UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE DE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

SISTEMA INFORMÁTICO ADMINISTRADOR DE COMITÉ DE AGUA POTABLE SANTA JULIA

Trabajo de Titulación para optar al Título
Técnico Universitario en INFORMÁTICA

Integrantes:
Stephanie Coiro Huber
Jesús Jiménez Jiménez
Profesor Guía:
Sr. Dagoberto Cabrera Tapia

2018

RESUMEN

KEYWORDS: sistema informatico, comité agua potable, comunidad santa julia, boletas, medidor de agua potable.

Con la finalidad de automatizar, controlar, administrar y mejorar la generación de boletas y comprobantes de pago del COMITÉ DE AGUA POTABLE SANTA JULIA, se decidió desarrollar un software capaz de realizar todas estas acciones y así facilitar la labor de los usuarios de la empresa.

En el capítulo I de este informe se describirá la situación actual de la empresa. La mayoría de las actividades informáticas de la empresa, siendo algunas de estas el registro de los nuevos clientes y medidores, el cálculo de consumo de agua por cliente, la generación de boletas y comprobantes de pago, se realizaban de manera manual y consumían una gran cantidad de tiempo y esfuerzo en completarlas. Por otro lado, la mala manipulación de los datos y no contar con un respaldo óptimo y actualizado, generaba bastantes problemas para la empresa, siendo uno de estos la inconsistencia de los datos. Para finalizar se presenta una posible solución de un sistema del tipo informático llamado ‘Sistema informático administrador de comité de agua potable santa julia’.

En el capítulo II se describen detalladamente los componentes del hardware y software a utilizar por el sistema informático a desarrollar. Además, se explican las tablas que integran el modelo de datos, con sus respectivos atibutos y tipo de datos a utilizar.

Para finalizar en el capítulo III, se presenta el diagrama modular, el cual permite apreciar todas las funcionalidades que integran el sistema, y el diagrama de menú que permite visualizar cómo el usuario se desplazará en el sistema. Además, se describirán de manera detallada las funciones desarrolladas en este sistema con sus correspondientes diagramas de bloques, reglas de negocios, vistas del sistema y, en algunos casos, su anexo.

ÍNDICE

INTRODUCCIÓN

COMITÉ DE AGUA POTABLE RURAL SANTA JULIA es una empresa dedicada a la producción y distribución de agua potable para las personas de Santa Julia, ciudad de Quintero, con un número de clientes, hasta el momento, que ascienden a los 100.

Una importante problemática de esta empresa es que no cuenta con un sistema informático para llevar a cabo las actividades relacionadas a ella. El registro de sus clientes, los cálculos necesarios para el cobro del servicio y la creación de boletas, entre otros, se hacen de manera manual para luego ser llevados a una planilla Excel, sin realizar un respaldo apropiado para sus archivos o documentos.

Por ello, se pretende desarrollar e implementar un sistema informático capaz de solucionar los problemas de tipo informático, que esta empresa presenta en la actualidad.

Por consecuencia, el "Sistema Informático Administrador de Comité de Agua Potable Santa Julia" tiene la intención de facilitar la labor de los usuarios en la realización de las diferentes tareas que la empresa lleva a cabo.

Entre otras cosas, el sistema permitirá efectuar los diferentes cálculos que la empresa necesita para poder conocer los totales a pagar por consumo de agua potable y para la emisión de las boletas, con sus comprobantes de pago, correspondientes a cada cliente, así como un registro eficiente de éstos.

Dado lo explicado la intención de este trabajo de título es desarrollar el sistema informático descrito, utilizando herramientas tales como Visual Basic para la interacción usuario-máquina y MariaDB para desarrollar la base de datos.

CAPÍTULO I:

ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

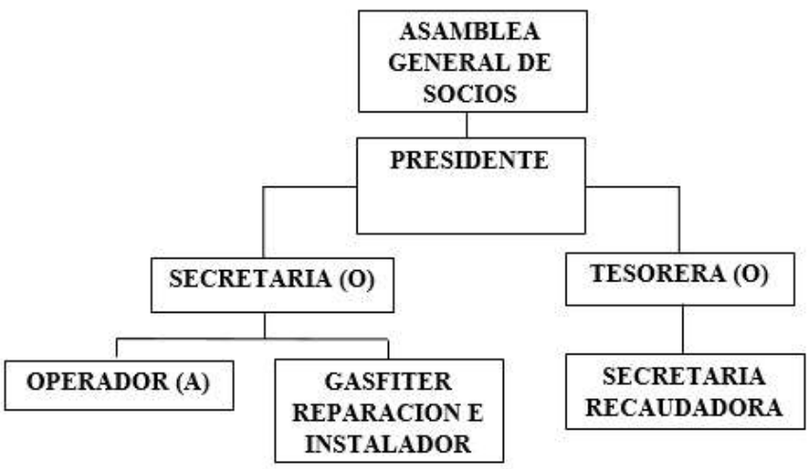
CAPÍTULO I: ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1.1. DESCRIPCIÓN DE LA ORGANIZACIÓN

El sistema a desarrollar corresponde a la empresa "COMITÉ DE AGUA POTABLE RURAL SANTA JULIA", la cual está a cargo de don Carlos Donoso Bernal, ubicada en Santa Julia, ciudad de Quintero (Ver figura 1.1).

(Fig. 1.1) Localización Santa Julia, Fuente Google Maps, Fecha Marzo del 2017

Organigrama de la organización



(Fig.1.2) Organigrama de la Empresa, Fuente COMITÉ DE AGUA POTABLE SANTA JULIA

La empresa se dedica a dar servicios de agua potable a las personas de Santa Julia, sector rural de Quintero.

El objetivo fundamental del Comité es administrar, operar, mantener y ampliar toda la infraestructura y reponer los equipos, correspondientes al Sistema de Agua Potable Rural, con la asesoría y la supervisión de la Dirección de Obras Hidráulicas del Ministerio de Obras Públicas. Sus objetivos específicos más destacados son:

- Adquirir y/o producir agua cruda para su tratamiento y su distribución como agua potable.
- Aplicar las tarifas por los consumos de agua potable.
- Cuenta con una oficina de carácter pública la que se dedica a la recepción, recaudación y distribución de los documentos generados por el consumo de agua potable.
- Adquirir los materiales necesarios para la reposición, mejoramiento y ampliación de las instalaciones del Sistema.

Como punto fundamental del comité es distribuir agua potable de calidad, cantidad y continuidad, de acuerdo a la normativa y a la capacidad técnica del servicio a todos los clientes pertenecientes a esta entidad.

1.2. DESCRIPCIÓN DE LA SITUACIÓN ACTUAL

Actualmente la empresa realiza todas sus funciones informáticas de manera manual, utilizando planillas Excel para los distintos registros y cálculos que se llevan a cabo.

“Comité de agua potable rural Santa Julia” cuenta con una oficina de atención al público, donde se realiza toda la actividad que tiene relación al registro o eliminación de clientes y la cancelación de los pagos de consumo de agua a través de boletas y comprobantes, así como cada uno de los cálculos.

Para conocer los detalles de consumo de cada cliente, y poder calcular cuánto es lo que éste debe cancelar al final de mes, se asigna a una persona encargada de registrar, en una plantilla de papel (ver figura 1.3), el consumo de cada cliente registrado.

		COMITÉ DE AGUA POTABLE RURAL STA. JULIA -		
N°		hoja N° 1	25-dic	25-nov
SOC		MES: DICIEMBRE- 20	ESTADO	ESTADO
		NOMBRE	ACTUAL	ANTER.
95	1	PATRICIA ALVAREZ NAVIA	1146	1134
8	2	MARGARITA CERON FIGU	1798	1784
56	3	ESTEBAN BUENO CERON	1388	1376
63	4	FABIOLA BUENO CERON	1208	1198
55	5	ELIANA BUENO MOYA	1386	1381
11	6	LUZ MOYA GALLEGOS	1504	1491
10	7	JIMENA MUÑOZ MUÑO	2364	2347
54	8	LUZ BUENO MUÑOZ	1845	1840

(Figura 1.3) Planilla de consumo cliente, Fuente COMITÉ DE AGUA POTABLE SANTA JULIA

El trabajador asignado para esta tarea recorre a pie todas las casas a las cuales la empresa presta servicio, escribiendo, en una hoja de papel, la información del estado de consumo de agua entregada por el medidor correspondiente a cada vivienda.

Luego esta información recolectada es llevada a la oficina donde el encargado, Carlos Donoso, hace el cálculo a mano de cuánto es lo que debe cada cliente.
Se debe tener presente que para la elaboración de este cálculo se toma en consideración 6 puntos:

1. Lo primero es conocer el consumo de agua actual de la vivienda, si existe un sobreconsumo (esto ocurre cuando se exceden los 20m3) se le cobrará dependiendo en el rango de consumo del cargo variable en que se encuentre.

Por ejemplo: si un cliente tiene un consumo de 36m3, se calcula el valor de los primeros 20m3, los cuales corresponden al primer rango (1- 20 m3), con un valor de $400 por m3. A este total se le suma el segundo rango (21 – 35 m3) que correspondería a 15m3 de los 36 totales, con un valor de $550 por m3 y, por último, se le suma un tercer rango (36 – 50 m3) que correspondería al m3 restante (20+15+1 = 36), con un valor de $850 por m3.

2. Una vez calculado el valor del consumo de agua, se le suma el valor de cargo fijo (en estos momentos equivale a $3200).

3. Un tercer punto son los clientes beneficiados con el subsidio entregado por la municipalidad. A estos clientes se les disminuye el total del cargo variable (primer cálculo) dependiendo del porcentaje de beneficio que obtengan, y se les cobra sólo la mitad del cargo fijo.

4. Se programa una reunión mensual entre el encargado y el cliente para exponer información nueva acerca de la empresa o si es que el cliente tiene alguna duda. Si el cliente no asiste a la reunión se le asigna una multa de 5 mil pesos.

5. Si el cliente no paga el total de su cuenta esta se va acumulando como deuda, la cual es cobrada en la próxima boleta.

6. Santa Julia es una comunidad cerrada con portones para una mejor seguridad. Al cliente se le cobra una cuota que permite la mantención de estos portones.

Con esta nueva información es actualizada la planilla Excel “Consumo” (ver figura 1.4). A partir de esta planilla se escriben a mano las boletas (ver figura 1.5) con sus comprobantes para luego ser entregadas en las casas que correspondan.

COMITÉ DE AGUA POTABLE RURAL STA. JULIA - QUINTERO

N°		Hoja N° 1	25-dic	25-nov			Sob.Cons	Sob.Cons	Sob.Cons	CARGO	VALOR	MULTAS	Falta/Rest	SUBS.	DUEDA	FACT. DEL	BOLETA	TOTAL	DUPON/PAGO	
SOC		MES: DICIEMBRE- 20	ESTADO	ESTADO	CONS	VALOR	CARGO	21 a 35m3	36 a 50m3	51 a 65m3	FIJO	TOTAL		INSCORP	AGUA	PEND.	MES	FACTURA	Cancelado	fecha-pago	SALDO
		NOMBRE	ACTUAL	ANTER.	M3	M3 $	VARIAB.	x $ 530	x $ 650	x $ 1050	$	AGUA	Otros/Multas	ARRANQ	del MES	ANT	Beneficio	N°	$	cec-17	
45	1	PATRICIA ALVAREZ NAVA	1145	1134	12	400	4.800				3.290	8.090		5.000	0	0	13.090	9639			13.090
8	2	MARGARITA CERON FIGU	1795	1784	14	400	5.600				3.290	8.890			0	0	8.890	8700	8.890	734	0
54	3	ESTEBAN BUENO CERON	1088	1078	10	400	4.000				3.290	7.290			0	0	7.290	8701	7.290	787	0
43	4	FABIOLA BUENO CERON	1208	1193	15	400	6.000				3.290	9.290			0	0	9.290	8702	9.290	735	0
55	5	ELIANA BUENO MOYA	1085	1080	5	400	2.000				3.290	5.290	650		0	5.290	9.230	8703	9.230	770	0
11	6	LUIZ MOYA GALLEGOS	1504	1491	13	400	5.200				3.290	8.490			0	0	8.490	8704	8.490	782	0
10	7	JIMENA MUÑOZ MUÑOZ	1384	1347	37	400	6.800				3.290	10.090	650		0	9.210	19.950	8705	9.210	785	10.740
54	8	LUIZ BUENO MUÑOZ	1845	1840	5	400	2.000				3.290	5.290			0	0	5.290	8706			5.290

(Figura 1.4) Planilla Consumo, Fuente COMITÉ DE AGUA POTABLE SANTA JULIA

(Figura 1.5) Boleta de Pago, Fuente COMITÉ DE AGUA POTABLE SANTA JULIA

Al terminar el proceso de cálculo y actualización de la planilla, la hoja de papel que se utilizó para recolectar el estado de consumo de cada casa es desechada y se da por terminado el cálculo de lo que debe cancelar cada cliente.

Es de importancia mencionar que la empresa tiene copias de las planillas, pero estas no están actualizadas y se encuentran en el mismo computador que las originales. Al suceso de algún accidente que destruya el computador la empresa perderá todos sus documentos.

Otro punto a destacar es que la toma de consumo de agua se realiza todos los meses, los clientes son 99 en este momento, y el cálculo del total a pagar más la escritura de las boletas con sus comprobantes tiene una duración de aproximadamente 1 día.

Por último, mencionar que algunas de las planillas (resumen de boleta) son elaboradas a partir de otras planillas, sin necesidad de ingresar datos vía teclado.

1.3. PROBLEMAS DETECTADOS

Debido a que la empresa realiza la mayor parte de sus operaciones de forma manual, se logró detectar los siguientes problemas:

- No existe un identificador clave único óptimo para cada cliente, sino que se utiliza un número correlativo o el nombre del cliente. Esto causa lentitud al momento de hacer las boletas y comprobantes para cada uno de ellos y puede producir redundancia e inconsistencia en los datos almacenados.

- No existe un respaldo de la información de los clientes ni de los comprobantes de boletas. La empresa tiene copias de cada planilla de Excel, pero estas no están actualizadas y se encuentran almacenadas en la misma computadora que las originales. Si el computador se extravía o es destruido por algún accidente, no existe una forma de recuperación de sus documentos.

- Todo lo que tiene relación con el cálculo de consumo de agua por cada cliente, y otros cálculos correspondientes a los ingresos y egresos de la empresa, son llevados a cabo de manera manual, para posteriormente ser ingresados en sus propias planillas Excel. Esto puede generar posibles errores el realizar los cálculos, lo cuales afectarán el pago final de los clientes, también existe un excesivo gasto de tiempo en realizarlos.

- Cada mes se emiten las boletas con sus respectivos comprobantes de pagos de forma manual por cada cliente, los cuales ascienden a 100 y van constantemente incrementando. Este llenado de boleta y comprobante, junto con la repartición de la primera por casa, demora alrededor de 1 día.

- Existen múltiples planillas Excel diferentes (usuarios, subsidio, entrega de boleta, cupones, entre otras), las cuales no tienen ningún identificador único que las diferencie y un orden, lo cual hace confuso los cálculos a realizar y sus registros.

1.4. DESCRIPCIÓN DEL SISTEMA PROPUESTO

1.4.1. Objetivos del Sistema Propuesto

Se pretende desarrollar un sistema que permita facilitar la labor de los usuarios del "Comité de agua potable rural Santa Julia", automatizando todo lo referente al registro de clientes (agregar, modificar, eliminar) con su respectivo consumo de agua potable y cálculos para la impresión de boletas y comprobantes de pago.

El sistema a desarrollar tiene los siguientes objetivos específicos:

- Administrar de una manera eficiente a los clientes, otorgándoles una clave única en su registro.
- Realizar un respaldo óptimo de cada mes y otro respaldo que sea anual.
- Realizar todo tipo de cálculo automáticamente, el usuario intervendrá solo si se necesita el ingreso de algún dato que permitirá estos cálculos.
- Generar las boletas con sus respectivos comprobantes de pagos de manera automática.
- Generar archivos, correspondientes a las planillas Excel actuales de la empresa, con su respectiva clave primaria.

Beneficios del Sistema Propuesto:

- La consulta de algún cliente en particular, para poder operar sobre sus datos, será más fácil y rápido de encontrar.
- Al ocurrir algún accidente en el que se vean destruidos o afectados de alguna manera los documentos, la empresa constará de un respaldo actualizado y óptimo de estos archivos para su recuperación.
- El tiempo dedicado en desarrollar todos los cálculos necesarios para la empresa se verá disminuido.
- Al ser automática la realización de los cálculos, los posibles errores serán eliminados.
- El tiempo y esfuerzo en generar las boletas con sus respectivos comprobantes se verá reducido considerablemente.
- El orden en los archivos creados, facilitará la manipulación de sus datos.

1.4.2. Descripción General de la Solución Propuesta

El sistema propuesto será capaz de facilitar la labor de los usuarios y automatizar el registro de las actividades realizadas por la empresa. Desde el ingreso, eliminación y modificación de sus clientes, la actualización del consumo de cada uno de estos, el resumen e impresión de boletas, con sus respectivos cálculos.

1.4.3. Estructura funcional del sistema

El sistema contendrá las siguientes funcionalidades:

- Mantenedor de usuario: permitirá realizar las funciones básicas para el registro de los usuarios del sistema.
- Mantenedor de clientes: permitirá realizar las funciones básicas para el registro de este, su ingreso, consulta, modificación o eliminación.
- Gestor de boletas: permitirá con la ayuda del consumo, generar un documento detallado de lo que deberá cancelar cada cliente, permitiendo la impresión de la boleta y el comprobante.
- Gestor de resumen de boleta: se genera un informe del detalle de las boletas a partir del archivo de consumo.
- Gestor registro de consumo: calcular y generar de manera automática, solo con el ingreso por vía teclado del consumo actual, el registro por cliente de su consumo total.

- Gestor de pago: permitirá generar el total a pagar del consumo de agua del cliente y el ingreso del pago por parte de este.
- Mantenedor medidor: permite ingresar un nuevo medidor de un cliente antiguo, modificar un medidor que se ha cambiado o eliminarlo si el cliente ya no está asociado a él.
- Mantenedor de valor-cargo: permite modificar el cargo fijo y el cargo variable, así como el ingreso de un nuevo rango en este último o la eliminación de uno ya existente.
- Mantenedor de asistencia: permite registrar la asistencia de los clientes a las reuniones mensuales, lo que permitirá generar a partir de esta las multas.
- Generar consultas e informes: permite consultar distintos aspectos de los clientes, mostrando por pantalla un listado correspondiente a la consulta. Algunos de los informes obtenidos son:
    - Listado de datos de los clientes.
    - Listado de clientes mostrando subsidio, deuda y multas.
    - Listado de medidor.
    - Listado de asistencia a reunión.
    - Listado de consumo.
    - Listado resumen de boleta.
    - Listado de usuarios.
    - Listado de pago.

1.4.4. Información que se manejará

Entradas

Datos de clientes:

- Corresponden a todos los datos importantes asociados a los clientes.

Datos de usuarios:

- Corresponden a los datos relacionados con los usuarios del sistema y al nivel de acceso que estos tienen.

Datos de consumo de agua:

- Corresponden a todos los datos asociados al consumo de agua de los clientes y el cálculo total a pagar por mes.

Datos de pago:

- Corresponde al pago del consumo de agua por parte del cliente.

Datos de subsidio:

- Datos correspondientes a los clientes que cuentan con subsidio. Es importante destacar que estos datos son entregados por la municipalidad cada vez que ocurre un cambio.

Datos de ajuste de cargo:

- Corresponde a la modificación del valor del cargo variable y, aunque es poco probable, del cargo fijo.

Datos de asistencia:

- Datos sobre la asistencia de los clientes a las reuniones.

Datos del medidor:

- Datos asociados a la modificación o ingreso de un medidor.

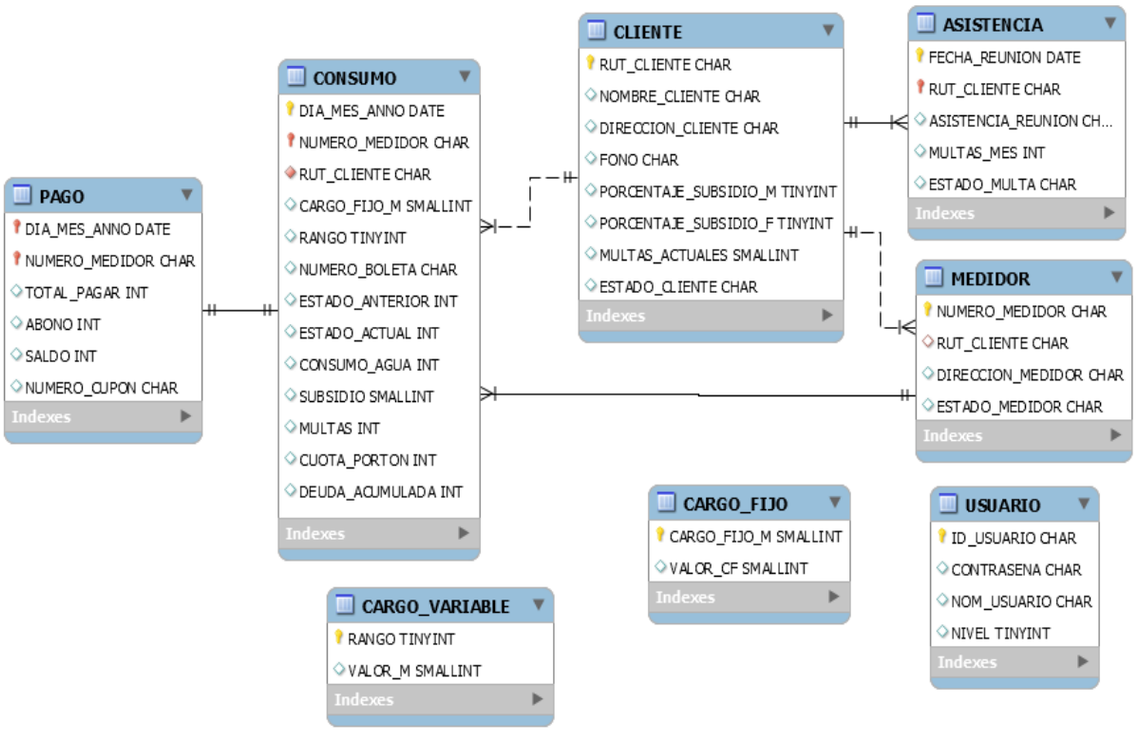
Salidas

- Emisión de boletas (pantalla o impresora).
- Emisión comprobante de boleta (pantalla o impresora).
- Listado de datos de los clientes.
- Listado de clientes mostrando subsidio, deuda y multas.
- Listado de consumo.
- Listado resumen de boleta.
- Listado de asistencia a reunión.
- Listado de pago.
- Listado de medidor.
- Listado de usuarios.

Entidades de información

- ***CLIENTE***, contiene datos de los clientes que están ingresados en el sistema.
- ***MEDIDOR***, contiene datos correspondientes a los medidores de cada cliente.
- ***CARGO_FIJO***, contiene datos del valor cargo fijo.
- ***CARGO_VARIABLE***, contiene datos de cargo variable que corresponde a los valores de metros cúbicos.
- ***CONSUMO***, contiene los datos correspondientes al mes de consumo de agua potable.
- ***PAGO,*** contiene los datos relacionados al total a pagar por el consumo de agua y del pago efectuado por el cliente.
- ***ASISTENCIA***, contiene datos de los clientes que asisten y no asisten a las reuniones del comité de agua potable.
- ***USUARIO,*** contiene los datos correspondientes a los usuarios del sistema.

Modelo de datos



(Figura 1.6) Modelo relacional del Sistema

1.4.5. Condicionantes de diseño

Se decidió utilizar Visual Studio 2015 (lenguaje visual studio), ya que es el lenguaje que mejor manejamos. Es importante mencionar que la empresa se hará cargo de los pagos necesarios para su utilización.

El programa podrá manejar dos tipos de usuarios, administrador o recepcionista. Cada usuario tendrá un nivel de acceso al software para evitar el uso incorrecto de información o pérdida de este por completo.

CAPÍTULO II.

MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

CAPÍTULO II. MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE LA BASE DE DATOS

2.1. Medio ambiente computacional

A continuación, se describen las principales características del recurso computacional que posee la empresa, en el cual funcionará el sistema propuesto.

2.1.1. Configuración del sistema

Equipos en los cuales se desarrolló el sistema

El sistema informático propuesto anteriormente se desarrollará en dos equipos:

1. Notebook HP.
    - Sistema operativo: Windows 8.1 Pro (64 bits)
    - Procesador: AMD E1-2100 APU con Radeon (TM) HD Graphics 1,00 GHz
    - RAM: 4 GB en memoria
2. Notebook Acer, Aspire V3
    - Sistema operativo: Windows 8.1 Pro (64 bits)
    - Procesador: Intel(R) Core(TM) i3-2328M CPU 2,20 GHz
    - RAM: 6 GB en memoria

Equipos de la empresa.

Notebook LENOVO

- Procesador: INTEL Celeron 1,6 GHz
- RAM: 4 GB en memoria
- Disco duro: 500 GB
- Pantalla: 14" IPS
- Batería: 4 celdas/ 32 WH
- Teclado tipo isla
- Puertos de entrada y salida: 1 USB 2.0, 2 USB 3.0, 1 HDMI, 1 VGA, 1 conector para Audífonos/ entrada de micrófono combinado.

Impresora Epson lx-350

- Método de impresión: Matriz de puntos de impacto en serie, 9 agujas.

Disco de almacenamiento externo TOSHIBA

- Capacidad: 2TB
- Interfaz: USB 3.0
- Velocidad de transferencia: hasta 5 Gb/s
- Velocidad: 5400 RPM
- Tiempo de búsqueda medio: 12ms

2.1.2. Software utilizado

Windows 10 Home Single.

Debido a que fue el más fácil de usar desde un principio, y por su amigable interfaz, Windows es el sistema operativo más utilizado a nivel mundial en la actualidad.

Windows 10 Home es una de las últimas versiones y es la edición estándar de Windows 10 siendo una variante de éste el Home Single.

Visual Studio 2015

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows.

Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco.

El lenguaje de programación Visual Basic utiliza una interfaz visual, es decir permite programar en un entorno gráfico y realizar un gran número de tareas sin escribir código, simplemente realizando operaciones con el ratón sobre la pantalla de la computadora. Es este el elegido para la construcción del sistema propuesto.

MariaDB

Una base de datos es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible, para que su contenido pueda ser tratado y analizado de manera rápida y sencilla.

MariaDB es un sistema de gestión de bases de datos derivado de MySQL con más funcionalidades y mejor rendimiento que este último, cuenta con licencia GPL (General Public License) y al ser creado a partir de MySQL tiene una alta compatibilidad con este.

Tipos de datos

Los tipos de datos numéricos a utilizar en el sistema y soportados por MariaDB son los siguientes:

- TINYINT - Este tipo de dato representa enteros pequeños que caen dentro del rango firmado -128 y 127, y el rango sin signo de 0 a 255.

- BOOLEAN - Este tipo de datos asocia un valor 0 con “falso”, y un valor de 1 con “verdadero”.

- SMALLINT - Este tipo de dato representa números enteros dentro de la gama firmado -32768 a 32768 y el rango sin signo de 0 a 65535.

- INT (INTEGER) - Este tipo de datos representa un número entero de tamaño normal. Cuando marcado como sin firmar, el alcance se extiende por 0 a 4294967295. Cuando firmado (the default setting), el alcance se extiende por -2147483648 a 2147483647. Cuando una columna se establece en ZEROFILL (an unsigned state), todos sus valores están precedidas por ceros a cabo M dígitos en el valor INT.

- DATE - Este tipo de dato representa un intervalo de fechas de “01/01/1000” a “9999-12-31”, y utiliza el formato de fecha “AAAA-MM-DD”.

- CHAR - Este tipo de datos representa una cadena derecha acolchado, de longitud fija que contiene espacios de longitud especificada. M representa longitud de la columna de caracteres en un rango de 0 a 255, el valor predeterminado es 1.

2.2. Descripción de tablas

A continuación, se describirán todas las tablas utilizadas en el SISTEMA INFORMÁTICO ADMINISTRADOR DE COMITÉ DE AGUA POTABLE SANTA JULIA.

Cliente

Nombre Lógico : CLIENTE

Descripción : En la siguiente tabla se almacenan los datos relacionados a los clientes de la empresa.

Clave Primaria : RUT_CLIENTE

Nombre Campo	Tipo	Longitud	Descripción
RUT_CLIENTE	CHAR	9	Rut del cliente formato 99999999X
NOMBRE_CLIENTE	CHAR	30	Nombre del cliente
DIRECCION_CLIENTE	CHAR	40	Dirección del cliente
FONO	CHAR	13	Teléfono del cliente formato +99999999999
PORCENTAJE_SUBSIDIO_M	TINYINT	3	Porcentaje del subsidio total del cliente formato 999
PORCENTAJE_SUBSIDIO_CF	TINYINT	3	Porcentaje del subsidio de cargo fijo del cliente formato 999
MULTAS_ACTUALES	SMALLINT	5	Multas del cliente, formato 99999

Medidor

Nombre Lógico : MEDIDOR

Descripción : En esta tabla se almacenan los datos de cada medidor de agua de la empresa.

Clave Primaria : NUMERO_MEDIDOR

Clave Foránea : RUT_CLIENTE (referencia a tabla CLIENTE)

Nombre Campo	**Tipo**	**Longitud**	**Descripción**
NUMERO_MEDIDOR	CHAR	20	Número del medidor
RUT_CLIENTE	CHAR	9	Rut del cliente formato 99999999X
DIRECCION_MEDIDOR	CHAR	40	Dirección en la cual está ubicado el medidor
ESTADO_MEDIDOR	CHAR	1	Asistencia del medidor: A ->> activo; I ->> inactivo

Asistencia

Nombre Lógico : ASISTENCIA

Descripción : En esta tabla se almacenan los datos de la asistencia de los clientes a las reuniones de la empresa.

Clave Primaria : FECHA_REUNION + RUT_CLIENTE

Clave Foránea : RUT_CLIENTE (referencia a tabla CLIENTE)

Nombre Campo	**Tipo**	**Longitud**	**Descripción**
FECHA_REUNION	DATE	10	Fecha de la reunión, formato AAAA-MM-DD
RUT_CLIENTE	CHAR	9	Rut del cliente formato 99999999X
ASISTENCIA_REUNION	CHAR	1	Asistencia de la reunión: S ->> sí; N ->> no
MULTA_MES	INT	6	Multa de por inasistencia
ESTADO_MULTA	CHAR	1	Estado de la multa: C->>cancelado; D->> deudor

Consumo

Nombre Lógico : CONSUMO

Descripción : En esta tabla se almacenan los datos relacionados al consumo mensual de agua potable por medidor.

Clave Primaria : DIA_MES_ANNO + NUMERO_MEDIDOR

Clave foránea : NUMERO_MEDIDOR (referencia tabla MEDIDOR)

RUT_CLIENTE (referencia tabla CLIENTE)

Nombre Campo	**Tipo**	**Longitud**	**Descripción**
DIA_MES_ANNO	DATE	10	Fecha de emisión del consumo, formato AAAA-MM-DD
NUMERO_MEDIDOR	CHAR	20	Número del medidor.
RUT_CLIENTE	CHAR	9	Rut del cliente formato 99999999X
CARGO_FIJO_M	SMALLINT	1	Número del cargo fijo, formato 9
RANGO	TINYINT	3	Rango del consumo de agua, formato 999
NUMERO_BOLETA	CHAR	10	Número que identifica la boleta a entregar, formato 99999999999
ESTADO_ANTERIOR	INT	10	Estado del mes anterior del consumo de agua
ESTADO_ACTUAL	INT	10	Estado del mes actual del consumo de agua en metros cúbicos.
CONSUMO_AGUA	INT	10	Consumo de agua del mes en metros cúbicos.
SUBSIDIO	SMALLINT	4	Valor total del subsidio del cliente, formato 9999
MULTAS	INT	6	Multa por faltar a la reunión, formato 999999
CUOTA_PORTON	INT	6	Cuota mantención portón, formato 999999
DEUDA_ACUMULADA	INT	10	Deuda de consumo de agua, formato 999999

Pago

Nombre Lógico : PAGO

Descripción : En la siguiente tabla se almacena el total a pagar por el cliente de la empresa.

Clave Primaria : DIA_MES_ANNO + NUMERO_MEDIDOR

Clave Foránea : DIA_MES_ANNO + NUMERO_MEDIDOR (referencia tabla CONSUMO)

Nombre Campo	**Tipo**	**Longitud**	**Descripción**
DIA_MES_ANNO	DATE	10	Fecha de emisión del consumo, formato AAAA-MM-DD
NUMERO_MEDIDOR	CHAR	20	Número del medidor
TOTAL_PAGAR	INT	6	Total de la cuenta
ABONO	INT	6	Abono a la deuda
SALDO	INT	6	Saldo de la cuenta
NUMERO_CUPON	CHAR	10	Número del cupón de pago

Usuario

Nombre Lógico : USUARIO

Descripción : En la siguiente tabla se almacenan los usuarios del sistema

Clave Primaria : ID_USUARIO

Nombre Campo	**Tipo**	**Longitud**	**Descripción**
ID_USUARIO	CHAR	8	Identificador de usuario
CONTRASEÑA	CHAR	10	Permite validar el login del sistema
NOM_USUARIO	CHAR	30	Nombre del usuario que inicio el sistema
NIVEL	TIYINT	1	Permite saber el nivel de acceso al sistema. nivel 1: administrador nivel 2: presidente nivel 3: secretaria

Cargo Fijo

Nombre Lógico : CARGO_FIJO

Descripción : en la siguiente tabla se almacena el cargo fijo de cobro de agua

Clave Primaria : CARGO_FIJO_M

Nombre Campo	Tipo	Longitud	Descripción
CARGO_FIJO_M	SMALLINT	1	Número del cargo fijo, formato 9
VALOR_CF	SMALLINT	4	Valor del cargo fijo, formato 9999

Cargo Variable

Nombre Lógico : CARGO_VARIABLE

Descripción : En la siguiente tabla se almacena el valor de los distintos rangos respecto al consumo de agua potable.

Clave Primaria : RANGO

Nombre Campo	Tipo	Longitud	Descripción
RANGO	TINYINT	3	Rango del consumo de agua, formato 999
VALOR_M	SMALLINT	4	Valor del consumo de agua, f ormato 9999

CAPÍTULO III. DESCRIPCIÓN DE PROGRAMAS

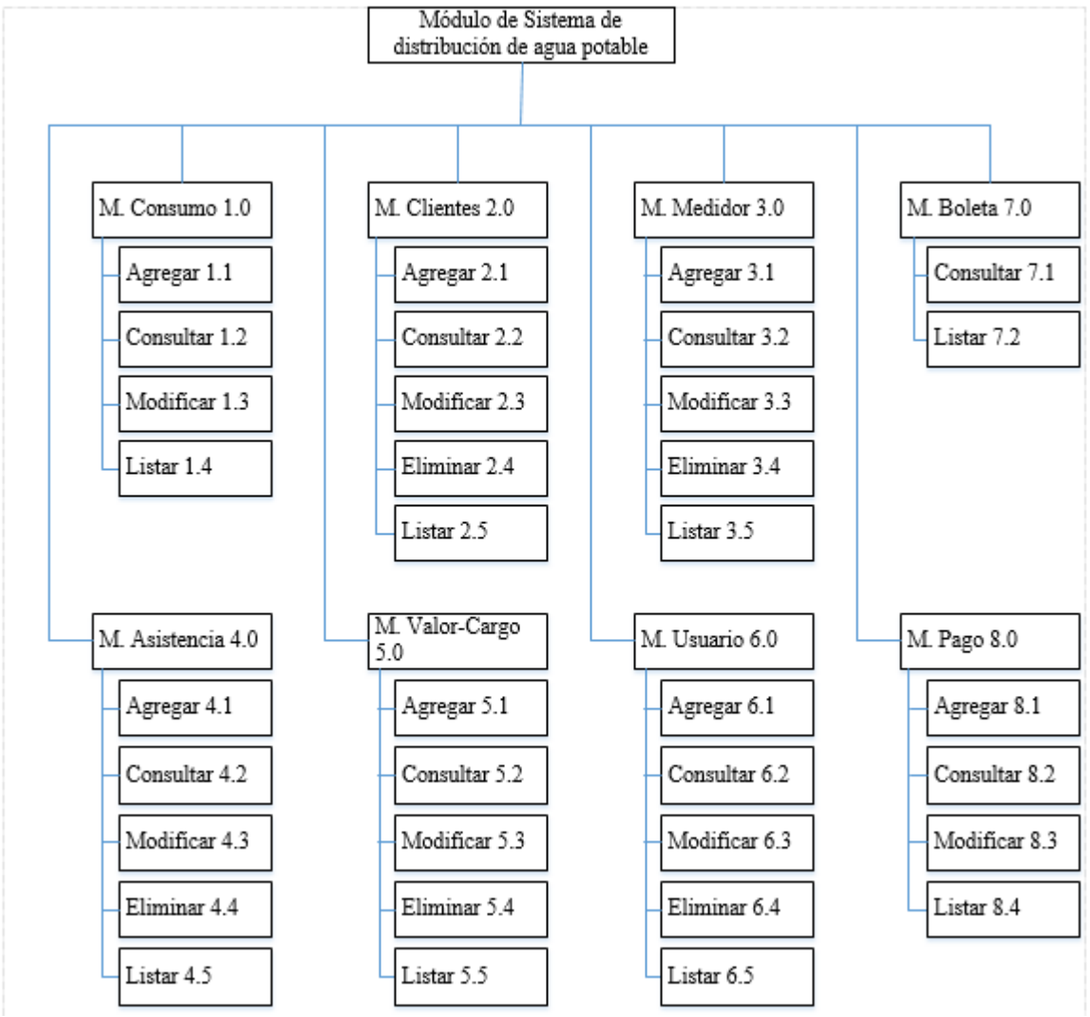
CAPÍTULO III. DESCRIPCIÓN DE PROGRAMAS

Una de las principales características para lograr crear un sistema informático eficiente es tener un total conocimiento y entendimiento de todas las funcionalidades que componen el sistema, considerando los requerimientos y opiniones del cliente durante el desarrollo de éste.

Otra característica al desarrollar un sistema informático es tener una interfaz amigable, sencilla de entender y utilizar por parte del usuario.

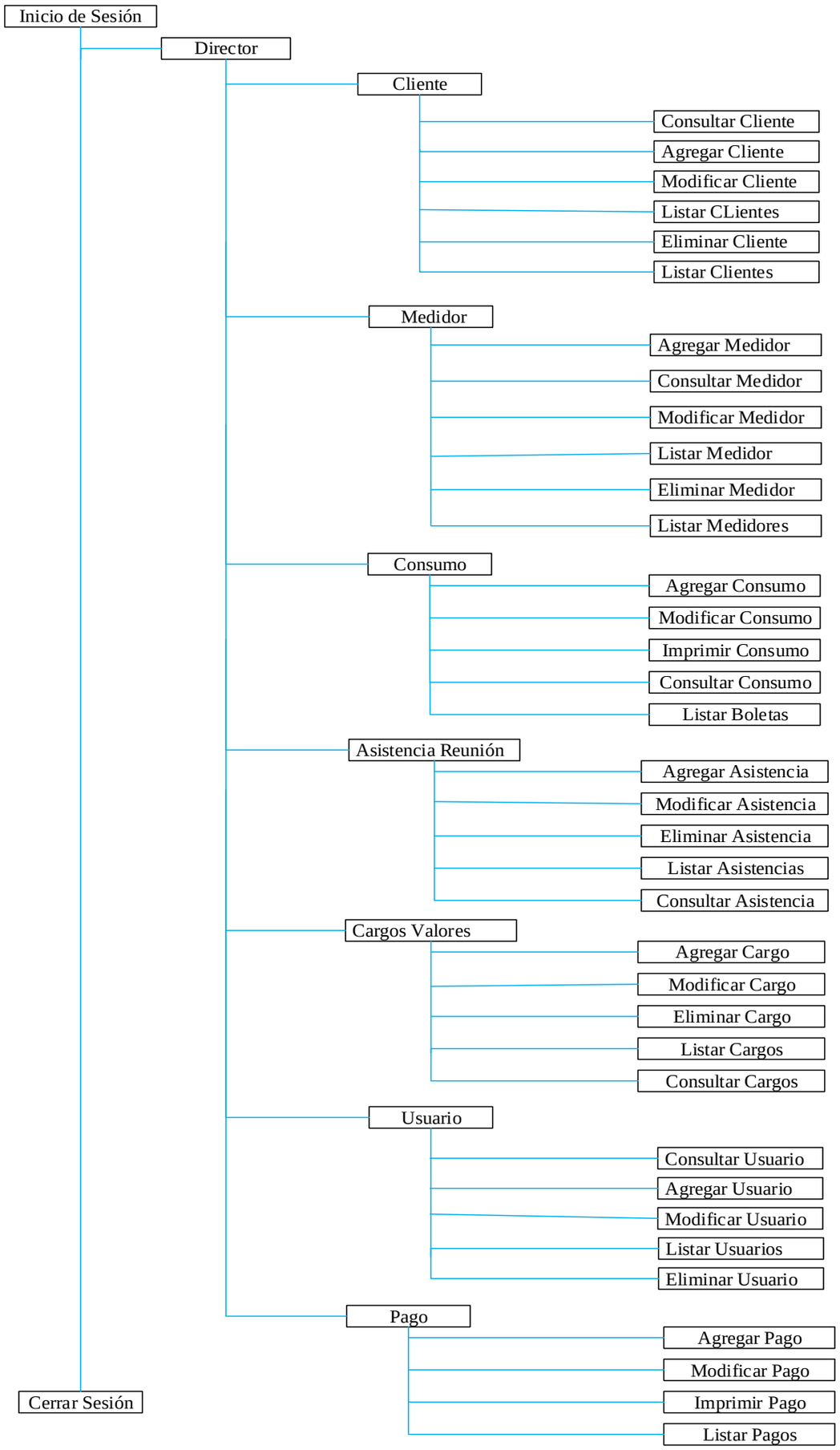
3.1. Diagrama Modular

A continuación se muestra el diagrama modular que presenta el sistema a desarrollar.



3.2. Diagrama de menús

A continuación se muestra el diagrama de menú que presenta el sistema a desarrollar.



3.3. Tabla de Programas Desarrollados

A continuación se presentan los programas que contiene el software desarrollado.

Nombre	Objetivo
Inicio de sesión (*)	Permitir o denegar el ingreso del usuario al sistema, identificando los permisos que este tiene según el tipo de usuario que tenga.
Menú Principal (*)	Permitir visualizar y accesar a las distintas funcionalidades que conforman el sistema.
Mantenedor de Cliente (*)	Permitir al usuario realizar las funciones CRUD sobre los clientes que son parte de la empresa.
Mantenedor de Medidor (*)	Permitir realizar las funciones CRUD sobre los medidores relacionados a los clientes que son parte de la empresa.
Gestor de Boleta (*)	Registrar el consumo de agua del cliente, realizar el cálculo del total a pagar por cliente e imprimir la boleta correspondiente. Además, permite realizar diferentes consultas respecto a los consumos.
Mantenedor de Reuniones (*)	Permitir al usuario registrar la asistencia a las reuniones calendarizadas por el comité y llevar un registro de éstas.
Mantenedor de Cargos (*)	Permitir realizar las funciones CRUD sobre los cargos variables del comité de agua potable y modificar el cargo fijo.
Mantenedor de Usuario (*)	Permitir realizar las funciones CRUD a los usuarios.
Gestor de Pago (*)	Registrar el pago de consumo de cada cliente por mes e imprimir el comprobante de pago.Además, permite realizar diferentes consultas respecto a los pagos.

(*) Programas a describir

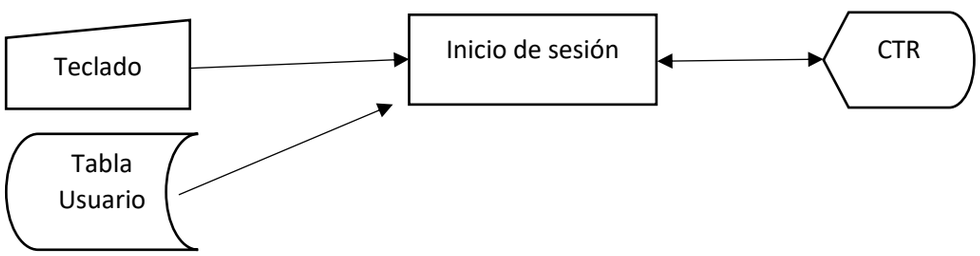
3.4. Descripción Detallada del Programa

3.4.1. Nombre Programa: Inicio de sesión

Objetivo: Permite al usuario ingresar al sistema para acceder a las distintas funcionalidades que componen a este.

Figura 1.1 (Inicio de sesión)

Diagrama de Bloques:



Reglas de Negocio:

Se presenta una pantalla de Inicio de Sesión en la cual el usuario debe ingresar su usuario y contraseña, para ingresar al sistema debe presionar el botón aceptar. En caso de que el usuario y/o contraseña sean incorrectos se desplegará un mensaje alertando el error, si el usuario y contraseña son correctas el sistema permitirá el acceso al menú.

Código fuente: Anexo página 57.

3.4.2. Nombre Programa: Menú Principal.

Objetivo: Presentar al usuario las diferentes aplicaciones que contiene el sistema.

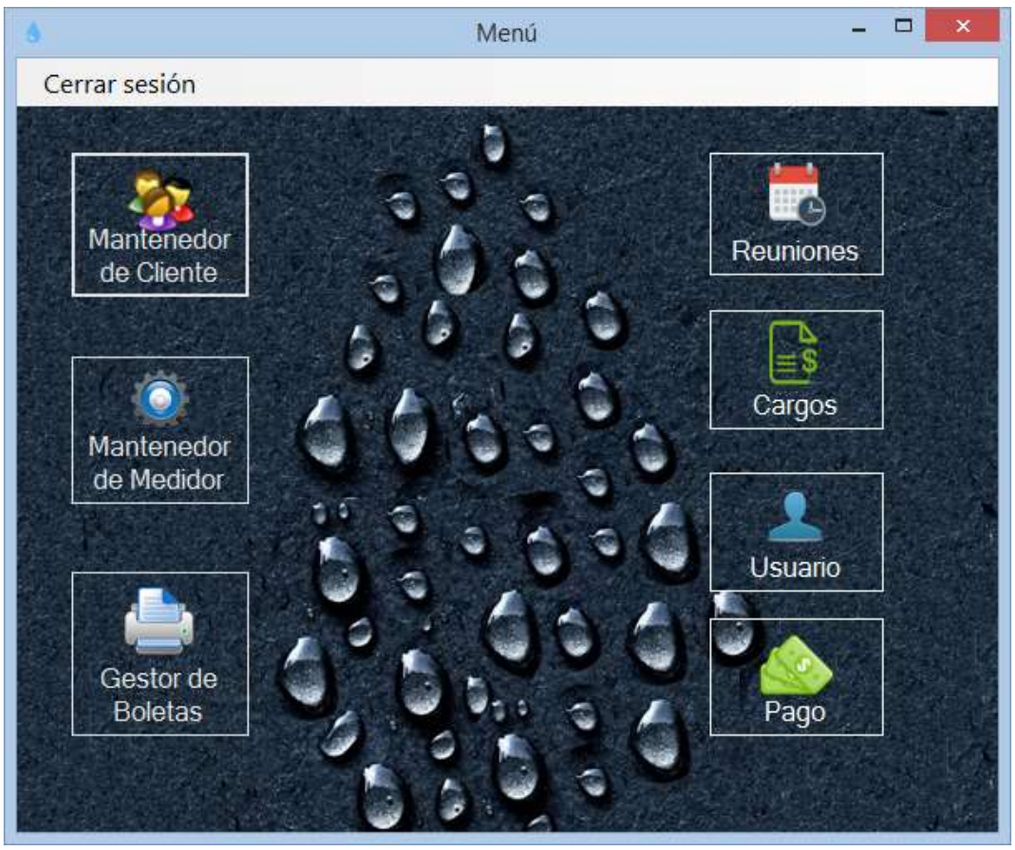
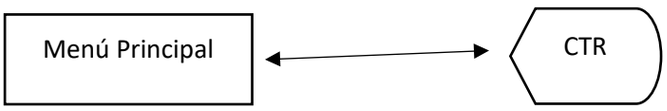


Figura 1.2 (Menú Principal)

Diagrama de Bloques:



Reglas de negocio:

Se presenta en la pantalla el menú principal, donde el usuario podrá realizar las siguientes funciones:

- **Ingresar a mantenedor de cliente:** el usuario debe presionar el botón "Mantenedor de Cliente" si desea ingresar, eliminar, modificar o consultar por algún cliente.
- **Ingresar a mantenedor de medidor:** el usuario debe presionar el botón "Mantenedor de Medidor" si desea ingresar, eliminar, modificar o consultar por algún medidor.
- **Ingresar a gestor de boletas:** el usuario debe presionar el botón "Gestor de Boletas" si desea ingresar, modificar o consultar un consumo de agua actual o si desea imprimir boletas.

- **Ingresar a reuniones:** el usuario debe presionar el botón “Reuniones” si desea ingresar, modificar o consultar por la asistencia a reunión de un cliente.
- **Ingresar a cargos:** el usuario debe presionar el botón “Cargos” si desea ingresar, eliminar, modificar o consultar por algún cargo variable o fijo.
- **Ingresar a usuario:** el usuario debe presionar el botón “Usuario” si desea ingresar, eliminar, modificar o consultar algún usuario.
- **Ingresar a pago:** el usuario debe presionar el botón “Pago” si desea ingresar un pago e imprimir su comprobante o consultar por algún pago ya efectuado.

Código fuente: Anexo página 58.

3.4.3. Nombre del programa: Mantenedor de Cliente.

Objetivo: el usuario tendrá una vista de los clientes registrados actualmente en la empresa con sus respectivos datos.

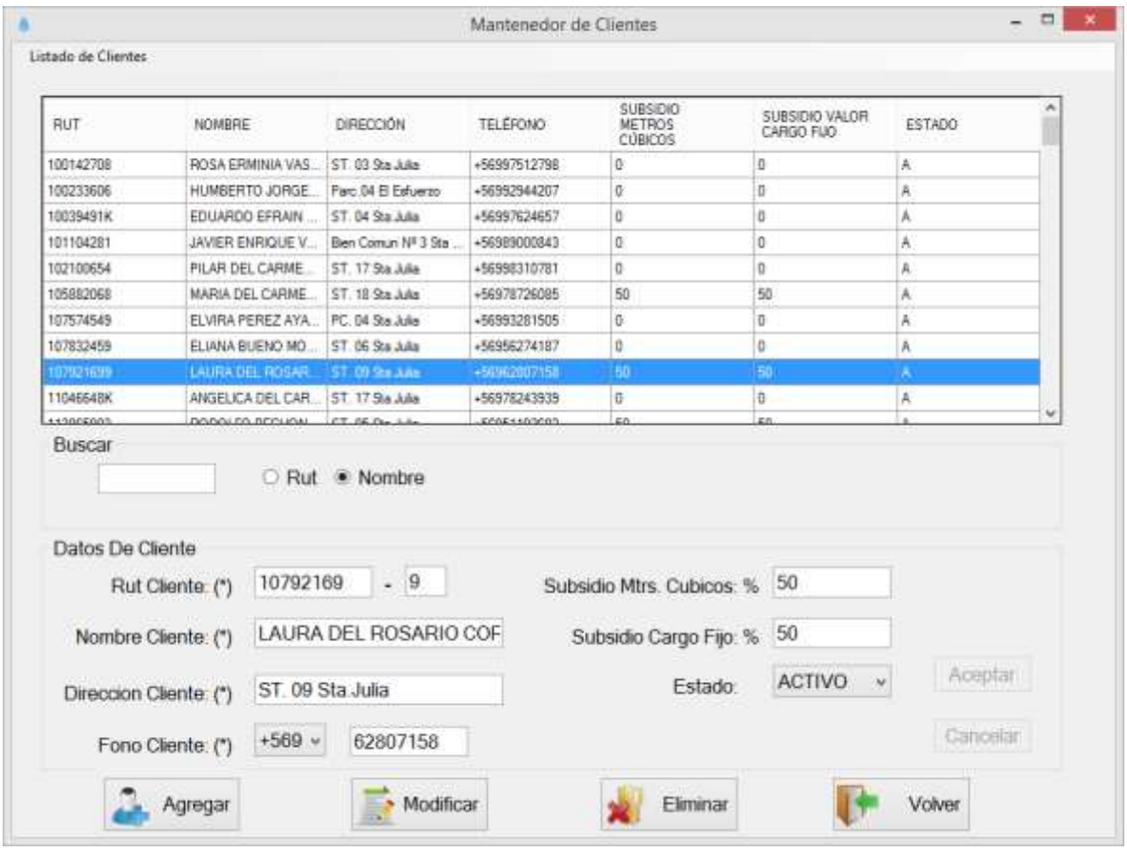
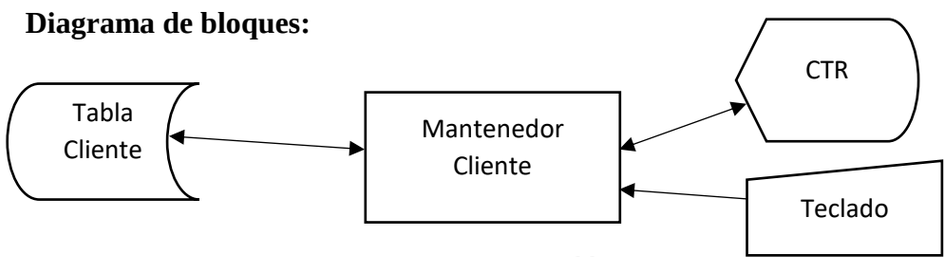


Figura (1.3 Mantenedor Cliente)

Diagrama de bloques:



Reglas de negocio:

Se presenta en la pantalla el mantenedor de clientes, donde el usuario podrá realizar las siguientes funciones:

- **Agregar Cliente:** el usuario debe presionar el botón Agregar, posteriormente se le habilitan las casillas, el botón aceptar y cancelar. Debe ingresar el Rut del cliente que desea registrar, en caso este sea erróneo el software alertará esta falla. Una vez ingresado todos los datos pedidos por obligación se procede a presionar el botón Aceptar. En caso de que el cliente exista se mostrará un mensaje por pantalla sobre esta anomalía, de caso contrario el mensaje será que el cliente fue registrado exitosamente.

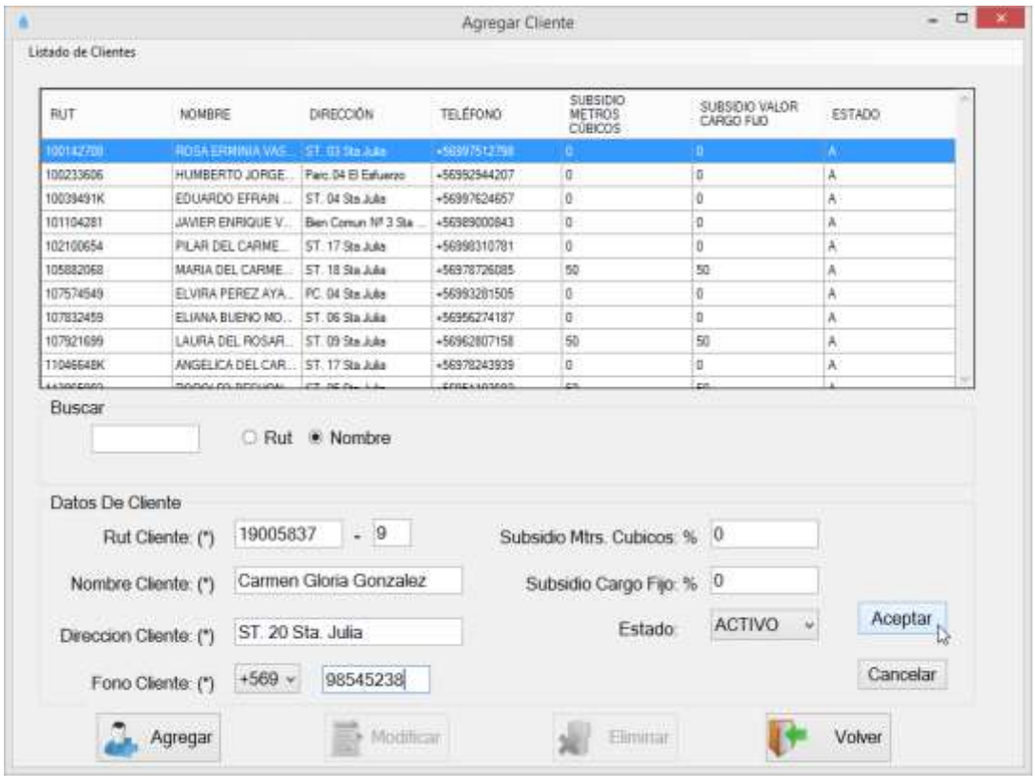


Figura (1.4 Agregar Cliente)

- **Modificar Cliente:** si el usuario desea modificar datos de un cliente debe presionar el botón Modificar, esta función permitirá modificar los campos nombre, dirección, fono, subsidio y estado.

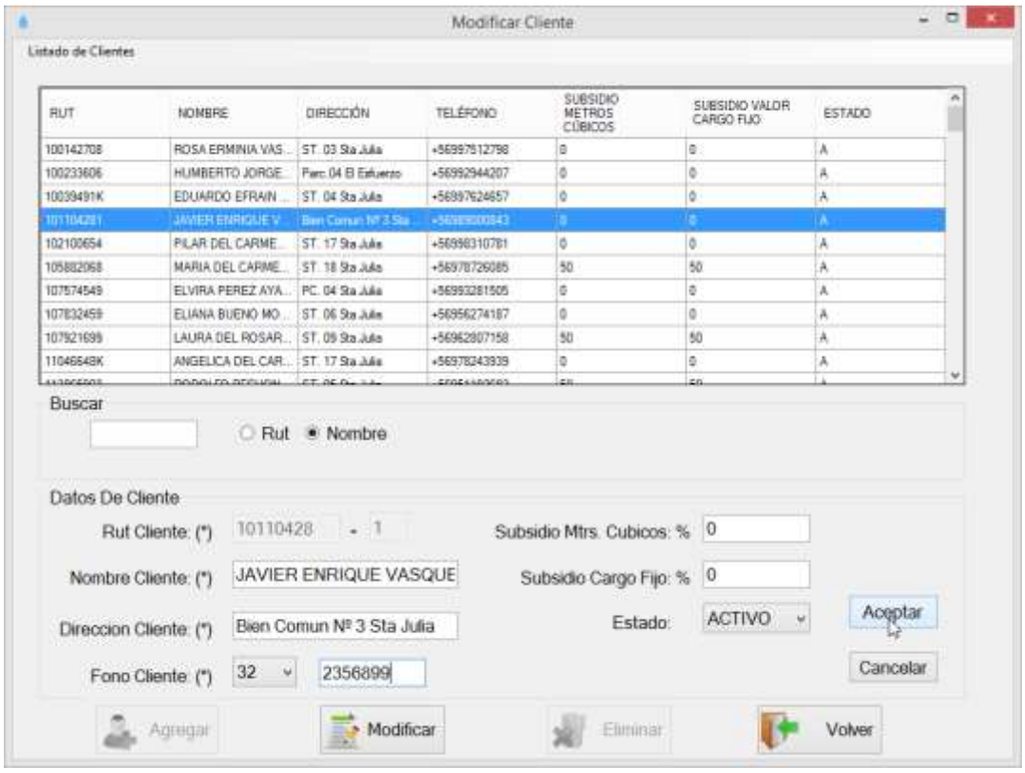


Figura (1.5 Modificar Cliente)

- **Eliminar Cliente:** permite al usuario eliminar un cliente, en caso de que este tenga medidor y consumos relacionados solo permitirá una eliminación lógica, la cual consiste en cambiar el estado a inactivo.

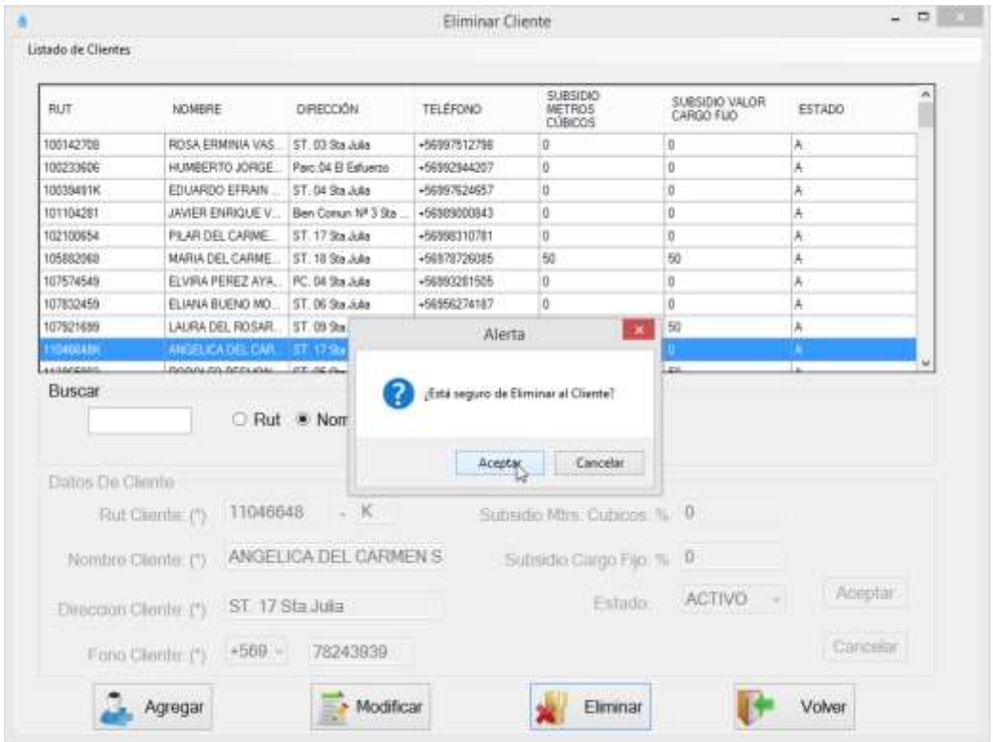


Figura (1.6 Eliminar Cliente)

- **Consultar Cliente:** el usuario podrá realizar consulta por cliente con subsidio, con multas, activo o inactivo.

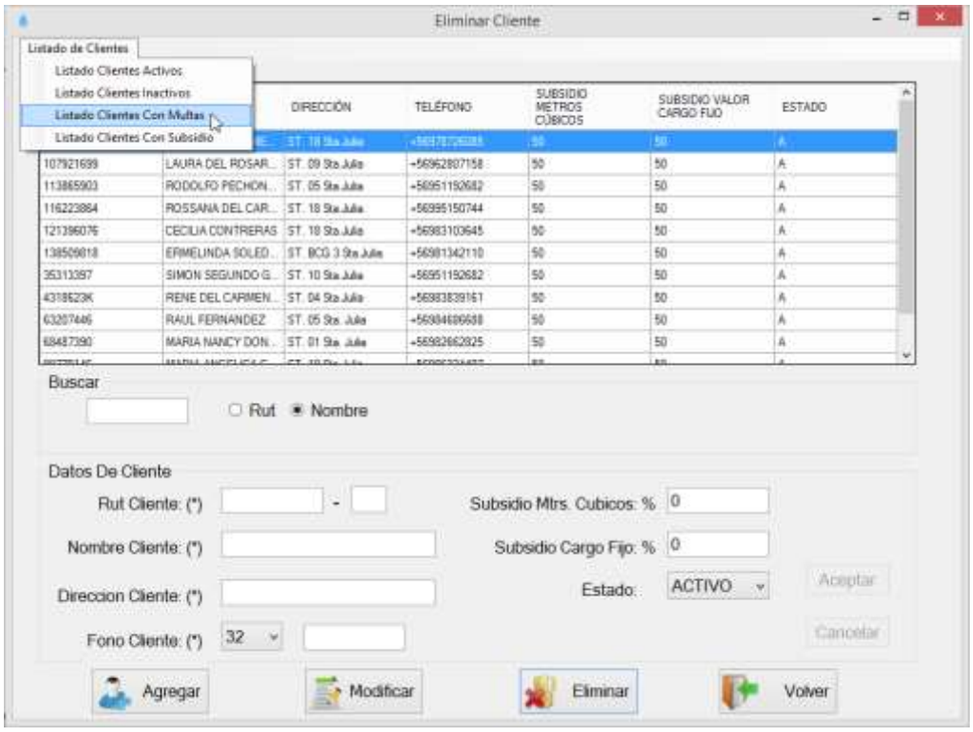


Figura (1.7 Consultar Cliente)

Código Fuente: Anexo página 59.

3.4.4. Nombre Programa: Mantenedor Medidor

Objetivo: se presenta un listado con todos los medidores registrados por cada cliente que integra la empresa.

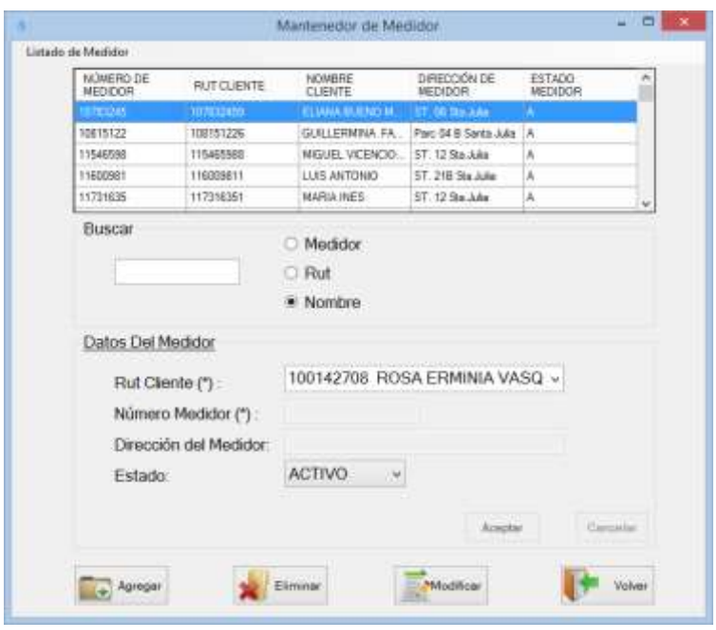
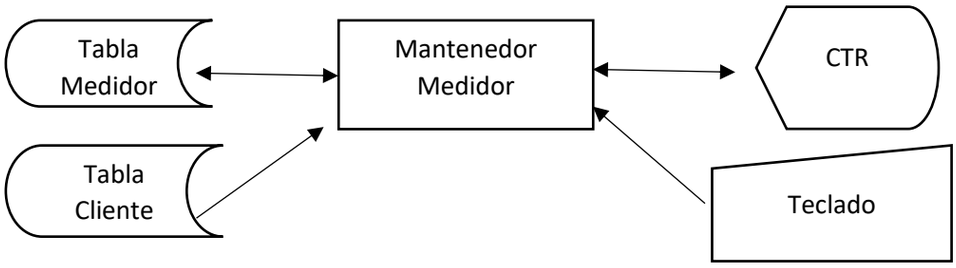


Figura (1.8 Mantenedor Medidor)

Diagrama de Bloques:



Reglas de Negocio:

Se presenta en la pantalla el mantenedor de medidor en el cual el usuario podrá realizar las siguientes acciones:

- **Agregar Medidor:** el usuario podrá agregar medidores a un cliente, este puede tener de 1 a N medidores, diferenciándose por su número de identificación. El usuario debe presionar el botón agregar, este le habilitará la casilla de Rut cliente, número de medidor, dirección y estado. Si se ingresa un Rut inválido el programa alertará de esta anomalía y no permitirá continuar, en caso contrario, se deben ingresar los campos obligatorios. En caso de que el medidor ya se encuentre registrado en el sistema se desplegará un mensaje advirtiendo que ya fue ingresado, sino el mensaje mostrará que el medidor fue agregado exitosamente.

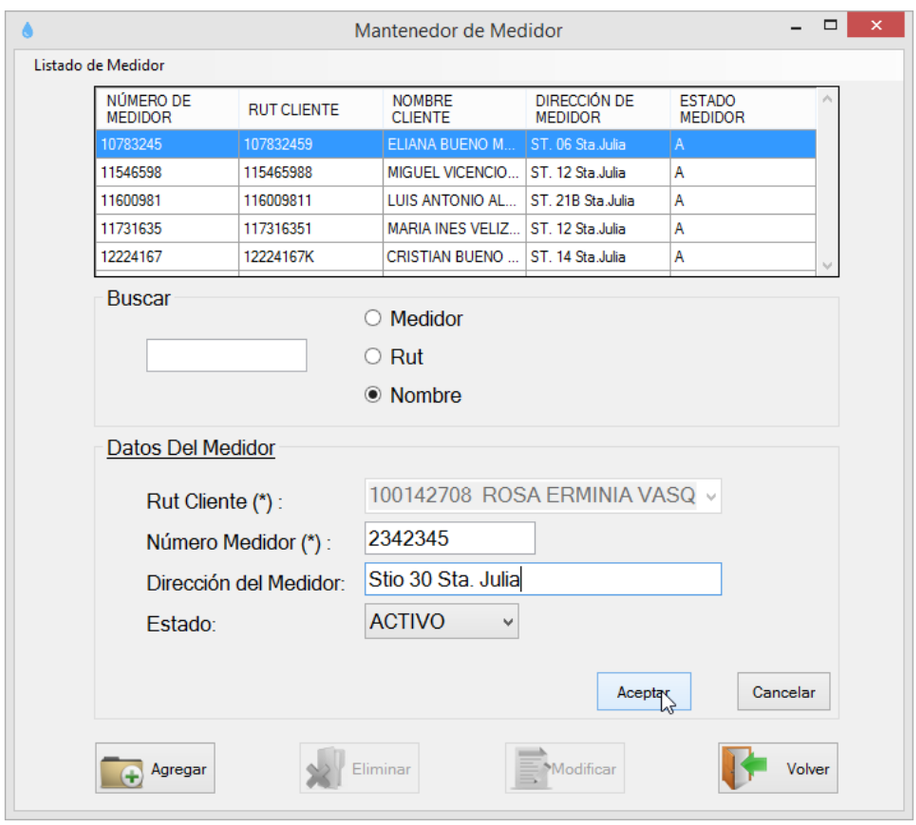


Figura (1.9 Agregar Medidor)

- **Modificar Medidor:** el usuario podrá modificar tan solo la dirección del medidor, presionando el botón modificar, habilitando la casilla dirección y estado. Una vez finalizada esta operación el programa desplegará un mensaje por pantalla que se logró realizar el cambio, en caso contrario, pedirá que ingrese datos en los campos obligatorios.

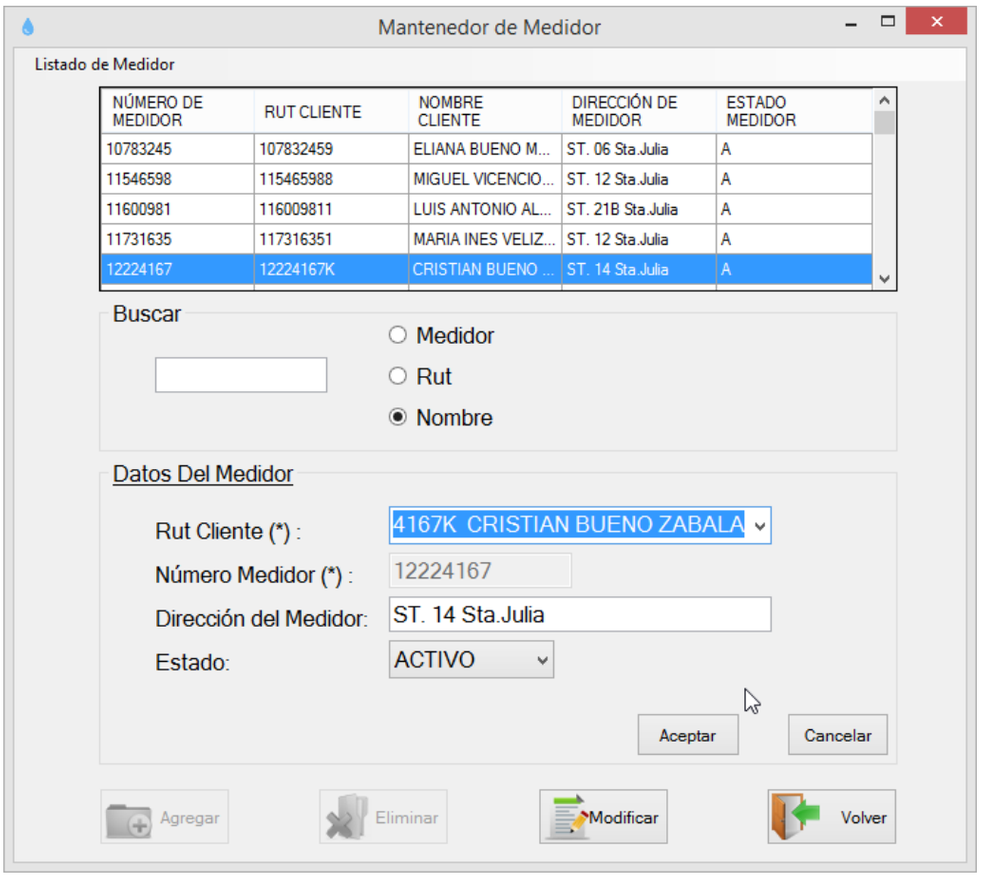


Figura (1.10 Agregar Medidor)

- **Eliminar Medidor:** se elige el medidor a eliminar para luego presionar el botón eliminar. Esta operación se realizará correctamente mientras el medidor no tenga datos asociados ni a consumo, ni a pagos. Si el medidor tiene datos asociados a otras tablas se puede realizar una eliminación lógica del medidor cambiando su estado a inactivo.

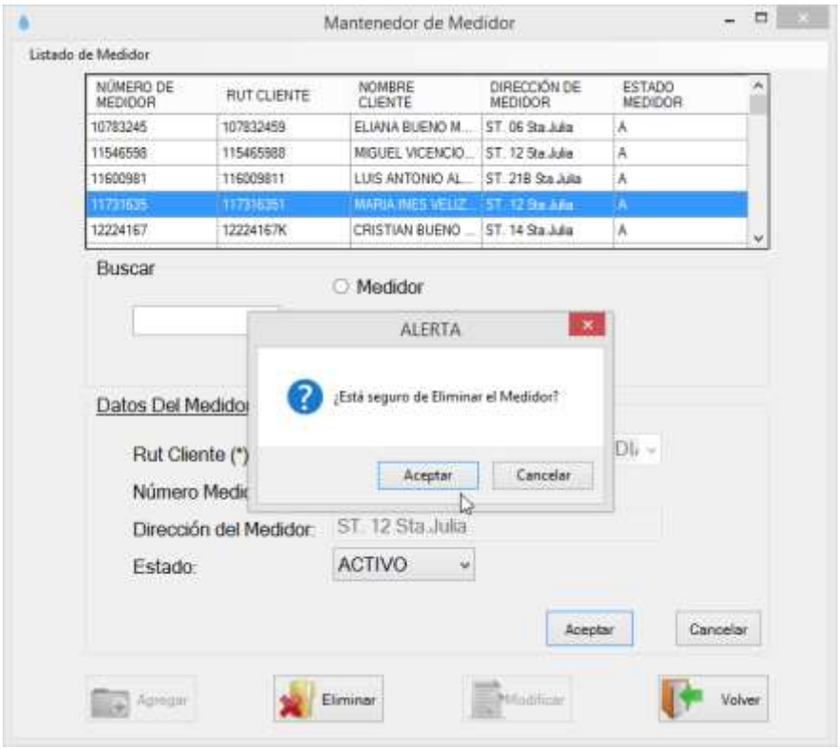


Figura (1.11 Agregar Medidor)

- **Consultar Medidor:** el usuario podrá consultar por medidores activos o inactivos, por número de medidor, Rut y nombre de cliente, dependiendo de las necesidades de este.

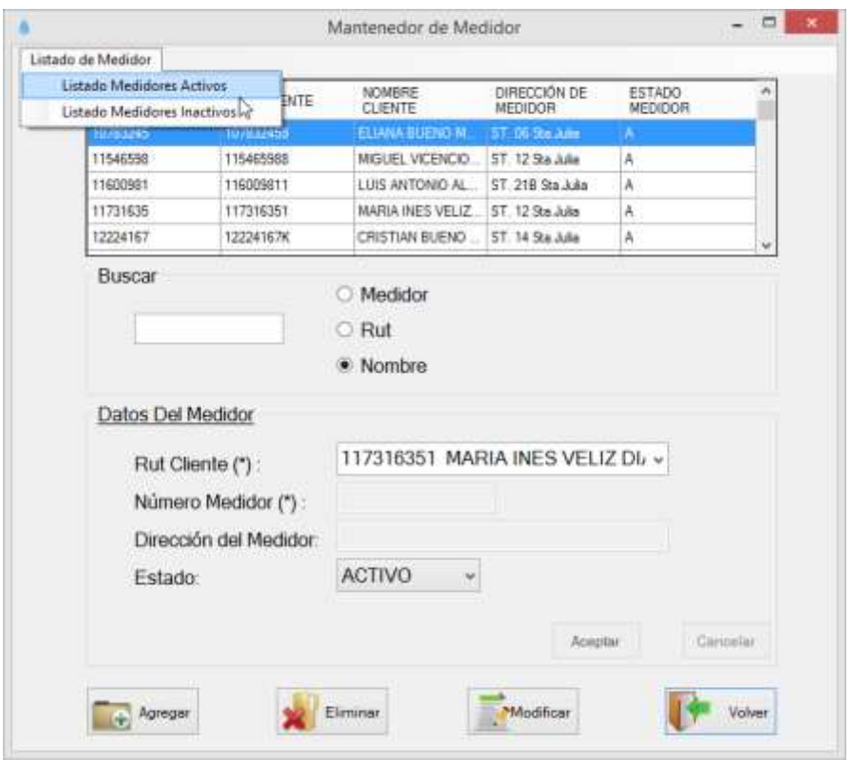


Figura (1.12 Agregar Medidor)

Código fuente: Anexo página 62.

3.4.5. Nombre del programa: Gestor de Boleta.

Objetivo: el usuario podrá realizar los cobros pertinentes de consumo de agua por cada cliente y medidor registrados en la empresa.

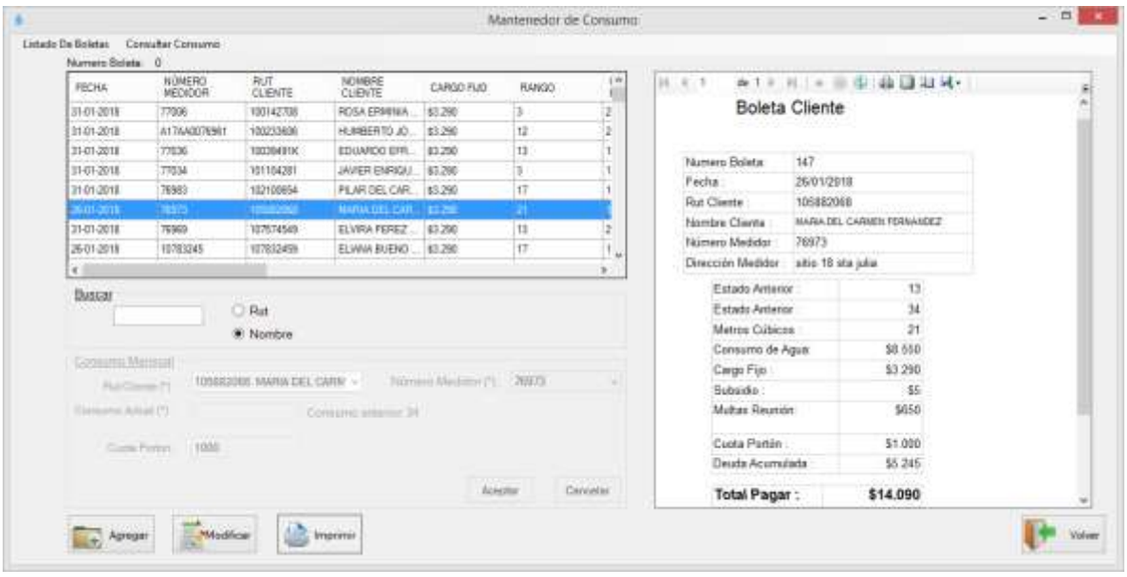


Figura (1.15 Mantenedor de Consumo y Vista de Boleta)

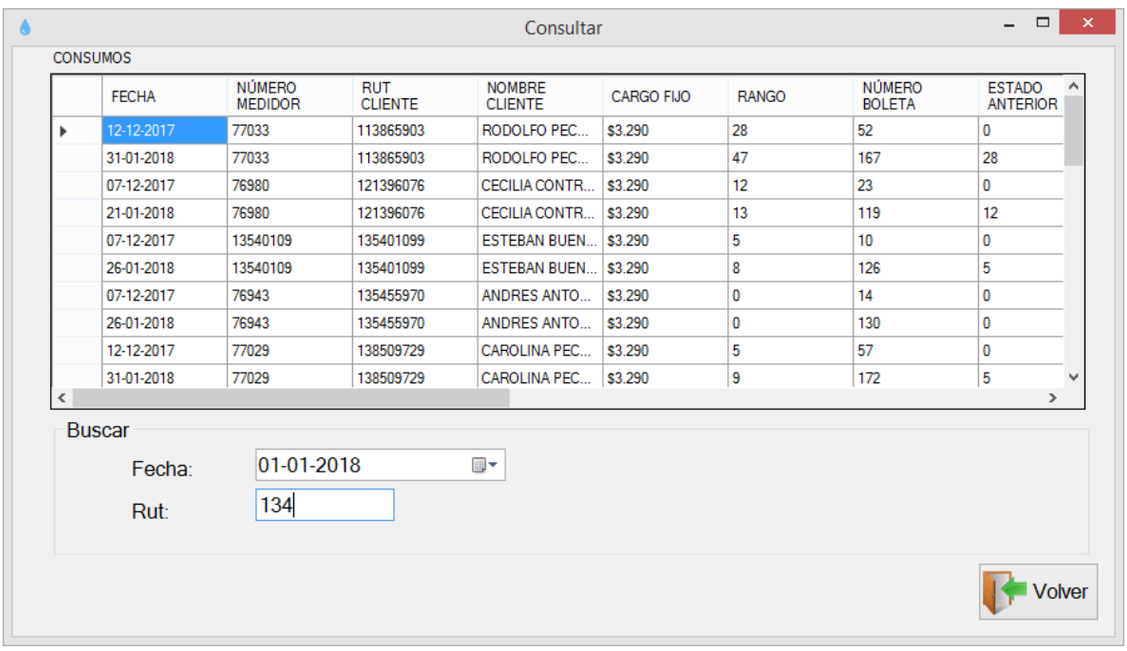


Figura (1.16 Consulta consumo de clientes)

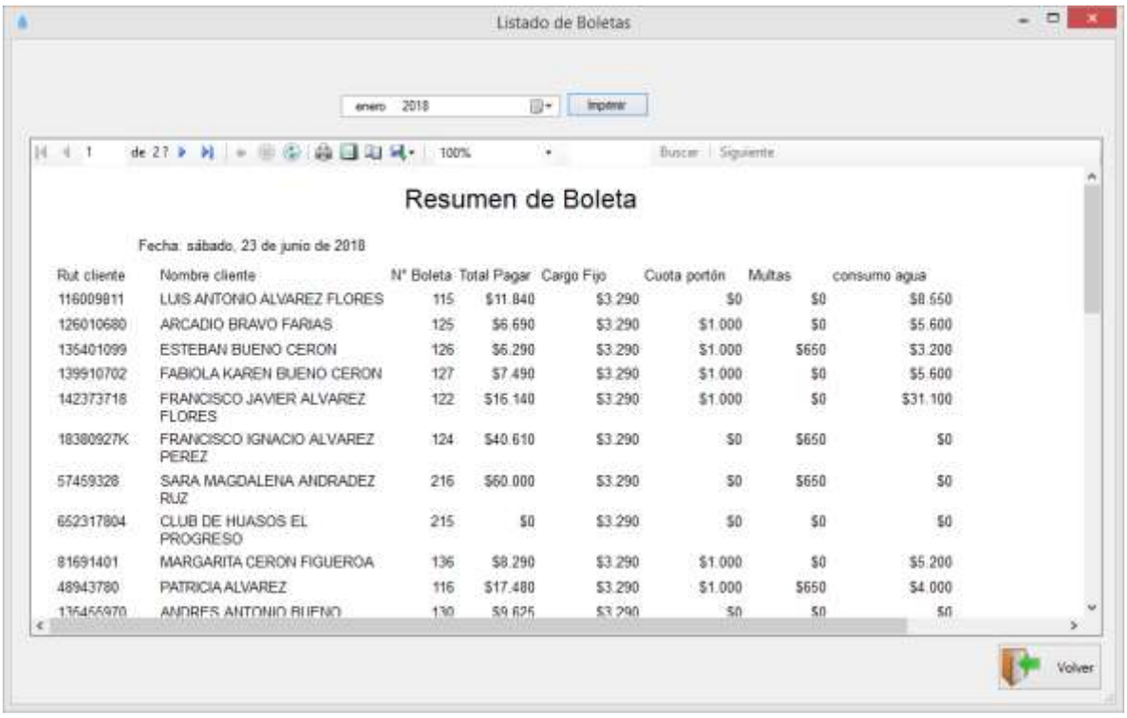
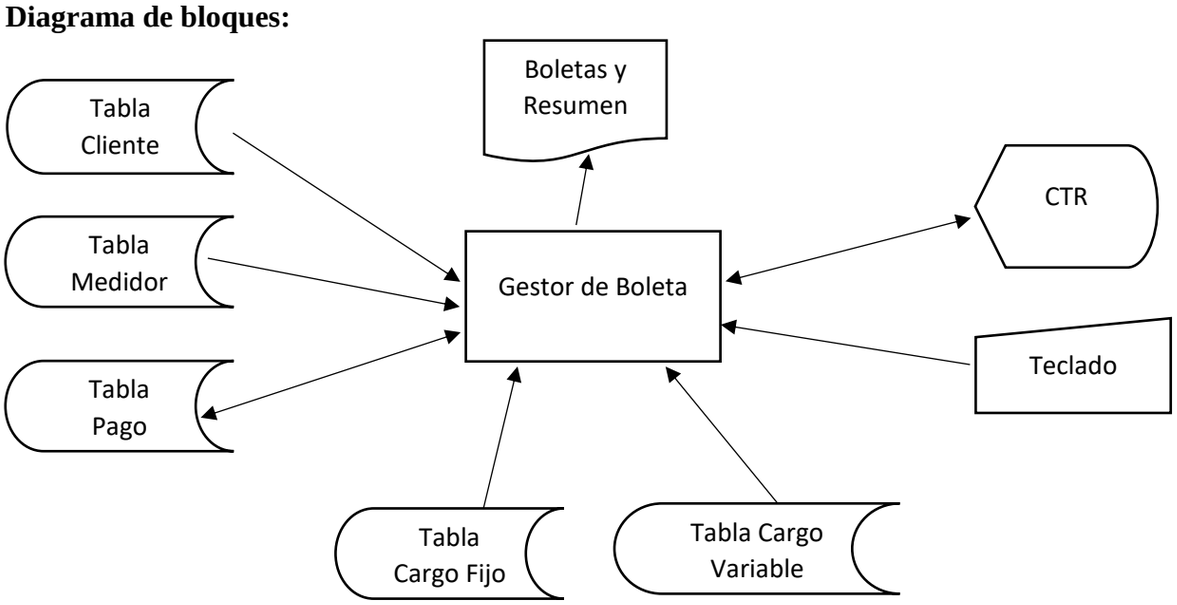


Figura (1.17 Listado de Boletas)

Diagrama de bloques:



Reglas de negocio:

Se presenta en la pantalla el gestor de boletas, donde el usuario podrá realizar las siguientes funciones:

- **Ingresar consumo:** el usuario debe presionar el botón Agregar, posteriormente se le habilitarán las casillas, el botón aceptar y el botón cancelar.

  Luego el usuario debe buscar, por Rut o nombre, al cliente y al medidor al cual corresponde el consumo a ingresar. Una vez ingresados todos los datos pedidos por

obligación, consumo y cuota portón, procede a presionar el botón Aceptar. En caso de que algún campo este vacío se mostrará un mensaje de error por pantalla, obligando al usuario a llenar todos los campos. Si el consumo ingresado es menor al consumo del mes anterior, se mostrará por pantalla un mensaje de error, ya que el consumo actual no puede ser menor a este. Por último, si el rango entre el consumo ingresado y el consumo anterior es mayor al rango del cargo variable máximo registrado, se mostrará un mensaje de error obligando al usuario a modificar la tabla de cargo variable. Si no existe ningún error el consumo será ingresado exitosamente.

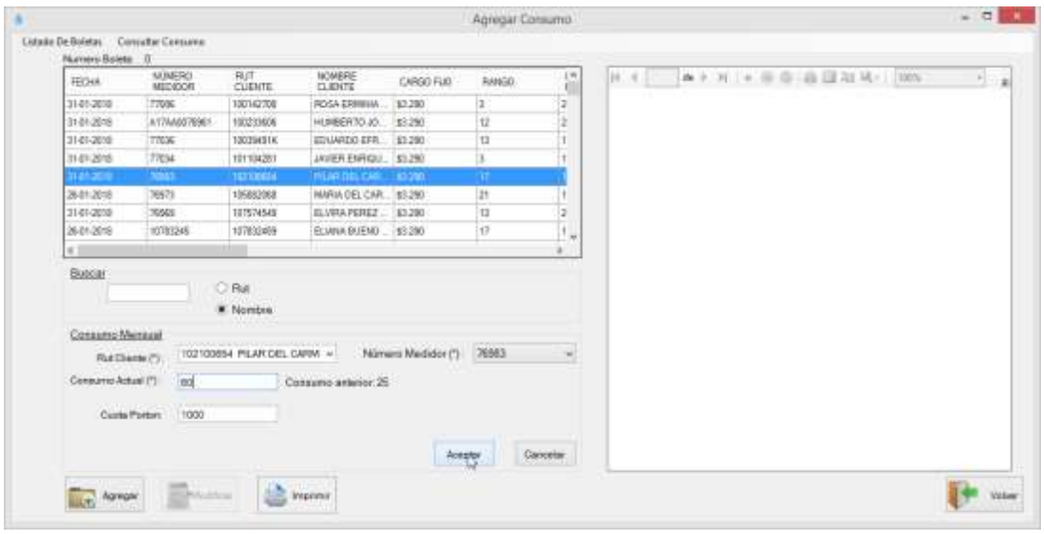


Figura (1.18 Agregar Consumo)

- **Modificar Consumo:** si el usuario desea modificar el consumo de algún medidor debe presionar el botón Modificar, posteriormente se le habilitarán las casillas, el botón aceptar y el botón cancelar y se deshabilitarán los botones de agregar e imprimir. Esta función permite sólo la modificación del último registro de cada medidor, por lo que el usuario, al querer modificar el ingreso de consumo, sólo tendrá acceso a estos. En caso de que algún campo este vacío se mostrará un mensaje de error por pantalla, obligando al usuario a llenar todos los campos. Si el consumo ingresado es menor al consumo del mes anterior, se mostrará por pantalla un mensaje de error, ya que el consumo actual no puede ser menor a este. Por último, si el rango entre el consumo ingresado y el consumo anterior es mayor al rango del cargo variable máximo registrado, se mostrará un mensaje de error obligando al usuario a modificar la tabla de cargo variable. Si no existe ningún error el consumo será modificado exitosamente.

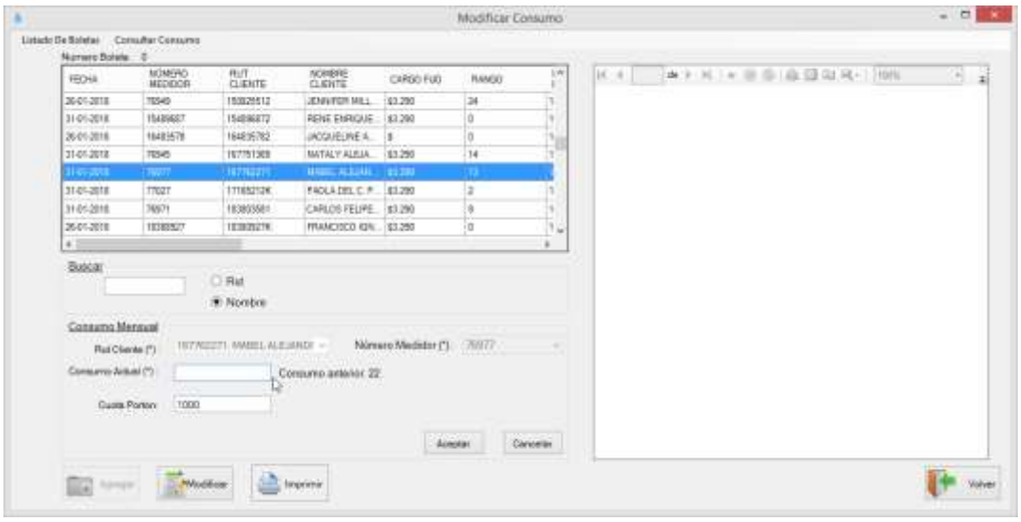


Figura (1.19 Modificar Consumo)

- **Imprimir boletas:** si el usuario desea imprimir la boleta de consumo de algún medidor debe buscar, por nombre o Rut, el cliente y elegir el medidor correspondiente al consumo que desea imprimir. Una vez elegido el cliente con su medidor, se debe presionar el botón imprimir, posteriormente se deshabilitarán los botones de agregar y modificar y aparecerá en pantalla la boleta correspondiente al consumo elegido. El usuario debe presionar la opción de imprimir la boleta obteniendo como resultado la boleta impresa.

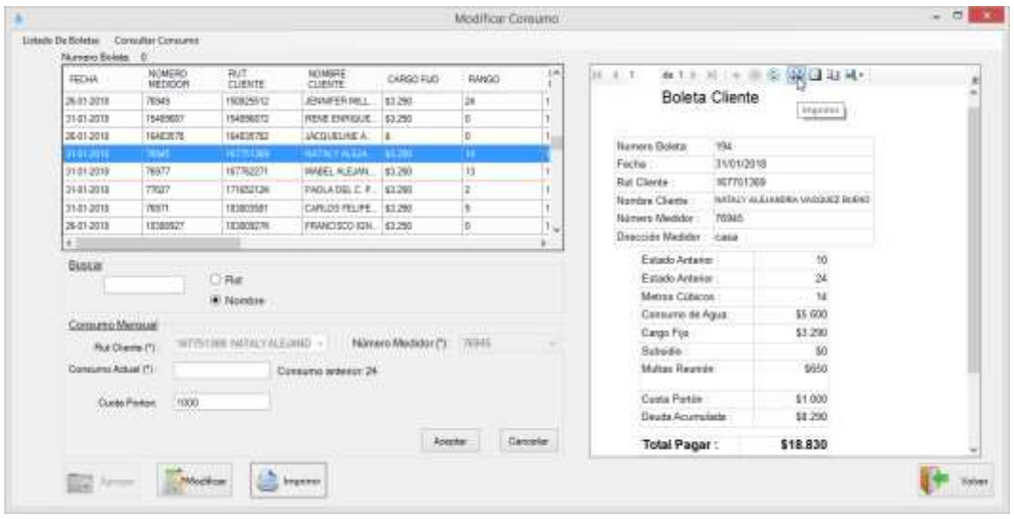
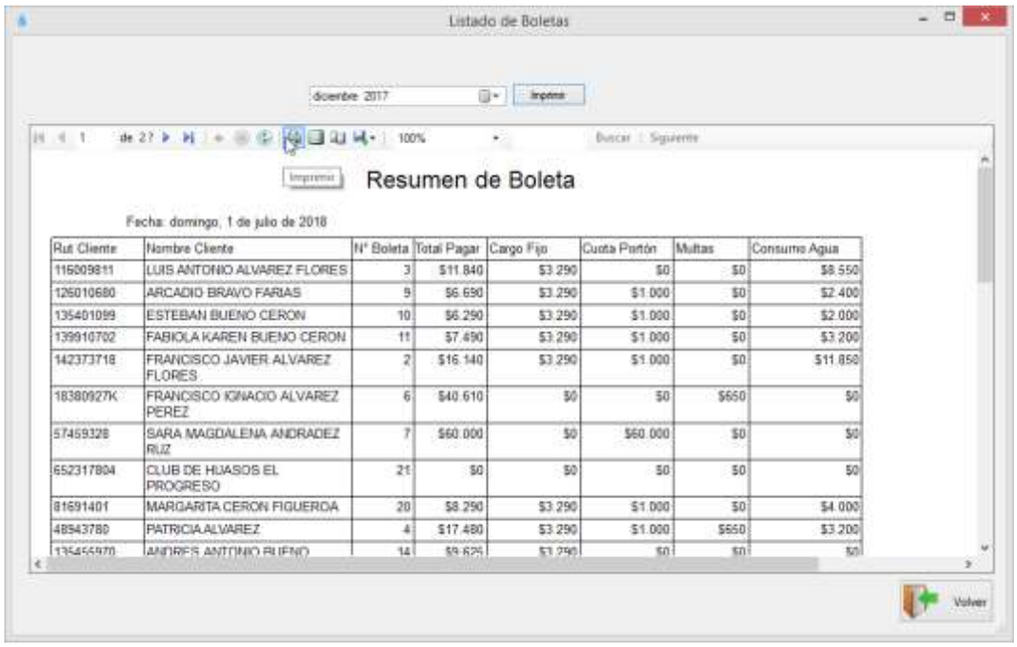


Figura (1.20 Imprimir Boleta)

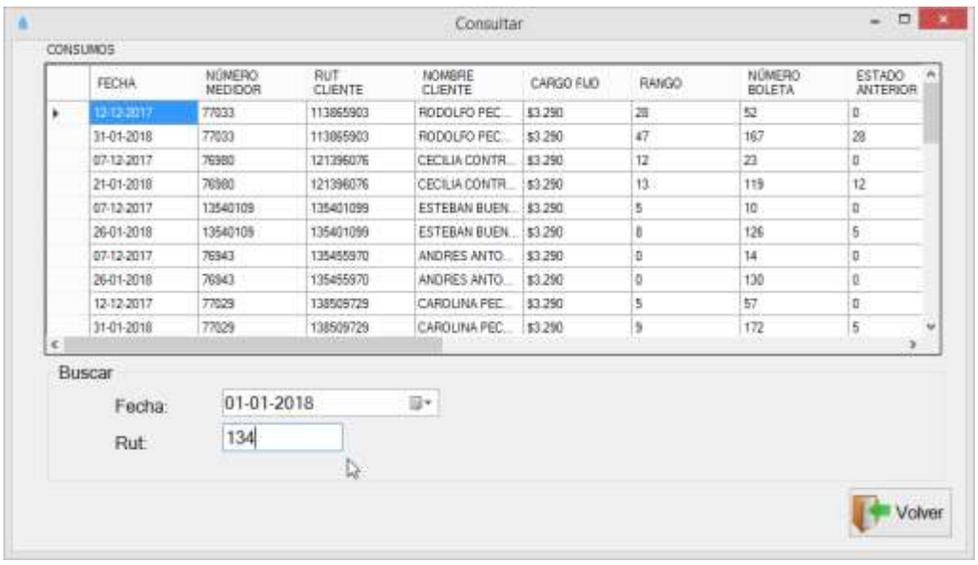
- **Imprimir Resumen de Boletas:** si el usuario desea imprimir o consultar un resumen de las boletas emitidas en el mes debe presionar el botón, ubicado en la parte superior de la ventana, Listado de Boletas, posteriormente aparecerá una nueva ventana, en la cual el usuario deberá elegir la fecha/mes que desea consultar. Si el usuario desea imprimir el resumen debe presionar el botón imprimir, obteniendo como resultado el resumen de boletas del mes seleccionado impreso.



Rut Cliente	Nombre Cliente	N° Boleta	Total Pagar	Cargo Fijo	Cuota Portón	Multas	Consumo Agua
116009811	LUIS ANTONIO ALVAREZ FLORES	3	$11.840	$3.290	$0	$0	$8.550
126010680	ARCADIO BRAVO FARIAS	9	$6.690	$3.290	$1.000	$0	$2.400
135401099	ESTEBAN BUENO CERON	10	$6.290	$3.290	$1.000	$0	$2.000
139910702	FABIOLA KAREN BUENO CERON	11	$7.490	$3.290	$1.000	$0	$3.200
142373718	FRANCISCO JAVIER ALVAREZ FLORES	2	$16.140	$3.290	$1.000	$0	$11.850
18380927K	FRANCISCO IGNACIO ALVAREZ PEREZ	6	$40.610	$0	$0	$650	$0
57459328	SARA MAGDALENA ANDRADEZ RUZ	7	$60.000	$0	$60.000	$0	$0
652317804	CLUB DE HUASOS EL PROGRESO	21	$0	$0	$0	$0	$0
81691401	MARGARITA CERON FIGUEROA	20	$8.290	$3.290	$1.000	$0	$4.000
48543780	PATRICIA ALVAREZ	4	$17.480	$3.290	$1.000	$650	$3.200
135455970	ANDRES ANTONIO BUENO	14	$9.625	$3.290	$0	$0	$0

Figura (1.21 Resumen De Boleta)

- **Consultar consumo:** el usuario podrá realizar consultas sobre el consumo, las cuales pueden ser por el nombre o Rut del cliente o por la fecha del consumo a buscar.



FECHA	NÚMERO MEDIDOR	RUT CLIENTE	NOMBRE CLIENTE	CARGO FIJO	RANGO	NÚMERO BOLETA	ESTADO ANTERIOR
12-12-2017	77033	113865903	RODOLFO PEC...	$3.290	28	52	0
31-01-2018	77033	113865903	RODOLFO PEC...	$3.290	47	167	28
07-12-2017	76980	121396076	CECILIA CONTR...	$3.290	12	23	0
21-01-2018	76980	121396076	CECILIA CONTR...	$3.290	13	119	12
07-12-2017	13540109	135401099	ESTEBAN BUEN...	$3.290	5	10	0
26-01-2018	13540109	135401099	ESTEBAN BUEN...	$3.290	8	126	5
07-12-2017	76943	135455970	ANDRES ANTO...	$3.290	0	14	0
26-01-2018	76943	135455970	ANDRES ANTO...	$3.290	0	130	0
12-12-2017	77029	138509729	CAROLINA PEC...	$3.290	5	57	0
31-01-2018	77029	138509729	CAROLINA PEC...	$3.290	9	172	5

Figura (1.22 Consultar Consumo)

Código Fuente: Anexo página 65.

3.4.6. Nombre del programa: Mantenedor de Reuniones

Objetivo: permite registrar y modificar la asistencia a las reuniones calendarizadas por la empresa.

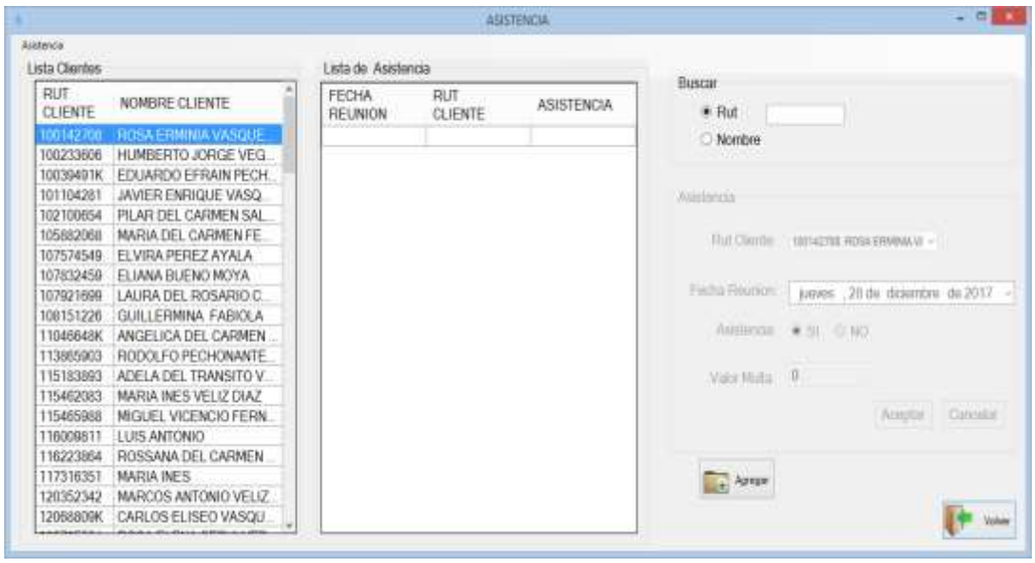


Figura (1.23 Asistencia Reunión)

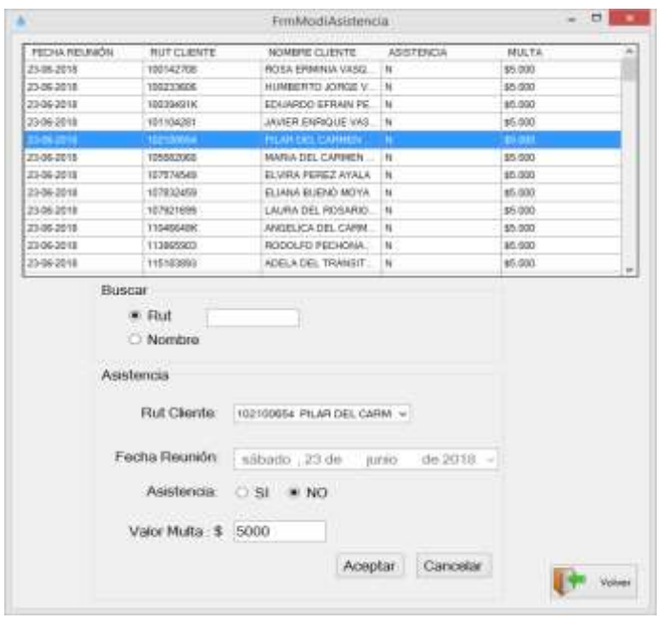
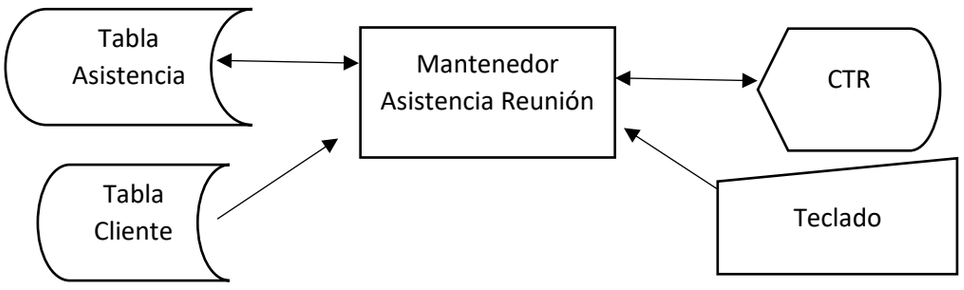


Figura (1.24 Modificar Asistencia)

Diagrama de bloques:



Reglas de negocio:

Se presenta en la pantalla el mantenedor de asistencia a reunión, donde el usuario podrá realizar las siguientes funciones:

- **Registrar Asistencia:** en el momento en que el cliente ingrese al lugar de citación a reunión el usuario, mediante el nombre o Rut, podrá registrar su asistencia. Para realizar esta acción se debe presionar el botón agregar y se habilitarán las casillas correspondientes a esta operación. En caso de que el Rut sea inválido el programa alertará sobre esta anomalía, sino registrará exitosamente la asistencia y agregará a la persona en la lista de "asistencia presente" en la vista.

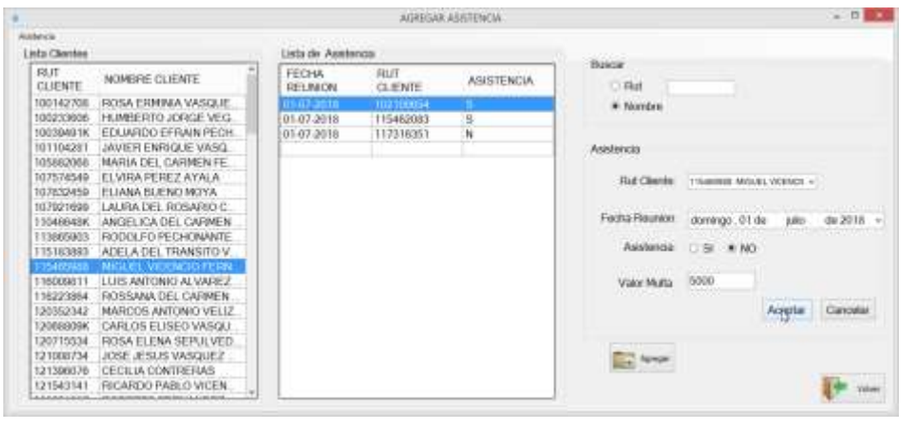


Figura (1.25 Agregar Asistencia)

- **Modificar Asistencia:** el usuario podrá modificar la asistencia del cliente mediante esta operación. Debe presionar modificar asistencia, se desplegará una pantalla con el listado de las asistencias, el usuario deberá buscar por nombre o Rut del cliente, una vez realizada la modificación el sistema mostrará un mensaje de que esta fue modificada exitosamente.

Figura (1.26 Modificar Asistencia)

Código Fuente: Anexo página 71.

3.4.7. Nombre Programa: Mantenedor de cargos

3.4.7.1. Nombre Subprograma: Mantenedor de cargo variable

Objetivo: el usuario puede apreciar los distintos cargos variables por metros cúbicos con la finalidad de aumentar o disminuir estos valores.

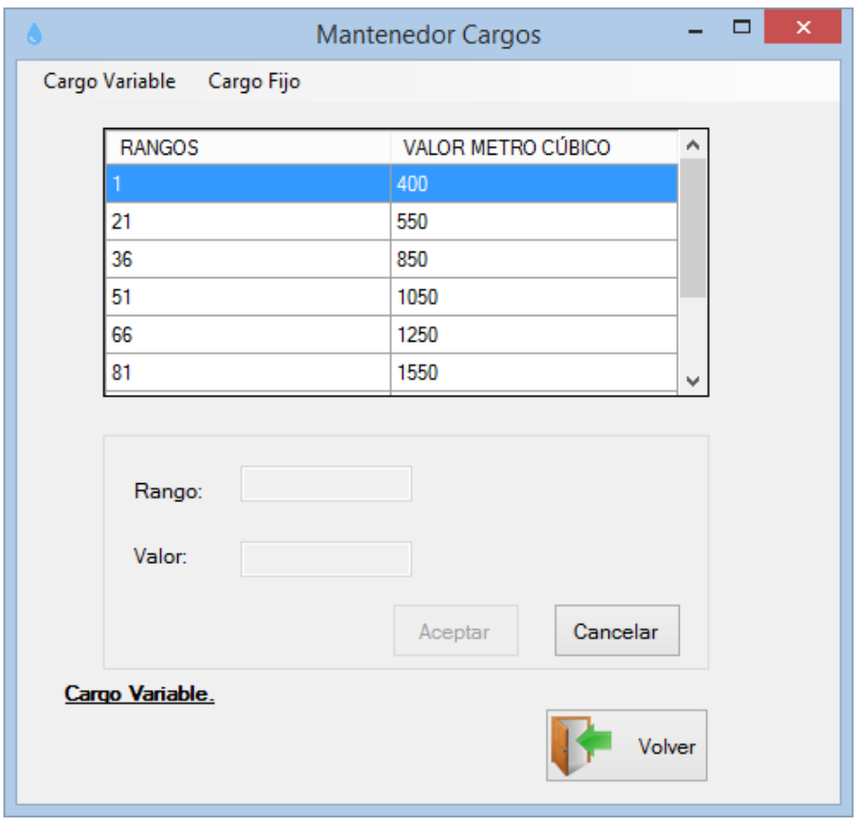
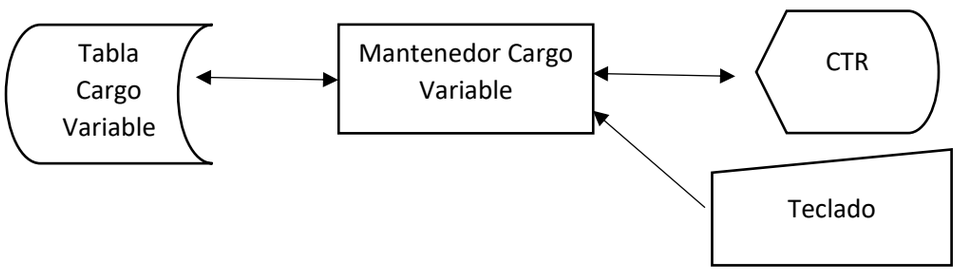


Figura (1.27 Mantenedor Cargo Variable)

Diagrama de Bloques:



Reglas de Negocio:

Se despliega una pantalla que contiene el listado de los valores de cargo variables los cuales se utilizan para el cálculo de consumo de agua potable. Al presionar el botón Cargo Variable se mostrarán las siguientes funcionalidades:

- **Agregar Cargo Variable**: permite agregar un nuevo rango de cargo variable. Se debe ingresar el código y valor de este, una vez llenado los campos se presiona aceptar y éste se registra en la tabla, sino despliega un mensaje de error por campos vacíos o de que el cargo ya fue ingresado.

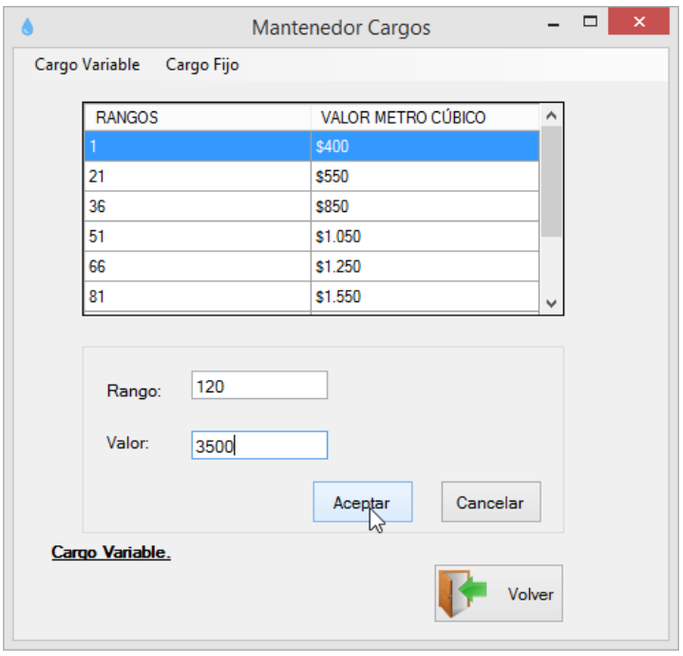


Figura (1.28 Agregar Cargo Variable)

- **Modificar Cargo Variable**: el usuario podrá modificar el valor del cargo variable aumentando su valor o disminuyéndolo. Una vez realizado el cambio de valor se debe presionar aceptar, si el valor es correcto se modificará, en caso contrario, desplegará un mensaje de error que no se pudo modificar el cargo variable.

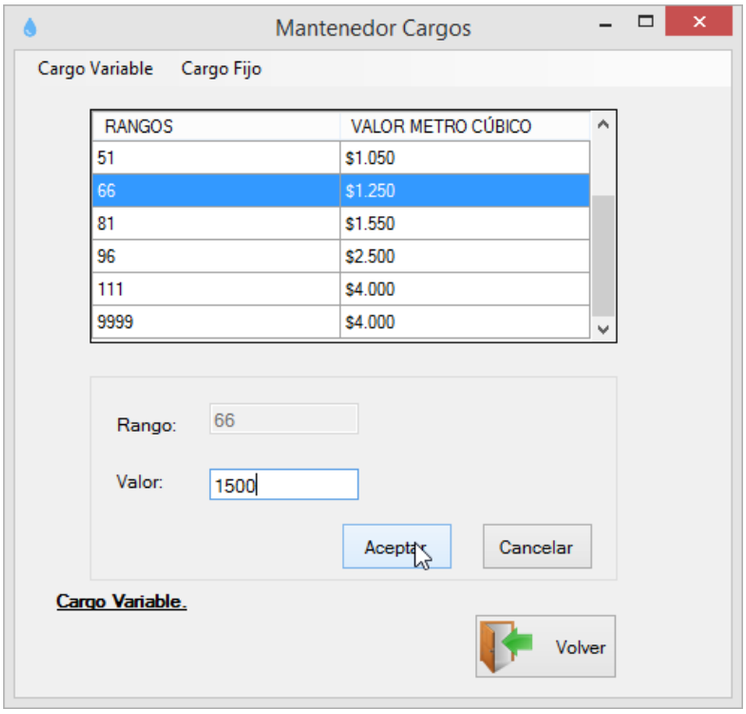


Figura (1.29 Modificar Cargo Variable)

- **Eliminar Cargo Variable:** se podrá eliminar un rango de consumo de agua. Se debe seleccionar el rango, posteriormente presionar aceptar y el cargo variable se eliminará.

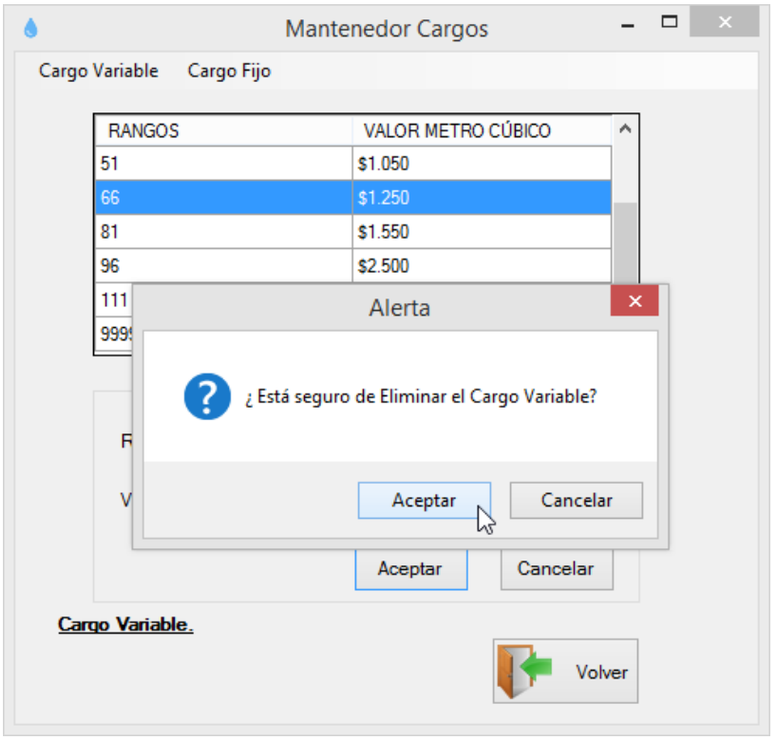


Figura (1.30 Eliminar Cargo Variable)

3.4.7.2. Nombre Subprograma: Mantenedor de cargo variable

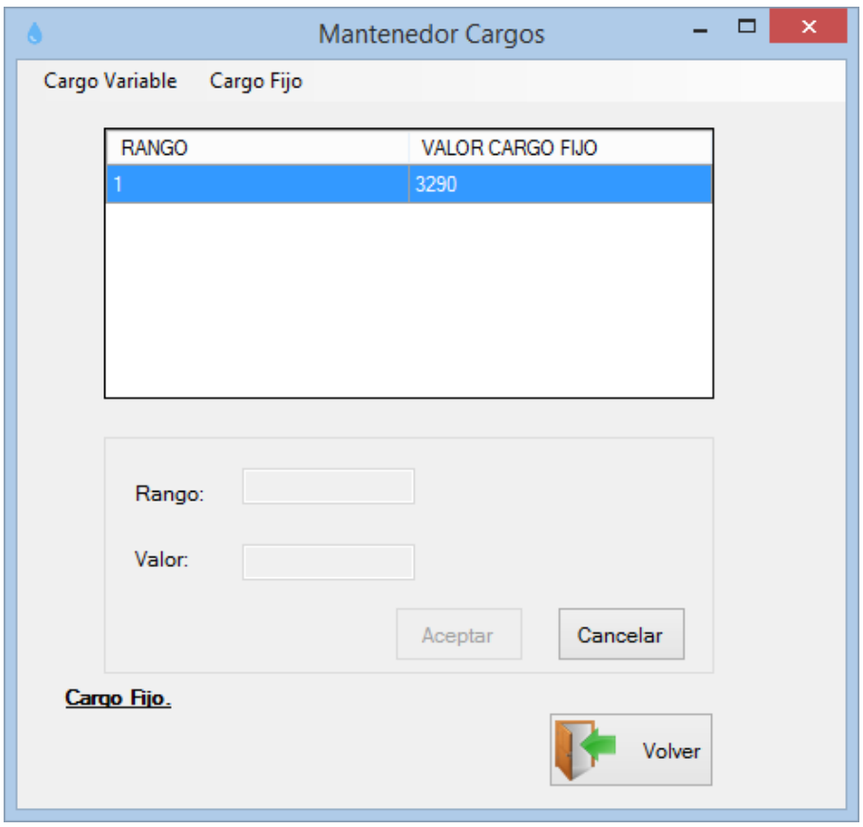
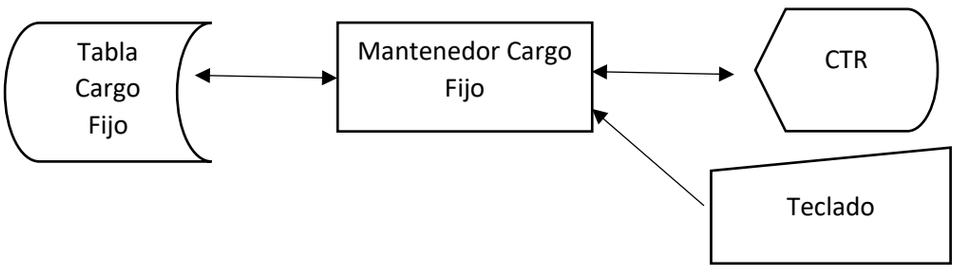


Figura (1.31 Mantenedor Cargo Fijo)

Diagrama de Bloques:



Reglas de Negocio: el cliente podrá ver el cargo fijo el cual se cobra en cada boleta de los clientes. Solo podrá modificar el valor de este ya que es un valor único. Para lograr esto debe cambiar el campo "Valor" posteriormente presionar el botón aceptar, se desplegará un mensaje que se logró modificar el cargo fijo.

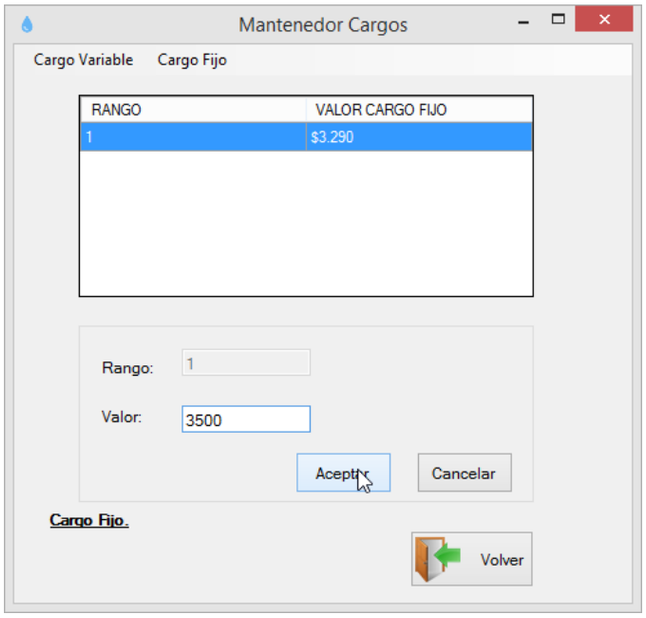


Figura (1.32 Modificar Cargo Fijo)

Código Fuente: Anexo página 73.

3.4.8. Nombre Programa: Mantenedor de Usuario

Objetivo: Permite visualizar los usuarios registrados en el sistema con sus diferentes niveles.

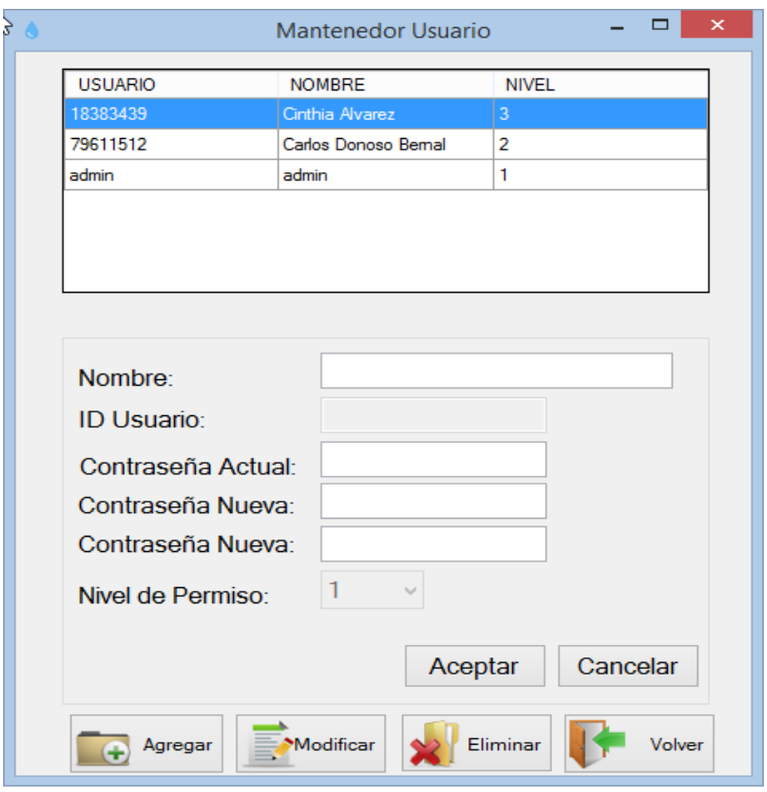


Figura (1.33 Mantenedor Usuario)

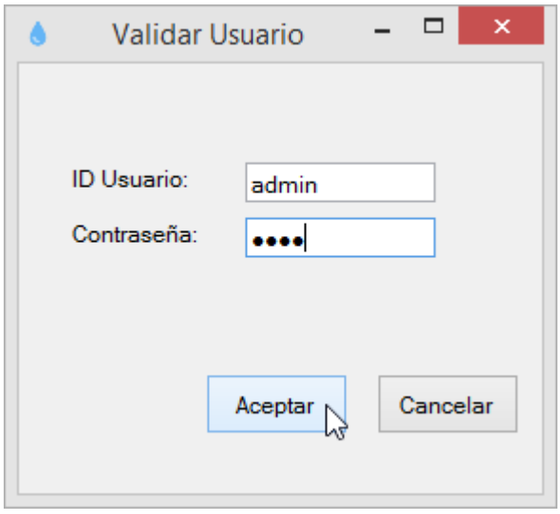
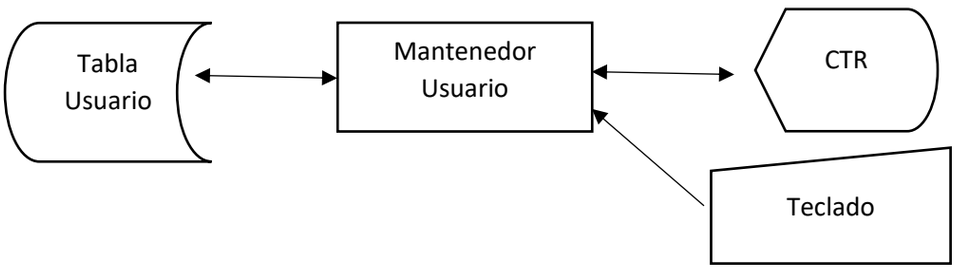


Figura (1.34 Validador de Usuario)

Diagrama de Bloques:



Reglas de Negocio:

En el sistema existen diferencias en el tipo de usuario que ingrese a esta función, el nivel 1 y 2 tienen los mismos permisos de agregar, modificar y eliminar, mientras que el nivel 3 solo puede modificar su contraseña.

Para realizar alguna acción primero se debe validar el usuario que está ingresando (figura 1.13 validador de Usuario). Si el usuario que ingresó corresponde al mismo que está validando permitirá ingresar, de lo contrario se desplegará un mensaje de error ya que no es el usuario correcto.

Posterior a esto se podrán realizar las siguientes operaciones:

- **Agregar Usuario:** se deben ingresar todos los campos solicitados por la vista (figura 1.12 Mantenedor Usuario), esto es permitido solo para los niveles 1 y 2. Luego se debe presionar aceptar. En caso que el usuario sea válido se procede a registrar, de lo contrario se desplegará un mensaje de que el usuario ya existe.

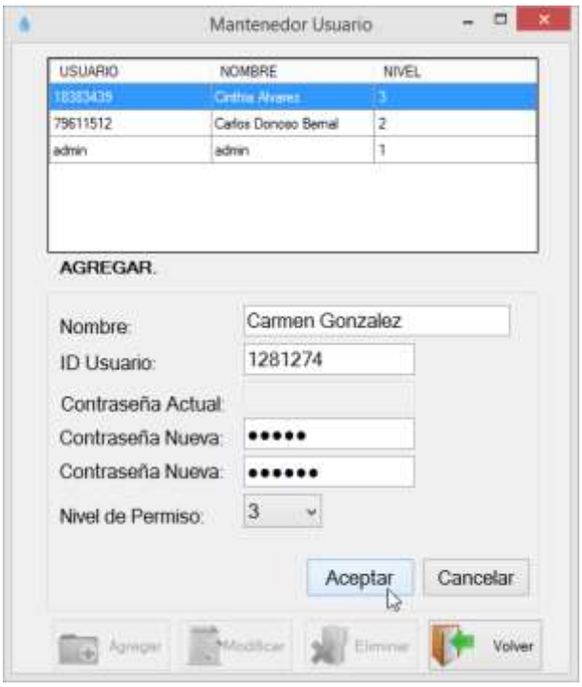


Figura (1.35 Agregar Usuario)

- **Modificar Usuario:** Permite modificar los campos habilitados, estos son: nombre usuario y contraseña, dependiendo de los requerimientos de éste. Una vez hecho el cambio se debe presionar aceptar para actualizar los datos. Si es válido despliega un mensaje de que el cambio se realizó con éxito, de lo contrario, se despliega un mensaje de error.

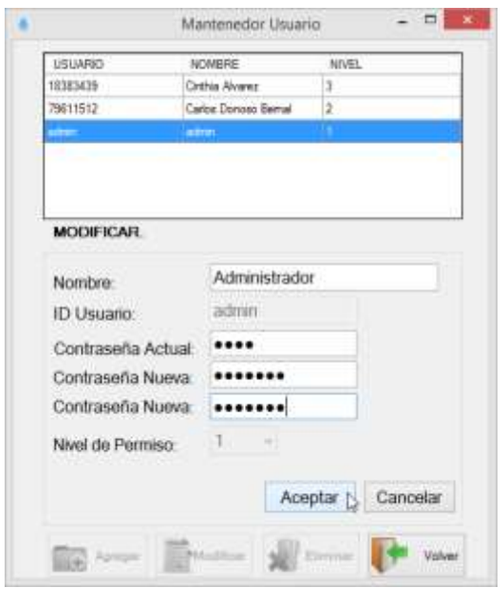


Figura (1.36 Modificar Usuario)

- **Eliminar Usuario:** solo los niveles 1 y 2 podrán realizar esta operación sobre los demás usuarios pero no sobre sí mismo. Se debe seleccionar el usuario luego presionar el botón eliminar para habilitar el botón aceptar y cancelar, luego presionar aceptar, si es afirmativa la respuesta se procede a eliminar el usuario.

Figura (1.37 Eliminar Usuario)

Código Fuente: Anexo página 76.

3.4.9. Nombre Programa: Mantenedor de Pago.

Objetivo: Se despliega una pantalla, la cual muestra los últimos registros de consumo de agua ingresados por medidor al lado izquierdo y los comprobantes a imprimir al lado derecho. Permite ingresar el pago del consumo de agua anterior del medidor e imprimir el comprobante de pago para el cliente. Además, permite realizar consultas de los pagos efectuados por cliente o fecha.

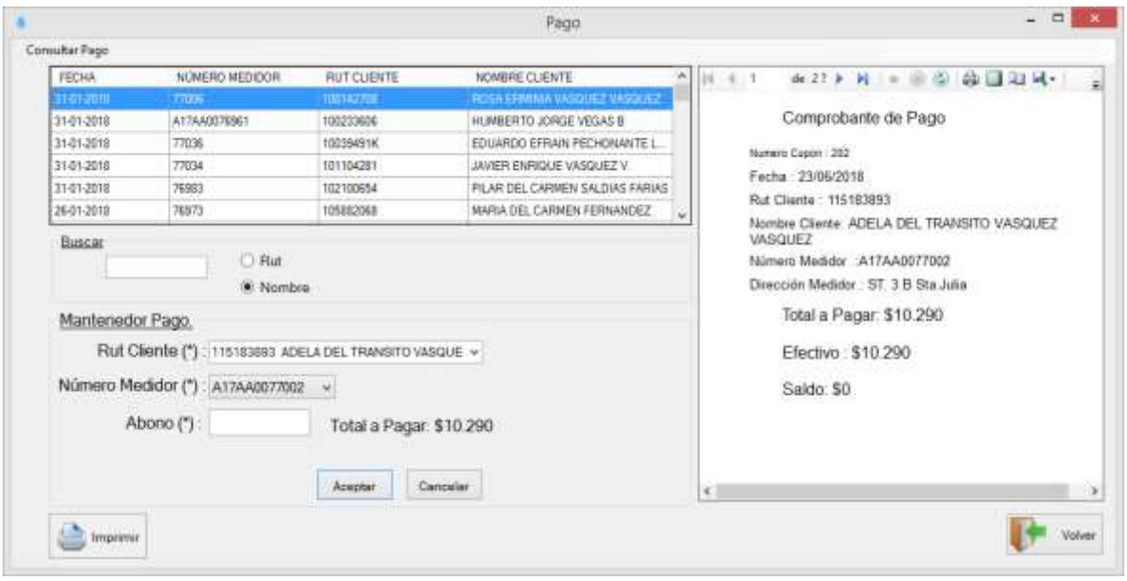


Figura (1.38 Pago de Boletas y Comprobante de Pago)

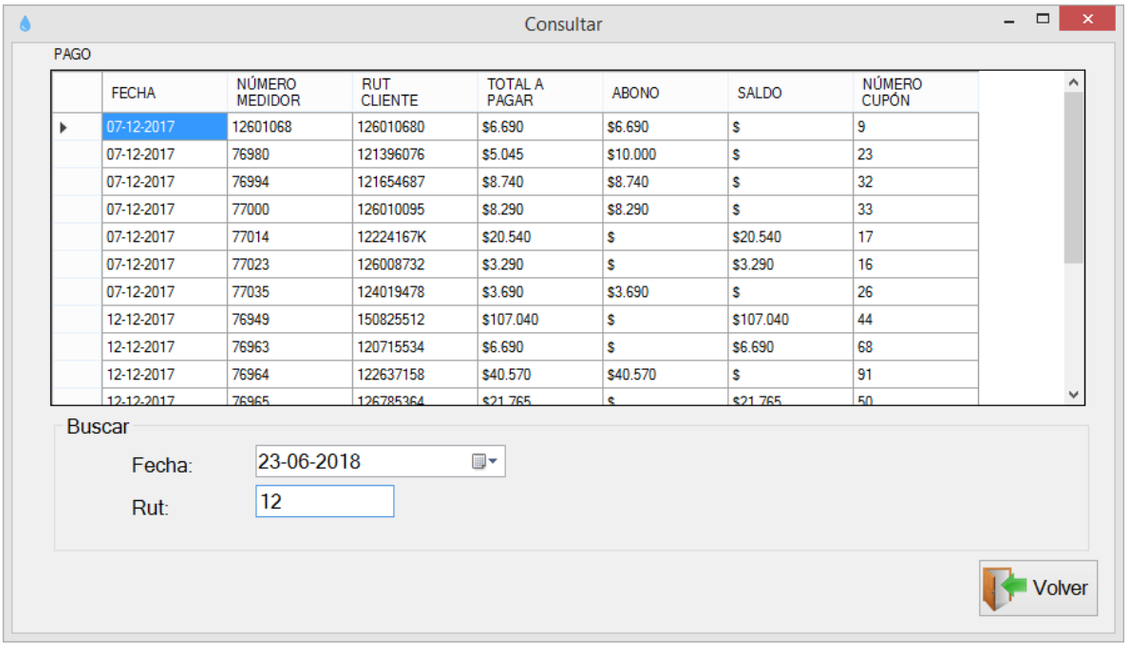
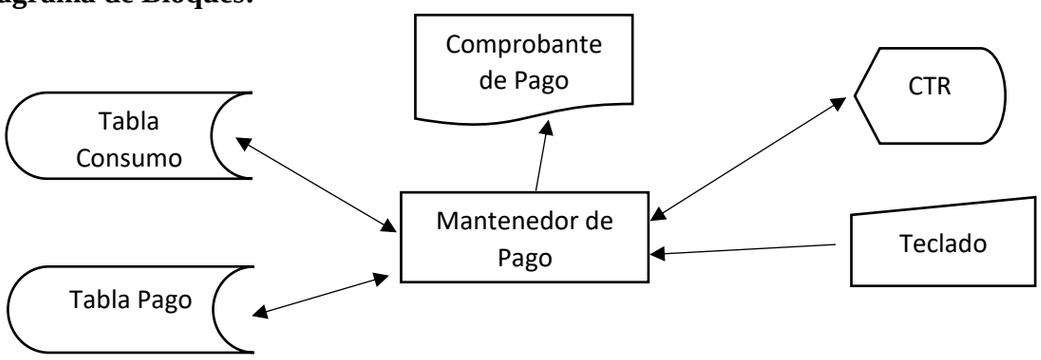


Figura (1.39 Vista Comprobante de Pago)

Diagrama de Bloques:



Reglas de negocio:

Se presenta en la pantalla el mantenedor de pago, donde el usuario podrá realizar las siguientes funciones:

- **Ingresar pago:** el usuario debe buscar, por Rut o nombre de cliente, y seleccionar el consumo correspondiente al pago que se está efectuando, esta función muestra por pantalla sólo los últimos consumos ingresados y que no se han pagado. Luego de seleccionar el consumo buscado debe escribir en la casilla correspondiente el monto de dinero que el cliente está pagando, una vez hecho esto se presiona el botón aceptar. Si al momento de presionar aceptar el campo abono se encuentra vacío se mostrará un mensaje de error, lo que obliga al usuario a llenar los campos, en caso contrario, el pago será ingresado correctamente.

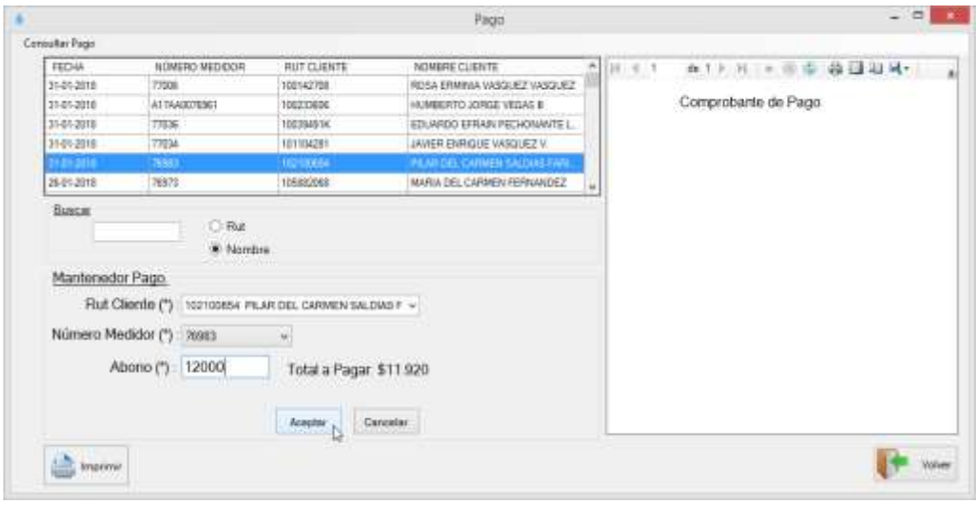


Figura (1.40 Ingresar Pago)

- **Imprimir comprobante de pago:** cuando el cliente realice el pago de la boleta y ya ingresado en el sistema, el usuario deberá presionar el botón de Imprimir obteniendo así el comprobante de pago impreso para entregar al cliente.

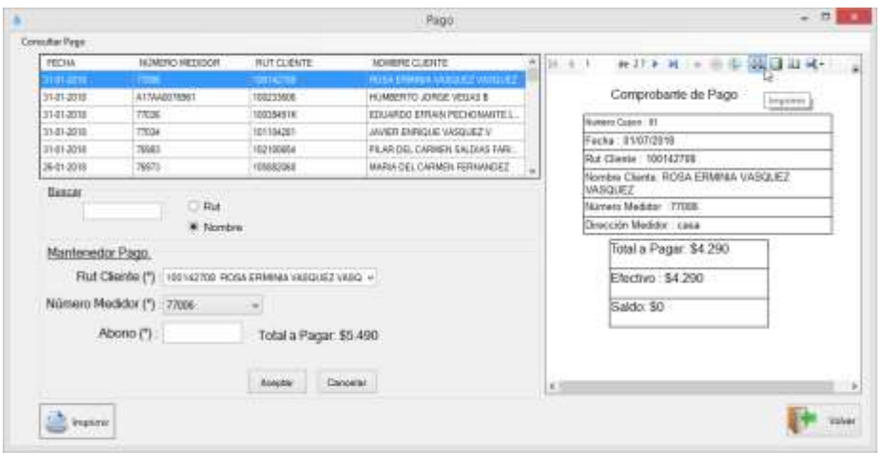


Figura (1.41 Imprimir Pago)

- **Consultar pago:** el usuario podrá realizar consultas, por Rut o nombre de cliente, sobre los últimos consumos de agua que se deben pagar y en otra ventana, presionando el botón Consultar Pagos ubicado en la parte superior de la pantalla, podrá consultar, por fecha o Rut de cliente, los pagos ya efectuados.

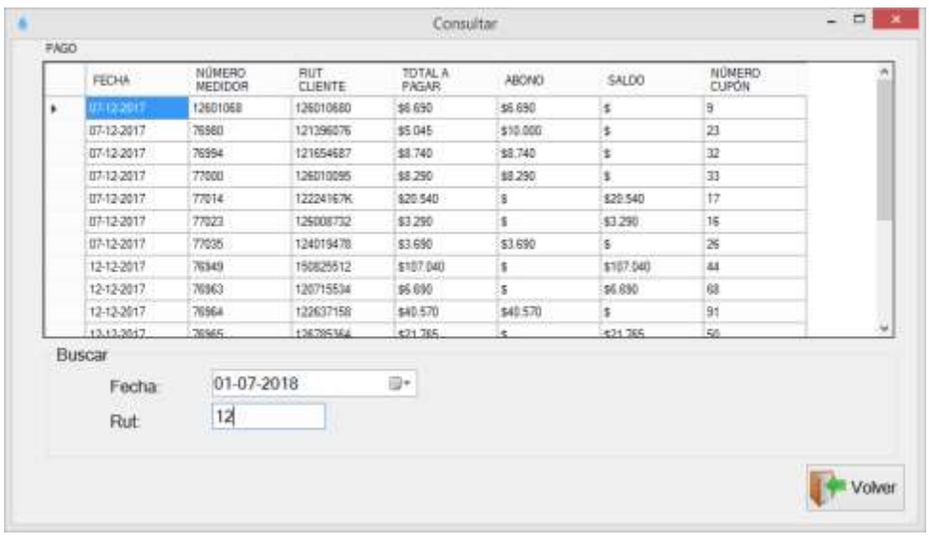


Figura (1.42 Consultar Pago)

Código Fuente: Anexo página 78.

CONCLUSIÓN

Comité de agua potable Santa Julia es una empresa de distribución de agua potable para la comunidad de Santa Julia, Quintero. Las actividades informáticas de esta empresa se realizan de manera manual, desde la toma del consumo de agua potable, el cálculo de lo que debe pagar cada cliente hasta la escritura de las boletas con sus respectivos comprobantes de pago. El sistema informático desarrollado pretende facilitar y reducir el tiempo del trabajo de los usuarios, automatizando todo lo relacionado con el registro de clientes y medidor, el cálculo de los consumos y la impresión de las boletas y comprobantes.

'Sistema informático administrador de agua potable Santa Julia' se desarrolló en el lenguaje de computación visual .net, con una interfaz amigable a la vista y sencilla de entender por parte del usuario. Tiene como principal objetivo facilitar la labor de los usuarios, por esta razón se consideraron todos los requerimientos que el cliente quería y necesitaba para su desarrollo, organizando reuniones periódicas para mostrar el avance del sistema y para la toma de nuevos requerimientos. A partir de estas necesidades se introdujo orden a los registros y archivos de la empresa haciendo sencillo las distintas funciones de ingresar, modificar, eliminar, consultar e imprimir, e incorporando un apropiado respaldo de la información ingresada al sistema, solucionando de esta manera las principales problemáticas que tenía la organización.

Una de las dificultades que se presentaron en el análisis y diseño fue la normalización de datos, la mala práctica de tener tablas con demasiadas columnas dificulta visualizar los datos que son relevantes. Se logró normalizar y crear las tablas con sus respectivos campos, desintegrando este documento con el fin de obtener datos consistente y sin redundancia.

Por otro lado, uno de los requerimientos más importantes para el cliente era lograr la impresión de las boletas con sus comprobantes de pago. Para lograr esto se requirió hacer todo un trabajo investigativo de cómo calcular el consumo de agua potable con sus respectivos descuentos, cargos, multas, entre otros. Esta fue una de las funcionalidades con mayor dificultad al desarrollar el sistema y en la que se invirtió el mayor tiempo, debido a la complejidad de los cálculos utilizados.

Actualmente la empresa tiene proyecciones para desarrollar e implementar más funcionalidades al sistema, además de crear un sitio web donde cada cliente podrá revisar los estados de cuentas, pagos, informes anuales, entre otros. Esto permitirá una mejor relación entre empresa-cliente y transparencia al momento de los cobros de consumos de agua potable.

Tras meses de arduo trabajo se logró conocer el rubro de esta empresa, lo que facilitó enormemente la compresión de los requerimientos que se solicitaban, adquiriendo tanto experiencia laboral como habilidades blandas. Queda como enseñanza la importancia de saber escuchar al cliente, conocer bien lo que hace la empresa y tener reuniones periódicas

con él para la toma de nuevos requerimientos, y lo importante que es tener la capacidad de adaptarse a las diferentes problemáticas que van apareciendo en el camino e ir aprendiendo de estas a lo largo del desarrollo del sistema.

BIBLIOGRAFÍA

Sistema de Gestión de base de datos MariaDB. (2009). Recuperado de https://mariadb.org/

Entorno de desarrollo integrado Visual Studio.(2015). Recuperado de https://www.visualstudio.com/es/downloads/?rr=https%3A%2F%2Fwww.google.cl%2F

Crear y configurar DataSet mediante una conexión de MySQL por ODBC en Vb.Net. (2012). Recuperado de https://adolfredobelizario.wordpress.com/2012/05/24/crear-y-configurar-dataset-mediante-una-conexion-de-mysql-por-odbc-en-vb-net/

Generación de reportes aleatorios en Visual Basic. (2014). Recuperado de https://www.youtube.com/watch?v=ivvrfk3UEb8

ANEXOS

Código fuente inicio de sesión

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click

    usuario = bsnUsuario.ValidarUsuario(TxtUsuario.Text)
    Dim nombre As String
    Dim contraseña As String

    If TxtUsuario.Text <> "" And TxtContraseña.Text <> "" Then

      If usuario.Rows.Count() = 0 Then
                        MsgBox("El usuario y/o la contraseña es incorrecta.", MsgBoxStyle.Information,
                        "ALERTA")
      Else
        nombre = usuario.Rows(0)(0)
        contraseña = usuario.Rows(0)(1)

                    If nombre <> TxtUsuario.Text Or contraseña <> TxtContraseña.Text Then
                        MsgBox("El usuario y/o la contraseña es incorrecta.", MsgBoxStyle.Information,
                        "ALERTA")
        Else
          FrmMenu.LblU.Text = TxtUsuario.Text
          Me.Hide()
          FrmMenu.Show()
        End If
      End If
    Else
      MsgBox("Llenar campos vacíos.")
    End If


  End Sub
```

Validar usuario

```
Public Function ValidarUsuario(ByVal usuario As String) As DataTable

    Dim command As New MySqlCommand
    Dim dataset As New DataSet

    command.Connection = conexion.GetConexion
    conexion.AbrirConexion()

    command.CommandText = "SELECT * FROM usuario WHERE id_usuario like'%" & usuario & "%'"

    Dim dt As New DataTable
    Dim reader As New MySqlDataAdapter
    reader.SelectCommand = command
    reader.Fill(dataset)

    conexion.CerrarConexion()

    Return dataset.Tables(0)

  End Function
```

Código Menú

```
Private Sub BtnMCliente_Click(sender As Object, e As EventArgs) Handles BtnMCliente.Click

    FrmMantenedorCliente.Show()
    Me.Hide()

End Sub

Private Sub BtnMedidor_Click(sender As Object, e As EventArgs) Handles BtnMedidor.Click

    Me.Hide()
    FrmMedidor.Show()

End Sub

Private Sub BtnBoleta_Click(sender As Object, e As EventArgs) Handles BtnBoleta.Click

    Me.Hide()
    FrmBoleta.Show()

End Sub

Private Sub BtnReuniones_Click(sender As Object, e As EventArgs) Handles BtnReuniones.Click

    Me.Hide()
    FrmAsistencia.Show()

End Sub

Private Sub BtnCargos_Click(sender As Object, e As EventArgs) Handles BtnCargos.Click

    Me.Hide()
    FrmCargos.Show()

End Sub

Private Sub BtnUsuario_Click(sender As Object, e As EventArgs) Handles BtnUsuario.Click

    Me.Hide()
    FrmValidar.Show()

End Sub

Private Sub BtnPago_Click(sender As Object, e As EventArgs) Handles BtnPago.Click

    Me.Hide()
    FrmMantenedorPago.Show()

End Sub

Private Sub CerrarSesiónToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
CerrarSesiónToolStripMenuItem.Click

    Me.Hide()
    frmLogin.Show()

End Sub
```

Código fuente Mantenedor Cliente

En esta parte del código se encuentra en común la funciones agregar, modificar del clientes.

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click
        Dim fono As String
        Dim respuesta As Boolean
        Dim rut As String
        Dim estado As String
        Dim str As String = TxtNombre.Text
        If TxtRut.Text <> String.Empty Then
            rut = TxtRut.Text + TxtDv.Text
            respuesta = bnsCLiente.ValidarRut(rut)
            cliente.Rut = rut
        Else
            Me.FrmMantenedorCliente_Load(Me, Nothing)
            MsgBox("Error, no se permite rut vacio")
            Return

        End If
        rut = cliente.Rut

        cliente.Nombre = str.ToUpper()

        cliente.Direccion = TxtDireccion.Text
        fono = CbCodigo.Text
        fono = fono + TxtFono.Text
        cliente.Fono = fono
        cliente.SubsidioMetrosCubicos = Convert.ToInt32(TxtMC.Text)
        cliente.SubsidioCargoFijo = Convert.ToInt32(TxtCF.Text)

        If CmbEstado.Text.Equals("ACTIVO") Then
            estado = "A"
        Else
            estado = "I"
        End If

        cliente.Estado = estado
        bsnMedidor.CambiarEstadoMedidor(estado, rut)

        If Me.Text = "Modificar Cliente" Then
            If CbCodigo.Text = "32" And TxtFono.Text.Length < 7 Then
                MsgBox("Largo de telefono menor a lo permitido", vbInformation)
                Return
            ElseIf CbCodigo.Text = "+569" And TxtFono.Text.Length < 8 Then
                MsgBox("Largo de telefono menor a lo permitido", vbInformation)
                Return
            End If
            bnsCLiente.ModificarCliente(cliente)
            Me.FrmMantenedorCliente_Load(Me, Nothing)
            bnsCLiente.ReiniciarValores(Me)
        Else
            If respuesta = True And TxtNombre.Text <> String.Empty And TxtDireccion.Text <>
              String.Empty And
                TxtFono.Text <> String.Empty Then
                bnsCLiente.InsertarCLiente(cliente)
                bnsCLiente.ReiniciarValores(Me)
                Me.FrmMantenedorCliente_Load(Me, Nothing)
            Else
                MsgBox("Deben estar todos lo campos llenados.")
            End If
        End If
    End Sub
```

Agregar Cliente:

```
Function InsertarCliente(cliente As Cliente)
```

```
'VARIABLES '
Dim consulta As String

    consulta = "INSERT INTO cliente  VALUES('" & cliente.Rut & "','" & cliente.Nombre & "','" &
    cliente.Direccion & "','" & cliente.Fono & "'," & cliente.SubsidioMetrosCubicos & "," &
    cliente.SubsidioCargoFijo & ",'" & cliente.Estado & "')"

Dim cmd = conexion.Comando(consulta)

Try
   conexion.AbrirConexion()
   Dim r As Integer

   r = cmd.ExecuteNonQuery

   If r > 0 Then

      MessageBox.Show("Cliente agregado exitosamente.")

   End If

Catch ex As Exception

    MessageBox.Show("Error, no se logró agregar al cliente, es posible que ya exista.")

End Try

conexion.CerrarConexion()
Return cliente

End Function
```

Modificar Cliente:

```
Public Function ModificarCliente(cliente As Cliente)

    Dim consulta As String

        consulta = "Update  cliente set Nombre_cliente='" & cliente.Nombre & "',Direccion_cliente='" &
        cliente.Direccion & "',
        Fono = '" & cliente.Fono & "', Porcentaje_subsidio_m = " & cliente.SubsidioMetrosCubicos & ",
        Porcentaje_subsidio_c =" & cliente.SubsidioCargoFijo & ", estado ='" & cliente.Estado & "'
         where Rut_cliente = '" & cliente.Rut & "'"

    Dim cmd = conexion.Comando(consulta)

    Try
       conexion.AbrirConexion()

       Dim r As Integer

       r = cmd.ExecuteNonQuery

       If r > 0 Then

          MessageBox.Show("Cliente modificado exitosamente.")

       End If
    Catch ex As Exception

       MessageBox.Show("Error, no se logró modificar al cliente.")

    End Try

    conexion.CerrarConexion()

    Return cliente
```

```
End Function
```

Eliminar Cliente:

```
Private Sub BtnEliminar_Click(sender As Object, e As EventArgs) Handles BtnEliminar.Click
        Dim index As Integer
        Me.Text = "Eliminar Cliente"
        GpbCLiente.Enabled = False

        If TxtRut.Text <> String.Empty Then

            index = DgvClientes.CurrentRow.Index
              If MessageBox.Show("¿Está seguro de Eliminar al Cliente?", "Alerta",
              MessageBoxButtons.OKCancel, MessageBoxIcon.Question) = DialogResult.OK Then
                cliente.Rut = DgvClientes.Item(0, index).Value
                GpbCLiente.Enabled = True
                Me.FrmMantenedorCliente_Load(Me, Nothing)
                bnsCLiente.EliminarCliente(cliente)
            Else
                'GpbCLiente.Enabled = True
                Me.FrmMantenedorCliente_Load(Me, Nothing)
                bnsCLiente.ReiniciarValores(Me)
            End If
        Else
            MsgBox("Debe seleccionar al cliente que desee eliminar")
            'GpbCLiente.Enabled = True
            Me.FrmMantenedorCliente_Load(Me, Nothing)
        End If

    End Sub



Function EliminarCliente(cliente As Cliente)
    Dim consulta As String
            consulta = "DELETE FROM cliente WHERE Rut_cliente ='" &   cliente.Rut       &  "' "

    Dim cmd = conexion.Comando(consulta)

        Try
            conexion.AbrirConexion()
            Dim r As Integer
            r = cmd.ExecuteNonQuery
            If r > 0 Then
                MessageBox.Show("Cliente eliminado exitosamente.")
            End If
        Catch ex As Exception
                    MessageBox.Show("Error, no se logró eliminar al cliente. El cliente tiene datos asociados a
                    otros archivos. ")

        End Try

        conexion.CerrarConexion()

        Return cliente

    End Function
```

Código fuente Mantenedor Medidor

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click
    If TxtMedidor.Text.Equals("") Or TxtDirMedidor.Text.Equals("") Then
    MsgBox("No se permiten campos vacíos.")
    Else
    If CmbEstado.Text = "ACTIVO" Then

        estado = "A"

     Else

        estado = "I"

     End If

    medidor.Medidor = TxtMedidor.Text
    medidor.Rut = rut
    medidor.Direccion = TxtDirMedidor.Text
    medidor.Estado = estado


     If LblAccion.Text.Equals("agregar") Then

        bsnMedidor.InsertarMedidor(medidor)

     Else

        If LblAccion.Text.Equals("modificar") Then

        bsnMedidor.ModificarMedidor(medidor)

        Else

        If LblAccion.Text.Equals("eliminar") Then

        If MessageBox.Show("¿Está seguro de Eliminar el Medidor?", "ALERTA",
     MessageBoxButtons.OKCancel, MessageBoxIcon.Question) = DialogResult.OK Then

    bsnMedidor.EliminarMedidor(medidor)
    End If

        End If

     End If

    End If
    TxtMedidor.Text = ""
    TxtDirMedidor.Text = ""
    BtnAceptar.Enabled = False
    BtnCancelar.Enabled = False


    TxtMedidor.Enabled = False
    CmbRut.Enabled = True
    TxtDirMedidor.Enabled = False


    BtnAgregar.Enabled = True
    BtnModificar.Enabled = True
    BtnEliminar.Enabled = True


     DgvMedidor.DataSource = bsnMedidor.GetMedidor()

     End If

     End Sub
```

Agregar Medidor:

```
Public Function InsertarMedidor(medidor As Medidor)

    Dim consulta As String

      consulta = “INSERT INTO medidor VALUES(‘” & medidor.Medidor & “’,’” & medidor.Rut &
          “’,’” & medidor.Direccion & “’,’” & medidor.Estado & “’)”

      Dim cmd = conexion.Comando(consulta)

      Try
         conexion.AbrirConexion()
         Dim respuesta As Integer
         respuesta = cmd.ExecuteNonQuery

         If respuesta > 0 Then
            MessageBox.Show(“Medidor agregado exitosamente.”)
         End If

      Catch ex As Exception

         MessageBox.Show(“Error, no se logró agregar el medidor, es posible que ya exista.”)

      End Try

      conexion.CerrarConexion()
      Return medidor

     End Function
```

Modificar Medidor:

```
Public Function ModificarMedidor(medidor As Medidor)

          Dim consulta As String

     consulta = “UPDATE medidor SET Rut_cliente=’” & medidor.Rut & “’,Direccion_medidor=’” &
medidor.Direccion & “’,Estado_medidor=’” & medidor.Estado & “’ WHERE Numero_medidor = ‘” &
medidor.Medidor & “’ “

      Dim cmd = conexion.Comando(consulta)

    Try
    conexion.AbrirConexion()
    Dim respuesta As Integer
    respuesta = cmd.ExecuteNonQuery
    If respuesta > 0 Then

          MessageBox.Show(“Medidor modificado exitosamente.”)

    End If
    Catch ex As Exception
    MessageBox.Show(“Error, no se logró modificar el medidor.”)
    End Try


    conexion.CerrarConexion()
    Return medidor

     End Function
```

Eliminar Medidor:

```
Public Function EliminarMedidor(medidor As Medidor)

    Dim consulta As String

    consulta = “DELETE FROM medidor WHERE Numero_medidor =’” & medidor.Medidor & “’ “

    Dim cmd = conexion.Comando(consulta)

    Try
```

```
conexion.AbrirConexion()
Dim respuesta As Integer
respuesta = cmd.ExecuteNonQuery


 If respuesta > 0 Then

      MessageBox.Show("Medidor eliminado exitosamente.")

 End If

 Catch ex As Exception

    MessageBox.Show("Error, no se logró eliminar el medidor. El medidor tiene datos asociados a
otros archivos.")
 End Try

 conexion.CerrarConexion()
 Return medidor

End Function
```

Consultar Medidor:

```
Private Sub ListadoMedidoresActivosToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
ListadoMedidoresActivosToolStripMenuItem.Click

    DgvMedidor.DataSource = bsnMedidor.GetMedidorActivo()

    End Sub

Function GetMedidorActivo()

    Dim command As New MySqlCommand
    Dim dataset As New DataSet


    command.Connection = conexion.GetConexion()
    conexion.AbrirConexion()


        command.CommandText = "SELECT m.Numero_medidor, m.Rut_cliente, c.Nombre_cliente,
     m.Direccion_medidor, m.Estado_medidor

    FROM medidor m, cliente c
    WHERE m.Rut_cliente = c.Rut_cliente
    AND c.Estado = 'A'
    AND m.Estado_medidor = 'A'"


    Dim reader As New MySqlDataAdapter
    reader.SelectCommand = command
    reader.Fill(dataset)


     conexion.CerrarConexion()

     Return dataset.Tables(0)

     End Function

     Private Sub ListadoMedidoresInactivosToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles ListadoMedidoresInactivosToolStripMenuItem.Click

          DgvMedidor.DataSource = bsnMedidor.GetMedidorInactivo()

     End Sub



    Function GetMedidorInactivo()
```

```
Dim command As New MySqlCommand
Dim dataset As New DataSet


command.Connection = conexion.GetConexion()
conexion.AbrirConexion()


      command.CommandText = “SELECT m.Numero_medidor, m.Rut_cliente, c.Nombre_cliente,
 m.Direccion_medidor, m.Estado_medidor

FROM medidor m, cliente c
WHERE m.Rut_cliente = c.Rut_cliente
AND m.Estado_medidor = ‘I’”


Dim reader As New MySqlDataAdapter

    reader.SelectCommand = command
    reader.Fill(dataset)


      conexion.CerrarConexion()

      Return dataset.Tables(0)

End Function
```

Código fuente Gestor de Boletas

Esta parte del código pertenece a las funciones agregar y modificar dependiendo de los requerimientos.

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click
    Dim EstadoActual, EstadoAnterior As Integer
    Dim estado As New DataTable
    Dim Tcliente, Tpago As New DataTable
    Dim medidor As New DataTable
    Dim Tconsumo As DataTable
    Dim con As New DataTable
    Dim pago As New Pago
    Dim suma As Integer
    Dim CamposValor As DataTable
    If (CmbRut.FindStringExact(CmbRut.Text) > -1) Then
        CmbRut.Enabled = False
    Else
        MsgBox("Error, debe ingresar un Rut de Cliente Valido")
        CmbRut.SelectedIndex = 0
        Return
    End If
    medidor = bsnConsumo.ValidarMedidor(CmbNumMedidor.Text)
    Tconsumo = bsnConsumo.SacarConsumo(CmbNumMedidor.Text)

    If TxtConsumo.Text = "" Or TxtPorton.Text = "" Or CmbNumMedidor.Text = "" Then
        MsgBox("No se permiten campos vacíos.")
    Else
        If Me.Text = "Agregar Consumo" Then
            estado = bsnConsumo.BuscarConsumoAnterior(CmbNumMedidor.Text, hoy)
            EstadoActual = TxtConsumo.Text
            If estado.Rows.Count = 0 Then
                EstadoAnterior = 0
            Else
                EstadoAnterior = estado.Rows(0)(0)
            End If
        ElseIf Me.Text = "Modificar Consumo" Then
            estado = bsnConsumo.BuscarConsumoActual(CmbNumMedidor.Text)
            EstadoActual = TxtConsumo.Text
            EstadoAnterior = estado.Rows(0)(0)
        End If
```

```
        If EstadoActual < EstadoAnterior Then
            MsgBox("El Consumo Actual no puede ser menor que el Consumo Anterior.")
        Else
            con = bsnConsumo.BuscarConsumoActualMes(CmbNumMedidor.Text)
            If con.Rows.Count <> 0 And Me.Text.Equals("Agregar Consumo") Then
                MsgBox("Error, el consumo ya fue ingresado.")
            Else
                If Tconsumo.Rows.Count <> 0 Then
                    Tpago = bsnConsumo.BuscarPago(CmbNumMedidor.Text, Tconsumo.Rows(0)(0))
                    If Tpago.Rows.Count = 0 Then
                        If MessageBox.Show("Error, No existe pago de consumo del mes. anterior. ¿Desea agregar el saldo anterior como deuda actual?", "Alerta", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = DialogResult.Yes Then
                            pago.Fecha = Tconsumo.Rows(0)(1)
                            pago.Medidor = CmbNumMedidor.Text
                            CamposValor = bsnConsumo.CamposValores(pago)
                            For cont As Integer = 0 To 5
                                If cont = 2 Then
                                    suma = suma - CamposValor.Rows(0)(cont)
                                Else
                                    suma += CamposValor.Rows(0)(cont)
                                End If
                            Next
                            pago.Total = suma
                            pago.Abono = 0
                            pago.Saldo = suma
                            bsnPago.InsertarPago(pago)
                        Else
                            Return
                        End If
                    End If
                End If
                consumo = bsnConsumo.CalcularConsumo(EstadoActual, EstadoAnterior, consumo)
                If Label1.Text = 0 Then
                    bsnConsumo.LlenarConsumo(CmbNumMedidor.Text, consumo, rutaux)
                Else
                    MsgBox("No se logro agregar el consumo ")
                End If
                Me.FrmBoleta_Load(Me, Nothing)
            End If
        End If
    End If
    TxtConsumo.Text = ""
    CmbRut.Enabled = True
End Sub

    Public Function CalcularConsumo(EstadoActual As Integer, EstadoAnterior As Integer, Boleta As Consumo)
        Dim TotalConsumo = 0
        Dim consumo = 0
        Dim TablaValor As DataTable
        Dim max As DataTable
        Dim cargo As New CargoVariable
        Dim i = 0
        Dim rango = 0
        Dim cargoFijo As DataTable
        max = daoCargos.GetMaxCargoVariable
        TablaValor = daoCargos.GetCargoVariable
        consumo = EstadoActual - EstadoAnterior
        cargoFijo = daoCargos.GetCargoFijo
        Boleta.Anterior = EstadoAnterior
        Boleta.Actual = EstadoActual
        If EstadoActual < EstadoAnterior Then
            'Return Boleta
        Else
            While consumo > 0
                If consumo > max.Rows(0)(0) Then
                    MsgBox("Error, no existe un rango de cargo variable para el consumo ingresado: " & consumo)
```

```
            consumo = 0
            FrmBoleta.Label1.Text = 1
          Else
            If consumo <= TablaValor.Rows(i)(0) Then
              If consumo = 1 Then
                TotalConsumo = 400
              Else
                consumo = consumo + 1
                TotalConsumo = (consumo - TablaValor.Rows(i - 1)(0)) * TablaValor.Rows(i - 1)(1)
                While i > 1
                  i = i - 1
                  rango = (TablaValor.Rows(i)(0) - TablaValor.Rows(i - 1)(0)) *
TablaValor.Rows(i - 1)(1)
                  TotalConsumo = TotalConsumo + rango
                End While
              End If
              consumo = 0
            End If
          End If
          i = i + 1
        End While
        Boleta.Anterior = EstadoAnterior
        Boleta.Actual = EstadoActual
        Boleta.Rango = EstadoActual - EstadoAnterior
        Boleta.ConsumoAgua = TotalConsumo
        Boleta.CargoFijo = cargoFijo.Rows(0)(1)
        Return Boleta
      End If
    End Function

Public Function LlenarConsumo(medidor As String, consumo As Consumo, rut As String)
      Dim Tconsumo As DataTable
      Dim tcliente As DataTable
      Dim cargofijo As DataTable
      Dim fecha As Date = Now
      Dim mes As String = Format(Month(fecha), "00")
      Dim numero_boleta As Integer
      Dim tmulta As DataTable
      Dim asistencia As New Asistencia
      Dim tdeuda As DataTable
      Dim ultima As DataTable
      Dim AsignacionSubsidio As DataTable
      Dim vMes As String
      ultima = daoConsumo.GetUltimaBoleta()
      tcliente = daocliente.BuscarClienteRut(rut)
      cargofijo = daoCargos.ValorCargoFijo
      If ultima.Rows.Count = 0 Then
        numero_boleta = 1
      Else
        numero_boleta = ultima.Rows(0)(0) + 1
      End If
      consumo.Fecha = fecha
      consumo.Rut = rut
      consumo.Boleta = numero_boleta
      consumo.CargoFijo = cargofijo.Rows(0)(0)
      AsignacionSubsidio = daoConsumo.BuscarSubsidio(rut)
      If AsignacionSubsidio.Rows.Count = 0 Then
        consumo.Subsidio = tcliente.Rows(0)(4) + tcliente.Rows(0)(5)
      Else
        vMes = Format(Month(AsignacionSubsidio.Rows(0)(0)), "00")
        If vMes = mes Then
          consumo.Subsidio = 0
        End If
      End If
      consumo.Medidor = medidor
      consumo.Porton = FrmBoleta.TxtPorton.Text
      tmulta = daoAsistencia.MultasReunion(rut)
      If tmulta.Rows.Count <> 0 Then
        Dim total As Integer
```

```
            total = tmulta.Rows.Count
            For index As Integer = 0 To total - 1
                If tmulta.Rows(index)(4) = "D" Then
                    asistencia.EstadoMulta = "C"
                    asistencia.FechaReunion = tmulta.Rows(index)(0)
                    asistencia.RutCliente = tmulta.Rows(index)(1)
                    consumo.Multa += tmulta.Rows(index)(3)
                    daoAsistencia.ActualizarEstadoMulta(asistencia)
                Else
                    consumo.Multa = 0
                End If
            Next
        End If

        If FrmBoleta.Text = "Agregar Consumo" Then
            Tconsumo = daoConsumo.BuscarConsumoAnterior(medidor, fecha)
            tdeuda = daoPago.BuscarPagoAnterior(medidor, fecha)
            If Tconsumo.Rows.Count() = 0 Then
                consumo.Anterior = 0
            Else
                consumo.Anterior = Tconsumo.Rows(0)(0)
                numero_boleta = Convert.ToInt32(Tconsumo.Rows(0)(2))
            End If
            If tdeuda.Rows.Count() = 0 Then
                consumo.Deuda = 0
            Else
                consumo.Deuda = tdeuda.Rows(0)(0)
            End If
            daoConsumo.InsertarConsumo(consumo)
        ElseIf FrmBoleta.Text = "Modificar Consumo" Then
            Tconsumo = daoConsumo.BuscarConsumoActual(medidor)
            consumo.Anterior = Tconsumo.Rows(0)(0)
            consumo.Fecha = Tconsumo.Rows(0)(2)
            daoConsumo.ActualizarConsumo(consumo)
        End If
        Return 0
End Function
```

Ingresar consumo

```
    Private Sub BtnAgregar_Click(sender As Object, e As EventArgs) Handles BtnAgregar.Click
        GbConsumo.Enabled = True
        BtnModificar.Enabled = False
        Me.Text = "Agregar Consumo"
    End Sub
    Public Function InsertarConsumo(consumo As Consumo)
        Dim consulta As String
        consulta = "INSERT INTO consumo  VALUES('" & Format(consumo.Fecha, "yyyy-MM-dd") &
"','" & consumo.Medidor & "',
                "' & consumo.Rut & "'," & consumo.CargoFijo & "," & consumo.Rango & ",'" &
consumo.Boleta & "'," & consumo.Anterior & ",
                " & consumo.Actual & "," & consumo.ConsumoAgua & "," & consumo.Subsidio & ","
& consumo.Multa & ",
                " & consumo.Porton & "," & consumo.Deuda & ")"
        Dim cmd = conexion.Comando(consulta)
        Try
            conexion.AbrirConexion()
            Dim r As Integer
            r = cmd.ExecuteNonQuery
            If r > 0 Then
                MessageBox.Show("Consumo agregado exitosamente.")
            End If
        Catch ex As Exception
            MessageBox.Show("Error, no se logró agregar el consumo, es posible que ya exista.")
        End Try
        conexion.CerrarConexion()
        Return consumo
    End Function
```

Modificar Consumo

```
Private Sub BtnModificar_Click(sender As Object, e As EventArgs) Handles BtnModificar.Click
    GbConsumo.Enabled = True
    BtnAgregar.Enabled = False
    CmbRut.Enabled = False
    CmbNumMedidor.Enabled = False
    Me.Text = "Modificar Consumo"
  End Sub

Public Function ActualizarConsumo(consumo As Consumo)
    Dim consulta As String
    consulta = "Update  consumo set rango=" & consumo.Rango & ", estado_anterior =" & consumo.Anterior & "
            , estado_actual= " & consumo.Actual & " , consumo_agua=" & consumo.ConsumoAgua & "
            where numero_medidor='" & consumo.Medidor & "' and dia_mes_anno= '" & Format(consumo.Fecha, "yyyy-MM-dd") & "' "
    Dim cmd = conexion.Comando(consulta)
    Try
      conexion.AbrirConexion()
      Dim r As Integer
      r = cmd.ExecuteNonQuery
      If r > 0 Then
        MessageBox.Show("Consumo modificado exitosamente.")
      End If
    Catch ex As Exception
      MessageBox.Show("Error, no se logró modificar el consumo.")
    End Try
    conexion.CerrarConexion()
    Return consumo
  End Function
```

Imprimir boletas

```
Private Sub BtnImprimir_Click(sender As Object, e As EventArgs) Handles BtnImprimir.Click
    cap_santa_juliaDataSet.EnforceConstraints = False
    Me.DataTableTableAdapter.Fill(Me.cap_santa_juliaDataSet.DataTable, rutaux, rutaux, CmbNumMedidor.Text)
    Me.RpBoleta.RefreshReport()
  End Sub
```

Imprimir Resumen de Boletas

```
Private Sub ListadoDeBoletasToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles ListadoDeBoletasToolStripMenuItem.Click
    Dim listado As New FrmListaBoleta
    Me.Hide()
    listado.ShowDialog()
  End Sub
```

Consultar consumo

```
Private Sub TxtBuscar_TextChanged(sender As Object, e As EventArgs) Handles TxtBuscar.TextChanged
    If RbBuscarRut.Checked = True Then
      If bsnConsumo.BuscarRut(TxtBuscar.Text).Rows.Count > 0 Then
        DgvConsumo.DataSource = bsnConsumo.BuscarRut(TxtBuscar.Text)
      End If
    ElseIf RbBuscarNombre.Checked = True Then
      If bsnConsumo.BuscarNombre(TxtBuscar.Text).Rows.Count > 0 Then
        DgvConsumo.DataSource = bsnConsumo.BuscarNombre(TxtBuscar.Text)
      End If
    End If
    TxtConsumo.Text = ""
  End Sub
```

```
    Private Sub ConsultarConsumoToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles ConsultarConsumoToolStripMenuItem.Click
        FrmConsultar.LblOpcion.Text = "CONSUMOS"
        FrmConsultar.Show()
        Me.Dispose()
    End Sub

Public Function BuscarPorFecha(fecha As Date)
        Dim command As New MySqlCommand
        Dim dataset As New DataSet
        Dim vMes As String = fecha.Month
        Dim vAño As String = fecha.Year
        Dim vDia As String
        Dim fechaInicio As Date
        Dim fechaTermino As Date
        vDia = DateTime.DaysInMonth(Convert.ToInt32(vAño), Convert.ToInt32(vMes))
        fechaInicio = 1 & "-" & vMes & "-" & vAño
        fechaTermino = vDia & "-" & vMes & "-" & vAño
        command.Connection = conexion.GetConexion()
        conexion.AbrirConexion()
        command.CommandText = "Select con.dia_mes_anno, con.numero_medidor, con.rut_cliente,
c.Nombre_cliente, con.cargo_fijo_m, con.rango,
                        con.numero_boleta, con.estado_anterior, con.estado_actual, con.consumo_agua,
con.subsidio, con.multas,
                        con.cuota_porton, con.deuda_acumulada
                        FROM consumo con, cliente c
                        WHERE dia_mes_anno BETWEEN '" & Format(fechaInicio, "yyyy-MM-dd") &
"' AND '" & Format(fechaTermino, "yyyy-MM-dd") & "'
                        AND c.Rut_cliente = con.Rut_Cliente"
        Dim reader As New MySqlDataAdapter
        reader.SelectCommand = command
        reader.Fill(dataset)
        conexion.CerrarConexion()
        Return dataset.Tables(0)
    End Function

Public Function BuscarConsumoRut(ByVal Rut As String) As DataTable
        Dim command As New MySqlCommand
        Dim dataset As New DataSet
        command.Connection = conexion.GetConexion()
        conexion.AbrirConexion()
        command.CommandText = "SELECT con.dia_mes_anno, con.numero_medidor, con.rut_cliente,
c.Nombre_cliente, con.cargo_fijo_m, con.rango,
                        con.numero_boleta, con.estado_anterior, con.estado_actual, con.consumo_agua,
con.subsidio, con.multas,
                        con.cuota_porton, con.deuda_acumulada
                        FROM consumo con, cliente c
                        WHERE con.Rut_cliente = c.Rut_cliente
                        AND con.Rut_cliente LIKE '%" & Rut & "%' "

        Dim dt As New DataTable
        Dim reader As New MySqlDataAdapter
        reader.SelectCommand = command
        reader.Fill(dataset)

        conexion.CerrarConexion()

        Return dataset.Tables(0)
    End Function
Public Function BuscarNombre(ByVal busqueda As String) As DataTable
        Dim command As New MySqlCommand
        Dim dataset As New DataSet
        Dim estado As String = "A"
        command.Connection = conexion.GetConexion()
        conexion.AbrirConexion()
        command.CommandText = "SELECT con.dia_mes_anno, con.numero_medidor, con.rut_cliente,
c.Nombre_cliente, con.cargo_fijo_m, con.rango,
```

```
                        con.numero_boleta, con.estado_anterior, con.estado_actual, con.consumo_agua,
    con.subsidio, con.multas,
                        con.cuota_porton, con.deuda_acumulada
                        FROM consumo con, cliente c
                        WHERE con.Rut_cliente = c.Rut_cliente and c.estado="' & estado & "'
                        AND dia_mes_anno=(SELECT MAX(dia_mes_anno) FROM consumo WHERE
    numero_medidor = con.numero_medidor)
                        AND c.Nombre_cliente LIKE '%" & busqueda & "%"'
        Dim dt As New DataTable
        Dim reader As New MySqlDataAdapter
        reader.SelectCommand = command
        reader.Fill(dataset)

        conexion.CerrarConexion()

        Return dataset.Tables(0)
      End Function
```

Código Fuente Mantenedor Asistencia Reunión

Agregar Asistencia

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click

    Dim rut As String
    Dim pos As Integer

    If (CmbRut.FindStringExact(CmbRut.Text) > -1) Then
       CmbRut.Enabled = False
    Else
       MsgBox("Error, debe ingresar un Rut de Cliente Valido")
       CmbRut.SelectedIndex = 0
       Return
    End If

    rut = CmbRut.Text
    pos = InStr(rut, " ")
    rut = rut.Substring(0, pos - 1)
    bsnAsistencia.InsertarAsistencia(rut)
    CmbRut.Enabled = True

  End Sub

Function InsertarAsistencia(rut As String)
    Dim asistencia As New Asistencia

    If FrmAsistencia.RbSi.Checked = True Then
       asistencia.FechaReunion = FrmAsistencia.DtpFecha.Value
       asistencia.RutCliente = rut
       asistencia.AsistenciaReunion = "S"
       asistencia.MultaMes = 0
       asistencia.EstadoMulta = "C"

    Else
       asistencia.FechaReunion = FrmAsistencia.DtpFecha.Value
       asistencia.RutCliente = rut
       asistencia.AsistenciaReunion = "N"
       asistencia.MultaMes = FrmAsistencia.TxtMulta.Text
       asistencia.EstadoMulta = "D"

    End If

    If FrmAsistencia.Text = "AGREGAR ASISTENCIA" Then

       For i As Integer = FrmAsistencia.DgvReunion.SelectedRows.Count - 1 To 0 Step -1
```

```
            FrmAsistencia.DgvReunion.Rows.RemoveAt(FrmAsistencia.DgvReunion.SelectedRows(i).Index)
        Next
        daoAsistencia.InsertarAsistencia(asistencia)

    End If

    FrmAsistencia.DgvAsistencia.DataSource = GetAsistencia()

    Return 0

  End Function

Function InsertarAsistencia(asistencia As Asistencia)

    Dim consulta As String
    consulta = "INSERT INTO asistencia  VALUES('" & Format(asistencia.FechaReunion, "yyyy-
        MM-dd") & "','" & asistencia.RutCliente & "',
                                '" & asistencia.AsistenciaReunion & "'," & asistencia.MultaMes & ",'" &
asistencia.EstadoMulta & "')"

    Dim cmd = conexion.Comando(consulta)

    Try
      conexion.AbrirConexion()
      Dim r As Integer
      r = cmd.ExecuteNonQuery
      If r > 0 Then
        MessageBox.Show("Se registro la asistencia exitosamente.")
      End If
    Catch ex As Exception
      MessageBox.Show("Error, no se logró registrar la asistencia, es posible que ya exista.")

    End Try

    conexion.CerrarConexion()
    Return asistencia

  End Function
```

Modificar Asistencia

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click
    Dim rut As String
    Dim pos As Integer

    If (CmbRut.FindStringExact(CmbRut.Text) > -1) Then
      CmbRut.Enabled = False
    Else
      MsgBox("Error, debe ingresar un Rut de Cliente Valido")
      CmbRut.SelectedIndex = 0
      Return
    End If

    rut = CmbRut.Text
    pos = InStr(rut, " ")
    rut = rut.Substring(0, pos - 1)
    bsnAsistencia.ModificarAsistencia(rut)
    Me.FrmModiAsistencia_Load(Me, Nothing)

  End Sub

Function ModificarAsistencia(rut As String)

    Dim asistencia As New Asistencia

    If FrmModiAsistencia.RbSi.Checked = True Then
```

```
            asistencia.FechaReunion = FrmAsistencia.DtpFecha.Value
            asistencia.RutCliente = rut
            asistencia.AsistenciaReunion = "S"
            asistencia.MultaMes = 0
            asistencia.EstadoMulta = "C"

        Else
            asistencia.FechaReunion = FrmAsistencia.DtpFecha.Value
            asistencia.RutCliente = rut
            asistencia.AsistenciaReunion = "N"
            asistencia.MultaMes = FrmModiAsistencia.TxtMulta.Text
            asistencia.EstadoMulta = "D"

        End If
        daoAsistencia.ActualizarAsistencia(asistencia)
        Return 0

    End Function

Function ActualizarAsistencia(asistencia As Asistencia)

        Dim consulta As String
            consulta = "UPDATE asistencia SET Asistencia_reunion = '" & asistencia.AsistenciaReunion &
                    "', Multas_mes = " & asistencia.MultaMes & ", Estado_multa= '" & asistencia.EstadoMulta
            & "'
                WHERE Fecha_reunion = '" & Format(asistencia.FechaReunion, "yyyy-MM-dd") & "'          and
Rut_cliente = '" & asistencia.RutCliente & "'"

        Dim cmd = conexion.Comando(consulta)

        Try
            conexion.AbrirConexion()
            Dim r As Integer
            r = cmd.ExecuteNonQuery

            If r > 0 Then
                MessageBox.Show("Estado asistencia modificado exitosamente.")
            End If

        Catch ex As Exception
            MessageBox.Show("Error, no se logró modificar el estado asistencia.")

        End Try

        conexion.CerrarConexion()
        Return asistencia

    End Function
```

Codigo fuente Mantenedor de Cargos

Codigo fuente común entre cargo fijo y varibable.

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click

        If TxtRango.Text = 0 Or TxtValor.Text = 0 Then
            MsgBox("Error, no se permiten valores 0")
        Else

            If TxtRango.Text.Equals("") Or TxtValor.Text.Equals("") Then
                MsgBox("Campos vacios.")
            Else
                cargoVariable.Rango = Convert.ToInt32(TxtRango.Text)
                cargoVariable.ValorMetroCubico = Convert.ToInt32(TxtValor.Text)

                cargoFijo.CargoFijoM = Convert.ToInt32(TxtRango.Text)
                cargoFijo.ValorCargoFijo = Convert.ToInt32(TxtValor.Text)
```

```
            If Opcion.Equals("AgregarV") Then
               bsnCargos.InsertarCargoVariable(cargoVariable)
            Else
               If Opcion.Equals("ModificarV") Then
                  bsnCargos.ModificarCargoVariable(cargoVariable)
               Else
                  If Opcion.Equals("EliminarV") Then
                     If TxtRango.Text = 1 Then
                        MsgBox("No se puede eliminar el cargo variable minimo")
                     Else
                        If MessageBox.Show("¿ Está seguro de Eliminar el Cargo Variable?", "Alerta",
                        MessageBoxButtons.OKCancel, MessageBoxIcon.Question) = DialogResult.OK Then
                           bsnCargos.EliminarCargoVariable(cargoVariable)
                        End If

                     End If

                  Else
                     If Opcion.Equals("ModificarF") Then
                        bsnCargos.ModificarCargoFijo(cargoFijo)
                     End If
                  End If
               End If
            End If
         End If
      End If
      Me.FrmCargos_Load(Me, Nothing)
      TxtRango.Text = ""
      TxtValor.Text = ""
      TxtRango.Enabled = False
      TxtValor.Enabled = False
      BtnAceptar.Enabled = False

   End Sub
```

Codigo fuente Mantenedor de Cargo Variable

Agregar Cargo Variable

```
Public Function InsertarCargoVariable(cargoVariable As CargoVariable)
      Dim cargoV As String
      cargoV = "INSERT INTO cargo_variable VALUES(" & cargoVariable.Rango & "," &
cargoVariable.ValorMetroCubico & ")"

      Dim cmd = conexion.Comando(cargoV)

      Try
         conexion.AbrirConexion()
         Dim respuesta As Integer
         respuesta = cmd.ExecuteNonQuery

         If respuesta > 0 Then
            MessageBox.Show("El cargo variable fue agregado exitosamente.")
         End If
      Catch ex As Exception
         MessageBox.Show("Error, no se logró agregar el cargo variable.")
      End Try

      conexion.CerrarConexion()

      Return cargoVariable
   End Function
```

Modificar Cargo Variable

```
   Public Function ModificarCargoVariable(cargoVariable As CargoVariable)
      Dim cargoV As String
      cargoV = "UPDATE cargo_variable SET valor_metro_cubico=" & cargoVariable.ValorMetroCubico & "
WHERE rango = " & cargoVariable.Rango & " "

      Dim cmd = conexion.Comando(cargoV)

      Try
         conexion.AbrirConexion()
         Dim respuesta As Integer
         respuesta = cmd.ExecuteNonQuery
         If respuesta > 0 Then
            MessageBox.Show("El cargo variable fue modificado exitosamente.")
         End If
      Catch ex As Exception
         MessageBox.Show("Error, no se logró modificar el cargo variable.")
      End Try

      conexion.CerrarConexion()
      Return cargoVariable
   End Function
```

Eliminar Cargo Variable

```
   Public Function EliminarCargoVariable(cargoVariable As CargoVariable)
      Dim cargoV As String
      cargoV = "DELETE FROM cargo_variable WHERE rango=" & cargoVariable.Rango & " "
      Dim cmd = conexion.Comando(cargoV)

      Try
         conexion.AbrirConexion()
         Dim respuesta As Integer
         respuesta = cmd.ExecuteNonQuery
         If respuesta > 0 Then
            MessageBox.Show("El cargo variable fue eliminado correctamente.")
         End If
      Catch ex As Exception
         MessageBox.Show("Error, el cargo variable no se logró eliminar.")
      End Try

      conexion.CerrarConexion()
      Return cargoVariable
   End Function
```

Codigo fuente Mantenedor de Cargo Fijo

Modificar Cargo Fijo

```
Public Function ModificarCargoFijo(cargoFijo As CargoFijo)
      Dim cargoF As String
      cargoF = "UPDATE cargo_fijo SET valor_cargo_fijo=" & cargoFijo.ValorCargoFijo & " WHERE
cargo_fijo_m= " & cargoFijo.CargoFijoM & " "

      Dim cmd = conexion.Comando(cargoF)

      Try
         conexion.AbrirConexion()
         Dim respuesta As Integer
         respuesta = cmd.ExecuteNonQuery
         If respuesta > 0 Then
            MessageBox.Show("Valor modificado.")
         End If
      Catch ex As Exception
         MessageBox.Show("El valor no se logró modiifcar ")
```

```
    End Try

    conexion.CerrarConexion()
    Return cargoFijo

End Function
```

Mantenedor de Usuario

Codigo en común con agregar, modificar e eliminar usuario.

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click

    actual = bsnUsuario.ValidarUsuario(TxtIdU.Text)
    Dim contraseña As String

    If TxtNombreU.Text.Equals("") Or TxtIdU.Text.Equals("") Then
       MsgBox("No se permiten campos vacíos.")
    Else
       usuario.ID_USUARIO = TxtIdU.Text
       If LblOpcion.Text.Equals("ELIMINAR.") Then
          If TxtIdU.Text.Equals(Label4.Text) Then
             MsgBox("No puede ser eliminado.")
          Else
             bsnUsuario.EliminarUsuario(usuario)
             sw = 1
          End If
       Else
          If TxtContraseña.Text.Equals("") Or TxtRepetir.Text.Equals("") Then
             MsgBox("No se permiten campos vacíos.")
          Else
             If TxtContraseña.Text.Equals(TxtRepetir.Text) Then
                usuario.NombreUsuario = TxtNombreU.Text
                usuario.Contrasenna = TxtContraseña.Text
                usuario.Nivel = Convert.ToInt16(CmbNivel.Text)

                If LblOpcion.Text.Equals("AGREGAR.") Then
                   bsnUsuario.InsertarUsuario(usuario)
                   sw = 1
                Else
                   If LblOpcion.Text.Equals("MODIFICAR.") Then
                      If TxtActual.Text.Equals("") Then
                         MsgBox("Debe ingresar la contraseña actual.")
                      Else
                         If actual.Rows.Count() = 0 Then
                                                MsgBox("Error, la contraseña actual es incorrecta.",
                            MsgBoxStyle.Information, "ALERTA")
                           TxtActual.Text = ""
                         Else
                           contraseña = actual.Rows(0)(1)
                           If contraseña <> TxtActual.Text Then
                                                  MsgBox("Error, la contraseña actual es incorrecta.",
                             MsgBoxStyle.Information, "ALERTA")
                              TxtActual.Text = ""
                           Else
                              bsnUsuario.ModificarUsuario(usuario)
                              sw = 1
                           End If
                         End If
                      End If
                   End If
                End If
             Else
                MsgBox("Error al ingresar la contraseña.")
                TxtContraseña.Text = ""
                TxtRepetir.Text = ""
```

```
            End If
        End If

    End If
End If

If sw = 1 Then
    If Label5.Text = 3 Then
        TxtContraseña.Text = ""
        TxtRepetir.Text = ""
        TxtNombreU.Text = ""
        TxtActual.Text = ""

        TxtNombreU.Enabled = False
        TxtIdU.Enabled = False
        TxtContraseña.Enabled = False
        TxtRepetir.Enabled = False
        BtnAceptar.Enabled = False

        CmbNivel.Enabled = False
        TxtActual.Enabled = False
        BtnModificar.Enabled = True
    Else
        TxtContraseña.Text = ""
        TxtRepetir.Text = ""
        TxtNombreU.Text = ""
        CmbNivel.SelectedIndex = 0

        TxtIdU.Text = ""
        TxtActual.Text = ""
        TxtNombreU.Enabled = False
        TxtIdU.Enabled = False
        TxtContraseña.Enabled = False
        TxtRepetir.Enabled = False
        BtnAceptar.Enabled = False

        CmbNivel.Enabled = False
        TxtActual.Enabled = False
        DgvUsuario.Enabled = True
        BtnAgregar.Enabled = True
        BtnModificar.Enabled = True
        BtnEliminar.Enabled = True
    End If

    Me.FrmMantenedorUsuario_Load(Me, Nothing)
End If

End Sub
```

Validar Usuario

```
Public Function ValidarUsuario(ByVal usuario As String) As DataTable
    Dim command As New MySqlCommand
    Dim dataset As New DataSet

    command.Connection = conexion.GetConexion
    conexion.AbrirConexion()
    command.CommandText = "SELECT * FROM usuario WHERE id_usuario like'%" & usuario & "%'"

    Dim dt As New DataTable
    Dim reader As New MySqlDataAdapter
    reader.SelectCommand = command
    reader.Fill(dataset)

    conexion.CerrarConexion()
    Return dataset.Tables(0)
End Function
```

Agregar Usuario

```
   Public Function InsertarUsuario(usuario As Usuario)
      Dim consulta As String
      consulta = "INSERT INTO usuario VALUES('" & usuario.ID_USUARIO & "','" & usuario.Contrasenna
& "','" & usuario.NombreUsuario & "'," & usuario.Nivel & ")"
      Dim cmd = conexion.Comando(consulta)

      Try
         conexion.AbrirConexion()
         Dim r As Integer
         r = cmd.ExecuteNonQuery
         If r > 0 Then
            MessageBox.Show("Usuario agregado exitosamente. ")
         End If
      Catch ex As Exception
         MessageBox.Show("No se logró agregar el usuario.")

      End Try

      conexion.CerrarConexion()
      Return usuario
   End Function
```

Modificar Usuario

```
   Public Function ModificarUsuario(usuario As Usuario)
      Dim consulta As String
      consulta = "UPDATE usuario SET nom_usuario='" & usuario.NombreUsuario & "', contrasena='" &
usuario.Contrasenna & "', nivel= " & usuario.Nivel & " WHERE id_usuario= '" & usuario.ID_USUARIO & "'
"

      Dim cmd = conexion.Comando(consulta)

      Try
         conexion.AbrirConexion()
         Dim respuesta As Integer
         respuesta = cmd.ExecuteNonQuery
         If respuesta > 0 Then
            MessageBox.Show("Usuario modificado exitosamente.")
         End If
      Catch ex As Exception
         MessageBox.Show("Error al modificar al Usuario.")
      End Try

      conexion.CerrarConexion()
      Return usuario
   End Function
```

Eliminar Usuario

```
Public Function EliminarUsuario(usuario As Usuario)
      Dim consulta As String
      consulta = "DELETE FROM usuario WHERE id_usuario ='" & usuario.ID_USUARIO & "' "
      Dim cmd = conexion.Comando(consulta)

      Try
         conexion.AbrirConexion()
         Dim respuesta As Integer
         respuesta = cmd.ExecuteNonQuery
         If respuesta > 0 Then
            MessageBox.Show("Usuario eliminado exitosamente")
         End If
      Catch ex As Exception
         MessageBox.Show("Error al eliminar al Usuario.")
      End Try
```

```
    conexion.CerrarConexion()
    Return usuario
End Function
```

Código fuente Mantenedor de Pago

Ingresar Pago

```
Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click
    Dim pago As New Pago
    If (CmbRut.FindStringExact(CmbRut.Text) > -1) Then
        CmbRut.Enabled = False
    Else
        MsgBox("Error, debe ingresar un Rut de Cliente Valido")
        CmbRut.SelectedIndex = 0
        Return
    End If
    bsnPago.LlenarPago(rut, CmbNumMedidor.Text)
    TxtAbono.Text = ""
    CmbRut.Enabled = True
End Sub
Public Function InsertarPago(pago As Pago)
    Dim cupon As DataTable
    cupon = daoConsumo.BuscarCupon(pago.Medidor)
    pago.Cupon = cupon.Rows(0)(0)
    Return daoPago.InsertarPago(pago)
End Function
Function LlenarPago(rut As String, medidor As String)
    Dim pago As New Pago
    'Dim Tpago As DataTable
    Dim consumo As DataTable
    Dim CamposValor As DataTable
    Dim Tencontre As DataTable
    ' Dim totalcolumnas As Integer
    Dim suma As Integer
    Dim cupon As DataTable
    consumo = daoConsumo.PagarConsumo(rut, medidor)
    If consumo.Rows.Count <> 0 Then
        pago.Fecha = consumo.Rows(0)(0)
        pago.Medidor = medidor
        CamposValor = daoConsumo.CamposValores(pago)
        'totalcolumnas = CamposValor.Rows.Count

        For cont As Integer = 0 To 5
            If cont = 2 Then
                suma = suma - CamposValor.Rows(0)(cont)
            Else
                suma += CamposValor.Rows(0)(cont)
            End If

        Next
        'Tpago = daoPago.NumeroPago
        pago.Total = suma

        'suma = Convert.ToInt32(Tpago.Rows(0)(0)) + 1

        cupon = daoConsumo.BuscarCupon(medidor)
        pago.Cupon = cupon.Rows(0)(0)
        pago.Abono = Convert.ToInt32(FrmMantenedorPago.TxtAbono.Text)
        saldo = pago.Total - pago.Abono
        If saldo < 0 Then
            saldo = saldo * -1
            MsgBox("Vuelto para el cliente: " & saldo)
        Else
            pago.Saldo = saldo
        End If
        Tencontre = daoPago.BuscarPago(medidor)
```

```
            If Tencontre.Rows.Count = 0 Then
                daoPago.InsertarPago(pago)
            Else
                MsgBox("Error, el consumo ya fue pagado.")
            End If
        Else
            MsgBox("Error, Este Cliente no registra consumo de agua")
        End If
        Return pago
    End Function

Public Function InsertarPago(pago As Pago)
        Dim consulta As String
        consulta = "INSERT INTO pago VALUES('" & Format(pago.Fecha, "yyyy-MM-dd") & "','" &
    pago.Medidor & "','" & pago.Cupon & "'," & pago.Total & "," & pago.Abono & "," & pago.Saldo &
    ")"
        Dim cmd = conexion.Comando(consulta)
        Try
            conexion.AbrirConexion()
            Dim r As Integer
            r = cmd.ExecuteNonQuery
            If r > 0 Then
                MessageBox.Show("Pago agregado exitosamente.")
            End If
        Catch ex As Exception
            MessageBox.Show("Error, no se logró agregar el pago, es posible que ya exista. ")
        End Try
        conexion.CerrarConexion()
        Return pago
    End Function
```

Imprimir comprobante de pago

```
    Private Sub BtnImprimir_Click(sender As Object, e As EventArgs) Handles BtnImprimir.Click
        cap_santa_juliaDataSet.EnforceConstraints = False
        Me.DataTable1TableAdapter.Fill(Me.cap_santa_juliaDataSet.DataTable1,
    CmbNumMedidor.Text, CmbNumMedidor.Text, rut)
        Me.RvComprobante.RefreshReport()
    End Sub
```

Consultar pago

```
    Private Sub TxtBuscar_TextChanged(sender As Object, e As EventArgs) Handles
    TxtBuscar.TextChanged
        If RbBuscarRut.Checked = True Then
            If bsnConsumo.BuscarRut(TxtBuscar.Text).Rows.Count > 0 Then
                DgvPago.DataSource = bsnPago.BuscarRut(TxtBuscar.Text)
            End If
        ElseIf RbBuscarNombre.Checked = True Then
            If bsnConsumo.BuscarNombre(TxtBuscar.Text).Rows.Count > 0 Then
                DgvPago.DataSource = bsnPago.BuscarNombre(TxtBuscar.Text)
            End If
        End If
    End Sub

    Private Sub ConsultarPagoToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
    ConsultarPagoToolStripMenuItem.Click
        FrmConsultar.LblOpcion.Text = "PAGO"
        FrmConsultar.Show()
        Me.Dispose()
    End Sub
```