



Universidad Nacional Autónoma de México

División de Ingeniería Eléctrica

Facultad de Ingeniería

Departamento de Ingeniería en Control



Bases de Datos

Grupo 1

Proyecto Final

Integrantes del equipo Toci Software:

**Arreguin Portillo Diana Laura
Brito Serrano Miguel Ángel
Marentes Degollado Ian Paul
Meza Vega Hugo Adrián**

Profesor Ing. Fernando Arreola Franco

Semestre: 2021-2

Fecha de entrega: 15 / 08 / 2021

ÍNDICE

OBJETIVO	3
DESCRIPCIÓN DEL PROBLEMA	3
PARTE UNO	3
PARTE DOS	4
INTRODUCCIÓN	5
PLAN DE TRABAJO	6
ANÁLISIS Y DISEÑO DE LA BASE DE DATOS	8
MODELO ENTIDAD-RELACIÓN	8
MODELO RELACIONAL	10
TRANSFORMACIÓN DE MER A MR	10
DIAGRAMA MODELO RELACIONAL	11
NORMALIZACIÓN	12
TABLA PROVEEDOR	12
TABLA TELEFONOPROV	13
TABLA PRODUCTO	14
TABLA CLIENTE	14
TABLA EMAILCLIENTE	15
TABLA SUMINISTRA	16
TABLA TIPO	17
TABLA TIPOPRODUCTO	17
TABLA VENTA	18
TABLA TIENE	18
TRANSFORMACIÓN DE MER A MR	19
DIAGRAMA MODELO RELACIONAL	20
IMPLEMENTACIÓN	21
IMPLEMENTACIÓN DE MODELO RELACIONAL A BASE DE DATOS	21
IMPLEMENTACIÓN Y DESCRIPCIÓN DE FUNCIONES, TRIGGERS Y STORED PROCEDURES NECESARIOS PARA CUMPLIR CON LAS REGLAS DE NEGOCIO DEL PROYECTO	21
PRESENTACIÓN	30
CONCLUSIONES	34
Arreguin Portillo Diana Laura	34
Brito Serrano Miguel Ángel	34
Marentes Degollado Ian Paul	35
Meza Vega Hugo Adrián	35
REFERENCIAS	36

OBJETIVO

El alumno analizará una serie de requerimientos y propondrá una solución que atienda a los mismos, aplicando los conceptos vistos en el curso.

DESCRIPCIÓN DEL PROBLEMA

PARTE UNO

Consiste en el diseño de una base de datos. 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:

- Al recibir el código de barras de un producto, regrese la utilidad.
- Cada que haya la venta de un artículo, deberá decrementarse el stock por la cantidad vendida de ese artículo.
- Si el valor llega a cero, abortar la transacción.
- Si hay menos de 3, emitir un mensaje.
- Dada una fecha, o una fecha de inicio y fecha de fin, regresar la cantidad total que se vendió en esa fecha/periodo.

- Permitir obtener el nombre de aquellos productos de los cuales hay menos de 3 en stock.
- De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una compra.
- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.

Tomar en cuenta las siguientes consideraciones:

- Puede haber distintas soluciones al problema.
- Los requerimientos enlistados anteriormente, deberán ser realizados por medio de PgSQL, con los elementos que se consideren adecuados para resolverlos.
- El número de venta debe tener un formato similar a "VENT-001", prefijo VENT, seguido de un guión y un número secuencial.
- Donde esté presente el atributo domicilio, está compuesto por estado, código postal, colonia, calle y número.
- El diseño debe satisfacer todos los principios de diseño, los requerimientos anteriores y un buen manejo de información.

PARTE DOS

La segunda parte de este proyecto se desarrollará de manera opcional y es bajo la decisión del equipo el desarrollar o no esta parte.

Una vez diseñada y lista la base de datos, se debe crear (de manera voluntaria) una interfaz gráfica vía app móvil o web, que permita:

1. Agregar la información de un cliente.
2. Ingresar una venta, de hasta 3 artículos, los cuales podrán seleccionarse de una lista de opciones, permitir ingresar la cantidad, calcular el costo total de cada artículo y el costo total de toda la venta. Ingresar dicha información en la base de datos, respetando todas las restricciones de integridad.

INTRODUCCIÓN

Este proyecto busca realizar algo que se podría solicitar en un ambiente real en el ámbito de trabajo donde nos pudiéramos encontrar en un futuro, se aplicaran conceptos, así como métodos visto en la clase y con ayuda tambien lo visto en el laboratorio de esta, la finalidad también de este proyecto es demostrar las habilidades aprendidas y adquiridas en la clase

Durante el desarrollo del proyecto se probará la base de datos para ver que el funcionamiento en cada etapa se desarrolle de manera adecuada, con todo el proceso del desarrollo de la base de datos nos iremos dando cuenta si existirán errores o algún otro problema en la base de datos y así poder corregir los problemas que existieran en esta, para que el dia de la entrega, se entregue algo completamente funcional.

En la parte de diseño, se buscará la mejor forma de analizar para que en el momento de realizar todo el diseño este sea claro en todo momento.

En el desarrollo del diseño del proyecto se tendrá el diseño de las tablas, cada una de estas tendrá atributos en donde se definirá que tipo de datos son estos mismo, a lado de estos se definirán las propiedades que estos tendrán, si sera necesario se usarán los contraints y la razón porque estos se usen.Por la parte del equipo es que estas tablas contengan todos los requerimientos solicitados y funcione de la manera correcta.

Para ver la importancia de las bases de datos, y lo fundamental que es el diseño de esta al momento de realizar consultas y/o un ingreso de información, esto de forma r rapida y buscando que la informacion almacenada este segura dando al cliente la confianza que ademas de tener un diseno optimo en relacion a la funcionalidad se tengan todas las medidas para evitar que la informacion que esta guarda, no se pueda vender o hacer cosas fuera de norma con toda esta informacion importante.

En el proyecto se requiere fundamentalmente usar conocimientos vistos en materias anteriores tambien, en combinación con los vistos en esta materia presente que es base de datos, se podría decir que una materia tambien muy importante para este proyecto por los conocimientos que adquirimos seria orientada a objetos, como programación estructurada, todo esto para poder hacer una buena aplicación a este proyecto.

PLAN DE TRABAJO

Se dividieron las tareas para cada integrante del proyecto al inicio de este mismo, se dieron fechas de revisión para cada avance del proyecto.

Se realizaron reuniones recurrentes mediante zoom y google meet, estas se hacían de acuerdo a un calendario establecido donde se fijaba una fecha para observar los avances del proyecto y cómo podemos mejorar parte de él, así como también dudas o observaciones del avance.

Las etapas en las que se planeó el proyecto fueron las siguientes:

- Desarrollo de la base de datos: En esta parte se realiza tanto el análisis, como el diseño y la programación de la base de datos. En esta parte es donde se diseña y analiza la entidad-relación, así como también el análisis y diseño del modelo relacional, normalización, se prueba la base de datos, se corrige diferentes problemas que pudiese llegar a tener la base de datos, se hacer la inserción de datos a la misma base de datos, y se procura que no tenga algún problema.
- Desarrollo de la interfaz gráfica: En esta parte se desarrolla la parte del diseño de la interfaz gráfica en esta etapa todo esto se optara por realizarlo en el lenguaje de programación JAVA, se decidirá esta opción ya que hay antecedentes sobre JAVA.
- Documento: Esta etapa tiene que ver con la redacción del documento formal escrito que se entregará, este contendrá todo lo de el desarrollo de la base de datos, así como el funcionamiento de este mismo, las conclusiones de cada integrante de cada integrante del proyecto como su punto de vista también del proyecto.

En la planeación del plan de trabajo el reparto de tareas se realizó de la mejor manera en la que cada integrante explotara sus habilidades y conocimientos para que el proyecto quede de la mejor manera, aquí también se puede decir que hay integrantes que pueden ofrecer un extra y esto es considerado muy bien para las tareas que requiere este proyecto.

La organización de todas las tareas se realizó de la mejor manera y la planeación del trabajo se realizó con éxito, ya que todas las partes que conformamos el proyecto estuvimos de acuerdo a la repartición de estas misma y que se sabía que se tenía el compromiso por cada etapa del proyecto.

The image consists of three vertically stacked screenshots of the Google Calendar interface, illustrating the scheduling of project-related events.

- Top Screenshot (July 2021):** Shows a weekly view from Sunday, July 4, to Saturday, July 10. A meeting titled "Reunión Proyecto 2 - 3pm" is scheduled for Tuesday, July 6, from 2 PM to 3 PM. Another meeting titled "Reunión Proyecto 3:30 - 4:30pm" is scheduled for Thursday, July 8, from 3:30 PM to 4:30 PM. The sidebar shows "Mis calendarios" with "Angel Brito" selected.
- Middle Screenshot (June 2021):** Shows a weekly view from Sunday, June 27, to Saturday, July 3. Two meetings titled "Proyecto Bases 12 - 1pm" are scheduled for Wednesday, June 30, from 12 PM to 1 PM each. The sidebar shows "Mis calendarios" with "Angel Brito" selected.
- Bottom Screenshot (August 2021):** Shows a weekly view from Sunday, August 1, to Saturday, August 7. Three meetings titled "Proyecto Bases 4:30 - 5:30pm" are scheduled for Monday, August 2, from 4:30 PM to 5:30 PM; Tuesday, August 3, from 4:30 PM to 5:30 PM; and Wednesday, August 4, from 4:30 PM to 5:30 PM. The sidebar shows "Mis calendarios" with "Angel Brito" selected.

Con ayuda de google calendar se programaron fechas para poder ir checando los avances de nuestro proyecto, esto nos ayudó bastante a poder organizarnos de manera colaborativa.

También de esta manera por google meet se programaron reuniones en la plataforma para presentar los avances a los miembros del proyecto, poder aportar ideas o consejos para que el proyecto se diseñará de una mejor manera a la que ya se tenía establecida, su trabajo de una manera excelente sin ningun atraso y con una buena idea del mismo proyecto.

ANÁLISIS Y DISEÑO DE LA BASE DE DATOS

MODELO ENTIDAD-RELACIÓN

Leímos a detalle los requerimientos de la base de datos para identificar las entidades, atributos y relaciones necesarias para llevar a cabo el desarrollo del modelo entidad-relación.

Nos apoyamos en el documento del profesor para identificar dichos elementos, a continuación se muestra la forma en que identificamos entidades y atributos.

Consiste en el diseño de una base de datos. 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:

- Al recibir el código de barras de un producto, regrese la utilidad.
- Cada que haya la venta de un artículo, deberá decrementarse el stock por la cantidad vendida de ese artículo. Si el valor llega a cero, abortar la transacción. Si hay menos de 3, emitir un mensaje.
- Dada una fecha, o una fecha de inicio y fecha de fin, regresar la cantidad total que se vendió en esa fecha/periodo.



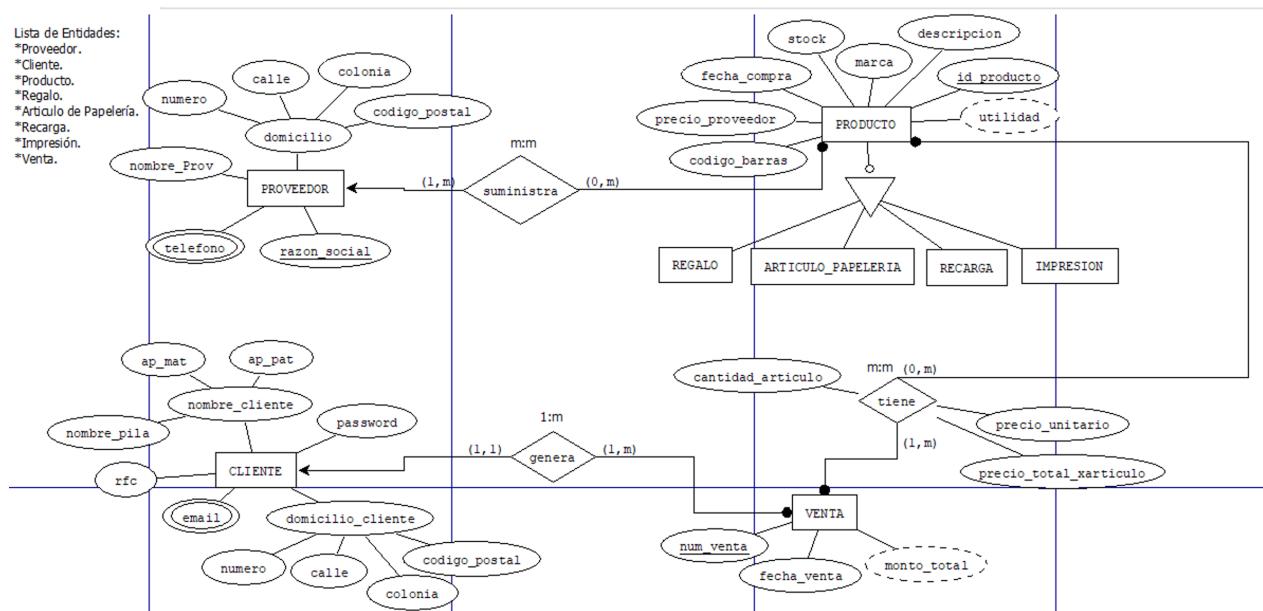
Posteriormente, enlistamos las entidades encontradas.

- Producto: Para este apartado, tomamos la decisión de utilizar el modelo extendido entidad-relación y usar una especialización para representar a los productos. Ya que en los requerimientos se habla de: regalos, artículos de papelería, impresiones y recargas. Cada uno con sus diferencias, pero sin dejar de ser productos ya que comparten atributos como la descripción, stock, precio, utilidad. Algunos atributos serán optionales, ya que dependiendo de cada tipo de producto, se utilizarán algunos atributos y otros simplemente serán nulos.
- Venta: Esta entidad hace referencia a todo lo vendido. Con esto se refiere al número de venta, precio total, IVA, fecha de venta, concepto de venta, cuya finalidad es poder ver los artículos vendidos.
- Cliente: Esta entidad se refiere a los datos personales del cliente potencial que puede llegar a adquirir algo, su RFC del cliente se tomará como la clave

Universidad Nacional Autónoma de México. Facultad de Ingeniería
 primaria de esta entidad ya que tiene varias virtudes para ser considerada así.

- Proveedor: Esta entidad hace referencia a los datos de los proveedores de la papelaria

Finalmente, presentamos el diagrama entidad-relación que se realizó de acuerdo a lo discutido entre integrantes del equipo. Cabe aclarar que se utilizó la herramienta DIA para llevar a cabo el diseño del diagrama ER.



MODELO RELACIONAL

Partiendo del modelo ER, se desarrolló el modelo Relacional correspondiente. Como primer paso, realizamos la transformación del modelo entidad-relación a modelo relacional siguiendo las reglas de transformación correspondientes.

TRANSFORMACIÓN DE MER A MR

Transformación del modelo Entidad-Relacion a modelo relacional siguiendo las reglas establecidas en clase para transformar entidades, relaciones y atributos.

PROVEEDOR={razon_social (PK) varchar(100) , nombre varchar(80), calle varchar(30), numero smallint, colonia varchar(30), estado varchar(30), codigo_postal int}

TELEFONOPROV={ razon_social (FK) varchar(100), telefono (PK) varchar(20)}

CLIENTE={rfc (PK) varchar(13), nombre_pila varchar(30), ap_paterno varchar(30), ap_materno varchar(30), calle varchar(50), numero int, colonia varchar(100), estado varchar(50), codigo_postal int, password varchar(50)}

EMAILCLIENTE={rfc (FK) varchar(13), email (PK) varchar(50) }

PRODUCTO={id_producto (PK) int, marca varchar(30), stock int, precio_unidad decimal(10,2) ,codigo_barras int, utilidad decimal(10,2), fecha_compra date, precio_proveedor decimal(10,2), descripcion varchar(100)}

TIPO={id_tipo (PK) int, descrip_tipo varchar(30)}

TIPOPRODUCTO={[id_producto (FK) int, id_tipo (FK) smallint](PK)}

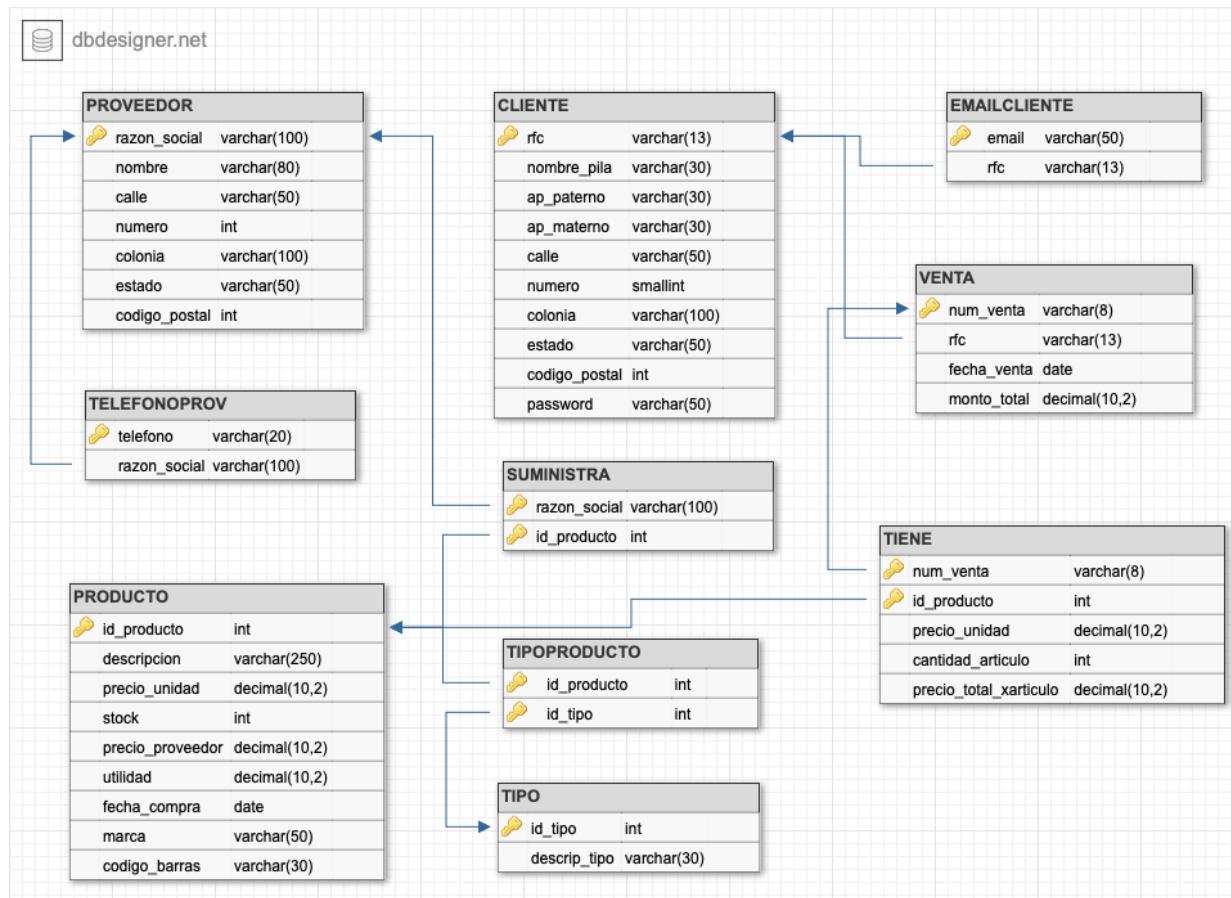
VENTA={num_venta (PK) varchar(8), fecha_venta date, monto_total decimal(10,2), rfc (FK) varchar(13)}

SUMINISTRA={[razon_social (FK) varchar(100), id_producto (FK) int](PK)}

TIENE={[num_venta (FK) varchar(8), id_producto (FK) int](PK), cantidad_articulo int, precio_unidad decimal(10,2), precio_total_xarticulo decimal(10,2)}

DIAGRAMA MODELO RELACIONAL

Decidimos, como equipo, utilizar la herramienta “DbDesigner” para desarrollar nuestro diagrama de modelo relacional. Elegimos utilizar esta herramienta ya que es gratuita, es una aplicación web lo cual nos permitió tener un desarrollo colaborativo y, además, tiene una gran ventaja ya que, una vez generado el diagrama del modelo relacional, Db Designer nos genera el código SQL para crear la base de datos. En nuestro caso nos generó el SQL para el manejador POSTGRESQL.



NORMALIZACIÓN

Para poder realizar la normalización fue necesario crear tablas con información para observar la manera en que se almacena la información y poder realizar el análisis. Para este proceso, se analizará a detalle cada una de las 10 tablas resultantes de nuestro diseño de la base de datos. Se trata de un total de 10 tablas donde verificaremos si se cumple hasta la tercera forma normal.

TABLA PROVEEDOR

razon_social [PK] character varying (100)	nombre character varying (80)	calle character varying (50)	numero integer	colonia character varying (100)	estado character varying (50)	codigo_postal integer
SERVICIOS COMERCIALES AMAZO...	Amazon México	Cuernavaca	106	Condesa	Ciudad de México	6140
OFFICE DEPOT DE MEXICO S.A. DE...	OFFICE DEPOT	Noche paz	49	Santa María Nonoalco	Ciudad de México	3700
ABASTECEDORA LUMEN S.A. DE C...	Lumen	Las flores	63	San Miguel	Estado de México	51260
HEWLETT PACKARD SERVICIOS P...	HP México	Reforma	700	Santa Fe	Ciudad de México	1210
REGALOS SIGLO XXI S.A. DE C.V.	Regalos siglo XXI	Cuernavaca	120	Condesa	Ciudad de México	6140
JOYERIA CANDILES Y REGALOS S...	Joyeria Candiles y Regalos	Rio Nazas	3800	Jardines de San Manuel	Puebla	72570
RECARGA DE PRODUCTOS Y SERVI...	MOBILMEX	Avenida Parque Via Poniente	418	Santa Anita 4a Sección	Aguascalientes	20164
MINISO MÉXICO S.A.P.I de C.V.	MINISO	Boulevard Manuel Avila Cam...	118	Lomas de Chapultepec V Sec...	Ciudad de México	11000

A

B

C

D

E

F

G

1 FN

Hay presencia de grupos de repetición, podemos observar en la parte de estado, que tenemos redundancia de información y vemos que hay atributos que no dependen de la llave primaria. Esto se debe a que el atributo “estado” no depende de la llave primaria, depende del código postal. A continuación se procede a realizar la normalización de la tabla.

A->={B,C,D,E}

G->=F

Se genera una nueva tabla llamada “estado” donde se encontrará el código postal y el estado al que hace referencia

razon_social [PK] character varying (100)	nombre character varying (80)	calle character varying (50)	numero integer	colonia character varying (100)	codigo_postal integer
SERVICIOS COMERCIALES AMAZO...	Amazon México	Cuernavaca	106	Condesa	6140
OFFICE DEPOT DE MEXICO S.A. DE...	OFFICE DEPOT	Noche paz	49	Santa María Nonoalco	3700
ABASTECEDORA LUMEN S.A. DE C...	Lumen	Las flores	63	San Miguel	51260
HEWLETT PACKARD SERVICIOS P...	HP México	Reforma	700	Santa Fe	1210
REGALOS SIGLO XXI S.A. DE C.V.	Regalos siglo XXI	Cuernavaca	120	Condesa	6140
JOYERIA CANDILES Y REGALOS S...	Joyeria Candiles y Regalos	Rio Nazas	3800	Jardines de San Manuel	72570
RECARGA DE PRODUCTOS Y SERVI...	MOBILMEX	Avenida Parque Via Poniente	418	Santa Anita 4a Sección	20164
MINISO MÉXICO S.A.P.I de C.V.	MINISO	Boulevard Manuel Avila Cam...	118	Lomas de Chapultepec V Sec...	11000

Universidad Nacional Autónoma de México. Facultad de Ingeniería

codigo_postal [PK] integer	estado character varying (50)
6140	Ciudad de México
3700	Ciudad de México
1210	Ciudad de México
11000	Ciudad de México
11435	Ciudad de México
14678	Ciudad de México
8500	Ciudad de México
3300	Ciudad de México
51260	Estado de México
56450	Estado de México
3145	Estado de México
72570	Puebla
20164	Aguascalientes

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA TELEFONOPROV

telefono [PK] character varying (20)	razon_social character varying (100)
2154780312	SERVICIOS COMERCIALES A...
5543128790	OFFICE DEPOT DE MEXICO S....
4576821435	ABASTECEDORA LUMEN S.A. ...
5643157823	HEWLETT PACKARD SERVICI...
5678145321	REGALOS SIGLO XXI S.A. DE ...
5478012453	JOYERIA CANDILES Y REGAL...
4576891653	RECARGA DE PRODUCTOS Y ...
5576824526	MINISO MÉXICO S.A.P.I de C.V.

1 FN

Cumple con primera forma normal, ya que no hay presencia de grupos de repetición y todos los atributos dependen únicamente de la llave primaria.

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA PRODUCTO

id_producto [PK] integer	descripcion character varying (250)	precio_unidad numeric (10,2)	stock integer	precio_proveedor numeric (10,2)	utilidad numeric (10,2)	fecha_compra date	marca character varying (5)	codigo_barras character varying (30)
1	Recarga telefónica telcel	100.00	100	90.00	0.00	2021-01-12	[null]	[null]
2	Recarga telefónica AT&T	100.00	100	90.00	0.00	[null]	[null]	[null]
3	Impresión blanco y negro	2.00	200	1.00	0.00	[null]	[null]	[null]
4	Impresión a color	8.00	200	5.00	0.00	[null]	[null]	[null]
5	Lápiz	10.00	50	5.00	10.00	2021-08-01	STAEDTLER	17892351
6	Pluma	15.00	100	10.00	15.00	2021-08-01	STAEDTLER	16820915
7	Paquete de gomas	35.00	50	25.00	0.00	2021-02-01	FACTIS	43813701
8	Paquete de 500 hojas bla...	800.00	20	700.00	0.00	2021-03-14	SCRIBE	76193291
9	Tijeras	30.00	50	20.00	0.00	2021-08-01	Maped	16239874
10	Resistol chico	15.00	50	10.00	0.00	2021-04-05	RESISTOL	41871209

1 FN

Cumple con primera forma normal, ya que no hay presencia de grupos de repetición y todos los atributos dependen únicamente de la llave primaria.

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA CLIENTE

rfc [PK] character varying (13)	nombre_pila character varying (30)	ap_paterno character varying (30)	ap_materno character varying (30)	calle character varying (50)	numero integer	colonia character varying (100)	estado character varying (50)	codigo_postal integer	passwd character varying (50)
APDL150898JLC	Diana Laura	Arreguin	Portillo	Avenida 503	84	San Juan de Aragón	Ciudad de México	11435	dianalaura
BSMA011296HMC	Miguel Ángel	Bravo	Serrano	Sur	12	Doctores	Ciudad de México	14678	miguelangel
MDIP240998QLP	Ian Paul	Marentes	Degollado	Avenida 503	54	San Juan de Aragón	Ciudad de México	11435	ianpaul
MVHA150998CZG	Hugo Adrián	Meza	Vega	Avenida Sur 4	84	Agrícola Oriental	Ciudad de México	8500	hugoadrian
LUGA210982PLA	Luis	García	[null]	Miguel Hidalgo	14	San Lorenzo	Estado de México	56450	luisgarcia
SADA240497UZP	Alejandra	Sánchez	Díaz	Rumania	3456	Portales Sur	Ciudad de México	3300	alejandasanchez
ROLE060695XLM	Luis Enrique	Rodrigues	[null]	Avenida Central	8	Maravillas	Estado de México	3145	luisenrique

A B C D E F G H I J

1 FN

Hay presencia de grupos de repetición, podemos observar en la parte de estado, que tenemos redundancia de información y hay atributos que no dependen de la llave primaria. Esto se debe a que el atributo “estado” no depende de la llave primaria. Depende del código postal. A continuación se procede a realizar la normalización de la tabla.

A->={B,C,D,E,F,G,J}

I->=H

Como se trata del mismo caso que en la tabla proveedor, ya que se manipula la misma información, se reutiliza la misma tabla llamada “estado” y solo se procede a modificar la tabla cliente. Por lo tanto, la normalización queda de la siguiente forma:

rfc [PK] character varying (13)	nombre_pila character varying (30)	ap_paterno character varying (30)	ap_materno character varying (30)	calle character varying (50)	numero integer	colonia character varying (100)	codigo_postal integer	passwd character varying (50)
APDL150898JLC	Diana Laura	Arreguin	Portillo	Avenida 503	84	San Juan de Aragón	11435	dianalaura
BSMA011296HMC	Miguel Ángel	Brito	Serrano	Sur	12	Doctores	14678	miguelangel
MDIP240998QLP	Ian Paul	Marentes	Degollado	Avenida 503	54	San Juan de Aragón	11435	ianpaul
MVHA150998CZG	Hugo Adrián	Meza	Vega	Avenida Sur 4	84	Agricola Oriental	8500	hugoadrian
LUGA210982PLA	Luis	García	[null]	Miguel Hidalgo	14	San Lorenzo	56450	luisgarcia
SADA240497UZP	Alejandra	Sánchez	Díaz	Rumania	3456	Portales Sur	3300	alejandasanchez
ROLE060695XLM	Luis Enrique	Rodrigues	[null]	Avenida Central	8	Maravillas	3145	luisenrique

codigo_postal [PK] integer	estado character varying (50)
6140	Ciudad de México
3700	Ciudad de México
1210	Ciudad de México
11000	Ciudad de México
11435	Ciudad de México
14678	Ciudad de México
8500	Ciudad de México
3300	Ciudad de México
51260	Estado de México
56450	Estado de México
3145	Estado de México
72570	Puebla
20164	Aguascalientes

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA EMAILCLIENTE

email [PK] character varying (50)	rfc character varying (13)
dlap-16@hotmail.com	APDL150898JLC
angelBrito@hotmail.com	BSMA011296HMC
ianpaulmarentes@gmail.com	MDIP240998QLP
hamv15@hotmail.com	MVHA150998CZG
luga@gmail.com	LUGA210982PLA
aleesd@yahoo.com	SADA240497UZP
luiserod@hotmail.com	ROLE060695XLM

1 FN

Cumple con primera forma normal, ya que no hay presencia de grupos de repetición y todos los atributos dependen únicamente de la llave primaria.

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA SUMINISTRA

razon_social [PK] character varying (100)	id_producto [PK] integer
RECARGA DE PRODUCTOS Y SERVI...	1
RECARGA DE PRODUCTOS Y SERVI...	2
HEWLETT PACKARD SERVICIOS P...	3
HEWLETT PACKARD SERVICIOS P...	4
OFFICE DEPOT DE MEXICO S.A. DE...	5
OFFICE DEPOT DE MEXICO S.A. DE...	6
ABASTECEDORA LUMEN S.A. DE C....	7
OFFICE DEPOT DE MEXICO S.A. DE...	8
OFFICE DEPOT DE MEXICO S.A. DE...	9
ABASTECEDORA LUMEN S.A. DE C....	10

1 FN

Cumple con primera forma normal, ya que no hay presencia de grupos de repetición y todos los atributos dependen únicamente de la llave primaria.

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA TIPO

id_tipo [PK] integer	descrip_tipo character varying (30)
1	Recarga telefónica
2	Regalo
3	Impresión
4	Artículo de papelería

1 FN

Cumple con primera forma normal, ya que no hay presencia de grupos de repetición y todos los atributos dependen únicamente de la llave primaria.

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA TIOPRODUCTO

id_product [PK] integer	id_tipo [PK] integer
1	1
2	1
3	3
4	3
5	4
6	4
7	4
8	4
9	4
10	4

1 FN

Cumple con primera forma normal, ya que no hay presencia de grupos de repetición y todos los atributos dependen únicamente de la llave primaria.

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA VENTA

num_venta [PK] character varying (8)	rfc character varying (13)	fecha_venta date	monto_total numeric (10,2)
VENT-1	APDL150898JLC	2021-10-08	110.00
VENT-2	MDIP240998QLP	2021-10-08	900.00

1 FN

Cumple con primera forma normal, ya que no hay presencia de grupos de repetición y todos los atributos dependen únicamente de la llave primaria.

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TABLA TIENE

num_venta [PK] character varying (8)	id_producto [PK] integer	precio_unidad numeric (10,2)	cantidad_articulo integer	precio_total_xarticulo numeric (10,2)
VENT-1	5	10.00	2	20.00
VENT-1	6	15.00	3	45.00
VENT-1	11	10.00	1	10.00
VENT-1	14	35.00	1	35.00
VENT-2	20	200.00	2	400.00
VENT-2	21	250.00	2	500.00

1 FN

Cumple con primera forma normal, ya que no hay presencia de grupos de repetición y todos los atributos dependen únicamente de la llave primaria.

2 FN

De acuerdo con los datos presentes en la tabla, podemos afirmar que se cumple con la segunda forma normal ya que no tenemos una llave primaria compuesta.

3 FN

Cumple con la tercera forma normal, ya que no hay presencia de dependencias funcionales transitivas.

TRANSFORMACIÓN DE MER A MR

Debido a la normalización, se requiere reflejar los cambios en las transformaciones de MER a MR.

PROVEEDOR={razon_social (PK) varchar(100) , nombre varchar(80), calle varchar(30), numero smallint, colonia varchar(30), codigo_postal (FK) int}

TELEFONOPROV={ razon_social (FK) varchar(100), telefono (PK) varchar(20)}

CLIENTE={rfc (PK) varchar(13), nombre_pila varchar(30), ap_paterno varchar(30), ap_materno varchar(30), calle varchar(50), numero int, colonia varchar(100), codigo_postal (FK) int, password varchar(50)}

EMAILCLIENTE={rfc (FK) varchar(13), email (PK) varchar(50) }

ESTADO={codigo_postal (PK) int ,estado varchar(30)}

PRODUCTO={id_producto (PK) int, marca varchar(30), stock int, precio_unidad decimal(10,2) ,codigo_barras int, utilidad decimal(10,2), fecha_compra date, precio_proveedor decimal(10,2), descripcion varchar(100)}

TIPO={id_tipo (PK) int, descrip_tipo varchar(30)}

TIPOPRODUCTO={[id_producto (FK) int, id_tipo (FK) smallint](PK)}

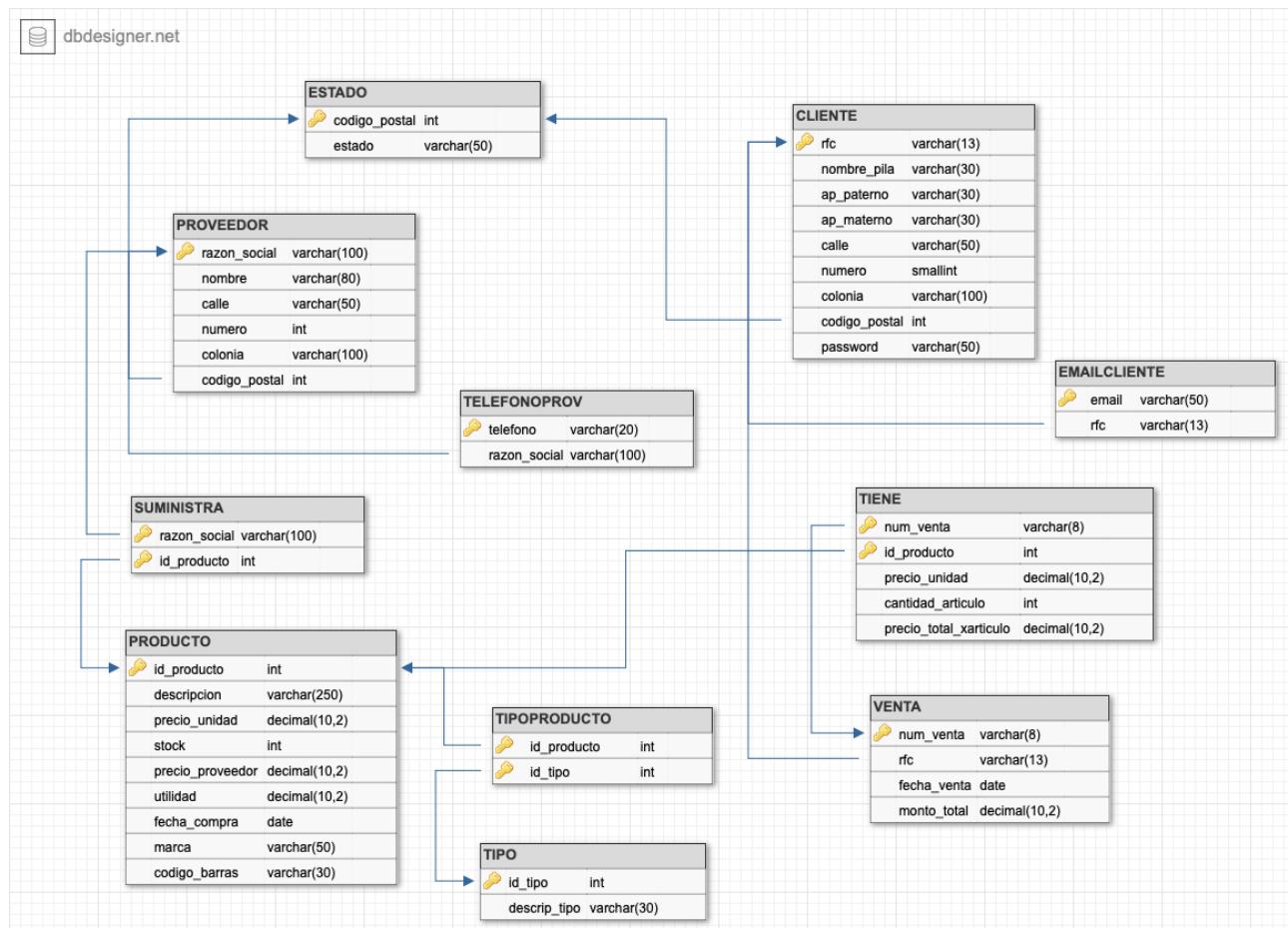
VENTA={num_venta (PK) varchar(8), fecha_venta date, monto_total decimal(10,2), rfc (FK) varchar(13)}

SUMINISTRA={[razon_social (FK) varchar(100), id_producto (FK) int](PK)}

TIENE={[num_venta (FK) varchar(8), id_producto (FK) int](PK), cantidad_articulo int, precio_unidad decimal(10,2), precio_total_xarticulo decimal(10,2)}

DIAGRAMA MODELO RELACIONAL

Posteriormente, de la misma forma, se modifica el diagrama del modelo relacional en DbDesigner debido a la normalización.



IMPLEMENTACIÓN

IMPLEMENTACIÓN DE MODELO RELACIONAL A BASE DE DATOS

Gracias a la herramienta DbDesigner, se simplificó la implementación de la base de datos en el manejador, ya que dicha herramienta nos generó el query para ejecutar directamente en postgresql.

IMPLEMENTACIÓN Y DESCRIPCIÓN DE FUNCIONES, TRIGGERS Y STORED PROCEDURES NECESARIOS PARA CUMPLIR CON LAS REGLAS DE NEGOCIO DEL PROYECTO

De acuerdo a lo solicitado en los requerimientos se realizaron las siguientes funciones:

- Para cumplir con el requerimiento “El número de venta debe tener un formato similar a “VENT-001”, prefijo VENT, seguido de un guión y un número secuencial” se propuso el crear una función que hará uso de una secuencia para regresar el número de venta con el formato “VENT-#”.
Solución: El primer paso fue crear una secuencia cíclica que comienza en 1 y con incremento unitario hasta 999999.

```
--Secuencia para la parte del id de venta
CREATE SEQUENCE numervent
START WITH 1
INCREMENT BY 1
MAXVALUE 999999
MINVALUE 1
CYCLE;
```

El segundo paso consiste en crear una función que regresa una cadena de caracteres que corresponde al número de venta en el formato requerido. A partir de la secuencia creada y el uso de las funciones “CONCAT” y “CAST” se crea el número de venta.

```
--Función que se encarga de generar el id de venta en formato "VENT-#"
CREATE OR REPLACE FUNCTION idventa() RETURNS VARCHAR AS $idventa$
DECLARE identificador varchar(8);
BEGIN
identificador:=CONCAT('VENT-',CAST((SELECT nextval('numervent')) AS VARCHAR));
RETURN identificador;
END;
$idventa$ LANGUAGE plpgsql;
```

Universidad Nacional Autónoma de México. Facultad de Ingeniería
Probando la función para verificar el formato correcto del número de venta:

<code>SELECT idventa();</code>
a Output Explain Messages
idventa character varying
VENT-9

- Al recibir el código de barras de un producto, regrese la utilidad.

Solución: La solución propuesta para este punto es una función que recibe un *varchar* que en este caso, se trata del código de barras y nos regresará un dato de tipo *decimal*, el cual representa la utilidad del producto requerido. Declaramos una variable del mismo tipo para almacenar la búsqueda de la utilidad y la retornamos al final de la función.

```
--Funcion que al ingresar el codigo de barras nos regrese la utilidad
CREATE FUNCTION utilidaddc(codbarra varchar(30))
RETURNS DECIMAL AS $utilidaddc$
DECLARE utilidadup decimal = (SELECT utilidad FROM producto WHERE codbarra = codigo_barras);
BEGIN
    RETURN utilidadup;
END;
$utilidaddc$ LANGUAGE plpgsql;
```

Probamos el correcto funcionamiento del código:

30 <code>SELECT utilidaddc('17892351');</code>	30 <code>SELECT utilidaddc('17892351');</code>
31 <code>SELECT utilidaddc('43813701');</code>	31 <code>SELECT utilidaddc('16820915');</code>
32 <code>SELECT utilidaddc('16820915');</code>	32
33	
Data Output Explain Messages Notific	Data Output Explain Messages Notific
utilidaddc numeric	utilidaddc numeric
1 10.00	1 15.00

Verificando la salida en nuestra base de datos:

<code>SELECT * FROM PRODUCTO WHERE codigo_barras = '17892351';</code>	<code>SELECT * FROM PRODUCTO WHERE codigo_barras = '16820915';</code>
Data Output Explain Messages Notifications	Data Output Explain Messages Notifications
id_producto [PK] integer	utilidaddc numeric
5 Lápiz	10.00
<code>SELECT * FROM PRODUCTO WHERE codigo_barras = '16820915';</code>	
Data Output Explain Messages Notifications	Data Output Explain Messages Notifications
id_producto [PK] integer	utilidaddc numeric
6 Pluma	15.00

Como se puede observar en las tablas, la información devuelta por la función es correcta.

- Cada que haya la venta de un artículo, deberá decrementarse el stock por la cantidad vendida de ese artículo. Si el valor llega a cero, abortar la transacción. Si hay menos de 3, emitir un mensaje.

Solución: Para cumplir este requerimiento fue necesario crear un trigger, que se ejecute al realizar una inserción en la tabla “tiene”. Ya que la base de datos recibe una venta. Dicha venta se almacena en la tabla “venta” y posteriormente se almacenan los productos comprados, en dicha venta, en la tabla “tiene”.

Adicional a este requerimiento, el trigger también se encarga de calcular la utilidad de cada producto que se compra, realizando la resta del precio al público menos el precio del proveedor y multiplicando por la cantidad comprada.

El primer paso fue crear la función que ejecutará el trigger. En dicha función se programa, haciendo uso de estructuras de control, las actualizaciones y cambios requeridos.

```

3 CREATE OR REPLACE FUNCTION atualiza_stock_utilidad() RETURNS TRIGGER AS $$ 
4     DECLARE stock_disp int, aux int, act_utilidad decimal(10,2);
5     BEGIN
6         stock_disp=(SELECT stock FROM producto WHERE id_producto=new.id_producto);
7         IF ( stock_disp > 0) THEN
8             IF (new.cantidad_articulo<=stock_disp) THEN
9                 aux=(SELECT stock FROM producto WHERE id_producto=new.id_producto) - new.cantidad_articulo;
10                UPDATE producto SET stock=aux WHERE id_producto=new.id_producto;
11                act_utilidad=((SELECT precio_unidad FROM producto WHERE id_producto=new.id_producto)-
12                (SELECT precio_proveedor FROM producto WHERE id_producto=new.id_producto))*new.cantidad_articulo+
13                (SELECT utilidad FROM producto WHERE id_producto=new.id_producto);
14                UPDATE producto SET utilidad=act_utilidad WHERE id_producto=new.id_producto;
15                stock_disp=(SELECT stock FROM producto WHERE id_producto=new.id_producto);
16                IF stock_disp<=3 THEN
17                    RAISE NOTICE 'Hay % productos en stock del producto con id: %', stock_disp,new.id_producto;
18                END IF;
19                ELSE
20                    RAISE NOTICE 'Se ha cancelado la transacción. No hay stock suficiente del producto con id: %', new.id_producto;
21                END IF;
22            ELSE
23                RAISE NOTICE 'Se ha cancelado la transacción. No hay stock del producto con id: %', new.id_producto;
24                RETURN NULL;
25            END IF;
26            RETURN new;
27        END $$ LANGUAGE plpgsql;
```

El segundo paso fue crear el trigger y especificar que se ejecute la función anterior antes de hacer una inserción a la tabla “tiene”.

```
--Creación del trigger que hace uso de la función atualiza_stock_utilidad()
CREATE TRIGGER atualiza_stock_utilidad
BEFORE INSERT ON tiene
    FOR EACH ROW EXECUTE FUNCTION atualiza_stock_utilidad();
```

Probamos el funcionamiento del trigger simulando la venta donde se compra 1 sacapuntas, 1 engrapadora y 1 paquete de plumones.

id_producto [PK] integer	descripcion character varying (250)	precio_unidad numeric (10,2)	stock integer	precio_proveedor numeric (10,2)	utilidad numeric (10,2)
16	Sacapuntas	125.00	50	100.00	0.00
17	Engrapadora	250.00	50	220.00	0.00
18	Paquete de plumones	600.00	30	550.00	0.00

Universidad Nacional Autónoma de México. Facultad de Ingeniería
 El total de nuestra venta es de \$975.00. Procedemos a almacenar la venta en la tabla “venta” con el rfc ‘BSMA011296HMC’ y con la fecha ‘2021-08-15’:

```
INSERT INTO venta(num_venta, rfc, fecha_venta, monto_total) VALUES
(idventa() , 'BSMA011296HMC' , '2021-08-15' , 975.00);
```

Verificamos la inserción de la venta:

Output Explain Messages Notifications			
num_venta [PK] character varying (8)	rfc character varying (13)	fecha_venta date	monto_total numeric (10,2)
VENT-8	BSMA011296HMC	2021-08-15	975.00

Después, agregamos los artículos comprados a la tabla “tiene”. Es en este momento donde se ejecutará el trigger y actualizará el stock y la utilidad de cada producto. Insertamos y verificamos que se hayan ingresado los registros a la tabla “tiene”:

```
INSERT INTO tiene(num_venta, id_producto, precio_unidad, cantidad_articulo, precio_total_xarticulo) VALUES
('VENT-8', 16, 125.00, 1, 125.00), ('VENT-8', 17, 250.00, 1, 250.00), ('VENT-8', 18, 600.00, 1, 600.00);
SELECT * FROM tiene;
```

Output Explain Messages Notifications				
num_venta [PK] character varying (8)	id_producto [PK] integer	precio_unidad numeric (10,2)	cantidad_articulo integer	precio_total_xarticulo numeric (10,2)
VENT-8	16	125.00	1	125.00
VENT-8	17	250.00	1	250.00
VENT-8	18	600.00	1	600.00

Finalmente, consultamos la tabla “producto” donde debemos ver reflejado el decremento del stock y el aumento en la utilidad de cada producto.

```
SELECT * FROM producto; --Consulta para verificar los cambios en el/los producto(s).
```

Output Explain Messages Notifications						
id_producto [PK] integer	descripcion character varying (250)	precio_unidad numeric (10,2)	stock integer	precio_proveedor numeric (10,2)	utilidad numeric (10,2)	
16	Sacapuntas	125.00	49	100.00	25.00	
17	Engrapadora	250.00	49	220.00	30.00	
18	Paquete de plumones	600.00	29	550.00	50.00	

Como se puede observar, el trigger se encargó, de manera correcta, de realizar el decremento del stock y el cálculo de la utilidad de cada producto.

- Dada una fecha, o una fecha de inicio y fecha de fin, regresar la cantidad total que se vendió en esa fecha/periodo.

Solución: Para este punto proponemos dos funciones, la primera de ellas recibirá una fecha (tipo *date*) como parámetro y nos devolverá un tipo de dato *decimal* en el cual estará contenida la cantidad total vendida en la fecha proporcionada. Declaramos una variable que almacenará la búsqueda, en

Universidad Nacional Autónoma de México. Facultad de Ingeniería dicha búsqueda haremos uso de la función de agregación *SUM*, la cual se encarga de sumar todos los valores de la columna especificada. Al final de la función retornamos el resultado de nuestra búsqueda, almacenado en nuestra variable *cant_vendida*.

```
--Funcion que al ingresar una fecha regrese la cantidad total que se vendio
CREATE OR REPLACE FUNCTION cantidad_vendida(fecha DATE)
RETURNS DECIMAL AS $cantidad_vendida$
DECLARE cant_vendida decimal = (SELECT SUM(monto_total) FROM venta WHERE fecha = fecha_venta);
BEGIN
    RETURN cant_vendida;
END;
$cantidad_vendida$ LANGUAGE plpgsql;
```

Realizamos una consulta para observar las ventas almacenadas en la base de datos

117 `SELECT * FROM VENTA;`

Data Output Explain Messages Notifications

	num_venta [PK] character varying (8)	rfc character varying (13)	fecha_venta date	monto_total numeric (10,2)
1	VENT-1	APDL150898JLC	2021-08-10	110.00
2	VENT-2	MDIP240998QLP	2021-08-10	900.00
3	VENT-3	ROLE060695XLM	2021-08-12	750.00

Ahora probamos el funcionamiento del código:

`SELECT * FROM cantidad_vendida('2021-08-10');`

Data Output Explain Messages Notifications

	cantidad_vendida numeric
	1010.00

Podemos observar que la función nos regresa la suma total de lo vendido en la fecha proporcionada como parámetro.

La segunda de las funciones es la que recibe una fecha de inicio y una fecha de fin como parámetros de tipo *date* y nos devolverá un dato de tipo *decimal* en el que, de misma manera que la función anterior, estará contenido la suma total de lo vendido en el rango de fechas especificado. Declaramos, para esta ocasión, tres variables, *cant_vendida_f1* en la cual almacenaremos la búsqueda dentro de la tabla venta en la cual irá la suma total de lo vendido en la fecha inicial recibida, *cant_vendida_f2* en donde almacenaremos la búsqueda en nuestra tabla venta en la cual irá la suma total de lo vendido en la fecha final ingresada, finalmente una variable llamada *cant_vendida_total* en la cual sumaremos lo obtenido de las anteriores variables y así retornar este resultado.

```
--Funcion que al ingresar un intervalo de fechas regrese la cantidad total que se vendio
CREATE OR REPLACE FUNCTION cantidad_vendida_rango(fecha1 DATE, fecha2 DATE)
RETURNS DECIMAL AS $cantidad_vendida_rango$
DECLARE cant_vendida_f1 decimal = (SELECT SUM(monto_total) FROM venta WHERE fecha1 = fecha_venta);
DECLARE cant_vendida_f2 decimal = (SELECT SUM(monto_total) FROM venta WHERE fecha2 = fecha_venta);
DECLARE cant_vendida_total decimal = (SELECT SUM(monto_total) FROM venta WHERE fecha_venta BETWEEN fecha1 AND fecha2);
BEGIN
    RETURN cant_vendida_total;
END;
$cantidad_vendida_rango$ LANGUAGE plpgsql;
```

Realizamos nuevamente una consulta a nuestra tabla venta para observar los registros incluidos en esta

117 `SELECT * FROM VENTA;`

Data Output Explain Messages Notifications

	num_venta [PK] character varying (8)	rfc character varying (13)	fecha_venta date	monto_total numeric (10,2)
1	VENT-1	APDL150898JLC	2021-08-10	110.00
2	VENT-2	MDIP240998QLP	2021-08-10	900.00
3	VENT-3	ROLE060695XLM	2021-08-12	750.00

Verificando el correcto funcionamiento de nuestro código:

`SELECT * FROM cantidad_vendida_rango('2020-10-08','2021-10-15');`

Data Output Explain Messages Notifications

cantidad_vendida_rango numeric
1760.00

Como podemos observar, en la tabla venta tenemos tres registros que van desde el 10 hasta el 12 de agosto del 2021, en nuestra función colocamos los parámetros desde el 10 de agosto de 2021 hasta el 15 de agosto del mismo año, y podemos observar que el resultado obtenido de la función es la suma de todo lo vendido en dicho rango de fechas.

- Permitir obtener el nombre de aquellos productos de los cuales hay menos de tres en stock.

Solución: En este punto, propusimos una función la cual no recibirá ningún parámetro pero que nos retornará una tabla en donde se almacenarán los nombres de aquellos productos que no haya más de tres en stock. En la estructura de la función retornamos un query (el cual es necesario para almacenar los datos obtenidos que se almacenarán en la tabla final), y realizamos una búsqueda dentro de nuestra tabla producto en donde se cumpla que el stock sea menor a tres.

```
--Funcion que obtiene los productos de los cuales hay menos de 3 en stock
CREATE OR REPLACE FUNCTION enstock()
RETURNS TABLE (Productos_con_menos_tres_stock varchar(250)) AS $enstock$
BEGIN
    RETURN QUERY
        SELECT descripcion FROM producto WHERE stock < 3;
END;
$enstock$ LANGUAGE plpgsql;
```

Probamos el funcionamiento de nuestro código:

73	<code>SELECT enstock();</code>
74	
<u>Data Output</u> Explain Mes	
	enstock character varying
1	Collar de regalo
2	Engrapadora

Vemos que la salida de la función es una tabla que nos muestra dos productos: *Collar de regalo* y *Engrapadora*, para verificar que estos datos sean correctos haremos una consulta a la tabla producto de nuestra base de datos:

<code>SELECT * FROM PRODUCTO;</code>
<u>Output</u> Explain Messages Notifications
id_producto [PK] integer descripcion character varying (250) precio_unidad numeric (10,2) stock integer precio_proveedor numeric (10,2) utilidad numeric
22 Collar de regalo 120.00 2 100.00
17 Engrapadora 250.00 1 220.00

Podemos observar que efectivamente los productos lanzados por nuestra función son los que cuentan con menos de tres en stock.

- De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una compra.

Solución: Se optó por crear una función que regrese una tabla con la información necesaria para asemejarse a una factura, esto es: nombre, apellido paterno, rfc, calle, número, colonia, código postal, monto total de la compra y fecha de la compra.

```
--Creación de la vista "factura"
CREATE OR REPLACE FUNCTION gen_factura(rfc_cons varchar(13), numvent varchar(8))
RETURNS
TABLE (
    nombre character varying,
    ap_paterno character varying,
    rfc character varying,
    calle character varying,
    numero integer,
    colonia character varying,
    codigo_postal integer,
    num_venta character varying,
    monto_total numeric,
    fecha_compra date)
AS $$ 
BEGIN
RETURN QUERY
SELECT cl.nombre_pila, cl.ap_paterno, cl.rfc, cl.calle, cl.numero, cl.colonia, cl.codigo_postal,
v.num_venta, v.monto_total, v.fecha_venta
FROM cliente cl
INNER JOIN venta v ON cl.rfc=v.rfc
WHERE cl.rfc=rfc_cons AND v.num_venta=numvent;
END;
$$ LANGUAGE plpgsql;
```

Se tomó esta decisión bajo el argumento de que una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla, una vista consta de un conjunto de columnas y filas de datos con un nombre. Una vista no existe como conjunto de valores de datos almacenados en una base de datos, ya que las filas y columnas proceden de tablas a las que se hace referencia en la consulta que define la vista. Dicho esto, nuestra función cumple con los requerimientos ya que, al ejecutarse, se obtiene una tabla no almacenada con la información necesaria para una factura.

Realizando pruebas para verificar el correcto funcionamiento de la función que crea una vista con la información de una factura.

De la tabla “venta”, elegimos un rfc junto con su número de venta:

a	Output	Explain	Messages	Notifications
	num_venta [PK] character varying (8)	rfc character varying (13)	fecha_venta date	monto_total numeric (10,2)
VENT-1		APDL150898JLC	2021-10-08	110.00
VENT-2		MDIP240998QLP	2021-10-08	900.00
VENT-3		APDL150898JLC	2021-08-12	500.00

Universidad Nacional Autónoma de México. Facultad de Ingeniería
 Posteriormente, ejecutamos la función, consultando todos los elementos que arroja la tabla que devuelve nuestra función:

```
--Prueba de la función
SELECT * FROM gen_factura('APDL150898JLC','VENT-1');
```

a	Output	Explain	Messages	Notifications																				
	<table border="1"> <thead> <tr> <th>nombre</th> <th>ap_paterno</th> <th>rfc</th> <th>calle</th> <th>numero</th> <th>colonia</th> <th>codigo_postal</th> <th>num_venta</th> <th>monto_total</th> <th>fecha_compra</th> </tr> </thead> <tbody> <tr> <td>Diana Laura</td> <td>Arreguin</td> <td>APDL150898JLC</td> <td>Avenida 503</td> <td>84</td> <td>San Juan de Aragón</td> <td>11435</td> <td>VENT-1</td> <td>110.00</td> <td>2021-10-08</td> </tr> </tbody> </table>	nombre	ap_paterno	rfc	calle	numero	colonia	codigo_postal	num_venta	monto_total	fecha_compra	Diana Laura	Arreguin	APDL150898JLC	Avenida 503	84	San Juan de Aragón	11435	VENT-1	110.00	2021-10-08			
nombre	ap_paterno	rfc	calle	numero	colonia	codigo_postal	num_venta	monto_total	fecha_compra															
Diana Laura	Arreguin	APDL150898JLC	Avenida 503	84	San Juan de Aragón	11435	VENT-1	110.00	2021-10-08															

- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.

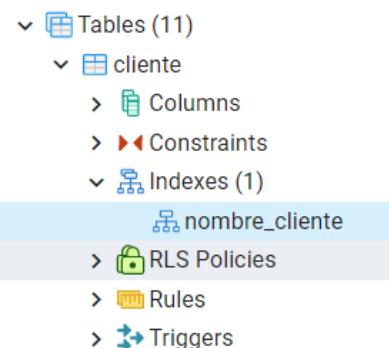
Solución: Para nuestra base de datos, creamos un índice tipo btree (ya que es el de uso más común) llamado *nombre_cliente* en nuestra tabla *cliente*. Este índice se creó con el objetivo de facilitar el acceso a los datos de dicha tabla y poder eficientizar las consultas por medio de la búsqueda del nombre.

```
199 --Creación del índice
200 CREATE INDEX nombre_cliente ON cliente USING btree (nombre_pila, ap_paterno, ap_materno);
201
```

Data	Output	Explain	Messages	Notifications
	CREATE INDEX			

Query returned successfully in 56 msec.

Verificación de la creación del índice.



PRESENTACIÓN

Para presentar la parte del desarrollo de una interfaz gráfica, optamos por la realización de una interfaz gráfica programada con el lenguaje de programación JAVA, ya que teníamos un conocimiento previo sobre el manejo de dicho lenguaje, y utilizamos Apache NetBeans como nuestro IDE.

Para poder realizar la conexión entre JAVA y Postgresql necesitamos descargar e importar en el IDE una biblioteca específica para dicha conexión.



Una vez instalado dentro de una de nuestras clases declaramos tres variables, cada una de ellas contendrá un dato que utilizaremos para la conexión a la base de datos, *URL* contendrá la ruta en la cual está la base de datos, *usuario* tendrá el usuario de la base de datos con el que queremos conectarnos y la variable *contraseña* almacenará la contraseña con la que podemos ingresar a nuestra base de datos.

```
public static final String URL = "jdbc:postgresql://localhost:5432/proyecto1";
public static final String usuario = "postgres";
public static final String contraseña = "1033255";
```

Se creó un objeto de tipo Connection con el cual obtendremos la conexión a nuestra base de datos, en el cual creamos una excepción para validar si fue exitosa la conexión o no.

```
public Connection getConnection(){
    Connection Conexion = null;

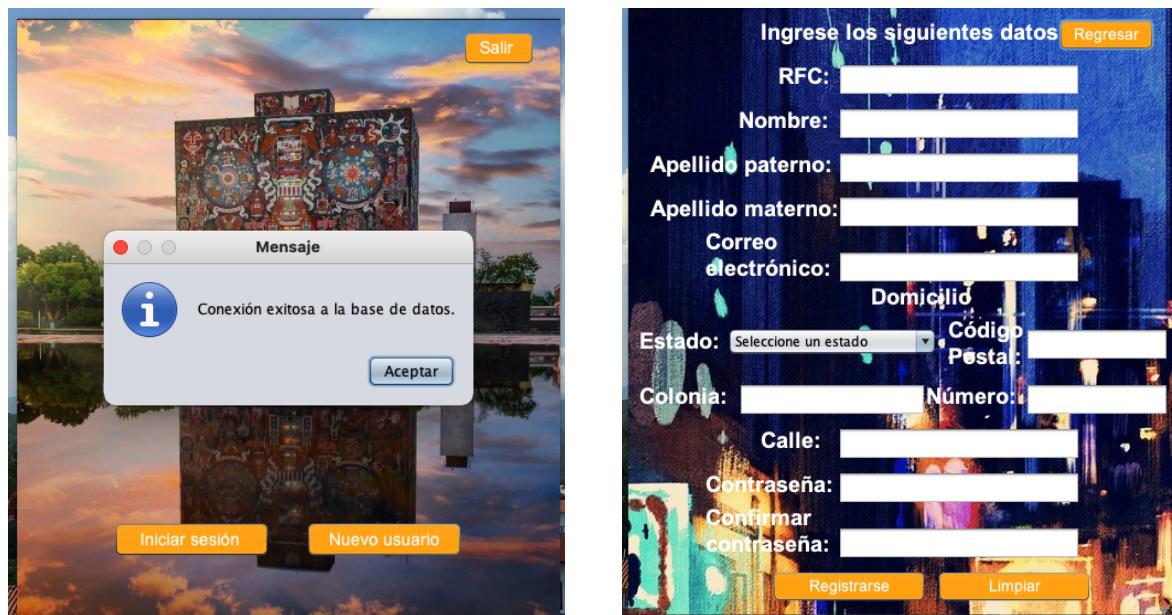
    try{
        Conexion = DriverManager.getConnection (URL, usuario, contraseña);
        //Conexion = (Connection) DriverManager.getConnection(URL, usuario, contraseña);
        JOptionPane.showMessageDialog(null, "Conexión exitosa a la base de datos.");
    }catch(Exception ex){
        System.err.println("Error: " + ex);
    }
    return Conexion;
}
```

Finalmente en otro de nuestros objetos mandamos a llamar el objeto *getConection()* para poder comenzar con la conexión

```
Connection conexion = getConnection();
```

Inicio de sesión

Para poder acceder a la aplicación, se decidió diseñar y crear una interfaz para iniciar sesión y registrar nuevos usuarios, ambas interfaces realizan la conexión a la base de datos llamada “ proyecto1”, realizan las propias operaciones, ya sean consultas, registro o validación de información.

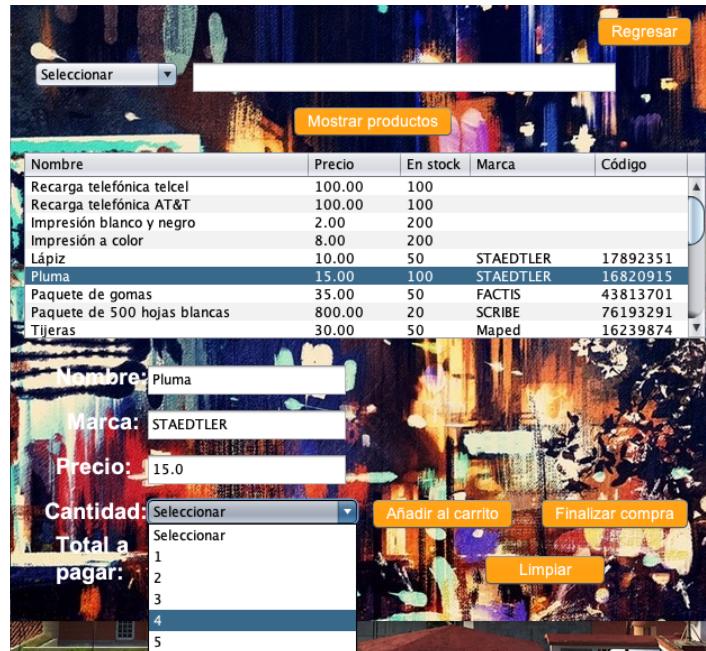


Se pensaron dos usuarios que utilizarían la aplicación, el primero de ellos sería el administrador de la papelería quién tendría acceso a la información referente a los proveedores, como se muestra a continuación.

En la interfaz anterior decidimos implementar las funciones básicas de un CRUD, es decir, crear, leer, actualizar y eliminar información de la tabla “PROVEEDOR” de nuestra base de datos, implementando así los conceptos que hemos visto en clase.

Continuando con la sesión del administrador de la papelería, se agregó una opción que mostrara los diferentes productos con los que cuenta dicha papelería, así como las características de estos, el nombre, el precio, la cantidad de artículos, etc; así mismo se implementó una búsqueda especializada con el fin de ubicar un producto por un atributo en específico.

En cuanto a la sesión de un usuario de tipo Cliente, se creó la siguiente ventana que visualizara la información de los productos y servicios registrados en la base de datos, con la opción de realizar una consulta general o en específico de un atributo, escoger la cantidad de artículos que deseaba añadir al carrito y finalizar la compra o adquisición de servicios, tomando en cuenta las restricciones e integridad de la información.



Por último, se añadió una interfaz en que el usuario podría modificar la información de su cuenta; como el nombre, los datos referentes al domicilio, el correo electrónico o la contraseña para iniciar sesión, cabe señalar que se agregó una función de encriptación de contraseñas con el fin de preservar la información del usuario.

```
|proyecto1=# SELECT * FROM cliente;
 rfc | nombre_pila | ap_paterno | ap_materno | calle | numero | colonia | codigo_postal | passwd
-----+-----+-----+-----+-----+-----+-----+-----+-----+
 APDL150898JLC | Diana Laura | Arreguin | Portillo | Avenida 503 | 84 | San Juan de Aragón | 11435 | dianalaurea
 BSMIA011296HMC | Miguel Ángel | Brito | Serrano | Sur | 12 | Doctores | 14678 | miguelangel
 MDIP240998QLP | Ian Paul | Marentes | Degollado | Avenida 503 | 54 | San Juan de Aragón | 11435 | ianpaul
 MVHA150998CZG | Hugo Adrián | Meza | Vega | Avenida Sur 4 | 84 | Agrícola Oriental | 8500 | hugoadrian
 LUGA210982PLA | Luis | García | Diaz | Miguel Hidalgo | 14 | San Lorenzo | 56450 | luisgarcia
 SADA240497UZP | Alejandra | Sánchez | Rodrígues | Rumania | 3456 | Portales Sur | 3300 | alejandrasanchez
 ROLE060695XLM | Luis Enrique | | | | | Maravillas | 3145 | luisenrique
 LOCI970319MDF | Ita | López | Chávez | Buena Vista | 4 | Cuauhtémoc | 19066 | 81dc9bdb52d04dc20036dbd8313ed055
 (8 rows)

|proyecto1=# SELECT * FROM cliente;
 rfc | nombre_pila | ap_paterno | ap_materno | calle | numero | colonia | codigo_postal | passwd
-----+-----+-----+-----+-----+-----+-----+-----+-----+
 APDL150898JLC | Diana Laura | Arreguin | Portillo | Avenida 503 | 84 | San Juan de Aragón | 11435 | dianalaurea
 BSMIA011296HMC | Miguel Ángel | Brito | Serrano | Sur | 12 | Doctores | 14678 | miguelangel
 MDIP240998QLP | Ian Paul | Marentes | Degollado | Avenida 503 | 54 | San Juan de Aragón | 11435 | ianpaul
 MVHA150998CZG | Hugo Adrián | Meza | Vega | Avenida Sur 4 | 84 | Agrícola Oriental | 8500 | hugoadrian
 LUGA210982PLA | Luis | García | Diaz | Miguel Hidalgo | 14 | San Lorenzo | 56450 | luisgarcia
 SADA240497UZP | Alejandra | Sánchez | Rodrígues | Rumania | 3456 | Portales Sur | 3300 | alejandrasanchez
 ROLE060695XLM | Luis Enrique | | | | | Maravillas | 3145 | luisenrique
 LOCI970319MDF | Ita | López | Chávez | Buena Vista | 4 | Cuauhtémoc | 19066 | 81dc9bdb52d04dc20036dbd8313ed055
 (9 rows)

proyecto1=#

```

CONCLUSIONES

Arreguin Portillo Diana Laura

Considero que en este proyecto se cumplió con el objetivo. Pudimos crear e implementar satisfactoriamente la base de datos con base en los requerimientos planteados para el proyecto y gracias a los conceptos aprendidos en clase, en conjunto con conocimientos adquiridos en el laboratorio y de la documentación de postgres. Al inicio tuvimos varias propuestas para desarrollar el proyecto, tanto de maneras de realizarlo, hasta los lenguajes que podíamos utilizar. Para poder trabajar de una manera ordenada utilizamos herramientas para trabajo colaborativo para la realización de los diagramas Entidad-Relación y el Modelo Relacional. Posterior a este proceso es donde comenzaron un poco de dificultades en cuanto al software que utilizamos para el desarrollo y el diseño mismo de la base de datos, pero pudimos resolverlo satisfactoriamente. Gracias al proyecto reforcé muchos de los conocimientos que se vieron en la teoría y con las investigaciones hechas para funcionalidades que no conocía me fue posible entender con mayor profundidad sobre las bases de datos.

Brito Serrano Miguel Ángel

A la conclusión de este proyecto me queda claro que el trabajo en equipo bien hecho es importante, se aprendió mucho de este proyecto, conocimientos que tal vez no me quedaron muy claros durante la clase y también se aprendió en varios aspectos, uno de ellos fue la creación de la interfaz gráfica, varios de los integrantes del proyectos aportaron su plus para que este proyecto quedará excelente, se uso todo lo visto pero de una manera un poco más enfocada a un reto de una empresa, obviamente también este proyecto me dio a saber mis debilidades, lo que tengo que hacer mejor y cómo hacerlo, se aplicó la creación de diseño de la base de datos, también el crear esta desde 0 y funcional, la inserción de información a esta, la aplicación de aplicaciones colaborativas para poder realizar con éxito el proyecto, claramente hubo obstáculos pero se superaron con ayuda de todos los integrantes, otra cosa nueva que aprendí o me quedó claro fue la implementación de los triggers, ya que pude ver una funcionalidad clara y como fue hecho, al parecer se cumple todos los requerimientos de nuestra base de datos con éxito, me quedó claro muchos conceptos de este proyecto y varios conocimientos aplicados.

Marentes Degollado Ian Paul

Al concluir con este proyecto, revisamos y aplicamos conceptos que hemos visto durante el transcurso del semestre, tanto las clases de teoría y laboratorio. Considero que cumplimos con los objetivos planteados al inicio, diseñar el modelo y la creación de una base de datos a partir de los requerimientos solicitados, realizar el registro de información, lectura, actualizaciones y eliminaciones; así mismo aplicamos los conceptos de normalización y consultas especializadas. Posiblemente el principal obstáculo para llevar a cabo el proyecto fue la falta de tiempo para revisar adecuadamente cada tema; sin embargo, logramos finalizarlo e inclusive conectar dicha base de datos a una aplicación que desarrollamos.

Meza Vega Hugo Adrián

A mi criterio, fue un proyecto del cual sacamos mucho provecho. Pusimos en práctica todo lo aprendido a lo largo de este semestre. Desde el análisis de requerimientos para la construcción de una base de datos, hasta la implementación y puesta en marcha de una base de datos en el sistema PostgreSQL. Además, no solo utilizamos conocimientos adquiridos en esta asignatura, también utilizamos conocimientos del área de la administración de proyectos de software, pues como equipo de trabajo tuvimos que organizar tiempos, delegar tareas y responsabilidades para cumplir con un tiempo establecido para el desarrollo de este proyecto.

Fue un gran reto el desarrollar este proyecto, pues implicó mucho esfuerzo de nuestra parte e hicimos uso de buenas prácticas para el desarrollo del proyecto. Realizamos la documentación apropiada y llevamos a cabo el diseño del diagrama entidad-relación utilizando herramientas CASE, utilizamos herramientas colaborativas para desarrollar el modelo relacional y complementamos la parte de diseño con la normalización. Una vez normalizada, pusimos en práctica la parte del lenguaje de definición de datos para la creación de la base de datos. Posteriormente la parte de la manipulación de datos la complementamos junto con la creación de funciones y triggers para cumplir con los requerimientos base del proyecto e incluso logramos conectar la base de datos con una aplicación programada con el lenguaje de programación JAVA. Pudimos proponer nuestra solución a la necesidad de desarrollar una base de datos incluido un programa de administración y compra para una cadena de papelerías.

REFERENCIAS

- Link de nuestro diagrama del modelo relacional en DBDesigner. <https://dbdesigner.page.link/eJEGwTZHwzn8PKY18>
- Creación de funciones. PostgreSQL 9.1.24 Documentation. <https://www.postgresql.org/docs/9.1/sql-createfunction.html> Consultado el 5 de agosto de 2021.
- Creación de triggers. PostgreSQL 9.1.24 Documentation. <https://www.postgresql.org/docs/9.1/sql-createtrigger.html> Consultado el 10 de agosto de 2021.
- Función para concatenar. String Functions and Operators. PostgreSQL 9.1.24 Documentation. <https://www.postgresql.org/docs/9.1/functions-string.html> Consultado el 5 de agosto de 2021.
- Funciones para date. Date/Time Function and Operators. PostgreSQL 9.1.24 Documentation. <https://www.postgresql.org/docs/9.1/functions-datetime.html> Consultado el 11 de agosto de 2021.
- Vistas en una base de datos. Documentación de SQL. Microsoft. <https://docs.microsoft.com/es-es/sql/relational-databases/views/views?view=sql-server-ver15> Consultado el 10 de agosto de 2021.
- Conectar JAVA con una base de datos PostgreSQL. ORIGEN(DATA); Uni bit más en este océano de la información. <https://origendata.com/2017/06/22/conectar-java-con-una-base-de-datos-postgresql/> Consultado el 28 de junio de 2021.
- Apuntes de clase de bases de datos. Diapositivas del profesor: ING. Fernando Arreola Franco. Universidad Nacional Autónoma de México, Ciudad de México, 2021.
- DB DESIGNER. Herramienta de diseño y modelado de esquemas de base de datos en línea. <https://www.dbdesigner.net>