



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

BASES DE DATOS

Proyecto Final:

Sistema de ventas de Papelería

Equipo:

**Reyes Avila David
Salinas Romero Daniel**

Profesor:

Fernando Arreola Franco

Lunes 09 de Agosto del 2021



Introducción:

Para el proyecto final de la materia se nos encarga suponer que ya somos un equipo encargado al desarrollo de Software contratado para resolver un problema, en este aspecto no tenemos problema tenemos una experiencia previa imaginando estos casos en materias anteriores, como lo fue ingeniería de Software. En este caso consideramos que nos vemos un poco limitados por el tiempo y las demás actividades académicas que debemos realizar, además en esta ocasión el objetivo no es realizar toda la documentación que debe hacerse en un proyecto de software con todas sus fases de desarrollo, pero si esperamos incluir unas en el proyecto para darle un aspecto más formal.

Además, consideramos que tenemos la gran ventaja de que nosotros, los dos integrantes del equipo ya nos conocemos de antes, de hecho desde el primer semestre; con esto queremos decir que ya conocemos las fortalezas y debilidades del otro, y permite que exista la confianza para delegar trabajo y manejar tiempos. Con respecto a los requerimientos del proyecto, como primeras impresiones parece laborioso, pero es un hecho de que todo lo hemos visto en clase. Tenemos dudas sobre si entendemos bien el contexto del problema, pero esperamos que se puedan solventar con la ayuda del profesor.

Donde lo que se nos solicita es dar solución al siguiente problema:

Una cadena de papelerías busca innovar la manera en que almacena su información, y los contratan para que desarrollen los sistemas informáticos para satisfacer los siguientes requerimientos:

Se desea tener almacenados datos como la razón social, domicilio, nombre y teléfonos de los proveedores, rfc, nombre, domicilio y al menos un email de los clientes. Es necesario tener un inventario de los productos que se venden, en el que debe guardarse el código de barras, precio al que fue comprado el producto, fecha de compra y cantidad de ejemplares en la bodega (stock). Se desea guardar la marca, descripción y precio de los regalos, artículos de papelería, impresiones y recargas, siempre y cuando se tenga su correspondiente registro en el inventario. Debe también guardarse el número de venta, fecha de venta y la cantidad total a pagar de la venta, así como la cantidad de cada artículo y precio total a pagar por artículo.

Además, se requiere que cuente con otros requerimientos tales como: poder regresar una utilidad por cada producto, tener un buen manejo de stock en la lista de productos, aunado a poder proporcionar un total de ganancias por fechas, poder distinguir los productos cuyo stock esté por terminarse, ser capaz proveer una vista de factura de compra, un buen manejo de información de las entidades y satisfacer con los principios de diseño y requerimientos.

Objetivos:

- Realizar el análisis del problema para la obtención del diseño conceptual de la base de datos que satisfaga el problema.
- Poner en práctica los conceptos revisados en clase.
- Elaborar las sentencias SQL que permitan la creación de tablas e inserción de elementos.
- Elaborar los documentos respectivos a cada una de las fases del diseño.
- De ser posible implementar la solución con la interfaz gráfica.
- Elaborar y preparar la presentación del proyecto final.

Propuesta de solución:

Para esta parte del proyecto, una vez teniendo el diseño, estructura y la base de datos con registros, se procedió con la segunda parte del proyecto. La cual consiste en implementar una interfaz gráfica que nos permita enlazar nuestra base de datos con una aplicación web o con una aplicación móvil. Dicha aplicación debe cumplir con los requisitos de: agregar la información de clientes, poder ingresar ventas de hasta tres artículos disponibles seleccionables, ingresar el número de artículos, calcular el costo total de cada artículo seleccionado así como poder calcular el costo total de la venta. Los objetivos de la parte 2 del proyecto son poder implementar la interfaz gráfica que pueda enlazar la base de datos pudiendo cumplir con los requisitos antes mencionados, así como también poder ingresar toda información en la base de datos respetando las restricciones de integridad.

Siendo que se trata de una base de datos que intenta resolver la problemática de un manejo o gestión de una papelería, al ser un caso similar a una tienda de abarrotes nos pareció buena propuesta investigar ejemplos de un sistema para una tienda de abarrotes. Encontramos como propuesta el crear una aplicación móvil que cumpla con todas las condiciones y objetivos del proyecto pudiéndose enlazar a la base de datos hecha en postgresql. Para dicho enlace se utilizó el servidor gratuito Heroku que nos permitió crear un repositorio para mejor modificación de archivos en equipo, compatible con lenguaje PHP el cual fue necesario para poder desarrollar la interfaz de la aplicación móvil. El código se desarrolló en visual studio code en conjunto con Android studio. Cabe mencionar que la implementación de la interfaz móvil fue basada en una serie de videos del canal "KAD" de youtube en la cual utilizamos la estructura base del proyecto de una tienda de abarrotes en aplicación móvil y modificando a nuestras necesidades del proyecto. Las fuentes de dicha serie de videos la compartimos en el apartado de referencias. Puesto que el objetivo principal del proyecto está centrado en la creación de una base de datos con una aplicación más formal, nos pareció buena propuesta de resolución dada nuestra situación tan apretada de tiempo y de falta de conocimientos en PHP.

Referencias:

- KAD. (Febrero 2, 2019). Sistema de Tienda. Julio 14, 2021, de KAD Sitio web:
 1. Instalación y Configuración del Servidor | Sistema de Tienda en Android con ...

Plan de trabajo.

Para la realización de este proyecto se tiene planeado dar inicio oficial el día 5 de julio de 2021. Como actividades a cumplir consideramos que las más importantes son:

1. La realización del modelo entidad relación.
2. El modelo relacional.
3. Normalización de tablas.
4. Creación de tablas y llenado de las mismas en PostgreSQL.
5. Programación a nivel de base de datos.
6. Implementación de manera gráfica.

En general la dinámica de las actividades está planeada con la reunión de ambos integrantes del equipo vía Zoom, cada día de la semana dos horas en promedio, para realizar los análisis, crear los entregables y discutir sobre el proyecto. Además, se piensa realizar un análisis previo y posterior a cada actividad y de ser posible consultar al profesor por cada una de las 5 actividades anteriormente mencionadas.

Para que las actividades se realicen y se trabaje en conjunto, se creará un repositorio en GitHub donde se guardan los adelantos del proyecto, principalmente del Código.

Para las primeras dos actividades se empezará el Software en línea Drawing ya que es el que se emplea en la clase para realizar los modelos dado que permite la sincronización remota entre usuarios por medio de Drive de Google.

La creación de tablas es la parte medular de este proyecto, por lo que es la actividad a la que se le destinará más tiempo, el código se escribirá en un editor de código como Visual Studio Code para guardar los avances que al final se entregarán, mismo que permite el enlace directo con el repositorio y que permite un fácil manejo de errores y versiones.

Con respecto al punto 5, dado que es un requerimiento opcional solo se planea realizar si todo sale de acuerdo con el plan, donde no nos enfrentemos a errores y contratiempos, además de que existe la posibilidad de que existan otras actividades académicas que ocupen nuestro tiempo. Las actividades realizadas por cualquiera de los integrantes serán mostradas en una bitácora que contendrá este proyecto.

BITACORA PROYECTO BASE DE DATOS			
DIA	INTEGRANTE	ACTIVIDAD	DESCRIPCION
27/06/2021	Profesor	Requerimientos del proyecto	El profesor entregó un documento con los requerimientos del proyecto.
30/06/2021	David	Inicio de MER	Se empezó a identificar entidades, relaciones y atributos.
	David	Realización de MER	Realización de la primera idea del Mer en drawio.
02/07/2021	David	Plante trabajo	Primera redacción del plan de trabajo a realizar.
04/07/2021	David	Introducción	Planteó una introducción preliminar para el proyecto.
	David	Inicio MR	Planteó el MR.
05/07/2021	Daniel	Repository	Creación del repositorio para el proyecto.
06/07/2021	Ambos	Revisión MER	Se revisó el MER y se cambiaron unas cosas.
	Daniel	Revisión MR	Se agregaron los atributos faltantes.
	Ambos	Redacción MER	Redacción del análisis realizado.
07/07/2021	Ambos	MR Completo	Realización del MR completo en drawio.
	Ambos	Tablas	Se crearon unas tablas.
08/07/2021	David	Normalización	Se normalizaron las tablas hasta la 3FN.
	David	Inserción de Productos	Se insertaron productos a la tabla.
09/07/2021	Ambos	Actualización a POSTGRESQL	Como se mostraron errores con POSTGRESQL, tuvimos que actualizar las
10/07/2021	Daniel	Inserción de Cliente	Se insertaron clientes a la tabla.
	Daniel	Inserción de Telefono	Se insertaron teléfonos a la tabla.
	Daniel	Inserción de Proveedor	Se insertaron proveedores a la tabla.
	Daniel	Inserción de Email	Se insertaron emails a la tabla.
12/07/2021	Ambos	Creacion del índice	Se creo el índice a la tabla producto
13/07/2021	Profesor	Comentarios de avance	El profesor dio algunos comentarios sobre el avance del proyecto.
	David	Correcciones	Se realizaron correcciones a los modelos y a las tablas en base a los comentarios del profesor.
	Ambos	Pruebas Servidor	Se instalaron y se realizaron pruebas para el correcto funcionamiento del servidor que se usara para la parte dos del proyecto.

14/07/2021	Daniel	Redacción inicial de la segunda parte	Se redactaron las primeras ideas sobre la segunda parte.
	Daniel	Inserción de Venta	Se insertaron ventas en la tabla.
	David	Prototipo	Realización del prototipo para la parte dos.
15/07/2021	David	Inserciones con php	Se crearon documentos php que permiten insertar en todas las tablas.
18/07/2021	David	Inserciones en Brinda	Se insertaron registros en la tabla. Lo importante es que esos datos no son al azar, ya fueron calculados y procesados en base a los datos de las demás tablas.
19/07/2021	David	Inserciones en Incluye	Se insertaron registros en la tabla. Lo importante es que esos datos no son al azar, ya fueron calculados y procesados en base a los datos de las demás tablas.
20/07/2021	Daniel	Inserción de Producto con AndroidStudio	Se creó el documento en AndroidStudio que permite insertar productos en la BD.
21/07/2021	David	Actualizaciones y eliminaciones con php	Se crearon documentos php que permiten actualizar y eliminar en todas las tablas.
	Daniel	Actualización, eliminación y listado de Productos con	Se crearon documento en AndroidStudio que permiten realizar diferentes tareas con los productos. Ademas se creo un menu para las
	Daniel	APK prueba	Creación de un APK con una versión de la aplicación.
23/07/2021	Daniel	Actualización, eliminación y listado de Proveedores con AndroidStudio	Se crearon documento en AndroidStudio que permiten realizar diferentes tareas con los proveedores. Además se creo un menu para las acciones a realizar.
	David	Consultas con php	Se crearon los docuemntos que permiten hacer las consultas dinamicas sobre las tablas.
24/07/2021	Daniel	Actualización, eliminación y listado de Clientes con AndroidStudio	Se crearon documento en AndroidStudio que permiten realizar diferentes tareas con los clientes. Además se creo un menu para las
28/07/2021	David	Redacción de codigo SQL	Se redactó de manera breve los códigos para crear tablas e insertar los datos, como se pide en el documento.
29/07/2021	David	Redacción del codigo de la implementacion	Se redactó de manera muy breve la explicación de los códigos php empleados para realizar funciones con los registros de las tablas.

31/07/2021	Daniel	Actualización, eliminación y listado de Ventas con AndroidStudio	Se crearon documento en AndroidStudio que permiten realizar diferentes tareas con los ventas. Además se creo un menú para las
2/8/2021	Ambos	Corección de errores	Corrección de errores con php y AndroidStudio.
3/8/2021	Daniel	APK final.	Se creó el apk final de la aplicación
	David	Mapa de navegación	Se creó el mapa de navegación, donde se explica como trabajar con la aplicación.
9/8/2021	Ambos	Documento final	Redacción del documento final.

Diseño.

Después de leer el problema a resolver debemos generar el modelo entidad relación y el modelo relacional para la creación de las tablas.

MER:

Como entidades encontramos PROVEEDOR, PRODUCTOR, CLIENTE y VENTA.

- PROVEEDOR: Tiene el atributo de razón social como clave primaria, los atributos compuestos de nombre y domicilio como se mencionan en las especificaciones posteriores del proyecto; como puede tener varios teléfonos, ese es un atributo multivaluado.
- PRODUCTO: Tiene como clave primaria al código de barras, tiene la fecha de compra, descripción, la marca, el precio de venta y el precio de compra como atributos obligatorios, además decidimos agregar el atributo llamado artículo para mencionar de qué artículo se trata. Stock es el atributo que indica cuántos artículos están disponibles para su venta, este atributo se debe ir cambiando según los proveedores nos lo BRINDAN o se consuma en una VENTA.
- VENTA: Se encuentran los atributos, número de venta que es la clave primaria, la fecha de la venta y el total, que es calculado con los precios de todos los productos que se contienen en esa venta.
- CLIENTE: RFC es la clave primaria, el RFC si es una cadena que contiene caracteres alfanuméricos, como con PROVEEDOR, su nombre y su domicilio son atributos compuestos, y dado que deben tener al menos un correo lo marcamos como multivaluado.

Solo encontramos 3 relaciones entre nuestras entidades:

- BRINDA: Con cardinalidad muchos a muchos, porque un proveedor de muchos productos y un producto es dado por muchos proveedores. Además se encuentra un atributo que permitiría tener un conteo de artículos proveídos, de esta manera podemos conocer exactamente a qué productos se hace referencia y qué proveedor los brindó.
- INCLUYE: Con cardinalidad muchos a muchos, porque en una venta se encuentran muchos productos y los productos pueden aparecer en muchas ventas; tiene un atributo que hace referencia al número de artículos comprados. Así mismo se agregó el atributo totalP que es calculado, (multiplicación del precioV por cantidadProd) y se cambió el atributo de VENTA total a totalV que también es calculado, (suma de todos los totalP).
- REALIZA: Con cardinalidad uno a muchos, porque un cliente puede realizar muchas ventas, pero una venta solo pertenece a un cliente.

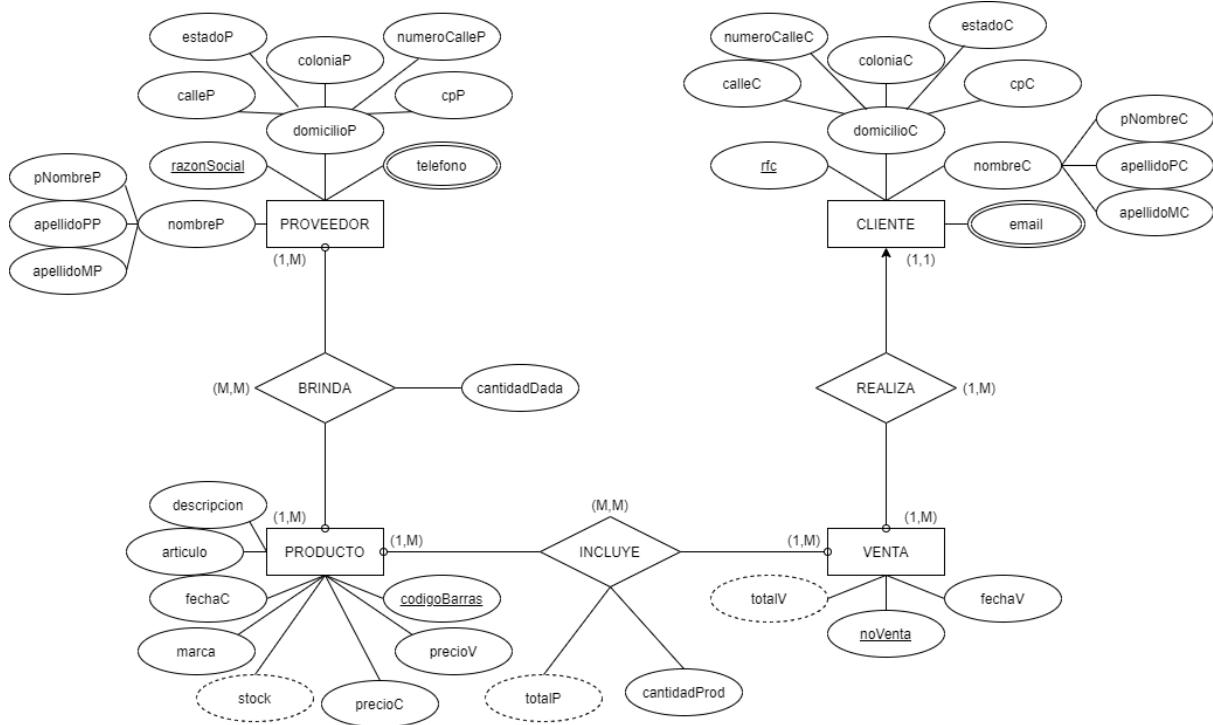


Figura 1 Modelo entidad Relación.

MR.

Del modelo entidad relación pasamos al modelo relacional, donde ya indicaremos los tipos y las restricciones que se marcan en los requerimientos.

Para proveedor no hay ninguna restricción, que se solicite, pero debemos recordar que la razón social no es un entero cualquiera, decidimos que deben ser seis números únicos por proveedor.

```

PROVEDOR{razonSocial int (PK),
          pNombreP varchar(30),
          apellidoPP varchar(40),
          apellidoMP varchar(40),
          calleP varchar(50),
          coloniaP varchar(50),
          cpP varchar(50),
          estadoP varchar(50),
          numeroCalleP int
}
    
```

Teléfono al ser un atributo multivaluado, necesitamos otra tabla, aprovechando que el teléfono es un número único, lo indicaremos como llave primaria y se necesita la razón social del cliente al que pertenece.

```

TELEFONO{tell int (PK),
          razonSocial int (FK)
}
    
```

Para la entidad producto si tenemos restricciones, primero el precio de venta y de compra deben ser mayores a cero y el stock debe ser mayor o igual a cero.

```

PRODUCTO{codigoBarras int (PK),
articulo varchar(50),
fechaC date,
precioC float(CH),
marca varchar(50),
precioV float(CH),
descripcion varchar(100),
stock int (CH)
}

```

Para venta se necesita la llave foránea para indicar a qué cliente pertenece y se necesitará calcular el total de la venta, que es la suma de todos los totales de producto de la relación BRINDA.

```

VENTA{noVenta int (PK),
fechaV date,
totalV float (C),
rfc int (FK)
}

```

Con cliente se tiene más o menos los mismos atributos pero el RFC es una cadena con símbolos alfanuméricos, y demás atributos se llaman igual con excepción de que terminan en una C, de cliente, no como en proveedor que terminan en una P.

```

CLIENTE{rfc int (PK),
pNombreC varchar(30),
apellidoPC varchar(40),
apellidoMC varchar(40),
calleC varchar(50),
coloniaC varchar(50),
cpC varchar(50),
estadoC varchar(50),
numeroCalleC int
}

```

Como el correo es un atributo multivaluado, se necesita otra tabla con la llave foránea que indique a qué cliente pertenece.

```

EMAIL{correo varchar(150) (PK),
rfc int (FK)
}

```

Brinda ya es una relación con llave primaria compuesta, formada por las dos llaves foráneas a las tablas que une, (proveedor y producto).

```

BRINDA{[razonSocial int (FK),codigoBarras int (FK)](PK),
cantidadDada int,
}

```

En incluye la llave primaria también es compuesta, además tiene el total por producto, que se obtiene con la multiplicación del precio de venta del artículo al que se hace referencia por la cantidad dada por el proveedor.

```

INCLUYE{[codigoBarras int (FK),noVenta int (FK)](PK),
cantidadProd int
totalP float (C)
}

```

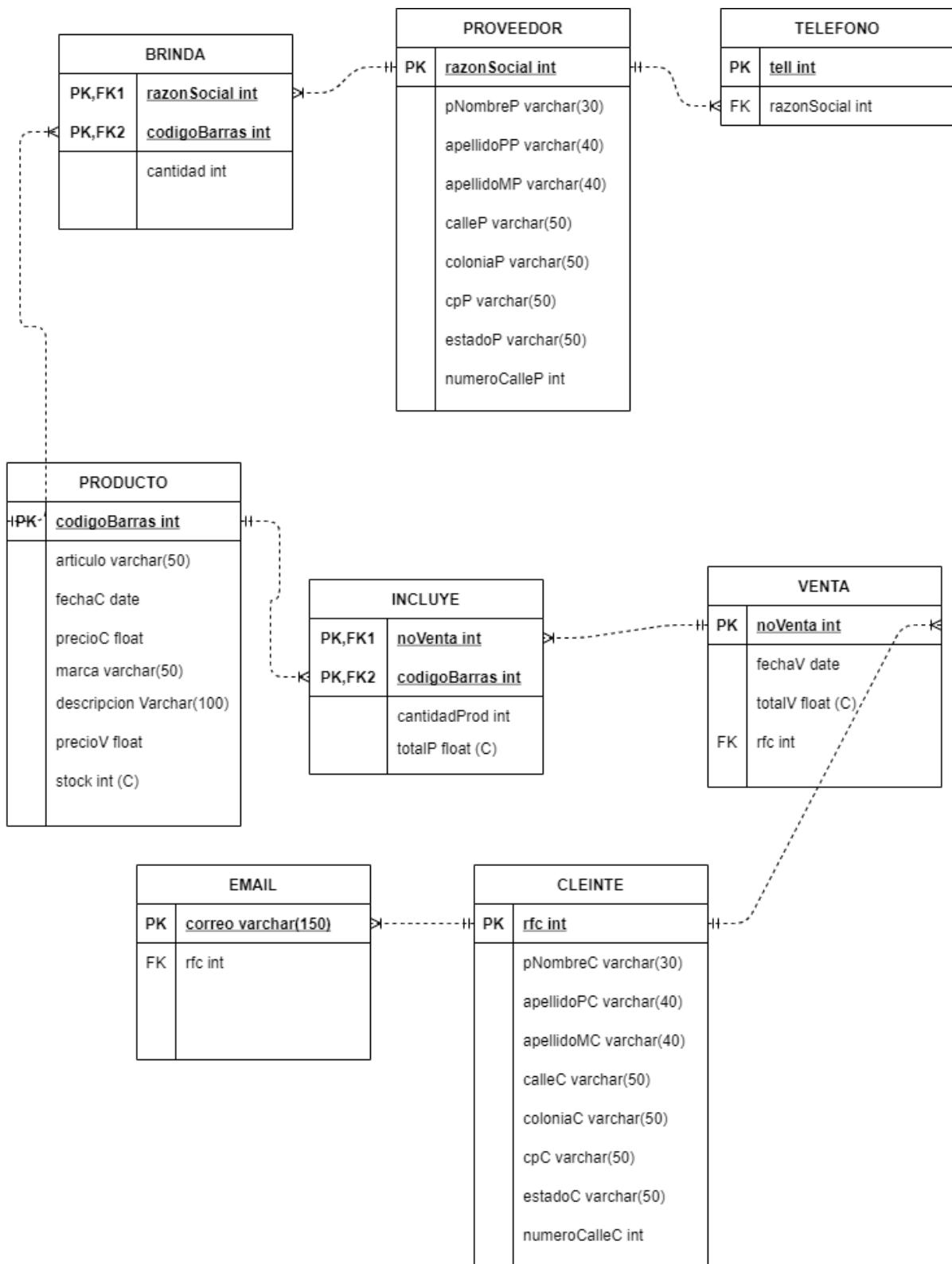


Figura 2 MR.

Normalización:

Producto:

A	B	C	D	E	F	G	H
CodigoBarras	Artículo	FechaC	PrecioC	Marca	Descripción	PrecioV	Stock
55363467	BOLÍGRAFO	08-07-2021	10.90	AZOR	NEGRO	16.50	12
14442988	BOLÍGRAFO	08-07-2021	9.70	AZOR	AZUL	15.0	13
56957154	BOLÍGRAFO	08-07-2021	8.30	AZOR	ROJO	14.90	3

Cumple 1FN porque no tiene grupos de repetición y cada columna contiene valores atómicos. PK=CodigoBarras.

Revisando si cumple 2FN:

$$A \rightarrow [B, C, D, E, F, G, H]$$

Cumple 2FN porque no tiene dependencias funcionales parciales.

Cumple 3FN porque no tiene dependencias transitivas.

Proveedor:

A	B	C	D	E	F	G	H	I
RazonSocial	PNombreP	ApellidoPP	ApellidoMP	CalleP	ColoniaP	CPP	NumeroCalleP	EstadoP
349813	DAVID	REYES	ÁVILA	LOMAS	BAÑOS	06450	121	CDMX
347610	JESUS	SALAZAR	DOMINGUEZ	JAIBA	FRESNO	12345	9	VERACRUZ
238654	RONALDO	RAMIREZ	HERNANDEZ	LOMAZ	MEZCAL	98061	67	EDO. MEX

Cumple 1FN porque no tiene grupos de repetición y cada columna contiene valores atómicos. PK=RazonSocial.

Revisando si cumple 2FN:

$$A \rightarrow [B, C, D, E, F, G, H, I]$$

Cumple 2FN porque no tiene dependencias funcionales parciales.

Cumple 3FN porque no tiene dependencias transitivas.

Teléfono:

A	B
Tell	RazonSocial
5538765427	349813
5534223454	349813
8441419556	347610

Cumple 1FN porque no tiene grupos de repetición y cada columna contiene valores atómicos. PK=Tell. FK=RazonSocial.

Revisando si cumple 2FN:

$$A \rightarrow B$$

Cumple 2FN porque no tiene dependencias funcionales parciales.

Cumple 3FN porque no tiene dependencias transitivas.

Cliente:

A	B	C	D	E	F	G	H	I
RFC	PNombreC	ApellidoPC	ApellidoMC	CalleC	ColoniaC	CPC	NumeroCalleC	EstadoC
BABJ2107077	EDUARDO	FLORES	NULL	NOGAL	PLATA	87037	3	ZACATECAS
SDF343347D	JAIRO	CAZAS	DEL VILLAR	ALMENDROS	MAZON	02144	172	GUERRERO

BH1KI23232	JORGE	ESPARZA	NULL	GALLOS	MAIREN	30098	16	CDMX
------------	-------	---------	------	--------	--------	-------	----	------

Cumple 1FN porque no tiene grupos de repetición y cada columna contiene valores atómicos. PK=RFC.

Revisando si cumple 2FN:

$$A \rightarrow [B, C, D, E, F, G, H, I]$$

Cumple 2FN porque no tiene dependencias funcionales parciales.

Cumple 3FN porque no tiene dependencias transitivas.

Email:

A	B
Correo	RFC
beiprouttunuxi-2829@gmail.com	BABJ2107077B2
diraujoquaubo-3404@hotmail.com	SDF343347D7S7
zavoutraceiyau-6934@yopmail.com	BH1KI23232JN2

Cumple 1FN porque no tiene grupos de repetición y cada columna contiene valores atómicos. PK=Correo. FK=RFC.

Revisando si cumple 2FN:

$$A \rightarrow B$$

Cumple 2FN porque no tiene dependencias funcionales parciales.

Cumple 3FN porque no tiene dependencias transitivas.

Venta:

A	B	C	D
NoVenta	FechaV	TotalV	RFC
VENT-1	21-08-2021	68.00	BABJ2107077B2
VENT-2	04-05-2021	80.00	SDF343347D7S7
VENT-3	26-07-2021	129.00	BABJ2107077B2

Cumple 1FN porque no tiene grupos de repetición y cada columna contiene valores atómicos. PK=Noventa. FK=RFC.

Revisando si cumple 2FN:

$$[A, B] \rightarrow [C, D]$$

Cumple 2FN porque no tiene dependencias funcionales parciales.

Cumple 3FN porque no tiene dependencias transitivas.

Brinda:

A	B	C
RazonSocial	CodigoBarras	CantidadDada
347610	55363467	20
238654	14442988	30
238654	56957154	17

Cumple 1FN porque no tiene grupos de repetición y cada columna contiene valores atómicos. PK=[RazonSocial,CodigoBarras]. F=RazonSocial, CodigoBarras.

Revisando si cumple 2FN:

$$[A, B] \rightarrow [C]$$

Cumple 2FN porque no tiene dependencias funcionales parciales.

Cumple 3FN porque no tiene dependencias transitivas.

Incluye:

A	B	C	D
CodigoBarras	NoVenta	CantidadProd	TotalP
47638900	VENT-1	2	68.00
83013729	VENT-1	4	12.00
52490509	VENT-2	6	20.04

Cumple 1FN porque no tiene grupos de repetición y cada columna contiene valores atómicos. PK=[CodigoBarras, NoVenta]. FK=CodigoBarras, NoVenta.

Revisando si cumple 2FN:

$$[A, B] \rightarrow [C, D]$$

Cumple 3FN porque no tiene dependencias transitivas.

Implementación:

Ahora explicaremos todos los comando SQL que para la creación de tablas y su llenado, hay que aclarar que están en dos documentos (TABLAS.sql y DATOS.sql); pero si desea poner todas las tablas y sus registros en su equipo, recomendamos que copie y pegue el documento Correr.sql, ya que ese tiene todo junto y todo lo que al final usamos.

Creación de tablas:

Para la creación de las tablas ocuparemos :

```
CREATE TABLE nombreTabla(  
    nombreAtributo tipoAtributo [tamañoAtributo] [Check],  
    ...  
    CONSTRAINT nombreConstraint PRIMARY KEY (atributoPK)  
);
```

Primero tenemos la tabla PRODUCTO que tiene todos los atributos que se mencionaron en el análisis previo, la llave primaria que indicamos con CONSTRAINT y ahora si marcamos las restricciones para los atributos de precio de venta y de compra además del stock

```
CREATE TABLE PRODUCTO(  
    CODIGOBARRAS BIGINT,  
    ARTICULO VARCHAR(50),  
    FECHAC DATE,  
    PRECIOC FLOAT CHECK (PRECIOC >0),  
    MARCA VARCHAR(20),  
    DESCRIPCION VARCHAR(100),  
    PRECIOV FLOAT CHECK (PRECIOV >0),  
    STOCK INTEGER CHECK (STOCK >=0),  
    CONSTRAINT PK_PRODUCTO PRIMARY KEY (CODIGOBARRAS)  
);
```

Ahora para PROVEEDOR indicaremos todos sus atributos con el tipo correcto de dato y al final marcamos a la razón social como la llave primaria.

```
CREATE TABLE PROVEEDOR(  
    RAZONSOCIAL INTEGER,  
    PNOMBREP VARCHAR(30),  
    APELLIDOPP VARCHAR(40),  
    APELLIDOMP VARCHAR(40),  
    CALLEP VARCHAR(50),  
    COLONIAP VARCHAR(50),  
    CPP INTEGER,  
    NUMEROCALEP INTEGER,  
    ESTADOP VARCHAR(50),  
    CONSTRAINT PK_PROVEEDOR PRIMARY KEY (RAZONSOCIAL)  
);
```

Para TELÉFONO, debemos indicar la llave primaria TELL y también la razón social del cliente al que pertenece, en este momento aún no se hace la relación para que sea una llave foránea.

```
CREATE TABLE TELEFONO(  
    TELL BIGINT,  
    RAZONSOCIAL INTEGER,  
    CONSTRAINT PK_TELEFONO PRIMARY KEY (TELL)  
);
```

Para el CLIENTE es lo mismo que para el PROVEEDOR, ya que todos sus atributos son los mismos, excepto la llave primaria que ahora es RFC.

```
CREATE TABLE CLIENTE(  
    RFC VARCHAR(13),  
    PNOMBREC VARCHAR(30),  
    APELLIDOPC VARCHAR(40),  
    APELLIDOMC VARCHAR(40),  
    CALLEC VARCHAR(50),  
    COLONIAC VARCHAR(50),  
    CPC INTEGER,  
    NUMEROCALEC INTEGER,  
    ESTADOC VARCHAR(50),  
    CONSTRAINT PK_CLIENTE PRIMARY KEY (RFC)  
);
```

Para el EMAIL es lo mismo que para TELEFONO, ya que también es un atributo multivaluado que pertenece a cada CLIENTE.

```
CREATE TABLE EMAIL(
    CORREO VARCHAR(150),
    RFC VARCHAR(13),
    CONSTRAINT PK_EMAIL PRIMARY KEY (CORREO)
);
```

En VENTA se necesita el atributo RFC para hacerlo llave foránea, ya que en el MER la relación entre VENTA-CLIENTE es de uno a muchos. Adicionalmente se agrega el CHECK al atributo de NOVENTA para que cumpla el requerimiento solicitado.

```
CREATE TABLE VENTA(
    NOVENTA VARCHAR(8),
    FECHAV DATE,
    TOTALV FLOAT CHECK (TOTALV >0),
    RFC VARCHAR(13),
    CONSTRAINT PK_VENTA PRIMARY KEY (NOVENTA)
);

ALTER TABLE VENTA ADD CONSTRAINT CH_NOVENTA CHECK (NOVENTA LIKE 'VENT-%');
```

Para la relación BRINDA (PROVEEDOR-PRODUCTO) se necesitan las dos llaves primarias de cada una de las entidades y con ellas hacer la llave primaria compuesta, junto con el atributo de esta relación.

```
CREATE TABLE BRINDA(
    RAZONSOCIAL INTEGER,
    CODIGOBARRAS INTEGER,
    CANTIDADDADA INTEGER,
    PRIMARY KEY (RAZONSOCIAL,CODIGOBARRAS)
);
```

Para la INCLUYE (relación entre VENTA-PRODUCTO), también se hace la llave primaria compuesta con las llaves primarias de las tablas de las que dependen.

```
CREATE TABLE INCLUYE(
    CODIGOBARRAS INTEGER,
    NOVENTA VARCHAR(8),
    CANTIDADPROD INTEGER,
    TOTALP FLOAT,
    PRIMARY KEY (CODIGOBARRAS,NOVENTA)
);
```

Finalmente agregamos una tabla de apoyo (TTOTALV) para poder realizar la consulta de las ventas por fecha. Nuestra intención era realizar esa consulta con una tabla temporal, pero por problemas al realizarla desde la implementación con php, se decidió dejarla permanente, ya que siempre se usa.

```
CREATE TABLE TTOTALV(
    DIAV DATE PRIMARY KEY,
    TOTALS FLOAT
);
```

Para hacer las llaves foráneas se debe modificar las tablas ya creadas:

```
ALTER TABLE tablaAModificar ADD CONSTRAINT nombreConstraint FOREIGN KEY
(atributoFK) REFERENCES tablaConPK (AtributoPK)
ON DELETE CASCADE ON UPDATE RESTRICT;
```

Para esto hay que tener en cuenta que los tipos y nombre de los atributos deben ser los mismos. Una ventaja de las llaves foráneas es que nos permiten referenciar a un registro en otra tabla, pero para esto ese valor ya debe estar en la otra tabla o no es posible realizarlo. También se indicó que si se borra el registro al que se hace referencia con la llave foránea borre en las demás tablas, esto porque no nos conviene tener valores vacíos que no nos indique nada, sobre todo para evitar problemas con las llaves primarias compuestas, ya que marcarían error.

```

ALTER TABLE TELEFONO ADD CONSTRAINT FK_PROVEEDOR_TELEFONO FOREIGN KEY (RAZONSOCIAL)
REFERENCES PROVEEDOR (RAZONSOCIAL)
ON DELETE CASCADE ON UPDATE RESTRICT;

ALTER TABLE EMAIL ADD CONSTRAINT FK_CLIENTE_EMAIL FOREIGN KEY (RFC)
REFERENCES CLIENTE (RFC)
ON DELETE CASCADE ON UPDATE RESTRICT;

ALTER TABLE VENTA ADD CONSTRAINT FK_CLIENTE_VENTA FOREIGN KEY (RFC)
REFERENCES CLIENTE (RFC)
ON DELETE CASCADE ON UPDATE RESTRICT;

ALTER TABLE BRINDA ADD CONSTRAINT FK_PROVEEDOR_BRINDA FOREIGN KEY (RAZONSOCIAL)
REFERENCES PROVEEDOR (RAZONSOCIAL)
ON DELETE CASCADE ON UPDATE RESTRICT;

ALTER TABLE BRINDA ADD CONSTRAINT FK_PRODUCTO_BRINDA FOREIGN KEY (CODIGOBARRAS)
REFERENCES PRODUCTO (CODIGOBARRAS)
ON DELETE CASCADE ON UPDATE RESTRICT;

ALTER TABLE INCLUYE ADD CONSTRAINT FK_PRODUCTO_INCLUYE FOREIGN KEY (CODIGOBARRAS)
REFERENCES PRODUCTO (CODIGOBARRAS)
ON DELETE CASCADE ON UPDATE RESTRICT;

ALTER TABLE INCLUYE ADD CONSTRAINT FK_VENTA_INCLUYE FOREIGN KEY (NOVENTA)
REFERENCES VENTA (NOVENTA)
ON DELETE CASCADE ON UPDATE RESTRICT;

```

Índice:

Ahora realizaremos la creación de un índice de tipo NON CLUSTERED, este atributo nos ayudará a obtener de manera más rápida el ARTÍCULO en las consultas. Se decidió así porque la PK ya es un índice clustered y así ya nos sirve para referenciar a los registros entre tablas, pero ARTÍCULO, es un atributo que nos servirá mostrar más, ya que es un “objeto del mundo real” no es un dato; por ejemplo, al mostrar los detalles de una venta, se muestra que objeto compraste, no su código de barras, porque es más fácil entenderlo así.

```

CREATE INDEX INDEX_PRODUCTO ON
PRODUCTO (ARTICULO);

```

Inserción de datos:

Para saber que todo funciona bien, y al momento de realizar la parte de la presentación, se muestran registros para poder trabajar con ellos, se deben insertar datos en las tablas. A continuación explicaremos cómo insertamos los datos en las tablas:

```
INSERT INTO tabla VALUES (atributo,...);
```

Para insertar en PRODUCTO solo llenamos los atributos en orden con respecto a cómo definimos la tabla, esto porque era lo más fácil para hacerlo varias veces. Además al atributo TOCK le damos un valor de cero, porque hasta después se actualizará su valor.

```

INSERT INTO PRODUCTO VALUES (55363467,'BOLIGRAFO','08-07-2021',10.9,'AZOR','NEGRO',16.5,0);
INSERT INTO PRODUCTO VALUES (14442988,'BOLIGRAFO','08-07-2021',10.9,'AZOR','ROJO',16.5,0);
INSERT INTO PRODUCTO VALUES (56957154,'BOLIGRAFO','08-07-2021',10.9,'AZOR','AZUL',16.5,0);
INSERT INTO PRODUCTO VALUES (60554372,'MARCADOR DE TEXTOS','08-07-2021',19.9,'AZOR','AMARILLO',21.9,0);
INSERT INTO PRODUCTO VALUES (19027769,'MARCADOR DE TEXTOS','08-07-2021',19.9,'AZOR','ROSA',21.9,0);
INSERT INTO PRODUCTO VALUES (93869183,'MARCADOR DE TEXTOS','08-07-2021',19.9,'AZOR','AZUL',21.9,0);

```

Después insertamos en PROVEEDOR, también ocupamos la misma manera de insertar que para los registros anteriores, en este caso se ve que un atributo lo dejamos como nulo, ya que ese atributo es opcional.

```

INSERT INTO PROVEEDOR VALUES (349813,'DAVID','REYES','AVILA','LOMAS','BAÑOS',70239,21,'CDMX');
INSERT INTO PROVEEDOR VALUES (499821,'SMITTY','WERBEN','MANJENSEN','','CEMENTERIO',70249,NULL,'FONDO DE BIKINI');
INSERT INTO PROVEEDOR VALUES (347610,'PETER','PARKER','','','FOREST HILLS',89239,3,'NEW YORK');
INSERT INTO PROVEEDOR VALUES (238654,'JESUS','SALAZAR','DOMINGUEZ','JAIBA','SINESQUINAS',70539,NULL,'VERACRUZ');
INSERT INTO PROVEEDOR VALUES (129476,'ROLANDO','ROSAS','HERNANDEZ','LOMAS','JODIDO',70239,1,'CDMX');
INSERT INTO PROVEEDOR VALUES (575302,'MIGUEL','BAZAN','SILVA','TULUM','FRESAS',70299,NULL,'QUINTANA ROO');

```

Ahora para BRINDA ya es algo más complicado, primero se inserta una tupla en BRINDA, con una RAZÓN SOCIAL de un proveedor creado con anterioridad y de un producto ya existente y el número de piezas que da para ese PRODUCTO. Después se va a modificar el PRODUCTO con el CODIGOBARRAS que se dio anteriormente, se va a modificar el

STOCK, con una suma, entre lo que ya está y lo que se dio en el registro de brinda de ese PRODUCTO.

Debemos tomar en cuenta que como la PK compuesta depende del PRODUCTO y el PROVEEDOR, estos no se pueden repetir en la misma combinación, (mismo producto con mismo proveedor), porque de esta manera se marcaría un error.

```
INSERT INTO BRINDA VALUES (349813,55363467,12);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 55363467) + (SELECT CANTIDADADADA FROM BRINDA WHERE CODIGOBARRAS = 55363467) WHERE CODIGOBARRAS = 55363467;
INSERT INTO BRINDA VALUES (349813,14442988,13);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 14442988) + (SELECT CANTIDADADADA FROM BRINDA WHERE CODIGOBARRAS = 14442988) WHERE CODIGOBARRAS = 14442988;
INSERT INTO BRINDA VALUES (349813,56957154,17);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 56957154) + (SELECT CANTIDADADADA FROM BRINDA WHERE CODIGOBARRAS = 56957154) WHERE CODIGOBARRAS = 56957154;
INSERT INTO BRINDA VALUES (349813,60554372,2);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 60554372) + (SELECT CANTIDADADADA FROM BRINDA WHERE CODIGOBARRAS = 60554372) WHERE CODIGOBARRAS = 60554372;
INSERT INTO BRINDA VALUES (349813,19027769,4);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 19027769) + (SELECT CANTIDADADADA FROM BRINDA WHERE CODIGOBARRAS = 19027769) WHERE CODIGOBARRAS = 19027769;
INSERT INTO BRINDA VALUES (349813,93869183,11);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 93869183) + (SELECT CANTIDADADADA FROM BRINDA WHERE CODIGOBARRAS = 93869183) WHERE CODIGOBARRAS = 93869183;
```

Después insertamos en TELEFONO, para esto también debemos ocupar la llave primaria de un PROVEEDOR ya creado.

```
INSERT INTO TELEFONO VALUES (5538765427,349813);
INSERT INTO TELEFONO VALUES (5534223454,349813);
INSERT INTO TELEFONO VALUES (8441419556,499821);
INSERT INTO TELEFONO VALUES (8959843624,499821);
INSERT INTO TELEFONO VALUES (6840947404,499821);
INSERT INTO TELEFONO VALUES (7925481394,347610);
INSERT INTO TELEFONO VALUES (6695614317,347610);
INSERT INTO TELEFONO VALUES (8697292844,347610);
```

Para insertar en CLIENTE ocupamos la misma manera que en las demás tablas, en orden e indicando todos los atributos.

```
INSERT INTO CLIENTE VALUES('BABJ2107077B2','JOANNY','BARRON','BALMASEDA','DELINCUENCIA','LAS AGUILAS',70631,NULL,'CDMX');
INSERT INTO CLIENTE VALUES('SDF343347D757','EMILIANO','BEJAR','MOLINEVO','LOMAS','BAÑOS',70239,21,'CDMX');
INSERT INTO CLIENTE VALUES('BH1K123232JN2','KAY','RICOY','AUJEA','PIZZA','PESTO',07784,2,'BERNOULLI');
INSERT INTO CLIENTE VALUES('BK01234H12333','GEORGINA','MECA','VILLABEL','VENUSTIANO','CARRANZA',12246,8,'PUEBLA');
INSERT INTO CLIENTE VALUES('K4HIUI235V54C','CARIDAD','ALDAZABAL','GALDIA','VILLAS','MAIREN',17668,2,'DURANGO');
INSERT INTO CLIENTE VALUES('JVY27266BHYU2','LUANA','SIERRO','BORREGO','CAMINOS','LOMAS',10047,13,'PUEBLA');
INSERT INTO CLIENTE VALUES('BHKJ272625N26','LEGOLAS','RUSOTO','MANCOBO','TORRES','PANCARDO',30841,NULL,'VERACRUZ');
INSERT INTO CLIENTE VALUES('OPO13247373OH','JERRY','ALFONSO','LANTARON','SILLAS','PANCHO VILLA',51819,45,'GUERRERO');
INSERT INTO CLIENTE VALUES('SHFOW7F886FG8','XOCHIQUETZAE','VIVAS','ERASO','SOLISTAS','UNAM',13505,2,'CHIAPAS');
INSERT INTO CLIENTE VALUES('PODBG3H3636H','DANIEL','SALINAS','ROMERO','ALMENDROS','EL GUAYACAN',23731,NULL,'NUEVO LEON');
INSERT INTO CLIENTE VALUES('ISDJ348374NJI','RAUL','PUENTE','MANCILLA','CUCHAREAR','PLATICAS',41974,13,'MEXICO');
INSERT INTO CLIENTE VALUES('SNC837492NI23','SERGIO','NOBLE','CAMARGO','RISK','ANTI-CHÉ',24565,NULL,'COLIMA');
```

Para EMAIL tenemos que poner el RFC de un cliente existente, y cada correo debe ser diferente, ya que es la PK.

```
INSERT INTO EMAIL VALUES('beiproutunnuxi-2829@gmail.com','BABJ2107077B2');
INSERT INTO EMAIL VALUES('diraujoquauvo-3404@hotmail.com','BABJ2107077B2');
INSERT INTO EMAIL VALUES('zavoutraceiyau-6934@yopmail.com','SDF343347D757');
INSERT INTO EMAIL VALUES('seifemmeppeixau-9726@gmail.com','BH1K123232JN2');
INSERT INTO EMAIL VALUES('praquauvigeimi-8321@yopmail.com','BK01234H12333');
INSERT INTO EMAIL VALUES('prufruturaura-2824@hotmail.com','K4HIUI235V54C');
INSERT INTO EMAIL VALUES('hisiffautanne-2975@gmail.com','JVY27266BHYU2');
INSERT INTO EMAIL VALUES('gapimmijoutro-4723@hotmail.com','BHKJ272625N26');
INSERT INTO EMAIL VALUES('grayozaqueici-4818@gmail.com','OPO13247373OH');
INSERT INTO EMAIL VALUES('yeufokeppihi-7714@yopmail.com','OPO13247373OH');
```

A continuación creamos una secuencia para evitar problemas al crear la llave primaria de VENTA, como el identificador de la VENTA, debe llevar un número es conveniente crearla, porque no queremos que ese número se repita, queremos que sean en secuencia. Hay que aclarar que la secuencia no cumple toda la condición del check en NOVENTA, solo nos apoya para el número.

```
CREATE SEQUENCE SQ_VENTA
START WITH 1
INCREMENT BY 1;
```

Finalmente insertamos en VENTA e INCLUYE:

Primero se inserta en venta donde el valor de NOVENTA será igual a la concatenación de la palabra 'VENT-' y el siguiente valor de la secuencia, después indicamos la fecha, el TOTALV se deja vacío porque aún no se calcula y se indica a qué cliente pertenece esa venta.

Después para cada artículo que se compre en esa venta se inserta una tupla en INCLUYE donde se indicará el CODIGOBARRAS del PRODUCTO, el NOVENTA, empleando la concatenación como en VENTA; el número de piezas consumidas y para calcular el total por esas piezas del producto se realiza una multiplicación entre CANTIDADPROD y el PRECIOV de ese PRODUCTO.

A continuación se resta el número de piezas compradas al STOCK del producto.

Y una vez que se pusieron todos los productos incluidos en esa VENTA, se calcula el total

```
INSERT INTO VENTA VALUES(concat('VENT-',nextval('SQ_VENTA')),'01-06-2021','BILL','BA032107077B2');
INSERT INTO INCLUYE VALUES (12386600,concat('VENT-',(SELECT last_value from SQ_VENTA)),,(SELECT last_value FROM PRODUCTO WHERE CODIGOBARRAS = 12386600)*1);
UPDATE PRODUCTO SET STOCK = (SELECT CANTIDADPROD FROM PRODUCTO WHERE CODIGOBARRAS = 12386600) - (SELECT CANTIDADPROD FROM INCLUYE WHERE NOVENTA = concat('VENT-',(SELECT last_value from SQ_VENTA)) AND CODIGOBARRAS = 12386600);
DELETE FROM INCLUYE WHERE NOVENTA = concat('VENT-',(SELECT last_value from SQ_VENTA));
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 12386600) - (SELECT CANTIDADPROD FROM INCLUYE WHERE NOVENTA = concat('VENT-',(SELECT last_value from SQ_VENTA)) AND CODIGOBARRAS = 12386600);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 67493765,concat('VENT-',(SELECT last_value from SQ_VENTA)));
INSERT INTO INCLUYE VALUES (17495317,concat('VENT-',(SELECT last_value from SQ_VENTA)),,(SELECT PRECIOV FROM PRODUCTO WHERE CODIGOBARRAS = 67493765)*1);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 17495317) - (SELECT CANTIDADPROD FROM INCLUYE WHERE NOVENTA = concat('VENT-',(SELECT last_value from SQ_VENTA)) AND CODIGOBARRAS = 67493765);
UPDATE PRODUCTO SET STOCK = (SELECT STOCK FROM PRODUCTO WHERE CODIGOBARRAS = 17495317) - (SELECT CANTIDADPROD FROM INCLUYE WHERE NOVENTA = concat('VENT-',(SELECT last_value from SQ_VENTA)) AND CODIGOBARRAS = 17495317);
UPDATE VENTA SET TOTALV = (SELECT SUM(TOTALP) FROM INCLUYE WHERE NOVENTA = concat('VENT-',(SELECT last_value from SQ_VENTA)));
UPDATE VENTA SET TOTALV = (SELECT SUM(TOTALP) FROM INCLUYE WHERE NOVENTA = concat('VENT-',(SELECT last_value from SQ_VENTA))) WHERE NOVENTA = concat('VENT-',(SELECT last_value from SQ_VENTA));
```

de todos los TOTALP donde la NOVENTA sea igual a la VENTA actual.

Código en php:

Para la explicación de la implementación con documentos PHP para realizar las tareas solicitadas. Como ya se mencionó la idea del funcionamiento del código para esta parte se obtuvo con una serie de videos de YouTube, donde solo hizo falta adecuarlos a nuestras necesidades, para que trabaje con la base de datos que nos brindó el servidor y las tablas con las que trabajaremos. Por fortuna la mayoría de los programas realizan las mismas funciones, donde solo cambian los parámetros recibidos, lo que nos permitió simplificar el trabajo al reutilizar código.

El primer documento es *db_config.php* donde se definen como constantes una serie de valores que nos permitirán la conexión con la base de datos que nos brinda el servidor que seleccionamos. Los valores son el nombre de usuario, su contraseña, el nombre de la base de datos y el servidor.

```
<?php
define('DB_USER', "ffeqfaddlyivdl"); // Usuario
define('DB_PASSWORD', "d7902e67df5d7b2acebd4332b5b649da994e35cf4620d22f6c1026de7161b42c"); // Password
define('DB_DATABASE', "dek18n144tf611"); // Nombre de la base de datos
define('DB_SERVER', "ec2-52-23-40-80.compute-1.amazonaws.com"); // host server
?>
```

Después *Database.php* donde se ocupa el documento anterior. En resumen este documento se encarga de crear la conexión entre la aplicación y la base de datos de PostgreSQL.

```

<?php

error_reporting(E_ALL ^ E_DEPRECATED);

require_once 'db_config.php';

class Database{
    private static $db = null;
    private static $pdo;

    final private function __construct(){
        try {
            self::getDb();
        } catch (PDOException $e) {
        }
    }

    public static function getInstance(){
        if (self::$db === null) {
            self::$db = new self();
        }
        return self::$db;
    }

    public function getDb(){
        if (self::$pdo == null) {
            self::$pdo = new PDO(
                'pgsql:dbname=' . DB_DATABASE .';host=' . DB_SERVER .';',
                DB_USER,
                DB_PASSWORD
            );

            self::$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }
        return self::$pdo;
    }

    final protected function __clone()
    {

    }

    function _destructor()
    {
        self::$pdo = null;
    }
}

?>

```

Como ya hay un documento que crea la conexión, ya podemos empezar a trabajar con la tabla, pero un antes debemos hablar sobre el documento *SQLGlobal.php*, donde se crean unas funciones que invocamos en los siguientes documentos; estas funciones nos permiten hacer consultas, insertar, actualizar y eliminar registros de las tablas, ya sea recibiendo parámetros para hacerlo con un filtro o que siempre realice la misma operación.

```

<?php
    require 'Database.php';

    class SQLGlobal{
        function __construct(){
        }

        //Consulta para una lista de Registro
        public static function selectArray($consulta){
            $resultado = Database::getInstance()->getDb()->prepare($consulta);
            $resultado->execute();
            return $resultado->fetchAll(PDO::FETCH_ASSOC);
        }

        //Consulta para una lista de Registro con filtros
        public static function selectArrayFiltro($consulta,$datos){
            $resultado = Database::getInstance()->getDb()->prepare($consulta);
            $resultado->execute($datos); //array de datos
            return $resultado->fetchAll(PDO::FETCH_ASSOC);
        }

        //Consultar solo un registro
        public static function selectObject($consulta){
            $resultado = Database::getInstance()->getDb()->prepare($consulta);
            $resultado->execute();
            return $resultado->fetch(PDO::FETCH_ASSOC); //fetch devuelve solo un registro
        }

        //Consultar solo un registro con filtros
        public static function selectObjectFiltro($consulta,$datos){
            $resultado = Database::getInstance()->getDb()->prepare($consulta);
            $resultado->execute($datos); //array de datos
            return $resultado->fetch(PDO::FETCH_ASSOC); //fetch devuelve solo un registro
        }

        //Create Update Delete
        public static function cud($consulta){
            $sentencia = Database::getInstance()->getDb()->prepare($consulta);
            if($sentencia->execute($datos)){
                return $sentencia->rowCount(); //devuelve el numero de filas afectadas
            }else{
                return 0; //devuelve el numero de filas afectadas
            }
        }

        //Create Update Delete con Filtro
        public static function cudFiltro($consulta,$datos){
            $sentencia = Database::getInstance()->getDb()->prepare($consulta);
            if($sentencia->execute($datos)){
                return $sentencia->rowCount(); //devuelve el numero de filas afectadas
            }else{
                return 0; //devuelve el numero de filas afectadas
            }
        }
    }

    ?>

```

Con los documentos previos ya podemos empezar a realizar operaciones sobre las tablas, como ya se mencionó, la mayoría de los documentos realizan las mismas funciones, pero sobre diferentes tablas, así que solo explicaremos los documentos para la tabla PRODUCTO.

Primero *Producto_GETALL.php*, este documento nos permite mostrar todos los atributos de todos productos disponibles, este documento no recibe parámetros, ya que no se necesita realizar un filtro en la consulta de SQL, por eso se ocupa la función *selectArray*, (que permite obtener un grupo de registros) que se creó en el documento anterior. La salida de programa, en caso de que no exista ningún error, crea una cadena en código Json, y el mensaje de que se obtuvieron correctamente los datos, de caso contrario, nos mostrará un mensaje con el detalle del error.

```

<?php
//Consulta a todos los Productos disponibles
require 'SQLGlobal.php';

if($_SERVER['REQUEST_METHOD']=='GET'){
    try{
        $respuesta = SQLGlobal::selectArray("SELECT * FROM PRODUCTO WHERE STOCK > 0");//Consulta a PRODUCTO sin filtro
        echo json_encode(array(
            'respuesta'=>'200',
            'estado' => 'Se obtuvieron los datos correctamente',
            'data'=>$respuesta,
            'error'=>''
        ));
    }catch(PDOException $e){
        echo json_encode(
            array(
                'respuesta'=>'-1',
                'estado' => 'Ocurrio un error, intentelo mas tarde',
                'data'=>'',
                'error'=>$e->getMessage()
            )
        );
    }
}
?>

```

El documento *Producto_FiltroGETArticulo.php*, permite realizar la consulta por el atributo ARTÍCULO de los productos disponibles. Este documento si debe recibir un parámetro dado por el usuario, para eso ocupamos el método GET, para este metodo se reciben los parámetros por el URL. Se ocupa la función, *selectArrayFiltro*, que se ocupa para obtener varios registros cuando se recibe un parámetro. Para esta consulta SQL se ocupa de un comodín, esto permite mostrar los registros en caso de que se escriba parcialmente el artículo a buscar; en el caso específico de que no se escriba nada, se muestran todos los productos.

Para las demás tablas se crearon documentos que realicen las consultas con filtros según nos parecieron pertinentes, lo único que cambia es la consulta que se realiza y los parámetros que se deben recibir por parte del usuario.

```

<?php
//Consulta dinamica por articulo de la tabla Producto
//sirve, pero si se deja sin parametro se trae todo
require 'SQLGlobal.php';

if($_SERVER['REQUEST_METHOD']=='GET'){
    try{
        $ARTICULO = $_GET["ARTICULO"]; // obtener parametros GET

        $respuesta = SQLGlobal::selectArrayFiltro(
            "SELECT * FROM PRODUCTO WHERE LOWER(ARTICULO) LIKE LOWER(?) AND STOCK > 0",
            array($ARTICULO.'%')
        );//con filtro ("El tamaño del array debe ser igual a la cantidad de los '?'")
        echo json_encode(array(
            'respuesta'=>'200',
            'estado' => 'Se obtuvieron los datos correctamente',
            'data'=>$respuesta,
            'error'=>''
        ));
    }catch(PDOException $e){
        echo json_encode(
            array(
                'respuesta'=>'-1',
                'estado' => 'Ocurrio un error, intentelo mas tarde',
                'data'=>'',
                'error'=>$e->getMessage()
            )
        );
    }
}
?>

```

Para el documento que nos permite insertar nuevos productos (*Producto_Insertar.php*) ocupamos la función *cudFiltro* del documento *SQLGlobal.php* y el método POST porque es más seguro, esto se refiere a que no se reciben por el URL, lo que permite que el usuario inserta registros en cualquier tabla sin permiso; para POST se deben recibir por un objeto Json que se crea y recibe todos los parámetros necesarios.

Se creó un documento que permite insertar en cada una de las tablas, para eso debemos tener presentes los parámetros necesarios y sus condiciones (Checks). Para tablas que necesitan FK se necesita insertar una ya existente, o no se podrá insertar.

Pero también hay documentos que en uno solo permiten registrar en varias tablas, esto porque unos datos se necesitan en varias tablas, las llaves foráneas, y para no solicitarlas al usuario varias veces, se prefiere preguntarlos en un solo documento como son *Venta_Insertar-Incluye.php* y *Producto_Insertar-Brinda.php*.

Para el caso particular del cálculo del total de una venta, no es posible insertar ese atributo, ese lo calcula PHP, porque si no el usuario podría cualquier valor ahí, y eso no puede ser.

```
<?php
//Añadir un nuevo Producto
//Solo lo hacemos con POST porque es más seguro
require 'SQLGlobal.php';

if($_SERVER['REQUEST_METHOD']=='POST'){
    try{
        $datos = json_decode(file_get_contents("php://input"),true);

        $CODIGOBARRAS = $datos["CODIGOBARRAS"]; //NO SE OCUPA PORQUE LA SECUENCIA LO HACE
        $ARTICULO = $datos["ARTICULO"];// obtener parametros POST
        $FECHAC = $datos["FECHAC"];
        $PRECIOC = $datos["PRECIOC"];
        $MARCA = $datos["MARCA"];
        $DESCRIPCION = $datos["DESCRIPCION"];
        $PRECIOV = $datos["PRECIOV"];
        $STOCK = $datos["STOCK"];
        $respuesta = SQLGlobal::cudFiltro(
            "INSERT INTO PRODUCTO VALUES (?, ?, ?, ?, ?, ?, ?, ?)",
            array($CODIGOBARRAS,$ARTICULO,$FECHAC,$PRECIOC,$MARCA,$DESCRIPCION,$PRECIOV,$STOCK)
        );//con filtro ("El tamaño del array debe ser igual a la cantidad de los '?')
        if($respuesta > 0){
            echo json_encode(array(
                'respuesta'=>'200',
                'estado' => 'Se inserto correctamente el producto',
                'data'=>'El numero de registros afectados es: '.$respuesta,
                'error'=> ''
            ));
        }else{
            echo json_encode(array(
                'respuesta'=>'100',
                'estado' => 'No se inserto correctamente le producto',
                'data'=>$respuesta,
                'error'=> ''
            ));
        }
    }catch(PDOException $e){
        echo json_encode(
            array(
                'respuesta'=>'-1',
                'estado' => 'ocurrio un error, intentelo mas tarde',
                'data'=>'',
                'error'=>$e->getMessage()
            )
        );
    }
}
?>
```

Para actualizar registros en la tabla PRODUCTO (*Producto_Actualizar.php*) también se ocupa el método POST y la función *cudFiltro*, en este caso se solicitan de nuevo todos los valores del registro a modificar, pero no es posible modificar su llave primaria. Hay que tener en consideración, que ese registro debió haberse creado anteriormente y se deben cumplir las mismas condiciones que cuando se inserta un registro.

Para el caso en concreto del PRODUCTO, desde este documento no es posible modificar el STOCK, tiene su documento propio que modifica solo ese atributo, esto porque sé ese es un atributo que depende de una tabla de relación, BRINDA, y crea problemas en las reglas de negocio, por ejemplo; esas piezas de ese producto las brindó un proveedor, no aparecieron de la nada.

```

php
Actualizacion de Productos (todos los atributos), todo funciona bien
require 'SQLGlobal.php';

if($_SERVER['REQUEST_METHOD']=='POST'){
    try{
        $datos = json_decode(file_get_contents("php://input"),true);

        $CODIGOBRARRAS = $datos["CODIGOBRARRAS"]; //Lo necesitamos para saber cual modificar AGIOH HU HU
        $ARTICULO = $datos["ARTICULO"];// obtener parametros POST
        $FECHAC = $datos["FECHAC"];
        $PRECIOC = $datos["PRECIOC"];
        $MARCA = $datos["MARCA"];
        $DESCRIPCION = $datos["DESCRIPCION"];
        $PRECIOV = $datos["PRECIOV"];
        // $STOCK = $datos["STOCK"];// aqui no se cambia esto
        $respuesta = SQLGlobal::cudFiltro(
            "UPDATE PRODUCTO SET ARTICULO = ?, FECHAC = ?, PRECIOC = ?, MARCA = ?, DESCRIPCION = ?, PRECIOV = ? WHERE CODIGOBRARRAS = ?",
            array($ARTICULO,$FECHAC, $PRECIOC, $MARCA,$DESCRIPCION,$PRECIOV, $CODIGOBRARRAS)
        );//con filtro ("El tamaño del array debe ser igual a la cantidad de los '?'")
        if($respuesta > 0){
            echo json_encode(array(
                'respuesta'=>'200',
                'estado' => 'Se actualizo correctamente',
                'data'=>'Número de filas afectadas: '.$respuesta,
                'error'=> ''
            ));
        }else{
            echo json_encode(array(
                'respuesta'=>'100',
                'estado' => 'El codigo de producto no existe',
                'data'=>'Número de filas afectadas: '.$respuesta,
                'error'=> ''
            ));
        }
    }catch(PDOException $e){
        echo json_encode(
            array(
                'respuesta'=>'-1',
                'estado' => 'Ocurrio un error, intentelo mas tarde',
                'data'=>'',
                'error'=>$e->getMessage()
            )
        );
    }
}
}

```

Para eliminar Productos se creó el documento *Producto_Eliminar.php*, también se usa POST y la función *cudFiltro* del documento *SQLGlobal.php*, como parámetro solo se necesita la llave primaria del registro. Para eliminar los registros, fue necesario indicar que se borren los registros en cascada, ya que de lo contrario solo se podría borrar registros de tablas que aun no se relacionen con otras a través de llaves foráneas, por ejemplo; un CLIENTE que no haya realizado ninguna VENTA.

```

<?php
//Eliminar Productos
require 'SQLGlobal.php';

if($_SERVER['REQUEST_METHOD']=='POST'){
    try{
        $datos = json_decode(file_get_contents("php://input"),true);

        $CODIGOBRARRAS = $datos["CODIGOBRARRAS"]; // obtener parametros POST, Solo se necesita el id
        $respuesta = SQLGlobal::cudFiltro(
            "DELETE FROM PRODUCTO WHERE CODIGOBRARRAS = ?",
            array($CODIGOBRARRAS)
        );//con filtro ("El tamaño del array debe ser igual a la cantidad de los '?'")
        if($respuesta > 0){
            echo json_encode(array(
                'respuesta'=>'200',
                'estado' => 'Se elimino correctamente el producto',
                'data'=>'El numero de filas afectadas es: '.$respuesta,
                'error'=> ''
            ));
        }else{
            echo json_encode(array(
                'respuesta'=>'100',
                'estado' => 'Ese producto no existe',
                'data'=>'El numero de filas afectadas es: '.$respuesta,
                'error'=> ''
            ));
        }
    }catch(PDOException $e){
        echo json_encode(
            array(
                'respuesta'=>'-1',
                'estado' => 'Ocurrio un error, intentelo mas tarde',
                'data'=>'',
                'error'=>$e->getMessage()
            )
        );
    }
}
?>

```

Como resumen podemos decir que se realizan las operaciones de consulta, insertar, actualizar y eliminar registros de cada una de las tablas. Pero debido a que se debe respetar las condiciones de integridad de las tablas, no se debe dar acceso al usuario final a modificar cualquier cosa; además se deben realizar operaciones que faciliten la obtención de datos y así se verifique la veracidad de todos los valores, hay documentos que realizan varias operaciones en uno solo, como *Venta_Insertar-Incluye.php* que primero crea la venta con su total vacío, luego crea un registro en la tabla INCLUYE, ocupando valores recibidos para la creación de la venta junto al primer producto que se compra, después en el producto se disminuye el STOCK, se resta el número de piezas compradas de ese PRODUCTO; y finalmente se actualiza la venta creada, esto para dar el total de la venta del nuevo producto.

Mapa de navegación.

Este sección fue elaborada con el fin de que el usuario final pueda entender el mecanismo de uso de la aplicación para dispositivos móviles.

- El *apk* para su instalación se encuentra en esta carpeta.

Esperamos le sea de utilidad.

Consideraciones generales:

- Pon atención a lo que se te pide llenar, si contestas algo que no corresponde no se podrá realizar la actividad.
- Es necesario que recuerdes valores de Productos, Proveedores, Clientes y ventas, ya que son necesarias para poder interactuar con ellas, además es necesario que tu las escribas.
- Las fechas están en el formato (MM-DD_AAAA).
- Si tienes dudas, presiona el botón de la actividad que quieres realizar y se mostrará un mensaje de que debes llenar.

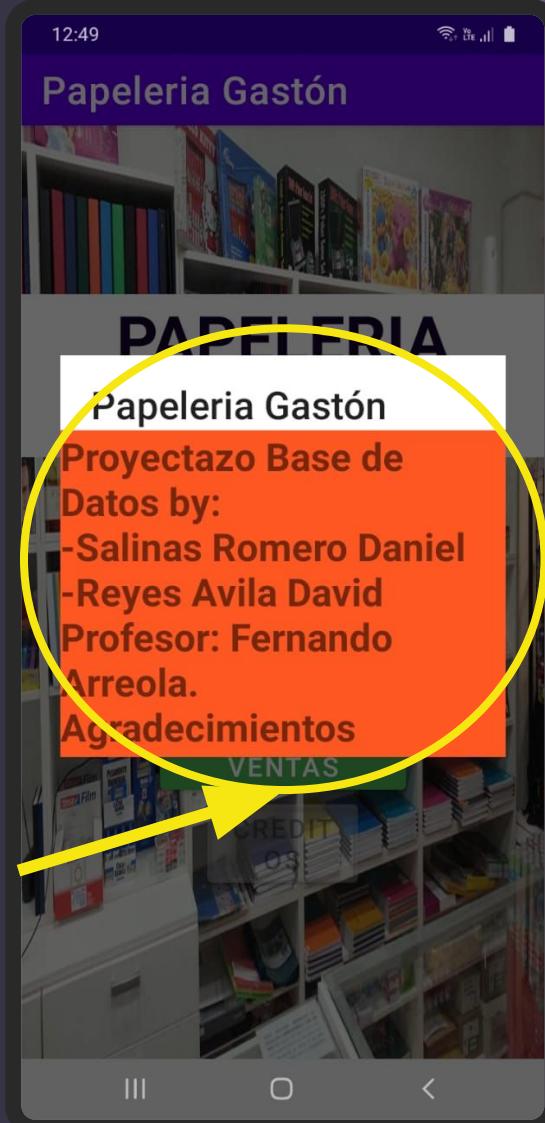
¡¡Bienvenido a la aplicación de **PAPELERÍA GASTÓN!!**

Cuando usted presione el icono desde su celular usted ingresara a la aplicación de la papelería y esto será lo primero que verá:

Aquí se encuentran 5 botones, 4 lo dirigirán a las tareas que se pueden hacer.



El restante, al presionarlo, le mostrará el mensaje con los créditos.



Operaciones con los proveedores

Primero revisaremos las Operaciones con los Proveedores, que son las personas que brindan los productos a la papelería:

Aquí se encuentran las opciones para:

- Insertar o Actualizar Proveedor.
- Borrar proveedor.
- Operaciones con los teléfonos.
- Listar los proveedores.

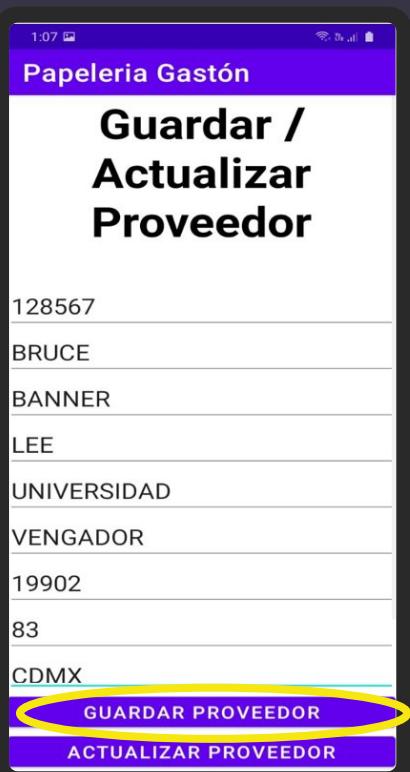


Guardar nuevo Proveedor y Actualizar uno existente.

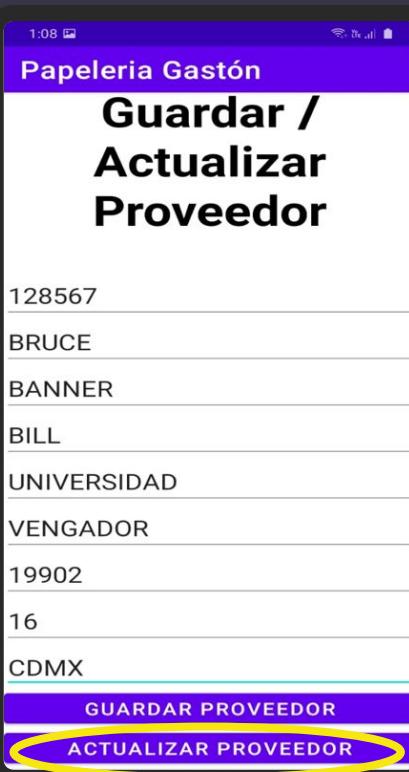
Para crearlo debemos conocer todos los valores de lo que se nos pregunte y debemos recordar que se deben llenar todos los campos, de caso contrario no podremos continuar.



Cuando hayamos llenado todo, podemos presionar el botón de guardar. Despues de presionar el botón te aparecerá un mensaje si el registro fue exitoso.



En caso de que te hayas equivocado, puedes cambiar los valores en esta misma pantalla. Pero ten en cuenta de que si quieres actualizarlo mucho despues, tendrás que escribir todos los valores aunque no los quieras cambiar.



Te sugerimos que tengas en cuenta que:

- La Razón social es un valor que no se puede repetir, ponle uno distinta a cada proveedor.
- Si quieres actualizar uno, debió ser creado con anterioridad, sino créalo.
- Asegúrate de escribir correctamente la razón social, de lo contrario no actualizamos nada.
- No olvides llenar todos los campos con lo que se te pide.

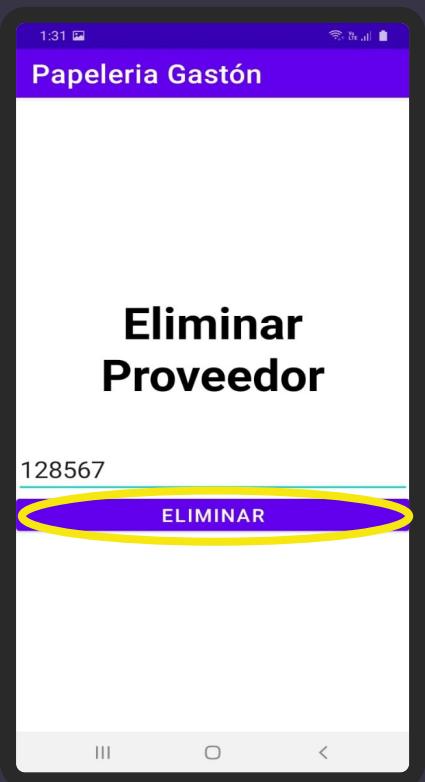


Eliminar Proveedores y listarlos.

Para poder eliminar a un proveedor, debió registrarse con anterioridad.

Solo hace falta colocar el botón y esperar el mensaje de confirmación.

Al eliminar un proveedor se eliminan todos sus teléfonos.

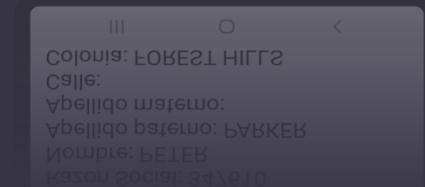


Esta opción lista todos los atributos de todos los proveedores registrados.



Esta opción es muy útil, para obtener la razón social de cualquier probador en caso de que te halla olvidado.

Nosotros recomendamos no realizar esta opción, ya que si eliminas un proveedor, se eliminarán todas las registros con ese proveedor.



Si no aparece el proveedor que buscas, revisa que no lo has borrado o le has cambiado el nombre.

No olvide las razones sociales, se utilizan para varias acciones.

Operaciones con teléfonos.

Los Proveedores registran uno o varios teléfonos para mantenerse en contacto.

Lo primero que puedes hacer es consultar los de cualquier proveedor que ya esté registrado, para eso solo debes llenar el campo de la razón social.



También puedes insertar un nuevo teléfono de contacto a cualquier proveedor. Para esto debes llenar los campos de razón social y teléfono.



Adicionalmente en caso de un error, se puede actualizar el teléfono. O en su defecto también puedes borrarlo con su botón. No se puede modificar la razón social.



La aplicación no muestra de manera automática los teléfonos del proveedor, solo lo colocamos para hacer mas entendible la explicación

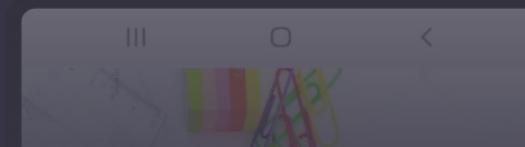


Operaciones con los productos.

Ahora revisaremos las operaciones que se pueden hacer los productos, mismos que brindan los proveedores y que compran los clientes.

Aquí se encuentran las opciones de:

- Insertar un nuevo producto.
- Actualizar producto.
- Borrar producto.
- Listar los productos.



Guardar nuevo Producto y Actualizar uno existente.

Para insertar un producto nuevo se deben llenar todos los valores que se solicitan. En este punto puedes poner el STOCK que deseas, indicando la razón social del proveedor que lo está brinda

Papeleria Gastón
Guardar Producto

349813
16457845
COMPÁS
12
BARRILITO
18.5
30
COLOR AZUL
8-3-2021
GUARDAR NUEVO PRODUCTO

Para actualizar un producto existente se llenan todos los atributos, con excepción de STOCK, ya que ello debe brindar algún proveedor; esto para evitar que se modifique sin distinción, generando problemas como por ejemplo, que se aumente el STOCK, sin que lo brindara nadie.

Papeleria Gastón
Actualizar Producto

16457845
COMPÁS
12
APPLE
200
UN COMPÁS NORMAL PERO CARO
8-3-2021
ACTUALIZAR

Finalmente está la opción que permite aumentar el STOCK de un producto existente. Para esto se indica cuánto se agrega (no el total) y que proveedor nos lo da. No se puede modificar su código de barras.

Papeleria Gastón
Actualizar Producto

349813
16457845
articulo
PrecioC
Marca
PrecioV
Stock
Descripción
Fecha (MM-DD-AAAA)
GUARDAR NUEVO PRODUCTO
15
AGREGAR STOCK

No es posible que el mismo proveedor brinde más de una vez el mismo producto.

Te sugerimos que tengas en cuenta que:

- La el código de barras es un valor que no se puede repetir, ponle uno distinta a cada producto.
- Si quieres actualizar uno, debió ser creado con anterioridad, sino créalo.
- Asegúrate de escribir correctamente el código, de lo contrario no actualizarás nada.
- No olvides llenar todos los campos con lo que se te pide.

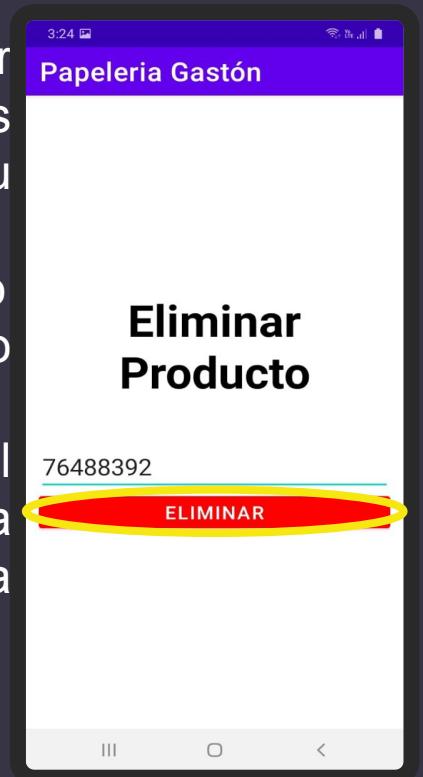


Eliminar Productos y listarlos.

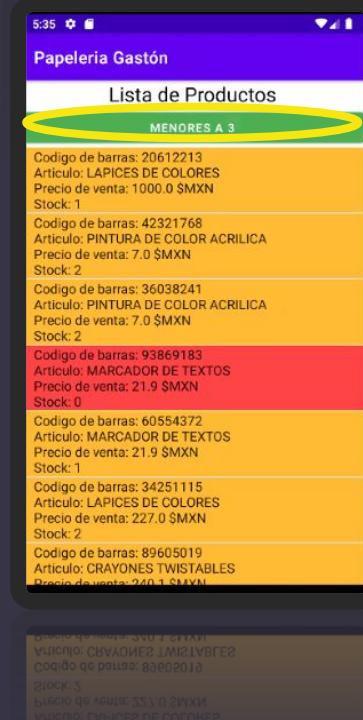
Para poder eliminar un producto es necesario conocer su código de barras.

Ten cuidado de no eliminar un producto por error.

En caso de que el código de barras sea incorrecto no elimina nada.



Nosotros recomendamos no realizar esta opción, ya que si eliminas un producto, se eliminaran todas las ventas donde se halla incluido ese producto.



Aquí se muestra la lista con todos los atributos de cada producto. Aquí no se muestra el proveedor, solo el stock actual.

Se muestra de color amarillo todos aquellos productos con stock menor a 3 y de color rojo los de stock igual a cero.

No olvide los códigos de barras, se utilizan para varias acciones.

Operaciones con los clientes.

Para los clientes, las operaciones son las mismas que con los proveedores, con excepción de que ahora se registran correos como medio de contacto, no teléfonos.

Aquí se encuentran las opciones de:

- Insertar o actualizar un cliente.
- Correos.
- Borrar un cliente.
- Listar los clientes.



Guardar nuevo Cliente y Actualizar uno existente.

Si se desea registrar un nuevo cliente, es necesario llenar todos los campos disponibles. En esta ocasión el RFC es una cadena alfanumérica.

Y dado que forzosamente todos los clientes necesitan un correo, se solicita uno.



En el caso de un error, se puede solucionar actualizando los valores del cliente, pero en este caso se debe dejar vacío el campo de correo. Si se quiere insertar otro, se puede hacer en otra página. No es posible modificar su RFC.



Te sugerimos que tengas en cuenta que:

- La el RFC es un valor que no se puede repetir, ponle uno distinta a cada cliente.
- Si quieres actualizar uno, debió ser creado con anterioridad, sino créalo.
- Asegúrate de escribir correctamente el RFC, de lo contrario no actualizamos nada.
- No olvides llenar todos los campos con lo que se te pide.

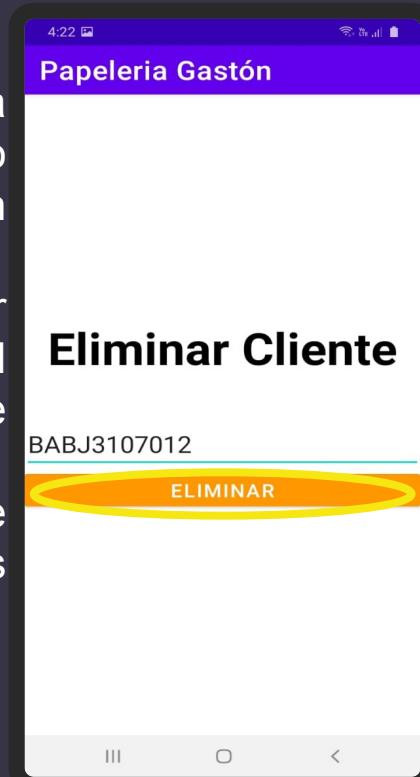


Eliminar Clientes y listarlos.

Para poder eliminar a un cliente, debió registrarse con anterioridad.

Solo hace falta colocar el botón y esperar el mensaje de confirmación.

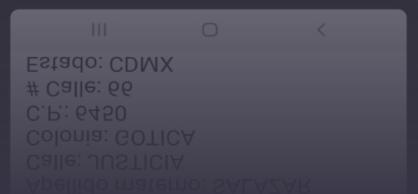
Al eliminar un cliente se eliminan todos sus correos.



Nosotros recomendamos no realizar esta opción, ya que si eliminas un cliente, se eliminaran todas las ventas que halla realizado..



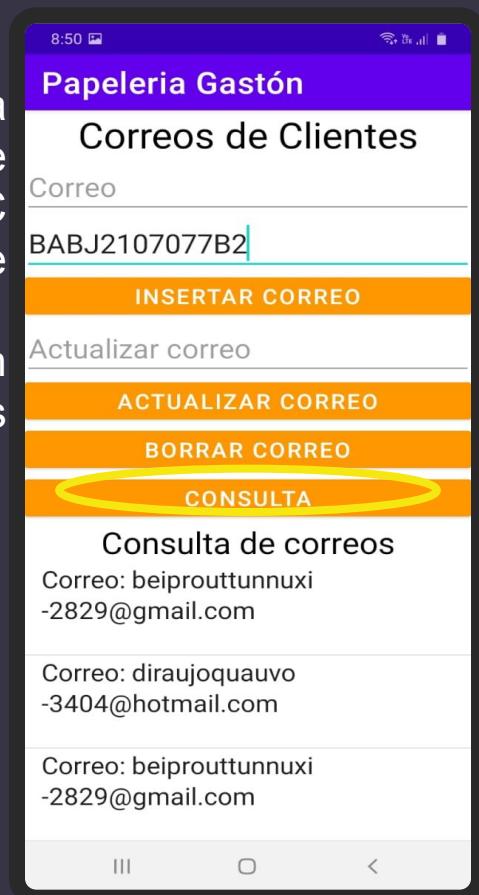
En la lista de clientes se muestran todos sus atributos. Esto es útil ya que para realizar las ventas es necesario conocer el valor del RFC de cada cliente.



No olvide los RFC, se utilizan para varias acciones.

Operaciones con correos.

Para realizar una consulta solo se necesita el RFC del cliente al que pertenece. Esta opción muestra todos sus correos.



Para insertar uno nuevo se debe llenar el campo de correo. En caso de que se desee actualizarlo, se debe escribir el anterior y llenar el campo restante, esto para saber exactamente a cual correo se refiere.



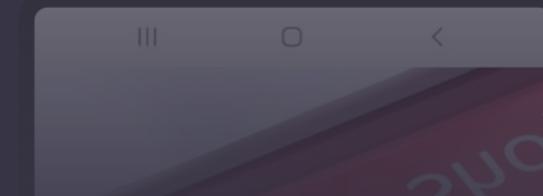
En este caso recomendamos más eliminar el correo e inserte uno nuevo.

Operaciones con Ventas.

Finalmente llegamos a las ventas, las ventas las realizan los clientes y estas incluye productos.

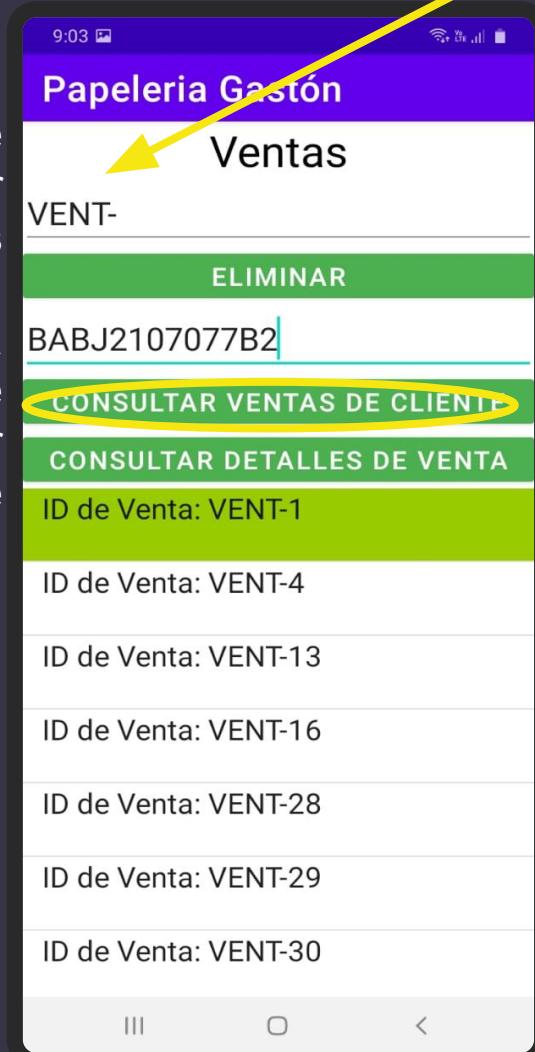
Aquí se encuentran las opciones:

- Listar ventas
- Crear una nueva venta
- Ganancias.



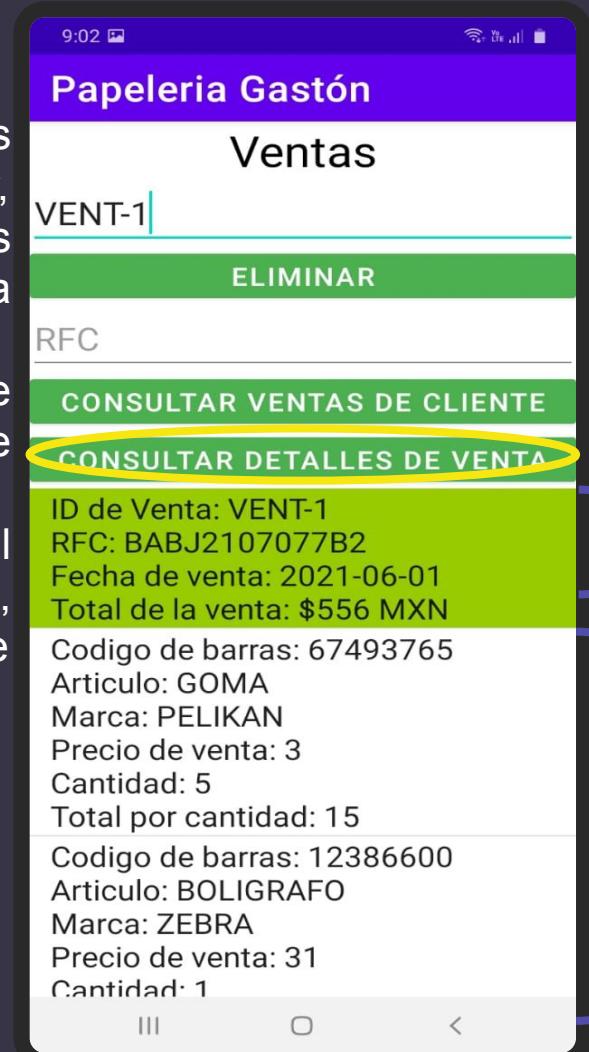
Listar ventas.

Lo primero que puede realizar es consultar las ventas realizadas por un cliente. Se muestra una lista de sus números de venta, para así poder consultar la que prefiera.



Aquí ya está escrito VENT-, que es algo que pertenece a todos los números de venta, solo faltaría a completar el número. Si lo borra, solo vuelva a escribirlo.

Como ya sabemos que venta consultar, podemos ver los detalles de cada una de las ventas. Esto es equivalente a un ticket de compra. Si no muestra el total de venta primero, por favor, da clic de nuevo.



Atributos de la venta.

Atributos de cada producto.

No olvide los números de venta, se utilizan para varias acciones.

Crear una nueva venta.

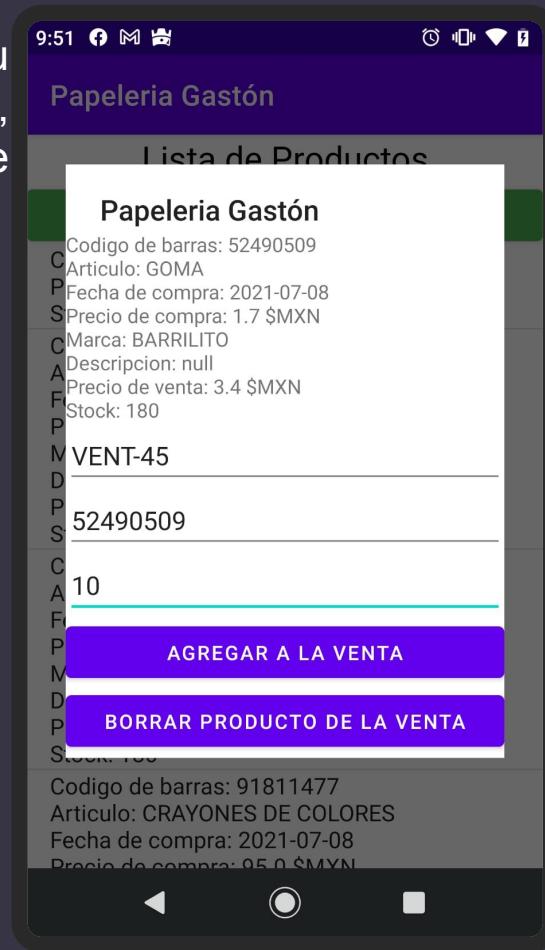
Al crear una venta se muestra la lista de productos, esto solo para apoyar para escribir el código de barras.

Se debe indicar el RFC del cliente que realiza la venta.



No es posible que tu insertes el número de venta, ni el total de venta, ya que estos valores se calculan.

Recuerda consultar el número de venta para poder comprobar tu ticket.



Para agregar un producto el procedimiento es el mismo, pero ahora se agrega el número de venta y se quita el proveedor y al fecha.

No está permitido comprar más artículos de los disponibles en el STOCK. Ten en cuenta que si ya pusiste un producto en tu venta, no es posible aumentar el stock ni regresarlo, tendrá que realizarse la venta tal y como se registró.



Ganancias.

Para el cálculo de dinero obtenido en las ventas se necesitan dos fechas para mostrar todo lo vendido en ese periodo.

Si se desea de un solo día, se debe poner la misma fecha en ambos espacios.



Ten en cuenta que si eliminas productos o clientes, las ventas en las que participan se borraran también y por lo tanto se modificara lo obtenido en esta consulta.



Conclusiones:

David:

Para este proyecto considero que el mayor problema fue el tiempo, ya que por el poco tiempo no pudimos realizar más acciones para la aplicación que fueran más óptimas y más entendibles para el usuario final, aunado a que en teoría o en el laboratorio hubieron temas que vimos demasiado tarde que nos hubieran sido de utilidad, como los JOINS que por la fecha de entrega no pudimos aplicar demasiado. Agradezco a ambos profesores, al de teoría y al de laboratorio por sacrificar su tiempo libre para completar el temario a pesar de todas las complicaciones que tuvimos en este curso y que nos permitió realizar la parte de diseño de los modelos, la creación y llenado de tablas SQL.

Considero que parte vital del éxito del proyecto fue una correcta planeación de tiempos y sobre todo un gran equipo que entendió la situación y la necesidad de completar el proyecto con la mejor disposición, como se comentó en la introducción; nos conocemos desde primer semestre y fue un alivio saber que mi compañero estaba ahí para resolver mis dudas y problemas con los que me encontré en cierto momento. Como se ve en las actividades de la bitácora, yo realicé la mayoría de documentos php para la implementación, cosa que fue un reto ya que no había trabajado con él antes, por fortuna se encontraron unos videos donde se realiza algo parecido y que ayudaron a marcar un camino a seguir. Por último no tengo nada más que agregar que me divertí, al desarrollar este proyecto, al frustrarme por encontrar otro error y descifrar cómo solucionarlo, a hacer una reunión más para que ahora si ya se termine.

Daniel:

El proyecto final cumple con poner a prueba los conocimientos adquiridos en teoría y busca adentrarnos de una manera lo más cercana posible a la realidad a una aplicación de las bases de datos para resolver una problemática. Poniendo en contexto el proyecto con la situación actual que vivimos como estudiantes a distancia y el tiempo ajustado con el que se contó, se pudo implementar por decir lo más justo los conceptos más importantes de la materia pudiendo aplicar lo visto en cada etapa de requerimientos, diseño e implementación. Considero que el proyecto no hubiera podido llevarse a cabo con éxito de no ser por el apoyo y la disponibilidad de los profesores tanto de teoría como de laboratorio, dado que muchos de los requerimientos del proyecto eran demasiado avanzados, también fue tarea nuestra y un obstáculo el poder trabajar en conjunto e investigar por nuestra cuenta los temas faltantes. En este caso el profesor nos facilitó muchas herramientas para poder ir cumpliendo en mayor o menor medida con cada punto del proyecto. La parte más complicada fue poder llevar lo aprendido e implementado en base de datos, con la parte extra que fue poder hacer más tangible el proyecto en forma de una aplicación móvil. Puesto a que no contábamos con conocimientos de estas áreas, nos tocó poner un esfuerzo extra si queríamos lograr un trabajo cuanto menos decente. Considero que una buena planeación fue vital para nuestro trabajo en equipo, una atención para resolver dudas por parte del profesor y tener el apoyo de nuestras casas. En cuanto a división de trabajo, considero que no hubo problema alguno pues se dividió de forma justa, tal y como se explica en la bitácora, una ventaja con la que contamos fue que los miembros del equipo ya conocemos perfectamente nuestras capacidades y habilidades. En resumen puedo afirmar que los objetivos se cumplieron, realizamos un análisis del problema con un diseño siguiendo todas las reglas, pusimos en práctica los conceptos vistos en clase, las sentencias en lenguaje SQL para la creación de tablas, inserción de elementos y pudimos crear una solución gráfica vistosa para el cliente.