

# 1. INTRODUCCIÓN

## 1.1. Introducción

Dentro de esta evidencia integradora se llevará a cabo la elaboración de una página web a través de los conocimientos vistos durante la clase, estos son participes del diseño web y programación en internet, el objetivo



principal de este proyecto es transformar dichos conocimientos en un Sistema que sea capaz de llevar el control de una papelería con el fin de aplicar los conocimientos adquiridos alrededor del cuatrimestre.

Los conocimientos de los cuales se hablan son: Lenguajes para programación web, Lenguajes del lado del servidor, Lenguajes del lado del cliente, Servidores web y Arquitectura MVC (Modelo vista controlador).

## 1.2. Herramientas

Existen varias herramientas que permiten el fácil diseño de una página web. Algunas de estas herramientas son las plantillas, las cuales nos ayudan para no comenzar una página web desde cero, es decir, que nos facilitan algunas partes de código para darnos una idea del diseño del sistema. Otra herramienta (de las más importantes) es el modelo MVC.

El modelo MVC o modelo vista controlador es un patrón de arquitectura que divide los datos y la lógica de la interfaz de usuario.

Las divisiones de estos módulos son:

- Vistas
- Modelos
- Controladores

Las vistas son la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica va dentro de este segmento. Este módulo es diferente a los demás ya que ni el modelo ni el controlador se preocupan de cómo se verán los datos o como se mostrarán al usuario, esa es solo responsabilidad de las vistas.

El modelo se encarga de los datos, (normalmente haciendo consultas a la base de datos). Ya sea para actualizaciones, consultas, búsquedas, etc.

Por último, el controlador, es el que se encarga de llevar un control, recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista. En pocas palabras el controlador es el intérprete entre las vistas y el modelo. En la ilustración 1 se puede ver un ejemplo de cómo trabaja el modelo MVC.

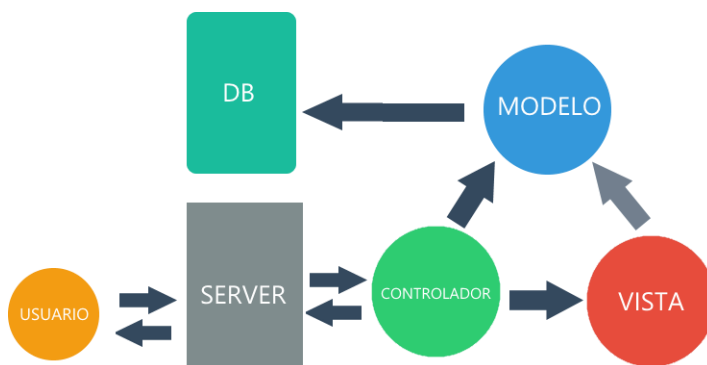


Ilustración 1 Arquitectura MVC

### 1.3. Laravel

La herramienta con la que se trabajará este sistema es Laravel.

Laravel es uno de los frameworks de código abierto más fáciles de asimilar para PHP. Es una herramienta simple y potente que tiene una interfaz elegante y fácil de usar. Dentro de Laravel se lleva el manejo del modelo MVC de una manera más práctica a través de carpetas que se dividen en segmentos al igual que el modelo vista controlador.



El objetivo de Laravel es ser un framework que permita el uso de una sintaxis refinada y expresiva para crear código de forma sencilla, permitiendo multitud de funcionalidades y herramientas para el desarrollo de páginas web.

Sus características son las siguientes:

- Sistema de ruteo, también RESTful
- Blade, Motor de plantillas
- Peticiones Fluent
- Eloquent ORM
- Basado en Composer
- Soporte para el caché
- Soporte para MVC
- Usa componentes de Symfony
- Adopta las especificaciones PSR-2 y PSR-4

### 1.4. PaperShöp

Paper shop es una página web desarrollada en el framework de Laravel para la gestión de productos, ventas, compras, proveedores y unidades de una papelería. Esta página web es producto de la evidencia integradora de la materia “Programación en Internet” como proyecto final.

Dentro de este proyecto se abarcan los siguientes temas:

- Login (función Laravel)
- Login con redes sociales (Facebook y Google)
- Validaciones
- Perfiles de usuario
- Agregación, modificación, consulta y eliminación de datos de distintos módulos
- Base de datos (con migración)
- Almacenamiento de archivos de manera local
- Generación de reportes (PDF)
- Gráficas

De los cuales cada uno fue visto dentro de la clase.

A continuación, se definirán conceptos y se mostrarán los pasos a la solución de cada uno de los temas mencionados anteriormente para explicar cómo funcionan y como fueron aplicados dentro del sistema. Además, se mostrarán capturas de pantalla para demostrar el perfecto funcionamiento de la página web.

## 2. MARCO TEÓRICO

### 2.1 Estructura Laravel

Laravel se divide en varias carpetas las cuales son:

#### Carpeta App

Esta carpeta tiene subcarpetas para comandos, comandos de consola, control de eventos, de excepciones, proveedores, servicios, etc. Los modelos se colocan dentro de esta misma. En la subcarpeta Http existen varias carpetas donde se guardan los controladores, middleware, entre otros.

#### Carpeta Bootstrap

Permite el sistema de arranque de Laravel.

#### Carpeta Config

Esta carpeta contiene toda una serie de archivos de configuración. La configuración de los componentes del framework se hace por separado. La configuración principal está en app.php.

#### Carpeta Database

Contiene las alimentaciones y migraciones de la base de datos.

#### Carpeta Public

Es el denominado "document root" del servidor web. Es el único subdirectorio que estará accesible desde fuera mediante el servidor web.

#### Carpeta Resources

En esta carpeta se guardan assets, archivos de idioma (lang) y vistas. Dentro de la subcarpeta views se almacenan los archivos .blade los cuales son las vistas de la página.

#### Carpeta Routes

Esta carpeta contiene varias subcarpetas, entre las cuales la más importante es la carpeta web que es donde se almacenan las rutas para poder re-direccionar entre las distintas vistas y controladores.

#### Carpeta Storage

Es el sistema de almacenamiento automático del framework, donde se guardan cosas como la caché, las sesiones o las vistas, logs, etc.

#### Carpeta Vendor

Esta carpeta contiene una cantidad de librerías externas, creadas por diversos desarrolladores que son dependencias de Laravel.

## 2.2 Comandos

Para el manejo de varias funciones en Laravel es necesario el uso de comandos por consola. Para la creación de un nuevo proyecto en Laravel el comando es el siguiente:

```
composer create-project laravel/laravel mi-proyecto-laravel
```

**NOTA:** Para poder crear el proyecto es necesario estar dentro de la carpeta htdocs de xampp.

Al crear el proyecto se crea una carpeta con el nombre asignado y dentro de esta se encuentran las carpetas que se mencionaron en el subtítulo anterior.

Para poner en marcha el proyecto el comando es el siguiente:

```
php artisan serve
```

Al momento de trabajar con base de datos Laravel tiene la opción de crear archivos para la migración de tablas. Para crear un archivo el comando es:

```
php artisan make:migration nombre_migracion
```

Para migrar todas las tablas a MySQL (base de datos a utilizar para el sistema) se usa el comando:

```
php artisan migrate
```

En el caso de la creación de modelos (que se crean dentro de la carpeta App) el comando es:

```
php artisan make:model nombre_modelo
```

## 2.3 Modelos

Para la creación de modelos ingresamos a la carpeta app y dentro de esta se crean archivos php para los diferentes modelos del sistema. La clase extiende de la librería Model (Illuminate\Database\Eloquent\Model).

Para definir a que tabla pertenece el modelo se utiliza la primera línea y para definir los atributos de la tabla se utiliza la segunda línea.

```
protected $table="Name_Table";
```

```
protected $fillable = ['attr1','attr2', 'attr3'...];
```

En el sistema a presentar los modelos a utilizar son los siguientes: Producto, Venta, Compra, Proveedor, Unidad, Usuario, VentaDetalles, CompraDetalles.

## 2.4 Controladores

Los controladores son para llevar el funcionamiento del sistema dentro de los controladores se llevan a cabo varias funciones que se definirán más adelante.

```
Use App\Name_Model;
```

La línea anterior nos permite hacer uso del modelo dentro del controlador como por ejemplo para agregar, modificar, eliminar o consultar valores en la base de datos. Las siguientes líneas son ejemplos de la utilización de los modelos dentro de los controladores:

```
$model= new Name_Model; (crea un nuevo elemento)
```

```
$model= Name_Model::all(); (consulta todos los valores de un elemento)
```

```
$model= Name_Model::findOrFail($id); (busca un elemento)
```

```
$model->delete(); (elimina un elemento después de buscarlo)
```

## 2.5 Vistas

Las vistas como ya se mencionó anteriormente son la interfaz del usuario con la que el sistema se puede comunicar con el usuario a través de texto imágenes, colores, botones, etc. Las vistas en Laravel se guardan con la extensión .blade.php dentro de la carpeta views. Dentro de estas se puede agregar código HTML para formar diseños agradables y fáciles de utilizar para el usuario. Además, se pueden utilizar links para hacer uso de otros archivos como CSS para el diseño o JavaScript para el dinamismo de la página.

## 2.6 Rutas

Para hacer que las vistas puedan re-direccionarse entre los controladores se utilizan las rutas de la carpeta web dentro de Routers. Las siguientes funciones son algunos ejemplos que se pueden utilizar para moverse entre las distintas vistas y controladores en el sistema:

- **Route::view:** Permite mandar de una vista a otra.
- **Route::post:** Permite mandar a un controlador y función definidas junto a un conjunto de información que se manda a través de la acción POST.
- **Route::get:** Permite mandar a una vista junto con un conjunto de valores que se obtienen de una función y controlador definidas para poder mostrarla al usuario.
- **Auth::routes():** Es una ruta definida por la implementación de Auth que agrega un conjunto de rutas específicas de esta función como lo son el login, registro, verificación, etc. de usuarios.



## 2.7 Middleware

Los middleware, son funciones que nos permiten agregar filtros a cada petición HTTP realizada por un usuario. Su finalidad de este componente es disminuir la carga de trabajo en los controladores y proporcionar una solución mucho más simple y estándar a la hora de aplicar las restricciones necesarias en el proyecto. El comando para hacer uso de ellos es:

```
php artisan make:middleware nombre_middleware
```

## 2.8 Autenticación

Laravel ofrece un sistema de autenticación predeterminado que se puede generar fácilmente con el comando:

```
php artisan make:auth
```

Este comando trabaja utilizando el modelo User que Laravel trae por defecto y la tabla users (que puede ser generada ejecutando las migraciones que ya vienen incluidas al crear el proyecto).

Además, incluye algunas carpetas que permiten una autenticación de usuarios de una manera más fácil y eficaz. En la Ilustración 2, se puede ver que se incluye una carpeta llamada Auth que implementa las funciones de Login, Registro, Verificación, Olvido y Reset de contraseñas.

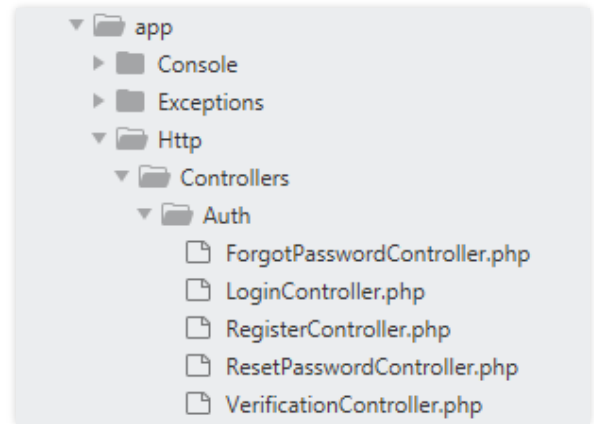


Ilustración 2 Carpeta Auth

## 2.9 Traits

Un trait es una agrupación de métodos con una funcionalidad específica para ser reutilizada en diferentes clases, son creados para la reutilización de código y nos permiten una mejor gestión de nuestros modelos. Para hacer uso de ellos podemos crear una carpeta llamada Traits dentro de la carpeta App. Dentro se crea un archivo php donde se hace uso de la palabra clave trait para crear una nueva función de este tipo.

Aquí se pueden crear diferentes funciones como por ejemplo la función de modificar u eliminar. En la Ilustración 3 se muestra un ejemplo de la función borrar. Para hacer uso de los traits es necesario implementarlos al principio de los modelos con la línea `use App\Traits\Name_Trait;` y con la línea `use Name_Trait;` dentro de las clases para así poder hacer uso de las funciones.

```
public static function borrar($id){
    return self::where('id',$id)->delete();
}
```

Ilustración 3 Ejemplo Trait

En los controladores se mandan a llamar de la siguiente forma: `Name_Model::borrar($id);`

## 2.10 Dompdf y Chart

Para la creación de reportes, se utiliza el componente `dompdf`, el cual sirve para convertir contenido HTML que es renderizado con PHP para obtener un archivo PDF. La facilidad de este se debe a que no se necesitan muchas instalaciones para poder hacer uso de la librería. Para hacer la descarga vía composer se utiliza la siguiente línea:

```
composer require barryvdh/laravel-dompdf
```

Este comando hace la instalación, pero también se deben declarar los providers de la siguiente manera. Dentro del archivo `config/app`, se agrega la siguiente línea dentro del arreglo providers:




```
Barryvdh\DomPDF\ServiceProvider::class
```

```
'PDF' => Barryvdh\DomPDF\Facade::class
```

Y el segundo comando dentro del arreglo aliases. Lo siguiente es configurar el controlador:

```
$pdf=PDF::loadView('reporte_ventas',compact('ventas','now','fe','fec'));  
return $pdf->stream('reporte_ventas.pdf');
```

Dentro de estas dos líneas se hace la conversión del documento, en la primera cargamos la vista “reporte\_ventas” (que es el archivo blade donde se imprimirán los datos) y el modelo (que este caso es el de ventas). Nota: Dentro de esta línea se hace uso del componente PDF de dompdf por lo que se debe agregar al controlador de la siguiente forma `use PDF`. En la segunda línea utilizamos la función `stream` para mostrar el archivo en formato pdf (en caso de que se quisiera descargar sin mostrar, se puede utilizar la función `download`).

7	2019-04-01	\$ 709	MARIA ARELY GABRIEL ROMERO	  
---	------------	--------	----------------------------	---

Dentro del archivo `reporte_ventas`, definimos el diseño para mostrar todos los datos de las ventas.

Para que el usuario pueda generar los reportes, debe presionar el botón de impresora como se muestra en la izquierda.

Ingrese el rango de fecha:

Fecha inicial:

abril de 2019

dom.

lun.

mar.

mié.

jue.

vie.

sáb.

31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

Aparecerá una venta donde se pide el rango de fechas para imprimir las ventas que se encuentren dentro de ese rango, luego se mandan al controlador para validarlas dentro de la consulta DB de la siguiente manera:

```
->whereBetween('ventas.fecha', [$fe, $fec])
```

Donde se seleccionan las ventas que están dentro del rango de las fechas que se reciben por método el POST y finalmente se retornan al archivo pdf.

Para hacer los reportes de las compras se hace de la misma manera, solo que los nombres de las tablas cambian al momento de hacer la consulta DB.

La siguiente función es charts la cual nos permite la creación de gráficas a través de datos que pueden ser obtenidos de la base de datos. Para hacer uso de esta función se necesita la siguiente línea:

```
<script type="text/javascript"
src="https://www.google.com/jsapi"></scrip
t>
```



Con esta línea se define que se hará uso del api de google charts (ya que es una herramienta desarrollada por google). Para la creación de la gráfica, primero se pide el mes que se desea para luego mandar el dato al controlador, el cual a través de una consulta DB `->whereMonth`, recibe todos los productos que han sido vendidos en el mes y los manda a la vista.

Lo siguiente es la creación de nuestra gráfica por lo que sea utiliza una nueva etiqueta script donde se define lo siguiente:

```
<script>
google.load("visualization", "1", {packages:["corechart"]});
google.setOnLoadCallback(dibujarGrafico);
function dibujarGrafico() {
var data = google.visualization.arrayToDataTable([
['Genre', 'Producto', { role: 'style' }],
@foreach($detalles as $d)
['{{ $d->nombre}}', {{ $d->cantidad}}, 'color: #439DA9; fontColor: #bbb '],
@endforeach ]]);
var options = {
title: 'Cantidad comprada de cada producto',
backgroundColor: '#E7F0F1',
}
new google.visualization.ColumnChart(
document.getElementById('GraficoGoogleChart-ejemplo-1')).draw(data,
options);
}
</script>
```

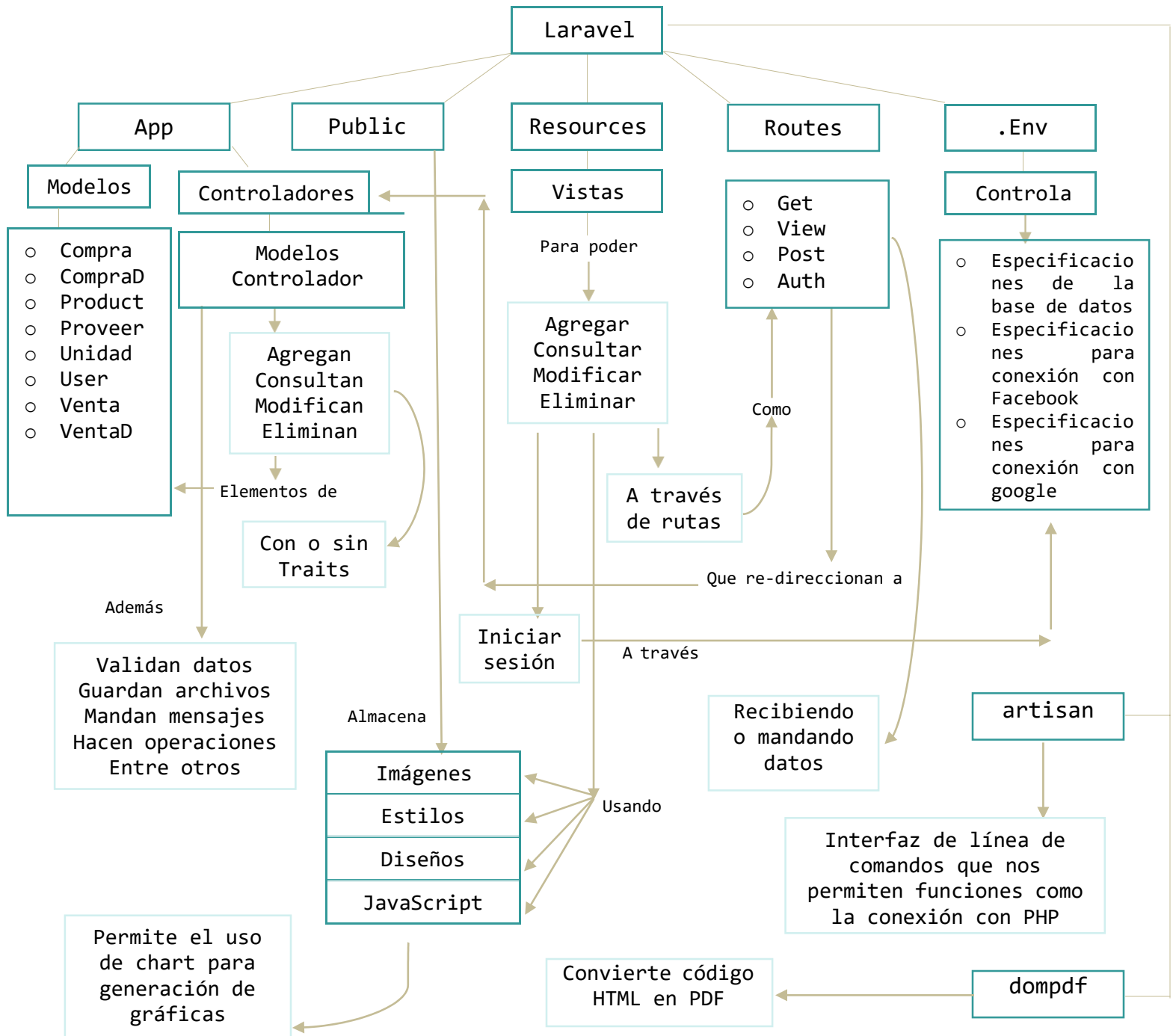
En la primera línea cargamos la gráfica, luego definimos una llamada a la función `dibujarGrafico` dentro de esta función se crean los elementos de la gráfica (en esta ocasión cargamos los valores a través de un foreach que recibe todos los datos de la tabla). Se reciben los productos y la cantidad de productos vendidos del mes los cuales se acomodan en forma de gráfica con el formato `[['{{ $d->nombre}}', {{ $d->cantidad}}]`.

Por último, se manda la gráfica para mostrarse en un div el cual tiene por id 'GraficoGoogleChart-ejemplo-1' con la línea `document.getElementById('GraficoGoogleChart-ejemplo-1').draw(data, options)`, donde data es la variable donde se encuentra la gráfica y options son opciones que se pueden agregar como el título de la gráfica o color de fondo.



# 3. DISEÑO

## 3.1 Funcionamiento



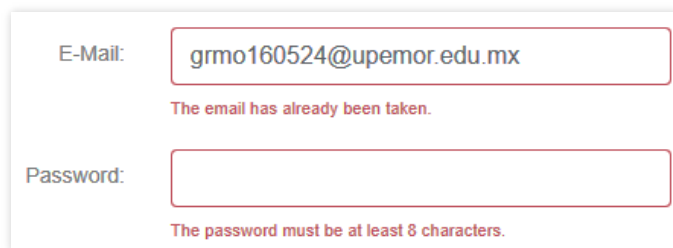
# 4. IMPLEMENTACIÓN

## 4.1 Registro

Dentro del sistema las primeras dos opciones que tiene el usuario es ingresar o registrarse. Para registrarse el sistema pide los siguientes datos:

- Nombre
- E-mail
- Password
- Confirmación de password
- Tipo de usuario
- Avatar

Al seleccionar en registrarse el sistema manda a una ruta situada en `Auth::routes()`, aquí se manda al controlador de registro que valida que los datos sean correctos. Gracias al componente de Auth se facilitan funciones como se puede ver en la Ilustración 6.



The screenshot shows a registration form with two input fields. The first field is labeled 'E-Mail:' and contains the text 'grmo160524@upemor.edu.mx'. Below this field, a red error message reads 'The email has already been taken.' The second field is labeled 'Password:' and is empty. Below this field, a red error message reads 'The password must be at least 8 characters.'

Ilustración 6 Login con Auth

Algunas validaciones que se hacen son: El tamaño de la contraseña, la validación del correo (si este ya existe, que es el caso en la Ilustración), Además se valida que las contraseñas coincidan.

Cuando los datos son correctos se hace el registro del usuario y automáticamente ingresa al sistema.

## 4.2 Login

Para el login se pide el email y la contraseña del usuario. Dentro de este solo se validan que ambos datos existan dentro de la base de datos, para así poder mandar al usuario a la página de inicio dependiendo del tipo de usuario que sea.

## Login con Facebook

Para ingresar a través de Facebook, dentro del sistema se encuentra un link que manda automáticamente a iniciar sesión en Facebook.

Si ya se tiene una sesión abierta el sistema simplemente pide permiso al usuario con un mensaje de cuadro como se muestra en la Ilust. 4.



Ilustración 4 Inicio con Facebook

## Login con Google

Para ingresar con google el sistema solo pide la cuenta con la cual se quiere ingresar (Ilust. 5) y automáticamente ingresa mandando los datos del usuario.

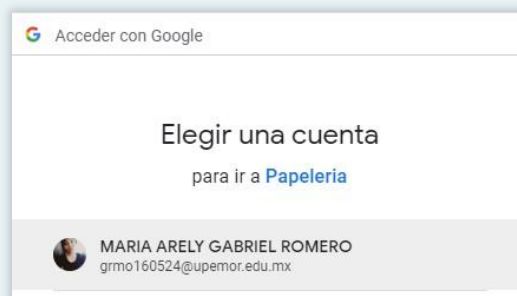


Ilustración 5 Inicio con Google

**NOTA:** Para hacer uso de los inicios de redes sociales es necesario la creación de identificadores que son creados desde las páginas oficiales de la red social. En estas se crean dos identificadores los cuales son el Id del cliente y el Id del cliente secreto. Para implementarlos dentro del sistema, se agregan dentro del archivo .env que se encuentra en la raíz del proyecto creado. En la Ilustración 7 se muestra un ejemplo de cómo es implementado:

```
FACEBOOK_CLIENT_ID=1979749172321386
FACEBOOK_CLIENT_SECRET=7610764c2b8eb720c9b8576d2d609080
FACEBOOK_REDIRECT=http://localhost:8000/login/facebook/callback

GOOGLE_CLIENT_ID=261562777304-ke536bo9lotb368gbaf0qa6qgj3qenpd.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=wthHVHD7C1FV12Qz3jfn_ICy
GOOGLE_REDIRECT=http://localhost:8000/login/google/callback
```

*Ilustración 7 Credenciales*

Como se puede ver se agregan los identificadores dentro de las variables correspondientes y además se agrega un link de redirección el cual manda al controlador para mandar al inicio en caso de que todo salga bien.

En este archivo también se puede hacer la modificación de puertos y nombre de la base de datos.

## 4.3 Inicio

Al ingresar al sistema, lo primero que se analiza es el tipo del usuario a través del siguiente método:

```
public function index()
{
    if (Auth::check()){
        if(Auth::user()->perfil_id == 1){
            return view('homeGeneral');
        }
        else{
            return view('homeAdmin');
        }
    }
}
```

Dentro de la imagen se hace uso del componente `Auth::check()`, el cual obtiene los datos del usuario que ha iniciado sesión. Posteriormente se obtiene el tipo de usuario de la siguiente manera:

`Auth::user()->perfil_id`, haciendo referencia al tipo de usuario a través de dos valores los cuales se validan y en el caso de que el valor obtenido sea 1, entonces se manda al inicio de usuario general, de lo contrario se manda al de usuario administrador.

La diferencia entre estos son las funciones que estos pueden hacer dentro del sistema. El usuario general solo puede agregar y consultar ventas y productos. Mientras que el administrador puede agregar, consultar, modificar y eliminar las ventas, compras, productos y proveedores. Ambos pueden configurar su cuenta, es decir que pueden cambiar su nombre, correo, avatar, etc.

Para el diferente menú de los usuarios se hace uso de la siguiente línea:

```
@extends(Auth::user()->perfil_id==1 ? 'layouts.app2' : 'layouts.app3')
```

Esta valida al igual que el método anterior el tipo de usuario para extender una vista distinta la cual muestra un tipo de menú para cada usuario.

Gracias al componente de Auth también se puede obtener el nombre del usuario y el avatar. Con estas variables se puede hacer la impresión de los datos del usuario sin necesidad de consultar a la base de datos.

## 4.4 Registros y consultas

Para el registro de datos se manda a una sección donde se hace petición de los valores del módulo correspondiente. Por ejemplo, para agregar productos la sección es la que se muestra en la Ilustración 8.

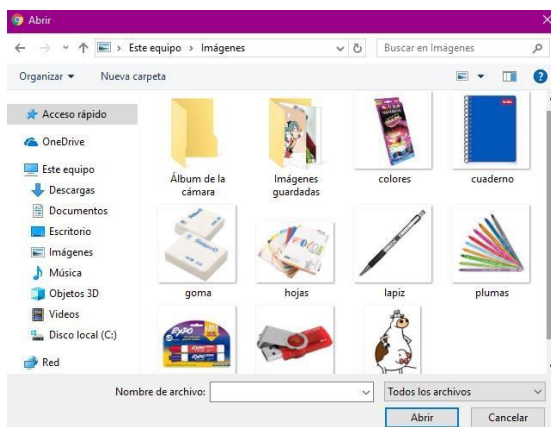
Dentro de esta vista se piden los valores que se muestran en la imagen. Al momento de ingresar el tipo de unidad y el proveedor, se despliega una lista con las unidades/proveedores que existen dentro de la base de datos.

ID Unidad:

Seleccione una unidad

- Seleccione una unidad
- Escolar
- Oficina

Para agregar una imagen al producto, se abre una ventana como se muestra a continuación:



Donde se puede seleccionar una imagen del PC. Por último, están las validaciones, las cuales se hacen por cada campo para que no esté vacío o sea del tipo de dato que se espera.

Cantidad:

Ingrese la cantidad

ID Unidad:

Seleccione una unidad

La unidad debe ser formato identificador

Ingrese los datos que se solicitan para un nuevo producto

Nombre:

Marca:

Descripción:

Precio:

Cantidad:

ID Unidad:

Proveedor:

Imágen:

Seleccionar archivo No se eligió archivo

AGREGAR

Ilustración 8 Agregación de productos

Para validar los datos se manda la información por una ruta definida hacia el controlador a través de un POST para ser analizados como se muestra a continuación:

Los valores se reciben en un `Request` para posteriormente ser analizados con el método `validate`. Dependiendo de cuál sea la validación, se manda un mensaje al usuario como se mostró en la imagen anterior.

Si los datos se encuentran correctos, entonces se reciben los valores con el modelo correspondiente. En este caso el modelo es el de `Producto`.

```
public function insertar(Request $request)
{
    request()->validate([
        'nom'=>'required|alpha',
        'mar'=>'required|',
        'des'=>'required|min:10',
        'pre'=>'required|numeric',
        'can'=>'required|numeric',
        'uni'=>'required|numeric',
        'pro'=>'required|numeric',
        'img'=>'required|image'
    ]);
}
```

```
$pro=new Producto;
$pro->nombre=request()->get('nom');
```

Se crea un nuevo producto y con el símbolo `->` se define el campo del modelo y con `request()->get()`; se reciben los datos del POST con su nombre correspondiente.

Para guardar la imagen del producto en una carpeta dentro del proyecto se hace lo siguiente:

```
$originalImage= $request->file('img');
$thumbnailImage = Image::make($originalImage);
$originalPath = public_path().'/img/products/';
$thumbnailImage->save($originalPath.time().$originalImage-
->getClientOriginalName());
$thumbnailImage->resize(150,150);
$pro->imagen=time().$originalImage->getClientOriginalName();
```

Donde se obtiene la imagen para guardarla dentro de la dirección `/img/products/` con el nombre que se le define a la nueva imagen. Por último, se define un tamaño y se guarda dentro del modelo. Finalmente, para guardar el modelo se hace lo siguiente:

```
$pro->save();
```

La función `save` es un componente de la librería `Model` (como se mencionó en la sección 2.4 Controladores) y es para insertar los valores dentro de la base de datos.

Para las consultas la función es más fácil ya que simplemente se mandan a llamar los valores de la base de datos:

```
$pro=Productos::all();
```

Donde solo se retorna la variable `$pro` para mostrar los datos en la vista. Si se necesita hacer consultas en varias tablas un método puede ser:

```
$productos=DB::table('productos as p')
->join('unidades','p.unidad_id','=','unidades.id')->select('p.*','unidades.unidad as unidad_id')->
orderBy('p.id', 'desc')->get();
```

Donde se hace uso de `DB` para hacer consultas con `join` (método para unir tablas).

## 4.5 Trait (modificación y eliminación)

La modificación y eliminación de valores se realiza a través Traits (véase la sección 2.9). Al usuario al consultar los modelos se le muestran dos botones donde puede seleccionar el elemento a borrar o modificar como por ejemplo con los Productos (Ilustración 9).



Ilustración 9 Consulta de productos

En la parte superior derecha se muestran las dos opciones, si se selecciona la de eliminar, se manda el ID del producto a la ruta, la cual lo manda al controlador:

```
public function eliminar($id)
{
    $pro=Producto::findOrFail($id);
    if(\File::exists(public_path('img/products/'.$pro->imagen))){
        \File::delete(public_path('img/products/'.$pro->imagen));
    }
    Producto::borrar($id);
    return \Redirect::to('consultarProductos')->with('msj','El producto ha sido eliminado');
}
```

Como se ve en la imagen, la función recibe la variable `$id` la cual nos permite encontrar cual es el producto con la línea `Producto::findOrFail($id)`. (En las siguientes dos líneas se obtiene la imagen del producto para eliminarla también). Por último, se manda a llamar la función trait borrar con la línea `Producto::borrar($id)`, la cual recibe el elemento a borrar y ejecuta el comando `delete`:

```
public static function borrar($id){
    return self::where('id',$id)->delete();
}
```

Al borrar el elemento, se retorna al controlador el cual redirecciona hacia la vista `consultarProductos` con un mensaje para hacerle saber al usuario que el elemento ha sido eliminado.

Si se selecciona la opción de modificar, entonces la ruta obtiene los datos del elemento con el id enviado de la siguiente manera:

```
public function modificar($id){
    $pro=Producto::findOrFail($id);
    return view('modificarProducto',compact('pro'));
}
```

Donde se obtiene el elemento con el id y se retornan sus datos para mostrarlos, como se puede ver en la Ilustración 10.

Ingrese los datos que se solicitan para modificar un producto

Nombre:

Marca:

Descripción:

Precio:

Cantidad:

ID Unidad:

ID Proveedor:

Imagen:  
 No se eligió archivo

Ilustración 10 Modificación de producto

Esta función es parecida a la de agregar ya que se piden los mismos datos, pero ya rellenos, para posteriormente mandar a modificar, donde se validan de la misma manera (ya que son los mismos datos para el POST).

La diferencia es al momento de buscar el elemento:

```
$pro=Producto::findOrFail($id);  
$pro->nombre=request()->get('nom');
```

Donde en vez de crear un nuevo producto, se busca el elemento con el componente `findOrFail` para luego obtener los valores con el `get` (como se hizo al agregar).

```
Producto::modificar($id,$pro);
```

Después mandamos el modelo y el id hacia la función `trait` de modificar.

```
public static function modificar($id,  
Model $model){  
    return self::where('id',$id)  
    ->update($model->attributes);  
}
```

En la función `modificar` se recibe el id y el modelo para luego buscar el id con el comando `where` y mandar a modificar con el comando `update` todos los atributos del modelo con la línea `$model->attributes`.

Por último, se retorna y se manda a la vista de consulta de productos con su mensaje de éxito.

Cada una de las funciones de agregación, consulta, modificación y eliminación, funcionan igual en cada modelo (solo cambia el nombre del modelo en los controladores), en este caso se desarrolló el de los productos ya que era el más completo.



## 4.6 Gestión compras y ventas

A continuación, se muestran algunos ejemplos de cómo se muestra en los modelos Compra y Venta:

Ingrese los datos que se solicitan para los detalles de la compra

Producto:

USB

Cantidad:

2

Compra:

2019-03-29

AGREGAR

Ilustración 12 Agregación detalles compra









#	ID compra	Precio total	Producto	Cantidad	
5	4	\$ 70	Plumas	2	 
6	5	\$ 35	USB	1	 
7	5	\$ 35	Cuaderno	1	 
8	4	\$ 70	USB	2	 

Ilustración 11 Consulta de detalles compras









#	ID venta	Precio total	Producto	Cantidad	
3	4	\$ 70	Cuaderno	2	 
4	5	\$ 35	Plumas	1	 
5	6	\$ 35	Cuaderno	1	 
6	5	\$ 140	Plumas	4	 

Ilustración 13 Consulta detalles ventas

Ingrese los datos que se solicitan para los detalles de la venta

Producto:

Plumas

Cantidad:

2

Venta:

2019-03-01

AGREGAR

Ilustración 14 Agregación detalles venta



Al agregar los detalles de la compra, se pide el producto, para ello se despliega una lista con los productos y se pide la fecha de la compra donde también se despliega una lista con las fechas, estas listas se obtienen de la siguiente manera:

```
$pro=Producto::all();
$co=Compra::all();
Return view('agregarDetallesCompra',compact
('pro','co'));
```

```
<br><label>{{ __('Compra:') }}</label>
<select name="fec" id="fec" class="form-control">
<option selected>Seleccione la fecha</option>
@foreach($co as $c)
<option value="{{ $c->id }}">{{ $c->fecha }}</option>
@endforeach
</select>
```

seleccionada mandar el id del producto o la compra para saber que producto o compra se seleccionó.

Para la gestión de compras se toman los valores recibidos del POST para posteriormente realizar las siguientes operaciones:

```
$id_pro=request()->get('pro');
$can=request()->get('can');
$id_co=request()->get('fec');
$pro=Producto::findOrFail($id_pro);
$tot=$pro->precio*$can;
$pro->cantidad=$pro->cantidad+$can;
$co=Compra::findOrFail($id_co);
if($co->id==$id_co){
    $co->total_compra=$co->total_compra+$tot;
}
```

En esta parte se reciben el id del producto, el id de la compra y la cantidad. Lo primero que se hace es buscar el producto con el id obtenido para crear una variable `$tot` donde se guardará cuanto será el precio total multiplicando la cantidad por el precio del producto y para editar su cantidad de la siguiente manera: `$pro->cantidad=$pro->cantidad+$can`, donde decimos de la nueva cantidad será igual a la cantidad que ya se tenía más la cantidad que ingresó el usuario para incrementar las existencias del producto.

Después buscamos la compra para a través de un `if`, saber a qué compra se está agregando estos nuevos `detalles_compra` y así aumentar el total de la compra dentro de la variable `total_compra` en el modelo `Compra`.

Por último, guardamos todos los cambios en los modelos:



ID compra	Fecha	Total compra	
4	2019-03-29	\$ 175	
5	2019-03-22	\$ 245	

Ilustración 15 Consulta de compras

```
$com->save();  
$pro->save();  
$co->save();
```

Y finalmente podemos ver las compras con su nuevo total como se ve en la Ilustración 15. Para las Ventas el proceso es similar solo que, al momento de definir las cantidades, estas se restan en vez de sumarse.

## 4.7 Configuración cuenta

Al iniciar sesión, se muestra una sección donde se ve el nombre del usuario y al dar clic en esta parte, se muestran las opciones para ver perfil y cerrar sesión. En la opción para ver perfil el usuario puede ver la información de su cuenta de la siguiente manera:

Donde el usuario puede ver toda su información editarla, como por ejemplo para cambiar su nombre de usuario, su correo, su contraseña, su tipo de usuario o su foto de perfil.

 MARIA ARELY GABRIEL ROMERO  
Usuario Administrador  
mariz.magr@gmail.com



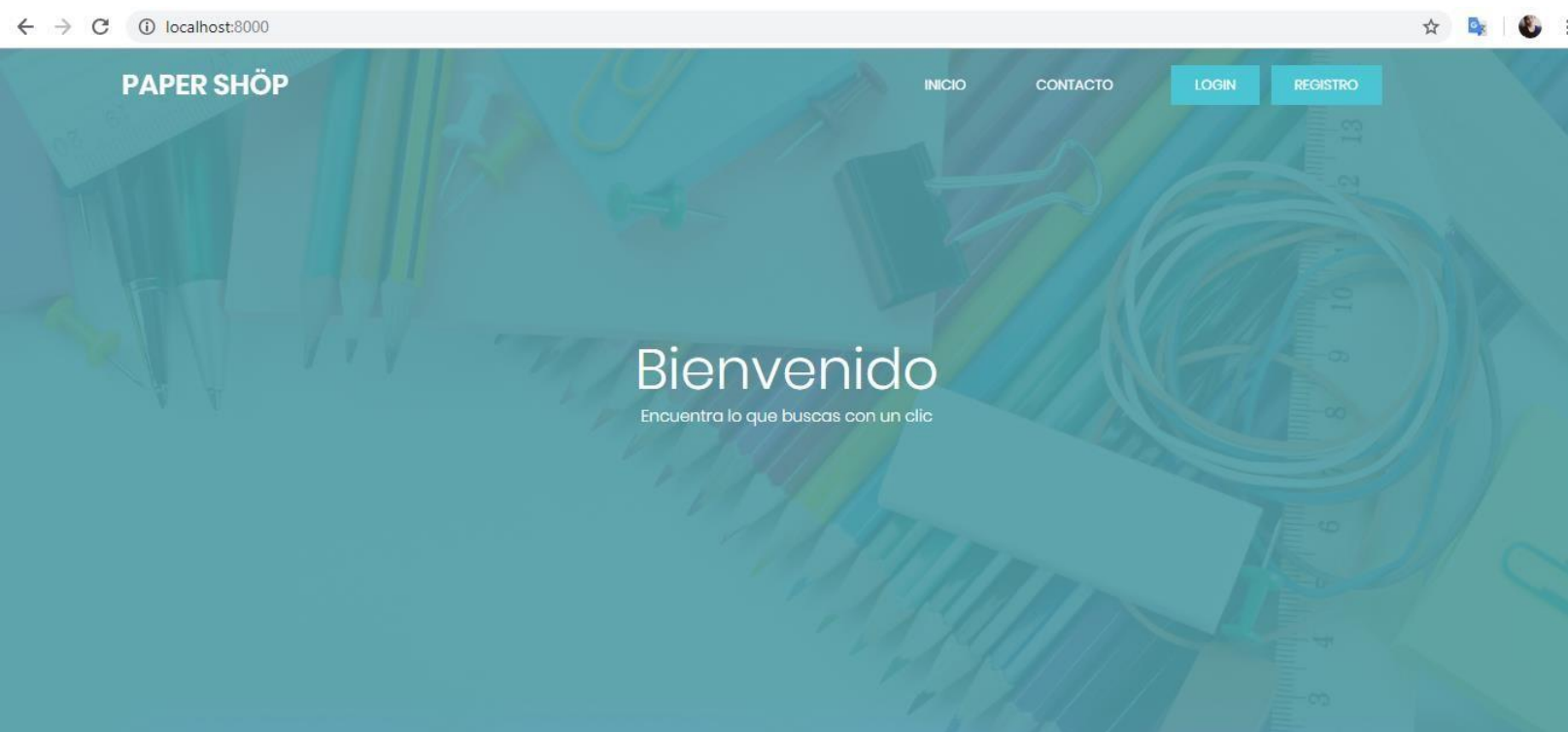
 Identificador: 10  
 Contraseña: ●●●●●●

 CONFIGURAR MI CUENTA

# 5. EVALUACIÓN

A continuación, se mostrarán las pruebas del funcionamiento y vistas del sistema

## 5.1 Index



*Ilustración 16 Página principal PaperShop*

En la Ilustración 16 se muestra la vista principal del sistema donde se puede apreciar las opciones que tiene el usuario que son:

- Registrarse
- Iniciar sesión

Las cuales se verán a detalle a continuación.

## 5.2 Registro

Registro

Nombre: Mary

E-Mail: mariz.magr@gmail.com

Password: .....

Confirmar Password: .....

Tipo user: Usuario general

Avatar: Seleccionar archivo vaca.png

Registrarme

*Ilustración 17 Registro de usuarios*

En la Ilustración se ve el registro de usuarios el cual pide los datos de registro:

- Nombre (Texto)
- E-mail (Correo)
- Password (8 caracteres)
- Confirmación password (8 caracteres)
- Tipo usuario (General / Administrador)
- Avatar (Imagen)

## 5.3 Login

This is a screenshot of a web form titled 'Login'. It contains two input fields: 'E-Mail:' with the value 'grmo160524@upemor.edu.mx' and 'Password:' with masked characters '\*\*\*\*\*'. Below these fields is a checkbox labeled 'Recordame'. There are three buttons: a teal 'Login' button, a link 'Olvide mi password?', and two social media login options: 'Iniciar sesión con Facebook' and 'Iniciar sesión con Google'.

*Ilustración 18 Login de usuarios*

This screenshot shows a confirmation screen for logging in via Facebook. At the top is the Facebook logo. Below it, text states 'PaperShop recibirá: tu nombre y foto del perfil y dirección de correo electrónico'. There is an 'Editar' link with a pencil icon. A large blue button says 'Continuar como Mary'. Below that is a 'Cancelar' link. At the bottom, a small lock icon and text say 'No permite que la app publique en Facebook.'

*Ilustración 19 Login con Facebook*

En la Ilustración 18 se puede ver el login de usuarios normal.

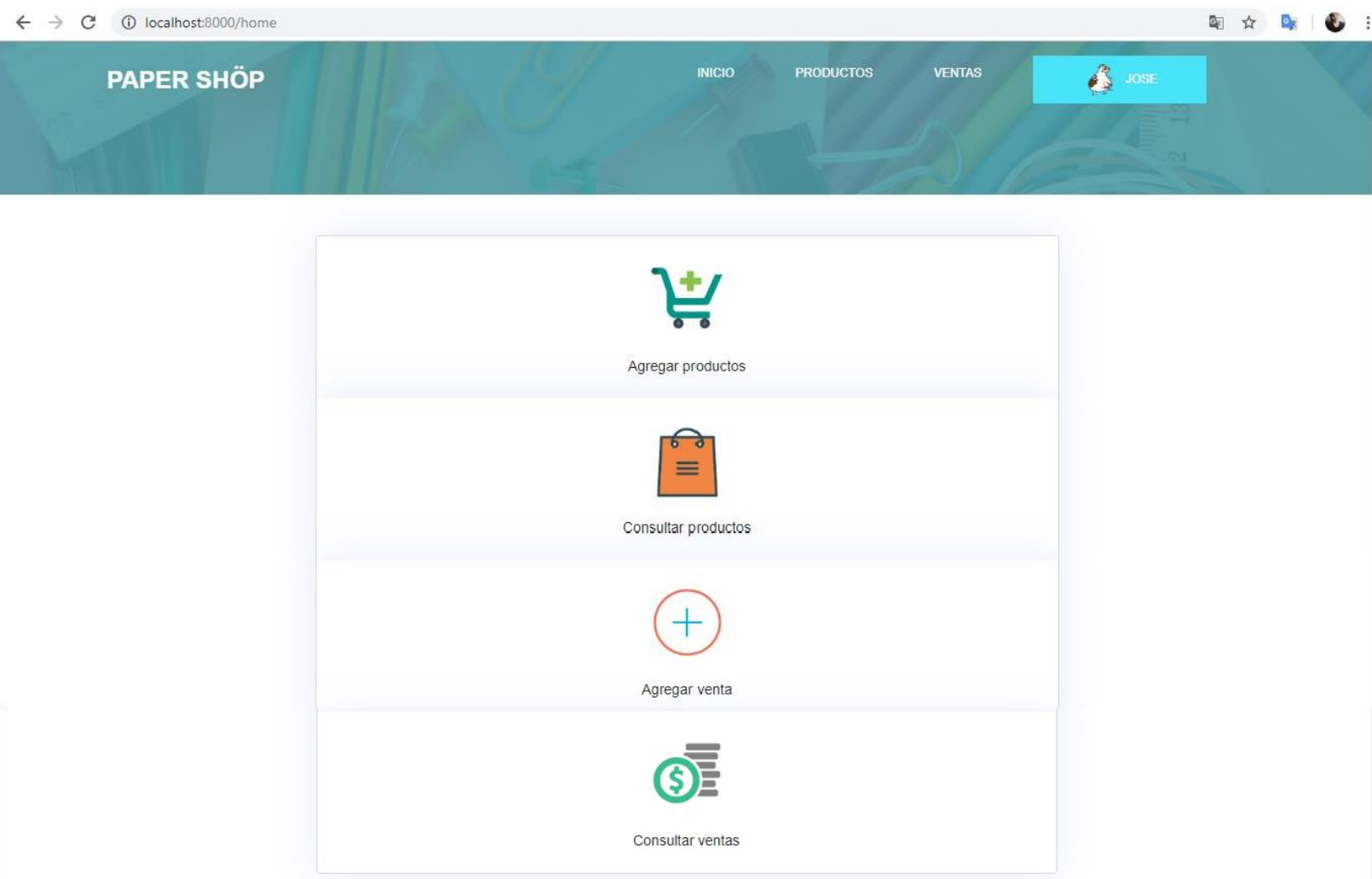
En la Ilustración 19 se muestra el inicio de sesión con Facebook donde PaperShop pide permisos al usuario para poder utilizar su información.

This screenshot shows the Google login interface. At the top is the Google logo and the text 'Acceder con Google'. Below is the heading 'Elegir una cuenta para ir a Papeleria'. A list of accounts is shown, each with a profile picture, name, and email address: 'MARIA ARELY GABRIEL ROMERO' (grmo160524@upemor.edu.mx), 'JOSÉ GUADALUPE SUAREZ GIL' (sgjo168325@upemor.edu.mx), 'Jose Gil' (jose.gil310897@gmail.com), and another 'Jose Gil' (jose.gpu310897@gmail.com).

*Ilustración 20 Login con Google*

Y en la Ilustración 20 se muestra el inicio con Google donde se muestran las distintas cuentas con las que puede ingresar (o agregar una nueva). Para poder utilizar su información también.

## 5.4 Usuario general




*Ilustración 21 Inicio usuario general*

En la Ilustración 21 se muestra el inicio del usuario general con las opciones:

- **Agregar productos**
- **Consultar productos**
- **Agregar venta**
- **Consultar venta**

← → ↻ localhost:8000/consultarProductos

PAPER SHÖP INICIO PRODUCTOS VENTAS JOSE



**USB**


Marca: AAA

Descripción: Memoria usb color rojo 1 pieza

Cantidad: \$ 8

Precio: \$130

Unidad: Escolar  
Proveedor: Mary



**PLUMAS**

Marca: Bic

Descripción: Paquete de plumas de colores (10 piezas)

Cantidad: Sin existencias

Precio: \$59

Unidad: Escolar

Este tipo de usuario al consultar los productos/ventas puede verlos, pero no puede eliminarlos ni modificarlos.

← → ↻ localhost:8000/consultarVentas

PAPER SHÖP INICIO PRODUCTOS VENTAS JOSE

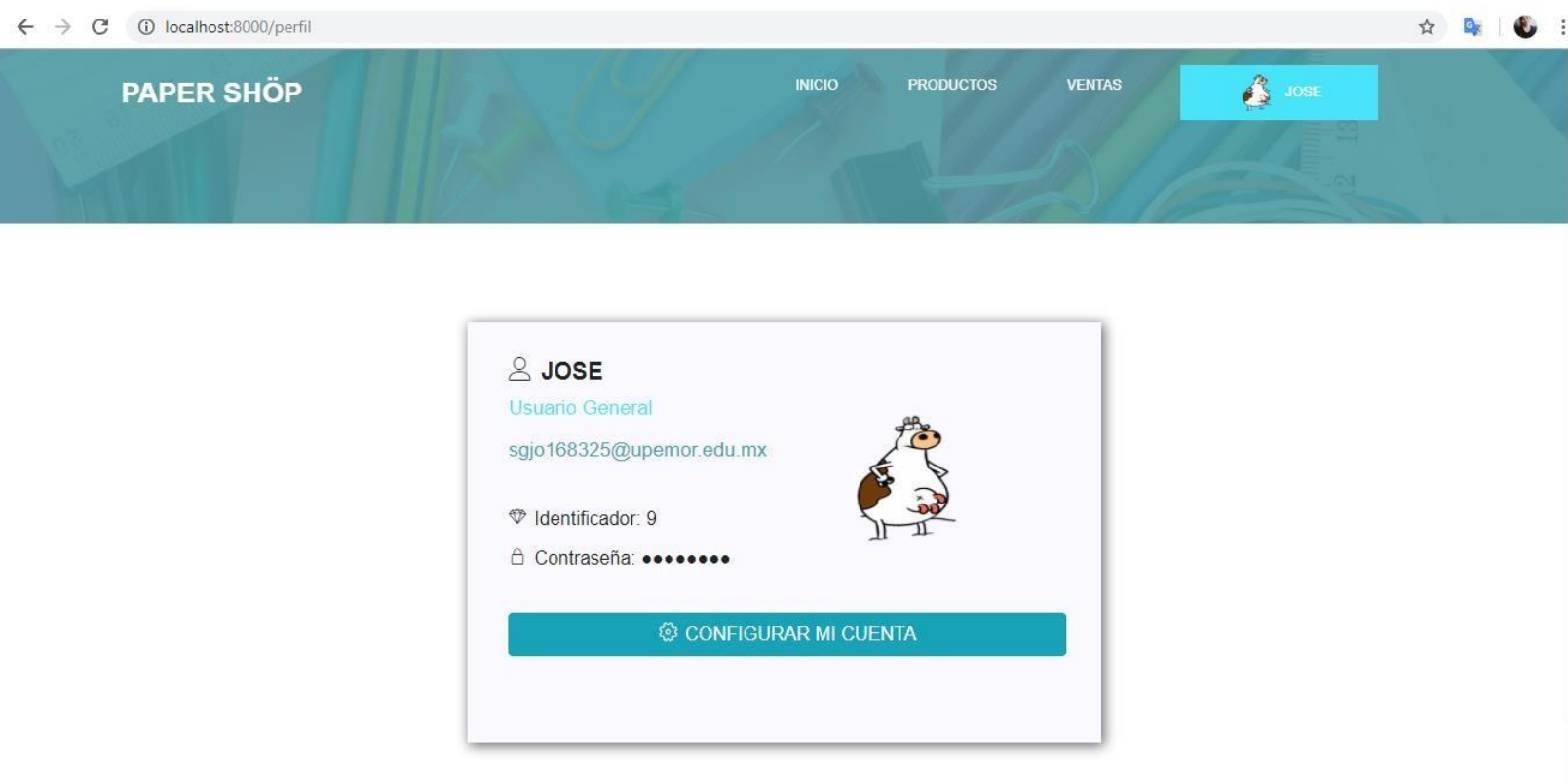
AGREGAR VENTA

#	Fecha	Total de venta	Usuario
4	2019-03-30	\$ 105	MARIA ARELY GABRIEL ROMERO
5	2019-03-01	\$ 175	MARIA ARELY GABRIEL ROMERO
6	2019-03-06	\$ 140	Jose

AGREGAR DETALLES

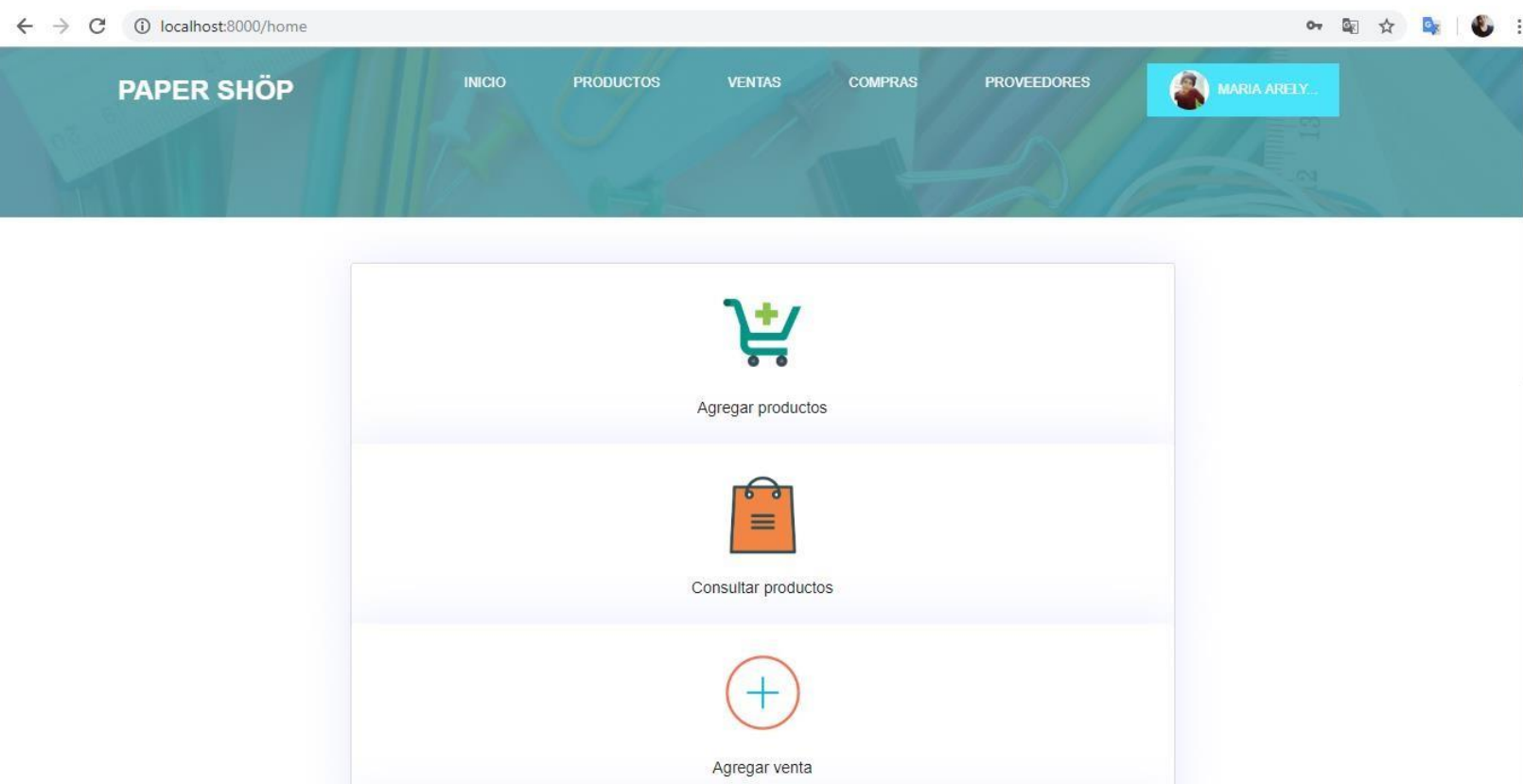
#	ID venta	Precio total	Producto	Cantidad
3	4	\$ 70	Cuaderno	2
4	5	\$ 35	Plumas	1
5	6	\$ 35	Cuaderno	1
6	5	\$ 140	Plumas	4

Además, puede hacer la configuración de su cuenta como se ve en la imagen de abajo





## 5.5 Usuario administrador




*Ilustración 22 Inicio usuario administrador*

En la Ilustración 22 se muestra el inicio del usuario administrador con las opciones:

- **Agregar productos**
- **Consultar productos**
- **Agregar venta**
- **Consultar venta**
- **Agregar compra**
- **Consultar compras**
- **Agregar proveedores**
- **Consultar proveedores**

← → ↻ localhost:8000/consultarProductos

**PAPER SHÖP** INICIO PRODUCTOS VENTAS COMPRAS PROVEEDORES MARIA ARELY...



**CUADERNO**


Marca: Scribe

Descripción: Cuaderno color azul tamaño oficio (cuadro grande)

Cantidad: \$ 1

Precio: \$35

Unidad: Escolar  
Proveedor: Carlitos



**USB**

Marca: AAA

Descripción: Memoria usb color rojo 1 pieza

Cantidad: \$ 8

Precio: \$130

Este usuario si puede eliminar y modificar los productos y las ventas como se puede ver en ambas imágenes.

← → ↻ localhost:8000/consultarVentas

**PAPER SHÖP** INICIO PRODUCTOS VENTAS COMPRAS PROVEEDORES MARIA ARELY...

AGREGAR VENTA

#	Fecha	Total de venta	Usuario
4	2019-03-30	\$ 105	MARIA ARELY GABRIEL ROMERO
5	2019-03-01	\$ 175	MARIA ARELY GABRIEL ROMERO
6	2019-03-06	\$ 140	Jose

AGREGAR DETALLES

#	ID venta	Precio total	Producto	Cantidad
3	4	\$ 70	Cuaderno	2
4	5	\$ 35	Plumas	1
5	6	\$ 35	Cuaderno	1
6	5	\$ 140	Plumas	4

← → ↻ localhost:8000/consultarProveedores ☆ [icon] [icon]

**PAPER SHÖP** INICIO PRODUCTOS VENTAS COMPRAS PROVEEDORES MARIA ARELY...

AGREGAR PROVEEDOR

#	Nombre	Dirección	Teléfono		
2	Mary	Temixco, Morelos 26601	3092781		
6	Carlitos	Temixco, Morelos 26601	3092781		

Además, puede llevar el control de otros dos módulos que son los proveedores y las compras de los productos.

← → ↻ localhost:8000/consultarCompras ☆ [icon] [icon]

**PAPER SHÖP** INICIO PRODUCTOS VENTAS COMPRAS PROVEEDORES MARIA ARELY...

AGREGAR COMPRA AGREGAR DETALLES

ID compra	Fecha	Total compra	
4	2019-03-29	\$ 175	
5	2019-03-22	\$ 245	

#	ID compra	Precio total	Producto	Cantidad	
5	4	\$ 70	Plumas	2	 
6	5	\$ 35	USB	1	 
7	5	\$ 35	Cuaderno	1	 
8	4	\$ 70	USB	2	 

## 5.6 Base de datos

The screenshot shows the phpMyAdmin interface in a web browser. The address bar indicates the URL: localhost/phpmyadmin/db\_structure.php?server=1&db=socialauth. The interface is in Spanish. On the left, a sidebar shows a tree view of databases, with 'socialauth' selected. The main area displays a table of database structures for the 'socialauth' database. The table has columns: Tabla, Acción, Filas, Tipo, Cotejamiento, Tamaño, and Residuo a depurar. The tables listed are: compras, detalle\_compras, detalle\_ventas, direcciones, migrations, password\_resets, perfiles, productos, proveedores, unidades, users, and ventas. Each table row includes icons for actions like 'Examinar', 'Estructura', 'Buscar', 'Insertar', 'Vaciar', and 'Eliminar'. At the bottom, there are buttons for 'Imprimir', 'Diccionario de datos', and 'Crear tabla'.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
compras	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_unicode_ci	16 KB	-
detalle_compras	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_unicode_ci	16 KB	-
detalle_ventas	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_unicode_ci	16 KB	-
direcciones	Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_unicode_ci	16 KB	-
migrations	Examinar Estructura Buscar Insertar Vaciar Eliminar	11	InnoDB	utf8mb4_unicode_ci	16 KB	-
password_resets	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_unicode_ci	16 KB	-
perfiles	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_unicode_ci	16 KB	-
productos	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_unicode_ci	16 KB	-
proveedores	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_unicode_ci	16 KB	-
unidades	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_unicode_ci	16 KB	-
users	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_unicode_ci	32 KB	-
ventas	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_unicode_ci	16 KB	-
12 tablas	Número de filas	36	InnoDB	latin1_swedish_ci	208 KB	0 B

Ilustración 23 Base de datos

En la Ilustración 23 se muestra el programa de phpMyAdmin mostrando las tablas de la base de datos del sistema:

- **compras**
- **detalle\_compras**
- **detalle\_ventas**
- **direcciones**
- **migrations**
- **password\_resets**
- **perfiles**
- **productos**
- **proveedores**
- **unidades**
- **users**
- **ventas**

## Tabla productos

localhost/phpmyadmin/sql.php?db=socialauth&table=productos&pos=0

Servidor: 127.0.0.1 » Base de datos: socialauth » Tabla: productos

Mostrando filas 0 - 2 (total de 3, La consulta tardó 0,0000 segundos.)

SELECT \* FROM `productos`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

+ Opciones

	id	nombre	marca	descripcion	precio	cantidad	unidad_id	imagen	proveedor_id	created_at	updated_at
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	22	Plumas	Bic	Paquete de plumas de colores (10 piezas)	59	0	1	1554021939plumas.jpg	2	2019-03-29 23:33:29	2019-03-29 23:33:29
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	23	USB	AAA	Memoria usb color rojo 1 pieza	130	8	1	1554022031usb.jpg	2	2019-03-30 00:12:05	2019-03-30 00:12:05
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	24	Cuaderno	Scribe	Cuaderno color azul tamaño oficio (cuadro grande)	35	1	1	1554019591cuaderno.jpg	6	2019-03-30 00:15:14	2019-03-30 00:15:14

Seleccionar todo | Para los elementos que están marcados: ☐ Editar ☐ Copiar ☐ Borrar ☐ Exportar

## Tabla users

localhost/phpmyadmin/sql.php?server=1&db=socialauth&table=users&pos=0

Servidor: 127.0.0.1 » Base de datos: socialauth » Tabla: users

Mostrando filas 0 - 3 (total de 4, La consulta tardó 0,0000 segundos.)

SELECT \* FROM `users`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

+ Opciones

	id	name	email	email_verified_at	password	remember_token
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	8	MARIA ARELY GABRIEL ROMERO	grmo160524@upemor.edu.mx	NULL	\$2y\$10\$HtZtd/8osg9azTmqFU8qfxu6fBQ0iQn44hWlz882V6V...	NULL
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	9	Jose	sgjo168325@upemor.edu.mx	NULL	\$2y\$10\$GPugGTCbAYMKVKPn.pv54.CzhZGMQhnZsjVKIR2XhK7...	NULL
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	10	maria arely gabriel romero	mariz_magr@gmail.com	NULL	NULL	NULL
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	11	Mary Biel	mariz_gaby@hotmail.com	NULL	NULL	NULL

Seleccionar todo | Para los elementos que están marcados: ☐ Editar ☐ Copiar ☐ Borrar ☐ Exportar

## Tabla proveedores

localhost/phpmyadmin/sql.php?server=1&db=socialauth&table=proveedores&pos=0

Servidor: 127.0.0.1 » Base de datos: socialauth » Tabla: proveedores

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0000 segundos.)

SELECT \* FROM `proveedores`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

+ Opciones

	id	nombre	direccion_id	telefono	created_at	updated_at
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	Mary	1	3092781	2019-03-29 09:46:03	2019-03-29 09:46:03
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	6	Carlitos	1	3092781	NULL	NULL

Seleccionar todo | Para los elementos que están marcados: ☐ Editar ☐ Copiar ☐ Borrar ☐ Exportar



## Tabla ventas

localhost/phpmyadmin/sql.php?server=1&db=socialauth&table=ventas&pos=0

phpMyAdmin

Reciente Favoritas

- Nueva
- blog
- cinema
- information\_schema
- libreria
- music
- mysql
- performance\_schema
- phpmyadmin
- socialauth
- compras
- detalle\_compras
- detalle\_ventas

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Más

Mostrando filas 0 - 2 (total de 3, La consulta tardó 0,0000 segundos.)

SELECT \* FROM `ventas`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

+ Opciones

			id	fecha	total_venta	user_id	created_at	updated_at			
<input type="checkbox"/>	Editar	<input type="checkbox"/>	Copiar	<input type="checkbox"/>	Borrar	4	2019-03-30	105	8	2019-03-30 03:47:32	2019-03-30 04:17:40
<input type="checkbox"/>	Editar	<input type="checkbox"/>	Copiar	<input type="checkbox"/>	Borrar	5	2019-03-01	175	8	2019-03-30 04:18:17	2019-03-30 05:13:51
<input type="checkbox"/>	Editar	<input type="checkbox"/>	Copiar	<input type="checkbox"/>	Borrar	6	2019-03-06	140	9	2019-03-30 04:21:11	2019-03-30 05:12:16

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

## Tabla compras

localhost/phpmyadmin/sql.php?server=1&db=socialauth&table=compras&pos=0

phpMyAdmin

Reciente Favoritas

- Nueva
- blog
- cinema
- information\_schema
- libreria
- music
- mysql
- performance\_schema
- phpmyadmin
- socialauth
- compras
- detalle\_compras

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Más

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0000 segundos.)

SELECT \* FROM `compras`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

+ Opciones

			id	fecha	total_compra	created_at	updated_at			
<input type="checkbox"/>	Editar	<input type="checkbox"/>	Copiar	<input type="checkbox"/>	Borrar	4	2019-03-29	175	2019-03-30 03:40:12	2019-03-30 05:02:47
<input type="checkbox"/>	Editar	<input type="checkbox"/>	Copiar	<input type="checkbox"/>	Borrar	5	2019-03-22	245	2019-03-30 03:40:21	2019-03-30 04:57:28

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

## Tabla unidades

localhost/phpmyadmin/sql.php?server=1&db=socialauth&table=unidades&pos=0

phpMyAdmin

Reciente Favoritas

- Nueva
- blog
- cinema
- information\_schema
- libreria
- music
- mysql
- performance\_schema
- phpmyadmin
- socialauth
- compras
- detalle\_compras

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Más

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0000 segundos.)

SELECT \* FROM `unidades`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

+ Opciones

			id	unidad	created_at	updated_at			
<input type="checkbox"/>	Editar	<input type="checkbox"/>	Copiar	<input type="checkbox"/>	Borrar	1	Escolar	NULL	NULL
<input type="checkbox"/>	Editar	<input type="checkbox"/>	Copiar	<input type="checkbox"/>	Borrar	2	Oficina	NULL	NULL

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

## 5.7 Dompdf y Charts

### Reporte de ventas

PAPER SHÖP

PAPER SHOP

Jiutepec, Morelos

2019-04-01 08:37:34.494614

Usuario: MARIA ARELY GABRIEL ROMERO

Puesto: Administrador

E-mail: grmo160524@upemor.edu.mx

Reporte de Ventas

Fecha inicio: 2019-04-01 - Fecha final: 2019-04-30

#	FECHA	PRODUCTO	PRECIO	CANTIDAD	TOTAL
8	2019-04-01	Plumas	\$ 59	1	\$ 59
8	2019-04-01	USB	\$ 130	5	\$ 650
8	2019-04-18	Cuaderno	\$ 35	1	\$ 35
TOTAL VENTA: \$ 744.00					

FIRMA

Ilustración 24 Reporte de ventas

En la Ilustración 24 se muestra un ejemplo de un reporte pdf de ventas, el cual muestra las ventas que se encuentran entre las fechas 2019-04-01 - 2019-04-30 e información del usuario (nombre, puesto y correo). Las ventas se muestran dentro de una tabla con su información correspondiente y por último se muestre el total de la venta que en este caso es de \$744.00.

## Reporte de compras

PAPER SHÖP

PAPER SHOP

Jiutepec, Morelos

2019-04-01 09:51:06.863945

Usuario: MARIA ARELY GABRIEL ROMERO

Puesto: General

E-mail: grmo160524@upemor.edu.mx

Reporte de Compras

Fecha inicio: 2019-04-01 - Fecha final: 2019-04-05

#	FECHA	PRODUCTO	PRECIO	CANTIDAD	TOTAL
9	2019-04-02	Cuaderno	\$ 35	3	\$ 105
TOTAL COMPRA:					\$ 105.00

FIRMA

*Ilustración 25 Reporte de compras*

En la Ilustración 25 se muestra de igual manera el archivo pdf pero de los reportes de las compras. En este también se muestra información del usuario y se imprimen las compras realizadas entre las fechas 2019-04-01 – 2019-04-05. En este caso solo se encuentra una compra realizada el 02 de Abril con un total de compra de \$105.00.



## Reporte de productos

**PAPER SHÖP**

**PAPER SHOP**  
Jiutepec, Morelos

2019-04-01 09:58:34.471612

### Reporte de Productos

#	Nombre	Marca	Descripción	Cantidad	Precio	Unidad	Imagen
25	BlockHojas	Bic	Paquete de hojas blancas tamaño carta (100 piezas)	4	\$ 95	Oficina	
24	Cuaderno	Scribe	Cuaderno color azul tamaño oficio (cuadro grande)	3	\$ 35	Escolar	
23	USB	AAA	Memoria usb color rojo 1 pieza	4	\$ 130	Escolar	
22	Plumas	Bic	Paquete de plumas de colores (10 piezas)	-1	\$ 59	Escolar	
TOTAL PRODUCTOS: 4							

Ilustración 26 Reporte de productos

En la Ilustración 26 se puede ver el reporte de los productos, el cual se muestra de manera ordenada dentro de una tabla, definiendo las características de cada producto incluyendo su imagen de muestra. En este reporte se obtienen todos los elementos sin definir rangos y en la parte final de la tabla se muestra el total de productos que existen por el momento.

## Gráfica de ventas y compras

Productos vendidos en el mes de Abril

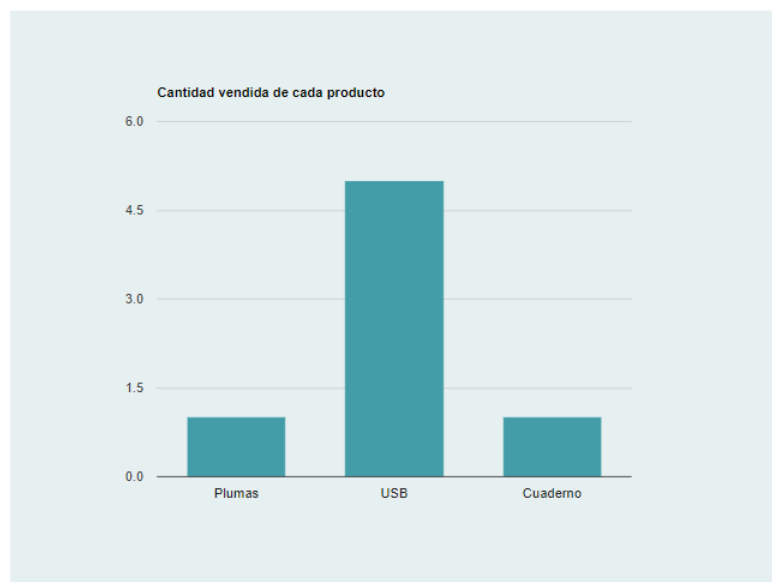


Ilustración 27 Gráfica ventas abril

Productos comprados en el mes de Marzo

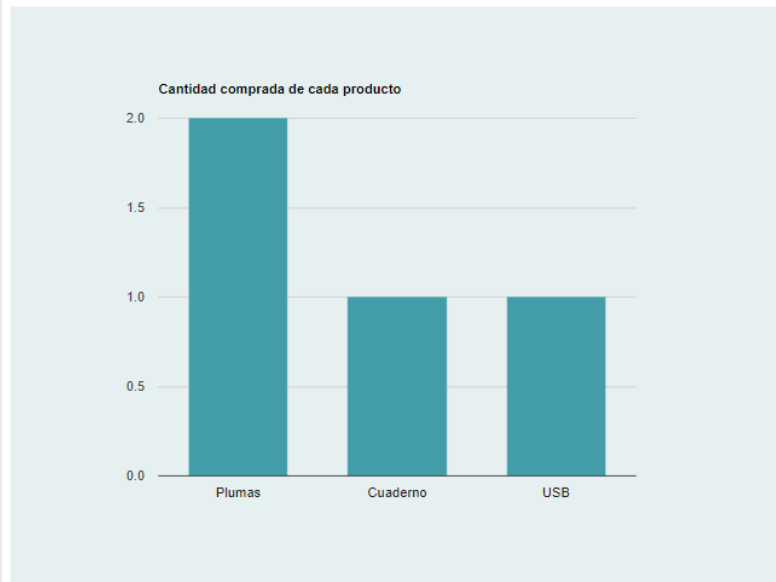


Ilustración 28 Gráfica compras marzo

En la Ilustración 28 se puede ver el ejemplo de una gráfica de ventas en el mes de abril donde se muestran los productos, plumas, usb y cuaderno donde se tienen ventas de un paquete de plumas, 5 usb y un cuaderno.

En la Ilustración 27 se puede ver el ejemplo de una gráfica de compras en el mes de marzo donde se muestran los productos, plumas, usb y cuaderno donde se tienen compras de dos paquetes de plumas, una usb y un cuaderno.

## 6. CONCLUSIÓN

El desarrollo de páginas web es un tema que ha ido evolucionando con el paso de los años, tanto que, de pasar a desarrollar en un block de notas, se han creado herramientas que permiten a los desarrolladores utilizar menos tiempo cuando se trata de una página web. Algunas herramientas son como por ejemplo Sublime Text 3 (herramienta con la que se trabajó el proyecto), que nos permite abrir proyectos completos para saber dónde está cada carpeta, además nos permite tener varias pestañas a la vez. Incluso se han creado varias plantillas con distintos diseños que nos permiten tener una base para la página y así no empezar de cero.

Unas de las herramientas que también se han ido desarrollando son los frameworks, como lo es Laravel que son programas independientes que permiten trabajar de una manera más fácil y completa ya que incluyen varios componentes que incluso nos facilitan ya partes de código como lo fue el Login que, con solo utilizar una línea de comando, crea carpetas con validaciones incluidas. Un ejemplo es las validaciones de las contraseñas o del correo.

El desarrollo de la página web con Laravel fue un nuevo reto para todos donde se aprendieron nuevos temas que a pesar que por ahora son complicados, en un futuro nos facilitarán el trabajo ya que no solo existe este framework sino que existen más con diferentes componentes pero es bueno ir acostumbrándose a trabajar con este tipo de herramientas.

Los usos de funciones dentro de este proyecto fueron muy interesantes, una de ellas fue el inicio de sesión con redes sociales ya que es algo que ya está implementado por las aplicaciones y solo se necesitan las credenciales para permitir al usuario un inicio más fácil sin necesidad de hacerle gastar tiempo, ya que varios usuarios por el mismo tiempo en ocasiones prefieren no ingresar al sistema solo por esos pequeños detalles.



A pesar del poco tiempo que se tuvo, pensamos que el sistema fue un éxito ya que obtuvieron varios aprendizajes al transcurso del desarrollo como fueron los Traits, los reportes y las gráficas. En general el desempeño fue lo esperado ya que visualmente el sistema quedó agradable y sobre todo funcional en casi su totalidad. Además, se aprendió una nueva herramienta que jamás había sido utilizada en cuatrimestres anteriores por lo que es un avance para nuestro crecimiento como programadores.

Como puntos para mejorar se incluye el trabajo en equipo ya que en ocasiones faltó buena comunicación que a parte por tiempo causó entregas tardías en algunas evidencias. Otro punto que se puede tomar es tratar de aprender nuevas herramientas para que nuestro conocimiento sea más extenso y de esta manera se tengan más oportunidades en el área laboral.

Por último, pero no menos importante se da un agradecimiento a la profesora del curso ya que a pesar de todo tuvo paciencia con el grupo y creemos que todos llevaremos un buen recuerdo para seguir teniendo sed de aprendizaje.

## 7. BIBLIOGRAFÍA



[1] GitBook (2016). “Laravel-5”. Lyon, Francia.  
Link: <https://richos.gitbooks.io/laravel-5/introduction.md>

[2] Miguel Ángel (2018). “Crear un proyecto Laravel con Composer”.  
Link: <https://desarrolloweb.com/articulos/crear-proyecto-laravel-composer.html>

[3] Duilio Palacios (2016). “Registro, login y recuperación con el comando Auth en Laravel 5.2”.  
Link: <https://styde.net/registro-login-y-recuperacion-de-clave-con-el-comando-makeauth-en-laravel-5-2/>

[4] Jessica Villa (2018). “Traits y como hacer funciones genéricas con Laravel 5”. EU.  
Link: <https://medium.com/@suhkha/es-traits-y-como-hacer-funciones-gen%C3%A9ricas-con-laravel-5-75f88246a6a5>