1. Manejadores de bases de datos

1.1 Bases de datos

Es frecuente pensar en una base de datos como una variedad de productos y sistemas en los cuales podemos almacenar cualquier cosa, desde una colección de archivos hasta estructuras complejas para interfaces de usuarios.

En vez de pensar en el tipo de datos que se pueden contener dentro de esta, la cantidad de registros que se puede almacenar, como se utilizará o bien un número de elementos que intervienen en una base de datos, la definición resulta un tanto más simple, una base de datos es una colección de datos que se relacionan entre sí de alguna forma. Por ejemplo una aplicación médica requerirá de datos de los pacientes, su nombre, dirección, teléfono, entre otros. Aunque una base de datos es más que simplemente varios datos relacionados. Los datos deben estar organizados y clasificados en un formato estructurado conocido como metadato, el cual es un tipo de datos que describe los datos que serán almacenados. En otras palabras, el metadato indica la forma en la que serán almacenados los datos dentro de la base de datos. De esta forma, los datos junto con el metadato proveen de un entorno lógico que organiza la forma en la cual se podrán acceder y mantener los datos de manera eficiente.

Para entenderlo mejor podríamos pensar en una analogía con la sección amarilla, la cual contiene un gran número de datos (entre nombres, teléfonos, direcciones), la forma en la que se organiza, ya sea por orden alfabético, número telefónico, zonas geográficas, etcétera, será el metadato y en conjunto estos nos permiten encontrar de manera eficiente alguna persona dentro del libro.

Sin embargo no todas las estructuras en las bases de datos tienen un mismo formato. Con el pasar de los años han emergido distintos modelos de datos, de

estos modelos, los tres más comúnmente utilizados son el jerárquico, el de red y el relacional.

1.2 El modelo jerárquico

El modelo jerárquico se basa en los registros individuales y la relación padre-hijo que forman un árbol invertido. Este árbol crea una estructura jerárquica en la cual los datos se descomponen en categorías lógicas y las subcategorías que utilizan los registros para representar unidades lógicas de datos.

Un registro padre puede tener múltiples registros hijos, pero un registro hijo solo puede tener un registro padre. Esta es una estructura similar a la que se puede ver en la estructura de directorios de un explorador de archivos.

Después de su introducción, el modelo jerárquico alcanzó un gran éxito. Una de las implementaciones más populares fue el Sistema de Administración de Información de IBM, el cual fue introducido en 1960 y actualmente se utiliza ampliamente en las computadoras mainframes de IBM.

Sin embargo, a pesar de la popularidad alcanzada por el modelo jerárquico, este modelo ya no ajustaba a las necesidades de las nuevas aplicaciones. Esto debido a la inherente cualidad del modelo, esta es una estructura rígida de la organización padre-hijo vistas que resulta en un proceso de navegación engorroso que requiere del desarrollo de aplicaciones para poder navegar a través de los registros para lograr dar con la información necesaria. Los registros tienen que ser accesados uno a la vez y de arriba para abajo a través de niveles jerárquicos, lo que hace que la modificación de la base de datos se torne muy laboriosa, lo que provocará un

proceso que consuma mucho tiempo. Además el modelo jerárquico no soporta las relaciones complejas entre los registros.

Incluso con las limitaciones del modelo jerárquico, en la actualidad existen diversos sistemas que ocupan este modelo para organizar sus datos y que han podido sobre llevar las limitantes del modelo, principalmente este modelo se ocupa para que un usuario pueda ir buscando un determinado archivo, llegando a él de modo directo observando los distintos niveles y las distintas iteraciones de los nodos.

1.3 El modelo de red

Para lidiar con las limitaciones del modelo jerárquico, surge el nuevo modelo de bases de datos, construido bajo el modelo jerárquico, emergió en los 70's. El modelo de red mejora al modelo jerárquico permitiendo a los registros entrar en relaciones de múltiples padres-múltiples hijos.

El modelo de red tiene aún muchas desventajas frente al modelo jerárquico, pero también provee mucha mayor flexibilidad al permitir a los programadores navegar a través de los registros. A pesar de la flexibilidad, los desarrolladores deben aún programar una aplicación para navegar por este modelo. Además de que cualquier cambio en la base de datos puede resultar en severas actualizaciones muy complicadas. Una base de datos debe ser bien planeada desde un principio, tomando en cuenta la navegación entre los distintos registros en el nivel de aplicación.

1.4 El modelo Relacional

Debido a las desventajas que presentaba el modelo jerárquico y el de red, un nuevo modelo comienza a ganar fama a principios de los 70's, y para finales de los

80´s, surgió como el estándar de la próxima generación de bases de datos. El modelo de datos relacional representa una salida de las estructuras rígidas de los modelos de red y jerárquica. Las aplicaciones que accedan a la base de datos descansan o confían en una implementación definida de las bases de datos, y la estructura de la base de datos debe ser fuertemente codificada dentro de un lenguaje de programación. Si la base de datos cambia, la aplicación debe cambiar.

Sin embargo una base de datos relacional es independiente de la aplicación. Es posible modificar el diseño de la base de datos sin afectar la aplicación, debido a que el modelo relacional reemplaza la relación padre-hijo con una estructura basada en filas y columnas que forman tablas de datos relacionados. Como resultado, se pueden definir relaciones complejas entre las tablas, sin las restricciones de los modelos anteriores.

1.5 Bases de datos históricas

Las bases de datos históricas consisten en el almacenamiento de información que por lo general tiene más de diez años de antigüedad a partir de que se inicia su almacenamiento, principalmente se intenta almacenar documentos que no están digitalizados, para mantener un acervo informativo que contenga la mayor cantidad de documentos posibles y que se mantenga fiel a los documentos originales o bien que mantenga en esencia la misma información que estos. También es válida la información que se encuentra digitalizada, pero que no esté organizada de ninguna forma, más que el almacenamiento común, es así como se va recopilando la información y almacenando en una base de datos, para que se puedan realizar diversas consultas de manera casi inmediata y mantener los datos bien organizados.

1.6 Sistemas de administración de bases de datos

La mayoría de las bases de datos reposan en un sistema de administración de bases de datos para administrar los datos almacenados dentro del sistema de base de datos y hacer los datos disponibles para los usuarios quienes necesitan acceder a tipos específicos de información. Un DBMS (Sistemas de administración de bases de datos) está hecho de un completo arreglo de herramientas cliente-servidor que ayudan con varias tareas administrativas relacionadas con los datos. Por ejemplo, la mayoría de los DBMS proveen de algún tipo de herramienta que permite al cliente interactuar directamente con los datos almacenados en la base de datos.

Un DBMS debe almacenar datos y permitir que estos sean recuperados y modificados de manera que se protejan los datos contra operaciones que pudieran causar inconsistencias en la base o corromper los datos. Aunque por lo general la mayoría de los sistemas proveen muchas más capacidades. En general la mayoría de los DBMS completos proveen los siguientes tipos de funcionalidad:

- ✓ Administración de almacenamiento
- ✓ Mantenimiento de la seguridad
- ✓ Mantenimiento de los metadatos
- ✓ Administración de las transacciones
- ✓ Apoyo en la conectividad
- ✓ Optimización del rendimiento
- ✓ Proveer mecanismos de respaldo y recuperación
- ✓ Procesar peticiones de de recuperación y modificaciones

Estas características se extienden a cualquier DBMS, pero existen muchos más con funciones variadas y de naturaleza específica, para saber las funciones particulares provistas por algún DBMS en particular, hay que referirse a la documentación del mismo.

1.7 Los RDBMS

Al evolucionar los modelos de bases de datos, también lo hacen los productos DBMS que soportan varios tipos de bases de datos. No es de sorprender, entonces, que si existen DBMS, también existan los RDBMS. MySQL es un sistema de este tipo, como lo son Oracle, DB2, SQL Server y PostgreSQL. Estos productos, como cualquier DBMS, permiten acceder y manipular datos dentro de las bases de datos, protegerlos de la corrupción e inconsistencias, y mantener los metadatos necesarios para definir donde los datos que serán almacenados. La diferencia primaria entre los DBMS y los RDBMS es que estos últimos son específicos de las bases de datos relacionales. Soportan no solo el almacenaje de datos en estructuras de tablas, sino también las relaciones entre estas tablas.

1.7.1 MySQL

Surgiendo como uno de los más grandes jugadores en el mercado de los RDBMS está MySQL. Que como otros productos RDBMS, MySQL provee un amplio conjunto de características que soportan un ambiente seguro para almacenar, mantener y acceder a los datos. MySQL es rápido, confiable y una alternativa escalable de los muchos RDBMS comerciales que existen en la actualidad. A continuación se mencionan de manera general algunas de las características que se encuentran en MySQL:

- **Escalabilidad:** MySQL puede manejar grandes bases de datos, lo cual se ha demostrado con sus implementaciones en organizaciones como Yahoo!, Cox Communications, Google, Cisco, Texas Instruments, UPS, Sabre Holdings, HP y la prensa asociada. Incluso en la NASA y en los censos de Estados Unidos se han implementado soluciones MySQL. De acuerdo a la documentación MySQL, algunas de las soluciones empleadas por MySQL AB, la compañía creadora de MySQL, contiene más de 50 millones de registros, y algunos usuarios de MySQL han reportado que sus bases de datos contienen 60,00 tablas y 5 mil millones de columnas.

- **Portabilidad:** MySQL corre sobre una variedad de sistemas operativos, incluyendo Unix, Linux, Windows, QS/2, Solaris y MacOS, MySQL puede también correr sobre diferentes arquitecturas, desde las PC de escritorio hasta los grandes Mainframes.

- **Conectividad:** MySQL está totalmente orientado a las redes, soporta sockets TCP/IP, sockets Unix y las llamadas pipes. En adición, MySQL puede ser

- accesado desde cualquier lugar en internet, y múltiples usuarios pueden accesar a las bases de datos MySQL simultáneamente. MySQL además provee una gran variedad de interfaces para distintas aplicaciones de programación (APIs) para soportar la conectividad desde distintas aplicaciones escritas en lenguajes tales como C, C++, Perl, PHP, Java y Python.

- **Seguridad:** MySQL incluye un poderoso sistema de control de acceso a los datos. El sistema utiliza una estructura basada en el anfitrión(host) y el usuario que controla quien puede accesar a la información específica y el nivel de acceso a esa información. MySQL también soporta el protocolo de capa segura de sockets(SSL) para poder permitir conexiones encriptadas.

- **Velocidad:** MySQL fue desarrollado con la velocidad en mente. El monto de tiempo que toma a las bases de datos MySQL responder una petición de datos es tán rápido o más rápido que muchos de los otros RDBMS comerciales. El sitio

- **Facilidad de uso:** MySQL es fácil de instalar e implementar. Un usuario puede tener una instalación MySQL lista y corriendo, minutos después de descargar los archivos. Incluso en un nivel administrativo, MySQL es relativamente fácil de optimizar, especialmente comparado con otros productos RDBMS-

- **Código de fuente abierta:** MySQL hace que el código fuente de MySQL esté disponible para cualquier persona para descargarlo y ocuparlo. La filosofía

- ❖ de código fuente abierto permite a una audiencia global participar en la revisión, pruebas y desarrollo del código.

Como se puede observar MySQL es un RDBMS rápido y confiable que además implementa las ventajas y flexibilidad de los códigos de fuente abierta, es fácil de instalar e implementar, es gratuito y puede ser accesado desde cualquier lugar vía internet.

1.7.2 PostgreSQL

Es un DBMS que incorpora el modelo relacional para sus bases de datos y que se basa en el lenguaje estándar SQL. PostreSQL ha mostrado ser bastante capaz y confiable, tiene buenas características de rendimiento. Es un manejador multiplataformas, nativamente corre en UNIX, pero es capaz de correr en sistemas como Linux, freeBSD, y Mac OS X, también funciona en sistemas Windows NT/2000/2003 Server, o incluso en sistemas Windows XP. Además utiliza un código de fuente libre.

PostreSQL puede ser comparado favorablemente contra otros DBMS, pues contiene las mismas características que los demás DBMS comerciales, además de algunos extras que no se encontrarán en otros lados.

Las características de PostreSQL incluyen:

- Transacciones
- Subselecciones
- Vistas
- Llaves externas con integridad referencial
- Bloqueo sofisticado
- Tipos de usuarios definidos
- Herencia
- Reglas
- Control de concurrencia de múltiples versiones

Desde la versión 6.5, PostgreSQL se ha vuelto bastante estable, con cada gran serie de pruebas de regresión para asegurar una estabilidad muy superior en cada lanzamiento. A partir del lanzamiento 7.x se ha llegado a un acercamiento mayor que en ningún otro hacia lo que es el lenguaje SQL92 y una restricción en el tamaño

de las filas que fue removido. En lanzamiento de la versión 8 se han agregado características tales como:

- Versión nativa de Microsoft Windows
- Espacios de la tabla
- Habilidad para alterar los tipos de columna
- Recuperación en tiempo de punto

PostgreSQL ha demostrado su confiabilidad en el uso. Cada lanzamiento ha sido controlado muy cuidadosamente, y los lanzamientos beta han sido sujetos de prueba al menos una vez al mes. Con una comunidad de usuarios más grande, con acceso al código fuente, los errores y problemas en el funcionamiento son reparados muy rápidamente.

El rendimiento de PostgreSQL ha sido mejorado en cada lanzamiento, y las últimas pruebas de rendimiento contra otras marcas, muestran que, en algunas circunstancias, se compara muy bien contra los productos comerciales.
Una de las fortalezas de PostgreSQL radica en su arquitectura, y es que es debido a esta, que se puede manejar en un entorno cliente/ servidor, el cual beneficia tanto a los desarrolladores como a los usuarios. El corazón de PostgreSQL radica en las instalaciones de bases de datos en los procesos de servidor. El cual corre en un solo servidor. Las aplicaciones que necesitan acceder a los datos almacenados dentro de la base de datos que requieren hacer vía el proceso de bases de datos. Los programas de tipo cliente no pueden acceder a los datos directamente, incluso si están corriendo en la misma máquina como el proceso servidor.

Esta separación entre el cliente y el servidor permiten que las aplicaciones sean de tipo distribuidos. Se puede utilizar la red para poder realizar la separación de los

clientes de tu servidor y desarrollar aplicaciones de cliente en un entorno que se ajuste a las necesidades de los usuarios. Por ejemplo, se puede implementar la base de datos en UNIX y crear programas cliente que corren sobre Microsoft Windows.

Con PostgreSQL, puedes accesar a tus datos en formas distintas:

Usando una línea de comandos para ejecutar enunciados SQL.

Montar SQL directamente sobre tu aplicación.

Usar llamadas de funciones para preparar y ejecutar enunciado SQL, examinar el ajuste de resultados, y desarrollar actualizaciones de una gran variedad de lenguajes de programación diferentes.

Acceder a la base de datos en PostgreSQL indirectamente utilizando como ODBC o el estándar JDBC, o bien utilizar una biblioteca estándar como PERL DBI.

1.7.3 Microsoft SQL Server

El primer lanzamiento de SQL Server ocurrió en 1989, fue un evento no muy notable en las bases de datos, las demás DBM eran superiores a este.

Microsoft SQL Server 2000 fue, en contraste, el punto de partida o rival a vencer para los DBMS, al final de la década estaba listo para además de su desarrollo, dejar fuera del mercado a numerosos DBMS, considerando sus numerosas características:

Máximo tamaño de sus bases de datos de 1 000 000 de terabytes. Como ejemplo, se podrían almacenar 100 megas de cada mujer, hombre, niño y perro en el planeta en un simple servidor de bases de datos SQL Server.

Hasta 16 instancias simultaneas de SQL Server corriendo en una sola computadora.

Soporta hasta 32 procesadores corriendo sobre una sola instancia.

Soporta hasta arriba de 64 gb en RAM de memoria física.

Otras características son:

- Construido con soporte para Lenguaje extensible de marcas XML
- Vistas indexadas
- Integridad de cascada referencial
- Capacidad mejorada de solicitudes distribuidas
- Soporte de servicios en análisis de minería de datos

1.7.4 Oracle

El servidor Oracle tiene todas las características de un RDBMS y que tiene un soporte amplio para entornos sofisticados cliente/servidor. Muchas de las características internas de Oracle están diseñadas para proveer una alta disponibilidad, máximo rendimiento, seguridad y un uso eficiente de los de los recursos del cliente. Aunque estas características son arquitectónicamente importantes par un servidor de base de datos, Oracle también incluye características basadas en el lenguaje que aceleran el desarrollo y mejoran el rendimiento del lado del servidor.

- **Lenguaje PL/SQL:** Un gran componente de Oracle es su máquina de procesamiento (Lenguaje de Procedimientos). PL/SQL está diseñado específicamente para procesos clientes/servidor en los que se activa un programa para bloquear la lógica que contiene la aplicación así como los enunciados que serán enviados al servidor en una sola petición.

- **Procedimientos almacenados:** Oracle permite la capacidad de de almacenar bloques de PL/SQL como objetos dentro de la base de datos en forma de procedimientos almacenados, funciones, y paquetes de la base de datos. Las porciones lógicas de la aplicación, especialmente aquellas que requieren acceso a la base de datos, pueden residir en donde son procesadas(en el servidor). Usar procedimientos almacenados incrementa la eficiencia de los sistemas cliente/servidor significativamente.

- **Activadores de la base de datos:** Los activadores de la base de datos reensamblan los procedimientos almacenados que residen en los bloques PL/SQL de la base de datos; la diferencia entre los dos radica en que los activadores son disparados automáticamente por el kernel de la RDBMS en respuesta a que se cumpla un evento del tiempo (como alguna operación update, delete o insert)
- **Integridad declarativa:** Cuando se define una tabla en Oracle, se puede incluir una restricción de integridad como parte de la definición de la tabla. Las restricciones son forzadas por el servidor cuando se insertan, actualizan o borran registros. En adición a las restricciones integrales referenciales que fuerzan las las relaciones entre las llaves primarias y foráneas, también se pueden definir las propias restricciones del usuario para controlar los valores del dominio de las columnas individuales de la tabla.

- **Funciones definidas por el usuario:** También se encontrarán bloques PL/SQL de funciones definidas por el usuario. Estas son similares a los procedimientos almacenados y también reducen el monto de codificación de la porción del cliente en la aplicación. Estas funciones no solo se pueden

- llamar desde PL/SQL , sino que también se pueden extender al set estándar de las funciones Oracle SQL. Se pueden colocar funciones definidas por el usuario dentro de sentencias SQL justo como cualquier otra función de Oracle SQL.

1.8 Justificación del manejador de bases de datos

En nuestro caso utilizaremos el manejador de base de datos MySQL, dado que es un manejador del tipo fuente abierta, es decir gratuito y que permite la modificación al código. Ofrece las características generales que cualquier manejador comercial y es un manejador robusto para una gran cantidad de información como ya se mencionó en la introducción, así mismo ofrece gran seguridad y un buen soporte por parte de la comunidad MySQL, por lo que nos será de gran ayuda, es muy fácil de instalar así como de usar.

1.9 Consultas generales sobre las bases de datos

Algunas de las características que utilizaremos de este manejador son:
Mostrar las bases de datos:

show databases;

Mostrar las bases de datos seleccionadas:

select databases();

Mostrar las tablas que contiene una base de datos:

show tables;

Seleccionar una base de datos:

use nombre_base;

Describir la estructura de campos de una tabla:

describe nombre_tabla;

Crear una base de datos:

create database nombre_base;

Creación de una tabla:

create [temporary] table [if no exists] nombre_tabla (

nombre_campo tipo[1] opciones[2] cláusulas[3],

“ “ “ ,

“ “ “ ,

[último campo]);

[temporary] --> la tabla existirá mientras exista la conexión con el cliente actual o hasta que se emita la instrucción drop table.

[if no exist] --> si existe la tabla no se crea una nueva.

1.9.1 Las opciones de tipo de campo

tinyint --> 1 byte

smallint --> 2 byte

mediumint --> 3 byte

int --> 4 byte

bigint --> 8 byte

float --> 4 byte

double --> 8 byte

decimal --> variable

char(n) --> cadena de caracteres de longitud fija

varchar(n) --> cadena de caracteres de longitud variables

tinyblob --> objeto binario largo (muy pequeño)

blob --> objeto binario largo (pequeño)

mediumblob --> objeto binario largo (medio)

longblob --> objeto binario largo (grande)

tinytext --> cadena de texto muy pequeña

text --> cadena de texto pequeña

mediumtext --> cadena de texto media

longtext --> cadena de texto larga

enum --> una enumeración

set --> un conjunto

date --> valor fecha (aaaa-mm-dd)

time --> valor de hora (hh-mm-ss)

datetime --> valor de fecha y hora

timestamp --> valor de lapso de tiempo (aaaammddhhmmss)

year --> valor de año

1.10 Implementación Casos de uso

Los actores que se utilizarán a lo largo del desarrollo de los distintos modelos de diseño serán aquellos mencionados en los casos de uso del trabajo anterior, los cuales son:

AC-1 Gestor Superior: Este es el encargado de los aspectos de negocios que tendrán una significancia para el proyecto, sus decisiones afectarán a todo el proyecto, su viabilidad, requisitos, necesidades, identificación de procesos, elaboración de documentación, diseño de flujo de datos, archivos, tareas de diseño.

AC-2 Gestor de proyecto: Conforma, motiva, organiza y controla al equipo de trabajo, además coordina y dirige la totalidad del proyecto.

AC-3 Profesional: Es el encargado de proveer el conocimiento técnico necesario para la ingeniería de un producto o aplicación. Puede especificar lo requisitos para la ingeniería de software y otros elementos que tienen una menor influencia en el proyecto.

1.10.1 Caso de uso 1 (tabla CU-01)

ID:	**CU-01**
Nombre:	Configurar la BD
Descripción:	Este caso de uso permite establecer el ID y el PWD del superusuario, decidir la dirección donde se guarda la BD y el establecimiento de los datos del superusuario.
Evento que inicia el caso de uso:	Ejecución del programa.
Actores:	Gestor superior.
Precondiciones:	1. Conocer el ID, PWD y ubicación de la BD. 2. Conocer la configuración a establecer. 3. Conocer quienes pueden acceder a la BD.
Poscondiciones:	1. Creación del ID y el PWD del superusuario. 2. Conocimiento de la ubicación de la BD.
Flujo normal:	1. Se establece el ID de la BD. 2. Se establece el PWD de la BD. 3. Se establece el nombre y la ubicación de acceso de la BD.
Flujo alternativo 1:	1. Se elige eliminar la BD. 2. Se elimina la BD.
Excepciones:	(No aplica)
Incluye:	(No aplica)
Suposiciones:	(No aplica)
Notas:	(No aplica)

Fig.1.1 Tabla CU-01

Configurar la base de datos

En este caso de uso, lo que se pretende es establecer lo que será el nombre y contraseña del usuario maestro o super usuario.

Para la siguiente acción se asume que el servidor MySQL se instalará por defecto con las opciones clásicas, es decir, una instalación por default en archivos de programa, sin especificar una ruta distinta

Lo que se necesita para poder realizar esta acción es lo siguiente:

A partir del directorio en que se haya realizado la instalación de MySQL se deberá acceder a la base de datos de mediante los siguientes comandos (Utilizando el símbolo del sistema):

C:\cd program files

C:\Program Files\cd MySQL

C:\Program Files\MySQL\cd mysql server 5.1

C:\Program Files\MySQL\ MySQL Server 5.1\cd bin

C:\Program Files\MySQL\ MySQL Server 5.1\bin\mysql –u root –p

El password que aparece es por default nulo, esto es para que nosotros lo podamos configurar de acuerdo a nuestras necesidades.

En este caso para crear la contraseña para el usuario root, tecleamos la instrucción:

Mysqladmin –u root password NUEVACONTRASEÑA

En donde NUEVACONTRASEÑA será la palabra que utilicemos para entrar a MySQL, sugerimos utilizar “unamfi” como contraseña.

Ahora para crear un nuevo usuario con una nueva contraseña utilizamos:

CREATE USER ‘usuario1’ IDENTIFIED BY ‘usuario1’;

En donde la siguiente palabra de USER será el nuevo usuario y la sentencia que le sigue a la instrucción IDENTIFIED BY será la nueva contraseña, es decir, en esta única sentencia tenemos tanto el nombre del usuario como su contraseña, ahora solo falta controlar los privilegios de nuestro nuevo usuario, para esto ocupamos la sentencia:

GRANT ALL TO user@host identified by 'password';

GRANT ALL TO hugi@host identified by ‘user’;

Para lograr borrar una base de datos se ejecuta la siguiente sentencia:

DROP DATABASE nombredelabasededatos;

Con esto habremos removido de MySQL la base de datos en cuestión, que es lo que necesitamos en un determinado momento en que se abandone el flujo normal del sistema.

Es de hacer notar que se podría borrar al usuario root para tener únicamente a nuestro usuario, más por razones de seguridad es preferible contar con al menos dos usuarios y en este caso mantendremos al que acabamos de crear, así como el usuario que viene preestablecido.

1.10.2 Caso de Uso 2

ID:	CU-02
Nombre:	Dar generalidades del proyecto
Descripción:	Este caso de uso permite ingresar el nombre del proyecto y sus características esenciales, la declaración de los miembros del equipo y sus características, así como la asignación del gestor del proyecto.
Evento que inicia el caso de uso:	Alta de miembros del equipo
Actores:	Gestor superior.
Precondiciones:	1. Conocer el nombre del proyecto. 2. Conocer la localización del plan del proyecto. 3. Conocer las características principales del proyecto. 4. Conocer el nombre de los miembros de equipo y sus características. 5. Conocer el nombre del gestor del proyecto.
Poscondiciones:	1. Establecimiento del nombre del proyecto y sus características. 2. Establecimiento de los miembros del equipo y del gestor del proyecto, así como la definición de sus características.
Flujo normal:	1. Se establece el nombre del proyecto. 2. Se establecen las características principales del proyecto: inicio y fin del proyecto, tiempo de duración estimado, tiempo de duración real, objetivos, objetivos no logrados, costo estimado del proyecto, costo real del proyecto. 3. Se establece la ubicación de las partes del plan del proyecto o se señala que todo está contenido en un solo documento y se da la ubicación. 4. Se define ID y PWD de cada miembro del staff de la empresa. 5. Se describe cada miembro staff: nombre, horas que trabaja, habilidades, experiencia, temporalidad y monto del pago. 6. Se asigna el proyecto para cada miembro del staff de la empresa y se elige que el gestor del proyecto de entre los miembros del equipo del proyecto.
Flujo alternativo 1:	1. Se establece el nombre del proyecto. 2. Se establecen las características principales del proyecto: inicio y fin del proyecto, tiempo de duración estimado, tiempo de duración real, objetivos, objetivos no logrados, costo estimado del proyecto, costo real del proyecto. 3. Se establece la ubicación de las partes del plan del proyecto o se señala que todo está contenido en un solo documento y se da la ubicación. 4. Se describe cada miembro del staff de la empresa: nombre, horas que trabaja, habilidades, experiencia, temporalidad y monto del pago. 5. Se asigna el proyecto para cada miembro del staff de la empresa y se elige al gestor del proyecto como el gestor superior.
Flujo alternativo 2:	1. Se escribe el nombre del proyecto. 2. Se elimina el proyecto o se modifican los datos del proyecto o del gestor del proyecto.
Excepciones:	(No aplica)
Incluye:	(No aplica)
Suposiciones:	(No aplica)
Notas:	El gestor superior y el gestor del proyecto pueden ser el mismo sólo se tiene que indicar.

Fig.1.2 Tabla CU-02

El caso de uso número 2 CU-02 consiste en que se den las características generales de los proyectos o proyecto a ingresar al sistema.

Para este caso lo que se pretende es que ya se cuenten con varias características del proyecto, se comenzará por ingresar ciertos datos que serán verificados por el sistema mediante los distintos tipos de datos de que dispone MySQL.

Para comenzar, el proyecto necesitará un nombre el cual será del tipo de dato VARCHAR con una longitud de 30, este será el nombre del proyecto.

Para este tipo se creará una tabla especial llamada **CU-02**, la cual controlará algunas características importantes, tales como fecha de inicio del proyecto, fecha del fin del proyecto, tiempo estimado de la duración del proyecto, objetivos, objetivos no logrados, costo estimado del proyecto y costo real del proyecto, así como el mismo nombre del proyecto, por lo tanto nuestra tabla tendría que ser de la siguiente forma:

cu2id	Nombre Proyecto	Fecha inicio	Fecha fin	Tiempo estimado	Duración	Objetivos	Objetivos no logrados	Costo estimado	Costo real

Tabla cu2id

Para generar esta tabla se introduce la siguiente sentencia en Mysql:

Create table CU02(cu2id int auto_increment, nombreProyecto varchar(50), fechaInicio date, fechaFin date, tiempoEstimado time, duracion time,objetivos varchar(250), objetivosNoLogrados varchar(250), costoEstimado smallint, costoReal smallint, primary key (cu2id));

```
CREATE TABLE `test`.`hola` (
  `UsuaroInicial` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  `ApellidoPaterno` VARCHAR(45) NOT NULL,
```

```
  `ApellidoMaterno` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`UsuarioInicial`)
)
ENGINE = InnoDB;
```

Se ingresará la ruta del plan del proyecto, dentro de la base de datos, que será un tipo de datos del tipo VARCHAR de longitud de 50, en esta se especifica la localización de los distintos archivos que se utilizarán dentro del proyecto, o bien, se reúnen todo en un solo documento y este es el que se ingresará a la base de datos.

Se da un nombre de usuario y su password para cada uno de los miembros del equipo.

Se realiza la tabla de datos para cada miembro del equipo, con sus características, tales como:

Nombre, número de horas que trabaja, habilidades, experiencia, temporalidad y monto de pago.

De acuerdo a estas características se añade un gestor o administrador de proyectos que será elegido del mismo equipo, de acuerdo a las habilidades y experiencia de los mismos.

```
CREATE USER ‘usuario’  IDENTIFIED BY ‘usuario’;
Create table Usuarios(usrid int auto_increment, nombreusr  varchar(50), numhoras time, habilidades varchar(100), experiencia varchar(100), temporalidad time, montopago int, caracPrinc varchar(200),  primary key (usrid));
```

En la tabla se registrarán las características principales del proyecto, que serán del tipo de dato VARCHAR con una longitud de 200.

Se introducirán los nombres de los miembros del equipo así como las características de estos, en este caso también se ocuparán datos del tipo VARCHAR con una longitud de 300, para así poder además seleccionar a los miembros que participarán en determinado proyecto.
Adicionalmente se agrega el nombre del gestor o administrador del proyecto con datos del tipo VARCHAR con longitud de 60.

Para el flujo alternativo se manejan las mismas funciones, con la excepción de que en este caso no se asignarán nombres de usuario y passwords para cada uno de los miembros.

1.10.3 Caso de Uso 3

ID:	CU-03
Nombre:	Establecer Actividades
Descripción:	Este caso de uso permite establecer las partidas en las cuales se divide el trabajo, sus responsables y su status.
Evento que inicia el caso de uso:	(No aplica)
Actores:	Gestor del proyecto.
Precondiciones:	1. Haber llevado a cabo los CU-01 y CU-02. 2. Conocer las actividades a desarrollar de cada miembro del equipo del proyecto. Conocer la duración de la jornada laborar de cada integrante del equipo. 3. Conocer el inicio y término de cada actividad.
Poscondiciones:	1. Establecer las actividades
Flujo normal:	1. Se establecen las actividades y se declara el status de la actividad de acuerdo al nivel en la cual esté. 2. Se establecen las fechas de inicio y término de cada actividad. 3. Se selecciona al responsable principal y al de apoyo.
Flujo alternativo:	1. Se elige modificar datos. 2. Se selecciona que información se va a eliminar o modificar. 3. Se cambian los datos referentes a las actividades y sus encargados o bien se elimina alguna tarea.
Excepciones:	(No aplica)
Incluye:	(No aplica)
Suposiciones:	(No aplica)
Notas:	(No aplica)

Fig.1.3 Tabla CU-03

En el caso de uso número tres CU-03 se permitirán establecer las partidas en las que se divide el trabajo, sus responsables y el estado del mismo.

En este actúa el gestor del proyecto.

Para poder llevar a cabo este caso se necesitan haber realizado los casos CU1 y CU2, saber cuáles son las actividades que desarrollan cada uno de los miembros del equipo, así como su jornada laboral y el inicio y término de cada actividad.

Al terminar este caso de uso se deberán de establecer las actividades a realizar.

Se lleva a cabo una tabla dentro del manejador de bases de datos, que tendrá la siguiente estructura:

idCu3	Actividades	Status	iniAct	finAct	resPrin	resApoy

Tabla idCu3

Create table CU3(idCu3 int auto_increment, actividades varchar(200), status varchar(20), iniAct date, finAct date, resPrin varchar(50), resApoy varchar(50));
En caso de que se requiera modificar alguno de los campos, como el tipo de Status, se utilizará el siguiente comando (encontrándonos dentro de la base de datos):

UPDATE cu3 status='Terminado' WHERE status=’En proceso';

1. INSERT INTO TABLE (field1, field2) VALUES (value1, value2)

Los demás casos de uso que se tienen en el trabajo anterior responden más a las actividades teóricas y de organización para introducir proyectos de software, ya que se refiere a las métricas de software y a la elección de estas por parte de las personas que introducirán los datos en la base, es por esto que esos casos de uso no se utilizarán de manera explícita, ya que están bien indicados en el trabajo anterior.