

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

BASES DE DATOS 2022-1

PROFESOR ING. FERNANDO ARREOLA FRANCO

GRUPO: 01

11/12/2021



PROYECTO FINAL

Equipo BD Developers:

López Sixtos Axel Aurelio

Pastenes Pérez Jorge Luis

Rodríguez Dávalos Carolina

Sánchez Rojo Juan Pablo

Trejo Nava Ana Maritza

1. Objetivo

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

2. Introducción

Las bases de datos son un herramienta de gran relevancia hoy en día, es por eso que aprender a diseñarlas y desarrollarlas es parte de la formación de un ingeniero en computación, debido a que presentan grandes ventajas como agrupar y almacenar grandes cantidades de datos, administrar, planear, controlar y tomar decisiones, es por eso que en nuestro proyecto se busca simular un problema semejante a uno de la vida laboral donde se almacenaran datos de una empresa de papelerías y nosotros aplicaremos nuestros conocimientos adquiridos a lo largo del curso de bases de datos tanto teórico como en el laboratorio para encontrar una solución óptima.

El objetivo principal de nuestro proyecto es que dados los requerimientos debemos analizar cuidadosamente cada uno de ellos para diseñar e implementar una solución a nuestro problema aplicando los conceptos aprendidos en bases de datos principalmente y en ingeniería de software, así como otras materias, de tal forma que presentemos una aplicación funcional y que cumpla con lo establecido.

Durante el desarrollo de nuestro proyecto después de analizar lo que se pide en la parte uno y dos procederemos con los diseños conceptuales (modelo entidad-relación y modelo relacional), pondremos especial atención al modelo entidad relación ya que será la base de todo nuestro diseño. Después de desarrollar la base de datos crearemos una interfaz la cual permita al usuario manipular la base de datos sin necesidad de tener que entrar al código directamente, es por eso que será requerido un backend y un frontend de forma que podamos establecer la conexión con nuestra base de datos.

3. Plan de trabajo

Para las actividades que se realizaran a lo largo del desarrollo del curso se hicieron varias reuniones, tanto para informar de avances como para consultar sobre acciones de trabajo futuras. Dadas las complicaciones del semestre, el proyecto lo iniciamos el 29 de diciembre. Dada la necesidad de realizar el proyecto en tiempo express, decidimos reunirnos mediante zoom para primero ponernos de acuerdo en cómo organizarnos en cuestión de tiempo y como esperamos que pueda desarrollarse el proyecto basado en nuestras expectativas y el tiempo que tenemos disponible tanto para esta materia como otras, es por eso que decidimos representar nuestra planeación mediante un diagrama de Gantt en el cual podamos ver reflejado las tareas que se han asignado a cada uno o bien a todos en conjunto. De esta forma hemos dividido las actividades

que consideramos se pueden realizar y en las reuniones comentar los avances que hemos tenido cada uno y los problemas a los que nos enfrentamos, de esta forma podemos ayudarnos y complementar con nuevas ideas y propuestas de solución.

[illegible]

4. Diseño

4.1. Modelo Conceptual

Recolección y análisis de los requerimientos

Entidades

PREVEEDOR

Atributos: id_proveedor, teléfono, razón_social, domicilio (estado, c.p, colonia, calle, número), nombre (pila, ap_paterno, ap_materno).

INVENTARIO

Atributos: id_inventario, costo_compra, fecha_compra, stock, utilidad, código_barras.

ARTICULO

Atributos: id_articulo, precio_venta, descripción, marca.

CATEGORÍA

Atributos: id_categoria, tipo.

VENTA

Atributos: num_venta, fecha_venta, monto_total.

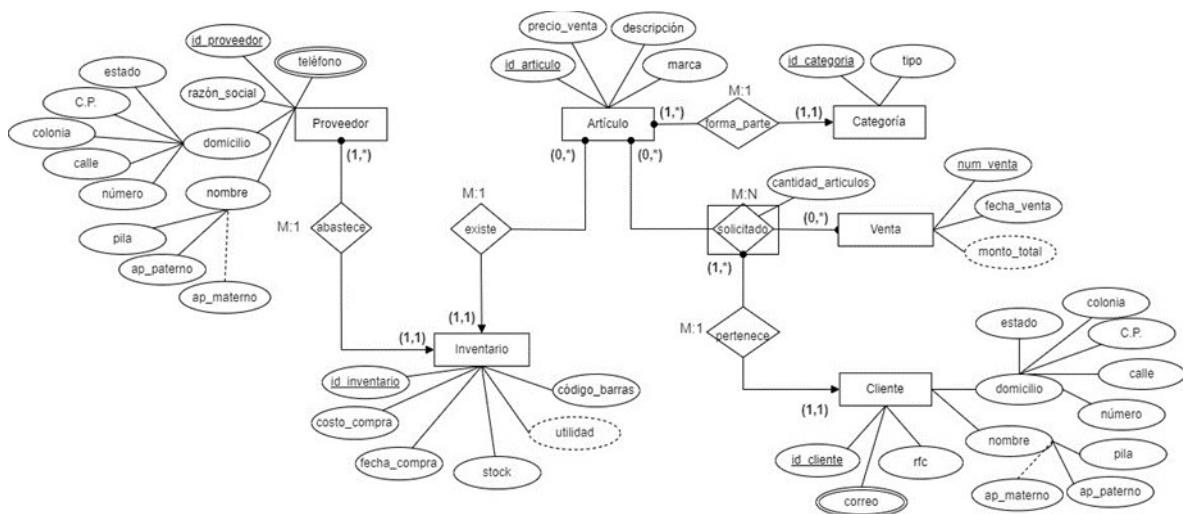
CLIENTE

Atributos: id_cliente, correo, rfc, nombre (pila, ap_paterno, ap_materno, ap_materno), domicilio(estado, c.p, co

Relaciones

1. Un proveedor surte un inventario y un inventario es surtido por varios proveedores
2. Un artículo puede o no existir en un inventario y en un inventario existen varios artículos.
3. Un artículo pertenece a una categoría y una categoría tiene varios artículos
4. Un artículo puede o no ser solicitado por una o muchas ventas. Una venta puede o no solicitar uno o muchos artículos
5. Un cliente pertenece a la venta de un artículo, la venta de uno o muchos artículos pertenecen a la compra de un cliente.

RelacionesDiagrama Entidad-Relación 1



Diseño de tablas

PREVEEDOR

PROVEEDOR = id_proveedor (PK) numeric(10,0), id_inventario (FK) numeric(10,0), razón_social varchar(30), nombre varchar(40), ap_paterno varchar(40), ap_materno varchar(40), estado varchar(20), c.p numeric(5,0), colonia varchar(30), calle varchar(40), numero numeric(5,0)

TEL_PROVERDOR

TEL_PROVEEDOR = id_tel_proveedor (PK) numeric(10,0), id_proveedor (FK) numeric(10,0), telefono numeric(10,0)

INVENTARIO

INVENTARIO = id_inventario (PK) numeric(10,0), cod_barra (U) numeric(13,0), costo_compra money(10,0), fecha_compra time/datatime, stock numeric(5,0), utilidad_c money(10,4)

ARTICULO

ARTICULO = id_articulo (PK) numeric(18,0), id_categoria (FK) integer, id_inventario(FK) numeric(10,0), precio_venta money(10,4), descripción varchar(30), marca varchar(20)

CATEGORÍA

id_categoria (PK) integer, tipo varchar(20).

VENTA

VENTA = num_venta (PK) varchar(8), fecha_venta time/datetime, monto_total money(10,4).

CLIENTE

CLIENTE = id_cliente (PK) numeric(10,0), rfc (U) varchar(9), nombre varchar(40), ap_paterno varchar(40), ap_materno varchar(40), estado varchar(20), c.p numeric(5,0), colonia varchar(40), calle varchar(40), numero numeric(5,0).

SOLICITUD_VENTA

SOLICITUD_VENTA = id_sol_venta (PK) numeric(10,0), id_cliente (FK) numeric(10,0), [id_articulo (FK) numeric(18,0), num_venta (FK) varchar(8)] (UK)

CLIENTE_CORREO

CLIENTE_CORREO = id_cliente_correo (PK) numeric(10,0), correo varchar(30), id_cliente (FK) numeric(10,0)

Diagrama Relacional

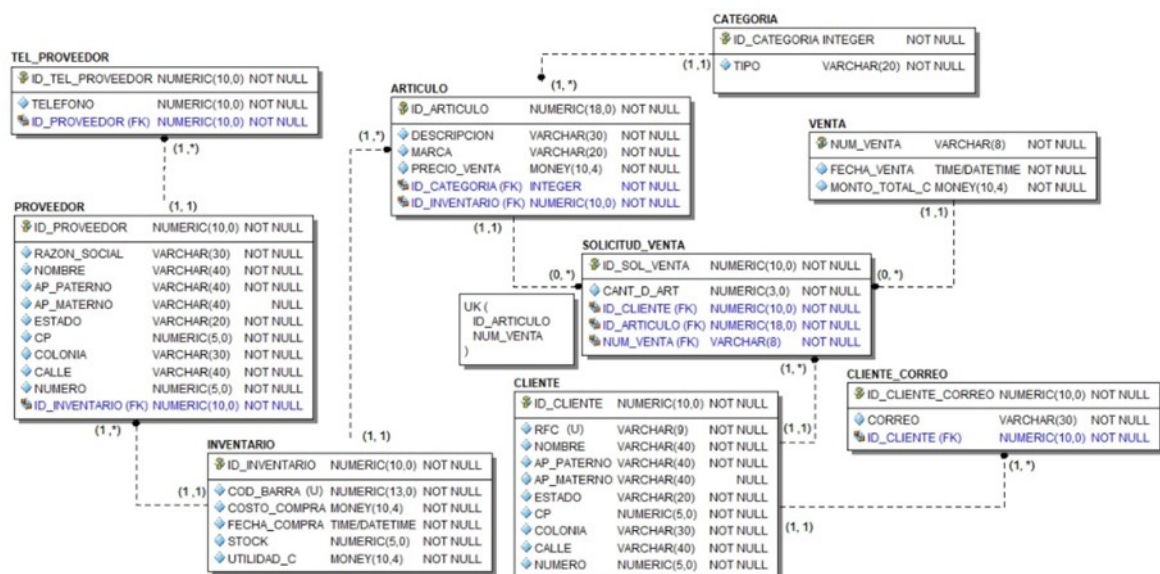


Diagrama Entidad-Relacion

Entidades

PREVEEDOR

Atributos: id_proveedor, teléfono, razón_social, domicilio (estado, c.p, colonia, calle, número), nombre (pila, ap_paterno, ap_materno).

ARTICULO

Atributos: id_articulo, precio_venta, descripción, marca.

CATEGORÍA

Atributos: id_categoria, tipo.

VENTA

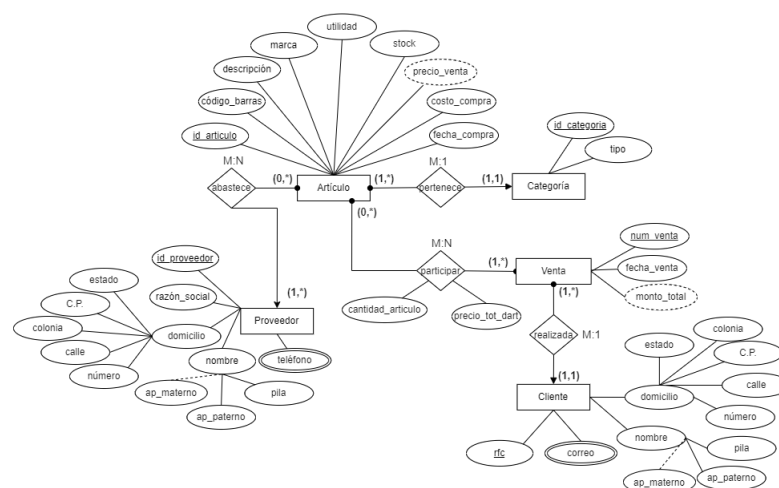
Atributos: num_venta, fecha_venta, monto_total.

CLIENTE

Atributos: id_cliente, correo, rfc, nombre (pila, ap_paterno, ap_materno, ap_materno), domicilio (estado, c.p, colonia, calle, número)..

Relaciones

1. Un proveedor surte un inventario y un inventario es surtido por varios proveedores.
2. Un artículo puede o no existir en un inventario y en un inventario existen varios artículos.
3. Un artículo pertenece a una categoría y una categoría tiene varios artículos.
4. Un artículo puede o no participar en una o muchas ventas. En una venta participan uno o muchos artículos.
5. Una venta es realizada por la compra de un cliente y un cliente puede hacer una o muchas compras asociadas a una venta.

RelacionesDiagrama Entidad-Relación 2

Diseño de tablas**PREVEEDOR**

PROVEDOR = id_proveedor (PK) numeric(10,0), razón_social (UK) varchar(30), nombre varchar(40), ap_paterno varchar(40), ap_materno varchar(40), estado varchar(20), c.p numeric(5,0), colonia varchar(30), calle varchar(40), numero numeric(5,0)

TEL_PROVEEDOR

TEL_PROVEEDOR = id_tel_proveedor (PK) numeric(10,0), id_proveedor (FK) numeric(10,0), telefono numeric(10,0)

ARTICULO

ARTICULO = id_articulo (PK) numeric(18,0), id_categoria (FK) integer, id_proveedor (FK) numeric(10,0), cod_barras varchar(13), descripcion varchar(30), marca varchar(20), stock numeric(40,0), utilidad money(10,4), costo_compra money(40,0), precio_venta_c money(40,0), fecha_compra date)

CATEGORÍA

id_categoria (PK) integer, tipo varchar(20).

VENTA

VENTA = num_venta (PK) varchar(8), id_cliente (FK) numeric(10,0), fecha_venta time/datatime, monto_total_c money(10,4)

PARTICIPA

PARTICIPA = id_articulo (FK) numeric(18,0), num_venta (FK) varchar(8), cantidad_articulo numeric(10,0), precio_total_dart money(40,0)

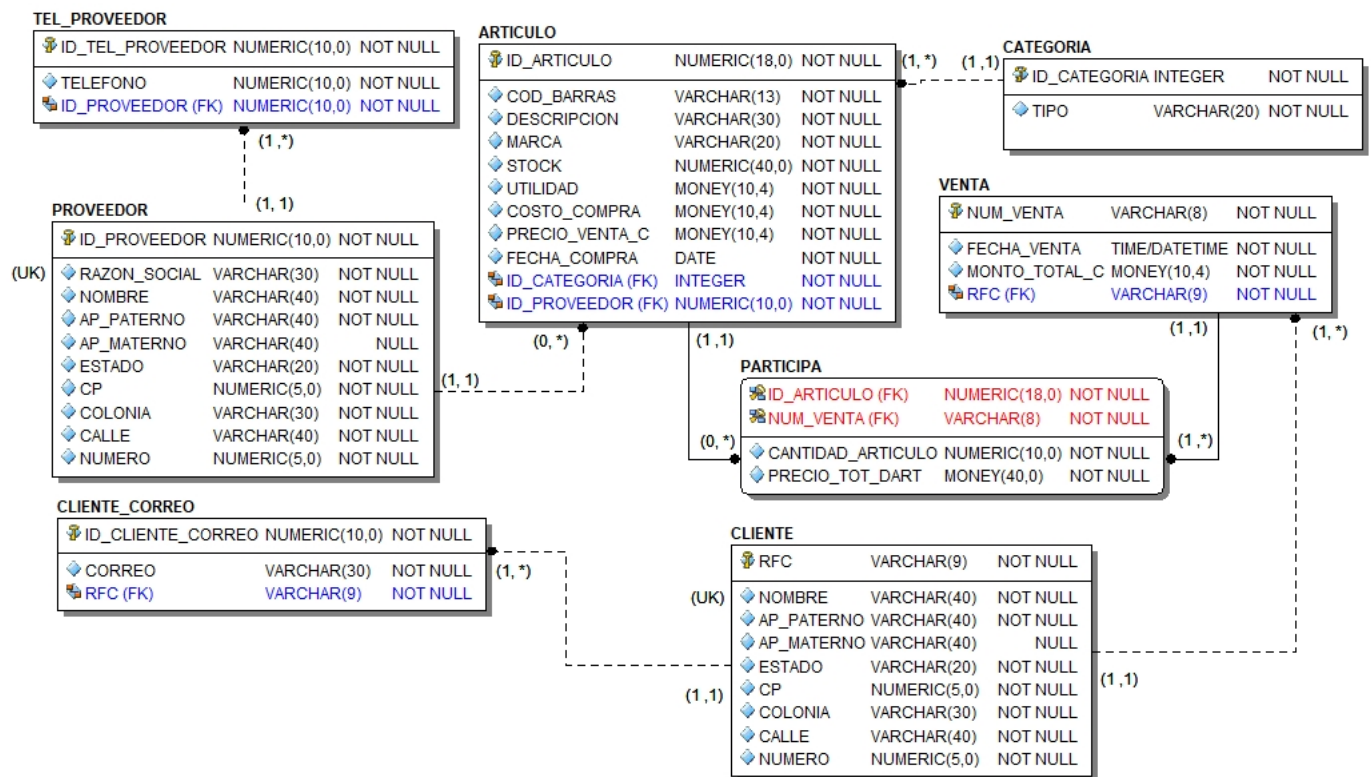
CLIENTE

CLIENTE = id_cliente (PK) numeric(10,0), rfc (UK) varchar(9), nombre varchar(40), ap_paterno varchar(40), ap_materno varchar(40), ap_materno varchar(40), estado varchar(20), c.p numeric(5,0), colonia varchar(40), calle varchar(40), numero numeric(5,0).

CLIENTE_CORREO

CLIENTE_CORREO = id_cliente_correo (PK) numeric(10,0), correo varchar(30), id_cliente (FK) numeric(10,0)

Diagrama Relacional 2



Nota: Para la implementación en el sistema manejador de bases de datos se utilizó el análisis 2; que corresponde al diagrama entidad-relación 2 y al diagrama relacional 2.

Implementación en PostgreSQL

Funciones de la base de datos

1. Entrada: Información almacenada en tablas.
2. Procesamiento: Cálculos con los datos.
3. Almacenamiento: Guardar información
4. Salida de información: Consultas

Manejador de Bases de Datos

Es un conjunto de módulos preprogramados que nos van a permitir manipular una base de datos. Se utilizó PostgreSQL como sistema gestor de bases de datos. Un sistema gestor de base de datos es un conjunto de aplicaciones que permiten interactuar con el manejador de base de

datos y la base de datos.

1. Seguridad
2. Concurrencia
3. Integridad

Arquitectura de la base de datos

- Permite separar las aplicaciones del usuario y la base de datos física.
- Garantiza la interacción entre sus niveles.

Se basa en una arquitectura de tres niveles:

1. Usuarios: Nivel Externo.
2. Diseño: Nivel conceptual.
3. Almacenamiento: Nivel interno.

Nivel externo: Describe la estructura física del almacenamiento de la base de datos. El esquema interno emplea un modelo físico de datos y describe todos sus detalles para su almacenamiento.

Nivel conceptual: Describe la estructura de la base de datos para una comunidad de usuarios. Oculta los detalles de las estructura físicas de almacenamiento y se concentra en describir entidades, tipos de datos, relaciones y restricciones.

Nivel externo: Es también conocido como las vistas del usuario. Cada esquema externo describe la parte de la base de datos, que interesa a determinados usuarios y oculta a ese mismo grupo el resto de la base de datos.

Independencia de los datos

Consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que sirven de ella.

Existe:

- Independencia Física
- Independencia Lógica

Independencia Física: Cambio del esquema interno sin necesidad de cambiar el esquema conceptual o los esquemas externos.

Independencia Lógica: Cambio del esquema conceptual sin cambiar las vistas externas o las aplicaciones.

Independencia de los datos

Permite especificar los elementos de datos que la integran, su estructura y las relaciones que existen entre ellas. Además permite la creación de procedimientos entre tablas (consultas, estructuras y relaciones, funciones y disparadores).

Ejemplos:

- Create
- Drop

Lenguaje de manipulación de datos (DML)

Es un lenguaje que proporciona un conjunto de operadores para permitir las manipulaciones básicas de los datos contenidos en la base de datos.

Ejemplos:

- Select
- Insert
- Update
- Delete

Lenguaje de control de datos (DCL)

Permite al administrador controlar el acceso a los datos contenidos en la base de datos.

Ejemplos

- Revoke
- Grant

Lenguaje Transaccional

Permite finalizar una transacción y confirmar los datos.

Ejemplos

- Commit
- Rollback

Lenguaje de definición de datos (DDL)

```
proyecto=# CREATE TABLE categoria
proyecto=# (
proyecto(#      id_categoria serial NOT NULL,
proyecto(#      tipo character varying(20) NOT NULL,
proyecto(#      constraint categoria_pk PRIMARY KEY (id_categoria)
proyecto(# );
CREATE TABLE
```

```
proyecto=# CREATE TABLE articulo
proyecto=# (
proyecto(#      id_articulo serial NOT NULL,
proyecto(#      cod_barras character varying(13) NOT NULL,
proyecto(#      descripcion character varying(30) NOT NULL,
proyecto(#      marca character varying(20) NOT NULL,
proyecto(#      stock int NOT NULL,
proyecto(#      utilidad numeric(10,2) NOT NULL,
proyecto(#      costo_compra numeric(10,2) NOT NULL,
proyecto(#      precio_venta numeric(10,2) NOT NULL,
proyecto(#      fecha_compra date NOT NULL,
proyecto(#      id_categoria bigint NOT NULL,
proyecto(#      id_proveedor bigint NOT NULL,
proyecto(#      constraint articulo_pk PRIMARY KEY (id_articulo),
proyecto(#      constraint art_cod_barras_uk UNIQUE (cod_barras),
proyecto(#      constraint art_id_categoria_fk FOREIGN KEY (id_categoria)
proyecto(#      references categoria(id_categoria) ON DELETE CASCADE ON UPDATE CASCADE,
proyecto(#      constraint art_id_proveedor_fk FOREIGN KEY (id_proveedor)
proyecto(#      references proveedor(id_proveedor) ON DELETE CASCADE ON UPDATE CASCADE
proyecto(# );
CREATE TABLE
```

```
proyecto=# CREATE SEQUENCE venta_id_venta_seq
proyecto=# START WITH 1
proyecto=# INCREMENT BY 1;
CREATE SEQUENCE
```

```
proyecto=# CREATE TABLE venta
proyecto=# (
proyecto(#      num_venta text NOT NULL DEFAULT 'VENT-'||nextval('venta_id_venta_seq'::regclass)::text,
proyecto(#      fecha_venta date NOT NULL,
proyecto(#      monto_total numeric(10,2) NOT NULL,
proyecto(#      constraint venta_pk PRIMARY KEY (num_venta)
proyecto(# );
CREATE TABLE
```

```

proyecto=# CREATE TABLE participa
proyecto=# (
proyecto=#     id_articulo bigint NOT NULL,
proyecto=#     num_venta text NOT NULL,
proyecto=#     cantidad_articulo bigint NOT NULL,
proyecto=#     precio_tot_dart numeric(10,2) NOT NULL,
proyecto=#     constraint participa_id_articulo_fk FOREIGN KEY (id_articulo)
proyecto=#     references articulo(id_articulo) ON DELETE CASCADE ON UPDATE CASCADE,
proyecto=#     constraint participa_num_venta_fk FOREIGN KEY (num_venta)
proyecto=#     references venta(num_venta) ON DELETE CASCADE ON UPDATE CASCADE,
proyecto=#     constraint participa_av_pk PRIMARY KEY (id_articulo,num_venta)
proyecto=# );
CREATE TABLE







```



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

```

1  --Consulta utilidad por código de barras
2  CREATE FUNCTION consulta_utilidad(codigo_barras varchar(13))
3  RETURNS DECIMAL AS $consulta_utilidad$
4  DECLARE carga_utilidad decimal = (SELECT utilidad_c FROM inventario WHERE cod_barras = codigo_barras);
5  BEGIN
6      RETURN carga_utilidad;
7  END;
8  $consulta_utilidad$ LANGUAGE plpgsql;co

```

Data Output		Explain	Messages	Notifications		
	 id_inventario [PK] integer	 cod_barras character varying (13)	 costo_compra money	 fecha_compra date	 stock integer	 utilidad_c money
1		1001	8759463210258	\$ 70.00	2020-12-31	5 \$ 8.90
2		1002	8967412587463	\$ 30.00	2020-12-31	20 \$ 40.00
3		1003	8726354789515	\$ 11.90	2020-12-31	15 \$ 6.60
4		1004	8793542641569	\$ 26.80	2020-12-31	32 \$ 7.70
5		1005	8968357421258	\$ 12.50	2020-12-31	64 \$ 8.50
6		1006	8965426841223	\$ 15.00	2020-12-31	4 \$ 4.00
7		1007	8974125632005	\$ 350.00	2020-12-31	2 \$ 46.00
8		1008	8852364352025	\$ 12.00	2020-12-31	3 \$ 6.00
9		1009	8888881151000	\$ 1.00	2020-12-31	21 \$ 1.50

Query Editor		Query History		
1 SELECT consulta_utilidad ('8793542641569');				
Data Output		Explain	Messages	Notifications
	consulta_utilidad numeric			
1		7.70		

- 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.

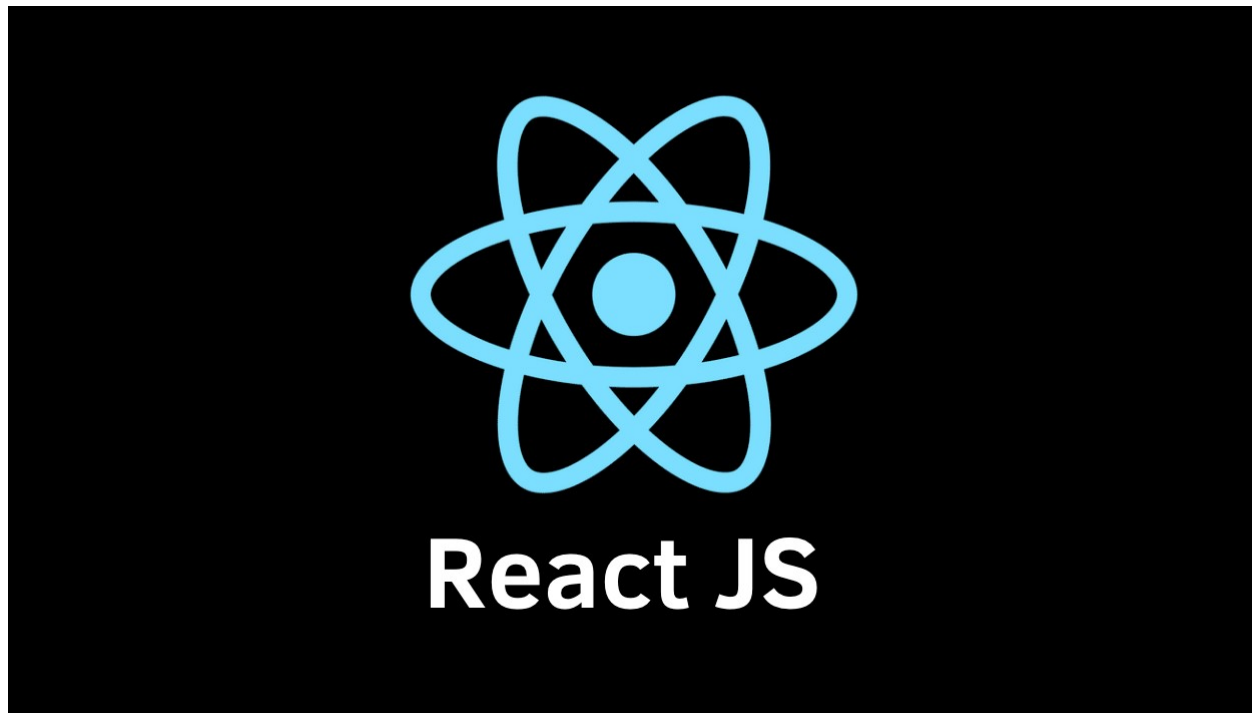
```
1  --Decremento por ventas
2  CREATE OR REPLACE FUNCTION decremento_stock() RETURNS TRIGGER AS $$
3      DECLARE stock_disponible int;
4      DECLARE var_aux int;
5      BEGIN
6          stock_disponible =(SELECT stock FROM inventario WHERE id_inventario=new.id_inventario);
7          IF ( stock_disponible > 0) THEN
8              IF (new.cantidad_articulo<= stock_disponible) THEN
9                  var_aux =(SELECT stock FROM inventario WHERE id_inventario=new.id_inventario) - new.cantidad_articulo;
10                 UPDATE producto SET stock= var_aux WHERE id_inventario=new.id_inventario;
11                 stock_disponible =(SELECT stock FROM inventario WHERE id_inventario=new.id_inventario);
12                 IF stock_disponible <=3 THEN
13                     RAISE NOTICE 'Quedan % productos del producto con id %', stock_disponible ,new.id_inventario;
14                 END IF;
15             ELSE
16                 RAISE NOTICE 'No hay stock suficiente del producto con id %. Se ha cancelado la transaccion', new.id_inventario;
17             END IF;
18         ELSE
19             RAISE NOTICE 'No hay stock del producto con id %. Se ha cancelado la transaccion. ', new.id_inventario;
20             RETURN NULL;
21         END IF;
22         RETURN new;
23     END
24 $$ LANGUAGE plpgsql;
```

Query Editor Query History

```
1  --Trigger que decrementa el stock
2  CREATE TRIGGER decremento_stock BEFORE INSERT ON venta
3  FOR EACH ROW EXECUTE FUNCTION decremento_stock();
4
```

Conexión a la base de datos

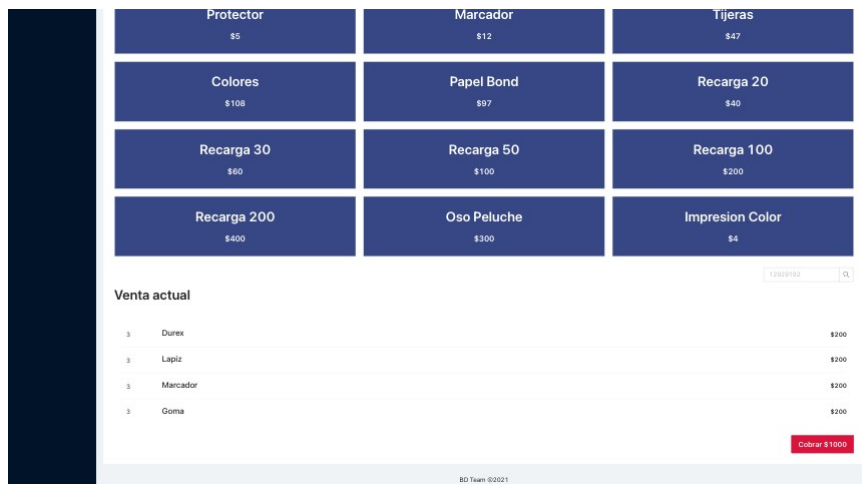
Para poder interactuar con la base de datos se utilizó flask que nos permitió insertar y hacer consultas de manera muy rápida. La arquitectura que se ocupó es cliente – servidor. Las tecnologías usadas en el Frontend son React, Javascript y CSS.



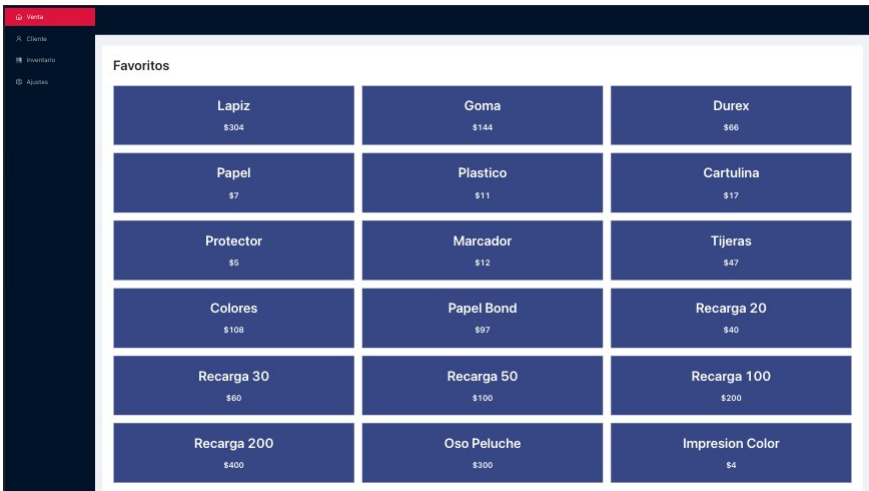
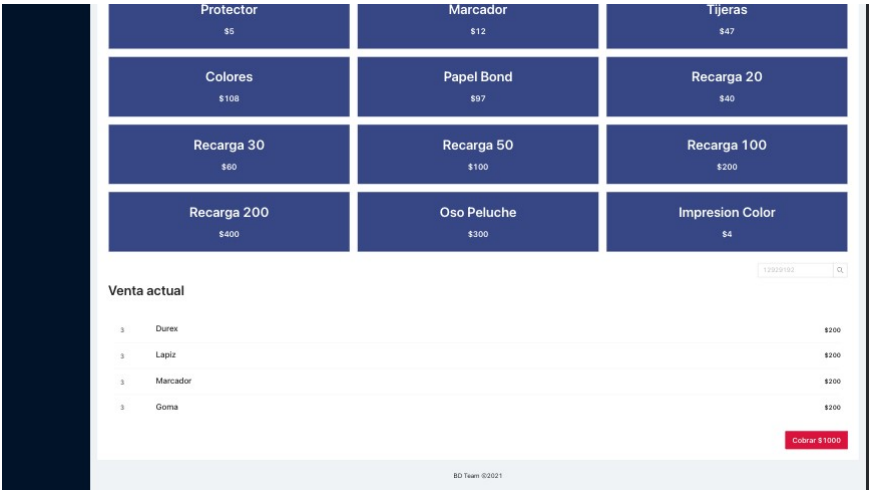
La conexión se realizó de la siguiente manera, para ello se uso Flask Alchemy.

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://postgres:password@localhost/dbv4'  
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False  
app.secret_key = 'secret string'  
db = SQLAlchemy(app)
```

La conexión se realizó de la siguiente manera, para ello se uso Flask Alchemy.



Imágenes de la aplicación(es responsiva)



Venta

Cliente

Inventario

Ajustes

Cliente

Ingresar los datos siguientes

* Nombre:

*Apellido P.:

Apellido M.:

* RFC:

* Calle:

* Colonia:

* Estado:

* Código P.:

* Número:

* Correo:

Registrar

BC Team ©2021

Cliente

Ingresar los datos siguientes

* Nombre

*Apellido P.

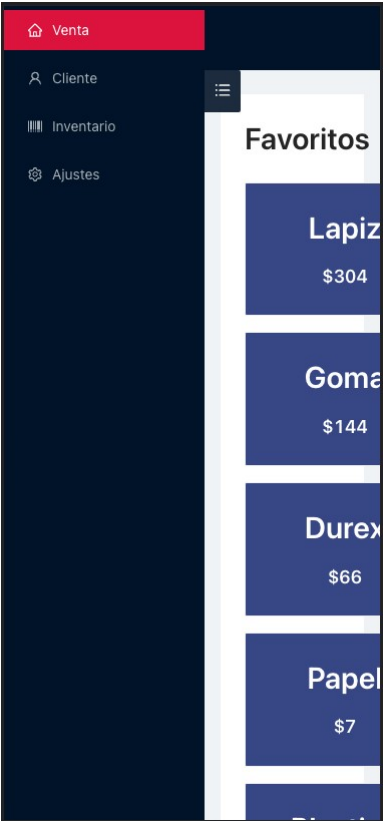
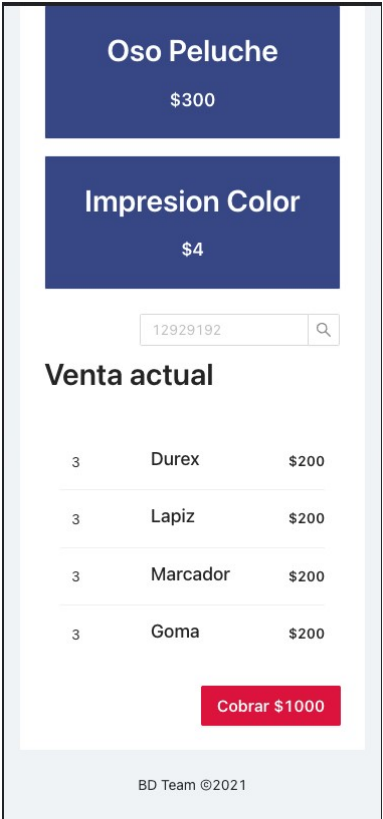
Apellido M.

* RFC

* Calle

* Colonia

* Estado





Conclusiones)

López Sixtos Axel Aurelio)

Se comprendió que el nivel conceptual describe la estructura de la base de datos para una comunidad de usuarios. Oculta los detalles de las estructuras físicas de almacenamiento y se concentra en describir entidades, tipos de datos, relaciones y restricciones y que el nivel externo es también son las vistas del usuario. Cada esquema externo describe la parte de la base de datos, que interesa a determinados usuarios y oculta a ese mismo grupo el resto de la base de datos. El mayor problema de la base de datos consistió en modificar n cantidades el diagrama relacional para poder realizar los procedimientos de manera correcta. 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. Fue muy difícil integrar el modelo lógico con el modelo físico. Gracias al proyecto aprendí a trabajar en equipo.

Pastenes Pérez Jorge Luis)

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 quedo 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 quedo claro muchos conceptos de este proyecto y varios conocimientos aplicados.

Rodríguez Dávalos Carolina)

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.

Sánchez Rojo Juan Pablo)

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.

Trejo Nava Ana Maritza)

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.