



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Asignatura: Bases de Datos (1644)

Documentación Proyecto final.

Profesor: Ing. Fernando Arreola Franco

Integrantes del equipo:

- *Bautista Pérez Brian Jassiel*
- *Guzmán Ramírez Aldo Yael*
- *Pacheco Salgado Mauricio*
- *Romero Rivera Geovanni*
- *Vargas Jordán Antonio*

Grupo: 01

Semestre: 2022-1

Fecha de entrega: 11 de diciembre de 2021

Índice

1. Introducción	2
2. Objetivos	2
3. Plan de trabajo	2
4. Diseño	3
5. Implementación	3
6. Presentación	10
7. Conclusiones	14

1. Introducción

Durante este proyecto haremos el diseño de una página web en donde el usuario podrá realizar las compras de papelería. El frontend fue realizado en HTML y CSS y para el backend utilizamos el framework de PHP Laravel. Para el almacenamiento de datos utilizamos el lenguaje SQL con ayuda del DBMS PostgreSQL. En esta lo primero que debe realizarse será el registro por parte del usuario para tener sus datos almacenados para posteriormente pueda realizar sus compras y que éstas sean almacenadas para que proceda a hacer el pago de sus productos. Lo que nuestra base almacenara será el nombre del usuario, así como una clave con la cual se pueda identificar si en otra ocasión decide volver a comprar. También se hará el registro de los productos que realice, así como la fecha, total de productos que se llevó y el total a pagar. Esto para mantener un registro en nuestra venta de nuestros productos. Podrán hacer consultas de la cantidad de productos vendidos, así como el ingreso total de ventas en un lapso dado. También podrá ver informe de los productos que quedan en el inventario.

2. Objetivos

- Se realizará la creación de una página web en la cual puedan realizarse compras de artículos de papelería.
- Usar una base de datos en la cual se almacenará la información necesaria para dicha página.
- Comprender el manejo de información entre la página web y la base de datos.
- Hacer uso de PL / SQL.

3. Plan de trabajo

La organización de todo el proyecto requirió de sesiones semanales o bien cada tercer día dependiendo de la disponibilidad de todos los integrantes. En una primera instancia se realizó este diagrama de Gantt con la intención de dividir este proyecto por fases, quedando de la siguiente forma:

PROYECTO FINAL BASES DE DATOS

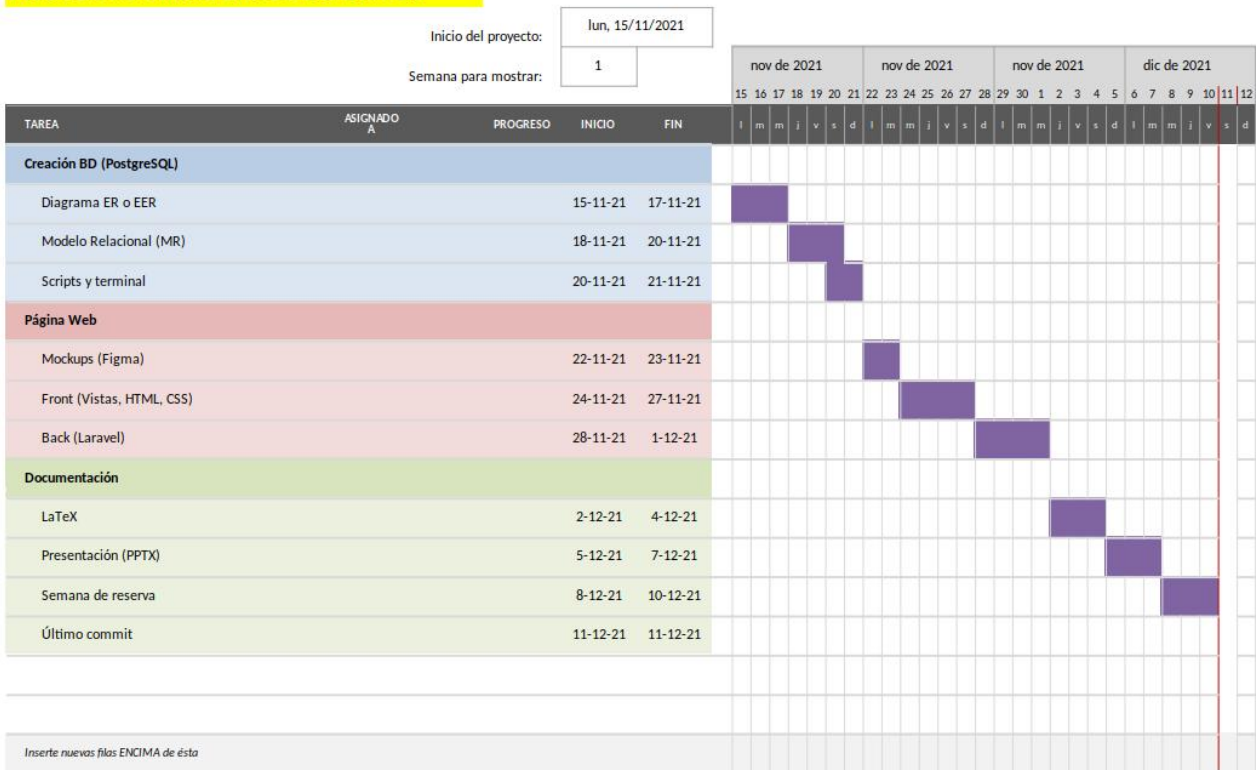


Figura 2: Diagrama de Gantt del proyecto.

4. Diseño

Se cubrieron tres rubros importantes: El diseño conceptual y lógico de la base de datos junto con su implementación en terminal, luego el diseño de la página web en el cual se realizó tanto frontend como backend y finalmente la realización de la documentación.

Cada parte se realizó conforme a la planeación realizada previamente, en la primera semana tocó cubrir la creación de la base de datos, en la cual se obtuvo el modelo entidad relación y el modelo relacional, posteriormente se crearon las vistas para la página tales como el inicio, el login, el register, la sección de las compras y el catálogo y después se creó el proyecto de Laravel para programar ya toda la página y conectarla a la base de datos.

5. Implementación

Como primer paso para el diseño de nuestra base de datos creamos un diagrama entidad relación en el que se crearon 3 entidades principales sobre las que se trabajo todo el proyecto, estas son: “CLIENTES”, “INVENTARIO” y “PROVEEDORES”. Consideramos que son las únicas entidades necesarias, ya que otros

aspectos como la compra de producto o la venta a los clientes se pueden detallar mediante la relación entre estas.

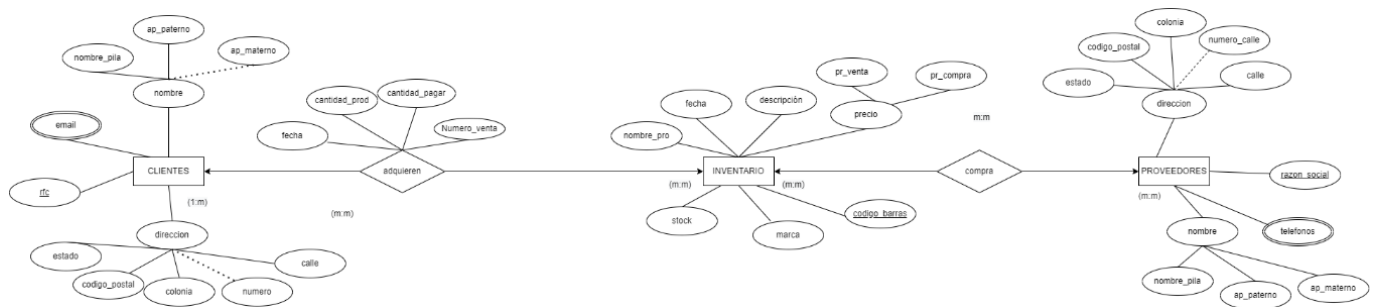


Figura 3: Diagrama de Entidad Relación.

Una vez definido eso se paso al mapeo intermedio y el diagrama relacional, en el cual definimos los tipos de datos e importancia de cada atributo, así como la cardinalidad final y paso de llaves foráneas.

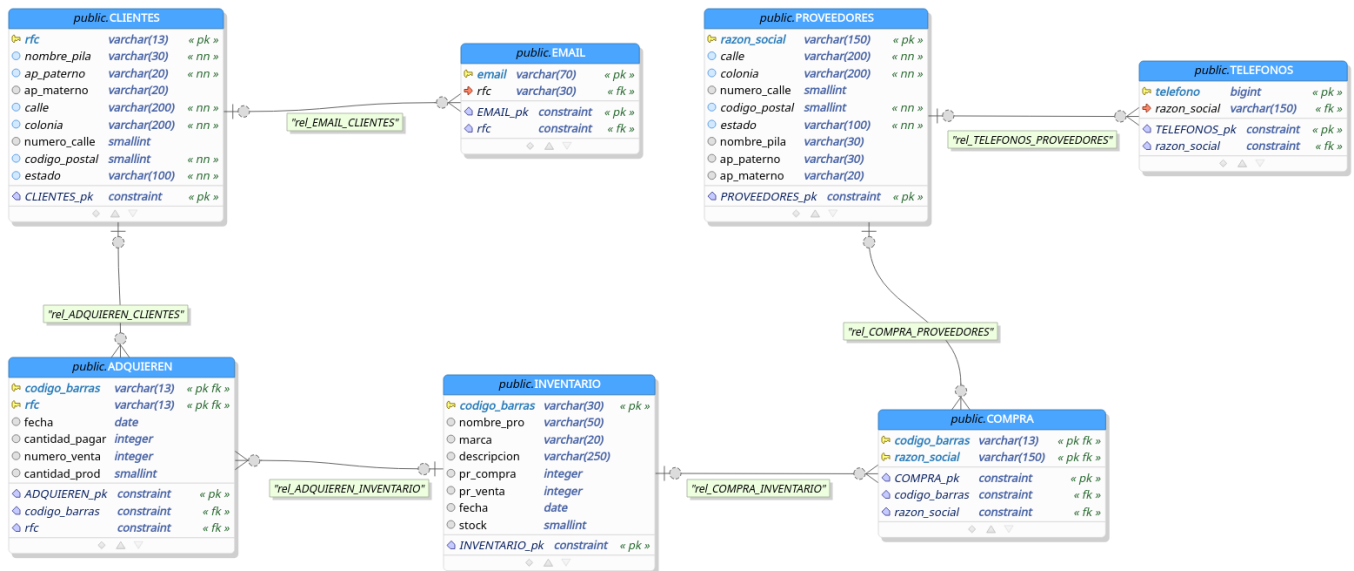


Figura 4: Modelo Relacional.

Creamos la base de datos

```
1 CREATE DATABASE papeleria_probd;
```

Creamos una secuencia en donde el mínimo valor sea 0 y el máximo sea 100 donde sec_ad se vaya incrementando uno en uno.

```
1 DROP TABLE public."ADQUIEREN";
2 DROP TABLE public."TELEFONOS";
3 DROP TABLE public."EMAIL";
4 DROP TABLE public."CLIENTES";
5 DROP TABLE public."COMPRA";
6 DROP TABLE public."INVENTARIO";
7 DROP TABLE public."PROVEEDORES";
8 DROP SEQUENCE SEC_AD;
```

creamos una tabla publica llamada clientes con 9 atributos con sus respectivos datos: varchar y smallint, utilizamos constraint en donde seleccionamos que el campo rfc tiene que ser único y para ello lo seleccionamos como llave primaria. y en la mayoría de los atributos los designamos not null para que no queden vacíos.

```
1 CREATE TABLE public."CLIENTES" (  
2   rfc varchar(13) NOT NULL,  
3   nombre_pila varchar(30) NOT NULL,  
4   ap_paterno varchar(20) NOT NULL,  
5   ap_materno varchar(20),  
6   calle varchar(200) NOT NULL,  
7   colonia varchar(200) NOT NULL,  
8   numero_calle smallint,  
9   codigo_postal smallint NOT NULL,  
10  estado varchar(100) NOT NULL,  
11  CONSTRAINT "CLIENTES_pk" PRIMARY KEY (rfc)  
12 )  
13 ;
```

Con el alter table y el owner to, cambiamos de propietario de la base de datos siendo postgres como su propietario.

```
1 ALTER TABLE public."CLIENTES" OWNER TO postgres;
```

Creamos una tabla publica llamada email con 2 atributos con sus datos varchar, siendo su llave primaria email utilizando constraint para su unicidad.

```
1 CREATE TABLE public."EMAIL" (  
2   email varchar(70) NOT NULL,  
3   rfc varchar(30),  
4   CONSTRAINT "EMAIL_pk" PRIMARY KEY (email)  
5 )  
6 ;
```

con el alter table y el owner to, cambiamos de propietario a la tabla publica email siendo postgres como su propietario

```
1 ALTER TABLE public."EMAIL" OWNER TO postgres;
```

Creamos una tabla publica llamada adquieren con 6 atributos con sus respectivos datos: varchar, date, smallint y float, en donde es una relacion de m:m entre las tablas clientes e inventario teniendo como llave primaria compuesta el codigo de barras y el rfc. En el atributo numero_venta designamos una restriccion llamada default, el cual nos permite colocar un dato por defecto seguido de un nextval que nos va a servir. Para recuperar los valores de una secuencia y en este caso la secuencia es la que creamos anteriormente "sec_ad." el cual va aumentando de 1 en 1 hasta 100 a medida que se ingrese el número de venta. y designando not null en algunos atributos.

```
1 CREATE TABLE public."ADQUIEREN" (  
2   codigo_barras varchar(13) NOT NULL,  
3   rfc varchar(13) NOT NULL,  
4   fecha date,  
5   cantidad_prod smallint,  
6   cantidad_pagar float,  
7   numero_venta varchar(9) DEFAULT 'VENT-00' || NEXTVAL('SEC_AD'),  
8   CONSTRAINT "ADQUIEREN_pk" PRIMARY KEY (codigo_barras, rfc)  
9 )  
10 ;
```

Cambiamos de propietario a la tabla pública email.

```
1 ALTER TABLE public."ADQUIEREN" OWNER TO postgres;
```

Creamos la tabla pública inventario con 8 atributos con sus respectivos datos varchar, float, date, smallint asignado como llave primaria codigo_barras.

```
1 CREATE TABLE public."INVENTARIO" (  
2     codigo_barras varchar(30) NOT NULL,  
3     nombre_pro varchar(30),  
4     marca varchar(20),  
5     descripcion varchar(250),  
6     pr_compra float,  
7     pr_venta float,  
8     fecha date,  
9     stock smallint,  
10    CONSTRAINT "INVENTARIO_pk" PRIMARY KEY (codigo_barras)  
11 )  
12 );
```

Cambiamos de propietario a la tabla pública inventario.

```
1 ALTER TABLE public."INVENTARIO" OWNER TO postgres;
```

Creamos la tabla pública llamada compra con 2 atributos con sus datos varchar siendo estos not null. Al ser una relación m:m entre las tablas inventario y proveedores, vamos a tener una llave primaria compuesta con sus respectivas llaves primarias: codigo_barras y razon_social.

```
1 CREATE TABLE public."COMPRA" (  
2     codigo_barras varchar(13) NOT NULL,  
3     razon_social varchar(150) NOT NULL,  
4     CONSTRAINT "COMPRA_pk" PRIMARY KEY (codigo_barras, razon_social)  
5  
6 );
```

Cambiamos de propietario a la tabla pública compra:

```
1 ALTER TABLE public."COMPRA" OWNER TO postgres;
```

Creamos la tabla pública proveedores con 9 atributos con sus respectivos datos: varchar, small int y integer siendo algunos de estos not null, asignando como llave primaria la razon social

```
1 CREATE TABLE public."PROVEEDORES" (  
2     razon_social varchar(150) NOT NULL,  
3     calle varchar(200) NOT NULL,  
4     colonia varchar(200) NOT NULL,  
5     numero_calle smallint,  
6     codigo_postal integer NOT NULL,  
7     estado varchar(100) NOT NULL,  
8     nombre_pila varchar(30),  
9     ap_paterno varchar(30),  
10    ap_materno varchar(20),  
11    CONSTRAINT "PROVEEDORES_pk" PRIMARY KEY (razon_social)  
12 )  
13 );
```

Cambiamos de propietario a la tabla pública proveedores.

```
1 ALTER TABLE public."PROVEEDORES" OWNER TO postgres;
```

Creamos la tabla pública llamada teléfonos con 2 atributos con sus respectivos datos: varchar y bigint, y asignamos la llave primaria teléfono.

```
1 CREATE TABLE public."TELEFONOS" (  
2     telefono bigint NOT NULL,  
3     razon_social varchar(150),  
4     CONSTRAINT "TELEFONOS_pk" PRIMARY KEY (telefono)  
5  
6 );
```

Cambiamos de propietario a la tabla a la tabla pública teléfonos.

```
1 ALTER TABLE public."TELEFONOS" OWNER TO postgres;
```

Con alter table modificamos la tabla email añadiendo una restricción de llave foránea siendo el atributo rfc y a la vez esta referenciado a la tabla clientes. La última sentencia se refiere a que, si en algún momento se elimina o se mejora una fila de la tabla principal, no va a realizar ninguna acción y va a devolver un error “evita eliminar un padre cuando hay hijos”.

```
1 ALTER TABLE public."EMAIL" ADD CONSTRAINT rfc FOREIGN KEY (rfc)  
2 REFERENCES public."CLIENTES" (rfc) MATCH FULL  
3 ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Modificamos la tabla adquieren añadiendo una llave foránea llamada codigo_barras de la tabla inventario. Y si en dado caso se eliminara una fila de la tabla principal va a devolver un error.

```
1 ALTER TABLE public."ADQUIEREN" ADD CONSTRAINT codigo_barras FOREIGN KEY (codigo_barras)  
2 REFERENCES public."INVENTARIO" (codigo_barras) MATCH FULL  
3 ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Modificamos la tabla adquieren añadiendo otra llave foránea llamada rfc de la tabla clientes. Recordemos que la tabla adquiere es una relación entre esas dos tablas y su llave primaria es una llave compuesta entre las dos llaves primarias de las dos tablas. Poniendo la misma sentencia de que si se modifica algo incorrecto en la tabla principal, no hará nada y devolverá un error.

```
1 ALTER TABLE public."COMPRA" ADD CONSTRAINT codigo_barras FOREIGN KEY (codigo_barras)  
2 REFERENCES public."INVENTARIO" (codigo_barras) MATCH FULL  
3 ON DELETE NO ACTION ON UPDATE NO ACTION;
```

La tabla compra es una relación entre las tablas inventario y proveedores entonces le agregamos otra llave foránea llamada razon_social de la tabla proveedores. Poniendo la misma sentencia de que si se modifica algo incorrecto en la tabla principal, no hará nada y devolverá un error

```
1 ALTER TABLE public."COMPRA" ADD CONSTRAINT razon_social FOREIGN KEY (razon_social)  
2 REFERENCES public."PROVEEDORES" (razon_social) MATCH FULL  
3 ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Modificamos la tabla teléfonos añadiéndole una llave foránea llamada razon_social de la tabla proveedores. Poniendo la misma sentencia de que si se modifica algo incorrecto en la tabla principal, no hará nada y devolverá un error.

```
1 ALTER TABLE public."TELEFONOS" ADD CONSTRAINT razon_social FOREIGN KEY (razon_social)  
2 REFERENCES public."PROVEEDORES" (razon_social) MATCH FULL  
3 ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Posterior a esto, procederemos a insertar los datos (Las inserciones no se mostrarán en este documento para no extender este documento, sin embargo, todas las inserciones están en el script)

Posterior a las inserciones se crearán las funciones PL/SQL en las que desarrollamos las tareas solicitadas, por ejemplo, el primer punto pedía que a través del código de barras de un producto nos regresara la utilidad, lo

que interpretamos como mostrar el nombre, descripción y precio de un producto. Cosa que logramos mediante un “RETURN QUERY SELECT” dentro de una función como se muestra a continuación.

```
1 CREATE OR REPLACE FUNCTION obten_utilidad(codigo varchar) RETURNS TABLE(nombre varchar,  
   descrip varchar,precio float) AS $$  
2 BEGIN  
3 RETURN QUERY SELECT  
4 nombre_pro,descripcion,pr_venta  
5 FROM public."INVENTARIO"  
6 WHERE codigo_barras ILIKE codigo;  
7 END  
8 $$ LANGUAGE PLPGSQL;
```

Lo que regresamos es una tabla con los atributos mencionados, y tal como podría haber sido de manera normal, solo tenemos que crear una consulta siendo específicos sobre lo que nos interesa encontrar. Finalmente, podemos observar como funciona

```
263 SELECT * FROM obten_utilidad('3923573943396');  
264  
265
```

	Data Output	Explain	Messages	Notifications
	nombre character varying	descrip character varying	precio double precision	
1	Tijeras	Tijera Maped de diferentes colores	35.45	

Este fue un proceso que repetimos bastante, ya que la mayoría de los puntos se resolvían con consultas, por lo que solo teníamos que ingeniar una manera de acomodar los datos y los atributos de manera que cumplieran nuestros requisitos, por ejemplo, el caso de obtener productos de los cuales solo quedaran menos de 3 en stock.

```
1 -- ARTICULOS CON MENOS DE 3 UNIDADES EN STOCK --  
2 CREATE OR REPLACE FUNCTION poco_stock() RETURNS TABLE(nombre varchar,existencias  
   smallint) AS $$  
3 BEGIN  
4 RETURN QUERY SELECT  
5 nombre_pro,stock  
6 FROM public."INVENTARIO"  
7 GROUP BY nombre_pro,stock  
8 HAVING stock < 3;  
9 END  
10 $$ LANGUAGE PLPGSQL;
```

Este tipo de ejercicios nos resultaron bastante útiles, pues de esta manera practicamos usando diversas herramientas de consulta y filtrado de datos, como el “GROUP BY” o “HAVING”

```
281 SELECT * FROM poco_stock();
```

```
282
```

Data Output

Explain

Messages

Noti

	nombre character varying	existencias smallint
1	Peluche	1

Para el punto donde teníamos que ingresar una o dos fechas y teníamos que obtener el total de artículos vendidos más la ganancia obtenida en ese periodo o fecha, presento un pequeño reto pues se tenía la intención de crear una única función que fuera capaz de aceptar uno o dos fechas como parámetros, sin embargo optamos por crear dos funciones con el mismo nombre, que fueran llamadas según los parámetros que se proporcionaran al ejecutarlas, tal como vimos un poco en una de las clases de teoría; obteniendo lo siguiente

```
1 -- VENTA Y GANANCIA PARA DOS FECHAS --
2 CREATE OR REPLACE FUNCTION venta_periodo(f_ini date, f_fin date) RETURNS TABLE(cant_vend
   bigint, ganancia float) AS $$
3 BEGIN
4 RETURN QUERY SELECT
5 SUM(cantidad_prod), SUM(cantidad_pagar)
6 FROM public."ADQUIEREN"
7 WHERE fecha BETWEEN f_ini
8 AND f_fin;
9 END
10 $$ LANGUAGE PLPGSQL;
11
12 -- VENTA Y GANANCIA PARA UNA FECHA --
13 CREATE OR REPLACE FUNCTION venta_periodo(f_ini date) RETURNS TABLE(cant_vend bigint,
   ganancia float) AS $$
14 BEGIN
15 RETURN QUERY SELECT
16 SUM(cantidad_prod), SUM(cantidad_pagar)
17 FROM public."ADQUIEREN"
18 WHERE fecha = f_ini;
19 END
20 $$ LANGUAGE PLPGSQL;
```




Como observamos, ambas funciones son exactamente la misma, con la diferencia como ya se mencionaba, de contar con un parámetro menos para así poder obtener la venta y ganancia de una única fecha dada, a continuación, una muestra del funcionamiento de ambas, y para que se puedan comprobar los resultados, se va a proveer la información registrada de dos ventas.

```
-- INSERTANDO DATOS A LA TABLA ADQUIEREN --
INSERT INTO public."ADQUIEREN"
VALUES (9883533229694,'AYGR123456780','2021/12/09',4,300,DEFAULT);
INSERT INTO public."ADQUIEREN"
VALUES (8779957272437,'AYGR123456780','2020/12/02',3,300,DEFAULT);
```

Como se ve, el 9 de diciembre de 2021 hubo una venta de 4 artículos por un precio de 300 pesos totales, y el 2 de diciembre de 2020 se vendieron 3 artículos por un precio total de 300 pesos también. Ahora veamos que arrojan nuestras funciones

```
312 SELECT * FROM venta_periodo('2020/12/02');
```




```
313
```

	Data Output	Explain	Messages	Notifications
	 cant_vend bigint		ganancia double precision	
1		3	300	

Pues bien, tenemos precisamente lo que esperábamos, dada la fecha del 2 de diciembre de 2020 hubo una venta total de 3 artículos con un precio total de 300 pesos

```
310 SELECT * FROM venta_periodo('2020/12/02','2021/12/09');
```

```
311
```

	Data Output	Explain	Messages	Notifications
	 cant_vend bigint		ganancia double precision	
1		7	600	

Finalmente, durante el periodo de las dos fechas, tenemos la venta de 7 artículos con una ganancia de 600 pesos

```
314 -- INDICE --
```

```
315 CREATE INDEX I_INDICE ON public."INVENTARIO"(nombre_pro);
```

```
316
```

Finalmente, como se pedía hemos creado un índice dentro de la tabla inventario y sobre la columna nombre, pues es una de las más consultadas y mediante éste podemos asegurarnos de tener consultas más rápidas y eficientes.

6. Presentación

Página web

Para conectar la página web a la base de datos se hizo el uso de Laravel, el cual es un framework de PHP que se puede integrar perfectamente con PostgreSQL y otros manejadores como MySQL, MariaDB, etc.

El procedimiento para instalar Laravel es el siguiente:

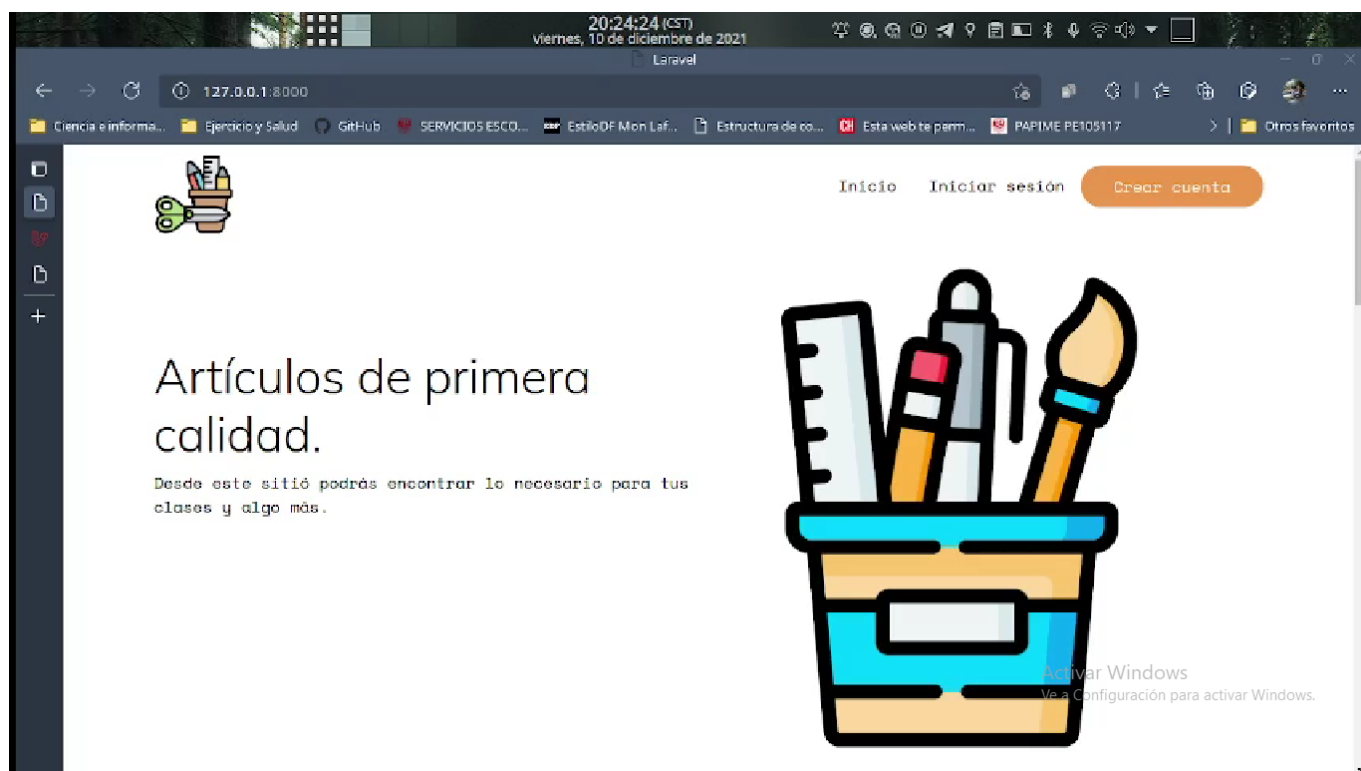
- Instalar PHP
- Instalar NodeJS (de preferencia una versión mayor a la 12)
- Instalar Composer que es el que hace posible la creación del proyecto

Al crear un proyecto en Laravel se crean diferentes archivos entre los cuales tenemos uno llamado .env, en dicho archivo se modifican los parámetros del proyecto tal como se muestra a continuación:

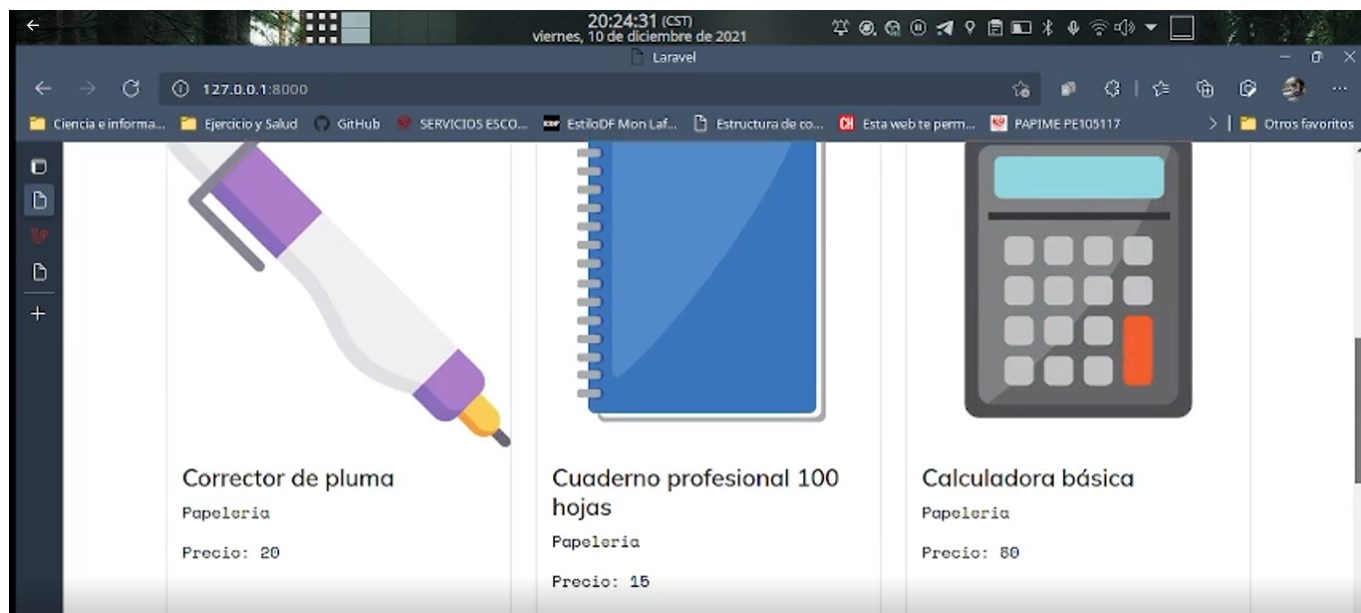
```
DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=papeleria_probdb
DB_USERNAME=brianb
DB_PASSWORD=123
```

Una vez creado el proyecto se integrará todo con las vistas creadas. (Este proceso se explicará en la presentación)

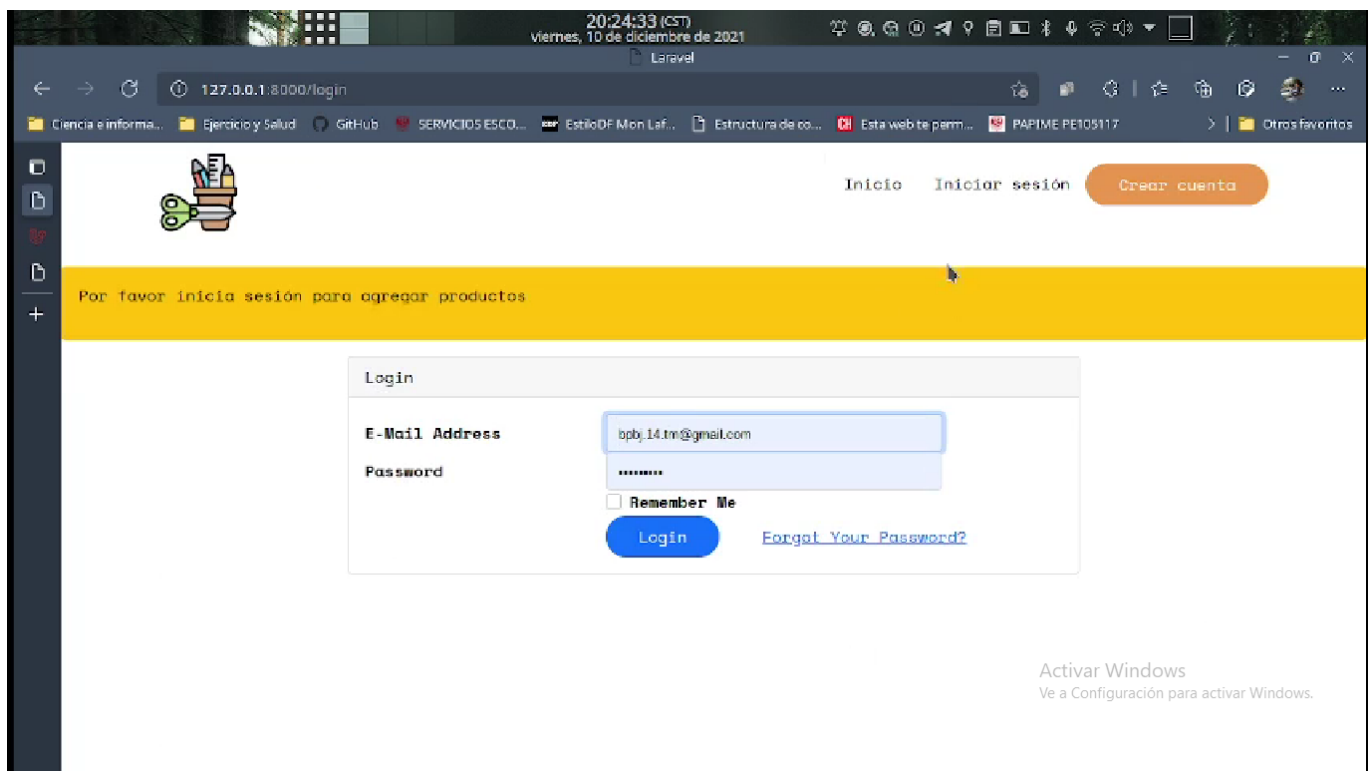
Esta ventana es nuestra vista principal en la cual podemos observar una bienvenida, así como una barra de menú en la cual están las opciones de inicio, iniciar sesión y crear cuenta.



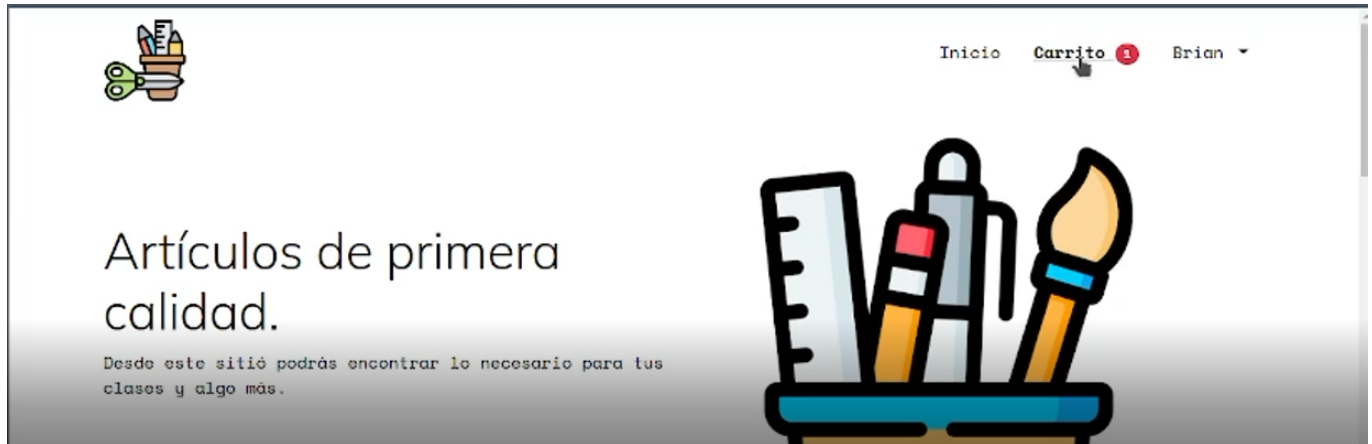
Si nos deslizamos hacia abajo, nos aparecerán los productos que deseemos comprar, si queremos añadirlos a nuestro carrito aun no podremos ya que tendremos que registrarnos primero.



Así que si le damos clic nos mandara a la parte de iniciar sesión.

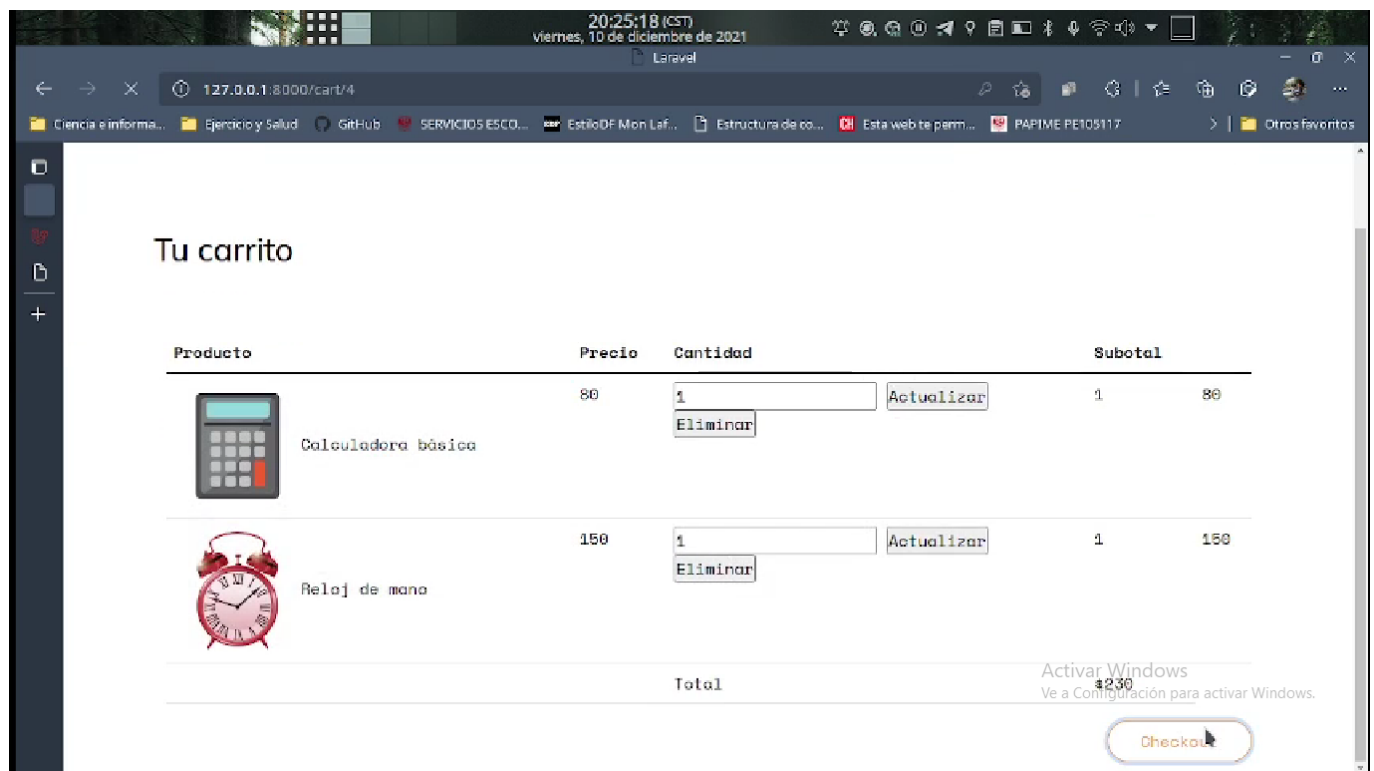


Una vez iniciando sesión el producto estará registrado en nuestro carrito de compras

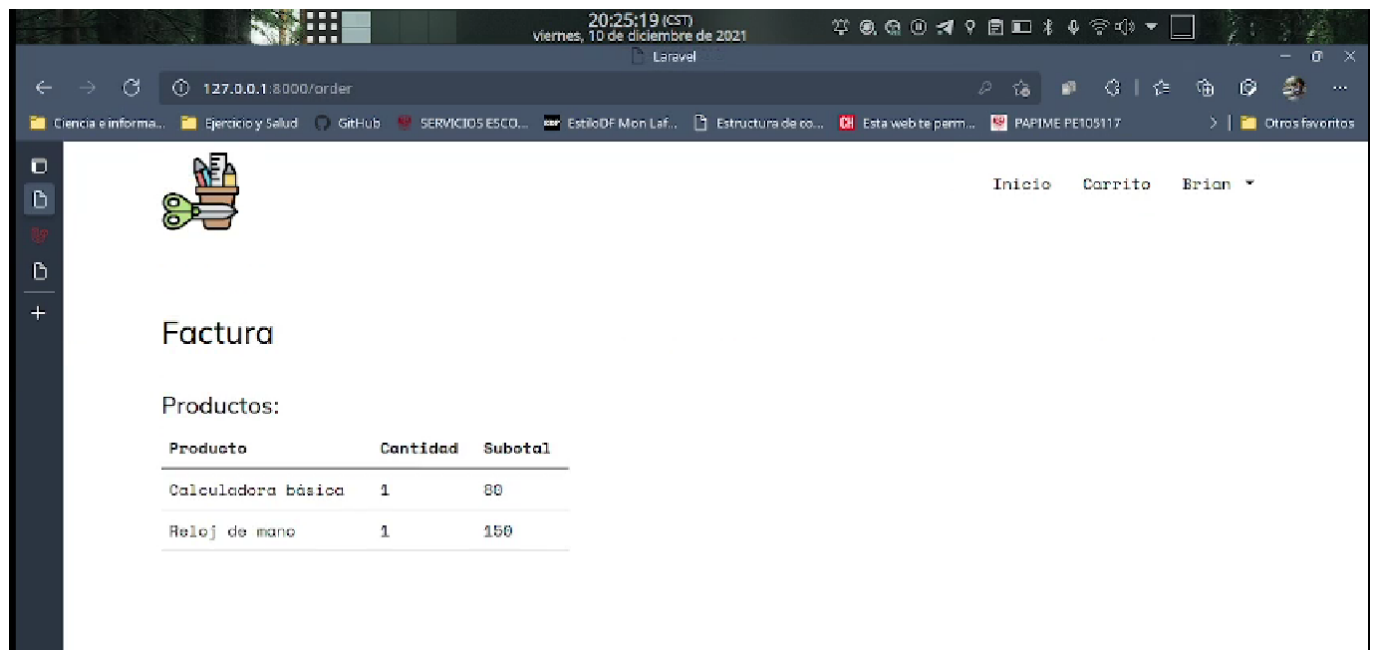


Vamos a esta vista: En este espacio podemos observar las siguientes características del producto tales como:

- Nombre del producto.
- Precio.
- Cantidad.
- Subtotal.



Y al hacer checkout se mostrará la factura de compra.



7. Conclusiones

- Brian Bautista: Respecto al proyecto, debo decir que fue un gran reto dado que sí fueron varios requerimientos y uno de ellos fue externo a lo que se vio en clase, sin embargo, no fue problema dado a que yo ya tenía un poco de experiencia utilizando Laravel, lo cual nos ahorró tiempo para crear la página. Debo

decir que quedo satisfecho dado que mayormente el proyecto quedo tanto a nivel PL / SQL como en la página, aunque en este último rubro tuvimos problemas para cubrir al cien por ciento la parte de las compras ya que sí podemos insertar productos y simular la compra, sin embargo el stock no decrementó en la página. Aún así creo que para ser un primer proyecto grande en Laravel quedó muy bien.

- Aldo Guzmán: Como modo de aprendizaje me llevo la experiencia de realizar la base de datos implementado a la página web. Así como empezar a crear funciones, triggers. Lo más pesado fue ingeniarse las funciones de tal forma que nosotros manipulemos la información para lograr lo requerido, así como la unión entre la base y la página. A pesar de que no logramos algunos puntos requeridos, gracias a todos mis compañeros de equipo realizamos la investigación para realizar la mayor parte de estos. Fue un proyecto realmente bueno el tiempo logro sobrepasarnos, así como también algunos conocimientos de programación que no logramos aprender con el tiempo. Pero nos quedamos con la experiencia de como organizarnos en un ámbito profesional y saber designar los roles para que cada miembro se centre en lo que mejor sabe hacer.
- Mauricio Salgado: Durante la realización de este proyecto tuvimos ciertas formas dudas y tuvimos que investigar algunas consultas previas que se nos complicaron a la hora de la programación de la base de datos. Fuera de esto, podemos decir que se cumplieron la mayoría de los puntos del proyecto, es un proyecto sencillo pero demandante, por lo que el trabajo en equipo fue esencial para que saliera todo bien y teniendo un orden para la realización de incluso la programación de las páginas web y los esquemas de entidad relación. Podemos decir que es un proyecto que nos adentra a lo que puede ser un ejercicio que nos adentra al campo laboral y que nos puede ayudar de referencia en futuros proyectos, por lo que es necesario saber entender en todo momento es un proyecto que contiene la mayoría del curso impartidos durante el semestre.
- Geovanni Romero: El proyecto fue de gran nivel y comenzamos a investigar desde cómo realizar una página web tanto el back end y front end. La parte que más me favoreció fue al momento de crear las funciones y los triggers. Mis compañeros de equipo supieron apreciar esto y me puse a trabajar en ello. La organización que tuvimos fue bastante buena debido a que todos colaborábamos en todo momento desde que iniciamos creando los modelos entidad relación y el relacional hasta la unión entre la base y la página. A la hora de programar funciones e implementar ciertas consultas en la base de datos, tuve que ingeniármelas con mis conocimientos previos obtenidos en el curso, era un poco confuso al inicio, pero una vez identificando los errores o mejor dicho la forma correcta de codificar en la base respetando la sintaxis y verificando que algunas palabras no fueran sensibles al lenguaje durante de esto. Todo fue más fácil después, por lo que en ciertos puntos eran un poco más sencillos de comprender que otros, o por lo menos unos tenían cierta dificultad personal. Pero nada que no se pueda realizar complementando conocimientos con mis compañeros de equipo e indagando un poco en lo que es la programación web implementada a las bases de datos desde postgres. Incluso puedo decir que algunos puntos eran complejos no por su dificultad, si no que en lo personal no había cierta información para implementarla en la web o en ciertos libros consultados para la parte de PL/SQL.
- Antonio Vargas: Fue uno de los proyectos mas interesantes y completos que he realizado durante todo mi transcurso en la Facultad porque a la hora de realizarlo se me presentaron distintas dificultades que sabia que tarde o temprano iban a presentarse como un reto si quería seguir adelante como estudiante de Ingeniería. Uno de los retos fue a la hora de realizar una pagina web, añadiendo un index.html y su css. Diría que fue una de las partes más difíciles para mí, porque no tenía tantas bases y no sabía cómo realizarla, pero al final, viendo videos y buscando información sobre html y css se logró, y puedo

decir que no fue tan difícil como pensaba y sobre todo es interesante y entretenido crear paginas a tu gusto. Otro de los retos a la hora de programación fue cumplir de manera exitosa los puntos de la Parte Uno ya que tuvimos que recordar lo visto de las clases pasadas, revisar los apuntes y ver de nuevo las clases y nos llevo más tiempo de lo previsto, pero aun que los puntos estuvieron complejos al principio se logró programar casi todo lo requerido ya que nos faltó que el stock decrementara por cada venta de ese artículo. Aunque se presentaron otros retos con menor dificultad, como enlazar todas las paginas que realizo el equipo o la forma en como conectar la Pagina a una base de datos, etc. Se cumplieron las expectativas del Proyecto, ya que al realizar la Pagina con ciertos requerimientos, así como tener toda su base de datos hizo que todo el equipo tuviera más conocimiento y más experiencia con el manejo de las Paginas conectadas a una base de datos. Me siento bastante satisfecho llegar al final de esto, hicimos una pagina de una Papeleria que cuenta con su sistema de login, creación de cuenta, carrito, factura y artículos de papeleria. En donde el usuario puede registrarse y así poder comprar distintos artículos y ser consciente de los productos que tiene en el carrito, así como tener a su dirección de envio y del lado del Dueño, tener control sobre que usuario se registra, que compra y cuanto compra, también tener acceso al stock de cada articulo y saber cuando contactar con los proveedores, pero sobre todo se logro de manera exitosa aplicar los conceptos vistos en clase. Hubo muchas gotas de sudor, muchas neuronas quemadas, muchas lágrimas, pero por fin el proyecto está terminado.