Escuela Superior de Ingeniería

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

REVISIÓN SISTEMÁTICA DE LA LITERATURA

AUTOR(A): Ángel Rafael González Toro

14 de mayo de 2017

Escuela Superior
de Ingeniería

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

REVISIÓN SISTEMÁTICA DE LA LITERATURA

AUTOR(A): ÁNGEL RAFAEL GONZÁLEZ TORO
DIRECTOR(A): IVAN RUÍZ RUBE

Cádiz, 14 de mayo de 2017

Agradecimientos

Introduzca aquí, si lo desea, los agradecimientos.

Resumen

Revision Sistemática de la Literatura consiste en la creación de una aplicación web para determinar el estado del arte sobre el cuál se sustenta una determinada área de investigación. La aplicación realizará una búsqueda de referencias bibliográficas en una serie de motores de búsquedas determinados y será capaz de filtrar la información obtenida acorde con los parámetros que le indique el usuario. Para este proceso, se desarrollará una web que permitirá automatizar cada una de las etapas del proceso para realizar revisiones sistemáticas de la literatura. Tras la ejecución de este proceso, se podrán exportar los metadatos de todos los estudios encontrados a través de gráficos o ser exportados a un gestor de referencias bibliográficas.

Palabras clave: revisión sistemática, literatura, referencia, systematic literature review, metadatos, investigación, mendeley.

Índice general

Índice de figuras

Índice de tablas

Parte I

Prolegómeno

Capítulo 1

Introducción

A continuación, se describe la motivación del presente proyecto y su alcance. También se incluye un glosario de términos y la organización del resto de la presente documentación.

1.1. Motivación

Es algo muy común en el ámbito educativo, que tanto estudiantes pre-doctorales como para cualquier persona ya doctorada se dedique en algún momento de su carrera a realizar una investigación sobre una determinada área. Para la realización de la misma, estas personas necesitarán realizar una búsqueda exahustiva de información o recursos y comprobar que existan, o no, evidencias de dicho conocimiento. Es decir, deben ser capaces de resumir la evidencia existente acerca de un tema o identificar, si se da el caso, de que haya un vacío de conocimiento sobre esa investigación.

Este estudio del arte sobre algún tema específico puede ayudarnos entre otras cosas:

- Resumir la evidencia existente acerca de una determinada área de investigación o tema de estudio.
- Identificar los vacíos en una investigación para sugerir más áreas o recursos de investigación.
- Creación de nuevas áreas o actividades de investigación.

Una **revisión sistemática de la literatura** (*Systematic Literature Review*) [Rube, 2013] es un medio para evaluar e interpretar la investigación disponible relativa a una determinada área de interés. Por tanto, una persona que desea realizar un estudio sobre un determinado tema, deberá buscar referencias bibliográficas por medio de artículos, aportes vía internet u otro tipos de documentos y saber interpretar dichos resultados para llegar a una conclusión sobre este estudio.

La mayor parte de la investigación comienza con una revisión sistemática de la literatura, por contra, éstas carecen de valor científico si no se trata de una revisión exhaustiva y justa. Éste es el motivo principal para el desarrollo de revisiones sistemáticas. Una revisión sistemática sintetiza de una manera eficaz e idónea el estudio del arte de un determinado tema. Sin embargo, hay que seguir una estrategia de búsqueda que deberá permitir la integridad de la búsqueda que se evaluará. Concretamente, los investigadores deberán hacer un esfuerzo por identificar e investigar

cada uno de los recursos que encuentre y presentar un informe que lo sustente.

Barbara Kitchenham propuso una guía [Kitchenham, 2007] o conjunto de normas para promover estos estudios de la literatura en Ingeniería del Software. Está dirigida principalmente a investigadores de ingeniería del software, incluyendo a los doctorados. Este documento no cubre ningún proceso estadístico para resumir los resultados cuantitativos en los diferentes estudios (meta-análisis), ni explicar las diferentes implicaciones que podrían obtenerse acorde a los resultados encontrados, pero sí las directrices correctas para facilitar el estudio y que están siendo tan empleadas en otras disciplinas como la medicina y las ciencias sociales.

Para estudiar la literatura de un tópico a evaluar, es necesario de una metodología confiable, rigurosa y extendida en la comunidad investigadora. Y este proyecto de fin de carrera se encargará de ayudar al investigador a encontrar las referencias bibliográficas que le ayude a facilitar el estudio y la exportación de los informes para determinar las conclusiones finales del estudio siguiendo la línea indicada por Kitchenham.

1.2. Alcance

Un **SLR** (*Systematic Literature Review*) no es más que una metodología rigurosa para identificar, analizar e interpretar de una forma no sesgada todas las evidencias referentes a una pregunta de investigación.

Este proyecto se plantea para **facilitar al investigador en el proceso de búsqueda de referencias bibliográficas en diferentes motores de búsquedas especializados** que proporciona internet **y ayudar al investigador a exportar los resultados obtenidos** con el fin de poder elaborar un informe con las conclusiones del estudio.

Para la creación de este PFC, se ha realizado una aplicación web donde el usuario puede realizar este proceso de búsqueda a través de varios motores de búsquedas específicos con el ámbito universitario e investigador de una sóla vez, evitando que el usuario tenga que repetir el mismo proceso en cada uno de los sitios. Estos motores de búsquedas implican una serie de reglas o unos formatos de búsqueda concretos, y la aplicación se encargará de esta tarea tediosa para el investigador.

La literatura encontrada en cada uno de estos medios universitarios será almacenada en un gestor de referencias que los estudiantes de la Universidad de Cádiz pueden manejar denominado **Mendeley**. La aplicación web conectará con este gestor y almacenará todas las referencias que han sido encontradas y creará un conjunto de carpetas y documentos donde el investigador podrá saber en todo momento en qué motor de búsqueda ha sido encontrada la referencia, la revisión sistemática a la que pertenece, así como todo el detalle disponible de cada una de las referencias.

Una vez que hemos obtenido toda la literatura a estudiar en cada uno de estos medios unversitarios, el usuario tendrá la difícil tarea de indicar si las referencias bibliográficas encontradas siguen los criterios necesarios para incluirlos en el proceso de estudio. Por medio de esta aplicación, el usuario podrá diferenciar cuáles cumplen los requisitos para ser incluidos a través de unos criterios previamente definidos por el usuario, así como incluir otra información que Mendeley

por defecto no ha podido encontrar.

Por último, y una vez realizado el estudio del arte, podemos obtener un informe con las valoraciones y criterios propuestos por el investigador de un simple vistazo por medio de la exportación de documentos en diferentes formatos de salida o gráficos que ilustre el estudio realizado, así como las conclusiones finales obtenidas.

A continuación se enumeran y describen los principales objetivos que se esperan alcanzar cuando el sistema a desarrollar esté en producción.

1.2.1. Objetivos

El objetivo principal de este proyecto, será ayudar al usuario a seguir la mayor parte de las directrices de Barbara Kitchenham, ya que el estudio del arte de una área de investigación predeterminada no puede ser automatizado. Sin embargo, podemos ayudar al usuario a quitarle tiempo de búsqueda de recursos a través de un buscador de referencias en motores de búsquedas, que en posteriores capítulos describiremos, y ayudar al usuario a determinar las conclusiones finales por medio de la exportación de la información a través de gráficos o ficheros en diferentes formatos.

Todo este proceso será desarrollado a través de una aplicación web donde el usuario podrá crear revisiones sistemáticas de la literatura, realizar búsquedas, clasificar los documentos y poder exportar la información a través de varios medios. Podemos ver más detalles de estos objetivos en tablas 1.1, 1.2, 1.3 y 1.4.

OBJ-01	Creación de una aplicación web para realizar Revisiones Sistemáticas de la Literatura.
Descripción	Se debe diseñar una aplicación web donde el usuario podrá crear revisiones sistemáticas de la literatura.

Tabla 1.1: OBJ-01: Creación de una aplicación web para realizar Revisiones Sistemáticas de la Literatura.

OBJ-02	Realizar Búsquedas de Referencias Bibliográficas.
Descripción	La aplicación web deberá poder realizar varias búsquedas de referencias bibliográficas con la información predefinida por el usuario y almacenando dichos recursos en un sistema de gestión de referencias (Mendeley) para su futuro estudio.

Tabla 1.2: OBJ-02: Realizar Búsquedas de Referencias Bibliográficas.

1.3. Glosario de Términos

Esta sección contiene una lista de los acrónimos específicos y principales términos del dominio del sistema.

OBJ-03	Clasificar las referencias bibliográficas.
Descripción	La aplicación web deberá poder ayudar al usuario a clasificar las referencias bibliográficas encontradas en las búsquedas de cada una de las revisiones sistemáticas de la literatura.

Tabla 1.3: OBJ-03: Clasificar las referencias bibliográficas.

OBJ-04	Exportar información de los SLR y conclusiones finales.
Descripción	La aplicación web deberá poder ayudar al usuario a realizar un informe con las conclusiones finales del SLR a través de gráficos que reflejen la situación del mismo o diferentes ficheros de texto.

Tabla 1.4: OBJ-04: Exportar información de los SLR y conclusiones finales.

1.3.1. Acrónimos

DRY Don't Repeat yourself

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

MVC Modelo Vista Controlador

OO Orientado a Objetos

PFC Proyecto Fin de Carrera

RSL Revisiones Sistemáticas de la Literatura

SLR Systematic Literature Review

UCA Universidad de Cádiz

UML Lenguaje Unificado de Modelado

1.3.2. Términos

Grails Framework para aplicaciones web libre desarrollado sobre el lenguaje de programación Groovy.

Groovy Lenguaje de programación orientado a objetos implementado sobre la plataforma Java.

Mendeley Aplicación web y de escritorio, proprietaria y gratuita. Permite gestionar y compartir referencias bibliográficas y documentos de investigación, encontrar referencias y documentos y

colaborar en línea.

Metadato Dato que describe otro dato. Por lo general, un grupo de metadatos se refiere a un grupo de datos que describen el contenido de una referencia bibliográfica.

1.4. Organización del documento

A continuación se describe de manera resumida el contenido de los capítulos en los que se encuentra dividido este PFC:

- En el Capítulo 1 se ha realizado un recorrido introductorio sobre el contexto donde nos moveremos así como la motivación al respecto, haciendo hincapié en los objetivos que se pretende satisfacer.
- En el Capítulo 2 se incluye la planificación del proyecto, plazos, entregables, recursos utilizados, así como la metodología de ingeniería de software empleada.
- En el Capítulo 3 ...
- En el Capítulo 4 ...
- En el Capítulo 5 ...
- En el Capítulo 6 ...
- En el Capítulo 7 ...
- En el Capítulo 8 ...
- En el Capítulo 9 ...
- En el Capítulo 10 ...
- En el Capítulo 11 ...

Respecto al software entregado en soporte informático, se distribuye en los siguientes directorios:

- **codigoFuente** Código fuente de todos los archivos creados para desarrollar el sistema.
- **instalación** Instrucciones necesarias para la instalación del sistema.
- **memoria** Archivos tex y pdf de la memoria del proyecto.
- **presentación** Archivos tex y pdf de la presentación del proyecto.
- **recursos** Archivos de diagramas, imágenes y otros archivos de interés.

Capítulo 2

Planificación

En esta sección se describen todos los aspectos relativos a la gestión del proyecto: metodología, organización, costes, planificación, riesgos y aseguramiento de la calidad.

2.1. Metodología de desarrollo

La metodología de desarrollo empleada es la metodología de desarrollo ágil Scrum (Ver 2.1), ya que el proyecto puede tener un cambio constante de requisitos y además se pretende mostrar un incremento ejecutable cada cierto tiempo, por lo que el trabajo se irá dividiendo en hitos o **sprints**.

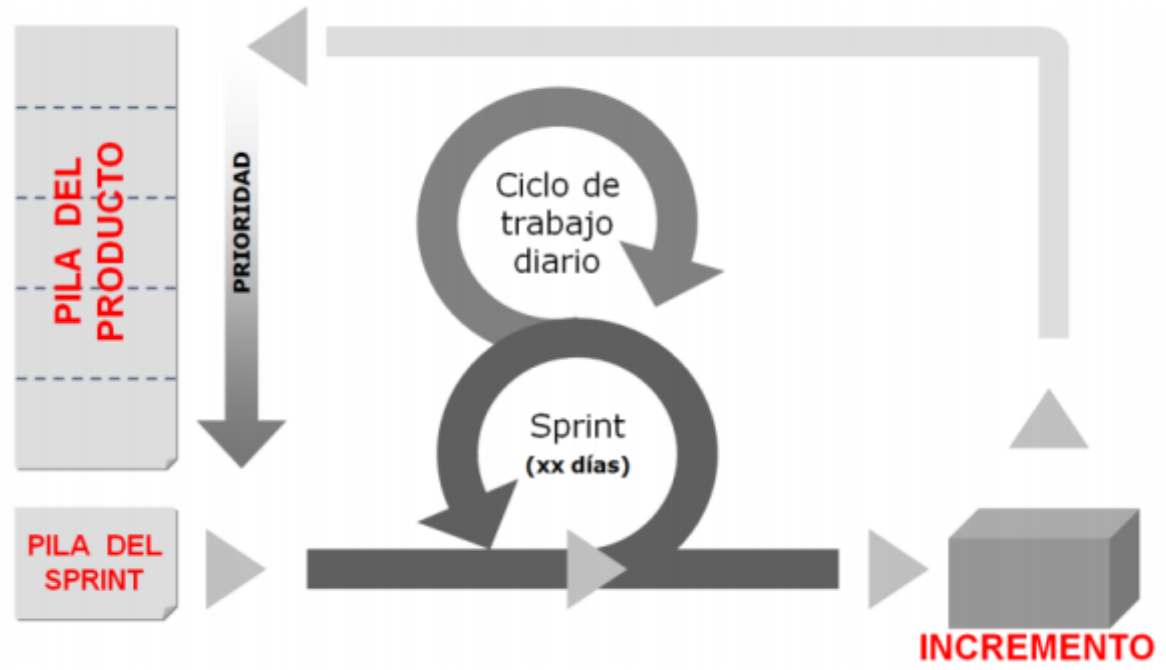

Figura 2.1: Diagrama del ciclo iterativo scrum

El marco técnico de scrum, [Bok, 2016] está formado por un conjunto de prácticas y reglas que dan respuesta a los siguientes principios de desarrollo ágil:

- Gestión evolutiva del producto, en lugar de la tradicional o predictiva.
- Calidad del resultado basado en el **conocimiento tácito de las personas**, antes que en el explícito de los procesos y la tecnología empleada.
- Estrategia de **desarrollo incremental** a través de iteraciones (sprints).

Con la visión general del proyecto, y a partir de ella se especifica y da detalle a las funcionalidades que se desean obtener en primer lugar. Cada ciclo de desarrollo o iteración (**sprint**) finaliza con la entrega de una parte operativa del producto (**incremento**). La duración de cada sprint ha sido entre una semana y dos meses.

Para diseñar el modelo de datos y la documentación se empleará UML (Lenguaje Unificado de Modelado) y un paradigma de programación OO (Orientado a Objetos). Además se buscará usar una arquitectura MVC (Modelo Vista Controlador) y tener cada parte lo más independiente posible del resto. Esta elección podrá permitir además una mejor integración entre todos los componentes y una mayor facilidad de mantenimiento y ampliación en el futuro.

2.2. Planificación del proyecto

En primer lugar, se realiza una planificación inicial y una investigación sobre las directrices que marca Barbara Kitchenham para crear una revisión sistemática de la literatura sobre un área determinada. Dentro de esta planificación inicial se comprueba que es necesario realizar un sistema de gestión para las distintas revisiones sistemáticas de un usuario, así como un sistema de búsquedas de referencias bibliográficas que pueda sincronizar toda la información con el gestor de referencias Mendeley, y, finalmente, su posterior clasificación según los criterios indicados por el usuario y la exportación de conclusiones finales.

Una vez realizada la planificación inicial y analizado todos los objetivos principales del proyecto se decide realizar el desarrollo del proyecto en dos subproyectos. El primero de ellos, será la aplicación web, donde el usuario podrá acceder a todas las opciones para las creaciones de las revisiones sistemáticas de la literatura. Por otro lado, el segundo de ellos consistirá en la creación de una librería JAR donde incluirá todo el proceso de búsquedas de referencias bibliográficas en segundo plano y de manera paralela a cualquier otra búsqueda que se realice. Esta librería, deberá ser incluida en el primer subproyecto, por lo que necesitará de un periodo de tiempo para integrar ambas aplicaciones.

Con toda esta información, se opta por dividir el roceso en ocho hitos tal y como podemos ver en la tabla 2.1. El tiempo se indica en dias. Cada uno de estos hitos, tendrá asociados varias subtareas o subobjetivos que deberán realizarse para dar por bueno el hito principal. El hito 0 corresponderá a la adquisición de conocimientos mientras que el hito 7 es asignado a la redacción de la memoria.

Para mostrar el desarrollo detallado de toda la aplicación y desarrollo del software se muestran en las figuras 2.2, 2.3, 2.4, 2.5 y 2.6 unos diagramas de Gantt ordenados cronológicamente haciendo referencia a cada uno de los hitos que hemos explicado anteriormente. Los diagramas han sido realizados con la herramienta GanttProject [Project, 2015]. Cabe indicar, que el hito de mayor duración es el último, que corresponde a la redacción de la documentación de este proyecto, ya

Sprint	Descripción	Estimado	Real
Hito 0	Propuesta inicial del proyecto	7	5
Hito 1	Investigación y adquisición de conocimientos	50	63
Hito 2	Desarrollo Aplicación Web	110	160
Hito 3	Desarrollo Aplicación Mendeley-REST en Java	60	100
Hito 4	Integración de aplicaciones	30	26
Hito 5	Pruebas del sistema y rendimiento	10	5
Hito 6	Despliegue de la aplicación	20	25
Hito 7	Redacción de memoria	180	280

Tabla 2.1: Tiempo estimado frente a tiempo real de cada Sprint

que se ha ido redactando a medida que se ha ido completando cada una de las fases del proyecto.

2.3. Organización

En este apartado recogemos las personas o roles que se encuentran involucrados en el proyecto así como una relación de los recursos inventariables utilizados en el proyecto: equipamiento informático, herramientas empleadas, etc.

2.3.1. Roles

En los proyectos que siguen una metodología ágil por SCRUM podemos encontrarnos con tres tipos de roles [Bok, 2016]:

- **Dueño del producto.** Es el representante del cliente, el cuál conoce el entorno de negocio del cliente, las necesidades y el objetivo que se persigue con el sistema que se construye. Tiene la visión del producto, así como las necesidades concretas del proyecto, para poder priorizar eficientemente el trabajo. El tutor de este proyecto es el más claro ejemplo de representante, así como la UCA ejerce de cliente.

- **Equipo de desarrollo.** Grupo o grupos de trabajo que desarrollan el producto. Todos conocen y comprenden la visión del proprietario del producto. Además, aportan y colaboran con el proprietario en el desarrolo de la pila del producto. En este proyecto el equipo está compuesto únicamente por el autor.

- **Scrum master.** Es el responsable del cumplimiento de las reglas de un marco de scrum técnico, asegurando que se entienden en la organización, y se trabaja confirme a ellas. Además, proporciona la asesoría y formación necesaria al proprietario del producto y al equipo. En el proyecto estará representado por el autor, por lo que éste representará tanto al equipo de desarrollo como al scrum master.

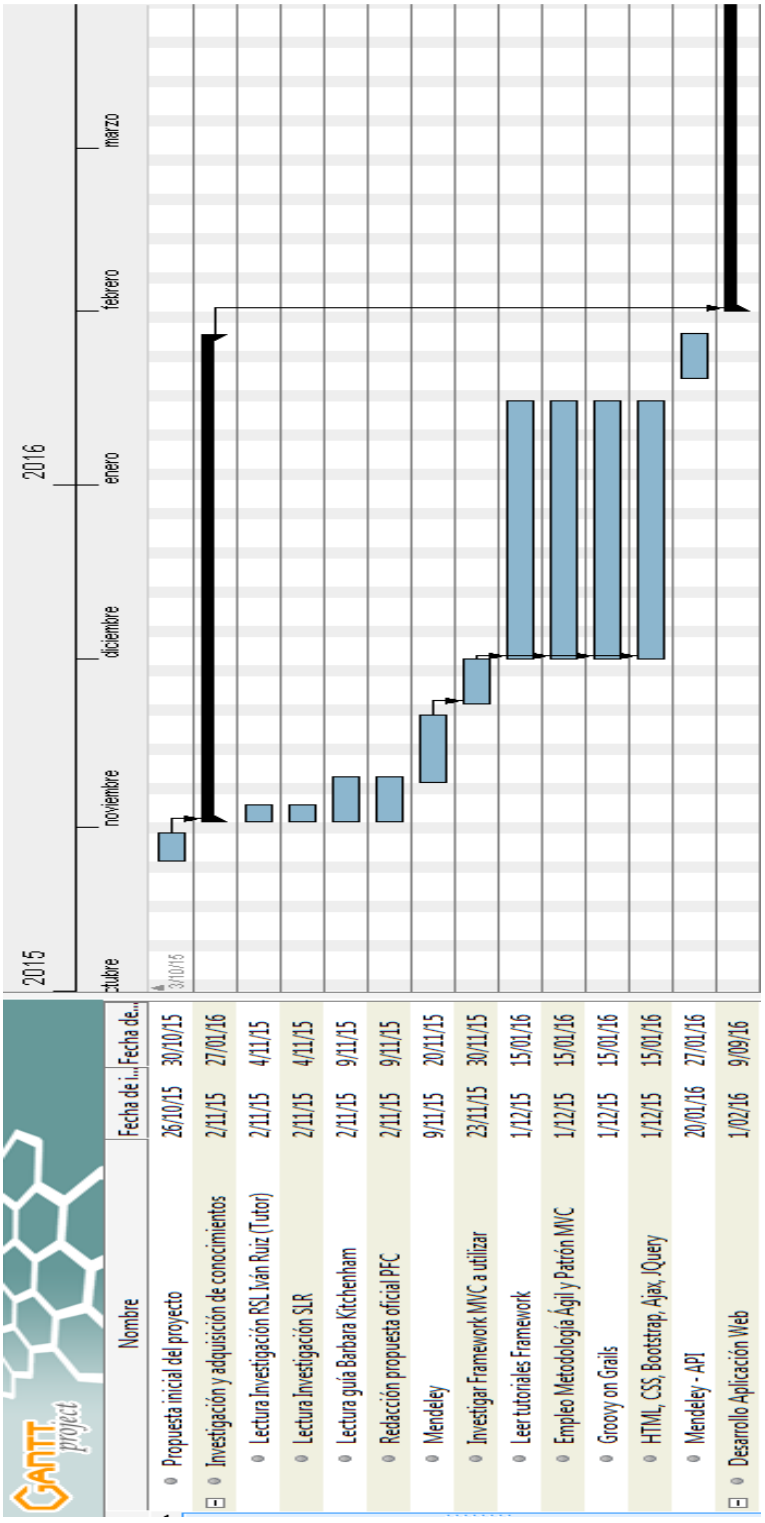

Figura 2.2: Diagrama de Gantt con la propuesta inicial y la adquisición de conocimientos.

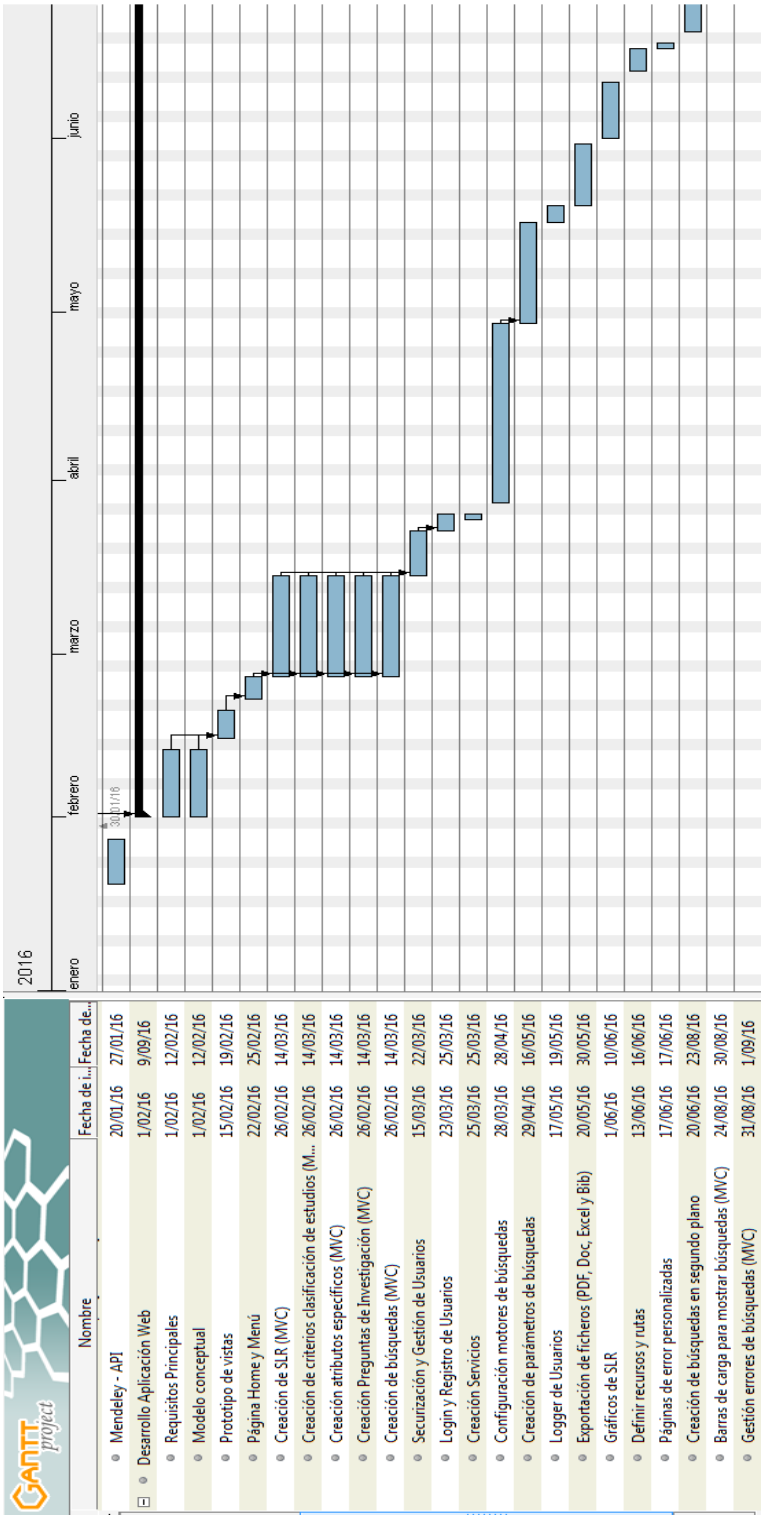

Figura 2.3: Diagrama de Gantt con el Desarrollo de la Aplicación Web.

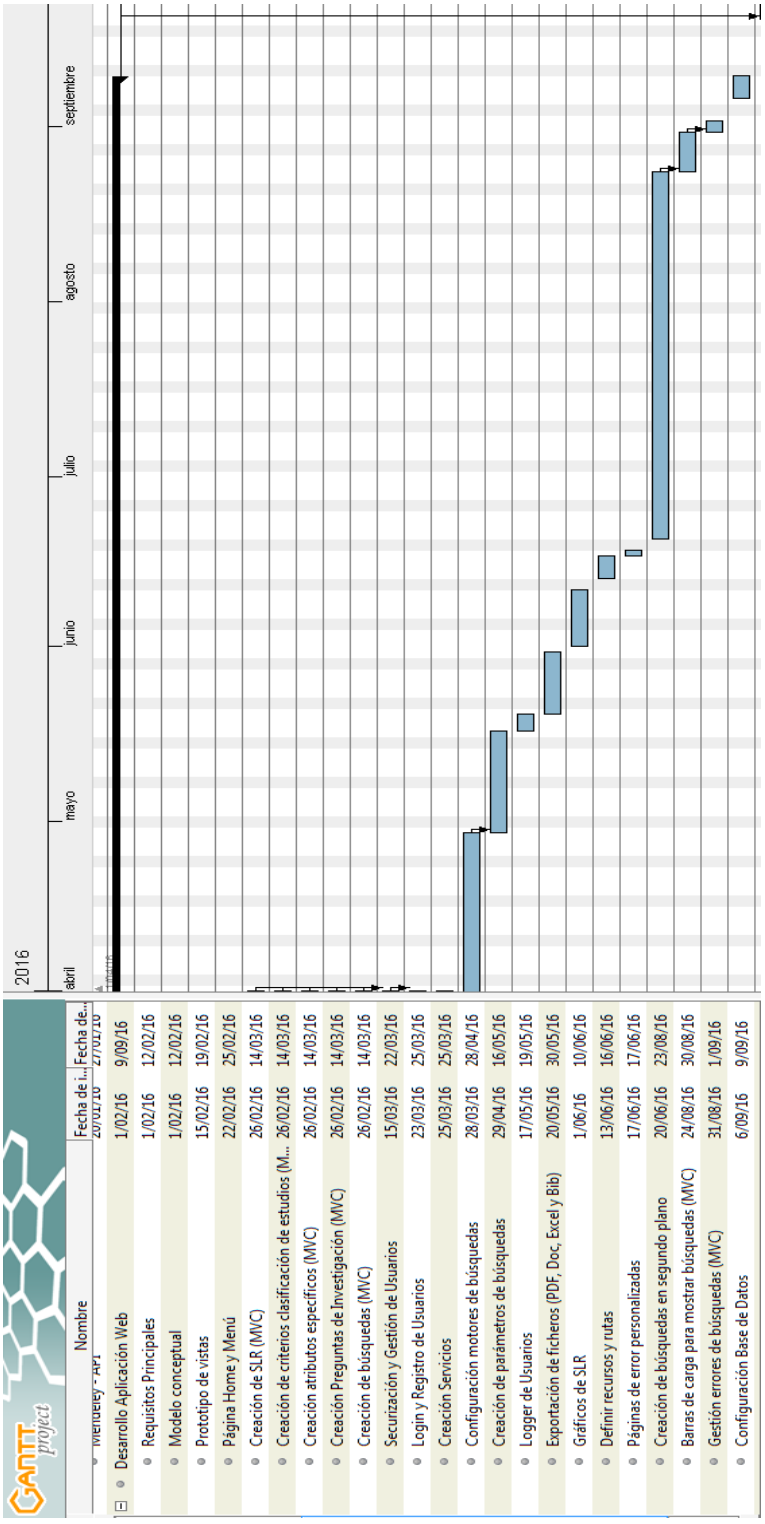

Figura 2.4: Diagrama de Gantt con el Desarrollo de la Aplicación Web (II).

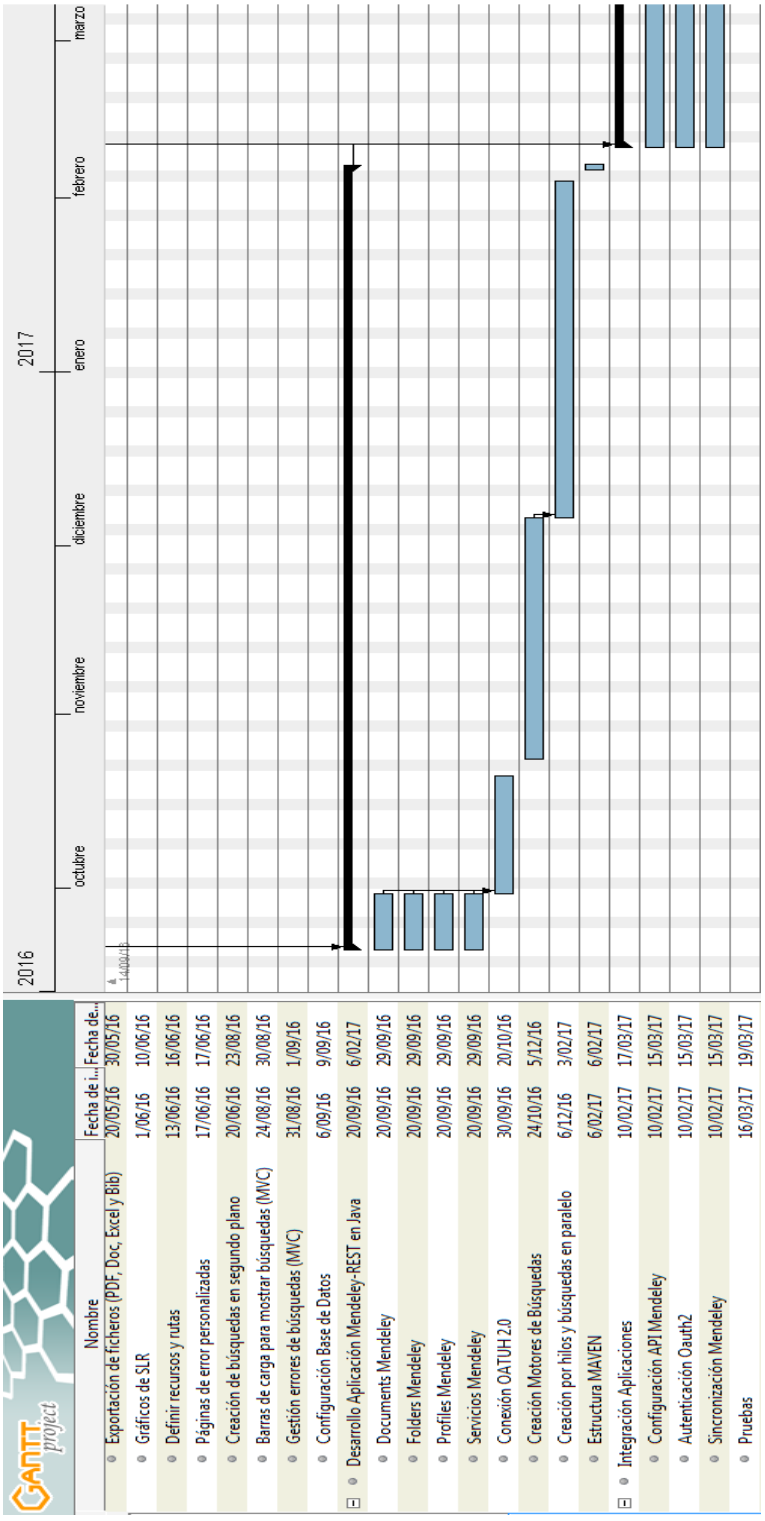

Figura 2.5: Diagrama de Gantt con el Desarroll de la Aplicación Mendeley-REST en Java.

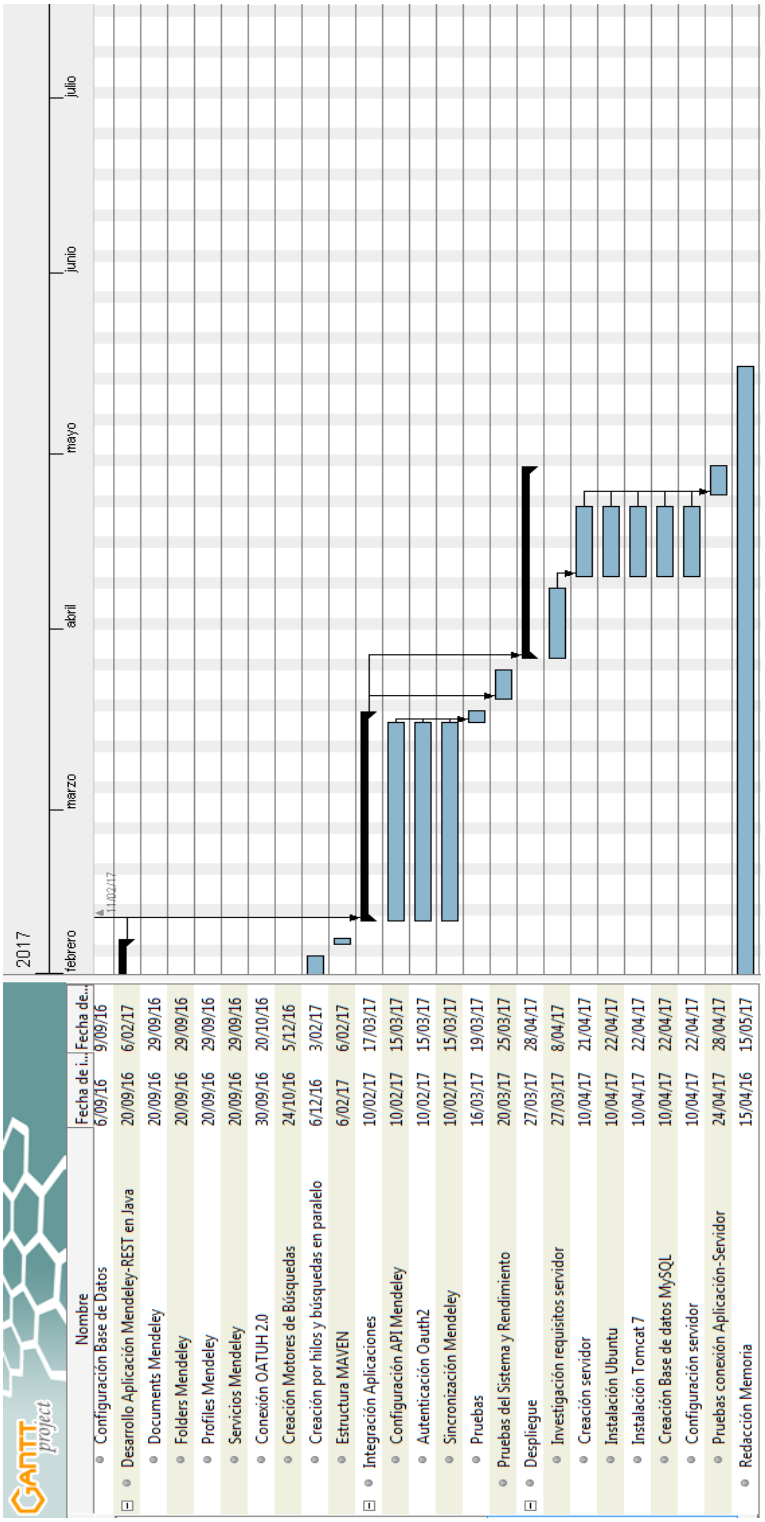

Figura 2.6: Diagrama de Gantt con la Integración de Aplicaciones, Despliegue y Redacción de la memoria.

2.3.2. Recursos

En este apartado se van a listar todos los recursos inventariables de hardware y software, así como las herramientas utilizadas y los lenguajes de programación.

En primer lugar se listan los recursos de hardware.

- Equipo donde se ha realizado el proyecto: Toshiba Satellite L750/L755, Intel ®Core ™i5-2410M CPU @ 2.30GHz x 2, 4GB RAM
- Servidor donde se ha desplegado el proyecto: 1 vCore CPU, 2048MB RAM, 40 GB SSD Storage, 2000 GB Bandwitch

A continuación se listan los recursos de software

- Equipo donde se ha realizado el proyecto:
  - SO: Windows 7 64 bits
  - API Externa: Google Charts y Mendeley
- Servidor donde se ha desplegado el proyecto:
  - SO: Ubuntu 16.04 7 64 bits
  - Contenedor Web: Tomcat 7 64 bits
  - Base de Datos: MySql

A continuación se listan las herramientas empleadas.

- IDE: Grails Tool Suite
- Control de versiones: Subversion
- Forja: Assembla
- SGBD: Mysql
- Diseño de diagramas: DIA
- Diseño de Mockups: Balsamiq Mockups
- Navegadores empleados: Firefox y Google Chrome
- Memoria y presentación: LaTeX
- Procesador de texto: Microsoft Word 2010
- Gestión Hojas de cálculo: Microsoft Excel 2010
- Vision de documentos: Adobe Acrobat Reader

Para finalizar, se listan los lenguajes de programación utilizados:

- Groovy
- Java
- JavaScript
- Ajax
- JQuery
- CSS3
- Bootstrap
- HTML

2.4. Costes

Para poder realizar una estimación de los costes del proyecto debemos tener en cuenta tanto los recursos humanos como los recursos en material empleados. Los costes indirectos como papel, bolígrafo, electricidad o conexión a Internet son los que denominaremos "Costes indirectos". El porcentaje de su valor puede oscilar entre el 10 % y el 20 % del gasto del personal, tomaremos el gasto medio de dicho intervalo, es decir, un 15 %.

De los recursos hardware debemos tener en cuenta tanto el equipo empleado para el desarrollo. Normalmente, los ordenadores suelen tener un periodo de amortización entre los 2 y 4 años. Al igual que antes, tomaremos la media de este periodo, 3 años. Si el equipo costó 750 euros, se amortizan unos 250 euros al año y como el tiempo de desarrollo del proyecto es de entre 2 y 3 años a tiempo parcial, tomaremos 2,5 años como periodo medio y el coste final será de 625 euros. El coste del servidor son de unos 10 euros al mes, sólamente ha estado desplegado en el servidor alrededor de 3 meses, por lo que su gasto alcanza los 30 euros.

Con respecto a los recursos del software, no tendrá ningún gasto, puesto que el sistema operativo Windows 7 viene incluido en los gastos del equipo con el que se ha desarrollado el proyecto, así como los paquetes informáticos (hojas de cálculo y editor de texto). Además, el servidor tiene instalado Ubuntu y tampoco supone ningún gasto económico puesto que viene en el gasto mensual del servidor.

Para el cálculo de costes de personal se ha consultado las tablas salariales de la UCA para el personal técnico de apoyo contratado laboral [CCOO, 2010]. El coste es de unos 20.298,30 euros anuales, lo equivale a 1.353,22 euros mensuales. Como el tiempo dedicado en este tiempo ha sido lo correspondiente a media jornada, tomaremos como un gasto mensual de 676.61 euros. De esta manera, el coste anual será de unos 8119.32 euros y alcanzaremos unos gastos anuales de 20298 euros.

En la tabla 2.2 desglosamos los costes mensuales y totales del proyecto:

Unidades	Descripción	Coste Unitario	Coste total
1	Ingeniero Técnico Informático	8119.32 €/año	20298 €
1	Ordenador personal	286.67 €/año	625 €
1	Alquiler Host	10 €/mes	30 €
1	Costes indirectos	1217 €/año	3044 €
Total			23967 €

Tabla 2.2: Costes del desarrollo

2.5. Riesgos

En esta sección vamos a describir los posibles riesgos del proyecto ordenados de mayor a menor prioridad, indicando su posible impacto (efecto que la ocurrencia del citado riesgo tendría en el desarrollo del proyecto) y la probabilidad de ocurrencia. Además, indicamos el plan necesario para reducir los efectos del riesgo una vez se haya materializado o disminuir que este ocurra. Esta información viene recogida en la tabla 2.3.

2.6. Aseguramiento de calidad

Para asegurar el cumplimiento de la calidad de este se contempla los siguientes aspectos:

- Realización de controles o pruebas para asegurar su calidad.
- Tomar medidas de actuación relacionadas con el control de calidad.

Se realizará un análisis para cada uno de los hitos o tareas, con lo que es necesario un plan de verificación y validación dentro de los mismos para poder asegurar la calidad del producto y el correcto funcionamiento de los mismos. Estas comprobaciones se realizarán a lo largo del proyecto y así comprobar que todo está correcto y se cumple los requisitos establecidos.

Riesgo	**Prob.**	**Magnitud**	**Plan a realizar**
Tiempo infraestimado para el desarrollo de las tareas	30 %	5-8 semanas	Se comunica al tutor del proyecto del posible retraso, modificando la planificación temporal del proyecto o bien el plazo de entrega para poder finalizarlo.
Desarrollo de métodos y funciones software incorrectos	30 %	3-4 semanas	Se comprueba la calidad del código y su validez, para así evitar el denominado **DRY** (*Don't Repeat yourself*) y estudiar su re-utilización una vez corregido.
Cambios en API de Mendeley	30 %	3 semanas	Comunicar al tutor del proyecto del posible retraso y realizar una rápida investigación de los cambios producidos en la API de Mendeley, así como el desarrollo de la nueva implementación de la misma.
Cambios en API de motores de búsquedas de referencias bibliográficas	30 %	3 semanas	Comunicar al tutor del proyecto del posible retraso y realizar una rápida investigación de los cambios producidos en los motores de búsquedas, como el desarrollo de la nueva implementación de los mismos.
Elección incorrecta en las herramientas empleadas de desarrollo	15 %	2-3 semanas	Comunicar al tutor del proyecto del posible retraso y realizar una rápida investigación de las nuevas herramientas a emplear, así como de la búsqueda y lectura de documentación.
Paro del proyecto	20 %	1-2 semanas	Revisar la planificación y ajustarla, así como añadir horas extras una vez que se retome el desarrollo.
Problemas en los tests	30 %	1-2 semanas	Se analizan los tests y se planteará la nueva revisión de éstos y, en el caso correspondiente, crear unos nuevos.

Tabla 2.3: Lista de riesgos del proyecto

Parte II

Desarrollo

En esta parte se debe describir el desarrollo del proyecto siguiendo la metodología empleada. Sus capítulos no deben ser una descripción exhaustiva de todos los documentos, diagramas, código fuente y, en general, entregables generados, sino más bien una explicación resumida del desarrollo, estructurada según las etapas principales del proceso de ingeniería. Deben seleccionarse aquellos diagramas, fragmentos de código y secciones de los entregables que sean más significativos para dicha explicación. La totalidad de los entregables resultado del proyecto se ubicarán en los anexos y/o en el material en CD/DVD que acompañe al proyecto.

Capítulo 3

Requisitos del Sistema

En esta sección se detalla la situación actual de la organización y las necesidades de la misma, que originan el desarrollo o mejora de un sistema informático. Luego se presentan los objetivos y el catálogo de requisitos del nuevo sistema. Finalmente se describen las diferentes alternativas tecnológicas y el análisis de la brecha entre los requisitos planteados y la solución base seleccionada, si aplica.

3.1. Situación actual

Esta sección debe contener información sobre la situación actual de la organización para la que se va a desarrollar el sistema software.

3.1.1. Procesos de Negocio

Esta sección debe contener información sobre los modelos de procesos de negocio actuales, que suelen ser la base de los modelos de procesos de negocio a implantar.

3.1.2. Entorno Tecnológico

Esta sección debe contener información general sobre el entorno tecnológico en la organización del cliente antes del comienzo del desarrollo del sistema software, incluyendo hardware, redes, software, etc.

3.1.3. Fortalezas y Debilidades

Esta sección debe contener información sobre los aspectos positivos y negativos del negocio actual de la organización para la que se va a desarrollar el sistema software.

3.2. Necesidades de Negocio

Esta sección debe contener información sobre los objetivos de negocio de clientes y usuarios, incluyendo los modelos de procesos de negocio a implantar.

3.2.1. Objetivos de Negocio

Esta sección debe contener los objetivos de negocio que se esperan alcanzar cuando el sistema software a desarrollar esté en producción.

3.2.2. Procesos de Negocio

Esta sección, debe contener los modelos de procesos de negocio a implantar, que normalmente son los modelos de procesos de negocio actuales con ciertas mejoras.

3.3. Objetivos del Sistema

Esta sección debe contener la especificación de los objetivos o requisitos generales del sistema.

3.4. Catálogo de Requisitos

Esta sección debe contener la descripción del conjunto de requisitos específicos del sistema a desarrollar para satisfacer las necesidades de negocio del cliente.

3.4.1. Requisitos funcionales

Descripción completa de la funcionalidad que ofrece el sistema.

3.4.2. Requisitos no funcionales

Descripción de otros requisitos (relacionados con la calidad del software) que el sistema deberá satisfacer: portabilidad, seguridad, estándares de obligado cumplimiento, accesibilidad, usabilidad, etc.

3.4.3. Reglas de negocio

En el desarrollo del sistema, hay que tener en cuenta las denominadas reglas de negocio, es decir, el conjunto de restricciones, normas o políticas de la organización que deben ser respetadas por el sistema, las cuales suelen ser cambiantes.

3.4.4. Requisitos de información

En esta sección se describen los requisitos de gestión de información (datos) que el sistema debe gestionar.

3.5. Alternativas de Solución

En esta sección, se debe ofrecer un estudio del arte de las diferentes alternativas tecnológicas que permitan satisfacer los requerimientos del sistema, para luego seleccionar (si procede) la herramienta o conjunto de herramientas que utilizaremos como base para el software a desarrollar.

3.6. Solución Propuesta

Si se ha optado por utilizar un software de base, debemos identificar y medir las diferencias entre lo que proporciona este software y los requisitos definidos para el proyecto.
El resultado de este análisis permitirá identificar cuáles de éstos requisitos ya están solventados total o parcialmente por el sistema base y cuales tendremos que diseñar e implementar la propuesta de solución.

Capítulo 4

Análisis del Sistema

Esta sección cubre el análisis del sistema de información a desarrollar, haciendo uso del lenguaje de modelado UML.

4.1. Modelo Conceptual

A partir de los requisitos de información, se desarrollará un diagrama conceptual de clases UML, identificando las clases, atributos, relaciones, restricciones adicionales y reglas de derivación necesarias.

4.2. Modelo de Casos de Uso

A partir de los requisitos funcionales descritos anteriormente, se emplearan los casos de uso como mecanismo para representar las interacciones entre los actores y el sistema bajo estudio. Para cada caso de uso deberá indicarse los actores implicados, las precondiciones y postcondiciones, los pasos que conforman el escenario principal y el conjunto de posibles escenarios alternativos.

4.2.1. Actores

En este apartado se describirán los diferentes roles que juegan los usuarios que interactúan con el sistema. Los actores pueden ser roles de personas físicas, sistemas externos o incluso el tiempo (eventos temporales).

4.3. Modelo de Comportamiento

A partir de los casos de uso anteriores, se crea el modelo de comportamiento. Para ello, se realizarán los diagramas de secuencia del sistema, donde se identificarán las operaciones o servicios del sistema. Luego, se detallará el contrato de las operaciones identificadas.

4.4. Modelo de Interfaz de Usuario

En esta sección se deberá incluir un prototipo de baja fidelidad o mockup de la interfaz de usuario del sistema. Además, es preciso elaborar un diagrama de navegación, reflejando la secuencia de pantallas a las que tienen acceso los diferentes roles de usuario y la conexión entre éstas.

Capítulo 5

Diseño del Sistema

En esta sección se recoge la arquitectura general del sistema de información, la parametrización del software base (opcional), el diseño físico de datos, el diseño detallado de componentes software y el diseño detallado de la interfaz de usuario.

5.1. Arquitectura del Sistema

En esta sección se define la arquitectura general del sistema de información, especificando la infraestructura tecnológica necesaria para dar soporte al software y la estructura de los componentes que lo forman.

5.1.1. Arquitectura Física

En este apartado, describimos los principales elementos hardware que forman la arquitectura física de nuestro sistema, recogiendo por un lado los componentes del entorno de producción y los componentes de cliente.

Se debe incluir un modelo de despliegue en el cual se describe cómo los elementos software son desplegados en los elementos hardware. También se incluyen las especificaciones y los requisitos del hardware (servidores, etc.), así como de los elementos software (sistemas operativos, servicios, aplicaciones, etc.) necesarios.

5.1.2. Arquitectura Lógica

La arquitectura de diseño especifica la forma en que los artefactos software interactúan entre sí para lograr el comportamiento deseado en el sistema. En esta sección se muestra la comunicación entre el software base seleccionado, los componentes reutilizados y los componentes desarrollados para cumplir los requisitos de la aplicación. También, se recogen los servicios de sistemas externos con los que interactúa nuestro sistema. Se debe incluir un diagrama de componentes que muestre en un alto nivel de abstracción los artefactos que conforman el sistema.

Existen diferentes patrones o estilos arquitectónicos. En los sistemas web de información es común la utilización del patrón Layers (Capas), con el cual estructuramos el sistema en un número apropiado de capas, de forma que todos los componentes de una misma capa trabajan en el mismo nivel de abstracción y los servicios proporcionados por la capa superior utilizan

internamente los servicios proporcionados por la capa inmediatamente inferior. Habitualmente se tienen las siguientes capas:

Capa de presentación (frontend) Este grupo de artefactos software conforman la capa de presentación del sistema, incluyendo tanto los componentes de la vista como los elementos de control de la misma.

Capa de negocio Este grupo de artefactos software conforman la capa de negocio del sistema, incluyendo los elementos del modelo de dominio y los servicios (operaciones del sistema).

Capa de persistencia Este grupo de artefactos software conforman la capa de integración del sistema, incluyendo las clases de abstracción para el acceso a datos (BD o sistema de ficheros) o a sistemas heredados.

Es común que a la capa de negocio y de datos de los sistemas web, se denomine conjuntamente como backend o modelo de la aplicación.
Opcionalmente, podemos disponer de un conjunto de artefactos software que pueden ser usados por elementos de cualquiera de las capas del sistema y que fundamentalmente proporcionan servicios relacionados con requisitos no funcionales (calidad).

5.2. Parametrización del software base

En esta sección, se detallan las modificaciones a realizar sobre el software base, que son requeridas para la correcta construcción del sistema. En esta sección incluiremos las actuaciones necesarias sobre la interfaz de administración del sistema, sobre el código fuente o sobre el modelo de datos.

5.3. Diseño Físico de Datos

En esta sección se define la estructura física de datos que utilizará el sistema, a partir del modelo de conceptual de clases, de manera que teniendo presente los requisitos establecidos para el sistema de información y las particularidades del entorno tecnológico, se consiga un acceso eficiente de los datos. La estructura física se compone de tablas, índices, procedimientos almacenados, secuencias y otros elementos dependientes del SGBD a utilizar.

5.4. Diseño detallado de Componentes

Para cada uno de los módulos funcionales del sistema debemos realizar un diagrama de secuencia, para definir la interacción existente entre las clases de objetos que permitan responder a eventos externos.

5.5. Diseño detallado de la Interfaz de Usuario

En esta sección se detallarán las interfaces entre el sistema y el usuario, incluyendo un prototipo de alta fidelidad con el diseño de la IU. Se definirá el comportamiento de las diferentes pantallas, indicando qué ocurre en los distintos componentes visuales de la interfaz cuando aparecen y qué acciones se disparan cuando el usuario trabaja con ellas.

Capítulo 6

Construcción del Sistema

Este capítulo trata sobre todos los aspectos relacionados con la implementación del sistema en código, haciendo uso de un determinado entorno tecnológico.

6.1. Entorno de Construcción

En esta sección se debe indicar el marco tecnológico utilizado para la construcción del sistema: entorno de desarrollo (IDE), lenguaje de programación, herramientas de ayuda a la construcción y despliegue, control de versiones, repositorio de componentes, integración contínua, etc.

6.2. Código Fuente

Organización del código fuente, describiendo la utilidad de los diferentes ficheros y su distribución en paquetes o directorios. Asimismo, se incluirá algún extracto significativo de código fuente que sea de interés para ilustrar algún algoritmo o funcionalidad específica del sistema.

6.3. Scripts de Base de datos

Organización del código fuente, describiendo la utilidad de los diferentes ficheros y su distribución en paquetes o directorios. Asimismo, se incluirá el script de algún disparador o un procedimiento almacenado, que sea de interés para ilustrar algún aspecto concreto de la gestión de la base de datos.

Capítulo 7

Pruebas del Sistema

En este capítulo se presenta el plan de pruebas del sistema de información, incluyendo los diferentes tipos de pruebas que se han llevado a cabo, ya sean manuales (mediante listas de comprobación) o automatizadas mediante algún software específico de pruebas.

7.1. Estrategia

En esta sección se debe incluir el alcance de las pruebas, hasta donde se pretende llegar con ellas, si se registrarán todas o sólo aquellas de un cierto tipo y cómo se interpretarán y evaluarán los resultados. También, se incluirá el procedimiento a seguir para las pruebas de regresión, esto es, la repetición de ciertas pruebas para comprobar que nuevos cambios que se vayan introduciendo no originen errores en el software ya probado.

7.2. Entorno de Pruebas

Incluir en este apartado los requisitos de los entornos hardware/software donde se ejecutarán las pruebas.

7.3. Roles

Describir en esa sección cuáles serán los perfiles y participantes necesarios para la ejecución de cada uno de los niveles de prueba.

7.4. Niveles de Pruebas

En este sección se documentan los diferentes tipos de pruebas que se han llevado a cabo, ya sean manuales o automatizadas mediante algún software específico de pruebas.

7.4.1. Pruebas Unitarias

Las pruebas unitarias tienen por objetivo localizar errores en cada nuevo artefacto software desarrollado, antes que se produzca la integración con el resto de artefactos del sistema.

7.4.2. Pruebas de Integración

Este tipo de pruebas tienen por objetivo localizar errores en módulos o subsistemas completos, analizando la interacción entre varios artefactos software.

7.4.3. Pruebas de Sistema

En esta actividad se realizan las pruebas de sistema de modo que se asegure que el sistema cumple con todos los requisitos establecidos: funcionales, de almacenamiento, reglas de negocio y no funcionales. Se suelen desarrollar en un entorno específico para pruebas.

Pruebas Funcionales

Con estas pruebas se analiza el buen funcionamiento de la implementación de los flujos normales y alternativos de los distintos casos de uso del sistema.

Pruebas No Funcionales

Estas pruebas pretenden comprobar el funcionamiento del sistema, con respecto a los requisitos no funcionales identificados: eficiencia, seguridad, etc.

7.4.4. Pruebas de Aceptación

El objetivo de estas pruebas es demostrar que el producto está listo para el paso a producción. Suelen ser las mismas pruebas que se realizaron anteriormente pero en el entorno de producción. En estas pruebas, es importante la participación del cliente final.

Parte III

Epílogo

En esta última parte quedarán recogidas las conclusiones y los manuales necesarios para el manejo de la aplicación resultado del desarrollo. Si se ha realizado algún tipo de evaluación de la solución proporcionada, más allá de las pruebas del sistema, también deberá venir recogida en un capítulo separado dentro de esta parte. Pueden consultarse diversos tipos de evaluaciones sobre sistemas de información en [Hevner et al., 2004]: casos de estudio, análisis estático, análisis dinámico, simulación, experimento controlado, etc.

Capítulo 8

Manual de implantación y explotación

Las instrucciones de instalación y explotación del sistema se detallan a continuación. Este manual resulta de aplicación en aquellos casos en que se requiere realizar la instalación del software objetivo de este proyecto sobre algún entorno de servidor o cuando su instalación no sea trivial.

8.1. Introducción

Resumen de los principales objetivos, ámbito, características y alcance del software desarrollado.

8.2. Requisitos previos

Requisitos hardware y software para la correcta instalación del sistema.

8.3. Inventario de componentes

Lista de los componentes hardware y software que se incluyen en la versión del producto.

8.4. Procedimientos de instalación

Procedimientos de instalación y configuración de cada componente hardware y software (base y desarrollado) para asegurar la correcta instalación y explotación del sistema, así como aquellos procedimientos necesarios de migración/carga de datos.

8.5. Pruebas de implantación

Descripción de las pruebas a realizar después de la instalación del sistema.

8.6. Procedimientos de operación y nivel de servicio

Procedimientos necesarios para asegurar el correcto funcionamiento, rendimiento, disponibilidad y seguridad del sistema: back-ups, chequeo de logs, etc. También, es preciso indicar claramente aquellas actuaciones precisas necesarias para el mantenimiento preventivo del sistema y así prevenir posibles fallos en el mismo.

Capítulo 9

Manual de usuario

Las instrucciones de uso del software se detallan a continuación. Este manual se dirige al usuario final del software objetivo de este proyecto.

9.1. Introducción

Resumen de los principales objetivos, ámbito, características y alcance del software desarrollado.

9.2. Instalación

Se detallarán los pasos necesarios para la obtención e instalación del software, así como los requisitos previos de hardware y software.

9.3. Uso del sistema

Describir todos los aspectos necesarios para una utilización efectiva y eficiente de las características del sistema por parte de los usuarios.

Capítulo 10

Manual del desarrollador

A continuación se recogen las instrucciones necesarias para evolucionar el software. Este manual está dirigido a los desarrolladores que pretenden extender o modificar el código fuente, con el fin de incorporar nuevas funcionalidades o modificar las ya existentes. A lo largo de este capítulo se deberán hacer referencias explicitas a aquellos epígrafes de los capítulos de Diseño, Construcción y Pruebas del Sistema que resulten de interés.

10.1. Introducción

Resumen de los principales objetivos, ámbito, características y alcance del software desarrollado.

10.2. Preparación del entorno de trabajo

Descripción de los requisitos (hardware y software) previos. Datos de interés relativos al control de versiones del software. Detalles sobre la instalación en local del entorno de desarrollo y, si fuesen necesarios, de otros componentes como bases de datos, servidores de aplicaciones, etc.

10.3. Consideraciones generales sobre el desarrollo

Aspectos importantes a tener en cuenta a la hora de modificar y extender el código fuente, guías de estilo, etc. Asimismo, se detallarán las directrices que sean de aplicación a la hora de realizar pruebas sobre las nuevas mejoras introducidas.

10.4. Instrucciones para construcción y despliegue

Secuencia de pasos requeridos para llevar a cabo la compilación del código fuente y así poder construir y depurar el software sobre una máquina de desarrollo.

Capítulo 11

Conclusiones

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

11.1. Objetivos alcanzados

Este apartado debe resumir los objetivos generales y específicos alcanzados, relacionándolos con todo lo descrito en el capítulo de introducción.

11.2. Lecciones aprendidas

A continuación, se detallan las buenas prácticas adquiridas, tanto tecnológicas como procedimentales, así como cualquier otro aspecto de interés.
Resumir cuantitativamente el tiempo y esfuerzo dedicados al proyecto a lo largo de su desarrollo que escribir un sencillo 'he trabajado mucho en este proyecto'.

11.3. Trabajo futuro

En esta sección, se presentan las diversas áreas u oportunidades de mejora detectadas durante el desarrollo del proyecto y que podrán ser abarcadas en futuras versiones del software.

Los elementos aquí descritos deben estar en relación con lo relatado en el apartado de objetivos y alcance del proyecto descritos en la introducción.

Bibliografía

[Bok, 2016] Bok, S. M. (2016). Scrum Manager - Guía de formación. http://www.scrummanager.net/bok/index.php?title=Scrum_Manager_BoK.

[CCOO, 2010] CCOO (2010). Tablas salariales 2010 IV Convenio Colectivo. http://www.uca.es/sindicato/ccoo/documentos/tabla-salarial-pas-laboral-2010.pdf.

[Hevner et al., 2004] Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105.

[Kitchenham, 2007] Kitchenham, B. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf.

[Project, 2015] Project, G. (2015). Gantt Project: free desktop project management app. http://www.ganttproject.biz/.

[Rube, 2013] Rube, I. R. (2013). Revisiones Sistemáticas de la Literatura en Ingeniería del Software: Un ejemplo práctico. http://www.scrummanager.net/bok/index.php?title=Scrum_Manager_BoK.

Información sobre Licencia

Incluir aquí la información relativa a la licencia seleccionada para la documentación y software del presente proyecto.