

Instituto Tecnológico de Ciudad Juárez
TecNM

Talle de Bases de Datos

Panadería Santa Mónica - Proyecto Final

Alumno

Romero Vidales Jesús Ángel

Salazar Guerrero David Armando

Saucedo Martínez Francisco Daniel

Torres Santos José Ángel

Vázquez Hernández Esteban

Docente

Juan Manuel Bernal Ontiveros

Ciudad Juárez, Chih., a 31 de mayo de 2021

Índice

Introducción	3
Definición del problema.....	4
Objetivo.....	4
Justificación	4
SQLite.....	5
¿Qué es SQLite?	5
DB Browser for SQLite.....	5
¿Qué es?.....	5
¿Que NO es?.....	5
¿Porque usarlo?	6
Características.....	6
Diagrama del Modelo Relacional.....	3
Implementación de la Base de datos	3
Estructura de la tabla <i>Empleados</i>	3
Estructura de la tabla <i>Sucursal</i>	4
Estructura de la tabla <i>Pedidos</i>	5
Estructura de la tabla <i>Productos</i>	6
Estructura de la tabla <i>Ventas</i>	7
Estructura de la tabla <i>Proveedores</i>	8
Alta de registros en la base de datos	9
Registros en tabla <i>Empleados</i>	9
Registros en tabla <i>Sucursal</i>	10
Registros en tabla <i>Productos</i>	11
Registros en tabla <i>Ventas</i>	12
Registros en tabla <i>Proveedores</i>	13
Registros en tabla <i>Pedidos</i>	14
Referencia SQLite.....	15
Descarga de la Referencia.....	15
Agregando la referencia a Visual Basic.NET	18
CRUD como base de la gestión de datos.....	23

CRUD en Visual Basic.NET	24
Estructurando el Módulo CRUDPan.....	25
Código fuente CRUDPan	26
Windows Forms Principales	28
Formulario 1: Form bienvenida.vb.....	28
Código fuente Form bienvenida.vb	29
Formulario 2: Form sistemaprincipal.vb	30
Código fuente Form sistemaprincipal.vb	30
Formulario 3: Form administracion.vb.....	32
Código fuente Form administracion.vb	33
Notas Importantes	34
Implementando las Sentencias SQL (CRUD).....	35
Formulario 4: Form empleados.vb	35
Formulario 5: Form proveedores.vb	37
Formulario 6: Form sucursales.vb	39
Formulario 7: Form productos.vb	41
Formulario 8: Form pedidos.vb	43
Formulario 9: Form ventas.vb	45
Windows Forms Material Design: Sistema Administración	47
Formulario 4: Form empleados.vb	47
Formulario 5: Form proveedores.vb	47
Formulario 6: Form sucursales.vb	48
Windows Forms Material Design: Sistema Principal.....	49
Formulario 7: Form productos.vb	49
Formulario 8: Form pedidos.vb	49
Formulario 9: Form ventas.vb	50

Introducción

SQLite es una herramienta de software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser una PDA o un teléfono celular. Pese a su evidente simplicidad, es increíble como este gestor de base de datos puede soportar tanto las consultas más básicas hasta las más complejas que podremos encontrar dentro de SQL.

La evolución de la informática nos ha proporciona sistemas que cada vez realizan más funciones facilitando en gran medida el trabajo, gracias a esta evolución podemos crear programas los cuales nos ayudan a almacenar grandes cantidades de datos en diversas bases de datos, con el objetivo de contar con diversos registros como los de: alumnos inscritos en una escuela, productos en un almacén, trabajadores de una determinada empresa, entre otros.

El objetivo de este proyecto es poder establecer una conexión entre el programa realizado en Visual Studio y la base de datos realizada en SQLite, con el propósito de que los datos manejados en el programa de Visual Studio puedan ser almacenados de una manera fácil y sencilla en la base de datos de SQLite.

Definición del problema

El presente trabajo pretende responder y aportar información a la comunidad educativa en relación a la siguiente pregunta: ¿Cómo establecer una conexión en SQLite mediante Visual Studio utilizando el lenguaje de programación Visual Basic? La pregunta de investigación planteada busca la relación entre las siguientes dos variables: 1) Por qué usar SQLite para integrarlo a un proyecto y 2) Porque este es importante.

La demanda de bases de datos para aplicaciones de escritorio y teléfonos celulares ha crecido exponencialmente en los últimos años debido a la necesidad de las empresas de tener la información al instante de lo que sucede en el campo y así responder más rápidamente ante la competencia. Esta necesidad ha provocado que el almacenamiento de los datos en estos softwares haya mejorado tanto en capacidad como en herramientas. Gracias a esto, actualmente contamos con diversas opciones de manejadores de bases de datos para aplicaciones de escritorio, y una de ellas es SQLite, que es en la que se enfoca este proyecto.

Objetivo

El objetivo de este proyecto es poder establecer una conexión entre el programa realizado en Visual Studio y la base de datos realizada en SQLite, con el propósito de que los datos manejados en el programa de Visual Studio puedan ser almacenados de una manera fácil y sencilla en la base de datos de SQLite.

Justificación

Un sistema de información es un conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su posterior uso, generados para cubrir una necesidad (objetivo). Por ello, es que se hace necesario para la panadería **Santa Mónica** crear un sistema de información para el pleno desarrollo y crecimiento, donde se trate y se administre bien la información de esta panadería.

SQLite

¿Qué es SQLite?

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB) biblioteca escrita en C. SQLite es un proyecto de dominio público.

En su versión 3 (.sqlite3), SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

DB Browser for SQLite

¿Qué es?

- DB Browser for SQLite (DB4S) es una herramienta visual de código abierto para crear, diseñar y editar archivos de bases de datos compatibles con SQLite.
- DB4S es para usuarios y desarrolladores que desean crear, buscar y editar bases de datos SQLite.
- DB4S utiliza una interfaz similar a una hoja de cálculo, y no es necesario aprender comandos complicados de SQL para poder desenvolverse correctamente bajo DB Browser for SQLite.

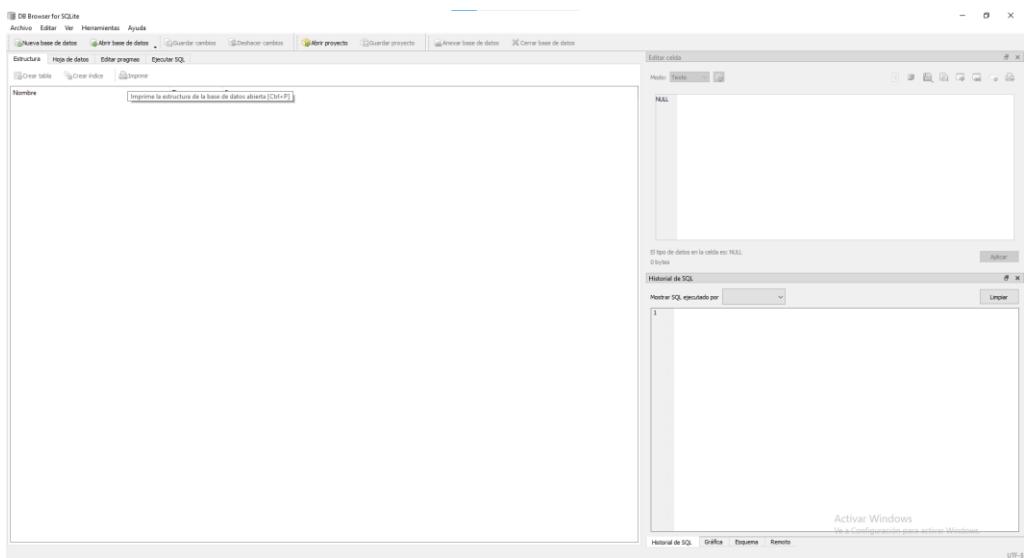
¿Que NO es?

- DB Browser for SQLite (DB4S) no es una herramienta para la introducción de líneas de comandos en base de datos SQLite.
- El programa no requiere de conocimientos en comandos SQL. Es una herramienta para ser utilizada tanto por los desarrolladores como por los usuarios finales, y debe ser tan simple de usar como sea posible para lograr los objetivos deseados.

¿Porque usarlo?

Usaremos el DB4SQLite, porque este software nos facilitara la creación y administración de las bases de datos creadas con SQLite, gracias a este software reduce la probabilidad de que algo salga mal y haga que nuestra base de datos deje de funcionar correctamente.

Funciona con una interfaz muy clara y sencilla de utilizar:



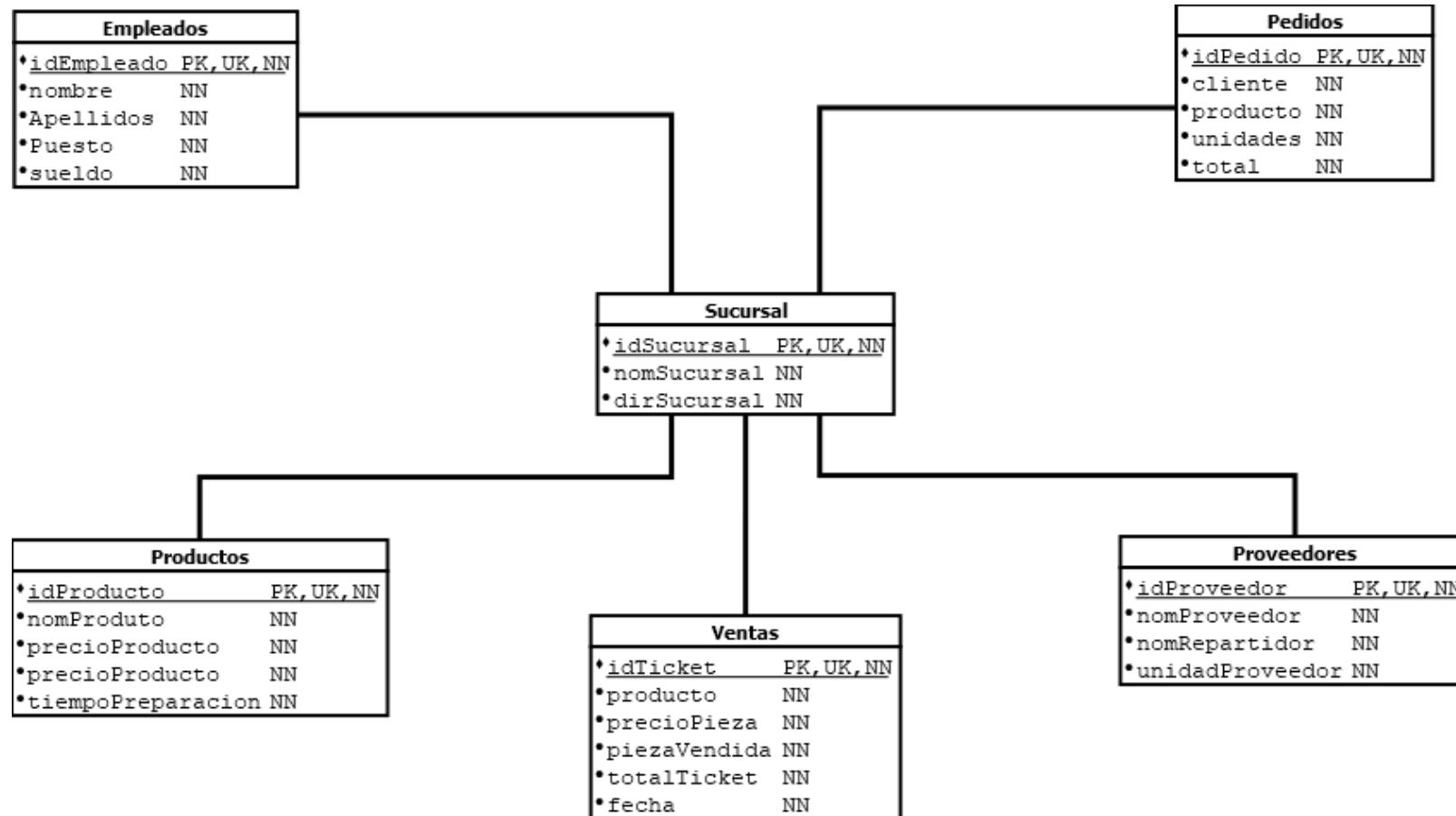
Características

Las características que nos ofrece DB Browser for SQLite son:

- Nos permite crear archivos de bases de datos y compactar archivos ya creados con SQLite.
- Permite crear, definir y eliminar tablas.
- Permite crear, definir y eliminar índices.
- Nos permite buscar, editar, añadir o eliminar entradas.
- Cuenta con un potente buscador de entradas.
- Importa y exporta entradas en modo texto.
- Importa y exporta tablas en ficheros CSV.
- Importa y exporta bases de datos en volcados SQL.
- Nos permite examinar los logs SQL.

Diagrama del Modelo Relacional

El diagrama de modelo relacional fue hecho en el software DIA, dicho software nos permite crear diagramas de todo tipo.



Implementación de la Base de datos

En este apartado mostraremos las capturas con las sentencias apropiadas para una estructura correcta de, las columnas para que base de datos funcione correctamente. Veamos.

Estructura de la tabla *Empleados*

Esta captura es sobre la estructura de, la tabla y sus columnas; sus tipos de datos y restricciones, así como su llave primaria.

The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is entered:

```
CREATE TABLE empleado (
    idEmpleado INTEGER,
    nombre TEXT,
    apellidos TEXT,
    puesto TEXT,
    sueldo REAL(6,2),
    PRIMARY key(idEmpleado))
```

The execution results pane at the bottom shows:

```
Ejecución terminada sin errores.
Resultado: consulta ejecutada con éxito. Tardó 2ms
En la linea 1:
CREATE TABLE empleado(
    idEmpleado INTEGER,
    nombre TEXT,
    apellidos TEXT,
    puesto TEXT,
    sueldo REAL(6,2),
    PRIMARY key(idEmpleado))
```

The right side of the window shows the table structure with columns: idEmpleado (INTEGER), nombre (TEXT), apellidos (TEXT), puesto (TEXT), and sueldo (REAL).

The screenshot shows the DB Browser for SQLite interface. In the main window, the table structure for 'empleado' is displayed in a table:

Nombre	Tipo	Esquema
Tablas (2)		
empleado		CREATE TABLE empleado(idEmpleado INTEGER, nombre TEXT, apellidos TEXT, puesto TEXT, sueldo REAL(6,2), PRIMARY key(idEmpleado))
idEmpleado	INTEGER	"idEmpleado" INTEGER
nombre	TEXT	"nombre" TEXT
apellidos	TEXT	"apellidos" TEXT
puesto	TEXT	"puesto" TEXT
sueldo	REAL(6, 2)	"sueldo" REAL(6, 2)
sucursal		CREATE TABLE sucursal(idSucursal INTEGER, nomSucursal TEXT, dirSucursal TEXT, PRIMARY KEY(idSucursal))
Indices (0)		
Vistas (0)		
Disparadores (0)		

Estructura de la tabla *Sucursal*

Esta captura es sobre la estructura de, la tabla y sus columnas; sus tipos de datos y restricciones, así como su llave primaria.

The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is displayed:

```
CREATE TABLE sucursal(
    idSucursal INTEGER,
    nomSucursal TEXT,
    dirSucursal TEXT,
    PRIMARY KEY(idSucursal))
```

Below the code, the message "Ejecución terminada sin errores." is shown, indicating successful execution. The SQL history panel on the right shows the execution of the CREATE TABLE statement for the sucursal table.

The screenshot shows the DB Browser for SQLite interface with the 'Estructura' (Structure) tab selected. The left pane displays a tree view of the database structure:

- Tablas (2):
 - empleado
 - sucursal
 - idSucursal (INTEGER)
 - nomSucursal (TEXT)
 - dirSucursal (TEXT)
- Indices (0)
- Vistas (0)
- Disparadores (0)

The right pane shows the detailed schema for the sucursal table:

Nombre	Tipo	Esquema
idSucursal	INTEGER	'idSucursal' INTEGER
nomSucursal	TEXT	'nomSucursal' TEXT
dirSucursal	TEXT	'dirSucursal' TEXT

Estructura de la tabla *Pedidos*

Esta captura es sobre la estructura de, la tabla y sus columnas; sus tipos de datos y restricciones, así como su llave primaria.

The screenshot shows the DB Browser for SQLite interface. In the main window, there is a SQL editor tab containing the following SQL code:

```
CREATE TABLE pedidos (
    idPedido INTEGER,
    cliente TEXT,
    producto TEXT,
    unidades INTEGER,
    precioTotal REAL(7,2),
    PRIMARY KEY(idPedido))
```

Below the SQL code, the status bar displays: "Ejecución terminada sin errores. Resultado: consulta ejecutada con éxito. Tardó 0ms". To the right of the main window, there is a "Historial de SQL" panel showing the executed SQL statements. The interface includes standard menu bars (Archivo, Editor, Ver, Herramientas, Ayuda) and toolbars with icons for creating new databases, opening existing ones, saving changes, and closing the application.

The screenshot shows the DB Browser for SQLite interface with the "Estructura" (Structure) tab selected. The left pane displays a tree view of database objects:

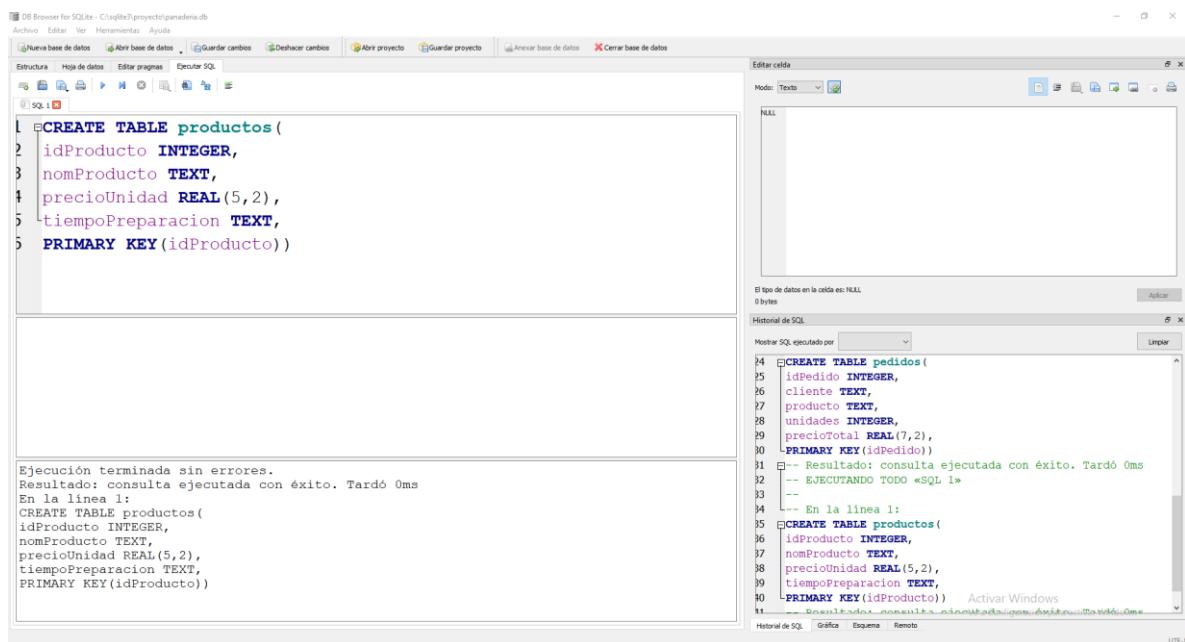
- Tables (3):
 - empleado
 - pedidos
 - idPedido (INTEGER)
 - cliente (TEXT)
 - producto (TEXT)
 - unidades (INTEGER)
 - precioTotal (REAL(7,2))
 - sucursal
- Índices (0)
- Vistas (0)
- Disparadores (0)

The "pedidos" table is expanded, showing its columns and their definitions. The "Esquema" (Schema) column contains the following details:

Nombre	Tipo	Esquema
idPedido	INTEGER	"idPedido" INTEGER
cliente	TEXT	"cliente" TEXT
producto	TEXT	"producto" TEXT
unidades	INTEGER	"unidades" INTEGER
precioTotal	REAL(7,2)	"precioTotal" REAL(7,2)

Estructura de la tabla *Productos*

Esta captura es sobre la estructura de, la tabla y sus columnas; sus tipos de datos y restricciones, así como su llave primaria.



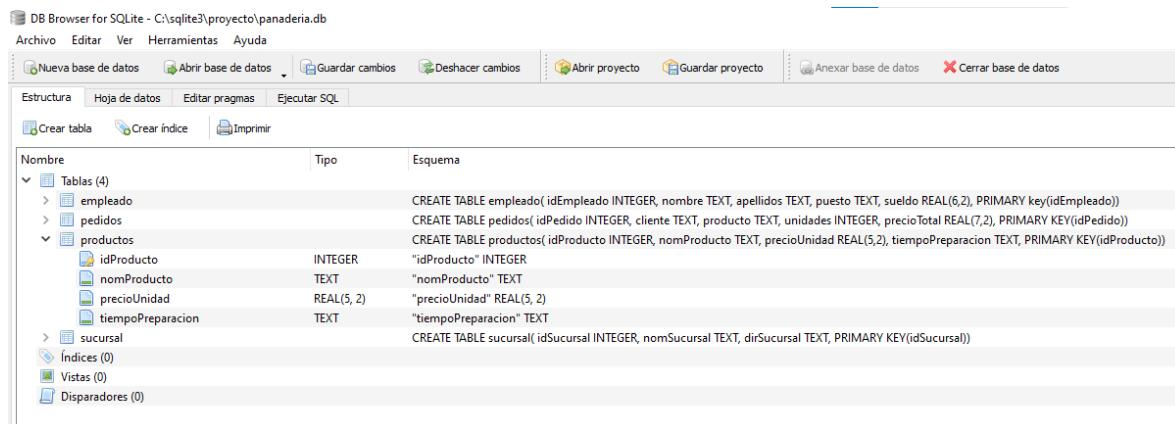
The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is displayed:

```
CREATE TABLE productos(
    idProducto INTEGER,
    nomProducto TEXT,
    precioUnidad REAL(5, 2),
    tiempoPreparacion TEXT,
    PRIMARY KEY(idProducto))
```

Below the code, the output of the execution is shown:

```
Ejecución terminada sin errores.  
Resultado: consulta ejecutada con éxito. Tardó 0ms  
En la linea 1:  
CREATE TABLE productos(  
    idProducto INTEGER,  
    nomProducto TEXT,  
    precioUnidad REAL(5, 2),  
    tiempoPreparacion TEXT,  
    PRIMARY KEY(idProducto))
```

The right side of the interface shows the history of executed SQL statements.



The screenshot shows the DB Browser for SQLite interface with the 'Estructura' (Structure) tab selected. It displays the database schema with the following tables and their columns:

Nombre	Tipo	Esquema
Tablas (4)		
empleado		CREATE TABLE empleado(idEmpleado INTEGER, nombre TEXT, apellidos TEXT, puesto TEXT, sueldo REAL(6,2), PRIMARY key(idEmpleado))
pedidos		CREATE TABLE pedidos(idPedido INTEGER, cliente TEXT, producto TEXT, unidades INTEGER, precioTotal REAL(7,2), PRIMARY KEY(idPedido))
productos		CREATE TABLE productos(idProducto INTEGER, nomProducto TEXT, precioUnidad REAL(5, 2), tiempoPreparacion TEXT, PRIMARY KEY(idProducto))
idProducto	INTEGER	"idProducto" INTEGER
nomProducto	TEXT	"nomProducto" TEXT
precioUnidad	REAL(5, 2)	"precioUnidad" REAL(5, 2)
tiempoPreparacion	TEXT	"tiempoPreparacion" TEXT
sucursal		CREATE TABLE sucursal(idSucursal INTEGER, nomSucursal TEXT, dirSucursal TEXT, PRIMARY KEY(idSucursal))
Índices (0)		
Vistas (0)		
Disparadores (0)		

Estructura de la tabla Ventas

Esta captura es sobre la estructura de, la tabla y sus columnas; sus tipos de datos y restricciones, así como su llave primaria.

The screenshot shows the DB Browser for SQLite interface. In the main window, the SQL tab contains the following code:

```
1 CREATE TABLE ventas (
2     idTicket INTEGER,
3     producto TEXT,
4     precioUnidad REAL(5, 2),
5     piezasVendidas INTEGER,
6     totalTicket REAL(10, 2),
7     fecha datetime,
8     PRIMARY KEY(idTicket)
9 )
```

Below the code, the status bar indicates: "Ejecución terminada sin errores. Resultado: consulta ejecutada con éxito. Tardó 2ms". The History tab on the right shows the executed SQL statement.

The screenshot shows the DB Browser for SQLite interface. In the main window, the Estructura tab displays the table structure:

Nombre	Tipo	Esquema
tablas (5)		
empleado		CREATE TABLE empleado(idEmpleado INTEGER, nombre TEXT, apellidos TEXT, puesto TEXT, sueldo REAL(6,2), PRIMARY key(idEmpleado))
pedidos		CREATE TABLE pedidos(idPedido INTEGER, cliente TEXT, producto TEXT, unidades INTEGER, precioTotal REAL(7,2), PRIMARY KEY(idPedido))
productos		CREATE TABLE productos(idProducto INTEGER, nomProducto TEXT, precioUnidad REAL(5,2), tiempoPreparacion TEXT, PRIMARY KEY(idProducto))
sucursal		CREATE TABLE sucursal(idSucursal INTEGER, nomSucursal TEXT, dirSucursal TEXT, PRIMARY KEY(idSucursal))
ventas		CREATE TABLE ventas(idTicket INTEGER, producto TEXT, precioUnidad REAL(5,2), piezasVendidas INTEGER, totalTicket REAL(10,2), fecha datetime, PRIMARY KEY(idTicket))
idTicket	INTEGER	"idTicket" INTEGER
producto	TEXT	"producto" TEXT
precioUnidad	REAL(5, 2)	"precioUnidad" REAL(5, 2)
piezasVendidas	INTEGER	"piezasVendidas" INTEGER
totalTicket	REAL(10, 2)	"totalTicket" REAL(10, 2)
fecha	datetime	"fecha" datetime

Estructura de la tabla *Proveedores*

Esta captura es sobre la estructura de, la tabla y sus columnas; sus tipos de datos y restricciones, así como su llave primaria.

The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is displayed:

```
CREATE TABLE proveedores(
    idProveedor INTEGER,
    nomProveedor TEXT,
    nomRepartidor TEXT,
    unidadProveedor INTEGER,
    PRIMARY KEY(idProveedor))
```

Below the code, the status bar indicates: "Ejecución terminada sin errores. Resultado: consulta ejecutada con éxito. Tardó 0ms". The bottom right corner of the window shows a small message: "Activar Windows".

The screenshot shows the DB Browser for SQLite interface. In the left sidebar under "Estructura", the "Tablas (6)" section is expanded, showing the "proveedores" table. The table has four columns: "idProveedor" (INTEGER), "nomProveedor" (TEXT), "nomRepartidor" (TEXT), and "unidadProveedor" (INTEGER). The "Esquema" column shows the corresponding SQL create statements for each column.

Nombre	Tipo	Esquema
tablas (6)		
empleado		CREATE TABLE empleado(idEmpleado INTEGER, nombre TEXT, apellidos TEXT, puesto TEXT, sueldo REAL(6,2), PRIMARY key(idEmpleado))
pedidos		CREATE TABLE pedidos(idPedido INTEGER, cliente TEXT, producto TEXT, unidades INTEGER, precioTotal REAL(7,2), PRIMARY KEY(idPedido))
productos		CREATE TABLE productos(idProducto INTEGER, nomProducto TEXT, precioUnidad REAL(5,2), tiempoPreparacion TEXT, PRIMARY KEY(idProducto))
proveedores		CREATE TABLE proveedores(idProveedor INTEGER, nomProveedor TEXT, nomRepartidor TEXT, unidadProveedor INTEGER, PRIMARY KEY(idProveedor))
sucursal		CREATE TABLE sucursal(idSucursal INTEGER, nomSucursal TEXT, dirSucursal TEXT, PRIMARY KEY(idSucursal))
ventas		CREATE TABLE ventas(idTicket INTEGER, producto TEXT, precioUnidad REAL(5,2), piezasVendidas INTEGER, totalTicket REAL(10,2), fecha datetime, PRIMARY KEY(idTicket))
Indices (0)		
Vistas (0)		
Disparadores (0)		

Alta de registros en la base de datos

Ya estructurada nuestras tablas, es momento de agregar registros a cada una de ellas; Se darán de alta por lo menos 10 registros respectivamente en cada tabla. En el siguiente apartado se muestran las capturas del momento del registro de datos en cada tabla. Veamos.

Registros en tabla *Empleados*

Las capturas muestran el, alta de registros con sus sentencias apropiadas y, también la consulta SELECT para mostrar los datos previamente registrados.

The screenshot shows the DB Browser for SQLite interface. On the left, the SQL tab contains the following code:

```
1 INSERT INTO empleado
2 VALUES
3 (123654,'roxana','montes','administrador',5824.85),
4 (234554,'erick','jaquez','administrador',5824.85),
5 (365201,'joselinne','palomino','cajera',1732.45),
6 (123698,'lezly','nicasio','cajera',1732.45),
7 (201549,'lesley','rodriguez','cajera',1732.45),
8 (540875,'elizabeth','arreola','cajera',1732.45),
9 (789456,'david','castillo','contador',4398.45),
0 (215420,'ilse','gomez','gerente',6924.14),
1 (175489,'gerardo','cisneros','panadero',2147.56),
2 (256410,'cesar','trejo','panadero',2147.56),
3 (120455,'luis','reveles','panadero',2147.56),
4 (245648,'ivan','torres','seguridad',2147.56)
5
```

The status bar at the bottom indicates: "Ejecución terminada sin errores. Resultado: consulta ejecutada con éxito. Tardó 1ms, 12 filas afectadas".

To the right, there is a floating window titled "Historial de SQL" which also contains the same SQL code.

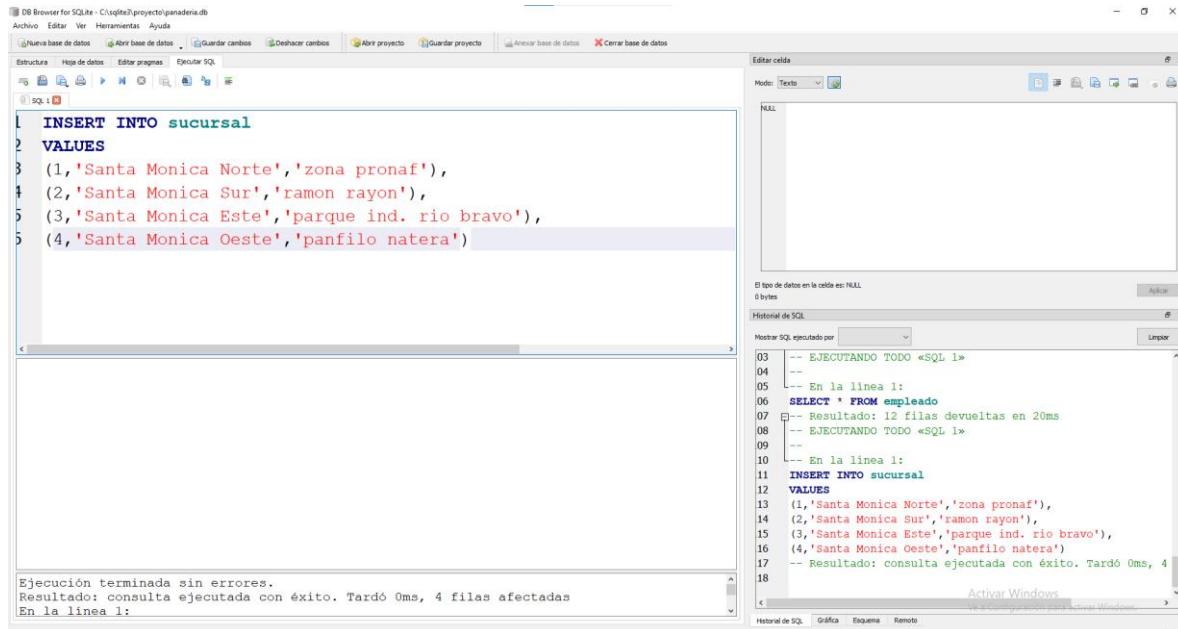
Below these, the SQL 1 tab displays the results of a SELECT query:

```
1 SELECT * FROM empleado
```

	idEmpleado	nombre	apellidos	puesto	sueldo
1	120455	luis	reveles	panadero	2147.56
2	123654	roxana	montes	administrador	5824.85
3	123698	lezly	nicasio	cajera	1732.45
4	175489	gerardo	cisneros	panadero	2147.56
5	201549	lesley	rodriguez	cajera	1732.45
6	215420	ilse	gomez	gerente	6924.14
7	234554	erick	jaquez	administrador	5824.85
8	245648	ivan	torres	seguridad	2147.56
9	256410	cesar	trejo	panadero	2147.56
10	365201	joselinne	palomino	cajera	1732.45
11	540875	elizabeth	arreola	cajera	1732.45
12	789456	david	castillo	contador	4398.45

Registros en tabla **Sucursal**

Las capturas muestran el alta de registros con sus sentencias apropiadas y, también la consulta SELECT para mostrar los datos previamente registrados.

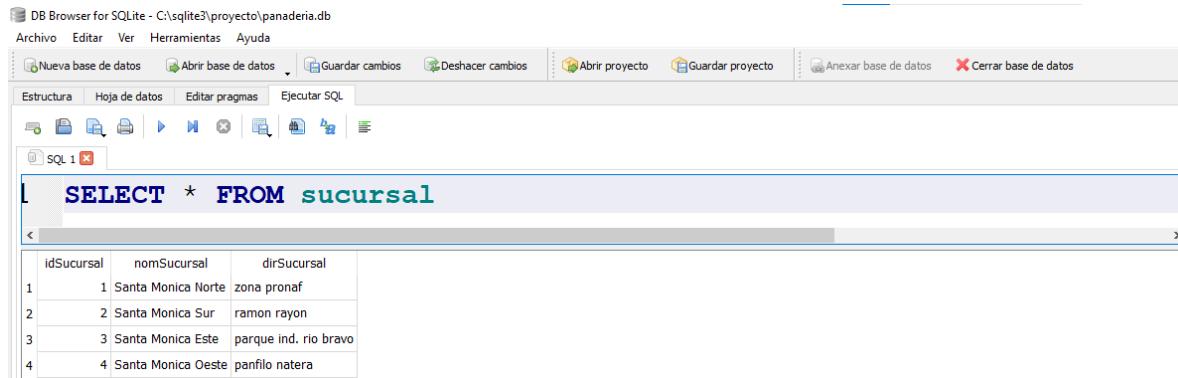


The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is entered:

```
INSERT INTO sucursal
VALUES
(1,'Santa Monica Norte','zona pronaf'),
(2,'Santa Monica Sur','ramon rayon'),
(3,'Santa Monica Este','parque ind. rio bravo'),
(4,'Santa Monica Oeste','panfilo natera')
```

The status bar at the bottom indicates: "Ejecución terminada sin errores. Resultado: consulta ejecutada con éxito. Tardó 0ms, 4 filas afectadas En la linea 1:"

In the bottom right corner of the interface, there is a small window titled "Historial de SQL" showing the executed SQL statements.



The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is entered:

```
SELECT * FROM sucursal
```

The results are displayed in a table below:

	idSucursal	nomSucursal	dirSucursal
1	1	Santa Monica Norte	zona pronaf
2	2	Santa Monica Sur	ramon rayon
3	3	Santa Monica Este	parque ind. rio bravo
4	4	Santa Monica Oeste	panfilo natera

Registros en tabla *Productos*

Las capturas muestran el alta de registros con sus sentencias apropiadas y, también la consulta SELECT para mostrar los datos previamente registrados.

The screenshot shows the DB Browser for SQLite interface. The SQL tab contains the following code:

```
2 VALUES
3 (3564,'birote',3.45,'26 min.'),
4 (2158,'bisquet',3.96,'30 min.'),
5 (7410,'hojaldra',5.52,'10 min.'),
6 (2583,'cocol',5.61,'21 min.'),
7 (9634,'corbata',5.40,'18 min.'),
8 (1596,'beno',5.84,'13 min.'),
9 (7523,'bigote',5.10,'10 min.'),
0 (8531,'garibaldi',5.73,'15 min.'),
1 (3719,'pan de yema',5.33,'25 min.'),
2 (5468,'oreja',5.42,'27 min.')
3
```

The status bar at the bottom indicates: "Ejecución terminada sin errores. Resultado: consulta ejecutada con éxito. Tardó 4ms, 10 filas afectadas En la linea 1".

The History tab shows the executed SQL statements:

```
40 -- En la linea 1:
41 INSERT INTO productos
42 VALUES
43 (3564,'birote',3.45,'26 min.'),
44 (2158,'bisquet',3.96,'30 min.'),
45 (7410,'hojaldra',5.52,'10 min.'),
46 (2583,'cocol',5.61,'21 min.'),
47 (9634,'corbata',5.40,'18 min.'),
48 (1596,'beno',5.84,'13 min.'),
49 (7523,'bigote',5.10,'10 min.'),
50 (8531,'garibaldi',5.73,'15 min.'),
51 (3719,'pan de yema',5.33,'25 min.'),
52 (5468,'oreja',5.42,'27 min.')
53 -- Resultado: consulta ejecutada con éxito. Tardó 4ms, 10 filas afectadas
54
```

The screenshot shows the results of a SELECT query in the SQL tab:

```
1 SELECT * FROM productos
2
```

	idProducto	nomProducto	precioUnidad	tiempoPreparacion
1	1596	beno	5.84	13 min.
2	2158	bisquet	3.96	30 min.
3	2583	cocol	5.61	21 min.
4	3564	birote	3.45	26 min.
5	3719	pan de yema	5.33	25 min.
6	5468	oreja	5.42	27 min.
7	7410	hojaldra	5.52	10 min.
8	7523	bigote	5.1	10 min.
9	8531	garibaldi	5.73	15 min.
10	9634	corbata	5.4	18 min.

Registros en tabla Ventas

Las capturas muestran el alta de registros con sus sentencias apropiadas y, también la consulta SELECT para mostrar los datos previamente registrados.

The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is visible:

```
2 VALUES
3 (8524, 'bisquet', 3.96, 10, 30.96, '2021-05-20'),
4 (9432, 'birote', 3.45, 21, 72.45, '2021-05-20'),
5 (6428, 'corbata', 5.40, 3, 16.20, '2021-05-20'),
6 (3548, 'oreja', 5.42, 9, 48.78, '2021-05-22'),
7 (2485, 'pan de yema', 5.33, 6, 31.98, '2021-05-25'),
8 (9631, 'oreja', 5.42, 4, 21.68, '2021-05-25'),
9 (8527, 'hojaldrá', 5.52, 15, 82.8, '2021-05-28'),
0 (2347, 'bisquet', 3.96, 7, 27.72, '2021-05-30'),
1 (1642, 'oreja', 5.42, 11, 59.62, '2021-05-31'),
2 (7354, 'oreja', 5.42, 10, 50.42, '2021-05-31')
```

The status bar at the bottom indicates: "Ejecución terminada sin errores. Resultado: consulta ejecutada con éxito. Tardó 0ms, 10 filas afectadas En la linea 1".

In the top right, there is an "Editor celda" window showing a single cell with the value "NULL". Below it is a "Historial de SQL" window displaying the executed SQL statements.

The screenshot shows the results of a SELECT query on the "ventas" table. The query is:

```
1 SELECT * FROM ventas
```

The resulting table is:

	idTicket	producto	precioUnidad	piezasVendidas	totalTicket	fecha
1	1234	bisquet	3.96	7	27.72	2021-05-20
2	1642	oreja	5.42	11	59.62	2021-05-31
3	2347	bisquet	3.96	7	27.72	2021-05-30
4	2485	pan de yema	5.33	6	31.98	2021-05-25
5	3548	oreja	5.42	9	48.78	2021-05-22
6	6428	corbata	5.4	3	16.2	2021-05-20
7	7354	oreja	5.42	10	50.42	2021-05-31
8	8524	bisquet	3.96	10	30.96	2021-05-20
9	8527	hojaldrá	5.52	15	82.8	2021-05-28
10	9432	birote	3.45	21	72.45	2021-05-20
11	9631	oreja	5.42	4	21.68	2021-05-25

Registros en tabla *Proveedores*

Las capturas muestran el alta de registros con sus sentencias apropiadas y, también la consulta SELECT para mostrar los datos previamente registrados.

The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is visible:

```
1 INSERT INTO proveedores
2 VALUES
3 (7410,'alberth polo','manuel flores',15),
4 (3659,'aldisa','diego rodriguez',03),
5 (9524,'horn la parra','luis juarez',68),
6 (5210,'mantepan','juan ramirez',36),
7 (2864,'grupo euroestrellas','edgar flores',64)
```

Below the code, the message "Ejecución terminada sin errores." is displayed, followed by "Resultado: consulta ejecutada con éxito. Tardó 0ms, 5 filas afectadas". The History tab shows the executed SQL statements, including the insertion and a previous SELECT query.

The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is visible:

```
1 SELECT * FROM proveedores
```

Below the code, the results of the query are displayed in a table:

	idProveedor	nomProveedor	nomRepardor	unidadProveedor
1	2864	grupo euroestrellas	edgar flores	64
2	3659	aldisa	diego rodriguez	3
3	5210	mantepan	juan ramirez	36
4	7410	alberth polo	manuel flores	15
5	9524	horn la parra	luis juarez	68

Registros en tabla *Pedidos*

Las capturas muestran el alta de registros con sus sentencias apropiadas y, también la consulta SELECT para mostrar los datos previamente registrados.

The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is displayed:

```
1 INSERT INTO pedidos
2 VALUES
3 (6245,'foxconn','oreja',60,325.2),
4 (6710,'uacj','hojalda',465,2566.8),
5 (6577,'rio bravo','oreja',842,4563.64),
6 (6344,'itcj','pan de yema',152,810.16),
7 (6014,'itcj','oreja',635,3441.7)
```

Below the code, the output window shows:

```
Ejecución terminada sin errores.
Resultado: consulta ejecutada con éxito. Tardó 0ms, 5 filas afectadas
En la linea 1:
INSERT INTO pedidos
VALUES
(6245,'foxconn','oreja',60,325.2),
(6710,'uacj','hojalda',465,2566.8),
(6577,'rio bravo','oreja',842,4563.64),
(6344,'itcj','pan de yema',152,810.16),
(6014,'itcj','oreja',635,3441.7)
```

The right panel displays the history of executed SQL statements, including the inserted data.

The screenshot shows the DB Browser for SQLite interface. In the main SQL tab, the following SQL code is displayed:

```
1 SELECT * FROM pedidos
```

Below the code, the output window shows the resulting table:

	idPedido	cliente	producto	unidades	precioTotal
1	6014	itcj	oreja	635	3441.7
2	6245	foxconn	oreja	60	325.2
3	6344	itcj	pan de yema	152	810.16
4	6577	rio bravo	oreja	842	4563.64
5	6710	uacj	hojalda	465	2566.8

Referencia SQLite

Descarga de la Referencia

A continuación, se muestra la en donde buscar y descargar la referencia de SQLite para posteriormente agregarla a Visual Studio.

1. Como primer paso hay que dirigirnos al siguiente enlace: [SQLite Download](https://www.sqlite.org/download.html)

Page

The screenshot shows a Microsoft Edge browser window displaying the SQLite Download Page at <https://www.sqlite.org/download.html>. The page features the SQLite logo and navigation links for Home, About, Documentation, Download, License, Support, and Purchase. A search bar is located in the top right corner. The main content area is titled "SQLite Download Page" and contains sections for "Pre-release Snapshots", "Source Code", "Documentation", and "Precompiled Binaries for Android" and "Linux". Each section lists files with their names, sizes, and SHA-3 hash values. A banner at the top right reads "Small. Fast. Reliable. Choose any three."

Pre-release Snapshots

sqlite-snapshot-202105171714.tar.gz (2.83 MiB)	The amalgamation source code, the command-line shell source code, configure/make scripts for unix, and a Makefile.msc for Windows. See the change log or the timeline for more information. (sha3: 23b7adc5bb2a332f8b529ca9016867047d492fa1c1ea15c33c26da10aca79c9)
---	--

Source Code

sqlite-amalgamation-3350500.zip (2.35 MiB)	C source code as an amalgamation, version 3.35.5. (sha3: 6175cbabe28c2274aa6fb305076116622a4ecb7829673b92290dc047fba9bbe6)
sqlite-autoconf-3350500.tar.gz (2.82 MiB)	C source code as an amalgamation. Also includes a "configure" script and TEA makefiles for the TCL Interface . (sha3: c84854020531aae4f731b91b47133d3573d3c026975a36a077d6b60325513696)

Documentation

sqlite-doc-3350500.zip (9.73 MiB)	Documentation as a bundle of static HTML files. (sha3: 762f0fd4c1478e572ff92a5227fd2b3dc0ee1325e0675cc7c12977bc0d8ac2e)
--	--

Precompiled Binaries for Android

sqlite-android-3350500.aar (3.12 MiB)	A precompiled Android library containing the core SQLite together with appropriate Java bindings, ready to drop into any Android Studio project. (sha3: 85b09ffcaf1f78c3cc9f6c354f4c4912c3c749d5f79b6991202707de1ed7d0a24c)
--	--

Precompiled Binaries for Linux

sqlite-tools-linux-x86-3350500.zip (2.06 MiB)	A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqlcipher program, and the sqlite3_analyzer program. (sha3: b62d75407e6d091c429b5fce3938f4486bfdd63887f473ade0d44fd79b92276b)
--	---

2. Vamos al apartado; “Precompiled Binaries for .NET”. Hacemos clic en sobre “System.Data.SQLite”.:

The screenshot shows the SQLite Download Page at <https://www.sqlite.org/download.html>. The 'Precompiled Binaries for .NET' section is highlighted with a red oval. It contains links for 'System.Data.SQLite' and 'Alternative Source Code Formats'. The 'System.Data.SQLite' link leads to the System.Data.SQLite.org website.

Precompiled Binaries for .NET

[System.Data.SQLite](#) Visit the System.Data.SQLite.org website and especially the [download page](#) for source code and binaries of SQLite for .NET.

Alternative Source Code Formats

[sqlite-src-3350500.zip](#) Snapshot of the complete (raw) source tree for SQLite version 3.35.5. See [How To Compile SQLite](#) for usage details.
(12.23 MB) (sha3: 980a78e5827e3a3f63ee76c0cbf5ca335261ac5cae25aaeb8bd9ef0f3e43)

[sqlite-preprocessed-3350500.zip](#) Preprocessed C sources for SQLite version 3.35.5.
(2.56 MB) (sha3: d09af0d010a3960bea655e3156389e571240a9a4630773d43362ebb509feb4c8)

Build Product Names

Build products are named using one of the following templates:

1. [sqlite-product-version.zip](#)
2. [sqlite-product-version.tar.gz](#)
3. [sqlite-product-os-cpu-version.zip](#)
4. [sqlite-product-date.zip](#)

Templates (1) and (2) are used for source-code products. Template (1) is used for generic source-code products and templates (2) is used for source-code products that are generally only useful on unix-like platforms. Template (3) is used for precompiled binaries products. Template (4) is used for unofficial pre-release "snapshots" of source code.

3. Nos va a redirigir a otra página, estando aquí deslizamos hacia abajo y veremos el apartado “List of Release Packages”:

The screenshot shows the System.Data.SQLite: Downloads page at <http://System.Data.SQLite.org/index.html/doc/trunk/www/downloads.wiki>. The 'List of Release Packages' section is visible.

List of Release Packages

Source Code

- [sqlite-netFx-source-1.0.113.0.zip](#) This ZIP archive contains all current source code for System.Data.SQLite 1.0.113.0 (3.32.1) combined into a single archive file.
(6.83 MB) (sha1: f2f89d1fc36658b1509bfa0ff7ae31203bf2a8)
- [sqlite-netFx-full-source-1.0.113.0.zip](#) This ZIP archive contains all current source code for System.Data.SQLite 1.0.113.0 (3.32.1) and the extra files needed to run the unit test suite, combined into a single archive file.
(13.26 MB) (sha1: 57a4a973c839314d2adbfb3e3c737feba8fdff72e)

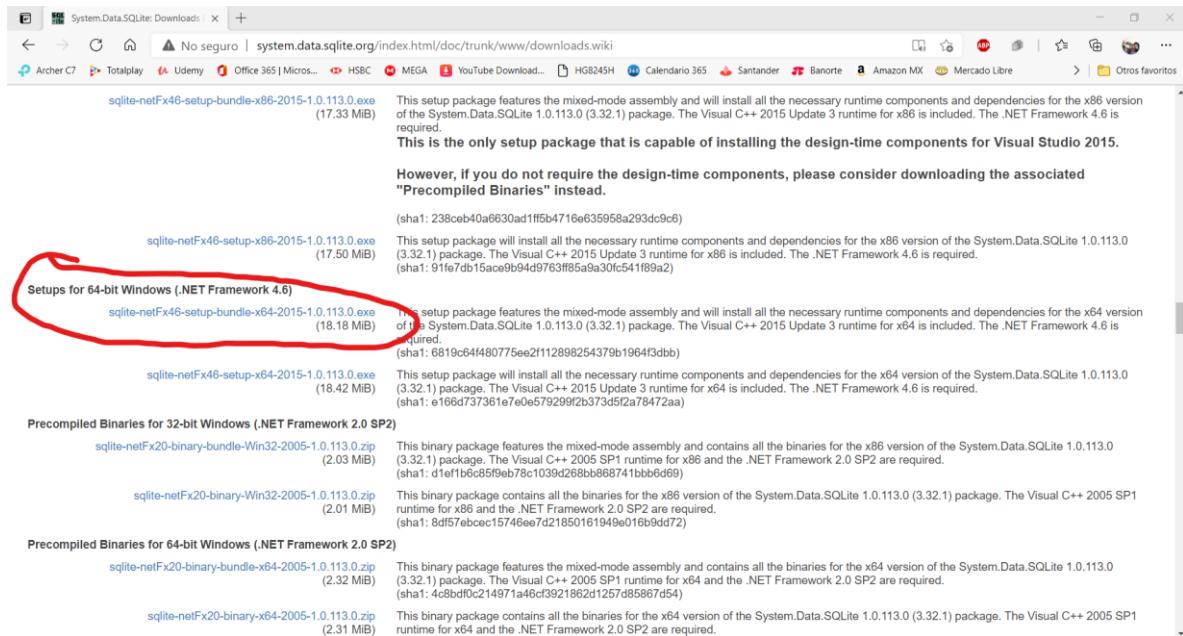
Setups for 32-bit Windows (.NET Framework 2.0 SP2)

- [sqlite-netFx20-setup-bundle-x86-2005-1.0.113.0.exe](#) This setup package features the mixed-mode assembly and will install all the necessary runtime components and dependencies for the x86 version of the System.Data.SQLite 1.0.113.0 (3.32.1) package. The Visual C++ 2005 SP1 runtime for x86 is included. The .NET Framework 2.0 SP2 is required.
(5.74 MB) **This is the only setup package that is capable of installing the design-time components for Visual Studio 2005.**
- [sqlite-netFx20-setup-x86-2005-1.0.113.0.exe](#) However, if you do not require the design-time components, please consider downloading the associated "Precompiled Binaries" instead.
(sha1: 297ce751e5ed391dc37238ff6483e24bad2bd)

Setups for 64-bit Windows (.NET Framework 2.0 SP2)

- [sqlite-netFx20-setup-bundle-x64-2005-1.0.113.0.exe](#) This setup package features the mixed-mode assembly and will install all the necessary runtime components and dependencies for the x64 version of the System.Data.SQLite 1.0.113.0 (3.32.1) package. The Visual C++ 2005 SP1 runtime for x64 is included. The .NET Framework 2.0 SP2 is required.
(6.39 MB) (sha1: 406278dc48086b2029db32c278faf0973a0721)

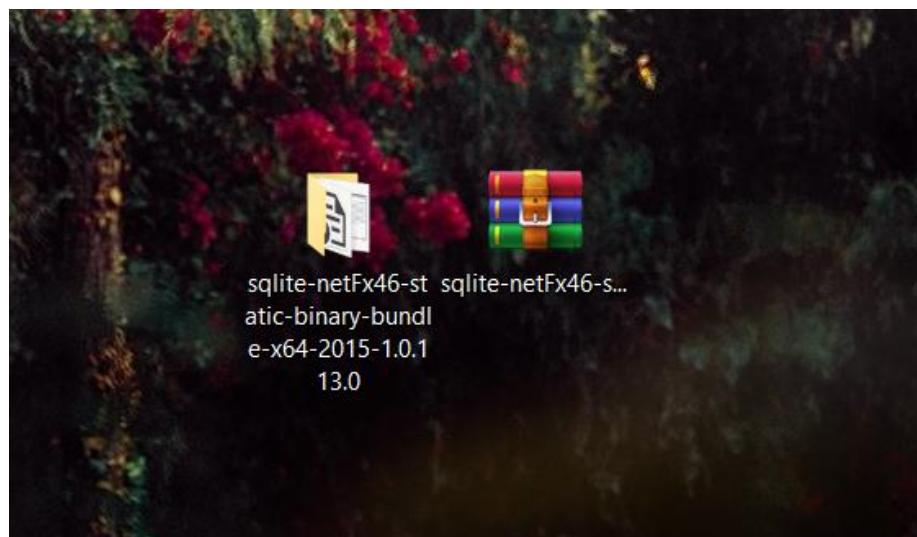
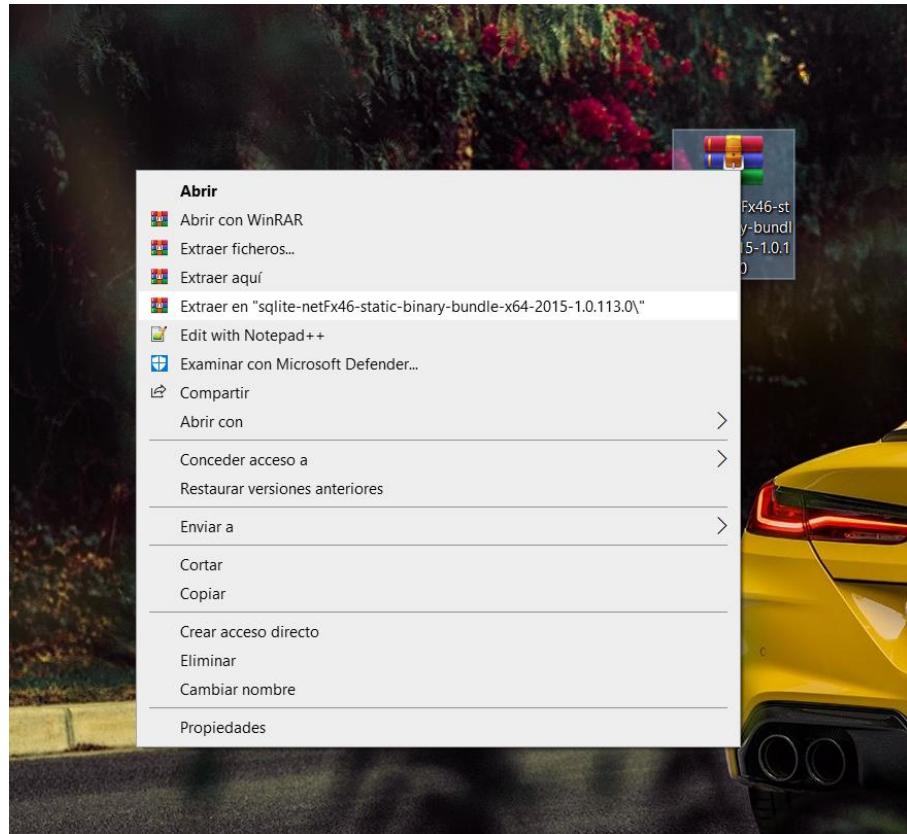
4. En la parte izquierda de la página, buscamos “Setups for 64-bit Windows (.NET Framework 4.6)”; Los 64-bits dependerá del sistema operativo que se tenga en la computadora, y el Framework es la arquitectura en la que estamos trabajado con el proyecto en Visual Studio:



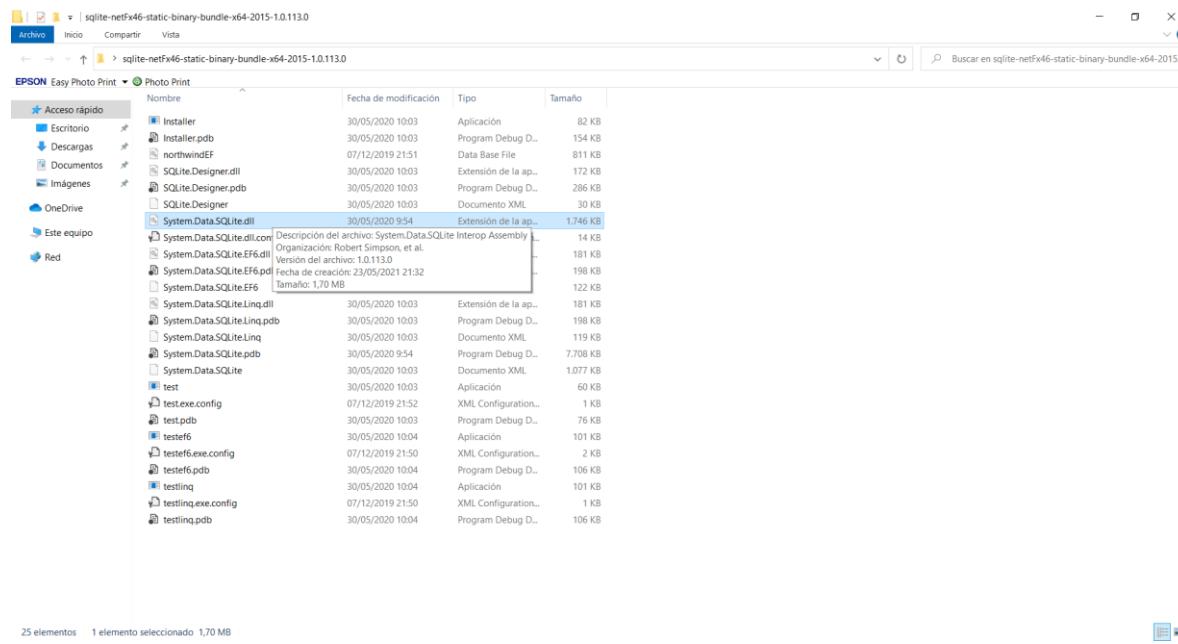
Nota: En este caso el proyecto se ejecuta en un sistema operativo 64-bit y se basa en el Framework 5.0 del Microsoft Visual Studio Community 2019.

Agregando la referencia a Visual Basic.NET

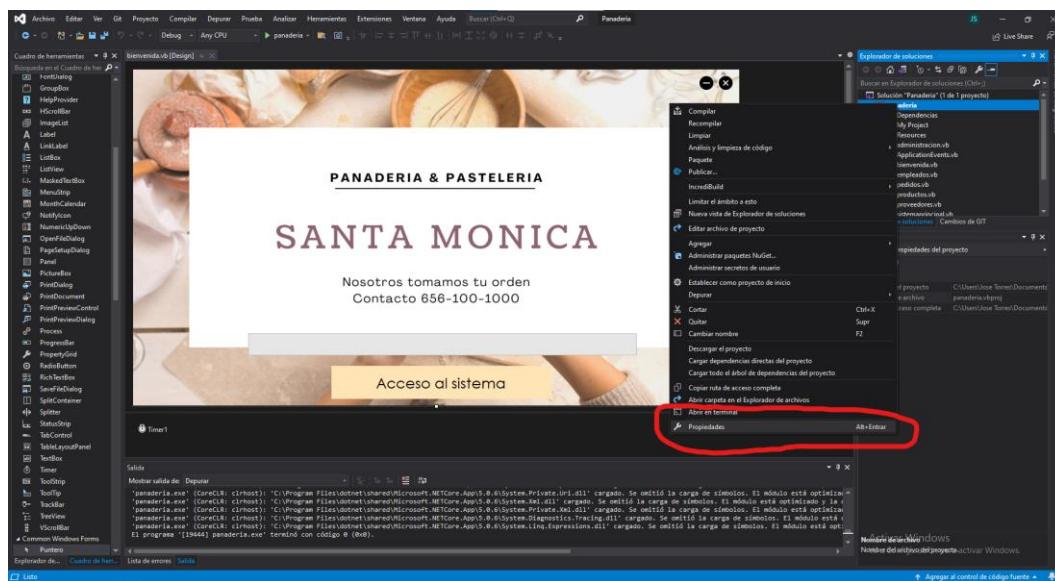
1. Una vez descargado, procedemos a descomprimirlo:



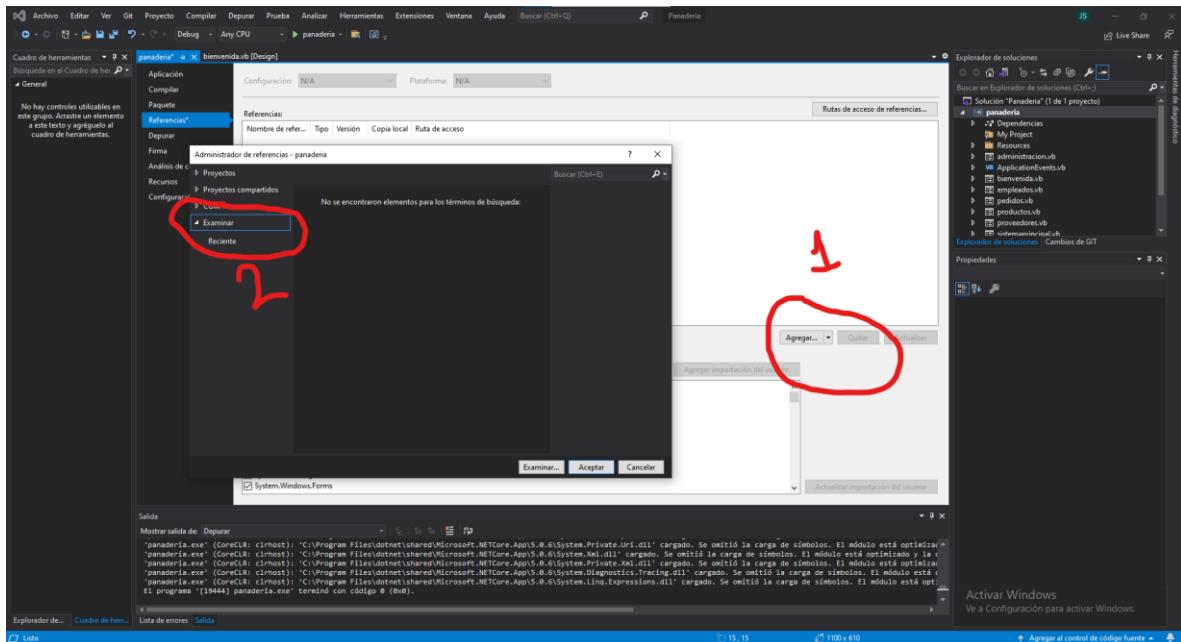
2. Abrimos la carpeta, buscamos el archivo “System.Data.SQLite” con la extensión (.dll); Este archivo es el que vamos a necesitar para agregarlo como referencia a Visual Studio:



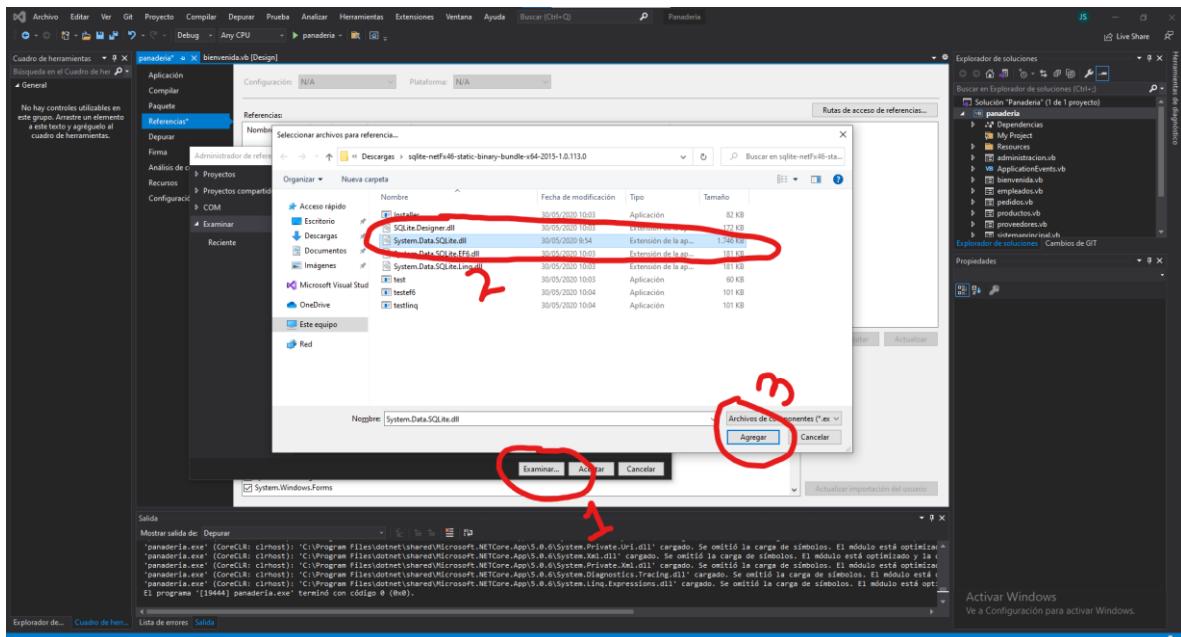
3. Teniendo nuestro proyecto abierto, ahora vamos a agregarlo al mismo; Damos clic derecho sobre el nombre de nuestro proyecto y vamos a las propiedades del mismo:



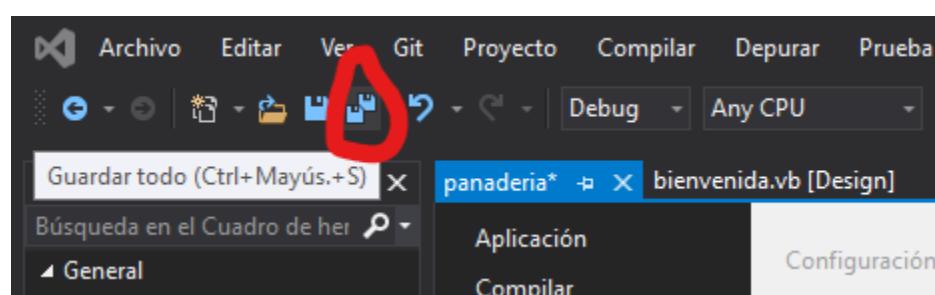
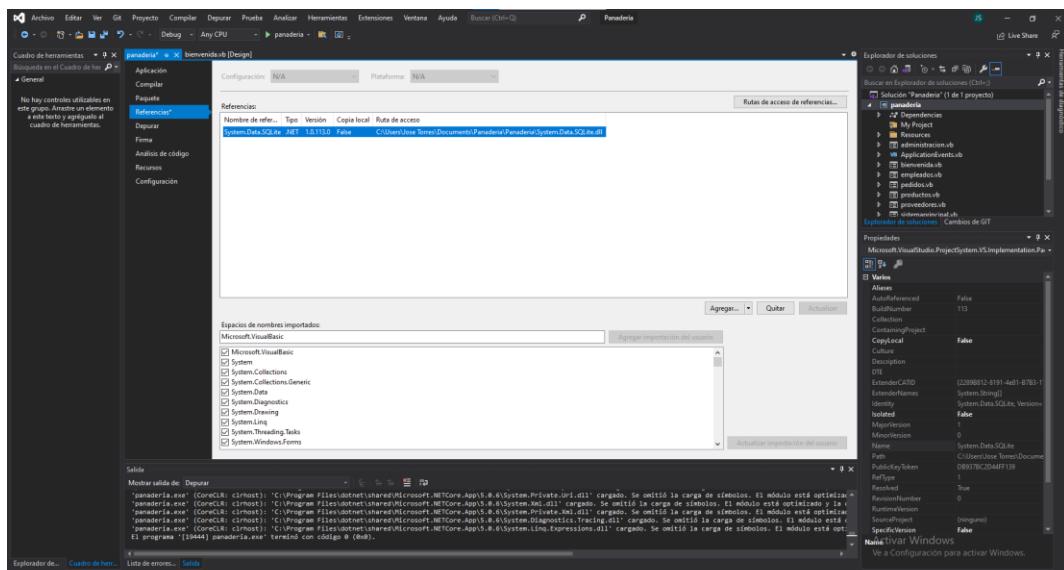
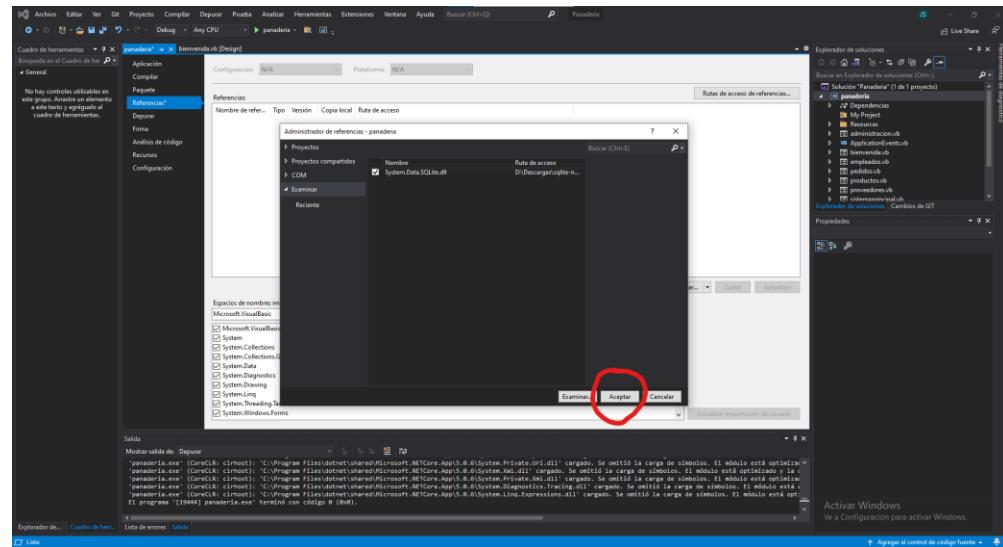
4. Vamos a Referencias y disponemos a agregar nuestro archivo (.dll), en el botón de agregar y este mismo nos abrirá una ventana, y vamos a Examinar:



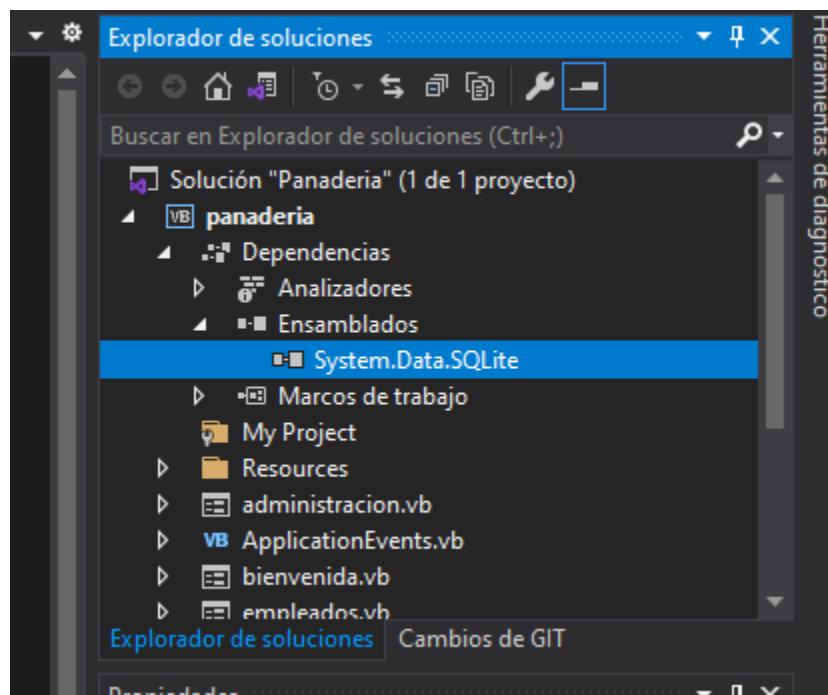
5. En esa ventana de navegación damos clic en el botón “Examinar...”, abrirá otra venta y es aquí donde buscamos nuestro archivo (.dll) y lo agregamos:



6. Hemos agregado nuestra nueva referencia correctamente a nuestro proyecto, queda solo dar Aceptar y vemos nuestra referencia agregada al proyecto y queda solo guardar los cambios realizados:



7. Por último, verificamos si la nueva referencia se ha agregado correctamente, vamos damo clic sobre el nombre del proyecto, abrimos “Dependencias”, y por último abrimos “Ensamblados”, vemos que si se agregó la referencia a nuestro proyecto:



Nota: Haciendo los pasos anteriores estamos listos para hacer la conexión entre SQLite y Visual Studio.

CRUD como base de la gestión de datos

El concepto CRUD está estrechamente vinculado a la gestión de datos digitales. CRUD hace referencia a un acrónimo en el que se reúnen las primeras letras de las cuatro operaciones fundamentales de aplicaciones persistentes en sistemas de bases de datos:

- Create (Crear registros)
- Read bzw. Retrieve (Leer registros)
- Update (Actualizar registros)
- Delete bzw. Destroy (Borrar registros)

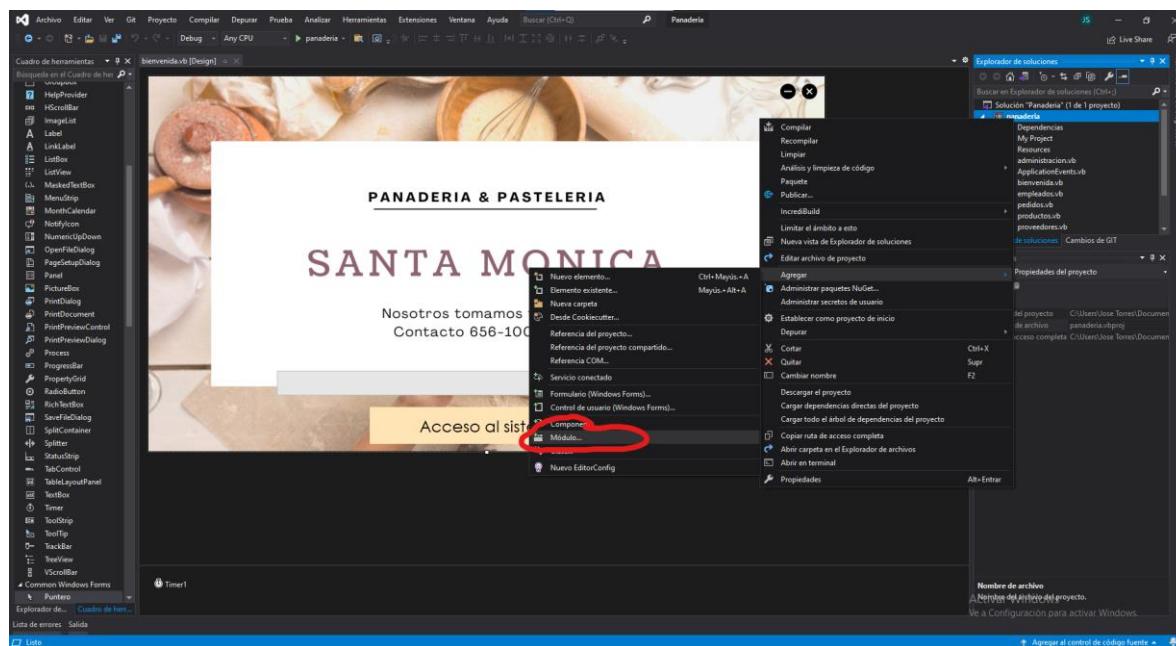
En pocas palabras, CRUD resume las funciones requeridas por un usuario para crear y gestionar datos. Varios procesos de gestión de datos están basados en CRUD, en los que dichas operaciones están específicamente adaptadas a los requisitos del sistema y de usuario, ya sea para la gestión de bases de datos o para el uso de aplicaciones.

Para los expertos, las operaciones son las herramientas de acceso típicas e indispensables para comprobar, por ejemplo, los problemas de la base de datos, mientras que, para los usuarios, CRUD significa crear una cuenta (create) y utilizarla (read), actualizarla (update) o borrarla (delete) en cualquier momento.

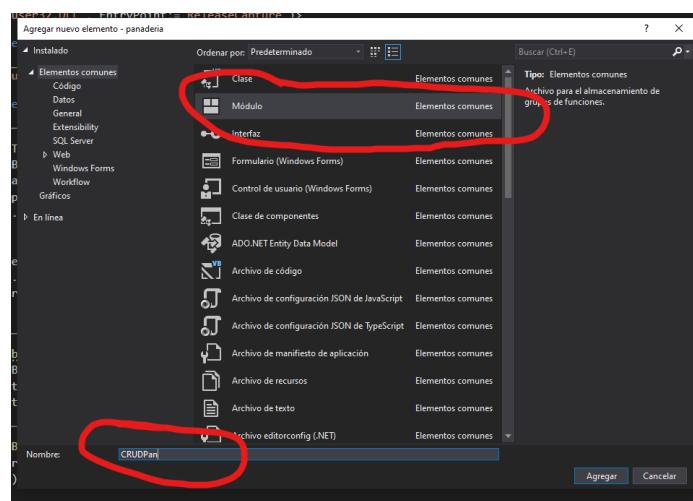
Dependiendo de la configuración regional, las operaciones CRUD pueden implementarse de diferentes maneras

CRUD en Visual Basic.NET

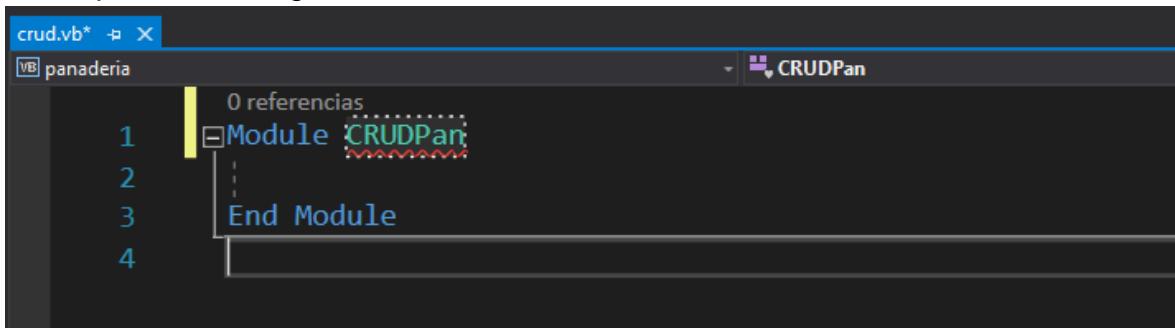
Ya hemos agregado la referencia, lo siguiente que necesitamos es agregar un nuevo módulo a nuestro proyecto, el cual lo llamaremos “CRUDPan”, para ello damos clic derecho en el nombre del proyecto, vamos al menú agregar y seleccionamos la opción “Modulo”:



Aparecerá una ventana para agregar el módulo y automáticamente nos selecciona el módulo y le ponemos un nombre:



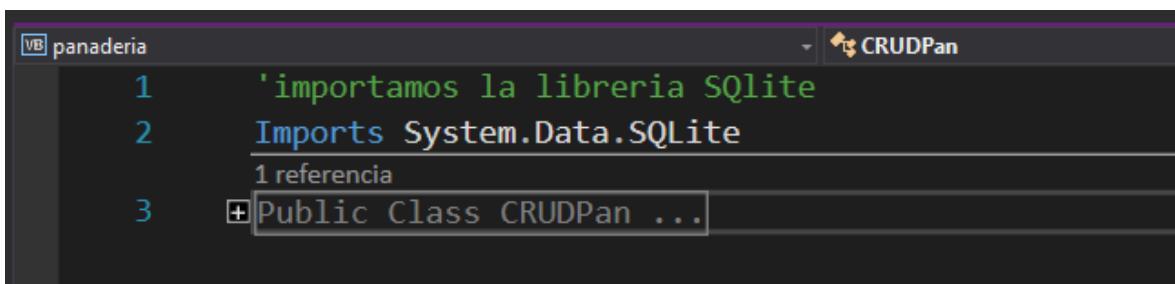
Solo aparecerá lo siguiente:



```
crud.vb*  X
VB panaderia  CRUDPan
Module CRUDPan
End Module
```

Estructurando el Módulo CRUDPan

Borramos esas líneas y procedemos programar nuestro CRUD correctamente:



```
'importamos la libreria SQLite
Imports System.Data.SQLite
Public Class CRUDPan ...
```

```
Public Class CRUDPan
    'Creamos la conexion de la base de datos
    Dim con As New SQLiteConnection("Data Source=panaderia.db; Version=3")
```

```
'creamos el procedimiento de conexion
Sub conexion()
    'usamos el Try para capturar errores
    Try
        'abrimos la conexion
        con.Open()
        'mensaje si es que se establece la conexion
        MsgBox("Conexion establecida correctamente", MsgBoxStyle.Information, "Santa Monica")
        'cerramos la conexion
        con.Close()
    Catch ex As Exception
        'mensaje si es que sale algun error
        MsgBox("Ha ocurrido un error, posible causa: " & ex.Message, MsgBoxStyle.Critical, "Santa Monica")
    End Try
End Sub
```

```

' creamos un procedimiento para la consulta y le indicamos que debe obtener 2 parametros
' para ejecutarse correctamente
2 referencias
Sub consulta(ByVal tabla As DataGridView, ByVal sql As String)
    'usamos Try para capturar los errores
    Try
        ' creamos un objeto de tipo SQLiteDataAdapter con dos parametros
        Dim sqlda As New SQLiteDataAdapter(sql, con)
        ' creamos ootr objeto de tipo DataTable
        Dim dt As New DataTable
        'psamos la informacion del SQLiteDataAdapter al DataTable mediante la propiedad Fill
        sqlda.Fill(dt)
        ' aqui mostramos los datos en el DataGridView
        tabla.DataSource = dt
    Catch ex As Exception
        MsgBox("Ha ocurrido un en la consulta, posible causa: " & ex.Message, MsgBoxStyle.Critical, "Santa Monica")
    End Try
End Sub

' creamos un procedimiento para Agregar,Actualizar y Eliminar,
' le indicamos que debe pedir 2 parametros para ejecutarse correctamente
1 referencia
Sub operaciones(ByVal tabla As DataGridView, ByVal sql As String)
    'abrimos la conexion
    con.Open()
    'usamos el Try para capturar errores
    Try
        ' creamos un objeto de tipo Command que almacenara las intrucciones
        'necesita 2 parametros para ejecutarse
        Dim cmd As New SQLiteCommand(sql, con)
        'ejecutamos la instruccion mediante la proiedad ExecuteNonQuery
        cmd.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox("Ha ocurrido un en la operacion, posible causa: " & ex.Message, MsgBoxStyle.Critical, "Santa Monica")
    End Try
    'cerramos la conexion
    con.Close()
End Sub
End Class

```

Código fuente CRUDPan

```

'importamos la libreria SQLite
Imports System.Data.SQLite
Public Class CRUDPan
    'Creamos la conexion de la base de datos
    Dim con As New SQLiteConnection("Data Source=panaderia.db; Version=3")

    'creamos el procedimiento de conexion
    Sub conexion()
        'usamos el Try para capturar errores
        Try
            'abrimos la conexion
            con.Open()
            'mensaje si es que se establece la conexion
            MsgBox("Conexion establecida correctamente",
                MsgBoxStyle.Information, "Santa Monica")
            'cerramos la conexion
            con.Close()
        Catch ex As Exception
            'mensaje si es que sale algun error
        End Try
    End Sub
End Class

```

```

        MsgBox("Ha ocurrido un error, posible causa: " & ex.Message,
MsgBoxStyle.Critical, "Santa Monica")
    End Try
End Sub

' creamos un procedimiento para la consulta y le indicamos que debe obtener 2
parametros
' para ejecutarse correctamente
Sub consulta(ByVal tabla As DataGridView, ByVal sql As String)
    'usamos Try para capturar los errores
    Try
        ' creamos un objeto de tipo SQLiteDataAdapter con dos parametros
        Dim sqlda As New SQLiteDataAdapter(sql, con)
        ' creamos ootr objeto de tipo DataTable
        Dim dt As New DataTable
        ' psamos la informacion del SQLiteDataAdapter al DataTable mediante
la propiedad Fill
        sqlda.Fill(dt)
        ' aqui mostramos los datos en el DataGridView
        tabla.DataSource = dt
    Catch ex As Exception
        MsgBox("Ha ocurrido un en la consulta, posible causa: " & ex.Message,
MsgBoxStyle.Critical, "Santa Monica")
    End Try
End Sub

' creamos un procedimiento para Agregar,Actualizar y Eliminar,
' le indicamos que debe pedir 2 parametros para ejecutarse correctamente
Sub operaciones(ByVal tabla As DataGridView, ByVal sql As String)
    'abrimos la conexion
    con.Open()
    'usamos el Try para capturar errores
    Try
        ' creamos un objeto de tipo Command que almacenara las intrucciones
        'necesita 2 parametros para ejecutarse
        Dim cmd As New SQLiteCommand(sql, con)
        'ejecutamos la instruccion mediante la proiedad ExecuteNonQuery
        cmd.ExecuteNonQuery()
    Catch ex As Exception
        MsgBox("Ha ocurrido un en la operacion, posible causa: " &
ex.Message, MsgBoxStyle.Critical, "Santa Monica")
    End Try
    'cerramos la conexion
    con.Close()
End Sub
End Class

```

Windows Forms Principales

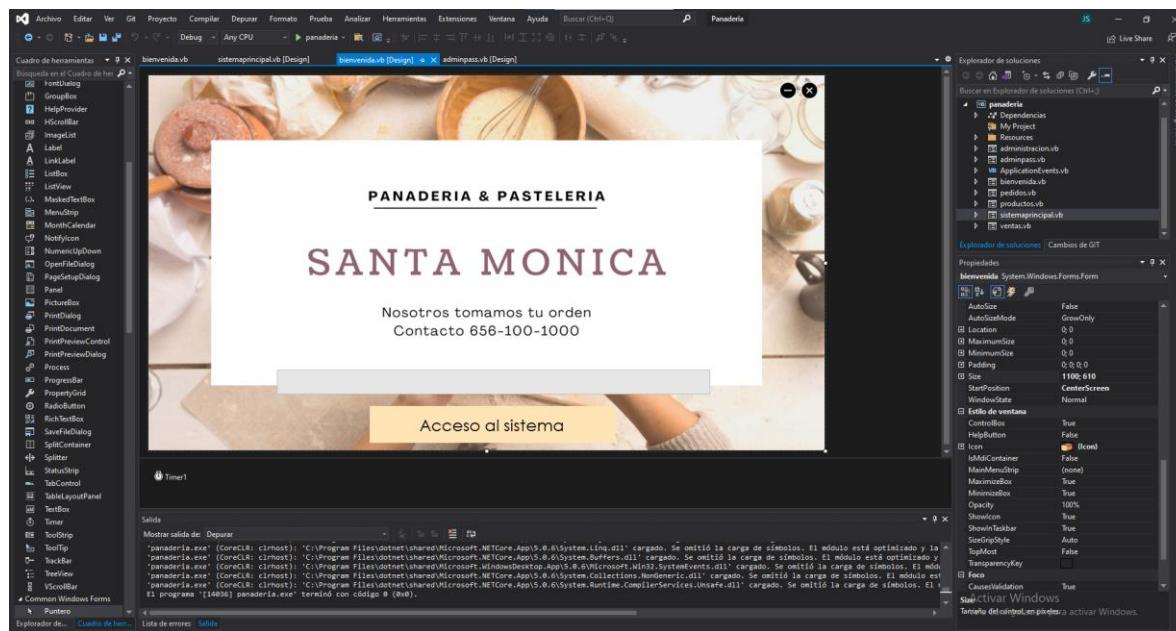
Como se mencionó al inicio del presente documento, este proyecto se trata de la creación, gestión y conexión entre, SQLite y Visual Basic.NET; Se mostrarán capturas de los formularios con su respectivo diseño y su código fuente.

- La versión usada de SQLite; SQLite3. Y la versión del software para programar la aplicación es, Microsoft Visual Studio Community 2019.

Este primer formulario, solo será un formulario de bienvenida, es importante decir que es nuestra presentación y como se verá para aquellos que experimentaran con el proyecto.

Formulario 1: Form bienvenida.vb

Este formulario consta de un botón para mostrar el formulario donde será el sistema principal, una barra de progreso y dos botones; función minimizar y cerrar el formulario.



Código fuente Form bienvenida.vb

```
Imports System.Runtime.InteropServices
Public Class bienvenida
    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
        End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
        End Sub
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        ProgressBar1.Value += 1
        If Me.Opacity < 1 Then
            Me.Opacity += 0.05
            Bbar.Visible = False
        End If

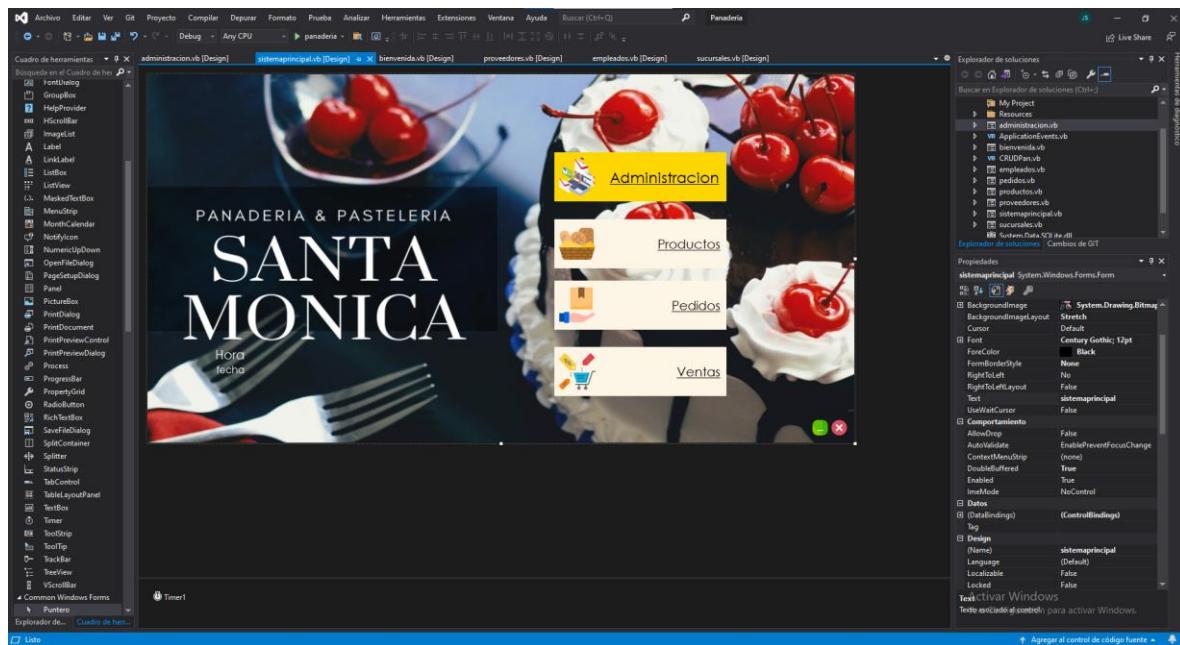
        If ProgressBar1.Value = 50 And 80 Then
            ProgressBar1.ForeColor = Color.Yellow
        End If

        If ProgressBar1.Value > 80 Then
            ProgressBar1.ForeColor = Color.Green
        End If

        If ProgressBar1.Value = 100 Then
            Bbar.Visible = True
            Timer1.Stop()
        End If
    End Sub
    Private Sub bienvenida_Load(sender As Object, e As EventArgs) Handles Me.Load
        ProgressBar1.Value = 0
        Me.Opacity = 0
        Timer1.Start()
    End Sub
    Private Sub Bbar_Clic(sender As Object, e As EventArgs) Handles Bbar.Clic
        sistemaprincipal.Show()
        Me.Hide()
    End Sub
    Private Sub Bcerrar_Clic(sender As Object, e As EventArgs) Handles Bcerrar.Clic
        Me.Close()
    End Sub
    Private Sub Bminimizar_Clic(sender As Object, e As EventArgs) Handles
Bminimizar.Clic
        Me.WindowState = FormWindowState.Minimized
    End Sub
    Private Sub bienvenida_MouseMove(sender As Object, e As MouseEventArgs) Handles
 MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
End Class
```

Formulario 2: Form sistemaprincipal.vb

En este formulario se encontrar 4 botones; Los botones principales abren otros formularios dando acceso a los datos.



Código fuente Form sistemaprincipal.vb

```
Imports System.Runtime.InteropServices
Public Class sistemaprincipal
    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
    End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
    End Sub
    Private Sub sistemaprincipal_MouseMove(sender As Object, e As MouseEventArgs)
Handles MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
    Private Sub sistemaprincipal_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Timer1.Enabled = True
    End Sub
    Private Sub Badmin_Clic(sender As Object, e As EventArgs) Handles Badmin.Clic
        administracion.Show()
    End Sub
    Private Sub Bproductos_Clic(sender As Object, e As EventArgs) Handles
Bproductos.Clic
        productos.Show()
    End Sub
```

```

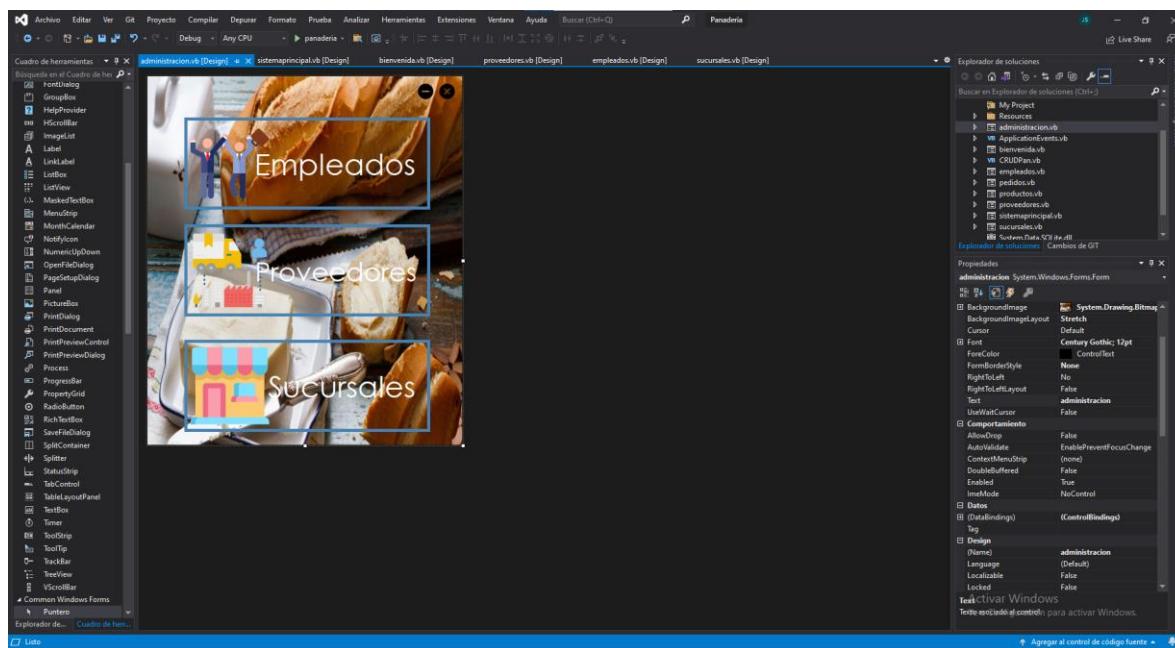
    Private Sub Bpedidos_Clic(sender As Object, e As EventArgs) Handles Bpedidos.Clic
        pedidos.Show()
    End Sub
    Private Sub Bventas_Clic(sender As Object, e As EventArgs) Handles Bventas.Clic
        ventas.Show()
    End Sub
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        Lhora.Text = DateTime.Now.ToString("hh:mm:ss")
        Lfecha.Text = DateTime.Now.ToString("dd/MM/yyyy")
    End Sub
    Private Sub Bcerrar_Clic(sender As Object, e As EventArgs) Handles Bcerrar.Clic
        bienvenida.Close()
        Me.Close()
    End Sub
    Private Sub Bregreso_Clic(sender As Object, e As EventArgs)
        bienvenida.Show()
        Me.Close()
    End Sub
    Private Sub Bminimizar_Clic(sender As Object, e As EventArgs) Handles Bminimizar.Clic
        Me.WindowState = FormWindowState.Minimized
    End Sub
    Private Sub Badmin_Paint(sender As Object, e As PaintEventArgs) Handles Badmin.Paint
        Dim botonpath As Drawing2D.GraphicsPath = New Drawing2D.GraphicsPath()
        Dim botonrect As Rectangle = Badmin.ClientRectangle
        botonrect.Inflate(0, 40)
        botonpath.AddEllipse(botonrect)
        Badmin.Region = New Region(botonpath)
    End Sub
    Private Sub Bproductos_Paint(sender As Object, e As PaintEventArgs) Handles Bproductos.Paint
        Dim botonpath As Drawing2D.GraphicsPath = New Drawing2D.GraphicsPath()
        Dim botonrect As Rectangle = Bproductos.ClientRectangle
        botonrect.Inflate(0, 40)
        botonpath.AddEllipse(botonrect)
        Bproductos.Region = New Region(botonpath)
    End Sub
    Private Sub Bpedidos_Paint(sender As Object, e As PaintEventArgs) Handles Bpedidos.Paint
        Dim botonpath As Drawing2D.GraphicsPath = New Drawing2D.GraphicsPath()
        Dim botonrect As Rectangle = Bpedidos.ClientRectangle
        botonrect.Inflate(0, 40)
        botonpath.AddEllipse(botonrect)
        Bpedidos.Region = New Region(botonpath)
    End Sub
    Private Sub Bventas_Paint(sender As Object, e As PaintEventArgs) Handles Bventas.Paint
        Dim botonpath As Drawing2D.GraphicsPath = New Drawing2D.GraphicsPath()
        Dim botonrect As Rectangle = Bventas.ClientRectangle
        botonrect.Inflate(0, 40)
        botonpath.AddEllipse(botonrect)
        Bventas.Region = New Region(botonpath)
    End Sub
End Class

```

Formulario 3: Form administracion.vb

El tercer formulario, tendrá solo 3 botones, cada botón llama a otro formulario, estos son los que entablaran la conexión con nuestra base de datos, los formularios mandados a llamar son:

- Empleados
- Proveedores
- Sucursales



Código fuente Form administracion.vb

```
Imports System.Runtime.InteropServices
Public Class administracion
    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
        End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
        End Sub
    Private Sub administracion_MouseMove(sender As Object, e As MouseEventArgs)
Handles MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
    Private Sub administracion_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        End Sub
    Private Sub Bcerrar_Clic(sender As Object, e As EventArgs) Handles Bcerrar.Clic
        Me.Close()
    End Sub
    Private Sub Bminimizar_Clic(sender As Object, e As EventArgs) Handles
Bminimizar.Clic
        Me.WindowState = FormWindowState.Minimized
    End Sub
    Private Sub Bempleado_Clic(sender As Object, e As EventArgs) Handles
Bempleado.Clic
        empleados.Show()
        Me.Close()
    End Sub
    Private Sub Bproveedor_Clic(sender As Object, e As EventArgs) Handles
Bproveedor.Clic
        proveedores.Show()
        Me.Close()
    End Sub
    Private Sub Bsucursales_Clic(sender As Object, e As EventArgs) Handles
Bsucursales.Clic
        sucursales.Show()
        Me.Close()
    End Sub
End Class
```

Notas Importantes

- Cuando se quiera **agregar datos**, puede o no poner todos los datos, con respecto a eso no hay problema a excepción del PRIMARY KEY (son los ID).
- Cuando se quiera **eliminar datos**, solo hay que escribir el ID en el TextBox ID.
- Cuando se quiera **actualizar datos**, se ponen todos los datos en los TextBox y con ello el dato a actualizar, si no se ponen todos se borrar los datos y solo se actualizar el dato deseado, de modo que las demás columnas queden vacías.

Implementando las Sentencias SQL (CRUD)

Cabe recalcar que cada código esta explicado dentro de la solución mediante comentarios.

Formulario 4: Form empleados.vb

```
Imports System.Runtime.InteropServices
Public Class empleados
    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
        End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
        End Sub
    Private Sub empleados_MouseMove(sender As Object, e As MouseEventArgs) Handles
 MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
    Dim obj As New CRUDPan
    Private Sub empleados_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        'mensaje si es que se establece la conexion
        MsgBox("Conexion establecida correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Private Sub Bagregar_Click(sender As Object, e As EventArgs) Handles
Bagregar.Click
        'Creamos la variable Sql que guardar la instruccion de tipo SQL
        Dim Sql As String = "Insert Into empleado (idEmpleado, nombre, apellidos,
puesto, sueldo) Values " & "(" & TbId.Text & ", " &
        " " & TbNombre.Text & ", " &
        " " & TbApellidos.Text & ", " &
        " " & TbPuesto.Text & ", " &
        " " & TbSueldo.Text & ");"
        obj.operaciones(DGempleado, Sql)
        MsgBox("Datos guardamos correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Private Sub Bactualizar_Click(sender As Object, e As EventArgs) Handles
Bactualizar.Click
        'Creamos la variable Sql que actualizara la instruccion de tipo SQL
        Dim Sql As String = "Update empleado Set nombre=' " & TbNombre.Text &
",apellidos=' " & TbApellidos.Text & ",puesto=' " & TbPuesto.Text & ",sueldo=' " &
TbSueldo.Text & " ' Where idEmpleado = " & TbId.Text & ""
        obj.operaciones(DGempleado, Sql)
        leerBD()
    End Sub
```

```

    Private Sub Beliminar_Click(sender As Object, e As EventArgs) Handles
Beliminar.Click
    'Creamos la variable Sql que eliminara la instruccion de tipo SQL
    Dim Sql As String = "Delete From empleado Where idEmpleado = " & TbId.Text &
"""
    obj.operaciones(DGEmpleado, Sql)
    leerBD()
End Sub
Sub leerBD()
    obj.conexion()
    Dim Sql As String = "Select * From empleado"
    obj.consulta(DGEmpleado, Sql)
End Sub
    Private Sub Brefresh_Click(sender As Object, e As EventArgs) Handles
Brefresh.Click
    leerBD()
End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
    Me.Close()
    administracion.Show()
End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
    Me.WindowState = FormWindowState.Minimized
End Sub
End Class

```

Formulario 5: Form proveedores.vb

```
Imports System.Runtime.InteropServices
Public Class proveedores

    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
        End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
        End Sub

    Private Sub proveedores_MouseMove(sender As Object, e As MouseEventArgs) Handles
 MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
    Dim obj As New CRUDPan
    Private Sub proveedores_Load(sender As Object, e As EventArgs) Handles
 MyBase.Load
        'mensaje si es que se establece la conexion
        MsgBox("Conexion establecida correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Sub leerBD()
        obj.conexion()
        Dim Sql As String = "Select * From proveedores"
        obj.consulta(DGproveedores, Sql)
    End Sub

    Private Sub Brefresh_Click(sender As Object, e As EventArgs) Handles
Brefresh.Click
        leerBD()
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.WindowState = FormWindowState.Minimized
    End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        Me.Close()
        administracion.Show()
    End Sub

    Private Sub Bagregar_Click(sender As Object, e As EventArgs) Handles
Bagregar.Click
        'Creamos la variable Sql que guardar la instruccion de tipo SQL
        Dim Sql As String = "Insert Into proveedores (idProveedor, nomProveedor,
nomRepartidor, unidadProveedor) Values " & "(" & TbId.Text & "," &
"'" & TbNombre.Text & "','" &
"'" & TbRepartidor.Text & "','" &
"'" & TbUnidad.Text & "');"
```

```

        obj.operaciones(DGproveedores, Sql)
        MsgBox("Datos guardamos correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub

    Private Sub Beliminar_Click(sender As Object, e As EventArgs) Handles
Beliminar.Click
        'Creamos la variable Sql que eliminara la instruccion de tipo SQL
        Dim Sql As String = "Delete From proveedores Where idProveedor = " &
TbId.Text & ""
        obj.operaciones(DGproveedores, Sql)
        leerBD()
    End Sub

    Private Sub Bactualizar_Click(sender As Object, e As EventArgs) Handles
Bactualizar.Click
        'Creamos la variable Sql que actualizara la instruccion de tipo SQL
        Dim Sql As String = "Update proveedores Set nomProveedor=''" & TbNombre.Text
& "'' ,nomRepartidor=''" & TbRepartidor.Text & "'' ,unidadProveedor=''" & TbUnidad.Text &
"'' Where idProveedor = " & TbId.Text & ""
        obj.operaciones(DGproveedores, Sql)
        leerBD()
    End Sub
End Class

```

Formulario 6: Form sucursales.vb

```
Imports System.Runtime.InteropServices
Public Class sucursales
    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
    End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
    End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        Me.Close()
        administracion.Show()
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.WindowState = FormWindowState.Minimized
    End Sub

    Private Sub sucursales_MouseMove(sender As Object, e As MouseEventArgs) Handles
 MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
    Dim obj As New CRUDPan
    Private Sub sucursales_Load(sender As Object, e As EventArgs) Handles
 MyBase.Load
        'mensaje si es que se establece la conexion
        MsgBox("Conexion establecida correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Sub leerBD()
        obj.conexion()
        Dim Sql As String = "Select * From sucursal"
        obj.consulta(DGsucursales, Sql)
    End Sub

    Private Sub Bagregar_Click(sender As Object, e As EventArgs) Handles
Bagregar.Click
        'Creamos la variable Sql que guardar la instruccion de tipo SQL
        Dim Sql As String = "Insert Into sucursal (idSucursal, nomSucursal,
dirSucursal) Values " & "(" & TbId.Text & ", " &
        " " & TbNomSucursal.Text & ", " &
        " " & TbDirSucursal.Text & ");"
        obj.operaciones(DGsucursales, Sql)
        MsgBox("Datos guardados correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub

    Private Sub Beliminar_Click(sender As Object, e As EventArgs) Handles
Beliminar.Click
```

```

'Creamos la variable Sql que eliminara la instruccion de tipo SQL
Dim Sql As String = "Delete From sucursal Where idSucursal = " & TbId.Text &
"""

    obj.operaciones(DGsucursales, Sql)
    leerBD()
End Sub

Private Sub Bactualizar_Click(sender As Object, e As EventArgs) Handles
Bactualizar.Click
    'Creamos la variable Sql que actualizara la instruccion de tipo SQL
    Dim Sql As String = "Update sucursal Set nomSucursal=''" & TbNomSucursal.Text
& " ',dirSucursal=''" & TbDirSucursal.Text & "' Where idSucursal = " & TbId.Text & ""
    obj.operaciones(DGsucursales, Sql)
    leerBD()
End Sub

Private Sub Brefresh_Click(sender As Object, e As EventArgs) Handles
Brefresh.Click
    leerBD()
End Sub
End Class

```

Formulario 7: Form productos.vb

```
Imports System.Runtime.InteropServices
Public Class productos
    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
        End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
        End Sub
    Private Sub productos_MouseMove(sender As Object, e As MouseEventArgs) Handles
 MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
    Dim obj As New CRUDPan
    Private Sub productos_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        'mensaje si es que se establece la conexion
        MsgBox("Conexion establecida correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Sub leerBD()
        obj.conexion()
        Dim Sql As String = "Select * From productos"
        obj.consulta(DGproductos, Sql)
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.WindowState = FormWindowState.Minimized
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        sistemaprincipal.Show()
        Me.Close()
    End Sub

    Private Sub Bagregar_Click(sender As Object, e As EventArgs) Handles
Bagregar.Click
        'Creamos la variable Sql que guardar la instruccion de tipo SQL
        Dim Sql As String = "Insert Into productos (idProducto, nomProducto,
precioUnidad, tiempoPreparacion) Values " & "(" & TbId.Text & "," &
" " & TbNomProducto.Text & "," &
" " & TbPrecioUni.Text & "," &
" " & TbPreparacion.Text & ");"
        obj.operaciones(DGproductos, Sql)
        MsgBox("Datos guardamos correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
```

```

    Private Sub Beliminar_Click(sender As Object, e As EventArgs) Handles
Beliminar.Click
    'Creamos la variable Sql que eliminara la instruccion de tipo SQL
    Dim Sql As String = "Delete From productos Where idProducto = " & TbId.Text
& ""
    obj.operaciones(DGproductos, Sql)
    leerBD()
End Sub

    Private Sub Bactualizar_Click(sender As Object, e As EventArgs) Handles
Bactualizar.Click
    'Creamos la variable Sql que actualizara la instruccion de tipo SQL
    Dim Sql As String = "Update productos Set nomProducto=' " &
TbNomProducto.Text & "',precioUnidad=' " & TbPrecioUni.Text & "',tiempoPreparacion=' "
& TbPreparacion.Text & " ' Where idProducto = " & TbId.Text & ""
    obj.operaciones(DGproductos, Sql)
    leerBD()
End Sub

    Private Sub Brefresh_Click(sender As Object, e As EventArgs) Handles
Brefresh.Click
    leerBD()
End Sub
End Class

```

Formulario 8: Form pedidos.vb

```
Imports System.Runtime.InteropServices
Public Class pedidos
    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
        End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
        End Sub
    Private Sub pedidos_MouseMove(sender As Object, e As MouseEventArgs) Handles
 MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
    Dim obj As New CRUDPan
    Private Sub pedidos_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        'mensaje si es que se establece la conexion
        MsgBox("Conexion establecida correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Sub leerBD()
        obj.conexion()
        Dim Sql As String = "Select * From pedidos"
        obj.consulta(DGpedidos, Sql)
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.WindowState = FormWindowState.Minimized
    End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        sistemaprincipal.Show()
        Me.Close()
    End Sub
    Private Sub Bagregar_Click(sender As Object, e As EventArgs) Handles
Bagregar.Click
        'Creamos la variable Sql que guardar la instruccion de tipo SQL
        Dim Sql As String = "Insert Into pedidos (idPedido, cliente, producto,
unidades, precioTotal) Values " & "(" & TbId.Text & "," &
        "" & TbCliente.Text & "," &
        "" & TbProductos.Text & "," &
        "" & TbUnidades.Text & "," &
        "" & TbTotal.Text & ");"
        obj.operaciones(DGpedidos, Sql)
        MsgBox("Datos guardamos correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Private Sub Beliminar_Click(sender As Object, e As EventArgs) Handles
Beliminar.Click
        'Creamos la variable Sql que eliminara la instruccion de tipo SQL
```

```

        Dim Sql As String = "Delete From pedidos Where idPedido = " & TbId.Text & ""
        obj.operaciones(DGpedidos, Sql)
        leerBD()
    End Sub
    Private Sub Bactualizar_Click(sender As Object, e As EventArgs) Handles
Bactualizar.Click
        'Creamos la variable Sql que actualizara la instruccion de tipo SQL
        Dim Sql As String = "Update pedidos Set cliente=''" & TbCliente.Text &
'',producto=''' & TbProductos.Text & ''',unidades=''' & TbUnidades.Text &
'',precioTotal=''' & TbTotal.Text & ''' Where idPedido= " & TbId.Text & """
        obj.operaciones(DGpedidos, Sql)
        leerBD()
    End Sub
    Private Sub Brefresh_Click(sender As Object, e As EventArgs) Handles
Brefresh.Click
        leerBD()
    End Sub
End Class

```

Formulario 9: Form ventas.vb

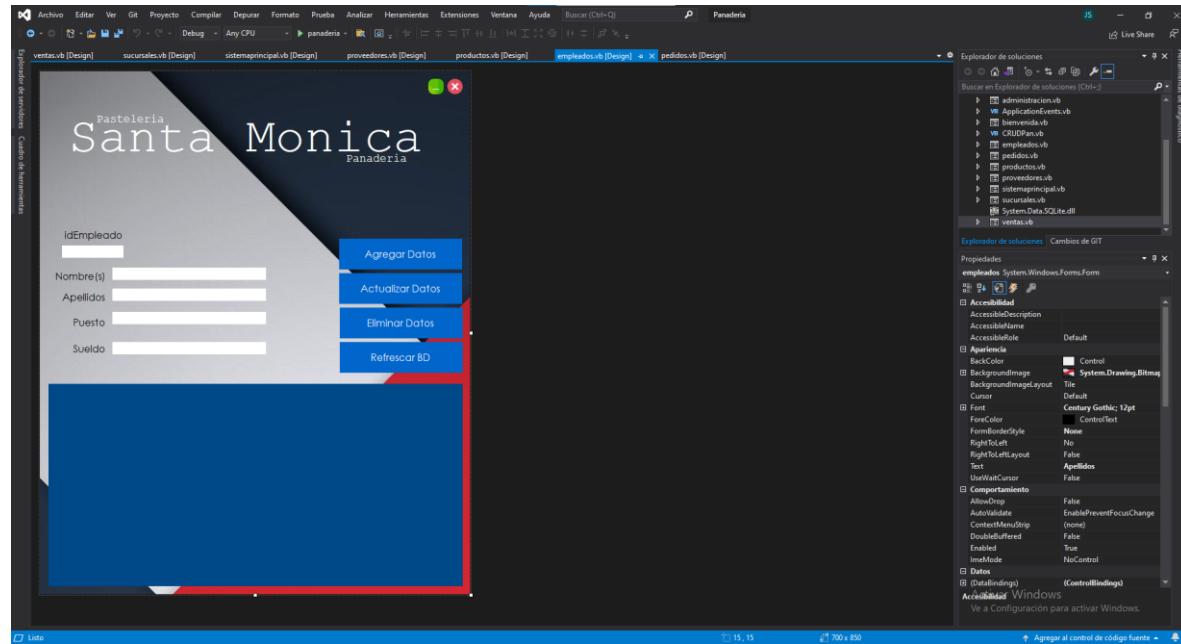
```
Imports System.Runtime.InteropServices
Public Class ventas
    <DllImport("user32.DLL", EntryPoint:="ReleaseCapture")>
    Private Shared Sub ReleaseCapture()
        End Sub
    <DllImport("user32.DLL", EntryPoint:="SendMessage")>
    Private Shared Sub SendMessage(ByVal hWnd As System.IntPtr, ByVal wMsg As
Integer, ByVal wParam As Integer, ByVal lParam As Integer)
        End Sub

    Private Sub ventas_MouseMove(sender As Object, e As MouseEventArgs) Handles
 MyBase.MouseMove
        ReleaseCapture()
        SendMessage(Me.Handle, &H112&, &HF012&, 0)
    End Sub
    Private Sub ventas_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        'mensaje si es que se establece la conexion
        MsgBox("Conexion establecida correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Dim obj As New CRUDPan
    Sub leerBD()
        obj.conexion()
        Dim Sql As String = "Select * From ventas"
        obj.consulta(DGventas, Sql)
    End Sub
    Private Sub Bagregar_Click(sender As Object, e As EventArgs) Handles
Bagregar.Click
        'Creamos la variable Sql que guardar la instruccion de tipo SQL
        Dim Sql As String = "Insert Into ventas (idTicket, producto, precioUnidad,
piezasVendidas, totalTicket, fecha) Values " & "(" & TbId.Text & "," &
        "" & TbProducto.Text & "," &
        "" & TbPreUnidad.Text & "," &
        "" & TbPieVendidas.Text & "," &
        "" & TbTotalTicket.Text & "," &
        "" & TbFecha.Text & ");"
        obj.operaciones(DGventas, Sql)
        MsgBox("Datos guardados correctamente", MsgBoxStyle.Information, "Santa
Monica")
        leerBD()
    End Sub
    Private Sub Brefresh_Click(sender As Object, e As EventArgs) Handles
Brefresh.Click
        leerBD()
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.WindowState = FormWindowState.Minimized
    End Sub
```

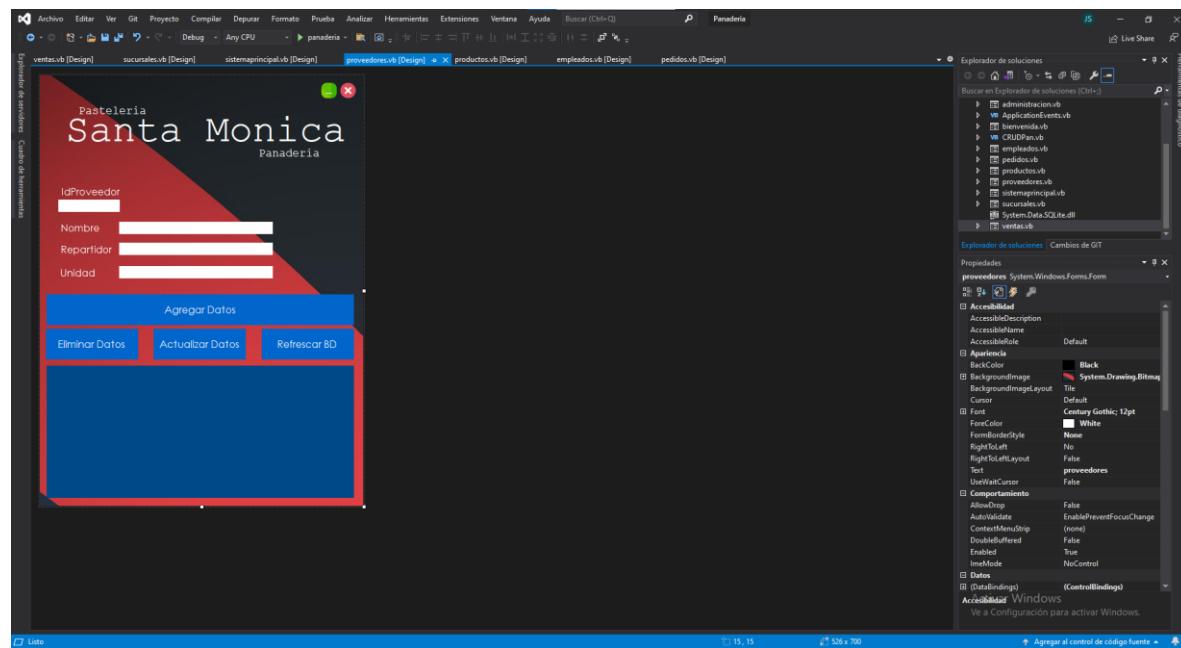
```
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
    sistemaprincipal.Show()
    Me.Close()
End Sub
End Class
```

Windows Forms Material Design: Sistema Administración

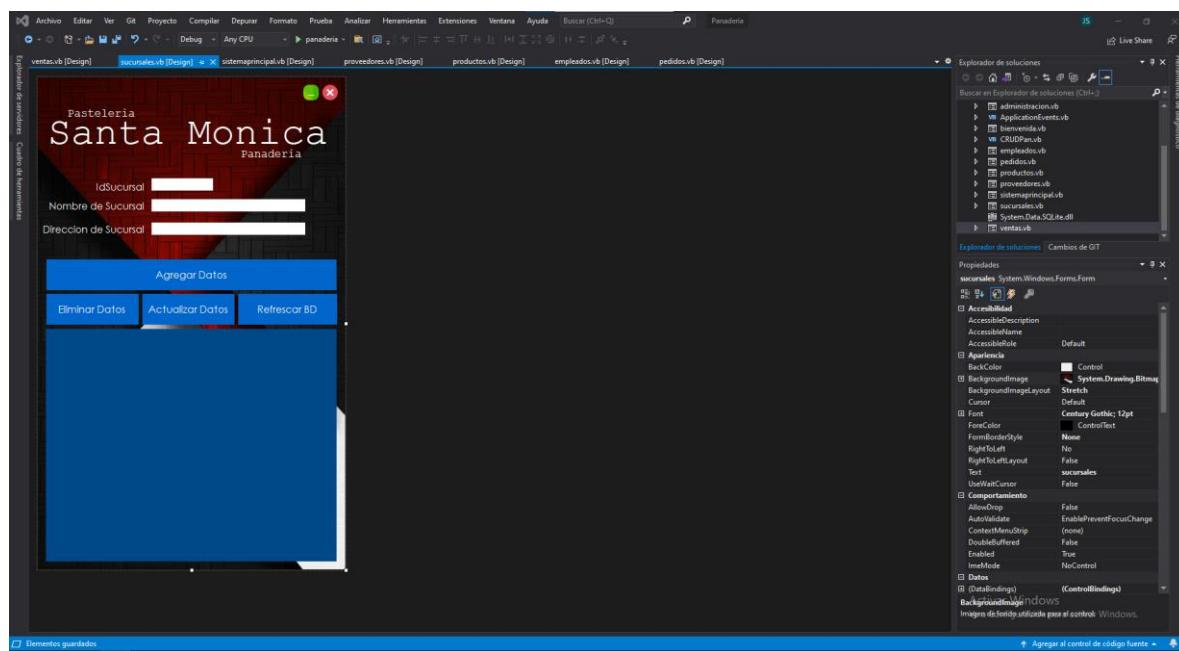
Formulario 4: Form empleados.vb



Formulario 5: Form proveedores.vb

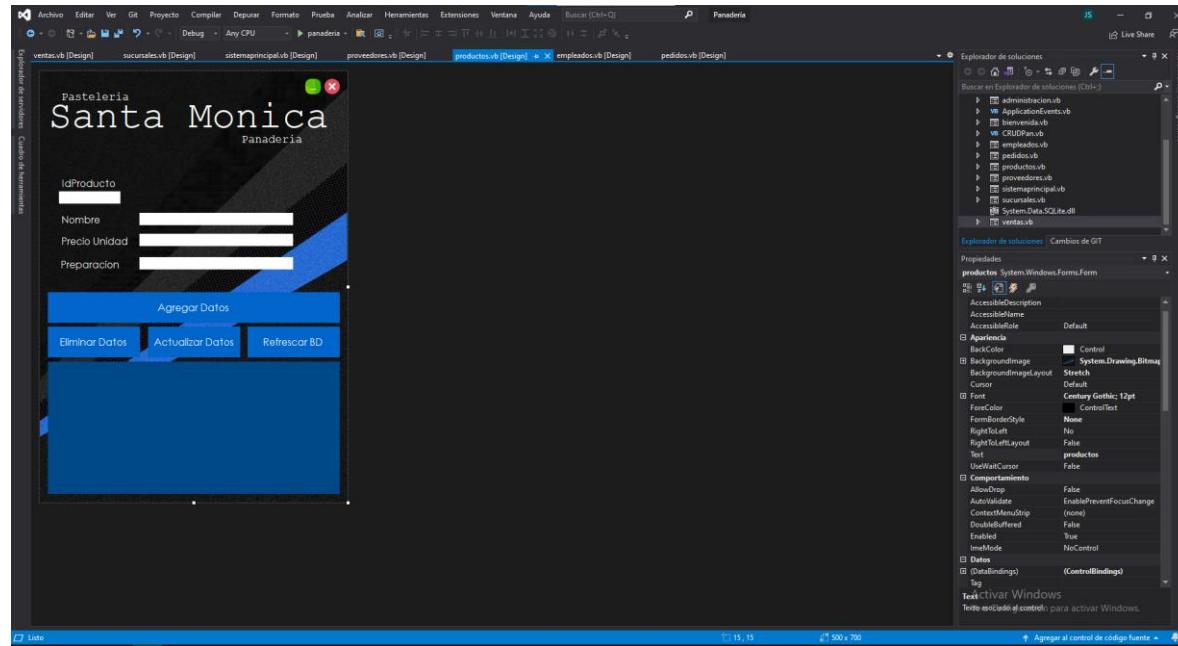


Formulario 6: Form sucursales.vb

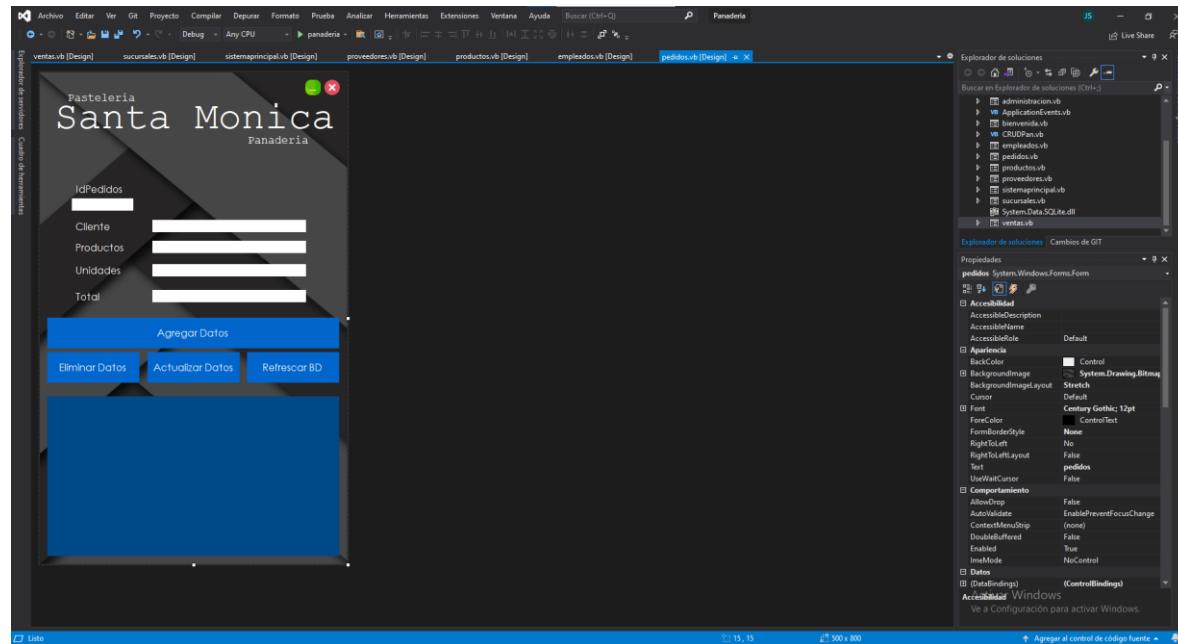


Windows Forms Material Design: Sistema Principal

Formulario 7: Form productos.vb



Formulario 8: Form pedidos.vb



Formulario 9: Form ventas.vb

