APLICACIÓN LISTA DE TAREAS CON REACT

Actividad 4: LISTA DE TAREAS CON REACT

PREPARACIÓN DEL PROYECTO

Eliminamos archivos que no vamos a utilizar

Vamos a trabajar en la carpeta ***src***, pero antes eliminamos archivos que no vamos a necesitar, por ejemplo, setupTests.js, reportWebVitals.js y App.test.js.

Limpiamos el archivo App.js

Queremos que aparezca en principio una página en blanco así que modificamos *App.js* eliminando todo lo que hay dentro del div principal, sin borrar ese div principal.

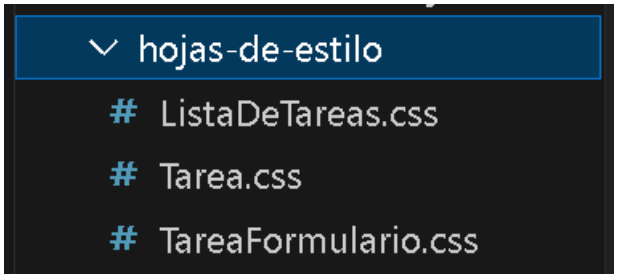
Eliminamos también la primera línea del archivo en la que se importa el logo.

Eliminamos líneas que no vamos a usar de index.js

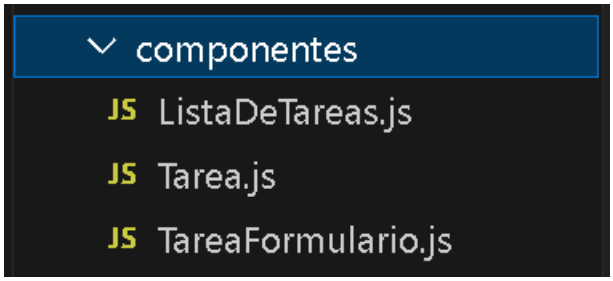
Éstas son todas las que hagan referencia a "reportwebVitals" que hemos eliminado anteriormente. Eliminamos también los comentarios.

CREAMOS CARPETAS NUEVAS

Para comenzar creamos la carpeta que almacenará los css de los elementos ListaDeTareas, Tarea y TareaFormulario descargados de la Moodle:



Y por último creamos otra carpeta que albergará nuestros archivos react con nuestros componentes ListaDeTareas, Tarea y TareaFormulario:



AÑADIMOS/SUSTITUIMOS LAS HOJAS DE ESTILO

Tendremos que actualizar las hojas de estilo App.css e index.css.

LLAMAMOS AL COMPONENTE LISTADETAREAS DESDE APP.JS

El código que devolverá App.js es el siguiente:

```
<div class=”aplicacion-tareas”>
      <div class=”tareas-lista-principal”>
        <h1>Mis Tareas</h1>
        /* Llamada al componente ListaDeTareas */
      </div>
</div>
```

CREAMOS EL COMPONENTE Tarea

Este componente recibirá (id, texto, completada, completarTarea, eliminarTarea) y devolverá el siguiente código:

```
<div class=”tarea-contenedor”>
     <div class=”tarea-texto” onClick=”completarTarea(id)”>
        /*Texto de la tarea*/
     </div>
     <div class=”tarea-contenedor-iconos”
          onClick=”eliminarTarea(id)”>
        /* Elemento icono con la clase ”tarea-icono”*/
      </div>
</div>
```

En el primer , si la tarea está completada también tendrá la clase **“completada”**.

Para insertar el icono instalaremos la librería de react con el siguiente comando:

```
npm install react-icons --save
```

En la siguiente página se muestran todos los iconos y como importarlos en nuestro archivo.
https://react-icons.github.io/react-icons/icons/ai/

CREAMOS EL COMPONENTE ListaDeTareas

El componente ***ListaDeTareas*** devolverá el siguiente código:

```
<>
/* Llamada al componente TareaFormulario */
      <div class=”tareas-lista-contenedor”>
        /* Llamada al componente Tarea enviándole una clave que
será el id, un identificador que será el id, el texto de la
tarea, si está completada, y las funciones completarTarea y
eliminarTarea */
     </div>
</>
```

La etiqueta vacía de apertura y cierre **<> </>** Se utiliza cuando vamos a enviar más de una etiqueta. React por defecto nos daría error al enviar la etiqueta <TareaFormulario> y . De modo añadiríamos una etiqueta vacía que englobe las otras dos.

Vamos a hacer uso de ***useState*** para actualizar las tareas. Pero esta vez la variable será un **array** y lo inicializaremos con corchetes vacíos [].

Crearemos una función para agregar tareas con un array de tareas que declararemos así:

```
const agregarTarea = tarea => {
    const tareasActualizadas = [tarea, ...tareas];
    setTareas(tareasActualizadas);
}
```

siendo ***tarea*** la tarea enviada al componente y ***tareas*** el array declarado en useState.

Crearemos una función para eliminar tareas del array anterior a través de su id con la siguiente línea de código:

```
const eliminarTarea = id => {
  const tareasActualizadas = tareas.filter(tarea => tarea.id !== id);
  setTareas(tareasActualizadas);
}
```

El método filter crea un nuevo array con todos los elementos que cumplan cierta condición
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

Y crearemos una función para marcar las tareas como completadas a través del id con el siguiente código:

```
const completarTarea = id => {
  const tareasActualizadas = tareas.map(tarea => {
    if (tarea.id === id) {
      tarea.completada = !tarea.completada;
    }
    return tarea;
  });
  setTareas(tareasActualizadas);
}
```

El método map va recorriendo los elementos del array, actualizando los valores.

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array/map

CREAMOS EL COMPONENTE TareaFormulario

El componente ***TareaFormulario*** devolverá:

```
<form class=”tarea-formulario” onSubmit=/*Llamada a la función que controla el envío*/>
      <input class=”tarea-input” type='text' placeholder='Escribe una Tarea' name='texto' onChange=/*Llamada a la función que controla el cambio*/
      />
      <button class=”tarea-boton”>
        Agregar Tarea
      </button>
  </form>
```

Haremos uso de ***useState*** para trabajar con la entrada del formulario, en principio la variable estará vacía (‘’).

Tendremos una función para controlar el cambio que recibirá el evento e. Es del objetivo de este evento (target) de donde sacaremos el valor introducido (value), y lo actualizaremos en la variable creada con useState.

```
const manejarCambio = e => {
  setInput(e.target.value);
}
```

Tendremos una función para controlar el envío que recibe el evento, y en primer lugar evitará que se cargue la página completa de nuevas con la línea que ya usábamos en JavaScript:

```
e.preventDefault();
```

Y añadirá una tarea nueva resultando el siguiente código:

```
const manejarEnvio = e => {
  e.preventDefault();

  const tareaNueva = {
    id: uuidv4(),
    texto: input,
    completada: false
  }

  props.onSubmit(tareaNueva);
}
```

Tendremos que instalar un paquete nuevo para controlar que nos añada identificadores únicos a nuestras tareas. Se hace tecleando el comando:

```
npm install uuid
```