene condicionada por su uso constante en diversas asignaturas, y por las ventajas que ofrece:

- Es gratuito: los usuarios pueden tener un número ilimitado de repositorios privados.
- Integración con LDAP que permite localizar y acceder a diversos recursos de internet. GitLab EE soporta varios servicios LDAP y sincronización de grupos.
- Soporta la importación de Git.
- GitLab opera bajo una licencia de código abierto (open source).
- Seguimiento de errores y edición de código basado en la web.

4.1.2. Herramientas de Diseño y Desarrollo

En este grupo de herramientas encontraremos las utilizadas para la realización de diagramas, el entorno de desarrollo con el que se realiza la aplicación, el entorno para realizar la documentación y la plataforma de despliegue utilizada.

Astah

Astah [21] una herramienta de diseño de sistemas que soporta UML, diagrama de relación de entidades, diagramas de flujo, CRUD, Diagrama de flujo de datos, Tabla de Requisiciones y Mapas Mentales. En este proyecto se utilizará la versión astah UML, que es gratuita para estudiantes.

La elección de está herramienta es por el intenso uso en asignaturas relacionadas con el diseño de software.

Eclipse

Eclipse [22] es un entorno de desarrollo, diseñado principalmente para Java (siendo uno de los mejores IDE para su desarrollo) pero que gracias a los plug-ins, se puede extender a otros lenguajes y herramientas de desarrollo.

Es un proyecto de código abierto y además cuenta con la ventaja de que se puede utilizar en diversos sistemas operativos como son Windows, cualquier distribución del sistema Linux o Solaris.

Es un proyecto desarrollado por la comunidad open source llamada Eclipse Foundation, por lo que posee una licencia de utilización pública.

Esta herramienta se ha utilizado para el desarrollo de la aplicación web por las ventajas que presenta:

- Dispone de un editor de texto con resaltador de sintaxis.
- Compilación en tiempo real.
- Posibilidad de realizar pruebas unitarias con (JUnit).
- Control de versiones con (CVS).
- Asistente para creación de proyectos.
- Plug-ins para aquellas ocasiones en las que el usuario necesite alguna funcionalidad adicional, a diferencia de otros entornos donde las funcionalidades están todas incluidas, se necesiten o no.
- Estos plug-ins permiten extenderse a Eclipse usando otros lenguajes de programación como son C/C++ y Python, además consiguen que Eclipse pueda trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y SGBD (Sistemas de gestión de bases de datos).

Overleaf

Overleaf [23] es un servicio de LateX colaborativo en línea con el cual puedes realizar documento utilizando LateX en la nube. La documentación del proyecto ha sido elaborada con este editor.

- Diversidad de diferentes plantillas.
- Posibilidad de descargar el código fuente de un documento.
- Previsualizar de forma automática o manual el documento.
- Elección de compilador con LaTeX.
- Importación de gráficos desde Plot.ly12
- Importación de documentos desde el ordenador o desde servicios como Google Drive o Dropbox.
- Posibilidad de sincronización con Dropbox.
- Uso del proyecto como un repositorio mediante comandos de GIT.

Heroku

Heroku [24] es una plataforma como servicio de computación en la Nube(PaaS), que soporta distintos lenguajes de programación y permite a los desarrolladores centrarse en el desarrollo de la aplicación y no preocuparse por la infraestructura que la albergará.

Ventajas que presenta la plataforma:

- Gratuito: Heroku ofrece una opción gratuita para alojar una aplicación durante 20 días. Adicionalmente la gran mayoría de los add-ons cuenta con planes gratuitos suficientes para poder albergar la aplicación web.
- Estabilidad: la infraestructura de Heroku utiliza AWS (Amazon Web Services) lo que proporciona excelente rendimiento, disponibilidad y estabilidad.
- Orientado a desarrolladores.
- Seguridad: los dynos (unidades que proveen capacidad de cómputo dentro de la plataforma) se encuentran aislados entre sí. Gracias a Heroku Shield es posible cumplir la regulación HIPPA, adicional las conexiones a los servicios se ofrecen a través de HTTPS, así mismo los add-ons utilizan protocolos seguros.
- Despliegues ágiles: en poco tiempo y comandos de terminal se puede tener una aplicación preparada para producción.
- Rapidez: las aplicaciones se ejecutan rápidamente.
- Documentación: gran cantidad de documentación útil.
- Multiplataforma: los dynos están basados en Linux, pero los comandos para interactuar con ellos se puede instalar en Windows, Linux y Mac.
- Escalabilidad: tanto los dynos como los add-ons son escalables y se ajustan a las necesidades.
- Git privado: Heroku ofrece hasta 100 repositorios git privados de máximo 500MB de manera gratuita.

Balsamiq Mockups

Balsamiq Mockups [25] es una herramienta para la creación de bocetos de interfaces de usuario. Con esta herramienta podemos tener un punto de partida del producto que queremos, y mostrarle al cliente una visión aproximada de como se verá la aplicación.

Ventajas que presenta la plataforma:

- Productividad: facilita y agiliza la creación de bocetos, ya que cuenta con una gran cantidad de componentes prediseñados.
- Los componentes cuentan con múltiples opciones de edición.
- Permite exportar y compartir los diseños.

4.1.3. Backend y Frontend

En esta sección encontraremos las herramientas utilizadas para el desarrollo en el lado del servidor y la parte del cliente.

Por lo tanto contiene las encargadas de la gestión de dependencias, los frameworks de aplicación, herramientas para la persistencia de datos y herramientas encargadas de la generación de gráficos.

Maven

Maven [26] es una herramienta de código abierto (open source) creada en 2001 para la gestión y construcción de proyectos Java, con el objetivo de simplificar los procesos de build (compilar y generar ejecutables a partir del código fuente).

Permite la gestión de un proyecto completo, desde la primera etapa en la que se comprueba que el código es correcto, hasta que se despliega la aplicación, pasando por la etapa de pruebas y generación de informes y documentación.

Razones por las que se ha usado en el proyecto:

- La principal ventaja que nos aporta es que evita tiempos de configurar e implementar el entorno de desarrollo. De esta manera nos podemos centrar en la parte realmente importante de nuestro proyecto Java: el desarrollo y la documentación del código.
- Aporta consistencia a los proyectos.
- Permite reutilizar distintos arquetipos para hacer otro más grande si fuera necesario. Gracias a Maven podemos elegir patrones determinados, en función de nuestras necesidades. Así empezaremos a trabajar prácticamente de forma directa sobre el proyecto.
- Ayuda con los procesos de estandarización.

Zk Framework

ZK [27] es un framework de aplicaciones web en AJAX basado en componentes que permite crear aplicaciones de Internet enriquecidas (RIA) y aplicaciones móviles sin tener que aprender JavaScript o AJAX.

Además de la programación basada en componentes y orientación a eventos, ZK soporta un lenguaje de marcación para la definición de una potente interfaz de usuario llamada ZUML.

Cuenta con las siguientes ventajas:

- No es necesario que el desarrollador tenga conocimientos de Ajax o JavaScript.
- Es un modelo basado en componentes intuitivos, dirigido por eventos.
- ZUML permite a los usuarios no expertos diseñar eficientemente interfaces de usuario; permite mezclar diferentes tipos de lenguaje de marcación (como XUL de Mozilla y XHTML) en la misma página; además es posible que los desarrolladores puedan embeber scripts en lenguaje Java.
- Empotrar script en Java ayuda al prototipado rápido y a las personalizaciones.
- Permite centrar toda la lógica de programación en el servidor.
- Plugin para Eclipse e IntelliJ, que proporciona herramientas intuitivas que abarcan todo el ciclo de vida de desarrollo de aplicaciones.

Spring

Spring [28] es un framework para el desarrollo de aplicaciones y contenedor de inversión de control. En este caso lo integraremos junto a ZK Framework. Actualmente es el framework más popular para Java empresarial, ya que nos ofrece la posibilidad de crear código de alto rendimiento, poco pesado y además reutilizable. Su principal objetivo es estandarizar, agilizar, manejar y resolver los problemas que puedan ir surgiendo en el transcurso de la programación.

- Organización modular: únicamente nos tendremos que ocupar de aquellos paquetes y clases que necesitemos para nuestro desarrollo.
- Gracias a la AOP (Programación Orientada a Aspectos), podemos separar la lógica comercial del registro, la auditoría, las transacciones declarativas, la seguridad, el almacenamiento en caché, etc.
- Accesibilidad a una API para traducir excepciones específicas de la tecnología (las generadas por JDBC, Hibernate, etc) en excepciones consistentes y no verificadas.

- Utiliza frameworks ORM, JEE, temporizadores Quartz y JDK, frameworks de registro y otras tecnologías de visualización.

- Existencia de plantillas para diversas tecnologías como son JDBC, Hibernate y JPA. Así no será necesario escribir código tan extenso, ya que con estas plantillas simplifica el trabajo en cuanto a los pasos básicos a implementar de estas tecnologías.

- Los contenedores de IoC (Inversion of Control) son poco pesados, sobre todo si se comparan con los Enterprise JavaBeans (EJB). Esta es una gran ventaja a la hora del desarrollo y despliegue de aplicaciones en equipos con poca memoria o recursos.

- Facilidad a la hora de probar aplicaciones escritas con Spring, ya que el código dependiente del entorno se traslada a este framework. Es más fácil utilizar la inyección de dependencia para hacer pruebas mediante el uso de JavaBeanstyle.

Spring security

Spring Security [29] es un framework que se centra en proporcionar autenticación y autorización a las aplicaciones Java. En este caso lo integramos junto a ZK.

Ventajas:

- Se puede dividir la lógica de nuestras aplicaciones del control de la seguridad, utilizando filtros para las peticiones al servidor de aplicaciones.

- Soporta gran diversidad de modelos de identificación de los usuarios, como son LDAP, OpenID, HTTP BASIC, etc.

- Capacidad de gestionar seguridad en varios niveles: URLs que se solicitan al servidor, acceso a métodos y clases Java, y acceso a instancias concretas de las clases.

- Portabilidad de la configuración de la seguridad de un servidor a otro.

JPA (Java Persistence API)

JPA es una API que nos ofrece Java para implementar un Framework Object Relational Mapping (ORM), de manera que se pueda interactuar con la base de datos por medio de objetos. Podemos deducir que JPA se encarga de convertir los objetos Java en instrucciones para el Manejador de Base de Datos (MDB). Tendrá como implementación Hibernate [30] y lo integraremos junto a ZK y Spring.

PostgreSQL

PostgresSQL [31] es un sistema de gestión de base de datos relacionales, está orientado a objetos, es multiplataforma y open source. En este caso se ha optado por migrar una base de datos MySQL a PostgreSQL por su compatibilidad con Heroku.

Ventajas:

- Robustez y fiabilidad de la información.
- Coste: es un gestor de base de datos de código libre y completamente gratuito.
- Estándar SQL: cuenta con la mayor parte de las funcionalidades principales del estándar SQL. Es posible incluir consultas y scripts.
- Estabilidad: PostgreSQL está en constante desarrollo y soporte de sus versiones.
- Escalabilidad y configuración: posibilidad de configurar de forma individual PostgreSQL, adaptándolo a los recursos de hardware disponibles en nuestro sistema.
- Compatibilidad: es compatible con prácticamente todas las tecnologías y sistemas operativos de la actualidad.
- Herramienta gráfica: administración de las bases de datos de forma sencilla gracias a la herramienta gráfica que incorpora.

Carewebframework-highcharts

Carewebframework-highcharts [32] es un componente ZK que soporta la librería de gráficos JavaScript HighChart. Se utiliza para la generación de gráficos con código Java.

4.1.4. Pruebas

En esta sección indicaremos las herramientas utilizadas para la automatización de pruebas de la aplicación.

JUnit

JUnit [33] es un framework para la automatización de pruebas (unitarias e integración) en Java. Nos permite mantener controlado el buen funcionamiento de todas las partes de la aplicación para asegurar así la consistencia y la funcionalidad de la misma.

Mockito

Mockito [34] es un framework Java que permite la creación de objetos simulados, configurar su comportamiento y estado para realizar pruebas unitarias. Es decir, permite probar la funcionalidad de diversos componentes de nuestra aplicación de forma aislada y llevar a cabo la pruebas unitarias sin depender del funcionamiento de otras partes o componentes de dicha aplicación.

Capítulo 5

Seguimiento

El origen del proyecto estaba programado para el 25 de febrero del 2019, mes en el que se empieza a abordar el primer sprint. Para llevar acabo el proyecto en su totalidad, se divide el trabajo en varios sprint en los que se implementan las historias de usuario, que abarcan desde el propio acceso a la aplicación, visualizar la información sobre la detección de Design Smell y añadir nuevo contenido al conocimiento ya obtenido, como se puede ver en Product Backlog final indicado en la tabla 5.11.

5.1. Sprint 1

5.1.1. Sprint planning

En este primer Sprint se conoce el alcance del proyecto, las ideas y los objetivos principales que debemos abordar, así como las tecnologías que utilizaremos y la viabilidad del propio proyecto.

Lo primero que tenemos que hacer es la elección y posterior estudio de las herramientas necesarias para su realización. Tras este primer reto, se optó por el desarrollo de una aplicación web Java utilizando ZK Framework, Spring y JPA con la implementación de Hibernate. Para la seguridad de la web se opta por Spring Security.

Una vez elegidas las herramientas necesarias, se creará el entorno de desarrollo para la integración de ZK framewok [35], utilizando un arquetipo Maven predefinido por el propio framework, junto con Spring y JPA con Hibernate. Se crearán tres paquetes relacionados con las capas de presentación, lógica de negocio y persistencia.

Con la integración hecha, se llevará acabo el desarrollo del primer caso de uso, por lo que se hará el diseño del diagrama entidad relación correspondiente a usuarios y roles, con la herramienta Astah UML.

Con el diseño de la base de datos hecho, se crearán las entidades para usuarios y roles utilizando JPA para el mapeo relacional. Una vez creadas las entidades, se pasará a crear los servicios de persistencia para users y authorities.

Se implementará la seguridad con Spring security para realizar nuestra propia autenticación en la aplicación [36], utilizando las entidades User y Authority .

Una vez realizadas las anteriores actividades, se plantea la creación de un boceto de página de inicio de sesión con el que tener una idea inicial para diseñar con ZUML la vista correspondiente al login. Se pretende, una vez hechas las tareas anteriores, hacer una revisión de código teniendo en cuenta las posibles excepciones que puedan ocurrir.

5.1.2. Sprint backlog

Historia de usuario	Descripción	Estimación (Horas)	Estado
-	Elección y estudio de herramientas.	24	Completado
-	Configuración del entorno de desarrollo.	4	Completado
-	Integración ZK +JPA+Spring.	12	Completado
-	Arquitectura de la aplicación.	4	Completado
US-001	Crear diseño lógico de la bd para usuarios.	2	Completado
	Crear entidades User y Authority.	2	Completado
	Servicios de persistencia para User y Authority.	2	Completado
	Implementar seguridad con Spring security.	4	Completado
	Crear vista para el login.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	2	Completado

Tabla 5.1: Tareas Sprint 1

5.1.3. Sprint review y Sprint retrospective

A la hora de presentar el trabajo realizado, se rechazó el primer caso de uso, ya que no se distingue entre un tipo de usuario y otro, por lo que será una nueva tarea a realizar en el siguiente Sprint.

En cuanto al rendimiento del equipo, se observa cómo el estudio de las herramientas a utilizar ha llevado más tiempo de lo previsto, por lo que para paliar este problema se propone asignar más horas a tareas relacionadas con la búsqueda de información sobre ellas. Este problema se corresponde con el riesgo ”Retrasos en la planificación” 3.5 y el riesgo ”Falta de experiencia en el equipo” 3.9 que podrían ocurrir en la planificación del proyecto.

5.2. Sprint 2

5.2.1. Sprint Planning

En este segundo Sprint, lo primero que planificamos es la realización del caso de uso para registrar un usuario.

Se creará la vista para el registro y modificaremos los servicios de persistencia de usuarios para que permita guardarlos en el sistema. Se creará un controlador para la vista, se comprobarán las excepciones que pueda producir y se revisará el código.

Lo segundo seŕa la implementación del cierre de sesión. Para ello se creará una plantilla que esté presente en toda la aplicación y manejarlo a través de Spring security.

Otra tarea a realizar será la modificación del primer caso de uso, en el que se añadirá un menú en el que aparezcan distintas opciones para el administrador y para un usuario común.

Por otra parte, se realizará el estudio y la configuración de cómo desplegar la aplicación en Heroku [38, 39, 40].

Por último, dado que se parte de un modelo conceptual del dominio y un diseño lógico de las tablas plasmado en una base de datos MySQL, una tarea será recrear el diseño lógico de las tablas en postgreSQL por su facilidad de trabajo y compatibilidad con Heroku [41, 42].

5.2.2. Sprint backlog

Historia de usuario	Descripción	Estimación (Horas)	Estado
US-002	Crear vista para registro.	3	Completado
	Modificar servicios persistencia usuarios.	2	Completado
	Controlador para el registro de usuario.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-003	Crear template para vista global.	3	Completado
	Implementar cierre sesión con Spring security.	3	Completado
	Manejo de excepciones	1	Completado
	Revisión de código.	1	Completado
1	Añadir al template global un menú.	2	Completado
1	Controlador para el menú.	2	Completado
-	Estudio y configuración despliegue en Heroku.	12	Completado
-	Recrear estructura de bd para PostgreSQL.	12	Completado
-	Configuración de PostgresSQL de Heroku.	4	Completado

Tabla 5.2: Tareas Sprint 2

5.2.3. Sprint review y Sprint retrospective

Tras la finalización del Sprint se acepta el caso de uso de cerrar sesión.

Para el registro de un usuario, es necesario realizar cambios para que un usuario sólo tenga un apellido y que a la hora de registrarse el sistema envíe un correo de comprobación al usuario. Estas modificaciones se dejan como tareas para el siguiente Sprint.

Para el inicio de sesión también será necesario realizar cambios, ya que a la hora de iniciarla hay que especificar que aún no ha validado su cuenta mediante la validación por correo electrónico.

En cuanto a la seguridad de la web, se añadió una nueva tarea en el backlog para que todas las peticiones sean seguras mediante el protocolo https.

En cuando al desempeño del equipo, se nota que no se han aceptado las historias de usuario debido a una mala comunicación de los requisitos con el cliente que corresponde con el riesgo "Mala comunicación entre cliente y equipo" 3.9. Los retrasos para configurar el despliegue en Heroku y para el desarrollo de los controladores debido a la falta de experiencia del equipo, se corresponden con el riesgo "Retrasos en la planificación" 3.5.

5.3. Sprint 3

5.3.1. Sprint Planning

Las tareas a realizar en este tercer Sprint son la modificación del primer y el segundo caso de uso. Dado que están relacionadas, lo primero que haremos será modificar la estructura de las tablas correspondientes con el registro y el inicio de sesión para después modificar la parte de Spring security para que tenga en cuenta si un usuario está desactivado o no.

Además será necesario validar el correo electrónico de un nuevo usuario, enviándole un correo de confirmación, y una vez dado click en el enlace enviado se valide su cuenta y pueda ingresar a la aplicación normalmente.

Puesto que la historia de usuario correspondiente a la visualización es compleja por la cantidad de información disponible, se hace una descomposición para dividir el trabajo: una para la búsqueda de artículos, para la parte de autoría, publicación y caracterización, para la parte de herramientas, evidencias y smells y por último la parte de proyectos.

Para este sprint se tiene planificado la implementación de la búsqueda de artículos y mostrar información general de uno seleccionado. Para ello primero se creará la vista de la búsqueda. De esa manera cuando se seleccione un artículo se mostrará el detalle general del

mismo. Además será necesario hacer visible la vista a través del menú de la aplicación, tanto para usuarios normales como para administradores. Una vez realizado esto se crearán las entidades relacionadas con la vista y crearemos sus servicios de persistencia correspondientes, y por último manejar las excepciones que puedan ocurrir y una revisión del código.

Otra tarea será la implementación de la seguridad https para las peticiones a la web, y por último se creará un documento LaTeX, para el informe del proyecto.

5.3.2. Spring backlog

Historia de usuario	Descripción	Estimación (Horas)	Estado
US-001 y US-002	Modificar estructura bd para usuarios.	3	Completado
	Modificar el login para usuario desactivado.	3	Completado
	Validar usuario por correo electrónico.	3	Completado
US-004	Crear vista para la búsqueda de Design smells.	4	Completado
	Crear entidades relacionadas con la búsqueda.	4	Completado
	Crear Servicios de persistencia para las entidades.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
-	Seguridad https.	4	Completado
-	Preparar la documentación en Overleaf.	2	Completado

Tabla 5.3: Tareas Sprint 3

5.3.3. Sprint review y Sprint retrospective

Al finalizar el sprint se acepta el caso de uso de inicio de sesión y se rechaza el de registro de usuario, ya que es necesario que el usuario acepte o no la política de privacidad, los términos de condiciones de uso y la política de cookies para poder hacer uso de la aplicación, por ello se creará una nueva tarea para esta actividad. Además se observa que la seguridad de la web no funciona como debería, por lo que se rechaza, y se pospone su implementación para un futuro Sprint.

También se ve que la historia de usuario correspondiente a la visualización de información se puede descomponer , que se considera muy grande para implementarla en un solo sprint, ya que en este caso únicamente dio tiempo a implementar la búsqueda de artículos y mostrar un detalle general.

En cuanto al rendimiento del equipo, aún sigue habiendo problemas a la hora de entender bien los requisitos 3.8. Por este motivo se decide hacer más reuniones durante la semana en lugar de sólo una en toda la duración del sprint. Por otra parte sigue habiendo problemas a la hora de estudiar como implementar la funcionalidad de la verificación de correo y para la seguridad de las peticiones de la web, llevando mucho más tiempo de lo previsto, esto corresponde los riesgos ”Retrasos en la planificación” 3.5 y ”Falta de experiencia en el equipo” 3.9.

5.4. Sprint 4

5.4.1. Sprint Planning

En este Sprint se ha acordado modificar las historias de usuario correspondientes al registro, añadiendo la posibilidad de aceptar o no las políticas de privacidad y cookies así como los términos y condiciones de uso [43].

Se creaŕan tres vistas dedicadas a la parte legal de la aplicación, de forma que sean accesibles en toda la web. Con esto hecho se pasará a modificar la vista de registro y el controlador para que sea posible aceptar o no las políticas de privacidad, de cookies y los términos y condiciones de uso.

También se aborda la parte de visualizar en detalle la información de un artículo. En un principio se implementará la parte correspondiente a publicaciones, autoría y caracterización, además se crearán los enumerados con sus converters correspondientes [46].

En este Sprint se tomará una semana de descanso debido a la Semana Santa, por lo que la duración del mismo será de tres semanas.

5.4.2. Spring backlog

Historia de usuario	Descripción	Estimación (Horas)	Estado
US-005	Crear template para el detalle.	3	Completado
	Crear vista para detalle general.	3	Completado
	Crear vista para detalle Caracterización.	3	Completado
	Creación de Enums y converters.	3	Completado
	Crear controlador template detalle.	3	Completado
	Crear controlador para detalle general.	3	Completado
	Crear controlador detalle de caracterización.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-002	Crear vista para política de privacidad.	2	Completado
	Crear vista para política de privacidad.	2	Completado
	Crear vista para términos de uso.	2	Completado
	Modificar template general, para hacer visible las vistas en toda la aplicación.	3	Completado
	Modificar vista registro.	2	Completado
	Modificar controlador registro.	2	Completado

Tabla 5.4: Tareas Sprint 4

5.4.3. Sprint review y Sprint retrospective

Al finalizar este Sprint se aceptan los casos de uso correspondientes al registro de usuario y al de visualizar la información de publicación, autoría y caracterización.

Se observa que hay caracteres extraños en las etiquetas de internacionalización por lo que se crea una tarea para el próximo sprint, para una revisión completa de las mismas. Además se observa que los enlaces referentes a la parte legal no funcionan correctamente, por lo que se crea otra tarea para solucionar este problema en el próximo sprint.

Otro detalle que se habla con el cliente, es la pequeña capacidad de almacenar datos en la base de datos que nos ofrece Heroku, por lo que se crea una tarea para montar el servidor de base de datos en una máquina virtual de la escuela.

En cuanto al rendimiento del equipo, vemos que la investigación sobre los temas legales correspondientes a la ley de protección de datos (GDPR) [44] y los converters para los tipos enumerados han llevado más tiempo de lo previsto, por lo que se ha producido otra vez el riesgo "Retrasos en la planificación" 3.5.

5.5. Sprint 5

5.5.1. Sprint Planning

Para este Sprint se tiene planificada la descomposición de la historia de usuario correspondiente a la visualización de estadísticas sobre la detección de Design Smells. Se pretende implementar la visualización de estadísticas sobre las publicaciones.

Para ello se creará la vista correspondiente y el controlador, en el que se utilizará la librería zhighcharts para la creación de gráficos fa[45].

Por otro lado también se pretende implementar la parte de visualizar la información relacionada con evidencias, herramientas y Design smells.

Se crearán los enumerados con sus converters correspondientes, las entidades relacionadas con las vistas, la vista correspondiente para la parte de smells, vista para las evidencias y vista para la parte de herramientas y los controladores relacionados con las vistas, así como el manejo de las excepciones y una revisión del código.

Se configurará la máquina proporcionada por la escuela para que almacene la base de datos de conocimiento, haciéndolo accesible desde Heroku.

5.5.2. Spring backlog

Historia de usuario	Descripción	Estimación (Horas)	Estado
US-008	Crear vista para estadísticas de publicaciones.	2	Completado
	Crear controlador para la vista.	3	Completado
	Crear servicios de persistencia para los datos.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-006	Crear vista para smells.	3	Completado
	Crear vista para evidencias.	3	Completado
	Crear vista para herramientas.	3	Completado
	Crear entidades relacionadas con la vista.	4	Completado
	Crear enums y converters.	3	Completado
	Crear controladores para las vistas.	6	Completado
	Manejo de excepciones	1	Completado
	Revisión de código.	1	Completado
-	Revisión de links.	2	Completado
-	Revisión etiquetas internacionalización.	2	Completado
-	Configuración máquina virtual con servidor PostgreSQL en la escuela.	6	Completado

Tabla 5.5: Tareas Sprint 5

5.5.3. Sprint review y Sprint retrospective

En este Sprint se aceptan las dos historias de usuario. Sin embargo, se observa que los gráficos no son lo suficientemente correctos, por lo que se propone un cambio de librería para que se adecue más con la vista. Las tareas de revisión de enlaces y etiquetas son correctas, ya no hay ningún carácter extraño y los enlaces funcionan correctamente.

A la hora de probar la navegación de la información, se observa un error de que no encuentra los parámetros pasados de una página a otra, por lo que se crea una tarea para solucionar este problema en el siguiente Sprint.

Por otro lado, la búsqueda de artículos se ve afectada por las nuevas entidades creadas ya que la carga en en memoria es grande. Una tarea será arreglar ese retardo a la hora de buscar los artículos.

En cuanto al rendimiento del equipo, nos vemos afectados nuevamente por riesgo ”Retrasos en la planificación” 3.5. Además vemos que el software utilizado para la creación de gráficos no se adecua bien con la vista. Por ello es necesario una búsqueda de un nuevo software que no tenga ese problema, que corresponde con el riesgo ”Problemas con software de terceros” 3.7.

5.6. Sprint 6

5.6.1. Sprint Planning

Para este sprint se tiene programado el cambio de librería de gráficos y la optimización correspondiente a la búsqueda de artículos. Una vez se haya cambiado la librería de gráficos, se decide implementar todos los gráficos restantes. Se realizará la historia de usuario para visualizar proyectos. Para ello se crearán las vistas necesarias, los controladores y los servicios de persistencia. Por último se abordará el fallo obtenido en la navegación.

5.6.2. Spring backlog

Historia de usuario	Descripción	Estimación (Horas)	Estado
US-007	Crear vista para proyectos.	3	Completado
	Crear controlador para la vista.	3	Completado
	Crear servicios de persistencia.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-009	Crear vista para estadísticas de smells.	2	Completado
	Crear controlador para la vista.	3	Completado
	Modificar servicios de persistencia para smells.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-010	Crear vista para estadísticas sobre caracterización.	2	Completado
	Crear controlador para la vista.	3	Completado
	Modificar servicios de persistencia para caracterización.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-011	Crear vista para estadísticas sobre herramientas.	2	Completado
	Crear controlador para la vista.	3	Completado
	Modificar servicios de persistencia para herramientas.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-012	Crear vista para estadísticas sobre evidencias.	2	Completado
	Crear controlador para la vista.	3	Completado
	Modificar servicios de persistencia para evidencias.	3	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-013	Crear vista para estadísticas sobre factores de calidad.	2	Completado
	Crear controlador para la vista.	3	Completado
	Modificar servicios de persistencia para factores de calidad.	3	Completado
	Manejo de excepciones	1	Completado
	Revisión de código.	1	Completado
-	Cambio librería de gráficos.	3	Completado
-	Optimizar página de búsqueda	3	Completado
-	Fallo en la navegación.	3	Completado

Tabla 5.6: Tareas Sprint 6

5.6.3. Sprint review y Sprint retrospective

En este Spring se acepta la parte de visualizar la información de proyectos. Sin embargo se observa un defecto, por lo que es necesario realizar una corrección para que a la hora de visualizar un proyecto no se pierda la evidencia. Se da por aceptada la nueva librería de gráficos y una vez mostrados todos los gráficos al cliente, se observa que faltan dos gráficos, uno para la parte publicaciones y otro para caracterización.

La optimización de la página de búsqueda se acepta, ya que ahora el tiempo que tarda en realizar la carga de los artículos en Heroku es adecuado. Los fallos de la navegación se corrigen y se da por aceptado.

En cuanto al rendimiento del equipo, se observa que la nueva librería de gráficos es más fácil de utilizar por lo que los tiempos a la hora de su implementación se adecuan a los tiempo reales.

5.7. Sprint 7

5.7.1. Sprint Planning

En este sprint se toma una semana para el estudio de los exámenes correspondientes a la carrera. Se abordará la posibilidad de poder añadir una nueva contribución en el conocimiento de la aplicación, por lo que en un primer momento se ve que no es una tarea que se pueda llevar a cabo en su totalidad, por lo que se divide para hacerlo más asequible.

Por tanto se crearán distintas vistas para añadir todo el contenido relacionado con un artículo, así como todos sus controladores. Las entidades añadidas durante la implementación de la visualización sufrirán un cambio para que los identificadores tomen valores autoincrementales, al igual que las tablas de la base de datos.

Otra tarea a realizar es arreglar las conexiones seguras de la web, por lo que se harán los cambios necesarios para usar el protocolo https y la configuración de Spring, de manera que haya dos fuentes de datos: una para la base de datos principal y otra para la auxiliar [47].

Por otra parte, arreglar fallos correspondientes a la parte de visualización para que al ver la información de un proyecto relacionado con una evidencia no se pierda la referencia de esta última.

Finalmente es necesario realizar dos gráficos que no han sido cubiertos: un gráfico en las estadísticas de caracterización y otro de burbujas para las estadísticas de publicaciones.

5.7.2. Spring backlog

Historia de usuario	Descripción	Estimación (Horas)	Estado
US-014	Cambio entidades para tomar valores por defecto e identificadores autoincrementales.	3	Completado
	Cambio en la estructura de la bd, para permitir identificadores autoincrementales.	3	Completado
	Vista para añadir nueva contribución para parte de artículo, caracterización y publicación.	3	Completado
	Vista modal para añadir la autoría.	2	Completado
	Controlador para guardar nueva contribución.	4	Completado
	Controlador para vista autoría.	2	Completado
	Manejo de excepciones.	1	Completado
	Revisión de código.	1	Completado
US-015	Modificar vista para añadir nueva contribución para parte de smells, evidencias y herramientas.	5	Completado
	Cambio en las estructura de la bd, para permitir identificadores autoincrementales.	3	Completado
	Cambio entidades para tomar valores por defecto e identificadores autoincrementales.	3	Completado
	Vistas modales para añadir smells, factores de calidad, proyectos y herramientas.	6	Completado
	Controladores para vistas modales.	5	Completado
	Vista modales para añadir evidencias y medidas.	3	Completado
	Controladores para vistas modales.	2	Completado
	Modificar controlador para añadir nueva contribución.	6	Completado
	Manejo de excepciones.	2	Completado
	Revisión de código.	2	Completado
-	Arreglar conexiones seguras para la web.	4	Completado
-	Configuración Spring para tener dos fuentes de datos.	4	Completado
-	Arreglar visualización referencia de proyecto.	3	Completado
-	Añadir gráficos faltantes a publicaciones y caracterización.	3	Completado

Tabla 5.7: Tareas Sprint 7

5.7.3. Sprint review y Sprint retrospective

En este Sprint se da por aceptada la implementación de las conexiones seguras en la web y el cambio necesario para la visualización de los proyectos desde una evidencia. Se añaden los gráficos restantes en publicaciones y caracterización.

La configuración para tener dos bases de datos para la aplicación funciona correctamente. Sin embargo, las historias de usuario correspondientes a la aportación de una nueva contribución en el sistema no funcionan como deberían por lo que se rechaza y se prevé que se finalicen en un nuevo Sprint, que se tomará para la corrección de errores.

En cuanto al rendimiento del equipo vemos que la configuración y el manejo de dos bases de datos distintas, al igual que configurar las entidades para que se guarden correctamente llevan mucho más tiempo de lo previsto por la "Falta de experiencia en el equipo" 3.9.

5.8. Sprint 8

5.8.1. Sprint Planning

Para este sprint se tiene planificado arreglar la parte de añadir una nueva contribución en la base de datos. Ya que sólo tenemos un controlador para la parte de añadir, lo modificaremos para arreglar ambas historias de usuario. Se ve un fallo a la hora de registrar un usuario porque no puede conectar con el servicio de correo electrónico [48]. Se arreglarán errores percibidos a la hora de probar toda la aplicación.

La carga de datos para añadir entidades ya existentes es excesiva por lo que se propone implementar la carga de entidades y sus relaciones en modo perezoso.

5.8.2. Spring backlog

Historia de usuario	Descripción	Estimación (Horas)	Estado
14 y 15	Arreglar controlador para añadir nueva contribución.	8	Completado
-	Fallo en registro usuario.	2	Completado
-	Actualizar base de datos.	4	Completado
-	Arreglo gráfico Distribución de los enfoques, fila para Combination of Approach.	2	Completado
-	Arreglo Gráfico frecuencia de los Design Smells a nivel de clase más detectados en los proyectos seleccionados.	2	Completado
-	Arreglo para que al dar click a una colección de tipos salga un popup.	2	Completado
-	Arreglo visualización de caracteres extraños al visualizar las autorías de los artículos.	1	Completado
-	Cambio entidades y relaciones para carga perezosa	3	Completado

Tabla 5.8: Tareas Sprint 8

5.8.3. Sprint review y Sprint retrospective

En este Sprint se dan por aceptadas las historias de usuario correspondientes a guardar una nueva contribución. Por otro lado todos los errores observados durante las pruebas se solventan, dando por aceptado los cambios.

En cuanto al rendimiento del equipo vemos que no ha ocurrido ningún riesgo que pueda provocar el retraso del proyecto cumpliéndose los tiempos planificados para este sprint.

5.9. Resultados

Como hemos visto a lo largo del seguimiento del proyecto, hemos tenido constantes retrasos con la planificación de las tareas. Esta circunstancia nos ha llevado a asignar más horas a cada sprint, ya sea por el estudio en las tecnologías utilizadas o por problemas en la implementación. Además cabe destacar que incrementamos en un sprint la duración total del proyecto, por lo que habrá un aumento en las horas necesarias para su realización.

Sprints	Horas estimadas	Horas requeridas
Sprint 1	60h	66h
Sprint 2	50h	54h
Sprint 3	30h	34h
Sprint 4	36h	40h
Sprint 5	44h	46h
Sprint 6	70h	70h
Sprint 7	70h	76h
Sprint 8	22h	22h
Total	**360h**	**408h**

Tabla 5.9: Horas estimadas vs horas reales.

Como se puede observar, algunas tareas en los sprint se alargaron más de lo previsto debido sobre todo a la implementación de la funcionalidad, condicionada por la falta de estudio y experiencia en las tecnologías utilizadas para la misma. Esto lleva a que el desarrollo del proyecto aumente unas 48 horas.

5.9.1. Costes reales

En el apartado 3.4 se especifica el análisis de coste inicial para el proyecto. Dado que han surgido cambios en estas primeras estimaciones, en este apartado vamos a indicar la diferencia entre los costes estimados frente a los reales.

En cuanto a los costes de hardware, utilizaremos un ordenador de sobremesa valorado en 1000€ y una monitor valorado en 152 €. Aplicando el coeficiente de amortización máximo para **Equipos para procesos de información** es del 25 % y teniendo en cuenta la duración real de 16 semanas (4 meses), contaremos la amortización de cada producto.

Producto	Precio	Amortización
Ordenador sobremesa	1000 €	83,33 €
Monitor	152 €	12,66 €
Total	1152 €	100 €

Tabla 5.10: Costes hardware.

Por otro lado los costes referentes a recursos humanos aumentarían en cuanto a las horas requeridas para la finalización de las tareas, por lo tanto el sobrecoste por horas será de 48 horas * 11,95€ = 478 €. Por lo tanto si sumamos el sobrecoste y el coste del hardware al coste inicial , tenemos un total de 478 € + 100 € + 4302 € = 4880 €.

5.9.2. Product backlog final

Como se ha podido observar durante el seguimiento del proyecto hemos tenido que refinar las historias de usuario, puesto que la complejidad de las mismas permiten una descomposición para poder dividir el trabajo y poder implementarla por partes, por lo que nos da como resultado un product backlog como el siguiente.

Historia de usuario	Descipción	Prioridad	Estimación
US-001	Como un usuario registrado, quiero iniciar una sesión, para poder utilizar la aplicación.	1	3
US-002	Como un usuario no registrado, quiero registrarme en el sistema, para poder utilizar la aplicación.	2	2
US-003	Como un usuario identificado, quiero cerrar sesión, para poder salir de la aplicación.	3	1
US-004	Como un usuario, quiero hacer una búsqueda de artículos sobre la detección de Design Smells, para estar informado sobre la detección.	4	3
US-005	Como un usuario, quiero visualizar en detalle un artículo (Publicación, Autoría y caracterización), para estar informado sobre la detección.	5	3
US-008	Como un usuario, visualizar estadísticas referentes a publicaciones, para estar informado sobre las publicaciones utilizadas en el estudio.	6	3
US-006	Como un usuario, quiero visualizar en detalle un artículo (Smells, Evidencias y herramientas), para estar informado sobre la detección.	7	3
US-007	Como un usuario, quiero visualizar en detalle un artículo (Proyectos), para estar informado sobre la detección.	8	2
US-009	Como un usuario, quiero visualizar estadísticas referentes a Design smells, para estar informado sobre los Design smells detectados en los artículos.	9	2
US-010	Como un usuario, quiero visualizar estadísticas referentes a enfoques, para estar informado sobre los enfoques utilizados en los artículos.	10	2
US-011	Como un usuario, quiero visualizar estadísticas referentes a herramientas, para estar informado sobre los herramientas utilizadas en los artículos.	11	2
US-012	Como un usuario, quiero visualizar estadísticas referentes a evidencias, para estar informado sobre la relación entre evidencias, proyectos y medidas.	12	2
US-013	Como un usuario, quiero visualizar estadísticas referentes a factores de calidad, para estar informado sobre la relación entre factores de calidad y Design smell.	13	2
US-014	Como un usuario, quiero contribuir con un nuevo elemento al conocimiento (Publicación, Autoría y caracterización), para aumentar el conocimiento sobre la detección de Design smells.	14	3
US-015	Como un usuario, quiero contribuir con un nuevo elemento al conocimiento (Smells, Evidencias y herramientas), para aumentar el conocimiento sobre la detección de Design smells.	15	3

Tabla 5.11: Product backlog final.

Capítulo 6

Diseño

En este apartado se mostrará el modelo conceptual de usuarios de la aplicación. Se proporciona el modelo lógico de la base de datos para la detección y usuarios, con una explicación de qué contiene cada tabla y los tipos de datos que almacena. Se mostrará una visión general de la arquitectura de la aplicación, los distintos frameworks utilizados, los diagramas de secuencia para el acceso y salvado de datos, la descomposición modular y dependencias. Además se indica cómo quedará desplegada la aplicación en diferentes nodos computacionales.

6.1. Modelo conceptual

Ya que se nos proporciona un modelo de dominio para la detección de Design Smells, se ha añadido una parte especifica para los usuarios de la aplicación como vemos en la figura 6.1. El desarrollo de la aplicación web, parte del modelo conceptual proporcionado 2.1 y de la parte de usuarios, con los que iremos creando las entidades correspondientes al dominio.

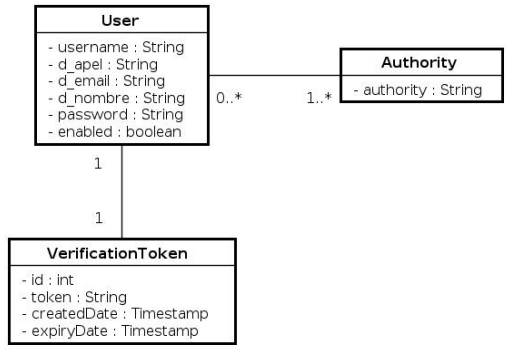

Figura 6.1: Modelo conceptual para usuarios.

6.2. Diseño lógico

Una vez utilizadas las técnicas de modelado para la obtención del modelo conceptual, los autores crean un modelo lógico de datos basado en este modelo para poder almacenar toda la información correspondiente a la detección de Design Smells. Esto lo podemos observar en la figura 6.2 y por otra parte lo que se ha hecho para los usuarios de la aplicación, como se puede ver en la figura 6.3 .

6.2.1. Detección de Design Smells

Tabla author

En esta tabla se guarda la información de los autores. El campo *surname*, *name* y *email* permiten almacenar 255 caracteres alfanuméricos, no permiten nulos y por defecto toman el valor 'NO DATA'.

Tabla institution

En esta tabla se guarda la información de las instituciones y si son o no académicas. Los campos *fullname*, *acronym*, *country* permiten almacenar 255 caracteres alfanuméricos, no permiten nulos y por defecto toman el valor 'NO DATA'. El campo *isacademic* permite almacenar números enteros de 2 bytes, no permite nulos y por defecto toma el valor '0'.

Tabla authorinstitution

Permite almacenar la autoría de los artículos. Los campos *article_id*, *author_id*, *institution_id* son claves foráneas que hacen referencias a las tablas *article*, *author* e *institution* respectivamente. Asimismo compondrán la clave primaria, puesto que identificarán entre las tres la autoría de un artículo.

Tabla article

En esta tabla se guarda información referente al artículo. El campo *title*, permite almacenar 255 caracteres alfanuméricos, no permite nulos y por defecto toma el valor 'NO DATA'. El campo *year* permite guardar números enteros de 4 bytes y por defecto toma el valor 'NULL'. Los campos *publicationtype_id*, *publication_id*, *approach_id* son llaves foráneas que hacen referencia a las tablas *publication*, *publicationtype* y *approach* respectivamente.

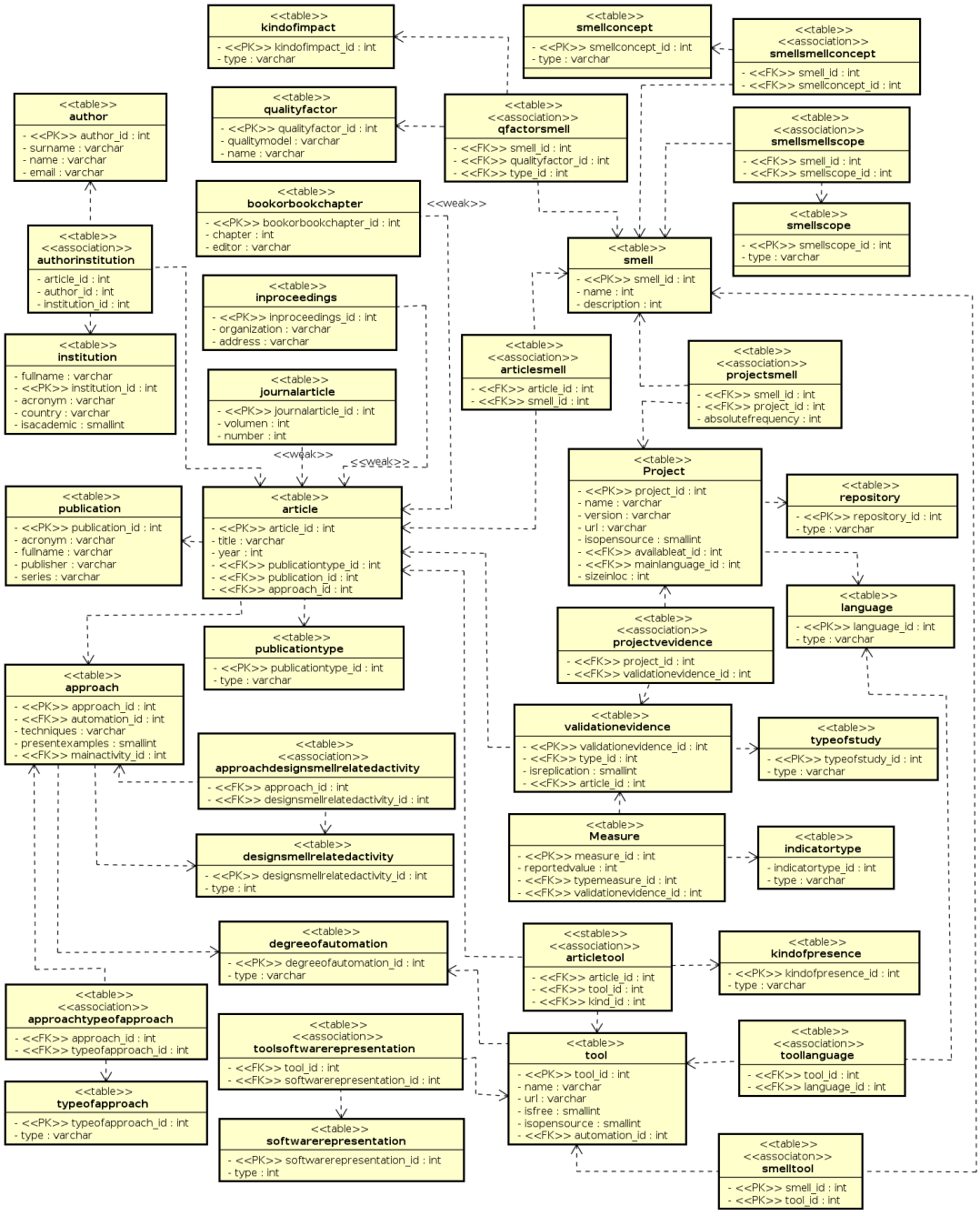

Figura 6.2: Modelo lógico del conocimiento sobre la detección de Design smell.

Tabla bookorbookchapter

Almacena información del libro o capítulo del libro si la publicación es de tipo BookChapter. El campo *chapter* permite almacenar enteros de 4 bytes y por defecto toma el valor 'NULL'. El campo *editor* permite almacenar 255 caracteres alfanuméricos, no permite nulos y por defecto toma el valor 'NO DATA'.

Tabla inproceedings

Almacena información sobre el acta de procedimiento si la publicación es de tipo Conference, Workshop o Symposium. Los campos *organization* y *address* permiten almacenar 255 caracteres alfanuméricos, no permiten nulos y por defecto toman el valor 'NO DATA.'

Tabla journalarticle

Almacena información sobre la revista si la publicación es de tipo Journal. Los campos *volumen* y *number* permiten almacenar enteros de 4 bytes y por defecto toman el valor 'NULL'.

Tabla publication

En esta tabla se guarda información referente a la publicación del artículo. Los campos *acronym*, *fullname*, *publisher* y *series* permiten guardar 255 caracteres alfanuméricos, no permiten nulos y por defecto tienen el valor 'NO DATA'.

Tabla publicationtype

En esta tabla se guarda información referente al tipo de publicación del artículo (conference, journal, workshop, symposium, chapter in a book). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla approach

En esta tabla se guarda información sobre el enfoque de detección, el grado de automatización, las técnicas de detección (algoritmos, heurísticas, estrategias), así como la actividad principal relacionadas con la detección (especificación, detección, etc.) y si presenta o no

ejemplos. El campo *techniques* permite almacenar 255 caracteres alfanuméricos, no admite nulos y por defecto toma el valor 'NO DATA'. El campo *presentexamples* permite almacenar enteros de 2 bytes y toma por defecto el valor '0'. Los campos *automation_id*, *mainactivity_id* son llaves foráneas que hacen referencia a las tablas *degreeofautomation* y *designsmellrelatedactivity*.

Tabla degreeofautomation

Almacena información sobre el grado de automatización de herramientas y enfoques (manual, asistido por computadora, semi automático y totalmente automático). El campo *type* spermite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla designsmellrelatedactivity

Almacena información sobre las actividades relacionadas con los Design smell (especificación, detección, corrección, visualización, priorización y análisis de impacto) desarrolladas en los trabajos seleccionados en el estudio. El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla approachdesignsmellrelatedactivity

Almacena información sobre otras actividades relacionadas con el enfoque. Los campos *approach_id* y *designsmellrelatedactivity_id* son claves foráneas que hacen referencia a las tablas *approach* y *designsmellrelatedactivity*. Así mismo compondrán la clave primaria puesto que identificarán que otras actividades estén relacionadas con el enfoque.

Tabla typeofapproach

Almacena información sobre los tipos de enfoques utilizados en los trabajos seleccionados en el estudio (basado en métricas, lógicos, basados en reglas, basados en aprendizaje, etc). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla approachtypeofapproach

Almacena información sobre los tipos de enfoques que se utilizan en un artículo. Los campos *approach_id* y *typeofapproach_id* son llaves foráneas que hacen referencia a las tablas

approach y *typeofapproach*. Asimismo compondrán la clave primaria puesto que identificarán qué tipos de enfoques están relacionadas con el enfoque de un artículo.

Tabla smell

Almacena la información sobre los Design Smells detectados en los trabajos seleccionados. El campo *name* permite almacenar 255 caracteres alfanuméricos, no permite nulos y por defecto toma el valor 'NO DATA'. El campo *description* no tiene límite de caracteres, no permite nulos y por defecto toma el valor 'NO DATA'.

Tabla qualityfactor

Almacena información sobre los factores de calidad detectados en los trabajos seleccionados (mantenibilidad, confiabilidad, etc) y los modelos de calidad (ISO / IEC 9126, Modelo de especificación de prueba, Modelo FURPS y Modelo de factor McCalls). Los campos *name* y *qualitymodel* permiten al almacenar 255 caracteres alfanuméricos, no permiten nulos y por defecto toman el valor 'NO DATA'.

Tabla kindofimpact

Almacena información sobre los tipos de impacto detectados en los trabajos seleccionados (positivo, negativo, sin impacto o impacto desconocido). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla smellconcept

Almacena información sobre los términos que se utilizan para referirse a los Design Smell en los trabajos seleccionados (Antipattern, Code Smell, Bad Smell, etc). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla smellscope

Almacenan información sobre los distintos alcances a nivel de código que tienen los Design Smell en los trabajos seleccionados (sistema, subsistema, método, operación, etc). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla qfactorsmell

Almacena información sobre el impacto que tienen los Design Smell en los factores de calidad. Los campos *smell_id*, *qualityfactor_id* y *type_id* son claves foráneas que hacen referencia a las tablas *smell*, *qualityfactor* y *kindofimpact*. Asimismo compondrán la clave primaria, puesto que identificarán qué Design Smells tienen un impacto sobre determinados factores de calidad.

Tabla smellsmellconcept

Almacena información sobre los términos utilizados para referirse a un Design Smell. Los campos *smell_id* y *smellconcept_id* son llaves foráneas que hacen referencia a las tablas *smell* y *smellconcept*. Asimismo compondrán la clave primaria puesto que identificarán con qué término se refieren al mencionar a un Design Smell.

Tabla smellsmellscope

Almacena información sobre el alcance que tiene un Design smell en el código. Los campos *smell_id* y *smellscope_id* son llaves foráneas que hacen referencia a las tablas *smell* y *smellconcept*. Asimismo compondrán la clave primaria, puesto que identificarán qué alcance tiene un Design smell en el código fuente.

Tabla tool

Almacena información de las herramientas mencionadas en los trabajos seleccionados para el estudio. Los campos *name* y *url* permiten almacenar 255 caracteres alfanuméricos, no permiten nulos y por defecto toman el valor 'NO DATA'. Los campos *isfree* y *isopensource* permiten almacenar enteros de 2 bytes, no permiten nulos y por defecto toman el valor '0'. El campo *automation_id* es clave foránea hace referencia a la tabla *degreeofautomation*.

Tabla kindofpresence

Almacena información sobre la caracterización que tienen las herramientas mencionadas en los artículos (presentar la herramienta por primera vez, mejorar una existente, compararla con otras herramientas y revisar alguna en profundidad). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla language

Almacena información sobre los lenguajes de programación que las herramientas pueden soportar y los lenguajes con los que se implementan los proyectos (Java, C++, C#, C, Ruby, JavaScript, etc.). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla softwarerepresentation

Almacena información sobre las representaciones internas para las herramientas presentes en los trabajos seleccionados (Graph, AST, Object Model, etc.). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla articletool

Almacena información sobre las herramientas mencionadas y su caracterización en un articulo. Los campos article_id, tool_id y kind_id son claves foráneas que hacen referencia a las tablas article, tool y kindofpresence. Asimismo compondrán la clave primaria, puesto que identificarán las herramientas mencionadas en un artículo y la caracterización que tienen en él.

Tabla smelltool

Almacena información de los Design smell que pueden detectar las herramientas. Los campos *smell_id* y *tool_id* son claves foráneas que hacen referencia a las tablas *smell* y *tool*. Asimismo compondrán la clave primaria puesto que identificarán los Design Smell que pueden detectar las herramientas y viceversa.

Tabla toollanguage

Almacena información de los lenguajes que soportan las herramientas. Los campos *tool_id*, *language_id* son claves foráneas que hacen referencia a las tablas *tool* y *language*. Asimismo compondrán la clave primaria puesto que identificarán los lenguajes de programación que pueden soportar las herramientas.

Tabla toolsoftwarerepresentation

Almacena información de la representaciones internas de las herramientas. Los campos *tool_id* y *softwarerepresentation_id* son claves foráneas que hacen referencia a las tablas *tool* y *softwarerepresentation*. Asimismo compondrán la clave primaria, puesto que identificarán los representaciones internas que tienen las herramientas.

Tabla validationevidence

Almacena información sobre las evidencias de validación presentes en un artículo. El campo *isreplication* permite almacenar números enteros de 2 bytes, no permite nulos y por defecto toma el valor '0'. Los campos *type_id* y *article_id* son claves foráneas que hacen referencias a las tablas *typeofstudy* y *article*.

Tabla typeofstudy

Almacena información sobre el tipo de estudio realizado en la validación en los trabajos seleccionados (un estudio de caso, una encuesta, un experimento, etc.). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla measure

Almacena la medidas de calidad de las evidencias presentes en un artículo. El campo *reportedvalue* permite almacenar números reales y de doble precisión, y por defecto toma el valor 'NULL'. Los campos *typemeasure_id* y *validationevidence_id* son claves foráneas que hacen referencia a las tablas *indicatortype* y *validationevidence* respectivamente.

Tabla indicatortype

Almacena los indicadores de calidad asociados a las medidas presentes en los trabajos seleccionados (precisión, recuperación, falso Positivo (FP), falso Negativo (FN), etc). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

Tabla project

Almacena información sobre los proyectos utilizados en la validación. Los campos *name*, *version* y *url* permiten almacenar 255 caracteres alfanuméricos, no permiten nulos y por defecto toman el valor 'NO DATA'. El campo *isopensource* permite almacenar números enteros de 2 bytes, no permite nulos y por defecto toma el valor '0'. Los campos *sizeinloc* y *numberofclasses* permiten almacenar enteros de 4 bytes y por defecto toman el valor 'NULL'. Los campos *availableat_id* y *mainlanguage_id* son claves foráneas que hacen referencia a las tablas *repository* y *language* respectivamente.

Tabla projectsmell

Almacena información sobre los Design smell detectados en los proyectos utilizados en la validación y la frecuencia con la que aparecen. El campo *absolutefrequency* permite almacenar números enteros de 4 bytes y no permite nulos. Los campos *smell_id* y *project_id* son claves foráneas que hacen referencia a las tablas *smell* y *project* respectivamente. Asimismo todos los campos compondrán la clave primaria, puesto que identificarán la frecuencia con la que aparecen los Design Smell en los proyectos mencionados.

Tabla projectvevidence

Almacena información sobre los proyectos utilizados en la validación. Los campos *project_id* y *validationevidence_id* claves foráneas que hacen referencia a las tablas *project* y *validationevidence* respectivamente. Asimismo compondrán la clave primaria, puesto que identifican qué proyectos son usados en la validación.

Tabla repository

Almacena información sobre los repositorios de los proyectos mencionados (SourceForge, GitHub,GitLab, etc.). El campo *type* permite almacenar 255 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

6.2.2. Usuarios

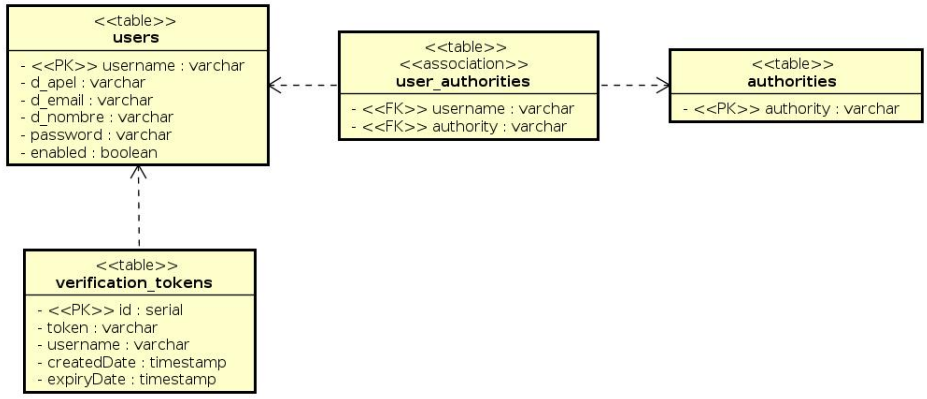

Figura 6.3: Modelo lógico para usuarios.

Tabla users

Almacena información sobre los usuarios que utilizan la aplicación web. Los campos *username*, *d_apel*, *d_email* y *password* permiten almacenar 50 caracteres alfanuméricos y no permiten nulos. El campo *enabled* es de tipo boolean y no permite nulos.

Tabla verification_tokens

Almacena información sobre el token único asignado a un usuario, junto con la fecha de vencimiento, para poder verificar el registro mediante correo electrónico. El campo *token* permite almacenar 36 caracteres alfanuméricos y no permite nulos. Los campos *createdDate* y *expiryDate* son de tipo timestamp y no permiten nulos. El campo *username* es clave foránea que hace referencia a la tabla users.

Tabla user_authorities

Almacena información sobre los roles asignados a los usuarios. Los campos *username* y *authority* hacen referencia a las tablas *users* y *authorities* respectivamente. Asimismo componen la clave primaria, puesto que identifican qué roles tiene un usuario en la aplicación.

Tabla authorities

Almacena información sobre el rol que tiene el usuario en la aplicación (ROLE_ADMIN, ROLE_USER). El campo *authority* permite almacenar 50 caracteres alfanuméricos, no permite nulos y se define como 'UNIQUE'.

6.2.3. Conclusión

Una vez obtenido el diseño de la base de datos relacional, el siguiente paso que se realiza es hacer un diseño físico de la base de datos en MySQL, que se tuvo que migrar a PostgreSQL para una mejor compatibilidad con Heroku. Sin embargo, como la capacidad que ofrecía Heroku en su versión gratuita para PostgreSQL no era capaz de soportar la cantidad de registros que tenia el conocimiento proporcionado, se decide montar una máquina virtual en los servidores de la escuela para poder almacenar todos los datos y que no haya problemas con el número de registros permitidos en Heroku.

Una vez establecida la base de datos principal, pasamos a analizar la base de datos auxiliar para almacenar nuevos datos proporcionados por los usuarios, que quedarían pendientes de aprobación por parte de un administrador del sistema. En un principio, se decide utilizar la misma estructura para almacenar los nuevos datos y una vez almacenados mostrarlos a los administradores para que puedan aprobar o no los nuevos datos introducidos. Una vez aceptados los datos, se deberían pasar a la base de datos principal para que formen parte del conocimiento.

Sin embargo, al hacer un análisis sobre la opción de tener los datos en otra base de datos con la misma estructura, vemos que no es viable. Por un lado no sabemos qué datos son nuevos y cuáles no, puesto que replicaríamos la información en ambas bases de datos para no perder las referencias de datos ya asociados. Por otro lado, las claves primarias de la base de datos auxiliar deberían ser las mismas que en la base de datos secundaria, tanto para nuevos datos como para los ya existentes.

Debido a esto se propone modificar la estructura de la base de datos para que las tablas tengan un nuevo campo boolean que indique si es o no un nuevo dato. También que todas las claves primarias se incrementen automáticamente al añadir un nuevo dato y al pasar a la base de datos principal conserven las mismas claves para mantener la integridad referencial.

Estos cambios se proponen al tutor del proyecto dando como respuesta una negativa, ya que el costo en tiempo para una modificación y su posterior implementación no es viable. Esto implicaría crear nuevas entidades en el dominio para la nueva base de datos ya que necesitarían un nuevo atributo boolean para el campo que indica si hay o no cambios, por lo que se deja en el Icebox como una historia de usuario a realizar.

6.3. Arquitectura

6.3.1. Visión general

El modelo arquitectónico de la aplicación web es de tipo cliente-servidor y se basa en un patrón arquitectónico de tres niveles que corresponden con la forma en que las capas lógicas (lógica de presentación, lógica de la aplicación y la lógica de acceso a datos) se encuentran distribuidas de forma física.

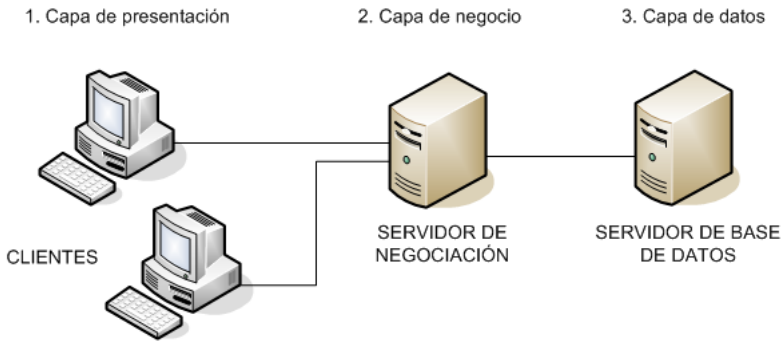

Figura 6.4: Modelo arquitectónico. Tomado de Wikipedia [55].

La arquitectura propuesta a nivel lógico sigue un patrón arquitectónico de tres capas en la que cada capa tiene dependencias con la capa inmediatamente inferior y cada una de ellas mantiene una relación de dependencia con las entidades del modelo de dominio como se puede ver en la figura 6.5.

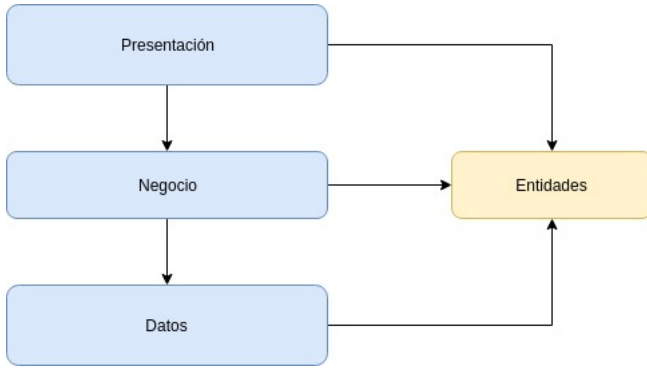

Figura 6.5: Esquema arquitectura en 3 capas de la aplicación.

Por lo tanto tendremos una arquitectura de tres niveles y tres capas para la aplicación.

Gracias a este patrón arquitectónico en capas, reducimos el acoplamiento entre componentes de la aplicación, ya que cada capa se encarga de operaciones especificas mejorando así la flexibilidad, el mantenimiento y la escalabilidad. Una vez indicadas las capas de la aplicación vemos qué frameworks se adecúan más a cada una de estas.

Para la capa de presentación se utiliza ZK Framework que nos permite crear interfaces de usuario altamente interactivas y receptivas en Java puro, proporcionándonos cientos de componentes con los que poder diseñar la parte visual de la web. ZK nos permite utilizar el enfoque MVC (Model-View-Controller) que permite manejar los componentes directamente y el enfoque MVVM (Model-View-ViewModel) que desacopla el controlador y la vista.

Elegimos el patrón arquitectónico MVC, ya que será un controlador el encargado de hacer de intermediario entre la vista y el modelo, mostrando la información del modelo así como los cambios que sufre tras la interacción del usuario con la vista.

Para la capa de lógica de negocio utilizaremos Spring framework ya que permite definir servicios correspondientes que servirán de intermediarios entre la capa de presentación y la capa de datos. Estos servicios no tendrán demasiada complejidad debido a que los requerimientos que se han abordado no han resultado difíciles. Sin embargo, en esta capa deben residir todas las operaciones correspondientes al manejo del dominio de la aplicación.

Para la capa de datos utilizaremos JPA (Hibernate) ya que permite un mapeo objeto relacional con el que podremos acceder a la base de datos PostgreSQL. Utilizaremos el Patŕon DAO de acceso a datos para encapsular todas las operaciones relacionadas con la base de datos y exponer la interfaz DAO para los servicios de la capa de negocio y así realizar operaciones de persistencia relacionadas con una entidad persistente particular.

Como resultado de la integración de ZK, Spring y JPA (hibernate) nos da una pila de aplicaciones en la que también se muestra Webapp Runner que permite iniciar una aplicación en un contenedor Tomcat, como se muestra en la siguiente figura 6.6.

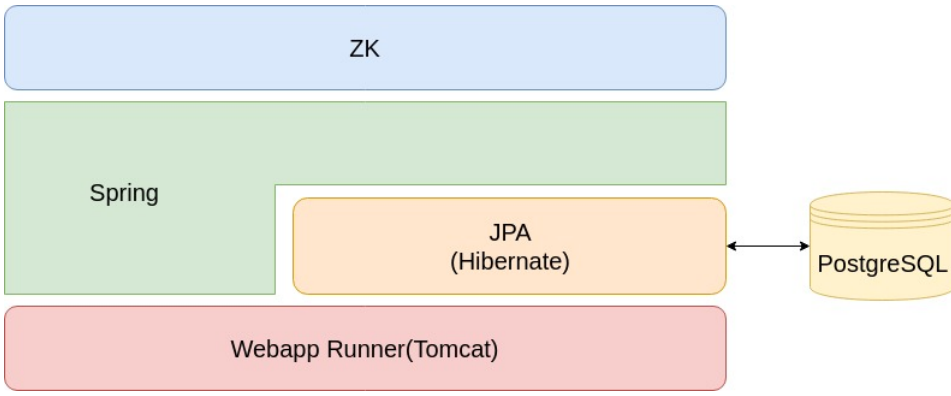

Figura 6.6: Pila de aplicaciones de Smellwisdom. Tomado de zkoss [56].

6.3.2. ZK Framework

ZK es un framework desarrollado por Potix Corporation. ZK permite crear aplicaciones web AJAX altamente interactivas y receptivas en Java puro, proporcionando cientos de componentes en un lenguaje con formato XML,ZUL. Todas las acciones de un usuario en una página se pueden manejar en un controlador y los cambios se visualizan automáticamente en el navegador. No necesitamos preocuparnos por los detalles de la comunicación entre el navegador y el servidor, ZK se encargará de la comunicación AJAX por nosotros.

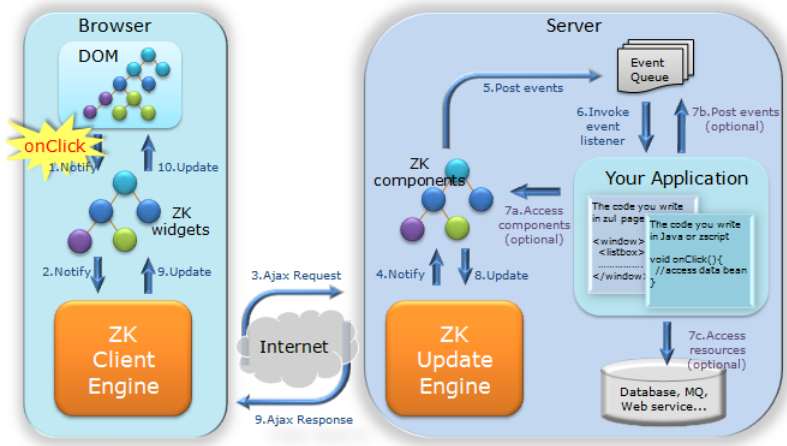

Figura 6.7: Arquitectura de ZK. Tomado de zkoss [57]

.

Cuando un navegador visita una página, ZK crea los componentes escritos en ZUL y los presenta en el navegador en una página HTML. Se pueden manipular componentes mediante un controlador para implementar la lógica de presentación de la interfaz de usuario. Todos los cambios que se realizan en los componentes se reflejarán automáticamente en el navegador y ZK se encargará de la comunicación entre el navegador y el servidor.

ZK tiene una arquitectura fusión de servidor + cliente, se puede acceder fácilmente a la pila de tecnología Java EE e integrar muchos frameworks Java de terceros como Spring o Hibernate. Además, ZK también admite el desarrollo centrado en el cliente que le permite personalizar el efecto visual o manejar las acciones del usuario en el lado del cliente.

ZK proporciona una amplia documentación que podemos encontrar aquí [49].

6.3.3. Spring Framework

Spring es un framework desarrollado por Pivotal Software. Spring proporciona una infraestructura de soporte para desarrollar aplicaciones Java. Spring se divide en varios módulos dedicados a diferentes ámbitos dentro de una aplicación, como podemos observar en la Figura 6.8

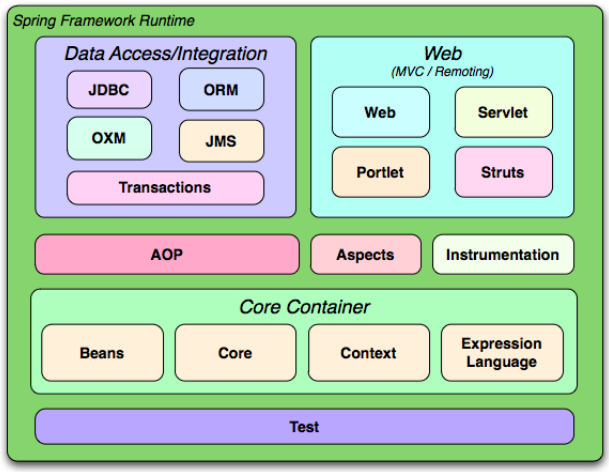

Figura 6.8: Arquitectura de Spring. Tomado de docs.spring.io [58]

Spring utiliza la inversión de control (IoC), un patrón de diseño originalmente expuesto por Martin Fowler. Es el framework el que invocará el código necesario como respuesta a eventos, y se encargará de la gestión de instancias en vez de ser la aplicación la que cree los nuevos objetos.

Spring posee un contenedor de objetos POJO, denominados Beans, que inyecta en cada objeto las instancias necesarias según la configuración definida. Para poder conectar los beans el contenedor necesita de una configuración para llevar a cabo su tarea. Esta configuración se puede definir de tres formas diferentes: mediante archivos de configuración de contexto en XML, o mediante el uso de anotaciones de las librerías del framework, o en objetos Java.

De la gestión de instancias realizada por Spring, aparece la denominada inyección de dependencias. Se trata de un patrón de diseño en el que básicamente la clase contenedora es la encargada de inyectar los objetos en una clase, en lugar de ser la clase la encargada de crear dichos objetos.

Spring facilita el acceso a datos y gestiona las transacciones con la base de datos. A su vez, maneja automáticamente las operaciones que terminan siempre con un resultado de éxito o de error, evitando que la base de datos quede en un estado inconsistente.

Spring proporciona una amplia documentación que podemos encontrar aquí [50].

6.3.4. JPA (Hibernate)

Java Persistence API (JPA) es una especificación de persistencia que ofrece un conjunto de interfaces y APIs para resolver el problema del almacenamiento de objetos en una base de datos relacional. En este caso usaremos una base de datos PostgreSQL e implementaremos una capa de persistencia utilizando el patrón DAO (Objeto de acceso a datos) para encapsular todas las operaciones relacionadas con la base de datos. De forma muy simple, una tabla se mapea contra una clase, y cada columna contra un atributo de dicha clase. Usaremos anotaciones de JPA para indicar estas relaciones. De esta forma, la implementación con JPA se encargará de ocultar la complejidad del acceso a datos, exponiendo solamente objetos.

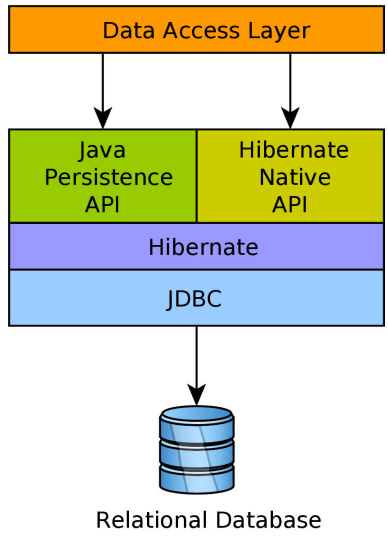

Figura 6.9: Arquitectura de JPA/Hibernate. Tomado de docs.jboss.org [59]

.

Utilizaremos Hibernate como ORM que implementa JPA, como usaremos las interfaces de JPA no habrá ninguna referencia directa en nuestro código a Hibernate, como podemos ver en la figura 6.9, se pueden utilizar las dos APIs para el acceso a datos pero nosotros sólo utilizaremos la de JPA. Como dijimos en el apartado anterior seguimos usando el framework Spring y lo integraremos con JPA.

Hibernate proporciona una amplia documentación para utilizar JPA que podemos encontrar aquí [51].

6.3.5. Patrón MVC

Como indicamos Anteriormente ZK permite utilizar el patrón Modelo-Vista-Controlador (MVC), para la parte de presentación de la aplicación. Según este patrón, la vista queda aislada del manejo del modelo, así como la gestión de eventos producidos por la interacción del usuario y las peticiones contra el servidor.

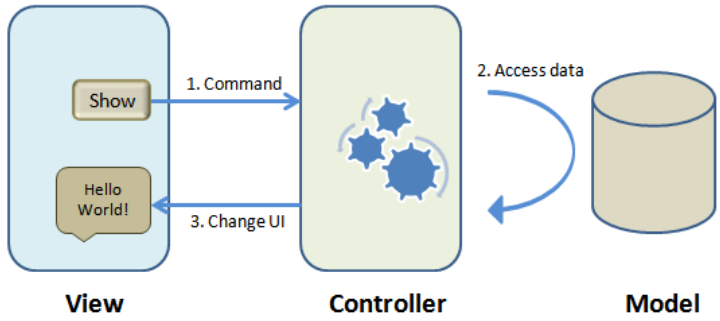

Figura 6.10: Patrón MVC. Tomado de zkoss [60]

La vista es la encargada de la representación visual de los datos y la interacción del usuario, que desencadena eventos que se enviarán al controlador. El controlador se encargará de coordinar la vista y el modelo, manejando los eventos recibidos de la vista para actualizar y recuperar datos para cambiar la vista. El modelo se encarga de los datos y de las reglas de negocio.

En ZK definimos controladores extendiendo de la clase SelectorComposer, que permite conectar componentes, variables y escuchadores de eventos automáticamente utilizando anotaciones como @Wire (que permite conectar con los componentes de la vista), @WireVariable (permite conectar con los beans gestionados por Spring) y @Listen (permite conectar un detector de eventos).

Para más información de cómo aplicar el patrón MVC con ZK podemos ver la siguiente documentación [52].

6.3.6. Patrón DAO

El patrón de objeto de acceso a datos (DAO) es una buena práctica para implementar una capa de persistencia. Este patrón encapsula la API de persistencia en un objeto DAO y expone la interfaz del objeto DAO al objeto de negocio para realizar operaciones de persistencia relacionadas con una entidad persistente particular como podemos ver en la figura 6.11.

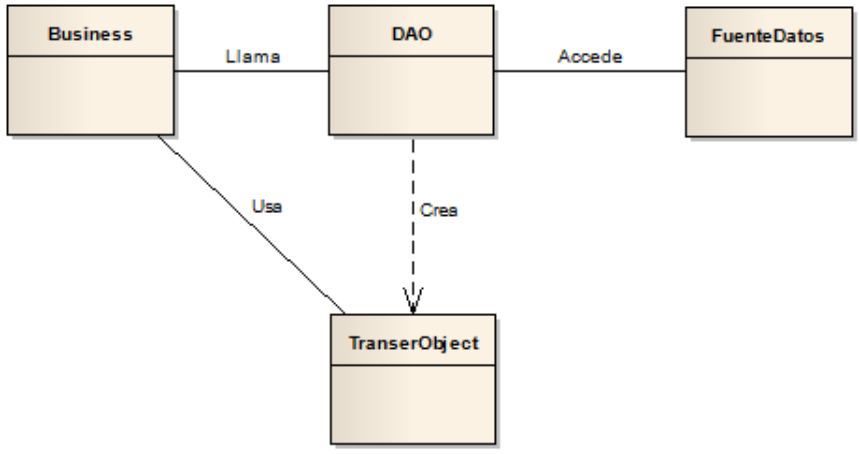

Figura 6.11: Patrón DAO, de rinconprogramadorcito.blogspot [61]

Debido a la integración de Spring con JPA, utilizamos la inyección de dependencia OpenEntityManagerInViewFilter de Spring con la que podemos aplicar el patrón entitymanager-por-solicitud. En la clase DAO, podemos recuperar fácilmente EntityManager mediante la inyección de dependencia de Spring. La gestión de transacciones declarativas y la regla de rollback de Spring también reducen nuestro trabajo.

Para más información de cómo aplicar el patrón DAO junto con Spring podemos ver la siguiente documentación [53].

6.4. Descomposición modular y dependencias

En esta sección veremos la estructura de paquetes de la aplicación como se puede observar en la figura 6.12. Muestra la organización del código y las dependencias de los mismos con respecto a otros paquetes.

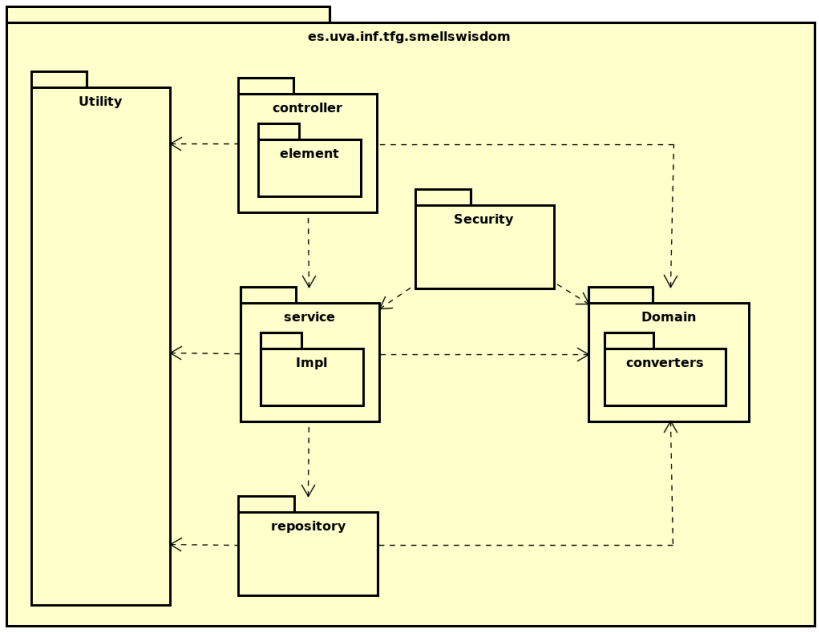

Figura 6.12: Estructura de paquetes.

- El paquete controller, contiene todos los controladores referentes a la parte de visualización de información, de acceso y registro a la aplicación. El paquete controller.element contiene los controladores específicos para añadir una nueva contribución.
- El paquete security, contiene las clases necesarias para implementar la seguridad bajo Spring Security.
- El paquete service, contiene todas las interfaces correspondientes a la capa de negocio. El paquete service.impl contiene las clases que implementan estas interfaces.
- El paquete repository, contiene todos las clases DAO de persistencia de datos.
- El paquete domain, contiene todas la clases de entidad. El paquete domain.converters contiene los convertidores para las tablas referentes a los tipos de los enumerados.

6.5. Modelos Dinámicos

En esta sección se incluyen los diagramas de secuencia, que reflejan el comportamiento de la aplicación en cuanto al flujo de operaciones. Para ello nos centraremos en dos casos de uso: uno para la inserción de nuevos datos y otro para la visualización de información.

Los diagramas que se ven en las figuras 6.13 6.14 y 6.15, muestran el escenario de la historia de usuario sobre el registro en la aplicación. En este caso se mostrará la secuencia de cuando un usuario se registra con éxito, quedando desactivado el nuevo usuario hasta que verifique su cuenta mediante correo electrónico. Además, mostrarán cómo se verifica la entrada que hace el usuario, cómo las peticiones llegan a la capa de servicios y posteriormente cómo esta hace peticiones a la capa de persistencia para operaciones de lectura y escritura.

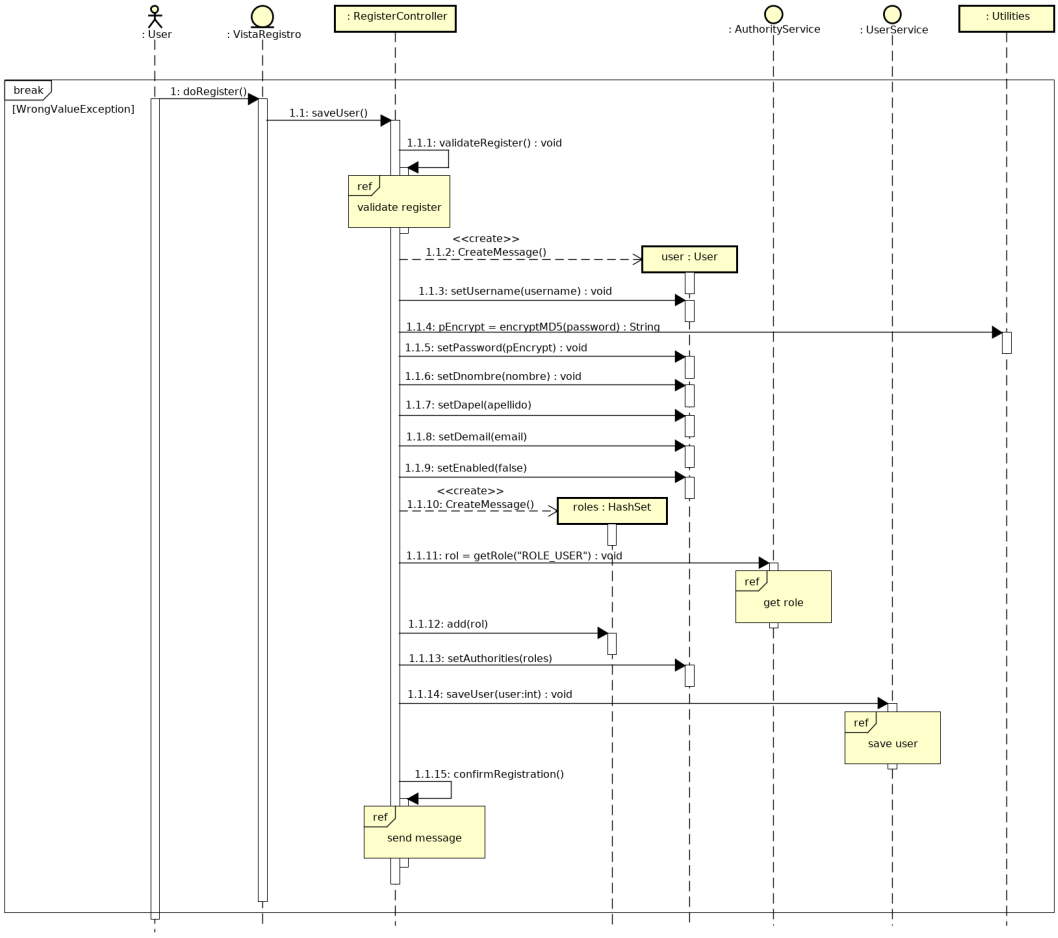

Figura 6.13: Diagrama de secuencia CU “Registro de usuario”.

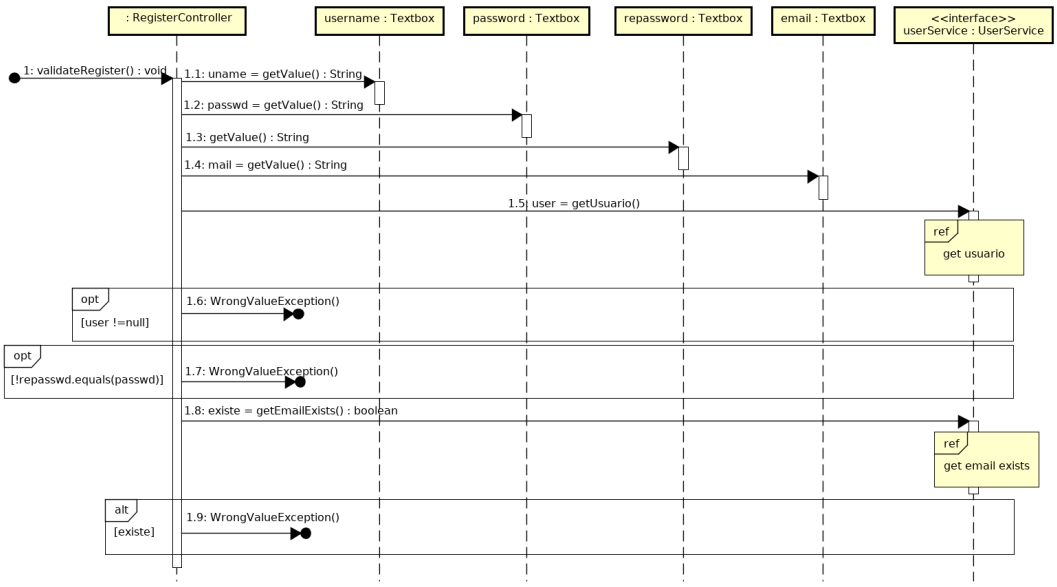

Figura 6.14: Subdiagrama “Validar registro”.

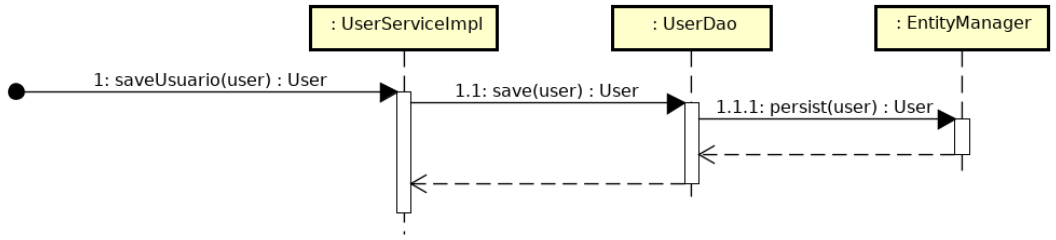

Figura 6.15: Subdiagrama “Guardar usuario”.

Los diagramas que se ven en las figura 6.16 y 6.17, muestran el escenario de la historia de usuario sobre una búsqueda de artículos en la aplicación. En este caso se mostrará un escenario de éxito, en el que la petición llega a la capa de servicios y posteriormente hace una petición a la capa de persistencia para una operación de lectura. Una vez filtrados los datos por la búsqueda del usuario, se podrá seleccionar un artículo de la lista mostrada, para visualizar información general de un artículo.

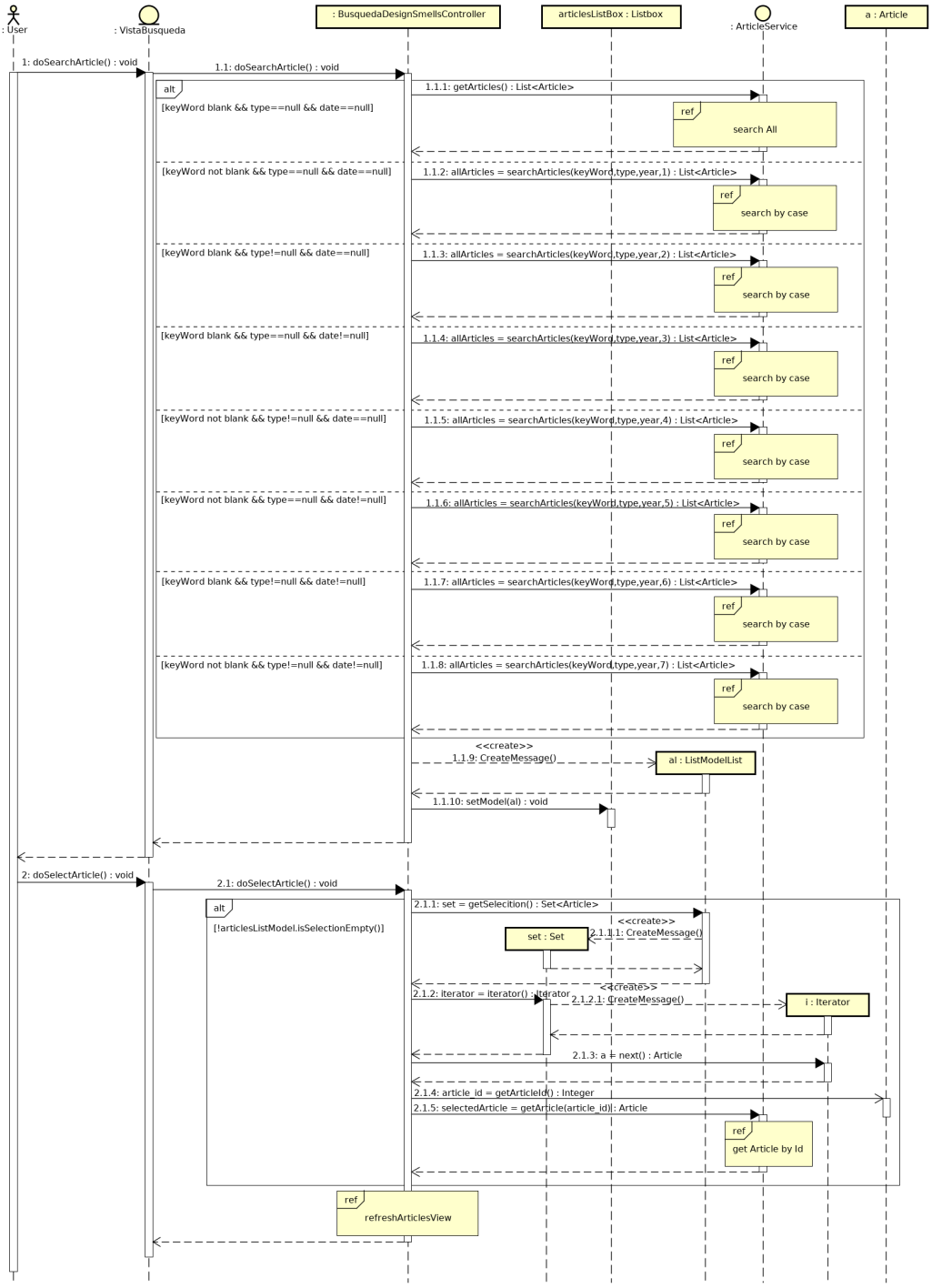

Figura 6.16: Diagrama secuencia búsqueda de artículos.

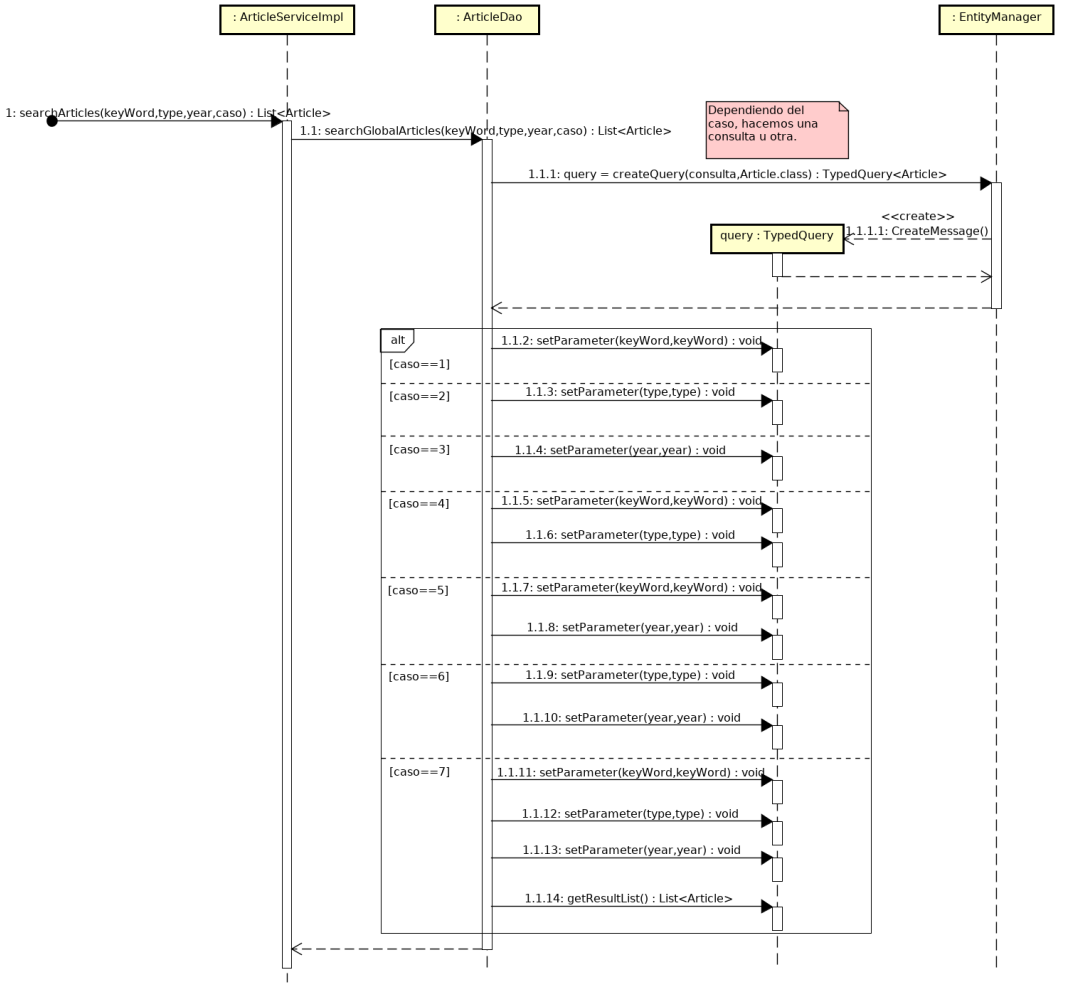

Figura 6.17: Subdiagrama búsqueda por caso.

6.6. Diagrama de despliegue

El siguiente diagrama muestra el modelo de despliegue de la aplicación web en tres nodos computacionales: el cliente, el servidor web y el servidor de base de datos.

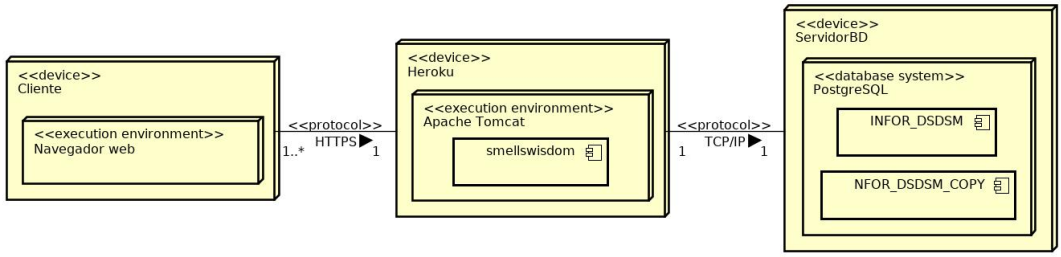

Figura 6.18: Modelo de despliegue.

La decisión de tener tres dispositivos para la aplicación, viene determinada por la poca capacidad de almacenamiento de datos de Heroku para su parte gratuita de PostgreSQL, por lo que es necesario separar la base de datos en otro servidor (en este caso de la escuela) para tener más capacidad de almacenamiento. Este diagrama de despliegue concuerda con el Modelo arquitectónico propuesto para la aplicación ya que separamos el procesamiento de la aplicación en tres niveles.

6.7. Diseño interfaz Gráfica

En esta sección mostraremos algunos bocetos que se han utilizado para guiar el diseño de la interfaz de usuario a través de ZUML.

Para la realización de los bocetos se ha utilizado la herramienta Balsamiq Mockups. Con estos bocetos se ha podido tener una visión general de la parte gráfica de las páginas de la aplicación. Estos se han ido creando tras cada sprint.

Por una parte tenemos la pantalla de Login que muestra un texto para el usuario y los cuadros para ingresar su usuario y contraseña como se puede ver en la figura 6.19. Es un diseño simple en el que únicamente se muestra lo esencial para que el usuario pueda iniciar sesión en el sistema.

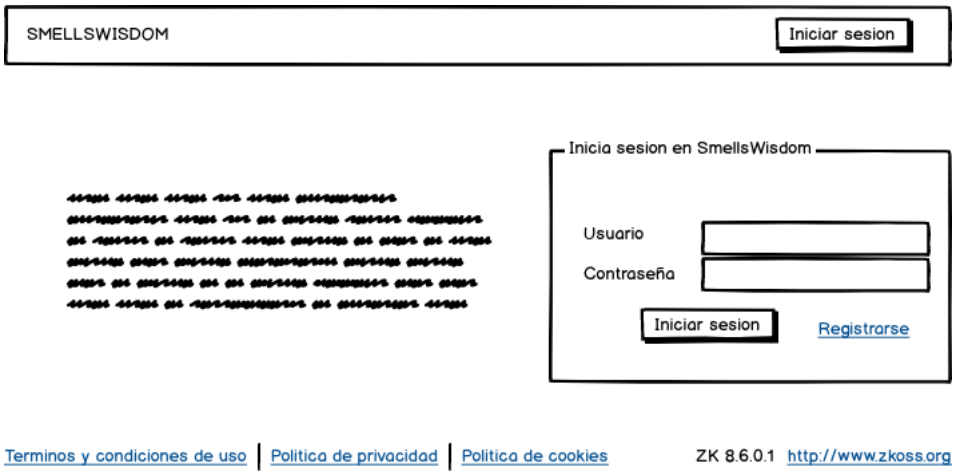

Figura 6.19: Boceto Login.

La figura 6.20 muestra cómo quedaría la pantalla de registro para usuario. En él tenemos un formulario para introducir los datos de un nuevo usuario del sistema. Incluye una opción para estar de acuerdo o no con la parte legal de la web y también incorpora visualmente la comprobación de los campos introducidos.

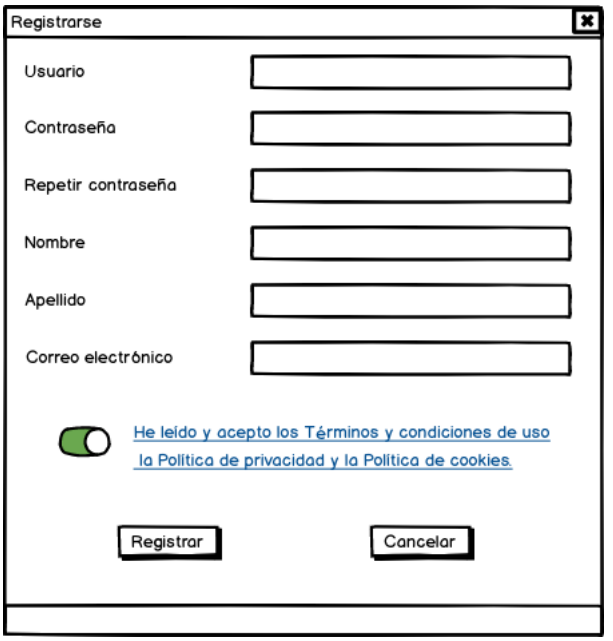

Figura 6.20: Boceto Registro.

La búsqueda de artículos muestra un cuadro para introducir palabras referentes a los artículos de la aplicación. Muestra un desplegable para elegir un filtro por tipo de publicación, permite introducir un año de publicación y el botón buscar. La lista contendrá un evento que en el caso de seleccionar un artículo, permitirá ver una pequeña información sobre lo que trata el artículo y la posibilidad de un botón que nos dirige a una parte más detallada en el que se muestra la información total del mismo. En un principio las cabeceras de la lista permiten ordenar el resultado de la búsqueda por Título, por Tipo de publicación y Año en el que se ha publicado.

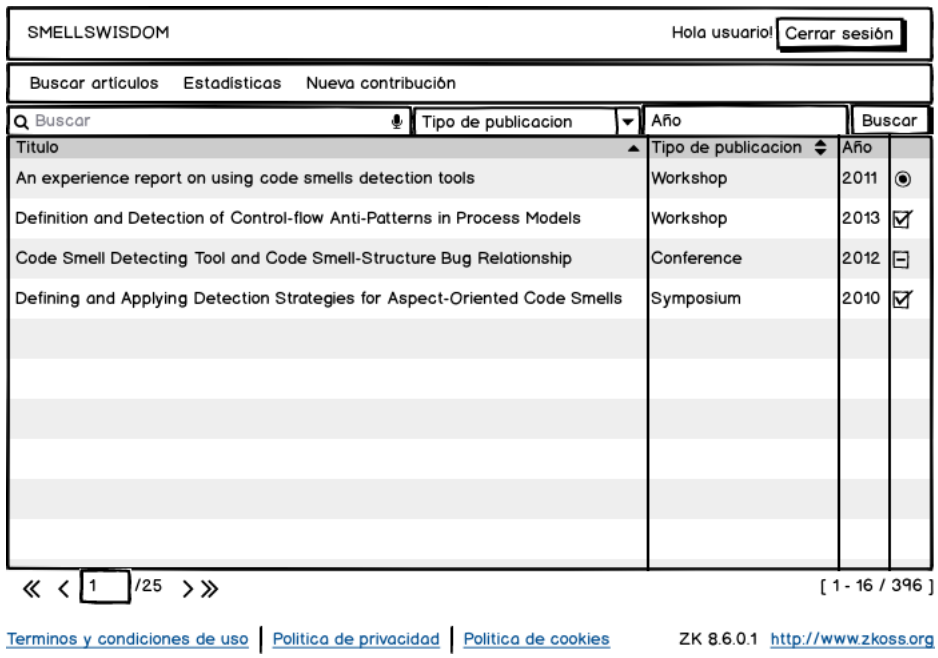

Figura 6.21: Boceto Búsqueda.

Para la parte de estadísticas tendremos un menú para poder visualizar todas las estadísticas que forman parte de la aplicación. En cada pestaña del menú podremos visualizar los gráficos generados con un título que describe qué datos muestra dicho gráfico, como se puede ver en la figura 6.22.

La figura 6.23 muestra la parte para añadir una nueva contribución. Para ello tenemos una vista general con formularios, listas, etc. para poder almacenar todos los datos que pueden aumentar el conocimiento de la aplicación. Tendremos diferentes apartados para añadir información general, la caracterización del artículo, los Design smells con los que trata, si hay evidencias de validación y las herramientas que menciona. Todas estas pestañas tendrán sus respectivas entradas para poder guardar información sobre lo que tratan. Tendrá un botón que permite guardar toda la información proporcionada por un usuario.

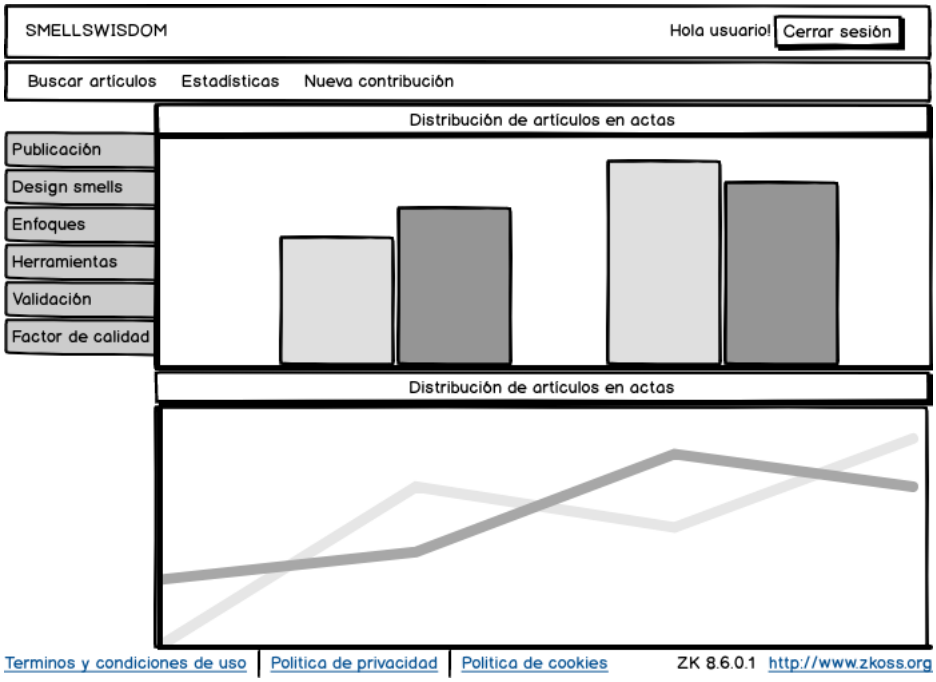

Figura 6.22: Boceto Estadísticas.

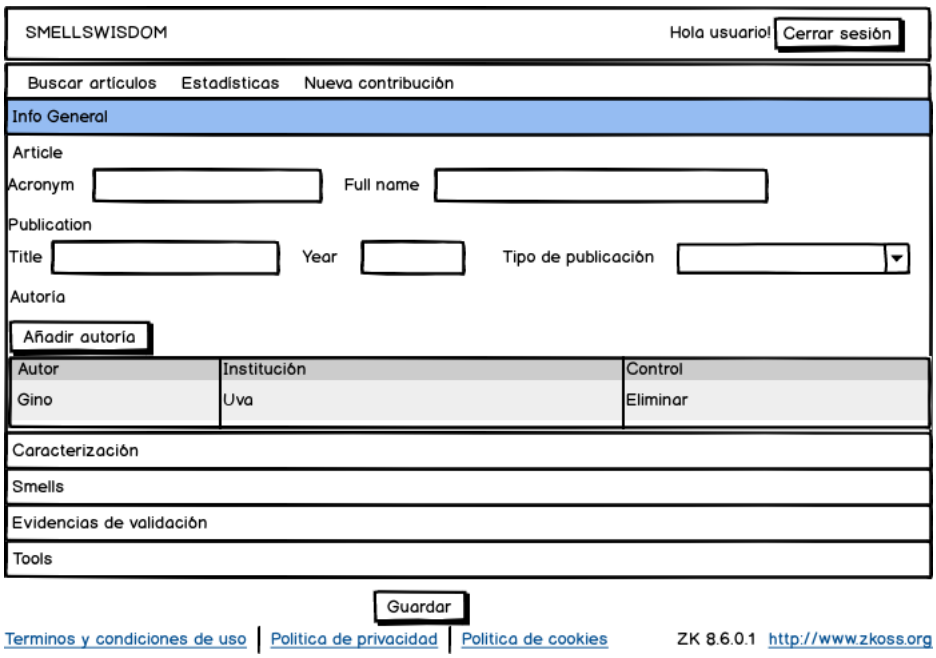

Figura 6.23: Boceto Nueva contribución.

6.8. Diseño de historias de usuario

En esta sección se hará un estudio detallado de las historias de usuario a implementar ya que como hemos visto durante el seguimiento del proyecto, algunas de ellas resultaron tener una complejidad excesiva denominadas *Epics*, las cuales hemos descompuesto en otras de menor tamaño. Para la descripción de las historias tendremos una definición y unos escenarios de aceptación en los que utilizaremos el lenguaje Gherkin [54], que define la estructura y una sintaxis básica para la descripción del comportamiento deseado.

6.8.1. Identificación de usuario

US-001	Prioridad	Estimación
	1	3
Definición		
Como usuario registrado Quiero iniciar sesión Para poder utilizar la aplicación.		
Escenario 1		
Identificación de usuario correcta.		
Dado que estoy registrado Cuando relleno los datos de usuario y contraseña Entonces debería ver la página de búsqueda de artículos.		
Escenario 2		
Identificación de usuario incorrecta.		
Dado que no estoy registrado Cuando relleno los datos de usuario y contraseña Entonces debería ver un mensaje de error en la misma pantalla de login.		

Tabla 6.1: El usuario podrá identificarse en la aplicación.

6.8.2. Registro de usuario

US-002	Prioridad	Estimación
	2	2
Definición		
Como usuario no registrado Quiero registrarme en el sistema Para poder utilizar la aplicación.		
Escenario 1		
Registro de usuario correcto.		
Dado que estoy registrado Cuando relleno los datos de usuario válidos Entonces debería ver un mensaje de creado correctamente y recibir un correo de confirmación.		
Escenario 2		
Registro de usuario incorrecto.		
Dado que no estoy registrado Cuando relleno los datos de usuario inválidos Entonces debería ver un mensaje de error en la misma pantalla de registro.		

Tabla 6.2: El usuario podrá registrarse en la aplicación.

6.8.3. Cerrar sesión

US-003	Prioridad	Estimación
	3	1
Definición		
Como usuario identificado Quiero cerrar sesión Para poder salir de la aplicación.		
Escenario 1		
Cierre sesión de usuario correcto.		
Dado que estoy logueado Cuando doy click en 'Cerrar sesión' en el template de la aplicación Entonces debería redirigirme a la página de login.		

Tabla 6.3: El usuario podrá cerrar sesión en la aplicación.

6.8.4. Consultar conocimiento sobre la detección de Design smells

Esta historia de usuario fue considera *epic*. Engloba las historias de usuario para visualizar información sobre la detección de Design smells. Hicimos una descomposición para que la implementación esté distribuida en más sprints.

US-004	Prioridad	Estimación
	4	3
Definición		
Como un usuario Quiero hacer una búsqueda de artículos sobre la detección de Design Smells Para estar informado sobre la detección.		
Escenario 1		
Búsqueda de artículos con éxito.		
Dado que estoy en la página de búsqueda Cuando introduzco una palabra, selecciono un tipo o introduzco un año Entonces debería filtrar los artículos por las opciones introducidas y mostrarlas como resultado, permitiendo seleccionar un artículo para mostrar información general de lo que trata.		
Escenario 2		
Búsqueda de artículos sin éxito.		
Dado que estoy en la página de búsqueda Cuando introduzco una palabra, selecciono un tipo o introduzco un año Entonces debería mostrar como resultado una lista vacía de artículos.		

Tabla 6.4: El usuario podrá buscar artículos en la aplicación.

US-005	Prioridad	Estimación
	5	3
Definición		
Como un usuario quiero visualizar en detalle un artículo (Publicación, Autoría y caracterización) Para estar informado sobre la detección.		
Escenario 1		
Visualizar en detalle un artículo (Publicación, Autoría y Caracterización)		
Dado que estoy en la página de detalle Cuando doy click en 'General' y 'Caracterización' Entonces debería visualizar información sobre la publicación, autoría y caracterización del artículo.		

Tabla 6.5: El usuario podrá el detalle un artículo (publicación, autoría y caracterización).

US-006	Prioridad	Estimación
	7	3
Definición		
Como un usuario Quiero visualizar en detalle un artículo (Smells, Evidencias y herramientas) Para estar informado sobre la detección.		
Escenario 1		
Visualizar en detalle un artículo (Smells, Evidencias y herramientas)		
Dado que estoy en la página de detalle Cuando doy click en 'Smells', 'Evidencias' o Herramientas Entonces debería visualizar información sobre smells, evidencias de validación y herramientas respectivamente.		

Tabla 6.6: El usuario podrá ver el detalle un artículo (smells, evidencias y herramientas).

US-007	Prioridad	Estimación
	8	3
Definición		
Como un usuario Quiero visualizar en detalle un artículo (Proyectos) Para estar informado sobre la detección.		
Escenario 1		
Visualizar en detalle un artículo (Proyectos)		
Dado que estoy en la página de smells Cuando selecciono un smell y doy click en 'se puede encontrar en proyectos' Entonces debería visualizar una lista de proyectos en los que se hace referencia al smell seleccionado.		
Escenario 2		
Visualizar en detalle un artículo (Proyectos)		
Dado que estoy en la página de evidencias Cuando selecciono una evidencia de validación Entonces debería visualizar una lista de proyectos usados en la evidencia.		

Tabla 6.7: El usuario podrá ver el detalle un artículo (proyectos)

6.8.5. Visualizar estadísticas sobre la detección de Design smells.

Esta historia de usuario fue considera *epic*. Engloba las historias de usuario para visualizar estadísticas sobre la detección de Design smells. Hicimos una descomposición para que la implementación esté distribuida en más sprints.

US-008	Prioridad	Estimación
	6	2
Definición		
Como un usuario Quiero visualizar estadísticas referentes a publicaciones Para estar informado sobre las publicaciones utilizadas en el estudio.		
Escenario 1		
Visualizar estadísticas de publicaciones		
Dado que estoy en la página de estadísticas Cuando doy click a la pestaña 'publicaciones' Entonces debería visualizar gráficos estadísticos referentes a publicaciones.		

Tabla 6.8: El usuario podrá visualizar estadísticas sobre publicaciones.

US-009	Prioridad	Estimación
	9	2
Definición		
Como un usuario Quiero visualizar estadísticas referentes a Design smells Para estar informado sobre los Design smells detectados en los artículos.		
Escenario 1		
Visualizar estadísticas de Design smells		
Dado que estoy en la página de estadísticas Cuando doy click a la pestaña 'Smells' Entonces debería visualizar gráficos estadísticos referentes a Design smells.		

Tabla 6.9: El usuario podrá visualizar estadísticas sobre Design smells.

US-010	Prioridad	Estimación
	10	2
Definición		
Como un usuario Quiero visualizar estadísticas referentes a enfoques Para estar informado sobre los enfoques utilizados en los artículos.		
Escenario 1		
Visualizar estadísticas de enfoques		
Dado que estoy en la página de estadísticas Cuando doy click a la pestaña 'Enfoques' Entonces debería visualizar gráficos estadísticos referentes a enfoques..		

Tabla 6.10: El usuario podrá visualizar estadísticas sobre enfoques.

US-011	Prioridad	Estimación
	11	2
Definición		
Como un usuario Quiero visualizar estadísticas referentes a herramientas Para estar informado sobre los herramientas utilizadas en los artículos.		
Escenario 1		
Visualizar estadísticas de herramientas		
Dado que estoy en la página de estadísticas Cuando doy click a la pestaña 'Herramientas' Entonces debería visualizar gráficos estadísticos referentes a herramientas.		

Tabla 6.11: El usuario podrá visualizar estadísticas sobre herramientas.

US-012	Prioridad	Estimación
	12	2
Definición		
Como un usuario Quiero visualizar estadísticas referentes a evidencias de validación Para estar informado sobre la relación entre evidencias, proyectos y medidas.		
Escenario 1		
Visualizar estadísticas de evidencias		
Dado que estoy en la página de estadísticas Cuando doy click a la pestaña 'Validación' Entonces debería visualizar gráficos estadísticos referentes a evidencias de validación.		

Tabla 6.12: El usuario podrá visualizar estadísticas sobre evidencias de validación.

US-013	Prioridad	Estimación
	13	2
Definición		
Como un usuario Quiero visualizar estadísticas referentes a factores de calidad Para estar informado sobre la relación entre factores de calidad y Design smells.		
Escenario 1		
Visualizar estadísticas de factores de calidad		
Dado que estoy en la página de estadísticas Cuando doy click a la pestaña 'Factor de calidad' Entonces debería visualizar gráficos estadísticos referentes a factores de calidad.		

Tabla 6.13: El usuario podrá visualizar estadísticas sobre factores de calidad.

6.8.6. Contribuir añadiendo conocimiento sobre la detección de Design smells.

Esta historia de usuario fue considera *epic*. Engloba las historias de usuario para guardar nuevo conocimiento sobre la detección de Design smells. Hicimos una descomposición para que la implementación esté distribuida en más sprints.

US-014	Prioridad	Estimación
	14	3
Definición		
Como un usuario Quiero contribuir con un nuevo elemento al conocimiento (Publicación, Autoría y Caracterización) para aumentar el conocimiento sobre la detección de Design smells.		
Escenario 1		
Guardar nueva contribución en el sistema con éxito		
Dado que estoy en la página de nueva contribución Cuando doy click a 'General' Entonces debería ver desplegables, cuadros de texto y una lista para poder ingresar nuevos datos o seleccionar datos ya existentes para la nueva contribución Cuando doy click a 'Enfoque' Entonces debería poder ver desplegables, cuadros de texto y listas para poder ingresar nuevos datos o seleccionar datos ya existentes para la nueva contribución Cuando doy click a 'guardar' Entonces debería mostrar un mensaje de que se ha guardado correctamente la contribución, si se han introducido correctamente los datos.		
Escenario 2		
Guardar nueva contribución en el sistema sin éxito		
Dado que estoy en la página de nueva contribución Cuando doy click a 'General' Entonces debería ver desplegables, cuadros de texto y una lista y para poder ingresar o seleccionar datos ya existentes a la nueva contribución Cuando doy click a 'Enfoque' Entonces debería poder ver desplegables, cuadros de texto y listas para poder ingresar o seleccionar datos ya existentes a la contribución Cuando doy click a 'guardar' Entonces debería mostrar un mensaje de que no se ha guardado correctamente la contribución y mostrar mensajes de error en los campos requeridos, si no se han introducidos datos correctos.		

Tabla 6.14: El usuario podrá contribuir añadiendo un nuevo elemento (publicación,autoría y caracterización).

US-015	Prioridad	Estimación
	15	3
Definición		
Como un usuario Quiero contribuir con un nuevo elemento al conocimiento (Smells, Evidencias y herramientas) para aumentar el conocimiento sobre la detección de Design smells.		
Escenario 1		
Guardar nueva contribución en el sistema con éxito		
Dado que estoy en la página de nueva contribución Cuando doy click a 'Smells' Entonces debería ver listas para poder ingresar nuevos datos o seleccionar datos ya existentes a la nueva contribución Cuando doy click a 'Evidencias de validación' Entonces debería poder ver listas para poder ingresar nuevos dato o seleccionar datos ya existentes a la nueva contribución Cuando doy click a 'Herramientas' Entonces debería poder ver listas para poder ingresar nuevos dato o seleccionar datos ya existentes a la nueva contribución Cuando doy click a 'guardar' Entonces debería mostrar un mensaje de que se ha guardado correctamente la contribución,si se han introducido correctamente los datos.		
Escenario 2		
Guardar nueva contribución en el sistema sin éxito		
Dado que estoy en la página de nueva contribución Cuando doy click a 'Smells' Entonces debería ver listas para poder ingresar nuevos dato o seleccionar datos ya existentes a la nueva contribución Cuando doy click a 'Evidencias de validación' Entonces debería poder ver listas para poder ingresar nuevos dato o seleccionar datos ya existentes a la nueva contribución Cuando doy click a 'Herramientas' Entonces debería poder ver listas para poder ingresar nuevos datos o seleccionar datos ya existentes a la nueva contribución Cuando doy click a 'guardar' Entonces debería mostrar un mensaje de que no se ha guardado correctamente la contribución y mostrar mensajes de error en los campos requeridos, si no se han introducidos datos correctos.		

Tabla 6.15: El usuario podrá contribuir añadiendo un nuevo elemento (smells, evidencias y herramientas).

Capítulo 7

Implementación y pruebas

Esta aplicación, de software libre, tiene una licencia **GNU General Public License v3.0** que permite continuar con el desarrollo del software siempre y cuando las versiones posteriores mejoradas que se publiquen tengan la misma licencia.

En este capítulo se describe cómo se ha construido el entorno de desarrollo utilizando Eclipse y Maven y cómo está compuesta la estructura de directorios. Se hace referencia a patrones importantes utilizados durante el desarrollo y las pruebas realizadas para verificar si el sistema software se comporta correctamente.

7.1. Construcción del entorno de desarrollo

En primer lugar se crea un proyecto Maven en Eclipse, utilizando un arquetipo proporcionado por ZK framework para la creación de un proyecto web empleando las dependencias de Zk framework Community Edition.

Maven se encarga de las dependencias del proyecto, organizar las estructuras de directorios y simplificar las operaciones de compilación y empaquetado, ejecución de tests y despliegue entre otras tareas como la generación de informes y documentación. Todo esto es posible definiendo un único fichero en formato XML denominado pom.xml, que tiene todo lo necesario para generar el ejecutable de nuestra aplicación.

Una vez creado el proyecto maven, añadiremos las dependencias de Spring, Spring security e Hibernate como implementación de JPA. Tras la comprobación del funcionamiento correcto de la aplicación creamos un repositorio git, alojado en la plataforma Gitlab de la escuela.

En cuanto a la estructura de directorios seguimos el estándar de Maven como se muestra en la figura 7.1. En el directorio src se encuentra el directorio main. Dentro de este tendremos

los directorios java, donde situaremos nuestras clases en distintos paquetes; el directorio resources donde situaremos los recursos (ficheros XML de configuración como los de Spring o Hibernate, ficheros de propiedades como los de Spring security y scripts sql) que puedan necesitar las clases java de nuestro proyecto. En el directorio webapp emplezaremos todo lo relativo a la parte web de la aplicación como archivos css, imágenes, las páginas zul, etc. El otro directorio test tiene un subdirectorio java donde situaremos el código fuente correspondiente a test unitarios y de integración.

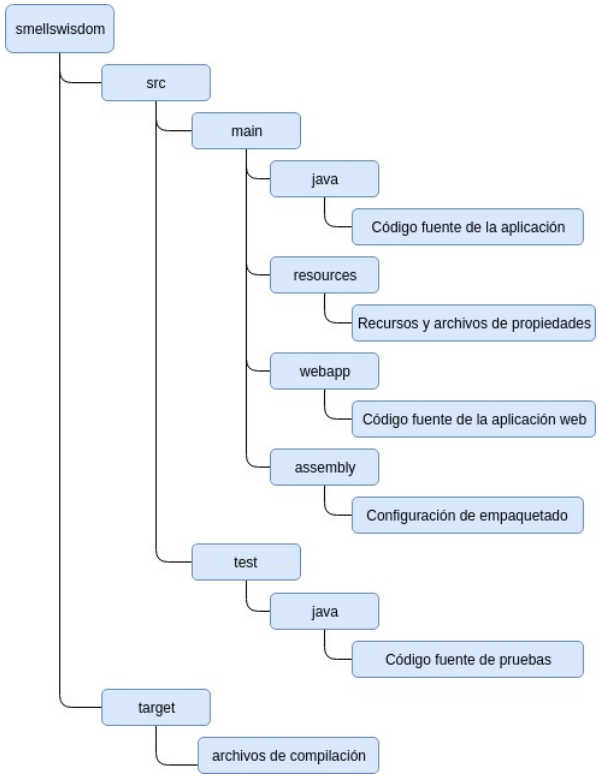

Figura 7.1: Estructura directorios.

7.2. Usando el patrón método factoría

Uno de los patrones de diseño más utilizados es el de método factoría, que permite delegar la creación de los objetos a las subclases. Lo utilizaremos para crear los diferentes tipos de artículos a la hora de querer guardarlos en la base de datos.

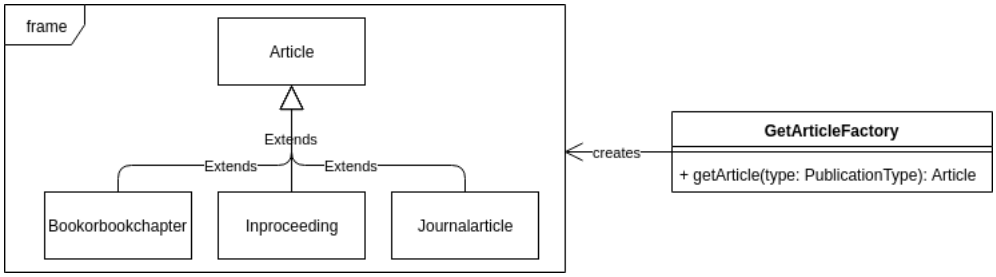

Figura 7.2: Patrón método factoría.

7.3. Pruebas

Durante cada sprint se realizan pruebas y las verificaciones necesarias para saber si el sistema software funciona correctamente, cumple con la especificación y resuelve los requisitos planteados inicialmente. Se prueban todas las funcionalidades de la aplicación web para comprobar los fallos que pueden aparecer o posibles mejoras.

Las pruebas realizadas en la aplicación se han llevado a cabo de manera manual. Las abordaremos cada vez que acabemos un sprint para aceptar o no la funcionalidad hecha. Las pruebas automatizadas han sido adoptadas utilizando JUnit y Mockito con las que probaremos el correcto funcionamiento de la aplicación, de manera aislada y en su conjunto, para asegurar su consistencia y robustez. En cuanto a las pruebas de sistema no las haremos con usuarios reales, puesto que sólo las realizamos con el tutor del trabajo de fin de grado.

Por otro lado, dado que realizamos entrega e integración continua, las pruebas siempre se realizan sobre el sistema desplegado de forma real en Heroku.

Algunos errores subsanados en la aplicación eran los ocasionados por la configuración de Spring security, ya que primero no encontraba al usuario correspondiente tras el inicio de sesión, por un error en el código. Otro error ocasionado fue por el cierre de sesión, ya que no estaba bien configurada la salida a la página de login, quedando una página en blanco tras salir de la sesión. Los mensajes de error tras un inicio de sesión inadecuado o si un usuario no tiene permitido el acceso, no se mostraban correctamente por lo que se hace una mejora añadiendo archivos de propiedades tanto en inglés como en español.

Para poder observar la distinción entre los roles de un usuario, se crea un menú para la búsqueda de artículos y otro para administración, así damos por aceptada la integración con Spring security y las historias de usuario de inicio de sesión y cierre de sesión.

Otro error relacionado con el registro en el sistema fue la imposibilidad de enviar el correo de confirmación de cuenta, debido a que el servicio de correo no puede iniciar sesión con la cuenta de Gmail.

En cuanto a la configuración de Heroku, el archivo Procfile que define los comandos que ejecuta al inicio, no funciona correctamente dando un error cuando Heroku interpreta este fichero. Se solventa creando un archivo Procfile nuevo, puesto que se cree que el archivo anterior no tiene la codificación de caracteres correcta.

Los enumerados de las entidades dan errores a la hora de cargarlos en memoria, ya que no se puede identificar el enumerado con un número. Esto quiere decir que al tener una tabla en la base de datos para un enumerado con un id y el tipo, es necesario indicar de alguna manera con qué id concuerda el enumerado definido en diseño. Para solventar este problema definiremos convertidores de JPA para todos los enumerados.

Otro error visto fue a la hora de generar los gráficos, ya que hay consultas SQL que no funcionan como deberían puesto que hay datos que no son los correctos. Por tanto es necesario invertir más tiempo a la hora de escribir una consulta.

A la hora de cargar las páginas surgen errores, como pueden ser los tiempos de respuesta excesivos de los artículos a la hora de listarlos. Esto lo arreglamos indicando que las relaciones entre entidades las cargue de forma perezosa.

Otro problema que tenemos a la hora de probar la navegación entre las páginas son los errores obtenidos en el momento de pasar parámetros entre ellas, ya que hay que distinguir entre inclusión de páginas en modo instantáneo y diferido. Con el arreglo de los enumerados , la carga de páginas y los problemas con la navegación, damos por aceptadas las historias de usuario referentes a la visualización de información y añadir nuevas contribuciones.

Capítulo 8

Conclusiones y Trabajo futuro

Los objetivos principales del proyecto no se cumplen en su totalidad, por un lado se realiza la aplicación web para que los usuarios puedan registrarse y poder acceder al conocimiento proporcionado. Sin embargo, la parte que permitir colaborar para aumentar la información no se completa, solo se abarca la posibilidad de añadir nueva información sin posibilidad de modificar ya sea aportando comentarios, sugerencias o nuevos detalles para los datos ya registrados. La posibilidad de seguir con el desarrollo de la aplicación se deja en manos de otros desarrolladores, ya que la aplicación utiliza una licencia de software libre para ello.

8.1. Conclusiones

Una de las cosas que he aprendido de este proyecto, es lo importante que es la detección de Design smells en la Ingeniería del Software, ya que los Design smells pueden afectar negativamente a la calidad del software. También conocer lo importante que es una revisión sistemática de la literatura, ya que hoy en día hay diversas investigaciones respecto a este tema, por lo que ayudaría a unificar todo este conocimiento para destinarlo a que desarrolladores e investigadores de software tomen mejores decisiones a la hora de construir un sistema. Se pretende que todo el conocimiento obtenido esté disponible para la comunidad a través de una aplicación web.

Otra de las cosas que he aprendido, es cómo las metodologías ágiles ayudan a que un proyecto crezca tras cada iteración y que los cambios que puedan surgir sean capaces de adaptarse rápidamente. Gracias a reuniones típicas del marco de trabajo SCRUM, podemos incluir mejoras o eliminar fallos tras la realización de cada sprint.

Una de las cosas que más problemas ocasionó fue la planificación del proyecto. Por una parte, la división general de las historias de usuario no es la correcta, ya que posteriormente se

observó que la complejidad de las mismas era elevada. Estas historias de usuario se denominan *epics*, por lo que se hizo un refinamiento para dividir el trabajo y hacer historias de usuario de menor tamaño.

En cuanto a la estimación de tareas en cada Sprint, ocurre que la implementación de la funcionalidad ha sido afectada por falta de experiencia y tiempo al estudiar la documentación de las herramientas a utilizar, dando como resultado un aumento en el coste total del proyecto.

El uso de herramientas y tecnologías gratuitas supone un gran ahorro directo a la hora de los costes del proyecto. Además estas cuentan con una gran comunidad que facilita la resolución de problemas. La utilización de estas tecnologías y herramientas en general ha sido un acierto, ya que algunas ya las había utilizado y otras me ha tocado aprender desde cero. Sin embargo, el coste de aprender a cómo utilizarlas ha sido una buena experiencia. En lo personal, la investigación inicial tanto de las nuevas como de las ya utilizadas, me ha permito por un lado ampliar conocimientos en ZK framework y JPA(Hibernate) y aprender frameworks como Spring, así como implementar la seguridad de la web con Spring security.

Otra de las cosas interesantes que he aprendido, es la infraestructura que ofrece Heroku a la hora de permitirnos desplegar nuestra aplicación. Por otra parte es una herramienta gratuita aunque limitada en cuanto a número de usuarios permitidos y la capacidad de almacén de datos en PostgresSQL. Esto nos ha llevado a montar un servidor de base de datos en la escuela.

Por último, cabe destacar que este trabajo de fin de grado ha sido una gran oportunidad no sólo para adquirir nuevos conocimientos y habilidades, sino también para poder desarrollar una de las actividades que más motivan, que es el desarrollo de software. Más allá de lo meramente económico, me complace saber que el desarrollo de esta aplicación puede ayudar a otros desarrolladores e investigadores de software a que tomen decisiones adecuadas a la hora de construir nuevos sistemas software.

8.2. Líneas de trabajo futuro

Dado el poco tiempo disponible para realizar el trabajo de fin de grado, quedan pendientes historias de usuario por cubrir. Estas historias quedan reflejadas en la pila de historias de usuario definida como *Icebox* en pivotal tracker, que se compone de todas aquellas historias del Backlog que no pasarán al sprint actual como podemos ver en la siguiente figura 8.1.

La funcionalidad faltante por una parte son aquellas historias de usuario destinadas a la añadir nueva información faltante, aportación de comentarios y sugerencias de corrección para la parte de usuarios.

Por otra parte la funcionalidad que falta por implementar es la dedicada a la parte del administrador, para que pueda aprobar o no los distintos tipos de aportaciones que puede

realizar un usuario en el sistema. Dado que la parte de administración depende del cambio de la base de datos auxiliar, este cambio se deja como una tarea a realizar en el *Icebox*.

Dado que tenemos multitud de datos, una de las mejoras sería una búsqueda avanzada de artículos, haciendo un filtrado por autores e instituciones, Design smells que trata, herramientas que menciona, etc.

En cuanto a la parte web de la aplicación una de las mejoras en un futuro es que las vistas tengan un diseño adaptable a multitud de dispositivos.

Una posible mejora sería la del rendimiento de la web, mejorando el código, optimizando o reduciendo los tiempos de carga, de base de datos, etc.

Por último, estas funcionalidades que faltan por implementar se podrán continuar desarrollando en un futuro ya que este proyecto tiene una licencia de código libre

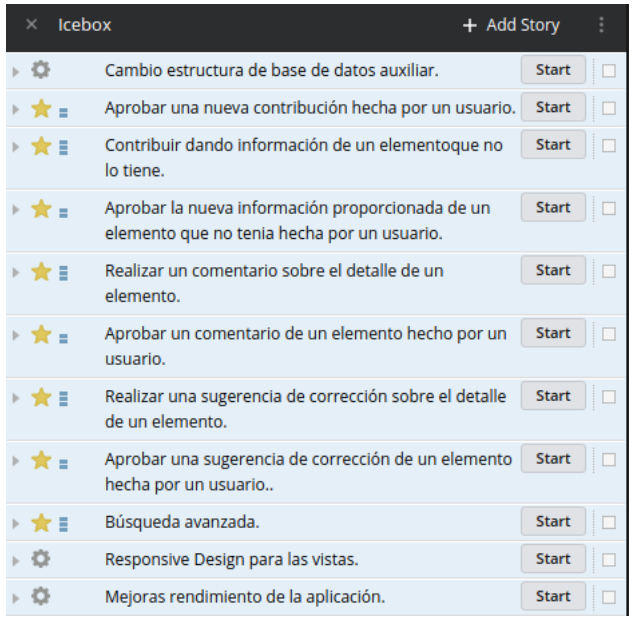

Figura 8.1: Icebox de Pivotal Tracker.

Apéndice A

Manual de mantenimiento

En este apartado se describirá la forma en la que podemos seguir con el desarrollo o hacer el mantenimiento de la aplicación en una máquina.

A.1. Requisitos

Para que funcione todo correctamente se aconseja tener instaladas y configuradas las siguientes herramientas (se indican las versiones que se han utilizado en las realización del proyecto):

- JDK (Java Development Kit) versión 1.8.0_212.
- Eclipse versión 2019-06 (4.12.0).
- Git versión 2.17.1.
- PostgreSQL 12.0.
- Navegador web como Google Chrome o Firefox en su versión más reciente.

Dado que utilizamos Eclipse como IDE de desarrollo, el proyecto es compatible con todos los sistemas operativos Windows, Linux y Mac.

Primero instalamos JDK, que provee de herramientas que permiten construir programas usando el lenguaje de programación Java. Se puede descargar de forma gratuita desde la siguiente dirección `https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`.

Tras la instalación de JDK, pasaremos a instalar Eclipse IDE for Enterprise Java Developers que nos facilitará el desarrollo de la aplicación web. Se puede descargar de forma gratuita en la siguiente dirección `https://www.eclipse.org/downloads/packages/`.

Con Eclipse instalado pasaremos a instalar el plugin ZK Studio, que proporciona herramientas que ayudan a mejorar de desarrollo de aplicaciones con ZK framework. Para ello lo buscaremos en Eclipse Marketplace y lo instalaremos.

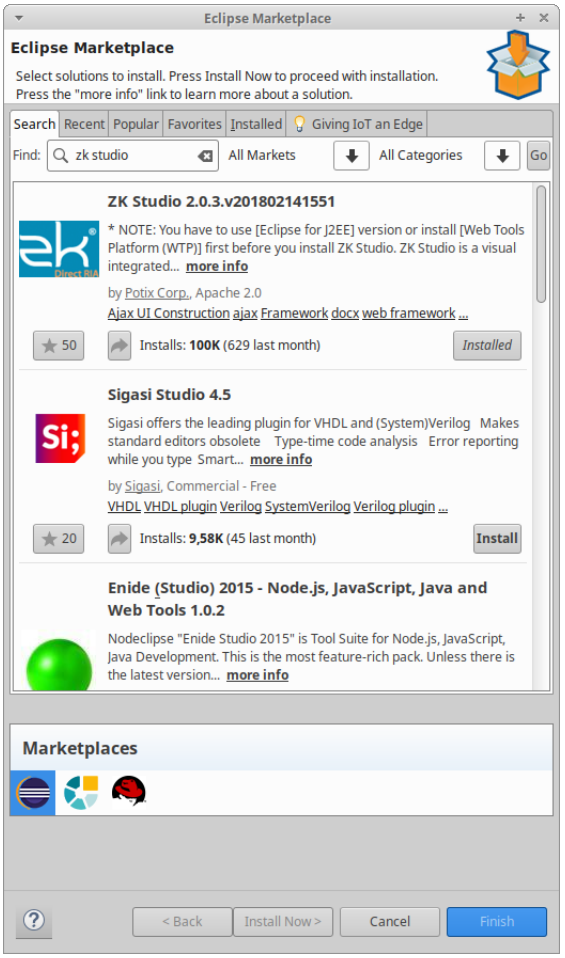

Figura A.1: ZK Studio en Eclipse Marketplace.

Una vez instalado Eclipse y el plugin necesario para ZK framework será necesario iniciar sesión con una cuenta de ZK. Para ello introduciremos las siguientes credenciales:

- Usuario: smellswisdom

- Contraseña: smeLL2019_tfg

Posteriormente pasaremos a instalar Git. Así traeremos a un repositorio local la aplicación alojada en el repositorio remoto GitLab de la escuela ubicado en la siguiente url `https://gitlab.inf.uva.es/ginquin/smellswisdom.git`, de modo que podamos continuar con el desarrollo de la misma. Para instalar Git podemos seguir las instrucciones proporcionadas en la siguiente dirección `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`.

Con git instalado pasaremos a crear una carpeta, que en este caso llamamos *repositorio*, la cual funcionará como repositorio local. Una vez creada la carpeta, pasaremos a clonar el proyecto con el siguiente comando:

```
~/repositorio$ git clone https://gitlab.inf.uva.es/ginquin/smellswisdom.git
```

Tras clonar el proyecto, pasaremos a importarlo en Eclipse. Para ello elegiremos la opción de importar proyectos existentes de Maven. Será necesario indicar el repositorio local en el que tenemos la aplicación, lo seleccionamos y damos finalizar.

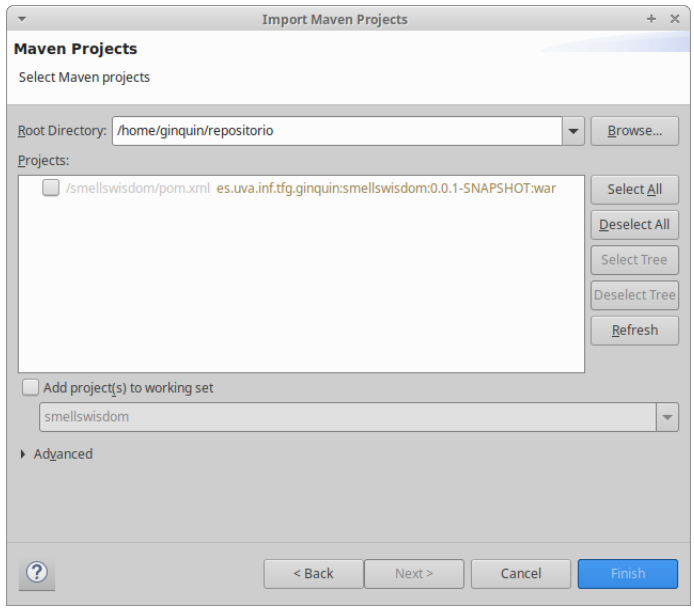

Figura A.2: Importar proyecto Maven.

Con el proyecto importado, pasaremos a realizar una configuración para que no dé errores en la parte de JPA. Para ello, en Eclipse iremos a 'Window' →'Preferences' →'Validation' →'JPA Validator' →'Settings' daremos click al botón 'Add Include Group'.

Al incluir un nuevo grupo, daremos click en 'Add Rule' y seleccionaremos 'folder or file name', en la ventana 'New Filter Rule Wizard' e indicaremos la carpeta de recursos de la aplicación 'src/main/resources' y damos finalizar.

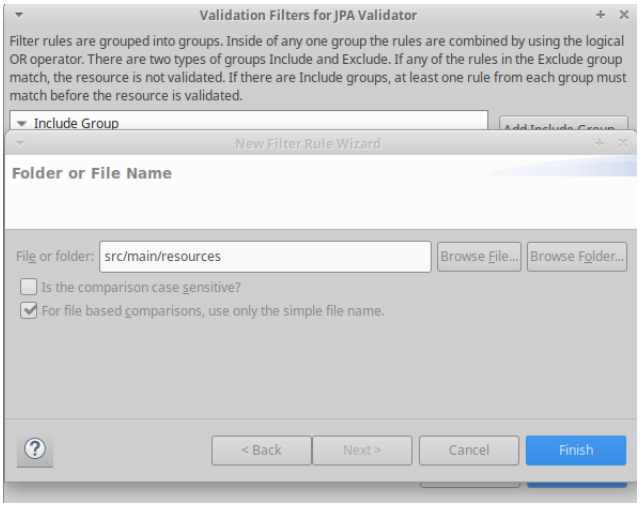

Figura A.3: Añadir carpeta de recursos al validador de JPA.

La instalación de PostgreSQl que utilizaremos como sistema de gestión de base de datos relacional para trabajar en local, se puede descargar de forma gratuita en la siguiente dirección `https://www.postgresql.org/download/`.

Una vez instalado, lo configuraremos para que el usuario sea *postgres* y la contraseña *postgres* para una mayor simplicidad. Crearemos dos bases de datos, una para la principal y otra para la auxiliar. En ellas cargaremos los scripts sql de estructura y datos, ya que ambas contendrían la misma información. En este caso hemos nombrado las dos bases de datos de la siguiente manera *INFOR_DSDSM* y *INFOR_DSDSM_COPY*.

Con todo lo anterior configurado pasaremos a indicar cómo podemos ejecutar la aplicación a través de Eclipse.

A.2. Ejecutar la aplicación

Para ejecutar la aplicación haremos la siguiente configuración en Eclipse, de manera que se inicie con Webapp Runner y pueda conectarse a la base de datos.

- Haremos click derecho en el proyecto y elegiremos 'Run As' →'Run Configurations...'

- En la ventana 'Run Configurations', crearemos una nueva configuración de inicio para la aplicación. Para ello nos situaremos en 'Java Application' y haremos click derecho y seleccionamos 'New Configuration'.

- Le daremos el nombre que queramos e ingresaremos el nombre del proyecto 'smellwisdom' en el cuadro 'Project'.

- En el cuadro 'Main class' ingresaremos 'webapp.runner.launch.Main'.

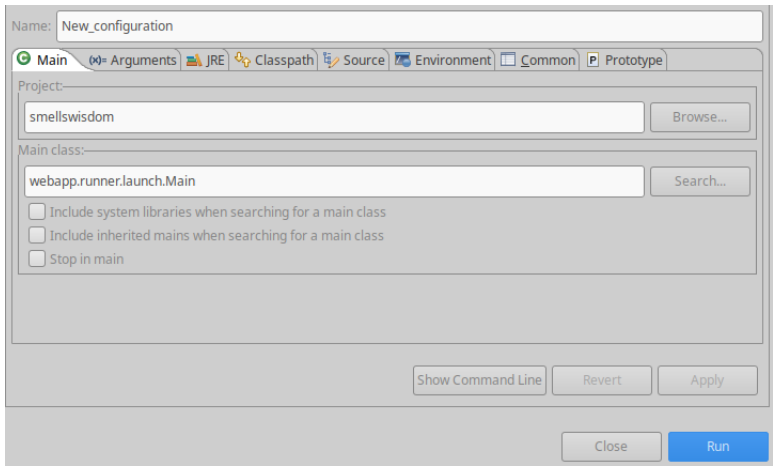

Figura A.4: Pestaña Main.

- En la pestaña 'Arguments' ingresaremos './src/main/webapp' en el cuadro 'Program 'arguments.

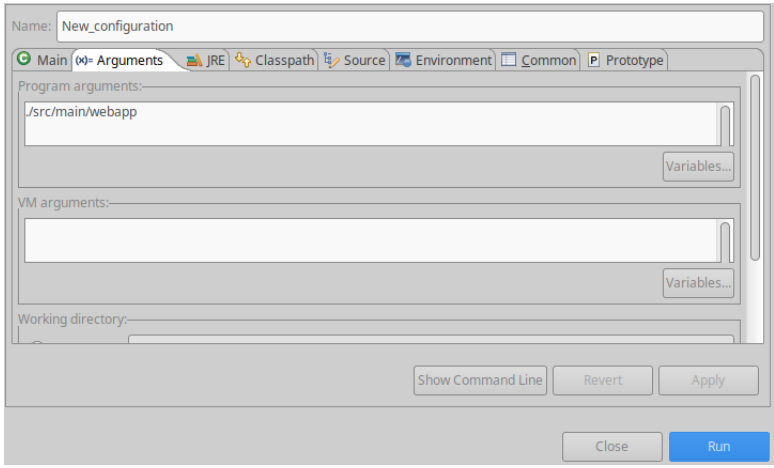

Figura A.5: Pestaña Arguments

- En la pestaña 'Environment', crearemos dos variables de entorno para poder conectarnos con la base de datos en local. Para ello daremos click en 'New...'
- En la ventana 'New Environment Variable', para la base de datos principal en el cuadro 'Name' ingresaremos 'DATABASE_URL' y en el cuadro 'Value' ingresaremos la url de la base de datos '`postgres://postgres:postgres@localhost:5432/INFOR_DSDSM`'.
- Para la base de datos auxiliar en el cuadro 'Name' ingresaremos 'DATABASE_URL_COPY' y en el cuadro 'Value' ingresaremos la url de la base de datos '`postgres://postgres:postgres@localhost:5432/INFOR_DSDSM_COPY`'.

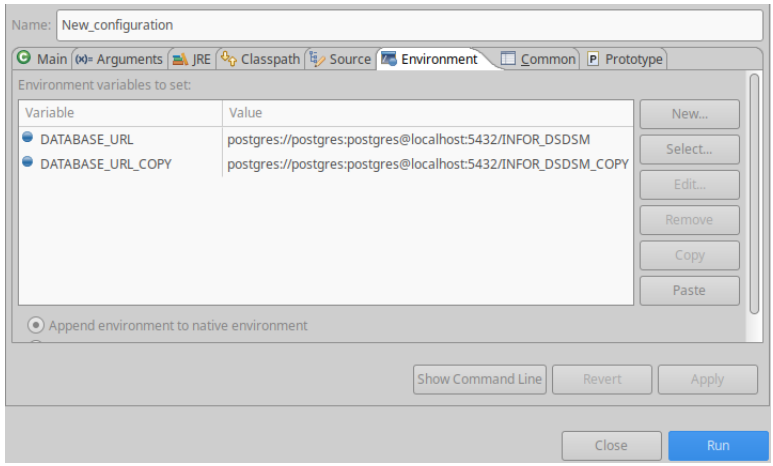

Figura A.6: Pestaña Environment.

- Damos click en 'Apply' y 'Run'.

Si se han seguido los pasos, la aplicación debería funcionar correctamente y se debería ver la salida del registro en la consola de Eclipse.

Se ha establecido un correo electrónico en Gmail con el que se han creado las diferentes cuentas utilizadas en el desarrollo del proyecto, así como el servicio de correo electrónico para el envío de mensajes de confirmación de correos. Por ello se comparten las credenciales:

- Usuario: testmailsmells2019@gmail.com
- Contraseña: smeLL2019_tfg

Apéndice B

Manual de despliegue

En este apartado indicaremos la configuración en la máquina de desarrollo, Heroku y el servidor de base de datos para que la aplicación quede desplegada para su uso.

Para el despliegue de la aplicación utilizaremos Heroku que es una plataforma de despliegue PaaS (Platform as a Service), la cual tiene un servicio gratuito y soporta el lenguaje de programación Java.

B.1. Máquina desarrollo

En la máquina con la que desarrollamos el proyecto, instalaremos Heroku Command Line Interface (CLI), que proporciona un entorno de administración en linea de comandos. Se puede descargar en la siguiente dirección `https://devcenter.heroku.com/articles/heroku-cli`.

Una vez tengamos instalado Heroku CLI, iniciaremos sesión en Heroku CLI utilizando la cuenta creada en el desarrollo del proyecto, que tiene las siguiente credenciales:

- Usuario: testmailsmells2019@gmail.com
- Contraseña: smeLL2019_tfg

```
~$ heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to
↪  https://cli-auth.heroku.com/auth/browser/8f84f6d9-c973-4960-88fb-ddce95fbf463
```

```
Logging in... done
Logged in as testmailsmells2019@gmail.com
```

Una vez iniciada la sesión, nos dirigimos al repositorio local donde tenemos la aplicación, accedemos a la carpeta raíz del proyecto y ejecutamos el siguiente comando en el cual indicamos el nombre de la aplicación Heroku que hemos creado durante el desarrollo del proyecto.

```
~/repositorio/smellswisdom$ heroku git:remote -a  guarded-citadel-24844
```

Para desplegar la aplicación, simplemente sincronizamos la rama máster de nuestro repositorio local de Git, con el que tenemos alojado en Heroku.

```
~/repositorio/smellswisdom$ git push heroku master
```

Una vez actualizado el código del proyecto en el repositorio de producción, Heroku se encargará automáticamente de compilar y desplegar la aplicación si todo va bien y no se encuentran fallos.

Para abrir la aplicación web podemos poner la siguiente url en el navegador `https://guarded-citadel-24844.herokuapp.com/` o desde la máquina de desarrollo podemos utilizar el comando siguiente.

```
~/repositorio/smellswisdom$ heroku open
```

B.2. Heroku

Para las conexiones con el servidor de base de datos de la escuela al igual que en la máquina local, crearemos dos variables de entorno con las urls de la base de datos principal y la base de datos secundaria.

Para ello una vez iniciado sesión en la web de Heroku, iremos a la aplicación 'guarded-citadel-24844' y en las pestañas de configuraciones de la aplicación iremos a la pestaña 'Settings'.

- En el apartado 'Config Vars' añadiremos primero la variable de entorno para la base de datos principal. Para ello en cuadro 'KEY' indicaremos 'DATABASE_URL' y en el cuadro 'VALUE' indicaremos `postgres://postgres:smeLL2019_Db@virtual.lab.inf.uva.es:20153/infor_dsdsm` y daremos al botón 'Add'.
- Para la base de datos secundaria en el cuadro 'KEY' indicaremos 'DATABASE_URL_COPY' y en el cuadro 'VALUE' indicaremos `postgres://postgres:smeLL2019_Db@virtual.lab.inf.uva.es:20153/infor_dsdsm_copy` y daremos al botón 'Add'.

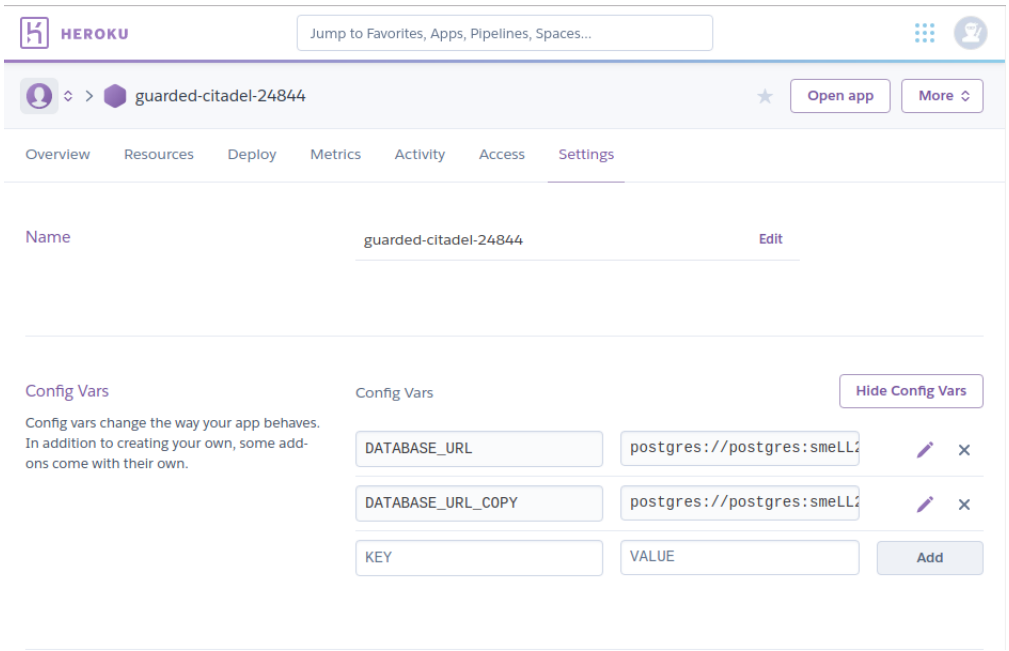

Figura B.1: Añadir variable de entorno en Heroku.

B.3. Servidor base de datos

Primero mostraremos los datos de la máquina virtual con la que poder acceder al servidor de base de datos:

- Host: virtual.lab.inf.uva.es
- Usuario: usuario
- Password: smeLL2019_tfg
- Puerto SSH: 20151

En el servidor de base de datos tendremos que tener instalado PostgreSQL y tenerlo configurado para que el usuario sea *postgres* y contraseña sea *smeLL2019_Db*. Crearemos dos bases de datos, una para la principal denominada 'infor_dsdsm' y otra para la secundaria 'infor_dsdsm_copy'. Cargaremos los scripts de estructura y datos en ambas bases de datos. Estos scripts se encontrarán situados en la carpeta raíz de la máquina.

Para que el servidor pueda permitir conexiones entrantes en este caso desde Heroku, tendremos que realizar la siguiente configuración en el fichero situado en la siguiente ruta '/etc/postgresql/11/main/pg_hba.config' como se muestra a continuación.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             0.0.0.0/0            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            md5
host    replication     all             ::1/128                 md5
```

El fichero situado en '/etc/postgresql/11/main/postgresql.conf' se debe modificar de la siguiente manera.

```
# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 5432                             # (change requires restart)
max_connections = 100                   # (change requires restart)
#superuser_reserved_connections = 3     # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of
 ↪  directories
```

Una vez modificados los archivos, se ejecutará el comando para reiniciar el servicio de PostgreSQL. Des esta manera cargará los nuevos valores.

```
~$ service postgresql restart
```

Apéndice C

Manual de usuario

La pagína principal de la aplicación corresponde con la del login, en la que se muestran las opciones de iniciar sesión y registrarse, como se muestra en la figura C.1.

Para poder acceder a la aplicación, es necesario crear una cuenta de usuario. Para el registro hay un botón **Registrarse** en la misma página de inicio de sesión, que se encuentra en el cuadro situado a la derecha de la vista, el cual muestra un formulario para introducir nuevos datos de un usuario.

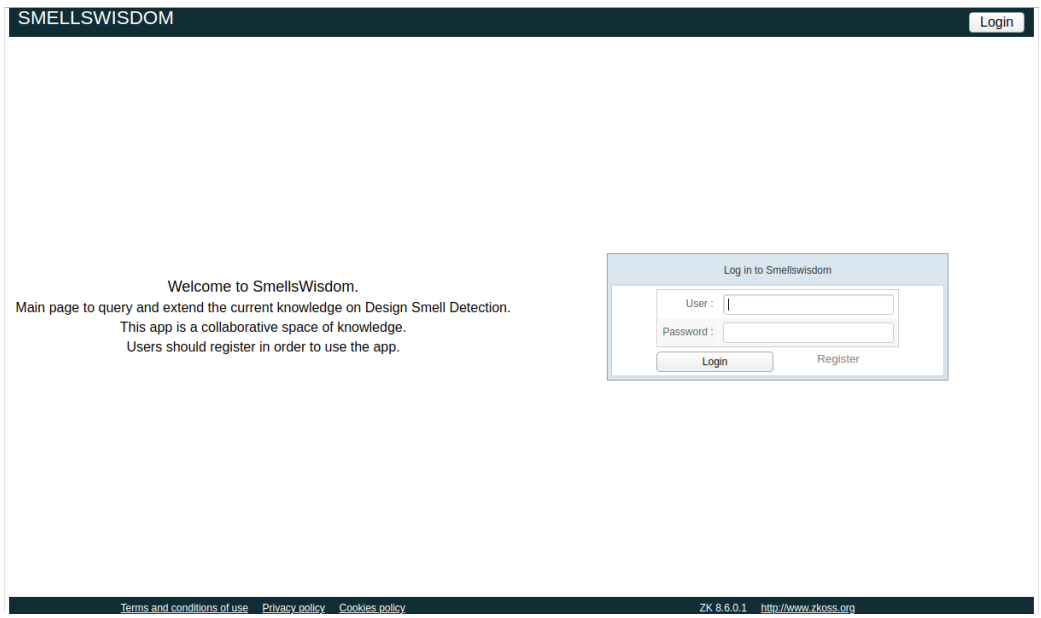

Figura C.1: Página identificación de usuario.

Este formulario se muestra en una vista modal, como podemos ver en la figura C.2, en el cual si no se introducen los datos correctamente, se mostrarán mensajes de error en los campos que requieran rectificación por parte del usuario.

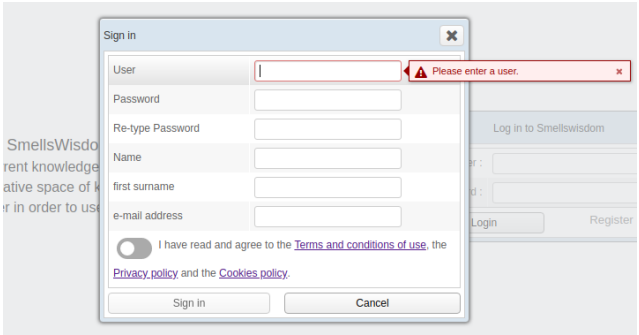

Figura C.2: Página modal de registro.

Creado un nuevo usuario, se mostrará un mensaje indicando que se ha creado correctamente y es necesario activar la cuenta mediante un enlace enviado a la dirección de correo electrónico del usuario registrado. Si no se activa la cuenta, saldrá un mensaje informando que el usuario está desactivado y que tiene que activar su cuenta.

Una vez activada la cuenta, se podrá acceder a la aplicación, la cual nos dirige a la vista de búsqueda de artículos C.3.

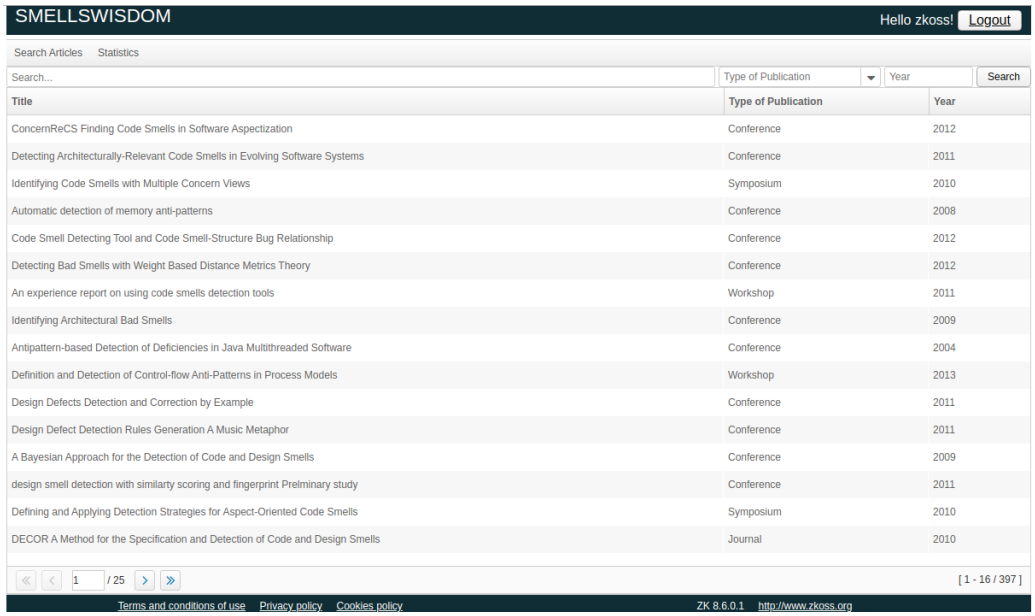

Figura C.3: Página búsqueda artículos.

En la página de búsqueda tenemos, por un lado, una barra de navegación y por otro lado unos filtros para buscar los artículos por título, tipo de publicación y año; también una lista en la que se muestran todos los artículos o los que coinciden con los filtros, además de poder ordenarse por título, tipo de publicación o año. Además, tenemos la posibilidad de ver un detalle general del artículo dando click a un elemento de la lista y la opción de ver en detalle el artículo seleccionado dando al botón **Ver más detalle**, como podemos ver en la siguiente figura C.4.

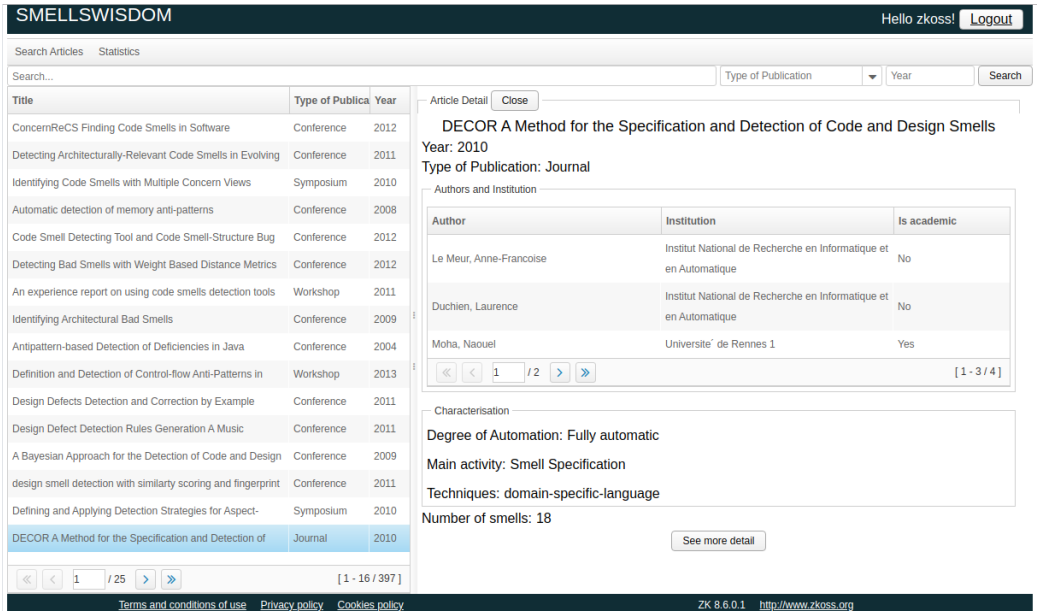

Figura C.4: Página búsqueda artículos, con un artículo seleccionado.

Al ver en detalle un artículo, se mostrará una barra de navegación para ir a las diferentes páginas dedicadas a mostrar la información de los diferentes apartados. Por defecto se muestra el detalle general, como podemos ver en la figura C.5.

En el apartado **Caracterización**, se muestra información sobre los métodos utilizados para la detección de los Design Smells, así como los tipos de enfoque que utiliza, los tipos de artefactos con los que trata y otras actividades.

En el apartado **Smells** se muestran todos los Design Smells de los que trata el artículo. Al dar click en un Design Smell de la lista mostrada, se podrán ver los factores de calidad al que afectan y la posibilidad de ver las herramientas con las que se puede detectar y los proyectos en los que se puede encontrar el Design Smell seleccionado. A través de dos botones **puede ser detectado por herramientas** y **puede ser encontrado en proyectos**, como podemos ver en la figura C.6.

En el apartado **Evidencias** se muestran todas la evidencias de validación realizadas en

el artículo. Al dar click en una evidencia de validación de la lista mostrada, se podrán ver las medidas hechas y los proyectos asociados a la evidencia, como se pude ver en la figura C.7.

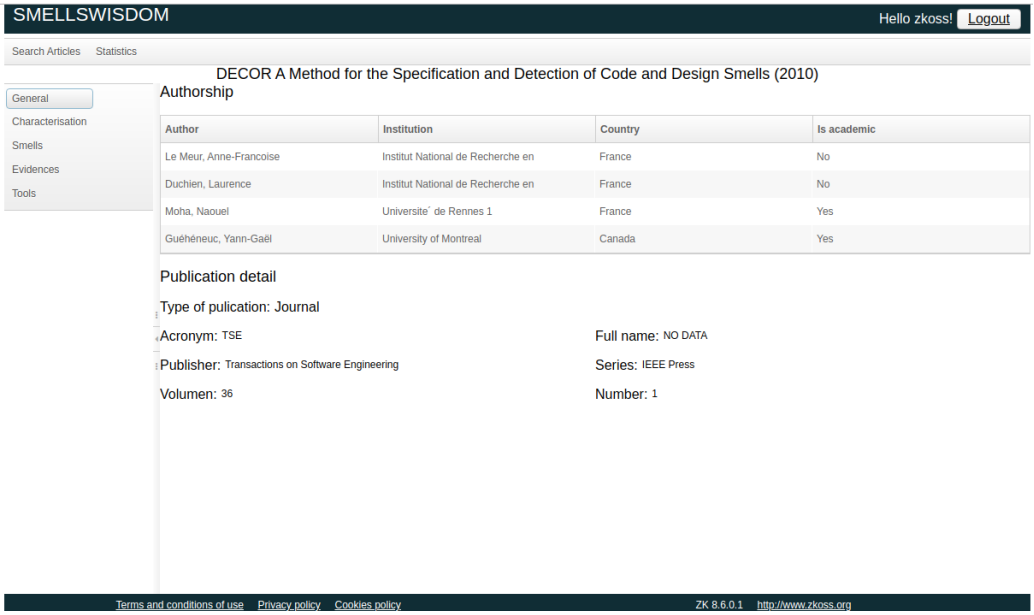

Figura C.5: Página detalle general.

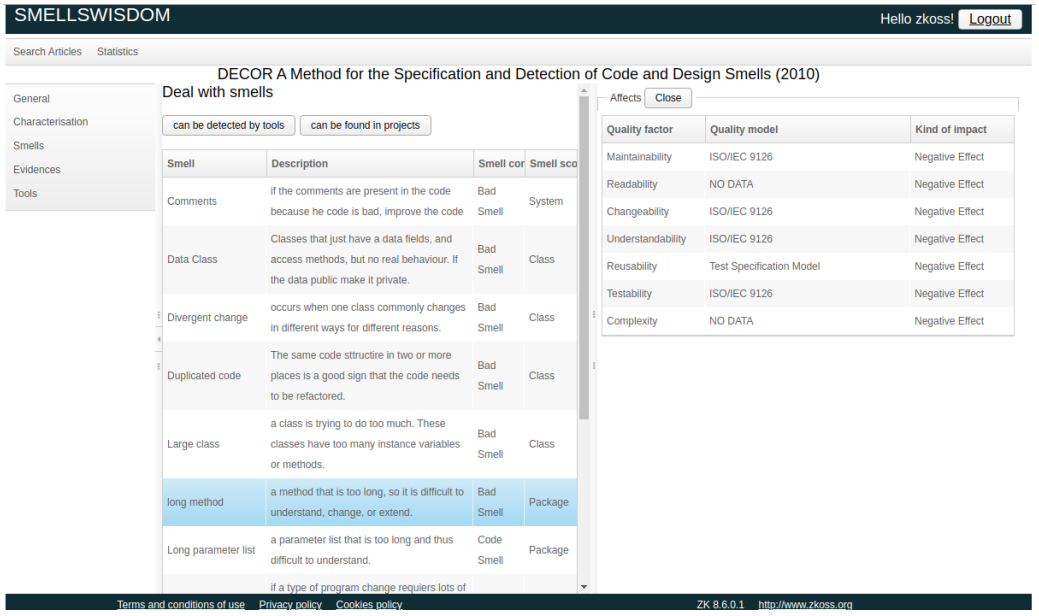

Figura C.6: Página detalle smells.

En el apartado **Tools** se muestran todas las herramientas mencionadas en el artículo. Al dar click en una herramienta se pueden ver los Design Smells que puede detectar. Esto se realiza a través del botón **Smells que puede detectar**, como se pude ver en la figura C.8.

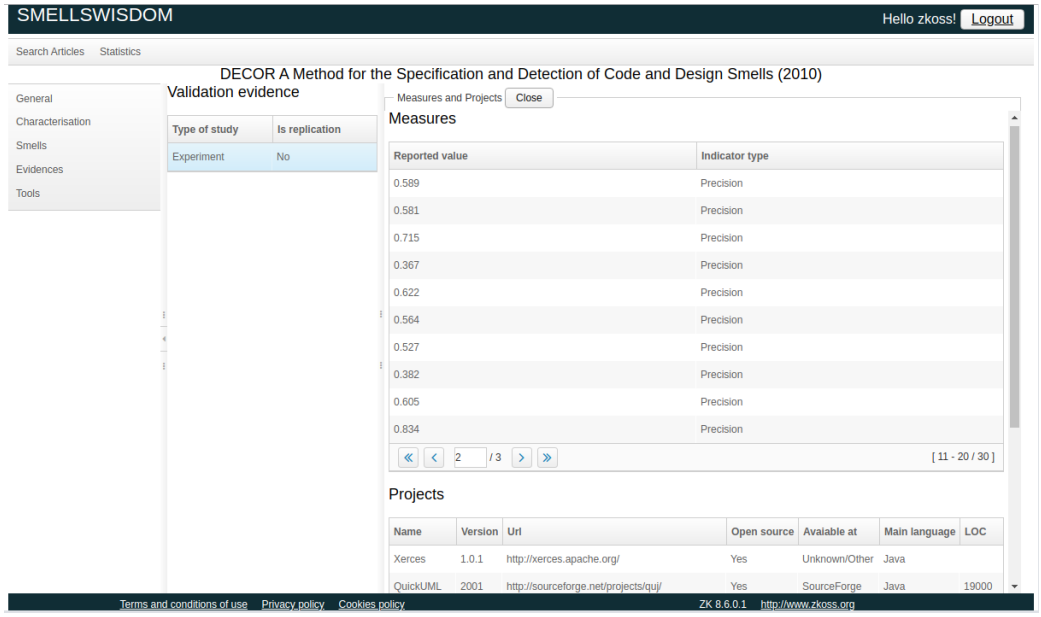

Figura C.7: Página detalle evidencias.

La página de estadísticas muestra distintos gráficos con los datos obtenidos en el mapeo sistemático, como pueden ser: los enfoques y herramientas más utilizados o los Design Smells más detectados. Se muestra en primer lugar la página de estadísticas de publicaciones. En ella hay una barra de navegación para ir viendo las distintas páginas de gráficas estadísticas agrupadas por el tema del que tratan, como se puede ver en la figura C.9.

El administrador podrá introducir nuevos datos al conocimiento. Para ello podrá acceder a la parte de **Nueva contribución**. En esta página hay diferentes apartados para ir introduciendo los datos de un artículo con sus respectivas relaciones con Autoría, Publicación, Caracterización, Design Smells, Evidencias y Herramientas. Al final de la página encontraremos un botón para guardar los nuevos datos.

En la parte de **Información general**, nos permite introducir datos sobre la publicación, datos del artículo como el tipo de publicación y su autoría, como podemos ver en la figura C.10. La autoría de un artículo se puede añadir a través de una página modal, que nos permite introducir nuevos datos o elegir datos ya existentes, como podemos ver en la figura C.11.

En la parte de **Enfoque**, nos permite introducir datos sobre el enfoque como el grado de automatización, técnica utilizada, actividad principal, tipo de enfoque, artefactos y otras

actividades.

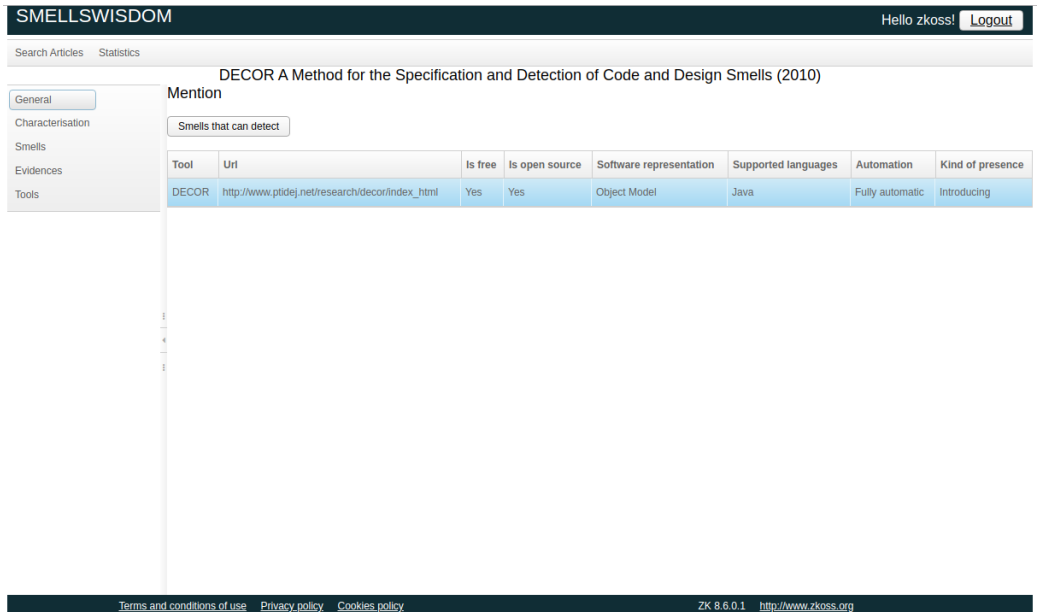

Figura C.8: Página detalle herramientas.

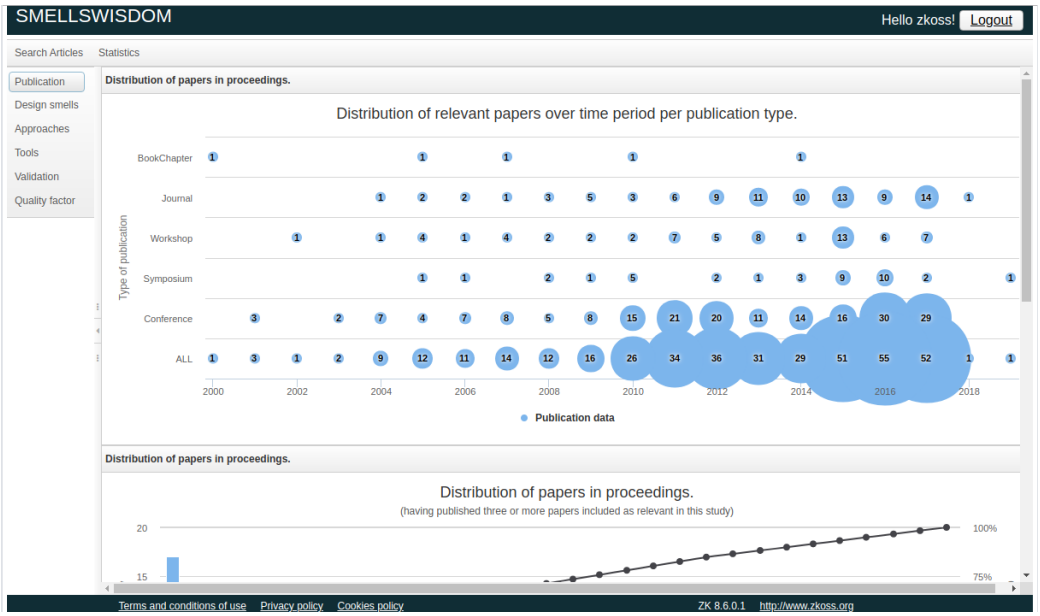

Figura C.9: Página estadísticas publicación.

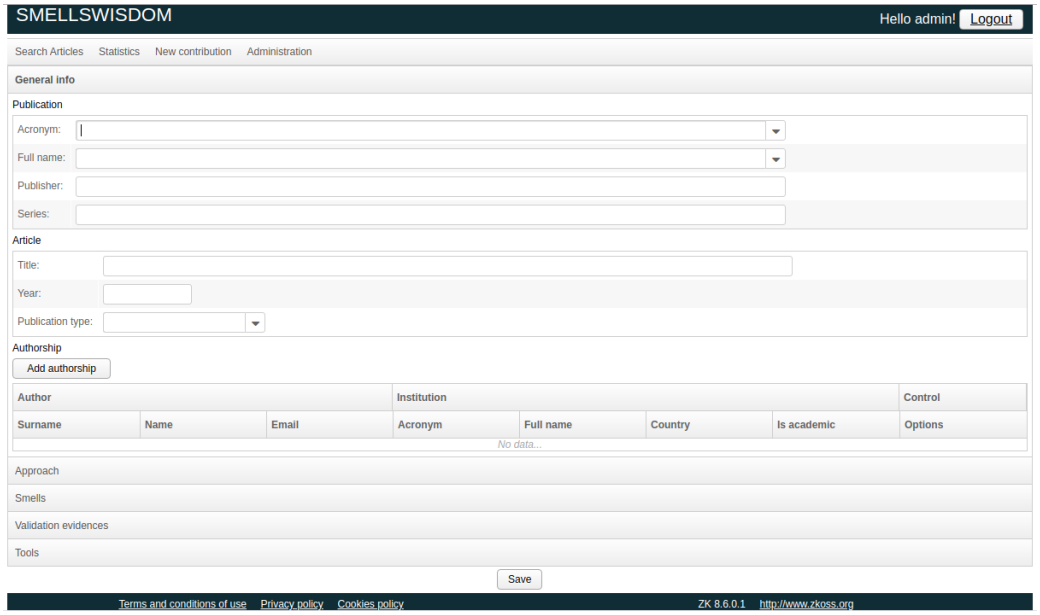

Figura C.10: Página nueva contribución.

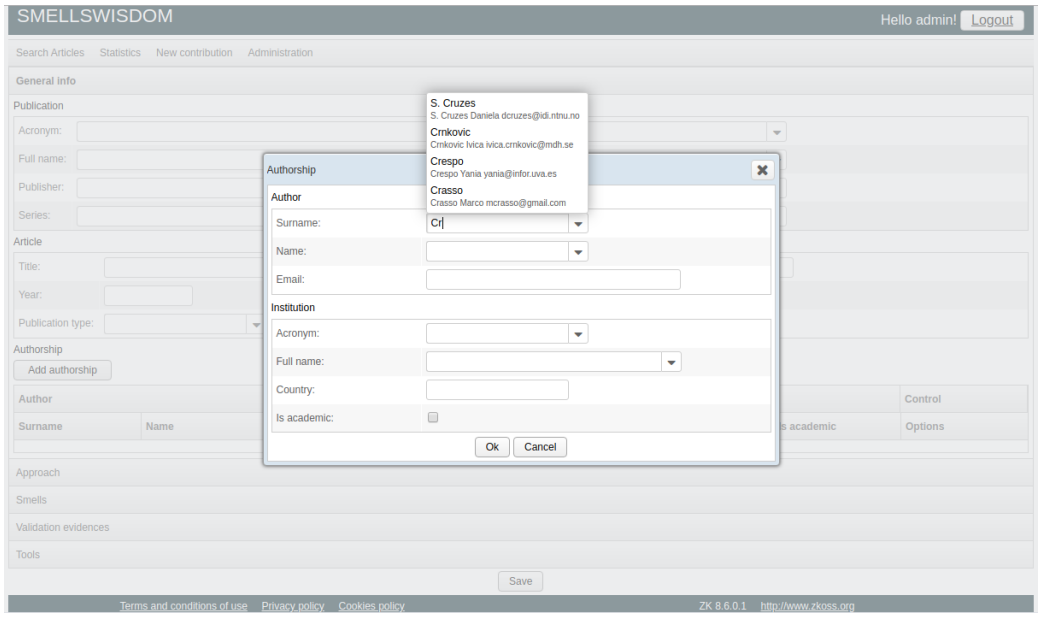

Figura C.11: Página modal autoría.

En la parte de **Smells** se podrán añadir nuevos Design smells o relacionar los ya existentes

con un nuevo artículo. Para ello habrá un botón que nos abra una página modal en la que se introducen todos los datos referentes a un nuevo Design smells, o en su caso seleccionar uno ya existente. Tras añadirlo, podemos seleccionarlo para añadir factores de calidad a los que afecta, proyectos donde se encuentra y las herramientas con las que se puede detectar.

En la parte de **Smells** se podrán añadir nuevos Design smells o relacionar los ya existentes con un nuevo artículo. Para ello habrá un botón que abre una página modal en el que se introducen todos los datos referentes a un nuevo Design smells, o en su caso seleccionar uno ya existente. Tras añadirlo, podemos seleccionarlo para añadir factores de calidad a los que afecta, proyectos donde se encuentra y las herramientas con las que se puede detectar.

En la parte de **Evidencias de validación** se podrán añadir nuevas evidencias de validación para el artículo. Para ello habrá un botón que abra un página modal en la que se introducen los datos de la evidencia. Tras añadirla, podemos seleccionarla para añadir proyectos que la validan y las métricas obtenidas.

En la parte de **Herramientas** se podrán añadir nuevas herramientas que se mencionan en un artículo. Para ello habrá un botón que abra una página modal en la que se introducen los datos de una herramienta o se elige una ya existente. Tras añadirla, podemos seleccionarla para añadir Design smells que puede detectar.

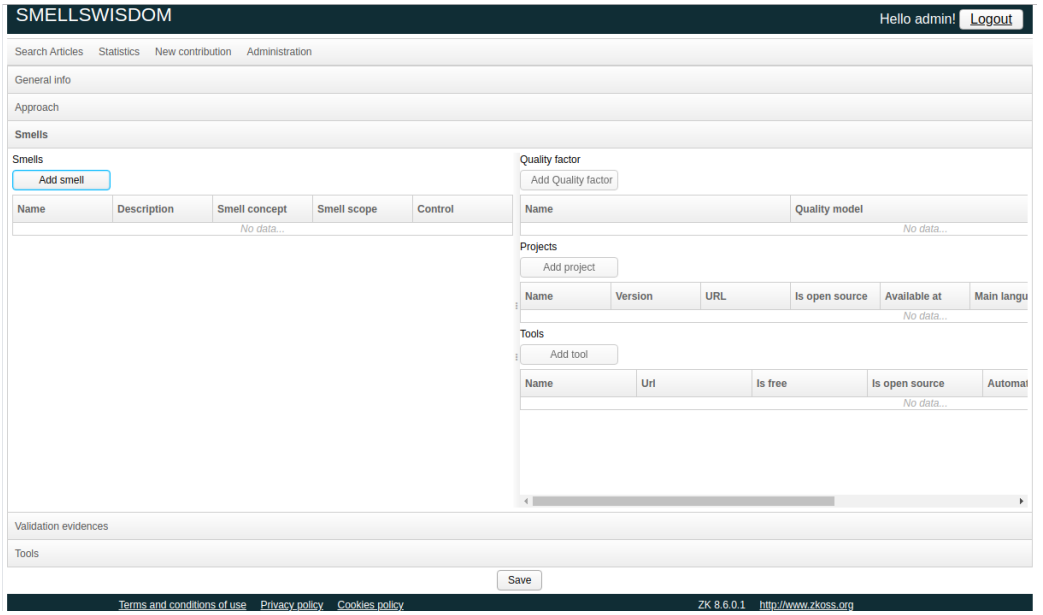

Figura C.12: Página nueva contribución añadir smells.

Apéndice D

Contenidos del CD-ROM

El contenido del CD-ROM que acompaña a esta memoria es el siguiente:

- Carpeta BD: Contiene los scripts sql para la estructura llamada *INFOR_DSDSP_estructura* y los datos *INFOR_DSDSP_datos*.
- Carpeta Código_fuente: Contiene el código fuente completo de la aplicación desarrollada.
- Carpeta Memoria: Contiene la memoria del proyecto en formato PDF, llamado memoria.pdf.

Bibliografía

[1] Alkharabsheh, Khalid and Crespo, Yania and Manso, Esperanza and Taboada, José A., *Software Design Smell Detection: a systematic mapping study*, 2019. Última consulta: 02-01-2020. Disponible en: https://link.springer.com/article/10.1007%2Fs11219-018-9424-8.

[2] Moha, Y. Gueheneuc, L. Duchien, and A. Le Meur., *Decor: A method for the specification and detection of code and design smells.*, 2010. Última consulta: 30-11-2019. Disponible en: http://dx.doi.org/10.1109/TSE.2009.50,doi:10.1109/TSE.2009.50.

[3] R. C. Martin., . *Agile Software Development, Principles, Patterns, and Practices.*, Addison-Wesley, 2003.

[4] M. Fowler., *Refactoring: improving the design of existing code.*, Addison-Wesley, 2003.

[5] S. Hassaine, F. Khomh, Y. Gueheneuc, and S. Hamel., *Ids: An immuneinspired approach for the detection of software design smells.*, 2010. Última consulta: 30-11-2019. Disponible en: https://ieeexplore.ieee.org/document/5655670.

[6] García-Peñalvo, F. J., *Revisión sistemática de literatura en los Trabajos de Final de Máster y en las Tesis Doctorales.*, Universidad de Salamanca, 2017.

[7] Saenz, A, *Leer e interpretar una revisión sistemática.*, BOLETÍN DE LA SOCIEDAD DE PEDIATRÍA DE ASTURIAS, CANTABRIA, CASTILLA Y LEÓN , 2001.

[8] CASCADE PROJECT, *Mapping in literature reviews.*, 2012. Última consulta: 11-12-2019. Disponible en: https://as.exeter.ac.uk/media/universityofexeter/academicservices/educationenhancement/cascade/Mapping_in_literature_reviews.pdf

[9] Zhang, M., Hall, T. and Baddoo, N, *Code Bad Smells: a review of current knowledge.*, 2011. Última consulta: 02-12-2019. Disponible en: https://doi.org/10.1002%2Fsmr.521

[10] Rattan, D., Bhatia, R. K. and Singh, M, *Software clone detection: a systematic review.*, 2013. Última consulta: 02-12-2019. Disponible en: https://doi.org/10.1016/j.infsof.2013.01.008

[11] Rasool, G. and Arshad, *A review of code smell mining techniques.*, 2015. Última consulta: 03-12-2019. Disponible en: `https://doi.org/10.1002%2Fsmr.1737`

[12] Fernandes, E., Oliveira, J., Vale, G., Paiva, T. and Figueiredo, E, *A review-based comparative study of bad smell detection tools.*, In Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, 2016.

[13] Singh, S. and Kaur, S, *A systematic literature review: Refactoring for disclosing code smells in object-oriented software.*,Ain Shams Engineering Journal, 2017.

[14] Gupta, A., Suri, B. and Misra, S, *A systematic literature review: code bad smells in java source code.*,In International Conference on Computational Science and Its Applications, 2017.

[15] *Qué es SCRUM*, 2019. Última consulta: 15-10-2019. Disponible en: `https://proyectosagiles.org/que-es-scrum/`.

[16] Fuente, Redondo, Pablo de la., *Presentación de SCRUM.*, 2019. Última consulta: 14-10-2019. Disponible en: `https://aulas.inf.uva.es/`.

[17] Fuente, Redondo, Pablo de la., *Gestión de riesgos*, 2019. Última consulta: 14-10-2019. Disponible en: `https://aulas.inf.uva.es/`.

[18] Pivotal Software, Inc, *Pivotal Tracker*. Última consulta: 28-11-2019. Disponible en: `https://www.pivotaltracker.com/`.

[19] Software Freedom Conservancy *Git*. Última consulta: 23-11-2019. Disponible en: `https://www.git-scm.com/`.

[20] GitLab Inc, *GitLab*. Última consulta: 16-10-2019. Disponible en: `https://about.gitlab.com/`.

[21] ChangeVision, Inc, *Astah*. Última consulta: 16-10-2019. Disponible en: `http://astah.net/`.

[22] Eclipse Foundation, Inc, *Eclipse*. Última consulta: 23-11-2019. Disponible en: `https://www.eclipse.org/`.

[23] Overleaf, *Overleaf*. Última consulta: 23-11-2019. Disponible en: `https://es.overleaf.com`.

[24] salesforce.com, inc., *Heroku*. Última consulta: 26-11-2019. Disponible en: `https://www.heroku.com/`.

[25] Balsamiq Studios, LLC, *Quick and Easy Wireframing Tool*. Última consulta: 26-11-2019. Disponible en: `https://balsamiq.com/wireframes/`.

[26] The Apache Software Foundation, *Maven*. Última consulta: 23-11-2019. Disponible en: `https://maven.apache.org/`.

[27] Potix Corporation, *ZK Framework*. Última consulta: 13-12-2019. Disponible en: `https://www.zkoss.org/`.

[28] Pivotal Software, Inc, *Spring*. Última consulta: 21-11-2019. Disponible en: `https://spring.io/`.

[29] Pivotal Software, Inc, *Spring Security*. Última consulta: 16-10-2019. Disponible en: `https://spring.io/projects/spring-security`.

[30] Hibernate, *Hibernate*. Última consulta: 16-10-2019. Disponible en: `https://hibernate.org/`.

[31] The PostgreSQL Global Development Group, *PostgreSQL*. Última consulta: 17-10-2019. Disponible en: `https://www.postgresql.org/`.

[32] CareWeb Framework, *Carewebframework-highcharts*. Última consulta: 16-10-2019. Disponible en: `https://github.com/carewebframework/carewebframework-highcharts`.

[33] JUnit, *JUnit*. Última consulta: 31-12-2019. Disponible en: `https://junit.org/junit4/`.

[34] Szczepan Faber y col., *Mockito*. Última consulta: 16-12-2019. Disponible en: `https://site.mockito.org/`.

[35] Potix Corporation, *ZK Essentials*. Última consulta: 06-11-2019. Disponible en: `http://books.zkoss.org/zkessentials-book/master/index.html`.

[36] Potix Corporation, *Create and Run Your First ZK Application with Eclipse and Maven*. Última consulta: 23-11-2019. Disponible en: `https://www.zkoss.org/wiki/ZK%20Installation%20Guide/Quick%20Start/Create%20and%20Run%20Your%20First%20ZK%20Application%20with%20Eclipse%20and%20Maven`.

[37] Rod Johnson y col, *Spring Security*. Última consulta: 17-10-2019. Disponible en: `https://www.zkoss.org/wiki/ZK_Developer's_Reference/Integration/Security/Spring_Security`.

[38] Potix Corporation, *Heroku*. Última consulta: 19-10-2019. Disponible en: `https://www.zkoss.org/wiki/ZK_Installation_Guide/Setting_up_Servers/Heroku`.

[39] salesforce.com, inc., *Getting Started on Heroku with Java*. Última consulta: 25-11-2019. Disponible en: `https://devcenter.heroku.com/articles/getting-started-with-java`.

[40] salesforce.com, inc., *Deploying Tomcat-based Java Web Applications with Webapp Runner*. Última consulta: 25-11-2019. Disponible en: `https://devcenter.heroku.com/articles/java-webapp-runner`.

[41] Ivan Cachicatari, *Migrar una base de datos MySQL a PostgresQL*. Última consulta: 19-10-2019. Disponible en: `http://www.latindevelopers.com/articulo/migrar-de-mysql-a-postgresql/`.

[42] salesforce.com, inc., *Heroku Postgres*. Última consulta: 25-11-2019. Disponible en: `https://devcenter.heroku.com/articles/heroku-postgresql#provisioning-heroku-postgres`.

[43] TACANDE.NET, *Aviso legal, política de privacidad y ley de cookies: ¿qué necesita tu página web?*. Última consulta: 21-10-2019. Disponible en: `https://tacande.net/aviso-legal-politica-de-privacidad-y-ley-de-cookies-que-necesita-tu-pagina-web`.

[44] ELCORRIOL.COM, *¿Cómo cumplir la RGPD o GDRPR en mi página web?*. Última consulta: 23-10-2019. Disponible en: `https://www.elcorriol.com/es/cumplir-rgpd-en-pagina-web/`.

[45] ZHIGHCHARTS, *ZHighCharts*. Última consulta: 21-10-2019. Disponible en: `http://zhighcharts.appspot.com/index.html`.

[46] DIEGO SILVA, *Convertidor de tipo de atributo con Java Persistence API*. Última consulta: 18-11-2019. Disponible en: `https://www.oracle.com/technetwork/es/articles/java/convertidor-java-persistence-api-2906142-esa.html`.

[47] ADRIAN MATEI, *How to setup multiple data sources with Spring and JPA*. Última consulta: 21-11-2019. Disponible en: `https://www.codepedia.org/ama/how-to-setup-multiple-data-sources-with-spring-and-jpa/`.

[48] GOOGLE, *Consulta tu correo de Gmail a través de otras plataformas de correo electrónico*. Última consulta: 18-11-2019. Disponible en: `https://support.google.com/mail/answer/7126229?p=BadCredentials&visit_id=637096962509399291-3672409167&rd=2#cantsignin`.

[49] POTIX CORPORATION, *ZK Framework Documentation*. Última consulta: 13-12-2019. Disponible en: `https://www.zkoss.org/documentation`.

[50] ROD JOHNSON Y COL, *Spring Framework Documentation*. Disponible en: `https://docs.spring.io/spring/docs/current/spring-framework-reference/`.

[51] VLAD MIHALCEA Y COL, *Hibernate ORM 5.2.18. Final User Guide*. Última consulta: 21-11-2019. Disponible en: `https://docs.jboss.org/hibernate/orm/5.2/userguide/html_single/Hibernate_User_Guide.html`.

[52] POTIX CORPORATION, *MVC*. Última consulta: 13-12-2019. Disponible en: `https://www.zkoss.org/wiki/ZK%20Developer's%20Reference/MVC`.

[53] POTIX CORPORATION, *JPA*. Última consulta: 21-11-2019. Disponible en: `https://www.zkoss.org/wiki/ZK%20Developer's%20Reference/Integration/Persistence%20Layer/JPA`.

[54] JOSEPABLOSARCO, *Lenguaje Gherkin*. Última consulta: 12-12-2019. Disponible en: `https://josepablosarco.wordpress.com/2015/03/11/lenguaje-gherkin/`.

[55] WILLIAM FERNANDO, *Programación por capas*. Última consulta: 30-11-2019. Disponible en: `https://es..wikipediaorg/wiki/Programaci%C3%B3n_por_capas`.

[56] IAN YT TSAI, *Starting A Web Application Based On ZK CDI JPA and Jetty*. Última consulta: 30-11-2019. Disponible en: `https://www.zkoss.org/wiki/Small_Talks/2012/Aug/Starting_A_Web_Application_Based_On_ZK_CDI_JPA_and_Jetty`.

[57] Tom M. Yeh, *Architecture Overview*. Última consulta: 13-12-2019. Disponible en: `https://www.zkoss.org/wiki/ZK_Developer's_Reference/Overture/Architecture_Overview`.

[58] Rod Johnson y col, *Introduction to Spring Framework*. Última consulta: 21-11-2019. Disponible en: `https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/overview.html`.

[59] Vlad Mihalcea y col, *Hibernate ORM 5.2.18. Final User Guide*. Última consulta: 21-11-2019. Disponible en: `https://docs.jboss.org/hibernate/orm/5.2/userguide/html_single/Hibernate_User_Guide.html`.

[60] Tom M. Yeh, *MVC*. Última consulta: 10-12-2019. Disponible en: `https://www.zkoss.org/wiki/ZK%20Developer's%20Reference/MVC`.

[61] Alverik, *Patrón DAO (Data Access Object)*. Última consulta: 21-11-2019. Disponible en: `http://rinconprogramadorcito.blogspot.com/2010/11/patron-dao-data-access-object.html`.