Universidad de las Ciencias Informáticas

Facultad 6

SVNAdmin 1.0: Aplicación web para la gestión de repositorios SVN.

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

Autores: Yailin Napoles Luis

José Eduardo Trujillo Cerviño

Tutores: MSc. Reynaldo Álvarez Luna

Ing. Carlos Osiel Rojas Velázquez

"La Habana, junio 2015"
Año 57 de la Revolución

"En la tierra hace falta personas que trabajen más y critiquen menos, que construyan más y destruyan menos, que prometan menos y resuelvan más, que esperen recibir menos y dar más, que digan mejor ahora que mañana"
Che.

Declaración de autoría

Declaramos ser autores de la presente tesis titulada SVNAdmin 1.0: Aplicación web para la gestión de repositorios SVN y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____
Yailin Napoles Luis

_____
José Eduardo Trujillo Cerviño

_____
MSc. Reynaldo Álvarez Luna

_____
Ing. Carlos Osiel Rojas Velázquez

Agradezco:

A mis padres por el apoyo que siempre me han dado, y sobre todo por la confianza que siempre han depositado en mí: mi mamá por enseñarme que de los débiles nunca se ha escrito nada y que los nervios son solo unos bichitos que hay que aplastarlos para que no nos venzan. Mi padre por todo ese cariño que siempre me ha dado y por hacerme sentir la persona más importante en su vida, por hacerme reír ante cualquier dificultad, y hacerme ver la cosas con carácter "deportivo". A mi segundo padre, Edelio por soportarme y estar siempre dispuesto a ayudarme ante cualquier situación.

A mi novio Rodolfo, por ser mi amigo, mi compañero, mi pañuelo de lágrimas, por apoyarme, por entenderme, por estar a mi lado aun cuando es difícil estarlo. Por enseñarme la diferencia entre "preocuparse" y "ocuparse". Te amo mi amor.

A mi compañero de tesis por compartir conmigo y hacer posible el cumplimiento de este sueño y a su novia Yaima, por hacerme entenderlo.

A mis suegros, por apoyarme tanto, y considerarme como una hija más.

A mi hermanita del alma Ilianet, por ser mi compañera de lucha durante estos 5 años, y brindarme siempre su apoyo, estando siempre dispuesta a escucharme por compartir conmigo tantos momentos de alegrías y tristezas. A su novio Andy, por convertirse en un gran amigo.

A Mili, Elizabeth, y a mi quinteto favorito inseparable de estos 5 años, Javier, Guillermo, Yasel, Eddy Roy, incluido Carlos, por el apoyo incondicional que siempre me han brindado.

A mis compañeras de apartamento, Jessie, Glennis, Sonia, Aryanna, Laritza, Yanet, por compartir momentos tan lindos juntas, y enseñarme el valor de la amistad.

A mis tutores, por haber sido la salvación de mi carrera, por haber aparecido en el momento preciso, por la labor y el papel tan importante que cumplieron para permitirme hacer este sueño realidad.

A mi "hada madrina", por haberse ganado ese apodo, y sobre todo mi cariño.

Al tribunal y a todos los que de una forma u otra aportaron su granito de arena durante estos 5 años para hacer posible el cumplimiento de este sueño. A ustedes gracias.

Dedicatoria:

A mi hermanito Yediel, ya que siempre me he sentido con la responsabilidad de ser su ejemplo a seguir. A ti mi hermanito, para demostrarte que cuando uno se propone una meta, siempre la puede cumplir. Te quiero mucho,

…muchas gracias.

Yailin

Agradezco:

Antes que todo a mi mamá por apoyarme en todo y estar siempre a mi lado.

A mi abuela y abuelo que me han ayudado todo lo que han podido.

A mi novia por estar presente en los momentos buenos y malos.

A mi tía que cuando he necesitado de ella siempre ha estado presente.

A mis suegros y cuñado por el apoyo brindado, a ustedes muchas gracias.

A mi amigos y amigas los que están ahora en la universidad y los que lamentablemente no pueden estar ahora aquí conmigo, pero que a lo largo de estos cinco años compartimos momentos inolvidables, a todos ustedes muchas gracias.

A mis amistades que estuvieron presente en un momento u otro.

A mis tutores que cuando más enredado estaba el año encontraron un tema de tesis para desarrollar, a ellos que han dedicado parte de su tiempo libre a nosotros gracias.

A mi dúo de tesis que se ha comportado de una manera excelente, a ti también muchas gracias.

A todo aquel que de una forma u otra me enseñó algo de cómo funciona la vida o me apoyó en la carrera, a todos los presentes muchas gracias por estar aquí en uno de los momentos más decisivos de mi vida y a los que no están y quisieron estar, gracias a ustedes también.

Dedicatoria:

A mi madre y a mi familia en general por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo perfectamente mantenido a través del tiempo, por dejarme siempre escoger mi camino y apoyarme en el mismo,

…muchas gracias.

José Eduardo

RESUMEN

La gestión de repositorios de código es una tarea de vital importancia para la gestión de proyectos, ya que brinda la garantía del desarrollo de buenas prácticas y seguridad en entidades dedicadas a la producción de software, cumpliendo como principal objetivo almacenar y llevar el control del versionado del software desarrollado. En particular la administración de los repositorios y la gestión adecuada de permisos se dificulta al depender de los administradores de los sistemas. El presente trabajo expone SVNAdmin 1.0, una aplicación desarrollada con el objetivo de garantizar la gestión de los repositorios SVN, sus usuarios, grupos y permisos, así como la generación de reportes y análisis de logs. La herramienta por su diseño tiene además el objetivo de contribuir a la automatización de procesos de gestión de proyectos, como la gestión de la configuración. Su impacto en las entidades de desarrollo permitirá llevar a nivel de proyecto la gestión completa de la configuración de sus repositorios y garantizará a su vez la implementación de las políticas que se definan institucionalmente con respecto a los repositorios de código. Para el desarrollo de la aplicación se utilizaron diferentes tecnologías como el lenguaje Python v2.7, para el cual su utilizó el framework de desarrollo Django v1.7 con el IDE PyCharm v4.0.2. Además fue utilizada la herramienta Visual Paradigm v8.0 para el modelado y diseño de la aplicación web.

Palabras clave: gestión de proyectos, gestión de repositorios, gestión de la configuración, Subversion.

ABSTRACT

The code repository management is a task of vital importance for project management as it provides a guarantee for the development of good practices and security entities engaged in the production of software, serving as main objective to store and keep track of versioning of these the certainty of developing best practices and security in entities dedicated to software production. In particular the administration of repositories and proper management of permits is difficult when depending on system administrators. The present work exposes SVNAdmin 1.0, a tool that guarantees complete management of SVN repositories, users, groups and permissions as well as reports generation and log analysis to assess certain indicators that contribute to project control and management. The tool for its design also aims to contribute to the automation of project management processes such as configuration management. Its impact on development entities will allow taking at project level the full management of its repositories setting ensuring at the same time implementing the policies institutionally defined according to code repositories. For the development of the application different technologies were used like Python v2.7 language, for which it was used the development framework Django v1.7 with IDE PyCharm v4.0.2. It was also used Visual Paradigm v8.0 for modeling and application design.

Keywords: configuration management, management repositories, project management, Subversion.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El avance de las Tecnologías de la Información y las Comunicaciones (TIC) ha servido como punto de apoyo para el desarrollo de la sociedad actual, contribuyendo con la formación y la productividad del hombre. En Cuba en los últimos tiempos ha incrementado la incorporación del uso de las TIC a la vida cotidiana del pueblo. Debido a este auge, el país se ha trazado como objetivo alcanzar un nivel relevante en la producción y desarrollo de software, el cual consiste en resolver problemas que existen en la vida cotidiana del hombre mediante el uso avanzado de las TIC, en aras de aumentar las relaciones entre el mundo informático y la población trabajadora.

Una característica esencial en la evolución de un software es el aporte que brindan las distintas personas que forman parte del equipo de desarrollo. De esta forma se evidencia el desarrollo colaborativo, relacionado con la creación de grupos mixtos de personas con habilidades y experiencias diversas en diferentes áreas del conocimiento, así como estudiantes y otras personas interesadas que se comprometen a colaborar en un proyecto. El desenvolvimiento de estos grupos de desarrollo, muestra que las actividades de este tipo son tareas que se realizan de forma distribuida y que envuelven a personas con competencias diversas y responsabilidades bien definidas (Dodero, y otros, 2014).

Durante el desarrollo de un software es importante siempre tener en cuenta que si no se trabaja con la responsabilidad que se requiere, puede ocurrir la pérdida del código fuente que se ha ido desarrollando, lo cual conlleva a la repetición de un trabajo después de largas jornadas de esfuerzos. Para evitar estos problemas lo común es realizar salvas del código periódicamente; sin embargo, esto no resuelve del todo la posible pérdida de la información, ya que si se está trabajando en un grupo grande de desarrollo diferentes personas manipulan al mismo tiempo ficheros que son modificados. Esto provoca problemas con el versionado, siendo engorrosa la modificación de la aplicación para los distintos usuarios, es decir, la ocurrencia de distintas versiones simultáneas. Una forma más segura de evitar estos problemas, es la utilización de sistemas de control de versiones, que diferentes autores como Ben Collins lo definen como *"el arte del manejo de los cambios en la información*" (Collins-Sussman, y otros, 2002).

Dentro de los sistemas de control de versiones que existen se encuentra CVS (Concurrent Versions System o Control de Versiones) el cual procura que las entidades desarrolladoras de software controlen el historial de los ficheros y documentos que se van generando a lo largo del desarrollo del software. Como una

alternativa a CVS, surge SVN[1] (Subversion), con el objetivo de corregir las debilidades que presentaba inicialmente CVS, preservando la misma metodología, pero sin duplicar sus defectos en el versionado de directorios, historial de versiones, entre otros. Por las utilidades que presentan las funcionalidades de SVN y el diseño modular del mismo, existen muchas herramientas administrativas que permiten gestionar los repositorios Subversion, entre las cuales se encuentran: WebSVN[2], VisualSVN[3], SVNManager[4], USVN[5], SVNAccess[6] y UberSVN[7]. El uso de este tipo de herramientas incide en una mejor usabilidad en la gestión de los repositorios de las aplicaciones informáticas que se desarrollan.

La Universidad de las Ciencias Informáticas (UCI), entidad desarrolladora de software, cuenta con una red de catorce centros de desarrollo (ver Anexo 1), los cuales gestionan proyectos utilizando repositorios independientes para el control de versiones del código generado en dichos proyectos. La mayoría de los repositorios utilizan como sistema de control de versiones Subversion para versionar el código de los proyectos de desarrollo. En diciembre de 2013 se generalizó el uso de la herramienta administrativa UberSVN para la gestión de estos repositorios. UberSVN convierte a Subversion en una plataforma abierta, diseñada para usuarios con cualquier nivel de habilidad y equipos de desarrollo de cualquier tamaño. Cada equipo de desarrollo tiene su propia página principal que describe a los miembros del equipo, enumera los proyectos en los que están trabajando, los repositorios que están usando y su última actividad y estado.

A pesar de sus características ventajosas, UberSVN en su versión libre solo brinda la posibilidad de la administración centralizada de los permisos. Esto implica que varias personas tengan acceso de edición al archivo de configuración de permisos de los diferentes repositorios, ocasionando problemas de seguridad. Otra de sus características es que no es de código abierto, por lo que no permite ser extendida y lograr la comunicación con los sistemas externos de la gestión de la producción como el GESPRO (utilizado para la gestión de proyectos). Por otra parte presenta limitaciones en el seguimiento de la actividad en los repositorios mediante la representación de reportes o gráficos, ya que los reportes que brinda el sistema están orientados al tamaño de los repositorios sin presentar detalle alguno sobre la actividad de los usuarios sobre estos.

[1] Sitio oficial disponible en: http://subversion.apache.org
[2] Sitio oficial disponible en: http:// w ebsvn.tigris.org
[3] Sitio oficial disponible en: https://w w w .visualsvn.com
[4] Sitio oficial disponible en: http://svnmanager .sourceforge.net
[5] Sitio oficial disponible en: http://w w w .usvn.info
[6] Sitio oficial disponible en: http://svn-access-mana.sourceforge.net/ o http://w w w .svn-access-manager.org
[7] Sitio oficial disponible en http://docs.ubersvn.com

Por la situación anteriormente descrita, se plantea como **problema a resolver**: ¿cómo mejorar la usabilidad y seguridad en la gestión de repositorios SVN en la red de los centros de desarrollo de la UCI? La investigación tiene como **objeto de estudio:** la gestión de repositorios SVN, enmarcado en el **campo de acción:** las aplicaciones web para la gestión de repositorios SVN.

Se presenta como **objetivo general:** desarrollar una aplicación web que garantice la usabilidad y seguridad para la gestión de repositorios SVN en la Red de Centros de desarrollo de la Universidad de las Ciencias Informáticas. Para orientar la investigación hacia el cumplimiento del objetivo general, se plantean las siguientes **preguntas científicas:**

- ¿Cuáles son las bases teóricas que fundamentan el desarrollo de aplicaciones web para la gestión de repositorios SVN?
- ¿Qué características debe cumplir la aplicación web para garantizar la usabilidad y seguridad en la gestión de repositorios SVN?
- ¿Qué mecanismos de validación deben realizarse para verificar que la aplicación web garantiza la usabilidad y seguridad en la gestión de repositorios SVN?

Para dar cumplimiento al objetivo general se proponen como **tareas de investigación:**

- Revisión bibliográfica de la gestión de repositorios SVN para la identificación de los conceptos fundamentales relacionados con este tema.
- Selección de las metodologías, herramientas y tecnologías a utilizar en el desarrollo de la aplicación web para la gestión de repositorios SVN.
- Descripción de sistemas de gestión de la producción para evaluar las posibles interacciones con la aplicación web a desarrollar.
- Definición de las funcionalidades de la aplicación web que mejoren la usabilidad y seguridad de la gestión de repositorios SVN.
- Diseño de la aplicación web a partir de las funcionalidades especificadas para representar de forma estructural cada uno de sus elementos.
- Implementación de las funcionalidades identificadas para obtener la aplicación web destinada a la gestión de repositorios SVN.
- Realización de pruebas para validar si la aplicación web responde a las necesidades de las funcionalidades identificadas.

Los **métodos de investigación científicos** que se tuvieron en cuenta para el desarrollo de esta investigación son los siguientes:

- Métodos teóricos:
  - Método Analítico – Sintético: permitió llegar a conclusiones a partir del análisis por separado de los componentes que integran el problema, como la función que cumplen las herramientas de gestión de repositorios y en qué consisten las mismas. Se analizaron las variantes existentes para garantizar el impacto en la seguridad y usabilidad como factores claves para el alcance del objetivo de la investigación.
  - Método de modelación: es una reproducción simplificada de la realidad. Permitió descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio planteado, es decir, a partir del estudio de este objeto, se modelaron las clases del dominio y sus relaciones, así como otros diagramas que muestran una abstracción del entorno de la solución planteada.
- Métodos empíricos:
  - Encuesta: permitió recopilar gran cantidad de información mediante cuestionarios realizados a usuarios administradores de Subversion, sobre las herramientas de gestión de repositorios y la importancia que tienen las mismas (ver Anexo 2).
  - Entrevista: permitió obtener información sobre la función que cumple actualmente la herramienta que se utiliza en la Universidad para llevar a cabo el control de versiones. Además, la misma permitió identificar los requerimientos del cliente y las deficiencias que presenta la herramienta actual (ver Anexo 3).

El contenido de este trabajo se encuentra distribuido de la siguiente forma:

Capítulo 1: *"Fundamentos teóricos de la gestión de repositorios SVN".* Constituye la base teórica de la investigación realizada. Se describen los principales conceptos asociados a la gestión de repositorios SVN. Además se describen las herramientas y metodologías utilizadas para el desarrollo de la aplicación web.
Capítulo 2: *"Análisis y diseño de la aplicación web para la gestión de repositorios SVN".* Se realiza la captura y especificación de los requisitos, por otra parte se realiza el modelado del sistema y se recogen los distintos artefactos generados por el mismo.
Capítulo 3: *"Implementación y prueba de la aplicación web para la gestión de repositorios SVN".* Se analizan los diferentes elementos que se tuvieron en cuenta para la implementación de la propuesta de solución. Por otra parte se exponen los resultados obtenidos una vez aplicadas las pruebas definidas.

CAPÍTULO I: Fundamentos teóricos de la gestión de repositorios SVN.

Introducción

En este capítulo se realiza un estudio sobre los principales conceptos asociados a la gestión de repositorios SVN, para lograr un mayor entendimiento del contenido de la presente investigación. Se analizan las principales características y el funcionamiento de la gestión de repositorios SVN en herramientas administrativas ya existentes. También se reflejan las herramientas, metodologías y tecnologías a utilizar en el proceso de desarrollo de la aplicación web.

1.1 Conceptos asociados a la gestión de repositorios SVN

Con el objetivo de comprender los diferentes términos que abarca la problemática de la propuesta de investigación presente, se han analizado una serie de conceptos a partir de las diferentes bibliografías consultadas.

Repositorio

El diccionario de la Real Academia de la Lengua Española define repositorio como el lugar donde se guarda algo (Diccionario, 2015). Otra definición más cercana, define repositorio como “un sitio web centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos” (Salamanca, 2015). Particularmente en el ámbito de Subversion, los repositorios son la parte principal, el almacén central de datos y versiones de un número indeterminado de proyectos. Este repositorio guarda información en forma de un árbol de archivos (una jerarquía típica de archivos y directorios). Cualquier número de clientes se conectan al repositorio para luego leer o escribir a estos archivos (Collins-Sussman, y otros, 2002).

Las fuentes consultadas relativas a la definición de repositorio en este ámbito, coinciden en que los repositorios representan el lugar o el sitio donde se almacena toda la información versionada y permite la interacción de diversos usuarios. Estos constituyen la base del desarrollo del control de versiones Subversion.

Subversion (SVN)

Subversion es un sistema de control de versiones libre y de código fuente abierto (Collins-Sussman, y otros, 2002). Permite a los programadores mantener el control de las versiones de lo que han desarrollado, como el código fuente, mediante un almacén central de datos conocido como repositorio. Ya que el repositorio almacena información como un sistema de ficheros con forma de árbol, permite que varios clientes accedan a estos ficheros y realicen acciones de modificación y lectura, y posibilita almacenar todos los cambios realizados tales como la adición, eliminación y la reordenación de ficheros y directorios (Collins-Sussman, y otros, 2002).

Para el uso de Subversion lo más recomendable es tener un repositorio por cada proyecto; la figura 1 muestra una buena práctica para la organización de los archivos de un repositorio.

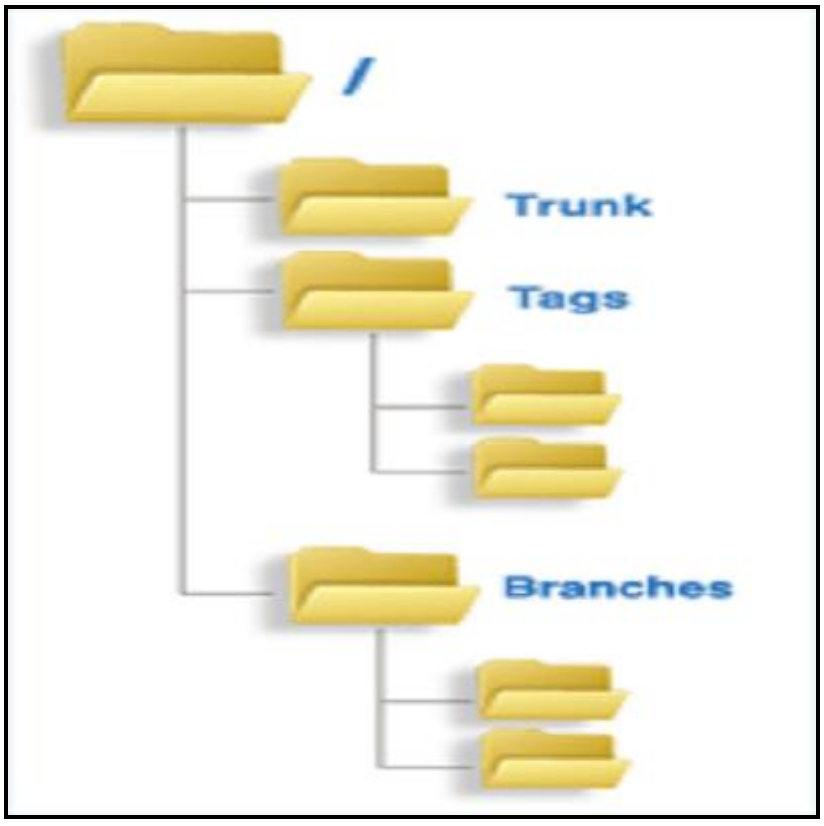

Fig. 1 Estructura de los Repositorios SVN

Estas definiciones sobre SVN, coinciden en un mismo punto: Subversion consiste en un sistema de control de versiones que almacena información sobre los cambios realizados en archivos de código fuente, donde se pueden realizar diferentes tareas de gestión sobre los repositorios que almacena.

Tareas de gestión en los repositorios SVN

Para garantizar la seguridad de los repositorios SVN es necesario establecer los permisos que limitarán el acceso de los usuarios, en dependencia de las acciones a realizar sobre un repositorio.

- **Tareas de administración**

Entre las estrategias para controlar el acceso a los repositorios se encuentra el control de acceso simple, que consiste en autorizar a ciertos usuarios el acceso de solo lectura o el acceso de lectura y escritura sobre el repositorio SVN de forma general. Otra estrategia para controlar el acceso a los repositorios es el control de acceso a directorios, que consiste en dar los permisos de lectura o lectura y escritura a determinados usuarios, para acceder a un directorio. Es importante mencionar que los permisos son heredados del directorio padre al hijo.

- **Tareas de usuario**

Cuando un usuario realiza modificaciones en el repositorio, SVN solo publicará estos cambios una vez que el usuario se lo indique explícitamente mediante el comando *"commit".* Una operación *"svn.commit"* puede publicar las modificaciones de cualquier número de archivos y carpetas como una transacción atómica, o sea, un usuario puede cambiar el contenido de los ficheros, crear, borrar, renombrar y copiar archivos y directorios y posteriormente publicar el conjunto completo de los cambios como una unidad. Otro comando que se utiliza sobre los repositorios SVN es la salida del comando *"svn.update",* el cual se utiliza para actualizar una copia de trabajo local con los cambios existentes en el servidor remoto; ya que usa la información en el directorio *".svn"*, que almacena datos administrativos de los directorios e información adicional en el repositorio, para decidir qué archivos necesitan ser extraídos y actualizados.

De forma general, tanto los administradores como los usuarios pueden realizar tareas sobre los repositorios, pero dichas tareas son restringidas por la configuración de los permisos que les son otorgados. Los usuarios solo pueden realizar aquellas acciones sobre los repositorios que el administrador le permita, ya que es este el encargado de otorgar dichos privilegios.

Por otra parte, existen varios comandos que proporcionan datos históricos del repositorio, entre los cuales se encuentran: el análisis de logs mediante la función "*svn log", para* mostrar mensajes de registro unidos a las revisiones, y la ruta de fichero cambiada en cada revisión. También se puede utilizar el comando "*svn diff*" que muestra los detalles específicos de cómo se cambió un fichero en un cierto plazo. Estos comandos de forma general, permiten realizar diferentes tareas sobre los repositorios SVN, como saber con qué

frecuencia ha trabajado un usuario en las últimas semanas, y a qué sesiones ha entrado. Estos permiten a los administradores o usuarios ver los cambios realizados en el repositorio y las actualizaciones más recientes (Collins-Sussman, y otros, 2002). La gestión de los permisos en SVN, requiere la intervención del administrador del servidor y se convierte en una tarea compleja para los miembros de los equipos de desarrollo. Por este motivo se han desarrollado varias herramientas con el objetivo de facilitar la gestión de la administración de los servidores SVN.

1.2 Herramientas para la gestión de repositorios SVN

Con el objetivo de definir las funcionalidades propias del sistema a desarrollar y adquirir nuevos conocimientos acerca de la gestión de repositorios SVN, se analizaron una serie de herramientas administrativas que permiten gestionar repositorios Subversion. A continuación se expone brevemente una caracterización sobre estas herramientas.

- **WebSVN**

WebSVN ofrece una vista sobre los repositorios Subversion que se han diseñado para reflejar la metodología de Subversion. Puede ver el registro de cualquier archivo o directorio y una lista de todos los archivos cambiados, añadido o eliminado en cualquier revisión dada. También puede ver las diferencias entre dos versiones de un archivo para ver exactamente lo que fue cambiado en una particular revisión (Thomas, 2011).

Características:

1. Interfaz fácil de usar.
2. Coloración de listados de archivos, para identificar los diferentes archivos mediante niveles de colores.
3. Está escrito en PHP.

- **VisualSVN**

VisualSVN Server es un paquete de instalación que incluye Subversion y el servidor web Apache ya configurados y listos para funcionar desde el primer momento. Su panel de control utiliza el aspecto familiar de las consolas de Windows, con un resumen de los repositorios activos y la dirección del servidor. Los repositorios se pueden crear o importar desde el mismo panel de VisualSVN Server. En las propiedades de cada elemento presenta una pestaña para cambiar los permisos de seguridad (VISUALSVN, 2015).

- **USVN**

USVN es una herramienta que presenta una interfaz web escrita en PHP utilizada para configurar los repositorios de Subversion. Su objetivo es facilitar la creación de nuevos proyectos sin tener que usar la interfaz de línea de comandos, por lo tanto, si no se tiene un acceso privilegiado al servidor se genera una lista de usuarios que permite el acceso a su código fuente (USVN, 2013).

- **SVN Access Manager**

SVN Access Manager es una herramienta utilizada para gestionar el acceso a los repositorios de Subversion. La herramienta proporciona administración de usuario y de grupo y los derechos de acceso (lectura/escritura) a un repositorio. SVN Access Manager soporta bases de datos MySQL, PostgreSQL y Oracle. Además utiliza proyectos para ofrecer a los usuarios el derecho a administrar sus propios módulos en un repositorio (ACCESSMANAGER, 2008).

- **UberSVN**

UberSVN es la última distribución de Subversion. Desarrollado para ampliar el alcance y la funcionalidad de Subversion (UBERSVN, 2012.). Es un programa gratuito, producto de software desarrollado por WANdisco. Como características de UberSVN se encuentran las siguientes:

1. Una interfaz de codificación social que permite que reveladores colaboren en tiempo real alrededor del código que destinan al depósito.
2. Una interfaz web para crear nuevos depósitos, usuarios gerentes y permisos, y administrar ajustes comunes del servidor.
3. Integración con un LDAP[8] servidor para autenticación delegada de usuarios.
4. Actualización en línea automatizada que hace la instalación actualizada rápida y confiable.
5. GUI[9] para permisos de usuarios gerentes y acceso al depósito.

Esta herramienta es la que se utiliza actualmente en la UCI para el control de versiones de los proyectos que se llevan a cabo en los centros.

Una vez realizado el estudio sobre estas herramientas de administración de Subversion, se realizó una comparación entre ellas (tabla 1), medida por un grupo de indicadores: interfaz, información que genera, gestión de permisos y licencia bajo la que se distribuye.

[8] Protocolo ligero de acceso a directorios (del inglés, Lightweight Directory Access Protocol)

[9] Graphic User Interface o Interfaz Gráfica de Usuario

Tabla. 1 Comparación entre las herramientas administrativas SVN

Herramienta /Indicador	Interfaz	Información	Gestión de permisos	Licencia bajo la que se distribuye
WebSVN	-Sencilla de utilizar	-Coloración de listados de archivos.	-Restricción basada en la ruta opcional de privilegios.	-WebSVN es liberado bajo la licencia de GNU (General Public License)
VisualSVN	-Fácil de usar	-Estado en tiempo real: VisualSVN rastrea cuidadosamente y muestra todos los cambios actuales realizados a su copia de trabajo. -Curva de aprendizaje corta: VisualSVN utiliza diálogos de TortoiseSVN y proporciona un asistente inteligente para poner sus fuentes bajo Subversion.	-En las propiedades de cada elemento hay una pestaña para cambiar los permisos de seguridad.	-VisualSVN está construido sobre la base del sistema de control de versiones Subversion de código abierto, que es un sistema de almacenamiento estándar para proyectos de software. -Licencia VisualSVN Comunidad: libre de cargo, licencia que permite utilizar VisualSVN en cualquier ordenador que no es miembro de un dominio de Active Directory. Es una licencia comunitaria que permite el uso comercial y es ideal para autónomos, estudiantes y aficionados.
USVN	-Fácil de usar	Exhibición del comentario del registro de los repositorios.	-Su objetivo es facilitar la creación de nuevos proyectos sin tener que usar la interfaz de línea de comandos, por lo tanto, sin un acceso privilegiado al servidor, USVN genera la lista de usuarios permitiendo el acceso a su código fuente. USVN permite una fácil gestión de restricción de acceso en los archivos de Subversion.	-USVN se encuentra bajo la licencia de CeCILL[10]

[10] Acrónimo en francés de "CEA CNRS INRIA Logiciel Libre" es una licencia francesa de softw are libre.

SVNAccess Manager	-Sencilla de utilizar	-El informe ofrece una visión general de acceso, sobre los derechos que presenta un grupo de usuarios.	-La herramienta proporciona administración de usuario y de grupo y los derechos de acceso (read /write) dedicados en un repositorio. -SVN Access Manager soporta bases de datos MySQL, PostgeSQL y Oracle.	-SVN Access Manager se distribuye bajo la GPL v2.[11]
UberSVN	-Una interfaz de codificación social que permite que reveladores colaboren en tiempo real alrededor del código que destinan al depósito.	-Proporciona reportes orientados al tamaño de los repositorios sin presentar detalle alguno sobre la actividad de los usuarios sobre estos.	-GUI para permisos del usuario gerentes y acceso al depósito.	-El producto está disponible de forma gratuita pero limitado, la versión completa tiene licenciamiento comercial.

El estudio realizado sobre estas herramientas y la comparación establecida entre ellas a partir de los diferentes indicadores definidos, aportó una serie de elementos a tener en cuenta para el desarrollo de la propuesta de solución. Dentro de estos elementos se destacan que en la interfaz gráfica se pueden diferenciar los distintos listados de archivos a través de la coloración de los mismos, la opción de mostrar a los administradores los derechos y permisos de cada grupo, así como la personalización de las opciones de seguridad para posibilitar la edición de los permisos de cada elemento por los usuarios.

1.3 Herramientas y sistemas del entorno

La Universidad cuenta con herramientas, cuya función es la organización y gestión de los proyectos pertenecientes a cada centro. Entre los sistemas que están estrechamente vinculados con Subversion, aportando elementos que apoyan la gestión de los permisos de cada proyecto y la gestión histórica de sus documentos, se encuentran:

[11] Licencia Pública General de GNU

- **GESPRO (suite para la gestión integrada de proyectos)**

Suite que permite a las entidades cuyos productos deban ser planificados en forma de proyectos, el seguimiento y control de estos, sean proyectos de inversión, construcción, desarrollo de software o mantenimiento. Posibilita además la toma de decisiones a tres niveles: nivel de proyecto, nivel de entidad ejecutora y nivel gerencial. El sistema se compone de trece módulos, los cuales posibilitan la gestión de alcance, tiempo, costos, calidad, recursos humanos, adquisiciones, riesgos, entre otros parámetros afines a la dirección integrada por proyectos (UCI, 2015).

Cuando se crea por primera vez un proyecto en GESPRO, se definen los roles para cada uno de los integrantes que lo desarrollarán. Luego estos roles son enviados a los administradores centrales de los repositorios, los cuales se encargan de configurar los permisos (de acuerdo al rol) en el repositorio correspondiente al proyecto que se comenzará a desarrollar.

- **Sistema de gestión de archivos Arkheia**

Aplicación informática que permite la creación de cuadros de clasificación para el registro y organización de la información a conservar en una institución. Particularmente en la UCI, a través de este sistema web se sube el código fuente y la documentación de los proyectos terminados, se guardan metadatos del software, para cuando se desee buscar información sobre el mismo. Además genera un filtrado que ayuda a buscar la información que se desea.
La función de este sistema es interactuar con el GESPRO para archivar toda la documentación de los proyectos terminados. Arkheia tendrá la posibilidad de lograr una comunicación con el repositorio Subversion, con el objetivo de acceder al código versionado de los proyectos cerrados.

1.4 Metodología de desarrollo

La metodología de desarrollo es un entorno de trabajo utilizado para estructurar, planificar, y controlar el desarrollo del producto, esta debe ser definida por el equipo de trabajo de acuerdo a las características del software y debe ser utilizado durante todo el ciclo de desarrollo del mismo. Existen diferentes metodologías de desarrollo, las cuales son clasificadas en dos grupos fundamentales: las metodologías tradicionales o pesadas y las metodologías ágiles o ligeras.

Dentro de las metodologías de desarrollo que más se destacan se encuentran: Programación Extrema (XP o Extreme Programming por sus siglas en inglés), OpenUp (Proceso Unificado Abierto), SCRUM (Modelo

Iterativo de Desarrollo de Software) y AUP (Proceso Unificado Abierto) como metodologías de desarrollo ágil. Por otra parte se encuentra RUP (Proceso Unificado de Desarrollo) como una metodología tradicional o pesada. Las metodologías ágiles permiten dar una mayor importancia al cliente y a la colaboración con el mismo, donde este llega a formar parte del equipo de trabajo. Además dan un mayor valor al desarrollo incremental del software con iteraciones muy cortas. Por otra parte, las metodologías tradicionales se centran únicamente en el control del proceso, donde se establecen rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas con las que se trabajarán, lo que quiere decir que el cliente no forma parte del equipo de trabajo, en este caso solo importa el proceso que se está llevando a cabo.

Para el desarrollo de la propuesta de solución, se seleccionó como metodología de desarrollo AUP, debido a que la Universidad ha definido que todos los proyectos se desarrollen basándose en la misma.

1.4.1 AUP (Proceso Unificado Ágil)

El Proceso Unificado Ágil (AUP, del inglés Agile Unified Process) es una versión simplificada de RUP, desarrollada por Scott Ambler, que describe una aproximación al desarrollo de aplicaciones que combina conceptos propios del proceso unificado tradicional con técnicas ágiles, con el objetivo de mejorar la productividad. AUP supone un enfoque intermedio entre XP y RUP, y tiene la ventaja de ser un proceso ágil que incluye explícitamente actividades y artefactos a los que la mayoría de desarrolladores ya están, de alguna manera, acostumbrados (Edeki, 2013.).

Se decidió utilizar una variación de la metodología AUP definida por la Universidad (AUP-UCI), la cual se adapta al ciclo de vida de los proyectos que se desarrollan en la misma, que consta de las siguientes fases que un proyecto atraviesa de forma secuencial (Sánchez, 2014):

- **Inicio:** En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

AUP-UCI, establece para el Modelado de Negocios tres variantes a utilizar en los proyectos: Casos de uso del Negocio (CUN), Descripción de Proceso de Negocio (DPN) y Modelo Conceptual (MC). Además también propone tres formas de encapsular los requisitos: Casos de uso del Sistema (CUS), Historias de Usuarios (HU) y Descripción de Requisitos por Proceso (DRP), que son agrupados por cuatro escenarios condicionados por el Modelo de Negocio que se utilice. Para llevar a cabo el Modelo de Negocio del presente trabajo se decidió utilizar el Modelo Conceptual por lo que solo se podrá modelar el sistema mediante el encapsulamiento de los requisitos a través de los CUS.

1.5 Lenguajes de modelado y de programación

En este subepígrafe se describe el lenguaje de modelado y de programación utilizado para el diseño e implementación de la aplicación web SVNAdmin 1.0.

1.5.1 Lenguaje de modelado

Un modelo constituye una ayuda para el análisis, diseño y comprensión de un sistema. El lenguaje de modelado consiste en un conjunto de normas estandarizadas que permiten la construcción de los modelos de dicho sistema. Para el diseño de la aplicación se decidió escoger el lenguaje de modelado que se define a continuación.

- **Lenguaje Unificado de Modelado (UML) v2.0**

UML (Unified Modeling Language), es un lenguaje de modelado unificado, el cual proporciona un medio gráfico de modelar varios componentes de un sistema de software. El componente diagrama de clase de UML se basa en diagramas E-R (Estructura general que permite expresar gráficamente el esquema de una empresa). Esta es una de las mejores herramientas para analizar y diseñar sistemas de software que ofrece un lenguaje común que todo desarrollador debe conocer. La misma es de suma importancia ya que permite tener una visión más completa del sistema mediante varios tipos de diagramas. Debido a las características del problema a resolver y la aplicación a desarrollar, se ha definido este lenguaje a utilizar ya que es de suma comodidad para el trabajo con lenguajes orientados a objetos, el cual será el utilizado en el proceso de implementación de la aplicación.

1.5.2 Lenguaje de Programación

Un lenguaje de programación es un conjunto de reglas o normas que permiten asociar a cada programa correcto un cálculo que será llevado a cabo por un ordenador (Almagro, 2011). Para el desarrollo de la propuesta de solución se utilizará el lenguaje de programación que se describe a continuación.

- **Python v2.7**

Python es un lenguaje de programación que presenta estructuras de datos de alto nivel[12], y una solución de programación orientada a objetos simple pero eficaz. Python es también adecuado como lenguaje de extensión para aplicaciones adaptables al usuario. Es un lenguaje interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. Permite escribir programas muy compactos y legibles. Los programas escritos en Python son típicamente mucho más cortos que sus equivalentes en C o C++, por varios motivos: los tipos de datos de alto nivel permiten expresar operaciones complejas en una sola sentencia, además no es necesario declarar los argumentos ni las variables (Lutz, 2013). Python además puede ser clasificado como un lenguaje interpretado, multiplataforma[13], de tipado dinámico[14]y multiparadigma[15] (Bahit, 2012). A diferencia de la mayoría de los lenguajes de programación, Python nos provee de reglas de estilos, a fin de poder escribir código fuente más legible y de manera estandarizada (Rossum, 2000). Para el desarrollo de la aplicación web SVNAdmin 1.0, se escogió este lenguaje de programación debido a las características que presenta descritas anteriormente, por lo que se decidió utilizar el framework de desarrollo que se describe a continuación.

Framework de desarrollo Django v1.7

Se decide utilizar como framework de desarrollo Django, ya que es un marco de desarrollo web sobre Python, que permite desarrollar rápidamente aplicaciones web (Ortega, y otros, 2007). Aunque se puede usar sin una base de datos, Django viene con un mapeado objeto-relacional a través del cual se puede describir la estructura de una base de datos en código Python. Django sigue el patrón arquitectónico MVC (Modelo Vista Controlador) tan al pie de la letra, que es conocido como un framework MVT (Modelo Vista Template), debido a que el "Controlador" es manejado por el mismo framework y la parte más importante se produce en los modelos, las plantillas y las vistas (Kaplan-Moss, 2013.). Fue seleccionado este framework de desarrollo debido a las características que presenta y la por su utilización en aplicaciones web sobre el lenguaje escogido.

1.6 Herramientas de desarrollo

- **Visual Paradigm v8.0**

[12] Consiste en una estructura sintáctica y semántica legible.
[13] Interpretado en diversos Sistemas Operativos como GNU/Linux, Windows, Mac OS, Solaris.
[14] Las variables, no requieren ser definidas asignando su tipo de datos, sino que éste, se auto-asigna en tiempo de ejecución, según el valor declarado.
[15] Acepta diferentes paradigmas (técnicas) de programación.

Es una herramienta de diseño UML y herramienta CASE diseñada para la ayuda en el proceso de desarrollo de software. Soporta estándares claves en la industria del software, tales como: Lenguaje de Modelado Unificado (UML), SysML[16], BPMN[17], XMI[18], entre otros. Ofrece un completo conjunto de herramientas de los equipos de desarrollo de software, necesario para la captura de requisitos, la planificación de programas, la planificación de controles, la clase de modelado y el modelado de datos. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y documentación. La herramienta UML también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (Paradigm, 2013). Para el modelado de la aplicación se decide utilizar Visual Paradigm, ya que cuenta con las características necesarias para el diseño y modelado del sistema a implementar. Además, en la Universidad se cuenta con la licencia para su uso y tiene la característica de ser multiplataforma.

- **Entorno de desarrollo PyCharm v4.0.2**

PyCharm es un IDE (Entorno de desarrollo integrado) desarrollado por la compañía Jetbrains, está basado en IntelliJ IDEA[19], el IDE de la misma compañía pero enfocado hacia Java y la base de Android Studio. Pycharm tiene cientos de funciones, por lo que puede verse como una herramienta muy pesada, pero muy útil, ya que ayuda con el desarrollo. Es multiplataforma, hay binarios para: Windows, Linux y Mac OS X. Proporciona análisis de código, depuración gráfica, integración con VCS (Sistema de Control de Versiones)/DVCS (Sistema de Control de Versiones Distribuidos) y soporte para el desarrollo web con Django (Turnquist, 2011). Se decidió escoger este entorno de desarrollo, ya que además de las ventajas y facilidades que presenta (mencionadas anteriormente), soporta el lenguaje de programación Python, el cual fue seleccionado para llevar a cabo el desarrollo de la aplicación web, descrito en el subepígrafe anterior.

- **Bootstrap v3.0**

Bootstrap es un framework CSS de código abierto que proporciona diseños y plugins jQuery (Niska, 2014). Incluye numerosos componentes CSS listos para utilizar y que cubren las necesidades más habituales de los diseños actuales para la web. Incluye 180 íconos creados mediante una fuente especial llamada Glyphicon Halflings. Aunque esta fuente normalmente no es gratuita, su creador permite utilizar estos íconos gratuitamente dentro de Bootstrap. Dentro de estos componentes se encuentran: los menús desplegables, grupo de botones, grupos de campos de formulario (Thornton, y otros, 2015). También se encuentran otros

[16] Systems Modeling Language es un lenguaje de especificación de sistemas, es un subconjunto ampliado de UML 2.0.
[17] Business Process Model and Notation (BPMN), en español Modelo y Notación de Procesos de Negocio)
[18] XMI o XML Metadata Interchange (XML de Intercambio de Metadatos).
[19] Entorno Java de desarrollo integrado

componentes como etiquetas, encabezado de página, imágenes en miniatura, mensajes de alerta, barras de progreso, listas de elementos y paneles que serán útiles para la interfaz de la propuesta de solución a desarrollar. Se utiliza este framework para el diseño de la interfaz del sistema por las características que presenta y las utilidades que brindan sus componentes.

1.7 Técnicas para la captura y validación de los requisitos

La identificación de los requisitos que debe cumplir un software se obtiene a partir de las necesidades del usuario. Es tarea del analista definir estas funcionalidades a partir de diferentes fuentes de información, es por ello que resulta tan importante la captura de dichos requisitos. Para lograr este objetivo, se utilizan diferentes técnicas de captura de requisitos que facilitan la comunicación con los interesados en el producto y así lograr la satisfacción del cliente. Asimismo, existen técnicas de validación de requisitos que demuestran que estos conducirán a funciones que el usuario realmente necesita.

Para la obtención de los requisitos con los que debe cumplir la solución a desarrollar, se llevaron a cabo las siguientes técnicas:

- **Entrevista:** realizada al cliente de forma informal, para lograr obtener información sobre las necesidades que este presenta y cómo desea que las mismas se corrijan.
- **Tormenta de ideas:** permitirá el intercambio de ideas con un grupo de profesionales sobre las herramientas de gestión de repositorios. Este debate permitirá concretar los requisitos identificados en la entrevista realizada al cliente.

Para la validación de los requisitos identificados se utilizará como técnica:

- **La Revisión Técnica Formal (RTF):** una vez identificados los requisitos, los mismos serán entregados al cliente con el objetivo de validar la especificación de los mismos. A partir de esta revisión se detectarán si existen deficiencias, ambigüedades, omisiones y errores, tanto de formato como de contenido en la especificación de los requisitos.

1.8 Conclusiones del capítulo

El desarrollo de este capítulo aportó una serie de elementos a tener en cuenta en el diseño e implementación de la propuesta de solución. Lográndose a partir de la comparación entre diferentes herramientas administrativas existentes que permiten gestionar los repositorios Subversion, las cuales presentan características similares a la que se pretende desarrollar como solución, pero no se adaptan al ambiente productivo de la Universidad. Además, en este capítulo se presentaron las principales tecnologías y

herramientas a utilizar para el desarrollo de la aplicación web SVNAdmin 1.0, así como la metodología que se utilizará para guiar dicho desarrollo.

CAPÍTULO II: Análisis y diseño de la aplicación web para la gestión de repositorios SVN.

Introducción

En este capítulo se presentará el modelo del dominio del entorno actual, para a partir de este, definir los pasos para comenzar con el modelado y diseño de la aplicación web SVNAdmin 1.0. Además se identificarán los requisitos funcionales y no funcionales con los que debe cumplir la aplicación, así como el diseño de las clases de la propuesta de solución.

2.1 Modelo de dominio

Un modelo de dominio captura los tipos más importantes de objetos que existen, o los eventos que suceden en el entorno donde estará el sistema, se identifica conceptos, se definen estos conceptos y se unen o relacionan en un diagrama de clases UML. Estos conceptos definen la realidad en que se desarrolla el negocio, su objetivo es comprender y describir los elementos más importantes dentro del contexto del sistema (Martínez, 2014).

Diagrama del modelo de dominio

La figura 2 muestra el diseño del diagrama del modelo de dominio que representa la relación entre los conceptos fundamentales del entorno donde se desarrollará la propuesta de solución.

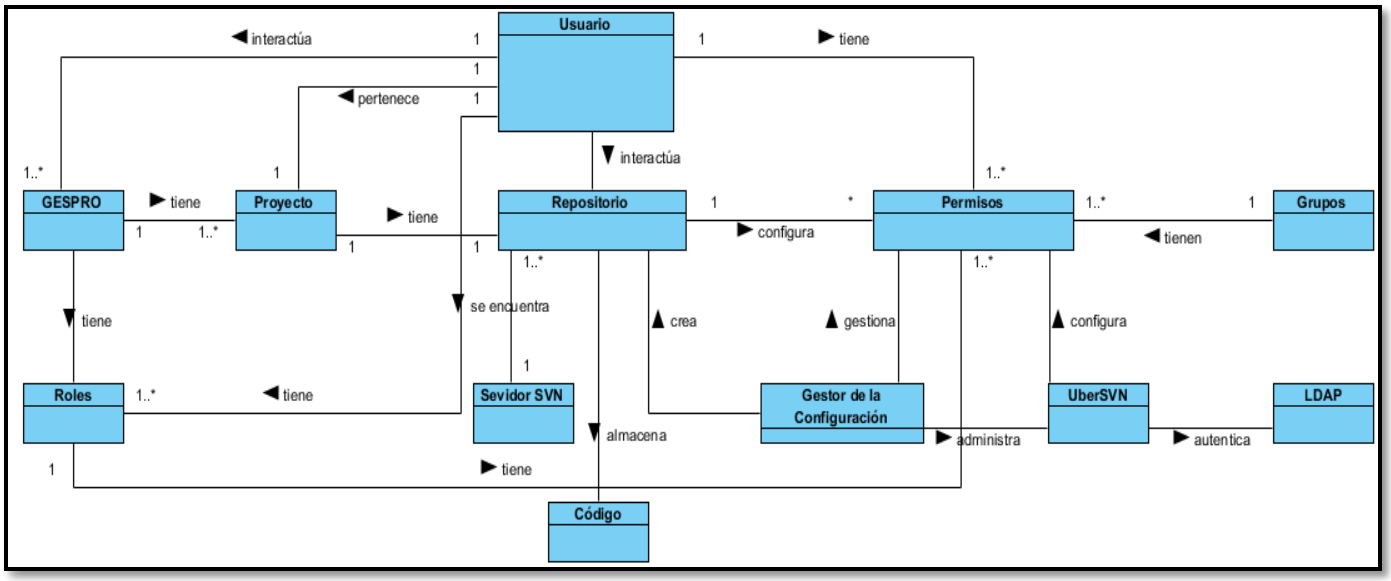

Fig. 2 Modelo de dominio

Descripción del modelo de dominio

El modelo de dominio representa como un usuario interactúa con el GESPRO, herramienta donde se gestionan los proyectos que se crean en cada uno de los centros. Este usuario pertenece a un determinado proyecto donde se le asigna un rol en dependencia del tipo de permiso que se le otorga, lo que le permite interactuar con el repositorio perteneciente a este proyecto. Dicho repositorio acumula el código versionado y se encuentra almacenado en el ServidorSVN. Los permisos sobre los repositorios son gestionados por el gestor de la configuración, encargado de crear dicho repositorio y de administrar la herramienta administrativa UberSVN, donde los usuarios se autentican mediante el servicio de Protocolo Ligero de Acceso a Directorios (LDAP). Estos usuarios pueden estar agrupados o no a grupos dentro de la aplicación.

Conceptos representados en el diagrama

- **Usuario**: es miembro del proyecto e interactúa con el repositorio según el rol que desempeñe.
- **GESPRO:** herramienta donde se gestionan los proyectos.
- **Proyecto:** propuesta de trabajo que se ha creado o se está desarrollando.
- **Repositorio**: sitio donde se almacena toda la información versionada y permite la interacción de diversos usuarios.
- **Permisos:** reglas que definen las acciones que puede realizar el usuario sobre el repositorio.
- **Grupos:** conjunto de usuarios que pertenecen a un mismo proyecto.
- **Roles:** agrupación de responsabilidades que pueden desempeñar los usuarios.
- **UberSVN:** sistema de administración de repositorios SVN usado actualmente en la Universidad.
- **LDAP:** protocolo ligero de acceso a directorios. Usado como directorio donde se almacena información sobre usuarios y grupos.
- **ServidorSVN:** servidor donde se encuentran almacenados todos los repositorios.
- **Gestor de la configuración**: interactúa directamente con la herramienta UberSVN y con la gestión de los permisos a los repositorios.
- **Código:** información que almacenan los repositorios sobre los proyectos.

Una vez analizado el modelo de dominio y definido los conceptos identificados en este, se realizó el modelado del sistema para la aplicación web SVNAdmin 1.0, partiendo de la especificación de los requisitos que la misma debe cumplir.

2.2 Modelado del sistema

A partir de este epígrafe se comienza con el modelado de la solución propuesta. Para ello se identifican tanto los requisitos funcionales como los no funcionales con los que debe cumplir dicha solución y se modelan los procesos a través de las clases de diseño de la misma.

2.2.1 Especificación de los requisitos de la aplicación

Para realizar una completa captura de los requisitos se utilizaron diferentes técnicas que aseguran y organizan la información referente a las funcionalidades con las que debe cumplir la aplicación, para que garantice el seguimiento y la usabilidad en la gestión de repositorios SVN en la red de los centros de desarrollo de la UCI. Entre estas técnicas se encuentra la entrevista, aplicada a los usuarios que se encargan de la gestión de la configuración en los repositorios SVN, con el objetivo de identificar cuáles eran sus principales necesidades. También se llevaron a cabo las tormentas de ideas, que posibilitaron debates con profesionales sobre este tema.

Requisitos Funcionales

Los requisitos funcionales son declaraciones o servicios que el sistema debe proporcionar, cómo el sistema debe reaccionar a entradas particulares y de cómo el sistema debe comportarse en situaciones particulares (Sommerville, 2007). Se identificaron como requisitos funcionales los que a continuación se presentan:

RF1: Adicionar usuario_aplicación
RF2: Eliminar usuario_aplicación
RF3: Asignar usuario_aplicación a grupo_aplicación
RF4: Listar usuario_aplicación
RF5: Adicionar usuario_repositorio
RF6: Eliminar usuario_repositorio
RF7: Asignar usuario_repositorio a grupo_repositorio
RF8: Listar usuario_repositorio
RF9: Adicionar grupo_aplicación
RF10: Eliminar grupo_aplicación
RF11: Listar grupo_aplicación
RF12: Adicionar grupo_ repositorio

RF13: Eliminar grupo_ repositorio

RF14: Listar grupo_ repositorio

RF15: Adicionar repositorio

RF16: Eliminar repositorio

RF17: Listar repositorio

RF18: Adicionar permiso

RF19: Eliminar permiso

RF20: Adicionar etiqueta

RF21: Eliminar etiqueta

RF22: Listar etiquetas

RF23: Adicionar reporte

RF24: Eliminar reporte

RF25: Listar reportes

RF26: Exportar reporte

RF27: Actualizar propietario de repositorio.

RF28: Autenticar usuario

La tabla 2 muestra la descripción del requisito “Adicionar permiso”.

Tabla. 2 Especificación del requisito funcional "Adicionar permiso"

Nº	Nombre	Descripción	Prioridad para el cliente	Complejidad	Referencias cruzadas
RF18	Adicionar permiso	El sistema debe ser capaz de adicionar un permiso a un usuario de repositorio o grupo de repositorio al seleccionar la trama del repositorio.	Alta	Alta	

La descripción de los restantes requisitos especificados se encuentra en el expediente de proyecto en el artefacto de ingeniería “Especificación de los requisitos de software”.

Requisitos no funcionales

Los requisitos no funcionales son las limitaciones en los servicios o funciones que ofrece el sistema. Ellos incluyen restricción de tiempo, las limitaciones en el proceso y el desarrollo de normas (Sommerville, 2007). A continuación se caracterizan brevemente los requisitos no funcionales identificados para el sistema divididos en diferentes categorías.

- **Usabilidad**

RNF1: Los mensajes de error del sistema deben incluir una descripción textual del error.

RNF2: El usuario no tiene que conocer la sintaxis de Subversion para asignar permisos en el repositorio.

RNF3: Las etiquetas de cada funcionalidad y los campos de cada interfaz deben tener títulos asociados a su función de negocio.

RNF4: El sistema debe tener una interfaz minimalista y orientada a usuarios especialistas en las Tecnologías de la Informática.

- **Interoperabilidad**

RNF5: Permitir la comunicación con otros sistemas de la producción a través de servicios.

- **Portabilidad**

RNF6: El despliegue del sistema debe estar diseñado para sistemas GNU/Linux.

RNF7: La solución debe garantizar la coexistencia con sistemas SVN.

- **Confiabilidad**

RFN8: El usuario que desee interactuar con el sistema debe autenticarse para poder realizar las funcionalidades que el mismo propone, de acuerdo con el rol que le sea asignado y los permisos que dicho rol permita.

- **Validación de información**

RNF9: El sistema debe validar automáticamente la información contenida en los campos de ingreso. En el proceso de validación de la información, se deben tener en cuenta aspectos tales como obligatoriedad de campos, longitud de caracteres permitida por campo y manejo de tipos de datos.

- **Requisitos del Software**

Las estaciones de trabajo del usuario final y servidores deberán tener como requisitos mínimos de software:

Servidores:

RNF10: Sistema Operativo GNU/Linux.

RNF11: Servidor Web Apache 2.0 o superior.

RNF12: Subversion (versión 1.7 o superior).

Cliente:

RNF13: Navegador Mozilla Firefox (versión 32 o superior), Chrome (versión 36 o superior y versiones para móviles).

- **Requisitos del Hardware**

Servidor:

RNF14: 512 de MB de memoria RAM (con 50 usuarios).

RNF15: Para la instalación 200 MB como mínimo de disco duro.

RNF16: Pentium 4 o superior, con velocidad de microprocesador a 2.4 GHz como mínimo.

Clientes:

RNF17: Al menos 256 MB de memoria RAM.

RNF18: Pentium 4 o superior, con velocidad de microprocesador a 512 MHz como mínimo.

Para agrupar los requisitos funcionales se definió el modelo de casos de uso, el cual se describe en el subepígrafe que se muestra a continuación.

2.2.2 Modelo de casos de uso

El modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema. Un caso de uso (CU) representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema. Cada caso de uso tiene una descripción que describe la funcionalidad que se construirá en el sistema propuesto (Ceria, 2001).

Actores del sistema

Un actor es un usuario del sistema. Incluye usuarios humanos y otros sistemas computarizados. Un actor usa un caso de uso para desempeñar alguna porción de trabajo que es de valor para el negocio. El conjunto de casos de uso al que un actor tiene acceso define su rol global en el sistema y el alcance de su acción (Ceria, 2001). La tabla 3 muestra los actores del sistema.

Tabla. 3 Actores del sistema

Actor	Descripción
Administrador Aplicación	Es el encargado de administrar la herramienta a nivel central. Este es el que administra los repositorios, crea los usuarios para el sistema y gestiona todos los posibles cambios que se deseen realizar en la herramienta SVNAdmin 1.0.

Administrador Repositorio	Es el rol que se le asigna a un integrante del proyecto que se encarga de asignar los permisos a los usuarios para que interactúen con la herramienta, también se encarga de gestionar repositorios.

Diagrama de casos de uso del sistema

Los requisitos funcionales identificados en el subepígrafe 2.2.1 fueron agrupados en 11 casos de uso:

- Gestionar usuario_aplicación
- Gestionar usuario_repositorio
- Gestionar grupo_aplicación
- Gestionar grupo_ repositorio
- Gestionar repositorio
- Gestionar permisos
- Gestionar etiquetas
- Gestionar reportes
- Exportar reporte
- Actualizar propietario de repositorio.
- Autenticar usuario

La figura 3 muestra el diagrama de casos de uso representado por los actores identificados anteriormente.

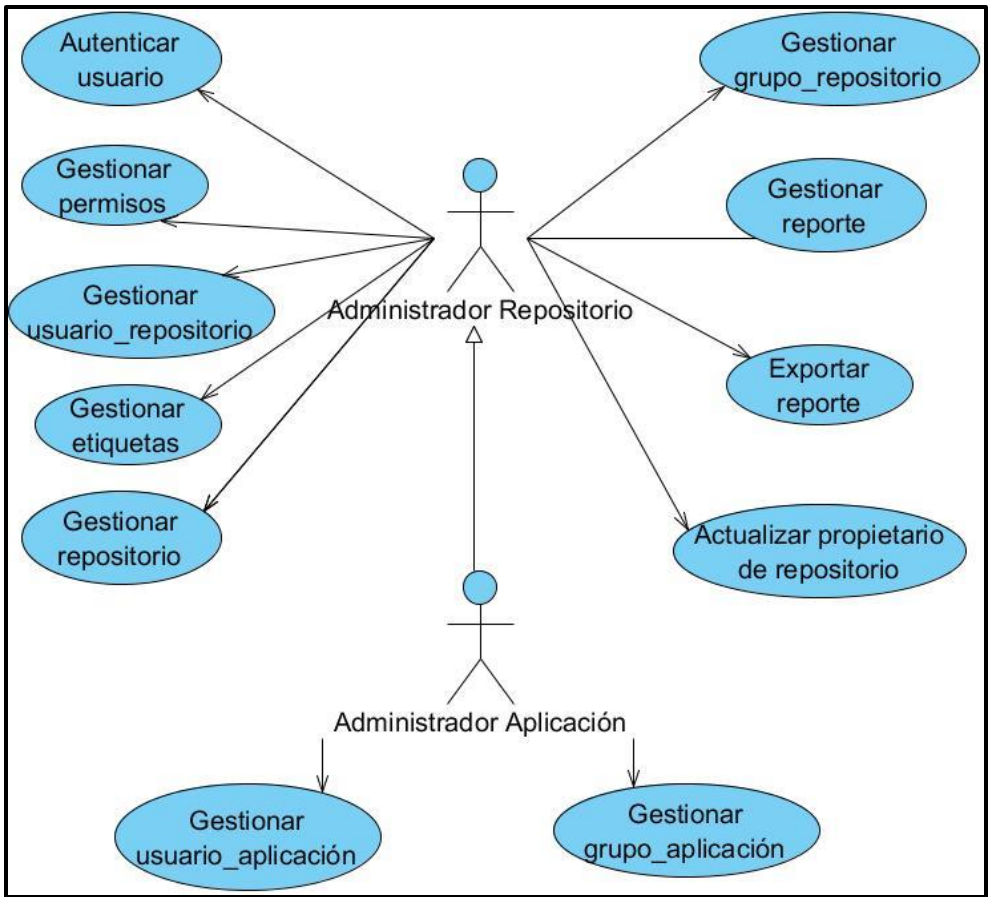

Fig. 3 Diagrama de casos de uso del sistema

Descripción textual del caso de uso del sistema

Una vez modelado el diagrama de casos de uso del sistema se describieron cada uno de los casos de uso que integran dicho diagrama. Para ver estas descripciones consultar el artefacto de Ingeniería del expediente de proyecto "Especificación de los casos de uso". La tabla 4 muestra la descripción del caso de uso "Gestionar permisos".

Tabla. 4 Descripción del caso de uso "Gestionar permisos"

Objetivo	Gestionar un permiso
Actores	Administrador Repositorio, Administrador Aplicación ((Inicia) Adicionar y Eliminar un permiso.)

Resumen	El caso de uso comienza al seleccionar la opción de adicionar un permiso para un usuario o "Eliminar" uno existente, permite adicionar un nuevo permiso para un usuario. Además brinda la opción de eliminar un permiso que tiene un usuario o grupo. Finaliza al guardar el permiso adicionado o al eliminar uno existente.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Para eliminar un permiso debe haber al menos un permiso existente y un usuario.
Postcondiciones	Es gestionado un permiso

Flujo de eventos

Flujo básico "Acciones del Sistema (**Adicionar Permiso**)"

	Actor	**Sistema**
1		Provee una interfaz llamada **"Listar Repositorios"** que presenta la opción de **"Editar/Ver Permisos"** al lado de cada repositorio, inicializado siempre por el actor. Permite realizar varias acciones con un permiso una vez que se selecciona esta opción, como: ➢ Adicionar un permiso. ➢ Eliminar un permiso. *Ver* ***Sección 1****: **"Eliminar* **un permiso***".*
2	Se selecciona la ruta a la que se desea adicionar nuevos permisos.	
3		El sistema muestra los usuarios que tienen permisos sobre la ruta seleccionada en el explorador.
4	Selecciona un nuevo usuario para el que se le aplicará la regla de permiso, selecciona el permiso que desea agregarle a este usuario y en el formulario de confirmación da clic en la opción "Guardar" en caso contrario **ver flujo alterno 1.**	

5		El sistema muestra un mensaje de color verde anunciando que se ha guardado satisfactoriamente la acción en caso contrario **ver flujo alterno 2.**
Flujo alterno 1 (al paso 4)		
Nº Evento "No selecciona la opción Guardar**".**		
	Actor	**Sistema**
1	Selecciona la opción "Cancelar".	
2		El sistema ignora la petición.
Flujo alterno 2 (al paso 5)		
Nº Evento "Errores al adicionar permiso**".**		
	Actor	**Sistema**
1		El sistema muestra un mensaje de color rojo anunciando que ha ocurrido un error al adicionar el permiso.
Sección 1: "Eliminar permiso"		
Flujo básico "Eliminar permiso**"**		
	Actor	**Sistema**
1	Selecciona la ruta a la que desea actualizar los permisos de los usuarios.	
2		El sistema muestra los usuarios y sus permisos sobre la ruta especificada.
3	Da clic en el ícono de eliminar sobre el permiso del usuario que desee eliminar y luego en el formulario de confirmación da clic en la opción "Guardar", en caso contrario **ver flujo alterno 1.**	
4		El sistema muestra un mensaje anunciando que se ha guardado satisfactoriamente la acción, en caso contrario **ver flujo alterno 2.**
Flujo alterno 1 (al paso 3)		
Nº Evento "No selecciona la opción Guardar**"**		
	Actor	**Sistema**
1	Selecciona la opción "Cancelar".	
2		El sistema ignora la petición.
Flujo alterno 2 (al paso 4)		

Nº Evento "Errores al eliminar permiso"**.**		
	Actor	**Sistema**
1		El sistema muestra un mensaje anunciando que ha ocurrido un error al eliminar el permiso.
Termina el caso de uso		
Relaciones	CU incluidos	N/A
	CU extendidos	N/A
Requisitos no funcionales	N/A	
Asuntos pendientes	N/A	

Para determinar si los casos de uso agrupan todos los requisitos funcionales se utilizó la técnica de matriz de trazabilidad.

2.2.3 Matriz de trazabilidad (RF/CU)

La matriz de trazabilidad es una técnica que consiste en relacionar los requisitos funcionales con los casos de uso, lo que permite determinar qué requisito está agrupado por un caso de uso determinado. La tabla 5 muestra la técnica de matriz de trazabilidad aplicada sobre los requisitos funcionales del sistema.

Tabla. 5 Matriz de trazabilidad (RF/CU)

RF/CU	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11
RF1	X										
RF2	X										
RF3	X										
RF4	X										
RF5		X									
RF6		X									
RF7		X									
RF8		X									
RF9			X								
RF10			X								
RF11			X								

RF12				X							
RF13				X							
RF14				X							
RF15					X						
RF16					X						
RF17					X						
RF18						X					
RF19						X					
RF20							X				
RF21							X				
RF22							X				
RF23								X			
RF24								X			
RF25								X			
RF26									X		
RF27										X	
RF28											X

Una vez aplicada esta técnica, se estableció como conclusión que los casos de uso del sistema satisfacen los requisitos funcionales identificados. De esta forma se procedió a realizar los diagramas de las clases del diseño para cada uno de los casos de uso que integran el sistema.

2.2.4 Diagrama de Clases del Diseño

El diagrama de clases del diseño constituye la representación de las clases del sistema, la estructura y organización de las mismas (Pressman, 2005). La arquitectura utilizada para el modelado del sistema, está basada en MVT, según define Django como framework de desarrollo especificado en el capítulo 1. Esto posibilita la organización del sistema en paquetes individuales, sin dependencias entre el modelo y las plantillas. A continuación se presenta una breve descripción de esta arquitectura representada además mediante el diagrama de paquetes del sistema, el cual se especifica más adelante.

Modelo Vista Template (MVT)

El patrón de diseño MVT, es una especificación del patrón Modelo Vista Controlador (MVC), donde (Alchin, 2013) :

- **M** significa **Model (Modelo)**, la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- **T** significa **Template (Plantilla)**, la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: cómo algunas cosas son mostradas sobre una página web u otro tipo de documento.
- **V** significa **View (Vista)**, la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada, constituye un puente entre los modelos y las plantillas.

Diagrama de Paquetes del Sistema

El diagrama de paquetes se diseña con el objetivo de mostrar la organización que presentan los paquetes con los que se trabaja en el desarrollo de la aplicación poniendo en evidencia la utilización de la arquitectura mencionada anteriormente. Este diagrama contiene un total de 4 paquetes. A continuación se presenta como quedó conformado el diseño del diagrama de paquetes de la aplicación.

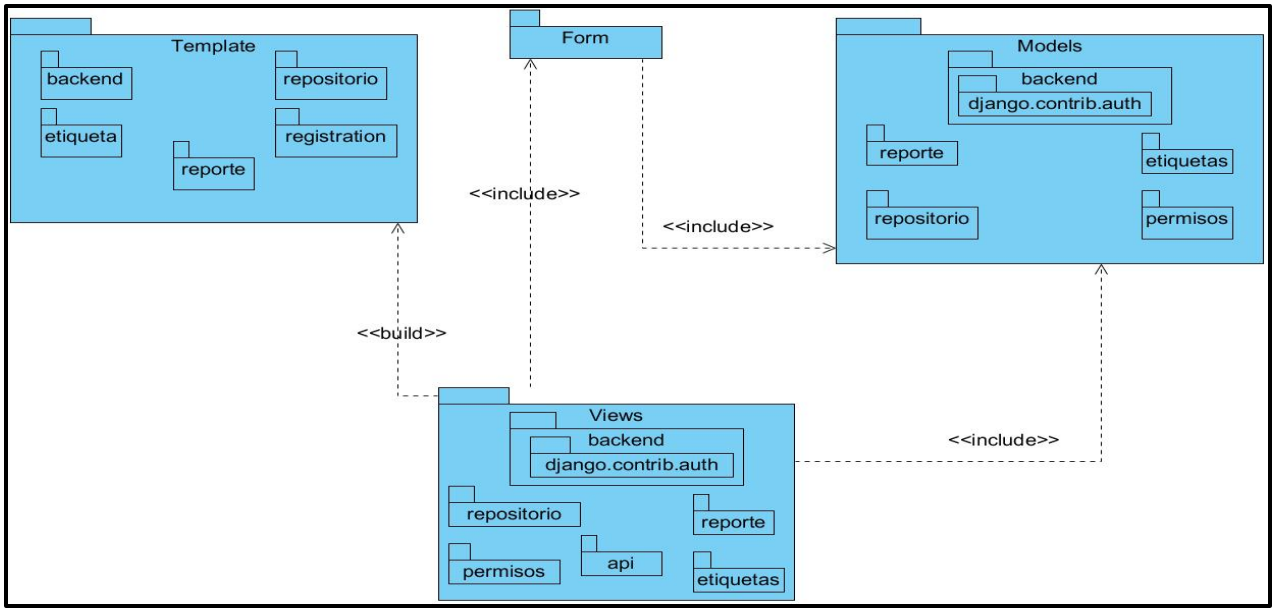

Fig. 4 Diagrama de Paquetes del Sistema

Descripción del diagrama de Paquetes del Sistema

El diagrama representa el paquete *View*, el cual funciona como intermediario entre los paquetes *Template* y *Models*, contiene la lógica del negocio y todos los métodos a ejecutar en la aplicación. Este interactúa con el paquete *Form* para introducir datos al modelo y valerse del mismo en la construcción del *Template*. Por su parte, el *Models* representa la capa de acceso a datos y se utiliza para crear las tablas de la base de datos y el trabajo con las mismas, permitiendo realizar consultas y gestionar los datos existentes. El paquete *Template* contiene las vistas que verá el usuario, siendo esta la capa de presentación para la construcción de *Template*. Los datos o *Form* son enviados desde la Views.

Paquetes representados en el diagrama

- **Views:** este paquete contiene la lógica del negocio, contiene los métodos de la aplicación.
- **Template:** es la capa de presentación de la aplicación, contiene las plantillas que se construirán con la views.
- **Form:** contiene los formularios de algunos de los modelos como etiquetas.
- **Models:** este paquete es la capa de acceso a datos, define la estructura de las tablas y permite el trabajo con las mismas.
- **Backend:** este paquete se encarga de la autenticación y la gestión de usuarios y grupos.
- **Repositorio:** es el paquete que define las funcionalidades para la gestión de repositorios y servidores.
- **Permisos:** este paquete se encarga de la gestión de permisos sobre repositorios de usuarios y grupos. Los permisos pueden ser: lectura, escritura y ninguno.
- **Etiquetas:** en este paquete se definen las funcionalidades relacionadas con las etiquetas que presentará el sistema.
- **Reporte:** en este paquete se definen las funcionalidades relacionadas con los reportes que el sistema permitirá gestionar.
- **Api:** en este paquete se definen los servicios que brindará el sistema.
- **Django.contrib.auth:** representa el paquete que tiene el modelo de usuarios y grupos del sistema.
- **Registration:** representa los formularios de autenticación y cierre de sesión.

Diagrama de clases del diseño del caso de uso Gestionar Permisos

A continuación se muestra el diagrama de clases del diseño del caso de uso Gestionar Permisos (figura 5), el cual muestra las relaciones existentes entre las clases más importantes para este caso de uso. Los diagramas de clases del diseño de los restantes casos de uso se encuentran en el expediente de proyecto, en el artefacto "Modelo del diseño".

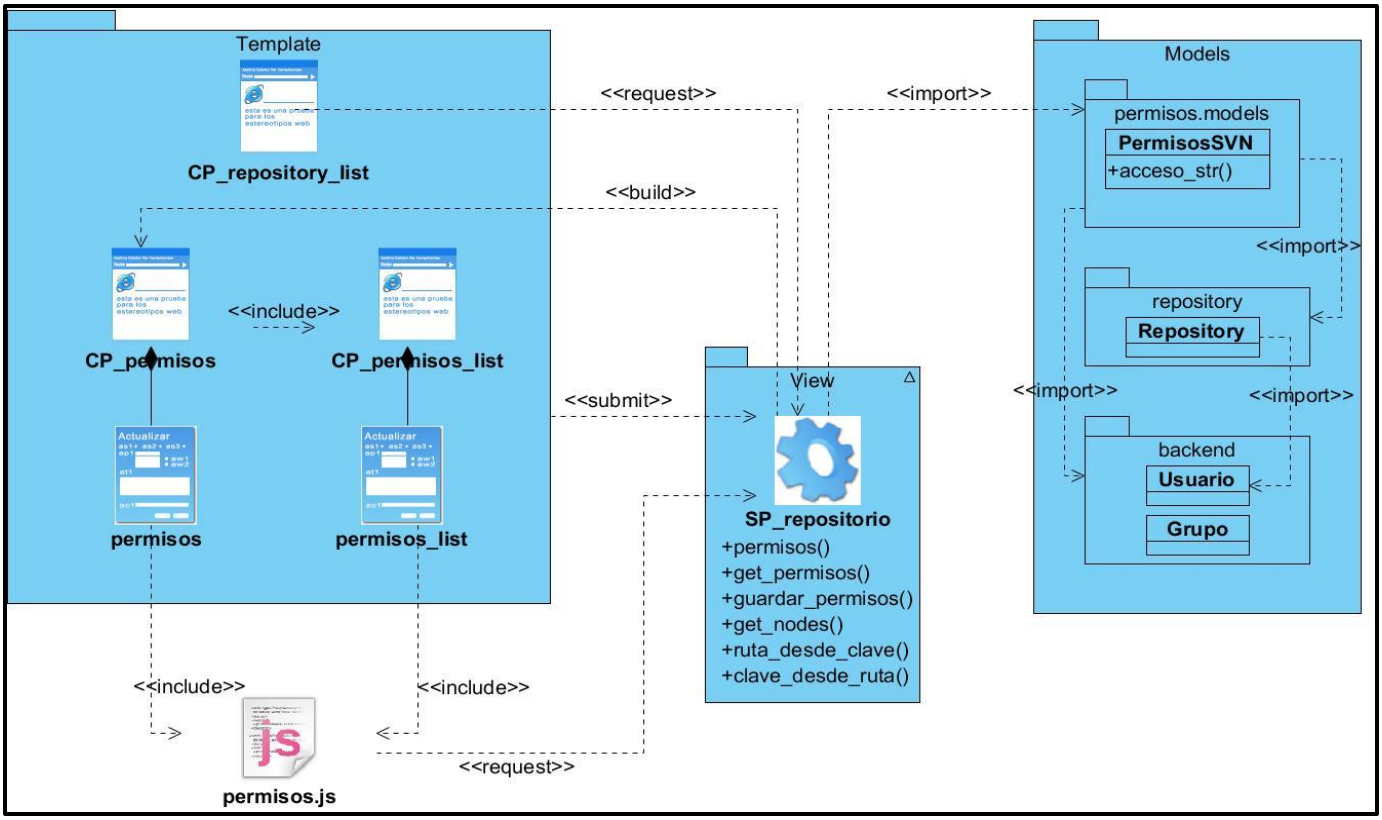

Fig. 5 Diagrama de clases del diseño del CU Gestionar Permisos

Descripción del diagrama de clases del diseño del caso de uso Gestionar Permisos

El caso de uso Gestionar Permisos comienza cuando el usuario selecciona la opción gestionar permisos de uno de los repositorios listados en la *cp_repository_list*. Esto envía una petición (id_repositorio) a la sp_repositorio, que se encarga de consultar el modelo *PermisosSVN* y obtener todos los permisos del repositorio seleccionado. Construye la *cp_permisos* y automáticamente se ejecuta un método de *permisos.js*, que realiza una petición a la *sp_repositorio* y utilizando la clase *util.py* construye el árbol de directorios del repositorio seleccionado enviándolo de nuevo a la *cp_permisos* donde se visualizará, la

misma contiene a la *cp_permisos_list* donde se encuentran todos los permisos del directorio seleccionado así como los que están preparados para actualizarse. De esta lista se puede eliminar un permiso seleccionando la opción eliminar de uno de ellos o, añadir un nuevo permiso seleccionando un usuario o grupo y diciéndole qué permiso tendrá. Cuando el usuario selecciona la opción guardar esto envía los datos a la *sp_repositorio* que utilizando el modelo *PermisosSVN* actualiza los permisos sobre el repositorio y luego actualiza el fichero de configuración del repositorio a través de los datos del modelo con la ayuda de la clase *util.py*.

2.2.5 Modelo de datos

A partir de las clases que se representan en el modelo se puede diseñar el modelo de datos, el cual permite relacionar de forma estructural las tablas que persisten en una base de datos, atendiendo a su tipo y las condiciones que deben cumplir para reflejar la realidad deseada.

Las clases representadas que pertenecen al modelo, son mapeadas a través del ORM de Django, generando usualmente una tabla de la base de datos por cada modelo. Por las características del ORM de Django, la gestión de la base de datos es independiente del gestor, en este caso se utilizó PostgresSQL por ser el RDBMS[20] más utilizado en la gestión de los servicios de la producción en la UCI. En la figura 6 se muestra el modelo de datos que además de las clases del modelo, muestra las tablas definidas por la gestión de la seguridad en la aplicación.

[20] Relational Database Management System o RDBMS - Sistema de Gestión de Base de Datos Relacional o SGBDR

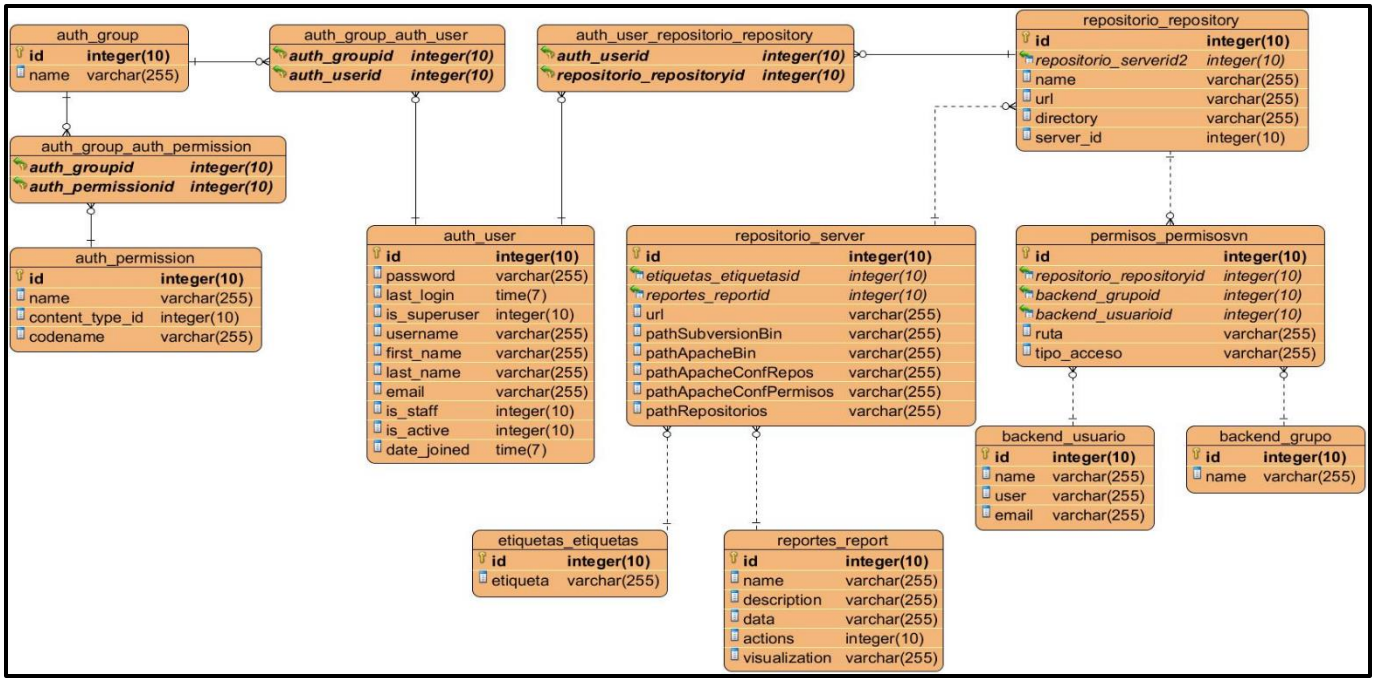

Fig. 6 Modelo de datos del sistema

2.2.6 Patrones utilizados en el diseño de la aplicación

Los patrones de diseño son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto. Estos a su vez identifican las clases, instancias, roles, colaboraciones y la distribución de responsabilidades (Gamma, 2008). Entre los patrones de diseño utilizados se encuentran los patrones de asignación de responsabilidades (GRASP[21]), los cuales representan los principios básicos de la asignación de responsabilidades a objetos, posibilitando mejorar su diseño. También se utilizan los patrones de la Banda de los Cuatro (GoF[22]), que permiten ampliar el lenguaje y aprender nuevos estilos para solucionar problemas comunes en el diseño orientado a objetos.

Patrones GRASP utilizados:

- **Bajo acoplamiento:** las clases del sistema cuentan con un bajo acoplamiento ya que existe poca dependencia entre ellas, permitiendo que no se afecte la aplicación completa cuando se modifique una parte de esta. Es en el modelo donde únicamente puede existir una dependencia entre las clases del sistema, pero esto no representa una gran jerarquía.

[21] General Responsibility Assignment Softw are Patterns (Patrones Generales de Asignación de Responsabilidades de Softw are)
[22] Gang of Four (Banda de los Cuatro)

- **Controlador:** este patrón se evidencia en la clase *views.py* contenida dentro de cada paquete de la aplicación, la cual funciona como una controladora que atiende los distintos eventos del sistema. Esta clase recibe información proveniente de la *Models* y decide qué datos enviar al *Template* para su posterior organización e interacción con el usuario, además de decidir qué funcionalidad se va a ejecutar para dar respuesta a determinada petición.

Patrones Gof utilizados:

- **Decorador:** este patrón posibilita añadir responsabilidades adicionales a un objeto de forma dinámica. El sistema cuenta con la clase *django.contrib.auth.decorators* que contiene todas las funcionalidades adicionales o decoradores que realizarán determinada acción en correspondencia con el objeto ejecutado y sus entradas. Un ejemplo del uso de este patrón, lo constituye el método *create_repository* de la clase *views.py*, que tiene la responsabilidad de crear un repositorio y en caso de no existir un servidor, redirecciona a la página inicial para crearlo, además se cuenta con el decorador "*@permission_required*" que añade la responsabilidad de comprobar si el usuario tiene permisos para adicionar este repositorio.

- **Fachada:** cuando un usuario accede a un repositorio y realiza determinada acción sobre el mismo, dicha acción provoca la ejecución de uno o varios servicios que se encargan de manipular los elementos con los que interactúa el usuario en la interfaz o fachada principal, evidenciándose de esta forma el patrón de diseño Fachada.

Fig. 7 Patrón Fachada

2.3 Conclusiones del capítulo

El desarrollo de este capítulo permitió la elaboración de un conjunto de artefactos que contribuyeron a llegar a una mejor comprensión del sistema a implementar, así como las condiciones específicas con las que este debe cumplir. Se detallaron con exactitud los requisitos funcionales y no funcionales del sistema, los actores que intervienen así como los casos de uso. Además se realizó el diseño del diagrama de clases, donde se utilizaron los diferentes patrones identificados. De esta forma se está en condiciones de comenzar con la implementación de la aplicación web SVNAdmin 1.0, la cual debe estar guiada por los requisitos y casos de uso descritos y presentados en este capítulo.

CAPÍTULO III: Implementación y pruebas de la aplicación web para la gestión de repositorios SVN.

Introducción

En el presente capítulo se muestran los resultados alcanzados en la implementación del sistema, como los componentes o funciones reutilizables, estándares de codificación utilizados y algunas interfaces de la aplicación. Para esto se especificará el diagrama de componentes como un artefacto principal para mostrar el encapsulamiento de la implementación. Por último se describirán las pruebas realizadas al software y los resultados obtenidos una vez aplicada las mismas.

3.1 Implementación

Entre las disciplinas propuestas por la metodología de desarrollo seleccionada, AUP-UCI, se encuentra la de Implementación. En esta se comienza con el resultado obtenido por el diseño y se procede a la construcción del sistema. El objetivo de esta disciplina es lograr la implementación de las clases y subsistemas que fueron encontrados durante el diseño, así como definir la organización del código.

3.1.1 Modelo de implementación

El Modelo de Implementación consiste en la colección de componentes y subsistemas de implementación, donde los componentes son generalmente ficheros ejecutables, ficheros de código fuente, entre otros (Pressman, 2005). Para estos, se utiliza el diagrama de componentes, el cual muestra las dependencias entre las partes del código del sistema.

Diagrama de componentes

El término de componente se define como una parte modular, desplegable y reemplazable de un sistema que encapsula implementación y expone un conjunto de interfaces (Pressman, 2005). Un diagrama de componentes es utilizado para estructurar el modelo de implementación en términos de subsistemas de implementación para de esta manera mostrar las dependencias entre los mismos. Dentro de estos componentes se pueden encontrar archivos, bibliotecas compartidas, módulos, ejecutables y paquetes. La figura 8 muestra el diagrama de componentes para el caso de uso “Gestionar Permisos”.

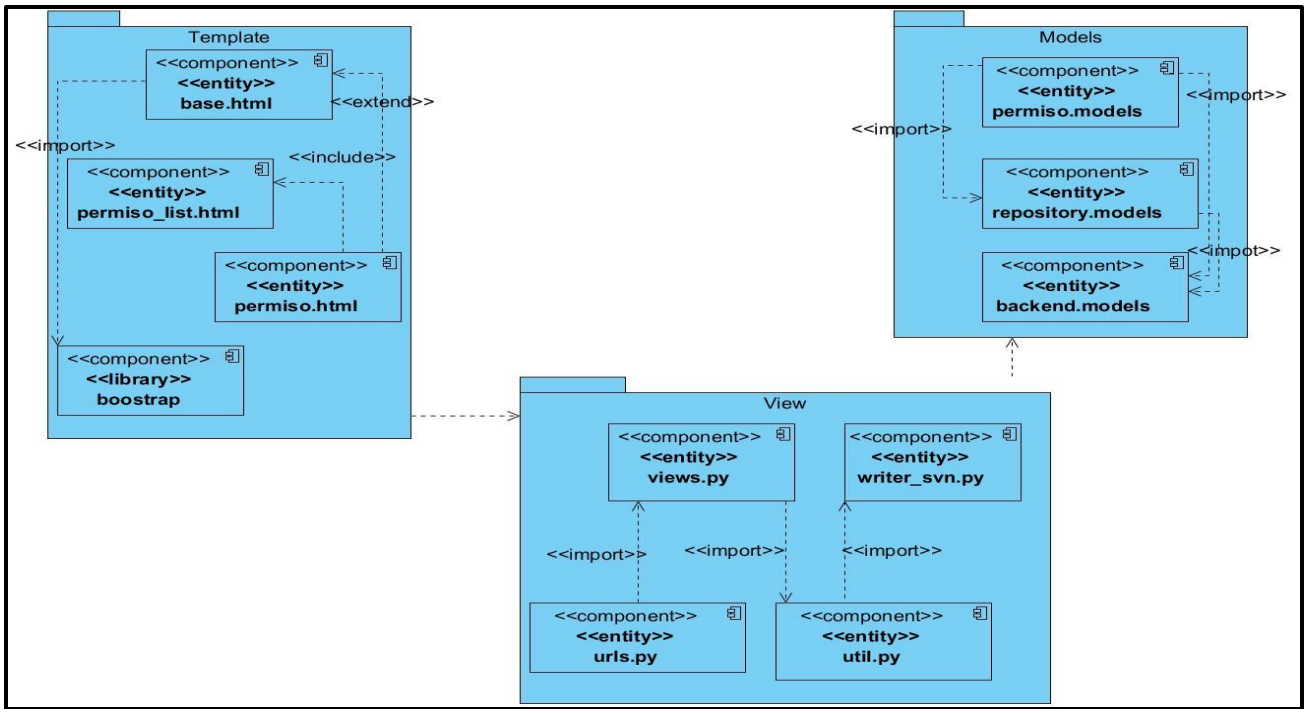

Fig. 8 Diagrama de Componentes CU Gestionar Permisos

Descripción del diagrama de componentes para el caso de uso Gestionar Permisos

El componente de url.py contiene los métodos del *view.py* que se ejecutan según una *url* específica. *View.py* utiliza *permisos.models* para la gestión de permisos a nivel de base de datos, mientras que *permisos.models* necesita de *backend.models* y *repositorio.models* para su construcción. Además *views.py* para ejecutar sus métodos se apoya en *writer_svn.py*, *util.py* que utiliza a *writer_svn.py* para escribir en los ficheros de configuración y es la encargada de construir a *permisos.html*. Esta a su vez importa a *base.html* e incluye a *permisos_list.html* para su completa construcción y visualización por el usuario. El componente *base.html* contiene lo general de los *Template*, lo que no cambia de uno a otros, por lo que se encarga de importar la librería de *boostrap* utilizada para la visualización del sistema.

Componentes representados en el diagrama

- **base.html**: template que contiene la estructura básica de todos los templates.
- **permiso.html**: template que contiene un árbol de directorio y los formularios necesarios para la gestión de permisos sobre los repositorios.
- **permiso_list.html**: template que contiene la lista de permisos sobre un directorio.

- **boostrap**: librería CSS.
- **views.py**: clase que contiene los métodos para la gestión de permisos.
- **urls.py**: clase que contiene las urls y los métodos asignados a las mismas.
- **util.py**: clase auxiliar para el trabajo con repositorio en Subversion.
- **writer_SVN.py**: clase auxiliar utilizado para escribir en los ficheros de configuración de los repositorios Subversion.
- **permiso.models**: contiene el modelo de permisos SVN.
- **repository.models**: contiene los modelos de los repositorios y servidores.
- **backend.models**: contiene los modelos de usuarios y grupos.

3.1.2 Mecanismos de integración

Con el objetivo de integrar el sistema con otros sistemas del entorno relacionados con la producción en la Universidad como el GESPRO, se implementaron una serie de servicios que permitirán esta integración, los mismos son utilizados mediante la API[23], biblioteca que se encarga del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a los servicios. La tabla 6 muestra la descripción de estos servicios.

Tabla. 6 Descripción de los Servicios

Servicios	Descripción
Detalle del usuario	Obtiene información de un usuario específico, el mismo puede ser modificado y eliminado.
Listar usuarios	Lista todos los usuarios que existen en el sistema y permite insertar nuevos usuarios.
Detalle del grupo	Obtiene los datos de un grupo específico.
Listar grupos	Lista todos los grupos que existen en el sistema mostrando sus datos, y permite modificarlos.
Detalle del repositorio	Detalla un repositorio específico del sistema.
Listar repositorios	Lista los repositorios que se encuentran en el sistema y permite insertar nuevos repositorios.
Detalle del permiso	Permite obtener los permisos que presenta un usuario o grupo en el sistema.
Listar permisos	Lista los permisos que existen en el sistema y seleccionar el que se desee modificar.

[23]API: Interfaz de Programación de Aplicaciones

3.1.3 Componentes o funciones reutilizables

Los componentes o funciones reutilizables representan aquellos paquetes de terceros o bibliotecas que se incluyeron y fueron utilizados para la implementación del sistema. Además, ejecutan un proceso de transformación de datos o cálculos y escriben datos de salidas a partir de datos de entradas (Pressman, 2005). Dentro de estos componentes se encuentran:

- **Django rest framework:** representa un framework de Django que brinda servicios a través de la creación de un API REST, permitiendo de esta manera ejecutar acciones desde el navegador web, además permite implementar y probar los servicios web que se hayan liberado.
- **Reportlab:** representa una biblioteca que permite crear archivos en formato PDF y exportarlos, por lo que es utilizado en el sistema para el trabajo con los reportes.

3.1.4 Estándares de codificación

Generalmente los lenguajes de programación poseen estándares de codificación, posibilitando de esta manera una mejor organización del código generado, lo cual logra una alta comunicación entre los programadores. Los estándares de codificación brindan mayor coherencia del código, permitiendo que pueda ser manipulado o modificado en el futuro por otro equipo de trabajo, ahorrándose la tarea de tener que ser reescrito el mismo por falta de claridad. Para la implementación de la aplicación propuesta se utilizó como estándar de codificación Pep8[24], el cual describe las pautas del código para una implementación escrita en Python. A continuación se presentan algunas muestras del estándar de codificación empleado durante la implementación de la aplicación.

Diseño del código:

- A la hora de comenzar a programar, las líneas de continuación están alineadas verticalmente con el caracter que se ha utilizado, como el paréntesis (), representando la indexación del código. Ejemplo:

```
urlpatterns = patterns('',
                       url(r'^create/$', create_permisosvn),
                       url(r'^mod/(?P<id_permisosvn>\d+)/$', mod_permisosvn),
                       url(r'^delete/(?P<id_permisosvn>\d+)/$', delete_permisosvn)
                       )
```

Fig. 9 Diseño del código

Líneas en blanco:

[24] Pep8: Guía de estilo para la programación en Python (http://recursospython.com/pep8es.pdf)

- Se separan las funciones de alto nivel y definiciones de clases con dos líneas. Ejemplo:

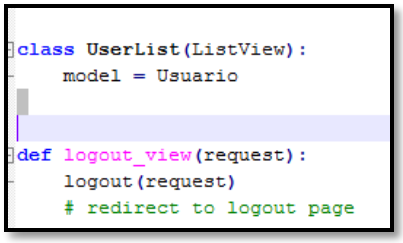

Fig. 10 Líneas en blanco para separar funciones

- Las definiciones de los métodos dentro de una clase son separadas por una línea en blanco. Ejemplo:

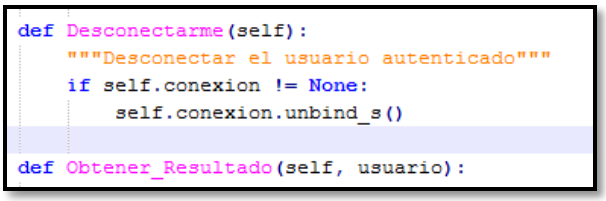

Fig. 11 Líneas en blanco entre las definiciones de los métodos

Importaciones:

- Las importaciones siempre estarán colocadas al comienzo del archivo.
- Las importaciones están agrupadas por:
  1. importaciones de biblioteca estándar
  2. importaciones terceras relacionadas
  3. importaciones locales de la aplicación.

Estilos de nombramiento:

- Se utiliza para el nombramiento de funciones “minúscula_con_guiones_bajos”. Ejemplo:

Fig. 12 Estilo de nombramiento para las funciones

- Las clases están representadas por palabras que comienzan con mayúsculas. Ejemplo:

Fig. 13 Estilo de nombramiento para las clases

Una vez explicado el estándar de codificación utilizado en la implementación, se define el diagrama de despliegue con el objetivo de visualizar dónde se desplegará la aplicación web.

3.2 Diagrama de despliegue

Un diagrama de despliegue permite indicar cómo se ubicarán las funcionalidades y los subsistemas dentro del entorno computacional físico que soportará el software (Pressman, 2005). El diagrama de despliegue diseñado para la aplicación, muestra cómo se distribuyen los servicios que brindará el sistema teniendo en cuenta los requisitos no funcionales de software identificados en epígrafes anteriores, destacándose que la comunicación entre los nodos PC_Cliente y Repositorio Central será mediante el protocolo HTTPS, y la comunicación entre los nodos Repositorio Central y LDAP será mediante el protocolo TCP/IP, tal y como se muestra en la figura 14.

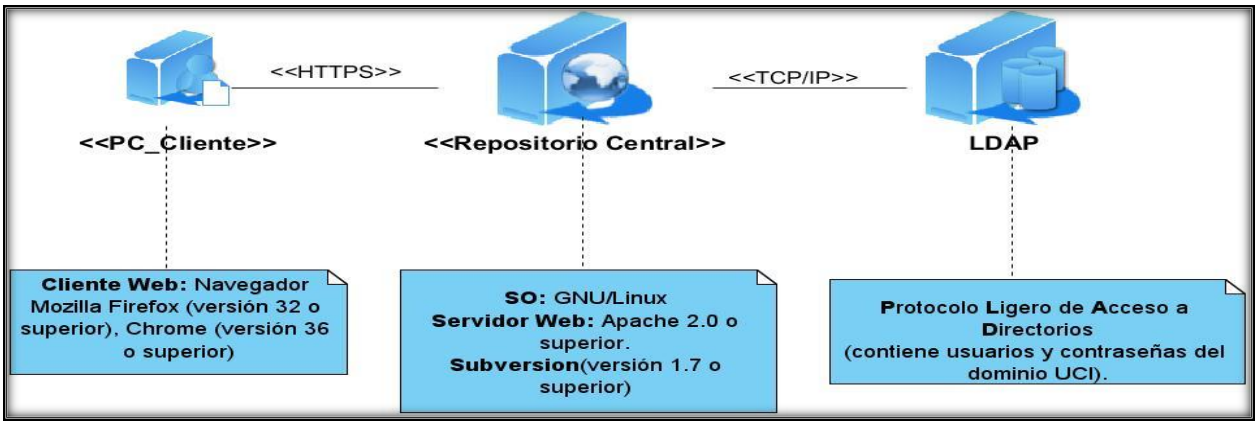

Fig. 14 Diagrama de Despliegue

PC_Cliente: computador en el cual el cliente ejecutará la aplicación a través de un navegador web.

Repositorio Central: servidor donde será montada la aplicación y se encontrará el repositorio de todos los proyectos que se desarrollen en los centros productivos.

LDAP: servicio de Protocolo Ligero de Acceso a Directorios, que contiene los usuarios y contraseñas del dominio UCI, el cual se utilizará para la obtención y autenticación de los usuarios en el sistema.

3.3 Resultados de la implementación

La aplicación implementada, permite a través de las interfaces que presenta, realizar una serie de funciones que cumplen de una forma u otra con los requisitos especificados en capítulos anteriores. Dentro de los

elementos definidos, enfocados a la mejora de la gestión de repositorios SVN se encuentran la usabilidad y seguridad. Se realizó una comparación centrada en estos aspectos entre la solución SVNAdmin 1.0 y UberSVN, cuyos resultados se muestran a continuación.

3.3.1 Usabilidad

El sistema implementado muestra una interfaz minimalista y orientada a usuarios especialistas en las Tecnologías de la Informática. Este requisito se refleja en la figura 15, la cual representa la interfaz de "Gestionar Usuarios y Grupos del Repositorio", donde se visualizan algunos íconos que permitirán realizar diferentes acciones en grupos, usuarios y repositorios.

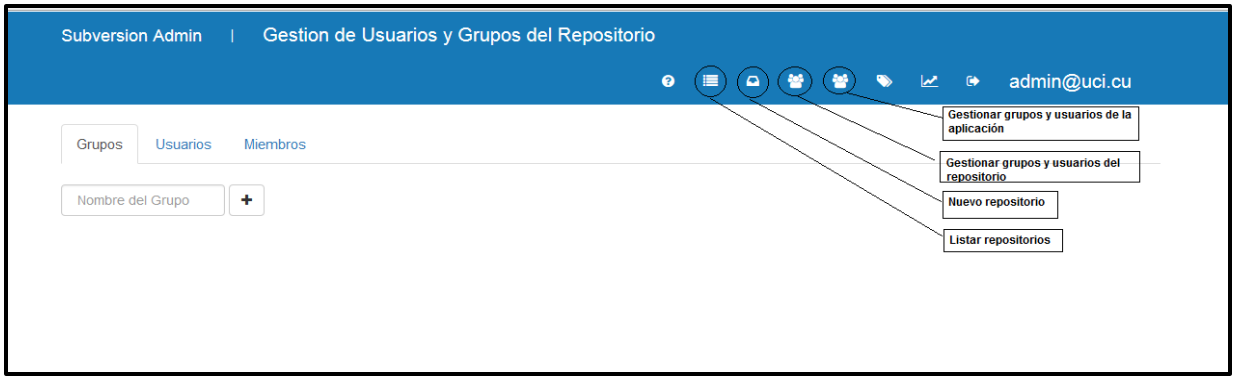

Fig. 15 Interfaz de Gestión de usuarios y grupos e íconos de funciones

Por otra parte, el sistema no requiere que el usuario que trabaje en ella tenga conocimiento alguno sobre la sintaxis de Subversion para gestionar permisos, ya que brinda una interfaz en la que solo tendrá que seleccionar la trama del repositorio a la que quiere agregarle los permisos y asignárselo al usuario que desee, el cual se ejemplifica en la figura 16. Esta función presenta gran ventaja con respecto al trabajo que se realiza en la herramienta administrativa UberSVN, ya que la misma, requiere que el usuario tenga conocimientos sobre la sintaxis antes mencionada, evidenciándose en la figura 17.

Fig. 16 Interfaz para asignar Permisos en SVNAdmin 1.0

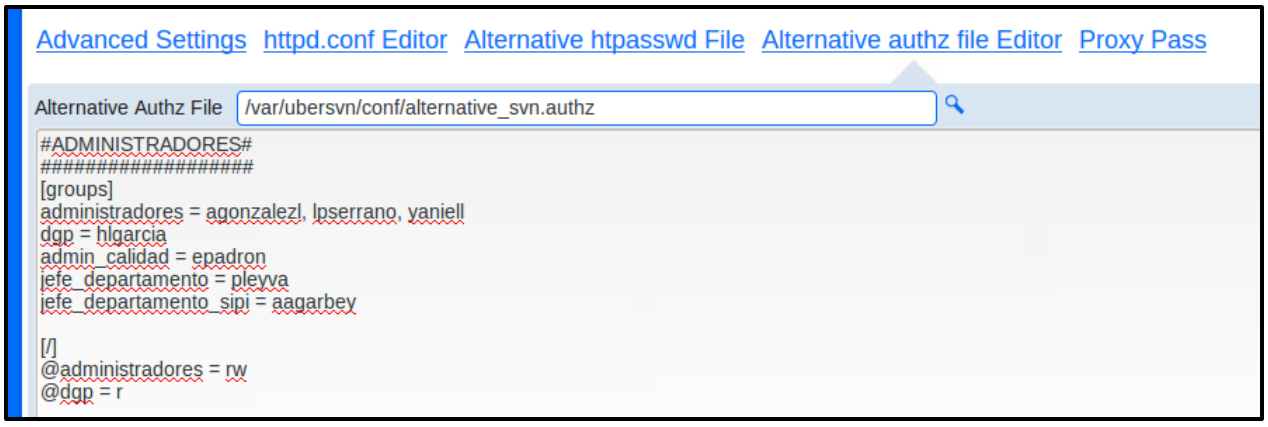

Fig. 17 Interfaz para asignar Permisos en UberSVN

SVNAdmin 1.0 está implementada para llevar la gestión de repositorios a nivel de proyectos, con el objetivo de que los proyectos que se desarrollen en cada centro tengan acceso a sus repositorios, por lo que se utiliza para el acceso a estos repositorios el protocolo seguro HTTPS.

3.3.2 Seguridad

El sistema de administración de servidores Subversion necesita de autenticación de los usuarios para mayor seguridad, asignándoles los permisos necesarios sobre el sistema. La figura 18 muestra la interfaz de autenticación en el sistema para los usuarios.

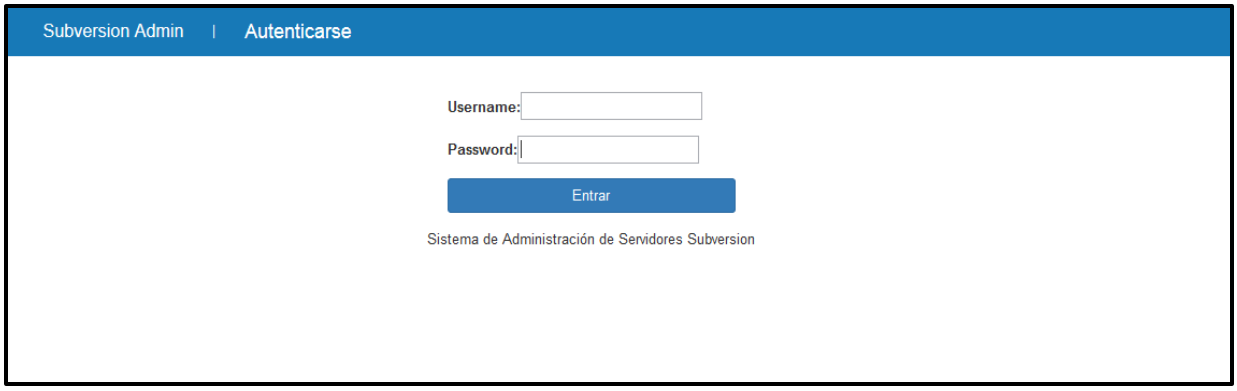

Fig. 18 Interfaz de Autenticación

Otro de los elementos que garantizan la seguridad en la aplicación, es que la misma utiliza la autenticación mediante el servicio LDAP, es decir, al agregar un nuevo usuario al sistema, este usuario se puede

autenticar mediante su usuario UCI y su contraseña. La figura 19 muestra un ejemplo de lo anteriormente descrito.

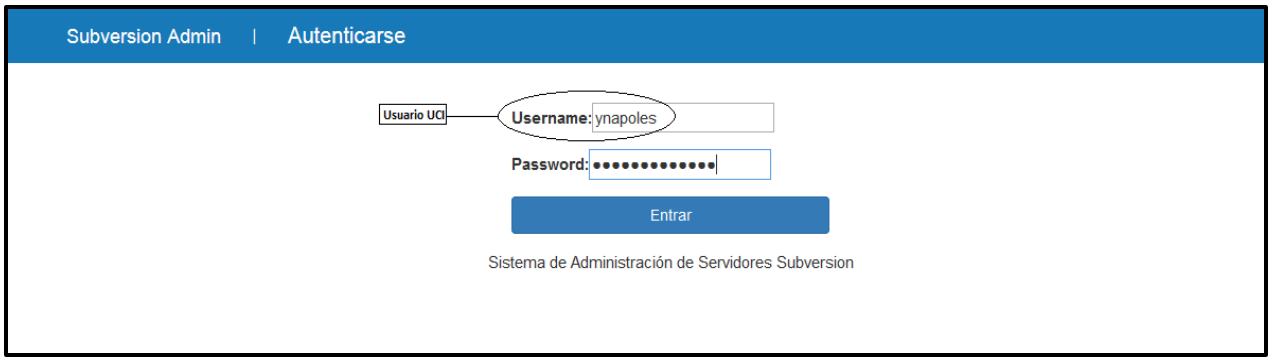

Fig. 19 Interfaz de Autenticación con usuario UCI

La aplicación al garantizar la gestión de acceso y permisos a nivel de proyecto, garantizará la seguridad en los repositorios debido a que solo tendrán acceso al repositorio aquellos usuarios que estén designados por proyecto, ejemplificado en la figura 20. Esto representa una ventaja con respecto a la herramienta UberSVN utilizada actualmente por la Universidad, ya que la misma realiza la gestión de los permisos de forma centralizada, lo que significa que contiene solamente un archivo de configuración donde se otorgan los permisos para todos los proyectos, por lo que resulta engorroso el cambio de privilegios en un repositorio específico, ya que hay que generar obligatoriamente un nuevo archivo de configuración con todos los permisos de todos los repositorios, esto se evidencia en la figura 21.

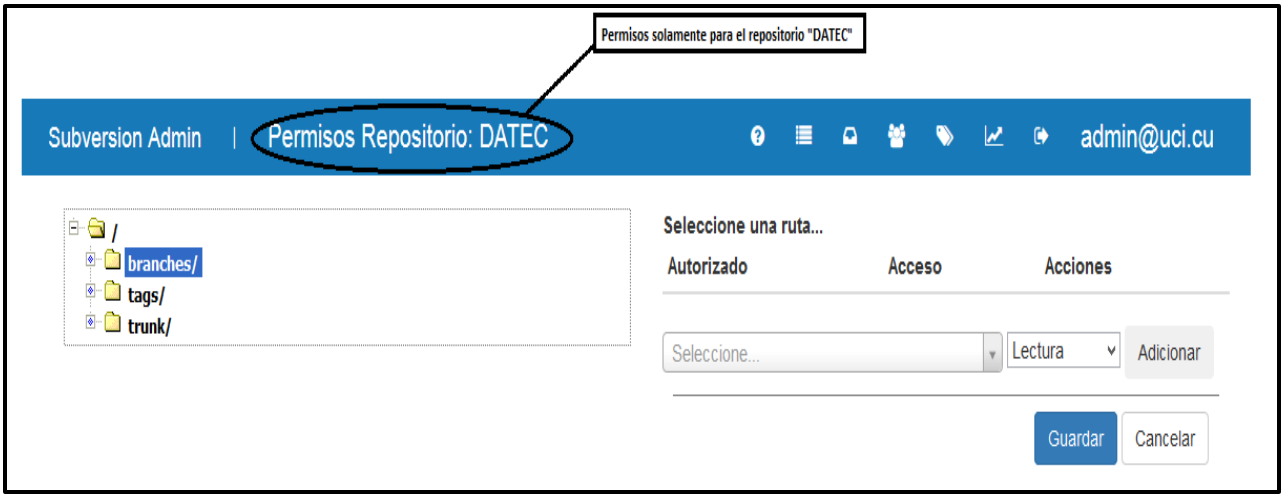

Fig. 20 Interfaz para asignar Permisos en SVNAdmin 1.0

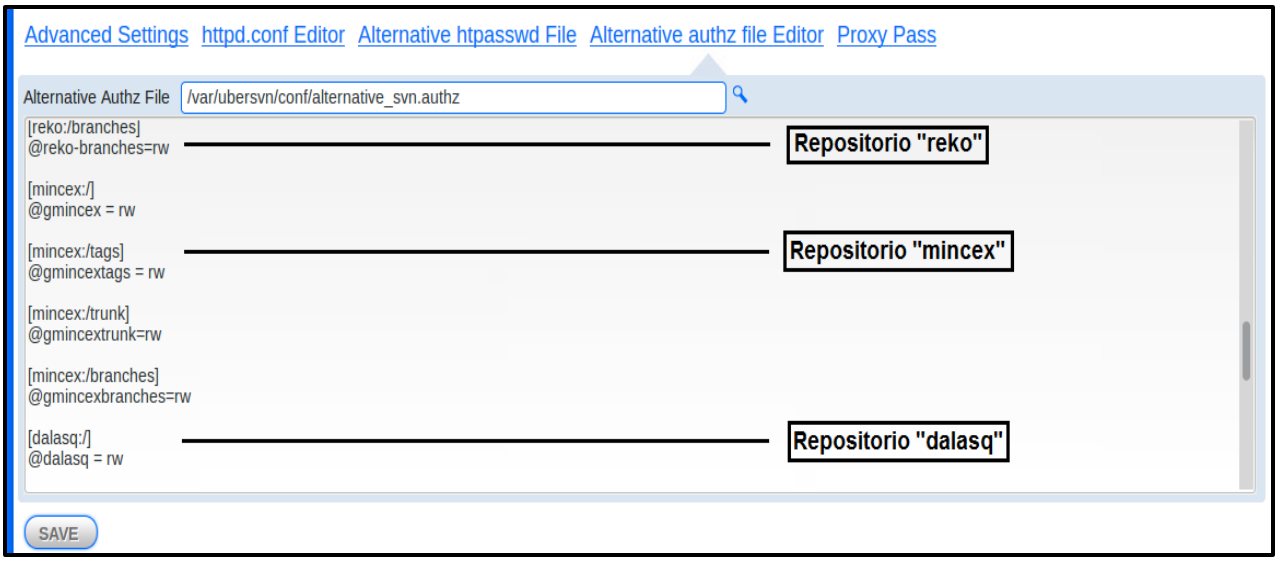

Fig. 21 Interfaz para asignar Permisos en UberSVN

3.4 Pruebas

Para encontrar los errores cometidos al diseñar y construir un software, se le realizan diferentes pruebas. Las pruebas son un conjunto de actividades que se planean con anticipado y se realizan de manera sistemática, por lo que se debe definir una plantilla para las pruebas de software que se realizarán, la cual se basa en un conjunto de pasos y procedimientos en que se puedan incluir técnicas y métodos específicos del diseño de casos de prueba (Pressman, 2005).

3.4.1 Pruebas Funcionales

Las pruebas funcionales están basadas en ejecución, revisión y retroalimentación de las funcionalidades implementadas para el software y las especificaciones definidas por el usuario. Con estas pruebas se busca evaluar el sistema mediante modelos de pruebas (Pressman, 2002). Al sistema implementado se le realizaron pruebas funcionales de Caja Negra las cuales se centran en los requisitos funcionales del software.

Método de prueba de Caja Negra

Las pruebas de Caja Negra se aplican a la interfaz del software, examinan algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software (Pressman, 2005). Estas

pruebas se concentran en los requisitos funcionales del software, tratando de encontrar los siguientes errores:

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores de estructuras de datos o en acceso a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y término.

Para cada uno de los requisitos funcionales fueron diseñados casos de prueba basados en casos de uso, los cuales se encuentran en el expediente de proyecto, además de los restantes artefactos entregados del sistema donde se encuentran todos los diseños de pruebas realizados. Estos casos de pruebas pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Para el diseño de los casos de pruebas de Caja Negra se utilizó la Técnica de Partición Equivalente, la cual divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. La partición equivalente se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse (Pressman, 2005). A continuación se presenta el diseño de caso de prueba que se utilizará para comprobar el funcionamiento del sistema.

Diseño de Caso de Prueba

La tabla 7 muestra la descripción de las variables que se encuentran asociadas al caso de uso “Gestionar Permisos”.

Tabla. 7 Descripción de las variables del CU Gestionar Permisos

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Explorador	Lista de selección	No	Estructura del repositorio donde el usuario puede tener permisos.

2	Autorizado	Lista desplegable	No	Usuario con los permisos de autorización sobre el repositorio.
3	Acceso	Lista desplegable	No	Permisos que presenta o puede presentar un usuario sobre un repositorio.

Se evaluó y probó la validez de cada una de las entradas al sistema, utilizando datos válidos (V), inválidos (I) y valores que no son necesarios de proporcionar (N/A). En la tabla 8 se muestra el escenario "Adicionar Permiso".

Tabla. 8 Escenario Adicionar Permiso

Escenario	Descripción	Variables (según la enumeración anterior)			Respuesta del sistema	Flujo Central
		1	2	3		
EC 1.1 Adicionar permiso	Se agregan permisos de usuarios o grupos al repositorio seleccionado.	**V** tags/	**V** ynapoles	**V** Escritura	El sistema guarda los datos seleccionados al seleccionar la opción "Guardar", y muestra una notificación "Se han guardado los cambios satisfactoriamente"	1-Se selecciona la opción de listar repositorios. 2- Se selecciona de un repositorio la opción Editar/Ver Permisos. 3-Se selecciona la trama del repositorio a agregar el permiso. 4- Se escoge el autorizado. 5- Se selecciona el acceso o permiso. 6- Se da clic en la opción "Adicionar". 7- Se da clic en la opción "Guardar".
EC 1.2 Adicionar permisos (datos nulos)	Se agregan permisos de usuarios o sin seleccionar la ruta del repositorio ni el autorizado.	**N/A**	**N/A**	**V** Lectura	El sistema muestra una notificación "Seleccione una ruta y el autorizado"	1-Se selecciona la opción de listar repositorios. 2- Se selecciona de un repositorio la opción Editar/Ver Permisos. 3- Se selecciona el acceso o permiso. 5- Se da clic en la opción "Adicionar".
		V	**V**	**V**		

EC 1.3 Adicionar permisos (usuario ya asignado)	Se agregan permisos de usuarios o grupos existentes al repositorio seleccionado.	tags /	ynapoles	Escritura	El sistema muestra una notificación "Ya existe una asignación para este usuario"	1-Se selecciona la opción de listar repositorios. 2- Se selecciona de un repositorio la opción Editar/Ver Permisos .3-Se selecciona la trama del repositorio a agregar el permiso. 4- Se escoge el autorizado. 5- Se selecciona el acceso o permiso. 6- Se da clic en la opción "Adicionar".
EC 1.4 Adicionar permisos(sin adicionar)	Se agregan permisos de usuarios o grupos al repositorio seleccionado.	**V** tags /	**V** ynapoles	**V** Escritura	El sistema muestra una notificación "Nada que guardar"	1-Se selecciona la opción de listar repositorios. 2- Se selecciona de un repositorio la opción Editar/Ver Permisos .3-Se selecciona la trama del repositorio a agregar el permiso. 4- Se escoge el autorizado. 5-Se selecciona el acceso o permiso. 6- Se da clic en la opción "Guardar".

Resultados de las pruebas de Caja Negra

Después de realizar las pruebas funcionales mediante el método de Caja Negra usando los casos de pruebas asociados para cada caso de uso, se comprobaron las funcionalidades del sistema y la correcta validación de sus campos. Fueron detectadas un total de 10 no conformidades durante 3 iteraciones, que consistían en errores funcionales del sistema y errores ortográficos, a las mismas se le dieron seguimiento y fueron resueltas a medida que se avanzó en el proceso de prueba. La figura 22 muestra un gráfico que representa las iteraciones realizadas.

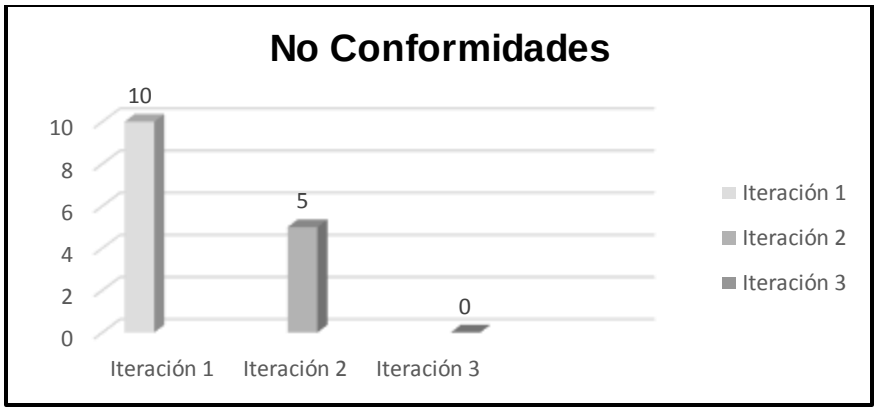

Fig. 22 Iteraciones de pruebas de Caja Negra

3.4.2 Pruebas de Rendimiento

Las pruebas de rendimiento se realizan con el objetivo de identificar cuál es la capacidad máxima del sistema ante una carga predefinida, de esta forma se garantiza la eficiencia del software (Sommerville, 2005). Dentro de las pruebas de Rendimiento existentes se encuentran las pruebas de carga y estrés, las cuales fueron aplicadas al sistema mediante la herramienta JMeter en su versión 2.13.

Pruebas de Carga

La prueba de carga se basa en un proceso que se le impone al software, el cual consiste en asignarle a este una cierta cantidad de peticiones predefinidas, con el objetivo de valorar el comportamiento de dicho software ante la situación planteada.

Pruebas de Estrés

Las pruebas de estrés de basan en determinar el comportamiento del sistema bajo un nivel de exigencia mayor al que es capaz de manejar. Estas pruebas reducen el riesgo de “caídas del sistema”, permitiendo aprovechar los recursos de forma más eficiente hasta conocer los límites que soporta el sistema. La aplicación de estas pruebas también permite tomar decisiones sobre configuraciones de hardware, ajustes de software y selección de arquitecturas posibilitando medir elementos como: el tiempo de respuesta, la cantidad de memoria consumida para resolver las peticiones y el número de transacciones realizadas en un determinado período de tiempo.

Resultados al aplicar las pruebas de Carga y Estrés

Al aplicar las pruebas de carga y estrés al sistema mediante la herramienta JMeter v2.13 se obtuvieron una serie de resultados, los cuales responden a los requisitos de hardware especificados en capítulos anteriores. Se realizaron varias pruebas con diferentes muestras para determinar el comportamiento del sistema ante un flujo de trabajo completo. Para aplicar esta prueba se utilizaron 2 computadoras, una realizando la función de PC cliente y la otra hizo la función de PC servidor (Repositorio Central), las prestaciones de las mismas fueron las siguientes:

- PC cliente: 2.4 GHz velocidad de microprocesador, Memoria RAM 2GB.
- PC servidor: Pentium Dual-Core, 2.4 GHz velocidad de microprocesador, Memoria RAM 2GB.

La figura 23 muestra el Plan de Pruebas realizada para el caso de uso Gestionar Permisos inicialmente con 50 usuarios (ver Anexo 4 para otras muestras), demostrando las peticiones realizadas para este flujo de trabajo.

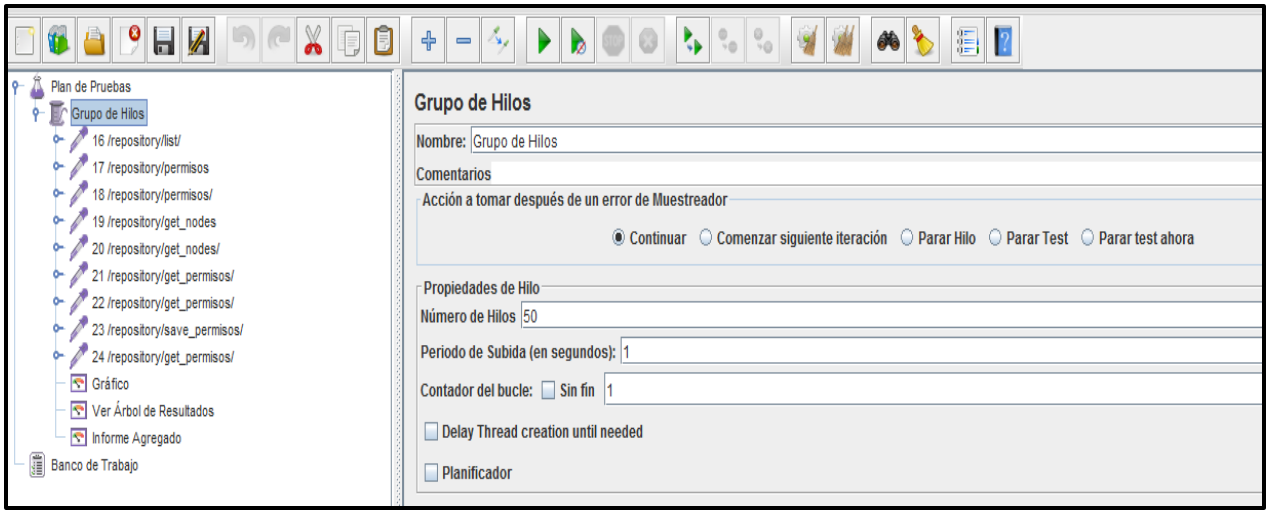

Fig. 23 Plan de Pruebas en JMeter

A partir de este Plan de Pruebas se realizaron diferentes peticiones sobre la aplicación con el objetivo de comprobar la respuesta del sistema ante estas peticiones. La figura 24 muestra el gráfico generado por la herramienta JMeter sobre el caso de uso Gestionar Permisos, mediante la cual se organizan las peticiones realizadas.

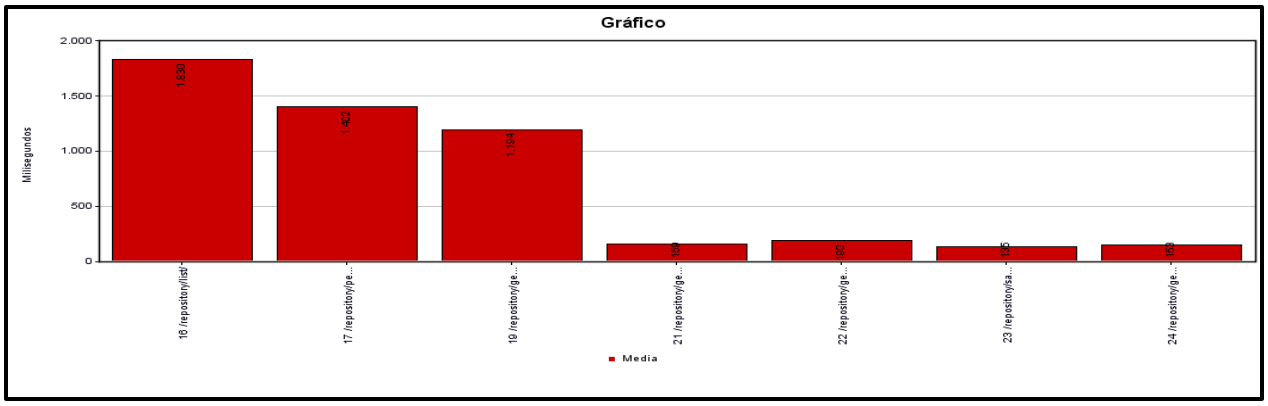

Fig. 24 Gráfico de resultados para el caso de uso Gestionar Permisos

La figura 25, muestra además el Informe Agregado generado por el caso de uso Gestionar Permisos, mostrando el rendimiento de las peticiones.

Informe Agregado

Nombre: Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimie...	Kb/sec
16 /repository/list/	50	2270	437	9037	9304	9439	197	9439	0,00%	4,8/sec	11,7
17 /repository/permisos	50	2025	596	9183	9434	9544	65	9544	0,00%	4,7/sec	12,7
18 /repository/permisos/	50	1007	424	3418	3440	9353	71	9353	0,00%	4,8/sec	11,7
19 /repository/get_nodes	50	802	571	1230	1550	3417	107	3417	0,00%	5,1/sec	14,5
20 /repository/get_nodes/	50	387	338	487	530	1111	175	1111	0,00%	5,4/sec	13,6
21 /repository/get_permisos/	50	236	136	238	509	3150	34	3150	100,00%	5,6/sec	15,5
22 /repository/get_permisos/	50	147	141	193	200	924	22	924	100,00%	5,8/sec	15,8
23 /repository/save_permisos/	50	372	144	217	3122	3203	17	3203	100,00%	5,9/sec	16,0
24 /repository/get_permisos/	50	189	120	195	238	3217	15	3217	100,00%	6,0/sec	16,4
Total	450	826	304	2487	3500	9434	15	9544	44,44%	37,9/sec	100,5

Fig. 25 Informe Agregado para el caso de uso Gestionar Permisos

La primera muestra realizada con 50 usuarios conectados al sistema demostró la prueba de Carga, donde el sistema respondió adecuadamente ante la cantidad de usuarios conectados simultáneamente definidos en los RNF. Para la realización de la prueba de Estrés se tomaron diferentes muestras de diferentes cantidades de usuarios para comprobar la respuesta del sistema ante momentos de carga extrema. La figura 26 muestra de forma general los resultados que se obtuvieron con las muestras correspondientes, a través de la gráfica.

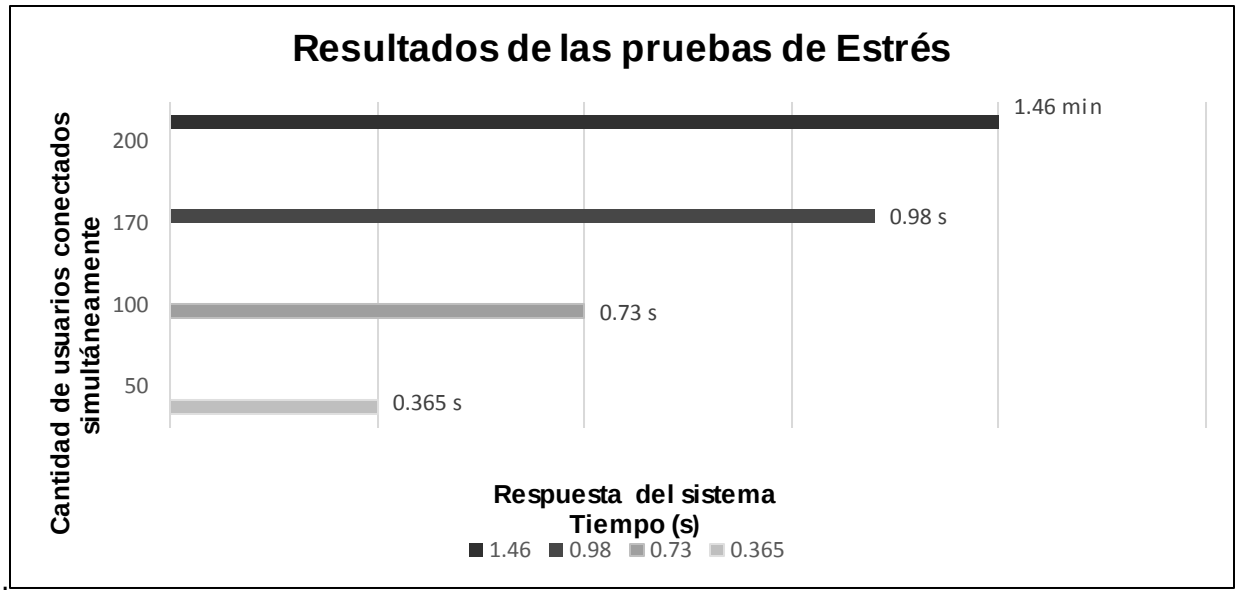

Fig. 26 Resultados de la prueba de Estrés realizada a la aplicación

El análisis de esta tabla permitió identificar el funcionamiento del sistema ante una carga extrema, con una respuesta de 0.365 segundos para 50 usuarios mientras que para 200 usuarios la respuesta fue de 1.46 minutos. Estas pruebas de rendimiento resultaron de gran utilidad, pues mediante ellas se comprobó la capacidad de respuesta del sistema ante situaciones predefinidas tanto de manera general, como de forma específica ante un flujo de trabajo determinado.

3.5 Conclusiones del capítulo

En este capítulo se realizó la implementación y validación de la aplicación web SVNAdmin 1.0, cumpliéndose con el estándar de codificación descrito para hacer más fácil el entendimiento del código por los programadores y el futuro mantenimiento a la aplicación. Se presentó también el Modelo de Implementación, donde fue descrita la distribución de los componentes del sistema utilizando la distribución física presentada en el diagrama de despliegue. Finalmente se realizaron pruebas al sistema, las cuales arrojaron resultados satisfactorios, demostrando la calidad del mismo.

CONCLUSIONES GENERALES

En el desarrollo del presente trabajo de diploma se obtuvo como resultado la creación de la aplicación web SVNAdmin 1.0, que garantiza la usabilidad y seguridad para la gestión de repositorios SVN en la Red de Centros de desarrollo de la Universidad, dando cumplimiento al objetivo general. Para esto:

- Se realizó un estudio sobre diferentes herramientas administrativas Subversion, lo cual aportó una serie de elementos que permitieron obtener una visión más completa para el desarrollo de la nueva aplicación web para la gestión de repositorios Subversion, SVNAdmin 1.0.

- Fueron generados los artefactos relacionados con el análisis y diseño, siguiendo la metodología de desarrollo seleccionada, AUP-UCI, los cuales constituyeron la base para la implementación exitosa del sistema.

- La aplicación desarrollada fue expuesta a pruebas de caja negra y pruebas de rendimiento, las cuales validaron y arrojaron resultados satisfactorios que permitieron demostrar la calidad de dicha aplicación.

- Mediante las comparaciones realizadas entre la herramienta UberSVN y la aplicación web SVNAdmin 1.0, teniendo en cuenta los archivos de configuración y las interfaces que presentan ambas, quedó demostrado que la aplicación desarrollada garantiza la usabilidad y seguridad en la gestión de repositorios Subversion, lo cual provocó gran impacto sobre los administradores centrales de los repositorios Subversion.

RECOMENDACIONES

Según los resultados obtenidos y las experiencias adquiridas en el transcurso del desarrollo de la aplicación, se recomienda:

- Incorporar al sistema el concepto de los roles predefinidos a nivel de proyecto.
- Incorporar funciones de coloración de listados de archivos, para identificar los diferentes archivos mediante niveles de colores, logrando una interfaz más amigable para el usuario.
- Incorporar funciones que permitan realizar intercambios con los restantes sistemas relacionados con la producción de la Universidad.

REFERENCIAS BIBLIOGRÁFICAS

- ACCESSMANAGER. 2008. SVN Access Manager Documentation. [En línea] 2008. www.accessmanager.co.uk.
- ALCHIN, M. 2013. ACCESSMANAGER, S. SVN Access Manager Documentation Understanding Django, in: Pro Django. Springer, pp. 1-10 In., 2008. 2013.
- ALMAGRO, C. U. 2011. Lenguajes de Programación: Capítulo 1. In Proceedings of 2011. 2011.
- BAHIT, E. 2012. Available from World Wide Web:<http://www.eugeniabahit.com/cursos2012>. 2012.
- BEN COLLINS-SUSSMAN; Brian W. Fitzpatrick; C. Michael Pilato, 2002. Control de versiones con Subversion: Revision 5001. Available from World Wide. 2002.
- Ceria, Santiago. 2001. casos de uso. Un método Práctico para Explorar Requerimientos. Edtion ed. 2001.
- Diccionario. 2015. Real Academia Española. [En línea] 2015. www.rae.es.
- DODERO, RUIZ-RUBE, I. TRAVERSO RIBÓN, M. PALOMO DUARTE. 2014. Recomendaciones formativas para la evaluación sostenible de proyectos colaborativos en. 2014.
- EDEKI, Charles. 2013. Agile Unified Process. 2013.
- Gamma, Erich. 2008. Patrones de Diseño: elementos de software orientados a objetos reutilizables. . In Proceedings of 2008. 2008.
- Jump Start Bootstrap. Edtion ed. RAHMAN, S. F. 2014. 2014.
- KAPLAN-MOSS, Jacob 2013. El libro de Django. 2008. 2013.
- LUTZ, Mark. 2013. Learning python. O´Reilly Media. 2013.
- Martínez, P. I. C. 2014. Clase 6: Modelo Conceptual/ Modelo de Dominio. In Proceedings of 2014. 2014.
- Niska, C. 2014. Extending Bootstrap. Understand Bootstrap and unlock its secrets to build a truly customized project. Edtion ed. 2014.
- ORTEGA, R. J. L., JESÚS M., y otros. 2007. Ingeniería del Software orientada al desarrollo web. Desarrollo rápido de aplicaciones Web 2.0 con Python y Django. Edtion ed. 2007.

- PARADIGM, VISUAL. 2013. Visual Paradigm Know-how. Business modeling, software development and enterprise architecture with great products, and knowledge. [En línea] www.visual-paradigm.com, 2013.
- Pressman, Roger. 2002. Ingeniería del Software: Un Enfoque Práctico. Edtion ed 6ta. 2002.
- Ingeniería del Software. Un enfoque práctico, SI: MCGRAW-HILL.Edtion ed. 2005.
- ROSSUM, Guido Van. 2000. Guia de aprendizaje de Python. Release 2.0 [online] http://es.tldp.org/Tutoriales/Python/tut.pdf. 2000.
- SALAMANCA, Universidad. 2015. Biblioteca. Información bibliográfica. [En línea] 2015. http://bibliotecas.usal.es.
- SÁNCHEZ, TAMARA. 2014. Metodología de desarrollo para la Actividad productiva de la UCI. 2014.
- Sommerville, Ian. 2005. Ingeniería de Software 7, Madrid: Pearson Educación S.A. 2005.
- Software Engineering. edited by V. EDICIÓN. Edtion ed., ISBN 13:978-0-321-31379-9. 2007.
- THOMAS, DIRK. 2011. WebSVN - Online subversion repository browser. In. [En línea] 2011. www.websvn.info.
- Thornton, Jacob; Eguiluz, Traductor Javier; Otto, Mark 2015. Bootstrap 3, Manual Oficial [online]Available from World Wide Web:<https://librosweb.es/libro/bootstrap_3/>. 2015.
- Turnquist, Greg L. 2011. Python Testing Cookbook. 2011.
- UBERSVN. 2012 UberSVN Powered by WANdisco. [En línea] 2012. http://docs.ubersvn.com.
- UCI. 2015. Catálogo de Productos y Servicios (INFORMÁTICAS(UCI). 2015.
- USVN. 2013. User-friendly SVN. [En línea] 2013. www.usvn.info.
- VISUALSVN. 2015. VisualSVN Server. [En línea] 2015. www.visualsvn.com.

BIBLIOGRAFÍA

- Especificación de requerimientos. Diseño de bases de datos. 2015, [cited 15 de febrero 2015]. Available from Internet:<http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf>. DECSAI. 2015. 2015.
- ACCESSMANAGER. 2008. SVN Access Manager Documentation. [En línea] 2008. www.accessmanager.co.uk.
- AENOR.UNE-EN-ISO. 2000. Gestión de la calidad y aseguramiento de calidad. 2000.
- Agile Web Development with Rails, 3rd ed (capítulos 2 y 3). In Agile Web Development with Rails, 3rd ed (capítulos 2 y 3). C, G. 2009. 2009.
- ALCHIN, M. 2013 ACCESSMANAGER, S. SVN Access Manager Documentation Understanding Django, in: Pro Django. Springer, pp. 1-10 In., 2008. 2013.
- ALMAGRO, C. U. 2011. Lenguajes de Programación: Capítulo 1. In Proceedings of 2011. 2011.
- BAHIT, E. 2013 Curso: Python para Principiantes [online]. 2012. Available from World Wide. [En línea], www.**cursos**deprogramacionadistancia.com. 2013.
- BAHIT, E. 2012. Available from World Wide Web:<http://www.eugeniabahit.com/cursos2012>. 2012.
- BEN COLLINS-SUSSMAN; Brian W. Fitzpatrick; C. Michael Pilato 2002. Control de versiones con Subversion: Revision 5001. Available from World Wide. 2002.
- Centro de Ensayos de Software, CES. CES. 1989. 1989.
- Ceria, Santiago 2001. casos de uso. Un método Práctico para Explorar Requerimientos. Edtion ed. 2001.
- Desarrollo colaborativo bajo Software Libre. In Proceedings of 2007. ROJAS, I. 2007. 2007.
- Diccionario. 2015. Real Academia Española. [En línea] 2015. www.rae.es.
- DODERO, RUIZ-RUBE, I. TRAVERSO RIBÓN, M. PALOMO DUARTE. 2014. Recomendaciones formativas para la evaluación sostenible de proyectos colaborativos en. 2014.
- Django. http://docs.djangoproject.com/en/1.8/topics/db
- EDEKI, Charles 2013 Agile Unified Process. 2013.

- Extreme Programming Installed. Addison-Wesley Professional. ISBN 0-201-70842-6. JEFFRIES, R. 2000. 2000.
- Gamma, Erich. 2008. Patrones de Diseño: elementos de software orientados a objetos reutilizables. . In Proceedings of 2008. 2008.
- Jump Start Bootstrap. Edtion ed. RAHMAN, S. F. 2014. 2014.
- KAPLAN-MOSS, Jacob 2013. El libro de Django [online], http://django-book.mkaufmann.com.ar/. 2008. 2013.
- LUTZ, Mark. 2013. Learning python. O´Reilly Media. 2013.
- Martínez, P. I. C. 2014. Clase 6: Modelo Conceptual/ Modelo de Dominio. In Proceedings of 2014. 2014.
- Niska, C. 2014. Extending Bootstrap. Understand Bootstrap and unlock its secrets to build a truly customized project. Edtion ed. 2014.
- ORTEGA, R. J. L., JESÚS M., y otros. 2007. Ingeniería del Software orientada al desarrollo web. Desarrollo rápido de aplicaciones Web 2.0 con Python y Django. Edtion ed. 2007.
- PARADIGM, VISUAL. 2013. Visual Paradigm Know-how. Business modeling, software development and enterprise architecture with great products, and knowledge. [En línea], www.visual-paradigm.com. 2013.
- Patrones del "Gang of Four". Unidad Docente de Ingeniería del Software. Facultad de Informática-Universidad Politécnica de Madrid . MEDINILLA, N. 2008. 2008.
- Pressman, Roger. 2002. Ingeniería del Software: Un Enfoque Práctico. Edtion ed 6ta. 2002.
- Ingeniería del Software. Un enfoque práctico, SI: MCGRAW-HILL.Edtion ed. 2005.
- Revista cubana de Ciencias Informáticas . DASIEL CORDERO MORALES, Y. R. C., YOANNY TORRES RUBIO. 2013
- ROSSUM, G. V. 2000. Guia de aprendizaje de Python. Release 2.0 [online], http://es.tldp.org/Tutoriales/Python/tut.pdf. 2000.
- SALAMANCA, U. D. 2015. Biblioteca. Información bibliográfica. [En línea] 2015. http://bibliotecas.usal.es.
- SÁNCHEZ, TAMARA. 2014. Metodología de desarrollo para la Actividad productiva de la UCI. 2014.
- Sommerville, Ian. 2005. Ingeniería de Software 7, Madrid: Pearson Educación S.A. 2005.

- Software Engineering. edited by V. EDICIÓN. Edtion ed., ISBN 13:978-0-321-31379-9. 2007.
- THOMAS, DIRK. 2011. WebSVN - Online subversion repository browser. In. [En línea] 2011. www.websvn.info.
- Thornton, Jacob; Eguiluz, Traductor Javier; Otto, Mark 2015. Bootstrap 3, Manual Oficial [online]Available from World Wide Web:<https://librosweb.es/libro/bootstrap_3/>. 2015.
- Turnquist, Greg L. 2011. Python Testing Cookbook. 2011.
- UBERSVN. 2012 UberSVN Powered by WANdisco. [En línea] 2012. http://docs.ubersvn.com.
- UCI. 2015. Catálogo de Productos y Servicios (INFORMÁTICAS(UCI). 2015.
- USVN. 2013. User-friendly SVN. [En línea] 2013. www.usvn.info.
- VISUALSVN. 2015. VisualSVN Server. [En línea] 2015. www.visualsvn.com.

ANEXOS

Anexo #1: Centros Productivos UCI

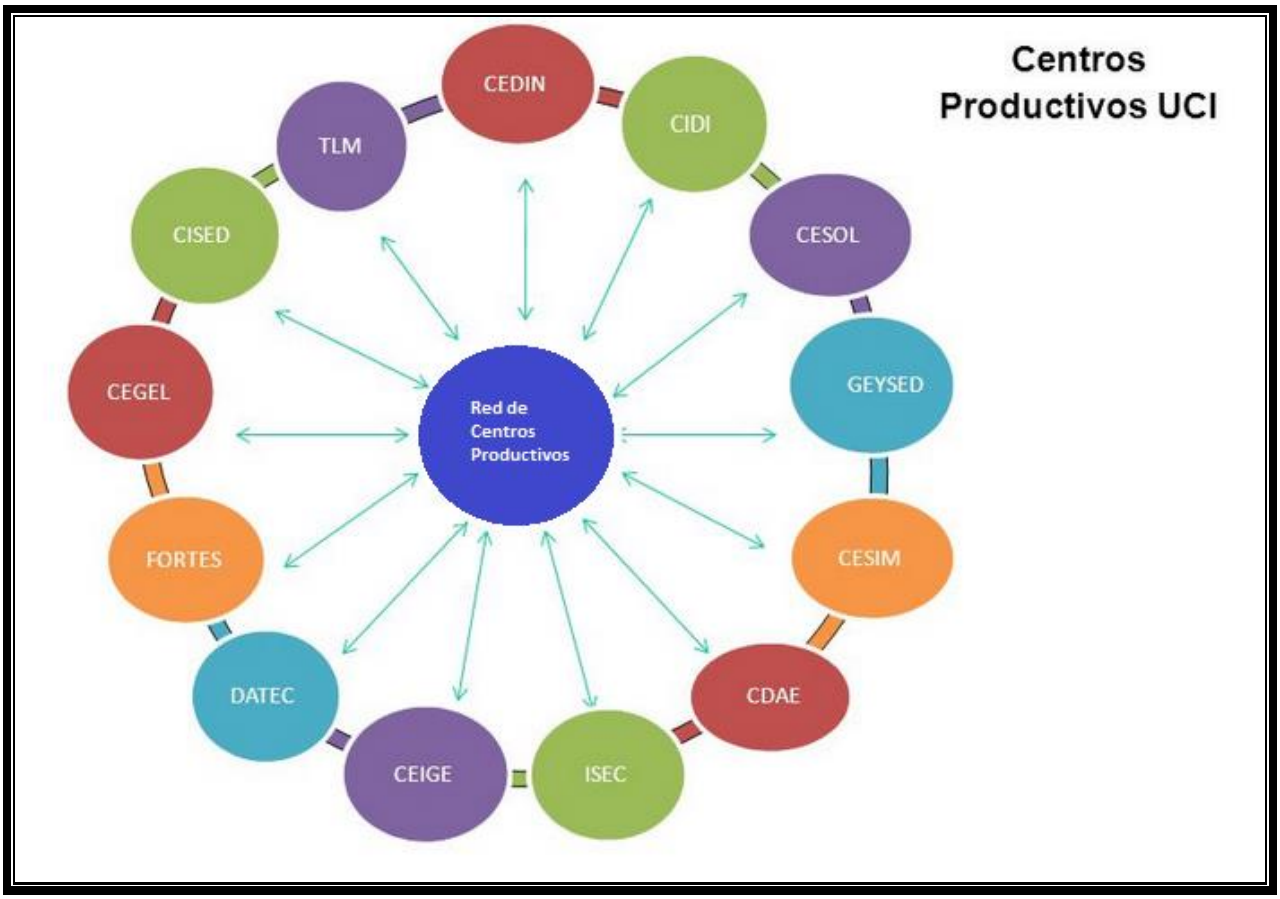

Fig. 27 Centros Productivos UCI

Anexo # 2: Encuesta

Cuestionario sobre Herramientas de Administración Subversion.

A: Administradores profesionales de Subversion.

Nota: *Este cuestionario permitirá recopilar información sobre las funciones que cumplen las herramientas de administración Subversion. Gracias por la atención.*

1. **Marque con una X las respuesta según su criterio:**

"Tengo mayor experiencia en la herramienta de administración:"

_SVNManager

_UberSVN

_USVN

_VisualSVN

_WebSVN

2. **Complete el espacio en blanco:**

Las herramientas de administración de Subversion tienen gran importancia porque_____
_____
_____.

3. **Complete el espacio en blanco:**

La estructura que presentan los repositorios Subversion es_____

_____

4. **Marque con X las respuestas correctas:**

"Entre las estrategias para controlar el acceso a los repositorios se encuentra":

_Control de acceso simple

_Control de acceso a directorios

_Ninguna de las anteriores

5. **Marque con una X las respuestas correctas:**

"Algunos de los comandos que se utilizan sobre los repositorios SVN son":

_svn.commit

_svn.update

_svn.manager

Anexo # 3: Entrevista

Introducción: Se realizó una entrevista a los profesionales que actualmente interactúan con la herramienta de administración Subversion UberSVN, para determinar cuáles son las necesidades que los mismos presentan a la hora de llevar a cabo su trabajo y la integración con las herramientas de gestión de la producción como GESPRO. De esta forma se pudieron recopilar una serie de requerimientos.

Se realizó una entrevista abierta que además de las preguntar que se exponen a continuación las mismas dieron lugar a otras preguntas. Fue llevada a cabo el día 11/03/2015 a las 2:00 pm, en un local perteneciente a la Dirección General de la Producción.

Objetivo: Realizada con el objetivo de recopilar información necesaria sobre el trabajo que actualmente se lleva sobre UberSVN y de esta manera recopilar los requisitos necesarios para el desarrollo de la solución propuesta.

Dirigida a: Administradores de la herramienta de administración Subversion.

1. Cuando se crea un proyecto en el GESPRO y se definen los permisos de los usuarios sobre el mismo, ¿qué función cumple la herramienta Subversion con el repositorio perteneciente a este proyecto?
2. ¿Cómo se gestionan los permisos en la herramienta actual?
3. ¿Cuáles son las dificultades que presenta la herramienta al presentar la información de las últimas gestiones realizadas a la herramienta?

Anexo # 4: Pruebas de Rendimiento en JMeter para el CU Gestionar Permisos

- Gráfico de resultados en JMeter para 100 usuarios

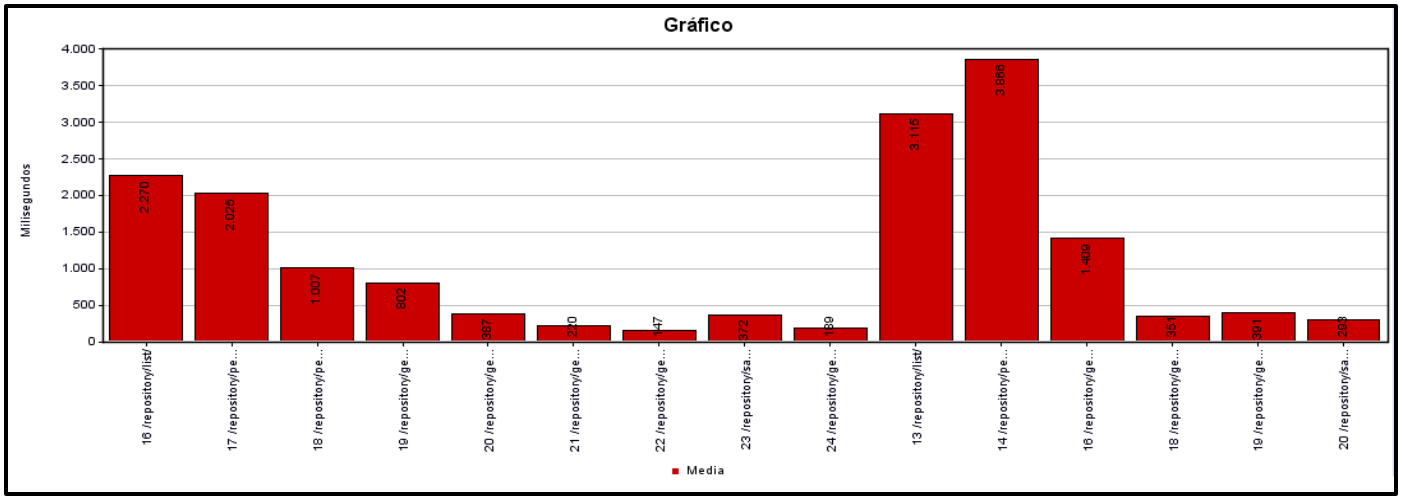

Fig. 28 Gráfico de resultados para 100 usuarios

- Gráfico de resultados en JMeter para 170 usuarios

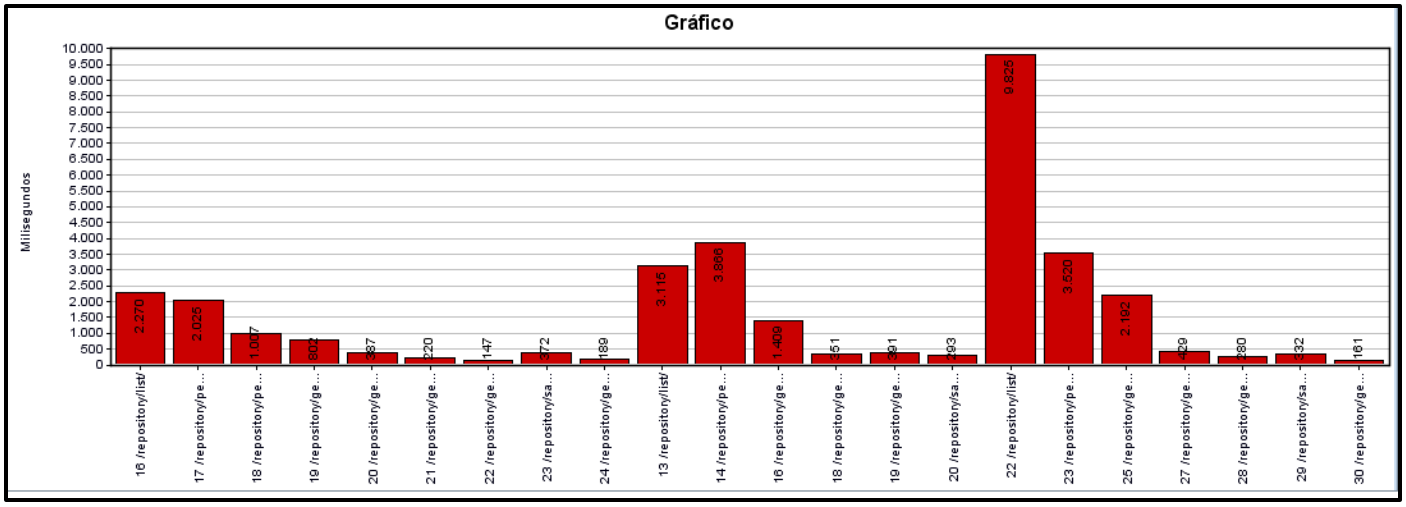

Fig. 29 Gráfico de resultados para 170 usuarios

- Gráfico de resultados en JMeter para 200 usuarios

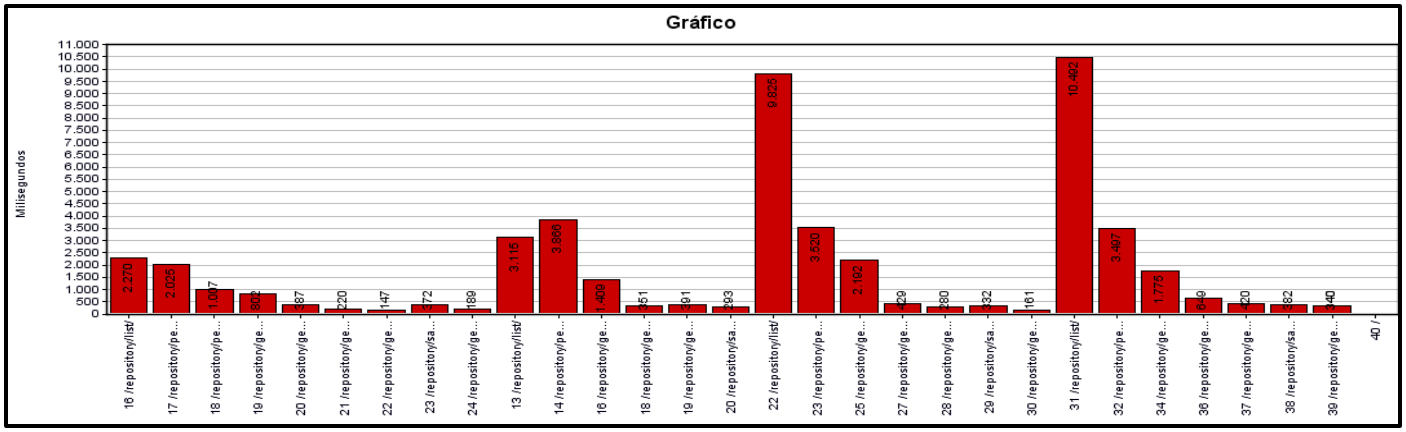

Fig. 30 Gráfico de resultados para 200 usuarios

GLOSARIO DE TÉRMINOS

- **Casos de prueba:** conjunto de entradas de pruebas, condiciones de ejecución y salidas del Sistema.
- **CSS:** Cascading Style Sheets u Hojas de Estilo en Cascada, lenguaje utilizado para separar la estructura de la presentación.
- **PHP:** Hypertext Preprocessor (PHP: Preprocesador de Hipertexto)
- **WANdisco:** proveedor líder de software basado en Apache Subversion.