**1.- Store.js :** Creamos un store con redux toolkit y envolvemos la app con este. En index.js importamos el store y “provider (react redux). Usamos la etiqueta provider y le pasamos como parámetro el store que hemos creado.

**2.- Reducers:** En esta app tenemos dos reducers:

* userSlice: Tiene la variable user que inicialmente tiene el valor de “null”. Este reducer podrá asignar un usuario o eliminarlo con “setUser” y “unsetUser”.
* visualsSlice: Tiene las variables isOpen (para abrir y cerrar el menú) y darkMode (para poner la aplicación en modo oscuro). Para cambiar el valor se usa “setMenu” y “setDarkMode”. Inicialmente las dos están en false.

**3.- App.js:** Aquí lo que hay que destacar el es useEffect que contiene el auth (una variable contiene el comando firebase.auth() situado en firebase.js). Este junto onAuthStateChanged (auth.onAuthStateChanged) lo que hace es estar atento a cualquier cambio en el usuario. En el caso de que exista un usuario al iniciar la app se establecerá un usuario (dispatch(setUser(user)) y si no, no hará nada.

**4.- Header.js**: Hay que destacar un useEffect que cierra el menú lateral en el caso de que no haya un usuario logueado o no existe el usuario. (isOpen === true && user === null).

En la parte del header de la derecha (header\_\_right), este se hace visible solo si existe un usuario logueado con la clase hideBm.

En el menú lateral, para evitar errores usamos el signo de interrogación (?) en la variable user. A esto se le denomina encadenamiento opcional y si no se consigue el valor de la variable, esta se convierte en undefined.

Por otro lado tenemos el elemento Switch que es un elemento de Material UI que hace que se active el darkMode o se desactive.

**5.- Login.js:** Aquí hay que tener en cuenta es la función signIn, que lo que hace es permitir al usuario loguearse y pasa el resultado al reducer “setUser” y este coloca como usario el resultado que ha tenido.

**6.- Main.js:** Lo primero es tener en cuenta el useEffect, el cual va a tomar de la base de datos de firebase los posts y va a ponerlas en el estado posts (useState) usando setPosts (crea un array de objetos que contiene la id de cada post y data mediante map) y por otro lado con closeMm() lo que hace es cerrar el menú de los mensajes para que al loguear no aparezcan abiertos.

Luego tenemos la función closeMm que lo que hace es cerrar el menú de cada post (Ver Likes, Eliminar Mensaje), cambiando el parámetro messmenu a false. Para llegar a este parámetro hacemos uso de los comandos de firebase.

Cuando el usuario escribe un mensaje y si da intro, se crea un post con una serie de parámetros, este post también se agrega a la base de datos de firebase con el comando add. Esta acción la hace la función handleSubmit.

La función updateScroll lleva al usuario al final del scroll en el caso de que haya un post nuevo.

En el div main\_\_posts, se hace un map de posts (estado creado useState), este map de posts va a crear un componente Post con una serie de parámetros y el usuario va a poder visualizarlo.

Este div (main\_\_posts) esta “envuelto” por el componente “FlipMove”, un componente para React que sirve en este caso para que los posts tenga un animación al aparecer.

**7.- Posts.js:** El componente posts.js recibe todos los parámetros que tiene cada post en la base de datos de firebase (de forma desestructurada)

La función removePost, elimina el post cuando el usuario da click en “Eliminar Mensaje” en el submenú que se abre dando click en el icono de los 3 puntos de cada post.

El sistema de eliminación de posts y de likes se organiza usando el useEffect. (en el inicio del logueo de usuario)

Por defecto, el estado “likes” aparece como un array vacio. Posteriormente cuando el usuario dé likes, en el array se almacenará un objeto con 3 parámetros: Nombre de usuario, email y la url del avatar. Entonces, el estado “likes” pasará a ser un array de objetos.

Cuando el usuario haga click en el corazón, se pondrá en marcha la función “likePost”, este función toma la id del post y cambia dos parámetros del post: isActive (true /false) y likes (Aquí tenemos un condicional ternario. En el caso de que isActive sea false se activa la función “removeLikes” y en el caso de que sea true, agrega al estado “likes” un objeto con los parámetros del usuario descritos más arriba)

La función removeLikes usa el método filter, el cual retorna todos los elementos del array cuyo email de usuario no coincida con el email del usuario logueado.

Volviendo al useEffect, en este, se comprueban 3 cosas:

* 1.- Se comprueba en cada post si en el array de Likes se incluye el email del usuario logueado. Si se incluye se pasa isActive a true y cambia de color el icono del corazón si no, no hace nada.
* 2.- Se obtienen los posts (si los hay) cuyo email de usuario (autor de los posts) es el mismo que el usuario logueado, cada post o mensaje contiene un campo llamado email que registra el email del autor del mensaje.
* Los posts que coincidan pasarán su parámetro “sameuser” a true y esto permitirá que el usuario vea en esos posts la opción “Eliminar mensaje”.
* 3.- Se obtienen los posts (si los hay) cuyo email de usuario (autor de los posts) NO es el mismo que el usuario logueado. En este caso estos posts pasarán su parámetro sameuser a false. Con esto el usuario no tendrán la opción de “Eliminar Mensaje” disponible.

Para abrir el submenú que muestra las opciones: “Ver likes” y “Eliminar mensaje” se usa la función toggleMm(), con esta función el parámetro “messmenu” cambia de false a true y esto hace que se muestre el submenú.

Para generar los likes cuando el usuario de el botón a “Ver likes”, usamos la función “arrayLikes” la cual crea un elemento html “li” (usando el método document.createElement) para crear los likes de los usuarios, a continuación, se crea un bucle for que va a recorrer todo el array de likes y por cada elemento se va a crear un li con:

1. Un img dentro cuya url de la imagen va a ser la url del avatar del usuario
2. Un texto con el nombre de usuario.

Estos datos se obtienen de la variable “userData” (situada arriba del todo).

Estos li los va a tener dentro un ul y en este ul se van a agregar los li con el método appendChild.

Si no hay likes, en lugar de una lista li se mostrará un h5 con el mensaje “este mensaje no tiene likes”. Este h5 se pondrá dentro del ul con appendchild.

El fondo de la lista de likes es una capa denominada post\_\_layer, que lo que hace es tapar el fondo de todos los elementos del post que quedan atrás.

Al cerrar el menú de likes, (cuando das al icono de la equis “X”), se activa la función arrayLikesRemove, esta función elimina todos los hijos de la ul de likes usando el método firstChild (hace referencia al primer hijo que se encuentre del elemento) y removeChild. Cuando no quede ningún hijo del ul de likes, se esconde con la clase “hide” listCointainer y postLayer.

La variable “ref” en este componente hace referencia a forwardRef que es la función que usamos para las animaciones de la librería react-flip-move y se aplica al primer div del componente post para que al crearlo, realice la animación.