**Carpeta Programación**

**Prof. Nicolas Barone**

GITHUB: https://github.com/Ddls4/IAE\_DB\_DDFJ



**Integrantes:**

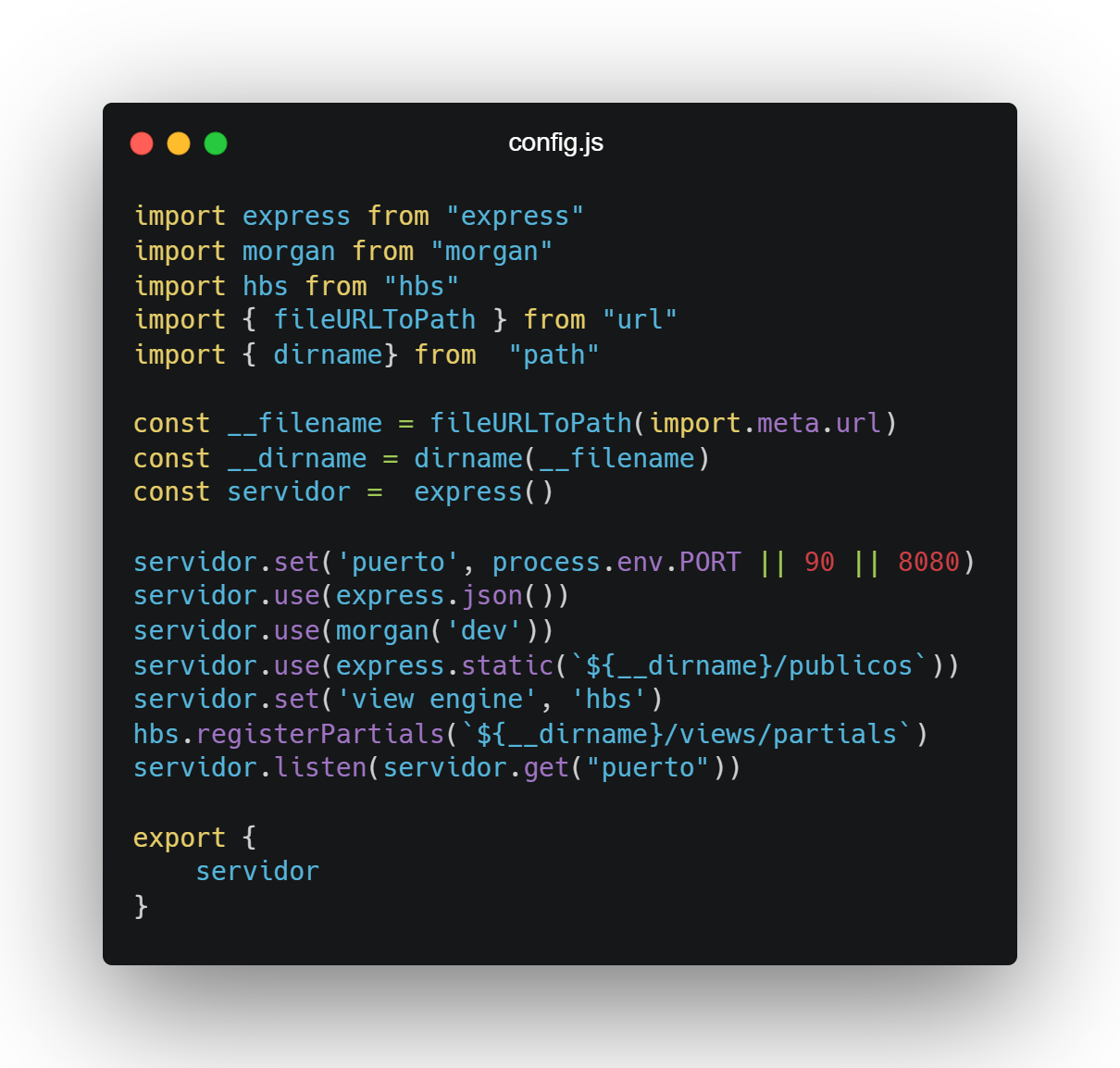
Diego Perez

Diego de los Santos

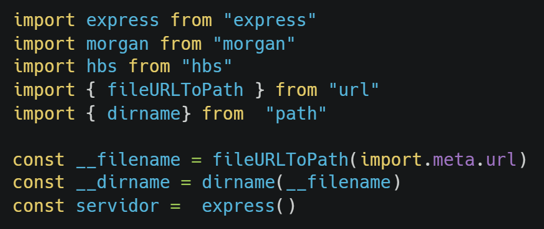
Facundo Sosa

Juanjo Caseres

Gastón Ferreira



Importamos las librerías que vamos a usar:



Express: La librería express sirve para poder crear la web y configurarla usando sus funciones get, post, put y delete.

Morgan: Esta librería nos permite tener opciones de “dev”, estas nos dan más información del estado de la página. Por ejemplo, cuando entras a una ruta.

HBS: Este es el motor de plantillas que vamos a usar. Este será el apartado visual



La primera línea es la que nos deja elegir que puerto vamos a usar, en nuestro caso es el puerto 90.

En esta sesión también establecemos los archivos “static” que serán los archivos públicos en donde estará el apartado vidual y algunas funciones para el funcionamiento de la página, no se mandan todas ya que algunas funciones pueden comprometer la seguridad de la misma y también le avisamos al servidor que nuestro motor de platillas es “HBS”.

Le pedimos a que express use los “. json ()” y que Morgan este activo.

Al final exportamos la variable servidor, esta era la nos permite usar express y está la mandaremos al index.js

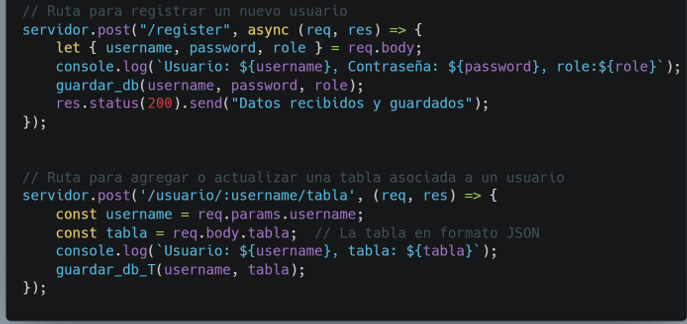


En el index.js lo primero que hacemos es traer de otras partes del código algunas cosas:

Estas son servidor, guardar\_db y guardar\_db\_T.



Esta parte lo que hace, es que cuando entremos a la ruta “/”, esto significa a la ruta principal, esta tiene que mostrar en pantalla “index.hbs”. Este es todo el apartado visual de la página.



La ruta “/register” la estamos usando para mandar los datos a la función de backend “guardar:db” esta será la que guarde el usuario en el MYSQL.

Mientras que “/usuario/:username/tabla” es para guardar las datos de las tablas a MYSQL.



Esto es lo que usa el backen para poder mandar a la base de datos los usuarios y los datos de los usuarios, estas toman los datos que les manda el fronten y con estos ejecutan guardar\_db y guardar\_db\_T. Estas no se encuentran en esta parte del código, para que este mas organizado.

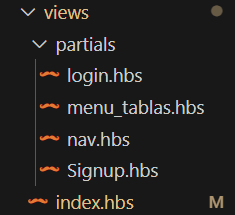
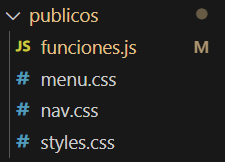
Estas funciones, se encargan de guardar a los usuarios y sus datos, necesitan la librería de mysql2 ya que esta es la que se encargar de transmitir la información del código a las tablas de mysql.

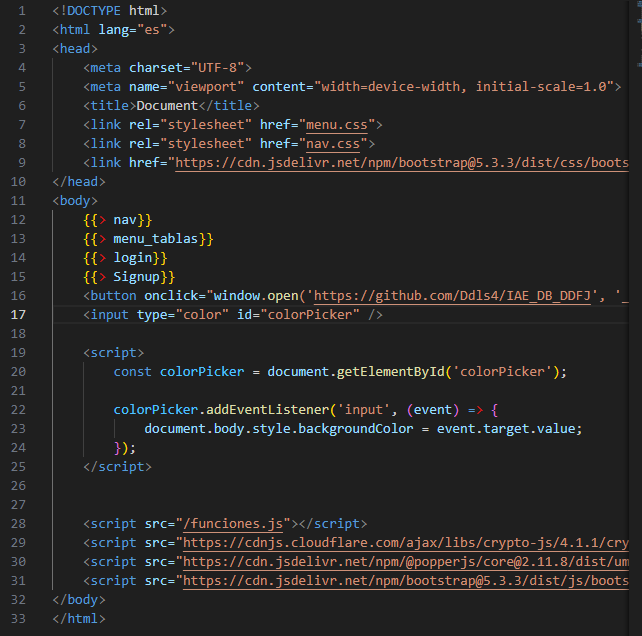
Las 2 funciones lo que hacen es recibir parámetros, ej. usuario y contraseña o usuario y tabla

Y estas se conecta y manda esta información con un query, este es muy importante ya que este es el que dice que se va hacer con esa información.

En guardar\_db el query lo que hace es ingresar un usuario y asignarle una contraseña para que este se pueda logear. Mientras que guardar\_db\_T su query lo que hace es buscar el nombre del usuario y este le guarda la información en Tabla\_Asociada esta va a estar en formato JSON.

El export es para que se puedan llamar las funciones desde el index.js.

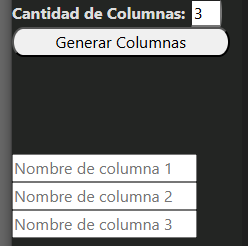


Estamos usando la versión 5.3.3 de bootstrap, popperjs y crypto-js, también importamos el CSS de menú.css y nav.css, Mientras que para el funcionamiento de la página los scripts son los de funciones.js.

El código se divide en 4 partíais para mantener el orden en el código estos son: nav, menu\_tablas, login y Signup.

En los archivos públicos encontramos 2 CSS estos son 1 para la barra superior “nav.css” y otro para la parte que genera las tablas y el menú el archivo “menú.css”

La función “generateColumnInputs” se activa al apretar click en el botón “Generar Columnas”.

Esta función toma el valor de cuantas columnas quieres y lo coloca en la variable “columnCount”.

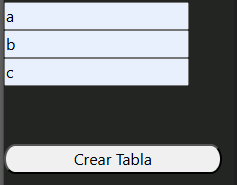
También toma el id de “columnNamesSection”

Este será el lugar donde se crearán los inputs para poner los nombres a las columnas.

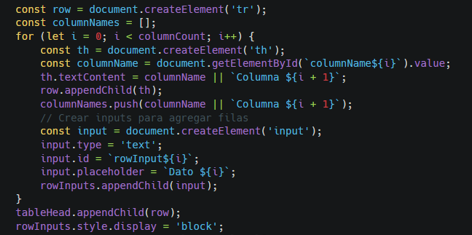
El for se ejecuta en relación a la cantidad de columnas elegida gracias a “i < columnCount”.

Estos inputs son de tipo texto, su placeholder va a ser “Nombre de columna” con el número correspondiente y le asigna un id.



Esta función se activa cuando apretamos en el botón “Crear Tabla” traemos los id de “columCont” ya que lo usaremos para ver cuántas veces se ejecuta el for. “tableHead” y “tableBody” lo usamos para saber dónde se crea la tabla y “rowInput” Es el lugar donde colocaremos los inputs que serán para agregar información a la tabla.

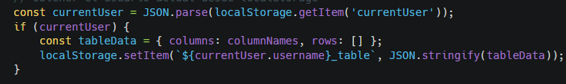
Limpiamos el encabezado y los inputs que agregaban información del anterior encabezado.



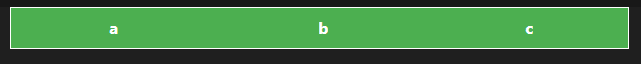
Creamos una “tr” que sería una fila nueva y creamos la variable “columnNames”, esta no servirá para mandarla a localsotrage.

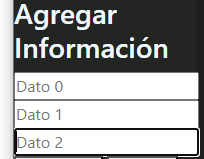
El for crea “th” que significa que es parte del encabezado, toma valores de “columnNames${i}” esto nos dará los nombre que nosotros elegimos en los inputs que creamos en “generateColumnInputs” y si no elegimos un nombre coloca “Columna ${i + 1}” y también colocamos con “push” el valor de “columnName” a “columnNanes” para tener todos los nombres en 1 variable para después subirla a localstorage.

La segunda parte del for es para agregar los inputs que agregaran la información a la tabla.



Esta parte es la que se encarga del localstorage, mira que usuario es el que se está usando el “currentUser” y crea la variable “tableData” que tiene la información del encabezado y la sube a “${currentUser.username} \_table” el “${currentUser.username}” significa toma el nombre de usuario del usuario actual.

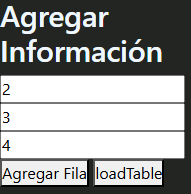
Resultado:



Info del localstorage:

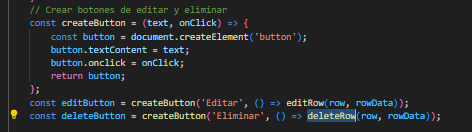




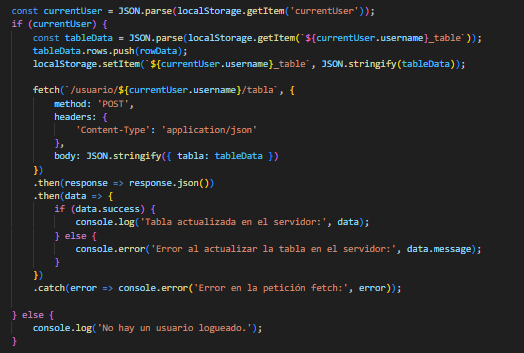
Esta función se activa al apretar el botón “Agregar Fila”. Usa los id de “columnCount” para saber cuántas veces se ejecuta el for, el “tablebody” para saber dónde se ejecutará y crea un “tr” que es una nueva fila. Crea la variable “rowData” que será la que usamos para agregar información al localstorage.

El for crea un “td” y toma los valores de los “inputs” y agrega la información del input a el “td” y este lo agrega a la “row” a la fila y para finalizar este mete la información a “rowData” para que este toda la información en 1 variable y mandarla a localstorage.

Por petición del profesor de “ingeniería de software” agregamos 2 botones editar y eliminar.



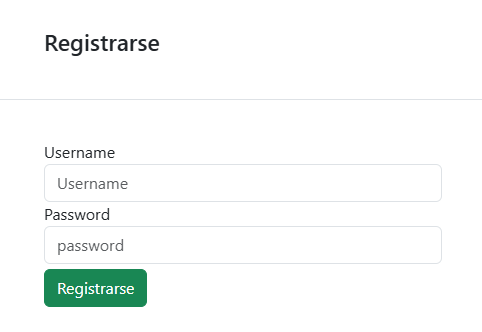
Esta parte de la función nos permite crear los botones, estos tienen 1 par por cada fila y esta se enlaza a la row y su rowData, para poder cambiar los datos de caga fila. El botón solo tiene las propiedades de su nombre el “textContent” y que ejecute 1 función al hacerle click (onclick).



Esta mira en que usuario esta y cuando ya está adentro toma el “${currentUser.username} \_table” que es donde guardamos la información del usuario y a esta le agregamos los datos de la tabla que almacenamos en “rowData” pero primero usamos “tableData.row.push(rowData)” y ahora si podemos mandarlo al localstorage ya que tiene el formato adecuado.

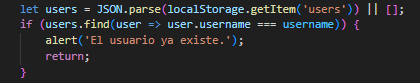
Después hacemos una petición fetch que será para mandar los datos del usuario al backend para que este lo mande a MYSQL.

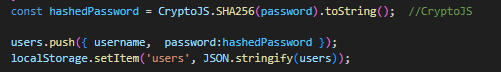
**Funciones para iniciar sesión y registrar usuario.**

****

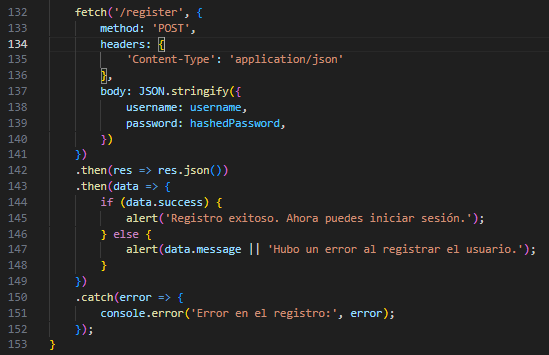
****

La función toma los datos de los inputs de “registerUsername” y “registerPassword” estos son los inputs del modal de registrase.

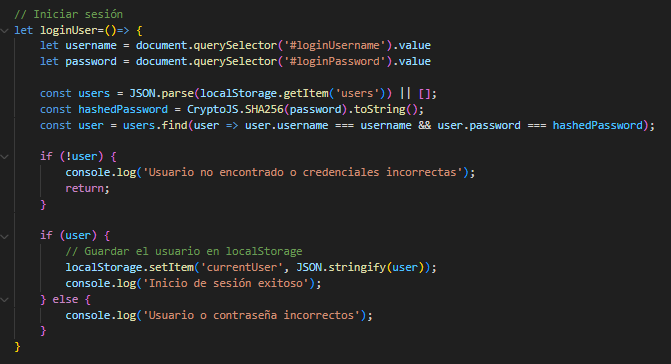


Con esto comprobamos que no existe otro usuario con el mismo nombre al comprobar esto cada usuario es único.

Antes de subir el usuario al localsotrage nosotros los hasheamos para evitar que una persona pueda ver las contraseñas y de otras personas. Una vez ya encriptadas lo subimos a la lista de usuarios “users”.



Esta petición fetch es la que nos permite comunicar el fronten con el backend y mandarlo a la base de datos esta tiene el nombre de usuario y la contraseña con hash en otras palabras ya encriptada.



La función “loginUser” toma los id de los inputs del modal “login.hbs”.

Esta función toma la tabla de usuarios y comprueba si los datos que pusiste son los correctos

Si el usuario no existe o esos no son los datos da error.

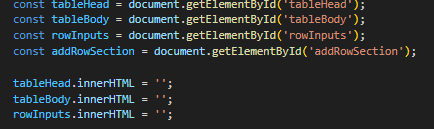
Pero si están correctos al se establece el usuario como “currentUser” en otras plabaras usuario actual.



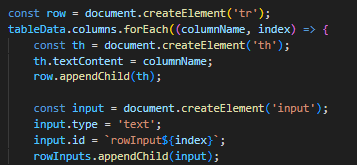


La función “loadTable” es para cargar los datos de usuario esto no es automático, sino que es al apretar el botón “loadTable”.

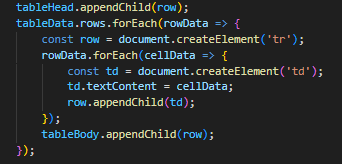
Esta comprueba el “currentUser” y toma la información que este en “${currentUser.username} \_table”.



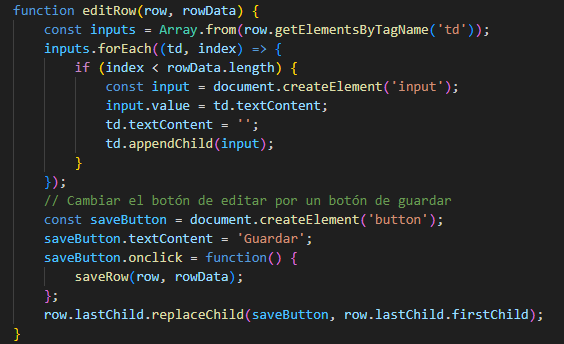
Elimina lo que estaba antes y en toma los datos de donde va a colocar la información.



Toma los datos de “columnName” para poder volver a crear el encabezado los “th” esto lo hace con un forEach en vez de un for que fue el cómo se creó. Trae los “rowInput” para que puedas seguir agregando datos.

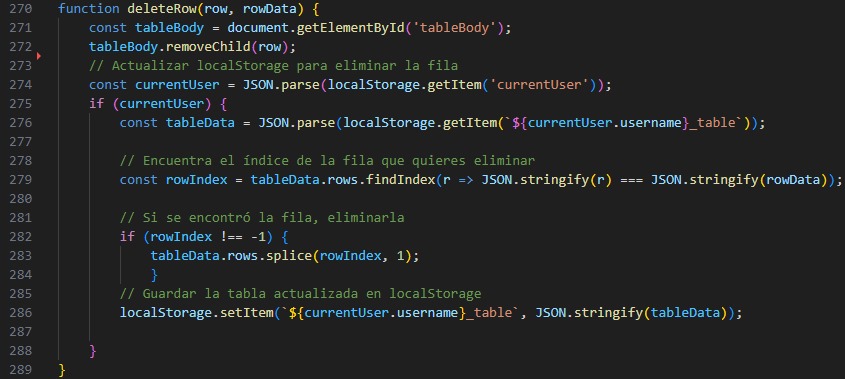


Esta parte crea los “td” que son los que contienen los datos de las tablas y este a diferencia de la primera vez este se crea con un forEach.



“editRow” toma 2 datos el “row” que es el elemento “tr” que seria toda la fila y “rowData” que son los valores de esa fila.

Toma todos los “td” dentro de la fila y los pasa a un array para usarlo en el forEach este tomara los “td” y los convierte a inputs.

Crea un botón para guardar que llama a la función “saveRow ()” y el botón "Guardar" reemplaza el botón de "Editar".

“deleteRow” primero toma la posición y se borra de la interfaz después mira cual es el usuario actual “currentUser” y toma los datos de la tabla

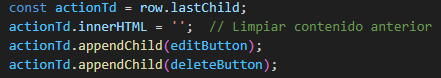
el “currentUser\_table” y busaca los datos que querías eliminar si los encuentra los borra y se guarda.



“saveRow” el inicio del código es el mismo que “deleteRow” ya que este para guardar la información primero tiene que borrar la antigua, la diferencia es que este no borra los datos de la interfaz.

Eso se logra al no usar “tableBody.removeChild(row);”

Ahora hacernos el proceso inverso a “editRow” ahora el forEach es para pasar los “inputs” a “td” y traemos otra vez los botones de “Editar” y “Eliminar” con sus funciones correspondientes “editRow” y “deleteRow”.



Estas son las que se encargan de agregarlas a la interfaz los botones. Al final agregamos la información que colocamos en los inputs al localStorage.

Nota: Borramos al inicio de la función porque si borramos después de agregar la información nueva este borrara del localStorage la información nueva y si lo colocamos después de los inpus pero antes de agregar la información nueva este no hace nada porque toma los valores de los inputs y no los encuentra en el localStorage.