Facultad # 4

Suite de Administración para Servidores de Entornos Productivos (SASEP). Módulo: SvnManager.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Leonardo Alexander Aymerich Reyes

Tutor:

Ing. Michel Miranda Cairo

La Habana 2012

"Año 54 de la Revolución"

Declaración de Autoría

Declaro que soy el único autor del presente trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) hacer el uso que estime pertinente con este trabajo.

Para que así conste firmo la presente a los ___ días del mes _____ del año _____.

Leonardo Alexander Aymerich Reyes

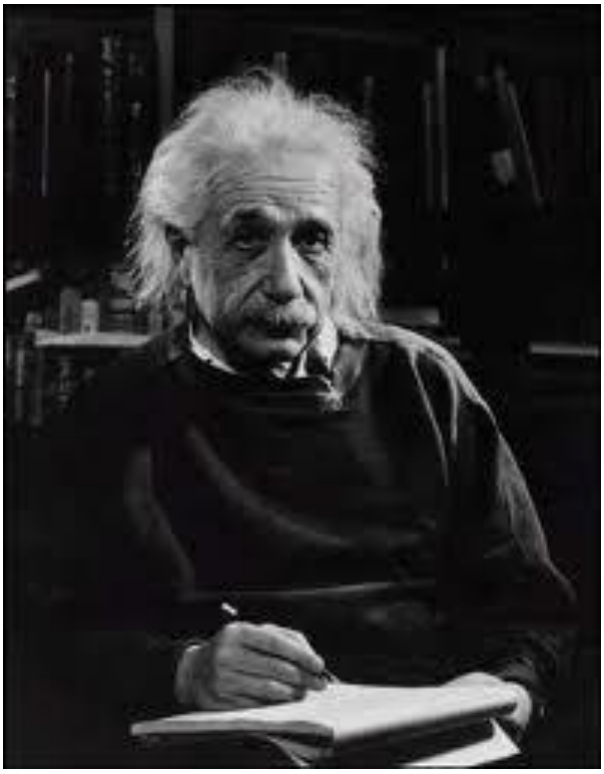
_____

Firma del Autor

Ing. Michel Miranda Cairo

_____

Firma del Tutor



"Si buscas resultados distintos, no hagas siempre lo mismo."

Albert Einstein.

Agradecimientos

A Dios, quien ha permitido que este día este disfrutando de este logro, pues ha sido mi luz y guía en los momentos difíciles de la vida.

A mi madre, no tengo palabras para expresar todo lo que mi corazón siente, has sido mi fuerza para levantarme y continuar cuando he caído, siempre has estado ahí para mi, sin cuestionar, sin flaquear, quitándote cada aliento de vida para dámelo a mí. Hoy te debo todo lo que soy madre y aunque sé que nunca podré pagar todo lo que has hecho por mí, siempre estarás en mi corazón y serás motivo de inspiración en cada paso que dé en la vida.

A mi abuelita, que aunque ya no esté entre nosotros, fue mi segunda madre, a ella también debo quien soy. Sé que estarías orgullosa de ver el hombre en que me convertido producto a la educación que me diste "abue".

A Carlos, eres como un padre para mí, he aprendido mucho de tu ejemplo, gracias por tu apoyo.

A mi familia, quienes desde pequeño me han sabido educar y guiar por buenos caminos, les debo mucho a todos ustedes. Mi tío Ramón, eres como un padre para mí. Mis tías: Tita, Baby, Neni. Mis primos que más que primos son como mis hermanos: Yandrita, Noni, Yuni, Angelita, Jose y Ale.

A mis amigos, dicen que quien tiene amigos tiene un tesoro. A todos mis amigos: Noldy, Edson, Leodys, el Flaco, Landy,

Josué, Charlie, Anett, Hassan, Yanetsy, Isied, los niches (Diego y Franklin), Erick, Sol, Jessy, Dianelys, Evelyn, Diosleidys, Dairelis. Y a todos los que han formado o forman parte de mi vida, los llevo en el corazón

A mis compañeros de la universidad, todos con los que he compartido cada pedacito importante que aquí he vivido: el aula, festival de artistas aficionados, eventos científicos, apartamento, fiestas, recre y todas las demás cosas que haya olvidado mencionar.

Al tribunal, gracias por sus consejos, por su guía y su respeto, en especial a Roberto y a mi tutor y amigo Michel.

A los profesores que durante estos cinco años han aportado un granito de arena a mi formación como profesional y persona. Muchos de ustedes a demás de profesores han sido mis amigos.

A todas y cada una de las personas que han hecho posible la realización de este trabajo, a todo el que en algún momento de la vida me tendió desinteresadamente una mano, me dio un consejo, me hizo una crítica constructiva.

Dedicatoria

A mi madre por haber confiado en mí y haberme apoyado incondicionalmente en toda mi carrera profesional y mi vida personal.

A mi familia, por haberme brindado todo su cariño y confianza, por sus consejos y la educación que me legaron.

A todos aquellos que de una forma u otra han hecho posible que este sueño se haya hecho realidad.

RESUMEN

Uno de los principales objetivos de las empresas de software, es lograr perfeccionar sus técnicas y procesos durante la implementación de sus productos y dentro de las tareas que contribuyen a ello se encuentran las relacionadas con la Gestión de Configuración, la cual engloba todo el proceso de acondicionar la plataforma de desarrollo de software en los entornos productivos.

Debido a las limitaciones que presentan los programas de administración para los servicios más comunes en los entornos productivos, entre ellos el servicio de control de versiones mediante Subversion y con el objetivo de proporcionar a los gestores de configuración de la UCI una herramienta que permita la administración del servidor Subversion de forma visual y remota, se decide desarrollar el módulo SvnManager integrable a la Suite de Administración para Servidores de Entornos Productivos (SASEP).

El presente trabajo incluye un estudio del estado del arte, las herramientas utilizadas, características del sistema, planificación, implementación y la estrategia de pruebas que se llevó a cabo. Además, un conjunto de conclusiones y recomendaciones que pueden ser de gran importancia para el tema abordado.

Palabras Claves: control de versiones, gestión de configuración, desarrollo de software, entornos productivos, servidor Subversion.

Índice de Contenidos

INTRODUCCIÓN

Los continuos avances de las Tecnologías de la Información y las Comunicaciones (TIC) han revolucionado el desarrollo científico en varias esferas de la vida, siendo los productos informáticos protagonistas en este proceso. Por tal motivo es comprensible que las organizaciones y empresas implicadas en el proceso de desarrollo de software, a diario enfoquen sus esfuerzos y recursos al mejoramiento de su desempeño. El proceso de desarrollo de software no es único, debido a que el mismo varía en dependencia a los contextos de proyectos de desarrollo en que sea aplicado.

Este proceso está compuesto por un conjunto de actividades fundamentales tales como: la Especificación de software, el Diseño e Implementación, la Gestión de Configuración, entre otras. Esta última es uno de los aspectos fundamentales que influye en la obtención de un producto final con calidad, conforme a que garantiza un entorno de desarrollo adecuado al implementar eficazmente un conjunto de servicios, entiéndase: servidores web, bases de datos, control de versiones, entre otros. El control de versiones es un servicio de suma importancia en los entornos productivos, debido a que brinda la facilidad de gestionar los cambios realizados sobre el material generado en un proyecto, durante todas sus etapas de desarrollo de forma eficiente y segura.

Existen tres tipos principales de servidores para el control de versiones:

1. Locales: Gestionan el proceso de control de versiones en la computadora local.
2. Centralizados: El proceso de control de versiones es realizado en un servidor remoto.
3. Distribuidos: Realizan el proceso de control de versiones en varios servidores sincronizados entre sí.

Garantizar el correcto funcionamiento de los servicios en los entornos productivos de software, suele resultar generalmente engorroso, producto a que se requiere de un profundo dominio sobre diversas temáticas tales como: sistemas operativos, protocolos de redes, conexiones remotas, configuraciones de seguridad, entre otras. Además la poca existencia de herramientas para el manejo de estos servicios y las limitaciones que las mismas presentan, obligan a los gestores de configuración a realizar su trabajo mediante el uso de líneas de comandos y la edición de archivos de configuración sensibles de forma manual, proceso que usualmente conlleva a la introducción de errores en las tareas de configuración, acarreando así una demora considerable en el resto del proceso de desarrollo de software.

Cuba no cuenta con una infraestructura productiva de software de gran impacto a nivel mundial, la producción se realiza de manera casi artesanal y es básicamente para consumo nacional, aunque en los últimos años con la creación de la Universidad de

las Ciencias Informáticas (UCI) dedicada, además de la docencia, a la producción de bienes y servicios informáticos, se han logrado importantes avances dentro del marco de los convenios del ALBA.

La UCI no está exenta a los problemas antes planteados que afectan el proceso de desarrollo de software, sumándole a estos el hecho de que en no pocas ocasiones el rol de gestor de configuración en entornos productivos es desempeñado por estudiantes o personas inexpertas.

La herramienta Suite de Administración para Servidores de Entornos Productivos (SASEP), surge con el objetivo de facilitar las tareas administrativas relacionadas con los servidores utilizados en los entornos productivos de la UCI. SASEP es una herramienta de integración modular compuesta por plugins, la cual provee mediante cada plugin funcionalidades para administrar un servidor de forma visual y remota. Actualmente SASEP no cuenta con un plugin para la administración del servidor Subversion.

Por lo anteriormente planteado se define el siguiente **problema científico:** ¿Cómo facilitar las tareas administrativas relacionadas con el servidor Subversion en los entornos productivos de la UCI?

A partir de lo antes expuesto se plantea como **objeto de estudio** el proceso de administración de servidores de control de versiones.

El **campo de acción** está enmarcado en las herramientas para la administración visual del servidor Subversion en los entornos productivos de la UCI.

Se plantea como **objetivo general** desarrollar un módulo integrable a SASEP para la administración visual del servidor Subversion.

Se plantea la siguiente **idea a defender:** con el desarrollo de un módulo integrable a SASEP que permita la administración visual del servidor Subversion, se logrará facilitar las tareas administrativas del proceso de control de versiones en los entornos productivos de la UCI.

Para dar cumplimiento al objetivo general antes expuesto se derivan los siguientes **objetivos específicos:**

1. Caracterizar las tendencias actuales de las herramientas para la administración visual del servidor Subversion.
2. Seleccionar metodología adecuada para la administración del servidor Subversion.
3. Desarrollar un módulo integrable a SASEP para la administración del servidor Subversion.

Para dar cumplimiento a los objetivos específicos planteados anteriormente se distribuyen las siguientes **tareas a cumplir:**

1. Investigación de tendencias actuales de las herramientas para la administración visual del servidor Subversion.
2. Descripción detallada del funcionamiento del servidor Subversion.
3. Identificación de características principales, buenas prácticas y políticas de seguridad para la administración del servidor Subversion.
4. Realización del análisis sobre Metodologías y Herramientas a utilizarse en la investigación.
5. Realización del proceso de análisis y diseño del módulo.
6. Análisis y definición de estándares de código y prácticas seguras de implementación.
7. Implementación de las interfaces de integración y las funcionalidades definidas.
8. Diseño de pruebas.
9. Realización de las pruebas.
10. Corrección de los errores detectados durante el proceso de prueba.

Para el desarrollo de la presente investigación se utilizaron algunos métodos tradicionales investigativos tanto teóricos como empíricos, los cuales son evidenciados a continuación:

1. Métodos Teóricos:

- **Histórico-lógico:**

Para el estudio crítico de trabajos que aborden el tema de herramientas para la administración visual de servidores Subversion tanto a nivel internacional, nacional y universitario. Se toman como puntos de referencia los resultados alcanzados con el desarrollo de los mismos, para la definición de los puntos de partida en el desarrollo de la presente investigación.

- **Analítico-sintético:**

Para el análisis de teorías, documentos y materiales relacionados con la administración del servidor Subversion, lo cual facilita profundizar en los conceptos que se utilizarán permitiendo un mejor entendimiento de lo que se quiere desarrollar.

2. Métodos Empíricos:

- **Entrevistas combinadas, abiertas y dirigidas:**

Para la obtención de información e ideas, se entrevistan a personas especializadas en el tema de gestión de la configuración, los mismos aportan diversos criterios para tomar decisiones en la realización del trabajo sobre aspectos referentes a las actividades que se deben realizar y los instrumentos que se utilizan en cada una de ellas.

La presente investigación está conformada por tres capítulos, los mismos quedan definidos de la siguiente manera:

Capítulo 1. “Fundamentación Teórica”, se estudian y analizan los principales conceptos relacionados con el objeto de estudio, los elementos teóricos que constituyen la base de la investigación tales como las diferentes aplicaciones para la administración visual del servidor Subversion que existen en la actualidad. Por último se estudian las tecnologías a utilizar para dar solución al problema planteado. Se justifican las opciones seleccionadas.

Capítulo 2. “Características del sistema”, este capítulo está enmarcado en las fases de exploración, planificación e iteraciones de la metodología de desarrollo XP. Se abordan temas relacionados con el funcionamiento del sistema, las personas que interactúan con este, las especificaciones funcionales del mismo, planificación de entrega y estimaciones de esfuerzo. Todo ello orientado a la conformación de una propuesta del sistema.

Capítulo 3. “Implementación y Pruebas”, se muestra detalladamente la evolución del sistema, a través del desarrollo de las fases de la metodología XP antes mencionadas. La implementación se realiza de forma iterativa e incremental, obteniéndose al final de cada iteración un producto funcional el cual es probado y mostrado al cliente. Todo ello permite lograr una constante retroalimentación desarrolladores-clientes, propiciando que los primeros puedan ampliar su visión del producto apoyados por la opinión de los segundos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Para tener una mejor visión de lo que se pretende lograr con la presente investigación, es necesario realizar un análisis de las tecnologías y conceptos relacionados con la administración del servicio de control de versiones Subversion.

El objetivo principal de este capítulo es brindar un marco conceptual adecuado para la comprensión del objeto de estudio y realizar el estado del arte referente a este tema.

La administración de servicios en los entornos productivos, es una de las tareas a realizar durante la gestión de la configuración en el proceso de desarrollo de software. Los servicios de control de versiones, entre los que se incluye Subversion, son de suma importancia debido a que permiten almacenar y versionar de forma eficiente, todo el material digital implicado en el proceso de desarrollo.

Producto al continuo y prolífero avance de las tecnologías, actualmente han aumentado en gran medida las aplicaciones informáticas, así como los fines con los que son desarrolladas y las funcionalidades que brindan. Entre estas se encuentran las herramientas para la administración visual de servidores, las que tienen como objetivo principal, facilitar la configuración de uno o varios servidores, brindando interfaces gráficas para la interacción de los administradores con estos.

1.2 Administración

Es un proceso muy particular consistente en las actividades de planificación, organización, ejecución y control, desempeñadas para determinar y alcanzar los objetivos señalados. Lleva consigo la responsabilidad de planear y regular en forma eficiente las operaciones implicadas para el logro de los propósitos dados. (1)

1.3 Servidor

Un servidor es un equipo que proporciona datos a otros equipos. Puede proveer de datos a los sistemas en una red de área local *(Local Area Network*) o una red de área amplia (*Wide Area Network*) a través de Internet. Existen muchos tipos de servidores, incluyendo servidores web, de correo electrónico, de archivos y otros. Cada tipo de servidor ejecuta software específico para el propósito del mismo.

Mientras que el software de servidor es específico para el tipo de servidor, el hardware no es tan importante. De hecho, algunas computadoras de escritorio normales pueden convertirse en servidores añadiendo los programas y configuraciones adecuadas. Por ejemplo, un ordenador conectado a una red doméstica puede ser designado como un servidor de archivos, servidor de impresión, o ambos. (2)

1.4 Sistemas de Control de Versiones

Los Sistemas de Control de Versiones por sus siglas en inglés (CVS), son sistemas encargados de registrar los cambios realizados sobre un archivo o conjunto de estos (imágenes, audio, código fuente y otros) a lo largo del tiempo. Estos permiten recuperar versiones específicas de los archivos versionados, así como comparar y revertir cambios a versiones anteriores.

Existen tres tipos de CVS: locales, centralizados y distribuidos. (3)

1.4.1 Subversion

Subversion (SVN) es un sistema de control de versiones centralizado, libre y de código fuente abierto. Este maneja ficheros y directorios a través del tiempo, creando un árbol en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios, lo que le permite recuperar versiones antiguas de sus datos o examinar el historial de cambios de los mismos. En este aspecto, muchas personas piensan en los sistemas de versiones como en una especie de “máquina del tiempo”.

Subversion puede acceder a los repositorios a través de redes, lo que le permite ser usado por personas que trabajan desde distintos ordenadores. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones, fomenta la colaboración y el trabajo en equipo. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada por la pérdida de este, si se ha hecho un cambio incorrecto a los datos, simplemente se deshace el mismo. (4)

1.5 Aplicación modular

Una aplicación modular, es una porción de código estrechamente acoplado, en la que cada unidad puede conectarse directamente con cualquier otra parte, se compone de pequeñas porciones de código que están bien aisladas y encapsuladas en módulos o plugins. Esas porciones pueden ser desarrolladas por equipos independientes con su propio ciclo de vida. Los resultados pueden ser montados entre sí por una entidad separada del distribuidor. (5)

1.5.1 Plugin

Programa de computación que se agrega al equipo o sistema, para que este pueda manejar cierto tipo de archivos o de información. (6)

1.6 Análisis de soluciones similares

La investigación realizada sobre las herramientas para la administración visual y remota del servidor Subversion, centró el estudio en tres de estas, por ser las de mayor aceptación entre los gestores de configuración, las mismas se abordan a continuación.

1.6.1 VisualSVN Server

VisualSVN Server permite instalar y gestionar fácilmente un servidor Subversion. Gracias a su robustez, facilidad de uso y sus características de nivel empresarial, el servidor VisualSVN Server es asequible tanto para las pequeñas empresas como para usuarios corporativos. Este se basa en estándares abiertos y ofrece una sólida estabilidad, seguridad y rendimiento.

Se distribuye como un único paquete de instalación con las últimas versiones de todos los componentes necesarios. El proceso de instalación es muy simple y le permite configurar un sistema completo y listo para utilizar el servidor Subversion en tan sólo unos pocos clics. **(Ver Anexo 1)**

VisualSVN Server aunque es reconocido como la forma más fácil de configurar un servidor de Subversion, es únicamente funcional en plataformas Windows. (7)

Sus principales características son:

1. *Active Directory Single Sign-On.*
2. Consola de gestión de gran alcance.
3. Administración del servidor remoto.
4. Acceso y registro de funcionamiento.

1.6.2 SVNManager

SVNManager es una herramienta web multiplataforma basada en PHP. La misma brinda funcionalidades, para administrar servidores Subversion configurados con el módulo DAV de Apache. Es una herramienta que permite realizar el trabajo de forma remota, pero debe existir una instancia de la misma en cada servidor SVN que se desee administrar. **(Ver Anexo 2)**

Entre las funcionalidades que brinda SVNManager se pueden mencionar:

1. Crear, borrar, cargar y descargar los repositorios.
2. Administrar cuentas de usuarios para el acceso a los repositorios.
3. Gestionar grupos de acceso a los repositorios.
4. Invitar a los usuarios por correo electrónico para crear una cuenta en el servidor.

La principal desventaja de esta herramienta se basa en que depende de dos Sistemas Gestores de Bases de Datos específicos: MySQL o SQLite. (8)

1.6.3 SVNControl

Es una herramienta de administración remota para Subversion desarrollada en Java.

SVNControl utiliza funciones para manipular de forma automática los permisos asignados a los usuarios y grupos en los repositorios y maneja ganchos de scripts con facilidad. **(Ver Anexo 3)**

Esta herramienta brinda funcionalidades como:

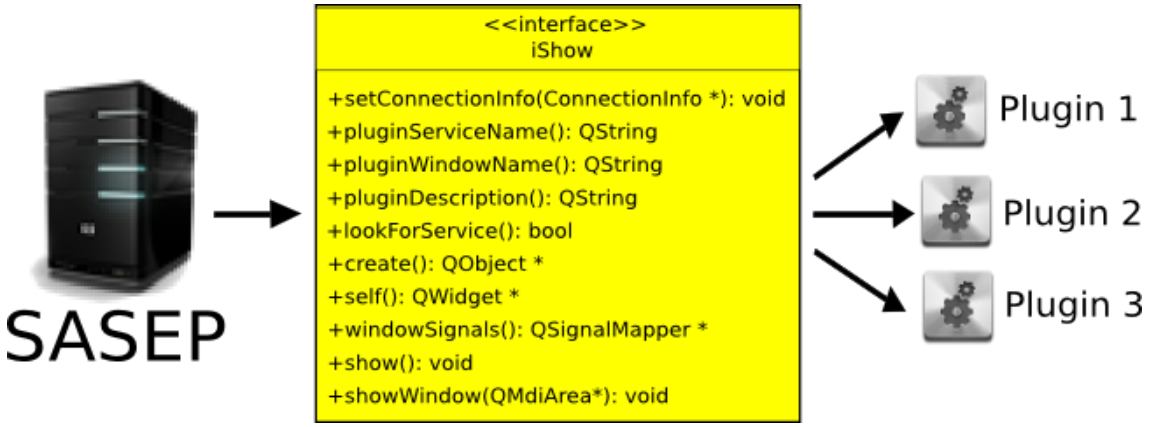
1. Administración remota a través de conexiones seguras SSL y no SSL.
2. Gestión de repositorios.
3. Gestión de usuarios (incluyendo LDAP, *Active Directory* y la autenticación de *Atlassian Multitud*).
4. Plugin (mecanismo para extender SVNControl con funcionalidades adicionales).
5. Sincronización de los archivos de acceso de Subversion entre varios servidores.

A pesar de la amplia y cómoda gama de funcionalidades que SVNControl ofrece, cuenta con grandes desventajas tales como: se debe encontrar Subversion 1.4.x, 1.5.x, o 1.6.x instalado, configurado y en funcionamiento en el servidor remoto. Se requiere instalar en las máquinas clientes y servidoras JRE versión 1.5.x como mínimo y que el JDK instalado en la máquina servidora sea compatible con el instalado en la cliente. (9)

1.7 Tendencias y Tecnologías actuales

SASEP es una herramienta que permite la administración remota y visual de servidores, desarrollada en el Departamento de Implantación y Soporte Técnico del Centro de Tecnologías para la Formación en la UCI. Está basada en una arquitectura modular extensible a través de plugins, los cuales posibilitan adicionarle nuevas funcionalidades para la administración de diferentes servicios.

Toda la comunicación de SASEP con el servidor es realizada usando el protocolo ssh, lo que garantiza una conexión con un nivel excelente de seguridad. Los plugins desarrollados para la herramienta deben implementar la interfaz que se muestra en la siguiente figura.



Al estar desarrollada en el lenguaje C++ y el framework Qt 4.7, es necesario para un correcto funcionamiento, que los plugins sean implementados siguiendo estas pautas. Por tal motivo en esta investigación no se analizan otras tecnologías, ejemplo: Java o .NET Framework.

1.7.1 Metodologías de Desarrollo

El desarrollo de software es un proceso sistemático, el cual debe garantizar la correcta integración y funcionamiento de los componentes que lo conforman. Una metodología de desarrollo de software puede definirse como un conjunto de procedimientos a seguir, orientados a obtener la calidad y los resultados esperados. Por ello la selección de la misma es de suma importancia en el proceso productivo de software. Actualmente existe dos tipos de metodologías: Las Tradicionales o Pesadas y las Ágiles o Livianas. (10) Por las características de la presente investigación, se centra el estudio en las metodologías ágiles.

A continuación se describen las características más relevantes de las posibles metodologías ágiles a aplicar, con el objetivo de seleccionar la idónea para llevar a cabo el desarrollo de la presente investigación.

1.7.1.1 SCRUM

SCRUM es una metodología ágil de gestión de proyectos, cuyo principal objetivo es elevar la productividad de un equipo al máximo. Esta reduce las actividades no orientadas a la producción de software y por medio de iteraciones o Sprints, genera resultados en breves periodos de tiempo. Dicha metodología es considerada ideal para el desarrollo de proyectos dónde los requerimientos varían rápidamente. Delega en el equipo de desarrollo, la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivos posibles.

A pesar de las facilidades antes expuestas, esta metodología no genera toda la evidencia o documentación que se desean y sólo abarca prácticas de gestión, por lo que no contempla las prácticas de desarrollo como lo hace la metodología XP, por ello en ocasiones se recomienda integrarla con esta última. (11)

1.7.1.2 Proceso Unificado Ágil

Proceso Unificado Ágil por sus siglas en inglés (AUP), fue desarrollada por Scott Ambler, es una versión simplificada de RUP. Esta metodología describe de manera simple la forma de desarrollar aplicaciones de software mediante el uso de técnicas ágiles tales como: el Desarrollo Dirigido por Pruebas (*Test Driven Development*), Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de Base de Datos para mejorar la productividad.

AUP tiene como objetivos: Identificar el alcance inicial del proyecto, proveer una arquitectura potencial para el sistema, y obtener un financiamiento inicial del proyecto y la aceptación de los implicados. A pesar de ello esta metodología es muy pesada debido a que su ciclo de vida es bastante abarcador, por ello es que la primera versión del producto se entrega aproximadamente a los 12 meses, la segunda a los 9 y las demás semestrales. (12)

1.7.1.3 Programación Extrema

La Programación Extrema por sus siglas en inglés (XP), es una metodología ágil de desarrollo, la cual nace como nueva disciplina de desarrollo de software hace aproximadamente unos quince años, formulada por Kent Beck y Jean. Como principal objetivo, XP define la satisfacción del usuario, tratando de dar a este, el software que necesita y cuando lo necesita, objeta además potenciar al máximo el trabajo en equipo.

Esta metodología brinda una amplia gama de facilidades tales como: desarrollo ágil y organizado de software, el cliente tiene el control sobre las prioridades; las actividades improductivas han sido eliminadas para reducir costos y frustraciones. A pesar de ello es recomendable emplearla solo en proyectos a corto plazo y es difícil predecir costo y tiempo de desarrollo. (12)

1.7.1.4 Fundamentación de la Metodología de Desarrollo a utilizar

Después de un profundo análisis sobre las metodologías de desarrollo de software anteriormente expuestas, se decidió utilizar XP debido a que la misma se adapta en gran medida al tipo de producto a desarrollar así como a las condiciones del equipo de trabajo. Se exponen a continuación razones concretas que conllevaron a la toma de esta decisión:

1. El producto de la investigación realizada es pequeño y XP está concebida para ser aplicada a estos tipos de proyectos.
2. El cliente y el equipo de desarrollo laboran en conjunto: mediante la aplicación de XP se logra una mayor retroalimentación y un producto que satisface las necesidades del cliente.
3. Permitirá definir en cada iteración cuáles son los objetivos de la siguiente.
4. Corrección de los errores antes de añadir nueva funcionalidades. Hacer entregas

frecuentes.

5. Refactorización del código (reescribir ciertas partes del código para aumentar su legibilidad sin modificar su comportamiento).
6. Simplicidad en el código.

1.7.2 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado por sus siglas en inglés (IDE), es un ambiente para escribir la lógica de aplicación y el diseño de interfaces de aplicaciones. Se puede caracterizar como el programa informático compuesto de un conjunto de herramientas, que utilizan los programadores para generar código. Estos pueden ser multiplataformas, soportar diversos lenguajes de programación, tener un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de forma amigable. (13)

Entre los diversos IDE que existen para el lenguaje seleccionado (C++), se estudiaron los tres más utilizados en plataformas libres.

1.7.2.1 Eclipse + Qt Plugin

Muchas personas piensan en Eclipse como sólo un IDE para Java, pero Eclipse es una plataforma basada en complementos para la creación de programas informáticos. A menudo se incluye con las herramientas de desarrollo Java, por lo que ofrece un IDE en esos casos.

El soporte para lenguajes de programación en Eclipse se realiza a través de plugins. Existen plugins para la mayoría de los lenguajes. El estándar de C++ Plugin para Eclipse se llama CDT (*C/C++ Development Tooling*). Contempla depuración GDB, gestionado de makefiles, refactorización sencilla y mucho más.

Eclipse integra a la perfección todo el flujo de trabajo de desarrollo de Qt:

1. Gestión de proyectos basado en qmake (no se imponen formatos de archivos específicos de Eclipse) con un importador de archivos pro y un editor gráfico de proyectos.
2. Incluido diseñador de interfaces de usuario y editor de acciones.
3. Asistentes para la creación de nuevas clases y proyectos de Qt.
4. Incluida referencia de Qt, un tutorial Qt Eclipse y una hoja de trucos. (14)**(Ver Anexo 4)**

1.7.2.2 Code Blocks

Code Blocks es un IDE libre de C++ diseñado para satisfacer las necesidades más exigentes de sus usuarios. Es muy extensible y totalmente configurable. Code blocks brinda una amplia gama de características, presenta un aspecto coherente y es

multiplataformas. Además es construido basado en un marco de plugins, por lo que se le pueden adicionar cualquier tipo de funcionalidades mediante la instalación o configuración de un plugin. Por ejemplo las funcionalidades de compilación y depuración están disponibles a través de estos. (15) **(Ver Anexo 5)**

Entre otras características Code Blocks posee:

1. Licencia GPL v3.
2. Escrito en C++ y no incluye librerías propietarias.
3. Múltiple soporte para compiladores.
4. Sistema de construcción muy rápido (sin necesidad de makefiles).
5. Plegado de código de C++ y archivos XML.

1.7.2.3 Qt Creator

Qt Creator es un entorno de desarrollo integrado multiplataforma, adaptado a las necesidades de los desarrolladores de Qt. **(Ver Anexo 6)**

Este IDE provee lo siguiente:

1. Editores de código de C++ y JavaScript.
2. Diseñador de interfaz de usuario integrado.
3. Proyecto y construcción de herramientas de gestión gdb y depuradores CDB.
4. Soporte para el control de versiones.
5. Simulador de interfaces de usuarios móviles.
6. Apoyo a dispositivos de sobremesa y portátiles.

Qt Creator se ejecuta en varios Sistemas Operativos, entiéndase: Windows, Linux/X11 y Mac OS X, y permite a los desarrolladores crear aplicaciones para escritorio y plataformas de dispositivos móviles. (16)

1.7.2.4 Fundamentación del IDE a utilizar

Después de un profundo análisis se decide utilizar el IDE de programación Qt Creator por las razones expuestas a continuación:

1. **Editor de código sofisticado:** El editor avanzado de código de Qt Creator ofrece soporte para la edición de C++ y QML (JavaScript), ayuda sensible al contexto, completamiento de código y navegación.
2. **Control de versiones:** Qt Creator se integra con la mayoría de los más populares sistemas de control de versiones, entiéndase: Git, Subversion, Bazaar, Perforce, CVS y Mercurial.
3. **Diseñadores de interfaz de usuario integrada:** Qt Creator proporciona dos editores visuales integrados: Qt Designer para la creación de interfaces de usuario de widgets de Qt y Qt Designer para el desarrollo de interfaces de usuario rápido de animación con el lenguaje QML.

4. **Gestión de Proyecto y Construcción:** Si se importa un proyecto existente o se crea uno nuevo, Qt Creator genera todos los archivos necesarios. Este brinda apoyo a Cmake y la compilación cruzada con qmake está incluida.
5. **Escritorio y dispositivos móviles:** Qt Creator ofrece soporte para crear y ejecutar aplicaciones Qt para equipos de sobremesa y dispositivos móviles.
6. **Simulador de Qt:** Disponible como parte del SDK de Qt, el Simulador de Qt para poner a prueba la aplicación Qt para dispositivos móviles en un entorno similar a la del dispositivo de destino.

1.8 Conclusiones

Una vez realizado el estudio del estado del arte y habiéndose expuesto las características y deficiencias de las aplicaciones existentes, se propone el desarrollo de un módulo integrable a la herramienta SASEP para la administración visual del servidor Subversion. Este módulo brindará funcionalidades de administración y configuración para el trabajo con el servidor antes mencionado, lo cual influirá positivamente en la labor de los gestores de configuración en los entornos productivos de la UCI.

Para llevar a cabo el desarrollo del módulo se propone emplear la metodología XP, la cual permitirá una implementación ágil y organizada del producto y garantizará la culminación del mismo en el menor plazo de tiempo. El IDE a utilizar será Qt Creator el cual brinda numerosas funcionalidades y viene integrado en un SDK con el Framework y el lenguaje de programación que se usarán.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

El capítulo actual tiene como objetivo hacer una valoración de las características principales del sistema a desarrollar. En el mismo se abarcan las tres primeras fases fundamentales de la metodología XP (Exploración, Planificación e Iteraciones). Se plantea además la propuesta de solución al problema principal de la presente investigación, mediante la descripción de un módulo integrable a la aplicación SASEP para la administración del servidor Subversion.

2.2 Descripción de los procesos vinculados al campo de acción

Actualmente el proceso de administración del servidor Subversion en los entornos productivos de la universidad contempla una serie de acciones no estandarizadas. Este proceso es realizado de diversas formas, entiéndase de manera manual (mediante el uso de comandos de consola) o a través del empleo de distintas herramientas con interfaz visual. Muchas de estas herramientas no abarcan el proceso de configuración en su totalidad y otras no son integrables entre sí o presentan serias dependencias de plataformas y/o tecnologías específicas.

2.3 Propuesta del sistema

Luego del análisis del proceso de configuración del servidor Subversion en los entornos productivos de la universidad, se propone la implementación de un módulo integrable a la aplicación SASEP, que provea funcionalidades orientadas a facilitar el trabajo de administración del servidor antes mencionado.

Este módulo brindará funcionalidades estándares (entiéndase trabajo con repositorios, grupos y usuarios) y avanzadas (copias de seguridad del servidor, configuración de conexión LDAP) para el trabajo con el servidor Subversion. En cuanto al trabajo con los repositorios, los administradores podrán crearlos, eliminarlos y listarlos. El trabajo con los grupos permitirá crear, eliminar y listar grupos; estos pertenecen a repositorios específicos y por lo general se crean según los roles del proyecto en cuestión. Además se brindarán funcionalidades de crear, eliminar y listar usuarios; estos son asignados a los grupos antes mencionados. Al crear un usuario se puede especificar si es nuevo, existente o perteneciente a una configuración LDAP.

De acuerdo a las funcionalidades avanzadas, el administrador podrá realizar copias de seguridad del servidor, estas no serán almacenadas localmente como medida preventiva ante algún incidente de seguridad informática. Por su parte la configuración de acceso utilizando LDAP permitirá la autenticación de los usuarios en el servidor mediante el uso de este protocolo.

Siendo esta última la más óptima por su estandarización e integración con los demás sistemas de la universidad.

2.3.1 Personas relacionadas con el sistema

Se define como persona relacionada con el sistema toda aquella que está vinculada al desarrollo o el uso del mismo.

Personas relacionadas con el sistema	Justificación
Desarrollador	Es la persona que implementará el sistema.
Gestor de configuración (Administrador)	Es la persona que utilizará el sistema para llevar a cabo la configuración del servidor Subversion.

Tabla 1 Personas relacionadas con el sistema.

2.4 Fase de Exploración

En esta fase se define una visión del alcance general del proyecto. En la misma el cliente plantea lo que necesita a través de sencillas historias de usuarios y los programadores estiman el tiempo de desarrollo en base a esta información. Vale aclarar que estas estimaciones son primarias y podrían variar en cada iteración.

2.4.1 Historias de Usuarios

Las Historias de Usuarios (HU) son los artefactos utilizados en XP para especificar los requisitos funcionales del software. Su tratamiento es muy dinámico y flexible, en cualquier momento estas pueden romperse, reemplazarse por otras más específicas o generales, añadir nuevas o modificarlas. Cada HU debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en pocas semanas.

Debido a que existen varias plantillas sugeridas para definir la información contenida en las historias de usuario, pero no existe un consenso al respecto, para la presente investigación se define la siguiente estructura:

- **Nombre**: Nombre descriptivo de la Historia de Usuario.
- **Prioridad**: Grado de prioridad (Alta, Media o Baja) asignado por el equipo de desarrollo a la HU en dependencia de las necesidades del cliente.
- **Complejidad**: Grado de complejidad (Alta, Media o Baja) asignado por el equipo de desarrollo a la HU luego de analizarla.
- **Iteración**: Número de la iteración en la cual será implementada la HU.
- **Descripción**: Descripción simple que brinda el cliente sobre lo que debe hacer la funcionalidad en cuestión.

- **Información Adicional:** Una breve información que ayude a comprender algún dato no especificado antes.

Historia de Usuario					**Número**	1
Nombre	Gestionar repositorios					
Prioridad	Alta	**Complejidad**	Media	**Iteración**	1	
Descripción						
Se encarga de crear, eliminar y listar los repositorios del servidor Subversion.						
Información Adicional						

Tabla 2: Descripción de la historia de usuario: Gestionar repositorios.

Historia de Usuario					**Número**	2
Nombre	Gestionar grupos					
Prioridad	Alta	**Complejidad**:	Alta	**Iteración**	1	
Descripción						
Se encarga de crear, eliminar y listar los grupos pertenecientes a un repositorio específico.						
Información Adicional						

Tabla 3: Descripción de la historia de usuario: Gestionar grupos.

Historia de Usuario					**Número**	3
Nombre	Configurar permisos de grupo					
Prioridad	Media	**Complejidad**:	Media	**Iteración**	1	
Descripción						
Permite configurar los permisos de acceso de un grupo determinado a un repositorio específico.						

Información Adicional

Tabla 4: Descripción de la historia de usuario: Configurar permisos de grupo.

Historia de Usuario					**Número**	4
Nombre	Gestionar usuarios.					
Prioridad	Alta	**Complejidad**	Alta	**Iteración**	2	
Descripción						
Se encarga de crear, eliminar y listar los usuarios asignados a un grupo.						
Información Adicional						

Tabla 5: Descripción de la historia de usuario: Gestionar usuarios.

Historia de Usuario					**Número**	5
Nombre	Detectar servicio remoto					
Prioridad	Media	**Complejidad**	Media	**Iteración**	2	
Descripción						
Detecta automáticamente si el servidor Subversion está instalado en la máquina servidora.						
Información Adicional						

Tabla 6: Descripción de la historia de usuario: Detectar servicio remoto.

Historia de Usuario					**Número**	6
Nombre	Configurar tipo de autenticación					
Prioridad	Alta	**Complejidad**	Media	**Iteración**	3	
Descripción						
Se encarga de configurar la forma en que los usuarios acceden a los repositorios.						

Información Adicional

Tabla 7: Descripción de la historia de usuario: Configurar tipo de autenticación.

Historia de Usuario					**Número**	7
Nombre	Realizar copia segura de repositorios.					
Prioridad	Media	**Complejidad**	Alta	**Iteración**	3	
Descripción						
Se encarga de realizar y restaurar una copia exacta y segura de un repositorio.						
Información Adicional						
La copia no se guardará localmente en el servidor para evitar pérdida de la misma en caso de algún incidente de seguridad informática.						

Tabla 8: Descripción de la historia de usuario: Realizar copia segura de repositorios.

2.5 Planificación de la entrega

En esta fase el cliente establece la prioridad de cada historia de usuario de acuerdo a sus necesidades. A partir de estas prioridades, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega no debe demorar más de tres meses.

2.5.1 Estimación de esfuerzo por historia de usuario

Las historias de usuarios deben ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias, si es menos de una semana, se debe combinar con otra historia. Para ello se decide realizar la estimación de esfuerzo que arroja cada historia de usuario con el objetivo de obtener un mejor desarrollo del sistema.

No.	**Historia de usuario**	**Puntos de estimación (en semanas)**
1	Gestionar repositorios	1
2	Gestionar grupos	2
3	Configurar permisos de	1

	grupo	
4	Gestionar usuarios	2
5	Detectar servicio remoto	1
6	Configurar tipo de autenticación	1
7	Realizar copia segura de repositorios	1

Tabla 9 Estimación de esfuerzo por historia de usuario.

2.6 Planificación de las Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se establece una arquitectura base del sistema. Esto se logra escogiendo las HU que fuercen la creación de la misma. Al final de la última iteración el sistema estará listo.

Iteración	**Historia de usuario**	**Duración (semanas)**
1	Gestionar repositorios	4
	Gestionar grupos	
	Configurar permisos de grupos	
2	Gestionar usuarios	3
	Detectar servicio remoto	
3	Configurar tipo de autenticación	2
	Realizar copia segura de repositorios	

Tabla 10 Plan de Iteraciones

2.7 Conclusiones

En el presente capítulo se creó la propuesta del sistema. Se determinaron las personas relacionadas con la aplicación. También se presentaron las historias de usuarios y se definió su distribución en cada iteración, así como la duración de las mismas medidas en semanas. A partir de este punto, se puede comenzar con el desarrollo de la solución propuesta.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

La Metodología XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo con la culminación de cada iteración un producto funcional el cual debe ser probado y mostrado al cliente para con la opinión del mismo, lograr incrementar la visión de los desarrolladores. En el presente capítulo se detallan las tres iteraciones abarcadas durante la etapa de construcción del sistema, se exponen además las tareas generadas por cada historia de usuario, así como las pruebas de aceptación efectuadas sobre el sistema.

3.2 Diseño del sistema

Para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases UML, en su lugar se usan técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración) (17). Estas constituyen una técnica para el diseño de software orientado a objetos creada por Kent Beck y Ward Cunningham, se caracterizan por ser sencillas de utilizar, de entender y permitir al equipo de trabajo en su totalidad participar en el diseño del sistema. No obstante el uso de diagramas UML puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación, no constituya un peso su mantenimiento, no sean extensos y se enfoquen en la información importante.

Con el objetivo de hacer entendible las funcionalidades brindadas por el módulo para configurar el servidor Subversion, se define una tarjeta CRC por cada una, con la finalidad de obtener un diseño simple y no incurrir en la implementación de características no necesarias.

TARJETA CRC	
Clase: GestorConfigSvn	
SúperClase(s): QWidget**,** iShow	
Sub Clase(s):-	
Responsabilidades: Genera la intefaz gráfica con la que interactúa el usuario.	**Colaboraciones**: QListWidget, ConnectionInfo, QProcess, RepositoryManager,

	GroupManager, UserManager, frmCreateRepository, frmAddUser, QFileDialog, frmConfigConectionType, QMessageBox, QString, QStringList

Tabla 11Tarjeta CRC Clase GestorConfigSvn

TARJETA CRC	
Clase: Manager	
Súper Clase(s): QObject	
Sub Clase(s): RepositoryManager**,** GroupManager**,** UserManager	
Responsabilidades: Define y brinda funcionalidades genéricas de administración que son utilizadas por otras clases. • Ejecutar comando. • Leer salida de consola. • Reiniciar servidor.	**Colaboraciones**: QProcess, QString, QStringList

Tabla 12Tarjeta CRC Clase Manager

TARJETA CRC	
Clase: RepositoryManager	
Súper Clase(s): Manager	
Sub Clase(s): -	
Responsabilidades: Brinda funcionalidades para la administración de los repositorios. • Crear repositorio.	**Colaboraciones**: QProcess, QString, QStringList, UserManager,

• Eliminar repositorio. • Realizar copia de seguridad de un repositorio. • Configurar tipo de autenticación de un repositorio.	GroupManager.

Tabla 13Tarjeta CRC Clase RepositoryManager

TARJETA CRC	
Clase: GroupManager	
Súper Clase(s): Manager	
Sub Clase(s): -	
Responsabilidades: Brinda funcionalidades para la administración de los grupos pertenecientes a un repositorio. • Crear un grupo. • Eliminar un grupo. • Adicionar usuarios. • Modificar permisos de acceso de un grupo.	**Colaboraciones**: QProcess, QString, QStringList, UserManager

Tabla 14Tarjeta CRC Clase GroupManager

TARJETA CRC	
Clase: UserManager	
Súper Clase(s): Manager	
Sub Clase(s): -	
Responsabilidades: Brinda funcionalidades para la administración de los usuarios pertenecientes a un grupo.	**Colaboraciones**: QProcess, QString, QStringList, RepositoryManager

• Eliminar un usuario. • Modificar permisos de un usuario.	

Tabla 15Tarjeta CRC Clase UserManager

3.3 Fase de Implementación

Con las iteraciones, se realiza la implementación de las historias de usuarios seleccionadas para ser realizadas en cada una de estas. Al principio de cada iteración se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de ser necesario. Como parte de este plan, se descomponen las historias de usuarios en tareas de desarrollo, asignando a un grupo de desarrollo o una persona, la responsabilidad de su implementación. Estas tareas son para el uso estricto de los programadores, las mismas pueden ser escritas en lenguaje técnico y no necesariamente entendible por el cliente.

El presente trabajo llevó a cabo tres iteraciones de desarrollo sobre el sistema, lo que permitió obtener un producto con todas las características y restricciones deseadas. A continuación se detallan cada una de las iteraciones.

3.3.1 Iteración 1

En esta iteración se implementarán las historias de usuario 1, 2 y 3, con el fin de obtener una versión del producto con algunas de sus funcionalidades y mostrar el resultado al cliente para tomar nuevas iniciativas y decisiones conforme al punto de vista de este.

3.3.1.1Tareas por historias de usuarios definidas en la iteración 1

Tarea	
Número de tarea: 1	**Número de HU:** 1
Nombre de la tarea: Crear repositorio.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 12 de marzo del 2012	**Fecha fin:** 14 de marzo del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Se creará un repositorio con el nombre que el usuario determine para el mismo, se verificará que el repositorio no exista.	

Tabla 16 Tarea Crear repositorio.

Tarea	
Número de tarea: 2	**Número de HU:** 1
Nombre de la tarea: Eliminar repositorio.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 15 de marzo del 2012	**Fecha fin:** 17 de marzo del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Se eliminará un repositorio seleccionado por el usuario.	

Tabla 17 Tarea Eliminar repositorio.

Tarea	
Número de tarea: 1	**Número de HU:** 2
Nombre de la tarea: Crear grupo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.7
Fecha inicio: 19 de marzo del 2012	**Fecha fin:** 22 de marzo del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Creará un grupo con el nombre que el usuario determine, este grupo se asociará al repositorio seleccionado, se verificará que el grupo no exista.	

Tabla 18 Tarea Crear grupo.

Tarea	
Número de tarea: 2	**Número de HU:** 2
Nombre de la tarea: Listar grupos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha inicio: 23 de marzo del 2012	**Fecha fin:** 24 de marzo del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Listará los grupos pertenecientes a un repositorio al seleccionarlo.	

Tabla 19 Tarea Listar grupos.

Tarea	
Número de tarea: 3	**Número de HU:** 2
Nombre de la tarea: Eliminar grupo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.83
Fecha inicio: 26 de marzo del 2012	**Fecha fin:** 30 de marzo del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Eliminará un grupo seleccionado del repositorio actual.	

Tabla 20 Tarea Eliminar grupo.

Tarea	
Número de tarea: 1	**Número de HU:** 3
Nombre de la tarea: Configurar permisos de grupo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 2 de abril del 2012	**Fecha fin:** 6 de abril del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Configurará los permisos con que el grupo seleccionado accede al repositorio que pertenece. Los permisos son: lectura (r) y lectura/escritura (rw).	

Tabla 21 Tarea Configurar permisos de grupo.

3.3.2 Iteración 2

En esta iteración se implementarán las historias de usuario 4 y 5, con el fin de obtener una segunda versión del producto con nuevas funcionalidades. La conclusión de esta iteración influirá positivamente en la satisfacción del cliente debido a que el producto estará concluido en más de un 50%.

3.3.2.1Tareas por historias de usuarios definidas en la iteración 2

Tarea	
Número de tarea: 1	**Número de HU:** 4
Nombre de la tarea: Adicionar usuario.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 0.83
Fecha inicio: 9 de abril del 2012	**Fecha fin:** 13 de abril del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Adicionará un usuario al grupo seleccionado. Se pueden adicionar 3 tipos de usuarios: nuevo, LDAP, existente en el servidor.	

Tabla 22 Tarea Adicionar usuario.

Tarea	
Número de tarea: 2	**Número de HU:** 4
Nombre de la tarea: Eliminar usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.7
Fecha inicio: 23 de abril del 2012	**Fecha fin:** 26 de abril del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Permitirá eliminar un usuario de un grupo seleccionado.	

Tabla 23 Tarea Eliminar usuario.

Tarea	
Número de tarea: 3	**Número de HU:** 4
Nombre de la tarea: Listar usuarios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha inicio: 27 de abril del 2012	**Fecha fin:** 28 de abril del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Listará los usuarios pertenecientes a un grupo seleccionado.	

Tabla 24 Tarea Listar usuarios.

Tarea	
Número de tarea: 1	**Número de HU:** 5
Nombre de la tarea: Detectar servicio remoto.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 30 de abril del 2012	**Fecha fin:** 5 de mayo del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Detectará si el servidor Subversion está ejecutándose en la máquina remota.	

Tabla 25 Tarea Detectar servicio remoto.

3.3.3 Iteración 3

En esta iteración se implementarán las historias de usuario 6 y 7, con el fin de obtener la tercera y última versión del producto con todas las funcionalidades definidas por el usuario.

3.3.3.1 Tareas por historias de usuarios definidas en la iteración 3

Tarea	
Número de tarea: 1	**Número de HU:** 6
Nombre de la tarea: Configurar tipo de autenticación.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 7 de mayo del 2012	**Fecha fin:** 12 de mayo del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Permitirá activar o desactivar el acceso al servidor mediante el uso del protocolo LDAP.	

Tabla 26 Tarea Configurar tipo de autenticación.

Tarea	
Número de tarea: 1	**Número de HU:** 7
Nombre de la tarea: Realizar copia de seguridad de repositorios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.83
Fecha inicio: 14 de mayo del 2012	**Fecha fin:** 18 de mayo del 2012
Programador responsable: Leonardo Alexander Aymerich Reyes.	
Descripción: Permitirá realizar copias de un repositorio seleccionado con el objetivo	

de migrar el mismo o restablecerlo ante un incidente de seguridad informática.

Tabla 27 Tarea Realizar copia segura de repositorios.

3.4 PRUEBAS

Uno de los pilares fundamentales de XP es el proceso de pruebas, el mismo estimula a los desarrolladores a realizar pruebas constantes en la medida que sea posible. Mediante esta filosofía se reduce el número de errores no detectados así como el tiempo entre la introducción de estos en el sistema y su detección. Todo ello contribuye a elevar la calidad de los productos desarrollados y la certeza de los programadores al introducir cambios o modificaciones. (18)

La metodología XP divide las pruebas en dos grupos:

Pruebas unitarias: Realizadas por los desarrolladores para verificar el código de forma automática.

Pruebas de aceptación: Realizadas para evaluar si al final de una iteración se obtuvo la funcionalidad requerida y si esta es la esperada por el cliente.

3.4.1 Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra que se definen a partir de las historias de usuarios. Durante las respectivas iteraciones, las HU seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican la perspectiva del cliente y los escenarios para probar que han sido implementadas correctamente. Una HU puede definir todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de éstas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable. Una HU no se considera completa hasta que no ha pasado por sus pruebas de aceptación.

3.4.1.1 Pruebas de aceptación definidas para la iteración 1

Caso de Prueba de Aceptación	
Código: HU1_P1	**Número de HU:** 1
Nombre: Crear repositorio.	
Descripción: Prueba para la funcionalidad crear repositorio.	
Condiciones de ejecución: Debe estar el servidor Subversion instalado	
Pasos de ejecución: Se intenta crear un repositorio con un nombre válido.	

Resultado esperado: El repositorio es creado sin generar error.
Evaluación: Prueba satisfactoria.

Tabla 28 Prueba de aceptación crear repositorio.

Caso de Prueba de Aceptación	
Código: HU1_P2	**Número de HU:** 1
Nombre: Eliminar repositorio.	
Descripción: Prueba para la funcionalidad eliminar repositorio.	
Condiciones de ejecución: Deben existir repositorios en el servidor y haber seleccionado uno.	
Pasos de ejecución: Se intenta eliminar un repositorio.	
Resultado esperado: El repositorio es eliminado correctamente.	
Evaluación: Prueba insatisfactoria.	

Tabla 29 Prueba de aceptación eliminar repositorio.

Caso de Prueba de Aceptación	
Código: HU1_P3	**Número de HU:** 1
Nombre: Listar repositorios	
Descripción: Prueba para la funcionalidad listar repositorios.	
Condiciones de ejecución: Deben existir repositorios en el servidor.	
Pasos de ejecución: Se intentan listar los repositorios que existen el servidor al conectarse a este.	
Resultado esperado: Los repositorios son listados exitosamente.	
Evaluación: Prueba satisfactoria.	

Tabla 30 Prueba de aceptación listar repositorios.

Caso de Prueba de Aceptación	
Código: HU2_P1	**Número de HU:** 2

Nombre: Crear grupo
Descripción: Prueba para la funcionalidad crear grupo.
Condiciones de ejecución: Se debe seleccionar un repositorio al que se le asignará el grupo a crear.
Pasos de ejecución: Se intenta crear un grupo en el repositorio seleccionado.
Resultado esperado: El grupo es creado sin generar errores.
Evaluación: Prueba satisfactoria.

Tabla 31 Prueba de aceptación crear grupo.

Caso de Prueba de Aceptación	
Código: HU2_P2	**Número de HU:** 2
Nombre: Eliminar grupo.	
Descripción: Prueba para la funcionalidad eliminar grupo.	
Condiciones de ejecución: Se debe seleccionar el grupo que se desea eliminar.	
Pasos de ejecución: Se intenta eliminar un grupo seleccionado.	
Resultado esperado: Se elimina el grupo sin generar errores.	
Evaluación: Prueba satisfactoria.	

Tabla 32 Prueba de aceptación eliminar grupo.

Caso de Prueba de Aceptación	
Código: HU2_P3	**Número de HU:** 2
Nombre: Listar grupos.	
Descripción: Prueba para la funcionalidad listar grupos.	
Condiciones de ejecución: Se debe seleccionar un repositorio para listar los grupos que pertenecen al mismo.	
Pasos de ejecución: Se intenta listar los grupos que pertenecen al repositorio seleccionado.	
Resultado esperado: Se listan los grupos del repositorio seleccionado	

satisfactoriamente.
Evaluación: Prueba satisfactoria.

Tabla 33 Prueba de aceptación listar grupos.

Caso de Prueba de Aceptación	
Código: HU3_P1	**Número de HU:** 3
Nombre: Configurar permisos de grupo.	
Descripción: Prueba para la funcionalidad configurar permisos de grupo.	
Condiciones de ejecución: Se debe seleccionar un grupo para configurar sus permisos.	
Pasos de ejecución: Se intenta configurar los permisos del grupo seleccionado.	
Resultado esperado: Los permisos se configuran sin generar errores.	
Evaluación: Prueba satisfactoria.	

Tabla 34 Prueba de aceptación configurar permisos de grupo.

3.4.1.2 Pruebas de aceptación definidas para la iteración 2

Caso de Prueba de Aceptación	
Código: HU4_P1	**Número de HU:** 4
Nombre: Adicionar usuario.	
Descripción: Prueba para la funcionalidad adicionar usuario.	
Condiciones de ejecución: Se debe seleccionar un grupo al que se le asignará el usuario.	
Pasos de ejecución: Se intenta crear un usuario en el grupo seleccionado.	
Resultado esperado: El usuario es creado sin generar errores.	
Evaluación: Prueba insatisfactoria.	

Tabla 35 Prueba de aceptación adicionar usuario*.*

Caso de Prueba de Aceptación

Código: HU4_P2	**Número de HU:** 4
Nombre: Eliminar usuario.	
Descripción: Prueba para la funcionalidad eliminar usuario.	
Condiciones de ejecución: Se debe seleccionar el usuario que se desea eliminar.	
Pasos de ejecución: Se intenta eliminar un usuario seleccionado.	
Resultado esperado: Se elimina el usuario sin generar errores.	
Evaluación: Prueba satisfactoria.	

Tabla 36 Prueba de aceptación eliminar usuario*.*

Caso de Prueba de Aceptación	
Código: HU4_P3	**Número de HU:** 4
Nombre: Listar usuarios.	
Descripción: Prueba para la funcionalidad listar usuarios.	
Condiciones de ejecución: Se debe seleccionar un grupo para listar los usuarios que pertenecen al mismo.	
Pasos de ejecución: Se intenta listar los usuarios que pertenecen al grupo seleccionado.	
Resultado esperado: Se listan los usuarios del grupo seleccionado satisfactoriamente.	
Evaluación: Prueba satisfactoria.	

Tabla 37 Prueba de aceptación listar usuarios*.*

Caso de Prueba de Aceptación	
Código: HU5_P1	**Número de HU:** 5
Nombre: Detectar servicio remoto.	
Descripción: Prueba para la funcionalidad detectar servicio remoto.	
Condiciones de ejecución: La aplicación SASEP debe intentar cargar el módulo.	
Pasos de ejecución: Se intenta detectar si el servidor Subversion está instalado en	

la máquina remota.
Resultado esperado: Se detecta si Subversion está o no instalado en la máquina remota.
Evaluación: Prueba satisfactoria.

Tabla 38 Prueba de aceptación detectar servicio remoto.

3.4.1.3 Pruebas de aceptación definidas para la iteración 3

Caso de Prueba de Aceptación	
Código: HU6_P1	**Número de HU:** 6
Nombre: Configurar tipo de autenticación.	
Descripción: Prueba para la funcionalidad configurar tipo de autenticación.	
Condiciones de ejecución: Se debe seleccionar un repositorio para configurar el tipo de autenticación del mismo.	
Pasos de ejecución: Se intenta configurar el tipo de autenticación del repositorio seleccionado.	
Resultado esperado: Se configura el tipo de autenticación sin generar errores.	
Evaluación: Prueba satisfactoria.	

Tabla 39 Prueba de aceptación configurar tipo de autenticación.

Caso de Prueba de Aceptación	
Código: HU7_P1	**Número de HU:** 7
Nombre: Realizar copia segura.	
Descripción: Prueba para la funcionalidad realizar copia segura	
Condiciones de ejecución: Se debe seleccionar un repositorio para realizarle la copia segura.	
Pasos de ejecución: Se intenta realizar una copia de seguridad del repositorio seleccionado.	
Resultado esperado: Se realiza la copia sin generar errores.	

Evaluación: Prueba satisfactoria.

Tabla 40 Prueba de aceptación realizar copia segura.

3.5 Conclusiones

En este capítulo se elaboraron los modelos necesarios para llevar a cabo el proceso de implementación del sistema. Se detallaron las tareas que fueron definidas a partir de las historias de usuarios para lograr así mejor solución de las mismas. Además, se formularon las pruebas de aceptación para fomentar la conformidad y seguridad del cliente con respecto al producto. Con el fin de este capítulo se da por terminada la propuesta de desarrollo del presente trabajo.

CONCLUSIONES GENERALES

Una vez finalizado el desarrollo de la presente investigación, se puede concluir con la satisfacción de haber obtenido un módulo integrable a la Suite de Administración para Servidores de Entornos Productivos (SASEP), el cual constituye una herramienta factible, que consolida y apoya en gran medida el trabajo de administración del servidor de control de versiones Subversion a los gestores de configuración en los entornos productivos de la UCI.

Se demostró la necesidad de desarrollar un sistema capaz de facilitar la administración del servidor Subversion, brindando así a los gestores de configuración la posibilidad de cumplimentar su trabajo de una forma menos tediosa.

Un aspecto novedoso, es que el sistema fue desarrollado basado en una arquitectura modular creada por el equipo de desarrollo con el objetivo de: obtener una herramienta portable adecuada a las necesidades particulares de cada administrador, evitar las instancias de las herramientas de administración en los servidores remotos, así como realizar el proceso de administración en varios servidores concurrentemente.

A su vez el uso de la metodología ágil de desarrollo XP, ha permitido desarrollar en un periodo de tiempo corto un producto con las funcionalidades especificadas y la documentación generada servirá de material de estudio para investigaciones posteriores.

La versión del producto obtenida, constituye una base para el trabajo del equipo y futuras versiones del sistema.

RECOMENDACIONES

Con el objetivo de dar continuidad al producto obtenido han surgido algunas ideas a tener en cuenta:

1. Continuar agregando nuevas funcionalidades, de acuerdo a las necesidades que surjan en los entornos productivos de la universidad y adecuando estas a las nuevas tendencias de las tecnologías.
2. Agregar al sistema la funcionalidad gestionar permisos de grupo, conforme a la estructura de los repositorios.
3. El trabajo sea tomado como material de estudio por aquellas personas que deseen desarrollar un sistema similar o futuras versiones.

REFERENCIAS BIBLIOGRÁFICAS

1. **Blogspot.** Blogspot. [En línea] 24 de Febrero de 2009. http://tutorial-administracion.blogspot.com/2009/02/11-definicion-y-objetivos.html.

2. **TechTerms.** Tech Terms. [En línea] 19 de Noviembre de 2011. http://www.techterms.com/definition/server.

3. **Chacon, Scott.** Pro Git. [En línea] 2007. http://progit.org/book/es/ch1-1.html.

4. **Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato.** *Version Control with Subversion.* 2006.

5. **Sun Microsystems.** Los beneficios de la programación modular. 2007.

6. **Asesoría informática.** Asesoría informática. [En línea] Septiembre de 2011. http://www.asesoriainformatica.com/definiciones_p.htm.

7. **Visual Svn Server.** VISUALSVNSERVER. [En línea] 2010. http://www.visualsvn.com/server/.

8. **Svn Manager.** SVNManager. [En línea] 2009. http://svnmanager.org/index.php?page=install.

9. **Tigris.** Tigris.org. [En línea] 18 de Diciembre de 2009. http://svncontrol.tigris.org/.

10. **Pressman, Roger S.** *Ingeniería del Software, un enfoque Práctico (Quinta edición edición).* 2003. 84-481-3214-9.

11. **Jocham, Ralph.** Scrum.org. [En línea] 12 de Abril de 2012. http://agiletips.blogspot.com/2012/04/separation-of-power-in-scrum.html.

12. **Chin, Gary.** *Agile Project Management: How to Succeed in the Face of Changing Project Requirements.* s.l. : AMACOM, 2004.

13. **Alex Homer, Dave Sussman, Rob Howard, Brian Francis, Karli Watson, Richard Anderson.** *Professional ASP.NET 1.1.* 2004. 0764558900.

14. **Aportale.** Qt Labs Developer Blogs. [En línea] 11 de Julio de 2007. http://labs.qt.nokia.com/2007/07/11/develop-qt-applications-in-eclipse/.

15. **Blocks, Equipo de desarrollo Code.** Code::Blocks. [En línea] 21 de Diciembre de 2011. http://www.codeblocks.org/features.

16. **Nokia Corporation.** Qt. [En línea] 2008. http://qt.nokia.com/products/developer-tools/.

17. **Beck, Addison Wesley.** *Extreme Programming Explained.* 200.

18. **Crispin L, House T, Addison Wesley.** *Testing Extreme Programming.* 2002.

19. **Randal L. Schwartz, brian d foy, Tom Phoenix.** *Learning Perl.* s.l. : OREILLY, 2011.

20. **Andrés Marzal, Isabel García e Isabel Gracia.** *Introducción a la programación o o con Python .* 2003.

21. **Bjarne Stroustrup, Addison Wesley.** *El lenguaje de programación C++.* Madrid : s.n., 1998. ISBN 84-7829-019-2.

22. **EnCoded Communications Group.** C++ Builder Developer's Journal. [En línea] 2005. http://bcbjournal.org/.

23. **Code Lite.** codelite. [En línea] 2009. http://www.codelite.org/Main/ReadMore.

24. **The GNOME Project.** GNOME DEV CENTER. [En línea] 2011. http://developer.gnome.org/gtkmm-tutorial/unstable/sec-gtkmm.html.es.

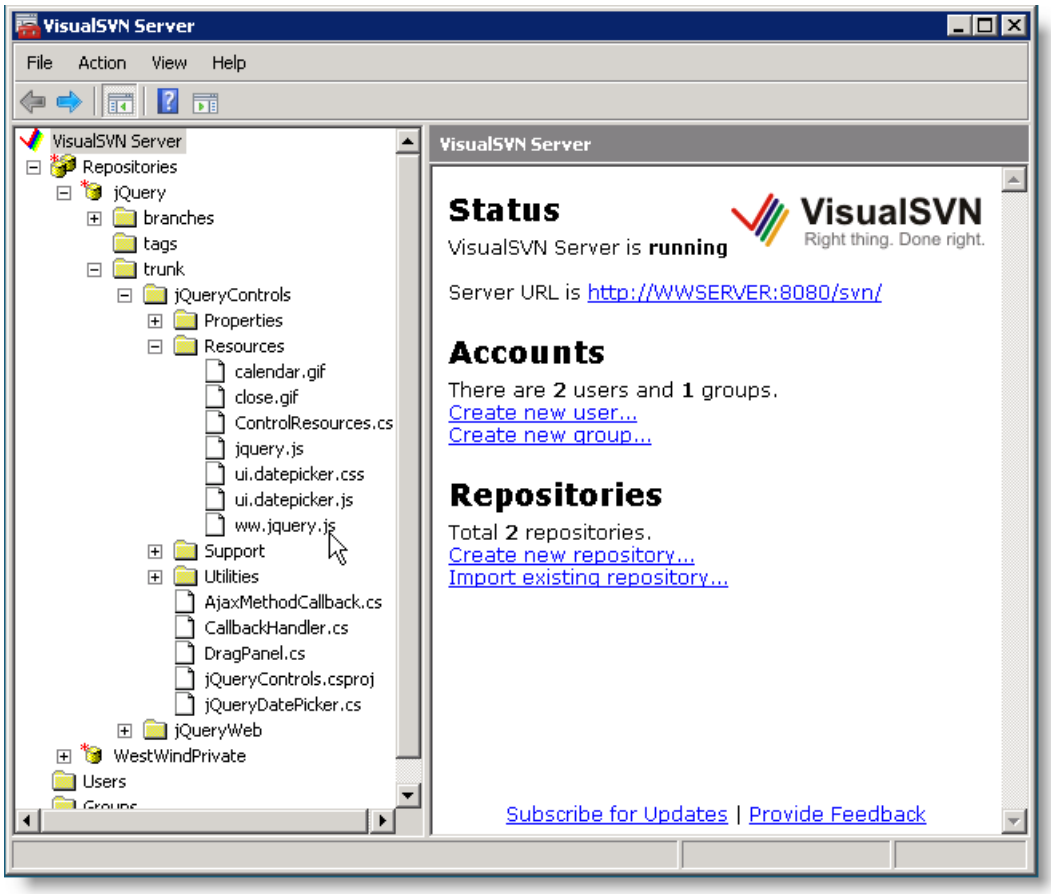
25. **wxWidgets Project.** wxWidgets. [En línea] http://www.wxwidgets.org/about/.

26. **Nokia Corporation.** Qt. [En línea] 2008. http://qt.nokia.com//products.

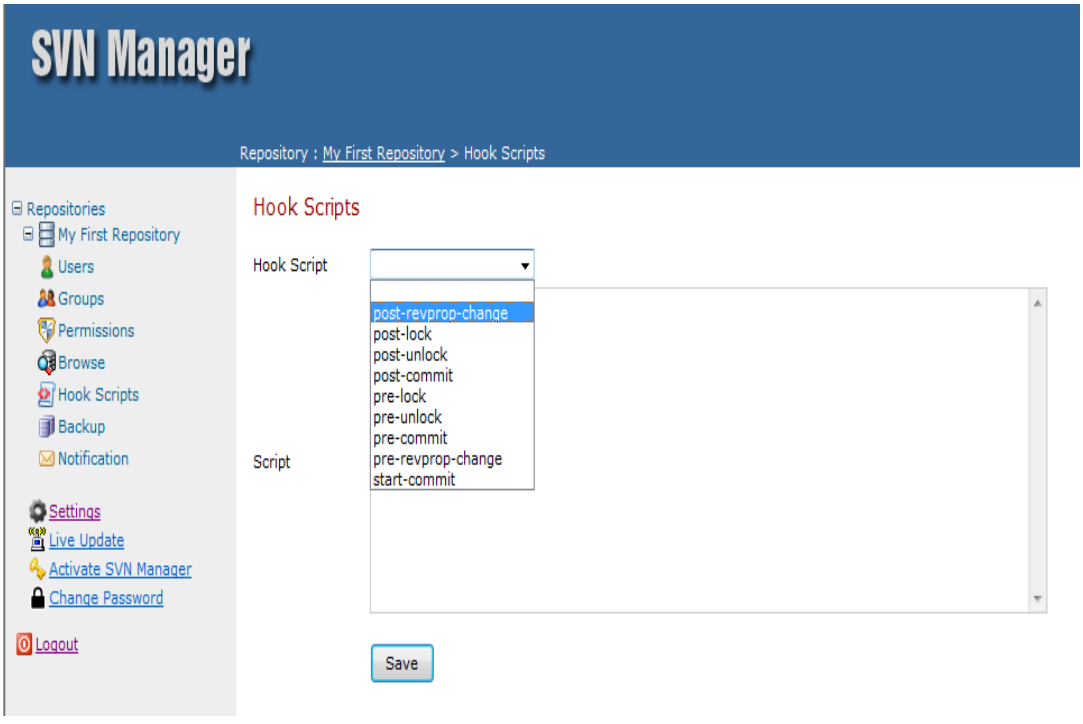
27. **Andrea Deluque, Katerin Torrenegra.** Lenguaje De Programación . [En línea] 13 de Mayo de 2012. [Citado el: 16 de Mayo de 2012.] http://lenguajesdeprogramacionupc.blogspot.com/.

ANEXOS

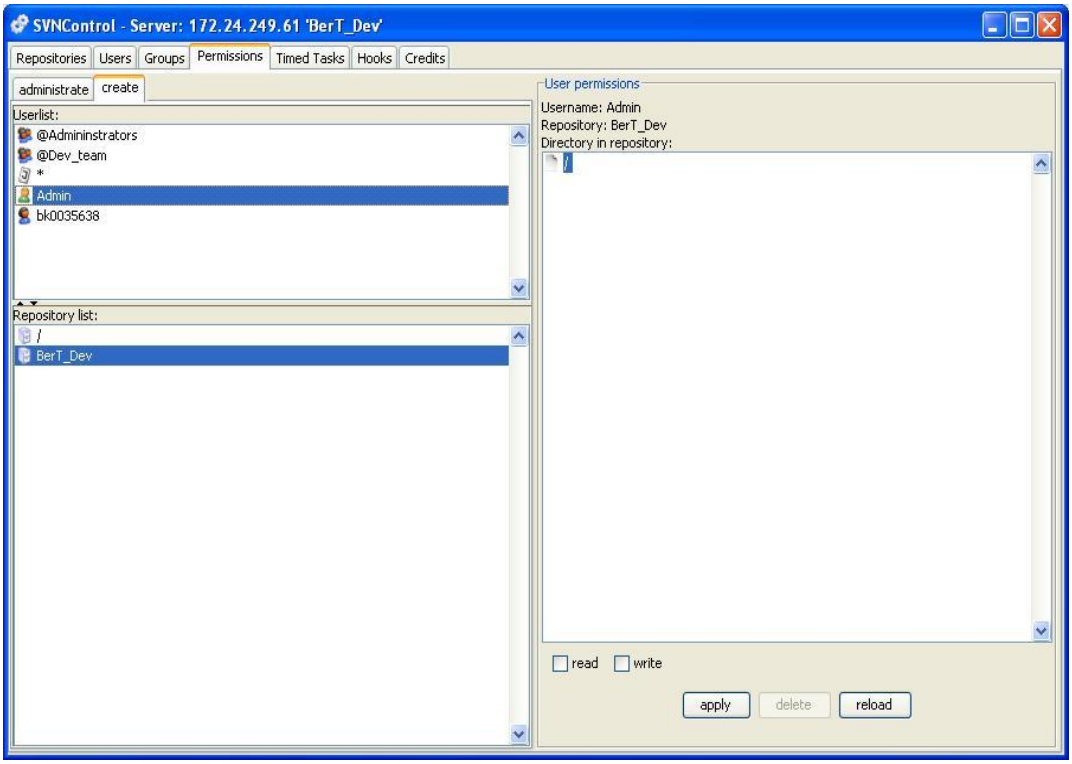
Anexo 1. VisualSVN Server



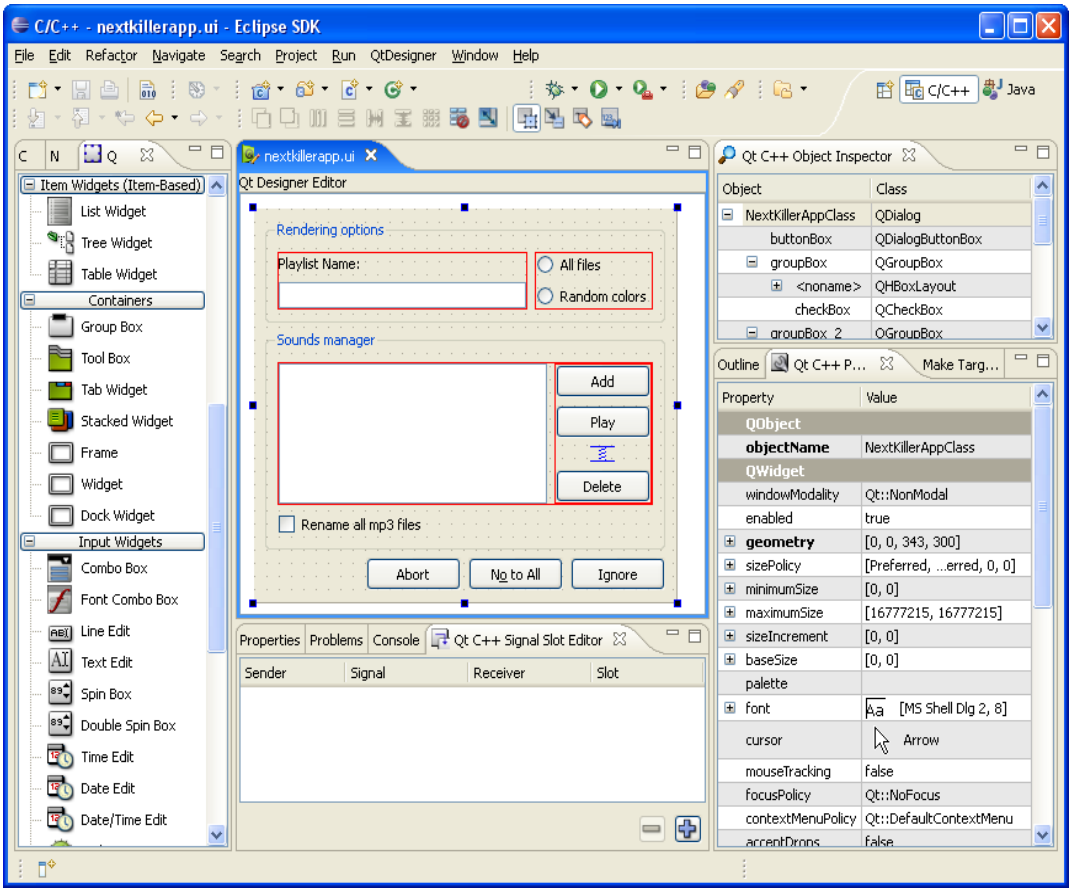
Anexo 2. SVNManager



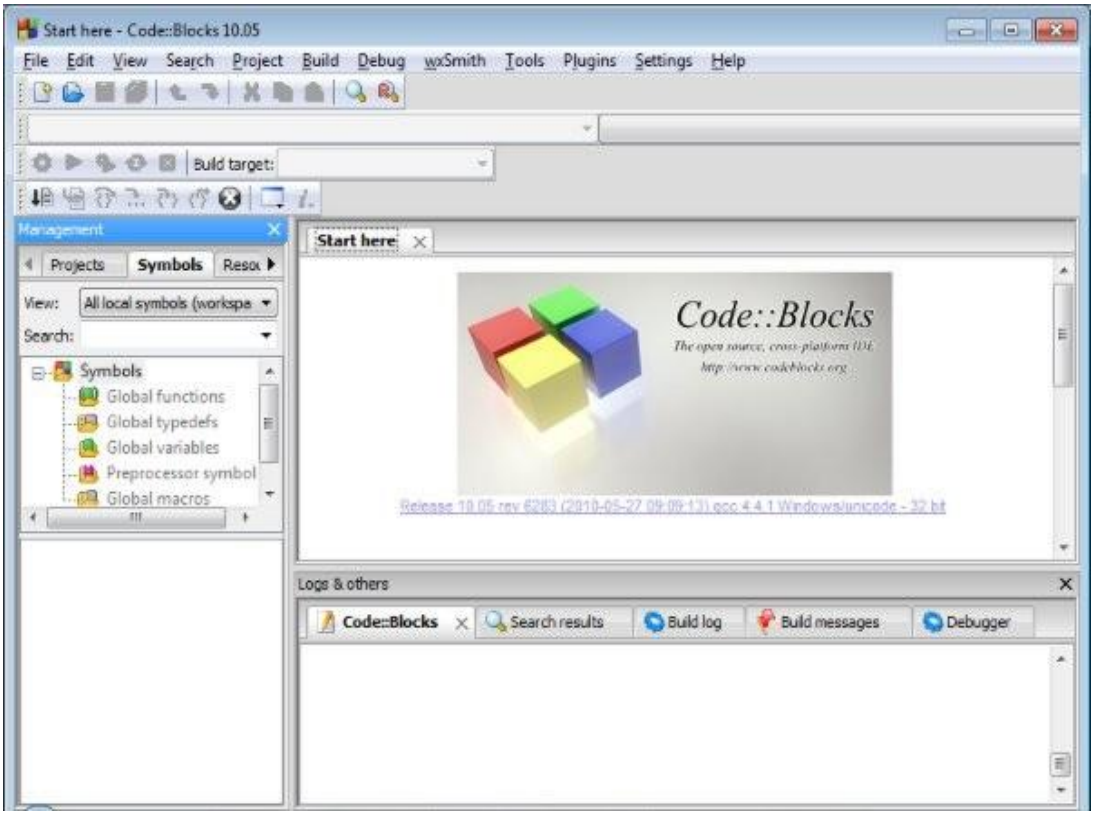
Anexo 3. SVNControl



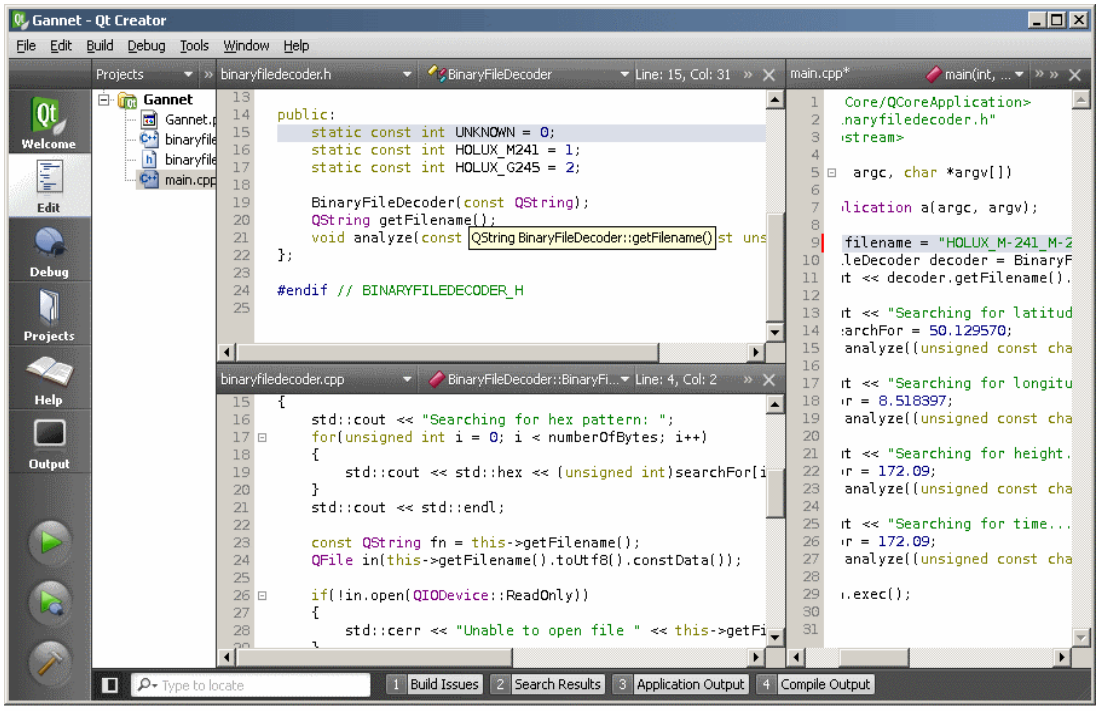
Anexo 4. Eclipse + Qt Plugin



Anexo 5. Code Blocks



Anexo 6. Qt Creator



GLOSARIO DE TÉRMINOS

API: Del inglés *Application Programming Interface* - Interfaz de Programación de Aplicaciones, es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrecen cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Active Directory (AD): Es el término que usa Microsoft para referirse a su implementación de servicio de directorio en una red distribuida de computadoras. Utiliza distintos protocolos (principalmente LDAP, DNS, DHCP, Kerberos y otros). Su estructura jerárquica permite mantener una serie de objetos relacionados con componentes de una red, como usuarios, grupos de usuarios, permisos y asignación de recursos.

Código abierto: Es una tendencia internacional del desarrollo de software que profesa la distribución del código junto a las aplicaciones, se rigen por licencias tales como GNU/GPL.

Compilador: Es un programa informático que traduce un programa escrito en un lenguaje de programación a otro, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser un código intermedio (bytecode), o simplemente texto. Este proceso de traducción se conoce como compilación.

GPL: Licencia Pública General de GNU o más conocida por su nombre en inglés *General Public License.* Está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Java Development Kit (JDK): Es un software que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red. En la unidad de red se pueden tener las herramientas distribuidas en varias computadoras y trabajar como una sola aplicación.

Java Runtime Environment (JRE): Es un conjunto de utilidades que permite la ejecución de programas Java. En su forma más simple, el entorno en tiempo de ejecución de Java está conformado por una Máquina Virtual de Java o JVM, un conjunto de bibliotecas Java y otros componentes necesarios para que una

aplicación escrita en lenguaje Java pueda ser ejecutada. El JRE actúa como un "intermediario" entre el sistema operativo y Java.

LDAP: *Lightweight Directory Access Protocol*, Protocolo Ligero de Acceso a Directorios, es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido.

Multiplataforma: Programa o aplicación que puede utilizarse sin inconvenientes en distintas plataformas de hardware y sistemas operativos.

SDK: Un kit de desarrollo de software o SDK (siglas en inglés de *software development kit*) es generalmente un conjunto de herramientas de desarrollo de software, que le permiten al programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos y otros.

SSH: *Secure SHell*, en español: intérprete de órdenes segura, es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos.

SSL: Del inglés *Secure Sockets Layer* es un protocolo criptográfico que proporciona comunicaciones seguras en Internet.

TIC: Tecnologías de la Información y las Comunicaciones, se encargan del estudio, desarrollo, implementación, almacenamiento y distribución de la información mediante la utilización de hardware y software como medio de sistema informático.

Unix: Sistema operativo atribuido a Ken Thompson y comercializado por la empresa AT& T en la década de los 70s que alcanzó mucho éxito, sobre todo en las universidades y posteriormente en las empresas. Entre sus principales características resalta que es: portable, robusto y flexible, actualmente goza de gran popularidad dentro de la tecnología de Internet.