Universidad de las Ciencias Informáticas

Facultad 2:"Telecomunicaciones y Seguridad Informática"

Título: Interfaz visual para servidor de control de versiones Subversion.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

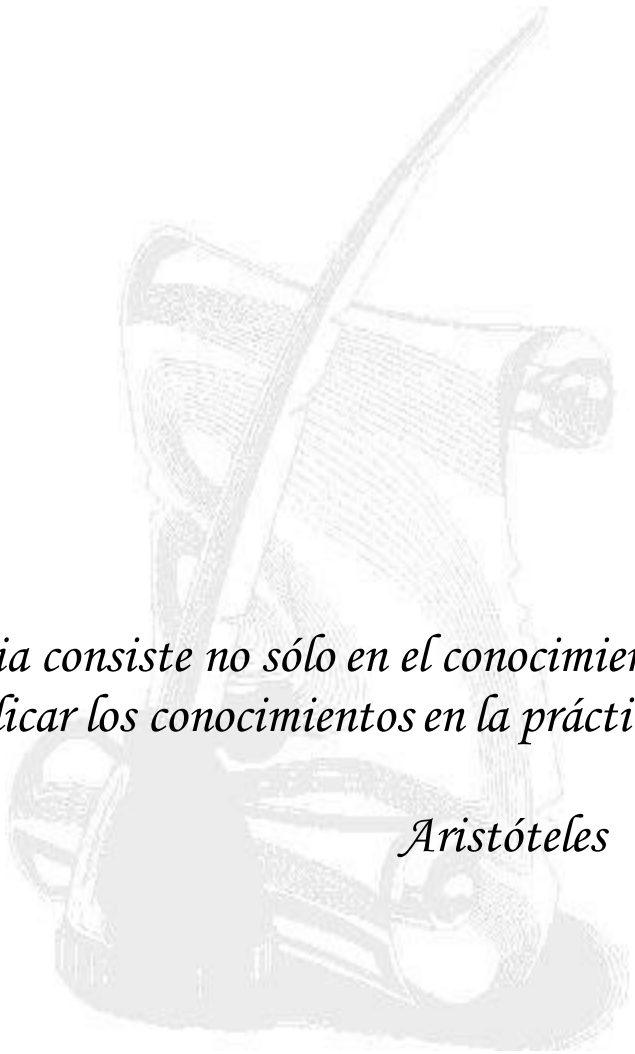
Autor(es): Roberto Linares Zamora.

Rafael Antonio Riquene Llorente.

Tutor(es): Rubén Enrique Goderich Ruiz.

La Habana

Junio 2013



"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica."

Aristóteles

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.
Para que así conste firmo la presente a los ____ días del mes de _____ del año 2013.

_____ _____

Firma del Autor
Roberto Linares Zamora

Firma del Autor
Rafael Antonio Riquene Llorente

_____

Firma del Tutor
Ing. Rubén Enrique Goderich Ruiz.

DATOS DE CONTACTO

Roberto Linares Zamora
Correo: rzamora@estudiantes.uci.cu
Ciudad de La Habana, Cuba

Rafael Antonio Riquene Llorente
Correo: rariquene@estudiantes.uci.cu
Ciudad de La Habana, Cuba

Ing. Rubén Enrique Goderich Ruiz
Correo: regoderich@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

AGRADECIMIENTOS

En los agradecimientos casi siempre se cometen injusticias, pues la memoria es a menudo traicionera. Pero aun sabiendo que no existirá una forma para agradecerles, hoy sin embargo, podemos afirmar sin lugar a dudas, que este trabajo no hubiese sido posible sin la ayuda de:

Nuestros padres que es lo más grande que tenemos y que tanto nos han apoyado para lograr nuestros objetivos. A ellos les debemos la vida y esperamos retribuirles lo que nos han dado.

Nuestro tutor ingeniero Rubén Enrique Goderich, ingeniera Barbara Triana Morales y al ingeniero Álvaro Luis Padilla Moya que han estado presente y dispuestos a ayudarnos cuando lo hemos necesitado.

Nuestra familia que nos ha apoyado en las buenas y las malas, nuestros más grande agradecimiento.

Nuestros compañeros, que aunque a muchos no los veremos más, no los olvidaremos.

La Universidad de las Ciencias Informáticas que nos dio la oportunidad de estudiar esta carrera para formarnos como ingeniero en tan importante ciencia, lo que nos permitirá contribuir al desarrollo tecnológico de nuestra gran nación.

DEDICATORIA

Roberto Linares Zamora.

Tengo en la vida dos personas importantes, los que siempre me han apoyado, sin importar lo que piensan. Gracias a ese apoyo he ganado confianza en lo que hago cada día y me he vuelto más independiente. Mis padres Andrea y Roberto que son los más grandes merecedores de todo mi agradecimiento y orgullo. A ellos va dedicado este trabajo.

Rafael Antonio Riquene Llorente.

Existen muchas personas importantes en mi vida las cuales siempre me ha apoyado en los momentos más difíciles, a todas ellas, gracias por ese apoyo que me ha guiado a ganar confianza en lo que hago cada día, en especial a mis padres Rafael y Loira, que son los más grandes merecedores de todo mi agradecimiento y orgullo. Este trabajo se lo dedicado a todas esas personas que ocupan un lugar en mi corazón, les juro que no los defraudaré. Gracias por todo.

RESUMEN

La mayoría de las entidades que desarrollan software necesitan un sistema para gestionar el control de versiones en los mismos. En Cuba existen varias instituciones dedicadas al desarrollo de software entre ellas se encuentra la Universidad de las Ciencias Informáticas (UCI), la misma está dividida en estructuras productivas especializadas en diversos temas informáticos, el centro de Telemática (TLM) forma parte de estas estructuras y en sus proyectos para gestionar el control de versiones utilizan la herramienta Subversion(SVN). La administración de svn es un poco engorrosa y puede tornarse lenta puesto que el administrador debe tener conocimiento de un gran número de componentes, subcomandos, la dirección de los archivos del repositorio, es a nivel de consola y para realizar una operación sobre los mismos debe hacerlo desde el servidor en que se encuentra la herramienta.

El objetivo fundamental de esta tesis es darle solución al problema de agilizar el proceso de administración del sistema SVN y aportar una solución para el mismo.

Así pues el aporte principal de este trabajo consiste en el diseño e implementación de una herramienta web que permita administrar los repositorios desde cualquier parte, de forma rápida y efectiva. En la presente investigación se propone la fundamentación acerca de todas las herramientas utilizadas, además de todos los elementos que se necesitan para el desarrollo de la herramienta web en el proceso de administración de SVN, ya que con la realización de la misma se pueden cubrir las necesidades expuestas anteriormente para las diferentes entidades del país que utilicen SVN como herramienta control de versiones y en especial las del centro TLM.

Palabras claves:

Administración, repositorio, Subversion

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las actividades relacionadas con el desarrollo de software son generalmente muy dinámicas, y los productos generados altamente susceptibles al cambio, por lo que se necesita llevar la gestión adecuada de estos productos para evitar el descontrol. Durante el proceso de desarrollo de software se producen elementos (códigos fuente, ejecutables, documentos y datos) que inevitablemente cambiarán. Los cambios se pueden presentar por distintos motivos ya sea por; requerimientos en las funcionalidades deseadas o cambios externos al proyecto; precisamente es que surge en este entorno el control de versiones. (1)

Se define como **control de versiones** a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra dicho producto en un momento dado de su desarrollo o modificación. Desde su surgimiento hasta la actualidad los sistemas de control de versiones (SVC, por sus siglas en inglés) han evolucionado lentamente; eran casi una labor manual. Ya existen múltiples sistemas de control de versiones y algunos de los más utilizados son: Sistema Concurrente de Versiones (CVS, por sus siglas en inglés), Subversion (SVN, por sus siglas en inglés), SourceSafe, ClearCase, Darcs, Bazaar, Plastic, Git, Mercurial, entre otros. (2)

La Universidad de las Ciencias Informáticas, se ha convertido en un centro vanguardia de la rama informática en nuestro país siendo una de las mayores productoras de software. Entre sus objetivos figura el de fortalecer la economía del país y para lograrlo, se ha dividido en estructuras productivas o centros especializados en diversos temas. El centro de Telemática utiliza como sistema de control de versiones Subversion para gestionar los cambios y versiones que realizan sus desarrolladores. Para la administración de dicho sistema de control de versiones, es necesario dominar un gran número de componentes y subcomandos, así como las direcciones de los archivos lo que puede traer consigo confusión y lentitud a la hora de su administración. Este proceso puede llegar a tornarse incómodo y engorroso ya que se realiza a nivel de consola donde el administrador siempre tiene que estar en la pc servidor para la administración del mismo.

Por lo antes expuesto se plantea como **Problema científico**: ¿Cómo agilizar el proceso de administración del sistema de control de versiones Subversion?

Objeto de estudio de la presente investigación: Sistemas de control de versiones.

Campo de acción Sistema de control de versiones Subversion.

Para dar solución al problema expuesto anteriormente se traza como **objetivo general**: Desarrollar un sistema que permita la administración de un repositorio Subversion mediante una interfaz web.

De dicho objetivo general se derivan los siguientes **objetivos específicos**:

- Realizar el diseño de la aplicación web de forma amigable y de fácil utilización.
- Seleccionar e implementar los comandos con los que ejecuta las acciones el servidor de control de versiones Subversion.
- Validar el sistema desarrollado con la ejecución de las pruebas de calidad definidas por la metodología de desarrollo a utilizar para comprobar el correcto funcionamiento del mismo.

Tareas Investigativas:

- Realizar el diseño teórico-metodológico, así como el estado del arte a nivel nacional e internacional.
- Analizar de los componentes y subcomandos del sistema de control de versiones Subversion.
- Analizar de los requisitos necesarios para el desarrollo de la interfaz visual para servidor de control de versiones Subversion.
- Realizar las Historias de Usuarios (HU, por sus siglas en español) del sistema para el desarrollo de la interfaz visual para servidor de control de versiones Subversion.
- Implementar la interfaz visual para servidor de control de versiones Subversion.
- Ejecutar las Pruebas de Calidad para validar el desarrollo de la interfaz visual para servidor de control de versiones Subversion.

Los **Métodos de Investigación** que se utilizaron en el desarrollo del trabajo, fueron los siguientes:

Teóricos:

- Analítico-sintético: Este método se utilizó para profundizar en el desarrollo y funcionamiento del sistema de control de versiones Subversion. Mediante la información obtenida del estudio del estado del arte, las tecnologías, metodologías, leguajes y herramientas; la utilización de este método permitió conocer ventajas y su correcta utilización para realizar el control de versiones.
- Modelación: Este método se utiliza para realizar los diagramas necesarios en el desarrollo y la mejor comprensión de la aplicación web.
- Histórico-lógico: Este método se utiliza para realizar un estudio tanto nacional como internacional de la evolución de los sistemas de control de versiones.

Este trabajo se encuentra estructurado por 4 capítulos, que mostramos a continuación:

Capítulo 1: Fundamentación Teórica. Se exponen conceptos, tecnologías y tendencias de la propuesta presentada.
Capítulo 2: Características del sistema, se ofrece una visión práctica del sistema, los requisitos funcionales y no funcionales, además de una propuesta del sistema.
Capítulo 3: Diseño del sistema, se presenta una vista interna del sistema, diagrama de clases del diseño.
Capítulo 4: Implementación y Pruebas del sistema, se muestran los elementos utilizados en la implementación del sistema y los resultados de la misma.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace un estudio sobre el sistema de control de versiones Subversion, así como sus principales conceptos, componentes y subcomandos. Además se describen las herramientas, lenguajes de programación y metodologías de desarrollo que serán empleadas en el desarrollo de la aplicación.

1.2 Conceptos básicos:

- **Subversion:** Es un sistema de control de versiones libre y de código fuente abierto, empleado en la administración de archivos utilizados en el desarrollo de software o contenido.
- **Repositorio:** Es un almacén central de datos que guarda la información en forma de árbol de archivos. (3)

1.3 Sistemas de control de versiones Subversion

Subversion, también conocido como SVN, es un sistema de control de versiones muy conocido, en especial dentro de la comunidad de desarrolladores de software libre. Está preparado para funcionar en red, y se distribuye bajo una licencia libre de tipo Apache.
De forma centralizada permite realizar el seguimiento de los cambios en archivos empleados en proyectos de software. Basado en el trabajo inicial de CVS (Concurrent Versions System), constituye una implementación más eficiente de este último y ha conseguido mayor popularidad en proyectos de software libre y abierto y en organizaciones empresariales. (4)

1.4 Características de Subversion

- Mantiene versiones no sólo de archivos, sino también de directorios.
- En los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- Posibilidad de elegir el protocolo de red. Además de un protocolo propio (svn), puede trabajar sobre http (o https) mediante las extensiones WebDAV.
- Soporte tanto de ficheros de texto como de binarios.

- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.
- Mayor eficiencia en la creación de ramas. (5)

1.5 Algunos de los componentes y sus subcomandos utilizados desde la línea de comandos:

svn: El programa cliente de línea de comandos.

- help: Muestra el mensaje de ayuda de svn.
- import: Envía de forma recursiva los cambios de PATH en URL.

svnadmin: Permite crear, configurar, o reparar un repositorio.

- help: Muestra el mensaje de ayuda de svnadmin.
- create: Crea un nuevo repositorio.
- dump: Vuelca el contenido de un sistema de ficheros por la salida estándar.
- load: Lee un flujo “con formato de volcado” de la entrada estándar.
- hotcopy: Realiza una copia del repositorio.
- recover: Recupera cualquier estado perdido en el repositorio.

svnlook: Permite examinar las revisiones y transacciones realizadas en un repositorio

- help: Muestra el mensaje de ayuda de svnlook.
- info: Muestra el autor, la fecha y el mensaje de informe de cambios.
- changed: Muestra qué rutas fueron modificadas.

svndumpfilter: Permite la eliminación de antecedentes de un archivo de volcado.

- include: Permite sólo al grupo de rutas solicitado, pasar al flujo de datos del volcado. (6)

1.6 Algunas herramientas existentes en el mundo para la interacción con el sistema de control de versiones Subversion.

Visual SVN Server:

Permite instalar y configurar un servidor de Subversion en la plataforma Windows. Gracias a su robustez y a su facilidad se puede utilizar tanto en una pequeña empresa como en una corporación. Está basado en estándares abiertos que ofrecen estabilidad, seguridad y performance. Sus principales características son: funciona con poca configuración, consola de administración poderosa y administración remota. A pesar de todas las cualidades que posee la herramienta; es un software privativo, no funciona sin la instalación en el equipo de un cliente SVN y está hecha solo para Windows. (7)

CommitMonitor:

Herramienta para monitorizar los nuevos commits que se produzcan en repositorios SVN, Se puede configurar con todos los repositorios que queramos, además de configurar la frecuencia de comprobación de nuevos cambios. Los nuevos commits se mostrarán en una notificación del sistema junto al icono de la aplicación. En el aviso veremos el número de cambios y los detalles cada commit. Es útil combinarlo con TortoiseSVN para comprobar en el visor los cambios que se han producido. CommitMonitor reside en memoria consumiendo muy pocos recursos. Se puede configurar como servicio y en Windows lo tendremos como un icono más del sistema junto al reloj. Con esta aplicación ya no hay que preguntar si alguien ha hecho commit o no, sino que estaremos atentos a cualquier cambio del repositorio en tiempo real y quién lo ha hecho. CommitMonitor está disponible bajo la GNU GPL v2. A pesar de todas las cualidades que posee la herramienta se debe destacar que es un una aplicación desktop y no funciona sin la instalación en el equipo de un cliente SVN. (8)

RapidSVN:

RapidSVN es una plataforma visual para el sistema Subversion escrito en C++. Producto de fácil manejo para los usuarios principiantes pero lo suficientemente potente y con herramientas interesantes para los usuarios avanzados. Entre sus características se encuentran:

- **Simple** - proporciona una interfaz fácil de usar para las funciones de Subversion.

- **Eficiente** - sencillo para los principiantes, pero lo suficientemente flexible como para aumentar la productividad de los usuarios de Subversion experimentados.
- **Portable** - se ejecuta en cualquier plataforma en la que Subversion y wxWidgets se puedan ejecutar: Linux, Windows, Mac OS / X, Solaris, etc. Está completamente escrito en C++ y licenciado bajo la GNU GENERAL PUBLIC v3 LICENCIA. (9)

La herramienta es una aplicación dependiente de un par de marcos. Si se necesita compilar RapidSVN se debe descargar, configurar y compilar estos marcos primero (a menos que se tenga acceso a un paquete binario precompilado con archivos de desarrollo como cabeceras y bibliotecas incluidas).

1.7 Metodología, Lenguajes y Herramientas de desarrollo.

1.7.1 Metodología de desarrollo.

Programación Extrema (XP)

Dentro de las metodologías ágiles o ligeras más conocidas y usadas está Extreme Programming o Programación Extrema (XP), que posibilita la entrega de un producto funcional en un breve intervalo de tiempo. Con XP, se trabaja directamente con el cliente realizando pequeñas iteraciones y como resultado mini entregas cada dos semanas, donde no existe más documentación que el propio código. Se decide emplear la metodología XP para el desarrollo del presente trabajo ya que se adapta a las condiciones en que se realiza el mismo.

A continuación se exponen las razones de la elección:

- Existen guías y documentación suficiente para investigar en este aspecto.
- Es un proyecto pequeño donde todo el trabajo se realiza por una pareja de programadores.
- Los requisitos tienden a cambiar frecuentemente, así, conforme avanza el trabajo, el cliente puede agregar nuevas HU, dividirlas o simplemente eliminarlas. XP permite al equipo la modificación de sus planes de forma ágil.
- No se necesita la generación de tantos artefactos y roles.
- Se tiene la posibilidad de desarrollar algo y probarlo permitiendo terminarlo e integrarlo todo. (10)

1.7.2 Herramienta de Modelado.

Visual Paradigm 8.0

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: así como el análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Visual Paradigm ayuda a una rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (11)

Lenguaje Unificado de Modelado (UML, por sus siglas inglés)

Es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas de software, así como también el modelado de negocios. Se encuentra definido por varios niveles, adecuado para los modelos orientados a objetos.
UML da la destreza necesaria para la creación de sistemas de software que se encuentren bien diseñados, que sean robustos y de fácil mantenimiento, debido que da la habilidad para analizar y diseñar un sistema desde la perspectiva de los objetos. (12)

1.7.3 Lenguaje de programación.

Hypertext Preprocessor (PHP, por sus siglas en inglés).

PHP es un lenguaje de programación multiplataforma, completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos, el código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable, es libre, por lo que se presenta como una alternativa de fácil acceso para todos y permite aplicar técnicas de programación orientada a objetos (POO, por sus siglas en español). (13)

Cascade StyleSheet 3 (CSS, pos sus siglas en inglés)
Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. CSS se utiliza para dar estilo a documentos HTML y XML,

separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. Con lo que se logra además de un ahorro de tiempo una mayor uniformidad en el diseño. (14)

Lenguaje de Marcado de Hipertexto 5 (HTML, por sus siglas en inglés)

Es el lenguaje con el que se escriben las páginas web. Es un lenguaje de hipertexto, es decir, un lenguaje que permite escribir texto de forma estructurada, y que está compuesto por **etiquetas**, que marcan el inicio y el fin de cada elemento del documento. Un documento hipertexto no sólo se compone de texto, puede contener imágenes, sonido, vídeos, etc., por lo que el resultado puede considerarse como un documento multimedia. Los documentos HTML deben tener la extensión **html** o **htm**, para que puedan ser visualizados en los navegadores que son los encargados de interpretar el código HTML de los documentos, y de mostrar a los usuarios las páginas web resultantes del código interpretado. (15)

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. (16)

1.7.4 Entorno de desarrollo integrado (IDE, por sus siglas en inglés)

NetBeans 7.3

Es un entorno de desarrollo muy completo y profesional. Contiene muchas funcionalidades, para distintos tipos de aplicaciones y para facilitar al máximo la programación, la prueba y la depuración de las aplicaciones que se desarrollan. Permite crear aplicaciones Web con PHP, un potente debugger integrado. Por lo que se decidió utilizar NetBeans ya que es multiplataforma, libre y se presenta como una alternativa de fácil acceso para todos. (17)

1.7.5 Framework utilizados

JQuery:

Es una biblioteca JavaScript creada por John Resig, rápido, pequeño y rico en funciones. JQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT Instituto Tecnológico de Massachusetts (del inglés Massachusetts Institute of Technology) y la Licencia Pública General de GNU v2 (del inglés GNU General Public License), permitiendo su uso en proyectos libres y privativos. Entre sus características tiene la facilidad de acceder al documento HTML, modificar la apariencia y contenido de la página y manipular estilos CSS. (18)

Bootstrap 2.0:

Es un framework que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Las mayores ventajas es que podemos crear interfaces que se adapten a los distintos navegadores, se integra perfectamente con las principales librerías Java Script, por ejemplo JQuery además de contar con numerosos componentes webs que nos ahorrarán mucho esfuerzo y tiempo. (19)

1.7.6 Servidor Web

Apache 2.2

Es un servidor web HTTP, para plataformas Unix, Microsoft Windows, Macintosh y otras. Presenta muchas características y funciones que lo califica como un servidor robusto y rápido de código abierto. (20)

1.7.7 Técnica de desarrollo

Asynchronous JavaScript and XML (Ajax, por sus siglas en ingles)

Es una técnica de desarrollo web para crear aplicaciones interactivas, describe un nuevo acercamiento a usar un conjunto de tecnologías existentes juntas, incluyendo las siguientes: HTML o XHTML, CSS, JavaScript, el DOM (Document Object Model, por sus siglas en inglés), entre otras. Ajax es una tecnología

asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. (21)

1.8 Conclusiones.

Durante la realización de este capítulo se hizo un estudio sobre el sistema de control de versiones Subversion, conceptos, componentes, subcomandos esenciales, así como el análisis de las herramientas existentes en el mundo para la interacción con el sistema de control de versiones Subversion. Por último se seleccionó la metodología, las tecnologías y herramientas necesarias para el desarrollo de la aplicación web.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se elabora una propuesta de la aplicación web a implementar detallando las características que debe tener el sistema para un mejor entendimiento. También se presentan los requisitos funcionales y no funcionales, indispensables en el proceso de desarrollo de la aplicación web y además se confeccionan las Historias de Usuario (HU) para cada iteración definida en correspondencia con la metodología de desarrollo seleccionada.

2.2 Propuesta del sistema

La solución propuesta está diseñada para aquellos especialistas encargados de la administración de los repositorios Subversion. El sistema a desarrollar es una aplicación web; la cual le brindará al administrador la posibilidad de gestionar los repositorios, así como gestionar usuarios , gestionar permisos, listar datos de revisión, realizar copia segura de los repositorios, recuperar repositorio, separar repositorios combinados, migrar repositorios e importar esquemas a un repositorio.

2.3 Modelo de Dominio

Para el mejor entendimiento de los conceptos manejados para el desarrollo del sistema propuesto se realiza el Modelo de Dominio, el cual captura los tipos de conceptos más importantes en el contexto del sistema, ayuda a comprender los conceptos que utilizan los usuarios y con los que deberá trabajar la aplicación. Los objetos del dominio representan los eventos que suceden en el entorno que trabaja el sistema Subversion.

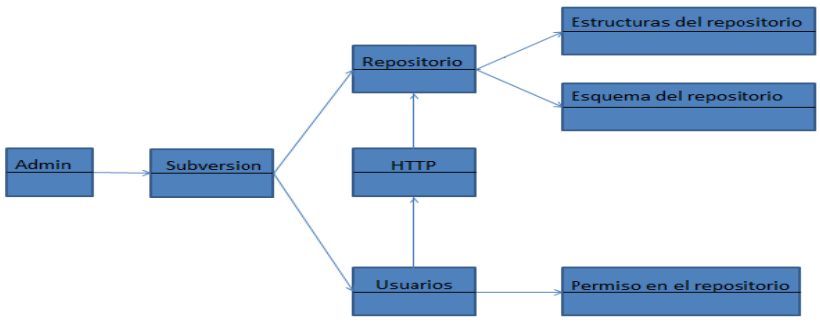


Figura. 1 Modelo de dominio

2.4 Requerimientos funcionales

- **Gestionar repositorio.**

  RF 1. Adicionar repositorio.

  RF 2. Eliminar repositorio.

- **Gestionar usuarios.**

  RF 3. Adicionar usuario.

  RF 4. Eliminar usuario.

  RF 5. Modificar usuario.

- **Gestionar permisos.**

  RF 6. Adicionar permisos.

  RF 7. Eliminar permisos.

  RF 8. Modificar permisos.

RF 9. Listar datos de revisión.

RF 10. Realizar copia segura de los repositorios.

RF 11. Recuperar repositorio.

RF 12. Separar repositorios combinados.

RF 13. Migrar repositorios.

RF 14. Importar esquemas a un repositorio

2.5 Requerimientos no funcionales

- Usabilidad:

El administrador necesita tener un nivel medio o superior en la administración de Subversion para operar con el sistema.

- Seguridad:

  - Confidencialidad: la información manejada por el sistema está protegida de acceso no autorizado.

- Integridad: la información manejada por el sistema es objeto de cuidadosa protección contra la corrupción, de la misma forma será considerada igual a la fuente o autoridad de los datos.
- Disponibilidad: a los usuarios autorizados se les garantizará el acceso a la información. Los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

- Hardware

Servidor: Se debe disponer de una computadora de 256 MB de RAM.

- Software:

Se requiere la instalación de un servidor web, del navegador Mozilla Firefox y la herramienta Subversion.

- Interfaz de usuario:

La aplicación propuesta poseerá una interfaz amigable, sencilla e intuitiva al usuario que se relacionen con el sistema.

2.6 Usuarios relacionados con el sistema

Se define como usuarios relacionados con el sistema a aquellos que interactúan de una u otra forma con la aplicación y obtienen un resultado de todos los procesos que se ejecutan en ella.

Tabla 1. Usuario relacionado con el sistema

Nombre	Justificación
Administrador	El Administrador es el encargado de gestionar todos los repositorios del sistema, así como mantener el control sobre ellos, además de gestionar los usuarios otorgándoles permisos y privilegios.

2.7 Fase de Exploración

La fase de Exploración es la primera fase definida por la metodología XP. Esta fase comienza por la creación de una serie de historias, llamadas HU, las cuales definen mediante su redacción qué es lo que

verdaderamente necesita el cliente. Es aquí donde se define el alcance real del sistema, permitiendo una familiarización del equipo de desarrollo con las herramientas y tecnologías a usar en la solución.

2.7.1 Historias de Usuarios

Las Historias de Usuarios son las técnicas que utiliza la metodología para especificar los requisitos del software. Estas son escritas desde la perspectiva del cliente, donde describen de forma breve las características que el sistema debe realizar, aunque también los desarrolladores pueden brindar su ayuda en la identificación de las mismas. El contenido de las historias debe ser concreto, sencillo, dinámico y flexible.

2.7.2 Clasificación de las Historias de Usuarios

- La prioridad en el negocio:

Alto: Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.
Medio: Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
Bajo: Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

Las HU serán representadas mediante tablas divididas por las siguientes secciones:

- Número: Número de la historia de usuario incremental en el tiempo.
- Nombre de Historia de Usuario: El nombre de la historia de usuario sería para identificarlas mejor entre los desarrolladores y el cliente.
- Usuario: Personas involucradas en el desarrollo de la HU.
- Iteración Asignada: Número de la iteración.
- Prioridad en negocio: se le otorga una prioridad (Alta, Media, Baja) a las HU de acuerdo a la necesidad de desarrollo.
- Riesgo en Desarrollo: se le otorga una medida de (Alto, Medio, Bajo), a la ocurrencia de errores en el proceso de desarrollo de la HU

- Puntos estimados: Tiempo estimado que se demorará el desarrollo de la HU; cada punto es considerado como una semana de trabajo.
- Puntos Reales: Tiempo que se demoró en realidad el desarrollo de la HU.
- Descripción: Breve descripción de la HU.
- Observaciones: Señalamiento o advertencia del sistema.
- Prototipo de interfaz: Prototipo de interfaz si aplica.

A continuación se expone una muestra de las HU definidas por el equipo de desarrollo el resto se describen en el **Anexo 1** del presente documento:

Tabla 2. HU Gestionar repositorio

Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Gestionar Repositorio
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Linares Zamora. Rafael A. Riquene Llorente	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados**: 2
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: El sistema permite introducir los datos del repositorio, también brinda la opción de eliminar dichos repositorios en caso de ser necesario.	
Observaciones: El administrador del sistema debe estar previamente autenticado.	
Prototipo de interfaz: 	

2.8 Fase de Planificación

El proceso de planeación en la metodología XP inicia con la creación de una serie de HU que se utilizan para especificar los requisitos del software a construir, se describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada HU. Este se expresa utilizando como medida el punto.

Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. Esta estimación incluye todo el esfuerzo asociado a la implementación de la HU.

2.8.1 Estimación de esfuerzo por Historias de Usuarios

Para el desarrollo de la aplicación propuesta se aplicó la estimación de esfuerzo para cada HU identificada, llegando a los resultados que se muestran a continuación.

Tabla 3. Estimación de esfuerzo por HU

Historia de Usuario	Puntos de estimación
Gestionar repositorio.	2
Gestionar usuario	1
Gestionar permisos	1
Listar datos de revisión.	1
Realizar copia segura de los repositorios.	1
Recuperar repositorio.	1
Separar repositorios combinados.	1
Migrar repositorios.	1
Importar esquemas a un repositorio	1

2.8.2 Plan de Iteraciones

Después de identificadas y descritas las HU y estimar el esfuerzo dedicado a la realización de cada una de ellas, se procede a la planificación de la fase de implementación donde se seleccionan las HU a desarrollar durante cada iteración para mejorar el desempeño del equipo de desarrollo.

Las HU responden al siguiente orden de desarrollo preestablecido:

Iteración 1
En esta iteración se implementan las HU 1, 2, 3 dando al sistema las primeras funcionalidades.

Iteración 2
En esta iteración se implementan las HU 4, 5, 6, 7, 8, 9 que facilitarán el mejor funcionamiento de la aplicación.

2.8.3 Plan de duración de las iteraciones

Este plan se encarga de mostrar las HU en el orden en que se implementarán en cada iteración, así como la duración estimada de las mismas. Las iteraciones, que fueron 2, duraron un total de 11 semanas. El tiempo de duración de las HU en cada iteración puede verse en la tabla siguiente:

Tabla 4. Plan de duración de iteraciones

Iteración	Orden de la HU a implementar.	Duración total
1	HU-1, HU-2, HU-3	4 semanas
2	HU-4, HU-5, HU-6, HU-7, HU-8, HU-9	6 semanas

2.8.4 Plan de entregas

En este plan se detalla la fecha fin de cada una de las iteraciones definidas. A continuación se muestra el plan de entrega para cada iteración:

Tabla 5. Plan de entregas

Iteración	Fecha de Entrega
1	17 de abril de 2013
2	29 de mayo de 2013

2.9 Conclusiones.

Con la realización de este capítulo se trazan las bases para el desarrollo del sistema ya que se describe la propuesta del sistema además quedan definas todas las funcionalidades del sistema de acuerdo a las necesidades de los expertos. Se realizó la fase de Exploración definida por la metodología, en cual quedan plasmadas todas las historias de usuario relacionadas con el sistema que fueron creadas para cada versión del programa, así como el plan de entrega de cada una de ellas, y la fecha donde se publican cada una de estas versiones. Se obtiene además una planificación del tiempo de desarrollo de las iteraciones y el orden en que se implementan las historias de usuario.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se describe la fase de diseño del sistema propia de la metodología. Se definen los patrones a seguir y la arquitectura utilizada para la implementación del sistema. Se realiza el diseño de la aplicación web para satisfacer los requerimientos del software y quedarán definidas las tarjetas CRC (Colaborador - Responsabilidad - Clase) como técnica de diseño.

3.2 Arquitectura de Software

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de Software o Arquitectura lógica. La misma hace referencia a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para guiar el desarrollo de software dentro de un sistema informático. De tal modo que los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común posibilitando la compatibilidad el objetivo deseado.

3.3 Patrón Arquitectónico

El patrón arquitectónico aplicado es el de n capas, específicamente 2 capas, presentación y lógica del negocio. Elegimos este patrón arquitectónico puesto que su objetivo principal es separar los diferentes aspectos del desarrollo, es decir, las cuestiones de presentación y lógica de negocio, permitiendo que un cambio en una de las capas no genere cambios en las demás.

Figura. 2 Arquitectura del sistema

La arquitectura del sistema está compuesta:

- En la capa de presentación: se encuentran las clases que son responsables de visualizar el contenido al usuario.
- La capa de negocio: incluye las clases que implementan la lógica del negocio.

La comunicación entre las capas se realiza a través de interfaces, lo que permite la organización de la implementación y la estandarización del código.

3.4 Patrones de diseño

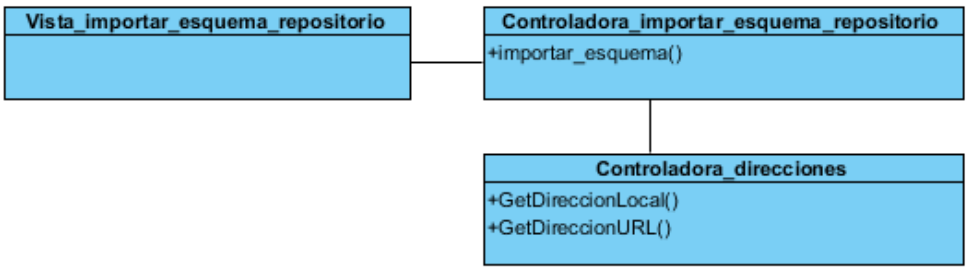
3.4.1 Patrones para asignar responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Se emplearon en el sistema los siguientes patrones GRASP:

- **Creador**

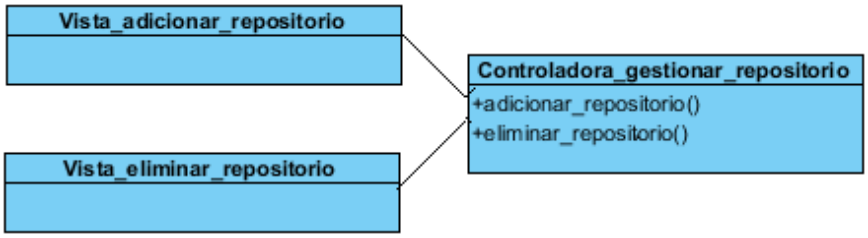
Este patrón es el responsable de asignarle a una clase la responsabilidad de crear una instancia de otra. En la solución, las clases controladoras tienen una instancia de la clase donde se encuentran las direcciones de los repositorios, tanto local como url, así como cada una de las vistas tiene una instancia de su controladora. Ejemplo la clase interfaz importar_esquema_repositorio es la encargada de crear una instancia de la controladora importar_esquema_repositorio.php, la cual a la vez crea una instancia de la controladora direcciones.php, como se muestra en el diagrama siguiente.



- **Controlador**

Este patrón asigna la responsabilidad del manejo de los eventos de un sistema a una o varias clases, es usado en cada una de las clases controladoras. En la solución, cada petición que se genera es manejada por una clase controladora.php. Ejemplo la clase controladora gestionar_repositorios la cual se encarga

de controlar las peticiones realizadas desde la clase interfaz adicionar_repositorio o eliminar_repositorio, como se describe en diagrama siguiente:



- **Bajo Acoplamiento**

Patrón que logra una escasa dependencia entre clases. Al realizarse un diseño de clases independientes que puedan soportar los cambios de una manera fácil y permitan la reutilización. Acoplamiento bajo significa que una clase no depende de muchas clases. Un ejemplo de donde se aplica este patrón es en la clase controladora importar_esquema_repositorio.php.

3.5 Tarjetas clase responsabilidad colaborador (CRC)

La metodología de desarrollo XP como parte de la fase de diseño propone el modelado de CRC, lo que constituye un modelo simple de organizar las clases más relevantes para las funcionalidades del sistema. Este modelado CRC utiliza tarjetas, con el objetivo de desarrollar una representación organizada de las clases, las cuales tienen los siguientes datos:

- Nombre
- Responsabilidades
- Colaboraciones

Una responsabilidad es cualquier cosa que la clase sabe o hace. Los colaboradores son aquellas clases que se requieren para que una clase reciba la información necesaria para completar una responsabilidad.

A continuación se muestra una breve descripción y las tarjetas CRC correspondientes a tres de las clases funcionales del sistema, el resto se detallan en el **Anexo 2**.

Clase gestionar_ repositorio

En esta clase se implementan las funcionalidades:

Adicionar repositorio: Para esto se especifica el nombre que identifica al repositorio en el sistema y la dirección local donde será ubicado el mismo. El comando a intercambiar con el servidor svn es: **svnadmin create** *directorioLocal.*

Eliminar repositorio: Para esto se especifica la dirección local donde se encuentra el mismo. El comando a intercambiar con el servidor svn es: **svn delete** *directorioLocal.*

Tabla 6. Tarjeta CRC Clase gestionar_ repositorio

Clase gestionar_repositorio	
Descripción: Clase para la gestión de los repositorios	
Responsabilidad:	**Colaborador**
Adicionar repositorio Eliminar repositorio	adicionar_repositorio, eliminar_repositorio

Clase importar_esquema_repositorio

En esta clase se implementan la funcionalidad:

Importar esquema al repositorio: Para ello se debe especificar el repositorio que contendrá el esquema y el directorio con el esquema que se desea aplicar. El comando que se intercambia con el servidor svn es: **svn import** *rutaEsquema urlRepositorio***.**

Tabla 7. Tarjeta CRC importar_esquema_repositorio

Clase importar_esquema_repositorio	
Descripción: Clase que importa un esquema al repositorio, consiste en agregar a un repositorio un conjunto de directorios que serán aplicados a las copias de trabajo que utilice el repositorio.	
Responsabilidad:	**Colaborador**

Importar esquema al repositorio	importar_esquema_repositorio

Clase migrar_repositorio

En esta clase se implementan la funcionalidad:

Migrar repositorio: Para ello se debe especificar el nombre del repositorio que se desea migrar y la ruta del nuevo repositorio. Los comandos a intercambiar con el servidor svn son: **Svnadmin dump** *[ruta del repositorio]* < *[nombre del fichero de volcado]* y **Svnadmin load** *[ruta del nuevo repositorio]* < *[nombre del fichero de volcado].*

Tabla 8. Tarjeta CRC Clase migrar_repositorio

Clase migrar_repositorio	
Descripción: Clase que permite migrar un repositorio a otro repositorio.	
Responsabilidad:	**Colaborador**
Migrar repositorio	migrar_repositorio

3.6 Conclusiones

En el presente capítulo se ejecutó la fase de diseño del sistema propia de la metodología. Se definieron los patrones a seguir y la arquitectura utilizada para la implementación del sistema. Se realizó el diseño de la aplicación web para satisfacer los requerimientos del software, así como la descripción de las historias de usuarios divididas por iteraciones y la planificación del esfuerzo dedicado al desarrollo de cada una de ellas, en el orden en que se les dará cumplimiento, y quedaron definidas las tarjetas CRC como técnica de diseño.

CAPÍTULO 4: IMPLEMENTACION Y PRUEBA

4.1 Introducción

En el presente capítulo se describen las tareas de ingeniería generadas por cada HU y las pruebas planteadas por la metodología, diseñando los casos de pruebas por cada HU definida, para así evaluar la aplicación. Además se presenta el diagrama de despliegue y se mostrarán a través de tablas los resultados obtenidos durante la realización de las pruebas de aceptación.

4.2 Fase de Implementación

La metodología plantea que la implementación de un software se hace iterativamente, obteniendo al culminar cada iteración un producto funcional, que debe ser probado y mostrado al cliente. Durante el transcurso de las iteraciones, se realiza la implementación de las HU definidas por el cliente y descritas por el equipo de desarrollo en la etapa de Planificación. Como parte de este plan, se descomponen estas HU en tareas de la ingeniería las cuales son asignadas a los programadores para ser implementadas durante la iteración correspondiente. (22)

4.3 Tareas de la ingeniería

Las tareas de la ingeniería serán representadas mediante tablas divididas por las siguientes secciones:

- Número tarea: los números deben ser consecutivos.
- Número historia de usuario: número de la historia de usuario a la que pertenece la tarea.
- Nombre tarea: nombre que identifica a la tarea.
- Tipo de tarea: las tareas pueden ser de: desarrollo, corrección, mejora, otra (especificar)
- Puntos estimados: tiempo estimado en semanas que se le asignará a su desarrollo.
- Fecha inicio: fecha en que inicia el desarrollo de la tarea.
- Fecha fin: fecha en que finaliza el desarrollo de la tarea.
- Programador responsable: nombre y apellidos del programador.
- Descripción: breve descripción de la tarea.

A continuación se muestran algunas de las tareas de ingeniería correspondientes a dos de las HU y el resto se detallan en el **Anexo 3** del presente trabajo.

Tabla 9. Tarea de ingeniería: Gestionar repositorio

Tarea de Ingeniería	
Número de tarea: 1	**Número de HU:** 1
Nombre de tarea: Gestionar repositorio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha de inicio: 20/3/2013	**Fecha de fin:** 3/4/2013
Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora	
Descripción: Implementar la funcionalidad de adicionar o eliminar un repositorio.	

Tabla 10. Tarea de Ingeniería: Gestionar usuario

Tarea de Ingeniería	
Número de tarea: 2	**Número de HU:** 2
Nombre de tarea: Gestionar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 4/4/2013	**Fecha de fin:** 10/4/2013
Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora	
Descripción: Implementar las funcionalidades de adicionar, eliminar o modificar un usuario.	

4.1 Pruebas del software

XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores y encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida además de comprobar que dicha funcionalidad sea la esperada por el cliente. (23)

4.2 Diagrama de despliegue

Un diagrama de despliegue representa la distribución física del sistema en términos de cómo se distribuirán las funcionalidades entre los nodos, cada nodo representa un recurso de cómputo, siendo estos procesadores o dispositivos hardware que se necesitarán para el despliegue del sistema.

La comunicación entre los nodos se rige por los protocolos HTTP para la asociación entre la computadora cliente y el servidor de aplicaciones.

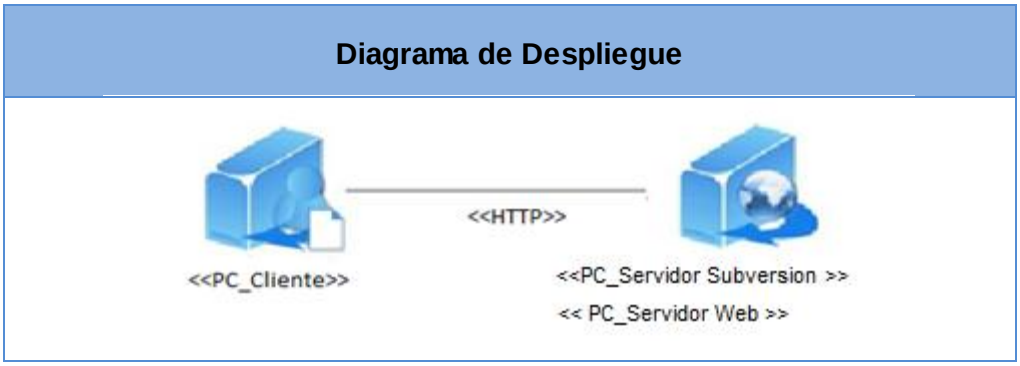


Figura. 3 Diagrama de Despliegue

- **PC_Cliente:** Es la pc que se conecta a la aplicación para visualizar los componentes web desarrollados.
- **PC_Servidor:** El servidor tendrá instalado el sistema operativo Windows y contiene la aplicación web y la herramienta Subversion. El servidor se encarga de controlar, ejecutar y gestionar todos los pedidos de acciones que el usuario realiza a través de la PC_Cliente, empleándose como protocolo de comunicación entre el cliente y el servidor web el HTTP.

4.3 Pruebas de Aceptación

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las HU. En estas serán probadas las funcionalidades exigidas por el cliente, descritas en las HU que se han implementado. Las pruebas de aceptación se llevarán a cabo redactando los casos de prueba, teniendo en cuenta el orden de las HU y la prioridad que ha sido asignada a las funcionalidades.

Las pruebas de aceptación correspondiente a cada una de las funcionalidades serán representadas mediante tablas divididas por las secciones siguientes:

- Código de la prueba de aceptación.
- Número de la historia de usuario a la que se le realiza la prueba.
- Nombre de la funcionalidad.
- Descripción de la funcionalidad.
- Condiciones de ejecución de la funcionalidad.
- Entrada y pasos de ejecución que realiza el usuario con el objetivo de obtener el resultado esperado.
- Resultado esperado.
- Evaluación de la prueba.

A continuación se muestra el caso de prueba de aceptación de la HU Gestionar repositorio y el resto se detallan en el **Anexo 4** del presente trabajo.

Tabla 11: Caso de prueba de aceptación: Gestionar repositorio.

Caso de prueba de aceptación	
Código: HU1-P1	**Número de HU:** 1
Nombre: Gestionar repositorio	
Descripción: La aplicación debe dar la oportunidad de adicionar un repositorio o de eliminarlo.	
Condiciones de ejecución: La solicitud de registrar la entrada de un repositorio o de la eliminación del mismo.	
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción de adicionar repositorio e	

introduce el dato siguiente: nombre del repositorio o escoge la opción eliminar repositorio y selecciona en el listado de repositorios cual desea eliminar.
Resultado esperado **Correcto:** Si los datos entrados son válidos, se creará un nuevo repositorio o se eliminara un repositorio. **Incorrecto:** Si el campo está vacío, se muestra un mensaje de error “El campo está vacío”.
Evaluación de la prueba: Prueba satisfactoria.

El resultado de las pruebas se representa en la siguiente figura.

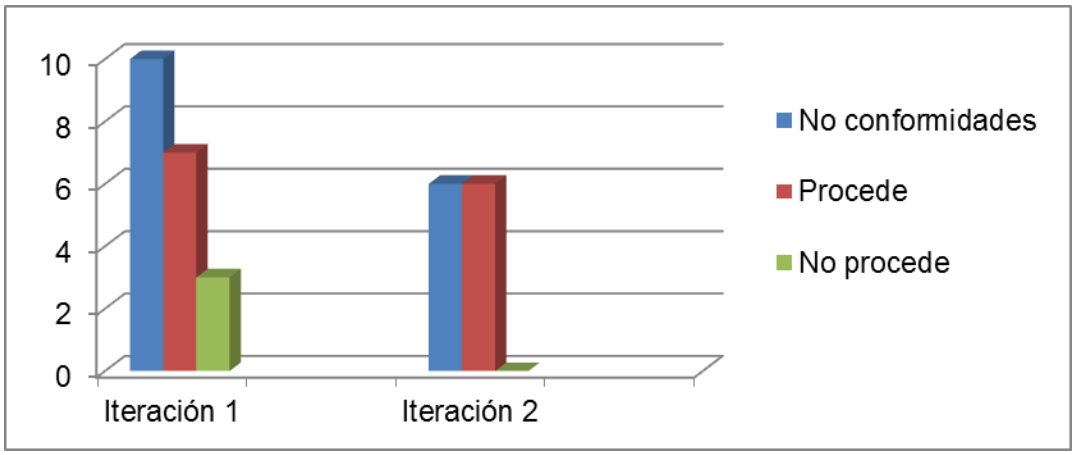


Figura. 4 Resultado de pruebas

4.4 Conclusiones

En este capítulo se realizaron las tareas de la ingeniería correspondientes a cada HU listas para su implementación y quedó diseñado el modelo de despliegue de la aplicación. Se plantearon además las pruebas a seguir durante la elaboración del mismo, donde se ejecutaron las pruebas de aceptación logrando la satisfacción del cliente con el software desarrollado, dándole de esta manera cumplimiento a los requerimientos planteados por el cliente.

CONCLUSIONES GENERALES

El objetivo fundamental de este trabajo de diploma “Interfaz visual para servidor de control de versiones Subversion”, consistió en proporcionar una solución al problema de como agilizar el proceso de administración del sistema de control de versiones Subversion. El aporte fundamental de esta investigación resultó ser el diseño e implementación de una aplicación web para la administración de repositorios.

Se realizó el estudio del sistema de control de versiones Subversion, conceptos, componentes, subcomandos esenciales, así como de algunas de las herramientas utilizadas en el mundo para la interacción con el mismo. Se seleccionó la metodología, las tecnologías y herramientas necesarias para el desarrollo de la aplicación.

Se trazaron las bases para el desarrollo describiendo la propuesta del sistema además quedaron definas todas las funcionalidades de acuerdo a las necesidades de los expertos. Se realizó la fase de Exploración definida por la metodología, en la cual quedaron plasmadas todas las historias de usuario relacionadas con el sistema. Se ejecutó la fase de diseño del sistema propia de la metodología, así como la definición los patrones a seguir y la arquitectura utilizada para la implementación del sistema. Se realizó el diseño de la aplicación web para satisfacer los requerimientos del software, quedando definidas las tarjetas CRC como técnica de diseño. Se realizaron las tareas de la ingeniería correspondientes a cada HU listas para su implementación. Se desarrolló una aplicación capaz de gestionar repositorios, usuarios, permisos entre otras funcionalidades, quedó diseñado el modelo de despliegue de la aplicación. Se realizaron las pruebas de aceptación lo que permitió demostrar que la aplicación cumple satisfactoriamente con los requisitos que garantizan su correcto funcionamiento. De esta manera se le da solución al problema científico cumpliendo los objetivos específicos planteados en la investigación del presente trabajo.

.

RECOMENDACIONES

A continuación se listan las recomendaciones en vista de posibles mejoras a la aplicación.

- Implementar una funcionalidad para lograr la conexión a los repositorios Subversion a través del protocolo de red https.
- Implementar nuevas funcionalidades para lograr la gestión de usuarios a través de LDAP.
- Implantar la herramienta en el Centro TLM.

BIBLIOGRAFÍA

1. **Oviedo, Juan.** *Control de Versiones.*
2. Control de Versiones. *Control de Versiones.* [En línea] http://producingoss.com/es/vc.html.
3. Control de versions subversion. *Control de versions subversion.* [En línea] http://svnbook.spears.at/nightly/es/svn-book.html.
4. El Mundo es Open Source. *El Mundo es Open Source.* [En línea] http://blogs.antartec.com/opensource/2008/11/antartec-es-open-source-parte-5-subversion-svn/.
5. [En línea] http://recursostic.educacion.es/observatorio/web/es/software/software-general/548-luis-garcia.
6. Control de versions subversion. *Control de versions subversion.* [En línea] http://svnbook.spears.at/nightly/es/svn-book.html.
7. [En línea] http://fabiantapia.wordpress.com/2010/11/18/subversion-en-windows-e-integrado-con-visual-studio.
8. [En línea] http://www.genbetadev.com/sistemas-de-control-de-versiones/commitmonitor-una-sencilla-aplicacion-para-estar-atento-a-cada-nuevo-commit-en-el-repositorio.
9. [En línea] http://translate.google.com.cu/translate?hl=es&sl=en&u=http://rapidsvn.tigris.org/&prev=/search%3Fq%3Drapidsvn%26biw%3D1366%26bih%3D596.
10. **Danae Pérez Arias, Rainer Segura Peña.** *Servicio de Facturación para Elastix*.
11. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.
12. [En línea] http://www.docirs.cl/uml.htm.
13. Monografías. *Monografías.* [En línea] http://foros.monografias.com/showthread.php/60249-Caracteristicas-del-PHP.
14. Guía breve de CSS. Guía breve de CSS. [En línea] http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.
15. aula.com. *aula.com*. [En línea] http://www.aulaclic.es/html/t_1_1.htm.
16. LIBROSWEB. *LIBROSWEB.* [En línea] http://librosweb.es/javascript/capitulo_1.html.
17. [En línea] http://www.fdi.ucm.es/profesor/luis/fp/devtools/NetBeansUso.html .
18. jQuery. *jQuery*. [En línea] http://jquery.com.
19. GENBETA. *GENBETA.* [En línea] http://www.genbetadev.com/frameworks/bootstrap .
20. Ciberaula. *Ciberaula.* [En línea] http://linux.ciberaula.com/articulo/linux_apache_intro .
21. *Introducción a AJAX*.
22. **Pressman., R. S.** *Ingeniería del Software: Un Enfoque Práctico.*
23. **Crispin, T. H. Lisa.** Testing Extreme Programming. *Testing Extreme Programming.* [En línea] [Citado el: 5 de 4 de 2013.] http://www.ebooksbay.org/The_Digital_Library/2008/01/23/Testing_Extreme_Programming__The_XP_Series_.

ANEXOS

Anexo1: Tablas de Historia de Usuario.

Historia de Usuario	
Número: 2	**Nombre de Historia de Usuario** Gestionar usuario.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Linares Zamora. Rafael A. Riquene Llorente	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: El administrador del sistema inserta los datos del usuario. El administrador puede también modificar o eliminar dichos usuarios en caso de ser necesario.	
Observaciones: El administrador del sistema debe estar previamente autenticado.	
Prototipo de interface: 	

Tabla 3: HU Nro. 2: Gestionar usuario.

Historia de Usuario	
Número: 3	**Nombre de Historia de Usuario:** Gestionar permisos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Linares Zamora.	**Iteración Asignada:** 1

Rafael A. Riquene Llorente	
Prioridad en negocio: Media	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: El administrador puede también modificar o eliminar permisos en caso de ser necesario.	
Observaciones: El administrador del sistema debe estar previamente autenticado.	
Prototipo de interface: 	

Tabla 4: HU Nro. 3: Gestionar permisos.

Historia de Usuario	
Número: 4	**Nombre de Historia de Usuario:** Listar datos de revisión.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Linares Zamora. Rafael A. Riquene Llorente	**Iteración Asignada:** 1
Prioridad en negocio: Media	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: El sistema muestra el autor, la fecha, la hora, el número de la revisión y el mensaje de la última revisión realizada sobre los repositorios.	

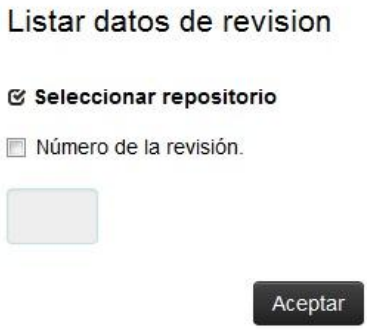
Observaciones: El administrador del sistema debe estar previamente autenticado.
Prototipo de interface: 

Tabla 5: HU Nro. 4: Listar datos de revisión.

Historia de Usuario	
Número: 5	**Nombre de Historia de Usuario:** Realizar copia segura de repositorio.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Linares Zamora. Rafael A. Riquene Llorente	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: El administrador del sistema copia el repositorio para un lugar determinado.	
Observaciones: El administrador del sistema debe estar previamente autenticado.	
Prototipo de interface:	

Tabla 6: Nro. 5: Realizar copia segura de repositorio.

Historia de Usuario	
Número: 6	**Nombre de Historia de Usuario:** Recuperar repositorio.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Linares Zamora. Rafael A. Riquene Llorente	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: El administrador del sistema realiza las tareas de recuperación en un repositorio determinado después de un fallo del mismo, donde se especifica (nombre del repositorio).	
Observaciones: El administrador del sistema debe estar previamente autenticado.	
Prototipo de interface: 	

Tabla 7: HU NRO. 6: Recuperar repositorio.

Historia de Usuario	
Número: 7	**Nombre de Historia de Usuario:** Separar repositorios combinados.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Linares Zamora. Rafael A. Riquene Llorente	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: El administrador del sistema separa aquellos proyectos que se encuentran ubicados en un mismo repositorio.	
Observaciones: El administrador del sistema debe estar previamente autenticado.	
Prototipo de interface: 	

Tabla 8: HU NRO. 7: Separar repositorios combinados**.**

Historia de Usuario	
Número: 9	**Nombre de Historia de Usuario:** Migrar repositorio.
Modificación de Historia de Usuario Número: Ninguna	

Usuario: Roberto Linares Zamora. Rafael A. Riquene Llorente	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: El administrador del sistema migra un repositorio a otro.	
Observaciones: El Administrador del sistema debe estar previamente autenticado.	
Prototipo de interface: 	

Tabla 9: HU NRO. 8: Migrar repositorio.

Historia de Usuario	
Número: 10	**Nombre de Historia de Usuario:** Importar esquema a un repositorio.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Linares Zamora. Rafael A. Riquene Llorente	**Iteración Asignada:** 2
Prioridad en negocio: Media	**Puntos estimados**: 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1
Descripción: El administrador del sistema agrega a un repositorio un conjunto de	

directorios que serán aplicados a las copias de trabajo que utilicen el repositorio.
Observaciones: El Administrador del sistema debe estar previamente autenticado.
Prototipo de interface: 

Tabla 10: HU NRO. 9: Importar esquema a un repositorio**.**

Anexo 2: Tarjetas CRC

Clase listar_datos_de_revisión

En esta clase se implementan la funcionalidad:

Listar datos de revisión: Para ello se debe especificar la ruta del repositorio. El comando a intercambiar con el servidor svn es: **svnlook info** *rutaRepositorio*.

Clase listar_datos_revisión	
Descripción: Clase que consiste en mostrar el autor, fecha, hora, cambios y mensaje del usuario.	
Responsabilidad:	**Colaborador**
Listar datos de revisión	listar_datos_revision.

Tabla 17: Tarjeta CRC listar_datos_de_revisión

Clase recuperar_repositorio

En esta clase se implementa la funcionalidad:

Recuperar repositorio: Para ello se especifica la ruta del repositorio. El comando a intercambiar con el servidor svn es: **svnadmin recover** *rutaRepositorio.*

Clase recuperar_repositorio	
Descripción: Clase que permite realizar tareas de recuperación en un repositorio después de un fallo en el mismo.	
Responsabilidad:	**Colaborador**
Recuperar repositorio	recuperar_repositorio

Tabla 18: Tarjeta CRC Clase recuperar_repositorio

Clase copia_segura_repositorio

En esta clase se implementa la funcionalidad:

Copia segura de repositorio: Para ello se especifica la ruta del repositorio y la ruta donde se deben guardar las copia del repositorio seleccionado. El comando a intercambiar con el repositorio svn es: **svnadmin hotcopy** *rutaRepositorio rutaLocalGuardar.*

Clase copia_segura_repositorio	
Descripción: Clase que permite realizar copia de un repositorio en un lugar especificado por el usuario.	
Responsabilidad:	**Colaborador**
Realizar copia segura de un repositorio	copia_segura_repositorio

Tabla 19: Tarjeta CRC Clase copia_segura_repositorio

Clase separar_repositorios_combinados

En esta clase se implementa la funcionalidad:

Separar repositorios combinados: Para ello se debe especificar ruta del repositorio, nombre de la carpeta incluida en el mismo que desea ser separada y los nombre del nuevo repositorio. Los comandos a intercambiar con el servidor svn son: **Svnadmin dump** *[ruta del repositorio] < nombre del fichero de volcado,* **Svndumpfilter include** *[directorio a incluir] < nombre del fichero de volcado > nombre de un nuevo fichero de volcado y* **Svnadmin load** *[ruta del repositorio] < nombre del fichero de volcado.*

Clase separar_repositorios_combinados	
Descripción: Clase que permite separa proyectos que se encuentran en un mismo repositorio en repositorios distintos.	
Responsabilidad:	**Colaborador**
Separar repositorios combinados	separar_repositorios_combinados

Tabla 20: Tarjeta CRC Clase separar_repositorios_combinados

Clase gestionar_ usuario

En esta clase se implementan las funcionalidades:

Adicionar usuario: Para esto se especifica el nombre, la contraseña del usuario, proyecto o repositorio al que va a tener permiso y los permisos otorgados. El comando a intercambiar es: **htpasswd –b** *rutaArchoivo nombre contraseña.*

Eliminar usuario: Para esto se especifica el nombre del usuario. El comando a intercambiar es: **htpasswd –D** *rutaArchoivo nombre.*

Modificar usuario: Para esto se especifica el nombre y la contraseña del usuario. El comando a intercambiar es: **htpasswd –b** *rutaArchoivo nombre contraseña.*

Clase gestionar_usuario	
Descripción: Clase que permite gestionar los usuarios	
Responsabilidad:	**Colaborador**
Adicionar usuario Eliminar usuario Modificar usuario	adicionar_usuario, eliminar_usuario, modificar_usuario.

Tabla 21: Tarjeta CRC Clase gestionar_usuario

Clase gestionar_ permisos

Clase gestionar_permisos	
Descripción: Clase que permite gestionar los permisos de los usuarios en un repositorio	
Responsabilidad:	**Colaborador**
Eliminar permiso Modificar permiso	eliminar_permiso, modificar_permiso.

Tabla 22: Tarjeta CRC Clase gestionar_permisos

Anexo 3: Tareas de ingeniería correspondientes a las HU

Tarea de Ingeniería	
Número de tarea: 3	**Número de HU:** 3
Nombre de tarea: Gestionar permisos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 11/4/2013	**Fecha de fin:** 17/4/2013
Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora	

Descripción: Implementar las funcionalidades eliminar o modificar permisos a un usuario sobre el repositorio.

Tabla 25 Tarea de Ingeniería: Gestionar permisos

Tarea de Ingeniería	
Número de tarea: 4	**Número de HU:** 4
Nombre de tarea: Listar datos de revisión.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio:18/4/2013	**Fecha de fin:** 24/4/2013
Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora	
Descripción: Implementar la funcionalidad para mostrar el autor, la fecha, la hora, el número de la revisión y el mensaje de la revisión realizada sobre los repositorios.	

Tabla 26 Tarea de Ingeniería: Listar datos de revisión.

Tarea de Ingeniería	
Número de tarea: 5	**Número de HU:** 5
Nombre de tarea: Realizar una copia segura de los repositorios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 25 /4/2013	**Fecha de fin:** 1/5/2013
Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora	
Descripción: Implementar la funcionalidad para copiar uno o varios repositorios para un lugar determinado, donde se especifica.	

Tabla 27 Tarea de Ingeniería: Realizar copia segura de repositorio.

Tarea de Ingeniería	
Número de tarea: 6	**Número de HU:** 6
Nombre de tarea: Recuperar un repositorio.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 2/5/2013	**Fecha de fin:** 8/5/2013
Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora	
Descripción: Implementar las funcionalidades para realiza las tareas de recuperación en un repositorio determinado después de un fallo del mismo, donde se especifica (nombre del repositorio).	

Tabla 28 Tarea de Ingeniería: Recuperar repositorio**.**

Tarea de Ingeniería	
Número de tarea: 7	**Número de HU:** 7
Nombre de tarea: Separar repositorios combinados.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 9/5/2013	**Fecha de fin:** 15/5/2013
Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora	
Descripción: Implementar la funcionalidad para separar aquellos proyectos que se encuentran ubicados en un mismo repositorio.	

Tabla 29 Tarea de Ingeniería: Separar repositorios combinados**.**

Tarea de Ingeniería	
Número de tarea: 8	**Número de HU:** 8
Nombre de tarea: Migrar repositorio.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio:16 /5/2013	**Fecha de fin:** 22/5/2013

Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora
Descripción: Implementar las funcionalidades para migra uno o varios repositorios a otros repositorios. Para ello se especifica (nombre de los repositorios que desea migrar y las rutas de los nuevos repositorios).

Tabla 30 Tarea de Ingeniería: Migrar repositorio**.**

Tarea de Ingeniería	
Número de tarea: 9	**Número de HU:** 9
Nombre de tarea: Importar esquema a un repositorio.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 23/5/2013	**Fecha de fin:** 29/5/2013
Programador responsable: Rafael Antonio Riquene Llorente Roberto Linares Zamora	
Descripción: Implementar las funcionalidades para agrega a un repositorio un conjunto de directorios que serán aplicados a las copias de trabajo que utilicen el repositorio.	

Tabla 31 Tarea de Ingeniería: Importar esquema a un repositorio.

Anexo 4: Casos de pruebas de aceptación de HU.

Caso de prueba de aceptación	
Código: HU2-P2	**Número de HU:** 2
Nombre: Gestionar usuario	
Descripción: La aplicación debe dar la oportunidad de adicionar, eliminar o modificar un usuario.	
Condiciones de ejecución: La solicitud de registrar la entrada de un usuario, eliminarlo o modificar el mismo.	
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción de adicionar usuario e introduce los datos siguientes: usuario contraseña y los permisos otorgados, o escoge la opción eliminar usuario e	

introduce el dato siguiente: usuario o escoge la opción modificar usuario e introduce los siguientes datos: usuario y contraseña.
Resultado esperado **Correcto:** Si los datos entrados son válidos, o se creará un nuevo usuario, o se eliminará o se modificará. **Incorrecto:** Si los campos están vacíos, se muestra un mensaje de error “No se pueden dejar campos en blanco”.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 32. Caso de prueba de aceptación: Gestionar usuarios**.**

Caso de prueba de aceptación	
Código: HU3-P3	**Número de HU:** 3
Nombre: Gestionar permisos	
Descripción: La aplicación debe dar la oportunidad de eliminar o modificar un permiso.	
Condiciones de ejecución: La solicitud de eliminar un permiso o modificar el mismo.	
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción de eliminar permiso e introduce el dato siguiente: nombre del usuario, o escoge la opción modificar permisos e introduce los siguientes datos: nombre del usuario, nuevos permisos y sobre que repositorio.	
Resultado esperado **Correcto:** Si los datos entrados son válidos, se elimina el permiso, o se modifica. **Incorrecto:** Si los campos están vacíos, se muestra un mensaje de error “No se pueden dejar campos en blanco”.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 33. Caso de prueba de aceptación: Gestionar permisos**.**

Caso de prueba de aceptación	
Código: HU4-P4	**Número de HU:** 4
Nombre: Listar datos de revisión	
Descripción: La aplicación debe dar la oportunidad de mostrar el autor, la fecha, la hora, que acción se realizó y el mensaje de revisión que se realiza sobre el repositorio.	

Condiciones de ejecución: La solicitud de listar los de datos de la revisión.
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción de listar datos de revisión, selecciona un repositorio e introduce el número de revisión en caso de seleccionar esa opción.
Resultado esperado **Correcto:** Si se selecciona el repositorio y el dato entrado es válido en caso de seleccionar la opción "Número de la revisión", la aplicación muestra los datos de la revisión especificada sino muestra los datos de la última revisión. **Incorrecto:** Si no se selecciona el repositorio y el dato entrado no es válido o se deja el campo en blanco en caso de seleccionar la opción "Número de la revisión", se muestra el mensaje de error "Existen datos no válidos", le permite volver a intentarlo.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 34. Caso de prueba de aceptación: Listar datos de revisión**.**

Caso de prueba de aceptación	
Código: HU5-P5	**Número de HU:** 5
Nombre: Realizar copia segura de repositorios	
Descripción: La aplicación debe dar la oportunidad de realizar una copia segura de un repositorio.	
Condiciones de ejecución: La solicitud de realizar copia segura de repositorio.	
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción realizar copia segura de repositorio, selecciona un repositorio e introduce la ruta donde se guardará.	
Resultado esperado **Correcto:** Si se selecciona el repositorio y el dato entrado es válido, se realiza la copia del repositorio. **Incorrecto:** Si no se selecciona el repositorio y el dato entrado no es válido o se deja el campo en blanco, se muestra el mensaje de error "Existen datos no válidos", le permite volver a intentarlo.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 35. Caso de prueba de aceptación: Realizar copia segura de repositorio.

Caso de prueba de aceptación	
Código: HU6-P6	**Número de HU:** 6
Nombre: Recuperar repositorio	
Descripción: La aplicación debe dar la oportunidad de realizar tareas de recuperación sobre un repositorio después de un fallo del mismo.	
Condiciones de ejecución: La solicitud de recuperar repositorio.	
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción recuperar repositorio y selecciona un repositorio.	
Resultado esperado **Correcto:** Si se selecciona el repositorio, la aplicación muestra la última revisión del repositorio. **Incorrecto:** Si no se selecciona el repositorio se muestra el mensaje de error “Se debe seleccionar un repositorio”, le permite volver a intentarlo.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 36. Caso de prueba de aceptación: Recuperar repositorio**:**

Caso de prueba de aceptación	
Código: HU7-P7	**Número de HU:** 7
Nombre: Separar repositorios combinados	
Descripción: La aplicación debe dar la oportunidad de separar aquellos proyectos que se encuentran ubicados en un mismo repositorio.	
Condiciones de ejecución: La solicitud de recuperar un repositorio.	
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción separar repositorios combinados, selecciona el proyecto que desea separar e introduce nombre del nuevo repositorio donde se quiere guardar el proyecto.	
Resultado esperado **Correcto:** Si se selecciona el repositorio y el dato entrado es válido se copia el proyecto en un nuevo repositorio. **Incorrecto:** Si no se selecciona el proyecto y el dato entrado no es válido o se deja el campo en blanco se	

muestra el mensaje de error “Existen datos no válidos”, le permite volver a intentarlo.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 36. Caso de prueba de aceptación: Separar repositorios combinados.

Caso de prueba de aceptación	
Código: HU8-P8	**Número de HU:** 8
Nombre: Migrar repositorio.	
Descripción: La aplicación debe dar la oportunidad de migrar un repositorio a otro.	
Condiciones de ejecución: La solicitud de migrar un repositorio.	
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción migrar repositorio, selecciona el repositorio que desea migrar e introduce nombre del repositorio a donde se quiere migrar.	
Resultado esperado **Correcto:** Si se selecciona el repositorio y el dato entrado es válido se migra el repositorio a otro. **Incorrecto:** Si no se selecciona el repositorio y el dato entrado no es válido o se deja el campo en blanco se muestra el mensaje de error “Existen datos no válidos”, le permite volver a intentarlo.	
Evaluación de la prueba:	

Tabla 36. Caso de prueba de aceptación: Migrar repositorio.

Caso de prueba de aceptación	
Código: HU9-P9	**Número de HU:** 9
Nombre: Importar esquema a un repositorio.	
Descripción: La aplicación debe dar la oportunidad de agregar a un repositorio un conjunto de directorios que serán aplicados a las copias de trabajo que utilicen el repositorio.	
Condiciones de ejecución: La solicitud de importar esquema a un repositorio.	
Entrada/Pasos: El usuario accede a la interfaz inicial, escoge la opción importar esquema a un repositorio, selecciona el repositorio al que se quiere importar el esquema e introduce la ruta local de donde se encuentra el esquema.	
Resultado esperado	

Correcto: Si se selecciona el repositorio y el dato entrado es válido se importa un esquema al repositorio **Incorrecto:** Si no se selecciona el repositorio y el dato entrado no es válido o se deja el campo en blanco se muestra el mensaje de error “Existen datos no válidos”, le permite volver a intentarlo.
Evaluación de la prueba:

Tabla 36. Caso de prueba de aceptación: Importar esquema a un repositorio.

GLOSARIO DE TÉRMINOS

WebDAV (Web-based Distributed Authoring and Versioning por sus siglas en inglés): Es un servicio que nos permite compartir, editar y manejar archivos en un servidor remoto vía HTTP.
Marcos WxWidgets: Las wxWidgets son unas bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje

C++: Lenguaje de programación orientado a objeto.

Framework: Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

LDAP (en español Protocolo Ligero de Acceso a Directorios): Es un protocolo de acceso unificado a un conjunto de información sobre una red. Habitualmente, almacena la información de autenticación (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otra información (datos de contacto del usuario, ubicación de diversos recursos de la red, permisos, certificados, etc). A manera de síntesis.

Commits: Envía cambios desde su copia de trabajo local al repositorio.