

**UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE
DEPARTAMENTO DE INGENIERÍA Y ARQUITECTURA**



**MATERIA:
TÉCNICAS DE PROGRAMACIÓN A INTERNET**

**DOCENTE:
ING. XENIA PEÑATE**

INTEGRANTES:

**Álvarez Morán, Cristina Soledad-AM17004
Martínez Pleitez, Melissa Maritza-MP17001
Vargas Morán, Mauricio Enrique-VM18042
Tepas Mazariego, Kenia Stephanie-TM17013**

**CONTENIDO:
PRÁCTICA - III**

**FECHA DE ENTREGA:
JUNIO 04, 2021.**

CREACIÓN DE PROYECTO CON LARAVEL:

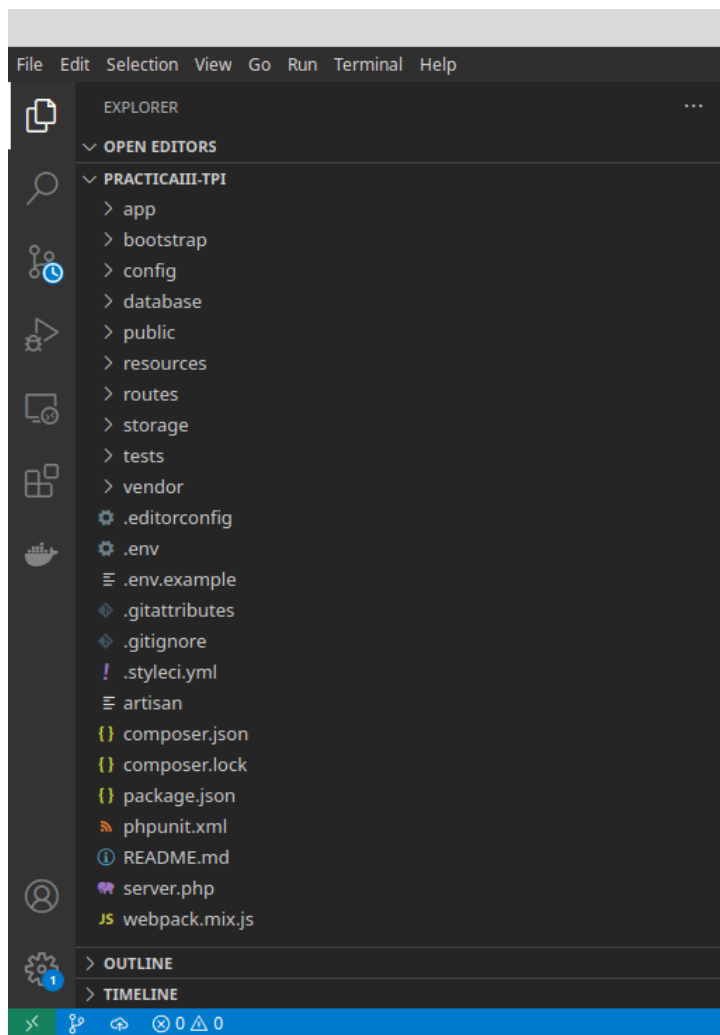
Previamente tener:

- Un entorno de desarrollo web: Apache, IIS, Nginx, PHP 5.3 o superior
- Base de datos: MySQL, Sqlite, Postgresql o sqlserver, etc
- Composer que es una herramienta para dependencias de php.

Independientemente del sistema operativo, ya sea Windows o Linux colocando en el directorio y carpeta de preferencia:

```
composer create-project laravel/laravel PracticaIII-TPI
```

Su estructura al crear proyecto es:



PASOS PARA LA CREACIÓN DE LA BASE DE DATOS:

Los primeros comandos para crear la base fueron los siguientes:

```
create database ventas character set utf8 collate utf8_general_ci;  
CREATE USER 'venta' IDENTIFIED BY 'vendedor';  
GRANT USAGE ON *.* TO 'venta'@localhost IDENTIFIED BY 'vendedor';  
GRANT ALL privileges ON `ventas`.* TO 'venta'@localhost;  
FLUSH PRIVILEGES;
```

Teniendo como resultado las siguientes tablas:

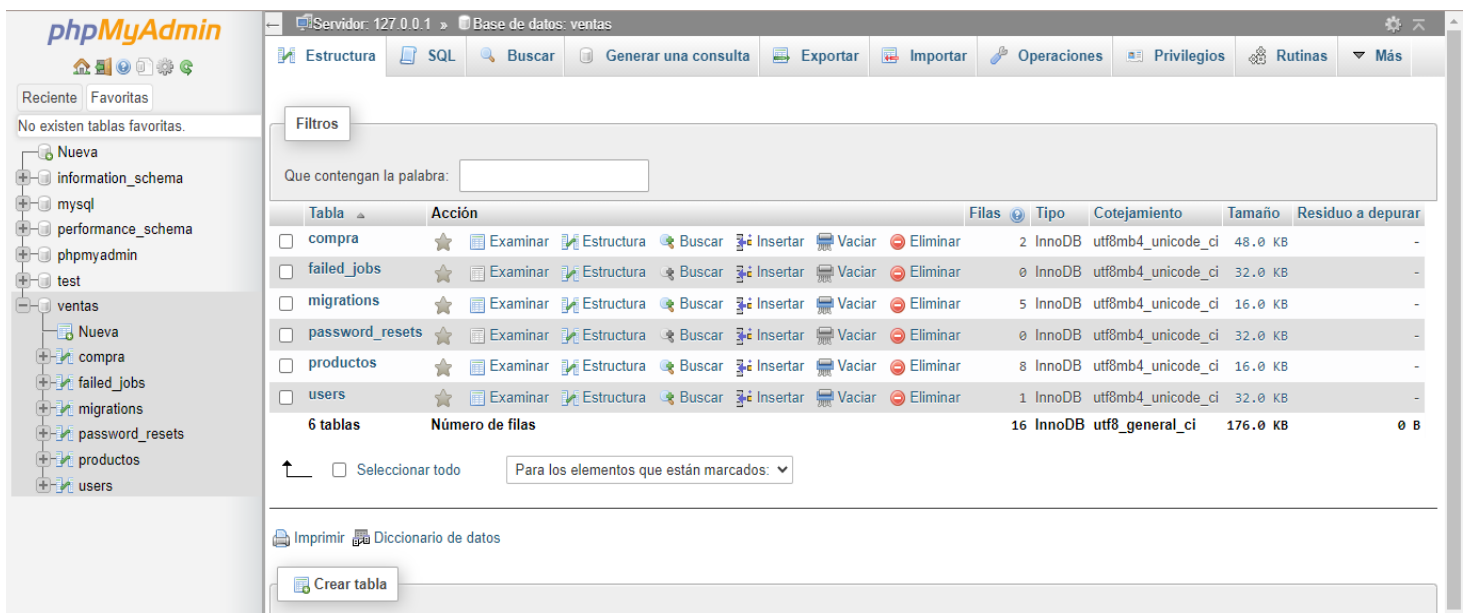


Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> compra	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_unicode_ci	48.0 KB	-
<input type="checkbox"/> failed_jobs	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
<input type="checkbox"/> migrations	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
<input type="checkbox"/> password_resets	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
<input type="checkbox"/> productos	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	8	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
<input type="checkbox"/> users	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
6 tablas	Número de filas	16	InnoDB	utf8_general_ci	176.0 KB	0 B

PASOS PARA ENV.EXAMPLE:

Contamos con un archivo predeterminado de env.example en visual, del cual se debe tomar el contenido, copiarlo en un txt y guardarlo como .env, su estructura es:

```
APP_NAME=Laravel  
APP_ENV=local  
APP_KEY=base64:CIYC8DTaUMoTP6FV9/2zOOIryjZUOUCu+i29HO3azQU=  
APP_DEBUG=true  
APP_URL=http://localhost
```

```
LOG_CHANNEL=stack
```

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=ventas
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="\${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="\${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="\${PUSHER_APP_CLUSTER}"

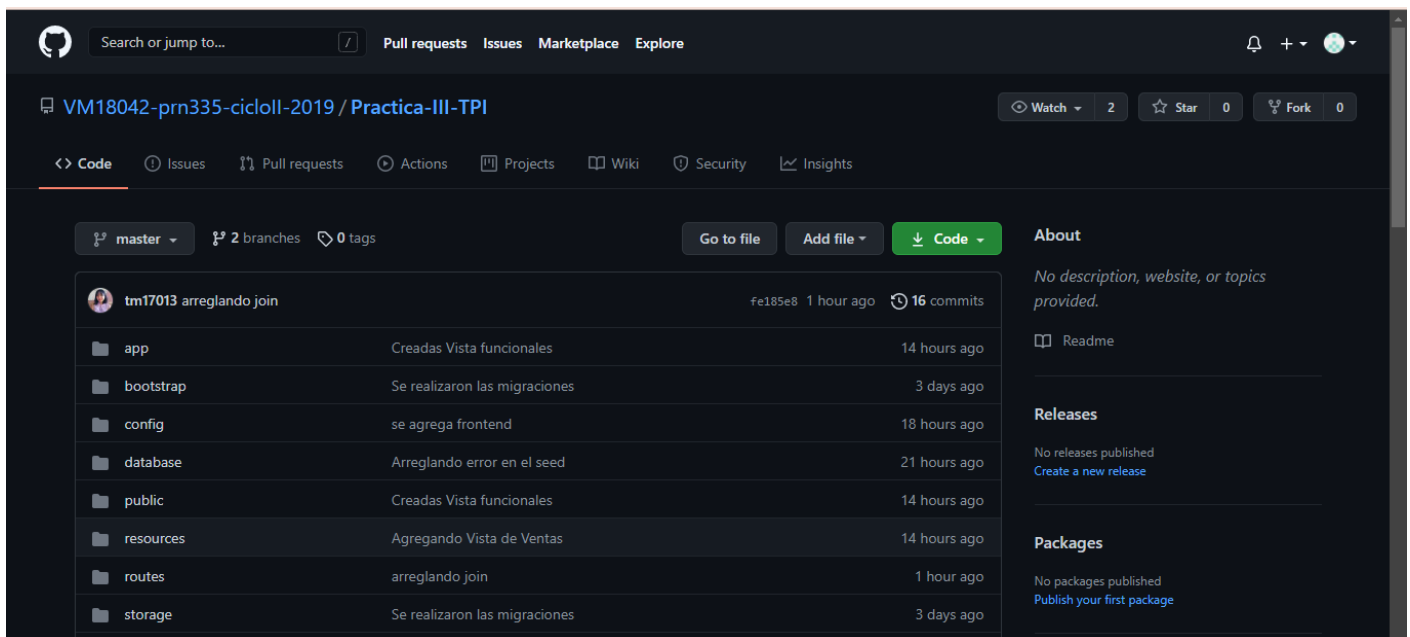
Especificando:

DB_DATABASE= (nombre de base de datos, el cual es ventas)
DB_USERNAME= (usuario que puede ser root o venta)
DB_PASSWORD= (contraseña de usuario)

PASOS PARA TRABAJAR DE FORMA COLABORATIVA:

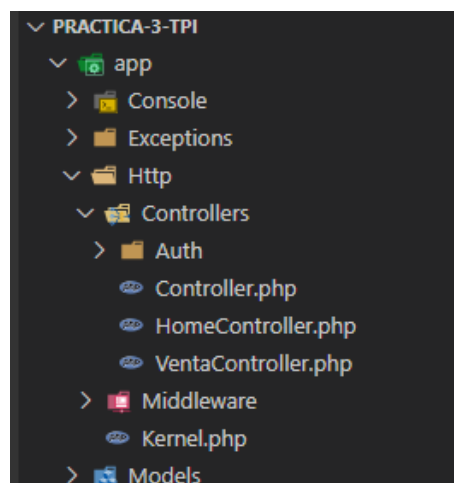
Como grupo se hizo uso de GitHub web para poder subir y bajar todos los cambios que se fueran realizando en el trabajo.

<https://github.com/VM18042-prn335-ciclolI-2019/Practica-III-TPI>



PASOS PARA PODER DESARROLLAR EL PROYECTO:

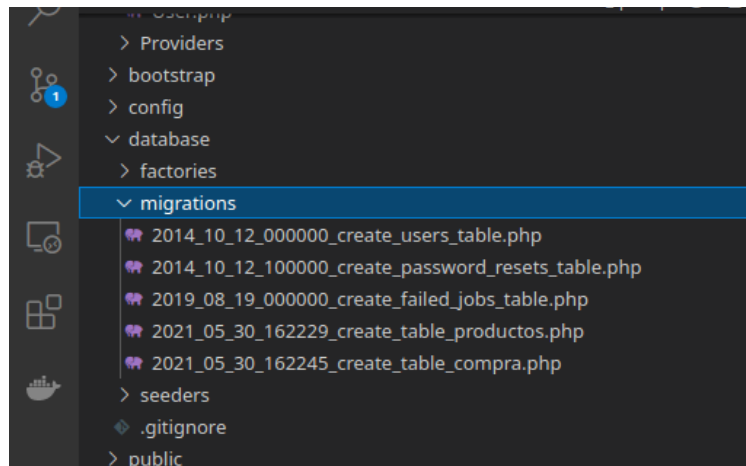
Primero se debe mencionar que utilizamos Visual Studio Code para poder llevarlo a cabo, PHPMyAdmin fue utilizado para crear la base con ayuda del SQL. Automáticamente se crean carpetas que están desplegadas en el visual como las siguientes:



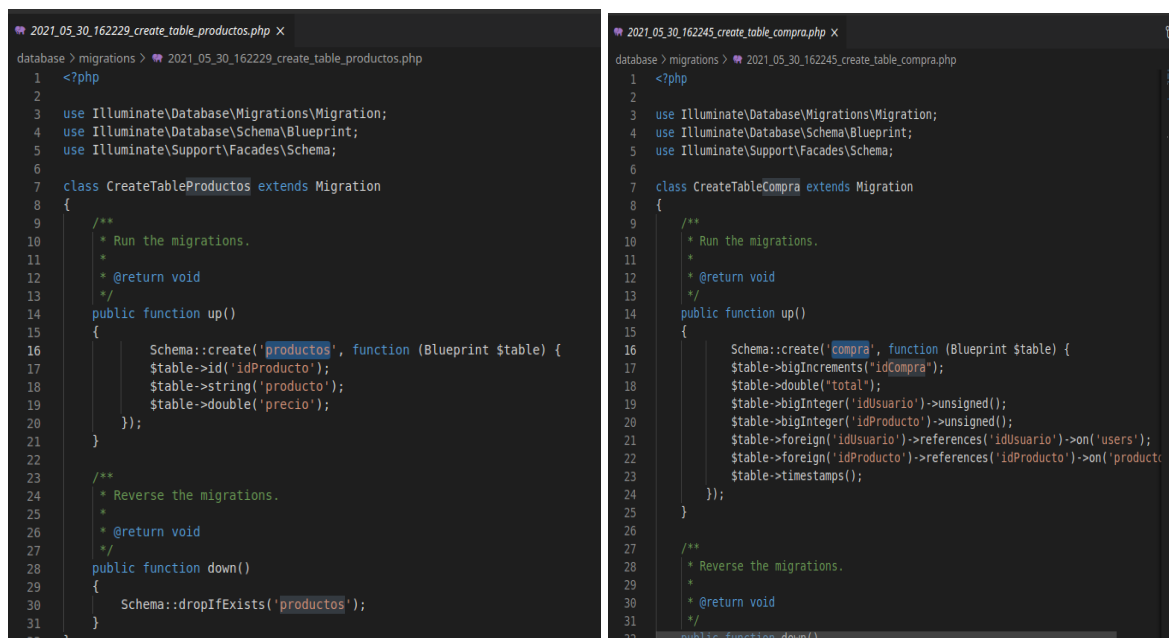
CONSTRUCCION DEL BACKEND:

MIGRATIONS:

Teniendo la parte de creación de base de datos, es necesario hacer las migraciones, utilizamos php artisan migrate.



Así sería su estructura; con respecto al contenido se creó la migración create_table_productos que corresponde a la tabla productos y la migración create_table_compra que corresponde a la tabla llamada compra:



MODELS:

Por defecto en nuestro programa está User.php correspondiente a Usuarios en App\Models. Luego se crearon los demás modelos, uno para Compra llamado

Comprar.php y otro para Producto llamado Productos.php. Se creó con php artisan make:model nombremodulo.

Esa es la estructura, cabe recalcar que se colocó la llave primaria para especificar la tabla de users como :protected \$primaryKey = 'idUserario';

```
User.php
app > Models > User.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Contracts\Auth\MustVerifyEmail;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Illuminate\Notifications\Notifiable;
9
10 class User extends Authenticatable
11 {
12     use HasFactory, Notifiable;
13
14     /**
15      * The attributes that are mass assignable.
16      *
17      * @var array
18      */
19     protected $fillable = [
20         'name',
21         'email',
22         'password',
23     ];
24
25     /**
26      * The attributes that should be hidden for arrays.
27      *
28      * @var array
29      */
30     protected $hidden = [
31         'password',
32     ];
33
34     /**
35      * The attributes that should be cast to native types.
36      *
37      * @var array
38      */
39     protected $casts = [
40         'email_verified_at' => 'datetime',
41     ];
42
43     protected $primaryKey = 'idUserario';
44
45
46
```

En Compras.php se especificó la tabla compra con protected \$table = 'compra'; y en Productos.php se colocó public \$timestamps = false; y const UPDATED_AT = null;

```
EXPLORER
OPEN EDITORS
  Productos.php app/Models
PRACTICA-III-TPI
  app
    Console
    Exceptions
    Http
    Models
      Compras.php
      Productos.php
      User.php
    Providers

Productos.php
app > Models > Productos.php
1  <?php
2
3  namespace App\Models;
4
5  //use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Productos extends Model
9  {
10     public $timestamps = false;
11     const UPDATED_AT = null;
12 }
13
```

```
EXPLORER
OPEN EDITORS
  Compras.php app/Models
PRACTICA-III-TPI
  app
    Console
    Exceptions
    Http
    Models
      Compras.php
      Productos.php
      User.php
    Providers
  bootstrap

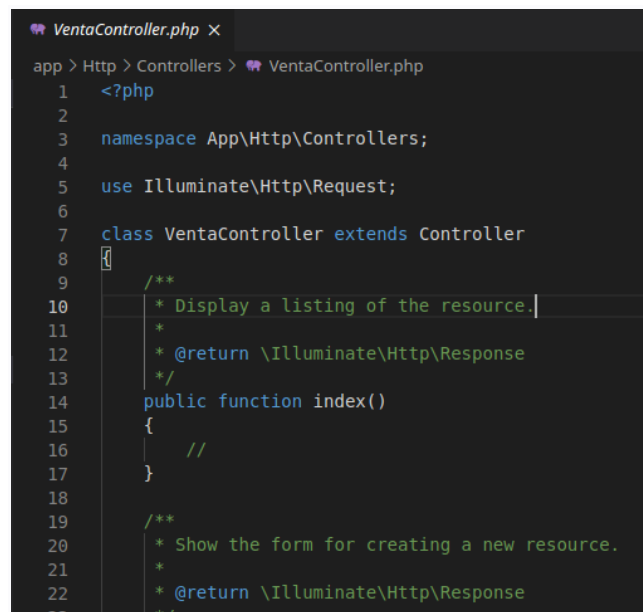
Compras.php
app > Models > Compras.php
1  <?php
2
3  namespace App\Models;
4
5  //use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Compras extends Model
9  {
10     protected $table = 'compra';
11 }
12
```

CONTROLLERS:

Para la ruta App\Http\Controllers podemos ver que ya están predeterminados dos Controller, el HomeController.php y Controller.php lo cual no se explicaran porque no se realizaron cambios.

Creamos un Controller para agrupar la lógica de manejo de solicitudes relacionadas en una sola clase. Lo cual, se llamo VentaController y para crearlo utilizamos php artisan make:controller VentaController --resource

Lo que hace --resource es desplegar recursos para hacer un CRUD en Laravel, pero no es lo que se pretende, así que el VentaController.php también se podía crear con php artisan make:controller VentaController.



```
VentaController.php x
app > Http > Controllers > VentaController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class VentaController extends Controller
8  {
9      /**
10       * Display a listing of the resource.
11       *
12       * @return \Illuminate\Http\Response
13       */
14     public function index()
15     {
16         //
17     }
18
19     /**
20      * Show the form for creating a new resource.
21      *
22      * @return \Illuminate\Http\Response
23      */
```

ROUTES:

En esta parte se modificó y trabajo en routes/web.php, el cual definimos las rutas de nuestra aplicación web, que son las que consultan nuestros usuarios desde el navegador.

Se agrego los módulos:

```
use App\Models\Productos
```

```
use App\Models\Compras
```

Se hicieron las respectivas actualizaciones:

```
Auth::routes();
```

```
Route::get('/home',[App\Http\Controllers\HomeController::class,'index'])->name('home');
```

```
Auth::routes();
```



```
Route::get('/home',[App\Http\Controllers\HomeController::class,'index']->name('home'));

```

```
Route::get("Productos", function(){
    $Productos= Productos::all();
    //↑-----↓
    return view('productos', compact('Productos'));
})->name('RutaListaProducto');
```

```
Route::get('Productos/crear',function(){
    return view('crear');
})->name('RutaCrearProducto');
```

```
Route::get('Ventas', function(){
    $NuevaVenta=Compras::join('productos','compra.idProducto','=', 'productos.idProducto')->where('idUsuario',auth()->id())->get();
    return view('ventas',compact('NuevaVenta'));
})->name('RutaProductosVendidos');
```

```
Route::post('Productos', function(Request $request){
    $request->all();
    $NuevoProducto= new Productos;
    //Nombre del campo de la tabla↓-----↓Nombre del input en el formulario
    $NuevoProducto->producto=$request->input('Nombre');
    $NuevoProducto->precio=$request->input('Precio');
    $NuevoProducto->save();
    return redirect()->route('RutaListaProducto');
})->name('RutaGuardarProducto');
```

```
Route::post('compras',function(Request $request){
    $request->all();
    $NuevaCompra= new Compras;
    $NuevaCompra->total=$request->input('total');
    $NuevaCompra->idUsuario=auth()->id();
    $NuevaCompra->idProducto=$request->input('idProducto');
    $NuevaCompra->save();
    return redirect()->route('RutaProductosVendidos');
})->name('RutaComprasProductos');
```

Se ve de esta manera:

```
web.php M x
routes > web.php
18
19 Route::get('/', function () {
20     return view('welcome');
21 });
22
23 //Auth::routes();
24
25 //Route::get('/home', 'HomeController@index')->name('home');
26
27
28 Auth::routes();
29
30 Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');
31
32 Auth::routes();
33
34 Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');
35
36 Route::get('Productos', function(){
37     $Productos= Productos::all();
38     //-----
39     return view('productos', compact('Productos'));
40 })->name('RutaListaProducto');
41
42 Route::get('Productos/crear',function(){
43     return view('crear');
44 })->name('RutaCrearProducto');
45
46 Route::get('Ventas', function(){
47     $NuevaVenta= Compras::join('productos','comptra.idProducto','=','productos.idProducto')->wher
48     return view('ventas',compact('NuevaVenta'));
49 })->name('RutaProductosVendidos');
```

CONSTRUCCIÓN DEL FRONTEND:

Con parte del backend previamente creada, comenzamos a trabajar de forma paralela el frontend. Para este hemos utilizado bootstrap y varias de sus funciones. Entre algunas de estas contamos con cards, navbar y footer.

Para comenzar a desarrollar el código utilizamos la arquitectura básica de html:5 !DOCTYPE y también de una card doble junto con todos sus componentes de CSS.

```
File Edit Selection View Go Run Terminal Help welcome.blade.php - practice-3-tpi - Visual Studio Code
resources > views > welcome.blade.php
Search
<head>
342 <title></title>
343 <meta charset="utf-8">
344 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
345 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aohXA+058RXPxPg
346 </head>
347 <body>
348 <div class="container">
349 <div class="row">
350 <div class="col-md">
351 <div class="card">
352 
354 <h4 class="card-title">¡Conoce Nuestros Productos!</h4>
355 <p class="card-text">En nuestra farmacia encontrarás los mejores productos con precios accesibles. Contamos con un mercado de medic
356 </div>
357 </div>
358 </div>
359 </div>
360 <div class="col-md">
361 <div class="card">
362 <div class="card-body">
363 <p class="card-text">¡Estamos dispuestos a servirte de la mejor manera! Regístrate gratis en nuestro sitio web y obtén descuentos e
364 </div>
365 
Universidad de El Salvador / @Tecnicas De Programación A Internet
</div>
</html>
```

Para la navbar y los botones de Login y Register utilizamos una navbar structure más los HREF que contienen los botones para poder conectar a la tabla asociada.

```
welcome.blade.php X
resources > views > welcome.blade.php
309 <body>
310 <nav class="navbar navbar-light bg-light">
311   <div class="container-fluid">
312     <a class="navbar-brand" href="#">
313       
314       Farmacia El Avast
315     </a>
316   </div>
317 </nav>
318 </body>
319
320
321 <body class="antialiased">
322
323
324 <div
325   class="relative flex items-top justify-center min-h-screen bg-gray-900 dark:bg-gray-900 sm:items-center py-4 sm:pt-0">
326   @if (Route::has('login'))
327     <div class="hidden fixed top-0 right-0 px-6 py-4 sm:block">
328       @auth
329         <a href="{{ url('/Productos') }}" class="text-sm text-gray-700 underline">Home</a>
330       @else
331         <a href="{{ route('login') }}" class="text-sm text-gray-700 underline">Log in</a>
332
333         @if (Route::has('register'))
334           <a href="{{ route('register') }}" class="ml-4 text-sm text-gray-700 underline">Register</a>
335         @endif @endauth
336     </div>
337   @endif
338
```

PASOS PARA EDITAR EL NOMBRE CON NAVBAR:

El elemento de bootstrap utilizado para este cambio de “Laravel” a “Farmacia El Avast” fue una navbar con el logo de la universidad por referencia a la práctica. El link adjuntado en el HREF pertenece al logo de la UES.

```
<nav class="navbar navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">
      
      Farmacia El Avast
    </a>
  </div>
</nav>
</body>
```

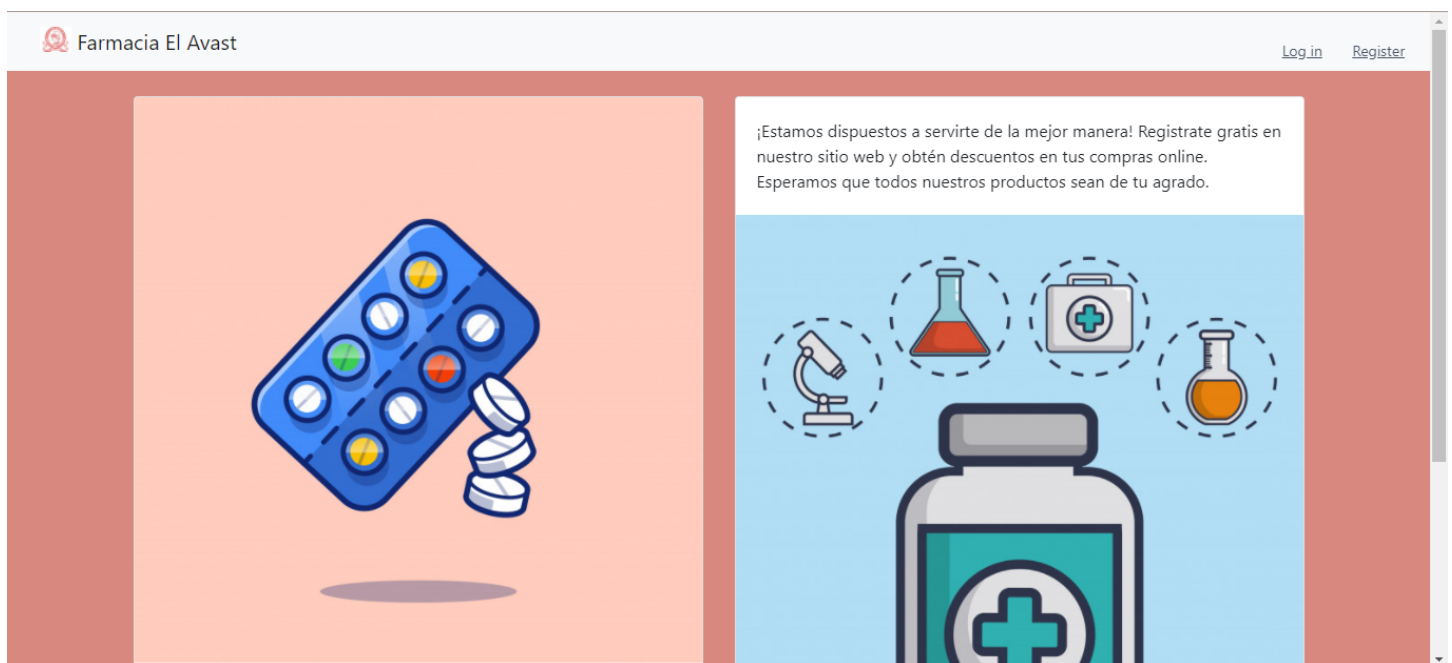
PASOS PARA EDITAR EL LOGIN:

Para esta parte del proyecto en específico hicimos uso de UI en Laravel 6, se tienen comandos para instalarlo los cuales son:

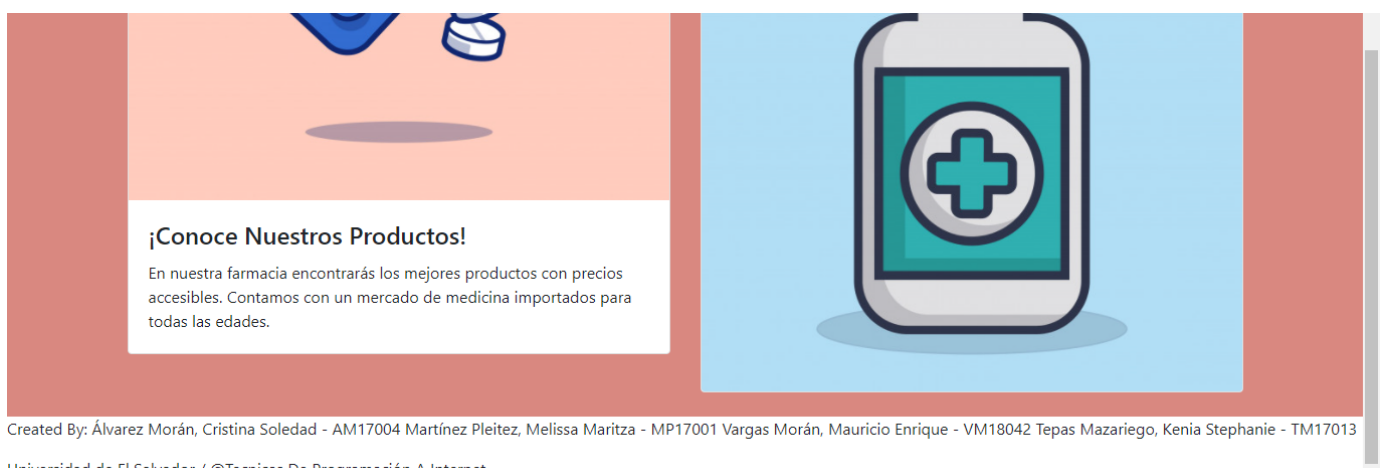
```
composer require laravel/ui  
php artisan ui bootstrap  
php artisan ui bootstrap --auth
```

RESULTADO DEL FRONTEND Y LAS VISTAS:

Para la vista del frontend se obtuvieron los siguientes resultados:
Como página principal:



Cómo footer:



Página de Login:

El Avast Login Register

Login

E-Mail Address

Password

☐ Remember Me

Login

Forgot Your Password?

Para Forgot Your Password:

El Avast Login Register

Reset Password

E-Mail Address

Send Password Reset Link

Página para Register:

El Avast Login Register

Register

Name

E-Mail Address

Password

Confirm Password

Register

Página de compras de productos:

El Avast

Cristina ▾

Listado de Productos		Ver Ventas
Producto	Precio	Acciones
Alcohol 90° 750ml	4	Comprar
Alcohol Gel Savon 500ml	2.5	Comprar
AERIUS 5MG X 1 TABLETA	1.99	Comprar
AMOXICILINA L.S. 500MG X 1 CAPSULA	0.25	Comprar
LORATADINA LS 10MG X 1 TABLETA	0.5	Comprar
ACIMED 15MG X 1 SOBRE DE 2 CAPSULAS	0.5	Comprar
ABRILAR JARABE FRASCO X 100 ML	12.99	Comprar
Mascarilla KN95 Unidad	1.89	Comprar

Página de compras realizadas:

El Avast

Cristina ▾

Productos Comprados		Comprar Mas Productos
Cristina		
Producto	Precio	
LORATADINA LS 10MG X 1 TABLETA	0.5	
Mascarilla KN95 Unidad	1.89	
Alcohol 90° 750ml	4	

PASOS PARA PODER CORRER EL PROYECTO:

Con el proyecto ya finalizado, utilizamos los siguientes comandos para levantarlo:

```
php artisan migrate:fresh --seed
php artisan serve
```

Haciendo uso de: <http://127.0.0.1:8000/> Para mostrar el Home.