

MEMORIA PROYECTO CFGS DESARROLLO DE APLICACIONES WEB

QuickFood



LORENA MOYANO MONTES
CFGS DESARROLLO DE APLICACIONES WEB
Curso 2019-2020

Contenido

1.	Introducción	2
1.1.	Objetivos	2
1.2.	Contexto	2
1.3.	Principales tecnologías y herramientas empleadas	2
2.	Descripción general	3
2.1.	Perspectiva del producto	3
2.2.	Interfaces de usuario.....	3
2.3.	Restricciones de diseño, supuestos y dependencias	4
3.	Implementación de la Base de Datos.....	4
3.1.	Tabla TIPO	6
3.2.	Tabla CLIENTE.....	6
3.3.	Tabla CARTA	7
3.4.	Tabla ALERGENOS	7
3.5.	Tabla ALERGENOS_CARTA.....	8
3.6.	Tabla PEDIDOS.....	8
3.7.	Tabla COMPRA	9
4.	Diseño de la aplicación.....	9
4.1.	Login y registro de usuarios	9
4.2.	Usuario no registrado.....	12
4.3.	Usuario registrado.....	14
4.4.	Usuario repartido	17
4.5.	Usuario administrador	17
5.	API	19
6.	Documentación del código de la aplicación.....	20
7.	Control de versiones	22
8.	Despliegue	22
8.1.	Despliegue en local	22
8.2.	Despliegue servidor.....	23
9.	Conclusiones.....	23
10.	Bibliografía y Webgrafía	24

1. Introducción

Este documento sirve para describir el proyecto realizado para el ciclo formativo de Desarrollo de Aplicaciones Web en el instituto de educación secundaria Campanillas, Málaga.

Dicho proyecto trata sobre una aplicación web que permite pedir comida a domicilio sin salir de casa ni tener que llamar por teléfono. Se ha elegido esta idea por los momentos que vive el país actualmente y la necesidad de los negocios de subsistir.

1.1. Objetivos

El objetivo de la aplicación web elegida es hacer llegar al máximo número de personas posibles las comidas de un restaurante en estos momentos tan difíciles. Ya no se necesita hacer una llamada sin saber si podrán atender el teléfono en ese instante o no, simplemente basta con entrar en la web y hacer pedido haciendo, también, que el restaurante no pierda clientes ya que no está la saturación como ocurre en la vía telefónica.

1.2. Contexto

El proyecto surgió de la idea de la multitud de negocios que han visto sus puertas cerradas a raíz de la pandemia de covid-19 que atraviesa actualmente España. Los restaurantes que tenían reparto a domicilio les ha sido más fácil sobrevivir a los que no lo tenían, de ahí surgió la idea de QuickFood, una aplicación que puede ser utilizada por cualquier restaurante ya que su sencillez hace que sea fácilmente adaptable a uno u otro.

1.3. Principales tecnologías y herramientas empleadas

Las tecnologías empleadas para la realización de este proyecto han sido varias.

En primer lugar, en la parte de back-end se ha utilizado PHP apoyándose en el framework de Laravel 8. Se ha elegido Laravel por su sencillez y su sistema de ficheros, haciendo mucho más fácil las rutas a la hora del despliegue ya que cuenta con un sistema de llamada al controlador y de ahí llamada a la vista. También se ha elegido para aprender a usar algo diferente a lo visto en clase.

En segundo lugar, para la Base de datos se ha empleado MySQL al ser de licencia libre y por ser la más conocida para mí al haberla trabajado durante dos años en clase. Se trata de un gestor de base de datos relacionales muy potente para realizar consultas de una forma muy intuitiva, además de estar integrada con Apache.

En cuanto al front-end se ha empleado JavaScript con JQuery y AJAX. Con JavaScript se han realizado la validación de los formularios y el que un sonido suene cuando ocurra un evento. JQuery ha sido empleado para varias partes en la aplicación, una de ellas ha sido para que el usuario, en el login, pueda ver su contraseña pulsando sobre el icono del ojo que le aparece en el campo del mismo nombre. En AJAX se ha realizado tanto el buscador como la paginación de las páginas de la carta y del panel de administración.

Además, se han empleado lenguajes de marcas como HTML5 para las vistas que ve el usuario apoyándose en hojas de estilo CSS3 realizadas mediante el procesador SASS.

Para el control de versiones se ha empleado GitHub. Se ha elegido ya que tiene muy buena integración con Visual Studio Code y resulta realmente sencillo subir los ficheros.

Por último, para el despliegue de la aplicación en remoto se ha utilizado lucushost. Lucushost es un hosting de origen español que tiene un periodo de prueba gratuito de 15 días con todas las funcionalidades.

2. Descripción general

2.1. Perspectiva del producto

La aplicación desarrollada pretende ser un medio por el que distintas personas puedan acceder a la carta de un restaurante y poder elegir qué comer sin necesidad de salir de su vivienda para ir a adquirirlo o tener que realizar una llamada. También está enfocada a personas con discapacidad auditiva a las que les resulta imposible realizar una llamada telefónica para escuchar a la otra persona.

2.2. Interfaces de usuario

En la aplicación existen 4 tipos de perfil de usuario, uno de ellos es el usuario no registrado o invitado. Cada uno tiene un tipo diferente de funcionalidades, siendo el usuario no registrado y el repartidor los que menos tienen, por el contrario, el usuario de administrador es quien tiene el mayor número de funcionalidades dentro de la aplicación. A continuación se describen brevemente lo que cada usuario puede hacer dentro de QuickFood.

- **Usuario no registrado:**
 - Loguearse en la aplicación. Podrá acceder a ella mediante un correo y contraseña con el que previamente se haya registrado.
 - Registro. El usuario tendrá que introducir sus datos para poder entrar en la aplicación a través de un formulario. Sólo se hace una vez.
 - Visionar la página de inicio donde hay una serie de información del restaurante.
 - Visionar la carta y utilizar el buscador para buscar algún producto pudiendo ver los datos del mismo y la tabla de alérgenos.
 - Visionar la página de contacto. En ella se encuentra una galería de imágenes, vídeo y la localización del restaurante tanto físicamente, telefónicamente y vía redes sociales.
- **Usuario registrado:**
 - Accede a la aplicación previo logueo en la misma.
 - Puede cerrar sesión desde el desplegable al pulsar sobre “perfil”
 - Puede comprar productos desde la vista de la carta pulsando sobre el botón de ver y añadiéndolos al carrito junto con la cantidad deseada.
 - Eliminar un producto que ha sido añadido al carrito previamente.
 - Visionado de su perfil, editarlo y eliminarlo.
 - Búsqueda de pedidos en una fecha elegida por el usuario en el datepicker que hay en su perfil.
 - Visionado de sus pedidos una vez elegida la fecha como se ha descrito anteriormente.
 - Poder pagar un pedido mediante tarjeta.
 - Ver si el pedido ha sido repartido o aún no.
- **Usuario repartidor:**
 - Accede a la aplicación mediante el logueo.
 - Puede ver la carta, pero sin poder comprar nada, sólo de modo informativo para los clientes que necesiten alguna aclaración.

MEMORIA PROYECTO CFGS DESARROLLO DE APLICACIONES WEB

- Tiene una vista dedicada para los repartos que aún no han sido entregados.
- **Usuario administrador:**
 - Acceso mediante formulario de logueo.
 - Puede cerrar su sesión.
 - Puede añadir, editar o borrar un producto de la carta.
 - Vista de todos los usuarios registrados en la aplicación con una búsqueda que filtra por el nombre del usuario.
 - Modificación del tipo de perfil del usuario.
 - Borrado de un usuario concreto.
 - Visión de un gráfico del número de compras que ha realizado cada usuario.

2.3. Restricciones de diseño, supuestos y dependencias

La aplicación requiere de una conexión a internet de cualquier tipo (fibra, ADSL, datos móviles...) además de un dispositivo que se encuentre conectado a la red ya sea un ordenador, tablet, móvil, televisor inteligente o cualquier aparato que disponga de internet.

Además de lo anterior, el usuario también necesita un navegador web para el visionado de la aplicación. El navegador podrá ser cualquiera que se encuentre actualizado a la última versión a ser posible, ya que algunos componentes podría no cargarlos bien.

3. Implementación de la Base de Datos

Para la creación de la base de datos de la que se alimenta la aplicación se ha optado por utilizar un modelo relacional.

El esquema de la base de datos sería el apreciado en la siguiente imagen.

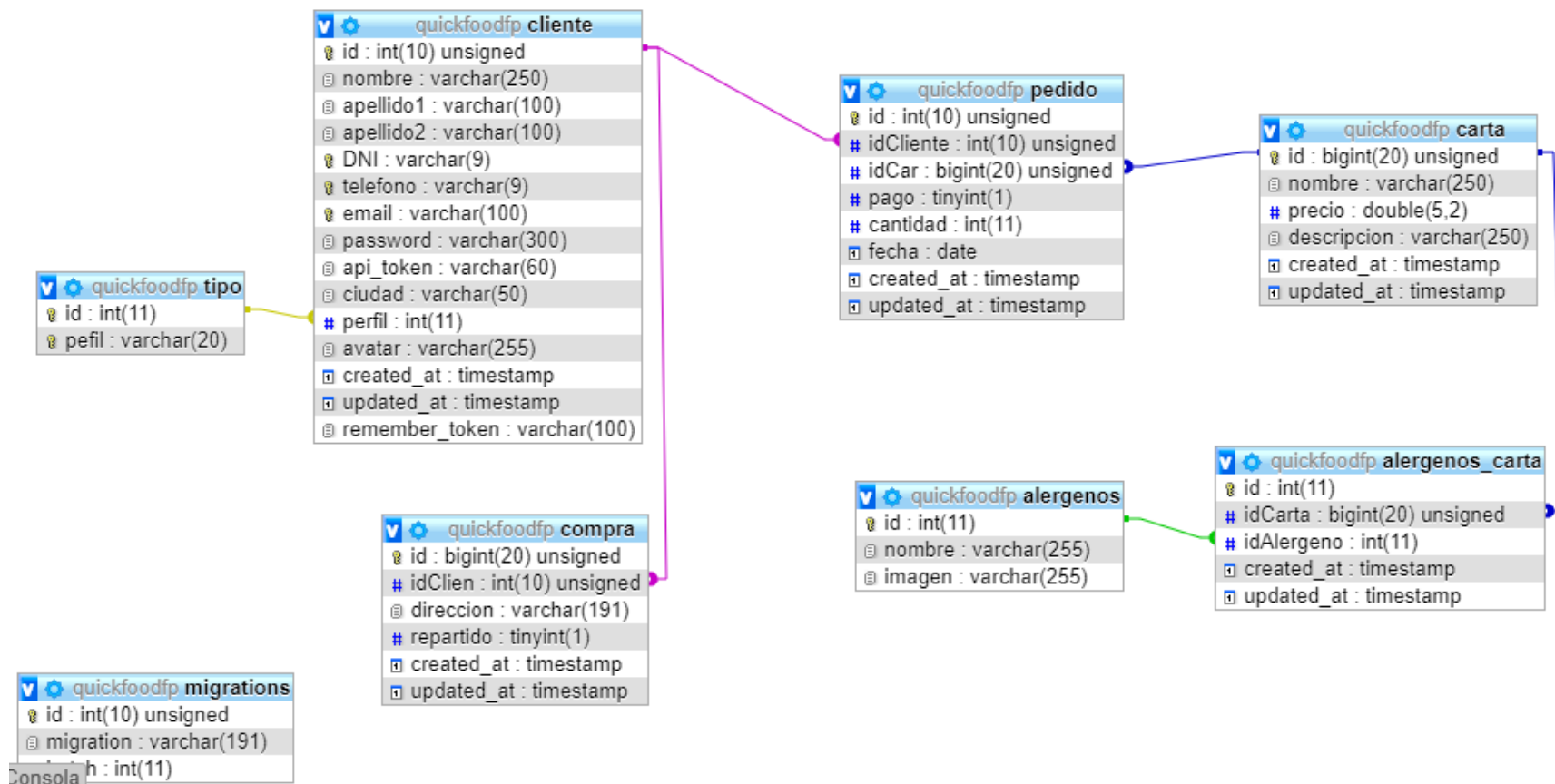


Ilustración 1 Base de datos

La base de datos cuenta con siete tablas en total. A continuación, una breve explicación de cada una de ellas.

3.1. Tabla TIPO

En primer lugar, se encuentra la tabla para los tipos de usuario. Contiene los tres tipos de perfiles con los que cuenta la aplicación que son administrador, repartidor y usuario tal y como se puede apreciar en la siguiente imagen.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
<input type="checkbox"/> 1	id 🔑	int(10)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
<input type="checkbox"/> 2	perfil	varchar(191)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 3	created_at	timestamp			Sí	NULL		
<input type="checkbox"/> 4	updated_at	timestamp			Sí	NULL		

Ilustración 2 Tabla tipo

3.2. Tabla CLIENTE

Seguidamente encontramos la tabla de usuarios/clientes. En ella se almacenan los usuarios que se han registrado en la aplicación junto con los datos con los que se registran, además, cuenta con un el campo “perfil” como clave foránea. Ahí se especifica el tipo de usuario que es. Por defecto, cuando un usuario se registra, se le asigna el tipo usuario (2).

Además de ese campo, podemos apreciar que tanto el DNI, el teléfono y el email son campos únicos, por lo que, si el usuario intenta registrarse utilizando alguno de ellos, Laravel le marcará un error de que ya existe no realizando la inserción del usuario y devolviéndole a la pantalla del registro para que corrija sus datos.

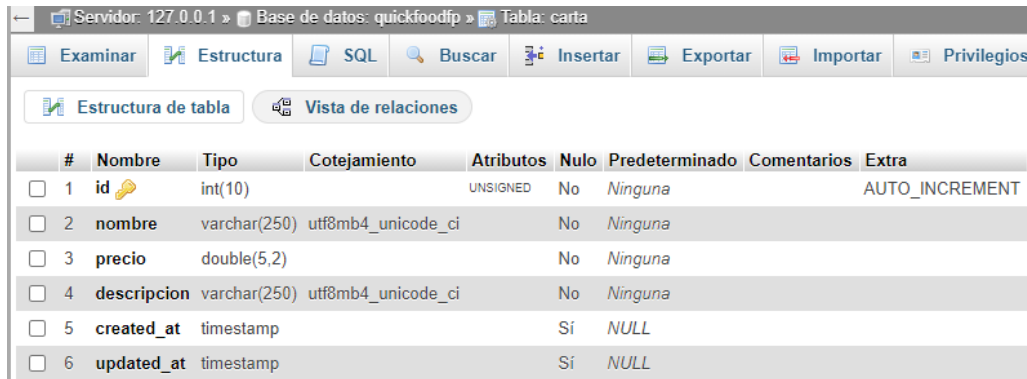
Servidor: 127.0.0.1 » Base de datos: quickfoodfp » Tabla: cliente								
<div> <div>Examinar</div> <div>Estructura</div> <div>SQL</div> <div>Buscar</div> <div>Insertar</div> <div>Exportar</div> <div>Importar</div> <div>Privilegios</div> </div>								
<div> <div>Estructura de tabla</div> <div>Vista de relaciones</div> </div>								
#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
<input type="checkbox"/> 1	id 🔑	int(10)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
<input type="checkbox"/> 2	nombre	varchar(250)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 3	apellido1	varchar(100)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 4	apellido2	varchar(100)	utf8mb4_unicode_ci		Sí	NULL		
<input type="checkbox"/> 5	DNI 🔑	varchar(9)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 6	telefono 🔑	varchar(9)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 7	email 🔑	varchar(100)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 8	password	varchar(300)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 9	ciudad	varchar(50)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 10	perfil 🔑	int(10)		UNSIGNED	No	Ninguna		
<input type="checkbox"/> 11	avatar	varchar(255)	utf8mb4_unicode_ci		Sí	NULL		
<input type="checkbox"/> 12	created_at	timestamp			Sí	NULL		
<input type="checkbox"/> 13	updated_at	timestamp			Sí	NULL		
<input type="checkbox"/> 14	remember_token	varchar(100)	utf8mb4_unicode_ci		Sí	NULL		

Ilustración 3 Tabla cliente

3.3. Tabla CARTA

A continuación, tenemos la tabla de carta que es donde se encuentran todos los productos del catálogo. El administrador es la única persona que puede añadir nuevos productos a la misma.

En esta tabla podemos apreciar que los datos que se guardan son el nombre del producto, el precio que aceptaría cinco caracteres, dos de ellos decimales, y una breve descripción.



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(10)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
2	nombre	varchar(250)	utf8mb4_unicode_ci		No	Ninguna		
3	precio	double(5,2)			No	Ninguna		
4	descripcion	varchar(250)	utf8mb4_unicode_ci		No	Ninguna		
5	created_at	timestamp			Sí	NULL		
6	updated_at	timestamp			Sí	NULL		

Ilustración 4 Tabla carta

3.4. Tabla ALERGENOS

Otra tabla sería la de alérgenos ya que a día de hoy se hace imprescindible conocer qué contienen los alimentos para las personas con algún tipo de alergia. Así pues, la tabla tendría la siguiente estructura.



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(10)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
2	nombre	varchar(191)	utf8mb4_unicode_ci		No	Ninguna		
3	imagen	varchar(191)	utf8mb4_unicode_ci		No	Ninguna		
4	created_at	timestamp			Sí	NULL		
5	updated_at	timestamp			Sí	NULL		

Ilustración 5 Tabla alérgeno

3.5. Tabla ALERGENOS_CARTA

A su vez, hay una tabla de alérgenos_carta ya que un producto puede tener más de un tipo de alérgeno, por lo que se ha hecho necesario la creación de una tabla intermedia. En ella encontramos tanto el id de la carta como del alérgeno para poder relacionarlos.



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(10)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
2	idCarta	int(10)		UNSIGNED	No	Ninguna		
3	idAlergeno	int(10)		UNSIGNED	No	Ninguna		
4	created_at	timestamp			Sí	NULL		
5	updated_at	timestamp			Sí	NULL		

Ilustración 6 Tabla alérgeno_carta

3.6. Tabla PEDIDOS

En cuanto a los pedidos del usuario, encontramos la tabla de pedidos donde se guarda el id del cliente además del id del producto que ha pedido y si se encuentra pagado o no por medio de un campo tinyint que nos guarda un 0 o un 1 dependiendo del estado del pedido. Por defecto, un pedido recién realizado y hasta que el usuario termina de pagarlo, el campo se encuentra en 0. Una vez se realiza el pago, automáticamente se actualiza con un 1.

Esta tabla se emplea también para saber el historial de pedidos del usuario, para el administrador conocer cuántas compras ha hecho el usuario... Dicha tabla cuenta con la siguiente estructura.



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(10)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
2	idCliente	int(10)		UNSIGNED	No	Ninguna		
3	idCar	int(10)		UNSIGNED	No	Ninguna		
4	pago	tinyint(1)			No	Ninguna		
5	cantidad	int(11)			No	Ninguna		
6	fecha	date			No	Ninguna		
7	created_at	timestamp			Sí	NULL		
8	updated_at	timestamp			Sí	NULL		

Ilustración 7 Tabla pedido

3.7. Tabla COMPRA

Por último, tenemos la tabla de compra. En ella se almacena el id del cliente y es la utilizada por el repartidor y saber si el pedido se encuentra en proceso de entrega (0) o entregado (1). Por defecto, cuando el usuario hace un pago, el campo “repartido” se guarda con un 0 como valor por defecto. Una vez entregado el pedido, se actualiza el campo poniéndolo a 1.

Esta tabla también cuenta con la dirección del cliente ya que no se le pide durante el formulario de registro.



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
<input type="checkbox"/> 1	id	int(10)		UNSIGNED	No	Ninguna		AUTO_INCREMENT
<input type="checkbox"/> 2	idClien	int(10)		UNSIGNED	No	Ninguna		
<input type="checkbox"/> 3	direccion	varchar(191)	utf8mb4_unicode_ci		No	Ninguna		
<input type="checkbox"/> 4	repartido	tinyint(1)			No	Ninguna		
<input type="checkbox"/> 5	created_at	timestamp			Sí	NULL		
<input type="checkbox"/> 6	updated_at	timestamp			Sí	NULL		

Ilustración 8 Tabla compra

Estas tablas también se emplean por medio de join para poder hacer realizar algunas funcionalidades más dentro de la aplicación.

4. Diseño de la aplicación

El diseño elegido ha sido MVC (Modelo-Vista-Controlador). Para su realización se ha empleado el framework de PHP, Laravel 8 junto con lenguajes como JavaScript para la realización de algunos pequeños detalles de la aplicación.

Como resultado se han obtenido una serie de vistas que se explican a continuación.

4.1. Login y registro de usuarios

Un usuario puede acceder a la aplicación por dos vías. La primera es por medio del logueo. Si el usuario se ha registrado previamente, podrá usar esta opción introduciendo en los campos correspondientes su usuario (email) y su contraseña. La vista cuenta con una función para ver la contraseña con texto en plano. Dicha función ha sido realizada en JQuery.

La otra vía que tiene el usuario para entrar es accediendo mediante el registro. Si el usuario no se ha registrado previamente, deberá usar esta función. Necesitará introducir una serie de datos como son su nombre, apellidos, número de teléfono, ciudad, etc.

Una vez le da al botón de guardar, la aplicación le redirige automáticamente al home entrando sin tener que introducir datos de logueo.

Ambas vistas son las siguientes.

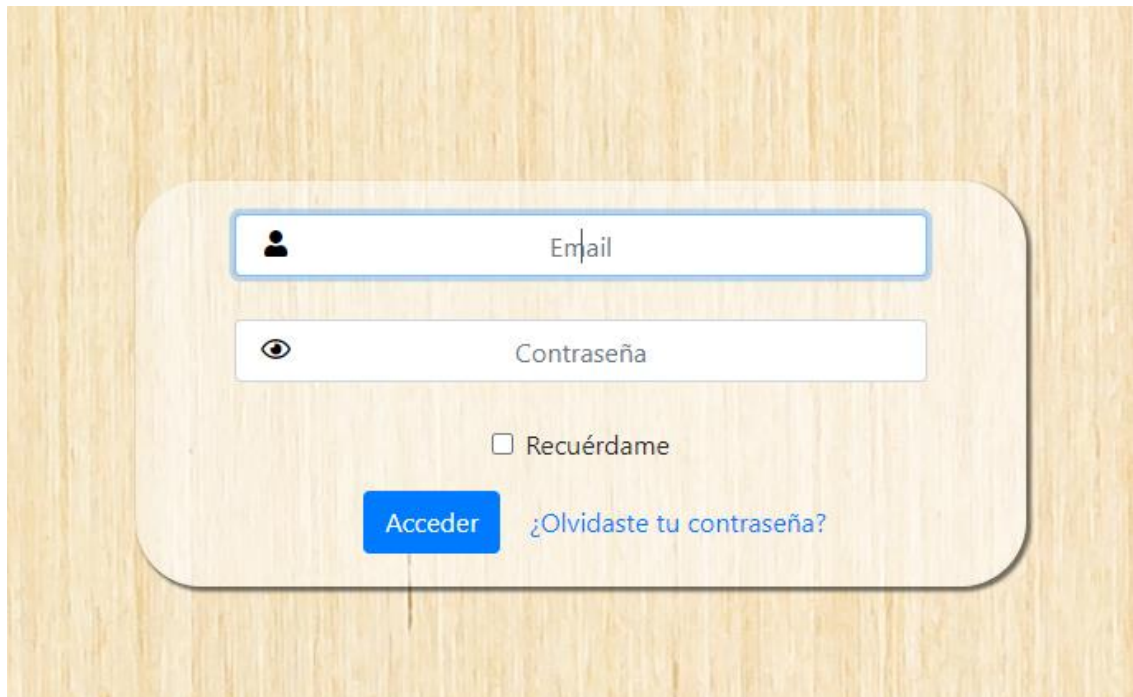
A login form on a light wood background. It features two input fields: the first is labeled 'Email' with a person icon, and the second is labeled 'Contraseña' with an eye icon. Below the password field is a checkbox labeled 'Recuérdame'. At the bottom left is a blue button labeled 'Acceder', and at the bottom right is a link labeled '¿Olvidaste tu contraseña?'.

Ilustración 9 Login

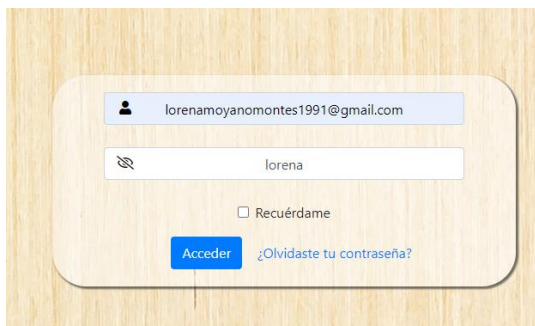
The login form with the email field filled with 'lorenamoyanomontes1991@gmail.com' and the password field filled with 'lorena'. The 'Recuérdame' checkbox is unchecked. The 'Acceder' button and the '¿Olvidaste tu contraseña?' link are at the bottom.

Ilustración 11 Mostrando la contraseña

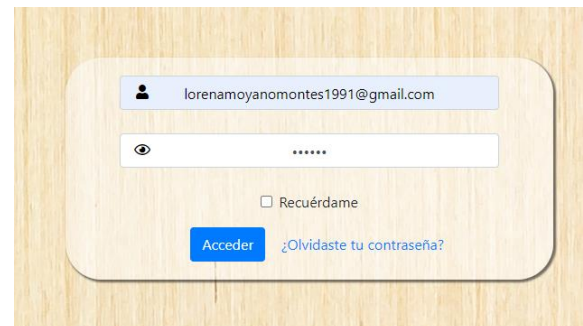
The login form with the email field filled with 'lorenamoyanomontes1991@gmail.com' and the password field filled with seven asterisks. The 'Recuérdame' checkbox is unchecked. The 'Acceder' button and the '¿Olvidaste tu contraseña?' link are at the bottom.

Ilustración 10 Contraseña oculta

Para la realización de la muestra u ocultación de la contraseña ha empleado JQuery de la siguiente manera

MEMORIA PROYECTO CFGS DESARROLLO DE APLICACIONES WEB

```

<script>
$(document).ready(function() {
    $('#checkbox').on('change', function() {
        $('#password').attr('type', $('#checkbox').prop('checked') == true ? "text" : "password");
    });
});
</script>
<script>
const togglePassword = document.querySelector('#togglePassword');
const password = document.querySelector('#password');
togglePassword.addEventListener('click', function(e) {
    const type = password.getAttribute('type') === 'password' ? 'text' : 'password';
    password.setAttribute('type', type);
    this.classList.toggle('fa-eye-slash');
});
</script>

```

Ilustración 12 Contraseña oculta / muestra

El primer código nos sirve para que el ojo se vea de forma normal o con un tachado sobre él.

El segundo código nos coge el id de la contraseña y hace que ese input sea visible en forma de texto o bien, se oculte como una contraseña.

En cuanto al formulario de registro, que no incorpora esta opción, se vería de la siguiente forma.

Ilustración 13 Formulario de registro

4.2. Usuario no registrado

Aun así, el usuario no tiene obligación de autenticarse en la aplicación si solo quiere mirar la carta o información sobre el sitio. En caso de que se produzca este caso, puede acceder a tres vistas más siendo la primera la de Inicio. En esta vista tenemos un banner con un poco de la historia del restaurante justo debajo.



Ilustración 14 Página principal de la aplicación

Desde esta vista se puede acceder directamente a la página de “nosotros” donde se encuentran los datos de contacto, una pequeña galería de imágenes y un vídeo promocional del restaurante.

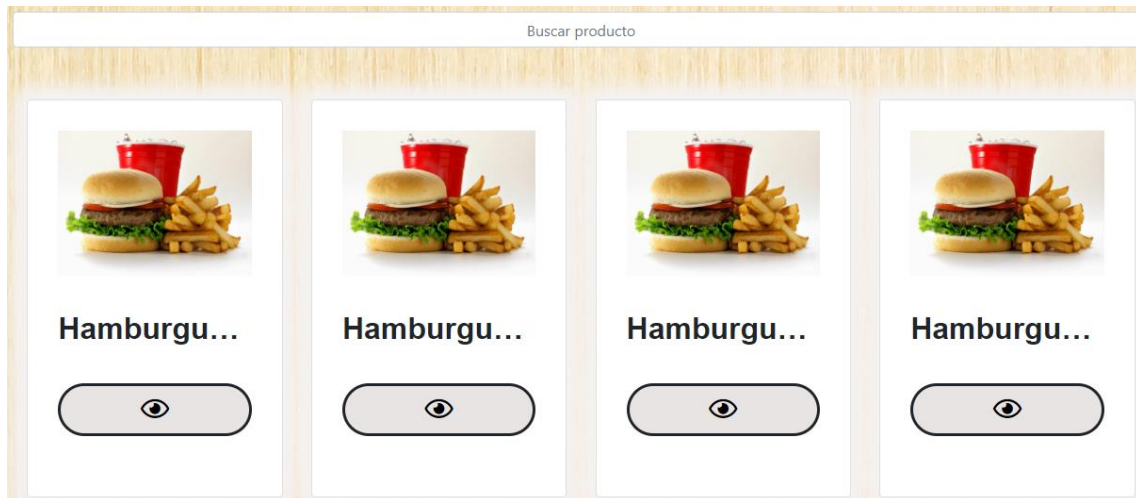
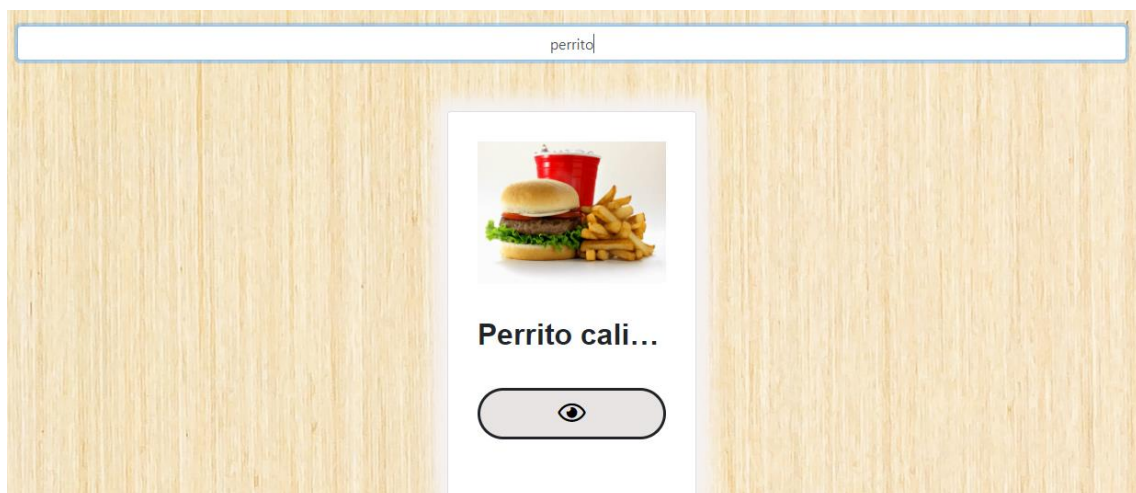
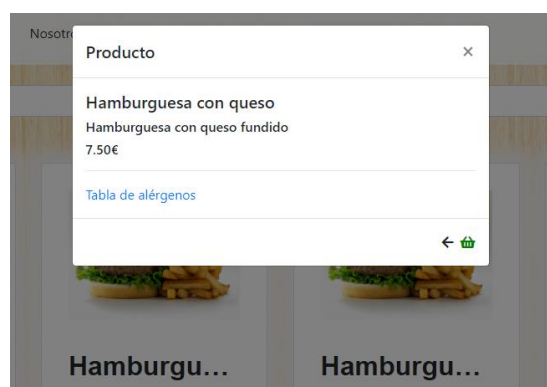


Ilustración 15 Galería y página de contacto

Lo último que podrá realizar el usuario no registrado será ver los productos de la carta junto con sus alérgenos, pero no podrá realizar ninguna compra. Si lo intenta, se le redirige al login automáticamente.

MEMORIA PROYECTO CFGS DESARROLLO DE APLICACIONES WEB

Además, la página cuenta con un buscador de productos que hace un filtrado en tiempo real de los productos de los que se dispone.

*Ilustración 16 Carta**Ilustración 17 Buscando un producto**Ilustración 18 Información del producto*

4.3. Usuario registrado

Si el usuario desea registrarse o iniciar sesión, se le volverá a llevar a la pantalla de Inicio. Una vez ha hecho esto, podrá comprar cualquier producto que se encuentre disponible. El funcionamiento es igual que el del usuario no registrado, salvo la opción de poder añadir al carrito de la compra que anteriormente no era posible.

Esta opción muestra un campo para poder elegir la cantidad deseada. Si posteriormente queremos añadir más cantidades, basta con volver al artículo y seleccionar otra cantidad, se añadirá automáticamente al carrito sumándose a las que había anteriormente.

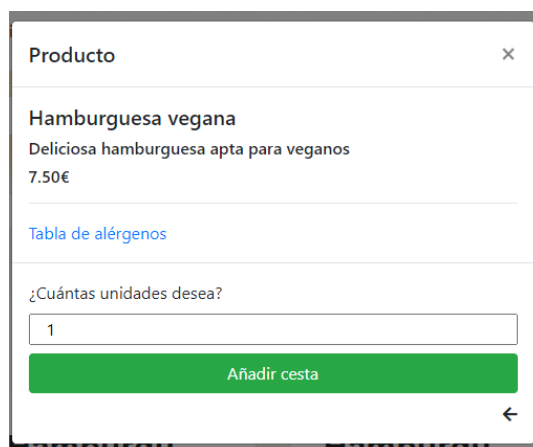


Ilustración 19 Usuario logueado puede comprar

Una vez añadido el pedido al carrito, nos vuelve a llevar a la ventana de productos y ya podemos seguir accediendo a las distintas partes de la aplicación.

La siguiente vista es la del carrito. En ella, el usuario puede borrar los productos que ha añadido o sencillamente pagarlos pulsando sobre el botón. Cuando el usuario pulsa el botón se abre una ventana modal donde debe introducir los datos solicitados. En caso de que no lo haga, la aplicación le mostrará errores tanto si se deja algún campo vacío como si introduce algo erróneo.

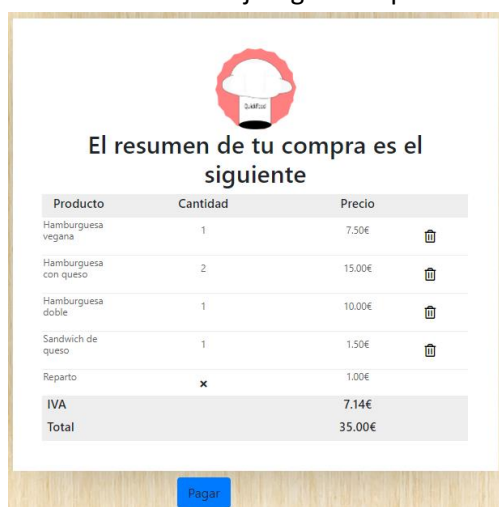


Ilustración 20 Carrito

MEMORIA PROYECTO CFGS DESARROLLO DE APLICACIONES WEB

Nombre	Teléfono
<input type="text" value="Rosa Soliz"/>	<input type="text" value="780908385"/>
Dirección	
<input type="text" value="Dirección"/>	
Número de la tarjeta	
<input type="text" value="Número tarjeta"/>	
Fecha de caducidad	
Mes	Año
<input type="text" value="Mes"/>	<input type="text" value="Año"/>
CVV	
<input type="text" value="CVV"/>	

Ilustración 21 Formulario de pago

Una vez pagado, podrá ir a su perfil. Desde allí puede acceder a sus datos, la contraseña se ha omitido por seguridad, y también podrá usar el buscador de pedidos. En caso de que el pedido no se encuentre entregado le aparecerá, pero con un aviso de que aún no ha sido entregado. En caso contrario, le dirá el número de pedidos realizados ese día además del total en una misma cuenta conjunta para que tenga un informe completo de lo gastado ese día. Se ha decidido hacer así porque no es típico pedir comida en un restaurante varias veces al día.

Perfil de Rosa Soliz



ALTRAMUCES

Nombre completo: Rosa Soliz García Montañez

Teléfono: 780908385

Email: amparo.casanova@gmail.com

Localidad: O Delrío

Tiempo en QuickFood: 1 segundo antes






Ilustración 22 Perfil del usuario

MEMORIA PROYECTO CFGS DESARROLLO DE APLICACIONES WEB

Para mostrar al usuario el pedido que realizó el día que él selecciona en su perfil, se ha realizado de esta manera en el controlador.

```
public function show_pedido(Request $request) {
    $fecha = $request->fecha;
    $id = Auth::user()->id;
    $show_pedido = DB::table('pedido')->join('carta', 'carta.id', '=', 'pedido.idCar')
        ->select('pedido.fecha', 'carta.precio', 'carta.nombre', 'pedido.cantidad')
        ->where('idCliente', '=', $id)
        ->where('pago', '=', '1')
        ->where('pedido.fecha', '=', $fecha)
        ->get();

    $precio_reparto = DB::table('comprar')->where('idClien', '=', $id)
        ->where('repartido', '=', '1')
        ->whereDate('created_at', '=', $fecha)
        ->count();

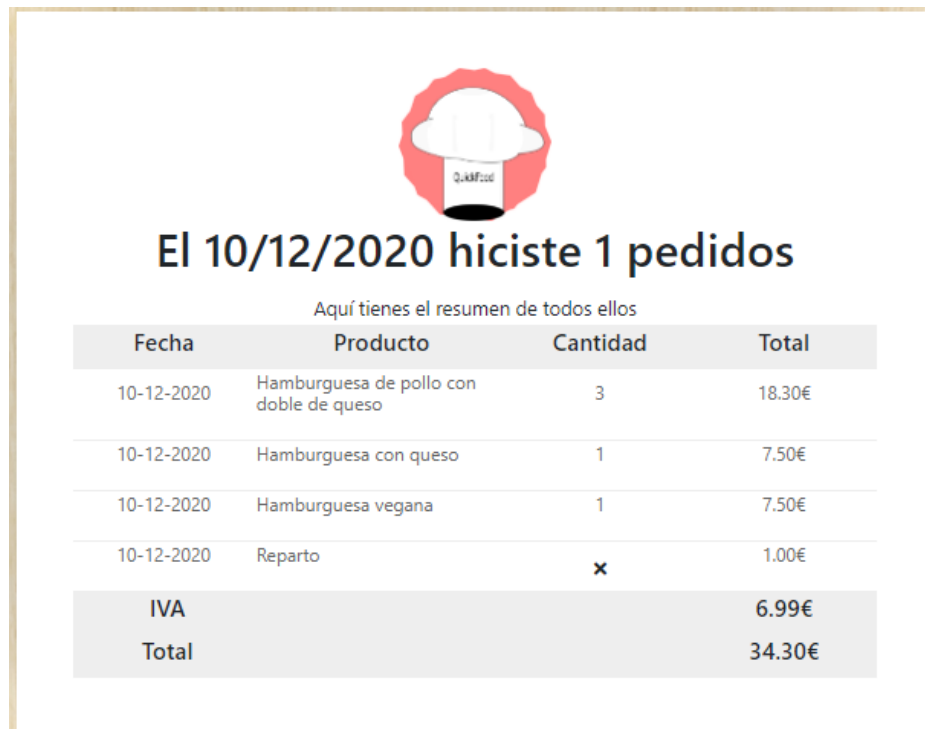
    return view('users.show_pedido', ['show_pedido' => $show_pedido, 'fecha' => $fecha,
        'precio_reparto' => $precio_reparto]);
}
```

Ilustración 23 Mostrar el pedido del usuario

En primer lugar, obtenemos todos los pedidos realizados en dicha fecha con un join entre la tabla de pedidos y carta para que nos muestre también el nombre y el precio del producto.

Seguidamente se hace otra consulta para saber el precio del reparto de dicho pedido por si el usuario ha hecho más de un pedido a lo largo del día.

Finalmente se devuelven ambos datos a la vista quedando de la siguiente manera todo.



Fecha	Producto	Cantidad	Total
10-12-2020	Hamburguesa de pollo con doble de queso	3	18.30€
10-12-2020	Hamburguesa con queso	1	7.50€
10-12-2020	Hamburguesa vegana	1	7.50€
10-12-2020	Reparto	x	1.00€
IVA			6.99€
Total			34.30€

Ilustración 24 Historial de pedidos

Por último, para el usuario, si desea salir de la aplicación, cuenta con un botón de cerrar sesión en el desplegable que se encuentra en su perfil.

4.4. Usuario repartido

El usuario repartidor es el que cuenta, junto con el no registrado, con menos funcionalidades en la aplicación ya que sus funciones se limitan a seleccionar si un pedido ha sido o no entregado. Para ello se vale de una tabla con los datos del usuario como son su nombre, dirección, teléfono y estado del pedido.

También se le ha dejado la vista de los productos por si el usuario, en el momento de la entrega, tuviera alguna duda sobre precio, ingredientes... el repartidor podría asesorarle en lo que necesite viendo la página de los productos. No obstante, no podrá realizar ningún pedido bajo la cuenta de repartidor.

Nombre cliente	Dirección	Teléfono	Status pedido
Rosa Soliz	Coín	780908385	No entregado

Ilustración 25 Tabla del reparto

4.5. Usuario administrador

El usuario administrador tiene un total control de la aplicación. Puede realizar cualquier tarea con la salvedad de que, al igual que el repartidor, no puede realizar ningún pedido bajo su cuenta.

El administrador puede añadir, eliminar o editar un producto. Eliminar usuarios, ver la tabla de todos los usuarios que se encuentran registrados en la aplicación y, además, podrá ver una gráfica con el número total de pedidos realizados por los usuarios. Para la realización de dicha gráfica se ha empleado un canvas y charjs.

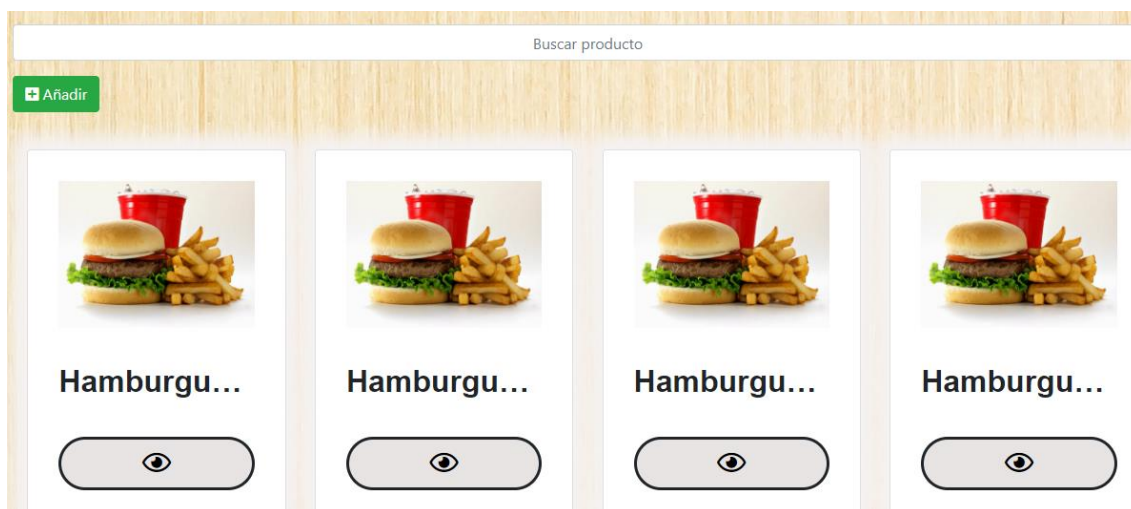


Ilustración 26 Botón añadir del administrador

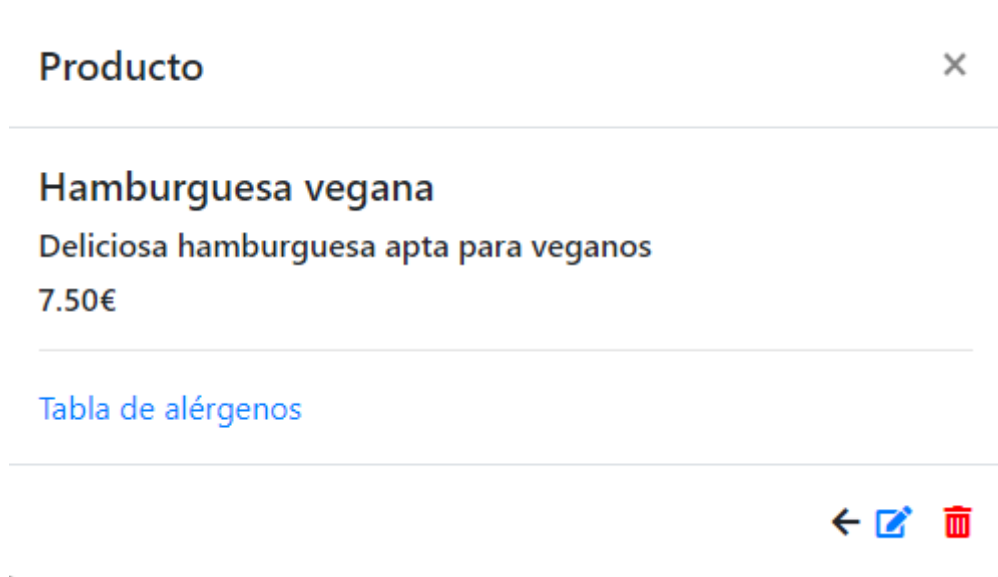


Ilustración 27 Editar y borrar producto

Buscar usuario									
Nombre	Apellidos	DNI	Teléfono	Email	Ciudad	Perfil	Unido	Borrar	
Lorena	Moyano Montes	34587521B	666111220	lorenamoyanomontes1991@gmail.com	Málaga	Administrar	1 segundo antes		
D. Rubén Sáenz	Lorente Luján	68638041h	799574781	ivelazquez@yahoo.es	L' Ayalaa	Repartidor	1 segundo antes		
Rosa Soliz	García Montañez	72100783k	780908385	amparo.casanova@gmail.com	O Delrío	Cliente	1 segundo antes		
Samuel Montez	Sevilla Rentería	29397821v	637282357	baez.ainara@terra.com	Los Roca	Cliente	1 segundo antes		
Olivia Pons	De Anda Acuña	81097861a	790378984	jaime.gallardo@terra.com	Barraza de las Torres	Cliente	1 segundo antes		
Martina Romo	Miguel Gracia	51074984m	673446387	olivia.santana@yahoo.com	El Barrientos	Cliente	1 segundo antes		
Ing. Bruno Urías	Lebrón Saucedo	81995508f	778546675	gluque@gmail.com	Cisneros de la Sierra	Cliente	1 segundo antes		
Sr. Mario Pedraza Segundo	Mora Lovato	29068952m	748258198	unai.morales@hotmail.es	Collado del Mirador	Cliente	1 segundo antes		

Ilustración 28 Panel de búsqueda de usuarios



Ilustración 29 ChartJS con el número de pedidos de los usuarios

5. API

Para la generación de la API se ha empleado un controlador con el mismo nombre y las rutas se han guardado dentro del archivo api.php que se encuentra dentro de la carpeta de routes/api.

Una vez tenemos creados los métodos que vamos a usar en nuestra API, pasamos a hacer las llamadas con PostMan. PostMan es una aplicación que se puede descargar en internet y que sirve para hacer llamadas a la API como pruebas. Su funcionamiento es muy simple, basta con poner la URL de la API en el campo correspondiente, elegir el método que vamos a utilizar, en mi caso ha sido GET y pulsar sobre el botón de send/enviar. Una vez hecho esto, nos genera un código JSON en la pantalla de la aplicación. Para guardarlo, sencillamente pulsamos sobre la consulta en el historial y pulsamos sobre guardar.

Cuando hayamos generado todo lo que necesitemos, nos dirigimos hacia Collections. Previamente habremos guardado todas las consultas realizadas en una carpeta. Pulsamos sobre esa carpeta y, bien podemos exportar el archivo en un JSON y guardarlo donde prefiramos o, también, podemos compartir el enlace que nos carga en el navegador las llamadas que hemos realizado con un enlace en cada una de ellas que nos lleva hasta la consulta.

[Enlace a la API](#)

```
public function productos(){
    $carta = DB::table('carta')->get();
    return response()->json($carta);
}

/**
 * View a product
 *
 * @param id
 * @return view json product
 */

public function producto($id){

    $carta = DB::table('carta')->where('id', $id)->get();
    return response()->json($carta);
}

/**
 * View all users in the database
 *
 * @param
 * @return view cliente json
 */

public function usuarios(){
    $cliente = DB::table('cliente')->get();
    return response()->json($cliente);
}
```

Ilustración 30 Controlador de la API con algunos métodos

6. Documentación del código de la aplicación

Para la documentación del código se ha empleado PHPDocumentor para Laravel y JSDoc para el código de JavaScript.

PHPDocumentor es un software que genera automáticamente un index.html con todos nuestros comentarios creados previamente en la aplicación.

Para su funcionamiento basta con descargar la versión de PHPDocumentor desde GitHub y guardarlo dentro de la carpeta de nuestro proyecto. Una vez hecho esto, escribimos el siguiente comando en nuestra consola de Visual Studio Code para que nos genere a la documentación, ésta tardará algo de tiempo dependiendo del número de archivos que queremos generar.

```
php phpDocumentor.phar
```

Una vez generado todo el código podemos acceder a él mediante la carpeta que nos ha creado dentro de nuestro proyecto. Por defecto la carpeta se llama .phpDoc. Dentro de ella encontramos dos carpetas, build y cache. Para conseguir el código sólo tenemos que entrar dentro de la carpeta de build y abrir el fichero index.html que nos ha generado.

Cuando estemos en el navegador, ya podemos acceder a los puntos que queramos de nuestra documentación. Para ello nos movemos con el menú de navegación accediendo, por ejemplo, a uno de los controladores creados por nosotros mismos.

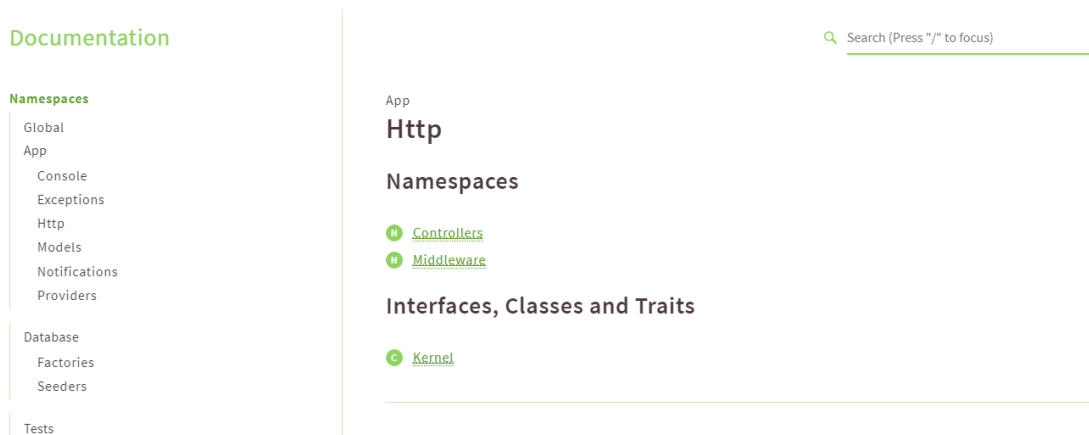


Ilustración 31 PHPDoc

App \ Http

Controllers

Namespaces

N [Auth](#)

Interfaces, Classes and Traits

C [AdminController](#)**C** [ApiController](#)**C** [CartaController](#)**C** [ClienteController](#)**C** [Controller](#)**C** [HomeController](#)**C** [PaginationController](#)**C** [PedidoController](#)*Ilustración 33 Controladores*

productos()

View all the products

```
public productos() : view
```

Return values

[view](#) —
json products

usuario()

View an user in the database

```
public usuario(mixed $id) : view
```

Parameters

\$id : mixed

Return values

[view](#) —

Ilustración 32 Comentario del controlador API

En cuanto a la generación de la documentación de JavaScript se ha optado por utilizar JSDoc por su fácil manejo y que sólo es necesario tener instalado Node en nuestra máquina.

Una vez generados los comentarios del código, hacemos el siguiente comando dentro de la carpeta donde tenemos nuestro código javascript.

```
C:\xampp\htdocs\quickfood\quickfood\public\js>jsdoc validar.js
```

Ilustración 34 Generar documentación de JavaScript

Cuando lo genera, nos muestra una página web y podemos navegar por los distintos métodos que hemos comentado.

quitarError(msg)
change the visibility of the message for the user

Parameters:

Name	Type	Description
msg	*	

Source: [validar.js, line 60](#)

sonido()
play a sound when the user click the pay button

Source: [validar.js, line 69](#)

validar_pago()
validete pay's form and send to the user a message if he write other type of date

Source: [validar.js, line 5](#)

validar_registro()
validate the register form

Ilustración 35 Documentación JavaScript

7. Control de versiones

Para el control de versiones se ha empleado GitHub por su fácil integración con Visual Studio Code. A través de su extensión se hace muy sencillo subir un código después de acabarlo.

Su uso se basa en seleccionar los archivos que queremos subir a GitHub y ponerles, o no, un commit. Tras eso dar al botón de + para subirlo.

Con el control de versiones es muy fácil volver a una versión anterior del código si algo que hayamos implementado nuevo nos falla en algún momento.

Se ha realizado todo el despliegue en la rama del master porque no se ha visto necesario emplear otro tipo de ramas para el desarrollo.

8. Despliegue

El despliegue de la aplicación se ha realizado en varios lugares.

En primer lugar, se ha desplegado en local vía xampp que cuenta con una versión de Apache.

En segundo lugar, se han empleado dos hostings para su alojamiento tal y como se explica en los siguientes apartados.

8.1. Despliegue en local

Para el despliegue en local se ha empleado XAMPP que incorpora una versión de Apache. Se ha trabajado durante todo el proceso con el proyecto desplegado en local para su visionado y pruebas.

Para empezar, es necesario tener la carpeta del proyecto dentro de la carpeta de htdocs de XAMPP. Después de eso, sólo hace falta irse al fichero de configuración de Apache, httpd, añadimos la siguiente configuración para que pueda conectarse.

```
<VirtualHost *:80>
DocumentRoot "C:/xampp/htdocs/quickfood/quickfood"
ServerName quickfood.lmm-ies.edu
ErrorDocument 404 'La página no se ha encontrado'
</VirtualHost>
```

Ilustración 36 Configuración Apache para despliegue local

Una vez realizada la modificación, buscamos el fichero hosts y añadimos la siguiente línea para que nos reconozca nuestra ruta

```
127.0.0.1 quickfood.lmm-ies.edu
```

Ilustración 37 Hosts

Después de estas configuraciones ya es posible acceder a nuestra página de forma local y será como si la tuviéramos en un servidor.

8.2. Despliegue servidor

El despliegue en el servidor se ha realizado tanto en byethost como en lucushost. Se han elegido dos servidores por ser byethost un hosting gratuito con los riesgos que eso supone, por su parte, lucushost es hosting de origen español que cuenta con una prueba gratuita de 15 días para poder probar todas sus funcionalidades gratuitamente.

Para la subida de ficheros se ha utilizado el programa de Filezilla en su versión de Windows. Únicamente habría que introducir los datos que nos proporciona el hosting a la hora del registro como son el servidor, el usuario, la contraseña y el puerto. Una vez realiza la primera parte, arrastramos la carpeta de nuestro proyecto a la carpeta de public.

El proceso varía en función de los ficheros que deban subirse.

Una vez subidos los ficheros, lo primero que debemos hacer es borrar el archivo que se encuentra en la carpeta de bootstrap para que no dé ningún error a la hora de abrir la aplicación.

Una vez hemos borrado el fichero, lo siguiente es introducir los datos de la nueva base de datos (anteriormente importada) en el archivo .env . La configuración quedaría de la siguiente manera en mi caso.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=jsnfzwaq_quickfoodfp
DB_USERNAME=jsnfzwaq_lorena
DB_PASSWORD=mNkBs7uU7wvaUTe
```

Ilustración 38 .env del proyecto

En mi caso, el hosting empleado utiliza como db_host el localhost, es por eso que ese campo no ha habido que cambiarlo y sólo ha sido necesario cambiar los demás.

Para que la Base de datos funcione, en este hosting es necesario crear un usuario y otorgarle los permisos necesarios para que pueda entrar en ella, si esto no se hace, nos marcará la aplicación un error no pudiendo acceder a las vistas que tengan por detrás una consulta.

En cuanto al resto, ha sido realmente sencillo ya que Laravel incorpora un sistema de rutas en su archivo web.php, no ha sido necesario cambiar rutas debido a la propia estructura del framework ya que utiliza las llamadas al controlador y éste devuelve las vistas sin usar rutas absolutas.

9. Conclusiones

Una vez finalizado el proyecto he llegado a una serie de conclusiones.

La primera es que ha sido un proyecto que podría ayudarme a mí y a otras personas a realizar sus pedidos. A día de hoy es difícil encontrar un negocio local que tenga una aplicación para poder entrar y elegir lo que quieres pedir, es por eso que ha sido realizado con tanta motivación, para ayudarles tanto a ellos a ganar clientela como a los propios clientes que buscan nuevas formas de hacer sus compras.

Lo segundo que obtengo como conclusión es que ha servido para aprender a manejar un framework que, por la falta de tiempo en clase, no se había podido ver de mejor manera. Para ello he tenido que leer, documentarme y probar cosas sola como se haría en un trabajo real cuando tienes que aprender un lenguaje nuevo, así que ha servido para enseñarme la dura tarea que puede resultar hacer algo sin tener al lado a alguien que te lo pueda explicar, aunque a la vez ha sido muy reconfortante ver como finalmente el proyecto ha podido ver la luz, aunque al principio no tenía gran idea de Laravel.

En tercer lugar, la aplicación ha servido como un repaso general de todo lo aprendido durante estos dos años en clase. Se ha utilizado muchas de las cosas que se vieron tanto en primer año como en el segundo, ya sea JavaScript, PHP, HTML5, CSS y un largo etcétera. Siento que ha sido una forma de ver el avance que he conseguido ya que entré al ciclo sin grandes conocimientos y a día de hoy sería capaz de hacer una aplicación web. En ella se puede ver el desarrollo obtenido durante estos dos años.

Para finalizar, quería exponer que la realización de este proyecto ha supuesto un éxito personal al haberlo acabado. Me ha enseñado que hay que aprender poco a poco y que es posible aprender un lenguaje nuevo si se le pone empeño y ganas, al final la constancia y las ganas de aprender algo nuevo son lo que hacen que las cosas tengan un buen resultado.

10. Bibliografía y Webgrafía

Para la realización de este proyecto han sido necesarias varias páginas de ayuda además de cursos que han sido de utilidad para aprender a utilizar Laravel o ampliar conocimientos que ya sabía.

- Curso de Laravel en Udemy

<https://www.udemy.com/course/master-en-php-sql-poo-mvc-laravel-symfony-4-wordpress/learn/>

- Curso de JS en Udemy

<https://www.udemy.com/course/master-en-javascript-aprender-js-jquery-angular-nodejs-y-mas/learn/>

- Documentación de Laravel

<https://laravel.com/docs/8.x>

- Documentación de JQuery

<https://api.jquery.com/>

- Información para hacer el buscador

<https://www.webslesson.info/2018/11/laravel-column-sorting-with-pagination-using-ajax.html>

- Documentación de Bootstrap4

<https://getbootstrap.com/docs/4.2/getting-started/introduction/>

<https://www.w3schools.com/bootstrap4/>

MEMORIA PROYECTO CFGS DESARROLLO DE APLICACIONES WEB

- PHPDocumentor

<https://github.com/phpDocumentor/phpDocumentor>

- Ayudas

<https://es.stackoverflow.com/>

- Faker

<https://github.com/fzaninotto/Faker>

- Apuntes de clase de Desarrollo en Entorno Servidor.