Universidad Nacional Autónoma de México

Facultad de Ingeniería

Proyecto Final

Asignatura: "Bases de Datos"
Grupo: 1
Profesor: Ing.Fernando Arreola

Integrantes:
Cortés González Zaira Yaritsi
Reyes Alonso Katherine
Rosales Velázquez Victor Daniel
Sánchez Maldonado Mario Alberto

Entrega: 25 de enero de 2021.

Índice de contenido

1 Introducción **2**

2 Plan de trabajo **3**
2.1 Cronograma 4

3 Diseño **6**
3.1 Diseño conceptual 6
3.2 Diseño lógico 7
3.3 Diseño fisico 7

4 Implementación **12**
4.1 Codificación 12

5 Presentación **17**

6 Conclusiones **21**

7 Bibliografía **22**

1 Introducción

El presente documento se redacta con carácter de Proyecto Final, donde se propone un sistema informático de base de datos como solución para almacenar la información de una cadena de papelerías.

El objetivo principal es cumplir con los requerimientos planteados mediante las herramientas y conceptos que nos ofreció el curso Bases de Datos, como son la seguridad de la información, el correcto diseño tanto físico como lógico, así como la aplicación de conceptos para la eficiencia de las operaciones de la base de datos.

En las secciones que se presentan a continuación se recopila el plan de trabajo que siguió el equipo para cada fase, donde se explicará brevemente las actividades que cubrió cada integrante y el tiempo que llevó completar cada actividad. Posteriormente se encuentra el diseño propuesto para la base de datos que consta de un modelo entidad-relación que da lugar a un modelo relacional para representar las tablas con los respectivos atributos que creamos en la implementación de la solución, estas fueron creadas siguiendo los conceptos para satisfacer todos los principios de diseño y de seguridad de la información. En la siguiente sección se explicará el código utilizado para la creación completa de la base de datos, la inserción de información, la funcionalidad que se compone por funciones, vistas y triggers que consideramos pertinentes para cumplir con lo solicitado. Por último, se mostrará el sistema web empleado con la forma de conexión a la base de datos para la interfaz gráfica del sistema y las funciones que permiten al usuario operar los datos en una vista agradable. Cada integrante plasmará su conclusión en la parte final del documento.

2 Plan de trabajo

La primera actividad asignada al equipo fue acordar canales de comunicación, herramientas colaborativas y plataformas para llevar a cabo el trabajo y tener comunicación con todos los integrantes; optamos por utilizar Google Drive, Asana, Zoom y WhatssApp durante todo el proyecto.

Con el conocimiento adquirido durante el curso decidimos analizar el proyecto y una vez identificados los temas a abordar y los problemas por resolver acordamos construir individualmente una propuesta de cronograma del proyecto para tener una estructura concreta del plan de trabajo. Al tener nuestro cronograma definitivo optamos por delimitar nuestras aptitudes y nuestras carencias para poder atacar las actividades en las que un miembro fuera mas apto y aunque todos los miembros participamos activamente en el desarrollo del proyecto, algunos trabajaron en áreas más específicas.

Las actividades se distribuyeron de la siguiente manera:

- Zaira Cortés fue encargada de búsqueda de recursos, diseño y creación de modelos, apoyo en la construcción y en la funcionalidad de la base de datos.
- Katherine Reyes fue encargada del apoyo en la búsqueda de recursos, apoyo en la creación de modelos, desarrollo de documentación en LaTeX y coordinación de reuniones y ensayos.
- Victor Rosales fue encargado de asignación de actividades al equipo, creación de cronograma, implementación del diseño: desarrollo de base de datos y funcionalidad de ésta, así como pruebas al sistema.
- Mario Sánchez fue encargado de desarrollo de funcionalidad de base de datos, creación de arquitectura cliente-servidor y desarrollo de back-end, front-end del sistema web.

A continuación se muestra el cronograma del proyecto.

2.1 Cronograma

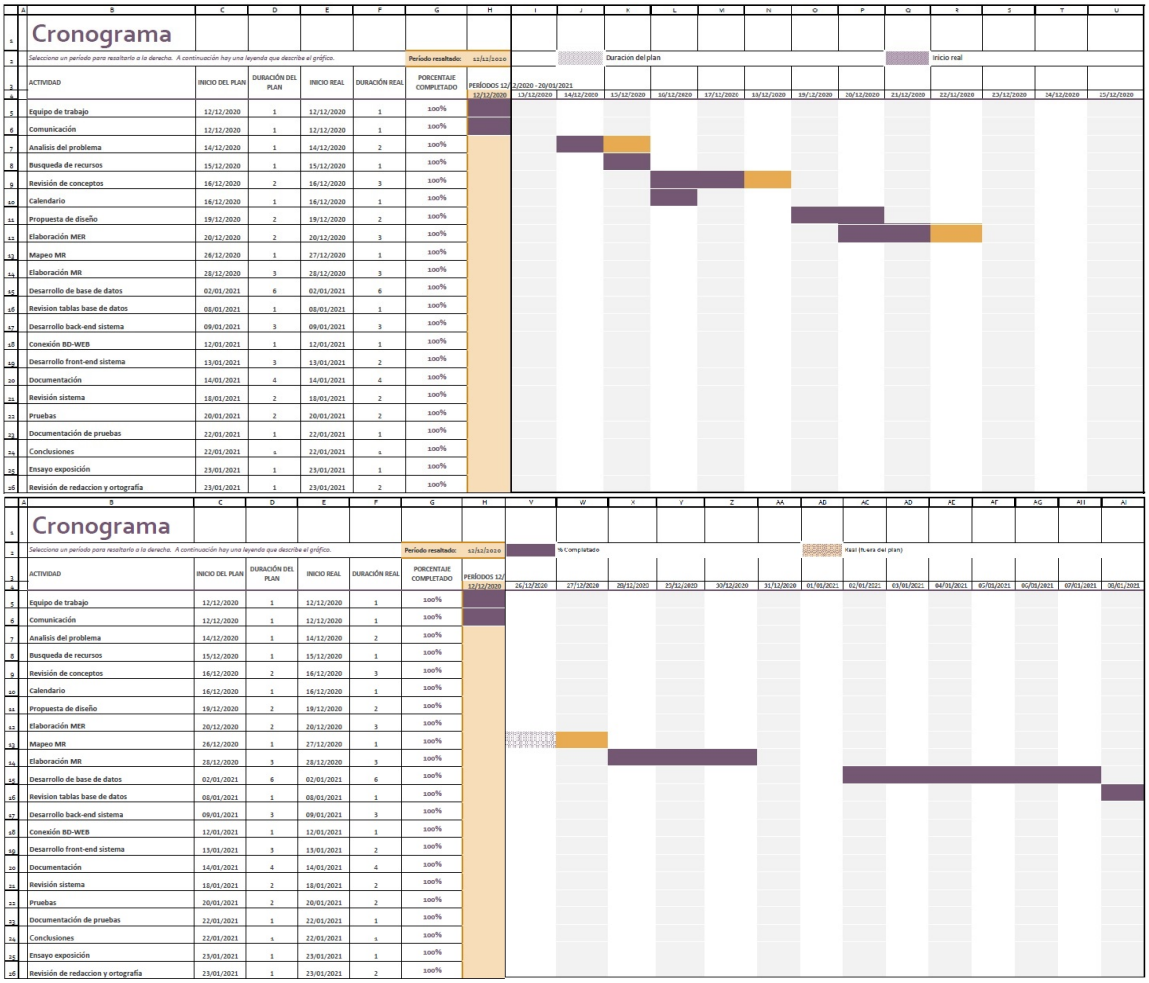
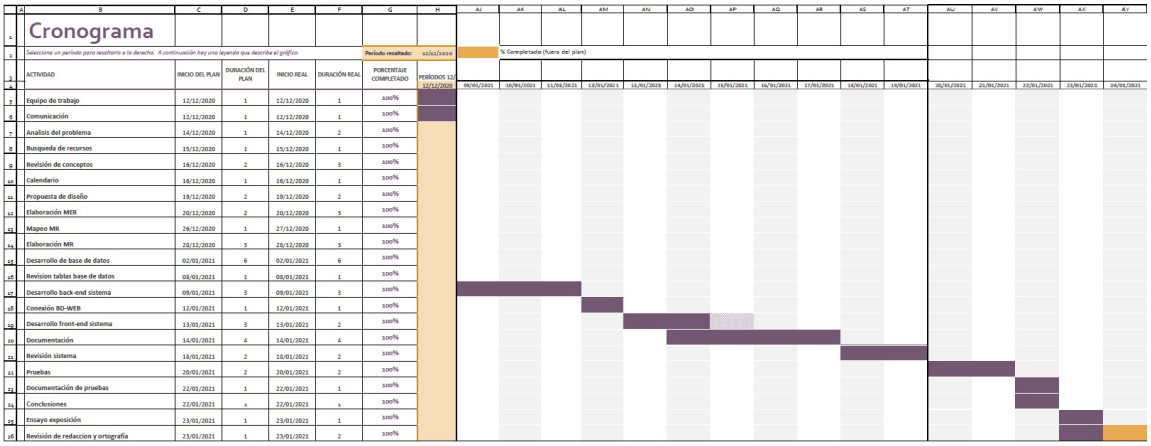


Figura 1: Plan de trabajo.



Cronograma

Seleccione un período para resaltarlo a la derecha. A continuación hay una leyenda que describe el gráfico.

Período resaltado: 12/12/2020

% Completado (fuera del plan)

ACTIVIDAD	INICIO DEL PLAN	DURACIÓN DEL PLAN	INICIO REAL	DURACIÓN REAL	PORCENTAJE COMPLETADO
Equipo de trabajo	12/12/2020	1	12/12/2020	1	100%
Comunicación	12/12/2020	1	12/12/2020	1	100%
Analisis del problema	14/12/2020	1	14/12/2020	2	100%
Busqueda de recursos	15/12/2020	1	15/12/2020	1	100%
Revisión de conceptos	16/12/2020	2	16/12/2020	3	100%
Calendario	16/12/2020	1	16/12/2020	1	100%
Propuesta de diseño	19/12/2020	2	19/12/2020	2	100%
Elaboración MER	20/12/2020	2	20/12/2020	3	100%
Mapeo MR	26/12/2020	1	27/12/2020	1	100%
Elaboración MR	28/12/2020	3	28/12/2020	3	100%
Desarrollo de base de datos	02/01/2021	6	02/01/2021	6	100%
Revision tablas base de datos	08/01/2021	1	08/01/2021	1	100%
Desarrollo back-end sistema	09/01/2021	3	09/01/2021	3	100%
Conexión BD-WEB	12/01/2021	1	12/01/2021	1	100%
Desarrollo front-end sistema	13/01/2021	3	13/01/2021	2	100%
Documentación	14/01/2021	4	14/01/2021	4	100%
Revisión sistema	18/01/2021	2	18/01/2021	2	100%
Pruebas	20/01/2021	2	20/01/2021	2	100%
Documentación de pruebas	22/01/2021	1	22/01/2021	1	100%
Conclusiones	22/01/2021	1	22/01/2021	1	100%
Ensayo exposición	23/01/2021	1	23/01/2021	1	100%
Revisión de redaccion y ortografía	23/01/2021	1	23/01/2021	2	100%

Figura 2: Plan de trabajo.

3 Diseño

Debido a que el diseño define la estructura de la base de datos se dedicó tiempo considerable en su análisis para obtener como resultado una buena administración de la información.

Una base de datos bien diseñada es imprescindible para garantizar la coherencia de la información, evitar datos redundantes, ejecutar consultas de manera eficiente y mejorar el rendimiento de la base de datos. En los siguientes apartados detallaremos lo que se realizó en cada etapa.

3.1 Diseño conceptual

Uno de los primeros pasos de la implementación de la solución fue crear nuestros modelos de datos, comenzando por la fase de diseño conceptual en donde después de analizar el problema pudimos identificar todas las entidades, así como sus atributos y las relaciones entre ellas, esto dio origen a obtener 6 entidades, cada una con los requisitos solicitados sobre los atributos. La siguiente figura ilustra lo antes descrito.

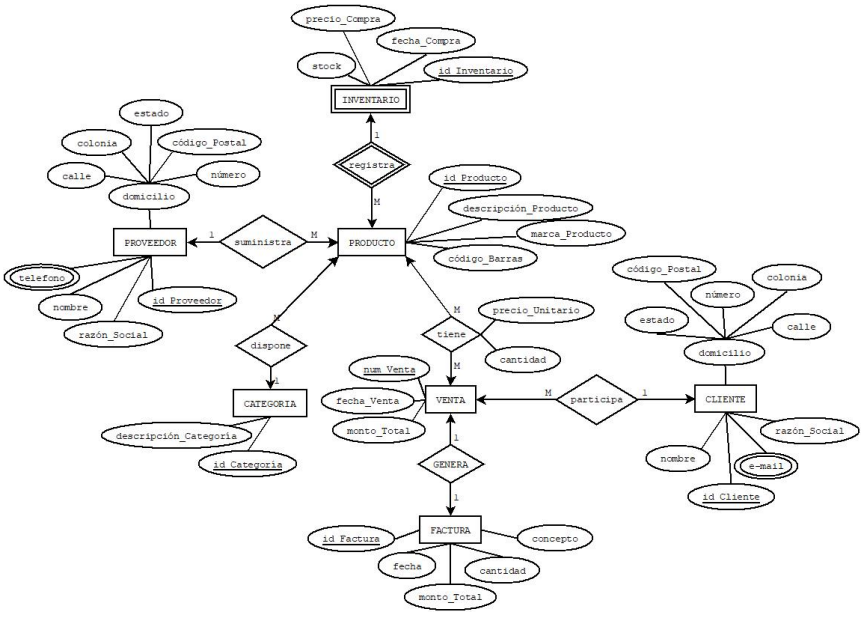


Figura 3: Modelo Entidad-Relación.

3.2 Diseño lógico

Una vez completando la fase del diseño conceptual podemos comenzar con la transformación de las entidades para poder construir nuestro Modelo Relacional siguiendo las reglas para la correcta transición de MER A MR. Cabe destacar que la creación del Modelo Relacional tiene la flexibilidad de realizar cambios conforme vayamos avanzando.

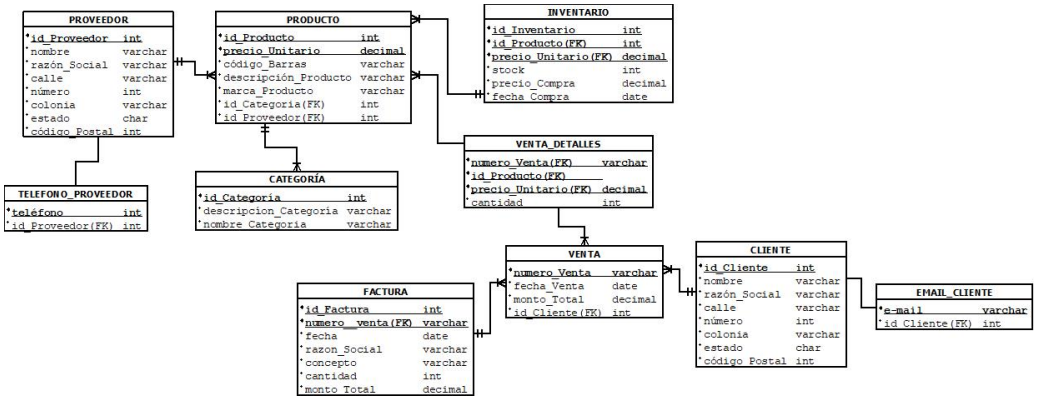


Figura 4: Modelo Relacional.

3.3 Diseño fisico

La siguiente etapa es la elaboración del diseño físico, basándonos en el modelo lógico creamos nuestra base de datos y las tablas que la conforman, aunque esta etapa radica sobre el diseño lógico puede existir variación al crear físicamente la base de datos.

A continuación se muestra la implementación en lenguaje estructurado de consulta (SQL por sus siglas en inglés).

Creación de la base de datos:

```
CREATE DATABASE "PROYECTO_PAPELERIA"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    CONNECTION LIMIT = -1;
```

Creación de tablas:

```
--TABLA 1
CREATE TABLE PROVEEDOR(
    id_proveedor INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    razon_social VARCHAR(50) NOT NULL,
    calle VARCHAR(25) NOT NULL,
    numero INT NOT NULL,
    colonia VARCHAR(25) NOT NULL,
    estado VARCHAR(20) NOT NULL,
    codigo_postal INT NOT NULL,
    CONSTRAINT PROVEEDOR_PK PRIMARY KEY(id_proveedor)
    );

--TABLA 2
CREATE TABLE TELEFONO_PROVEEDOR(
    telefono VARCHAR(15) NOT NULL,
    id_proveedor INT NOT NULL,
    CONSTRAINT TELEFONO_PROVEEDOR_PK PRIMARY KEY(telefono),
    CONSTRAINT PROVEEDORT_FK FOREIGN KEY(id_proveedor)
    REFERENCES PROVEEDOR(id_proveedor)
    );

--TABLA 3
CREATE TABLE CATEGORIA(
    id_categoria INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    nombre_categoria VARCHAR (50) NOT NULL,
    descripcion_categoria VARCHAR(50) NOT NULL,
    CONSTRAINT CATEGORIA_PK PRIMARY KEY (id_categoria)
```

```
    ) ;

--TABLA 4
CREATE TABLE PRODUCTO(
    id_producto INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    precio_unitario DECIMAL NOT NULL,
    codigo_barras VARCHAR(20) NOT NULL,
    descripcion_producto VARCHAR(50) NOT NULL,
    marca_producto VARCHAR(25) NOT NULL,
    id_categoria INT NOT NULL,
    id_proveedor INT NOT NULL,
    CONSTRAINT PRODUCTO_PK PRIMARY KEY(id_producto,
    precio_unitario),
    CONSTRAINT PROVEEDORP_FK FOREIGN KEY(id_proveedor)
    REFERENCES PROVEEDOR(id_proveedor),
    CONSTRAINT CATEGORIAP_FK FOREIGN KEY(id_categoria)
    REFERENCES CATEGORIA(id_categoria)
    );

--TABLA 5
CREATE TABLE INVENTARIO(
    id_inventario INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    id_producto INT NOT NULL,
    precio_unitario DECIMAL NOT NULL,
    stock INT NOT NULL,
    precio_compra DECIMAL NOT NULL,
    fecha_compra DATE NOT NULL,
    CONSTRAINT INVENTARIO_PK PRIMARY KEY(id_inventario),
    CONSTRAINT PRODUCTOI_FK FOREIGN KEY(id_producto,
    precio_unitario)
    REFERENCES PRODUCTO(id_producto,precio_unitario)
    );

--TABLA 6
CREATE TABLE CLIENTE(
    id_cliente INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    razon_social VARCHAR(50) NOT NULL,
```

```
    calle VARCHAR(25) NOT NULL,
    numero INT NOT NULL,
    colonia VARCHAR(25) NOT NULL,
    estado VARCHAR(20) NOT NULL,
    codigo_postal INT NOT NULL,
    CONSTRAINT CLIENTE_PK PRIMARY KEY (id_cliente)
    );

--TABLA 7
CREATE TABLE EMAIL_CLIENTE(
    email VARCHAR (80) NOT NULL,
    id_cliente INT NOT NULL,
    CONSTRAINT EMAIL_CLIENTE_PK PRIMARY KEY(email),
    CONSTRAINT CLIENTEEM_FK FOREIGN KEY(id_cliente)
    REFERENCES CLIENTE(id_cliente)
    );

--TABLA 8
CREATE TABLE VENTA(
    numero_venta VARCHAR(10) NOT NULL,
    id_cliente INT NOT NULL,
    fecha_venta DATE NOT NULL,
    CONSTRAINT VENTA_PK PRIMARY KEY(numero_venta),
    CONSTRAINT CLIENTEV_FK FOREIGN KEY(id_cliente)
    REFERENCES CLIENTE (id_cliente)
    );

--TABLA 9
CREATE TABLE VENTA_DETALLES(
    numero_venta VARCHAR(10) NOT NULL,
    id_producto INT NOT NULL,
    precio_unitario DECIMAL NOT NULL,
    cantidad INT NOT NULL,
    CONSTRAINT VENTAD_FK FOREIGN KEY (numero_venta)
    REFERENCES VENTA (numero_venta),
    CONSTRAINT PRODUCTOD_FK FOREIGN KEY
    (id_producto, precio_unitario)
    REFERENCES PRODUCTO (id_producto,
```

```
    precio_unitario)
    );

--TABLA 10
CREATE TABLE FACTURA(
    id_factura INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    numero_venta VARCHAR(10) NOT NULL,
    fecha DATE NOT NULL,
    razon_social VARCHAR(50) NOT NULL,
    concepto VARCHAR(50) NOT NULL,
    cantidad INT NOT NULL,
    monto_total NUMERIC NOT NULL,
    CONSTRAINT FACTURA_PK PRIMARY KEY(id_factura),
    CONSTRAINT FACTURAV_FK FOREIGN KEY (numero_venta)
    REFERENCES VENTA (numero_venta)
    );
```

4 Implementación

Posteriormente de la etapa de diseño, procedemos con la funcionalidad de la base de datos creando funciones, vistas y triggers para responder a los requerimiento.

4.1 Codificación

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

La función recibe un código de barras de un producto y devuelve la cantidad calculada de la resta del precio de venta que obtenemos del precio unitario del producto menos la ganancia que obtenemos del registro de precio de compra del producto. Al finalizar asignamos el valor de la resta en una variable declarada como utilidad y este será nuestro valor requerido.

```
CREATE FUNCTION fUtilidad(codigo_barras_prd VARCHAR(30))
RETURNS NUMERIC LANGUAGE plpgsql
AS $$
DECLARE
        precio_venta NUMERIC;
        ganancia NUMERIC;
        utilidad NUMERIC;
BEGIN
        SELECT precio_unitario
        FROM PRODUCTO
        WHERE codigo_barras=codigo_barras_prd
        INTO precio_venta;

        SELECT precio_compra
        FROM INVENTARIO
        INTO ganancia;

        utilidad:= precio_venta-ganancia
        RETURN utilidad;
END
$$;
```

- 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.

La función emitirá un mensaje y desechará la transacción con el uso de ROLLBACK cuando el campo stock de la tabla inventario tenga un valor de cero, en el caso de que se realice una venta actualizará el campo de stock con una resta de la cantidad vendida contra lo que actualmente hay en stock, si no hay productos en stock emitirá un mensaje de alerta. Por ultimo emitirá un mensaje de advertencia en el caso que existan menos de tres productos en stock. Crearemos el trigger correspondiente para que se lance la función en el momento indicado.

```
CREATE OR REPLACE FUNCTION decStock()
RETURNS trigger LANGUAGE plpgsql
AS
$$
BEGIN
    IF stock<=0 THEN
        ROLLBACK;
        RAISE NOTICE 'Sin productos disponibles';
    ELSE
        UPDATE inventario SET stock=stock-cantidad
        WHERE inventario.precio_unitario=venta_detalles
        .precio_unitario;
            IF stock<0 THEN
                ROLLBACK;
                RAISE NOTICE 'Producto agotado';
            ELSIF unidades_stock<3 THEN
                RAISE NOTICE 'Quedan menos de tres
productosen almacen';
            END IF;
    END IF;
END
$$;



CREATE TRIGGER tdecStock
```

```
BEFORE INSERT
ON VENTA_DETALLES
FOR EACH ROW
EXECUTE PROCEDURE decStock();
```

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

La función recibe dos fechas, pueden ser diferentes o iguales para posteriormente utilizarlas. Realiza la suma de todos los datos del campo cantidad y hace uso de un INNER JOIN para obtener las fechas de venta, despues lo asigna en una variable y es lo que devuelve esta.

```
CREATE FUNCTION fCantidad_Total_Periodo(fecha_Inicio
DATE, fecha_Fin DATE)
RETURNS NUMERIC LANGUAGE plpgsql
AS $$
DECLARE
        venta_Total_Periodo NUMERIC:=0;
BEGIN
        SELECT SUM(cantidad)
        FROM VENTA_DETALLES VD
        INNER JOIN VENTA V ON V.numero_venta=VD.numero_venta
        WHERE V.fecha_venta BETWEEN fecha_Inicio AND fecha_Fin
        INTO venta_Total_Periodo;

        RETURN venta_Total_Periodo;
END
$$;
```

- Permitir obtener el nombre de aquellos productos de los cuales hay menos de 3 en stock.

Creamos una vista para no generar la consulta siempre, esta consulta hace uso de un inner join para recuperar el nombre del producto y la cantidad en stock y devolver los productos que tengan menos de 3 unidades en stock.

```
CREATE VIEW vwProductosPorAgotarse
AS
SELECT descripcion_producto
FROM PRODUCTO PR
INNER JOIN INVENTARIO IO ON IO.id_producto=PR.id_producto
WHERE IO.stock <= 2;
```

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

La función generará una factura recuperando los datos de las tablas de cliente, venta, venta detalles y producto al momento de una venta, posteriormente se irán asignando en cada columna de la tabla vacía llamada factura y con uso de un trigger generaremos estos datos siempre, aunque no se consulten todas las veces.

```
CREATE OR REPLACE FUNCTION fFactura()
RETURNS TRIGGER LANGUAGE plpgsql
AS
$$
DECLARE
        fNumero_venta VARCHAR(10);
        fFecha DATE:=current_date;
        fRazon_social VARCHAR(50);
        fConcepto VARCHAR(50);
        fCantidad INT:=0;
        fMonto_total NUMERIC;
BEGIN
        fNumero_venta:=new.numero_venta;

        SELECT razon_social
        FROM CLIENTE C
        INNER JOIN VENTA V ON V.id_cliente=C.id_cliente
        INNER JOIN VENTA_DETALLES VD
        ON VD.numero_venta=V.numero_venta
        WHERE VD.numero_venta=new.numero_venta
        INTO fRazon_social;
```

```
        SELECT descripcion_producto
        FROM PRODUCTO
        WHERE id_producto=new.id_producto
        INTO fConcepto;

        fCantidad:=new.cantidad;

        fMonto_total:=new.cantidad*new.precio_unitario;

        INSERT INTO FACTURA(numero_venta,fecha,razon_social,
        concepto,cantidad,monto_total)
        VALUES(fNumero_venta,
        fFecha,fRazon_social,fConcepto,
        fCantidad,fMonto_total);

        return new;
END
$$;

CREATE TRIGGER tFactura AFTER INSERT ON VENTA_DETALLES
FOR EACH ROW
EXECUTE PROCEDURE fFactura()
```

5 Presentación

Por ultimo con la utilización de una arquitectura cliente-servidor creada con javascript hicimos la interfaz gráfica que nos permitirá agregar datos de un cliente y tener la función de realizar una venta de hasta tres artículos.

Aplicación basada en modulos

Del lado del cliente, se optó por recurrir a una herramienta muy conocida de javascript como lo es react, fue algo muy útil la integración de algunos componentes. Utilizamos Matirial UI para darle una mejor vista a nuestro sistema.

React es una biblioteca de JavaScript para construir interfaces de usuario. React nos va a ayudar a crear aplicaciones web interactivas compuestas, mediante pequeñas y aisladas piezas de código llamadas componentes; React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien.

index.js

```
// const express = require('express');

import express from 'express';
import bodyParser from 'body-parser';
import session from 'express-session';
import cors from 'cors';

import router from './api/routes/routes';

const port = process.env.PORT || 4000;
const app = express();
app.use(cors());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
    res.setHeader('Access-Control-Allow-Credentials', true);
    next();
});
app.use(session({
  secret: 'conduit',
  cookie: {
    maxAge: 60000
  },
  resave: false,
  saveUninitialized: false
}));
app.use('/api/v1',router);
app.use((req, res) => {
  res.status(404).send({
    url: req.originalUrl + ' not found'
  })
});
const server = app.listen( port, () => {
  console.log('Listening on port ' + server.address().port);
});
```

Figura 5: Backend server .

Vista inicio

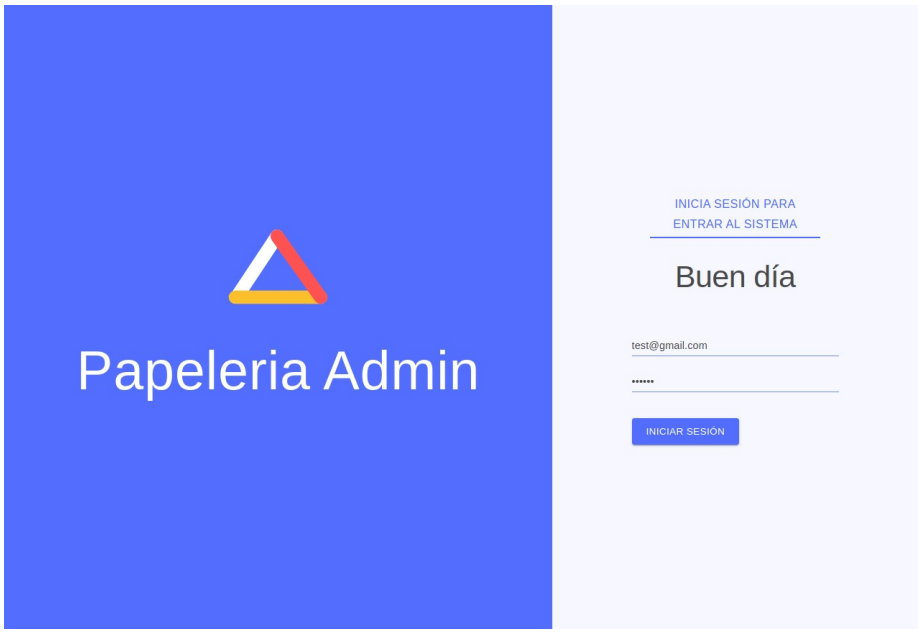


Figura 6: Credenciales de acceso.

Usamos los jwt para darle seguridad a nuestro sistema validando cada endpoint, se visualizó el sistema como un administrador, el cual nos permite realizar tareas y crear nuevas, considerando la escalabilidad de nuestro producto.

Decidimos no utilizar un patrón de diseño MVC, ya que, desde nuestra experiencia, a la hora de realizar una integración es más fácil dar soporte por separado y actualizar los elementos más rápido y eficientemente.

Por parte del servidor optamos utilizar un stack full javascript utilizando express para el manejo de nuestro backend haciendo una restful API haciendo una conexión con una base de datos Postgres.

Actualmente aún hay muchas empresas que siguen utilizando el clásico WebService viejo con XML. Las API REST están aquí para salvarnos de esa complejidad ya que gracias al envío de datos por medio de JSON o su recepción por este medio podemos hacer un desarrollo más rápido.

Aprovechando los métodos HTTP, desde un simple POST o GET hasta métodos personalizados, sin embargo, nosotros utilizamos únicamente POST, GET, para fines del proyecto.

Decidimos implementar JWT para la seguridad de nuestros usuarios, validando con tokens cada petición que se realiza.

Ingresar datos de cliente

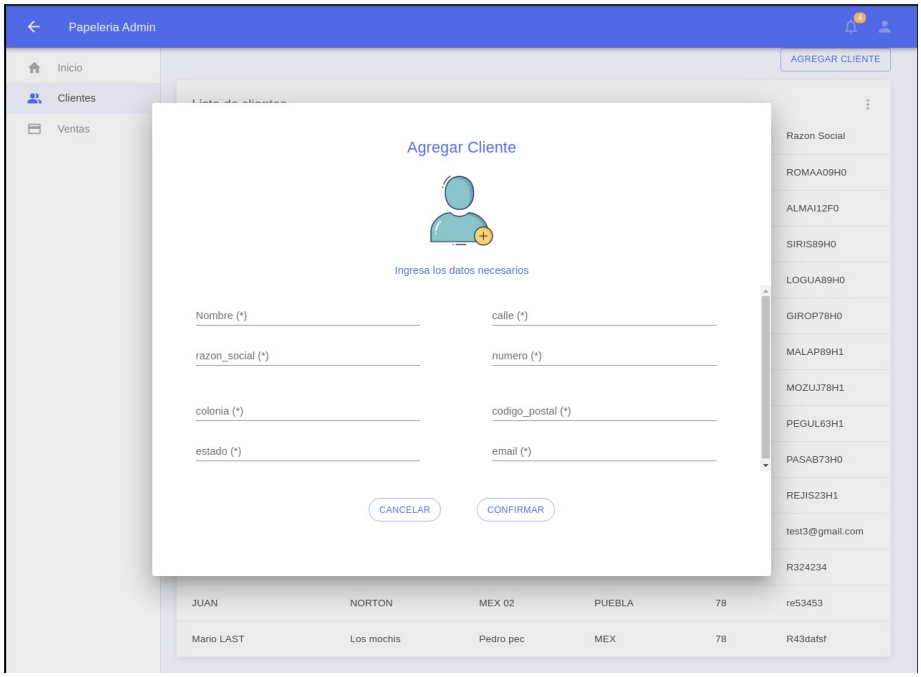


Figura 7: Ingresar cliente.

Realizar venta

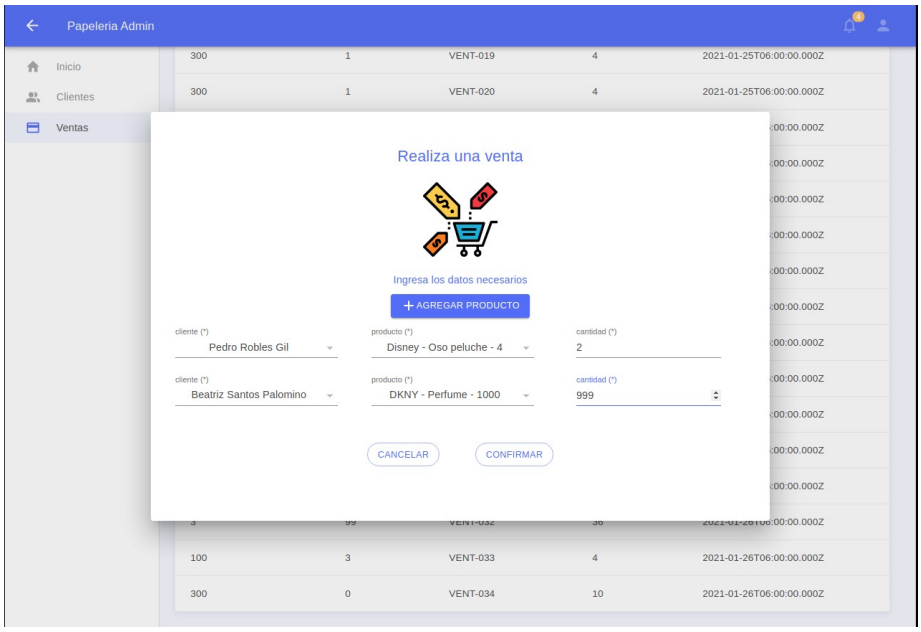


Figura 8: Realizar venta.

6 Conclusiones

- Cortés González Zaira Yaritsi:
  Con la elaboración de este proyecto, hemos aprendido a desarrollar los conocimientos adquiridos en clase, a su vez, se tuvo que hacer investigación para cumplir con uno de los requerimientos del proyecto, me refiero a la creación de una interfaz gráfica, también se pudo comprobar que existen demasiadas maneras de resolver el mismo problema, es por eso por lo que es importante la comunicación en el equipo para llegar a un acuerdo y así desarrollar el producto de una manera más eficiente.

  El proyecto cumple con el objetivo planteado y se aprendió a definir tareas específicas para que la planeación del proyecto sea acertada.

- Reyes Alonso Katherine:
  Con el desarrollo del Proyecto final de la asignatura Base de Datos podemos decir que se llegó la conclusión de que se cumplió con el objetivo del proyecto que fue analizar los requerimientos, de la parte uno, aplicamos todos los conceptos aprendidos en clase para hacer el diseño de la base de datos de la cadena de papelerías, una vez hecho el diseño de la base de datos proseguimos con la parte dos que fue realizar la interfaz gráfica que fue un proceso más complejo.

  Con el desarrollo del mismo fuimos creando la documentación que fue parte importante de la organización para poder manejar los tiempos y tener el proyecto para poder presentarlo e intentar venderlo al profesor como un servicio de manera formal.

- Rosales Velázquez Víctor Daniel:
  Realizar un proyecto de esta escala conlleva haber obtenido no solo el conocimiento del curso, si no también, horas de aprendizaje autogestivo y la formación de una personalidad autodidacta ya que se enlaza todos los temas que el curso puede ofrecer e inclusive mas allá de lo estipulado.

  Considero que el proyecto se realizó con una estricta planeación y detalle para poder abarcar todos los objetivos propuestos y una vez plasmado en este documento las pruebas de lo realizado por el equipo considero que se han logrado de manera satisfactoria.

Encuentro en este tipo de proyectos la aplicación real del enfoque ingenieril ya que implica un correcto trabajo en equipo y una buena administración de proyecto, en este caso, de software.

- Sánchez Maldonado Mario Alberto:
  Tomando en cuenta los objetivos de la asignatura de bases de datos, nos queda claro que más allá de la interacción con este ambiente de trabajo, es importante conocer los factores que rodean este gran ecosistema, el proyecto llevó al límite nuestra creatividad como ingenieros y para resolver problemas.

  Hablando del sistema en general, se utilizaron elementos bastante rigurosos en el mercado como es el tema de la seguridad, ya que al contar con una base de datos conlleva tener una gran responsabilidad, es por eso que al momento de pensar la arquitectura de nuestro producto, analizamos cuál sería el plus y llegamos a la conclusión que la implementación de un pequeño factor de autenticación aumentaron un valor importante en nuestro sistema.

  Por último y no menos importante se pensó en la escabilidad de este mismo, es por eso que se tomaron las mejores tecnologías en el mercado para desarrollar este sistema, en este caso se utilizó uno de los lenguajes más versátiles para esta implementación, como lo es javascript, utilizando sus herramientas más populares y robustas, por el lado del front end react y del lado del servidor Express.

  Sin duda un sistema bastante completo y con posibilidades de escalabilidad.

7 Bibliografía

1. Arellano, L. Hernández, L. Manual de prácticas de la asignatura de Bases de Datos. México. UNAM.

2. Tema II. Ing. Fernando Arreola. .Consultado el día 20 de diciembre de 2020. Recuperado de: https://github.com/FernandoArreolaF/Bases1UNAM /blob/master/Presentaciones/Tema$_I I$.*pdf*