APLICACIÓN PARA BARES Y RESTAURANTES

1. **Descripción del proyecto**
   1. **¿Qué es?**

Es un proyecto pensado para el sector de hostelería, concretamente para poder realizar un proceso normal de toma de pedido de un cliente en un restaurante, el proceso que se intenta emular en el software que se va a crear es el siguiente:

* El camarero se acerca a la mesa con una libreta de notas.
* El camarero le pregunta al cliente que es lo que va a pedir para comer y beber.
* El camarero escribe el pedido del cliente en su librera de notas.
* El camarero le pasa el pedido a la cocina para que preparen los platos de comida, y a la barra para que preparen las bebidas que el cliente pidió.
* El camarero se acerca a preguntar a los clientes si ya quieren pagar para pasarles el ticket con el precio total que costo todo su pedido.

Para que este proceso sea más rápido y eficiente, el software guardara el pedido de los clientes con el respectivo precio de cada producto que ha pedido para que a la hora de pagar y hacer el cálculo de la cuenta se pueda hacer de manera eficiente tanto como si paga en efectivo o con tarjeta.

* 1. **¿Por qué?**

Las razones por las cuales he decidido realizar este proyecto son las siguientes.

* He trabajado en el sector de hostelería por muchos años y se cómo se organiza un sistema en un entorno de restaurante, con mi software y mis bastos conocimientos del sector se que podre hacer mas eficiente y ordenado el trabajo de un camarero.
* A lo largo de mi paso por el sector de hostelería he trabajado con diferentes aplicaciones para poder realizar mi trabajo, y detecte muchos errores que se repetían con los diferentes softwares con los que trabajaba, siendo el más recurrente un error muy fatal, la persona que programaba el software no sabía nada de hostelería y ni siquiera se tomaba la molestia de pasar a preguntar como era la forma de trabajar en un restaurante.
* Realizo este proyecto porque creo que con mis conocimientos de software y del sector de hostelería podre realizar un buen trabajo debido a que además de realizar la parte de programación también tendré en todo momento un trato amable y directo con el cliente y los trabajadores del restaurante y corregir de manera rápida los posibles errores que vallan surgiendo durante la ejecución del programa.

1. **Proceso de la creación del software**
   1. **Ejemplo en el cual base mi proyecto**

En una ocasión tuve la oportunidad de trabajar con el sistema “Cuiner” y lo que mas me gusto de este sistema era su simplicidad y sobre todo su rapidez a la hora de navegar entre las diferentes páginas.



Esta es la página principal de la aplicación “Cuiner” se puede ver el nombre del camarero, el menú del restaurante, bebidas postres, comidas, etc.



Es muy fácil interactuar con este sistema, pero cuando yo lo utilicé daba problemas a la hora de separar productos del mismo tipo como en este caso el agua x3, otro problema que daba era que cuando me salía del rango de la wifi o el internet se venia abajo, el programa se bloqueaba y a veces le daba varios clicks sin querer a un botón, y cuando se volvía a conectar a internet todos los clicks que había dado se ejecutaban.



Por último, esta es la pantalla de cobro, en esta parte no encontré ningún problema, todo esta muy claro, si pago con efectivo se ven los billetes, si pago con tarjeta también, y están debidamente nombrados las diferentes entidades que existen cuando se paga con checkes restaurantes.



Cuando se paga por ejemplo con visa, aparecía la cantidad en un cuadro rojo, si quería corregir el tipo de pago que había hecho simplemente tenía que darle click al cuadro rojo y el tipo de pago desaparecía.

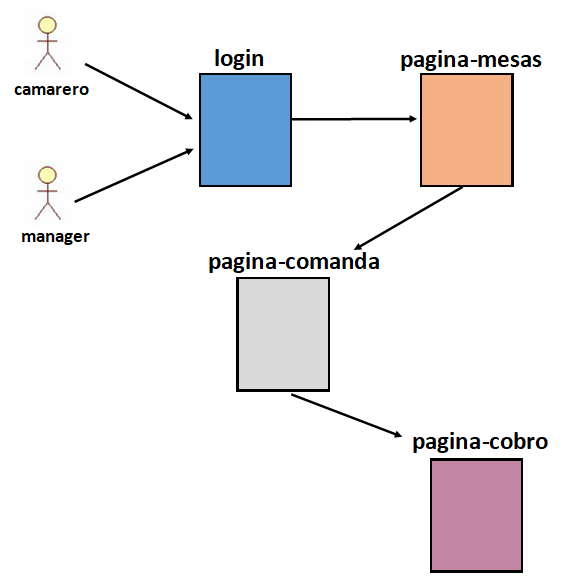
* 1. **Esquema de las páginas por las cuales se va a interactuar**

Pagina donde los usuarios iniciaran sesión, además también existirá un usuario llamado “admin@admin” el cual servirá para hacer el CRUD de los usuarios.

Existirán dos tipos de usuarios:

1. **Camarero**
2. **Manager. -** tendrá privilegios como, crear usuarios, consultar usuarios, o modificar datos de los usuarios.

En esta página es donde estarán las mesas divididas en categorías, y cada mesa tendrá su debida numeración.

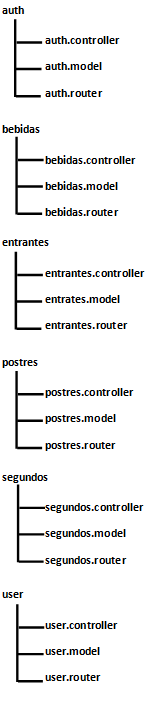


En la página comanda estarán los platos de comida con sus respectivos precios.

En la página de cobro se visualizarán las diferentes formas de cobrar como efectivo y/o tarjeta.

* 1. **Esquema y estructura del código del Back-End y Front-End**

1. Back-End



En esta parte ira el código que me ayudara a autentificar a los usuarios usando la API de mongoDB.

En esta parte ira el código que me ayudara a hacer el CRUD a las bebidas usando la API de mongoDB.

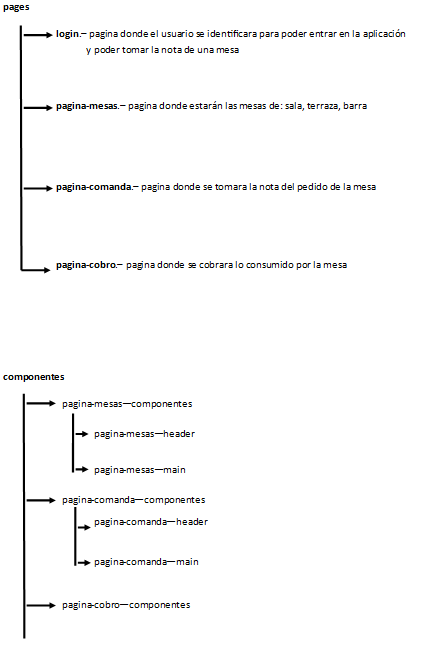
En esta parte ira el código que me ayudara a hacer el CRUD a los entrantes usando la API de mongoDB.

En esta parte ira el código que me ayudara a hacer el CRUD a los postres usando la API de mongoDB

En esta parte ira el código que me ayudara a hacer el CRUD a los segundos usando la API de mongoDB.

En esta parte ira el código que me ayudara a hacer el CRUD a los usuarios usando la API de mongoDB.

1. Front-End

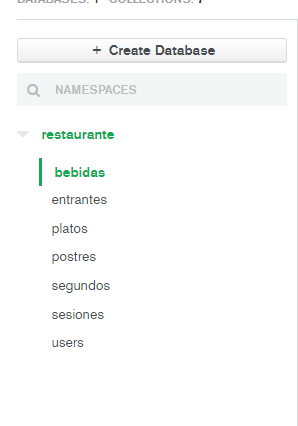


En una carpeta llamada “pages” administrare mis cuatro páginas principales.

En una carpeta llamada “componentes” gestionaré los componentes de mis cuatro páginas principales.

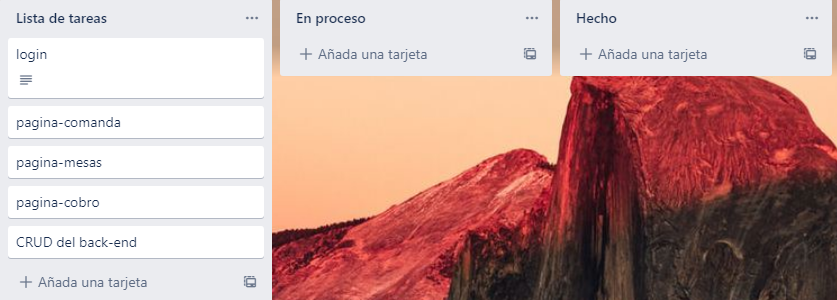
* 1. **API del mongoDB**

Para crear mi propia API usare el mongoDB, en el crearé alrededor de 5 entidades, desde el front end accederé a la base de datos “restaurante” para hacer el CRUD.



* 1. **Método de trabajo**

El método de trabajo que utilizare será el Kanban debido a su simplicidad.



Dividiré el trabajo en dos grupos:

* Back-end
* Front-end

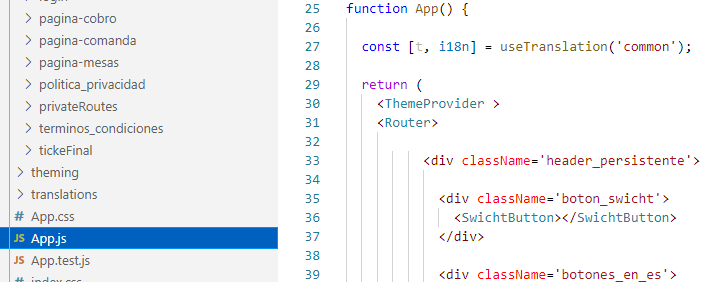
A su vez el Front-end será dividido en 4 partes, uno por cada página que formará parte del software:

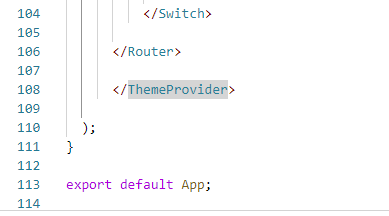
* Página-comanda
* Página-mesas
* Página-cobro
* login
  1. **Tecnologías usadas**

1. Theming. -

**MOTIVO. -** Aplicar un tono uniforme a todas las páginas.

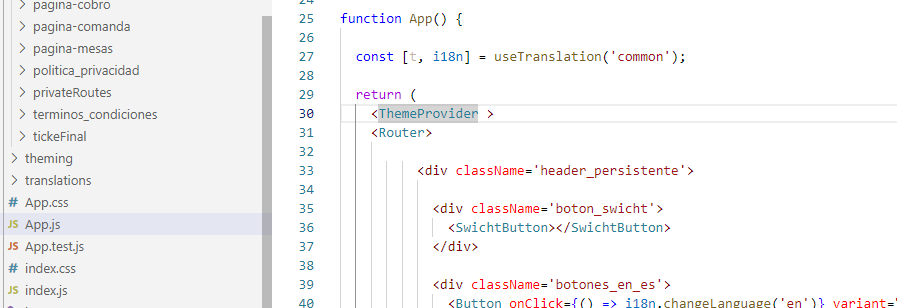
**PROCESO. -** En el “App.js” se encontrarán todas las paginas por donde se va a interactuar, por lo tanto, la etiqueta “ThemeProvider” va a envolver todo lo que hay en el “App.js”.

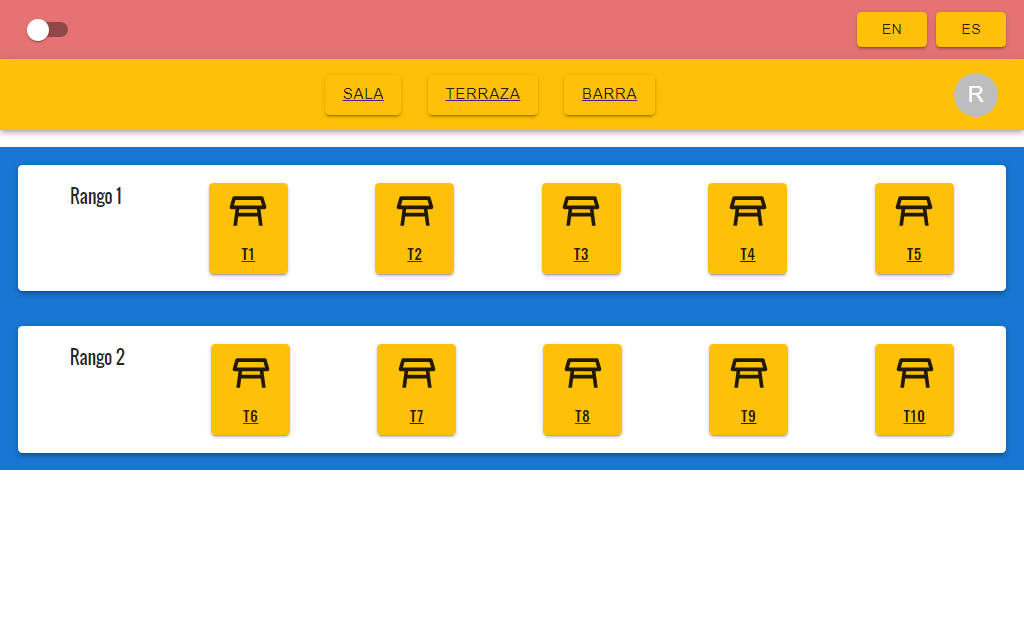




**PROBLEMA. -** Agregar un Switch general para todas las páginas para poder cambiar de un modo claro a un modo oscuro y viceversa.

**SOLUCION. –** Crear un componente en el App, ya que el App tiene englobado a todas las páginas, donde se encuentre el switch para poder cambiar de modos.

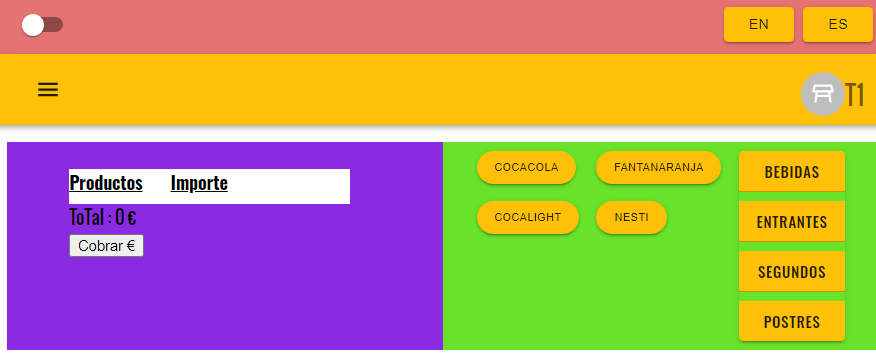






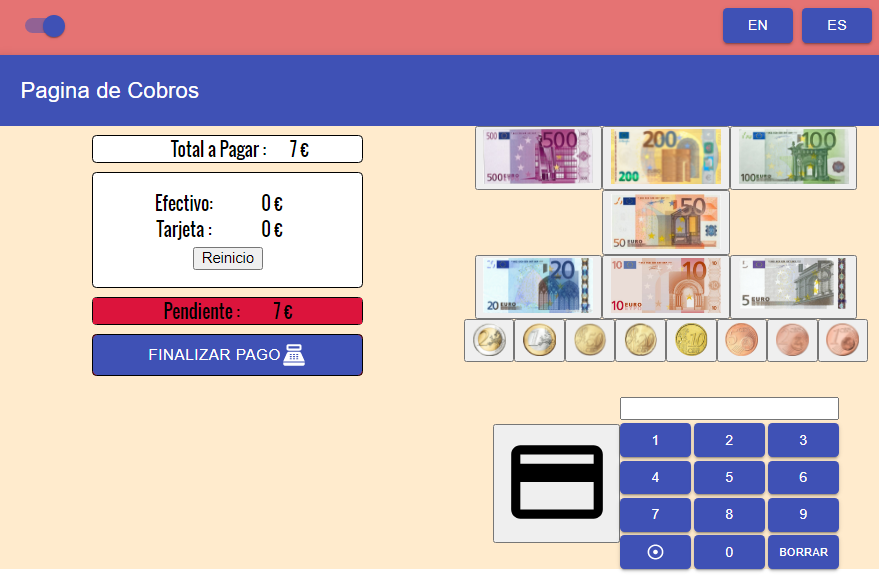
Claro

Oscuro



Oscuro

Claro



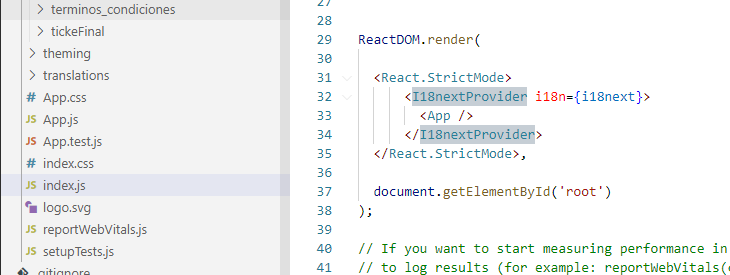
Oscuro

Claro

1. I18nextProvider. -

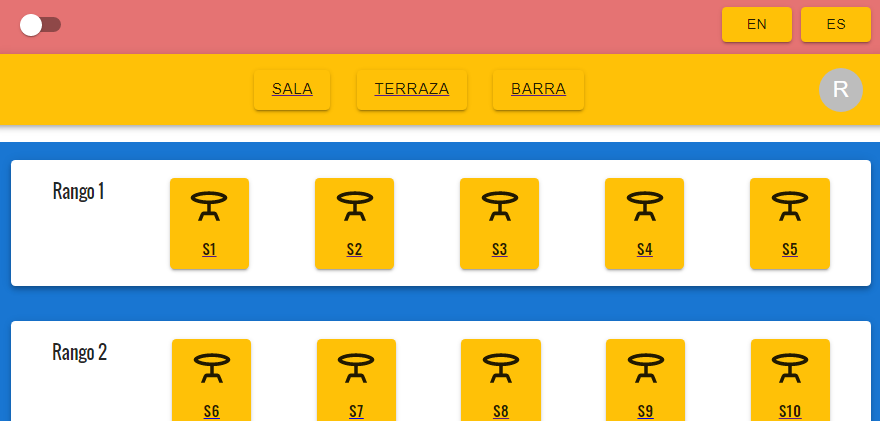
**MOTIVO. -** Aplicar un cambio de idioma de inglés-español o español-inglés a toda la aplicación.

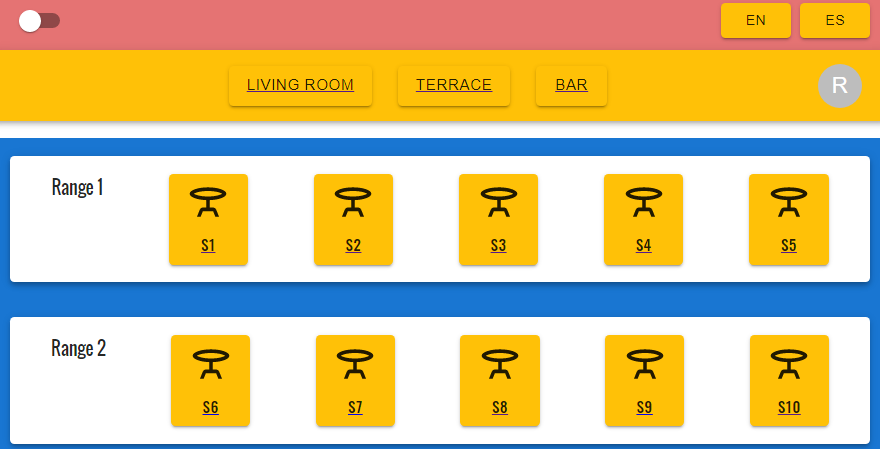
**PROCESO. -** En el “index.js” englobo la etiqueta App con el I18nextProvider, para que de esa manera a la hora de cambiar el idioma afecte a toda la aplicación.



**PROBLEMA. -** Agregar dos botones, uno “ES” para cambiar el idioma a español y uno “EN” para cambiar el idioma a inglés, que se pueda utilizar en toda la aplicación.

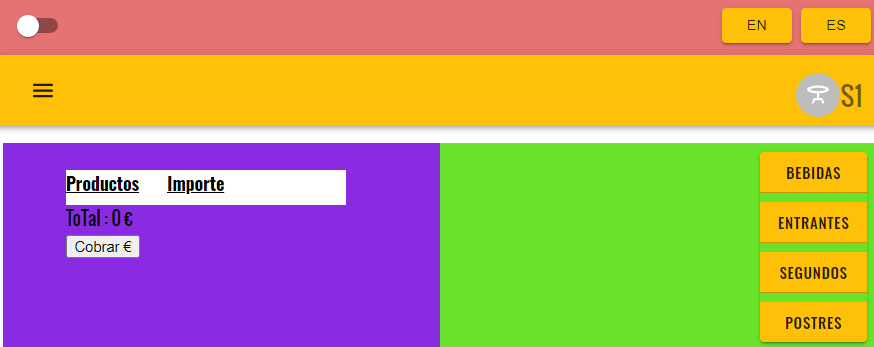
**SOLUCION. –** Crear un componente en el App, ya que el App tiene englobado a todas las páginas, donde se encuentren los botones, para que de esa forma puedan ser utilizados en toda la aplicación

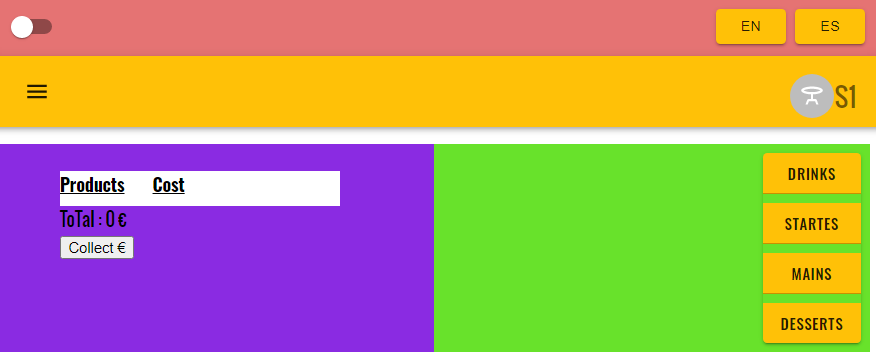




EN

ES





EN

ES



EN

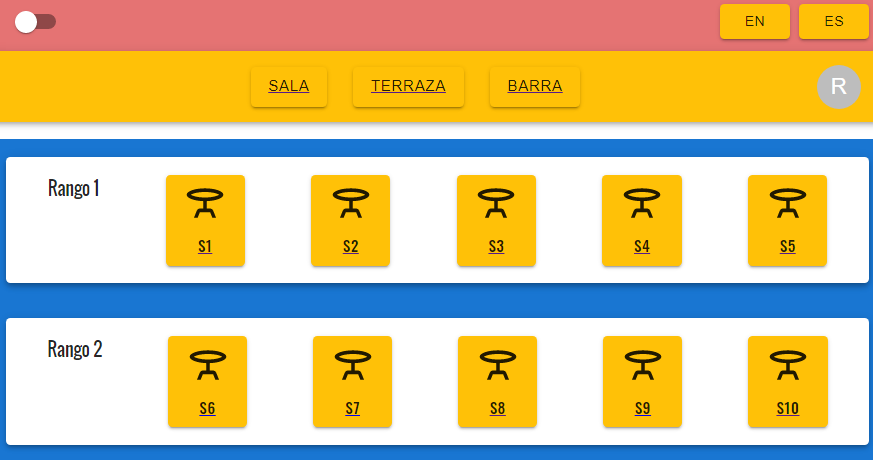
ES

1. useHistory. -

**MOTIVO. –** Navegar entre las diferentes páginas.

**PROCESO. –** Definir el “history.push” en las páginas donde tenga que redirigirme a otras después de ciertas acciones, como por ejemplo después de identificarme en la página login debo dirigirme a la página de mesas.





1. Autentificación de usuarios. -

**MOTIVO. –** Hay que identificar a los usuarios y contrastarlos con la API de mongoDB.

**PROCESO. –** Definir el “method”, “headers” y el fetch en el código de la página login.

En method, elijo el tipo de CRUD que quiero utilizar, el este caso “POST”, en headers defino el tipo de dato “json”.



En el fetch accedo al router específicamente a la parte donde estoy realizando la autentificación.

**PROBLEMA. –** Usar el puerto 3000 daba conflictos.

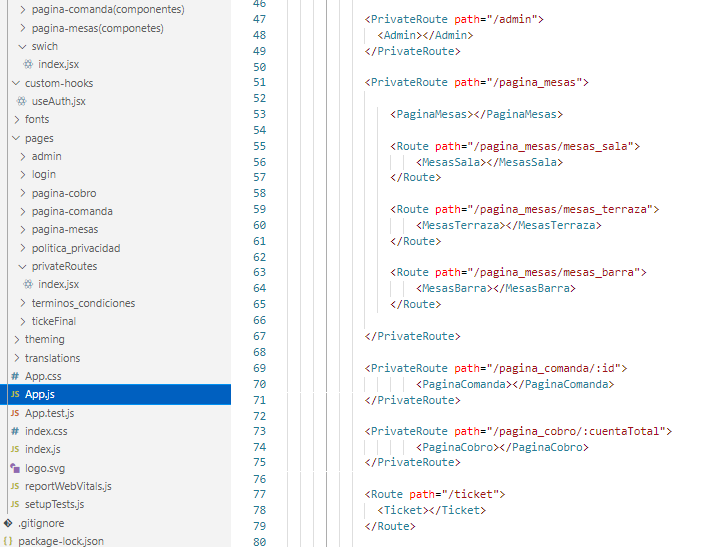
**SOLUCION. –** Utilizar los puertos arriba de 3000.

1. Private Routes. -

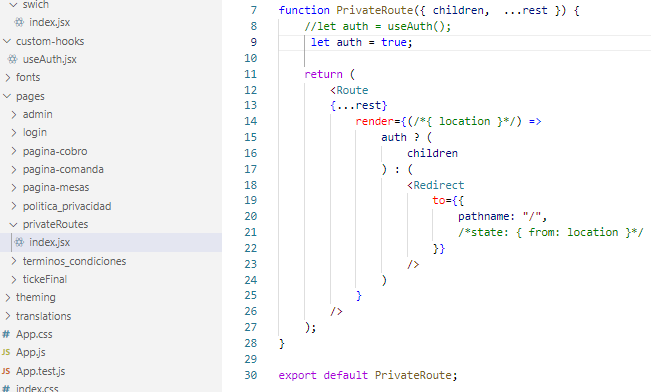
**MOTIVO. –** Poder navegar de manera segura a través de las diferentes páginas de la aplicación.

**PROCESO. –** Definir en el “App.js” todas las páginas por las que se va a navegar, y envolverlas en la etiqueta “PrivateRoute”.

Por otra parte, definir un componente “privateRoutes” donde se definirá una comprobación para saber si el usuario esta identificado y empezar a navegar entre las páginas o si no lo está que se redirija a la página raíz “/”.



..rest se utiliza para que las rutas privadas acepten los query params.



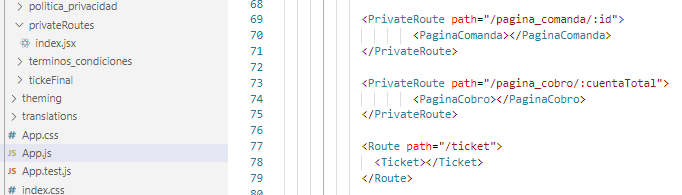
1. Query params. -

**MOTIVO. –** El número de las mesas es necesario saberlo, y para ello se usará los query params para que al pasar a la pagina donde se empezara a tomar nota del pedido, también se quede grabado el número de la mesa.

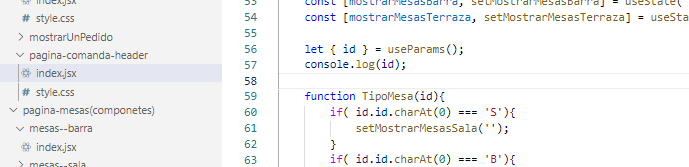
**PROCESO. –** Definir en el “App.js” el código necesario para utilizar los query params, es decir escribir los “ : ” dentro del “PrivateRoute” en el atributo path.

En las páginas donde están los números de las mesas, en la etiqueta “Link”, en el atributo “to” escribir lo que el query params recibirá.

En el atributo “path” defino el “ : ” seguido de un nombre, en este caso es “id”.



Con el parámetro useParams, se puede capturar el query param del PrivateRoute.



En este caso con ayuda del useParams se podrá visualizar el número de la mesa de la página anterior.



En la página de las mesas se capturará el número de la mesa.



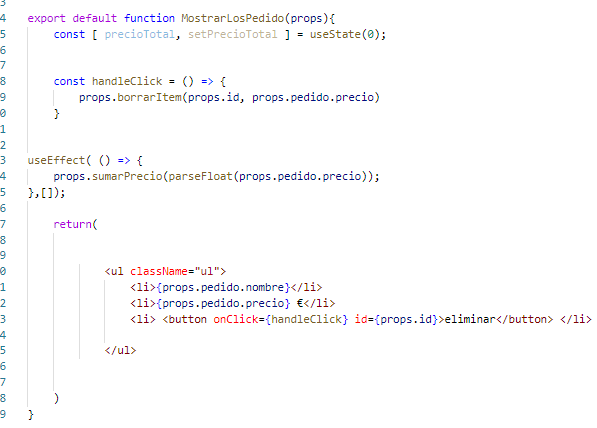
En el atributo “to” de la etiqueta Link capturo el número de la mesa para visualizarlo en la página siguiente.



1. Herencia Padre-Hijo Hijo-Padre. -

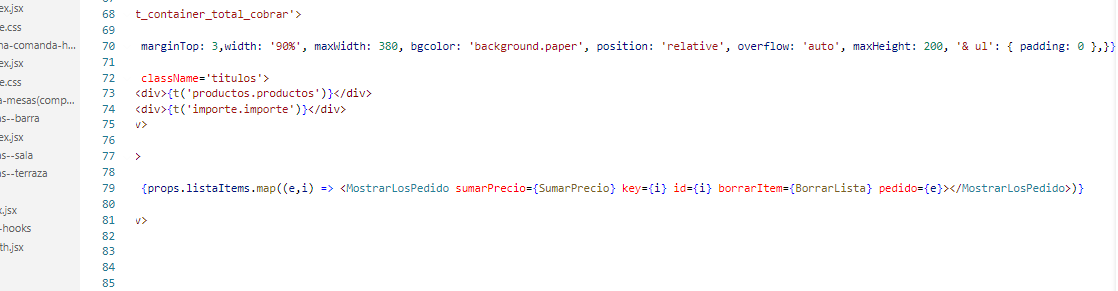
**MOTIVO. –** Muchos datos como el número de mesa, el precio de los productos o el listado de lo que pidió el cliente, será necesario pasar esos datos de una página a otra y para ello se utilizará la herencia del padre al hijo y del hijo al padre.

**PROCESO. –** Definir en un componente aparte los props, luego en las páginas principales darle valor a esos props como atributo de las funciones.



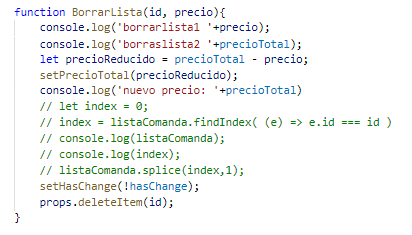
Es este componente se definen los props.

Es este otro componente le doy valor a los props

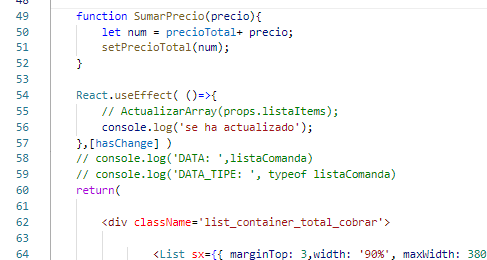


**PROBLEMA. –** Algunos valores que se pasaban por props, era necesario que se le sumaran o restaran valores, en muchos de los casos el valor no se podía modificar debido al mal uso del ciclo de vida del componente, en muchos de los casos el ciclo de vida se quedaba en el Mounting haciendo que los valores no se pudieran modificar.

**SOLUCION. –** Utilizar el useEffect para que el ciclo de vida del componente no se quede solo en la fase de montado.



Cuando la función note un cambio en el hasChange la función BorrarLista se ejecutará.

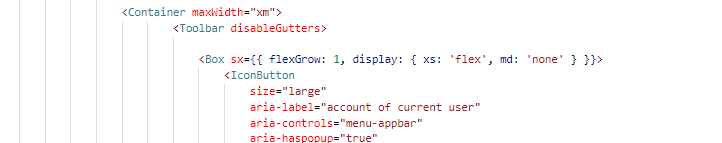


1. Responsive. -

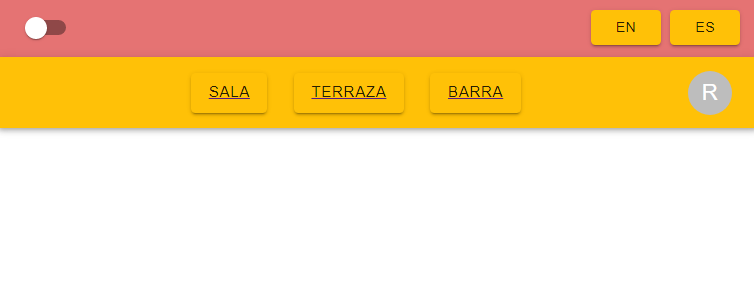
**MOTIVO. –** La página debe poder visualizarse en 3 tipos diferentes de pantallas, en pantallas de mesa, en tables y en móviles, para eso se usa el responsive para que a la hora de que la pantalla cambie de tamaño, el contenido de la página siga siendo correctamente visualizado.

**PROCESO. –** En el código descargado del mui.com, se puede modificar el responsive cambiando los valores del atributo display de manera bastante sencilla.

En el código restante usar el vh y el vw, para que el contenido se ajuste a al tamaño de la ventana.







Xs: flex

el menú hamburguesa aparece

Md : none

el menú hamburguesa desaparece

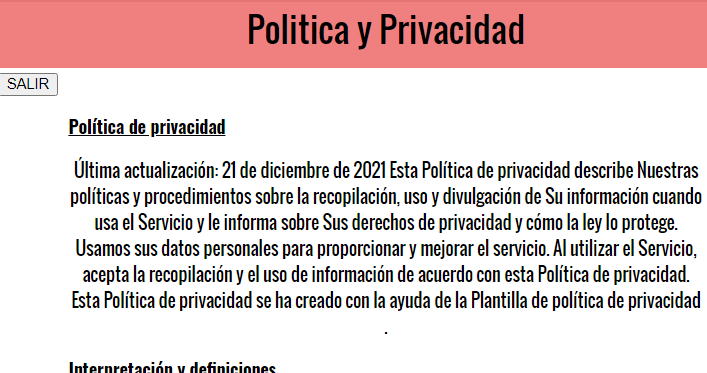
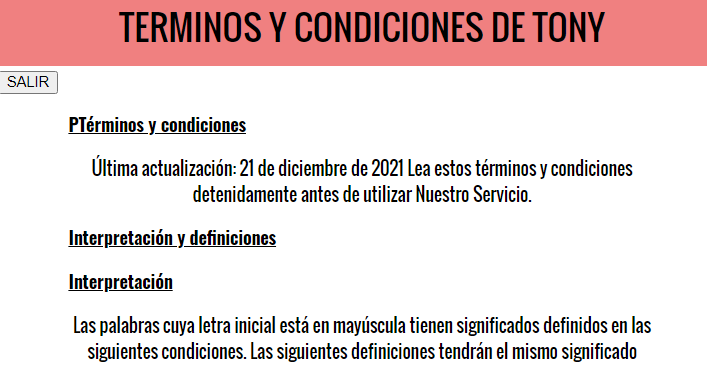
1. Políticas de privacidad. -

**MOTIVO. –** Cuando una web esta tratando con datos privados de usuarios, es necesario y por ley informarle al usuario como va a manejar dichos datos y determinar si el usuario esta de acuerdo o no con el manejo de sus datos privados.

**PROCESO.–** Acceder a la página <https://app.termsfeed.com/wizard/terms-conditions>, introducir los datos necesarios para que se genere automáticamente las políticas de privacidad.

Las políticas de privacidad deben estar visibles al principio, por lo tanto, se ubicarán en el login del programa.

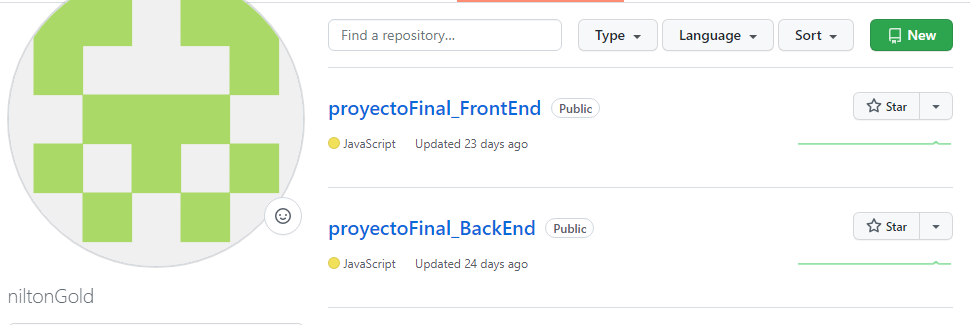


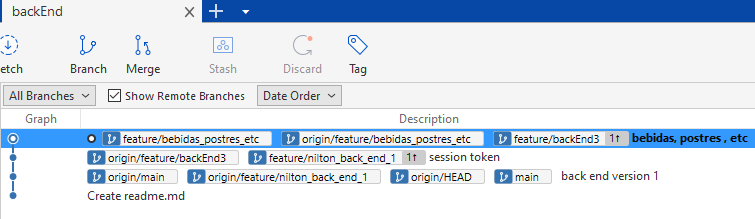


1. Git Hub. -

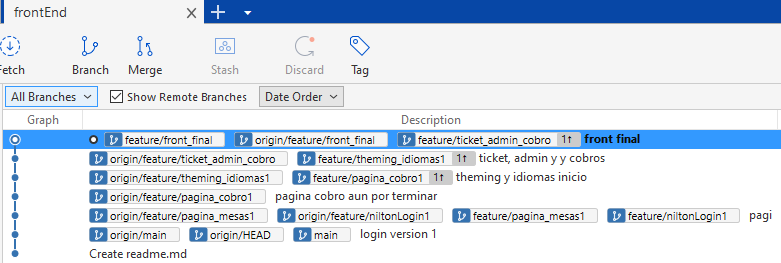
**MOTIVO. –** Para poder evitar la perdida del trabajo que se ha estado avanzando, se utiliza el github que es un sistema de almacenamiento online para poder guardar archivos y poder visualizar las versiones de los archivos subidos.

**PROCESO. –** Crear un repositorio para el front end y otro para el back end y luego subir ramas feature cada vez que se realice un cambio en el código.





Con el SourceTree se pueden gestionar de manera bastante visual las ramas de las diferentes versiones subidas al repositorio.



1. **Fases y tiempos**

Las fases y tiempos las dividí en 4, porque el software esta conformado por 4 páginas principales, escribí el código del front end y el back end casi a la par en cada fase de tiempo.

Debido a que el tiempo para entregar el proyecto eran 3 semanas, le dediqué aproximadamente 5 días a la creación de cada página.

Para el back end utilicé la arquitectura MVC dividido en 3 ficheros:

* Router.js.- Define las rutas.
* Controller.js .- Define los handlers de las rutas, aquí se llamaran a las funciones del model que gestionan los datos.
* Model.js .- Se gestionan los datos y las operaciones.

1. Login. -

* Front-end

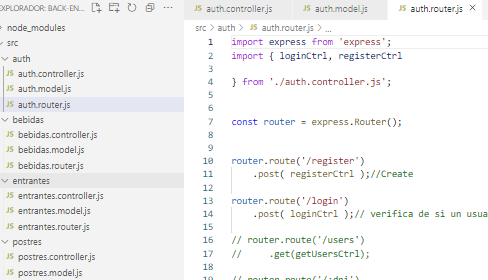


La página de login me servirá para poder navegar por las diferentes páginas con un usuario valido.

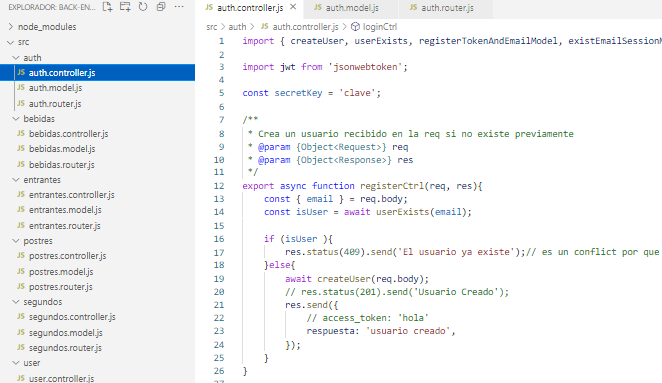


En el código del login, utilizo formularios, métodos POST con el fecht para llamar a las entidades del mongoDB, etc.

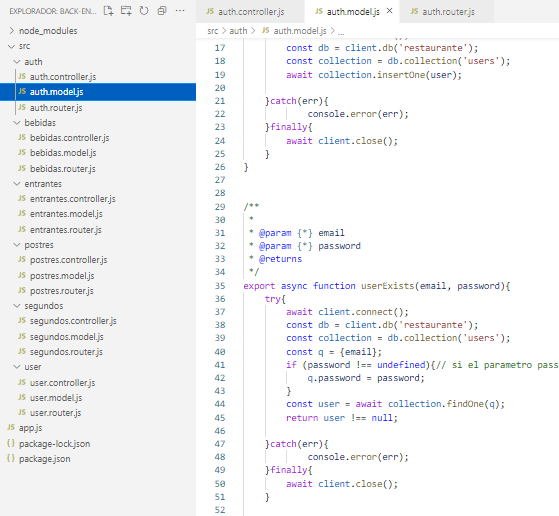
* Back-end



**Router.-** Defino las rutas para poder hacer login y poder registrarme.



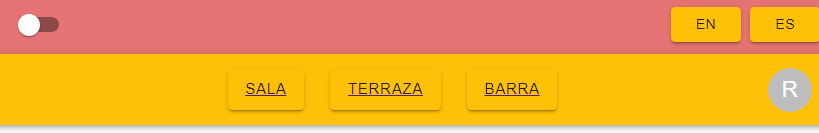
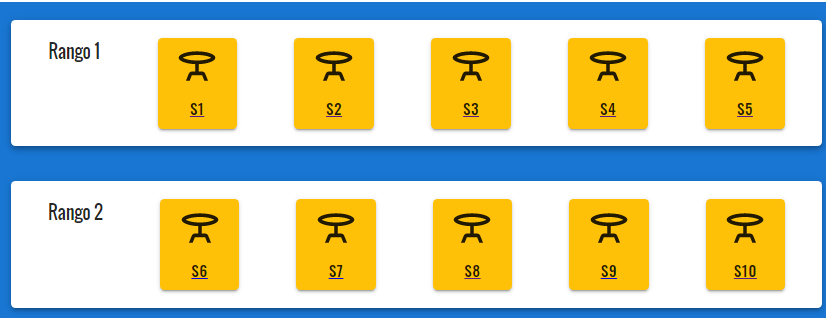
**Controller.-** Defino funciones para poder registrar y logear usuarios.



**Models.-** Defino las funciones para crear usuarios o si existe un usuario por email o nombre.

1. Pagina-mesas. –

* Front-end



La página de las mesas me servirá para poder elegir la mesa en donde se tomará nota del pedido del cliente.

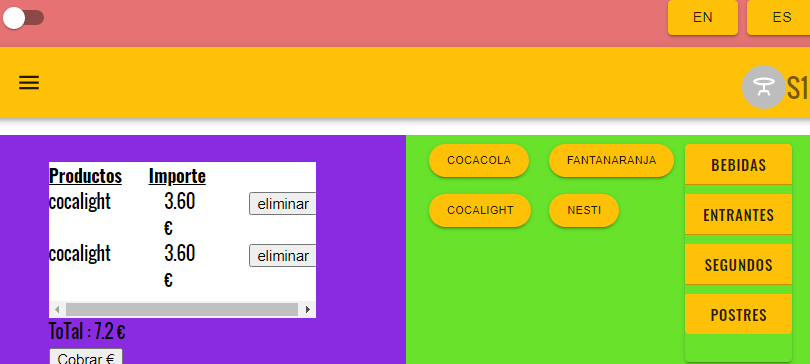


En el código del front end defino un header y al hacer click en sala, terraza o barra, aparecerán las respectivas paginas en la parte inferior.

* Back-end .- La pagina mesas no tiene un back end asociado.

1. Pagina-comanda

* Front-end

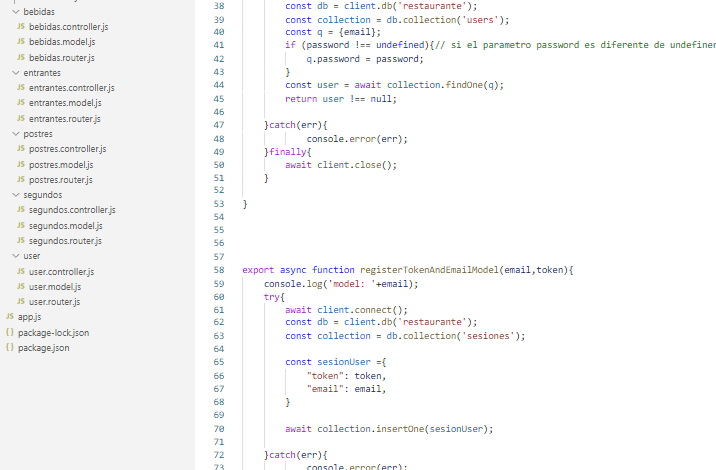


La página comanda tendrá dos partes, por un lado, estará una lista dividida en categorías de los platos que se sirve en el restaurante, esta lista cogerá datos del back end, por otro lado, se guardará un array con los platos seleccionados



En el código llamare a dos funciones, una que será la lista de las categorías de los platos y otra que será el array donde se guardaran los platos elegidos.

* Back-end



En el back end definiré las funciones que llamaran a los datos de las entidades almacenadas en el mongoDB.

1. Pagina-cobro. –

* Front-end



La página de cobro calculará los pagos que se realizan con pagos en efectivo o pagos con tarjeta.



El código contendrá, funciones que son las que gestionarán los pagos, las formas de pago, mediante los valores que las paso por props.

* Back-end.- Esta página no está relacionado con el back end.