



Man ual del sistema



SENA Food-site es un aplicativo web, el cual está diseñado para permitirle a los usuarios ver los productos que se venden en el restaurante ubicado en el SENA complejo sur (carrera 30 con 1 de mayo), y así mismo adquirir y pagar dichos productos por medio del aplicativo, para luego recogerlos en el restaurante.

Requerimientos de hardware y software:

- Apache 2.4.47
- MySQL 5.7.33
- PHP 8.1.9
- phpMyAdmin 5.2.0
- Laragon 5.0



Se realiza la instalación de Laragon, en este caso se usó la versión 5.0, del cual usaremos phpMyAdmin, Apache y MySQL.

Al abrir Laragon nos encontraremos lo siguiente:



En este caso daremos “Iniciar Todo” para encender el Apache y el MySQL, una vez encendidos le daremos en “Terminal” para entrar a la terminal, allí introduciremos: “cd mysql”, después “mysql -u root -p” y al salir la contraseña le daremos simplemente enter, después crearemos nuestra base de datos para el proyecto, en este caso le pondremos foodsite, quedando algo así:



```
database.sql
C:\Users\> PC > Desktop > database.sql

49 CREATE TABLE `usuarioproducto` (
50   `id` int(11) NOT NULL,
51   `id_usuario` int(30) DEFAULT NULL,
52   `id_producto` int(30) DEFAULT NULL
53 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
54
55 ALTER TABLE `comentario`
56   ADD PRIMARY KEY (`id`),
57   ADD KEY `id_usuario` (`id_usuario`);
58
59 ALTER TABLE `historialpedidos`
60   ADD PRIMARY KEY (`id`),
61   ADD KEY `id_usuario` (`id_usuario`);
62
63 ALTER TABLE `producto`
64   ADD PRIMARY KEY (`id`);
65
66 ALTER TABLE `productohistorial`
67   ADD PRIMARY KEY (`id`),
68   ADD KEY `id_producto` (`id_producto`),
69   ADD KEY `id_historialpedido` (`id_historialpedido`);
70
71 ALTER TABLE `tipousuario`
72   ADD PRIMARY KEY (`id`);
73
74 ALTER TABLE `usuario`
75   ADD PRIMARY KEY (`id`),
76   ADD KEY `id_tipousuario` (`id_tipousuario`);
77
78 ALTER TABLE `usuarioproducto`
79   ADD PRIMARY KEY (`id`),
80   ADD KEY `id_usuario` (`id_usuario`),
81   ADD KEY `id_producto` (`id_producto`);
82
83 ALTER TABLE `comentario`
84   MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
85
86 ALTER TABLE `historialpedidos`
87   MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
88
89 ALTER TABLE `producto`
90   MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
91
92 ALTER TABLE `productohistorial`
93   MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
94
```

```
+-----+
| Tables_in_foodsite |
+-----+
| categoria           |
| centroaprendizaje   |
| comentario          |
| disponibilidad      |
| estadopedido        |
| estadosoporte       |
| historialpedidos    |
| metodopago          |
| pedidos             |
| preparacionpedido   |
| producto            |
| productohistorial   |
| tipousuario         |
| usuario             |
| usuarioproducto     |
+-----+
```



En este caso se han creado las tablas, y las relaciones entre tablas tales como usuario- producto o historial-producto, así mismo a cada una de estas tablas se les añadió una llave primaria y se les indico que estas incrementaran de manera automática, a medida que se agreguen más registros.

```
database.sql
C:\Users\PC\Desktop > database.sql
95 ALTER TABLE `tipousuario`
96     MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
97
98 ALTER TABLE `usuario`
99     MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
100
101 ALTER TABLE `usuarioproducto`
102     MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
103
104 ALTER TABLE `comentario`
105     ADD CONSTRAINT `comentario_ibfk_1` FOREIGN KEY (`id_usuario`) REFERENCES `usuario` (`id`);
106
107 ALTER TABLE `historialpedidos`
108     ADD CONSTRAINT `historialpedidos_ibfk_1` FOREIGN KEY (`id_usuario`) REFERENCES `usuario` (`id`);
109
110 ALTER TABLE `productohistorial`
111     ADD CONSTRAINT `productohistorial_ibfk_1` FOREIGN KEY (`id_producto`) REFERENCES `producto` (`id`),
112     ADD CONSTRAINT `productohistorial_ibfk_2` FOREIGN KEY (`id_historialpedido`) REFERENCES `historialpedidos` (`id`);
113
114 ALTER TABLE `usuario`
115     ADD CONSTRAINT `usuario_ibfk_1` FOREIGN KEY (`id_tipousuario`) REFERENCES `tipousuario` (`id`);
116
117 ALTER TABLE `usuarioproducto`
118     ADD CONSTRAINT `usuarioproducto_ibfk_1` FOREIGN KEY (`id_usuario`) REFERENCES `usuario` (`id`),
119     ADD CONSTRAINT `usuarioproducto_ibfk_2` FOREIGN KEY (`id_producto`) REFERENCES `producto` (`id`);
120 COMMIT;
121
```

Ahora a las tablas que están relacionadas a otras se les añadió una llave foránea, para que tengan una relación entre sí a la hora de llenar los datos de los usuarios, las tablas ya creadas dentro de la terminal de Laragon se verán así:

```
MariaDB [proyecto00]> show tables;
+-----+
| Tables_in_proyecto00 |
+-----+
| comentario            |
| historialpedidos       |
| producto              |
| productohistorial      |
| tipoproducto          |
| usuario               |
| usuarioproducto       |
+-----+
7 rows in set (0.001 sec)

MariaDB [proyecto00]> describe usuario;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| Nombres    | varchar(30)   | YES  |     | NULL    |               |
| Apellidos  | varchar(30)   | YES  |     | NULL    |               |
| contraseña | varchar(30)   | YES  |     | NULL    |               |
+-----+
4 rows in set (0.010 sec)

MariaDB [proyecto00]> describe usuarioproducto;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| id_usuario | int(30)       | YES  |     | NULL    |               |
| id_producto | int(30)      | YES  |     | NULL    |               |
+-----+
3 rows in set (0.010 sec)

MariaDB [proyecto00]> describe tipoproducto;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| Nombre_usuario | varchar(30) | YES  |     | NULL    |               |
+-----+
2 rows in set (0.011 sec)

MariaDB [proyecto00]> describe producto;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| Nombre     | varchar(30)   | YES  |     | NULL    |               |
| Categoria  | varchar(30)   | YES  |     | NULL    |               |
| Precio     | int(30)       | YES  |     | NULL    |               |
+-----+
4 rows in set (0.010 sec)

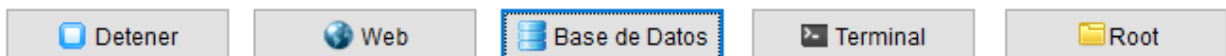
MariaDB [proyecto00]> describe productohistorial;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| id_producto | int(30)      | YES  |     | NULL    |               |
| id_historialpedido | int(30)      | YES  |     | NULL    |               |
+-----+
3 rows in set (0.010 sec)

MariaDB [proyecto00]> describe historialpedidos;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| Fecha      | int(30)       | YES  |     | NULL    |               |
| Cantidad_producto | int(30)      | YES  |     | NULL    |               |
| id_producto | int(30)      | YES  |     | NULL    |               |
| id_usuario | int(30)      | YES  |     | NULL    |               |
+-----+
5 rows in set (0.010 sec)

MariaDB [proyecto00]> describe comentario;
+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| Tipo       | varchar(30)   | YES  |     | NULL    |               |
| Comentario | varchar(100)  | YES  |     | NULL    |               |
| id_usuario | int(30)       | YES  |     | NULL    |               |
+-----+
4 rows in set (0.010 sec)

MariaDB [proyecto00]> _
```

Para ver las relaciones de la llave principal y la llave foránea debemos ingresar a phpMyAdmin.



Para ingresar debemos darle clic en el botón “Base de Datos” que nos llevara a la página de phpMyAdmin, donde seleccionaremos nuestra base de datos, que en este caso lleva el nombre de foodsite.



phpMyAdmin - Servidor: localhost:3306 - Base de datos: foodsite

Filtros

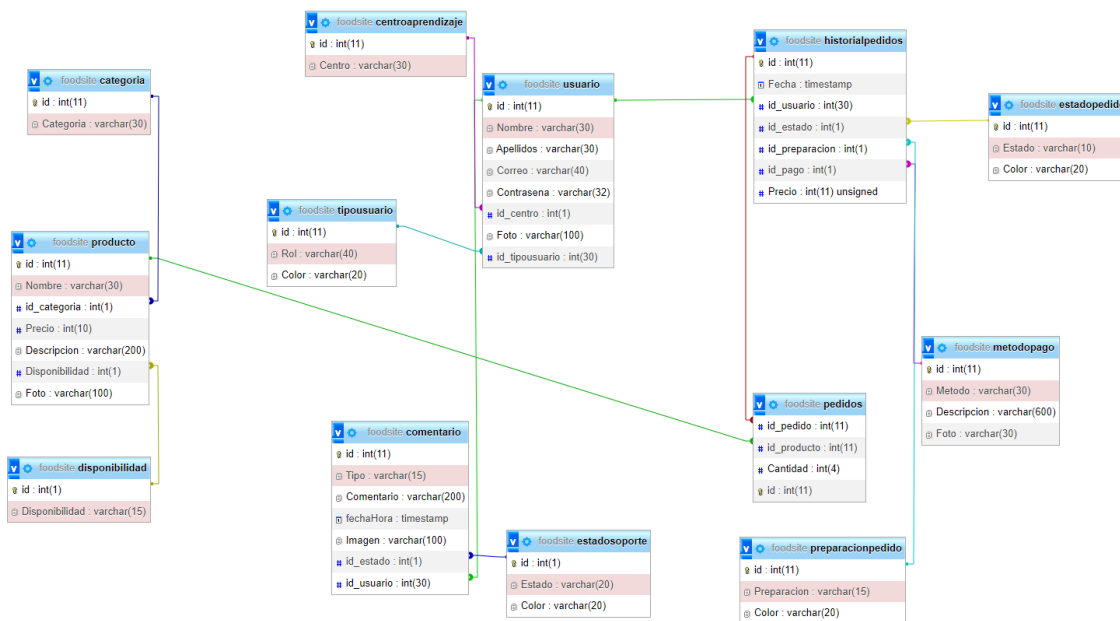
Que contengan la palabra:

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
categoria	Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	latin1_swedish_ci	16.0 KB	-
centroaprendizaje	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	latin1_swedish_ci	16.0 KB	-
comentario	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
disponibilidad	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	latin1_swedish_ci	16.0 KB	-
estadopedido	Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	latin1_swedish_ci	16.0 KB	-
estadosoporte	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	latin1_swedish_ci	16.0 KB	-
historialpedidos	Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	utf8mb4_general_ci	96.0 KB	-
metodopago	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	latin1_swedish_ci	16.0 KB	-
pedidos	Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	latin1_swedish_ci	48.0 KB	-
preparacionpedido	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	latin1_swedish_ci	16.0 KB	-
producto	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	48.0 KB	-
productohistorial	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
tipousuario	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	-
usuario	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	64.0 KB	-
usuarioproducto	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-

15 tablas Número de filas 43 InnoDB latin1_swedish_ci 528.0 KB 0 B

Seleccionar todo Para los elementos que están marcados

Aquí podemos ver las tablas que creamos con anterioridad; en la parte superior de la página donde dice más, le daremos en diagramas y nos mostrará lo siguiente:



Aquí se encuentran las tablas creadas junto con sus relaciones a otras tablas, indicando cual es la llave principal de cada tabla y cuál es la foránea.

Una vez generada la base de datos se puede empezar la creación de la página, en este caso iniciamos con el frontend del proyecto, manejando HTML, CSS y JavaScript (para algunas funciones del frontend). Iniciamos con la creación de un repositorio en GitHub (para trabajar cooperativamente), en este caso se manejan 3 ramas/sucursales en el repositorio del proyecto de GitHub, las cuales son la main (la cual se genera por defecto), frontend y master (la cual contiene el backend).



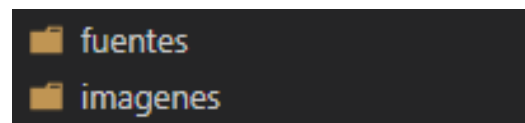
En la rama frontend se encuentran los archivos del aplicativo en una carpeta llamada SENA Food-site frontend.

fuente	Letra
imagenes	Icono
node_modules	index 2 con bootstrap
RegistroPrueba.html	Registro (en proceso)
Thumbs.db	Add files via upload
admin.html	correcciones
carrito.html	correccion en carrito
index2.html	correcciones
indexVendedor.html	correcciones
login.html	cambios anteriores
metodosPago.html	metodos de pago
mobile.js	Add files via upload
modificarUsuario.html	admin
package-lock.json	index 2 con bootstrap
package.json	index 2 con bootstrap
pedidos.html	Actualización archivo
qr.html	qr
registro.html	Registro
soporte.html	soporte técnico
styleRegistro.css	Responsive
styles.css	Merge branch 'Frontend' of https://github.com/Santiago713/Proyecto in...
styles.css.map	correcciones
styles.scss	Merge branch 'Frontend' of https://github.com/Santiago713/Proyecto in...
stylesCatalogo.css	boton buscador

Aquí se encuentran los siguientes archivos HTML:

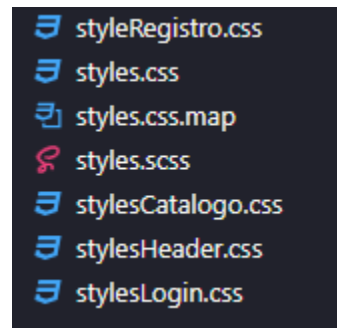
- admin.html
- carrito.html
- index2.html
- indexVendedor.html
- login.html
- metodosPago.html
- modificarUsuario.html
- pedidos.html
- qr.html
- registro.html
- RegistroPrueba.html
- soporte.html
- tablaErrores.html
- vistaProducto.html
- vistaProductoVendedor.html

Estos archivos están enlazados junto a archivos de tipo css, Estos manejan la manera en cómo se ve la página y son lo que verá principalmente el usuario; dentro de la carpeta principal encontramos otras carpetas las cuales son: fuentes, imágenes, que como indica su nombre contienen las imágenes y fuentes utilizadas en el desarrollo del frontend del aplicativo.





Se encuentran los siguientes archivos CSS:



En este caso vemos los estilos que se encuentran vinculados a los archivos HTML, en estos se maneja la manera en que se mostrará el contenido de las ventanas HTML.

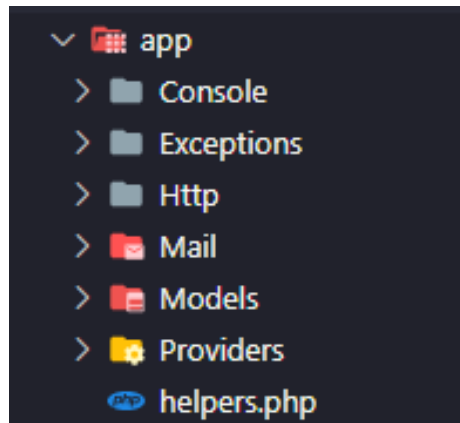
Hay menos archivos CSS debido a que algunos archivos HTML comparten características en la vista.

En la rama master, se encuentran todos los archivos usados para el backend, aquí también encontramos la base de datos, la cual lleva como nombre foodsite.sql

app	Imágenes
bootstrap	Laravel
config	Emails de verdad
database	información del usuario en las vistas
imagenes	Imágenes
lang/en	Laravel
php.ini	php.ini
public	Ignore de la carpeta storage
resources	Mensaje
routes	Arreglo en los middlewares
storage	Ignore de la carpeta storage
tests	Laravel
vendor	Imágenes
.editorconfig	Laravel
.env	login arreglado
.env.example	Revert "Revert "Revert "Se QultO cOmEnTaRiO""
.gitattributes	Laravel
.gitignore	Ignore de la carpeta storage
.htaccess	estilos
README.md	Laravel
artisan	Laravel
composer.json	SQL
composer.lock	Imágenes
foodsite.sql	Update foodsite.sql

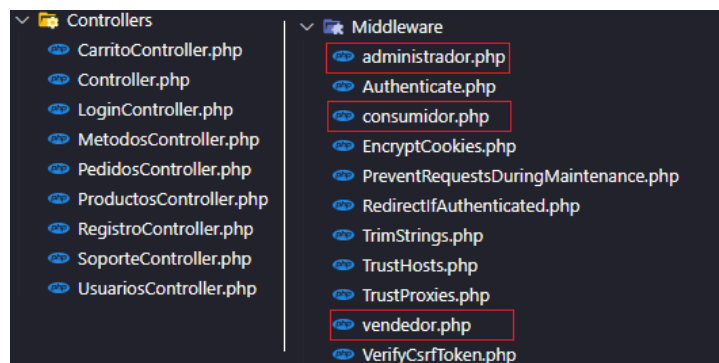


La primera carpeta que encontramos es “app”, aquí encontramos lo siguiente:

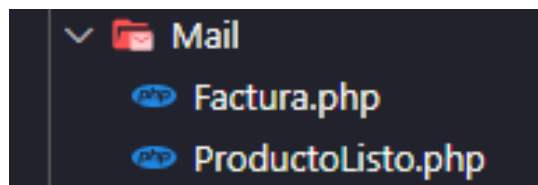


De estas carpetas se usó la carpeta HTTP la cual contiene los “middleware” y los “Controllers”, los Controllers nos sirven para definir las funciones de algunos botones o acciones del programa, tales como ver las páginas, registrarse, ingresar al aplicativo, modificar usuarios, agregar o modificar productos; mientras que los middlewares se encarga de autenticar o verificar el tipo de usuario (rol) que entró en la aplicación, para así mostrar diferentes acciones o restringir vistas dependiendo del usuario.

Tanto en los Controllers como en los Middlewares podemos ver archivos de tipo PHP, esto debido a que estos están conectados a la base de datos, por que como ya se menciona algunos de estos modifican, crean o eliminan registros de las bases de datos, dentro de los middlewares archivos encerrados en un rectángulo rojo son los que se encargan de verificar el tipo de usuario, y dependiendo de esto muestran ciertas funcionalidades ya definidas para cada usuario.



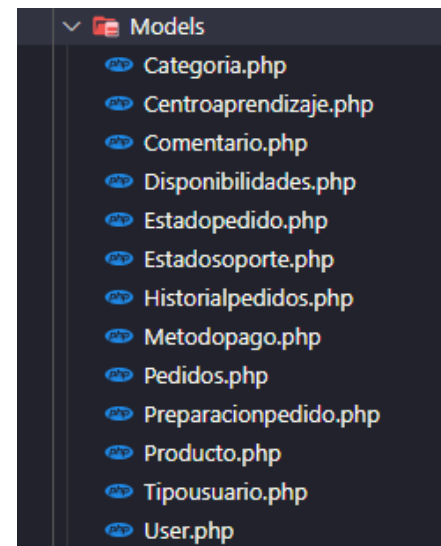
Otra carpeta que encontramos en app es “mail”.



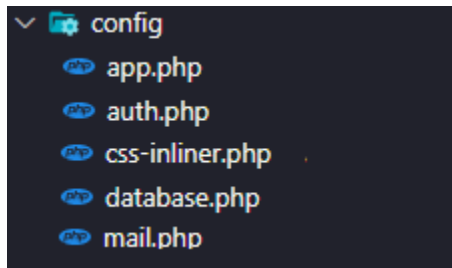
En esta encontramos dos archivos: Factura y ProductoListo, estos se encargan de decirle al aplicativo que debe ir en el correo que se envía al usuario cuando pide productos y cuando su pedido está listo.



Otra de las carpetas que encontramos es “Models”, aquí encontramos algunas definiciones de objetos, tales como las tablas que se usarán y que columnas se usarán en algunos casos determinados.

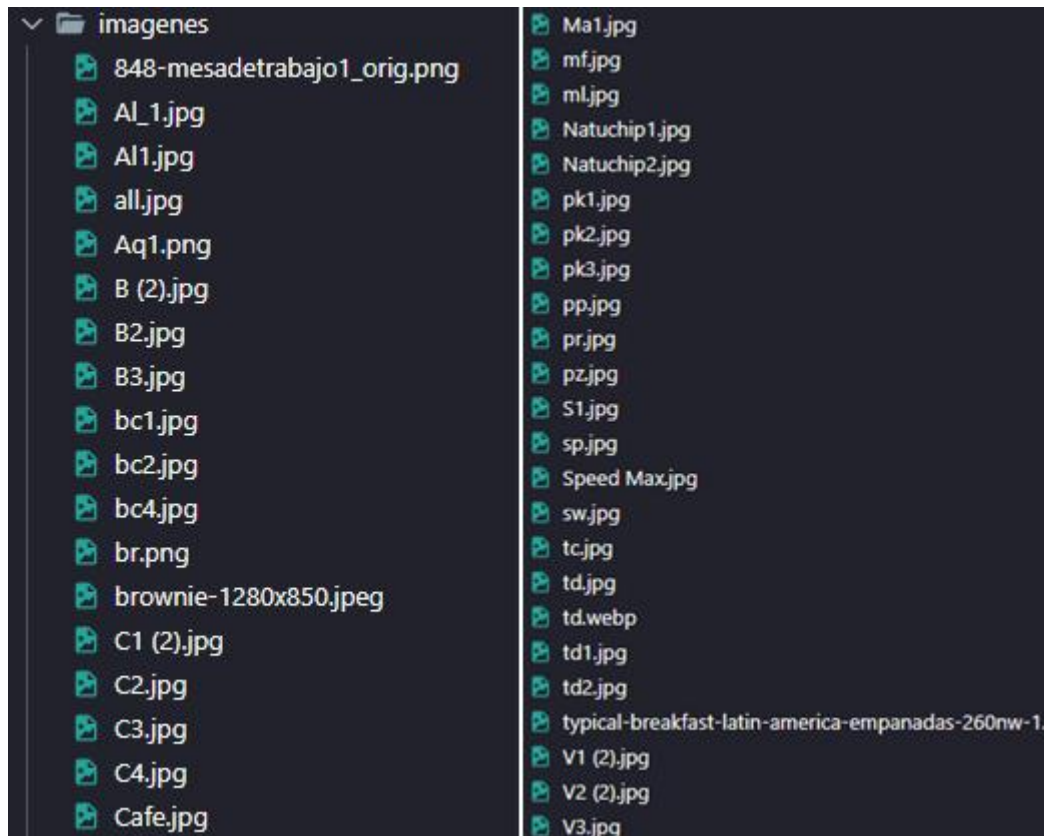


En la carpeta “config” a pesar de encontrar varios archivos, los más importantes son los siguientes:

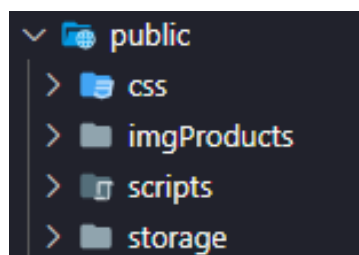


El archivo database es el encargado de la conexión, en este se define qué base de datos se usará, la manera en que se usará y el gestor de bases de datos, dentro del auth se definen algunos usos de la autenticación.

Dentro de la carpeta imágenes se encuentran imágenes de los productos que se muestran en la página principal.

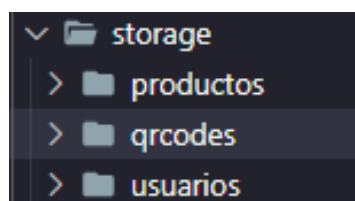


Dentro de la carpeta “public” encontramos otras carpetas tales como:

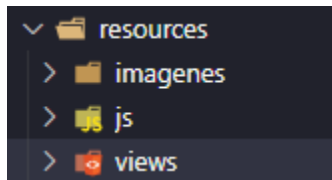


La carpeta de css contiene los archivos CSS usados en el frontend (mostrados anteriormente), dentro de imgProducts encontramos algunas imágenes usadas en el aplicativo, tales como el logo, o las imágenes que se encuentran en las opciones de pago; dentro de scripts están los archivos de JavaScript usados.

Dentro de la carpeta storage de imágenes que se suben en el encuentran implementadas en carpeta usuarios, la cual guarda usuarios.

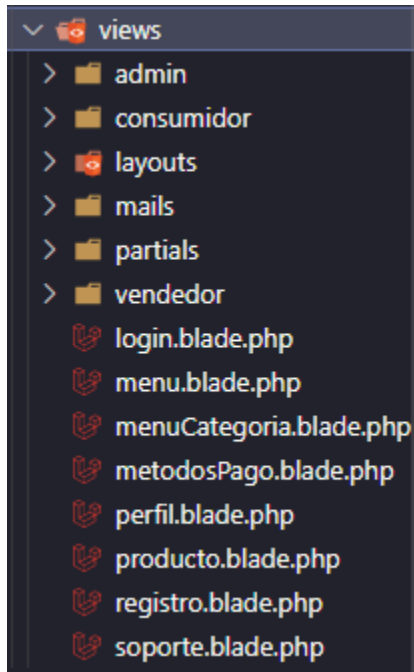


encontramos carpetas aplicativo o que ya se este, aquí vemos la las fotos de perfil de los



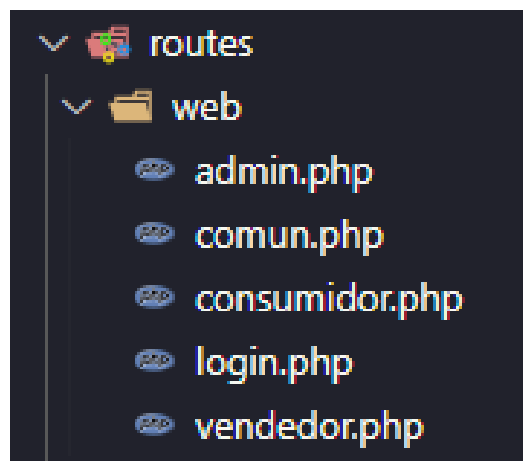
Dentro de la carpeta resources encontramos tres carpetas más, pero en este caso solo nos enfocaremos en una, la cual es views, ya que esta nos muestra las vistas del aplicativo, es decir, el frontend, pero en este caso con algunas “implementaciones”, las cuales hacen parte de backend y le agregan funcionalidad al aplicativo, aquí se define que pueden ver o no los usuarios dependiendo de su rol, o a donde lleva cada botón al ser presionado.

Dentro de la carpeta views se encuentran otras subcarpetas, que están divididas según el rol:



Aquí vemos las carpetas: admin, que contiene la vista de las personas cuyo rol sea administrador, ellos pueden gestionar el soporte y gestionar a los usuarios; otra de las carpetas es consumidor, la cual tiene el carrito de pedidos, los pedidos que genera (el mismo usuario) y el id y código del pedido; por último encontramos la carpeta vendedor, la cual contiene las vistas del vendedor, en la cual puede gestionar los pedidos y los pedidos del día, los cuales genera el consumidor; entre otras carpetas vemos “layouts” (la cual contiene la plantilla principal del aplicativo), la carpeta “partials” que contiene algunas vistas como el header y el footer del aplicativo, los productos o las notificaciones, la carpeta mails contiene los correos que se envían al usuario después de generar su pedido; por ultimo vemos unos archivos tipo “.blade.php”, los cuales son vistas en común de todos los roles existentes, los cuales solo cambian por cosas mínimas.

Luego encontraremos la carpeta “routes”, donde veremos una carpeta llamada web, aquí encontraremos las rutas definidas para cada uno de los roles, y dentro contienen la definición de las rutas dependiendo de la acción que se esté realizando, la acción fue definida con anterioridad en la carpeta Controllers.



Aquí también se define en qué momento usar los middlewares (también definidos con anterioridad).



Modelo entidad – relación:

