Universidad de las Ciencias Informáticas

Facultad 6

Título: "Desarrollo de una herramienta para la generación e integración de bases de datos".

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

Autores: Mirelys Benítez Pérez.
Yunier Santana Aldana.

Tutoras: Ing. Irina Collazo Ávila.
Ing. Yadira Robles Aranda.
Ing. Elvismary Molina de Armas.

Ciudad de la Habana, Cuba

Junio 2009.

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte".

Ernesto Che Guevara

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yunier Santana Aldana **Mirelys Benítez Pérez**

_____ _____

Firma del Autor Firma del Autor

Ing. Irina Collazo Ávila **Ing. Elvismary Molina de Armas** **Ing. Yadira Robles Aranda**

_____ _____ _____

Firma de la Tutora Firma de la Tutora Firma de la Tutora

Datos del Contacto.

Tutora.

Ing. Irina Collazo Ávila.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Correo electrónico: icollazo@uci.cu

Tutora.

Ing. Elvismary Molina de Armas.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Correo electrónico: emolina@uci.cu

Tutora.

Ing. Yadira Robles Aranda.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Correo electrónico: yrobles@uci.cu

Agradecimientos

A Fidel y la Revolución por la creación de esta Universidad.

A nuestros padres por ser el centro de nuestras vidas, y por habernos guiado para estar en el lugar que hoy estamos.

A todos nuestros amigos que durante 5 largos años han estado de nuestro lado en las buenas y en las malas.

A todos los profesores que han contribuido con nuestra formación profesional.

A todos nuestros compañeros de proyecto Ingrid, Curiel, Leinys, Diole, Lexy, Ezequiel, Randy y Ángel Manuel por aportar su granito de arena y contribuir con sus críticas.

A nuestros tutores Irina, Elvismary y Yadira por apoyo y colaboración para poder lograr una realización exitosa de este trabajo.

A nuestros líderes de proyecto Yusdenys y Yosuan por su ayuda y por enseñarnos a trabajar como un equipo.

A la profesora Dayana por haber sido una tutora más y por su ayuda incondicional cuando necesitamos de ella.

A todos muchas gracias…

Dedicatoria

De Mirelys

A los seres que han hecho que cada minuto de mi existencia sea un verdadero derroche de amor y cariño, a las personas que más quiero en la vida, mi mamá, mi papá y mi hermanita, ellos que siempre me brindaron amor, por todo su apoyo y fuerza, por su confianza y por enseñarme a ser fuerte en los momentos más difíciles, a ellos le debo gran parte de mi triunfo…los quiero mucho.

A mi familia en general, por confiar siempre en mí, por confiar en mi capacidad de poder escalar hasta esta elevación, por aconsejarme cuando lo necesite, por estar siempre alegres y por darme todo su cariño.

A mi grupo de amigas Las Super Nenas: Yasnay, Kilmenys, Maidelys, Yeneisi, Yuriskenia, Lisset y Kenia, a todas por estar a mi lado cuando lo necesité, por tenerlas junto a mí en los momentos lindos y tristes, por estar juntas en cada una de locuras que se nos han ocurrido, porque cada una ha dejado caer su gotica de agua y he podido aprender de todas, créanme chicas que el recuerdo más lindo que me llevo de esta universidad son ustedes…nunca las olvidaré.

A una personita muy especial en mi vida, Disnier Camejo, por ayudarme en estos largos años de universidad, por estar ahí cada vez que me hizo falta, por soportar todas mis malacrianzas y por hacerme reír en todo momento, por quererme tanto y por formar parte de mi vida nunca te voy a olvidar.

A mis viejos amigos de siempre Marcia, Yamila, Alcide, Darian, Isis, Eduardo, Dairon, Roly, Mailyn, a ellos porque siempre han estado ahí para mí, por apoyarme, por compartir tantos momentos juntos, por dejarme formar parte de sus vidas, a todos los adoro.

A todos los que de una forma u otra se preocupan por saber como estoy, si he dejado de mencionar algún nombre quiero que sepa que está en mi corazón, que le agradezco ser parte de mi vida y siempre tendrá mis agradecimientos.

De Yunier

Llega el momento más difícil de mi carrera, y es el de intentar resumir en nombres de personas lo agradecido que en este momento me siento. Pienso que ésta sección, quizás, no sea suficiente para mencionar a todos aquellos que han puesto un poquito de empeño y su ayuda desinteresada para cumplir éste, mi sueño, y si existe alguien que no sienta su presencia en mis palabras, reciba mis más sinceras disculpas y tenga la certeza que siempre tendrá mi gratitud.

Quiero agradecer a mi madre, por estar siempre en cada paso que he dado de mi carrera, ser la principal fuente de inspiración para seguir adelante y contribuir con su amor a formar el profesional en el que me he convertido. A mi padre que aunque no esté físicamente conmigo siempre está a mi lado guiándome en cada paso que doy. A los dos los adoro.

A mis hermanas Mariela, Yaneisy y Maité por su cariño, por comprenderme y por confiar siempre en mí, por estar presentes en todos los momentos de mi vida…las quiero mucho.

A toda mi familia por siempre confiar en mi capacidad de poder llegar hasta el lugar en el que hoy me encuentro, especialmente a mi hermana Mariela, mi cuñado Alberto y mi prima Maida.

A mis amigos, más que amigos, hermanos que siempre han estado en los buenos y malos momentos, por dejarme entrar en sus vidas, JuanCa, Ingrid, William, Felipe, y demás…. nunca los olvidaré.

De JuanCa e Ingrid ni hablar, pues los catalogo como héroes por soportar mis malcriadeces durante tanto tiempo. A los dos les agradezco de corazón pues han sido de las mejores personas que he conocido en mi vida y serán siempre mis papás de la UCI.

A Lina Martha por todo el apoyo y la ayuda que me brindó en los últimos años de mi carrera, le estoy muy agradecido.

Y a todo aquel que luego del abrazo y del saludo, preguntó ¿Cómo va la Tesis?, muchas, muchas gracias.

Resumen

La presente investigación surge como una variante para simplificar los problemas de conectividad que actualmente presentan muchos sistemas automatizados en los diferentes sectores de la sociedad cubana, sobre todo en la red de salud pública. El aporte de dicha investigación se concentra en una aplicación de escritorio que permita desacoplar e integrar bases de datos.

Se tiene como objetivo fundamental el diseño y la implementación de una herramienta que sea capaz de desfragmentar y acoplar bases de datos de un sistema informático determinado mediante una interfaz legible y fácil de usar. La solución propuesta posibilitará el desarrollo y publicación de un producto que ayudará a centralizar la información de diversas entidades a lo largo de todo el país.

Palabras claves: base de datos, sincronizar, generar, migrar.

Índice

INTRODUCCIÓN

La era de la tecnología ha hecho que el mundo del siglo XXI sea totalmente cambiante. Las organizaciones buscan soluciones tecnológicas que les den flexibilidad y capacidad de reacción ante las exigencias de rápida adaptabilidad que mueve el mercado actual. Una de las necesidades más solicitadas por las empresas es la agilización de procesos. En una gran mayoría de situaciones las soluciones más comunes se implementan mediante la integración de sistemas y desarrollo de aplicaciones, que permitan agilizar flujos de información dentro y fuera de las organizaciones. Además buscan soluciones que sean fácilmente actualizables y reutilizables en múltiples escenarios.

La tecnología ha contribuido en favor de que exista un mayor intercambio de información a nivel mundial. Este intercambio es realizado a través de sistemas de información cada vez más complejos y necesarios en el desarrollo de cualquier entidad. Para suplir las necesidades de gestión de toda esta información se desarrollan diariamente por todo el mundo un gran número de aplicaciones informáticas con diversas herramientas y métodos de desarrollo.

Cuba no ha quedado exenta de estos avances tecnológicos, durante los últimos 20 años diversas instituciones cubanas han desarrollado sistemas encaminados a lograr niveles de informatización para distintos sectores. Con la implementación en el 2003, de los programas de la revolución, que incluyen como prioridad la informatización de los servicios, se inicia en Cuba un avance vertiginoso hacia la informatización de la sociedad, orientado en primer lugar a la superación y desarrollo profesional, que a su vez se ha extendido a la automatización de los servicios médicos, la investigación, la información científico-técnica y el apoyo en la toma de decisiones.

Para contribuir al desarrollo de la sociedad cubana se ha convocado a un grupo de instituciones del Ministerio de Informática y las Comunicaciones (MIC), entre las que se encuentra la Universidad de las Ciencias Informáticas (UCI).

La UCI como parte del grupo de apoyo a la informatización del país desarrolla productos para el bienestar de la sociedad entre los que se encuentran los cuatro pilares fundamentales del proceso revolucionario cubano: la educación, la salud, la seguridad social y la cultura. Esta institución a pesar de ser muy joven aún, juega un papel importante en el desarrollo informático de la sociedad cubana.

Uno de los principales problemas existentes en los sistemas automatizados en Cuba, es que un número importante de entidades no están conectados a la red nacional, trayendo consigo dificultades en el despliegue de sistemas centralizados de datos, donde la información esté disponible desde un único punto al cual puedan acceder todas las entidades implicadas desde cualquier ubicación del país.

Debido a la falta de conexión en estos sistemas, cuando se desean realizar investigaciones o simplemente recopilar la información de una determinada región o entidad ubicada en uno de los municipios que no poseen conectividad con la red nacional, muchas veces se opta por replicar las bases de datos completas, causando redundancia, información innecesaria para determinada investigación, lentitud en las consultas, problemas de seguridad y confidencialidad de la información, así como dificultad a la hora de sincronizar los datos.

Tomando como base la **situación problémica** planteada anteriormente surge como **problema científico de la investigación:** ¿Cómo posibilitar el desacoplamiento e integración de un sistema informático a nivel de base de datos?

Teniendo como **objeto de estudio** el proceso de extracción, transformación y carga de datos y como **campo de acción** derivado del mismo, proceso de extracción, transformación y carga de datos de una base de datos en MySql.

Para dar solución al problema planteado se ha trazado como **objetivo general**: Desarrollar una herramienta informática que permita desacoplar e integrar los datos de un sistema.

Entre los **objetivos específicos** propuestos se encuentran:

- Definir las funcionalidades de la herramienta informática.
- Diseñar la herramienta informática.
- Implementar la herramienta informática.

Para poder cumplir los objetivos y lograr una solución adecuada a la situación problémica especificada se plantean las siguientes **tareas investigativas**:

- Estudio de las técnicas existentes para la extracción, transformación y carga de datos.

- Definición de las funcionalidades de la herramienta a desarrollar.
- Realización del diseño de la herramienta informática.
- Implementación de los componentes.

El presente documento está estructurado en cuatro capítulos. En los que se describen los métodos y procedimientos a seguir para dar cumplimiento a los objetivos trazados. A continuación una breve descripción de cada uno de ellos.

Capítulo 1: Fundamentación teórica

En este capítulo comprende el estudio del estado del arte acerca de los elementos fundamentales a tener en cuenta para la solución del problema planteado. Se argumenta la selección de las herramientas y metodología que se usarán para dar solución al problema científico.

Capítulo 2: Características del sistema

Este capítulo describe la propuesta de solución para el problema científico de esta investigación, mostrando los procesos del negocio mediante el Modelo del Dominio. Se definen los requisitos funcionales y no funcionales de la herramienta y se presenta el diagrama de caso de uso del sistema con sus descripciones correspondientes.

Capítulo 3: Diseño del sistema.

En este capítulo se hace referencia al flujo de trabajo análisis y diseño, en el cual se describen las clases más importantes, además se muestran los diferentes diagramas de clases pertenecientes a cada uno de los casos de uso, así como se realiza el modelo de despliegue de la aplicación.

Capítulo 4: Implementación del sistema.

Está enfocado a la implementación del sistema con el objetivo de darle solución a los requisitos funcionales. Se realiza el modelo de implementación y se aplican patrones de diseño y de arquitectura a cada una de las clases del diseño.

Capítulo 1: FUNDAMENTACIÓN TEÓRICA

Este capítulo hace un análisis de las tendencias de las diferentes herramientas que se utilizan para desacoplar bases de datos e integrar información en las mismas. Se hace una selección del lenguaje de programación, herramientas y metodologías que se utilizarán para dar solución al problema planteado.

1.1 Proceso de extracción, transformación y carga de datos.

Las tecnologías utilizadas en el proceso de extracción, transformación y carga de datos, conocidas como herramientas ETL, son en esencia procesos que permiten a las organizaciones mover datos desde múltiples fuentes, reformatearlos, limpiarlos y cargarlos en otra base de datos, o en otro sistema operacional para apoyar un proceso de negocio. El objetivo principal de las herramientas ETL es facilitar el desarrollo y la reutilización de aplicaciones que migran datos aplicando transformaciones.

Dentro de las desventajas de estas tecnologías tenemos que los procesos ETL pueden ser muy complejos. Un sistema de este tipo mal diseñado puede provocar importantes problemas operativos. Además en un sistema operacional el rango de valores de los datos o la calidad de éstos pueden no coincidir con las expectativas de los diseñadores a la hora de especificarse las reglas de validación o transformación. [1]

Debido a que las herramientas ETL engloban un amplio nivel de funcionalidades, muchas de las cuales no se ajustan a lo que se necesita, fue necesario restringir los conceptos de extracción, transformación y carga de datos de acuerdo a las necesidades actuales de esta investigación.

Extraer: Es el proceso de obtener la información necesaria de una base de datos.

Transformar: Cuando se vaya a migrar la información de una base de datos para otra, se puede dar el caso de que se deba actualizar un registro existente con información antigua, sobrescribiendo este.

Cargar: Permite cargar la información procesada en una base de datos y guardarla en una base de datos Máster.

1.2 Herramientas existentes para la extracción, transformación y carga de datos.

1.2.1 Herramienta de gestión de bases de datos: Replicación

Es una herramienta que se encuentra dentro del paquete Data Integration Suite de Sybase, que brinda soporte a un entorno empresarial para la gestión de los datos mediante la replicación y la sincronización de copias de Sybase ASE, Oracle®, SQL Server y otros sistemas de bases de datos. Proporciona un mecanismo seguro, fiable de alto rendimiento para la distribución de datos a lo largo de toda la empresa, permitiendo afrontar las siguientes necesidades: [2]

- La transferencia, distribución y sincronización de los datos.
- La migración entre sistemas.
- La continuidad del negocio.

Esta herramienta y otras que se encuentran dentro del paquete son muy poderosas, debido a la amplia gama modular de soluciones tecnológicas con herramientas avanzadas de administración, desarrollo y modelado integrados. Existen varios motivos por los cuales no se decidió utilizar esta tecnología, primeramente no es un producto libre, hay que comprar la licencia, y otro factor fundamental es que no le permite el acceso de licencias a Cuba.

1.2.2 Herramienta de gestión de bases de datos: Unicenter Database Command Center.

El producto Unicenter Database Command Center (DCC) es una consola de gestión que puede ser descargada en cualquier estación de trabajo dotada con un navegador. La herramienta no requiere instalación de ningún software cliente y ofrece una visión unificada cuando se trabaja con diversos sistemas. Esta es la solución al problema que se presentaba cuando se quería trabajar con múltiples configuraciones y diferentes bases de datos. Cada suministrador de bases de datos proporcionaba una herramienta para gestionarla, pero el sistema de código abierto que ofrece DCC permite la gestión de las diferentes bases de datos con una misma herramienta.

La plataforma incluye gestión centralizada de esquemas y una herramienta de gestión de cuentas para plataformas cruzadas. Brinda la posibilidad de importar y exportar datos de las diferentes bases de datos con capacidades de reorganización. Además de la funcionalidad de descarga, permite a los administradores otorgar o revocar privilegios de acceso y asignar derechos a los usuarios basados en roles. [3]

Esta herramienta es una buena solución ante el creciente número de bases de datos dentro de las empresas, y la necesidad de unificar su administración. Permite en un momento determinado integrar bases de datos que han sido fragmentadas, pero por si solo no brinda la posibilidad de fragmentar una base de datos.

Conclusiones a las que se arriban:

Diseñar una herramienta capaz de fragmentar una base de datos, dando la posibilidad de crear bases de datos más pequeñas con las cuales se pueda trabajar de forma más eficiente y rápida. Una vez gestionada la información en esta subbase de datos, la herramienta debe migrar todo el flujo de datos hacia la base de datos que le dio origen.

1.3 Fundamentación de las tecnologías y herramientas a utilizar.

1.3.1 Metodología de Desarrollo de Software.

Una Metodología de desarrollo es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo producto de software. Una metodología representa el camino para desarrollar software de una manera sistémica. Estas persiguen tres necesidades principales:

- Mejores aplicaciones, que conduzcan a una mejor calidad.
- Un proceso de desarrollo controlado.
- Un proceso normalizado en una organización, no dependiente de un personal.

Proceso Unificado del Software (RUP).

El Proceso Unificado del Software (RUP) es un proceso de desarrollo de software, que constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es un proceso para el desarrollo de un proyecto de un software que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto. Como características esenciales posee que está dirigido por los casos de uso, centrado en la arquitectura y es iterativo e incremental.

RUP divide el proceso en 4 fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

- Inicio: Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos. Se define el alcance del proyecto.

- Elaboración: Se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.

- Construcción: Se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.

- Transición: Se instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Características del proceso unificado de desarrollo

- **Iterativo e Incremental**

El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones. Estas iteraciones ofrecen como resultado un incremento del producto desarrollado, que añade o mejora las funcionalidades del sistema en desarrollo.

Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación

y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

- **Dirigido por los casos de uso**

En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc. el proceso dirigido por casos de uso es el RUP.

- **Centrado en la arquitectura**

El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc.

1.3.2 Lenguaje Unificado de Modelado: UML.

El Lenguaje Unificado de Modelado (UML según sus siglas en inglés) está considerado como el lenguaje de modelado gráfico y visual por excelencia, utilizado para especificar, visualizar, construir y documentar los componentes de un sistema de software. Está pensado para poder aplicarse en cualquier medio que necesite capturar requerimientos y comportamientos del sistema que se desee construir. Ayuda a comprender y a mantener de una mejor forma un sistema basado en un área que el analista o desarrollador puede desconocer. UML surgió con el propósito de lograr una estandarización universal al crecido número de metodologías de desarrollo que habían estado apareciendo.

Este lenguaje permite captar información sobre la estructura estática y dinámica de un sistema, en donde la estructura estática proporciona información sobre los objetos que intervienen en determinado proceso y las relaciones que existen entre ellos, y la estructura dinámica define el comportamiento de los objetos a lo largo de todo el tiempo que estos interactúan hasta llegar a su, o sus objetivos. Una característica sobresaliente de UML es que no es un método, sino un lenguaje de modelado. Un método define su notación y su proceso a seguir durante el ciclo de vida de desarrollo del software. Se encarga de la notación gráfica y su significado, a partir de la cual se crearán los diseños de sistemas y no depende de un proceso, el cual sería el encargado de orientar los pasos a seguir para elaborar el diseño. Mejorar la comunicación es la idea a seguir

con la creación de los diagramas mediante UML, sin dudas ayuda a que los desarrolladores de software se comuniquen con un mismo lenguaje de modelado independiente de las metodologías empleadas. Por esta razón la mayoría de las fuentes coinciden en que la principal ventaja de UML sobre otras notaciones orientadas a objetos (OO) es que elimina las diferencias entre semánticas y notaciones.

UML tiene como características:

- Tecnología orientada a objetos
- Viabilidad en la corrección de errores.
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- Cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos.
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar. [4]

1.3.3 Herramientas Case: Visual Paradigm 3.1

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

Características Visual Paradigm:

- Genera código en PHP, Java, C++, y otros. Permite incorporar los dibujos del Visio en cualquier diagrama de UML.

- No solo realiza diagramas de UML normales, sino también posibilita modelar hardware, dominio-específico, el software, conectando una red de computadoras los componentes usando sus iconos, más allá de las anotaciones de UML normales.

- Es multiplataforma, está disponible en las plataformas Windows y Linux.

- Está orientada a la creación de diseños usando el paradigma de programación orientado a objetos.

1.3.4 Servidor de Base de Datos: MySQL 5.0.6

MySQL es un sistema de gestión de base de datos relacional, multihilo mediante hilos del kernel y multiusuario. Opera en una arquitectura cliente/servidor, de tal manera que el servidor sólo tiene que enviarle una cadena de caracteres y esperar la devolución de los datos. Es un sistema de gestión de bases de datos más usado. Todo el mundo puede acceder al código fuente y contribuir con ideas, elementos, mejoras o sugerir optimizaciones. MySQL ha pasado de ser una pequeña base de datos a una completa herramienta. Es una tecnología útil para la creación de bases de datos seguras al poseer un sistema de privilegios y contraseñas que es muy flexible y seguro, que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de ellas a través de la Web está encriptado al conectarse con un servidor.

Las principales características de este gestor de bases de datos son las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.

- Soporta gran cantidad de tipos de datos para las columnas.

- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP).

- Proporciona sistemas de almacenamiento transaccional y no transaccional.

- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.

- Gran portabilidad entre sistemas.[5]

Hay que tener en cuenta que para migrar información de una base de datos para otra no se difiere en la versión del servidor de MySql. Sin embargo para desacoplar una base de datos si es obligatorio que se trabaje sobre un servidor de MySql igual o superior a la versión 5.0.6. Los motivos son los siguientes:

El servidor de MySql contiene una base de datos interna del sistema llamada **INFORMATION_SCHEMA,** que almacena toda la información necesaria acerca de las bases de datos que están embebidas. Esta a su vez contiene varias tablas que guardan un grupo de información específica de las bases de datos. Una de estas tablas es **KEY_COLUMN_USAGE**. La misma describe qué columnas claves tienen restricciones.

En versiones anteriores de las 5.0.6, si la restricción era una clave foránea, solamente se tenía referencia de la columna de la clave foránea, no la columna a la que la clave foránea hace referencia. Ya en la versión 5.0.6 se añadieron a esta tabla tres campos: **REFERENCED_TABLE_SCHEMA, REFERENCED_TABLE_NAME** y **REFERENCED_COLUMN_NAME.** Los mismos contienen información de la base de datos a la cual un índice hace referencia, la tabla dentro de la base de datos a la que se hace referencia y la columna a la cual se referencia respectivamente. Se propone usar un servidor de MySql 5.0.6 dada la utilización del sistema de los atributos antes planteados.

1.3.5 Sistema de Control de Versiones: Subversion 1.4

Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal características es que mantiene la historia de los cambios y modificaciones que se han realizado sobre ellos a lo largo del tiempo. De esta forma, el sistema es capaz de “recordar” las versiones antiguas de los datos, lo que nos permite examinar el histórico de cambios o recuperar versiones anteriores de un fichero, incluso aunque haya sido borrado.

Características de Subversion:

- Mantiene versiones no sólo de archivos, sino también de directorios
- También se mantienen versiones de los metadatos asociados a los directorios.
- Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.

- Atomicidad de las actualizaciones. Una lista de cambios constituye una única transacción o actualización del repositorio. Esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Posibilidad de elegir el protocolo de red. Además de un protocolo propio (svn), puede trabajar sobre http (o https).
- Soporte tanto de ficheros de texto como de binarios.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos. [6]

1.3.6 Entorno de Desarrollo: NetBeans 6.5

NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Además es un producto libre y gratuito sin restricciones de uso. [7]

NetBeans es una aplicación de código abierto ("open source") diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java.

Dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintáxis y por si fuera poco sus funcionalidades son ampliables mediante la instalación de packs.

Cambios recientes de NetBeans IDE

- Soporte completo para 3 lenguajes nuevos: PHP, Groovy y Grails.
- Soporte mejorado para JavaScript, AJAX y Ruby.
- Compilación y desplegamiento automático para aplicaciones Java EE.
- Editor SQL Mejorado.
- Cambiado el método de depuración paso a paso.

1.3.7 Lenguajes de Programación:

Java

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma.

El lenguaje Java se creó con cinco objetivos principales:

- Debería usar la metodología de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Orientado a objetos (OO).

Se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos.

Independiente de plataforma

La independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Es lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, ‘’write once, run everywhere’’.

Sintaxis

La sintaxis de Java se deriva en gran medida de C++. Pero a diferencia de éste, que combina la sintaxis para programación genérica, estructurada y orientada a objetos, Java fue construido desde el principio para ser completamente orientado a objetos. Todo en Java es un objeto (salvo algunas excepciones), y todo en Java reside en alguna clase (recordar que una clase es un molde a partir del cual pueden crearse varios objetos).

C Sharp (C#)

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cuál es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (entre ellos Delphi).

Características del lenguaje C#

- Es auto contenido. Un programa en C# no necesita de ficheros adicionales al propio código fuente, como los ficheros de cabecera (.h) de C++, lo que simplifica la arquitectura de los software desarrollados con C++.

- Es actual. C# incorpora en el propio lenguaje elementos que se han demostrado ser muy útiles para el desarrollo de aplicaciones como el tipo básico decimal que representa valores decimales con 128 bits, lo que le hace adecuado para cálculos financieros y monetarios, incorpora la instrucción foreach, que permite una cómoda iteración por colecciones de datos.

- Encapsulación: además de los modificadores de acceso convencionales: public, private y protected, C# añade el modificador internal, que limita el acceso al proyecto actual.

- C# sólo admite herencia simple.

- Todos los métodos son, por defecto, *sellados*, y los métodos re definibles han de marcarse, obligatoriamente, con el modificador virtual.

C++

C++ es un lenguaje versátil, potente y general. Mantiene ventajas del lenguaje de programación C en cuanto a riquezas en operadores, expresiones, flexibilidad, concisión y eficiencia. Además ha eliminado algunas de las dificultades y limitaciones del C original. [8]

Respecto a la POO, las características más importantes de C++ son:

- Herencia múltiple.
- Sobrecarga de operadores y funciones.
- Derivación.
- Funciones virtuales.
- Plantillas.
- Gestión de excepciones.
- No es multiplataforma. Para lograr aplicaciones que se ejecuten en varios sistemas operativos(SO), se requiere de cierto esfuerzo.
- No presenta una arquitectura estándar de desarrollo orientado a Internet.

Para el desarrollo de la aplicación el lenguaje de programación seleccionado fue Java por las siguientes razones:

- Simple. Elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. La filosofía de programación orientada a objetos es diferente a la programación convencional.
- Familiar. Como la mayoría de los programadores están acostumbrados a programar en “C” o en “C++”, la sintaxis de Java es muy similar al de estos.
- Robusto. El sistema de Java maneja la memoria de la computadora por ti. No te tienes que preocupar por apuntadores, memoria que no se esté utilizando, etc. Java realiza todo esto sin necesidad de que se le indique.

- Seguro. El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.

- Portable. Como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.

- Independiente a la arquitectura. Al compilar un programa en Java, el código resultante es un tipo de código binario conocido como byte code. Éste código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.

- Multithreaded. Un lenguaje que soporta múltiples hilos es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo.

- Interpretado. Java corre en máquina virtual, por lo tanto es interpretado.

- Dinámico. Java no requiere que se compilen todas las clases de un programa para que este funcione.

1.4 Roles y artefactos.

Un rol es una definición abstracta de un conjunto de actividades realizadas y de artefactos obtenidos. Los roles son realizados típicamente por un individuo, o un conjunto de individuos, trabajando juntos en equipo.

En el desarrollo del presente trabajo de diploma de acuerdo a las necesidades actuales del proyecto se desarrollarán los roles de: analista, diseñador e implementador, pertenecientes al grupo de Desarrolladores definido por el Proceso Unificado de Desarrollo (RUP).

1.4.1 Rol Analista

Define el alcance del sistema e identifica a los actores y casos de uso que permiten modelar completa y consistentemente el sistema, y estructura el modelo de casos de uso. [9]

Los artefactos que serán producidos por el analista del sistema son los siguientes:

- Glosario de términos: Define los términos más importantes que se usarán en el proyecto. Es muy útil para alcanzar un consenso entre los desarrolladores relativo a la definición de diversos consensos y para reducir el riesgo de confusiones.

- Actor del sistema: En este artefacto quedan representados mediante uno o más actores cada uno de los tipos de usuarios y de los sistemas externos con los que interactúa el sistema.

- Caso de uso del sistema: Es el artefacto que representa cada forma en que los actores utilizan el sistema.

- Vista de caso de uso: Incluyen los casos de usos que describen funcionalidades críticas, es decir describe los casos de uso significativos para la arquitectura.

1.4.2 Rol Diseñador

En la metodología RUP el diseñador es el responsable de diseñar una parte del sistema cumpliendo con las restricciones de los requerimientos, arquitectura y proceso de desarrollo del proyecto, identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos de diseño. Debe asegurarse que el diseño es consistente con la arquitectura del software y que está detallado al punto que se puede proceder con la implementación. El diseñador debe tener un conocimiento amplio acerca de los requerimientos del sistema, la arquitectura, las técnicas de diseño de software, así como de las tecnologías con las cuales se implementará el sistema. [10]

Los artefactos realizados por el diseñador que se obtendrán en el presente trabajo son:

- Realización de casos de uso del diseño: Es una colaboración en el modelo de diseño que describe como se realiza un caso de uso específico, y como se ejecuta en términos de casos de uso del diseño. Una realización de caso de uso del diseño proporciona una traza directa a una realización de caso de uso del análisis en el modelo de análisis.

- Clases del diseño: Una clase es una descripción de un conjunto de objetos que comparten las mismas responsabilidades, las relaciones, las operaciones, atributos, y la semántica.

- Paquetes de diseño: Es una colección de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes que estén de alguna forma relacionados. Es usado para estructurar el modelo de diseño dividiéndolo en partes más pequeñas.

- Subsistema de diseño: Es una parte del sistema que encapsula el comportamiento, incluye interfaces y paquetes.

1.4.3 Rol Implementador

En la metodología RUP el rol implementador es responsable de desarrollar y de probar componentes de acuerdo con los estándares adoptados del proyecto para la integración en subsistemas. El implementador es también responsable de desarrollar y probar los componentes de prueba y los subsistemas correspondientes. [10]

Los artefactos que genera el rol implementador que se obtendrán en el presente trabajo de diploma son:

- Elementos de implementación: Los elementos de implementación son la parte física de la implementación, incluyen los archivos y directorios. Incluyen ficheros de código (fuentes, binarios o ejecutables), de datos y de documentación como son ficheros de ayuda online.

1.5 Patrones de casos de uso, diseño y de arquitectura.

1.5.1 Patrones de casos de uso

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento. Dado un contexto y un problema a resolver, estas técnicas han mostrado ser la solución adoptada en la comunidad del desarrollo de software. Se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática. Estos patrones se enfocan hacia el diseño y las técnicas utilizadas en modelos de alta calidad, y no en cómo modelar usos específicos.

Utilizando estos patrones, arquitectos, analistas, e implementadores pueden lograr mejores resultados de forma más rápida. [11]

1.5.2 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Pueden incrementar o disminuir la capacidad de comprensión de un diseño o de una investigación, por lo que se consideran una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra característica es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

1.5.3 Patrones de Arquitectura.

Los patrones arquitectónicos son los que definen la estructura de un sistema de software, los cuales a su vez se componen de subsistemas con sus responsabilidades. También tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

Los patrones arquitectónicos permiten:

- Un intento de formalizar la forma en que se comunica y rehúsa la experiencia de diseño.
- Capturan la experiencia probada de diseño de software.
- Describen problemas recurrentes que surgen en determinados contextos.
- Describen esquemas de soluciones probados.
- Herramientas para los no expertos.
- Un paso más hacia la ingeniería de software:
    - Permite que gente común haga cosas que antes requerían virtuosismo.

Conclusiones

Para realizar el diseño e implementación de la Herramienta para Desacoplar e Integrar Base de Datos con mayor calidad, a través de los roles de Analista, Diseñador e Implementador, se emplea la metodología RUP y la notación UML. Como herramienta CASE Visual Paradigm, para el control de versiones Subversion, como entorno de desarrollo NetBeans, MySQL como gestor de base de datos y para implementar la herramienta el lenguaje Java.

Capítulo 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

En el presente capítulo se describe la solución propuesta utilizando los componentes del modelo del dominio de la metodología RUP. Se definen además los requisitos funcionales y no funcionales, representados a través del diagrama de casos de uso del sistema y las descripciones textuales de los casos de uso del sistema para una mejor comprensión del funcionamiento de la aplicación que se diseñará.

2.1 Modelo de dominio

2.1.1 Definición de los conceptos principales

Un Modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los principales conceptos. Muchos de los objetos o clases pueden obtenerse de una especificación de requisitos.

La modelación del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes en el sistema.

En el sistema a desarrollar ha sido difícil identificar las especificidades de los procesos del negocio (generar esquema de datos y migrar base de datos), así como las fronteras y los actores del mismo, razón por la cual se hace necesario la confección de un modelo de dominio, para el cual se identificaron las siguientes entidades y conceptos (objetos):

- Base de datos origen: Contendrá todo el esquema de datos, se visualizarán y seleccionarán las tablas para conformar el nuevo esquema de datos y se establecerán las relaciones.
- Script: Es el resultado final del proceso de generación de un esquema de datos.
- Flujo de migración: Se determina cual de las tablas migran primero, validando que la llave primaria no se repita.
- Base de Datos Destino: Es el resultado de la integración de la información de dos bases de datos.

2.1.2 Representación del modelo del dominio

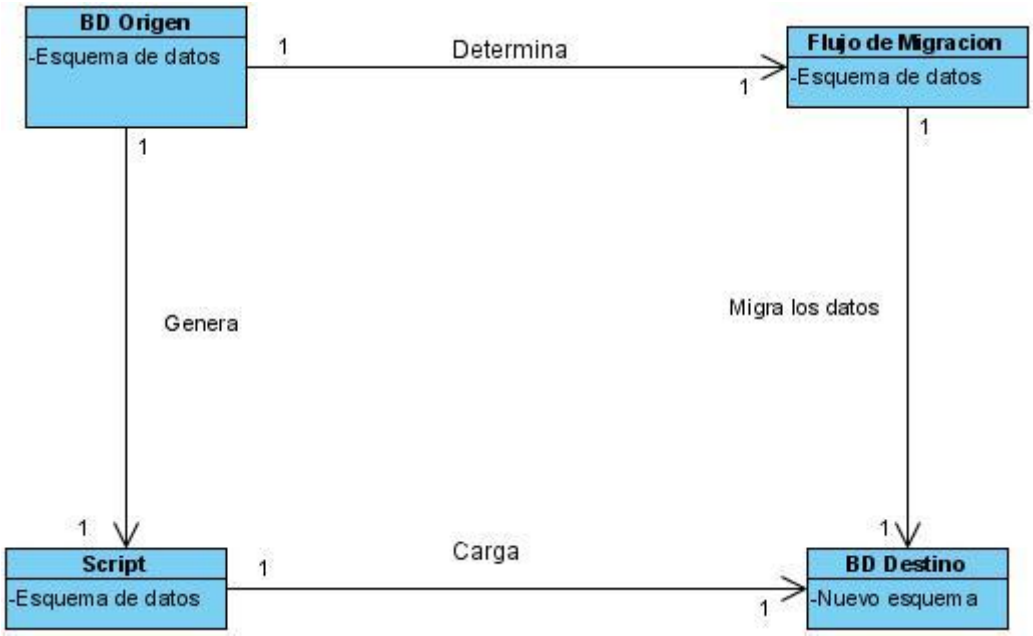


Fig. 1 Diagrama de clases del dominio.

2.2 Especificación de los Requisitos.

En el flujo de trabajo de requerimientos se realizó un levantamiento de 12 requisitos funcionales y 18 requisitos no funcionales.

2.2.1 Requisitos Funcionales.

RF 1 Gestionar conexión a base de datos

RF 1.1 Conectar a una base de datos.

RF 1.2 Mostrar conexiones.

RF 1.3 Cerrar conexiones.

RF 1.4 Visualizar datos de una tabla.

RF 2 Generar esquema de datos

RF 2.1 Visualizar las tablas.

RF 2.2 Seleccionar las tablas para conformar el nuevo esquema.

RF 2.3 Mostrar reporte de las tablas que se van a generar.

RF 2.4 Generar el script del esquema definido.

RF 3 Migrar base de datos

RF 3.1 Determinar el flujo de migración.

RF 3.2 Guardar configuración de una base de datos.

RF 3.3 Cargar configuración de una base de datos.

RF 3.4 Migrar datos.

2.2.2 Requisitos no Funcionales.

- **Requerimiento de Software**

RNF1. Sistema Operativo Windows o Linux.

RNF2. Servidor de Base Datos, MySql, una versión igual o superior a la 5.0.6.

RNF3. Maquina virtual de Java versión 1.3 o superior.

- **Requerimiento de Hardware**

RNF4. Una computadora que tenga instalada la máquina virtual de Java.

RNF5. 128 Megabytes (MB) de memoria RAM o más.

- **Restricciones en el diseño y la implementación**

RNF6. Lenguaje de programación utilizado en la implementación Java.

RNF7. Uso de herramienta CASE Visual Paradigm.

RNF8. Uso de la herramienta controladora de versiones Subversion 1.4.

RNF9. Uso del servidor de MySql 5.0.6.

RNF10. Uso de las librerías JDBC de MySQL para Java, y xstream para cargar y salvar ficheros de datos.

- **Requisitos de Rendimiento**

RNF11. Los tiempos de respuestas deben ser generalmente rápidos al igual que la velocidad de procesamiento de la información.

- **Requerimientos de Seguridad.**

RNF12. La información manejada solamente va a estar disponibles para administradores de las bases de datos.

RNF13. Se les garantizará a los usuarios autorizados el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento determinado.

- **Requerimientos de Usabilidad**

RNF14. El software debe ser usado por usuarios que conozcan el funcionamiento de las bases de datos con las que trabajan y un mínimo conocimiento del uso de la PC.

RNF15. Será útil solamente por los usuarios que tengan privilegios de administración de una base de datos.

- **Requisito de Apariencia o interfaz externa.**

RNF16. El sistema debe tener una interfaz fácil de usar y amigable para que pueda ser utilizada sin mucho entrenamiento por el usuario.

RNF17. Empleo de imágenes y colores adecuados para el sistema.

- **Requisito de Legales**

RNF18. Las herramientas y las tecnologías en que está basado el sistema deberán cumplir con las licencias de software GNU/GPL.

2.3 Justificación del actor del sistema.

Actor	Justificación
Administrador	Conectar a una base de datos, mostrar conexiones, mostrar datos de las tablas, cerrar conexiones, seleccionar las tablas para conformar el nuevo esquema, generar el script del esquema definido, determinar el flujo de migración y sincronizas dos bases de datos.

2.4 Patrón de casos de uso utilizado.

Extensión concreta

Es una relación de un caso de uso de extensión a un caso de uso base, que especifica cómo el comportamiento definido por el caso de uso de extensión puede insertarse dentro del comportamiento definido por el caso de uso base. La relación contiene una condición y referencia una secuencia de puntos de extensión en el caso de uso base. Una vez que una instancia del caso de uso ejecuta un comportamiento referenciado por el primer punto de extensión de la relación, la condición es evaluada. Si se cumple, la secuencia de la instancia se extiende para incluir la secuencia del caso de uso extensión. Las diferentes partes del caso de uso extensión son insertadas en los lugares definidos por la secuencia de puntos de extensión de la relación: una parte en cada punto de extensión referenciado.

Una relación de este tipo (extend) se emplea para mostrar alguna de las siguientes situaciones:

- Funciones opcionales
- Funciones complementarias que pueden ejecutarse en base a la selección del actor.

2.5 Definición de los casos de uso del sistema.

2.5.1 Diagrama de casos de uso del sistema

Para la realización del diseño se deben conocer las características principales del sistema especificando las diferentes acciones que el sistema debe permitir y los actores que intervienen en las mismas. Las relaciones que se establecen entre estos dos elementos se reflejan en el siguiente diagrama de casos de uso del sistema.

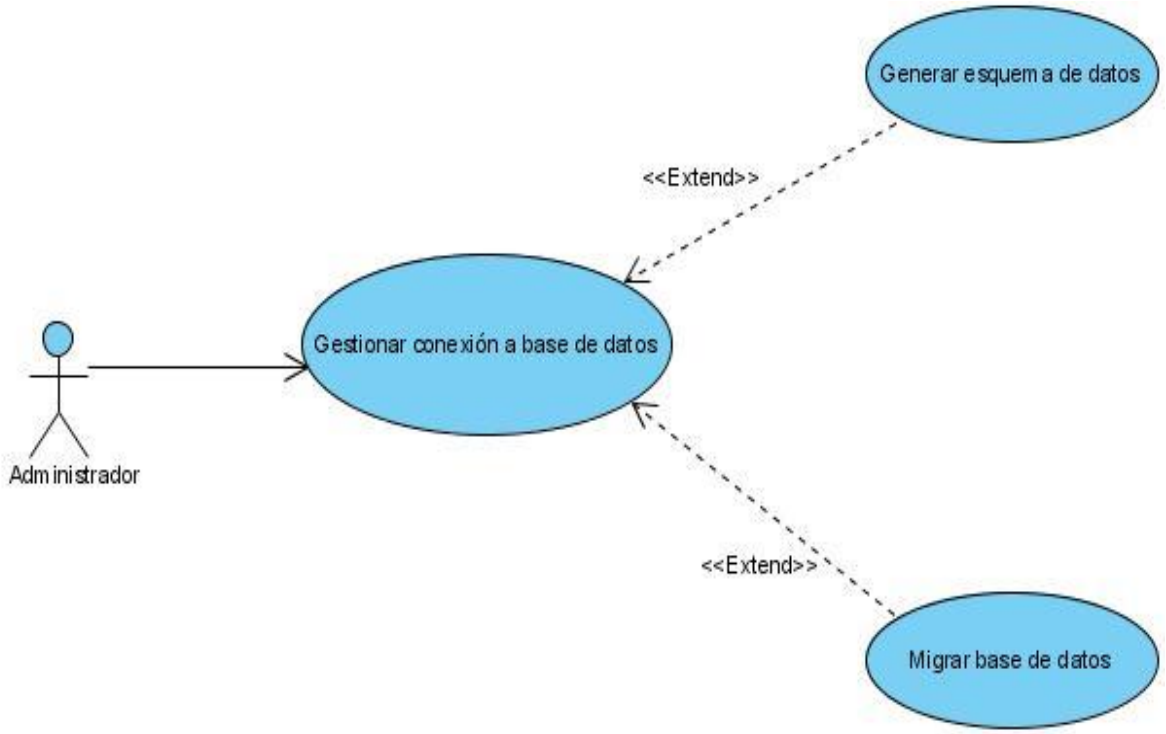


Fig.2 Diagrama de casos de uso del sistema.

2.5.2 Listado de los casos de uso del sistema.

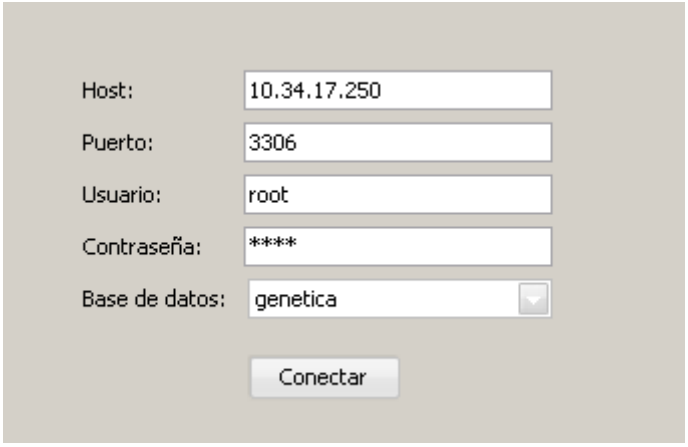
CU Gestionar conexión a base de datos.

Caso de Uso:	Gestionar conexión a base de datos

Actores:	Administrador
Resumen:	El caso de uso inicia cuando el administrador de una base de datos decide realizar una conexión a la misma con el fin de mostrar las conexiones realizadas ó visualizar el contenido de una tabla perteneciente a una conexión específica. El caso de uso termina cuando se cierra dicha conexión.
Precondiciones:	El Administrador debe haberse conectado a la base de datos satisfactoriamente.
Referencias	R1
Prioridad	Alta.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. El administrador desea realizar una de las opciones: • Realizar conexión a una base de datos. • Mostrar las conexiones realizadas. • Visualizar el contenido de una tabla perteneciente a una conexión. • Cerrar conexiones.	1.1. El sistema ejecuta alguna de las siguientes acciones. a) Si se decide “Realizar conexión a una base de datos” ir a la sección Realizar conexión. b) Si se deciden “Mostrar las conexiones realizadas” o “Cerrar conexiones” ir a la sección Mostrar conexiones. c) Si se deciden “Visualizar el contenido de una tabla” ir a la sección Visualizar contenido de una Tabla.

Sección “Realizar conexión a una base de datos”	
1-El administrador selecciona la opción Realizar conexión a una base de datos.	1.1-El sistema muestra una interfaz donde solicita los datos necesarios para realizar la conexión a la base de datos. • Host. • Puerto. • Usuario. • Contraseña.
2-El administrador introduce los datos necesarios y se selecciona la opción “Siguiente”.	2.1- El sistema verifica los datos y realiza una conexión al servidor y muestra las bases de datos que se encuentran disponibles.
3-El administrador escoge la base de datos a la cual se desea conectar.	3.1- El sistema realiza la conexión. 3.2- El sistema agrega la conexión con sus tablas en el árbol de conexiones de la interfaz.
Flujo alternativo	
	2.1. a- Si los datos introducidos por el usuario no son correctos el sistema emite un mensaje para que llene los campos obligatorios y regresa a la acción 2 del flujo normal de eventos.
Prototipo de Interfaz	

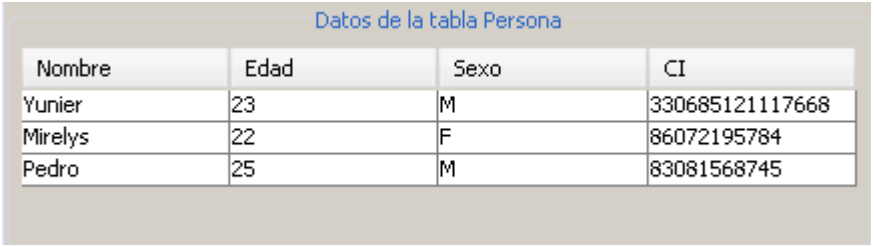


Sección "Mostrar conexiones"	
1-El administrador selecciona la opción Mostrar las conexiones realizadas o la opción Cerrar conexiones.	1.1- El sistema muestra una interfaz con la información necesaria de cada conexión, además le brinda la posibilidad de seleccionar las conexiones que desea cerrar.
2- Si decide cerrar alguna conexión pulsa la opción Cerrar Conexión.	2.1- El sistema cierra las conexiones e informa al administrador.

Prototipo de Interfaz

Servidor	Base de datos	Host	Puerto
MySQL	genetica	10.34.17.250	3306
MySQL	minppal	10.7.9.200	3389
MySQL	alasLIPO	10.7.19.200	3389
MySQL	bd_test	localhost	

Sección "Visualizar contenido de una Tabla"	
1- El administrador selecciona la opción Visualizar el contenido de una tabla.	1.1- El sistema muestra una interfaz con todos los datos de la tabla seleccionada.

Prototipo de Interfaz 
Poscondiciones — Conectarse a una base de datos.

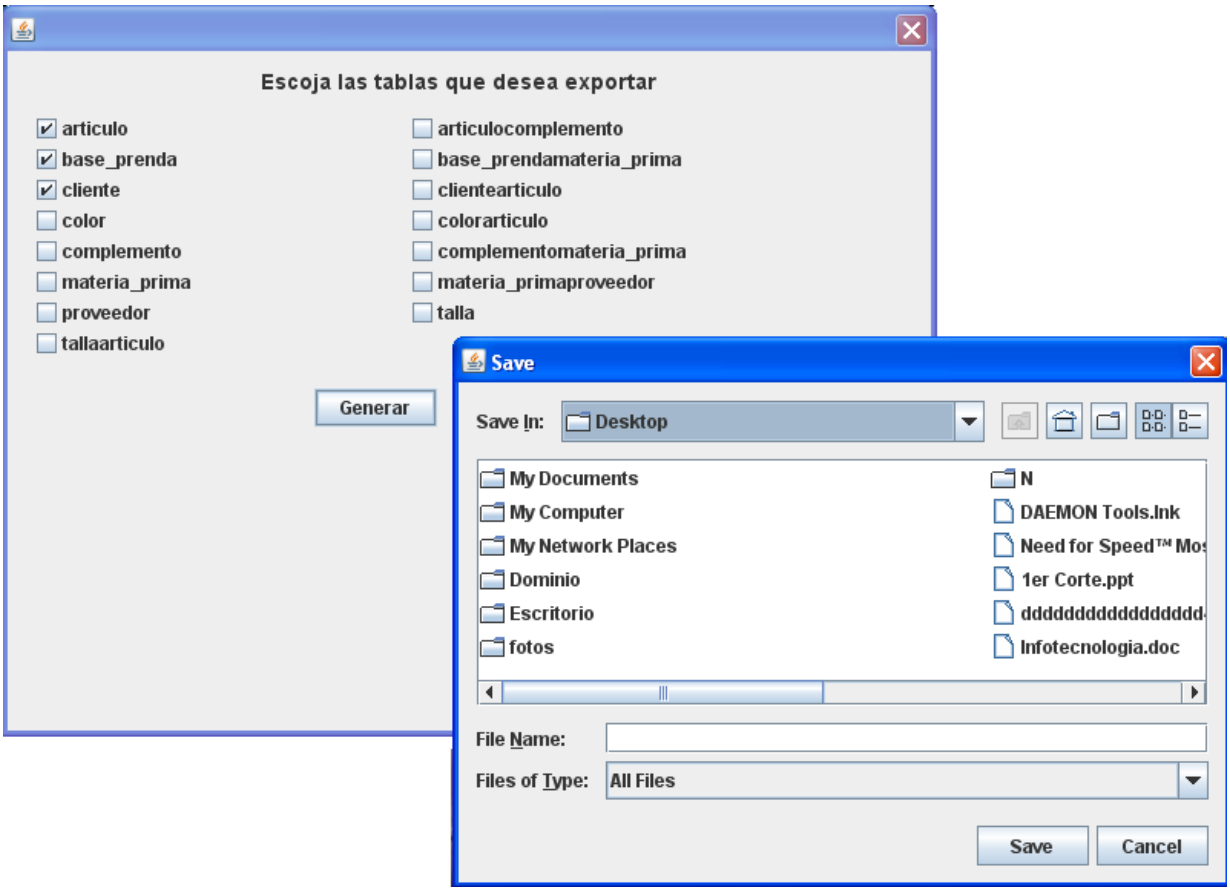
CU Generar esquema de datos

Caso de Uso:	Generar esquema de datos
Actores:	Administrador
Resumen:	El caso de uso inicia cuando el administrador de una base de datos decide generar una subbase de datos a partir de una base de datos madre.
Precondiciones:	Debe haberse conectado previamente a la base de datos.
Referencias	R2
Prioridad	Alta.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. El administrador selecciona la opción generar	1.1- El sistema muestra una interfaz con las

una sub base de datos.	bases de datos que están disponibles.
2-El usuario selecciona una base de datos.	2.1- El sistema muestra una interfaz con las tablas de la base de datos.
3- El administrador selecciona las tablas que contendrá el nuevo esquema.	3.1- El sistema verifica que se haya seleccionado al menos una tabla.
4- El administrador desea realizar una de las opciones: • Mostrar resultados con las tablas que se van a generar en el esquema. • Generar el esquema de las tablas seleccionadas.	4.1-. El sistema ejecuta alguna de las siguientes acciones. a) Si se decide “Mostrar resultados” ir a la sección Mostrar resultados. b) Si se deciden “Generar el esquema de las tablas seleccionadas” ir a la sección Generar esquema.
Sección “Mostrar resultados”	
1- El administrador selecciona la opción Mostrar resultados.	1.1- El sistema muestra una interfaz con todos los nombres de las tablas que se van a generar en el esquema de datos.
Sección “Generar esquema”	
1- El administrador selecciona la opción Generar esquema.	1.1- El sistema muestra una interfaz para que el usuario seleccione donde desea guardar el archivo.
2- El administrador indica nombre del archivo y lugar donde se guardará.	2.1- El sistema salva el fichero.

Flujo alternativo	
	2.1.a- Si el servidor de base de datos se encuentra en otra computadora y se cae la conexión mientras se está generando el esquema de datos se le informa al usuario.

Prototipo de Interfaz

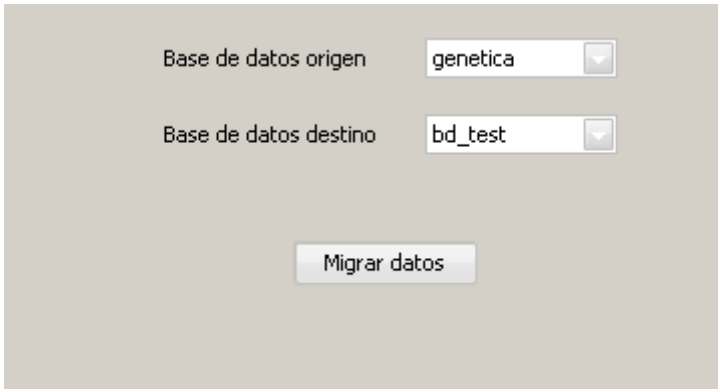


Poscondiciones	Generar base de datos

CU Migrar base de datos

Caso de Uso:	Migrar base de datos
Actores:	Administrador
Resumen:	El caso de uso inicia cuando se decide migrar los datos de una base de datos hacia otra.
Precondiciones:	Deben haberse realizado al menos dos conexiones a bases de datos.
Referencias	R3
Prioridad	Alta.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1-El administrador selecciona la opción migrar una base de datos.	1.1- El sistema muestra una interfaz donde solicita seleccionar las bases de datos.
2-El administrador selecciona una base de datos origen y una base de datos destino.	2.1- El sistema muestra las tablas de la base de datos y brinda la opción de configurar la forma en que se va a migrar la información.
3-El administrador determina el flujo de migración. • Nunca sobre escribir. • Sobre escribir siempre. • Personalizar: Se configura la forma en que	

se van a migrar los datos de cada tabla.	
4- El administrador escoge la opción de: • Guardar configuración. • Cargar configuración. • Migrar datos.	4.1- El sistema ejecuta alguna de las siguientes acciones. a) Si desea Guardar configuración ir a la sección “Guardar configuración”. b) Si desea Cargar configuración ir a la sesión “Cargar configuración”. c) Si se desea Migrar datos ir a la sesión “Migrar datos”.
Prototipo de Interfaz 	
Sección “Guardar configuración”	
1- El administrador selecciona la opción Guardar configuración.	1.1- El sistema muestra una ventana donde se especifica el nombre y el lugar donde se guardará el fichero.
2- El administrador indica nombre del archivo y lugar donde se guardará.	2.1- El sistema salva el fichero.

Sección “Cargar configuración”	
1- El administrador selecciona la opción Cargar configuración	1.1- El sistema muestra una ventana donde se escoge el archivo a ser cargado.
2- El administrador selecciona el archivo a cargar.	2.1- El sistema valida los datos cargados y si se corresponden con la base de datos que esta seleccionada le aplica la configuración.
Sección “Migrar datos”	
1- El administrador selecciona la opción Migrar datos.	1.1- El sistema realiza la migración de los datos de la base de datos origen a la base de datos destino según la configuración que se realizó.
Flujo alterno	
	1.1. a- Si el servidor de base de datos se encuentra en otra computadora y se cae la conexión mientras se están sincronizando las bases de datos se le informa al usuario. 1.2. b- Si ocurre algún problema de referencias entre tablas mientras se están sincronizando las bases de datos se le informa al usuario y regresa a la acción 1 del flujo normal de eventos.
Poscondiciones	Migras los datos de una base de datos hacia otra.

Conclusiones

En este capítulo se definieron los principales procesos que se llevan a cabo en el desarrollo de la herramienta informática, los cuales fueron representados a través de la realización del modelo de dominio. Se identificaron un total de 12 requisitos funcionales, agrupados en 3 casos de uso que se representaron en el diagrama de casos de uso del sistema. Además se hizo un levantamiento que arrojó 18 requisitos no funcionales. Y para una mejor comprensión del funcionamiento del sistema, se realizaron las descripciones textuales de los casos de uso del sistema.

CAPÍTULO 3: DISEÑO DEL SISTEMA

Introducción

Este capítulo aborda todo lo referido al flujo de trabajo de análisis y diseño. Se representarán de dicho flujo de trabajo la realización de los CU formada por los diagramas de clases del diseño y los diagramas de secuencia, así como la descripción general de las tablas más importantes y el modelo de despliegue.

3.1 Modelo de Diseño

Para poder realizar un sistema que soporte todos los requerimientos especificados, tanto funcionales como no funcionales, se necesita modelarlo, lo cual se hace a través del diseño. El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar.

Propósitos del flujo de trabajo diseño:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia además de las tecnologías de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.
- Capturar las interfaces entre los subsistemas antes en el ciclo de vida del software, lo cual es muy útil cuando se crean interfaces como elementos de sincronización entre diferentes equipos de desarrollo.

3.1.1 Descripción de las clases más importantes del diseño.

Nombre: conexionControlador	
Tipo de clase : Controladora	
Responsabilidades: Controla la información necesaria de las conexiones a las bases de datos	
Nombre:	**setConexionTemporal(String usuario, String contrasena, String host, int puerto)**
Descripción:	Se crea una conexión temporal al servidor, con el objetivo de escoger una de las bases de datos que se encuentran disponibles.
Nombre:	**setConexionTemporal()**
Descripción:	Este método cierra una conexión temporal que ha sido desecha.
Nombre:	**getConexionTemporal()**
Descripción:	Devuelve la conexión temporal con el objetivo de seleccionar una base de datos.
Nombre:	**addConexion(String nombreBD)**
Descripción:	Este método obtiene todos los atributos de la conexión temporal y le adiciona el nombre de la base de datos seleccionada y crea una conexión real a una base de datos y la adiciona a la lista de conexiones.
Nombre:	**deleteConexion(int index)**
Descripción:	Cierra y elimina de la lista de conexiones la conexión que se encuentra en la posición especificada.
Nombre:	**deleteAllConexiones()**

Descripción:	Cierra todas las conexiones
Nombre:	**obtenerNombresBasesDatos()**
Descripción:	Obtiene el nombre de todas las bases de datos de una conexion a un servidor.
Nombre:	**getConexiones()**
Descripción:	Devuelve la lista de conexiones
Nombre:	**setConexiones(List<conexion> conexiones)**
Descripción	Cambia la lista de conexiones por la pasada por parámetros
Nombre:	**buscarConexion(String urlConexion)**
Descripción:	Devuelve una conexión cuya url sea la pasada por parámetros.
Nombre:	**posicionConexion(String urlConexion)**
Descripción:	Devuelve la posición en la que se encuentra una conexión en la lista de conexiones.
Nombre:	**nombreTablasConexion(int index)**
Descripción:	Retorna una lista con los nombres de las tablas de la conexión que se encuentre en la posición dada en la lista de conexiones.
Nombre:	**getTablasConexiones()**
Descripción:	Devuelve una lista que contiene en cada posición la lista de tablas de cada conexión.
Nombre:	**setTablasConexiones(List<List<tabla>> tablasConexiones)**
Descripción:	Cambia la lista que contiene las listas con las tablas de cada conexión por la pasada por parámetros

Nombre:	**bdCompatibles(int index)**
Descripción	Pasada la posición de una base de datos se busca en la lista de conexiones las bases de datos que son compatibles con ella, es decir con las cuales se puede establecer la migración de datos.
Nombre:	**obtenerDatosTabla(String url, String nombreTabla)**
Descripción:	Se obtiene los datos que contiene una tabla.
Nombre:	**generarBD(List<String> nombreTablas, List<String> nombreTablasNomencladoras, File archivo, int index)**
Descripción:	Llama al método **generarSalvarBD** que se encuentra en la clase **salvarSincronizarBDControlador.**
Nombre:	**acoplarBD(String urlOrigen, String urlDestino, baseDatosConfig base, String criterio)**
Descripción:	Llama al método **acoplarDatos** que se encuentra en la clase **salvarSincronizarBDControlador**.
Nombre:	**obtenerTablasReporte(List<String> nombreTabla, List<String> nombreTablasNomencladoras, int index)**
Descripción:	Se devuelven los nombres de las tablas con las cuáles se va a generar el esquema de la base de datos.

Nombre: salvarSincronizarBDControlador	
Tipo de clase : Controladora	
Responsabilidades: Esta es la clase que contiene toda la lógica del negocio necesaria para generar el esquema de una base de datos y migrar la información de una base de datos hacia otra.	
Nombre:	**obtenerTabla(String nombre, ArrayList<tabla> tablasConexion)**
Descripción:	Se obtiene de la lista de tablas la que tenga como nombre el especificado por parámetros.
Nombre:	**obtenerNombresFinales(List<String> nameTables, List<String> nameTablesNomencladoras, List<tabla> tablasConexion)**
Descripción:	Se obtienen los nombres de todas las tablas que van a ser guardadas en un fichero.
Nombre:	**salvarBD (List<String> nombres, List<String> nombreNomencladores, File archivo, conexion cone, List<tabla> tablasConexion)**
Descripción:	En este método es donde se realiza todo el proceso de guardar el esquema de una base de datos en un fichero de texto.
Nombre:	**ordenarPrioridad(List<String> nombresTablas, List<tabla> tablasConexion)**
Descripción:	Se establece una prioridad de acuerdo al flujo de migración de las tablas para que no ocurran conflictos a la hora de cargar el fichero en un servidor.
Nombre:	**buscarReferencia(int pos, List<List<String>> listaReferenciasTabla_A_AutoIncrement)**
Descripción:	Se obtiene el nombre de la tabla a la cual un atributo de otra tabla hace

	referencia, o es clave foránea.
Nombre:	**valorNuevaReferencia(int valorViejo, List<List<String>> listaReferenciasActualizar, String nombreTabla)**
Descripción:	Retorna el nuevo valor que toma un atributo autoincremental en una tabla luego de ser insertada una fila completa.
Nombre:	**actualizarListaAtributosAutoIncrementales(tabla tabla, int posicionAutoIncrement, int j, List<List<String>> datos, IService serviceDestino, String nombreTabla, String valorViejo, List<List<String>> listaReferenciasActualizar)**
Descripción:	Cuando un campo de una tabla es de tipo autoincremental, al ser insertada la fila a la cual pertenece en otra base de datos, este atributo puede cambiar de valor, por lo que se guarda el valor que tenía y el nuevo valor que le es asignado para no perder la referencia que puedan tener a él otras tablas.
Nombre:	**acoplarDatos(conexion origen, conexion destino, List<tabla> tablasBDOrigen, baseDatosConfig base, String criterio)**
Descripción:	Este método es el que migra los datos de una base de datos hacia otra.

3.2 Diagramas de clases del diseño

El diseño de la aplicación esta básicamente dividido en cuatro paquetes:

AccDatos: Contiene todas las clases necesarias para acceder y gestionar la información de una base de datos. Está dividido en tres partes, el subpaquete entidades, en el que se almacena la clase encargada de realizar la conexión a un servidor de bases de datos. Otro subpaquete es el dao, que además contiene un subpaquete daoImpl, en el que se realiza la implementación de la clase sqlDaoImpl que es de tipo IsqlDao. De igual manera el subpaquete servicios contiene un subpaquete servImpl, en el cual se realiza la implementación de la clase sqlServiceImpl que es de tipo IsqlService.

Controladores: Contiene las clases de la lógica del negocio, para acceder y gestionar la información de un servidor de bases de datos.

Presentación: Contiene las clases que le presentan el sistema al usuario.

XML Configuración: En este paquete es donde se modela la forma en la que se desea guardar o recuperar de un XML la configuración de las tablas de una base de datos.

A continuación se expone el diagrama de clases del diseño, además de los diagramas correspondientes a los casos de uso Gestionar Conexión a Base de Datos, Generar Esquema de Datos y Migrar Base de Datos en el orden correspondiente. (Para una mejor visualización de cada diagrama remitirse al expediente de proyecto).

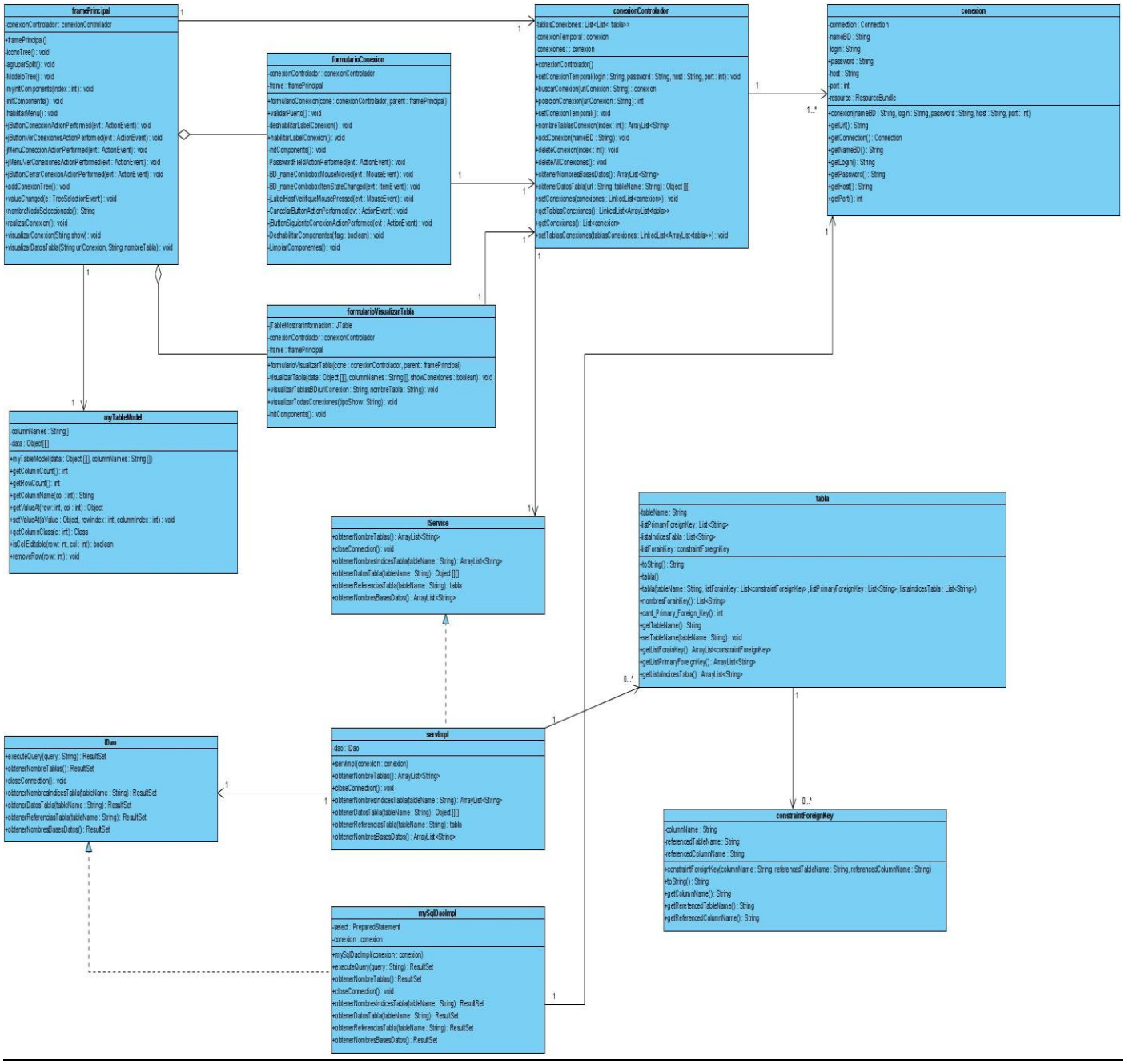


Fig. 3 Diagrama de clases del diseño del Caso de Uso Gestionar Conexión a Base de Datos.

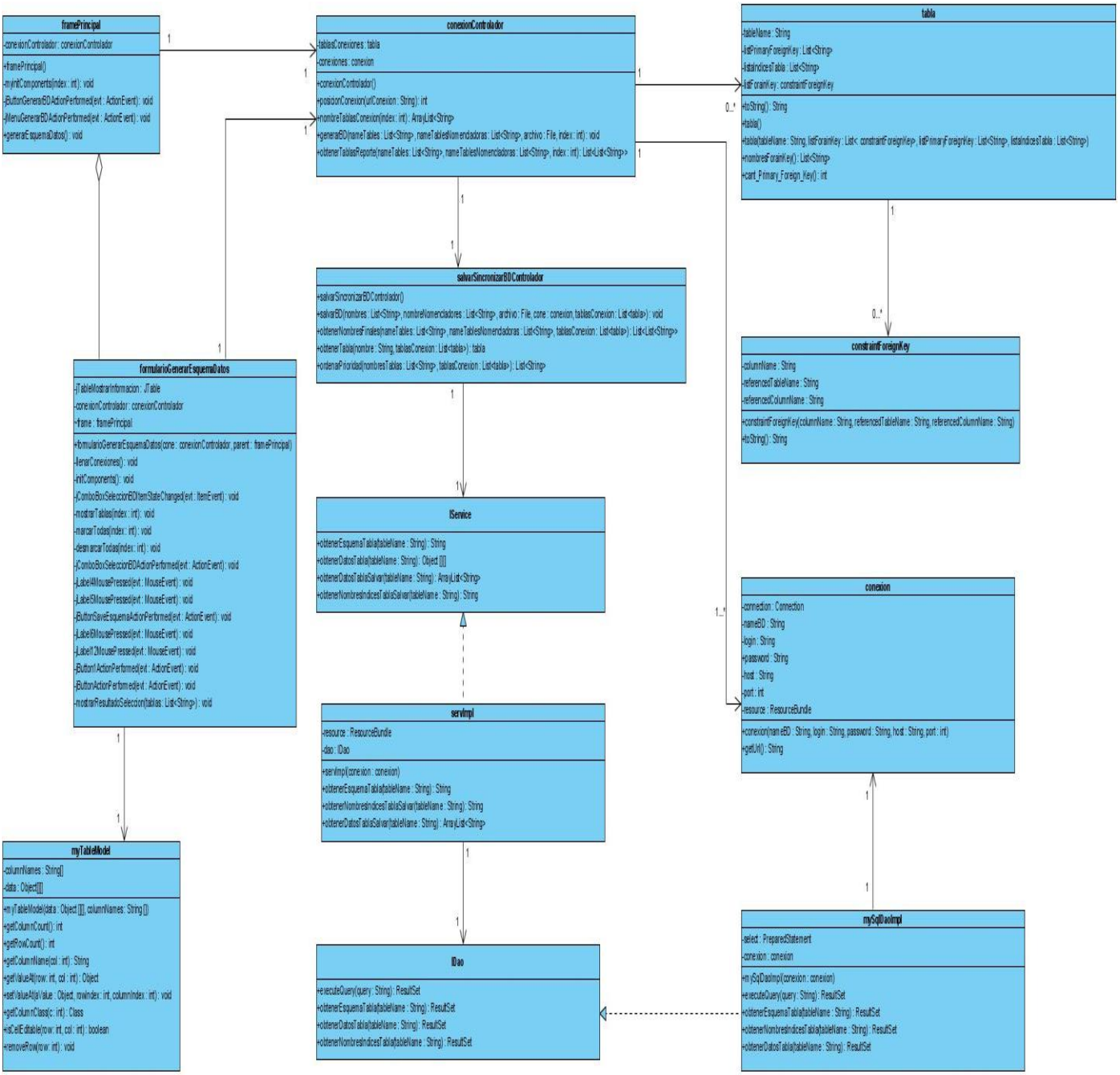


Fig. 4 Diagrama de clases del diseño del Caso de Uso Generar esquema de datos.

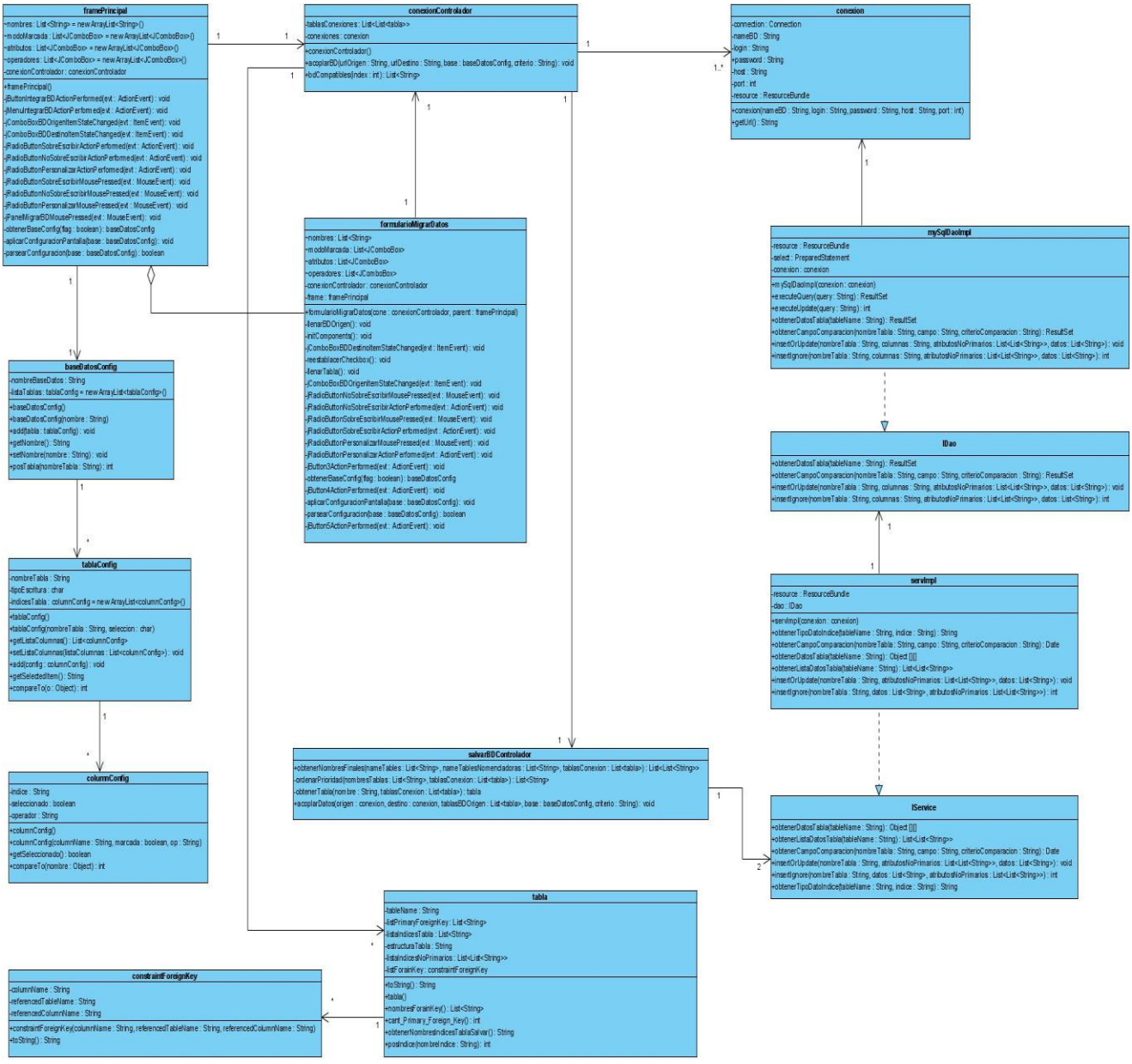


Fig.5 Diagrama de clases del diseño del Caso de Uso Migrar base de datos.

3.3 Diagramas de Interacción: Secuencia.

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. Un diagrama de interacción consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Los diagramas de secuencia destacan el orden temporal de los mensajes. [10]

Diagramas de secuencia

Diagrama de secuencia que representa el CU Gestionar conexión a base de datos: Escenario Realizar Conexión. Aquí se evidencia la iteración entre las clases que intervienen cuando se desea conectar a una base de datos. Es bueno destacar que lo primero a realizar para establecer la conexión es tener los permisos pertinentes, especificar el host y el puerto. Luego el administrador determina el gestor de base de datos al que se conectará. Una vez establecida la conexión el administrador escoge el nombre de la base de datos que desea. (Para una mejor visualización del diagrama remitirse al expediente de proyecto).

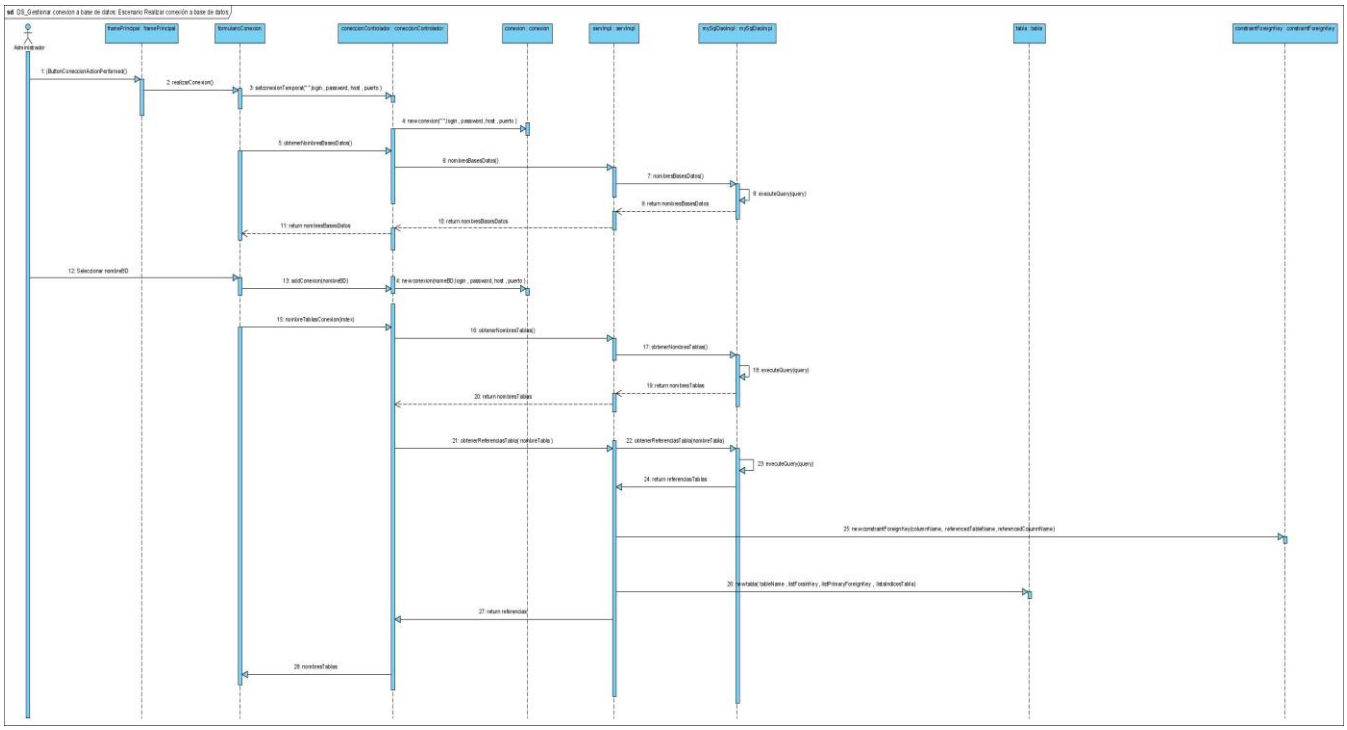


Fig. 6 Diagrama de secuencia caso de uso Gestionar Conexión a Base de Datos: Escenario Realizar Conexión a Base de Datos.

El segundo diagrama de secuencia pertenece al mismo caso de uso: Escenario Mostrar Conexiones. Una vez realizada la conexión a la base de datos, el administrador puede escoger la opción mostrar conexiones y se le mostrarán todas las conexiones realizadas, una vez mostradas todas las conexiones el administrador tiene la opción de cerrar una o varias conexiones. (Para una mejor visualización del diagrama remitirse al expediente de proyecto).

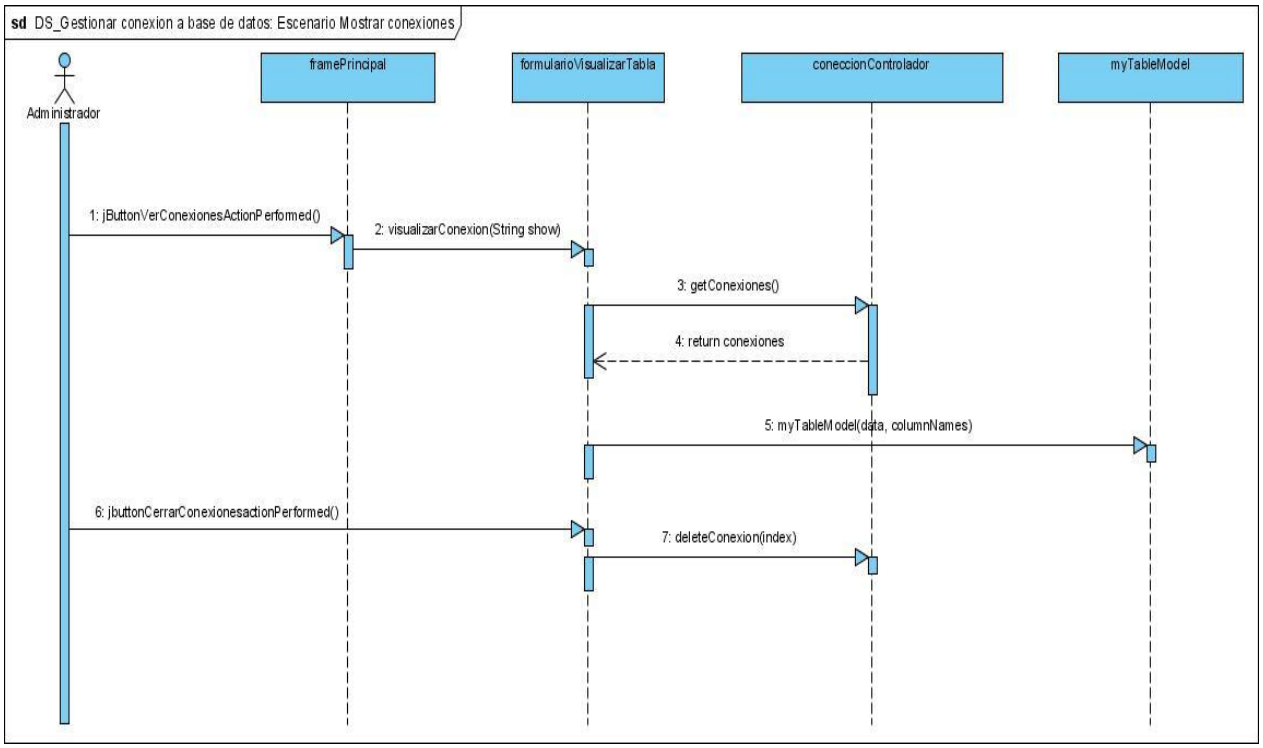


Fig. 7 Diagrama de secuencia caso de uso Gestionar conexión a base de datos: Escenario Mostrar y Cerrar Conexiones.

El siguiente diagrama de secuencia pertenece al mismo caso de uso: Escenario Visualizar contenido de una tabla. Una vez realizada la conexión, se muestran todas las tablas de dicha base de datos el administrador puede escoger la opción visualizar el contenido de una tabla y se le mostrará todo el contenido de la misma. (Para una mejor visualización del diagrama remitirse al expediente de proyecto).

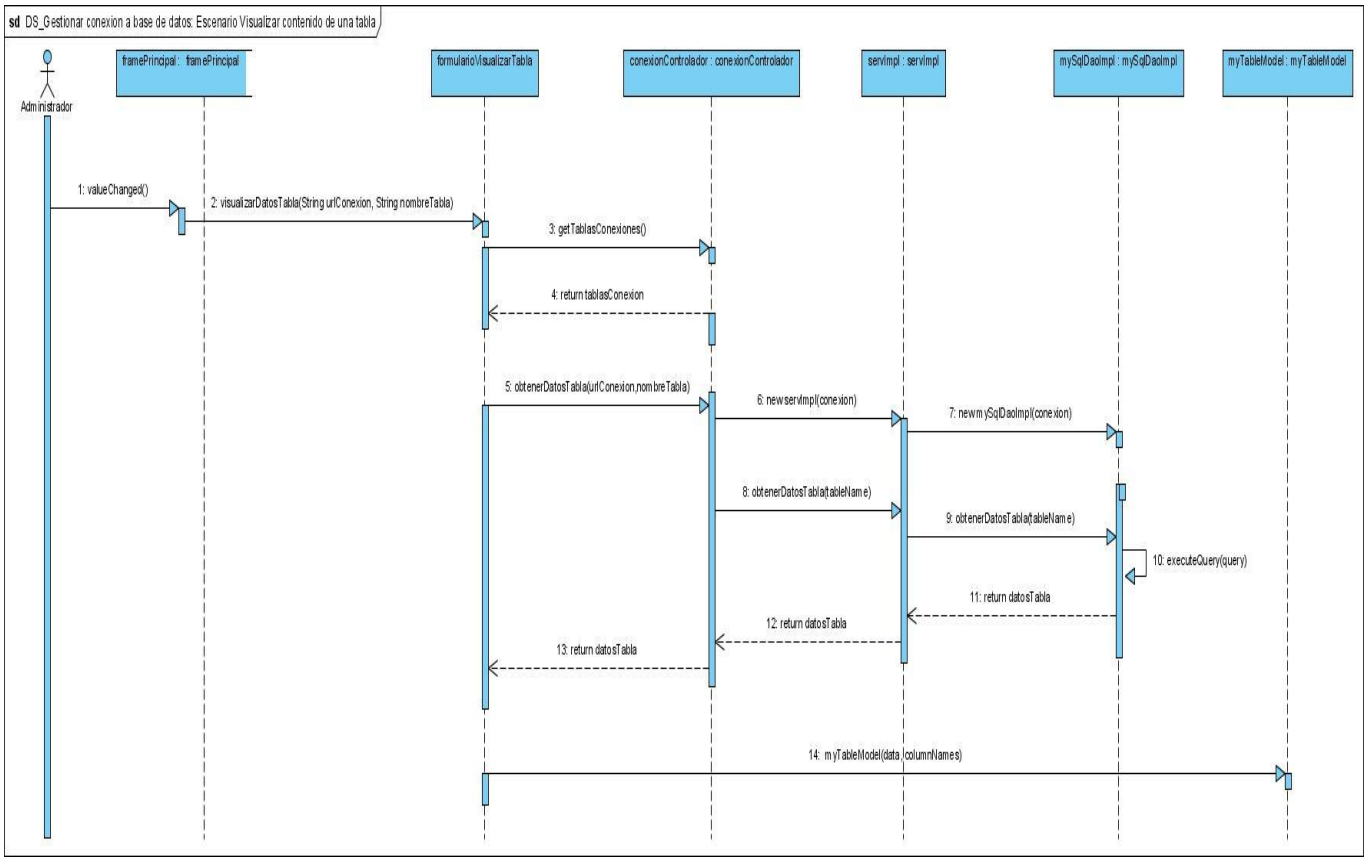


Fig. 8 Diagrama de secuencia caso de uso Gestionar conexión a base de datos: Escenario Visualizar contenido de una tabla.

El próximo diagrama de secuencia pertenece al caso de uso Generar esquema de datos. Aquí se representa la iteración entre las clases que intervienen cuando se desea generar un nuevo esquema. Especificar que para generar el esquema es necesario haberse conectado previamente a la base de datos. El usuario escoge la opción de generar un nuevo esquema, aquí se le muestran las bases de datos disponibles, una vez escogida una de ellas, se le muestran todas las tablas que contiene la misma, el usuario escoge las tablas con las que generará el nuevo esquema y luego determina la dirección donde guardará el fichero. (Para una mejor visualización del diagrama remitirse al expediente de proyecto).

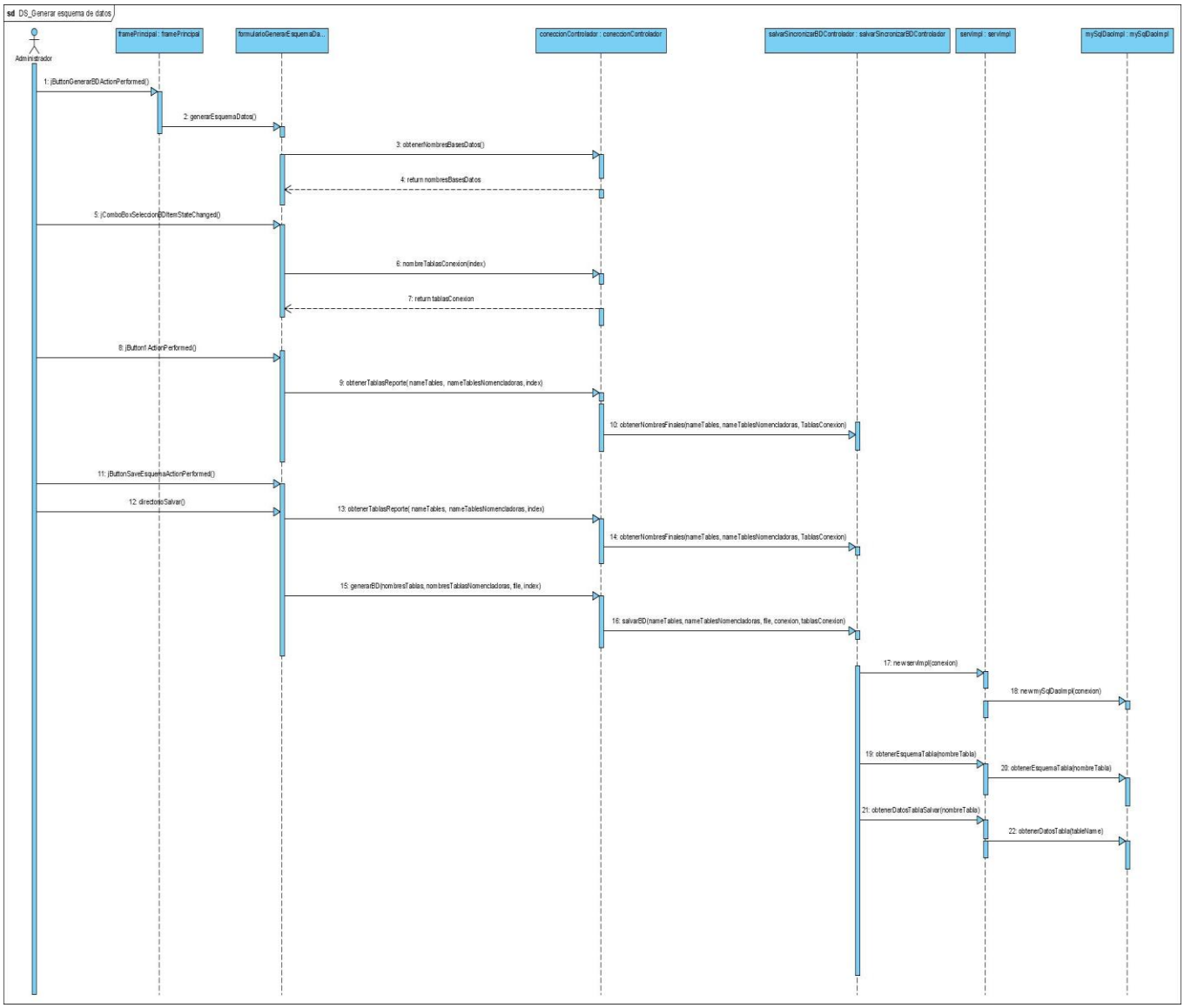


Fig. 9 Diagrama de secuencia caso de uso Generar esquema de datos.

El siguiente diagrama de secuencia pertenece al caso de uso Migrar base de datos Escenario Migrar datos. Aquí se representa la iteración entre las clases que intervienen cuando se desea migrar el contenido de una base de datos a otra. Es bueno destacar que para realizar la migración deben haberse hecho al menos 2 conexiones a base de datos homogéneas. El administrador especifica la base de datos origen y destino, determina el flujo de migración y se migran los datos. (Para una mejor visualización del diagrama remitirse al expediente de proyecto).

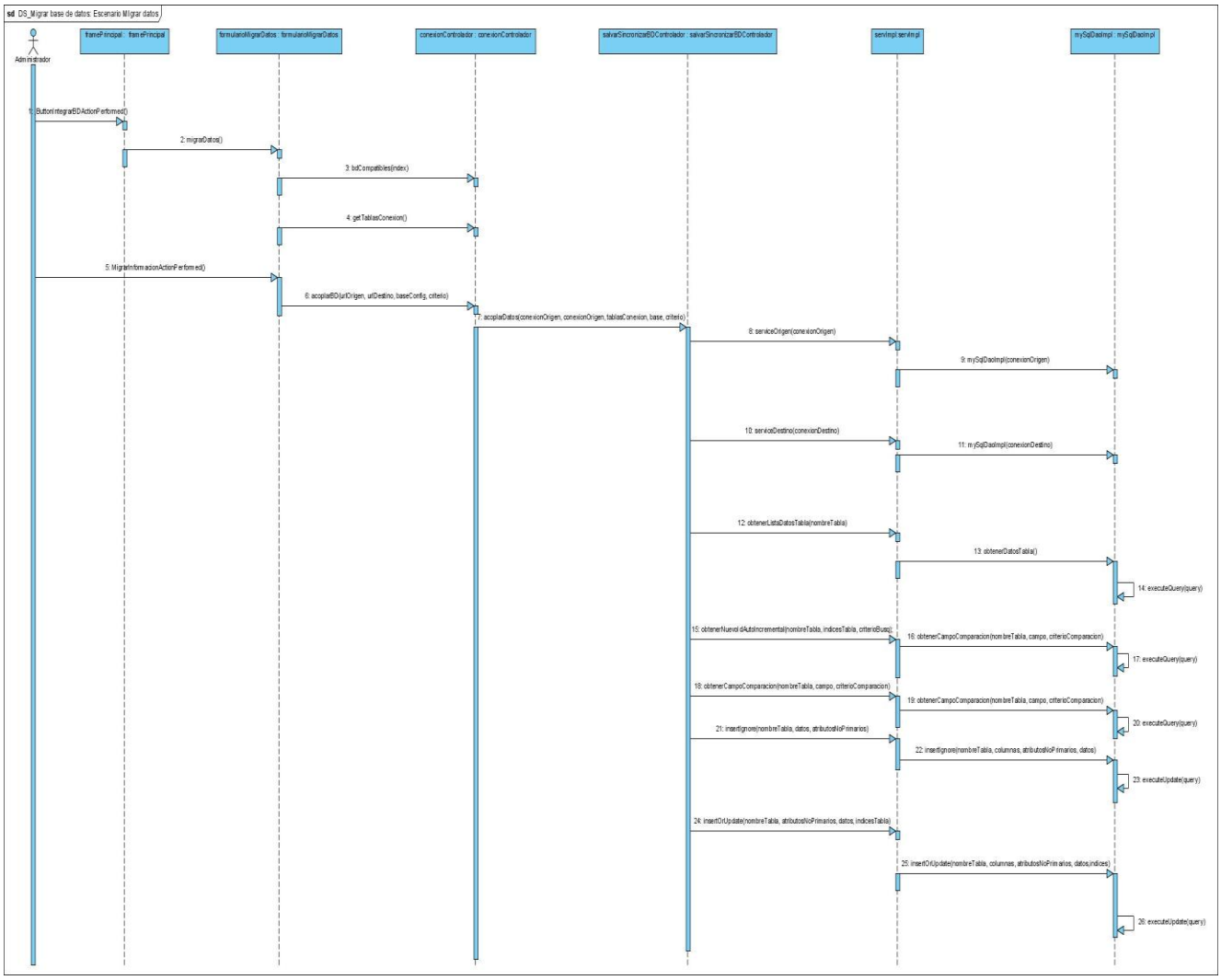


Fig. 10 Diagrama de secuencia caso de uso Migrar base de datos: Escenario Migrar datos.

3.4 Patrones de diseño utilizados.

Un patrón es una solución recurrente a un problema dentro de un contexto dado. Los patrones surgen de la experiencia de seres humanos al tratar de lograr ciertos objetivos, además capturan la experiencia existente y probada para promover buenas prácticas.

Según Larman, un patrón es una descripción de un problema y su solución, que recibe un nombre y que puede emplearse en otros contextos; además indica la manera de utilizarlo en circunstancias diversas.

Los Patrones de Diseño (Design Patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Ventajas de los patrones de diseño:

- Proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo.
- Están basados en la recopilación del conocimiento de los expertos en desarrollo de software.
- Es una experiencia real, probada y que funciona. Es Historia y nos ayuda a no cometer los mismos errores.

3.4.1 Patrones Generales de Asignación de Responsabilidades (GRASP)

Para el resultado de la investigación se emplea un grupo de patrones relacionados con el diseño de software, llamados patrones GRASP los que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Este grupo de patrones está muy relacionado con los problemas básicos del diseño. La asignación correcta de las responsabilidades en el diseño orientado a objetos garantiza la alta cohesión de las clases y el bajo acoplamiento de los mismos, lo que posibilita más extensibilidad, adaptabilidad y menos tiempo para el mantenimiento del diseño.

Experto.

Propone asignar la responsabilidad a la clase que cuenta con la información necesaria para cumplir la responsabilidad. Permitiendo que se conserve el encapsulamiento, soportando un bajo acoplamiento y una alta cohesión. La clase **conexionControlador** es la que contiene la información necesaria de las bases de datos a las que se están conectadas.

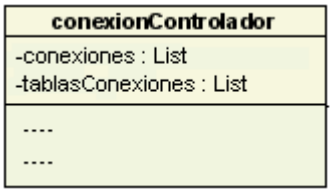


Fig. 11 Ejemplo de aplicación del patrón Experto.

Creador.

Propone asignarle a una clase la responsabilidad de crear los objetos de la otra en los casos de contener, agregar, registrar o utilizar. Brindando soporte de bajo acoplamiento, lo cual supone menos dependencias entre clases y posibilidades. La clase **servImpl** es la encargada de crear instancias de tipo **mySqlDaoImpl**, **tabla** y **constraintForeignKey**.

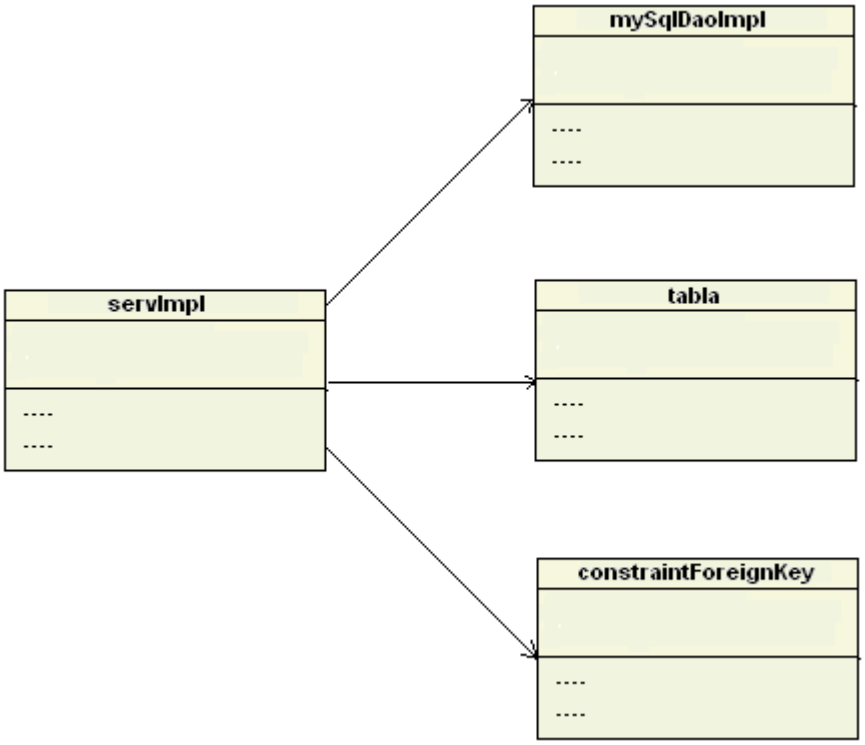


Fig. 12 Ejemplo de aplicación del patrón Creador.

Bajo Acoplamiento.

Este patrón se basa principalmente en la concepción de tener las clases lo menos ligadas entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, disminuyendo la dependencia y además son fáciles de entender por separadas.

Alta Cohesión.

Este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el

diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan un bajo acoplamiento, soporta mayor capacidad de reutilización.

Patrón Fabricación Pura.

La fabricación pura se da en las clases que no representan un ente u objeto real del dominio del problema, si no que se ha creado intencionadamente para disminuir el acoplamiento, aumentar la cohesión y/o potenciar la reutilización del código. Estas clases solo se centran en un conjunto muy específico de tareas relacionadas.

En virtud del patrón Experto, en cierto modo se justifica asignar las responsabilidades de salvar en un fichero el esquema de una base de datos especificada por el usuario y además acoplar la información de dos bases de datos en una de ellas a la clase **conexionControlador.**

Pero no sería una buena idea por lo siguiente:

La clase **conexionControlador** es la encargada de almacenar la información de las conexiones hechas a las bases de datos. Si además se le asignan las responsabilidades antes planteadas, que requieren de un número relativamente amplio de operaciones, se vería afectada la reutilización de esta clase. En virtud al patrón Fabricación Pura se decidió crear una nueva clase nombrada **salvarSincronizarBDControlador** que fuera la encargada de realizar dichas operaciones.

Fig. 13 Ejemplo de aplicación del patrón Fabricación Pura.

Patrón Indirección

Se asigna la responsabilidad a una clase intermedia para que medie entre otros componentes o servicios, y éstos no terminen directamente acoplados. El intermediario crea una indirección entre el resto de los componentes o servicios. La clase **servImpl** es un ejemplo de este patrón, ya que sirve de intermediaria entre los servicios de las bases de datos y el resto de la aplicación.

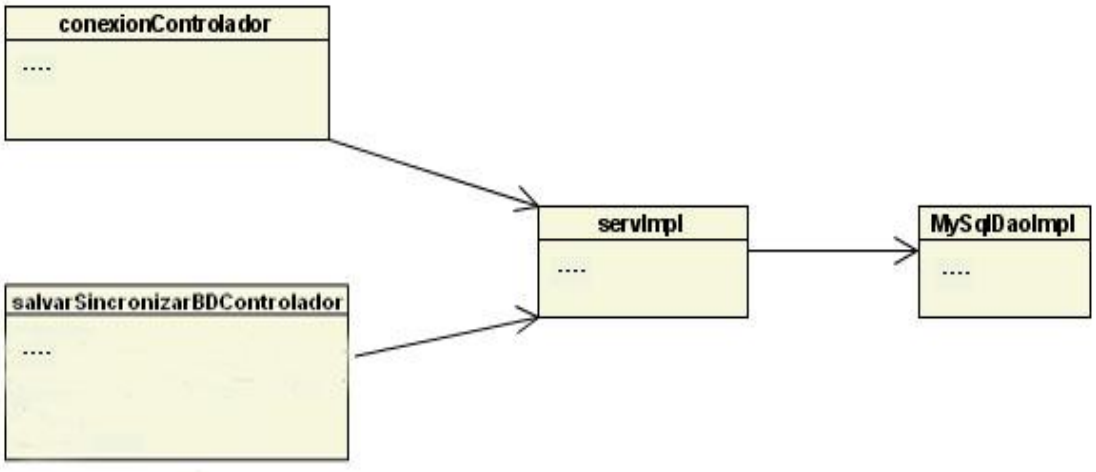


Fig. 14 Ejemplo de aplicación del patrón Indirección.

3.4.2 Patrones GoF

Patrón Singleton.

El patrón de diseño Singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su propósito es asegurar que sólo exista una instancia de una clase y proporcionar un punto de acceso global a ella.

El patrón se implementa creando en la propia clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

El patrón Singleton provee una única instancia global gracias a que:

- La propia clase es responsable de crear la única instancia.
- Permite el acceso global a dicha instancia mediante un método de la clase.
- Declara el constructor de clase como privado para que no sea instanciable directamente.

```
private conexionControlador() {
    conexiones = new LinkedList<conexion>();
    tablasConexiones = new LinkedList<List<tabla>>();
    conexionTemporal = null;
}

private static void CreateInstance()
  {
    if (Instance == null)
      {
        Instance = new conexionControlador();
      }
  }

public static conexionControlador GetInstance()
  {
    if (Instance == null) CreateInstance();
    return Instance;
  }
```

Fig. 14 Ejemplo de aplicación del patrón Singleton.

3.5 Patrones de arquitectura utilizados.

Arquitectura en tres capas.

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario. [12]

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

En el diseño de sistemas informáticos actuales se suele usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo

que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

El diseño utilizado actualmente es el diseño en tres niveles (o en tres capas).

Fig. 15 Arquitectura en tres capas.

Capa de presentación: es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario.

Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Ventajas de la arquitectura en 3 capas

Separación clara de la interfaz de usuario de la lógica de la aplicación. Esta separación permite tener diferentes presentaciones accediendo a la misma lógica. La redefinición del almacenamiento de información no tiene influencia sobre la presentación.

3.6 Modelo de despliegue

El modelo de despliegue de la aplicación está compuesto por dos nodos que representan a una PC cliente y a una servidora, ambas están interconectados utilizando el protocolo TCP/IP.

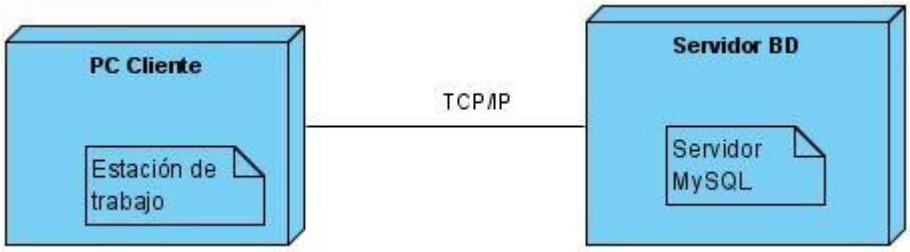


Fig. 16 Diagrama de despliegue.

Conclusiones

Teniendo en cuenta los casos de uso obtenidos en se elaboraron algunos de los artefactos propuestos por RUP para el desarrollo de software, tales como, la estructura de los paquetes de diseño, los diagramas de clases y los diagramas de interacción y despliegue. En el presente capítulo se describieron además las clases más importantes del diseño así como la especificación de todos los patrones de diseño implementados en dicha aplicación.

CAPÍTULO 4: IMPLEMENTACION DEL SISTEMA

Introducción

Este capítulo está enfocado en el flujo de trabajo de implementación para dar solución a los requisitos especificados. Se presenta el diagrama de componentes, que muestra la distribución física de los componentes para la implementación, además se evidencia la presencia de código relevante en la construcción de la herramienta.

4.1 Diagrama de componentes.

UML define el diagrama de componentes para la representación de un sistema de software en componentes físicos, como ejemplo: archivos, cabeceras, módulos y paquetes, y establecer las dependencias entre estos componentes.

A continuación se representa el diagrama de componentes de la herramienta desarrollada, compuesto por dos librerías encapsuladas en el subsistema lib. Además de la representación física de cada una de las clases .java y las interfaces que implementan dichas clases

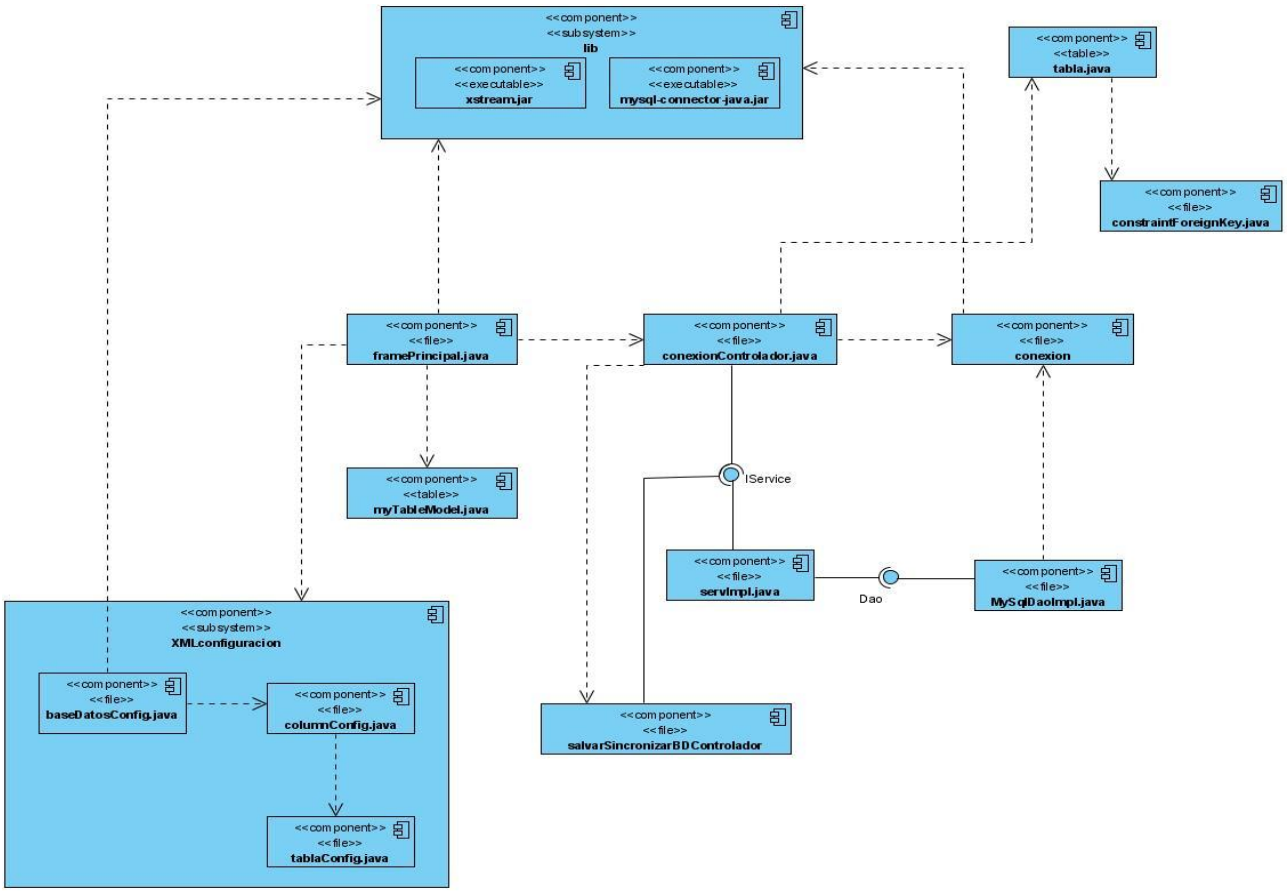


Fig. 17 Diagramas de componentes

4.2 Estándares de codificación

Un estándar de codificación es un documento que describe detalladamente las convenciones que los desarrolladores deben seguir para crear el código fuente de un sistema. Todos los programadores escriben el código de la misma manera, lo que permite que el trabajo en grupo sea más efectivo y que se logre la homogeneidad del código, así un desarrollador que ve parte del código fuente sabe que esperar de la aplicación entera.

La combinación de técnicas de codificación sólidas y las buenas prácticas de programación con el objetivo de lograr un código robusto, es de vital importancia para la calidad del software. La aplicación continua de un estándar de codificación puede contribuir a obtener un sistema de software fácil de comprender y de mantener.

Algunas reglas generales para codificar o definir un estándar de codificación

- No usar métodos y variables multipropósito.

- El tiempo de vida de las variables debe ser lo más corto posible.
- Manejar siempre las situaciones de error, preferentemente mediante rutinas de captura y tratamiento de excepciones.
- Usar las sentencias SWITCH-CASE en lugar de las sentencias IF-ELSE repetitivas.
- Escribir todos los nombres de las estructuras en el mismo idioma.
- Usar nombres objetivos para las variables y métodos, evitando la ocurrencia de ambigüedad en el código y contribuyendo a la comprensión del mismo.
- Evitar contener más de una clase por archivo.
- Usar la misma sangría en todo el código
- Comentar todos los métodos o funciones así como las partes del código que pudieran ser difíciles de comprender.

Estándar de codificación de la herramienta para desacoplar e integrar base de datos:

Nomenclatura

- Los nombres de cada uno de los elementos del programa deben ser significativos; su nombre debe explicar en lo posible el uso del elemento.
- La mayoría de los elementos se deben nombrar usando sustantivos.
- La forma de construir los nombres será colocando primero el verbo o el sustantivo, seguido de cada uno de sus complementos con la primera letra en mayúscula.

Ejemplos:

```
public ArrayList<String> obtenerNombreTablas() throws Exception;

public void cerrarConnection() throws Exception;

public String obtenerEsquemaTabla(String tableName) throws Exception;

public ArrayList<String> obtenerNombresIndicesTabla(String tableName)

public Object[][] obtenerDatosTabla(String tableName) throws Exception;
```

Estructura de archivos

Todos los archivos fuente deben tener la siguiente estructura básica general:

```
package xxxx;


//============================================================================

//  BIBLIOTECAS REQUERIDAS

//============================================================================

import xxx ;

//============================================================================

// FECHA CREACIÓN:

// AUTOR:

// …. Comentarios generales

//============================================================================


....
```

Ejemplo:

```
package appsGesBD.accDatos.servicios;

import appsGesBD.accDatos.entidades.tabla;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Creado en Febrero 18, 2009, 10:22:10 PM
 * @author Yunier Santana Aldana
 */
```

Comentarios de métodos analizadores

```
/**

  * Descripción de lo que hace el método

  * @param p Descripción del parámetro.

  * @return valor a retornar

  */
```

Ejemplo:

```
  /*
   * Este metodo me devuelve el esquema de una tabla
   * @param nombreTabla
   * @return el esquema de la tabla
   */
 public String obtenerEsquemaTabla(String nombreTabla) throws Exception {
     ResultSet resultado = dao.obtenerEsquemaTabla(nombreTabla);
     String result = null;
     while (resultado.next()) {
         result = resultado.getString(2);
     }
     return result;
 }
```

Definición de métodos

```
.... ... ..... ( .... , .... , .... ) {

    ..... ;

    ..... ;

}
```

Ejemplo:

```
public void cerrarConnection() throws Exception {
    dao.closeConnection();
}
```

4.3 Código fuente de las clases y funciones principales.

En este epígrafe se realizará una breve descripción de algunos de los métodos de mayor peso en el desarrollo de la herramienta informática, se hará mención por tanto, a los métodos que por su particularidad sobresalen ente los demás y entre ellos se encuentra los métodos que se pertenecen a la clase **salvarSincronizarBDControlador**. Esta clase es la que contiene toda la lógica del negocio necesaria para generar el esquema de una base de datos y migrar la información de una base de datos hacia otra.

En este método es común para los casos de uso Generar esquema de datos y Migrar bases de datos, ya que en ambos casos es el que se encarga de determinar el flujo de migración de las tablas. En otras palabras, este método devuelve el orden en que las tablas van a ser procesadas.

```
public List<List<String>> obtenerNombresFinales(List<String> nameTables, List<String> nameTablesNomencladoras, List<tabla> tablasConexion) {

    List<String> nombresFinales = nameTables;

    List<String> nombresNomencladoras = nameTablesNomencladoras;

    List<String> nombresVerdaderos = new ArrayList<String>();

    boolean flag = true;

    while (flag) {

        flag = false;

        for (int i = 0; i < nombresFinales.size(); i++) {

            String nombre = nombresFinales.get(i);

            tabla tabla = obtenerTabla(nombre, tablasConexion);

            if (!nombresVerdaderos.contains(nombre)) {
```

```
            nombresVerdaderos.add(nombre);
        }
        nombresFinales.remove(i);
        i--;

        List<String> listaTemporal = tabla.nombresForeignKey();
        for (int j = 0; j < listaTemporal.size(); j++) {
            if (!nombresFinales.contains(listaTemporal.get(j))) {
                nombresFinales.add(nombresFinales.size(), listaTemporal.get(j));
            }
            if (nombresNomencladoras.contains(nombre)) {
                if (!nombresNomencladoras.contains(listaTemporal.get(j))) {
                    nombresNomencladoras.add(nombresNomencladoras.size(), listaTemporal.get(j));
                }
            }
        }
    }



    List<tabla> posiblesTablasResultadosRelaciones_M_M = new ArrayList<tabla>();
    for (int i = 0; i < tablasConexion.size(); i++) {
        if (tablasConexion.get(i).cant_Primary_Foreign_Key() > 0) {
            if (!nombresVerdaderos.contains(tablasConexion.get(i).getTableName())) {
                posiblesTablasResultadosRelaciones_M_M.add(tablasConexion.get(i));
            }
```

```
        }
    }

    for (int i = 0; i < posiblesTablasResultadosRelaciones_M_M.size(); i++) {
        boolean banderaTabla = true;
        boolean banderaNomenclador = true;
        for (int j = 0; j < posiblesTablasResultadosRelaciones_M_M.get(i).cant_Primary_Foreign_Key(); j++) {
            if (posiblesTablasResultadosRelaciones_M_M.get(i).getListPrimaryForeignKey().get(j) != null) {
                if (!nombresVerdaderos.contains(posiblesTablasResultadosRelaciones_M_M.get(i).getListPrimaryForeignKey().get(j))) {
                    banderaTabla = false;
                }
                if (!nombresNomencladoras.contains(posiblesTablasResultadosRelaciones_M_M.get(i).getListPrimaryForeignKey().get(j))) {
                    banderaNomenclador = false;
                }
            }
        }

        if (banderaTabla) {
            nombresFinales.add(0, posiblesTablasResultadosRelaciones_M_M.get(i).getTableName());
            flag = true;
        }
        if (banderaNomenclador) {
            nombresNomencladoras.add(0, posiblesTablasResultadosRelaciones_M_M.get(i).getTableName());
        }
    }
```

```
    }

    List<List<String>> listasNombresSave = new ArrayList<List<String>>();

    listasNombresSave.add(nombresVerdaderos);

    listasNombresSave.add(nombresNomencladoras);

    return listasNombresSave;

}
```

4.4 Validación a nivel de desarrollador

Una forma de verificar la consistencia de los datos de la aplicación es introduciendo datos erróneos o dejando campos en blanco que son de carácter obligatorio. Esto podría considerarse como un paso de avance y ayuda para el flujo de trabajo de pruebas.

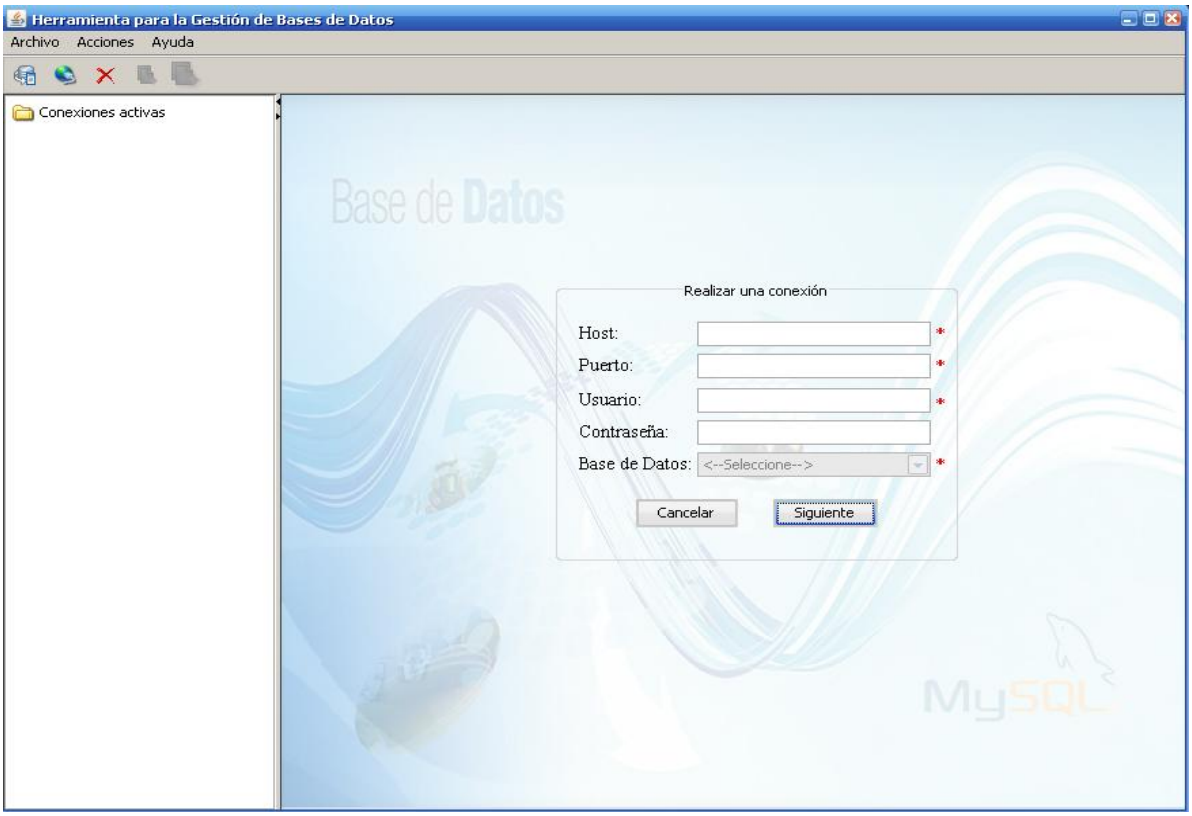


Fig.18 Configurar conexión a base de datos.

4.5 Interfaces de la aplicación informática.

Encaminando los pasos a lograr interfaces agradables, sencillas y atractivas a la vista del cliente, se obtiene las siguientes interfaces de la aplicación.

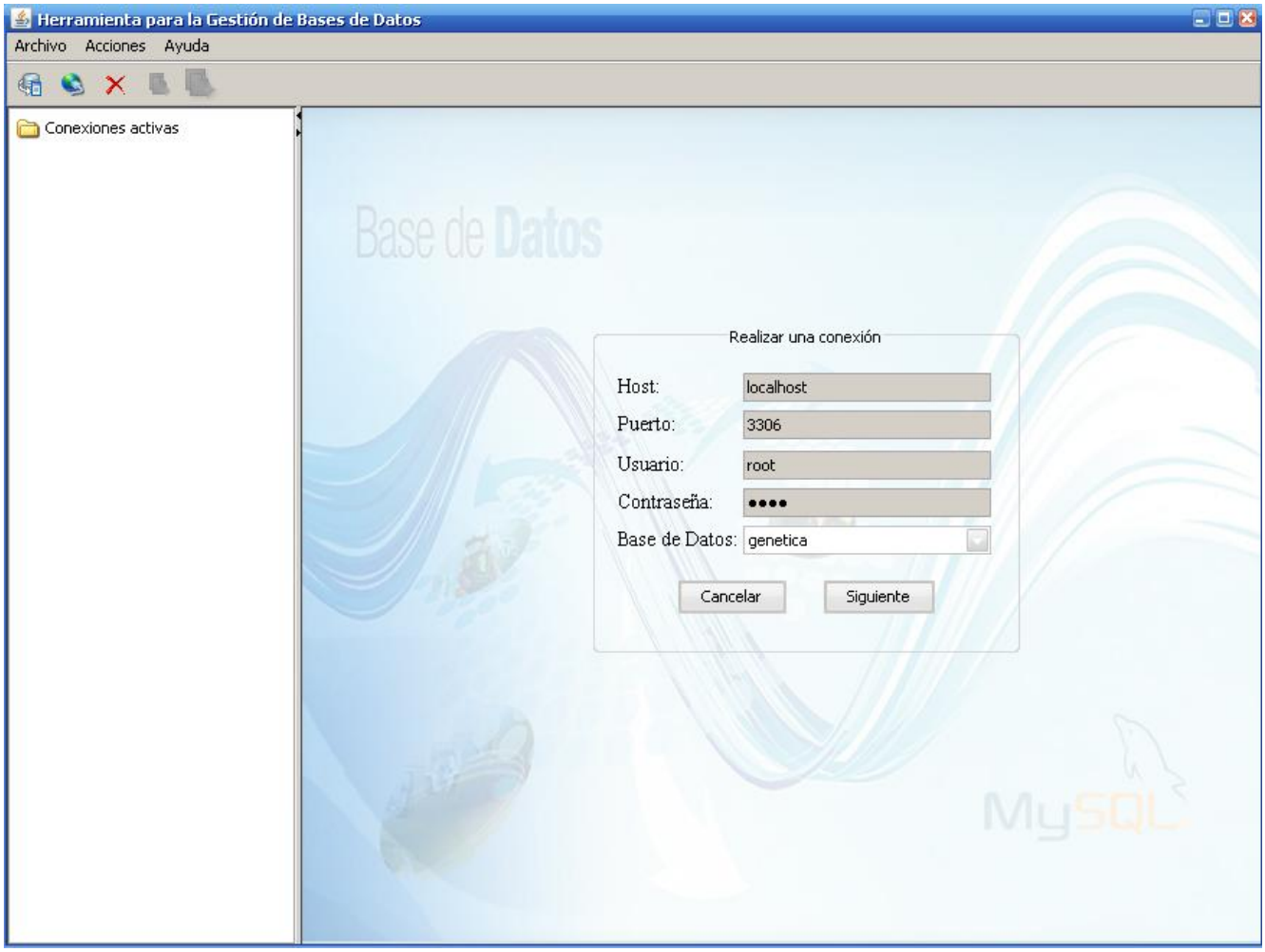


Fig. 19 CU Gestionar conexiones: Escenario Realizar conexión.

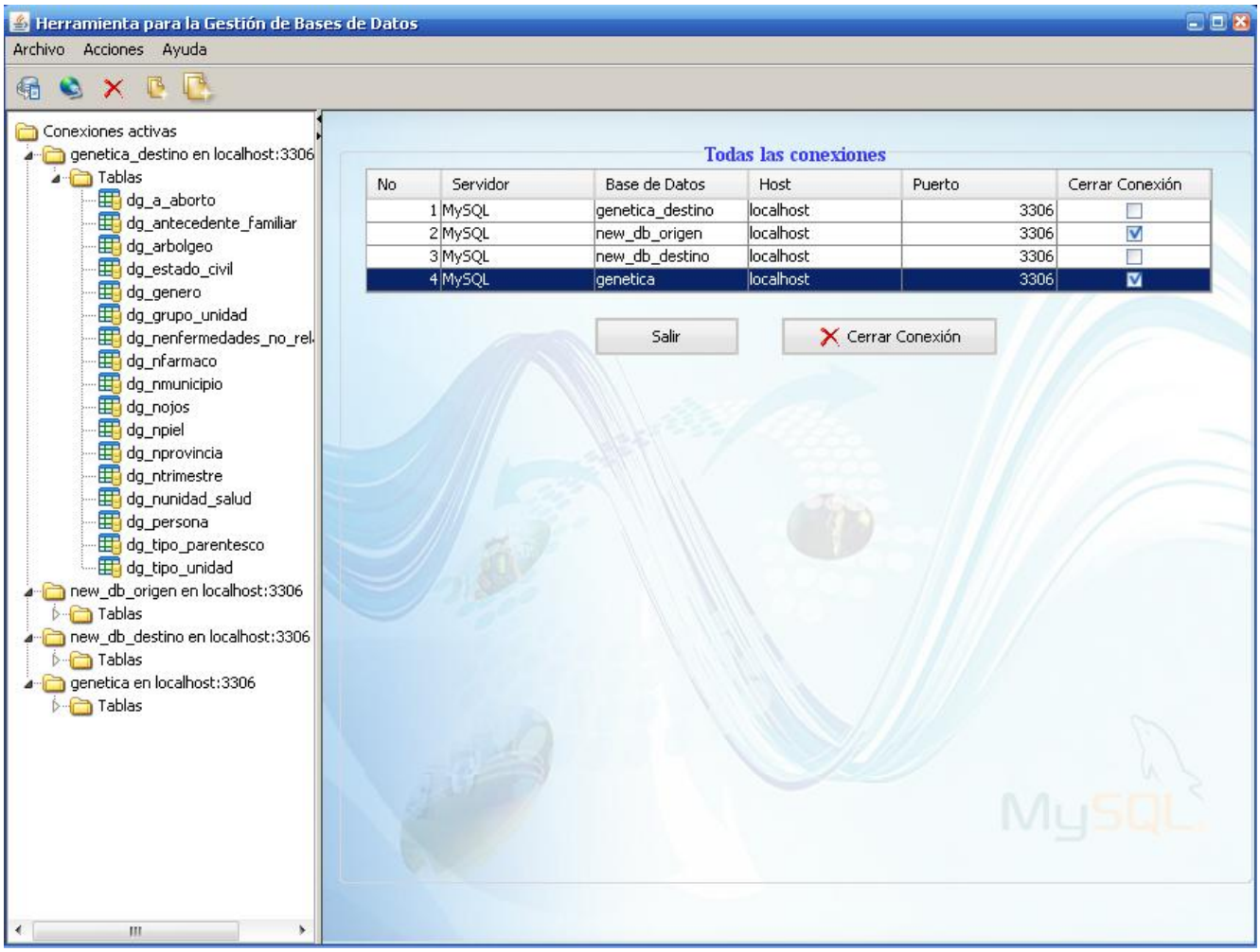


Fig. 20 CU Gestionar conexiones: Escenarios Mostrar y Cerrar conexiones.

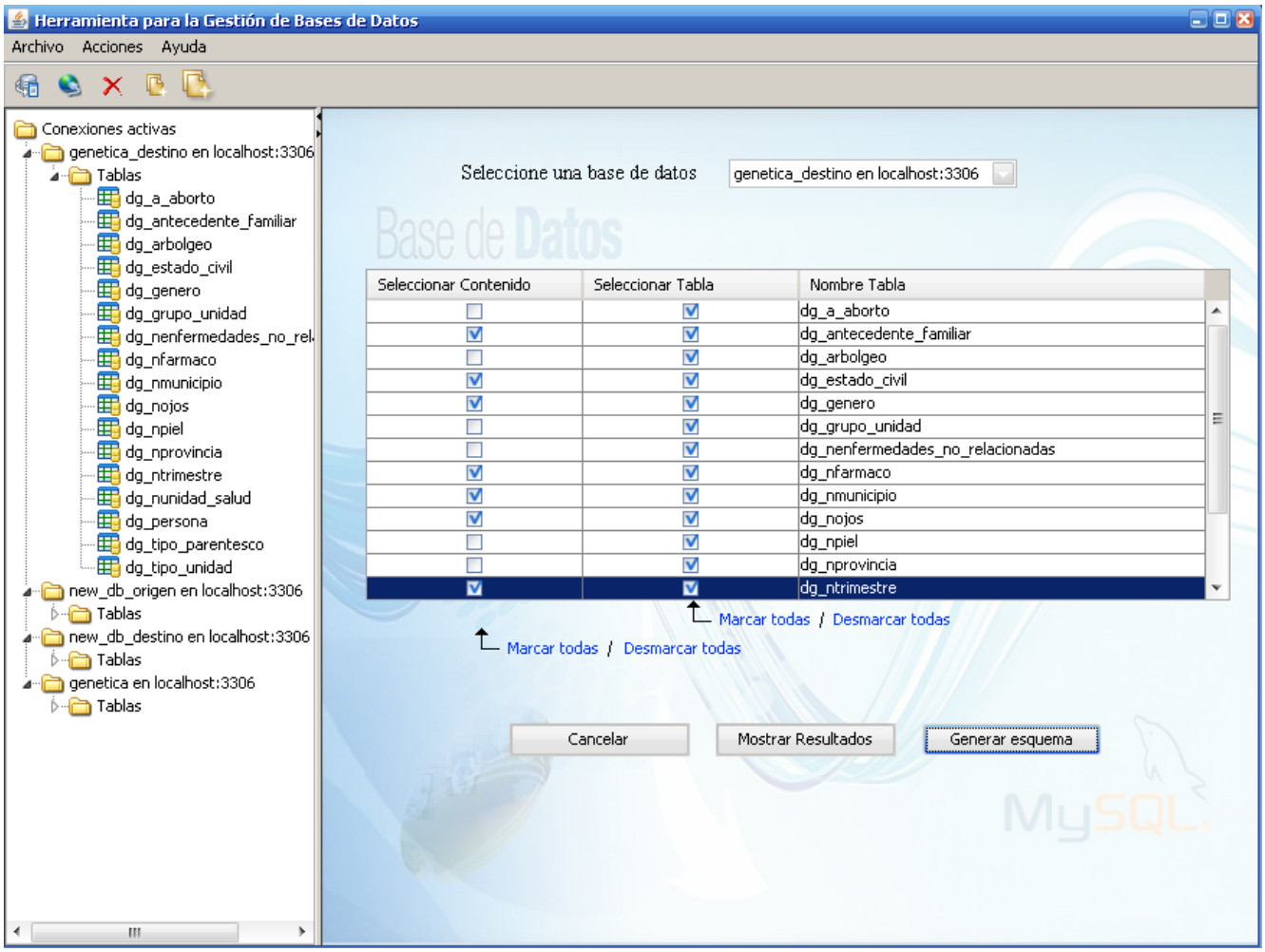


Fig. 21 CU Generar esquema de datos.

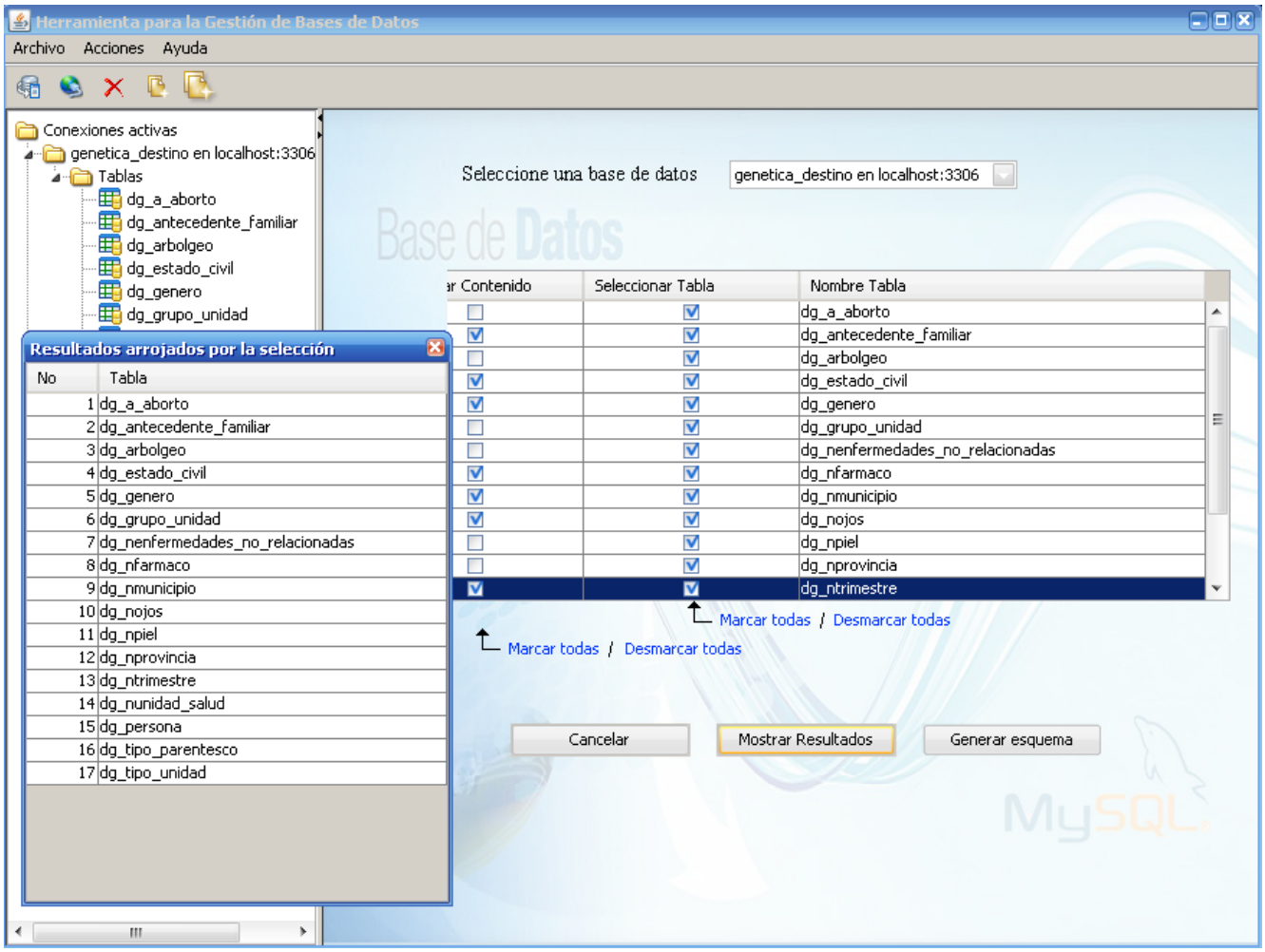


Fig. 22 CU Generar esquema de datos: Interfaz para mostrar relaciones arrojadas por la selección de las tablas.

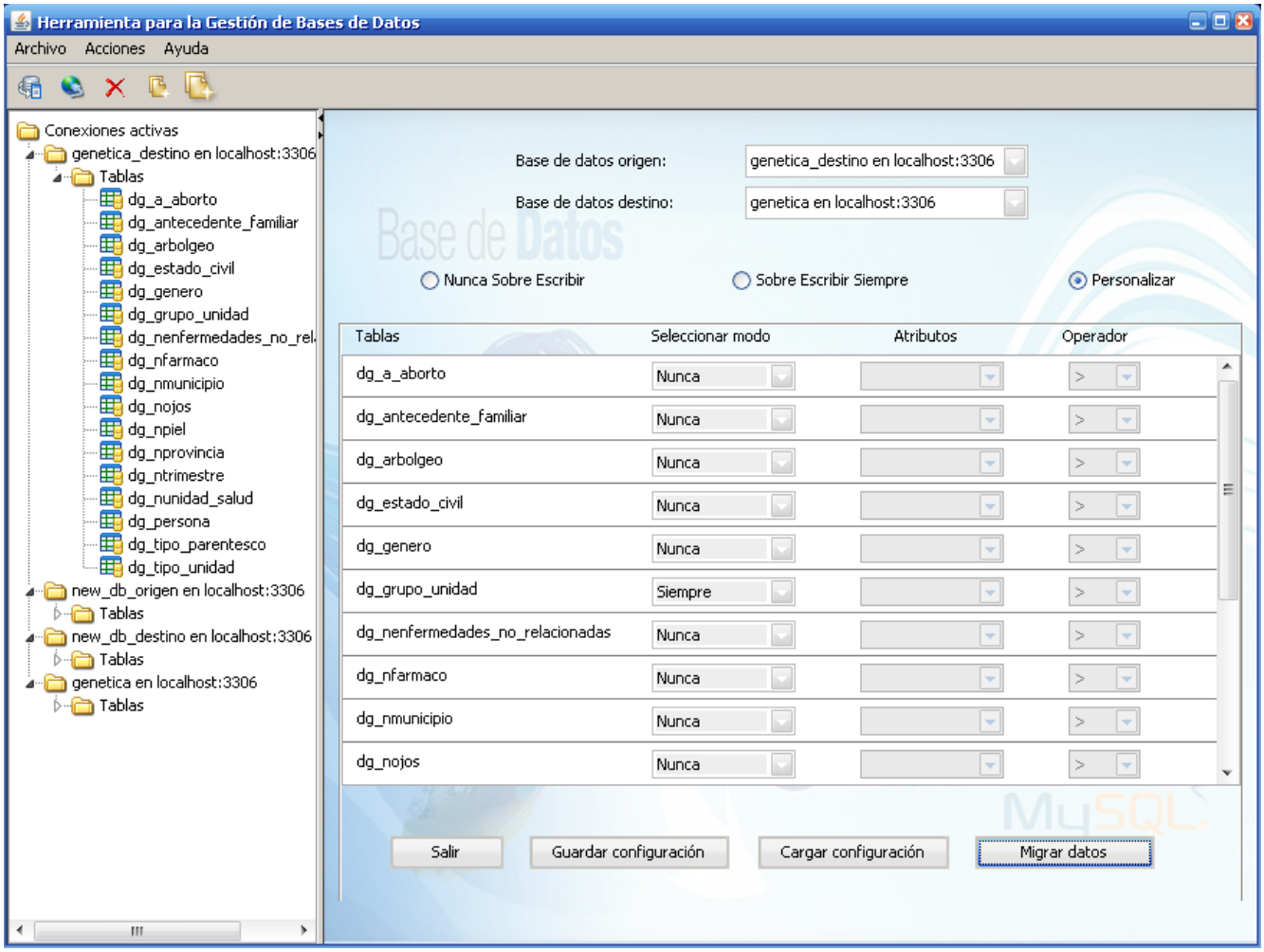


Fig. 23 CU Migrar base de datos: Escenario Migrar datos.

Conclusiones

Como resultado de este capítulo se obtuvo el diagrama de componentes, los estándares de codificación que se tuvieron en cuenta para la generación del código fuente. Además, una breve descripción de las principales clases, métodos, y las interfaces de la aplicación informática.

CONCLUSIONES GENERALES

La herramienta que se elaboró como resultado de la investigación realizada constituye un importante aporte a las entidades que no poseen conexión con la red nacional en todos sus municipios, permitiendo un avance en las investigaciones y en la recopilación de la información de forma eficiente.

- Se identificaron las funcionalidades críticas de la herramienta informática.
- Mediante la utilización de los patrones de diseño se diseñó una solución para hacer más eficiente el proceso de codificación.
- Con la implementación de las clases se obtuvo una aplicación que permitió la fragmentación e integración de base de datos en MySql.

RECOMENDACIONES

Tomando en cuenta las conclusiones del trabajo se recomienda:

- Realizar las pruebas exploratorias a la aplicación.
- Desarrollar una segunda versión para incorporar otros gestores de base de datos como Postgres y Oracle.
- Implementar como una funcionalidad de la aplicación, la extracción y carga directa de una base de datos de un servidor para otro.

REFERENCIAS BIBLIOGRÁFICAS

1. **Wikimedia Foundation, Inc.** Wikipedia. [En línea] 05 de 2009. http://es.wikipedia.org/wiki/ETL.

2. **Inc, SyBase.** SyBase. Replication. [En línea] 2009. http://www.es/products/dataintegration/replication.

3. **IDG.es.** CA ofrece una herramienta gratuita de gestión de bases de datos multiplataforma. [En línea] 04 de 2006. http://www.idg.es/cio/mostrarNoticia.asp?id=47268&seccion=tecnologias.

4. **IBEROAMERICANA, ADDISON-WESLEY.** El lenguaje Unificado de Modelado (UML). [aut. libro] RUMBAUGH J, JACOBSON I. EL LENGUAJE UNIFICADO DE MODELADO. ESPAÑA : PEARSON EDUCACION, Edición 2006.

5. **Sun Microsystems, Inc. 2008-2009.** MySQL. [En línea] 2008-2009. http://dev.mysql.com/doc/refman/5.0/es/features.html.

6. **García, Luis.** Educación Observatorio Tecnológico. Sistema de control de versiones: SUBVERSION . [En línea] 01 de 2008. http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=548.

7. NetBeans. [En línea] http://www.netbeans.org/index_es.html.

8. **Bustamante, Paul, y otros.** abcDatos. [En línea] 2004. http://www.tecnun.es/asignaturas/Informat1/ayudainf/aprendainf/Cpp/basico/cppbasico.pdf.

9. **Entorno Virtual de Aprendisaje**. Fase de Inicio. Flujo de trabajo de requerimientos. [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=12103

10**. Entorno Virtual de Aprendisaje**. FT Análisis y Diseño. Modelo de Diseño. [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=14069

11. **Entorno Virtual de Aprendisaje.** Flujo de Trabajo de requerimientos. [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=12443

12. **Mty.Coders**. Programación por Capas. [En línea] 03 de 2009. http://mtycoders.com/?p=359.

BIBLIOGRAFÍA

1. **Arias, Manuel.** Estándares de codificación. [En línea] 2007. http://62.204.199.21/carmen/estilo-codificacion.pdf.

2. **CA.** CA. Transforming IT Management. [En línea] 12 de 2008. http://www.ca.com/us/trials/collateral.aspx?cid=85443.

3. **Collins-Sussman, Ben, . Fitzpatrick, Brian W y Michael, Pilato C.** Version Control with Subversion For Subversion 1.4 (book compiled from Revision 1337). [En línea] Copyright © 2002, 2003, 2004, 2005. http://svnbook.red-bean.com/.

4. C# Online.NET. [En línea] http://es.csharp-online.net/.

5. **Entorno Virtual de Aprendisaje**. Modelo de dominio. [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=14077

6. **Entorno Virtual de Aprendisaje**. Patrones de Diseño y Arquitectura. [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=14077

7. **Entorno Virtual de Aprendisaje.** UML y RUP. [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=11406.

8. **ETL-Tools.Info.** ETL-Tools.Info. [En línea] 2006. http://etl-tools.info/es/.

9. **ETL-Tools.Info.** ETLTools Info. [En línea] (c) 2006-2009. http://etl-tools.info/es/.

10. **García, Luis.** Educación Observatorio Tecnológico. Sistema de control de versiones: SUBVERSION . [En línea] 01 de 2008. http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=548.

11. **González, Jose Antonio. 1999-2006.** Programación en castellanos. [En línea] 1999-2006. http://www.programacion.com/tutorial/csharp/.

12. Grupo soluciones GSInnova. Rational Undefine Process. [En línea] http://www.rational.com.ar/herramientas/rup.html.

13. **IBEROAMERICANA, ADDISON-WESLEY.** El lenguaje Unificado de Modelado (UML). [aut. libro] RUMBAUGH JAMES JACOBSON IVAR. EL LENGUAJE UNIFICADO DE MODELADO. ESPAÑA : PEARSON EDUCACION, 2006.

14. **IBM**. Grupo soluciones GSInnova. Rational Undefine Process. [En línea] http://www.rational.com.ar/herramientas/rup.html.

15. **IBM.** IBM. IBM Rational Unified Process . [En línea] http://www-01.ibm.com/software/awdtools/rup/.

16. **IDG.es.** CA ofrece una herramienta gratuita de gestión de bases de datos multiplataforma. [En línea] 04 de 2006. http://www.idg.es/cio/mostrarNoticia.asp?id=47268&seccion=tecnologias.

17. **Larman, Craig.** UML y Patrones. Introducción al análisis y diseño orientado a objetos. s.l. : Prentice Hall., 1999.

18. Lenguajes de programación. [En línea] 2009. http://www.lenguajes-de-programacion.com/programacion-java.shtml.

19. **MINSAP-MIC, Grupo de Arquitectura.** Arquitectura, normas y tecnologías para el desarrollo de aplicaciones. Ciudad de la Habana : s.n., Junio,2007.

20. **OMG.** Unified Modeling Language. UML Resource Page. [En línea] Copyright © 1997-2008 Object Management Group. http://www.uml.org/.

21. **Schmuller, Joseph.** Aprendiendo UML en 24 horas. 1999 : Prentice Hall.

22. **Sun Microsystems, Inc. 2008-2009.** MySQL. [En línea] 2008-2009. http://dev.mysql.com/doc/refman/5.0/es/features.html.

23. **Sun Microsystems, Inc. 2008-2009.** MySQL. [En línea] 2008-2009. http://dev.mysql.com.

24. **Sun Microsystems, Inc. 2008-2009.** MySQL. [En línea] 2008-2009. http://dev.mysql.com/doc/refman/5.0/en/key-column-usage-table.html

25. Sun Microsystem. [En línea] 1994-2009. http://www.sun.com/.

26. **Tigris.org. Open Source Software.** ArgoUML. [En línea] © 2006 CollabNet. CollabNet is a registered trademark of CollabNet, Inc. http://argouml.tigris.org/index.html.

27. **Tigris.org. Open Source Software Engineering Tools.** Subversion. [En línea] © 2006 CollabNet. . http://subversion.tigris.org/.

28. **Visual-Paradigm**. [En línea] http://www.visual-paradigm.com/.

29. **Visual Paradigm International .** Visual Paradigm. [En línea] 2008. http://www.visual-paradigm.com/.

30. **Wikimedia Foundation, Inc.** Wikipedia. [En línea] 05 de 2009. http://es.wikipedia.org/wiki/ETL.

ANEXOS

Anexo 1 Descripción de las clases del diseño

Nombre: MySqlDaoImpl	
Tipo de clase : accDatos	
Responsabilidades: Esta es la clase que se encarga de realizar las consultas a una base de datos.	
Nombre:	**executeQuery(String query)**
Descripción	Realiza una consulta a la base de datos.
Nombre:	**obtenerNombreTablas()**
Descripción:	Se obtiene el nombre de todas las tablas de una base de datos.
Nombre:	**closeConnection()**
Descripción:	Cierra una conexión con la base de datos.
Nombre:	**obtenerEsquemaTabla(String tableName)**
Descripción:	Con este método se obtiene la estructura de una tabla.
Nombre:	**obtenerNombresIndicesTabla(String tableName)**
Descripción:	Este método devuelve los nombres de los índices de una tabla.
Nombre:	**obtenerDatosTabla(String tableName)**
Descripción:	Este método devuelve los datos de una tabla.
Nombre:	**obtenerReferenciaTablas(String tableName)**
Descripción:	Se obtiene los nombres de las tablas a las cuales hace referencia la tabla cuyo nombre es el que se pasa por parámetros.

Nombre:	**obtenerNombresBasesDatos()**
Descripción:	Se obtienen los nombres de todas las bases de datos de un servidor.
Nombre:	**MigrarDatos(String nameTabla, List<String> Atributos, Object[][] listaDatos)**
Descripción:	Exporta los datos de una base de datos origen hacia otra base de datos destino.

Nombre: serviceImpl	
Tipo de clase : accDatos	
Responsabilidades: Esta clase es la encargada de aislar el resto de la aplicación del código SQL, convirtiéndolo a código java.	
Nombre:	**closeConnection()**
Descripción:	Cerrar la coneccion a la base de datos.
Nombre:	**obtenerEsquemaTabla(String tableName)**
Descripción:	Con este método se obtiene la estructura de una tabla.
Nombre:	**obtenerNombreTablas()**
Descripción:	Se obtiene el nombre de todas las tablas de una base de datos
Nombre:	**obtenerDatosTablaSalvar(String tableName)**
Descripción:	Se devuelven los datos de una tabla, con la particularidad de que la información de cada fila se devuelve en una única cadena, así se hace más fácil a la hora de generar el esquema de esa tabla.

Nombre:	**obtenerNombresIndicesTablaSalvar(String tableName)**
Descripción:	Este método me devuelve los índices de una tabla en una sola cadena, separándolos por coma, así se hace más fácil a la hora de generar el esquema de esa tabla.
Nombre:	**obtenerReferenciaTablas(String tableName)**
Descripción:	Se obtiene los nombres de las tablas a las cuales hace referencia la tabla cuyo nombre es el que se pasa por parámetros.
Nombre:	**obtenerNombresBasesDatos()**
Descripción:	Se obtienen todos los nombres de las bases de datos.
Nombre:	**obtenerNombresIndicesTabla(String tableName)**
Descripción:	Se obtiene la descripción de una tabla, de la cual se quiere saber toda la información referente a sus índices.
Nombre:	**MigrarDatos(String nameTabla, List<String> Atributos, Object[][] listaDatos)**
Descripción:	Exporta los datos de una base de datos origen hacia otra base de datos destino
Nombre:	**obtenerDatosTabla(String tableName, int cantIndices)**
Descripción:	Se obtienen los datos de una tabla.

Anexo 2 Diagramas de secuencia

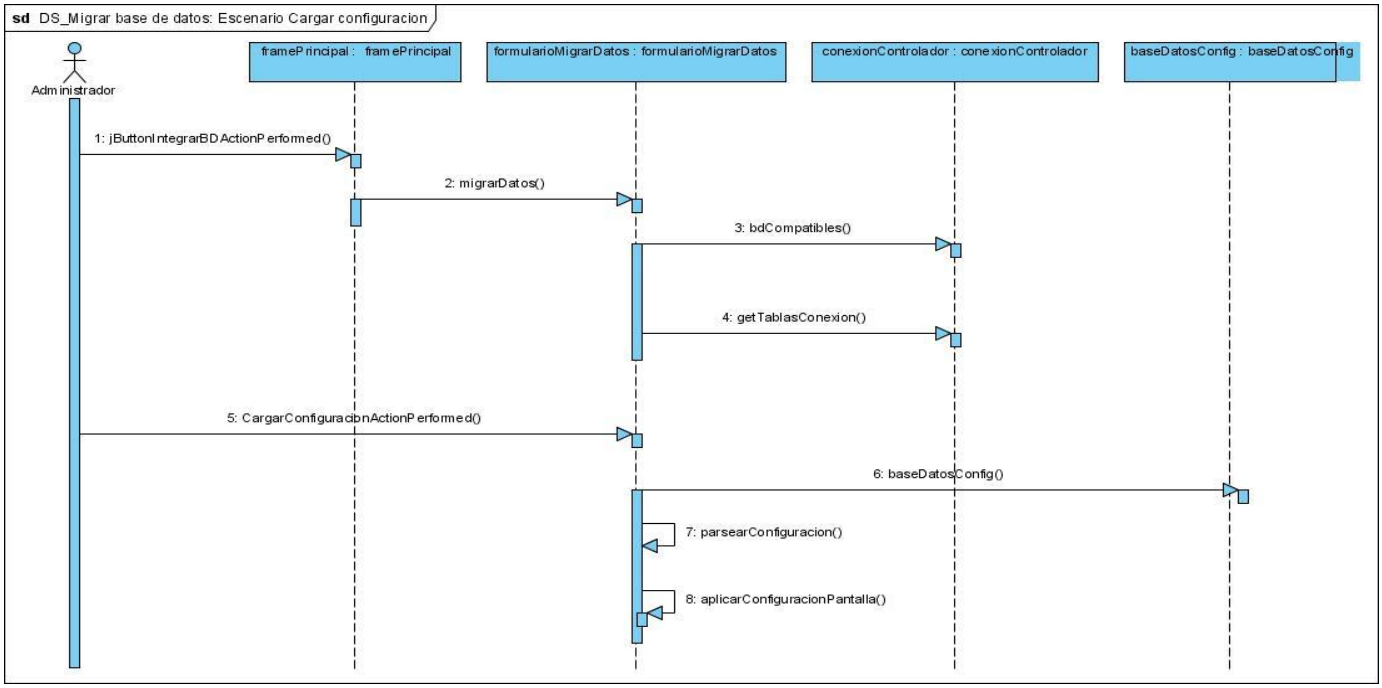


Fig. 24 Diagrama de secuencia CU Migrar base de datos: Escenario Cargar configuración.

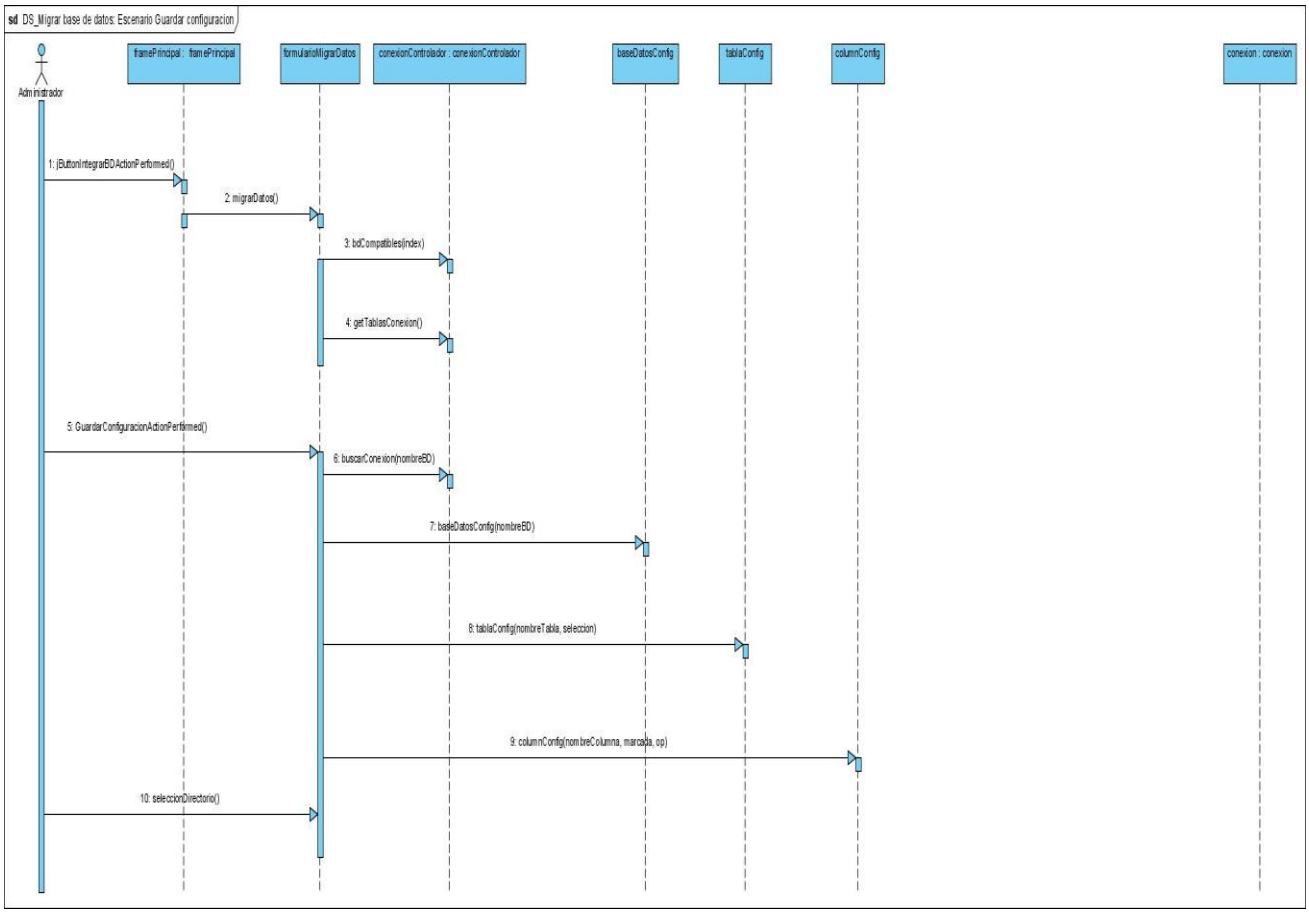


Fig. 25 Diagrama de secuencia CU Migrar base de datos: Escenario Guardar Configuración.

GLOSARIO DE TÉRMINOS

1. AJAX: Técnica de desarrollo web para crear aplicaciones interactivas cuyas siglas vienen dadas por su nombre en inglés Asynchronous JavaScript And XML que significa JavaScript asincrónico y XML.

2. API's: Son Interfaces de Programación de Aplicaciones que da a los programadores los medios para desarrollar aplicaciones Java.

3. C++: Lenguaje de programación para el desarrollo de software.

4. C#: Lenguaje de programación para el desarrollo de software.

5. CU: Un caso de uso es una técnica de ingeniería de software para la captura de requisitos deseados por el cliente en el desarrollo de un nuevo sistema o una actualización de software.

6. Delphi: Lenguaje de programación para el desarrollo de software.

7. ETL: Herramientas que se utilizan para la extracción, transformación y carga de datos.

8. GNU/GPL: La Licencia Pública General.

9. Grails: Es un framework para aplicaciones web de código abierto desarrollado sobre el lenguaje de programación Groovy.

10. GRASP: Patrones generales de software para asignación de responsabilidades.

11. Groovy: Lenguaje de programación orientado a objetos.

12. Herramientas CASE: Conjunto de aplicaciones informáticas orientadas al incremento de la productividad en el desarrollo de software, las siglas CASE vienen dadas por su nombre en inglés Computer Aided Software Engineering que se conoce como Ingeniería de Software Asistida por Computadoras.

13. http: Protocolo de transferencia de hipertexto.

14. https: Protocolo seguro de transferencia de hipertexto.

15. IBM: Empresa internacional mundialmente conocida por la venta de tecnología informática cuyas siglas vienen dadas por su nombre en inglés International Business Machines Corporation.

16. IDE: Un entorno de desarrollo integrado es un software compuesto por un grupo de herramientas integradas para facilitar el trabajo del programador cuyas siglas vienen dadas por su nombre en inglés Integrated Development Environment.

17. Java: Lenguaje de programación para el desarrollo de software.

18. Java EE: Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java.

19. JavaScript: Lenguaje de programación interpretado, que no requiere compilación, utilizado principalmente en páginas web.

20. JDBC: Es un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

21. Linux: Sistema operativo de código abierto.

22. MAC's: Nombre que recibe una computadora personal diseñada, desarrollada, construida y comercializada por la compañía Apple Inc.

23. MIC: Ministerio de la Informática y las Comunicaciones en Cuba.

24. MySQL: Sistema Gestor de Base de Datos.

25. NetBeans: Es una plataforma para el desarrollo de aplicaciones en Java.

26. Oracle: Sistema Gestor de Base de Datos.

27. PHP: Lenguaje de programación para el desarrollo de software.

28. POO: Programación orientada a objetos.

29. Postgres: Sistema Gestor de Base de Datos.

30. RAM: Es la memoria de cualquier computadora desde la cual el procesador recibe las instrucciones y guarda los resultados.

31. Ruby: Es un lenguaje de programación interpretado, reflexivo y orientado a objetos.

32. RUP: Metodología de desarrollo cuyas siglas vienen dadas por su nombre en inglés Rational Unified Process. Establece para el desarrollo de un software una serie de flujos de trabajo agrupados en cuatro fases fundamentales.

33. Script: Fichero de texto.

34. SO: Sistema operativo.

35. SQL: Es un Lenguaje de consulta estructurado.

36. svn: Software de sistema de control de versiones.

37. TCP/IP: Conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

38. UML: Lenguaje visual para especificar, construir y documentar un sistema de software. Sus siglas vienen dadas por su nombre en inglés Unified Modeling Language.

39. Web: Sistema de documentos interconectados por enlaces de hypertexto, disponibles en Internet.

40. Windows: Sistema operativo propietario.

41. XML: Estándar de información cuyas siglas vienen dadas por su nombre en inglés extensible Markup Language.

42. xstream: Es una librería que permite transformar objetos de Java a XML y viceversa.

43. .NET: Plataforma de desarrollo de software.