Universidad de las Ciencias Informáticas
Facultad 7

Título: ClioBD. Sistema de control de versiones para bases de datos

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Gerardo Morgade Donato

Tutores: MSc. Maypher Román Durán

Lic. Maykell Sánchez Romero

Ciudad de La Habana, Junio de 2009

"Año del 50 aniversario del triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 18 días del mes de junio del año 2009.

Gerardo Morgade Donato

_____
Firma del Autor

MSc. Maypher Román Durán

_____
Firma del Tutor

Lic. Maykell Sánchez Romero

_____
Firma del Tutor

“Aerodinámicamente, el cuerpo de una abeja no está hecho para volar; lo bueno es que la abeja no lo sabe”

Poster de la NASA

DATOS DE CONTACTO

Maypher Román Durán: Graduado de Ingeniería Informática del Instituto Superior Politécnico Jose Antonio Echeverría (ISPJAE) de Ciudad de la Habana, profesor Asistente y Máster en Ciencias en Informática Aplicada. Ha sido Asesor Técnico Docente del Departamento Docente Central de Ingeniería y Gestión de Software de la UCI de las asignaturas Sistemas de Bases de Datos y Gestión de Software, Diseñador principal de Bases de Datos de proyectos productivos de la Facultad 6.
Correo electrónico: maypher@uci.cu.

Maykell Sánchez Romero: Graduado de Licenciatura en Ciencias de la Computación de la Universidad de La Habana, profesor instructor. Ha sido Jefe de Departamento de Ciencias de la Especialidad de la facultad 7, arquitecto de la facultad 7, Jefe de Grupo de Investigación y Jefe de Proyecto.
Correo electrónico: maykell@uci.cu.

AGRADECIMIENTOS

A mis padres por ser como son, por formarme como hombre y por darme un buen ejemplo a seguir.

A mis tías y tíos por siempre apoyarme. Por cada cual considerar que lo que yo soy se lo debo a ellos. Tienen toda la razón.

A Marquetty por sus tantos consejos, que escucho más de lo que él cree.

A mis tutores por no dudar ni un momento al seguirme en ese proyecto incierto que fue este trabajo en sus inicios, por el apoyo incondicional y por ser antes que tutores, amigos.

A mi oponente por no utilizar su característico "No chama" en el transcurso de esta tesis, y por ser más tutor que oponente.

A Keyla por ser otra tutora aunque no le tocara serlo.

A Alexander por ser más que un amigo, un hermano y siempre estar presente.

A los "Clásicos" por siempre estar ahí, por soportarme las pesadeces y por retarme, que es lo que me ha hecho superarme día a día y es lo que me convierte en un profesional mejor.

DEDICATORIA

Dedico este trabajo de diploma a mis hermanos.

Por lograr que yo me esfuerce cada día por darles una meta a alcanzar.

Por darme fuerzas para seguir adelante.

Por ver mis logros como propios y tomarlos como retos.

Espero este reto esté a su altura.

RESUMEN

Cuba apuesta por el desarrollo de software como una vía de ingresos a la economía nacional, para esto se crean equipos de trabajo cada vez más grandes. La sincronización del trabajo de cada uno de los miembros de estos equipos no es una tarea fácil, lo que crea una oportunidad para los sistemas de apoyo a la gestión de la configuración. El control de versiones es una rama fundamental dentro de la gestión de la configuración teniendo entre otros a la base de datos como uno de los posibles elementos de configuración a versionar.

ClioBD es una solución de software orientada al control de versiones para bases de datos. El mismo fue desarrollado empleando la metodología RUP y las ventajas de la plataforma .NET, con el empleo en todo el proceso de desarrollo herramientas y librerías pertenecientes al movimiento del software libre.

Con este trabajo se obtiene una herramienta que permite a los diseñadores de bases de datos mantener las versiones por las que transcurre el diseño de datos en el proceso de desarrollo de software. Así como lograr una mayor capacidad de integración en los equipos de modelado de datos.

PALABRAS CLAVE

Gestión de la configuración, sistema de control de versiones, base de datos, PostgreSQL.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El mundo se encuentra en la llamada “sociedad de la información”, actualmente se utilizan un sinnúmero de recursos con el fin de adquirir, almacenar y procesar datos. Este flujo de información es cada vez mayor y viene aparejado con la necesidad de programas cada vez más complejos y elaborados. Para la realización de los mismos, se necesitan nuevas tecnologías que permitan a los desarrolladores de software dar solución a diversos problemas en cronogramas más exigentes.

El uso de las bases de datos se ha generalizado en el mundo informático y sus diseños son más elaborados y extensos con el paso del tiempo. Los gestores de bases de datos evolucionan para responder a los crecientes requerimientos de los clientes. Lo que trae como resultado que los diseñadores de bases de datos desarrollen un conjunto de habilidades lo que ha contribuido a la diversificación de especialidades dentro de esta área de la informática.

Se puede definir una **base de datos** como un conjunto no redundante de datos pertenecientes a un mismo contexto, almacenados para su uso posterior y accesible en tiempo real, con usuarios concurrentes con necesidades de información diferentes y no predecibles en el tiempo. Entre los principales beneficios que reportan se podría incluir: la independencia de los datos, ya que el cambio en estos no implica cambio en los programas y viceversa; la coherencia de resultados, ya que evita la redundancia. También se mejora la disponibilidad de los datos mientras se apoya en mecanismos de control de accesos y optimiza la gestión del almacenamiento.

A pesar de esto en la actualidad no es posible enmarcar una base de datos como un conjunto de tablas y funciones, pues en dependencia del gestor de bases de datos estas tienen diferentes vías para ser manipuladas. Hoy existen bases de datos orientadas a objetos, distribuidas, Data WareHouse (almacén de datos) y otra infinidad de tipos, lo que hace casi imposible la definición de un esquema homogéneo con el cual se pueda definir un modelo genérico de las mismas.

Las empresas de desarrollo de software modernas, cuentan con grandes equipos de desarrollo que trabajan concurrentemente en un mismo producto. Lo que hace necesario el uso de **sistemas de control de versiones**, para poder llevar el seguimiento de las actualizaciones hechas por cada miembro del grupo de trabajo, sobre los muchos componentes que conforman una aplicación informática. En estos casos, dichos sistemas de control de versiones se convierten en una potente herramienta para el equipo de trabajo.

Un sistema de control de versiones, puede definirse como un programa que permite conocer los cambios de un archivo o grupo de archivos determinados en el proceso de elaboración de estos. Las funcionalidades básicas que debe proveer cualquier sistema de este tipo son: uno o más mecanismos de almacenamiento de información, la posibilidad de realizar cambios sobre los elementos almacenados. Así como, el control de un registro histórico de los cambios realizados por cada miembro del equipo del proyecto sobre cada elemento o conjunto de estos.

Estos sistemas de control de versiones se pueden dividir en dos grandes grupos: los exclusivos donde para modificar un elemento se debe marcar este y el sistema se encargará que ningún otro usuario pueda modificarlo. El otro grupo son los colaborativos en los que cada usuario trabaja con una copia local de los elementos y el sistema se encarga de mezclar las diferentes modificaciones hechas sobre estos. El principal problema del último grupo mencionado es la posible aparición de conflictos que deban ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas.

Los controladores de versiones colaborativos son muy utilizados en el desarrollo de software por la capacidad de los mismos para brindar un alto nivel de paralelismo dentro del equipo de desarrollo. Pero esta semántica no es factible utilizarla en el trabajo con ficheros binarios, ya que es casi imposible encontrar una forma coherente de mezclar los cambios realizados sobre el mismo archivo por dos o más usuarios.

Incluso con todos los avances que se han obtenido en los campos de las bases de datos y los sistemas de control de versiones, los programas existentes para lograr un punto de convergencia entre estas tecnologías están desarrollados para un grupo restringido de gestores de bases de datos, excluyendo los sistemas libres en la mayoría de los casos. Esto trae consigo que los equipos de especialistas que realizan el diseño de base de datos dentro de un proyecto no cuenten con una aplicación que les permita controlar los cambios realizados sobre la estructura definida en su base de datos a medida que transcurre el proceso de desarrollo. Todo ello ocasiona en muchos casos la redundancia de datos y atrasos en los cronogramas de entrega.

Se puede destacar que un controlador de versiones para bases de datos relacionales proporcionaría un grupo de beneficios al proceso de diseño de bases de datos. Entre ellos los más relevantes podrían ser: un aumento de la coordinación en los grupos de diseñadores de base de datos así como un aumento de la capacidad de revisión de los líderes de estos grupos. Lo que le proporciona en su conjunto, una mayor velocidad para la liberación de los modelos de datos que se necesitan en la industria.

En mayo del 2002 se crea en Cuba la Universidad de las ciencias informáticas (UCI), con el fin de potenciar el estudio de esta ciencia y convertirla en una de las principales ramas del país. Esta debe integrarse con las demás instituciones de la rama que existen en el país y convertirse en el centro de la informática en Cuba, donde se desarrollen los principales sistemas de software tanto para la informatización de la sociedad como para la exportación.

En la mayoría de los proyectos de desarrollo de software que se realizan en la UCI, se lleva un control de las versiones de los componentes generados en los mismos. Pero en la mayoría de los casos, no se gestionan las versiones por las que pasa el diseño de la base de datos por carecer de una herramienta que automatice este proceso. Gran parte de estos proyectos presentan diseños complejos de bases de datos, por lo que es necesario que intervengan en su elaboración varios diseñadores de base de datos, lo que ocasiona los inconvenientes anteriormente mencionados.

Actualmente en la UCI, en un proyecto determinado, no se pueden conocer los cambios realizados por cada desarrollador sobre la base de datos al transcurrir el ciclo de desarrollo; ni llevar un control histórico de dichos cambios. Tampoco es posible regresar a una versión dada de la base de datos en un momento determinado del proyecto. Por todo esto, se hace necesaria la creación de un sistema de control de versiones colaborativo para bases de datos relacionales. Este debe ser lo suficientemente flexible como para adaptarse a la diversidad de gestores de base de datos que allí se utilizan.

En este sentido, se define como **Problema a resolver**: ¿Cómo gestionar las versiones por las que transita una base de datos durante el proceso de desarrollo de un producto de software en la UCI?

El **Objeto de estudio** lo constituye el proceso de control de versiones durante el desarrollo de un producto de software. Mientras que el **Campo de acción** el proceso de control de versiones de una base de datos durante el desarrollo de un producto de software en la UCI.

Para resolver el problema identificado se propone el siguiente **Objetivo general**: Desarrollar una herramienta que gestione las versiones por las que transita una base de datos durante el proceso de desarrollo de un producto de software en la UCI.

Para conseguir este objetivo, se desglosó en los siguientes **Objetivos específicos**:

1. Realizar un estudio de los sistemas de control de versiones existentes, sus algoritmos para la gestión de cambios y su utilización sobre bases de datos.
2. Identificar los principales conceptos presentes en el dominio del problema.
3. Describir los requisitos del sistema a desarrollar.

4. Diseñar las clases para cumplir con los requisitos descritos.
5. Implementar las clases propuestas en el diseño a través de componentes.

Para dar cumplimiento a los objetivos anteriormente planteados se definen las siguientes **Tareas de la investigación**:

1. Realizar un estudio sobre el estado del arte de sistemas de control de versiones.
2. Realizar un estudio sobre el estado del arte de sistemas de gestión de bases de datos.
3. Realizar un estudio sobre el estado del arte del control de versiones sobre bases de datos.
4. Realizar un estudio sobre las tendencias y tecnologías.
5. Realizar un estudio sobre los lenguajes de programación.
6. Realizar un estudio sobre los algoritmos para la gestión de cambios.
7. Realizar un estudio sobre las metodologías de desarrollo.
8. Realizar el modelo de dominio con los conceptos presentes en el proyecto.
9. Describir los requisitos del sistema a desarrollar.
10. Diseñar las clases para cumplir con los requisitos descritos.
11. Implementar las clases diseñadas mediante componentes.

Para un mayor entendimiento el documento estará estructurado de la manera siguiente:

Capítulo 1: Fundamentación Teórica: Estudio preliminar de los sistemas que puedan ser utilizados como controladores de versiones de bases de datos. Estudio de algoritmos para la gestión de cambios. Tecnologías y herramientas de desarrollo a utilizar.

Capítulo 2: Características del sistema: Descripción del negocio a automatizar e información que se maneja. Descripción de los requisitos y casos de uso del sistema.

Capítulo 3: Arquitectura y diseño del sistema: Descripción y análisis de la solución que se propone para dar respuesta a la problemática planteada.

Capítulo 4: Implementación: Estrategias de codificación, descripción del flujo de implementación y estilos a utilizar.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se realiza la descripción de los sistemas de control de versiones y los principales conceptos asociados a estos. Además se exponen las características de los sistemas más populares de este tipo. También se analizan las tecnologías, metodologías y software utilizados en la actualidad. Así como, la fundamentación de la selección de aquellas que fueron empleadas para la presente investigación.

Sistemas de control de versiones

El *Proceso de Desarrollo de Software* "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" (Jacobson, et al., 2000).

En el proceso de desarrollo de software son inevitables los cambios, estos pueden producirse debido a cambios en los requisitos del sistema o por errores cometidos en el proceso como tal. Normalmente los sistemas informáticos se desarrollan en equipos, por lo que es necesario llevar el control y el registro de los cambios con el fin de reducir los errores, aumentar la calidad y la productividad y evitar los problemas que puede acarrear una incorrecta sincronización de dichos cambios, al afectar otros elementos del sistema o las tareas realizadas por otro miembro del equipo. Esta actividad del proceso se conoce como *Gestión de Configuración de Software*.

La Gestión de Configuración de Software es una disciplina que se preocupa de:

- Identificar y documentar las características funcionales y físicas de los elementos de configuración.
- Controlar los cambios a tales características.
- Reportar el proceso de tales cambios y su estado de implantación. (Mejoramiento del Proceso de Gestión, 2004).

La tarea de controlar los cambios por los que pasan los elementos de configuración en un proyecto no es para nada trivial, por lo que el responsable de gestión de configuración generalmente se apoya en

un sistema que automatice dicho proceso. Este tipo de programas es conocido como *sistema de control de código* o *sistema de control de versiones*.

Un sistema de control de versiones es una herramienta de software que administra el acceso a un conjunto de archivos que componen un proyecto, y mantiene un historial de los cambios realizados sobre los mismos y su principal propósito es registrar información del usuario que realizó algún cambio, cuándo y cuál es la diferencia con relación a la versión anterior del mismo archivo. (Sistema de Control de Versiones. Caso de estudio: Subversion, 2008)

Por la disposición en que almacenan los datos los sistemas de control de versiones pueden agruparse en *centralizados* y *distribuidos*. Los *centralizados* son aquellos que funcionan según el paradigma cliente-servidor, o sea que existe un servidor que contiene un repositorio con el que interactúan todos los miembros del grupo de trabajo, ejemplos de este grupo son *Concurrent Versions System* (CVS), Subversion y Visual SourceSafe. En los *distribuidos* como su nombre lo indica no existe un repositorio central, cada uno de los miembros de un proyecto informático tiene en su poder una copia de trabajo y la sincronización de los cambios se hace entre pares, aunque puede existir un súper-par que permita que este grupo funcione de manera similar a los sistemas centralizados, ejemplos de este grupo son el Git y Bazaar. (Sistema de Control de Versiones. Caso de estudio: Subversion, 2008)

Existen dos modelos que definen el funcionamiento de un sistema de control de versiones, el modelo bloquea/modifica/desbloquea conociéndose como sistemas de control de versiones exclusivos o el modelo copia/modifica/fusiona conociéndose como sistema de control de versiones colaborativos, aunque es válido aclarar que gran parte de los sistemas de control de versiones modernos permiten utilizar los dos modelos en dependencia de las configuraciones que hagan los usuarios.

Sistemas de control de versiones exclusivos

Los sistemas de control de versiones exclusivos son aquellos que utilizan el modelo bloquea/modifica/desbloquea (Imagen 1), estos garantizan que no existirán conflictos producto de los cambios realizados por los miembros del equipo de desarrollo sobre ninguno de los elementos de configuración. Su modelo de trabajo les permite trabajar de manera eficiente sobre todos los elementos de configuración independientemente del formato en que estos se encuentren.

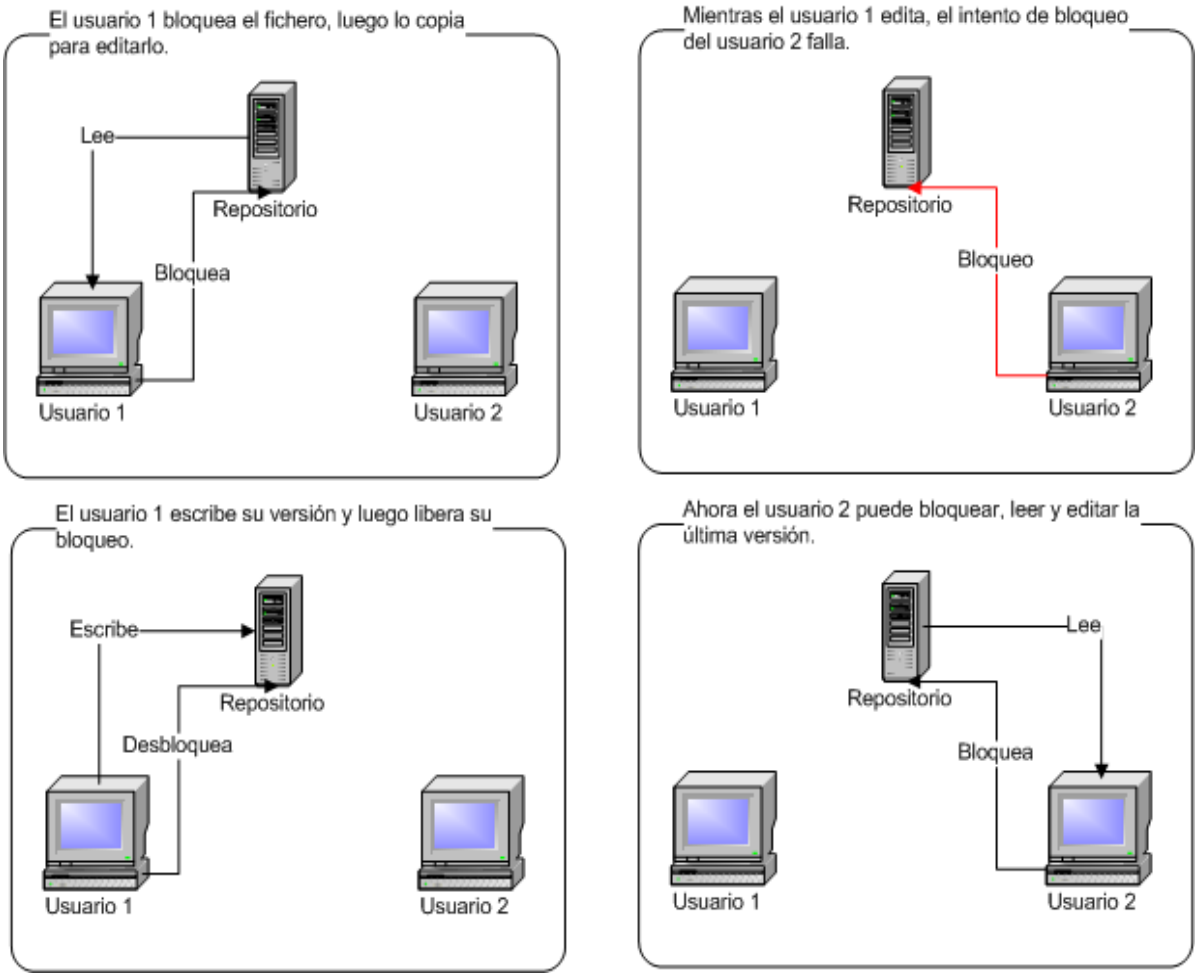


Imagen 1: Modelo de los sistemas de control de versiones exclusivos

La mayor desventaja de este tipo de sistemas de control de versiones radica en su filosofía como tal, ya que el modelo de bloqueo no permite a más de un usuario trabajar sobre el mismo elemento, reduciendo drásticamente la cantidad de tareas que pueden ser llevadas de manera paralela, atentando esto, contra los tiempos de entrega de los sistemas a desarrollar y provocando en gran parte de los casos problemas administrativos. El modelo de bloqueo también tiende a causar procesos en serie innecesarios y puede llegar a causar la falsa sensación de seguridad.

Este tipo de sistemas son mayormente utilizados cuando se piensan realizar cambios muy grandes sobre los elementos de configuración, que puedan traer dificultades en el momento de fusionar los cambios (Tigris.org). Un ejemplo de este grupo de sistemas de control de versiones es el MKS Integrity for Sotfware Configuration Management también conocido como MKS Source Integrity (MKS). Algunos sistemas a pesar de soportar ambos modelos utilizan este como el modelo predefinido, ejemplo de esto es el Visual Source Safe (Microsoft).

Sistemas de control de versiones colaborativos

Los sistema de control de versiones colaborativos son aquellos que utilizan el modelo copia/modifica/fusiona (Imagen 2). La principal ventaja de estos radica en el alto nivel de paralelismo que le brindan al proceso de desarrollo de software ya que permiten que hayan tantos miembros del equipo de desarrollo trabajando sobre el mismo elemento de configuración como sean necesarios.

La principal desventaja de este grupo proviene de los métodos de fusión de cambios, ya que los algoritmos existentes no pueden decidir en los casos en que dos miembros del equipo modifiquen la misma región de un elemento de configuración determinado. Lo que se conoce como conflicto, estos conflictos deben ser solucionados de manera manual y generalmente implican a más de un miembro del equipo. Además, producto de los métodos de fusión, pueden aparecer inconsistencias al fusionar los cambios realizados por miembros del equipo no coordinados.

Este tipo de sistemas no es factible utilizarlo sobre elementos de configuración en formato binario ya que es casi imposible encontrar un método de fusionar los cambios de manera coherente. Aunque la mayoría pueden manejar eficientemente las versiones por las que transcurre un elemento de configuración en este formato.

Ya que en los procesos de desarrollo de software, el elemento de configuración fundamental es el código fuente y este no se encuentra en formato binario, lo que permite que se puedan aplicar sobre este los algoritmos de fusión. Por lo que estos tipos de sistema son los más utilizados en este sentido y son ampliamente recomendados para los grandes equipos de desarrollo gracias al alto nivel de sincronismo que permite en las tareas, sin importar los elementos de configuración que se vean afectados en el proceso. Como ejemplo de este grupo de sistemas de control de versiones están Bazaar, Git. Algunos sistemas soportan ambos modelos y utilizan este como el modelo predefinido como es el caso de Subversion (Tigris.org).

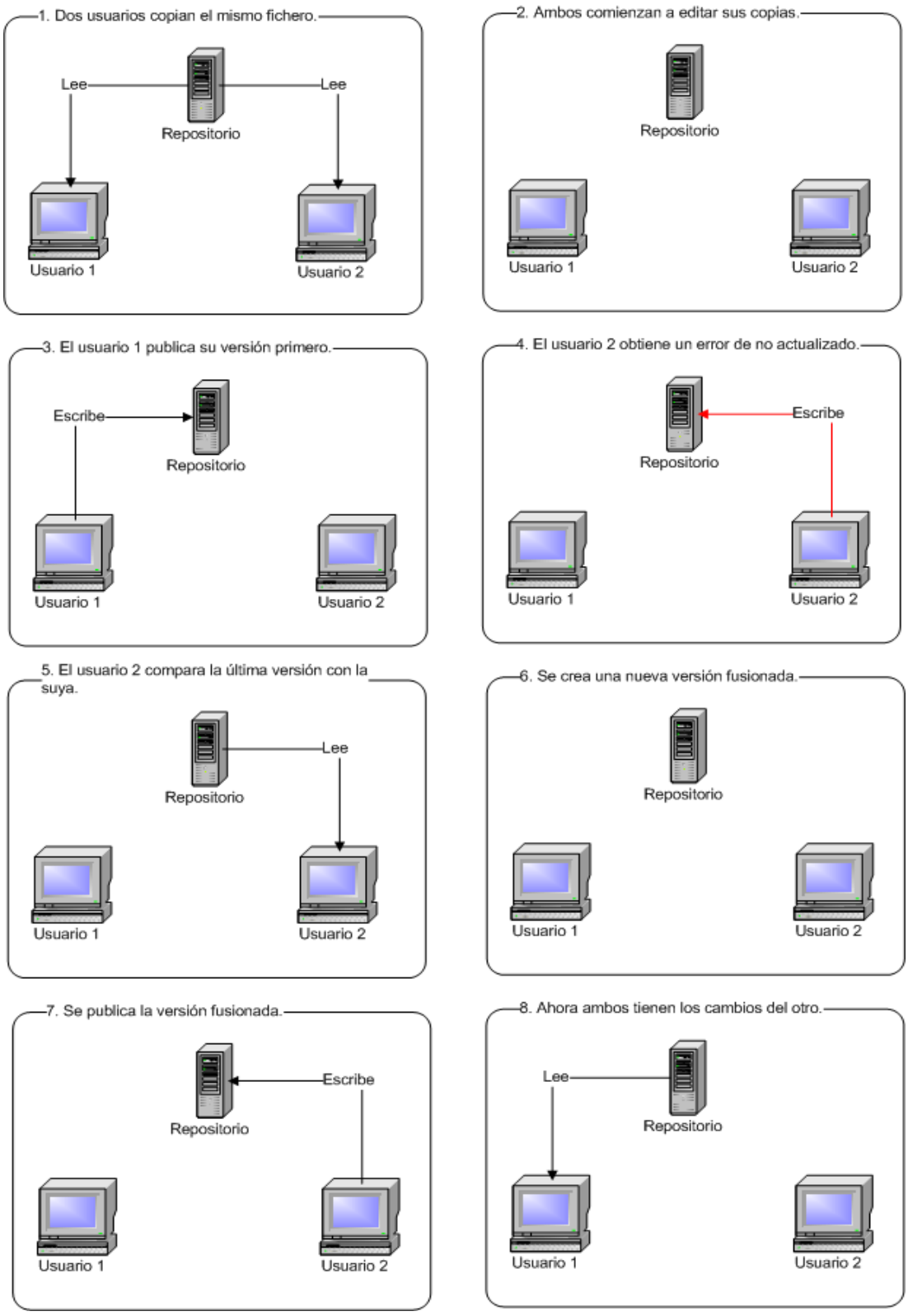


Imagen 2: Modelo de los sistemas de control de versiones colaborativos.

Sistemas de control de versiones utilizados en el mundo

Actualmente existen en el mundo un sinnúmero de sistemas para el control de versiones. Las características de estos difieren por lo que se hace el estudio y análisis de los más utilizados o conocidos en la industria del desarrollo de software.

Visual SourceSafe

Este sistema de control de versiones es un software propietario desarrollado por la Microsoft, es solo compatible con el sistema operativo Windows. Lo cual no solo obliga a sus usuarios a adquirir la licencia del sistema, sino que también los restringe a un entorno de trabajo propietario. A pesar de los costos es un sistema muy usado en las grandes empresas; por la integración que proporciona con el resto de los productos de software proporcionados por sus propietarios y por brindar altos niveles de productividad y eficiencia.

Microsoft Visual SourceSafe es un sistema de control de versiones en el nivel de archivos, que permite a muchos tipos de organizaciones trabajar en distintas versiones de un proyecto al mismo tiempo. Esta funcionalidad es especialmente ventajosa en un entorno de desarrollo de software, donde se usa para mantener versiones de código paralelas. Sin embargo, el producto también se puede utilizar para mantener archivos en cualquier otro tipo de equipo.

Visual SourceSafe incluye, como mínimo, las siguientes funciones:

- Ayuda al equipo a evitar la pérdida accidental de archivos.
- Permite realizar un seguimiento de las versiones anteriores de un archivo.
- Admite la bifurcación, el uso compartido, la combinación y la administración de versiones de archivos.
- Realiza el seguimiento de las versiones de proyectos completos.
- Realiza el seguimiento del código modular (un archivo que se reutiliza, o se comparte, en varios proyectos).

Visual SourceSafe permite compartir archivos entre proyectos de forma rápida y eficaz. La organización de los archivos en proyectos hace que la coordinación de los equipos sea un proceso intuitivo. Cuando se agrega un archivo a Visual SourceSafe, este archivo se almacena en la base de datos y queda a disposición de otros usuarios. Los cambios realizados en él se guardan para que cualquier usuario pueda recuperar una versión anterior en todo momento. Los miembros de su equipo

podrán ver la última versión de un archivo, realizar cambios en sus copias locales y guardar nuevas versiones en la base de datos. Cuando un conjunto de archivos está listo para entregarse, Visual SourceSafe permite compartir y obtener las distintas versiones del conjunto con facilidad. (Microsoft, s.a.)

***Concurrent Versions System* (CVS)**

Este sistema de control de versiones pertenece al movimiento de software libre utilizando una licencia GPL, el cual es mantenido actualmente por su comunidad de desarrolladores, es compatible con los sistemas operativos basados en Unix, con Windows y con Mac Os X.

CVS permite almacenar la historia de los archivos de código y los documentos. Este permite ejecutar scripts con los que se pueden abastecer las trazas del CVS o forzar políticas especiales del sistema.

Su sistema Cliente/Servidor permite mantener unidos como un equipo a desarrolladores separados geográficamente o comunicados mediante lentas conexiones por módem. El historial de las versiones es almacenado en un solo servidor central y los ordenadores de los clientes contienen una copia de todos los ficheros en los que los desarrolladores están trabajando **(GNU)**.

Subversion

Este sistema de control de versiones pertenece al movimiento de software libre utilizando una licencia Apache estilo BSD, fue desarrollado por la empresa CollabNet Inc, es compatible con los sistemas operativos basados en Unix, con Windows y con Mac Os X. Originalmente fue diseñado para ser mejor que CVS por lo que sus características e interfaces son muy similares a las de este.

Subversion es un sistema de control de versiones en el nivel de archivos que controla las carpetas tanto como los ficheros. Incluye las acciones de copiado, renombrado y eliminado como operaciones de versionado. Este sistema permite adicionar propiedades arbitrarias a cualquier fichero o directorio, estas propiedades son pares de llaves con valor y son versionadas junto con los objetos en que están incluidas. También permite adicionar propiedades del tipo llave/valor a una revisión aunque estas propiedades nos son versionadas como las adicionadas a los objetos.

Este sistema implementa el método de revisiones atómicas, o sea, que ninguna parte de la revisión tiene efecto hasta que la revisión no se haya completados satisfactoriamente. Permite el uso de

Branches los cuales son similares a una copia de una parte del proyecto aunque están configurados para ocupar un menor espacio.

A partir de la versión 1.5 de subversión este introduce lo que se conoce como “merge tracking” lo cual es un asistente con manejador de flujo de cambios entre líneas de desarrollos y con mezcla de los Branches con sus raíces. Esta funcionalidad soporta los escenarios más comunes y puede ser extendida en futuras versiones del sistema (Tigris.org).

Bazaar

Este sistema de control de versiones pertenece al movimiento de software libre utilizando una licencia GPL, fue desarrollado por la empresa Canonical Ltd., es compatible con los sistemas operativos basados en Unix, con Windows y con Mac Os X. Es un sistema de control de versiones distribuido lo cual lo hace más flexible para los desarrollos en las comunidades.

Bazaar funciona con un sistema de mezclas en las que solo se envían los cambios necesarios, minimiza los falsos conflictos y no es confundido por cambios idénticos. Permite desarrollar bajo una gana de pruebas, facilitando chequear y verificar ramas de desarrollo en cualquier momento.

Es un sistema fácil de usar, ya que los usuarios se pueden considerar productivos dominando solamente cinco comandos, además, presenta una amplia ayuda para todos los comandos del mismo (Canonical Ltd).

Git

Este sistema de control de versiones pertenece al movimiento de software libre utilizando una licencia GPL, fue desarrollado inicialmente por Linus Torvalds aunque en la actualidad la comunidad de desarrollo es mucho mayor, es compatible con los sistemas operativos basados en Unix, con Posix, con Windows y con Mac Os X. Es un sistema de control de versiones distribuido lo cual lo hace más flexible para los desarrollos en las comunidades.

Como la mayoría de los otros sistemas de control de versiones modernos Git proporciona a cada desarrollador una copia local y los cambios son enviados de un usuario a otro como si este fuera el repositorio. Estos cambios son importados como ramas de desarrollo adicionales y pueden ser mezcladas con la línea de desarrollo local. Los repositorios pueden ser fácilmente accedidos mediante

el protocolo Git. Con el uso de SSL para la autenticación y la seguridad o simplemente HTTP se puede publicar el repositorio local en cualquier lugar sin ninguna configuración inicial del servidor web.

Git soporta una rápida y conveniente división en ramas de desarrollo, con las funcionalidades para la mezcla de las mismas; además de incluir poderosas herramientas para visualizar y navegar por un historial de desarrollo no lineal.

Este sistema es muy rápido y escalable incluso cuando se trabaja con grandes proyectos y largos historiales. Es generalmente más rápido que la mayoría de los otros sistemas de control de versiones. Además utiliza un extremadamente eficiente formato de almacenamiento de paquetes para revisiones de gran tamaño lo que lo que lo sitúa a la cabeza de cualquier otro sistema de control de versiones de código abierto. (Chacon, s.a.)

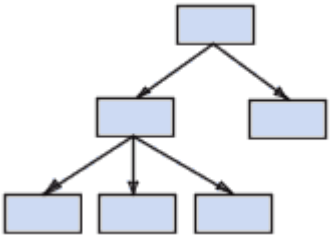
Bases de datos

En la actualidad se destinan muchos recursos con el fin de adquirir, almacenar y procesar datos. Este flujo de información es cada vez mayor, lo que conlleva a la demanda de programas informáticos que manejen esta información constante crecimiento. Para la elaboración de estos programas, los desarrolladores de software deben apoyarse en nuevas tecnologías, que les permitan solucionar problemas cada vez más complejos en tiempos cada vez más cortos.

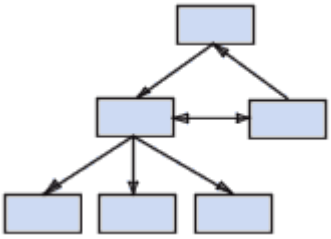
Una de las tecnologías que ha tenido auge producto de esta demanda son las bases de datos. Una base de datos es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un ordenador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha base de datos (Martínez, et al., 2006).

Las bases de datos se pueden clasificar en cinco modelos por la forma en que representan los datos almacenados, estos son:

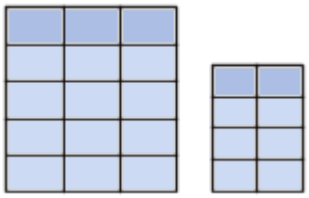
- **El modelo jerárquico**: los datos se organizan jerárquicamente mediante un árbol invertido. Este modelo utiliza punteros para navegar por los datos almacenados. Fue el primer modelo de bases de datos. (CommentCaMarche, 2008)



- **El modelo de red:** al igual que el modelo jerárquico, este modelo utiliza punteros hacia los datos almacenados. Sin embargo, no necesariamente utiliza una estructura de árbol invertido. (CommentCaMarche, 2008)

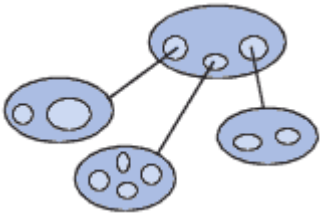


- **El modelo relacional** (**RDBMS**, Relational database management system [Sistema de administración de bases de datos relacionales]): los datos se almacenan en tablas de dos dimensiones (filas y columnas). Los datos se manipulan según la teoría relacional de matemáticas. (CommentCaMarche, 2008) Este es el modelo más utilizado en la actualidad.



- **El modelo deductivo:** los datos se representan como una tabla, pero se manipulan mediante cálculos de predicados. (CommentCaMarche, 2008)

- **El modelo de orientación a objetos** (**ODBMS**, object-oriented database management system [sistema de administración de bases de datos orientadas a objetos]): Los datos se representan como objetos de estructuras denominadas clases. (CommentCaMarche, 2008)



Sistemas gestores de bases de datos

Para el manejo de una base de datos se utiliza lo que se conoce como sistema de gestor de bases de datos. Estos pueden ser definidos como un software que proporciona servicios para la creación, el almacenamiento, el procesamiento y la consulta de la información almacenada en bases de datos de forma segura y eficiente. (Sicilia, 2008)

Estos sistemas gestores de bases de datos además de por el modelo de bases de datos en que se basan se clasifican de la siguiente forma:

Número de usuarios:

- Monousuario.
- Multiusuario.

Número de sitios:

- Centralizados.
- Distribuidos: Homogéneos, Heterogéneos.

Ámbito de aplicación:

- Propósito general.
- Propósito específico.

Sistemas gestores de bases de datos utilizados en el mundo

Actualmente existen en el mundo un sinnúmero de sistemas para la gestión de bases de datos. Las características de estos difieren de uno a otro por lo que se hace el estudio y análisis de los más utilizados o conocidos en la industria del desarrollo de software.

SQLServer

SQL Server 2005 es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial (BI). El motor de la base de datos SQL Server 2005 ofrece almacenamiento más seguro y confiable tanto para datos relacionales como estructurados, lo que le permite crear y administrar aplicaciones de datos altamente disponibles y con mayor rendimiento para utilizar en su negocio.

El motor de datos SQL Server 2005 constituye el núcleo de esta solución de administración de datos empresariales. Asimismo, SQL Server 2005 combina lo mejor en análisis, información, integración y notificación. Esto permite que su negocio cree y despliegue soluciones de BI rentables que ayuden a su equipo a incorporar datos en cada rincón del negocio a través de tableros de comando, escritorios digitales, servicios Web y dispositivos móviles.

La integración directa con Microsoft Visual Studio, el Microsoft Office System y un conjunto de nuevas herramientas de desarrollo, incluido el Business Intelligence Development Studio, distingue al SQL Server 2005. Ya sea que usted se desempeñe como encargado de desarrollo, administrador de base de datos, trabajador de la industria de la información o dirija una empresa, SQL Server 2005 ofrece soluciones innovadoras que le ayudan a obtener más valor de sus datos. (Microsoft, 2006)

PostgreSQL

PostgreSQL es un sistema gestor de bases de datos Objeto-Relacionales basado en POSTGRES Versión 4.2, desarrollado por la Universidad de California en su departamento Berkeley de Ciencias de la Computación. POSTGRES fue pionero en muchos conceptos que solo estuvieron disponibles en algunos sistemas gestores de bases de datos comerciales mucho después. (PostgreSQL Global Development Group, 2005)

PostgreSQL es un sistema de código abierto descendiente del código original de Berkeley. Soporta una gran parte del SQL estándar y ofrece muchas funcionalidades modernas como:

- Consultas complejas
- Llaves foráneas
- Disparadores
- Vistas
- Integridad transaccional
- Control de concurrencia multiversiones

Además, PostgreSQL puede ser extendido por el usuario de muchas formas, por ejemplo se adicionan nuevos:

- Tipos de datos
- Funciones
- Operadores
- Funciones agregadas
- Métodos de indexado
- Lenguajes procedurales

Producto de su licencia BSD, PostgreSQL puede ser usado, modificado y distribuido por cualquiera libre de costo para cualquier propósito, sea privado, comercial o académico.

Oracle

Oracle Database 11*g* Standard Edition es un sistema gestor de bases de datos costeable, con todas las funcionalidades para servidores con más de cuatro núcleos. Este incluye el Oracle Real Application Clusters para una mayor disponibilidad, provee desempeño y seguridad de clase empresarial, es simple de administrar y es fácilmente escalable si la demanda aumenta.

- Su licenciamiento permite comprar solo lo que se necesita y aumentarlo si la demanda cambia.
- Esta soportado por los sistemas Windows, Linux y Unix; puede ser fácilmente administrable de manera automática debido a sus capacidades de auto-administración. (Oracle, s.a.)

Sistemas de control de versiones para bases de datos

Dentro de los elementos de configuración las bases de datos presentan características que las hacen diferentes al resto. Las bases de datos generalmente no se encuentran almacenadas en la computadora del desarrollador sino que están almacenadas en un sistema gestor de base de datos, estos sistemas almacenan las mismas de una forma propia o sea que no existe un estándar que rija este procedimiento. Esto dificulta el proceso de fusión de los cambios y en la mayoría de los cambios se hace imposible encontrar una forma coherente de ejecutar el mismo. Además una base de datos como tal está compuesta por muchos elementos que a su vez podrían convertirse en elementos de configuración, lo cual reduciría el proceso de fusión de cambios, ya este solo ocurriría en caso de que más de un miembro del equipo trabaje sobre el mismo elemento dentro de la base de datos.

Producto de los inconvenientes que se presentan en la actualidad para gestionar las versiones por las que transita una base de datos la mayoría de los proyectos no controlan las mismas, lo que conlleva a que en la actualidad no se pueda controlar que cambios fueron hechos por un desarrollador en un momento determinado del proyecto y que en caso de necesidad no se pueda regresar a un punto determinado del diseño de la base de datos. Por problemas como este se considera que la base de datos siempre debe estar bajo un sistema de control de versiones junto con el código de la aplicación. (Atwood, 2006)

Para poder mitigar los riesgos que implica el no tener versionada la base de datos, en algunos proyectos se utiliza el código SQL de la base de datos como un elemento de configuración a incluir en su sistema de control de versiones. Esto es una solución parcial que permite revertir los cambios en el tiempo; pero dificulta la tarea de saber los cambios realizados por cada desarrollador. Esta solución también trae como deficiencia el hecho de que cada vez que se realicen cambios en la base de datos tenga que generarse el código SQL de esta lo cual implica un gasto en tiempo.

Sistemas de control de versiones para bases de datos utilizados en el mundo

Actualmente existen en el mundo varios de sistema para el control de versiones para bases de datos. Las características de estos difieren de uno a otro, así como los gestores de bases de datos que soportan, por lo que se hace el estudio y análisis de los más utilizados o conocidos en la industria del desarrollo de software.

SASSI v2.0

SASSI v2.0 proporciona una interface de desarrollo integrada Microsoft SQL Server y Microsoft Visual SourceSafe. SASSI permite a los ingenieros colaborar de manera eficiente en el desarrollo de tu base de datos SQL y provee un número de funcionalidades y herramientas fáciles de usar para crear, modificar y manejar los objetos del servidor SQL.

Las funcionalidades más importantes de SASSI v2.0 son:

- Manejo de los procedimientos almacenados, las funciones definidas por el usuario, las vistas y los triggers.
- Control de versiones flexible incluyendo recuperación de objetos eliminados, completa inversión de cambios e historial de los objetos.
- Sincronismo avanzado de objetos entre Microsoft SQL Server y Microsoft SourceSafe.
- Ventana de consultas SQL para pruebas y revisión de código.
- Editor de permisos fácil de usar.
- Poderoso navegador de objetos de la base de datos.
- Generación automático de script de sincronización entre servidores (Dinamic3 LT Consulting GmbH, s.a.)

Idera SQL change manager

Idera SQL change manager es una poderosa solución para simplificar y automatizar el manejo de cambios para bases de datos sobre el Microsoft SQL Server. SQL change manager racionaliza los procedimientos de gestión de cambio de la base de datos con la captura de instantáneas periódicas de los esquema de datos, poniendo de relieve los cambios de una línea de base, y que facilita la reversión de cambios, puesta en marcha y la recuperación de objetos dañados o perdidos. Diseñado para satisfacer las necesidades de las empresas que implementan sobre SQL Server, fácilmente escalable para controlar y automatizar los cambios de esquema en entornos de cualquier tamaño.

Las funcionalidades más importantes de Idera SQL change manager son:

- Gestionar y realizar seguimiento de los cambios en los esquemas.
- Controlar bases de datos desde el desarrolla hasta la producción.
- Aplicar cambios a través de múltiples sistemas de producción.

- Comparar y sincronizar esquemas.
- Revertir cambios en los esquemas.
- Recuperar objetos perdidos o dañados de la base de datos.
- Monitoreo de cambios. (BBS Tecnologies, Inc., s.a.)

ApexSQL Edit

ApexSQL Edit es un poderoso entorno de desarrollo especialmente diseñado para desarrolladores de SQL Server. Este incluye la integración con control de versiones, formato de SQL, edición de datos, mapeo centralizado en control de versiones para equipos de desarrollo, explorador de objetos, notas en objetos, menús configurables y otras funcionalidades que no están presentes en el SQL Server Management Studio.

Las funcionalidades más importantes de ApexSQL Edit son:

- Integración con un sistema de control de versiones.
- Buscador de esquemas de bases de datos.
- Manejador visual de objetos, permitiendo ejecutar un conjunto de operaciones en múltiples objetos.
- Edición de propiedades extendidas.
- Descripciones embebidas a objetos.
- Edición de datos.
- Construcción visual de consultas.
- Menús completamente configurables.

Embarcadero Change Manager

Embarcadero Change Manager ofrece a los administradores de base de datos y a los desarrolladores un poderoso grupo de herramientas para simplificar y automatizar la gestión de cambios en la base de datos. Es un comparador de esquemas y datos, posibilita la sincronización y permite la configuración de reporte para la audición de los cambios efectuados a la base de datos.

Utilizando este los administradores pueden identificar rápidamente los cambios y corregir los problemas en menos tiempo. Monitoreando los ajustes de la configuración los diseñadores de bases

de datos pueden asegurar los diseños con políticas de seguridad y estándares de rendimiento. Change Manager soporta IBM DB2, Microsoft SQL Server, Oracle y Sybase.

Las funcionalidades más importantes de Embarcadero Change Manager son:

- Provee una salva de la base de datos en cualquier momento del desarrollo.
- Comparador de un objeto a muchos.
- Administración de usuarios, servidores y objetos del sistema.
- Mapeo de objetos automatizado.
- Archivado y comparación de los ajustes de configuración de la base de datos.
- Integración con un sistema de control de versiones (Embarcadero Tecnologies, 2008).

Después de analizar estos sistemas se puede constatar que en su mayoría no soportan gestores de bases de datos como MySQL o PostgreSQL. Al añadirle a esto, todos son de carácter propietarios, presentado en su gran mayoría elevados costos para su adquisición. Lo que dificulta la independencia tecnológica que quiere alcanzar el país. Debido a esto, se hace necesaria la creación de un sistema de control de versiones para bases de datos, que pueda ser ajustado a los gestores de bases de datos comúnmente utilizados en la universidad.

Tendencias y tecnologías actuales a considerar

El auge alcanzado en el desarrollo de software empuja cada vez más el desarrollo de tecnologías y metodologías que sean utilizadas en este propósito. Producto de esto, en la actualidad existen un grupo de herramientas que permiten acortar los tiempos de entrega de los productos informáticos, debido a las facilidades que estas brindan. Esta constante evolución de las tecnologías y las metodologías, ha conllevado a que en la actualidad para desarrollar un software se deba tener en cuenta ciertos aspectos, como son las metodologías, los lenguajes de programación y los estándares a utilizar. Como consecuencia de esto a continuación se presenta la propuesta tecnológica realizada para la elaboración de este proyecto.

Metodologias de desarrollo

RUP

El Proceso Racional Unificado o RUP (Rational Unified Process), es un proceso de desarrollo de software que apoyándose en el Lenguaje Unificado de Modelado UML, constituye una de las metodologías estándares más populares para el desarrollo de sistemas orientados a objetos.

Sus principales características se centran en:

- Implementar las mejores prácticas en Ingeniería de Software.
- Forma disciplinada de asignar tareas y responsabilidades, quién hace qué, cuándo y cómo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios y modelado visual del software.

Esta metodología propone un desarrollo iterativo e incremental, dividiendo el proceso de elaboración del software en ciclos y se obtiene un producto final al término de cada uno de estos. Está guiada por los *casos de uso*, los que describen un fragmento de las funcionalidades del sistema, también es centrada en la arquitectura, lo que da a los desarrolladores una mayor visibilidad del sistema.

RUP utiliza como lenguaje de modelado UML (Unified Modeling Language) el mismo permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas “guerras de métodos” que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos. (Grau, 2004)

Extreme Programing (XP)

Es una de las metodologías de desarrollo de software más exitosas de las utilizadas en la actualidad para proyectos de corto plazo, equipo pequeño y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Características de XP, la metodología se basa en:

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara la obtención de posibles errores.
- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

¿Qué es lo que propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua
- El manejo del cambio se convierte en parte sustantiva del proceso
- El costo del cambio no depende de la fase o etapa
- No introduce funcionalidades antes que sean necesarias
- El cliente o el usuario se convierte en miembro del equipo

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales. (¿Qué metodología debo usar para el desarrollo de un Software?, 2004)

Microsoft Solution Framework (MSF)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

MSF tiene las siguientes características:

- Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.
- Tecnología Agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación (¿Qué metodología debo usar para el desarrollo de un Software?, 2004)

Tecnologías de desarrollo

.NET

El corazón de la plataforma.NET es el CLR (Common Language Runtime), que es una aplicación similar a una máquina virtual que se encarga de gestionar la ejecución de las aplicaciones para ella escritas. A estas aplicaciones les ofrece numerosos servicios que facilitan su desarrollo y mantenimiento y favorecen su fiabilidad y seguridad. Entre ellos los principales son:

- Modelo de programación consistente y sencillo, completamente orientado a objetos.
- Eliminación del temido problema de compatibilidad entre DLLs conocido como "infierno de las DLLs"
- Ejecución multiplataforma

- Ejecución multilenguaje, hasta el punto de que es posible realizar acciones como capturar en un programa escrito en C# una excepción escrita en Visual Basic.NET que a su vez hereda de un tipo de excepción escrita en Cobol.NET.
- Recolección de basura.
- Aisla la memoria entre procesos y brinda comprobaciones automáticas de seguridad de tipos en las conversiones.
- Soporte multihilo.
- Gestión del acceso a objetos remotos que permite el desarrollo de aplicaciones distribuidas de manera transparente a la ubicación real de cada uno de los objetos utilizados en las mismas.
- Seguridad avanzada, hasta el punto de que es posible limitar los permisos de ejecución del código en función de su procedencia (Internet, red local, CD-ROM, etc.), el usuario que lo ejecuta o la empresa que lo creó.
- Interoperabilidad con código preexistente, de manera que es posible utilizar con facilidad cualquier librería de funciones u objetos COM y COM+ creados con anterioridad a la aparición de la plataforma .NET. (Seco, 2001)

Java

Java ha sido probado, mejorado y ampliado por una comunidad especializada de más de 6,5 millones de desarrolladores, una de las más activas del mundo. Gracias a su versatilidad, eficiencia y portabilidad, Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma
- Crear programas para que funcionen en un navegador web y en servicios web
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados.
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital. (Sun Microsystem, s.a.)

Framework

Mono

Mono es una plataforma de software diseñada para permitir a los desarrolladores crear fácilmente sus aplicaciones. Es una implementación a código abierto del .Net Framework de la Microsoft basado en el estándar ECMA para C# y el *Common Language Runtime*.

Estos son varios de los componentes que conforman Mono:

- Compilador de C# - Es una funcionalidad que permite compilar C# 1.0 y 2.0 (ECMA), y además contiene muchas de las funcionalidades del C# 3.0.
- Mono Runtime – Implementa la infraestructura de lenguaje común o CLI (*Common Language Infrastructure*). Este provee un compilador a tiempo real, un pre-compilador, un intérprete de librerías, el recolector de basura, el sistema de hilos y funcionalidades de interoperabilidad.
- Base Class Library – La plataforma Mono provee un amplio grupo de clases que proporciona una solida base para construir aplicaciones. Estas clases son compatible con las clases del .Net Framework de Microsoft.
- Mono Class Library – Mono además provee librerías que van fuera de las clases básicas proporcionadas por la Microsoft. Estas funcionalidades son muy útiles, sobre todo para implementar aplicaciones para Linux. Algunos ejemplos son clases para Gtk+, archivos Zip, LDAP, OpenGL, Cairo, POSIX, etc.

Beneficios que se obtienen seleccionando Mono para el desarrollo de aplicaciones:

- Popularidad – Implementar en .Net, hay millones de desarrolladores que tienen experiencia realizando aplicaciones en C#. Existen alrededor de diez mil libros, páginas web, tutoriales y ejemplos de código para ayudar con cualquier problema imaginado.
- Programación de alto nivel – Todos los lenguajes de Mono se benefician con las funcionalidades del CLR, como el manejo automático de memoria, reflexión, genericidad y trabajo con hilos. Estas funcionalidades permiten al desarrollador concentrarse solo en conformar su aplicación.
- Base Class Library - Contiene una librería con miles de clases para incrementar la productividad.

- Multiplataforma – Mono está construido para ser multiplataforma. Mono trabaja sobre Linux, Microsoft Windows, Mac OS X, Sun Solaris, Nintendo Wii y Apple IPhone. Además trabaja sobre x86, x86-64, IA64, PowerPC, SPARC (32), ARM, Alpha, s390, s390x (de 32 y 64 bits) y más. Las aplicaciones desarrolladas con Mono están cerca de ser ejecutadas en cualquier computadora que exista. (Novell, s.a.)

Microsoft .NET Framework versión 2.0

Microsoft .NET Framework versión 2.0 Redistributable Package instala el entorno en tiempo de ejecución y los archivos asociados de .NET Framework necesarios para ejecutar aplicaciones desarrolladas para .NET Framework v2.0.

.NET Framework versión 2.0 mejora la escalabilidad y el rendimiento de aplicaciones gracias a características mejoradas como el almacenamiento en caché, el desarrollo de aplicaciones y la actualización con ClickOnce; además, es compatible con la gama más amplia de exploradores y dispositivos con servicios y controles ASP.NET 2.0.

Lenguajes de programación

C#

C# es el lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg. Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es **el lenguaje nativo de .NET.**

Características que definen al lenguaje de programación C#:

- **Modernidad**

  Al ser C# un lenguaje de última generación, incorpora elementos como tipos decimales o booleanos, un tipo básico *string*, así como una instrucción que permita recorrer colecciones con facilidad (instrucción *foreach*). Estos elementos hay que simularlos en otros lenguajes como C++ o Java.

- **Orientado a objetos**

  C# como lenguaje de última generación, y de propósito general, es orientado a objetos.

- **Recolección de basura**

  Como ya se comentó, todo lenguaje incluido en la plataforma .NET tiene a su disposición el recolector de basura del CLR.

- **Instrucciones seguras**

  En C# se han impuesto una serie de restricciones en el uso de instrucciones de control más comunes. Por ejemplo, la evaluación de toda condición ha de ser una expresión condicional y no aritmética. Así se evitan errores por confusión del operador igualdad con el de asignación. Otra restricción que se impone en la instrucción de selección *switch*, imponiendo que toda selectora de la instrucción finalice con una instrucción *break* o *goto* que indique cuál es la siguiente acción a realizar.

- **Unificación de tipos**

  En C# todos los tipos derivan de una superclase común llamada System.Object, por lo que automáticamente heredarán todos los miembros definidos en esta clase. A diferencia de Java, en C# esta característica también se aplica para los tipos básicos.

- **Extensión de los operadores básicos**

  Para facilitar la legibilidad de código y conseguir que los nuevos tipos de datos que se definan a través de las estructuras estén al mismo nivel que los elementos predefinidos en el lenguaje, al igual que C++ pero a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores (incluidos el de la conversión) cuando se apliquen a diferentes tipos de objetos.

- **Eficiente**

  En C#, todo el código incluye numerosas restricciones para garantizar su seguridad, no permitiendo el uso de punteros. Sin embargo, y a diferencia de Java, existen modificadores

para saltarse esta restricción, pudiendo manipular objetos a través de punteros. Para ello basta identificar regiones de código con el identificador *unsafe*, y podrán usarse en ellas punteros de forma similar a como se hace en C++. Esta característica puede resultar de utilidad en situaciones en las que se necesite gran velocidad de procesamiento.

Algunas de estas características no son propias del lenguaje, sino de la plataforma .NET, aunque se listan aquí ya que tienen una implicación directa en el lenguaje (Mayoral, 2005).

Visual Basic .Net

Visual Basic .Net posibilita a los desarrolladores niveles de control y productividad sin precedentes. Cuenta con total acceso a la plataforma .NET de Microsoft permitiendo construir aplicaciones de escritorio tradicionales, aplicaciones Web, servicios web de nueva generación y software para móviles.

Visual Basic .NET incluye las siguientes características:

- Herencia.
- Control de excepciones.
- Sobrecarga.
- Redefinición de propiedades y métodos.
- Constructores y destructores.
- Tipos de datos.
- Interfaces.
- Delegados.
- Miembros compartidos.
- Referencias.
- Espacios de nombres.
- Ensamblados.
- Atributos.
- Subprocesamiento múltiple.
- Operadores de desplazamiento.
- Declaración de variables de bucle. (Paredes, 2005)

Entorno de desarrollo (IDE)

SharpDevelop

SharpDevelop (#Develop) es un entorno de desarrollo (IDE) gratuito y de código abierto para C# y VB.NET bajo la plataforma Microsoft .NET.

Entre sus características están:

- Diseñador de formularios para C#, VB. NET y Boo.
- Completa el código para C#, VB.NET y Boo (Soporta CTRL + espacio).
- Integra un debugger.
- Análisis de código con FxCop.
- Soporte Multi-framework (.NET 1.1 y 2.0, Mono, Compact Framework)
- Edición XML Editing con búsqueda XPath (WebAdictos).

Microsoft Visual Studio 2008

Visual Studio es un completo conjunto de herramientas para la creación tanto de aplicaciones de escritorio como de aplicaciones web empresariales para trabajo en equipo. Aparte de generar aplicaciones de escritorio de alto rendimiento, se pueden utilizar las eficaces herramientas de desarrollo basado en componentes y otras tecnologías de Visual Studio para simplificar el diseño, desarrollo e implementación en equipo de soluciones empresariales. (Microsoft, 2007)

Entre sus características están:

- Construir aplicaciones utilizando la última tecnología web con soporte para AJAX y controles web incluidos en la *Microsoft AJAX Library*.
- Construir aplicaciones web fácilmente con soporte para CSS, JavaScript y con vistas de código y diseño separadas.
- Maneja y construye aplicaciones que utilizan múltiples versiones del .NET Framework con una sola herramienta.
- Crea aplicaciones conectadas usando el nuevo diseñador visual para *Windows Communication Foundation* y *Windows Workflow Foundation* (Microsoft, 2008).

XML

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. (W3C, 2008)

Conclusiones

Al realizar un análisis de los sistemas existentes para el control de versiones para bases de datos, se pudo concluir que la posibilidad de utilizar estos sistemas en la UCI e incluso en el país es casi nula. Debido a que van contra la política de independencia tecnológica trazada por la dirección del país. Además, los que podrían ser utilizados solo soportan bases de datos de gestores propietarios lo que incrementa los costos de desarrollo de un proyecto determinado.

Todo lo anterior, indica que es necesario desarrollar un sistema de control de versiones para bases de datos, que se adapte a las características del proceso de desarrollo de software de la UCI. Además que sea flexible para la variedad de gestores que se utilicen en este centro, aunque el marco de esta investigación solo brinda la configuración necesaria para su utilización sobre el gestor de bases de datos PostgreSQL.

Para la realización de este sistema en el menor tiempo posible y minimizando sus costos, se determinó utilizar las siguientes herramientas para el desarrollo del mismo:

- RUP como metodología de desarrollo de software.
- .Net como tecnología de desarrollo de software.
- Mono como plataforma de software.
- C# como lenguaje de programación.
- SharpDevelop como entorno de desarrollo.
- XML como y transmisión formato de almacenamiento de datos.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se realiza la descripción de la solución propuesta para el sistema de control de versiones para bases de datos ClioBD. Al no existir una definición clara de los procesos del negocio en el que se desplegara el sistema, se determina desarrollar un Modelo de Dominio, donde se abarcan las definiciones asociadas a la aplicación y las relaciones definidas entre ellas. Se enuncian los requisitos del sistema y los casos de uso del sistema que abarquen estos requisitos agrupados en un diagrama de casos de uso del sistema.

Modelo de Dominio

El Modelo de Dominio o Modelo Conceptual es una representación visual de los conceptos u objetos que se manejan en el dominio del sistema. Los objetos o conceptos incluidos en el modelo de dominio no describen clases u objetos del software; sino entidades o conceptos del mundo real que están asociadas al problema en cuestión. Dicho modelo podrá ser utilizado como una base de las abstracciones relevantes en el proceso de construcción del sistema.

Conceptos fundamentales del dominio

Con la finalidad de una mejor comprensión del Diagrama del Modelo de Dominio a continuación se dará una breve descripción de los conceptos encontrados en el ámbito del problema. Se debe recordar que estos conceptos estarán en gran medida asociados a los conceptos existentes en las bases de datos actuales. Estos son:

Atributo: Es una agrupación que incluye un nombre y un valor, es la abstracción más pequeña que existe asociada al problema y en general se utiliza para componer abstracciones más complejas.

Objeto: Representa un componente de la base de datos como las tablas o funciones. Tiene atributos que defines sus valores básicos y conforma el mismo una estructura arbórea ya que un objeto puede contener otros objetos que se consideren hijos de este.

Adición: Es la acción de crear un nuevo objeto en la base de datos.

Eliminación: Es la acción de eliminar un objeto existente en la base de datos.

Modificación: Es la acción de cambiar los valores de los atributos de un objeto presenta en la base de datos.

Cambio: Es la abstracción que representa cualquiera de las posibles variaciones (adición, eliminación y modificación) que se pueden encontrar en un proceso de diseño de base de datos sobre un objeto de la misma.

Usuario: Es la persona que realiza el proceso de diseño de la base de datos y a su vez es el responsable de los cambios que realiza en la misma.

Versión: Es el conjunto de cambios realizados por un usuario en un momento determinado del proceso de diseño de la base de datos.

Historial: Es el conjunto de versiones por las que ha transcurrido una base de datos en el proceso de elaboración de esta.

Diagrama del Modelo de dominio

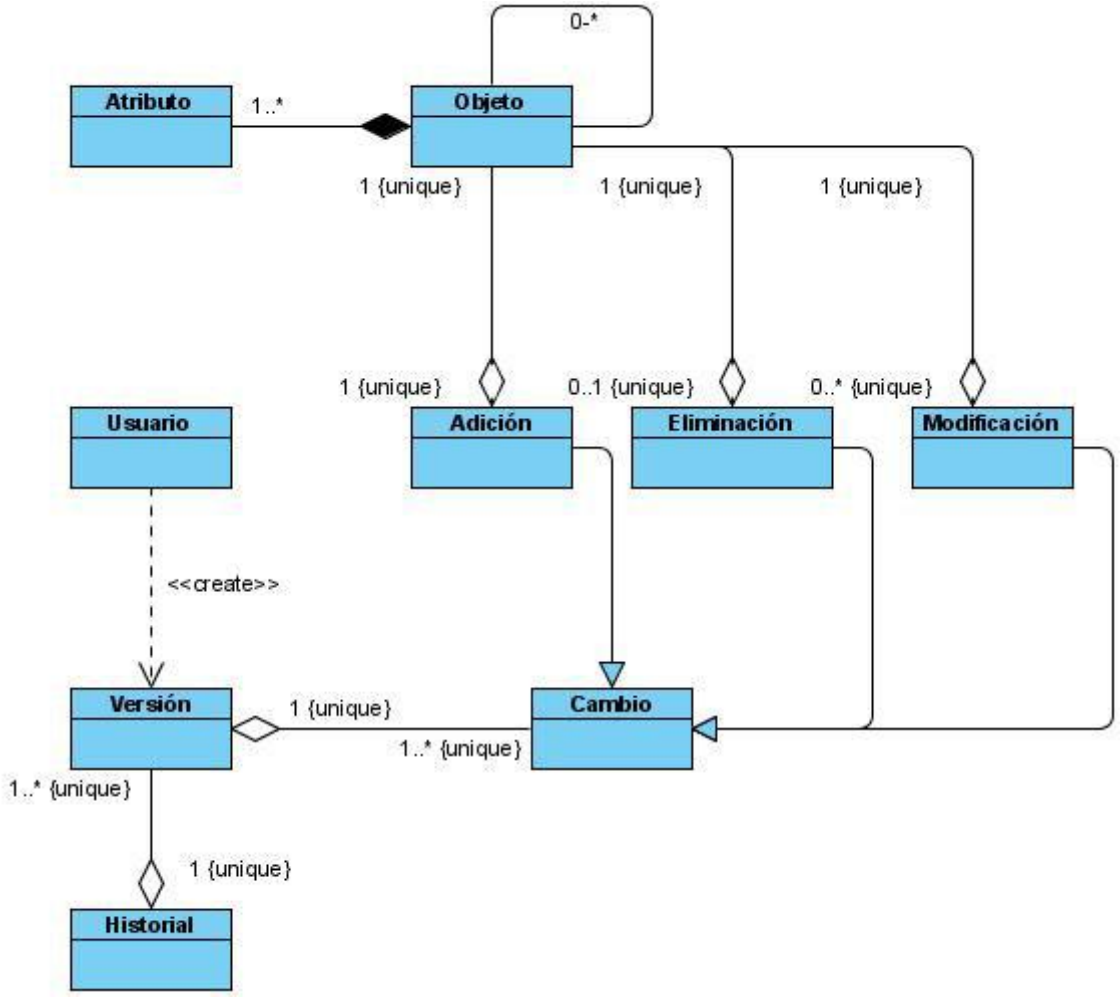


Imagen 3: Diagrama del Modelo de Dominio

Propuesta de Sistema

El objetivo de esta sección es diseñar un sistema capaz de controlar las versiones por las que transcurre una base de datos en su proceso de elaboración. Este sistema servirá de soporte al proceso de desarrollo de software en particular al diseño de bases de datos.

Especificación de los requisitos de software

Un requisito de software podría definirse como una condición o capacidad que debe encontrarse o estar en un sistema o componente para satisfacer un contrato, norma, especificación u otro documento

impuesto formalmente. El conjunto de todas las necesidades es el fundamento para el consiguiente desarrollo del sistema o componente. (IEEE, 2007)

Requisitos Funcionales

El sistema deberá permitir:

RF 1 - Autenticar usuario

RF 1.1 - Autenticar usuario en el gestor de bases de datos local

RF 1.2 - Autenticar usuario en el gestor de bases de datos del servidor

RF 2 - Crear copia local de la base de datos

RF 3 - Consultar estructura actual de la base de datos local

RF 4 - Almacenar estructura de la base de datos local

RF 5 - Buscar estructuras de base de datos anteriores

RF 6 - Comparar estructuras de bases de datos

RF 7 - Consultar cambios de estructura de la base de datos

RF 8 - Almacenar historial de cambios de estructura de la base de datos

RF 9 - Modificar diseño de base de datos

RF 10 - Crear respaldo del historial de versiones

RF 11 - Restaurar respaldo del historial de versiones

RF 12 - Comprimir ficheros

RF 13 - Descomprimir ficheros

RF 14 – Mezclar cambios

Requisitos No Funcionales

Requisitos de apariencia o interface externa

RNF 1 – La interface debe ser sencilla y amigable al usuario. Diseño sencillo, con pocas entradas de datos, posibilitando que el entrenamiento para el uso del sistema sea mínimo.

Requisitos de usabilidad

RNF 2 – El sistema debe brindar un acceso fácil y rápido a todas las funcionalidades, permitiendo que sea usado por usuarios con una preparación mínima.

Requisitos de rendimiento

RNF 3 – El sistema debe permitir que se conecten múltiples usuarios sincrónicamente, garantizando tiempos de respuesta mínimos en todos los casos.

Requisitos de soporte

RNF 4 – El sistema debe contar con un mantenimiento periódico a su servidor, garantizando la integridad de los datos almacenados en el mismo.

Requisitos de portabilidad

RNF 5 – El sistema podrá ser utilizado bajo cualquier arquitectura o sistema operativo para el cual exista una implementación del estándar .Net.

Requisitos de confiabilidad

RNF 6 – El sistema debe trasmitir la información mediante canales seguros. Los procesos de modificación de información deberán ser transaccionales y se chequeara constantemente la integridad de los datos.

Requisitos de hardware

RNF 7.1 – Requerimientos mínimos para clientes:

- Ordenador Pentium IV o superior.
- 512 MB de Memoria RAM o superior.
- Disco duro de 10 GB o superior.

RNF 7.2 – Requerimientos mínimos para el servidor:

- Ordenador Pentium IV o superior.
- 1 GB de Memoria RAM o superior.
- Disco duro de 40 GB o superior.

Requisitos de software

RNF 8 *– Tanto el cliente como el servidor deben tener Mono versión 2.2 y un sistema operativo Linux, Windows o Unix.*

Diagrama de casos de uso del sistema

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema (HispaNetwork, 2006). Los diagramas de casos de uso permiten que los desarrolladores y los clientes lleguen a un entendimiento sobre las condiciones y el alcance del sistema antes de emprender la fase de construcción del mismo. El sistema a desarrollar cuenta con dos partes bien definidas: el cliente del sistema de control de versiones que deberá estar instalado en los ordenadores de los diseñadores de base de datos y el servidor del sistema de control de versiones que será el que se encargará de centralizar la información las funcionalidades de estos fueron modeladas individualmente.

Imagen 2 Diagrama de paquetes del sistema

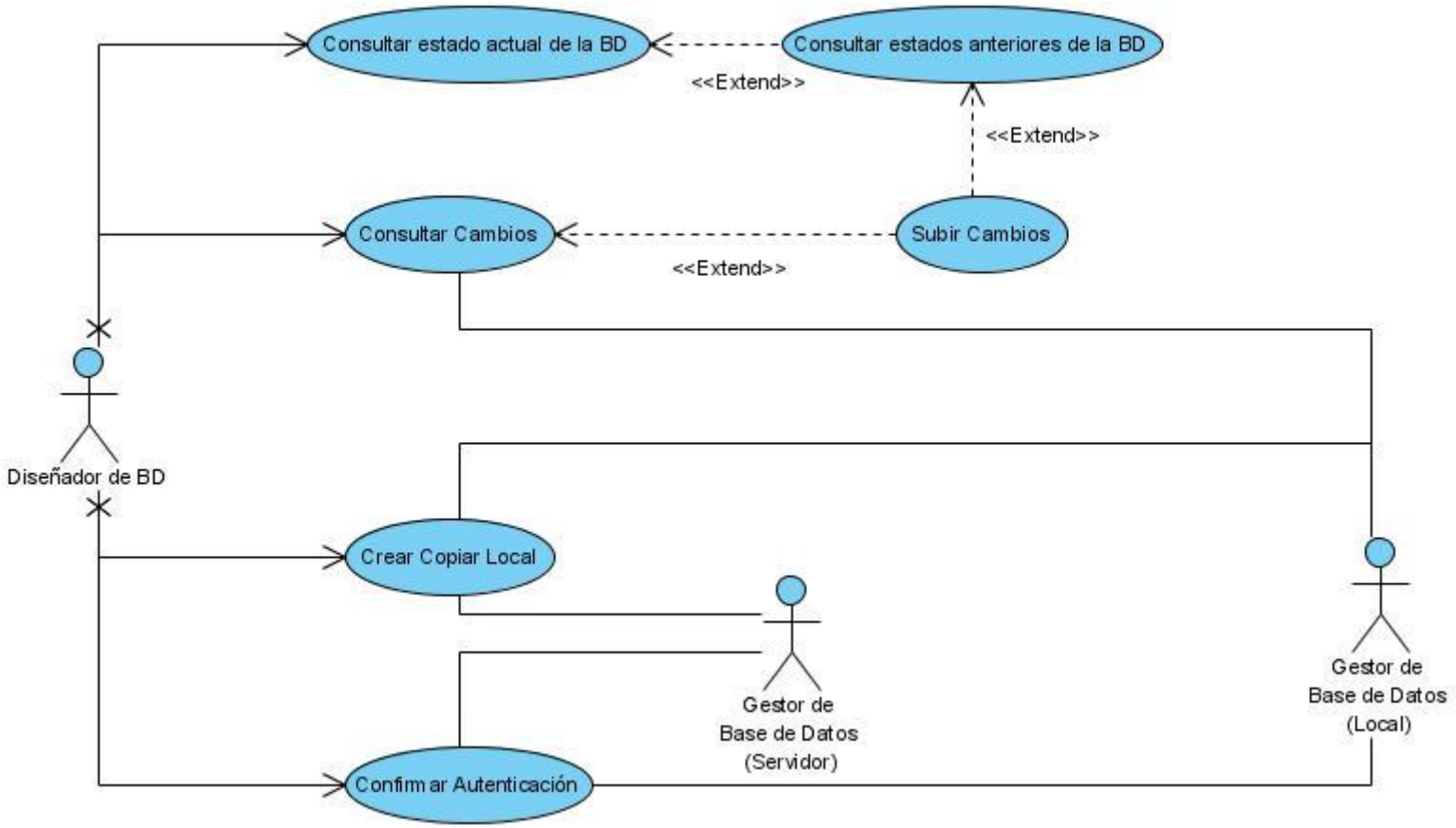


Imagen 3 Diagrama de CU Paquete Cliente

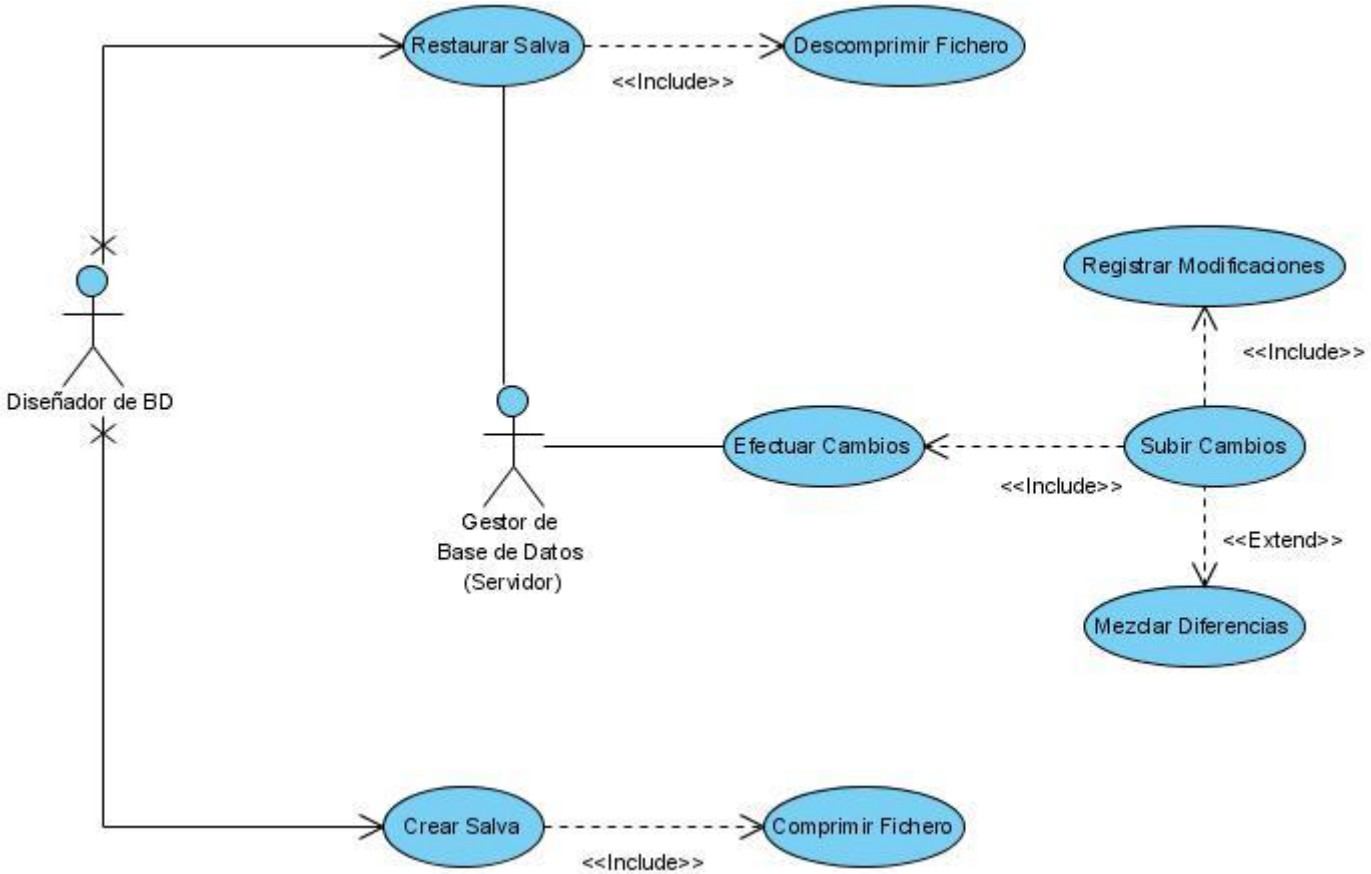


Imagen 4 Diagrama de CU Paquete Servidor

Descripción textual de los casos de uso del sistema

La descripción textual de un caso de uso describe los procesos o flujos de actividades que son objeto de automatización en el mismo, lo que incluye la forma de interacción de los actores con este. A continuación se da una breve descripción de los casos de uso identificados para el sistema ClioBD.

CU-1	Confirmar Autenticación
Actor	Diseñador de BD
Propósito	Definir la identidad del usuario, así como dar al sistema acceso a las bases de datos local y central.
Resumen	El Caso de Uso comienza cuando el actor accede al sistema e ingresa sus credenciales de acceso para los servidores de bases de datos local y central, accediendo al sistema. El caso de uso termina cuando el actor haya accedido al sistema.
Referencia	**RF 1, RF 1.1, RF 1.2**
Precondiciones	El usuario debe conocer las credenciales de acceso a la base de datos local y central.
Poscondiciones	Permite el acceso al sistema de control de versiones. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 1 Descripción textual CU-1: Confirmar Autenticación

CU-2	Crear Copia Local
Actor	Diseñador de BD
Propósito	Brindarle al usuario la posibilidad de obtener una copia de una base de datos existente en el servidor del sistema de control de versiones.
Resumen	El Caso de Uso comienza cuando el actor accede a la opción de Crear Copia Local y este selecciona la base de datos deseada. El sistema crea una copia local de la base de datos seleccionada por el usuario y termina el caso de uso.
Referencia	**RF 2**
Precondiciones	El usuario debe estar autenticado y tener permisos de lectura y escritura en la base de datos seleccionada.
Poscondiciones	El sistema crea una copia de la base de datos central en el servidor de bases de datos local. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 2 Descripción textual CU-2: Crear Copia Local

CU-3	Consultar Cambios
Actor	Diseñador de BD
Propósito	Mostrarle al usuario los cambios ejecutados en la base de datos local que no han sido enviados a la base de datos central.
Resumen	El Caso de Uso comienza cuando el actor accede al sistema. El sistema muestra al usuario los cambios ejecutados en la base de datos local que no han sido enviados a la base de datos central, dándole la posibilidad de enviarlos y termina el caso de uso.
Referencia	**RF 3, RF 6, RF 7**
Precondiciones	El usuario debe estar autenticado y tener permisos de lectura en la base de datos local.
Poscondiciones	El sistema muestra al usuario los cambios ejecutados en la base de datos local que no han sido enviados a la base de datos central. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 3 Descripción textual CU-3: Consultar Cambios

CU-4	Subir Cambios [Extendido CU-3: Consultar Cambios]
Actor	Diseñador de BD
Propósito	Efectuar los cambios hechos en la base de datos local sobre la base de datos central.
Resumen	El Caso de Uso comienza cuando el actor selecciona uno o más objetos que hayan tenido cambios en la base de datos local y accede a la opción Subir Cambios. El sistema ejecuta los cambios en la base de datos central y termina el caso de uso.
Referencia	**RF 4, RF 9**
Precondiciones	El usuario debe estar autenticado y tener permisos de escritura en la base de datos central.
Poscondiciones	El sistema ejecuta los cambios en la base de datos central. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 4 Descripción textual CU-4: Subir Cambios

CU-5	Consultar Estado Actual de la BD
Actor	Diseñador de BD
Propósito	Mostrar al usuario el esquema actual de la base de datos.
Resumen	El Caso de Uso comienza cuando el actor accede a la opción Consultar Estado Actual de la BD. El sistema muestra el esquema actual de la base de datos y brinda la posibilidad de buscar esquemas anteriores, el caso de uso termina.
Referencia	**RF 3**
Precondiciones	El usuario debe estar autenticado y tener permisos de lectura en la base de datos local.
Poscondiciones	El sistema muestra la estructura actual de la base de datos. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 5 Descripción textual CU-5: Consultar Estado Actual de la BD

CU-6	Consultar Estados Anteriores de la BD [Extendido CU-5: Consultar Estado Actual de la BD]
Actor	Diseñador de BD
Propósito	Brindarle al usuario una comparación entre el esquema actual de la base de datos con esquemas anteriores de esta misma base de datos.
Resumen	El Caso de Uso comienza cuando el actor accede a la opción Consultar Estados Anteriores de la BD y selecciona la versión con la que quiere comparar. El sistema muestra una comparación entre el esquema actual de la base de datos y el esquema que existía en la versión seleccionada y brinda la posibilidad de regresar la base de datos a este esquema, el caso de uso termina.
Referencia	**RF 5, RF 6, RF 9**
Precondiciones	El usuario debe estar autenticado y tener permisos de lectura en la base de datos local.
Poscondiciones	El sistema muestra la estructura actual de la base de datos. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 6 Descripción textual CU-6: Consultar Estados Anteriores de la BD

CU-7	Registrar Modificaciones [Incluido CU-4: Subir Cambios]
Actor	Diseñador de BD
Propósito	Registrar las modificaciones hechas por el usuario en la base de datos central.
Resumen	El Caso de Uso comienza cuando se efectúan cambios en la base de datos central. El sistema almacena los cambios y el usuario que los realiza, el caso de uso termina.
Referencia	**RF 8**
Precondiciones	El usuario debe estar autenticado y tener permisos de escritura en la base de datos central.
Poscondiciones	El sistema almacena los cambios realizados por el usuario. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 7 Descripción textual CU-7: Registrar Modificaciones

CU-8	Efectuar Cambios [Incluido CU-4: Subir Cambios]
Actor	Diseñador de BD
Propósito	Realizar las modificaciones hechas por el usuario en la base de datos central.
Resumen	El Caso de Uso comienza cuando se efectúan cambios en la base de datos central. El sistema realiza los cambios en la base de datos centra, el caso de uso termina.
Referencia	**RF 9**
Precondiciones	El usuario debe estar autenticado y tener permisos de escritura en la base de datos central.
Poscondiciones	El sistema ejecuta en la base de datos central los cambios realizados por el usuario. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 8 Descripción textual CU-8: Efectuar Cambios

CU-9	Crear Salva
Actor	Diseñador de BD
Propósito	Brindarle al usuario una salva de todo el historial de las versiones por las que han transcurrido las bases de datos versionadas por el sistema.
Resumen	El Caso de Uso comienza cuando el actor accede a la opción Crear Salva. El sistema crea un archivo con el historial de las bases de datos versionadas por este, el caso de uso termina.
Referencia	**RF 5, RF 10**
Precondiciones	El usuario debe estar autenticado y tener permisos de lectura en la base de datos central.
Poscondiciones	El sistema crea un archivo con el historial de las bases de datos versionadas por este. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 9 Descripción textual CU-9: Crear Salva

CU-10	Restaurar Salva
Actor	Diseñador de BD
Propósito	Brindarle al usuario la opción de restaurar un sistema de versionado sobre bases de datos a partir de una salva realizada con anterioridad.
Resumen	El Caso de Uso comienza cuando el actor accede a la opción Restaurar Salva y selecciona el archivo de salva a restaurar. El sistema se reestructura a la configuración existente al momento de crearse dicha salva, el caso de uso termina.
Referencia	**RF 9, RF 11**
Precondiciones	El usuario debe estar autenticado y tener permisos de escritura en la base de datos central.
Poscondiciones	El sistema se reestructura a la configuración existente al momento de crearse la salva. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 10 Descripción textual CU-10: Restaurar Salva

CU-11	Comprimir Fichero [Incluido CU-9: Crear Salva]
Actor	Diseñador de BD
Propósito	Comprimir en un solo archivo todos los archivos asociados a una salva del sistema.
Resumen	El Caso de Uso comienza cuando se crea una salva del sistema. El sistema crea un fichero comprimido con todos los archivos asociados a una salva, el caso de uso termina.
Referencia	**RF 9, RF 11**
Precondiciones	El usuario debe estar autenticado y tener permisos de lectura en la base de datos central.
Poscondiciones	El sistema crea un fichero comprimido con todos los archivos asociados a una salva. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 11 Descripción textual CU-11: Comprimir Fichero

CU-12	Descomprimir Fichero [Incluido CU-10: Restaurar Salva]
Actor	Diseñador de BD
Propósito	Descomprimir los archivos asociados a una salva del sistema.
Resumen	El Caso de Uso comienza cuando se restaura una salva del sistema. El sistema descomprime todos los archivos asociados a una salva, el caso de uso termina.
Referencia	**RF 9, RF 11**
Precondiciones	El usuario debe estar autenticado y tener permisos de escritura en la base de datos central.
Poscondiciones	El sistema descomprime todos los archivos asociados a una salva. Si esta operación no ocurre el sistema mostrara un mensaje de error.

Tabla 12 Descripción textual CU-12: Descomprimir Fichero

CU-13	Mezclar Cambios [Extendido CU-4: Subir Cambios]
Actor	Diseñador de BD
Propósito	Unificar los cambios hechos por dos desarrolladores diferentes sobre un mismo diseño de base de datos.
Resumen	El Caso de Uso comienza cuando se suben cambios a la base de datos central y estos solapan cambios hechos por otro desarrollador. El sistema muestra una posible unión de los cambios desarrollados por ambos desarrolladores, el caso de uso termina.
Referencia	**RF 14**
Precondiciones	El usuario debe estar autenticado y tener permisos de escritura en la base de datos central.
Poscondiciones	El sistema unifica los cambios hechos por dos desarrolladores diferentes sobre un mismo diseño de base de datos.

Tabla 13 Descripción textual CU 13: Mezclar Cambios

Conclusiones

En el presente capítulo se obtuvo una descripción de la solución propuesta para el sistema de control de versiones para bases de datos ClioBD. Se desarrolló el Modelo de Dominio de dicho sistema, donde se abarcaron las definiciones asociadas a la aplicación y las relaciones definidas entre ellas. Se enunciaron los requisitos del sistema y los casos de uso del sistema que abarcan estos requisitos agrupándolos en un diagrama de casos de uso del sistema.

CAPÍTULO 3: ARQUITECTURA Y DISEÑO DEL SISTEMA

En el presente capítulo se definen las clases de diseño del software, así como sus atributos y responsabilidades. Entre los artefactos que se modelarán en este están los modelos de diseño, explicándose la estructura y la definición de elementos que este posee. También se definen los diagramas de clases del diseño de los casos de uso arquitectónicamente significativos y la descripción de las clases presentes en estos diagramas.

Modelo arquitectónico

Una de las formas en que se expresan los estilos arquitectónicos es mediante las cuatro C: los componentes o elementos que forman el sistema; las conexiones que existen entre estos elementos ya sean dispositivos de redes, protocolos de comunicación u otros; las restricciones tanto de comportamiento o como de características y finalmente la configuración que rige la unión de los elementos.

ClioBD es un software con una arquitectura cliente servidor como se puede observar en el diagrama de interacción de los componentes del sistema. Entre los elementos presentes en este diagrama se destacan la existencia de una copia de la base de datos por cada cliente, se debe recordar que la creación de copias locales es una de las características que defines los sistema de control de versiones colaborativos. También se puede observar como los clientes están separados del servidor, como lo define el estilo arquitectónico predominante, ya que esto permite que exista una alta cohesión y un bajo acoplamiento de los elementos y así puedan interactuar entre si y coexistir uno en ausencia de otro.

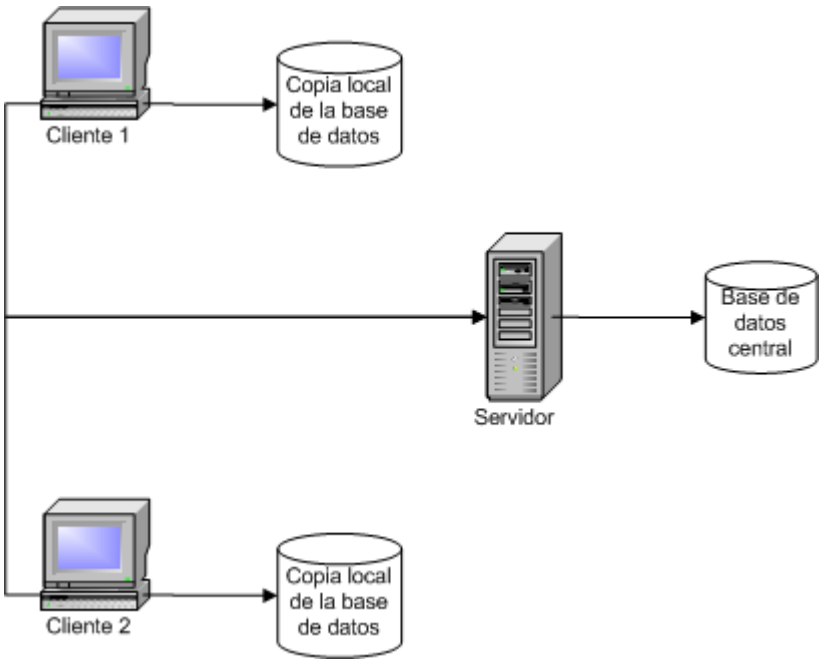


Imagen 5 Diagrama de interacción entre los componentes del sistema ClioBD

Modelo de diseño

El modelo de diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a implementación. Representa a los casos de uso en el dominio de la solución.

Estructuración

Previo al desarrollo del Modelo de Diseño es preciso definir la descomposición de este en diagramas de paquetes donde se incluirán las implementaciones de los casos de uso definidos en el sistema. En el caso del sistema ClioBD los diagramas de paquetes vienen definidos por los paquetes de casos de uso existentes en el sistema.

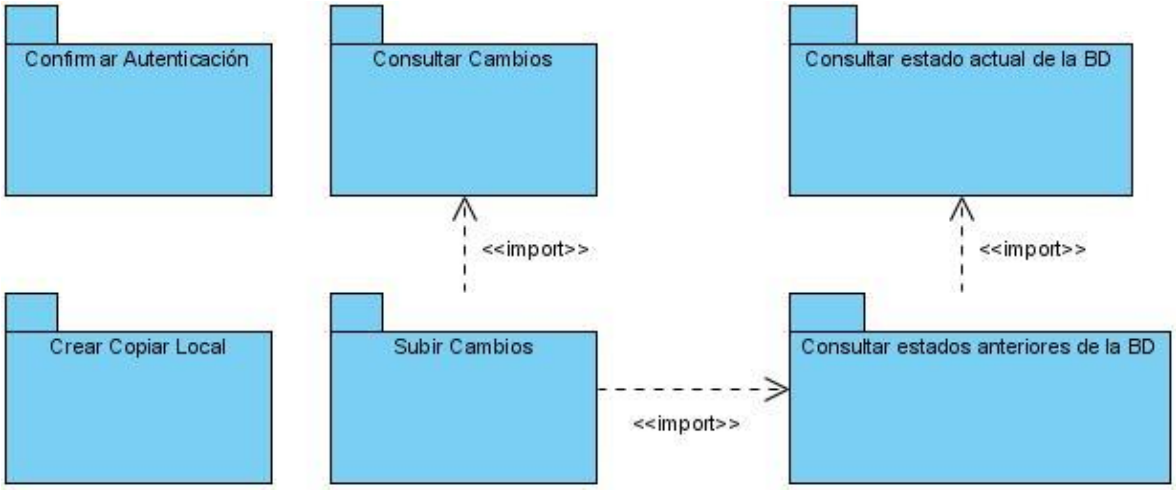


Imagen 6 Diagrama de paquetes del diseño del cliente

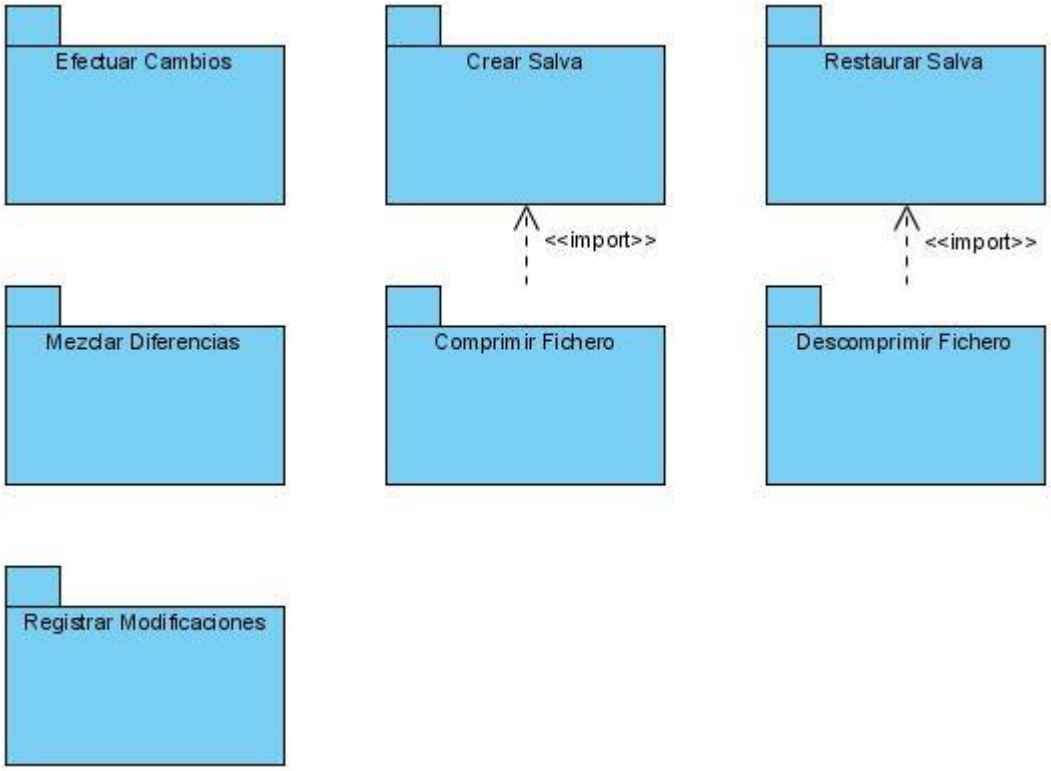


Imagen 7 Diagrama de paquetes del diseño del servidor

Diagrama de clases del diseño

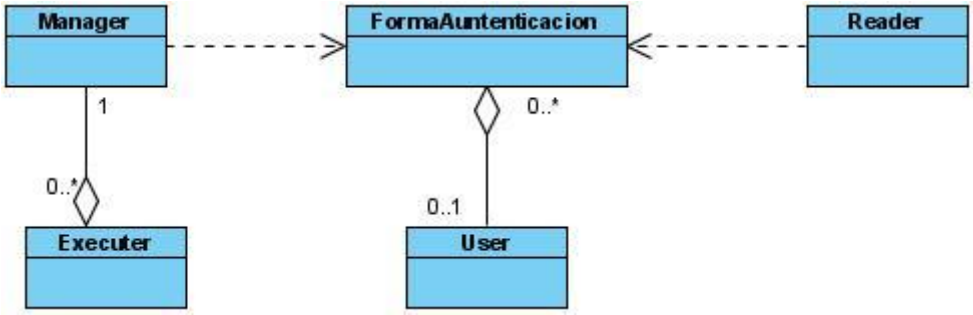


Imagen 8 Diagrama de clases del diseño CU Confirmar Autenticación

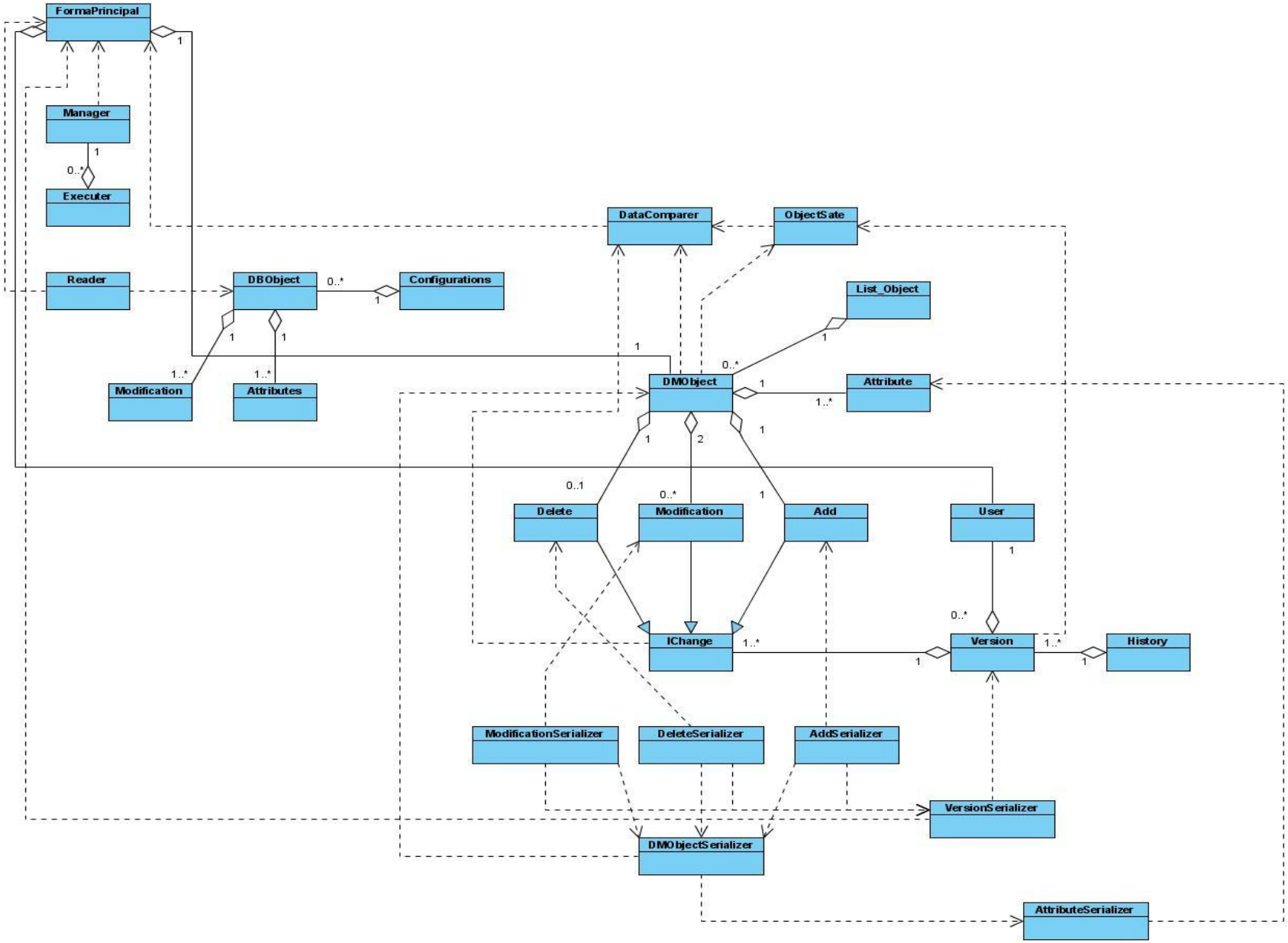


Imagen 9 Diagrama de clases del diseño CU Consultar Cambios

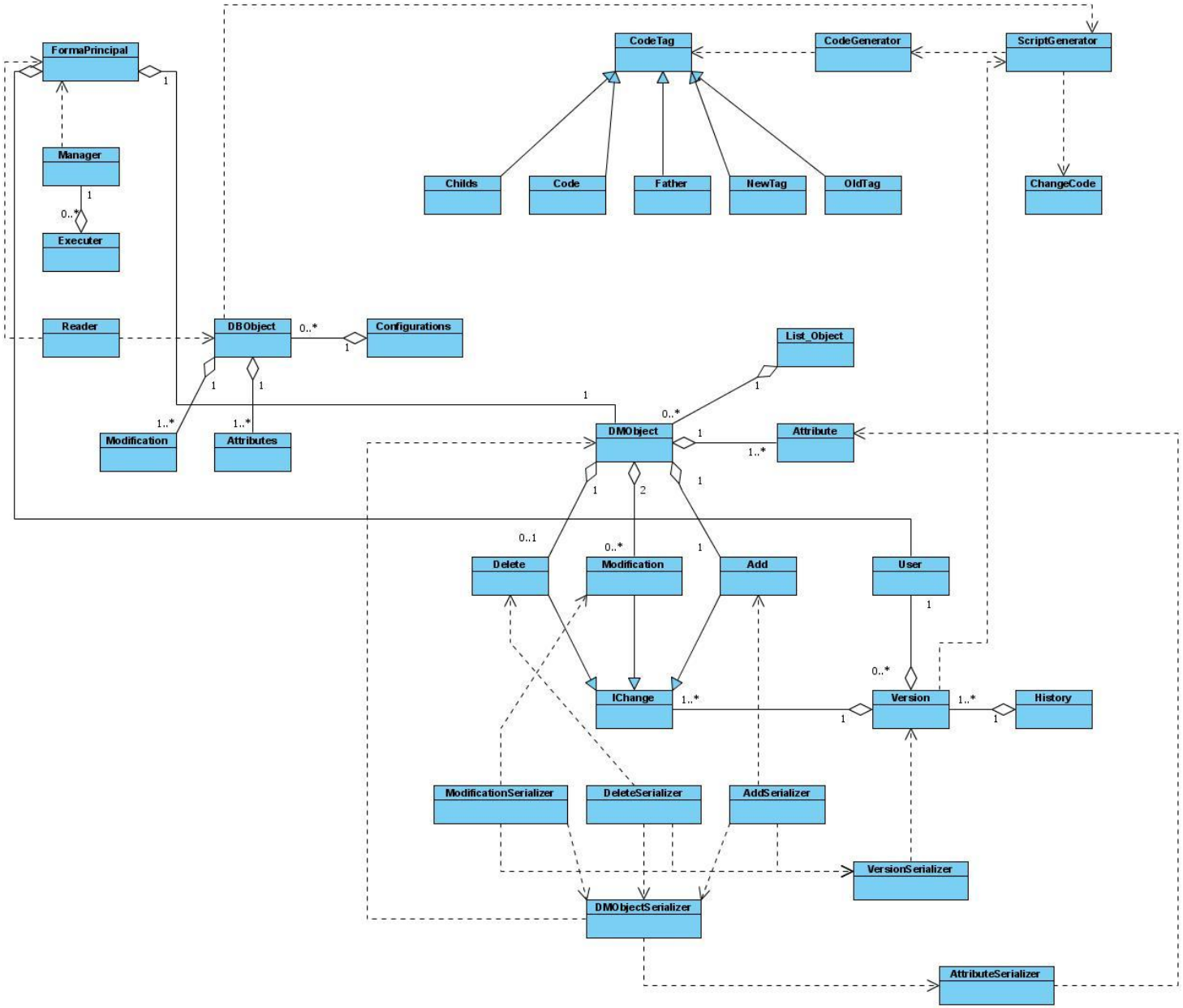


Imagen 10 Diagrama de clases del diseño CU Crear Copia Local

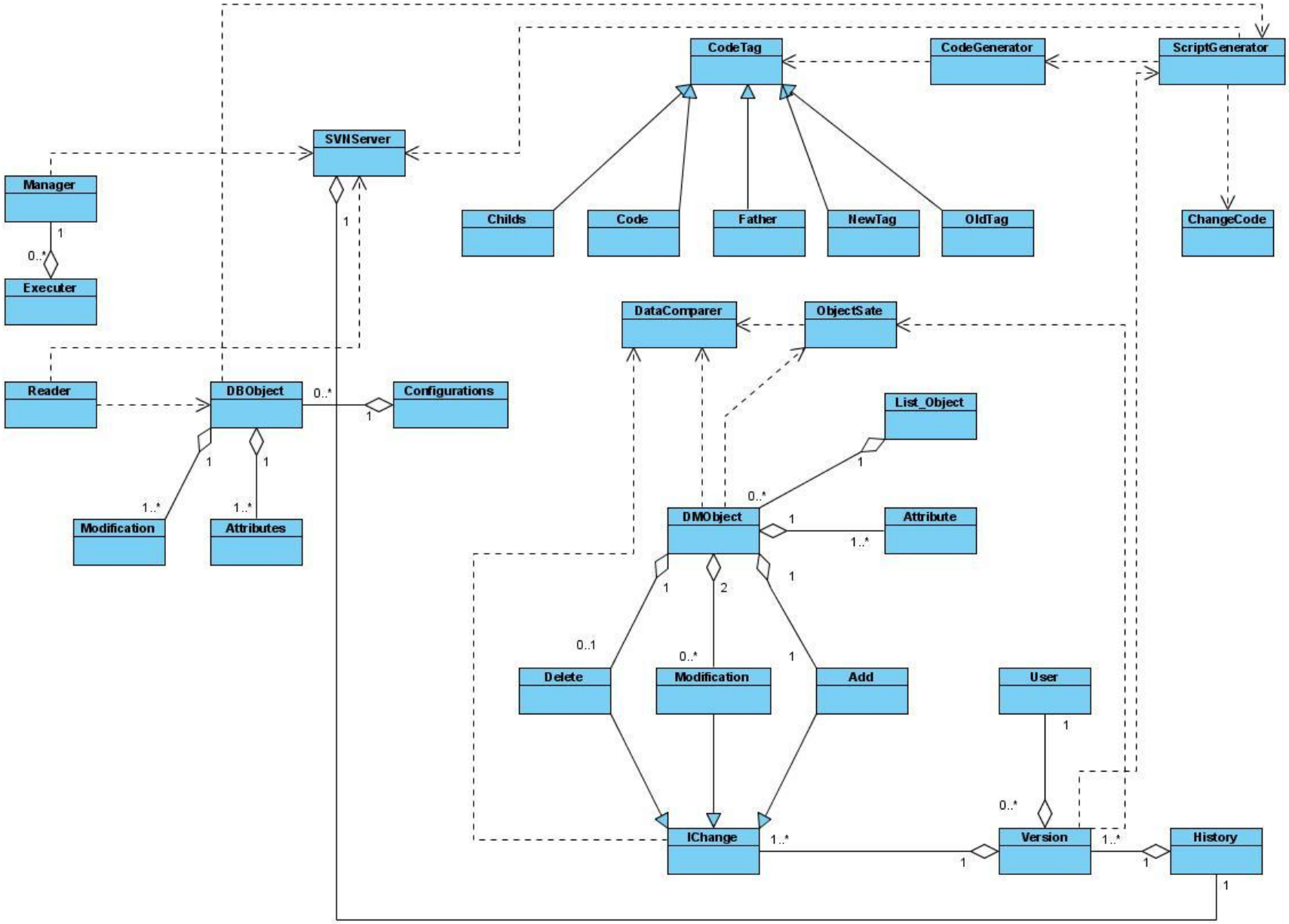


Imagen 11 Diagrama de clases del diseño CU Efectuar Cambios

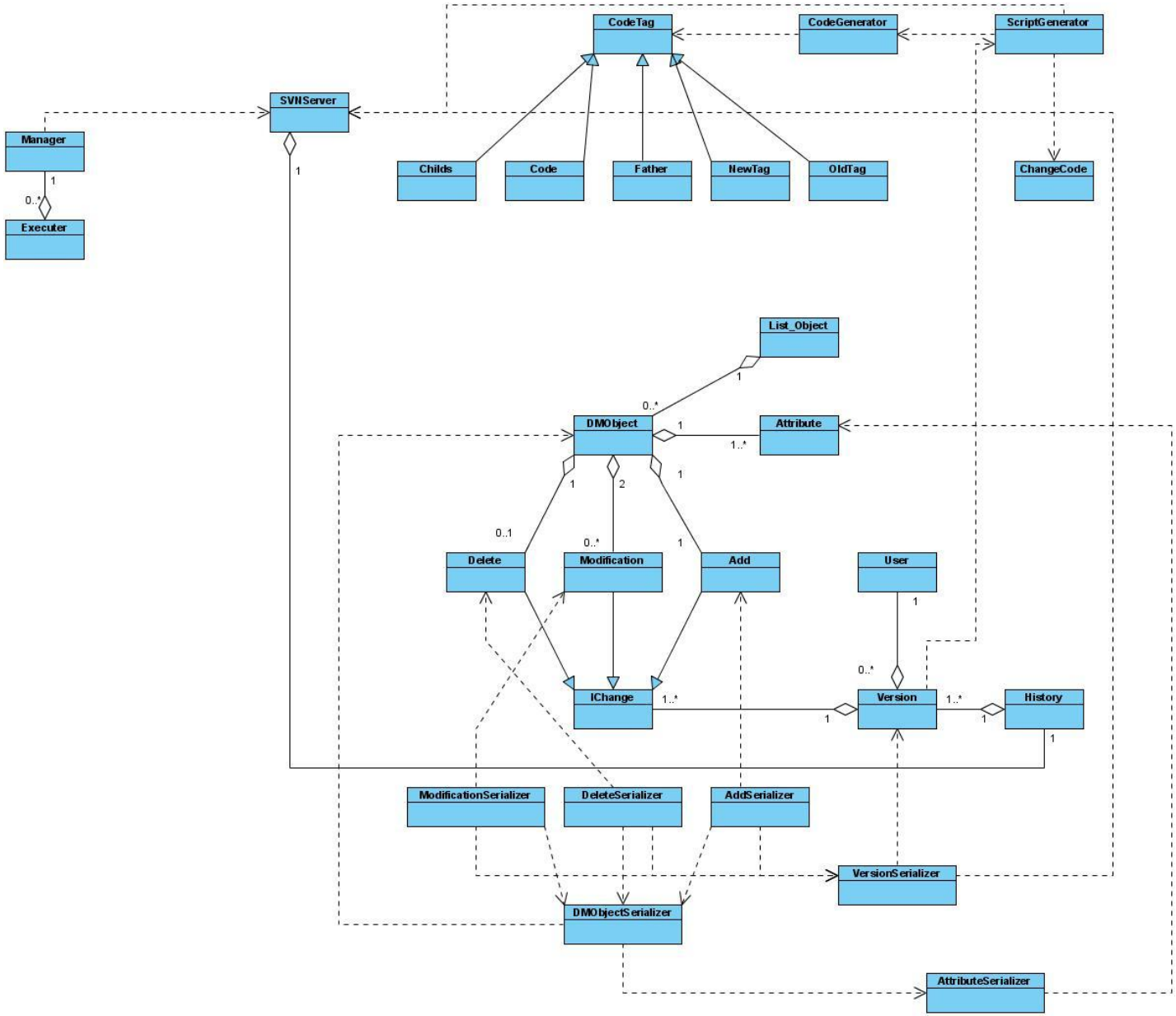


Imagen 12 Diagrama de clases del diseño CU Registrar Modificaciones

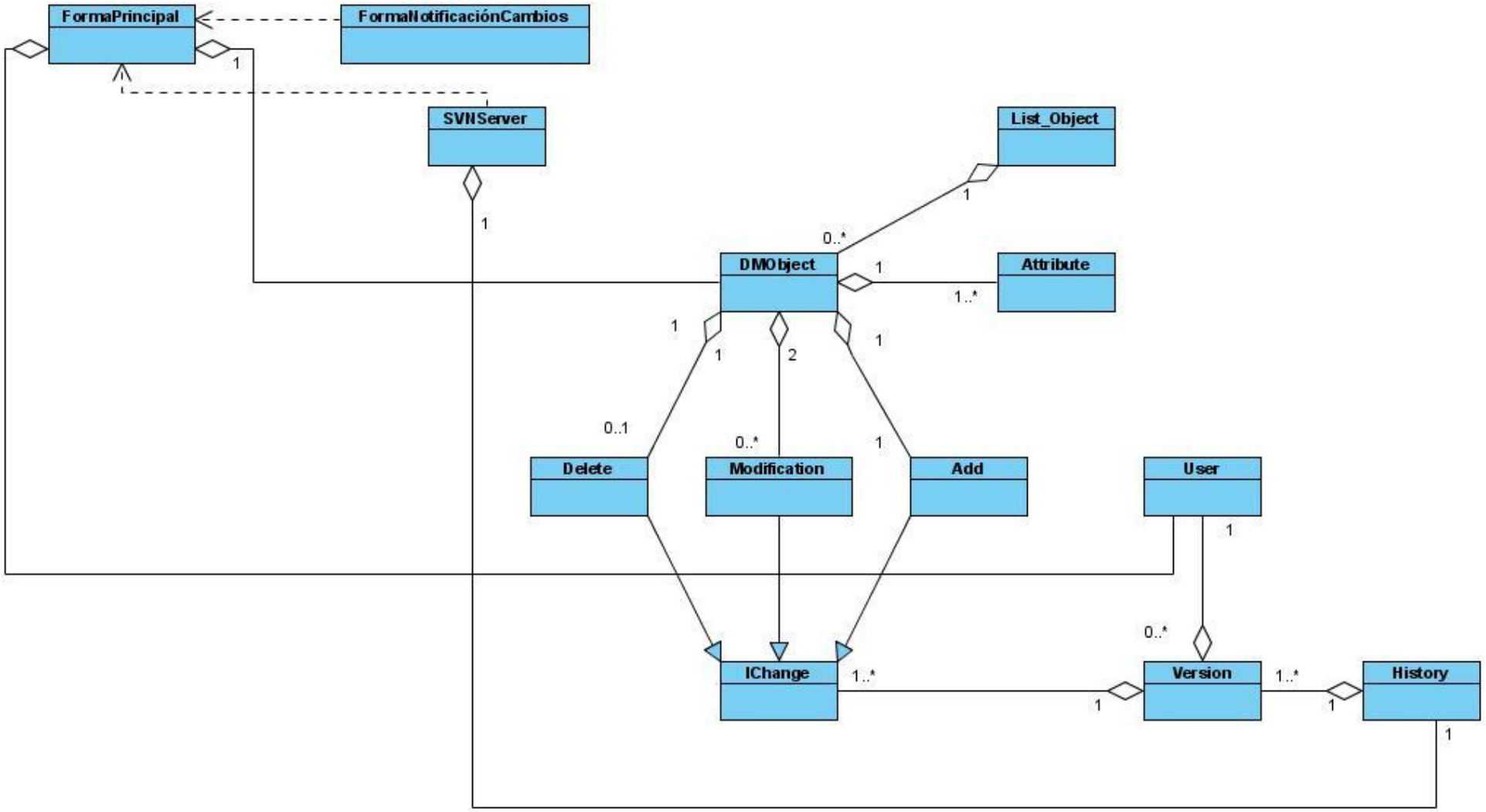


Imagen 13 Diagrama de clases del diseño CU Subir Cambios

Descripción de clases

A continuación se describen algunas de las clases que han sido identificadas en el diseño para su futura implementación con el fin de obtener una mayor compresión del sistema en desarrollo.

Nombre: Executer
Descripción General: La misma tiene el objetivo de interactuar con el gestor de base de datos ya sea del cliente o del servidor. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Confirmar Autenticación • Efectuar Cambios

Tabla 14 Descripción textual clase Executer

Nombre: Manager
Descripción General: La misma tiene el objetivo de gestionar los objetos de conexión con la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Confirmar Autenticación • Efectuar Cambios

Tabla 15 Descripción textual clase Manager

Nombre: BDObject
Descripción General: La misma tiene el objetivo definir el método de obtener la estructura de objeto existente en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 16 Descripción textual clase BDObject

Nombre: Attributes
Descripción General: La misma tiene el objetivo definir el método de obtener los atributos de un objeto existente en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 17 Descripción textual clase Attributes

Nombre: Reader
Descripción General: La misma tiene el objetivo consultar las configuraciones para el gestor de base de datos seleccionado. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Confirmar Autenticación • Efectuar Cambios

Tabla 18 Descripción textual clase Reader

Nombre: Modification
Descripción General: La misma tiene el objetivo definir el método de realizar las modificaciones de un objeto existente en la base de datos con relación a los atributos que fueron modificados. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 19 Descripción textual clase Modification

Nombre: Configuration
Descripción General: La misma tiene el objetivo mantener los objeto existentes en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 20 Descripción textual clase Configuration

Nombre: CodeTag
Descripción General: La misma tiene el objetivo definir un patrón a implementar por las clases asociadas a la generación de código. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 21 Descripción textual clase CodeTag

Nombre: Code
Descripción General: La misma tiene el objetivo definir un patrón para la generación de código, específicamente del código que es estático. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 22 Descripción textual clase Code

Nombre: Father
Descripción General: La misma tiene el objetivo definir un patrón para la generación de código, específicamente para obtener algún dato del objeto padre. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 23 Descripción textual clase Father

Nombre: Childs
Descripción General: La misma tiene el objetivo definir un patrón para la generación de código, específicamente para obtener algún dato de los objetos hijos. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 24 Descripción textual clase Childs

Nombre: OldTag
Descripción General: La misma tiene el objetivo definir un patrón para la generación de código, específicamente para obtener algún dato del objeto antes de que fuera modificado. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 25 Descripción textual clase OldTag

Nombre: NewTag
Descripción General: La misma tiene el objetivo definir un patrón para la generación de código, específicamente para obtener algún dato del objeto después que fuera modificado. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 26 Descripción textual clase NewTag

Nombre: CodeGenerator
Descripción General: La misma tiene el objetivo generar el código asociado a una modificación realizada en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 27 Descripción textual clase CodeGenerator

Nombre: IChange
Descripción General: La misma tiene el objetivo definir un patrón para las clases que representan modificaciones en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Mezclar Diferencias

Tabla 28 Descripción textual clase IChange

Nombre: Add
Descripción General: La misma tiene el objetivo definir una modificación asociada a la creación de un objeto en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Mezclar Diferencias

Tabla 29 Descripción textual clase Add

Nombre: Delete
Descripción General: La misma tiene el objetivo definir una modificación asociada a la eliminación de un objeto en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Mezclar Diferencias

Tabla 30 Descripción textual clase Delete

Nombre: Modification
Descripción General: La misma tiene el objetivo definir una modificación asociada a la modificación de un objeto en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Mezclar Diferencias

Tabla 31 Descripción textual clase Modification

Nombre: User
Descripción General: La misma tiene el objetivo gestionar los datos asociados a los usuarios del sistema. Es utilizada en los casos de uso: • Subir Cambios • Registrar Modificaciones • Crear Copia Local • Confirmar Autenticación

Tabla 32 Descripción textual clase User

Nombre: DMObject
Descripción General: La misma tiene el objetivo gestionar los datos asociados a los objetos existentes en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Mezclar Diferencias • Crear Copia Local

Tabla 33 Descripción textual clase DMObject

Nombre: Attribute
Descripción General: La misma tiene el objetivo gestionar los datos asociados a los atributos de los objetos existentes en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Mezclar Diferencias • Crear Copia Local

Tabla 34 Descripción textual clase Attribute

Nombre: List_Objects
Descripción General: La misma tiene el objetivo manejar los objetos existentes en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Mezclar Diferencias • Crear Copia Local

Tabla 35 Descripción textual clase List_Objects

Nombre: DataComparer
Descripción General: La misma tiene el objetivo buscar modificaciones los datos asociados a los objetos existentes en la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones

Tabla 36 Descripción textual clase DataComparer

Nombre: ObjectState
Descripción General: La misma tiene el objetivo buscar modificaciones los datos asociados a un objeto de la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Subir Cambios • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones

Tabla 37 Descripción textual clase ObjectState

Nombre: ScriptGenerator
Descripción General: La misma tiene el objetivo generar el código asociado a los cambios existentes en una versión de la base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 38 Descripción textual clase ScriptGenerator

Nombre: ChangeCode
Descripción General: La misma tiene el objetivo gestionar el código asociado a un cambio realizado en base de datos. Es utilizada en los casos de uso: • Consultar Cambios. • Crear Copia Local • Efectuar Cambios

Tabla 39 Descripción textual clase ChangeCode

Nombre: VersionSerializer
Descripción General: La misma tiene el objetivo serializar a xml los datos asociados a una versión de la base de datos y obtener estos datos a partir del mismo xml. Es utilizada en los casos de uso: • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Crear Copia Local

Tabla 40 Descripción textual clase VersionSerializer

Nombre: AddSerializer
Descripción General: La misma tiene el objetivo serializar a xml los datos asociados a un cambio de adición en la base de datos y obtener estos datos a partir del mismo xml. Es utilizada en los casos de uso: • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Crear Copia Local

Tabla 41 Descripción textual clase AddSerializer

Nombre: DeleteSerializer
Descripción General: La misma tiene el objetivo serializar a xml los datos asociados a un cambio de eliminación en la base de datos y obtener estos datos a partir del mismo xml. Es utilizada en los casos de uso: • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Crear Copia Local

Tabla 42 Descripción textual clase DeleteSerializer

Nombre: ModificationSerializer
Descripción General: La misma tiene el objetivo serializar a xml los datos asociados a un cambio de modificación en la base de datos y obtener estos datos a partir del mismo xml. Es utilizada en los casos de uso: • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Crear Copia Local

Tabla 43 Descripción textual clase ModificationSerializer

Nombre: DMObjectSerializer
Descripción General: La misma tiene el objetivo serializar a xml los datos asociados a un objeto de la base de datos y obtener estos datos a partir del mismo xml. Es utilizada en los casos de uso: • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Crear Copia Local

Tabla 44 Descripción textual clase DMObjectSerializer

Nombre: AttributeSerializer
Descripción General: La misma tiene el objetivo serializar a xml los datos asociados a un atributo de un objeto de la base de datos y obtener estos datos a partir del mismo xml. Es utilizada en los casos de uso: • Consultar estado actual de la BD • Consultar estados anteriores de la BD • Registrar Modificaciones • Crear Copia Local

Tabla 45 Descripción textual clase AttributeSerializer

Conclusiones

En el presente capítulo se definieron las clases de diseño del software, así como sus atributos y responsabilidades. Se realizó el modelo de diseño, explicándose la estructura y la definición de elementos que este posee. También se definieron los diagramas de clases del diseño de los casos de uso arquitectónicamente significativos y la descripción de las clases presentes en estos diagramas.

CAPÍTULO 4: IMPLEMENTACIÓN

En el presente capítulo se presenta el modelo de implementación obtenido para el sistema ClioBD, compuesto por los modelos de despliegue y componentes. Además se brinda una descripción detallada de los archivos de configuración presentes en el sistema, a fin de posibilitar el ajuste del mismo a los diferentes gestores de bases de datos.

Diagrama de Despliegue

El diagrama de despliegue permite mostrar la arquitectura en tiempo de ejecución del sistema respecto a hardware y software (Benet, 2003). Este diagrama es un grafo de nodos unidos por conexiones de comunicación, donde cada nodo será una unidad de computación de algún tipo que pueden contener instancias de componentes de software.

La aplicación tiene cuatro elementos fundamentales: Los servidores de bases de datos tanto cliente como servidor, los cuales mantienen la estructura de bases de datos versionada por el sistema; el cliente, donde se instalará la aplicación de escritorio para el manejo de las versiones y el servidor central del sistema, donde se montara el servicio al que notificaran los cambios cada uno de los clientes y mantendrá actualizada la base de datos central.

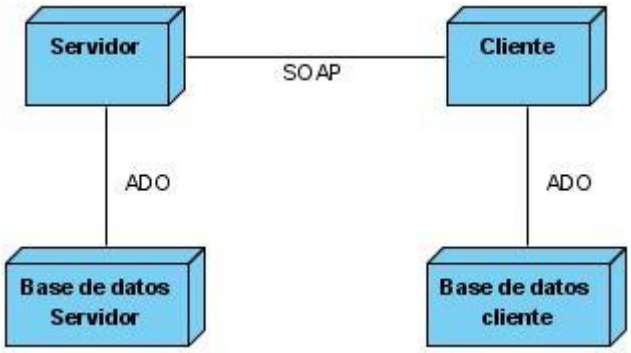


Imagen 14 Diagrama de componentes

Diagrama de Componentes

Un componente es una parte física del sistema, como puede ser un fichero ejecutable, una librería de clases o un archivo de configuración. Estos se agrupan en diagramas de componentes para mostrar cómo queda dividido el sistema y las dependencias existentes entre estos.

A pesar que en muchos casos los diagramas de componentes son divididos por capas o módulos en dependencia de la arquitectura definida para el sistema, en este caso no se realizará así, ya que las estructuras básicas de la arquitectura (Cliente y Servidor) incluyen la mayoría de los componentes por lo que en una única agrupación se obtiene una mejor visión de la estructura final de la aplicación.

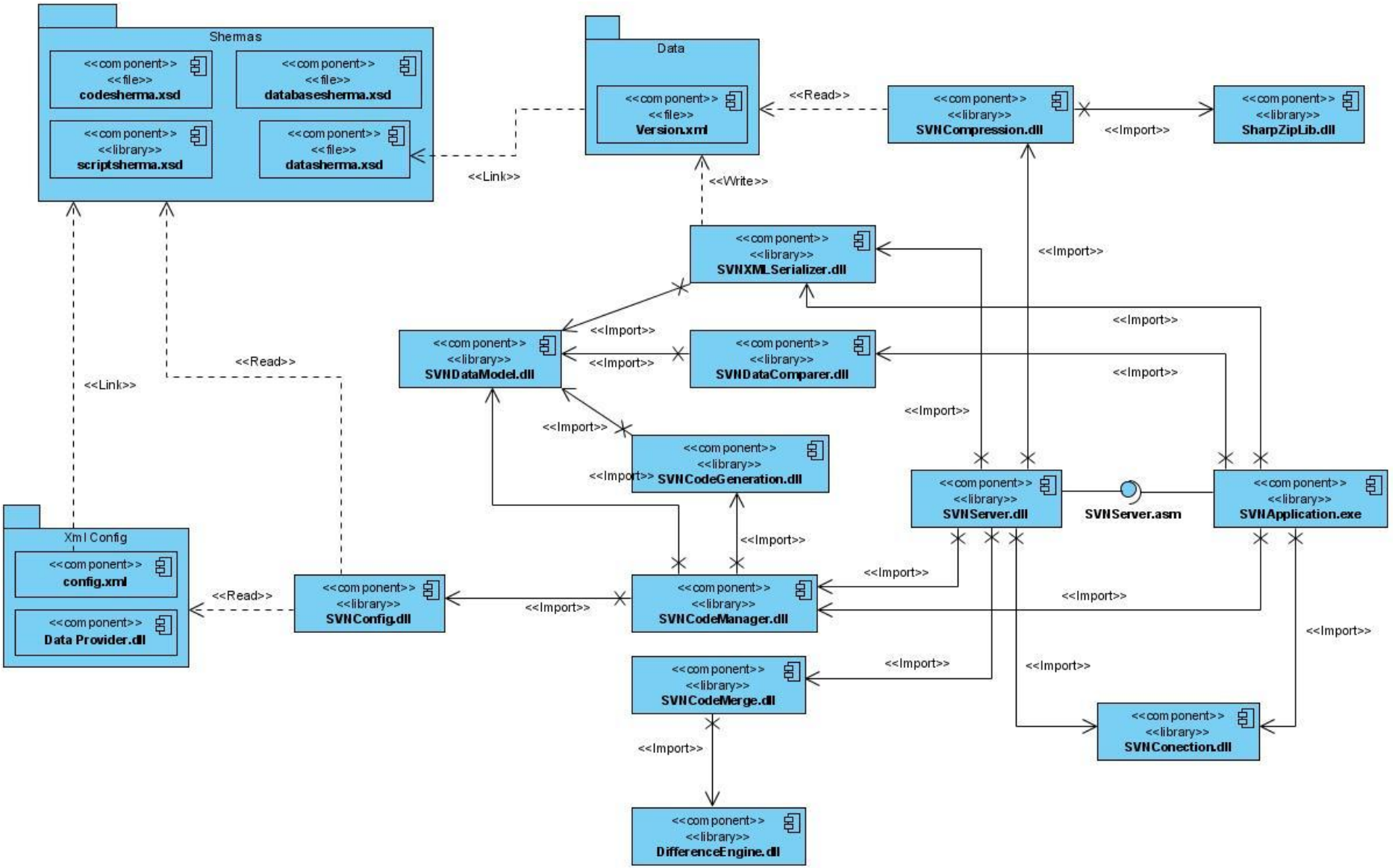


Imagen 15 Diagrama de componentes

Archivos de configuración del sistema

A pesar que en el objetivo de este trabajo de diploma sea el desarrollo de un sistema de control de versiones para bases de datos sobre el gestor PostgreSQL; ClioBD está elaborado de forma que pueda ser configurado para cualquier gestor de base de datos que implementen el modelo relacional. Esta configuración es almacenada en una serie de archivos xml, que se definen por cuatro esquemas diferentes en dependencia de su uso.

Definición del gestor:

El fichero dbconnection.xml define la librería y la estructura de la cadena de conexión. Su estructura básica es la siguiente:

<dbdefinition dll="" conection="" command="" dataadapter="" stringconection=""/>

Este fichero solo presenta el tag dbdefinition el cual contiene cuatro atributos:

- *dll*. En este atributo se define la ubicación física de la librería de clases que se utilizara para realizar las conexiones a la base de datos, esta librería debe implementar las interfaces de ADO.Net.
- *conection*. En este atributo se define el nombre de la clase que implementa la interface IDbConnection dentro de la librería seleccionada y servirá para obtener una conexión con el servidor de base de datos.
- *command*. En este atributo se define el nombre de la clase que implementa la interface IDbCommand dentro de la librería seleccionada y permitirá realizar las consultas necesarias a la base de datos.
- *dataadapter*. En este atributo se define el nombre de la clase que implementa la interface IDbDataAdapter dentro de la librería seleccionada y permitirá obtener las respuestas dadas por el gestor de bases de datos a los comandos que se le envían a este.

Definición de la estructura de datos:

El fichero dbdata.xml define la forma lógica de almacenamiento de las estructuras a versionar para el gestor de base de datos configurados. Su estructura básica es la siguiente:

```
<datadefinition>
   <object type="" scriptdefiner="" idposition="" istable=""
           insertpriority="" deletepriority="">
           <attribute name="" position=""/>
           <childs>
           </childs>
   </object>
</datadefinition>
```

Este fichero presenta inicialmente el tag *datadefinition* dentro del cual se incluye una estructura de formada por varios tags del tipo *object*. Dentro del tag *object* podrán ser ubicados tantos tags del tipo *attribute* como sean necesarios y un solo tag childs. El tag childs contendrá una estructura de formada por varios tags del tipo *object* de manera que se podrá representar los objetos que incluidos dentro de otros.

El tag *object* representa un tipo de objeto existente dentro de la base de datos como puede ser una tabla, una función, un trigger, etc. Este tag contiene seis atributos: *type,* este atributo define el tipo de objeto al que se hace referencia (Ej: Tabla, Atributo, Función); *scriptdefiner,* este atributo define la dirección física del fichero donde se definen la configuración de comandos; *idposition,* este atributo define la posición dentro de la consulta de selección, definida para este tipo objeto, en la que se obtendrá el campo que se utilizara como identificador para el mismo; *insertpriority*, este atributo define la prioridad de inserción de este tipo de objeto con respecto al resto; *deletepriority*, este atributo define la prioridad de eliminación de este tipo de objeto con respecto al resto; *istable,* este atributo permite especificar si el tipo de objeto que se desea representar es una tabla o no.

El tag *attribute* representa un atributo existente dentro de un tipo de objeto ya definido. Este tag contiene dos atributos: *name, permite* definir el nombre tendrá el atributo; *position,* este atributo define la posición de este atributo dentro de la consulta de selección, definida para el tipo objeto que lo contiene.

Definición de los comandos:

Este grupo de ficheros permite definir donde están ubicadas las plantillas para la generación del código necesario para la manipulación de un tipo de objeto determinado. Su estructura básica es la siguiente:

```
<scriptdefinition>
   <select script=""/>
   <insert script=""/>
   <delete script=""/>
   <update script="" attribute=""/>
</scriptdefinition>
```

Estos ficheros presentan inicialmente el tag *scriptdefinition* dentro del cual se incluye una estructura de formada por un tag del tipo *select,* uno del tipo *insert*, uno del tipo *delete* y varios del tipo *update*. En cada uno de estos tags el atributo scritp define la ubicación física de la plantilla de generación de código relativa a la operación definida por el nombre del tag. En el caso del tag *update* se incluye el atributo *attribute* en el cual se definirá el campo a modificar, este atributo puede tomar el valor de default y le será asignado a todos los campos que no tengan un tag de *update* definido.

Plantillas de código:

Este grupo de ficheros permite las plantillas para la generación del código necesario para el funcionamiento del sistema. Su estructura básica es la siguiente:

```
<generate>
        <code>
        </code>
        <father attribute="" generation=""/>
        <childs count="" type=" " script="" new=""/>
        <new attribute=""/>
        <old attribute="">
</generate>
```

Estos ficheros presentan inicialmente el tag *generate* dentro del cual se incluye una estructura de formada por varios tags de los tipos *code, father, childs, new y old,* ordenados de forma aleatoria.

Dentro del tag *code* se pondrá la porción del código que no está sujeta a cambios, producto de los datos del objeto. El tag *father* definirá datos a obtener de los objetos que son padres al tipo de objeto para el que se define este código, este incluye dos atributos: *attribute*, donde se define el valor que se

quiere obtener del objeto padre; *generation,* donde se define cuanto hay que subir en la estructura arbolea para obtener el valor. El tag *child* representa porciones de código a incluir por cada objeto hijo presente en el objeto al que se le genera al código, este incluye cuatro atributos: *count,* define la cantidad de hijos que van a ser incluidos, si el valor de este atributo es -1 de asumirá que se desean todos los hijos existentes; *type,* define que tipo de hijos serán analizados; *script,* define la ubicación física de la plantilla de generación de código; *new,* define si los hijos serán obtenidos del objeto antes o después de las modificaciones. Los tags *new* y *old* se refieren a un valor campo del objeto, definido por el atributo *attribute* con la diferencia de que el tag *new* se refiere al objeto antes de las modificaciones y el *old* se refiere al objeto después de las modificaciones.

Conclusiones

En el presente capítulo se definió el modelo de implementación obtenido para el sistema ClioBD, expresado en sus modelos de despliegue y componentes. Además se mostró la estructura detallada de los archivos de configuración presentes en el sistema, dando así a los futuros usuarios la posibilidad de ajustar el mismo a los diferentes gestores de bases de datos.

CONCLUSIONES

Con la realización de este trabajo se concluye de forma general lo siguiente:

- Los procesos actuales de desarrollo de software tienden a la generación de equipos de trabajo cada vez más grandes, lo que refuerza la necesidad de una gestión de la configuración eficiente.
- La base de datos, como elemento de configuración, requiere que sus versiones sean controladas en el transcurso del proceso de desarrollo.
- Los sistemas existentes para el control de versiones de bases de datos no cumplen con las características requeridas para la producción de software en la UCI.
- Se propuso la creación de un sistema de control de versiones para bases de datos, obteniéndose una versión estable del mismo.

RECOMENDACIONES

En el transcurso de este proyecto, han surgido ideas que podrían incorporarse en un futuro a la aplicación con el fin de que esta sea más abarcadora, para lo cual se recomienda:

- Elaborar las plantillas de configuración para los gestores de bases de datos MySQL y Oracle.
- Desarrollar una herramienta que ayude en el manejo de las plantillas de configuración.
- Realizar pruebas de calidad al sistema.

REFERENCIAS BIBLIOGRÁFICAS

Jacobson, I. y Booch, G. y Rumbaugh, J. *El Proceso Unificado de Desarrollo de software.* s.l. : Addison-Wesley, 2000.

Acevedo, Rodolfo Villarroel. *Mejoramiento del Proceso de Gestión.* La Mancha : Universidad de Castilla-La Mancha, 2004.

González, Guillermo. *Sistema de Control de Versiones. Caso de estudio: Subversion.* Asunción : Universidad Nacional de Asunción, 2008.

Tigris.org. subversion: Subversion Features. *COLLABNET Enterprise Edition.* [En línea] Tigris.org. [Citado el: 21 de Enero de 2009.] http://subversion.tigris.org/features.html.

MKS. MKS Integrity for Software Configuration Management (SCM). [En línea] [Citado el: 21 de Enero de 2009.] http://www.mks.com/products/source.

Microsoft. MSDN. [En línea] [Citado el: 2009 de Enero de 21.] http://msdn.microsoft.com/es-es/library/3h0544kx(VS.80).aspx.

GNU. CVS - Concurrent Versions System. [En línea] [Citado el: 2009 de Enero de 23.] http://www.nongnu.org/cvs/.

Canonical Ltd. BzrFeatures - Bazaar Version Control. *Bazaar Version Control.* [En línea] Canonical Ltd. [Citado el: 21 de Enero de 2009.] http://bazaar-vcs.org/BzrFeatures.

Chacon, Scott. Git - Fast Version Control System. *Git - Fast Version Control System.* [En línea] github. [Citado el: 21 de Enero de 2009.] http://git-scm.com/about.

Martínez, Angeles Maldonado y Yunta, Luis Rodríguez. *La información especializada en Internet.* Madrid : CSIC, 2006. 84-00-08436-5.

CommentCaMarche. Kioskea.net. [En línea] ShareAlike 2.0 France, 16 de Octubre de 2008. [Citado el: 15 de Febrero de 2009.] http://es.kioskea.net/contents/bdd/bddtypes.php3.

Sicilia, Miguel Angel. Connexions. [En línea] 23 de Septiembre de 2008. [Citado el: 10 de Febrero de 2009.] http://cnx.org/content/m17543/latest/.

Microsoft. Microsoft SQL Server: ¿Qué es SQL Server 2005? [En línea] Microsoft, 25 de Mayo de 2006. [Citado el: 12 de Febrero de 2009.] http://www.microsoft.com/spain/sql/productinfo/overview/what-is-sql-server.mspx.

The PostgreSQL Global Development Group. *PostgreSQL 8.1.0 Documentation.* California : University of California, 2005.

Oracle. Oracle Database 11g Standard Edition. [En línea] Oracle. [Citado el: 10 de Febrero de 2009.] http://www.oracle.com/database/standard_edition.html.

Atwood, Jeff. Coding Horror: Is Your Database Under Version Control? [En línea] 12 de Diciembre de 2006. [Citado el: 21 de Enero de 2009.] http://www.codinghorror.com/blog/archives/000743.html.

Dinamic3 LT Consulting GmbH. SASSI. [En línea] [Citado el: 10 de Diciembre de 2008.] http://www.sqlassi.net/Features.htm.

BBS Tecnologies, Inc. SQL change manager. [En línea] [Citado el: 10 de Diciembre de 2008.] http://www.idera.com/products/sqlchange/.

Embarcadero Tecnologies. Embarcadero Change Manager - Database Change Management. [En línea] Octubre de 2008. [Citado el: 10 de Diciembre de 2008.] http://www.embarcadero.com/products/changemanager/index.html.

Grau, Xavier Ferré. Tutorial UML, Desarrollo Orientado a Objetos con UML. [En línea] 2004. [Citado el: 23 de Enero de 2009.] http://www.clikear.com/manuales/uml/introduccion.aspx.

Sanchez, María A. Mendoza. *¿Qué metodología debo usar para el desarrollo de un Software?* Lima : Informatizate, 2004.

Seco, José Antonio González. Desarrollo Web. [En línea] 08 de Noviembre de 2001. [Citado el: 23 de Enero de 2009.] http://www.desarrolloweb.com/articulos/592.php.

Sun Microsystem. Conozca más sobre la tecnología Java. [En línea] [Citado el: 27 de Enero de 2009.] http://java.com/es/about/.

Novell. Mono-Project. [En línea] [Citado el: 2009 de Enero de 23.] http://mono-project.com/What_is_Mono.

Mayoral, Antonio Gutiérrez. Caracteristicas principales de C#. [En línea] 11 de Febrero de 2005. [Citado el: 23 de Enero de 2009.] http://gsyc.es/~agutierr/pfc-tecnica-html/node22.html.

Paredes, Arbis Percy Reyes. Historia de Visual Basic. NET. [En línea] 3 de Septiembre de 2005. [Citado el: 28 de Febrero de 2009.] http://www.elguille.info/colabora/NET2005/Percynet_Historia_Visual_Basic_NET.htm.

WebAdictos. SharpDevelop, IDE para .NET open source. [En línea] [Citado el: 24 de Enero de 2009.] http://www.webadictos.com.mx/2008/10/03/sharpdevelop-ide-para-net-open-source/.

Mircrosoft. Visual Studio. [En línea] Noviembre de 2007. [Citado el: 27 de Enero de 2009.] http://msdn.microsoft.com/es-es/library/52f3sw5c.aspx.

Microsoft. Microsoft Visual Studio 2008 Standard Edition. [En línea] 2008. [Citado el: 2009 de Enero de 27.] http://www.microsoft.com/visualstudio/en-us/products/standard/default.mspx.

W3C. Guía Breve de Tecnologías XML. [En línea] 09 de Enero de 2008. [Citado el: 23 de Enero de 2009.] http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML.

Ministerio de administraciones publicas. *Metodologia MÉTRICA Versión 3.* Madrid : s.n., 2008.

IEEE. Glosario de Terminología Informática. [En línea] IEEE, 9 de Abril de 1997. [Citado el: 20 de Febrero de 2009.] http://www.tugurium.com/gti/termino.asp?tr=requirement.

HispaNetwork Publicidad y Servicios, S.L. Glosario.Net. [En línea] HispaNetwork Publicidad y Servicios, S.L., 27 de Octubre de 2006. [Citado el: 18 de Febrero de 2009.] http://tecnologia.glosario.net/terminos-tecnicos-internet/casos-de-uso-267.html.

Benet Campderrich Falgueras, Ramon Maspons. *Ingeniería del software.* Cataluña : UOC, 2003. 9788484297932.

BIBLIOGRAFÍA

Sanchez, María A. Mendoza. ¿Qué metodología debo usar para el desarrollo de un Software?: Informatizate, Lima: 2004.

Atwood, Jeff. 2006. Coding Horror: Is Your Database Under Version Control? [En línea] 12 de Diciembre de 2006. [Citado el: 21 de Enero de 2009.] http://www.codinghorror.com/blog/archives/000743.html.

BBS Tecnologies, Inc. SQL change manager. [En línea] [Citado el: 10 de Diciembre de 2008.] http://www.idera.com/products/sqlchange/.

Benet Campderrich Falgueras, Ramon Maspons. 2003. Ingeniería del software. Cataluña : UOC, 2003. 9788484297932.

Canonical Ltd. BzrFeatures - Bazaar Version Control. Bazaar Version Control. [En línea] Canonical Ltd. [Citado el: 21 de Enero de 2009.] http://bazaar-vcs.org/BzrFeatures.

Chacon, Scott. Git - Fast Version Control System. Git - Fast Version Control System. [En línea] github. [Citado el: 21 de Enero de 2009.] http://git-scm.com/about.

CommentCaMarche. 2008. Kioskea.net. [En línea] ShareAlike 2.0 France, 16 de Octubre de 2008. [Citado el: 15 de Febrero de 2009.] http://es.kioskea.net/contents/bdd/bddtypes.php3.

Dinamic3 LT Consulting GmbH. SASSI. [En línea] [Citado el: 10 de Diciembre de 2008.] http://www.sqlassi.net/Features.htm.

Embarcadero Tecnologies. 2008. Embarcadero Change Manager - Database Change Management. [En línea] Octubre de 2008. [Citado el: 10 de Diciembre de 2008.] http://www.embarcadero.com/products/changemanager/index.html.

Ferguson, Jeff, Patterson, Brian y Beres, Jason. 2003. La Biblia C#. Madrid : ANAYA MULTIMEDIA, 2003.

GNU. CVS - Concurrent Versions System. [En línea] [Citado el: 2009 de Enero de 23.] http://www.nongnu.org/cvs/.

Grau, Xavier Ferré. 2004. Tutorial UML, Desarrollo Orientado a Objetos con UML. [En línea] 2004. [Citado el: 23 de Enero de 2009.] http://www.clikear.com/manuales/uml/introduccion.aspx.

HispaNetwork Publicidad y Servicios, S.L. 2006. Glosario.Net. [En línea] HispaNetwork Publicidad y Servicios, S.L., 27 de Octubre de 2006. [Citado el: 18 de Febrero de 2009.] http://tecnologia.glosario.net/terminos-tecnicos-internet/casos-de-uso-267.html.

IEEE. 1997. Glosario de Terminología Informática. [En línea] IEEE, 9 de Abril de 1997. [Citado el: 20 de Febrero de 2009.] http://www.tugurium.com/gti/termino.asp?tr=requirement.

Jacobson, I. y Booch, G. y Rumbaugh, J. 2000. El Proceso Unificado de Desarrollo de software. s.l. : Addison-Wesley, 2000.

Lopez, Angel. 2008. El camino hacia el Modelo de Dominio y Domain-Driven Design. [En línea] 03 de Septiembre de 2008. [Citado el: 15 de Octubre de 2008.] http://msmvps.com/blogs/lopez/archive/2008/09/03/el-camino-hacia-el-modelo-de-dominio-y-domain-driven-design.aspx.

Martínez, Angeles Maldonado y Yunta, Luis Rodríguez. 2006. La información especializada en Internet. Madrid : CSIC, 2006. 84-00-08436-5.

Mayoral, Antonio Gutiérrez. 2005. Caracteristicas principales de C#. [En línea] 11 de Febrero de 2005. [Citado el: 23 de Enero de 2009.] http://gsyc.es/~agutierr/pfc-tecnica-html/node22.html.

Acevedo, Rodolfo Villarroel. 2004. Mejoramiento del Proceso de Gestión. La Mancha : Universidad de Castilla-La Mancha, 2004.

Microsoft. 2006. Microsoft SQL Server: ¿Qué es SQL Server 2005? [En línea] Microsoft, 25 de Mayo de 2006. [Citado el: 12 de Febrero de 2009.] http://www.microsoft.com/spain/sql/productinfo/overview/what-is-sql-server.mspx.

Microsoft. 2007. Visual Studio. [En línea] Noviembre de 2007. [Citado el: 27 de Enero de 2009.] http://msdn.microsoft.com/es-es/library/52f3sw5c.aspx.

Microsoft. 2008. Microsoft Visual Studio 2008 Standard Edition. [En línea] 2008. [Citado el: 2009 de Enero de 27.] http://www.microsoft.com/visualstudio/en-us/products/standard/default.mspx.

Microsoft. MSDN. [En línea] [Citado el: 2009 de Enero de 21.] http://msdn.microsoft.com/es-es/library/3h0544kx(VS.80).aspx.

Ministerio de administraciones publicas. 2008. Metodologia MÉTRICA Versión 3. Madrid : s.n., 2008.

MKS. MKS Integrity for Software Configuration Management (SCM). [En línea] [Citado el: 21 de Enero de 2009.] http://www.mks.com/products/source.

Novell. Mono-Project. [En línea] [Citado el: 2009 de Enero de 23.] http://mono-project.com/What_is_Mono.

Oracle. Oracle Database 11g Standard Edition. [En línea] Oracle. [Citado el: 10 de Febrero de 2009.] http://www.oracle.com/database/standard_edition.html.

Paredes, Arbis Percy Reyes. 2005. Historia de Visual Basic. NET. [En línea] 3 de Septiembre de 2005. [Citado el: 28 de Febrero de 2009.] http://www.elguille.info/colabora/NET2005/Percynet_Historia_Visual_Basic_NET.htm.

Seco, José Antonio González. 2001. Desarrollo Web. [En línea] 08 de Noviembre de 2001. [Citado el: 23 de Enero de 2009.] http://www.desarrolloweb.com/articulos/592.php.

Sicilia, Miguel Angel. 2008. Connexions. [En línea] 23 de Septiembre de 2008. [Citado el: 10 de Febrero de 2009.] http://cnx.org/content/m17543/latest/.

González, Guillermo. 2008. Sistema de Control de Versiones. Caso de estudio: Subversion. Asunción : Universidad Nacional de Asunción, 2008.

Castillo, Carlos. 2008. s.l.Sistemas gestores de bases de datos. : UPF, 2008. Sistemas de Información II. Tema 2. págs. 1-49.

Sun Microsystem. Conozca más sobre la tecnología Java. [En línea] [Citado el: 27 de Enero de 2009.] http://java.com/es/about/.

The PostgreSQL Global Development Group. 2005. PostgreSQL 8.1.0 Documentation. California : University of California, 2005.

Tigris.org. subversion: Subversion Features. COLLABNET Enterprise Edition. [En línea] Tigris.org. [Citado el: 21 de Enero de 2009.] http://subversion.tigris.org/features.html.

Vignaga, Andrés. 2006. Transformación de un Modelo de Dominio y Diagramas de Comunicación en un Diagrama de Clases de Diseño. Departamento de Ciencias de la Computación - Universidad de Chile.

W3C. 2008. Guía Breve de Tecnologías XML. [En línea] 09 de Enero de 2008. [Citado el: 23 de Enero de 2009.] http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML.

WebAdictos. SharpDevelop, IDE para .NET open source. [En línea] [Citado el: 24 de Enero de 2009.] http://www.webadictos.com.mx/2008/10/03/sharpdevelop-ide-para-net-open-source/.

Yunta, Luis Rodríguez. 2001. BASES DE DATOS DOCUMENTALES: ESTRUCTURA Y PRINCIPIOS DE USO. Madrid : CINDOC, 2001.

ANEXOS

Anexo 1 Esquemas para los archivos de configuración.

Esquema para la definición del gestor:

```
<?xml version="1.0" encoding="Windows-1252"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="dbdefinition">
                <xs:complexType>
                        <xs:attribute name="dll" type="xs:string" use="required" />
                        <xs:attribute name="conection" type="xs:string" use="required" />
                        <xs:attribute name="command" type="xs:string" use="required" />
                        <xs:attribute name="dataadapter" type="xs:string" use="required" />
                        <xs:attribute name="stringconection" type="xs:string" use="required" />
                </xs:complexType>
        </xs:element>
</xs:schema>
```

Esquema para la definición de la estructura de datos:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="datadefinition">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="object" type="objecttype" />
                        </xs:sequence>
                        <xs:attribute name="triggerscript" type="xs:string" use="required" />
                </xs:complexType>
        </xs:element>

        <xs:complexType name="objecttype">
                <xs:sequence>
                        <xs:element name="attribute">
                                <xs:complexType>
                                        <xs:attribute name="name" type="xs:string" use="required" />
                                        <xs:attribute name="position" type="xs:integer"
                                             use="required" />
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="childs">
                                <xs:complexType>
                                        <xs:sequence minOccurs="0">
                                                <xs:element name="object" type="objecttype" />
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
                <xs:attribute name="type" type="xs:string" use="required" />
                <xs:attribute name="scriptdefiner" type="xs:string" use="required" />
                <xs:attribute name="idposition" type="xs:integer" use="required" />
                <xs:attribute name="istable" type="xs:boolean" use="required" />
                <xs:attribute name="insertpriority" type="xs:integer" use="required" />
                <xs:attribute name="deletepriority" type="xs:integer" use="required" />
        </xs:complexType>
</xs:schema>
```

Esquema para la definición de los comandos:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="scriptdefinition">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="select" minOccurs="1" maxOccurs="1">
                                        <xs:complexType>
                                                <xs:attribute name="script" type="xs:string"
                                                        use="required" />
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="insert" minOccurs="1" maxOccurs="1">
                                        <xs:complexType>
                                                <xs:attribute name="script" type="xs:string"
                                                        use="required" />
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="delete" minOccurs="1" maxOccurs="1">
                                        <xs:complexType>
                                                <xs:attribute name="script" type="xs:string"
                                                        use="required" />
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="update" minOccurs="1"
                                        maxOccurs="unbounded">
                                        <xs:complexType>
                                                <xs:attribute name="script" type="xs:string"
                                                        use="required" />
                                                <xs:attribute name="attribute" type="xs:string"
                                                        use="required" />
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

Esquema para la plantilla de código:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="generate">
                <xs:complexType>
                        <xs:sequence maxOccurs="unbounded">
                                <xs:element name="code" type="xs:string"
                                        minOccurs="0" maxOccurs="unbounded"/>
                                <xs:element name="father" minOccurs="0"
                                        maxOccurs="unbounded" >
                                        <xs:complexType>
                                                <xs:attribute name="attribute” type="xs:string"
                                                        use="required" />
                                                <xs:attribute name="generation" type="xs:integer"
                                                        use="required" />
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="childs" minOccurs="0"  maxOccurs="unbounded" >
                                        <xs:complexType>
                                                <xs:attribute name="count" type="xs:integer"
                                                        use="required" />
                                                <xs:attribute name="type" type="xs:string"
                                                        use="required" />
                                                <xs:attribute name="script" type="xs:string"
                                                        use="required" />
                                                <xs:attribute name="new" type="xs:boolean"
                                                        use="required" />
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="new" minOccurs="0" maxOccurs="unbounded" >
                                        <xs:complexType>
                                                <xs:attribute name="attribute" type="xs:string"
                                                        use="required" />
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="old" minOccurs="0" maxOccurs="unbounded" >
                                        <xs:complexType>
                                                <xs:attribute name="attribute" type="xs:string"
                                                        use="required" />
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
                </xs:complexType>
        </xs:schema>
```

GLOSARIO

BSD: La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

DLL: Dynamic Linking Library (Bibliotecas de Enlace Dinámico), término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda del programa por parte del sistema operativo.

ECMA: Ecma International es una organización internacional basada en membrecías de estándares para la comunicación y la información

GPL: La GPL (General Public License o licencia pública general) es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Gtk+: Conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario (GUI).

LDAP: Protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

Linus Torvalds: es un ingeniero de software finlandés; es más conocido por desarrollar la primera versión del núcleo (kernel) del sistema operativo GNU/Linux.

OpenGL: Especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

SSL: *Secure Sockets Layer*. Protocolo de Capa de Conexión Segura

XPath: Lenguaje que permite construir expresiones que recorren y procesan un documento XML.