Universidad
Carlos III de Madrid

GESTOR DE PROYECTOS CON METODOLOGÍA SCRUM

Trabajo Fin de Carrera

AUTOR: Oscar Rodríguez Campos

TUTOR: David Palomar Delgado

9-6-2014

Título: Gestión de proyectos con metodologías ágiles

Autor: Oscar Rodriguez Campos

Tutor: David Palomar Delgado

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Trabajo Fin de Carrera el día ___ de _____ de 2014 en Colmenarejo, en la escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Quiero dedicar en primer lugar y de forma muy especial este proyecto a mis padres, Santi y Conchi, porque si no fuera por ellos no estaría hoy aquí escribiendo estas líneas, porque su esfuerzo y dedicación han hecho que fuera mejor persona y que haya intentado mejorar día a día, no solo a nivel personal, sino también profesional. Esto es para vosotros, para que veáis que todo esfuerzo tiene su recompensa.

A mi hermana Marta, porque es la persona que mas ánimos me ha dado para hacer este proyecto. También es la persona que en los momentos más difíciles más me ha ayudado. Además, es la persona que más me ha aconsejado en el tema de presentación de las pantallas. Una parte importante del diseño se ha conseguido gracias a sus consejos.

Al resto de mi familia, que aunque no os pueda nombrar uno a uno siempre que me ven me preguntan qué tal me va con la carrera y me dan ánimos para continuar.

Quiero agradecer también a toda esa gente maravillosa que he conocido a lo largo de estos años, porque de esta experiencia no me llevo solo conocimientos y un título, sino que también me llevo muchas personas increíbles, muchos amigos que sé que estarán ahí siempre. Simplemente, gracias por todo.

También quiero agradecer al esfuerzo realizado por mi mujer, que en estos años me ha ayudado a continuar y a no dejarlo atrás. Me ha animado y ha tenido una gran paciencia conmigo.

Gracias a mi tutor, David, por la confianza depositada en mí para la realización de este proyecto, por su paciencia y comprensión con tanto retraso y cambio de fechas, por todo el esfuerzo y dedicación que ha puesto para que el proyecto saliese adelante de forma satisfactoria.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1. Introducción y objetivos

1.1. Introducción

En el actual mercado existen multitud de aplicaciones de gestión de proyectos para metodologías tradicionales basadas en ciclos de vida en cascada. En cambio, si se busca una aplicación para gestionar proyectos basados en metodologías ágiles la variedad no es tan amplia. Por ello se ha considerado de utilidad el crear una aplicación Web para gestionar este tipo de proyectos es una gran iniciativa.

Esta aplicación podrá ser colgada en un servidor para que su uso no dependa del ordenador con el que se accede sino con los credenciales de la persona que ingresa.

Con esta aplicación se podrá tener una visión clara de lo que el proyecto requiere (Product Backlog) y de su planificación y progreso (Sprints). Por ello es una herramienta válida, tanto para el análisis del proyecto con para la gestión de su desarrollo.

Para ello, se procederá a estudiar cómo trabajan las metodologías ágiles en general y Scrum en particular. Scrum es una metodología que divide las partes de un proyecto en partes tangibles y entregables (Sprints). Gracias a estas entregas parciales se podrá ir mejorando el producto hasta su fin. Al dividir el proyecto en partes prácticamente independientes, se podrán añadir los módulos que hagan falta sin apenas problemas de acoplamiento.

Para realizar el proyecto hay que proceder con una formación en Java y en JavaServer Faces, que es el lenguaje y framework que se va a utilizar. Después, se analizará la aplicación. De este análisis obtendremos las historias de usuario y su descomposición en tareas. A partir de ahí comenzaremos con su implementación y pruebas unitarias. La implementación está basada en el patrón Modelo-Vista-Controlador. En este documento contará con un ejemplo de cada tipo.

1.2. Objetivos

El objetivo fundamental de este proyecto es crear una aplicación Web (conjunto de páginas Web que interactúan entre sí, con el usuario y con diversos recursos en un servidor Web) de gestión de proyectos para la metodología SCRUM. Se busca realizar una aplicación consistente y dinámica utilizando tecnologías apropiadas y modernas, para lo cual se hará un estudio previo de las tecnologías existentes en el mercado.

Al finalizar la aplicación, un equipo podrá ser capaz de gestionar un proyecto desde la misma, desde la creación de las historias hasta la gestión del tiempo dedicado a ellas. La aplicación debe ser visual ya que es una de las características principales de Scrum. Por ello el ScrumMaster deberá tener acceso a un panel con todas las historias, otro con el progreso de los Sprints, además de gráficas para ver la evolución de los mismos.

En base a este objetivo se proponen los siguientes objetivos parciales:

- Estudio sobre las tecnologías basadas en Modelo-Vista-Controlador. Se realizará un pequeño estudio sobre diferentes frameworks que aplican el patrón MVC.
- Comparativa de las tecnologías y selección de una de ellas para realizar el Proyecto Fin de carrera. Se comparan los diferentes frameworks y se elije uno para implementar la aplicación.
- Estudio de la metodología Scrum. Se analizarán sus ventajas e inconvenientes frente a las metodologías tradicionales. Se estudiarán los roles de las partes implicadas, la división de tareas (casos de uso) y la manera de comprobar el progreso.
- Realización del análisis del proyecto a analizar. Se analizan los distintos requisitos que se quieren plasmar en la aplicación. Realizamos un análisis de los casos de uso y su descomposición en tareas, para después planificarlos en sus Sprints correspondientes. Se definen las bases de datos.
- Implementación del proyecto. Construyo la aplicación para que cumpla todos los casos de uso analizados anteriormente. Una vez terminado se realizarán las pruebas de integración.

- Realización del presupuesto del proyecto. En base al tiempo y recursos utilizados, se obtendrá un presupuesto del proyecto.

1.3. Fases de desarrollo

En este punto se comentarán las diferentes fases de desarrollo llevadas a cabo para la correcta elaboración del mismo.

- **Estudio de la tecnología Modelo-Vista-Controlador**: en este punto se analizarán el patrón MVC. Después se elijarán 4 frameworks basados en dicho patrón y se estudiarán en profundidad. Por último se compararán para establecer cuál es el óptimo para este proyecto.
- **Documentación sobre Scrum**: la documentación la metodología que vamos a utilizar es un paso importante para conseguir que la aplicación Web sea lo más eficiente posible. Para ello, se estudiarán libros y manuales además de alguna aplicación que ya existe.
- **Estudio del lenguaje de programación y del framework que se va a utilizar**: en este punto se aprenderá a programar en JAVA utilizando las librerías de JavaServer Faces. Habrá que estudiar en profundidad el lenguaje de programación JAVA así como las distintas librerías que serán útiles a lo largo de la implementación. También se deben conseguir los conocimientos suficientes para la configuración del PC que se va a utilizar en la implementación.
- **Análisis**: en esta fase se estudiarán los distintos casos de uso que se le quiere dar a la aplicación. Aunque esta fase se realizará antes de comenzar el diseño y la implementación, se podrá completar según avance el proyecto.
- **Diseño**: este punto se tratará de resolver el problema planteado en el análisis especificando los componentes del sistema, así como las relaciones entre ellos y la arquitectura utilizada.
- **Implementación**: una vez se tenga el análisis y el diseño de la aplicación, se procede a codificarla según el lenguaje y framework elegidos anteriormente. En esta fase se realizan también las pruebas unitarias.
- **Pruebas de integración**: una vez finalizada la implementación y las pruebas unitarias, se realizarán las pruebas de integración de la aplicación completa. En esta fase se debe comprobar que están todos los casos de uso que se describieron en el Análisis.

- **Documentación**: una vez terminada la aplicación se procede a generar la documentación. A la documentación del proyecto hay que añadir el manual de usuario que se ha realizado para la perfecta comprensión de la aplicación.

1.4. Medios empleados

A continuación describiremos brevemente los medios empleados, tanto hardware como software, para la realización del presente proyecto.

1.4.1. Elementos Hardware

Los elementos hardware utilizados son los detallados a continuación:

- Ordenador. Se ha utilizado un ordenador portátil tanto para el desarrollo de la aplicación, como para la elaboración de la documentación relacionada. El modelo utilizado ha sido un HP Pavilion dv6 con procesador Intel Core i5, 4 GB de memoria RAM y sistema operativo Windows 7 Home Premium.

1.4.2. Elementos Software

Los elementos software utilizados para el desarrollo del proyecto son los siguientes:

- **Eclipse Galileo**: es el entorno de programación que se ha utilizado. Este entorno es útil para la programación de aplicaciones Web en JAVA, lenguaje que se ha utilizado para este proyecto.
- **MySQL GUI v8.4**: base de datos que se ha utilizado. El software MySQL proporciona un servidor de base de datos SQL.
- **Tomcat v6.0**: es el servidor utilizado. Apache Tomcat es una implementación de código abierto de software de las tecnologías Java Servlet y JavaServer Pages.
- **Mojarra Project v1.2**: librerías con el framework de JSF con el que se va a implementar la aplicación.
- **Richfaces v.3.3.1**: librerías utilizadas para integrar fácilmente capacidades de Ajax en la aplicación JSF.
- **JFreeChart v1.0.16**: librería para la creación de gráficas.
- **Javamail**: librería que permite el envío de e-mails desde la aplicación.

1.5. Estructura de la memoria

El presente documento ha sido organizado en diferentes capítulos, de forma que en cada uno de ellos tratemos un tema específico, haciendo que la memoria tenga una mejor organización y sea más fácil de leer y comprender. A continuación se explicará brevemente el contenido de cada uno de estos capítulos.

En el **primer capítulo** (Introducción y objetivos), se expondrá una introducción en la que se mostrará la motivación por la que se eligió este proyecto, los objetivos que se quieren conseguir una vez la aplicación esté implementada, las diferentes fases del proyecto y los medios que han sido necesarios para la elaboración del mismo. De esta forma, conoceremos de forma breve y concisa en qué consiste el proyecto.

En el **segundo capítulo** (Estudio sobre las tecnologías MVC), realizaremos un estudio sobre las tecnologías basadas en Modelo-Vista-Controlador. Esta tecnología, divide la estructura del programa en 3 partes. La parte del Modelo que se encarga de gestionar las bases de datos, la del Controlador, que implementa la funcionalidad del proyecto y la de la Vista, que son las interfaces de usuario.

En el **tercer capítulo** (Comparativa de tecnologías), se estudian 4 frameworks distintos que se basan en MVC. Las 4 tecnologías que se comparan son: Structs, Tapestry, SpringMVC y JavaServer Faces. Mediante una serie de tablas comparativas se llega a la conclusión que el mejor para este caso es JavaServer Faces.

En el **cuarto capítulo** (Estudio de la metodología SCRUM), se estudia en profundidad la metodología. Scrum es una metodología ágil que está orientada a la usabilidad. Trata de dividir la aplicación que se desea realizar en partes distintas e independientes para así poder modificar las partes con mucha facilidad. Además, propone entregas parciales del proyecto al usuario final de manera que se puedan corregir cosas a tiempo.

En el **quinto capítulo** (análisis), se expone el análisis realizado para el proyecto. El análisis está realizado según la metodología Scrum. En el análisis, sacamos las historias y posteriormente las descomponemos en las tareas correspondientes. Por último vamos planificando los Sprints.

En el **sexto capítulo** (Implementación), se realizan los comentarios sobre la implementación del proyecto. Al ser un proyecto basado en MVC se pondrá un ejemplo de Modelo, otro de Vista y otro de Controlador.

El **octavo capítulo** (Conclusiones) se expondrá las conclusiones alcanzadas una vez finalizado el proyecto. Veremos si los objetivos indicados al inicio de este documento han sido cumplidos y las impresiones finales obtenidas tras el desarrollo.

En el **noveno capítulo**, estudiaremos las posibles mejoras que podrían implementarse en el futuro, nuevas funcionalidades, mejoras o evolución del proyecto.

El **noveno capítulo** (Presupuesto), se creará un presupuesto del proyecto.

Para finalizar, se incluirán dos apartados. Por un lado, dispondremos de un glosario de términos, con el fin de que el lector pueda consultarlo en caso de no conocer alguno de los acrónimos utilizados durante el documento. Por otro lado, se incluye una bibliografía con enlaces de interés sobre el proyecto.

2. Estudio sobre las tecnologías basadas en Modelo-Vista-Controlador

2.1. Patrón Modelo-Vista-Controlador

Definición

Modelo-Vista-Controlador (MVC) es un patrón de arquitectura del software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la interfaz de usuario y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Bases de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Historia

El patrón fue descrito por primera vez en 1979 por Trygve Reenskaug, entonces trabajando en Smalltalk en laboratorios de investigación de Xerox. La implementación original está descrita a fondo en Programación de Aplicaciones en Smalltalk-80™ (Goldberg & Robson).

Partes y responsabilidades de cada parte (Descripción)**:**

1. El **modelo** es responsable de:
    - Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
    - Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: “Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor”.
    - Lleva un registro de las vistas y controladores del sistema.
    - Si estamos ante un modelo activo, notificará a las vistas los cambios que

en los datos pueda producir un agente externo (por ejemplo, un fichero batch que actualiza los datos, un temporizador que desencadena una inserción, etc.).

2. El **controlador** es responsable de:
    - Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
    - Contiene reglas de gestión de eventos, de tipo "Si evento Z, entonces acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".
3. Las **vistas** son responsables de:
    - Recibir datos del modelo y mostrarlos al usuario.
    - Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
    - Pueden dar el servicio de "Actualización( )", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

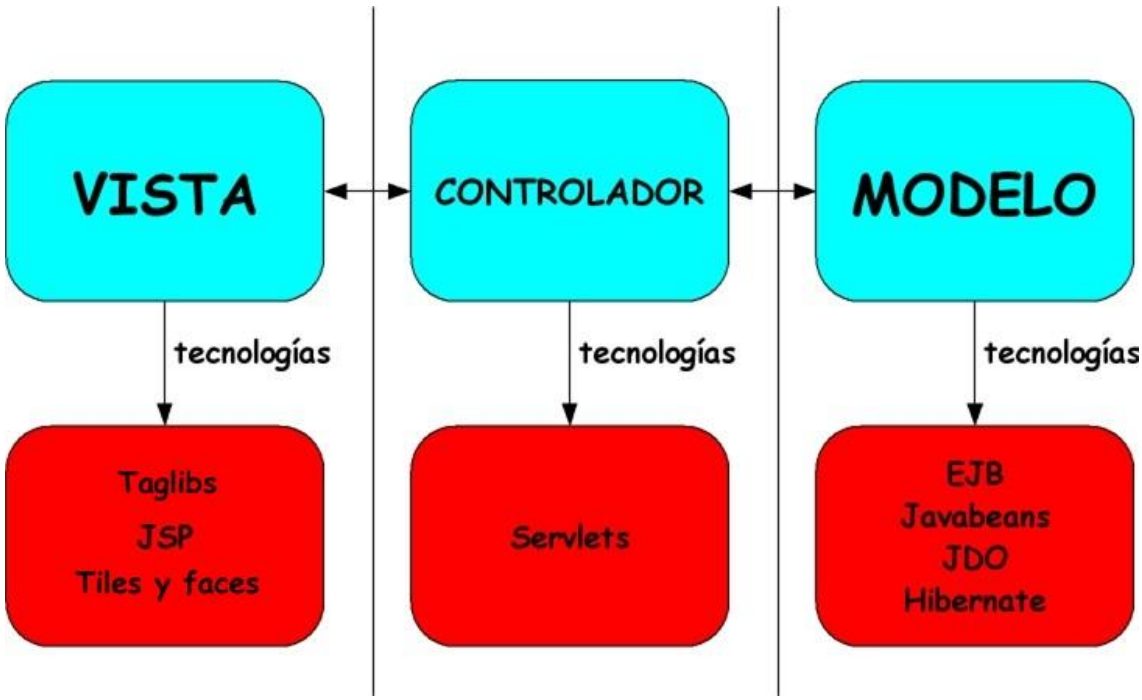


Ilustración 1: División del Modelo-Vista-Controlador

Breve descripción del funcionamiento del MVC

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

3. El controlador accede al modelo, actualizando, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador[1] puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en si mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. *Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista*.

[1] El patrón observador (observer) desacopla los objetos mediante un sistema de notificaciones, mientras que el patrón comando (command) encapsula las operaciones como objetos

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente[2].

[2] Otro escenario que se puede dar es que el controlador genere vistas sin modelo, por ejemplo en los casos de un asistente wizard, que sólo al terminar el asistente, se recogen los datos que pasarán al modelo, pero hasta ese momento el controlador lo único que hace es pasar de una vista a otra.

2.2. Tecnologías del Modelo-Vista-Controlador

Algunos de los frameworks que trabajan sobre el Modelo-Vista-Controlador son: STRUTS, TAPESTRY, JSF y SPRINGMVC.

STRUTS: Apache Struts es un framework de código abierto para la creación de aplicaciones Web en Java. Las aplicaciones Web se diferencian de los sitios Web convencionales en que las aplicaciones Web pueden crear una respuesta dinámica, mientras que la mayoría de los demás sitios Web sólo manejan páginas estáticas. Una aplicación Web puede interactuar con bases de datos y con lógicas de negocio para personalizar las respuestas.

Ilustración 2: Struts

TAPESTRY: Tapestry es un framework para el desarrollo de aplicaciones Web basado en componentes y escrito en Java. Tapestry no usa JSPs ni ningún otro motor de plantillas sino que define un modelo de componentes que pueden ser renderizados para construir la interfaz visual de la aplicación Web. En Tapestry todo son componentes.

Ilustración 3: Tapestry

JSF: La tecnología Java Server Faces es un marco de desarrollo de los componentes de la interfaz de usuario del lado del servidor y es válido para todas aquellas aplicaciones Web basadas en la tecnología JAVA.

Ilustración 4: JavaServer Faces

SPRINGMVC: SpringMVC es un framework de java que facilita la creación de aplicaciones java. Diseñado en módulos, facilita el desarrollo de funcionalidades específica.

Ilustración 5: SpringMVC

2.2.1. Struts

¿Qué es STRUTS?[3]

STRUTS es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE . Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Struts permite reducir el tiempo de desarrollo. Su carácter de "software libre" y su compatibilidad con todas las plataformas en que Java Enterprise esté disponible, lo convierte en una herramienta altamente disponible.

Struts 2 es la siguiente generación de Apache Struts. El objetivo de Struts 2 es simple, hacer más fácil el desarrollo Web para el programador. Para lograr este objetivo, Struts 2 proporciona características para reducir la configuración de XML a través de valores por defecto inteligentes.

Con la versión 2 del Framework se introdujeron algunas mejoras (diseño mejorado, nuevos tags incorporados, mejora la gestión de checkboxes, integración sencilla para SpringMVC, formularios POJO, añadir plugins fácilmente, etc.) sobre la primera versión, de cara a simplificar las tareas más comunes en el desarrollo de aplicaciones web, así como mejorar su integración con AJAX, etc.

[3] Información obtenida de la wikipedia y del libro Starting Struts 2 de Ian Roughley

¿Para qué sirve?

Evidentemente, como todo Framework intenta, simplifica notablemente la implementación de una arquitectura según el modelo MVC. El controlador ya se encuentra implementado por Struts, pero si fuera necesario se puede heredar, modificar o ampliar.

¿Licencia?

Struts está disponible bajo la licencia "free-to-use-license" de la Apache Software Foundation.

Internacionalización

La información que se muestra al usuario está compuesta de partes diferentes que han de trabajar en conjunto de forma coordinada para que la información sea accesible y universal, es decir, esas partes que integran la Web han de funcionar bajo cualquier circunstancia, en cualquier país, con cualquier idioma y cultura. Por este motivo la internacionalización podría definirse como un proceso a través del cual se van a diseñar sitios Web adaptables a diferentes idiomas y regiones sin necesidad de realizar cambios en el código.

Struts brinda soporte para internacionalización extendiendo la funcionalidad que ya provee java. Permite internacionalizar una aplicación utilizando archivos de texto conteniendo archivos de datos con el formato clave=valor y referenciando estas claves en los archivos de la vista.

Validación de entradas

Struts provee mecanismos de validación de las entradas ingresadas. Existen dos maneras principales: Redefiniendo el método validate() de los ActionForms o a través de lo que primero fue un plugin y luego se incorporó a la versión principal y que se

denomina struts-validator[4].

Herramientas de desarrollo

Algunas de las herramientas de desarrollo más comunes para el framework Struts son:

- Eclipse.

Ilustración 6: Eclipse

- IntelliJ (las últimas versiones también trabajan con struts2).
- JCreator.
- NetBeans.

2.2.2. Tapestry

¿Qué es Tapestry? [5]

Tapestry es un framework de código abierto para la creación de aplicaciones Web de forma dinámica, robusta y altamente escalable en java.

Tapestry complementa y construye desde el estándar Java Servlet API, funcionando también en cualquier servidor contenedor de servlets o contenedor de aplicaciones. Tapestry divide una aplicación Web en un conjunto de páginas, cada una compuesta de componentes. Esto le otorga una estructura consistente, lo que permite que Tapestry asuma la responsabilidad de las tareas clave tales como la construcción y

[4] Actualmente struts-validator es un proyecto autónomo que puede funcionar con cualquier aplicación que necesite validar datos sin necesidad de utilizar struts
[5] Información obtenida de la página oficial: http://tapestry.apache.org/

el envío de URL, la validación de usuario, la localización, la internacionalización y la presentación de informes.

Desarrollar aplicaciones Tapestry implica crear plantillas HTML usando HTML plano, y combinando las plantillas con pequeños trozos de código Java usando un descriptor de archivos xml (opcional). En Tapestry, tu creas tus aplicaciones en términos de objetos, y los métodos y propiedades de estos objetos, y no especificando términos de URLs y términos de consulta.

Tapestry es un framework de arquitectura a pequeña escala, está diseñado para la creación de aplicaciones Web fáciles de crear. Tapestry se integra fácilmente con varios tipos de tecnología, por ejemplo: JEE, HiveMind, SpringMVC, Hibernate,…

¿Para qué sirve?

Tapestry es un framework que está pensado para realizar aplicaciones Web en Java que sean dinámicas, robustas y altamente escalables. Su filosofía se basa en lo siguiente: simplicidad en la creación de aplicaciones Web, consistencia a la hora de que distintos desarrolladores pueden encontrar soluciones similares a problemas similares, eficiencia (aplicaciones escalables) y reacción ante los errores aportando diagnósticos.

¿Licencia?

Tapestry está disponible bajo la licencia “free-to-use-license” de la Apache Software Foundation.

Internacionalización

La información que se muestra al usuario está compuesta de partes diferentes que han de trabajar en conjunto de forma coordinada para que la información sea accesible y universal, es decir, esas partes que integran la Web han de funcionar bajo cualquier circunstancia, en cualquier país, con cualquier idioma y cultura. Por este motivo la internacionalización podría definirse como un proceso a través del cual se van a diseñar sitios Web adaptables a diferentes idiomas y regiones sin necesidad de realizar cambios en el código.

Tapestry brinda soporte para internacionalización extendiendo la funcionalidad

que ya provee java. Permite internacionalizar una aplicación utilizando archivos de texto conteniendo archivos de datos con el formato clave=valor y referenciando estas claves en los archivos de la vista.

Validación

La validación la realizan los componentes que reciben los datos de entrada del usuario. No permite métodos propios de validación a no ser que desarrolles tu propio componente de validación. Esto se hace mediante validation.xml. Para poder utilizar el validador habría que configurar las reglas en el validation.xml.

Herramientas de desarrollo

Algunas de las herramientas de desarrollo más comunes para el framework Tapestry son:

- Eclipse.
- IntelliJ (la última versión, la 9, permite trabajar con Tapestry).

Ilustración 7: IntelliJ

- JCreator.
- NetBeans.

2.2.3. JSF

¿Qué es JSF?[6]

JavaServer Faces (JSF) es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa

[6] Wikipedia y página oficial de JSF: http://java.sun.com/javaee/5/docs/tutorial/doc/bnaph.html

JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL[7].

Los principales componentes del framework JSF son:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, conversión de datos, definir un esquema de navegación y dar soporte para internacionalización y accesibilidad; además proporciona extensibilidad para todas estas características.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos librerías de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.
- Beans administrados.

¿Para qué sirve?

JSF brinda una separación entre capas, un desacoplamiento de lo que el usuario ve y lo que el "Sistema" realmente hace, es decir, vista y datos están separados logrando una independencia el uno del otro. Es aplicable aconsejablemente desde un punto de vista personal a sistemas Web de tamaño medio hacia arriba, dada su curva de aprendizaje empinada al principio, pero que sin embargo se ve compensado por una agilidad en el desarrollo una vez que se domina.

¿Licencia?

JavaServer Faces (JSF) está disponible bajo la licencia Open Source de Sun.

Internacionalización

Un requisito muy común hoy en día para cualquier aplicación, es que sea internacional, con esto nos referimos a que de soporte a distintos lenguajes y formatos de datos.

[7] Más información sobre XUL en: https://developer.mozilla.org/es/XUL

Ejemplos de componentes de la aplicación que se ven afectados por esto, son: mensajes, etiquetas de la interfaz de usuario, conjuntos de caracteres y formatos de fechas y monedas. Aunque la internalización es un tema de gran importancia hoy en día en la mayoría de aplicaciones, en las aplicaciones Web es vital, ya que la globalización de Internet hace de ella un requisito prácticamente indispensable.

Validación

En JSF existe un mecanismo para validar los datos locales de todo componente editable, las validaciones.

Estas validaciones ocurren antes de que se actualice el modelo de datos para que concuerde con el valor en local. Como en el modelo de conversión, JSF define una serie de clases estándares para llevar a cabo las validaciones de datos más comunes.

Herramientas de desarrollo

Algunas de las herramientas de desarrollo más comunes para el framework JSF son:

- Eclipse.
- IntelliJ.
- JCreator.

Ilustración 8: JCreator

- NetBeans.

2.2.4. SPRINGMVC

¿Qué es SPRINGMVC?[8]

El Spring Framework (también conocido simplemente como SpringMVC) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. La primera versión fue escrita por Rod Jonhson, quien lo lanzó primero con la publicación

[8] Wikipedia y manual oficial: http://static.springsource.org/spring/docs/2.5.x/reference/index.html

de su libro Expert One-on-One Java EE Design and Development (Wrox Press, octubre 2002). También hay una versión para la plataforma .NET, SpringMVC.net.

SpringMVC Framework está diseñado en torno a un DispatcherServlet que maneja las solicitudes de envío con handler-mappings configurables. Por defecto, un handler es un controlador de interfaz muy simple.

El framework fue lanzado inicialmente bajo Apache 2.0 License en junio de 2003. El primer gran lanzamiento hito fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros hitos en septiembre de 2004 y marzo de 2005.

A pesar de que SpringMVC Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa y sustituto del modelo de Enterprise JavaBean. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.

Mientras que las características fundamentales de este framework pueden emplearse en cualquier aplicación hecha en Java, existen muchas extensiones y mejoras para construir aplicaciones basadas en web por encima de la plataforma empresarial de Java (Java Enterprise Platform).

A partir de 2009 las actualizaciones del producto (en su forma binaria) estarán disponibles únicamente para la última versión publicada del Framework. Para acceder a las actualizaciones en forma binaria para versiones anteriores habrá que pagar una subscripción. Sin embargo, estas actualizaciones estarán disponibles libremente (y gratuitamente) en forma de código fuente en los repositorios públicos del proyecto. No está planteado un cambio de licencia.

¿Para qué sirve?[9]

SpringMVC es un framework para desarrollar aplicaciones Java basadas en Web. Dos de los objetivos más importantes de SpringMVC es permitir que el desarrollo

[9] Spring MVC es un framework de desarrollo de aplicaciones web, pero Spring sirve para todo tipo de aplicaciones, basa su funcionamiento en la inversión de control.

se concentre en la lógica del negocio y que se haga empleando buenos principios de diseño orientado a objetos.

Para lograrlo se utiliza un concepto muy interesante llamado Inversión del Control, también conocido como el principio Hollywood: "No nos llames, nosotros te llamaremos." Esto permite que el código escrito por los desarrolladores para la lógica principal del sistema no tenga dependencias sobre las clases del *framework*; lo cual redunda en un código mucho más limpio y con la posibilidad de utilizar todas las ventajas de la programación orientada a objetos (específicamente la herencia).

¿Licencia?

SpringMVC está disponible bajo la licencia "free-to-use-license" de la Apache Software Foundation.

Internacionalización

La información que se muestra al usuario está compuesta de partes diferentes que han de trabajar en conjunto de forma coordinada para que la información sea accesible y universal, es decir, esas partes que integran la Web han de funcionar bajo cualquier circunstancia, en cualquier país, con cualquier idioma y cultura. Por este motivo la internacionalización podría definirse como un proceso a través del cual se van a diseñar sitios Web adaptables a diferentes idiomas y regiones sin necesidad de realizar cambios en el código.

SpringMVC brinda soporte para internacionalización extendiendo la funcionalidad que ya provee java. Permite internacionalizar una aplicación utilizando archivos de texto conteniendo archivos de datos con el formato clave=valor y referenciando estas claves en los archivos de la vista

Validación

La arquitectura de validación en SpringMVC tiene las siguientes características:

- No está atada a HttpServletRequest.
- No está atada a la capa Web.

- Validación de objetos de dominio.
- Entrada de los clientes remotos también necesita validación.
- Puede ser probada desde fuera del contenedor.
- Implementación independiente.
- Conversión de errores son no-fatal.

Herramientas de desarrollo

Algunas de las herramientas de desarrollo más comunes para el framework SpringMVC son:

- Eclipse.
- IntelliJ.
- JCreator.
- NetBeans.

Ilustración 9: NetBeans

3. Comparativa de tecnologías

3.1. Tabla comparativa

	STRUTS	TAPESTRY	JSF	SPRING MVC
Documentación existente sobre la tecnología	**9**	4	**9**	7
Facilidad de aprendizaje	4	**8**	7	7
Soporte on-line, foros, etc.	9	5	**9**	7
Resolución de dudas	8	**9**	8	7
Funcionalidades extras	8	7	**9**	8
Rapidez en los tiempos de desarrollo	6	**8**	7	**8**
Escalabilidad de las aplicaciones creadas	**9**	**9**	**9**	**9**
Tutoriales y manuales oficiales	**6**	**6**	**6**	**6**
Tipos de licencia de uso	**10**	**10**	**10**	**10**
Medición de la curva de aprendizaje	4	**7**	6	5
Media	**7,3**	**7,3**	**8**	**7,4**

Tabla 1: Comparativa de tecnologías

	STRUTS	TAPESTRY	JSF	SPRING MVC
Documentación existente sobre la tecnología	**9**	4	**9**	7

Tabla 2: Documentación existente

Primero compararemos la **documentación existente** de cada uno de los frameworks. Como podemos comprobar la documentación existente sobre Struts y JSF es mayor que la documentación existente sobre el resto, pero a esto hay que añadirle que es mayor pero también que al haber gran variedad de versiones se hace un poco más difícil encontrar algo, y en el caso de Struts, mucha de esa documentación es de Struts1 y no de la última versión Struts2. SpringMVC también posee bastante documentación y algo menos Tapestry. Por esto se considera que Struts y JSF son mejores en este aspecto.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Facilidad de aprendizaje	4	**8**	7	7

Tabla 3: Facilidad de aprendizaje

En cuanto a la **facilidad de aprendizaje** hay que decir que iniciarse en cualquiera de estos frameworks posee bastante dificultad. Struts, desde un punto de vista personal, es el más complicado de entender, esto es debido a la gran variedad de versiones y la diferencia entre ellas (Struts y Struts2) y a la gran variedad de librerías que entraña. Por ello, en este aspecto considero que Tapestry, e incluso también JSF y SpringMVC son más sencillas de aprender.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Soporte on-line, foros, etc.	**9**	5	**9**	7

Tabla 4: Soporte

Ahora compararemos el **soporte on-line y los foros** en internet que hablan sobre estos frameworks. En este aspecto cabe destacar de nuevo a Struts y JSF. Struts es un framework muy consolidado y por ello que mucha gente utiliza, en cambio JSF es menos maduro pero desde el primer momento a captado gran acogida. Por ello, hay mucha más variedad de ayuda on-line y tutoriales no oficiales al igual que manuales en Internet. SpringMVC también cuenta con bastante documentación on-line aunque es mucho menor que la de Struts y JSF. Por último, Tapestry es el framework de los estudiados del que menos foros hablan en Internet.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Resolución de dudas	8	**9**	8	7

Tabla 5: Resolución de dudas

La **resolución de dudas** es una de las ventajas de usar estos tipos de frameworks. Como estos frameworks han tenido gran aceptación los utiliza gran cantidad de personas y por ello, en foros on-line se pueden exponer las dudas que se tengan con respecto al framework y te contestan rápidamente. Por otro lado, también hay que hablar sobre la resolución de dudas que proporciona Apache. En este aspecto destaca Tapestry. Por ello, podemos concluir en este punto que la resolución de dudas es buena sea cual sea el framework.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Funcionalidades extras	8	7	**9**	8

Tabla 6: Funcionalidades extra

En cuanto a las **funcionalidades** aportadas por cada uno de los frameworks se puede decir que son fácilmente ampliables debido a su gran modularidad. Cada vez que s e crean nuevas librerías se pueden compartir para así añadir nuevas funcionalidades al framework. En este aspecto destaca JSF, debido a que es el framework estudiado con mayor funcionalidad, aunque como he dicho anteriormente todos posee una buena funcionalidad muy ampliable.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Rapidez en los tiempos de desarrollo	6	**8**	7	**8**

Tabla 7: Tiempos de desarrollo

Pasamos a hablar de la **rapidez en los tiempos de desarrollo**. Iniciarse en cualquiera de los frameworks es bastante difícil y costoso pero en cuanto se toma agilidad con el framework, la rapidez en los tiempos de desarrollo es menor en Tapestry y SpringMVC. En JSF también se programa con rapidez, siendo el framework más lento Struts.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Escalabilidad de las aplicaciones creadas	**9**	**9**	**9**	**9**

Tabla 8: Escalabilidad

La **escalabilidad** de las aplicaciones creadas con los frameworks estudiados es muy alta. Debido a la gran modularidad con la que se pueden programar las aplicaciones Web, hay gran facilidad para ampliar la aplicación. Además, gran parte de las versiones que salen nuevas sobre los frameworks acostumbran a ser complementarias con las ya existentes (salvo el caso de Struts a Struts2). Por ello, en este aspecto podemos decir que todos los frameworks son muy escalables.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Tutoriales y manuales oficiales	**6**	**6**	**6**	**6**

Tabla 9: Tutoriales

Los **manuales** de Apache son de gran ayuda, son amplios y detallados y ayudan mucho al programador. Ningún manual ni tutorial destaca por encima de los demás. El único problema que tienen los manuales es que únicamente se pueden conseguir en inglés.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Tipos de licencia de uso	**10**	**10**	**10**	**10**

Tabla 10: Licencia

La **licencia** usada por los frameworks estudiados es "free-to-use-license" de la Apache Software Foundation. Esta licencia, que es la misma para todos, destaca por su coste (muy bajo o gratuito), evita la dependencia de un solo proveedor y permite la creación de comunidades de usuarios que puedan aumentar la valoración de la empresa en los próximos años.

	STRUTS	TAPESTRY	JSF	SPRING MVC
Medición de la curva de aprendizaje	4	**7**	6	5

Tabla 11: Curva de aprendizaje

Por último miraremos las **curvas de aprendizaje** de los diferentes frameworks. Todas ellas son muy escarpadas al principio pero otorgan gran soltura en cuanto se manejan con agilidad. Tapestry es el framework que tiene la curva de aprendizaje

menos escarpada para en principiante, al contrario que Struts que posee mayor dificultad para una persona que no ha programado nunca con él. En el lado contrario, JSF destaca por ser el framework que mayor rapidez de desarrollo proporciona una vez se entienda. En el lado contrario, también se encuentra Struts.

3.2. Elección de la tecnología

Por todo lo anterior, y desde un punto de vista personal, considero que los factores más importantes a la hora de elegir un framework son la documentación existente sobre la tecnología, la facilidad de aprendizaje, el soporte on-line y en foros, la rapidez en los tiempos desarrollo y la curva de aprendizaje. El resto de valores que se han analizado en este documento son muy parecidos y en algunos casos iguales, por lo que los dejaríamos en un segundo plano.

En cuanto a los parámetros en los que nos tenemos que fijar para seleccionar un framework, destaca por encima de todos JSF. SpringMVC también sería una buena opción y mi tercer framework que elegiría seria Struts, relegando a Tapestry a la cuarta posición en mi elección.

En cualquier caso repito que la dificultad es alta sino se disponen conocimientos del framework, siendo la primera aplicación la más difícil de desarrollar. Para implementar una aplicación en cualquiera de estos frameworks, el programador deberá conocer el lenguaje de programación JAVA. Por ello, para que un desarrollador acometa un proyecto en uno de estos frameworks, sería un requisito el conocimiento de JAVA y otro el tener una mínima noción del framework con el cual se dispone a trabajar.

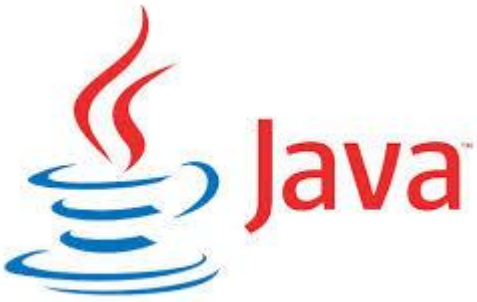


Ilustración 10: JAVA

4. Estudio de la metodología Scrum

4.1. Desarrollo tradicional de software

La forma tradicional de construcción de productos software utiliza en casi todas las ocasiones un ciclo de vida en cascada. Hay veces que dependiendo de la empresa y del proyecto se realizan pequeñas modificaciones como puede ser el ciclo de vida en V. En este ciclo de vida el trabajo de construcción del software se divide en pasos:

- Primero se realiza una fase de planificación donde se elabora un detallado informe de cómo será el producto final y se diseña. Se determinan las tareas necesarias para realizar en el diseño y se realiza una estimación de los recursos necesarios para el desarrollo del producto. Todo esto se realiza con la ayuda de aplicaciones como el proyect y con diagramas como el Gant.

Ilustración 11: Microsoft Project

- Una vez realizado este documento de planificación, los distintos implicados deberán dar el visto bueno y a partir de ahí realizar su trabajo. Estos trabajos son muy especializados y se asemejan con una producción en cadena, donde cada uno tiene su misión y una vez que la realiza se la pasa al siguiente para que realice la suya.
- Durante todo el proceso se realizan estrictos controles donde se evalúa la desviación del proyecto con la estimación establecida en la fase de planificación.

Diagrama de ciclo de vida en cascada o similar

Esta forma de construcción de software tiene sus ventajas e inconvenientes. Si los requisitos del problema se tienen muy claros y se saben que no van a cambiar en el tiempo que se realiza el proyecto y se tiene una fase de planificación bien diseñada y detallada, las ventajas que ofrece este ciclo de vida son muy importantes, ya que el trabajo se realiza rápido, sin sorpresas y todo bajo un fuerte control.

El problema viene cuando los requisitos pueden variar algo en el proceso de creación del software ya que habría que volver a realizar todos los pasos una y otra vez y se retrasaría el proyecto mucho con respecto a la primera estimación.

Otro problema de las metodologías basadas en ciclos de vida en cascada es la inserción de nuevas ideas al proyecto. Si un empleado tiene una buena idea pero se le ocurre cuando él está terminando su parte del proyecto, la inserción de la idea sería muy dificultosa ya que debería repetir la mayor parte del trabajo realizado, además de tener que dar parte para modificar la documentación, e incluso otras partes del proyecto.

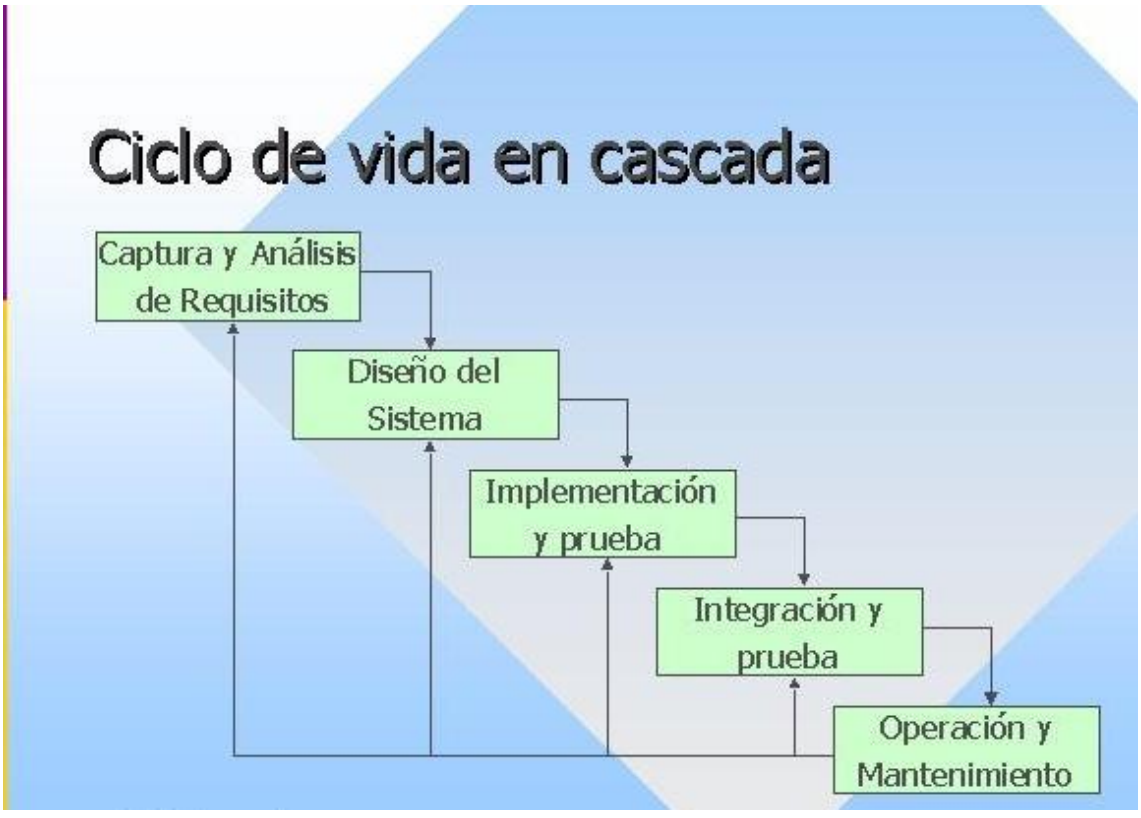


Ilustración 12: Ciclo de vida en cascada

Otro problema de las metodologías basadas en ciclos de vida en cascada es la documentación del proyecto. Al ser tan importante la fase de planificación, la documentación que se obtiene al terminarla, es demasiado amplía, y a veces difícil de leer por los siguientes en el proceso. A veces resulta difícil encontrar la información que es útil a una persona del proyecto. Además, al tener que trabajar todos con los mismos procesos (cada uno su parte: planificación, análisis, diseño, codificación y pruebas) se pueden encontrar discrepancias entre unos y otros, como por ejemplo: "Me está pidiendo construir algo que no está en la especificación", "Está cambiando la idea", "No puedo hacerme responsable de algo que no controlo", etc.

4.2. Métodos ágiles y Scrum

Por todos los inconvenientes anteriores empezaron a desarrollarse proyectos con ciclos de vida incremental e iterativo.

Diagrama ciclo de vida incremental o iterativo

La evolución de estos ciclos de vida fue la que ocasionó la creación de los métodos ágiles. Lo que en principio se quería era una metodología más cercana a la realidad humana en la que tuviera mucha importancia el destinatario del proyecto. Además se pretendía que la aplicación estuviera disponible para pruebas lo más pronto posible (que funcionara aunque no cumpliera todos los requisitos) por si fuera necesario algún cambio.

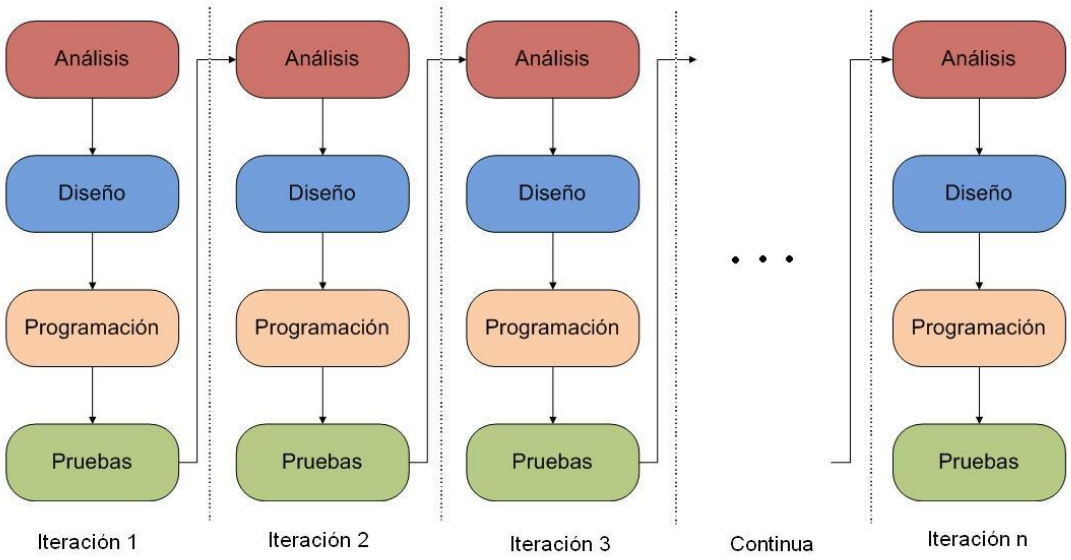


Ilustración 13: Ciclo de vida iterativo

Una de las grandes diferencias con las metodologías basadas en ciclos de vida secuenciales es el rol de cada uno de los empleados del proyecto. Mientras en los proyectos divididos por funcionalidades había una jerarquía con roles muy diferenciados, en las metodologías ágiles se forman grupos pequeños de trabajo con capacidad de decisión independiente con respecto al resto de grupos. Esto es, al dividir el trabajo cada grupo toma sus propias decisiones y crea parte del programa "a su manera". Esto se unifica muy a menudo y se concretan reuniones con el cliente del

producto para que vea las mejoras, opine y proponga cambios.

Historia de SCRUM

Scrum es el método ágil más popular. Haciendo un pequeño histórico, el primer equipo de trabajo que utilizó Scrum lo creó Jeff Sutherland en Easel Corporation en 1993 y el marco de trabajo Scrum lo formalizó Ken Schwaber en 1995. Hoy en día Scrum es usado por empresas de todos los tamaños como Yahoo!, Microsoft, Google, Lockheed Martin, Motorota, SAP, Cisco, Ge, Capitalote y la Reserva Federal de EEUU. Muchos equipos que usan Scrum dicen haber obtenido mejoras sustanciales, y en algunos casos una completa transformación de la productividad y la moral.

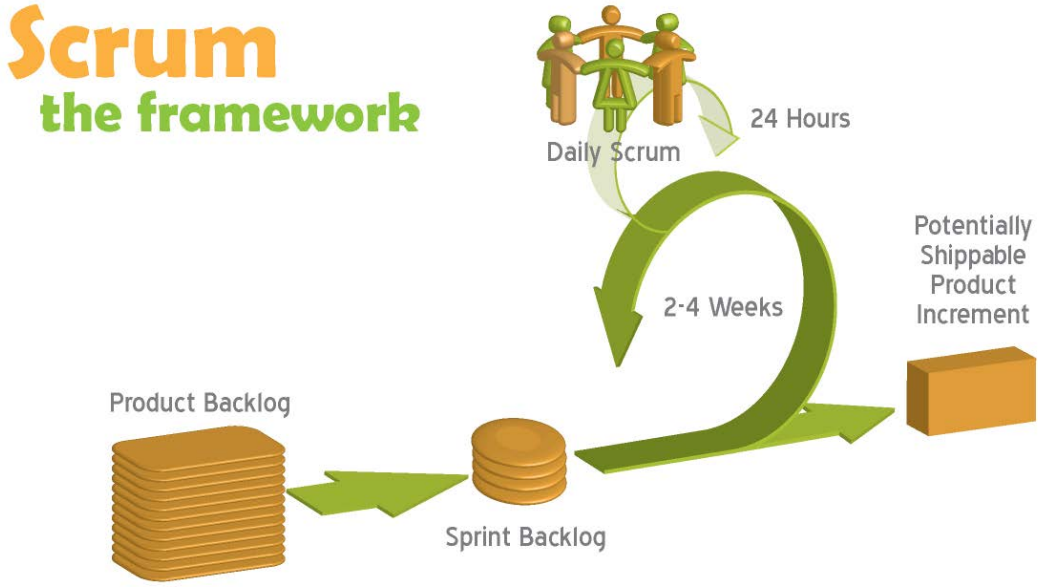


Ilustración 14: Scrum

4.3. Características de Scrum

4.3.1. Roles

La metodología SCRUM tiene 4 roles diferentes:

- **ScrumMaster**: Persona encargada del proyecto. Esta persona será responsable de hablar con el Product Owner (PO, dueño del producto) para confirmar la funcionalidad requerida para el proyecto. El ScrumMaster será, junto con el Product Owner de analizar las distintas funcionalidades que debe tener el proyecto y plasmarlas en historias. También será el encargado de la creación de SPRINTS, según las prioridades y dependencias de las historias.

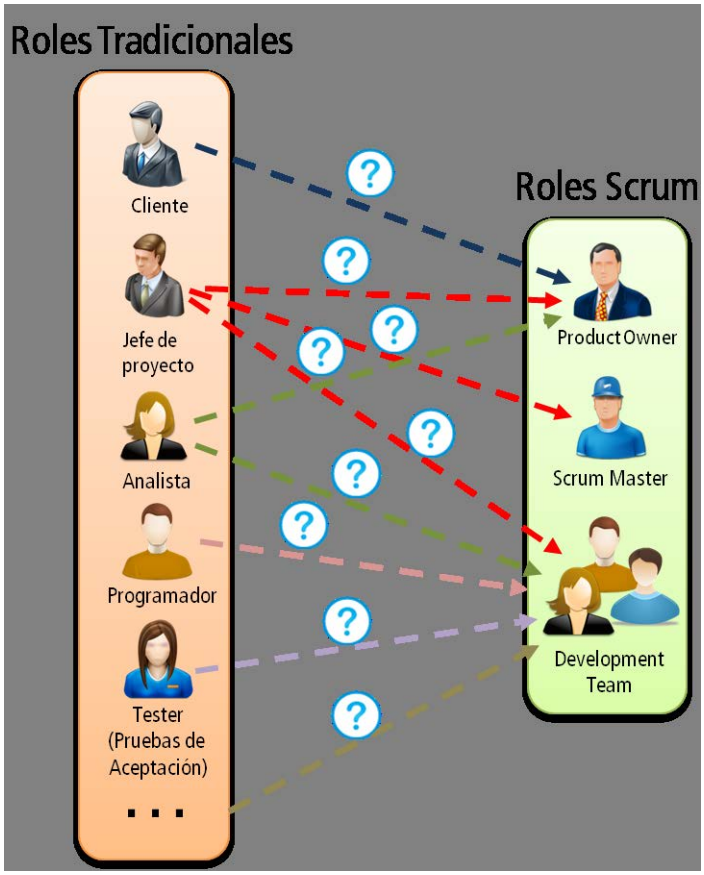


Ilustración 15: Roles

Valorará el rendimiento del equipo de manera que pueda adaptarlos a los SPRINTS.

- **Product Owner (PO)**: es la persona para la que es destinada el proyecto, el "dueño". Es la persona que establece las prioridades de las historias. Debe ser parte del equipo, o como mínimo debe estar localizable para los miembros del equipo puedan preguntarle cualquier cosa. Esto será necesario porque al cambiar los requisitos continuamente pueden surgir dudas con respecto a la funcionalidad.

- **Administrador**: en algunos casos también se puede tener un administrador que provea al equipo, dueño y ScrumMaster de los accesos necesarios a las aplicaciones para llevar a cabo la gestión del proyecto.

- **Equipo de trabajo**: son un conjunto de trabajadores cuya misión será desarrollar el proyecto.

  Su tarea se divide en varias fases, desde la descomposición de las historias en tareas para que puedan ser tratadas, hasta la realización de las pruebas del producto terminado, pasando por la estimación de las tareas, el desarrollo de las mismas, las pruebas unitarias,…

  Cada miembro del equipo de trabajo se juntará a diario en una reunión llamada **Daily Meeting** en la que se hablará de lo realizado en ese día para que el ScrumMaster pueda comprobar la evolución del proyecto. Estas reuniones no deben durar más de 15 minutos.

4.3.2. Descripción de conceptos básicos

Historia: cada uno de los requisitos del proyecto.

Las historias deben contener al menos los siguientes campos:

- Código: será un código único que servirá para diferenciar la historia de las demás y evitar duplicados. El código será siempre fijo.
- Nombre: será un nombre identificativo de la historia para diferenciarla del resto. También debe ser único pero se podrá cambiar según transcurra el proyecto.
- Descripción: será una breve descripción en la que quede claro el cometido de la historia. Debe ser suficientemente claro como para que el Product Owner comprenda aproximadamente de que estamos hablando.
- Prioridad: establece la importancia de la historia.
- Estimación: la valoración del equipo acerca de cuanto trabajo es necesario para implementar la historia. La medida es "puntos de historia", aunque muchas veces los puntos de historia tienen la equivalencia en horas.

- Esta estimación se puede realizar una vez el equipo de trabajo haya realizado la descomposición posterior en tareas.
- Comentarios: cualquier información que sirva para completar la información de la historia.

  Las historias no deberían contener información técnica.

  Ejemplos de historias:

Código	login
Nombre	Un usuario puede entrar en Aplicación rellenando el login
Descripción	Para entrar debe estar registrado
Prioridad	100
Estimación	5
Comentarios	Los datos que rellene en el formulario serán sus datos de sesión

Tabla 12: Ejemplo historia 1

Código	Registro
Nombre	Un usuario puede registrarse para entrar en Aplicación
Descripción	Para registrarse debe rellenar un formulario correctamente
Prioridad	100
Estimación	6
Comentarios	El usuario tiene que poder registrarse en la aplicación mediante un formulario en el que se le pedirán los siguientes datos: nombre, apellidos, teléfono, e-mail

Tabla 13: Ejemplo historia 2

Tarea: Es la descomposición de las historias en partes básicas.

Las tareas deben contener al menos los siguientes campos:

- Código: será un código único que servirá para diferenciar la tarea de las demás y evitar duplicados. El código será siempre fijo.
- Nombre: será un nombre identificativo de la tarea para diferenciarla del resto. También debe ser único pero se podrá cambiar según transcurra el proyecto.
- Descripción: será una breve descripción en la que quede claro el cometido de la tarea. Debe ser una descripción técnica en la que se pueda apoyar para lograr el desarrollo.
- Estimación: la valoración del equipo acerca de cuanto trabajo es necesario para implementar la tarea. La suma de las estimaciones de todas las tareas de una historia debe coincidir con la estimación de la historia. La estimación de la historia se puede hacer basándose en la estimación de las tareas.
- Comentarios: cualquier información o referencias que ayude al desarrollo de la tarea.
- Historia: historia a la que pertenece la tarea. Sirve para relacionar la historia con su tarea cuando sea necesario.

Ejemplos de tareas

Código	loginCrearJSP
Nombre	Login Crear JSP
Descripción	Se debe de crear una página .jsp con un formulario que pida "Usuario" y "Password". Debe de tener un enlace para poder crear una cuenta nueva
Estimación	2
Comentarios	El diseño de la página debe ser el mismo que el resto de las páginas con cuestionario del proyecto. La página tiene que ser multilenguaje.
Historia	login

Tabla 14: Ejemplo tarea 1

Código	loginValidar
Nombre	Login Validar datos
Descripción	Se deben recoger los datos recibidos por el usuario y validarlos con los datos de la tabla correspondiente de la base de datos.
Estimación	2
Comentarios	No hay comentarios
Historia	login

Tabla 15: Ejemplo tarea 2

Sprint: es la agrupación de varias historias que se deben realizar en un determinado tiempo. El tiempo de duración de un Sprint es igual para todos los Sprint de proyecto. La duración suele ser de entre 3 y 4 semanas.

Para poder realizar un Sprint se deben de realizar una reunión de planificación del Sprint en la que ya se debe de tener claro lo siguiente:

- El Product Backlog, con todos los campos completos y correctos.
- Debe presentarse el Product Owner.
- Todas las historias de Product Backlog deben de tener los campos de Prioridad y Estimación.

Los objetivos de esa reunión son los de determinar la meta del Sprint, el equipo que realizará el trabajo y su dedicación (si es completa o se dedica a mas cosas también), la lista de historias que formarán el Sprint (Sprint Backlog) y una fecha concreta de fin y de presentación del Sprint.

Las reuniones deben de tener una duración fija en la que se establezca la meta y el Sprint Backlog. Siempre se debe de cumplir ese tiempo por lo que se deben de tener muy controlados los tiempos. Es una buena idea para las reuniones tener una agenda en

la que está planificado el tempo de la reunión.

Hay 4 factores de vital importancia en la elaboración del Sprint Backlog. Son el alcance, la estimación, la prioridad y la calidad. Todos estos factores deben de considerarse para la elaboración del Sprint. El alcance y la estimación los fija el dueño de producto (por eso es necesario que esté presente en la reunión de creación del Sprint. La estimación la fija el equipo de trabajo. En algunos casos la estimación que ha realizado el equipo de una historia no es la que el Product Owner esperaba por lo que es posible que tenga que cambiar su percepción del Sprint.

El último factor de gran importancia en la creación del Sprint es la calidad. En cuanto a la calidad, se puede distinguir la calidad externa y la calidad interna. La calidad externa se refiere a lo que "el usuario ve", como un interfaz lento, poco intuitivo; mientras que la calidad interna es lo que "el usuario no ve", como diseño de sistemas, cobertura de pruebas, legibilidad del código.

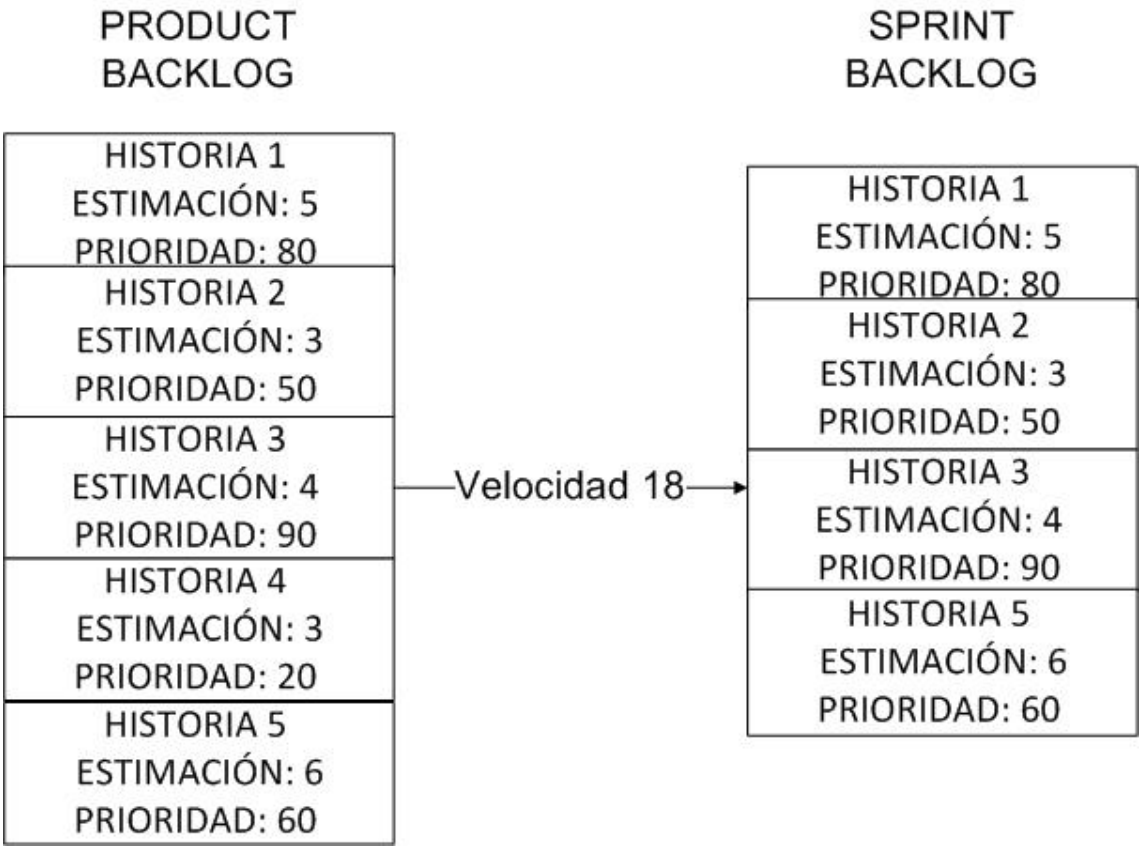


Ilustración 16: Product Backlog vs. Sprint Backlog

La calidad interna es primordial. Un aplicativo con baja calidad interna es difícil que tenga buena calidad externa. En una metodología ágil, como SCRUM, la calidad interna es primordial ya que asumimos que puede haber muchos cambios de requisitos, por lo que una buena legibilidad del código ayuda a su mantenimiento y adaptación.

La calidad externa puede ser mayor o menor dependiendo del alcance del Sprint. Esta es una de las cosas que se debe tratar en la reunión de planificación del Sprint.

Las historias a incluir en el Sprint se deben seleccionar de la siguiente manera. Primero se buscarán las que se necesitan para lograr que se cumpla la meta del Sprint. Ahora se mirará la duración de las historias y su prioridad para seleccionar las que mejor se adapten.

Velocidad: Para determinar la velocidad de un grupo debemos fijarnos en los últimos Sprints que han realizado. De ahí se puede sacar una estimación de la velocidad del grupo. También nos deberíamos fijar en la dedicación que van a tener los miembros del equipo en el proyecto, es decir, si es exclusiva, si lo compaginan con otro, etc.

Las tareas se suelen estimar en una reunión. La estimación se hace entre todos. En la reunión, cada miembro del equipo tendrá unas tarjetas con números y cuando llegue el momento de la estimación, por cada historia a estimar, cada persona dará su opinión sobre la estimación. Esta técnica está pensada para que el equipo de trabajo no desconecte de la reunión en ningún momento.

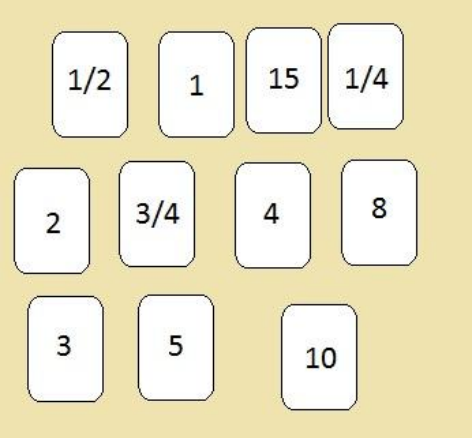


Ilustración 17: Tarjetas estimación

Como se puede ver no están todos los números. Esto se debe a que las historias grandes son más difíciles de estimar y se quiere recalcar que es una estimación, no una cifra exacta. Cabe destacar que la estimación no tiene porque realizarse refiriéndose a un esfuerzo en horas o días, sino que muchas veces es una estimación relativa refiriéndose al tamaño de la historia (por ejemplo se hacen estimaciones con barajas de póker, tallas de ropa, etc.).

La forma de presentar las tareas para su estimación (se pueden presentar ya

divididas en las tareas) puede ser de manera manual o mediante alguna aplicación:

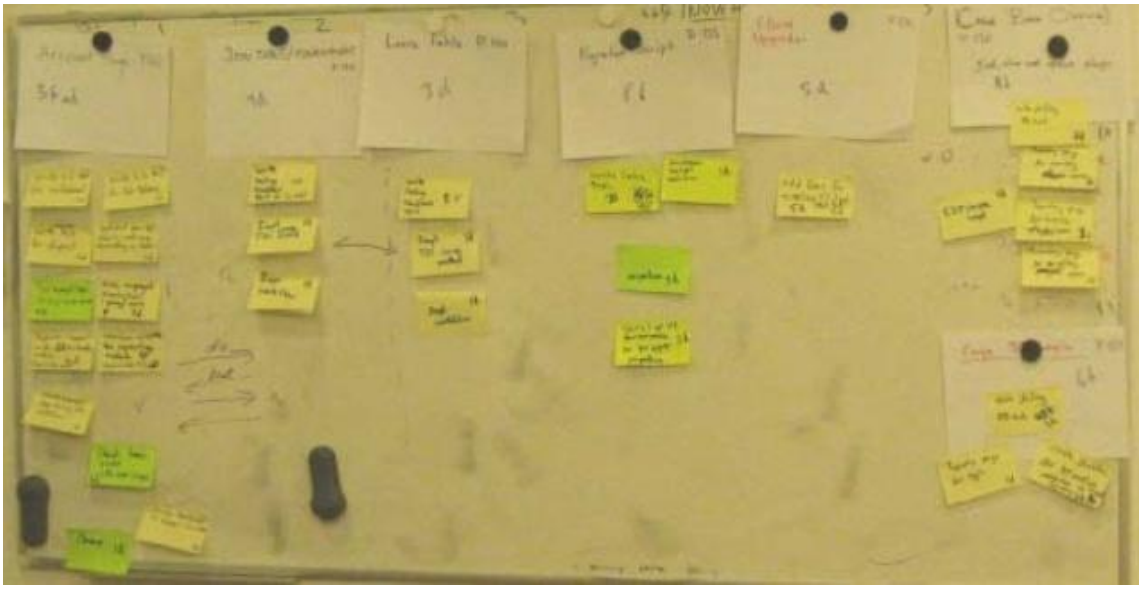


Ilustración 18: Product Backlog

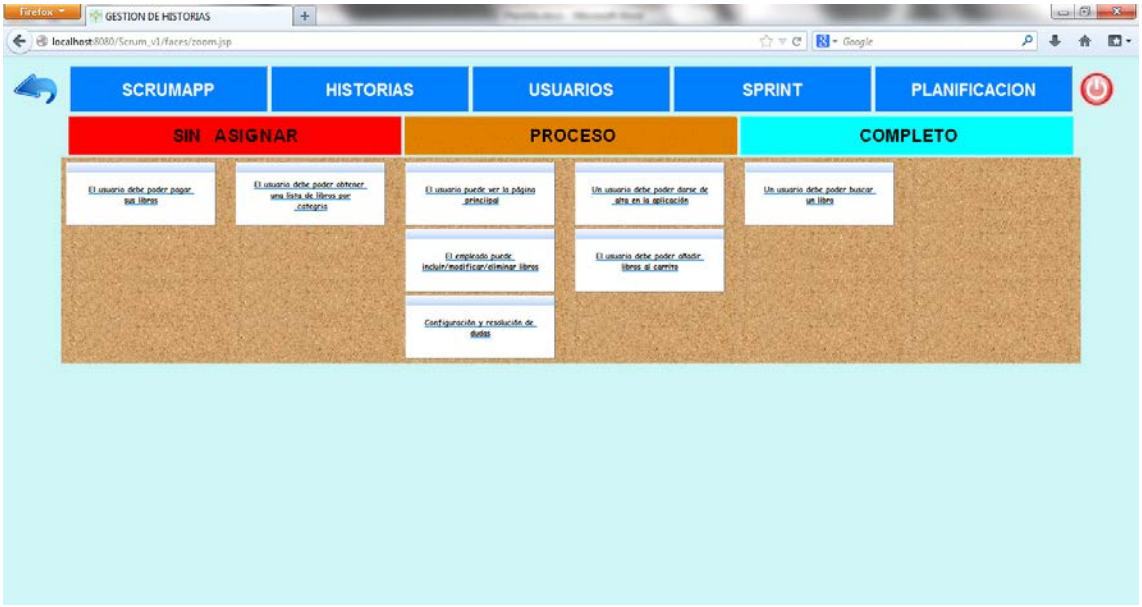


Ilustración 19: Product Backlog (aplicación)

Scrum diario: es una reunión diaria en la que se determina el trabajo que se va a desempeñar y lo que se ha hecho hasta entonces. Debe ser una reunión corta, no más de 15 minutos y se debe tener todos los días sin excepción.

Demo: la demo del Sprint es una presentación que se hace al final de Sprint para el Product Owner para que vea lo realizado en el Sprint, si le gusta o si hay algo que preferiría de otro modo.

Como preparar la demo. Hay que asegurarse de que el objetivo del sprint quede muy claro. En la demo deben de participar las personas implicadas en la realización del Sprint. La duración aproximada será de 1 hora por cada semana que dure el Sprint. En esta reunión, el equipo de trabajo explicará lo realizado y expondrá el resultado; y finalmente será el Product Owner el encargado de decidir si es lo que esperaba o si necesita algún cambio.

En la DEMO también es un buen momento para que el equipo de trabajo exponga todo lo que observaron durante el Sprint y algunas ideas del producto que han aparecido.

5. Análisis

5.1. Estructura del análisis

Como hemos mencionado en el apartado anterior, la metodología Scrum difiere de las metodologías "tradicionales". El análisis está orientado a que se puede hacer con la aplicación.

Para el documento de Análisis he creado unas tablas con la funcionalidad que debe tener el proyecto. Las tablas son "PRODUCT BACKLOG" y "SPRINT BACKLOG".

El Product Backlog tiene la funcionalidad principal del proyecto, el que debe hacer. Por ejemplo, un usuario se puede dar de alta, un Product Owner puede crear una historia nueva en el Product Backlog.

El Sprint Backlog recoge las historias y tareas asignadas a cada usuario para cada periodo del proyecto. En mi caso, como tan sólo estoy yo asignado al proyecto todas las tareas serán para mí.

Por último, el tiene todas las historias en detalle (descompuestas en tareas).

5.2. Detalle del análisis

5.2.1. Product Backlog

Código	Configuración
Nombre	Configuración y aprendizaje
Descripción	Se configura el entorno de desarrollo y se adquieren los conocimientos necesarios.
Prioridad	100
Estimación	775
Comentarios	

Tabla 16: Casos de uso - Configuración

Código	CrearCuenta
Nombre	El usuario puede crearse una cuenta para acceder a la aplicación
Descripción	El usuario debe poder crearse una cuenta para acceder a la aplicación. Al usuario se le presentará una página que le mostrará un formulario con el usuario que quiera tener, su nombre, apellido, teléfono, email y contraseña. Si los datos son correctos, se le dará de alta en el sistema.
Prioridad	100
Estimación	5,25
Comentarios	

Tabla 17: Casos de uso - Crear cuenta

Código	Identificación
Nombre	El usuario puede identificarse en la aplicación
Descripción	El usuario debe poder identificarse en la aplicación si está registrado
Prioridad	100
Estimación	3,5
Comentarios	Para ello se creará una página que le muestre un "login" (usuario y contraseña). Se le dará la opción de crear una cuenta nueva

Tabla 18: Casos de uso - Identificación

Código	Salir
Nombre	El usuario puede salir de la aplicación
Descripción	El usuario debe poder salir de la aplicación
Prioridad	100
Estimación	1
Comentarios	Se deben crear iconos para poder salir de la aplicación desde cualquier punto de ella.

Tabla 19: Casos de uso - Salir

Código	ListadoProyectos
Nombre	El usuario puede ver el listado de proyectos en los que participa
Descripción	El usuario puede ver el listado de los proyectos en los que participa
Prioridad	100
Estimación	12
Comentarios	Al usuario le debe aparecer una lista con los proyectos en los que participa. La lista estará compuesta por el código de proyecto, el nombre del proyecto, el rol dentro del proyecto, la descripción y las fechas de inicio y fin.

Tabla 20: Casos de uso - Listado proyectos

Código	CrearProyecto
Nombre	El usuario puede crear un proyecto
Descripción	El usuario debe poder crear un proyecto
Prioridad	100
Estimación	6
Comentarios	Se mostrará una pantalla con el siguiente formulario: - Código de proyectos * - Nombre de proyecto * - Descripción del proyecto - ScrumMaster * - Versión y Revisión - Fecha de inicio * - Fecha prevista de fin * - Comentarios - Presupuesto.

Tabla 21: Casos de uso - Crear proyecto

Código	VerProyecto
Nombre	El usuario puede ver un proyecto en el que participa
Descripción	El usuario debe poder ver un proyecto en el que participa
Prioridad	100
Estimación	6,5
Comentarios	El usuario debe poder ver un proyecto en el que participa. Para ver el proyecto debe haberlo seleccionado en la página de la lista de proyectos. Se deben mostrar todos los campos que se han pedido para crearlo. En esta página habrá enlaces a: - Entrar al proyecto - Modificar el proyecto (sólo ScrumMaster) - Borrar el proyecto (sólo ScrumMaster) - Añadir empleados al proyecto (sólo ScrumMaster) - Borrar empleados del proyecto (sólo ScrumMaster) - Generar documentación

Tabla 22: Casos de uso - Ver proyecto

Código	EditarProyecto
Nombre	El ScrumMaster puede modificar los datos generales de un proyecto suyo
Descripción	El ScrumMaster podrá modificar los datos generales de un proyecto suyo
Prioridad	100
Estimación	11
Comentarios	El ScrumMaster podrá modificar los siguientes datos: descripción, versión y revisión, fecha de inicio, fecha prevista de fin, comentarios y presupuesto

Tabla 23: Casos de uso - Modificar proyecto

Código	BorrarProyecto
Nombre	El ScrumMaster puede borrar un proyecto suyo
Descripción	El ScrumMaster puede borrar un proyecto suyo
Prioridad	100
Estimación	9
Comentarios	Se deben borrar todos los datos de ese proyecto salvo los usuarios que participan. Se debe pedir una confirmación antes de borrar

Tabla 24: Casos de uso - Borrar proyecto

Código	EntrarProyecto
Nombre	Un usuario puede entrar en un proyecto en el que participe
Descripción	Un usuario puede entrar en un proyecto en el que participe
Prioridad	100
Estimación	5
Comentarios	El usuario debe poder entrar en un proyecto en el que participe. La pantalla que le tiene que salir es la de gestión de historias

Tabla 25: Casos de uso - Entrar al proyecto

Código	CrearHistoria
Nombre	El ScrumMaster puede crear una historia en un proyecto suyo
Descripción	El ScrumMaster debe poder crear una historia en un proyecto suyo
Prioridad	100
Estimación	6
Comentarios	El usuario debe poder crear un proyecto. Para ello se le mostrará una pantalla en la que deberá rellenar un formulario con los siguientes datos: - Código de historia * - Nombre de historia * - Descripción de la historia - Comentarios - Prioridad *

Tabla 26: Casos de uso - Crear una historia

Código	BorrarHistoria
Nombre	El ScrumMaster puede borrar una historia en un proyecto suyo
Descripción	El ScrumMaster puede borrar una historia en un proyecto suyo
Prioridad	100
Estimación	7
Comentarios	

Tabla 27: Casos de uso - Borrar historia

Código	EditarHistoria
Nombre	El ScrumMaster puede modificar una historia en un proyecto suyo
Descripción	El ScrumMaster puede modificar una historia en un proyecto suyo
Prioridad	100
Estimación	10,5
Comentarios	Los valores que puede modificar el ScrumMaster son descripción de la historia, comentarios de la historia y prioridad de la historia

Tabla 28: Casos de uso - Modificar historia

Código	VerHistoria
Nombre	Un usuario puede ver una historia de un proyecto en el que está registrado
Descripción	Un usuario puede ver una historia de un proyecto en el que está registrado
Prioridad	100
Estimación	7
Comentarios	Se podrán consultar todos los campos relativos a la historia. Aparecerán una serie de opciones que serán: - Modificar la historia (ScrumMaster) - Borrar la historia (ScrumMaster) - Fin de pruebas (cualquier persona) - indica que se ha terminado de implementar y probar - Ver tareas de la historia (cualquier persona)

Tabla 29: Casos de uso - Ver historia

Código	VerTarea
Nombre	Un usuario puede ver una tarea de un proyecto en el que participa
Descripción	Un usuario puede ver una tarea de un proyecto en el que participa
Prioridad	100
Estimación	5
Comentarios	Se mostrarán todos los datos que tenga la tarea

Tabla 30: Casos de uso - Ver tarea

Código	ListaTareas
Nombre	Un usuario puede ver la lista de tareas de una historia de un proyecto en el que participa
Descripción	Un usuario puede ver la lista de tareas de una historia de un proyecto en el que participa
Prioridad	100
Estimación	7
Comentarios	Los datos que se mostrarán en la lista son los siguientes: - Código de la tarea - Nombre de la tarea - Descripción de la tarea - Persona asignada - Tipo de tarea Debe mostrar un acceso a crear una nueva tarea y otro para ver el detalle de la tarea

Tabla 31: Casos de uso - Lista de tareas

Código	CrearTarea
Nombre	Un usuario puede crear una tarea en un proyectoo en el que participa
Descripción	Un usuario puede crear una tarea de una historia de un proyecto en el que participa
Prioridad	100
Estimación	7,5
Comentarios	La tarea debe tener un código único en la historia. Los datos que se deben pedir para registrar una tarea son: - Código de la tarea - Nombre de la tarea - Descripción de la tarea - Comentarios de la tarea - Duración de la tarea

Tabla 32: Casos de uso - Crear tarea

Código	BorrarTarea
Nombre	Un usuario / ScrumMaster puede borrar una tarea que tenga asignada
Descripción	Un usuario puede borrar una tarea que tenga asignada. También puede borrar la tarea el ScrumMaster
Prioridad	100
Estimación	5
Comentarios	

Tabla 33: Casos de uso - Borrar tarea

Código	VerSprint
Nombre	Un usuario puede ver el Sprint de los proyectos en los que participa
Descripción	Un usuario puede ver el Sprint de los proyectos en los que participa
Prioridad	100
Estimación	19,5
Comentarios	El usuario verá los datos generales del Sprint además de todos los datos del PROGRESO de ese Sprint

Tabla 34: Casos de uso - Ver Sprint

Código	ElegirSprint
Nombre	Un usuario puede cambiar de Sprint en un proyecto en el que participa
Descripción	Un usuario podrá moverse por los Sprint que tengan los proyectos en los que participa
Prioridad	100
Estimación	8,5
Comentarios	Para ello se le dará una lista con los Sprint que tiene el proyecto y algún dato característico de ellos. La lista tendrá los siguientes campos: - Nombre del sprint - Número Sprint - Fecha de inicio - Fecha de fin

Tabla 35: Casos de uso - Listar Sprints

Código	CrearSprint
Nombre	El ScrumMaster puede crear un Sprint en un proyecto suyo
Descripción	El ScrumMaster puede crear un Sprint en un proyecto suyo
Prioridad	100
Estimación	24
Comentarios	Los datos que se necesitan del Sprint son: el nombre y la fecha de inicio y fin del Sprint. Una vez creado, se podrán añadir historias al Sprint antes de que comience. La fecha de inicio del Sprint tiene que ser superior a la fecha de fin del Sprint anterior

Tabla 36: Casos de uso - Crear Sprint

Código	AñadirEmpleado
Nombre	El ScrumMaster puede añadir empleados al proyecto
Descripción	El ScrumMaster puede añadir empleados al proyecto. Para ello tendrá una pantalla donde le aparecerá una lisa de los empleados que ya hay asignados al proyecto y unas cajas para poder dar de alta otros.
Prioridad	100
Estimación	7,5
Comentarios	Los usuarios que se quieran dar de alta en el proyecto ya tienen que haberse dado de alta en el sistema

Tabla 37: Casos de uso - Añadir empleado

Código	EliminarEmpleado
Nombre	El ScrumMaster puede eliminar usuarios de un proyecto suyo
Descripción	El ScrumMaster puede eliminar usuarios de un proyecto. Para poder hacerlo, le aparecerá una lista con los usuarios que hay dados de alta en el proyecto y podrá borrar uno de ellos
Prioridad	100
Estimación	7
Comentarios	

Tabla 38: Casos de uso - Eliminar empleado

Código	EditarEmpleado
Nombre	Un usuario podrá modificar los datos personales
Descripción	Un usuario podrá modificar los datos personales con los que se ha dado de alta
Prioridad	100
Estimación	12
Comentarios	Los datos que puede modificar son: - Teléfono - E-mail - Velocidad - Password

Tabla 39: Casos de uso - Modificar empleado

Código	Product Backlog
Nombre	Un usuario puede ver el Product Backlog
Descripción	Un usuario puede ver el Product Backlog
Prioridad	100
Estimación	30
Comentarios	Se le mostrará una página en la que estén las historias que participan en un proyecto ordenadas por porcentaje realizado. Esta pantalla tendrá un enlace a una pantalla en la que se vea el progreso de las historias "en curso"

Tabla 40: Casos de uso - Product Backlog

Código	CrearInforme
Nombre	Un usuario podrá crear un informe del trabajo diario
Descripción	Un usuario podrá crear un informe del trabajo que ha hecho ese día.
Prioridad	100
Estimación	14
Comentarios	El usuario deberá rellenar los informes de las tareas que ha realizado ese día. Primero aparecerá una pantalla con las tareas que tiene asignadas del proyecto y cuando seleccione alguna de ellas un formulario en el que se le pregunte cuanto ha trabajado en ella

Tabla 41: Casos de uso - Crear informe

5.2.2. Descomposición en tareas

Usuario	Historia	Tareas	Estimación
Todos	**El usuario puede crearse una cuenta para acceder a la aplicación**		**730**
		Formación JAVA	325
		Formación JSF	200
		Configurar eclipse, mysql, tomcat, richfaces, …	200
		Crear página para crear la cuenta	1
		Crear formulario	0,5
		Crear módulo controlador para recibir y tratar los datos	1
		Validar formulario (longitudes, email, …)	0,5
		Crear módulo modelo para insertar datos en tabla	1
		Retornar a la página de login	0,25
		Crear archivo Properties con los literales	0,75
Todos	**El usuario puede identificarse en la aplicación**		**3,75**
		Crear página para identificarse (login)	1
		Crear formulario de login	0,5
		Crear módulo controlador para recibir y tratar los datos	1
		Una vez verificado el usuario, añadirlo a USER_SESSION_KEY	0,25
		Direccionar a la página de listado de proyecto	0,25
		Modificar archivo Properties con los literales	0,75
Todos	**El usuario puede salir de la aplicación**		**1**
		Crear módulo controlador para hacer logout	1

Tabla 42: Descomposición en tareas

Usuario	Historia	Tareas	Estimación
Todos	**El usuario puede ver el listado de proyectos en los que participa**		**12**
		Crear página para ver los proyectos	1
		Crear el módulo modelo para acceder a la tabla	1
		Crear el módulo controlador que agrupe en un formato los	2
		Realizar el Datatable para mostrar los datos	2
		Recibir los datos por código en un INPUTTEXT o por enlace	2
		Crear módulo controlador para comprobar si ese usuario tiene permisos en ese proyecto	3
		Direccionar a la página de crear proyecto si se quiere crear	0,25
		Modificar archivo Properties con los literales	1
Todos	**El usuario puede crear un proyecto**		**6**
		Crear página para crear los proyectos	1
		Crear formulario	0,5
		Crear módulo controlador para recibir y tratar los datos	1
		Validar formulario (longitudes, email, …)	0,5
		Crear módulo modelo para insertar datos en tablas	2
		Retornar a la página de ver proyecto	0,25
		Modificar archivo Properties con los literales	0,75
Todos	**El usuario puede ver un proyecto en el que participa**		**6,5**
		Crear página para ver un proyecto	1
		Crear módulo modelo para poder acceder a la tabla	2
		Crear módulo controlador que agrupe en un formato los datos a mostrar	1
		Realizar el Datatable para mostrar los datos	1
		Direccionar a la página de modificar los datos/características de un proyecto	0,25
		Direccionar a la página de borrar el proyecto	0,25
		Direccionar a la página de entrar al proyecto	0,25
		Modificar archivo Properties con los literales	0,75
Todos	**El ScrumMaster puede modificar los datos generales de un proyecto suyo**		12
		Crear página para ver los proyectos	1
		Crear el módulo modelo para acceder a la tabla	1
		Crear el módulo controlador que agrupe en un formato los	2
		Realizar el Datatable para mostrar los datos	2
		Poder recibir los datos o por código en una caja de texto o	2
		Crear módulo controlador para comprobar si ese usuario	3
		Direccionar a la página de crear proyecto si se quiere crear	0,25
		Modificar archivo Properties con los literales	0,75

Usuario	Historia	Tareas	Estimación
Scrum Master	**El ScrumMaster puede borrar un proyecto suyo**		**9**
		Crear módulo controlador que verifique que se tienen permisos para borrar	1
		Crear módulo modelo para poder acceder a la tabla	5
		Crear módulo controlador que gestione la query	1
		Realizar el borrado de las tablas	1
		Direccionar a la página ver listado de proyectos	0,25
		Modificar archivo Properties con los literales	0,75
Empleado	**Un usuario puede entrar en un proyecto en el que participe**		**5**
		Crear página para ver el progreso de un proyecto	1
		Crear modulo modelo que busque las historias del proyecto	2
		Crear el módulo controlador que agrupe en un formato los datos a mostrar	1
		Direccionar a crear nueva historia	0,25
		Direccionar a ver una historia a partir de un enlace o por una caja de texto	0,25
		Modificar archivo Properties con los literales	0,5
Scrum Master	**El ScrumMaster puede crear una historia en un proyecto suyo**		**6**
		Crear página para crear las historias	1
		Crear formulario	0,5
		Crear módulo controlador para recibir y tratar los datos	1
		Validar formulario (longitudes, email, ...)	0,5
		Insertar datos en tablas	2
		Direccionar a la página de entrar al proyecto	0,25
		Modificar archivo Properties con los literales	0,75
Scrum Master	**El ScrumMaster puede borrar una historia en un proyecto suyo**		**7**
		Crear módulo controlador que verifique que se tienen permisos para borrar	1
		Crear módulo modelo para poder acceder a la tabla	3
		Crear módulo controlador que gestione la query	1
		Realizar el borrado de las tablas	1
		Direccionar a la página de entrar al proyecto	0,25
		Modificar archivo Properties con los literales	0,75

Usuario	Historia	Tareas	Estimación
Scrum Master	**El ScrumMaster puede modificar una historia en un proyecto suyo**		**10,5**
		Crear página para modificar una historia	1
		Crear módulo modelo para poder acceder a la tabla	1
		Crear módulo controlador que agrupe en un formato los datos a mostrar	2
		Realizar el Datatable para mostrar los datos	1
		En el datatable algunos campos serán modificables	1
		Crear módulo controlador que reciba los datos modificados	1,5
		Crear módulo controlador para comprobar que tiene permisos	1
		Crear módulo modelo que modifique los datos	1
		Direccionar a la página ver una historia	0,25
		Modificar archivo Properties con los literales	0,75
Empleado	**Un usuario puede ver una historia de un proyecto en el que está registrado**		**7**
		Crear página para ver una historia	1
		Crear módulo modelo para poder acceder a la tabla	2
		Crear módulo controlador que agrupe en un formato los datos a mostrar	1
		Realizar el Datatable para mostrar los datos	1
		Direccionar a la página de modificar los datos/características de un historia	0,25
		Direccionar a la página de crear una nueva historia	0,25
		Direccionar a la página de borrar la historia	0,25
		Direccionar a la página de entrar a un listado de las tareas de la historia	0,25
		Direccionar a la página de añadir empleados	0,25
		Modificar archivo Properties con los literales	0,75

Usuario	Historia	Tareas	Estimación
Empleado	**Un usuario puede ver la lista de tareas de una historia de un proyecto en el que participa**		**5**
		Crear página para ver una tarea	1
		Crear módulo modelo para poder acceder a la tabla	1
		Crear módulo controlador que agrupe en un formato los datos a mostrar	1
		Realizar el Datatable para mostrar los datos	1
		Direccionar a la página de ver una tarea en particular	0,25
		Modificar archivo Properties con los literales	0,75
Empleado	**Un usuario puede ver una tarea de un proyecto en el que participa**		**7**
		Crear página para ver el listado de las tareas de una historia	1
		Crear módulo modelo para poder acceder a la tabla	2
		Crear módulo controlador que agrupe en un formato los datos a mostrar	1
		Realizar el Datatable para mostrar los datos	1,5
		Direccionar a la página de modificar los datos/características de una tarea	0,25
		Direccionar a la página de crear una nueva tarea	0,25
		Direccionar a la página de borrar la tarea	0,25
		Modificar archivo Properties con los literales	0,75

Usuario	Historia	Tareas	Estimación
Empleado	**Un usuario puede crear una tarea en un proyectoo en el que participa**		**7,5**
		Crear página para crear una tarea nueva	1
		Crear formulario	1
		Crear módulo controlador que verifique que se tienen permisos	1
		Validar formulario (longitudes, email, ...)	0,5
		Crear módulo controlador para recibir y tratar los datos	1
		Crear módulo modelo para insertar datos en tablas	2
		Direccionar a la página de ver listado de tareas de una historia	0,25
		Modificar archivo Properties con los literales	0,75
Empleado	**Un usuario / ScrumMaster puede borrar una tarea que tenga asignada**		**5**
		Crear módulo controlador que verifique que se tienen permisos para borrar	1
		Crear módulo modelo para poder acceder a la tabla	1
		Crear módulo controlador que gestione la query	1
		Realizar el borrado de las tablas TAREA	1
		Direccionar a la página de ver listado de tareas de una historia	0,25
		Modificar archivo Properties con los literales	0,75
Empleado	**Un usuario puede ver el Sprint de los proyectos en los que participa**		**19,5**
		Crear página para poder ver un Sprint	3
		Crear módulo modelo para poder acceder a la tablas	1
		Crear módulo controlador que agrupe en un formato los datos a mostrar	4
		Realizar el Datatable para mostrar los datos	3
		Realizar esquema para mostrar el avance del Sprint	7
		Direccionar a modificar Sprint	0,25
		Direccionar a ver otros Sprint	0,25
		Direccionar a crear nuevo Sprint	0,25
		Modificar archivo Properties con los literales	0,75

Usuario	Historia	Tareas	Estimación
Empleado	**Un usuario puede cambiar de Sprint en un proyecto en el que participa**		**8,5**
		Crear página para poder elegir Sprint	1
		Crear formulario	1
		Crear módulo controlador que rellene el formulario	2
		Crear módulo controlador que recoja el Sprint elegido	1
		Realizar esquema para mostrar el avance del Sprint	2
		Direccionar a ver el Sprint actual (actual es el elegido)	0,25
		Direccionar a ver otros Sprint	0,25
		Direccionar a crear nuevo Sprint	0,25
		Modificar archivo Properties con los literales	0,75
Scrum Master	**El ScrumMaster puede crear un Sprint en un proyecto suyo**		**24**
		Crear página para crear un Sprint	15
		Crear formulario	2
		Crear módulo controlador que verifique los permisos para crear un Sprint nuevo	1
		Validar formulario (longitudes, email, ...)	0,5
		Crear módulo controlador para recibir y tratar los datos	2,5
		Crear módulo modelo para insertar datos en tablas	2
		Direccionar a la página de ver el Sprint actual	0,25
		Modificar archivo Properties con los literales	0,75
Scrum Master	**El ScrumMaster puede añadir empleados al proyecto**		**7,5**
		Crear página para añadir usuarios a los proyectos	1
		Crear formulario	1
		Crear módulo controlador que verifique los permisos	1
		Validar formulario (longitudes, email, ...)	0,5
		Crear módulo controlador para recibir y tratar los datos	1
		Crear módulo modelo para insertar datos en tablas	2
		Direccionar a la página de ver el proyecto	0,25
		Modificar archivo Properties con los literales	0,75

Usuario	Historia	Tareas	Estimación
Scrum Master	**El ScrumMaster puede eliminar usuarios de un proyecto suyo**		**7**
		Crear página para eliminar usuarios de los proyectos	1
		Crear formulario	1
		Crear módulo controlador para comprobar si puede eliminar usuarios	1
		Crear módulo controlador para recibir y tratar los datos	1
		Crear módulo modelo para eliminar datos en tablas	2
		Direccionar a la página de ver el proyecto	0,25
		Modificar archivo Properties con los literales	0,75
Empleado	**Un usuario podrá modificar sus datos personales**		**12**
		Crear página para modificar los datos personales	1
		Crear módulo modelo para poder acceder a la tabla	1
		Crear módulo controlador que agrupe en un formato los datos a mostrar	2
		Realizar el Datatable para mostrar los datos	1
		En el datatable algunos campos serán modificables no lo será el código)	1
		Crear módulo controlador que reciba los datos modificados	2
		Crear módulo controlador para comprobar que tiene permisos	1
		Crear módulo modelo que modifique los datos	2
		Direccionar a la página ver un Sprint	0,25
		Modificar archivo Properties con los literales	0,75
Empleado	**Un usuario puede ver el Product Backlog**		**30**
		Crear página para ver el Product Backlog	5
		Crear módulo modelo para poder acceder a las tabla	2
		Crear módulo controlador que agrupe en un formato los datos a mostrar	4
		Realizar el Datatable para mostrar los datos	3
		Poder hacer zoom en el apartado "progreso"	15,25
		Modificar archivo Properties con los literales	0,75

Usuario	Historia	Tareas	Estimación
Empleado	**Un usuario podrá crear un informe del trabajo diario**		**14**
		Crear página para crear informe diario	1
		Crear módulo modelo para poder acceder a la tabla	1
		Crear módulo controlador que agrupe en un formato los datos a mostrar	2
		Realizar el Datatable para mostrar los datos	1
		Crear módulo controlador que reciba los la tarea a informar	2
		Crear módulo controlador para ver si tiene tareas a informar	1
		Crear página que permita introducir el informe de esa tarea	1
		Crear módulo controlador que reciba el informe	1
		Crear módulo modelo para modificar las tablas	3
		Direccionar a la página ver un Sprint	0,25
		Modificar archivo Properties con los literales	0,75

5.2.3. Sprints

Los Sprints que se van a mostrar son todos menos el inicial (aprendizaje y configuración).

El primer Sprint tiene una parte de la configuración y el primer caso de uso. Poder crearse una cuenta.

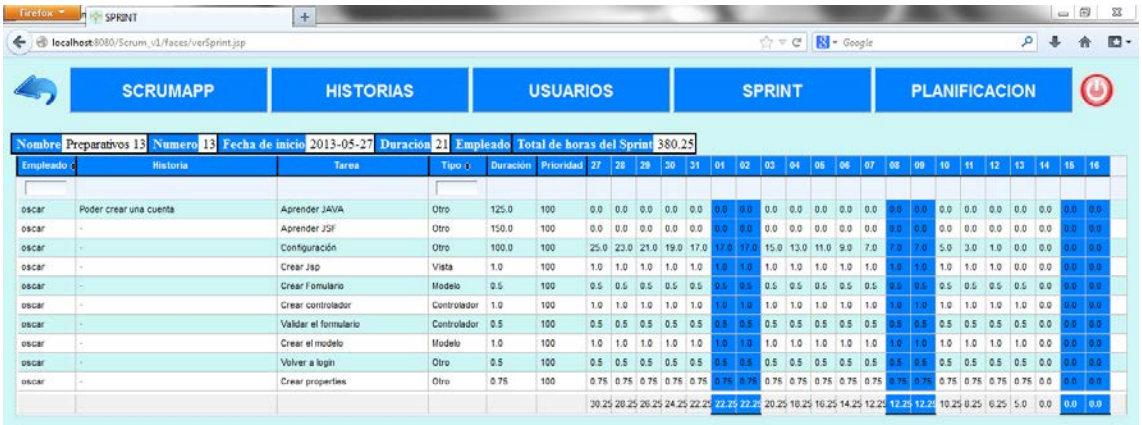


Ilustración 20: Sprint - Poder crearse una cuenta

El siguiente Sprint es: poder crear y ver proyectos.

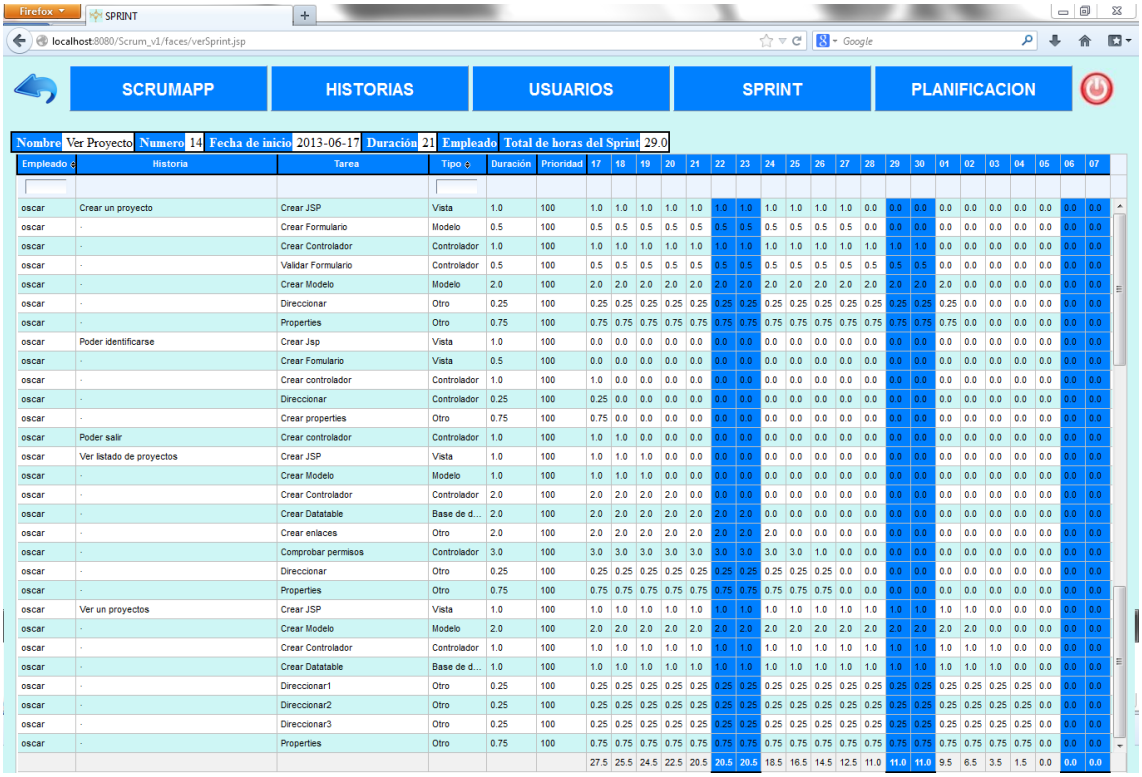


Ilustración 21: Sprint - Poder crear y ver proyectos

El siguiente Sprint es: Gestión de proyectos.

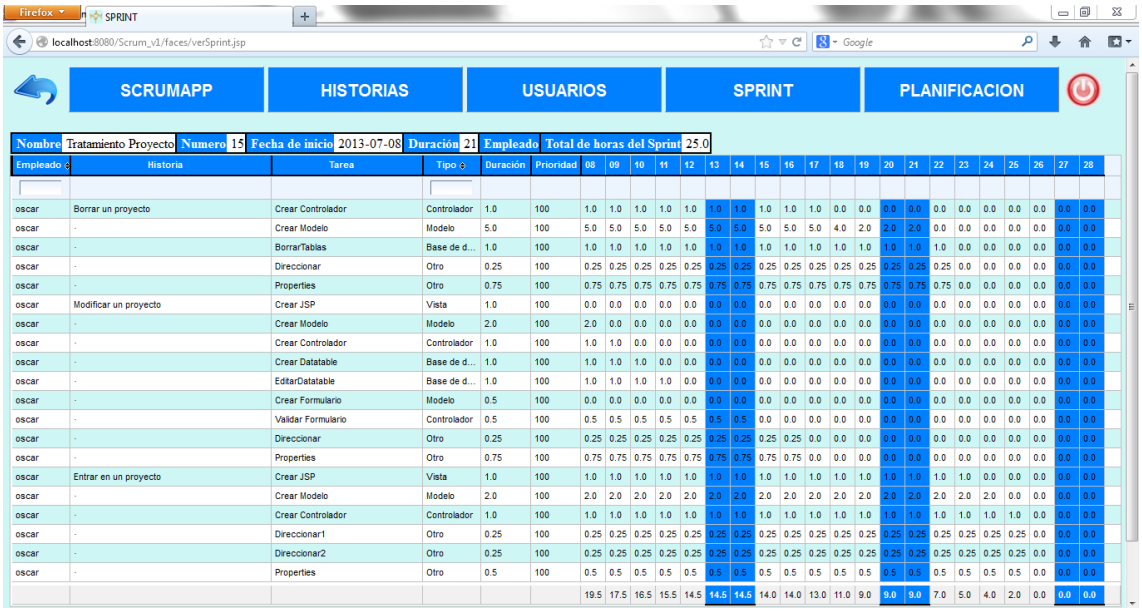
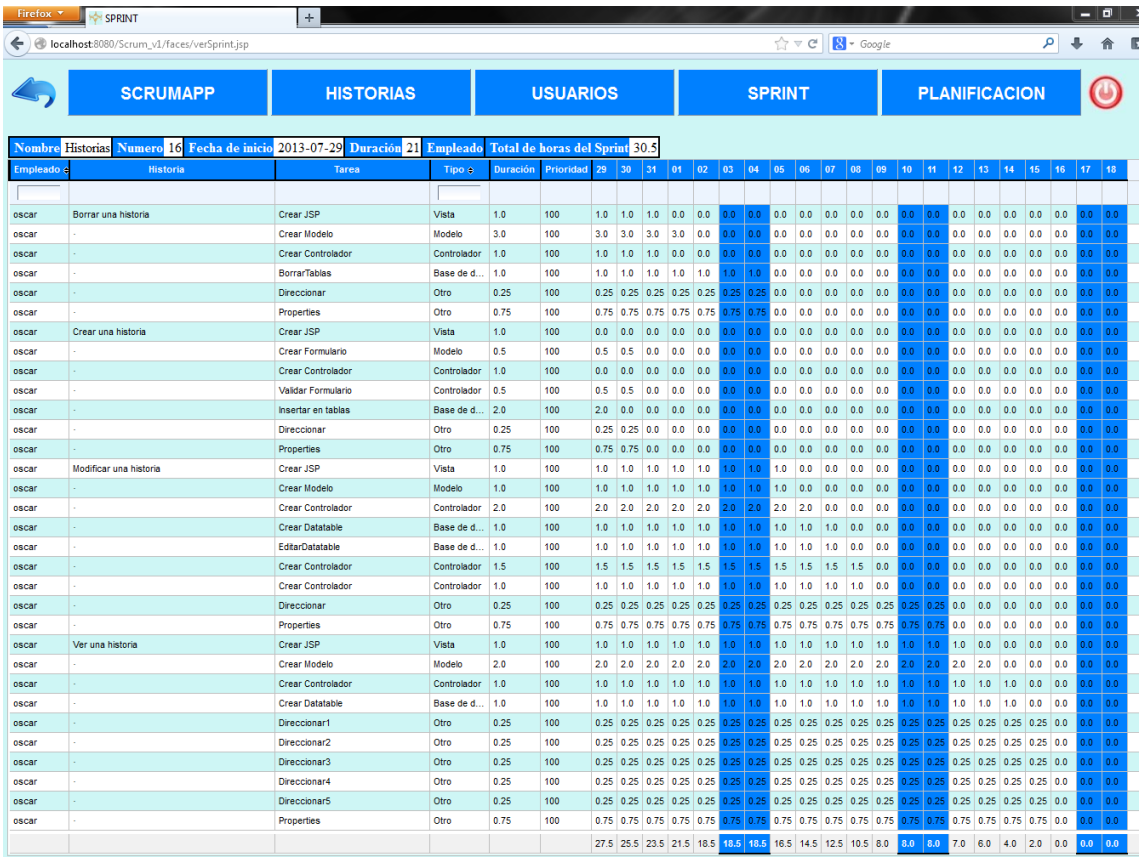


Ilustración 22: Sprint - Gestión de proyectos

El siguiente Sprint es: Gestión de historias.



Empleado	Historia	Tarea	Tipo	Duración	Prioridad	29	30	31	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18
oscar	Borrar una historia	Crear JSP	Vista	1.0	100	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Modelo	Modelo	3.0	100	3.0	3.0	3.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Controlador	Controlador	1.0	100	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	BorrarTablas	Base de d...	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Direccionar	Otro	0.25	100	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Properties	Otro	0.75	100	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	Crear una historia	Crear JSP	Vista	1.0	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Formulario	Modelo	0.5	100	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Controlador	Controlador	1.0	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Validar Formulario	Controlador	0.5	100	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Insertar en tablas	Base de d...	2.0	100	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Direccionar	Otro	0.25	100	0.25	0.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Properties	Otro	0.75	100	0.75	0.75	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	Modificar una historia	Crear JSP	Vista	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Modelo	Modelo	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Controlador	Controlador	2.0	100	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Datatable	Base de d...	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	EditarDatatable	Base de d...	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Controlador	Controlador	1.5	100	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Controlador	Controlador	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Direccionar	Otro	0.25	100	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Properties	Otro	0.75	100	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	Ver una historia	Crear JSP	Vista	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Modelo	Modelo	2.0	100	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Controlador	Controlador	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
oscar	-	Crear Datatable	Base de d...	1.0	100	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
oscar	-	Direccionar1	Otro	0.25	100	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.0	0.0	0.0
oscar	-	Direccionar2	Otro	0.25	100	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.0	0.0	0.0
oscar	-	Direccionar3	Otro	0.25	100	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.0	0.0	0.0
oscar	-	Direccionar4	Otro	0.25	100	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.0	0.0	0.0
oscar	-	Direccionar5	Otro	0.25	100	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.0	0.0	0.0
oscar	-	Properties	Otro	0.75	100	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.0	0.0	0.0
						27.5	25.5	23.5	21.5	18.5	18.5	18.5	16.5	14.5	12.5	10.5	8.0	8.0	8.0	7.0	6.0	4.0	2.0	0.0	0.0	0.0

Ilustración 23: Sprint - Gestión de historias

El siguiente Sprint es: Gestión de Tareas.

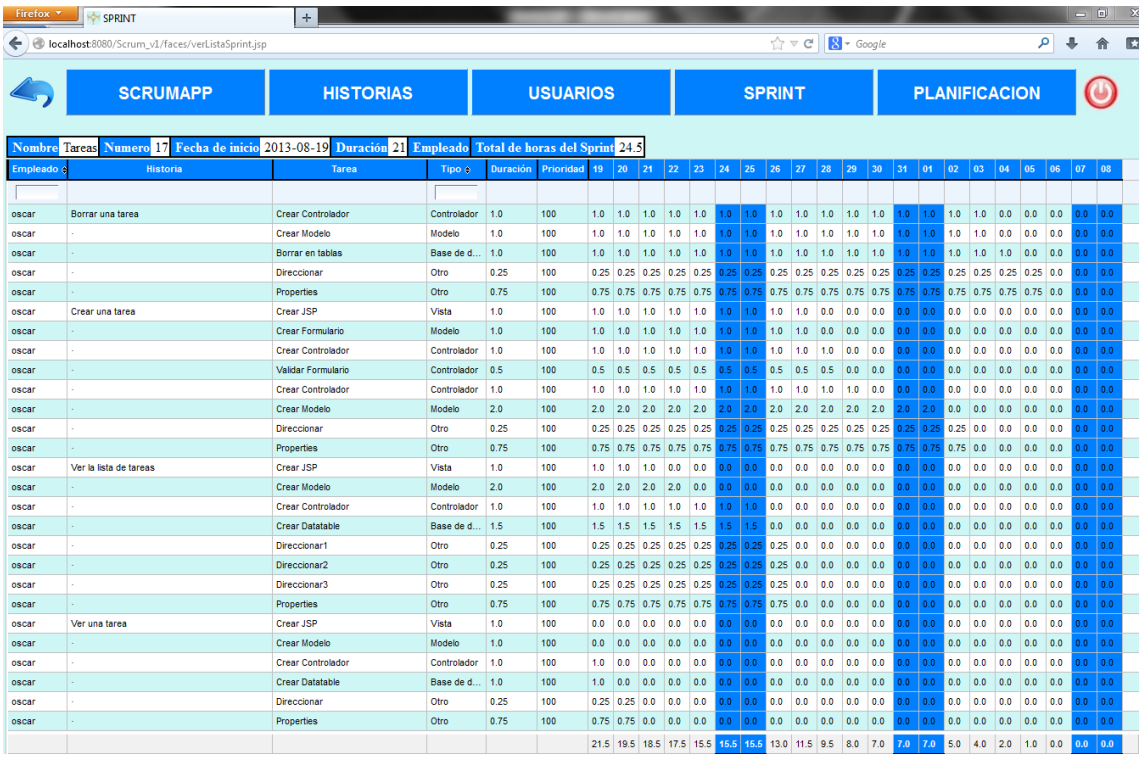


Ilustración 24: Sprint - Gestión de tareas

El siguiente Sprint es: Poder crear un Sprint.

Ilustración 25: Sprint - Poder crear un Sprint

El siguiente Sprint es: Gestión de Sprint.

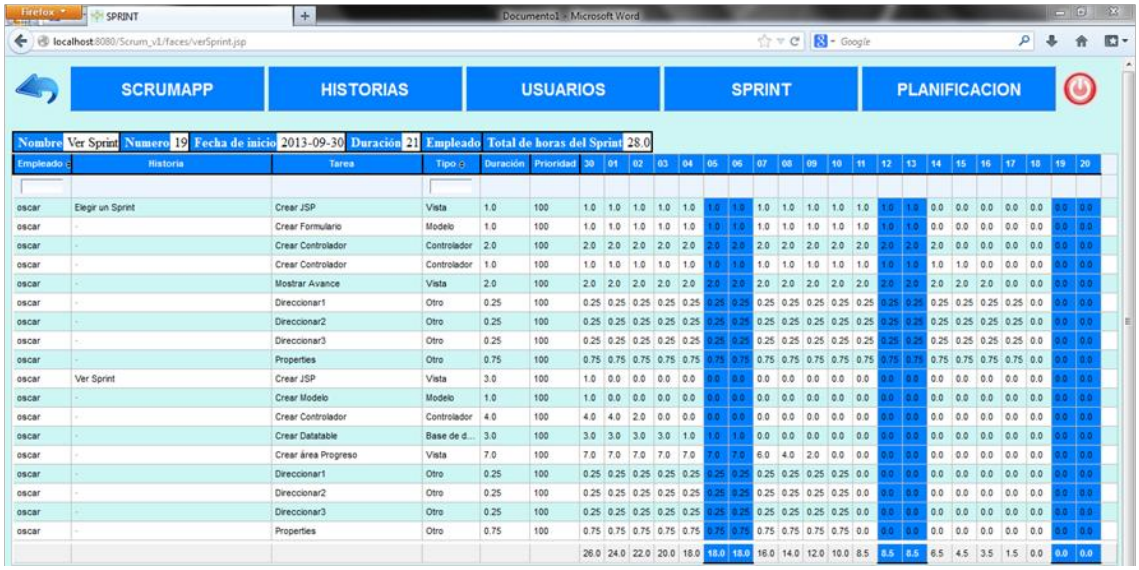


Ilustración 26: Sprint - Gestión de Sprint

El siguiente Sprint es: Gestión empleados.

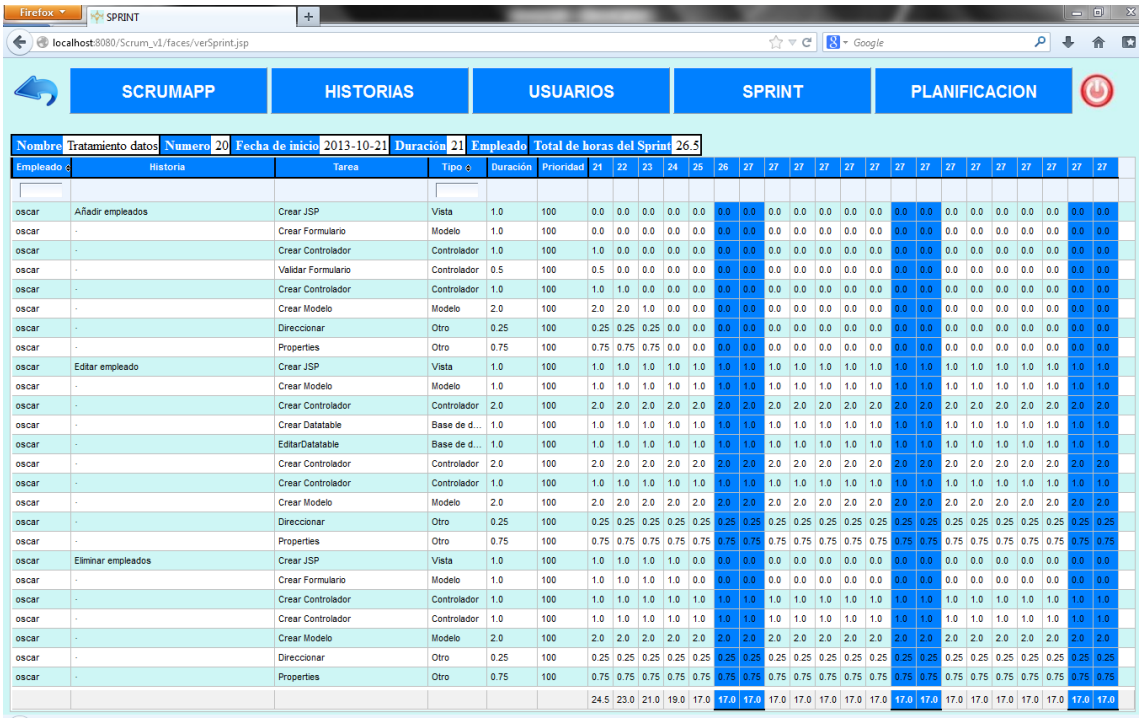


Ilustración 27: Sprint - Gestión empleados

El siguiente Sprint es: Crear informes.

Ilustración 28: Sprint - Crear informes

El siguiente Sprint es: Product Backlog.

Ilustración 29: Sprint - Product Backlog

5.2.4. Tablas

En cuanto a las tablas de base de datos necesarias para la realización de la aplicación Web se puede definir los siguientes aspectos:

- **Diagrama Entidad/Relación**

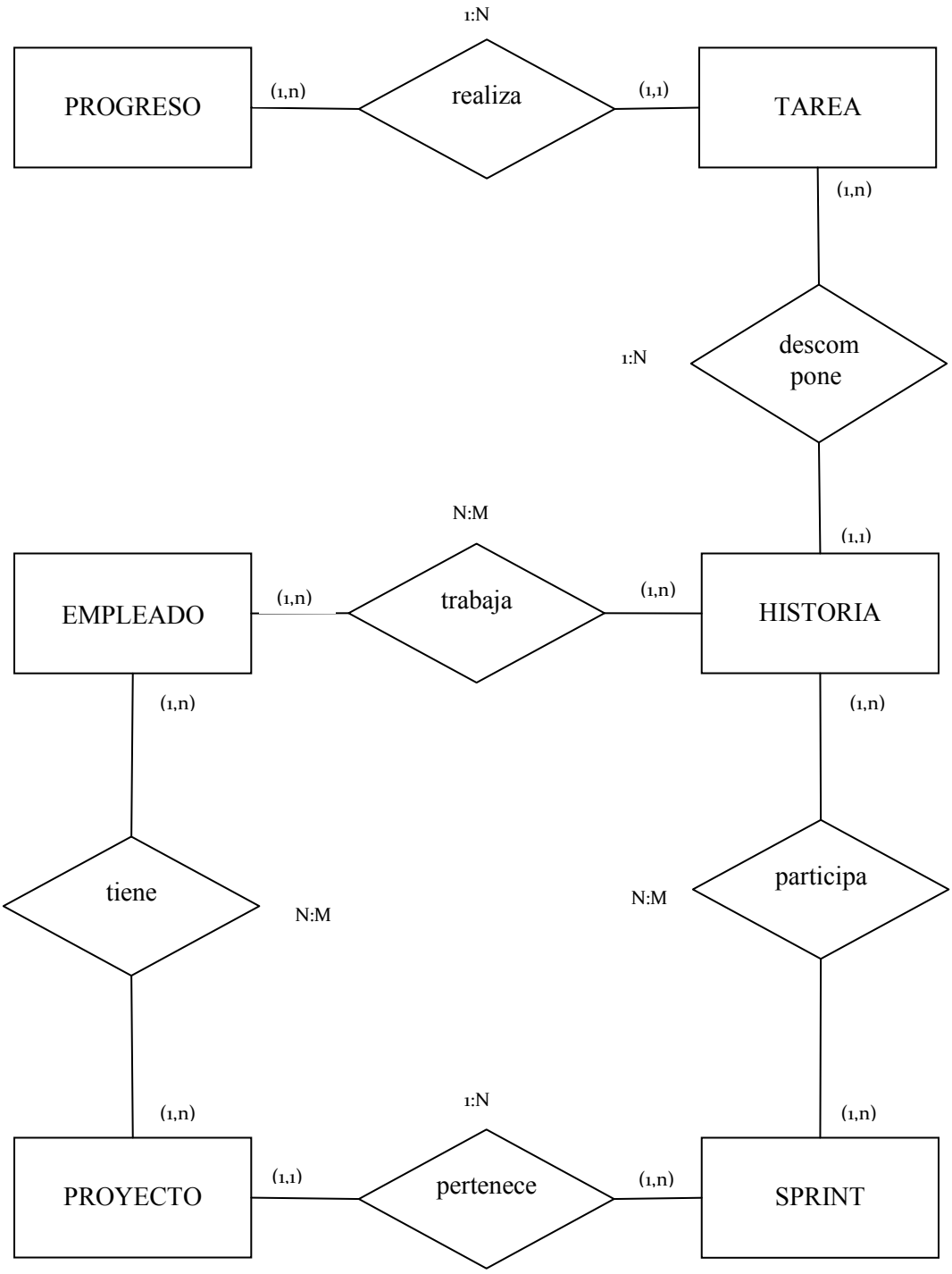


Ilustración 30: Diagrama Entidad-Relación

- **Definición de entidades**

A continuación se define las entidades conceptuales de la base de datos del sistema y la relación que existe entre cada una de ellas. Esta parte referente al diseño de la base de datos es extremadamente importante.

PROYECTO: contiene todos los datos generales relativos al proyecto, por ejemplo, fechas de inicio y fin, ScrumMaster, descripción, etc.

EMPLEADO: esta tabla se refiere a cada persona que está dada de alta en el sistema, ya sea Product Owner, ScrumMaster o desarrollador. Esta tabla contendrá datos generales tales como identificación del empleado, datos personales, etc.

HISTORIA: contendrá los datos relativos a todas las historias dadas de alta en el sistema. Se identificará por su código, que deberá ser único y estará relacionado con el proyecto.

TAREA: es una parte de la descomposición de una historia. Contendrá datos del tipo de nombre, descripción, comentarios, duración, etc. Se diferenciará del resto por su código.

SPRINT: aquí se guarda la información relativa a un sprint, como por ejemplo el nombre, las fechas de inicio y fin, el proyecto al que pertenece, etc.

PROGRESO: contiene los datos relativos al progreso de los proyectos. Servirá para saber que progreso se ha realizado de todas las tareas que están en el Sprint.

PROYEMPL: contiene la relación de empleados que trabajan en un proyecto.

HISTSPRINT: contiene las historias que hay en un Sprint.

TRABAJA: es una relación entre empleados, sprint e historias. Se crea esta tabla para que sea menos costoso mostrar los datos de los Sprints.

GRAFICA: contiene las gráficas de cada sprint y proyecto.

- **Definición de relaciones**

Realiza (PROGRESO-TAREA): relación 1:N. la relación entre estas dos tablas significa que cada tarea, cada día tendrá una nueva entrada en la tabla PROGRESO. Esto servirá para que podamos saber que tareas se han realizado en un día concreto.

Descompone (TAREA-HISTORIA): relación 1:N en la que una historia contiene 1 o varias tareas pero una tarea sólo puede pertenecer a una historia.

Trabaja (EMPLEADO-HISTORIA): relación 1:N en la que un empleado puede trabajar en una o varias historias y en una historia pueden trabajar uno o varios empleados. En principio la aplicación está diseñada para que en una historia sólo pueda trabajar un empleado pero se prevé que en un futuro se pueda hacer esta ampliación. Mientras utilizamos esta tabla para clarificar ciertos procesos.

Tiene (EMPLEADO-PROYECTO): relación N:M en la que un empleado puede participar en varios proyectos y un proyecto puede tener varios empleados. Para mayor comodidad se ha llamado a la tabla PROYEMPL. El dato más significativo de esta tabla será “rol”. Es el rol que tiene el empleado en ese proyecto, ya que en otro proyecto puede tener otro rol.

Participa (HISTORIA-SPRINT): relación N:M en la que una historia puede estar en uno o varios Sprints y un sprint puede contener una o varias historias. Para mayor comodidad se ha llamado a la tabla HISTSPRINT.

Pertenece (PROYECTO-SPRINT): relación 1:N en la que un proyecto puede tener uno o varios Sprints pero un sprint solo puede pertenecer a un proyecto.

Supuestos semánticos no reflejados.

La duración del Sprint no puede ser mayor que la diferencia entre fecha de fin menos fecha de inicio.

La suma de las duraciones de todas las tareas de una historia tiene que ser igual a la duración de la historia.

- **Modelo relacional**

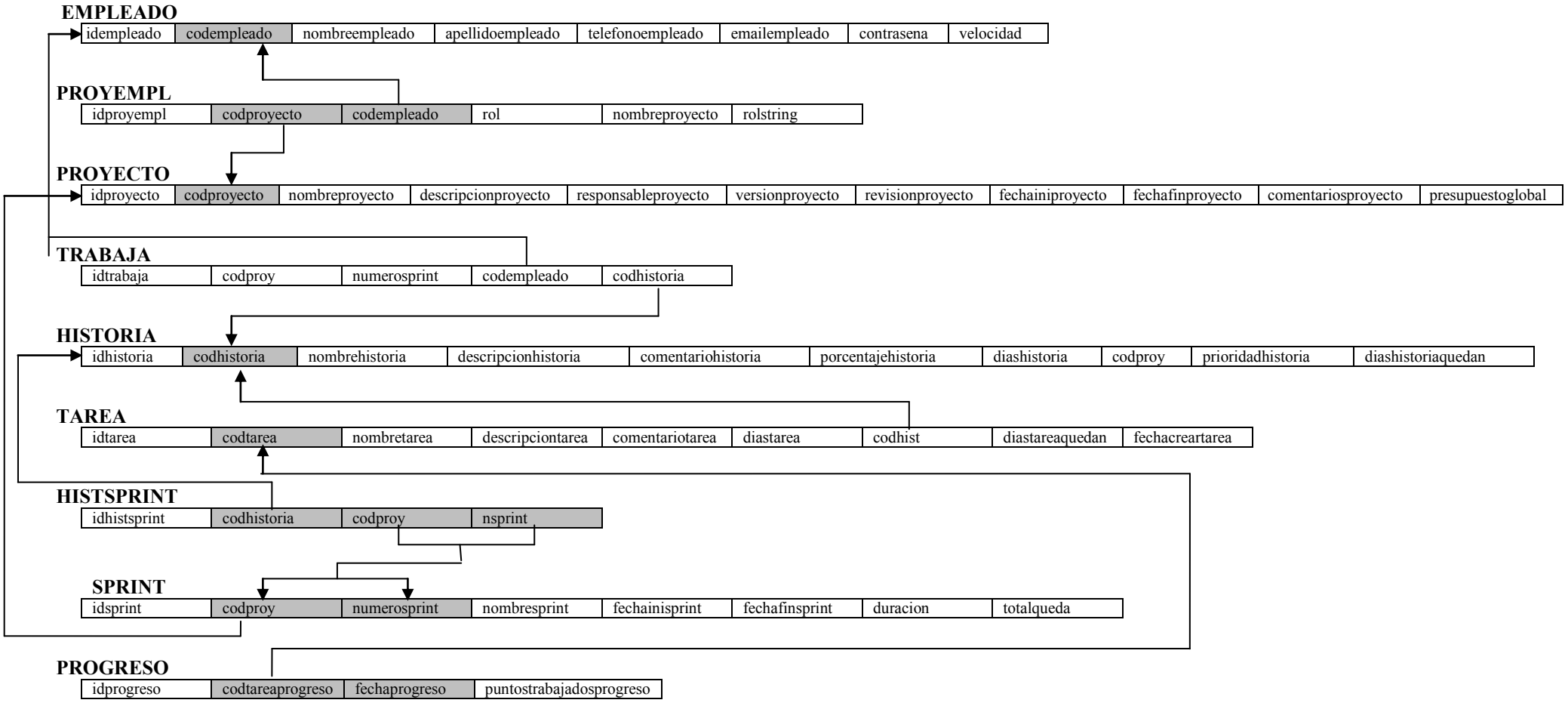


Tabla 43: Modelo Relacional

5.3. Estimación

La estimación de este proyecto es algo compleja debido a varios motivos.

El desconocimiento de la tecnología a utilizar (JavaServer Faces) es una de las cuestiones principales de la estimación, ya que se debe de aprender la tecnología antes de empezar con el proyecto. En este caso se dio una estimación de 525 horas, repartidas en 325 horas para aprender JAVA y 200 horas para aprender JSF.

Otro punto de la estimación es la configuración del entorno de trabajo. En este caso se utilizó Tomcat y MySQL. En la configuración del entorno de trabajo, se estimó la cantidad de 200 horas.

El resto de estimaciones de las tareas están basadas en las tareas en las que se divide cada historia. Poniendo unos baremos: 0.25 horas por cada redirección; 1, 2 ó 3 horas para crear módulos acceso a tablas; etc. Estos valores son orientativos, ya que cada proceso requiere un estudio particular.

Con estos cálculos, la estimación del proyecto es: 1050 horas. Esta duración es la duración del análisis, diseño, implementación y pruebas de la aplicación Web. A esto hay que añadir los estudios previos de Scrum (100 horas). Por tanto la estimación total del proyecto es de **1150** horas. La duración final es de 27 meses con un porcentaje de dedicación del 25%.

Concepto	Tiempo
Formación JAVA	325 horas
Formación JSF	200 horas
Configuración	200 horas
Formación Scrum	100 horas
Desarrollo	325 horas
TOTAL	**1150 horas**

Tabla 44: Estimación

6. Implementación

6.1. Fases

Para implementar el proyecto, se ha realizado un estudio de la tecnología que se ha utilizado. También se ha realizado un estudio de la metodología SCRUM para poder realizar el proyecto correctamente.

Tras esto, se realizó un análisis de los requisitos según la metodología SCRUM y se plasmaron en un documento EXCEL destinado a ello. En él se procedió a descomponer las historias en tareas. Tras esto, se estimaron las duraciones de las tareas y de las historias. Todo esto seguía plasmado en la EXCEL.

A partir de entonces se crearon los primeros Sprints. Estos sólo contenían las horas de estudio necesarias para realizar el proyecto. Una vez transcurridas esas horas se procedió a continuar con el resto de historias del proyecto.

Una vez realizado el análisis, se procede a implementar el proyecto. Para ello, se configura primero la base de datos a utilizar (MySQL), el servidor (Tomcat), el entorno de trabajo (Eclipse) y las librerías (JSF).

Después de implementar cada historia se realizan pruebas unitarias de la misma, con la finalidad de confirmar que funciona correctamente para el Sprint.

Según se va avanzando en la implementación y cuando se tienen varias historias relacionadas entre sí, se procede a realizar pequeñas pruebas de sistema en las que se corrobora que todas las partes están correctamente acopladas.

Cuando se termina el proyecto se realizan unas pruebas de sistema para comprobar que el proyecto está perfectamente terminado. El resultado de esas pruebas se puede observar con el Análisis completo de este proyecto en la herramienta creada.

Una vez terminada la implementación se crea este documento.

6.2. Comentarios

El proyecto está creado en el lenguaje de programación JAVA, según el Modelo-Vista-Controlador y la tecnología JavaServer Faces. Por ello la implementación se puede dividir en tres partes fundamentales:

- Modelo: clases java que interactúan con la base de datos.

Un ejemplo del módulo modelo es:

```
package modelo;

public class Historia {

    private Integer idhistoria;
    private String codhistoria;
    private String nombrehistoria;
    private String descripcionhistoria;
    private String comentariohistoria;
    private Integer porcentajehistoria;
    private double diashistoria;
    private String codproy;
    private Integer prioridadhistoria;
    private double diashistoriaquedan;

    public Integer getIdhistoria() {
        return idhistoria;
    }
    public void setIdhistoria(Integer idhistoria) {
        this.idhistoria = idhistoria;
    }
    public String getCodhistoria() {
        return codhistoria;
    }
    public void setCodhistoria(String codhistoria) {
        this.codhistoria = codhistoria;
    }
    public String getNombrehistoria() {
        return nombrehistoria;
    }
    public void setNombrehistoria(String nombrehistoria) {
        this.nombrehistoria = nombrehistoria;
    }
    public String getDescripcionhistoria() {
        return descripcionhistoria;
    }
    public void setDescripcionhistoria(String descripcionhistoria) {
        this.descripcionhistoria = descripcionhistoria;
    }
    public String getComentariohistoria() {
        return comentariohistoria;
    }
    public void setComentariohistoria(String comentariohistoria) {
        this.comentariohistoria = comentariohistoria;
    }
    public Integer getPorcentajehistoria() {
        return porcentajehistoria;
    }
    public void setPorcentajehistoria(Integer porcentajehistoria) {
        this.porcentajehistoria = porcentajehistoria;
    }
    public Double getDiashistoria() {
        return diashistoria;
    }
    public void setDiashistoria(Double diashistoria) {
        this.diashistoria = diashistoria;
```

```
    }
    public String getCodproy() {
        return codproy;
    }
    public void setCodproy(String codproy) {
        this.codproy = codproy;
    }
    public Integer getPrioridadhistoria() {
        return prioridadhistoria;
    }
    public void setPrioridadhistoria(Integer prioridadhistoria) {
        this.prioridadhistoria = prioridadhistoria;
    }
    public Double getdiashistoriaquedan() {
        return diashistoriaquedan;
    }
    public void setdiashistoriaquedan(Double diashistoriaquedan) {
        this.diashistoriaquedan = diashistoriaquedan;
    }

}
```

- Vista: páginas .jsp que muestran la aplicación. Estas páginas soportan la internacionalización, es decir, se pueden ver en varios idiomas según el idioma seleccionado en el navegador.

Un ejemplo del módulo vista es el Product Backlog:

```
<%--
    Document   : gestionHistorias
    Author     : oscar
--%>


<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@page import = "javax.faces.context.FacesContext" %>
<%@page import = "java.util.LinkedList"%>

<html>
   <f:view>
      <h:form>
         <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
            <title><h:outputText
    value="#{bundle.gesHistCabecera}"></h:outputText></title>
         </head>
         <body BGCOLOR="#FFFF99">
            <table>
               <tr>
                  <td><A  HREF="gestionProyecto.jsp"><IMG  ALT="Gestion  de  proyectos"
    SRC="imagenes/ScrumApp.gif"></A></td>
                  <td><A  HREF="gestionHistorias.jsp"><IMG  ALT="Gestion  de  historias"
    SRC="imagenes/GestionHistorias.gif"></A></td>
                  <td><h:commandLink
    action="#{empleadomanager.mostrarEmpleado}"><h:graphicImage
    value="imagenes/GestionUsuarios.gif"/></h:commandLink></td>
                  <td><h:commandLink
    action="#{sprintManager.mostrarSprint}"><h:graphicImage
    value="imagenes/Sprint.gif"/></h:commandLink></td>
                  <td><A     HREF="productBacklog.jsp"><IMG     ALT="Product     Backlog"
    SRC="imagenes/Planificacion.gif"></A></td>
               </tr>
            </table>
            <center>
               <h3><h:outputText value="#{bundle.gesHistTitulo}"></h:outputText></h3>
               <h:messages style="color: red" showDetail="true"/>
               <h:dataTable    id="table"    value="#{listarHistoria.list}"    var="his"
    style="width: 650px">
```

```
                <h:column headerClass="columnheader">
                   <f:facet name="header">
                      <b><h:outputText  value="#{bundle.gesHistCodHist}"  style="text-
    align: left"></h:outputText></b>
                   </f:facet>
                   <h:commandLink action="#{historiaManager.recogerHistoria}" id="ir">
                      <f:param name="codHistoria" value="#{his.codhistoria}">
                         <CENTER><h:outputText  id="codhis"  value="#{his.codhistoria}"
    style="background-image:       imagenes/nota.gif"       styleClass="background-image:
    url=imagenes/nota.gif"/></CENTER>
                      </f:param>
                   </h:commandLink>
                </h:column>
                <h:column headerClass="columnHeader">
                   <f:facet name="header">
                      <b><h:outputText
    value="#{bundle.gesHistNombHist}"></h:outputText></b>
                   </f:facet>
                   <CENTER><h:outputText                                    id="nombpro"
    value="#{his.nombrehistoria}"/></CENTER>
                </h:column>
             </h:dataTable>
             <br />
             <h:inputText id="codigohistoria" value="#{historiaManager.codhistoria}"/>
             <h:commandButton                              value="#{bundle.gesHistVer}"
    action="#{historiaManager.verHistoria}"/>
             <hr>
             <h:commandButton                          value="#{bundle.gesHistCrear}"
    action="#{historiaManager.crearHistoria1}"/>
             <h:commandButton                          value="#{bundle.gesHistSalir}"
    action="#{salirSesion.logout}" />
          </center>
       </body>
     </h:form>
   </f:view>
</html>
```

- Controlador: son las clases java que relacionan los datos de las tablas con las páginas .jsp. Estas clases tratan los datos obtenidos mediante llamadas a las tablas y los transforman en los datos necesarios para la salida por pantalla.

Ejemplos de controladores

```
package manager;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

import com.sun.org.apache.bcel.internal.generic.LCONST;

import modelo.Empleado;
import modelo.Historia;
import modelo.Tarea;

public class TareaManager {

    public static final String TAREA_SESSION_KEY = "tarea";
```

```
private Integer idtarea;
private String codtarea;
private String nombretarea;
private String descripciontarea;
private String comentariotarea;
private Double diastarea;
private String codhist;
private Double diastareaquedan;

public Integer getIdtarea() {
    return idtarea;
}
public void setIdtarea(Integer idtarea) {
    this.idtarea = idtarea;
}
public String getCodtarea() {
    return codtarea;
}
public void setCodtarea(String codtarea) {
    this.codtarea = codtarea;
}
public String getNombretarea() {
    return nombretarea;
}
public void setNombretarea(String nombretarea) {
    this.nombretarea = nombretarea;
}
public String getDescripciontarea() {
    return descripciontarea;
}
public void setDescripciontarea(String descripciontarea) {
    this.descripciontarea = descripciontarea;
}
public String getComentariotarea() {
    return comentariotarea;
}
public void setComentariotarea(String comentariotarea) {
    this.comentariotarea = comentariotarea;
}
public Double getDiastarea() {
    return diastarea;
}
public void setDiastarea(Double diastarea) {
    this.diastarea = diastarea;
}
public String getCodhist() {
    return codhist;
}
public void setCodhist(String codhist) {
    this.codhist = codhist;
}
public Double getDiastareaquedan() {
    return diastareaquedan;
}
public void setDiastareaquedan(Double diastareaquedan) {
    this.diastareaquedan = diastareaquedan;
}


//---------------------------------------------------------------------------
// METODOS PUBLICOS
//---------------------------------------------------------------------------

// Metodo que comprueba permisos para ver si se puede crear
// una tarea nueva. Solo podrán crear tareas nuevas el
// Desarrollador y el ScrumMaster. Si la tarea es de una historia que ya
// tiene tareas, tienen que ser del mismo desarrollador o del SM
public String crearTarea1() throws SQLException{

    // Recojo las variables de entorno
    FacesContext context = FacesContext.getCurrentInstance();
    Object pro = new Object();
    pro                                                                    =
context.getExternalContext().getSessionMap().get(manager.ProyectoManager.PROYECTO_SE
SSION_KEY);
    String nombrePro = pro.toString();
```

```
    Object emp = new Object();
    emp                                                                                    =
context.getExternalContext().getSessionMap().get(manager.EmpleadoManager.USER_SESSIO
N_KEY);
    String nombreEmp = emp.toString();
    Object his = new Object();
    his                                                                                    =
context.getExternalContext().getSessionMap().get(manager.HistoriaManager.HISTORIA_SE
SSION_KEY);
    String nombreHis = his.toString();

    // Busco el rol de este empleado en este proyecto
    int r = buscarol(nombrePro, nombreEmp);
    if (r == 2) {
           FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                  "Error en el acceso.",
                  "No tiene permisos para crear una tarea nueva");
           context.addMessage(null,message);
           return null;
    }
    return "crearTarea";
}

// Metodo que crea una tarea nueva para una determinada historia
public String crearTarea() throws SQLException{

    // Recojo datos de entorno
    FacesContext context = FacesContext.getCurrentInstance();
    Object emp = new Object();
    emp                                                                                    =
context.getExternalContext().getSessionMap().get(manager.EmpleadoManager.USER_SESSIO
N_KEY);
    String eaux = emp.toString();
    Object pro = new Object();
    pro                                                                                    =
context.getExternalContext().getSessionMap().get(manager.ProyectoManager.PROYECTO_SE
SSION_KEY);
    String codproyecto = pro.toString();
    Object his = new Object();
    his                                                                                    =
context.getExternalContext().getSessionMap().get(manager.HistoriaManager.HISTORIA_SE
SSION_KEY);
    String haux = his.toString();

    // Compruebo si ha caducado la sesion
    if (emp==null)
           return "login";
    if (his==null)
           return "gestionProy";

    // Compruebo que los datos son correctos
    if (codtarea == ""){
        FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
           "Falta el cogido de la tarea ",
           "Por favor escribelo.");
       context.addMessage(null, message);
       return null;
    }
    if (nombretarea == ""){
        FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
           "Falta el nombre de la tarea ",
           "Por favor escribelo.");
       context.addMessage(null, message);
       return null;
    }
    if (diastarea == 0 | diastarea == null){
           FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                  "Falta la duración de la tarea ",
                  "Por favor escribela.");
           context.addMessage(null, message);
           return null;
    }

    // Creo la conexion y hago la query para ver si el código ya
    // existe. En caso contrarioinserto en la tabla TAREA
    Connection con = crearConexion();
```

```
String sentencia = "SELECT * from tarea WHERE codtarea='"+codtarea+"'";
PreparedStatement Query3 = con.prepareStatement(sentencia);
ResultSet result1 = Query3.executeQuery();
if (!result1.next()){
        Tarea tarea = new Tarea();
        Date fecha = new Date();
        int dia = fecha.getDate();
        int mes = fecha.getMonth();
        int ano = fecha.getYear();
        java.sql.Date fechasql = new java.sql.Date(ano, mes, día);

        // Si no existe esa tarea se añade a la tabla TAREA
     sentencia = "INSERT INTO tarea(" +
                        "codtarea, " +
                        "nombretarea, " +
                        "descripciontarea, " +
                        "comentariotarea, " +
                        "diastarea, " +
                        "diastareaquedan, " +
                        "codhist, " +
                        "fechacreartarea)" +
                        " VALUES ('"+
                        codtarea+"', '"+
                        nombretarea+"', '"+
                        descripciontarea+"', '"+
                        comentariotarea+"', '"+
                        diastarea+"', '"+
                        diastarea+"', '"+
                        haux+"', '"+
                        fechasql+"')";
        Statement stat = con.createStatement();
        stat.executeUpdate(sentencia);

        // Añado la duración a la duración de su historia

        sentencia = "UPDATE historia SET " +
     "diashistoria = (diashistoria + " + diastarea + "), " +
     "diashistoriaquedan = (diashistoriaquedan + " + diastarea + ") " +
     " WHERE codhistoria = '" + haux + "'";
     stat = con.createStatement();
     stat.executeUpdate(sentencia);

     // Compruebo que esa historia no esta en TRABAJA y si no está la añado
        String sentencia2 = "SELECT * " +
        "FROM trabaja " +
        "WHERE codhistoria='"+haux+"' " +
        "AND codproy = '"+codproyecto + "' " +
        "AND codempleado = '"+eaux + "'";
        PreparedStatement Query = con.prepareStatement(sentencia2);
        ResultSet result2 = Query.executeQuery();
        if (!result2.next()){
         String sentencia3 = "INSERT INTO trabaja(" +
                                                "codhistoria, " +
                                                "codempleado, " +
                                                "numerosprint, " +
                                                "codproy)" +
                                                " VALUES ('"+
                                                haux+"', '"+
                                                eaux+"', "+
                                                -1+", '"+
                                                codproyecto+"')";
         Statement stat2 = con.createStatement();
         stat2.executeUpdate(sentencia3);
        }

     con.close();
        return "verListaTarea";
}else{
     FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
     "El codigo de la tarea '"
       + codtarea
       + "' ya existe  ",
     "Por favor cambie el codigo.");
      context.addMessage(null, message);
        con.close();
        return null;
```

```
    }
}

// Recoge la tarea metida por variable (recoge del enlace)
public String recogerTarea() throws SQLException{
    codtarea                                                                    =
FacesContext.getCurrentInstance().getExternalContext().getRequestParameterMap().get(
"codTarea").toString();
    return verTarea();
}

// Muestra una tarea en concreto
public String verTarea() throws SQLException{

    // Recojo los valores de contexto
    FacesContext context = FacesContext.getCurrentInstance();
    Object emp = new Object();
    emp                                                                         =
context.getExternalContext().getSessionMap().get(manager.EmpleadoManager.USER_SESSIO
N_KEY);
    String nombreEmp = emp.toString();
    Object pro = new Object();
    pro                                                                         =
context.getExternalContext().getSessionMap().get(manager.ProyectoManager.PROYECTO_SE
SSION_KEY);
    String nombrePro = pro.toString();
    Object                                  his                                 =
context.getExternalContext().getSessionMap().get(manager.HistoriaManager.HISTORIA_SE
SSION_KEY);
    String nombreHis = his.toString();

    if (codtarea == null){
            Object                                  tar                         =
context.getExternalContext().getSessionMap().get(manager.TareaManager.TAREA_SESSION_
KEY);
            codtarea = tar.toString();
    }

     Connection con = crearConexion();

     // Compruebo que la tarea existe
    String sentencia = "SELECT * from tarea WHERE codtarea='"+codtarea+"'";
    PreparedStatement Query3 = con.prepareStatement(sentencia);
    ResultSet result1 = Query3.executeQuery();
    if (!result1.next()){
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                  "Error en el acceso.",
                  "La tarea '" + codtarea + " ' no existe");
         context.addMessage(null,message);
         con.close();
         return null;
    }
    else{

            // Si existe recojo los valores a mostrar y retorno a verTarea
            nombretarea=result1.getString("nombretarea");
            descripciontarea=result1.getString("descripciontarea");
            comentariotarea=result1.getString("comentariotarea");
            diastarea=result1.getDouble("diastarea");
            codhist=result1.getString("codhist");
            diastareaquedan=result1.getDouble("diastareaquedan");
            if (codhist.equalsIgnoreCase(nombreHis)){

    context.getExternalContext().getSessionMap().put(TAREA_SESSION_KEY,codtarea);
                    con.close();
                    return "verTarea";
            }
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                  "Error en el acceso.",
                  "La tarea '" + codtarea + " ' no existe");
         context.addMessage(null,message);
         con.close();
         return null;
    }
}
```

```
// Metodo que comprueba permisos y que rellena la página
// de editar para mostrarla
public String editarTarea1() throws SQLException{

    // Recojo las variables de entorno
    FacesContext context = FacesContext.getCurrentInstance();
    Object emp = new Object();
    emp                                                                    =
context.getExternalContext().getSessionMap().get(manager.EmpleadoManager.USER_SESSIO
N_KEY);
    String nombreEmp = emp.toString();
    Object pro = new Object();
    pro                                                                    =
context.getExternalContext().getSessionMap().get(manager.ProyectoManager.PROYECTO_SE
SSION_KEY);
    String nombrePro = pro.toString();
    Object tar = new Object();
    tar                                                                    =
context.getExternalContext().getSessionMap().get(manager.TareaManager.TAREA_SESSION_
KEY);
    String codtar = tar.toString();

    // Busco el rol de este empleado en este proyecto
    int r = buscarol(nombrePro, nombreEmp);
    if (r == 2) {
           FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                  "Error en el acceso.",
                  "No tiene permisos para editar la tarea");
           context.addMessage(null,message);
           return null;
    }

    // Creo la conexión y busco los datos a mostrar
    Connection con = crearConexion();
    String sentencia = "SELECT * FROM tarea WHERE codtarea = '" + codtar + "'";
    PreparedStatement Query = con.prepareStatement(sentencia);
    ResultSet result = Query.executeQuery();
    if (result.next()){
           codtarea = codtar;
           nombretarea = result.getString("nombretarea");
           descripciontarea = result.getString("descripciontarea");
           comentariotarea = result.getString("comentariotarea");
           diastarea = result.getDouble("diastarea");
           codhist = result.getString("codhist");
           diastareaquedan = result.getDouble("diastareaquedan");
           con.close();
           return "editarTarea";
    }else{
           FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                  "Error en el acceso.",
                  "La tarea no existe");
        context.addMessage(null,message);
        return null;
    }
}

// Método para editar una tarea
public String editarTarea() throws SQLException{

    // Recojo las variables de contexto
    FacesContext context = FacesContext.getCurrentInstance();
    Object emp = new Object();
    emp                                                                    =
context.getExternalContext().getSessionMap().get(manager.EmpleadoManager.USER_SESSIO
N_KEY);
    String nombreEmp = emp.toString();
    Object pro = new Object();
    pro                                                                    =
context.getExternalContext().getSessionMap().get(manager.ProyectoManager.PROYECTO_SE
SSION_KEY);
    String nombrePro = pro.toString();
    Object tar = new Object();
    tar                                                                    =
context.getExternalContext().getSessionMap().get(manager.TareaManager.TAREA_SESSION_
KEY);
    String nombreTar = tar.toString();
```

```
    // Compruebo que la sesión no está caducada
    if (tar==null)
            return "login";

    // Busco el rol de este empleado en este proyecto
    int r = buscarol(nombrePro, nombreEmp);
    if (r == 2) {
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                    "Error en el acceso.",
                    "No tiene permisos para editar la tarea");
             context.addMessage(null,message);
             return null;
    }

    // Saco los datos para el caso de volver
    Connection con = crearConexion();
    String sentencia = "SELECT * FROM tarea WHERE codtarea='"+nombreTar+"'";
    PreparedStatement Query3 = con.prepareStatement(sentencia);
    ResultSet result1 = Query3.executeQuery();
    if (result1.next()){
            codtarea = nombreTar;
            nombretarea=result1.getString("nombretarea");
            diastarea=result1.getDouble("diastarea");
            codhist=result1.getString("codhist");
            diastareaquedan=result1.getDouble("diastareaquedan");

            // Edito la tarea
            sentencia = "UPDATE tarea SET " +
        "descripciontarea = '" + descripciontarea + "', " +
        "comentariotarea = '" + comentariotarea + "', " +
        "diastarea = '" + diastarea + "', " +
        "diastareaquedan = '" + diastareaquedan + "' " +
        " WHERE codtarea = '" + nombreTar + "'";
        Statement stat = con.createStatement();
        stat.executeUpdate(sentencia);

        // Compruebo que esa historia no esta en TRABAJA y si no está la añado
            String sentencia2 = "SELECT * " +
            "FROM trabaja " +
            "WHERE codhistoria='"+codhist+"' " +
            "AND codproy = '"+nombrePro + "' " +
            "AND codempleado = '"+nombreEmp + "'";
            PreparedStatement Query = con.prepareStatement(sentencia2);
            ResultSet result2 = Query.executeQuery();
            if (!result2.next()){
             String sentencia3 = "INSERT INTO trabaja(" +
                                                    "codhistoria, " +
                                                    "codempleado, " +
                                                    "numerosprint, " +
                                                    "codproy)" +
                                                    " VALUES ('"+
                                                    codhist+"', '"+
                                                    nombreEmp+"', "+
                                                    -1+", '"+
                                                    nombrePro+"')";
             Statement stat2 = con.createStatement();
             stat2.executeUpdate(sentencia3);
            }

            con.close();
        return "verTarea";
    }else{
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                    "Error en el acceso.",
                    "La tarea " + nombreTar + "no existe");
             context.addMessage(null,message);
             return null;
    }
}

// Metodo que permite borrar una tarea, siempre que no este en un Sprint
public String borrarTarea() throws SQLException{

    // Recojo las variables de contexto
    FacesContext context = FacesContext.getCurrentInstance();
```

```
    Object emp = new Object();
    emp                                                                    =
context.getExternalContext().getSessionMap().get(manager.EmpleadoManager.USER_SESSIO
N_KEY);
    String nombreEmp = emp.toString();
    Object pro = new Object();
    pro                                                                    =
context.getExternalContext().getSessionMap().get(manager.ProyectoManager.PROYECTO_SE
SSION_KEY);
    String nombrePro = pro.toString();
    Object tar = new Object();
    tar                                                                    =
context.getExternalContext().getSessionMap().get(manager.TareaManager.TAREA_SESSION_
KEY);
    String nombreTar = tar.toString();

    // Compruebo que la sesión no está caducada
    if (tar==null)
            return "login";

    // Busco el rol de este empleado en este proyecto
    int r = buscarol(nombrePro, nombreEmp);
    if (r == 2) {
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                    "Error en el acceso.",
                    "No tiene permisos para borrar la tarea");
             context.addMessage(null,message);
             return null;
    }

    // Creo la conexion
    Connection con = crearConexion();

    // Compruebo que la tarea no está planificada (no está en ningun sprint
    String sentencia = "SELECT * FROM progreso where codtareaprogreso = '" +
nombreTar + "'";
    PreparedStatement Query = con.prepareStatement(sentencia);
    ResultSet result = Query.executeQuery();
    if (result.next()){
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR,
                    "Error en el acceso.",
                    "Esta tarea ya está planificada");
             context.addMessage(null,message);
             con.close();
             return null;
    }else{

            // Borramos la tarea de (TAREA y PROGRESO)
            String sentencia1 = "DELETE FROM tarea WHERE codtarea='" + nombreTar +
"'";
            PreparedStatement query2 = con.prepareStatement(sentencia1);
            int result2 = query2.executeUpdate(sentencia1);
    }

    con.close();
    return "verListaTarea";
}

// Metodo que crea una lista con las tareas de una historia
// Saca los datos de la tabla TAREA
public static List <Tarea> verTareas(){
     List <Tarea> tarea = new ArrayList<Tarea>();
    try{

            // Recojo las variables de contexto
            FacesContext context = FacesContext.getCurrentInstance();
            Object his = new Object();
            his                                                            =
context.getExternalContext().getSessionMap().get(manager.HistoriaManager.HISTORIA_SE
SSION_KEY);
            String haux = his.toString();

            // Creo la conexión con la base de datos y hago la query
            Connection con = crearConexion();
            String sentencia = "SELECT * from tarea WHERE codhist='"+haux+"'";
            PreparedStatement Query3 = con.prepareStatement(sentencia);
```

```
                ResultSet result1 = Query3.executeQuery();
                while (result1.next()){
                        Tarea tar = new Tarea();
                        tar.setIdtarea(result1.getInt("idtarea"));
                        tar.setCodtarea(result1.getString("codtarea"));
                        tar.setCodhist(haux);
                        tar.setComentariotarea(result1.getString("comentariotarea"));
                        tar.setDescripciontarea(result1.getString("descripciontarea"));
                        tar.setDiastarea(result1.getDouble("diastarea"));
                        tar.setNombretarea(result1.getString("nombretarea"));
                        tar.setDiastareaquedan(result1.getDouble("diastareaquedan"));
                        tarea.add(tar);
                }
                con.close();
            }catch (Exception e) {
                    // TODO: handle exception
            }
            return tarea;

        }


        //---------------------------------------------------------------------------
        // METODOS PRIVADOS
        //---------------------------------------------------------------------------

        // Método en el que se crea la conexión
        public static Connection crearConexion() throws SQLException {

            try {
                    InitialContext ctx = new InitialContext();
                    DataSource source = (DataSource) ctx.lookup("java:comp/env/jdbc/mydb");
                    if (source==null) throw new SQLException("No Data Source");
                    Connection conn = source.getConnection();
                    if (conn==null) throw new SQLException("No connection");
                    else return conn;

            }catch (NamingException ex){
                    ex.printStackTrace();
            }
            return null;
            }

        //Metodo para sacar el rol de un epleado en un proyecto
        private Integer buscarol(String proy, String empl) throws SQLException{
             Connection con = crearConexion();
            String sentencia = "SELECT * from proyempl WHERE codproyecto='" + proy + "' AND
        codempleado='"+empl+"'";
            PreparedStatement Query3 = con.prepareStatement(sentencia);
            ResultSet result1 = Query3.executeQuery();
            int rol1;
            if (!result1.next()){
                    con.close();
                    return -1;
            }else
                    rol1 = result1.getInt("rol");
                    con.close();
                    return rol1;
        }

    }
```

Por otro lado están los ficheros Bundle.properties que tienen los literales que deben mostrar los .jsp. Estos ficheros son necesarios para la internacionalización y se debe tener uno por cada idioma con las traducciones de los literales. Una parte de ellos:

Bundle.properties

```
-------index.jsp-------
login=Login
```

```
ScrumApp=ScrumApp
indexIntroduzca=Por favor, introduzca su codigo de empleado y su password
indexCodigo=Código
indexPassword=Password
indexAceptar=Aceptar
indexCrearNuevo=Crear nueva cuenta

-------crearEmpleado.jsp-------
crearEmplCabecera=Crear nuevo empleado
crearEmplTitulo=Crear un nuevo empleado
crearEmplCodigo=Código de usuario:
crearEmplNombre=Nombre:
crearEmplApellido= Apellido:
crearEmplTelefono=Teléfono:
crearEmplEmail:Email:
crearEmplPassword=Password:
crearEmplPasswordv=Password(verificar):
crearEmplCrear=Crear
crearEmplVolver=Volver
```

Bundle_en.properties

```
-------index.jsp-------
login=Login
ScrumApp=ScrumApp
indexIntroduzca=Please, enter your employee code and password
indexCodigo=Employee code
indexPassword=Password
indexAceptar=Enter
indexCreateNuevo=Create new acount

-------crearEmpleado.jsp-------
crearEmplCabecera=Create new employee
crearEmplTitulo=Create a new employee
crearEmplCodigo=Employee code:
crearEmplNombre=Name:
crearEmplApellido=Surname:
crearEmplTelefono=Telephone:
crearEmplEmail:Email:
crearEmplPassword=Password:
crearEmplPasswordv=Password(verify):
crearEmplCreate=Create
crearEmplVolver=Return
```

6.3. Vistas de la aplicación

6.3.1. Inicio

Esta es la página inicial que se visualiza cuando un usuario carga en su navegador la aplicación. El usuario puede entrar en la aplicación escribiendo su usuario y contraseña, o bien, darse de alta.

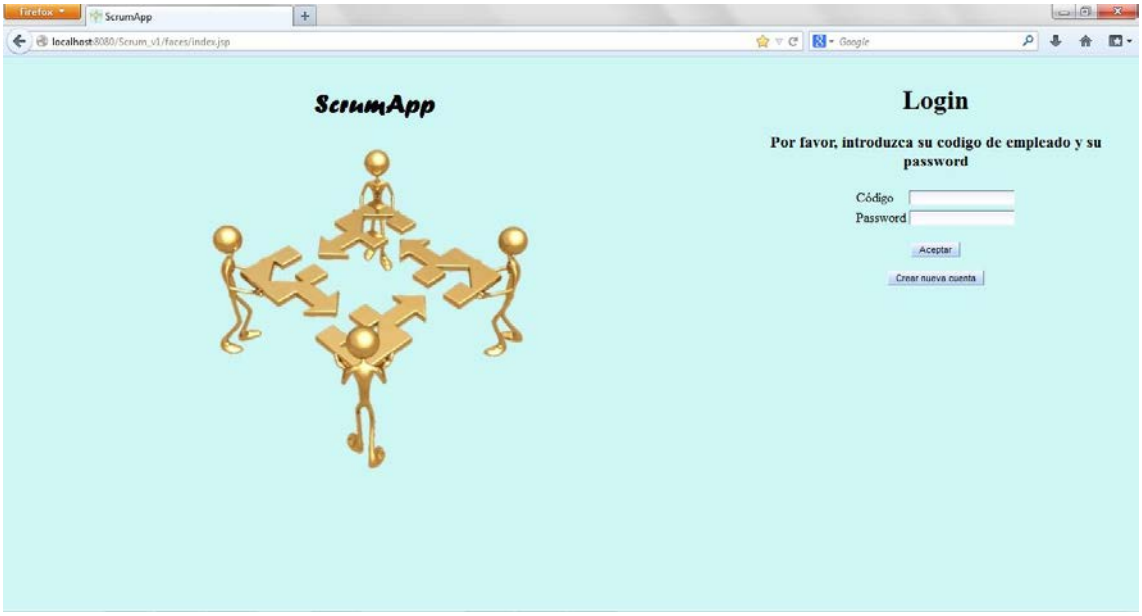


Ilustración 31: Vistas - Inicio

6.3.2. Crear una cuenta nueva

En esta pantalla podemos darnos de alta en la aplicación. Para ello debemos rellenar los datos que nos piden que después se validen. A esta pantalla sólo se puede acceder mediante la página de inicio.

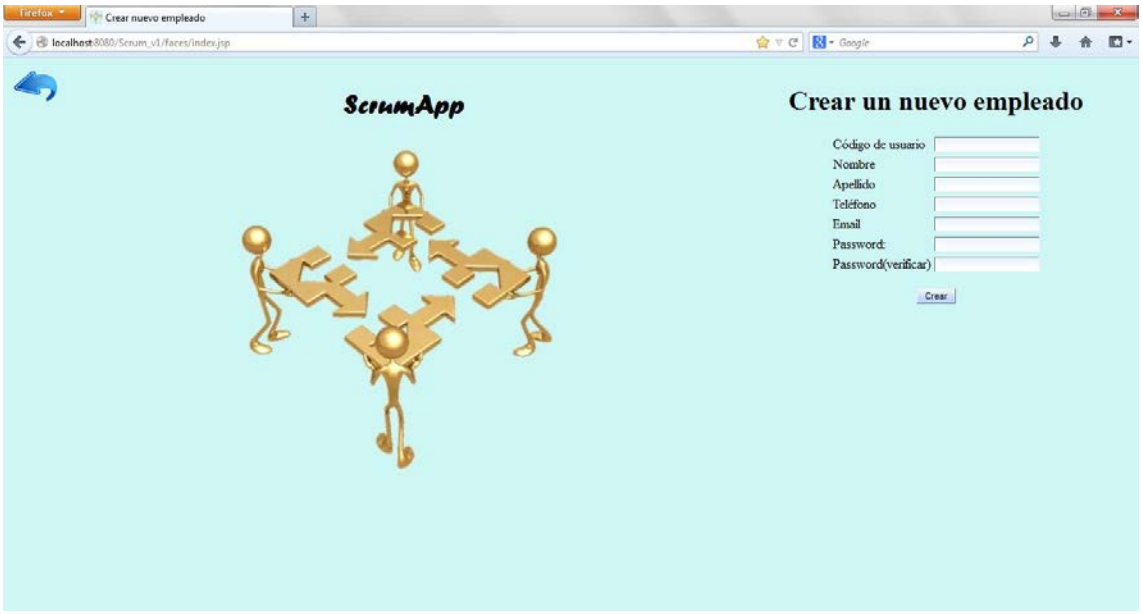


Ilustración 32: Vistas - Crear cuenta nueva

6.3.3. Gestión de proyectos

En esta pantalla podemos ver los proyectos en los que participamos, así como el rol que tenemos en cada uno de ellos. Podemos acceder a uno de ellos, podemos crear un nuevo proyecto o podemos modificar nuestros datos personales.

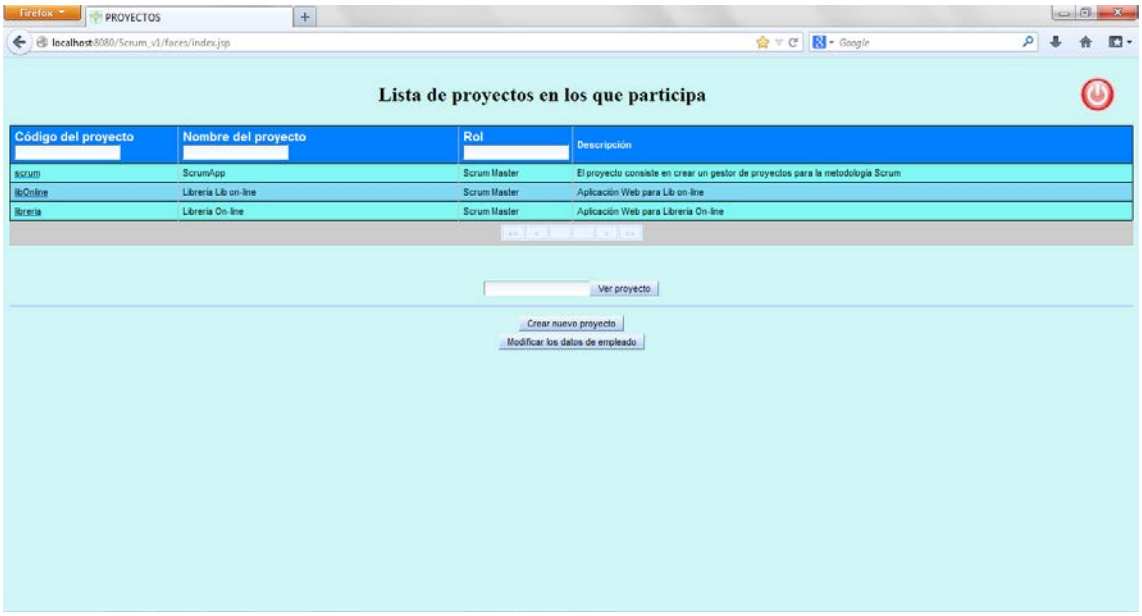


Ilustración 33: Vistas - Gestión de proyectos

6.3.4. Crear nuevo proyecto

En esta pantalla podemos crear un nuevo proyecto. Tendremos que rellenar los datos relativos al proyecto.

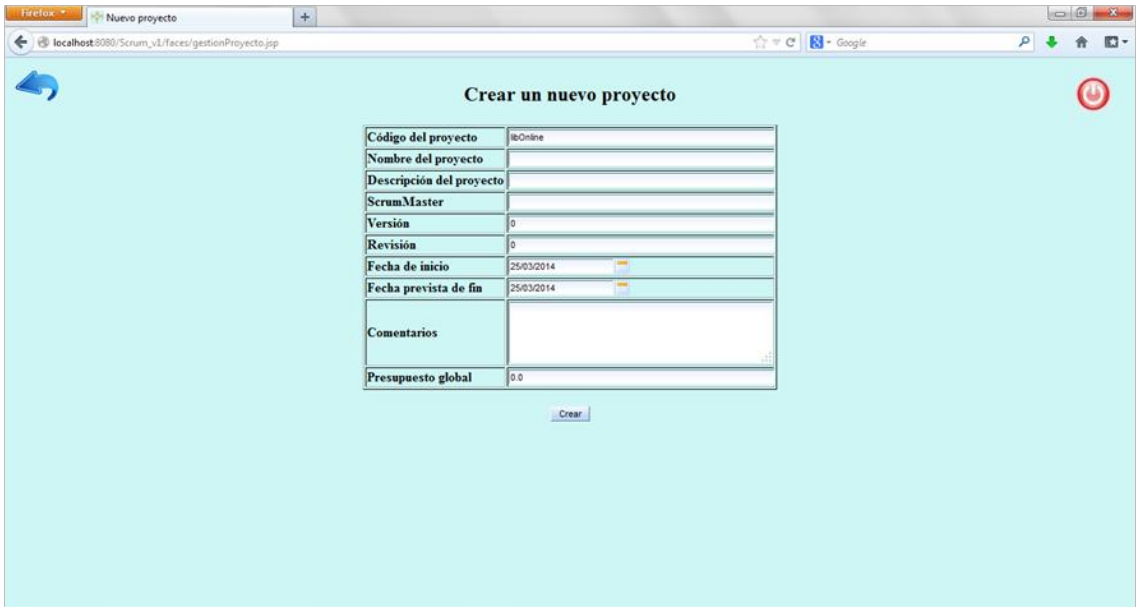


Ilustración 34: Vistas - Crear nuevo proyecto

6.3.5. Editar datos personales

En esta pantalla podemos editar nuestros datos personales. Si todavía no hemos entrado en ningún proyecto.

Ilustración 35: Vistas - Editar datos personales (I)

Si ya estamos dentro de un proyecto.

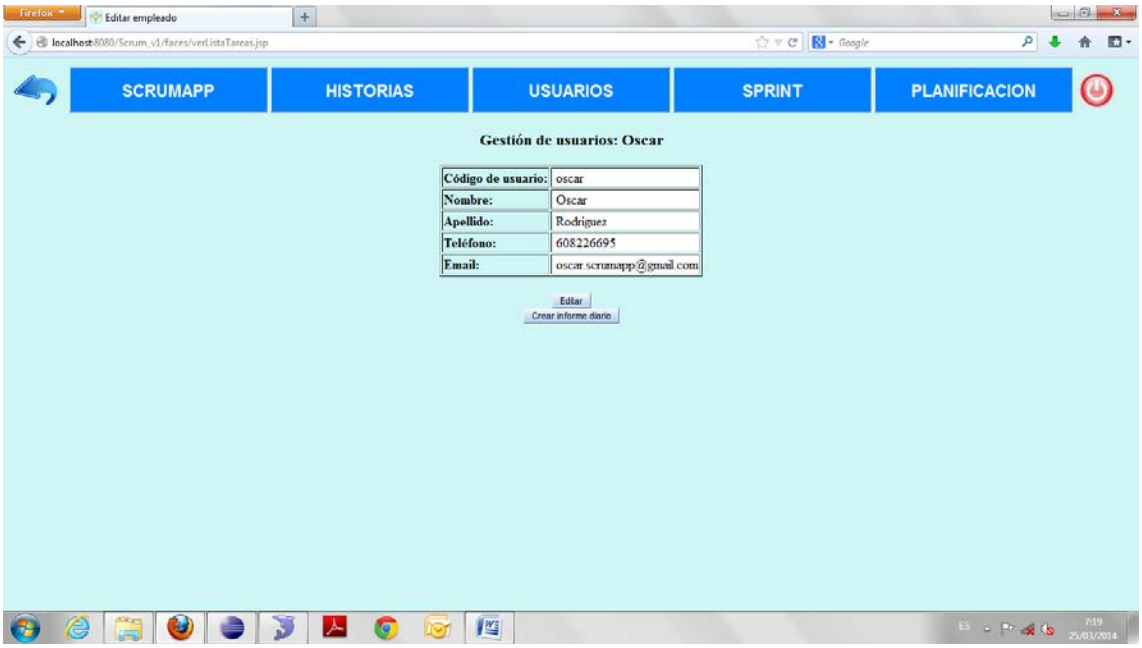


Ilustración 36: Vistas - Editar datos personales (II)

6.3.6. Ver proyecto (características principales)

En esta pantalla podemos ver las características principales del proyecto. Además podremos: modificar dichas características, borrar el proyecto (con todo lo asignado a él), añadir empleados al proyecto, borrar empleados del proyecto y entrar al proyecto. También se puede generar la documentación automática del proyecto. Siempre según permisos.

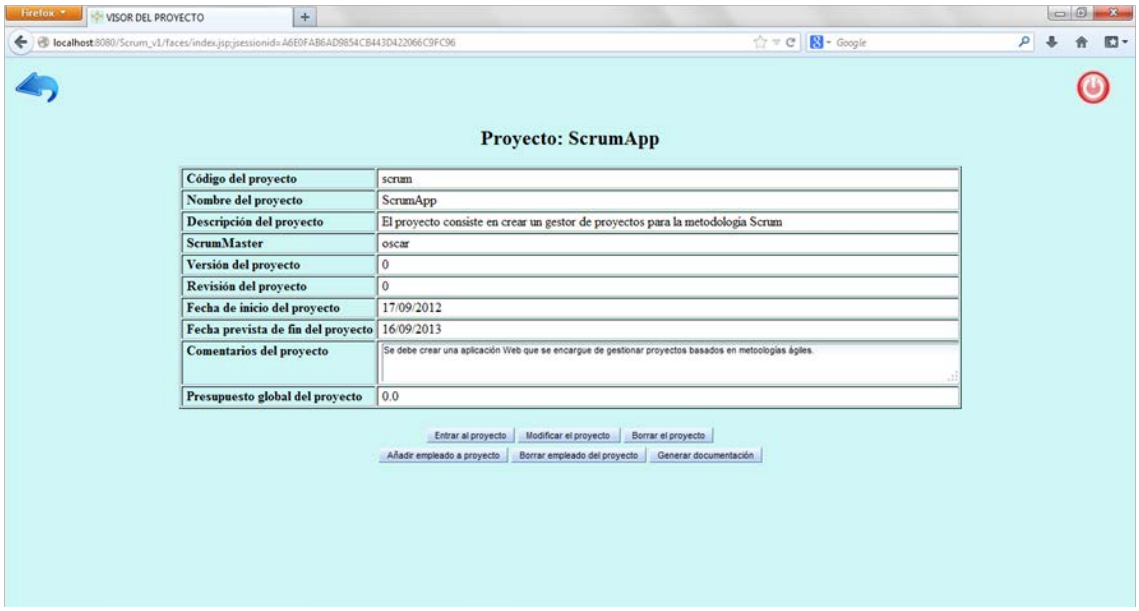


Ilustración 37: Vistas - Ver proyecto

6.3.7. Modificar proyecto (características principales)

En esta pantalla podemos modificar las características principales del proyecto. Para ello debemos estar dados de alta como ScrumMaster.

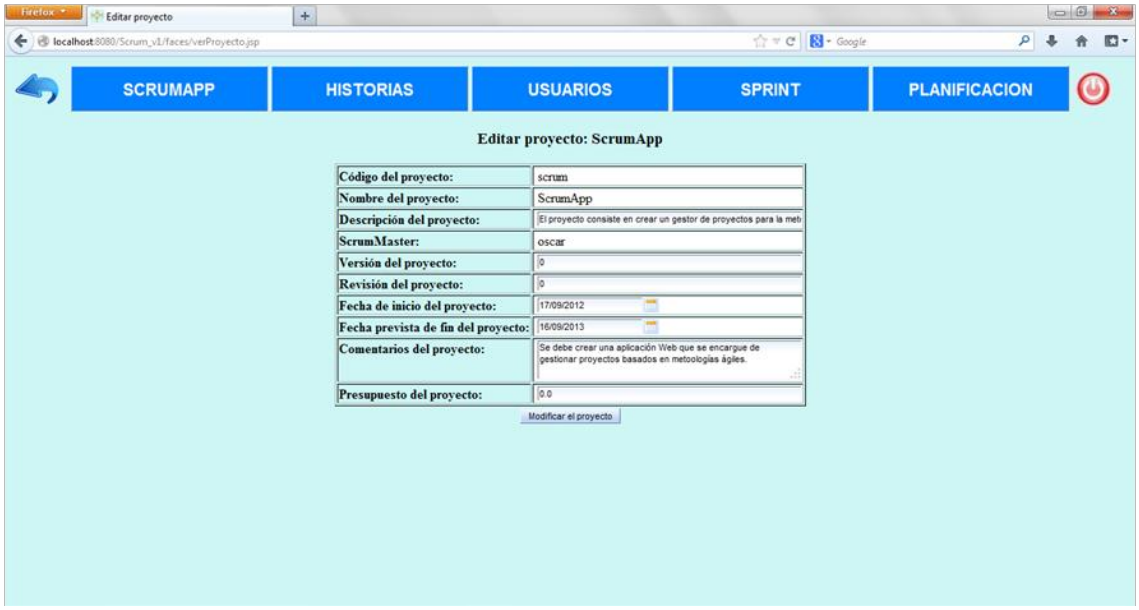


Ilustración 38: Vistas - Modificar proyecto

6.3.8. Añadir empleados al proyecto

En esta pantalla podemos añadir empleados al proyecto. Para ello debemos estar dados de alta como ScrumMaster y los empleados que se quieran añadir deben estar dados de alta en el sistema.

Ilustración 39: Vistas - Añadir empleados al proyecto

6.3.9. Eliminar empleados del proyecto

En esta pantalla podemos eliminar empleados del proyecto. Para ello debemos estar dados de alta como ScrumMaster y los empleados que se quieran eliminar deben estar dados de alta en el proyecto (aparecen en la lista).

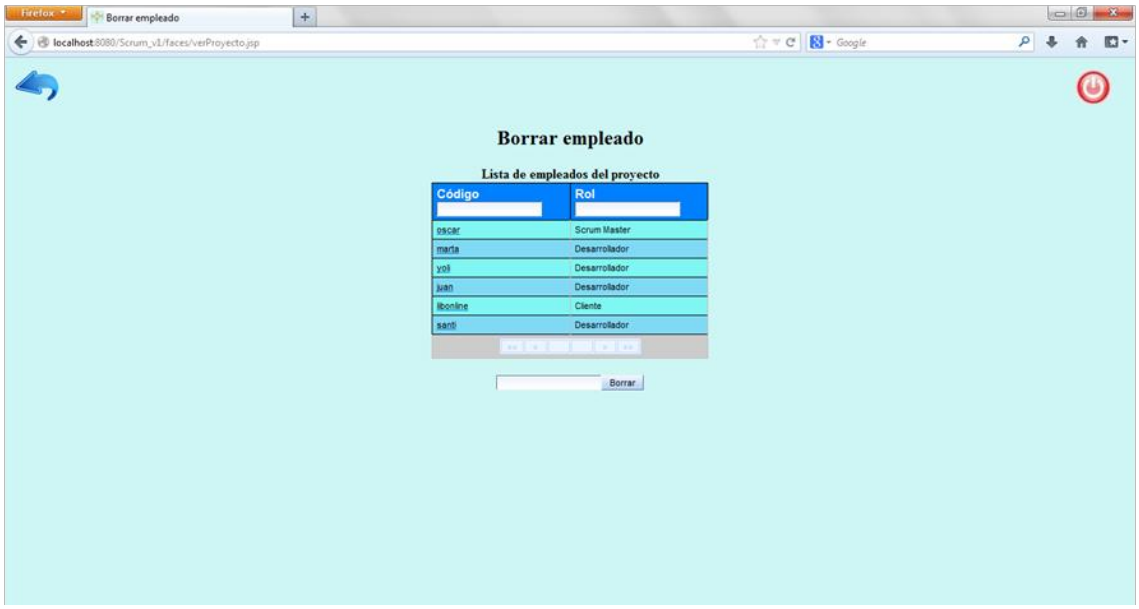


Ilustración 40: Vistas - Eliminar empleados del proyecto

6.3.10. Product Backlog (entrar al proyecto)

En esta pantalla podemos ver el Product Backlog del proyecto. Se puede acceder a la información de una historia mediante su enlace o escribiendo el código en la caja de texto que hay al final de todas las historias. Además se puede crear una historia nueva (según permisos).

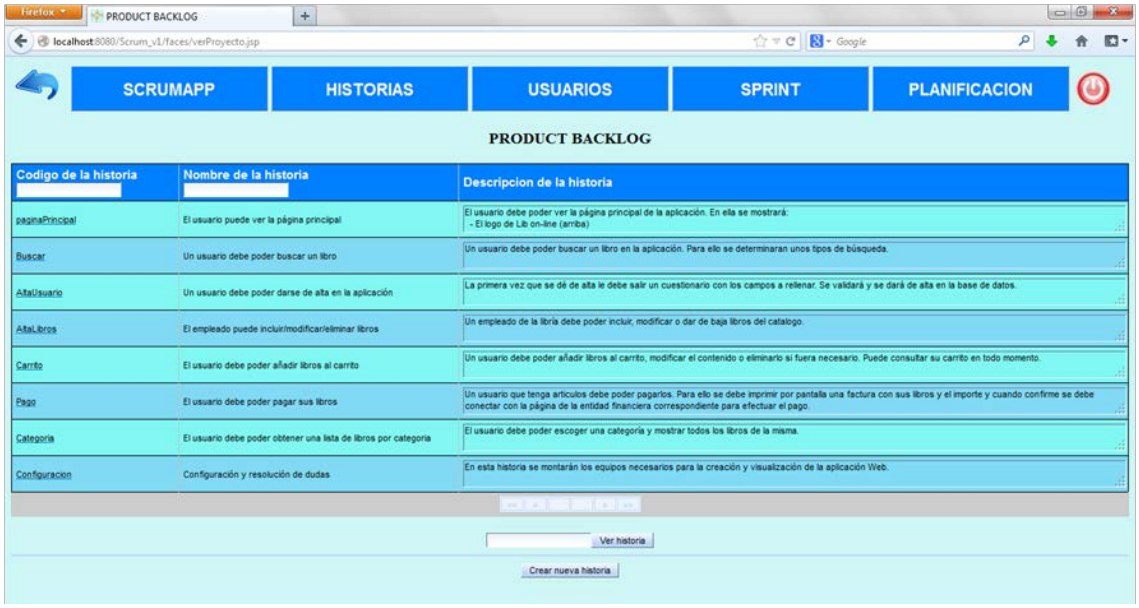


Ilustración 41: Vistas - Product Backlog

6.3.11. Crear historia

En esta pantalla podemos crear una historia nueva. Tendremos que rellenar los datos relativos a la historia.

Ilustración 42: Vistas - Crear historia

6.3.12. Ver características de la historia

En esta pantalla podemos ver las características principales de una historia. Desde aquí se puede acceder a modificar las características de la historia, borrar la historia, definir la historia en estado de pruebas o como fin de pruebas y ver las tareas en las que se descompone.

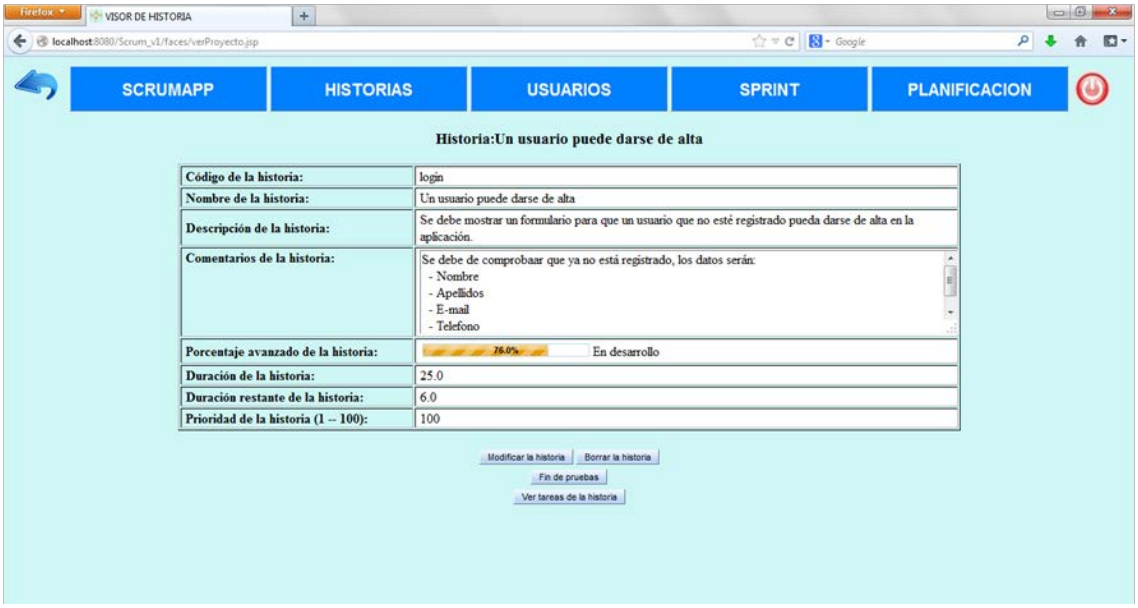


Ilustración 43: Vistas - Ver características de la historia

6.3.13. Modificar la historia

En esta pantalla podemos modificar las características principales de una historia.

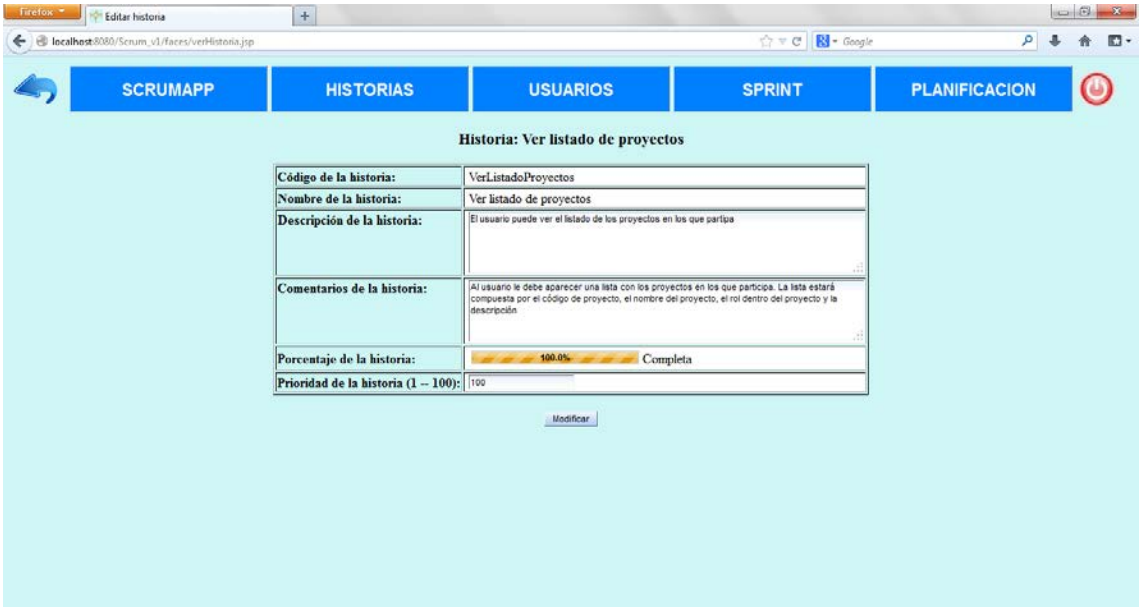


Ilustración 44: Vistas - Modificar historia

6.3.14. Ver tareas

En esta pantalla podemos ver un listado con las tareas que forman la historia. Se puede acceder a una tarea en concreto o se puede crear una tarea nueva.

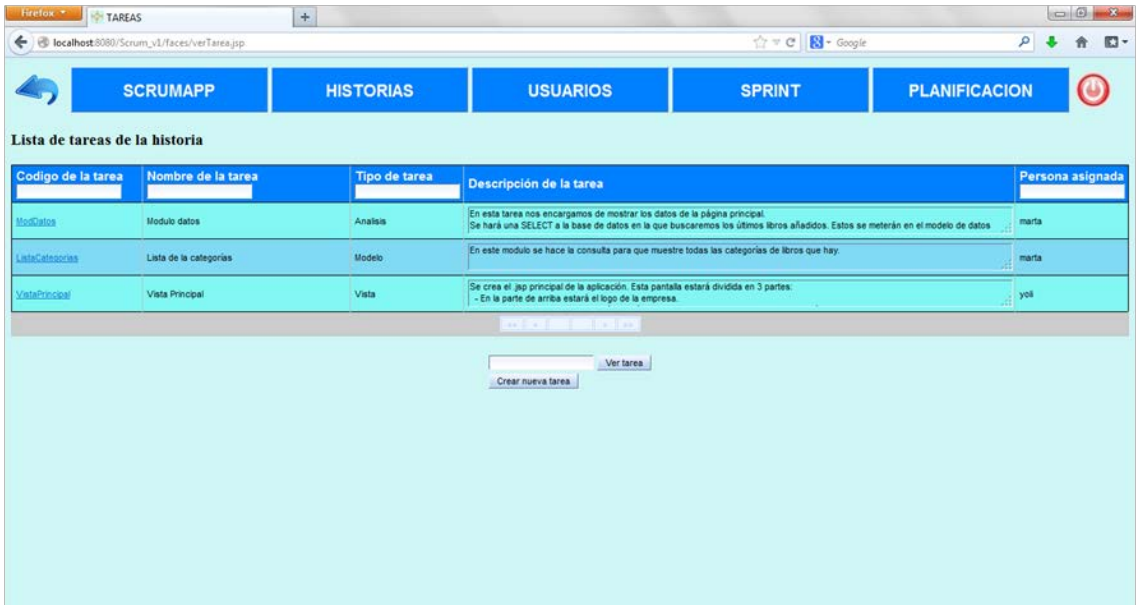


Ilustración 45: Vistas - Ver tareas

6.3.15. Crear nueva tarea

En esta pantalla podemos crear una tarea nueva. Para ello se debe rellenar el formulario.

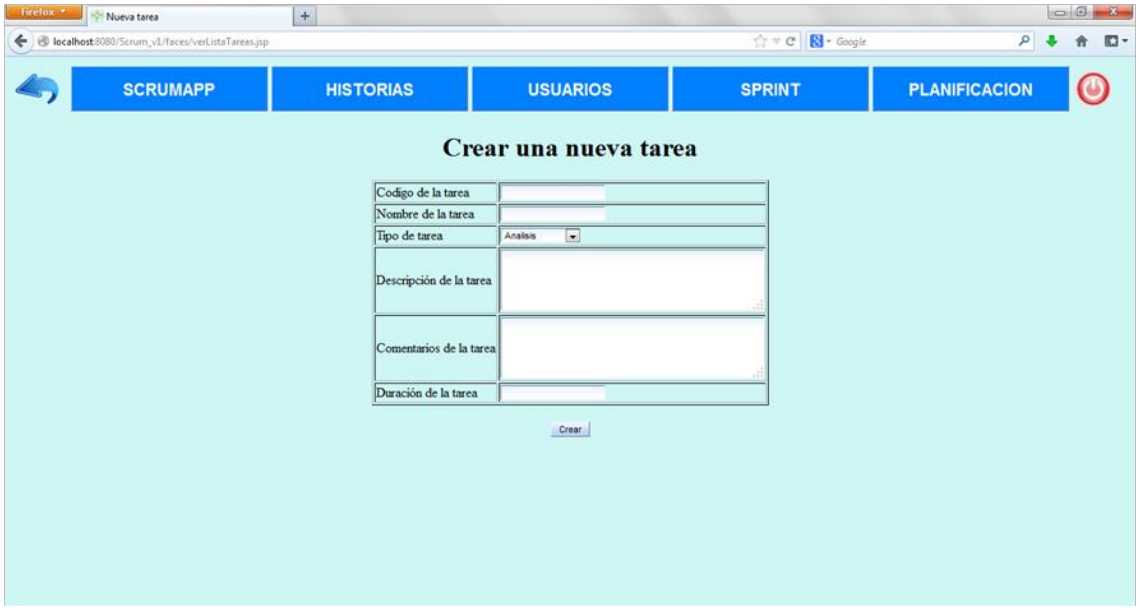


Ilustración 46: Vistas - Crear tarea

6.3.16. Ver características de una tarea

En esta pantalla podemos ver las características de una tarea. También podemos acceder a borrarla o a modificarla (las características).

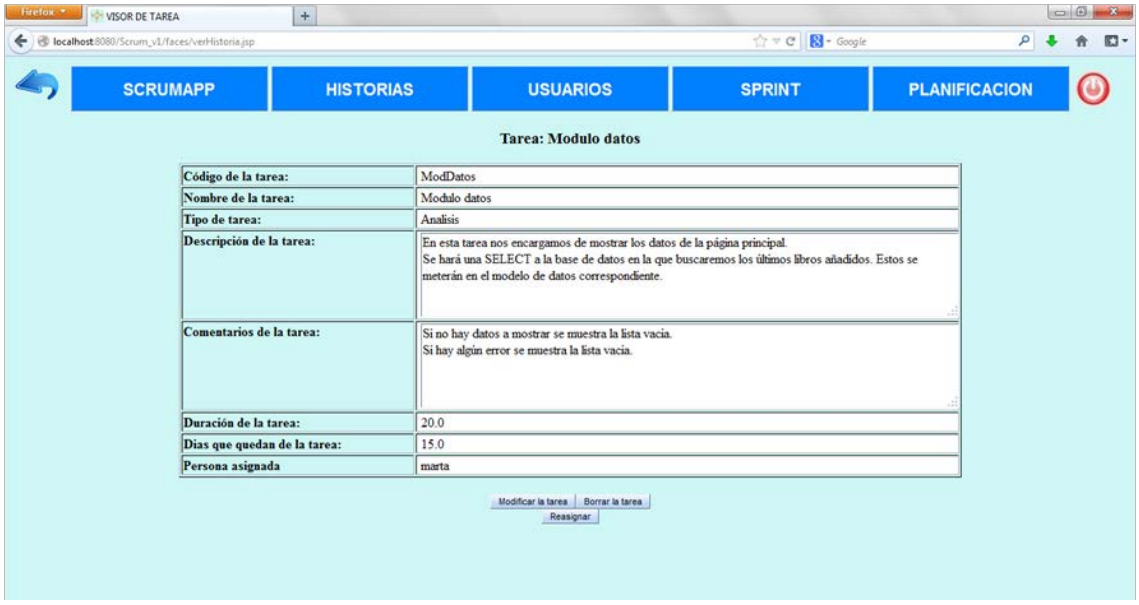


Ilustración 47: Vistas - Ver características de una tarea

6.3.17. Modificar características de una tarea

En esta pantalla podemos modificar las características de una tarea rellenando los campos oportunos del formulario.

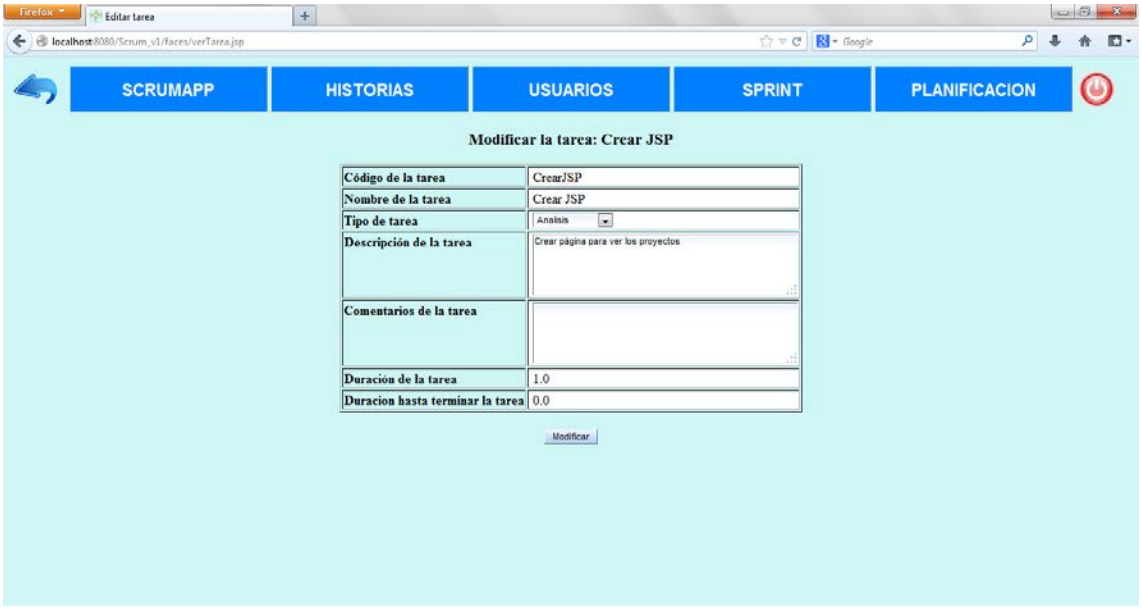


Ilustración 48: Vistas - Modificar características de una tarea

6.3.18. Reasignar tarea

En esta pantalla podemos reasignar una tarea de un empleado del proyecto a otro.

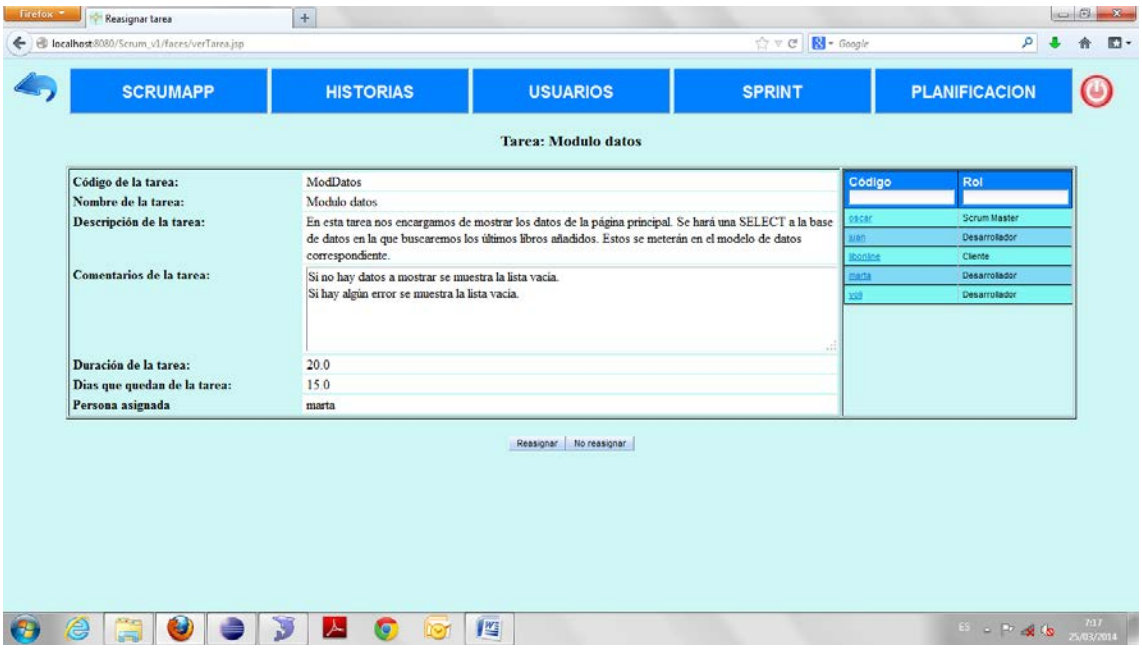


Ilustración 49: Vistas - Reasignar tarea

6.3.19. Sprint

En esta pantalla podemos ver un sprint del proyecto.

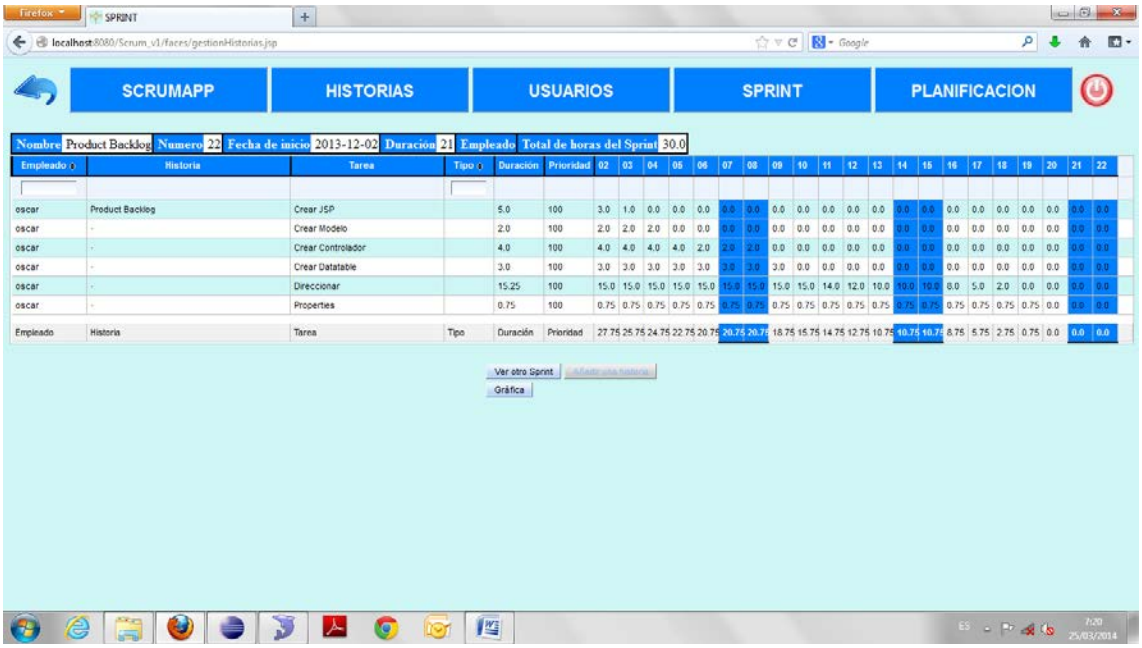


Ilustración 50: Vistas - Sprint

6.3.20. Gráfica

Esta pantalla emergente tiene la gráfica del progreso del sprint.

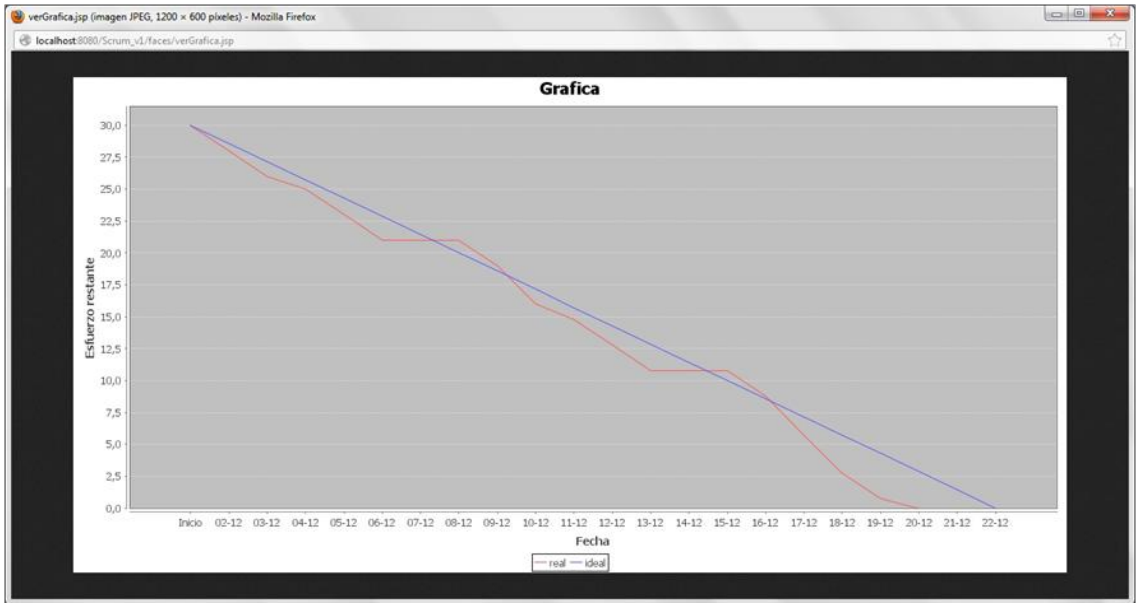


Ilustración 51: Vistas - Gráfica

6.3.21. Ver otro Sprint

Esta pantalla muestra todos los Sprint que tiene el proyecto. Desde aquí se puede crear un Sprint nuevo.

Ilustración 52: Vistas - Ver otro Sprint

6.3.22. Crear Sprint

Esta pantalla permite crear un Sprint nuevo.

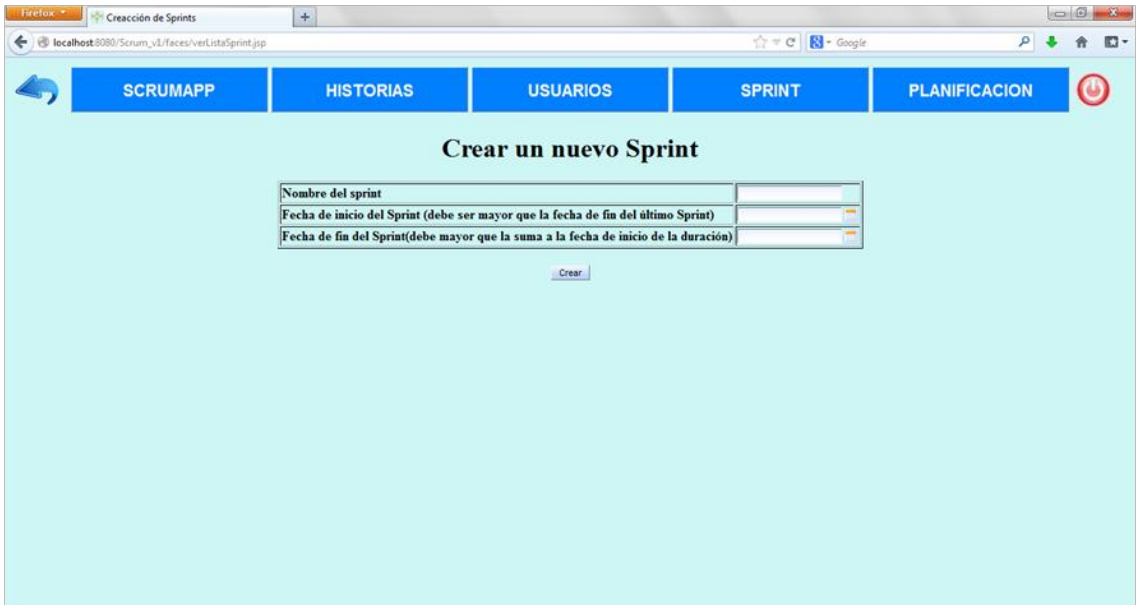


Ilustración 53: Vistas - Crear Sprint

6.3.23. Imputación (1)

Esta pantalla nos muestra una lista con las tareas pendientes de un proyecto.

Ilustración 54: Vistas - Imputación (I)

6.3.24. Imputación (2)

En esta pantalla se rellena el esfuerzo que se ha dedicado a la tarea

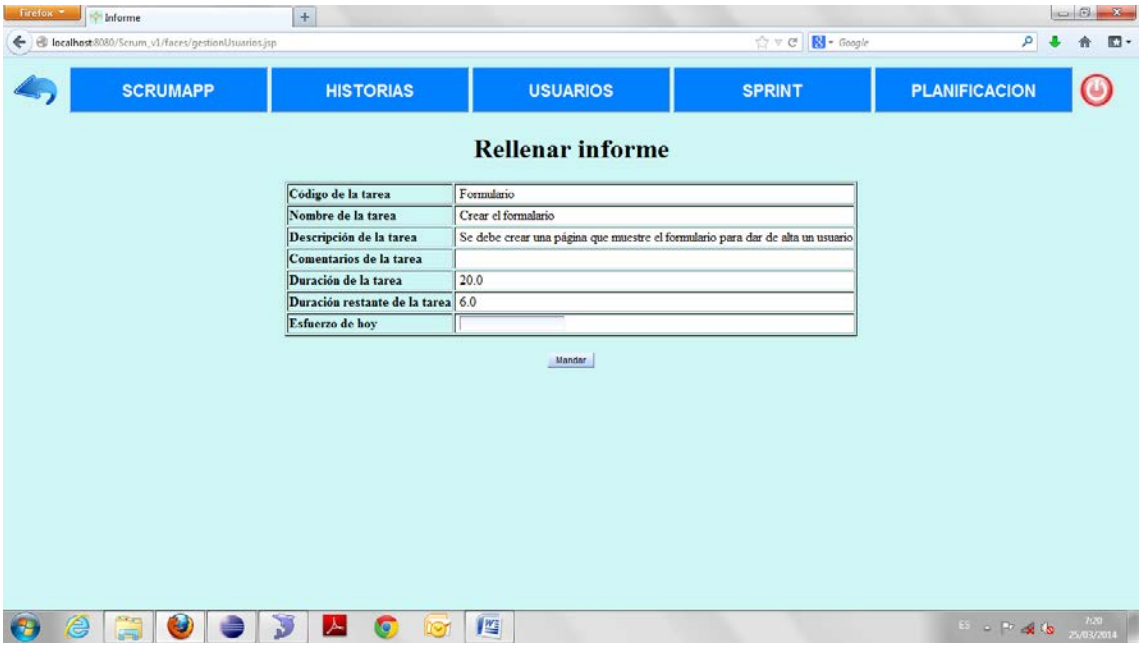


Ilustración 55: Vistas - Imputación (II)

6.3.25. Planificación

Esta pantalla permite tener una visión general del proyecto. Vienen todas las historias colocadas según su estado.

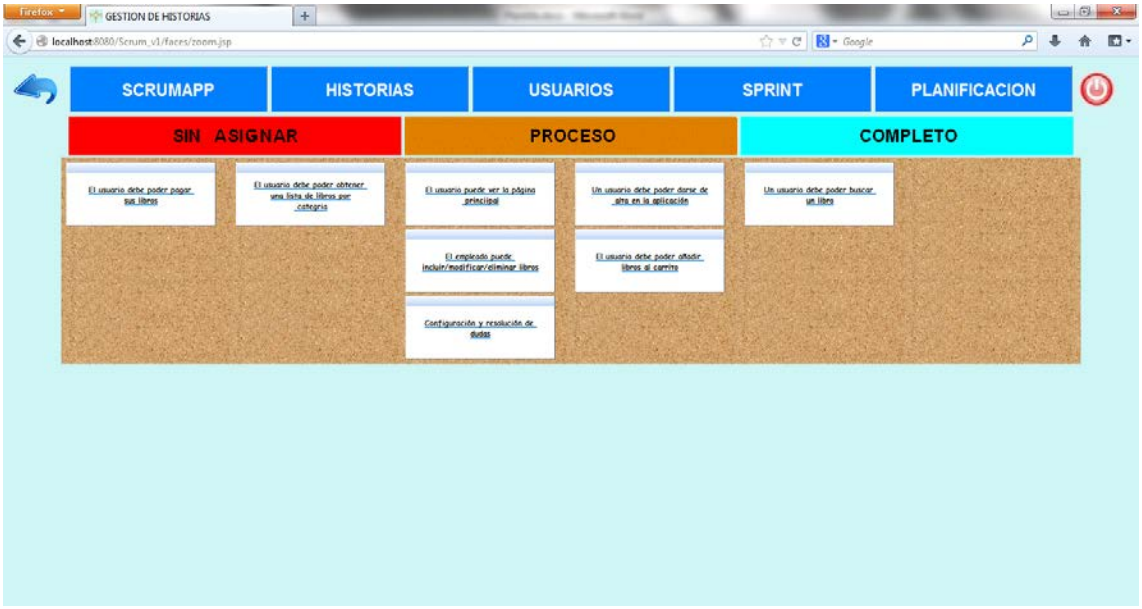


Ilustración 56: Vistas - Planificación

6.3.26. Planificación (Progreso)

Esta pantalla permite tener una visión general de las historias que progreso del proyecto y el porcentaje virtual avanzado.

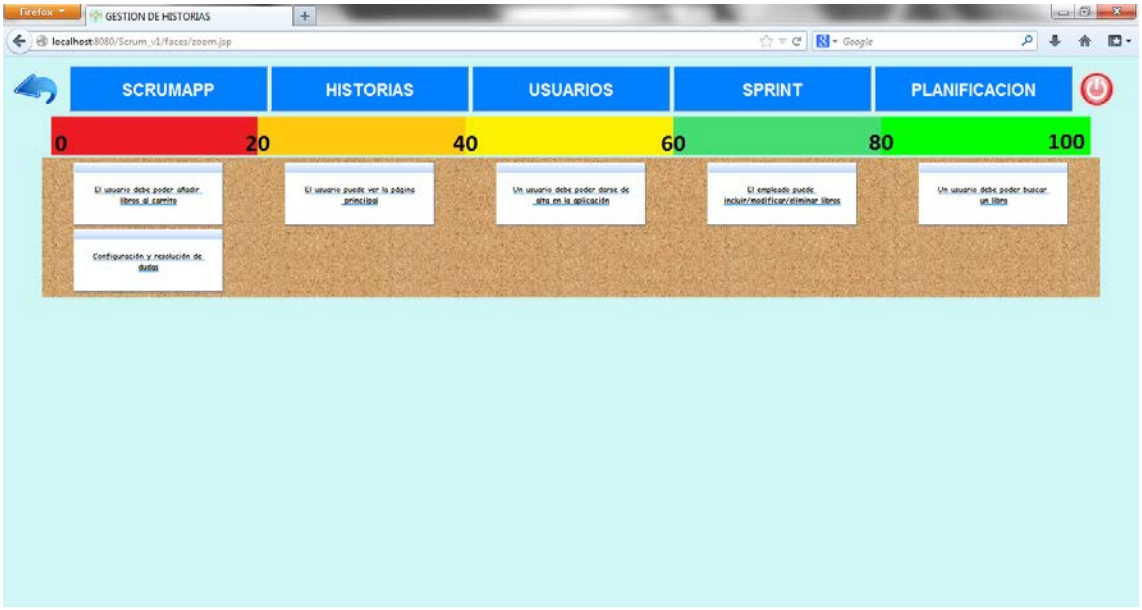


Ilustración 57: Vistas - Planificación - Progreso

7. Presupuesto

Se procede a detallar los gastos relacionados por la realización del proyecto. En este punto, se incluirán tanto los costes directos (personal, materiales, etc.), como indirectos (desplazamientos, dietas, etc.).

Tal y como se ha visto en la planificación, el tiempo estimado en la realización del proyecto es de 611,75 (1 año y 8 meses al 25%). Esta planificación se ha visto alterada ya que se han requerido cambios una vez que el proyecto ya estaba definido. Por ello la duración total de este proyecto ha sido de 3 años. Esta duración tiene en cuenta desde que empieza y se recogen los requisitos y se realizan todos los estudios previos, hasta que se termina esta documentación. Con este dato, se obtendrán los diferentes costes asociados. A continuación se detallarán estos gastos.

7.1. Costes directos

7.1.1. Equipos

En cuanto a costes materiales, ha sido necesario adquirir un ordenador portátil.

Descripción	Coste	% uso dedicado al proyecto	Dedicación (meses)	Periodo de deprecación	Coste imputable (euros)
Ordenador	699	100%	36	60	419,40€

Tabla 45: Costes - Equipos

7.1.2. Software

Descripción	Licencias	Coste/Licencia	Coste imputable
Microsoft Office 2007	1	99	99,00 €
Eclipse	1	0	0,00 €
Tomcat	1	0	0,00 €
Richfaces	1	0	0,00 €

Tabla 46: Costes - Software

7.1.3. Personal

Puesto	Salario Anual	% uso dedicado al proyecto	Meses Trabajados	Coste imputable
Programador Junior	18.000€	25%	27	10.125,00€

Tabla 47: Costes - Personal

7.2. Costes indirectos

Entre los gastos indirectos se pueden encontrar los viajes y dietas de los trabajadores para la realización de reuniones con el cliente, gastos de consumo como electricidad, agua, gas o teléfono o el alquiler del inmueble.

7.2.1. Viajes

Tipo	Coste unitario	Viajes	Coste imputable
Viajes en coche	100 km * 0,21 €/km	10	210 €

Tabla 48: Costes - Viajes

7.3. Coste total

A continuación se resumirán los diferentes costes mencionados y se calculará el coste total del proyecto incluyendo el Impuesto sobre el Valor Añadido (IVA).

Concepto	Coste total
Equipos	419,40
Software	99,00
Personal	10.125,00
Viajes	210,00
Total sin IVA	10.853,40
IVA (21%)	2.279,21
TOTAL	**13.132,62**

Tabla 49: Costes - Total

8. Conclusiones

El objetivo de este proyecto fue el de que un equipo de personas pudieran gestionar un proyecto con Scrum desde la aplicación. Vamos a pasar a evaluar hasta que punto esto se puede hacer con la aplicación que se ha creado.

Para poder gestionar un proyecto, el usuario debe poder registrarse y entrar en la aplicación.

Proyecto. A partir de entonces el usuario debe poder crear un proyecto y asignárselo a las demás personas del equipo. Debe poder modificarlo y borrarlo. Como añadido, se ha generado una opción en la que se puede extraer toda la información del Sprint a un documento a modo de documentación.

Historias. Una vez creado el proyecto, el ScrumMaster debe poder dar de alta las historias y las tareas (estas las pueden dar de alta el equipo de trabajo también). Deben poder asignárselas entre ellos y poder modificarlas y borrarlas.

Sprint. Una vez que haya historias y tareas, el ScrumMaster debe poder crear Sprints. Además debe poder añadir historias a los Sprint. Los usuarios que tengan asignado el proyecto podrán ver el progreso de un Sprint. Como añadido a este apartado cabe destacar que la aplicación también permite la creación de gráficas para poder ver el progreso de una manera más visual.

Product Backlog. El equipo de trabajo debe poder comprobar la situación global del proyecto, por tanto, la aplicación también permite ver la situación de todas las historias de manera visual.

Progreso. Para que todo esto sea posible, todo el equipo de trabajo debe imputar el trabajo a diario (así no habrá desviaciones en las gráficas). Por ello, la aplicación permite dar de alta esa imputación.

Por todo lo anterior, podemos decir que, actualmente la aplicación Web puede ser usada para gestionar un proyecto mediante la metodología Scrum.

Por otro lado, debo destacar que la aplicación es una ayuda en la gestión del proyecto. Por tanto, las reuniones diarias que se deben hacer cuando se trabaja con

Scrum no se deben omitir (aunque la aplicación puede servir de ayuda en estos casos). Tampoco se pueden omitir las reuniones con el Product Owner ni las de estimación (aunque igualmente la aplicación puede servir de ayuda).

Conclusiones personales: una vez comprobado que la aplicación cuenta con lo necesario para poder ser utilizada, se van a realizar una serie de conclusiones personales.

Este proyecto ha sido un reto personal ya que mis conocimientos anteriores en las metodologías ágiles eran inexistentes y gracias al proyecto he conseguido obtener unos fundamentos sobre Scrum muy valiosos. Otro reto fue el lenguaje de programación JAVA que antes desconocía y del que he adquirido unos buenos conocimientos que probablemente me ayuden tanto en mi desarrollo personal como profesional.

Además de lo anterior, la gestión de un proyecto completo te ayuda a tener otra visión general del proceso y del porqué de tener una metodología bien definida. Además me he dado cuenta de lo importante que son los pasos iniciales de aprendizaje para el futuro del proyecto.

9. Trabajos futuros

En este punto, se tratarán las posibles mejoras y nuevas funcionalidades que se podría añadir a la aplicación para hacerla más completa, con el fin de mejorar la experiencia de usuario y ampliar las posibilidades que nos ofrece.

9.1. Un empleado puede valorar una historia

Una funcionalidad para un trabajo futuro podría ser el crear una nueva pantalla en la que los desarrolladores puedan valorar las historias on-line. En Scrum, las valoraciones de los casos de uso las realiza el equipo entero en una reunión. Se podría adaptar la aplicación para que estas valoraciones pudieran hacerse on-line.

9.2. Cálculo del presupuesto del proyecto

Otra funcionalidad que podría implementarse es el cálculo automático del presupuesto de un proyecto según el número de horas y el precio de cada persona.

9.3. Rectificación de imputaciones

Otra posible mejora sería que un empleado pudiera cambiar su imputación. Actualmente una vez que imputa a una tarea, el empleado ya no puede modificar su imputación. En un futuro podría añadirse esta funcionalidad.

9.4. Imputación de varios días

Otra funcionalidad extra es la imputación de varios días. En la actualidad, un empleado debe imputar diariamente. Si un día no imputa, no podrá hacerlo el día siguiente. En una metodología ágil, se debería imputar a diario para que el ScrumMaster tenga una visión de cómo avanza el Sprint pero siempre viene bien que se pueda imputar lo de días anteriores por si hay algún problema con la herramienta algún día.

9.5. Herramienta valoradora

Otra mejora sería una herramienta que actuara como valorador. Esta se podría definir al crear el proyecto y que dependiendo del tipo de tarea y de su complejidad diera una posible estimación de la tarea. Esto se podría complementar con la mejora de la valoración de las historias por parte de equipo de trabajo.

Glosario

Acrónimo	Definición
API	Aplication programing Interface
GB	Gigabyte
JSF	Java Server Faces
JSP	Java Server Pages
J2EE	Java2 Enterprise Edition
MVC	Modelo-Vista-Controlador
PO	Product Owner
RAM	Random Access Memory
XML	eXtensible Markup Language
XUL	XML-based UserInterface Languaje

Tabla 50: Glosario

Bibliografía

Descripción. (s.f.). Obtenido de http://www.lsi.us.es/docencia/get.php?id=5438

funcionamiento del MVC. (s.f.). Obtenido de http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista2.shtml

Goldberg, A., & Robson, D. *The languaje and its implementation.*

JSF. (s.f.). Obtenido de http://java.sun.com/javaee/5/docs/tutorial/doc/bnaph.html

JSF Wikipedia. (s.f.). Obtenido de es.wikipedia.org/wiki/JSF

Kniberg, H. (2007). *Scrum y XP desde las trincheras.* C4Media Inc.

Manual oficial SpringMVC. (s.f.). Obtenido de http://static.springsource.org/spring/docs/2.5.x/reference/index.html

Roughley, I. (2007). *Starting Struts 2.* LULU Press.

Spring wikipedia. (s.f.). Obtenido de es.wikipedia.org/wiki/Spring_Framework

Structs wikipedia. (s.f.). Obtenido de es.wikipedia.org/wiki/Apache_Struts

Tapestry. (s.f.). Obtenido de http://tapestry.apache.org/