Universidad de las Ciencias Informáticas

Facultad 4

Desarrollo de un Generador de Datos para el proyecto SIGEP.

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

Autor(es): Richar Hernández Herrera

Giorbis Santiesteban Marín

Tutor(es): Ing. Michel Díaz Llerena

Ing. Maikel Pérez Martínez

Cotutora: Ing. Yoenia María Martínez Díaz

Ciudad de la Habana, junio del 2009.

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 23 días del mes de junio del año 2009.

Richar Hernández Herrera

_____

Giorbis Santiesteban Marín

_____

Ing. Michel Díaz Llerena

_____

Ing. Maikel Pérez Martínez

_____

Datos de Contacto

Ing. Michel Diaz Llerena

Graduado en Julio del 2007 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas con Título de Oro y Promedio General de 4.82 puntos. Ha impartido las asignaturas de Base de Datos y Data WareHouse. Participó como desarrollador en el proyecto UCIMATICA y en un Sistema de control de Grupos Electrógenos. Desarrollador de base de datos del Sistema de Gestión Penitenciaria para la República de Venezuela. Actualmente diseñador de base de datos Principal del Sistema de Gestión Penitenciaria.

Correo electrónico: mdiazl@uci.cu

Ing. Maikel Pérez Martínez

Graduado en Julio del 2007 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Ha impartido las asignaturas de Introducción a la Programación, Programación I y IV, y el curso de Patrones de Diseño en la Universidad de las Ciencias Informáticas. Diseñador y desarrollador del Subsistema Sala Situacional del Sistema de Gestión Penitenciaria para la República de Venezuela, para el que ha elaborado e impartido curso de capacitación.

Correo electrónico: mperezma@uci.cu

Agradecimientos

De Richar

Gracias a:

- Mis padres y hermanas, por confiar siempre en mí, el título es de ustedes.
- Mi novia y tutora, Yeny por estar siempre presente.
- Mis amistades, principalmente a aquellos que siempre han estado cerca Darién, José Luis, Joel, Víctor, Alejandro, Leosvel, Ángel Gustavo, Ernesto, Hardam, Ernesto y Danny.
- Mi compañero de tesis, al fin somos ingenieros.

De Giorbis

Gracias a:

- Mi madre por toda la confianza y el amor brindado durante todos estos años.
- Mi hermana Misleidis por ser como mi madre durante todo este tiempo.
- Mis demás hermanas y familiares.
- La novia de mi compañero de tesis y nuestra tutora Yoenia que sin ella no hubiese sido posible la realización de este trabajo, para ella mi mayor agradecimiento.
- Mi compañero de tesis Richar.
- Lianet por compartir, soportarme y ser de la forma en que ha sido durante todos estos años.
- Mi novia.
- Mis amigos Karel, Ángel, Leosvel, José Luis, Joel, Ernesto, Lissuan, Víctor y todos los que me han ayudado todo este tiempo.

A todos aquellos que nos ayudaron en el desarrollo del trabajo, Víctor, Joel, José Luis, Alejandro, Ángel Gustavo, Jorge Ernesto, Julio, Yeny, Ana Marys y Lissett. A nuestros tutores por su trabajo y ayuda durante este año. A Julio César por sus valoraciones críticas y sus sugerencias que fueron de gran ayuda durante el desarrollo de esta investigación.

Dedicatoria

De Richar

- A mis padres y hermanas.
- A mi novia Yeny.

De Giorbis

- A mis padres y hermanas.
- A mi novia Leixy.

Resumen

Una práctica muy útil al diseñar sistemas que impliquen la utilización de bases de datos, es ingresar datos de prueba para ver de qué manera son manejados por nuestro sistema. Pero en el caso de implementación de bases de datos que soportarán sistemas de gran escala, el ingreso de datos de prueba de manera manual puede ser una tarea realmente desesperante. Afortunadamente, existen herramientas que nos permiten generar de manera automática grandes volúmenes de datos de prueba. En el presente trabajo se describe el proceso de creación de una herramienta que permita generar datos para poblar una base de datos de manera automática.

En el presente trabajo se realiza un estudio de las herramientas generadoras de datos más conocidas a nivel mundial, así como de las metodologías y técnicas de desarrollo de software, con el objetivo de lograr un producto de calidad y que satisfaga las necesidades del usuario final, contribuyendo de esta forma al avance de la Universidad de Ciencias Informáticas (UCI) y en general de nuestro país, en el área del desarrollo de herramientas para bases de datos.

Palabras Clave

Bases de datos, Datos de prueba, generar, poblar.

Índice

Declaración de Autoría I

Datos de Contacto II

Agradecimientos III

Dedicatoria IV

Resumen V

Palabras Clave V

Índice de ilustraciones VIII

Introducción 1

Capitulo 1. Diseño Teórico 2

1.1 Situación problémica 2
1.2 Objeto de estudio 2
1.3 Campo de acción 2
1.4 Objetivo General 2
1.5 Tareas de la investigación. 3
1.6 Fundamentación teórica 4
1.6.1 Introducción. 4
1.6.2 Transacciones de la Base de datos. 5
1.6.3 Restricciones 6
1.6.3.1 Restricciones de unicidad 6
1.6.3.2 Restricciones de No-Nulo 6
1.6.3.3 Restricciones de Dominio 7
1.6.4 Metadatos 7
1.6.5 Algoritmos de generación de datos. 8
1.6.5.1 KRIMP Generator 8
1.6.5.2 GADGET 8
1.6.6 Algoritmos de inserción de datos. 9
1.6.7 Generadores de datos. 9
1.6.7.1 DB Data Generator 9
1.6.7.2 forSQL Data Generator. 10
1.6.7.3 Generatedata 10
1.6.7.4 DTM DataGenerator v1.22.05 11
1.6.7.5 TDG 1.2c 11
1.6.7.6 EMS Data Generator for Oracle 11
1.6.7.7 Advanced Data Generator 12
1.6.7.8 Datatect Generator v1.6 12
1.6.8 Propuesta de solución 13

1.7 Conclusiones 14
Capítulo 2: Descripción de la solución propuesta 15
2.1 Introducción 15
2.2 Técnicas y herramientas 16
2.2.1 RUP 16
2.2.2 UML 17
2.2.3 Visual Paradigm 18
2.2.4 Java 19
2.2.5 Oracle 19
2.2.6 JDBC 20
2.3 Requerimientos 21
2.3.1 Requerimientos funcionales 22
2.3.2 Requerimientos no funcionales 23
2.4 Actores del Sistema 24
2.5 Diagrama de Casos de Uso del Sistema 25
2.6 Casos de uso expandidos 25
2.7 Conclusiones 30
Capítulo 3: Análisis y diseño de la solución propuesta 31
3.1 Introducción 31
3.3 Análisis del Sistema 31
3.3.1 Diagramas de clases del análisis 31
3.3.2 Diagramas de interacción 33
3.4 Diseño del Sistema 36
3.4.1 Diagrama de clases del diseño 36
3.4.2 Descripción de las clases de diseño 37
3.5 Conclusiones 43
Capítulo 4: Implementación y Prueba 44
4.1 Introducción 44
4.2 Implementación 44
4.2.1 Algoritmo de generación 44
4.2.2 Algoritmo de inserción de datos. 45
4.2.3 Diagrama de Componente 46
4.2.4 Diagrama de Despliegue 47
4.3 Prueba 47
4.4 Conclusiones 50
Conclusiones 51
Recomendaciones 52
Referencias bibliográficas 53
Bibliografía 54
Glosario de términos 55

ANEXOS 56
Anexo 1. Plantilla para la Descripción de un Caso de Uso. 56
Anexo 2. Plantilla para la Descripción de un Caso de Prueba. 57

Índice de ilustraciones

Fig. 1 Fases de RUP 17
Fig. 2 Lenguaje de Modelado Unificado 18
Fig. 3 Visual Paradigm 18
Fig. 4 Java 19
Fig. 5 JDBC 20
Fig. 6 Arquitectura JDBC 21
Fig. 7 Diagrama de Casos de Uso del Sistema 25
Fig. 8 Diagrama de clases del Análisis (CUS: Crear Conexión) 32
Fig. 9 Diagrama de clases de Análisis (CUS: Generar Datos) 32
Fig. 10 Diagrama de clases del diseño (CUS: Gestionar Datos) 33
Fig. 11 Diagrama de secuencia (CUS: Gestionar Conexión Sección: Crear Conexión) 34
Fig. 12 Diagrama de secuencia (CUS: Gestionar Conexión Sección: Cerrar Conexión) 34
Fig. 13 Diagrama de secuencia (CUS: Generar Datos) 35
Fig. 14 Diagrama de secuencia (CUS: Gestionar Datos Sección Mostrar Tablas) 35
Fig. 15 Diagrama de secuencia (CUS: Gestionar Datos Sección Eliminar Datos) 36
Fig. 16 Diagrama de secuencia (CUS: Gestionar Datos Sección Propiedades) 36
Fig. 17 Diagrama de clases del diseño 37
Fig. 18 Diagrama de componentes 46
Fig. 19 Diagrama de despliegue 47

Introducción

En muchas ocasiones es muy difícil para los desarrolladores contar con los datos necesarios para probar las nuevas aplicaciones que están desarrollando, por tanto las pruebas se realizan con una pequeña cantidad de datos que son producidas de forma manual, lo que los guía a un proceso de prueba fallido sobre la aplicación, puesto que la aplicación puede dar errores en el ambiente de desarrollo, debido a condiciones imprevistas creadas por los datos entrados manualmente.

Mientras se desarrollan estas nuevas aplicaciones, los desarrolladores se encuentran en una situación donde tienen un nuevo esquema de base de datos, pero sin ninguna información dentro para probarlo. Además en los casos donde una aplicación existente es modificada, la base de datos puede no tener la cantidad de datos suficientes para realizar pruebas a las modificaciones que se están realizando sobre la aplicación.

En muchos casos esto puede llevar a situaciones donde la aplicación no es probada debidamente para abarcar todos los escenarios presentes en la misma, o sea, esto puede resultar en pruebas defectuosas. Además se vuelve prácticamente imposible probar el funcionamiento de la aplicación para grandes juegos de datos y no es factible entrar grandes cantidades de datos de forma manual. Por tanto resulta beneficioso utilizar una herramienta que genere datos de prueba del tamaño requerido automáticamente, con las mínimas entradas posibles por parte del usuario.

Capitulo 1. Diseño Teórico

1.1 Situación problémica

El proceso de desarrollo del proyecto SIGEP[1], particularmente en el área concerniente a la base de datos del sistema, se ha visto afectado en varias ocasiones a la hora de probar el funcionamiento de la misma, sobre todo debido a que la inserción de datos es totalmente manual, lo que requiere de un tiempo considerable para contar con un juego de datos consistente y lo suficientemente voluminoso como para probar el comportamiento de la base de datos bajo condiciones extremas. Saber hasta qué cantidad de información es capaz de manejar la base de datos sin que se afecte el rendimiento y la fiabilidad de la misma, se hace difícil y muy costoso en tiempo debido a que requiere de horas o tal vez días de trabajo del equipo de desarrolladores del proyecto, insertar juegos de datos de manera manual.

1.2 Objeto de estudio

Los algoritmos de población automática de base de datos y los modelos de almacenamiento de la meta-información en Oracle.

1.3 Campo de acción

El proceso de población de la base de datos de SIGEP.

1.4 Objetivo General

Implementar una herramienta que permita generar juegos de datos para hacer pruebas sobre la base de datos de SIGEP.

[1] Sistema de Gestión Penitenciaria

1.5 Tareas de la investigación.

1. Analizar el mecanismo para almacenar la meta-información en el gestor Oracle.
2. Analizar las tendencias actuales en el desarrollo de los sistemas de generación automática de datos.
3. Definir el uso de un algoritmo de generación de datos.
4. Diseñar una aplicación capaz de generar e insertar juegos de datos.
5. Implementar una aplicación capaz de generar e insertar juegos de datos en la Base de Datos de SIGEP.

1.6 Fundamentación teórica

1.6.1 Introducción.

Uno de los grandes problemas con los que usualmente nos enfrentamos a la hora de desarrollar una aplicación web, es el hecho de que el rendimiento en las etapas iniciales no es similar a cuando hay muchos datos y muchos usuarios concurrentes. Para determinar si el volumen de datos de la base de datos creada se corresponde con el que realmente manejará la aplicación, hay que llenar las tablas con datos de prueba. Una de las tareas menos populares es crear un juego de datos para probar las bases de datos, esta es una actividad que puede tomar varias horas o días de un trabajo aburrido y tedioso, pues cada juego de datos debe ser creado manualmente, (copiando y pegando en el mejor de los casos), luego los parámetros de los datos deben ser cambiados uno a uno para asegurarnos de que todo funcione correctamente.

La solución a este inconveniente tan molesto, o sea, al problema de contar con un juego de datos lo suficientemente grande y en el menor tiempo posible, es la implementación de un generador de datos que haga todo el trabajo pesado por el programador, que se encargue de manejar automáticamente las relaciones entre las tablas de la base de datos, los tipos de datos, y que genere tantos juegos de datos como sean necesarios para determinar el volumen de información que acepta la base de datos en cuestión.

En la actualidad esa es una práctica muy utilizada por los desarrolladores; y aunque no se ha difundido mucha información sobre el tema, el número de generadores de datos desarrollados sigue creciendo exponencialmente, ya sea para SQL, Oracle, MySQL y muchos otros Sistemas Gestores de Bases de Datos.

Para generar datos automáticamente, es necesario obtener la mayor cantidad de información posible de la base de datos, la cual generalmente es conveniente obtenerla de los metadatos de las tablas de la base de datos y está referida a las tablas y las columnas que se van a poblar durante el proceso de generación,

dicha información se refiere a las restricciones que debe respetar dicho proceso, las reglas que debe cumplir al insertar en la base de datos, el orden de inserción de la base de datos, etc. A partir de la estructura interna de la misma, la utilización de un generador permite generar datos coherentes para realizar pruebas con nuestra base de datos, y evitar encontrar en el sistema nombres como "fdwt55" o "/*45es65", además en dependencia de la herramienta que se utilice existirá la posibilidad de generar datos aleatorios, a partir de determinados orígenes de datos y además controlar algunos aspectos del proceso de generación.

Dentro de un esquema de base de datos no solo se encuentra una colección de tablas y columnas, sino que además contiene otros datos relacionados con las tablas, el concepto de atributos únicos, llave primaria y llaves foráneas no nulas, así como los tipos de datos que también forman parte del esquema de base de datos. Por esta razón, cuando los datos son insertados en el esquema no solo basta con entrar dichos datos, si no que deben ser confirmados según las restricciones presentes en el esquema para luego realizar la inserción de los mismos. Por tanto, los generadores de datos deben garantizar además el manejo de transacciones en la base de datos, chequear las restricciones presentes en la misma, o sea, mantener la estructura de la base de datos en cuestión, para lo cual debe manejar los metadatos, también llamados meta-información. Veamos algunos conceptos que serán útiles para la comprensión del tema.

1.6.2 Transacciones de la Base de datos.

Una transacción no es más que una colección de operaciones que realizan una única función lógica en una aplicación de base de datos. (1)

El servidor de Oracle garantiza la consistencia de los datos con base en transacciones. Las transacciones proporcionan mayor flexibilidad y control cuando los datos cambian y ello asegura la consistencia de los datos en el caso de un fallo en el proceso ya sea del usuario o del sistema. Las transacciones consisten en sentencias DML[2] que componen un cambio consistente en los datos.

[2] Data Manipulation Language

1.6.3 Restricciones

Entre las restricciones que con más frecuencia se presentan en las bases de datos se encuentran:

- Restricciones de unicidad en los valores de una columna.
- Llave primaria para una tabla.
- Llave foránea referenciando otra columna de otra o la misma tabla.
- Restricciones de dominio.

Los mayores esfuerzos en las investigaciones sobre la generación de datos se han centrado en satisfacer estas restricciones mientras se llena la base de datos automáticamente con datos aleatorios. (2)

1.6.3.1 Restricciones de unicidad

Las restricciones de unicidad especifican que los valores de cada fila de una columna dada deben ser únicos, o sea, no se permiten repeticiones. El manejo de las restricciones de unicidad mientras se llenan las tablas requiere recoger todos los valores insertados anteriormente o chequear de alguna manera si el valor a insertar no se encuentra ya en la tabla. Esto puede hacerse de dos formas, la primera es guardando un registro de los valores insertados en la tabla ya sea en la memoria principal o en un archivo, y la segunda, es realizando consultas a la base de datos antes de que cada valor sea insertado para verificar si dicho valor existe o no en la tabla. (2)

1.6.3.2 Restricciones de No-Nulo

Las restricciones de no-nulo especifican que cada fila en la columna especificada debe tener un valor no-nulo. El manejo de este tipo de restricciones es bastante simple, solo es necesario verificar que el valor NULL no es insertado en la columna de la tabla que está marcada con una restricción de no-nulo. (2)

1.6.3.3 Restricciones de Dominio

Las restricciones de dominio especifican los posibles valores de un atributo, por ejemplo se pueden restringir los valores enteros a un rango dado. Este tipo de restricciones debe ser tratado para cada caso en particular. Para valores numéricos cuando es especificado un rango, los valores aleatorios pueden ser generados dentro de ese rango, usando un generador de números aleatorios que confirme la distribución estadística especificada. (2)

1.6.4 Metadatos

El término metadatos no tiene una definición única. Según la definición más difundida metadatos son «datos sobre los datos». Los metadatos pueden describir colecciones de objetos y también los procesos en los que están involucrados, describiendo cada uno de los eventos, sus componentes y cada una de las restricciones que se les aplican. Los metadatos definen las relaciones entre los objetos, como las tuplas en una base de datos o clases en orientación a objetos, generando estructuras. (3)

Hablamos de metadatos o meta-información para referirnos a la información que describe los elementos de la base de datos. Por ejemplo, las columnas de una tabla, los parámetros de un procedimiento almacenado, etc. (4)

1.6.5 Algoritmos de generación de datos.

En la actualidad existe una tendencia a la utilización de algoritmos de generación automática de datos, que permitan automatizar el proceso de inserción de datos de prueba en una base de datos. Algunos de ellos son:

1.6.5.1 KRIMP Generator

Este algoritmo se basa en una tabla de código obtenida a partir de la base de datos original, y la generación de datos se realiza basándose en el código almacenado en dicha tabla, así como analizando las transacciones y restricciones de la base de datos original. (5) Es un algoritmo muy potente, para generar datos parte de una base de datos con información real para realizar el proceso de generación.

1.6.5.2 GADGET[3]

GADGET se basa en la aplicación de un Algoritmo Genético (GA) para resolver el problema de la generación de datos de pruebas. Los algoritmos genéticos son uno de los algoritmos más populares, los mismos son un método de búsqueda basados en los principios de selección natural y genética, de manera general trabajan sobre un grupo de soluciones candidatas de un problema dado, las cuales son evaluadas teniendo en cuenta su habilidad para resolver el mismo. (6)

La ventaja de usar GADGET es que a diferencia de otras soluciones evita quedar atrapado en algún punto sin salida, sin embargo tiene algunas desventajas que deben ser tomadas en cuenta a la de hora de decidir su uso, con GADGET solo las condiciones que son total o parcialmente satisfechas son analizadas, las demás se dejan al azar; además de que hereda también las desventajas de los algoritmos genéticos y de que genera datos para satisfacer un determinado caso de prueba.

[3] Genetic Algorithm for Data Generation Test

1.6.6 Algoritmos de inserción de datos.

Cuando se habla de inserción automática de datos en una base de datos, el principal problema que sale a relucir es la conservación de la consistencia de los datos contenidos en la misma, surge entonces la problemática de cómo garantizar que se respetan las restricciones de integridad de la base de datos, pues la mayoría de las veces una cosa está estrechamente ligada a la otra, puesto que no se les presta la debida atención desde etapas tempranas del desarrollo, lo cual se traduce en un peligro considerable de que se viole la integridad de la base de datos mediante la introducción de inconsistencias.

La utilización de algoritmos de inserción de datos, posibilita un chequeo minucioso y formal de cada una de las restricciones de la base de datos y obliga a su estricto cumplimiento. Teniendo en cuenta que mantener la consistencia de los datos en una base de datos garantiza que la información contenida en ella refleje lo más exacta posible la realidad modelada, es posible clasificar esta actividad como un requisito fundamental.

1.6.7 Generadores de datos.

1.6.7.1 DB Data Generator

DB Data Generator es un generador de datos simple y poderoso, que permite a los desarrolladores poblar las bases de datos fácilmente con miles de columnas de datos significativos y sintácticamente correctos, con el propósito de probar la base de datos, estos datos pueden ser cargados de una base de datos experimental. El mismo lee la base de datos y muestra las tablas y columnas con su configuración de generación de datos. Solo son necesarias unas simples entradas para generar datos de prueba comprensibles.

Los datos generados pueden ser seleccionados de forma aleatoria de la definición del usuario a partir de datos de prueba (como listas definidas por el usuario, archivos CVS[4], datos de tablas de otra base de datos) o generados aleatoriamente.

DB Data Generator automáticamente determina la mejor configuración de generación de datos (método de llenado) para los campos de las tablas. Las configuraciones de los campos dependen del tipo de datos, relaciones con otras tablas (llave foránea), restricciones de no vacíos y otras restricciones de tabla o columna. DB Data Generator puede insertar los datos generados directamente en la base de datos o también podemos elegir ponerlos en un script SQL[5] de tipo insertar. (7)

1.6.7.2 forSQL Data Generator.

La simplicidad y facilidad de uso hacen del programa una herramienta obligada para aquellos desarrolladores que quieran ahorrar esfuerzos y tiempo en pruebas de bases de datos. Puede trabajar con varios servidores usando ODBC[6]. Otra característica notable del Generador de Datos para SQL es su capacidad de llenar campos usando varios métodos diferentes que van desde valores aleatorios hasta llenarlos con datos reales desde plantillas. forSQL Data Generator lee automáticamente la información acerca de la estructura de la base de datos y las llaves primarias, su módulo Generador de Datos de Prueba, es muy utilizado para poblar bases de datos, respetando todas las validaciones de las tablas, usando diferentes métodos de generación de datos como son: números aleatorios, secuencias, rangos de valores, patrones de textos, BLOB[7] y CLOB[8]. (8)

1.6.7.3 Generatedata

Es libre. Es un sistema que funciona en php, javascript y sobre una base de datos MySQL. Genera datos "coherentes", es fácil de agregar opciones, ya que tiene una base de datos propia con datos, los cuales

[4] Concurrent Versioning System
[5] Structured Query Language
[6] Open Database Connectivity
[7] Binary Large Objects
[8] Character Large Object

utiliza en el proceso de generación. Los resultados que genera los puede devolver en HTML[9], EXCEL, CSV[10] y SQL. La versión gratuita permite generar hasta 200 registros de una vez. "Donando" 20 dólares permite generar 5.000 de golpe. (9)

1.6.7.4 DTM DataGenerator v1.22.05

Es un generador de datos automático capaz de crear grandes cantidades de datos, para realizar pruebas a bases de datos a gran escala. Ofrece la posibilidad de crear datos "reales" e insertarlos rápida y automáticamente en una base de datos. Es la herramienta que encabeza el mercado mundial de los generadores de datos. Su principal fortaleza radica en su capacidad para reconocer Llaves foráneas.

Además incluye una herramienta llamada "Rules Wizard", el sistema reconoce el chequeo de las restricciones para los gestores de bases de datos más comunes, las cuales se tienen en cuenta durante el proceso de creación de los datos. Creado por DTM Soft, compañía de consultoría, desarrollo y diseño de software, que se especializa en herramientas para manipulación, conversión y transferencia de datos. (10)

1.6.7.5 TDG 1.2c

Test Data Generator, es suficiente seleccionar una tabla de la base de datos y la herramienta lee la estructura de dicha tabla automáticamente, solo hay que especificar la descripción de los datos que se desea insertar y automáticamente comienza el proceso de generación. La descripción de dichos datos los almacena en un script, de manera que se puede utilizar esos mismos datos generados para varias bases de datos. Ofrece la posibilidad de mostrar los datos actualizados en la base de datos. (11)

1.6.7.6 EMS Data Generator for Oracle

Permite llenar varias tablas de una base de datos Oracle simultáneamente, define tablas y campos para la generación de datos, establece rangos de valores, permite establecer parámetros para tipo de campo,

[9] Hyper Text Markup Language
[10] Comma-separated values

ofrece la posibilidad de ver la información generada previamente, posibilita la utilización de los resultados de consultas SQL como una lista de valores útiles para la generación de nuevos datos, control automático sobre la integridad referencial entre las tablas relacionadas, deshabilita los triggers durante el proceso de generación y guarda los parámetros de generación en un fichero de configuración. (12)

1.6.7.7 Advanced Data Generator

Ha sido probado con Microsoft SQL Server, Oracle, Advantage Database y otros. Es una herramienta que sirve para la generación automática de datos de una manera muy sencilla, entre las características fundamentales que implementa se encuentran las siguientes: los datos que genera están basados en información de la vida real, contenida en un repositorio sin que sea una preocupación la privacidad y seguridad de esos datos, dicho repositorio incluye apellidos en español e inglés, además de una serie de nombres tanto femeninos como masculinos, además de ciudades y calles de diferentes países, además de códigos y nombres de estados. Puede conectarse a cualquier base de datos, y soporta varios tipos de datos, incluyendo BLOB y CLOB. Ofrece la posibilidad de reutilizar proyectos anteriores y la habilidad de ejecutar múltiples proyectos a la vez. Genera miles de registros por segundo, o genera los datos para un Script SQL, archivos CSV, así como a otros formatos. (13)

1.6.7.8 Datatect Generator v1.6

Programa intuitivo y poderoso para la generación de una variedad virtualmente ilimitada de datos de pruebas reales para archivos flat ASCII o directamente para una base de datos Oracle, Sybase, SQL Server , Infomix y Access, usando ODBC. Utiliza una gran variedad de tipos de datos e incluso permite definir tipos de datos personalizados, permite ordenar los diferentes tipos de datos en el formato de salida establecido. Cuenta con una vasta ayuda *on-line* e incluso con tutoriales de ayuda al usuario. (14)

Son muchas las herramientas desarrolladas con el fin de facilitar la inserción de datos de prueba a una base de datos, puesto que ya se ha mencionado que resulta un proceso largo y engorroso. Cada uno de los generadores de datos antes expuestos, resuelven y algunos optimizan, el proceso de generación e inserción de datos, algunos son aplicaciones de escritorio, otros son aplicaciones web, lo cual según las especificaciones del cliente, constituye un inconveniente, sin embargo el inconveniente más relevante de las aplicaciones mencionadas es el hecho de que es necesario abonar un precio para poder utilizarlas, y ¿por qué comprar un producto, bajo el riesgo de que en algún momento aparezca alguna funcionalidad que no pueda satisfacer, aunque sea muy pequeña, si es posible crear uno ajustado a las necesidades particulares de una organización, con esfuerzos y gastos mínimos?

Por esta razón se ha decidido crear una aplicación de escritorio según la propuesta de solución que se expone a continuación.

1.6.8 Propuesta de solución

Se propone la creación de un software que permita minimizar el tiempo de inserción de datos en la base datos, así como disminuir la cantidad de personas que se utilizan para esta labor, reemplazándolas con un sistema generador, que sea capaz de crear datos automáticamente con la mínima cantidad de entradas posibles por parte del usuario. Debe permitir además comprobar hasta qué punto es capaz de soportar la base de datos el manejo de grandes cantidades de datos en condiciones extremas sin que se afecte su rendimiento normal.

1.7 Conclusiones

Teniendo en cuenta las necesidades del equipo de proyecto con respecto a la aplicación y la cantidad de datos generados, en este capítulo se pudo constatar que no es factible utilizar en nuestra universidad ninguna de las soluciones existentes para este tipo de situación, algunas razones están ligadas a la licencia, pues algunos de los generadores existentes son propietarios y se corre el riesgo de que en un futuro surjan nuevas necesidades y al ser una herramienta propietaria no podremos modificarla para satisfacer dichas necesidades; otros son herramientas web, y aunque eso no constituye realmente un inconveniente, se considera innecesario cuando en realidad solo se requiere una herramienta de escritorio para ser utilizada por una persona simultáneamente, algún encargado del proyecto para poblar la base de datos con datos de prueba. Algunas de las herramientas analizadas implementan varias funcionalidades que también resultan innecesarias para la situación que se estudia. Por esta razón se hace necesaria la implementación de una herramienta que permita generar datos de prueba para la base de datos de la aplicación “SIGEP”, y que además ofrezca al usuario la posibilidad de decidir si desea insertar datos a una tabla específica (y que este proceso incluya automáticamente a sus dependencias) o a toda la base de datos de una sola vez.

Luego de haber estudiado varios algoritmos de generación de datos, como los antes expuestos y a pesar de que ambos son robustos y con grandes aplicaciones, en la situación que se analiza no se hace necesaria la aplicación de un algoritmo de esa complejidad, puesto que se trabajará con tipos de datos estándares de manera que el proceso de generación se pueda realizar de forma sencilla, se ha decidido el uso de un algoritmo de generación de datos creado por los autores de este trabajo.

Capítulo 2: Descripción de la solución propuesta

2.1 Introducción

Para el logro exitoso de una aplicación, es indispensable utilizar metodologías y herramientas de desarrollo de software, por lo que es necesario tener un mínimo conocimiento de las mismas. A continuación se detallan los conceptos, definiciones y características principales de las metodologías y herramientas utilizadas en el desarrollo de la herramienta que se pretende construir.

Una metodología es un proceso de software detallado y completo. Las mismas se basan en una combinación de los modelos de proceso genéricos, además debe definir con precisión los artefactos, roles y actividades involucrados en el proceso de desarrollo, junto con prácticas y técnicas recomendadas, guías de adaptación al proyecto, guías para uso de herramientas de apoyo, etc.

Para el diseño de la solución se seguirá la metodología RUP[11] basada en el lenguaje de modelado UML[12], haciendo uso de la herramienta CASE[13] Visual Paradigm, puesto que es muy poderosa e integra perfectamente con UML.

Teniendo en cuenta las condiciones en las cuales deberá usarse este software, y las características del entorno de aplicación, se realizará una herramienta de escritorio desarrollada en el lenguaje de programación Java. El sistema gestor de base de datos que se utilizará será Oracle, puesto que es el utilizado actualmente en la aplicación para la cual se construye esta herramienta y la conexión a la base de datos será mediante JDBC[14].

[11] Rational Unified Process
[12] Unified Modeling Language
[13] Computer Aided Software Engineering
[14] Java Data Base Connectivity

2.2 Técnicas y herramientas

2.2.1 RUP

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes, en ella se define claramente "quién" debe hacer "qué", "cómo" y "cuándo" durante el ciclo de vida del proyecto, tiene como principal característica que es una metodología guiada por casos de uso (CU), iterativa e incremental y centrada en la arquitectura.

RUP divide el proceso de desarrollo en cuatro fases: "Inicio" con el objetivo de determinar la visión del proyecto, "Elaboración" donde se determina la arquitectura óptima, "Construcción" donde se obtiene la capacidad operacional inicial del sistema y "Transición" que no es más que obtener una versión funcional del proyecto.

Cada fase es desarrollada por ciclos de iteraciones donde el objetivo de cada una depende de las iteraciones que le preceden, en todas las fases se destacan diferentes flujos de trabajo como es el Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Instalación, Administración de Configuración y Cambio, Administración del Proyecto y Ambiente.

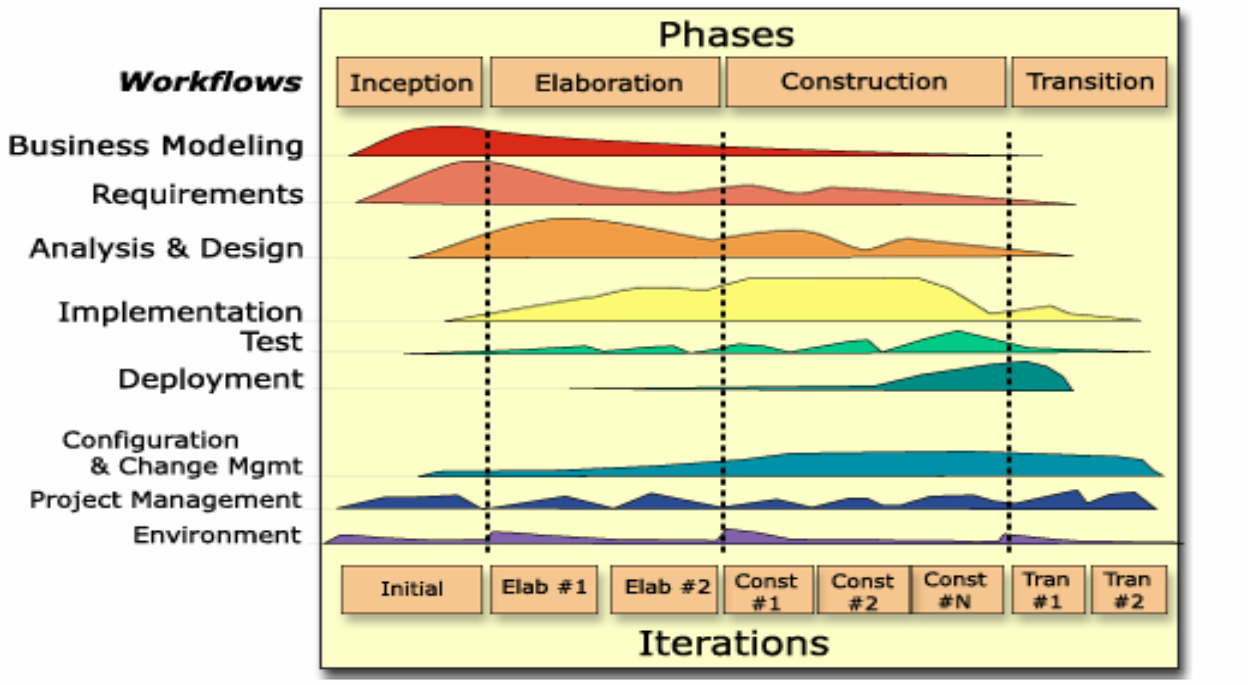

Fig. 1 Fases de RUP

2.2.2 UML

Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas.

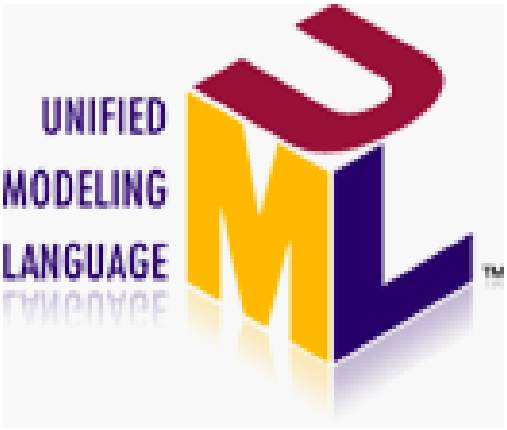

Fig. 2 Lenguaje de Modelado Unificado

2.2.3 Visual Paradigm

Visual Paradigm es una herramienta CASE, soporta el ciclo de vida completo del desarrollo de software y permite a varios usuarios trabajar sobre el mismo proyecto. Dentro de las características del Visual Paradigm se destacan su robustez, usabilidad y portabilidad. Permite realizar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta apoya las notaciones de Java y UML, y está diseñada para un amplio grupo de usuarios, incluyendo Ingenieros de software, Analistas de sistema, Analistas de negocio y Arquitectos de sistema, usuarios que están interesados en realizar software a grandes escalas y siguiendo el paradigma de la programación orientada a objeto.

Fig. 3 Visual Paradigm

2.2.4 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principio de los años 90´s. La plataforma J2EE[15] define una estándar para desarrollar aplicaciones empresariales, simplificando el desarrollo de estas basándose en componentes modulares y estandarizados, permite manejar muchos detalles del comportamiento de las aplicaciones sin necesidad de complejas líneas de código. Proporciona además un conjunto de clases potente y flexible, no posee punteros, además en caso de que se quiera ejecutar el programa en otra máquina no es necesario re-escribir el código, también tiene algunas desventajas como por ejemplo su velocidad, debido a que los programas hechos en java son interpretados, su velocidad se ve un poco reducida con respecto a otros lenguajes.

Fig. 4 Java

2.2.5 Oracle

La compañía es la líder mundial como distribuidora de programas para administración de la información, y la segunda compañía de software independiente del mundo. Oracle es uno de los Sistemas Gestores de Bases de Datos (SGBD) más comunes y poseedor de las características más avanzadas. Está compuesto por un grupo de procesos que se ejecutan en el sistema operativo. Estos procesos administran cómo son guardados los datos y cómo acceder a ellos, se ejecuta en segundo plano, manteniendo y ubicando la información en el disco duro.

[15] Java 2 Enterprise Edition

2.2.6 JDBC

El API[16] JDBC está diseñado como una interfaz para ejecutar sentencias SQL, y no como una etiqueta con un alto nivel de abstracción para el acceso a los datos, por tanto, como no está diseñado para mapear automáticamente las clases Java con filas en una base de datos, permite escribir una gran cantidad de aplicaciones sin necesidad de preocuparse por las características particulares de la base de datos que se está usando, o sea JDBC es independiente de la base de datos que se use y no hay necesidad de hacer re-ingeniería para una base de datos especifica. Desde el punto de vista del usuario final, una aplicación Java usando JDBC sería como la figura que se muestra a continuación.

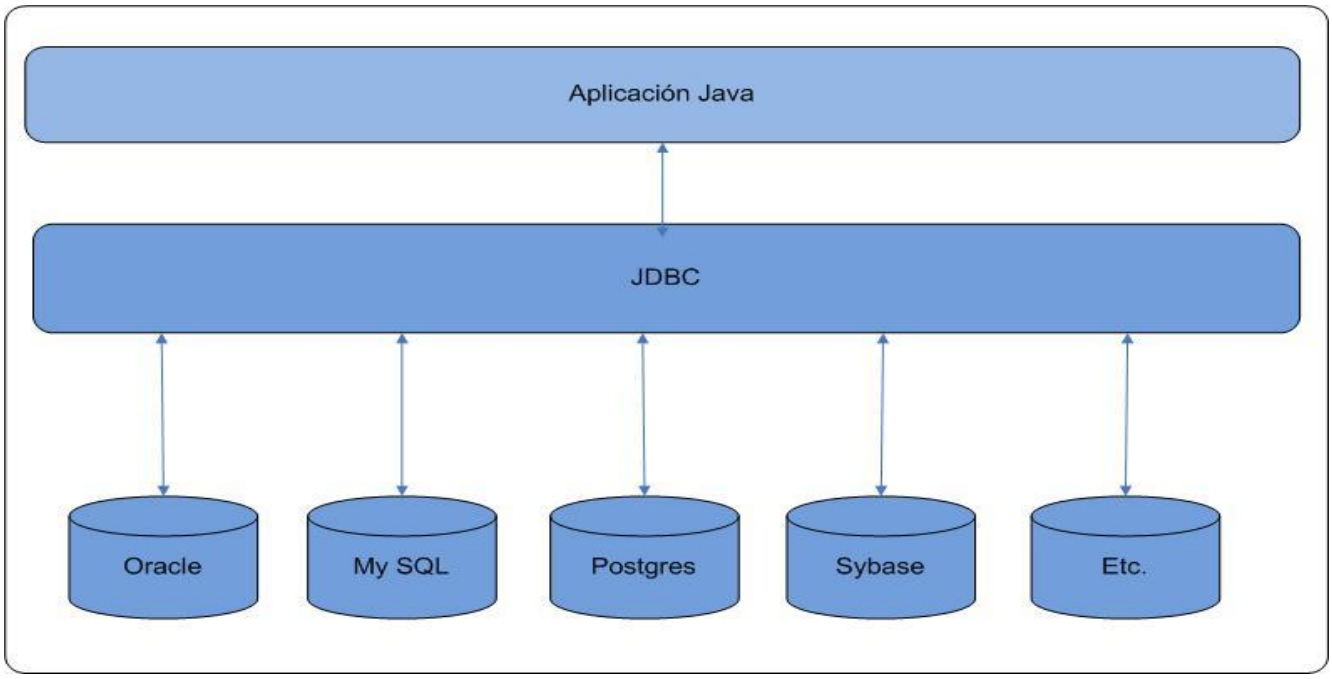

Fig. 5 JDBC

JDBC opera con una gran variedad de sistemas gestores de bases de datos relacionales debido a que posee una interfaz para cada base de datos específica, conocida como driver, este maneja el mapeo de las llamadas de los métodos Java en las clases JDBC a la base de datos. La arquitectura de JDBC está

[16] Application Programming Interface

basada en una colección de interfaces y clases Java que unidas permiten realizar conexiones a bases de datos, crear y ejecutar sentencias SQL, y retornar y modificar información de la base de datos, como se muestra en la figura siguiente.

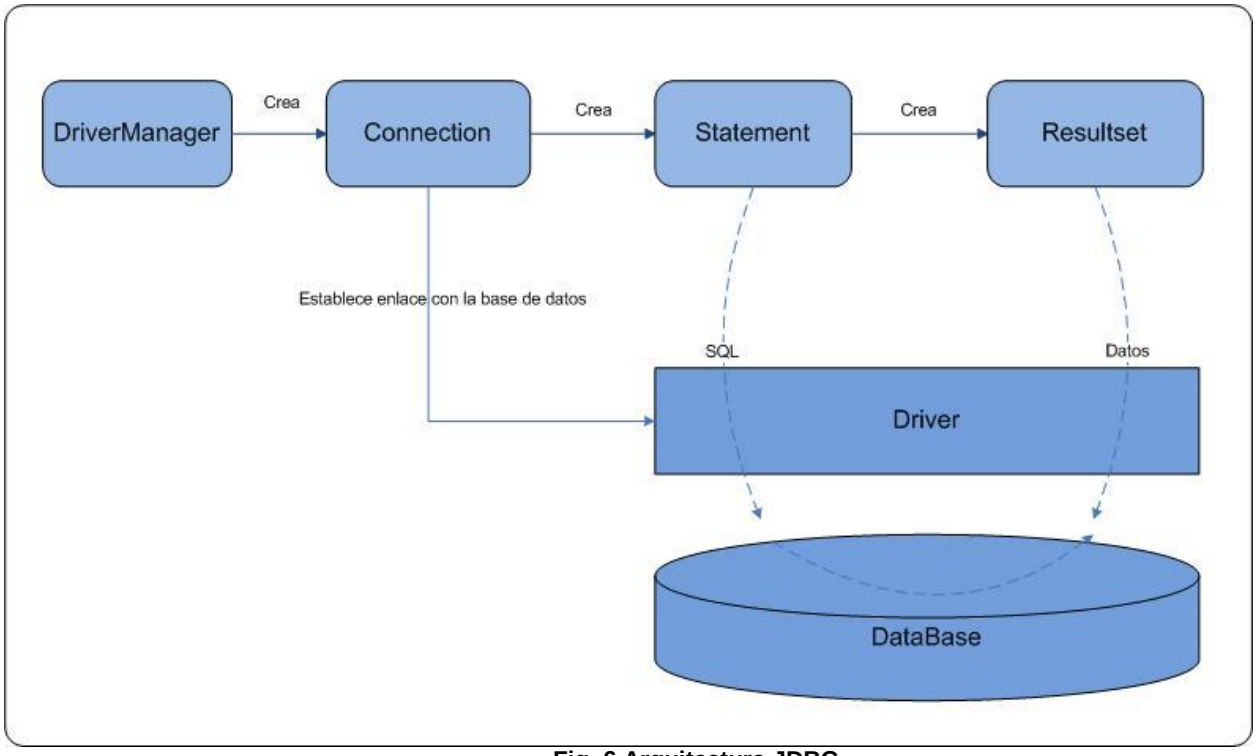

Fig. 6 Arquitectura JDBC

2.3 Requerimientos

Antiguamente, cuando se deseaba realizar un software, en muchas ocasiones existía el problema de que el cliente solicitaba el producto que necesitaba y cada miembro del equipo de desarrollo se quedaba con una versión totalmente diferente, lo que llevaba a que la versión final del producto no coincidiera con lo que realmente quería el cliente y en muchas ocasiones no satisfacía todas sus necesidades, o el cliente

no sabía explicar bien lo que deseaba o no lo sabía a ciencia cierta, lo que hacía que el equipo de desarrollo perdiera horas de trabajo por desacuerdo o insatisfacción con los clientes.

En la actualidad, todo es muy distinto, existen múltiples metodologías de desarrollo de software, las cuales dan vital importancia a las fases iniciales del desarrollo, RUP específicamente propone la realización de una captura de requisitos, que no es más que llegar a un acuerdo con el cliente sobre lo que desea y las actividades que debe realizar el sistema para satisfacer sus necesidades.

Algunas definiciones de requerimientos según la IEEE[17] son las siguientes:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Es una característica que un sistema debe tener para cubrir alguna de las necesidades que lo motivan.

2.3.1 Requerimientos funcionales

Los requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. Por lo general se describen mejor a través del modelo de Casos de uso y los Casos de uso como tal. Por lo tanto los requerimientos funcionales especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto.

Los requerimientos funcionales del sistema desarrollado por los autores de esta investigación son:

RF1 Establecer la conexión con la base de datos.

RF2 Eliminar la conexión con la base de datos.

[17] Institute of Electrical and Electronics Engineers

RF3 Chequear las restricciones presentes en la base de datos.

RF4 Generar tantos registros como el usuario necesite.

RF5 Persistir los registros en la base de datos.

RF6 Eliminar los registros de la base de datos.

RF7 Mostrar información referente a la base de datos.

2.3.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez que se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Teniendo en cuenta las necesidades del cliente, así como el nivel tecnológico que posee el mismo, y las características del entorno en que se desarrolla esta aplicación, se definen las siguientes propiedades o cualidades que dicha aplicación debe poseer.

1. Requerimientos de Software
    a. El sistema debe desarrollarse para el Sistema Operativo Windows XP pues es el que utiliza el equipo de desarrollo.
    b. El sistema deberá ejecutarse en una máquina donde se encuentre instalada la máquina virtual de java.
2. Requerimientos de Hardware
    a. El sistema deberá ejecutarse en una máquina con las siguientes propiedades mínimas: procesador Pentium 166MHz o más rápido con al menos 32 megabytes de memoria

RAM[18]. Se requiere además 70 megabytes de espacio libre en disco para la instalación de la Máquina Virtual de Java. Si se desea instalar la documentación de la misma, se necesitan 120 megabytes adicionales de espacio libre.

3. Restricciones en el diseño y la implementación
    a. El sistema deberá implementarse en lenguaje de programación Java, sobre la plataforma J2EE, pues es la utilizada para el desarrollo de la aplicación.
4. Requerimientos de apariencia o interfaz externa
    a. El sistema deberá desarrollarse según las pautas de diseño definidas por el Equipo de Diseño Gráfico en la Universidad, con los colores gris o azul que son los que están presentes tanto en el Sistema de Gestión Penitenciaria como en el Portal de la aplicación.
    b. Deberá poseer una interfaz sencilla y fácil de comprender.

Una vez definidos cuales son tanto las capacidades como las propiedades que debe poseer la aplicación, se definen los siguientes actores del sistema.

2.4 Actores del Sistema

Cada trabajador del negocio (incluso si es un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

- No son parte de él.
- Pueden intercambiar información con él.
- Pueden ser un recipiente pasivo de información.
- Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado.

[18] Random Access Memory

Actores	Descripción
Usuario	Es quien tiene acceso al sistema, mediante el cual accede a toda la información de una determinada base de datos e inicia el proceso de generación de datos para la población de la misma.

Tabla 1 Actores del Sistema

2.5 Diagrama de Casos de Uso del Sistema

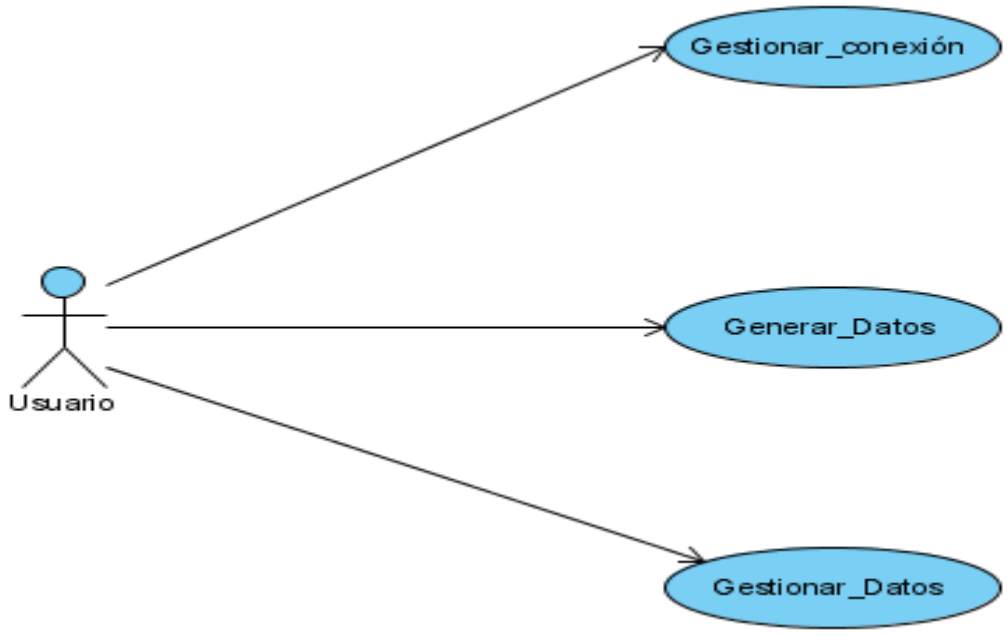

Fig. 7 Diagrama de Casos de Uso del Sistema

2.6 Casos de uso expandidos

A continuación se realiza un análisis más detallado de los casos de uso del sistema. Este análisis consiste en describir ciertos parámetros del caso de uso basándose en la plantilla disponible para este fin. (**Anexo 1**)

Caso de Uso	Gestionar Conexión
Actor	Usuario
Propósito	Conectarse a una base de datos, para obtener información sobre la misma e iniciar el proceso de población automática.
Resumen	El caso de uso inicia cuando el usuario selecciona la opción crear conexión, cerrar conexión y termina cuando el sistema muestre un mensaje explicando que se ha llevado a cabo la operación.
Precondiciones	
Poscondiciones	
Referencia	RF1, RF2
Casos de Uso Relacionados	
Sección	Crear conexión.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción crear conexión. 3. El usuario inserta los datos de la conexión.	2. El sistema muestra una interfaz para que el usuario inserte los datos necesarios para realizar la conexión, base de datos, puerto, usuario y contraseña. 4.1 El sistema verifica que no haya ningún campo vacío. 4.2 El sistema verifica que los datos sean correctos. 4.3 El sistema muestra un mensaje indicando que la conexión ha sido satisfactoria.
Flujos Alternos	
	4.1 El sistema verifica que algún campo está vacío y por lo tanto muestra un mensaje de error: “Debe llenar todos los campos.” 4.2 El sistema verifica que algún dato es incorrecto y muestra un mensaje de error: “Los datos son incorrectos.”

Prioridad	Crítico
Sección	Cerrar conexión.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción cerrar conexión.	2. El sistema verifica que está conectado a una base de datos. 3. El sistema cierra la conexión y le muestra un mensaje al usuario haciéndole saber que se ha desconectado de la base de datos.
Flujos Alternos	
	2.1 El sistema verifica que no está conectado a una base de datos y muestra un mensaje de error indicando que no existe ninguna conexión a una base de datos.
Prioridad	Crítico

Tabla 2 Caso de Uso <<Crear Conexión>>

Caso de Uso	Generar datos
Actor	Usuario
Propósito	El sistema debe generar datos automáticamente y luego poblar la base de datos.
Resumen	El caso de uso inicia cuando el usuario selecciona la opción generar datos, y termina cuando el sistema inserta los datos generados en la base de datos.
Precondiciones	El sistema debe estar conectado a una base de datos.
Poscondiciones	
Referencia	RF3, RF4, RF5
Casos de Uso Relacionados	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El usuario selecciona la opción generar datos. 3. El usuario establece la cantidad de registros a generar. 6. El usuario define las reglas para la generación de datos, teniendo en cuentas las restricciones presentes las tablas que el sistema le muestra.	2. El sistema muestra una interfaz con las tablas de la base de datos. 4. El sistema busca las restricciones presentes en las tablas que se van a poblar. 5. El sistema muestra las tablas y las restricciones presentes en cada una de ellas. 7. El sistema comienza la generación de los datos. 8. Cada vez que el sistema ha generado cien registros, inserta los mismos en la base de datos, a la vez que continúa generando otros registros, hasta llegar a la cantidad especificada por el usuario.
Flujos Alternos	
Prioridad	Crítico

Tabla 3 Caso de Uso <<Generar Datos>>

Caso de Uso	Gestionar datos
Actor	Usuario
Propósito	El sistema debe ser capaz de mostrar información sobre la base de datos a la cual se conecta, como las tablas y sus columnas, además debe mostrar las propiedades de la base de datos, así como de eliminar los datos insertados en la misma.
Resumen	El caso de uso inicia cuando el usuario selecciona la opción mostrar tablas, propiedades, eliminar datos, y termina cuando el sistema muestra el listado de tablas de la base de datos, las propiedades de la misma o toda la información almacenada es eliminada.
Precondiciones	El sistema debe estar conectado a una base de datos.

Poscondiciones	
Referencia	RF6, RF7
Casos de Uso Relacionados	
Sección	Mostrar tablas.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción mostrar tablas.	2. El sistema verifica que exista una conexión a una base de datos. 3. El sistema muestra una interfaz con las tablas de la base de datos y sus columnas.
Flujos Alternos	
	2.1 El sistema no está conectado a una base de datos, por tanto envía un mensaje de error haciéndoselo saber al usuario.
Prioridad	Crítico
Sección	Eliminar datos
Flujo Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción eliminar datos. 4. El actor selecciona Sí.	2. El sistema verifica que exista una conexión a una base de datos. 3. El sistema muestra un mensaje al usuario preguntando si esta seguro que desea eliminar la información almacenada en la base de datos. 5. El sistema elimina toda la información almacenada en la base de datos.
Flujos Alternos	
4.1 El actor selecciona No.	2.1 El sistema no está conectado a una base de datos, por tanto envía un mensaje de error haciéndoselo saber al usuario.
Sección	Propiedades
Flujo Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción Propiedades.	2. El sistema verifica que exista una conexión a una base de datos. 3. El sistema muestra una interfaz con las propiedades de la base de datos,

	como nombre del producto, versión, driver, versión del driver, dirección de la base de datos, el usuario con el que se accede a la misma, etc.
	2.1 El sistema no está conectado a una base de datos, por tanto envía un mensaje de error haciéndoselo saber al usuario.
Prioridad	Auxiliar

Tabla 4 Caso de Uso <<Gestionar Datos>>

2.7 Conclusiones

En este capítulo quedaron reflejadas las principales características relacionadas con la herramienta y los puntos a tener en cuenta en el sistema. Se trataron además todas las herramientas utilizadas en la creación de la solución propuesta, así como la metodología de desarrollo de software a seguir y el lenguaje de programación a utilizar. Se describieron además de forma breve los casos de uso, que sirvieron como punto de partida para la construcción del sistema, siempre teniendo en cuenta el planteamiento de los principales requisitos.

Capítulo 3: Análisis y diseño de la solución propuesta

3.1 Introducción

En el presente capítulo se traducen las necesidades del cliente en requerimientos y casos de uso del sistema, mediante la captación de las actividades fundamentales que el sistema debe realizar para satisfacer las necesidades del cliente.

El modelado del negocio propuesto por RUP como parte inicial del proceso de desarrollo de software no se lleva a cabo en el presente trabajo, debido a que se identificó un solo proceso y de poca complejidad en cuanto a su comprensión y dominio, el cual consiste en que los desarrolladores necesitan introducir gran cantidad de datos al sistema para verificar si realmente cumple con los requisitos planteados.

En el desarrollo del presente capítulo se verán las investigaciones realizadas para dar solución al problema de los clientes, se verán además los diagramas correspondientes al análisis y diseño de la solución propuesta, obteniendo de esta manera las clases y objetos necesarios para la implementación del sistema.

3.3 Análisis del Sistema

Hasta el momento solo se han definido las actividades fundamentales que debe realizar el software para satisfacer las necesidades del usuario final, obteniendo de esta forma una visión general y abstracta del sistema visto desde fuera. En lo adelante, el análisis estará basado fundamentalmente en los requisitos funcionales del sistema, los cuales serán transformados mediante la implementación propia del mismo, dicho análisis consiste en obtener una visión del sistema que se centra en determinar qué hace, de modo que sólo se tienen en cuenta los requisitos funcionales. A continuación se destacan los principales diagramas de este flujo de trabajo.

3.3.1 Diagramas de clases del análisis

Las clases del análisis son la representación de conceptos y relaciones del dominio del problema, las cuales pueden ser clasificadas en tres tipos:

- Interfaces

- Controladoras
- Entidades

Las clases interfaces son aquellas mediante las cuales el usuario interactúa con el sistema, las controladoras que definen el flujo de control y las transacciones dentro de un caso de uso (o varios de ellos) delegando el trabajo a otros objetos, y finalmente las clases entidades son aquellas que poseen información vital para el sistema, que debe ser persistente y duradera. A continuación se presentan más detallados los diagramas de clases del análisis para cada caso de uso del sistema, en los cuales se representan los conceptos del dominio del problema, o sea es una representación de las cosas del mundo real, no de la implementación automatizada de las mismas.

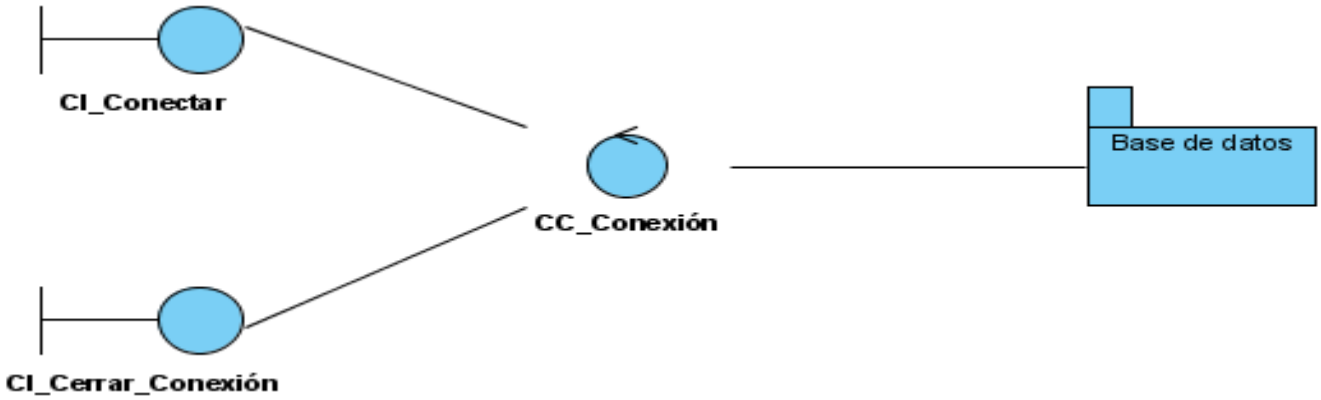

Fig. 8 Diagrama de clases del Análisis (CUS: Crear Conexión)

Fig. 9 Diagrama de clases de Análisis (CUS: Generar Datos)

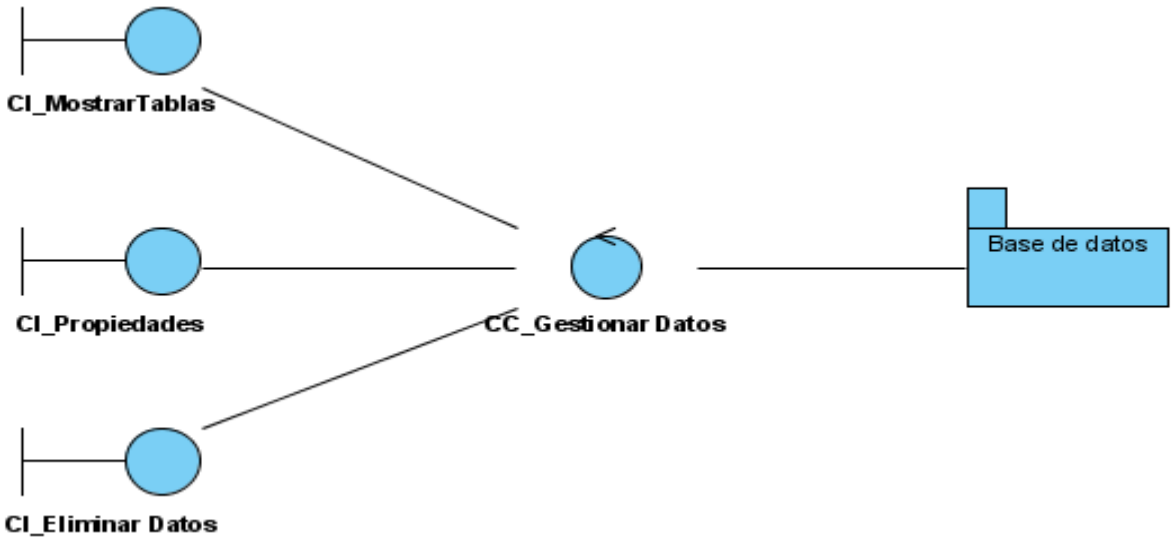

Fig. 10 Diagrama de clases del diseño (CUS: Gestionar Datos)

3.3.2 Diagramas de interacción

Los diagramas de interacción muestran la relación entre objetos y junto a ellos los mensajes enviados. Este tipo de diagrama se utiliza para modelar los aspectos dinámicos de un sistema.

Existen dos tipos de diagramas de interacción, los diagramas de secuencia y los de colaboración; los diagramas de secuencia muestran las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto y los de colaboración son una forma alternativa al diagrama de secuencia de mostrar un escenario, este tipo de diagrama muestra las interacciones entre objetos organizadas en torno a los objetos y los enlaces entre ellos. A continuación se muestran los diagramas de secuencia para cada sección de cada caso de uso del sistema.

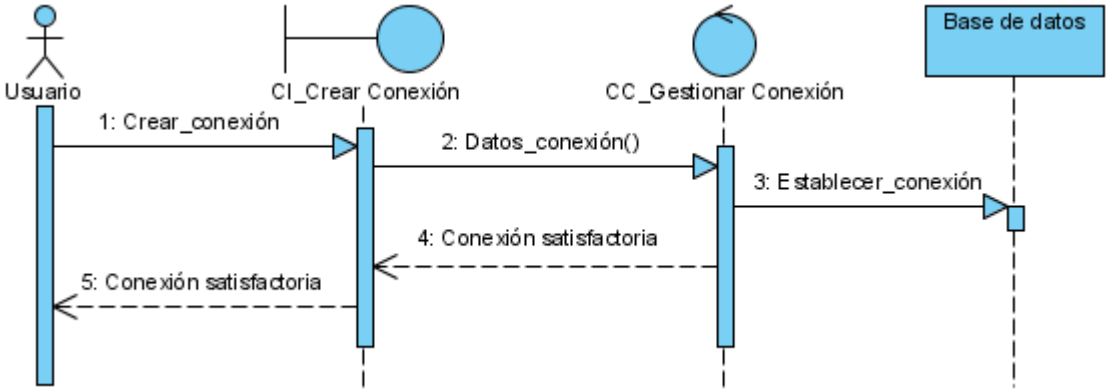

Fig. 11 Diagrama de secuencia (CUS: Gestionar Conexión Sección: Crear Conexión)

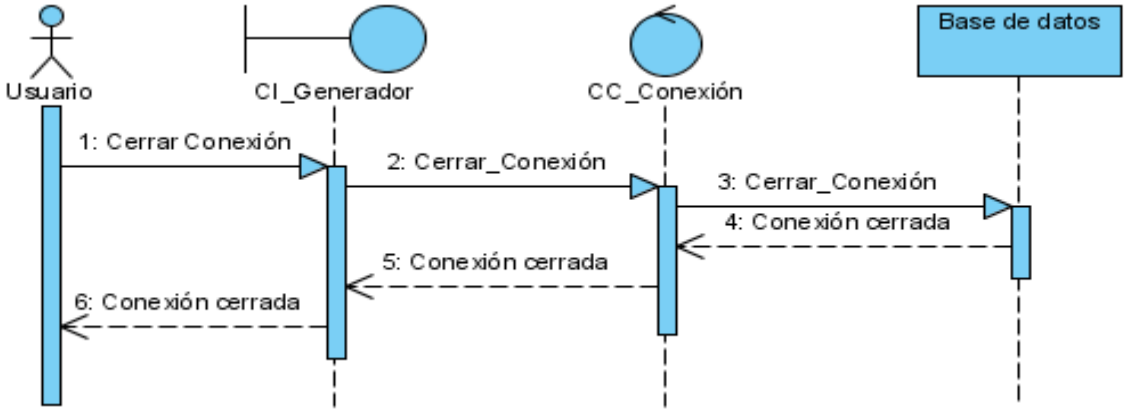

Fig. 12 Diagrama de secuencia (CUS: Gestionar Conexión Sección: Cerrar Conexión)

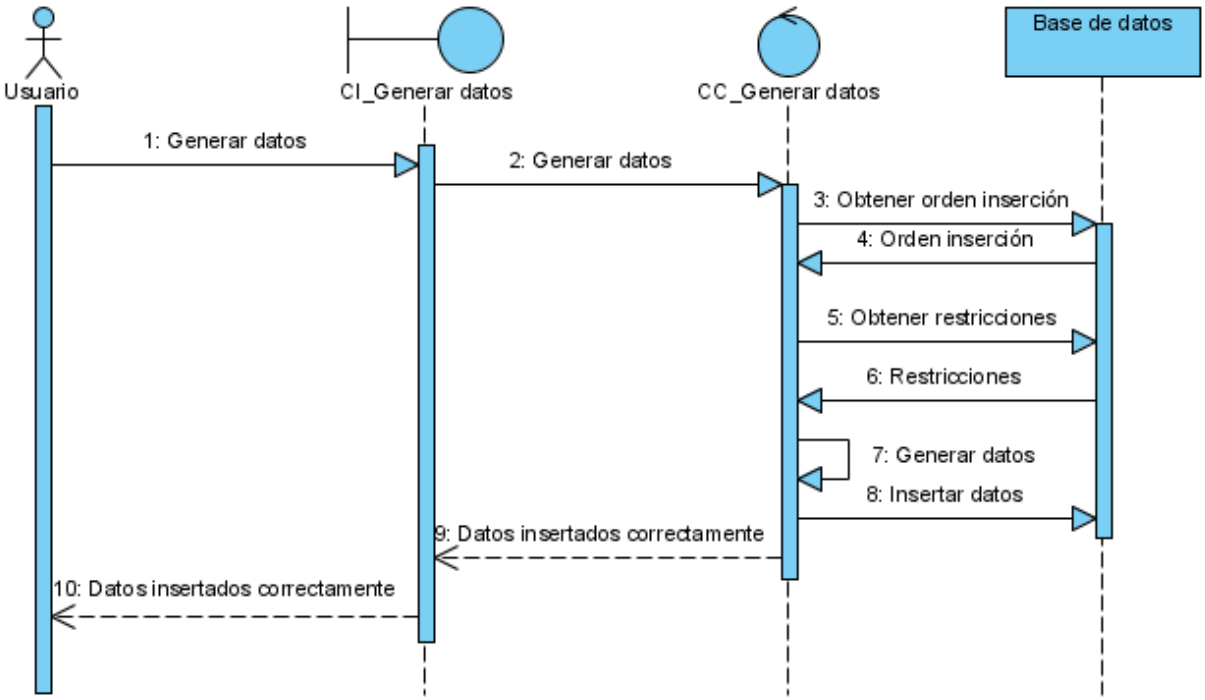

Fig. 13 Diagrama de secuencia (CUS: Generar Datos)

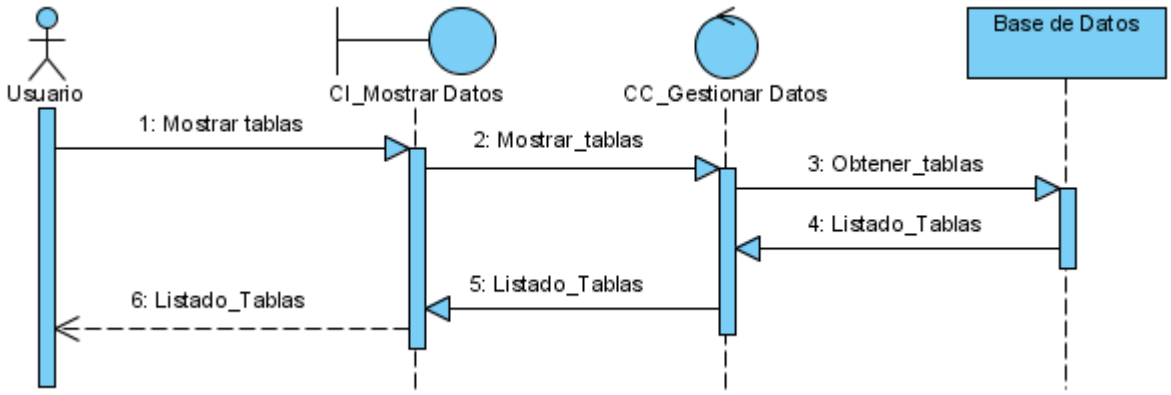

Fig. 14 Diagrama de secuencia (CUS: Gestionar Datos Sección Mostrar Tablas)

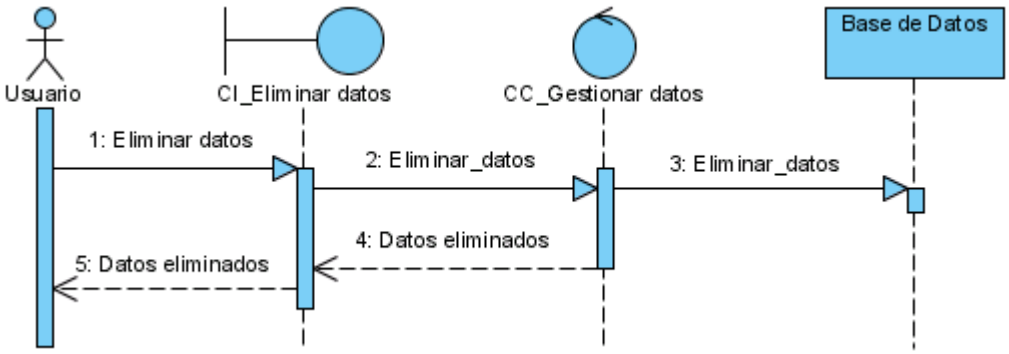

Fig. 15 Diagrama de secuencia (CUS: Gestionar Datos Sección Eliminar Datos)

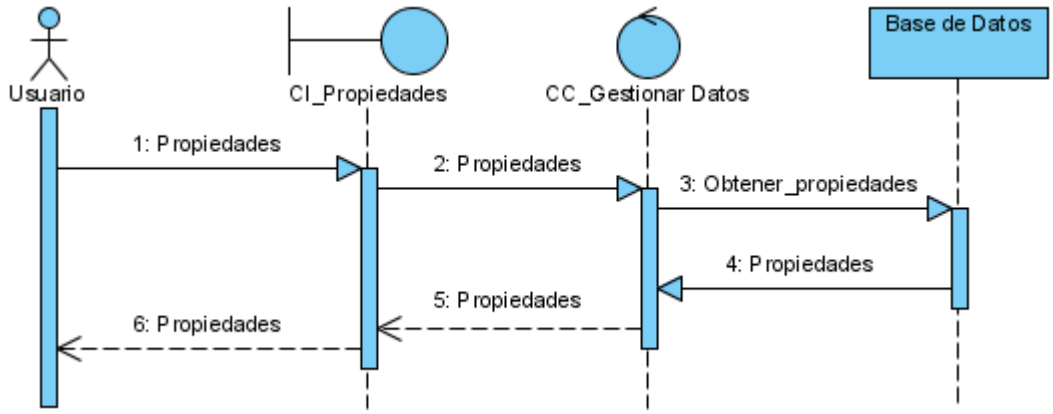

Fig. 16 Diagrama de secuencia (CUS: Gestionar Datos Sección Propiedades)

3.4 Diseño del Sistema

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, es en definitiva cómo cumple el sistema sus objetivos y además debe ser lo suficientemente claro como para que el sistema pueda ser implementado sin ambigüedades.

3.4.1 Diagrama de clases del diseño

El diagrama de clases del diseño representa la parte estática del sistema, las clases y sus relaciones y muestra cómo puede ser construido. Son importantes para visualizar, especificar y documentar modelos estructurales y construir sistemas ejecutables.

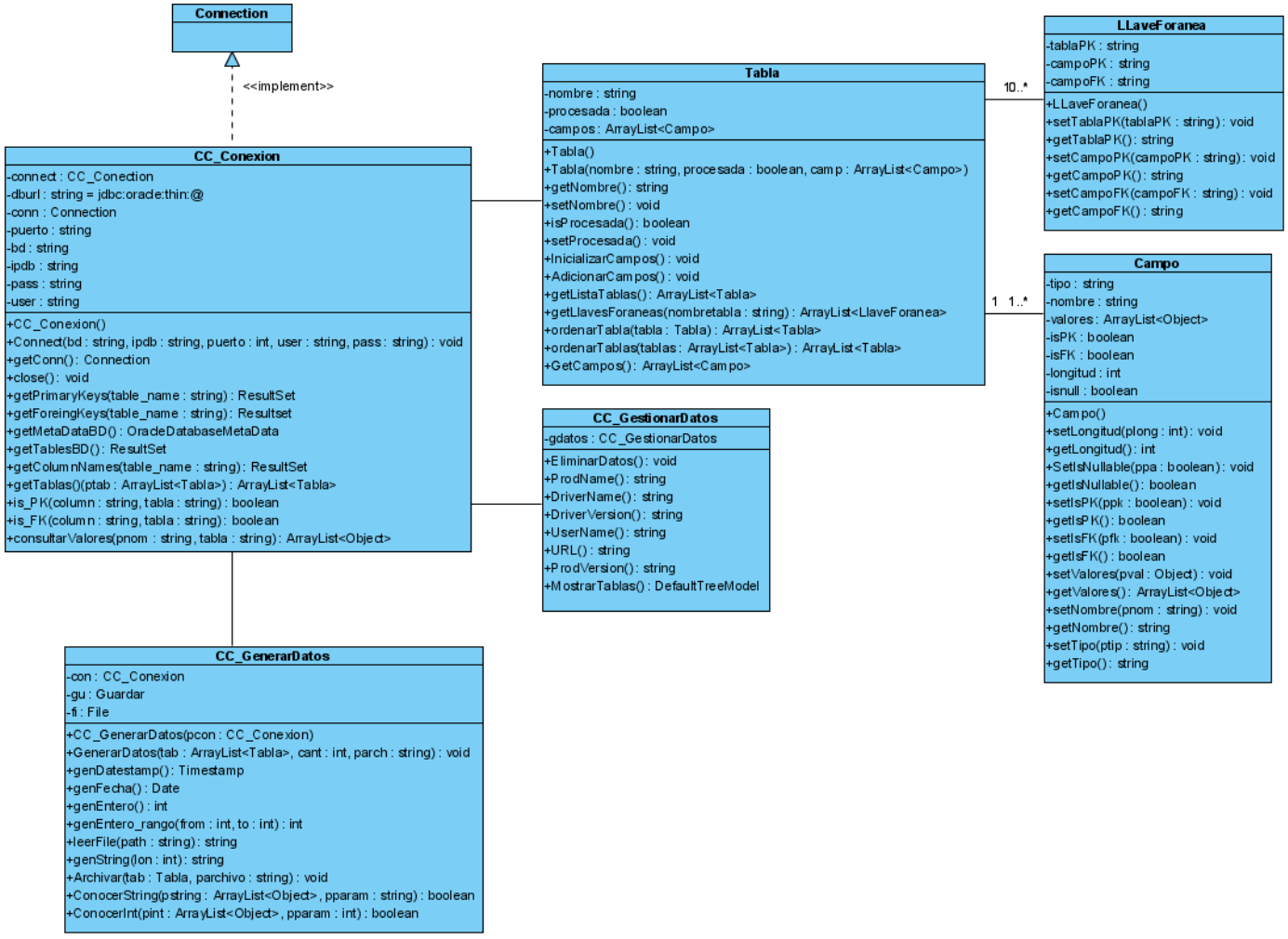

Fig. 17 Diagrama de clases del diseño

3.4.2 Descripción de las clases de diseño

Nombre: CC_Conexion	
Tipo de clase: Controladora	
Atributo	**Tipo**
dburl	String
connect	CC_Conexion
conn	Conection

puerto	Int
bd	String
ipdb	String
user	String
pass	String
Responsabilidad:	
Nombre:	CC_Conexion()
Descripción	Constructor sin parámetros.
Nombre	Connect(String bd, String ipdb, int puerto, String user, String pass):void
Descripción	Conecta la herramienta a una base de datos
Nombre:	getConn():Connection
Descripción	Devuelve el objeto conexión.
Nombre	close():void
Descripción	Cierra la conexión a la base de datos, en el caso de que exista la misma.
Nombre	getPrimaryKeys(String table_name):ResultSet
Descripción	Devuelve un resultset con la descripción de las llaves primarias de una tabla.
Nombre	getForeignKeys(String table_name):ResultSet
Descripción	Devuelve un resultset con la descripción de las llaves foráneas de una tabla.
Nombre	getMetaDataBD():OracleDatabaseMetaData
Descripción	Devuelve los metadatos de la base de datos con que se trabaja.
Nombre	getTablesBD():ResultSet
Descripción	Devuelve un resultset con la descripción de las tablas pertenecientes al usuario conectado a la base de datos.
Nombre	getColumnNames (String table_name):ResultSet
Descripción	Devuelve un resultset con la descripción de las columnas una tabla de la base de datos.
Nombre	getTablas(ArrayList<Tabla> ptab): ArrayList<Tabla>
Descripción	Devuelve una lista con la descripción de las tablas de la base de datos.
Nombre	is_PK(String column,String tabla):Boolean
Descripción	Retorna si una columna es llave primaria o no.
Nombre	is_FK(String column,String tabla)
Descripción	Retorna si una columna es llave foránea o no.
Nombre	consultarValores(String pnom, String tab): ArrayList<Object>
Descripción	Retorna la lista de valores de un campo de una tabla dada.
Nombre	createStatement():Statement
Descripción	Crea un objeto Statement.

Nombre: Tabla	
Tipo de clase: Auxiliar	
Atributo	**Tipo**
tabla	Tabla
nombre	String
procesada	boolean
campos	ArrayList<Object>
Responsabilidad:	
Nombre	Tabla()
Descripción	Constructor sin parámetros.
Nombre	Tabla(String nom, boolean proc, ArrayList<Campos> camp)
Descripción	Constructor con parámetros.
Nombre	getNombre():String
Descripción	Retorna el nombre de una tabla.
Nombre	setNombre(String nom):void
Descripción	Establece el nombre de una tabla.
Nombre	isProcesada():Boolean
Descripción	Retorna si una tabla esta procesada o no.
Nombre	isProcesada(boolean proc):void
Descripción	Establece una tabla a procesada o no.
Nombre	inicializarCampos():void
Descripción	Inicializa la lista de campos de una tabla.
Nombre	adicionarCampos(Campos camp):void
Descripción	Adiciona una campo a la lista de campos de una tabla.
Nombre	getListaTablas():ArrayList<Tabla>
Descripción	Devuelve la lista de tablas de la base de datos.
Nombre	getLlavesForaneas(String nombreTabla): ArrayList<LlaveForanea>
Descripción	Devuelve la lista de llaves foráneas de una tabla de la base de datos.
Nombre	ordenarTabla(Tabla tabla):ArrayList<Tabla>
Descripción	Devuelve la lista de tablas que necesitan llenarse antes de llenar una tabla pasada como parámetro.
Nombre	ordenarTablas(ArrayList<Tabla> tablas):ArrayList<Tabla>
Descripción	Devuelve la lista de tablas de la base de datos ordenadas de forma tal que puedan insertarse los datos en la misma.

Nombre: Campo

Tipo de clase: Auxiliar	
Atributo	**Tipo**
nombre	String
tipo	String
valores	ArrayList<Object>
isPK	boolean
isFK	boolean
longitud	int
isnullable	boolean
Responsabilidad:	
Nombre	Campos()
Descripción	Constructor sin parámetros.
Nombre	setLongitud(int plong):void
Descripción	Establece la longitud de un campo.
Nombre	getLongitud():int
Descripción	Retorna la longitud de un campo.
Nombre	setIsNullabe(boolean ppa):void
Descripción	Establece si un campo es nullable o no.
Nombre	getIsNullable():Boolean
Descripción	Retorna si un campo es nullable o no.
Nombre	setIsPK(boolean ppk):void
Descripción	Establece si un campo es llave primaria o no.
Nombre	getIsPK():boolean
Descripción	Retorna si un campo es llave primaria o no.
Nombre	setIsFK(Boolean pfk):void
Descripción	Establece si un campo es llave foránea o no.
Nombre	getIsFK():boolean
Descripción	Retorna si un campo es llave foránea o no.
Nombre	setValores(Object pval):void
Descripción	Establece un valor a un campo.
Nombre	getValores():ArrayList<Object>
Descripción	Retorna la lista de valores de un campo.
Nombre	setNombre(String pnom):void
Descripción	Establece el nombre de un campo.
Nombre	getNombre():String
Descripción	Retorna el nombre de un campo.
Nombre	setTipo(String ptip):voi
Descripción	Establece el tipo de un campo.
Nombre	getTipo():String
Descripción	Retorna el tipo de un campo.

Nombre: LlaveForanea	
Tipo de clase: Auxiliar	
Atributo	**Tipo**
tablaPK	String
campoPK	String
campoFK	String
Responsabilidad:	
Nombre	LLaveForanea()
Descripción	Constructor sin parámetros.
Nombre	setTablaPK(String tablaPK):void
Descripción	Establece la tabla desde donde viene una llave foránea.
Nombre	getTablaPK ():String
Descripción	Retorna la tabla desde donde viene una llave foránea.
Nombre	setCampoPK(String campoPK):void
Descripción	Establece la llave primaria que está siendo importada.
Nombre	getCampoPK():String
Descripción	Retorna la llave primaria que está siendo importada.
Nombre	setCampoFK(String campoFK):void
Descripción	Establece la llave foránea.
Nombre	getCampoFK():String
Descripción	Retorna la llave foránea.

Nombre: CC_GestionarDatos	
Tipo de clase: Controladora	
Atributo	**Tipo**
gdatos	CC_GestionarDatos
Responsabilidad:	
Nombre	CC_GestionarDatos()
Descripción	Constructor sin parámetros.
Nombre	EliminarDatos():void
Descripción	Elimina la información almacenada en la base de datos.
Nombre	ProdName ():String
Descripción	Retorna el nombre del producto.
Nombre	DriverName ():String
Descripción	Retorna el nombre del driver.
Nombre	DriverVersion ():String
Descripción	Retorna la versión del driver.
Nombre	UserName():String
Descripción	Retorna el usuario que está conectado a la base de datos.
Nombre	URL():String
Descripción	Retorna dirección de la base de datos.

Nombre	ProdVersion():String
Descripción	Retorna la versión del producto.
Nombre	MostrarTablas():DefaultTreeModel
Descripción	Retorna la lista de tablas de la base de datos y sus columnas.

Nombre: CC_GenerarDatos	
Tipo de clase: Controladora	
Atributo	**Tipo**
con	CC_Conexion
gu	Guardar
fi	File
Responsabilidad:	
Nombre	CC_GenerarDatos()
Descripción	Constructor sin parámetros.
Nombre	GenerarDatos(ArrayList<Tabla> tab, int cant, String parch):void
Descripción	Generar datos y guardarlos en un archivo.
Nombre	genDatestamp ():TimeStamp
Descripción	Generar un TimeStamp.
Nombre	genFecha ():Date
Descripción	Generar un Date.
Nombre	genEntero ():int
Descripción	Genera un entero.
Nombre	genEntero_rango (int from, int to):int
Descripción	Genera un entero dentro de un rango.
Nombre	leerFile():String
Descripción	Lee un archivo y retorna el contenido del mismo.
Nombre	genString(int lon):String
Descripción	Genera un string con una longitud lon.
Nombre	Archivar(Tabla tab, string parchivo):void
Descripción	Guarda los datos generados para una tabla en un archivo.
Nombre	ConocerString(ArrayList<Object> pstring, String pparam):boolean
Descripción	Determinar si param esta dentro de la lista de obejtos string.
Nombre	ConocerInt(ArrayList<Object> pint, String pint):boolean
Descripción	Determinar si param esta dentro de la lista de obejtos int.

3.5 Conclusiones

En este capítulo se desarrolló la propuesta de solución del sistema a partir del análisis y diseño del mismo. Se realizaron los diagramas de clases del análisis, diagramas de interacción y el diagrama de clases del diseño, que servirán de entrada para la siguiente fase y el flujo de trabajo de implementación, facilitando así la construcción del sistema en general.

Capítulo 4: Implementación y Prueba

4.1 Introducción

En el presente capítulo se traducen los resultados del diseño a lenguaje de programación Java, se implementa el sistema en términos de componentes, para dar respuesta a la petición del cliente por medio de la aplicación, a la cual se le realizan las pruebas necesarias para identificar y corregir los errores antes de que el software sea entregado al usuario final. Se especifican además los diagramas que forman parte del modelo de implementación, o sea, el diagrama de despliegue y el diagrama de componentes, así como los casos de prueba necesarios para identificar errores.

4.2 Implementación

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. El modelo de implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros.

4.2.1 Algoritmo de generación

Para generar los datos que se necesitan insertar se estableció un algoritmo de generación que consiste en la utilización de ficheros que contienen diccionarios de nombres, apellidos y cadenas de caracteres. Los datos que se generan fundamentalmente son String, Int y Date, para ellos se utilizan métodos de generación aleatoria basados en la clase Random de Java y los mismos son generados siempre teniendo en cuenta las restricciones presentes en la base de datos, evitando de esta forma que se rechacen algunos juegos de datos.

4.2.2 Algoritmo de inserción de datos.

Con el objetivo de lograr que en el proceso de inserción se respeten las restricciones de la base de datos se hizo necesaria la implementación de un algoritmo para este fin. El mismo devuelve el orden en que se deben ir llenando las tablas de la base de datos. Este resultado se obtiene utilizando la estructura de datos ArrayList <Tabla>, el resultado final es una lista ordenada de tablas según las dependencias que tienen entre ellas, a través del chequeo de las llaves foráneas de cada tabla. Este procedimiento se realiza partiendo de una lista que contiene todas las tablas de la base de datos.

```
public ArrayList<Tablas> ordenarTablas(ArrayList<Tablas> tablas)throws Exception
    {
      ArrayList<Tablas> resultado = new ArrayList<Tablas>();
        try
        {
            for (int i = 0; i < tablas.size(); i++) {
                ArrayList<Tablas> t = ordenarTabla(tablas.get(i));
                for (int j = 0; j < t.size(); j++) {
                  if(!resultado.contains(t.get(j)))
                  {
                      resultado.add(t.get(j));
                  }
                }
                if(!resultado.contains(tablas.get(i)))
                    resultado.add(tablas.get(i));
            }
        } catch (Exception e) {
              throw new Exception(e);
        }
        return resultado;
   }
```

Partiendo entonces de la lista ordenada que devuelve el método anterior, y con una cantidad de registros establecida por el usuario, haciendo uso del algoritmo de generación descrito anteriormente se generan los datos para las columnas de cada tabla de la lista y luego son insertados en la base de datos.

4.2.3 Diagrama de Componente

El diagrama de componente muestra los subsistemas de implementación organizados en capas y sus dependencias. Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten.

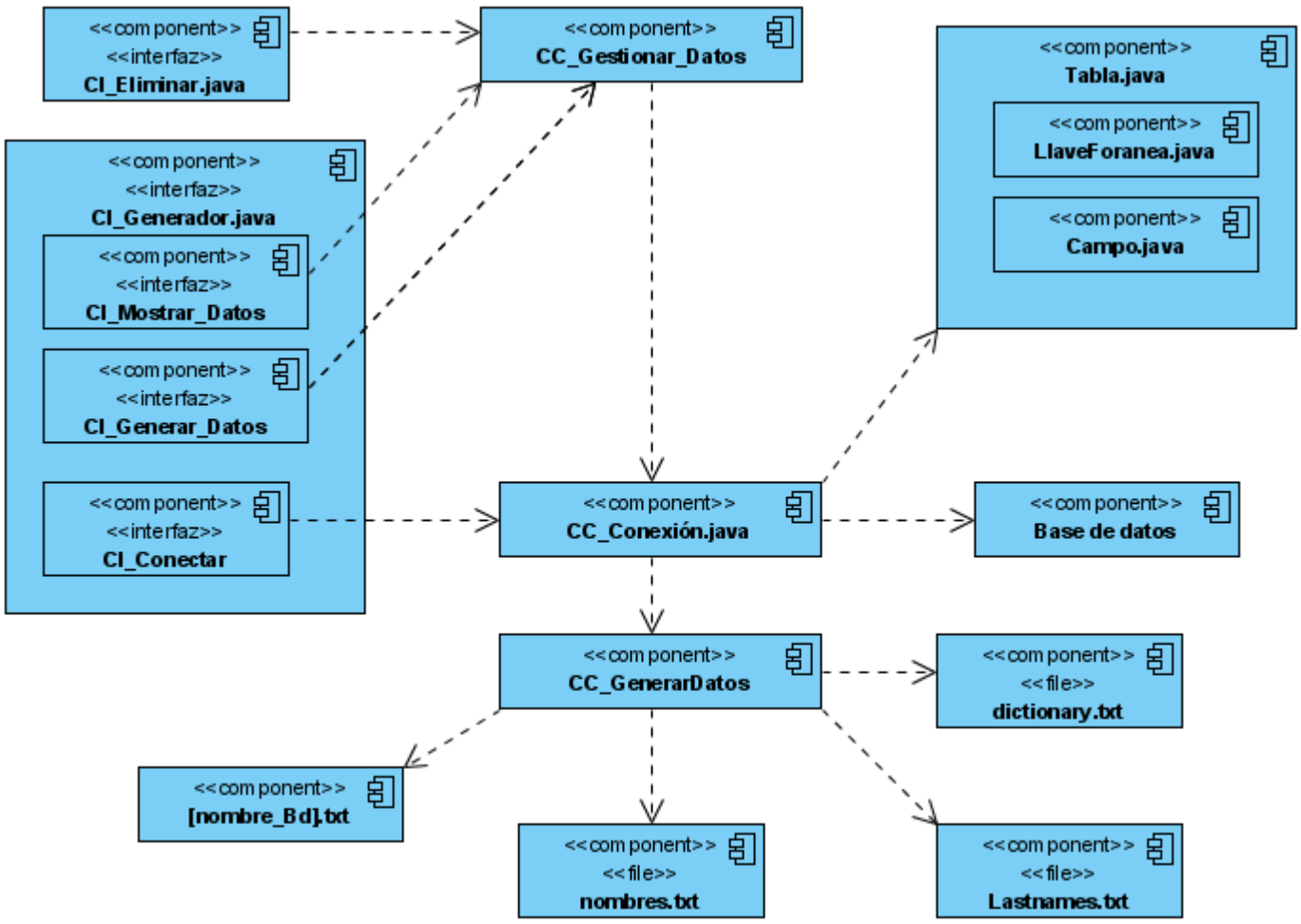

Fig. 18 Diagrama de componentes

4.2.4 Diagrama de Despliegue

El diagrama de despliegue representa la configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos. Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

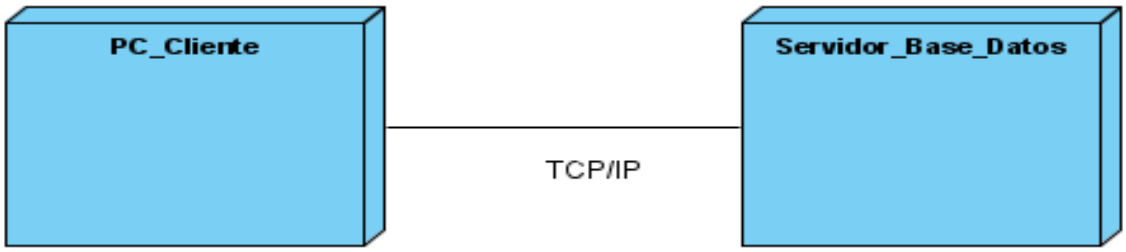

Fig. 19 Diagrama de despliegue

4.3 Prueba

El flujo de trabajo de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos específicos, los resultados de las mismas son observados y registrados, y se realiza una evaluación de algún aspecto del sistema o componente. Los objetivos de este flujo de trabajo son encontrar y documentar los defectos que puedan afectar la calidad del software, validar que el software trabaje como fue diseñado y validar y probar los requisitos que debe cumplir el software, además de validar que los requisitos fueron implementados correctamente.

Existen dos métodos de prueba, el método de prueba de Caja Blanca y de Caja Negra. Las pruebas de Caja Negra son aquellas que se le hacen a la interfaz del software, donde el objetivo es demostrar que la entrada y el resultado de la información son correctos mediante casos de prueba. En las pruebas de Caja Blanca se comprueban los caminos lógicos del software proponiendo casos de prueba. Estos modelos

permiten examinar el estado del programa en varios puntos y así poder verificar si su estado real coincide con el esperado.

A continuación se muestran los resultados de aplicar las pruebas de Caja Negra sobre la herramienta generadora de datos desarrollada en este trabajo, mediante el uso de casos de pruebas y la descripción de ciertos parámetros del caso de uso basándose en la plantilla correspondiente con este fin.(**Anexo** 2)

Caso de Uso	Gestionar conexión		
Caso de Prueba	**Entrada**	**Resultado esperado**	**Resultados**
Crear conexión	Los datos necesarios para realizar la conexión insertados por el usuario son correctos.	El sistema debe realizar la conexión a la base de datos de manera exitosa y mostrar un mensaje en señal de que así fue.	Se realiza la conexión a la base de datos de manera exitosa y la aplicación muestra un mensaje en señal de que así fue.
	Entre los datos necesarios para realizar la conexión insertados por el usuario hay alguno incorrecto.	El sistema no debe realizar la conexión a la base de datos y debe mostrar un mensaje indicando cual o cuales de los datos son incorrectos.	El sistema no realiza la conexión a la base de datos y muestra un mensaje indicando cual o cuales de los datos son incorrectos.
	El usuario olvida insertar uno o varios datos necesarios para realizar la conexión a la base de datos.	El sistema no debe realizar la conexión a la base de datos y debe mostrar un mensaje indicando que existen campos vacíos.	El sistema no realiza la conexión a la base de datos y muestra un mensaje indicando que existen campos vacíos.

4.4 Conclusiones

Con la implementación de la solución propuesta se da respuesta finalmente a la necesidad del cliente, obteniendo una aplicación capaz de satisfacer sus necesidades, y a la cual se le han realizado pruebas para verificar que cumple con los objetivos planteados de forma tal que el usuario quede satisfecho con el rendimiento de la misma.

Conclusiones

Con la culminación de este trabajo se ha logrado una herramienta capaz de generar juegos de datos de manera automática e insertar los datos generados en una base de datos, ahorrándole tiempo y trabajo a los integrantes del equipo de desarrollo del proyecto, pues de otra forma tendrían que poblar la base de datos de forma manual para poder realizar prueba sobre la misma, por lo que serían necesarias horas y días de trabajo, además el hecho de poblar todas las tablas de la base de datos le da la posibilidad a los desarrolladores de probar la aplicación y la base de datos acortando considerablemente el tiempo invertido en las pruebas a la base de datos, dándole cumplimiento al objetivo principal del trabajo.

Recomendaciones

Se recomienda:

- ✓ Continuar el desarrollo de la herramienta hasta lograr que sea independiente del sistema gestor de bases de datos.
- ✓ Extender los tipos de datos que se generan.
- ✓ Optimizar el algoritmo de generación e inserción de datos.

Referencias bibliográficas

1. Chays, David. Test Data Generation for Relational Database. 2005.
2. Pratap Gupta, Bhanu and Nemidas Vira, Devang. Automatic Generation of Test Databases for Database Applications. Mumbai, Bombai, India : s.n. p. 25.
3. SISTEMAS AVANZADOS DE RECUPERACIÓN DE INFORMACIÓN. iEspaña. [Online] [Cited: mayo 9, 2009.] http://sistemasavanzadosderecuperaciondeinformacion.iespana.es/.
4. Seara, Daniel. Recuperando metadata de la base de datos. El Guille. [Online] octubre 11, 2006. [Cited: mayo 9, 2009.] http://www.elguille.info/NET/ADONET/firmas_dani_Recuperando_metadata_base_datos.htm.
5. Vreeken, Jilles, van Leeuwen, Matthijs and Siebes, Arno. Preserving Privacy through Data Generation.
6. Miller, James, Reformat, Marek and Zhang, Howard. Automatic test data generation using genetic algorithm. Canadá : s.n., 2005.
7. DB Data Generator V2. Datanamic. [Online] [Cited: mayo 10, 2009.] http://www.datanamic.com/datagenerator/index.html.
8. forSQL Data Generator. forSQL. [Online] [Cited: mayo 10, 2009.] http://www.forsql.com/.
9. GenerateData.com. [Online] [Cited: mayo 9, 2009.] http://www.generatedata.com/#about.
10. DTM Data Generator. SQL Edit. [Online] [Cited: mayo 10, 2009.] http://www.sqledit.com/dg/.
11. Test data generator TDG 1.2c. Download 3k. [Online] febrero 2, 2006. [Cited: mayo 10, 2009.] http://www.download3k.com/Business-Finance/Database-Management/Download-Test-data-generator-TDG.html.
12. SQLManager.net. [Online] octubre 7, 2008. [Cited: mayo 9, 2009.] http://www.sqlmanager.net/en/products/oracle/datagenerator.
13. Advanced Data Generator. Upscene Production. [Online] [Cited: mayo 10, 2009.] http://www.upscene.com/products.adg.moreinfo.php.
14. Datatect 1.6. WareSeeker. [Online] mayo 19, 2007. [Cited: mayo 10, 2009.] http://wareseeker.com/Network-Internet/datatect-1.6.zip/27140

Bibliografía

1. **Pressman, Roger S.** *Ingenieria de Software. Un enfoque práctico.* Ciudad de La Habana : Editorial Félix Varela, 2005.
2. Oracle. *Oracle.* [Online] [Cited: mayo 10, 2009.] http://www.oracle.com/index.html.
3. Model-Code_Deploy Platform. *Visual paradigm.* [Online] [Cited: mayo 10, 2009.] http://www.visual-paradigm.com/product/vpuml/.
4. **Horton, Ivor.** *Ivor Horton's Beginning Java™ 2, JDK™ 5 Edition.* Indiana : Wiley Publishing , 2005. 0-7645-6874-4.
5. **Lea, Doug.** *PROGRAMACION CONCURRENTE EN JAVA. PRINCIPIOS DE DISEÑO Y PATRONES.* Madrid : PEARSON EDUCACION, 2001. 84-7829-038-9.
6. **López Gaona, Amparo.** *JDBC Básico.* Mexico : s.n., 2004.

Glosario de términos

- ✓ **Algoritmo de generación de datos**: secuencia de instrucciones para generar datos aleatorios.
- ✓ **Algoritmo de inserción de datos**: secuencia de instrucciones para insertar datos.
- ✓ **Algoritmo genético**: algoritmo basado en los principios de selección natural y genética.
- ✓ **Base de datos:** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.
- ✓ **Generador de datos**: herramienta para generar datos aleatorios automáticamente.
- ✓ **Metadatos**: información sobre la información.
- ✓ **Poblar base de datos**: llenar una base de datos.
- ✓ **Restricción**: condición que siempre debe cumplirse.
- ✓ T**ransacción**: colección de operaciones que realizan una única función lógica en una aplicación de base de datos.

ANEXOS

Anexo 1. Plantilla para la Descripción de un Caso de Uso.

Caso de Uso	<<nombre>>
Actor	<<nombre de los actores>>
Propósito	<<Breve descripción del objetivo del proceso>>
Resumen	<<Descripción del proceso completo indicando quién inicia y cómo se inicia, quién finaliza el proceso y cómo se hace>>
Precondiciones	<< Cosas que tienen que cumplirse en el sistema para que se ejecute el CU>>
Poscondiciones	<<Condiciones en las que queda el sistema cuando termina la ejecución del CU>>
Referencia	<< Listado de requerimientos y casos de uso asociados, indicando tipo de asociación (include o extend) >>
Casos de Uso Relacionados	<<Listado de casos de uso incluidos y extendidos de este caso de uso base, indicand006F el tipo de relación>>
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
Se indica al actor (o actores) y la interacción que tiene con el sistema	
Flujos Alternos	
	Comportamiento que no está en el flujo normal y que ocurre bajo ciertas condiciones que pueden darse en el flujo normal.
Prioridad	Indicar cuál es la prioridad de este proceso dentro del negocio que se modela.

Anexo 2. Plantilla para la Descripción de un Caso de Prueba.

Caso de Uso	<nombre del caso de uso>		
Caso de Prueba	**Entrada**	**Resultados esperados**	**Resultados**
<Nombre del Caso de Prueba>	<Descripción textual de lo que ocurre en el mundo real que hace necesario ejecutar el caso de prueba, precisando la data de entrada y los comandos a dar por el actor. Descripción textual del estado de la información almacenada>	<Descripción textual del la respuesta que se espera de el sistema>	<Descripción textual del estado en el que queda la información y las alertas que puedan generarse, una vez ejecutado el caso de uso con los valores y el estado especificado en la entrada>