Proyecto Final: Diseño e implementación de un ecommerce para la empresa ProntoMueble, usando bases de datos relacionales.

Heiver Alejandro Suarez Cifuentes - 20191578010

Brian Steven Cañón Rojas – 20191578110

Nicolás Rocha Padilla - 20151578023

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD TECNOLÓGICA

TECNOLOGÍA EN SISTEMATIZACIÓN DE DATOS

BASES DE DATOS AVANZADAS.

Año 2020.

BOGOTA D.C

Tabla de contenido

Definición del problema **3**
Descripción de funcionalidades. 3
Descripción del entorno de ejecución. 9
Definición de las reglas del negocio. 13

Diagrama modelo entidad relación **15**

Diagrama del modelo relacional **16**

Diccionario de datos **17**
Descripción de tablas y columnas. 17
Descripción de reglas. 54
Descripción de procedimientos y triggers. 55
Descripción de vistas. 56

Código fuente **58**
Estructura de la base de datos. 58
Inserción de datos. 70
Conexión de la aplicación con la BD. 73

Referencias **74**

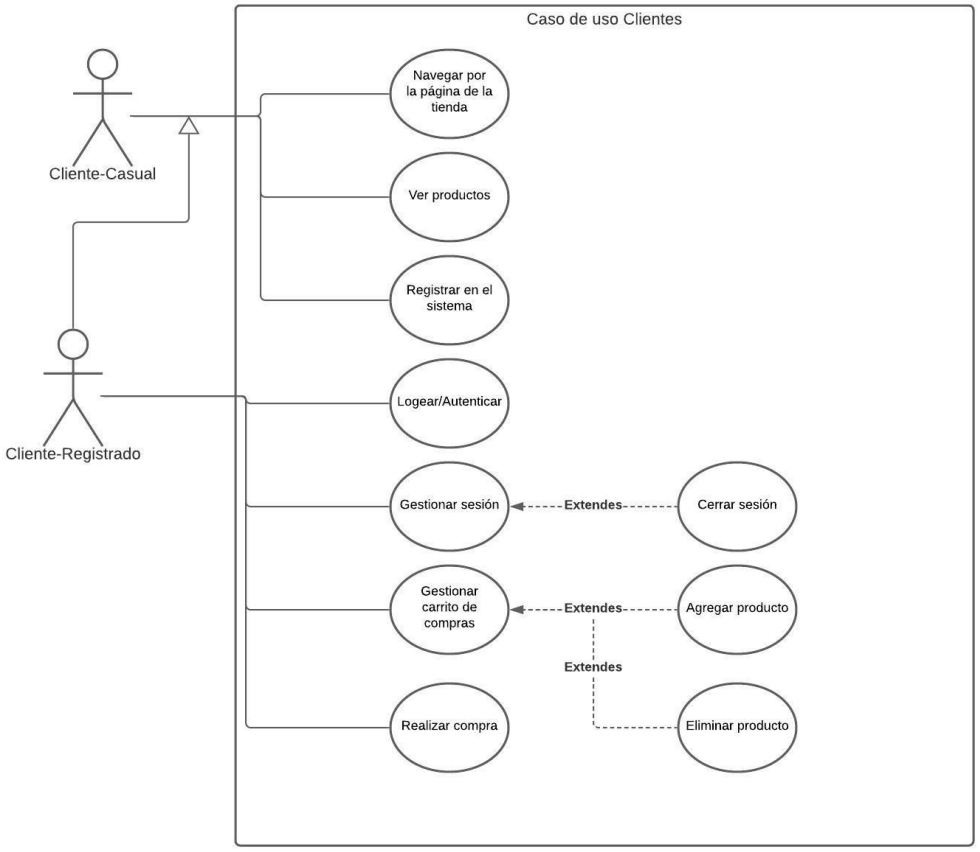
Definición del problema

A) Descripción de funcionalidades.

La empresa prontomueble ha hecho solicitud de un sistema tipo ecommerce el cual, de respuesta a las múltiples necesidades de la empresa, para ello desde la solución de sistemas se decidió implementar una respuesta al problema, para dar dicha respuesta se presentó la opción de diseñar una aplicación con las siguientes funcionalidades:

- El sistema debe tener varios tipos de roles, es decir el sistema debe permitir roles administrativos, de ventas y comunes, donde el rol administrativo se encargará de hacer estadísticas del sistema tales como: Ver los usuarios dentro del sistema, además de editarlos y crear nuevos, también podrá ver los roles y módulos existentes de la empresa y a su vez conceder el acceso a las distintas funcionalidades que ofrece la plataforma.
- Para el sistema de ventas este se implementará bajo la modalidad de carrito de compras, es decir el cliente o usuario común podrá agregar los muebles que necesite para comprar y a la final este los podrá comprar, después de dicho proceso el sistema generará la factura que será enviada al correo donde podrá hacer seguimiento de su compra.
- Para el sistema de proveedores será(n) el(los) administrador(es) del sistema que ingresará(an) los productos nuevos al inventario y por ende los pondrán a disposición para su venta.

A continuación, se presentan los respectivos diagramas de caso de uso para cada uno de los roles que hay en el sistema:

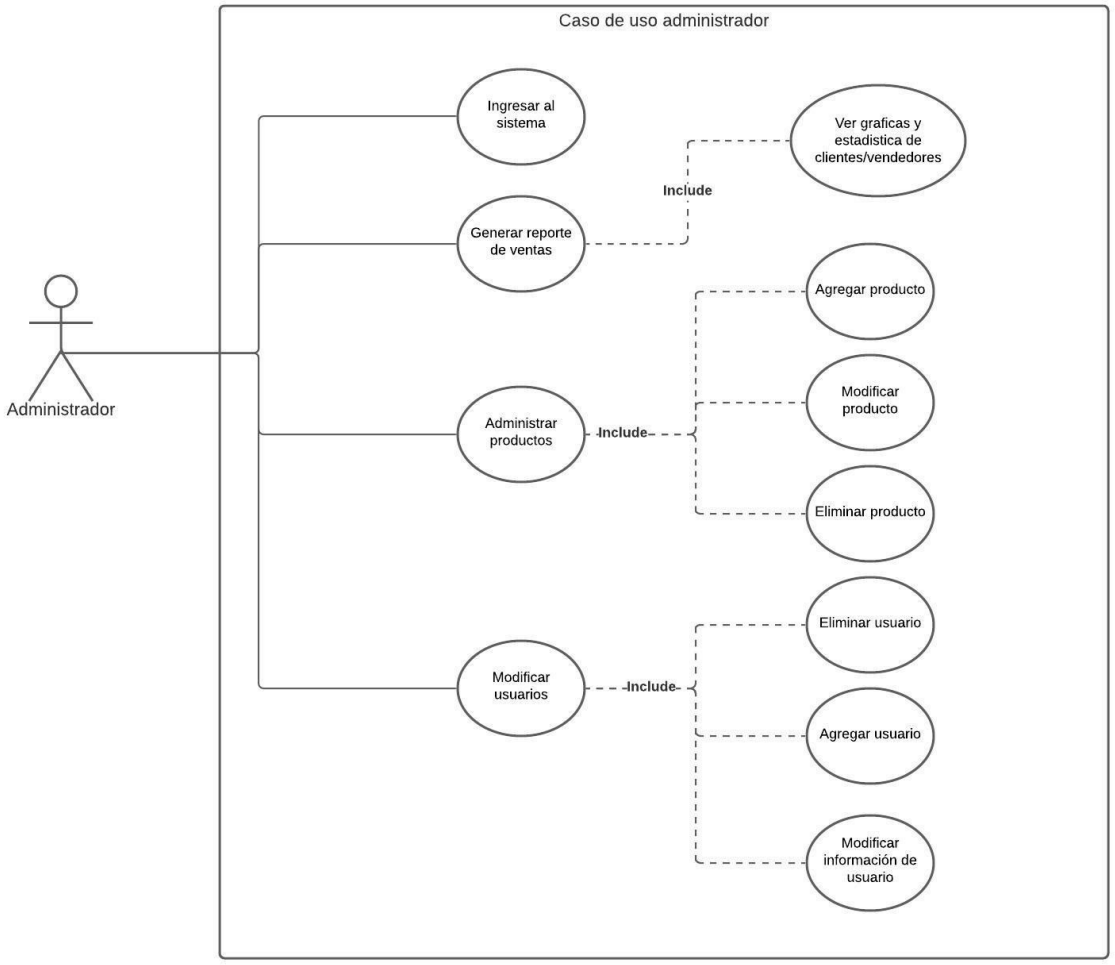


Como podemos observar el caso de uso presenta una solución a este problema, pero para dejar más claro aún se presentará una tabla que describe a detalle cada una de las funcionalidades del sistema, dicha tabla es la siguiente:

Función	**Descripción**
Navegar por la página de la tienda	Esta función hace alusión a lo que uno primero ve a la hora de entrar a una página web de una tienda o comercializadora de productos.
Ver productos	Como toda página web que promocione productos o tenga un catálogo en línea podrá ver productos, observar fotos físicas de estos y a su vez podrá mirar a futuro si los compra o no.

Registrar en el sistema	Esta función hace alusión a que cada usuario que desee comprar en la tienda podrá hacerlo con tan solo llenar un formulario y llenado de datos para que de esta forma el sistema le permita registrar su pago.
Gestionar sesión	Como toda página web se debe garantizar que el usuario pueda gestionar su estadía en la misma, es decir que cada persona podrá cerrar sesión en el sistema y sin perder su información, este modelo se manejara mediante el uso de cookies que provee PHP y demás.
Gestionar carrito de compras	Esta función hace a referencia a un ejemplo en la vida real, dicho ejemplo es que cuando uno va un supermercado, cada cliente tiene un carrito de compras donde guardará sus productos y después los comprará en alguna caja registradora, para este caso en cualquier momento los podrá comprar.
Realizar compra	Esta función como lo describe su nombre hace referencia al momento de comprar, se hará una vinculación con paypal para poder pagar el producto y así de una vez dar de alta a este mismo y hacer la gestión necesaria. Nota: Por razones de prueba y testeo el sistema contará con la vinculación temporal de PayPal para evitar mover dinero en la fase de pruebas.

Ahora para el caso del administrador tenemos las siguientes funcionalidades:

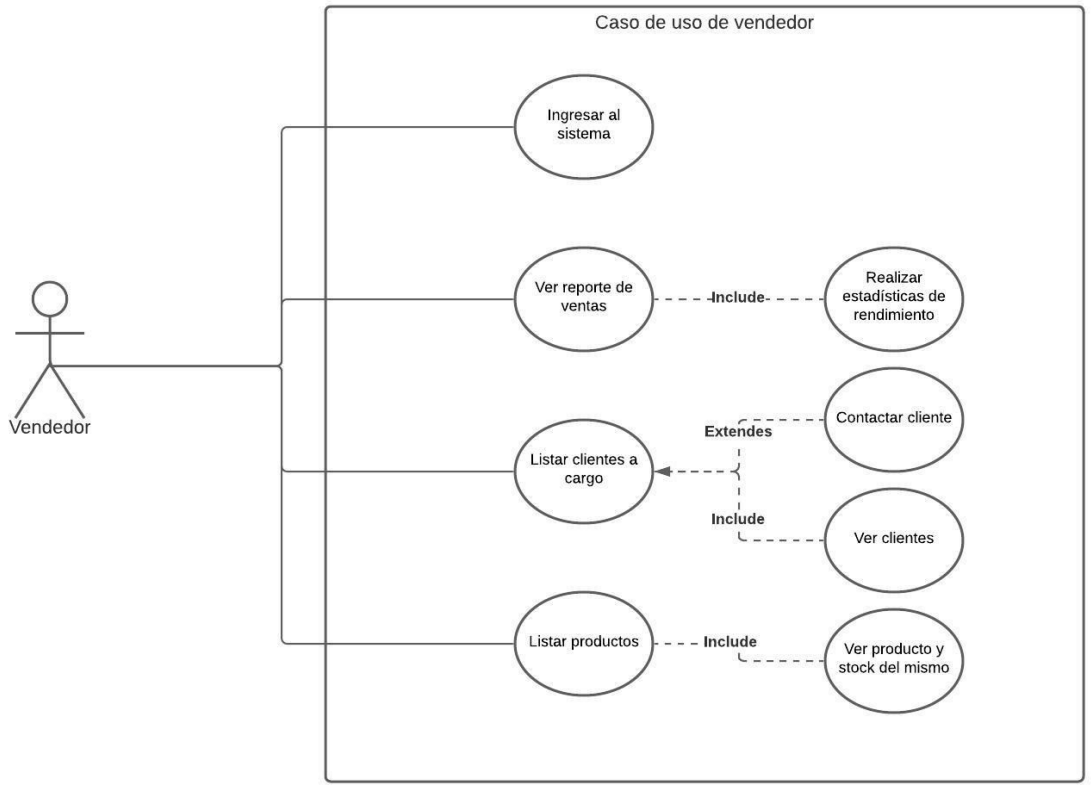


Para este caso la tabla de descripción de funcionalidades es la siguiente:

Función	Descripción
Generar reporte de ventas	En esta función el administrador podrá, generar un gráfico que le muestra cómo va el negocio en término de ventas y de productos.
Administrar productos	Esta funcionalidad le permite al administrador insertar, eliminar y modificar un producto además de generar estadísticas de los mismos tales como el producto más vendido y el menos vendido y demás.

Modificar usuarios	Como el nombre de la función lo indica, esta función le permite al administrador listar, modificar, insertar y eliminar usuarios para llevar un control correcto de la información.
Modificar permisos de usuarios y ver módulos existentes	Esta funcionalidad le permite al administrador dar acceso a los distintos permisos que hay en la aplicación, para ello él podrá concederle a un usuario permisos de edición,vista y lectura a determinado usuario.

Para finalizar con la descripción de funcionalidades se presentará el diagrama de caso de uso para el vendedor y su respectiva tabla de descripción, el respectivo diagrama y tabla se presenta a continuación:



Para dicho rol se tiene la siguiente tabla descriptiva:

Función	**Descripción**
Ver reporte de ventas	Para esta parte el vendedor podrá ver qué producto fue el más vendido y a su vez ver las ventas que ha realizado.
Listar clientes	En este apartado el vendedor podrá ver qué clientes hay dentro de la tienda, es decir, verá a los clientes que se les ha generado un pedido para que los pueda contactar.
Listar productos	En este apartado el vendedor podrá ver qué productos hay disponibles y a su vez llevar un inventario detallado.

B) Descripción del entorno de ejecución.

Para este apartado presentaremos una tabla para dar a conocer el software, lenguaje de programación, motor de base de datos y demás, que se va a utilizar en el desarrollo del proyecto, de mismo modo se recalca que este proyecto fue diseñado para cualquier tipo de sistema operativo e incluso es adaptable a cualquier dispositivo con una conexión a internet estable, para ello la tabla es la siguiente:

Recurso	Descripción	¿Por qué?
MySQL	MySQL es un motor de base de datos, especializado en el manejo de bases de datos de tipo relacional, este motor cuenta con una gran afinidad de funciones y métodos para la administración de bases de datos y mediante su administrador conocido como phpMyAdmin se hace una administración sútil de la base de datos.	Este motor fue escogido por requerimientos del sistema, dado que como la aplicación será hosteada gracias a la empresa de hostinger, este hosting exige que sus bases de datos estén en MySQL ya que es el motor más compatible con desarrollos web y su conexión con la aplicación es más fácil de realizar a diferencia de PostgreSQL y otros motores.
HTML	HTML o lenguaje de hypermarcas de texto en español, es un lenguaje de marcado reconocido a nivel mundial para el diseño de páginas web, este lenguaje de marcado es indiferente al sistema operativo por lo cual este puede correr en cualquier dispositivo que posea una conexión a internet mediante un navegador.	Dado a que se va a diseñar una aplicación web, es obligatorio el uso de HTML para que este pueda ser leído y procesado por el navegador del cliente o de quien quiera usar la aplicación, cabe destacar que el cliente final no necesita de conocimientos previos en HTML para el manejo de la aplicación.

	CSS o como es conocido en español hojas de estilo en cascada, es un lenguaje de diseño gráfico que permite crear estilos únicos a cualquier aplicación web, cabe destacar que CSS es un complemento de HTML y que sin este CSS no puede funcionar ya que este toma elementos del lenguaje de marcado de texto y los procesa para crear diseños agradables al cliente.	Para dar una experiencia agradable a la vista del cliente, es necesario integrar CSS para dar estilos únicos a la página web además de agregar estilos únicos y hacer que nuestra página web se distinga sobre las demás.
	JavaScript es un lenguaje de programación interpretado,este lenguaje está orientado a objetos e imperativo, en su mayoría es utilizado para dar vida a la página por medio de animaciones simples.	Para nosotros es importante que el cliente se sienta agusto con la página web, para lograr esto utilizaremos JavaScript para integrar animaciones simples y geniales para el usuario.
	AJAX o javaScript asíncrono y XML es un conjunto de técnicas de programación web que permiten el procesamiento en el servidor de forma de segundo plano, evitando que la aplicación se congestione y a la final caiga por completo dado a la infinidad de peticiones que este pueda tener.	Para nosotros es importante asegurar que nuestra aplicación o desarrollo sea fluido y no presente problemas de procesamiento, para ello, es importante vincular AJAX de esta manera se logra evitar una caída del servidor ante una eventual elevación de peticiones o de procesos en el lado del servidor.

	JSON o notación de objetos en JavaScript es un formato de texto sencillo para el intercambio de datos. Al igual que XML, este permite encapsular un desarrollo web y dejar que los datos se accedan de manera más segura, por lo general este tipo de lenguajes se complementa con archivos de tipo .htaccess.	Para nosotros es importante evitar fugas de información en la aplicación que se diseñe para eso se incluye JSON para evitar dichas fugas y a su vez evitar dejar el sistema expuesto a posibles vulneraciones de seguridad.
Bootstrap & jQuery	Bootstrap y JQuery son dos frameworks para desarrollo front-end, que se usan para complementar la parte estética de la página web.	Para garantizar una experiencia agradable a la vista es necesario integrar este framework que ya viene con estilos predefinidos y solo integra elementos como botones y demás, facilitando el trabajo del front-end.
	PHP es un lenguaje de programación enfocado al desarrollo web a nivel profesional, este lenguaje ofrece una gran gama de métodos, formas de trabajar únicas y simples, cabe destacar que este lenguaje de programación solo corre bajo servidores Apache y no permite correrse sin este servidor, por lo general viene bajo software como XAMPP, WAMPP y demás.	Para garantizar un correcto funcionamiento en nuestra aplicación debemos implementar PHP ya que este nos va a permitir la conexión con la base de datos, la manera en la que se comportan cada una de las funciones, cabe destacar que este desarrollo implementará el patrón de MVC o modelo vista controlador que permite una encapsulación total del sistema en carpetas o clases usando POO o programación orientada a objetos.

A modo de conclusión se debe resaltar que este sistema no tiene un requerimiento mínimo y máximo en términos de hardware por parte del cliente, este sistema al ser desarrollado de manera online, permite a nuestros usuarios entrar desde cualquier navegador o dispositivo móvil que posea una conexión a internet y pueda correr un navegador, de esta manera garantizamos la universalidad y estabilidad del sistema, algo a destacar es que esta página web será hosteada gracias a la empresa de Hostinger que ofrece planes a bajos precios y cómodos, además de ofrecer un SSL gratuito de por vida.

Nota: Para más información puede consultar https://www.hostinger.co y nuestro dominio actualmente registrado es https://prontomueble.store cuya fecha de caducidad es el 07/04/2021.

C) Definición de las reglas del negocio.

Como se ha explicado a lo largo de este documento, la empresa prontomueble ha hecho una solicitud de un diseño de una plataforma de tipo ecommerce que permita la gestión, venta y creación de sus productos, para ello, se creará una aplicación web que contendrá la información de la empresa, mostrará información relevante de la empresa acompañada de un catálogo de productos, para dicha aplicación se tienen las siguientes reglas:

- Debe de haber un usuario administrativo, que permita la creación y supervisión tanto de los empleados como de los productos y de los pedidos de los clientes, para esto se usará un panel administrativo que genere gráficas o tablas en tiempo real y que sean en base a lo obtenido en la base de datos.
- Como en toda empresa debe de haber un vendedor o empleado, este vendedor tendrá la tarea de supervisar, crear y manipular la información de los productos además de consultar información importante y precisa para aumentar su productividad.
- Para los clientes, esta parte se enfocará en diseñar una página principal para que ellos sean los que decidan qué comprar y a su vez lo agreguen al carrito de compras para luego finalizar con su compra mediante pago contra entrega o paypal.
- En la empresa cada empleado tendrá un rol, dicho rol será almacenado en la base de datos, dicho rol tendrá permisos asociados a vista,lectura y escritura (Como un sistema UNIX) sobre la información del sistema, en pocas palabras cada usuario tendrá limitaciones sobre lo que puede o no hacer en el sistema siendo el administrador el “superusuario” y el cliente una persona con permisos de vista/lectura únicamente.
- La página se dividirá en dos: la parte de la tienda y el módulo administrativo, en la primera parte será la que siempre se le va a mostrar al cliente y la segunda parte será la que usarán los empleados del establecimiento para gestionar las ventas y demás, el cliente solo tendrá acceso al apartado de sus pedidos (Si intenta acceder a otros pedidos mediante URL no podrá) .

- Para evitar que el usuario guarde datos innecesarios en la base de datos, está solo contendrá la información básica a la hora de realizar un pedido este deberá especificar el lugar de destino.
- La aplicación debe permitir generar archivos PDF,Excel y CSV para guardar la información presentada en las tablas.
- Los proveedores de prontomueble solo se especializan en una categoría, es decir, los proveedores no pueden vender productos de otras categorías, a modo de ejemplo supongamos que la empresa cuenta con el proveedor muebles Claudia y este está especializado en cocinas integrales por ende no podrá proveer productos diferentes a estos.
- La empresa solo almacenará información personal básica pero omite la fecha de nacimiento dado a que no maneja sistema de recompensas por cumpleaños, también la dirección de residencia solo se podrá ingresar a la hora de realizar un pedido.
- La empresa por medio de su app solo podrá hacer el reembolso si el pago ha sido realizado si solo si por medio de PayPal.
- Al momento de que el cliente se registre en el sistema su contraseña será generada automáticamente por el sistema donde después podrá cambiarla.
- Al momento de realizar un pedido se enviará un correo electrónico notificando de dicho pago.

La empresa destaca que, su página web principal debe de tener todo lo necesario para llamar la atención del cliente, esta debe de tener diseños únicos y elegantes a su vez, para garantizar el acceso a dicho portal la empresa contrató un servicio de hosting en el cual podrán poner su página web en línea.

Para concluir, la empresa dentro de sus políticas resalta que el cliente no podrá modificar su información personal básica, tales como cédula, nombres e información de origen, solo podrá hacer esto si se contacta con un administrador, de cualquier manera el usuario podrá editar el resto de su información y hacerlo si quiere.

Diagrama modelo entidad relación

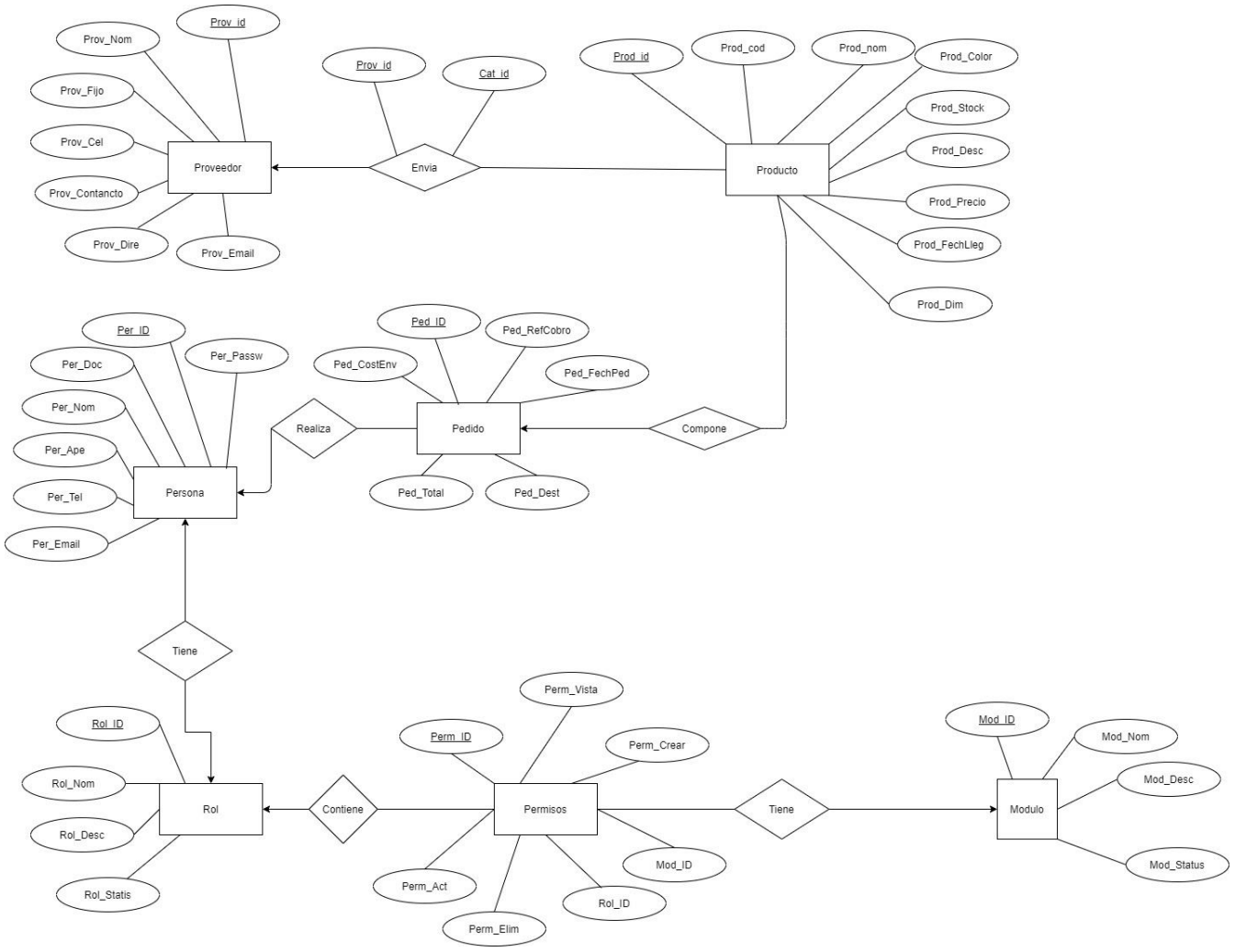
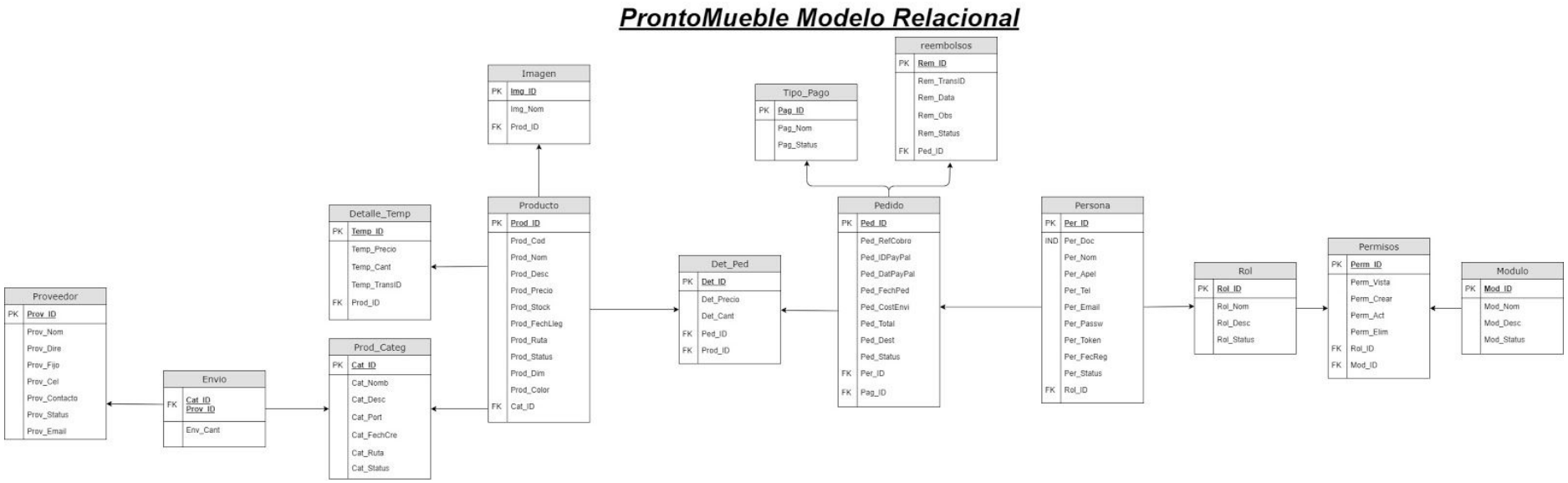


Diagrama del modelo relacional



Nota: En caso de no poder ver bien estos diagramas, este se podrá encontrar junto el código en el siguiente repositorio:https://github.com/xCopyDogsx/ProntoMueble

Diccionario de datos

A) Descripción de tablas y columnas.

Tabla Detalle_Temp (O detalle temporal):

Descripción:
La funcionalidad principal de esta tabla es permitir que se guarden algunos datos importantes de las transacciones que se realicen dentro de la aplicación tales como el precio del envío más el producto, así mismo como la cantidad y los productos asociados a esta transacción.

Detalle:

Columna	Tipo de dato	PK	FK	NULL	Descripción
Temp_ID	BIGINT(20) UNSIGNED	**YES**	**NO**	**NO**	Este va a ser el item que me permita identificar las transacciones dentro del sistema.
Tem_Precio	DECIMAL(11,2)	**NO**	**NO**	**NO**	Este campo sirve para almacenar el precio temporal de la transacción.
Tem_Precio	INT(11)	**NO**	**NO**	**NO**	La finalidad es guardar el precio

					de la transacción.
Tem_TransID	VARCHAR(255)	**NO**	**NO**	**NO**	Su finalidad es guardar el ID único durante la transacción en el sistema.
Prod_ID	BIGINT(20) UNSIGNED	**NO**	**YES**	**NO**	El objetivo de este campo es relacionar la tabla del producto con el de las transacciones.

SQL:

```sql
CREATE TABLE `detalle_temp` (
  `Temp_ID` bigint(20) UNSIGNED NOT NULL,
  `Temp_Precio` decimal(11,2) NOT NULL,
  `Temp_Cant` int(11) NOT NULL,
  `Temp_TransID` varchar(255) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prod_ID` bigint(11) UNSIGNED DEFAULT NULL,
  `Per_ID` bigint(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Observaciones:
Esta tabla está muy sujeta a momentos específicos para almacenar la información, será utilizada cuando haya fallas en el servidor o con el cliente y se usará para almacenar detalles temporales de tipo volátil.

Tabla Det_Pedido (O detalle del pedido)

Descripción:
Esta tabla servirá para almacenar todos los datos de los pedidos con sus detalles respectivos tales como el precio, la cantidad y el producto asociado a este.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Det_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo sirve para identificar nuestra tabla.
Det_Precio	**DECIMAL(11,2)**	**NO**	**NO**	**NO**	La funcionalidad de este campo es almacenar el precio del pedido incluyendo el IVA.
Det_Cant	**INT(11)**	**NO**	**NO**	**NO**	Este campo tiene como finalidad almacenar la cantidad de productos que se llevarán.
Prod_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este campo es la relación con el la tabla de producto que sirve para referenciarlos.

```
SQL:
CREATE TABLE `det_ped` (
  `Det_ID` bigint(20) UNSIGNED NOT NULL,
  `Det_Precio` decimal(11,2) NOT NULL,
  `Det_Cant` int(11) NOT NULL,
  `Ped_ID` bigint(20) UNSIGNED DEFAULT NULL,
  `Prod_ID` bigint(20) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Tabla Envio

Descripción:
Esta tabla tiene como finalidad almacenar los envíos que hacen los proveedores a la empresa, como se definió anteriormente los productos que se proveen son de un número limitado de proveedores especializados en dicha categoría.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Cat_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este campo ayuda a relacionar la tabla con la categoría de productos.
Prov_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este campo ayuda a relacionar la tabla con los proveedores.
Env_Cant	**INT(11)**	**NO**	**NO**	**NO**	Este campo ayuda a almacenar las cantidades suministradas por el proveedor.

```
SQL: CREATE TABLE `envio` (
  `Cat_ID` bigint(20) UNSIGNED NOT NULL,
  `Prov_ID` bigint(20) UNSIGNED NOT NULL,
```

```
  `Env_Cant` int(10) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Tabla imagen:

Descripción:
Esta tabla tiene como finalidad evitar la sobrecarga de código a la hora de buscar una imagen para un producto, almacenando los datos necesarios para que estos puedan ser encontrados de manera más sencilla, con el nombre del archivo para el compilador será más sencillo procesar la búsqueda en los directorios.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Img_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo sirve para almacenar el índice que identificará la tabla.
Img_Nom	**VARCHAR(100)**	**NO**	**NO**	**NO**	Acá se almacenará todo el nombre del archivo que relaciona la imagen del producto.
Prod_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	En este apartado se almacenará la relación con el producto en cuestión.

```sql
SQL:
CREATE TABLE `imagen` (
  `Img_ID` bigint(20) UNSIGNED NOT NULL,
  `Img_Nom` varchar(100) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prod_ID` bigint(11) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Tabla módulo:

Descripción:
Para la empresa es importante tener al día los distintos sectores de la empresa y de la aplicación para así a su vez entender cómo los módulos se dividen, por eso se da nacimiento a esta tabla para guardar dicha información actualmente solo se tienen 7 módulos para la empresa donde estan Dashboard(Panel principal), Usuarios,Clientes,Productos,Pedidos,Categorías(De los productos) y proveedores.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Mod_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo sirve para identificar la tabla de los módulos para relacionar con otras.
Mod_Nom	**VARCHAR(50)**	**NO**	**NO**	**NO**	Acá se almacenará todo lo relacionado con el nombre de los distintos módulos de la base de datos.
Mod_Desc	**TEXT**	**NO**	**NO**	**NO**	Al igual que el anterior el sistema acá permitirá guardar una breve descripción acerca del módulo.

Mod_Status	**INT(11)**	**NO**	**NO**	**NO**	Este campo sirve para almacenar si dicho módulo está activo o inactivo en el sistema.

```
SQL:
CREATE TABLE Modulo(
Mod_ID BIGINT(20) UNSIGNED AUTO_INCREMENT NOT NULL,
      Mod_Nom VARCHAR(50) NOT NULL,
      Mod_Desc TEXT NOT NULL,
      Mod_Status INT(11) DEFAULT 1,
      PRIMARY KEY(Mod_ID)
            )ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Observaciones:
En el campo del status a la hora de insertar el dato, este tomará por defecto un valor de 1 a menos de que se exprese lo contrario.

Tabla pedido:

Descripción:
En esta tabla se procederá a guardar toda la información sobre el pedido que realice un cliente con el debido estado sobre este.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Ped_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo sirve para identificar la tabla de los pedidos para relacionar con otras.
Ped_RefCobro	**varchar(255)**	**NO**	**NO**	**YES**	Este campo se usa para almacenar las referencias de cobro en caso de que el pedido sea por medio de contraentrega.
Ped_IDPayPal	**varchar(255)**	**NO**	**NO**	**YES**	Este campo es utilizado para guardar el ID que entrega el API de PayPal a la hora de realizar el pago por medio de esta.

Ped_DatPayPal	**TEXT**	**NO**	**NO**	**YES**	Este campo es utilizado para almacenar todo el JSON que se genera a la hora de hacer peticiones al Api de PayPal.
Ped_CostEnv	**decimal(10,2)**	**NO**	**NO**	**NO**	Este campo será utilizado para almacenar el costo de envío de cada ítem para realizar la facturación.
Ped_Total	**decimal(10,2)**	**NO**	**NO**	**NO**	Acá se almacenará todo el total de la compra a la hora de realizar el pago.
Ped_Dest	**TEXT**	**NO**	**NO**	**NO**	En este apartado se guardará toda la información acerca del destino del pedido para que la compañía de mensajería haga el envío

Ped_Status	**varchar(100)**	**NO**	**NO**	**NO**	Acá se podrá almacenar el estado del envío en caso de que sea pendiente, aprobado o completado.
Per_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este es el campo utilizado para asociar a la persona con su pedido y toda la información de este.
Pag_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este es el campo utilizado para asociar al tipo de pago usado por la persona a la hora de realizar la compra en la tienda de prontomueble
Ped_FechPed	**Timestamp**	**NO**	**NO**	**NO**	Este campo almacena la fecha del pedido según la hora que se inserte en la base

					de datos.

SQL:

```sql
CREATE TABLE `pedido` (
  `Ped_ID` bigint(20) UNSIGNED NOT NULL,
  `Ped_RefCobro` varchar(255) COLLATE utf8mb4_swedish_ci DEFAULT NULL,
  `Ped_IDPayPal` varchar(255) COLLATE utf8mb4_swedish_ci DEFAULT NULL,
  `Ped_DatPayPal` text COLLATE utf8mb4_swedish_ci DEFAULT NULL,
  `Ped_FechPed` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `Ped_CostEnv` decimal(10,2) NOT NULL DEFAULT 0.00,
  `Ped_Total` decimal(11,2) NOT NULL,
  `Ped_Dest` text COLLATE utf8mb4_swedish_ci NOT NULL,
  `Ped_Status` varchar(100) COLLATE utf8mb4_swedish_ci DEFAULT NULL,
  `Per_ID` bigint(20) UNSIGNED DEFAULT NULL,
  `Pag_ID` bigint(20) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Observación:
La empresa tiene por norma utilizar el costo de envío para cualquier pedido en $15.000(COP) en caso de no ser especificado el costo de envío será asumido en $0.0 por la base de datos.

Tabla permisos:

Descripción:
Esta tabla será usada para almacenar los permisos de los distintos usuarios en las plataforma, dado a que la plataforma por el momento cuenta con 7 módulos (Dashboard,Usuarios,etc) estos deberán tener permisos un ejemplo de esto es que un cliente solo podrá acceder al módulo de pedidos pero no podrá tener permisos de escritura,actualización y eliminación solo de lectura, de esta manera mediante enteros se maneja el tema del permiso asociado a cada módulo siendo el 1 el estado de permitido y 0 (o null) el estado de denegado.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Perm_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este es el campo de identificación del permiso que sirve para relacionarlo tanto con la tabla módulos y roles.
Perm_Vista	**Int(11)**	**NO**	**NO**	**YES**	Este campo asocia el permiso de lectura que puede tener determinado rol sobre X módulo.
Perm_Crear	**Int(11)**	**NO**	**NO**	**YES**	Este campo asocia el permiso de crear (o agregar algo nuevo) que puede

					tener determinado rol sobre X módulo.
Perm_Act	**Int(11)**	**NO**	**NO**	**YES**	Este campo asocia el permiso de cambiar cosas que puede tener determinado rol sobre X módulo.
Perm_Elim	**Int(11)**	**NO**	**NO**	**YES**	Este campo asocia el permiso de eliminar que puede tener determinado rol sobre X módulo.
Rol_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este es el campo de asociación que nos dice que permiso tiene cada rol dentro de la empresa bien sea de lectura o demás.
Mod_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este campo relaciona los permisos sobre dichos módulos de

					la empresa o aplicación

```
SQL:
CREATE TABLE `permisos` (
  `Perm_ID` bigint(20) UNSIGNED NOT NULL,
  `Perm_Vista` int(11) DEFAULT NULL,
  `Perm_Crear` int(11) DEFAULT NULL,
  `Perm_Act` int(11) DEFAULT NULL,
  `Perm_Elim` int(11) DEFAULT NULL,
  `Rol_ID` bigint(10) UNSIGNED DEFAULT NULL,
  `Mod_ID` bigint(20) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Tabla persona:

Descripción: En esta tabla se procederá a almacenar la información personal de los clientes y de los empleados de la empresa con todo lo relacionado a ellos tanto como sus roles y permisos.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Per_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este será el campo usado a la hora de identificar a una persona en el sistema y en la base de datos en general.
Per_Doc	**Varchar(15)**	**NO**	**NO**	**NO**	Este campo es usado para guardar el documento de identidad de la persona (Además de ser un índice para evitar repetidos).
Per_Nom	**Varchar(50)**	**NO**	**NO**	**NO**	Acá se almacenarán los nombres de la persona en caso de que tenga dos o

					uno.
Per_Ape	**Varchar(50)**	**NO**	**NO**	**NO**	Este campo será utilizado para almacenar los apellidos de las personas o usuarios del sistema.
Per_Tel	**Varchar(30)**	**NO**	**NO**	**NO**	Acá se almacenará el teléfono de la persona sin importar que sea fijo o celular.
Per_Email	**Varchar(100)**	**NO**	**NO**	**NO**	Este será el identificador de nuestro usuario a la hora de logear dado a que acá se almacena el email de la persona y es un índice ya que no hay dos correos iguales.
Per_Passwd	**Varchar(75)**	**NO**	**NO**	**NO**	Acá se almacenará la contraseña de la persona en el sistema encriptada

					por medio de AES-128-ECB.
Per_Toke	**Varchar(100)**	**NO**	**NO**	**YES**	Dado a que el sistema generará una contraseña al usuario a la hora de registrarse creará un token que será enviado al correo para que la cambie por motivos de seguridad.
Per_FecReg	**Timestamp**	**NO**	**NO**	**NO**	En este espacio se almacenará la fecha en la que se registró la persona en el sistema de manera automática.
Per_Status	**Int(11)**	**NO**	**NO**	**NO**	Este campo indica el estado de la persona en el sistema en caso de estar inactiva no podrá logear en el y realizar todo lo que un usuario

					normal puede realizar en el sistema.
Rol_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Esta llave foránea sirve para asociar el rol que puede tener determinada persona en el sistema bien sea un administrador y demás podrá hacer uso de este campo para asociar también los permisos de su rol en el sistema.

```
SQL:
CREATE TABLE `persona` (
  `Per_ID` bigint(20) UNSIGNED NOT NULL,
  `Per_Doc` varchar(15) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Per_Nom` varchar(50) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Per_Ape` varchar(50) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Per_Tel` varchar(30) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Per_Email` varchar(100) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Per_Passw` varchar(75) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Per_Toke` varchar(100) COLLATE utf8mb4_swedish_ci DEFAULT NULL,
```

```
  `Per_FecReg` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `Per_Status` int(11) DEFAULT 1,
  `Rol_ID` bigint(20) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Observaciones: Cada vez que se cree un usuario este tendrá un estado de activo por defecto siendo el 1 el valor que represente esto y el 0 el valor que representa lo inactivo.

Tabla producto:

Descripción: Esta tabla será usada para almacenar todos los datos referentes a los productos ofrecidos por la empresa.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Prod_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo será el identificador del producto en la base de datos, será el que nos ayude a relacionar con las demás tablas.
Prod_Cod	**Varchar(30)**	**NO**	**NO**	**NO**	Acá se almacenará el código de barras del producto para efectos de inventario de la empresa.
Prod_Nom	**Varchar(255)**	**NO**	**NO**	**NO**	Este campo almacena el nombre del producto entero ya sea largo o corto no importa por el tamaño del campo.

Prod_Desc	**Text**	**NO**	**NO**	**YES**	En este campo se guarda todo lo relacionado a la descripción del producto.
Prod_Precio	**Decimal(11,2)**	**NO**	**NO**	**NO**	Este campo almacenará el precio del producto.
Prod_Stock	**Int(11)**	**NO**	**NO**	**NO**	Acá se almacenará la cantidad del producto que hay en la tienda o también conocido como stock.
Prod_FecLleg	**Timestamp**	**NO**	**NO**	**NO**	En este campo se almacenará todo lo relacionado al día en que este producto entró a la empresa.
Prod_Ruta	**Varchar(255)**	**NO**	**NO**	**NO**	Dado a que PHP trabaja con URL se debe almacenar la ruta para encontrar este producto, este campo es la

					solución a dicho problema.
Prod_Status	**Int(11)**	**NO**	**NO**	**NO**	Este campo nos indica si el producto está activo o inactivo en el almacén.
Prod_Dim	**Varchar(80)**	**NO**	**NO**	**NO**	En este apartado se almacenará todo lo relacionado a las dimensiones del producto.
Prod_Color	**Varchar(80)**	**NO**	**NO**	**NO**	Acá se guarda el color del producto y demás.
Cat_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este campo categoriza el producto en su respectiva categoría gracias a la relación que hay.

```
SQL:
CREATE TABLE `producto` (
  `Prod_ID` bigint(20) UNSIGNED NOT NULL,
```

```
  `Prod_Cod` varchar(30) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prod_Nom` varchar(255) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prod_Desc` text COLLATE utf8mb4_swedish_ci DEFAULT NULL,
  `Prod_Precio` decimal(11,2) NOT NULL,
  `Prod_Stock` int(11) NOT NULL,
  `Prod_FecLleg` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `Prod_Ruta` varchar(255) COLLATE utf8mb4_swedish_ci DEFAULT NULL,
  `Prod_Status` int(11) NOT NULL DEFAULT 1,
  `Prod_Dim` varchar(80) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prod_Color` varchar(80) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_ID` bigint(11) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Tabla prod_cat (O categoría del producto):

Descripción: Dado a que los productos pertenecen a determinada categoría, está table nace para evitar toda esa información dentro de la tabla producto.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Cat_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo será la identificación de la categoría del producto en la base de datos que ayuda a las relaciones con otras tablas.
Cat_Nom	**Varchar(255)**	**NO**	**NO**	**NO**	Acá se almacenará el nombre de la categoría del producto.
Cat_Des	**Text**	**NO**	**NO**	**NO**	Este campo guarda la descripción de la categoría del producto que se mostrará después en el banner.

Cat_Port	**Varchar(100)**	**NO**	**NO**	**NO**	Dado a que el sistema se maneja por categorías cada una de estas tiene una imagen de portada que se mostrará en la tienda.
Cat_FecCre	**Timestamp**	**NO**	**NO**	**NO**	Acá se almacenará la fecha en que esta categoría fue creada en el sistema.
Cat_Ruta	**Varchar(255)**	**NO**	**NO**	**NO**	Dado a que PHP trabaja con URL se debe almacenar la ruta para encontrar esta categoría, este campo es la solución a dicho problema.
Cat_Status	**Int(11)**	**NO**	**NO**	**NO**	Este campo indica si la categoría dentro del almacén está activa o inactiva.

```
SQL:
CREATE TABLE `prod_cat` (
  `Cat_ID` bigint(10) UNSIGNED NOT NULL,
  `Cat_Nom` varchar(255) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_Des` text COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_Port` varchar(100) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_FecCre` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `Cat_Ruta` varchar(255) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_Status` int(11) DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Tabla proveedor:

Descripción: Esta tabla sirve para almacenar los datos del proveedor y toda su información de contacto.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Prov_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo es el identificador del proveedor en la base de datos.
Prov_Nom	**Varchar(255)**	**NO**	**NO**	**NO**	Este campo almacenará el nombre del proveedor.
Prov_Dir	**Text**	**NO**	**NO**	**NO**	Este campo almacenará la dirección del proveedor de la oficina principal.
Prov_Fijo	**Varchar(20)**	**NO**	**NO**	**NO**	Este campo almacenará el teléfono fijo del proveedor.
Prov_Cel	**Varchar(20)**	**NO**	**NO**	**NO**	Acá se guardará el número celular del proveedor.

Prov_Status	**Int(11)**	**NO**	**NO**	**NO**	Esto indica si el proveedor está activo o inactivo en el sistema.
Prov_Contacto	**Varchar(90)**	**NO**	**NO**	**NO**	Este campo almacena el nombre de la persona de contacto del proveedor.
Prov_Email	**Varchar(100)**	**NO**	**NO**	**NO**	Este campo almacena el email del proveedor.

SQL:

```sql
CREATE TABLE `proveedor` (
  `Prov_ID` bigint(20) UNSIGNED NOT NULL,
  `Prov_Nom` varchar(255) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Dir` text COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Fijo` varchar(20) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Cel` varchar(20) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Status` int(11) DEFAULT 1,
  `Prov_Contacto` varchar(90) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Email` varchar(100) COLLATE utf8mb4_swedish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Tabla reembolsos:

Descripción: Esta tabla solo almacenará los reembolsos que se hagan por medio de la api de PayPal dado a que se pueden generar ID y objetos JSON para verificar la veracidad de la transacción.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Rem_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo identifica el reembolso dentro de la base de datos.
Rem_TransID	**Varchar(255)**	**NO**	**NO**	**NO**	Este campo guarda el ID PayPal del reembolso.
Rem_Data	**Text**	**NO**	**NO**	**NO**	Este campo almacena el JSON sobre el reembolso realizado al cliente.
Rem_Obs	**Text**	**NO**	**NO**	**NO**	Acá se almacena el porqué se realizó el reembolso en la compañía.

Rem_Status	**Varchar(150)**	**NO**	**NO**	**NO**	En este campo se almacena el estado de la transacción en la api de PayPal.
Ped_ID	**BIGINT(20) UNSIGNED**	**NO**	**YES**	**NO**	Este campo relaciona el pedido que se reembolso y que se modificó.

```
SQL:
CREATE TABLE `reembolsos` (
  `Rem_ID` bigint(20) NOT NULL,
  `Rem_TransID` varchar(255) CHARACTER SET utf8mb4 NOT NULL,
  `Rem_Data` text CHARACTER SET utf8mb4 NOT NULL,
  `Rem_Obs` text CHARACTER SET utf8mb4 NOT NULL,
  `Rem_Status` varchar(150) CHARACTER SET utf8mb4 NOT NULL,
  `Ped_ID` bigint(20) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Tabla rol:

Descripción: Esta tabla tiene como finalidad almacenar los roles de los usuarios del sistema y de todos los nuevos usuarios.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Rol_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo sirve para identificar el rol dentro de la base de datos y así poder relacionarlo.
Rol_Nom	**Varchar(50)**	**NO**	**NO**	**NO**	Este campo almacena el nombre del rol ej: Administrador.
Rol_Desc	**Text**	**NO**	**NO**	**NO**	Este campo almacena la descripción del rol.
Rol_Status	**Int(11)**	**NO**	**NO**	**NO**	Este campo almacenará si el rol en el sistema está activo o inactivo.

```
SQL:
CREATE TABLE `rol` (
  `Rol_ID` bigint(20) UNSIGNED NOT NULL,
  `Rol_Nom` varchar(50) COLLATE utf8mb4_swedish_ci DEFAULT NULL,
  `Rol_Desc` text COLLATE utf8mb4_swedish_ci DEFAULT NULL,
  `Rol_Status` int(11) DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

Tabla tipo_pago (Medios de pago):

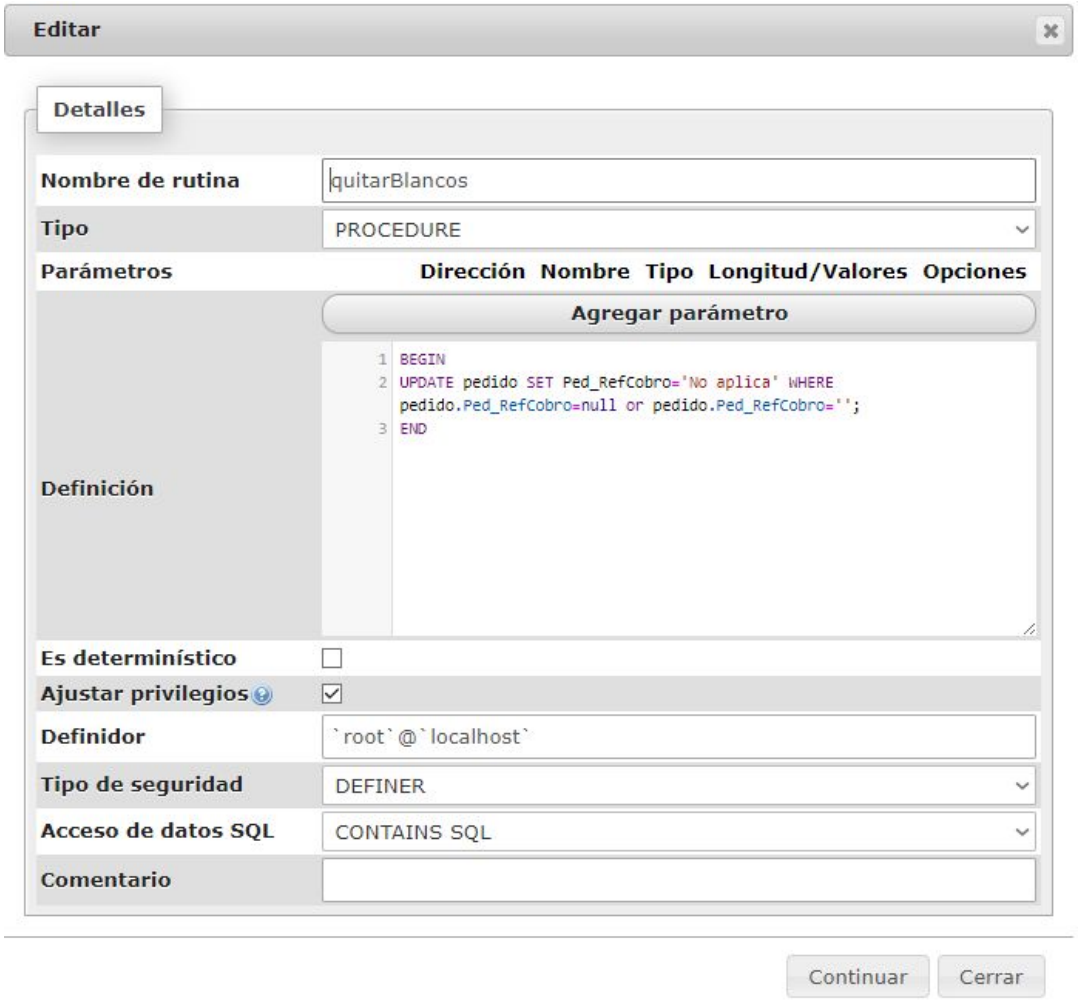
Descripción: Esta tabla tiene como finalidad almacenar los medios de pago de la empresa y los que acepta a la hora de realizar el pago.

Columna	Tipo de dato	PK	FK	NULL	Descripción
Rol_ID	**BIGINT(20) UNSIGNED**	**YES**	**NO**	**NO**	Este campo tiene como finalidad identificar el medio de pago en la base de datos.
Pag_Nom	**Varchar(100)**	**NO**	**NO**	**NO**	Este campo almacena el nombre del método de pago.
Pag_Status	**Int(11)**	**NO**	**NO**	**NO**	Este campo indicará si el método de pago es aceptado o no en la empresa.

```
SQL:
CREATE TABLE `tipo_pago` (
  `Pag_ID` bigint(20) UNSIGNED NOT NULL,
  `Pag_Nom` varchar(100) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Pag_Status` int(11) NOT NULL DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_swedish_ci;
```

B) Descripción de reglas.

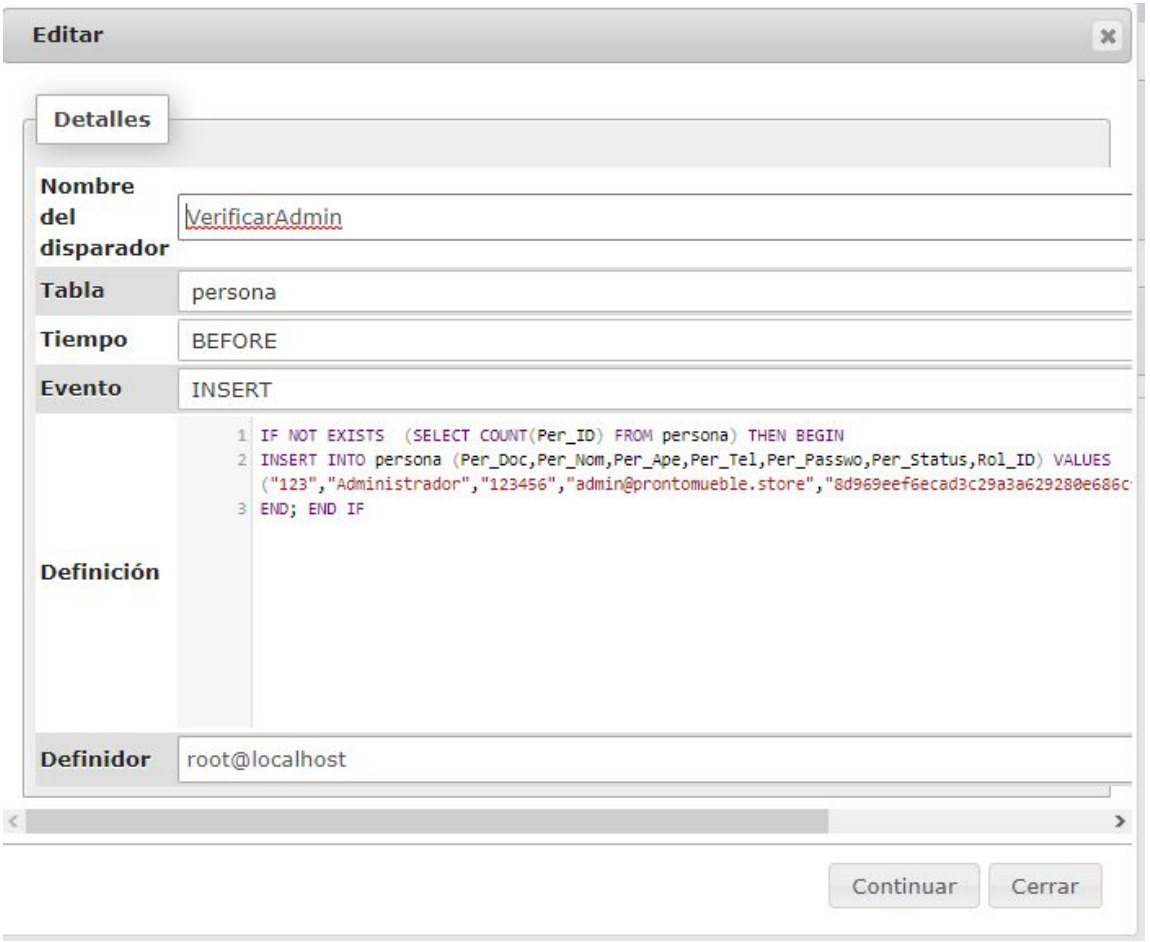
Dado a que MySQL es un motor limitado en temas de creación de reglas y no permite crearlas, se creó una rutina almacenada que a diferencia de las reglas en postgreSQL se deben llamar mediante la palabra reservada CALL, esta rutina que se creó permite quitar los espacios en blanco cada vez que se ingresa un nuevo valor en la tabla pedidos para el campo de referencia de cobros, dado a que el sistema no contempla referencias de cobro a la hora de hacer un pedido de contra entrega, el código SQL es el siguiente:




```
DELIMITER $$
CREATE PROCEDURE quitarBlancos()
BEGIN
UPDATE pedido SET Ped_RefCobro='No aplica' WHERE
pedido.Ped_RefCobro=null or pedido.Ped_RefCobro='';
END$$
DELIMITER ;
```

C) Descripción de procedimientos y triggers.

Para este apartado se consideró utilizar un trigger momentáneo el cual consiste en llenar un campo de la tabla personas si esta está vacía generando así nuestro usuario con privilegios máximos, dicho procedimiento o trigger es el siguiente:



Siendo el código SQL el siguiente:

```
CREATE TRIGGER `VerificarAdmin` BEFORE INSERT ON `persona`
 FOR EACH ROW IF NOT EXISTS  (SELECT COUNT(Per_ID) FROM persona)
THEN BEGIN
INSERT INTO persona
(Per_Doc,Per_Nom,Per_Ape,Per_Tel,Per_Passwo,Per_Status,Rol_ID)
VALUES
("123","Administrador","123456","admin@prontomueble.store","8d969e
ef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92",1,1);
END; END IF
```

E) Descripción de vistas.

Para el diseño de vistas en este caso solo se usaron para generar estadísticas y evitar que el motor de la base de datos se cargue con consultas que pueden ser almacenadas en las vistas y que solo sean consultadas a la hora de requerirse con un simple select, además de ahorrar memoria a la hora de procesar dicha información, por eso las vistas son las siguientes:

1. **Vista estadisticas-cliente:**

Esta vista almacenará la cantidad de clientes que se han registrado en el sistema y el código SQL es el siguiente:

```
CREATE VIEW `estadisiticascliente`  AS SELECT count(0) AS
`COUNT(*)` FROM `persona` WHERE `persona`.`Rol_ID` = 7 ;
```

Resultado:

COUNT(*)
3

2. **Vista estadisticas-empleado:**

Esta vista almacenará la cantidad de empleados que se han registrado en el sistema y el código SQL es el siguiente:

```
CREATE VIEW `estadisticasempleados`  AS SELECT count(0) AS
`COUNT(*)` FROM `persona` WHERE `persona`.`Rol_ID` <> 7 ;
```

Resultado:

COUNT(*)
1

3. **Vista ventas:**

Esta vista almacenará todas las ventas completas que se han generado a lo largo del tiempo, su código SQL es el siguiente:

```
CREATE VIEW `ventas`  AS SELECT count(0) AS `COUNT(*)` FROM
`pedido` WHERE `pedido`.`Ped_Status` = 'Completado' OR
`pedido`.`Ped_Status` <> 'Pendiente' ;
```

Resultado:

COUNT(*)
2

4. **Vista ventas-pendientes:**

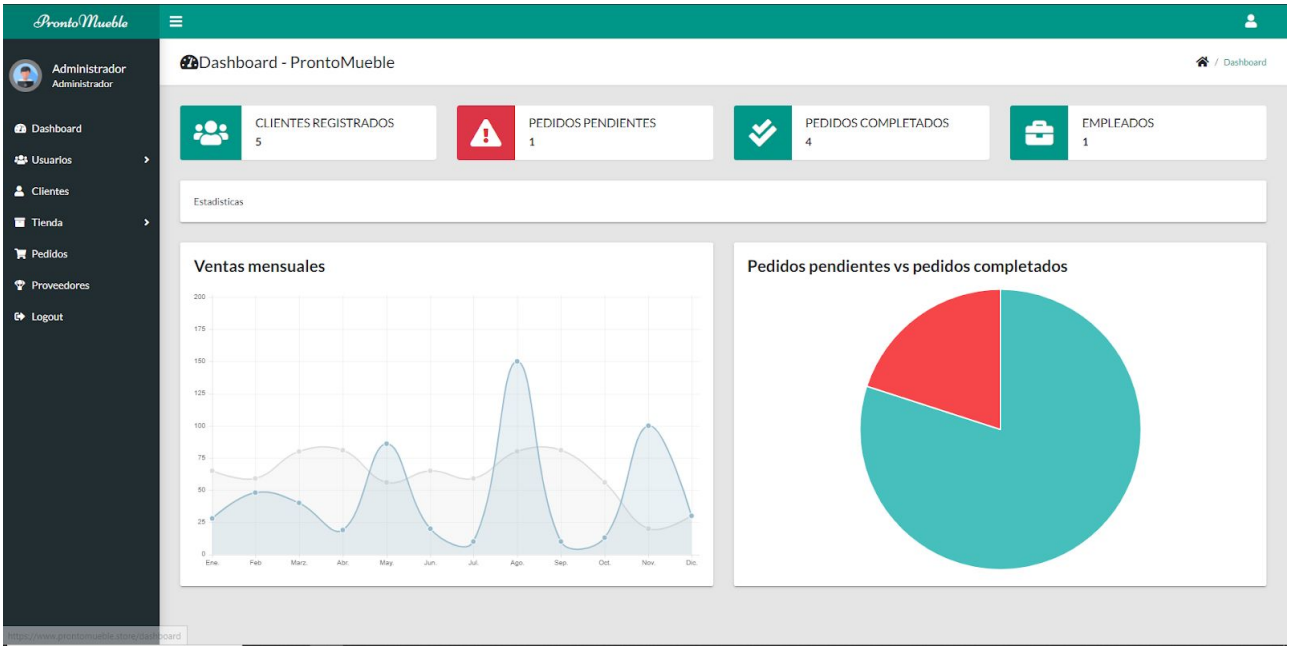
Esta vista tiene como finalidad almacenar las ventas que no se han completado y tienen un estado de pendiente en la tienda, el código SQL es el siguiente:

```
CREATE VIEW `ventaspendientes`  AS SELECT count(0) AS `COUNT(*)`
FROM `pedido` WHERE `pedido`.`Ped_Status` = 'Pendiente' ;
```

Resultado:

COUNT(*)
0

El resultado de estas vistas se podrá apreciar en la siguiente vista del sistema:



Código fuente

A) Estructura de la base de datos.

Para expresar lo visto en el diagrama relacional, el código será el siguiente (Solo se incluye el DDL para la creación de las tablas necesarias y sus debidas relaciones):

```
/*-----------DDL de las tablas ---------------*/

--
-- Estructura de tabla para la tabla `detalle_temp`
--

CREATE TABLE `detalle_temp` (
  `Temp_ID` bigint(20) UNSIGNED NOT NULL,
  `Temp_Precio` decimal(11,2) NOT NULL,
  `Temp_Cant` int(11) NOT NULL,
  `Temp_TransID` varchar(255) NOT NULL,
  `Prod_ID` bigint(11) UNSIGNED NOT NULL,
  `Per_ID` bigint(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;

--
-- Estructura de tabla para la tabla `det_ped`
--

CREATE TABLE `det_ped` (
  `Det_ID` bigint(20) UNSIGNED NOT NULL,
  `Det_Precio` decimal(11,2) NOT NULL,
  `Det_Cant` int(11) NOT NULL,
  `Ped_ID` bigint(20) UNSIGNED NOT NULL,
  `Prod_ID` bigint(20) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;

--
-- Estructura de tabla para la tabla `envio`
--
```

```sql
CREATE TABLE `envio` (
  `Cat_ID` bigint(20) UNSIGNED NOT NULL,
  `Prov_ID` bigint(20) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Estructura de tabla para la tabla `imagen`
--

CREATE TABLE `imagen` (
  `Img_ID` bigint(20) UNSIGNED NOT NULL,
  `Img_Nom` varchar(100) NOT NULL,
  `Prod_ID` bigint(11) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;

--
-- Estructura de tabla para la tabla `modulo`
--

CREATE TABLE `modulo` (
  `Mod_ID` bigint(20) UNSIGNED NOT NULL,
  `Mod_Nom` varchar(50) NOT NULL,
  `Mod_Desc` text NOT NULL,
  `Mod_Status` int(11) DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;

--
-- Estructura de tabla para la tabla `pedido`
--

CREATE TABLE `pedido` (
  `Ped_ID` bigint(20) UNSIGNED NOT NULL,
  `Ped_RefCobro` varchar(255) DEFAULT NULL,
  `Ped_IDPayPal` varchar(255) DEFAULT NULL,
  `Ped_DatPayPal` text DEFAULT NULL,
  `Ped_FechPed` timestamp NOT NULL DEFAULT current_timestamp() ON
UPDATE current_timestamp(),
  `Ped_CostEnv` decimal(10,2) NOT NULL DEFAULT 0.00,
  `Ped_Total` decimal(11,2) NOT NULL,
  `Ped_Dest` text NOT NULL,
```

```
  `Ped_Status` varchar(100) DEFAULT NULL,
  `Per_ID` bigint(20) UNSIGNED DEFAULT NULL,
  `Pag_ID` bigint(20) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;

--
-- Estructura de tabla para la tabla `permisos`
--

CREATE TABLE `permisos` (
  `Perm_ID` bigint(20) UNSIGNED NOT NULL,
  `Perm_Vista` int(11) DEFAULT NULL,
  `Perm_Crear` int(11) DEFAULT NULL,
  `Perm_Act` int(11) DEFAULT NULL,
  `Perm_Elim` int(11) DEFAULT NULL,
  `Rol_ID` bigint(10) UNSIGNED DEFAULT NULL,
  `Mod_ID` bigint(20) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;

--
-- Estructura de tabla para la tabla `persona`
--

CREATE TABLE `persona` (
  `Per_ID` bigint(20) UNSIGNED NOT NULL,
  `Per_Doc` varchar(15) NOT NULL,
  `Per_Nom` varchar(50) NOT NULL,
  `Per_Ape` varchar(50) NOT NULL,
  `Per_Tel` varchar(30) NOT NULL,
  `Per_Email` varchar(100) NOT NULL,
  `Per_Passw` varchar(75) NOT NULL,
  `Per_Toke` varchar(100) DEFAULT NULL,
  `Per_FecNac` date DEFAULT NULL,
  `Per_FecReg` timestamp NOT NULL DEFAULT current_timestamp() ON
UPDATE current_timestamp(),
  `Per_Status` int(11) DEFAULT 1,
  `Rol_ID` bigint(20) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;
```

```sql
--
-- Estructura de tabla para la tabla `producto`
--

CREATE TABLE `producto` (
  `Prod_ID` bigint(20) UNSIGNED NOT NULL,
  `Prod_Cod` varchar(30) NOT NULL,
  `Prod_Nom` varchar(255) NOT NULL,
  `Prod_Desc` text DEFAULT NULL,
  `Prod_Precio` decimal(11,2) NOT NULL,
  `Prod_Stock` int(11) NOT NULL,
  `Prod_FecLleg` timestamp NOT NULL DEFAULT current_timestamp() ON
UPDATE current_timestamp(),
  `Prod_Ruta` varchar(255) DEFAULT NULL,
  `Prod_Status` int(11) NOT NULL DEFAULT 1,
  `Prod_Dim` varchar(80) NOT NULL,
  `Prod_Color` varchar(80) NOT NULL,
  `Cat_ID` bigint(11) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;
--
-- Estructura de tabla para la tabla `prod_cat`
--

CREATE TABLE `prod_cat` (
  `Cat_ID` bigint(10) UNSIGNED NOT NULL,
  `Cat_Nom` varchar(255) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_Des` text COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_Port` varchar(100) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_FecCre` timestamp NOT NULL DEFAULT current_timestamp() ON
UPDATE current_timestamp(),
  `Cat_Ruta` varchar(255) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Cat_Status` int(11) DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;
--
-- Estructura de tabla para la tabla `proveedor`
--

CREATE TABLE `proveedor` (
  `Prov_ID` bigint(20) UNSIGNED NOT NULL,
  `Prov_Nom` varchar(255) COLLATE utf8mb4_swedish_ci NOT NULL,
```

```sql
  `Prov_Dir` text COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Fijo` varchar(20) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Cel` varchar(20) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Status` int(11) DEFAULT 1,
  `Prov_Contacto` varchar(90) COLLATE utf8mb4_swedish_ci NOT NULL,
  `Prov_Email` varchar(100) COLLATE utf8mb4_swedish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;
--
-- Estructura de tabla para la tabla `reembolsos`
--

CREATE TABLE `reembolsos` (
  `Rem_ID` bigint(20) NOT NULL,
  `Rem_TransID` varchar(255) CHARACTER SET utf8mb4 NOT NULL,
  `Rem_Data` text CHARACTER SET utf8mb4 NOT NULL,
  `Rem_Obs` text CHARACTER SET utf8mb4 NOT NULL,
  `Rem_Status` varchar(150) CHARACTER SET utf8mb4 NOT NULL,
  `Ped_ID` bigint(20) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;

--
-- Estructura de tabla para la tabla `rol`
--

CREATE TABLE `rol` (
  `Rol_ID` bigint(20) UNSIGNED NOT NULL,
  `Rol_Nom` varchar(50) NOT NULL,
  `Rol_Desc` text  NOT NULL,
  `Rol_Status` int(11) DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_swedish_ci;
--
-- Estructura de tabla para la tabla `tipo_pago`
--

CREATE TABLE `tipo_pago` (
  `Pag_ID` bigint(20) UNSIGNED NOT NULL,
  `Pag_Nom` varchar(100) NOT NULL,
  `Pag_Status` int(11) NOT NULL DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

```sql
COLLATE=utf8mb4_swedish_ci;


/*---------------Indices para las tablas----------------*/
--
-- Indices de la tabla `detalle_temp`
--
ALTER TABLE `detalle_temp`
  ADD PRIMARY KEY (`Temp_ID`),
  ADD KEY `detalle_temp_ibfk_1` (`Prod_ID`);

--
-- Indices de la tabla `det_ped`
--
ALTER TABLE `det_ped`
  ADD PRIMARY KEY (`Det_ID`),
  ADD KEY `det_ped_ibfk_1` (`Ped_ID`),
  ADD KEY `det_ped_ibfk_2` (`Prod_ID`);

--
-- Indices de la tabla `envio`
--
ALTER TABLE `envio`
  ADD KEY `envio_ibfk_1` (`Cat_ID`),
  ADD KEY `envio_ibfk_2` (`Prov_ID`);

--
-- Indices de la tabla `imagen`
--
ALTER TABLE `imagen`
  ADD PRIMARY KEY (`Img_ID`),
  ADD KEY `imagen_ibfk_1` (`Prod_ID`);

--
-- Indices de la tabla `modulo`
--
ALTER TABLE `modulo`
  ADD PRIMARY KEY (`Mod_ID`);

--
-- Indices de la tabla `pedido`
--
```

```sql
ALTER TABLE `pedido`
  ADD PRIMARY KEY (`Ped_ID`),
  ADD KEY `pedido_ibfk_1` (`Per_ID`),
  ADD KEY `pedido_ibfk_2` (`Pag_ID`);

--
-- Indices de la tabla `permisos`
--
ALTER TABLE `permisos`
  ADD PRIMARY KEY (`Perm_ID`),
  ADD KEY `permisos_ibfk_1` (`Rol_ID`),
  ADD KEY `permisos_ibfk_2` (`Mod_ID`);

--
-- Indices de la tabla `persona`
--
ALTER TABLE `persona`
  ADD PRIMARY KEY (`Per_ID`),
  ADD UNIQUE KEY `Per_Email` (`Per_Email`),
  ADD UNIQUE KEY `Per_Doc` (`Per_Doc`),
  ADD KEY `persona_ibfk_1` (`Rol_ID`),
  ADD KEY `Per_Tel` (`Per_Tel`);

--
-- Indices de la tabla `producto`
--
ALTER TABLE `producto`
  ADD PRIMARY KEY (`Prod_ID`),
  ADD KEY `producto_ibfk_1` (`Cat_ID`);

--
-- Indices de la tabla `prod_cat`
--
ALTER TABLE `prod_cat`
  ADD PRIMARY KEY (`Cat_ID`);

--
-- Indices de la tabla `proveedor`
--
ALTER TABLE `proveedor`
  ADD PRIMARY KEY (`Prov_ID`);
```

```sql
--
-- Indices de la tabla `reembolsos`
--
ALTER TABLE `reembolsos`
  ADD PRIMARY KEY (`Rem_ID`),
  ADD KEY `Ped_ID` (`Ped_ID`);
--
-- Indices de la tabla `rol`
--
ALTER TABLE `rol`
  ADD PRIMARY KEY (`Rol_ID`);

--
-- Indices de la tabla `tipo_pago`
--
ALTER TABLE `tipo_pago`
  ADD PRIMARY KEY (`Pag_ID`);

--
-- AUTO_INCREMENT de las tablas volcadas
--

--
-- AUTO_INCREMENT de la tabla `detalle_temp`
--
ALTER TABLE `detalle_temp`
  MODIFY `Temp_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `det_ped`
--
ALTER TABLE `det_ped`
  MODIFY `Det_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `imagen`
--
ALTER TABLE `imagen`
  MODIFY `Img_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `modulo`
```

```sql
--
ALTER TABLE `modulo`
  MODIFY `Mod_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `pedido`
--
ALTER TABLE `pedido`
  MODIFY `Ped_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `permisos`
--
ALTER TABLE `permisos`
  MODIFY `Perm_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `persona`
--
ALTER TABLE `persona`
  MODIFY `Per_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `producto`
--
ALTER TABLE `producto`
  MODIFY `Prod_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `prod_cat`
--
ALTER TABLE `prod_cat`
  MODIFY `Cat_ID` bigint(10) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `proveedor`
--
ALTER TABLE `proveedor`
  MODIFY `Prov_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `rol`
```

```sql
--
ALTER TABLE `rol`
  MODIFY `Rol_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `tipo_pago`
--
ALTER TABLE `tipo_pago`
  MODIFY `Pag_ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT;

/*------------Relaciones y llaves
foraneas-----------------------*/

--
-- Relaciones para la tabla `detalle_temp`
--
ALTER TABLE `detalle_temp`
  ADD CONSTRAINT `detalle_temp_ibfk_1` FOREIGN KEY (`Prod_ID`)
REFERENCES `producto` (`Prod_ID`) ON DELETE CASCADE ON UPDATE
CASCADE;

--
-- Relaciones para la tabla `det_ped`
--
ALTER TABLE `det_ped`
  ADD CONSTRAINT `det_ped_ibfk_1` FOREIGN KEY (`Ped_ID`)
REFERENCES `pedido` (`Ped_ID`) ON DELETE CASCADE ON UPDATE
CASCADE,
  ADD CONSTRAINT `det_ped_ibfk_2` FOREIGN KEY (`Prod_ID`)
REFERENCES `producto` (`Prod_ID`) ON DELETE CASCADE ON UPDATE
CASCADE;

--
-- Relaciones para la tabla `envio`
--
ALTER TABLE `envio`
  ADD CONSTRAINT `envio_ibfk_1` FOREIGN KEY (`Cat_ID`) REFERENCES
`prod_cat` (`Cat_ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `envio_ibfk_2` FOREIGN KEY (`Prov_ID`) REFERENCES
`proveedor` (`Prov_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
```

```sql
-- Relaciones para la tabla `imagen`
--
ALTER TABLE `imagen`
  ADD CONSTRAINT `imagen_ibfk_1` FOREIGN KEY (`Prod_ID`)
REFERENCES `producto` (`Prod_ID`) ON DELETE CASCADE ON UPDATE
CASCADE;

--
-- Relaciones para la tabla `pedido`
--
ALTER TABLE `pedido`
  ADD CONSTRAINT `pedido_ibfk_1` FOREIGN KEY (`Per_ID`) REFERENCES
`persona` (`Per_ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `pedido_ibfk_2` FOREIGN KEY (`Pag_ID`) REFERENCES
`tipo_pago` (`Pag_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Relaciones para la tabla `permisos`
--
ALTER TABLE `permisos`
  ADD CONSTRAINT `permisos_ibfk_1` FOREIGN KEY (`Rol_ID`)
REFERENCES `rol` (`Rol_ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `permisos_ibfk_2` FOREIGN KEY (`Mod_ID`)
REFERENCES `modulo` (`Mod_ID`) ON DELETE CASCADE ON UPDATE
CASCADE;

--
-- Relaciones para la tabla `persona`
--
ALTER TABLE `persona`
  ADD CONSTRAINT `persona_ibfk_1` FOREIGN KEY (`Rol_ID`)
REFERENCES `rol` (`Rol_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Relaciones para la tabla `producto`
--
ALTER TABLE `producto`
  ADD CONSTRAINT `producto_ibfk_1` FOREIGN KEY (`Cat_ID`)
REFERENCES `prod_cat` (`Cat_ID`) ON DELETE CASCADE ON UPDATE
CASCADE;
```

```sql
--
-- Filtros para la tabla `reembolsos`
--
ALTER TABLE `reembolsos`
  ADD CONSTRAINT `reembolsos_ibfk_1` FOREIGN KEY (`Ped_ID`)
REFERENCES `pedido` (`Ped_ID`);
COMMIT;
```

B) Inserción de datos.

Al tener un lenguaje de programación potente, insertar datos es más sencillo, gracias a que la aplicación está diseñada con el patrón MVC (Modelo, Vista y Controlador), la base de datos puede convertirse en un un objeto más para el sistema y fácilmente se pueden hacer ciertos requerimientos desde los modelos para persistir la información en la base de datos, un ejemplo de inserción, sería el siguiente:

```
<?php
      class RolesModel extends Mysql{
            public $intIdrol;
            public $strRol;
            public $strDescripcion;
            public $intStatus;
            public function __construct()
            {
                  parent::__construct();
            }
            public function insertRol(string $rol, string
$descripcion, int $status){
                  $return = "";
                  $this->strRol = $rol;
                  $this->strDescripcion = $descripcion;
                  $this->intStatus = $status;
                  $sql = "SELECT Rol_Nom FROM rol WHERE Rol_Nom =
'{$this->strRol}' ";
                  $request = $this->select_all($sql);

                  if(empty($request))
                  {
                        $query_insert  = "INSERT INTO
rol(Rol_Nom,Rol_Desc,Rol_Status) VALUES(?,?,?)";
                  $arrData = array($this->strRol,
$this->strDescripcion, $this->intStatus);
                  $request_insert =
$this->insert($query_insert,$arrData);
                  $return = $request_insert;
                  }else{
                        $return = "exist";
                  }
                  return $return;
```

```
            }
      } ?>
```

Como se puede ver la clase RolesModel está heredando de la clase MySQL donde esta tiene las instrucciones necesarias para ejecutar en caso de que el requerimiento sea una consulta, una actualización o inserción, de este modo el modelo solo debe de pasar la sentencia armada para que la clase MySQL sea la encargada de correrla en el servidor, evitando así que esta se esté ejecutando todo el tiempo y por el contrario solo se use en los momentos necesarios, dicha clase MySQL es la siguiente:

```
<?php
      class Mysql extends Conexion
      {
            private $conexion;
            private $strquery;
            private $arrValues;

            function __construct()
            {
                  $this->conexion = new Conexion();
                  $this->conexion = $this->conexion->conect();
            }
            //Insertar un registro
            public function insert(string $query, array $arrValues)
            {
                  $this->strquery = $query;
                  $this->arrVAlues = $arrValues;
            $insert = $this->conexion->prepare($this->strquery);
            $resInsert = $insert->execute($this->arrVAlues);
            if($resInsert)
               {
                  $lastInsert = $this->conexion->lastInsertId();
               }else{
                  $lastInsert = 0;
               }
               return $lastInsert;
            }
            //Busca un registro
            public function select(string $query)
            {
```

```
                $this->strquery = $query;
            $result = $this->conexion->prepare($this->strquery);
                $result->execute();
            $data = $result->fetch(PDO::FETCH_ASSOC);
            return $data;
            }
            //Devuelve todos los registros
            public function select_all(string $query)
            {
                $this->strquery = $query;
            $result = $this->conexion->prepare($this->strquery);
                $result->execute();
            $data = $result->fetchall(PDO::FETCH_ASSOC);
            return $data;
            }
            //Actualiza registros
            public function update(string $query, array $arrValues)
            {
                $this->strquery = $query;
                $this->arrVAlues = $arrValues;
                $update =
$this->conexion->prepare($this->strquery);
                $resExecute = $update->execute($this->arrVAlues);
              return $resExecute;
            }
            //Eliminar un registros
            public function delete(string $query)
            {
                $this->strquery = $query;
            $result = $this->conexion->prepare($this->strquery);
                $del = $result->execute();
            return $del;
            }
      }
 ?>
```

Como se puede ver la clase MySQL hereda o implementa la clase conexión que se explicará más adelante pero como se puede ver los métodos se dividen en insertar, actualizar y eliminar, dónde siempre recibirá un array para ejecutar los valores necesarios de la consulta reemplazando los ? de la instrucción sql por los valores que están dicho array.

C) Conexión de la aplicación con la BD.

Como se explicó anteriormente, la aplicación está dividida por capas donde cada una se conecta con la otra, para el caso de la conexión a la base de datos se hace de la siguiente manera:

```
<?php
class Conexion{
      private $conect;
      public function __construct(){
            $connectionString =
"mysql:host=".DB_HOST.";dbname=".DB_NAME.";.DB_CHARSET.";
            try{
                  $this->conect = new PDO($connectionString,
DB_USER, DB_PASSWORD);
                  $this->conect->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
            }catch(PDOException $e){
                  $this->conect = 'Error de conexión';
                echo "ERROR: " . $e->getMessage();
            }
      }

      public function conect(){
            return $this->conect;
      }
}

?>
```

Como se puede ver las clase conexion utiliza las variables globales tales como: DB_Name, DB_Host y demás para conectarse con la base de datos, dichas variables fueron declaradas en un archivo de configuración para que el programa las asimile solas, por último las conexiones se realizan gracias a la clase conocida como PDO (PHP Data Object) para realizar la conexión final con el servidor y así persistir de manera constante mientras la aplicación está en ejecución.

Referencias

Murphy, K., & Cabral, S. K. (2009). *MySQL Administrator's Bible* [Libro electrónico]. Wiley. https://www.amazon.com/-/es/Ian-Gilfillan/dp/8441515581

Udemy - Abel Oj Shicay. (2020–2021, diciembre 8–febrero 23). *Desarrollo web en PHP MVC y MySQL* [Curso virtual]. Desarrollo web en PHP MVC y MySQL, Bogotá, Colombia. https://www.udemy.com/course/desarrollo-web-en-php-mvc-poo-y-mysql-tienda-virtual/

Udemy - Francisco Javier Arce Anguiano. (2020–2021, noviembre 30–febrero 23). *Crea una tienda virtual con PHP y MySQL con el patrón MVC* [Curso virtual]. Crea una tienda virtual con PHP y MySQL con el patrón MVC, Bogotá, Colombia. https://www.udemy.com/course/crea-una-tienda-virtual-con-php-y-mysql-con-el-patron-mvc/

Udemy - Nicolas Schurmann. (2020–2021, noviembre 24–febrero 23). *Aprende Javascript ES9, HTML, CSS3 y NodeJS desde cero* [Curso virtual]. Aprende Javascript ES9, HTML, CSS3 y NodeJS desde cero, Bogotá, Colombia. https://www.udemy.com/course/aprende-javascript-es9-html-css3-y-nodejs-desde-cero/