Universidad de las Ciencias Informáticas
Facultad 5

Subsistema genérico de gestión y archivo de datos para videojuegos.

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

Autor(es): Aniuska Leyva Cortina.

Dairelys Díaz Benítez.

Tutor: Ing. Abdelaziz de la Horra Díaz.
Co-Tutor: Ing. Orlay García Docunge.

Junio 2010

“En los momentos de crisis sólo la imaginación es más importante que el conocimiento”.

Albert Einstein.

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 _____.

_____	_____
Aniuska Leyva Cortina	Dairelys Díaz Benítez
Autora	Autora
_____	_____
Ing. Abdelaziz de la Horra Díaz	Ing. Orlay García Docunge
Tutor	Co Tutor

DATOS DE CONTACTO

Ing. Abdelaziz de la Horra Díaz.

Graduado de Ingeniería en Ciencias Informáticas. Categoría docente: Adiestrado. Años de experiencia en el tema: 2. Años de graduado: 1. Líder de la línea Base de Datos de Históricos del Polo de Hardware y Automática.

E-Mail: adelahorra@uci.cu

Ing. Orlay García Docunge

Graduado de Ingeniería en Ciencias Informáticas. Categoría docente: Adiestrado. 3 años de experiencia trabajando con gráficos por computadora. Años de graduado: 1. Líder de desarrollo, en el área de Simulación del Polo de Realidad Virtual.

E-Mail: oducunge@uci.cu

DEDICATORIA

A mis padres, a mi novio
A toda mi familia.

Aniuska.

A mi mamá, a mi papá, a mi novio
A toda mi familia.

Dairelys.

AGRADECIMIENTOS GENERALES

La elaboración de este trabajo ha estado guiada por muchas personas que directa o indirectamente han contribuido a su terminación, a ellos y ellas queremos hacerles llegar nuestros más sinceros agradecimientos. Agradecerle a la Revolución y a nuestro comandante en jefe Fidel Castro Ruz, por crear esta universidad, donde tuvimos el privilegio de estudiar y la oportunidad de convertirnos en los profesionales del mañana y hoy poder escribir estos agradecimientos al cumplir nuestro más añorado sueño. De forma especial al Ing. Leonardo Antonio Nieblas Palau, dispuesto en todo momento a atender las dudas y propuestas que iban surgiendo y a nuestro tutor el Ing. Abdelaziz de la Horra Díaz por el constante apoyo y ayuda ante cada dificultad. A todos los miembros del tribunal por el apoyo y la dedicación prestada durante todo este período, dispuestos en todo momento a cooperar en la elaboración y corrección de este trabajo de diploma. A los miembros del grupo 5305 los cuáles son para nosotros como nuestros hermanos, con los que contamos en las buenas y las malas, presentes en todo momento de alegría y dolor, cómplices de tantas aventuras, a todos ellos muchas gracias por haber estado siempre presente. Agradecemos además a todos los profesores que participaran en nuestra formación durante estos 5 años de carrera universitaria, a ustedes muchas gracias por todo el tiempo y esfuerzo empleado. A todas las amistades nuevas y no tan nuevas por su apoyo y cooperación. A cada una de las personas que contribuyeron a este sueño muchas gracias.

AGRADECIEMIENTOS DE ANIUSKA:

Este trabajo de diploma lo dedico especialmente a mis padres: Concepción Cortina Hernández y Eugenio Leyva Salazar, quienes han sido para mí más que unos simples padres, representan la guía de mi vida, el amor y la educación, la sinceridad, la dedicación, la sabiduría y la amistad, por estar siempre a mi lado apoyando cada paso y decisión que he tomado, tendiéndome su mano frente a los tropiezo y disfrutando cada uno de mis logros, por todo esto y sobre todo por el amor que así mi siempre ha profesado, muchas gracias.

A mi familia de forma general agradecerle su ayuda incondicional en este período y su amor de cada día, a mis abuelos por darme su experiencia, a mi tía mima por sé una madre para mí, a mi tío Rey por sus consejos, a mis primos queridos Yordy, Marquito, Yuni, Yudi, Yaimé y Maikel, a todos ustedes gracias por su amor y por compañía. A todos y cada uno de los que conforman mi gran familia gracias.

Agradecerle a mi novio: Leonardo Antonio Nieblas Palau por todo su amor, apoyo, dedicación, comprensión y paciencia durante estos 4 años y medio que llevamos de relación. Por estar a mi lado en los momentos buenos y malos, aun cuando mis padres no podían estar él estaba presente apoyándome y dándome ánimos, para seguir adelante. Muchas gracias por acompañarme y complacerme en cada uno de mis caprichos, por todo esto y por mucho más siempre te amaré.

Un agradecimiento a Abde por ser más que mi tutor mi amigo y compañero por su comprensión y complicidad, por tantas maldades compartidas, por ser esa personita especial que poco a poco se ha ido ganado un lugarcito dentro de mí. A Danay muchas gracias por sus consejos y ayuda y por esa sobrinita linda que tengo Ali que suele ser en ocasiones muy suspicaz en las cosas que quiere.

A todas mis amistades Lisandra, Dairelys, Katia, Linnet, Mario, Duniel y Pedro por su ayuda incondicional, por su amistad sincera, por ser mí abrigo ante las dificultades, por todos esos momentos lindos y no tan lindos que hemos compartido, a todos ustedes muchas gracias por permitirme compartir

estos 5 años con ustedes. Muchas gracias además a todas sus amistades porque juntos también hemos compartidos muchas cosas.

A los muchachos de cuarto: Eniel, Karel, Nestor, Orlando, Sachie y el Toca, muchas gracias por soportarme todo un año con mis pleitos y locuras, por permitirme ser parte de tantas aventuras vividas, por su ayuda ante cada dificultad presentada.

A los muchachos de diseño por toda su cooperación y por escuchar o pretender que los hacían, todas mis disparates, especialmente a Aquino, Derick, Oscar, Yasser, Loison, el gordo e Isidoro.

De forma general agradecerles a todas esas personitas que hayan compartido conmigo aunque sea un pedacito de este tiempo en la UCI, que son muchas y no quisiera que se quedara ninguna a todos ustedes muchas gracias.

AGRADECIMIENTOS DE DAIRELYS:

Agradecerle y dedicarle esta tesis a mi madre Mercedes Benítez Díaz y a mi padre Segundo Díaz Martínez, por demostrar que pueden sacar adelante a sus hijos sin importar los tropiezos que se presenten, por enseñarme que el trabajo siempre tiene sus frutos y recompensas por muy tardío y difícil que sea, porque creyeron en mí y me dieron el apoyo necesario para crecer y ser lo que ahora soy, agradecerle también a mi hermano, por estar conmigo durante la etapa más importante mi vida, por crecer juntos, y por estar siempre apoyándome.

A mis amigas: Yailén, Maidelys y Yanai por estar presente, por compartir conmigo no solo en esta etapa de la universidad, sino desde niñas. Por todos esos momentos que vivimos juntas muchas gracias.

A Susej, Yurian, Lisandra, Rosalina, Aniuska, Leonardo, Duniel y Abedelaziz, por el apoyo, la ayuda incondicional, porque muchas veces nos desvelamos juntos terminados proyectos de última hora en fin por todos los momentos vividos por buenos o malos que estos fueron.

A mí tu tutor: Abdelaziz de la Horra Díaz, una de las personas que más admiro por su inteligencia y sus conocimientos, por contribuir en la corrección de este trabajo y por su paciencia en la revisión del mismo.

A mi novio Yonier Abreu García: Por estar conmigo en esta etapa de mi vida, la cual considero que es la mejor que estoy viviendo, por amarme y enseñarme muchas cosas, por toda la paciencia que tiene y porque es un ser humano excepcional, gracias por la ayuda, por ser más que mi pareja, te agradezco todo el apoyo, tu cariño y estimulo que me has dado. Te amo.

A toda mi familia: Mis tíos Milagro y Gustavo, mis abuelos Ocilia, Rosa y Trujillo, mis primos, en general a toda mi familia por su amor, compresión y apoyo incondicional que me han otorgado durante este tiempo y que ha sido imprescindible en mi formación como profesional.

Quiero agradecer muy especialmente a mi compañera de tesis Aniuska Leyva Cortina y a su novio Leonardo Antonio Nieblas porque sin ellos esta tesis no se hubiese realizado, por su apoyo incondicional, porque han estado conmigo en las buenas y malas, porque nunca me han dejado caer, gracias por haberme brindado todo su apoyo, colaboración, ánimo y sobre todo cariño y amistad.

En general quisiera agradecer a todas y cada una de las personas que hayan vivido conmigo la realización de esta tesis, con sus altos y bajos y que no necesite nombrar porque tanto ellas como yo sabemos que desde los más profundo de mi corazón les agradezco el haberme brindado todo el apoyo.

RESUMEN

Con el transcurso del tiempo las computadoras se han convertido en una herramienta fundamental para el desarrollo de disímiles campos de la sociedad. Actualmente las aplicaciones de Realidad Virtual tienen una gran demanda entre los cibernautas y estas a su vez incorporan sistemas de bases de datos para el almacenamiento de la información persistente que necesitan.

La comunicación entre estas aplicaciones y los sistemas de bases de datos por lo general no se realiza de forma directa sino a través de una aplicación intermediaria, encargada de manipular los datos y hacerlos entendibles para ambos participantes. Es por ello que este trabajo aborda el diseño y la implementación de una aplicación gráfica que permite la configuración de diagramas de base de datos y la generación de una biblioteca que se encarga de manipular toda la información contenida en la misma, con el objetivo de establecer la comunicación aplicación-base de dato.

Para darle cumplimiento a los objetivos del trabajo se realiza un estudio de diversos módulos de gestión y archivo de datos, además se analizaron varios Sistemas Gestores de Base de Datos (SGBD), principalmente aquellos que se utilizan en los videojuegos. También fue necesario realizar una investigación de las principales bibliotecas para la construcción de interfaces gráficas de usuario, así como se investigó sobre las características fundamentales del formato XML, y las ventajas que brinda para el almacenamiento de información.

SUMMARY

Over time, computers have become an essential tool for the development of many fields of the society. Currently, the Virtual Reality applications are in high demand among Internet users and look to incorporate these database systems for storing persistent information they need

Communication between these applications and database systems usually are not done directly but through an intermediary application, responsible for handling data and making them understandable to both participants. That is why this approach to the design and implementation of a graphical application that allows you to set database diagrams, generation of a library that is responsible for handling all information contained in it, with the aim of establishing application-based communication of data.

To give effect to the objectives of the work is a study of several models to manage and archive data, and analyzed various management system database (DBMS), especially those used in video games. It was also necessary to conduct an investigation of the main libraries for building graphical user interfaces, and was investigated on the key features of the XML format, and the benefits provided for storage of information.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Desde que el hombre comenzó a pensar y a razonar con un pensamiento más avanzado que el del resto de los animales que le rodeaban, tubo la necesidad de guardar ciertas informaciones que a su modo de ver eran importantes y que su memoria no era capaz de retener por mucho tiempo.

De ahí que como primer medio de almacenamiento que se empleó fue el tallado en piedra, luego un poco más adelante surge el papel y la tinta y con ellos una nueva forma de archivar información y que hasta hoy se emplean en muchas empresas. Pero el desarrollo no se detiene y a lo largo de la historia se ha buscado la forma de encontrar el sistema más pequeño físicamente, que permita almacenar grandes volúmenes de datos y que posea la capacidad de tratarlos rápidamente, siendo ya posible almacenar miles de carpetas con información en papel de forma digital, ya sea en CD, DVD, Disco duro, Memorias USB, etcétera. [1]

Más recientemente los sistemas se han extendido hacia la producción y gestión de la información, y se han convertido en un recurso vital para las empresas, en este ámbito se destacan las *bases de datos (BD)*.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, Estados Unidos. Para el desarrollo de la actividad profesional tienen gran importancia, destacándose dos colectivos que a menudo son beneficiados por las mismas: Por una parte, los investigadores y, por otra, los profesionales. [1]

Para algunos autores el término base de datos podría definirse como:

- Cualquier recopilación organizada de información sobre la que haya habido análisis documental y que disponga de un sistema de búsqueda específica. [1]
- Un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos. [3]
- Conjunto exhaustivo no redundante de datos estructurados y organizados independientemente de su organización e implementación en máquinas accesibles en tiempo real y compartible con usuarios concurrentes que tienen necesidad de información diferente y no predecible en el tiempo. [4]

De acuerdo con el estudio de las definiciones anteriores se puede decir que las bases de datos son un conjunto de datos pertenecientes a un mismo contexto, interrelacionados entre sí y almacenados sistemáticamente para su posterior uso.

Las BD se han constituido como una de las herramientas más ampliamente difundidas en la actual sociedad de la información, utilizadas como fuentes secundarias en la recuperación y almacenamiento de información en todos los campos a nivel científico, social, económico, político y cultural.

En los últimos 10 años la Realidad Virtual (RV) ha sido el centro de atención para numerosas investigaciones y desarrollo. La RV puede estar inmersa en disímiles aplicaciones, ya sean de interés científico-investigativo, o en áreas de la recreación, donde su principal rol lo desempeña en los videojuegos.

Casi cualquier aplicación de RV requiere de almacenamiento persistente de datos, por ejemplo, en los juegos se necesita guardar los datos de los jugadores y sus *récords*, lo cuál puede ser implementado sencillamente con algún fichero, pero a medida que la aplicación es más seria, por ejemplo, un simulador que deba emitir una evaluación de los estudiantes en entrenamiento, se necesitan Sistemas de Bases de Datos (SBD) para acumular mayor información y acceder a ella con mayor velocidad.

Tanto en juegos como en simuladores se requiere de una funcionalidad importante conocida como “replay”, que consiste en permitir grabar y repetir escenas, para un juego de béisbol, por ejemplo, resulta muy atractivo repetir una jugada, y para simuladores evaluadores, es importante que el estudiante pueda ver nuevamente su desempeño.

Esta funcionalidad requiere del almacenamiento de posiciones de objetos móviles, estados (características físicas como velocidad en cada instante), propiedades de visualización (transparencia, colores), elementos de inteligencia artificial (decisiones tomadas por los objetos), etc. de forma tal que se repita la escena con la mayor exactitud posible. Tanta información resulta engorrosa y lenta manipularla en un fichero, por lo que también requiere del uso de bases de datos.

El éxito de los videojuegos está creciendo día a día y poseen un futuro prometedor para beneficio de la industria de los videojuegos. Los usuarios son cada vez más exigentes y no buscan sólo un juego entretenido sino que también demandan una excelente calidad visual. Por tal motivo es que aparejado a los videojuegos vienen las BD que sustentan a los mismos, ya que ellas proporcionan un enfoque de mayor realidad, proporcionándole al juego fuertes herramientas para su desempeño como pueden ser las salvas del mismo, la repetición de escenas y el almacenamiento de los récords de los jugadores.

En el Departamento de Realidad Virtual de la facultad 5 se desarrollan aplicaciones gráficas, muy vinculadas a los videojuegos. El personal que trabaja en la construcción de estas aplicaciones actualmente se encuentra más especializado en el trabajo con matemática discreta, gráficos por computadoras y geometría computacional que con el trabajo con BD.

Por tal motivo la producción se ve retrasada cuando en las aplicaciones que se desarrollan, se necesita almacenar los datos con los que se están trabajando, los cuales pueden ser por ejemplo el ranking de cada jugador, los perfiles de estos, los datos de configuración de las aplicaciones, entre otros.

De ahí que se plantee la siguiente **situación problémica:** en la Universidad de las Ciencias Informáticas específicamente en la facultad 5, en el Departamento de Realidad Virtual, no existe una plataforma común de BD que se pueda acoplar a las aplicaciones que se desarrollan por los diferentes proyectos sin la necesidad de desarrollar una capa de acceso a dato específica para cada uno de ellos. Provocando con ello que cada proyecto defina su propia forma de trabajar en este sentido y muchas veces además de repetir esfuerzos desarrollando una misma funcionalidad, no se logra su potabilidad a otros sistemas y plataformas. Derivándose como **problema científico**: ¿Cómo almacenar y manejar información en los videojuegos utilizando BD?

El **objeto de investigación** que enmarca el trabajo comprenderá a: los Módulos de gestión y archivo de datos. Y como **campo de acción**: La gestión y archivo de datos para videojuegos.

El **objetivo general** que se propone para esta investigación es: Desarrollar un subsistema de gestión y archivo de datos con la capacidad de ajustarse a las necesidades de los videojuegos desarrollados en el Departamento de Realidad Virtual.

Objetivos Específicos:

- Proporcionar una interfaz gráfica de usuario (GUI de sus siglas en inglés), capaz de ajustarse a las especificidades requeridas del software.
- Proporcionar el modo de establecer la configuración del subsistema de gestión y archivo de datos.
- Proporcionar los mecanismos de almacenamiento y acceso a los datos persistentes.
- Implementar la funcionalidad de reciclado de los datos considerados antiguos o en desuso, así como un mecanismo para impedir el desbordamiento de memoria.
- Proporcionar las interfaces necesarias para establecer el intercambio de información con el videojuego.

Para darle cumplimiento a los objetivos se plantean las siguientes **Tareas de Investigación:**

- Desarrollo del diseño teórico-metodológico.
- Estudio de las bibliotecas para el desarrollo de GUI, con el objetivo de determinar cuál de ellas será utilizada en la implementación de las interfaces gráficas de usuario de la aplicación final.
- Estudio de los sistemas gestores de bases de datos (SGBD), para determinar cuál sería el adecuado en la solución que se brinda.
- Estudio de los módulos de archivo de datos, para identificar las principales funcionalidades que estos deben ofrecer.
- Estudio del formato XML para el almacenamiento temporal de la información.
- Caracterización de las herramientas a utilizar en el desarrollo de la propuesta de solución.
- Elaboración de la arquitectura que presentará el sistema a implementar.
- Diseño de las interfaces gráficas de usuario.
- Implementación de las interfaces gráficas de usuario.
- Diseño de las interfaces de recepción de variables, configuración y la capa de acceso a datos.
- Implementación de las interfaces de recepción de variables, configuración y la capa de acceso a datos.
- Integración de los subsistemas que componen el módulo.

Durante el desarrollo de este trabajo de diploma se emplearán diferentes **métodos científicos** como son:

- *Método Analítico-sintético,* el mismo se utilizará durante el estudio de las diversas bibliografías con el objetivo de caracterizar los diversos SGBD así como identificar aquellos que por sus características se adecuen a la solución. Este método será utilizado además para sintetizar el estudio realizado a los diferentes módulos de gestión y archivos de datos.
- *Método Análisis histórico-lógico*, con el empleo de este método se analizarán los aportes científicos y funcionales que se han incluido en las diferentes versiones de los gestores de bases de datos así como en los módulos de gestión y archivo de datos creados, en el desarrollo de los videojuegos.

La estructura del presente trabajo de diploma estará organizada por capítulos distribuidos de la siguiente manera:

Capítulo 1: Fundamentación teórica, incluyendo el estudio del arte, abordando temas referentes a las tendencias actuales de los sistemas de gestión y archivo de datos.

Capítulo 2: Soluciones Técnicas, dentro de este capítulo se explican las características del sistema. Se realiza una descripción del flujo actual de los procesos involucrados en el campo de acción, haciéndose un análisis crítico, de cómo se desarrollan estos procesos actualmente. Además se brinda una propuesta de cómo debe funcionar el sistema.

Capítulo 3: Análisis y Diseño, en este capítulo se muestran los diagramas de clases necesarios para modelar el comportamiento del sistema, se describen las clases propuestas, así como, la arquitectura de la información que presentará la propuesta de solución.

Capítulo 4: Implementación del Software, en el mismo se describe el estándar de codificación utilizado en la realización de la aplicación, además se detalla a través de un diagrama de componente como está distribuida la estructura del sistema. También se esquematizan las condiciones físicas necesarias para el despliegue del software.

Posibles resultados: Como resultado de este trabajo de diploma se pretende obtener un subsistema genérico de gestión y archivo de datos, que pueda ser empleado en el desarrollo de diversos videojuegos en el Departamento de Realidad Virtual, evitando la necesidad de implementar una capa de acceso a datos específica para cada uno y de ellos disminuyendo así el costo en tiempo y recursos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En la actualidad es muy común convivir con o entre ordenadores, el avance de la tecnología ha conllevado a ser dependientes de ella y para muchos ya es impensable vivir sin un teléfono móvil, o sin él interactuar entre las tantas redes sociales existentes y más complejo aún es el hecho de que para la mayoría de los cibernautas es inevitable el permanecer varias horas conectados a los videojuegos, que te llevan cada vez más a un mundo virtual.

Esta nueva forma de atarnos a la tecnología está dada por el realismo presente en los videojuegos y este a su vez es producto de la Realidad Virtual (RV) la cuál desde el punto de vista de la tecnología, puede definirse como:

- La suma de los sistemas de hardware y de software que aspiran a construir una ilusión sensorial de estar presente en otro ambiente, en otra realidad. [2]
- Para otros es un sistema tecnológico, basado en el empleo de ordenadores y otros dispositivos, cuyo fin es producir una apariencia de realidad que permita al usuario tener la sensación de estar presente en ella. [5]

A partir de las definiciones anteriores se llega a la conclusión de que la RV es un conjunto de dispositivos de software y hardware que brindan la posibilidad de adentrarse en un mundo totalmente virtual pero que trata de reflejar hechos y sucesos de la realidad.

El mundo de los videojuegos es una realidad cambiante y en continua evolución. En la actualidad existe una gran tendencia a la utilización de las Bases de Datos (BD), en los mismos. El empleo de las BD viene dado por el almacenamiento de informaciones como pueden ser: el estado del juego, información del perfil de los jugadores, o el ranking de ellos para posteriormente poder manejarla con mayor facilidad y rapidez. [7]

La comunicación entre el juego y la BD no se realiza de forma directa, sino que se utiliza una especie de intermediario que viene a ser en este caso los Sistemas Gestores de Bases de Datos (SGBD).

1.1 Sistemas Gestores de Bases de datos (SGBD).

1.1.1 ¿Qué es un gestor de base de datos?

Los *sistemas de gestión de bases de datos* o SGBD (en inglés Database Management System, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. [4]

De forma general los SGBD son los encargados de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización. [7]

1.1.2 ¿Cuáles son los principales objetivos que debe cumplir un SGBD?

Existen diversos objetivos que deben cumplir los SGBD:

- **Abstracción de la información**. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Dentro de este contexto da lo mismo si una BD ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción. [4]
- **Independencia**. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una BD sin tener que realizar cambios en las aplicaciones que se sirven de ella. [4]
- **Consistencia**. En aquellos casos en los que no se ha logrado eliminar la redundancia en las BD, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la BD representa una realidad determinada que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones. [4]
- **Seguridad**. La información almacenada en una BD puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos. [4]
- **Manejo de transacciones**. Una transacción es un programa que se ejecuta como una sola operación. Esto quiere decir que luego de una ejecución en la que se produce una falla es el

mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos. [4]

- **Tiempo de respuesta**. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en ofrecer la información solicitada y en almacenar los cambios realizados. [4]

1.1.3 Ventajas de los SGBD.

Los SGBD poseen indiscutibles ventajas entre las que se encuentran:

- Facilidades para la manipulación de grandes volúmenes de datos. Entre éstas:
  - Simplifican la programación de equipos de consistencia.
  - Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores correctamente, etc.
  - Organizan los datos con un impacto mínimo en el código de los programas.
  - Bajan drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.
- Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos. [7]

1.1.4 Desventajas de los SGBD.

Los SGBD no están exentos de inconvenientes entre los que se pueden encontrar:

- El incremento de los costos de operación en una empresa. En ocasiones, es necesario disponer de una o más personas que administren las bases de datos, en la misma forma en que suele ser necesario en instalaciones de cierto porte disponer de una o más personas que administren los sistemas operativos. Esto puede llegar a provocar un aumento en los costes de la empresa por conceptos de salario al personal. Sin embargo, hay que balancear este aspecto con la calidad y confiabilidad del sistema que se obtiene. [7]
- Si se tienen muy pocos datos que son usados por un único usuario por vez y no hay que realizar consultas complejas sobre los datos, entonces es posible que sea mejor usar una hoja de cálculo. [7]

- Complejidad: el software es muy complejo y las personas que vayan a usarlo deben tener conocimiento de las funcionalidades del mismo para poder aprovecharlo al máximo. [7]
- Tamaño: la complejidad y la gran cantidad de funciones que tienen hacen que sea un software de gran tamaño, y en ocasiones requieren de gran cantidad de memoria para poder correr. [7]
- Coste del hardware adicional: los requisitos de hardware para correr un SGBD por lo general son relativamente altos, por lo que estos equipos pueden llegar a costar gran cantidad de dinero. [7]

1.2 Principales SGBD.

1.2.1 Berkeley DB (DB)

Berkeley DB (DB) es un gestor de base de datos integrado (embebido), de propósito general que es capaz de proporcionar una gran cantidad de servicios de gestión de datos. Ha sido diseñado desde cero para aplicaciones de alto rendimiento que requieren un proceso a prueba de balas en lo que respecta a la gestión de datos de misión crítica. DB se puede ajustar correctamente desde la gestión de unos pocos bytes a terabytes de datos. En su mayor parte, DB sólo está limitado por los recursos disponibles de su sistema físico. [10]

DB se puede utilizar a través de una serie de APIs de programación que le dan la capacidad de leer y escribir datos, administrar la base de datos(s) y realizar otras actividades más avanzadas como la gestión de transacciones. [10]

Debido a que DB es un motor de BD integrada, es extremadamente rápido. Permite ser compilado y vinculado en una aplicación de la misma manera como se haría con cualquier biblioteca. Esto significa que DB se ejecuta en el mismo espacio del proceso que tiene su aplicación, lo que le permite evitar el alto costo de las comunicaciones entre procesos realizados por los servidores independientes BD. [10]

Para mejorar aún más el rendimiento, DB ofrece una memoria caché de memoria diseñada para proporcionar un acceso rápido a sus datos más utilizados. Una vez configurado, el uso de caché es transparente. Se requiere muy poca atención por parte de los desarrolladores de aplicaciones. [10]

Más allá de la velocidad pura, DB es también muy configurable. Ofrece varias maneras diferentes de organizar los datos en la BD. Conocido como métodos de acceso (Btree, Hash, Queue o Recno), cada

uno de estos mecanismos de organización de datos proporciona características diferentes que son apropiados para diferentes perfiles de gestión de datos. [10]

Para mejorar aún más su capacidad de configuración, DB ofrece diversos subsistemas, cada uno de los cuales se puede utilizar para ampliar sus capacidades. Por ejemplo, muchas aplicaciones requieren protección de escritura para sus datos para garantizar que nunca queden en un estado incoherente por algún motivo, como pueden ser, bugs presentes en el software o fallas en el hardware. Para este tipo de aplicaciones, un subsistema de transacción puede ser activado y utilizado para brindar protección transaccional al escribir en la BD. [10]

La lista de sistemas operativos en los que DB está disponible es extensa, basta con decir que está disponible en todos los principales sistemas operativos comerciales, así como en muchas plataformas embebidas. [10]

DB está disponible en una gran variedad de lenguajes de programación y está soportado oficialmente por C, C++ y Java, pero la biblioteca también está disponible en muchos otros lenguajes, especialmente en los lenguajes de scripting como Perl y Python. [10]

Es importante mencionar que DB no es una BD relacional (aunque se puede usar para construir una). Fuera de su entorno, DB no ofrece características de nivel superior tales como triggers, o un lenguaje de alto nivel de consulta como SQL. En su lugar, DB provee a las APIs los mínimos requerimientos para almacenar y recuperar datos de la manera más eficiente posible. [10]

Conceptualmente, las bases de datos DB contienen registros. Lógicamente, cada registro representa una única entrada en la BD. Cada registro contiene dos tipos de información: una clave y los datos. [10]

Debido a la vinculación clave/datos utilizados en las bases de datos, en DB a veces son considerados como una tabla de dos columnas. Sin embargo, los datos (y a veces las claves, dependiendo del método de acceso) puede contener datos arbitrariamente complejos. Con frecuencia, las estructuras de C y otros mecanismos, se almacenan en el registro. Esto conforma efectivamente las 2-columnas en una tabla con n columnas, donde n-1 de las columnas son proporcionadas por los campos de la estructura utilizada. [10]

Se debe tener en cuenta que una base de datos DB es muy parecido a una tabla en una BD relacional, donde la mayoría de las aplicaciones DB utilizan más de una BD (al igual que en la mayoría de de BD relacionales se utilizan más de una tabla). [10]

Con frecuencia las solicitudes de DB están diseñadas de modo que una sola BD almacena un tipo específico de datos (como en un sistema de BD relacional, una sola tabla contiene entradas con un conjunto específico de campos). Debido a que en la mayoría de las aplicaciones es necesario gestionar múltiples tipos de datos, una aplicación DB con frecuencia utiliza varias BD. [10]

Las aplicaciones DB logran eficiencia en el uso de varias BD utilizando un mecanismo opcional llamado *environment.* Se puede interactuar en la mayoría de las API DB usando estructuras especiales que contienen punteros a funciones. Estas devoluciones de llamada (callbacks) se definen como métodos porque tienen un aspecto muy similar a un método en una clase C++. La variable que se utiliza para acceder a estos métodos se refiere a menudo como un *handle (*manejador). Por ejemplo, al utilizar una base de datos se obtendrá un *handle* para esa base de datos. [10]

La recuperación de un registro en una base de datos se denomina a veces obtener el registro y se utiliza la opción get () para realizar esta función. Del mismo modo, el almacenamiento de registros en la BD se refiere en ocasiones a poner en disco por lo cual se utiliza la opción de put () para hacerlo. [10]

Cuando se almacena, o se hace put (), a un registro utilizando el handle, el registro se almacena de acuerdo con algún criterio de ordenación que esté siendo usado por la BD. Dicho ordenamiento se realiza principalmente sobre la base de la clave, pero a veces los datos se consideran también. Si se coloca un registro con una clave que ya existe en la base de datos, el registro existente se reemplaza con los nuevos datos. Sin embargo, si la base de datos soporta registros duplicados (es decir, los registros con claves iguales, pero con datos diferentes), entonces dicho registro se almacena como un registro duplicado y los registros existentes no se sobrescriben. [10]

Si la BD soporta registros duplicados, entonces se puede utilizar el handle de la BD sólo para recuperar el primer registro en el conjunto de registros duplicados. [10]

Además de utilizar un *handle* en la BD, también se pueden leer y escribir datos mediante un mecanismo especial llamado *cursor*. Los cursores son esencialmente iteradores que se pueden utilizar

para moverse sobre los registros de la BD. Se pueden utilizar los cursores para iterar desde el primer registro hasta el último, y desde el último al primero. También se pueden utilizar los cursores en la búsqueda de un registro. En el caso de que soporte registros duplicados, los cursores son la única forma de acceder al conjunto de dichos registros. [10]

DB proporciona un tipo especial de BD llamada BD secundaria. Las BD secundarias sirven como índice a las BD normales (llamadas BD principales para distinguirlas de las secundarias). BD secundarias son interesantes debido a que los registros DB pueden contener tipos de datos complejos, pero al buscar un determinado registro se realiza sólo sobre la base de la clave. Si se quiere buscar un registro sobre la base de alguna otra pieza de información que no es la clave, se puede lograr esto a través de la utilización de las BD secundarias. [10]

1.2.2 PostgreSQL

PostgreSQL es un sistema de gestión de BD objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. Es una derivación libre (Open Source) de este proyecto, y utiliza el lenguaje SQL92/SQL99. [8]

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de BD puramente orientado a objetos. [8]

Entre sus principales características se encuentran:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP…), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.

- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este SGBD se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos. [8]

PostgreSQL es un magnífico GBD, capaz de competir con muchos gestores comerciales, aunque carezca de alguna característica casi imprescindible. Como puede ser un conjunto de herramientas que permitan una fácil gestión de los usuarios y de las BD que contenga el sistema. Por otro lado, la velocidad de respuesta que ofrece este gestor con BD relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar BD realmente grandes, cosa que resulta loable. [9]

Posee prácticamente todo lo que tienen los gestores comerciales, haciendo de él una muy buena alternativa GPL. A pesar de ello, el primer encuentro con este gestor es un poco "duro", ya que la sintaxis de algunos de sus comandos no es nada intuitiva. También resulta engorroso las pequeñas variaciones que presenta este gestor en algunos de los tipos de datos que maneja, siendo el problema más comentado el referente al tipo "serial". [9]

Una vez adaptados a su sintaxis y mostrando atención en estos pequeños detalles (que por otro lado están totalmente documentados), PostgreSQL es un gestor magnífico, que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios web que posean alrededor de 500.000 peticiones por día. [9]

1.2.3 SQL Server

Microsoft SQL Server es un SGBD basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. [9]

Constituye la alternativa de Microsoft a otros potentes SGBD como son *Oracle*, *Sybase ASE*, *PostgreSQL*, *Interbase*, *Firebird* o *MySQL*. [9]

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de BD pero orientado a proyectos más pequeños, que en sus versiones 2005 y 2008 pasan a ser el SQL Express Edition, que se distribuye en forma gratuita. [9]

Es común desarrollar completos proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project). De esta forma, se completa la BD (Microsoft SQL Server), con el entorno de desarrollo (VBA Access), a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows. [9]

En el manejo de SQL mediante líneas de comando se utiliza el SQLCMD. Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos Windows. [9]

Entre sus características se tiene:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además, permite administrar información de otros servidores de datos. [9]

La nueva BD contiene mayor seguridad, integración con PowerShell, remueve la consola configuración del área expuesta (consola para configurar seguridad), tiene correctores de sintaxis del lenguaje Transact-SQL e inteligencia (una característica del Visual Studio que permite a la BD sugerir objetos existentes mientras uno escribe la mitad de la palabra). Así mismo incluye nuevos tipos de datos y funciones. [9]

1.2.4 MySQL

MySQL es un SGDB relacional, licenciado bajo la GPL de la GNU. Su diseño multi-hilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca. [13]

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL. [13]

Este GBD es, probablemente, uno de los más usados en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de bibliotecas y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. [13]

Las principales características son las siguientes:

- Aprovecha la potencia de sistema multiprocesador, gracias a su implementación multi-hilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc.).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos. [13]

MySQL surgió cómo una necesidad de un grupo de personas sobre un GDB rápido, por lo que sus desarrolladores fueron implementando únicamente lo que precisaban, intentando hacerlo funcionar de forma óptima. Es por ello que, aunque MySQL se incluye en el grupo de SBD relacionales, carece de algunas de sus principales características:

- *Sub-consultas*: tal vez ésta sea una de las características que más se echan en falta, aunque gran parte de las veces que se necesitan, es posible reescribirlas de manera que no sean necesarias.
- *SELECT INTO TABLE:* Esta característica propia de Oracle, todavía no está implementada.

- *Triggers y Procedimientos:* Se tiene pensado incluir el uso de procedimientos almacenados en la base de datos, pero no el de triggers, ya que los triggers reducen de forma significativa el rendimiento de la BD, incluso en aquellas consultas que no los activan.
- *Transacciones:* a partir de las últimas versiones ya hay soporte para transacciones, aunque no por defecto (se ha de activar un modo especial).
- *Integridad referencial:* aunque sí que admite la declaración de claves ajenas en la creación de tablas, internamente no las trata de forma diferente del resto de los campos. [13]

Los desarrolladores comentan en la documentación que todas estas carencias no les resultaba un problema, ya que era lo que ellos necesitaban. De hecho, MySQL fue diseñada con estas características, debido a que lo que buscaban era un GBD con una gran rapidez de respuesta. Pero ha sido con la distribución de MySQL por Internet, cuando más y más gente les está pidiendo estas funcionalidades, por lo que serán incluidas en futuras versiones del gestor. [13]

1.2.5 SQLite

Es un SGBD datos relacional compatible con ACID, y que está contenida en una relativamente pequeña (~500kb) biblioteca en C. Es un proyecto de dominio público creado por Dr. Richard Hipp. [11]

A diferencia de los SGBD cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. [11]

Esto reduce la latencia en el acceso a la BD, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la BD (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de BD al principio de cada transacción. [11]

En su versión 3, SQLite permite BD de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB. El autor de SQLite ofrece formación, contratos de soporte técnico y características adicionales como compresión y cifrado. [11]

Implementa un completo motor de BD multiplataforma que no precisa configuración. Se distribuye bajo licencia de dominio público. Es muy rápido y la ventaja fundamental es que permite utilizar un amplio subconjunto del lenguaje estándar SQL. SQLite destaca también por su versatilidad. El motor de PHP 5 incluye soporte interno para SQLite. [11]

Combina el motor y el interfaz de la BD en una única biblioteca, y almacena los datos en un único archivo de texto plano. Esto hace que cada usuario pueda crear tantas BD como desee sin la necesidad de la intervención de un administrador de BD que gestione los espacios de trabajo, usuarios y permisos de acceso. El hecho de almacenar toda la BD en un único archivo, facilita la portabilidad de los datos, y solamente tiene la restricción del espacio de disco asignado al usuario en el servidor. [11]

Su potencia se basa fundamentalmente en la simplicidad, lo que hace que no sea una buena solución en entornos de tráfico muy elevado y/o alto acceso concurrente a datos. SQLite encapsula toda la BD en un único fichero. [11]

Se puede utilizar SQLite de dos formas:

- *Como gestor de base de datos local en un PC*. De esta forma, se pueden gestionar bases de datos con SQLite igual que si se estuviese trabajando con un sistema gestor de base de datos como MySQL sin necesidad de instalar nada, ya que SQLite se compone de un único archivo ejecutable.
- *Como una extensión más de PHP*, utilizando las funcionalidades de SQLite configuradas, o bien como módulo de PHP, o como biblioteca; sin necesidad de tener instalado o conectar con un servidor de base de datos. Ofrece una rápida interfaz de base de datos almacenado en archivo de texto plano. [11]

SQLite dispone de una completa interfaz orientada a objetos, con distintas funciones que facilitan la manipulación de datos. Funciones muy similares a las que se pueden manejar con MySQL. [11]

Lita es un administrador de bases de datos *SQLite*, que está desarrollado con *Adobe Air* y que básicamente permite:

- Abrir, crear, compactar bases de datos.
- Crear, renombra, eliminar tablas vacías.

- Crear, renombrar y eliminar columnas.
- Crear, modificar y eliminar registros.
- Encriptar o reencriptar bases de datos:
- Correr sentencias SQL personalizadas.
- Crear y eliminar índices. [11]

Como está desarrollado con *Adobe Air* es multiplataforma, su interfaz es muy moderna y funcional. SQLite usa un sistema de tipos inusual. En lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede insertar un *string* en una columna de tipo entero (a pesar de que SQLite tratará en primera instancia de convertir la cadena en un entero). Algunos usuarios consideran esto como una innovación que hace que la BD sea mucho más útil, sobre todo al ser utilizada desde un lenguaje de scripting de tipos dinámicos. Otros usuarios lo ven como un gran inconveniente, ya que la técnica no es portable a otras BD SQL. SQLite no trataba de transformar los datos al tipo de la columna hasta la versión 3. [11]

Varios procesos o hilos pueden acceder a la misma BD sin problemas. Varios accesos de lectura pueden ser servidos en paralelo. Un acceso de escritura sólo puede ser servido si no se está sirviendo ningún otro acceso concurrentemente. En caso contrario, el acceso de escritura falla devolviendo un código de error (o puede automáticamente reintentarse hasta que expira un timeout configurable). Esta situación de acceso concurrente podría cambiar cuando se está trabajando con tablas temporales. Sin embargo, podría producirse un deadlock debido al proceso multi-hilo. [11]

1.2.5.1 Lenguajes de programación

- La biblioteca puede ser usada desde programas en C/C++, aunque enlaces para Tcl y muchos otros lenguajes de scripts están disponibles.
- SQLite se encuentra embebido en el REALbasic framework, haciendo posible que aplicaciones desarrolladas en REALbasic para Windows, Linux o Mac OS X usen la base de datos SQLite.
- Existe un módulo DBI/DBD para Perl disponible en CPAN, *DBD::SQLite*, no es una interfaz para SQLite, sino que incluye el motor completo de SQLite en sí mismo por lo cuál no necesita ningún software adicional.
- Hay también un módulo para Python llamado PySQLite.
- Hay otro módulo para Visual Basic 6 llamado VBSqlite.

- Desde Delphi se puede usar SQLite a través de los componentes libres ZeosLib.
- PHP incluye SQLite, desde la versión 5. SQLite también funciona con PHP 4 pero no viene incluido en él.
- Desde Lazarus 0.9.8 y Free Pascal 2.0.0, SQLite está disponible para programadores de Pascal.
- Mac OS X v10.4 incluye SQLite, y es una de las opciones en la Core Data API de Apple. AppleScript puede abrir, crear, y manipular base de datos SQLite por medio de la aplicación de ayuda "Database Events" de Mac OS X 10.4.
- BlitzMAX posee un MOD que permite trabajar con bases de datos SQLite.
- El componente de base de datos (gb.db) de Gambas soporta SQLite en sus versiones 1, 2 y 3. [11]

1.2.5.2 Productos que utilizan SQLite

- Desde hace tiempo Mozilla Firefox usa SQLite para almacenar, por ejemplo, las cookies.
- SQLite es usado por el entorno de base de datos Kexi como un motor de base de datos interno por defecto.
- SQLite se ha usado para guardar el índice para un set de DVD conteniendo todos los números publicados de la revista *The New Yorker*.
- Yum, la herramienta de gestión de paquetes de Fedora Core, ha cambiado a SQLite y pysqlite para el almacenamiento de datos y parseo de XML desde el núcleo de Fedora 4. De acuerdo con los comentarios de los usuarios, el incremento de funcionamiento es impresionante, así como la reducción en el consumo de memoria.
- Los desarrolladores de OpenOffice.org han considerado embeber SQLite en el modelo de base de datos de Base, pero esto depende en gran manera del progreso de sqlite-sdbc-driver, que está todavía en estado de alpha. Actualmente han decidido usar HSQLDB, pero la opción está todavía abierta siempre y cuando el driver sqlite-sdbc acabe siendo maduro y de confianza. [11]

1.2.6 Oracle

Oracle es un SGBD relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporation. Surge a finales de los 70 bajo el nombre

de a partir de un estudio sobre SGBD de George Koch. Se considera a Oracle como uno de los más completos. [4]

Destacándose en:

- Soporte de transacciones,
- Estabilidad,
- Escalabilidad y
- Soporte multiplataforma. [4]

Oracle es básicamente una herramienta cliente/servidor para la gestión de BD. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de páginas web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras BD, por ejemplo, Access, MySQL, SQL Server, etc. [4]

Para desarrollar en Oracle se utiliza PL/SQL un lenguaje de 5ª generación, potente para tratar y gestionar la BD, también por norma general se suele utilizar SQL al crear un formulario. Es posible lógicamente atacar a la BD a través del SQL plus, incorporado en el paquete de programas Oracle para poder realizar consultas, utilizando el lenguaje SQL. [4]

El Developer es una herramienta que permite crear formularios locales, es decir, mediante esta herramienta se pueden crear formularios, compilarlos y ejecutarlos, pero si se quiere que los otros trabajen sobre este formulario se debe copiar regularmente en una carpeta compartida para todos, de modo que, cuando quieran realizar un cambio, deberán copiarlo de dicha carpeta y luego volverlo a subir a la carpeta. [4]

Como desventajas de este sistema se observa que es bastante engorroso y poco fiable pues es bastante normal que las versiones se pierdan y se machaquen con frecuencia. Pero el principal y más notable problema es la falta de un entorno visual para diseñar el formulario, es decir, aparece una estructura como de árbol en la cuál se insertan los formularios, a la vez dentro de éste son insertados bloques o módulos que son las estructuras que contendrán los elementos del formulario, que pueden estar basados en tablas o no. [4]

La principal ventaja de esta herramienta es que es bastante intuitiva y dispone de un modo que permite componer el formulario, tal y como lo hace Visual Basic o en Visual C. [4]

1.3 SGBD utilizados en videojuegos.

Entre los juegos más difundidos en el mundo se encuentran el World of Warcraft, Start Craft, Call of Dutty, FIFA, Quake, Need for Speed y Battle Field Vietnam entre muchos otros, que hoy día son los que tienen la gran misión de entretener a niños y jóvenes cada vez más hiperactivos.

La gran mayoría de estos juegos utilizan como sistema de almacenamiento ficheros, ya que la información que se almacena de los mismos suele ser sencilla y poca. Entre los datos más comunes se encuentran los perfiles de los jugadores (nombre, equipo, caracter entre otras) y la posición dentro del juego en la que se encontraban al salvar la información.

Pero otros como el World of Warcraft o el Battle Field Vietnam, hacen de uso de sistemas de bases de datos (SDB) para el almacenamiento de su información, debido a que los registros que se almacenan son de mayor volumen y complejidad. Los datos que se suelen almacenar en este tipo de juegos tienden a ser más complejos porque almacenan información detallada de los equipos como pueden ser la cantidad de misiones cumplidas, el tiempo en que demoraron en alcanzar el objetivo, el armamento con el que cuentan y el ranking de los equipos que intervienen en el enfrentamiento entre otros. Además de estos datos colectivos se guarda información de cada jugador en particular como puede ser el rango que ocupan dentro del equipo, el nombre, el equipo y el caracter.

Por la poca existencia de información sobre sistemas de BD tradicionales para el trabajo con aplicaciones de RV. Durante el desarrollo de este trabajo se centrará la atención en aquellos juegos que utilizan SBD y principalmente en las características de estos SGBD.

En la información consultada para realizar esta investigación se destaca como SGBD, MySQL por las potencialidades que este brinda como son el gran número de conexiones que soporta, el hecho de poseer una infinidad de bibliotecas y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, su portabilidad a otros sistemas así como su fácil instalación y configuración. Actualmente la desventaja más significativa de este SGBD es que desde su versión 5 este pasó a ser de licencia propietaria.

1.4 SGBD que operan utilizando la RAM.

1.4.1 SBD de Memoria principal.

Los SBD residentes en la memoria principal (Main Memory Database Systems, MMDB), almacenan sus datos en la memoria física principal y brindan un acceso muy rápido. Los SBD convencionales están optimizados para los mecanismos particulares de almacenamiento en discos. Los MMDB, por otra parte, usan diferentes optimizaciones en la estructura y organización de los datos. [17]

En los SBD convencionales, los datos se almacenan en el disco duro y pueden ser “cacheados" en memoria para su acceso; en los SBD de memoria principal, los datos se almacenan de forma permanente en la memoria física y pueden tener un “backup" en el disco duro. El hecho de que la copia primaria esté en memoria tiene importantes implicaciones, como por ejemplo, cómo estructurar y acceder la BD. Algunas diferencias en cuanto a tener los datos en memoria o en el disco, son las siguientes:

- El tiempo de acceso a la memoria es mucho menor que en el disco duro.
- La memoria principal es normalmente volátil y el disco duro no.
- La memoria principal es accedida directamente por el CPU, por lo que los datos en la memoria son más vulnerables a errores de software que en el disco. [17]

Estas diferencias tienen efectos en casi todos los aspectos del manejo de una BD, desde el control de concurrencias hasta la interfaz de aplicación. [17]

En muchas aplicaciones de tiempo real, los datos deben estar residentes en la memoria, y por tanto la BD necesariamente tiene que tener menor tamaño que la memoria disponible. Sin embargo, existen casos en que la BD no cabe en la memoria, y para estos casos las BD tradicionales continúan siendo importantes. [17]

No obstante, incluso para las aplicaciones grandes, es común tener diferentes tipos de datos: “hot" data, que son frecuentemente accedidos, usualmente de poco volumen y con requerimientos de tiempo de acceso restringidos; y “cold" data, accedidos raramente y de mayor volumen. En estos casos, es posible particionar los datos en varias BD lógicas, y almacenar las “calientes" en la memoria principal. [17]

Dada la mayor velocidad de acceso a datos en la memoria principal, existen diferencias en la implementación de ambas bases de datos en cuanto al control de concurrencia, procesamiento de los commit, métodos de acceso, representación de datos, procesamiento de consultas, recuperación, rendimiento, protección y clustering y migración de datos. [17]

Se han propuesto o implementado algunos sistemas de manejo de BD residentes en memoria, como MM-DBMS, MARS y HALO (solamente diseñadas), OBE, TPK y System M (a nivel de prototipos y pruebas) y Fast Path (sistema comercial) (estos ejemplos no son abarcadores, solamente son algunos sistemas representativos). Una comparación entre estos sistemas se puede encontrar en “Main Memory Database Systems: An Overview". [17]

Existen propuestas de implementación de sistemas de BD de memoria principal orientadas a objeto. En “Object-Oriented Design of Main-Memory DBMS for Real-Time Applications", se presenta la arquitectura de M^2RT, un DBMS de tiempo real basado en memoria principal, y un diseño orientado a objeto de su sistema de almacenamiento, llamado M2RTSS, el cuál brinda clases que implementan las funcionalidades del manejo del almacenamiento, las transacciones en tiempo real y la recuperación. M^2RTSS puede fácilmente incorporar nuevos desempeños en los requerimientos de las aplicaciones de tiempo real. [17]

1.5 Bibliotecas para el desarrollo de interfaces gráficas de usuario en GNU/Linux.

1.5.1 Qt

Actualmente Qt constituye una amplia plataforma de desarrollo que incluye un amplio conjunto de clases, bibliotecas y herramientas para la producción de Interfaces Gráficas de Usuario, y hace uso del lenguaje C++ como lenguaje de programación, a su vez es capaz de operar en varias plataformas. Con esta biblioteca se pueden llegar a desarrollar enriquecidas aplicaciones gráficas, incluye soporte para nuevas tecnologías como OpenGL, XML, BD, Implementación de Aplicaciones de Redes, internacionalización, entre otras. [14]

Esta biblioteca está provista de una amplia gama de herramientas que facilitan la creación de formas, botones y ventanas de diálogo con el uso del ratón por citar algunos ejemplos. Las aplicaciones desarrolladas bajo Qt son muy elegantes, se ven y se operan mejor que las aplicaciones nativas. [14]

Está provista de tres grandes ventajas ante otras bibliotecas similares y que la rivalizan a su vez:

- Qt es completamente gratuito para Aplicaciones de Código Abierto.
- Es una biblioteca que está disponible para casi todas las plataformas, ya sea Sistemas Unix, Linux, MacOS, Solaris, así como para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código, siendo así, que esta se visualizará y actuará mejor que una aplicación nativa.
- Qt hace uso de una extensa biblioteca de clases y herramientas para la creación de Aplicaciones Gráficas, estas bibliotecas de clases están bien documentadas, así como de fácil uso y tienen una gran herencia de programación orientada a objetos, lo cuál hace de la programación de interfaces gráficas de usuario una aventura placentera. [14]

1.5.2 Gtk

Gtk+ o The Gimp Toolkit (conjunto de rutinas para GIMP), es un conjunto importante de bibliotecas o rutinas para desarrollar Interfaces Gráficas de Usuario (GUI) principalmente para entornos gráficos GNOME, XFCE y ROX de sistemas Linux. GTK es software libre (licenciado bajo la licencia LJPL), multiplataforma y parte importante del proyecto GNU. [14]

En sus inicios GTK fue concebido para desarrollar el programa de manejo de imágenes GIMP, sin embargo, actualmente es muy usada por muchos otros programas en los sistemas GNU/Linux. [14]

GTK fue diseñada para permitir la programación con lenguajes como C, C++, Java, Perl o Python. Actualmente, su última versión (GTK 2), la cuál incluye una gran cantidad de mejoras importantes, lo que lo hace superior a sus versiones anteriores. Sin embargo, no son compatibles su primera versión y la segunda. [14]

1.6 Formato de fichero XML

1.6.1 Historia del XML.

Extensible Markup Language (XML) de sus siglas en inglés (lenguaje de marcas extensible), proviene de un lenguaje inventado por IBM en los años setenta, llamado GML (*Generalized Markup Language*), que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información. Este lenguaje tubo muy buena aceptación por la ISO, por lo que en 1986 trabajaron para normalizarlo, creando SGML (*Standard Generalized Markup Language*), capaz de adaptarse a un gran abanico de problemas. A partir de él se han creado otros sistemas para almacenar información. [2]

XML es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML. [2]

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. [2]

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable. [2]

1.6.2 Ventajas del XML.

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera, se evitan *bugs* y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. [2]

1.6.3 Estructura de un documento XML.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de pedazos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman *elementos*, y se las señala mediante etiquetas. [2]

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma *<nombre>,* donde *nombre* es el nombre del elemento que se está señalando. [2]

1.6.4 Partes de un documento XML.

Un documento XML está formado por el prólogo y por el cuerpo del documento así como texto de etiquetas que contiene una gran variedad de efectos positivos o negativos en la referencia opcional a la que se refiere el documento, hay que tener mucho cuidado de esa parte de la gramática léxica para que se componga de manera uniforme. [2]

- **Prólogo**: aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.
    - El prólogo de un documento XML contiene:
    - Una declaración XML. Es la sentencia que declara al documento como un documento XML.
    - Una declaración de tipo de documento. Enlaza el documento con su DTD (definición de tipo de documento), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
    - Uno o más comentarios e instrucciones de procesamiento. [2]

- **Cuerpo**: no es opcional en un documento XML, el cuerpo debe contener un y solo un elemento raíz, característica indispensable también para que el documento esté bien formado. Sin embargo, es necesaria la adquisición de datos para su buen funcionamiento. [2]
- **Elementos**: XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos. [2]
- **Atributos:** que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas. [2]
- **Entidades**: para representar caracteres especiales para que, de esta forma, no sean interpretadas como marcado en el procesador XML. [2]

1.6.5 Validez de un documento XML.

Que un documento esté "bien formado" solamente se refiere a su estructura sintáctica básica, es decir, que se componga de elementos, atributos y comentarios como XML especifica que se escriban. Cada aplicación de XML, es decir, cada lenguaje definido con esta tecnología, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento. [2]

Esta relación entre elementos se especifica en un documento externo o definición (expresada como DTD *(Document Type Definition = Definición de Tipo de Documento)* o como XSchema). Crear una definición equivale a crear un nuevo lenguaje de marcado, para una aplicación específica. Lo que puede verse como una especie de ayuda donde se recojan todas las reglas seguidas para la construcción de fichero XML, dicho documento será el que valide la veracidad del fichero de acuerdo a las pautas que en el se recojan como normas de creación.

1.6.6 Herramientas para trabajar con documentos XML

Cualquier procesador de texto, que sea capaz de producir archivos .txt es capaz de generar XML, aunque en los entornos de desarrollo como Eclipse o Visual Studio, se facilita, ya que reconoce los formatos y ayuda a generar un XML bien formado. [2]

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Descripción de la solución propuesta

Para darle solución al problema científico se implementa una aplicación que le permite al usuario crear proyectos o lo que sería lo mismo crear diferentes BD, para realizar esta funcionalidad se brindan controladores tanto a nivel de menú como de botones donde el actor puede ir agregando una a una de las tablas que componen su BD, así como los campos que contendrán cada tabla y sus respectivos tipos de datos. Concluido este proceso se deben de generar las bibliotecas, los códigos fuentes con las cabeceras de las clases principales y la BD en sí misma.

Los archivos que se obtienen finalmente se localizan en la dirección que el usuario especifique en el paso “Generar Biblioteca”. De esta dirección deben de ser copiados hacia la dirección donde se encuentre la aplicación para la que finalmente fueron creados, posteriormente sería necesario incluir la biblioteca y el archivo DBClient.h, para que cuando se compile la aplicación esta pueda linkearse con estos nuevos ficheros y de esta manera tener acceso a la base de datos y a las funcionalidades de inserción, actualización y visualización de los datos que se almacenan en la misma. En la figura 1 se ilustra de una forma gráfica el proceso de funcionamiento de la aplicación que se realizará.

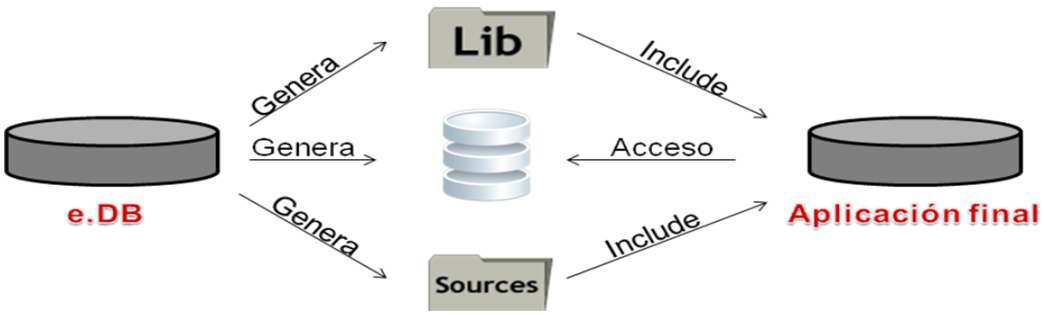

Figura # 1: Descripción del proceso de funcionamiento de la aplicación.

2.1.1 Configuración de los tipos de datos.

Los tipos de datos que manejará la aplicación serán los cuatro tipos de datos básicos (entero, text, flotante y booleano) los mismos serán definidos a través de un enumerador.

2.1.2 Herramientas de desarrollo y lenguaje de programación utilizados.

El uso de estas herramientas estará presente durante todos y cada una de las fases de desarrollo de esta Biblioteca, así como el lenguaje de programación que se referencia.

2.1.2.1 Herramienta case: Visual Paradigm

Visual Paradigm es una herramienta UML de tipo profesional, la cual soporta el ciclo completo de desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Este software de modelado UML constituye una gran ayuda en la rápida construcción de aplicaciones de calidad a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generador de código desde diagramas y generar documentación. Esta herramienta CASE (Computer Aided Software Engineering) como comúnmente se conoce esta provista de un gran número de tutoriales de modelado UML, demostraciones interactivas de UML, así como proyectos de este tipo. Algunas de sus principales características son:

- Diagrama de Procesos de Negocio – Proceso, Decisión, Actor de Negocio, Documento Modelado colaborativo con CVS y Subversion (nueva característica).
- Ingeniería de Ida y Vuelta.
- Ingeniería Inversa, Código a modelo, código a diagrama;
- Generación de Código, Modelo a Código, Diagrama a Código.
- Diagrama de flujo de datos.
- Generación de bases de datos, Transformación de Diagramas Entidad-Relación en tablas de bases de datos.
- Generador de informes para elaboración de documentación.
- Distribución automática de diagramas, Reorganización de las figuras y conectores de los diagramas UML.
- Importación y Exportación de ficheros XMI, así como archivos de proyectos Rational Rose MDL y CAT.
- Carácter multiplataforma, ya que hace uso de esta herramienta tanto en sistemas Linux, Unix o Windows.

2.1.2.2 Lenguaje de Modelado UML 2.0

Lenguaje Unificado de Modelado (UML, “Unified Modeling Language”) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

Entre sus principales características se encuentran:

- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así, modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación `*Ingeniería directa e inversa*`.
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas y versiones).
- Cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos.

UML es un lenguaje para especificar y no un método o un proceso, se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir el lenguaje en el que está descrito el modelo.

2.1.2.3 Lenguaje de programación: C++

Es un lenguaje imperativo orientado a objetos derivado de **C**. En realidad un superconjunto de **C**, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. Posee mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido y multiplataforma.

2.1.2.4 IDE para la implementación: Eclipse

¿Qué es eclipse? En la web oficial de Eclipse (www.eclipse.org), se define como “un IDE para todo y para nada en particular” (“An IDE for everything and nothing in particular”). Eclipse es únicamente un armazón (workbench) sobre el que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados. La arquitectura de plugins de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para bibliotecas, etc. Eclipse posee generación automática de esqueletos, get y set. Una de las características más curiosas del IDE Eclipse es el modo en que se compilan los proyectos. No existe en Eclipse ningún botón que permita compilar individualmente un fichero concreto. La compilación es una tarea que se lanza automáticamente al guardar los cambios realizados en el código. La principal diferencia entre un simple editor y un buen entorno de desarrollo es que éste integre, o no, una buena herramienta visual para depurar los programas escritos. Eclipse incluye un depurador potente, sencillo y muy cómodo de utilizar. [15]

2.1.2.5 Herramientas de autoconstrucción: Automake.

Automake intenta que el programador pueda escribir un archivo makefile en un lenguaje de alto nivel en vez de tener que escribir todo el makefile manualmente. En los casos simples es suficiente con dar: una línea que declare el nombre del programa a construir; una lista de archivos fuente; una lista de opciones para pasar al compilador (los directorios donde localizar los archivos de cabecera); una lista de opciones para pasar al enlazador (las bibliotecas que necesita el programa y en qué directorios se encuentran). Con esta información Automake genera un archivo makefile que permite que el usuario pueda: compilar el programa; limpiar (eliminar los archivos resultantes de la compilación); instalar el programa en los directorios estándar; desinstalar el programa donde se haya instalado; crear un archivo fuente de distribución (comunmente llamado *tarball*); comprobar que el archivo es autosuficiente y en concreto que el programa puede ser compilado en un directorio diferente al que se descomprimió los fuentes. Automake también puede ayudar en la compilación de bibliotecas generando automáticamente archivos makefile que invocarán la herramienta GNU Libtool. El programador no necesita conocer cómo llamar directamente a Libtool y el beneficio de usar una utilidad de creación de bibliotecas portables. [16]

2.1.2.6 SQLite

Es un SGBD que a diferencia de los SGBD cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. [11]

Esto reduce la latencia en el acceso a la BD, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la BD (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. En su versión 3, SQLite permite BD de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB. Implementa un completo motor de BD multiplataforma que no precisa configuración. Se distribuye bajo licencia de dominio público. Es muy rápido y la ventaja fundamental es que permite utilizar un amplio subconjunto del lenguaje estándar SQL. SQLite destaca también por su versatilidad. Combina el motor y el interfaz de la BD en una única biblioteca, y almacena los datos en un único archivo de texto plano. Esto hace que cada usuario pueda crear tantas BD como desee sin la necesidad de la intervención de un administrador de BD que gestione los espacios de trabajo, usuarios y permisos de acceso. El hecho de almacenar toda la BD en un único archivo, facilita la portabilidad de los datos, y solamente tiene la restricción del espacio de disco asignado al usuario en el servidor. [11]

2.1.2.7 Qt

Esta biblioteca está provista de una amplia gama de herramientas que facilitan la creación de formas, botones y ventanas de diálogo con el uso del ratón por citar algunos ejemplos. Las aplicaciones desarrolladas bajo Qt son muy elegantes, se ven y se operan mejor que las aplicaciones nativas. Qt es completamente gratuito para Aplicaciones de Código Abierto. Es una biblioteca que está disponible para casi todas las plataformas, ya sea Sistemas Unix, Linux, MacOS, Solaris, así como para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código, siendo así, que esta se visualizará y actuará mejor que una aplicación nativa. Qt hace uso de una extensa biblioteca de clases y herramientas para la creación de Aplicaciones Gráficas, estas bibliotecas de clases están bien documentadas, así como de fácil uso y tienen una gran herencia de programación orientada a objetos, lo cual hace de la programación de interfaces gráficas de usuario una aventura placentera. [14]

2.2 Diseño del sistema.

2.2.1 Arquitectura del software.

El patrón arquitectónico diseñado para este trabajo de diploma es el: “Modelo en capas”. Actualmente el sistema cuenta con 2 capas una de aplicación, donde se encuentran las clases de interfaz de usuario que representan las pantallas de la aplicación que el usuario ve, algunas de las clases que se encuentran en esta capa son: DBEditorGUI, NewProjectGUI, NewTableGUI, GenLibGUI, TableGUI y EditTableGUI, además se cuenta con una capa de servicio de negocio la cual contiene todas las clases controladoras que representan el comportamiento de la aplicación, según los casos de uso, dentro de esta capa se ubican las clases ProjectManajer, LibManajer, DBObject y TableObject por citar algunas.

2.2.2 Patrones de diseño utilizados.

Para llevar a cabo el proceso de implementación del software se definieron los patrones de diseño que se utilizarían, con el objetivo de diseñar a un mayor nivel de abstracción así como contribuir en la reutilización del diseño, identificando aspectos claves de su estructura, reduciendo de esta manera los esfuerzos de desarrollo y mantenimiento.

Los patrones aplicados pertenecen al grupo GRASP (General Responsibility Assignment Software Patterns) que nos son más que los patrones encargados de asignar responsabilidades.

Dentro de los patrones GRASP utilizados se encuentran:

- *Experto*: el cuál es el encargado de asignar las responsabilidades de una acción al experto en la información.
- *Creador*: es el que decide quién es el responsable de crear una nueva instancia de alguna clase.
- *Controlador*: es el encargado de decidir quién debe atender un evento del sistema.
- *Bajo acoplamiento*: el que provee al sistema de una mínima dependencia y un aumento de la reutilización.

2.2.3 Modelo de dominio.

Para un mejor entendimiento de los procesos que se necesitan integrar en esta aplicación se realiza el modelo de dominio ya que aún no se tiene bien definidos cuáles serán las principales funciones, actores y responsabilidades que se tendrán dentro de este sistema informático.

El modelo de dominio se ilustran los conceptos fundamentales que son manipulados en el contexto de este trabajo de diploma así como las relaciones que entre ellos se pueden establecer.

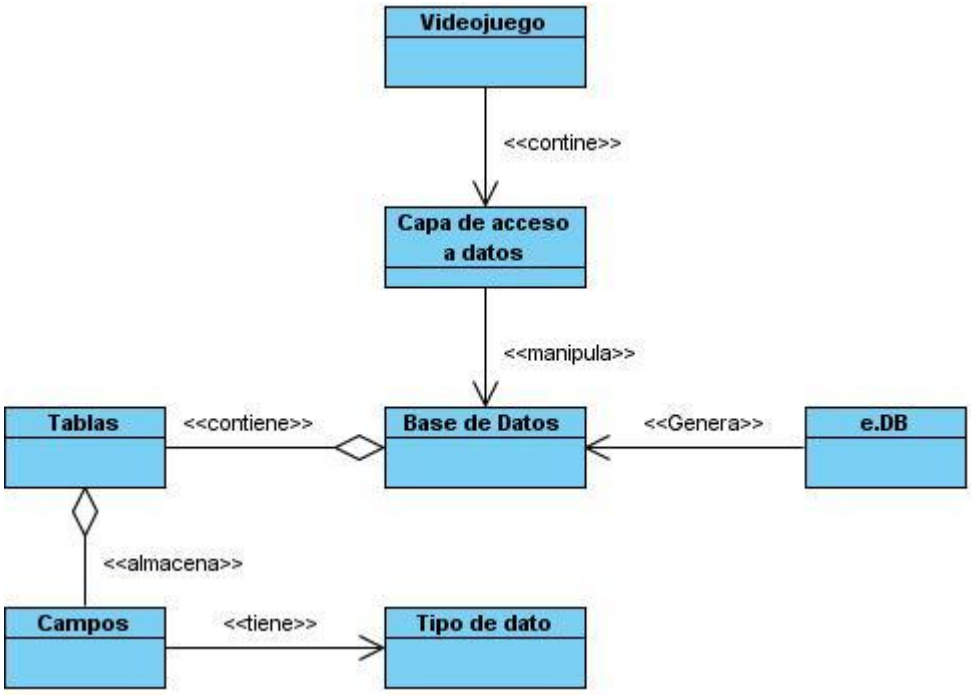

Figura # 2: Modelo de dominio.

2.2.4 Captura de Requisitos.

El primer paso para diseñar el sistema fue la selección de los requisitos o indicadores que debían cumplir el sistema, partiendo de sus principales necesidades, estos se agrupan en funcionales y no funcionales. Los requisitos funcionales no son más que las capacidades o condiciones que el sistema debe cumplir, y los requisitos no funcionales, representan propiedades o cualidades con las que el producto debe contar.

2.2.4.1 Requisitos funcionales.

1. Diseñar BD.
   1.1. Crea proyecto.
   1.2. Insertar tablas.
   1.3. Insertar campos.
   1.4. Definir tipos de datos.
2. Generar la biblioteca.
   2.1. Verificar si existe la biblioteca.
   2.2. Verificar si la BD está creada.
   2.3. Compilar la biblioteca.
   2.4. Generar la BD.

2.2.4.2 Requisitos no funcionales.

- **Requerimientos de Software:**

La aplicación fue desarrollada en el Sistema Operativo GNU/Linux Debian. Se utilizó Eclipse como IDE de programación, así como Qt como Biblioteca de desarrollo de Interfaces Gráficas de Usuarios.

- **Requerimientos de Hardware:**

Es necesaria una computadora con CPU Intel Pentium IV, 80 Gb de HDD, 512 Mb de RAM.

- **Restricciones en el Diseño y la Implementación:**

  - Lenguaje de programación: C++.
  - Herramientas de Desarrollo: Eclipse, Automake, y Berkeley DB.
  - Biblioteca de desarrollo: Qt.

- **Requerimientos de Usabilidad:**

Los futuros usuarios de la aplicación, no necesariamente tienen que ser programadores. Solo se necesita tener conocimiento acerca de BD relacionales y los diferentes reportes que necesitan realizar.

- **Requerimientos de Soporte:**

La aplicación debe de ser compatible con el Sistema Operativo GNU/Linux Debian

2.2.5 Modelo de Casos de Uso

2.2.5.1 Definición de Actores.

Actor	Justificación
Programador	Realiza todas las acciones (diseño de la BD, establecer los niveles de acceso a la BD, generar y compilar la biblioteca) dentro de la aplicación a implementar.

Tabla # 1: Definición del actor del sistema.

2.2.5.2 Descripciones abreviadas de los Casos de Usos del Sistema.

Caso de Uso	Configurar Estructura de la BD
Actor	Programador
Descripción	Se crearán las estructuras para diseñar la BD que necesita el programador para su aplicación.
Referencia	RF 1, 1.1, 1.2, 1.3, 1.4

Tabla # 2: Caso de Uso "Configurar Estructura de la BD"

Caso de Uso	Compilar biblioteca
Actor	Programador
Descripción	El programador decidirá en qué dirección salvará la biblioteca que generará y compilará la misma.
Referencia	RF 2.3

Tabla # 3: Caso de Uso "Compilar Biblioteca".

Caso de Uso	Generar Biblioteca
Actor	Programador

Descripción	Luego de configurada la BD y su acceso a la misma se generará la biblioteca necesaria para manejar sus datos.
Referencia	RF 2, 2.1, 2.2, 2.3, 2.4

Tabla # 4: Caso de Uso "Generar Biblioteca".

2.2.5.3 Casos de uso del Sistema.

Figura # 3: Diagrama de Casos de Uso del Sistema.

2.2.5.4 Descripciones de los casos de uso del sistema.

Caso de Uso:	Configurar la Estructura de la BD
Actores:	Programador
Trabajadores:	-
Resumen:	El caso de uso comienza cuando el programador decide configurar los elementos que compondrán la base de datos de la aplicación que va a desarrollar.
Referencia:	RF 1, 1.1, 1.2, 1.3, 1.4
Precondiciones:	Haber abierto la aplicación "DBEditor" en el espacio de trabajo "Diseñador de base de datos".

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. El programador inicia creando un nuevo proyecto. 2. Introduce el nombre del proyecto necesarios para crear el mismo y con el	1.1 Visualiza la interfaz crear proyecto. 2.1 cambia el nombre que tiene la barra de la aplicación por defecto (DBEditor) por el nombre que el actor definió mostrando que

que se salvará la BD finalmente.	fue creado satisfactoriamente.
3. El programador inicia el proceso de inserción de las tablas que componen la BD.	1.2 El sistema muestra la interfaz donde se brindan las opciones de: - Nombrar la tabla. - Nombre del campo. - Añadir los tipos de datos. - Adicionar campos. - Eliminar campo.
4. El actor solicita **insertar una nueva tabla**.	1.3 El sistema muestra las opciones anteriores.
5. El programador puede seleccionar la opción **editar tabla**.	1.4 El sistema muestra una interfaz con todos los datos que contiene la tabla y la posibilidad de modificarlos según lo que requiera.
6. El programador además puede seleccionar una tabla y escoger la opción **eliminar tabla**.	1.5 El sistema eliminaría esta tabla de la interfaz principal.
Post condiciones	Queda configurada la BD de la aplicación a implementar.
Prioridad	Crítico.

Tabla # 5: Expansión del caso de uso "Configurar la Estructura de la Base de Datos".

Caso de Uso:	Generar biblioteca
Actores:	Programador
Trabajadores:	-
Resumen:	El caso de uso inicia cuando el actor selecciona la opción del menú contextual generar biblioteca o del botón generar biblioteca de la barra de acceso rápido, desencadenándose una serie de pasos que tienen

	como fin obtener los archivos necesarios para la aplicación.
Referencia:	RF2, 2.1, 2.2, 2.3, 2.4
Precondiciones:	Haber configurado la estructura de la BD.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Selecciona la opción **Generar biblioteca.**	1.2 El sistema muestra la interfaz perteneciente a esta opción. Solicitando la dirección donde se salvara la biblioteca que se generara.
2. Selecciona la dirección donde se salvará la biblioteca, el nombre por defecto de de la misma será el que le puso al proyecto inicialmente.	2.2 Verifica que no exista una biblioteca con el mismo nombre en la dirección especificada. 2.3 Si existe la biblioteca se muestran las opciones: - **Sobrescribir la biblioteca existente** (Flujo Alterno de Eventos #1). - **Renombrar la biblioteca a salvar** (Flujo Alterno de Eventos #2). - **Cambiar la dirección** donde se va a salvar la biblioteca (Flujo Alterno de Eventos #3). 2.4 Si no existe la biblioteca: - Comprueba que la BD esta creada. - Verifica si tiene configurado los niveles de acceso correctamente. - Compila la biblioteca con esos datos. - Almacena la biblioteca en la dirección especificada con el nombre

	seleccionado.
Flujo Alterno de Eventos	
Acción del usuario	**Acción del sistema**
1. Selecciona **"Sobrescribir la biblioteca existente".**	1.2 Continúa con el flujo normal de eventos.
2. Selecciona **"Renombrar la biblioteca existente".**	2.1 El sistema muestra una interfaz para capturar el nuevo nombre de la biblioteca a salvar. 2.2 Continúa con el flujo normal de eventos.
3. Selecciona **"Cambiar la dirección".**	3.1 El sistema muestra una interfaz para que el usuario entre la nueva dirección. 3.2 Continúa con el flujo normal de eventos.
Post condiciones	Queda creada y salvada la biblioteca que el programador necesita para su aplicación.
Prioridad	Crítico.

Tabla # 6: Expansión del caso de uso "Generar Biblioteca".

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SOFTWARE

3.1 Modelo de Clases del análisis.

La interfaz que se implementa posee como una de sus características principales que es intuitiva, significando con esto que los usuarios que posean acceso a la misma podrán con solo ver los iconos de los botones de la barra de acceso rápido identificar claramente cuáles serán las acciones que a realizar.

A continuación se muestran imágenes de cómo quedaría organizado el sistema por interfaces y clases en general.

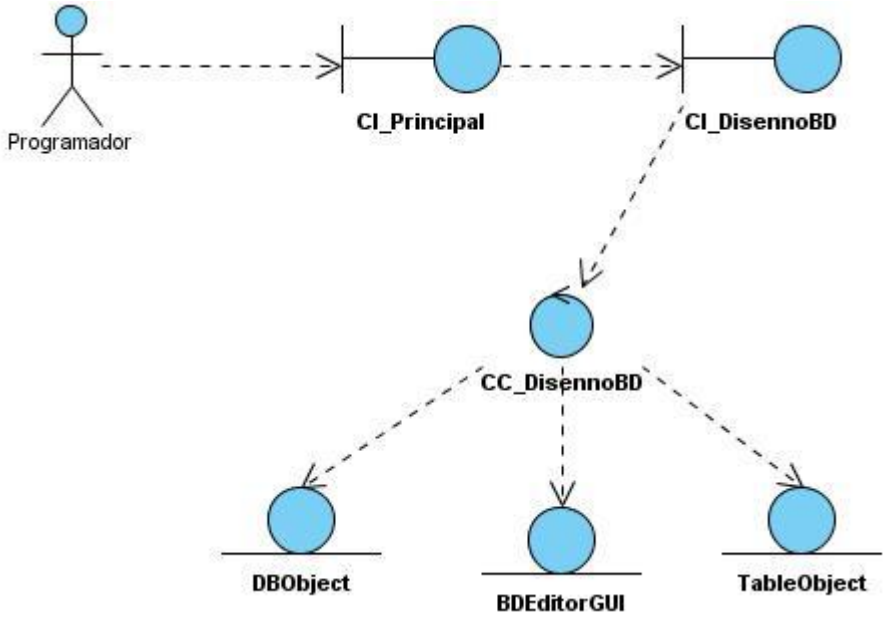

Figura # 4: Diagrama de clase del análisis caso de uso "Configurar estructura de la BD".

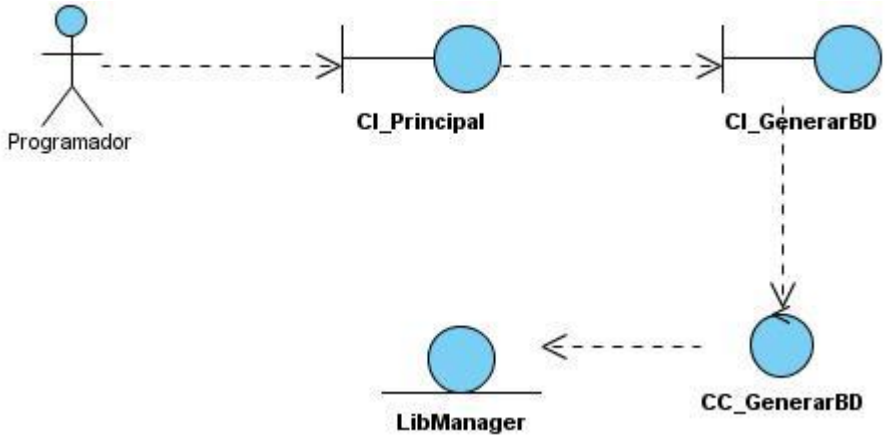

Figura # 5: Diagrama de clase del análisis caso de uso "Generar Biblioteca".

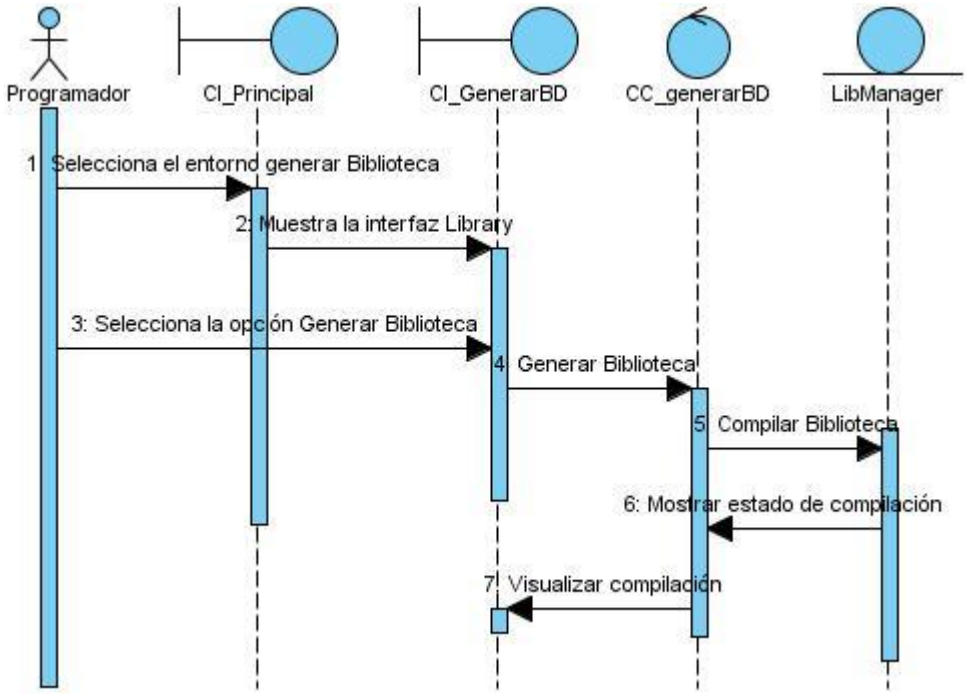

Figura # 6: Diagrama de secuencia caso de uso "Generar Biblioteca".

3.2 Diagramas de Interacción.

3.2.1 Diagramas de secuencia.

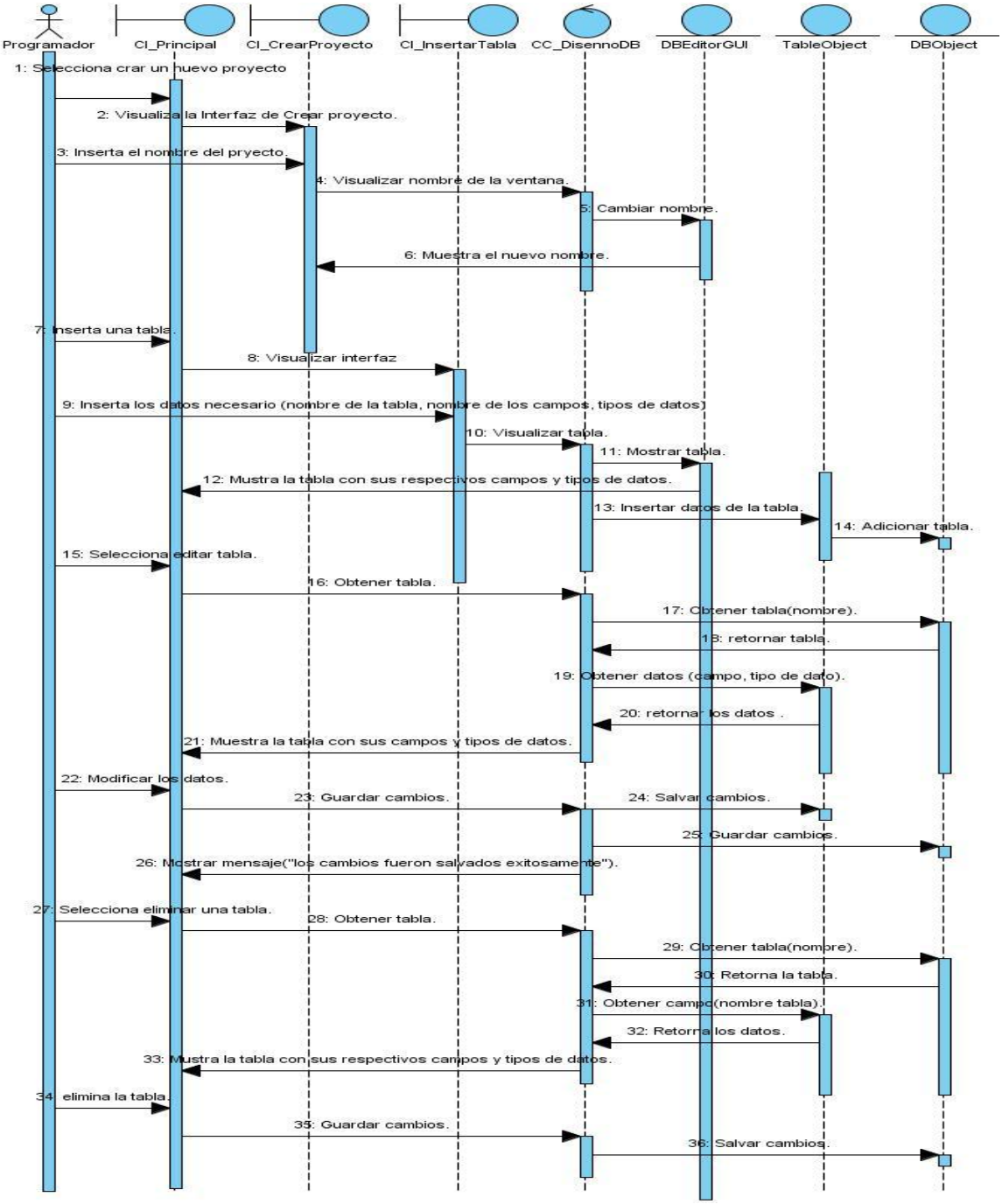

Figura # 7: Diagrama de secuencia del caso de uso "Configurar la Estructura de la BD".

3.3 Diagrama de clases del diseño.

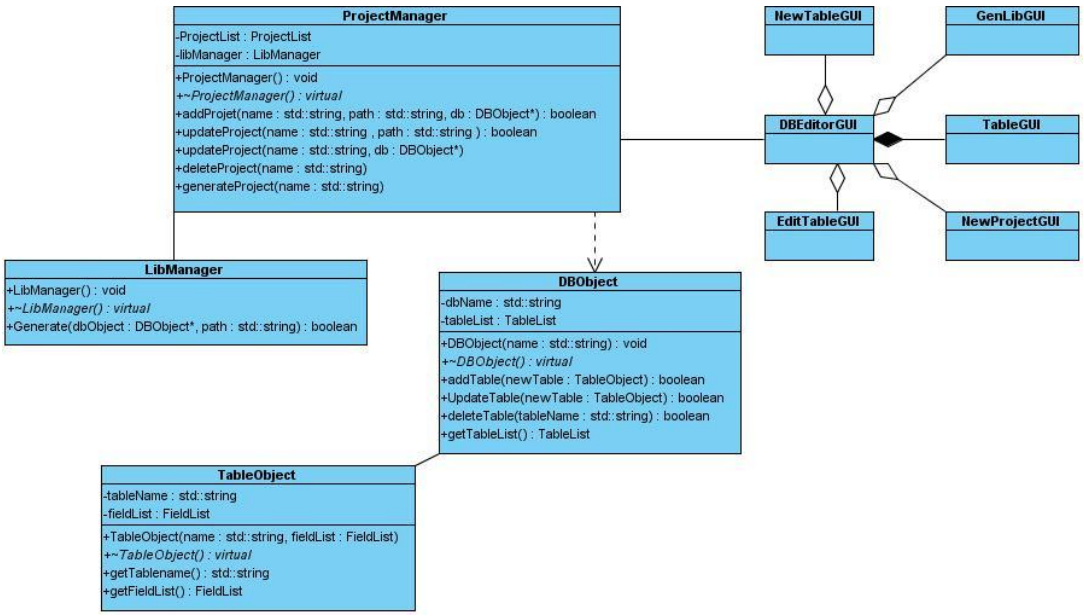

Figura # 8: Diagrama de clases del diseño.

3.4 Descripción de las clases del diseño.

Nombre: ProjectManager	
Tipo de clase: controladora	
Atributo	**Tipo**
projectList	ProjectList
libManager	LibManager
Para cada responsabilidad:	
Nombre:	ProjectManager()
Descripción:	Constructor de la clase principal.
Nombre:	~ProjectManager ()
Descripción:	Destructor de la clase principal
Nombre:	addProject (std::string name, std::string path, DBObject* db =NULL)
Descripción:	Este método adiciona una nueva base de datos, con nombre y dirección de salvado según se especifique en los parámetros, la misma inicialmente

	estará vacía.
Nombre:	updateProject (std::string name, std::string path)
Descripción:	Es el encargado de actualizar los cambios realizados dentro de la base de datos creada, para este caso en específico el camino o dirección donde esta guardada.
Nombre:	updateProject (std::string name, DBObject* db)
Descripción:	Es el encargado de actualizar los cambios realizados dentro de la base de datos creada, para este caso en específico la base de dato creada.
Nombre:	deleteProject (std::string name)
Descripción:	Elimina una base de datos creada dado su nombre.
Nombre:	generateProject (std::string name)
Descripción:	Es el encargado de generar la biblioteca y la base de dato a partir de las configuraciones diseñadas en la GUI.

Tabla # 7: Descripción de la clase del diseño "ProjectManager".

Nombre: LibManager	
Tipo de clase: Secundaria	
Para cada responsabilidad:	
Nombre:	LibManager ()
Descripción:	Constructor de la clase.
Nombre:	~LibManager ()
Descripción:	Destructor de la clase.
Nombre:	Generate (DBObject* db, std::string path)
Descripción:	Genera los archivos necesarios para la configuración de un proyecto específico, compilar e instalar la biblioteca en el directorio que se entra por parámetro, generando así el archivo .so, las cabecera y el archivo .pc correspondiente.

Tabla # 8: Descripción de la clase del diseño "LibManager".

Nombre: DBObject	
Tipo de clase: Secundaria	
Atributo	**Tipo**
dbName	std::string
tableList	TableList
Para cada responsabilidad:	
Nombre:	DBObject ()
Descripción:	Constructor de la clase.
Nombre:	~DBObject()
Descripción:	Destructor de la clase.
Nombre:	addTable (TableObject newTable)
Descripción:	Adiciona una nueva tabla a la base de datos creada, utilizando los parámetros definidos.
Nombre:	updateTable (TableObject newTable)
Descripción:	Actualiza los cambios que puedan sufrir las tablas ya creadas en la base de dato, remplazándolos o adicionándolos según sea el caso.
Nombre:	deleteTable (std::string tableName)
Descripción:	Elimina una tabla de la base de datos dado su nombre.
Nombre:	getTableList ()
Descripción:	Retorna una cadena la lista de nombre de las tablas que contiene la base de dato separados por coma.

Tabla # 9: Descripción de la clase del diseño "DBObject".

Nombre: TableObject	
Tipo de clase: Secundaria	
Atributo	**Tipo**
tableName	std::string
fieldList	FieldList
Para cada responsabilidad:	
Nombre:	TableObject ()

Descripción:	Constructor de la clase.
Nombre:	~TableObject()
Descripción:	Destructor de la clase.
Nombre:	getTableName ()
Descripción:	Retorna una cadena que contiene el nombre de la tabla.
Nombre:	getFieldList ()
Descripción:	Retorna la lista de campos que contiene una tabla.

Tabla # 10: Descripción de la clase del diseño "TableObject".

CAPÍTULO 4: IMPLEMENTACIÓN DEL SOFTWARE

4.1 Estándar de codificación.

Nombres

- Los nombres de las clases son sustantivos singulares.
- Los nombres deben reflejar el “que” y no el “como”.
- Escoger nombres lo suficientemente largo para ser expresivos, pero evitando manejar nombres que dificulten la labor de implantación.
- Evitar nombre que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear mayúscula para el inicio de cada palabra y minúscula para el resto de las letras para el caso de los nombres de métodos y funciones.
- Los nombres de constantes deben contener solo letras mayúsculas.
- Minimizar el uso de abreviaciones. En caso de ser requeridas, se debe ser consistente en su uso y cada abreviación debe significar solo una cosa. En general agregar a la documentación las abreviaturas.
- Los nombres de los métodos son frases que incluyen verbos.
- Los nombres de los atributos y parámetros son frases con sustantivos.
- Evitar el rehúso de nombres para distinto propósito.

Manejo de Errores.

- Se pueden manejar los errores mediante mecanismos de excepciones o mediante valores de retorno, aunque esto debe ser uniforme dentro de un mismo objeto.
- Es buena práctica emplear herramientas para identificar errores en la codificación en caliente.

Documentación y Comentarios.

- En el código debe documentarse en forma explicativa los pasos que se van ejecutando.
- Emplear oraciones completas al documentar código.
- Documentar mientras se programa.

- Documentar cualquiera cosa que no sea obvia en el código.
- Documentar eliminación de errores y cambios sobre el código.
- Al modificar el código se deben actualizar todos los comentarios y documentación asociada.
- Documentar cada rutina agregando: nombre del desarrollador, fecha, parámetros de entrada, valores de retorno, precondiciones, postcondiciones, dependencia con otros métodos o funciones y descripción general del algoritmo.
- Evitar agregar comentarios al final de líneas de código, salvo en el caso de declaraciones. En este caso tales comentarios deben estar alineados.
- Antes de la entrega de la aplicación, eliminar todos los comentarios superfluos y/o temporales con la finalidad de evitar confusiones en su mantenimiento.

Codificación.

- Se establece un tamaño de identación estándar de cuatro (4) espacios, sin tabulaciones. Alinear secciones del código.
- Alinear verticalmente llaves de apertura y cierre.
- Usar espacios antes y después de los operadores que el lenguaje de programación permita.
- Emplear líneas en blanco para organizar el código, permitiendo crear párrafos de código para una mejor lectura.
- Evitar colocar más de una sentencia por línea.
- Emplear constantes en sustitución de números o cadenas de caracteres literales.
- Emplear cada variable y rutina solo para un propósito.
- Evitar el uso de variables públicas, sustituirlas por variables privadas y métodos que provean el valor de tal variable, para mantener el encapsulamiento.
- Emplear select-case o switch en sustitución de if anidados sobre la misma variable.
- Liberar apuntadores de manera explícita.
- Emplear i, j, k, l, p, q, r para contadores en ciclos.
- Emplear al máximo, operadores del tipo: +=, *=, /=, -=, ++, --, etc.
- Emplear correctamente los tipos de ciclos: si es al menos una vez usar do-while, si es ninguna o más veces usar while-do, y si se conoce el número exacto de ciclos usar for.
- Inicializar todas las variables.
- Emplear líneas en blanco para separar pasos lógicos (declaraciones, lazos, etc.).
- Siempre asignar NULL a los apuntadores luego de ser destruidos (solo aplica para C).

- Evitar prácticas que incrementan explosivamente la complejidad, como lo son: objetos y variables globales y saltos tipo go to.

4.2 Diagrama de componente.

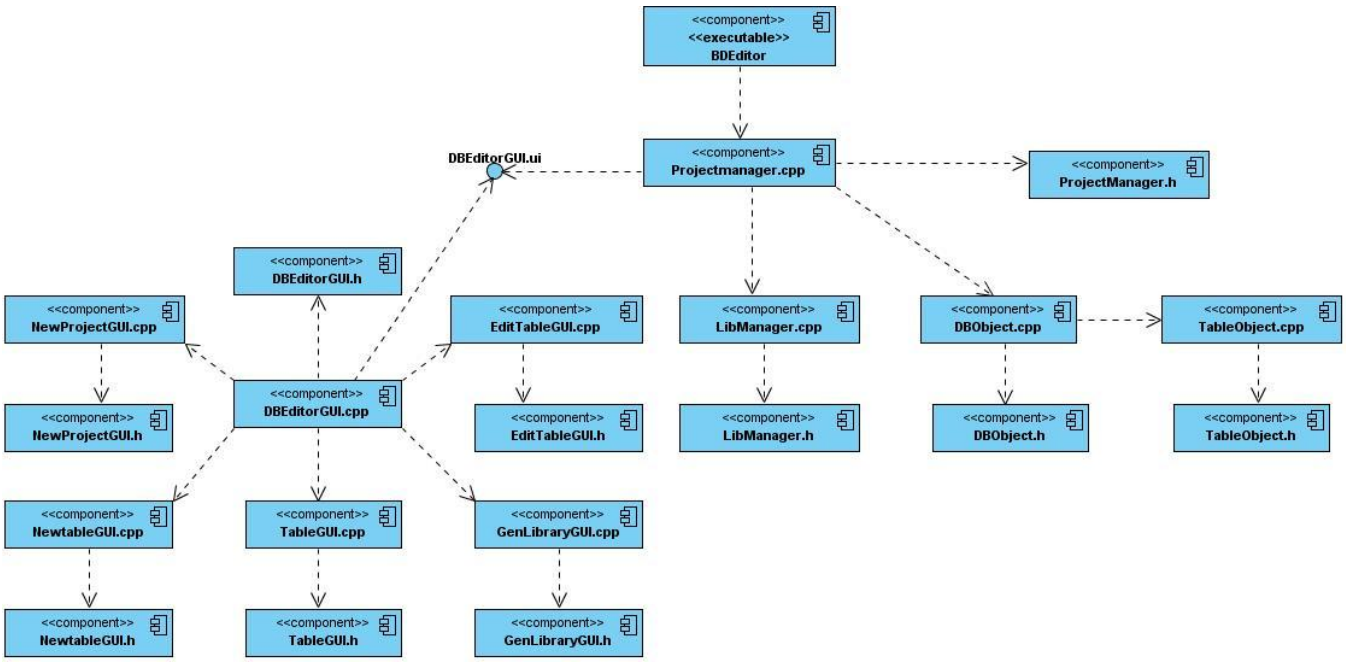

Figura # 9: Diagrama de componente.

4.3 Diagrama de despliegue.

Figura # 10: Diagrama de despliegue.

CONCLUSIONES

Dándole cumplimiento a los objetivos trazados al inicio de este trabajo de diploma, se hizo un estudio de diversos módulos de gestión y archivo de datos, además se analizaron varios Sistemas Gestores de Base de Datos (SGBD), principalmente aquellos que se utilizan en los videojuegos. El estudio realizado se centra esencialmente en la necesaria comunicación de las aplicaciones específicamente las de Realidad Virtual (RV) con los SGBD, para el almacenamiento de las informaciones persistentes de las mismas.

Además, fue necesario realizar una investigación de las principales bibliotecas para la construcción de interfaces gráficas de usuario, también se investigó acerca de las características fundamentales del formato XML, así como las ventajas que brinda para el almacenamiento de información. Proponiendo de esta forma un conjunto de soluciones técnicas, que posibilitarían llevar a cabo la implementación del software que da cumplimiento a la investigación desarrollada.

Al concluir el estudio del arte en estas materias se inició el proceso de Ingeniería de Software, haciendo uso de la Metodología de Desarrollo de Software RUP (Proceso Unificado Software). Se realizó el proceso de captura de requisitos, así como la identificación de los casos de usos del sistema junto con su descripción en formato expandido, se diseñaron las clases y se creó el diagrama de componentes, contenedor de las clases del sistema.

Como parte del ciclo de vida de RUP se llevó a cabo la implementación del sistema, del cuál se obtuvo una aplicación gráfica para la configuración de las bases de datos (BD) y la generación de la biblioteca encargada de manipular dicha BD.

La aplicación implementada cumple con todos los requisitos que se propusieron, permitiéndole al usuario final, una fácil configuración de las bases de datos (BD) y finalmente generar la biblioteca capaz de manipular la información almacenada en la misma, llegando así a reducir en gran medida el esfuerzo requerido por parte de los programadores a la hora de realizar dichas configuraciones a los diferentes proyectos del Departamento de Realidad Virtual.

RECOMENDACIONES

Al concluir el presente trabajo recomienda:

- Continuar el desarrollo de la aplicación incluyéndole la funcionalidad de salvado temporal en formato XML con el objetivo de que los usuarios no sean obligados a trabajar en un proyecto por espacio de un día o sencillamente no cerrar la aplicación porque perderían toda la configuración creada.
- Realizar las pruebas y despliegue del software para corroborar de que cumple con las expectativas para las que fue creado.
- Optimizar el manejo de errores dentro de la aplicación.

BIBLIOGRAFÍA CONSULTADA

1. **De la Horra Diaz, Abdelaziz.** *Desarrollo del subsistema de comunicación para el.* Ciudad de la Habana : s.n., Julio 2008.

2. **Nieblas Palau, Leonardo Antonio y Cubela Medina, Yasmany.** *Interfaz Visual para la configuración de Entornos Virtuales desarrollados con la Herramienta Scene Tool Kit.* Ciudad de la Habana : s.n., Mayo 2009.

REFERENCIAS BIBLIOGRÀFICAS

1. **Rodríguez, Ruben.** Base de Datos, Historia. *Sonria.com.* [En línea] 2 de Enero de 2004. [Citado el: 2 de Mayo de 2009.] http://www.fudim.org/comunicacion/notas/nota.php?id=22&a=Adim..

2. **Arévalos, Alonso.** *"Documents in Information Science" (DoIS): portal internacional de referencia para el profesional de la Información.* La Habana (Cuba) : In Proceedings Congreso Internacional de Información INFO 2004, 2004.

3. **SILBERSCHATZ, Abraham.** *Fundamentos de bases de datos.* 4a ed. Madrid : s.n., 2002. pág. 787. ISBN 8448136543.

4. **Salvador, OLIVÿN y ULLATE, ANGÓS.** *Directorio de Bases de Datos Internacionales.* [En línea] 8 de Mayo de 2006. [Citado el: 20 de Octubre de 2009.] http://wzar.unizar.es/perso/bdl/Directorio.pdf.

5. **Ortega, Octavio.** El futuro de los videojuegos: LA REALIDAD VIRTUAL. *GameOver.es.* [En línea] 18 de Abril de 2007. [Citado el: 15 de Enero de 2010.] http://www.gameover.es/gameover/el-futuro-de-los-videojuegos-la-realidad-virtual.html..

6. **Anónimo.** Realidad Virtual. *www.wikipedia.org.* [En línea] 10 de febrero de 2010. [Citado el: 15 de febrero de 2010.] http://es.wikipedia.org/wiki/Realidad_virtual..

7. —. Sistemas Gestores de Base de Datos. *www.wikipedia.org.* [En línea] 7 de Enero de 2010. [Citado el: 20 de Enero de 2010.] www.wikipedia.org..

8. PostgreSQL vs Mysql. *netpecos.com.* [En línea] [Citado el: 6 de Enero de 2010.] http://www.netpecos.org/docs/mysql_postgres/index.html..

9. About PostgreSQL. *postgresql.org.* [En línea] [Citado el: 6 de Enero de 2010.] http://www.postgresql.org/about/..

10. **Anónimo.** Instituto de Tecnologías Educativas. *Ministerio de Educacion España.* [En línea] [Citado el: 8 de Febrero de 2010.] http://usuarios.pntic.mec.es/sqlite.php#1..

11. —. QTLite. *QTLiteManagenet.org.* [En línea] 2008. [Citado el: 8 de Febrero de 2010.] http://www.sqlitemanager.org/.

12. **Masip, David.** Que es Oracle? *DesarrolloWeb.* [En línea] 19 de Julio de 2005. [Citado el: 8 de Febrero de 2010.] http://www.desarrolloweb.com/articulos/840.php.

13. **Hernández Orallo, José.** *La Disciplina de los Sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas.* Dep. de Sistemes Informàtics i Computació : Universitat Politècnica de València, mayo 2002.

14. **Mecklenburg**, **Robert** (Noviembre de 2004). *Managing Projects with GNU Make* (pdf - Bajo licencia GFDL), 3ª edición (en inglés), O'Reilly, pp. 300. ISBN 0-596-00610-1.

15. **Chavarría, Raúl Eduardo.** *IDE ECLIPSE, breve guía.* s.l. : Univercidad de Antioquia, 2006. Ingenieria de Sistema.

16. **Anónimo.** GNU Operating System. *Automeke, GNU Project.* [En línea] Free Software Fundation (FSF), 11 de enero de 2010. [Citado el: 5 de febrero de 2010.] http://www.gnu.org/software/automake/#documentation.

17. **Anónimo.** *SUBREGIONAL ENVIRONMENTAL MONITORING AND INFORMATION SYSTEM PHASE II (SEMIS II).* [En línea] mayo de 2002. [Citado el: 4 de mayo de 2010.]

APÉNDICES

Glosario de Abreviaturas

API: Application Programming Interface o Interfaz de Programación de Aplicaciones.

BD: Siglas que responden al concepto de base de dato.

DBMS: Database Management System o Sistemas Gestores de Base de Datos.

GUI: Graphic User Interface o Interfaz Gráfica de Usuario.

GPL: La GNU General Public License en inglés: Licencia Pública General, es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software.

GRASP: General Responsibility Assignment Software Patterns de sus siglas en inglés, traducido como patrones generales de asignación de responsabilidades.

IDE: IDE son las siglas de: Integrated Development Environment u Entorno Integrado de Desarrollo.

UML: Unified Modeling Language de sus siglas en inglés, traducido a Lenguaje Unificado de Modelado.

MMDB: Main Memory Database Systems o Sistema de Base de Dato de memoria principal.

QT: Es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario.

RV: Son las siglas que responden al concepto de Realidad Virtual.

SGML: Son las siglas de Standard Generalized Markup Language o "Lenguaje de Marcación Generalizado".

SGBD: Son las siglas de Sistema Gestor de Bases de Datos.

SBD: Siglas de Sistema de Bases de Datos.

XML: E**x**tensible **M**arkup **L**anguage o Lenguaje de Marcas Extensible, es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

Glosario de Términos

G:

GNU/Linux: Nombre por el que se conoce al sistema operativo formado por el conjunto de utilidades de sistema GNU y el núcleo Linux más otras aplicaciones libres creadas por terceros.

M:

Multiplataforma: Término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en Windows en un procesador x86, en GNU/Linux en un procesador x86, y en Mac OS X en uno x86.

S:

Sistema de Realidad Virtual: Sistema informático interactivo que ofrece una percepción sensorial al usuario de un mundo tridimensional sintético que suplanta al real.

SQL: Son las siglas de Structured Query Language, Lenguaje de consulta estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre la misma.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS