Universidad de las Ciencias Informáticas

FACULTAD 4

Título: Generador de datos

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autora: Diormis de la Caridad Hernández Sarmiento

Tutor: Lic. Hayron Ruiz Corrales

Co-tutora: Ing. Lissett Díaz Mesa

Ciudad de La Habana, Julio 2008

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 firmo la presente a los ____ días del mes de _____ del año _____.

Diormis de la Caridad Hernández Sarmiento

_____

Firma del Autor

Ing. Lissett Díaz Mesa

_____

Firma del Co-Tutor

DATOS DE CONTACTOD

Lic. Hayron Ruiz Corrales (Tutor), graduado en Licenciatura de Ciencias de la Computación en la Universidad de La Habana en el año 2003, actualmente profesor instructor de la Universidad de las Ciencias Informáticas donde se desempeña como Administrador de Base de Datos en el proyecto de Prisiones.

Ing. Lissett Díaz Mesa (Co-Tutora), graduada en Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2007, en la cual se desempeña actualmente como jefa de colectivo de la asignatura Ingeniería de Software y analista principal del proyecto Banco Nacional.

AGRADECIMIENTOS

Les agradezco a:

Mis padres por traerme al mundo, ayudarme y mostrarme confianza; a mis hermanos Diorgis y Diorkis por el apoyo brindado, a mi familia en general por mostrarme el camino a seguir. Al compañero Fidel Castro por permitirme estudiar en esta universidad.

Mis hermanitas Yadira Calimano, Virtudes Figueredo y a mi novio Daynier Moreno por ayudarme, tener la paciencia suficiente y aguantar todos mis cambios de humor.

Los profesores Hayron Ruiz Corrales (Tutor), Lissett Díaz Mesa (Co-Tutora) y Yulierky Torres por dedicarme tiempo y apoyarme en todo momento, a Julio Cesar Díaz por ayudarme cuando lo necesité.

Mis compañeros Rainier Pérez, Yilenis Rosario Grillo, Jose Antonio Sánchez, y Jose Alejandro Lugo por aclararme dudas y servir de apoyo a lo largo de mi carrera.

A todos aquellos que me dieron su ayuda incondicional aportándome conocimientos básicos para que de una forma u otra, este trabajo de diploma se realizara con la calidad requerida.

DEDICATORIA

A mi abuela Juana Iris García, a mis padres, hermanos Diorgis y Diorkis; a mis verdaderos amigos y a mi novio, por ser las personas más especiales de mi vida.

RESUMEN

Este trabajo se lleva a cabo debido a que los desarrolladores emplean mucho tiempo introduciendo los datos de las base de datos para hacerles pruebas a los software. Por lo que la investigación tiene como objetivo desarrollar una herramienta que permita generar datos aleatorios a las base de datos, permitiendo que los programadores de la Universidad de las Ciencias Informáticas puedan utilizarlo para poblar las mismas y realizarle pruebas a sus software en el menor tiempo posible.

Primeramente se hace un estudio de los principales Gestores de Base de Datos, se utilizan herramientas, metodologías, tipos de arquitectura, lenguajes, todo lo necesario para realizar con mayor calidad la aplicación Web.

Esta investigación contribuye al desarrollo de la Informática tanto en la Universidad de las Ciencias Informáticas como en nuestro país y su solución es una base de lo que pueda ser el comienzo del desarrollo de los generadores para las bases de datos que utilicen cualquier tipo de gestor.

INDICE

INTRODUCCIÓN

A medida que el hombre ha evolucionado con él se han desarrollado los diferentes campos de la ciencia y la tecnología. Debido a esto cada día son más lo avances en la rama de la electrónica, las comunicaciones y la informática, por lo que la sociedad se traza nuevas metas y soluciones buscadas a diferentes tipos de problemas.

En conjunto a este desarrollo, los clientes exigen un trabajo más fácil, de rápido uso y con la misma o mejor calidad, por lo que en la mayoría de las ocasiones esto se transforma en una gran interrogante para los programadores, ya que tienen la obligación de cumplir con las necesidades del cliente, para esto es necesario hacerle pruebas a los software que realizan de manera rápida; pero segura.

La mayoría de los sistemas informáticos utilizan gestores de bases de datos relacionales, por esta vía, se guarda la información necesaria para el correcto funcionamiento de la aplicación de forma permanente y por ende los desarrolladores se ven obligados a realizar la entrada de los datos a estas bases de datos de forma manual.

Esto conlleva a que el tiempo de prueba del software se demore aún más, por lo que en la mayoría de las ocasiones, no se realizan las pruebas necesarias o con grandes cantidades de datos, dificultando de esta manera el éxito de las aplicaciones.

Debido a que los desarrolladores emplean mucho tiempo a la hora de llenar las bases de datos para ejecutarle pruebas a los software, se ha determinado realizar una herramienta que genere datos de manera automática, evitando la demora en el momento de poblar las bases de datos, permitiendo realizar las pruebas a sus aplicaciones con grandes cantidades de datos y facilitarle tiempo a los programadores, impidiendo que la entrada de estos datos se convierta en un trabajo más para ellos.

La UCI[1] a pesar de tener como principal objetivo ser la base de la automatización de la nación, informatizando los principales sectores de la sociedad cubana y de otros países, no está ajena a dificultades y contratiempos; por lo que, para cumplir con todos los compromisos

[1]UCI: Universidad de las Ciencias Informáticas

trazados por los diferentes proyectos productivos sin afectar la calidad de los mismos, se hace necesario contar con una herramienta que le permita poblar sus bases de datos de manera eficaz.

Este se convierte en el ***problema a resolver*** ¿Qué herramienta de población automática debe utilizarse en los proyectos productivos de la universidad de las ciencias informáticas?

Debido a las diferentes investigaciones que se han realizado sobre algunas de las aplicaciones presentes en el mercado internacional, se llegan a conocer las variedades de herramientas propietarias, puesto que Cuba no posee una economía que sustente a la compra de las claves de registro, estas no pueden ser utilizadas por las universidades, además de que las mismas no eran capaces de integrar en un único modelo los principales gestores que se utilizan en la UCI; MySQL[2] PostgreSQL[3], Oracle, SQL[4], entre otros.

Para ello se definen como ***objeto de estudio*** los algoritmos de población automática de base de datos y los modelos de almacenamiento de meta-información en los gestores relacionales. Previéndose accionar directamente sobre el proceso de población de las bases de datos en los proyectos productivos de la universidad de las ciencias informáticas.

Se define como ***campo de acción*** el proceso de población de las bases de datos para realizar pruebas a las aplicaciones.

Se tiene como ***objetivo general*** el desarrollar una herramienta de población automática para almacenar los datos en las tablas de la base de datos.

Como ***posible resultado*** está el de llegar a obtener una aplicación con interfaz Web capaz de poblar una base de datos de manera automática.

Para realizar lo anteriormente expuesto se deben de realizar diferentes ***tareas de investigación***:

[2] MySQL**:** Servidor de base de datos relacional orientado a objeto
[3] PostgreSQL**:** Servidor de base de datos relacional orientado a objeto
[4] SQL**:** El Lenguaje de consulta estructurado (en inglés Structured Query Language)

- Detallar el mecanismo presente en cada uno de los gestores para almacenar la meta información.
- Estudio del estado del arte de los sistemas automáticos de generación de datos.
- Realizar la Ingeniería de Software correspondiente que de respuesta al problema científico.
- Investigar sobre el lenguaje que se va a utilizar para la aplicación
- Modelar e implementar un algoritmo de generación de datos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En la UCI, que es la universidad actualmente encargada de informatizar al país, no existe una herramienta que conlleve a la población automática de las bases de datos que utilizan la mayoría de los software, por lo que es necesario realizar una aplicación que logre generar los datos para que los programadores no tengan que realizar esta función de forma manual.

En el presente capítulo se hará un estudio sobre los generadores de datos existentes y los gestores de base de datos más utilizados, así como los conceptos más importantes respecto al tema. Se mencionará la metodología, herramientas, arquitectura y lenguaje utilizados como soporte al proceso de desarrollo de software.

1.2. Conceptos básicos

Antes de entrar en detalle primeramente se debe comprender por qué es necesario e importante un generador de datos, para así poder entender con mayor facilidad el presente trabajo: el **generador de datos de una base de datos** surge de la necesidad de llenar los campos de forma rápida de una base de datos que son manipuladas por algunos de los gestores de datos, son de gran importancia ya que simplifican el trabajo de los programadores a la hora de llenar las bases de datos y por ende disminuyen el tiempo de prueba del software; una **base de datos** no es más que una colección de datos y/o documentos digitales que pueden ser homogéneos o no, que disponen de sistemas de gestión de bases de datos (relacionales o documentales) y un conjunto de aplicaciones que hacen posible su publicación, integración y consulta dentro o fuera de Internet (1), o lo que es lo mismo, conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora (2).

Para poder usar o manipular los datos guardados en estas bases de datos, existe el software comúnmente llamado **sistema de gestión de bases de datos (SGBD[5]),** que no es más que la herramienta que permite la utilización o actualización de los datos almacenados en una o varias base de datos, por uno o varios usuarios, desde diferentes puntos de vista y a la vez (3). Los SGBD tienen como objetivo fundamental suministrar al usuario las herramientas necesarias para manipular los datos, también podemos destacar como principal función para facilitar el diseño de aplicaciones y que los tratamientos sean más eficientes y rápidos a la independencia de los datos y los programas de aplicación, la minimización de la redundancia, integración y sincronización de las bases de datos, integridad de los datos, seguridad y recuperación, facilidad de manipulación de la información y el control centralizado.

Dentro de las ventajas de estos sistemas se pueden mencionar las siguientes:

- Facilidad de manejo de grandes volúmenes de información.
- Gran velocidad en muy poco tiempo.
- Independencia del tratamiento de información.
- Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
- No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
- Integridad referencial.

Existen varios sistemas de gestión de base datos, unos libres, otros gratuitos y muchos de forma comercial, a continuación se muestran ejemplos de algunos de ellos.

1.3. Sistemas de gestores de base de datos (SGBD)

1.3.1. Gestores de base de datos libres

1.3.1.1. PostgreSQL

[5] SGBD: Sistema de gestión de base de datos

PostgreSQL es un gestor de base de datos de código abierto orientado a objeto, es uno de los más reconocidos y avanzados en el mundo. Sus tablas son objetos y las tuplas[6] instancias de ese objeto. Posee diversas ventajas y características las cuales se podrían mencionar algunas:

- **DBMS [7]Objeto Relacional**: aproxima los datos a un modelo objeto-relacional, siendo capaz de manejar complejas rutinas y reglas. Su funcionalidad es tan desarrollada que permite realizar consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.
- **Altamente Extensible**: Soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- **Soporte SQL Comprensivo**: soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.
- **Integridad Referencial (llaves foráneas)**: es utilizada para garantizar la validez de los datos de la base de datos.
- **[8]API Flexible**: tal flexibilidad ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS[9] PostgreSQL. Estas interfaces incluyen Objeto Pascal, Python, Perl, PHP[10], ODBC[11], Java/JDBC, Ruby, TCL, C/C++, y Pike.
- **MVCC**[12] o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios manteniendo una ruta a todas las transacciones realizadas por los usuarios siendo capaz de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.
- **Cliente/Servidor**

Ventajas

- Instalación ilimitada.

[6] Tuplas: Grupo de ocurrencias de campos relacionados
[7] DBMS: Sistemas de Gestión de Base de Datos (Database management system)
[8] API: Interfaz de Programación de Aplicaciones (Application Programming Interface)
[9] RDBMS: Sistema de gestión de base de datos relacional (Relational Data Base Management System)
[10] PHP: Hypertext Pre-processor
[11] ODBC: Open Database Connectivity
[12] MVCC: Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control)

- Mejor soporte que los proveedores comerciales.
- Ahorros considerables en costos de operación.
- Estabilidad y confiabilidad legendarias.
- Extensible.
- Multiplataforma.
- Diseñado para ambientes de alto volumen.

En caso del diseñado para ambientes de alto volumen, existen varias herramientas (Tora, Data Architect) y administración de base de datos (pgAdmin, pgAccess).

1.3.1.2. MySQL

MySQL es un sistema de administración relacional de bases de datos que almacena información en tablas separadas. Es software libre, cualquier persona puede utilizar y modificar el código ajustándolo a sus necesidades sin pagar por hacerlo. Ofrece gran velocidad y flexibilidad por lo que posee un mayor rendimiento. Es muy rápido y consume pocos recursos de CPU y de memoria. Su conectividad y robustez hacen de él un buen sistema de gestor de bases de datos. Posee diversas características, algunas de ellas son:

- Interioridades y portabilidad: funciona en diferentes plataformas y para su portabilidad usa GNU[13] Automake, Autoconf, y Libtool.
- Proporciona sistemas de almacenamientos transaccionales y no transaccionales.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.
- Es modulador, es decir se puede elegir el tipo de una tabla al momento mismo de crearla. Si algunas de las tablas necesitan transacciones, se puede elegir el tipo de

[13] GNU: GNU no es Unix (acrónomo recursivo)

tabla que mejor se acomode a las necesidades, por lo que no se necesita tener la sobrecarga de las transacciones en todas las tablas.

- Integridad referencial: se pueden definir llaves foráneas entre tablas para asegurarse de que un registro no puede ser eliminado de una tabla si aún está siendo referenciado por otra tabla.
- Tiene soporte nativo de SSL[14] por lo que no será necesario configurar un canal seguro con SSH[15], o usar *stunnel*[16] para establecer una conexión encriptada entre un servidor y un cliente MySQL.

Entre sus ventajas podemos destacar las siguientes:

- Mayor rendimiento. Mayor velocidad al conectar con el servidor
- Mejores utilidades de administración (backup, recuperación de errores, etc.).
- Mejor integración con PHP.
- No hay límites en el tamaño de los registros.
- Mejor control de acceso, en el sentido de qué usuarios tienen acceso a qué tablas y con qué permisos.

Existen muchos otros como el Firebird basada en la versión 6 del Interbase, SQLite la cual su librería se enlaza con el programa y así pasa a hacer parte del mismo, esta librería implementa parte del estándar SQL-92, DB2 Express-C, esta es la versión gratuita del DB2 permitiendo desarrollar, implementar y distribuir aplicaciones que no usen las características avanzadas de las versiones del DB2 y el Apache Derby, el cual es una base de datos basada en Java y trabaja sobre aplicaciones de pequeños sitios web.

1.3.2. Gestores de base de datos gratuitos

1.3.1.3. *SQL Server Compact Edition*

SQL Server posee soluciones para recuperación de desastres y tiene mayor disponibilidad del sistema a usuarios de la base de datos a través de tecnologías de alta disponibilidad.

[14] SSL: Secure Sockets Layer
[15] SSH: Secure Shell
[16] Stunnel: Es un programa de computadora libre y multi-plataforma

Ofrece funcionalidades de Inteligencia de Negocios con altos estándares, permitiendo que pueda obtener mayor ventaja de los datos almacenados en la organización. Se puede utilizar para construir, administrar e implementar aplicaciones de negocios.

El tamaño de una base de datos que trabaje sobre este gestor está totalmente limitada a Megas. El grupo de sentencias SQL que soporta están limitadas a las restricciones de consulta más básicos. Utiliza un fichero para el almacenamiento de datos, como también para guardar la estructura interna de las base de datos; estos tienen extensión .sdf. Constituye la mejor base de datos para Internet y Extranet. Posee grandes mejoras en programabilidad y lenguaje. Posee una amplia seguridad.

Existen otros gestores de base de datos como el **Sybase ASE** [17]el cual es de alto rendimiento, soporta grandes cantidades de datos así como transacciones y usuarios, además permite almacenar datos de forma segura, tener acceso y procesar datos de manera inteligente y movilizar estos datos.

1.3.3. Gestores de base de datos comerciales

Dentro de los gestores de base de datos comerciales están el Advantage Database, dBase, FileMarker, Fox Pro, Interbase, Magic, Open Access, SQL Server, Pardox y muchos otros.

1.3.1.4. Oracle

Oracle es un sistema de gestión de base de datos relacional (o RDBMS). Se destaca por su estabilidad, escalabilidad, por ser un sistema multiplataforma y el soporte de transacciones que posee. Utiliza los recursos informáticos en toda la arquitectura del hardware y software, sin tener que cambiar alguna línea de código. Por ser uno de lo más completos es el que mayormente se usa. Puede correr en computadoras personales, microcomputadoras, mainframes y computadoras con procesamiento paralelo. Almacena: los códigos de los programas para empezar a ejecutarse, los datos necesarios durante la ejecución y la información sobre como es la transferencia entre procesos y periféricos.

[17] ASE: Adaptive Server Enterprise

Oracle está compuesto por su Kernel, las instancias del sistema de base de datos y sus archivos relacionados a este sistema.

Como los Sistemas de gestores de base de datos, en el mundo también existen diferentes tipos de generadores de datos, mayormente aparecen en el mercado de manera comercial. En este trabajo se nombran algunos de estos generadores.

1.4. Tipos de generadores de datos

En el mundo existen varios tipos de generadores de datos propietarios, a continuación veremos ejemplos de estos:

1.4.1. Generadores Propietarios

- **Data Generator para MySQL 2.3** es un utilitario para generar datos de prueba a tablas de bases de datos MySQL. La aplicación asistente te permite definir tablas para generar datos, configurar valores de rangos, generar campos de tipo carácter y de objetos binarios, y muchas otras características más para generar datos de prueba de manera simple y directa. También te provee una aplicación consola, que te permite generar datos de un "clic" usando plantillas de generación (4).

- **Data Generator para SQL Server 2.2** es un utilitario para generar datos de prueba a tablas de bases de datos Microsoft SQL (5). La aplicación asistente te permite definir tablas para generar datos, configurar valores de rangos, generar campos y objetos binarios, y muchas características más para generar datos de prueba de manera simple y directa. También te provee una aplicación consola, que te permite generar datos de un "clic" usando plantillas de generación.

- **Data Generator 2005 para Oracle 2.3** es una utilidad impresionante para generar datos de prueba a varias tablas de base de datos de Oracle inmediatamente. La aplicación de mago permite definir tablas y campos para generar datos, poner variedades de valor, generar campos de trabajo, conseguir listas de valores de preguntas de SQL y muchos otros rasgos para generar datos de prueba de un modo simple y directo (6).

- **forSQL Data Generator 1.0** permite la generación automática de datos de prueba para base de datos de escala grandes y Garantía de Calidad. Los reveladores de base de datos y los probadores se beneficiarán de la oportunidad de llenar una base de datos de juegos de datos correctos y realistas.

- **Data Generator 2005 para PostgreSQL 2.3** es un poderoso utilitario para generar datos de prueba a tablas de bases de datos PostgreSQL de una vez. La aplicación asistente te permite definir tablas para generar datos, configurar valores de rangos, generar campos de tipo carácter y de objetos binarios, y muchas otras características más para generar datos de prueba de manera simple y directa. También te provee una aplicación consola, que te permite generar datos de un "clic" usando plantillas de generación (7).

- **Data Generator para DB2** es una poderosa utilería para generar inmediatamente datos de prueba para varias tablas de base de datos DB2. La aplicación asistente te permite definir tablas para generar datos, configurar valores de rangos, generar campos de tipo carácter y de objetos binarios, y muchas características más para generar datos de prueba de manera simple y directa (8). También te provee una aplicación consola, que te permite generar datos de un "clic" usando plantillas de generación.

1.5. Tecnología utilizada

Para realizar la aplicación web se ha utilizado internet, por ser donde más información de todo el mundo se va a encontrar y donde esta información está mucho más actualizada. Internet posee diversas características las cuales son mencionadas a continuación.

1.5.1. Internet

Internet es la red de redes, conjunto formado por millones de computadoras diferentes que se comunican entre sí mediante protocolos informáticos (9). Posee gran avance en el desarrollo de la Informática y las comunicaciones donde se puede obtener información valiosa sobre diversos temas, como también constituye un importante medio de comunicación por lo que entre sus características se destaca la total interactividad y su

formidable capacidad de transmisión, por lo que cualquier usuario puede acceder a su información.

Surgió en los Estados Unidos buscando la comunicación entre investigadores y militares. Aunque los países desarrollados se benefician más de esta herramienta, en los países subdesarrollados y bloqueados también existen muchos usuarios que visitan internet. Posee diversas herramientas como por ejemplo: el correo electrónico que permite el intercambio de mensajes que pueden incluir como anexo archivos, fórum donde encontraremos intercambio de opiniones de diversos temas, el chat, fotolog, páginas personales, descarga de programas, WWW el cual es el servicio con mayor auge y el sistema de navegación de documentos basado en una interfaz de usuario gráfica y amigable, FTP que permite la transferencia de archivos entre computadoras conectadas a la red.

Mediante esta tecnología se pueden realizar cualquier tipo de transacciones como también permite realizar aplicaciones desde el hogar o la empresa permitiendo que cada persona realice sus funciones diarias, además de esto permite la realización de videoconferencias y por ende el acceso a imágenes.

1.6. Estilos arquitectónicos

Existen diversos estilos arquitectónicos entre los que se destacan: la arquitectura en dos capas; arquitectura en tres capas la cual es la que se va a utilizar puesto que es mucho más avanzada que la arquitectura en dos capas, sus capas están mejores estructuradas, me permite trabajar con el patrón Modelo Vista Controlador y lo más importante, separa la lógica del negocio; y la arquitectura en n capas. A continuación se verán algunas de las principales características de estas arquitecturas.

1.6.1. Arquitectura en capas

La arquitectura en capas no es más que la vista conceptual de una aplicación donde se muestran bloques de construcción físico y lógico que describe las interacciones entre los bloques de construcción. Tiene como objetivo minimizar los efectos de cambios en una capa y aislar la lógica del negocio, además sirve de gran ayuda a la hora de identificar qué

puede reutilizarse y proporciona una estructura que nos ayuda a tomar decisiones sobre qué partes comprar y qué partes construir. Hoy en día existen tres tipos de arquitecturas de capas:

- Arquitectura de dos capas
- Arquitectura de tres capas
- Arquitectura de n capas

1.6.1.1. Arquitectura en dos capas

La arquitectura de dos capas fueron las primeras en utilizar la estructura cliente-servidor. Presenta dos capas o niveles: el Nivel de Aplicación que es donde se dispone de toda la interface y donde el usuario puede realizar las actividades y el Nivel de Base de Datos o Repositorio de Datos la cual es útil pues es utilizada para guardar toda la información del sistema. Algunas de las herramientas que se pueden utilizar son: Visual Basic, Access y SQL

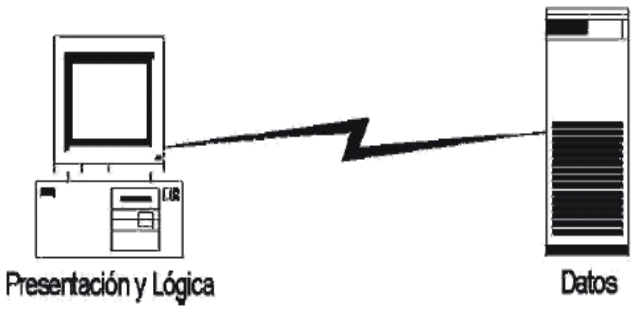

Fig. 2. Arquitectura en dos capas.

1.6.1.2. Arquitectura en tres capas

Esta arquitectura está representada por tres niveles: el Nivel de Aplicación, que a diferencia de la de la arquitectura de dos capas, esta solo trabaja con la arquitectura semántica de la aplicación sin importar su implementación ni su estructura física; el Nivel de Dominio de la Aplicación, la cual es la encargada de la estructura física y el dominio de

la aplicación, una de las ventajas de esta aplicación es que los cambios se realizan únicamente en el servidor; y el Nivel de Repositorio donde se almacenan todos los datos.

Esta capa utiliza herramientas como Visual Basic y Visual Studio .Net para la capa de aplicación, SQL Server y Oracle para el repositorio y MTS para el nivel de dominio de la aplicación.

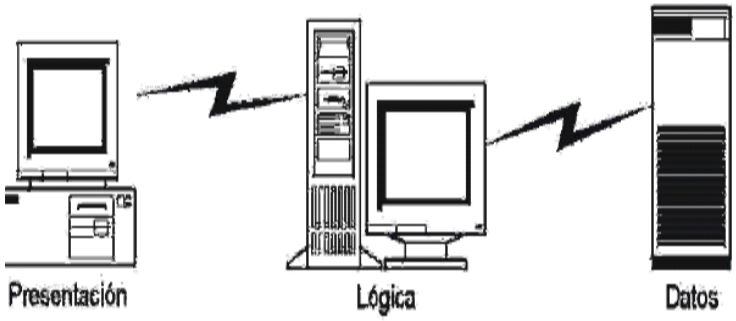

Fig. 3. Arquitectura en tres capas.

Capa de presentación: la capa de presentación está formada por la lógica de aplicación y los componentes de la interface, la misma prepara datos para su envío a la capa de cliente y procesa solicitudes desde la capa de cliente para su envío a la lógica de negocios del servidor. También puede incluir un servicio de portal que proporcione acceso personalizado y seguro a los servicios de negocios en la capa de servicios de negocio.

Capa de servicios de negocios: la capa de servicios de negocio consiste en la lógica que realiza las funciones principales de la aplicación: procesamiento de datos, implementación de funciones de negocios, coordinación de varios usuarios y administración de recursos externos como, por ejemplo, bases de datos o sistemas heredados. Se construyen desde entidades, componentes, agentes e interfaces con otras aplicaciones. Los servicios de negocios también se pueden crear como servidores independientes como por ejemplo, un servidor de mensajería o un servidor de calendario empresarial. El punto de entrada de esta capa es la interface.

Capa de datos: la capa de datos está formada por los servicios que proporcionan los datos persistentes utilizados por la lógica de negocios. Los datos pueden ser de aplicaciones almacenados en un sistema de administración de bases de datos o pueden incluir información de recursos y directorios almacenada en un almacén de datos de protocolo ligero de acceso a directorios (en inglés: Lightweight Directory Access Protocol; LDAP[18]). Los servicios también pueden incluir alimentación de datos de orígenes externos o datos a los que se puede obtener acceso desde sistemas informáticos heredados.

1.6.1.3. Arquitectura en n capas

La arquitectura de n capas posee cuatro niveles: el Nivel de Presentación, Nivel de Aplicación, Nivel de Dominio de la Aplicación y Nivel de Repositorio.

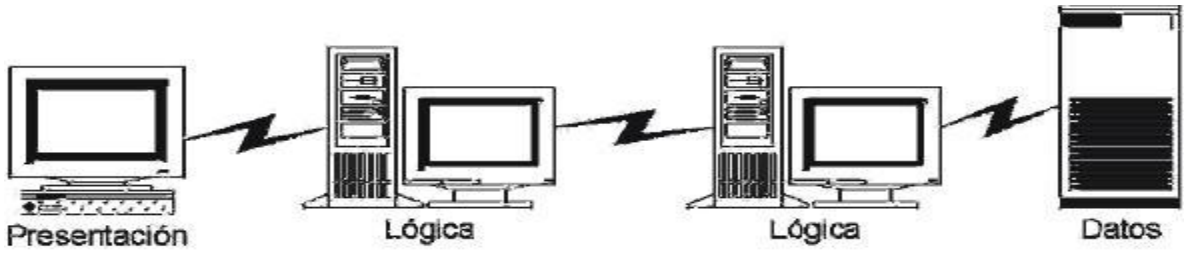

Fig. 4. Arquitectura en n capas.

1.7. Patrones

Un patrón no es más que un modelo a seguir, describe un problema que ocurre más de una vez, se utiliza en la mayoría de las ocasiones en campos como la arquitectura y en el desarrollo de software orientado a objeto, su objetivo es crear un lenguaje común donde se pueda expresar las experiencias y soluciones de problemas, un patrón debe cumplir con varios requisitos fundamentales tales como:

- Solucionar un problema.
- Ser un concepto probado.
- La solución no es obvia.
- Describe participantes y relaciones entre ellos.

[18] LDAP: Protocolo ligero de acceso a datos (Lightweight Directory Access Protocol)

- Tiene un componente humano alto: estética y utilidad.

Como principal objetivos los patrones:

- Ayudan a construir la experiencia colectiva de Ingeniería de Software.
- Son una abstracción de "problema – solución".
- Se ocupan de problemas recurrentes.
- Identifican y especifican abstracciones de niveles más altos que componentes o clases individuales.
- Proporcionan vocabulario y entendimiento común.

Existen diversos tipos de patrones de los cuales podemos destacar:

1. Patrones de Procesos
   - Procesos y Organizacionales
   - Administración
   - Diseño
   - Análisis
   - Prueba.
2. Patrones de Productos de Software.
   - Análisis
   - Arquitectura
   - Diseño
   - Idioma

De todos estos tipos de patrones en este proyecto se utilizarán los patrones de arquitectura, los cuales describen un problema en específico y presentan un esquema genérico y probado de su solución; dentro de los arquitectónicos se utiliza el patrón Modelo Vista Controlador.

1.7.1. Modelo Vista Controlador (MVC[19])

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Maneja la visualización de la información.

Controlador: Controla el flujo entre la vista y el modelo (los datos).

Entre las ventajas que posee este tipo de patrón están:

Soporte de múltiples vistas: Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes.

Adaptación al cambio: Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

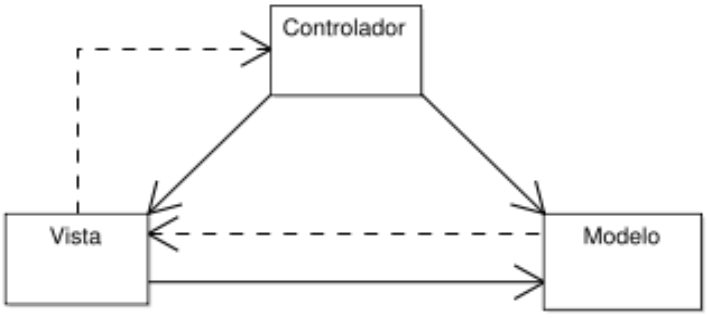

Fig. 5. Modelo Vista Controlador.

[19] MVC**:** Patrón Modelo Vista Controlador

Para el logro exitoso de esta aplicación web ha sido imprescindible utilizar metodologías y herramientas de las que ya existen, por lo que es necesario tener un mínimo conocimiento de lo que significa la palabra “metodología”, a continuación se detalla su concepto, importancia y algunas de ellas.

1.8. Metodologías y herramientas

La metodología es un proceso que sirve de ayuda a los integrantes de un proyecto a realizar con mejor calidad el software a automatizar, es decir, conjunto de métodos empleados para el desarrollo de sistemas automatizados, es muy importante pues así se podrá llegar al objetivo real del cliente.

Para utilizar alguna metodología, ya sea echa por uno mismo o utilizando ya una realizada, se deben cumplir varias funcionalidades; primeramente la tecnología escogida debe: ajustarse a los objetivos, cubrir el ciclo entero de desarrollo del software e integrando las distintas fases de este ciclo, incluir la realización de validaciones, soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo, ser la base de una comunicación efectiva, funcionar en un entorno dinámico orientado al usuario, especificar claramente los responsables de resultados, poder emplearse en un entorno amplio de proyectos software, poder enseñar, estar soportada por herramientas CASE[20], soportar la eventual evolución del sistema y contener actividades conducentes a mejorar el proceso de desarrollo de software.

Cada metodología proporciona una guía para estimar costos, manejo del proyecto en las tareas y entregas, medidas y métricas, formas definidas y dirección en las entregas de la construcción, políticas y procedimientos para garantizar la calidad del software, descripciones de los roles y programas de entrenamiento detallados, ejemplos totalmente trabajados, ejercicios de entrenamiento, técnicas para adaptar el método y las técnicas definidas.

[20] CASE: Ingeniería de Software Asistida por ordenador (*Computer Aided Software Engineering*)

Entre muchas metodologías que existen en el mundo podemos encontrar el Rational Unified Process (RUP[21]); OPEN, MÉTRICA 3; Extreme Programming (XP[22]); Microsoft Solutions Framework[23] (MSF[24]); FDD[25]; entre otras.

Al igual que las metodologías, existen diferentes herramientas CASE, que no son más que, aplicaciones Informáticas para mejorar y aumentar el desarrollo del software a menor costo en tiempo y dinero. Estas herramientas nos ayudan durante todo el ciclo de vida del software, por ejemplo, en el diseño de un proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Cualquier tipo de herramienta que se utilice debe tener como objetivo el mejoramiento de: la productividad en el desarrollo y mantenimiento del software; el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos; la planificación de un proyecto, aumentar la: calidad del mismo; la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos, automatizar el desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto; ayudar a la reutilización del software, portabilidad y estandarización de la documentación.

Entre las herramientas que existen en el mundo se encuentran: ArgoUML, BPWin, CASE Studio 2, EasyCase, Embarcadero ER/Studio, Enterprise Architect, GeneXus, INNOVATOR, Rational Rose, SILVERRUN, System Architect, Visual Paradigm for UML[26], Xcase Database Design Software, entre muchos otros. A continuación se da a conocer las características de algunas metodologías y herramientas.

[21] RUP: Proceso Unificado del Racional (Rational Unified Process).
[22] XP: Programación extrema (Extreme Programming)
[23] Framework: Representa una arquitectura del software
[24] MSF: Microsoft Solution Framework
[25] FDD: Desarrollo guiado por la funcionalidad (Feature driver development) 002E
[26] UML: Lenguaje unificado de modelado (Unified Modeling Language)

1.8.1. Rational Unified Process (RUP)

La metodología RUP está dividida en cuatro fases: **Inicio** con el principal objetivo de determinar la visión del proyecto, **Elaboración** donde se determina la arquitectura óptima, **Construcción** aquí se obtiene la capacidad operacional inicial y **Transición** que no es más que obtener una versión 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, Requerimiento, Análisis y Diseño, Implementación, Prueba, Instalación, Administración de configuración y cambio, Administración del Proyecto y Ambiente.

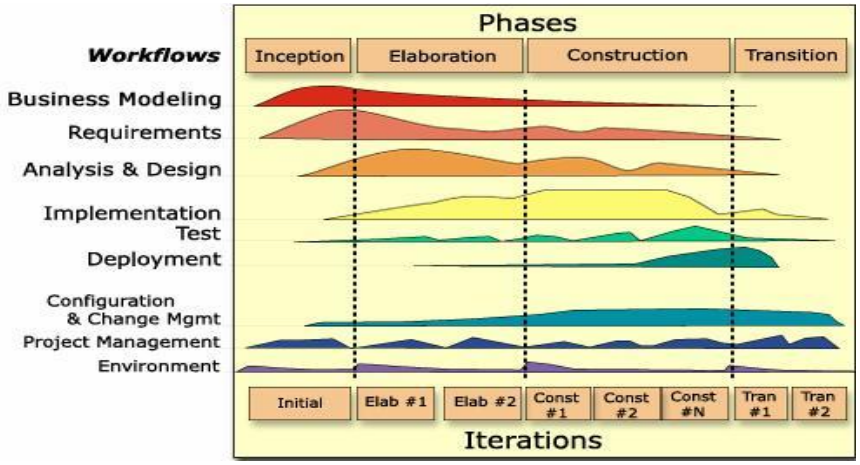

Fig. 6. Rational Unified Process.

El ciclo de vida de RUP está caracterizado por ser: **Dirigido por casos de uso** ya que los casos de uso recogen y representan lo que el usuario final realmente desea, **Centrado en la arquitectura** donde se muestra una visión del sistema en general, **Iterativo e incremental** puesto que cada fase se desarrolla mediante iteraciones que involucra un conjunto de actividades y flujos de trabajo.

RUP posee varios elementos como por ejemplo: **Actividades** que no son más que los procesos que determinan en cada iteración, **Trabajadores** los cuales son las personas

involucradas en cada proceso y **Artefactos** que puede ser un documento, un modelo, o un elemento de modelo.

1.8.2. Extreme Programming (XP)

Esta metodología es usada en proyectos de corto plazo y poco personal, uno de sus requisitos es que el usuario final sea uno de los del equipo. Posee características como: la realización de las **Pruebas unitarias**, estas pruebas se realizan dependiendo de los errores que ocurrirán en el futuro; **Refabricación** la cual no es más que la reutilización de código utilizando patrones para que el software sea más flexible con los cambios; **Programación en pares**, esto permite que dos desarrolladores puedan trabajar al mismo tiempo en un determinado trabajo.

En esta metodología el cliente es quien decide qué es lo que se implementa, tiene derecho de quitar o cambiar los requisitos de acuerdo a su interés, el desarrollador simplemente decide cómo implementar los procesos, aclarar dudas con el usuario, estimar esfuerzos y cambiar requerimientos según nuevos descubrimientos.

Lo más importante de esta metodología es: la comunicación entre los usuarios y los desarrolladores; la simplicidad al desarrollar y codificar los módulos del sistema; la retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

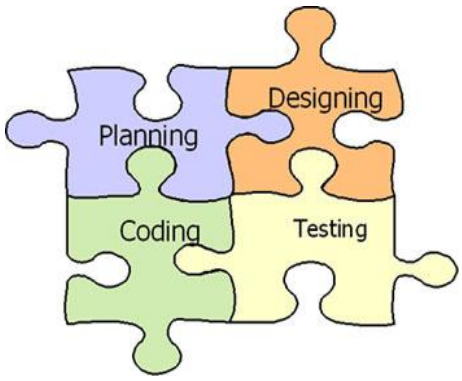

Fig. 7. Extreme Programming.

1.8.3. Microsoft Solutions Framework (MSF)

MSF está relacionada con conceptos, prácticas que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos, como primer objetivo tiene los modelos de procesos y equipos y en un segundo nivel las elecciones tecnológicas, uno de sus mejores aspectos es que sirve para cualquier tipo de proyecto, ya sea de corto o largo plazo y a cualquier tecnología.

Esta metodología es **adaptable** por lo que es usado en cualquier parte y en un lugar específico, tipo mapa; es **escalable** puesto que soporta cualquier tipo cantidad de personas, ya sea equipos de 3 hasta más de 50 personas; es **flexible** ya que es utilizado en el ambiente de cualquier cliente; posee una **tecnología agnóstica** por poderse usar en soluciones sobre cualquier tipo de tecnología.

Posee varios modelos, entre los cuales se encuentra el modelo de arquitectura del proyecto, modelo del equipo, modelo de proceso, modelo de gestión del riesgo, modelo de diseño de proceso y el modelo de aplicación. Los modelos están basados en fases (previsión, planeamiento, desarrollo, estabilización e implementación), puntos de transición y de carga de forma iterativa que se puede aplicar en el desarrollo de aplicaciones tradicionales, soluciones empresariales para comercio electrónico así como aplicaciones Web distribuidas.

Fig. 8 Microsoft Solutions Framework

1.8.4. ArgoUML

ArgoUML es una herramienta para modelar sistemas, mediante el cual se realizan diseños en UML [Lenguaje Unificado de Modelado] basada en Java. Puede crear la mayoría de los diagramas estándares de UML (10).

Esta herramienta es de código abierto y licencia libre, además genera código de java. Esta herramienta está basada en estándares abiertos: XMI, SVG y PGML, posee características como: multiplataforma (Java 1.2); permite algunos diagramas: estado, casos de uso, actividad, colaboración, despliegue, secuencia, tiene soporte para base de datos y exporta los diagramas a distintos formatos. Tiene como desventaja que consume muchos recursos de la máquina de java, es de desarrollo incompleto y no posee el botón deshacer.

1.8.5. BPwin

BPwin es una herramienta que es utilizada para analizar, documentar y mejorar procesos de negocio, proporciona en tiempo real qué es lo que hace la organización y si esta trabaja de forma eficiente.

Esta herramienta posee una interface intuitiva, diseño automatizado de Procesos, propiedades definidas por el usuario, técnicas de Integración, métrica y análisis de Costos, pre-evaluación, explorador de modelos, diccionarios, diálogo de propiedades y asociación de entidades y datos.

1.8.6. Rational Rose

El Rational une al equipo de desarrollo a través del modelamiento. Es la herramienta predominante en el mercado para el análisis, modelamiento, diseño y construcciones orientado a objeto, puesto que tiene un buen soporte UML y al equipo, desarrollo basado en componentes con soporte para arquitecturas líderes en la industria y modelos de componentes, facilidad de uso, integración optimizada, entre otras características.

Esta herramienta permite visualizar, entender y refinar los requerimientos y arquitectura antes de realizar el código, lo cual permite desarrollar con mayor facilidad y rendimiento el software. Es muy bueno pero tiene como desventaja que necesita alta capacidad de

procesamiento. Aunque el ingeniero no tenga un conocimiento previo, con esta herramienta puede realizar una gestión aceptable que cumpla con todos los objetivos del cliente.

1.8.7. Lenguaje Unificado de Modelado (en inglés: Unified Modeling Language; UML)

UML salió por primera vez en 1997, y no es más que un conjunto de herramientas que permiten modelar (analizar y diseñar) sistemas orientados a objetos (11). Es utilizado para visualizar, especificar, construir, detallar los artefactos en el sistema y documentar un sistema de software. Este lenguaje consta de elementos, relaciones y varios tipos de diagramas, entre las características más destacadas están: viabilidad en la corrección de errores, desarrollo incremental e iterativo, participación del cliente en todas las etapas del proyecto.

Fig. 9. Lenguaje Unificado de Modelado.

1.8.8. Visual Paradigm

Visual Paradigm es una herramienta CASE (Computer-Aided Software Engineering) de uso libre, 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 dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Está diseñado para todos aquellos usuarios que quieran realizar software a grandes escalas y orientado a objeto. Esta herramienta apoya las notaciones de Java y UML

Fig. 10. Visual Paradigm.

Para dar cumplimiento a las peticiones del cliente y realizar la herramienta generadora de datos, se utiliza para este trabajo la metodología RUP, ya que describe paso a paso lo que el ingeniero debe de hacer y sus casos de uso representan lo que el usuario final realmente desea. También se trabaja con el Visual Paradigm por se una herramienta que permite dibujar todos los tipos de diagramas de clases, apoyar las notaciones de UML, generar código desde diagramas e importar y exportar imágenes.

1.9. Lenguaje de programación

1.9.1. Java

Java es un lenguaje simple que posee muchas características, es orientado a objeto, distribuido, robusto, seguro, indiferente a la arquitectura, portable, dinámico, es de alto rendimiento, produce applets[27], entre otras.

Proporciona un conjunto de clases potente y flexible, no posee punteros, variables globales, asignación de memoria, además en caso de que se quiera ejecutar el programa en otra máquina no es necesario de escribir el código de nuevo y las páginas web no tienen que ser estáticas, a parte de poseer todas estas ventajas y más, también tiene desventajas como por ejemplo su velocidad, ya que los programas hechos en java son interpretados, por lo que su velocidad no es el de un ejecutable.

[27] Applets: Componente de una aplicación.

1.9.2. PHP

PHP es un lenguaje de script libre y orientado a objeto, interpretado en el lado del servidor. Es utilizado para la generación de páginas Web dinámicas el cual es su principal objetivo. Se ejecuta en el lado del servidor, por lo que se puede acceder a los recursos del mismo, por ejemplo a una base de datos.

Posee grandes características como son: soporte para grandes cantidades de base de datos; integración con varias bibliotecas externas; permite una solución simple y universal para las paginaciones dinámicas del Web de fácil programación; es más fácil de mantener que el código desarrollado en otros lenguajes de programación. Su rapidez, facilidad de aprendizaje y su soporte multiplataforma es una de las grandes ventajas que tiene como servidores HTTP.
Es completamente expandible, no obstante, posee algunas contradicciones como lo es el manejo de errores, que no es tan sofisticado y que no posee un debugger o IDE[28].

1.9.3. .Net

.NET es un proyecto realizado por Microsoft con el objetivo de crear una plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones e independencia de lenguaje a los programadores. Provee servicios automáticos al código que se ejecuta, como por ejemplo cargador de clases, recolector de basura, verificador de tipos, entre muchos otros.

Este proyecto posee una biblioteca de clases que maneja las operaciones básicas, se podría mencionar la interacción con los dispositivos periféricos, manejo de los datos, administración de memoria, cifrado de datos, manejo y administración de excepciones, generación de código, operaciones aritméticas, etc.… Posee varios servicios entre los que se pueden destacar el acceso a las bases de datos, los servicios de directorio y las transacciones distribuidas.

[28] IDE**:** Entorno de desarrollo integrado (integrated development environment).

Dispone de Visual Studio.NET, la cual posee un editor de código en múltiples lenguajes, un compilador, editor de recursos, conexión a base de datos, editor XML, depurador, ayuda en línea, etc.

Fig11. Biblioteca de clases.

1.9.4. C#

Es un lenguaje simple de programación orientado a objeto lanzado por Microsoft, diseñado para escribir aplicaciones empresariales. Su código es parecido al de C++ y trabaja bajo la plataforma de .Net. Posee muchos compiladores, entre ellos está el Microsoft .Net framework SDK[29], Microsoft Visual Studio .NET, #develop es un IDE libre para C# bajo licencia LGPL, Mono desarrollado por Novell, Delphi 2006 de Borland Software Corporation, dotGNU Portable.NET, de la Free Software Foundation.

Proporciona seguridad de tipos, beneficio de los servicios de Common Language Runtime[30] (CLR[31]), permite diseñar aplicaciones sencillas utilizando el diseñador de interfaz de usuario intuitivo arrastrar y soltar. Los Snippets de código IntelliSense son fragmentos de código utilizados para completar los espacios en blanco, lo cual es muy útil y hace más fácil el trabajo, ya que simplifican la cantidad de código que se escribe a mano facilitando plantillas de código.

[29] SDK: kit de desarrollo de software (Software Development Kit).
[30] Runtime: Serie de DLL's que cualquier programa escrito y compilado en C o C++ necesita para su ejecución
[31] CLR: Lenguaje común de rutina (Common Language Runtime)

Se podrían destacar entre las características de C#:

- **Sencillez**: elimina elementos que otros lenguajes tienen y que aquí son innecesarios.
- **Modernidad:** incorpora elementos necesarios y que en otros lenguajes hay que simular**.**
- **Orientado a Objeto:** no admiten funciones ni variables globales, todo el código y datos han de especificarse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.
- **Eficiencia:** su código posee restricciones para alcanzar seguridad, no utiliza punteros.

1.9.5. Mono

Mono es un proyecto de código abierto iniciado por Ximian, basado en GNU/Linux y compatibles con .NET, fue sacado al mercado el 30 de junio de 2004, también es conocido como la implementación libre del CLI [32](Common Language Infraestructure) y C#. Su primer objetivo era implementar en un entorno de software libre para el mundo Unix las especificaciones ECMA[33].

Posee variados componentes para desarrollar software entre los que están: **Mono Runtime** el cual es parecido al CRL e implementa el JIT[34] para el CIL[35] de la máquina virtual, un compilador Ahead-of-Time (AOT), un cargador de clases, un recolector de basuras, las librerías de acceso a los metadatos y un intérprete denominado Mint, que puede ejecutarse sobre la plataforma HP-UX.

Funciona sobre varios sistemas operativos ya sea Linux, MacOS X, BSD, SUN SOLARIS y Microsoft Windows. El Runtime permite que las aplicaciones echas en mono se puedan comunicar con otras o utilizar librerías externas como por ejemplo PInvoke, que no es más que el mecanismo que se utiliza desde el código gestionado para acceder a librerías externas, esta se está utilizando en mono para acceder a llamadas de la API de Unix, así como para comunicarse con librerías del sistema; **Class Library** (librería de clases) la cual se está desarrollando en C# y puede ser utilizada por cualquier lenguaje debido al Common

[32] CLI: Lenguaje común de infraestructura (Common Language Infraestructure)
[33] ECMA: Asociación europea de computadoras (European Computer Manufacturers Association
[34] JIT: just-in-time
[35] CIL: Common Intermediate Language

Language Specification; **Compilador C#** de mono (MCS) que está escrito en C#. A parte de esto también tiene un compilador para los lenguajes MonoBasic, Java y Python.

1.10. Conclusiones

En este capítulo se realiza una recopilación de los conceptos necesarios para poder entender y desarrollar la aplicación deseada. Se concluye la herramienta que se utilizará para un mejor desarrollo de la herramienta es Visual Paradigm, la metodología RUP y el lenguaje de C#.

CAPÍTULO 2: REQUERIMIENTOS, ANÁLISIS Y DISEÑO DEL SISTEMA

2.1. Introducción

En el presente capítulo se lleva las necesidades del cliente al sistema, mediante la captación de las actividades fundamentales que el mismo debe cumplir, como también a través de un análisis y diseño más detallado, de forma tal que el usuario quede satisfecho con el software.

El modelamiento del negocio propuesto por RUP como parte inicial del proceso de desarrollo de software no se lleva a cabo en la presente investigación, debido a que solo se identificó un solo proceso y de poca complejidad en cuanto a su comprensión y dominio del problema. El mismo consiste en que cada desarrollador necesita introducir gran cantidad de datos al software para verificar si realmente cumple con los requisitos planteados.

En esta sección, se podrán observar algunas de las investigaciones realizadas para darle solución a la necesidad del cliente, se darán a conocer los requerimientos, los casos de uso del sistema así como sus descripciones, los diagramas existentes tanto del análisis como del diseño, obteniendo así las clases para su necesaria implementación

2.2. Requerimientos

En épocas anteriores, cada vez que se iba a realizar un software era un problema, puesto que el cliente pedía un producto y cada miembro del proyecto se llevaba una versión totalmente diferente, al final, el software quedaba completamente distinto a lo que realmente el cliente quería. En otras ocasiones era el usuario el que no se sabía explicar o no sabía realmente lo que deseaba, simplemente siempre era un gran problema que conllevaba horas de trabajo e insatisfacción con el usuario final.

Hoy en día es menos el trabajo que se pasa para ponerse de acuerdo con el cliente, ya que después de realizar una descripción del verdadero negocio se realiza una captura de requisitos, que no es más que lo que el cliente realmente desea y lo que el software en sí debe de cumplir para satisfacer a los usuarios finales. No obstante, en muchos casos se lleva una amplia comunicación con el usuario para poder realizar una buena captura de requisitos.

Por todo lo planteado anteriormente es que este flujo de trabajo posee una amplia importancia en la realización de cualquier sistema.

2.2.1. ¿Qué es Requerimiento?

La definición de requerimiento que se encuentra en la IEEE[36] (Standard Glossary of Engineering Terminology) no es más que la siguiente:

(a) una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo.

(b) una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal.

(c) una representación en forma de documento de una condición o capacidad como las expresadas en (a) o en (b).

Mientras que la que aparece en [DOD, 1994] es más concisa:

- Característica del sistema que es una condición para su aceptación.

Otra posible definición es la siguiente [GOG, 1994]:

- Propiedad que un sistema debería tener para tener éxito en el entorno en el que se usará

En general, los requerimientos no son más que requisitos indispensables que el software debe cumplir para llegar a satisfacer las necesidades del usuario final. Estos requerimientos deben ser especificados por escrito, posible de probar y verificar, conciso, completo, consistente y no ambiguo, además tienen como objetivo definir: el ámbito e interfaz de usuarios para el sistema, enfocada a las necesidades y metas del cliente; establecer y mantener un acuerdo entre clientes y otros involucrados sobre lo que el sistema debería hacer; proveer: a los desarrolladores un mejor entendimiento de los requerimientos del sistema, una base para estimar recursos y tiempo de desarrollo del sistema y para la planeación de los contenidos técnicos de las iteraciones.

[36] IEEE: Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers)

2.2.2. Requerimientos funcionales y No funcionales

2.2.2.1. Requerimientos Funcionales:

Los requerimientos funcionales describen las interacciones entre el sistema y su entorno (usuarios u otros sistemas), sin tener en cuenta cuestiones de implementación. Son capacidades o funciones que el sistema debe cumplir. Dentro de los requisitos funcionales se encuentran:

R1. Autenticar Usuario: el usuario una vez entrada a la página debe autenticarse con el puerto, IP del servidor, tipo de gestor, usuario y contraseña, para así conectarse a la base de datos del determinado gestor.

R1.1 Verificar entrada del puerto.

R1.2 Verificar entrada del servidor.

R1.3 Verificar tipo de gestor.

R1.4 Verificar usuario.

R1.5 Verificar contraseña.

R2. Mostrar tablas: el sistema a partir de la selección del usuario de una base de datos, debe ser capaz de mostrar las tablas contenidas en la misma.

R3.1 Mostrar tablas dado un nombre de una base de datos.

R3. Generar datos: el usuario una vez que seleccione una de las tablas a generar y la cantidad de filas con las que desea llenar los campos que posee la misma, el sistema debe de generar automáticamente estos datos.

R4.1 Permitir seleccionar la tabla y la cantidad de filas a generar.

R4. Guardar datos: una vez generados los datos, el sistema debe permitir guardarlos en la base de datos.

2.2.2.2. Requerimiento No Funcional

Describen aspectos del sistema visibles por el usuario que no se relacionan en forma directa con el comportamiento funcional del mismo (restricciones en el tiempo de respuesta, precisión de los resultados, etc.). Son propiedades o cualidades que el producto debe tener. El sistema cuenta con varios tipos de requerimientos no funcionales de los cuales se mencionan algunos a continuación:

- **Requerimientos de software**

  Se debe disponer del Sistema Operativo Windows.

- **Requerimientos en el diseño y la implementación.**

  La herramienta a utilizar debe ser el Visual Paradigm por ser libre y orientado a objeto, con la arquitectura de tres capas para poder separar la lógica del negocio y el lenguaje de C#.

- **Requerimientos de apariencia o interfaz externa.**

  El sistema consta de una aplicación web, con una interfaz de fácil accesibilidad y uso para los usuarios.

- **Requerimientos de Seguridad**

  Confidencialidad: Los usuarios deberán de autenticarse para poder tener acceso a la base de datos del gestor de un determinado servidor.

  Integridad: La información almacenada tendrá que ser la adecuada para aquellos usuarios que puedan acceder a ella.

  Disponibilidad: La información almacenada tendrá que estar disponible para todos aquellos usuarios que puedan acceder a ella de manera rápida y precisa.

- **Requerimientos de Usabilidad**

  El sistema lo podrá usar cualquier tipo de persona que posee la autenticación adecuada de algún determinado servidor.

- **Requerimientos de Soporte**

Una vez terminado el software se les deben de hacer las pruebas y configuraciones de los puertos necesarios para que el usuario pueda acceder a su servidor de manera rápida y exitosa.

2.2.3. Actores del sistema

Actores del sistema y sus funciones

Actores	***Justificación***
Usuario	Es el que tiene acceso al sistema web, por lo cual con la debida autentificación accede a las toda la información de un determinado gestor.
Gestor de BD	Mediante este actor se obtiene toda la información requerida por el usuario.

Tabla 2.1 Actores del Sistema

Representación de los actores del sistema

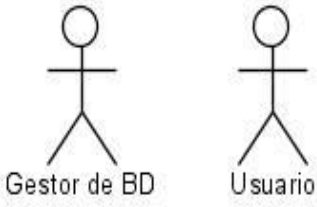

Fig. 2.1 Actores del Sistema

2.2.4. *Diagrama de caso de uso del sistema* (DCUS)

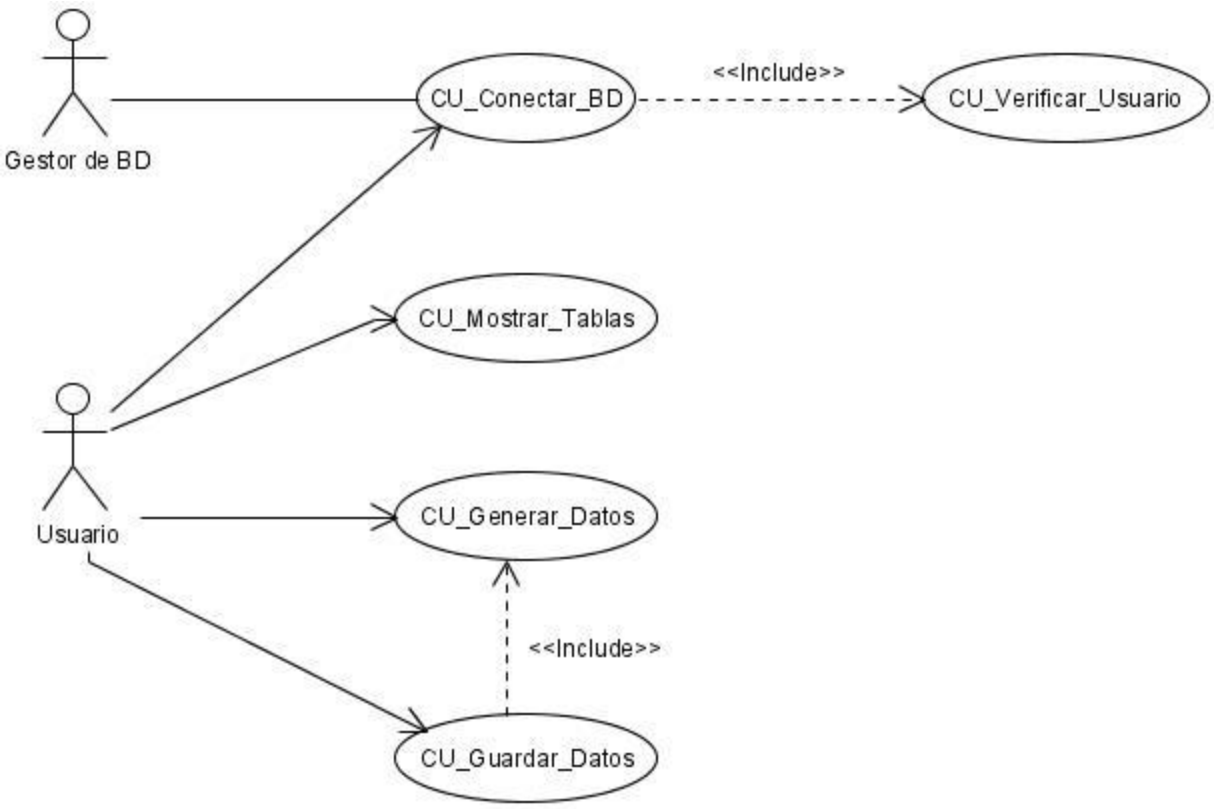

Fig. 2.2 Diagrama del Caso de Uso del Sistema

2.2.5. 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 correspondiente con este fin, (Ver Anexo 1).

Descripciones de los casos de uso del sistema

Caso de Uso (CU_1)	CU_Conectar_BD
Actor	Usuario (Inicia)
Propósito	Conectarse al servidor para obtener toda la información de un gestor determinado.
Resumen	El caso de uso se inicia cuando el usuario seleccione la

	opción que le permite conectarse al servidor.
Precondiciones	El usuario debe haberse autenticado.
Poscondiciones	
Referencia	R1
Casos de Uso Relacionados	CU_Verificar_Usuario (Incluido)
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
2. El usuario se autentica con el servidor, puerto, tipo de gestor, base de datos, usuario y contraseña. 3. El usuario elige la opción de conectarse.	1. El sistema muestra una interfaz para que el usuario entre sus datos: servidor, puerto, tipo de gestor, base de datos, usuario y contraseña. 3.1. El sistema verifica que ninguno de los campos a llenar por el usuario estén vacío. 3.2. El sistema verifica que los datos entrados por el usuario estén correctos. 3.3. El sistema le permite al usuario obtener todas las tablas de la base de datos de un determinado gestor.
Flujos Alternos	
	3.1 Si el sistema verifica que algún campo de los que debe llenar el usuario está vacío emite un mensaje de error: “Debe de llenar todos los campos”. 3.2 Si el sistema verifica que algún dato esté incorrecto emite un mensaje de error: “Datos de entrada incorrectos”.
Prioridad	Crítico

Tabla 2.2 Caso de Uso <<CU_Conectar_BD>>

Caso de Uso (CU_2)	CU_Mostrar_Tablas
Actor	Usuario (Inicia)
Propósito	Mostrar todas las tablas de una base de datos determinada.
Resumen	El caso de uso se inicia cuando el usuario seleccione la opción que le permite acceder a las tablas.
Precondiciones	El usuario debe haber seleccionado al menos una base de datos.
Poscondiciones	
Referencia	R2
Casos de Uso Relacionados	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la base de datos de las cuales desea ver las tablas. 2. El usuario elige la opción de ver tablas.	2.1. El sistema verifica que se haya seleccionado una base de datos. 2.2. El sistema muestra todas las tablas de la base de datos señalada.
Flujos Alternos	
	2.1. Si el sistema verifica que no hay ninguna base de datos señalada emite un mensaje de error “Señale al menos una base de datos”.
Prioridad	

Tabla 2.3 Caso de Uso <<CU_Mostrar_Tablas>>

Caso de Uso (CU_4)	CU_Generar_Datos
Actor	Usuario (Inicia)
Propósito	El sistema debe generar los datos automáticamente.
Resumen	El caso de uso se inicia una vez que el usuario seleccione la opción generar.
Precondiciones	El usuario debe haber seleccionado la cantidad de filas que desea poblar y seleccionado la opción generar.
Poscondiciones	El sistema le muestra al usuario todos los campos poblados.
Referencia	R3
Casos de Uso Relacionados	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario elige la cantidad de filas a poblar y la tabla a la cual desea introducirle los datos. 2. El usuario elige la opción de generar datos.	2.1. El sistema verifica que se haya seleccionado una tabla y elegido la cantidad de filas a llenar. 2.2. El sistema le permite al usuario ver todos los campos con los datos generados.
Flujos Alternos	
	2.1. Si el sistema verifica que no se ha elegido la tabla o la cantidad de filas a llenar emite un mensaje de error: “Debe de elegir una tabla y la cantidad de filas que desea generar”.
Prioridad	

Tabla 2.4 Caso de Uso <<CU_Generar_Datos>>

Caso de Uso (CU_5)	CU_Guardar_Datos
Actor	Usuario (Inicia)
Propósito	Guardar todos los campos de las tablas llenos.
Resumen	El caso de uso se inicia cuando el usuario seleccione la opción que le permite guardar los campos.
Precondiciones	Deben de estar todos los campos llenos.
Poscondiciones	Se guardan los campos modificados.
Referencia	R4
Casos de Uso Relacionados	CU_Generar_Datos (Incluido)
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario verifica que todos los campos estén llenos. 2. El usuario elige la opción de guardar.	2.1. El sistema le permite al usuario guardar los datos.
Flujos Alternos	
Prioridad	

Tabla 2.5 Caso de Uso <<CU_Guardar_Datos >>

Caso de Uso (CU_6)	CU_Verificar_Usuario
Actor	Usuario (Inicia)
Propósito	Verificar que el usuario esté autorizado a ver las bases de datos de un determinado servidor.
Resumen	El caso de uso se inicia cuando el usuario seleccione la opción que le permite acceder al servidor.
Precondiciones	El usuario debe haber llenado todos los campos con

	sus datos.
Poscondiciones	El usuario accede a la información de la base de datos que contiene el servidor.
Referencia	R1
Casos de Uso Relacionados	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario introduce todos los datos. 2. El usuario da en la opción conectarse.	2.1. El sistema verifica que se hayan llenado todos los campos con los datos del usuario. 2.1. El sistema verifica que los datos introducidos estén correctos. 2.2. El sistema le permite al usuario conectarse al servidor seleccionado.
Flujos Alternos	
	2.1. Si el sistema verifica que algún campo está vacio emite un mensaje de error: “Debe de llenar todos los campos”. 2.2. Si el sistema verifica que algún dato del usuario no es válido emite un mensaje de error:”Datos incorrectos”.
Prioridad	

Tabla 2.6 Caso de Uso <<CU_Verificar_Usuario>>

2.3. Análisis del sistema

Hasta ahora lo que se ha echo es simplemente, definir las actividades fundamentales que debe cumplir el sistema para satisfacer las necesidades del cliente y también de esa forma poder obtener una visión externa del sistema. De ahora en adelante esos requisitos se

transforman en el cómo implementar el sistema, por lo que el análisis se centra fundamentalmente en aquellos requisitos funcionales. A continuación se destacan los diagramas fundamentales de este flujo de trabajo.

2.3.1. Diagramas de clases del análisis

Las clases del análisis representan conceptos y relaciones del dominio, dentro de ellas encontramos la clase interfaz, que no es más que la interacción del sistema con el usuario; la controladora, las cuales definen el control de las transacciones dentro de un caso de uso y la entidad que poseen información necesaria y de larga vida. Ahora se encuentran detallados los diagramas de clase del análisis para cada caso de uso del sistema.

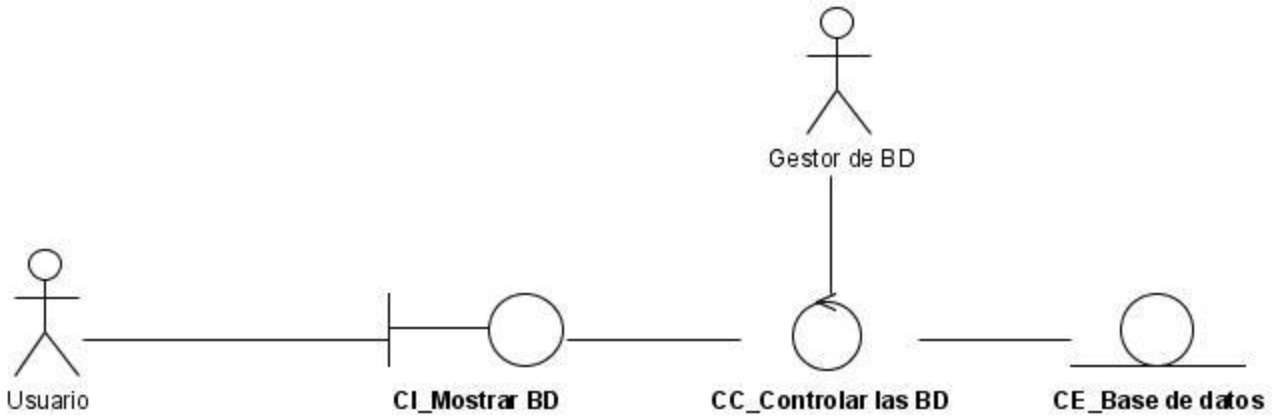

Fig. 2.3 Clase del análisis <<CU_Conectar_BD>>

Fig. 2.4 Clase del análisis <<CU_Mostrar_Tablas>>

Fig. 2.5 Clase del análisis <<CU_Generar_Datos>>

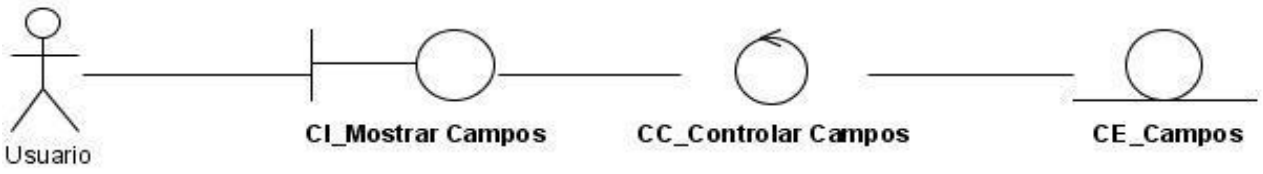

Fig. 2.6 Clase del análisis <<CU_Guardar_Datos>>

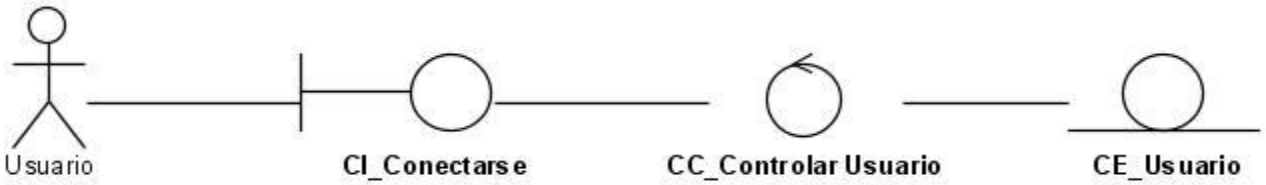

Fig. 2.7 Clase del análisis <<CU_Verificar_Usuario>>

2.3.2. 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 diagrama de interacción: el diagrama de colaboración, que representan lo principal de un escenario, mostrando las interacciones de la organización estructural de los objetos; el diagrama de secuencia, el cual es un diagrama de interacción que destaca la ordenación temporal de los mensajes. A continuación se muestran los diagramas de colaboración perteneciente a este sistema.

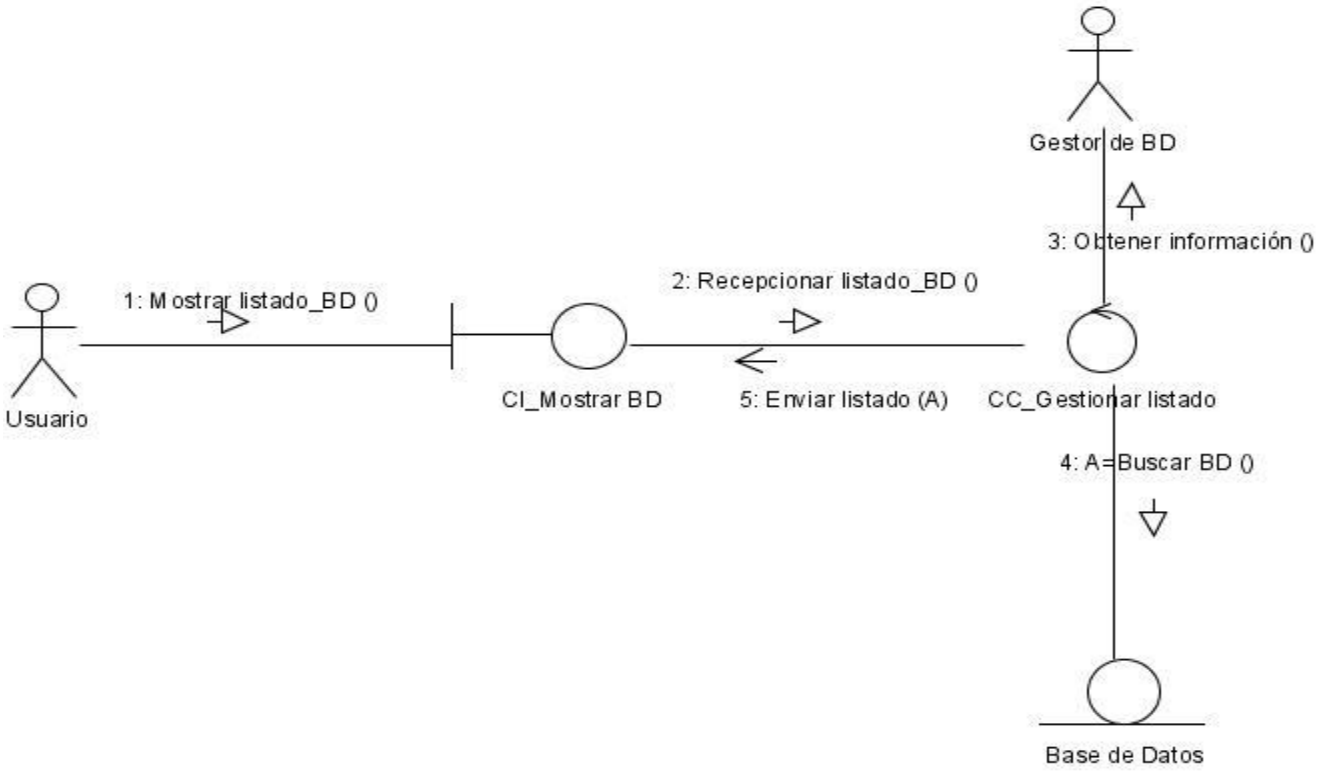

Fig. 2.8 Diagrama de Colaboración <<Conectar_BD>>

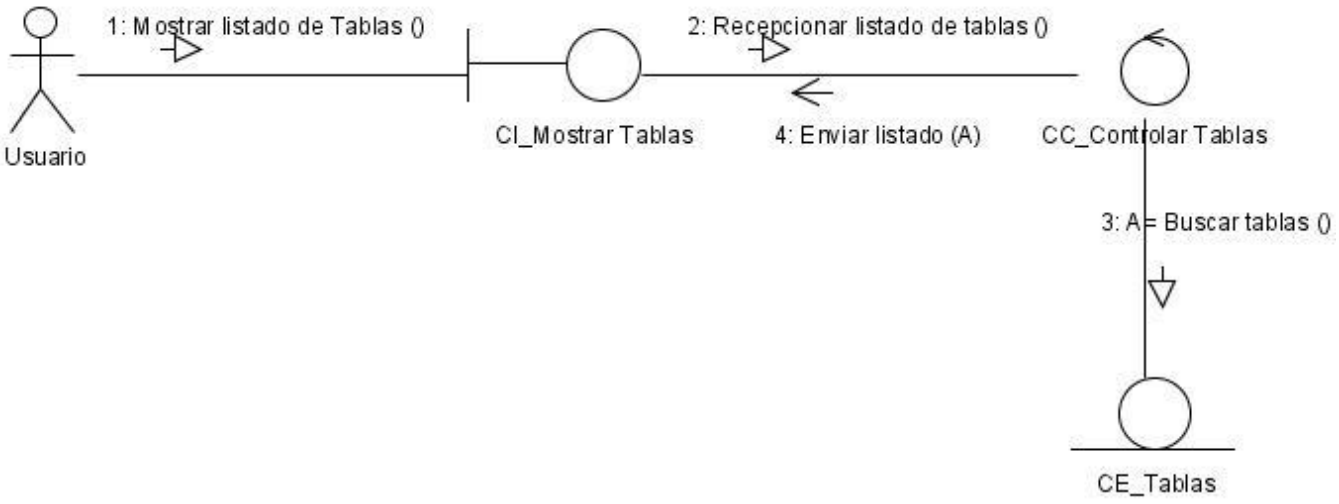

Fig. 2.9 Diagrama de Colaboración <<CU_Mostrar_Tablas>>

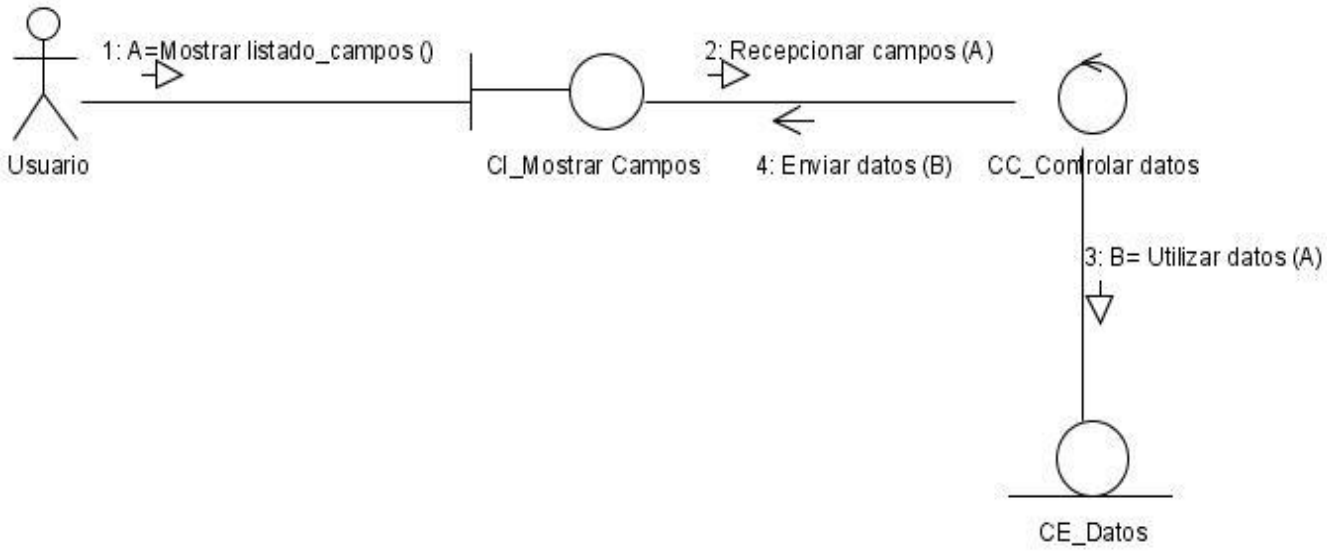

Fig. 2.10 Diagrama de Colaboración <<CU_Generar_Datos>>

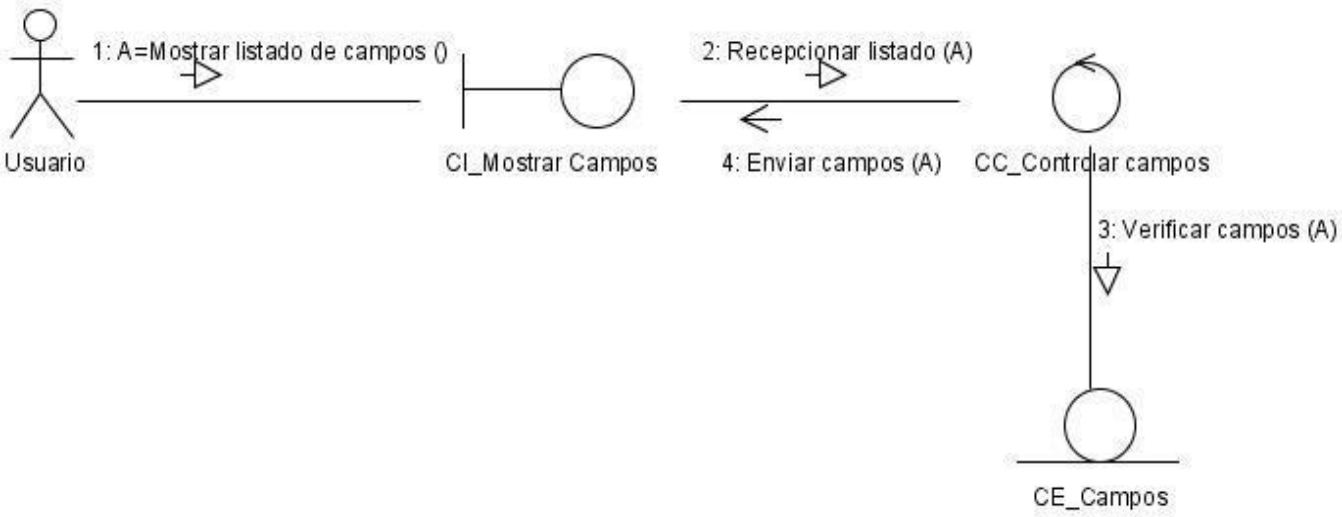

Fig. 2.11 Diagrama de Colaboración <<CU_Guardar_Datos>>

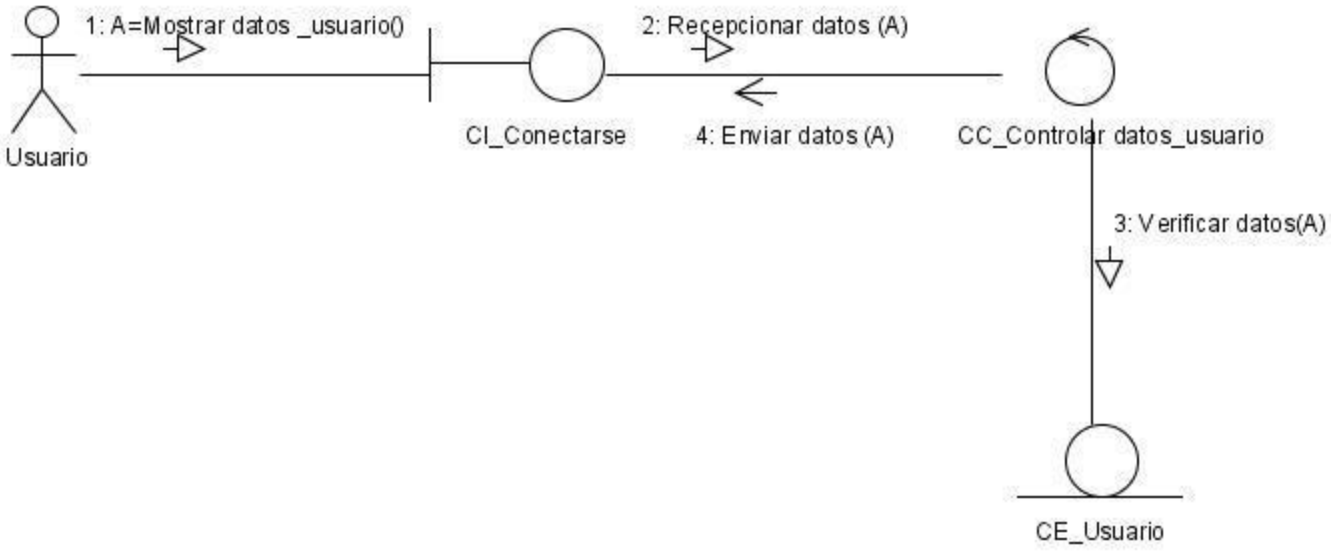

Fig.2.12 Diagrama de Colaboración <<CU_Verificar_Usuario>>

2.4. Diseño del sistema

En el diseño no se hace más que refinar lo que se hizo en el análisis, teniendo en cuenta los requisitos no funcionales. Debe de realizarse de forma tal que a la hora de implementar no existan dudas ni ambigüedades.

2.4.1. Diagrama de clases del diseño

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

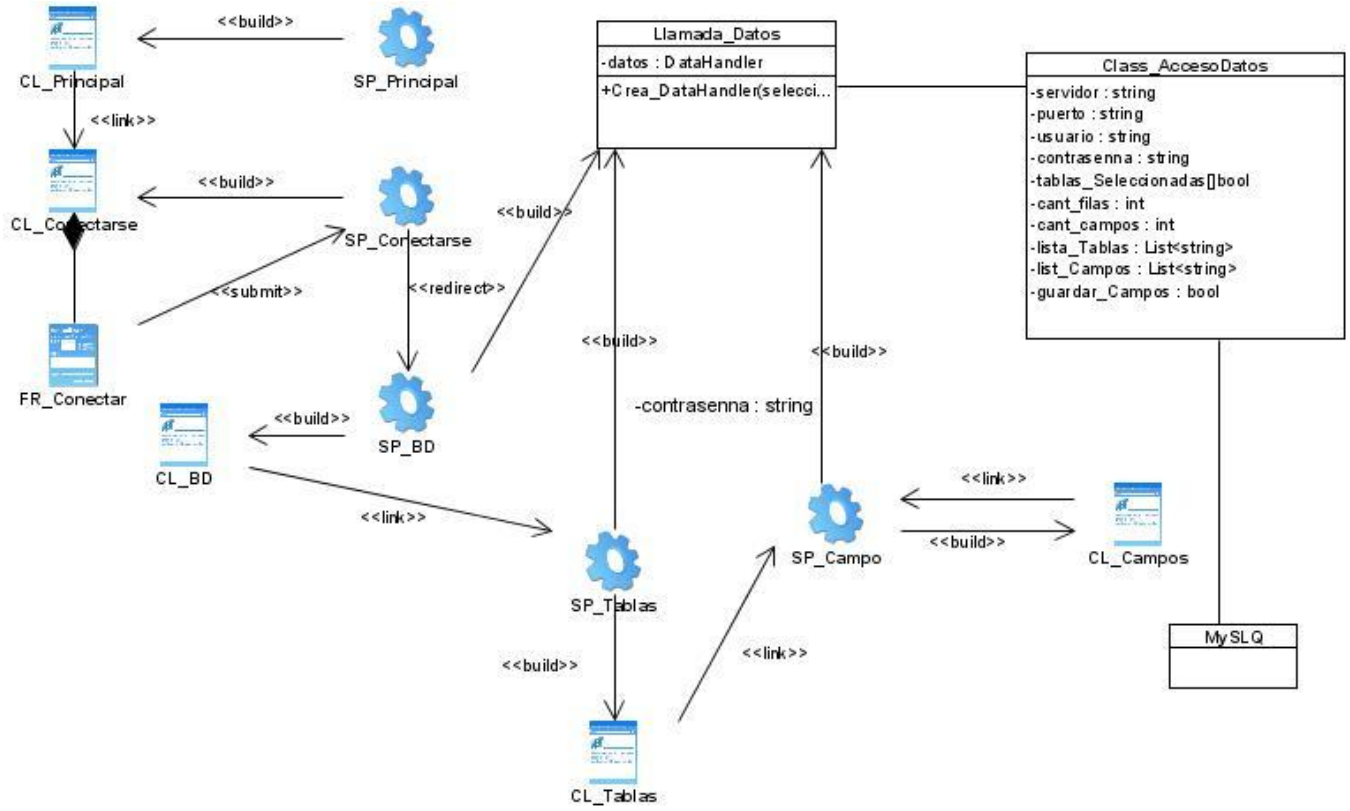

Fig. 2.13 Clase del Diseño

2.4.2. Diagramas de interacción del diseño

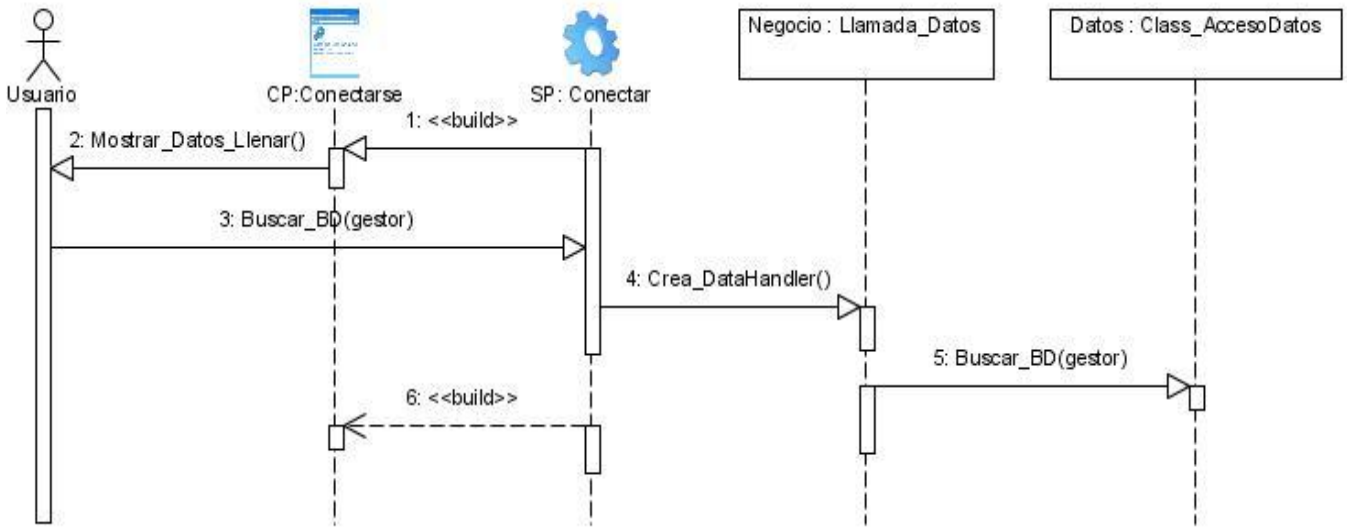

Fig. 2.14 Diagrama de Secuencia<<CU_Conectar_BD>>

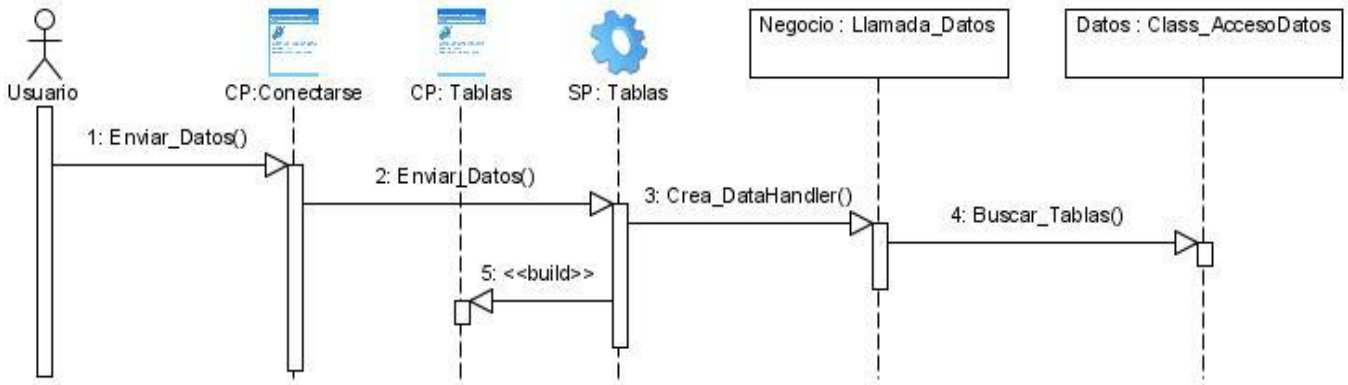

Fig. 2.15 Diagrama de Secuencia <<CU_Mostrar_Tablas>>

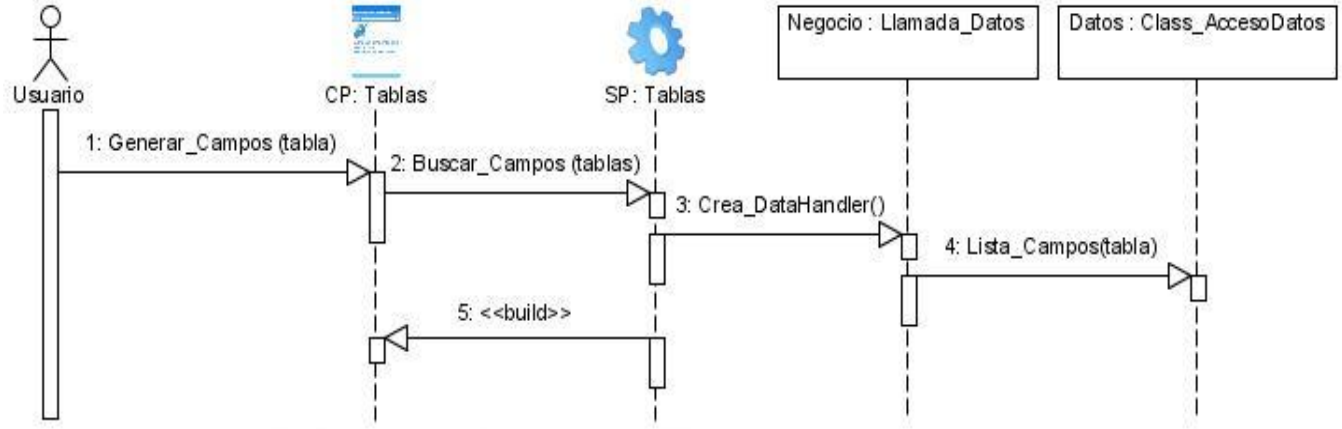

Fig. 2.16 Diagrama de Secuencia <<CU_Generar_Datos>>

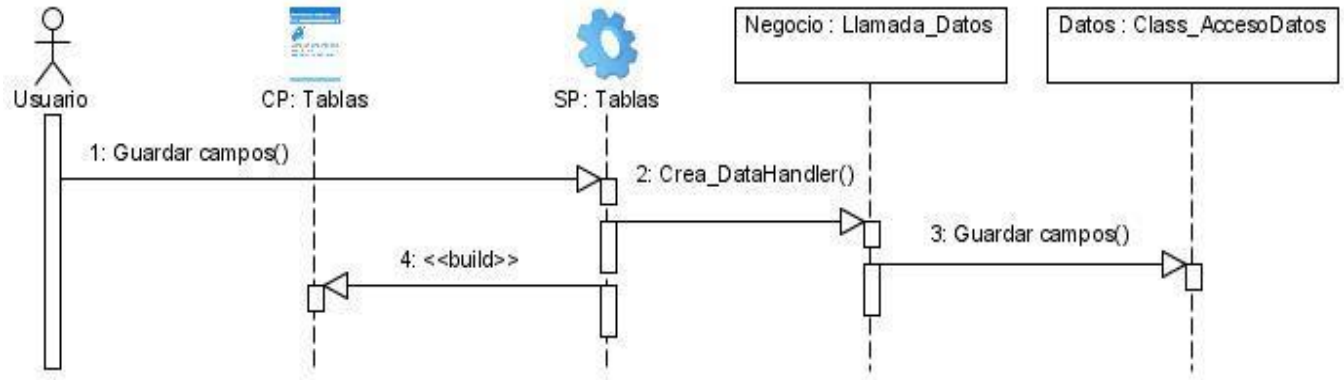

Fig. 2.17 Diagrama de Secuencia <<CU_Guardar_Datos>>

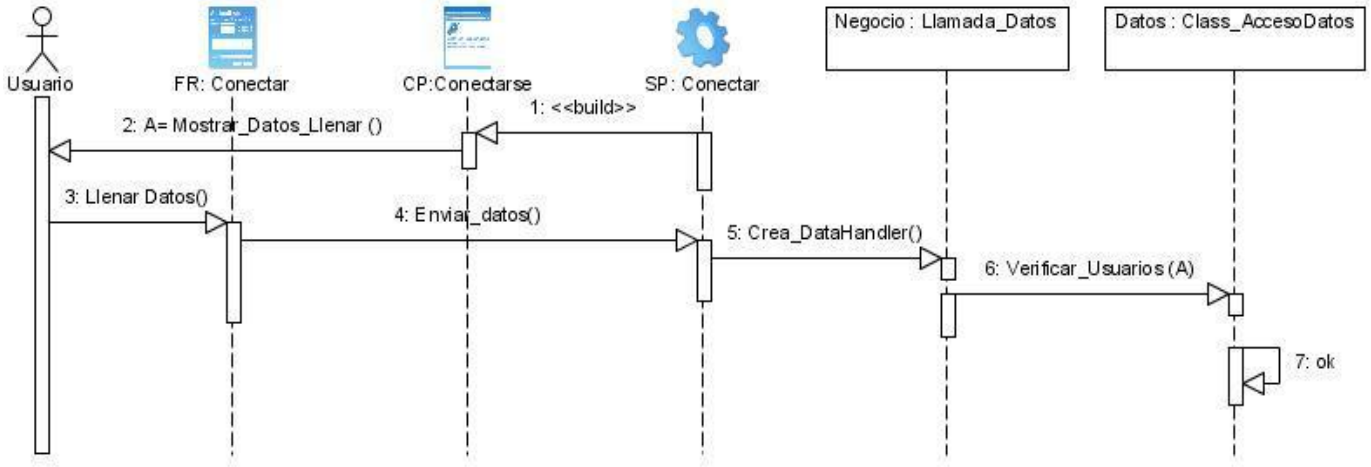

Fig. 2.18 Diagrama de Secuencia <<CU_Verificar_Usuario>>

2.4.3. Diagrama de despliegue

El diagrama de despliegue tiene como objetivo capturar los elementos de configuración de las conexiones.

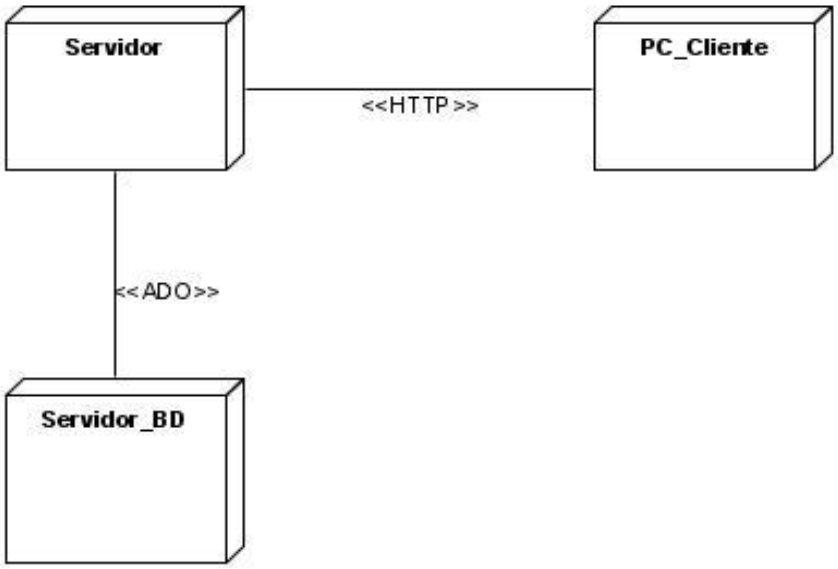

Fig. 2.19 Diagrama de Despliegue del Diseño

2.5. Conclusiones

En este capítulo se recogen los principales requerimientos funcionales y no funcionales, con el objetivo de realizar una aplicación que satisfaga las necesidades reales del cliente. Se realiza el análisis y diseño necesario para llevar estos requerimientos al sistema y así tener una visión clara a la hora de implementar.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1. Introducción

En el presente capítulo, se lleva el resultado del diseño a un lenguaje de programación, es decir, se implementa el sistema en términos de componentes. Se trata de dar respuesta a la petición del cliente por medio de una aplicación, a la cual después de terminada, se le realizan las pruebas necesarias para encontrar los errores antes de que el software sea entregado al usuario final.

Se especifican los artefactos que conforman el modelo de implementación (diagrama de despliegue y de componente), así como los casos de prueba necesarios para encontrar los errores al software

3.2. Implementación

Este flujo de trabajo tiene como objetivo completar la implementación de la línea base de la arquitectura, asignar componentes ejecutables a nodos en el diagrama de despliegue, definir la organización del sistema en subsistemas e implementar los elementos del diseño.

3.2.1. Diagrama de Componente

El diagrama de componente muestra los subsistemas de implementación organizados en capas y sus dependencias.

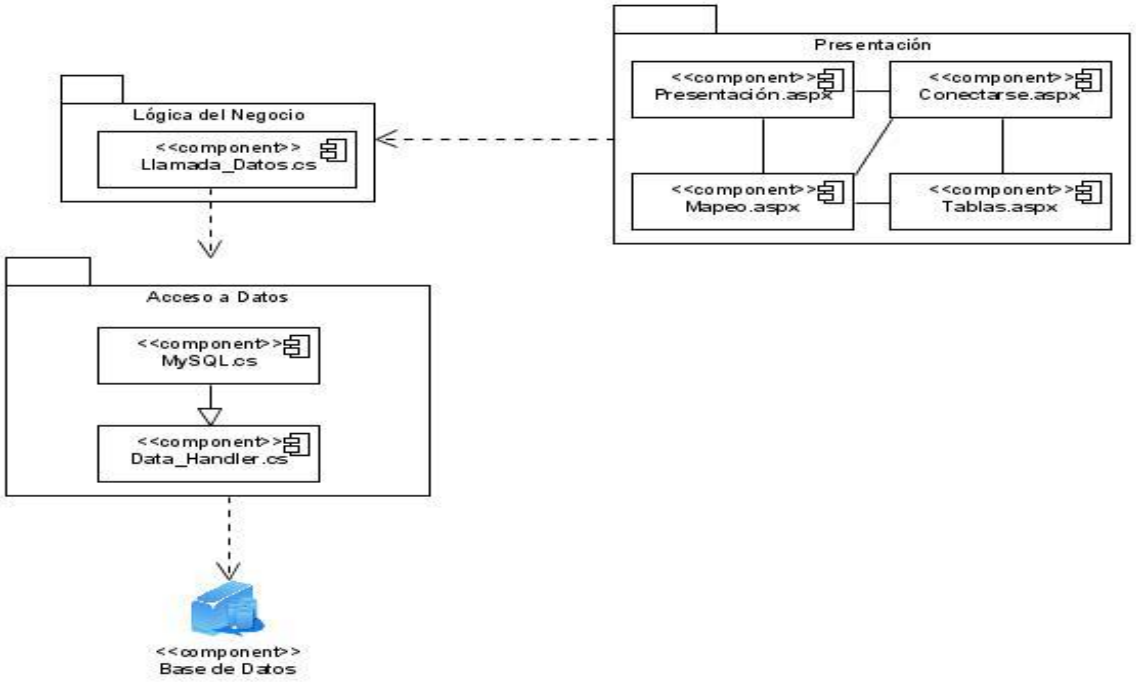

Fig. 3.1 Diagrama de Componente

3.2.2. Diagrama de Despliegue

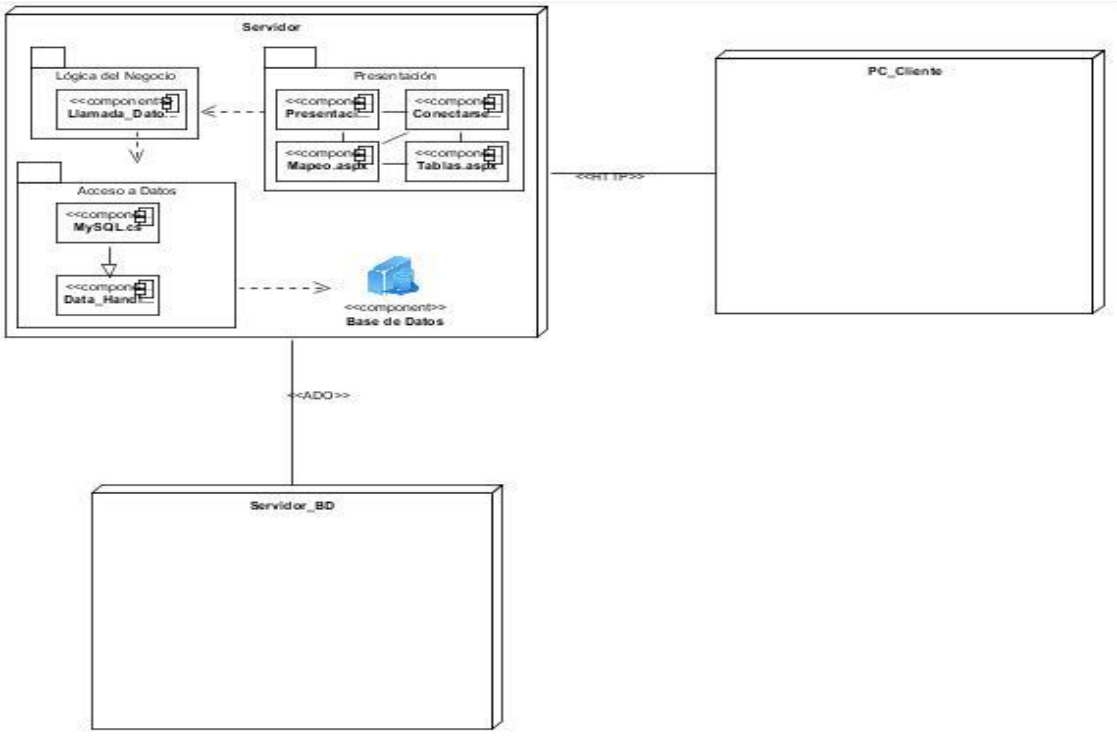

Fig. 3.2 Diagrama de Despliegue (Implementación)

3.3. Prueba

Las pruebas son realizadas con el objetivo de encontrar la mayor parte de los errores antes de que el software sea entregado al cliente. Tiene como objetivo demostrar y hacer posible que la aplicación trabaje con la mayor calidad posible y que satisfaga las necesidades del usuario final. Existen dos métodos de prueba, la prueba de Caja Blanca y la prueba de Caja Negra.

Las pruebas de **Caja Negra** no son más que experimentos 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. Permite examinar el estado del programa en varios puntos y así poder verificar si su estado real coincide con el esperado.

Al sistema se le aplicó la prueba de Caja Negra, a continuación se muestran los casos de prueba[37], la misma consiste en describir ciertos parámetros del caso de uso basándose en la plantilla correspondiente con este fin, (Ver Anexo 2).

- **1** Caso de Prueba donde los datos del servidor, puerto, usuario y contraseña sean correctos.
- **2** Caso de Prueba donde los datos del servidor, puerto y usuario sean correctos y la contraseña sea incorrecta.
- **3** Caso de Prueba donde los datos del servidor, puerto y contraseña sean correctos y el usuario sea incorrecta.
- **4** Caso de Prueba donde los datos del servidor, usuario y contraseña sean correctos y el puerto sea incorrecto.

[37] **Casos de prueba:** Conjunto de entradas de pruebas con condiciones y resultados esperado0073

Caso de Uso	CU_Conectar_BD		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Servidor = ”127.0.0.1” Puerto = ”3306” Usuario = ”root” Contraseña = ”tesis”	Al ser los datos correctos el usuario debe de conectarse al servidor satisfactoriamente.	Se visualizan todas las tablas.
2	Servidor = ”127.0.0.1” Puerto = ”5432” Usuario = ”root” Contraseña = ”lolo”	Como la contraseña es incorrecta no se puede conectar al servidor y debe de notificarse un mensaje de error.	Muestra mensaje: “Contraseña incorrecta.”
3	Servidor = ”127.0.0.1” Puerto = ”5432” Usuario = ”pepe” Contraseña = ”tesis”	Como el usuario es incorrecto no se puede conectar al servidor y debe de notificarse un mensaje de error.	Muestra mensaje: “Usuario incorrecto.”
4	Servidor = ”127.0.0.1” Puerto = ”1001” Usuario = ”pepe” Contraseña = ”tesis”	Como el puerto es incorrecto no se puede conectar al servidor y debe de notificarse un mensaje de error.	Muestra mensaje: “Puerto incorrecto, por favor inserte el 3306.”

Tabla 3.1. Caso de prueba Conectar a la Base de Datos

- **1** Caso de Prueba donde el usuario introduce una base de datos correcta.
- **2** Caso de Prueba donde el usuario introduce una base de datos incorrecta.

Caso de Uso	CU_Mostrar_Tablas		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Base de datos = “tesis”	Al ser seleccionada la base de datos se pueden mostrar las tablas.	Se visualizan las tablas de la base de datos seleccionada.
2	Base de datos = “”	Como no ha sido seleccionada ninguna base de datos no se pueden mostrar las tablas.	Muestra mensaje: “Seleccione una base de datos.”

Tabla 3.2. Caso de prueba Mostrar Tablas

Caso de Uso	CU_Generar_Datos		
Caso de Prueba	**Entrada**	**Condiciones**	**Resultados**
1	Tabla = “diormis”	Al ser seleccionada la tabla se pueden generar los datos.	Se visualizan los campos de la tabla seleccionada con los valores generados.
2	Tabla = “”	Como no ha sido seleccionada ninguna tabla no se pueden generar los datos.	Muestra mensaje: “Seleccione al menos una tabla.”

Tabla 3.3. Caso de Uso Generar datos.

3.4. Conclusiones

La implementación realizada es para darle la solución al problema planteado. Las pruebas de este capítulo se realizan para garantizar la calidad de la aplicación resultante y garantizar que el software llegue con menos errores al cliente final.

CONCLUSIONES

Con la realización de esta investigación se puede llegar a la conclusión que al utilizar una herramienta capaz de poblar las base de datos de manera automática, los desarrolladores emplean menos tiempo a la hora de probar su software, ya que no deben de llenar las bases de datos de manera manual.

Se hizo un estudio sobre los diferentes generadores de datos, gestores de base de datos, herramientas, metodologías y lenguajes existentes. Este trabajo sigue la dinámica de RUP basada en fases y flujos de trabajo, lo que permitió obtener una aplicación web con una interfaz amigable para el usuario, siendo su principal objetivo el de lograr una herramienta que genere datos aleatorios ayudando de esta forma a los desarrolladores en la fase de prueba de su software.

RECOMENDACIONES

- Se exhorta a los programadores a investigar sobre el compilador mono del lenguaje de C# y así se podrá llevar esta aplicación a software libre.
- Se sugiere realizar investigaciones sobre la integridad referencial y llevarlo a cabo para esta herramienta.

BIBLIOGRAFÍA REFERENCIADA

(1) **Moreno, Mª Antonia García.** Concepto de base de datos. p. 5.

(2) **García, Rosa María Mato.** *Sistemas de Bases de Datos.* [ed.] Ing. José Quesada Pantoja. Segunda Edición. s.l. : Pueblo y Educación, 2005. p. 3

(3) *Sistemas de Base de Datos.* [ed.] Ing. José Antonio Quesada. Segunda Edición. s.l. : Pueblo y educación, 2005. p. 4. 959-13-1273-3

(4) *freedownloadmanager.* [Online] 2007. [Cited: febrero 5, 2008.] http://www.freedownloadmanager.org/es/downloads/Generador_de_Datos_de_SME_2005_para_MySQL_37201_p

(5) *fileheaven.com.* [En línea] http://www.fileheaven.com/descargar/ems-data-generator-for-sql-server/62642.htm

(6) *freedownloadscenter.* [Online] [Cited: febrero 5, 2008.] http://www.freedownloadscenter.com/es/Programacion/Base_de_Datos_y_Redes/EMS_Data_Generator_2005_for_Oracle.html

(7) *freedownloadscenter.* [En línea] 13 de febrero de 2007. [Citado el: 5 de febrero de 2008.] http://www.freedownloadscenter.com/es/Programacion/Base_de_Datos_y_Redes/EMS_Data_Generator_2005_for_PostgreSQL.html

(8) *freedownloadmanager.* [En línea] 2007. http://www.freedownloadmanager.org/es/downloads/Generador_de_Datos_de_SME_2005_para_DB2_47112_p/

(9) *abcpedia.* [Online] [Cited: Marzo 17, 2008.] http://www.abcpedia.com/diccionario/definicion-internet.html.

(10) *lugcix.* [En línea] http://www.lugcix.org/tutoriales/argouml/uml_linux.php

(11) **Garcia, Joaquin.** *ingenierosoftware.* [En línea] 7 de Mayo de 2005. http://www.ingenierosoftware.com/analisisydiseno/uml.php

BIBLIOGRAFÍA CUNSALTADA

García, Rosa María MAto. *Sistemas de Bases de Datos.* [ed.] Ing. José Quesada Pantoja. Segunda Edición. s.l. : Pueblo y Educación, 2005. págs. 4-6. 959-16-1273-3

Worsley, John y Drake, Joshua. *PostgreSQL Práctico.* [ed.] Andrew Brookins y Michael Holloway. [trad.] Adolfo Pachón. pág. 1.7

tiendalinux. [En línea] 31 de Mayo de 2003. http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html

DBAsupport. [En línea] 6 de febrero de 2006. http://www.dbasupport.com.mx/index.php?option=com_content&task=view&id=118&Itemid=30

fileheaven.com. [En línea] http://www.fileheaven.com/descargar/ems-data-generator-for-sql-server/62642.htm

Rojas, Jose Fabricio. *wordpress.* [En línea] http://devsoftx.wordpress.com/2008/04/24/arquitectura-en-n-capas

Bastidas, Jhuliana Lizeth Bautista. blog grupo de Telecomunicaciones. [En línea] http://www.utpl.edu.ec/blog/jlbautista/2008/02/04/10/#more-10

Generator FD. [En línea] http://www.generatorfd.com/Arquitectura.aspx

Oktaba, Hanna. [En línea] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html

Sanchez, María A. Mendoza. informatizate. [En línea] 7 de junio de 2004. [Citado el: 10 de febrero de 2008.]

http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

Paredes, Edwin Ángeles. [En línea] http://www.usmp.edu.pe/publicaciones/boletin/fia/info7/bpwin.htm

IBM Rational Rose. pág. 8. G507-1936-00

visual-paradigm. [En línea] http://www.visual-paradigm.com/product/vpuml/

Grau, Xavier Ferré y Segura, María Isabel Sánchez. clikea.com. [En línea] 1.2. http://www.clikear.com/manuales/uml/

unav. [En línea] http://www.unav.es/cti/manuales/Java/indice.html

Hinostroza, Raul Rodas. linuxcentro. [En línea] 2, 22 de febrero de 2007 http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP

microsoft. [En línea] http://www.microsoft.com/spanish/msdn/netframework/framework20_InformacionCaract.mspx.

microsoft. [En línea] http://msdn.microsoft.com/es-es/library/410zh1ty(VS.80).aspx

Seco, José Antonio González. El lenguaje de programación C#. pág. 264.

Ojeda, Francisco Charte. Visual c# .Net. [ed.] Juan Ignacio Luca de Tena. s.l. : ANAYA MULTIMEDIA. pág. 637. 84-415- 1392-9.

mono. [En línea] http://www.mono-project.com/Main_Page

gotmono. [En línea] http://www.gotmono.com/

mono. [En línea] [Citado el: 11 de marzo de 2008.] http://www.go-mono.com/docs/

mono. [En línea] http://www.mono-project.com/PostgreSQL

González, MSc. Guillermo. Ingeniería de Requerimientos. 2008. págs. 2-4. Vol. II.

Navarro, Ing. Jose Angel Franco. UML en acción. Modelando Aplicaciones Web. La Habana : s.n.

Cid, M. A. (2007). *Como hacemos del software un producto de ingeniería.* Madrid.

lab. [En línea] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/calidad.htm

José, Ferrón María y Pablo, González Juan. ldc.usb. [En línea] http://www.ldc.usb.ve/~teruel/ci3711/test1/index.html

Panadero, M. Carmen Fernández y Román, Julio Villena. Pruebas de Programación.

Martín, Tomás. elguille. [En línea] 12 de Octubre de 2005. http://www.elguille.info/colabora/NET2005/Tomasmm_3Capas.htm

pgfoundry. [En línea] http://pgfoundry.org/frs/?group_id=1000140

tldp. [En línea] http://es.tldp.org/Postgresql-es/web/navegable/user/intro.html

Rosa, Javier de la. elguille. [En línea] 30 de Noviembre de 2004. http://www.elguille.info/colabora/puntoNET/versae_MySQLNET.htm

Pozo, Salvador. *mysql.conclase.* [En línea] Enero de 2005. http://mysql.conclase.net/curso/index.php?cap=007

mysql. [En línea] http://www.mysql.com/products/connector/

msdn.microsoft. [En línea] http://msdn.microsoft.com/es-es/library/ms178587(VS.80).aspx.51

msdn.microsoft. [En línea] http://msdn.microsoft.com/es-es/library/system.web.ui.webcontrols.gridview_members(VS.80).aspx

msdn.microsoft. [En línea] http://msdn.microsoft.com/es-es/library/system.web.ui.webcontrols.sqldatasource(VS.80).aspx

msdn.microsoft. [En línea] http://msdn.microsoft.com/es-es/library/87069683(VS.80).aspx

ANEXOS

Anexo 1: Plantilla para la Descripción de los Casos de Uso

Caso de Uso (CU_1)	<<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 Caso de Prueba

Caso de Uso	<nombre del caso de uso>		
Caso de Prueba	**Entrada**	**Condiciones**	**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>	<Condiciones que deben cumplirse mientras se ejecuta el caso de prueba>	<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>

GLOSARIO

API: Interfaz de Programación de Aplicaciones (Application Programming Interface)

Applet: Componente de una aplicación.

ASE: Adaptive Server Enterprise

CASE: Ingeniería de Software Asistida por ordenador (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*)

Casos de prueba: Conjunto de entradas de pruebas con condiciones y resultados esperados.

CIL: Common Intermediate Language

CLI: Lenguaje común de infraestructura (Common Language Infraestructure)

CLR: Lenguaje común de rutina (Common Language Runtime)

DBMS o SGBD: Sistemas de Gestión de Base de Datos (*Database management system*)

DLL: Bibliotecas de Enlace Dinámico (Dynamic Linking Library)

ECMA: Asociación europea de computadoras (European Computer Manufacturers Association)

FDD: Desarrollo guiado por la funcionalidad (Feature driver development).

Framework: Representa una arquitectura del software.

GNU: GNU no es Unix (acrónomo recursivo)

IDE: Entorno de desarrollo integrado (integrated development environment).

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers)

JIT: just-in-time

LDAP: Protocolo ligero de acceso a datos

MSF: Microsoft Solution Framework

MVC: Patrón Modelo Vista Controlador

MVCC: Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control)

MySQL: Servidor de base de datos relacional orientado a objeto.

ODBC: Open Database Connectivity.

PHP: Hypertext Pre-processor.

PostgreSQL: Servidor de base de datos relacional orientado a objeto.

RDBMS: Sistema de gestión de base de datos relacional (Relational Data Base Management System).

Runtime: Serie de DLL's que cualquier programa escrito y compilado en C o C++ necesita para su ejecución.

RUP: Proceso Unificado del Racional (*Rational Unified Process).*

SDK: kit de desarrollo de software (Software Development Kit).

SGBD: Sistema de gestión de base de datos.

SQL: El Lenguaje de consulta estructurado (en inglés Structured Query Language).

SSH: Secure Shell

SSL: Secure Sockets Layer.

Stunnel: Es un programa de computadora libre multi-plataforma.

Tuplas: Grupo de ocurrencias de campos relacionados.

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje unificado de modelado

XP: Programación extrema (Extreme Programming).