UNIVERSIDAD JOSÉ CARLOS MARIÁTEGUI

VICERRECTORADO DE INVESTIGACIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

TRABAJO DE SUFICIENCIA PROFESIONAL

"JAVA Y BASE DE DATOS"

PRESENTADO POR:

BACHILLER ERICKA AMANDA BOCÁNGEL LAMAS

ASESOR:

ING. JULIAN MANUEL FLORES MANCHEGO

PARA OPTAR POR EL TÍTULO PROFESIONAL DE INGENIERA DE SISTEMAS E INFORMÁTICA

MOQUEGUA-PERÚ

2016

CONTENIDO

CAPÍTULO IV
CONCLUSIONES Y RECOMENDACIONES

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

RESUMEN

El presente trabajo de suficiencia profesional presenta una revisión teórico y práctico sobre el tema, con el fin de mostrar y conocer los fundamentos de Java, en cuanto a base de datos se usó un sistema de gestión de bases de datos relacional como es Mysql.

Para el desarrollo de la aplicación de control de laboratorio de cómputo se implementó en el IDE Netbeans, el cual fue elaborado esencialmente para el lenguaje de programación Java que brinda un entorno agradable e intuitivo. El sistema de gestión de base de datos es MySQL desarrollada con la herramienta SQLyog, la cual es una eficiente interfaz gráfica para MySQL.

Palabras clave: Aplicación, Java, MySQL, control.

ABSTRACT

The present work of professional proficiency presents a theoretical and practical review on the subject, in order to show and know the basics of Java, in terms of data base a relational database management system such as Mysql is used.

For the development of the computer lab control application was implemented in the Netbeans IDE, which was mainly made for the Java programming language that also provides a pleasant and intuitive environment. The database management system is MySQL developed with the SQLyog tool, which is an efficient graphical interface for MySQL.

Keywords: Application, Java, MySQL, control.

CAPÍTULO I

INTRODUCCIÓN

La fama de Java se basa en muchas de sus particularidades. Java es un lenguaje simple, o todo lo simple que puede ser un lenguaje orientado a objetos. Es un lenguaje autónomo de plataforma, un programa hecho en Java se ejecutará exacto en un PC con Windows que en un PC de altas prestaciones basada en Unix. Hay que recalcar su inmunidad, elaborar programas que ingresen indebidamente a la memoria o crear caballos de Troya es una labor solo de gigantes.

Corresponde indicar además su capacidad multihilo, su potencia. Pero es su simplicidad, portabilidad e inmunidad lo que le han conformado un lenguaje de tanto interés.

MySQL es un sistema de base de datos apoyado en el modelo relacional, multihilo y multiusuario. Multihilo es que el sistema designa automáticamente las labores a ejecutar entre los procesadores libres, mejorando el rendimiento. El nombre procede de la fusión de My con SQL. My era la hija del cofundador de la compañía iniciadora de la idea.

MySQL Es código libre, lo que implica que es gratis de usar y que se puede alterar. Su uso es muy amplio: desde sistemas gestores de contenidos tanto WordPress y Drupal, a grupos de compañías como Prisa. Es muy simple de aprender y usar, al ser muy intuitivo.

En el presente trabajo aparte de sus fundamentos, desarrolla una aplicación de Control de Laboratorios de Cómputo, la cual brinda una visión sobre parte del funcionamiento en Java y Base de datos (MySQL).

CAPÍTULO II

OBJETIVOS

2.1 Objetivo general

Mediante Fundamentos Teóricos y prácticos de Java y Base de datos que nos permite desarrollar en esta ocasión una aplicación para el control de laboratorios de cómputo.

2.2 Objetivos específicos

Conocimiento Teórico y práctico de Java y Base de datos (MySQL) para tener una base para el desarrollo de la aplicación.

Desarrollo de una aplicación que permita registrar equipos de cómputo e informes de equipos en mal estado o perdidas brindadas de la aplicación de Control de un Laboratorio de Cómputo.

CAPÍTULO III
DESARROLLO DEL TEMA

3.1 Marco teórico

3.1.1 Java

Java es un lenguaje de programación desarrollado por Sun Microsystems. Java fue anunciado en la segunda mitad del año 1995 y desde ese momento es un lenguaje de programación muy conocido. Java es un lenguaje muy valorado ya que los programas Java se pueden ejecutar en distintas plataformas con sistemas operativos (Windows, Mac OS, Linux o Solaris. James Gosling), el director del grupo de trabajo encargado de desarrollar Java, hizo real la promesa de un lenguaje independiente de la plataforma. Se buscaba diseñar un lenguaje que admitiera programar una aplicación una sola vez y que luego se pudiera ejecutar en diferentes ordenadores y sistemas operativos. Para obtener la portabilidad de los programas Java se usa un entorno de ejecución para los programas compilados. Este entorno se llama Java Runtime Environment (JRE). Es gratis y está disponible para los principales sistemas operativos. Esto confirma que el mismo programa Java pueda ejecutarse en Mac OS, Windows, Linux o Solaris (Martínez, 2015, p. 2).

“Write Once, Run Anyware”, que se traduciría como “programar una sola vez y luego ejecutar los programas en diferentes sistemas operativos”, era el propósito del grupo de desarrollo de Java. Esto resume el concepto de portabilidad. Los programas Java son portables, quiere decir que son independientes de la plataforma, ya que pueden ejecutarse en cualquier PC o dispositivo móvil, individualmente del sistema operativo que esté instalado: Un programa Java puede ser ejecutado en una PC de escritorio, PC portátil, tableta, teléfono, reproductor de música o en cualquier otro dispositivo móvil con cualquier S.O. (Martínez, 2015, p. 2-3).

Figura 1. Java puede ejecutarse en cualquier S.O.

Fuente: Martínez, 2015

3.1.2 Plataforma de java

Los programas Java se compilan a un lenguaje intermedio, llamado Bytecode. El mismo código es traducido por la máquina virtual de Java del entorno de ejecución (JRE), de esta manera se consigue la portabilidad en diferentes plataformas. El JRE es una parte intermedia entre el código Bytecode y los diferentes sistemas operativos que hay en el mercado. Un programa Java compilado en Bytecode se puede ejecutar en sistemas operativos como Linux,

Windows, Solaris, Mac Os, BlackBerry OS, Android o iOsn empleando el entorno de ejecución de Java (JRE) adecuado (Martinez, 2015, p. 3).

Una de las particularidades más resaltantes de los lenguajes de programación actuales es la portabilidad. Como se ha mencionado antes, un programa es portable cuando es independiente de la plataforma y puede ejecutarse en distintos sistemas operativos y dispositivo físico. Los programas Java son portables porque se ejecutan en distintas plataformas. Pasa algo semejante con los ficheros o las fotografías PDF. Las fotografías con formato JPEG son portables ya que un archivo JPEG lo podemos ver con diferentes visores de fotografías y en dispositivos como PC'S, tabletas o teléfonos. El formato JPEG es un estándar para guardar archivos de imagen. Todas las imágenes JPEG tienen el mismo formato y los visores de fotos están hechos para visualizar las imágenes con este formato. De manera parecida, los archivos PDF (Formato de documento portátil) son portables. El formato PDF fue elaborado por Adobe Systems con la idea de que estos archivos se visualicen en distintos dispositivos que tengan instalado Adobe Acrobat Reader, el software de visualización de documentos PDF (Martínez, 2015, p. 3).

La portabilidad de Java ha ayudado a que varias empresas hayan desarrollado sus sistemas de comercio electrónico y sistemas de información en Internet con Java. El proceso de desarrollo y de mantenimiento de los sistemas es menos costoso y las aplicaciones son compatibles con diferentes sistemas operativos (Martínez, 2015, p. 3).

El avance del lenguaje de programación Java ha sido muy rápida. La plataforma de desarrollo de Java, llamada Java Development Kit (JDK), se ha

ido incrementando y cada vez integra a una cantidad mayor de programadores en todo el mundo. Java no solo es un lenguaje de programación. Java es un lenguaje, una plataforma de desarrollo, un entorno de ejecución y un conjunto de librerías para desarrollo de programas sofisticados. Las librerías para desarrollo se denominan Java Application Programming Interface (Java API) (Martinez, 2015, p. 3).

La siguiente imagen presenta los componentes de la plataforma Java, desde el código fuente, el API de Java, el compilador, los programas compilados en Bytecode y el entorno de ejecución de Java. Este entorno de ejecución (JRE) y la máquina virtual (JVM) hacen que un programa compilado Java se ejecute en diferentes sistemas operativos (Martinez, 2015, p. 4).

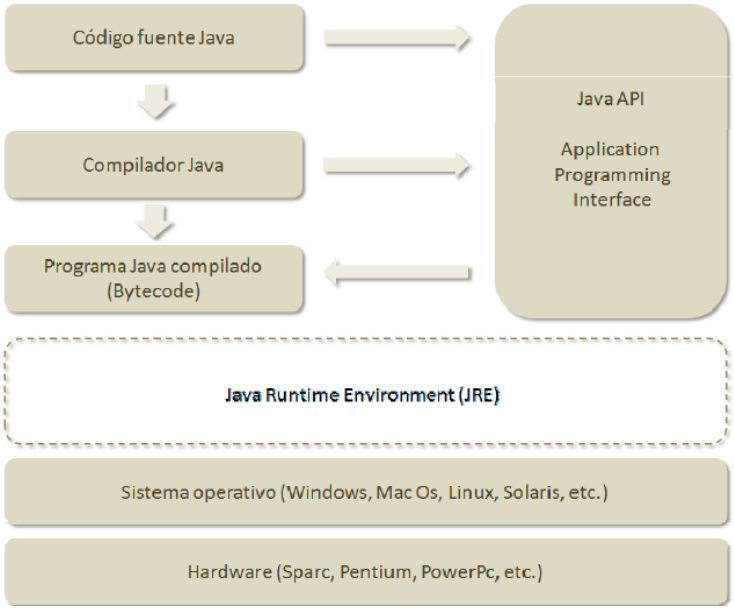


Figura 2. Elementos de la plataforma de Java

Fuente: Martínez, 2015

3.1.3 Entornos de desarrollo para java

Existen distintos entornos de desarrollo de aplicaciones Java. Este tipo de productos ofrecen al programador un entorno de trabajo integrado para facilitar el proceso completo de desarrollo de aplicaciones, desde el diseño, la programación, la documentación y la verificación de los programas. Estos productos se denominan IDE (Integrated Development Environment). Existen entornos de distribución libre como: NetBeans, Eclipse o BlueJ. Entre los productos comerciales están JBuilder o JCreatorPro. Para utilizar un entorno de desarrollo es necesario instalar el Java Runtime Environment (JRE) apropiado para el sistema operativo. El JRE se descarga de la página de Oracle Java (Martínez, 2015, p. 4-5).

3.1.4 El proceso de edición y compilación

En Java, al igual que en otros lenguajes de programación, se sigue el siguiente proceso: edición del código fuente, compilación y ejecución. Los programas Java se desarrollan y se compilan para obtener un código denominado Bytecode que es interpretado por una máquina virtual de Java (Java Virtual Machine) (Martínez, 2015, p. 5).

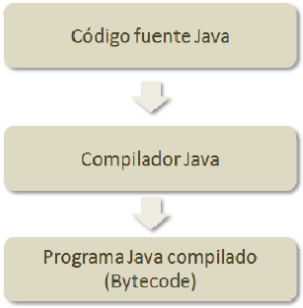


Figura 3. Proceso de Edición, Compilación y Ejecución

Fuente: Martínez, 2015

Según Martínez (2015) "**La edición** del programa fuente se elabora escribiendo el programa Java en un editor de texto como el Bloc de notas o usando un entorno integrado de desarrollo. El código fuente se guarda en un fichero de tipo .java" (p. 5).

La compilación se efectúa con el compilador Java javac o usando un entorno integrado de desarrollo. Mientras el proceso de compilación comprueba que el código fuente satisface la definición léxica, semántica y sintáctica de Java. Esto quiere decir que el compilador verifica que el código fuente está formado por palabras válidas en Java y que los comandos Java tienen la forma sintáctica apropiada. Si mientras el proceso de compilación el compilador encuentra los errores que ha incurrido el programador y le anuncia de los problemas que ha hallado para que pueda subsanarlos. Si durante la compilación no se encuentran errores, se crea un fichero de tipo class en Bytecode. Ya terminado la fase de compilación se puede ejecutar el programa. Para esto, es indispensable que la máquina virtual de Java traduzca el código Bytecode y ejecute la aplicación (Martínez, 2015, p. 5).

3.1.5 La codificación de programas java

El forma de programación o codificación de los programas Java es muy importante. La legibilidad de un programa describe en buena medida que se haya desarrollado adecuadamente y que el resultado final sea eficiente.

Legibilidad > Corrección > Eficiencia (Martínez, 2015, p. 6).

a. Legibilidad.

Un programa Java debe ser sencillo de leer y comprender, inclusive para una persona que no ha intervenido en su desarrollo. La legibilidad es un parte muy

valiosa porque permite el mantenimiento del software, la modificación de la funcionalidad de la aplicación la corrección de errores con menor coste (Martínez, 2015, p. 6).

b. Corrección.

Un programa debe hacer lo que debe hacer, ni más, ni menos. Esto es lo que se comprende por corrección. Un programa debe satisfacer estrictamente los requerimientos funcionales y técnicos de la fase de especificación. En la fase de prueba se revisa que el programa marcha adecuadamente y que satisface los requisitos funcionales y técnicos (Martínez, 2015, p. 6).

c. Eficiencia.

La eficiencia es el tiempo que un programa demora en ejecutarse y a los recursos que gasta. Cuanto más veloz sea un programa y use menos disco duro o memoria, el diseño es superior. La eficiencia no es un inconveniente que estar pendiente cuando se aprende a programar. Ahora lo mejor es usar los mecanismos de optimización propios de los compiladores. La eficiencia se debe comparar solo cuando un programa trabaja adecuadamente y cumple con las condiciones técnicas determinadas (Martínez, 2015, p. 6).

3.1.6 El proceso de desarrollo de software

Según Martínez (2015) "El proceso de desarrollo de los programas Java no es diferente de la gran parte de los lenguajes de programación. Es indispensable seguir una serie de pautas para desarrollar apropiadamente un producto software" (p. 6).

Según Martínez (2015) "La Ingeniería del Software estudia los diferentes pasos de desarrollo de software. El IEEE precisa Ingeniería del Software como la aplicación disciplinada, sistemática y cuantificable de un proceso de desarrollo, operación y mantenimiento de un producto software" (p. 6).

Según Martínez (2015) "Las fases clásicas de desarrollo de software es extensamente usado por su simplicidad. Este proceso se consituye de las siguientes fases: especificación, diseño, codificación, prueba y mantenimiento" (p. 7).

a. Especificación.

En esta etapa se dispone la funcionalidad, las propiedades técnicas de una aplicación y sus cualidades de uso. En esta fase es indispensable contestar a las posteriores preguntas:

¿Para qué se va a emplear la aplicación?

¿Cuáles son los requisitos funcionales de los usuarios?

¿Cuál es el perfil de los usuarios de la aplicación?

¿En qué plataforma correrá la aplicación?

¿Cuáles son sus condiciones de operación?

¿Cómo se va a emplear? (Martínez, 2015, p. 7).

b. Diseño.

En esta etapa se emplea toda la información recabada en la etapa de especificación y se sugiere una solución que satisfaga a los requisitos del usuario y pueda desarrollarse. En esta etapa se determina la arquitectura de la aplicación. Es indispensable especificar la estructura y la organización del programa y cómo se vinculan las diferentes partes de la aplicación (Martínez, 2015, p. 7).

c. Codificación.

Según Martínez (2015) "Esta etapa se basa en la programación en Java de las especificaciones de diseño de la anterior etapa. Durante esta etapa de codificación o implementación se fijan reglas de programación para ayudar la legibilidad de los programas Java" (p. 7).

d. Prueba.

En esta etapa se compila y se corre la aplicación para comprobar que satisface con los requisitos funcionales y técnicos detallados en la etapa de especificación. Si el programa no satisface con todos los requerimientos, puede ser por errores de diseño o de programación. En tal sentido, es indispensable subsanar los errores que se hayan localizado y hacer de nuevo la fase de diseño y codificación. Durante la etapa de prueba se comprueba que la aplicación cumple con los pautas de calidad dispuestos en el proyecto: facilidad de uso, eficiencia, corrección, integridad, flexibilidad, fiabilidad, facilidad de prueba, facilidad de mantenimiento, capacidad de reutilización, portabilidad e interoperabilidad (Martínez, 2015, p. 7).

Según Martínez (2015) "Una vez que la aplicación se ha demostrado y satisface con los requisitos dispuestos, se pone en marcha y empieza la etapa de operación para que sea usada para el fin que ha sido desarrollada" (p. 7).

e. Mantenimiento.

Una vez que la aplicación se ha puesto en marcha da inicio a la etapa de mantenimiento. En esta etapa se subsanan errores de funcionamiento de la aplicación, se cambia la funcionalidad o se agregan las nuevas funcionalidades que solicitan los usuarios. La etapa de mantenimiento es la de más durabilidad,

pues podrían pasar muchos años desde el comienzo de la operación hasta que el producto es retirado (Martínez, 2015, p. 8).

3.1.7 ¿Cómo se desarrolla un programa en java?

Java no tiene un editor propio, sin embargo, existen IDE's que permiten desarrollar aplicaciones de forma sencilla, por ejemplo: NetBeans, Eclipse, JBuilder, JDeveloper. Pueden desarrollarse aplicaciones usando el block de notas, para ello es necesario instalar el JDK (Java Development Kit), o Kit de desarrollo de Java.

El JDK puede ser descargado de la página de Sun (que recientemente ha sido adquirido por Oracle), http://netbeans.org/downloads/index.html

Aplicaciones de consola con JDK

Luego de instalar el JDK, se va crear una carpeta dentro de archivos de programa con la siguiente ruta c:\Archivos de programa\Java\jdk1.8.0_65, dentro de las que existen la carpeta bin, y donde se encuentran dos archivos muy importantes para la compilación (creación del código interpretado) y la ejecución de programas: el javac y java.

a. Javac: Es un programa que realiza la compilación del programa y la generación del código interpretado de extensión class.

b. Java: Es el archivo que permite la ejecución del programa de java de extensión class.

Ejemplo de Aplicación de Java a nivel consola:

```
import java.io.*;
```

```
public class suma
{
	public static void (String [] args)
	{
		InputStreamReader isr  = new InputStreamReader(System.in);
		BufferedReader br = new BufferedReader (isr);
		try
		{
			System.out.print("Sumando 1 : ");
			int s1 = Integer.parseInt(br.readLine());
			System.out.print("Sumando 2 : ");
			int s2 = Integer.parseInt(br.readLine());
			int suma=s1+s2;
			System.out.println ("La suma es " + s1 + "+" + s2 +"="+
suma);
		}
		catch (Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

3.1.8 La clase math en java

La clase Math representa la librería matemática de Java. Las funciones que contiene son las de todos los lenguajes, parece que se han metido en una clase solamente a propósito de agrupación, por eso se encapsulan en Math, y lo mismo sucede con las demás clases que corresponden a objetos que tienen un tipo equivalente (Character, Float, etc.). El constructor de la clase es privado, por lo que no se pueden crear instancias de la clase. Sin embargo, Math es public para que se pueda llamar desde cualquier sitio y static para que no haya que inicializarla (Froufe, 1999, párr. 1).

a. Funciones matemáticas

Si se importa la clase, se tiene acceso al conjunto de funciones matemáticas estándar:

Math.abs(x) para int, long, float y double

Math.sin(double)

Math.cos(double)

Math.tan(double)

Math.asin(double)

Math.acos(double)

Math.atan(double)

Math.atan2(double,double)

Math.exp(double)

Math.log(double)

Math.sqrt(double)

Math.ceil(double)

Math.floor(double)

Math.rint(double)

Math.pow(a,b)

Math.round(x)	para double y float
Math.random()	devuelve un double
Math.max(a,b)	para int, long, float y double
Math.min(a,b)	para int, long, float y double
Math.E	para la base exponencial
Math.PI	para PI (Froufe, 1999, párr. 2).

He aquí un ejemplo, Mates.java , de uso de algunas funciones de la clase Math:

```
class Mates {
public static void main( String args[] ) {
int x;
 double rand,y,z;
 float max;
rand = Math.random();
 x = Math.abs( -123 ); y = Math.round( 123.567 );
 z = Math.pow( 2,4 );
 max = Math.max( (float)1e10,(float)3e9 );
System.out.println( rand );
System.out.println( x );
System.out.println( y );
System.out.println( z );
```

System.out.println(max);

} } (Froufe, 1999, párr. 3).

3.1.9 Entorno de desarrollo integrado de Netbeans

Netbeans es un IDE(Entorno de Desarrollo Integrado), desarrollado por Sun Microsystem, tiene varias versiones. actualmente estamos sobre la 8.1, y varias ediciones, una edición compacta donde solo se pueden desarrollar aplicaciones elementales en Java, una edición completa y una edición full.

Se llama IDE, pues contiene componentes y elementos que permiten desarrollar de forma fácil y rápida una aplicación en java (estándar, empresarial, móvil o web), así como en otros lenguajes como c++, PHP y Ruby. Podemos observar la ventana principal del NetBeans:

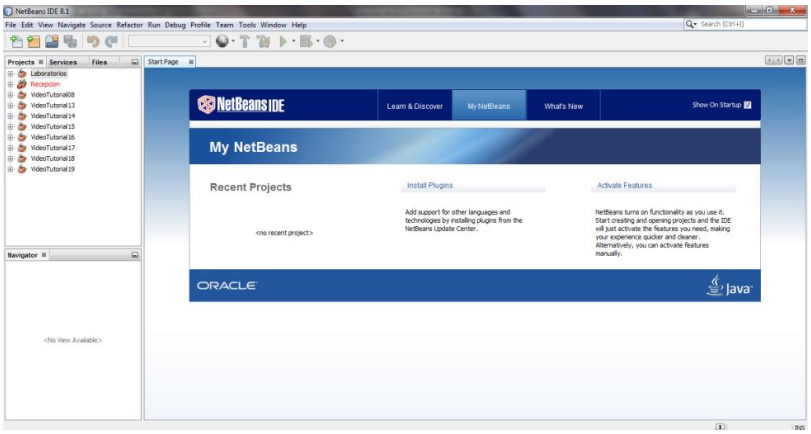


Figura 4. Ventana Principal del Netbeans

3.1.10 Creación de proyecto de java en Netbeans.

Una aplicación se crea o desarrolla a través de un proyecto el mismo que está conformado por un grupo de paquetes (package), y los mismos que agrupan a un grupo de clases (class).

El procedimiento para crear un proyecto contiene 2 pasos:

Primero: Debe seleccionar la categoría y tipo de proyecto, como se observa en la figura tenemos aplicaciones estándar "Java" (consola y de escritorio), aplicaciones web "Java Web", aplicaciones empresariales "Java EE" y aplicaciones moviles "Java ME"

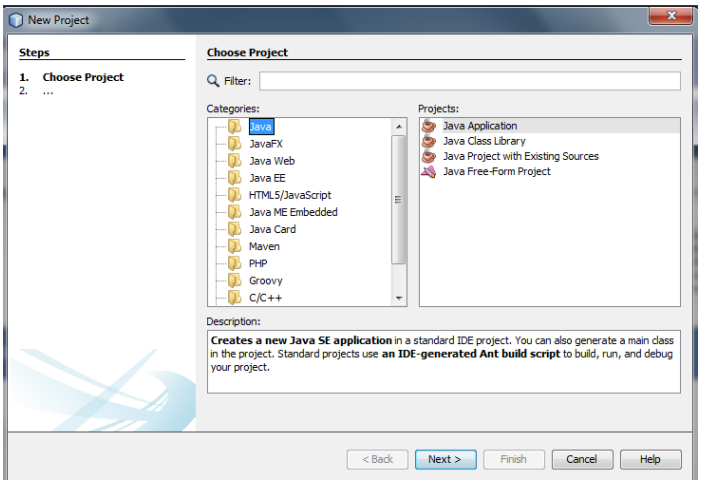


Figura 5. Selección de Categoría y tipo de proyecto

a. Creando una aplicación de consola

Para crear una aplicación de consola se deben seleccionar la **Categoría Java, y Java Application**. Todo proyecto debe tener un nombre, el mismo que se guardará en una carpeta cuyo nombre es idéntico al del proyecto creado. Pero además debe tener por lo menos, una clase principal, el mismo que contiene al procedimiento "void main", que es el primero en ser ejecutado:

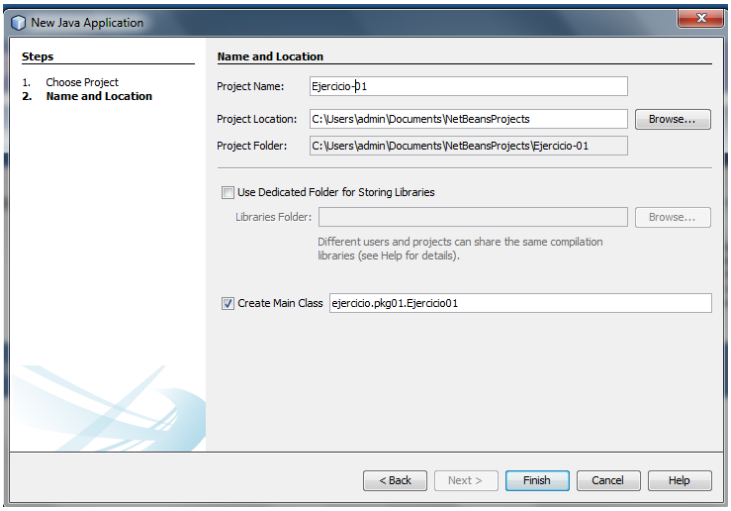


Figura 6. Creación de una aplicación de consola

Luego de haber realizado los dos pasos anteriores, podemos ver el entorno de desarrollo de NetBeans, para las aplicaciones de consola:

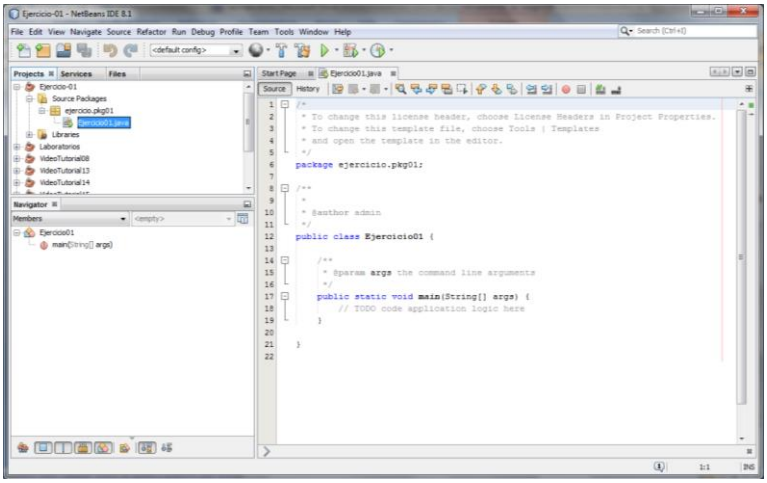


Figura 7. Ventana de Entorno de Desarrollo de Netbeans

b. Creando una aplicación de escritorio

Para crear una aplicación de escritorio, luego de haber creado una Aplicación Java, debe seleccionar las opciones que observa a continuación:

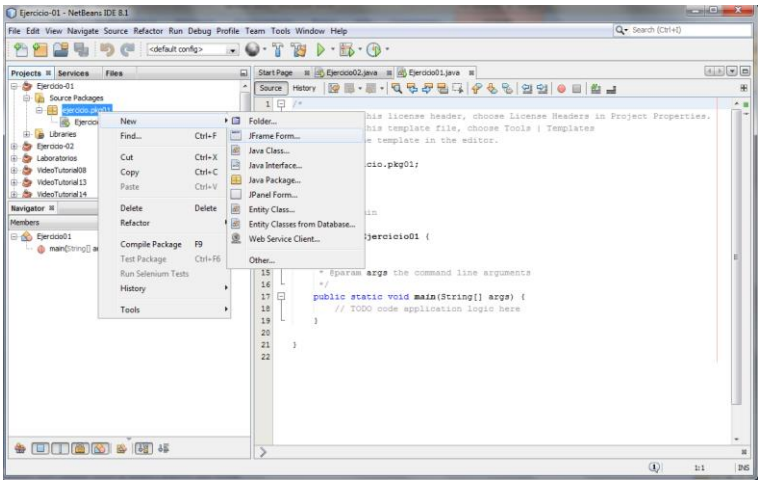


Figura 8. Creación de Aplicación de Escritorio

Luego de presionar en JFrame Form para continuar con el siguiente paso:

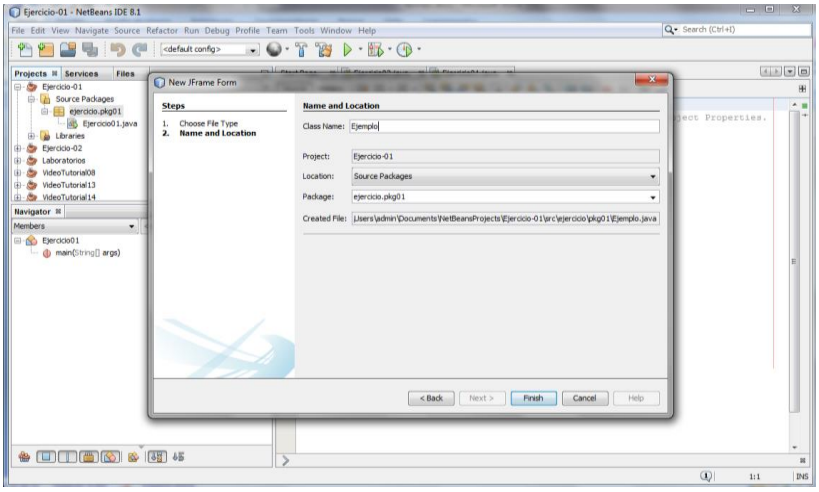


Figura 9. Ventana de especificación del nombre de clase.

En ésta ventana se debe especificar el nombre de la clase. Finalmente presionar Terminar, con lo cual podemos ver el siguiente entorno:

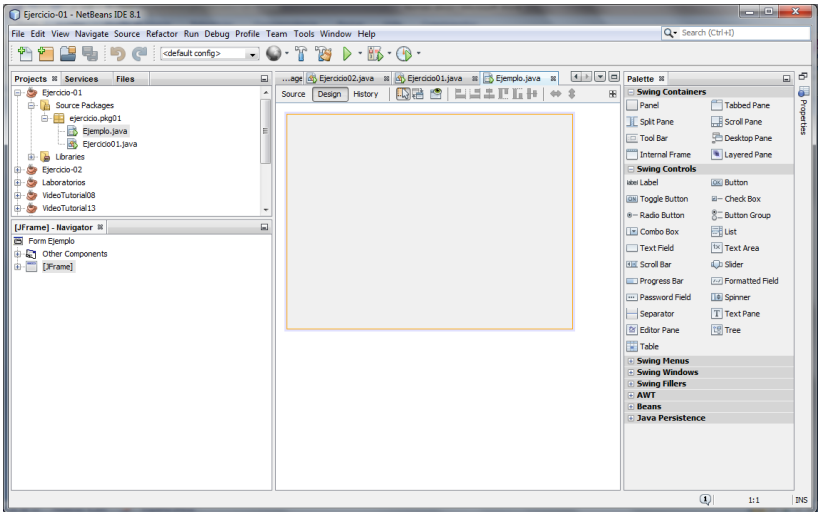


Figura 10. Ventana de desarrollo de Aplicaciones de Escritorio de Netbeans.

Esta ventana es el entorno de desarrollo de aplicaciones de escritorio en NetBeans, y contiene los siguientes elementos principales:

a. Ventana de Proyectos

b. Paleta de Controles

c. Ventanas Navegador/Insector

d. Ventana de Propiedades

e. Ventana de Formulario/Código

3.1.11 Estructuras secuenciales

Estructura de un programa en java

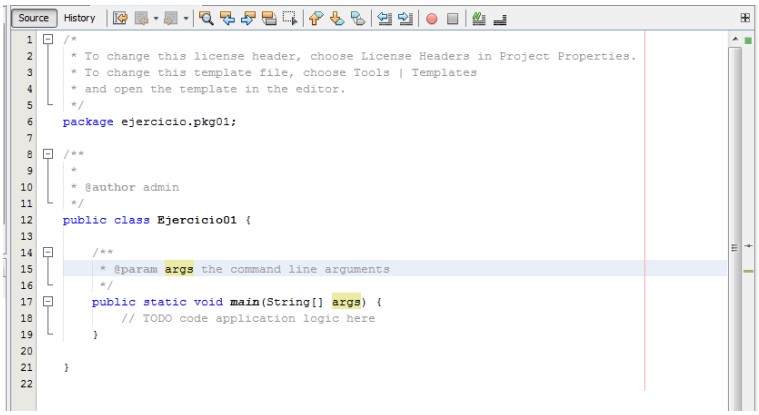


Figura 11. Estructura de un programa en java

a. Tipos de datos primitivos

En JAVA los tipos de dato básicos o simples o primitivos se clasifican en numéricos, caracteres o de texto y booleanos o lógicos. A continuación, se ilustra en una tabla los tipos de dato simple que maneja JAVA, el tamaño que ocupa en memoria y el rango de valores que puede tomar. Recuerde que un Byte equivale a 8 bits y que un bit puede ser un valor 0 ó 1. (Padilla, 2013, Datos en Java, párr. 1).

Tabla 1

Tipos de datos primitivos y sus rangos

Tipos de variable	Bytes que ocupa	Rango de valores
Boolean	2	True, false
Byte	1	-128 a 127
Short	2	-32 768 a 32 767
Int	4	-2 147 483 648 a 2 147 483 649
Long	8	$-9X10^{18}$ a $9X10^{18}$
Double	8	$-1,79X10^{308}$ a $1,79X10^{308}$
Float	4	$-3,4X10^{38}$ a $3,4X10^{38}$
Char	2	Caracteres(en Unicode)

Fuente: Padilla, 2013

b. Instrucciones básicas de entrada/salida

Imprimir en pantalla

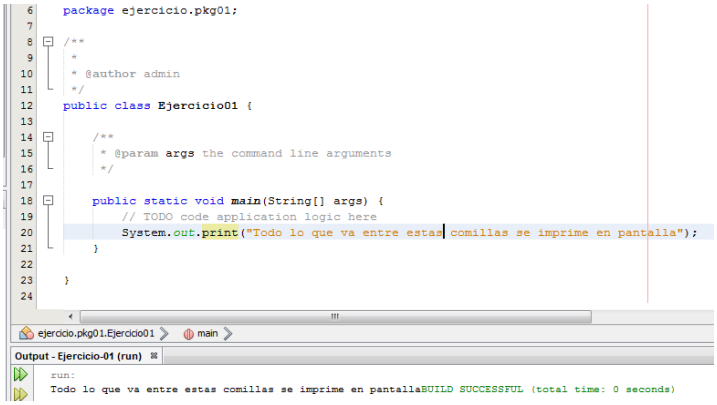


Figura 12. Imprimir en pantalla.

Entrada de datos por teclado

Clase BufferedReader

La clase BufferedReader permite crear un objeto que lee datos del teclado en forma de cadena de texto.

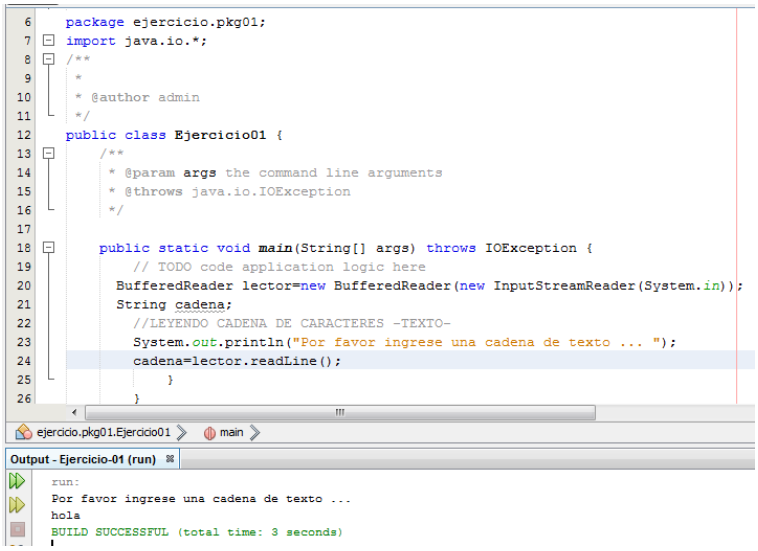


Figura 13. Entrada de datos por teclado.

c. Instrucciones de entrada/salida con JOptionPane

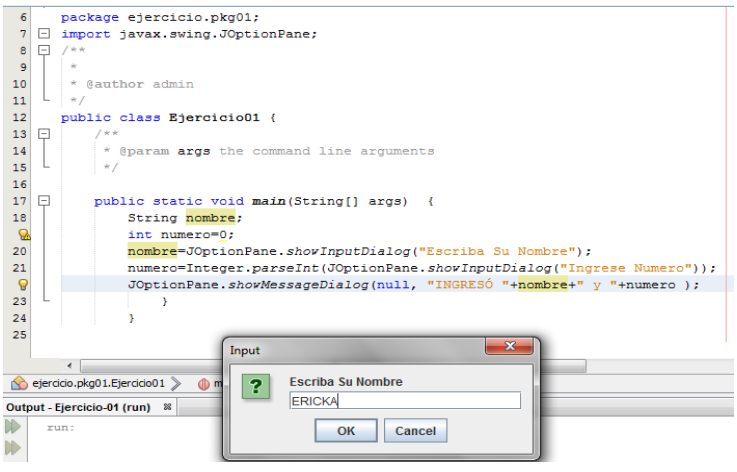


Figura 14. Entrada con JOptionPane.

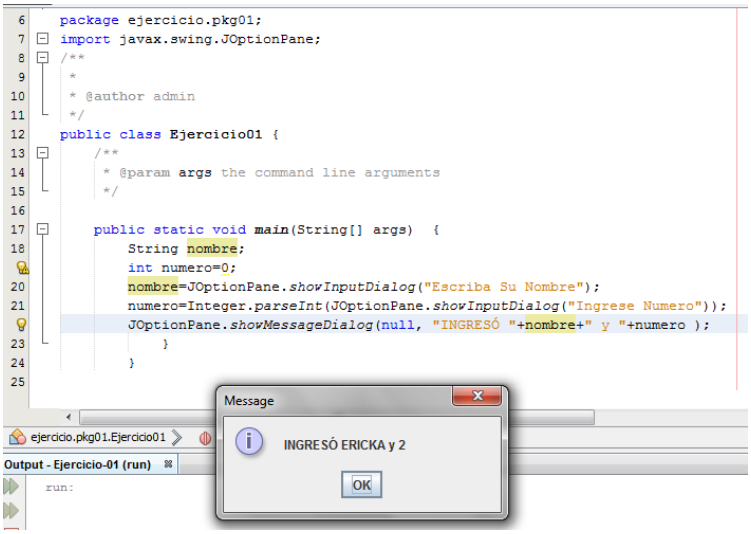


Figura 15. Salida con JOptionPane.

3.1.12 Estructuras de decisión

Operadores

Padilla (2013) afirma:

"Cada línea de código en programación también se conoce como sentencia y éstas sentencias en muchas ocasiones representan cálculos u operaciones que transforman datos. Todas las operaciones realizadas en el programa se escriben por medio de expresiones" (Operadores, párr. 1).

Padilla (2013) afirma:

a. Operadores Aritméticos

"En programación se usa uno para obtener el residuo de una división y se llama mod o residuo y se representa con el símbolo de porcentaje %" (Operadores, párr. 2).

Tabla 2

Operadores Aritméticos

Operador en Java	Significado
+	Suma
–	Resta
*	multiplicación
/	división
%	Residuo

Fuente: Padilla, 2013

El operador / funciona de diferente manera si trabaja con datos de tipo entero o de tipo flotante. Con datos de tipo flotante funciona de la manera tradicional; pero al realizarse una división entre dos números enteros, el operador / regresa el cociente de la división entera; es decir, regresa la parte entera del resultado (si hay fracción la elimina) (Padilla, 2013, Operadores, párr. 3).

Existen otros tipos de operadores que pueden de alguna manera ahorrarnos algo de código:

b. Operadores incrementales:

"++" : Incremento en 1 de la variable

"--" : Decremento en 1 de la variable

c. Operadores de asignación:

Tabla 3

Operadores de Asignación

Operador en Java	Significado
=	Igual
+=	Mas igual
*=	Por igual
-=	Menos igual
/=	Dividido igual
%=	Residuo igual

Fuente: Padilla, 2013

Ejemplo donde podemos apreciar el resultado luego de la aplicación de estos operadores, fijémonos bien en las variables utilizadas, y los resultados devueltos:

```java
package ejercicio.pkg01;
public class Ejercicio01 {

    public static void main(String[] args)  {
        int num1=5, num2=8, num3=10;

        num1++;
        System.out.println("resultado 1 con ++: "+num1);

        num2--;
        System.out.println("resultado 2 con --: "+num2);

        num1+=4;
        System.out.println("resultado 3 con +=: "+num1);

        num2-=2;
        System.out.println("resultado 4 con -=: "+num2);

        num3*=3;
        System.out.println("resultado 5 con *=: "+num3);

        num3/=2;
        System.out.println("resultado 6 con /=: "+num3);

        num3%=4;
        System.out.println("resultado 7 con %=: "+num3);
            }
        }
```

Figura 16. Ejemplo de aplicación de operadores

El resultado del código es:

```
ejercicio.pkg01.Ejercicio01 > main >
Output - Ejercicio-01 (run)
run:
resultado 1 con ++: 6
resultado 2 con --: 7
resultado 3 con +=: 10
resultado 4 con -=: 5
resultado 5 con *=: 30
resultado 6 con /=: 15
resultado 7 con %=: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figura 17. Resultado de aplicaciones de operadores

d. Operadores relacionales

Padilla (2013) afirma:

"Los operadores relacionales sirven para hacer comparaciones y de ellos se obtiene como resultado un valor de verdad. A diferencia de los operadores aritméticos donde obtenemos un número como resultado" (Operadores, párr. 4).

Tabla 4

Operadores Relacionales

Operador en Java	Significado
>	Mayor
<	Menor
>=	Mayor o igual
<=	Menor o igual
==	Igual
!=	Diferente

Fuente: Padilla, 2013

Suponga que se quiere saber si el valor que contiene una variable supera una cantidad determinada. Por ejemplo, en una solución donde queremos saber si el usuario del programa es mayor de edad, se captura su edad en una variable y se averigua si supera los 18 años.

edad >= 18;

Dependiendo del valor que esté almacenado en la variable edad, esta expresión dará como resultado FALSO o VERDADERO. Todos los operadores relacionales dan como resultado un valor de verdad. (Padilla, 2013, Operadores, párr. 5-6).

e. Operadores lógicos

Padilla (2013) afirma:

"En lógica formal existen de operadores para formalizar razonamientos y enlazar o conectar premisas. En programación de aquellos operadores mencionados solo se utilizan tres. La conjunción, la disyunción y la diferencia o negación"(Operadores, párr. 7).

Tabla 5

Operadores Lógicos

Operador en Java	Significado
&&	AND … Y
||	OR … O
!	NOT … NO

Fuente: Padilla, 2013

El operador para la conjunción (Y) se realiza con doble ampersant, la disyunción con barra vertical u operador de canalización y la negación se realiza con el operador signo de admiración hacia abajo. Estos operadores pueden unir expresiones completas, dan como resultado un valor de verdad FALSO o VERDADERO y es muy aconsejable el uso de paréntesis para apoyar la construcción de las expresiones evaluadas (x > 18) && (x < 25);

Se averigua si x es mayor que 25 y al mismo tiempo menor que 25. Algunas generalidades de estos operadores:

La conjunción es verdadera solamente si las expresiones evaluadas son verdaderas

La disyunción es verdadera si alguna de las expresiones evaluadas es verdadera, solamente es falsa si todas las expresiones evaluadas son falsas

La negación es un operador unario. Es decir que opera sobre un único operando. A diferencia de los anteriores que exigen al menos dos operandos.

En todos los casos los operandos pueden ser expresiones compuestas por más de una variable, como en el caso del ejemplo arriba (Padilla, 2013, Operadores párr. 8-9).

f. Estructura de decisión simple - If

Esta estructura permite la ejecución de un bloque de código siempre que se cumpla una condición (simple o compuesta), previamente evaluada, opcionalmente puede ejecutarse otro bloque (else) siempre que la condición no se cumpla, podemos observar la sintaxis de la instrucción:

if(<condición>)

{

sentencias si condición es true>;

}

else

[{

<sentencias si condición es false>;

}]

Ejemplo:

```
6   package ejercicio.pkg01;
7
8   import javax.swing.JOptionPane;
9   /**
10   *
11   * @author admin
12   */
13  public class Ejemplo02 {
14      public static void main(String[]args)
15      {  int num;
16      num=Integer.parseInt(JOptionPane.showInputDialog("Ingrese el Número: "));
17      if (num%2==0)
18          JOptionPane.showMessageDialog(null,"El numero es PAR");
19      else
20          JOptionPane.showMessageDialog(null,"El numero es IMPAR");
21
22      }
23  }
```

Figura 18. Ejemplo de estructura de decisión IF.

g. Estructura de decisión múltiple Switch()

Java incorpora al "switch()" como instrucción de decisión múltiple, aunque está solamente evalúa los posibles valores de una variable o expresión, veamos la sintaxis:

switch(<operador/variable>) {

case <valor 1>: <sentencias si valor 1>; break;

case <valor 2>: <sentencias si valor 2>; break;

...

case <valor n>: < sentencias si valor n>; break;

[default: <sentencias en otro caso>; break;]}

Ejemplo:

```
public class ejemplo03 {
    static int numero = (int)(1000*Math.random());
    static int resto = numero%6;
    public static void main(String args[]){
        System.out.println("El numero random: " + numero);
        System.out.println("Su resto en la division: " + resto);
        switch (resto) {
            case (0):
                System.out.println("Confirmo, el resto es cero");
                break;
            case (1):
                System.out.println("Confirmo, el resto es uno");
                break;
            case (2):
                System.out.println("Confirmo, el resto es dos");
                break;
            case (3):
                System.out.println("Confirmo, el resto es tres");
                break;
            default:
                System.out.println("El resto es mayor que tres");
                break;
        }
    }
}
```

ejercicio.pkg01.ejemplo03 > main > switch (resto) > default:

ut - Ejercicio-01 (run)

```
run:
El numero random: 74
Su resto en la division: 2
Confirmo, el resto es dos
```

Figura 19. Ejemplo de estructura de decisión múltiple SWITCH.

3.1.13 Estructuras repetitivas

a. Estructura For(). Esta estructura permite la repetición de un proceso, siempre que se cumpla una condición, y además incluye una forma de controlar el incremento o decremento de una variable de control.

Utilizar ésta estructura cuando se conozca con precisión cuántas veces debe repetir un proceso.

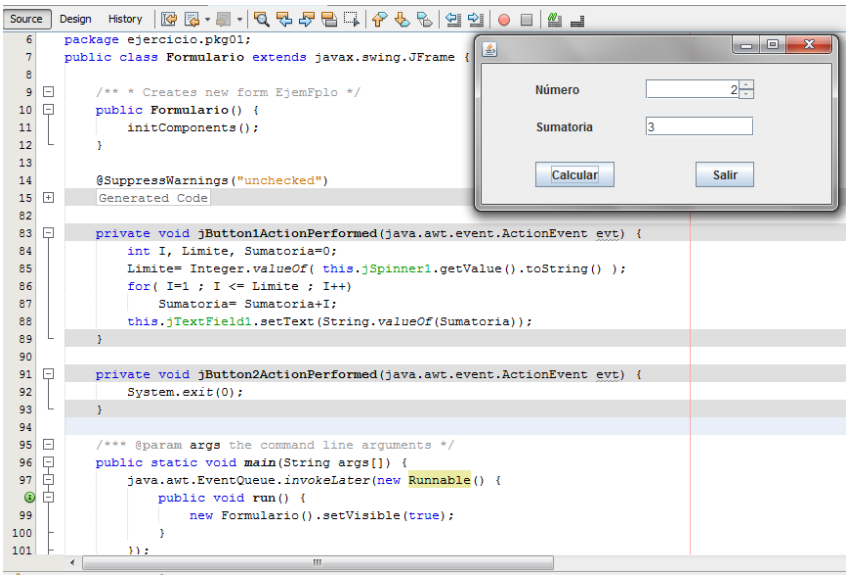
for(<Valor Inicial>, <Condición>,<Incremento/Decremento>)

{

<Sentencias mientras condición se cumpla>;

}

Ejemplo:




```
package ejercicio.pkg01;
public class Formulario extends javax.swing.JFrame {

    /** * Creates new form EjemFplo */
    public Formulario() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        int I, Limite, Sumatoria=0;
        Limite= Integer.valueOf( this.jSpinner1.getValue().toString() );
        for( I=1 ; I <= Limite ; I++)
            Sumatoria= Sumatoria+I;
        this.jTextField1.setText(String.valueOf(Sumatoria));
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    /*** @param args the command line arguments */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Formulario().setVisible(true);
            }
        });
```

Figura 20. Ejemplo de estructura for.

b. Estructura While().

Existen 2 variantes las cuales veremos luego, ésta estructura al igual que la anterior, permite la repetición de un proceso o bloque de código tantas veces sea posible y mientras una condición se cumpla, a diferencia del for() esta no posee la variable de control.

while(<condición>)

{

<Sentencias mientras condición es true>;

}

O en todo caso se puede hacer uso de la siguiente variante:

do

{

<Sentencias mientras condición es true>;

}while(<condición>)

El ejemplo anterior también se puede hacer utilizando estas estructuras, de la siguiente forma:

```
83      private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
84          int I=0, Limite, Sumatoria=0;
85          Limite= Integer.valueOf( this.jSpinner1.getValue().toString() );
86          while(I < Limite)
87          {
88              I++;
89              Sumatoria= Sumatoria+I;
90          }
91          this.jTextField1.setText(String.valueOf(Sumatoria));
92      }
```

Figura 21. Ejemplo 1 de estructura while

```
83      private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
84          int I=0, Limite, Sumatoria=0;
85          Limite= Integer.valueOf( this.jSpinner1.getValue().toString() );
86          do
87          {
88              I++;
89              Sumatoria= Sumatoria+I;
90          }while(I<Limite);
91          this.jTextField1.setText(String.valueOf(Sumatoria));
92      }
```

Figura 22. Ejemplo 2 de estructura do while

3.1.14 Funciones y procedimientos

Los procedimientos o funciones son rutinas o bloques de código, que calculan o procesan algún tipo de información. La diferencia entre ambos radica en que un procedimiento no retorna valores, solamente realiza una tarea, en cambio una función al margen de que realiza algún tipo de tarea, retorna un valor como resultado de la función.

a. ¿Cómo crear procedimientos?

Para crear un procedimiento podemos utilizar la siguiente sintaxis:

private void<nombre del procedimiento>(<Lista de parámetros>)

{

<Sentencias mientras del procedimiento>;

}

Ejemplo: Programa que permita calcular el área de un cuadrado o círculo.

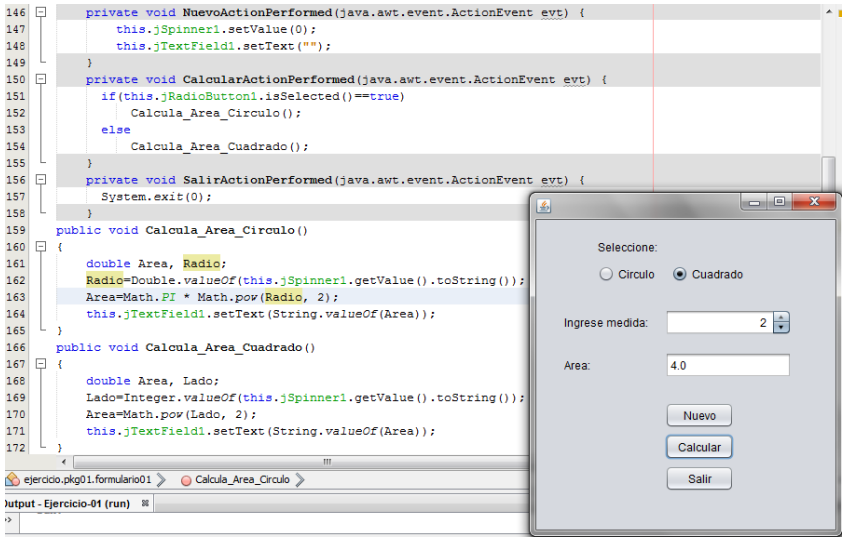


Figura 23. Área de un cuadrado o círculo con procedimientos

Como se observa, luego de haber seleccionado una de las opciones, es posible calcular cualquiera de las áreas, con el simple hecho de realizar una llamada a su respectivo procedimiento.

b. ¿Cómo crear funciones?

Las funciones son similares a los procedimientos, sin embargo, éstas devuelven un valor que en este será el resultado del cálculo realizado.

Ejemplo: Utilizaremos el mismo formulario del ejemplo anterior, el código sería:

```
    private void NuevoActionPerformed(java.awt.event.ActionEvent evt) {
        this.jSpinner1.setValue(0);
        this.jTextField1.setText("");
    }
    private void CalcularActionPerformed(java.awt.event.ActionEvent evt) {
      double Respuesta=0, Valor;
      Valor=Double.valueOf(this.jSpinner1.getValue().toString());
        if(this.jRadioButton1.isSelected()==true)
          Respuesta=Calcula_Area_Circulo(Valor);
      else
          Respuesta=Calcula_Area_Cuadrado(Valor);
        this.jTextField1.setText(String.valueOf(Respuesta));
    }
    private void SalirActionPerformed(java.awt.event.ActionEvent evt) {
      System.exit(0);
    }
public double Calcula_Area_Circulo(double Radio)
{
    double Area;
    Area=Math.PI * Math.pow(Radio, 2);
    return Area;
}
public double Calcula_Area_Cuadrado(double Lado)
{
    double Area;
    Area=Math.pow(Lado, 2);
    return Area;
}
```

Figura 24. Área de un cuadrado o círculo con funciones

3.1.15 Base de datos y mysql

3.1.15.1 Definición de base de datos

Según Montenegro (2012): "Una base de datos (DB) es una colección de datos interrelacionados, pertenecientes a un mismo contexto, y almacenados sistemáticamente para su posterior uso" (p. 2).

Según Montenegro (2012): "Un sistema gestor de bases de datos (DBMS) es un programa que almacena y accede a la información contenida en las bases de datos" (p .2).

Según Montenegro (2012): "Una base de datos modela la información sobre ciertas entidades, y sobre relaciones entre las mismas"(Montenegro, 2012, p.2).

a. Características

Entre las principales características de los sistemas de base de datos podemos mencionar:

- Independencia lógica y física de los datos
- Redundancia mínima
- Acceso concurrente por parte de múltiples usuarios
- Integridad de los datos
- Consultas complejas optimizadas
- Seguridad de acceso y auditoría
- Respaldo y recuperación
- Acceso a través de lenguajes de programación estándar

(Pérez, 2007, párr.5)

b. Bases de datos relacionales

Entidades:

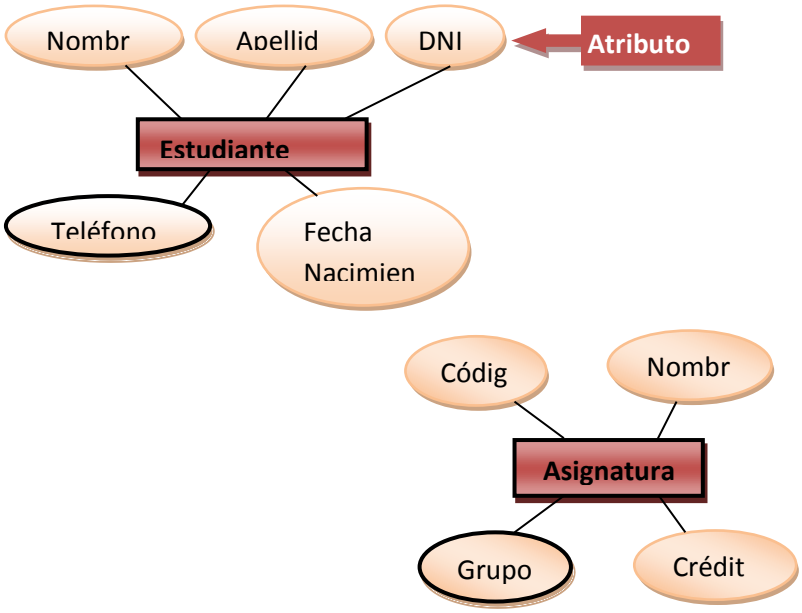


Figura 25. Entidades y sus atributos

Fuente: Bases de datos y JDBC

Relaciones:

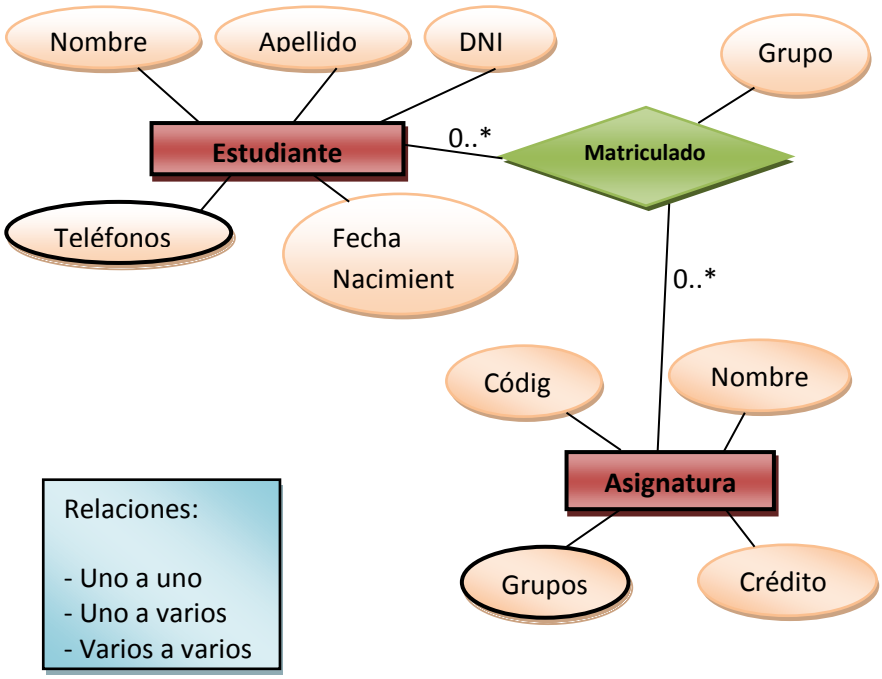


Figura 26. Tipos de Relaciones

Fuente: Montenegro, 2012

Según Montenegro (2012): "El modelo relacional de base de datos es el más usado en la actualidad. Se basa en la idea de relación, considerada como un conjunto de tuplas. Cada relación es una tabla con filas (registros) y columnas (atributos)" (p. 9).

Tabla 6

Ejemplo Estudiante

DNI	Nombre	Apellidos	Fecha Nac.	Teléfonos
12345673V	Ricardo	Fernández Aguinaga	20/04/1980	912421124
82122314X	Luis	Díaz Castro	25/04/1978	913111564
…	…	…	…	…

Fuente: Montenegro, 2012

Tabla 7

Ejemplo Asignatura

Código	Nombre	Créditos	Grupos
101	Álgebra	15	A,B,C
102	Funciones de una variable	12	A,B,C
…	…	…	…

Fuente: Montenegro, 2012

Tabla 8

Matriculado y sus relaciones

DNI Estudiante	Cod. Asignatura	Grupo
12345673V	101	A
12345673V	102	A
82122314X	103	B
…	…	…

Fuente: Montenegro, 2012

c. *Sistemas gestores de bases de datos*

Según Berzal (s.f.): "Software con capacidad para definir, mantener y utilizar una base de datos. Un sistema de gestión de bases de datos debe permitir definir estructuras de almacenamiento, acceder a los datos de forma eficiente y segura, etc." (p.9).

Existe una gran cantidad de gestores de bases de datos relacionales.

a. Oracle

b. Microsoft SQL Server

c. Microsoft Access

d. **MySQL**

e. PostgreSQL

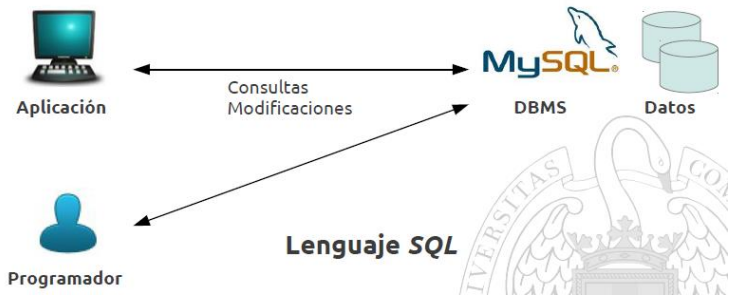


Figura 27. DBMS MYSQL

Fuente: Bases de datos y JDBC

3.1.15.2 Lenguaje SQL (Structured Query Language)

Es un lenguaje declarativo de acceso a bases de datos relacionales.

a. DDL (Data definition language)

Modificación de la estructura de la base de datos. Creación de tablas.

b. DML (Data manipulation language)

Consulta, inserción, y eliminación de registros dentro de una tabla (Montenegro, 2012, p.23).

3.1.15.3 MySQL

MySQL, es un Sistema de Administración de Base de Datos de código abierto, es licenciado bajo la GPL (General Public License) de la GNU. Fue creada por la empresa sueca MySQL AB. MySQL es el sistema administrador de base de datos más usado en el mundo del software libre, debido a su gran rapidez, confiabilidad y facilidad de uso. (Montalvo, 2015, p.2).

a. Características de MySQL

Internas y de portabilidad:

Escrito en C y C++.

Está disponible en diferentes plataformas: Linux, Solaris, FreeBSB, Mac OS X, HP-UX, AIX, Windows, etc.

Disponibilidad de APIs para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y Tcl. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.

Tablas Hash en memoria, son usadas como tablas temporales.

El código de MySQL ha sido probado (Tested) con las principales herramientas del mercado.

El servidor está disponible como un programa separado para ser usado en un ambiente cliente/servidor. (Montalvo, 2015, p.6)

Tipos de Columna soportados:

a. INTEGER de 1, 2, 3, 4, y 8 Bytes
b. FLOAT
c. DOUBLE
d. CHAR
e. VARCHAR
f. TEXT
g. BLOB
h. DATE, TIME, DATETIME, TIMESTAMP, YEAR
i. Tipos espaciales OpenGIS (Montalvo, 2015, p.7).

Sentencias y Funciones:

a. Soporte para las cláusulas GROUP BY y ORDER BY.
b. Pueden usarse las funciones: COUNT(), COUNT(DISTINCT ...),AVG(), STD(), SUM(), MAX() y MIN().
c. Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN usando notación SQL estándar.
d. Soporte para alias sobre: tablas y columnas usando SQL estándar.
e. Las sentencias DELETE, INSERT y UPDATE retornan el número de filas que han sido afectadas.
f. Se puede mezclar tablas de diferentes bases de datos en la misma consulta.(Montalvo, 2015, p.8)

Seguridad

Según Montalvo (2015): "Maneja un sistema de privilegios muy seguro, la verificación se hace basado en host" (p. 9).

Escalabilidad y Límites

a. Maneja base de datos grandes. Su uso se extiende a más de 50 millones de registros. Se tiene conocimiento de algunos usuarios que usan el servidor MySQL con más de 60,000 tablas y cerca de 5.000.000.000 de filas.
b. Hasta 64 índices por tabla son permitidos. Cada índice puede consistir de 1 a 16 columnas. (Montalvo, 2015, p. 9).

Conectividad

a. Los clientes pueden conectarse al servidor MySQL usando TCP/IP sobre cualquier plataforma.
b. El conector/ODBC (MyODBC) provee soporte a programas cliente que usen ODBC (Open Database Connectivity).
c. La interfase conector/J provee soporte para programas cliente java que usan JDBC (Montalvo, 2015, p. 9).

3.1.15.4 Procedimientos almacenados y funciones

"Los procedimientos almacenados y funciones son nuevas funcionalidades de la versión de MySQL 5.0"(Guebs, Cap.19, 2007).

Un procedimiento almacenado es un conjunto de comandos SQL que pueden almacenarse en el servidor. Una vez que se hace, los clientes no necesitan relanzar los comandos individuales, pero pueden en su lugar referirse al procedimiento almacenado. Los procedimientos almacenados pueden mejorar el rendimiento ya que se necesita enviar menos información entre el servidor y el cliente. El intercambio que hay es que aumenta la carga del servidor de la base de datos ya que la mayoría del trabajo se realiza en la parte del servidor y no en el cliente (Guebs, Cap.19, 2007).

"Considere esto sí muchas máquinas cliente (como servidores Web) se sirven a sólo uno o pocos servidores de bases de datos"(Guebs, Cap.19, 2007).

Los procedimientos almacenados le permiten tener bibliotecas o funciones en el servidor de base de datos. Esta característica es compartida por los lenguajes

de programación modernos que permiten este diseño interno, por ejemplo, usando clases. Usando estas características del lenguaje de programación cliente es beneficioso para el programador incluso fuera del entorno de la base de datos. Algunas situaciones en que los procedimientos almacenados pueden ser particularmente útiles:

a. Cuando múltiples aplicaciones cliente se escriben en distintos lenguajes o funcionan en distintas plataformas, pero necesitan realizar la misma operación en la base de datos.

b. Cuando la seguridad es muy importante. Los bancos, por ejemplo, usan procedimientos almacenados para todas las operaciones comunes. Esto proporciona un entorno seguro y consistente, y los procedimientos pueden asegurar que cada operación se loguea apropiadamente. En tal entorno, las aplicaciones y los usuarios no obtendrían ningún acceso directo a las tablas de la base de datos, sólo pueden ejecutar algunos procedimientos almacenados (Guebs, Cap.19, 2007).

3.1.15.5 SQLyog

Es una magnífica interfaz gráfica diseñada particularmente para trabajar de forma más agradable y rápida con el servidor de base de datos MySQL. El programa está dirigido a usuarios que ya tienen algunos conocimientos de SQL y requieren un intérprete gráfico sin demasiados adornos, rápido y funcional. **SQLyog** te posibilita administrar usuarios y permisos, y hacer múltiples solicitudes a BD.

Asimismo puedes agregar sencillamente, a través de plantillas, solicitudes que hagas frecuentemente, como creación de tablas. Otras alternativas son la exportación de datos en formato HTML, CSV y XML, la optimización de bases

de datos, la posibilidad de guardar scripts de SQL como preferidos (Ecured, s.f, párr.1)

Características importantes

a. Soporte Unicode/UTF8 total
b. Productividad desarrollador / usuario
c. Documentación de esquema HTM
d. Atajos para crear sentencias SQL DML desde definición de esquema
e. Editor de consultas con pestañas múltiples y editor de resultados
f. Ejecución múltiple de consultas
g. Ejecución de consultas multi-hilo - Posibilidad de detener consultas amplias
h. Plantillas SQL
i. Interfaz de cuadrícula estilo Excel para visualizar / actualizar los resultados
j. Editor de Blob multi-formato
k. Se pueden visualizar datos en texto o modo cuadrícula
l. Exportación de Resultados / Datos a CSV / Excel / HTML / XML
m. Soporte total de versiones desde 3.23.38 hasta la última
n. Exportación de resultados / archivo amigable con Excel / datos de tabla al portapapeles
o. Editor de tabla y resultado sin diálogo
p. Restaura / Importa volcados SQL amplios
q. Soporta objetos MySQL 5.x
r. Administra MySQL hospedado
s. Administra índices
t. Administra relaciones/claves foráneas

u. Reordenar columnas

v. Copiar con un solo clic objetos a otro servidor

w. Diagnóstico de tablas

x. Crear/borrar bases de datos

y. Explorador de objetos

z. Herramientas para despejar

aa. Optimizado para la gestión de MySQL hospedado

bb. Administración MySQL de alta rapidez. Utiliza una API MySQL nativa en C - la forma más rápida de comunicarse con un servidor MySQL

cc. 100% manejable con el teclado

dd. Binario pequeño y compacto

ee. Uso reducido del registro - migración sencilla de las prioridades del usuario arrastrando y soltando archivos de configuración.

ff. Aspecto visual ordenado, se pueden visualizar/ocultar paneles

(Ecured, s.f, párr. 2)

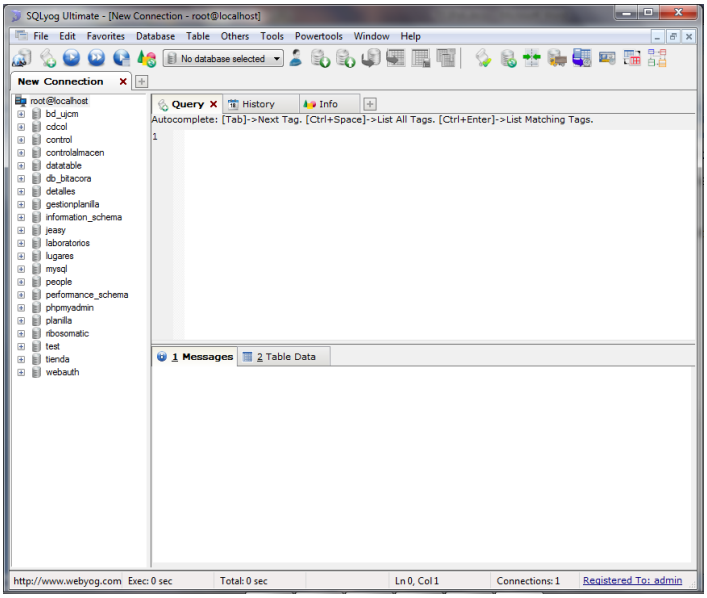


Figura 28. Ventana de Interfaz de SQLyog

Fuente: Elaboración propia

3.1.15.6 Ejercicio en MYSQL

a. Creando la base de datos DbVentas

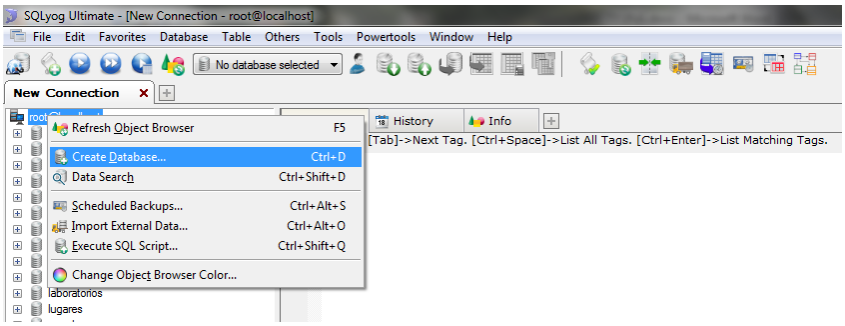


Figura 29. Creación de base de datos en SQLyog

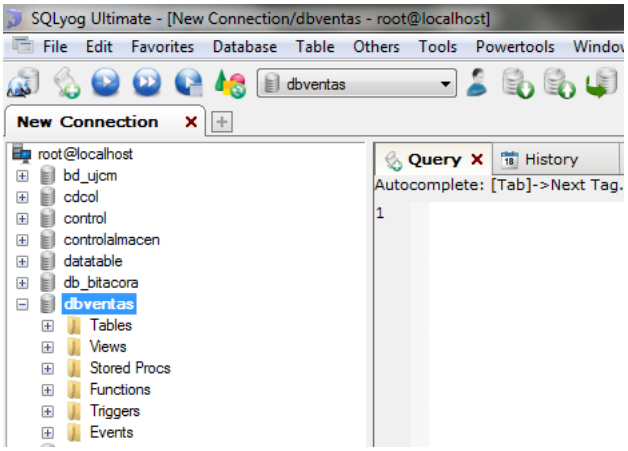


Figura 30. Ventana mostrando la DBVentas

Se puede crear tablas, vistas, procedimientos almacenados, triggers y eventos.

b. Creación de tablas

En la carpeta **tables** se elige **create table** e ingresamos el nombre de la tabla y los campos como lo muestra la imagen:

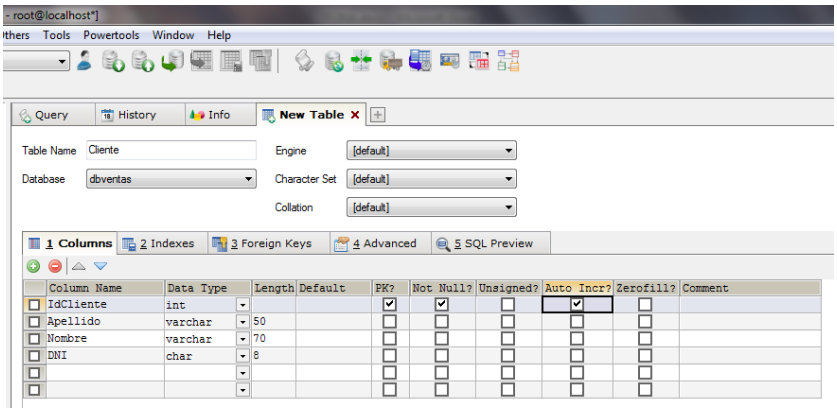


Figura 31. Ventana de creación de tabla Cliente.

c. Insertando registros en la tabla:

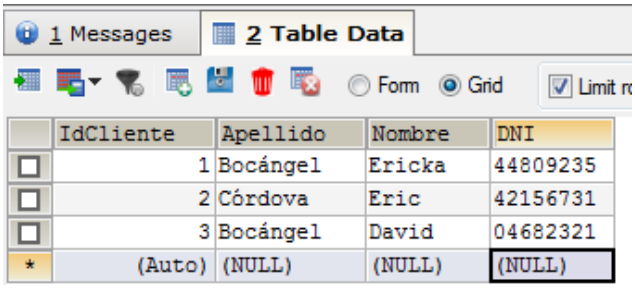


Figura 32. Ventana inserción de registros.

Código T-SQL

d. Desarrollando consultas T-SQL

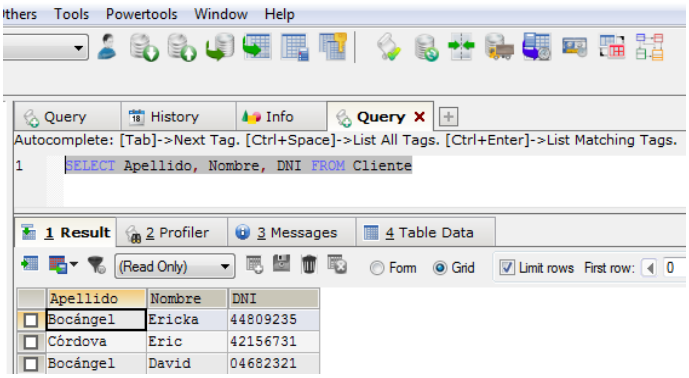


Figura 33. Desarrollo de Consulta Select.

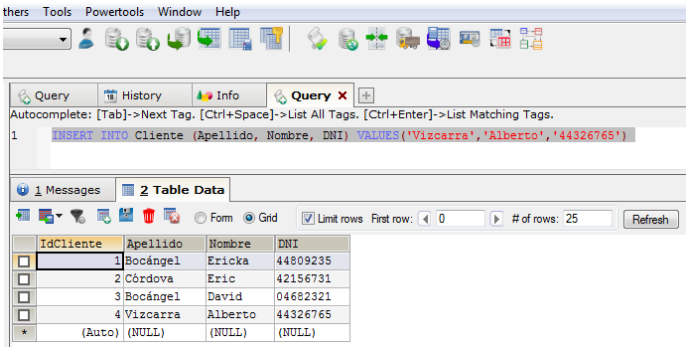


Figura 34. Desarrollo de consulta Insert

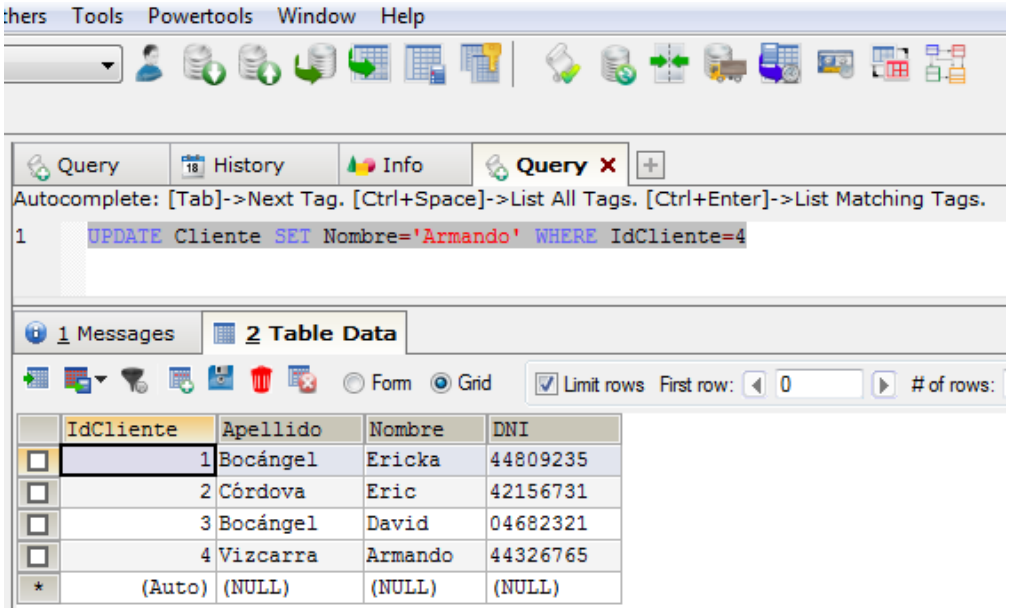


Figura 35. Desarrollo de consulta Update

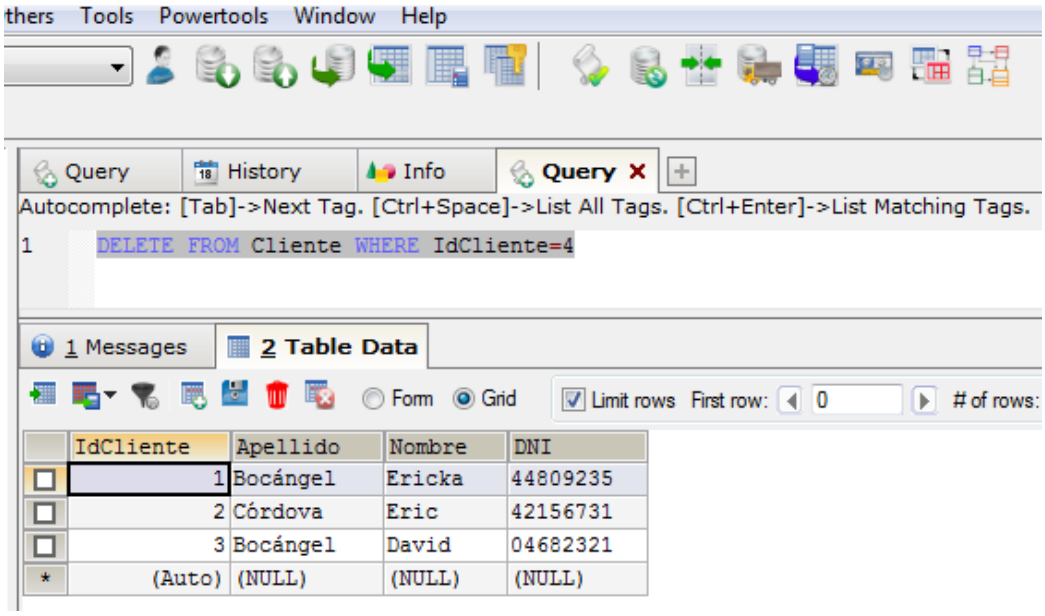


Figura 36. Desarrollo de consulta Delete

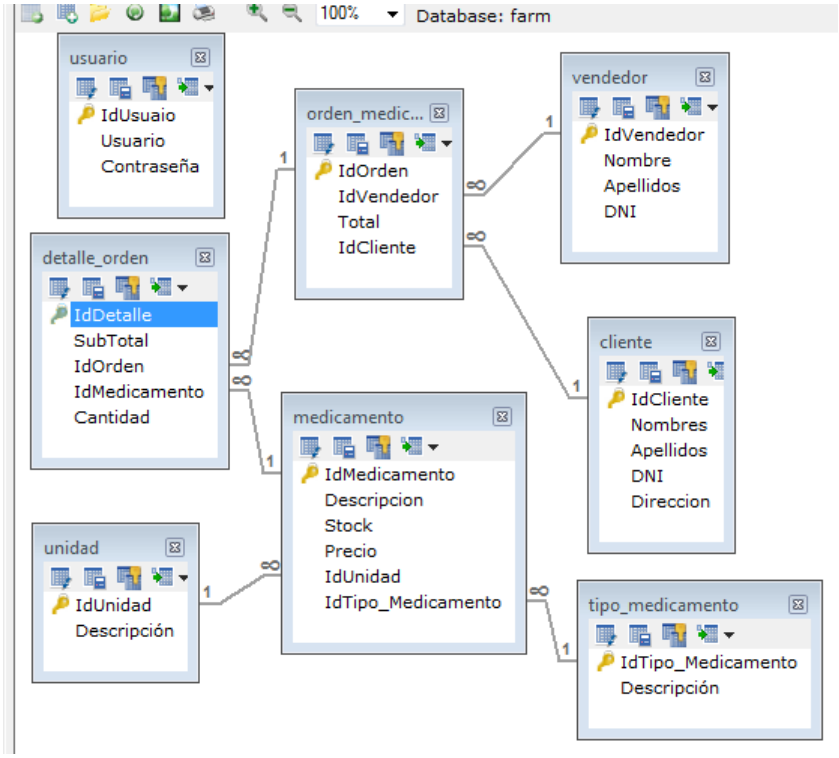


Figura 37. Base de Datos en SQLyog

3.2 Caso práctico

3.2.1 Recursos de software

Para el desarrollo de la aplicación:

a. Microsoft Windows 7 Home Premium 64 bits.

b. XAMPP 3.2.1

c. SQLyog 9.6.3.0

d. NetBeans 8.1

3.2.2 Implementación

a. Descripción

El caso práctico se desarrolló con el fin de demostrar cómo realizar una aplicación con Java y Mysql partiendo de lo teórico, se realizó la base de datos de Control de Laboratorio y se desarrolló algunos procedimientos almacenados y demás procesos necesarios.

La aplicación fue implementada en SQLyog y en el entorno de Netbeans con el Lenguaje Java habiendo hecho la conexión con la base de datos, se implementó dos formularios realizando las acciones básicas.

b. Alcance

La aplicación, abarca el registro de equipos, informes para mantener un control del laboratorio de cómputo en cuanto a existencias, perdidas y estado de los equipos.

c. Diagramas UML obtenidos

Caso de uso

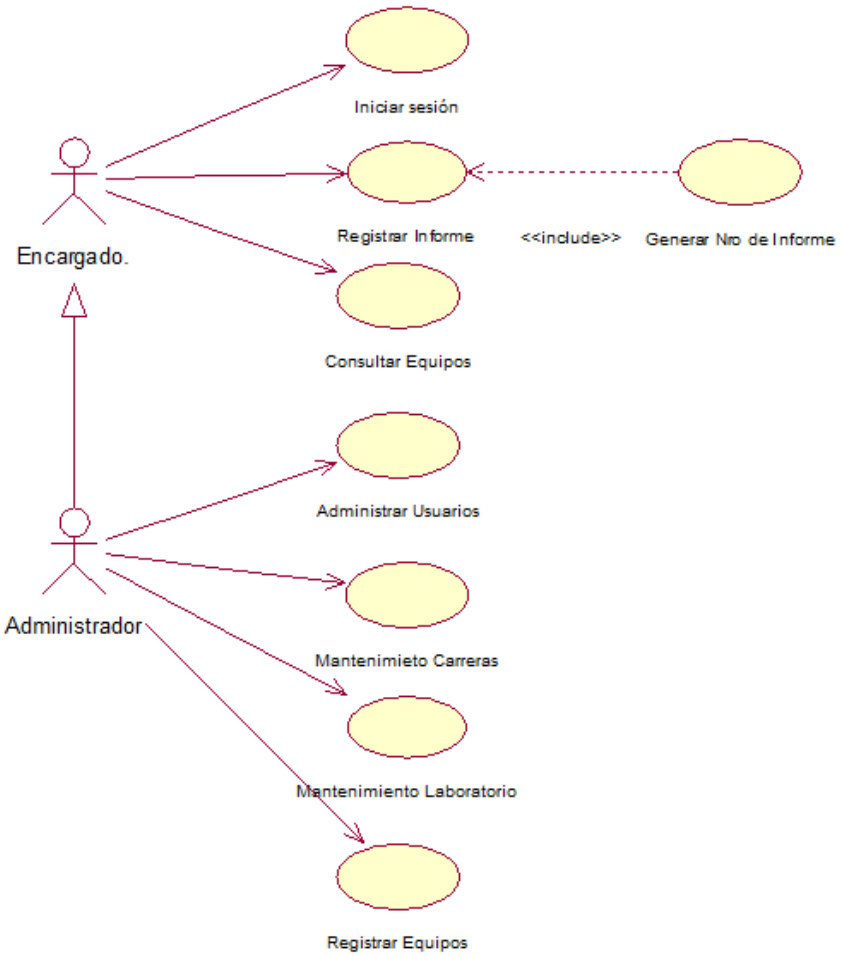


Figura 38. Diagrama de caso de uso

Escenarios de casos de uso

Tabla 9

Escenario de caso de uso iniciar sesión

Iniciar sesión	
ID:	CU01
Actores:	Encargado
Descripción:	Permite el ingreso del encargado a la aplicación.
Pasos a realizar:	1. El encargado accede a la aplicación. 2. La aplicación le solicita credenciales al encargado. 3. El encargado ingresa sus credenciales 3.1 Si los datos ingresados son incorrectos la aplicación mostrará mensaje de error y podrá volver a intentarlo. 4. La aplicación permite el ingreso al encargado.
Precondiciones:	El usuario cuenta con credenciales válidas.
Prioridad:	Alta

Tabla 10

Escenario de caso de uso registrar informe

Registrar informe	
ID:	CU02
Actores:	Encargado
Descripción:	Permite el registro y edición de informes.
Pasos a realizar:	1. El encargado accede al formulario de Informes. 2. El encargado ingresa los datos del nuevo informe, edita o busca según el responsable. 3. La aplicación registra, actualiza o busca resultados de los datos ingresados.
Precondiciones:	Debe generar primero nro. de informe para poder ingresar los detalles del mismo.
Prioridad:	Alta

Tabla 11

Escenario de caso de uso generar nro. de informe

Generar nro. de informe	
ID:	CU03
Actores:	Encargado
Descripción:	Permite generar el nro. de informe.
Pasos a realizar:	1. El encargado accede al formulario de Informes. 2. El encargado ingresa los datos para generar el nro. del nuevo informe. 5. La aplicación registra los datos.
Precondiciones:	El administrador debe haber ingresado datos de equipos anteriormente.
Prioridad:	Alta

Tabla 12

Escenario de caso de uso consultar equipos

Consultar equipos	
ID:	CU04
Actores:	Encargado
Descripción:	Permite la búsqueda por estado del equipo al encargado.
Pasos a realizar:	3. El encargado accede al formulario de Consultar Equipos. 4. El encargado ingresa el estado a buscar. 5. La aplicación muestra el registro o registros del estado buscado.
Precondiciones:	El estado a buscar debe existir en la base de datos.
Prioridad:	Baja

Tabla 13

Escenario de caso de uso administrar usuario

Administrar usuario	
ID:	CU05
Actores:	Administrador
Descripción:	Permite ingresar un nuevo Usuario y editar los registros.
Pasos a realizar:	1. El administrador accede al formulario Administrar Usuarios. 2. El administrador ingresa los datos del nuevo usuario, edita o busca un registro. 3. La aplicación registra, actualiza o busca los datos.
Precondiciones:	El usuario debe ser administrador.
Prioridad:	Alta

Tabla 14

Escenario de caso de uso mantenimiento de carreras

Mantenimiento de carreras	
ID:	CU06
Actores:	Administrador
Descripción:	Permite ingresar una nueva Carrera y editar los registros.
Pasos a realizar:	1. El administrador accede al formulario Mantenimiento de Carreras. 4. El administrador ingresa los datos de la nueva carrera, edita, elimina o busca según la carrera. 2. La aplicación registra, actualiza o busca resultados de los datos ingresados.
Precondiciones:	El usuario debe ser administrador.
Prioridad:	Alta

Tabla 15

Escenario de caso de uso mantenimiento de laboratorios

Mantenimiento de laboratorios	
ID:	CU07
Actores:	Administrador
Descripción:	Permite ingresar un nuevo Laboratorio y editar los registros.
Pasos a realizar:	5. El administrador accede al formulario Mantenimiento de Laboratorios. 6. El administrador ingresa los datos del nuevo laboratorio, edita o elimina un registro seleccionado. 7. La aplicación registra o actualiza los datos.
Precondiciones:	El usuario debe ser administrador.
Prioridad:	Alta

Tabla 16

Escenario de caso de uso registrar equipos

Registrar equipos	
ID:	CU08
Actores:	Administrador
Descripción:	Permite ingresar un nuevo Equipo y editar los registros.
Pasos a realizar:	8. El administrador accede al formulario Equipos. 9. El administrador ingresa los datos del nuevo equipo, edita o busca según su estado. 10. La aplicación registra, actualiza o busca resultados de los datos ingresados.
Precondiciones:	El usuario debe ser administrador.
Prioridad:	Alta

Diagrama de clases

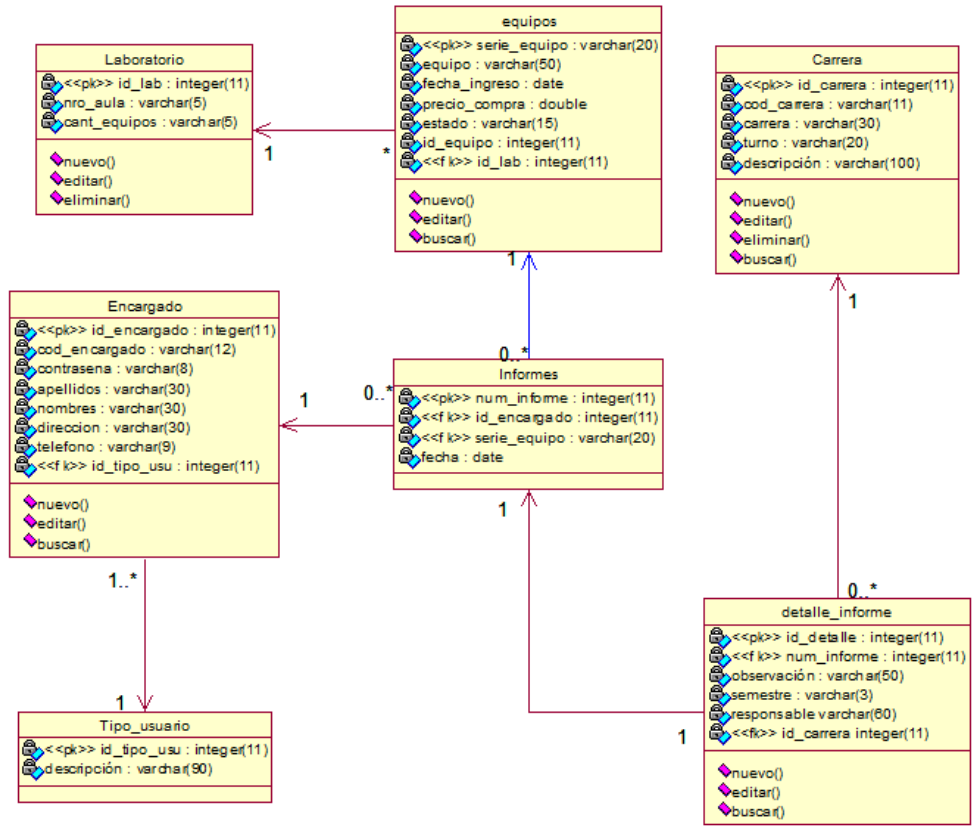


Figura 39. Diagrama de clases

Diagrama de actividades

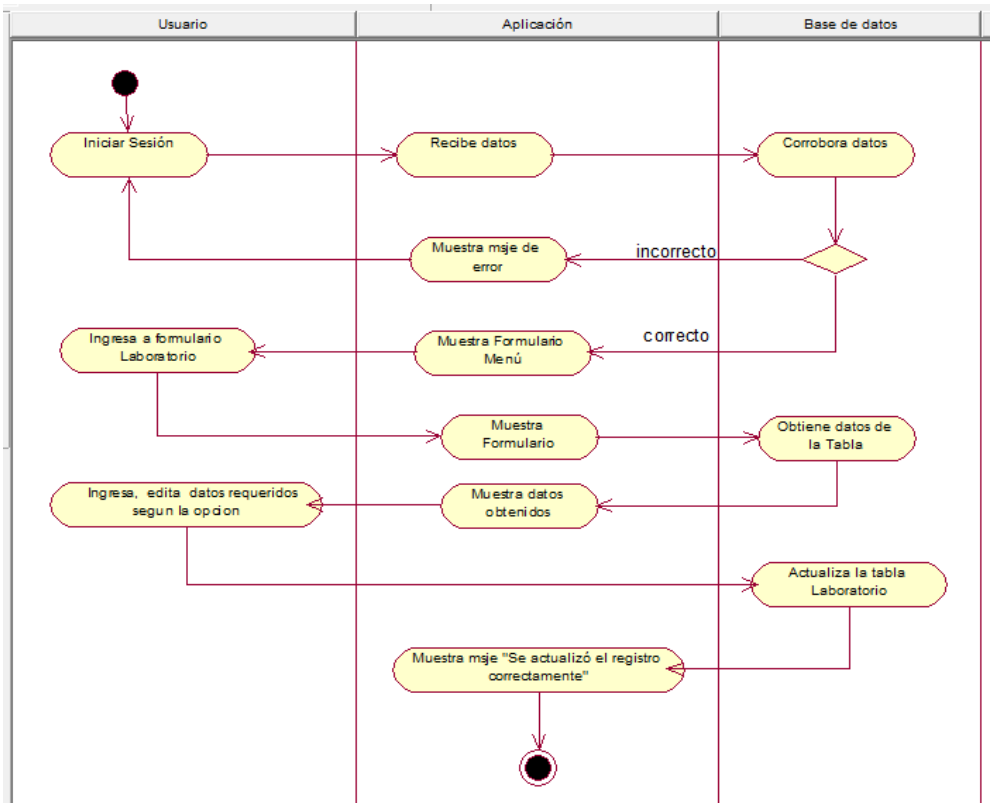


Figura 40. Diagrama de actividades - mantenimiento laboratorios

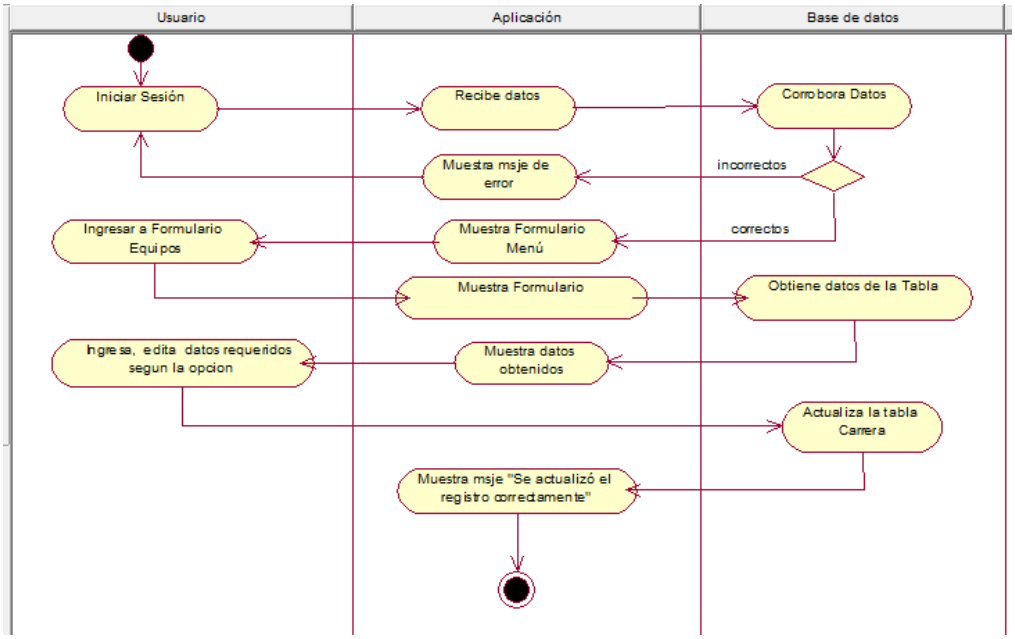


Figura 41. Diagrama de actividades - equipos

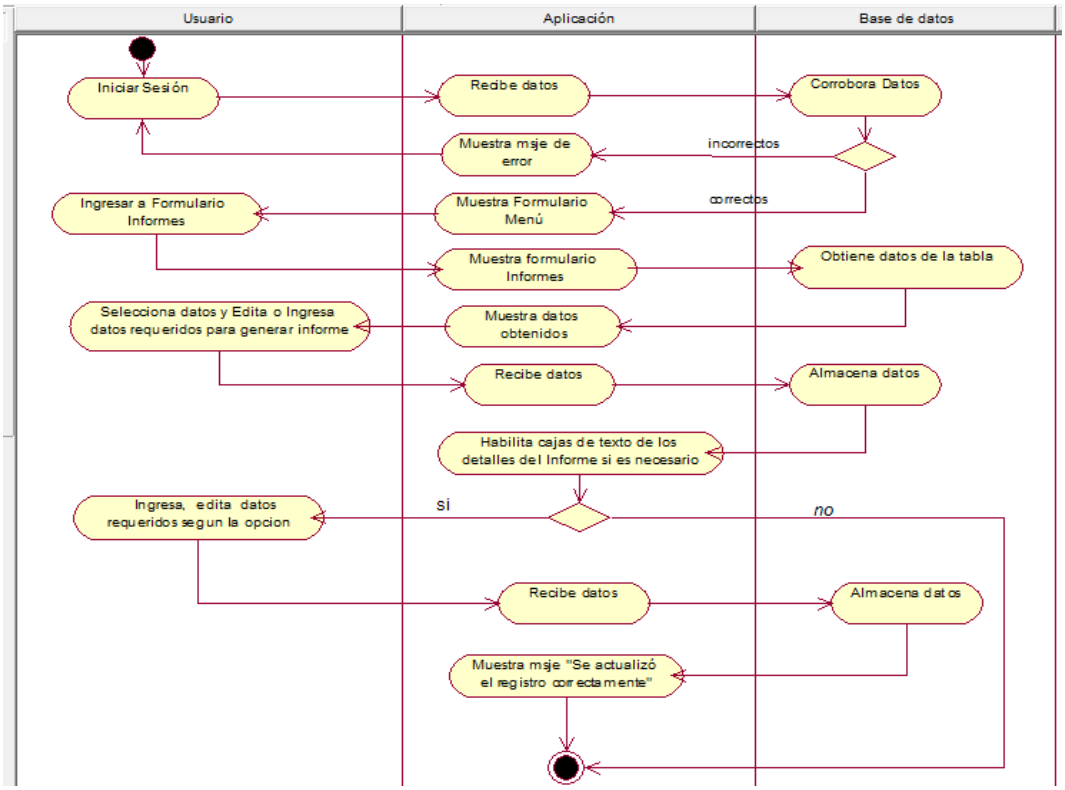


Figura 42. Diagrama de actividades – informes

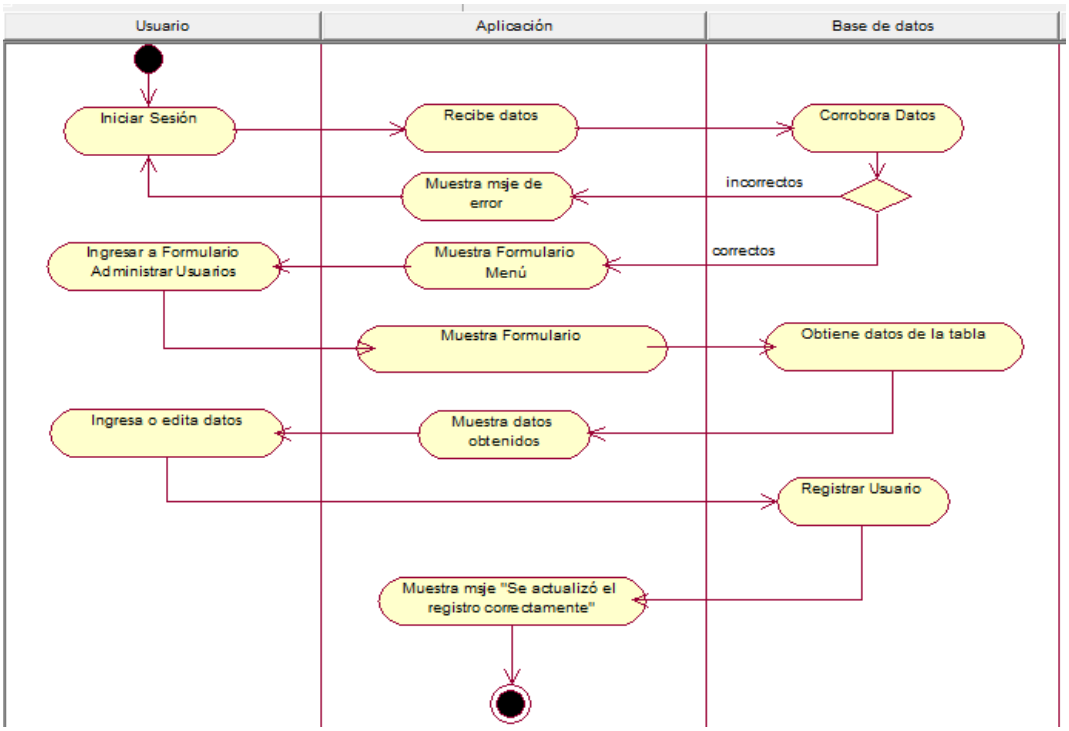


Figura 43. Diagrama de actividades - administrar usuarios

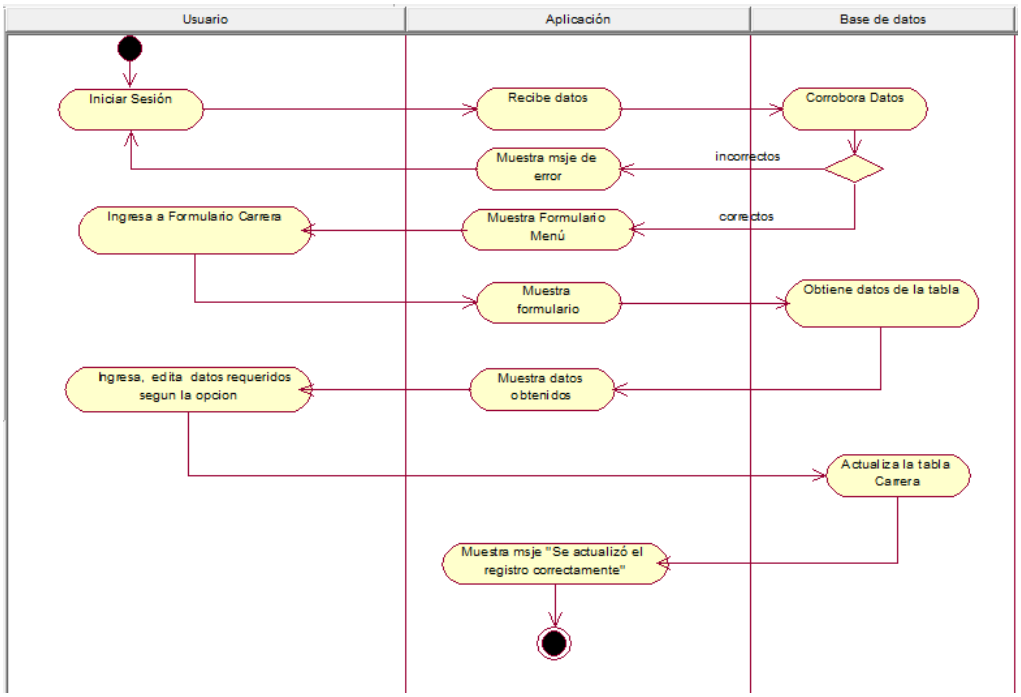


Figura 44. Diagrama de actividades - mantenimiento de carreras

d. Base de datos en MySQL

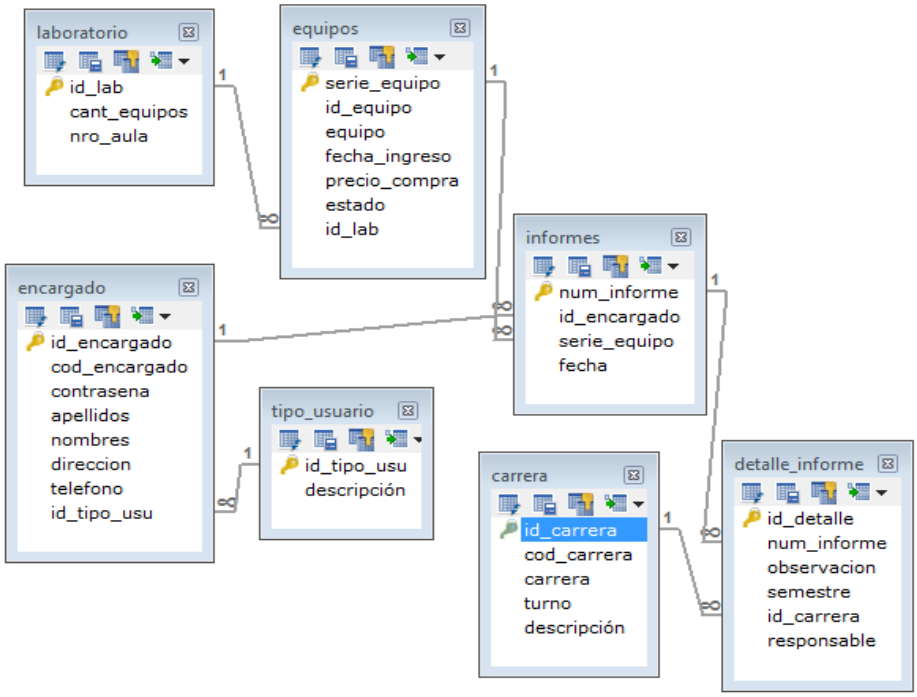


Figura 45. Base de datos (control de laboratorio de cómputo) en MySQL

3.3 Representación de resultados

Se implementó una Aplicación, y se obtuvo el resultado esperado para la comprensión de programación de aplicaciones en Java con Netbeans y Base de datos creada en Mysql.

A continuación, se detalla el trabajo realizado a través de vistas para una mejor explicación de la aplicación desarrollada.

3.3.1 Login

Permite validar el inicio de sesión a la aplicación.

Figura 46. Formulario login

3.3.2 Menú

Muestra las opciones para ingresar y gestionar los diferentes procesos.

Figura 47. Formulario menú

3.3.3 Carrera

La aplicación permite gestionar la tabla que servirán para tener la información de las carreras, poder ingresar nuevas carreras, modificar algunos datos de las mismas, eliminar alguna carrera, buscar una determinada carrera por nombre y así mismo mostrar los registros de la tabla.

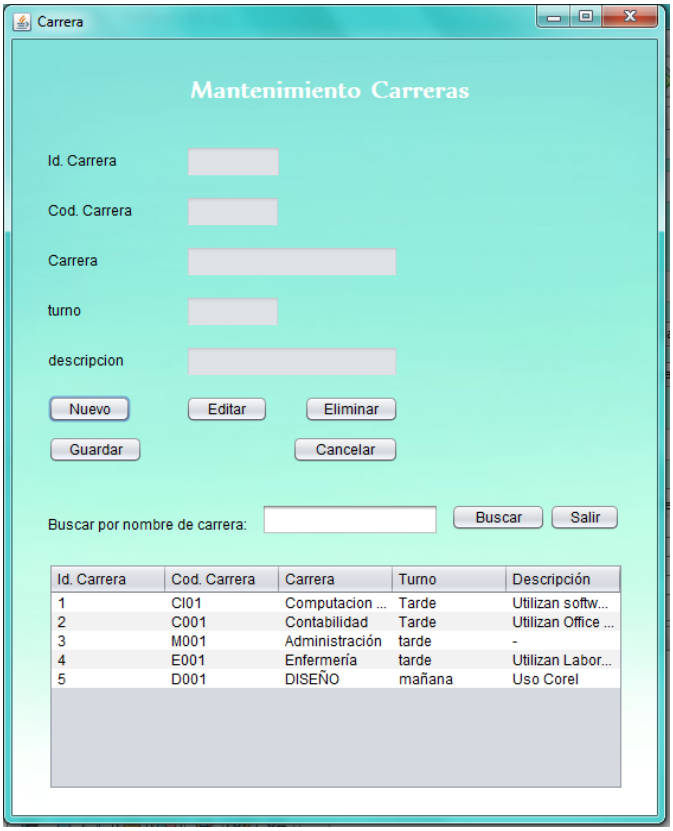


Figura 48. Formulario carrera

3.3.4 Laboratorios

La aplicación permite gestionar la tabla que servirán para tener la información de los laboratorios, poder ingresar nuevos laboratorios, modificar algunos datos de las mismas, eliminar algún laboratorio, y así mismo mostrar los registros de la tabla.

Figura 49. Formulario laboratorios

3.3.5 Equipos

La aplicación permite gestionar los equipos que servirán para llevar un control de los mismos, poder ingresar nuevos equipos, modificar algunos datos de las mismas, buscar equipos por estado y así mismo mostrar los registros de la tabla.

Figura 50. Formulario equipos

3.3.6 Informes

La aplicación permite gestionar los Informes que servirán para llevar un control de los equipos y ver cuál es el incidente, poder ingresar nuevos informes, modificar algunos datos, buscar informes por nombre del responsable y así mismo mostrar los registros de la tabla.

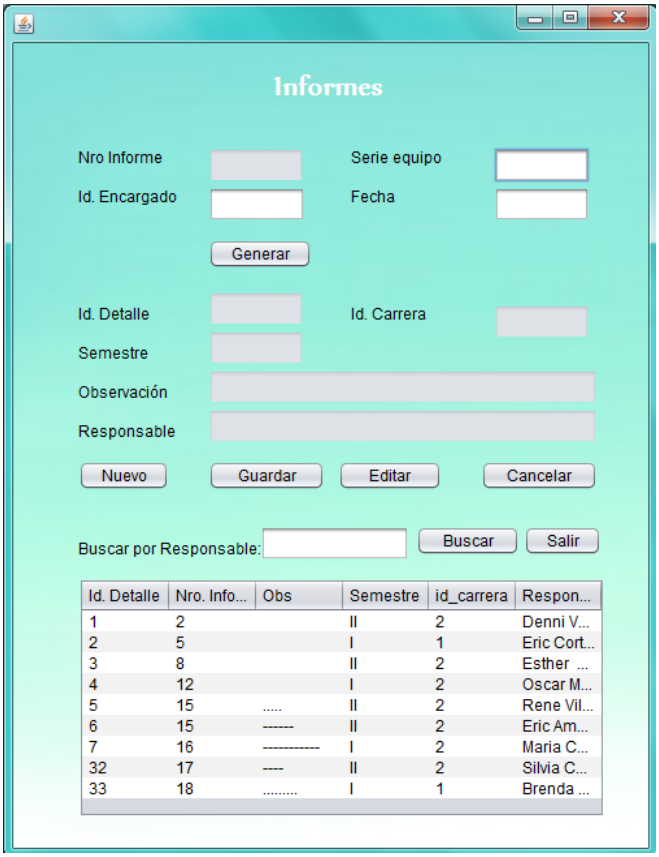


Figura 51. Formulario informes

Elaboración propia

3.3.7 Administrar usuarios

La aplicación permite administrar los Usuarios que servirán para llevar un control del acceso a la aplicación, poder ingresar nuevos usuarios, modificar algunos datos de los mismos, buscar usuarios por apellido de usuario y así mismo mostrar los registros de la tabla.

Figura 52. Formulario administrar usuarios

Elaboración propia

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Primera. Poner en práctica los conocimientos de Java y Base de datos nos permitió desarrollar la aplicación con las ventajas de las características de Java y las funciones útiles de MySQL de manera sencilla y rápida.

Segunda. La aplicación nos permitió llevar un control de las existencias y el estado de los equipos. El registrar informes en cuanto al estado o pérdida de un equipo aseguró un buen control del laboratorio de cómputo.

4.2 Recomendaciones

Primera. Conocer primero los fundamentos teóricos y prácticos de los lenguajes, entornos y sistemas con los que trabajaremos ya que será necesaria para el desarrollo de la aplicación.

Segunda. Es recomendable el uso de una aplicación para llevar un buen control, disminuyendo así errores y tiempo para actualizar la información y a la vez quede el registro de datos guardado para cualquier consulta.

REFERENCIAS BIBLIOGRÁFICAS

Martínez,L. J. (2015). Fundamentos de programación en Java. Recuperado de https://issuu.com/danielvillanuevamontoya/docs/fundamentos_de_programacion_en_java

Froufe, A. (1999). Tutorial de Java. Recuperado de http://dis.um.es/~bmoros/Tutorial/parte8/cap8-1a.html#Cap8_1_1

Padilla, S. (2013). Curso de Java. Recuperado de https://javacurso.wordpress.com

Pérez Valdés, D. (2007). Qué son las bases de datos? Recuperado de http://www.maestrosdelweb.com/que-son-las-bases-de-datos/

Montalvo, V. (2015). Fundamentos de MySQL. Recuperado de http://slideplayer.es/slide/2765705/

Berzal, F. (s.f.). Introducción a las bases de datos. Recuperado de http://elvex.ugr.es/idbis/db/

Montenegro, M. (2012). Bases de datos y JDBC. Recuperado de http://dalila.sip.ucm.es/~manuel/JSW1/Slides/BasesDeDatos.pdf

Guebs. (2008). Mysql 5.0 Manual de Referencia. Recuperado de http://manuales.guebs.com/mysql-5.0

Ecured. (s.f.). SQLyog . Recuperado de http://www.ecured.cu/SQLyog

Quevedo, N. (2014). Citas y Referencias, Recomendaciones y aspectos básicos del estilo APA.

Universidad José Carlos Mariátegui. Manual de elaboración de Tesis, Trabajo de suficiencia profesional y Artículo Científico. ISO 9001:2015