Universidad Autónoma del Estado de México
Centro Universitario UAEM Zumpango

Ingeniería en computación

Propuesta de un sistema Web para registro de solicitudes de apoyos municipales en Zumpango, Estado de México

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA:

Guillermo Crescencio Castillo

DIRECTOR DE TESIS:
Dr. en C.C. Asdrúbal López Chau

Zumpango, Estado de México. Septiembre 2021

Resumen

En esta tesis desarrollamos un sistema de registro de solicitudes de apoyos municipales. Este sistema se compone de dos partes principales:

a) Una aplicación de escritorio encargada de gestionar el registro de usuarios; y crear, leer, actualizar y borrar datos (operaciones CRUD). Esta parte del sistema fue desarrollada con el lenguaje de programación Java.

b) Una aplicación web utilizada para monitorear las solicitudes diarias de los usuarios. Esta parte del sistema se desarrolló con el lenguaje de programación PHP como back-end y HTML para el front-end.

Los requisitos funcionales se obtuvieron mediante la técnica observacional, identificando doce funciones importantes. Todas estas funciones se probaron con datos del mundo real. Según las pruebas, el sistema funciona correctamente.

Abstract

In this thesis we develop a system for registration of requests for municipal supports. This system is composed of two main parts:

a) A desktop application in charge of managing user registration; and create, read, update and delete of data (CRUD operations). This part of the system was developed with the Java programaming languaje.

b) A Web application used to monitor the daily requests of users. This part of the system was developed with PHP programming language as back-end, and HTML for the front-end.

The functional requirements were obtained by the observational technique, identifying twelve important functions. All of these functions were tested with real-world data. According to the tests, the system works correctly.

Dedicatoria

A Dios, a mi Abuela Candelaria y a mi tío Francisco, quienes me acompañaron y desde el cielo guían mi camino. A mis padres, quienes son los pilares fundamentales en mi vida, con mucho amor y cariño, les dedico todo mi esfuerzo, en reconocimiento a todo el sacrificio puesto para que yo pueda estudiar, se merecen esto y mucho más. A Diana y Ariadna mis queridas hermanas por ser mi apoyo incondicional. A Beatriz, mi gran amor por ser mi compañera inseparable de cada día.

A todos ustedes, con amor.

Agradecimientos

Quiero expresar mi agradecimiento a la Universidad Autónoma del Estado de México y al Centro Universitario UAEM Zumpango, por abrirme sus puertas y formarme como ingeniero.

A mi asesor, Dr. Asdrúbal López Chau por haberme guiado, no sólo en la elaboración de este trabajo de titulación, sino a lo largo de mi carrera universitaria y haberme brindado el apoyo para desarrollarme profesionalmente y seguir cultivando mis valores.

A mis revisores, Ing. en Informática Hazem Alvarez Rodríguez y M. en C.C Rafael Rojas Hernández por las observaciones y sugerencias que enriquecieron considerablemente el presente documento. A todos mis profesores, por todos los conocimientos y apoyo compartido a lo largo de mi formación y estancia académica.

Contenido

Lista de figuras

Lista de tablas

Capítulo 1

INTRODUCCIÓN

Una de las obligaciones de los gobiernos municipales es gestionar y llevar un registro minucioso de los apoyos que se ofrecen en sus poblaciones. Algunos ejemplos de apoyos que se entregan actualmente son el de Mujeres Embarazadas, Adultos Mayores, Apoyo a Discapacitados, Jóvenes Construyendo el Futuro y Beca Benito Juárez, entre otros.

Para el registro de apoyos municipales se necesita almacenar el nombre, dirección y demás datos de los solicitantes. Así como generar diferentes reportes, entre los que se incluye el reporte que se entrega a Tesorería, en el que se muestra el número de solicitudes requeridas y los apoyos entregados. Al iniciar esta tesis, en el municipio de Zumpango de Ocampo, Estado de México, se usan hojas de cálculo para gestionar el control del registro de las solicitudes. En escenarios donde la cantidad de usuarios es pequeña y los reportes requeridos son mínimos es válido el uso de herramientas ofimáticas de propósito general (como las hojas de cálculo). En contraste con escenarios que demandan una cantidad mayor de usuarios, el uso de estas herramientas no es escalable ni eficiente, es propenso a errores humanos y genera retrasos obstruyendo la entrega de apoyos municipales.

Por otra parte, los sistemas de software de propósito específico facilitan la captura, almacenamiento, almacenamiento, procesamiento y generación de reportes, ayudando a hacer más eficientes los procesos administrativos.La creación de estos sistemas

requiere del uso de varias tecnologías de cómputo, una metodología de desarrollo, y la aplicación correcta de conceptos de programación. Con el objetivo de generar software útil, eficiente y confiable, se desarrolla un Sistema Web diseñado de manera específica para gestionar el registros de apoyos municipales en Zumpango.

1.1 Planteamiento del problema

La atención a la ciudadana para la entrega de apoyos municipales en Zumpango de Ocampo en el Estado de México, se realiza a través de los Centro de Desarrollo Comunitario (CDC). En enero de 2020, se estima que se realizan 150 solicitudes diarias en cada uno de los CDC y representa un total de 12,000 solicitudes al mes en todos los CDC. Considerando que actualmente las solicitudes se capturan manualmente en hojas de cálculo, esto conlleva a una serie de problemáticas, las cuales se listan a continuación:

- Captura inadecuada de los datos de los solicitantes, al momento del llenado manual de formatos por errores humanos.
- El filtrado de solicitudes se complica, debido a que la información se captura en hojas de cálculo.
- Actualmente no se notifica el estado del trámite de solicitud o seguimiento, y el ciudadano debe de comunicarse o acudir personalmente para poder conocer el estado de su solicitud.

Con el fin de resolver estas problemáticas, se desarrollo un sistema Web que gestione la mayoría de los procesos anteriores, logrando una mejor respuesta a las solicitudes.

1.2 Objetivos

1.2.1 Objetivo general

Diseñar, desarrollar e implementar un sistema para gestionar el registro y seguimiento de solicitudes de apoyos a la ciudadanía del municipio de Zumpango, Estado de México.

1.2.2 Objetivos específicos

Los objetivos específicos de esta tesis son los siguientes:

1. Identificar y levantar los requerimientos para desarrollar el sistema.
2. Diseñar el sistema informático que gestione el registro, bajo un enfoque orientado a objetos y basado en la metodología de desarrollo Scrum.
3. Desarrollar el sistema en lenguaje Java y en PHP, el primero para la parte de escritorio y el segundo para la parte Web.
4. Realizar las pruebas al sistema desarrollado.

1.3 Hipótesis

La hipótesis de esta tesis es la viabilidad de implementar un sistema de registro de solicitudes de apoyo para el municipio de Zumpango, Estado de México. El sistema agilizará el tiempo de captura de información, evitará lo datos incompletos y generará reportes específicos.

1.4 Justificación

La computación tiene un campo de aplicación inmenso y toca prácticamente todas las áreas que existen. Actualmente, es difícil imaginarse algún problema en el que la

computación no pueda realizar alguna aportación o intervención, ya sea como soporte principal o como auxiliar para la realización de diversas tareas.

El desarrollo de un software de propósito específico se justifica por que resuelve un problema real, con el potencial de ayudar a una gran cantidad de solicitantes de apoyos municipales en Zumpango de Ocampo, Estado de México.

1.5 Alcance del proyecto

Con la finalidad de limitar el alcance del sistema las actividades a desarrollar son las siguientes:

- Administrar los usuarios del sistema.
- Gestionar el acceso al sistema mediante usuario y contraseña.
- Almacenar las solicitudes y los datos de los ciudadanos que requieren el apoyo municipal.
- Generar reportes especificas de forma automática.
- Gestionar los apoyos que brinda el Municipio.
- Monitorear el servicio de solicitudes mediante una pagina Web.

El sistema se podrá utilizar mediante una interfaz gráfica capaz de ejecutarse en sistemas operativos como: Windows 7 y posteriores que tengan una máquina virtual de Java versión 8.0.

Capítulo 2

MARCO TEÓRICO

En este capítulo se describen las tecnologías involucradas para el desarrollo del sistema Web. El lenguaje de programación utilizado para la creación de la mayor parte del sistema es Java, por lo que se comienza describiendo un poco de su historia, sus características principales y fundamentos. Además se presentan los controles gráficos utilizados para desarrollar la interfaz gráfica de usuario, así como los conceptos básicos de bases de datos relacionales, empleadas. Para la parte Web de monitoreo de solcitudes, se usó HTML y PHP, por lo que el capítulo comienza con una breve y concisa descripción de estos últimos.

2.1 Tecnologías utilizadas

2.1.1 HTML

Historia

HTML, son las siglas de HyperText Markup Language, que significa lenguaje de marcas de hipertexto. Fue desarrollado por Tim Berners Lee en 1989 en un laboratorio de partículas en Suiza con el fin de compartir información a través de computadoras. La estandarización se logro por un grupo de trabajo que surgió de la primera conferencia de WWW realizada en Ginebra en 1994 [40].

Características de HTML

La estructura del lenguaje se divide en dos secciones las cuales contienen definiciones especificas. En la primera sección se define el titulo de la pagina así como las especificaciones del documento y la segunda sección se encuentra cada componente como: Texto, Imágenes, Vídeos entre otros.

Para la realización de una pagina Web es necesario contar con un editor de Texto (NotePad++, Bloc de Notas, Netbeans, etc.), el cual contendrá el código de la pagina Web y un Navegador de Internet (Chrome, IExplorer, FireFox, Safari, Microsoft Edge, etc.) con el fin de probar la pagina creada.

2.1.2 Etiquetas en HTML

En HTML se utilizan los símbolos de mayor (>) y menor (<) para marcar un comienzo y una finalización de una etiqueta en HTML. En la Tabla 2.1 se definen cada una de las etiquetas en HTML.

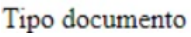
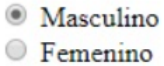
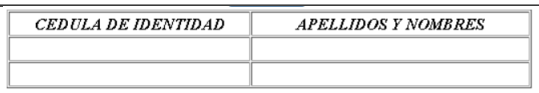
Tabla 2.1: Etiquetas en HTML

Nombre	**Descripción**
$< i >< /i >$	Texto en cursiva.
$< b >< /b >$	Texto en negrita.
$< tt >< /tt >$	Texto mono espaciado.
$< sub >< /sub >$	Indice Inferior.
$< sup >< /sup >$	Indice Superior.
$< strong >< /strong >$	Texto destacado.
$< em >< /em >$	Texto enfatizado.
$< big >< /big >$	Texto grande.

2.1.3 Objetos en HTML

Los objetos en HTML son cada uno de los elementos de los que se conforma un formulario estos son: caja de texto, etiqueta, botón de opción, tabla, área de texto, caja de selección mencionando algunos. En la Tabla 2.2 se describen cada uno de los objetos.

Tabla 2.2: Tipos de objetos en HTML

Nombre	Objeto	Descripción
Caja de Texto		Permite al usuario insertar información textual para ser usada por el programa.
Etiqueta		Indican al usuario que información deben introducir o seleccionar en una pagina Web.
Botón de Opción		Permite indicar al usuario una opción dentro de un formulario.
Tabla		Distribuye la información mediante filas y columnas.
Área de Texto		Permite al usuario insertar una gran cantidad de lineas de texto.
Caja de Selección		Permite al usuario seleccionar una opción dentro de una lista.

2.1.4 PHP

El lenguaje PHP (Hypertext Pre-processor) es un lenguaje interpretado, embebido en paginas HTML por parte del servidor el cual no necesita ser compilado para ejecutarse. El lenguaje se puede ejecutar en la mayoría de los sistemas operativos [4].

Historia

PHP surgió en 1994 como un paquete CGI Common Gateway Interface ("Interfaz de Entrada Común") creado por Rasmus Ledford, y sustituyo un conjunto de scripts en Perl que se utilizaban en una pagina privada. Un nuevo paquete fue lanzado en el año de

1997 con el nombre de PHP/FI el cual contiene las herramientas de Forms Interpreter y un interpretador de comandos en MySQL [5].

Características de PHP

PHP tiene como propósito resolver problemas en plataformas Web y en la actualidad continúa en desarrollo con con funciones nuevas gracias al grupo PHP. A continuación se enumeran algunas características.

1. Velocidad y robustez.
2. Estructurado y orientado a objetos
3. Portabilidad
4. Open Source

2.1.5 Variables en PHP

Para la declaración de variables en PHP se utiliza el símbolo $ seguido de una letra o de un guión bajo. En la Tabla 2.3 se definen cada una de las variables en PHP.

Tabla 2.3: Variables en PHP

Tipo	**Descripción**
Enteras	Números enteros no tienen punto decimal.
Dobles	Números con punto decimal.
De cadena	Texto.
Booleanas	”true” y ”false”.
Nulas	Null.

2.1.6 Java

Historia

Java es un lenguaje de programación que tuvo sus inicios en 1991, y fue escrito en 18 meses en la empresa Sun Microsystems. Al inicio el nombre de este importante lenguaje de programación no fue Java, sino Oak, el cual se le utilizó inicialmente como un proyecto interno de la compañía.

El propósito al desarrollar Oak fue crear un lenguaje que fuera orientado a objetos, y también que fuera multiplataforma, es decir, capaz de ejecutarse en diversas arquitecturas de computadoras, desde aquellas muy simples como un microcontrolador, hasta computadoras con altas prestaciones, sin importar el sistema operativo sobre el cual se ejecutara. Se buscaba que Oak permitiera la creación de software que ejecutara en cualquier computadora o equipo electrónico, ya que fue inicialmente pensado para su uso en la electrónica doméstica, como hornos de micro ondas, lavadoras o refrigeradores.

Oak tuvo por nombre Green, y en 1995 se le nombró como se conoce ahora: Java. El nombre Oak ya era utilizado por la empresa Oak Techologies que diseñaba tarjetas gráficas y lectores de CD, entre otros productos. Oak Technologies cerró operaciones en 2003. Cuando Java se liberó de manera pública, de inmediato tuvo un éxito rotundo [17].

En sus inicios, los programas en Java eran de dos tipos principales: aplicaciones y applets (programa pequeño o programita). Estas últimas ya no se utilizan, es decir, están obsoletas. Las applets eran programas en entorno Web, y fueron particularmente populares en 2011. Se ejecutaban en páginas Web, en específico, en los exploradores Web. Para abrir una página Web con un applet se requería tener instalado un plug-in (programa -casi siempre de tamaño pequeño- que se agrega a un software para añadir alguna funcionalidad). A partir de la versión Java 9, Oracle (la actual propietaria de Java) ya no da soporte para applets [19].

Para hacer uso de Java en un equipo electrónico, se debe instalar previamente la máquina virtual de Java (JVM, por sus siglas en inglés Java Virtual Machine). Los programas Java al ser compilados, generan un código máquina llamado *bytecode* (códigos de bytes), que son interpretados por la JVM, la cual es independiente del hardware que lo ejecuta.

Características de Java

Algunas de las características más importantes de Java son las siguientes:

- **Compilado:** Esto le permite tener dos importantes ventajas sobre otros lenguajes de programación:

  1. Al ser compilado, se evitan errores de sintaxis antes de la ejecución.
  2. Se interpreta en la JVM, por lo que se comporta igual que en otros sistemas operativos.

  A diferencia de los lenguajes que son únicamente compilados, como C o C++, tienen comportamientos distintos dependiendo de la plataforma en la que se ejecuta.

- **Tipado:** En los lenguajes de programación tipados, los valores que son asignados a una variable no pueden ser de cualquier tipo, sino únicamente del mismo que fue declarada la variable, a menos que se utilice el mecanismo de casting. En Java, la tipificación se realiza durante la fase de compilación.

- **Distribuido:** Una de las característica de Java es incluir bibliotecas y herramientas para que los programas tengan la capacidad trabajar con diferentes protocolos y máquinas virtuales en diferentes computadoras [1].

- **Robusto:** Java realiza una búsqueda para notificar si existen problemas o errores en tiempo de compilación y ejecución [1].

- **Arquitectura Neutral:** Java no depende de un software o un hardware especial para ejecutar un programa, ya que genera un *bytecode* que se puede ejecutar sobre cualquier equipo que disponga de la JVM.
- **Multihilo:** Java soporta la creación de programas multiproceso a través de hilos concurrentes.
- **Portable:** Las aplicaciones desarrolladas en lenguaje de programación Java pueden ejecutarse en diferentes sistemas operativos, esto se debe a la representación intermedia denominada bytecode [16].

2.1.7 Etapas de un programa en Java

Normalmente un programa en Java pasa por cinco etapas:

- **Etapa 1 Edición del programa:** Se realiza en un editor de texto (bloc de notas) el código fuente del programa. El desarrollador escribe el programa y lo almacena en el disco, siendo .java la extensión estándar para los archivos Java [13].
- **Etapa 2 Compilación del programa:** Durante el proceso de compilación se verifica que el código fuente cumpla la definición léxica, sintáctica y semántica de Java, si la compilación tiene éxito se genera el archivo con la extensión .class el cual contiene los *bytecodes* [10].
- **Etapa 3 Carga:** La JVM carga en la memoria principal el archivo .class que se creó en la compilación.
- **Etapa 4 Verificador de bytecode:** Mediante la JVM se confirman que todos los códigos de bytes son validos y no presentan violaciones de seguridad, esto ayuda a evitar errores en nuestro sistema.
- **Etapa 5 Interpretación del programa:** Bajo el control del S.O., se interpreta el programa un código de bytes a la vez, ejecutando de esta manera las instrucciones del programa [10].

2.1.8 Estructura de un programa en Java

Un programa detalla como una computadora debe interpretar las ordenes de un programador para que ejecute y realice las instrucciones. Un programador utiliza los elementos que tiene el lenguaje de programación para realizar acciones definidas y dar solución a los problemas [9].

En la figura 2.1 se muestra una programa en lenguaje Java, que imprime la cadena "Hola Mundo". Este código es el ejemplo más simple de un programa que puede crearse en Java, y del que el usuario puede observar un resultado.

```
/*
* Este programa escribe el texto "Hola Mundo" en la consola
* utilizando el método System.out.println()
*/

public class HolaMundo {

   public static void main (String[] args) {
      System.out.println("Hola Mundo");
   }

}
```

Figura 2.1: Código en lenguaje Java

En el código se muestran los siguientes elementos del lenguaje Java: comentarios, definición de clase, definición de método y sentencias.

- **Comentarios:** En las primeras lineas del programa mostrado en la figura 2.1 se aprecia el delimitador de inicio de comentario (/*) y el delimitador de fin de comentario (*/). Los comentarios ayudan a explicar aspectos del código y lo hacen mas legible, esto no afecta al código dado que el compilador los ignora. Los comentarios de una sola línea pueden escribirse usando doble diagonal, es decir: (//) [9].

- **Definición de clase:** En la primera línea del código (después de los comentarios) se define el nombre de la clase, para este ejemplo la clase se llama HolaMundo. Para definir una clase se coloca el carácter de llave abierta ({) al inicio y al final se coloca llave cerrada (}) [9].

- **Sentencia:** Las sentencias que se incluyen dentro del método main sirven para mostrar el texto en consola [9].

Al término del programa es necesario compilar y ejecutar el código para revisar que no existan errores. La compilación de un programa en Java se realiza mediante el programa **javac**.

Tipos de datos en Java

Los datos primitivos en Java representan un único dato simple y no cuentan con capacidades especiales. A continuación en la tabla 2.4 se muestran los diferentes tipos de datos primitivos que existen en Java.

Tabla 2.4: Tipos de datos primitivos de Java

Nombre	**Descripción**
byte	Es el tipo de dato numérico más pequeño en Java, de 8 bits y sus valores van desde -128 a 127.
short	Almacena valores numéricos enteros del tamaño de 16 bits, desde -32,768 a 32,767.
int	Es un tipo de dato numérico de tamaño de 32 bits, con valores desde -2,147,483,648 a 2,147,483,647.
long	Tiene un tamaño de 64 bits, Puede contener enteros en el intervalo -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807.
float	Es un tipo de dato numérico que almacena números reales de tamaño de 32 bits y almacena desde -3.402823e38 a 3.402823e38.
double	Es un tipo de dato numérico que almacena números reales de tamaño de 64 bits y almacena desde -1.79769313486232e308 a 1.79769313486232e308.
boolean	Define un tipo de dato true o false.
char	Cuenta con un tamaño de 16 bits, capaz de almacenar valores de Unicode.

Operadores en Java

En Java existen distintos operadores que se usan con uno o más datos de tipo primitivo. Los tipos de operadores en Java se pueden clasificar como: aritméticos, lógicos, de asignación y relacionales [6].

Operadores Aritméticos En Java los operadores aritméticos corresponden a las operaciones básicas de suma, resta, multiplicación y división, que se usan con números enteros y reales (su resultado siempre es otro valor numérico). Estos operadores se describen en la tabla 2.5.

Tabla 2.5: Tabla de Operadores Aritméticos

Operador	**Descripción**	**Ejemplo**	**Resultado**
+	Suma	9+9	18
-	Resta	8-7	1
*****	Multiplicación	9*9	81
/	División	7/2	3
o/o	Residuo	20 o/o 7	6

Operadores Lógicos

En Java los operadores lógicos o condicionales sólo pueden obtener dos resultados de tipo booleano (falso o verdadero) de tipo booleano, porque utiliza el principio de álgebra de Boole, en la tabla 2.6 se describen los operadores lógicos.

Tabla 2.6: Tabla de Operadores Lógicos

Operador	**Descripción**	**Ejemplo**	**Resultado**
!	Negación - Not	$!false$	true
|	Suma - Or	$true \vert true$	true
&	Producto- And	true & false	false

Operadores de Asignación

Los operadores de asignación como lo indica su nombre, se encargan de otorgar a una variable el resultado de una expresión matemática o el valor de otra variable dependiendo del operador que se utiliza. En la tabla 2.7 se describen los operadores de asignación.

Tabla 2.7: Tabla de Operadores de Asignación

Operador	**Descripción**	**Ejemplo**	**Resultado**
=	Igualación	a=5	5=a
+=	Suma Combinada	a+=b	a=a+b
-=	Resta Combinada	a-=b	a=a-b
*=	Producto Combinado	a*=b	a=a*b
/=	División Combinada	a/=b	a=a/b
&=	Resto Combinado	a&=b	a=a&b

Operadores Relacionales Los operadores relacionales o de comparación se encargan de comparar dos datos de tipo primitivos obteniendo como resultado un dato de tipo booleano (true o false). En la tabla 2.8 se describen los operadores de relacionales.

Tabla 2.8: Tabla de Operadores de Relacionales

Operador	**Descripción**	**Ejemplo**	**Resultado**
==	Igual que	7==80	false
!=	Distinto que	9!=8	true
<	Menor que	9<8	false
>	Mayor que	9>8	true
<=	Menor o igual que	9<=8	false
>=	Mayor o igual que	9 > =8	true

Operadores de Incremento y Decremento

Los operadores de incremento (++) y decremento (- -) son operadores aritméticos, cuya función es aumentar o disminuir el valor de una variable dependiendo el signo y la posición en la que se coloquen estos operadores, es decir, el operador de incremento (++) suma uno a su operando y el operador de decremento (- -) resta uno a su operando. A continuación en la tabla 2.9 se muestra un resumen de estos operadores.

Tabla 2.9: Tabla de Operadores de Incremento y Decremento

Operador	**Definición**	**Descripción**
++	Incremento	• **Primer caso i++:** Se utiliza la variable y después se le suma uno. • **Segundo caso ++i:** Se le incrementa a la variable uno y se utiliza.
−−	Decremento	• **Primer caso i−−:** Se utiliza la variable y después se le resta uno. • **Segundo caso −−i:** Se le resta uno a la variable y se utiliza.

2.1.9 Programación Orientada a Objetos con Java

La Programación Orientada a Objetos (POO) fue desarrollada para resolver problemas grandes y/o complejos, para los cuales los paradigmas de programación estructurada no tenían la capacidad de enfrentarlos apropiadamente [21].

La Programación Orientada a Objetos (POO) se basa en los conceptos de abstracción, clases, encapsulamiento, herencia, polimorfismo, sobrecarga y sobre escritura de métodos.

Abstracción

La abstracción consiste en extraer los comportamientos y características más importantes de un objeto del mundo real (objeto físico), es decir, aquello que lo distingue de los otros objetos. Estos comportamientos y características son utilizados para modelar de manera lógica (mediante código fuente) los objetos físicos. Cabe aclarar que la palabra modelar se refiere al concepto de clase [10].

Clases y objetos

Una clase es una plantilla a partir de la cual se pueden crear objetos, es decir, es un tipo de dato definido por el desarrollador, que incluye la estructura de los datos y operaciones asociadas con la estructura. Un objeto, por otra parte, es una instancia de una clase al igual que una variable es una instancia de un tipo de dato [9].

En pocas palabras, una clase se utiliza para describir de manera lógica a los objetos del mundo real que se requieren modelar en un software.

Por otro lado, las clases en Java están conformadas por atributos (datos) y métodos (comportamientos) a los que se les denomina miembros. En la figura 2.2 se muestra una clase muy simple en Java, que modela a los atributos de cualquier persona (nombre, apellido paterno y apellido materno), y el comportamiento para obtener nombre.

```java
public class Persona {
    String Nombre;
    String Paterno;
    String Materno;

    public Persona(String Nombre, String Paterno, String Materno) {
        this.Nombre = Nombre;
        this.Paterno = Paterno;
        this.Materno = Materno;
    }

      public void obtener(){
        System.out.println("El nombre es: "+Nombre+Paterno+Materno);
    }

}
```

(a) Ejemplo de Clase

```java
public static void main(String[] args) {
Persona a = new Persona("Guillermo","Crescencio","Castillo");
a.obtener();

}
```

(b) Método Main

```
run:
El nombre es: GuillermoCrescencioCastillo
BUILD SUCCESSFUL (total time: 0 seconds)
```

(c) Salida

Figura 2.2: Ejemplo de una Clase en Java

Métodos de una Clase

Los métodos en Java son instrucciones que sirven para operar los campos de una clase y se definen dentro del programa [17]. Los métodos en Java tienen la siguiente sintaxis general:

```
<tipo de retorno> nombre ([parámetros])
```

El tipo de retorno de un método se refiere a los tipos de datos de Java, anteriormente

mencionados en la tabla 2.4. Cabe resaltar que un método puede aceptar cero, uno o más parámetros, y cada parámetro debe de tener especificado su tipo de dato.

En general, los métodos tienen acceso a los datos del mismo objeto, pero no a los de otros objetos. En la figura 2.3 se muestra la declaración de un método (suma), así como su llamado en el método main.

```
public void suma(int x , int y){
    int resultado=x+y;
    System.out.println("resultado:"+resultado);
}
```

(a) Método suma

```
public static void main(String[] args) {
    // TODO code application logic here
    Formulas obj = new Formulas();
    obj.suma(10, 5);

}
```

(b) Invocación del método suma de un objeto

```
run:
resultado:15
BUILD SUCCESSFUL (total time: 1 second)
```

(c) Salida

Figura 2.3: Ejemplo de declaración y llamado de un método en Java

2.1.10 Encapsulamiento

Una de las características de la POO es el encapsulamiento de datos, mediante esta característica se definen los datos y métodos dentro de una clase con el cual se obtiene

una estructura más ordenada, y también más segura (en el sentido de que los datos no se puedan corromper con valores inapropiados), es decir, que pueden protegerse para que no se corrompan [9].

En Java, el encapsulamiento se realiza agregando un modificador de acceso a los miembros. Los modificadores de acceso son public, protected, default y private.

`public:`

El modificador de acceso public permite al programador acceder a las clases, métodos y atributos desde cualquier parte del programa. En la figura 2.4 se muestra el uso del modificador de acceso public.

```
public class Ver{

    public void  mostrar(){
        System.out.println("Java");
}
```

(a) clase Ver

```
public class Imprime {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Ver obj = new Ver();
        obj. mostrar ();


    }
```

(b) clase Imprime

```
run:
Java
BUILD SUCCESSFUL (total time: 0 seconds)
```

(c) Ejecución

Figura 2.4: Ejemplo de modificador de acceso public

El método mostrar() de la clase Ver, es llamado desde la clase principal (Imprime), ya que al ser un método de tipo public se puede acceder a el sin restricciones.

`protected:`

El modificador de acceso protected permite al programador acceder a los miembros y atributos desde la misma clase, y desde las subclases del mismo paquete. En la figura 2.5 se muestra el uso de protected como modificador de acceso.

```
class AccesoProtected{
    protected void mostrar (){
    System.out.println("Java");
}
}
```

(a) clase AccesoProtected

```
public class Imprime extends AccesoProtected{
    public static void main(String[] args) {

        AccesoProtected obj =  new AccesoProtected();
        obj. mostrar ();
    }
}
```

(b) clase Imprime

```
run:
Java
BUILD SUCCESSFUL (total time: 0 seconds)
```

(c) Ejecución

Figura 2.5: Ejemplo de modificador de acceso protected

El acceso protectec es un modificador que se encarga de proteger los métodos de la clase AccesoProtected (mostrar()), es decir, evita que el programador acceda a la clase si no hereda sus métodos.

`default:`

El acceso default se declara cuando no se usa ninguno de los modificadores de acceso: private, protected o public. Sólo se permite acceder a los miembros y atributos si pertenecen a la claseo a las subclases del mismo paquete. A continuación se muestra

en la figura 2.6 el uso del default como modificador de acceso.

```
 * @author Guillermo
 */
class AccesoDefault{
    void   mostrar(){
    System.out.println("Java");
}
}
```

(a) clase AccesoDefault

```
public class Llamado{
    public static void main(String[] args) {
        AccesoDefault obj = new AccesoDefault();
        obj.mostrar();
    }

}
```

(b) clase Llamado

```
run:
Java
BUILD SUCCESSFUL (total time: 0 seconds)
```

(c) Ejecución

Figura 2.6: Ejemplo de modificador de acceso default

Acceso default es un modificador de acceso que no requiere una palabra reservada, ya que se declaró el método mostrar() sin ningún modificador de acceso,es decir, no se declara o escribe, por lo que no es visible y existe al no ser definido como se muestra en la figura 2.6.

`private:`

El modificador de acceso private, es un modificador que se utiliza cuando se requiere restringir el acceso a los métodos y atributos de una clase, y sólo se le permite al programador ingresar desde la misma clase. En la figura 2.7 se muestra el uso private como modificador de acceso.

```
class AccesoMiembros1{
    private void mostrar(){
    System.out.println("Java");
}
}
```

(a) clase AccesoMiembros1

```
public class AccesoMiembros2{
    public static void main(String[] args) {
        // TODO code application logic here
        AccesoMiembros1 obj =  new AccesoMiembros1();
        obj.mostrar();
    }
}
```

(b) clase AccesoMiembros2

```
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - mostrar() has private access in metodosdeacceso.accesoMiembros1
```

(c) Ejecución

Figura 2.7: Ejemplo de modificador de acceso private

El propósito del código anterior es demostrar que no se permite acceder a un método de tipo private desde una clase diferente. Como se demuestra en el código de la figura2.7 existe un error en la ejecución, ya que se llama al método mostrar() de la clase AccesoMiembros1 desde la clase AccesoMiembros2 y estos se encuentran en diferentes clases.

Sobrecarga de Métodos

En Java se pueden declarar varios métodos con un mismo nombre dentro de una clase con la condición de que cada uno reciba parámetros diferentes, ya sea en cantidad o en tipo; a esto se le conoce como sobrecarga de métodos [6]. Cuando se llama a alguno de los métodos sobrecargados, el compilador sólo hace referencia al método basándose en el tipo de parámetros que solicita para realizar las instrucciones del método llamado. La sobrecarga de métodos suele utilizarse para generar varios métodos que comparten el mismo nombre y que realizan tareas similares. En la figura 2.8 se muestra un ejemplo del uso de sobrecarga de métodos.

```
public class Suma{
    public void suma(int x,int y){
    int resultado=x+y;
        System.out.println(resultado);
    }

    public void suma(int x,int y,int z){
    int resultado=x+y+z;
        System.out.println(resultado);
    }
}
```

(a) Código de Sobrecarga del método Suma

```
public class Clas{
    public static void main(String[] args) {
        // TODO code application logic here
        Suma obj =  new Suma();
        obj.suma(4,4);
        obj.suma(4,4,8);
    }
}
```

(b) Ejecución

```
run:
8
16
BUILD SUCCESSFUL (total time: 0 seconds)
```

(c) Salida

Figura 2.8: Código uso de Sobrecarga de Métodos

En la figura 2.8 se demuestra el código de la clase Suma, que contiene dos métodos con el mismo nombre (suma), pero cada uno realiza una tarea diferente; en el primero se reciben dos parámetros de tipo entero (x,y) en el cual se realiza una suma con los dos parámetros e imprime el resultado, en el segundo se reciben tres parámetros de tipo entero(x, y, y z) se suman los tres parámetros e imprime el resultado.

Herencia

La herencia, es un mecanismo que permite solucionar problemas como el código duplicado, debido a que se define una clase general que contenga los atributos generales de las subclases [6]. A la clase principal donde las clases derivadas surgen se le denomina superclase o clase padre, y a las clases derivadas se les conoce como subclases. La superclase sirve de modelo para las subclases, quienes heredan las características o comportamientos [19]. La herencia permite que se produzca un software más seguro, rápido y se pueda ajustar de manera sencilla a las peticiones del cliente. La relación que se establece entre las clases derivadas y la súper clase es del tipo "es-un", por ejemplo, si la clase Carro hereda de la clase Vehículo, se lee "Carro es un Vehículo", lo que suena lógico tanto en el mundo real como en un modelo lógico de software. En Java, se usa la palabra reservada `extends` para indicar la relación de herencia entre dos clases. Una característica de Java es que sólo una subclase puede tener una superclase directa, es decir, no se soporta herencia múltiple. En la figura 2.9 se muestra un ejemplo del uso de herencia.

En la figura 2.9 se muestra un ejemplo del uso de herencia, donde se demuestra el código de tres clases: Vehiculo, Moto y Trabajo. En la súper clase o clase padre (Vehiculo) se declaran tres variables: combustible, color y marca.

La subclase Moto utiliza el método descripcion(), en el cual se asigna un valor a las variables: combustible, color y marca, atributos de la clase padre (Vehiculo).

En la clase Trabajo, se crea una instancia de tipo Moto y se llama al método descripcion() de la subclase Moto, al ejecutar el programa se muestran los valores asignados a los atributos.

```java
package trabajo;

/**
 *
 * @author Guillermo
 */
public class Vehiculo {
    String combustible;
    String color;
    String marca;

}
```

(a) Clase Padre

```java
public class Moto extends Vehiculo {

    public void descripcion(){
    combustible="gasolina";
    color="azul";
    marca="Italika";
        System.out.println(combustible);
        System.out.println(color);
        System.out.println(marca);


    }
```

(b) Clase Hijo

```java
public class Trabajo {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Moto obj = new Moto();
        obj.descripcion();
    }

}
```

(c) Método main

```
run:
gasolina
azul
Italika
BUILD SUCCESSFUL (total time: 0 seconds)
```

(d) Salida

Figura 2.9: Código uso de Herencia

Polimorfismo

Es la propiedad que le permite a una operación tener el mismo nombre en clases diferentes, y actuar de modo distinto en cada una de ellas, en Java el polimorfismo permite que un objeto determine el tiempo de ejecución de las operaciones a realizar. Cabe destacar, que tiene la característica de ejecutar el método clases diferentes, por ejemplo: el método limpiar() puede aplicarse para limpiar un JTextArea (campo para la entrada de texto multilínea) o un un JTextField (campo de texto) [25]. El polimorfismo se realiza en tiempo de ejecución, por lo que se debe de esperar hasta que el programa se esté ejecutando para especificar el tipo de objeto que estará en una referencia particular del objeto [19]. A continuación en la figura 2.10 se muestra un

ejemplo del Polimorfismo.

```
public class Animal {

    String tipo;

    public void hacerRuido() {
        System.out.println("Que ruido puedo hacer?");
    }

}
```

(a) Clase Animal

```
public class Gato extends Animal {

    @Override
    public void hacerRuido() {
        System.out.println("miau");
    }
}
```

(b) Clase Gato

```
public class Perro extends Animal{

    @Override
    public void hacerRuido() {
        System.out.println("Guau!");
    }
}
```

(c) Clase Perro

```
public class Tarea  {
public static void main(String[] args) {
    Animal a = new Gato();
    test(a);


                        }
public static void test(Animal animal) {
        animal.hacerRuido();  }  }
```

(d) Método main

```
run:
miau

BUILD SUCCESSFUL (total time: 0 seconds)
```

(e) Salida

Figura 2.10: Código uso de Polimorfismo

En la figura 2.10 se demuestra el código de cuatro clases: Animal, Gato, Perro y Tarea, el cual ejemplifica el uso del Polimorfismo. En la clase padre o súper clase

(Animal) se utiliza el método hacerRuido() el cual imprime en consola "Que ruido puedo hacer?" cuyo objetivo es verificar que el método funcione correctamente.

En la subclase Gato se sobreescribe el método hacerRuido(), para dar otro tipo de funcionalidad el cual imprime "miua" en salida.

Asimismo en la subclase Perro se sobreescribe el método hacerRuido(), para dar otro tipo de funcionalidad en el cual se imprime "Guau!" en consola.

En la clase Tarea, el método main() crea una instancia de la clase Animal de tipo Gato y utilizando el método test() de la clase Tarea recibe un parámetro de tipo Animal que ejecuta el método hacerRuido(), de la clase gato y en consola se imprime "miua".

2.1.11 Diseño de interfaces gráficas con Java

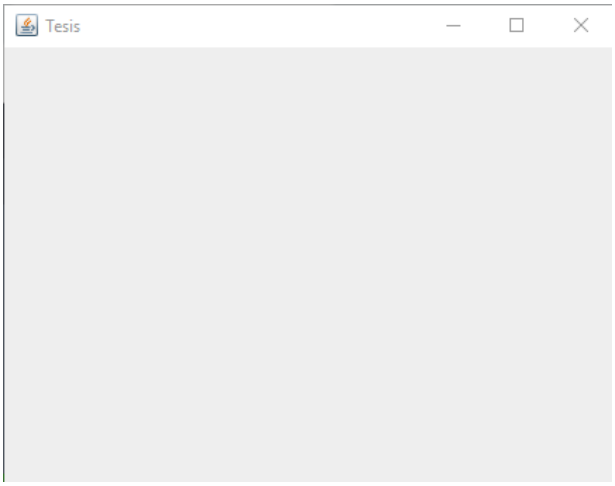
Una interfaz gráfica, permite al usuario manipular de manera más intuitiva la información, ya que proporciona componentes con los que el usuario se pueden familiarizar dentro de una aplicación y así mejorar la productividad de su trabajo [18].

En la realización de aplicaciones Java, con interfaz gráfica (componentes gráficos que permiten la interacción entre el usuario y la aplicación), es necesario, importar la biblioteca Swing para usar las herramientas, tales como: botones, cajas de texto y listas desplegables.

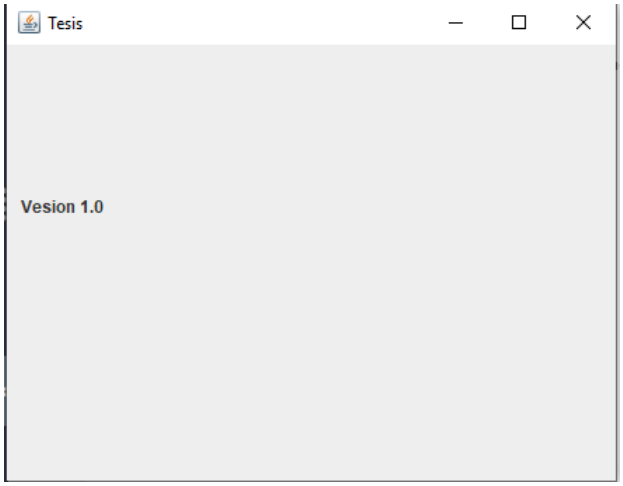
La biblioteca Swing está formada por las clases: JFrame, JLabel, JButton, JTextField, JTextArea, JComboBox, mismas que se describen en la Tabla 2.10 y se representan gráficamente en la figura 2.11, ya que son usadas en el desarrollo del sistema.

Tabla 2.10: Tabla de componentes de la biblioteca Swing

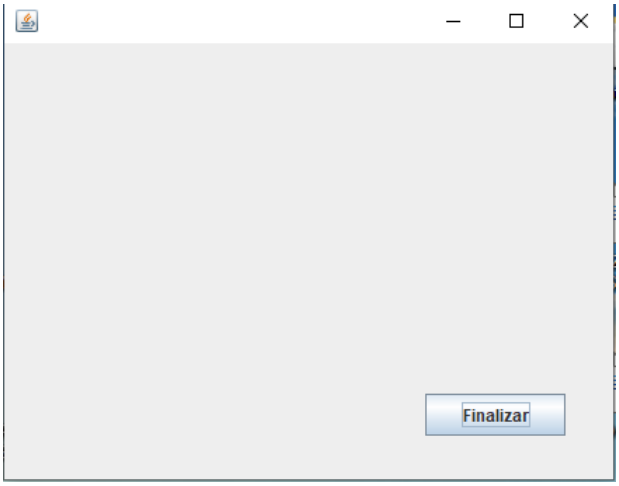
Nombre	**Descripción**	**Métodos Representativos**
JFrame	Es un contenedor en el cual se colocan diferentes componentes de la biblioteca Swing de Java.	**setSize()**: Ajusta el tamaño del JFrame. **setLocation()**: Coloca el JFrame en una posición especifica.
JLabel	Permite colocar una etiqueta en un contenedor en el que se puede mostrar una línea de texto o una imagen.	**setForeground()**: Permite asignarle un color al texto. **setText()**: Define el texto de nuestro JLabel.
JButton	Permite presentar un botón dentro de un JFrame con el que se puede activar una o múltiples funcionalidades en tiempo de ejecución.	**getLabel()**: Obtiene el texto colocado en el botón.
JTextField	Es un componente de la biblioteca Swing, que se presenta como un recuadro de texto una sola línea y se coloca dentro de un JFrame.	**getText()**: Obtiene el contenido del JTextField. **setText()**: Muestra contenido dentro de JTextField.
JTextArea	Una de las propiedades de JTextArea es contar con fila y columna que sirven para determinar el tamaño del componente.	**getText()**: Obtiene el contenido del JTextArea. **setText()**: Muestra contenido dentro de JTextArea.
JComboBox	Es un componente que permite mostrar una lista de opciones desplegable en la que el usuario selecciona un valor.	**getItemAt()**: regresa el objeto que esta en la posición dada por index. **getItemCount()**: regresa la cantidad de elementos que hay en la lista.



(a) JFrame

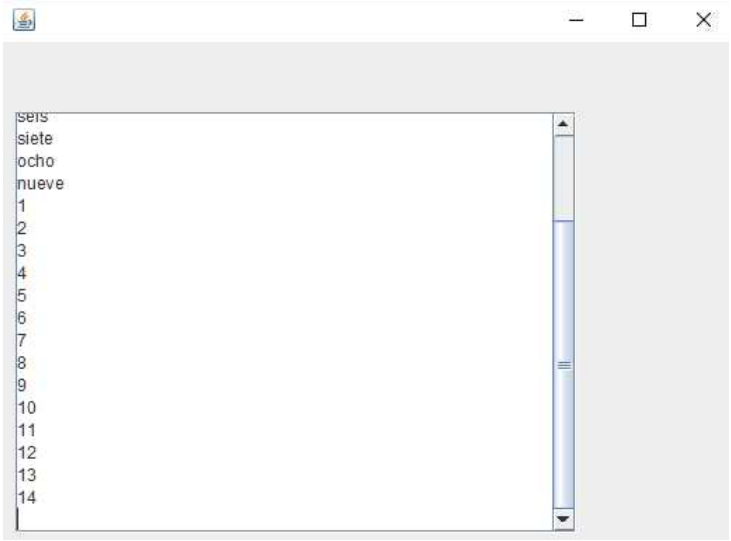


(b) JLabel

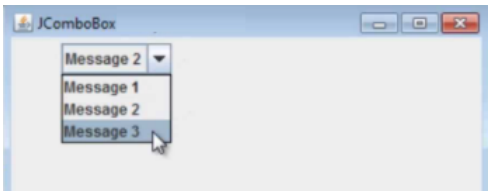


(c) JButton

(d) JTextField



(e) JTextArea



(f) JComboBox

Figura 2.11: Controles Swing utilizados

2.2 Bases de datos

2.2.1 Concepto y origen de las Bases de Datos

Las bases de datos aparecieron a mediados de los años sesenta, y eran representadas como un conjunto de archivos planos, estos archivos no tenían alguna relación entre sí, y los datos sólo mostraban las referencias de manera simbólica, tenían demasiada redundancia y la integración de la información que se representaba era de muy poca ayuda [22].

El problema que predominaba en ese entonces era la manera de estructurar el almacenamiento de la información, Al principio los dispositivos que realizaban la tarea de entrada y salida de la información almacenada, era mediante un puerto serial, por lo cual la estructuras de almacenamiento tenían que ser de manera secuencial ya que el tiempo de respuesta era muy extenso y dependía en su totalidad del hardware.

Con la llegada de dispositivos de almacenamiento, que permiten un proceso de entrada y salida de información más rápido, se desarrollaron procedimientos de acceso directo a la información ya que estos eran más rápidos, por no depender del hardware que se usaba para realizar las consultas o ingresar la información.

Las bases de datos son utilizadas como una solución para la mayoría de los problemas del mundo real en los que se debe conservar y mantenerla información disponible para su consulta. Una de las ventajas de las bases de datos es que se pueden encontrar y compartir de manera simultánea los datos a diferentes usuarios, siempre y cuando tengan los permisos adecuados.

Debido a el gran avance de la computación, y con la gran importancia del internet, las bases de datos se encuentran con desafíos más grandes, como el número de usuarios, velocidad, fiabilidad, simplicidad, integridad, seguridad y privacidad.

Actualmente, una base de datos se concibe como un conjunto de datos relacionados entre sí, que sirve para almacenar grandes cantidades de información y que mediante un sistema permite la manipulación o consulta de la información. Las bases de datos tradicionales se organizan por campos, registros y archivos.

2.2.2 Características de las bases de datos

La información con la que se forma una base de datos debe de cumplir una serie de características, las cuales se describen a continuación:

Afinación: Según la autora Luque et al.(2002) en su libro "Bases de Datos desde Chen hasta Codd con Oracle", la afinación hace referencia a la organización física de la información de la base de datos [35]. Es decir, el lugar físico donde se almacenan los bancos de información deben de ser flexibles a las modificaciones que pueda tener la base de datos y no verse afectada por diferentes procedimientos u otras maneras de representar la información.

Interfaz: Luque en su libro menciona que las necesidades de la organización cambian continuamente y, por lo tanto, cambia la información correspondiente al subsistema o dominio del problema de la misma [35]. Es decir, las bases de datos deber de ser capaces de adecuarse a diferentes software que ayuden a ser gestionadas y modificadas en el caso de que la organización o la empresa lo requieran.

Seguridad: La seguridad de una base de datos hace referencia a la capacidad de ésta para proteger los datos contra su pérdida total o parcial por fallos del sistema o por accesos accidentales o intencionados a los mismos [35]. Luque et al.(2002). Esto se refiere a que la seguridad de una base de datos no sólo debe hacer referencia al acceso, información y personas no autorizadas, sino que, se debe proteger la información en el caso de haber una pérdida total o parcial por una falla en el sistema, ya sea accidental o intencional.

Acceso: Luke describe, describe que, una base de datos debe ser capaz de responder, en un tiempo aceptable, a cualquier consulta sobre la información que mantiene, sin restricciones graves en cuanto a los ítem, relaciones, formato, etc., solicitados en la misma, y respondiendo al usuario rápidamente [35]. Esto quiere decir, que las bases de datos debe ser capaces de soportar múltiples consultas a la vez, en un tiempo aceptable y responder a el usuario de una manera óptima.

Redundancia: Una de las principales razones por las que surgió la tecnología de

las bases de datos fue el evitar el alta redundancia que se presentaba en los sistemas de procesamiento de información debido al uso de archivos con estructuras planas, según menciona Luque en su libro [35]. Dicho de otra forma, la redundancia implica que se almacene el mismo valor registrado en nuestra base de datos, lo que provoca que ocupemos mas espacio y que nuestras consultas sean más inconsistentes. El objetivo de tener una base de datos es desaparecer la redundancia para optimizar el sistema, ya que si existen varias copias de un mismo registro no se puede garantizar la consistencia de la información cuando las copias de los registros son usadas al mismo tiempo.

Desempeño: Luque plantea que, las bases de datos deben asegurar un tiempo de respuesta adecuado en la comunicación hombre-máquina, permitiendo el acceso simultáneo al mismo o distinto conjunto de ítems de datos por el mismo o distintos procedimiento [35]. Señalando que las bases de datos deben cumplir con un tiempo de respuesta óptimo, permitiendo a diferentes usuarios consultar y modificar los registros sin carecer de velocidad en las respuesta del sistema.

Integridad: Con base en el libro "Bases de Datos desde Chen hasta Codd con Oracle", la integridad de una base de datos hace referencia a la veracidad de los datos almacenados con respecto a la información existente en el dominio del problema que trata la misma [35]. Este concepto sugiere que los procedimientos que manejan la información deben de asegurar que el sistema pueda garantizar la integridad de la información, en el caso de existir un error en el sistema.

Simplicidad: Luque indica que la base de datos representa el dominio de un problema que se necesita tratar computacionalmente. La naturaleza de este problema puede ser muy variada y, por tanto existir en el mismo número de objetos variables que se relacionan de múltiples formas [35].

De acuerdo con Luque, una base de datos debe representar de manera clara la naturaleza del problema a solucionar, además de que se garantice que está estructurada de tal manera que pueda llevar a cabo modificaciones en la base de datos, de tal forma que al añadir más relaciones e información no ocasione un conflicto en el desempeño.

Representación: Empleando las palabras de Luque et al.(2002), la

representación es un procedimiento, un programa de aplicación, que maneja la información correspondiente a un problema puede, por tanto, tener en cuenta sólo parte del conjunto de información, mientras que otro procedimiento puede considerar a otro conjunto diferente, o no, de información del mismo problema [35]. El concepto anterior sugiere que se utiliza un programa que permite modificar o actualizar los registros de la base de datos en caso de que se requiera, permitiendo que diferentes procedimientos (modificar o actualizar) puedan generar registros a partir de la información existente.

2.2.3 Tipos de bases de datos

Los modelos de bases de datos son abstracciones, que permiten el desarrollo de un sistema eficaz, por lo general, se refiere a una serie de pasos, y conceptos matemáticos [7].

Existen cuatro tipos de bases de datos: jerárquicas, de red, orientadas a objetos y relacionales.

Bases de Datos Jerárquicas: Estructuran la información de tal manera, que forman un árbol donde todos los datos dependen de una entidad padre y estos pueden tener cero, uno o más hijos. La relación que existe es de uno a muchos y para acceder a una entidad siempre se debe de hacer desde la raíz y recorrer las entidades hijos. Este tipo de bases de datos se utilizan en buscadores.

Bases de Datos en Red: Para eliminar el conflicto que existe dentro de las bases de datos jerárquicas, se crearon diferentes tipos de relaciones, ahora una entidad hijo puede tener varias entidades padres, denominándose propietario.

Estas son similares a las jerárquicas con la diferencia de que pueden tener más de un padre. Con el modelo de red se pueden representar relaciones varios a varios, es decir, una tabla puede tener distintos registros asociados a otra tabla y viceversa [37].

Bases de Datos Orientadas a Objetos: Representan la información en objetos, los cuales, son utilizados en la programación orientada a objetos por su alto rendimiento de almacenamiento, ya que este permite que las clases padres hereden atributos a sus

clases hijas [23]. Por la simplicidad que representa es uno de los modelos más vigentes, y que en la actualidad siguen desarrollándose. Casi todos los sistemas usan las bases de datos orientadas a objetos.

Bases de Datos Relacionales: Según el artículo "Modelo Entidad-Relación" [33], el esquema relacional fue desarrollado por Edgar Frank Codd a finales de los años sesenta a partir de la teoría de conjuntos. Los datos relacionales se almacenan en una base de datos que pueden tener las mismas características. Al conjunto de los datos de bases relacionales se le denomina dominio; el cual se refiere aún conjunto finito de datos.

2.2.4 Sistemas Gestores de Bases de Datos

Los Sistemas Gestores de Base de Datos (SGBD), son aplicaciones que permiten a los usuarios realizar diversas modificaciones a las bases de datos como: procesar, describir, administrar y recuperar la información, otorgando seguridad en caso de existir pérdida total o parcial de la información. Este tipo de aplicación permite al usuario tener un mejor control de la información mediante una interfaz [36].

El SGBD gestiona las cuatro operaciones fundamentales: inserción, consulta, actualización y borrado.

2.2.5 Tipos de SGBD

Existen diferentes tipos de SGBD, los cuales se agrupan en tres modelos diferentes: Lógico, Usuario y Aplicación.

Modelo lógico

1. Modelo Jerárquico.
2. Modelo de Red.
3. Modelo Relacional.
4. Modelo Orientado a Objetos.

Modelo de Usuarios

1. Monousuario.
2. Multiusuario.

Modelo de Aplicación

1. Centralizados.
2. Distribuidos: homogéneos y heterogéneos.

2.2.6 Modelo Entidad-Relación

Este modelo fue presentado a mediados de los años setenta por el Dr. Peter Pin-Shan Chen, como un medio para representar la visión de un sistema de forma global mediante un conjunto de representaciones gráficas y lingüísticas [20] [32].

Las características de este modelo permiten representar cualquier tipo de sistema y nivel de abstracción, esto ayuda a la representación de problemas que vayan a ser tratados mediante un sistema [34].

Existen conceptos básicos para comprender el modelo Entidad-Relación son:

Conjunto: Se define conjunto como una serie de objetos elementales mediante una función de pertenencia.

Entidad: La entidad corresponde a la caracterización de objetos del mundo real los cuales son definidos y caracterizados del resto de los objetos.

Intensión y Extensión: La intensión define los requisitos (atributos) de información de una entidad, por ejemplo, la entidad VEHICULO cuenta con los atributos: marca, color, precio y combustible.

La extensión define los atributos de las entidades, por el ejemplo, en la entidad VEHICULO se describen los atributos de la siguiente manera: "Italika", "Azul", "8.700" y "Gasolina".

Dominio: Es el conjunto de valores permitidos para un atributo, por ejemplo: el atributo precio de la entidad VEHICULO sólo permite colocar números.

Relación: Se le denomina relación a un conjunto que representa una correspondencia (número de entidades a las que puede asociarse otra entidad mediante una relación) entre dos o más conjuntos. Las relaciones pueden ser binarias, ternarias o n-arias y pueden ser definidas como el producto cartesiano (operación entre dos conjuntos) de los conjuntos que intervienen en la relación.

Atributo: Identifica la semántica (estudio de diversos aspectos del significado) de un dominio para la descripción de un problema; es decir, identifica el significado del dato, que forma parte del sistema en un mundo real.

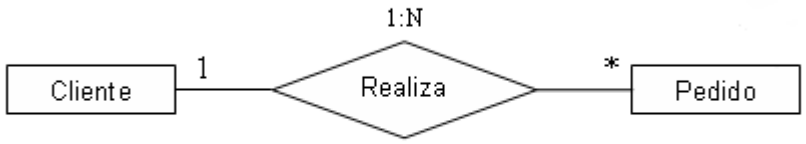
Entidad fuerte: Es aquella que no depende de otra entidad para su existencia, por ejemplo: en una hospital, la entidad DOCTOR no depende de una entidad PACIENTE, dado que al no existir un paciente no se elimina la entidad DOCTOR.

Interrelación: Describe la relación que existe entre las entidades, por ejemplo: la interrelación (relación mutua) que existe entre la entidad DOCTOR y PACIENTE.

Clave primaria o principal (primary key): Es la clave candidata que selecciona el programador de la BD (Base de Datos). Una clave candidata no puede contener valores nulos (vacíos), debido a que se utilizan como identificadores de un registro en base de datos [24].

El modelo entidad-relación es un modelo que se utiliza en la actualidad debido a que permite dar una solución de manera rápida y sencilla, por lo que se decidió utilizar durante este proyecto de investigación; "Propuesta de un sistema Web para registro de solicitudes de apoyos Municipales en Zumpango, Estado de México", ya que este proyecto realizó un análisis del problema para definir los campos del modelo Entidad-Relación (conjunto, entidad, dominio, relación, atributo, entidad fuerte , interrelación, clave primaria, intensión y extensión) esto ayuda a plasmar el problema del mundo real al digital. Es decir, va a permitir que se realicen los registros de manera eficaz al tener un mejor control de las solicitudes, lo que garantiza una respuesta más eficiente al cubrir las necesidades (registro de usuarios, registro de solicitudes, registro de apoyos y generación de informes) del área de Bienestar Social, del municipio de Zumpango ubicado en el Estado de México.

La figura 2.12 muestra el diagrama de un Modelo Entidad-Relación orientado a la entrega de un paquete.



(a) Diagrama

Figura 2.12: Modelo Entidad-Relación

En la figura 2.12: se ejemplifica la entrega de un paquete a través del Modelo Entidad-Relación, en el que se representan dos entidades (Cliente y Pedido) y la relación que existe (uno a varios).

2.2.7 Lenguaje SQL

Structured Query Language (SQL) o como lo indica su nombre en español Lenguaje Estructurado de Consultas, define el lenguaje estándar ANSI/ISO de definición, manipulación y control de bases de datos relacionales [28]. El desarrollo de SQL(Lenguaje Estructurado de Consultas) tiene origen a principios de los años setenta por International Business Machines Corporation (IBM), en el Laboratorio de Investigación de San José. International Business Machines Corporation implemento el lenguaje Structured English Query Language (SEQUEL), como parte de un proyecto llamado System R. para la manipulación y control de las bases de datos, y actualmente conocido como SQL [15].

Es decir, el SQL puede definir, manipular y controlar una base de datos relacional. SQL es un lenguaje de alto nivel para un conjunto de registros, esto hace referencia a que una línea de código permite modificar un número grande de registros.

2.2.8 Bases de Datos Relacionales

Una base de datos relacional permite la relación de forma lógica de un conjunto de datos llamado: tabla.

Una tabla, es un lugar donde se almacenan datos que tienen características similares y conforman la estructura de la base de datos. Las tablas cuentan con filas y cada una de ellas tiene campos que albergan valores ya definidos. A continuación en la figura 2.13 se describe una tabla en consola, esta tabla contiene dos campos (nombre y edad). El campo nombre es de tipo varchar (permite una cadena de caracteres) y el campo edad es de tipo int (permite colocar números enteros).

```
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| nombre | varchar(30) | YES  |     | NULL    |       |
| edad   | int(11)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.015 sec)
```

Figura 2.13: Tabla en MySQL

2.2.9 Ejemplos de Bases de Datos Relacionales

En la actualidad existen diferentes gestores de bases de datos relacionales. En la siguiente tabla 2.11 se describen los más usados.

Tabla 2.11: Ejemplos de Bases de Datos Relacionales

Nombre	**Licencia del SGBD**	**Descripción**
MYSQL	LIBRE	MySQL es un sistema gestor de bases de datos (SGBD) usado por su facilidad y rendimiento [38].
SQL SERVER	PAGA	Es una base de datos de Micrososoft se programa de manera flexible mediante el lenguaje visual studio,visual basic,o C.
ORACLE	PAGA	Se utiliza para un nivel empresarial ya que permite almacenar un gran volumen de datos.
ACCESS	PAGA	Es una base de datos de Microsoft que viene incorporado en microsoft office no es recomendable para almacenar grandes cantidades de datos.

2.2.10 Tipos de Sentencias en SQL

El lenguaje SQL permite realizar mediante líneas de código (sentencia); funciones de definición, control y gestión de la base de datos.

Las sentencias en SQL se clasifican según su finalidad, dando origen a tres lenguajes que se describen en la siguiente tabla 2.12.

2.2.11 Tipos de Datos en SQL

Para cada atributo es necesario especificar un dominio, en el cual se puedan tomar valores de esa propiedad del objeto [35]. El lenguaje SQL reconoce una serie de tipos de datos que ya están predefinidos, los cuales se describen en la siguiente tabla 2.13.

Tabla 2.12: Sentencias SQL

Nombre	**Función**
Lenguaje de Definición de Datos (DDL)	Creación de tablas, relaciones entre las tablas, claves ,etc.
Lenguaje de Manipulación de Datos (DML)	Creación, modificación y consulta de registros.
Lenguaje de Control de Datos (DCL)	Administración y control de las bases de datos.

Tabla 2.13: Tipos de datos SQL

Nombre	**Descripción**
CHAR	Representa una cadena de caracteres de longitud fija de 2.000 bytes [35].
VARCHAR	Representa una cadena de caracteres de longitud variable de hasta un máximo de 4.000 caracteres [35].
NUMBER	Representa una información numérica de longitud fija o flotante en un rango de 1 a la -130 a 10 a la 125 [35].
BOOLEAN	Representa información lógica que puede tomar valores TRUE, FALSE y NULL [35].
DATE	Representa información cronológica de fechas validadas, incluyendo horario desde el 1 de Enero del 4712 a.C hasta el 31 de Diciembre del 4712 d.C. [35].
RAW	Representa datos binarios de tamaño máximo de 2000 bytes [35].
LONG RAW	Representa datos binarios de tamaño máximo de 2 Gb [35].
RAWID	Tipo de dato binario que representa la dirección de una tupla de una tabla [35].
CLOB	Representa objetos de caracteres de hasta 4 Gb [35].
BLOB	Representa objetos binarios de hasta 4 Gb [35].

2.2.12 Restricciones en SQL

Una restricción, consiste en añadir una regla a una columna (atributo) de una tabla. Esta regla se encargara de verificar que al momento de insertar un valor a la tabla, esta cumpla con la regla, en el caso de ser correcto se inserta el valor en la tabla, de lo contrario lo descartará. A continuación en la siguiente tabla 2.14 se describe cada una de las restricciones en SQL.

Tabla 2.14: Tipos de restricciones SQL

Nombre	**Descripción**
NOT NULL	Indica que el atributo no permite valores nulos (vacíos).
PRIMARY KEY	Indica que el atributo es la clave principal y por lo tanto no se puede repetir.
UNIQUE	No permite que exista el mismo valor en el atributo.
FOREING KEY	Es una clave (campo de una columna) que sirve para relacionar dos tablas [35].

2.2.13 Creación de una Base de Datos

Las Bases de datos se encargar de guardar de manera organizada las tablas, las cuales guardan un conjunto de datos diferentes, están divididas entre filas y columnas.

En la figura 2.14 se ejemplifica la creación de una base de datos, para realizar la conexión se utiliza la sentencia **mysql -u root -p** es necesario introducir el password, una vez ingresado se crea la base de datos tesis con la sentencia **CREATE DATABASE tesis;**, una vez realizado se sugiere revisar las bases de datos con la sentencia **SHOW DATABASES;** en el que se ve tesis en la tabla Database y para usarla se utiliza **USE tesis;**

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.14-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE tesis;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sistema_registro   |
| tesis              |
| test               |
+--------------------+
7 rows in set (0.003 sec)

MariaDB [(none)]> USE tesis;
Database changed
MariaDB [tesis]>
```

Figura 2.14: Sintaxis creación de una Base de Datos

2.2.14 Definición de Tablas

Una tabla, es una estructura básica en la que se representan datos que se organizan a través de un formato de filas y columnas, similar a una hoja de cálculo. Cada fila representa un registro ingresado y cada columna representa un campo de la tabla [8].

En la figura anterior 2.15 se muestra la sintaxis SQL en consola para crear una tabla de nombre alumno, en el cual se insertan tres atributos: nombre, edad y grado, donde se define el dominio de cada atributo, el nombre es de tipo varchar (cadena de caracteres), edad es de tipo entero (valores numéricos) y grado de tipo varchar.

```
[tesis]> CREATE TABLE alumno(
-> nombre VARCHAR(20), edad int(20),
-> grado VARCHAR(20));
Query OK, 0 rows affected (0.047 sec)
```

Figura 2.15: Sintaxis creación de una tabla

2.2.15 Inserción de datos en Tablas

Para añadir un registro a una tabla, debemos de hacer uso de la sentencia **INSERT**, de tal forma, que debemos colocar un valor para cada una de las columnas (atributos) de la tabla, de manera en la que fueron añadidos al momento de crearla.

En la figura anterior 2.16 se muestra un ejemplo del uso de la sentencia INSERT, el cual sirvió para añadir un registro en la tabla alumno donde sus atributos son: nombre ,edad y grado, es decir, los valores insertados son: "GUILLERMO", "27" y "QUINTO".

```
[tesis]> INSERT INTO alumno VALUES('GUILLERMO ','27','QUINTO');
Query OK, 1 row affected (0.011 sec)
[tesis]> select * from alumno;
+------------+------+--------+
| nombre     | edad | grado  |
+------------+------+--------+
| GUILLERMO  | 27   | QUINTO |
+------------+------+--------+
```

Figura 2.16: Ejemplo de inserción de datos en una Tabla

2.2.16 Consulta de datos en Tablas

Para realizar una consulta en una tabla de la base de datos se debe escribir en consola la sentencia; SELECT, seguido de los nombres de los atributos los cuales se separaran a través de comas, quedando de la siguiente forma: nombre, edad y grado. Después de escribir los nombres de los atributos se coloca la sentencia FROM y por último el nombre de la tabla. En caso de realizar una consulta general se coloca un asterisco (*) entre SELECT y FROM.

En la figura 2.17 se ejemplifica la consulta de la tabla alumno la cual tiene el propósito de almacenar los datos de los estudiantes, para acceder a tal información se deberá colocar en consola la siguiente sentencia: **SELECT * FROM alumno**, el cual describe el contenido y agrupa la información en columnas (nombre, edad y grado).

```
[tesis]> SELECT *FROM alumno;
+------------+------+--------+
| nombre     | edad | grado  |
+------------+------+--------+
| GUILLERMO  | 27   | QUINTO |
| DIANA      | 27   | QUINTO |
| ARIADNA    | 27   | QUINTO |
| LUISA      | 27   | QUINTO |
| EFRAIN     | 27   | QUINTO |
| FRANCISCO  | 27   | QUINTO |
| CANDELARIA | 27   | QUINTO |
+------------+------+--------+
7 rows in set (0.001 sec)
```

Figura 2.17: Ejemplo de consulta de datos en general

Como se menciono anteriormente para buscar el registro de un estudiante en especifico se deberá colocar en consola la sentencia: **SELECT * FROM alumno WHERE nombre=”ARIADNA”** el cual se muestra en la figura 2.18.

```
[tesis]> SELECT * FROM alumno WHERE nombre="ARIADNA";
+----------+------+--------+
| nombre   | edad | grado  |
+----------+------+--------+
| ARIADNA  | 27   | QUINTO |
```

Figura 2.18: Ejemplo de consulta especifica

2.2.17 Modificación de registros de una Tabla

La modificación de la información consiste en actualizar los valores de los atributos para una o varias tuplas de una tabla [35]. En caso de que se necesite actualizar algún registro de nuestra tabla se necesita utilizar la sentencia UPDATE y opcionalmente puede incluirse la sentencia WHERE para especificar el registro a modificar.

En la figura 2.19 se ejemplifica el cambio de edad de GUILLERMO de la tabla alumno el cual es de 27 y se requiere cambiar a 25, mediante la sentencia **UPDATE alumno set edad=”25” where nombre =”GUILLERMO”**, para verificar que se

hizo de la manera correcta la consulta con la sentencia **SELECT * FROM alumno**, en donde podemos ver que la edad del estudiante GUILLERMO se cambio a 25.

```
       [(none)]> use tesis;
Database changed
       [tesis]> SELECT *FROM alumno;
+------------+------+--------+
| nombre     | edad | grado  |
+------------+------+--------+
| GUILLERMO  | 27   | QUINTO |
| DIANA      | 27   | QUINTO |
| ARIADNA    | 27   | QUINTO |
| LUISA      | 27   | QUINTO |
| EFRAIN     | 27   | QUINTO |
| FRANCISCO  | 27   | QUINTO |
| CANDELARIA | 27   | QUINTO |
+------------+------+--------+
7 rows in set (0.010 sec)

       [tesis]> UPDATE alumno set edad='25' where nombre ='GUILLERMO';
Query OK, 1 row affected (0.016 sec)
Rows matched: 1  Changed: 1  Warnings: 0

       [tesis]> SELECT *FROM alumno;
+------------+------+--------+
| nombre     | edad | grado  |
+------------+------+--------+
| GUILLERMO  | 25   | QUINTO |
| DIANA      | 27   | QUINTO |
| ARIADNA    | 27   | QUINTO |
| LUISA      | 27   | QUINTO |
| EFRAIN     | 27   | QUINTO |
| FRANCISCO  | 27   | QUINTO |
| CANDELARIA | 27   | QUINTO |
+------------+------+--------+
```

Figura 2.19: Ejemplo de Modificación.

2.2.18 Eliminación de registros de una Tabla

La eliminación de información consiste en borrar los registros de una tabla de la base de datos [35]. En caso de eliminar un registro de una tabla se utiliza la sentencia DELETE, como su nombre lo indica su función es la de eliminar los datos de una tabla.

En la figura 2.20 se ejemplifica la eliminación del registro GUILLERMO mediante la sentencia **SELECT * FROM alumno**, se ven en consola los registros de la tabla alumno, en donde se ubican los atributos: nombre, edad y grado de cada uno de los estudiantes, para eliminar el registro del estudiante GUILLERMO se debe escribir la

siguiente sentencia: **DELETE FROM alumno WHERE nombre = "GUILLERMO"**, para verificar que se hizo de la manera correcta se utiliza la sentencia SELECT * FROM alumno, en donde podemos ver que la información del estudiante GUILLERMO se elimino.

```
       [tesis]> SELECT *FROM alumno;
+------------+------+--------+
| nombre     | edad | grado  |
+------------+------+--------+
| GUILLERMO  | 25   | QUINTO |
| DIANA      | 27   | QUINTO |
| ARIADNA    | 27   | QUINTO |
| LUISA      | 27   | QUINTO |
| EFRAIN     | 27   | QUINTO |
| FRANCISCO  | 27   | QUINTO |
| CANDELARIA | 27   | QUINTO |
+------------+------+--------+
7 rows in set (0.010 sec)

       [tesis]> DELETE FROM  alumno where nombre ='GUILLERMO';
Query OK, 1 row affected (0.012 sec)

       [tesis]> SELECT *FROM alumno;
+------------+------+--------+
| nombre     | edad | grado  |
+------------+------+--------+
| DIANA      | 27   | QUINTO |
| ARIADNA    | 27   | QUINTO |
| LUISA      | 27   | QUINTO |
| EFRAIN     | 27   | QUINTO |
| FRANCISCO  | 27   | QUINTO |
| CANDELARIA | 27   | QUINTO |
+------------+------+--------+
6 rows in set (0.001 sec)
```

Figura 2.20: Ejemplo de Eliminación.

2.2.19 Operador unión (JOIN)

Un componente importante de cualquier base de datos relacional es la correlación que puede existir entre dos tablas cualesquiera. Esta relación le permite al usuario publicar datos en una tabla con datos en otra tabla [27].

El uso de JOIN nos permite combinar dos o más tablas siempre y cuando tengan un campo que sea común entre ellas, en este trabajo se muestran los operadores de unión más utilizados (INNER JOIN, LEFT JOIN y RIGHT JOIN) [14].

2.2.20 INNER JOIN

El objetivo de esta sentencia es indicar si existen coincidencias entre las tablas de la base de datos seleccionada, esto con la finalidad de comprobar si existen registros en común, en el caso de no existir un registro que coincida con otra tabla; se excluirá y sólo aparecerán los registros que tengan relación con otra tabla [3]. En la figura 2.21 se muestra el Diagrama de Venn representando el INNER JOIN entre los conjuntos A y B.

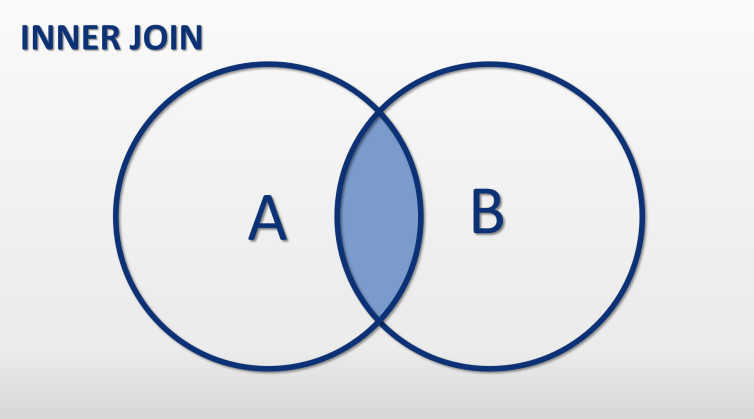


Figura 2.21: Diagrama de Venn INNER JOIN.

Al aplicar INNER JOIN a las tablas cliente y pedidos como se muestra en la figura 2.22, podemos acceder a la información de ambas tablas y como se ha hecho mención anteriormente regresa todos los registro que se encuentren presentes en ambas tablas. Para este ejemplo, se obtienen los valores de las columnas ClienteID de la tabla clientes y ClienteID de la tabla pedidos, desde la intersección de la unión de la tabla clientes y pedidos se retorna los registros cuando el valor de la tabla clientes en la columna ClienteID sea igual al valor de la tabla pedidos en la columna ClienteID y como resultado final, todos los registros encontrados los ordena de acuerdo al valor de la tabla Salida. Nótese que en este resultado no se encuentran los usuarios Guillermo, Diana y Marco, esto es debido a que no existe una relación de dichos registros.

```
[tesis]> select * from clientes;
+-----------+---------------+----------+
| ClienteID | NombreCliente | Contacto |
+-----------+---------------+----------+
| 1         | Guilllermo    | 123      |
| 2         | Diana         | 456      |
| 3         | Ariadna       | 789      |
| 4         | Luisa         | 122      |
| 5         | Marco         | 112      |
+-----------+---------------+----------+
5 rows in set (0.001 sec)
```

(a) Tabla clientes

```
[tesis]> select * from pedidos;
+----------+-----------+---------+
| PedidoID | ClienteID | Factura |
+----------+-----------+---------+
| 234      | 4         | 90      |
| 235      | 3         | 91      |
| 236      | 3         | 92      |
| 237      | 4         | 93      |
+----------+-----------+---------+
4 rows in set (0.001 sec)
```

(b) Tabla pedidos

```
[tesis]> SELECT Clientes.NombreCliente, Pedidos.PedidoID FROM Clientes
    -> INNER JOIN Pedidos ON Clientes.ClienteID=Pedidos.ClienteID
    -> ORDER BY Clientes.NombreCliente;
+---------------+----------+
| NombreCliente | PedidoID |
+---------------+----------+
| Ariadna       | 235      |
| Ariadna       | 236      |
| Luisa         | 237      |
| Luisa         | 234      |
+---------------+----------+
4 rows in set (0.001 sec)
```

(c) Tabla Salida

Figura 2.22: Ejemplo de INNER JOIN

2.2.21 LEFT JOIN

LEFTH JOIN devuelve todos los registros de la tabla izquierda y los registros coincidentes de la tabla derecha, en el caso de que no existan resultados se coloca NULL de lado derecho de la tabla. En la figura 2.23 se muestra un ejemplo de LEFT JOIN en SQL.

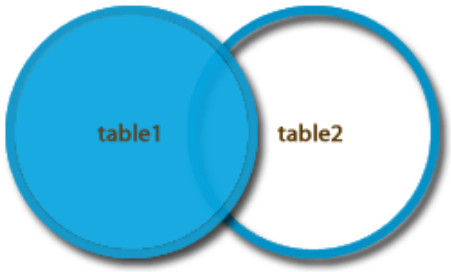


Figura 2.23: Diagrama de Venn LEFT JOIN.

```
[tesis]> select * from clientes;
+-----------+---------------+----------+
| ClienteID | NombreCliente | Contacto |
+-----------+---------------+----------+
| 1         | Guilllermo    | 123      |
| 2         | Diana         | 456      |
| 3         | Ariadna       | 789      |
| 4         | Luisa         | 122      |
| 5         | Marco         | 112      |
+-----------+---------------+----------+
5 rows in set (0.001 sec)
```

(a) Tabla clientes

```
[tesis]> select * from pedidos;
+----------+-----------+---------+
| PedidoID | ClienteID | Factura |
+----------+-----------+---------+
| 234      | 4         | 90      |
| 235      | 3         | 91      |
| 236      | 3         | 92      |
| 237      | 4         | 93      |
+----------+-----------+---------+
4 rows in set (0.001 sec)
```

(b) Tabla pedidos

```
[tesis]> SELECT Clientes.NombreCliente, Pedidos.PedidoID
    -> FROM Clientes LEFT JOIN Pedidos
    -> ON Clientes.ClienteID=Pedidos.ClienteID
    -> ORDER BY Clientes.NombreCliente;
+---------------+----------+
| NombreCliente | PedidoID |
+---------------+----------+
| Ariadna       | 236      |
| Ariadna       | 235      |
| Diana         | NULL     |
| Guilllermo    | NULL     |
| Luisa         | 237      |
| Luisa         | 234      |
| Marco         | NULL     |
+---------------+----------+
7 rows in set (0.018 sec)
```

(c) Tabla Salida

Figura 2.24: Ejemplo de LEFT JOIN

En la figura 2.24 se ejemplifica el uso de LEFT JOIN entre las tablas clientes y pedidos, al aplicar LEFT JOIN a las tablas clientes y pedidos se puede acceder a la información persistente en ellas. En la sentencia mostrada, se obtienen los valores de las columnas ClienteID de la tabla clientes y PedidoID de la tabla pedidos, como se puede apreciar en la tabla Salida, regresan todos los registros de la tabla cliente que se encuentran o no asociados con la tabla pedidos, a diferencia de INNER JOIN, aquí se muestran los usuarios Diana, Guillermo y Marco y en el campo PedidoID se les asigna el valor nulo, ya que no poseen ningun pedido.

2.2.22 RIGHT JOIN

RIGHT JOIN se mantienen todas las filas de la tabla derecha. Si las filas de la tabla izquierda solo se mostraran si hay coincidencia con las de la derecha,Si existen valores en la tabla derecha pero no en la tabla izquierda se mostraran como NULL. En la figura 2.25 se muestra el Diagrama de Venn representando el RIGHT JOIN.

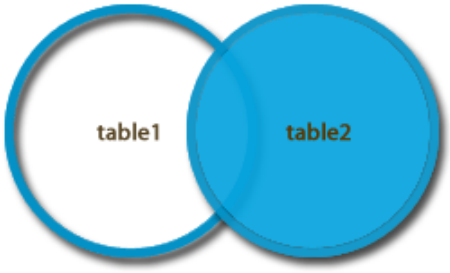


Figura 2.25: Diagrama de Venn RIGHT JOIN.

En la Figura 2.26 se muestra el uso de RIGHT JOIN, al aplicar RIGTH JOIN a las tablas cliente y pedidos, en la sentencia mostrada, se obtienen los valores de la columna PedidosID de la tabla pedidos, así como de los valores en la columna ClienteID de la tabla clientes, con la intersección de ambas tablas. Como se puede apreciar, regresa todos los registros de la tabla pedidos que se encuentran o no asociados con la tabla clientes.

```
[tesis]> select * from clientes;
+-----------+---------------+----------+
| ClienteID | NombreCliente | Contacto |
+-----------+---------------+----------+
| 1         | Guilllermo    | 123      |
| 2         | Diana         | 456      |
| 3         | Ariadna       | 789      |
| 4         | Luisa         | 122      |
| 5         | Marco         | 112      |
+-----------+---------------+----------+
5 rows in set (0.001 sec)
```

(a) Tabla clientes

```
[tesis]> select * from pedidos;
+----------+-----------+---------+
| PedidoID | ClienteID | Factura |
+----------+-----------+---------+
| 234      | 4         | 90      |
| 235      | 3         | 91      |
| 236      | 3         | 92      |
| 237      | 4         | 93      |
+----------+-----------+---------+
4 rows in set (0.001 sec)
```

(b) Tabla pedidos

```
MariaDB [tesis]> SELECT Pedidos.PedidoID, Clientes.NombreCliente
    -> FROM Clientes RIGHT JOIN Pedidos
    -> ON Clientes.ClienteID=Pedidos.ClienteID
    -> ORDER BY Pedidos.PedidoID;
+----------+---------------+
| PedidoID | NombreCliente |
+----------+---------------+
| 234      | Luisa         |
| 235      | Ariadna       |
| 236      | Ariadna       |
| 237      | Luisa         |
+----------+---------------+
4 rows in set (0.002 sec)
```

(c) Tabla Salida

Figura 2.26: Ejemplo de RIGHT JOIN

2.2.23 Entornos de de Desarrollo Integrado(IDE)

Los IDEs tal y como su nombre lo indica, son entornos de desarrollo integrados. En un mismo programa es posible escribir el código Java, compilarlo y ejecutarlo sin tener que cambiar de aplicación [29].

Los IDE ayudan al programador facilitando los procesos de un proyecto, ya que sobre el mismo IDE se lleva a cabo todo el proyecto. Los IDE apoyan a la creación, edición, compilación, implementación y depuración del programa.

Actualmente existen diferentes IDEs de uso libre a continuación se describen algunos:

Eclipse: Es uno entorno muy utilizado ya que soporta un gran número de lenguajes como: Java, C++, PHP, Perl ,etc. Permite realizar aplicaciones de escritorio y aplicaciones web por lo que nos brinda una gran versatilidad. En la figura 2.27 se muestra se muestra la interfaz gráfica del entorno de desarrollo del IDE de Eclipse.

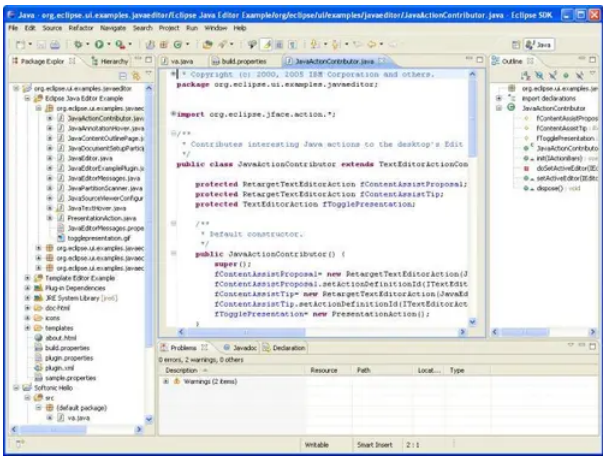


Figura 2.27: IDE Eclipse

Netbeans: Es un entorno multilenguaje en el cual se desarrolla software de alta calidad. Permite crear aplicaciones web y de escritorio además de contar con una cantidad muy grande de plugins. Netbeans fue creado por Oracle y su creación fue para ser el IDE de Java. En la figura 2.28 se muestra la interfaz gráfica del entorno de desarrollo de NetBeans.

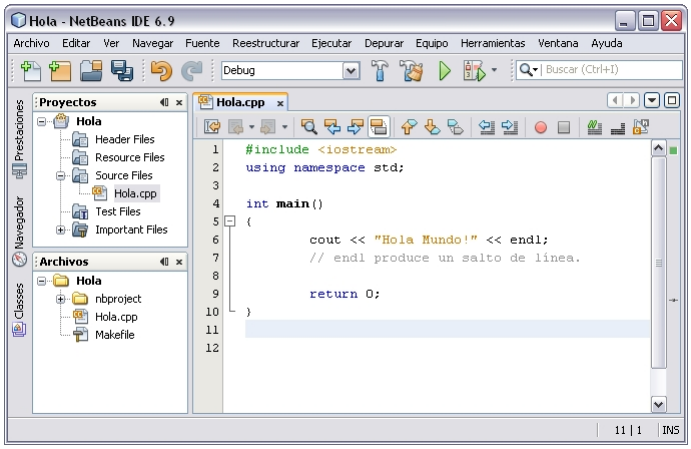


Figura 2.28: IDE Netbeans

Visual Studio: Fue diseñado por Microsoft para manejar lenguajes como C# y C++ el inconveniente es que no es multilenguaje. En la figura 2.29 se muestra la interfaz gráfica del entorno de desarrollo de Visual Studio.

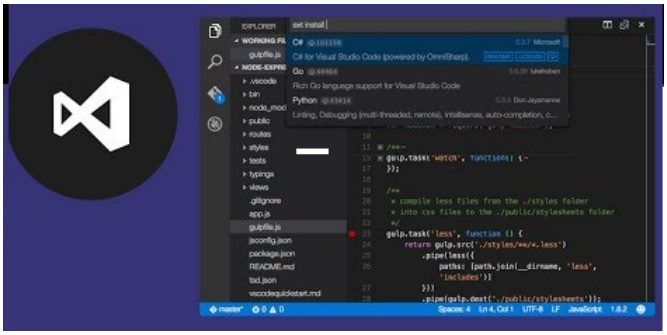


Figura 2.29: IDE Visual Studio

CodeLite: Es un IDE de código abierto y libre bajo la licencia GNU (General Public License), se instala en diversos sistemas operativos y soporta lenguajes como: C/C++, PHP y Node.js. En la figura 2.30 se muestra la interfaz gráfica del entorno de desarrollo de CodeLite.

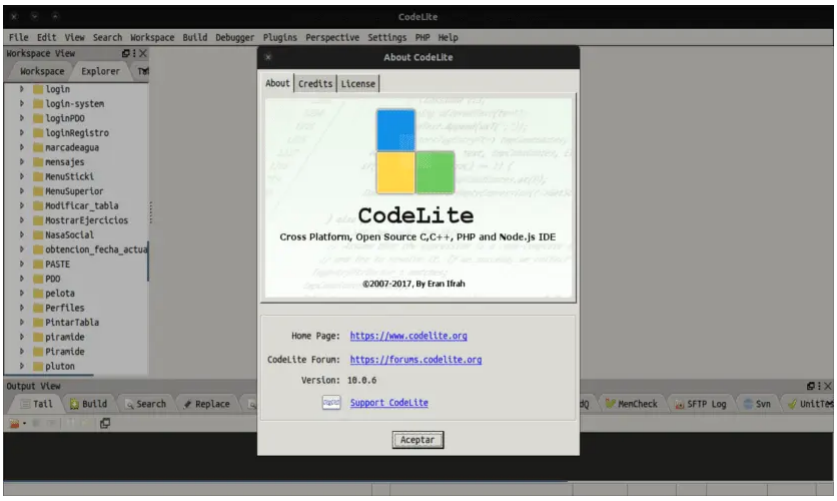


Figura 2.30: IDE CodeLite

Para el desarrollo del proyecto, se eligió NetBeans debido a que es un IDE de desarrollo Java, cuenta con un diseño personalizable el cual permite modificar el tipo de texto e interfaz, permite crear vistas gráficas con una gran cantidad de componentes,

NetBeans cuenta con diferentes plugins que se pueden descargar dependiendo de las herramientas que necesitemos para nuestro proyecto, para el proyecto se utilizó el plugin iReport, ya que permite generar la estructura de los reportes así como las consultas a base de datos para obtener información.

2.3 Back-End y Front-End

2.3.1 Back-End

Se le denomina Back-End a toda la parte lógica de la aplicación, esto hace referencia a la arquitectura interna del programa. El Back-End es la capa encarga de interactuar con bases de datos, gestionar el acceso de sesión de usuarios y de integrar al usuario con la información de la aplicación [31].

2.3.2 Front-End

Es la forma en la que se representa la aplicación al usuario generalmente de manera gráfica, la cual incluye los estilos, los colores, los fondos y tamaños de cada objeto de nuestra interfaz. Normalmente el Front-End se encarga de estilizar la presentación de la aplicación al usuario de tal manera que sea mas fácil de utilizar [31].

2.4 Metodologías de desarrollo de software

Las metodologías para el desarrollo de software nos indican la forma de organizar, administrar y desarrollar proyectos de forma efectiva antes de implementar el software, de esta manera se asegura su calidad y tiempo de desarrollo. A continuación, se describen dos de las metodologías más utilizadas.

2.4.1 Modelo Cascada

Es un modelo secuencial, el cual ya debe de contar con los requerimientos definidos y visibles; se concibe como un conjunto de etapas que se ejecutan una tras otra y los cambios durante el proyecto son mínimos. En la Figura 2.31 se muestra el diagrama del modelo, cuyas etapas se describen a continuación. [30]

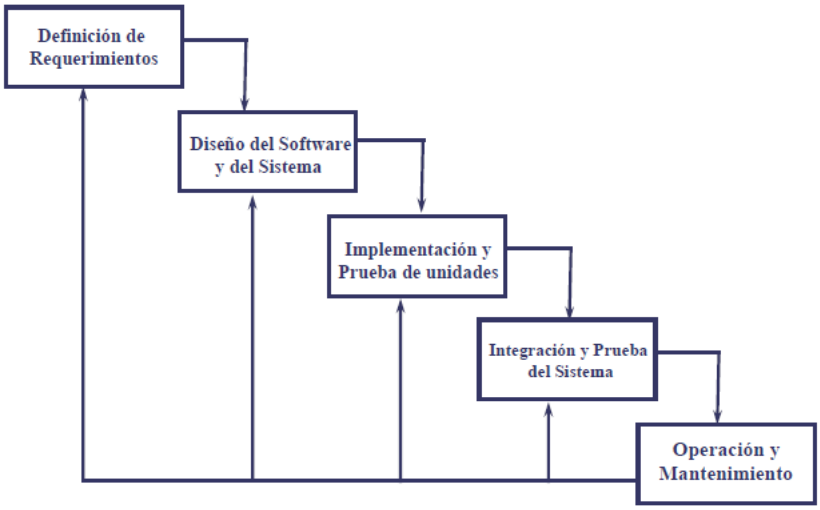


Figura 2.31: Modelo Cascada
[30]

- **Definición de Requerimientos**: Realiza un análisis de las necesidades del cliente para identificar las características del software que se desea desarrollar, y se específica todo lo que se debe hacer en el sistema sin entrar en detalles técnicos. En esta etapa se realiza una lista detallada de los requisitos entre el cliente, la empresa y lo que el producto debe de hacer. [2]

- **Diseño de Software y del Sistema**: En esta etapa se describe la estructura interna y las relaciones entre las entidades del software con base en los requerimientos planteados en la fase anterior. [11]

- **Implementación y Prueba de Unidades**: Se lleva a cabo la etapa de programación del software en algún lenguaje de programación, donde se realizan

los requisitos acordados haciendo uso de las estructuras de datos, cabe mencionar que aun faltan otros procesos para completar la solución. [11]

- **Integración y Prueba del Sistema**: Una vez terminada la fase de implementación, y de verificar que todos los componentes del sistema funcionen de manera correcta, se realizan las pruebas necesarias para validar que el sistema se encuentre estable para entregarlo al cliente. [11]

- **Operación y Mantenimiento**: Aquí se le brinda al cliente asesoría y soporte para atender errores, o para realizar las modificaciones necesarias al sistema desarrollado. Para llevar correctamente la fase de mantenimiento, se debe llevar acabo un plan que nos prepare para todas los tipos de escenarios que puedan producirse. [11]

Es importante mencionar que aunque esta metodología de desarrollo fue de aplicación extendida hace algunas décadas, actualmente no es la más popular. A continuación se presenta Scrum, quizás la manera más común de desarrollar proyectos de cualquier tipo actualmente.

2.4.2 Metodología Scrum

En la década de los 80s, Hirotaka Takeuchi e Ikujiro Nonaka, desarrollaron una estrategia en la cual todo el equipo de desarrollo trabajara de manera más eficiente para la elaboración de un producto. Ellos se percataron que el desarrollo de un producto no debe de ser realizado de una manera secuencial y en un solo sentido, sino que debe tener la capacidad de moverse hacia adelante o hacia atrás en cada una de las etapas. Es decir, aceptaron lo que sucede comúnmente en el mundo real: los requisitos o necesidades de los clientes pueden variar de un momento a otro (el cambio es inevitable), y el equipo debe ajustarse rápidamente a esos cambios. [12]

En 1995 Ken Schwaber y Jeff Sutherland crearon el concepto de Scrum, como se aplica en el desarrollo de software. En los últimos años, Scrum es el modelo de desarrollo

de software más utilizado en organizaciones a nivel mundial. En la Figura 2.32 se muestra un esquema del modelo. [12]

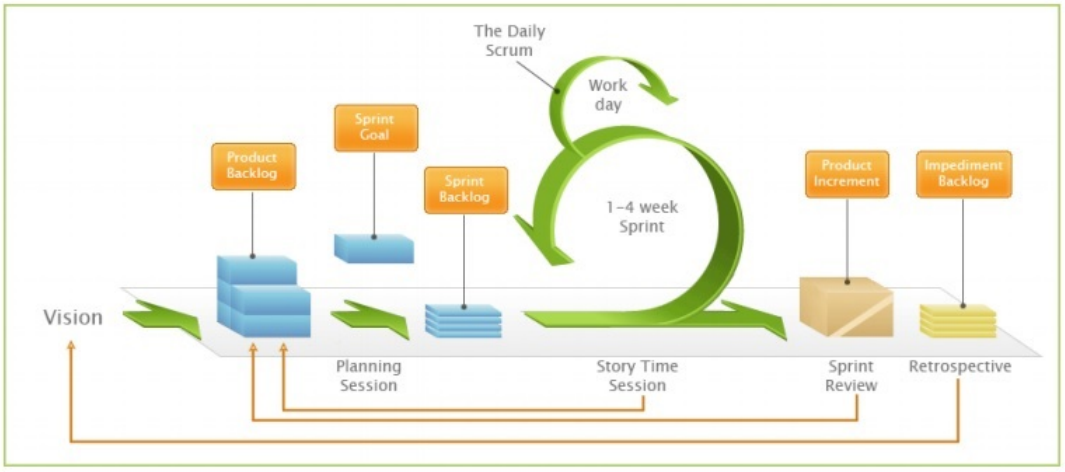


Figura 2.32: Modelo Scrum [39]

En este modelo existen tres tipos de roles, los cuales se describen a continuación:

Roles en Scrum

- **Propietario del producto**: Toma cada una de las decisiones del proyecto, ya que conoce cada uno de los requerimientos del cliente, por lo que es responsable de comunicar e informar el progreso y funcionalidad del mismo. [26]
- **Scrum Master**: Su función principal es entrenar, motivar y encargarse de la interacción que tiene el equipo de trabajo. Además, ayuda a resolver problemas que impidan al equipo continuar con sus actividades y avanzar. [26]
- **Equipo de desarrollo**: Son los responsables del desarrollo del producto, servicio o de cualquier otro resultado. Se conforma por un equipo de tres a nueve personas las cuales se encargan de estructurar y empoderar a los equipos de desarrollo para que estos organicen y gestionen su propio trabajo. [26]

Algunos de los términos de Scrum que conviene conocer son presentados en los siguiente párrafos.

Sprint: Scrum estructura los bloques de tiempo con una duración de entre 1 y 4 semanas. Durante este periodo, se desarrolla un incremento de producto utilizable y potencialmente desplegable. Entiéndase por desplegable, como algo utilizable, que el propietario de producto o alguien más puede probar. Un Sprint sólo puede ser pausado por el propietario del producto, generalmente ocurre cuando el cliente requiere modificar el proyecto. [39]

Artefactos de Scrum Los artefactos Scrum están diseñados específicamente para maximizar la transparencia de la información, los cuales se mencionan a continuación:

- **Product backlog:** Se refiere a una lista que enumera todas las características, funcionalidades, mejoras y requisitos que constituyen cambios a realizarse sobre el producto. un Product backlog es dinámico, ya que cambia constantemente para identificar lo que el producto necesita. [39]

- **User Stories:** User Stories son descripciones cortas y simples de una característica contada desde la perspectiva de un usuario o cliente. Es un requisito de entrega definido, el cual posee un valor agregado para el cliente. A continuación se muestra una plantilla para generar User Stories:

  - Como < Usuario>
  - Quiero <algún objetivo>
  - Para que <motivo>

- **Sprint backlog:** Es un conjunto de elementos de la lista de Product backlog que se puede desglosar en tareas más pequeñas. El equipo decide el número de historias que puede realizar durante el Sprint, en otras palabras, el Sprint backlog es una parte del product backlog desglosada en pequeñas partes. [39]

Ceremonias de Scrum

Las ceremonias Scrum son prácticas que definen la metodología Scrum, son necesarias para facilitar los métodos de comunicación, algo que beneficia al proyecto. A continuación se define cada una de ellas:

- **Sprint planning:** Permite que todo el equipo de desarrollo este preparado y que en cada Sprint se realicen las actividades correctamente. Cada Sprint planning se realiza al comienzo de cada Sprint y por lo general dura una hora por cada semana de Sprint. [39]

- **Daily Scrum:** Es una reunión que se realiza a diario con una duración mínima de 15 minutos, en la cual cada uno de los desarrolladores del equipo Scrum debe informar de manera individual sobre el estatus actual del proyecto.

- **Sprint review:** Es una sesión que se realiza al final de cada Sprint, cuya finalidad es mostrar los resultados del trabajo durante el Sprint, la cual tiene una duración máxima de cuatro horas. [39]

- **Retrospective:** Se realiza al terminar un Sprint review. El Retrospective tiene como duración tres horas por un Sprint de un mes. La sesión consiste en obtener una retroalimentación rápida con el propósito de mejorar el desarrollo. [39]

En esta tesis se empleó la metodología de Scrum para el desarrollo del proyecto. Esto debido a la característica de tener respuestas en corto tiempo, y una excelente manera de planear cada una de las etapas. Cabe de mencionar que debido a que el autor de esta tesis estuvo en todos los roles Scrum, es decir, identificó el problema, pensó en una solución, la implementó y la probó, no se trabajo al pie de la letra con la metodología Scrum. A pesar de ello, aplicar dicha metodología facilitó el avance y culminación del proyecto.

Capítulo 3

ANÁLISIS, DISEÑO Y ARQUITECTURA DEL SISTEMA

Introducción

En este capítulo se describe el análisis de los requerimientos del sistema, así como el diseño y la arquitectura del Sistema Web para el Registro de Solicitudes de Apoyos Municipales Zumpango, Estado de México. Se implementó un sistema que permite gestionar las solicitudes que se realizan diariamente, así como la generación de reportes en el Área de bienestar social y una pagina Web que permite monitorear el registro de solicitudes por día.

El sistema se divide en dos capas; la capa back-end, es la encargada de interactuar con las peticiones que se realizan a la base de datos desde la interfaz gráfica de la aplicación. La capa front-end presenta al usuario la manera gráfica de interactuar con la aplicación.

3.1 Análisis

3.1.1 Requerimientos del Sistema

Se utilizo la técnica de observación para el levantamiento de requerimientos del sistema, es necesario, importante y relevante observar y conocer el modo de operación del registro de solicitudes, esto permitirá definir de manera más explicita los requerimientos que se necesitan. Para el sistema desarrollado se identificaron los siguientes requerimientos:

1. Diseñar y desarrollar una interfaz multiplataforma, debido a que hay diversos usuarios con diferentes equipos de computo y sistemas operativos (Windows o Mac). La interfaz debe de ser implementada como una aplicación de escritorio.

2. Monitorear el servicio de registro de solicitudes que se hacen por día mediante una pagina Web.

3. Acceder a la página Web y autenticar a los usuarios mediante nombre de usuario y una contraseña.

4. Restringir la edición y actualización de las solicitudes mediante la pagina Web.

5. Gestionar los apoyos municipales, mediante la edición, actualización o alta de apoyos que brinda el municipio de Zumpango.

6. Capturar las nuevas solicitudes al sistema. Los usuarios deben ingresar de manera manual los datos de las solicitudes.

7. Modificar las solicitudes capturadas en el sistema. Para corregir o actualizar errores de captura en los datos del solicitante.

8. Eliminar las solicitudes capturadas en el sistema y registradas en la Base de Datos.

9. Generar el reporte individual de cada una de las solicitudes. La generación de reportes individuales es fundamental, ya que al finalizar el mes estos se entregan al área de Controlaría o Tesorería de manera física.

10. Generar el reporte general, donde se muestre el consolidado de todas las solicitudes registradas mediante una lista con el número de registro, fecha de alta y la información del solicitante.

11. Gestionar el acceso a usuarios. El usuario con el rol de ADMINISTRADOR dará de alta a nuevos usuarios otorgándoles el rol de USUARIO o ADMINISTRADOR, con el fin de asignar a las personas encargadas de registrar las solicitudes dentro del área de Bienestar Social.

12. Modificar la información de los usuarios registrados en el sistema. El usuario con el rol ADMINISTRADOR podrá realizar las modificaciones a los usuarios, incluyendo su rol.

Para satisfacer los requerimientos que se mencionaron anteriormente, se diseñó el sistema bajo un enfoque orientado a objetos. A continuación se presenta el back-end y posteriormente el front-end.

3.1.2 Resumen de la aplicación de la metodología de desarrollo para el proyecto

A continuación, se resumen cada uno de los Sprint que se utilizaron en las distintas etapas de desarrollo del sistema.

Sprint 1 Se asignó un tiempo de 4 semanas para este, en el cual se definió el modelado de datos y el diseño de las interfaces de usuario. En Tabla 3.1 se describen cada uno de las etapas.

Tabla 3.1: SPRINT 1

Tarea	**En Proceso**	**Con errores a resolver**	**En pruebas**	**Finalizadas**
Diagrama Entidad-Relación	01/03/2020		06/03/2020	07/03/2020
Diagrama de clases Java	01/03/2020		06/03/2020	07/03/2020
Bocetos de Interfaz Java	01/03/2020		06/03/2020	07/03/2020
Interfaz de Validación de usuario	08/03/2020		12/03/2020	13/03/2020
Interfaz del panel de control	08/03/2020		12/03/2020	13/03/2020
Interfaz del Registro del Usuario	08/03/2020		12/03/2020	13/03/2020
Interfaz para modificar al usuario	08/03/2020		12/03/2020	13/03/2020
Interfaz para registrar una solicitud	15/03/2020		19/03/2020	20/03/2020
Interfaz para modificar una solicitud	15/03/2020		19/03/2020	20/03/2020
Interfaz para generar reportes	15/03/2020		19/03/2020	20/03/2020
Interfaz para realizar ajustes	15/03/2020		19/03/2020	20/03/2020
Interfaz de validación de usuario Web	22/03/2020		26/03/2020	27/03/2020
Interfaz lista de Registros Web	22/03/2020		26/03/2020	27/03/2020

Sprint 2

Se asignó un tiempo de 3 semanas en el Sprint 2 el cual implica el uso de bases de datos como gestor de información, en este caso se decidió por MySQL por ser de fácil uso e instalación, además de estar familiarizado con su uso, ya que durante la formación profesional se aplicó en varios proyectos. En la tabla 3.2 se muestra cada uno de los procesos de este Sprint.

Tabla 3.2: SPRINT 2

Tarea	**En Proceso**	**Con errores a resolver**	**En pruebas**	**Finalizadas**
Investigar acerca de MySQL	05/04/2020		09/04/2020	10/04/2020
Conectar a Mysql con Java	12/04/2020		16/04/2020	17/04/2020
Conectar a Mysql con Pagina Web	12/04/2020		16/04/2020	17/04/2020
Pruebas de conexión	19/04/2020		23/04/2020	24/04/2020

Sprint 3

Con un tiempo considerado de 3 semanas, en el Sprint 3 se realizan pruebas de comunicación entre la base de datos con la interfaz en Java, también se valida que PHP realice consultas correctas y no presente errores. En la tabla 3.3 se presenta cada uno de los procesos.

Tabla 3.3: SPRINT 3

Tarea	**En Proceso**	**Con errores a resolver**	**En pruebas**	**Finalizadas**
Validación de usuario	03/05/2020		07/05/2020	08/05/2020
Registro del Usuario	03/05/2020		07/05/2020	08/05/2020
Modificación de usuario	03/05/2020		07/05/2020	08/05/2020
Registrar una solicitud	10/05/2020		14/05/2020	15/05/2020
Modificar una solicitud	10/05/2020		14/05/2020	15/05/2020
Generar reportes	10/05/2020		14/05/2020	15/05/2020
Realizar ajustes	17/05/2020		21/05/2020	22/05/2020
Validación de usuario Web	17/05/2020		21/05/2020	22/05/2020
Lista de Registros Web	17/05/2020		21/05/2020	22/05/2020

3.2 Diseño

3.2.1 Back-end

Base de datos

Una de los elementos más importantes del sistema es la base de datos, debido a que permite almacenar la información de las solicitudes del sistema, para la implementación se eligió el Sistema Gestor de Base de Datos MySQL; esto por su popularidad, su disponibilidad en los sistemas operativos más comunes en computadoras de escritorio, además cuenta con una versión de licencia pública general, es decir, esta versión puede utilizarse sin pago de licencia en proyectos como el presente en esta tesis.

La Base de Datos del Sistema de Registro de Solicitudes se diseñó con base al Modelo Relacional, el cual permite crear relaciones entre cada una de las tablas mediante el uso de llaves foráneas, permitiendo que se ejecuten las diferentes operaciones (modificar o

eliminar) en los registros y asegurando su integridad, ya que al ejecutar una operación en un registro esta modificará todas las tablas con las que el registro tenga relación.

En la figura 3.1 se muestra el diagrama Entidad-Relación del Sistema de Registro de Solicitudes, integrado por nueve tablas. A continuación se describe cada una de ellas:

Tabla usuario: Contiene información referente a los usuarios, la cual permite que al ingresar al Sistema de Registro de Solicitudes el usuario pueda acceder siempre y cuando se encuentre registrado y habilitado dentro de la tabla. Cabe destacar que la tabla usuario esta relacionada con las tablas de roles y solicitudes.

Tabla roles: Contiene los diferentes roles (USUARIO y ADMINISTRADOR) que el usuario puede tener dentro del Sistema de Registro de Solicitudes, la cual se encuentra relacionada con la tabla Usuario.

Tabla solicitudes: Esta tabla contiene la información de las peticiones que un usuario ingresa en el Sistema de Registro de Solicitudes, la cual tiene relación con las tablas: vivienda, localidades, bienes_solicitante y apoyo, puesto que, comparten información de las solicitudes.

Tabla apoyo: Esta relacionada con la tabla solicitudes y su propósito es almacenar los apoyos que el usuario con el rol de ADMINISTRADOR dé de alta en el Sistema de Registro de Solicitudes, proporcionados por el Municipio de Zumpango.

Tabla localidades: Almacena los códigos postales y nombres de asentamientos de cada una de las localidades del Municipio de Zumpango. Cabe mencionar que esta tabla esta relacionada con la tabla solicitudes.

Tabla vivienda: Esta relacionada con la tabla solicitudes, contiene información del tipo de estructura del inmueble registrado en la solicitud, para tener referencia de su vivienda en caso de ser acreedor a un apoyo.

Tabla bienes_solicitante: Hace referencia al ID de la solicitud y el id de la tabla bienes, donde se registran todos los bienes que aparecen en la solicitud, esta tabla esta relacionada con la tabla solicitudes y la tabla bienes.

Tabla bienes: Esta tabla se relaciona con la tabla bienes_solicitante, debido a que contiene los registros de los bienes, tales como: DVD, televisión, cama, lavadora,

etcétera, para poder ser seleccionados en el sistema y así añadirlos a la solicitud.

Tabla materiales: Contiene registrado la lista de materiales con los que esta construida la infraestructura del inmueble.

Diagrama de clases

La figura 3.1 muestra el diagrama de la base de datos diseñada la cual se relaciona estrechamente con la estructura estática del sistema. Dicha estructura se muestra en el diagrama UML de las clases de la figura 3.2. A continuación se describen las principales clases que se diseñaron para el desarrollo del sistema.

La clase **GestorBD** se encarga de crear la conexión del sistema con la base de datos. El diagrama de clase UML que se muestra corresponde a la figura 3.3. El dato miembro **Connection** es el encargado de realizar la conexión a la base de datos. Los otros datos miembros son auxiliares para poder realizar las consultas SQL. La clase cuenta con métodos sobrecargados que permiten agregar diferentes elementos (usuarios, solicitantes, solicitudes y apoyos) a la base de datos, y también para actualizar los datos de estos elementos.

La clase **GestorBDSolicitante** se presenta en la figura 3.4, es una subclase de la clase GestorBD. Su propósito es gestionar de una manera más sencilla las operaciones que se realizan con los solicitantes de apoyos, implementando los métodos abstractos de su superclase. Además, agrega métodos sobrecargados para eliminar y buscar solicitudes o localidades. En esta clase se implementa la generación de reportes de solicitantes.

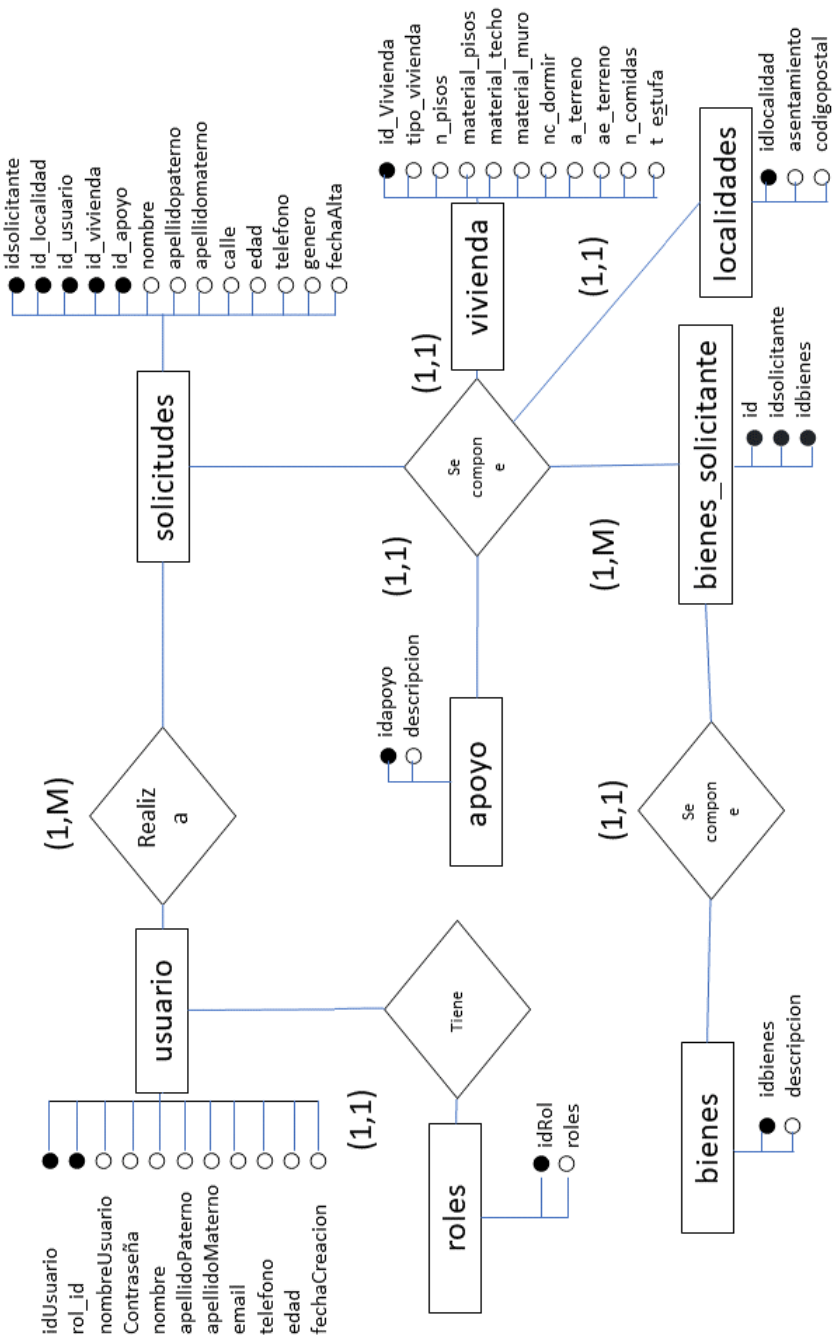


Figura 3.1: Diagrama Entidad-Relación de la base de datos

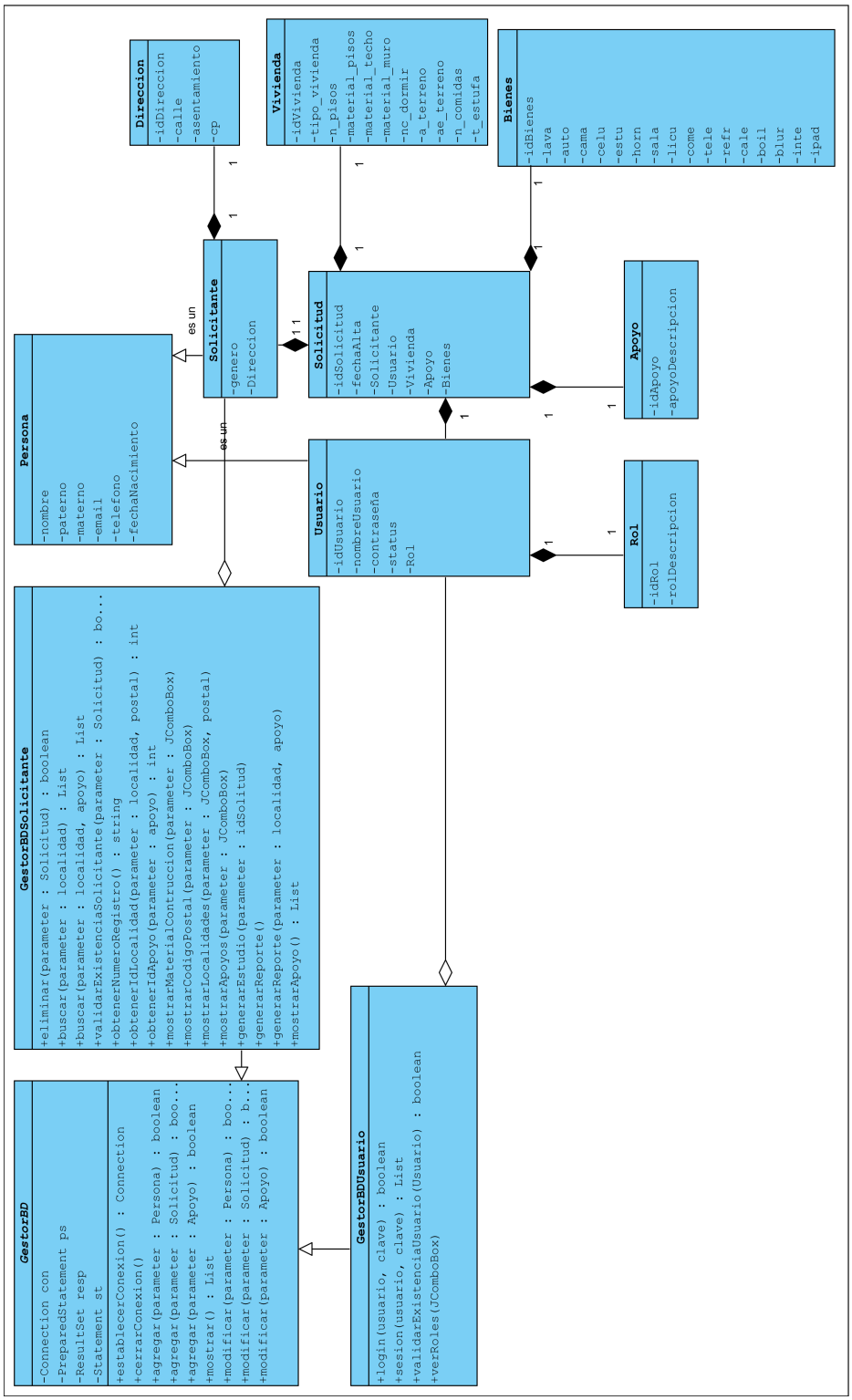


Figura 3.2: Estructura estática del sistema desarrollado

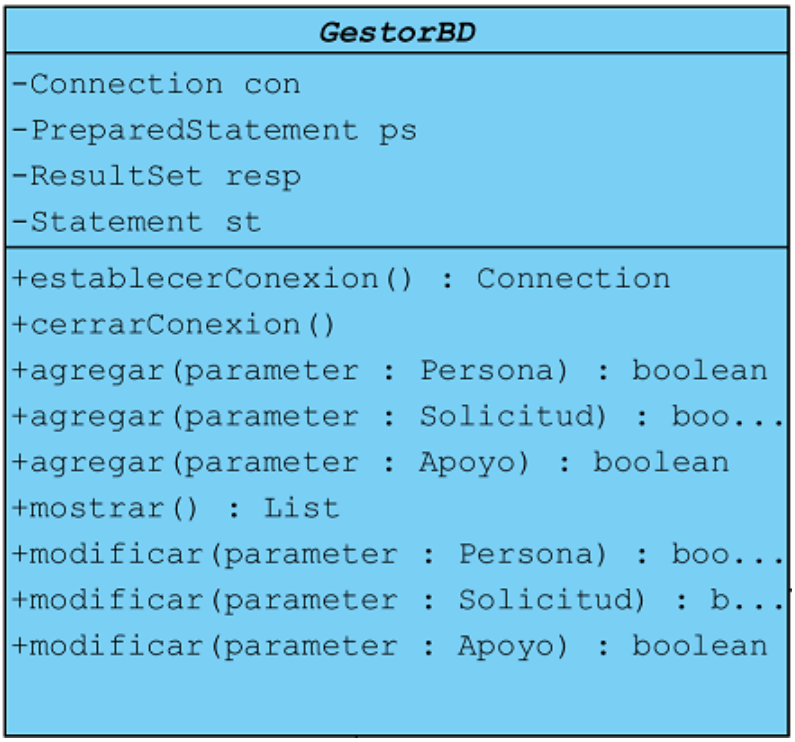


Figura 3.3: Clase GestorBD

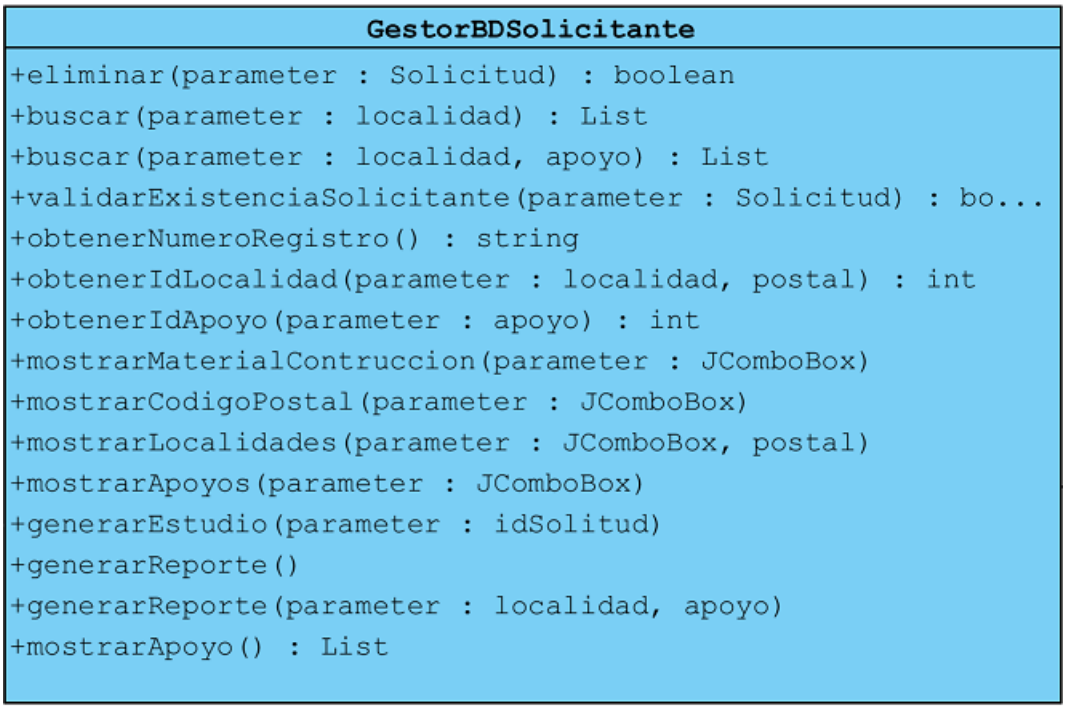


Figura 3.4: Clase GestorBDSolicitante

La clase **GestorBDUsuario** es una subclase de la clase de GestorBD, la cual se muestra en la figura 3.5, y es la encargada de gestionar las operaciones que tiene un usuario en el sistema, las cuales son: permitir el ingreso al sistema, validar la existencia de los usuarios, iniciar sesión de usuario y gestionar los roles.

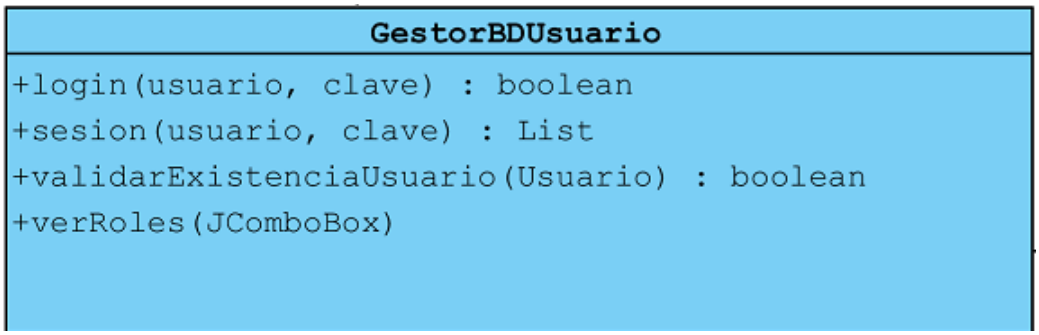


Figura 3.5: Clase GestorBDUsuario

En la figura 3.6 se presenta la clase **Persona** que se encarga de almacenar las características generales de una persona (nombre, apellido paterno, apellido materno, correo electrónico, número de teléfono y fecha de nacimiento) y hereda sus atributos a las subclases Usuario y Solicitante.

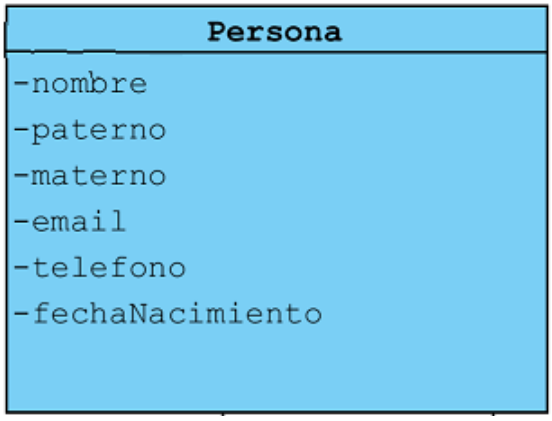


Figura 3.6: Clase Persona

La clase **Usuario** representada la figura 3.7, es una subclase de la clase Persona. Esta compuesta por la clase Rol y se encarga de contener la información que identifica al usuario dentro del sistema.

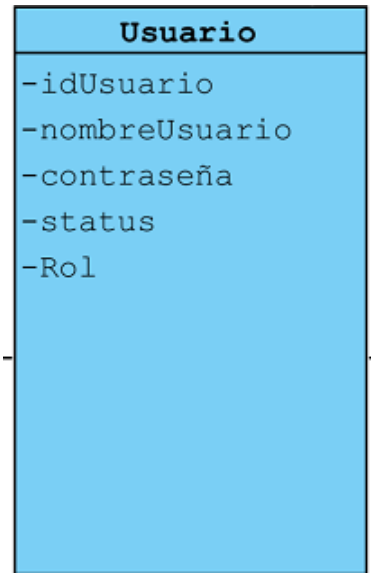


Figura 3.7: Clase Usuario

La figura 3.8 ejemplifica la clase **Rol**, esta se encarga de almacenar la información para definir un rol a cada usuario dentro del sistema, con el fin de brindar los permisos correspondientes al usuario.

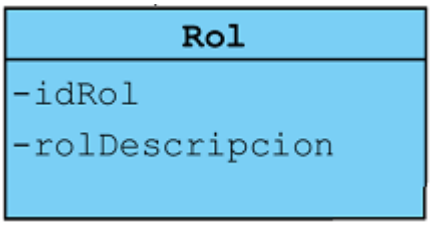


Figura 3.8: Clase Rol

La clase **Solicitante** esta compuesta por la clase Direccion y se encarga de guardar las características que identifican a un solicitante del Sistema de Registro de Solicitudes de Apoyos de Gobierno en Zumpango, Estado de México, es decir, la clase Solicitante se compone de dos elementos: Género y Dirección, las cuales guardan el género y la dirección del usuario, respectivamente. Cabe destacar que es una subclase de la clase Persona, la cual se ejemplifica en la figura 3.9.

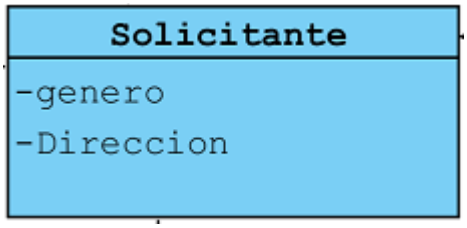


Figura 3.9: Clase Solicitante

La clase **Direccion** se encarga de contener las características que ayudan a definir la dirección de la solicitud, la cual contiene los siguientes datos: identificador, nombre de la calle, asentamiento y código postal, dichos datos pertenecen a la clase Direccion. A continuación en la figura 3.10 se muestra el diagrama de la clase Direccion.

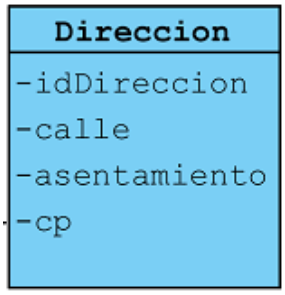


Figura 3.10: Clase Direccion

La clase **Solicitud** esta compuesta por cinco clases (Solicitante, Vivienda, Bienes, Apoyo y Usuario), la cual se representa en la figura 3.11, y su propósito es contener las características que conforman a la solicitud del sistema (identificador, fecha de alta, solicitante, usuario, vivienda, apoyo y bienes).

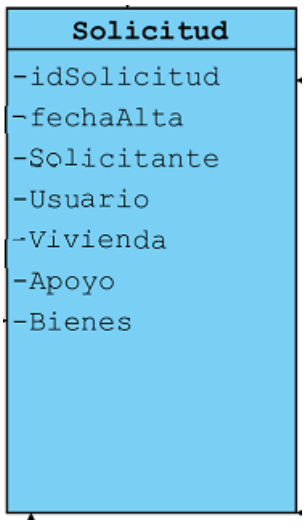


Figura 3.11: Clase Solicitud

La clase **Bienes** registra la propiedad del solicitante al momento de aplicar una Solicitud, en ella se encuentran los siguientes campos: identificador, lavadora, automóvil, cama, teléfono celular, estufa, horno de micro-hondas, sala, licuadora, comedor, televisión, refrigerador y calentador solar. Se representa en la figura 3.12

Figura 3.12: Clase Bienes

La clase **Vivienda** contiene los campos que definen el tipo de inmueble registrado

en la solicitud, en el cual se observa el identificador, tipo de vivienda, número de pisos, material del piso, material del techo, material del muro, número de cuartos, red de agua, número de comidas y tipo de estufa, la cual se representa en la figura 3.13.

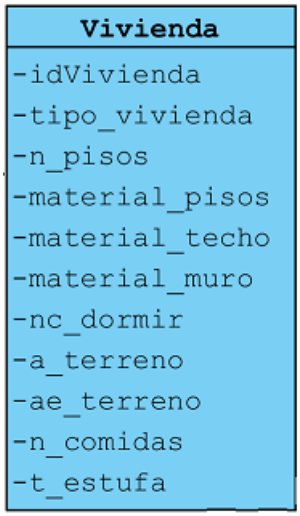


Figura 3.13: Clase Vivienda

La Figura 3.14 ejemplifica la clase **Apoyo** que se encarga de registrar los apoyos brindados por el Municipio de Zumpango.

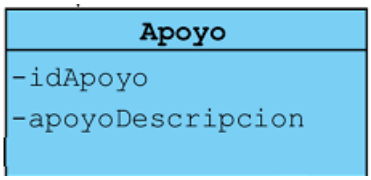


Figura 3.14: Clase Apoyo

3.2.2 Front-End

Para que el usuario manipule la información de una aplicación se crean diferentes interfaces gráficas, las cuales permiten al usuario interactuar de manera más intuitiva. Las interfaces gráficas que se utilizaron en este proyecto de investigación se desarrollaron en el IDE de Netbeans y son clases que se generan en la carpeta del proyecto por el IDE cada vez que se crea un JFrame o un JPanel.

3.3 Implementación

3.3.1 Interfaz en Java

Para que el usuario interactué con el sistema de manera simple y sencilla es necesario crear interfaces gráficas intuitivas. A continuación se describen cada una de las interfaces implementadas para el sistema.

En la figura 3.23 se presenta la Interfaz de Validación de Usuario, la cual se muestra con el logotipo del Ayuntamiento de Zumpango, los campos de usuario - contraseña y los botones de ingresar - salir. En esta interfaz el usuario tendrá que escribir tanto su nombre de usuario como su contraseña correspondiente. De esta manera es como se implementa la seguridad en el sistema.

Figura 3.15: Interfaz de Validación de usuario

Una vez que el usuario a ingresado se mostrará la Interfaz del Panel de control, el cual contiene en la pare superior izquierda el rol y nombre del usuario. Así como un

menú de opciones las cuales son: Gestión de Usuarios, Gestión de Solicitudes, Reportes y Ajustes. en la parte izquierda de la interfaz. En la figura 3.16 se muestra la interfaz del panel de control.

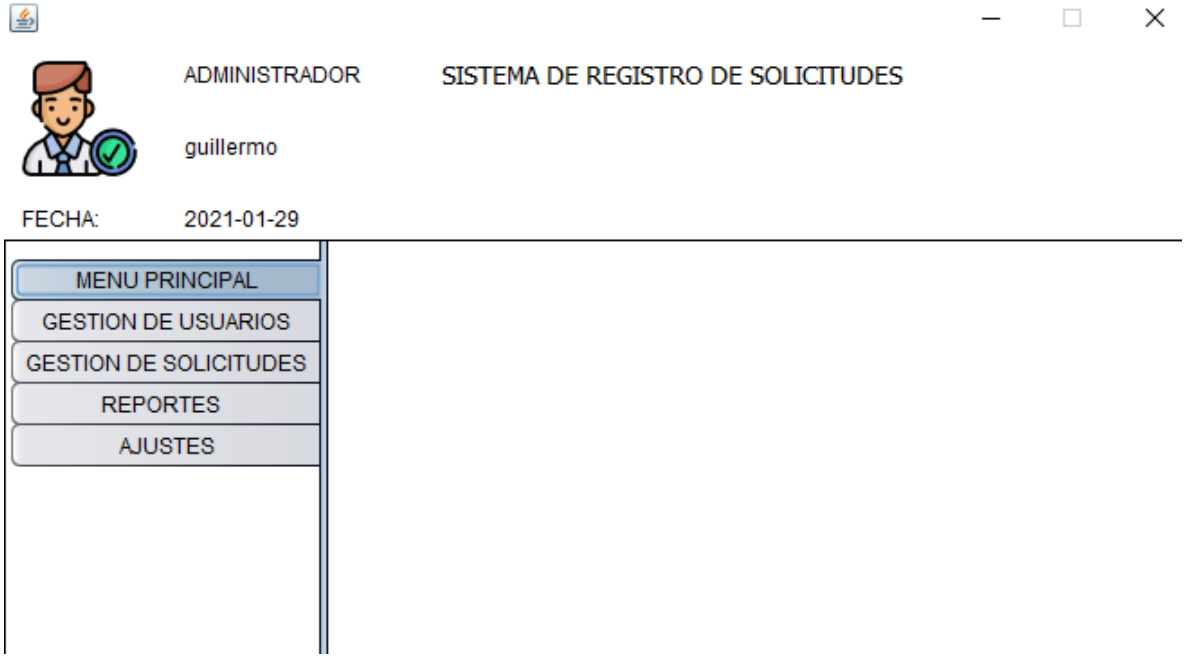


Figura 3.16: Interfaz del panel de control

En el panel de control, al dar clic sobre Gestión de Usuarios se muestran dos pestañas: Ver usuarios y Registrar un nuevo usuario.

La pestaña Registrar un nuevo usuario se muestra en la figura 3.17, se tiene un formulario con los datos que se solicitan a la persona que gestionará la aplicación estos son: Nombre, Apellido Paterno, Email, Fecha de Nacimiento, Teléfono, Rol, Nombre de usuario y clave, una vez registrados se le da clic al botón registrar usuario.

La opción Ver usuarios se muestra en la figura 3.18, coloca en una tabla los usuarios registrados en el sistema y permite modificar los datos de los usuarios seleccionando al usuario dentro de la tabla.

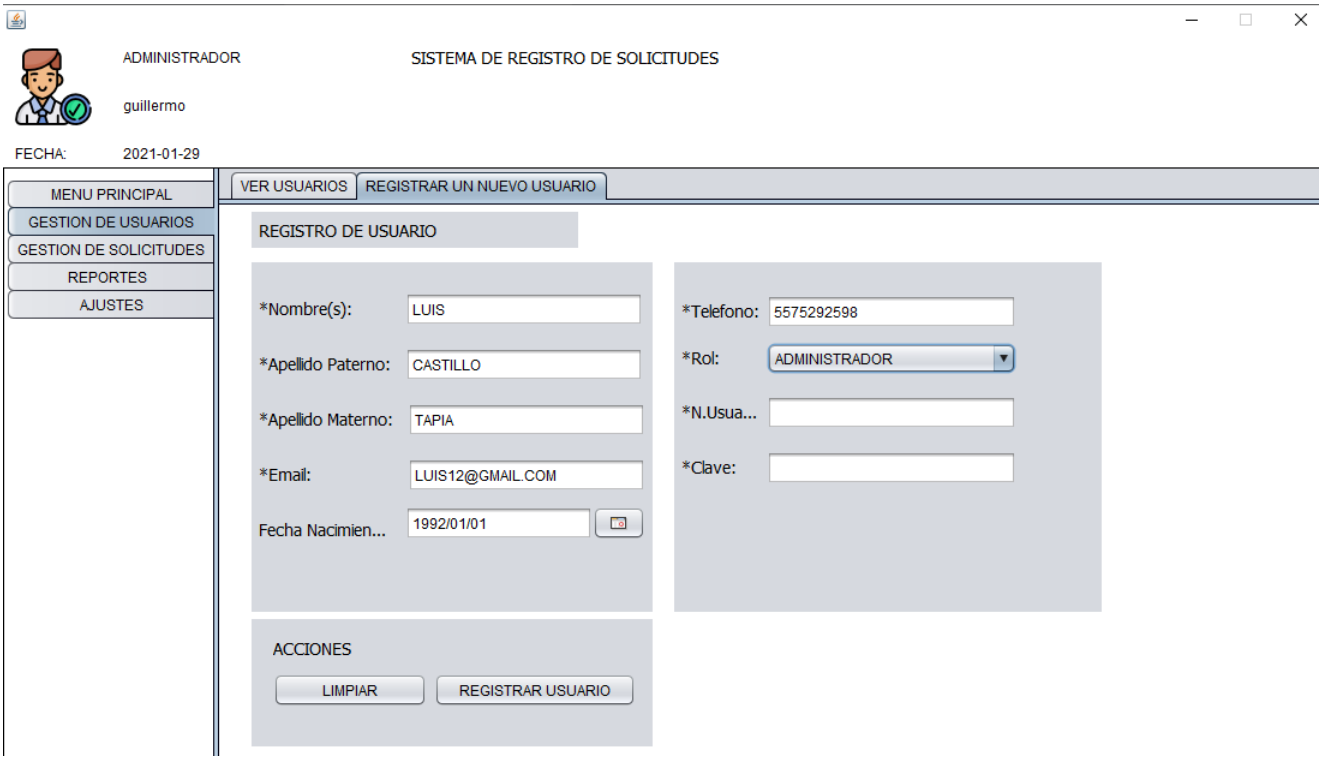


Figura 3.17: Interfaz del Registro del Usuario

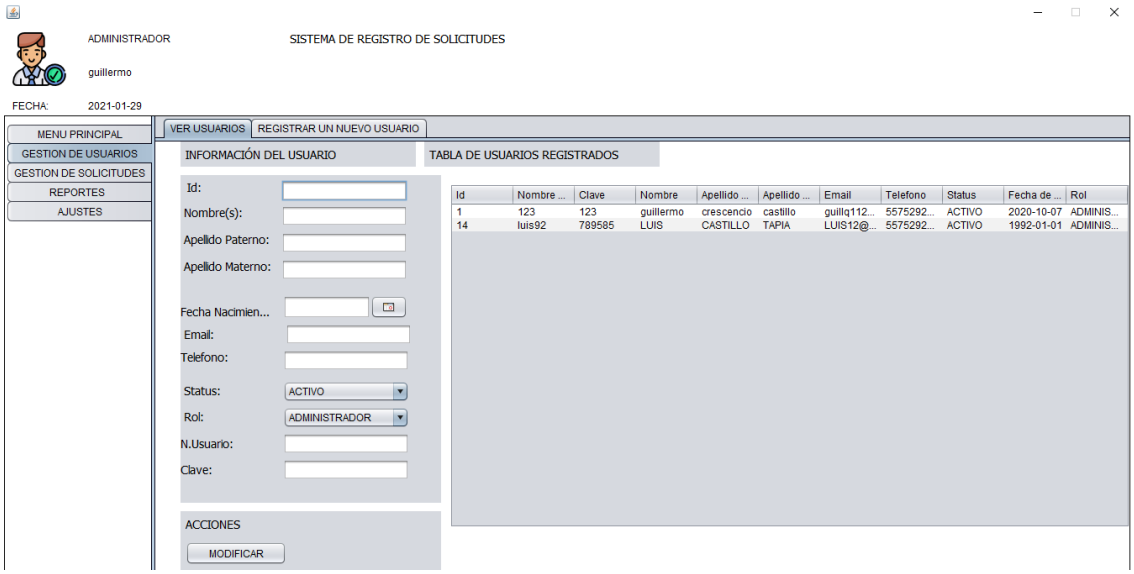


Figura 3.18: Interfaz para modificar al usuario

En la ventana de Gestión de Solicitudes se muestran dos pestañas: Ver Solicitudes y Registrar solicitudes.

La pestaña Registrar solicitudes se muestra en la figura 3.19 se tiene un formulario dividido en tres secciones: Datos del solicitante, Vivienda del solicitante y Bienes del Solicitante. Cabe mencionar que se deben de llenar cada uno de los campos.

Sección Datos del Solicitante, se le asigna el apoyo y se obtienen los datos de la persona (Nombre, Apellido Paterno, Apellido Materno, Código Postal, Localidad, Calle, Teléfono, Fecha de Nacimiento, Género, y Tipo de Apoyo).

En la Vivienda del Solicitante, se obtienen los datos del tipo de lugar donde vive el solicitante (Tipo de Vivienda en la que vives, Número de Pisos, ¿De qué material es su Piso?, ¿De qué material es el techo de la vivienda? , ¿De qué material son los muros?, ¿Cuántos cuartos se usan para dormir?, ¿Llega el agua entubada al terreno?, ¿Llega el agua al interior de la vivienda?, ¿Cuántas comidas realizan al día?, ¿Qué tipo de estufa utilizan?).

Sección Bienes del Solicitante, se seleccionan los dispositivos, transporte, y servicios con los que cuentan (Lavadora, Automóvil, Camas, Celular, Estufa, Horno de Micro Ondas, Sala, Licuadora, Comedor, Televisión, Refrigerador, Calentador Solar, Boiler, Blu ray, Internet y Ipad).

Al terminar de registrar todos los campos se le debe dar clic al botón registrar.

La pestaña Ver Solicitudes se muestra en la figura 3.20, donde se presenta en forma de tabla, así como realizar la búsqueda por código postal - localidad, al seleccionar la solicitud de la tabla, permite modificar los datos de tal solicitud.

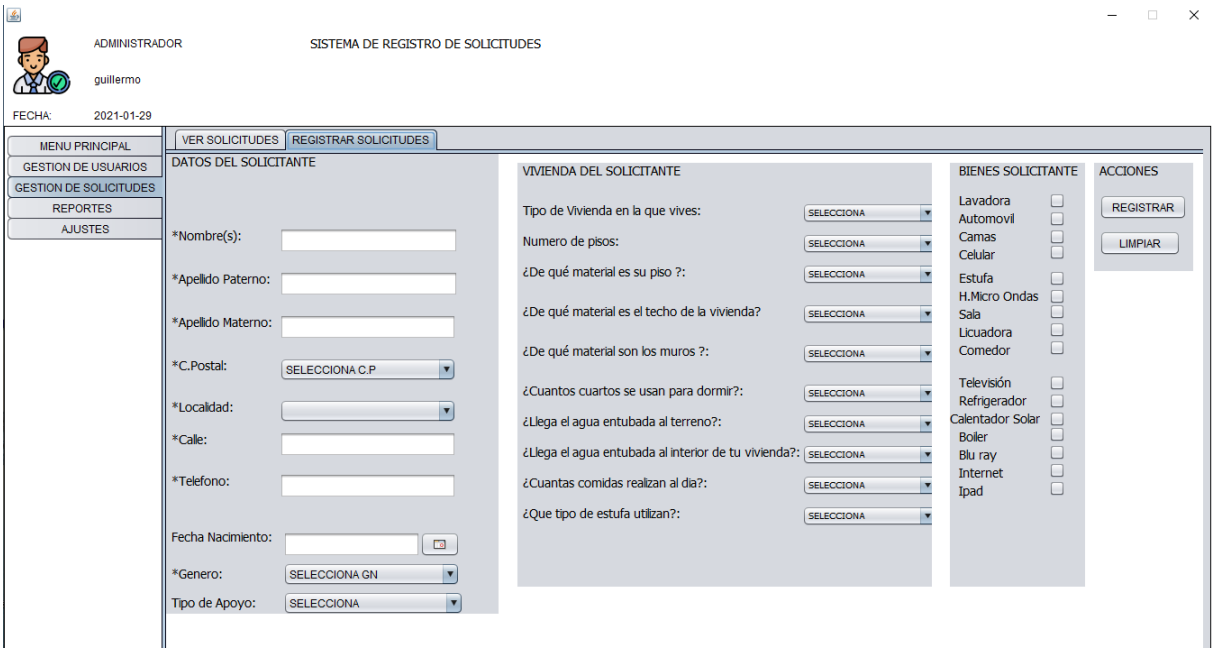


Figura 3.19: Interfaz para registrar una solicitud

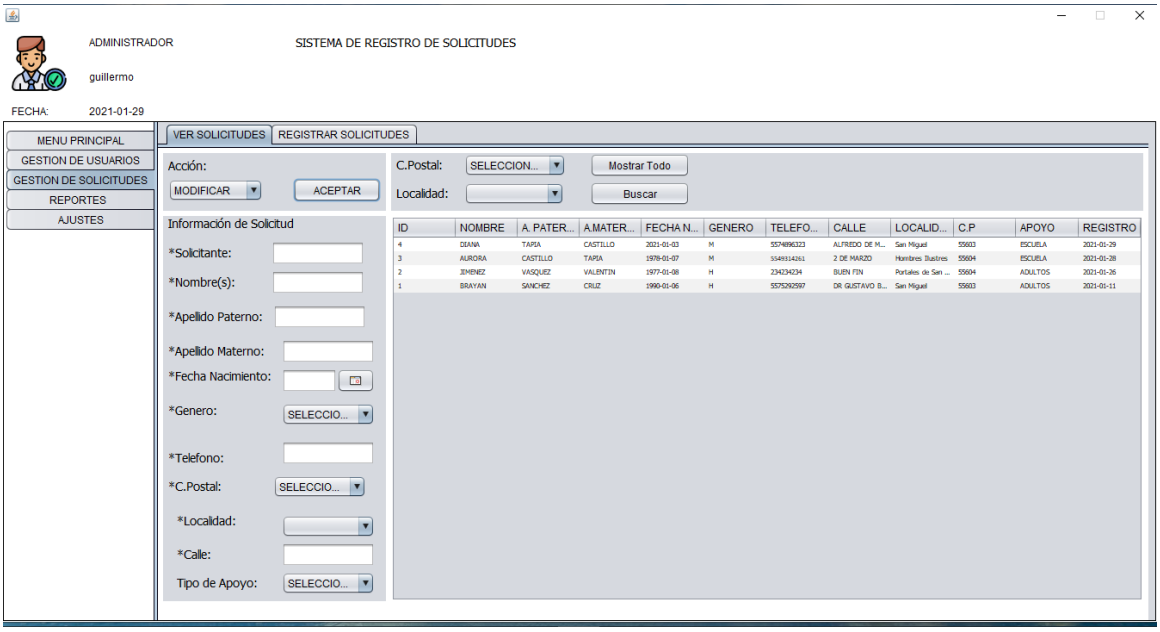


Figura 3.20: Interfaz para modificar una solicitud

En la opción de Reportes se muestra una pestaña figura 3.21, la cual contiene una tabla con todas las solicitudes registradas, se pueden generar tres tipos de reportes: por estudio, localidad - apoyo y reporte completo.

Reporte estudio: se genera cuando se selecciona un registro de la tabla y se oprime el botón de generar estudio. Este reporte contiene la información de la solicitud detalladamente como se muestra en la figura 4.5.

Reporte localidad - apoyo: se obtiene al seleccionar el código postal, localidad y el tipo de apoyo, se oprime el botón buscar localidad y apoyo, se da clic en el botón generar reporte,posteriormente mostrará un reporte como el de la figura 4.7, el cual contiene los datos de las solicitudes que corresponde con el código postal, la localidad y el tipo de apoyo seleccionado.

Reporte completo: se genera al oprimir el botón Reporte completo, este reporte contiene la información de todas las solicitudes registradas como el reporte que se muestra en la figura 4.6.

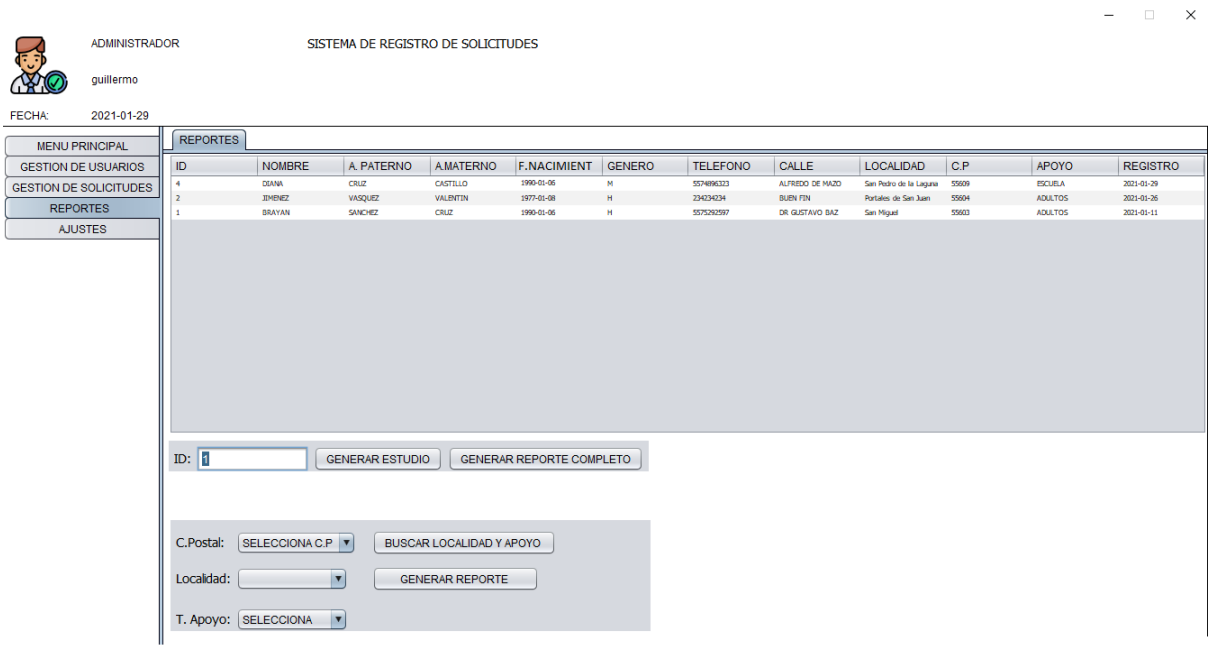


Figura 3.21: Interfaz para generar reportes

En la ventana de Ajustes figura 3.22, se muestran los apoyos que se brindan actualmente. Sin embargo el sistema permite modificar y agregar apoyos adicionales.

En el caso de Agregar un nuevo apoyo, debe seleccionar la acción de agregar y colocar el nombre del apoyo en la descripción y posteriormente dar clic en el botón aceptar.

En la ventana de Ajustes Figura 3.22, se muestran los apoyos que se brindan actualmente. Sin embargo el sistema permite modificar y agregar apoyos adicionales.

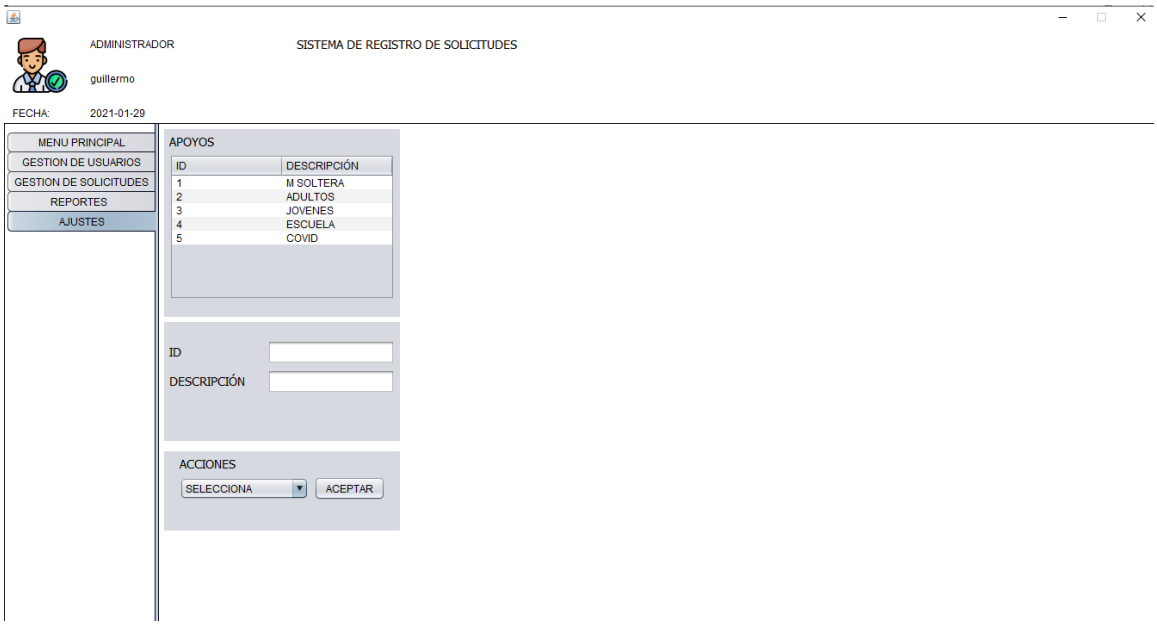


Figura 3.22: Interfaz para realizar ajustes

3.3.2 Página Web para monitoreo de solicitudes

La pantalla de la pagina Web contiene datos informativos específicos de la institución, como el logotipo del ayuntamiento. Además, cuenta con dos campos de texto donde el usuario coloca su nombre de usuario y contraseña.El botón Enviar, tiene como función validar el usuario y contraseña ingresados anteriormente. Si los datos son correctos, se direccionara a la página de registros. En la figura 3.23 se muestra el inicio de la página Web para ingresar al sistema de monitoreo de solicitudes.

Figura 3.23: Interfaz de validación de usuario Web

A continuación, se muestra el código en PHP para ingresar en la página Web, donde se muestra en la figura 3.24, en la línea 16 se realiza una consulta del usuario, contraseña y se verifica el estado del usuario, en el caso de ser correcto se redirecciona a la página de registros, si hay un error se muestra un mensaje de alerta con la leyenda "Usuario incorrecto".

```php
<?php
$alert  = '';
session_start();
if(!empty($_SESSION['active']))
{
    header('location: sistema/');
}else{
if(!empty($_POST))
{
 if(empty($_POST['usuario']) || empty($_POST['clave'])) {
  $alert = 'Ingresa usuario y clave';
 }else{
  require_once "conexion.php";
  $user=$_POST['usuario'];
  $clave=$_POST['clave'];
  $query= mysqli_query($connection, "SELECT * FROM usuario WHERE nombreUsuario='$user' AND contraseña = '$clave'
    AND status='ACTIVO' ");
  $result=mysqli_num_rows($query);
  if ($result > 0 ){
    $data = mysqli_fetch_array($query);
    $_SESSION['active'] = true;
    $_SESSION['nombreUsuario'] = $data['nombreUsuario'];
    $_SESSION['email'] = $data['email'];

            header('location: sistema/');

  }else{
    $alert = 'Usuario incorrecto';
    session_destroy();
  }
 }
}

}
?>
```

Figura 3.24: Código en PHP de validación de usuario

La página de monitoreo del sistema de registro de solicitudes muestra una tabla con los datos del ID, Nombre, Apellido Paterno, Apellido Materno, Fecha de Nacimiento, Género, Teléfono, Calle, Asentamiento, Código Postal, Descripción y Fecha de alta, con las solicitudes registradas del día actual; con el propósito de verificar si de manera remota existe comunicación. En la figura 3.25 se muestra la página con la tabla de registros, con los campos ID, Nombre, Apellido Paterno, Apellido Materno, Fecha de Nacimiento, Genero, Teléfono, Calle, Asentamiento, Código Postal, Descripción y Fecha

de Alta.

Registro de Usuarios México, 06 de Junio de 2021 | DIRECTOR

SOLCITUDES

ID	Nombre	Apellido Paterno	Apellido Materno	Fecha de Nacimiento	Genero	Teléfono	Calle	Asentamiento	Código Postal	Descripción	Fecha de Alta
1	BRAYAN	SANCHEZ	CRUZ	1990-01-06	H	5575292597	DR GUSTAVO BAZ	San Miguel	55603	SESENTAYMAS	2021-06-06
2	JIMENEZ	VASQUEZ	VALENTIN	1977-01-08	H	234234234	BUEN FIN	Portales de San Juan	55604	SESENTAYMAS	2021-06-06
4	DIANA	CRUZ	CASTILLO	1993-01-03	M	5574896323	ALFREDO DE MAZO	San Pedro de la Laguna	55609	CALENTADORSOLAR	2021-06-06
5	MARCELO	CASTILLO	TAPIA	1990-02-01	H	5575292597	RIO PACHUCA S/N	San Miguel	55603	CUARTODORMITORIO	2021-06-06

Figura 3.25: Interfaz lista de Registros Web

El código PHP que genera la tabla realiza la consulta de las solicitudes desde la línea 47 a la línea 51 del programa, dicha consulta se encarga de mostrar sólo las solicitudes del día actual. En la figura 3.26 se muestra el código fuente correspondiente.

```
46  <?php
47          $query= mysqli_query($connection, "SELECT s.idsolicitante,s.nombre,s.apellidopaterno,
48          s.apellidomaterno,s.fechaNacimiento,s.genero,s.telefono,s.calle,l.asentamiento,
49          l.codigopostal,a.descripcion,s.fechaAlta FROM solicitudes s INNER JOIN localidades l
50          ON  s.id_localidad=l.idlocalidad INNER JOIN apoyo a  ON  s.id_apoyo=a.idapoyo
51          WHERE DATE_FORMAT(s.fechaAlta, '%Y-%m-%d') = CURDATE() ");
52      $result =mysqli_num_rows($query);
53      if($result > 0 ){
54          while ($data = mysqli_fetch_array($query)) {
55          ?>
56              <tr>
57              <td><?php echo $data["idsolicitante"] ?></td>
58              <td><?php echo $data["nombre"] ?></td>
59              <td><?php echo $data["apellidopaterno"] ?></td>
60              <td><?php echo $data["apellidomaterno"] ?></td>
61              <td><?php echo $data["fechaNacimiento"] ?></td>
62              <td><?php echo $data["genero"] ?></td>
63              <td><?php echo $data["telefono"] ?></td>
64              <td><?php echo $data["calle"] ?></td>
65              <td><?php echo $data["asentamiento"] ?></td>
66              <td><?php echo $data["codigopostal"] ?></td>
67              <td><?php echo $data["descripcion"] ?></td>
68              <td><?php echo $data["fechaAlta"] ?></td>
69              </tr>
70              <?php}} ?>
```

Figura 3.26: Código en PHP de tabla de registros

Capítulo 4

PRUEBAS REALIZADAS

A continuación se muestran las pruebas realizadas al sistema para verificar su correcto funcionamiento.

4.1 Agregar usuario

La primera prueba que se presenta es la de agregar un usuario al sistema. Se ingresa un nuevo usuario con rol administrador, para gestionar a los demás usuarios. Cabe mencionar que previamente ya se habían agregado usuarios usando la consola del sistema gestor de base de datos, para efectos de pruebas durante el desarrollo. Para demostrar el funcionamiento, se usarán los siguientes datos: el nombre del usuario es Luisa Castillo Jiménez, quien nació el 01 de febrero de 1990, cuenta con el correo LUISA1004@GMAIL.COM y su número de teléfono es el 5575292597, su nombre de usuario es Administradora y la contraseña es Admini1123680. En la figura 4.1 se muestran las vistas del llenado del formulario que tiene el registro de usuario, y su registro en la tabla de usuarios. Con esto se demuestra que el sistema almacena correctamente los datos que se capturaron.

(a) Formulario de Registro de Usuario

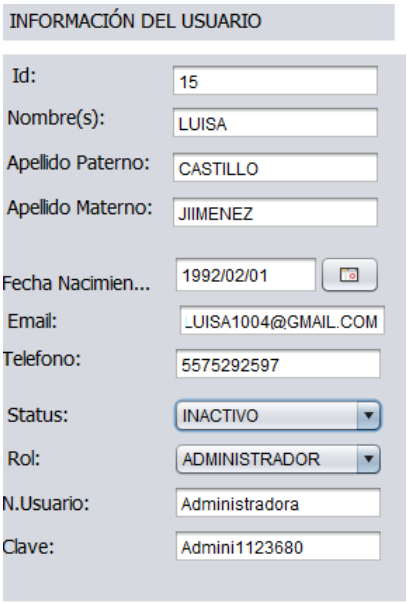
(b) Tabla de Usuarios Registrados

Figura 4.1: Prueba ingresar usuario

4.2 Editar Usuario

La modificación de los datos y cambio de estatus de un usuario, es una función crucial del sistema desarrollado. Al modificar el estatus del usuario Administradora (agregado en la prueba anterior) a "INACTIVO", permite validar que el usuario no puede acceder al sistema. Prueba que demuestra el correcto funcionamiento del sistema, y permite validar que el usuario no puede acceder al sistema. Prueba que demuestra el correcto funcionamiento del sistema. Esta modificación provocará que dicho usuario ya no tenga acceso al sistema. En la figura 4.2 se muestran las vistas de la modificación del status

del usuario, quedando demostrado el funcionamiento correcto de esta parte del sistema.



(a) Modificación de Usuario

Id	Nombre ...	Clave	Nombre	Apellido ...	Apellido ...	Email	Telefono	Status	Fecha de ...	Rol
1	123	123	guillermo	crescencio	castillo	guillq112...	5575292...	ACTIVO	2020-10-07	ADMINIS...
14	Auxiliar1	123456	LUIS	CASTILLO	TAPIA	LUIS12@...	5575295...	INACTIVO	1992-01-01	USUARIO
15	Administr...	Admini11...	LUISA	CASTILLO	JIIMENEZ	LUISA10...	5575292...	INACTIVO	1990-02-01	ADMINIS...

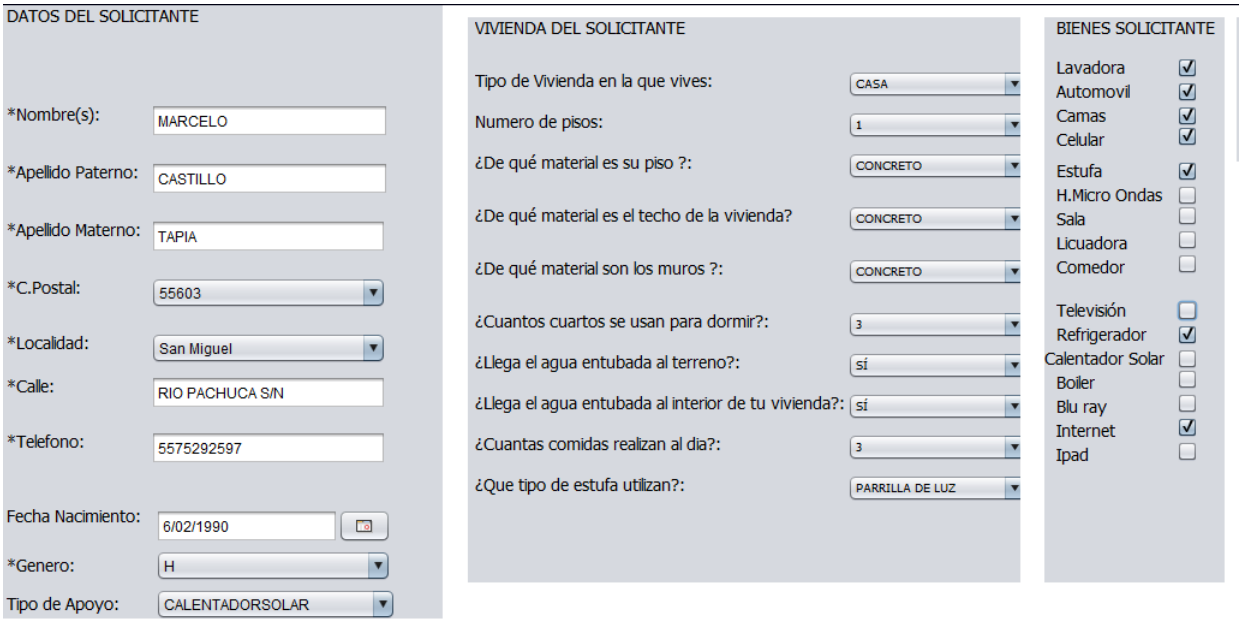
(b) Tabla de Usuarios

Figura 4.2: Prueba editar datos de usuario

4.3 Ingresar Solicitud

El ingreso de solicitudes es una de las funciones mas comúnmente usadas del sistema más comúnmente usadas. Para probar la captura de una solicitud de apoyo, se utilizará el ejemplo de Calentador Solar. El usuario debe colocar los datos del solicitante en la interfaz mostrada en la figura 3.19 anteriormente, para este ejemplo se registra al señor Marcelo Castillo Tapia con fecha de nacimiento del 06 de Febrero de 1990, con domicilio en la calle Rio Pachuca s/n ,Barrio de San Miguel, código postal 55603; la infraestructura de la vivienda es de concreto de un solo piso, cuenta con seis cuartos de los cuales tres son dormitorios, el número de comidas al día son tres y cuenta con toma de agua de la red pública; sus bienes son: parrilla de luz, lavadora, automóvil,

camas, celular, estufa, refrigerador e internet. En la figura 4.3 se muestran las vistas del registro de la solicitud así como la tabla de solicitudes donde se muestra que el registro de la solicitud fue exitoso.



(a) Llenado de Campos

ID	NOMBRE	A. PATER...	A.MATER...	FECHA N...	GENERO	TELEFO...	CALLE	LOCALID...	C.P	APOYO	REGISTRO
5	MARCELO	CASTILLO	TAPIA	1990-02-06	H	5575292597	RIO PACHUCA ...	San Miguel	55603	CALENTADORS...	2021-02-06
4	DIANA	CRUZ	CASTILLO	1993-01-03	M	5574896323	ALFREDO DE M...	San Pedro de la ...	55609	CALENTADORS...	2021-01-29
2	JIMENEZ	VASQUEZ	VALENTIN	1977-01-08	H	234234234	BUEN FIN	Portales de San ...	55604	ADULTOS	2021-01-26
1	BRAYAN	SANCHEZ	CRUZ	1990-01-06	H	5575292597	DR GUSTAVO B...	San Miguel	55603	ADULTOS	2021-01-11

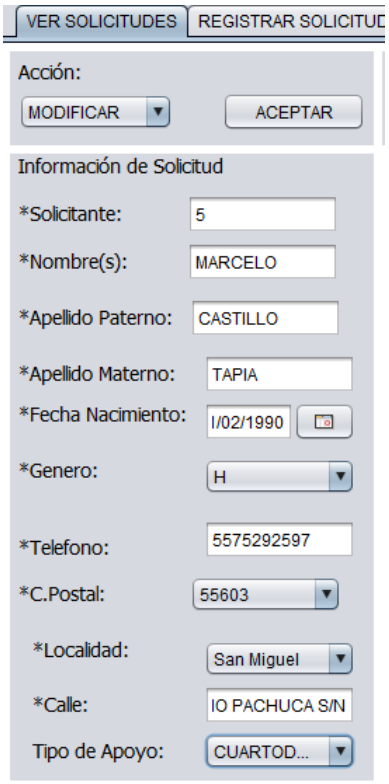
(b) Tabla de Solicitudes

Figura 4.3: Prueba ingresar Solicitud

4.4 Modificar Solicitud

El sistema permite realizar cambios en los datos de una solicitud, derivado de factores como errores en la captura o por el cambio de domicilio del solicitante. A continuación se muestra la actualización de los datos de la solicitud de prueba, capturada anteriormente. Se modifica el apoyo de la solicitud de Marcelo Castillo Tapia, ya que el apoyo que requiere no es el de Calentador Solar, sino el de Cuarto Dormitorio. En la figura 4.4 se

muestran las vistas del cambio de apoyo de la solicitud de Marcelo Castillo Tapia y la tabla de solicitudes con el cambio de apoyo a Cuarto Dormitorio.



(a) Datos de Solicitud

ID	NOMBRE	A. PATER...	A.MATER...	FECHA N...	GENERO	TELEFO...	CALLE	LOCALID...	C.P	APOYO	REGISTRO
5	MARCELO	CASTILLO	TAPIA	1990-02-01	H	5575292597	RIO PACHUCA ...	San Miguel	55603	CUARTODORMI...	2021-02-06
4	DIANA	CRUZ	CASTILLO	1993-01-03	M	5574896323	ALFREDO DE M...	San Pedro de la ...	55609	CALENTADORS...	2021-01-29
2	JIMENEZ	VASQUEZ	VALENTIN	1977-01-08	H	234234234	BUEN FIN	Portales de San ...	55604	ADULTOS	2021-01-26
1	BRAYAN	SANCHEZ	CRUZ	1990-01-06	H	5575292597	DR GUSTAVO B...	San Miguel	55603	ADULTOS	2021-01-11

(b) Tabla de Solicitudes

Figura 4.4: Prueba modificar Solicitud

4.5 Generación de reportes

El sistema tiene como objetivo generar diferentes tipos de reportes. Para verificar su correcto funcionamiento, se prueba uno a uno de forma particular.

4.5.1 Reporte por solicitante

Se genera como ejemplo el formato de solicitud del usuario Marcelo Castillo Tapia para imprimir o para enviar en formato electrónico. Primero debe ingresar a la ventana de reportes del sistema, seleccionar al usuario en la tabla correspondiente y dar click en el botón de generar reporte. En la figura 4.5 se muestra la vista de ventana de reportes, y el formato de solicitud generado exitosamente en PDF.

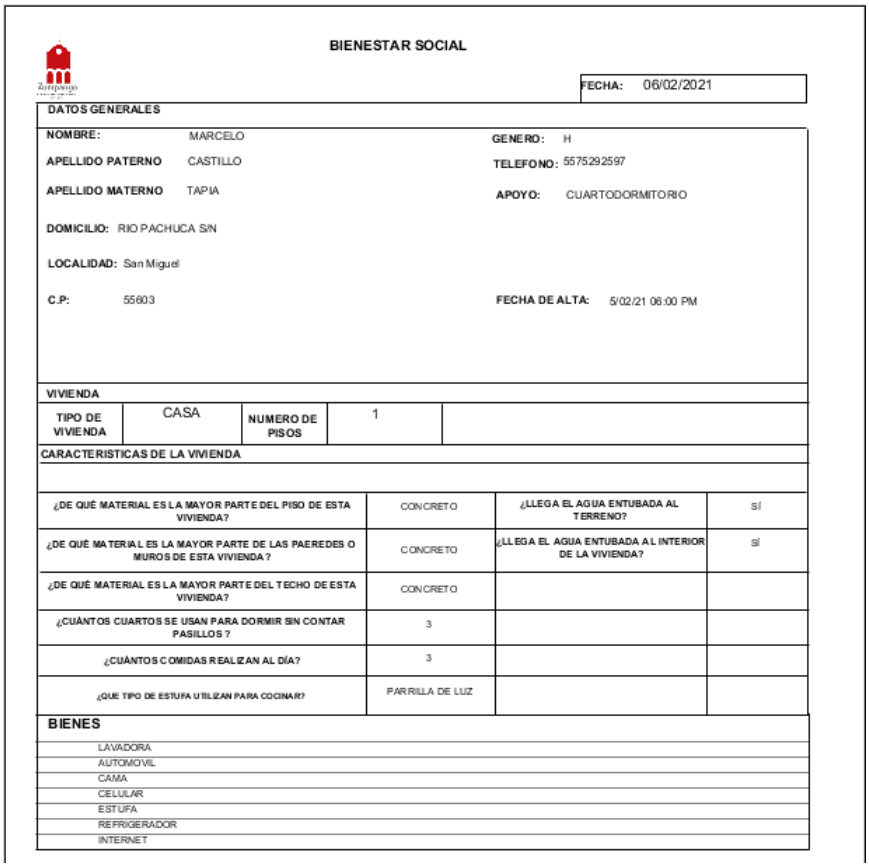
4.5.2 Generación de Reporte global

De acuerdo al flujo de trabajo de la presidencia municipal, se debe presentar a Tesorería un reporte completo de todas las solicitudes realizadas, esto con el fin de brindar los apoyos registrados. Para esto, se debe ingresar a la ventana de reportes y dar click en el botón de generar reporte completo. En la figura 4.6 se muestra la ventana de Reportes la cual contiene el botón de generar reporte completo y la vista del reporte generado.

4.5.3 Generación de reporte por localidad y tipo de apoyo

Otro de los reportes que se debe de entregar en Tesorería es el de localidad y apoyo especifico, con la finalidad de validar el número de solicitudes. Como ejemplo, se requiere verificar el número de personas que solicitaron el apoyo de Cuarto dormitorio en el Barrio de San Miguel de Zumpango. Por lo que debe ingresar, se debe ingresar a la ventana de reportes y seleccionar el código postal, localidad y tipo de apoyo que Tesorería solicita, dar click en el botón de buscar localidad y apoyo. Después dar click al botón generar reporte. En la figura 4.7 se muestra la selección del código postal, localidad, tipo de apoyo y la vista del reporte generado.

(a) Ventana de Reportes



BIENESTAR SOCIAL

FECHA: 06/02/2021

DATOS GENERALES

NOMBRE: MARCELO
APELLIDO PATERNO CASTILLO
APELLIDO MATERNO TAPIA
DOMICILIO: RIO PACHUCA S/N
LOCALIDAD: San Miguel
C.P: 55603
GENERO: H
TELEFONO: 5575292597
APOYO: CUARTODORMITORIO
FECHA DE ALTA: 5/02/21 06:00 PM

VIVIENDA

TIPO DE VIVIENDA	CASA	NUMERO DE PISOS	1	

CARACTERISTICAS DE LA VIVIENDA

¿DE QUÉ MATERIAL ES LA MAYOR PARTE DEL PISO DE ESTA VIVIENDA?	CONCRETO	¿LLEGA EL AGUA ENTUBADA AL TERRENO?	SÍ
¿DE QUÉ MATERIAL ES LA MAYOR PARTE DE LAS PAEREDES O MUROS DE ESTA VIVIENDA ?	CONCRETO	¿LLEGA EL AGUA ENTUBADA A L INTERIOR DE LA VIVIENDA?	SÍ
¿DE QUÉ MATERIAL ES LA MAYOR PARTE DEL TECHO DE ESTA VIVIENDA?	CONCRETO		
¿CUÁNTOS CUARTOS SE USAN PARA DORMIR SIN CONTAR PASILLOS ?	3		
¿CUÁNTOS COMIDAS REALIZAN AL DÍA?	3		
¿QUE TIPO DE ESTUFA UTILIZAN PARA COCINAR?	PARRILLA DE LUZ		

BIENES

LAVADORA
AUTOMOVIL
CAMA
CELULAR
ESTUFA
REFRIGERADOR
INTERNET

(b) Reporte Generado

Figura 4.5: Prueba generación de reporte de individual

(a) Ventana de Reportes

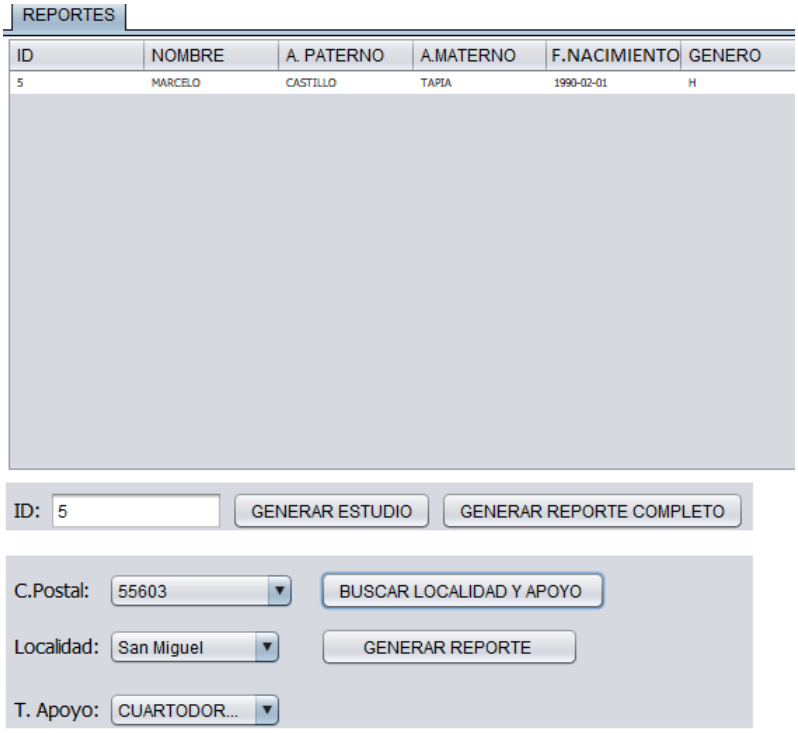
BIENESTAR SOCIAL

REGISTROS: 4 FECHA: 06/02/2021

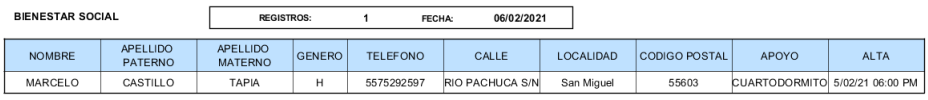
NOMBRE	PATERNO	MATERNO	GENERO	TELEFONO	DIRECCION	LOCALIDAD	C.P	DESCRIPCIÓN	ALTA
BRAYAN	SANCHEZ	CRUZ	H	5575292597	DR GUSTAVO BAZ	San Miguel	55603	ADULTOS	10/01/21 06:00
JIMENEZ	VASQUEZ	VALENTIN	H	234234234	BUEN FIN	Portales de San Juan	55604	ADULTOS	25/01/21 06:00
DIANA	CRUZ	CASTILLO	M	5574896323	ALFREDO DE MAZO	San Pedro de la	55609	CALENTADORSOLAR	28/01/21 06:00
MARCELO	CASTILLO	TAPIA	H	5575292597	RIO PACHUCA S/N	San Miguel	55603	CUARTODORMITORIO	5/02/21 06:00 PM

(b) Reporte Generado

Figura 4.6: Prueba generación de reporte global



(a) Ventana de Ajustes

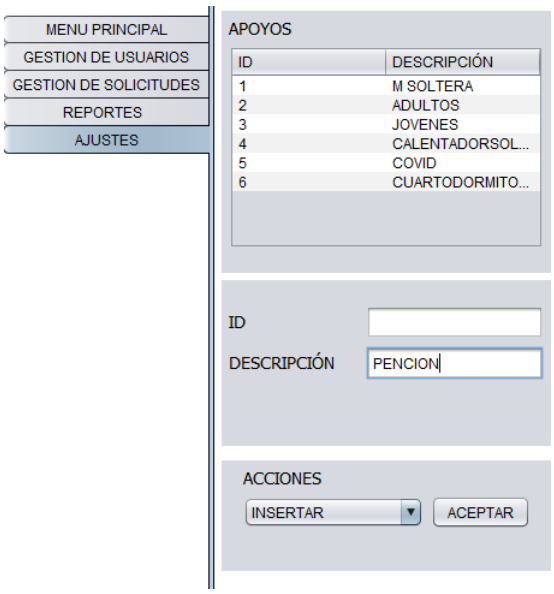


(b) Reporte Generado

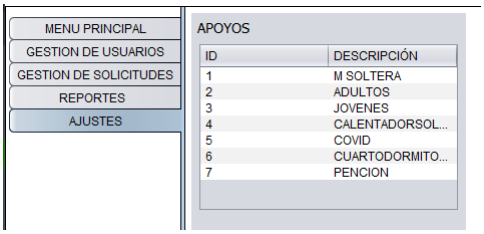
Figura 4.7: Generación de Reporte por Localidad y Apoyo

4.6 Agregar nuevo tipo de apoyo

El sistema permite agregar o eliminar nuevos tipos de apoyo, lo que le da la flexibilidad para adaptarse a nuevas necesidades. Por ejemplo si el Ayuntamiento de Zumpango ubicado en el Estado de México, realizó la apertura del apoyo Pensión. Y si el apoyo no esta dado de alta, se puede registrar en el sistema. En este caso se debe ingresar a la ventana de ajustes, seleccionar la acción de insertar y en el campo de descripción debe colocar el nombre del apoyo requerido. En la figura 4.8 se muestra la ventana de ajustes donde se selecciona la opción insertar, se rellena el campo de descripción con el nombre del apoyo y la tabla apoyos con el nuevo registro.



(a) Ventana de Ajustes



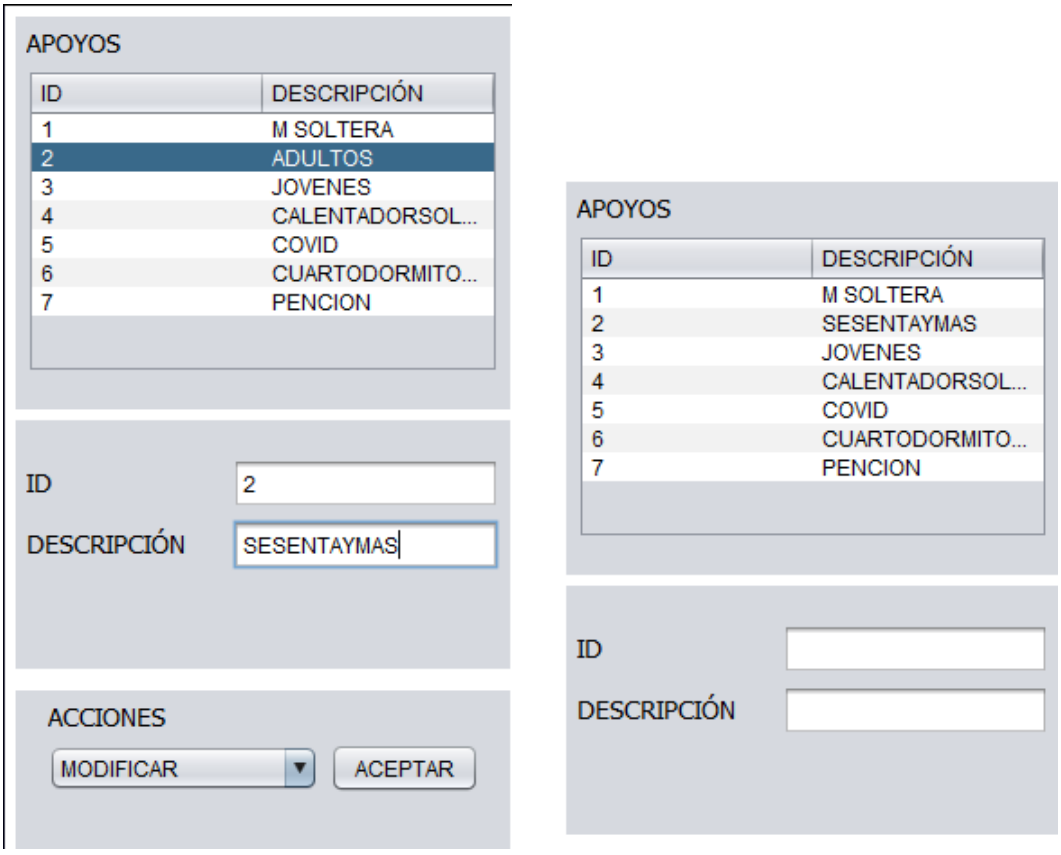
(b) Tabla de Apoyos

Figura 4.8: Prueba de insertar nuevo apoyo

4.7 Modificar Apoyo

Para modificar los datos del sistema fácilmente, se implementó la opción para editar la información de los diferentes tipos de apoyo. Suponer el escenario siguiente: por decisión del Ayuntamiento de Zumpango, Estado de México, se requiere cambiar el nombre del apoyo Adultos por el de "Sesenta y Más". Para realizar dicho cambio se debe ingresar a la ventana de ajustes, seleccionar la acción de modificar, seleccionar el apoyo a modificar y cambiar el nombre del campo de descripción.

En la figura 4.9 se muestra la ventana de ajustes, en la cual se selecciona la opción modificar, se elige el apoyo y se cambia la descripción a Sesenta y Más. En la tabla de apoyos se muestra el nuevo nombre capturado.



(a) Ventana de Ajustes (b) Tabla de Apoyos

Figura 4.9: Prueba modificar apoyo

4.8 Validar los Registros por día

Con el objetivo de dar seguimiento a las solicitudes y entregas de apoyos a la ciudadanía municipal, se requiere validar el registro de las solicitudes registradas anteriormente, desde la página Web. A la que puede ingresar proporcionando usuario y contraseña. En caso de que los datos sean correctos, se muestra la lista de solicitudes que se realizaron el día del acceso. En la figura 4.10 se muestra la página de acceso. Después de agregar los datos correctos, se muestra la lista de los solicitantes registrados.

(a) Interface de Validación de usuario Web

Registro de Usuarios — México, 06 de Junio de 2021 | DIRECTOR

SOLCITUDES

ID	Nombre	Apellido Paterno	Apellido Materno	Fecha de Nacimiento	Genero	Teléfono	Calle	Asentamiento	Código Postal	Descripción	Fecha de Alta
1	BRAYAN	SANCHEZ	CRUZ	1990-01-06	H	5575292597	DR GUSTAVO BAZ	San Miguel	55603	SESENTAYMAS	2021-06-06
2	JIMENEZ	VASQUEZ	VALENTIN	1977-01-08	H	234234234	BUEN FIN	Portales de San Juan	55604	SESENTAYMAS	2021-06-06
4	DIANA	CRUZ	CASTILLO	1993-01-03	M	5574896323	ALFREDO DE MAZO	San Pedro de la Laguna	55609	CALENTADORSOLAR	2021-06-06
5	MARCELO	CASTILLO	TAPIA	1990-02-01	H	5575292597	RIO PACHUCA S/N	San Miguel	55603	CUARTODORMITORIO	2021-06-06

(b) Interface lista de Registros Web

Figura 4.10: Pagina Web

De acuerdo a las pruebas realizadas al sistema, se encontró que cumple y ejecuta correctamente todas y cada una de las funciones implementadas. Por lo que se considera

que el sistema es correcto en diseño y funcionamiento.

CONCLUSIONES

En esta tesis se implementó una propuesta de solución para resolver un problema del área de Bienestar Social del Ayuntamiento de Zumpango, en el Estado de México. Al momento de empezar esta tesis, en el Municipio de Zumpango se realizaba el registro de solicitudes del área de Bienestar Social de una manera manual, apoyado en hojas de cálculo, lo que provocaba una atención lenta a los solicitantes, además de otros inconvenientes, como una mala comunicación entre el personal del área y los solicitantes, esto al no poder darles a estos últimos una respuesta concisa y rápida.

Por los motivos antes expuestos, se desarrolló un sistema informático web capaz de cubrir las necesidades del registro de solicitudes de dicha área.

Entre las principales problemáticas de llevar un registro manual, y que el sistema propuesto resuelve, se encuentran las siguientes: datos incompletos en las solicitudes capturadas, falta de seguimiento de la solicitud y pérdida de información, y retrasos en la entrega de reportes. La identificación de estos problemas, así como la elección de la metodología de desarrollo, herramientas y tecnologías a emplear fueron parte de los objetivos de esta tesis. De esta forma, los objetivos específicos planteados al inicio de este trabajo se cumplieron satisfactoriamente de la siguiente manera:

- Se presentaron las herramientas y tecnologías usadas en el desarrollo del sistema.
- Se identificaron los requerimientos del sistema, mediante la técnica de observación.
- Se presentó el diseño del sistema, así como la representación gráfica de la información en las interfaces y la generación de diferentes reportes en el sistema.

- Se usó UML para presentar la arquitectura del sistema. Para la parte del modelo de la Base de Datos, la cual está basada en el Modelo Relacional.

- Se aplicó la metodología Scrum para el desarrollo del sistema. Al ser el autor de este documento y el único integrante del equipo Scrum, asumí todos los roles, por lo que se omitieron algunas de las etapas de reuniones y documentación.

- Se realizaron las pruebas pertinentes para verificar el correcto funcionamiento del sistema.

En una etapa posterior, y como un proyecto futuro, se planea la integración de notificaciones a las personas que solicitan su apoyo mediante correo electrónico o mensaje de texto a celular, así como generar una aplicación para Android y también para iOS.

Referencias

[1] Abenza, P. P. G. *Comenzando a programar con JAVA*. Universidad Miguel Hernández, 2015.

[2] Aguirre Barrera, J., Aguirre Barrera, S., et al. Metodologías para el desarrollo de proyectos.

[3] Alvarez, R., Jareño, A., Salvat, P., Robles, R. L., Folch, J. S., Alvarez, M. A., and Alvarez, S. Tutorial de sql. *DesarrolloWeb. Com*.

[4] Arias, M. A. *Introducción a PHP*. IT Campus Academy, 2013.

[5] Arias, M. Á. *Aprende Programacion Web con PHP y MySQL: 2ª Edicion*. IT campus Academy, 2017.

[6] Barnes, D. J., Kölling, M., and Brenta, B. I. *Programación orientada a objetos con Java*. Pearson Educación, 2007.

[7] Bertino, E., and Martino, L. *Sistemas de bases de datos orientadas a objetos: conceptos y arquitecturas*. Ediciones Díaz de Santos, 1995.

[8] Cardona, M. G., Sorza, J. D., Posada, S. L., Carmona, J., Ayala, S. A., and Alvarez, O. L. Establecimiento de una base de datos para la elaboración de tablas de contenido nutricional de alimentos para animales. *Revista Colombiana de Ciencias Pecuarias 15*, 2 (2002), 240–246.

[9] de Guevara, J. M. L. *Fundamentos de programación en Java*. Consultar, 2014.

[10] Deitel, H. M., and Deitel, P. J. *Como programar en Java*. Pearson Educación, 2003.

[11] Delgado Olivera, L. d. l. C., and Díaz Alonso, L. M. Modelos de desarrollo de software. *Revista Cubana de Ciencias Informáticas 15*, 1 (2021), 37–51.

[12] Durán, R. P., et al. *Scrum-¡ Guía definitiva de prácticas ágiles esenciales de Scrum!* Babelcube Inc., 2016.

[13] Fraga, E. S., and Ribeiro, J. R. O poder do java. *INICIA 37* (2003), 58.

[14] Gabillaud, J. *SQL Server 2014: SQL, Transact SQL, diseño y creación de una base de datos (con ejercicios prácticos corregidos)*. Ediciones ENI, 2015.

[15] Godoc, E. *SQL: Los fundamentos del lenguaje*. Ediciones ENI, 2014.

[16] Goytia, L. L., and González, Á. G. *Programación orientada a objetos C++ y Java*. Grupo Editorial Patria, 2014.

[17] Groussard, T. *JAVA 7: Los fundamentos del lenguaje Java*. Ediciones Eni, 2012.

[18] hacer Ser, T. S. S. Unidad temática 5 sistema gestor de bases de datos (sgbd). *Universidad Tecnológica de Puebla Tecnologías de la Información y Comunicación* (2012), 59.

[19] Harvey, M. D., and DEITEL, P. *Como Programar en Java*. Prentice Hall, 1998.

[20] Korth, H., and Silberschatz, A. *Fundamentos de bases de datos*. Madrid, 1993.

[21] Llinás, L. F. G. *Programación orienta a objetos en Java*. Universidad del Norte, 2010.

[22] Marqués Andrés, M., et al. *Bases de datos*. Universitat Jaume I, 2011.

[23] Martin, J., Odell, J. J., and Velasco, O. A. P. *Análisis y diseño orientado a objetos*. Prentice Hall Hispanoamericana, 1994.

[24] Martín, M. J. R., Martín, A. R., and Rodríguez, F. M. *Sistemas gestores de bases de datos*. Madrid, España: McGraw-Hill, 2006.

[25] Mundo, H. Programación orientada a objetos.

[26] Nazareno, R., Leone, H. P., and Gonnet, S. M. Trazabilidad de procesos ágiles: un modelo para la trazabilidad de procesos scrum. In *XVIII Congreso Argentino de Ciencias de la Computación* (2013).

[27] Oppel, A., and Sheldon, R. *SQL*. McGraw-Hill Professional Publishing, 2008.

[28] Paré, R. C., Costa, D. C., and Escofet, C. M. *Bases de datos*. UOC papers, 2002.

[29] Payne, A., and Phillips, N. *Desarrollo*. Alianza Editorial Madrid, 2012.

[30] Prieto Álvarez, C. G., et al. Adaptación de las metodologías tradicionales cascada y espiral para la inclusión de evaluación inicial de usabilidad en el desarrollo de productos de software en méxico. *REPOSITORIO NACIONAL CONACYT* (2015).

[31] Quingaluisa, A., Parra, R., and Gómez, E. *Análisis de Test Driven Development y Scrum para el desarrollo rápido de aplicaciones basadas en Java y AngularJs*.

[32] Ricardo, C. M. *Bases de datos*. McGraw Hill Educación, 2009.

[33] Rivas, Arzaluz, M. D. L. Modelo entidad-relación. UAEMEX. http://hdl.handle.net/20.500.11799/35201, 2015. Consultado: 2021-01-25.

[34] Rivera, F. L. O. *Base de datos relacionales*. ITM, 2008.

[35] Ruiz, I. L., and Gómez-Nieto, M. Á. *Bases de Datos Desde Chen Has Codd Con Oracle*. Alfaomega Grupo Editor, 2002.

[36] Sánchez, J. Diseño conceptual de bases de datos. *Recuperado de https://www. freelibros. me/base-de-datos/dise-no-conceptual-de-bases-de-datos* (2004).

[37] Sánchez, J. Principios sobre bases de datos relacionales. *Informe, Creative Commons 11* (2004), 20.

[38] Santillán, L. A. C., Ginestà, M. G., and Mora, Ó. P. *Bases de datos en MySQL*. 2014.

[39] Trigás Gallego, M. Metodologia scrum. Universitat Oberta de Catalunya. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf, 2012. Consultado: 2021-08-01.

[40] Vértice, E. *Diseño básico de páginas web en HTML*. Editorial Vértice, 2009.