Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Aplicación para la Gestión de Proyectos en Ambientes Sociales y Colaborativos bajo el enfoque de la Web 2.0

Trabajo Especial de Grado presentado ante la ilustre Universidad Central de Venezuela por la Br. Elisa Conesa Izquierdo, con cédula de identidad V15.487.139 para optar por el título de Licenciado en Computación. Este trabajo ha sido tutelado por la Prof. Nora Montaño

Caracas, octubre del 2013

ACTA DEL VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por la Bachiller **Elisa Conesa Izquierdo**, titular de la cédula de identidad No. 15.487.139, con el título “Aplicación para la Gestión de Proyectos en Ambientes Sociales y Colaborativos bajo el enfoque de la Web 2.0”, con el fin de optar por el título de Licenciado en Computación, dejan constancia de lo siguiente:

Una vez leído este trabajo por cada uno de los miembros del Jurado, se fijó el día 28 de octubre de 2013 a las 10:00 am, para que su autora lo defienda en forma pública, en la Sala de Seminarios del Centro ISYS de la Escuela de Computación en la Facultad de Ciencias de la Universidad Central de Venezuela. Una vez realizada esta defensa, a través de una presentación oral del contenido, el Jurado decidió aprobarlo.

En fe de lo anterior, se levanta la presente Acta, en Caracas a los veintiocho (28) días del mes de octubre del año dos mil trece (2013).

_____
Profa. Nora Montaño (Tutora)

_____ _____
Prof. Eugenio Scalise (Jurado) Prof. Esmeralda Ramos (Jurado)

RESUMEN

Los avances y facilidades que ofrecen en la actualidad la Web, han motivado a ampliar el uso de las plataformas sociales y colaborativas como ambientes de aprendizaje. Esto ha motivado que muchas investigaciones se direccionen al reuso de estas plataformas con fines educativos, sin embargo, nuevas características son requeridas pudiendo ser necesario modificar a nivel de código la plataforma. El objetivo de este trabajo especial de grado es enriquecer una plataforma social open source (*Oxwall Software*) dotándola con una aplicación para la gestión de proyecto colaborativo con característica de la web 2.0. Para llevar a cabo el desarrollo, se utiliza un bosquejo de las características actuales en *HTML5* y, bajo el método de desarrollo *SCRUM*, se inicia el desarrollo de la aplicación sobre la red social *Oxwall*. La tecnología utilizada fue *HTML5*, *PHP* y *MySQL*..

Palabras claves: Web 2.0, Open Source, redes sociales, modelo educativo, proyectos colaborativos

TABLA DE CONTENIDOS

RESUMEN 5

INTRODUCCIÓN 9

MARCO CONCEPTUAL 11

CONTEXTO 12

OPEN SOURCE 12

MOZILLA PUBLIC LICENSE (MPL) 15

COMMON PUBLIC ATTRIBUTION LICENSE (CPAL) 15

WEB 2.0 16

GESTIÓN DE PROYECTOS 23

MARCO APLICATIVO 27

DESCRIPCIÓN DE LA SITUACIÓN ACTUAL 27

DESCRIPCIÓN TECNOLÓGICA 27

CAPTURA DE REQUERIMIENTOS 28

INICIO DEL DESARROLLO DE LA APLICACIÓN 34

CICLO DE DESARROLLO DE LA APLICACIÓN 36

ASIGNACIÓN #1 - REALIZAR MODELO DE ENTIDAD/RELACIÓN 36

ASIGNACIÓN #2 - REALIZAR DIAGRAMA DE CASOS DE USOS 39

ASIGNACIÓN #3 - REALIZAR DIAGRAMA DE ACTIVIDADES 52

ASIGNACIÓN #4 - REALIZAR DIAGRAMA DE CLASES 53

ASIGNACIÓN #2.1 – DISEÑAR COMPONENTES BÁSICOS DE LA APLICACIÓN 54

ASIGNACIÓN #3 – GESTIONAR PROYECTOS 61

ASIGNACIÓN #4 – GESTIONAR MODELOS DE DESARROLLO 61

ASIGNACIÓN #5 – GESTIONAR FASES Y PROYECTO-FASES 63

ASIGNACIÓN #3 – GESTIONAR PROYECTOS (CONTINUACIÓN) 63

ASIGNACIÓN #6 – GESTIONAR ACTIVIDADES Y PROYECTO-ACTIVIDADES 64

ASIGNACIÓN #3 – GESTIONAR PROYECTOS (CONTINUACIÓN) 64

ASIGNACIÓN #7 – PROGRAMAR PROYECTO 65

ASIGNACIÓN #2.2 – DISEÑAR COMPONENTE COMPLEJOS DE LA APLICACIÓN 65

ASIGNACIÓN #7 – PROGRAMAR PROYECTO (CONTINUACIÓN) 67

ASIGNACIÓN #8 – GESTIONAR TAREAS 67

ASIGNACIÓN #3 – GESTIONAR PROYECTOS (CONTINUACIÓN) 68

ASIGNACIÓN #9 – GESTIONAR USUARIOS 68
ASIGNACIÓN #10 – GESTIONAR PARTICIPANTES DEL USUARIO 68
ASIGNACIÓN #3 – GESTIONAR PROYECTOS (CONTINUACIÓN) 69
ASIGNACIÓN #10 – GESTIONAR RESPONSABLES (CONTINUACIÓN) 69
ASIGNACIÓN #11 – REALIZAR ENTREGA DE TAREA 69
ASIGNACIÓN #10 – GESTIONAR RESPONSABLES (CONTINUACIÓN) 70
ASIGNACIÓN #6 – GESTIONAR ACTIVIDADES Y PROYECTO-ACTIVIDADES (CONT.) 70
ASIGNACIÓN #13 - REALIZAR PRUEBAS DE LA APLICACIÓN 70
ASIGNACIÓN #14 - IMPLEMENTAR LA INTEGRACIÓN DEFINITIVA DE LA APLICACIÓN COMO PLUGIN 72

ESCENARIO DE USOS 73

SUGERENCIAS 79

CONCLUSIONES 81

REFERENCIAS 83

ANEXOS 85

FRAMEWORK DE DESARROLLO OXWALL 85
SCRUM 87

INTRODUCCIÓN

La Web ha evolucionado significativamente desde sus inicios. Sobre aquellas antiguas páginas estáticas, se fueron incorporando tecnologías que las han enriquecido. La tecnología Web actual permite llevar los procesos laborales y educativos a sitios Web. Esta automatización de los procesos permite un mayor control y manejo de los procesos y hacen que hoy en día la tecnología Web penetre en casi todas las actividades laborales y educativas de la sociedad.

En el artículo Modelo de enseñanza colaborativa basado en la Web 2.0 para el fortalecimiento de la enseñanza en ciencia y tecnología (Miguel et al., 2012) se ha planteado la elaboración de una metodología que propulse el aprendizaje a través de las facilidades ofrecidas por las herramientas Web. Es posible sustentar el hecho de que las herramientas sociales permiten la exposición y edición del conocimiento de forma inmediata y conjunta entre todos los miembros de la comunidad.

En este trabajo, se describen aquellas características que hicieron crecer a la Web. Se presentan las políticas *open source,* las cuales surgen para permitir el compartir partes o software completos. Con ello, se proseguirá a la elaboración de una aplicación que de un aporte tecnológico que, en conjunto con muchos otros aportes, encamine el levantamiento de una plataforma para la metodología.

En Modelo de enseñanza colaborativa basado en la Web 2.0 para el fortalecimiento de la enseñanza en ciencia y tecnología (Miguel et al., 2012), también fue propuesto el uso de la red social Oxwall como una solución base gratuita y *open source* para la implantación de la metodología, dadas las características favorables de esa red social para la implantación. Por ello, la aplicación a desarrollar se realizará en dicha red social, a fin de implementar diversos módulos que satisfagan determinados objetivos presentados más adelante. Actualmente, la red social Oxwall requiere disponer de un gestor de proyectos en líneas de tiempo. Con el presente trabajo, se quiere cubrir tal necesidad.

Bajo lo expuesto, se concretan el objetivo general y los objetivos específicos del trabajo.

Objetivo General:

Enriquecer una plataforma social *open source* (*Oxwall Software*) dotándola con una aplicación para la gestión de proyectos colaborativos con características de *Web 2.0*.

Objetivos Específicos:

1. Analizar el desarrollo de una aplicación para la gestión de proyectos a través de las líneas de tiempo.
2. Desarrollar la aplicación siguiendo un método de desarrollo a la medida que permita un mejor seguimiento de los proyectos colaborativos.
3. Reutilizar y generar componentes reutilizables que tengan coherencia y consistencia con respecto al software *Oxwall*.

Este documento se componen de una estructura dividida en un marco conceptual y un marco aplicativo.

El marco conceptual tiene la finalidad de presentar los aspectos teóricos requeridos y obtenidos de la investigación consecuente al trabajo. Esto incluye la problemática que motiva la realización del trabajo, el contexto del mismo, entre otros aspectos a tratar.

Por otra parte, en el marco aplicativo se trabajan los aspectos técnicos del desarrollo y documentación de los detalles del método de desarrollo y respectivos artefactos del mismo.

Finalmente, el documento concluye especificando la experiencia obtenida del desarrollo y sugerencias para posibles extensiones de la aplicación.

MARCO CONCEPTUAL

Desde los inicios de la Web han ido surgiendo varias tecnologías que han hecho de ella un ambiente muy distinto. Las posibilidades que ofrece hoy en día han hecho que sea utilizada para el estudio de nuevas tendencias en la educación.

Entre las tendencias surgidas en la educación está la elaboración de proyectos colaborativos desde la Web. Un proceso de interés debido a que es un ambiente propicio para que el aporte de cada miembro sea medido y supervisado de forma inmediata. Además de que los miembros de la comunidad pueden ayudarse mutuamente, lo que hace que haya un mayor beneficio grupal e individual.

En la Web se pueden encontrar varias herramientas para el seguimiento de proyectos, sin embargo es poco efectivo el valor que ofrecen éstas para contextos colaborativos libres. En este trabajo, se desea abordar esta problemática. Debe saberse que, en particular, la comunidad del Centro de Ingeniería de Software y Sistemas hace uso de varias herramientas colaborativas. De lograrse la mejora de estas aplicaciones en dicho contexto, ampliaría las operaciones y logros de dicho centro.

Antes de iniciar el análisis del desarrollo de la aplicación Web, se necesita investigar los aspectos actuales de la tecnología Web y qué se necesita para lograr extender las funcionalidades de una de las aplicaciones que utiliza la comunidad. Este desarrollo también daría un aporte para la comunidad libre dado que se utilizan herramientas libres.

Como punto inicial al estudio de la tecnología actual, quiere darse una introducción a la característica *open source*. Se desea exponer cuáles son sus condiciones y detallar las licencias de las herramientas que se utilizarán en este trabajo. Posteriormente, se expondrá algunos aspectos sobre la evolución de la Web y los principios por la cual se genera una versión 2 de la misma.

CONTEXTO

Con este trabajo, se desea enriquecer una aplicación *open source* bajo determinada licencia. Se amerita el conocimiento de las condiciones de uso de dicha licencia antes de iniciar el desarrollo. Adicionalmente, debe conocer de dónde viene la motivación al nacimiento de las licencias *open source*.

Open Source

Desde hace varios años los asuntos sobre la protección de la propiedad intelectual han sido un tema en discusión. Todo producto se distribuye bajo alguna licencia que indica sus condiciones de uso. La violación de dichas condiciones puede implicar una muy costosa multa dependiendo del país de procedencia. Muchas empresas se han visto involucradas en fuertes demandas por infligir la propiedad intelectual de software privados. No obstante, algunos grupos notaron que, en algunos casos, el propietario lograba monopolizar el mercado. Esto llevó que empezaran ha surgir licencias que lo evitasen.

Aquel software cuyo código pueda ser visto, editado y/o distribuido sin infligir derechos de autor es llamado *software* o*pen source* (De *OSI*, 2013). Una licencia *open source* establece una marca de certificación de propiedad a un *software open source.* Entre algunas licencias *open source* a mencionar está *Common Public Attribution License (CPAL), Apache License*, Berkeley Software Distribution License (*BSD).*

A partir de la existencia de las licencias *open source* y de *software libre* se permite el acceso público a los códigos fuente del *software*. De este modo, el código *per se* es utilizado a mayor escala porque más gente puede accederlo. Según el mantra “Release early, release often”, el uso a una mayor escala lleva a la tendencia de que se desarrollen mejoras y limpiezas de *bugs.* Así mismo, se logra un incremento de los lanzamientos de nuevas y mejores versiones oficiales. El software *open source* inclusive facilita el lanzamiento de versiones compatibles con mayor cantidad de dispositivos, ya que cada fabricante puede adaptar el software a sus dispositivos sin tener problemas legales.

La libertad de acceder al código, además, permite que la solución a un problema dado esté expuesta. Este hecho hace que esa solución pueda ser

reutilizada en otros contextos. La reutilización de código, a su vez, reduce los costos de producción de *software*.

Las comunidades de software *open source* suelen ser más diversas, dado que no suelen haber privativas para ser parte de ella. No obstante, el software *open source* no tiene el financiamiento que dispone el software propietario. En muchos casos, la documentación del software *open source* depende del aporte que den dichas comunidades y puede estar incompleto o inexistente. Incluso es común que el soporte de mucho software *open source* sea a través de foros y blogs; herramientas que no facilitan una presentación organizada.

Las condiciones de uso para un *software open source* son las siguientes (De *OSI*):

1. Libre distribución

2. Cualquier parte interesada puede vender o distribuir gratuitamente el software. No se requieren regalías por tal venta o distribución.

3. Acceso al código fuente

4. En la distribución del software deben estar incluidos los códigos fuentes y binarios (de existir). Si alguna versión del producto no distribuye el código fuente, éste debe encontrarse publicado en algún medio bien conocido. De requerirse un costo de reproducción, éste debe ser moderado. El formato del código debe ser el apropiado para su modificación por parte de los programadores. No se permite el ofuscamiento deliberado del código fuente. Dicho código no puede haber sido alterado por traductores o preprocesadores.

5. Trabajos derivados

6. Se permite realizar modificaciones o trabajos derivados de un software y pueden ser distribuidos bajos los mismos términos del software original.

7. Integridad de código fuente del autor

8. Puede estar restringido la distribución de versiones modificadas sólo si se permite la distribución de archivos *patch* a fin de modificar el software en tiempo de compilación.

9. No discriminar ningún grupo

10. Los términos con que se certifica el software no deben marginar o discriminar a ninguna persona ni grupo.

11. No restringir aplicaciones

12. No debe restringirse los campos en que puedan aplicarse el software.

13. Distribución de licencia

14. Los derechos establecidos al software deben ser válidos para cualquiera que redistribuya el software sin que haya necesidad de aplicar términos adicionales de otras partes.

15. Los términos no deben ser específicos para un producto.

16. Los derechos establecidos para el programa no deben depender de ser parte de alguna distribución del software en especifico. En caso de extraerse alguna parte del software y ésta sea redistribuida, todas las partes para quienes el software está siendo redistribuido deben conservar las mismas condiciones que fueron expuestas en la distribución original.

17. No deben restringir a otros software

18. Aquellos *software* distribuidos conjuntamente con el *software* licenciado no deben estar sujetos a las restricciones del *software* licenciado.

19. Debe haber tecnología-neutral

20. Los términos del producto no pueden estar sujetos a una tecnología en particular.

Las condiciones numeradas anteriormente expresan que un *software open source* puede que sea o no sea gratuito. Del mismo modo que un software gratuito puede que no sea *open source*.

Se presentan las licencias *open source* relacionadas con este trabajo.

Mozilla Public License (MPL)

Mozilla Public License (MPL) puede presentarse como una licencia intermedia entre las licencias *GNU General Public License* (GPL) y *Berkeley Software Distribution License* (BSD).

Acorde al resumen de Rowan Wilson (2005), sus condiciones de uso se dividen entre condiciones del autor original y condiciones de otras personas.

Las condiciones del autor original son las siguientes:

1. Puede usar, reproducir, modificar, desplegar, sub-licenciar y distribuir los fuentes incluyendo aquellos de versiones modificadas.

2. Puede patentar derechos de uso y poner disponible el código original.

3. Puede distribuir trabajos, los cuales el código contenga una combinación con código nuevo, y licenciar el nuevo código como el distribuidor lo prefiera.

Las condiciones de otras personas son las siguientes:

- Puede usar, reproducir, modificar, desplegar, sub-licenciar y distribuir los fuentes de sus modificaciones.

- Puede patentar derechos de uso y poner disponible sus modificaciones o el trabajo entero (código original más modificaciones).

- Puede distribuir el trabajo que contenga código modificado, y licenciar el nuevo código tal como el distribuidor prefiera.

Common Public Attribution License (CPAL)

Common Public Attribution License (CPAL) es una adaptación de *Mozilla Public License,* aprobada por *OSI* en julio de 2007. Como expresa Rowan Wilson (2008), esta adaptación conserva las condiciones de *MPL*, a excepción de dos puntos a mencionar seguidamente:

1. El software derivado debe incluir los créditos del autor original. Esto es un *copyright* que incluya el nombre, logo y *URL* correspondiente al autor original.

2. La distribución del código fuente debe mantenerse a través de la Web, de forma que la comunidad siga beneficiándose.

Como puede notarse en el segundo punto, la licencia *CPAL* es una licencia que promueve la distribución del software por la Web.

Este trabajo tiene la motivación de enriquecer una aplicación web, y siguiendo ella una aplicación de la *Web 2.0* es necesario presentar las características y principios de la *Web 2.0.*

Web 2.0

Desde el advenimiento de la Web como medio masivo de difusión de información, la Web ha experimentado una gran cantidad de avances tecnológicos que han permitido posicionarla como una plataforma de comunicación más interactiva y colaborativa. Es de apreciar la gran cantidad de opciones que ofrece la web actualmente, a diferencia de sus inicios.

La Web en los años 90, la que es referida en este documento como *Web 1.0*, se caracterizaba por estar compuesta de páginas estáticas, cuya función era únicamente mostrar contenido (de la *W3C, 2013*). A medida que fue aumentando el tráfico de usuarios en la Web, se incrementa el interés de tener presencia en la Web. Las tiendas tradicionales empiezan a anunciar y a publicar el catálogo de sus productos en línea a finales de los años 90. La visualización de catálogos en línea dio la posibilidad de adquirir dichos productos directamente desde la Web.

No obstante, ofrecer un servicio para la venta de productos a través de una tienda en línea requería cierta plataforma. También era necesario que el usuario percibiese la Web como un ambiente seguro y atractivo. Un conjunto de tecnologías fueron desarrolladas para tales propósitos.

Netscape Communications en 1994 desarrolla el protocolo *Secure Socket Layer (SSL)* para su navegador *Navigator*. En 1995, desarrolla *Javascript*. Los *Java Applets* son desarrollados por *Sun Mycrosistems* en 1995. La plataforma *Flash*

surge en 1996. El lenguaje *XML* es publicado en *1996.* En el *2000* surge *Asynchronous JavaScript and XML (AJAX).*

A partir de ese momento, el surgimiento de las tiendas en línea (o tiendas digitales) fue una realidad.

La Web de ese momento (denominada en citas fuentes como *Web 1.5*) parecía un negocio muy atractivo. El servicio de compras (nuevo para aquel entonces) daba la comodidad de concretar una compra y recibir el producto a domicilio; razón que llevó a pensar a muchas empresas, que esas tiendas digitales sustituirían totalmente a las tiendas tradicionales. Fueron muchos los activistas de capital riesgo quienes invirtieron en ella. Algunos de estos activistas se enriquecieron enormemente. No obstante según Jesse Colombo en su artículo *The Dot-com Bubble* (La burbuja Dot-com) del año 2004, el personal de las empresas estaba compuesto mayoritariamente por jóvenes no profesionales y las lógicas de negocios de estas empresas no eran precisas (sección *How the Internet Age Started, 2do párrafo).* Los activistas estaban invirtiendo sin analizar los riesgos en una Web aún no consolidada.

A pesar de la comodidad del servicio de compras en línea, las tiendas tradicionales aún conforman un mercado importante. Bien sea por tecnofobia o por miedo a ser victima de un fraude electrónico, ha existido un público significativo con mucha desconfianza hacia las tiendas digitales.

Los ingresos de las tiendas digitales no fueron los esperados y los precios de las acciones fueron cayendo drásticamente. De esta forma en septiembre del 2000, surgió la llamada crisis *dot-com* en donde muchas de las empresas que invirtieron en la Web quedaron en bancarrota; a pesar de ello, no hubo un estancamiento. Las tecnologías Web continuaron desarrollándose y e hizo que hoy en día se disponga de sitios que permitan la comunicación, estudio y trabajo a distancia; y continúa creciendo. Las actividades en la llamada Web 2.0 se diversifican y, en muchos casos, abarca más allá de la tecnología Web (e.g. *software peer to peer, iTunes*).

El término Web 2.0 fue utilizado por primera vez en un debate entre *O'Reilly* y *MediaLive International*. *O'Reilly* y otras compañías sobrevivientes a la crisis *dot-com* conformaron la *Web 2.0 Conference* (actualmente Web 2.0 Submit) en el año 2004. El objetivo de la aún vigente *Web 2.0 Submit* es el intercambio de

las experiencias de las empresas participantes con la finalidad de extraer cuáles son los modelos y horizontes exitosos en la Web.

Sin embargo, la definición de Web 2.0 no es clara. Tim O'Reilly en su artículo "*What Is Web 2.0*" nos menciona sobre siete principios identificados en la primera sesión de *Web 2.0 Submit*, cada uno de ellos explicados seguidamente.

La Web como plataforma

Una característica notable de las aplicaciones en la *Web 2.0* es proveer los medios para compartir información. Diversas compañías ofrecen aplicaciones que generan información consumible a través de sus *API* (*Application Programming Interface*). Otras aplicaciones llamadas *mashups* consumen dicha información para proporcionar nuevos servicios. Un ejemplo podría ser la aplicación *vici.org.* En dicha aplicación son señalados los sitios históricos del Imperio Romano en un mapa, utilizando data obtenida de la *API* de *Google Maps (O'Reilly,* 2005*).*

Las *API* también permiten la comunicación entre aplicaciones web para prestar servicios que involucran varias empresas. Por ejemplo, una tienda digital puede comunicarse con una entidad bancaria para que un cliente realice su compra. Nótese que la tienda digital y la entidad bancaria son sistemas totalmente separados, posiblemente, implementados con lenguajes y tecnología diferentes. Pero comunicables a través de una API utilizando la Web como plataforma.

Las *API* abren numerosas alternativas al mercado ofreciendo diversos servicios vía Web, siendo hoy en día posible la implementación de funcionalidades impensables al momento en que se inició la Web.

Aprovechar la inteligencia colectiva

La incorporación de la Web como un entorno social tomó cierto tiempo. En principio eran pocos los sitios que existían en la Web. Según el artículo de "What Is the Origin of Blogging?", la compañía *Netscape* catalogaba en un documento Web las páginas nuevas a través de su navegador *Navigator*, muy famoso en redor de 1995 (contribuidor de *eHow*, 1994). Posteriormente, surgió la publicación de artículos de opinión por parte de programadores, en los cuales se promocionaban sitios web de empresas particulares.

La motivación del usuario común para publicar artículos de opinión más la exposición del código fuente de los sitios web, permitió que personas no programadores pudiesen también divulgar sus pensamientos. Hecho que con el tiempo evoluciona al desarrollo de los *blogs* y, seguidamente, al nacimiento de las *redes sociales*. Ambos sitios cubrían la posibilidad de compartir e interactuar con amigos o con personas de intereses afines.

Todo este ambiente social hace que la Web crezca cada día más. La participación de la gente hace de la Web un sitio dinámico e interesante para sus usuarios y esta siempre prevalece porque la misma participación mantiene la web activa con novedades.

Las fuentes de datos son la base de la Web

Un factor importante que ha llevado al éxito a la Web en los últimos años es la gran cantidad de información disponible en ella. La Web se ha convertido en un gran centro de información que está en constante aumento. El usuario depende de ella en gran parte para acceder a la información y estar al día. Grandes empresas han invertido gran cantidad de dinero para montar una gran infraestructura que capture datos que son llevados a la Web por otras empresas importantes, tal es el ejemplo *NavTeq* (O'Reilly, 2005). *NavTeq* dispone de la infraestructura para la captura masiva de imágenes que conforman los mapas del mundo, y esta información es consumida por otras empresas como *Google Maps*. A la vez el *API* de *Google Maps* permite a otras empresas ofrecer gran cantidad de servicios utilizando sus mapas. Estos servicios le están proporcionando mucha información de interés a sus usuarios. Sin embargo, estos servicios no pudieran existir sin la fuente de datos que les otorga la información.

Por otra parte, la Web es un medio de comunicación y la participación de los usuarios permite que se genere más información. Los usuarios al calificar la información hacen que exista la motivación de publicar artículos que puedan obtener una mayor calificación. Podemos ver a la Web como una gran fuente de datos donde empresas y usuarios aportan constantemente nuevos datos.

El fin de la etapa de lanzamiento de software: el "beta permanente"

La distribución y actualización de software es muy diferente si se trata de un entorno de escritorio o de un entorno web. Las aplicaciones de escritorio suelen requerir la descarga de un parche o de un paquete de actualizaciones. En cambio las actualizaciones en las aplicaciones web suelen ser inmediatas, debido a que al estar alojadas en un servidor, los cambios por actualizaciones son percibidos directamente por el usuario. De esta manera, se acelera la velocidad de las actualizaciones de *software* en la Web acabando de cierta manera con el ciclo tradicional de lanzamiento programados de nuevas versiones de software en determinadas fechas (O'Reilly, 2005).

Algunas aplicaciones web (en su mayoría de *open source*) salen al público en versión *alpha,* siendo sólo accesibles por invitaciones (de Wikipedia, artículo *Software Realease Life*). El objetivo es de esto es poder hacer una mejor depuración de errores al prestarse el servicio en el ambiente definitivo con un público controlado.

Adicionalmente, el entorno web permite incorporar pequeñas actualizaciones y comprobar si son exitosas o no. Se puede pensar que la aplicación siempre se encuentra en un estado "beta permanente", debido a que esta está constantemente sujeta a cambios graduales y progresivos (O'Reilly, 2005). En cambio, un entorno de escritorio no es un entorno favorable para ese tipo de desarrollo ni para realizar ese tipo de pruebas.

Modelos de programación ligeros

Nuevas tendencias en las técnicas de programación han evolucionado con la Web. Los modelos de programación ligeros han liderado entre las tecnologías de desarrollo en la Web. Los servicios web por *REST* han permitido exponer vastas fuentes de datos en la Web de forma flexible y rápida. Consecuentemente, se ha acelerado el desarrollo de las aplicaciones web que utilizan estos mecanismos.

Los criterios de diseño de las aplicaciones también han evolucionado. En general, las aplicaciones de escritorio requieren ejecutarse en un computador potente. Las aplicaciones web en cambio se ejecutan en los navegadores, los cuales son aplicaciones muy simples que pueden correr en cualquier computador. Sin embargo, la velocidad de conexión a Internet que tenga un

usuario si ha sido un criterio a considerar para la construcción de aplicaciones web. Una aplicación web debe dar sus respuestas con rapidez y la velocidad de conexión influye directamente en ello. Esto ha llevado a la construcción de aplicaciones con interfaces ligeras que puedan ser cargadas en un tiempo razonable. Por lo tanto, las interfaces ligeras tienden a tener mayor éxito.

Por otra parte, la mercadotecnia ha influido en las tendencias de la Web en cuanto a las técnicas de programación. Los sitios debe ser actualizados con mucha frecuencia y el desarrollo web debe estar sujeto a estrictos plazos de tiempos. El desarrollo de nuevos métodos de desarrollo que fuesen ligeros como una alternativa (Highsmith, 2001). Entre ellos se encuentran los métodos de desarrollo ágiles, los que han sido muy populares por adaptarse a las condiciones mencionadas previamente. El uso de patrones de diseño, como el patrón *Modelo-Vista-Controlador (MVC),* de igual manera han contribuido al desarrollo de aplicaciones que, de entre muchos beneficios, sean débilmente acopladas.

El software supera el límite de un único dispositivo

A medida que se han ido incorporado nuevos dispositivos al mercado, más aplicaciones han sido agregadas a éstos. Específicamente en el mercado de los dispositivos móviles las aplicaciones web han tenido gran demanda. Actualmente, una gran cantidad de plataformas permiten ejecutar multitud de aplicaciones web, lo que contrasta sensiblemente con la antigua época cuando las aplicaciones web se ejecutaban mayoritariamente sobre el navegador *Internet Explorer* en un único dispositivo (los computadores personales). La portabilidad de las aplicaciones web es de suma importancia y dejar por fuera su correcta ejecución en los dispositivos móviles puede significar el fracaso de la misma. Por ello desde hace tiempo se puede considerar que el límite de un único dispositivo para las aplicaciones en la Web se ha superado.

No obstante, la portabilidad no es la única característica que ha cambiado. El punto clave de las aplicaciones en los dispositivos es que no tienden a ser aplicaciones empaquetadas sino servicios (O'Reilly, 2005). Un caso de estudio a mencionar en particular es la evolución de las aplicaciones *standalone*. Muchas de ellas han tenido que enriquecer sus servicios para mejorar la experiencia del usuario. Esta característica es prácticamente invisible. Aplicaciones como *iTunes* utilizan Internet para ofrecer a sus usuarios

servicios tales como la reproducción de estaciones de radio, comprar álbumes en línea, entre otras posibilidades, sin que el usuario tenga que percatarse de que los contenidos son descargados de Internet. El usuario percibe a estas aplicaciones cliente/servidor como una aplicación *standalone* cualquiera. Sin embargo, estas nos ofrecen servicios que van más allá de los datos locales del usuario.

Las nuevas tendencias en las aplicaciones van mucho más allá del avance tecnológico que representan. Muchas aplicaciones están revolucionando los medios de comunicación. Aplicaciones como *Twitter* son una muestra de esto. Actualmente, se transmite a través de *tweets* desde simples ofertas de un negocio hasta comunicados de importantes Jefe de Estados y líderes religiosos. *Twitter* se ha convertido en un tipo de periódico colectivo en donde el valor de cada cuenta *tweet* está en el número de seguidores que tiene.

La manera de distribuir las aplicaciones web también se está unificando en una sola tendencia. Muchos dispositivos incluyen buscadores (e.g. *App Store*, *Google Play*) y se está llevando a una tendencia centralizada para distribuir las aplicaciones. Estos buscadores permiten que el usuario filtren las aplicaciones por cierto criterio de búsqueda. Además los resultados se muestran en un formato único, solo son sugeridas las aplicaciones compatibles con el dispositivo, y se conoce la recepción que ha tenido el producto a través del *rating* y de los comentarios. A esto se suma el hecho de que muchas compañías desarrolladoras de software están montado sus aplicaciones en estos buscadores; posiblemente con la intención de tener presencia fuera del navegador, ya que el futuro de los navegadores es incierto.

Una experiencia de usuario enriquecida

Mientras la Web se fue convirtiendo en un lugar más amplio de múltiples fines - entre ellos el comercio y la educación – ha evolucionado la manera de presentar los sitios web. Varias tecnologías han liderado durante ese trayecto. *AJAX* (*Asynchronous JavaScript And XML*) ha sido una de las más destacadas presentando características que permiten realizar llamadas asíncronas al servidor utilizando el objeto *XMLHttpRequest*. En una solicitud *AJAX* son enviados ciertos datos, comúnmente, en *XML* o *JSON*. Posteriormente, se recibe una respuesta *AJAX* en la que se modifica el *DOM* de la página para mostrar el contenido pertinente en *HTML* y *CSS*.

No obstante, *AJAX* y *HTML* han traído muchos inconvenientes debido a las diferencias en cómo es interpretado el código en los navegadores. Por ello, la *W3C* lanza una quinta versión del lenguaje *HTML*, *HTML5*, a fin de estandarizar la forma de presentar el contenido en la Web (del sitio web de *W3C*). En esta quinta versión han sido reanalizadas todas las etiquetas llevando a la modificación de algunas y a la definición de nuevas etiquetas. Integra en la nueva etiqueta *canvas* los gráficos de vector escalable (*SVG*) y *MathML* para las fórmulas matemáticas. Su *API* ofrece varias características como *Web Storage*, *Geolocation*, gestión de base de datos local, entre otras.

Una vez tratado el tema de la tecnología, se analiza el contexto actual en la gestión de proyectos.

Gestión de Proyectos

A medida que la tecnología ha avanzado, se ha buscado automatizar los procesos de las organizaciones y crear soluciones a la medida. Ello supone un mayor desenvolvimiento y desempeño en el trabajo. Sin embargo, esa automatización de procesos no se realiza de forma arbitraria. Un análisis, un diseño y una elaboración organizada son necesarias para tal fin. Esos procesos en conjunto se denominan *proyecto*. Concretamente, como se define en Metodología de Evaluación de la Cooperación Española, “un proyecto es un conjunto autónomo de inversiones, actividades, políticas y medidas institucionales o de otra índole, diseñado para lograr un objetivo específico de desarrollo en un período determinado”. Resumidamente, para este trabajo se define *proyecto* como aquel procedimiento que involucra una serie de fases y actividades para lograr un objetivo en el rango de un tiempo definido.

Los proyectos pueden clasificarse de muchas maneras. A pesar de ello, en este documento se hablará exclusivamente de los proyectos colaborativos. Se define *proyecto colaborativo* como aquel proyecto que es elaborado utilizando el conocimiento de sus participantes de forma conjunta. De la muy conocida frase “La unión hace la fuerza” podría verse un lema oculto para los proyectos colaborativos. Debido a que se espera que si se dispone de las personas adecuadas, el resultado de un proyecto colaborativo podrá abarcar mucho más que el resultado de varios individuos que trabajan separadamente.

Los proyectos colaborativos también posibilitan que éstos sean realizados por un conjunto multi-disciplinario. Interrelacionar disciplinas para el desarrollo

de un proyecto podría hacer que sean resaltados los pros y disminuidos los contras de cada una de ellas.

Se considera que la realización de un proyecto implica el uso de los siguientes componentes:

1. Línea de Tiempo

   La definición de proyecto indica que éste debe realizarse en un rango de tiempo definido. Esto lleva a que todos los procesos de un proyecto se programen en función del tiempo. Por lo tanto, la visualización del proyecto a través de componentes cronológicos es necesaria y las líneas de tiempo permiten dicha visualización.

   Un línea de tiempo es un gráfico que nos muestra un campo de valores (eventos) que ocurrieron en una fecha específica. Aunque es frecuente que solo haya un evento por fecha, la ocurrencia de eventos puede ser múltiple.

2. Diagramas de *Gantt*:

   El diagrama de *Gantt* es un esquema gráfico que modela la programación de las tareas en un proyecto. Fue creada por Henry L. Gantt en 1917.

3. Recursos

   La elaboración de un proyecto implica el uso de cierta cantidad de recursos humanos y físicos. Según las dimensiones del proyecto, existen una cantidad mínima y máxima ideal de recursos. La asignación mínima requerirá mayor tiempo de desarrollo y la máxima se considera el ideal. Superar la cantidad máxima puede representar pérdidas o sub-utilización de recursos.

4. Herramientas

   Una herramienta es cualquier componente que facilita la realización de una tarea. El uso adecuado de herramientas permite sacar mayor provecho de los recursos y obtener resultados satisfactorio a corto plazo.

En el campo de la Ingeniería del Software, todo proyecto es modelado y desarrollado a través de un método de desarrollo. Existen muchos métodos de desarrollo, sin embargo es posible identificar ciertos aspectos comunes entre ellos. Se listan y se describen a continuación:

1. Fases: Se trata de un periodo de tiempo destinado a la realización de un conjunto de actividades de fin común. Ese conjunto de actividades conforma una etapa del proyecto que puede ser análisis, diseño, implementación, mantenimiento, entre otras.

2. Actividades: Se refiere a un conjunto de asignaciones necesarias para cumplir cierto requerimiento y puede que abarque varias fases. Las actividades se definen dentro de una fase.

3. Tareas: Son aquellos conjuntos de trabajo para completar el requerimiento especificado por una actividad. Cada tarea implica la generación de un documento que indique que se realizó en ella.

MARCO APLICATIVO

Una vez que fueron presentados varios aspectos teóricos, se prosigue con la elaboración de una aplicación Web que permita enriquecer una aplicación de la Web 2.0 para lograr mayor productividad en los desarrollos colaborativos. En primera instancia, se debe definir claramente qué se quiere realizar.

Descripción de la Situación Actual

Una red social *open source* llamada *Oxwall* fue propuesta en el seminario de Maspons (2012) para ser utilizada como medio de comunicación para algunas asignaciones del Centro de Ingeniería del Software y Sistemas (ISYS). Esta aplicación actúa al estilo de un portal Web, en donde pueden ser instaladas y configuradas ciertas aplicaciones (e.g. *blogs*, administrador de grupos, etc) a conveniencia de los administradores de la red. Lo interesante de la red *Oxwall* es que permite que agregar aplicaciones realizadas por aquellos grupos de desarrolladores que establezca ISYS.

Las asignaturas que utilizan la red como herramienta tienen actualmente la necesidad de tener una aplicación para la gestión de proyectos. Dicha aplicación no se encuentra disponible al momento en *Oxwall* (ver Anexo *Framework Oxwall* para mayor información).

Con el presente Trabajo Especial de Grado, se desea aportar una aplicación *plug-in open source* dentro de la red social *Oxwall* con tecnología Web 2.0 que permita el seguimiento de proyectos en ambientes colaborativos. Este aporte permitirá el uso de nuevas tecnologías que de mejores soluciones para la necesidad expresada.

Descripción Tecnológica

Para la realización de la aplicación se debe utilizar algún método de desarrollo de software que guie el desarrollo de la misma.

Seguidamente, son presentadas las características que tendrá el desarrollo de la aplicación para seleccionar aquel método que se ajuste mejor a las condiciones:

- Se dispone de plazo de tiempo limitado para el desarrollo del producto. El mismo debe estar planificado dentro del periodo establecido de 16 semanas en promedio que tiene un semestre universitario.
- Los requerimientos de la aplicación están sujetos a posibles cambios.
- Las actividades para un desarrollo web deben preferiblemente tener una duración corta de pocas horas de trabajo.
- Se requiere entregar avances de la aplicación cada 1 ó 2 semanas.

Analizando el escenario a tratar, se considera que el método *Scrum* (ver Apartado SCRUM para más detalles de este método) presenta las características apropiadas para el alcance y duración de este trabajo.

Para levantarse el documento *Product Backlog* del método SCRUM, debe previamente capturarse de requerimientos del sistema.

Captura de Requerimientos

Para realizar la captura de requerimientos, se decide utilizar análisis de dominio. Se quiere utilizar como referencia las líneas de tiempo en contexto generales y los programas de gestión de proyectos en la Web, por lo que se necesita realizar dos análisis de dominio para cada caso.

Análisis de Dominio en Líneas de Tiempo

El análisis de dominio requiere que se seleccione un conjunto de aplicaciones populares presentes en la Web para capturar las características generales de este tipo de componente.

Las siguientes aplicaciones fueron seleccionadas:

TimelineJS: (en la Figura 1 se presenta una captura de pantalla de la interfaz)

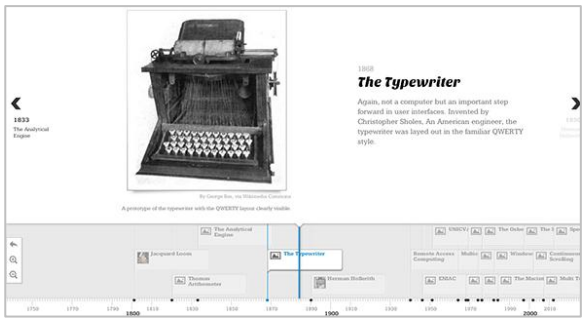


Figura 1 – Pantalla de la aplicación web TimelineJS

Dipity: (la Figura 2 presenta una captura de pantalla de la interfaz)

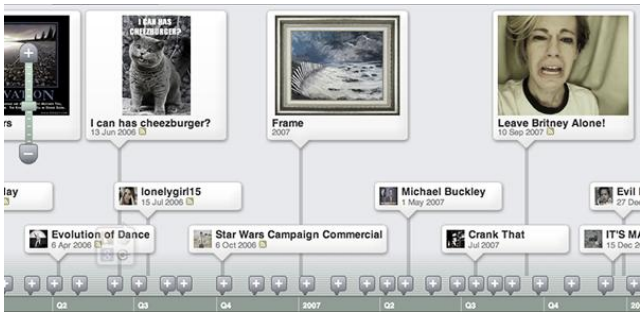


Figura 2 - Pantalla de la aplicación web Dipity

Time Toast: (en la Figura 3 se presenta una captura de pantalla de la interfaz)

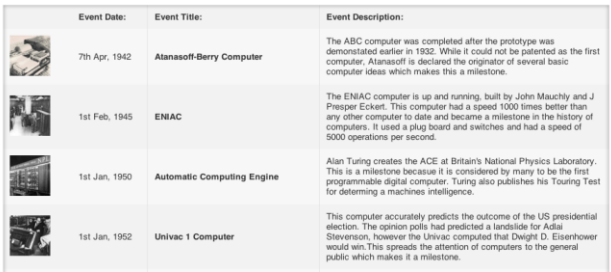


Figura 3 - Pantalla de la aplicación web Time Toast

Se extrajo un conjunto de características presentes en las aplicaciones seleccionadas y, en la Tabla Comparativa, se realiza una comparación entre las aplicaciones con respecto a la selección.

Tabla 1 – Características de las aplicaciones de líneas de tiempo

Características	*TimelineJS*	*Dipity*	*Time Toast*
Introducción	si	no	no
Navegación lineal	si	si	no
Navegación por eventos	si	si	si
Navegación por palabras clave	no	no	si

Línea siempre visible	si	no	no
Comentarios en eventos	no	si	no
Escalabilidad	Permite aplicar zoom en las líneas de tiempo	si	no
Cambio de tipo de visualización	no	no	si

Acorde la comparativa realizada y en consideración de los objetivos de la aplicación, se tiene lo siguiente:

- Presencia de texto introductorio: Un resumen acerca de qué trata el proyecto se considera necesario para orientar rápidamente a los visitantes y/o participantes.

- Permitir navegación lineal: Las actividades pueden estar ejecutándose en paralelo por lo que una navegación lineal, no parece conveniente.

- Permitir navegación arbitraria: Es necesario poder consultar cualquier actividad en todo momento.

- Permitir navegación por palabras clave: Esta característica es conveniente; sin embargo, su implementación requiere realizar un buscador que queda fuera del alcance de este trabajo.

- Línea siempre visible: Visualizar la línea temporal siempre durante la navegación sobre un proyecto, puede resultar contraproducente porque dicha línea consume demasiado espacio en la pantalla.

- Comentarios en las Actividades: Las actividades deben realizarse en equipo de 2 ó 3 personas, por lo que es deseable que ellos puedan disponer de un muro. El muro se trata de un espacio en que son visualizados aquellos mensajes que sean publicados por los participantes de la actividad

- Escalabilidad: La escabilidad de la línea de tiempo puede considerarse como una característica deseable, sin embargo es de baja prioridad porque no agrega productividad a la aplicación.

- Cambio de tipo de visualización: Las líneas de tiempo son un tipo de visualización del proyecto, sin embargo visualizaciones más tradicionales podría facilitar la exposición de datos que no son presentados en la línea.

En resumen, se seleccionan las siguientes características en base al análisis realizado:

1. Descripción del Proyecto (originalmente referido como texto introductorio)
2. Navegación arbitraria por las actividades
3. Línea de tiempo sólo visible en la consulta del proyecto
4. Visualización alternativa de fases y actividades
5. Permitir comentarios en las actividades

Análisis de Dominio en Gestión de Proyectos

Seguidamente, se realiza un análisis de dominio sobre dos sistemas para la gestión de proyectos:

GanttProject (recuperado de http://www.ganttproject.biz/);

AceProject (recuperado de http://www.aceproject.com/)

La meta es identificar las características generales de este tipo de sistema y generar una tabla comparativa de los resultados obtenidos, como se muestra en la Tabla 2.

Tabla 2 – Características de aplicaciones de gestión de proyectos

Características	*GanttProject*	*AceProject*
Crear estructuras para el desglose del trabajo	si	si
Definir metas	si	no
Asignar participantes a tareas	si	si
Visualizar el proyecto en un diagrama de Gantt	si	si
Generar reportes	si	no

Ver el progreso de todas las tareas	no	si
Visualizar detalles de una tarea (status, porcentaje de progreso, prioridad, etc.)	no	si
Adjuntar documentos a tus tareas	no	si

Considerando los datos obtenidos en la Tabla 2 y los objetivos de presente trabajo, se tomaron las siguientes decisiones:

1. La característica “Crear estructuras para el desglose del trabajo” es necesaria para planificar el proyecto en diferentes etapas.

2. La característica “Definir metas” es necesaria, sin embargo debe llevarse de manera implícita a través de las tareas del proyecto.

3. La característica “Asignar participantes a tareas” es necesaria, ya que se trata de gestionar proyectos colaborativos.

4. La característica “Visualizar el proyecto en un diagrama de Gantt” es necesaria. Se considera que la solución de representar actividades en paralelo mediante un diagrama de Gantt es de práctica común en los sistemas de gestión de proyectos, debido que dicha característica fue encontrada en muchos de ellos.

5. La característica “Generar reportes” es opcional, ya que da un agregado, mas no es un objetivo de este trabajo.

6. La característica “Ver el progreso de todas las tareas” no es necesaria en el contexto de este trabajo, ya que la tarea se realiza por un documento y la elaboración de éste no es realizada en el sistema. Esto hace que no se conozca el avance objetivo de la tarea.

7. La característica “Visualizar detalles de una tarea” es necesaria. La tarea puede ser suficientemente compleja como para requerir una vista exclusiva.

8. La característica “Adjuntar documentos a tus tareas” es necesaria. Es uno de los objetivos del trabajo.

En resumen, fueron seleccionadas las siguientes características:

1. Crear estructuras para el desglose del trabajo
2. Asignar participantes a tareas
3. Visualizar el proyecto en un diagrama de Gantt
4. Visualizar detalles de una tarea
5. Adjuntar documentos a tus tareas

Requerimientos de la Aplicación

Analizando los resultados obtenidos en los análisis de dominios realizados, se capturan los siguientes requerimientos:

1. Realizar modelado de la aplicación
2. Gestionar proyectos
3. Planificar el proyecto en fases
4. Gestionar métodos de desarrollo de manera que un proyecto pueda seguir un método existente.
5. Asignar actividades en las fases
6. Describir el proyecto de forma breve y clara.
7. Visualizar fases y actividades en una línea de tiempo
8. Visualizar fases y actividades en listados
9. Presentar actividades en un diagrama de Gantt
10. Poder navegar en las actividades presentadas en el diagrama de Gantt
11. Línea de tiempo sólo visible en la consulta del proyecto
12. Gestionar fases, actividades y tareas.
13. Gestionar usuarios y responsables de la aplicación.
14. Permitir comentarios en las actividades

15. Asignar participantes a tareas

16. Visualizar detalles de una tarea

17. Permitir realizar entregas de tareas por medio de documentos.

Inicio del Desarrollo de la Aplicación

Tomando en cuenta los requerimientos de la aplicación, se presenta el documento *Product Backlog* en la Figura 6. El orden expuesto en la lista marca la prioridad de la asignación. Se recuerda que en este trabajo es el grupo de desarrolladores es sólo uno, por tanto no hay realización de asignaciones en paralelo.

Tabla 3 – Asignaciones de la aplicación

# Item	Descripción
1	Realizar modelado de la aplicación a través del UML
1.1	Realizar modelo de entidad/relación
1.2	Realizar diagrama de casos de usos
1.3	Realizar diagrama de actividades
1.4	Realizar diagrama de clases
2	Diseñar aplicación
2.1	Diseñar componentes básicos
3	Gestionar proyectos
3.1	Realizar diagrama de clases para los proyectos
3.2	Crear/Editar proyecto
3.3	Listar proyectos
3.4	Eliminar proyecto
4	Gestionar modelos de desarrollo
4.1	Crear/editar modelos
4.2	Listar modelos
4.3	Eliminar modelos
4.4	Crear proyectos que siguen modelos de desarrollo
5	Gestionar fases
5.1	Crear/editar fases

# Item	Descripción
5.2	Listar fases
5.3	Eliminar fases
3.5	Crear proyectos planificados por fases
3.6	Actualizar Punto 2.3 con eliminar implícito de fases del proyecto
6	Gestionar actividades
6.1	Crear/editar actividades
6.2	Listar actividades
6.3	Eliminar actividades
3.7	Actualizar Punto 2.6 con eliminar implícito de las actividades del proyecto
7	Programar proyecto
7.1	Solicitar fechas de inicio y fin de fases
7.2	Solicitar fechas de inicio y fin de actividades
2.2	Diseñar componentes complejos
7.3	Mostrar calendario de fechas
7.4	Permitir gestionar proyecto desde la vista de “Planificar Proyecto”
8	Gestionar tareas
8.1	Crear/editar tarea
8.2	Listar tareas
8.3	Eliminar tarea
3.8	Actualizar Punto 2.7 con eliminar implícito de las tareas del proyecto
9	Gestionar usuarios
9.1	Crear/editar usuario
9.2	Listar usuarios
9.3	Eliminar usuario
10	Gestionar participantes
10.1	Crear/Editar particpante
10.2	Listar particpantes
10.3	Eliminar particpantes
3.9	Actualizar Punto 2.8 con eliminar implícito de los responsables del proyecto

# Item	Descripción
10.4	Actualizar Punto 9.1 para asignar responsables de tarea
11	Realizar entrega de tarea
11.1	Adjuntar documento
11.2	Guardar documento en el directorio adecuado
11.3	Descargar documento
10.5	Aplicar mecanismos de autorización sobre las operaciones de las tareas
10.6	Aplicar mecanismos de autorización sobre las operaciones de las actividades y proyectos
6.4	Verificar status de las actividades por la comprobación de las fechas de inicio y fin respectivas
6.5	Notificar sobre los cambios de status de las actividades.
12	Realizar Pruebas de la aplicación
13	Implementar la integración definitiva de la aplicación como plugin
13.1	Implementar la instalación de la aplicación como plug-in
13.2	Implementar la desinstalación de la aplicación como plug-in
13.3	Verificar que el funcionamiento de la aplicación se preserva

Figura 4

Ciclo de Desarrollo de la Aplicación

Se inicia el ciclo de desarrollo del proyecto. Se indicarán las actividades de cada iteración seguidamente hasta que culmine el desarrollo de la aplicación.

Se documenta cada una de las asignaciones *Product Backlog* seguidamente.

Asignación #1 - Realizar Modelo de Entidad/Relación

El modelo de entidad/relación permite visualizar las entidades que deben conformar la aplicación y tener una mejor comprensión de cómo se relacionan los objetos involucrados.

El modelo presenta dos conjuntos separados (ver Figura 8). En la parte superior se representa el conjunto relacionado con la definición del Proceso de Desarrollo. La parte inferior representa un proyecto en desarrollo.

El Proceso de Desarrollo consta de 3 entidades: Método de Desarrollo, Fase y Actividad. Estas entidades pretenden representar el concepto de un Modelo de Desarrollo que se planifica en fases y éstas, a su vez, en actividades.

El Proyecto en Desarrollo por su parte contiene las entidades Proyecto, Proyecto-Fase, Proyecto-Actividad y Tarea, los que representan la estructura necesaria para planificar un proyecto. Este diagrama da soporte a que la planificación de un proyecto no es arbitraria. El Método de Desarrollo da una plantilla de cómo debe crearse, mientras que el Proyecto representa la implantación en de un proceso de desarrollo en un problema determinado para generar una solución. El detalle de cada una de estas entidades y de sus relaciones son descritas a continuación.

Entidades

Método de Desarrollo: (nombre, descripción, status)

Un método de desarrollo generaliza la planificación y organización de un proyecto, encapsulando los datos generales del modelo (i.e. nombre, descripción). Al crear un proyecto puede especificarse el método a seguir.

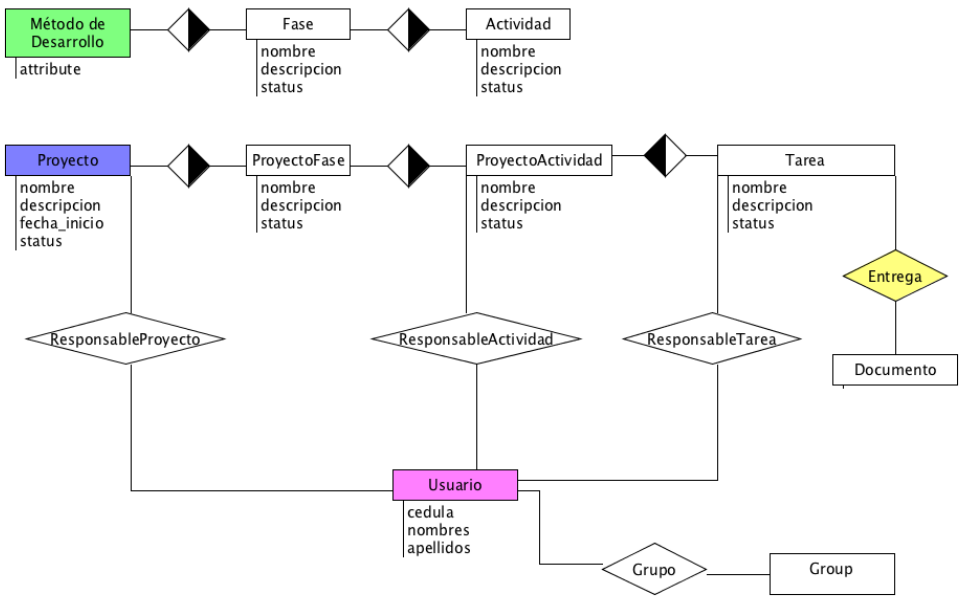


Figura 5 – Modelo E/R de la aplicación

Fase: (ID del Proceso, nombre, descripción, duración estimada)

Una fase define una etapa de desarrollo del método (e.g. análisis, diseño, etc) en la que se establece un plazo de tiempo para cumplir cierto objetivo.

Actividad: (ID de la Fase, nombre, descripción, duración estimada)

Una actividad agrupa un conjunto de tareas para lograr cierta meta. La meta tiene igual o menor alcance que la del objetivo de la fase a la que pertenece.

Proyecto: (nombre, descripción, fecha de inicio, status)

Un proyecto representa el seguimiento del procedimiento a realizar para completar un producto. En el proyecto se lleva una planificación y coordinación de un conjunto de actividades, organizadas en fases.

Proyecto-Fase: (ID del Proyecto, nombre, descripción, duración, fecha de inicio estimada, fecha fin estimada, fecha de inicio, fecha fin, status)

Un *Proyecto-Fase* representa la particularización de una *fase* en un proyecto. A diferencia de la fase, un *proyecto-fase* almacena cuáles son los plazos de tiempos para finalizar la misma.

Para proporcionar la personalización de cada elemento del proyecto, se permite la edición de todos los atributos sin afectar el método de desarrollo original.

Proyecto-Actividad: (ID del Proyecto-Fase, nombre, descripción, duración, fecha de inicio estimada, fecha fin estimada, fecha de inicio, fecha fin, status)

Un *Proyecto-Actividad* representa la particularización de una *Actividad* de un proyecto. A diferencia de la *Actividad*, un *Proyecto-Actividad* almacena cuáles son los plazos de tiempos para finalizar la misma.

Al igual que *Proyecto-Fase*, se permite la edición de todos los atributos sin afectar el método de desarrollo original.

Los status de un Proyecto-Fase son Activo, Finalizado, Suspendido.

Tarea (ID del Proyecto-Actividad, nombre, descripción, duración, fecha de inicio estimada, fecha fin estimada, fecha de inicio, fecha fin, status)

Asignación que enuncia un trabajo por realizar. Un trabajo podría ser, por ejemplo, la elaboración de un modelo de diagrama UML.

Documento (Nombre del Documento, Calificación Cualitativa del Documento)

Resultado de una tarea emitido por un responsable de la misma.

Usuario (ID del Usuario Oxwall, Cédula de Identidad, Nombres, Apellidos, Status)

Registro particularizado de un usuario de la red.

Grupo (ID del Group, ID del Usuario, Status)

Agrupación de usuarios bajo cierto criterio.

Relaciones

Entrega (ID de la Tarea, Ubicación del documento, Fecha, Status)

Registro que especifica a qué tarea pertenece y en qué fecha se realizó la entrega.

Responsable-Proyecto (ID del Proyecto, ID del Usuario, Status)

Responsable-Actividad (ID del Proyecto-Actividad, ID del Usuario, Status).

Responsable-Tarea (ID de la Tarea, ID del Usuario, Status)

Cada entidad *Responsable* encapsula un perfil definido para los usuarios del sistema. El *Responsable del Proyecto* se encarga de definir la programación de plazos y fechas límites y edición del proyecto. Este perfil lo adquiere el usuario que creó el proyecto. El *responsable de una actividad* es el encargado de crear/editar tareas, definir sus responsables y evaluar las entregas que realicen dichos responsables. El *responsable de una tarea* es aquel que fue asignado para realizar una tarea. Puede realizar entregas al subir un documento al sistema.

Asignación #2 - Realizar Diagrama de Casos de Usos

El modelo de casos de usos permite describir las funcionalidades que debe tener la aplicación.

Usuarios

Se definen los actores que intervienen en la aplicación:

Administrador del Sistema: Se dispondrán de uno o más administradores de la aplicación, los cuales se encargan de realizar un mantenimiento general del sistema (e.g. gestión de usuarios, respaldar proyectos, etc.)

Administrador de Proyecto: Usuario encargado de un Proyecto. El Administrador o Responsable de un proyecto será el usuario que cree el proyecto y podrá administrar y planificar el proyecto.

Responsable de Actividad: Usuario que gestiona una actividad. El responsable de una actividad se encargará de gestionar las tareas, designar responsables de tareas y corregir las entregas que ellos realicen de las tareas.

Responsable de Tarea: Usuario que realiza las tareas que se le hayan asignado.

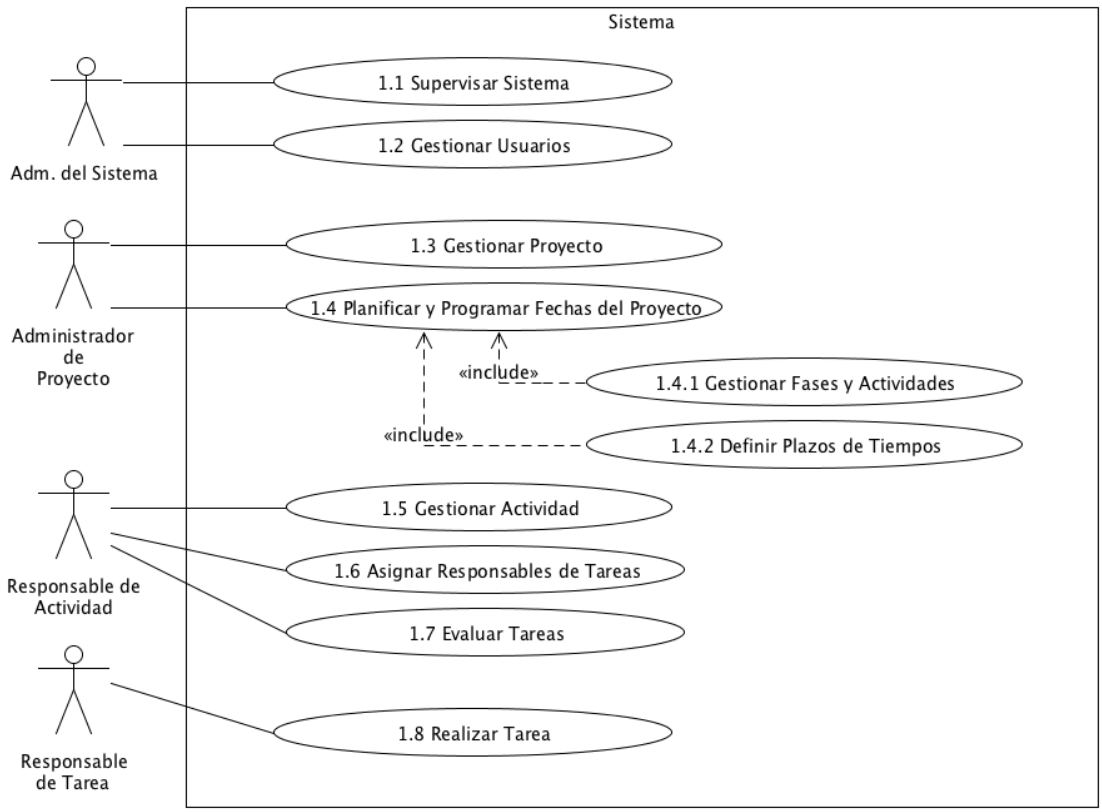


Figura 6 – Diagrama de Casos de Usos de la aplicación

Descripción de los Casos de Uso

A continuación, se describe cada caso de uso.

CU-1 Supervisar Sistema

Descripción: El Administrador del Sistema deberá mantener el sistema correctamente, esto incluye tareas como: realizar respaldos, eliminar registros innecesarios, archivar proyectos.

CU-2 Gestionar Usuarios

Descripción: El administrador del sistema gestiona los usuarios de la aplicación. La gestión de usuarios incluye el manejo de todas las respectivas operaciones *CRUD* (*Create, Read, Update, Delete*).

CU-2.1 Crear Usuario

Pre-condición: Esta operación solo puede ser realizada por el administrador del sistema.

Secuencia Principal:

1. El administrador solicita crear un usuario nuevo.
2. El sistema solicita los datos del usuario.
3. El administrador introduce la información del usuario.
4. El sistema almacena el usuario.

Secuencias alternativas:

1. Si el administrador cancela la operación de crear, el sistema redirecciona al caso de uso CU-1.2.3 Listar Usuarios.
2. Si el administrador introduce algún dato incorrectamente, el sistema muestra un mensaje de error. Se repite el paso 2.

Post-Condición: Hay un usuario nuevo en el sistema.

CU-2.2 Mostrar Usuario

Pre-condición: Esta operación solo puede ser realizada por el administrador del sistema.

Secuencia Principal:

1. El administrador solicita ver a un usuario.
2. El sistema muestra los datos del usuario.

Secuencias alternativas:

Si alguno de los datos del usuario está en un estado inválido, cuando el sistema muestra los datos del usuario. El sistema debe ser robusto y mostrar solo los datos válidos.

Post-Condición: Ninguna.

CU-2.3 Listar Usuarios

Pre-condición: Esta operación solo puede ser realizada por el administrador del sistema.

Secuencia Principal:

1. El administrador solicita ver el listado de los usuarios.

2. El sistema muestra el listado de los usuarios

Secuencias alternativas:

Si alguno de los registros está en un estado inválido, debe omitirse dicho registro y mostrarse el resto del listado.

Post-Condición: Ninguna.

CU-2.4 Editar Usuario

Pre-condición: Esta operación solo puede ser realizada por el administrador del sistema.

Secuencia Principal:

1. El administrador solicita editar determinados datos del usuario.
2. El sistema muestra los campos editables de los datos del usuario.
3. El administrador edita los datos del usuario.
4. El sistema almacena la información ingresada

Secuencias alternativas:

1. Si el administrador ingresa datos inválidos, debe aparecer un mensaje de error.
2. Si ocurre un error al intentar almacenar los datos, el administrador debe intentar en otro momento. Si el problema persiste debe verificarse posibles fallas en los servidores.

Post-Condición: Ninguna.

CU-2.5 Eliminar Usuario

Pre-condición: Esta operación solo puede ser realizada por el administrador del sistema.

Secuencia Principal:

1. El administrador solicita eliminar a un usuario.
2. El sistema solicita confirmar la operación de eliminar.

3. Si el administrador confirma, el sistema elimina el usuario. En caso contrario, regresa a la vista inicial de cuando realizó la solicitud.

Secuencias alternativas:

Si el administrador quiere eliminar el registro pero existe una dependencia de otros registros, el usuario debe eliminar dichas dependencias antes.

Post-Condición: El usuario fue eliminado.

CU-3 Gestionar Proyecto

Descripción: La gestión de proyectos incluye el manejo de todas las operaciones CRUD (Create, Read, Update, Delete).

CU-3.1 Crear Proyecto

Pre-condición: El administrador del sistema y responsable del proyecto pueden realizar la gestión de proyectos de la aplicación.

Secuencia Principal:

1. El administrador solicita crear un proyecto nuevo.
2. El sistema solicita los datos del proyecto.
3. El administrador introduce la información del proyecto. Puede que indique que se va a seguir determinado método de desarrollo.
4. El sistema almacena los datos del proyecto y genera la estructura del proyecto.

Secuencias alternativas:

1. Si el administrador cancela la operación de crear, el sistema redirecciona al caso de uso CU-1.2.3 Listar Proyectos.
2. Si el administrador deja algún dato obligatorio vacio, el sistema muestra un mensaje de error. Se repite el paso 2.

Post-Condición: Hay un proyecto nuevo en el sistema.

CU-3.2 Mostrar Proyecto

Pre-condición: Ninguna

Secuencia Principal:

1. El usuario solicita ver a un proyecto.
2. El sistema muestra los datos del proyecto

Secuencias alternativas:

Si alguno de los datos del proyecto está en un estado inválido en la vista de mostrar usuario, el sistema debe ser robusto y mostrar sólo los datos válidos.

Post-Condición: Ninguna.

CU-3.3 Listar Proyectos

Pre-condición: Ninguna

Secuencia Principal:

1. El administrador solicita ver el listado de los proyectos.
2. El sistema muestra el listado de los proyectos

Secuencias alternativas:

Si alguno de los registros está en un estado inválido, debe omitirse dicho registro y mostrarse el resto del listado.

Post-Condición: Ninguna.

CU-3.4 Editar Proyecto

Pre-condición: Esta operación solo puede ser realizada por el administrador del sistema o responsable de ese proyecto.

Secuencia Principal:

1. El administrador solicita editar determinados datos del proyecto.
2. El sistema muestra los campos editables de los datos del proyecto.
3. El administrador edita los datos.
4. El sistema almacena la información

Secuencias alternativas:

1. Si el administrador ingresa datos inválidos, debe aparecer un mensaje de error.

2. Si ocurre un error al intentar almacenar los datos, el administrador debe intentar en otro momento. Si el problema persiste debe verificarse posibles fallas en los servidores.

Post-Condición: Ninguna.

CU-3.5 Eliminar Proyecto

Pre-condición: Esta operación solo puede ser realizada por el administrador del sistema o responsable de ese proyecto.

Secuencia Principal:

1. El usuario solicita eliminar un proyecto.

2. El sistema pide confirmar la operación de eliminar.

3. Si el usuario confirma, el sistema elimina el proyecto. En caso contrario, regresa a la vista anterior a la solicitud.

Secuencias alternativas:

Si el usuario quiere eliminar el registro pero existe una dependencia con otros registros del sistema, el usuario debe eliminar dichas dependencias antes.

Post-Condición: El proyecto fue eliminado.

CU-4 Planificar y Programar Fechas del Proyecto

Descripción: El responsable del proyecto se encarga de la planificación y programación de proyecto. La planificación se refiere a administración de fases y actividades del proyecto. La programación se refiere a la definición de plazos de tiempos de cada fase y actividad.

El usuario visualiza el listado de las fases en la vista correspondiente a este caso de uso.

CU-4.1 Definir Plazos de Tiempos

Descripción: Las fases y actividades tienen fechas de inicio y fin que deben ser definidas por el responsable del proyecto.

Pre-condición: Esta operación solo puede ser realizada por el responsable de ese proyecto.

Secuencia Principal:

1. El usuario dispone de las fechas para las que quiere programar el proyecto. Solicita al sistema dicha información.
2. El sistema muestra un calendario y los campos para ingresar las fechas de cada fase u actividad.
3. Si el usuario ingresa la información y la guarda.
4. El sistema verifica los datos y los almacena.

Secuencias alternativas:

Si el usuario ingresa una programación inválida, el sistema debe pedir que se corrija la situación.

Post-Condición: El proyecto fue programado.

CU-4.2 Gestionar Fases

Descripción: El responsable del proyecto puede agregar/eliminar fases del proyecto.

CU-4.2.1 Crear Fase

Pre-condición: Solo el responsable del proyecto puede realizar esta operación.

Secuencia Principal:

1. El administrador solicita crear una fase nueva.
2. El sistema solicita los datos de la fase.
3. El administrador introduce la información solicitada.
4. El sistema almacena la fase.

Secuencias alternativas:

1. Si el administrador cancela la operación de crear, el sistema redirecciona al caso de uso CU-1.4 Planificar y Programar Fechas del Proyecto.

2. Si el administrador introduce algún dato incorrectamente, el sistema muestra un mensaje de error. Se repite el paso 2.

Post-Condición: El proyecto dispone de otra fase.

CU.4.2.2 Editar Fase

Pre-condición: Solo el responsable del proyecto puede realizar esta operación.

Secuencia Principal:

1. El usuario solicita editar determinados datos de una fase.
2. El sistema muestra los campos editables de los datos de la fase.
3. El usuario edita los datos.
4. El sistema almacena la información

Secuencias alternativas:

1. Si el administrador ingresa datos inválidos, debe aparecer un mensaje de error.
2. Si ocurre un error al intentar almacenar los datos, el administrador debe intentar en otro momento. Si el problema persiste debe verificarse posibles fallas en los servidores.

Post-Condición: Ninguna.

CU-4.2.3 Eliminar Fase

Pre-condición: Solo el responsable del proyecto puede realizar esta operación.

Secuencia Principal:

1. El administrador solicita eliminar una fase.
2. El sistema solicita confirmar la operación de eliminar.
3. Si el administrador confirma, el sistema elimina la fase y redirecciona al CU-1.4 Planificar y Programar Fechas del Proyecto. Si rechaza la confirmación, regresa a la vista inicial de cuando realizó la solicitud.

Secuencias alternativas:

Si el usuario quiere eliminar el registro y ocurre un error, debe intentar más tarde. Si el problema persiste, debe notificarle al administrador del sistema.

Post-Condición: La fase del proyecto fue eliminada.

CU-4.3 Gestionar Actividad

Descripción: El responsable de la actividad puede realizar cualquier operación CRUD sobre una actividad o tarea.

CU-4.3.1 Crear Actividad

Pre-condición: Solo el responsable del proyecto puede realizar esta operación.

Secuencia Principal:

1. El usuario solicita crear una actividad nueva.
2. El sistema solicita los datos de la actividad.
3. El usuario introduce la información de la actividad.
4. El sistema almacena la actividad.

Secuencias alternativas:

1. Si el usuario cancela la operación de crear, el sistema redirecciona al caso de uso CU-1.4 Planificar y Programar Fechas del Proyecto.
2. Si el usuario introduce algún dato incorrectamente, el sistema muestra un mensaje de error. Se repite el paso 2.

Post-Condición: El proyecto dispone de una actividad nueva.

CU-4.3.2 Mostrar Actividad

Pre-condición: Solo el responsable del proyecto puede realizar esta operación.

Secuencia Principal:

1. El usuario solicita ver una actividad.
2. El sistema muestra los datos de la actividad

Secuencias alternativas:

Si alguno de los datos de la actividad está en un estado inválido en la vista de mostrar actividad, el sistema debe ser robusto y mostrar sólo los datos válidos.

Post-Condición: Ninguna.

CU-4.3.4 Editar Actividad

Pre-condición: Solo el responsable del proyecto puede realizar esta operación.

Secuencia Principal:

1. El usuario solicita editar determinados datos de una actividad.
2. El sistema muestra los campos editables de los datos de una actividad.
3. El administrador edita los datos.
4. El sistema almacena la información

Secuencias alternativas:

1. Si el administrador ingresa datos inválidos, debe aparecer un mensaje de error.
2. Si ocurre un error al intentar almacenar los datos, el administrador debe intentar en otro momento. Si el problema persiste debe verificarse posibles fallas en los servidores.

Post-Condición: La actividad fue editada.

CU-4.3.5 Eliminar Actividad

Pre-condición: Solo el responsable del proyecto puede realizar esta operación.

Secuencia Principal:

1. El usuario solicita eliminar una actividad.
2. El sistema solicita confirmar la operación de eliminar.
3. Si el administrador confirma, el sistema elimina la actividad. En caso contrario, regresa a la vista inicial de cuando realizó la solicitud.

Secuencias alternativas:

Si ocurre un error al intentar eliminar los datos, el usuario debe intentar en otro momento. Si el problema consultar con el administrador.

Post-Condición: El usuario fue eliminado.

CU-5 Asignar Responsables de Tareas

Descripción: El responsable de la actividad debe designar quiénes realizarán las tareas que tenga la actividad.

Pre-condición: Solo el responsable de la actividad puede realizar esta operación.

Secuencia Principal:

1. El usuario solicita asignar una tarea.
2. El sistema solicita indicar a quien desea asignar.
3. El usuario busca a la persona y solicita la asignación
4. El sistema asigna la persona a la tarea y redirecciona a Listar Responsables.

Secuencias alternativas:

Si ocurre un error al intentar asignar a una persona, debe consultar con el administrador.

Post-Condición: La persona es asignada para realizar una tarea.

CU-6 Descargar Tareas

Descripción: Cualquier usuario puede descargar una tarea subida.

Pre-condición: Ninguna

Secuencia Principal:

1. El usuario solicita descargar una tarea.
2. El sistema manda un solicitud del recurso solicitado. El navegador resuelve cómo se despliega el recurso (e.g. diálogo para abrir o guardar)

Secuencias alternativas:

Si el sistema no encontró el recurso, el usuario debe consultarlo con el administrador.

CU-7 Evaluar Tareas

Descripción: Toda tarea realizada deberá ser evaluada por el responsable de la actividad.

Pre-condición: Solo el responsable de la actividad puede realizar esta operación.

Secuencia Principal:

1. El usuario solicita evaluar una tarea.
2. El sistema solicita indicar los resultados de la evaluación.
3. El usuario ingresa los datos.
4. El sistema almacena los datos y redirecciona a Mostrar Tarea.

Secuencias alternativas:

Si ocurre un error al intentar evaluar una tarea, el usuario debe consultar con el administrador.

CU-8 Realizar Tarea

Descripción: El usuario debe realizar la tarea que le fue asignada en el tiempo estipulado. La realización de una tarea implica la realización de un documento que, al finalizar, debe subir al sistema.

Pre-condición: Solo el responsable de la tarea puede realizar esta operación.

Secuencia Principal:

1. El usuario realizó la tarea en un documento. solicita evaluar una tarea.
2. El sistema solicita indicar los resultados de la evaluación.
3. El usuario ingresa los datos.
4. El sistema almacena los datos y redirecciona a Mostrar Tarea.

Secuencias alternativas:

Si ocurre un error al intentar evaluar una tarea, el usuario debe consultar con el administrador.

Asignación #3 - Realizar Diagrama de Actividades

El diagrama de actividades permite analizar y modelar cuál es el flujo de acciones que debe tener una aplicación o un sistema.

En la Figura 7, se observa que el proceso puede iniciar con un *nuevo proyecto* o con un *nuevo método*. En caso de un *nuevo proyecto*, el usuario debe especificar un nombre, descripción, y seleccionar el método de desarrollo que seguirá para la planificación del proyecto. Luego, deben ser definidos los usuarios y los perfiles que tiene dichos usuarios en el proyecto.

Seguidamente, se pasa a la planificación del proyecto, el líder del proyecto programa las fechas de cada fase y actividad.

Los usuarios supervisores son quienes crean las tareas del proyecto y corrigen las entregas correspondientes de las tareas. Las tareas deben ser realizadas antes de que finalice el plazo de tiempo para realizar la evaluación, como muestra la Figura 7 con el temporizador de Fecha Tope de Entrega.

El ciclo de entregas y correcciones iterará tantas veces como tantas actividades tenga el sistema. El proyecto concluirá cuando todas las actividades hayan culminado, bien sea porque se entregaron todas las tareas o porque se terminó el plazo de tiempo establecido para ellas.

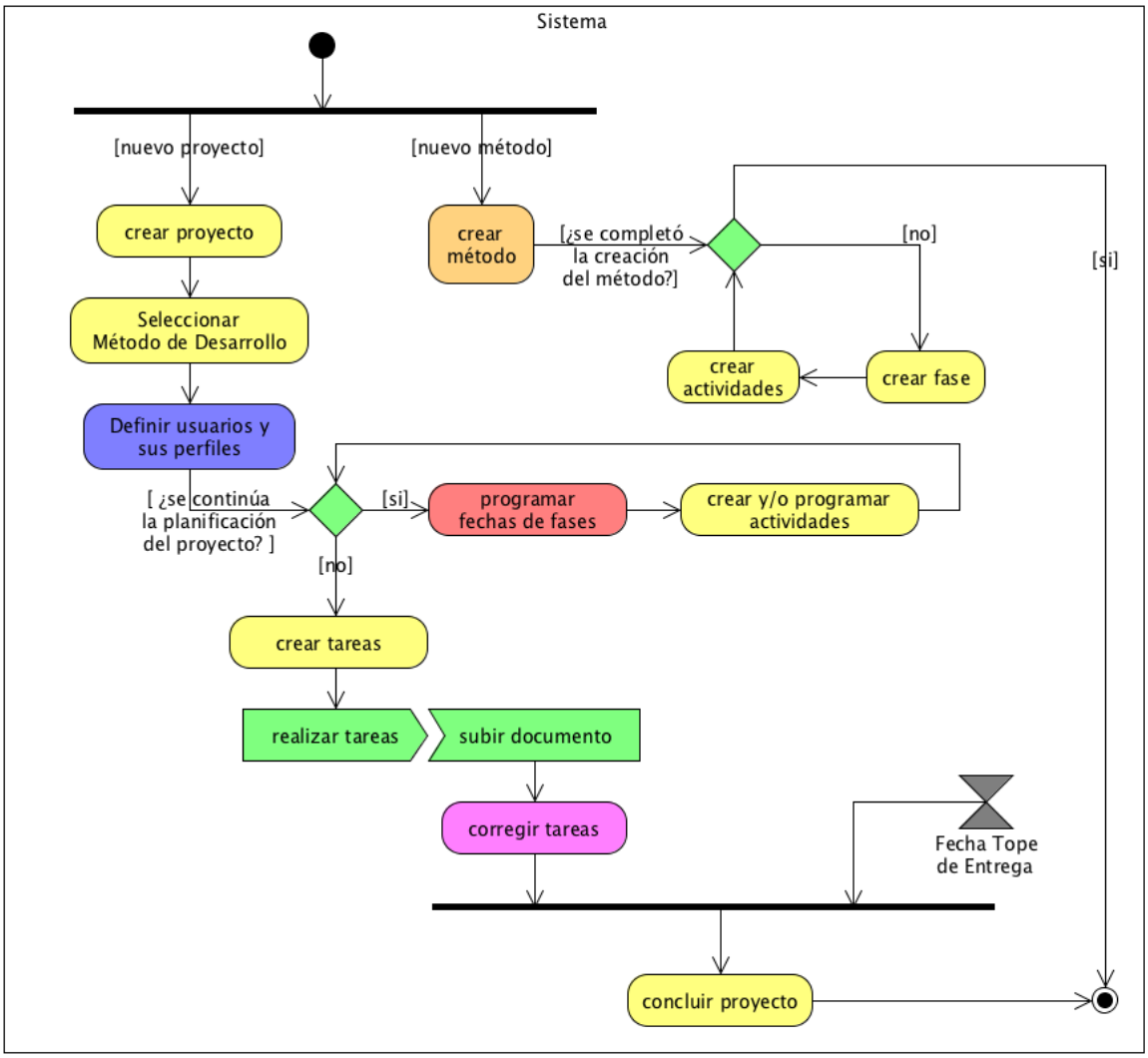


Figura 7 – Diagrama de Actividades de la aplicación

Asignación #4 - Realizar Diagrama de clases

El diagrama de clases permite definir la estructura estática de la aplicación.

Debe acotarse que la aplicación se está diseñando con el patrón Modelo-Vista-Controlador (MVC), por lo que se ha decidido realizar un diagrama exclusivamente para los modelos.

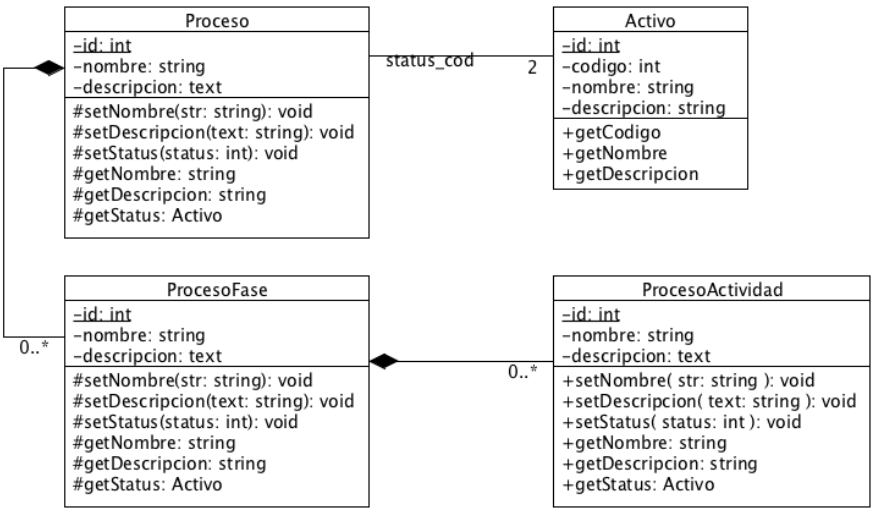


Figura 8 – Diagrama de Clases relativo al proceso de desarrollo

En la Figura 8, se presentan los modelos correspondientes al proceso de desarrollo. La clase *Proceso* representa los métodos de desarrollo existentes en el sistema. Las clases *Proceso-Fase* y *Proceso-Actividad* corresponden a las fases y actividades del método de desarrollo respectivamente. Además, un método puede estar en status *Activo* o *Inactivo*, especificada por el rol *status_cod* (de tipo entero) hacia la clase *Activo*.

Asignación #2.1 – Diseñar Componentes Básicos de la Aplicación

Una vez que se ha hecho un análisis de la aplicación que se desea a nivel estructural y funcional, debe realizarse un diseño de los elementos básicos que estarán por toda la aplicación, tales como menú de acciones, botones, formularios, listados.

Cada uno de los componentes serán analizados y se indicará cuál es la solución a tomar para cada caso.

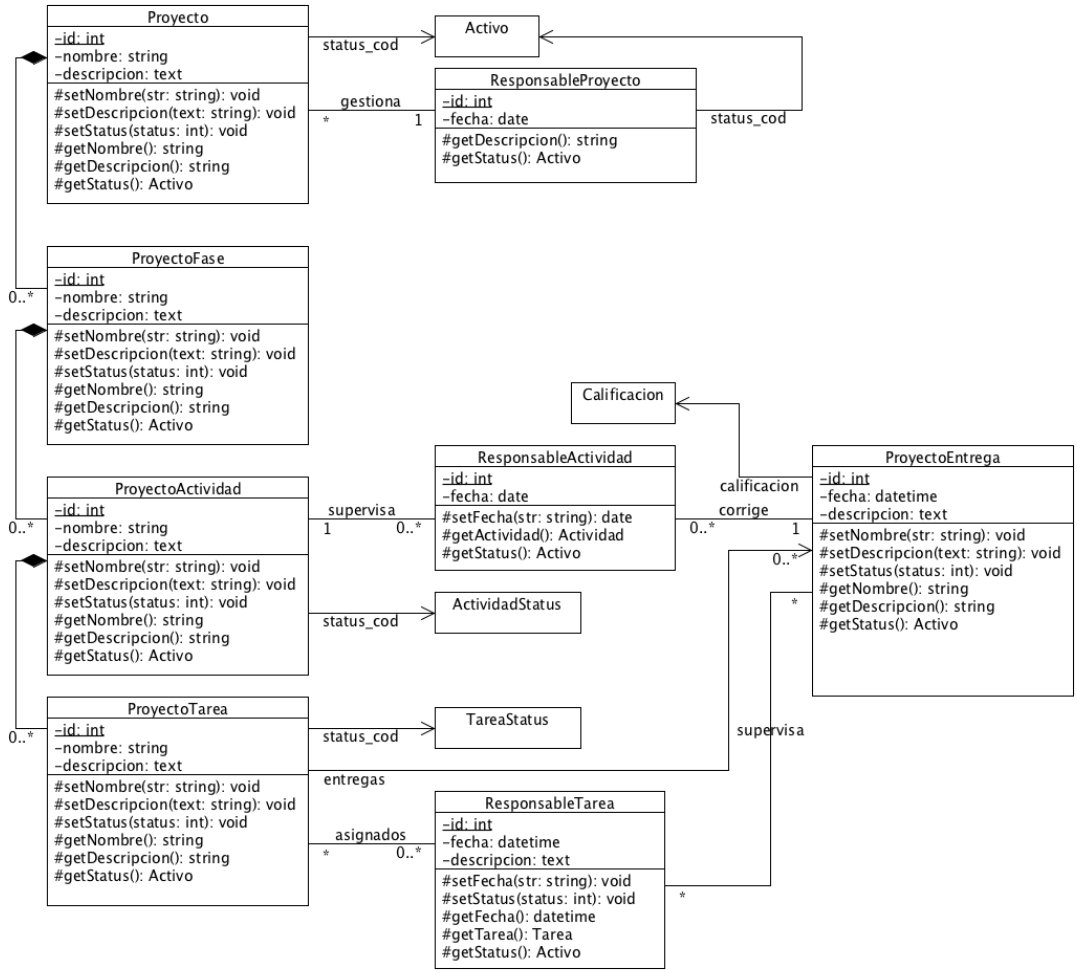


Figura 9 - Diagrama de Clases relativo al proyecto

Presentación de Tablas con operaciones CRUD

Problema: Se requiere listar objetos del sistema y las acciones posibles en ellos.

Resolución: De la biblioteca *Twitter Bootstrap* (descargable del sitio http://getbootstrap.com/) se utiliza la solución para la presentación de tablas y botones (ver Figura 10 y Figura 11).

#	First Name	Last Name
1	Mark	Otto
2	Jacob	Thornton
3	Larry	the Bird

Figura 10 – Ejemplo de una tabla realizada con la biblioteca *Bootstrap*

Además, también se utilizan los iconos propuestos para los

botones.

Figura 11 – Ejemplo de un botón realizado con la biblioteca *Bootstrap*

Se utiliza CSS3 para enriquecer el conjunto de etiquetas para tablas utilizando las propiedades de *border-radius, text-shadow* y la función *gradient* para agregarle bordes redondeados, sombra al encabezado y un color de fondo en gradiente.

El código para desplegar la tabla es el siguiente:

```
1    <table class="table table-striped listado">
2      <thead>
3        <th> # columna 1 # </th>
4        <th> # columna 2 # </th>
5        # … #
6        <th> # columna n # </th>
7        <th style="width:25%">Acciones</th>
8      </thead>
9      <tbody>
10       # contenido #
11     </tbody>
12   </table>
```

Solución: El resultado es una tabla más llamativa, manteniendo el estilo de la fuente.

Presentación de Menú de Acciones

Problema: Se requiere listar las acciones posibles de la vista.

Resolución: La etiqueta *menu* (incorporada en HTML5) ofrece varios beneficios que se listan a continuación:

- El uso de la etiqueta *menu* permite reconfiguración de los elementos de la página al desplazarse en dispositivos de diversas dimensiones (i.e. mayor portabilidad).

- El uso de etiqueta *menu* es útil para indicarle a los buscadores que se trata de un elemento de navegación,

por lo que puede ser ignorado. Esto evita el ruido en los resultados de las búsquedas.

- El uso de la etiqueta menu permite mantener un estilo estandarizado. Dicha etiqueta ofrece un estilo por defecto que puede ser fácilmente personalizado a través de las hojas de estilo.

Por las razones listados anteriormente, se utiliza para resolver el problema.

El *framework Oxwall* ofrece un *decorator* para los botones que permite estandarizar los botones de toda la red. Además, el elemento soporta internalización, de manera que puede personalizarse el idioma de forma sencilla. Este *decorator* se utiliza sin modificaciones.

Cabe destacar que el estilo del decorator es indicado por el *theme* que tenga la red configurada.

El código plantilla correspondiente al elemento es:

```
1  <menu>
2    {decorator name='button'
   onclick="location.href='# ruta #'"}
3  </menu>
```

Solución:

Presentación de Paginación

Problema: Se requiere paginar los listados de objetos del sistema.

Resolución: El *framework Oxwall* ofrece un *paging* estandarizado para toda la red. Este elemento se utiliza sin modificaciones.

Solución:

Migas de Pan

Problema: Se desea que el usuario pueda visualizar en todo momento:

dónde está y cómo llegó ahí.

Resolución: Se utiliza la ya conocida solución de las *migas de pan* o *hilos de Ariadna* de la biblioteca *Twitter Bootstrap*.

Este componente debe ser utilizado en la mayoría de vistas, por lo que se implementó un *decorator,* denominado *breadcrumbs,* que facilita su reutilización. El código es el que sigue:

```
1   <ul class="breadcrumb">
2     {foreach
3        from=$data.breadcrumbs
4        key='key'
5        item='breadcrumb'}
6      <li>
7        <a href="{$breadcrumb.url}">
8         {$breadcrumb.label}
9        </a>
10       <span class="divider">/</span>
11     </li>
12    {/foreach}
13    <li class="active">{$data.page}</li>
14  </ul>
```

Resumidamente, en el código anterior se coloca una lista desordenada con la etiqueta *ul.* El valor *breadcrumb* en el atributo *class* hace que coloque el estilo de migas de pan del *Bootstrap*. Luego, se itera en la lista de rutas y se coloca cada "miga" en una etiqueta *li.* Por ultimo, se indica la página activa o actual.

El código para desplegar un *breadcrumbs* es el siguiente:

```
1  {decorator name='breadcrumbs'
     breadcrumbs=$breadcrumbs
     page=$heading }
```

En la variable *PHP* $breadcrumbs, se encuentra la lista de las rutas correspondientes a las migas de pan. El atributo *breadcrumbs* recibe la lista para que sea creado el elemento. En el atributo *page* se especifica la página actual.

Solución:

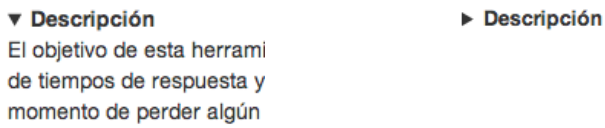
Texto expandible

Problema: Se desea que el usuario pueda visualizar u ocultar ciertos textos que deben ser mostrados, mas no necesitados en todo momento.

Resolución: Se utiliza la etiqueta *details* (incorporada en HTML5). La plantilla a seguir es:

```
1  <details>
2    <summary># Texto (e.g Descripción)</summary>
3    # contenido #
4  </details>
```

Solución:



Formularios

Problema: Se requiere solicitar los datos necesarios para los registros de objetos como proyectos, fases, actividades, entre muchos otros.

Resolución: Se utiliza la solución propuesta por Twitter Bootstrap para formularios con los botones de Cancelar y Submit del framework Oxwall.

Se listan los pasos a seguir:

1. En primer lugar, se aprovecha el estilo para la etiqueta *fieldset* de HTML en *Bootstrap* para encapsular el formulario (Línea 1). El título del formulario se coloca en la etiqueta *legend*.

2. Luego, se utiliza el elemento *form* de *Smarty* (tecnología utilizada por *Oxwall* en las vistas) para los formularios (Línea 3). El uso de dicho elemento requiere que se haya definido e incluido en el controlador un objeto *Form*.

3. Se utiliza la plantilla de *Bootstrap* para los formularios

(de la Línea 4 a la 12). Cada campo de entrada debe estar definido en el controlador de *Oxwall* con algún objeto heredado de *FormElement.* Si se especificó un *label* al objeto, debe utilizarse el elemento *label* para desplegarla (Línea 5). El campo de entrada se despliega utilizando el elemento *input* (Línea 9). Los anchos de los campos de entrada son descritos en el atributo *class* del *input* (Línea 10*).*

4. Finalmente, se colocan los botones de cancelar (en la Línea 15) y de submit (en la Línea 16).

```
1   <fieldset>
2     <legend># nombre de modelo #</legend>
3     {form name='# id del objeto Form #'}
4       <div class="control-group">
5         <label class="control-label">
6           {label name='# id del elemento #'}
7         </label>
8         <div class="controls">
9           {input name='# id del elemento #'
10              class='input-xxlarge'}
11        </div>
12      </div>
13      <div class="control-group" >
14        <div class="controls">
15         {decorator name='button'}
16         {submit name='# id del submit #' }
17        </div>
18      </div>
19    {/form}
20  </fieldset>
```

Solución:

Asignación #3 – Gestionar Proyectos

Esta asignación es el inicio de la creación de la estructura necesaria para la gestión de proyectos como sistema. A partir de este momento, se irá construyendo las clases necesarias para el sistema.

La estructuración del sistema se divide en varias asignaciones y comienza por las clases relativas a los proyectos, por considerarse la unidad más importante del sistema. En la Figura 11, se presenta el diagrama de clases correspondiente. Los modelos son clases heredadas de *Ow_Entity* y contienen los atributos, los *getters* y *setters* que se consideren necesarios para el manejo de dichos atributos. Por convención del *framework*, el nombre de los atributos debe coincidir con el nombre dado en la base de datos.

Las operaciones de consultas a la base de datos deben colocarse en la clase *DAO* respectiva. Por ejemplo, en la clase *ProyectoDao* se encuentran todas las funciones para realizar las consultas hacia la tabla *ow_ucvline_proyecto*, la cual guarda los registros de los proyectos. La clase *service* es una clase *wrapper* de todas las clases *DAO*, de modo que un controlador no haga directamente consultas a la base de datos. En el controlador, se coloca las validaciones de las entradas del usuario y la lógica de negocio. La información que debe mostrarse en la vista, es organizada y enviada a la vista respectiva.

Las acciones del *controlador Proyecto* para listar, mostrar, crear, editar y configurar proyectos tienen *vistas* asociadas.

Asignación #4 – Gestionar Modelos de Desarrollo

De forma similar a la estructura disponible para los proyectos, es creada la estructura para los procesos de desarrollo (ver diagrama de clases en la Figura 12). La clase *service* de la Figura 11 y de la Figura 12 son iguales. Sin embargo, se ha incluido en esta ultima figura únicamente las clases relacionadas con los procesos. Los proyectos no presentan una relación directa en el sistema. No obstante cuando un proyecto vaya a ser construido y sigue cierto método de desarrollo, se creará la estructura del proyecto según cómo sea la estructura del proceso de desarrollo.

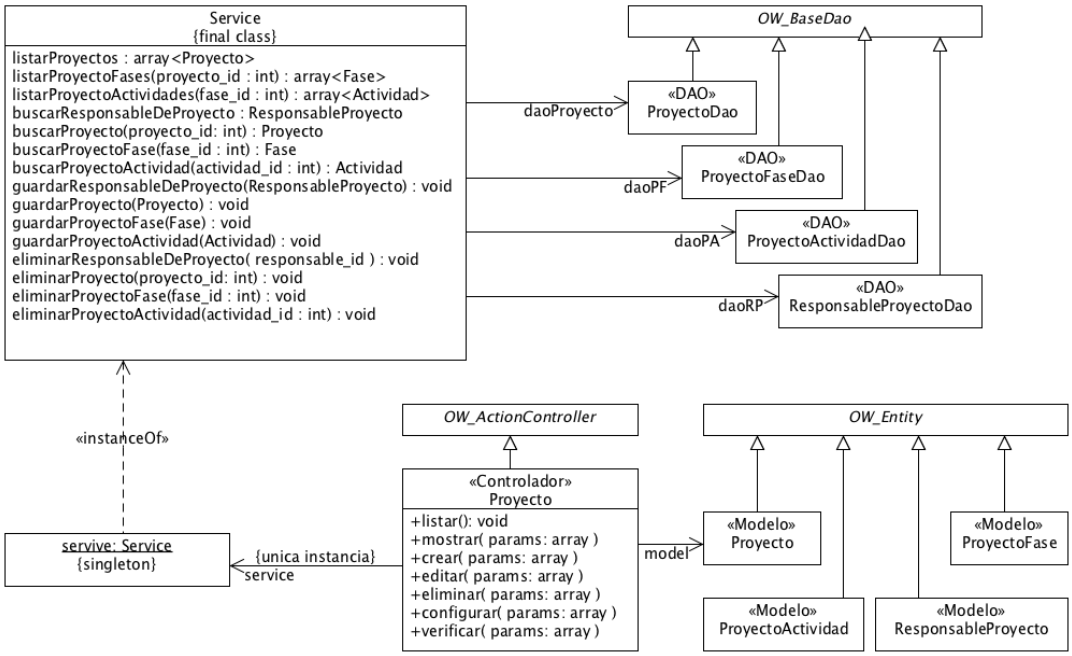


Figura 12 – Diagrama de clases relativo al proyecto

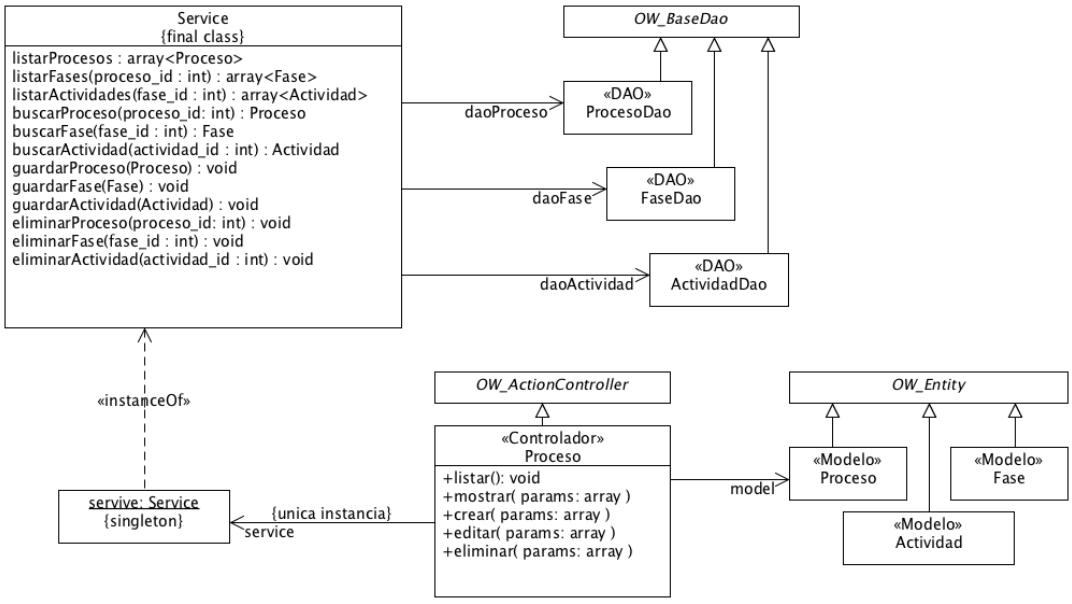


Figura 13– Diagrama de clases relativo al proceso

Asignación #5 – Gestionar Fases y Proyecto-Fases

Tal como se había mencionado con anterioridad, los procesos de desarrollo están divididos en fases. La gestión de fases se realiza separadamente, manteniéndose sólo el ID de proceso a la que pertenece.

La estructura necesaria para la gestión de fases del proceso de desarrollo se muestra en el diagrama de clases de la Figura 14.

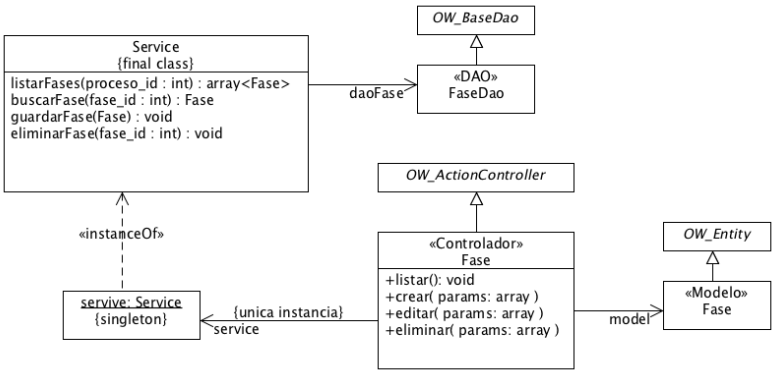


Figura 14 – Diagrama de Clases relativo a las fases

Las acciones del *controlador Fase* para listar, crear y editar fases tienen *vistas* asociadas.

La división por fases de los proyectos es muy similar a la división por fases del proceso de desarrollo, por tanto sólo se ha incluido en la misma asignación. En la Figura 14, se presenta el diagrama de clases para las fases de los proyectos. La clase *ProyectoFase* representa una fase del proyecto.

Asignación #3 – Gestionar Proyectos (Continuación)

Ya definida la estructura de la fases de los procesos de desarrollo y de los proyectos, se realizan las modificaciones pertinentes para que cada uno incluya la respectiva creación de fases. La siguiente lista indica dichas modificaciones:

1. Tras la creación de un proyecto o proceso, se pasa a la creación de fases respectiva. Podrán crearse tantas fases como sea necesario.

2. La visualización del proyecto o proceso muestra las fases que éste contenga.

3. El eliminar un proyecto o un proceso debe eliminar también las fases de dicho proyecto o proceso.

Asignación #6 – Gestionar Actividades y Proyecto-Actividades

Se prosigue con la creación de la estructura para las actividades. La clase *Actividad* gestiona las actividades de los procesos y la clase *ProyectoActividad* gestiona las actividades de los proyectos. En la Figura 13, se presenta el diagrama de clases correspondiente para ambas gestiones.

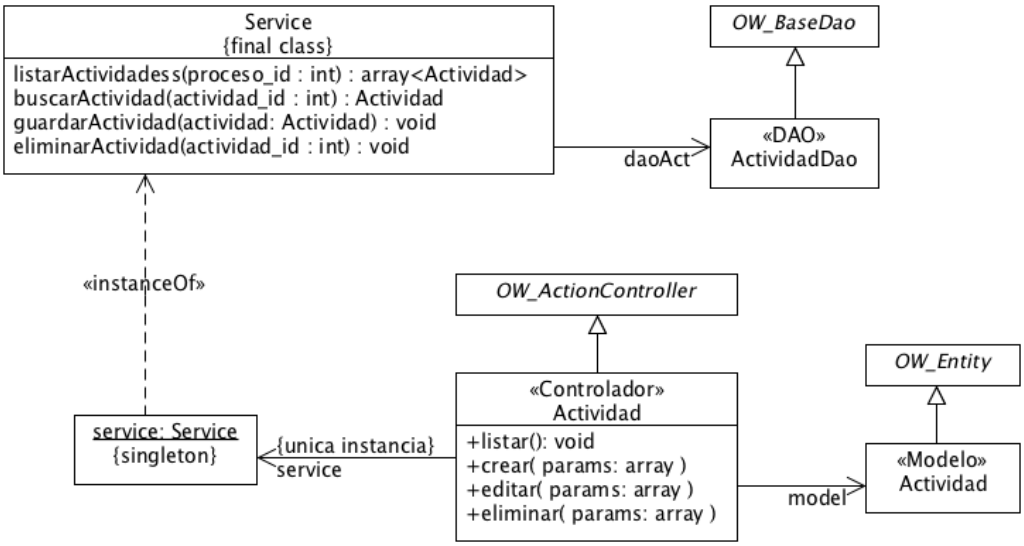


Figura 15 – Diagrama de clases relativo a la actividad

Asignación #3 – Gestionar Proyectos (Continuación)

Nuevamente son actualizados los proyectos y procesos para que funcionen con actividades. La siguiente lista indica dichas modificaciones:

1. En la creación de un proyecto o proceso mientras se crea cada fase, también podrá crearse las actividades que la fase contenga. Podrán crearse tantas actividades como la fase lo requiera.

2. La visualización del proyecto o proceso muestra las actividades que contenga cada fase.

3. El eliminar un proyecto o un proceso debe eliminar también las actividades de dicho proyecto o proceso.

Las acciones del *controlador Actividad* para mostrar, crear y editar actividades tienen *vistas* asociadas. Las acciones de listar actividades esta acoplada al mostrar proyecto.

Asignación #7 – Programar proyecto

Durante el análisis del sistema, se decidió que la programación de fechas debía realizarse en conjunto (en una misma vista) porque la programación de las fases y de las actividades dependen de las fases y de las actividades dadas previamente. No obstante, esto lleva al problema de que no todas las actividades pueden mostrarse a la vez. Por ello, se ofrece la solución de colapsar los elementos. La Figura 16 muestra un ejemplo con las fases de un proyecto.

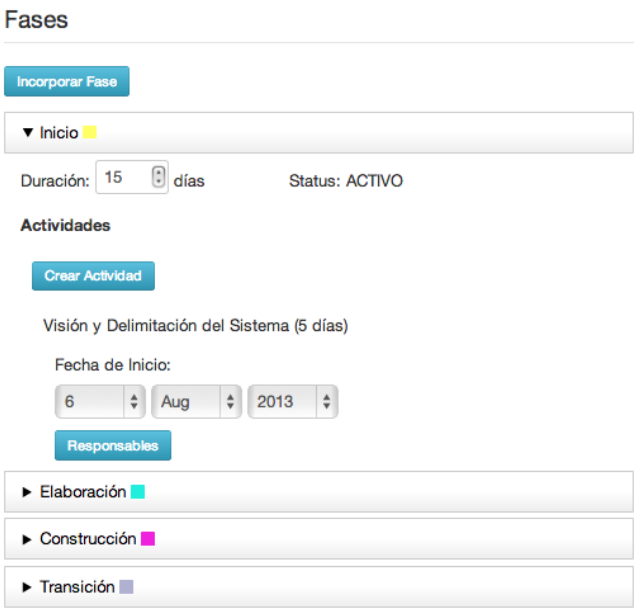


Figura 16 – Pantalla de la Gestión de Fases y Actividades

Además, se tomó la decisión de diseño de no solicitar la fecha de fin, ya que esta puede calcularse por la duración de la fase o actividad.

Cuando finaliza la programación de las fechas, se verifica los status de las actividades. La verificación permite saber si ya culminó el plazo de tiempo para entregarlas.

Asignación #2.2 – Diseñar Componente Complejos de la Aplicación

En esta asignación, se diseñaron los componentes complejos del sistema. Estos se describen a continuación:

Nombre: Calendario de Fases

Problema: Se desea que el usuario pueda visualizar en un calendario el plazo de tiempo de cada fase.

Resolución: Se utilizó la etiqueta *canvas* de *HTML5* para implementar un calendario dibujado a través del *API* de *HTML5* en *Javascript*. En un arreglo se colocan las fechas de inicio y fin de cada fase, las cuales se consideran mientras se dibuja el calendario.

Se utilizó como código plantilla el expuesto en https://github.com/ccallebs/HTML5-Canvas-Calendar. Sin embargo, la lógica para dibujar los días se rediseñó totalmente, dado que el código plantilla no resolvía la visualización de los plazos de tiempo.

Solución:

Nombre: Línea de Tiempo

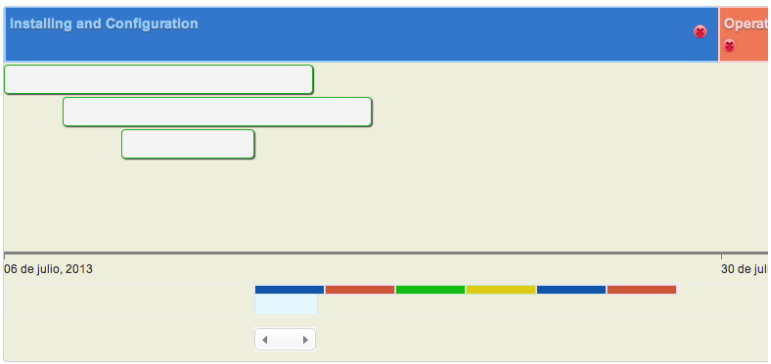
Problema: Se requiere visualizar determinado contenido dinámico en una línea temporal.

Resolución: La complejidad para resolver el problema está en que el contenido es dinámico. La planificación del proyecto se construye según el método de desarrollo e incorporación de otros fases y/o actividades. Toda la información debe ser consultada con el servidor. Además, se requiere que los elementos, cuya función es activada según los eventos del *ratón* ocurridos. Estas consideraciones hacen que la solución

más simple sea utilizar etiquetas potenciadas con *Javascript*.

Cada etiqueta representa una parte del proyecto (una fase o una actividad). En la parte superior, colocan cajas de colores que representan las fases y en la parte inferior al estilo de un diagrama de *Gantt* son colocadas las actividades.

Solución:



Asignación #7 – Programar proyecto (continuación)

Una vez que se han implementado el calendario de fases, éste puede ser incluido en la vista de programación de proyectos. La Figura 17 muestra un ejemplo de ello.

Figura 17 – Pantalla de la Planificación y Programación de Fases

Asignación #8 – Gestionar Tareas

En la visualización de cada actividad podrán gestionarse las tareas que ésta tenga. En la Figura 18, se muestra el diagrama de clases correspondiente a este requerimiento.

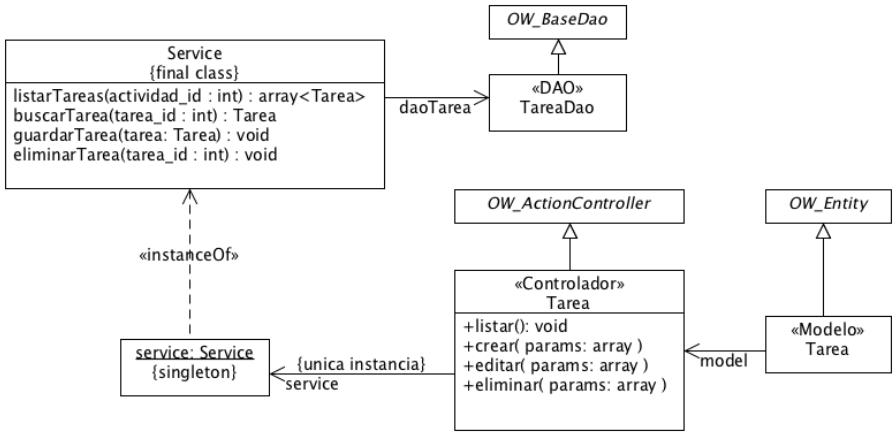


Figura 18 – Diagrama de clases relativo a la tarea

Las tareas solo podrán ser creadas por un responsable de la actividad.

Asignación #3 – Gestionar Proyectos (continuación)

Se actualiza la acción eliminar del controlador Proyecto para que elimine también las tareas del proyecto.

Asignación #9 – Gestionar Usuarios

En la Figura 19, se muestra el diagrama de clases correspondientes a los usuarios.

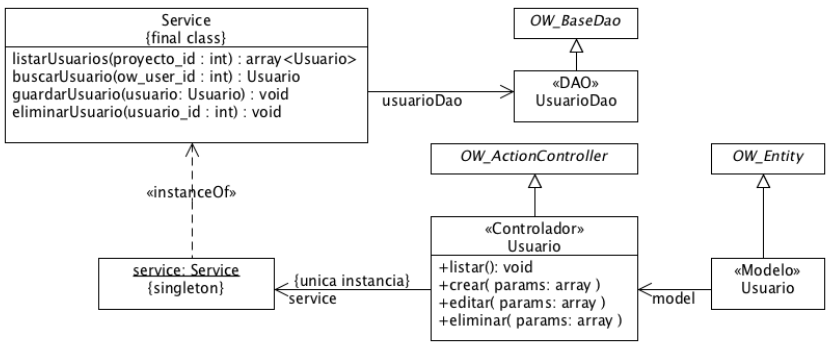


Figura 19 – Diagrama de clases relativo al usuario

Asignación #10 – Gestionar Participantes del Usuario

En la Figura 20, se muestra el diagrama de clases correspondiente a todos los participantes del aplicación.

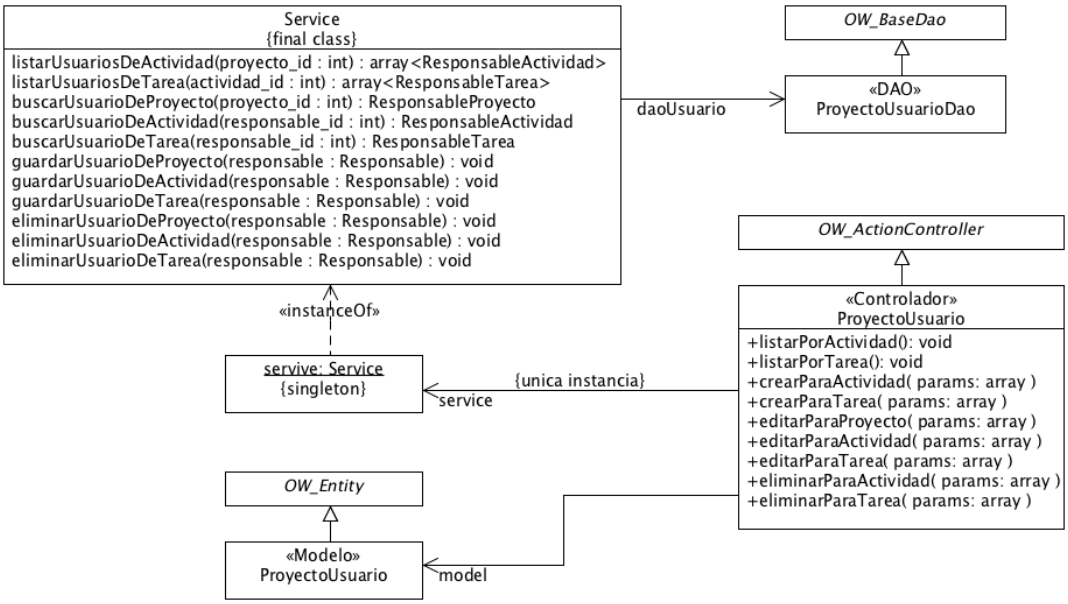


Figura 20– Diagrama de clases relativo a los participantes

Asignación #3 – Gestionar Proyectos (continuación)

Se actualiza la acción eliminar del controlador Proyecto para que elimine el o la responsable del proyecto.

Asignación #10 – Gestionar Responsables (continuación)

Se actualiza la acción mostrar Actividad para agregar la gestión y creación de responsables de tarea.

Asignación #11 – Realizar entrega de tarea

Se actualiza la acción mostrar Tarea para que puedan realizarse entregas de la tarea. La entrega de una tarea se realiza mediante la subida de un documento. La subida del documento se realiza a través de un formulario con el tipo de contenido *multipart/form-data*. Un formulario de tipo *multipart/form-data* se utiliza para enviar archivos o datos binarios. La especificación de este tipo de formulario se encuentra en el RFC2388. El documento se selecciona con un campo de entrada de tipo *file* y es almacenado en el mismo nombre de dominio que el servidor del sistema. Esto es importante indicar, ya que si el dominio fuese distinto es necesario aplicar otros mecanismos para que no sea bloqueada la solicitud para subir el archivo.

El documento es guardado en una ruta organizada jerárquicamente para el proyecto.

Asignación #10 – Gestionar Responsables (continuación)

En esta asignación, se implementan los mecanismos de autorización a las operaciones del sistema según se especificó en la descripción de actores en el diagrama de casos de usos inicialmente.

Asignación #6 – Gestionar Actividades y Proyecto-Actividades (cont.)

En esta asignación, se implementan la comprobación de las fechas de inicio y fin para determinar si ha cambiado el status de las actividades. Cada status se resalta con un color diferente para que sea fácilmente visualizado el status en la línea de tiempo. El cambio de status también influye en el comportamiento del sistema. Si una actividad ya ha finalizado no se le podrán agregar más tareas.

Asignación #13 - Realizar Pruebas de la aplicación

En esta asignación se realizan casos de prueba verificando la entrada y salida esperada de caso de uso.

Caso de Uso C01 - Gestionar Usuarios

CU-2.1 Crear Usuario

Descripción del Caso de Prueba

Debe verificarse el comportamiento ante entradas inválidas.

Condiciones de ejecución

El usuario es el administrador del sistema.

Entrada

Se ingresan los siguientes datos: {username: Felipe, ci: 12345678, nombres: Felipe, apellidos: Monagas}

Resultado Esperado

El sistema muestra una mensaje de error que indica que el username felipe no existe.

Evaluación de la Prueba

Satisfactoria. Efectivamente ese usuario no existe y se muestra el mensaje.

CU-2.2 Eliminar Usuario

Descripción del Caso de Prueba

Debe verificarse cuando un usuario con dependencias es eliminado.

Condiciones de ejecución

El usuario es el administrador del sistema.

Entrada

El usuario *alexa*, quien es responsable de un proyecto va a ser eliminada del sistema.

Resultado Esperado

El sistema muestra una mensaje de error que indica que el usuario no puede ser eliminado.

Evaluación de la Prueba

Satisfactoria.

CU-1.2.1 Planificar y Programar Fechas del Proyecto

Descripción del Caso de Prueba

Debe verificarse que no haya incongruencias en las fechas seleccionadas para la programación del proyecto.

Condiciones de ejecución

El usuario es el responsable del proyecto.

Entrada

La fecha del inicio es 6 de agosto de 2013.

La primera fase tiene una duración de 15 días, dura del 6 al 21 de agosto.

Se suministra 22 de agosto como fecha de inicio para la primera actividad de la fase (fecha inválida).

Resultado Esperado

El sistema muestra una mensaje de error que indica que debe modificarse la fecha.

Evaluación de la Prueba

Satisfactoria.

CU-1.2.1 Asignar Responsables de Tareas

Descripción del Caso de Prueba

Debe verificarse cuando es insertado un usuario que no existe para que sea responsable de una tarea.

Condiciones de ejecución

El usuario es el administrador del sistema.

Entrada

El usuario de prueba es *Felipe.*

Resultado Esperado

El sistema muestra una mensaje de error que indica que el usuario no existe.

Evaluación de la Prueba

Satisfactoria.

Los casos de usos omitidos tienen campos de entradas con validación simple o no reciben entrada del usuario, por tanto no requieren ejecutar un caso de prueba detallado.

Asignación #14 - Implementar la integración definitiva de la aplicación como plugin

La instalación de la aplicación como plug-in requiere implementar un archivo install.php en el directorio que aloja el plug-in. En él se indica lo siguiente:

- Secuencias SQL para crear la estructura y datos en la base de datos.
- Importación de los idiomas.
- Creación de grupos en el componente de autorización de Oxwall.

Debe implementarse también un archivo para desinstalar la aplicación, que consiste en eliminar las tablas correspondientes a la aplicación de la bases de datos. Por último, los archivos para activar y desactivar la aplicación, en donde

se indica que se coloque/quite el enlace para acceder a la aplicación desde el menú.

Con la asignación #14, se concluye el desarrollo de la aplicación de gestión de proyectos para el Centro de Ingeniería de Software y Sistemas.

Escenario de Usos

La aplicación inicia con el listado de proyectos (véase Figura 21). En esta primera pantalla pueden realizarse las operaciones de Crear Proyecto, Importar Usuario, Consultar alguno de los proyectos del listado. Para este escenario, se va a asumir que los usuarios necesarios la gestión del proyecto ya han sido importados previamente.

Figura 21 – Pantalla de Gestión de Proyectos

Al accederse a la operación de *Crear Proyecto* se muestra la pantalla de la Figura 22. En ella deben ingresarse los datos generales del proyecto: nombre, descripción, fecha en la que inicia el proyecto, y opcionalmente el método de desarrollo que se utilizará para el mismo.

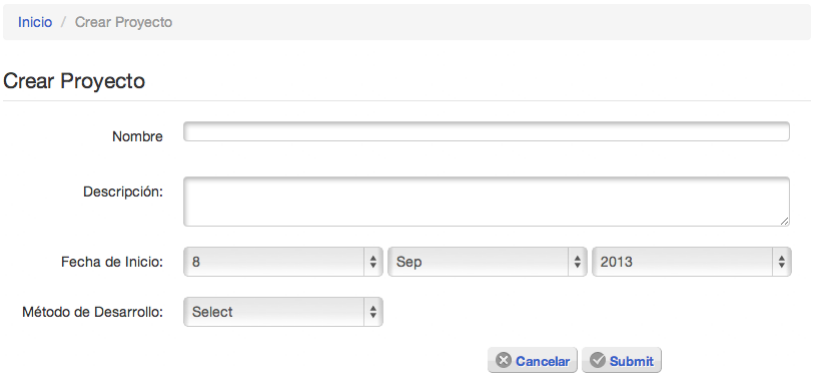


Figura 22 – Pantalla de *Crear Proyecto*

Al enviar los datos generales del proyecto, se solicita agregar de uno en uno cada participante del proyecto (ver Figura 23), siempre indicando el rol que tiene cada uno de ellos.

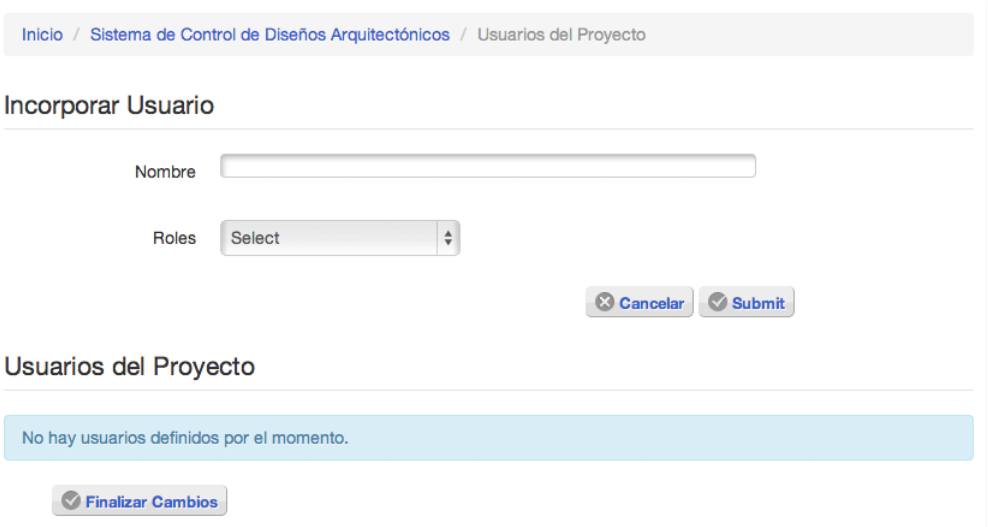


Figura 23 – Pantalla de *Incorporar Usuario*

Cuando se haya agregado el ultimo participante, se presiona Finalizar Cambios y se prosigue con la programación de fechas del proyecto. Si se hubiera seleccionado un método de desarrollo, se hubieran autogenerado las fases y actividades que soportan dicho método.

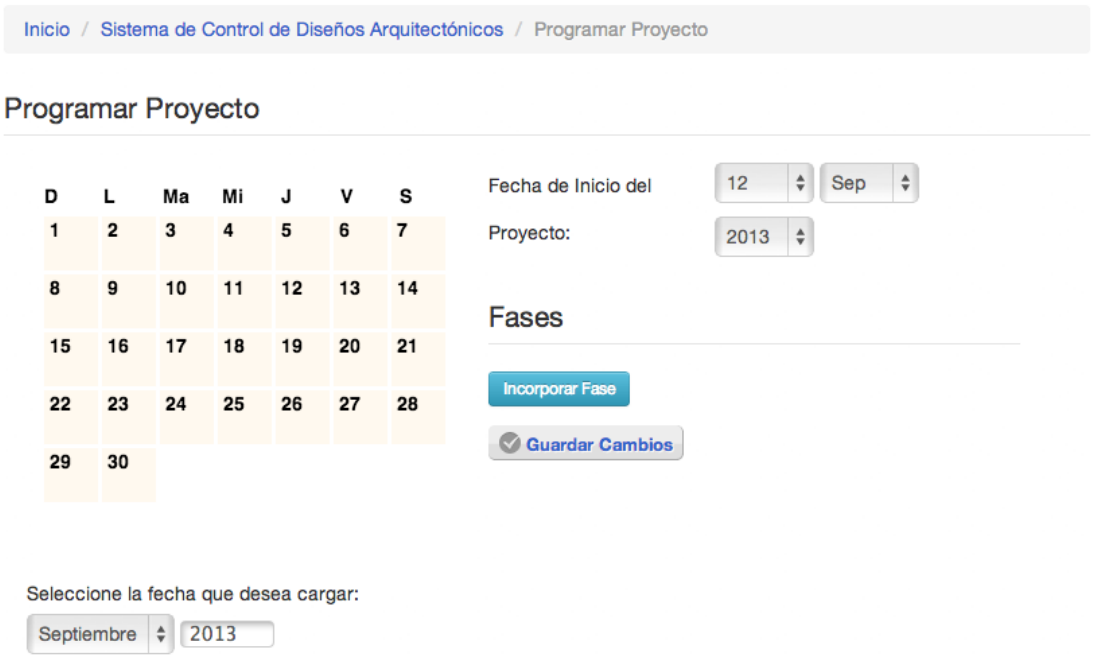


Figura 24 – Pantalla de *Programar Proyecto*

Para incorporar una fase al proyecto, se presiona el botón *Incorporar Fase*.

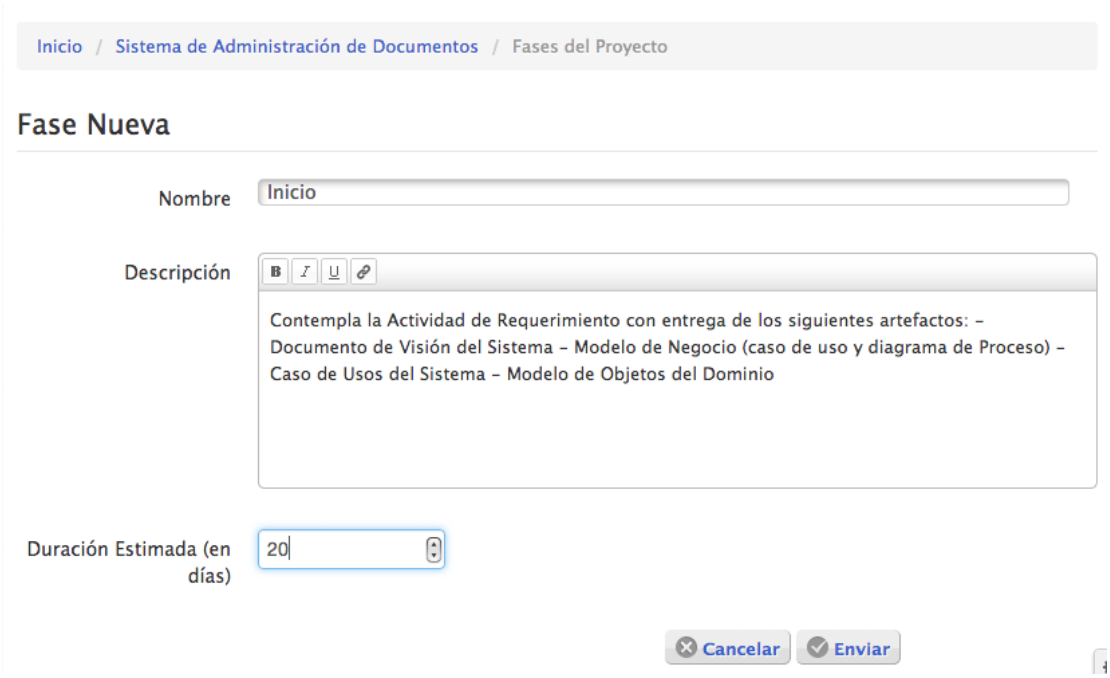


Figura 25 – Pantalla de *Crear Fase*

En Incorporar Fase, se muestra una pantalla como muestra la Figura 25. Si el proyecto tiene fases, las mismas aparecerán en una tabla (ver Figura 26).

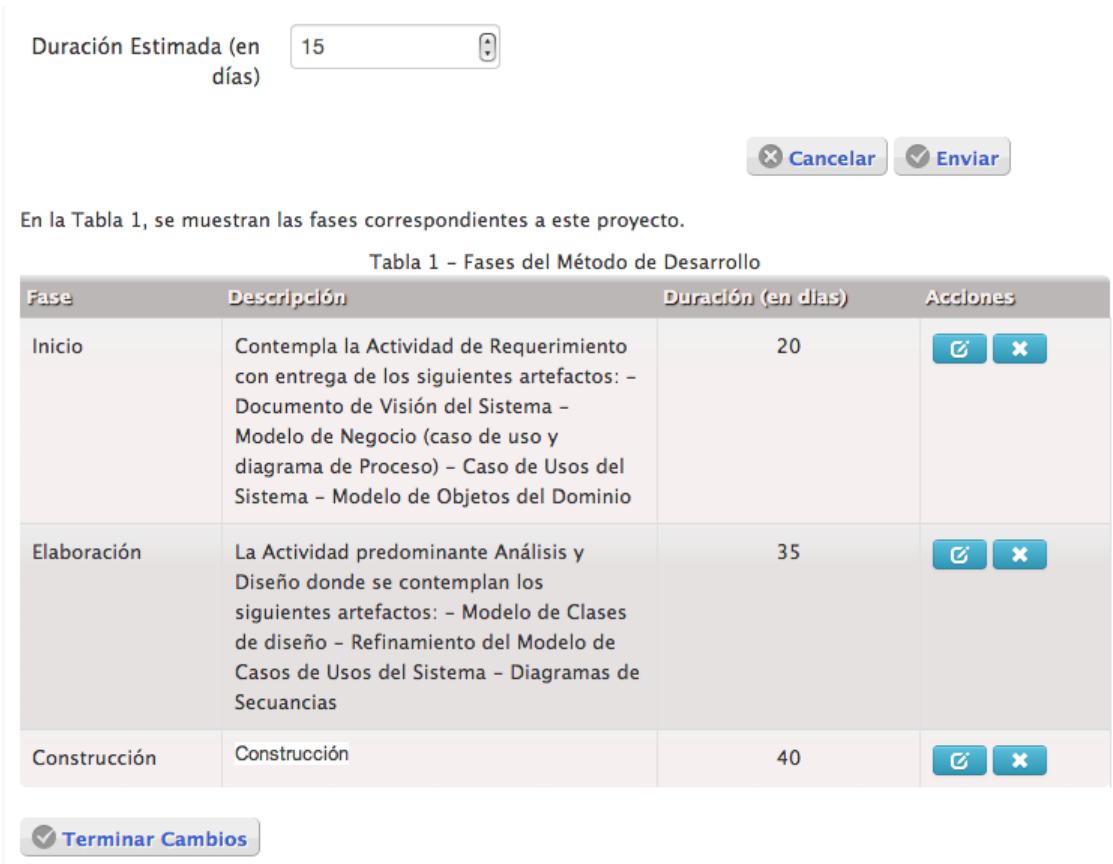


Figura 26 – Extracto de Pantalla de *Gestionar Fases*

Los botones a la derecha permiten editar/eliminar la fase, si fuere necesario. Podrán agregarse tantas fases como sea necesario. Para ello, deben ingresarse los datos indicados y presionar Submit. La fecha de inicio y de fin de la fase será calculada acorde con la fecha de inicio del proyecto.

Una vez que han sido incorporado todas las fases, se presiona Finalizar Cambios. En la pantalla de Programación de Proyecto se muestran las fases colapsadas. Bien puede expandirse alguna y Crear Actividad (ver Figura 27).

Figura 27 – Pantalla de Programar Fechas del Proyecto

La pantalla para Crear Actividad se muestra en la Figura 28. Se insertan los datos generales de la actividad: nombre, duración y descripción.

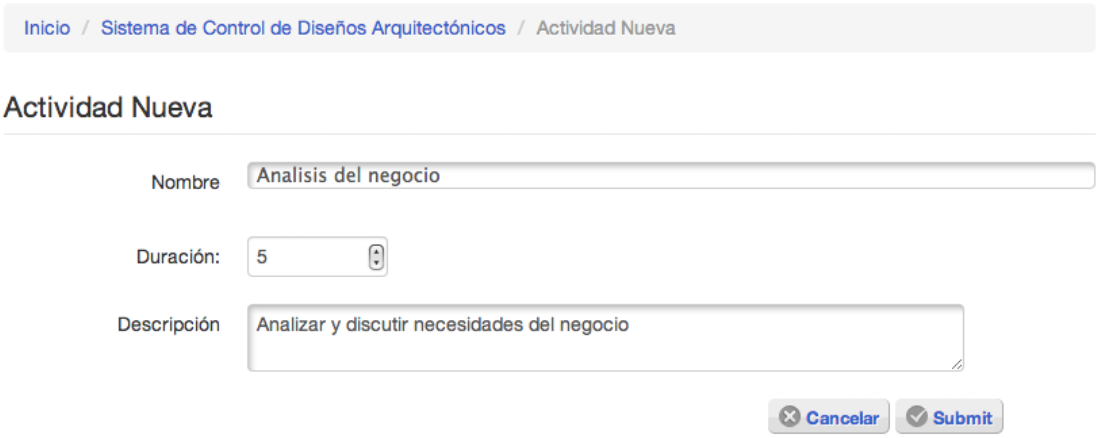


Figura 28 – Pantalla de *Crear Actividad*

Después de planificarse y programarse las estructuras y fechas del proyectos, se guardan los cambios y se pasa a la pantalla de visualización de la línea de tiempo del proyecto. La Figura 29 muestra dicha pantalla en la que se

encuentra el nombre del proyecto, un menú de operaciones sobre el proyecto, la descripción del proyecto, y la línea del tiempo.

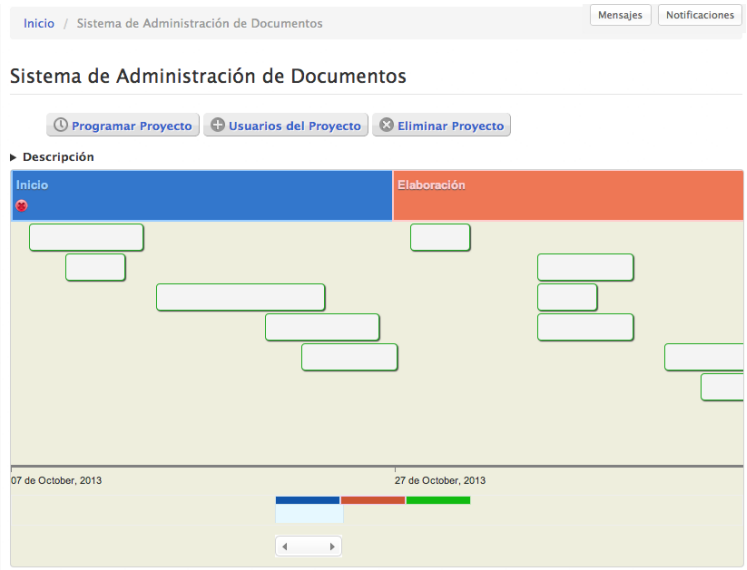


Figura 29 – Pantalla de *Consultar Proyecto*

El componente con color de fondo beige es la línea de tiempo. Esta dispone en la parte superior cajas coloreadas que representan las fases del proyecto (en la Figura 29, la caja azul correspondiente a la fase Elaboración). En el espacio bajo la fase Elaboración, se colocan línea a línea las actividades de dicha fase. Cada actividad está representada por un rectángulo gris con un *tooltip* que indica su nombre.

En la parte inferior de la línea de tiempo, se encuentra la barra de desplazamiento. En la mayoría de los casos, la línea de tiempo abarcará mucho más de una pantalla. La barra de desplazamiento pertenece realizar un desplazamiento horizontal sobre la misma.

Al presionarse cualquiera de la actividades del proyecto, se irá a la pantalla de consulta de la actividad, tal como muestra la Figura 30.

Figura 30 – Pantalla de *Consultar Actividad*

La pantalla de Consultar Actividad indica el status de la actividad, para cuál fecha fue programada su entrega, y una descripción de la actividad. Luego muestra sus responsables y las tareas asignadas para la actividad.

Desde la pantalla de Consultar Actividad pueden gestionarse las tareas de la misma. Al presionar el titulo de una de las tareas se redirecciona a Consultar Tarea (ver Figura 31).

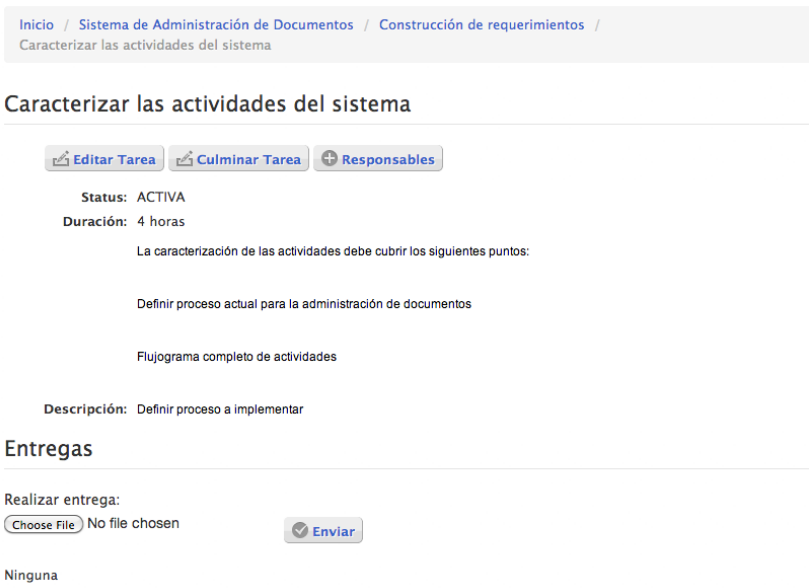


Figura 31 – Pantalla de Consultar Tarea

La pantalla de *Consultar Tarea* permite subir el documento referente a la realización de la tarea para que sea evaluado. En la Figura 32, se ha subido un documento de prueba para mostrar la gestión de entregas. Cada vez que se suba una nueva entrega, se colocará debajo de la anterior. No podrán sobre-escribirse las tareas, a manera de mantener un histórico de los documentos subidos.

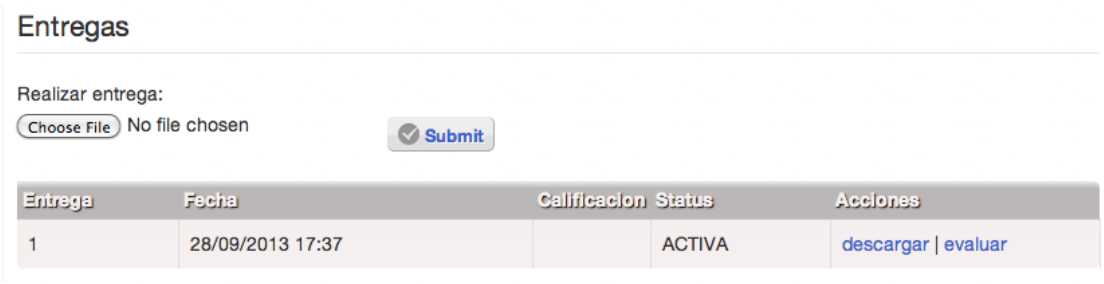


Figura 32 – Pantalla de Listado de Entregas

Cualquier usuario con privilegios podrá descargar y calificar cualitativamente la entrega. Seleccionando la calificación que considere, como muestra la Figura 33.

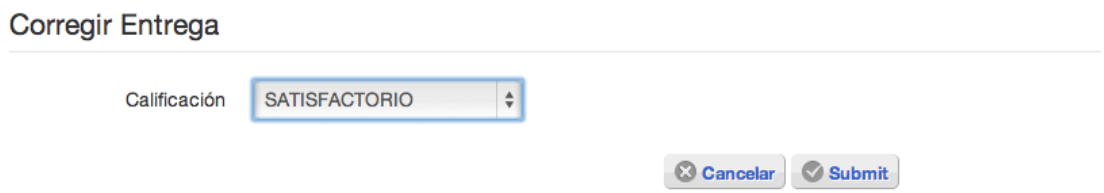


Figura 33 – Pantalla de Corregir Tarea

Cuando la tarea se considere culminada, podrá presionarse *Finalizar Tarea* en la pantalla de *Consultar Tarea*. De igual forma, se podrá finalizar una actividad, presionando *Finalizar Actividad*. Una actividad finalizada puede ser reactivada en la pantalla de *Editar Actividad*.

Estas fueron las pantallas principales del sistema, las cuales permiten manejar un proyecto.

Sugerencias

La documentación del *Oxwall* no dispone del *API* para implementar las aplicaciones ni es explicado cómo se organiza el *framework*. Esto dificulta bastante la implementación en el mismo. Tampoco dispone de generadores de

vistas, característica ofrecida por múltiples *framework* que disminuye ampliamente los tiempos de desarrollo.

Como sugerencia se propone la implementación de un módulo para autogenerar componentes del *framework* (modelos, archivos DAO, controladores, etc.) junto a un manual web de usuario que explique un subconjunto de las funciones del *Oxwall*.

CONCLUSIONES

En este documento se realizó un estudio sobre cómo implementar una aplicación de la Web 2.0 en la plataforma social *Oxwall*, a fin de extender la funcionalidad de dicha plataforma con la gestión de proyectos colaborativos.

Dicho propósito se cumplió utilizando la estructura requerida para realizar extensiones en Oxwall, la cual está bajo el patrón de diseño *Modelo-Vista-Controlador*.

Se utilizó HTML5 para implementar algunas funcionalidades como el calendario en la vista de Planificación y Programación del Proyecto, dando una solución atractiva y más ligera con la etiqueta *canvas* que utilizando las etiquetas tradicionales de HTML. Además de ser una alternativa gratuita que permite dar soluciones que, anteriormente, solo ofrecía *Flash*.

Por otra parte, algunas características del HTML5 aún no disponen de soporte en importantes navegadores como *Firefox, Opera, IE*. La internalización parece no estar siendo contemplada en *HTML5*, lo que representó una limitación con respecto al uso de soluciones tradicionales en bibliotecas en *Javascript*. Razón por la que se considera la necesidad de realizarse más trabajos en *HTML5* que lo enriquezcan. Debe recordarse que los *framework* tradicionales en *Javascript* tienen bastante tiempo en el mercado y, por ello, ofrecen soluciones bastante trabajadas y probadas.

La implementación de una línea de tiempo para visualizar la planificación y programación del proyecto, dió una solución sencilla y manejable hacia la gestión de proyectos con respecto a las grandes aplicaciones del mercado que resultan ser demasiado complicadas y no son recomendables para principiantes.

Otra tecnología utilizada y de importancia es el *CSS3*. Recientemente, han sido incorporados nuevos selectores que permiten generar interfaces más elegantes. Una selección de etiquetas de forma más específica es posible con las pseudo clases y pseudo elementos ofrecidos en esas actualizaciones de los selectores. Por ejemplo, puede seleccionarse una etiqueta que sea la primera hijo de un div, todos los hijos excepto el primero y el último, entre otros casos de selección. Esto permite diseñar interfaces de mejor calidad con menor esfuerzo y más legibles.

REFERENCIAS

W3C (2013). The creation of World Wide Web, W3C. Recuperado en marzo del 2013.

OSI (2013). The Open Source Definition, Open Source Initiative. Recuperado en marzo del 2013.

O'Reilly (2005). What is the web 2.0. O'Reilly. Recuperado en http://oreilly.com/web2/archive/what-is-web-20.html, marzo del 2013.

Colombo (2004). The Dot-com Bubble, Stock Market Crash!, Recuperado en marzo del 2013.

Highsmith (2001). History: The Agile Manifesto. Agile Alliance. Recuperado en http://www.agilemanifesto.org/history.html, marzo del 2013.

Wilson (2008). Common Public Attribution License - An Overview. OSS Watch. Recuperado en http://www.oss-watch.ac.uk/resources/cpal, marzo del 2013.

Wilson (2005). Mozilla Public License - An Overview. OSS Watch. Recuperado en http://www.oss-watch.ac.uk/resources/mpl, marzo del 2013.

Wikipedia (2013). Software Release life Cicle – Alpha. Recuperado en http://en.wikipedia.org/wiki/Software_release_life_cycle#Alpha, mayo del 2013.

Miguel, V; Montaño, N.; Fernández, M.; Rondón, J.; Lucci, V.; Díaz, K.; Flores, J (2012) - Modelo de enseñanza colaborativa basado en la Web 2.0 para el fortalecimiento de la enseñanza en ciencia y tecnología - Doctorado en Educación, Universidad Católica Andrés Bello.

Maspons (2012) – Seminario, Escuela de Computación, UCV

Anexos

Framework de Desarrollo Oxwall

Actualmente, el centro utiliza una red social para promover el aprendizaje dirigido a través de la Web. Dicha red social contiene varias aplicaciones colaborativas integradas. Sin embargo, se presenta la necesidad de una aplicación que permita el seguimiento de proyectos colaborativos en la Web.

Oxwall es un *framework* de desarrollo y una red social desarrolla en *PHP/MySQL.* Permite el desarrollo de *plug-ins* instalables mediante la interfaz administrativa de la red social. También permite la implementación de componentes reutilizables para diferentes vistas o *plug-ins.* Ambos elementos (*plug-ins y componentes*) siguen la estructura del patrón de diseño *Modelo-Vista-Controlador* (*MVC*). Admite internacionalización por medio de la gestión de idiomas en la sección de administración. Las vistas utilizan la tecnología *Smarty* que facilita la inclusión de iteradores y componentes decorativos.

En estos momentos, *Oxwall* tiene una documentación muy limitada. Sin embargo, su jerarquía de directorios, archivos *PHP* y los *plug-ins* predeterminados permiten el estudio de cómo utilizar el *framework*.

La motivación de enriquecer dicha red social también se sustenta sobre el principio *DRY* (*Don't Repeat Yourself*), cuyo interés es reducir código repetido y reutilizar código propio y de terceros. Este principio permite reducir los plazos de tiempo para implementar software con nuevas funcionalidades.

El uso adecuado de los pertinentes patrones de diseño y técnicas de programación lograran que la nueva aplicación cubra los requerimientos no funcionales imprescindibles como:

1. Código legible
2. Código reutilizable para soluciones similares
3. Código mantenible

SCRUM

Este método es iterativo y de desarrollo ágil para la gestión y desarrollo software.

Scrum define varios roles principales encargados de desarrollar y supervisar los avances del proyecto que puedan ser adaptados al desarrollo de este software.

El rol *ScrumMaster*, el director del proyecto, vendría siendo el rol del tutor. El rol *ProductOwner* representa a los *stakeholders.* Este rol puede ser cubierto por aquellas personas relacionadas al sistema que estén presentes en las reuniones. Los *stakeholders* forman parte de los procesos y, por lo tanto, pueden aportar información útil para la captura de requerimientos. El rol *DevelopmentTeam* es el desarrollador de la aplicación.

Varios artefactos son definidos para llevar el método Scrum.

Product Backlog

El documento *Product Backlog* es un artefacto del método Scrum. Éste lista los requisitos y descripciones genéricas a alto nivel para definir el trabajo a realizar durante todo el desarrollo. Cada elemento del *Product Backlog* se organiza por prioridades según un valor denominado *Retorno de Inversión,* medida que calcula y compara la eficacia de las inversiones realizadas.

Sprint Backlog

El documento *Sprint Backlog* trata de detallar cómo el *Team* va a implementar los requisitos en cada iteración. El *Sprint Backlog* contiene tareas que duran a lo sumo 16 horas, de donde los miembros del equipo de desarrollo seleccionan cuales tareas van a implementar (las tareas no son asignadas).

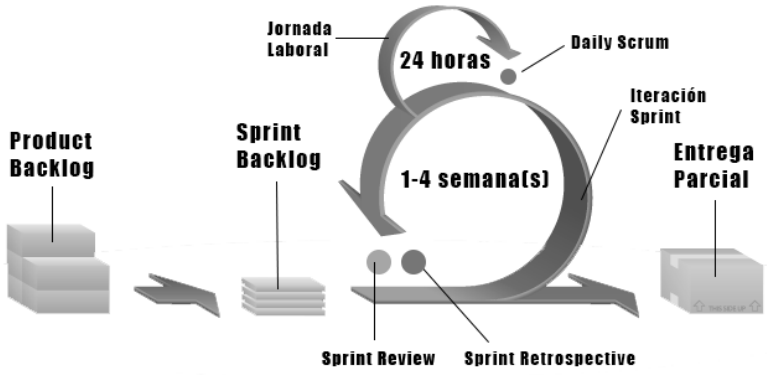


Figura 34 – Ciclo SCRUM

Como muestra la Figura 34, el flujo de trabajo comienza con la definición del *Product Backlog*, a lo que sigue el ciclo de iteraciones. Cada iteración es llamada *sprint* y consiste en el plazo de tiempo para realizar una entrega parcial. La duración de cada iteración va de entre una a cuatro semanas. El principio de cada *sprint* es marcado por una reunión de planificación donde se define el conjunto de elementos del *Product Backlog* (i.e. *Sprint Backlog*) que contendrá ese *sprint*. Estas reuniones permiten que el desarrollo sea dinámico y pueda haber flexibilidad en los requerimientos del software. Diariamente son supervisados los avances en reuniones diarias, llamadas *Daily Scrum*. En ellas se comenta sobre los avances y quehaceres del día a día del desarrollo.

En la reunión de planificación debe concentrarse un acuerdo entre el *ProductOwner* y el *DevelopmentTeam*. El *ProductOwner* define que aquellos elementos quiere que se implementen, siempre utilizando el *Product Backlog*. Mientras que el *DevelopmentTeam* indica qué es lo que puede implementar para esa iteración. Además, se intenta que la reunión se de en un espacio abierto que permita la participación de otros *stakeholders* además de los representantes del cliente como secretarias, auditores, etc.

Una reunión de mayor alcance es la reunión *Sprint Review* realizada al finalizar el *sprint* para verificarse qué logros se completaron y cuáles no. Además, hay una reunión de retrospectiva, llamada *Sprint Retrospective*, en la que se reflexiona sobre el *sprint* recién superado.

Adicionalmente, hay un artefacto llamado *Burn Down* en donde se muestra una gráfica que mide la cantidad de elementos pendientes del *Product Backlog* para el *sprint*.