Universidad Zaragoza

Trabajo Fin de Grado

Sistema para la integración de productos fitosanitarios

Autor

Catalin Dumitrache

Director

Francisco Javier López Pellicer

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. _____,

con nº de DNI _____ en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) _____, (Título del Trabajo)

_____

_____

_____

_____

_____

_____,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, _____

Fdo: _____

AGRADECIMIENTOS

Agradezco a ...

RESUMEN EJECUTIVO

Este proyecto se centra en presentar y demostrar un modelo de integración de información relativa a productos fitosanitarios y su aplicación en diferentes productos agrícolas provenientes de fuentes heterogéneas en un esquema único y escalable.

Para lograr esto, se han seleccionado diferentes fuentes de datos sobre productos fitosanitarios, presentes en distintos formatos y se han analizado varias tecnologías de integración para elegir el stack tecnológico que más se adecue al problema en cuestión.

Así pues, en este proyecto se han usado herramientas de almacenamiento elegidas bajo un criterio de escalabilidad futura y herramientas de procesado de datos bajo el criterio de proporcionar soporte al mayor abanico de fuentes heterogéneas posibles. *Apache Hadoop* fue el indicado como sistema responsable del almacenamiento en conjunto con *Apache Hive* para facilitar el acceso a los datos mediante una aproximación relacional. Para el procesado de los datos se ha empleado *Talend Big Data*.

Para demostrar la viabilidad del sistema, se ha desarrollado un prototipo funcional que recoge datos de los productos fitosanitarios de España y Europa y complementa la información de ambas fuentes, constituyendo un primer paso hacia ese modelo compartido donde varias fuentes heterogéneas concuerdan en un mismo esquema congruente. Los datos se pueden ver mediante una aplicación web desarrollada con *JHipster*, un generador de proyectos ligero sobre Java capaz de desplegar rápidamente una aplicación con una cuidada GUI.

Palabras clave: Productos fitosanitarios, Hadoop, Hive, Integración, ETL, Base de datos, JHipster, MySQL, Modelo único, BigData, Escalabilidad, Sqoop, Fuentes heterogéneas.

Tabla de contenidos

1. Introducción **1**

1.1. Estructura del documento 2

2. Contexto, motivación y restricciones **3**

2.1. Contexto y necesidades reales 3

2.2. Motivación y objetivos 5

2.3. Restricciones 7

3. Análisis **8**

3.1. Análisis del problema 8

3.2. Propuesta de valor e intermediarios 8

3.3. Análisis de requisitos - Técnica *MoSCoW* 10

3.4. Análisis de riesgos 12

3.5. Análisis del contexto tecnológico 13

3.6. Elección del Stack Tecnológico 15

3.7. Captura de requisitos final 17

4. Diseño **19**

4.1. Diseño conceptual 19

4.2. Diseño final 20

4.3. Arquitectura del sistema 22

4.3.1. Diagrama de despliegue. 22

4.3.2. Diagrama de componente y conector. 23

4.3.3. Diagrama de clases y paquetes. 24

4.3.4. Diagrama de secuencia. 24

4.3.5. Diagrama de datos. 26

4.4. Estrategia de integración y expansión 28

5. Implementación **30**

5.1. Prueba de concepto 30

5.2. Prototipo real 37
5.3. Problemas encontrados 41

6. Gestión **47**
6.1. Metodología 47
6.2. Pautas e imposiciones 48
6.3. Organización y control de versiones 49
6.4. Control de esfuerzos 51
6.5. Estimación del coste 53

7. Conclusiones **55**
7.1. Resultados y objetivos 55
7.2. Conocimientos adquiridos 57

8. Bibliografía **59**

Glosario **63**

Lista de Figuras **66**

Lista de Tablas **67**

Anexos **68**

A. Datos **69**
A.1. Esquema de datos 69
A.2. Flujo de datos 71

B. Clientes **72**
B.1. Clientes potenciales 72

C. Análisis **75**
C.1. Análisis de riesgos completos 75
C.2. Análisis de diseños alternativos 82

D. Gestión **85**
D.1. Cálculo del coste de la hora de trabajo del alumno 85

Capítulo 1

Introducción

Actualmente el proceso de consulta del uso y aplicación de los productos fitosanitarios sobre determinados productos resulta una tarea en ocasiones tediosa, sobre todo, cuando se trata de comprobar las especificaciones y regulaciones que imponen diferentes países en operaciones de importación o exportación de determinados productos agrícolas. Hoy en día, una persona que quiere comercializar estas sustancias debe tener en cuenta varios factores; en primer lugar, existen varios manuales extensos que recogen medidas de seguridad, buenas prácticas y pautas sobre la aplicación de los productos fitosanitarios. Dichos manuales se deben cumplir en todo momento [manual seguridad, aplicación fitosanitarios, buenas prácticas]. No solo eso, sino que por otra parte, las bases de datos o almacenes que recogen la información sobre productos fitosanitarios en muchas ocasiones no están bien gestionadas, son difíciles de encontrar, la información se presenta en formatos heterogéneos e incluso se puede encontrar desactualizada. Los avances conseguidos en este TFG pretenden reducir la complejidad de esa tarea de búsqueda de información acerca de productos fitosanitarios, facilitando a una persona el acceso a un esquema común con toda la información centralizada y actualizada.

El objetivo principal de este proyecto es **proponer y validar un proceso de recogida, transformación y presentación de la información sobre productos fitosanitarios con el resultado en forma de esquema compartido estandarizable** que pueda beneficiar tanto a agricultores como a instituciones nacionales o internacionales, **y facilitar la consulta de dicha información de manera más rápida, simple y accesible que los métodos actuales**.

Entre los retos planteados figuran:

- Desarrollar un sistema capaz de visualizar los datos ya integrados nutriéndose únicamente de las fuentes originales sin intervención de una persona en el proceso

- La consistencia de los datos y su almacenamiento tanto en formato original como

en su formato procesado e integrado en la versión final

- El diseño de una solución escalable y actualizada en todo momento
- La posibilidad de añadir características de trazabilidad y mantenimiento a la aplicación.

Estructura del documento

Este documento se presenta dividido en varios bloques conceptuales; el primero abarca los primeros tres apartados (Resumen ejecutivo, Introducción y Definiciones) y se corresponde a una introducción al trabajo realizado. En él, se ofrece una visión completa y resumida del problema junto con su solución y se dan algunas definiciones técnicas de algunos de los términos empleados en esta memoria. El bloque de análisis abarca los apartados 4 y 5 (Phytoscheme y Análisis) y presentan el trabajo de análisis que se llevó a cabo, desde el estudio del entorno, hasta el análisis del stack tecnológico, pasando por el de riesgos y la captura de requisitos. El tercer bloque se corresponde a la solución desarrollada en sí; abarca los apartados 6 y 7 (Diseño e implementación) y se plasma el trabajo desde las fases tempranas del desarrollo de la solución hasta el momento de la finalización del software como solución tecnológica al problema. Los últimos bloques se corresponden a los detalles de gestión del proyecto, a las conclusiones tanto del proyecto como del alumno a nivel personal y a los diferentes anexos recogidos durante toda la duración del TFG.

Capítulo 2

Contexto, motivación y restricciones

Contexto y necesidades reales

Los productos fitosanitarios son un elemento imprescindible en la producción agrícola, tanto en los sistemas convencionales de agricultura como en los sistemas de agricultura integrada o ecológica. Sin su existencia, muchos cultivos de las zonas de producción de mayor interés económico y social serían inviables hoy en día debido a los estragos potenciales de las diferentes clases de plagas.

No obstante, el uso de dichos productos fitosanitarios debe estar regulado ya que una aplicación indebida de los mismos puede tener otros efectos no deseables. Dichos efectos de ninguna manera deben suponer un peligro para la salud humana y tampoco riesgos inaceptables para el medio ambiente.

Por ello el Estado sólo aprueba la comercialización de aquellos productos fitosanitarios que sean útiles para combatir las plagas pero no impliquen otros riesgos colaterales. Para que un producto fitosanitario pueda comercializarse debe estar inscrito necesariamente en el Registro Oficial de Productos Fitosanitarios - *Página web del ministerio de agricultura y pesca, alimentación y medio ambiente de España* [1]

Es, por tanto, necesario y casi obligatorio que la información del Registro de Productos Fitosanitarios llegue de manera precisa a todos los implicados en el área del uso de los productos fitosanitarios

La *Directiva 2009/128/CE* [2] establece el marco de la actuación comunitaria para conseguir un uso sostenible de los productos fitosanitarios. Esta Directiva implica, por ejemplo, la obligación del registro del uso de productos fitosanitarios. Un ejemplo del desarrollo de esta Directiva es el *Cuaderno de Explotación* [3]. Este cuaderno aglutina de manera ordenada y armonizada todos los elementos que deberán registrar los titulares de las explotaciones agrícolas, con el objetivo de facilitar el cumplimiento

de la Directiva.

Actualmente, hay varias empresas que ofrecen aplicaciones (p. ej. *aGROSLab* [4], *Agricolum* [5] o el *Cuaderno de Campo Agronev* [6]) que implementan el Cuaderno de Explotación. Un valor añadido que suelen incorporar estas aplicaciones es una base de datos con información acerca de los productos fitosanitarios autorizados. El problema al que se enfrentan estas empresas es que esta información no se publica de forma uniforme en toda la Unión Europea. Es decir, hay al menos un publicador por país miembro, la información publicada es heterogénea y los formatos normalmente son difícilmente accesibles. Además, a nivel Europeo, hay una *base de datos de referencia* [7] de las restricciones más o menos comunes en el uso de productos fitosanitarios. Dado que no hay un estándar de publicación establecido, una consecuencia adicional de esta situación es que es complicado verificar si un producto tratado con una serie de productos fitosanitarios en un país miembro puede ser exportado a otro, ya que la normativa fitosanitaria aplicable puede diferir.

En cuanto a la importación de productos desde un país miembro de la Unión Europea, el artículo 52 del *Reglamento (CE) 11/07/2009* [8] se refiere a este trámite como comercio paralelo y se especifica que para poder llevarlo a cabo, el Estado Miembro donde se desee introducir deberá determinar que el producto fitosanitario es idéntico en composición a otro ya autorizado en su territorio al que se denominará "producto de referencia". Para realizar el trámite, actualmente el proceso consta en rellenar la *solicitud* del certificado [9] (disponible en la página web del ministerio), entregarla a la Subdirección General de Higiene Vegetal y Forestal y esperar a que sea aprobada - *Preguntas frecuentes Mapama* [10]

Los principales problemas actuales de este proceso vienen por una parte en el lado del agricultor/empresa que solicita la importación puesto que no solo el tiempo de respuesta en ocasiones puede ser muy largo (de hasta de 45 días - Artículo 52, punto 2 del Reglamento (CE) 11/07/2009) sino que además, el agricultor o bien tiene poca información acerca de los productos permitidos en una importación/exportación o el acceso a dicha información es bastante tedioso y complicado. Por otra parte en el lado de la institución certificadora encargada de comprobar la solicitud el proceso no es del todo eficiente debido al hecho de tener que verificar manualmente si el producto cumple con los requisitos de composición del correspondiente producto en el país de importación/exportación. Estos problemas se pueden observar claramente en el siguiente diagrama, que muestra un flujo típico en un trámite de comercio paralelo entre un agricultor/empresa que quiere importar un producto a un determinado país, en este caso, España:

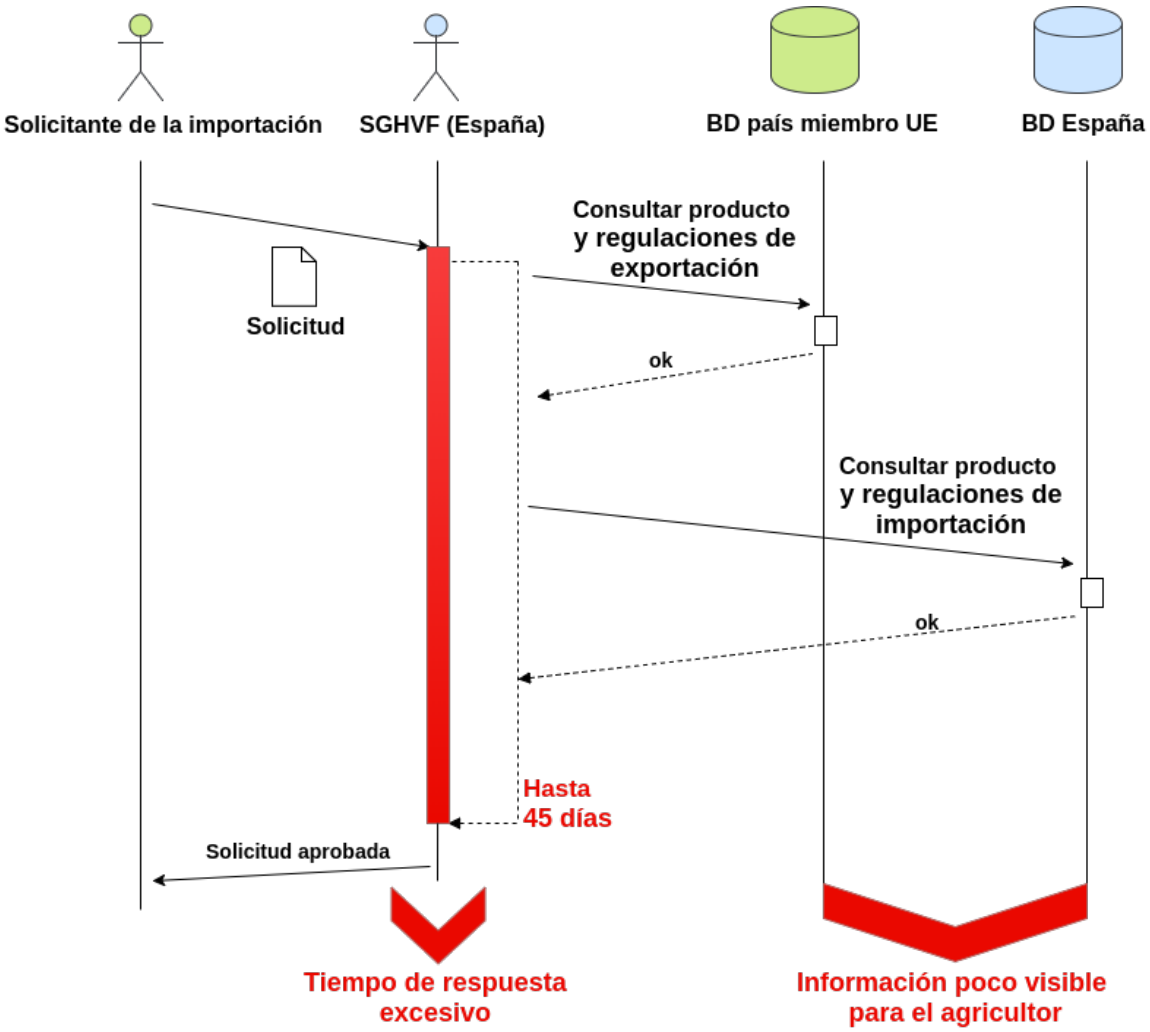


Figura 2.1: Diagrama de flujo del proceso actual de importación de un producto fitosanitario

*SGHVF = Subdirección General de Higiene Vegetal y Forestal

Es razonable desarrollar un sistema que provea un esquema único donde los datos estuvieran integrados y congruentes entre los diferentes países de la unión europea para facilitar un consumo posterior por las aplicaciones, e, incluso, directamente por los agricultores.

Motivación y objetivos

Habiendo visto el panorama descubierto en el apartado anterior, era evidente que se podían introducir mejoras al proceso actual que pueden beneficiar a las partes involucradas (tanto a agricultores y empresas como a las instituciones reguladoras). Con ese objetivo en mente, se propone desarrollar una aplicación prototipo que

valide el proceso de integración de la información de diferentes países en un esquema único que recogiese los datos que de otra manera tendrían que ser recopilados manualmente, con largos tiempos de espera y con una incertidumbre por parte de los agricultores/empresas en lo que a datos sobre productos fitosanitarios respecta.

El siguiente diagrama muestra el potencial proceso que tendría que seguir una persona interesada en realizar una importación y los problemas que sería capaz de solucionar el desarrollo de la aplicación planteada en este proyecto:

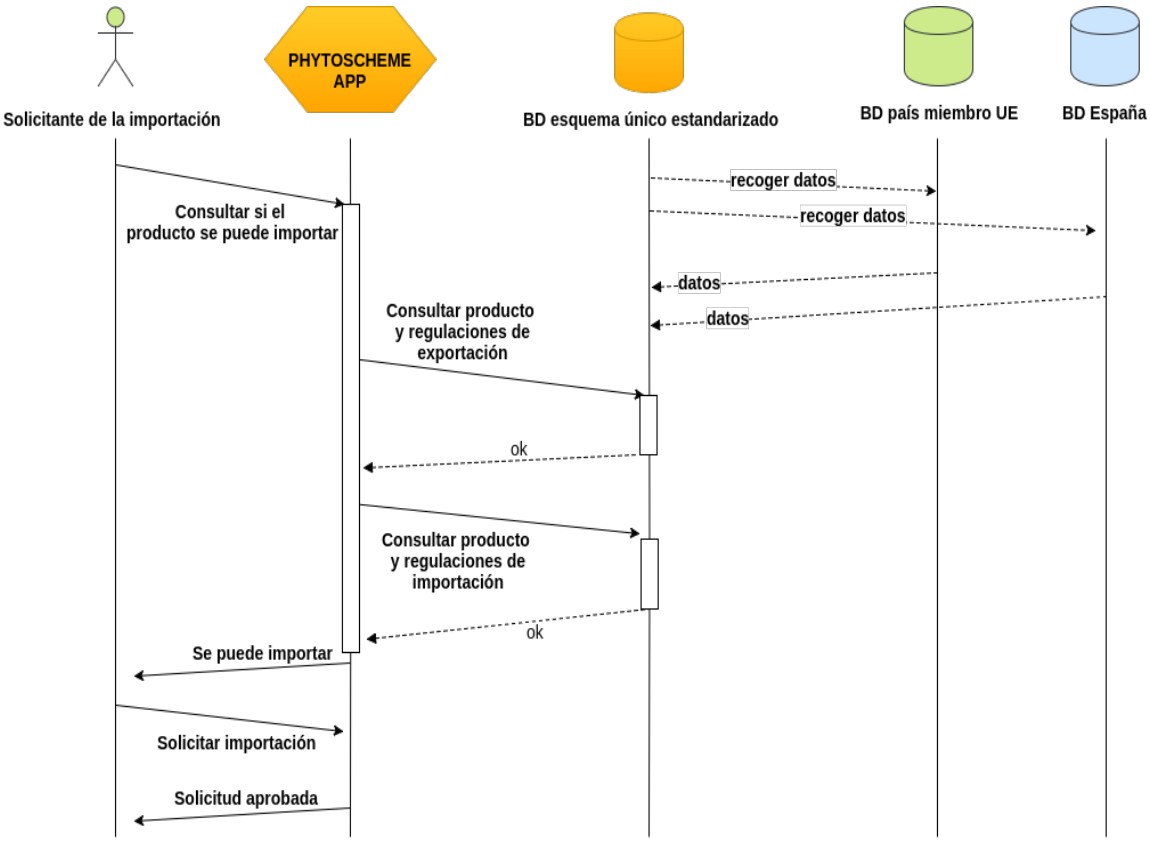


Figura 2.2: Diagrama de flujo del potencial proceso de importación de un producto fitosanitario

Se puede apreciar que con este desarrollo se conseguiría no solo reducir los plazos de respuesta al agricultor/empresa sino también una mayor transparencia en la consulta de los datos, puesto que al estar integrados en un modelo único, el agricultor/empresa o los interesados podrían tener acceso a toda la información y consultar cualquier aspecto que de otra manera habría sido prácticamente inaccesible. No solo eso, sino que además, al conseguir un modelo potencialmente estandarizado, no haría falta del trabajo manual y tedioso de comprobación de los requisitos por parte de la Subdirección General de Higiene Forestal y Vegetal sino que sería el propio sistema el encargado de comprobar si la petición del solicitante cumple con la *legislación fitosanitaria* vigentede cada país.

Uno de los aspectos a tener en cuenta al comienzo del proyecto fue el hecho de que

para que la solución se pudiera aplicar a un nivel real del problema, se necesitaba una gran capacidad de almacenamiento. Los datos deberían almacenarse periódicamente y además, potencialmente podrían provenir de multitud de fuentes, en diferentes formatos, con unos tamaños variables y constantes actualizaciones. Muchas fuentes equivalen a muchos datos, y muchos datos equivalen a la necesidad de un sistema de almacenamiento capaz de soportar toda esa carga.

Unido a esto, la elección de dicho sistema de almacenamiento suponía un punto crítico, puesto que no bastaría con cualquier base de datos sino que tendría que cumplir ciertas características como la posibilidad de guardar los datos en sus formatos originales, la necesidad de generar una jerarquía para su almacenamiento, o la posibilidad de integración con los trabajos de transformación y procesado de los datos.

Por ello, el proyecto tuvo como objetivo principal desarrollar una solución prototipo para demostrar la validez de los procesos y herramientas involucradas en la integración de varias fuentes heterogéneas de productos fitosanitarios en un modelo único.

Restricciones

Teniendo en cuenta que el proyecto actual es un TFG y no un proyecto comercial, se tienen que tomar en cálculo una serie de restricciones que vienen dadas tanto por la naturaleza del proyecto como por los partícipes del mismo. En primer lugar, siendo un TFG, debe realizarse una cantidad de esfuerzo equivalente a 12 ECTS.. Dado que el desarrollo realizado en este proyecto está enfocado a formar parte de un plan mucho mayor, donde el trabajo realizado se retomará y ampliará en futuros TFG, desde el principio se delimitaron aquellas características que se deseaban incluir en el proyecto y también aquellas que no corresponden a esta iteración. Siendo una fase temprana de dicho plan maestro, a esta fase le correspondía la parte de validación del modelo, de las tecnologías empleadas y de la viabilidad del producto, no siendo primordial tener un producto final robusto sino demostrar que las tecnologías en su totalidad se complementaban y funcionaban acorde a lo esperado.

Otro factor restrictivo dada la naturaleza del proyecto es el hecho de que existe un director, que puede marcar las pautas de desarrollo, sugerir e imponer metodologías de trabajo o incluso herramientas del stack tecnológico que pueden suponer una ventaja o una desventaja en el desarrollo del proyecto. El alumno debe ser capaz de discutir estas tendencias con el director y razonar las decisiones que se tomen, en conjunto, y exponiendo sus puntos de vista con el objetivo de llegar a un acuerdo común.

Capítulo 3

Análisis

Análisis del problema

Como ya se ha aclarado varias veces a lo largo de este documento, actualmente hay una necesidad real en el mundo agrícola; la de disponer de la información sobre productos fitosanitarios de una manera centralizada, actualizada y de fácil acceso. Hoy en día esta necesidad se ha intentado abordar de varias maneras, y por ello hay disponibles varias aplicaciones que intentan apoyar al consumidor en las tareas agrícolas que involucran productos fitosanitarios. Algunos ejemplos son los productos que ofrecen empresas como aGROSLab, Cuaderno de Campo Agronev, o Agricolum. No obstante, estas aplicaciones se enfrentan al mismo problema; la inexistencia de una base de datos estandarizada cuya información sea congruente a través de los diferentes países de la Unión Europea. Por eso mismo, estas aplicaciones tienen que implementar, desarrollar y mantener ellas mismas las bases de datos que permitan acceder a la información deseada. Nuestra solución ofrecería un sistema dotado de una base de datos estandarizada, congruente en su información y completa, de código abierto y altamente escalable, por lo tanto todas las aplicaciones mencionadas arriba podrían convertirse en potenciales clientes consumidores de nuestro sistema, sustituyendo sus bases de datos por nuestra solución, con costes de integración mínimos.

Propuesta de valor e intermediarios

En este apartado se van a presentar los diferentes factores que van a intermediar en el proyecto: los proveedores de los datos, los recursos de los que se dispone, la propuesta de valor, los diferentes clientes del proyecto y los costes del mismo.

- **Proveedores:** Terceros de los que el proyecto se nutre para obtener los datos a emplear en la aplicación:

Cada fuente de datos proviene de algún portal web (nacional o internacional). En el proyecto se monta una infraestructura alrededor de dichos portales y por lo tanto la desaparición de alguno de estos sitios web significaría el cese de aprovisionamiento de los datos provenientes de dicha fuente. Actualmente, el proyecto se nutre de dos fuentes distintas: los productos autorizados recogidos del portal del *Mapama* [1] y los datos provenientes de la *base de datos de pesticidas a nivel Europeo* [7].

- **Recursos:** Elementos con los que el alumno ha contado para desarrollar el proyecto.

  - Equipo de trabajo (portátil propio)
  - Laboratorio de investigación - provisto por el director del proyecto, ubicado en el laboratorio L2.09 del edificio Ada Byron, en la Escuela de Ingeniería y Arquitectura de Zaragoza.
  - Herramientas open source - Todo el stack tecnológico, debido a un presupuesto nulo ha tenido que ser gratuito.
  - 300 horas contables de trabajo, las recomendadas para proyectos de este tipo.

- **Propuesta de valor:** Beneficios que aporta el proyecto respecto a la situación actual:

  - Diseño de una solución de integración que recoge datos sobre productos fitosanitarios de fuentes heterogéneas y los integra en un esquema único - que pretende solucionar algunos de los problemas actuales.
  - Desarrollo de una aplicación que muestre los resultados de manera gráfica - para demostrar los resultados tanto al tribunal como a los potenciales clientes.
  - Redacción de un documento explicativo del proceso anterior - la memoria, que servirá para que el tribunal comprenda la dedicación y el esfuerzo invertido en el proyecto y los avances y logros obtenidos.

- **Segmentos de clientes:** Los diferentes tipos de clientes objetivos a los que está dirigida la aplicación final:

  Por una parte están los agricultores que comercializan sus productos, las instituciones encargadas de validar y certificar la importación / exportación de productos agrícolas o las empresas importadoras / exportadoras de productos

agrícolas. Este segmento se beneficiaría diréctamente de los resultados del proyecto puesto que el proceso de comprobación de los requisitos fitosanitarios sobre los productos agrícolas / pesqueros / alimentarios se conseguiría de manera mucho más sencilla. Por otro lado también se podrían beneficiar de la infraestructura conseguida en este proyecto empresas que ya implementan sus propias aplicaciones fitosanitarias, pero carezcan de ese esquema y datos estandarizados que este proyecto propone.

Análisis de requisitos - Técnica *MoSCoW*

Por recomendación del director del proyecto, el análisis y la captura de requisitos han estado desde el principio regidos por el método *MoSCoW*, una técnica de priorización de requisitos usada en la gestión de proyectos, análisis de negocio y desarrollo de software con objetivo de llegar a un acuerdo común con los *stakeholders* (integrantes del proyecto) sobre la importancia que se debería dar a cada requisito. Esta técnica también es conocida bajo los nombres de *priorización MoSCoW* o *análisis MoSCoW*.

La metodología *MoSCoW* dicta cuatro categorías mediante las que se pueden priorizar los requisitos de un sistema o proyecto:

1. **Debe tener:** Son aquellos requisitos críticos para que el trabajo realizado durante un periodo de tiempo determinado (en nuestro caso desde ahora hasta junio 2017) sea considerado un éxito (*TFG* aprobado). Si uno de estos requisitos no se incluye, el proyecto debera ser considerado un fallo (no se puede presentar el *TFG*). *Debe tener* en *MoSCoW* se refiere a *MUST*, que puede ser considerado un acrónimo de *Minimum Usable SubseT*. En ese sentido se puede entender como la unión de los requisitos del producto mínimo viable con los requisitos legales (p.ej. documentación en forma de memoria de *TFG*, cumplimiento *LOPD*, etc.) y de seguridad (en el sentido de robustez y calidad de la solución) obligatorios o acordados. Los requisitos del proyecto acordados para esta categoría han sido los siguientes:

   - Recolectar datos oficiales tanto de productos fitosanitarios autorizados de España como de las sustancias activas a nivel europeo.
   - Almacenar la última versión de los datos en formato original y además mantener todas las versiones descargadas.
   - Monitorizar y almacenar los procesos de recolección de los datos de entrada así como las rutas de su procesado.

- Ofrecer la infraestructura y las herramientas de configuración necesarias para una expansión futura del proyecto.
- Implementar un modelo de aplicación consistente, ejemplificando el ciclo de vida típico de los datos desde su recogida hasta su presentación visual.
- Una memoria extensa y detallada.

2. **Debería tener:** Son aquellos que son importantes pero no necesarios para ser realizados durante el periodo de tiempo determinado. Pueden posponerse para ser realizados en el siguiente periodo. Son vitales pero no obligatorios, si no se implementan la solución es viable pero es doloroso no hacerlo. En el caso de nuestro proyecto, dentro de esta categoría se han identificado los siguientes requisitos:

   - Mecanismo de control de esfuerzos.
   - Mecanismo de control de versiones.
   - Módulo de trazabilidad de los datos desde las fuentes originales hasta su visualización.
   - Mecanismo de detección de errores e inconsistencias en los datos provenientes de diferentes fuentes.

3. **Podría tener:** Son aquellos que comparados con los anteriores son los menos deseados o tienen menor impacto. Hay que tenerlos controlados ya que sólo se podrán entregar si se dan las mejores condiciones (por ejemplo, el proyecto va más rápido de los esperado). Si hay algún riesgo en la entrega del proyecto estos requisitos serían los primeros en ser descartados. A continuación se pueden ver los requisitos del proyecto que se han ubicados dentro de esta categoría:

   - Genericidad en cuanto al soporte de integración de los datos de diferentes fuentes basado en un fichero de claves y valores.
   - Soporte para la búsqueda de registros (datos) desde la interfaz web.
   - Desarrollo dirigido por un paradigma de inversión de independencias para conseguir un control centralizado.

4. **No tendrá:** Son aquellos que no van a ser entregados durante el periodo considerado. Se mantienen en esta lista de alcance para clarificar el alcance de la solución. Esto evita que informalmente sean introducidos más tarde. El objetivo de esta categoría es ayudar a mantener el foco en una solución estricta.

- Rediseño y desarrollo en el lado del *Front-End*.
- Desarrollo de tests adicionales en el proyecto de *JHipster*
- Integración de más de dos fuentes de datos heterogéneas.

Análisis de riesgos

En la fase inicial del proyecto se abordó el proceso de gestión de riesgos, para determinar los diferentes factores que podrían afectar a un proyecto de esta envergadura. Dicho proceso consta de varios pasos que en conjunto permiten tener una visión clara de aquello que puede entorpecer, frenar o incluso imposibilitar la finalización del proyecto. A continuación se listan dichos pasos y se presentan las conclusiones principales extraídas de cada una de ellas, dejando para los anexos la versión completa:

1. En primer lugar, **la identificación de riesgos** permite determinar la lista de riesgos capaces de romper la planificación del proyecto. Durante esta fase se estudió qué factores podrían influenciar, en mayor o menor medida el flujo de trabajo normal del proyecto. Se agruparon en diferentes categorías para delimitar las zonas a las que afectaría cada riesgo. Así pues, aparecen 31 riesgos divididos en 4 clases:

   a) 4 riesgos globales (referentes a todo el proyecto)

   b) 5 riesgos tecnológicos (referentes a los aspectos más técnicos y tecnológicos del proyecto)

   c) 7 riesgos de alcance (referentes al tamaño y alcance de la solución)

   d) 14 riesgos de entorno de desarrollo (referentes tanto a la gestión como a las diferentes partes del entorno del desarrollador)

2. El **análisis del riesgo** ofrece una medición de la probabilidad y el impacto de cada riesgo. Maneja tres valores que determinan la gravedad de un riesgo: la probabilidad con la que se puede dar un riesgo, el impacto que tendría en el resultado final un riesgo y la aceptación del riesgo, una medida delimitadora que define aquellos riesgos que son considerados aceptables y aquellos ante los que se deben tomar medidas. En esta fase se detectaron un total de 6 riesgos reseñables, que se presentan en el siguiente punto.

3. La **priorización de riesgos**, fase donde se puede ver la lista de todos los riesgos anteriores ordenados por su gravedad. A continuación se mencionan aquellos que

han tenido un factor de gravedad superior a 4, límite del criterio de aceptación. Todos los riesgos que aparecen aquí han obtenido una puntuación de 6/6:

   a) RG_1. Riesgo global "Plazos".

   b) RT_2. Riesgo de tecnologías "Software no probado".

   c) RA_1. Riesgo de alcance "Tamaño estimado".

   d) RA_6. Riesgo de alcance "Número de cambios".

   e) RE_9. Riesgo de entorno de desarrollo "Formación".

4. Finalmente, la **planificación de la gestión de riesgos**, fase relativa al plan para tratar cada riesgo significativo. En este apartado la estrategia a seguir fue recoger los seis riesgos anteriores y proponer para cada uno una solución mitigadora. Los resultados de esta fase se pueden observar en el apartado *Análisis de riesgos* de los *Anexos*.

Análisis del contexto tecnológico

Dado que se trata de un escenario caracterizado por la heterogeneidad en contenidos y formatos de los datos, la necesidad de su procesamiento y de una escalabilidad futura, el proyecto está situado en un círculo de tecnologías Big Data, que presenta los siguientes retos:

- **Datos.** La información sobre los productos fitosanitarios no está habitualmente disponible en un formato estructurado. Es decir, la solución debe poder trabajar con datos en formatos no estructurados y ser capaz de procesarlos, idealmente convirtiéndolos a un formato relacional.

- **Esquema.** No existe un esquema de referencia compartido para integrar la información de productos fitosanitarios de diferentes países, pudiendo darse el caso de la existencia de varios esquemas distintos en función del caso de uso. Es decir, la solución debe poder reconfigurar el esquema de integración con facilidad.

- **Procesado.** Derivado del reto anterior, los datos no se pueden procesar y almacenar directamente en el esquema de integración sino que deben guardarse en el formato original y ser procesados bajo demanda teniendo en cuenta las características específicas de cada fuente.

- **Almacenamiento.** No se dispone de un presupuesto para invertir en un gran sistema de almacenamiento que permita almacenar los datos en formato original.

Por ello, toda solución deberá ser de código abierto y poder ejecutarse sobre hardware de bajo coste. Presentación de los datos. Es necesario desarrollar una interfaz visual para presentar los datos una vez integrados en un modelo único.

- **Agilidad.** La solución debe poder evolucionar con facilidad. Cualquier desarrollador que utilice la solución debe poder reconfigurar rápidamente el esquema común y los servicios que exponen dicho esquema para su uso.

- **Plazos.** Las limitaciones temporales y la priorización de tareas son influyentes en la elección del stack tecnológico, aunque en menor medida que los puntos anteriores.

Habiendo mencionado los factores anteriores, a continuación se presentan las tecnologías a las que se va a hacer referencia en el siguiente apartado con el objetivo de familiarizar al lector con las herramientas utilizadas:

- ***Apache Hadoop.*** [11] La librería software Apache Hadoop es un framework que permite el procesamiento distribuido de múltiples conjuntos de datos a través de clusters de ordenadores mediante modelos de programación sencillos. Está diseñado para escalar desde servidores únicos hasta miles de máquinas, donde cada una ofrece computación y almacenamiento local. Más que depender del hardware para prestar una alta disponibilidad, la librería en sí está diseñada para detectar y gestionar fallos en la capa de aplicación y así permitir una alta disponibilidad a pesar de que los equipos individuales fallen.

- ***Apache Hive.*** [12] El software de data warehouse Apache Hive facilita la lectura, escritura y gestión de grandes conjuntos de datos residentes en almacenes distribuidos de datos mediante SQL. La estructura de datos puede ser proyectada sobre los datos que ya existen en almacenamiento. Apache Hive provee una herramienta de línea de comandos y un driver JDBC para que los usuarios se puedan conectar a Hive.

- ***Apache Sqoop.*** [13] Apache Sqoop es una herramienta diseñada para transferir bloques de datos entre Apache Hadoop y almacenes de datos estructurados como las bases de datos relacionales.

- ***JHipster.*** [14] JHipster es una plataforma de desarrollo para generar, desarrollar y lanzar aplicaciones con tecnología Spring Boot, AngularJS y Spring microservices.

- ***Spring Framework.*** [15] El framework Spring provee un modelo de programación y configuración sencillo para aplicaciones Java. Un punto clave de Spring es el soporte infraestructural a nivel de aplicación.

- ***Spring Boot.*** [16] Spring Boot es un framework ligero que tiene la intención de simplificar el proceso de configuración de una aplicación hecha con Spring.

- ***AngularJS.*** [17] AngularJS es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

- ***Talend Big Data.*** [18] Talend es un software open-source de integración, procesado y transformación de datos. Permite trabajar con paradigmas Big Data y ofrece una interfaz gráfica para diseñar y programar cómodamente procesos ETL.

Elección del Stack Tecnológico

Debido a los factores mencionados en el apartado anterior, el *stack* tecnológico se vio restringido a las herramientas mencionadas anteriormente; tal como se ha dicho, la solución necesitaba almacenar la información en crudo hasta el momento de su procesamiento y es por ello que una aproximación relacional habría sido inviable. Por lo tanto, viendo las diferentes opciones *noSQL* disponibles (*MongoDB*, *Cassandra*, *DynamoDB*, *HBase*, etc) y por recomendación del director del proyecto, la elección más clara fue elegir *Hadoop* como sistema de almacenamiento. *Hadoop* es una gran herramienta para el escalado de aplicaciones y lo utilizan grandes empresas para *Big Data*. Se adapta perfectamente a las necesidades de este proyecto y el software es gratuito. Además, hay una gran comunidad de personas capaces de ayudar con cualquier proyecto y la documentación disponible es lo suficientemente extensa como para salir de cualquier situación problemática. Sin embargo, la instalación y configuración del sistema supuso un reto para el alumno debido a las diferentes fuentes de información disponibles *ad hoc*. Se necesitaron varios intentos hasta dejarlo completamente funcional y dar por finalizada su instalación en el equipo.

Hadoop por sí mismo no era capaz de solucionar todos los problemas del proyecto; este sirve para almacenar los datos y, si bien es cierto que las operaciones de *MapReduce* permiten transformar dichos datos, dada la necesidad del desarrollo ágil del proyecto, lo mejor fue disponer de algún mecanismo más sencillo para el procesado de los

mismos. Por ello, se decidió emplear *Hive* como herramienta de traducción de los datos almacenados en *Hadoop*, tanto en crudo como procesados, a una estructura relacional, sobre la que se podían hacer preguntas *SQL*. La propia instalación de *Hive* no supuso un gran problema, y, a pesar de que comparte nociones con *MySQL*, gran parte de su sintáxis es diferente. Por ello, el alumno tuvo que aprender a usar el lenguaje para poder trabajar con las estructuras de *Hive*.

El otro reto planteado fue la presentación de los datos integrados mediante una interfaz web, lo que vendría siendo el lado del cliente de la solución. Indudablemente hay casi infinidad de posibilidades para el desarrollo de esta parte. No obstante, realmente la parte de visualización no fue una parte crítica del proyecto, ni el objetivo del mismo. Es por ello que se adoptó una postura de desarrollo rápido del lado del cliente. *JHipster* (generador de aplicaciones sobre *Spring*) resultó la herramienta indicada, puesto que facilmente, en unos cuantos pasos era capaz de generar una aplicación sobre un esquema de base de datos (en este caso *MySQL*) con una interfaz gráfica cuidada que satisfacía las necesidades del proyecto.

Teniendo en cuenta que los anteriores puntos constituyen el core tecnológico del proyecto, conforme se avanzaba se veía que no eran suficientes para conseguir los resultados deseados. Se hicieron múltiples pruebas para intentar conectar *JHipster* diréctamente a *Hive* sustituyendo la base de datos de *JHipster* (*MySQL*) por *Hive*, aunque todas con resultados negativos que se recogen en detalle en la sección 5.3. Se llegó a la conclusión de la necesidad de emplear otras herramientas adicionales para diferentes tareas como el procesado de los datos previo a su exposición en *Hive*, o la traducción de los datos y estructuras de *Hive* a la base de datos que emplea *JHipster*. Las elegidas fueron *Talend Big Data* (herramienta de procesado de ficheros capaz de conectarse a *Hadoop*, extraer la información contenida en sus nodos, procesarla y volver a guardarla con los cambios efectuados, tal como se le indique) y *Sqoop*, una herramienta diseñada para transferir bloques de datos entre *Hadoop* o *Hive* y almacenes de datos estructurados como las bases de datos relacionales. No obstante, antes de dar con la solución de *Talend* se probó durante el período del verano el progama *Kettle* [19] y se intentó integrar de numerosas maneras con el proyecto de *JHipster*, todas resultando en conclusiones negativas.

Para poder conectar la base de datos de *Hadoop* con la base de datos *MySQL* de *JHipster*, el paso previo necesario fue conectar *Hadoop* con *Hive* y realizar la exportación de los datos de *Hive* a *MySQL* mediante *Sqoop*. La dificultad en este apartado fue no sólo aprender la sintáxis de *Sqoop* sino también ser capaz de preparar los datos en el lado de *Hive* para que encajen perféctamente todos los registros, filas y columnas con los correspondientes en la parte de *MySQL*, puesto que *Sqoop* falla a no

ser que ambas partes sean idénticas en cuanto a estructura.

Captura de requisitos final

Una vez realizado un análisis detallado tanto del problema como del marco conceptual y los riesgos en las fases previas al arranque del proyecto, lo siguiente que se determinaron fueron los requisitos del sistema. Gracias al análisis previo realizado mediante la técnica *MoSCoW* plasmado en la sección 3.3 y a las conclusiones obtenidas de los apartados anteriores, la tarea de captura de requisitos *funcionales* tanto del sistema como del proyecto resultó en una sencilla selección y categorización de los mismos. Además, como se puede observar, algunos de los requisitos priorizados en las categorias inferiores de *MoSCoW* en la sección 3.3 han sido descartados. Los requisitos no funcionales surgieron tanto de recomendaciones del director del proyecto como del anális previo. A continuación se recogen en tablas tanto los requisitos funcionales como los no funcionales, divididos en sistema y proyecto, siendo los de sistema los referentes al propio producto tecnológico en sí, y los de proyecto los referentes a la gestión del mismo:

Nombre	**Descripción**
RFS_1	El sistema deberá recolectar los datos oficiales tanto de productos fitosanitarios autorizados de España como de las sustancias activas a nivel europeo.
RFS_2	El sistema deberá almacenar la última versión de los datos recolectados en el RFS_1 en su formato original y además mantener todas las versiones descargadas de las mismas.
RFS_3	El sistema deberá monitorizar y almacenar los procesos de recolección de los datos de entrada, así como las rutas de su procesado.
RFS_4	El sistema deberá ofrecer la infraestructura y herramientas de configuración necesarias para que futuros desarrolladores puedan integrar otras fuentes de datos de manera rápida y eficiente.
RFS_5	El sistema deberá implementar un modelo de aplicación consistente, ejemplificando un ciclo de vida típico de los datos, desde su recogida, su procesamiento, su posterior integración en un modelo más completo y su presentación en un *Front-End* de ejemplo.
RFS_6	El sistema deberá implementar un mecanismo de detección de errores e inconsistencias en los datos provenientes de fuentes heterogéneas.

Tabla 3.1: Requisitos Funcionales del Sistema

Nombre	Descripción
RNFS_1	Los información de los productos fitosanitarios deberá ser recogida de portales como *Mapama* [1] y los datos sobre los pesticidas de la *Base de datos europea* [20].
RNFS_2	Se hará uso de algún tipo de *Crawler web* [21] para la descarga periódica de los datos sobre productos fitosanitarios y pesticidas.
RNFS_3	Como sistema de almacenamiento de los datos recogidos sobre productos fitosanitarios autorizados y sustancias activas se usará *Hadoop*.
RNFS_4	Para monitorizar, almacenar y mostrar los procesos de recolección de los datos de entrada se usará *Talend Big Data*.
RNFS_5	Para conseguir unos desarrollos posteriores más ágiles se hará uso de la herramienta *Hive*, que permite una aproximación relacional directamente sobre *Hadoop*.
RNFS_6	La presentación de los datos en su formato final se hará mediante una aplicación con *GUI*, desarrollada mediante la herramienta *JHipster*.

Tabla 3.2: Requisitos No Funcionales del Sistema

Nombre	Descripción
RFP_1	El proyecto deberá incluir una memoria en la que se documentan todos los pasos y procesos involucrados en su construcción.
RFP_2	Se deberá mantener constancia del esfuerzo dedicado durante el proyecto.
RFP_3	El proyecto deberá mantener un control de versiones actualizado en todo momento.

Tabla 3.3: Requisitos Funcionales del Proyecto

Capítulo 4

Diseño

Diseño conceptual

Con el análisis finalizado, el diseño de la solución prácticamente surgió por sí mismo; teniéndo en cuenta los requisitos, se llegó a la conclusión de que los desarrollos realizados a lo largo de la duración de este proyecto no serían más que el comienzo para algo mucho más grande; se debía ofrecer tanto la infraestructura necesaria como el soporte para que la solución pudiera escalar y ser expandida mediante el trabajo de futuros desarrolladores.

El diseño conceptual del sistema se presenta a continuación a una escala abstracta y de alto nivel. Básicamente, el proyecto hará uso de numerosas fuentes de información, en este caso sobre productos fitosanitarios y sus derivados. Dicha información deberá ser recogida de manera periódica y almacenada en algún sistema de persistencia tal como proviene de sus fuentes, es decir, en una versión *en crudo*. Una vez almacenada en su versión original, los datos pasarán a una siguiente fase en la que serán procesados bien para extraer la información relevante o de interés para el proyecto bien para añadir datos útiles como información de trazabilidad, todo esto con el objetivo en mente de conseguir ese esquema unificado. Una vez procesados, los datos se almacenarán en otro sistema de persistencia, o incluso en el mismo de antes, si es posible. En este punto habría dos opciones; o que los datos ya estuvieran integrados en un único modelo, o que estuvieran individualmente separados en el almacén anterior. En este segundo caso, los datos deberían ser unificados para conseguir ese esquema único y posteriormente poder ser visualizados, ya sea mediante un navegador con una aplicación web, o mediante otro método de visualización. En el diagrama de la figura 4.1 se puede observar el diseño conceptual descrito anteriormente de manera gráfica.

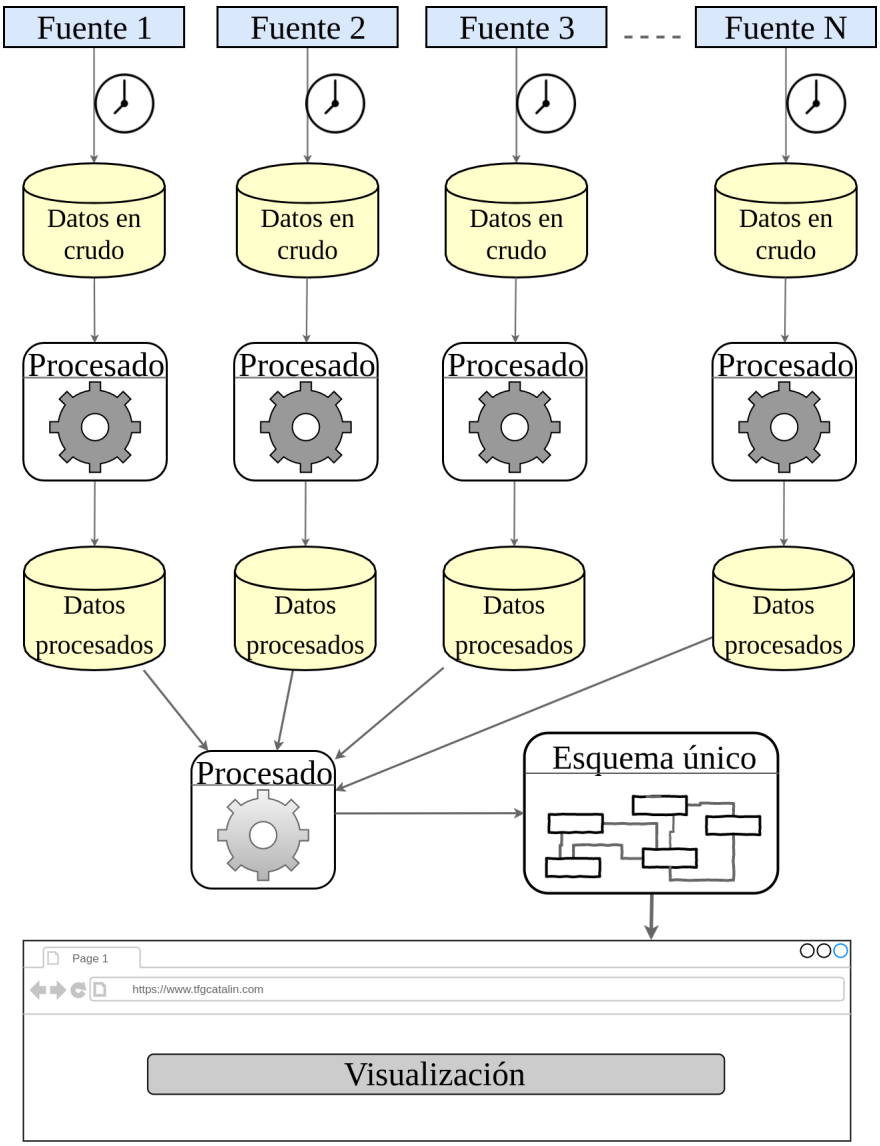


Figura 4.1: Diseño conceptual de la solución

Diseño final

En el apartado anterior se ofreció una visión de alto nivel del diseño de la solución, sin hablar de herramientas ni de tecnologías concretas. En este apartado se va a profundizar en el diseño y se va a explicar en detalle la manera en que los datos pasan a través de las diferentes herramientas dentro del sistema, desde la descarga de los datos desde sus fuentes hasta su visualización en pantalla, pasándo por las diferentes etapas de procesado y almacenamiento.

En el diagrama de la figura 4.2 se puede observar un mapeo casi dirécto entre los elementos del diseño final y los del diseño conceptual de la figura 4.1. Se puede observar que el diseño final incluye a *Talend* como responsable tanto de los procesos que descargan y almacenan los *datos en crudo* como de los que posteriormente procesan

y almacenan los datos como *datos procesados*. En cuanto al sistema de almacenamiento, que guardará tanto los datos originales como los modificados se utilizará *Hadoop* para los *datos en crudo* y *Hive* dentro de *Hadoop* para los *datos procesados*. Una vez los datos estén transformados y almacenados corréctamente, se usará *Sqoop* para transferirlos a ese esquema único que se persigue como objetivo, y que estará almacenado dentro de una base de datos *MySQL*. Para que este último paso sea posible, debe ser el propio *Talend* quien prepare los datos para ser integrados diréctamente en el esquema unificado. *JHipster* tendrá una conexión directa a la base de datos *MySQL* y gracias a ello será capaz de leer los datos y mostrarlos en pantalla con su propia interfaz web.

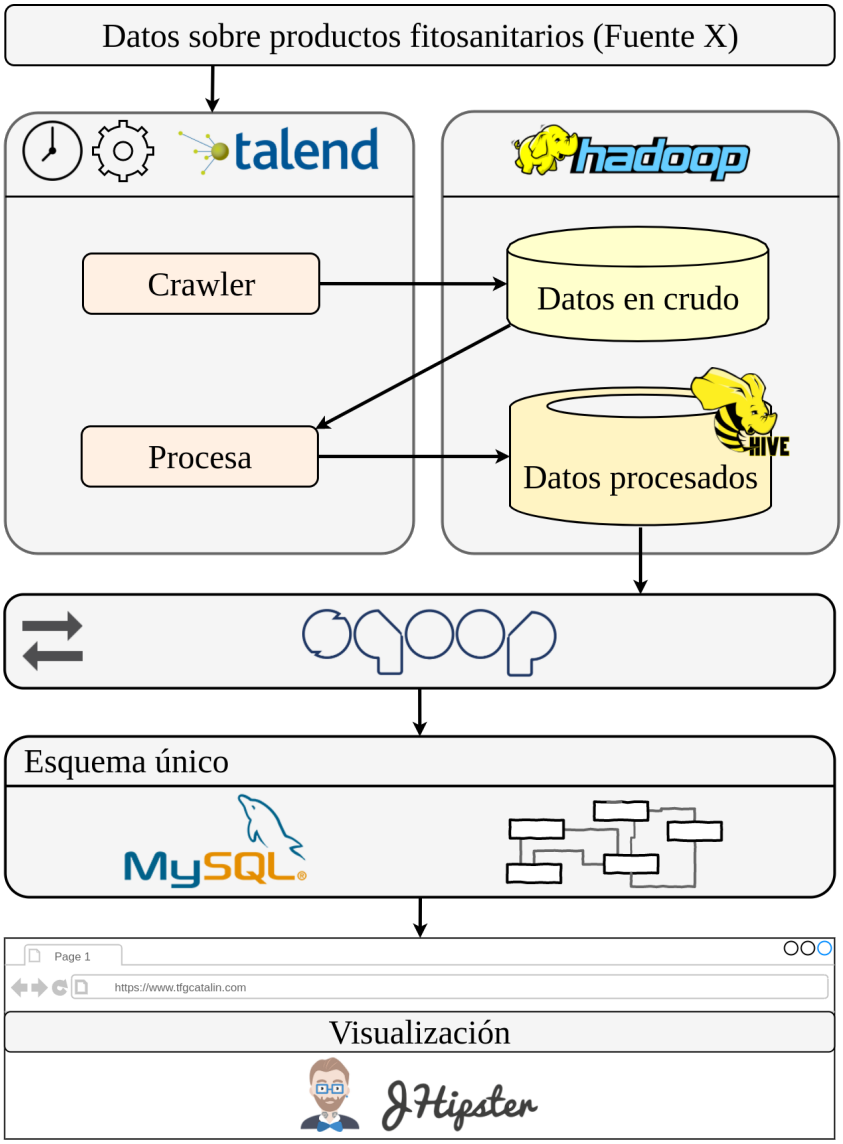


Figura 4.2: Diseño final de la solución

Arquitectura del sistema

Este apartado tratará de dar una visión arquitectural del sistema, pasando por diferentes diagramas para representar el proyecto de manera gráfica y completa. Esta sección abarcará los siguientes diagramas: diagrama de despliegue, diagrama de componente y conector, diagrama de clases y paquetes, diagrama de secuencia y diagrama de datos.

Diagrama de despliegue.

Como se puede observar, el despliegue de la aplicación consta de un nodo principal, el Servidor. Este almacena tanto la aplicación de *JHipster* como la base de datos *MySQL* y *Sqoop*. La aplicación de *JHipster*, al tratarse de una aplicación *Spring* se representa como un artefacto dentro de un contenedor *Spring*, el *Spring Container*. Además, allí se define el módulo de *schedulling* implementado por el alumno como un artefacto separado del de la aplicación, aunque a nivel práctico realmente su implementación está ubicada dentro del propio módulo de la aplicación. *Hadoop* aparece en el diagrama como una *base de datos* aparte, por varias razones: en primer lugar, *Hadoop* se puede desplegar en un nodo separado del Servidor. En segundo lugar, se pueden crear varias instancias en diferentes nodos del mismo para conseguir almacenar una cantidad mayor de datos, y proporcionar la aplicación de un componente de almacenamiento escalable.

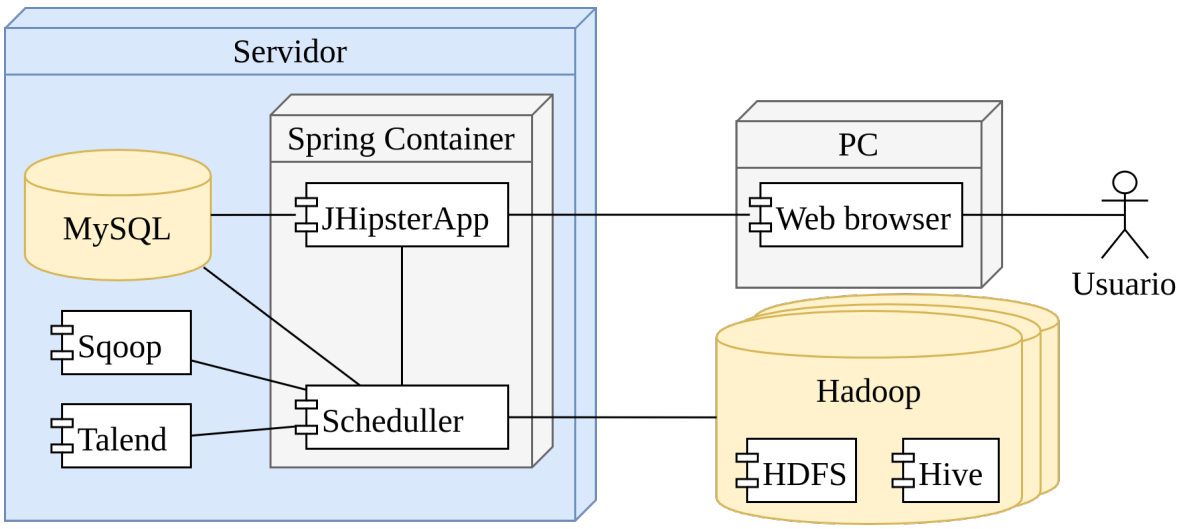


Figura 4.3: Diagrama de despliegue de la solución

Diagrama de componente y conector.

El próximo diagrama muestra la visión arquitectural del sistema a nivel de componentes y conectores. En él se representan todos los elementos del sistema junto con las interfaces que ofrecen y utilizan cada uno de ellos. Se observan todas las conexiones que hay entre los distintos componentes y se puede interpretar como una expansión, o una visión de más bajo nivel del diagrama de despliegue. Se entiende el componente *Talend* como los distintos procesos desarrollados con esta herramienta, más que una conexión con el propio programa ya que en ningún momento se requiere de *Talend* más allá que para la construcción de dichos procesos.

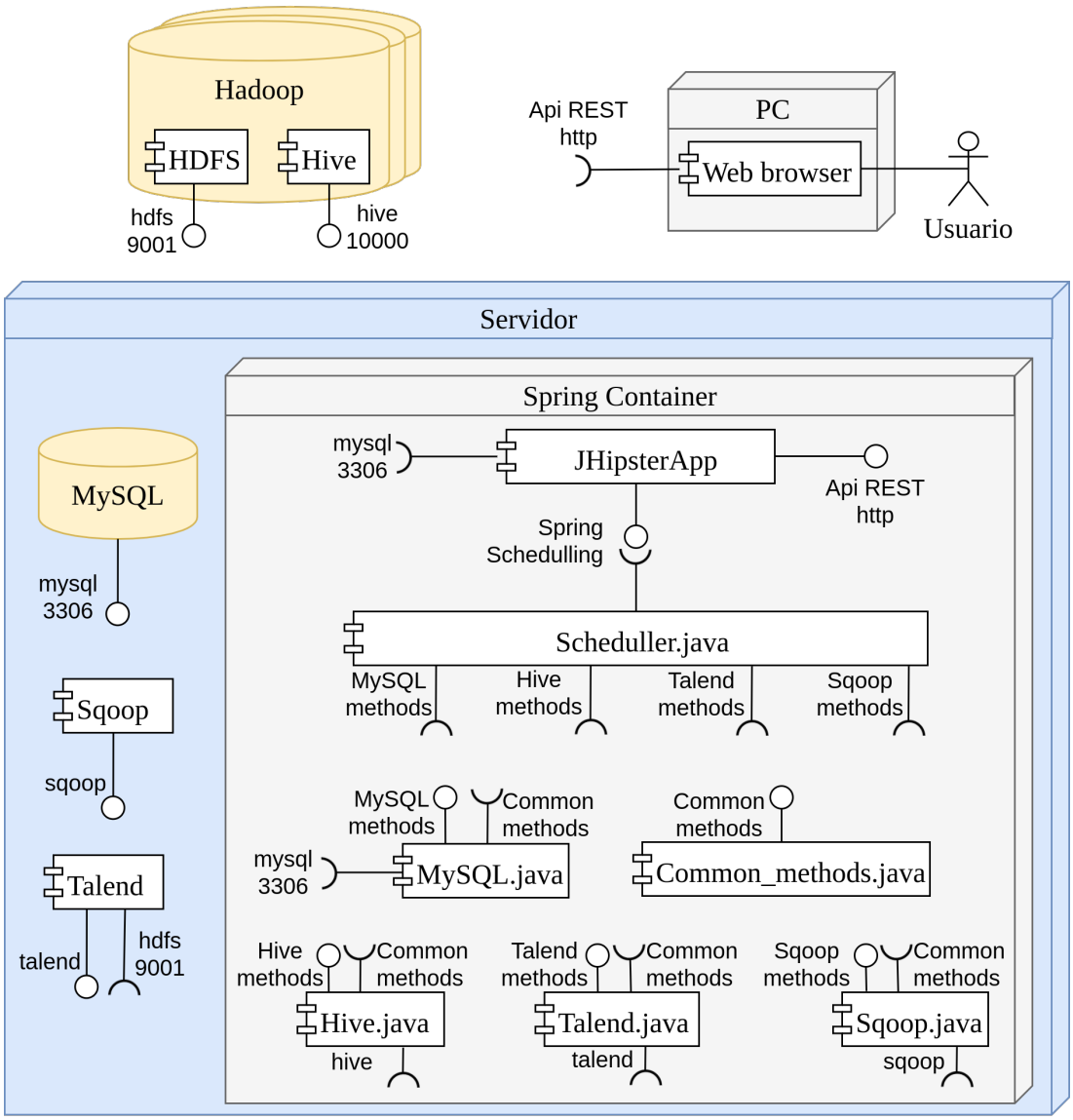


Figura 4.4: Diagrama de componente y conector

Diagrama de clases y paquetes.

En el diagrama de la figura 4.5 se adjunta el modelo de clases y paquetes correspondiente a la infraestructura que se construyó para soportar el comportamiento mencionado en la sección Implementación del prototipo real (Sección 5.2). Dicho diagrama no incluye aquellas partes del código que se generan durante la instanciación de la aplicación con *JHipster* sino únicamente las que el alumno ha desarrollado.

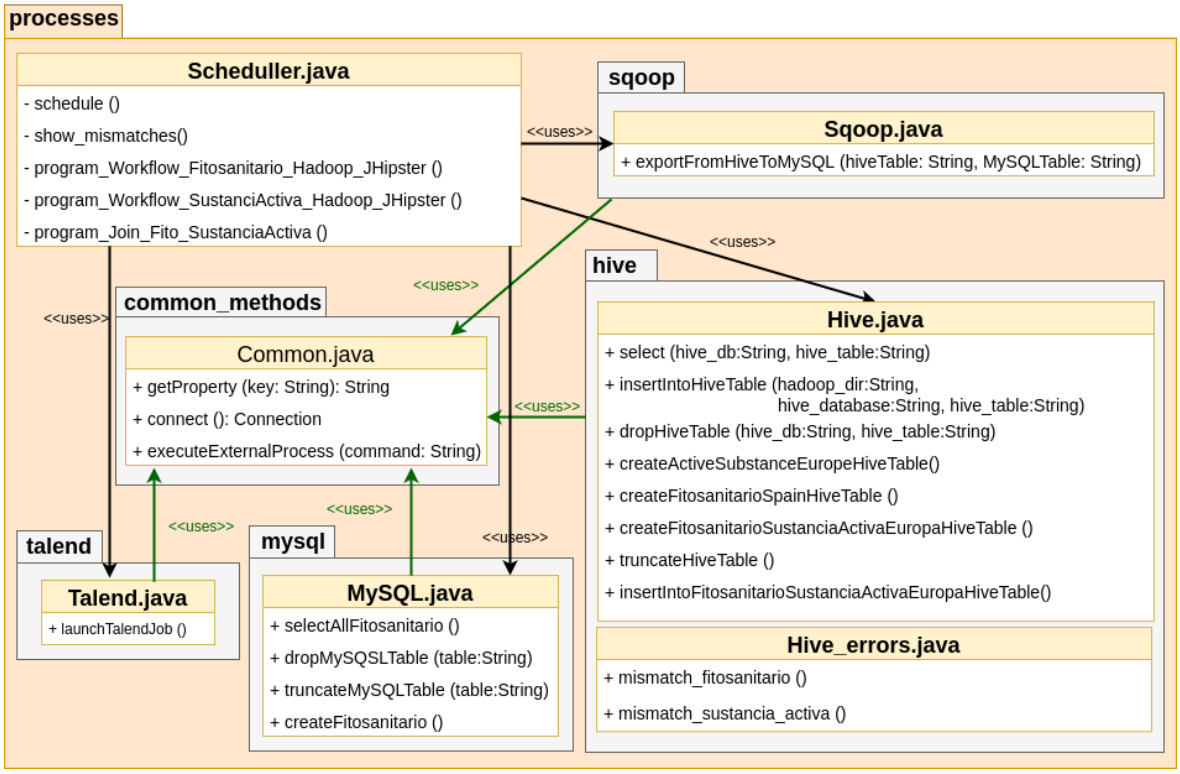


Figura 4.5: Diagrama de clases y paquetes para soportar la automatización del *workflow*

Diagrama de secuencia.

Una vez vista la estructura del diagrama anterior, a continuación se presenta un diagrama de secuencia de ejemplo para ilustrar la interacción de los diferentes componentes y el rol que juegan en el *workflow* desde que los datos se descargan hasta que pasan a visualizarse mediante *JHipster* Para ello se ha hecho uso de un ejemplo *vertical* para los datos de los productos fitosanitarios autorizados de España. Periódicamente, los datos se descargan desde la web del *Magrama* [1], son procesados y almacenados en *Hadoop*, se exponen en *Hive*, se transfieren con *Sqoop* a *MySQL* y *JHipster* es capaz de visualizarlos.

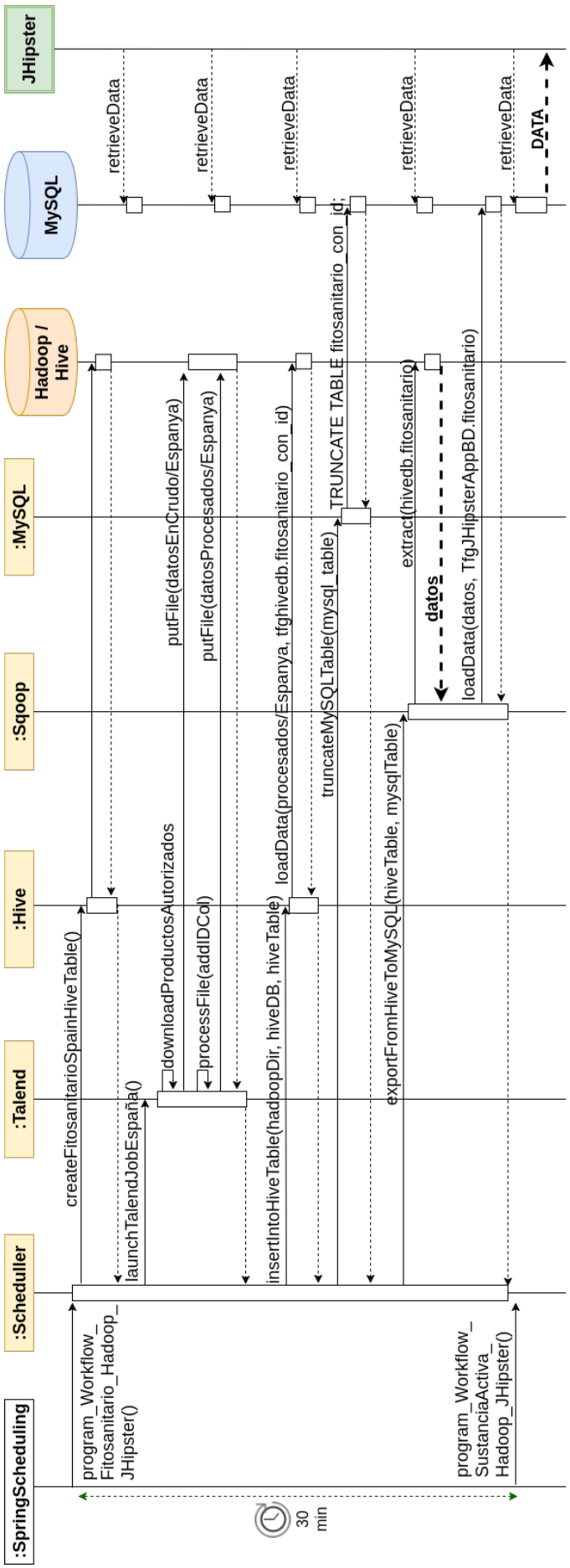


Figura 4.6: Diagrama de secuencia del *workflow* implementado

Diagrama de datos.

A continuación se muestra el diagrama de datos tal como están almacenados en *Hive*. A pesar de que también se almacenan datos en la base de datos *MySQL*, la estructura presente allí es relativamente sencilla en comparación con la de *Hive* y por lo tanto se ha decidido almacenarla en los anexos, en la sección A.1.

Como se puede observar, en la figura 4.7 aparecen, mediante una estructura de diagrama entidad-relación las entidades que han sido creadas en *Hive*. Por una parte tenemos las entidades *fitosanitario_con_id* y *sustancia_activa_europa*, que son las entidades principales de la solución. La primera se crea a partir de los datos obtenidos de la primera fuente integrada en el proyecto, proveniente del listado de productos fitosanitarios autorizados en España disponibles en la página web del *Magrama* [1]. La segunda son los datos de las sustancias activas provenientes de la base de datos abierta sobre pesticidas a nivel europeo. El resto de entidades presentes en el modelo son el resultado de diferentes operaciones sobre estas dos tablas. A continuación se explica cómo se han conseguido dichas entidades y qué representan.

1. En primer lugar, la relación marcada con un *1* en el diagrama representa una cierta similitud en el campo *formulado* de la primera tabla y el campo *name* de segunda: El *formulado* incluye el *name* de la segunda tabla. No obstante, los datos no son perfectos: por una parte está el problema del idioma; muchos de los fitosanitarios aparecen en español mientras que las sustancias activas están en inglés, por lo que un mapeo dirécto no daría el 100 % de los *matches*. Otro problema es el de los *campos múltiples*, esto es, en la primera tabla, algunos registros del campo *formulado* incluyen no solo uno, sino varios nombres de sustancias activas. Por lo tanto, haría falta una búsqueda para poder hacer el matching con todas las sustancias activas encontradas.

2. La relación marcada con un *2* representa la tabla *fito_active_substance*, que es el resultado de hacer un mapeo casi directo de la relación mencionada en el punto anterior: Primero, del campo *formulado* nos quedamos únicamente con el nombre de la sustancia activa. Después se hace una operación de *JOIN* para conseguir el *real_id* (id real de la sustancia activa) de la segunda tabla.

3. Las relaciones marcadas con un *3* y un *4* surgen de la necesidad de registrar los errores; las tablas *mismatch_fitosanitario_fito_sustancia* y *mismatch_sustancia_fito_sustancia* recogen aquellos fitosanitarios de la primera tabla que no aparecen en la tabla *integrada* (*fito_active_substance*) y aquellas sustancias activas de la segunda tabla que tampoco aparecen, respectívamente.

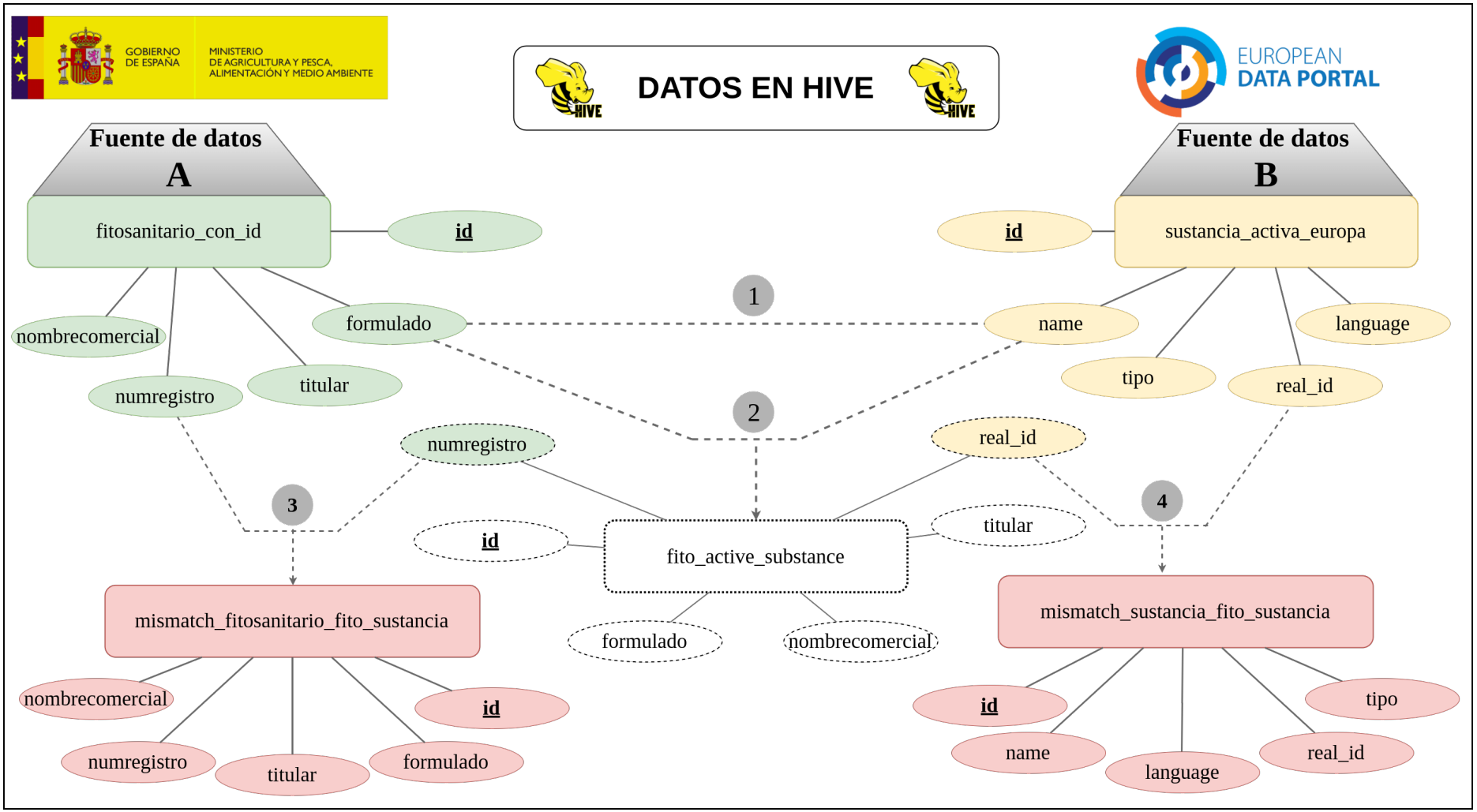


Figura 4.7: Esquema de datos importados en *Hive*

Estrategia de integración y expansión

Una vez visto el diseño del sistema, en esta sección se hablará de la estrategia de integración y expansión que gobernará los futuros desarrollos a partir de la infraestructura montada en la realización de este proyecto.

Como se observará en el desarrollo del prototipo real de la sección 5.2, realmente gracias a la infraestructura conseguida y a la arquitectura que se ha montado, realizar nuevas funcionalidades y expandir el proyecto no debería suponer un reto y debería ser bastante asequible. En este apartado caben destacar varias líneas posibles de expansión en el proyecto:

Aumento de la capacidad de almacenamiento. Este apartado es bastante trivial, puesto que gracias a *Hadoop*, esto se puede conseguir fácilmente. Tal como se ha comentado, *Hadoop* tiene el potencial de crecimiento mediante nodos adicionales, desplegados en la misma o en diferentes máquinas, con capacidad de almacenamiento extra. Así pues, para que el sistema escalase en cuanto a capacidad de almacenamiento, lo único que se tendría que hacer es desplegar más nodos de *Hadoop* para tener la información repartida en más espacio de almacenamiento.

Integración de datos nuevos. Para integrar nuevos datos en el sistema de nuevas fuentes, la estrategia a seguir debería ser la que se observa en el diagrama de la figura 4.8.

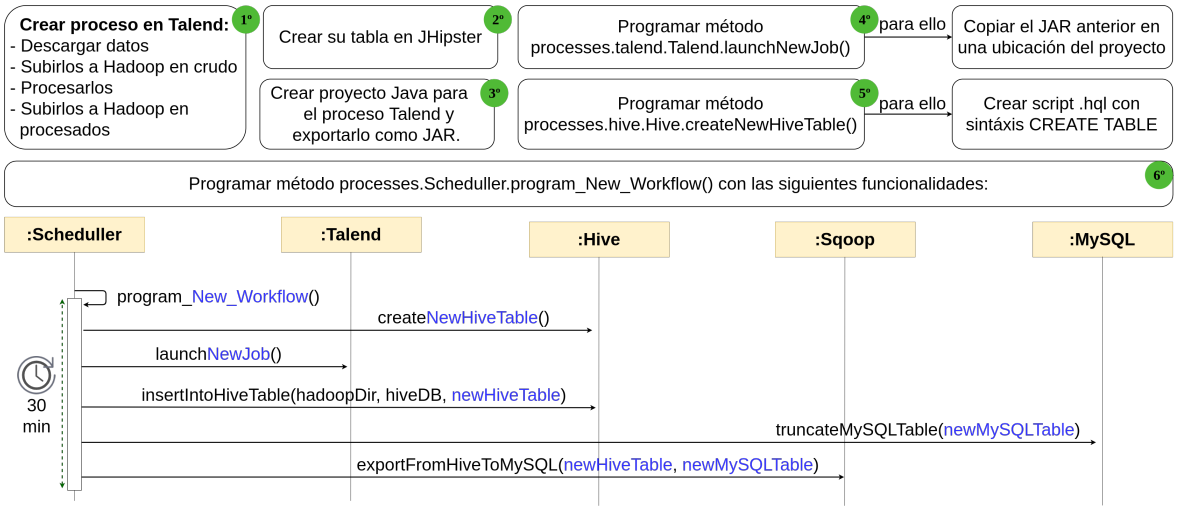


Figura 4.8: Diagrama de clases y paquetes para soportar la automatización del *workflow*

Funcionalidades extra. Aunque realmente este proyecto se ha desarrollado pensando en una expansión futura únicamente en cuanto a la integración de nuevos

datos y la esacalabilidad de almacenamiento, gracias a la estructura de la aplicación de *JHipster* es posible dotar al sistema de tantas funcionalidades extra como se desee.

Capítulo 5

Implementación

Prueba de concepto

La primera fase de desarrollo de la solución fue la prueba de concepto; su objetivo fue encontrar las herramientas adecuadas para el *Stack Tecnológico* y demostrar que las elegidas son viables y que funcionan en conjunto. Además, se estudiaron y valoraron los problemas que puedan presentar y los retos que podrían suponer desde una aproximación tecnológica. Conceptualmente, este apartado podría verse englobado dentro de la sección de Análisis (Capítulo 3) ya que, como se ha mencionado, la prueba de concepto fue la que determinó el *Stack Tecnológico* y, por ende, incluyó una correspondiente parte de análisis, esto es, búsqueda, investigación, test de viabilidad, etc. No obstante, dado que realmente formó parte del desarrollo de la solución se ha decidido redactarlo como una primera parte del apartado de Implementación de la solución. Así pues, esta sección presentará tanto el *Stack Tecnológico* utilizado, los problemas encontrados en esta fase y las diferentes alternativas que se han probado junto con los motivos por los que se han desechado de la decisión final. En el diagrama de la figura 5.1 se puede observar el panorama global de los pasos que se han dado y las herramientas que se han utilizado para montar toda la infraestructura necesaria para una versión final de la prueba de concepto dentro de la primera fase del desarrollo del proyecto.

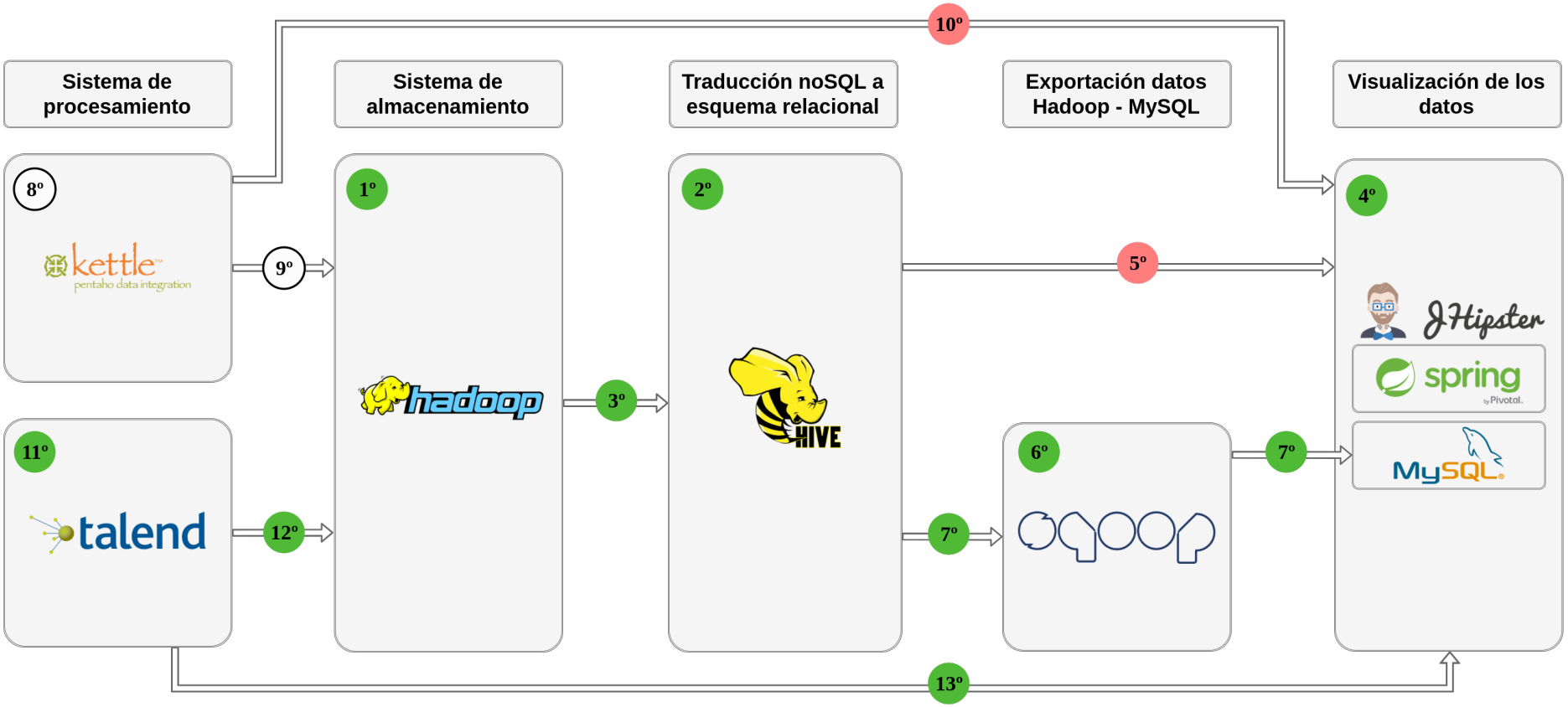


Figura 5.1: Diagrama de las etapas de la prueba de concepto.

Pasos reflejados en el diagrama de la figura 5.1:

- **1º.** Instalación y configuración de *Apache Hadoop*.
- **2º.** Instalación y configuración de *Apache Hive*.
- **3º.** Configuración de la conexión de *Apache Hadoop* con *Apache Hive*.
- **4º.** Instalación y configuración de *JHipster*.
- **5º Fallido.** Intento fallido de conexión directa entre *JHipster* y *Apache Hive*.
- **6º.** Instalación y configuración de *Apache Sqoop*.
- **7º.** Exportación de datos desde *Apache Hive* a *MySQL* (base de datos de *JHipster*) con *Apache Sqoop*.
- **8º.** Instalación y configuración de *Kettle Pentaho*.
- **9º.** Desarrollo de procesos *ETL* para *Apache Hadoop* mediante *Kettle Pentaho*.
- **10º Fallido.** Intento fallido de importación de procesos de *Kettle Pentaho* en *JHipster*.
- **11º.** Instalación y configuración de *Talend Big Data*.
- **12º.** Desarrollo de procesos *ETL* para *Apache Hadoop* mediante *Talend Big Data*.
- **13º.** Importación de procesos de *Talend Big Data* en *JHipster*.

A continuación se van a presentar detalladamente las herramientas usadas y las pruebas realizadas reparándo sólamente en aquellos pasos más conflictivos, y se van a exponer algunos de los retos enfrentados durante la prueba de concepto:

Hadoop. Inicialmente la instalación de *Hadoop* supuso algunos problemas puesto que el alumno no había tenido contacto con la herramienta previamente, y la *información* [22] que el alumno seleccionó como base para la instalación estaba desafortunadamente incorrecta (la instalación disponible en dicha web había sido probada con Ubuntu Linux 10.04, pero no con la versión del alumno, la 16.04). Por lo tanto, se tuvo que empezar de cero, eliminando cualquier rastro de la primera instalación de *Hadoop* del sistema. Después de ello, en un segundo intento, y gracias al *tutorial de instalación de Hadoop de Digital Ocean* [23] el programa funcionó correctamente y se procedió a instalar el siguiente bloque software necesario para el funcionamiento del sistema: *Hive*.

Hive. Para la instalación de *Hive* ocurrió un problema similar al de *Hadoop*. La fuente elegida para su instalación no fue la adecuada en un principio; el alumno eligió el tutorial expuesto en la web *Tutorial's Point* [24], que provee información no solo excesiva sino en ocasiones confusa. Como en el caso anterior, se tuvo que erradicar *Hive* del sistema para proceder con una instalación más limpia, esta vez desde la *página web oficial de Hive* [25], puesto que lo único requerido para su instalación fue su descarga y la declaración de las variables de entorno necesarias para su ejecución. De esta manera, se consiguió instalar la versión 2.2.1 de *Hive* sin dificultades.

JHipster: Instalación. Lo siguiente que se probó fue a instalar *JHipster*. Como *Java* y *Node.js* [26], dos de los componentes necesarios para su instalación ya estaban configurados en el equipo, lo único que se tuvo que configurar fue *Yarn*, que se hizo siguiendo los pasos recomendados para *Linux* en la *página oficial de Yarn* [27] para poder instalar *JHipster* con el comando *yarn global add generator-jhipster*, tal como indica la página oficial de *JHipster*. Esto en sí fue facil, y no supuso mayor problema; No obstante, el desconocimiento de *Yarn* junto con las actualizaciones periódicas que se introducían en *JHipster* hicieron que más de una vez se tuviese que borrar *JHipster* del equipo y volver a instalarlo en su última versión.

JHipster: Aplicación de prueba. Con *JHipster* instalado y configurado en el sistema, el siguiente paso obvio fue crear una aplicación y verla en funcionamiento. Para ello se siguió el tutorial del que se provee en la *página oficial* [14]. La creación de la aplicación con JHipster resultó bastante sencillo puesto que se trata de pasos secuenciales. No obstante, dado que en la web no existe mucha información acerca de la integración de un proyecto *JHipster* con *IntelliJ* [28] (entorno de desarrollo usado por el alumno), el arranque resultó bastante frustrante. El poco dominio que el alumno tenía tanto de *Gradle* como del propio entorno supuso un reto en las fases tempranas del proyecto, que se superó a base de leer documentación y realizar diversos intentos hasta que por fin se consiguió una versión de la aplicación corriendo en local, en el puerto 8080.

Integración Hadoop - Hive - JHipster. Si bien es cierto que la integración entre *Hive* y *Hadoop* resulta casi trivial, la integración entre *Hive* y *JHipster* es todo lo contrario. En primer lugar, se quería conectar *Hive* con *Hadoop* para disponer de una ayuda relacional para poder hacer consultas sobre datos en formatos no

relacionales. Su configuración se realizó modificando los ficheros de configuración dentro del directorio de instalación de *Hive*, para permitir una conexión entre los servicios de *Hive* y los nodos de *Hadoop*. En segundo lugar, la conexión entre *Hive* y *JHipster* se quería realizar para tener un flujo directo entre la información que se muestra en pantalla desde *JHipster* y los datos que son importados en *Hadoop* desde las diferentes fuentes. Para ello hay que tener en cuenta que cuando se crea una aplicación con *JHipster*, este pregunta por la base de datos que se quiera usar tanto en producción como en desarrollo. Actualmente *JHipster* ofrece soporte para las siguientes bases de datos: *MongoDB*, *Apache Cassandra*, o una base de datos SQL (*H2 Database Engine*, *Microsoft SQL Server*, *MariaDB*, *PostgreSQL*, *Microsoft SQL Server*, *Oracle Database*). Como se puede observar, *JHipster* no ofrece soporte oficial para una base de datos correspondiente ni a *Hive*, ni a *Hadoop*. Por eso, inicialmente la aplicación de *JHipster* se creó con una base de datos relacional *MySQL* puesto que su sintáxis es la más parecida a la de *Hive* (aunque no es igual). El objetivo en este caso fue conectar *JHipster* con *Hive* diréctamente, sustituyendo de alguna manera la base de datos *MySQL* y cambiando cualquier interacción que se tuviera con ella. No obstante, las tablas que se crean durante la creación de la aplicación se crean con una sintáxis propia de *MySQL*, en la base de datos *MySQL* y con un esquema a priori no visible. Tras muchas horas invertidas, muchos portales consultados, muchas preguntas en diversos foros de Internet, esta tarea se marcó como inalcanzable y se procedió a buscar otras soluciones con una viabilidad más alta.

Sqoop. Visto el resultado de la prueba anterior y por lo tanto el abandono de ese camino, el paso más lógico que se debía tomar a continuación era añadir un componente intermedio capaz de transferir datos de *Hadoop/Hive* a *MySQL*. Afortunadamente, ese componente existe y se trata de la herramienta *Sqoop*, que permite transferir datos de una tabla de *Hive* a otra tabla de una base de datos relacional, con un formato parecido o igual a la de *Hive*. Así pues, gracias a *Sqoop* conseguíamos disponer del mecanismo mediante el que los datos importados en *Hadoop* podían ser visualizados casi diréctamente (con su previa carga en *Hive*) en el *Front-End* provisto por *JHipster*.

Estado de la prueba de concepto. Con *Hive* conectado a *Hadoop*, *Sqoop* en marcha y una aplicación *JHipster* de prueba para ver un primer resultado de la integración de los datos, el sistema funcionaba acorde a las espectativas del *workflow* de la información: Almacen de los datos en *Hadoop* tanto en su versión "en crudo" como en una versión procesada, recuperación de los datos procesados desde *Hive*, transferencia de los mismos

a la base de datos *MySQL* mediante *Sqoop* y visualización por pantalla mediante la aplicación de *JHipster*. No obstante, analizando el estado de la prueba de concepto, se podía observar que en el anterior proceso faltaban tres cosas:

- **Automatización** de las tareas involucradas: Hasta este punto, cualquier parte del proceso requería de la intervención manual de un usuario, esto es, descarga de los datos desde sus fuentes, procesado de los mismos, carga de la información en *Hadoop*, creación de una tabla en *Hive* correspondiente a los datos de dicha fuente particular, carga de los datos desde *Hadoop* a *Hive* y la transferencia de los mismos mediante *Sqoop* hacia la base de datos de *JHipster*, *MySQL*. Viene siendo evidente la necesidad de un mecanismo que permita la automatización de todas estas tareas, con una intervención mínima por parte de un usuario. Esto permitiría aparte de un incremento considerable en el tiempo de resolución del *workflow*, un desarrollo futuro más ágil y sencillo.

- **Procesado de los datos** de manera eficaz: El *TFG* requería de un módulo de procesado de datos puesto que, como ya se ha explicado en ocasiones durante esta memoria, los datos pueden provenir de diferentes fuentes en formatos heterogéneos. Así pues, una carencia en este punto era esa herramienta o módulo que permitiera trabajar con datos en distintos formatos de una manera rapida y eficaz. Previamente los datos se habían "procesado" manualmente, con un sencillo editor de texto.

- **Actualización periódica** de la ejecución de todas las tareas: Teniendo en mente una visión futura y acabada del proyecto, otro aspecto que se echaba en falta en este punto era la posibilidad de que todo el *workflow* se ejecutase de manera periódica, obteniendo con esto una gran ventaja: la de proveer al consumidor de unos datos actualizados en todo momento.

Kettle. Dejándo de lado la *automatización de las tareas involucradas* y *la actualización periódica de la ejecución de todas las tareas*, lo siguiente que se abordó durante la prueba de concepto fue el problema del *procesado de los datos*. Para ello se requería del uso de alguna herramienta capaz de conectarse con *Hive* o con *Hadoop*, cuya especialidad sean las operaciones *ETL*. El director del proyecto impuso para esto la herramienta *Pentaho - Kettle* [19] dada su previa experiencia con este tipo de programas, sobretodo en el área *"GEO"*. Así pues, dentro del abanico de los diferentes productos de *Pentaho* [29] se encontraba *Pentaho Data Integration*, también conocida como *Kettle*, herramienta libre y gratuita con un diseñador gráfico para

realizar operaciones *ETL* que, según mencionaba, permitía una integración sencilla con diferentes tecnologías como *Hive* y *Hadoop*, e incluso ofrecía soporte para una integración con proyectos *Java*. Según las especificaciones del producto, parecía que encajaba perfectamente con las necesidades del proyecto. No obstante, resultó en un dolor de cabeza constante desde el momento de su instalación. No solo la interfaz del programa presentaba fallos, mezclando módulos en español con módulos en inglés o duplicando algunas funcionalidades, sino que además, el intento de migrar los procesos construidos en *Pentaho Data Integration* a la aplicación de *JHipster* resultaron en muchas horas de frustración, errores y hasta largas tutorías con el director del proyecto para intentar portar el código. Tras muchos días o incluso semanas de intentos y diversas formas de abordar el problema, lo que realmente acabó por apartar *Kettle* del *Stack Tecnológico* fue la adquisición de *Pentaho* por parte de *Hitachi* [30], privatizando el producto bajo el nombre de *Hitachi Vantara* [31] y ofreciéndo sólamente una versión de prueba del mismo.

Talend. Tras la privatización de *Kettle* y las tantas horas dedicadas a su integración dentro del proyecto de *JHipster*, se descartó *Pentaho* como parte del *Stack Tecnológico* y se empezó a buscar otras alternativas. La primera herramienta explorada fue *KNIME* [32] por recomendación de un compañero. Tras hacer algunas pruebas rápidas, se descubrió que realmente, aunque se ofertase como herramienta libre y gratuita, que lo era, algunas de sus funcionalidades eran de pago. La siguiente opción explorada fue *Talend* [18], que resultó ser la pieza clave para el funcionamiento del proyecto gracias a la sencillez de sus componentes, a la efectividad de su editor gráfico y gracias a una documentación extensa y bien organizada. A diferencia de las otras opciones para el procesado de los ficheros, con *Talend* se consiguió realizar una demostración de su funcionamiento mediante un sencillo proceso integrado en un proyecto *Java* nuevo, totalmente funcional. Ese proyecto posteriormente se empaquetó en un *.jar* y se exportó al proyecto de *JHipster* desde el que se pudo ejecutar con éxito, sin ningún problema de compatibilidad con el código ya existente.

Conclusiones. Una vez conseguidos los pilares fundamentales de la integración dentro del proyecto (*Hadoop*, *Hive*, *JHipster*, *Sqoop*, *Talend*) realmente las únicas preocupaciones que quedaban eran la automatización íntegra del proceso y la ejecución periódica del mismo para disponer de los datos en su versión actualizada. No obstante, para estas tareas no fue necesaria una prueba de concepto puesto que todo esto se podía conseguir desde el propio proyecto de *JHipster*, mediante *Spring* y código *Java*,

cosas con las que el alumno ya estaba familiarizado. Dándo por finalizada la prueba de concepto, se empezó a diseñar y construir el prototipo real que quedaría como solución real del proyecto.

Prototipo real

Primera iteración para conseguir una integración y automatización completa - Productos fitosanitarios autorizados de España

Lo primero que se hizo entrando en el desarrollo del prototipo real fue implementar un simple proceso mediante la interfaz gráfica de Talend. Este proceso realiza las siguientes operaciones:

1. Descarga desde la web del *Mapama* el fichero excel de los productos fitosanitarios autorizados.

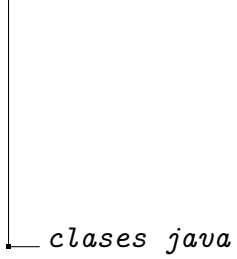
2. Convierte dicho excel a un formato openoffice para poder ser procesado desde Talend con los componentes excel correspondientes.

3. Sube a Hadoop una versión sin procesar del fichero

4. Procesa el fichero añadiendole una columna llamada ID al principio y lo sube como versión procesada a Hadoop.

A continuación se exportó el proceso desde Talend: *Archivo* → *Export* → *Java* → *JAR file*. Esto exporta las clases y librerias que Talend necesita para lanzar el job en un archivo comprimido llamado ¡nombre_job¿.jar El siguiente paso fue descomprimir el JAR en cuestión, analizar su contenido y ver cómo se podría importar en un proyecto Java. El JAR contenía varias carpetas y ficheros pero lo que interesa es lo siguiente:

```
Nombre_del_jar
├── lib
│   └── librerias jar
├── Nombre_del_proyecto
│   └── Nombre_del_job
│       └── clase java principal del job
└── routines
    └── system
        ├── api
        │   └── clases java
        ├── xml
        │   └── sax
        │       └── clases java
        └── clases java
```



Así pues, a continuación se creó un nuevo proyecto Java con IntelliJ y Maven (TalendCrawler) y se copiaron todas las clases Java con su correspondiente estructura de carpetas. Dentro del fichero pom.xml del proyecto TalendCrawler donde se importaron todas las dependencias de Talend que figuraban como librerías locales en la carpeta lib. Para ello se tuvo que definir el *repositorio de Cloudera* [33], que es desde donde Maven buscaría la mayoría de librerías. Tras comprobar que la aplicación arrancaba y se comportaba correctamente, el próximo paso fue encapsular y exportar la aplicación como un Jar, en conjunto con sus librerías. Para ello se hizo uso del plugin *one-jar* de Maven que recoge las dependencias del proyecto y las empaqueta junto a las otras clases en un único jar.

En el proyecto de JHipster lo que se hizo fue crear una clase llamada Talend, desde la que periódicamente (mediante @Scheduled) se ejecutaba el Jar anterior a través del comando Runnable.

Teniendo ya el proceso de *Talend* integrado en la aplicación de *JHipster*, el siguiente problema a abordar fue el de la automatización de su ejecución. Se sabe que los productos fitosanitarios autorizados son actualizados periódicamente en la web de *Mapama*. Por eso mismo, nuestra aplicación requería también de una descarga periódica de dichos datos, para asegurarse de que en todo momento el programa tiene la versión actualizada de los fitosanitarios autorizados de España. Esto se consiguió gracias al *módulo de scheduling*[34] de *Spring* que permite programar la ejecución de un método de manera periódica. Como decisión estratégica se propuso lanzar el proceso de *Talend* cada media hora. Resuelto este problema también, el siguiente objetivo fue automatizar toda la ejecución del proceso, desde la descarga del fichero de los productos autorizados hasta la visualización de los datos mediante *JHipster*. Aprovechandose del mismo módulo anterior de scheduling, el desarrollo tendría que seguir el siguiente esquema:

- Primero, los datos deberían descargarse y procesarse y almacenarse en *Hadoop* mediante el módulo de *Talend*.

- A continuación, se debería implementar otro módulo encargado de la carga de dichos datos procesados a una tabla de *Hive*.

- Después de eso, se deberían transferir los datos de *Hive* a la base de datos *MySQL* que emplea *JHipster*.

Así pues, para cada uno de los módulos mencionados se creó un paquete con una clase que contenía los métodos necesarios para lograr sus tareas particulares.

Segunda iteración para conseguir una integración y automatización completa - Sustancias activas de Europa

La primera iteración supuso los mayores problemas debido no solo al desconocimiento previo de las tecnologías sino también al hecho de no saber exactamente si dichas tecnologías iban a funcionar en conjunto. Una vez conocidas las tecnologías y tomado un primer contacto con ellas (el alumno no había trabajado con *Talend* previamente) la segunda parte de la integración se llevó a cabo de una manera mucho más fluída. Para esta iteración se conocía previamente el *modus operandi* para automatizar todo el proceso, desde la descarga de los datos hasta su visualización con *JHipster*. Por lo tanto, lo único diferente con respecto a la primera iteración fue desarrollar el trabajo de procesado específico de los datos de entrada.

Para la segunda iteración se eligieron los datos expuestos en la *Base de datos europea sobre pesticidas* [20] para seguir expandiendo la solución. Como ya se ha explicado, el objetivo de este proyecto es conseguir validar un modelo de integración para datos sobre productos fitosanitarios. En la primera iteración se obtuvieron los datos sobre los productos fitosanitarios autorizados en España. Estos contenían un campo llamado *Formulado*. Dicho campo se refiere a la *sustancia activa* de cada producto. Resulta que los datos descargados de la *base de datos europea* contienen una amplia estructura de datos e información relativa a los productos fitosanitarios. No obstante, dicha cantidad de información también resulta excesiva. Por ello, se ha optado por una aproximación minuciosa, cogiendo y procesando un solo elemento de todos los disponibles a la vez. En este caso dicho elemento corresponde a un fichero con la información relativa a las *sustancias activas*. Esta aproximación permitire ese objetivo de integración puesto que gracias a ello se puedo hacer un mapeo casi directo con los datos sobre productos autorizados de España.

De igual manera que en la primera iteración, se implementó en *Talend* el workflow necesario para procesar los datos de las sustancias activas. Esto es, por una parte, descargarlos de la página web, añadir la fecha y hora del momento de la descarga y guardarlos en *Hadoop* como datos en crudo de España sin alterar ni su formato ni su contenido. Por otra parte se formateó el contenido, para almacenar en *Hadoop* un fichero *.csv* con sólamente la información relevante del fichero original y con una columna extra para el identificador de las filas. El mismo proceso de Talend también se encarga de subir este *.csv* a Hadoop en la carpeta de datos procesados de Europa.

A continuación se preparó la infraestructura necesaria para soportar la carga de datos en *Hive* mediante una nueva tabla que se mantendrá actualizada con los datos más recientes sobre sustancias activas de Europa. Esto se consiguió gracias al desarrollo

implementado en el proyecto de *JHipster* desde el que periódicamente se lanza el workflow anterior de Talend, y posteriormente se realiza una importación de los datos a *Hive*. Además, en el lado del cliente, en *JHipster* se creó la tabla correspondiente a la d *Hive* en *MySQL* y, una vez más, periódicamente, los datos de *Hive* son transferidos a la base de datos *MySQL* a través de *Sqoop*. El resultado de esta iteración es que periódicamente, en *JHipster* se pueden visualizar los datos actualizados de las sustancias activas europeas sin necesidad de que el usuario tenga que intervenir o interactuar con el sistema en ningún momento.

Tercera iteración para conseguir una integración y automatización completa - unión de los datos anteriores en una nueva tabla - Fitosanitario_Sustancia_Activa_Europa

Mientras que las dos primeras iteraciones se centraron en recoger datos periódicamente de fuentes independientes, subirlas a *Hadoop* y luego importarlas en *Hive* y *MySQL* para ser consumidas por *JHipster*, la tercera iteración tuvo que ver con la integración de dichas fuentes independientes dentro del sistema. Como se ha mencionado anteriormente, los datos de las sustancias activas europeas se eligieron como fuente para este proyecto dado que encajaban en cierta medida con los datos de los productos fitosanitarios autorizados en España: Estos ultimos contienen un campo referente a las sustancias activas involucradas en el producto autorizado y gracias a eso se pudo hacer un *mapping* entre ellos. No obstante, el *mapping* no fue directo, puesto que los datos no venían en el mismo formato: en el caso de los productos autorizados, el campo en cuestión contenía además de los nombres de las sustancias activas en mayúscula la cantidad en la que podian estar presentes, mientras que en el caso de las sustancias activas europeas, los nombres venían en minúscula y sin la cantidad correspondiente. Así pues, en una primera aproximación lo que se hizo fue crear una tabla que contuviera los datos de los productos autorizados de España más una columna que fuera el identificador real de la **primera** sustancia activa involucrada en el producto.

Esta aproximación no es la solución perfecta, no obstante, es una primera iteración que soluciona una parte del problema. Se consiguió gracias a una consulta en *Hive* que partía los datos del campo "formulado"(referente a las sustancias activas que forman el producto) de los productos autorizados de España, se quedaba con la primera cadena de sólamente literales y hacía el *JOIN* con el nombre de la sustancia activa (pasado a mayúsculas) de la tabla de las sustancias activas europeas.

Como primera solución provisional, se consigue hacer un *matching* exitoso de unos cuatrocientos registros de un total de aproximadamente mil trescientas sustancias

activas. Los problemas que presenta son los siguientes:

- Hay productos autorizados que tienen mas de una sustancia activa como parte de su formulado y la consulta solo reconoce la primera de ellas.
- Hay sustancias activas que aparecen en los productos autorizados de España que vienen en español y la consulta no es capaz de reconocerlos puesto que las sustancias activas de europa tienen su nomenclatura en inglés.

Fichero de configuración

Para simplificar el acceso a los recursos se ha hecho uso de un fichero de configuración a los que acceden varios componentes: En primer lugar, el script bash que descarga los datos de los productos autorizados del *Mapama* [1]. Este Script usa una función bash para solicitar los valores del fichero de propiedades de la web del *Mapama*, y saber la ruta en el sistema donde guardar dicho fichero. Si en cualquier momento se quiere modificar dicha localización, gracias al fichero de configuración, el único sitio que se debería modificar sería en el propio fichero.

En segundo lugar, la aplicación Java del Job de Talend también accede a dicho fichero de configuración, puesto que en él se han establecido tanto rutas de almacenamiento dentro del HDFS de Hadoop, como el nombre del nodo o del usuario. No obstante, tal como se ha comentado en el apartado anterior, esta aplicación Java ha tenido que ser empaquetada en un Jar único y conjunto con todas sus librerías. Entonces ... ¿cómo accede a dicho fichero de configuración?. La solución ha sido hacer que el Jar reciba la ruta a dicho fichero mediante un argumento, de forma robusta, tal que si no recibe argumentos, o si el fichero que se le pasa no es un fichero de propiedades, el proceso alerta del error y se detiene.

Problemas encontrados

Tal como se ha explicado a lo largo del desarrollo de esta memoria, tanto en la fase de la prueba de concepto como en la fase de desarrollo del prototipo se han encontrado diversos problemas de naturaleza técnica o tecnológica:

- **Incertidumbre inicial.** El primero de los problemas que se detectaron tienen que ver con el arranque del proyecto. Debido a una incertidumbre inicial en cuanto a la estructura de su desarollo, en el arranque del proyecto no se pudo realizar una planificación inicial para dejar definida una visión global de toda su duración. Por ello surgieron problemas como que el alumno entendió que todo el trabajo de

instalación y configuración de las herramientas sería una fase previa al desarrollo del proyecto en sí, lo cuál no fue asi, puesto que más tarde se establecería que dicho trabajo formaría parte de una primera fase del proyecto: la prueba de concepto. Además, dicha incertidumbre dificultó la gestión del proyecto: la definición de las tareas, el control de los esfuerzos y un análisis adecuado desde el principio.

- **Sistema operativo.** Si bien es cierto que el sistema operativo donde se desarrollase el proyecto no era un requisito, apareció desde el principio como un derivado de las herramientas a utilizar. Se propuso un sistema operativo *Linux* [35] sobre el que llevar a cabo la implementación de la solución. La propia instalación del sistema resultó inicialmente problemática en el portátil del alumno debido a la inexistencia de los *drivers* de *Linux* necesarios para la tarjeta gráfica *Nvidia* [36] presente en el equipo, que resultaba en el no arranque del sistema. Tras unos días de consultas y búsquedas en páginas y foros de Internet, la solución al problema fue añadir la instrucción *nouveau.modeset=0*, que desactiva los drivers libres de *Nvidia* en el menú *GRUB* [37] durante el arranque del sistema permitiendo que la gráfica que se ejecuta sea la otra presente en el equipo, la de *Intel*.

- ***Hadoop.*** Como previamente se ha mencionado durante la prueba de concepto, la instalación de *Hadoop* no fue la óptima desde el principio puesto que el alumno instaló una versión correspondiente a *Ubuntu 14.04*, mientras que el sistema operativo instalado en el equipo era *Ubuntu 16.04*. Debido a eso inicialmente *Hadoop* dió problemas y en una fase posterior se tuvo que eliminar esta versión del equipo e instalar la correcta. Una vez solucionado ese problema, otro de los retos a los que se tuvo que enfrentar fue el entendimiento conceptual del sistema en sí. Se tuvieron que invertir horas en aprender a utilizar el sistema de ficheros *HDFS*, algo necesario para el almacén de los datos de entrada de las diferentes fuentes.

  Otro problema que surgió con *Hadoop* fue debido a la falta de espacio en disco. Llegó un punto a lo largo de la duración del *TFG* donde el disco duro del equipo del alumno se llenó y debido a eso las operaciones MapReduce de *Hadoop* se encolaban, se enmarcaban dentro de un estado de *Pendiente* y nunca se ejecutaban. Pasaron varios días hasta que se llegó a la raíz del problema y, una vez liberada algo de memoria del disco duro, las operaciones MapReduce de *Hadoop* ya se podían realizar corréctamente.

- ***Hive.*** Como ya se comentó, la instalación de *Hive* en sí no presentó problemas. No

obstante, el hecho de tratarse de un lenguaje *SQL* nuevo, a pesar de su similitud con la sintáxis de *MySQL* junto con las peculiaridades del sistema de archivos de *Hadoop*, *HDFS* sobre el que *Hive* trabaja dificultaron un avance fluido del proyecto. En muchas ocasiones los datos de entrada daban problemas a la hora de importarlos en *Hive*, por diferentes razones: inclusión de una cabecera que no debería aparecer en los datos de entrada, formateo incorrecto de los datos e incluso, por falta de experiencia, el uso incorrecto del delimitador en el lado de *Hive*. Aparte de estas dificultades, *Hive* también presentó problemas a la hora de intentar conectarlo diréctamente con *JHipster*, tal como se verá en los siguientes apartados.

- **Sqoop.** Al igual que *Hive*, la instalación y configuración de *Sqoop* no supuso un problema. No obstante, su manejo es lo que más dificultades presentó, puesto que al igual que los demás componentes, se trataba de una herramienta nueva para el alumno. Las tablas tanto de orígen (base de datos *Hive* presente en *HDFS*) como de destino (base de datos *MySQL* de *JHipster*) tenían que coincidir en estructura y tomó varios intentos hasta tener la configuración adecuada.

- ***JHipster.*** Con *JHipster* se tuvo varios problemas, empezando por su instalación en el equipo. *JHipster* utiliza *Yarn*, *Bower*, *Node.js*, *Gulp* y *Yeoman* y, ya que el equipo tenía preinstaladas algunas de estas herramientas en sus antiguas versiones, al principio la instalación resultó en fallos que tomaron tiempo para solventar. Otro problema, una vez solucionado el anterior fue a la hora de la importación de una aplicación generada con *JHipster* como proyecto dentro de *IntelliJ*. *Graddle* inicialmente no estaba corréctamente configurado en el equipo y por ello el proyecto era incapaz de descargar y configurar sus dependencias. Tras haber instalado *Graddle* e importado el proyecto corréctamente, otro de los problemas encontrados apareció a la hora de importar un esquema de datos sobre *JHipster* mediante *JDL-Studio* [38], el editor gráfico para creación de modelos de datos de *JHipster*. Si bien la creación del propio modelo y su descarga en un fichero *.jh* no presentó dificultades, la importación del esquema dentro del proyecto mediante el comando *jhipster import-jdl fichero.jh* provocaba errores en el proyecto en el momento de su ejecución. Gracias al control de versiones implementado mediante *GIT* [39] se pudo volver a la versión previa y desechar los cambios provocados por el comando anterior. Se descubrió que si las entidades se crean individualmente mediante el comando *jhipster entity nombreEntidad* el fallo anterior ya no ocurre y se puede continuar con una ejecución correcta.

  Otro problema con *JHipster* ocurrió al intentar actualizar la versión del mismo.

Según las instrucciones de la página web, el proceso debería ser aparentemente sencillo. Se trata de ir a la localización de la aplicación creada con *JHipster* y ejecutar el comando *jhipster upgrade*. No obstante, dicho comando en ocasiones funcionaba, y en otras, tras esperar el tiempo de actualización, todo aparentaba normalidad hasta que al arrancar la aplicación aparecían errores referentes a determinados *beans* de *Spring* que *JHipster* era incapaz de encontrar. A pesar de intentar solucionar dicho problema, en esos casos el alumno prefirió volver a la versión anterior del proyecto y desechar los cambios realizados por el comando *upgrade*. Como *JHipster* es un sistema en constante evolución, periódicamente los desarrolladores lanzan *parches* mediante los que solucionan problemas como el descrito anteriormente.

– ***Hive* y *JHipster*.** Tal como se explica en la prueba de concepto (sección 5.1), inicialmente la arquitectura del sistema debía reflejar una conexión dirécta entre la aplicación de *JHipster* y la base de datos *Hive*, mediante la sustitución de la base de datos de *JHipster* (*MySQL*) por *Hive* en los ficheros de configuración del proyecto. Teniendo en mente que *JHipster* diréctamente **no ofrece soporte** ni para *Hive* ni para *Hadoop*, la meta era *hardcodear* todos los parámetros y conexiones necesarias para *engañar* a la aplicación y conseguir que trabaje diréctamente con *Hive*. Esto consistía en importar las librerías necesarias, crear las entidades de *JHipster* dentro de *Hive*, indicarle el driver *JDBC* para conectarse con *Hive* y mapear cada acceso a la base de datos previa *MySQL* a la de *Hive*. El primer gran obstáculo que se detectó fue la importación de las librerías. Resulta que al importar las librerías necesarias para *Hive*, estas presentaban conflictos con las librerías que ya venían importadas por *JHipster*. Encontrar los paquetes conflictivos llevó mucho tiempo, puesto que el árbol de dependencias del proyecto tenía un tamaño considerable y no se podía saber a priori cuál de los múltiples duplicados existentes fallaba. Como se verá a continuación, el problema de las librerías externas dentro de *JHipster* seguirá apareciéndo con otros componentes, por lo que se puede entender que *JHipster* no es la mejor elección cuando se quiera expandir el proyecto con muchas librerías externas o de terceros. Por otro lado, el problema de las entidades se abordó poco a poco, intentando migrar las tablas una a una. No obstante, *JHipster* creaba dichas tablas con una sintáxis y unas propiedades y atributos acordes a *MySQL*, algunos de los cuales eran inviables de construir en *Hive* debido a la misma inexistencia de dicha funcionalidad. Las aplicaciones que *JHipster* generan tienen una gran complejidad, con cientas de clases interoperando y compartiendo información, dificultando la depuración

de la ejecución del programa y haciendo que una aproximación como la que se intentaba realizar en este período fuera inviable. Tras semanas de intentos frustrados, se consiguió que la aplicación arrancara conectándose a *Hive*, pero era incapaz de realizar cualquier función que se le pedía, como iniciar sesión con un usuario o registrar uno nuevo así que la arquitectura del sistema cambió, se desechó la idea de conectar diréctamente *Hive* con *JHipster* y se abordó una aproximación que involucrara un intermediario (*Apache Sqoop*) entre *Hive* y *JHipster* y manteniendo la base de datos *MySQL*.

- ***Pentaho Kettle.*** Aparte del problema anterior, el otro gran obstáculo que supuso un retraso del proyecto fue la elección de *Pentaho Kettle* como sistema para realizar operaciones *ETL* sobre los datos previo a su importación en *Hadoop*. Aparentemente *Pentaho Kettle* era la herramienta que se buscaba complementaria al *Stack Tecnológico* del proyecto. No obstante, con el tiempo se observó que más que una ayuda resultó un inconveniente. Desde su instalación, que resultó conflictiva puesto que por alguna razón al descargar el programa en español, este se descargó con la mitad de sus componentes en inglés y la otra mitad en español hasta su editor gráfico, que no es exáctamente muy *user-friendly*. Se tuvo problemas para conectarse a *Hadoop* ya que, por culpa de una documentación pobre no se da a entender que antes de intentar acceder a *Hadoop* mediante sus componentes se debía configurar un *Cluster* de *Hadoop* dentro del propio editor gráfico. Además se observó que el lanzamiento de un mismo proceso varias veces podía resultar en una ejecución exitosa o fallar estrepitósamente por razones aparentemente arbitrarias.

- ***Pentaho Kettle* y *JHipster*.** A la hora de integrar *Pentaho Kettle* y *JHipster*, al igual que ocurría con *Hive*, el problema principal fueron las librerías necesarias para poder trabajar con *Pentaho Kettle* dentro de la aplicación de *JHipster*. En este caso, además, reincidiéndo en el problema de la existencia de una documentación pobre, no se listan las librerías necesarias para poder trabajar con *Pentaho Kettle* desde un proyecto *Java*. Por esa misma razón, si bien es cierto que para trabajos o procesos sencillos (de prueba) de *Pentaho Kettle* se consiguió el pack de librerías necesarias a importar, en cuanto los procesos eran un poco más complejos (como los que realmente necesitaba el proyecto), las librerías empezaban a producir conflictos en el proyecto, resultando en la imposibilidad del arranque de la aplicación. El problema fue que conforme se solucionaban los errores, aparecían otros, y cada vez en más cantidad, haciéndo inviable la opción de *Pentaho Kettle* para los requisitos que tenía el proyecto. Por ello, tras reiterados

intentos de solucionar dichos problemas se optó por buscar otras alternativas y se encontró *Talend* como herramienta definitiva.

- ***Talend.*** El único problema que presentó *Talend* realmente no supuso un retraso tan considerable. *Talend* autogenera código *Java* conforme el usuario diseña sus procesos en el editor gráfico. Resulta que uno de sus componentes (*tHDFSInput*) en una primera instalación fallaba pues al insertar el código *Java* correspondiente, se dejaba por cerrar una llave, haciéndo que el programa diese error en tiempo de compilación y no se pudiese probar el proceso. No obstante, tras reinstalar *Talend*, este problema cesó y todos sus componentes funcionaron acordes a su especificación.

- ***Talend* y *JHipster.*** Reapareciéndo por tercera vez, el problema de las librerías importadas en *JHipster* volvió a surgir al intentar integrar el código generado por *Talend* de uno de sus procesos en el proyecto de *JHipster*. Habiéndo aprendido la lección de los anteriores intentos, esta vez no se reparó mucho en intentar solventar este problema sino que se adoptó una solución alternativa: crear un proyecto *Java* nuevo que integrase todas las dependencias necesarias para el proceso de *Talend* y encapsular todo su contenido en una librería ejecutable, fácilmente accesible desde *JHipster*. Esta solución también se probó con *Pentaho Kettle*, pero como ya se mencionó, debido a la falta de documentación acerca de las librerías a importar, se desistió en su resolución.

Capítulo 6

Gestión

Este capítulo engloba aquellos aspectos que tienen que ver diréctamente con la gestión del proyecto: la metodología que se ha seguido a lo largo de todo su desarrollo, la recogida de los esfuerzos y la organización del trabajo así como el control de versiones de los diferentes componentes del proyecto. Se reserva un apartado para mencionar las diferentes pautas e imposiciones que han afectado a un desarrollo libre del proyecto y se recoge un estudio acerca de la estimación del coste del proyecto en sí.

Metodología

Este apartado explica la metodología que se ha seguido para llevar a cabo el desarrollo del proyecto, desde su fase inicial llamada prueba de concepto (Sección 5.1) hasta su finalización. Se podrían distinguir dos fases conceptuales acordes a los diferentes *modus operandi* del desarrollo:

Prueba de concepto. El desarrollo de la prueba de concepto ha sido guiado por el director del proyecto. Esto es, el director del proyecto marcó inicialmente el panorama global y el diseño que se quería seguir. A partir de alli, el trabajo del alumno fue instalar y configurar las herramientas propuestas por el director y validar su integración mediante un flujo de datos de prueba. Si tras un período de pruebas exhaustivas algún componente fallaba o no cumplía con los requisitos especificados en el análisis del proyecto, dicho componente era desechado y se buscaba una alternativa viable al mismo. Lo mismo ocurría si alguna conexión de integración concreta fallaba; por ejemplo, en el caso de intentar conectar *Hive* con *JHipster* diréctamente, al comprobar que era una solución inviable que no cumplía con los requisitos del proyecto se decidió que la alternativa sería mantener la base de datos original de *JHipster* (*MySQL*) y agregar un componente intermedio (*Sqoop*) para la transferencia de los datos desde *Hive* a *MySQL*.

Prototipo real. Una vez integradas las diferentes herramientas y elegido el *stack tecnológico* final, la implementación del prototipo real se ha llevado a cabo mediante una metodología iterativa. Teniendo en mente que se trata de un proyecto con un potencial de crecimiento casi ilimitado, la aproximación más lógica fue agregar valor al proyecto mediante una aproximación de iteraciones agregativas, empezando por la integración de los productos fitosanitarios autorizados de España y siguiendo por los datos acerca sustancias activas de la base de datos Europea sobre pesticidas. Así pues, en la primera iteración se diseñó e implementó el flujo capaz de descargar los datos acerca de los productos fitosanitarios autorizados de España de la fuente, almacenarlos en *Hadoop*, transformarlos y prepararlos para su inserción en *Hive*, su transferencia a *MySQL* y su posterior visualización en *JHipster*. En una segunda iteración se realizó lo mismo pero con los datos de las sustancias activas extraídos de la base de datos Europea de pesticidas. Siguiendo esta metodología y con el objetivo fijado en el crecimiento del proyecto se puede observar que los futuros avances del sistema se pueden realizar de la misma manera. Otro desarrollador podría retomar el trabajo en este punto y hacer que el programa siga creciendo mediante la expansión del número de iteraciones que agreguen nuevos flujos de datos para dar sporte a nuevas fuentes. La tercera iteración supuso la agregación del soporte capaz de mapear los datos de la primera iteración con los de la segunda, en una versión más que nada ilustrativa; a pesar de que dicha integración no es capaz de mapear el 100 % de los datos, esto no es un problema peusto que no era el objetivo perseguido. Lo que se perseguía era validar el modelo de integración y dar soporte a un crecimiento sencillo de la solución. Por último, siguiendo esta filosofía de iteraciones se agregó en una cuarta iteración un mecanismo para la detección de errores o inconsistencias en los datos integrados provenientes de diferentes fuentes.

Pautas e imposiciones

La gestión y desarrollo del proyecto, en todas sus fases se ha visto restringida por diferentes pautas, imposiciones o recomendaciones provenientes de terceras partes. Este apartado pretende aclarar algunas de estas cuestiones para reflejar aquellas decisiones que han condicionado, para bien o para mal, el desenvolvimiento del alumno.

Desde el inicio del proyecto el director impuso algunas de las herramientas a utilizar, así como el diseño a priori de la solución. *Hadoop*, *Hive* y *JHipster* fueron el *core* tecnológico que el director estableció para la realización del proyecto. Como primer diseño, además, el director expuso un modelo en el que los datos tanto procesados como sin procesar serían almacenados en *Hadoop*, consumidos desde *Hive* e importados

diréctamente a *JHipster*, sustituyendo la base de datos de *JHipster* por *Hive*. Tras observar que este modelo no cumplía con los requisitos del proyecto, se optó por la otra variante, mediante *Apache Sqoop*, tal como se ha mencionado anteriormente.

Otra de las herramientas recomendadas por el director del proyecto fue *Pentaho Kettle* y, como se puede observar en la sección 5.3, fue una de las piezas que más problemas acabó dando. Ante esta situación, el director recomendó *Talend*, que resultó ser un mejor componente y que satisfacía con los requisitos de la fase de análisis.

Otro aspecto que se debe tener en cuenta es que el proyecto se trata de un *TFG* y no de una solución comercial. Por ello, hay unas normas o pautas establecidas que delimitan y guían en el desarrollo del mismo: la limitación de las horas de dedicación recomendadas, que se corresponden a los 12 créditos ECT, la inclusión de una memoria suficientemente extensa para recopilar todos los aspectos del desarrollo del proyecto e incluso la limitación económica implícita, esto es, no existe una remuneración monetaria para el alumno tras el desarrollo del proyecto.

Organización y control de versiones

Otro área de la gestión del proyecto es su organización, a través de sus diferentes componentes. En este apartado se pretende dar una visión global de las estructuras y tecnologías involucradas en la organización del proyecto.

En primer lugar, cabe mencionar que las diferentes herramientas que constituyen el *core* tecnológico del proyecto (*Hadoop*, *Hive*, *Sqoop*, *JHipster*, *Talend*, *MySQL*) se han instalado sobre el equipo del alumno, en una partición local del disco duro. Esto proporcionó rapidez de despliegue y desarrollo para el alumno pero podría suponer dificultades a la hora de expandir el proyecto e incluso riesgos adicionales debido a una inexistencia de tolerancia a fallos o copias de seguridad. No obstante, tratándose de un *TFG* se asumieron los riesgos y se adoptó esta postura como la más adecuada.

En segundo lugar, la propia gestión de las tareas a desarrollar durante el proyecto se ha controlado mediante *Trello* [40] a través de un *Kanban* de cuatro columnas (*To do*, *Doing*, *Problem* y *Done*):

- La columna *To do* almacena aquellas tareas que están pendientes de realizar o figuran como *features* posibles a desarrollar.

- La columna *Doing* contiene aquellas tareas que el alumno desarrollaba en cada momento.

- La columna *Problem* sirve para almacenar aquellas tareas que presentan algún problema y dificultan su terminación. A través del mecanismo de comentarios de

Trello el alumno dejaba redactado el problema que ha tenido en dicha tarea para tener constancia de ello en todo momento y posteriormente poder arreglarlo.

– La columna *Done* es donde se arrastraban todas las tareas que eran terminadas.

En la figura 6.1 se puede observar el tablero que el alumno ha usado a lo largo de casi toda la duración del proyecto.

Figura 6.1: Tablero *Trello* para la gestión de las tareas del proyecto.

Otro aspecto de la organización se centra en la aplicación desarrollada con *JHipster*. Inicialmente, esta se instaló al igual que las herramientas anteriores en el equipo local del alumno. No obstante, dado que sería una pieza fundamental y sobre la que se desarrollaría el software en sí, se decidió subirla a *GIT* para mantener un control de versiones sobre ella. Profundizando más acerca de la organización del software desarrollado, dentro de la aplicación de *JHipster* se creó un paquete encargado de mantener todo el código desarrollado por el alumno. Este paquete, llamado *processes*, junto con sus subpaquetes y clases se puede observar en la figura 4.5. Existe una clase principal llamada *Schedule* encargada de lanzar los diferentes procesos. Aparte, se han designado distintos subpaquetes en función de las herramientas contra las que atacan: el paquete *talend* es el que contiene los métodos encargados de ejecutar los trabajos desarrollados con *Talend*. El paquete *sqoop* contiene los métodos necesarios para poner en marcha una transferencia de *Sqoop* desde *Hive* a *MySQL*. El paquete *hive* contiene métodos que atacan contra la base de datos de *Hive* mientras que el paquete *mysql* contiene métodos que atacan contra la base de datos de *MySQL*. El

paquete *common_methods* es el único especial y contiene métodos públicos que puedan ser usados desde cualquiera de los demás paquetes.

Como esta memoria también se quería mantener bajo un control de versiones riguroso, también se decidió que debería formar parte del software subido a *GIT*. Así pues, en la carpeta raíz del proyecto de *JHipster* se creó una carpeta llamada *MEMORIA* donde se almacenaba todo lo referente a esta memoria.

Por último lugar, lo único restante de los diferentes componentes del proyecto son los diagramas desarrollados por el alumno tanto para el diseño de la aplicación como para los diferentes capítulos de la memoria y las hojas de gestión de esfuerzos. Estos componentes se crearon en *Google Drive* y se han ido actualizando allí mismo. Dado que el alumno usa la aplicación web *draw.io* propietaria de *Google Drive* para realizar los diagramas, esta aproximación se consideró como la más adecuada.

Control de esfuerzos

En la primera fase del proyecto el alumno desconocía el panorama global del desarrollo del *TFG*; desconocía el hecho de que habría dos fases, una en la que se realizaría una prueba de concepto y otra en la que se desarrollaría un prototipo real a partir de la validación de las herramientas empleadas en esa prueba de concepto; teniendo esto en mente, cabe destacar que el alumno consideró que el primer contacto con las herramientas, es decir, su instalación y configuración formarían parte de una fase previa, una especie de requisitos previos al arranque del proyecto, que no contabilizarían como esfuerzos en sí. Es por ello que al principio del proyecto el alumno no tomó nota de las horas precisas invertidas en aquella primera fase que más tarde se le revelaría que formaría parte de la prueba de concepto. No obstante, gracias a las herramientas como *Drive*, *Trello* o *GIT*, posteriormente se pudo hacer una recopilación aproximada de los esfuerzos invertidos durante esta fase. Así pues, una vez que se determinó la estructura final del proyecto se empezó a tener constancia de las horas a través de una hoja de cálculo almacenada en *Drive* y gracias a ello se pueden presentar los esfuerzos divididos en las siguientes categorías:

- **Investigación.** El apartado de investigación incluye los diferentes esfuerzos realizados por el alumno para comprender la problemática actual que se intenta resolver en este *TFG*, desde lecturas de manuales fitosanitarios hasta portales web que explican los procesos actuales de importación y exportación de los mismos. Se han dedicado aproximadamente 10 horas a este bloque.

- **Análisis.** La fase de análisis comprendió un máximo de 20 horas aproximadas,

entre la determinación de los requisitos, el análisis de los riesgos y la propia elección del *Stack Tecnológico*. Aunque este último va diréctamente asociado a la prueba de concepto, cabe mencionarlo durante esta fase puesto que es donde a priori se analizaban las diferentes herramientas posibles de todo el elenco disponible.

- **Diseño.** El diseño del sistema tomó un máximo de 10 horas entre las diferentes variantes conceptuales; conforme se demostraba que un diseño aparente no cumplía con los requisitos del sistema, se procedía a diseñar otro, mejor adaptado a las necesidades del proyecto.

- **Prueba de concepto.** Al igual que las fases anteriores, gran parte de esta se ha desarrollado cuando aún no se tenía constancia precisa de las horas invertidas. No obstante, se pueden deducir alrededor de 80 horas totales que incluyen la instalación y configuración de las diferentes herramientas probadas junto con sus alternativas, las diferentes pruebas realizadas para conseguir un flujo de los datos desde su descarga hasta su transformación y presentación y la solventación de los diferentes errores que iban apareciéndo por el camino.

- **Implementación del prototipo real.** La implementación del prototipo real reune todos sus esfuerzos en la hoja de cálculo de *Drive*, con un total de 63.5 **CAMBIAR SI NECESARIO** horas de dedicación. En ellas están incluidos los diferentes procesos desarrollados en *Talend*, los diferentes mecanismos para su integración en *Java*, los *crawlers* implementados en lenguaje *bash* así como los componentes software desarrollados en *Java*.

- **Reuniones con el director del proyecto.** Se pueden deducir unas 18 horas de reuniones con el director del proyecto aunque este apartado se puede entender como algo más flexible que los anteriores, ya que, si bien es cierto que no han habido muchas reuniones planificadas con el profesor, este iba pasando por el laboratorio en el que el alumno desarrollaba el trabajo para revisar con él los avances conseguidos y apoyarle en la consecución de los objetivos.

- **Redacción de la memoria.** Las horas invertidas en la redacción de la memoria, al igual que en el bloque anterior, se recogen en su totalidad en la hoja de cálculo de *Drive*. Han resultado un total de 93.5 **CAMBIAR** horas.

Estimación del coste

En este apartado se van a recoger los cálculos que se han llevado a cabo para calcular tanto el coste económico del *TFG* como de la hora de trabajo de una persona que desarrollaría de manera comercial un proyecto como este. Para ello se ha hecho uso de la página web *Calculadora Freelance* [41] y los resultados se pueden ver en los párrafos siguientes.

Para calcular el coste económico del proyecto, se deben conocer a priori tanto las horas totales invertidas en el proyecto como el precio de la hora de trabajo del alumno; el resultado de la multiplicación de estos factores, sumado a otros elementos (como la gestión del proyecto, la gestión de configuraciones o el aseguramiento de la calidad), se corresponde al coste total del proyecto. Las horas invertidas en el proyecto han sido recogidas en el apartado anterior, mientras que la hora de trabajo del alumno se ha calculado a partir de los siguientes parámetros:

- **Sueldo esperado.** 1500€ mensuales.
- **Días de vacaciones.** 21 días anuales.
- **Días festivos.** 14 días anuales en España.
- **Días de inactividad.** 7 días anuales.
- **Porcentaje reuniones, presupuestos, ventas, etc.** 50 %.
- **Gastos alquiler.** 100€ mensuales.
- **Gastos en servicios (luz, móvil, etc.).** 50€ mensuales.
- **Impuesto autónomos.** 260€ mensuales.
- **Otros gastos.** 50€ mensuales.
- **Porcentaje beneficios.** 20 %.

Con los datos anteriores, el coste mínimo de la hora de trabajo del alumno sería de **28.70€**.

El desglose detallado de los cálculos realizados para llegar al resultado anterior se pueden observar en la sección D.1 de los Anexos.

A continuación, en la tabla 6.1 se recogen los diferentes componentes y tareas realizadas por el alumno junto con las horas dedicadas y el coste calculado, para conseguir el coste total del proyecto:

Tarea/Componente	**Horas**	**Coste (€)**
Investigación	10 horas	287.00€
Análisis	20 horas	574.00€
Diseño	10 horas	287.00€
Prueba de concepto	80 horas	2,296.00€
Implementación prototipo	63.5 horas	1,822.45€
Reuniones	18 horas	516.60€
Redacción Memoria	93.5 horas	2,683.45€
TOTAL Tareas/Componentes	**295 horas**	**8,466.50€**
Gestión (G)	295 h x 0.15	1,269.97€
Gestión de configuraciones (GC)	295 h x 0.05	423.32€
Aseguramiento de la calidad (AC)	295 h x 0.07	592.65€
TOTAL GESTIÓN Y CALIDAD	**80 horas**	**2,296.00€**
Transporte (T)	60 viajes x 2.70€	162.00€
TOTAL MACROS	$G+GC+AC+T$	**2,458.00€**
Amortización estaciones de trabajo	(295h + 80h) x $\frac{800}{295h}$	1,016.95€
TOTAL		**11,941.45€**

Tabla 6.1: Costes económicos del proyecto

Capítulo 7

Conclusiones

En este capítulo se hablará de las conclusiones sacadas tras finalizar el proyecto. ¿Se han conseguido los objetivos propuestos? ¿Están todos los requisitos cubiertos? ¿Está bien documentada la solución? ¿Es escalable? Se intentará responder a estas preguntas de la manera más concisa y sincera posible.

Resultados y objetivos

(**HAY QUE HACER EL DE LA TRAZABILIDAD!!!**)

Haciendo una retrospectiva global de los avances conseguidos en este proyecto se puede llegar a la conclusión de que se han conseguido todos los objetivos propuestos en el inicio del mismo. A continuación se van a exponer los requisitos funcionales que han sido establecidos durante la fase de análisis (recogidos en la sección 3.7) y se va a razonar el porqué de la consecución y finalización de cada uno de ellos.

- **RFS_1.** *El sistema deberá recolectar los datos oficiales tanto de productos fitosanitarios autorizados de España como de las sustancias activas a nivel europeo.* - Este requisito está cumplido puesto que, como se puede observar tanto en el capítulo del Diseño (4) como en el de la Implementación (5), tanto los datos sobre productos fitosanitarios autorizados de España como las sustancias activas a nivel europeo forman parte de las fuentes que se han integrado en el sistema.

- **RFS_2.** *El sistema deberá almacenar la última versión de los datos recolectados en el RFS_1 en su formato original y además mantener todas las versiones descargadas de las mismas.* - Este requisito también se cumple puesto que los datos originales descargados periódicamente se almacenan en *Hadoop* en una carpeta llamada *Datos_en_crudo* con la fecha y hora exacta de su descarga y además dichos datos nunca son borrados.

- **RFS_3.** *El sistema deberá monitorizar y almacenar los procesos de recolección*

de los datos de entrada, así como las rutas de su procesado. - Otro requisito cumplido, puesto que la aplicación de *JHipster* es la que se encarga tanto de almacenar los procesos de *Talend* en su formato *JAR* como de programar su ejecución de manera periódica y asegurarse de un funcionamiento correcto del mismo.

- **RFS_4.** *El sistema deberá ofrecer la infraestructura y herramientas de configuración necesarias para que futuros desarrolladores puedan integrar otras fuentes de datos de manera rápida y eficiente.* - Como se puede ver en la sección 4.4 del capítulo del Diseño, se ha creado una infraestructura capaz de ofrecer un mecanismo sencillo para futuros desarrollos. Tanto integrar nuevos datos como expandir el proyecto en otros ámbitos funcionales no debería ser un problema para los siguientes programadores que retomen este trabajo; por lo tanto se puede dar por cumplido este requisito.

- **RFS_5.** *El sistema deberá implementar un modelo de aplicación consistente, ejemplificando un ciclo de vida típico de los datos, desde su recogida, su procesamiento, su posterior integración en un modelo más completo y su presentación en un Front-End de ejemplo.* - El requisito RFS_5 también se puede considerar como conseguido dado que, tanto para los datos sobre productos fitosanitarios autorizados de España como para las sustancias activas a nivel europeo dicho ciclo de vida típico ha sido implementado; los datos se descargan, se procesan, se almacenan en *Hadoop*, se transfieren de *Hive* a *MySQL* y se visualizan con *JHipster*.

- **RFS_6.** *El sistema deberá implementar un mecanismo de detección de errores e inconsistencias en los datos provenientes de fuentes heterogéneas.* - Como se ha observado en el capítulo de la Implementación del prototipo real (5.2), se ha implementado un módulo que se encarga de mostrar los datos inconsistentes resultantes de la integración de las dos fuentes principales de este proyecto. Así pues, este requisito también se da por válido.

En cuanto a los requisitos funcionales del proyecto como desarrollo global, aparecían los siguientes en la captura de requisitos:

- **RFP_1.** *El proyecto deberá incluir una memoria en la que se documentan todos los pasos y procesos involucrados en su construcción.* - Requisito validado puesto que esta es dicha memoria.

- **RFP_2.** *Se deberá mantener constancia del esfuerzo dedicado durante el*

proyecto. - Requisito cumplido puesto que los esfuerzos se han ido manteniendo mediante la hoja de cálculo de *Drive*.

– **RFP_3.** *El proyecto deberá mantener un control de versiones actualizado en todo momento.* - Requisito cumplido puesto que el proyecto se ha subido a *GitHub* y a través de su sistema de commits se ha mantenido un riguroso control de versiones.

A pesar de haber comprobado que todos y cada uno de los requisitos funcionales han sido cumplidos al 100 %, hay que mencionar y tener en cuenta las siguientes consideraciones:

– El módulo de gestión de errores está en una versión primitiva. No se ha profundizado en su desarrollo, y al momento de la finalización de este *TFG* únicamente se recogen aquellos registros que no han podidos ser integrados, sin hacer ningún análisis posterior. A efectos de este *TFG* no es un problema puesto que no era un aspecto que se perseguía.

– El módulo de la integración de varias fuentes en un esquema único también aparece dentro de un desarrollo primitivo; A pesar de una transformación previa de los datos de ambas tablas con el objetivo de conseguir un *match* razonable, únicamente se consigue un porcentaje de coincidencias del 30 % y por tanto tan sólo esa cantidad de los datos resulta integrada. A pesar de ello, es casi trivial la manera en la que un futuro desarrollador pueda mejorar esta funcionalidad. Además, conseguir una integración perfecta no formaba parte de los objetivos del *TFG* y por lo tanto no supone un problema ni un inconveniente en esta iteración.

Conocimientos adquiridos

Resumir los conocimientos tanto a nivel personal como a nivel de tecnologías adquiridos.

En cuanto a herramientas y tecnologías:

– Hadoop - BigData

– Hive

– Talend

– Kettle

– JHipster

- Spring
- Ficheros de configuracion
- Maven
- Sqoop
- Latex (texmaker)

En cuanto a nivel personal:

- Perseverancia
- Aceptar el cambio (Kettle - Talend)
- Buscar alternativas
- Confiar más en mi mismo y darme cuenta de que soy capaz de tomar decisiones por mi cuenta?????

Capítulo 8

Bibliografía

[1] Página web del ministerio de agricultura y pesca, alimentación y medio ambiente de españa. http://www.mapama.gob.es/es/agricultura/temas/sanidad-vegetal/productos-fitosanitarios/registro/menu.asp.

[2] Directiva 2009/128/ce. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:309:0071:0086:es:PDF.

[3] Cuaderno de explotación. http://www.mapama.gob.es/es/prensa/noticias/el-ministerio-de-agricultura-alimentacion-y-medio-ambiente-aprueba-un-modelo-armonizado-de-cuaderno-de-explotacion-de-los-productos-fitosanitarios/tcm7-311275-16. Online; Último acceso: 22 - October - 2017.

[4] Cuaderno de explotación agrícola campo agroslab. http://www.cuadernoexplotacion.es.

[5] Cuaderno de campo agrícola agricolum. https://agricolum.com.

[6] Cuaderno de campo agronev. http://jnevado.com/CUADERNOCAMPO/.

[7] Base de datos de pesticidas a nivel europeo. http://ec.europa.eu/food/plant/pesticides/eu-pesticides-database/public/?event=homepage&language=EN.

[8] Reglamento (ce) nº 1107/2009 del parlamento europeo y del consejo, de 21 de octubre de 2009, relativo a la comercialización de productos fitosanitarios y por el que se derogan las directivas 79/117/cee y 91/414/cee del consejo. https://www.boe.es/buscar/doc.php?id=DOUE-L-2009-82202.

[9] Formulario para solicitudes relativas al registro oficial de productos y material fitosanitario. http://www.mapama.gob.es/agricultura/pags/fitos/registro/fichas/pdf/Modelo

[10] Preguntas y respuestas frecuentes sobre el registro de productos fitosanitarios del ministerio de agricultura, alimentación y medio ambiente. http://www.mapama.gob.es/agricultura/pags/fitos/registro/fichas/pdf/FAQ.pdf.

[11] Apache™ hadoop®. http://hadoop.apache.org. Version XXXXXX.

[12] Apache hive™. https://hive.apache.org. Version XXXXXX.

[13] Apache sqoop™. http://sqoop.apache.org.

[14] Jhipster. http://www.jhipster.tech. Version XXXXXX.

[15] Spring framework. https://projects.spring.io/spring-framework/.

[16] Spring boot. https://projects.spring.io/spring-boot/.

[17] Angularjs. https://angularjs.org.

[18] Talend. https://www.talend.com.

[19] Pentaho data integration. http://www.pentaho.com/product/data-integration.

[20] Base de datos europea sobre pesticidas. https://data.europa.eu/euodp/es/data/dataset/s8QJJ4blyMdeI2AM1TtmXA/resource/ce5c6843-eb27-4168-9c28-b0f13b4ccbb8.

[21] Wikipedia - crawler web. https://en.wikipedia.org/wiki/Web_crawler.

[22] Tutorial de instalación de hadoop incorrecto para la versión ubuntu (16.04) del alumno. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/.

[23] Tutorial de instalación de hadoop de digital ocean para ubuntu 16-04. https://www.digitalocean.com/community/tutorials/how-to-install-hadoop-in-stand-alone-mode-on-ubuntu-16-04.

[24] Tutorial de tutorial's point para la instalación de hive. https://www.tutorialspoint.com/hive/index.htm.

[25] Página de hive - sección de instalación. https://cwiki.apache.org/confluence/display/Hive/GettingStarted.

[26] Node.js ®. https://nodejs.org/en/.

[27] Página oficial de yarn - instalación en linux ubuntu. https://yarnpkg.com/en/docs/install.

[28] Intellij idea. https://www.jetbrains.com/idea/.

[29] Pentaho. http://www.pentaho.com.

[30] Hitachi. http://www.hitachi.com.

[31] Hitachi vantara. http://www.pentaho.com/product/data-integration.

[32] Knime. https://www.knime.com.

[33] Repositorio de cloudera. https://repository.cloudera.com/content/repositories/releases/.

[34] Spring scheduling module. https://spring.io/guides/gs/scheduling-tasks/. Online; Ultimo acceso: asiqwjf.

[35] Wikipedia - linux. https://en.wikipedia.org/wiki/Linux.

[36] Wikipedia - nvidia. https://en.wikipedia.org/wiki/Nvidia.

[37] Grub. https://www.gnu.org/software/grub/.

[38] Jdl-studio. https://start.jhipster.tech/jdl-studio/.

[39] Git. https://git-scm.com.

[40] Trello. https://trello.com.

[41] Página web calculadora frepágina. http://www.calculadorafreelance.com.

[42] Centro científico tecnológico (cct) mendoza. http://www.cricyt.edu.ar/enciclopedia/terminos/ProducFito.htm.

[43] Wikipedia - fármaco. https://es.wikipedia.org/wiki/Fármaco.

[44] Glosario de términos fitosanitarios. https://www.ippc.int/static/media/files/publication/es/2016/06/ISPM_05_2016_Es_2016-06-23_FullReviewLRG-CPAM.pdf.

[45] Glosario de términos del mapama. http://www.mapama.gob.es/es/agricultura/temas/sanidad-vegetal/glosario-de-terminos-f-a-o/.

[46] Gnu general public license. https://www.gnu.org/licenses/gpl.html.

[47] Java ™. https://java.com/es/.

[48] Mysql. https://www.mysql.com. Version XXXXXX.

[49] Oracle corporation. https://www.oracle.com/index.html.

[50] Wikipedia - big data. https://es.wikipedia.org/wiki/Big_data.

[51] Wikipedia - apache cassandra. https://en.wikipedia.org/wiki/Apache_Cassandra.

[52] Wikipedia - operaciones etl. https://es.wikipedia.org/wiki/Extract,_transform_and_load.

[53] Wikipedia - h2 (dbms). https://en.wikipedia.org/wiki/H2_(DBMS).

[54] Wikipedia - kanban. https://es.wikipedia.org/wiki/Kanban_(desarrollo).

[55] Wikipedia - mapreduce. https://es.wikipedia.org/wiki/MapReduce.

[56] Wikipedia - mariadb. https://en.wikipedia.org/wiki/MariaDB.

[57] Wikipedia - mongodb. https://en.wikipedia.org/wiki/MongoDB.

[58] Método MoSCoW. https://en.wikipedia.org/wiki/MoSCoW_method.

[59] Wikipedia - microsoft sql server. https://en.wikipedia.org/wiki/Microsoft_SQL_Server.

[60] Wikipedia - mysql. https://en.wikipedia.org/wiki/MySQL.

[61] Oracle database. https://en.wikipedia.org/wiki/Oracle_Database.

[62] Wikipedia - postgresql. https://es.wikipedia.org/wiki/PostgreSQL.

Glosario

Apache Cassandra. Sistema de gestión de bases de datos distribuidas NoSQL gratis y libre diseñada para gestionar grandes cantidades de datos a través de diferentes servidores. *Wikipedia - Apache Cassandra* [51].

Big Data. Concepto que hace referencia a un conjuntos de datos tan grandes que aplicaciones informáticas tradicionales de procesamiento de datos no son suficientes para tratar con ellos y a los procedimientos usados para encontrar patrones repetitivos dentro de esos datos. Wikipedia - Big Data [50].

Certificado fitosanitario. Documento oficial que atestigua el estatus fitosanitario de cualquier envío sujeto a reglamentaciones fitosanitarias [FAO, 1990]. *Glosario de términos Mapama* [45].

H2 Database Engine. Sistema de gestión de bases de datos relacionales escrito en Java. Puede ser embebido en aplicaciones *Java* o lanzarse en modo cliente-servidor. *Wikipedia - H2 (DBMS)* [53].

Kanban. Un sistema de gestión de proceso visual que le indica qué producir, cuándo producirlo, y cuánto producir.. *Wikipedia - Kanban* [54].

Legislación fitosanitaria. Leyes básicas que conceden la autoridad legal a la organización nacional de protección fitosanitaria a partir de las cuales podrán elaborarse las reglamentaciones fitosanitarias [FAO, 1990; revisado FAO, 1995]. *Glosario de términos fitosanitarios* [44].

MapReduce. Modelo de programación para dar soporte a la computación paralela sobre grandes colecciones de datos en grupos de computadoras. Por regla general se utiliza en problemas con datasets de gran tamaño, alcanzando los petabytes de tamaño. *Wikipedia - MapReduce* [55].

MariaDB. Fork del sistema de gestión de bases de datos relacionales *MySQL* con el objetivo de mantener una versión libre de *MySQL* dada la adquisición del mismo por *Oracle*. *Wikipedia - MariaDB* [56].

Microsoft SQL Server. Sistema de gestión de bases de datos relacional desarrollado por *Microsoft*. *Wikipedia - Microsoft SQL Server* [59].

MongoDB. Base de datos NoSQL gratis, libre y multiplataforma orientado a documentos (formato JSON) con un esquema. *Wikipedia - MongoDB* [57].

MySQL Sistema de gestión de bases de datos relacional gratuito, libre y publicado bajo una licencia *GNU GPL* [46]. *Wikipedia - MySQL* [60].

Método MoSCoW. Técnica de priorización usada en la gestión de proyectos, análisis de negocio y desarrollo de software para llegar a un acuerdo común con los *stakeholders* (integrantes del proyecto) sobre la importancia que se debería dar a cada requisito. El término *MoSCoW* en sí mismo es un acrónimo derivado de la primera letra de cada categoría de priorización: *Must have* (debe tener), *Should have* (debería tener), *Could have* (podría tener) y *Won't have* (no tendrá). *Wikipedia - MoSCoW* [58].

Oracle Database. Sistema de gestión de bases de datos relacional orientado a objetos producido y desarrollado por *Oracle Corporation* [49]. *Wikipedia - Oracle Database* [61].

PostgreSQL Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia PostgreSQL. *Wikipedia - PostgreSQL* [62].

Producto fitosanitario. De acuerdo con la Organización Mundial de la Salud (OMS), se define al producto fitosanitario como la sustancia o mezcla de sustancias destinadas a prevenir la acción de, o destruir directamente, insectos, ácaros, moluscos, roedores, hongos, malas hierbas, bacterias y otras formas de vida animal o vegetal perjudiciales para la salud pública y también para la agricultura. Inclúyase en este ítem los plaguicidas, defoliantes, desecantes y las sustancias reguladoras del crecimiento vegetal o fitorreguladores. *CCT Mendoza* [42].

Sustancia, Sustancia activa, Fármaco. Un fármaco (o sustancia activa) es toda sustancia química purificada utilizada en la prevención, diagnóstico, tratamiento, mitigación y cura de una enfermedad, para evitar la aparición de un proceso

fisiológico no deseado o bien para modificar condiciones fisiológicas con fines específicos. En el dominio de aplicación actual, nos referiremos en concreto a aquellos fármacos utilizados en la prevención, diagnóstico, tratamiento, mitigación y cura de enfermedades relacionadas con los productos agrícolas, marinos o alimenticios. *Wikipedia - Fármaco* [43].

Tratamiento fitosanitario. Procedimiento oficial para matar, inactivar o eliminar plagas o para esterilizarlas o desvitalizarlas [FAO 1990; revisado FAO, 1995; NIMF 15, 2002; NIMF 18, 2003; CIMF, 2005]. *Glosario de términos fitosanitarios* [44].

Lista de Figuras

2.1. Diagrama de flujo del proceso actual de importación de un producto fitosanitario 5
2.2. Diagrama de flujo del potencial proceso de importación de un producto fitosanitario 6

4.1. Diseño conceptual de la solución 20
4.2. Diseño final de la solución 21
4.3. Diagrama de despliegue de la solución 22
4.4. Diagrama de componente y conector 23
4.5. Diagrama de clases y paquetes para soportar la automatización del *workflow* 24
4.6. Diagrama de secuencia del *workflow* implementado 25
4.7. Esquema de datos importados en *Hive* 27
4.8. Diagrama de clases y paquetes para soportar la automatización del *workflow* 28

5.1. Diagrama de las etapas de la prueba de concepto. 31

6.1. Tablero *Trello* para la gestión de las tareas del proyecto. 50

A.1. Esquema de datos importados en *MySQL* y *JHipster* 70

C.1. Diseño primitivo del sistema. 82
C.2. Segunda iteración del diseño del sistema. 83
C.3. Tercera iteración del diseño del sistema. 84

Lista de Tablas

3.1. Requisitos Funcionales del Sistema 17
3.2. Requisitos No Funcionales del Sistema 18
3.3. Requisitos Funcionales del Proyecto 18

6.1. Costes económicos del proyecto 54

C.1. Riesgos globales del proyecto 75
C.2. Riesgos tecnológicos del proyecto 76
C.3. Riesgos de alcance del proyecto 76
C.4. Riesgos de entorno de desarrollo del proyecto 77
C.5. Probabilidad de un riesgo 77
C.6. Impacto de un riesgo 77
C.7. Aceptación de un riesgo 78
C.8. Valoración riesgos globales del proyecto 78
C.9. Valoración riesgos tecnológicos del proyecto 78
C.10. Riesgos de alcance del proyecto 79
C.11. Valoración de los riesgos de entorno de desarrollo del proyecto 79
C.12. Priorización de riesgos del proyecto 80

Anexos

Anexos A

Datos

Esquema de datos

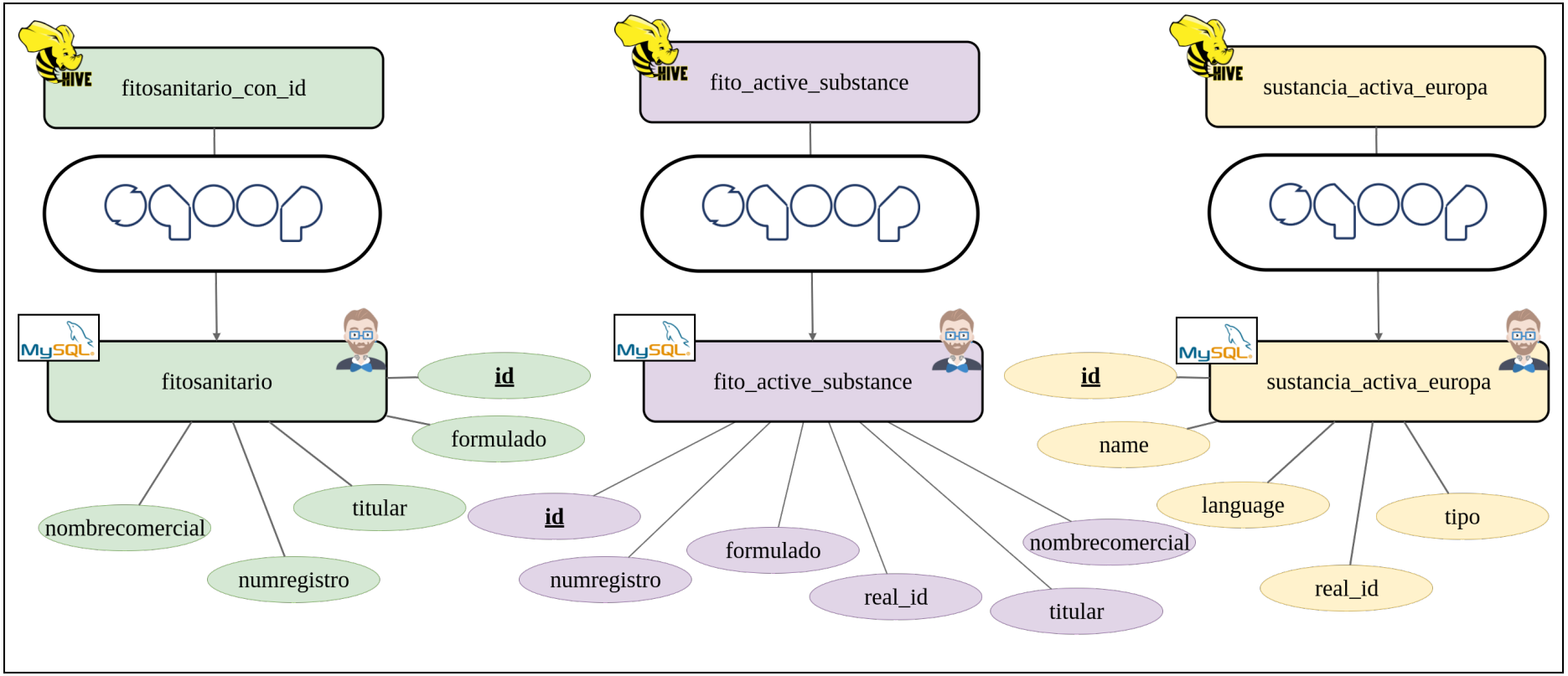


Figura A.1: Esquema de datos importados en *MySQL* y *JHipster*

Flujo de datos

Anexos B

Clientes

Clientes potenciales

- ***aGROSLab*** [4]

    - Registro de Explotaciones - Las parcelas se presentan para su selección en un innovador formato de celdas, facilitando la visualización de todas sus características (identificación, cultivo, superficie, . . .), el acceso al visor GIS, la aplicación de filtros y su selección individual o en bloque.
    - Registro de Parcelas Agrícolas - permite cargar las parcelas que componen su explotación a partir de la información generada por el aplicativo de gestión de la Solicitud de Ayudas PAC, para la mayoría de las Comunidades Autónomas.
    - Compras de Productos Fitosanitarios - incorpora un registro de compras de productos fitosanitarios, en el que podrá archivar todas sus facturas de compra en formato PDF y a partir del cual podrá registrar los tratamientos realizados.
    - Registro de Tratamientos Fitosanitarios- filtra los productos autorizados para cada uno de sus cultivos, presenta las plagas para las que puede ser aplicado según la nomenclatura del MAGRAMA y le informa del tipo y rango de dosis que puede ser aplicada.
    - Registro de Comercialización de Cosecha - facilita el registro de la comercialización de su cosecha, presentado en formato de celdas el conjunto de parcelas de su explotación con un determinado cultivo e informando gráficamente de aquellas en las que puede existir un problema con los plazos de seguridad de un tratamiento fitosanitario.
    - Receta Fitosanitaria

- Visor GIS con Capas - permite al agricultor visualizar gráficamente las parcelas que componen su explotación y la información de cultivos y los tratamientos realizados. Una herramienta especialmente útil a la hora de identificar sus diferentes parcelas y tener una visión de conjunto de toda su explotación.
- Unidades Homogéneas de Cultivo
- Control de Consumos de Fitosanitarios - permite llevar el control de los productos (fitosanitarios y fertilizantes) adquiridos y los aplicados
- Importaciones y Exportaciones
- Importaciones y Exportaciones

– ***Agricolum*** [5]

- Web + APP móvil y tableta
- Validación dosis cuaderno de campo
- Gestión de personal y maquinaria
- Informes personalizados y oficiales
- Control por GPS
- Control stock
- Gestión económica
- Rendimientos por campos y cultivos
- Soporte telefónico y por internet
- Gestión del cuaderno de campo
- Aplicación conectada con los datos del Sigpac y fitosanitarios MAGRAMA
- Sincronización de la información desde cualquier dispositivo
- Vista de tiempo actual y previsión semanal
- Ver histórico de todas las tareas realizadas
- Saber en tiempo real el precio del mercado
- Exportación de la información en otros formatos
- Importación de los datos de la PAC
- Generación del cuaderno de explotación oficial

– ***Cuaderno de campo Agronev*** [6]

- Labores - Asignación de labores a parcelas, siembra, semilla tratada, aperos, imputación de costes
- Abonado - Registro de Fertilización y Abonado. Composición de los Abonos, Forma de Abonado
- Tratamientos - Tratamientos fitosanitarios en parcelas, eficacia, asesor, equipo de aplicación
- Análisis de plaguicidas - Análisis de productos fitosanitarios, boletín de análisis, residuos detectados
- Recolección - Registro de recolección y loteado. Asignación de venta directa, imputación de costes
- Otros tratamientos - Aplicación de otros tratamientos fitosanitarios (Post-cosecha, Locales, Vehículos)
- Costes - Imputación de gastos / costes a parcelas. Directos / Selectivos.
- Gestión comercial - Compras, ventas, gastos, facturación, domiciliación bancaria SEPA, libro de fitos

Anexos C

Análisis

Análisis de riesgos completos

A continuación se exponen las diferentes fases del análisis de riesgos de manera detallada:

1. **Identificación de los riesgos**

   Se han intentado considerar el máximo número de riesgos y se han clasificado en diferentes categorías:

   - **Riesgos globales**

ID	Nombre	Explicación
RG_1	Plazos	El proyecto no se finaliza para la convocatoria de junio, septiembre o diciembre.
RG_2	Fallo del equipo	El equipo principal de desarrollo falla, se pierde o estropea.
RG_3	Incorporación mercado	El alumno se incorpora al mercado laboral durante el desarrollo del proyecto, a falta de varios meses de su finalización.
RG_4	Experiencia del alumno	El alumno no dispone de los conocimientos y preparación suficiente para el desarrollo del proyecto.

Tabla C.1: Riesgos globales del proyecto

   - **Riesgos tecnológicos**

ID	Nombre	Explicación
RT_1	Tecnología nueva	Se trata de una tecnología nueva.
RT_2	Software no probado	Se debe interactuar con software que no ha sido probado.

ID	Nombre	Explicación
RT_3	Interfaz especializada	Es requerida una interfaz de usuario especializada.
RT_4	Componentes diferentes	Se necesitan componentes de programa diferentes a los hasta ahora desarrollados.
RT_5	Rendimiento	Se debe interactuar con un B.D. cuya funcionalidad y rendimiento no ha sido probada.

Tabla C.2: Riesgos tecnológicos del proyecto

– **Riesgos de alcance**

ID	Nombre	Explicación
RA_1	Tamaño estimado	Tamaño estimado del proyecto
RA_2	Confianza en la estimación	Confianza en la estimación
RA_3	Número de elementos	Número de programas, archivos y transacciones
RA_4	Tamaño almacenamiento	Tamaño de las bases de datos involucradas
RA_5	Número de usuarios	Número de usuarios
RA_6	Número de cambios	Número de cambios en los requisitos
RA_7	Software reutilizado	Cantidad de software utilizado

Tabla C.3: Riesgos de alcance del proyecto

– **Riesgos de entorno de desarrollo**

ID	Nombre	Explicación
RE_1	Gestión proyectos	Hay herramientas de gestor de proyectos
RE_2	Gestión proceso desarrollo	Hay herramientas de gestión del proceso de desarrollo
RE_3	Análisis y diseño	Se usan métodos y herramientas específicas para el análisis y diseño
RE_4	Generadores de código	Hay generadores de código apropiados para la aplicación
RE_5	Pruebas	Hay herramientas de pruebas apropiadas
RE_6	Gestión de configuración	Hay herramientas de gestión de configuración apropiadas
RE_7	Base de datos	Se hace uso de una base de datos o repositorio centralizado
RE_8	Integración	Están todas las herramientas de desarrollo integradas
RE_9	Formación	Se ha proporcionado formación a todos los miembros del equipo de desarrollo

ID	Nombre	Explicación
RE_10	Expertos	Hay expertos a los cuales solicitar ayuda acerca de las herramientas
RE_11	Ayuda online	Hay ayuda en línea y documentación disponible
RE_12	Diseño arquitectónico	Se utiliza un método específico para el diseño arquitectónico y de datos
RE_13	Métricas de calidad	Se disponen métricas de calidad para todos los proyectos de software
RE_14	Métricas de productividad	Se disponen de métricas de productividad

Tabla C.4: Riesgos de entorno de desarrollo del proyecto

2. **Análisis del riesgo**

   Para esta fase se han empleado los tres medidores del riesgo: la probabilidad, el impacto y la aceptación:

   - **Tabla para estimar la probabilidad de un riesgo:**

Valor	Descripción
Bajo (1)	La amenaza se materializa a lo sumo una vez cada año.
Medio (2)	La amenaza se materializa a lo sumo una vez cada mes.
Alto (3)	La amenaza se materializa a lo sumo una vez cada semana.

Tabla C.5: Probabilidad de un riesgo

   - **Tabla para estimar el impacto de un riesgo:**

Valor	Descripción
Bajo (1)	El daño derivado de la materialización de la amenaza no tiene consecuencias relevantes para la consecución de los objetivos.
Medio (2)	El daño derivado de la materialización de la amenaza tiene consecuencias reseñables para la consecución de los objetivos.
Alto (3)	El daño derivado de la materialización de la amenaza tiene consecuencias graves reseñables para la consecución de los objetivos.

Tabla C.6: Impacto de un riesgo

   - **Tabla para estimar la aceptación de un riesgo:**

Valor	Descripción
Riesgo $\leq$	La organización considera el riesgo poco reseñable.

Valor	Descripción
Riesgo ≥ 4	La organización considera el riesgo reseñable y debe proceder a su tratamiento.

Tabla C.7: Aceptación de un riesgo

La aceptación es una medida delimitadora que define aquellos riesgos que son considerados aceptables y aquellos ante los que se deben tomar medidas. Para esta medida se ha establecido un criterio de aceptación de 4. Cualquier riesgo cuyo valor sea menor que 4 se considera aceptable y por tanto un riesgo poco reseñable, mientras que aquellos que se encuentran por encima de 4 se consideran reseñables y se debe proceder a su tratamiento.

El cálculo de la gravedad del riesgo y su aceptación se realiza de la siguiente manera: se multiplica la probabilidad por el impacto, y si dicho valor excede el límite del criterio de aceptación, el riesgo se considera reseñable. A continuación, en base a las métricas anteriores, se especifican los riesgos de la fase 1 en las mismas categorías iniciales. Se resaltan en rojo aquellos riesgos cuya aceptación supera el 4.

- **Riesgos globales**

ID	Nombre	Probabilidad	Impacto	Riesgo
RG_1	**Plazos**	**2**	**3**	**6**
RG_2	Fallo del equipo	1	3	3
RG_3	Incorporación mercado	1	2	2
RG_4	Experiencia del alumno	2	2	4

Tabla C.8: Valoración riesgos globales del proyecto

- **Riesgos tecnológicos**

ID	Nombre	Probabilidad	Impacto	Riesgo
RT_1	Tecnología nueva	3	1	3
RT_2	**Software no probado**	**2**	**3**	**6**
RT_3	Interfaz especializada	1	1	1
RT_4	Componentes diferentes	3	1	3
RT_5	Rendimiento	2	2	4

Tabla C.9: Valoración riesgos tecnológicos del proyecto

– **Riesgos de alcance**

ID	Nombre	Probabilidad	Impacto	Riesgo
RA_1	**Tamaño estimado**	**2**	**3**	**6**
RA_2	Confianza en la estimación	2	2	4
RA_3	**Número de elementos**	**2**	**3**	**6**
RA_4	Tamaño almacenamiento	1	3	3
RA_5	Número de usuarios	1	3	3
RA_6	**Número de cambios**	**2**	**3**	**6**
RA_7	Software reutilizado	1	1	1

Tabla C.10: Riesgos de alcance del proyecto

– **Riesgos de entorno de desarrollo**

ID	Nombre	Probabilidad	Impacto	Riesgo
RE_1	Gestión proyectos	1	1	1
RE_2	Gestión proceso desarrollo	1	1	1
RE_3	Análisis y diseño	1	2	2
RE_4	Generadores de código	1	1	1
RE_5	Pruebas	2	2	4
RE_6	Gestión de configuración	2	2	4
RE_7	Base de datos	1	3	3
RE_8	Integración	1	1	1
RE_9	**Formación**	**2**	**3**	**6**
RE_10	Expertos	2	1	2
RE_11	Ayuda online	2	2	4
RE_12	Diseño arquitectónico	2	1	2
RE_13	Métricas de calidad	3	1	3
RE_14	Métricas de productividad	3	1	3

Tabla C.11: Valoración de los riesgos de entorno de desarrollo del proyecto

3. **Priorización de riesgos**

Esta fase incluye todos los riesgos, ordenados de mayor a menor severidad. Se resaltan en rojo los riesgos que habrá que considerar en un plan de defensa estratégico posterior:

ID	Nombre	Riesgo
RG_1	**Plazos**	**6**
RT_2	**Software no probado**	**6**
RA_1	**Tamaño estimado**	**6**
RA_6	**Número de cambios**	**6**
RE_9	**Formación**	**6**
RT_5	Rendimiento	4
RA_2	Confianza en la estimación	4
RE_5	Pruebas	4
RE_11	Ayuda online	4
RG_2	Fallo del equipo	3
RT_1	Tecnología nueva	3
RT_4	Componentes diferentes	3
RA_4	Tamaño almacenamiento	3
RA_5	Número de usuarios	3
RE_7	Base de datos	3
RE_13	Métricas de calidad	3
RE_14	Métricas de productividad	3
RG_3	Incorporación mercado	2
RE_3	Análisis y diseño	2
RE_10	Expertos	2
RE_12	Diseño arquitectónico	2
RT_3	Interfaz especializada	1
RT_7	Software reutilizado	1
RE_1	Gestión proyectos	1
RE_2	Gestión proceso desarrollo	1
RE_4	Generadores de código	1
RE_8	Integración	1

Tabla C.12: Priorización de riesgos del proyecto

Como se puede apreciar, hay 5 riesgos cuyo factor de gravedad es preocupante y deben ser tratados acordemente:

a) RG_1. Riesgo global "Plazos". Tiene que ver con el hecho de no acabar el proyecto dentro de los plazos establecidos para su defensa. Hay 2 fechas recomendables para su defensa, la primera en Junio de 2017 y la segunda en Septiembre de 2017. No obstante, se dispone de otra oportunidad en

Diciembre de 2017, aunque sería la menos recomendable dado que supondría el retraso de la defensa y con ello la dificultad del estudiante de realizar otras actividades mientras tanto. A partir de Diciembre, la consecuencia sería volver a matricularse en el proyecto y aportar las tasas de la matrícula por segunda vez.

b) RT_2. Riesgo de tecnologías "Software no probado". Tiene que ver con la probabilidad de usar en el proyecto software que previamente no ha sido probado y pueda fallar. Obtuvo una valoración de gravedad de 6/6 puesto que si bien es cierto que todas las tecnologías han sido probadas individualmente y se sabe que funcionan bien, el proceso en su conjunto no ha sido probado. No se sabe si es viable o no.

c) RA_1. Riesgo de alcance "Tamaño estimado". Tiene que ver con el hecho de que el proyecto resulte mucho más grande de lo estimado inicialmente, y por diferentes circunstancias no se llegue a finalizar.

d) RA_6. Riesgo de alcance "Número de cambios". Otro riesgo es el hecho de que los requisitos cambien constantemente, bien porque los clientes lo solicitan bien porque las propias tecnologías lo imponen.

e) RE_9. Riesgo de entorno de desarrollo "Formación". Este riesgo trata con el hecho de que el alumno disponga de la formación necesaria y suficiente para lograr los objetivos propuestos.

4. **Planificación de la gestión de riesgos**

En esta fase se recogen las conclusiones mitigadoras acerca de los riesgos "preocupantes"del proyecto, en relación a su factor de gravedad:

a) RG_1: Plazos - Como medida mitigante, el alumno deberá dedicar un horario de jornada completa a la realización del proyecto durante el verano del año 2017.

b) RT_2: Software no probado - La contrapartida y defensa de este riesgo es desarrollar o experimentar primero con una prueba de concepto para validar que las tecnologías en su conjunto funcionen correctamente.

c) RA_1: Tamaño estimado - La solución desde un principio debe definir bien el alcance y determinar aquellas cosas que formarán parte de la solución y aquellas que no lo harán.

d) RA_6: Número de cambios - El alumno y el profesor deben acordar al principio unos requisitos fijos que no podrán ser modificables, en conjunto

con el hecho de definir claramente el alcance de la solución.

e) RE_9: Formación - Para mitigar este riesgo, el alumno debe estar en constante aprendizaje, utilizando los manuales y tutoriales de las diferentes herramientas de las que va a hacer uso durante el proyecto. Además, el alumno tendrá a su disposición al director del proyecto para consultar dudas y a los diferentes foros tecnológicos de Internet.

Análisis de diseños alternativos

Al igual que los requisitos, el diseño de la solución también ha sufrido constantes cambios. Inicialmente la propuesta de trazabilidad de la solución es la que se puede observar en la figura 1.

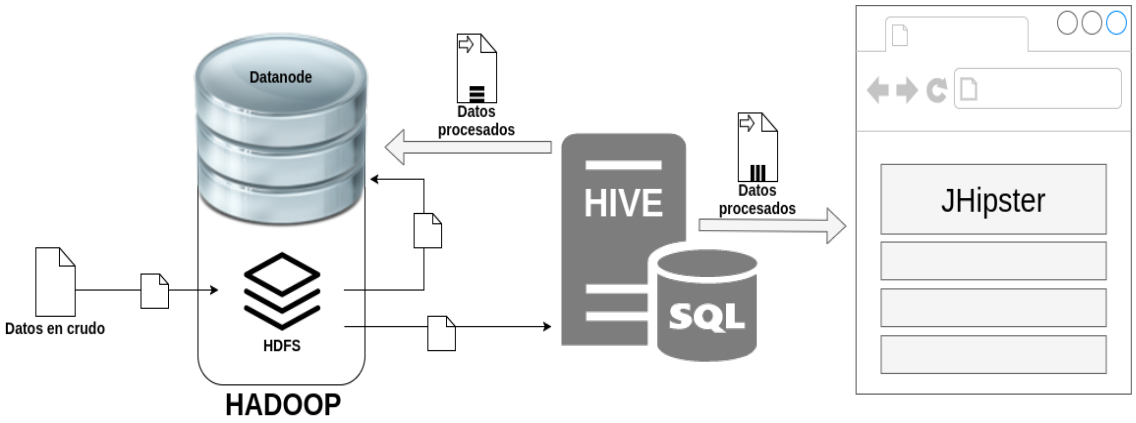


Figura C.1: Diseño primitivo del sistema.

Como se puede observar, inicialmente el concepto giraba alrededor de las tres tecnologías core: Hadoop, Hive y JHipster. No se tuvo en consideración otras herramientas puesto que se pensaba que era suficiente para resolver el problema. Los datos en crudo, extraídos de la web del magrama o de otras fuentes heterogéneas, serían almacenados en Hadoop, en un nodo local mediante el sistema de ficheros HDFS, y posteriormente sería HIVE el encargado de procesarlos en su totalidad hasta conseguir almacenarlos en un esquema común. Además, la misma "base de datos" de Hive funcionaría como base de datos para la aplicación desarrollada con JHipster, sirviendo en todo momento ese esquema único para la visualización del mismo en un navegador web. Este diseño, conceptualmente fue la solución ideal para el problema planteado; no obstante, debido a que JHipster no ofrece soporte para cambiar la base de datos con la que se construye la aplicación y mucho menos soporte para Hive o Hadoop, tras muchos intentos frustrados de conseguir esta conectividad directa, se optó por una solución diferente, alejada de este diseño ideal pero rebelde, en contra de

los paradigmas de programación que JHipster impone. La alternativa a este diseño que dió resultados y eliminó complicaciones fue la que se observa en la figura 2.

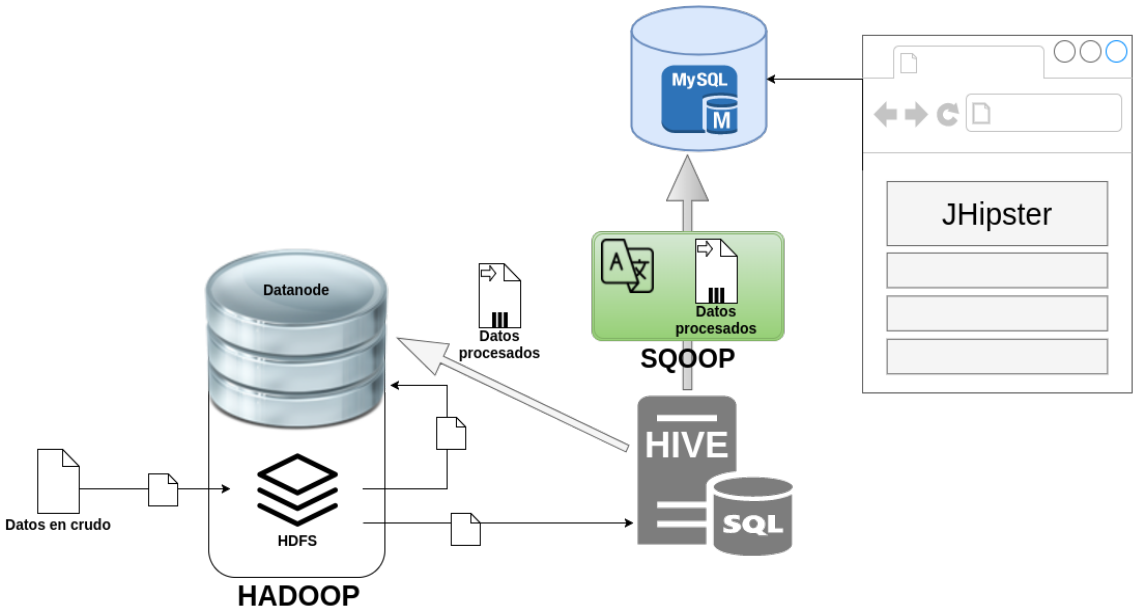


Figura C.2: Segunda iteración del diseño del sistema.

En la segunda fase del diseño, se observa la evolución de la idea, condicionada por los problemas anteriormente mencionados. En primer lugar, se conservó la base de datos nativa de JHipster, en este caso una base de datos MySQL convencional. En ella se almacenaria únicamente la estructura final del esquema unificado, con los datos finales. Dichos datos tendrían que ser pasados desde Hive mediante una herramienta de transformación y transporte de datos. En este caso se usó Sqoop, una herramienta gratuita que permite transportar los datos desde Hive a MySQL, puesto que ofrece soporte tanto para Hadoop, HDFS y Hive como para MySQL.

Una vez resuelto el problema de la visualización de los datos, lo próximo que se detectó fue esa necesidad de procesamiento de los datos en crudo antes de incluso exponerlos como un esquema relacional en Hive. Para eso, lo mejor era hacer uso de algún programa de procesado de ficheros y una de las mejores opciones aparentes era Pentaho, un programa completo de transformación de ficheros. Pentaho venía con soporte para Hadoop y HDFS, por lo que gracias a eso se pudo trabajar con ficheros directamente extraídos de Hadoop, y almacenarlos directamente en Hadoop. Los cambios se pueden observar en la figura 3.

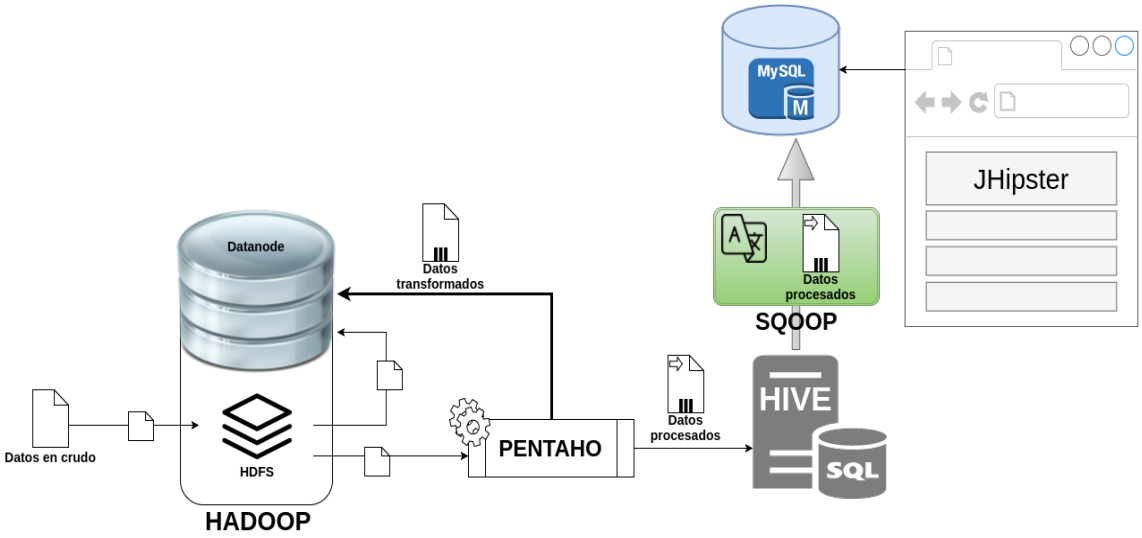


Figura C.3: Tercera iteración del diseño del sistema.

INACABADO

Anexos D

Gestión

Cálculo del coste de la hora de trabajo del alumno

A continuación se exponen los cálculos desglosados y en detalle para calcular el coste de la hora de trabajo del alumno. Para conseguir el precio por hora para un salario de 1,500€ hacemos lo siguiente:

1. Calculamos el salario bruto anual:

   Salario mensual × 12 meses

   1,500.00 × 12 = **18,000€**

2. Hallamos el precio básico por hora:

   Salario anual ÷ 2,080 horas posibles (8 horas × 5 días × 52 semanas)

   18,000.00 ÷ 2,080 = **8.65€**

3. Sumamos el precio de las horas que no trabajarás:

   14 días festivos × 8 hs. = 112 hs. +

   21 días de vacaciones × 8 = 168 hs. +

   7 días de incapacidad × 8 hs. = 56 hs.

   336 hs. × 8.65 de coste básico = 2,907.69€

4. Calculamos el valor del tiempo que dedicas a presupuestos, reuniones, venta, formación... (tiempo administrativo):

   50 % (2,080 hs. posibles - 336 horas que no trabajarás) × 8.65 coste básico.

   50 % × 1,744 hs. × 8.65 = 7,546.15€

5. Calculamos el total de tus gastos fijos:

   Alquiler mensual: 100.00 +

Servicios mensuales: 50.00 +

Autónomos mensual: 260.00 +

Otros gastos fijos mensuales: 50.00

Gastos fijos mensuales 460.00 × 12 meses = **5,520€ fijos anuales**

6. Sumamos el valor de las horas que no trabajas más el valor del tiempo de administración y el de los gastos fijos para obtener el precio extra anual por tu trabajo.

   2,907.69 + 7,546.15 + 5,520.00 = **15,973.85€ de precio extra anual**

7. Ahora calculamos lo que ganarás al año:

   2,080 hs. posibles al año - 336 hs. de vacaciones, festivos e incapacidad - 872 hs. de reuniones y presupuestos × 8.65 coste básico

   872 horas de trabajo × 8.65 = 7,546.15€ de beneficio anual.

8. Calculamos el porcentaje de rentabilidad dividiendo el coste entre el beneficio:

   15,973.85 ÷ 7,546.15 = 211.682 %

9. Por último para calcular la hora de trabajo sumamos el porcentaje de rentabilidad y el porcentaje de beneficio deseado a nuestro precio básico:

   8.65 + 8.65 × 211.682 % + 8.65 × 20 % = **28.70€ por hora de trabajo**.