



Jessica Rodríguez Mira

Proyecto Final

Desarrollo de Aplicaciones Web IES Campanillas

Email [Jessica.rod.mir@gmail.com](mailto:Jessica.rod.mir@gmail.com)

## Contenido

1-Introducción a la aplicación.....	3
Descripción.....	3
2-Tecnologías.....	4
Laravel .....	4
Vue.js .....	4
Tailwind.....	4
Otras librerías .....	5
3-"Manual de usuario" (Vistas).....	5
Parte Cliente.....	5
Index.....	5
Conocenos.....	6
Reservas Cliente.....	6
Registro .....	7
Login .....	7
Parte Usuarios /Admin.....	8
Home .....	8
Productos.....	10
4-Proceso de Desarrollo.....	15
ERD .....	15
Migraciones .....	16
Models.....	16
Index.....	17
Conócenos.....	18
Reservas .....	19
Registro .....	19
Login .....	19
Home .....	20
Productos y Drinks .....	22
5-Despliegue.....	27
Configuración Laravel .....	27
Configuración Node.js .....	29
Control de versiones.....	29
Configuración de la base de datos .....	30



## 1-Introducción a la aplicación

En este punto se va a hacer una presentación de la aplicación, a que público va destinada, cual es su funcionalidad, y descripción de la aplicación.

### Descripción

TastyKit es una aplicación destinada al sector de la hostelería, se trata de un sistema gestor de productos, bebidas, platos, reservas, creación de comanda y gestión de tickets. TastyKit tiene todo lo necesario para poder gestionar un restaurante desde un aspecto un poco más amigable que el de algunos productos que se encuentran hoy en día en el mercado. Por otro lado, también consta de una parte para los clientes, ellos van a poder acceder a estos recursos y podrán familiarizarse un poco más con nuestro restaurante, aparte de poder realizar su reserva desde aquí, todo esto sin necesidad de logueo por parte del cliente. En la parte de los usuarios nos encontramos con toda la funcionalidad nombrada anteriormente. Por cada producto, bebida, plato, usuario, reserva o comanda vamos a poder realizar una creación de está, una modificación o un borrado. Para la parte de usuarios nos vamos a encontrar con ciertas restricciones, ya que la aplicación también cuenta con un sistema de roles en el que actualmente tenemos el rol de Admin y el de User. Los Admin van a tener permiso, sobre todo, pudiendo estos otorgar el permiso de Admin a cualquier usuario y de esta manera este podrá acceder a la gestión de usuarios. En la parte de perfil de usuario de la aplicación vamos a tener la opción de cambiar nuestros datos, cambiar nuestra contraseña, habilitar una verificación en dos pasos con Google Authenticator y código QR, borrar la cuenta o cerrar sesiones en otro navegador, esto nos permite un control total sobre nuestro perfil.

Una vez que conocemos un poquito más está aplicación debemos hacer una pequeña anotación sobre el nombre. ¿Por qué TastyKIT? La respuesta sería porque es el KIT más sabroso para la gestión de tu restaurante.

## 2-Tecnologías

Para el desarrollo de este proceso se han usado las tecnologías que se van a nombrar a continuación:

### Laravel

Laravel es un framework de código abierto para PHP, lenguaje que se ha utilizado para el backend de nuestra aplicación. Laravel consta de una serie de Scaffolding (Starter kits) predeterminados para empezar a desarrollar un proyecto

- Laravel/ui
- Laravel Breeze
- Laravel Jetstream

Este último ha sido el utilizado para el desarrollo de el proyecto, es el Scaffolding más completo de Laravel. Podemos diferenciarlo en dos tecnologías propias, Livewire e Inertia. Livewire utiliza el sistema de plantillas de Blade para la parte del front, mientras que Inertia utiliza Vue.js para nuestro proyecto se ha usado Inertia con Vue.js. Laravel Jetstream cuenta con implementación de registro, verificación de correo electrónico, autenticación en dos pasos y administración de sesiones, esta parte nos facilita un poco el principio del desarrollo del proyecto.

### Vue.js

Esta es la tecnología que se utiliza con Inertia para la parte del front. Vue es un framework de JavaScript que nos permite dar lógica y mucha funcionalidad a nuestro front, es un framework muy utilizado y demandado actualmente.

### Tailwind

Para la parte de diseño se ha usado el framework de CSS Tailwind, este es un framework que nos permite mucha libertad ya que es muy parecido a CSS y nos permite escribir esto en un tiempo menor y con algunas funcionalidades muy tediosas de construir.

Esta tecnología junto con Laravel y Vue son la base y grosor de nuestro proyecto.

## Otras librerías

- Moment.js :Se ha utilizado la librería Moment para el formateo de fechas y horas.
- Spatie: para la gestión de permisos de Laravel. Esta librería crea automáticamente una serie de tablas que nos proporciona mucha ayuda sobre la gestión de los distintos roles



- Tone.js: para el uso de sonido sobre los iconos de la página Index.

## 3-“Manual de usuario” (Vistas)

La aplicación podemos dividirla en dos partes principales, la destinada a clientes que no es necesario el logueo, y la parte de gestión del restaurante destinada a los usuarios.

### Parte Cliente

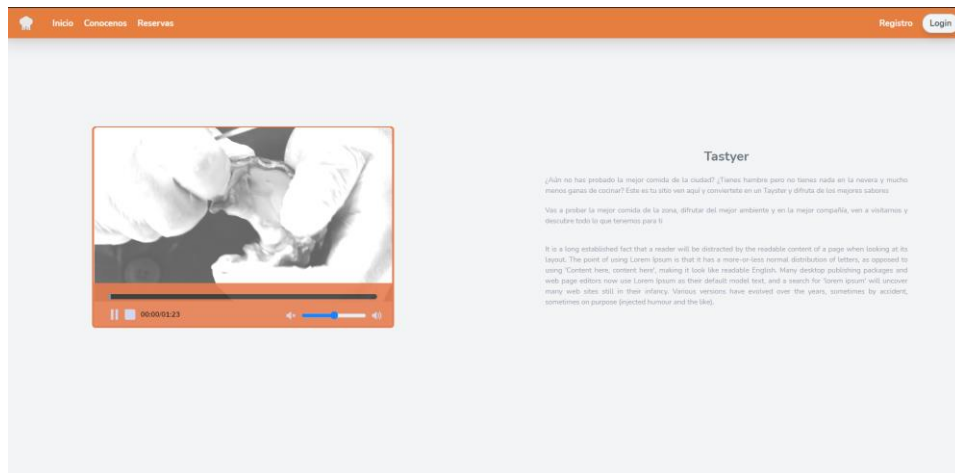
#### Index

Nada más entramos a la página nos encontramos con la página de Inicio, está a su vez permite el acceso a las Reservas, la sección multimedia, el logueo y el registro. En esta vista si el usuario pulsa sobre el logo podrá escuchar un sonido.



## Conocenos

Si pulsamos en la barra de navegación sobre el link Conócenos, nos llevará a esta página, esta página está compuesta de un video y un texto, en esta parte el cliente podrá acercarse más a nosotros viendo nuestra producción y conociendo algún detalle más.

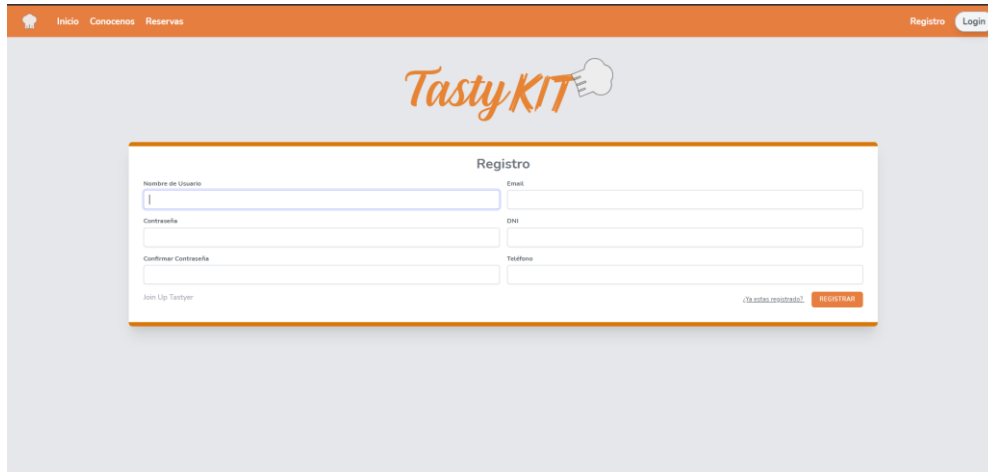


## Reservas Cliente

Pulsando el botón de Reservas, nos llevará a un formulario en el cual el cliente podrá registrar su reserva, de esta manera el restaurante más adelante podrá ponerse en contacto con el cliente para cualquier necesidad. Este formulario cuenta con reglas de validación, de esta manera nos aseguramos que los datos rellenos no sean erróneos, o tengamos la falta de ellos.

## Registro

En la parte superior de la derecha nos encontramos con un link Registro, este nos envía a otro formulario que en este caso servirá para poder acceder a la aplicación como usuario, este formulario también cuenta con reglas validación para asegurar su correcto relleno.



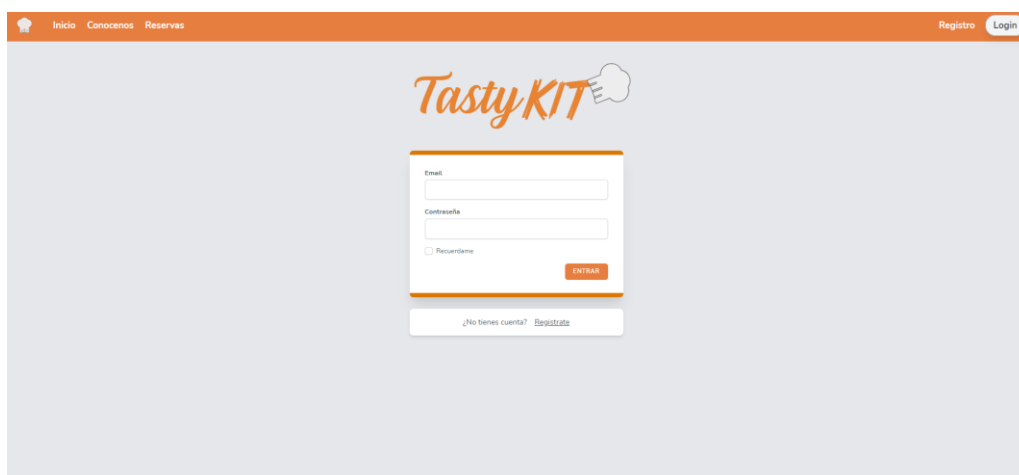
The screenshot shows the registration form for TastyKIT. The form is titled "Registro" and is located in the center of the page. It has a light gray background and a white border. The form contains the following fields:

- Nombre de Usuario
- Contraseña
- Confirmar Contraseña
- Email
- DNI
- Teléfono

At the bottom of the form, there is a link "¿Ya tienes cuenta?" and a button "REGISTRAR".

## Login

A su derecha tenemos el botón de login mediante el cual si el usuario tiene cuenta podrá loguearse, en esta vista contamos también con el link de registro en la parte inferior del formulario.



The screenshot shows the login form for TastyKIT. The form is titled "Login" and is located in the center of the page. It has a light gray background and a white border. The form contains the following fields:

- Email
- Contraseña

Below the password field, there is a checkbox labeled "Recuérdame". At the bottom of the form, there is a button "ENTRAR".

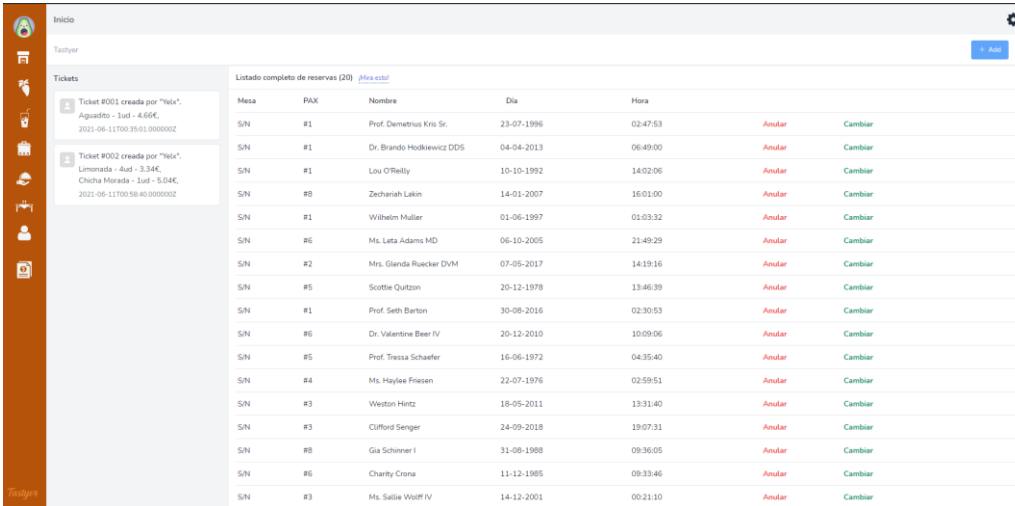
Below the login form, there is a link "¿No tienes cuenta? Regístrate".



## Parte Usuarios /Admin

### Home

En esta sección tras nuestro logueo o registro vamos a poder acceder a la pestaña de Inicio. En esta pestaña vamos a poder gestionar las reservas pudiendo asignar mesa a una reserva concertada, dar de alta una reserva, modificar o anular una reserva, en la parte de la izquierda tenemos un panel en el que vamos a poder ir viendo los tickets que se van creando. Por último destacar que esta pestaña sería el icono de la casita en el menú de navegación.

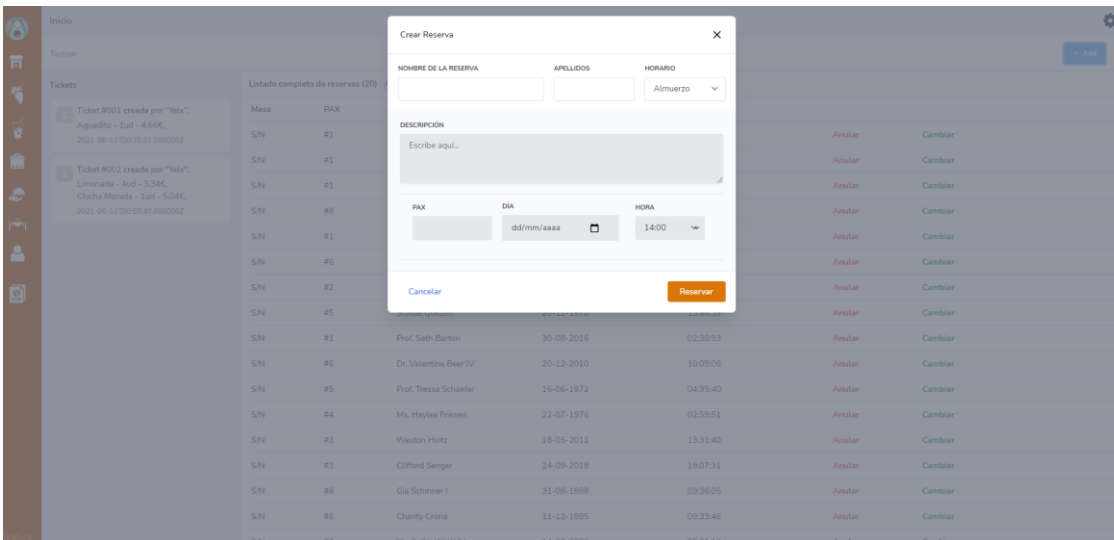


The screenshot shows the Home dashboard. On the left is a sidebar with navigation icons. The main content area has a header with 'Inicio' and a settings icon. Below the header is a 'Tickets' section with two tickets listed. To the right is a 'Listado completo de reservas (20)' section with a table of reservations.

Mesa	PAX	Nombre	Día	Hora		
SIN	#1	Prof. Demetrius Kins Sr.	23-07-1996	02:47:53	Anular	Cambiar
SIN	#1	Dr. Brando Hockiewicz DDS	04-04-2013	06:49:00	Anular	Cambiar
SIN	#1	Lou O'Reilly	10-10-1992	14:02:06	Anular	Cambiar
SIN	#5	Zechariah Lukin	14-01-2007	16:01:00	Anular	Cambiar
SIN	#1	Wilhelm Muller	01-06-1997	01:03:32	Anular	Cambiar
SIN	#6	Ms. Leta Adams MD	06-10-2005	21:49:29	Anular	Cambiar
SIN	#2	Mrs. Glenda Ruacker DVM	07-05-2017	14:19:16	Anular	Cambiar
SIN	#5	Scottie Quitzon	20-12-1978	13:46:39	Anular	Cambiar
SIN	#1	Prof. Seth Barton	30-08-2016	02:30:53	Anular	Cambiar
SIN	#6	Dr. Valentine Beer IV	20-12-2010	10:09:06	Anular	Cambiar
SIN	#5	Prof. Tressa Schaefer	16-06-1972	04:35:40	Anular	Cambiar
SIN	#4	Ms. Haylee Friesen	22-07-1976	02:59:51	Anular	Cambiar
SIN	#3	Weston Hintz	18-05-2011	13:31:40	Anular	Cambiar
SIN	#3	Clifford Senger	24-09-2018	19:07:31	Anular	Cambiar
SIN	#5	Gia Schinner I	31-08-1988	09:36:05	Anular	Cambiar
SIN	#6	Charity Crona	11-12-1985	09:33:46	Anular	Cambiar
SIN	#3	Ms. Sallie Wolff IV	14-12-2001	00:21:10	Anular	Cambiar

La creación, modificación y anulado de reservas cuentan con una ventana modal.

### Crear Reserva



The screenshot shows the 'Crear Reserva' modal form. It has fields for 'NOMBRE DE LA RESERVA', 'APELLIDOS', and 'HORARIO' (with a dropdown menu). There is a 'DESCRIPCIÓN' field with a placeholder 'Escribe aquí...'. Below these are fields for 'PAX', 'DÍA' (with a date picker), and 'HORA' (with a time picker). At the bottom are 'Cancelar' and 'Reservar' buttons.

## Modificar Reserva

Inicio

Tasty

Tickets

**Ticket #001 creada por "Yela".**

Aguaflo - 1ud - 4.65€, 2021-08-11T00:35:01.000000Z

**Ticket #002 creada por "Yela".**

Limonada - Aud - 3.34€, Chicha Morada - 1ud - 5.04€, 2021-08-11T00:58:40.000000Z

Listado completo de reservas (20)

Mesa	PAX	NOMBRE DE LA RESERVA	APELLIDOS	HORARIO	
S/N	#1	Prof. Demetrius Kris Sr.	Prosacco	Almuerzo	
S/N	#1	<b>DESCRIPCIÓN</b>			
S/N	#1	Inqui dolores corporis velit qui officia harum est. Iusto sed est voluptatum voluptas vel corrupti rerum. Eos ut modi dicta ex. Soluta ab quo nihil harum et.			
S/N	#0	PAX	MESA	DÍA	HORA
S/N	#1	1		23/07/199E	
S/N	#0				
S/N	#2				
S/N	#5				
S/N	#1	Prof. Seth Barton	30-08-2016	02:30:53	
S/N	#6	Dr. Valentine Beer IV	20-12-2010	10:09:06	
S/N	#5	Prof. Treessa Schaefer	16-06-1972	04:35:40	
S/N	#4	Ms. Haylee Fiesan	22-07-1976	02:59:51	
S/N	#3	Winston Hintz	18-05-2011	13:31:40	
S/N	#3	Clifford Senger	24-09-2018	19:07:31	
S/N	#0	Gia Schinner I	31-08-1988	09:36:05	
S/N	#6	Charity Crona	11-12-1985	09:33:46	
S/N	#3	Mr. Salma Waelff Jr	14-13-2001	00:11:10	

Modificar Reserva

Cancelar

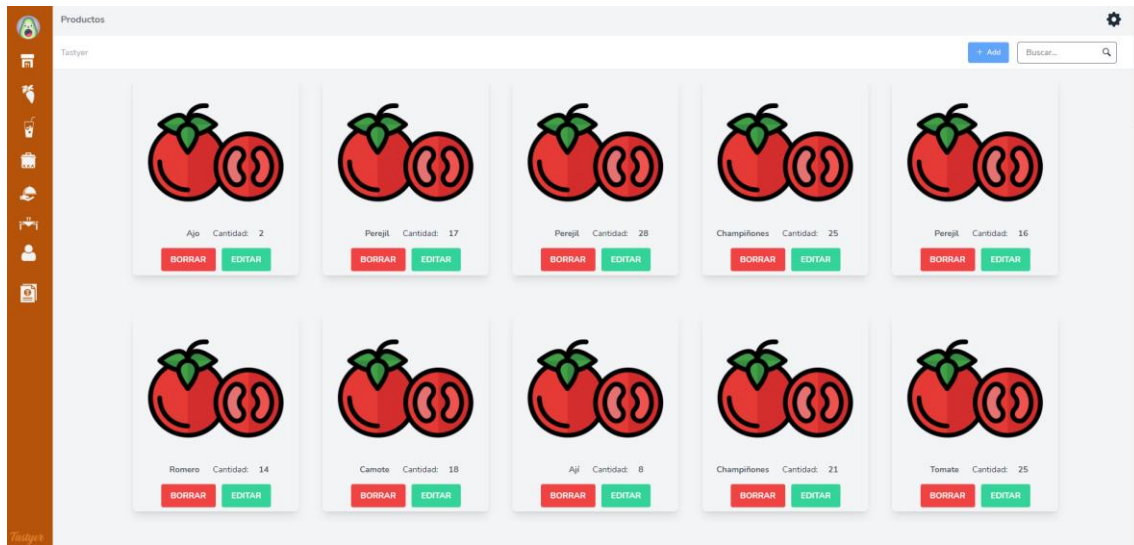
Guardar

## Anular reserva

Reservas (20) <a href="#">¡Mira esto!</a>		
Nombre	Día	Hora
Prof. Demetrius Kr		02:47:53
Dr. Brando Hodkie		06:49:00
Lou O'Reilly	La reserva a nombre de Prof. Demetrius Kris Sr. para 1pax del día 1996-07-23	
Zechariah Lakin		14:02:06
		16:01:00
Wilhelm Muller		01:03:32
Ms. Leta Adams MD	06-10-2005	21:49:29
Mrs. Glenda Ruecker DVM	07-05-2017	14:19:16
Scottie Quitzon	20-12-1978	13:46:39

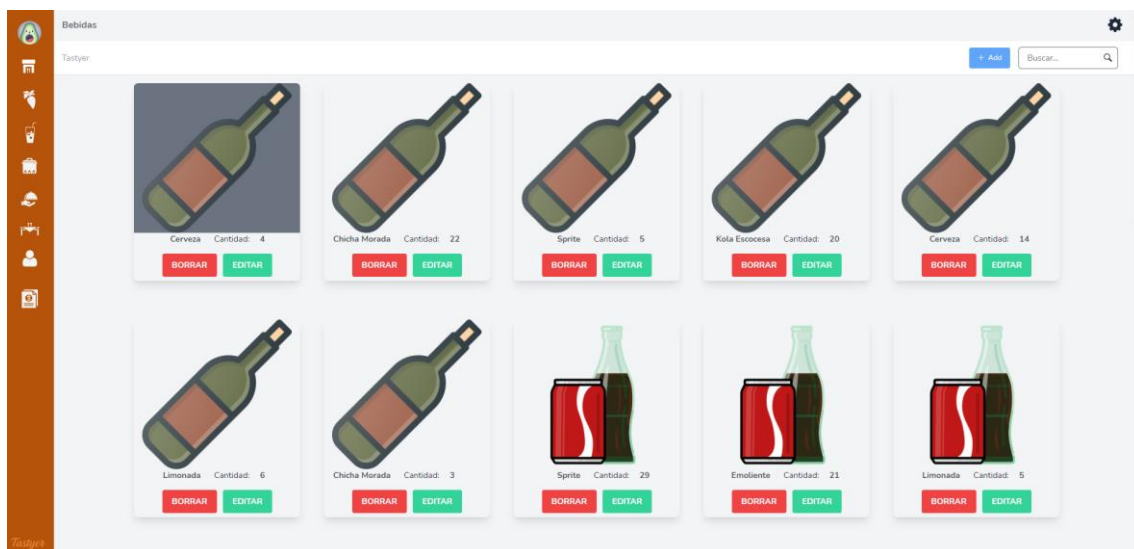
## Productos (Icono Zanahoria)

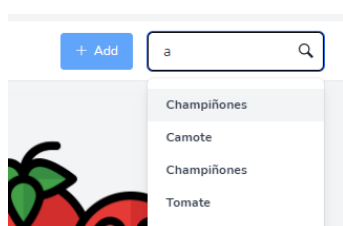
Esta sección cuenta con unas cards que contendrán datos de nuestros distintos productos, sobre el botón de editar aparecerá una modal similar a la anterior y es aquí donde podremos editar nuestro producto, mientras que en el botón Add podremos añadir un producto nuevo, y por último en el botón de borrar podremos eliminar nuestro producto.



## Bebidas

Esta sección es muy parecida a la anterior lo único que cambia es la card el contenido, hay que destacar que en estos dos objetos tenemos en la parte superior de la derecha una barra buscadora que si lo encuentra nos abre una modal para poder modificarlo.





Alta Bebida

Nombre de la bebida

Cantidad

Precio

Categoría

Descripción

Escribe aquí...

☐ Imagen por defecto

Cancelar

Guardar

Mas tarde nos encontramos con la sección de platos, en está sección los platos están ordenados por tipo Entrante, primeros... Por otro lado, se podrán editar, generar y borrar dicho plato. Los bloques se pueden colapsar para una mayor comodidad

</

En la sección de comandas tenemos una tabla en la que vamos a poder seleccionar todos los platos mesa y bebidas del cliente, teniendo a la derecha el ticket que estamos generando, contando este con un salto de página.

Comandas

GENERADOR DE COMANDAS

SELECCIONAR MESA

Mesa	1	Pax: 1-5	LIBRE	X	SELECCIONAR
Mesa	2	Pax: 1-7	LIBRE	X	SELECCIONAR
Mesa	3	Pax: 1-5	LIBRE	X	SELECCIONAR
Mesa	4	Pax: 1-4	LIBRE	X	SELECCIONAR
Mesa	5	Pax: 1-8	LIBRE	X	SELECCIONAR
Mesa	6	Pax: 1-9	LIBRE	X	SELECCIONAR
Mesa	7	Pax: 1-9	LIBRE	X	SELECCIONAR
Mesa	8	Pax: 1-3	LIBRE	X	SELECCIONAR
Mesa	9	Pax: 1-7	LIBRE	X	SELECCIONAR
Mesa	10	Pax: 1-9	LIBRE	X	SELECCIONAR
Mesa	11	Pax: 1-5	LIBRE	X	SELECCIONAR
Mesa	12	Pax: 1-8	LIBRE	X	SELECCIONAR
Mesa	13	Pax: 1-3	LIBRE	X	SELECCIONAR

COMANDA

Nº Ticket: #00003 Mesa: 000

TOTAL 0€

GENERAR COMANDA

Mesa	22	Pax: 1-2	LIBRE	X	SELECCIONAR
Mesa	23	Pax: 1-9	LIBRE	X	SELECCIONAR
Mesa	24	Pax: 1-1	LIBRE	X	SELECCIONAR

BEBIDAS

Con Alcohol	Cerveza	7.76€	X	5	AÑADIR
Con Alcohol	Chicha Morada	8.56€	X	3	AÑADIR
Con Alcohol	Sprite	9.02€	X	8	AÑADIR
Con Alcohol	Kola Escocesa	5.93€	X	6	AÑADIR
Con Alcohol	Cerveza	7.88€	X	8	AÑADIR
Con Alcohol	Limonada	3.34€	X	8	AÑADIR
Con Alcohol	Chicha Morada	5.04€	X	1	AÑADIR
Refrescos	Sprite	3.82€	X	6	AÑADIR
Refrescos	Emuliente	3.96€	X	3	AÑADIR
Refrescos	Limonada	2.69€	X	6	AÑADIR
Refrescos	Sprite	2.2€	X	1	AÑADIR
Refrescos	Refresco de paja	2.17€	X	1	AÑADIR

COMANDA

Nº Ticket: #00003 Mesa: 000

- Cerveza	8Ud	7.88€
- Kola Escocesa	6Ud	5.93€
- Sprite	8Ud	9.02€
- Chicha Morada	3Ud	8.56€
- Limonada	8Ud	3.34€
- Emuliente	3Ud	3.96€
- Refresco de paja	3Ud	3.95€
- Pisco	2Ud	6.17€
TOTAL		294.75€

Siguiente

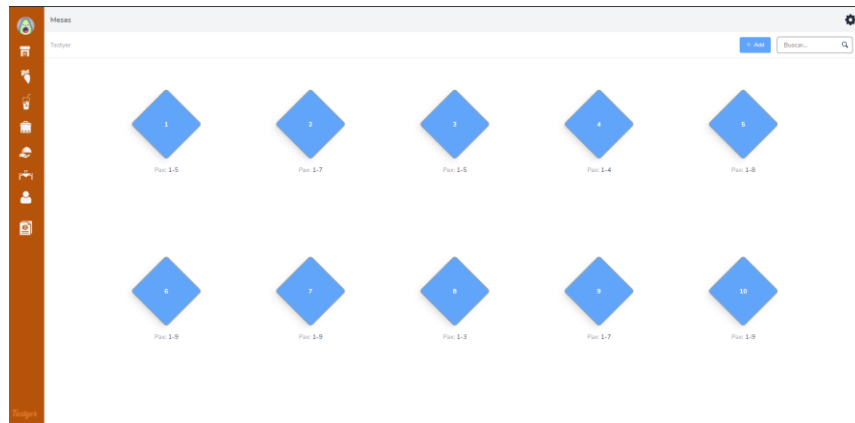
GENERAR COMANDA

Cuando generemos la comanda vamos a poder visualizarla en la última pestaña, facturas. Esto tiene un control de que no puedas dejar la mesa vacía, por otro lado si se elige una mesa hay que deseleccionarla antes de elegir otra

Mesa	12	Pax: 1-8	LIBRE	X	SELECCIONAR
Mesa	13	Pax: 1-3	OCUPADA	X	SELECCIONAR

## Mesas

En esta sección se podrán visualizar las mesas viendo si están ocupadas o no.



## Usuarios

En la siguiente pestaña tenemos la pestaña de usuarios en esta podemos ver datos sobre los usuarios y modificar algunos datos, esto solo se podrá acceder con un rol de Admin ya que tanto la ruta como el botón están habilitados solo para ese rol. En la parte derecha volvemos a ver los tickets que se están generando en el momento

DNI	Puesto	Nick	Telefono	Email	BORRAR	EDITAR
572896306	Admin	Carley Mueller	1-928-249-0012	henry81@example.com	BORRAR	EDITAR
710011327X	Admin	Krystal Powlowski	214-726-9435	glady80@example.org	BORRAR	EDITAR
30537887X	Admin	Leonel Reichert	1-341-264-2038	gruecker@example.com	BORRAR	EDITAR
7485116762	Admin	Arnulfo Bergstrom	+1-442-545-0409	medhurst.kayleigh@example.net	BORRAR	EDITAR
0856295256	User	Tomasz Murphy I	1-959-351-7489	joan.crooks@example.com	BORRAR	EDITAR
8291440913	User	Mada Olson	+1 (651) 848-9503	zoconner@example.org	BORRAR	EDITAR
9088011753	User	Nash Marquardt	+1 (727) 590-2530	zboncak.florine@example.com	BORRAR	EDITAR
1309644586	User	Dominique Conn	+1 (843) 905-6794	wolff.morton@example.com	BORRAR	EDITAR
7820166606	User	Mr. Misael Collins MD	+1-520-501-1671	carmelo62@example.net	BORRAR	EDITAR
248888491X	User	Rowan Donnelly	435-966-0305	marcos.beier@example.com	BORRAR	EDITAR
421570202X	User	Jarred Hauck	(970) 986-5154	bauch.burley@example.org	BORRAR	EDITAR
9901540348	User	Ruthe McClure	1-551-563-8800	mcglynn.karianne@example.com	BORRAR	EDITAR
9477314307	User	Mr. Abdullah Kuhn	820.914.6408	swaniawski.mathilde@example.org	BORRAR	EDITAR
705608947X	User	Kathleen Runolfsdottir V	743.920.1697	istamm@example.com	BORRAR	EDITAR
6491670366	User	Mrs. Eusebia Rouse	(879) 667-1116	ismay118@example.com	BORRAR	EDITAR

### Tickets

Ticket #001 creada por "Yela".  
Aguadito - 1ud - 4.66€,  
2021-06-11T00:35:01.000000Z

Ticket #002 creada por "Yela".  
Limonada - 4ud - 3.34€,  
Chicha Morada - 1ud - 5.04€,  
2021-06-11T00:58:40.000000Z

## Tickets

En la última pestaña tenemos los tickets, las facturas en esta podremos ver el total recolectado hasta el momento y las facturas que tenemos abiertas o cerradas, pudiendo cerrar alguna aquí si quisiéramos.

**Facturas**

Abiertas / Cerradas    Total ganado en caja

1	1	6.75€
ABIERTOS	CERRADOS	TOTAL

Listado completo de reservas (0) [Ver más](#)

Nº Ticket	Comanda	Camarero	Precio	
#1	Aguafrito - Sud - 4.66€	Yelx	6.75€	CERRADO
#2	Limónada - Sud - 3.34€, Chicha Morada - Sud - 5.04€	Yelx	18.4€	CERRAR

**ABIERTO** Ticket #002

Comanda listado

Consumiciones:

Limónada - Sud - 3.34€, Chicha Morada - Sud - 5.04€

Creada por

Yelx

TOTAL: 18.4€

## Perfil

Para acceder a nuestro perfil tenemos en la parte superior derecha una ruedecita, esta nos llevará a nuestro perfil o nos permitirá volver al inicio o cerrar sesión.

**Perfil**

**Inicio**  
**Perfil**  
**Cerrar Sesión**

**Información del Perfil**  
Aquí podrás actualizar tu información de perfil.

Nombre	Apellido
<input type="text" value="Yelx"/>	<input type="text" value="jordi.morales@gmail.com"/>
Correo	Telefono
<input type="text" value="YELX@GMAIL.COM"/>	<input type="text" value="622456789"/>

**Actualizar Perfil**

**Cambiar Contraseña**  
Puedes cambiar tu contraseña actual por una nueva.

Contraseña Actual	Nueva Contraseña	Confirmar Nueva Contraseña
<input type="password"/>	<input type="password"/>	<input type="password"/>

**Actualizar Contraseña**

**Verificación en dos pasos**  
Nuestro sistema requiere una verificación en dos pasos para garantizar la seguridad de tu cuenta.

**No ha habido la verificación en dos pasos**  
Cuando se solicita la verificación en dos pasos, se le pedirá un código de verificación.

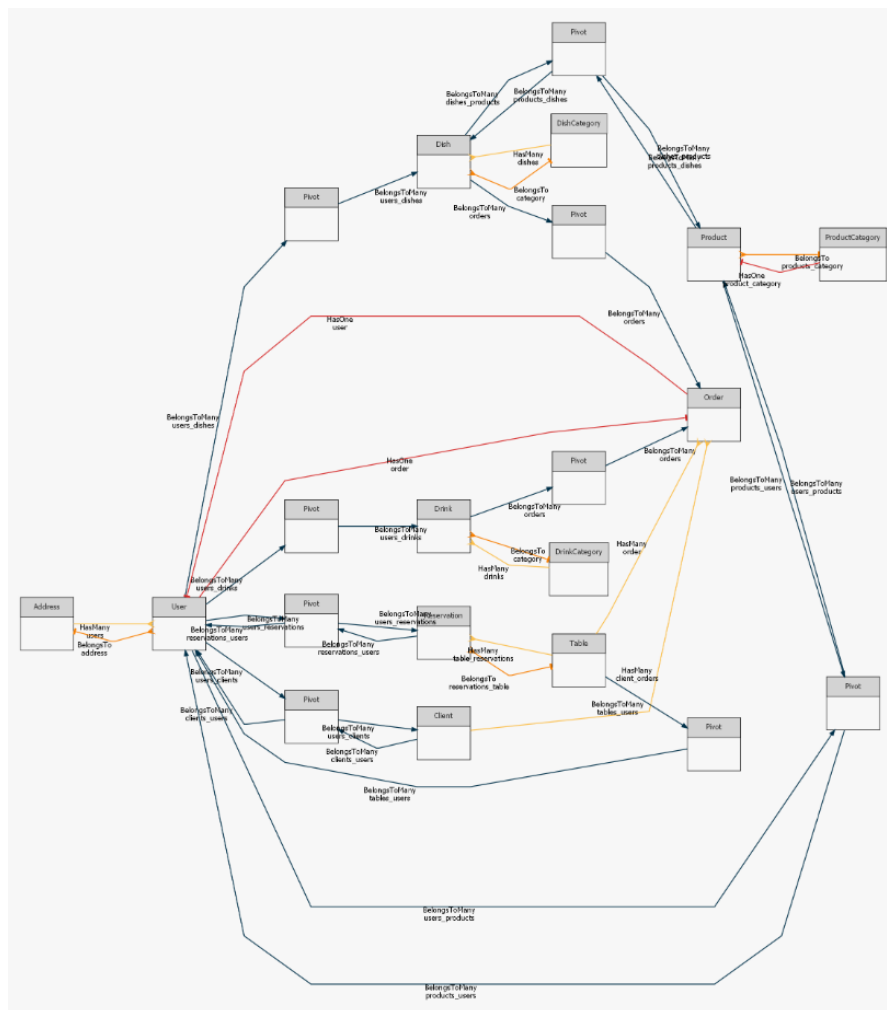
Estas serían todas las vistas de la aplicación, en el siguiente punto vamos a entrar un poco más en detalles técnicos del desarrollo.

## 4-Proceso de Desarrollo

En este apartado vamos a desglosar el proceso de desarrollo de la aplicación en distintas partes según el orden de creación que se ha seguido.

### ERD

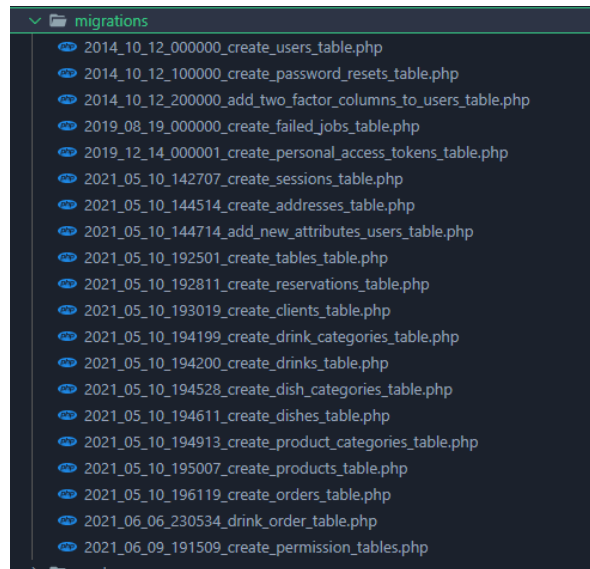
Lo primero que se planteó fue el diagrama de entidad relación, tras una serie de modificaciones tras el transcurso del desarrollo del proceso, el diagrama de entidad relación sería el que se muestra a continuación.





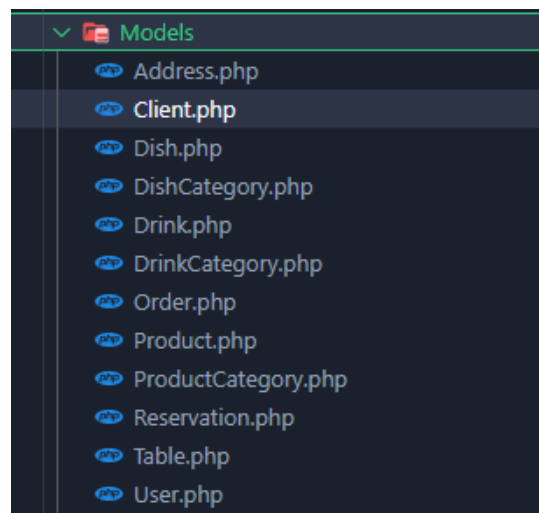
## Migraciones

El siguiente paso a realizar fue las migraciones, una vez teníamos nuestras tablas había que crearlas quedando la siguientes:



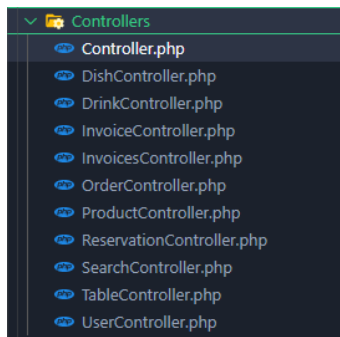
## Models

Tras esto se crearon los modelos en lo que íbamos a tener las distintas relaciones y métodos relacionados con ese objeto propio.



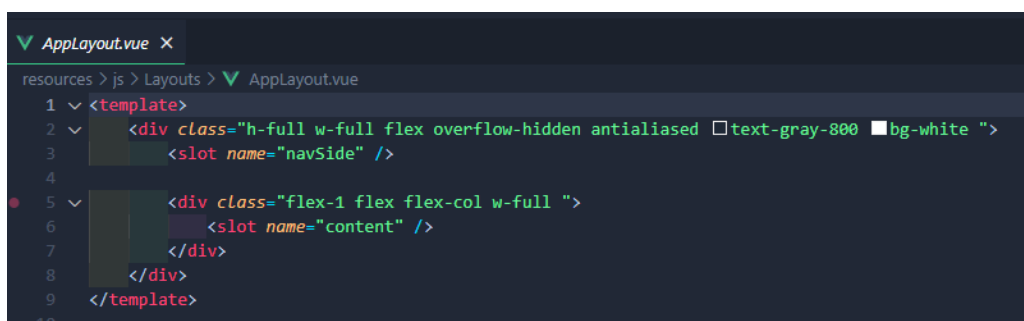
## Controladores

El siguiente paso fue crear los controladores que íbamos a usar de los modelos quedándonos con los siguientes



De esta forma teníamos el “esqueleto principal”. Tras esto podemos hablar de flujos:

Se ha reutilizado un componente Vue principal que se encuentra en Layouts para utilizarlo de base de nuestras distintas vistas.



En este componente se le han creado dos slot uno para la barra de navegación lateral y otro para el contenido de la página.

## Index (Cliente)

Para esta vista se ha reutilizado el componente que trae Laravel index, pero se ha modificado entero, en esta vista se hace uso de 2 componentes también creados navBar y letras. Metodos del componente:

- Audio : al pulsar click sobre las letras o el icono se activara un sonido que por defecto se autoiniciara pero tras 1 segundo se detendrá

```

1  <!-- an hour ago * Despliegue -->
2  <template>
3  <div class="h-screen bg-gray-100">
4    <!--Añadimos barra de navegación -->
5    <nav-bar />
6    <div class="h-4/5 flex items-center w-full">
7      <div class="w-full flex-wrap text-center items-center justify-center">
8        <h1 class="font-bold text-gray-400 text-3xl">
9          Welcome Tastyer
10        </h1>
11        <!-- Letras del Logo -->
12        <div class="flex justify-center @click=audio()" style="width: 100%;">
13          <letras width="50%"/>
14        </div>
15      </div>
16    </div>
17  </div>
18 </template>
19
20
21 <script>
22   import Letras from '@/Components/Letras'
23   import NavBar from '@/Components/NavBar'
24   import anime from 'animejs'
25   import * as Tone from 'tone'
26   export default {
27     components: {
28       Letras,
29       NavBar
30     },
31     methods: {
32
33       audio(){
34
35         var player = new Tone.Player("@../audio/pop.mp3").toMaster();
36
37         player.autostart = true;
38         setTimeout(() => {
39           player.autostart = false
40         }, 1000);
41       }
42     }
43   }
44 </script>

```

El componente Letras solo contiene el logo con las letras, mientras que el componente NavBar es la barra de navegación superior que vemos nada más entrar en la página, este lo único que contiene son esos links.

## Conócenos

Conocenos llamado Images.vue en la carpeta Pages->Client, se compone de un canvas en el que se le ha insertado un video, se ha procesado en blanco y negro y se le ha añadido una interfaz de reproducción, a su derecha encontramos el texto descriptivo. Los métodos que contiene este componente son:

- updateVolumen : método que sube el volumen del video
- mute(): método que silencia el video.
- barProgress : este controla el tiempo transcurrido de video para plasmarlo en forma de barra progresiva.
- Time : método que calcula la duración del video y los minutos transcurridos.
- Stopped este se encarga del stop del video
- procesarCuadros : este es el encargado de recoger el canvas, dibujar el video y cambiarle el color a blanco y negro.

En este componente se ha utilizado scss, se ha creado una carpeta scss a la que se le ha añadido dos ficheros scss uno con los estilos globales (app) y otro con variables (var).

## Reservas

El componente Reservation.vue contiene un formulario del cual recogemos los datos para dar de alta una reserva. Los métodos de Reservation son:

- checkForm: chequea si están los campos necesarios rellenos, si no es así lanza un error.
- -createRes: método asíncrono (utilizando axios) que crea una reserva.
- changeModal : este método es el encargado de abrir y cerrar la modal de confirmación al intentar crear una reserva, si está en true devuelve false y viceversa. Este último método se ha usado para todos los casos del modal.

## Registro

Para este registro nos hemos beneficiado de las ventajas que ofrece Laravel, teniendo ya este creado y solo teniendo que agregar un par de cambios y añadirle estilos, en este caso cuando vamos a registrarnos tenemos un validator en mitad comprobando que todo este correcto, este es CreateNewUser en la carpeta Fortify.

## Login

En este caso ocurre lo mismo que en el anterior, tenemos un Login con 2 campos email y contraseña, este formulario controla y lanza un error si el usuario no existiera o la contraseña no fuera la correcta.

Tanto el registro como el login nos llevan al siguiente componente Home, este está relacionado con el citado anteriormente App\_layout y Dashboard ya que en este caso se está cargando un componente home desde Dashboard, siendo esta la vista de inicio de los usuarios.

```
You, 2 hours ago | 1 author (You)
<template>
  <app-layout>
    <template #navSide>
      <nav-side />
    </template>

    <template #content>
      <home />
    </template>
  </app-layout>
</template>

<script>

import Home from '@Pages/Admin/Home'
import AppLayout from '@Layouts/AppLayout'

export default {
  components: {
    AppLayout,
    Home,
  },
}
</script>
```

## Home (Inicio users)

Este componente utiliza como plantilla app-layout y a su vez utiliza una serie de componentes, métodos, data y watchers:

### Data

- total: default 0, variable que va a contener el total de registros de reservas.
- Reservas: array que va a contener todas las reservas se va a rellenar con el método asíncrono getReservas.
- Reservald: default String vacío, esta data se va a rellenar con la emisión del id desde el hijo TableReservas.
- showModal: default false, encargada del cierre y apertura de la modal.
- upda: default false, se va a utilizar desde un watch que va a estar mirando si su valor cambia a true para refrescar las reservas y el total con los métodos asíncronos getReservas y getTotal.
- Reserv: object con estructura de una reserva, se va a utilizar junto con el método asíncrono createReserv(reserv), se le pasará como parámetro.
- showModalAdd: default false, misma funcionalidad que la de showModal, pero en ese caso se va a utilizar con la modal de dar de alta una reserva.
- errorDay: default false, utilizada para lanzar error del input day.
- errorTime: default false, utilizada para lanzar error en el input time.
- errorPax: default false, se usa para lanzar el error del input pax.
- Error: default false, se utiliza para asignar error global en el método checkForm

### Métodos

- changeModalAdd: método que cambia el valor a showModalAdd
- changeModal: método que cambia el valor de showModal
- checkForm: método con filtros para el control de datos erróneos en el formulario
- resetDataMod: resetea los datos de la modal modificación
- resetData: resetea los datos de la modal de alta

### Asíncronos

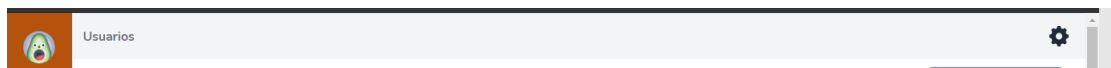
- updateReserva(reserva,id): con los datos de la reserva comprueba que no haya ningún error y con axios llama a /reservation-update/{id}, esta ruta tiene un controlador a la que le asigna los métodos que se muestran en la foto
- createReserv: método que llama a la ruta /create-reservation encargada de dar de alta una reserva
- getReserva método que llama a la ruta / reservation{id} devuelve el registro por el id
- getReservas: método que llama a la ruta /reservation-list, este devuelve un array de todos los registros de las reservas
- getTotal: llama a la ruta /reservation-total devuelve el número total de registros.
- 

```
Route::post('/create-reservation', [ReservationController::class, 'create'])->name('create-reservation');
```

```
//Reservation
Route::get('/reservation-list', [ReservationController::class, 'getReserv'])->name('reservation-list');
Route::get('/reservation-total', [ReservationController::class, 'getTotal'])->name('reservation-total');
Route::get('/reservation/{id}', [ReservationController::class, 'show'])->name('reservation-show');
Route::put('/reservation-update/{id}', [ReservationController::class, 'updateReserv'])->name('reservation-update');
Route::delete('/delete-reservation/{id}', [ReservationController::class, 'destroy'])->name('delete-reservation');
```

## Componentes

- navSide: barra de navegación lateral esta barra solo contiene los inertia-link para poder navegar.
- 
- headerNav: componente que forma parte de la todas las vistas que contiene el tipo de registro en el que nos encontramos y el icono con el desplegable de settings.



- headTab componente creado para la que coge el valor total de los registros de la página.

Listado completo de reservas (20) <a href="#">¡Mira esto!</a>		
Mesa	PAX	Nombre

- navTop: componente creado que contiene el boton de agregar de la página para arreglar cualquier registro., por otro lado también contiene la barra de búsqueda.
- Tickets-resume: componente que se encarga de recoger el resume de los tickets. Esta llama al método asíncrono getInvoices para obtener el total de los tickets y asigna el response data a un data de vue.
- TableReserva: Este componente es el eje central del Home,este tiene como parámetros:
  1. reservas: es un props de este componente, acepta un array de reservas.
  2. @updateReserv: emite el id de la reserva
  3. @upadeList: método que emite un true para informar al padre que debe actualizar las reservas y el total tras un cambio
  4. @resetID: método que emite un String vacio, se utiliza para resetear el id de la reserva en el padre (Home)

- Modal: Este componente pertenece al paquete de Jetstream, se ha utilizado para el modal del proyecto, necesita:
  - :show: props true o false que cumpla la función para cerrar y abrir la ventana
  - @close: para su cierre llama al método changeModal ya nombrado anteriormente.

## Productos y Drinks

Estos componentes utilizan como platilla app-layout y a su vez utiliza una serie de componentes, métodos, data y watchers, estos dos son prácticamente iguales cambiando ciertos aspectos puntuales, por lo que se van a nombrar sus métodos y data en el mismo apartado.

**Data**(las data de Drinks son iguales pero llamándose drink por tanto se van definir aquí)

- showModal: default false, mismo uso que el nombrado en el componente anterior
- showModalMod: default false, mismo uso que el nombrado en el componente anterior
- image: default null, data usada para recoger si subimos una imagen al servidor
- productName: default String vacío, utilizada en el v-modal del input name producto
- productAmount: default String vacío, utilizada en el v-modal del input amount producto
- descriptionProduct: default String vacío, utilizada en el v-modal del textarea description producto
- imageDefault: default false, variable para usar foto por defecto cargada en el servidor para producto o bebida
- url: default null, usada para crear una url object con la foto.
- productCategory: default 1, utilizada en el v-modal del input category producto,
- productos: default array vacío, almacenará todos los registros de los productos para más tarde listarlos
- productID: default String vacío,
- produc: default array vacío, almacenara el producto que se muestra en la modal, se rellenará desde el componente hijo mediante un \$emit
- change: default false, variable que cambiara de valor cuando se actualice un producto desde el hijo, esta se encuentra en un watch, con lo cual tras ponerse en true llamara a getProducts y se pondrá en false.
- deletePro: default false, variable que se utiliza para saber si se ha borrado un producto, está se encuentra en un watch y cuando esté en true actualizara mediante getProducts y la cambiara a false

## Métodos

- selectImage: selecciona el elemento con id image
- filechange(evento): crea una url object con el file
- resetData: método encargado de resetear los datos de la modal de alta productos
- resetDataUp: método encargado de resetear los datos de la modal de modificación productos
- changeStateModal: método que cambia el valor a showModal
- changeStateModalMod: método que cambia el valor a showModalMod

## Asincronos

- createProduct: método de creación de producto mediante la url mostrada en la imagen
- getProducts: método de recogida del total de registros de producto
- updateProduct(id): método de update de producto mediante la url mostrada en la imagen
- getProduct(id): método que devuelve el producto por el id

```
//Products
Route::get('/products', [ProductController::class, 'index'])->name('products');
Route::get('/product/{id}', [ProductController::class, 'show'])->name('product');
Route::get('/products-list', [ProductController::class, 'getProducts'])->name('products-list');
Route::post('/create-product', [ProductController::class, 'create'])->name('create-product');
Route::put('/update-product/{id}', [ProductController::class, 'updateProduct'])->name('update-product');
Route::delete('/delete-product/{id}', [ProductController::class, 'destroy'])->name('delete-product');
```

## Componentes

Los componentes nombrados anteriormente no se van a volver a explicar, pero si se van a nombrar para tener en cuenta que se usan en este componente también.

- navSide
- headerNav
- navTop
- modal
- cardData: este componente es la card completa del producto tiene como params:
  - @changeModal: método que emite el cambio del data showModal para abrir la modal
  - @modalData: emite el id de las bebidas o de los productos para pasarlo por la modal que se encuentra en el padre(Products/Drinks)
  - data: props que contiene el producto o la bebida
  - categories: props que contiene todas las categorías de los productos o las bebidas
  - @emitDelete: emite el valor de si se ha borrado se recarguen los datos en el padre.



## Platos/Usuarios (Dishes/users)

Estos componentes utilizan como plantilla app-layout y a su vez utiliza una serie de componentes, métodos, data y watchers, estos dos son prácticamente iguales cambiando ciertos aspectos puntuales, por lo que se van a nombrar sus métodos y data en el mismo apartado.

### Data

- deleteDish: default String vacío, controlará si un plato ha sido borrado desde un watcher
- editDish: default String vacío, controlará si un plato ha sido editado desde un watcher
- dish: default array vacío, array que contiene un plato,
- showModalDish: default false, mismo uso que el nombrado en el componente anterior
- showModalDishMod: default false, mismo uso que el nombrado en el componente anterior
- showDele: default false, variable que se va a encargar de la modal de delete
- dato: default array vacío, se va a llenar con el registro del plato.
- borrado: default false, va a estar desde el watch observando si hay algún cambio para recargar variables.
- change: default false, va a estar desde el watch observando si hay algún cambio para recargar variables.
- dishID: default String vacío, se va a rellenar con los datos del id del plato
- type: default 'Plato', indica el tipo de registro

### Métodos (los métodos nombrados con anterioridad nos e van a volver a explicar)

- changeStateModal
- changeStateModalMod
- resetDataUp

Asincronos (en este caso volvemos a la misma estructura anterior, por tanto se van a listar los métodos sin explicarlos)

- sendData
- createDish
- updateDish
- getDish

```
//Dishes
Route::get('/dishes', [DishController::class, 'index'])->name('dishes');
Route::get('/dishes-list', [DishController::class, 'getDishes'])->name('dishes-list');
Route::post('/create-dish', [DishController::class, 'store'])->name('create-dish');
Route::get('/dish/{id}', [DishController::class, 'show'])->name('dish');
Route::put('/update-dish/{id}', [DishController::class, 'updateDish'])->name('update-dish');
Route::delete('/delete-dish/{id}', [DishController::class, 'destroy'])->name('delete-dish');
```

### Componentes

- navSide
- headerNav
- navTop
- tableDish (este tiene los mismos params que en los casos anteriores.)
- modal
- modalAlert: este componente es una modal creada de alerta para cuando vamos a borrar un plato, tiene como parámetros:
  - data: registro a borrar
  - @changeStatusDelete: nos avisa de si se ha borrado el registro
  - showDelete: con esta data abrimos y cerramos la modal
  - @changeModalStatus: cambia el valor de la data showDelete

### Comandas(Orders)

Este componente utiliza como plantilla app-layout y a su vez utiliza una serie de componentes, métodos, data y watchers:

#### Data

- item: default array vacío, este va a recibir el ítem seleccionado
- mesa: default String vacío, este va a recibir el número de la mesa
- incremento: default false, esta data nos va a permitir incrementar el número de unidades
- resetProp: default false, esta variable nos va a resetear la data.

### Components

- navSide
- headerNav
- tableComanda: este componente es uno de los dos grandes bloques de la Comanda, este tiene el contenido de toda la tabla en la que podemos seleccionar productos mesas, y añadir cantidades.

#### Data

- dishes: array vacío, recibirá todos los registros de los platos

- tables array vacio, recibirá todos los registros de las mesas
- drinks: array vacio, recibirá todos los registros de las bebidas
- ticket: array vacio, recibirá el ticket
- selecTable: default false, variable que define si una mesa está ocupada

- select: default false, variable que define si una mesa está ocupada
- ocultoEn: default String vacio, se usa para ocultar las partes de la tabla
- ocultoSe: "
- ocultoPo: "
- ocultoOt: "
- ocultoSel: "
- ocultoBe: "

### Métodos

- closed: cambia el estado de la mesa a close
- showModal: emite evento para informar de la visión de la modal
- showModalDel emite evento para informar de la visión de la modal
- noSelection: deselectiona la mesa
- selection: selecciona la mesa

### Asincronos

- getDishes
- getTables
- getDrinks

Estos tres métodos sirven para recoger los registros de estos objetos.

- Tickets: este es el otro componente que tiene el grosor de este, este es el ticket lateral en el que se agregan los productos.

### Data

- NUM\_RESULTS: 8, esta variable se usa para la paginación 8 resultados
- pag: 1, indica el número de la pagina en la que estamos
- array: default this.items, contiene los ítems que se van añadiendo
- elements: default array vacio, usada para añadir los precios de los ítems

- numTicket: String vacío, se le añadirá el número de ticket
- showModal: default false,
- total: default 0, contendrá el total de la suma de los ítems

#### Métodos

- pageChange: asigna el número de página en el momento
- changeModal

#### Asíncronos

- createTicket(items,table,total,num): crea el ticket a través de este método
- getNumTicket: coge el último número del ticket y lo asigna al actual +1.

Este sería el proceso por el que pasan las distintas vistas y flujos. Para todo el contenido se ha tenido en cuenta el diseño responsive mediante media queries con tailwind, se ha añadido elementos multimedia (audio y video), canvas, y elementos svg, tanto proporcionados por páginas como por creación propia, en este caso este sería el logo de nuestra aplicación TastyKit.

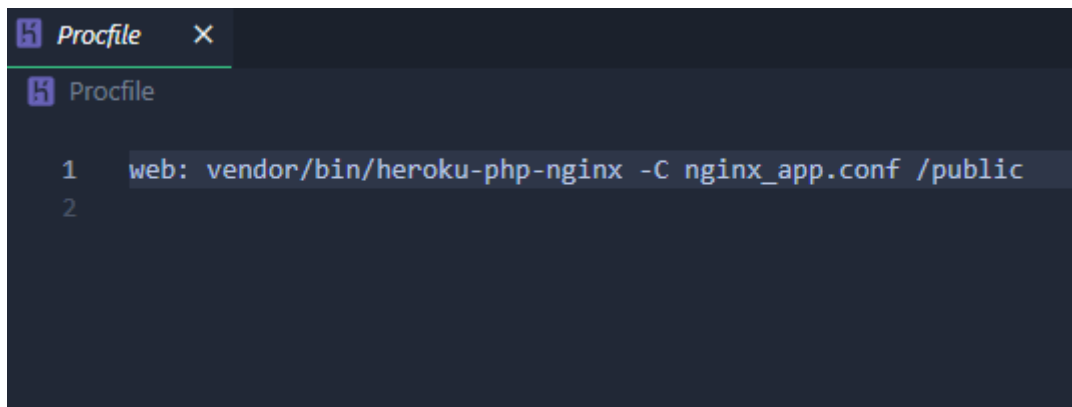
## 5-Despliegue

El despliegue de la aplicación se ha realizado en la plataforma Heroku. Heroku permite el despliegue en la nube desde diferentes lenguajes de programación. Para ello, primero hemos descargado Heroku CLI. Este CLI instala Heroku y permite loguearte y realizar gestiones desde la línea de comandos.

### Configuración Laravel

El siguiente paso ha sido crear dos archivos, uno llamado Procfile y otro llamado nginx\_app.conf. El archivo Procfile le indica a Heroku la configuración, esta puede ser apache2, que viene por defecto, o nginx, de la que hemos hecho uso. El segundo archivo, por su parte, sirve para personalizar la configuración de Nginx.

## Procfile



```
1 web: vendor/bin/heroku-php-nginx -C nginx_app.conf /public
2
```

## nginx\_app.conf



```
1 location / {
2     # try to serve file directly, fallback to rewrite
3     try_files $uri @rewriteapp;
4 }
5
6 location @rewriteapp {
7     # rewrite all to index.php
8     rewrite ^(.*)$ /index.php/$1 last;
9 }
10
11 location ~ ^/index\.php(/|$) {
12     try_files @heroku-fcgi @heroku-fcgi;
13     # ensure that /index.php isn't accessible directly, but only through a rewrite
14     internal;
15 }
```

El siguiente paso es loguearnos en Heroku. Para ello, escribimos en la línea de comandos: `heroku login`. Este comando hará que se abra una ventana en nuestro navegador en la cual, si pulsamos sobre el boton Login, podremos autenticarnos. Una vez logueados, mediante el comando `heroku create` podemos crear nuestra app, en este caso la hemos llamado `tastykit`.















Para la configuración de Laravel iremos, dentro de la plataforma Heroku, a `Settings>Config Vars`. Allí debemos introducir las variables que encontramos en nuestro archivo `.env` de nuestro proyecto Laravel.

## Configuración Node.js

**Config Vars**

Config vars change the way your app behaves. In addition to creating your own, some addons come with their own.

**Config Vars** Hide Config Vars

APP_DEBUG	true	 
APP_ENV	production	 
APP_KEY	base64:73uEjGxj+LLq0cjU8WRr8ZKov11i19Xgo2	 
APP_NAME	tastykit	 
APP_URL	https://tastykit.herokuapp.com/	 
DATABASE_URL	postgres://wydtzifqujsjtw:117061ffa70f710	 
KEY	VALUE	 

KEY

VALUE

Add

Por su parte, para configurar Node.js se ha ejecutado el comando: `heroku buildpacks:add heroku/nodejs`. Este comando será instalado en despliegue pero no en devDependencies. Para solventar esto debemos ejecutar: `heroku config:set NPM_CONFIG_PRODUCTION=false`.

Por último, añadiremos en el package.json la última línea con `postinstall`:

```
"scripts": {
  "dev": "npm run development",
  "development": "mix",
  "watch": "mix watch",
  "watch-poll": "mix watch -- --watch-options-poll=1000",
  "hot": "mix watch --hot",
  "prod": "npm run production",
  "production": "mix --production"
  "postinstall": "npm run prod"
```

## Control de versiones

Para el control de versiones se ha utilizado git, haciendo uso de los siguientes comandos:

`git branch -M "rama"`

`git . add`

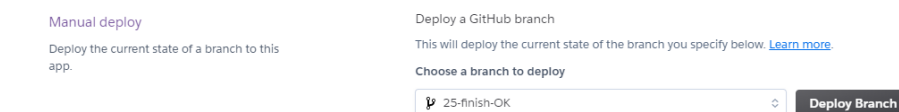
`git commit -m "Comentario"`

`git push origin rama`

Una vez, realizado push a nuestro repositorio de github podemos vincularlo con Heroku. Para ello, en la pestaña Deploy de la plataforma Heroku podemos realizar la conexión:

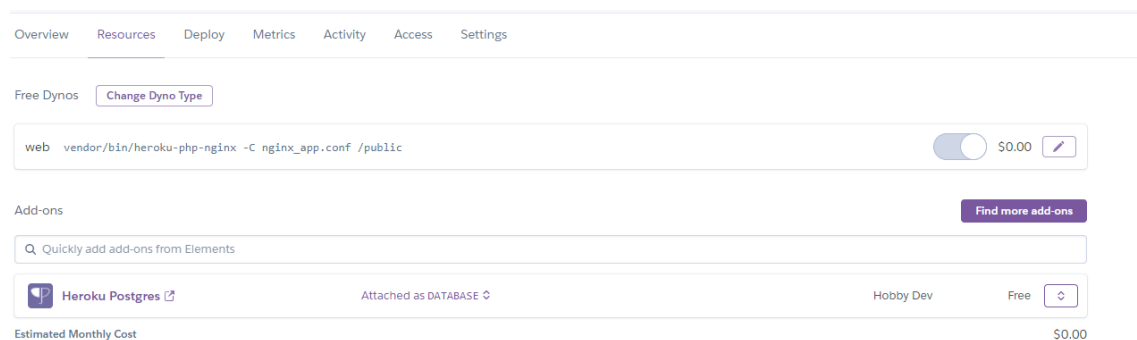


Una vez conectados, podemos elegir la rama a la que realizar push:



## Configuración de la base de datos

Finalizados estos pasos, quedaría configurar la base de datos. En la pestaña Resources de la app de Heroku podemos encontrar diferentes recursos que instalar. Buscamos Heroku Postgre y pulsamos sobre install.



A continuación, si pulsamos sobre Heroku Postgre se nos abrirá una nueva ventana, dentro de ellas en la pestaña Settings pulsamos Database Credentials y se nos mostrará las credenciales de nuestro proyecto.

#### Database Credentials

Get credentials for manual connections to this database.

Please note that **these credentials are not permanent**.

Heroku rotates credentials periodically and updates applications where this database is attached.

Host	ec2-34-247-118-233.eu-west-1.compute.amazonaws.com
Database	d9e3797dma8eqa
User	wydtzifqujsjtw
Port	5432
Password	117061ffa70f71041ce7e2fc34e287132e5686e5af337966b189124e0ac4e222
URI	postgres://wydtzifqujsjtw:117061ffa70f71041ce7e2fc34e287132e5686e5af337966b189124e0ac4e222@ec2-34-247-118-233.eu-west-1.compute.amazonaws.com:5432/d9e3797dma8eqa
Heroku CLI	heroku pg:psql postgresql-silhouetted-74842 --app tastykit

Copiamos la URI y la pegamos en las variables de heroku, que configuramos anteriormente. Añadimos como valor la URI y como key DATABASE\_URL.

DATABASE\_URL

postgres://wydtzifqujsjtw:117061ffa70f71041ce7e2fc34e287132e5686e5af337966b189124e0ac4e222@ec2-34-247-118-233.eu-west-1.compute.amazonaws.com:5432/d9e3797dma8eqa

Ya estaría la base de datos configurada y podemos realizar los comandos de php artisan necesarios. Realizaremos el comando: php artisan migrate, para realizar las migraciones de nuestra base de datos.

Por último, en nuestro archivo AppServiceProvider.php que se encuentra en app>Providers en nuestro proyecto Laravel debemos completar la función boot con:

```
public function boot()
{
    if(config('app.env') === 'production') {
        \URL::forceScheme('https');
    }
}
```

Esto nos permitirá el uso de https. Con estos pasos ya tendríamos nuestra aplicación corriendo en Heroku. Se puede acceder a ella desde el siguiente enlace:  
<https://tastykit.herokuapp.com/>