Universidad de las Ciencias Informáticas
Facultad 1

Título: DB Evolution. Automatización de procesos en la Gestión de Configuración a los Sistemas de Bases de Datos.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Abdiel Carrazana Saucedo
Rafael Ambruster Crespo

Tutor: MsC. Erik de la Vega García.

Ciudad de la Habana, Junio 2010

No hay nada permanente excepto el cambio.

Heráclito, 500 AdC

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Abdiel Carrazana Saucedo

_____

Autor.

Rafael Ambruster Crespo

_____

Autor.

Erik de la Vega García

_____

Tutor.

AGRADECIMIENTOS

A **mis padres,** por enseñarme todo lo que soy y lo que seré, por su sacrificio y dedicación, por hacer posible que yo llegara aquí y ser siempre el motivo por el que dar lo mejor de mí.

A **mi familia** por darme siempre un lugar en donde encontrar cariño, consejos y alegrías. A **mi hermano** y **mi hermanita** que los quiero con la vida, a **mis abuelos** que son mis segundos padres, a **mis tíos** que me quieren como otro hijo desde que nací. Gracias por ser hijo de tantas personas.

A mi **otra familia**. A **Susana**, mi amor y mi futuro. Todo esto también lo logré gracias a ti mi vida, porque tú lo significas todo, tu eres mi mayor tesoro. A **Rosita** y **Humberto** por acogerme en su casa como un hijo más y quererme y preocuparse por mí. Yo también los quiero mucho.

A **mi tutor** por convertirse en mi amigo más que en mi guía, gracias por confiar en nosotros.

A **Rafa**, yo sabía que no me ibas a fallar aunque los momentos fueran difíciles. Eso te lo voy a agradecer toda la vida mi hermano.

A todos **mis amigos** que son demasiados como para nombrar. Los de hoy y los de ayer, yo siempre voy a estar ahí para Uds.

A **mis profesores**, los que me dieron clases y los que no, Uds. también son parte de esto.

Muchísimas gracias a todos... **Abdiel**.

A **mis padres** por ser el mejor ejemplo que un hijo puede tener, por guiarme a tomar el camino correcto, por enseñarme el propósito de la vida, por confiar en mí, por hacer posible que me convirtiera en la persona que soy y por ser como son, así, perfectos.

A mi **hermana** por mostrarme parte de la realidad de la vida, por brindarme la ayuda necesaria durante el transcurso de la carrera y a su esposo **Julio** por ser un gran amigo.

A **Ileana** que aunque la vida dio una vuelta que ninguno esperaba, me ayudo muchísimo y siempre se lo voy a agradecer.

A mi **familia** en general por apoyarme tanto en los momentos buenos como en los malos.

A mi tutor **Erik** por enseñarme gran parte del conocimiento con el que hoy cuento.

A mi compañero de tesis **Abdiel**, gracias por ser mi segundo hermano, la verdad existen pocas personas como tú, se que vas a lograr grandes cosas en la vida y en mi siempre tendrás alguien en quien confiar.

Al **grupo de desarrollo de software del proyecto identidad**, sería injusto no mencionarlos a todos, porque todos contribuyeron a que lográramos realizar este sueño.

Gracias a todos… **Rafael**.

DEDICATORIA

A nuestros padres…

RESUMEN

Durante la producción de software los cambios son frecuentes, por lo que se hace necesario un proceso que los monitorice y gestione durante todo el ciclo de vida del proyecto. Este proceso es la Gestión de Configuración del Software e incluye tareas como la Identificación del Cambio, el Control de Cambios y el Control de Versiones.

El Centro de Identificación y Seguridad Digital mantiene en funcionamiento un conjunto de bases de datos que son operadas por los desarrolladores. El cambio es algo frecuente en ellas, por lo que es necesario mantener un control sobre las modificaciones a las mismas. No existe en el centro una herramienta que automatice los principales procesos de la Gestión de Configuración a las bases de datos debido a que las usadas históricamente son privadas, solamente dan soporte para gestores específicos, tienen limitaciones en los procesos que automatizan, y únicamente pueden ser usadas en el sistema operativo Windows.

Como resultado de la investigación se desarrolló una herramienta que permite la automatización de los principales procesos de la Gestión de Configuración a los Sistemas de Bases de Datos, principalmente el Control de Cambios y el Control de Versiones. La misma es capaz de establecer este control desde el propio gestor de bases de datos, lo cual permite una mayor velocidad en los procesos, así como un estricto control de la evolución de la base de datos.

PALABRAS CLAVE: Bases de Datos, Cambio, Gestión de Configuración del Software

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Durante el proceso de desarrollo de software los cambios son inevitables. Desde la concepción del producto y la captura de requisitos, y posteriormente desde el inicio del mantenimiento hasta su retiro, se van realizando una serie de cambios, tanto en el código fuente como en la documentación asociada. Los proyectos deben prepararse para el cambio, aplicando medidas que conlleven a evitar confusiones que pueden desperdiciar meses de trabajo de los desarrolladores. La confusión surge cuando los cambios no se analizan antes de realizarlos, no se registran antes de implementarlos, no se reportan a quienes deben saberlo o no se controlan de forma que mejore la calidad y se reduzcan los errores.

Se hace por tanto necesario un proceso que sea capaz de monitorizar los cambios durante todo el ciclo de desarrollo. Esta necesidad se materializa en el proceso de Gestión de Configuración del Software el cual figura en el segundo nivel del CMMI (Modelo de Integración de la Capacidad y Madurez). La Gestión de Configuración del Software es un conjunto de actividades desarrolladas para gestionar los cambios a lo largo del ciclo de vida de un proyecto. Es una actividad de garantía de calidad del software que se aplica en todas las fases del proceso de ingeniería.

Las bases de datos no son ajenas a este proceso. Constantemente los equipos de desarrollo crean nuevas aplicaciones, modifican las existentes y ejecutan actualizaciones de las aplicaciones que involucran cambios a las bases de datos. Existen varios productos disponibles para el tratamiento de los cambios en las bases de datos, generalmente cada uno destinado a realizar funcionalidades específicas. La mayoría de ellos son para uno o un grupo reducido de gestores de bases de datos, que no siempre se corresponden con las necesidades de los equipos de desarrollo. Además, los mejores y más eficientes productos están monopolizados por compañías privadas que imponen grandes sumas de dinero por el uso comercial de sus herramientas.

El Centro de Identificación y Seguridad Digital mantiene en funcionamiento un grupo de bases de datos que son usadas y operadas por los desarrolladores. Los cambios son frecuentes en las mismas, ya que los proyectos suelen pasar por diferentes fases de desarrollo, las cuales no solo tienen como objetivo desarrollar nuevas funcionalidades a las aplicaciones, sino la mejora de las ya existentes. Estas fases provocan gran cantidad de cambios en cada una de las bases de datos, los cuales que son necesarios registrar, archivar e informar a los afectados. No existe en el centro la disponibilidad de una herramienta

que sea capaz de llevar a cabo los principales procesos que se producen durante la Gestión de Configuración en las bases de datos de una forma automatizada y que se adapte a las características propias del proceso de desarrollo. Las usadas comúnmente además de contar con el inconveniente de ser privadas, por lo que su uso legal llevaría al pago de grandes sumas de dinero, son para gestores de bases de datos específicos, por lo que alguno de los gestores que se usan en el centro no se incluyen y además tienen algunas limitaciones en los procesos que ejecutan. Otra desventaja está en que solo es posible usarlas en el sistema operativo Windows, no así para Linux, sistema operativo en el que están instalados varios de los servidores y aplicaciones del centro.

Dada la situación problémica planteada se define como **problema científico** de la investigación: ¿Cómo automatizar los principales procesos de la Gestión de Configuración en los Sistemas de Bases de Datos de los proyectos desarrollados por el Centro de Identificación y Seguridad Digital?

El **objeto de estudio** de la investigación está determinado por el proceso de Gestión de Configuración en los Sistemas de Bases de Datos, aunque se enmarcará el **campo de acción** en el proceso de Gestión de Configuración en los Sistemas de Bases de Datos del Centro de Identificación y Seguridad Digital.

El **objetivo general** de la investigación es desarrollar una herramienta que automatice los principales procesos de la Gestión de Configuración en los Sistemas de Bases de Datos. Para dar cumplimiento a este objetivo se trazaron los siguientes **objetivos específicos:**

1. Realizar el análisis y diseño de la herramienta.
2. Implementar los procesos para el Control de Cambios y Control de Versiones en Sistemas de Bases de Datos.
3. Realizar el diseño y la ejecución de pruebas para la herramienta propuesta.

Se plantea como **hipótesis** de la investigación que: con el desarrollo de una herramienta se logrará automatizar los principales procesos de la Gestión de Configuración en los Sistemas de Bases de Datos de los proyectos desarrollados por el Centro de Identificación y Seguridad Digital. Los anteriores objetivos específicos se desglosan en las siguientes **tareas de la investigación:**

1. Elaboración del Estado del Arte sobre las tendencias actuales en el desarrollo del proceso de Gestión de Configuración en los Sistemas de Bases de Datos.

2. Definición de los requisitos del sistema a implementar.
3. Selección de una metodología de desarrollo de software que guíe el desarrollo de la herramienta para lograr la calidad de la misma.
4. Definición de una arquitectura que garantice que el modelo de negocio sea independiente del Sistema Gestor de Bases Datos en que se encuentren las bases de datos.
5. Implementación de una librería para el acceso a los datos en Oracle.
6. Implementación de un módulo capaz de mantener el control sobre todos los cambios en los objetos que conforman la estructura de una base de datos.
7. Automatización del proceso de extracción de todos los objetos de la base de datos, agrupándolos y clasificándolos en versiones separadas.
8. Automatización de comparaciones entre las estructuras de dos bases de datos.
9. Automatización de la sincronización de dos bases de datos.
10. Ejecución de pruebas de unidad y del sistema para la validación de la herramienta.

El documento está estructurado en cuatro capítulos:

Capítulo 1. Fundamentación Teórica: En este capítulo se abordan los diferentes conceptos que serán tratados a lo largo de toda la investigación, así como los principales aspectos teóricos relacionados con el proceso de Gestión de Configuración del Software. Además se hace una caracterización de las diferentes herramientas y tecnologías que fueron seleccionadas para la construcción de la herramienta.

Capítulo 2. Propuesta de Solución: En este capítulo se hace una descripción de la solución propuesta. Se documentan los primeros pasos en la construcción del sistema, los requerimientos de la herramienta, así como algunos artefactos propuestos por la metodología Scrum.

Capítulo 3. Diseño e Implementación: En este capítulo se documentan los diferentes módulos y componentes del software. Se documenta la arquitectura de la aplicación, así como se realizan diferentes artefactos relacionados con estas fases de desarrollo.

Capítulo 4. Pruebas: En este capítulo se presenta el diseño de un conjunto de pruebas de unidad y del sistema para la validación de la herramienta, así como los resultados alcanzados con las mismas.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el siguiente capítulo se presentan los principales conceptos relacionados con la Gestión de Configuración del Software, Control de Versiones, Control de Cambios, así como su aplicación en el contexto de las bases de datos. Además se caracterizan algunos Sistemas Gestores de Bases de Datos y herramientas que automatizan los procesos anteriores para estos gestores. Se realiza un estudio de las tecnologías de programación, metodologías de desarrollo de software y entornos de desarrollo que serán usados en la implementación de la herramienta.

1.1. Gestión de Configuración del Software.

La Gestión de Configuración del Software (GCS) ha sido un concepto ampliamente tratado por diversos autores. Se puede definir como la identificación única, almacenamiento controlado, control del cambio y el reporte del estado de los productos intermedios del trabajo seleccionados, componentes del producto y el producto, durante la vida de un sistema. (Jonassen Hass, 2002).

Para la IEEE es la disciplina que abarca todo el ciclo de vida de la producción de software y productos asociados. Específicamente, requiere de la identificación de los componentes a controlar y la estructura del producto, controla todos los cambios sobre los elementos y garantiza mecanismos para auditar todas las acciones. (IEEE, 1987).

Sin embargo uno de los conceptos más completos es el ofrecido por Pressman en el libro “Ingeniería de Software. Un enfoque práctico.”, en el que plantea que la GCS es un “…conjunto de actividades diseñadas para controlar el cambio identificando los productos del trabajo que probablemente cambien, estableciendo relaciones entre ellos, definiendo mecanismos para gestionar distintas versiones de estos productos, controlando los cambios realizados, y auditando e informando de los cambios realizados”. (Pressman, 2002). En este libro además se definen cinco tareas fundamentales dentro de este proceso, Identificación, Control de Versiones, Control de Cambios, Auditorías de Configuración y Generación de Informes, algunas de las cuales serán abordadas posteriormente.

Resumiendo, la Gestión de Configuración del Software es una disciplina de control dentro del proyecto, que constituye una garantía de la integridad del producto y que contribuye de forma decisiva en la calidad del producto a desarrollar. Estas características justifican su aparición como uno de los

eslabones principales para alcanzar el nivel dos del CMMI (Modelo de Integración de la Capacidad y la Madurez).

Las bases de datos al igual que los sistemas de software raramente se mantienen estables de acuerdo a su implementación principal. Aunque las estimaciones difieren, la mayoría concuerda en que alrededor del 50 % o más de los esfuerzos de los programadores surgen como resultado de las modificaciones de los sistemas después de terminada su implementación (Roddick, 1995). La aparición de nuevas aplicaciones, actualización de las existentes, nuevas problemáticas y nuevos enfoques en las soluciones, entre otros factores, pueden significar cambios en las bases de datos; cambios que requieren un estricto control y documentación de los mismos. Para ello se hace indispensable una correcta aplicación del proceso de Gestión de Configuración para todo el Sistema de Bases de Datos[1]. Este proceso incluye el Control de Cambios y el Control de Versiones en cada una de ellas.

1.1.1. Elementos de Configuración del Software (ECS).

Entre los primeros pasos en el proceso de Gestión de Configuración del Software se encuentra la identificación de los Elementos de Configuración del Software. Pueden definirse como cualquier producto de trabajo, tanto producto final como productos intermedios y tanto productos entregables al cliente como productos internos del proyecto, cuyo cambio pueda resultar crítico para el buen desarrollo del proyecto. (LNCS, 2008). Estos deben permitir ser identificados de forma única y tener especificadas sus relaciones con el mundo exterior y con otros elementos de configuración. Se pueden considerar como elementos de configuración los siguientes:

1. El plan del proyecto de software.
2. Especificaciones de requisitos.
3. Un prototipo, ejecutable o en papel.
4. El diseño preliminar.
5. El diseño detallado.
6. El código fuente.

[1] Conjuntos de bases de datos utilizadas en el desarrollo de software con alguna relación entre ellas.

7. Programas ejecutables.
8. Manuales de usuario.
9. El manual de operación e instalación.
10. El Plan de Pruebas.
11. Los estándares y procedimientos de ingeniería de software utilizados.
12. Los productos de hardware y software utilizados durante el desarrollo.
13. El diseño de las bases de datos.
14. Otros escogidos por la empresa u organización.

Cada empresa u organización define cuáles elementos serán reconocidos como Elementos de Configuración del Software en base a sus necesidades y requerimientos, así como la determinación de los atributos que permitirán su organización e identificación única.

Uno de los ECS de mayor importancia en el proceso de desarrollo es la versión de la base de datos. En el repositorio del proyecto se almacenan las diferentes versiones por las que va transitando la base de datos, la cual se asocia a un identificador. Una versión no es más que un script[2] que contiene todos los elementos necesarios para la construcción de la base de datos desde cero hasta la versión en cuestión. Por lo tanto este puede contener tablas, procedimientos almacenados, disparadores, vistas, paquetes, funciones, etc., en dependencia del gestor que se trate.

1.1.2. Línea base.

Un concepto fundamental en el tema de la Gestión de Configuración lo constituye la línea base. Una línea base se entiende por un punto de referencia en el desarrollo de software que queda marcado por el envío de uno o más elementos de configuración y que va a servir como base para otros desarrollos posteriores. Antes de que un Elemento de Configuración del Software se convierta en línea base, el cambio se puede llevar a cabo de forma rápida e informalmente, una vez que forma parte de ella solamente puede cambiarse mediante procedimientos formales de control de cambios (Pressman, 2002).

[2] Generalmente es un conjunto de sentencias SQL listas para aplicar en un gestor de bases de datos.

La línea base de un gestor de bases de datos estará conformada por los elementos y objetos por los que está compuesto el mismo. No todos los gestores contienen los mismo tipos de objetos en sus esquemas, incluso algunos ni siquiera soportan varios esquemas en una misma instancia de la base de datos. La forma más común de obtener la lista completa de estos objetos es mediante consultas a los diccionarios de datos, concepto que se tratará en posteriores epígrafes.

1.1.3. Control de Versiones.

El Control de Versiones combina procedimientos y herramientas para gestionar las versiones de los objetos de configuración creados durante el proceso de desarrollo de software. La versión debe expresar el desarrollo histórico de los elementos de configuración, información que pudiera ser la única vía de distinguir a un elemento de otro. La Gestión de Configuración permite a un usuario especificar configuraciones alternativas del sistema de software mediante la selección de las versiones adecuadas. Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego especificar y construir una configuración, describiendo el conjunto de atributos deseado. Una versión del software puede estar constituida por diferentes variantes. Una variante es un conjunto diferente de objetos del mismo nivel de revisión y por tanto, coexiste con otras variantes.

En la actualidad sacar una versión completa de las base de datos es una tarea sencilla y que se puede realizar con una gran cantidad de herramientas. Esto es posible dado que los diferentes gestores proporcionan fáciles mecanismos para hacerlo, generalmente con simples consultas a las estructuras destinadas para tales funciones. La mayoría de ellos proveen a los usuarios de vistas que resumen todos los objetos con algunas de sus características principales, por lo que es muy fácil extraer la información siempre que se cuente con los permisos necesarios. Estas vistas reciben el nombre de Diccionario de Datos o Catálogo del Sistema y generalmente se limitan solo a proveer información, no a modificarla.

Muchos gestores de bases de datos implementan el estándar SQL-99, el cual establece el uso de ciertas estructuras para especificar los metadatos del sistema. En resumen, plantea la creación de un esquema independiente llamado INFORMATION_SCHEMA, el cual seguido de nombres de tablas que también serán estándar, proveerá información sobre los objetos que contiene la base de datos. Gestores como Oracle, SQL Server y otros implementan esta característica, la cual permite la portabilidad de muchas aplicaciones de acceso y gestión de bases de datos.

1.1.4. Control de Cambios.

El propósito del Control de Cambios es el de controlar completamente todos los pedidos de cambios para un producto, así como para los cambios a este implementados anteriormente. Para cualquier ECS debe ser posible la identificación de sus cambios en relación a sus predecesores. Cualquier cambio debe ser posible de rastrear hacia el elemento donde el cambio fue implementado (Jonassen Hass, 2002). El Control de Cambios combina los procedimientos humanos y las herramientas automáticas para proporcionar un mecanismo para el seguimiento a los cambios que ocurren.

El Control de Cambios en las bases de datos debe tener en cuenta dos tipos de cambios fundamentalmente, aquellos realizados directamente sobre los datos que contiene la base de datos, así como los aplicados a los diferentes objetos que conforman su esquema. La forma tradicional de detectar los cambios en una base de datos es mediante la extracción regular de una versión completa de la misma y su comparación con versiones extraídas anteriormente de ella o de otra base de datos. Este proceso puede dar lugar a un script de cambios, el cual no es más que las instrucciones necesarias para trasformar una base de datos de una versión a otra.

Otra de las vías para el seguimiento de estos cambios se puede lograr mediante el uso de disparadores. En la actualidad una buena cantidad de gestores de bases de datos tienen la capacidad de crear disparadores que escuchan los eventos DDL [3]y pueden realizar acciones en consecuencia de estos. De esta forma es posible la inserción de un conjunto de estructuras dentro de la base de datos, que internamente pueden mantener un control de los cambios que van ocurriendo en una base de datos y dar un informe sobre estos, así como la agilización en los procesos de extracción y comparación con otras bases de datos.

En cuanto a los cambios en los objetos DML[4], la mayoría de las aplicaciones hacen una comparación de las tuplas de dos versiones de las tablas y generan un script de cambios, para insertar, eliminar o modificar según sea el caso, a los objetos que resulten diferentes. Es importante destacar que esto solo es posible si los esquemas de las tablas son iguales y no tienen relaciones de integridad con otras tablas. De no cumplirse estas características es bastante riguroso intentar sincronizar ambas tablas.

[3] Se define en el epígrafe 1.2.
[4] Se define en el epígrafe 1.2.

Generalmente la sincronización DML solo se recomienda para los llamados nomencladores[5], los cuales son muy fáciles de manipular dada su gran simplicidad.

1.2. Sistemas Gestores de Bases de Datos (SGBD).

Un Sistema Gestor de Bases de Datos es un conjunto de datos relacionados entre sí y un grupo de programas para tener acceso a esos datos, permitiendo concurrencia y recuperación. Uno de sus objetivos más importantes es proporcionar a los usuarios una visión abstracta de los datos, es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos, pero sin embargo se deben extraer eficientemente. Entre las características más importantes en estos sistemas está la persistencia y la concurrencia. La persistencia hace referencia a la conservación de los datos después de la finalización del proceso que los creó y la concurrencia se refiere a la capacidad del sistema para gestionar a múltiples usuarios interactuando al mismo tiempo.

Todos los SGBD ofrecen lenguajes e interfaces apropiadas para cada tipo de usuario: diseñadores, administradores, programadores de aplicaciones y usuarios finales. Los lenguajes van a permitir al administrador de la base de datos especificar los datos que necesita, su estructura, las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso, las características de tipo físico y las vistas externas de los usuarios. Los lenguajes de los Sistemas Gestores de Base de Datos se clasifican en:

- Lenguaje de definición de datos (DDL): Se utiliza para especificar el esquema de la base de datos, las vistas de los usuarios y las estructuras de almacenamiento. Es el que define el esquema conceptual y el esquema interno. Lo utilizan los diseñadores y los administradores de la base de datos.
- Lenguaje de manipulación de datos (DML): Se utiliza para leer y actualizar los datos de la base de datos. Es el utilizado por los usuarios para realizar consultas, inserciones, eliminaciones y modificaciones.

Otro elemento importante en los Sistemas Gestores de Bases de Datos son los esquemas de objetos. Un esquema de objetos es una colección de objetos de la base de datos que es propiedad de un

[5] Es un término muy usado por los desarrolladores para definir un tipo especial de tabla que solo tiene un identificador y una descripción textual.

usuario. Esto quiere decir que el usuario es el propietario de los objetos de su esquema. Los objetos dentro de los esquemas incluyen estructuras tales como tablas, vistas, disparadores, procedimientos almacenados, índices, etc. Esto hace posible definir reglas de acceso y permisos sobre los usuarios que acceden a la base de datos, lográndose una mayor seguridad. Una gran cantidad de gestores en la actualidad soportan esta característica, sobre todo los más potentes y complejos.

1.2.1. Oracle

Oracle es un sistema de gestión de bases de datos relacional desarrollado por Oracle Corporation. Se considera como uno de los más completos destacando el soporte de transacciones, la estabilidad, escalabilidad y el soporte multiplataforma. La tecnología de Oracle puede encontrarse en casi todos los sectores. Es la primera empresa de software en desarrollar e implementar software empresarial cien por ciento activado por internet en toda su línea de productos: bases de datos, aplicaciones comerciales y herramientas para el soporte de decisiones y el desarrollo de aplicaciones.

El Diccionario de Datos es una parte fundamental de la base de datos Oracle. Está formado por tablas, vistas y paquetes a los que se puede acceder para obtener información. Las tablas se crean automáticamente durante la instalación y permiten saber la estructura lógica o física de la base de datos, los usuarios, las restricciones de integridad sobre las tablas, el espacio asociado a cada objeto en la base de datos y la cantidad que se está utilizando por los distintos objetos creados por los usuarios. El usuario SYS es el que tiene todos los permisos sobre cualquier objeto de la base de datos, así como de los demás usuarios. El Diccionario de Datos de Oracle está compuesto por:

- Tablas base: Una serie de tablas a las que el servidor de bases de datos accede cada vez que se procesa una instrucción DDL o en algunos comandos DML.
- Vistas estáticas: Se crean durante la instalación del sistema y decodifican y resumen la información contenida en las tablas base. Durante la creación de estas vistas se generan sinónimos públicos para proveer el acceso a los usuarios del gestor. Estas vistas deben ser utilizadas para las labores de administración rutinarias que necesiten información específica sobre la configuración y estado de la base de datos. Tienen el nombre de estáticas porque no mantienen información relacionada con las sesiones. Se dividen en tres categorías:

1. Vistas con prefijo USER: Puede utilizarlas cualquier usuario de la base de datos y se refieren a objetos poseídos por dicho usuario.
2. Vistas con prefijo ALL: Las podrá usar cualquier usuario y además añaden la columna OWNER al resto de información. Con estas vistas se puede tener acceso a la información de los objetos que el usuario es dueño, además de los objetos públicos y los que el usuario tiene acceso.
3. Vistas con prefijo DBA: Dan información sobre todos los objetos de la base de datos. Usualmente también tienen la columna OWNER. Sólo las puede utilizar el administrador o usuarios con privilegio SELECT ANY TABLE o que pertenezca a un rol que incluya este privilegio.
4. Vistas dinámicas: Incluyen información sobre las condiciones actuales de operación del gestor. La mayoría de ellas son creadas durante la instalación del gestor y algunas se crean específicamente para monitorear ciertas actividades. Todas se identifican por el prefijo V$.

Oracle cuenta con uno de los esquemas más complejos en cuanto a variedad de estructuras y objetos, lo que lo hace una herramienta potente y de gran personalización. En general estos objetos se dividen en dos grupos, los pertenecientes al servidor y los propios de cada una de las bases de datos. Los objetos que describen características del servidor son los directorios, perfiles, roles, parámetros de configuración, usuarios, y los espacios de trabajo, mientras que los relacionados a cada una de las instancias de una base de datos son los clusters, funciones, librerías, vistas, vistas materializadas, paquetes y sus cuerpos, tablas, procedimientos almacenados, secuencias, sinónimos, tipos y sus cuerpos, disparadores y los códigos java.

1.2.2. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto el desarrollo de PostgreSQL no es manejado por una sola empresa, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

PostgreSQL es una poderosa herramienta, de código abierto. Tiene más de 15 años de desarrollo activo y una probada arquitectura que va ganando una fuerte reputación por confiabilidad, integridad de los datos y exactitud. Se ejecuta en la mayoría de los sistemas operativos, incluidos Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows (PostgreSQL Global Dvelopment Group, 2010). Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, no es un sistema de gestión de bases de datos puramente orientado a objetos.

PostgreSQL tiene dos formas fundamentales de ofrecer metainformación. Una es mediante el Esquema de Información y la otra mediante los catálogos. El Esquema de Información en sí es un esquema llamado INFORMATION_SCHEMA que existe en todas las bases de datos. Su dueño es el usuario inicial en el cluster de bases de datos y ese usuario naturalmente tiene todos los privilegios en este esquema. En general consiste en un conjunto de vistas que contienen información sobre los objetos definidos en cada instancia de una base de datos. (PostgreSQL, 2010) Está definido en el estándar SQL-99, que hace referencia a la aparición de este INFORMATION_SCHEMA y vistas como COLUMN, TABLES, TRIGGERS, VIEWS, entre otras.

Cada base de datos PostgreSQL posee además un esquema denominado pg_catalog, el cual contiene todas las tablas del sistema. Estas tablas del sistema almacenan los metadatos del sistema en sí. Ejemplos de estas tablas son pg_database, pg_class, pg_attribute, las cuales pueden proveer información muy valiosa.

El proyecto PostgreSQL continúa haciendo lanzamientos principales anualmente y lanzamientos menores de reparación de bugs, todos disponibles bajo la licencia BSD, basados en contribuciones de proveedores comerciales, empresas contribuyentes y programadores de código abierto mayormente.

1.2.3. SQL Server

SQL Server es un sistema de bases de datos relacional cliente/servidor que usa Transact-SQL para enviar mensajes entre el cliente y la base de datos. Constituye la alternativa de Microsoft a otros potentes Sistemas Gestores de Bases de Datos como son Oracle, Sybase ASE, PostgreSQL,

Interbase y MySQL. Tiene una gran integración con la plataforma Windows de 32 bit. En particular está diseñado para tomar ventaja de las funcionalidades de los sistemas operativos Windows NT, como son la encriptación, seguridad de red y algunos de sus protocolos de transmisión.

SQL Server cuenta con una estructura llamada Catálogo del Sistema el cual es el que provee los metadatos para este gestor. Está compuesto por un conjunto de vistas que describen los objetos de una instancia de SQL Server. Este catálogo tiene cinco vías fundamentales de acceder a él:

1. Vistas del Catálogo: Provee acceso a los datos almacenado en todas las bases de datos del servidor. Esta es la vía más eficiente de acceder a los metadatos del servidor ya que es la vía más directa de obtener y transformar estos metadatos. Además los nombres de las vistas y sus columnas son muy descriptivos.
2. Información de las Vistas del Esquema: Están basados en la definición de vistas del catálogo del estándar ISO. Estas presentan la información del catálogo en un formato que es independiente de la implementación de cualquier tabla del catálogo y por tanto no son afectados por los cambios en estas tablas.
3. Vistas de Compatibilidad. Son un conjunto de tablas que existían en versiones antiguas de SQL Server y fueron implementadas por motivos de mantener la compatibilidad entre todas las versiones del gestor. No es recomendado extraer los metadatos de estas vistas dado que presentan algunas deficiencias que fueron solucionadas en las Vistas del Catálogo.
4. OLE DB Schema Rowset: Define una interface IDBSchemaRowset que contiene la información del catálogo. Es un esquema específicamente diseñado para los proveedores de SQL Server Native Client OLE DB.
5. Funciones del Catálogo ODBC: Definen un conjunto de funciones del catálogo que son métodos estándar soportados por diferentes drivers ODBC. Soporta dos tipos de funciones que ayudan a obtener la información del catálogo de los servidores conectados, SQLLinkedServers6 y SQLLinkedCatalogs7.

[6] Lista de los servidores conectados definidos en el servidor local.
[7] Lista de catálogos contenidos en el servidor conectado.

Antes de SQL Server 2005 cada esquema era equivalente a un usuario de la base de datos, sin embargo en la actualidad cada esquema es un espacio de nombres distinto que existe de forma independientemente del usuario que lo creó, es decir, un esquema simplemente es un contenedor de objetos. Cualquier usuario puede ser propietario de un esquema, y esta propiedad es transferible.

1.3. Herramientas para la Gestión de Configuración en bases de datos.

Muchas son las herramientas que se usan a nivel mundial para la automatización de los procesos en la Gestión de Configuración en los Sistemas de Bases de Datos. Prestigiosas compañías como Embarcadero, Oracle y otras se han dedicado a proveer facilidades para la ejecución de estas tareas que a veces se vuelven engorrosas. De forma general estas herramientas cuentan con las siguientes características:

- Un repositorio o lugar de almacenamiento: Incluyen un lugar en que se van almacenando las versiones de los diferentes objetos que son de interés por los desarrolladores, a los cuales pueden acceder los diferentes miembros del equipo.
- Monitorización del cambio: Presentan mecanismos para monitorizar los cambios de los objetos de interés, ya sea en el mismo momento en que ocurre o al producirse los análisis programados.
- Comparación del código o de ficheros: Contienen algoritmos capaces de comparar y guardar los cambios que se detectan de un fichero a otro. Para esta tarea en muchas ocasiones hacen uso de otros programas o librerías externas que incluyen en su aplicación.
- Revisión histórica: Con cada versión se almacenan las notas y comentarios que se hacen durante cada revisión.

Otra de las características de las herramientas de este tipo es la de realizar el Control de los Cambios de las bases de datos a través de comparaciones entre versiones de los esquemas de dos instancias de bases de datos, los cuales son extraídos en el momento en que se decide realizar la comparación. Este modelo aunque resuelve la necesidad principal de identificar el cambio, no permite, por ejemplo, determinar en qué momento se produjo el cambio o quién lo realizó, lo cual representa muchas veces un serio inconveniente. Las comparaciones entre los esquemas necesitan de la extracción explícita de los textos DDL de todos los objetos a comparar de las dos bases de datos, lo que puede consumir una

gran cantidad de tiempo y recursos. En epígrafes posteriores quedará demostrado que esta no es la mejor forma de realizar estos procesos con la proposición de un nuevo modelo.

1.3.1. Embarcadero Change Manager.

Una de las herramientas más usadas y de mayor prestigio internacional es Embarcadero Change Manager de la compañía Embarcadero. La misma está compuesta por un poderoso conjunto de herramientas para la administración del ciclo de vida de las bases de datos. Change Manager es un comparativo de la base de datos, una herramienta para los cambios y la sincronización, que genera reportes y reconcilia diferencias entre las bases de datos, tablas, esquemas y otros objetos de la base de datos. Con Change Manager es posible comparar fácilmente entre diversos tipos de bases de datos y crear bases de esquemas y configuraciones. Las capacidades para el mapeo sofisticado de objetos permiten seleccionar sólo aquellos necesarios y acelerar el trabajo de comparación.

Soporta los siguientes gestores de bases de datos:

- IBM.
- DB2 para LUW v8, v9 y v9.5.
- Microsoft SQL Server 2000, 2005 y 2008.
- Oracle 8i, 9i, 10g y 11g.
- Sysbase ASE 12.5 y 5.

Change Manager ofrece a los desarrolladores visibilidad de lo que está sucediendo en el desarrollo, en las pruebas y en los entornos de producción. Se pueden capturar instantáneas de los objetos cambiados y generar automáticamente una secuencia de comandos de cambio, eliminando la necesidad de determinar la sintaxis correcta de los objetos dependientes y garantizar que se conserven los datos. Se integra con el control de código fuente, para que los desarrolladores puedan correlacionar SQL, esquema o cambios de datos de referencia con los cambios de código de la aplicación mediante la vinculación de un archivo con la versión correspondiente del código. Ayuda a los administradores a hacer cambios de rumbo a las estructuras, los usuarios, los permisos y la configuración de bases de datos. Change Manager también permite especificar los estándares de

configuración, que pueden ser utilizados para verificar cientos de bases de datos e identificar rápidamente los riesgos de seguridad.

Permite la automatización de tareas manuales como las comparaciones entre bases de datos, presentación de informes de diferencia, y el desarrollo de secuencias de comandos de sincronización. Además, proporciona apoyo para solucionar problemas de bases de datos mediante la retención de los estados de las bases de datos y versiones anteriores, además de la fuente viva. Embarcadero Change Manager no admite una opción de sincronización directa, sino que siempre genera una secuencia de comandos SQL, los que a su vez pueden ser ejecutados de inmediato en la base de datos. Una clara ventaja de este procedimiento es el aspecto de seguridad del método. El riesgo de accidente de ejecución de un script que puede causar daños a los datos de producción es reducido.

Una característica muy notable de la herramienta es la de guardar los resultados de sus tareas en archivos que pueden ser explorados, o utilizados para otros fines. Puede actuar como "referencia" o "maestro" para comparar con bases de datos en línea u otros archivos para generar un informe histórico que indica lo que ha cambiado entre los puntos en el tiempo. Proporciona mecanismos para generar reportes en HTML y los incluyen en correos; por ejemplo, todos los componentes de Change Manager para almacenar un informe se generan en “.csv” como formato de salida. De esta forma se puede usar la salida de Change Manager con muchas otras aplicaciones de terceros como Microsoft Excel.

1.3.2. Oracle Change Management Pack para Oracle.

Otra de las herramientas más usadas a nivel mundial es el Oracle Change Management Pack. Oracle Change Management Pack para bases de datos Oracle ofrece una solución integrada para que los administradores de base de datos y los desarrolladores de aplicaciones administren los cambios de la base de datos. Permite comparar los objetos de los esquemas antes y después de una actualización, así como rastrear los cambios en los parámetros de inicialización, autorización y almacenamiento de la base de datos. El mismo está totalmente integrado con Oracle Enterprise Manager y permite a los desarrolladores de aplicaciones acceder a las definiciones de esquema y realizar comparaciones de líneas de referencia, versiones y bases de datos.

Change Management Pack puede generar definiciones SQL para objetos específicos, un esquema determinado o toda la base de datos. Los desarrolladores y administradores cuentan con la misma flexibilidad en la realización de comparaciones para objetos específicos, un grupo de esquemas o toda la base de datos. Además es posible rastrear rápidamente, analizar y administrar los cambios de la base de datos a fin de acelerar los ciclos de actualización y reducir el tiempo de baja de las aplicaciones.

Management Pack introduce herramientas de captura y comparación de las definiciones de los metadatos. Estas herramientas proporcionan un sistema de gran alcance para el seguimiento de los cambios en el Diccionario de Datos dentro de una única base de datos y las diferencias entre varias bases de datos existentes. Permite a los usuarios llevar a cabo actualizaciones y otros cambios complejos del esquema en la base de datos, con seguridad y sin pérdida de datos. Management Pack integra perfectamente la capacidad de capturar las versiones y comparar las definiciones del diccionario de metadatos. Los usuarios acceden a las bases de datos de referencia y a las comparaciones. Se puede programar la captura inmediata o diferida, las versiones pueden ser capturadas en un horario irregular por lo que los trabajos se pueden programar para momentos en que la base de datos no está ocupada. Cada nueva versión de la base de datos de referencia contiene las definiciones, tal como existen en ese momento. Después de que la versión de referencia inicial es capturada, las nuevas versiones se procesan rápidamente y toman poco espacio en el repositorio.

Los usuarios pueden examinar el contenido de la versión, navegar a través de las detalladas definiciones de los objetos. También pueden generar definiciones de SQL para objetos individuales o para una versión completa.

1.4. Tecnología de programación Java.

Java es una tecnología orientada al desarrollo de software con la cual podemos realizar cualquier tipo de programa. Está compuesta básicamente por dos elementos: el lenguaje Java y su plataforma. La plataforma se refiere a la Máquina Virtual de Java (Java Virtual Machine). Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE.

1.4.1. Lenguaje Java.

Java es un lenguaje de programación orientado a objetos creado por Sun MicroSystems al comienzo de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Una de las principales características que favorece el crecimiento y difusión del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Uno de los primeros triunfos de Java fue que se integró en el navegador Netscape y permitió ejecutar programas dentro de una página web, hasta entonces impensable con el HTML.

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir conexiones con otras máquinas y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria. Soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Además Java se beneficia todo lo posible de la tecnología orientada a objetos, no intentando conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Las librerías nuevas o actualizadas no paralizan las aplicaciones actuales (siempre que mantengan el API anterior).

El conjunto de paquetes disponibles, que siempre está creciendo, ya sea por aportes nuevos de Sun o por la creación de paquetes por otras empresas, permite que Java sea una tecnología totalmente extensible. También permite conectarse con librerías nativas, con lo que esta extensibilidad prácticamente no tiene límites

1.4.2. Máquina Virtual de Java.

La Máquina Virtual de Java(JVM) es la responsable de ejecutar el código escrito en lenguaje Java resultante de la compilación conocido como bytecode, y de interpretarlo a las instrucciones nativas de la plataforma destino, permitiendo que una misma aplicación Java pueda ser ejecutada en una gran variedad de sistemas con arquitecturas diferentes. La desventaja de utilizar los bytecodes es la velocidad. Estos programas tienen una menor velocidad de ejecución, ya que previamente a

ejecutarse sobre el sistema, deben ser procesados por el intérprete. Esto se compensa con la ventaja de poder ejecutar programas Java sobre casi cualquier sistema operativo o arquitectura. La JVM incluye las herramientas necesarias para escribir, testear, y depurar aplicaciones y applets de Java. Incluye tres plataformas principales:

- Plataforma Java, Edición Estándar (Java Platform, Standard Edition), o Java SE
- Plataforma Java, Edición Empresa (Java Platform, Enterprise Edition), o Java EE
- Plataforma Java, Edición Micro (Java Platform, Micro Edition), o Java ME

Existen varias versiones en orden cronológico, de la máquina virtual de Java; actualmente está disponible la versión 6. En general la definición del Java bytecode no cambia significativamente entre versiones, y si lo hace, los desarrolladores del lenguaje procuran que exista compatibilidad hacia atrás con los productos anteriores.

1.4.3. Netbeans IDE.

Netbeans es un Entorno de Desarrollo Integrado de código abierto y gratuito para los desarrolladores de software. Ofrece todas las herramientas necesarias para crear aplicaciones de escritorio profesionales, empresariales, web y móviles con el lenguaje Java, JavaFX, C/C ++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. Netbeans es fácil de instalar y se puede ejecutar tanto en Windows, Linux, Mac OS X como en Solaris (Sun Microsystem, 2010). La plataforma Netbeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases escritas para interactuar con las APIs de Netbeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma Netbeans pueden ser extendidas fácilmente por otros desarrolladores de software.

Netbeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de cien socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto Netbeans en junio del 2000 y continúa siendo el patrocinador principal de los proyectos. Es considerado el IDE estándar en el desarrollo con Java, por lo que su

constante renovación y mejora es una prioridad en la comunidad mundial que colabora con este lenguaje.

1.4.4. JDBC para Java.

JDBC es un API para ejecutar sentencias SQL (Structured Query Language) que ofrece una interfaz estándar para el acceso a las bases de datos. Ofrece un acceso uniforme a un amplio número de bases de datos. El código de esta API está completamente escrito en Java. Básicamente JDBC permite: establecer una conexión con una base de datos, enviar sentencias SQL y procesar los resultados. Actualmente se ha liberado la versión 2.0 de JDBC, en la que se ha mejorado el acceso a datos, haciéndolo más potente a través de la creación de diferentes tipos de cursores para acceder a los datos. (Esteban, 2000).

JDBC es un paquete que no forma parte de la Máquina Virtual de Java, pero está incluido como una parte estándar del API general de Java. No es necesaria ninguna preinstalación de la librería para su uso y la misma es suministrada por varios vendedores de gran prestigio a nivel mundial. Entre ellos podemos encontrar a Borland Internacional Inc., IBM's Database 2 (DB2), Oracle Corporation, Sybase, Inc., Symantec, XDB Systems, Inc., así como muchas otras compañías. (Patel, 1996). Es independiente de la plataforma al estar escrito completamente en Java.

1.5. Metodologías de desarrollo de software.

En un proyecto de desarrollo de software la metodología define quién debe hacer qué, cuándo y cómo debe hacerlo. No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería. Generalmente se clasifican en dos grupos, las tradicionales como el Proceso Unificado de Rational (RUP), y las metodologías ágiles como Extreme Programing (XP) y Scrum. Las primeras están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y mantener una buena relación con el cliente.

1.5.1. Scrum.

Scrum es una metodología de desarrollo muy simple, donde la gestión no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Es un modelo de referencia que define un conjunto de prácticas y roles que pueden tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el Scrum Master, que mantiene los procesos y trabaja de forma similar al director de proyecto, el Product Owner, que representa a los stakeholders (clientes externos o internos), y el Scrum Team que incluye a los desarrolladores.

En la **Figura 1** se muestran sus características más importantes. El desarrollo de software se realiza mediante iteraciones, denominadas sprint, con una duración entre 15 y 60 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Algunos elementos importantes dentro de Scrum son:

- Product Backlog: Contiene los requisitos del sistema. Se parte de la visión del resultado que se desea obtener y evoluciona durante el desarrollo. Es el inventario de las características que el propietario del producto desea obtener, ordenado por orden de prioridad.
- Sprint Backlog: Contiene la lista de los trabajos que realizará el equipo durante el sprint para generar el incremento previsto. El equipo asume el compromiso de la ejecución. Las tareas están asignadas a personas y tienen estimados el tiempo y los recursos necesarios. (Palacios, 2007).

Muchos especialistas consideran a Scrum como una metodología para la gestión del proyecto, y es que ese ha sido su uso en la mayoría de los casos. Scrum es una guía para el desarrollo de software y su aplicación junto con otra metodología ágil como puede ser XP forma una poderosa fusión para el desarrollo de software. Esto es posible gracias a la gran similitud que existen entre ambas metodologías, donde los principios de una no contradicen a los de la otra, sino que los complementan. Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de muchos sistemas de software.

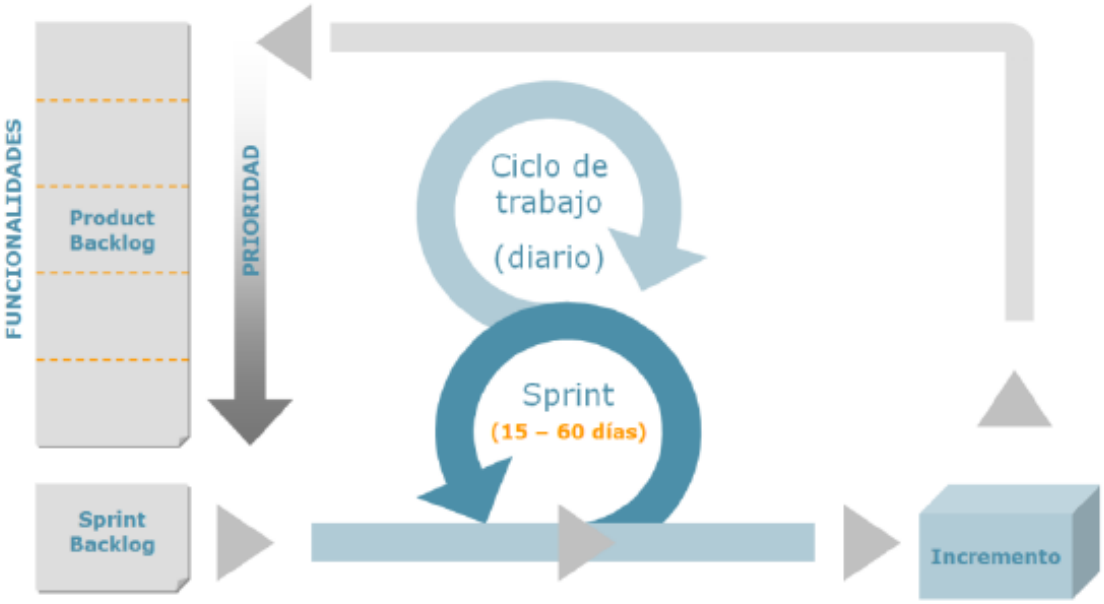

Figura 1. Ciclo de vida de Scrum.

Conclusiones.

En este capítulo ha sido expuesta la información recopilada referente al proceso de Gestión de Configuración del Software, diferentes Sistemas Gestores de Bases de Datos, tecnologías de programación y metodologías de desarrollo utilizadas en el desarrollo de la herramienta. Para la implementación de la nueva herramienta se escoge como lenguaje de programación a Java, sobre todo por su característica de poder adaptarse a varios sistemas operativos, como por ejemplo Linux, el cual es uno de los más usados en el centro para los servidores de bases de datos; además por ser un lenguaje fuerte y completo y en el que mayor entrenamiento tiene el equipo de desarrollo. Como metodología para el desarrollo se usará Scrum que es la más adecuada para el proceso de desarrollo, que cuenta con solo dos personas, poco tiempo para el desarrollo y frecuentes cambios en los requisitos de la aplicación. La herramienta además debe estar diseñada para interactuar con varios gestores de bases de datos, entre los que se priorizará al gestor Oracle por ser el de mayor importancia para el centro.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN

En este capítulo se hará la descripción de la propuesta de solución. Se definirán los requisitos funcionales para el sistema, así como los no funcionales, teniendo en cuenta la importancia de cada uno para el cliente. Se realizan diferentes artefactos para la gestión del proyecto como son el Product Backlog y cada uno de los Sprint Backlog en los cuales se definen las acciones a realizar durante cada iteración.

2.1. Descripción de la solución. DB Evolution.

DB Evolution es una herramienta desarrollada en la plataforma Java, enfocada al control de los cambios y sincronización entre bases de datos. Fue diseñada con el principio de proveer un entorno de desarrollo fácil y amigable a los desarrolladores que se relacionan con el control y la administración de las bases de datos.

El objetivo fundamental de la herramienta es automatizar los principales procesos de la Gestión de Configuración a los Sistemas de Bases de Datos, pudiendo configurar el proceso de acuerdo a las necesidades del equipo de desarrollo en particular. La herramienta cuenta con una arquitectura que le permite adaptarse a varios gestores de bases de datos en dependencia de los diferentes módulos que le sean agregados en forma de plugins, contando inicialmente sólo con el módulo perteneciente al gestor Oracle.

De este momento en adelante se tratará al usuario encargado de interactuar con la aplicación como el Gestor de Configuración de Bases de Datos, siendo este el rol definido para esa función. Dentro de las funcionalidades de la herramienta se encuentra primeramente el registro de las bases de datos dentro de la aplicación, necesitándose para ello los datos correspondientes del servidor y una cuenta con privilegios de acceso a la base de datos, aunque la opción de conectarse no es necesaria ejecutarla hasta el momento de realizar una tarea específica. El Gestor de Configuración de Bases de Datos podrá entonces seleccionar una o varias bases de datos con las que desea interactuar y realizar diferentes tareas como pueden ser extraer una versión completa o parcial del esquema de objetos de la base de datos. Le será posible comparar dos bases de datos, en línea o almacenadas por la aplicación, en busca de diferencias y generar una secuencia de comandos en forma de script que lograrían la sincronización entre estas dos instancias, o sea, hacer a una de ellas idénticamente igual a su compañera.

La herramienta guarda en ficheros el resultado de cada una de sus tareas, como por ejemplo una versión de una base de datos. Estos ficheros funcionan como una base de datos en vivo, por lo que es posible utilizarlos para comparar con otras bases de datos conectadas, así como con otros ficheros, con el objetivo de obtener las diferencias en un período de tiempo determinado.

Dada la característica de la herramienta de interactuar con datos sensibles e importantes como son los almacenados en una bases de datos, fue necesario aplicar un grupo de medidas de seguridad que garantizaran la integridad de los datos obtenidos y almacenados en la herramienta. Primeramente se implementó la protección del repositorio de la herramienta en el cual están almacenados todos los datos extraídos por la misma y todos los parámetros de conexión, mediante un algoritmo simétrico reversible, en este caso Advanced Encryption Standard (AES), también conocido como Rijndael, el cual es uno de los algoritmos más populares usados en criptografía simétrica y con un alto nivel de seguridad. Para el envío y recibo de la información con las bases de datos se debe tener en cuenta los algoritmos soportados por el SGBD. En el caso de Oracle para la encriptación de la información se utilizó el algoritmo RC4_40, y para proteger la integridad de la información se realizará la suma de chequeos por el algoritmo MD5.

2.2. Especificación de requisitos.

Según la IEEE un requisito se define como:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación documentada de una condición o capacidad como en 1 ó 2.

(IEEE, 1990).

Todas las ideas que los clientes, usuarios y miembros del equipo del proyecto tengan acerca de lo que debe hacer el sistema, deben ser analizadas como candidatas a requisitos. De forma general estos se clasifican en dos categorías: requisitos funcionales y requisitos no funcionales.

2.2.1. Requisitos funcionales.

Los requisitos funcionales son las capacidades o condiciones que el sistema debe cumplir. Un requisito que especifica una función que el sistema o un componente del sistema debe ser capaz de realizar.

A continuación se exponen los requisitos funcionales para la herramienta DB Evolution.

Rol	**Objetivo**
Gestor de Configuración de Bases de Datos	Es el encargado de realizar el proceso de Gestión de Configuración en las bases de datos.

Tabla 1. Roles del sistema.

RF 1. **Guardar configuración.**

Propósito	Permite guardar automáticamente la configuración existente en la aplicación al cerrar.
Roles	Se realiza de forma automática.
Precondiciones	1. No puede estar ejecutándose ninguna tarea.
Descripción	1.1. Desconectar las bases de datos. 1.1.1. Cerrar conexión con el servidor en las bases de datos conectadas. 1.2. Salvar configuración. 1.2.1. Salvar las bases de datos registradas. 1.2.2. Salvar los grupos de bases de datos registrados. 1.2.3. Salvar las tareas creadas.
Validaciones	Para realizar las operaciones: 1. Validar que las bases de datos estén desconectadas antes de salvar.
Postcondiciones	1. Todas las bases de datos registradas y sus objetos están guardados en un fichero local.
Prototipo	

Tabla 2. Requisito funcional “Guardar configuración”.

RF 2. **Cargar configuración**.

Propósito	Permite cargar automáticamente una configuración guardada al abrir la aplicación.
Roles	Se realiza de forma automática.
Precondiciones	1. El sistema debe estar cerrado.
Descripción	2.1 Cargar el fichero de configuración. 2.1.1 Buscar fichero. 2.1.2 Adicionar la configuración guardada a la aplicación. 2.1.2.1 Adicionar las bases de datos guardadas a la aplicación. 2.1.2.2 Adicionar los grupos de bases de datos guardadas a la aplicación. 2.1.2.3 Adicionar las tareas guardadas a la aplicación.
Validaciones	Para realizar las operaciones: 1. Validar las excepciones de serialización.
Postcondiciones	1. El sistema quedará listo para el uso y con todas las bases de datos y sus objetos disponibles.
Prototipo	

Tabla 3. Requisito funcional “Cargar configuración”.

RF 3. **Conectar base de datos.**

Propósito	Permite establecer la conexión con una base de datos registrada.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. La base de datos debe estar anteriormente registrada en el sistema.
Descripción	3.1. Seleccionar la base de datos. 3.2. Abrir la conexión con el servidor. 3.2.1.Pedir los parámetros de conexión que no eran necesarios para el registro de la base de datos pero si para establecer la conexión. 3.3. Cambiar estado de la base de datos en el explorador de la aplicación. 3.3.1.Asignar a esta base de datos el estado conectado.

Validaciones	Para realizar las operaciones: 1. Validar que la base de datos esté desconectada. 2. Validar que se hayan llenado todos los parámetros de conexión. 3. Validar el formato de los parámetros de conexión.
Postcondiciones	1. La base de datos queda conectada y lista para realizar tareas sobre la misma.
Prototipo	

Tabla 4. Requisito funcional “Conectar base de datos”.

RF 4. **Desconectar base de datos.**

Propósito	Permite cerrar la conexión de una base de datos conectada.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. La base de datos debe estar anteriormente registrada en el sistema. 2. La base de datos debe encontrarse conectada.
Descripción	4.1. Seleccionar la base de datos. 4.2. Cerrar la conexión con el servidor. 4.3. Cambiar el estado de la base de datos en el explorador de la aplicación. 4.3.1.Asignar a esta base de datos el estado desconectado
Validaciones	
Postcondiciones	1. La base de datos queda desconectada .Se cierra su conexión con el servidor de bases de datos.
Prototipo	

Tabla 5. Requisito funcional “Desconectar base de datos”.

RF 5. **Instalar plugin.**

Propósito	Permite instalar en tiempo de ejecución un plugin disponible para el acceso a las bases de datos de un gestor determinado.

Roles	Se realiza de forma automática. Gestor de Configuración de Bases de Datos.
Precondiciones	1. Existencia de un plugin disponible para la aplicación que no esté instalado.
Descripción	5.1. Buscar plugin. 5.1.1.Mostrar Buscar Plugin. (para el Gestor de Configuración). 5.1.2.Cargar el fichero seleccionado en el sistema. 5.2. Instalar plugin. 5.2.1.Verificar que el fichero contiene un plugin. 5.2.2.Adicionar el plugin a la colección de plugins de la aplicación.
Validaciones	Para realizar las operaciones: 1. Validar que exista el directorio /plugins en la carpeta de instalación. 2. Validar que el plugin cumpla con la estructura requerida por la aplicación.
Postcondiciones	1. El sistema se encuentra listo para interactuar con las versiones del gestor de bases de datos para el cual implementa sus acciones el plugin instalado.
Prototipo	

Tabla 6. Requisito funcional “Instalar plugin”.

RF 6. **Registrar base de datos.**

Propósito	Permite registrar una base de datos en la aplicación.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. Debe existir una base de datos disponible para conectarse. 2. Debe estar instalado el plugin para el acceso a datos a la base de datos que se quiere registrar.
Descripción	6.1. Mostrar los plugins disponibles para el acceso a datos. 6.2. Registrar los parámetros necesarios para el plugin seleccionado. 6.2.1.Mostrar los parámetros necesarios que establece el plugin. 6.3. Registrar la ubicación de la base de datos dentro del explorador de la

	aplicación. 6.3.1. Mostrar el explorador de la aplicación. 6.4. Mostrar la configuración general de la base de datos. 6.4.1. Mostrar los parámetros necesarios que establece el plugin y los valores entrados por el Gestor de Configuración de Bases de Datos. 6.4.2. Mostrar la dirección de la ubicación de la base de datos dentro del explorador de la aplicación. 6.5. Agregar la base de datos al explorador de la aplicación.
Validaciones	Para realizar las operaciones: 1. Validar que los parámetros mínimos necesarios para establecer la conexión sean entrados. 2. Validar que el formato de cada uno de los parámetros de conexión entrados sean correctos.
Postcondiciones	1. Se agregará un nuevo nodo al explorador de bases de datos de la aplicación para representar a la base de datos registrada
Prototipo	**Anexo 2**.

Tabla 7. Requisito funcional “Registrar base de datos”.

RF 7. **Modificar registro de una base de datos.**

Propósito	Permite modificar los parámetros de conexión para una base de datos registrada.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. La base de datos a modificar debe estar desconectada.
Descripción	7.1. Seleccionar la base de datos. 7.2. Mostrar los valores de los parámetros de conexión de la base de datos. 7.2.1. Mostrar los parámetros que establece el plugin para esta base de datos y los valores registrados por el Gestor de Configuración de Bases de Datos.

	7.3. Registrar los nuevos parámetros de conexión para esta base de datos.
Validaciones	Para realizar las operaciones: 1. Validar que sean entrados los parámetros mínimos necesarios para el plugin de la base de datos. 2. Validar que el formato de cada uno de los parámetros de conexión entrados sean correctos
Postcondiciones	1. La base de datos queda con los nuevos parámetros de conexión registrados.
Prototipo	**Anexo 3**.

Tabla 8. Requisito funcional “Modificar registro de una base de datos”.

RF 8. **Eliminar registro de una base de datos.**

Propósito	Permite eliminar una base de datos registrada.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. La base de datos debe estar previamente registrada en el sistema.
Descripción	8.1. Seleccionar la base de datos. 8.2. Eliminar la base de datos del explorador de la aplicación. 8.2.1.Desconectar la base de datos. 8.2.1.1. Cerrar la conexión con el servidor. 8.2.2.Eliminar la base de datos.
Validaciones	Para realizar las operaciones: 1. Validar que la base de datos esté desconectada antes de eliminar.
Postcondiciones	1. Se elimina el registro de la base de datos de la aplicación.
Prototipo	

Tabla 9. Requisito funcional “Eliminar registro de una base de datos”.

RF 9. **Crear grupo de bases de datos.**

Propósito	Permite crear un grupo de bases de datos donde cada grupo tendrá una base de

	datos maestra y varias bases de datos esclavas.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. Todas las bases de datos deben estar registradas en la aplicación.
Descripción	9.1. Seleccionar la base de datos maestra. 9.2. Seleccionar las bases de datos esclavas. 9.3. Insertar los parámetros de registro del grupo. 9.3.1.Insertar el nombre del grupo. 9.3.2.Insertar la descripción del grupo. 9.4. Instalar el Monitor de Esquema en las bases de datos. 9.4.1.Instalar el Monitor de Esquema en la base de datos maestra. 9.4.2.Instalar el Monitor de Esquema en cada base de datos esclava. 9.5. Agregar un nuevo grupo al explorador de la aplicación.
Validaciones	Para realizar las operaciones: 1. Validar que la base de datos maestra y todas las esclavas se encuentren conectadas. 2. Validar que la conexión de todas las bases de datos tenga privilegios de administración. 3. Validar que el Monitor de Esquema no haya sido instalado anteriormente en una de las bases de datos. 4. Validar que una base de datos solo puede ser esclava de un solo grupo. 5. Validar que las bases de datos maestras y esclavas tengan asociado el mismo plugin de acceso a datos.
Postcondiciones	1. Queda registrado en la aplicación un nuevo grupo de bases de datos listo para realizar acciones sobre él.
Prototipo	**Anexo 4**.

Tabla 10. Requisito funcional “Crear grupo de bases de datos”.

RF 10. **Eliminar grupo de base de datos.**

Propósito	Permite eliminar un grupo de bases de datos de la aplicación.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. Debe existir al menos un grupo de bases de datos registrados en la aplicación.
Descripción	10.1. Seleccionar un grupo. 10.2. Eliminar el grupo de bases de datos. 10.2.1. Desinstalar el Monitor de Esquema de todas las bases de datos del grupo. 10.2.2. Eliminar el grupo de bases de datos de la aplicación.
Validaciones	Para realizar las operaciones: 1. Validar que todas las bases de datos del grupo se encuentren conectadas con privilegios de administración. 2. Validar que el Monitor de Esquema se encontrara instalado en todas las bases de datos.
Postcondiciones	1. Se elimina el grupo de bases de datos de la aplicación.
Prototipo	

Tabla 11. Requisito funcional “Eliminar grupo de base de datos”.

RF 11. **Versionar grupo de bases de datos.**

Propósito	Crea una nueva tarea que permitirá almacenar en el gestor de bases de datos una versión de la estructura de la base de datos maestra de uno de los grupos registrados. Esta versión será para uno de los esquemas que contiene la base de datos.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. Debe haber al menos un grupo de bases de datos registrado.
Descripción	11.1. Seleccionar grupo de bases de datos. 11.1.1. Seleccionar base de datos maestra de uno de los grupos.

	11.2. Seleccionar esquema a versionar. 11.2.1. Mostrar los esquemas disponibles en la base de datos seleccionada. 11.2.2. Seleccionar el esquema. 11.3. Registrar la tarea. 11.3.1. Crear una nueva tarea Versionar Grupo. 11.3.2. Adicionar la tarea a la aplicación. 11.4. Guardar una nueva versión. 11.4.1. Guardar una nueva versión de la base de datos en el Monitor de Esquema de la base de datos maestra seleccionada.
Validaciones	Para realizar las operaciones: 1. Validar que la base de datos maestra se encuentre conectada.
Postcondiciones	1. Queda registrada en la aplicación una nueva tarea Versionar Grupo para su futura ejecución.
Prototipo	**Anexo 5**.

Tabla 12. Requisito funcional “Versionar grupo de bases de datos”.

RF 12. **Extraer versión.**

Propósito	Crea una nueva tarea que permitirá almacenar en el repositorio de la aplicación una de las versiones guardadas en el gestor de bases de datos de la base de datos maestra perteneciente a un grupo.
Roles	Gestor de Configuración de Bases de Datos.
Precondiciones	1. Debe haber al menos un grupo de bases de datos registrado. 2. Debe existir al menos una versión guardada en el servidor de la base datos maestra del grupo seleccionado.
Descripción	12.1. Seleccionar el grupo de bases de datos. 12.1.1. Seleccionar la base de datos maestra de uno de los grupos.

	12.2. Seleccionar la versión a extraer. 12.2.1. Mostrar todas las versiones guardadas en servidor de la base de datos maestra. 12.2.2. Seleccionar una versión. 12.3. Seleccionar los tipos de objetos a extraer. 12.3.1. Mostrar los tipos posibles de objetos a extraer de acuerdo a los especificados por el plugin de acceso a datos de la base de datos seleccionada. 12.3.2. Seleccionar los tipos de objetos. 12.4. Seleccionar los objetos específicos a extraer. 12.4.1. Mostrar los objetos que contiene la versión seleccionada en la base de datos maestra que se corresponden con el tipo especificado en los objetos generales. 12.4.2. Seleccionar los objetos específicos. 12.5. Registrar una nueva tarea. 12.8.8. Crear tarea Extraer versión. 12.8.9. Adicionar tarea a la aplicación. 12.9. Extraer objetos. 12.9.8. Guardar los objetos seleccionados en le repositorio de la aplicación.
Validaciones	Para realizar las operaciones: 1. Validar que la base de datos maestra se encuentre conectada.
Postcondiciones	1. Queda registrada en la aplicación una nueva tarea Extraer Versión para su futura ejecución.
Prototipo	**Anexo 6**.

Tabla 13. Requisito funcional “Extraer versión”.

RF 13. **Sincronizar bases de datos Maestra (en línea) - Esclava.**

Propósito	Crea una nueva tarea que permitirá la generación de un script de cambios que sincronizará a una base de datos esclava con una versión guardada en el gestor de la base de datos maestra de su grupo.
Roles	Gestor de la Configuración de Bases de Datos.
Precondiciones	1. Debe existir al menos un grupo de bases de datos registrado. 2. Debe existir al menos una versión guardada en el gestor de la base de datos maestra del grupo seleccionado.
Descripción	13.1. Seleccionar un grupo de bases de datos. 13.1.1. Seleccionar la base de datos maestra de uno de los grupos. 13.2. Selecciona una versión a comparar. 13.2.1. Mostrar todas las versiones guardadas en servidor de la base de datos maestra. 13.2.2. Seleccionar la versión. 13.3. Seleccionar la base de datos esclava a sincronizar. 13.4. Seleccionar los tipos de objetos generales a comparar. 13.4.1. Mostrar los tipos posibles de objetos a comparar de acuerdo a los especificados por el plugin de acceso a datos de la base de datos origen relacionada con la versión seleccionada. 13.4.2. Seleccionar varios tipos de objetos a comparar. 13.5. Seleccionar los objetos específicos a comparar. 13.5.1. Mostrar los objetos que contiene la versión seleccionada en la base de datos que se corresponden con el tipo especificado en los objetos generales. 13.5.2. Seleccionar varios objetos a comparar. 13.6. Comparar los objetos seleccionados. 13.6.1. Obtener cada objeto seleccionado de la versión origen.

	13.6.2. Obtener cada objeto seleccionado de la base de datos esclava. 13.7. Comparar texto DDL de ambos objetos. 13.8. Generar el script de cambio. 13.8.1. Seleccionar los objetos que resultaron diferentes en la comparación. 13.8.2. Obtener el script de cambio para los objetos diferentes. 13.9. Aplicar el script de cambio en la base de datos esclava.
Validaciones	Para realizar las operaciones: 1. Validar que la base de datos maestra asociada a la versión esté conectada. 2. La base de datos esclava debe estar conectada por el usuario al cual pertenece el esquema de la versión seleccionada.
Postcondiciones	1. La base de datos esclava queda sincronizada con la base de datos maestra de su grupo y por lo tanto con el mismo número de versión.
Prototipo	**Anexo 7**.

Tabla 14. Requisito funcional “Sincronizar bases de datos Maestra (en línea) - Esclava”.

RF 14. **Sincronizar bases de datos Maestra (versión) - Esclava.**

Propósito	Crea una nueva tarea que permitirá la generación de un script de cambios que sincronizará a una base de datos esclava con una versión guardada en el repositorio de la herramienta de la base de datos maestra de su grupo.
Roles	Gestor de la Configuración de Bases de Datos.
Precondiciones	1. Debe existir al menos un grupo de bases de datos registrado. 2. Debe existir al menos una versión guardada en el repositorio de la herramienta de la base de datos maestra del grupo seleccionado.
Descripción	14.1. Seleccionar un grupo de bases de datos. 14.1.1. Seleccionar la base de datos maestra de uno de los grupos. 14.2. Selecciona una versión a comparar.

	14.2.1. Mostrar todas las versiones guardadas en el repositorio de la herramienta de la base de datos maestra. 14.2.2. Seleccionar la versión. 14.3. Seleccionar una base de datos esclava a sincronizar. 14.4. Seleccionar los tipos generales de objetos a comparar. 14.4.1. Mostrar los tipos posibles de objetos a comparar de acuerdo a los contenidos en la versión seleccionada. 14.4.2. Seleccionar varios tipos de objetos a comparar. 14.5. Seleccionar los objetos específicos a comparar. 14.5.1. Mostrar los objetos que contiene la versión seleccionada que se corresponden con el tipo especificado en los objetos generales. 14.5.2. Seleccionar varios objetos a comparar. 14.6. Comparar los objetos seleccionados. 14.6.1. Obtener cada objeto seleccionado de la versión origen. 14.6.2. Obtener cada objeto seleccionado de la base de datos eslava. 14.7. Comparar el texto DDL de ambos objetos. 14.8. Generar el script de cambio. 14.8.1. Seleccionar los objetos que resultaron diferentes en la comparación. 14.8.2. Obtener el script de cambio para los objetos diferentes. 14.9. Aplicar el script de cambio en la base de datos esclava.
Validaciones	Para realizar las operaciones: 1. La base de datos eslava debe estar conectada por el usuario al cual pertenece el esquema de la versión seleccionada.
Postcondiciones	1. La base de datos esclava queda sincronizada con la base de datos maestra de su grupo y por lo tanto con el mismo número de versión.

Prototipo	**Anexo 9**.

Tabla 15. Requisito funcional “Sincronizar bases de datos Maestra (versión) - Esclava”.

2.2.2. Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son fundamentales en el éxito del producto y normalmente están vinculados a los requisitos funcionales, es decir una vez se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener y cuán rápido o grande debe ser.

Los requisitos no funcionales para el sistema a implementar son los siguientes:

Requisitos de Software:

1. La aplicación debe ser soportada en todas las plataformas de Microsoft Windows de (32 bit).
2. La aplicación debe ser soportada en todas las plataformas de Linux.
3. Instalación de la Máquina Virtual de Java.

Requisitos de Hardware:

4. Mínimo 1 GHz de velocidad de microprocesador.
5. Mínimo 512 MB de memoria RAM.
6. Mínimo 50 MB de espacio en el disco duro.

Restricciones en el diseño y la implementación:

7. La arquitectura de la herramienta debe permitir que el acceso a datos sea independiente del negocio de manera que se puedan incorporar plugins con la implementación del acceso a datos para el uso de la herramienta con diferentes gestores de bases de datos.
8. Las interfaces deben ser genéricas y que sean adaptables a las características del gestor con que se esté trabajando.
9. PLSQL para la creación de los script SQL encargados de crear la infraestructura a nivel del servidor de bases de datos en el caso del SGBD Oracle.

Requisitos de Seguridad:

10. Los parámetros de conexión a las bases de datos estarán encriptados a nivel de la aplicación.
11. La conexión a las bases de datos se realizará de forma segura partiendo de que la conexión se realice a través de una VPN por hardware o software, dependiendo de la arquitectura planteada en la distribución de los servidores de bases de datos.
12. Los script que son enviados a las bases de datos remotas son encriptados para prevenir inyecciones SQL.

2.3. Roles de Scrum para el sistema propuesto.

Para la realización del sistema propuesto se analizó la estructura de roles que propone la metodología Scrum y se adaptaron estas propuestas a las condiciones reales del sistema y el equipo de desarrollo. Para la realización de las tareas relacionadas con la construcción de la herramienta se propone la siguiente estructura de roles:

Roles de Scrum	Integrante del equipo de desarrollo.
Scrum Owner	Abdiel Carrazana Saucedo
Scrum Master	Rafael Ambruster Crespo
Scrum Team	Abdiel Carrazana Saucedo, Rafael Ambruster Crespo
Usuarios	Gestores de Configuración de Bases de Datos del Centro de Identificación y Seguridad Digital.

Tabla 16. Roles de Scrum.

2.4. Product Backlog.

El Product Backlog es la relación de las funcionalidades, mejoras, tecnologías y corrección de errores que deben incorporarse al producto a través de las sucesivas iteraciones del desarrollo. Representa todo aquello que esperan los clientes, usuarios, y en general todos los interesados en el producto. Todo lo que suponga un trabajo que debe realizar el equipo tiene que estar reflejado en el Product Backlog. A

continuación se presenta el Product Backlog para el cumplimiento de los requisitos funcionales de la herramienta a desarrollar. Tras un acuerdo con el cliente se llegó a la conclusión de dar prioridad máxima (1) a las tareas de más importancia en la aplicación, así como prioridad mínima (3) a las tareas de menos interés.

Identificador	Tarea	Prioridad	Estado	Estimación	Real	Responsable	Sprint
1	Definir las funcionalidades del sistema.	1	Concluido	7 días	7 días	Abdiel Carrazana	1
2	Definir el prototipo de interfaz de usuarios.	1	Concluido	7 días	7 días	Abdiel Carrazana	1
3	Diseñar la arquitectura del sistema.	1	Concluido	14 días	12 días	Abdiel Carrazana	1
4	Diseñar la arquitectura de un plugin de acceso a datos.	1	Concluido	14 días	14días	Rafael Ambruster	1
5	Diseño del Monitor de Esquema.	1	Concluido	14 días	12 días	Rafael Ambruster	1
6	Implementación de las clases del negocio.	1	Concluido	14 días	14 días	Abdiel Carrazana	2
7	Implementación de las interfaces	1	Concluido	7 días	7 días	Abdiel Carrazana	2

	visuales principales.						
8	Implementación de excepciones.	1	Concluido	7 días	7 días	Abdiel Carrazana	2
9	Implementación del Monitor de Esquema.	1	Concluido	14 días	14 días	Rafael Ambruster	2
10	Implementar conexión del plugin para Oracle.	1	Concluido	14 días	14 días	Rafael Ambruster	2
11	Implementar carga y salva de las variables del sistema.	2	Concluido	7 días	5 días	Abdiel Carrazana	3
12	Implementar carga de plugins.	2	Concluido	7 días	7 días	Abdiel Carrazana	3
13	Implementar gestión de bases de datos.	2	Concluido	14 días	14 días	Abdiel Carrazana	3
14	Implementar las clases del esquema del plugin de acceso a datos para Oracle.	2	Concluido	14 días	10 días	Rafael Ambruster	3
15	Implementar las tareas del plugin de acceso a datos para	2	Concluido	14 días	14 días	Rafael Ambruster	3

	Oracle.						
16	Implementar gestionar grupos de bases de datos.	2	Concluido	7 días	7 días	Rafael Ambruster	4
17	Refinamiento del análisis de las tareas.	2	Concluido	7 días	6 días	Abdiel Carrazana	4
18	Implementar versionar bases de datos.	2	Concluido	14 días	10 días	Rafael Ambruster	4
19	Implementar extraer una versión de bases de datos.	2	Concluido	21 días	21 días	Abdiel Carrazana	4
20	Hacer pruebas a las funcionalidades.	2	Concluido	7 días	7 días	Rafael Ambruster	4
21	Implementar Comparar Online-Online	3	Concluido	14 días	14 días	Rafael Ambruster	5
22	Implementar Comparar Versión-Online	3	Concluido	14 días	14 días	Abdiel Carrazana	5
23	Implementar Sincronización de una base de datos.	3	Concluido	7 días	7 días	Rafael Ambruster	5
24	Realizar pruebas	3	Concluido	14 días	13 días	Abdiel	5

	a las funcionalidades.					Carrazana	
25	Refinar interfaces visuales	3	Concluido	7 días	7 días	Rafael Ambruster	5

Tabla 17. Product Backlog.

2.5. Sprint Backlog.

El Sprint Backlog define un subconjunto de trabajos o tareas, que el equipo selecciona del Product Backlog, para realizarlas en un sprint y convertirlas en un incremento de funcionalidad entregable del producto. Solo el equipo puede cambiar el Sprint Backlog. Para la realización de la herramienta se propone la realización de cinco sprints los cuales son expuestos a continuación.

Sprint # 1: (**Anexo 9**, **Anexo 10**).

Sprint # 2: (**Anexo 11**, **Anexo 12**).

Sprint # 3: (**Anexo 13**, **Anexo 14**).

Sprint # 4: (**Anexo 15**, **Anexo 16**).

Sprint # 5: (**Anexo 17**, **Anexo 18**).

Conclusiones

En este capítulo se realizó una descripción de la solución propuesta. Se llegaron a conclusiones sobre las funcionalidades a implementar, la seguridad y las principales características de la herramienta y posteriormente se realizaron algunos artefactos propuestos por la metodología Scrum. Se definieron las tareas por iteración y prioridad, así como una estimación del tiempo de desarrollo de las mismas.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN

En este capítulo se abordan los principales conceptos relacionados con el diseño e implementación de la herramienta. En los siguientes epígrafes se describe la arquitectura general del sistema y sus componentes. Además se presenta el Modelo de Datos usado para efectuar el Control de Cambios desde el propio gestor, así como la distribución física de cada una de las partes que componen a la aplicación.

3.1. DB Evolution en la Gestión de Configuración en bases de datos. Aportes.

La herramienta DB Evolution está diseñada con el objetivo de servir como apoyo fundamental para realizar el proceso de Gestión de Configuración en los Sistemas de Bases de Datos. La misma debe por tanto realizar los procesos claves que caracterizan a la Gestión de Configuración de una forma eficiente y amigable para el usuario.

DB Evolution puede mantener el control de un conjunto de bases de datos que son registradas en la aplicación y a partir de ahí realizar las diferentes acciones sobre ellas. Entre los procesos básicos que ofrece están la extracción del esquema de la base de datos, la comparación con otros esquemas, así como la generación de un script de cambio para la sincronización de una o varias bases de datos. Una novedosa forma de controlar los cambios es hacerlo desde el propio gestor de base de datos. Los gestores más modernos y potentes ofrecen disparadores DDL para auditar todos los cambios del esquema. Esta característica puede ser utilizada para realizar tareas administrativas en la base de datos tales como la auditoría y la regulación de las operaciones de la base de datos.

DB Evolution está diseñada para invocar los cambios registrados previamente en un gestor de bases de datos y trabajar en base a estos. Está organizada en servidores maestros y servidores esclavos previamente elegidos por el Gestor de Configuración de Bases de Datos. Los servidores maestros son aquellos en los cuales se registran los cambios y los servidores esclavos son a los que se aplican dichos cambios. En un ambiente real la distribución sería un servidor de desarrollo en el cual los administradores de bases de datos prueban y guardan los cambios en versiones de las bases de datos y los servidores reales son aquellos a los que se le aplican los cambios previamente probados. En el epígrafe 3.6 se hace una descripción más detallada del proceso mediante el cual se logra una correcta coordinación entre las bases de datos maestras y las esclavas. La herramienta también cuenta con un repositorio en el que se

almacenan las diferentes versiones por las que pasan los servidores maestros, el cual sirve como base para posibles revisiones y auditorías.

La realización de estos procesos en la base de datos, al ser un proceso invasivo a la misma, requiere de la autorización explícita del Gestor de Configuración de Bases de Datos, así como del acceso a una conexión con privilegios administrativos. Igualmente será posible retirar todas las estructuras de este esquema de la base de datos en el momento que se estime conveniente.

Con la utilización de la herramienta se provee una interfaz común para la realización del proceso de Gestión de Configuración en los Sistemas de Bases de Datos al realizar todas las tareas necesarias desde el mismo entorno de trabajo. Esto es posible gracias a que la cantidad de plugins que puede admitir la herramienta son tantos como sean posibles desarrollar por los programadores. Esto elimina el inconveniente de usar herramientas distintas para cada tipo de gestor, así como para cada tarea a realizar.

La implementación del Control de Cambios desde el propio gestor de bases de datos es un concepto nuevo para herramientas de este tipo, ya que es posible gracias a la aparición reciente de los disparadores DDL. Estos nuevos tipos de disparadores permiten mantener un control sobre las modificaciones en las bases de datos que es imposible para las herramientas existentes hasta el momento, ya que permite hacerlo en el mismo momento que se produce y almacenar datos importantes como son la fecha y hora en que se produce, el usuario que lo realiza, así como el texto DDL de la modificación. Estas características son muy importantes a la hora de realizar auditorías y reportes sobre la base de datos.

Otro de los aportes de la herramienta es la creación de los grupos de bases de datos y la división en bases de datos maestras y esclavas. Estos conceptos son característicos de herramientas de réplica que solo se encargan de la replicación de los cambios de una base de datos a otra, pero es nuevo en herramientas de Control de Cambios y Control de Versiones como DB Evolution. Estos grupos mantienen una mayor organización de las bases de datos, pudiendo establecer dependencias entre ellas y hacer posible controlar un gran número de instancias por una sola persona.

Otra de las ventajas de la nueva herramienta se produce como resultado de realizar el Control de Cambios desde el gestor y la división en los grupos ya mencionados. Estas dos características permiten un nuevo enfoque en las tareas de comparación y sincronización entre bases de datos pudiéndose crear

estructuras que almacenen los hash de los objetos que cambiaron y no sea necesario extraer los textos de cada objeto, que muchas veces son extensos, sino enfocar todos los recursos en los que están notificados como cambios o los que muestran diferencias en los hash. Nótese que es mucho más rápido y eficiente comparar el hash de dos objetos que no todo su texto DDL y en caso de que resulten iguales solo es necesario extraer el texto del objeto de una sola base de datos. Estas y otras características necesitaban de un diseño y una arquitectura que pudiera realizar todos los procesos que se necesitaban de una forma eficiente y con el mayor aprovechamiento posible de los recursos.

3.2. Arquitectura base.

DB Evolution cuenta con una arquitectura flexible y consistente, basándose en el uso de patrones de diseño para la construcción de sistemas. De forma general la herramienta está compuesta por un módulo de clases e interfaces pertenecientes a la aplicación principal, y varias librerías de clases, cada una de las cuales implementan el acceso a datos para una o varias versiones de un determinado gestor de bases de datos. Estas librerías pueden ser importadas en tiempo de ejecución y su estructura está determinada en un paquete de interfaces que sirven de intermediarios entre el negocio y el acceso a datos de la aplicación. De forma general el sistema se divide en tres capas individuales las cuales serán explicadas a continuación.

3.2.1. Separación lógica en capas.

El modelo de la arquitectura está definido en tres capas independientes que facilitan el diseño modular (cada capa encapsula un aspecto concreto del sistema) y permite la construcción de sistemas débilmente acoplados (si minimizamos las dependencias entre capas, resultará más fácil sustituir la implementación de una capa sin afectar al resto del sistema). (Berzal Galiano). **Figura 2** .De esta forma cada capa sólo puede comunicarse con la capa inmediata inferior y nunca con las superiores .Esta separación además permite avanzar en la programación de una forma ordenada al ser dividida la aplicación general en varios módulos y capas que pueden ser tratados de manera independiente y hasta en forma paralela.

3.2.2. Capa de Presentación.

Es la capa que interactúa directamente con el usuario y contiene las interfaces visuales de la aplicación. Estas interfaces registran los datos insertados por el Gestor de Configuración de Bases de Datos y en ella

se hacen la mayoría de las validaciones de entrada, a excepción de aquellas que tienen que ver con un gestor de bases de datos específico. Contiene todos los objetos visuales y los componentes que se muestran a los usuarios, tales como exploradores, tablas, paneles, ventanas, etc. Esta capa solo tiene acceso a la capa de Negocio y ninguna otra capa tiene acceso a ella.

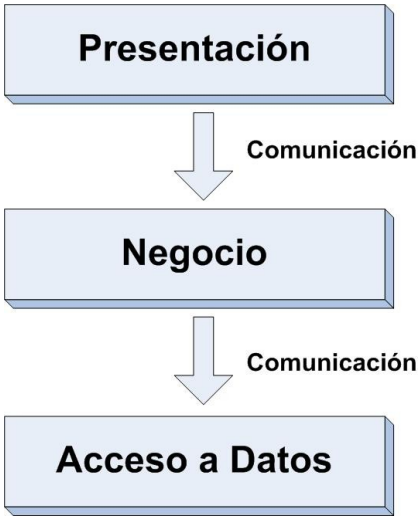

Figura 2. Capas de la aplicación.

3.2.3. Capa de Negocio.

En esta capa se desarrolla la lógica de toda la aplicación. Implementa todos los procesos del negocio identificados en los requisitos funcionales, ofrece los servicios que necesita la capa de presentación para interactuar con el usuario y además es la capa que invoca los métodos de persistencia para cada uno de los gestores disponibles.

3.2.4. Capa de Acceso a Datos.

Esta capa contiene los datos necesarios para extraer e insertar información en cada una de las bases de datos registradas por la herramienta. En realidad se materializa en librerías de clases cuya estructura está regida por un conjunto de interfaces que permite la comunicación entre el negocio y la capa en cuestión. Sus componentes tienen la capacidad de proveer métodos de acceso a las bases de datos para una o varias versiones de un gestor de bases de datos específico, por lo que la herramienta puede soportar tantos gestores como librerías tenga importadas. Es importante destacar que en esta capa no está

definida la implementación de ningún proceso de los que ejecuta la herramienta, sólo contiene funciones y objetos que los procesos necesitan para ejecutarse.

3.2.5. Vista vertical de la arquitectura.

En la vista vertical de la arquitectura se hace un análisis más detallado de cada uno de los módulos que componen a la aplicación. **Figura 3**. En el módulo Vista están contenidos todos los componentes visuales que contiene la aplicación y con los cuales interactúa directamente el usuario. Las acciones que pueden realizar estos componentes fueron separados a un módulo Acciones, que permite la modularidad y reutilización del código al permitir que las mismas acciones sean ejecutadas por varios componentes.

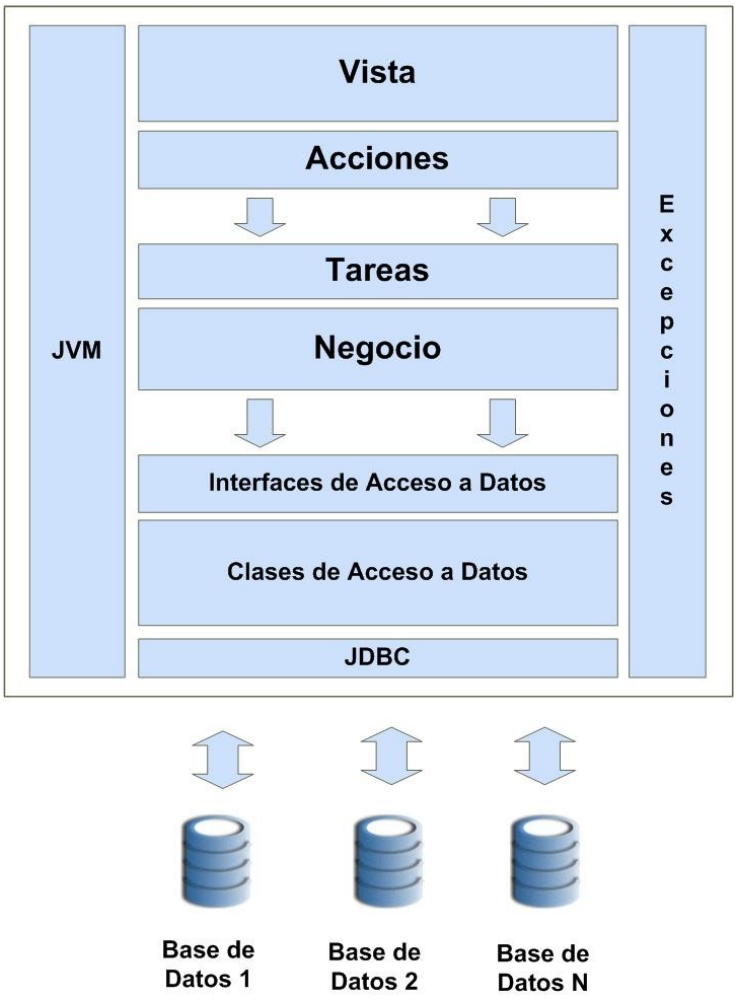

Figura 3. Vista vertical de la arquitectura.

Estas acciones hacen las llamadas al módulo Tareas, el cual contiene los procesos que puede ejecutar la aplicación y que fueron especificados en los requisitos funcionales de la herramienta. Estas tareas hacen uso de las clases del negocio para desarrollarse, en las cuales están contenidas las dependencias y reglas del sistema. Es válido aclarar que estas tareas son independientes del gestor de bases de datos que esté implicado en cada una de ellas, lo cual se logra gracias a un módulo de interfaces que regula el comportamiento de la capa de acceso a datos y por tanto la comunicación con ella.

Las clases de acceso a datos implementan estas interfaces y definen de forma particular el acceso a los datos en dependencia del gestor que se trate. Estas clases están compuestas por funciones que pueden realizar una acción determinada para un gestor específico, las cuales pueden ser usados en los diferentes procesos del negocio y llevar a cabo una funcionalidad de la herramienta en sí. Para acceder a las bases de datos se usan las librerías del estándar JDBC de la plataforma Java. Estas librerías son ofrecidas de forma gratis por cada una de las compañías a las que pertenecen cada uno de los gestores y pueden ser fácilmente adicionadas a la aplicación. De forma vertical y asociado a cada uno de los módulos descritos anteriormente está el módulo de las excepciones, el cual garantiza de forma dinámica la incorporación de nuevos tipos de errores, así como un tratamiento específico para cada uno. La Máquina Virtual de Java (JVM) es la encargada de traducir el código para todo el sistema y convertirlo en acciones

3.3. Patrones de diseño.

Un patrón de diseño es una buena práctica documentada o el núcleo de una solución que ha sido aplicada de forma exitosa en múltiples entornos para resolver un problema que puede repetirse en un conjunto específico de situaciones. (Kuchana, 2004). De forma general describen soluciones elegantes a problemas específicos que se repiten en muchas aplicaciones diferentes, aplicando técnicas para flexibilizar el código haciéndolo satisfacer ciertos criterios.

En el diseño de la herramienta DB Evolution se hizo uso de una gran cantidad de estos patrones. Algunos de los principales que fueron utilizados son descritos a continuación:

- Instancia única (Singleton): Este patrón es usado cuando sólo es requerida una instancia de un objeto durante el tiempo en que se está ejecutando la aplicación, ya sea porque solo una es suficiente o para evitar conflictos. El mismo fue usado en varios escenarios, por ejemplo para asegurar que solo haya

una instancia de los distintos exploradores que se muestran al usuario, como son el explorador de bases de datos, el explorador de tareas y el de control de versiones.

- Fábrica (Factory): El patrón Fábrica no es más que encargar a un método específico de la creación de un tipo determinado de objeto en dependencia de una condición dada, así como una fábrica construye distintos tipos de productos. El empleo del mismo se muestra a la hora de construir las interfaces visuales para insertar los parámetros de conexión a la hora de registrar una nueva base de datos. Teniendo en cuenta que la aplicación registrará bases de datos de diferentes gestores con diferentes tipos y cantidades de parámetros, se hace necesario encargar a un método factory () de la creación de los componentes visuales para obtener dichos parámetros. En este caso la condición que determina cuál componente crear es el tipo del parámetro de conexión que necesita el gestor.
- Observador (Observer): El patrón Observador permite monitorizar el comportamiento de un objeto observado por uno o varios observadores que estarán pendientes de los cambios del mismo. Los lenguajes de programación modernos como Java hacen de este patrón una poderosa herramienta al poder suscribir varios objetos a los eventos de otro y ejecutar acciones en consecuencia. Este patrón es ampliamente usado para mantener actualizadas las interfaces visuales que muestran el estado de los objetos al Gestor de Configuración de Bases de Datos, permitiendo hacer cambios internos a los mismos y que automáticamente dichos cambios sean detectados por las vistas y actualizadas en consecuencia.
- Mixto (Composite): El patrón Mixto consiste en agrupar dos o más objetos para ser tratados de forma uniforme. Este es un patrón muy utilizado en la programación orientada a objetos ya que permite descartar las diferencias entre objetos y tratarlos solo por las características en común. La forma más simple de representarlo es mediante el uso de las interfaces para una colección de objetos. Este patrón permite a la aplicación tratar a cada plugin importado de una forma uniforme, ya que las interfaces que implementan cada uno de ellos aseguran la similitud en su estructura y comportamiento.

3.4. Diagramas de componentes.

Un diagrama de componentes ilustra los fragmentos de software, controladores embebidos, que conformarán un sistema. Un diagrama de componentes tiene un nivel de abstracción más elevado que un diagrama de clases. Usualmente un componente se implementa por una o más clases (u objetos) en

tiempo de ejecución. Estos son bloques de construcción, como así eventualmente un componente puede comprender una gran porción de un sistema.

En la **Figura 4** se muestra el diagrama de componentes para la aplicación principal. De forma general se muestra la separación entre los componentes visuales y los del negocio, así como las acciones y trabajos a realizar. Esta modulación permite una mayor claridad dado la gran complejidad del sistema, así como promueve la reutilización de código. La comunicación con el plugin ocurre mediante la interfaz IPlugin representada en el diagrama.

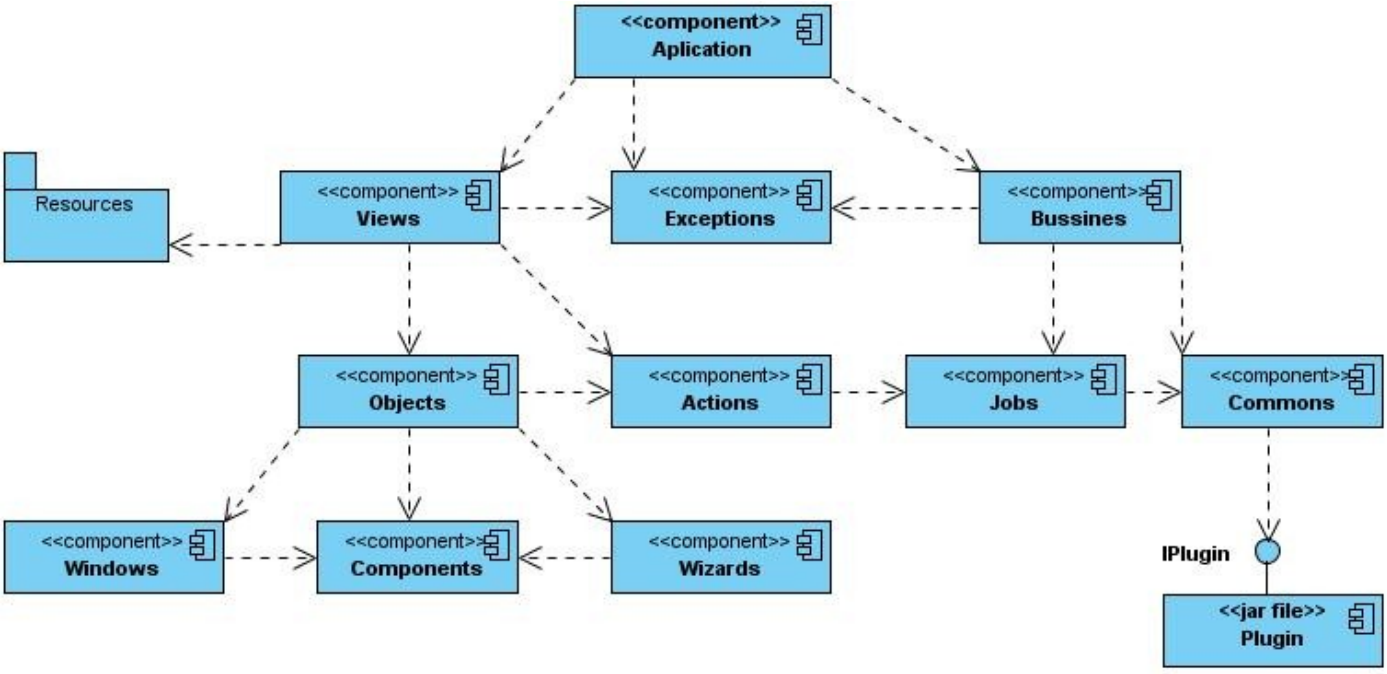

Figura 4. Diagrama de Componentes de la aplicación principal.

La **Figura 5** representa el diagrama de componentes para un plugin. Son claramente visibles las interfaces que deben implementar los componentes y que por lo tanto establecen las reglas y el comportamiento de cada uno.

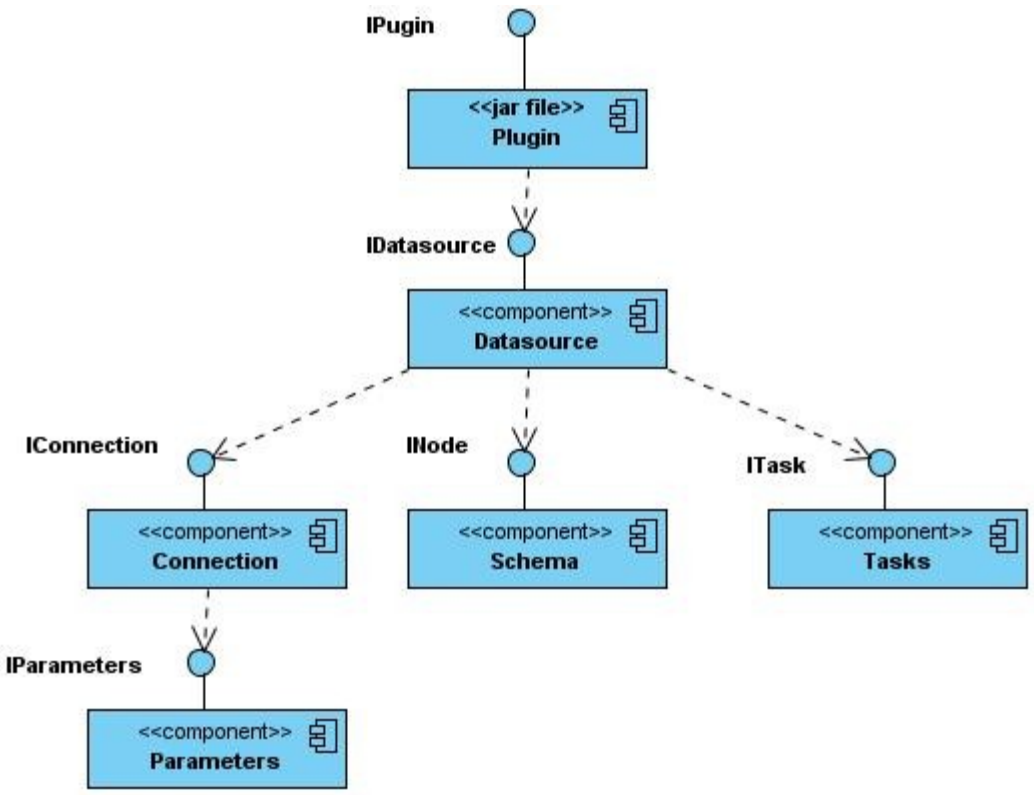

Figura 5. Diagrama de Componentes de un plugin.

3.5. Diagramas de clases.

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases generalmente son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro. En el **Anexo 20** se presentan los diagramas de clases para el plugin desarrollado correspondiente al SGBD Oracle, los cuales fueron generados utilizando técnicas de ingeniería inversa.

3.6. Control de Cambios desde el gestor de bases de datos.

Una de las mayores potencialidades de la herramienta DB Evolution es la capacidad de realizar el Control de Cambios usando al propio gestor de bases de datos para ello. Este proceso se describe a continuación.

Existe un servidor maestro y varios servidores esclavos los cuales conforman un grupo de bases de datos. El servidor maestro representa la línea base por la cual se guiarán el resto de los servidores del grupo. En esta línea base comienzan a registrarse versiones para los objetos cada vez que sufren cambios. Los objetos van pasando por diferentes estados de versionado que son considerados no estables, hasta que el Gestor de Configuración de Bases de Datos determina marcar un nuevo punto en el análisis de la base de datos y es cuando la base de datos en conjunto pasa a una versión superior que se considera estable. **Figura 6**. De esta forma queda almacenada en el gestor una traza de las diferentes versiones estables por las que ha pasado la base de datos, pudiéndose almacenar cada una en el repositorio de la herramienta y ser posible incluso regresar la base de datos a una versión anterior.

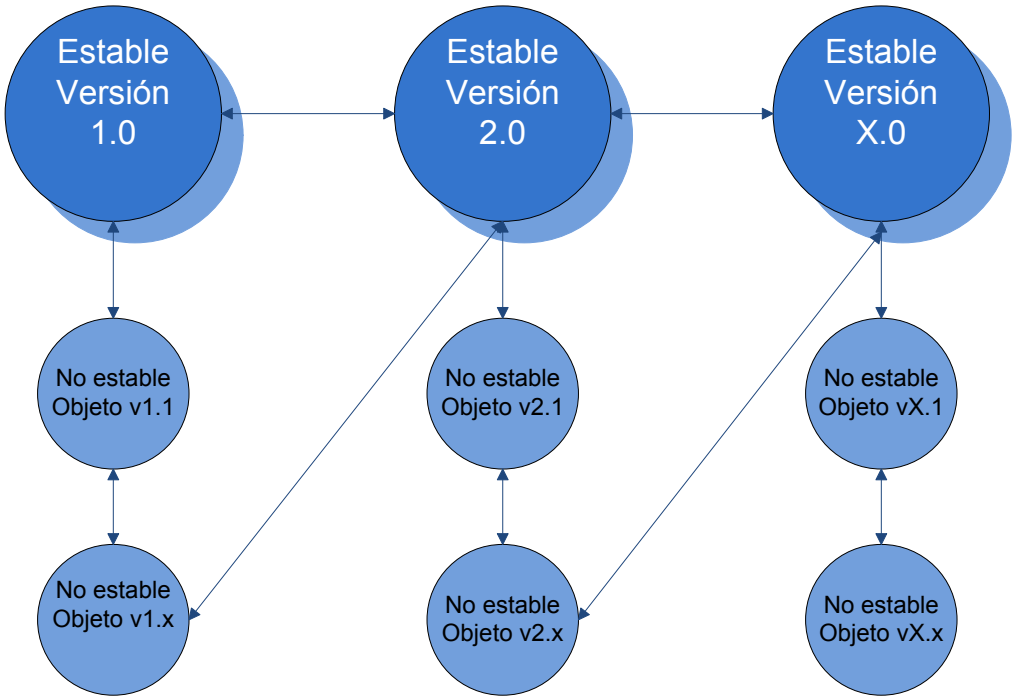

Figura 6. Sistema de versionado de DB Evolution.

A nivel de gestor de bases de datos, la versión contiene el texto DDL de todos los objetos existentes, la fecha de la última modificación y el hash del objeto que es calculado usando el algoritmo MD5. Esta

actualización se lleva a cabo registrando el último cambio que se produjo sobre un objeto, que puede ser la modificación de un procedimiento almacenado, un paquete, una función, etc. Para comodidad y seguridad, DB Evolution permite crear un esquema en el gestor de la base de datos, en el cual estarán los objetos que usa. Tanto en el servidor maestro como en el esclavo la comparación de las estructuras de los objetos utiliza el hash de estos para establecer las diferencias. La aplicación toma los hash de los objetos que están notificados en el servidor maestro como que fueron modificados o creados, extrae dichos objetos en el servidor esclavo y realiza la comparación verificando que exista similitud. En caso de existir diferencias procede a notificar el cambio al Gestor de Configuración de Bases de Datos mostrando las diferencias .El proceso que se efectúa en el servidor se describe en la **Figura 7**.

El proceso comienza con la modificación del lenguaje de definición de datos o DDL desde un cliente de bases de datos. En el servidor existe un disparador DDL encargado de capturar los cambios realizados en la estructura de los objetos. Este inserta estas acciones en una tabla filtro (AUDIT_DDL) en la cual existe un disparador DML (INSAUDITL_DDL) encargado de tomar solo los cambios que le son necesarios al sistema para su correcto funcionamiento, ya que en la tabla filtro se insertan todos los cambios a nivel de gestor y muchos de estos cambios no son necesario procesar. Una vez filtrado los cambios, estos son insertados en una tabla de objetos (OBJSTRUCTURE) y son notificados actualizando el campo (AMENDED) si la acción es ALTER o CREATE, y el campo DELETED en caso de que la acción sea DROP.

Una vez estando el objeto notificado según la acción que se realizó sobre él, existe una tarea programada o (JOB) llamado (PROCESS_CRTL_VERSIONS) que se encarga procesarlo. El proceso consiste en ejecutar paralelamente el cálculo del hash a la estructura del objeto usando MD5 como algoritmo e insertándolo, o actualizándolo si ya existe en la tabla de hash (COMPAREHASH) y procesar el objeto para ser versionado en la tabla de versiones (VERSIONOFSCHEMA) usando los paquetes (PKG_CTRL_VERSION_FUNCTIONS) y (PKG_CTRL_VERSION_PROCESS) en los cuales están las funciones y procedimientos almacenados que describen el proceso.

El proceso de versionado consiste en guardar las versiones por las que ha ido pasando la estructura de cada objeto en líneas bases y versiones de estas. Teniéndose así una traza de líneas bases, para así llevar a cabo el control de las versiones por los que ha pasado la base de datos.

El proceso de Control de Cambios consiste en ir actualizando el hash de cada uno de los objetos e ir notificándolos, de esta forma se tiene un control centralizado de cada una de las acciones sobre la base de datos y es posible conocer que objetos sincronizar hacia otros servidores de bases de datos.

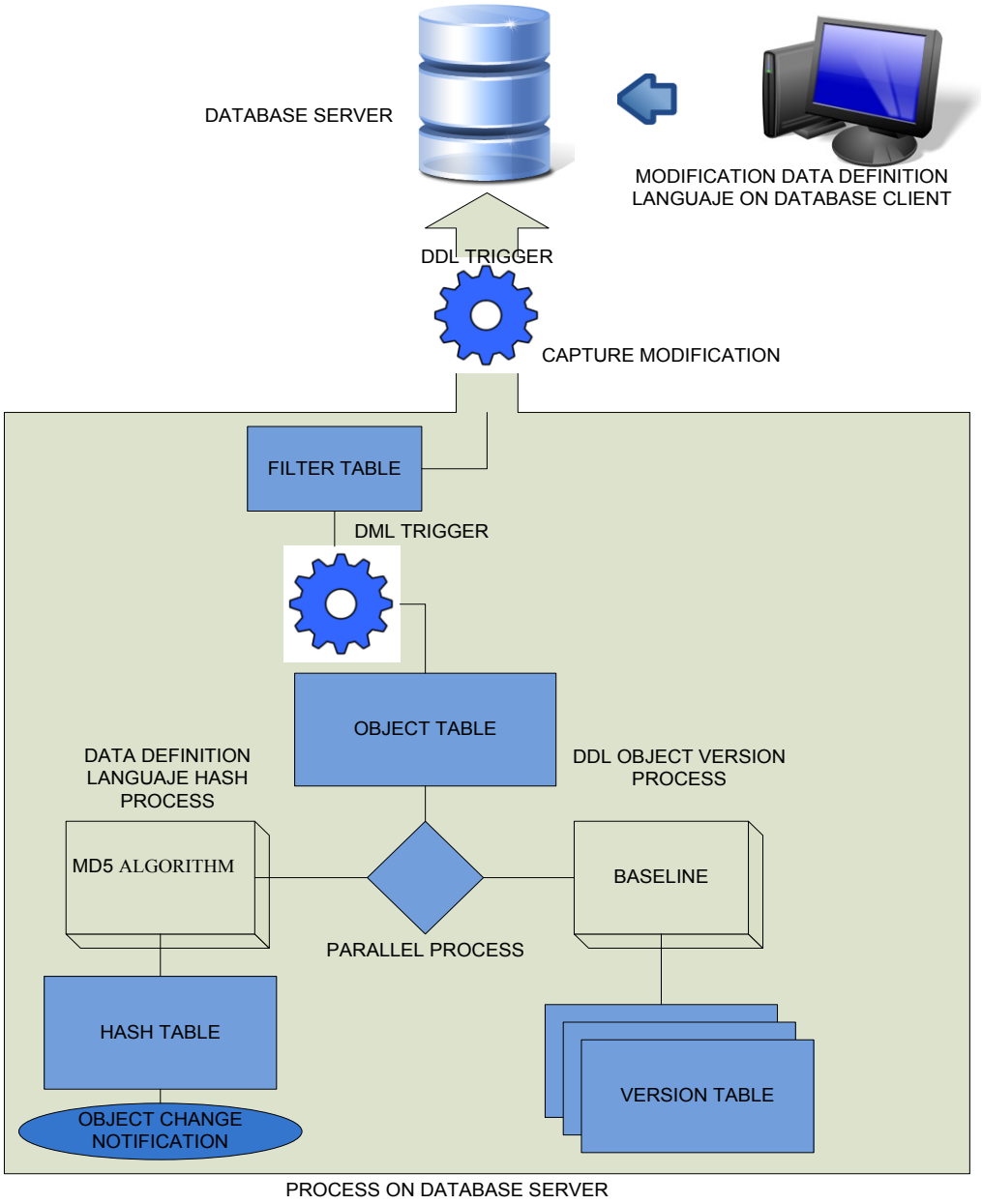

Figura 7. Control de Cambios desde el gestor de bases de datos.

La **Figura 8** muestra el Modelo de Datos del esquema de Control de Cambios desde el gestor para el SGBD Oracle.

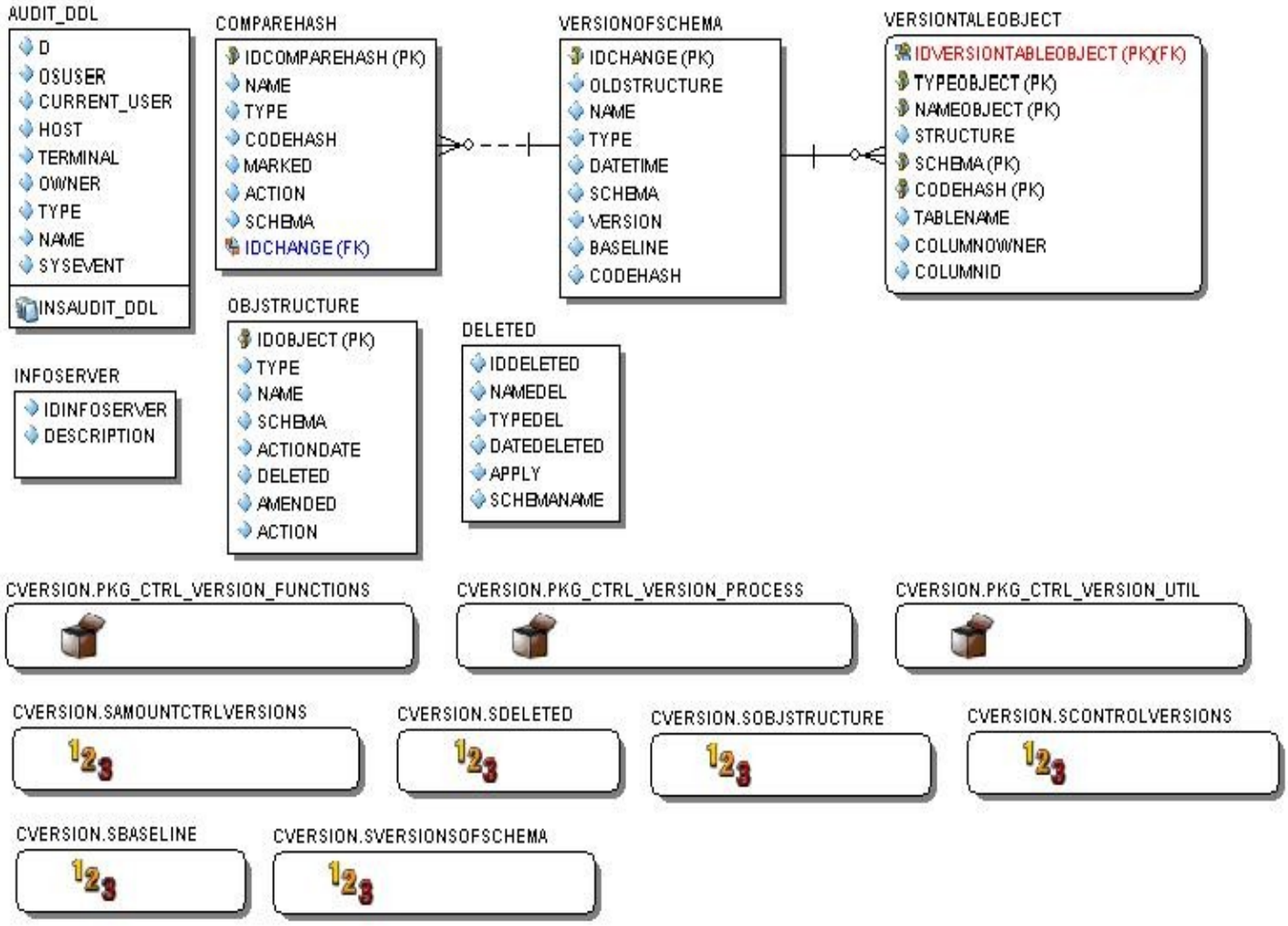

Figura 8. Modelo de Datos.

3.7. Diagrama de Despliegue.

Los diagramas de despliegues describen la arquitectura física del sistema durante la ejecución en términos de procesadores, dispositivos y componentes de software. La **Figura 9** muestra la distribución física de la herramienta, la cual debe estar instalada en un servidor desde el cual será operada por los Gestores de Configuración de Bases de Datos. Las bases de datos deben estar en servidores remotos

desde los cuales la aplicación tendrá acceso mediante el protocolo JDBC que implementa la plataforma Java.

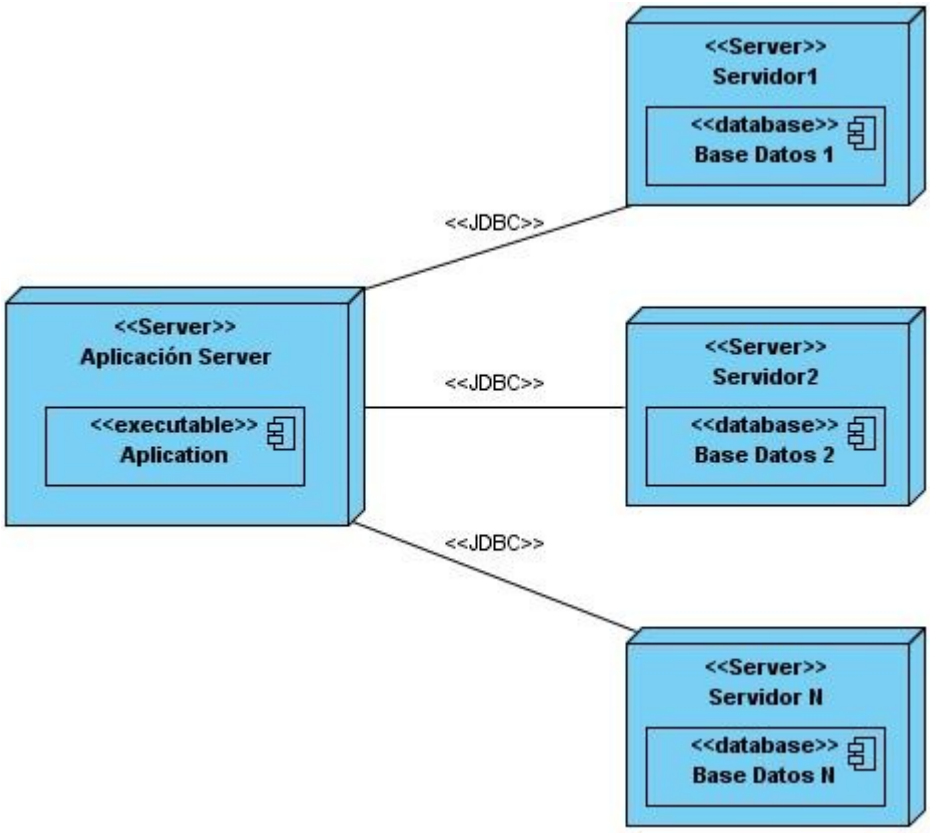

Figura 9. Diagrama de Despliegue.

Conclusiones

En este capítulo se hizo una amplia descripción de los aspectos concernientes al diseño e implementación de la herramienta DB Evolution. Su arquitectura está diseñada para poder importar en tiempo de ejecución plugins que le permitan el acceso a los datos de diferentes gestores de bases de datos, logrando la extensibilidad de la herramienta. Fueron implementadas las funcionalidades que permitirán a los Gestores de Configuración de Bases de Datos mantener un seguimiento a las modificaciones que se realicen sobre las bases de datos, controlar estos cambios, así como la obtención de cada una de las versiones que la conforman.

CAPÍTULO 4. PRUEBAS

El objetivo fundamental que se persigue con la realización de las pruebas es verificar la correcta ejecución del software, teniendo en cuenta un conjunto de casos de pruebas, con la intención de detectar fallas o errores que puedan existir, para dar al cliente un mayor nivel de confiabilidad en el sistema desarrollado. En este capítulo se describen las diferentes pruebas a las que fue sometida la herramienta, entre ellas las de unidad y del sistema. Además se realizó un estudio de la factibilidad del software con el objetivo de calcular diferentes datos de importancia para el cliente.

4.1. Pruebas de unidad

Nivel de prueba: Pruebas de unidad.

Método de prueba: Caja blanca.

Las pruebas de unidad testean el comportamiento de las unidades dentro del código de la aplicación. El tamaño de una unidad es definido por el desarrollador, que es quien realiza este tipo de pruebas, y pueden ser tan pequeñas como un método de una clase o también pudieran agrupar varias clases que de alguna manera se puedan probar en conjunto. Para realizar las mismas por tanto se debe contar con el código terminado de una unidad completa, el cual no debería sufrir cambios posteriores. En la actualidad muchas de estas pruebas es posible automatizarlas mediante herramientas y librerías dedicadas para estas funciones como es el caso de JUnit para el desarrollo con Java.

JUnit es una librería de código abierto adecuada para el desarrollo dirigido por las pruebas, la cual es utilizada para la realización de pruebas unitarias sobre código Java. Estas pruebas se hacen de manera controlada sobre los distintos métodos que conforman una clase, haciendo una comparación entre el valor esperado y el valor que realmente retorna el método. Las pruebas se realizan de manera automática y JUnit se encarga de crear los casos y clases de prueba. JUnit no es más que uno de varios marcos de trabajos (frameworks) conocidos en su conjunto por XUnit, los cuales son utilizados para hacer pruebas de unidad en varios lenguajes de programación, por ejemplo PUnit para PHP o NUnit para .Net.

Durante el desarrollo de la herramienta DB Evolution se fueron ejecutando diferentes pruebas de unidad en paralelo con el desarrollo de las clases. Al ser tan numerosas las funciones implementadas y al mismo tiempo las pruebas de unidad realizadas, sólo se presentan algunas de las más representativas e

importantes para los desarrolladores, como son algunas de las ejecutadas al componente de tareas de la aplicación, el cual encierra las funcionalidades principales de la herramienta. En los casos de fallas en las mismas se analizaron las posibles causas, y no se prosiguió con el resto de las pruebas hasta tanto no se hubiera subsanado el error y completado satisfactoriamente dicha prueba.

Cargar y salvar la configuración.

En este método se prueba el correcto funcionamiento de la salva y carga de los elementos del explorador de bases de datos de la aplicación. De forma general los pasos a seguir fueron los siguientes: se crea una nueva instancia de la aplicación principal que contiene al explorador, se crea un nuevo explorador y se le asigna a la aplicación principal. Luego se salva la configuración completa de la aplicación. Al crear una nueva instancia de la clase principal y cargar la configuración guardada se pudo comprobar que el nodo raíz del explorador creado al inicio coincidía con el cargado por la segunda aplicación con lo cual se da por vencida la prueba para el sistema.

```
@Test
    public void testSaveLoadConfiguration() {
        System.out.println("saveLoadConfiguration");
        DB_Evolution evolution = new DB_Evolution();
        Explorer explorer = Explorer.getExplorer();
        evolution.getPanel_explorer().setExplorer(explorer);
        DefaultMutableTreeNode root = evolution.getPanel_explorer().getExplorer().getRoot();
        ConfigurationAction.saveConfiguration(evolution);
        DB_Evolution new_evolution = new DB_Evolution();
        ConfigurationAction.loadConfiguration(new_evolution);
        DefaultMutableTreeNode new_root = new_evolution.getPanel_explorer().getExplorer().getRoot();
        assertEquals(root, new_root);
    }
```

Importación de los plugins.

Con este método se probó el correcto funcionamiento de la carga en tiempo de ejecución de la librería para el gestor Oracle, el cual contenía dos plugins para la interacción con este gestor.

```
public void testLoadPlugins() {
    System.out.println("LoadPlugins");
    DB_Evolution evolution = new DB_Evolution();
    PanelConsole console = evolution.getConsole();
    String gestor = "Oracle";
    IPlugin[] result = PluginAction.LoadPlugins(console);
    assertEquals(gestor, result[0].getInformation().getGestor());
    assertEquals(gestor, result[1].getInformation().getGestor());
}
```

Conexión y desconexión a una base de datos.

Con esta prueba se probó el correcto funcionamiento de la conexión y desconexión a una base de datos. De forma general los pasos que se siguieron fueron los siguientes: se creó una nueva instancia de la aplicación principal, se cargaron los plugins disponibles, se creó un nuevo objeto Database y se configuraron sus parámetros de conexión para permitir la conexión con una base de datos real. Después se procedió a la conexión y desconexión de la misma. Tras el resultado satisfactorio de esta prueba se dio por vencida la misma por la aplicación.

```
@Test
public void testConnectDisconnectDatabase() throws Exception {
    DB_Evolution evolution = new DB_Evolution();
    PanelConsole console = evolution.getConsole();
    IPlugin[] plugins = PluginAction.LoadPlugins(console);
    Database database = new Database(plugins[0].getDatasource());
    IParameter[] parameters = database.getDatasource().getConnection().getParameters().getParameters();
```

```
    IParameter host = parameters[0];
    host.setValue("master.uci.cu");
    IParameter user = parameters[1];
    user.setValue("user_db");
    IParameter password = parameters[2];
    password.setValue("password_db");
    IParameter logon = parameters[3];
    logon.setValue(LogonMode.normal);
    System.out.println("ConnectDatabase");
    DatabaseActions.ConnectDatabase(evolution, database);
    System.out.println("DisconnectDatabase");
    DatabaseActions.DisconnectDatabase(evolution, database);
  }
```

Instalación y desinstalación del Monitor de Esquema.

Esta prueba verifica el correcto funcionamiento de la funcionalidad instalar y desinstalar el Monitor de Esquema para un grupo de bases de datos. De forma general el procedimiento fue el siguiente: se crea una nueva aplicación principal, se cargan los plugins disponibles, se crean dos objetos Database que representen a dos bases de datos reales que actuaran como maestra y esclava, se conectan ambas por el usuario SYS y se crea un grupo de bases de datos con ellas. Acto seguido se llaman a las funcionalidades de instalar y desinstalar el Monitor de Esquema de todas las bases de datos del grupo. Tras la finalización correcta de la prueba se considera a la misma como vencida por la aplicación.

```
@Test
  public void testInstallUninstallCVersion() throws Exception {
    System.out.println("installCVersion");
    DB_Evolution evolution = new DB_Evolution();
    PanelConsole console = evolution.getConsole();
```

```
IPlugin[] plugins = PluginAction.LoadPlugins(console);
//Creando la base de datos maestra.
Database master = new Database(plugins[0].getDatasource());
IParameter[] parameters = master.getDatasource().getConnection().getParameters().getParameters();
IParameter host = parameters[0];
host.setValue("master.uci.cu");
IParameter user = parameters[1];
user.setValue("sys");
IParameter password = parameters[2];
password.setValue("password_sys");
IParameter logon = parameters[3];
logon.setValue(LogonMode.SYSDBA);
//Creando la base de datos esclava.
Database slave = new Database(plugins[0].getDatasource());
IParameter[] parameters1 = slave.getDatasource().getConnection().getParameters().getParameters();
IParameter host1 = parameters1[0];
host1.setValue("slave.uci.cu");
IParameter user1 = parameters1[1];
user1.setValue("sys");
IParameter password1 = parameters1[2];
password1.setValue("password_sys");
IParameter logon1 = parameters1[3];
logon1.setValue(LogonMode.SYSDBA);
//Conectar las bases de datos.
DatabaseActions.ConnectDatabase(evolution, master);
```

```
    DatabaseActions.ConnectDatabase(evolution, slave);
    //Crear el grupo de bases de datos.
    List<Database> slaves = new LinkedList<Database>();
    slaves.add(slave);
    GroupVersion group = new GroupVersion("Grupo1", "..", master, slaves);
    //Instalar Monitor de Esquema.
    MonitorAction.installCVersion(evolution, group, console);
    //Desinstalar el Monitor de Esquema.
    MonitorAction.deinstallCVersion(evolution, group, console);
    //Desconectar las bases de datos.
    DatabaseActions.DisconnectDatabase(evolution, master);
    DatabaseActions.DisconnectDatabase(evolution, slave);
}
```

4.2. Pruebas del sistema.

Nivel de prueba: Pruebas del sistema.

Método de prueba: Caja negra.

Las pruebas del sistema se realizan por una persona dentro del equipo de desarrollo que está en ese momento asumiendo el rol de probador. Debe haber sido terminada completamente una funcionalidad completa del software para poder realizarle estas pruebas. En los casos de fallas se guardaron las no conformidades encontradas y tras el análisis y la corrección de los errores se procedió a otra iteración de pruebas para esta funcionalidad, hasta que la misma fue vencida completamente por el sistema. En el **Anexo 21** se presentan todos los casos de pruebas del sistema definidos para los requisitos funcionales de la herramienta DB Evolution.

4.3. Análisis de los resultados.

Tras la ejecución de las diferentes pruebas al software se alcanzaron los siguientes resultados:

	Cantidad	Fallos	Éxitos	Cobertura
Pruebas de unidad	4	0	4	100%
Pruebas del sistema	14	0	14	
Total	14			

Tabla 18. Análisis de las pruebas.

Se logró el éxito de todas las pruebas de unidad realizadas, aunque en algunas el resultado solamente fue satisfactorio luego de varias iteraciones. Las mismas fueron de suma importancia al determinar el correcto o no funcionamiento de cada unas las funcionalidades que iban surgiendo. Además permitieron la agilización del trabajo del equipo al ir marcando puntos de control hasta donde se sabía que la aplicación funcionaba bien y permitía enfocar todos los esfuerzos en los fragmentos de código que hasta ese momento no habían sido comprobados. Se comprobó que la herramienta desarrollada cumple con todas las funcionalidades definidas en los requisitos funcionales de la misma.

4.4. Estimación de esfuerzo del sistema.

Los Puntos de Función miden a la aplicación desde una perspectiva del usuario, dejando de lado los detalles de codificación. Es una técnica totalmente independiente de todas las consideraciones del lenguaje y ha sido aplicada en más de 250 lenguajes diferentes. (Dreger, 1989).

A continuación se presentan las funciones disponibles para el usuario presentes en la herramienta DB Evolution:

Nombre de la entrada externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (simple, media, compleja)
Cargar bases de datos.	1	7	Simple
Cargar grupos de bases de datos.	1	4	Simple
Cargar tarea para versionar.	1	4	Simple

Cargar tarea para extraer.	1	4	Simple
Cargar tarea para comparar.	1	7	Simple
Cargar versión.	1	7	Simple
Instalar plugin para Oracle.	10	50	Complejo
Registrar base de datos.	1	7	Simple
Conectar base de datos.	1	3	Simple
Modificar registro de una base de datos.	1	6	Simple
Eliminar registro de una base de datos.	1	1	Simple
Crear grupo de bases de datos.	1	4	Simple
Eliminar grupo de bases de datos.	1	1	Simple
Extraer versión.	1	6	Simple
Sincronizar bases de datos.	1	6	Simple
Total	24	117	

Tabla 19. Entrada Externa

Nombre de la salida externa	Cantidad de ficheros.	Cantidad de elementos de datos	Clasificación (simple, media, compleja)
Guardar bases de datos.	1	7	Simple
Guardar grupos de bases de datos.	1	4	Simple
Guardar tarea para versionar.	1	4	Simple
Guardar tarea para extraer.	1	4	Simple

Guardar tarea para comparar.	1	7	Simple
Guardar versión.	1	7	Simple
Generar script de base de datos.	2	1	Simple
Instalar Monitor de Esquema.	2	10	Media
Desinstalar Monitor de Esquema.	2	2	Simple
Total	12	46	

Tabla 20. Salida Externa

Nombre de la petición	Cantidad de ficheros.	Cantidad de elementos de datos	Clasificación (simple, media, compleja)
Buscar bases de datos.	1	1	Simple
Buscar grupos de bases de datos.	1	1	Simple
Buscar versiones en la aplicación.	1	20	Media
Buscar versiones en la base de datos.	1	3	Simple
Buscar esquemas en la base de datos.	1	1	Simple
Buscar DDL de los objetos.	1	23	Media
Buscar estructura de los objetos.	1	4	Simple
Total	7	53	

Tabla 21. Peticiones

Nombre del fichero interno	Cantidad de ficheros.	Cantidad de elementos de	Clasificación (simple, media,

		datos	compleja)
Bases de datos.	1	7	Simple
Grupos de bases de datos.	1	4	Simple
Tarea para versionar.	1	4	Simple
Tarea para extraer.	1	4	Simple
Tarea para comparar.	1	7	Simple
Versión.	1	7	Simple
Total	6	33	

Tabla 22. Ficheros Internos

Nombre de la interfaz externa.	Cantidad de ficheros.	Cantidad de elementos de datos	Clasificación (simple, media, compleja)
-	-	-	-
Total		**0**	

Tabla 23. Interfaces Externas

Elementos	Simple		Medio		Complejo		Subtotal
	No	Peso	No	Peso	No	Peso	
Entradas externas	14	3	0	4	1	6	48
Salidas externas	8	4	1	5	0	7	37
Ficheros internos	7	7	0	10	0	15	49
Peticiones	5	3	2	4	0	6	23
Interfaces externas	0	5	0	7	0	10	0
Total							**157**

Tabla 24.Puntos de función desajustados.

Cálculo de instrucciones fuentes, esfuerzo, tiempo de desarrollo, cantidad de hombres y costo.

Como SCRUM carece de casos de uso se hace necesario entonces realizar la estimación utilizando el modelo COCOMO II usando Puntos de Función y/o Líneas de Código Fuente (SLOC). COCOMO II basado en puntos de función contempla solamente los puntos de función desajustados.

Características	Valor
Puntos de función desajustados	157
Lenguaje (Java)	53
Instrucciones fuentes por puntos de función	8321
Instrucciones fuentes	8.321 KSLOC

Tabla 25. Características de la herramienta.

El valor escalar se obtiene a partir de variables que indican las características que el proyecto presenta en lo que a su complejidad y entorno de desarrollo se refiere. Las variables son las siguientes.

Nombre	Valor	Justificación
PREC	3.72	Existen proyectos de este tipo pero no son similares aunque tienen aspectos parecidos.
FLEX	1.01	Se cuentan con algunos acuerdos de requerimientos preestablecidos aunque no con interfaces externas pre-existentes. Existía mucha flexibilidad para los desarrolladores.
TEAM	5.48	Interacciones entre los miembros del equipo muy difíciles.
RESL	1.41	La herramienta posee altos riesgos críticos que llevaban un alto grado de dificultad en su tratamiento.
PMAT	4.68	No existe una experiencia previa en el desarrollo de aplicaciones de este tipo aunque si en las herramientas utilizadas.

Total (SF)	16.3

Tabla 26. Factor de escala.

Para la obtención de los multiplicadores de esfuerzo se utilizó el modelo de Diseño Preliminar el cual ajusta el esfuerzo nominal usando siete factores de costo.

Nombre	Valor	Justificación
RCPX	1.30	El sistema presenta un nivel de complejidad alto y debe ser altamente confiable.
RUSE	1.07	El nivel de reusabilidad es alto.
PDIF	0.87	Uso de la memoria y almacenamiento bajo, plataforma estable.
PREX	0.87	El nivel de experiencia en el uso del lenguaje y la plataforma de trabajo es alto.
PERS	0.83	La capacidad del personal es alta.
FCIL	0.73	Se utilizan herramientas de modelado que facilitan el trabajo y entornos de desarrollo integrados.
SCED	1.00	Se utilizó un poco más de tiempo que el planificado para el desarrollo del sistema.
Total(EM)	0.95	

Tabla 27. Multiplicadores de esfuerzo.

Calculando las variables:

$$B = 0.91 + 0.01 \times \sum (W_i)$$

$$B = 0.91 + 0.01 \times \sum (3.72 + 1.01 + 5.48 + 1.41 + 4.68)$$

$$B = 1.073$$

$$F = D + 0.2 \times (B - 1.01)$$

$$F = 0.33 + 0.2 \times (1.073 - 1.01)$$

$F = 0.34$

Variables:

$A = 2.94,\ B = 1.073, C = 3.67,\ D = 0.33,\ F = 0.34$

Calculando el esfuerzo nominal requerido en meses- hombre ($PM_{nominal}$).

$PM_{nominal} = A \times (KSLOC)^{B}$

$PM_{nominal} = 2.94 \times (8.321)^{1.073}$

$PM_{nominal} = 28.55$

El esfuerzo calculado en la ecuación anterior es un valor nominal y debe ser ajustado en base a las características del proyecto utilizando los multiplicadores de esfuerzo (ME) calculados anteriormente. El esfuerzo ajustado se calcula mediante la siguiente ecuación:

$$PM_{estimado} = PM_{nominal} \times \prod_{i=1}^{7} ME_i$$

$PM_{estimado} = 28.55 \times 0.95$

$PM_{estimado} = \mathbf{27.12}$

Para el cálculo del tiempo de desarrollo se utiliza la siguiente ecuación:

$TDEV = C \times (PM_{estimado})^{F}$

$TDEV = 3.67 \times (27.12)^{0.34}$

$TDEV = \mathbf{11.27}$

A partir del esfuerzo y el tiempo de desarrollo calculado se puede obtener la cantidad de hombres que se necesitan (CH).

$$CH = \frac{PM_{estimado}}{TDEV}$$

$$CH = \frac{27.12}{11.27}$$

$CH = \mathbf{3}$

Los costos en los que se incurriría de desarrollarse el sistema serían.

$Costo = CH \times Sal \times PM_{estimado}$

$Costo = 3 \times 100 \times 27.12$

$Costo = \mathbf{8136}$

En la siguiente tabla se resumen los resultados obtenidos.

Cálculo de	Valor
Esfuerzo	27.12 hombres/mes
Tiempo de desarrollo	11.27 meses
Cantidad de hombres	3 hombres
Salario medio	$ 100
Costo	$ 8136

Tabla 28. Resultados de la estimación.

Todo producto informático conlleva a un costo de producción, el cual debe ser justificado en base a los beneficios reportados por el mismo. El desarrollo de la herramienta DB Evolution no conlleva a grandes gastos, puesto que solo requiere el salario de los desarrolladores, por lo que su implementación es factible. Este resultado es posible gracias a la utilización de diferentes APIs y herramientas libres que no requieren el pago de licencias.

Conclusiones

Tras haber sido efectuadas todas las pruebas previstas y haberse alcanzado el 100% de las pruebas con resultado satisfactorio se puede concluir que la aplicación es segura, robusta y cumple con los objetivos definidos. La misma es por tanto capaz de dar solución al problema de la investigación automatizando los procesos claves de la Gestión de Configuración en Sistemas de Bases de Datos como son el Control de Cambios y el Control de Versiones. Además se determinó la factibilidad de la herramienta propuesta.

CONCLUSIONES

Con la realización del trabajo expuesto, de forma general, se le ha dado cumplimiento a cada uno de los objetivos generales y específicos de la investigación que habían sido definidos. Se realizó un estudio del estado del arte de los principales conceptos relacionados con el proceso de Gestión de Configuración en los Sistemas de Bases de Datos, pudiéndose definir las pautas que guiaron el desarrollo de la nueva herramienta.

Se seleccionó la metodología Scrum para guiar el desarrollo, por ser la que más se adaptaba a las condiciones reales del proceso de producción a realizar, así como el lenguaje de programación Java para la implementación. Se realizó la documentación asociada al nuevo producto que había sido acordada con el cliente la cual fue aprobada satisfactoriamente por el mismo.

Quedó implementada una herramienta que automatiza los procesos de Control de Cambios y Control de Versiones a los Sistemas de Bases de Datos, utilizando al propio gestor de la base de datos para ello. La misma es extensible para cualquier gestor de bases de datos e implementa funcionalidades comunes para todos. Fue desarrollado el plugin correspondiente al gestor Oracle dado que era el más importante para el cliente.

Se realizaron además un conjunto de pruebas de unidad y pruebas del sistema a la herramienta, las cuales pudieron determinar el correcto funcionamiento de cada una de las funcionalidades implementadas y que el producto cumplía con los requisitos especificados por el cliente. También se realizó un estudio de la factibilidad del sistema propuesto.

RECOMENDACIONES

Tras haber finalizado el desarrollo de la herramienta DB Evolution se recomienda:

- Que la herramienta sea usada en el Centro de Identificación y Seguridad Digital, así como en cualquier otro proyecto donde sea necesario implementar el proceso de Gestión de Configuración en Sistemas de Bases de Datos.
- Continuar con el desarrollo de la herramienta agregándole nuevas funcionalidades, así como módulos de reportes y auditorías.
- Implementar plugins para otros Sistemas Gestores de Bases de Datos como pueden ser PostgresSQL y SQL Server.
- Implementar un módulo de migración de bases de datos para permitir la interacción de las bases de datos sin tener en cuenta el tipo de gestor.

BILBLIOGRAFÍA CITADA

Berzal Galiano, Fernando. *Diseño de arquitecturas.* [Documento PDF]

Dreger, J. B. 1989. *Function Point Analysis.* s.l. : Prentice-Hall, 1989.

Esteban, Ángel. 2000. *ACCESO A BASES DE DATOS CON JAVA - JDBC 2.0.* Madrid (España) : Grupo EIDOS Consultaría y Documentación Informática, 2000. ISBN 84-88457-16-2.

IEEE. 1987. *IEEE Guide to Software Configuration Management.* s.l. : American National Standar, 1987. Std. 10421987.

—. 1990. *IEEE Standard Glossary of Software Engineering Terminology.* [PDF] New York, USA : IEEE, 1990. 0-7381-0391-8, SS13748.

Jonassen Hass, Anne Mette. 2002. *Configuration Management Principles and Practice.* s.l. : Addison Wesley, 2002. ISBN : 0-321-11766-2.

Kuchana, Partha. 2004. *Software Architecture Design Patterns in Java.* United States of America : AUERBACH PUBLICATIONS, 2004. 0-8493-2142-5.

LNCS. 2008. *GUÍA PRÁCTICA DE GESTIÓN DE CONFIGURACIÓN.* [Guía] España : INTECO Instituto Nacional de tegnologías de la Comunicacion, 2008.

Palacios, Juan. 2007. *Flexibilidad con Scrum.* España : SafeCreative, 2007. identificador: 0710210187520.

Patel, Pratik. 1996. *Java Database Programming with JDBC.* s.l. : The Coriolis Group, 1996. ISBN: 1576100561.

PostgreSQL Global Dvelopment Group. 2010. PostgreSQL. [En línea] PostgreSQL Global Dvelopment Group, 2010. [Citado el: 25 de 01 de 2010.] http://www.postgresql.org/about/.

PostgreSQL. 2010. PostgreSQL 8.4.2 Documentation. [En línea] 2010. [Citado el: 22 de 02 de 2010.] http://www.postgresql.org/docs/current/static/information-schema.html.

Pressman, Roger S. 2002. *Ingeniería del Software.Un enfoque práctico.* s.l. : McGraw-Hill, 2002. ISBN : 8448132149, 9788448132149.

Roddick, John F. 1995. *A Survey of Schema Versioning Issues for Database Systems.* University of South Australia : Advanced Computing Research Centre,School of Computer and Information Science, 1995.

Sun Microsystem. 2010. [En línea] 20 de 01 de 2010. http://es.sun.com/sunnews/press/2009/20091214.jsp.

BILBLIOGRAFÍA CONSULTADA

Arora, Geetanjali, Aiaswamy, Balasubramaniam and Pandey, Nitin. 2002. *Programación c#.* Madrid(España) : ANAYA MULTIMEDIA, 2002. ISBN: 84-4 15-1420-8.

Berzal Galiano, Fernando. *Diseño de arquitecturas.* [Documento PDF]

Canós, José H., Letelier, Patricio and Penadés, Carmen. *Metodologías Ágiles en el Desarrollo de Software.* Valencia : DSIC -Universidad Politécnica de Valencia.

de la Vega García, Erik. 2009. *Control de cambios y gestión de la configuración de la base de datos.* [PDF] La Habana,Cuba : s.n., 2009.

Dreger, J. B. 1989. *Function Point Analysis.* s.l. : Prentice-Hall, 1989.

Embarcadero Change Mannanger. [Online] Embarcadero Technologies. [Cited: enero 19, 2009.] http://www.embarcadero.com/products/change-manager.

Esteban, Ángel. 2000. *ACCESO A BASES DE DATOS CON JAVA - JDBC 2.0.* Madrid (España) : Grupo EIDOS Consultaría y Documentación Informática, 2000. ISBN 84-88457-16-2.

Ferguson, Jeff, Patterson, Brian and Beres, Jason. 2003. *La Biblia C#.* Madrid(España) : Anaya Multimedia, 2003. ISBN: 84-415-1484-4.

Galindo, Jose, Urrutia, Angélica and Piattini, Mario. 2005. *Fuzzy Databases: Modeling , Design and Implementation.* United States of America : Idea Group Publishing, 2005. ISBN 1-59140-326-X.

IEEE. 1987. *IEEE Guide to Software Configuration Management.* s.l. : American National Standar, 1987. Std. 10421987.

—. 1990. *IEEE Standard Glossary of Software Engineering Terminology.* [PDF] New York, USA : IEEE, 1990. 0-7381-0391-8, SS13748.

International Function Point Users' Group (IFPUG). 2009. [Online] IFPUG, 03 2009. [Cited: 05 01, 2010.] http://www.ifpug.org/.

Jonassen Hass, Anne Mette. 2002. *Configuration Management Principles and Practice.* s.l. : Addison Wesley, 2002. ISBN : 0-321-11766-2.

Kuchana, Partha. 2004. *Software Architecture Design Patterns in Java.* United States of America : AUERBACH PUBLICATIONS, 2004. 0-8493-2142-5.

LNCS. 2008. *GUÍA PRÁCTICA DE GESTIÓN DE CONFIGURACIÓN.* [Guía] España : INTECO Instituto Nacional de tegnologías de la Comunicacion, 2008.

Melton, Jim and Simon, Alan R. 2002. *SQL:1999: Understanding Relational Language Components.* United States of America : Academic Press, 2002. ISBN: 1-55860-456-1.

Microsoft. 2010. MSDN Visual Studio. [Online] Microsoft, 01 21, 2010. http://msdn.microsoft.com/es-es/library/zw4w595w.aspx.

MSDN Microsoft Developer Network. 2010. Querying the SQL Server System Catalog. [Online] MSDN, 2010. [Cited: 02 23, 2010.] http://msdn.microsoft.com/en-us/library/ms189082.aspx.

—. 2010. Separación de esquemas de usuario. [Online] MSDN, 2010. [Cited: 02 23, 2010.]

OCELOT. 2002. SQL-92 INFORMATION SCHEMA. [Online] 2002. [Cited: 02 23, 2010.] http://www.ocelot.ca/is.htm.

Palacios, Juan. 2007. *Flexibilidad con Scrum.* España : SafeCreative, 2007. identificador: 0710210187520.

Patel, Pratik. 1996. *Java Database Programming with JDBC.* s.l. : The Coriolis Group, 1996. ISBN: 1576100561.

PostgreSQL Global Dvelopment Group. 2010. PostgreSQL. [Online] PostgreSQL Global Dvelopment Group, 2010. [Cited: 01 25, 2010.] http://www.postgresql.org/about/.

PostgreSQL. 2010. PostgreSQL 8.4.2 Documentation. [Online] 2010. [Cited: 02 22, 2010.] http://www.postgresql.org/docs/current/static/information-schema.html.

Pressman, Roger S. 2002. *Ingeniería del Software.Un enfoque práctico.* s.l. : McGraw-Hill, 2002. ISBN : 8448132149, 9788448132149.

Programación.com. 2010. Programación.com. [Online] Programación.com, 01 21, 2010. http://www.programacion.com/tutorial/csharp/3/.

Roddick, John F. 1995. *A Survey of Schema Versioning Issues for Database Systems.* University of South Australia : Advanced Computing Research Centre,School of Computer and Information Science, 1995.

Sun Microsystem. 2010. [Online] 01 20, 2010. http://es.sun.com/sunnews/press/2009/20091214.jsp.

System, Sparx. 2010. Guía de Usuario de Enterprise Architect 7.0. [Online] 2010. [Cited: 04 01, 2010.] http://www.sparxsystems.com.ar/download/ayuda/index.html?componentdiagram.htm.

GLOSARIO DE TÉRMINOS

API: (*Application Programming Interface*).Conjunto de funciones y procedimientos que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

applet: Componente de una aplicación que se ejecuta en el contexto de otro programa.

bytecode: Código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable.

bug: Resultado de un fallo o deficiencia durante el proceso de creación de programas de ordenador o computadora.

CMMI: (*Capability Maturity Model Integration*). Modelo de Integración de la Capacidad y Madurez es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

DBMS :(*Database Management System*). Los Sistemas de Gestión de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

hash: Se refiere a una función o método para generar claves o llaves que representen de manera casi unívoca a un documento, registro, archivo, etc., resumir o identificar un dato a través de la probabilidad, utilizando una función hash o algoritmo hash. Un hash es el resultado de dicha función o algoritmo.

IEEE: (Institute of Electrical and Electronics Engineers). Asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías.

ISO:(International Organization for Standardization). La Organización Internacional para la Estandarización o ISO es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica

inyecciones SQL: Vulnerabilidad informática en el nivel de la validación de las entradas a la base de datos de una aplicación.

licencia BSD: (*Berkeley Software Distribution*). Es una licencia de software libre permisiva, con menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público, que permite el uso del código fuente en software no libre.

MD5: (*Message-Digest Algorithm 5*). El Algoritmo de Resumen del Mensaje 5 es un algoritmo de reducción criptográfico de 128 bits ampliamente usado a nivel mundial.

metainformación: Se refiere a información sobre la información.

plugin: Aplicación que se relaciona con otra para aportarle una funcionalidad nueva y generalmente muy específica.

prototipo: Objeto diseñado para una demostración de cualquier tipo.

sprint: Se refiere al período de tiempo en que ocurre una iteración completa de Scrum.

SQL: (*Structured Query Language*). Lenguaje formal declarativo, estandarizado por la ISO para manipular información en una base de datos.

script: Se refiere a un archivo que contiene un conjunto de instrucciones o comandos.

tuplas: Se refiere a parejas o más datos insertados en los campos de las tablas de una base de datos.

UML: (*Unified Modeling Language*). El Lenguaje Unificado de Modelado es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

VPN: (Virtual Private Network). Se refiere a una Red Virtual Privada**.**