Lista de tareas con IONIC 4+Javascript

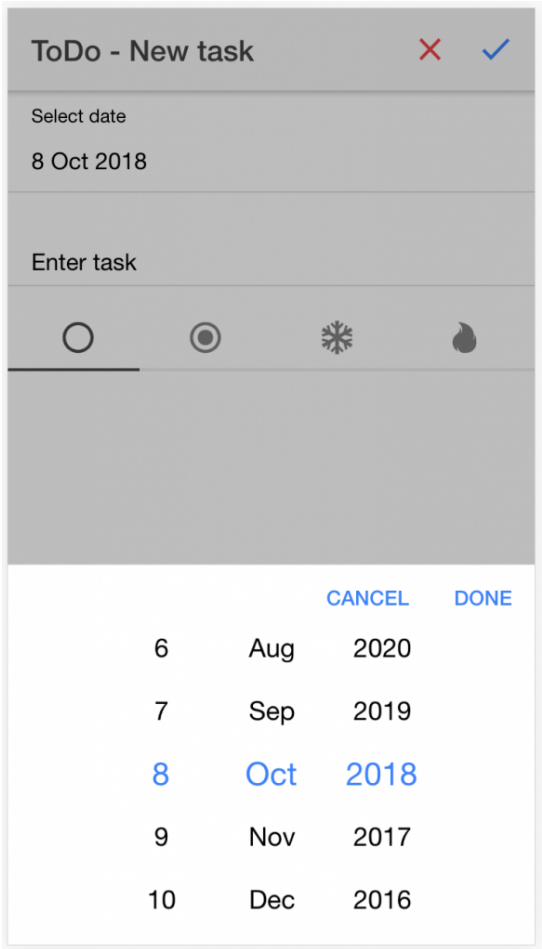
1. El objetivo

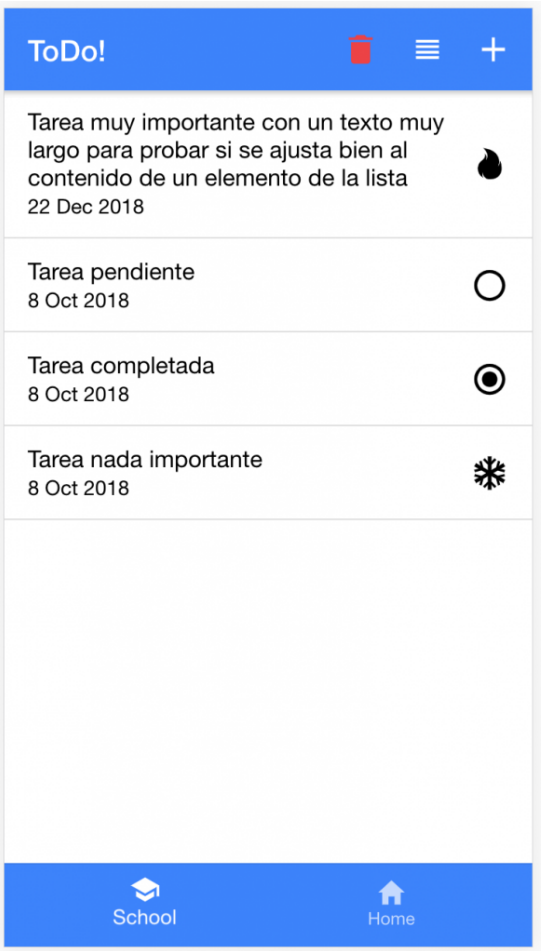
En este ejercicio vamos a desarrollar una aplicación para gestionar varias listas de tareas.

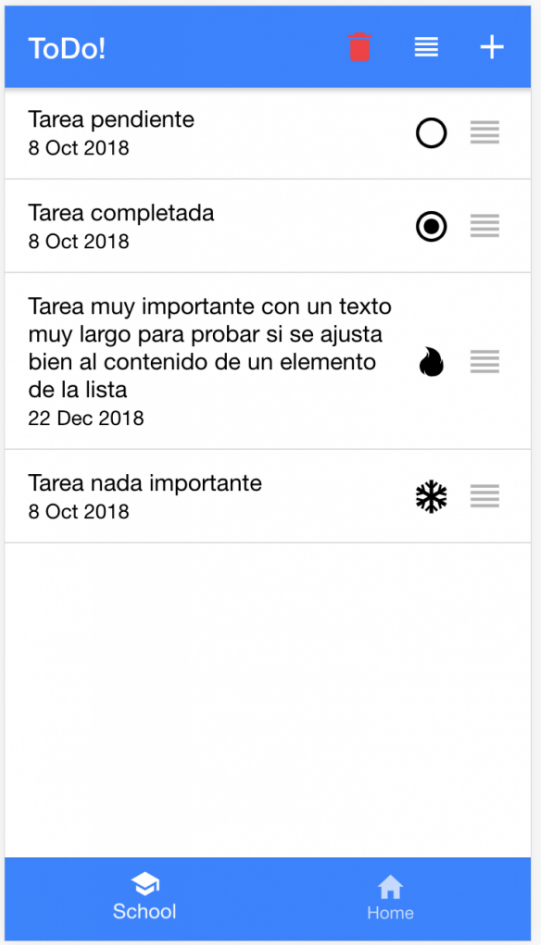
La funcionalidad a implementar será la siguiente:

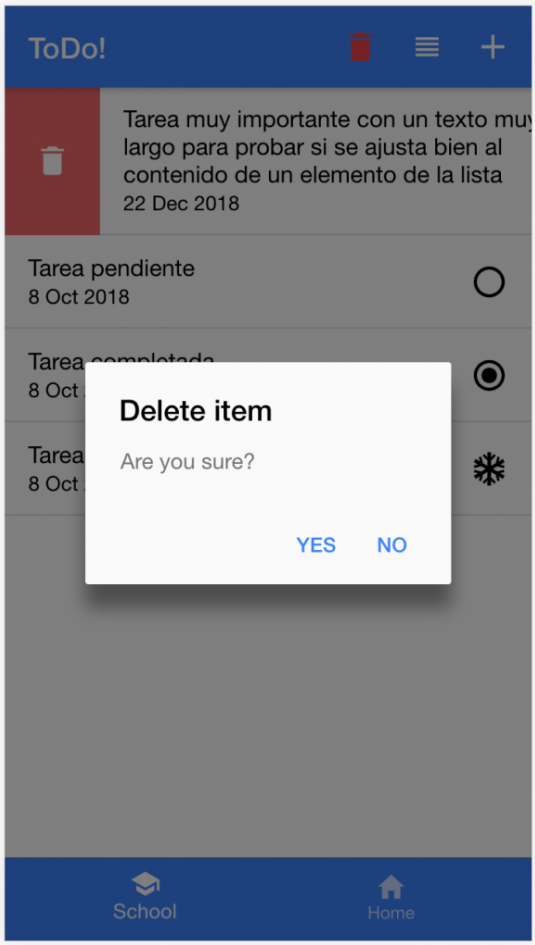
- Gestionar dos listas de tareas independientes.
- Añadir tareas indicando la fecha, la descripción de la tarea, y un icono identificativo de la prioridad.
- Borrar de golpe todas las tareas de una lista.
- Borrar una sola tarea de una lista determinada mediante un botón oculto que se visualizará al deslizar hacia la derecha la tarea a borrar.

- Pedir confirmación antes de borrar cualquier tarea.
- Visualizar y editar los detalles de cada tarea haciendo clic sobre el elemento correspondiente de la lista.
- Permitir ordenar las tareas arrastrando cualquier elemento a una nueva posición.









2. Elementos HTML de IONIC 4

2.1. El encabezado principal de la aplicación

Utilizaremos tres botones de tipo icono para acceder a las funciones básicas desde la pantalla principal:

- `<ion-icon name="trash"></ion-icon>` : Para borrar todas las tareas de la lista seleccionada.
- `<ion-icon name="reorder"></ion-icon>` : Para activar los botones que nos permitirán reordenar las tareas arrastrando y soltando.
- `<ion-icon name="add"></ion-icon>` : Para crear una nueva tarea.

Los agruparemos junto con el título de la aplicación en el encabezado del fichero ***index.html*** de la siguiente forma:

```
1.  <ion-header>
2.    <ion-toolbar color="primary">
```

```
      <ion-title>ToDo!</ion-title>
      <ion-buttons slot="primary">
        <ion-button onclick="deleteItem()" color="danger">
          <ion-icon slot="icon-only" name="trash"></ion-icon>
        </ion-button>
        <ion-button onclick="toggleReorder()">
          <ion-icon slot="icon-only" name="reorder"></ion-icon>
        </ion-button>
        <ion-button onclick="addEditItem()">
          <ion-icon slot="icon-only" name="add"></ion-icon>
        </ion-button>
      </ion-buttons>
    </ion-toolbar>
  </ion-header>
```

Más adelante veremos el código JavaScript necesario para llevar a cabo las funciones correspondientes:

- `deleteItem()`
- `toggleReorder()`
- `addEditItem()`

2.2. Las listas de tareas

Para mantener dos listas separadas, desde el archivo ***index.html*** utilizaremos el elemento `<ion-tabs></ion-tabs>` que personalizaremos escogiendo un icono y un texto descriptivo. Por ejemplo, si quisiéramos utilizar una lista de tareas para el instituto y otra para casa podríamos utilizar los iconos `school` y `home` respectivamente:

```
  <ion-tabs color="primary">
    <ion-tab tab="school">
      <ion-list lines="full">
        <ion-reorder-group>
          ...
        </ion-reorder-group>
      </ion-list>
    </ion-tab>
    <ion-tab tab="home">
      <ion-list lines="full">
        <ion-reorder-group>
          ...
```

```
      </ion-reorder-group>
    </ion-list>
  </ion-tab>
  <ion-tab-bar slot="bottom" color="primary">
    <ion-tab-button tab="home"><ion-icon name="home"></ion-icon></ion-tab-button>
    <ion-tab-button tab="school"><ion-icon name="school"></ion-icon></ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

Además utilizaremos el nuevo elemento `<ion-reorder-group></ion-reorder-group>` que nos permitirá reordenar las tareas que coloquemos dentro. Bastará con pulsar encima de un botón habilitado a tal efecto, y sin soltar arrastrar la tarea en cuestión a la nueva posición. En la documentación de IONIC podemos observar cómo esta simple etiqueta gestiona todo el proceso de arrastrar y soltar.

2.3. Las tareas dentro de las listas

Utilizaremos elementos `<ion-item-sliding></ion-item-sliding>` para cada tarea dentro de las listas. De esta forma podremos tener oculto un botón rojo con una papelera, que aparecerá al deslizar la tarea hacia la derecha, y nos permitirá borrar de la lista sólo ese elemento:

```
<ion-item-sliding>
  ...
  <ion-item-option color="danger" onclick="...">
    <ion-icon slot="icon-only" name="trash"></ion-icon>
  </ion-item-option>
</ion-item-sliding>
```

Utilizaremos también un elemento `<ion-item></ion-item>` para agrupar todos los datos de la tarea. De esta forma, bastará con hacer clic sobre una determinada tarea para acceder a los detalles de la misma: `<ion-item onclick="..."></ion-item>` . Un poco más adelante detallaremos la funcionalidad JavaScript necesaria para crear y mostrar un formulario que nos permita visualizar y editar la tarea seleccionada.

Resaltaremos la descripción de la tarea con una etiqueta `<h2></h2>`, y a continuación mostremos la fecha de la misma dentro de un párrafo (etiqueta `<p></p>`). Encerraremos ambos elementos con una etiqueta `<ion-label text-wrap></ion-label>` que permitirá que el tamaño de cada elemento de la lista se ajuste para contener todo el texto descriptivo de la tarea.

Resumiendo, cada elemento de la lista de tareas vendrá especificado por el siguiente código HTML:

```
<ion-item-sliding>
  <ion-item onclick="...">
    <ion-label text-wrap>
      <h2>...</h2>
      <p>...</p>
    </ion-label>
    <ion-icon slot="end" name="..."></ion-icon>
    <ion-reorder slot="end"></ion-reorder>
  </ion-item>
  <ion-item-options side="start">
    <ion-item-option color="danger" onclick="...">
      <ion-icon slot="icon-only" name="trash"></ion-icon>
    </ion-item-option>
  </ion-item-options>
</ion-item-sliding>
```

2.4. El formulario con los detalles de cada tarea

Para visualizar y editar los detalles de cada tarea (al crear una nueva, o al hacer clic sobre una ya existente) utilizaremos el siguiente código HTML:

```
<ion-header>
  <ion-toolbar>
    <ion-title>ToDo - Task details</ion-title>
    <ion-buttons slot="primary">
      <ion-button color="danger"><ion-icon slot="icon-only" name="close"></ion-icon></ion-button>
      <ion-button color="primary"><ion-icon slot="icon-only" name="checkmark"></ion-icon></ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
```

```
10.
11.   <ion-content>
12.     <ion-list>
13.       <ion-item>
14.         <ion-label position="floating">Select date</ion-label>
15.         <ion-datetime display-format="D MMM YYYY" max="2050-12-
    31" value="..."></ion-datetime>
16.       </ion-item>
17.       <ion-item>
18.         <ion-label position="floating">Enter task</ion-label>
19.         <ion-input value="..."></ion-input>
20.       </ion-item>
21.     </ion-list>
22.
23.     <ion-segment value="...">
24.       <ion-segment-button value="radio-button-off">
25.         <ion-icon name="radio-button-off"></ion-icon>
26.       </ion-segment-button>
27.       <ion-segment-button value="radio-button-on">
28.         <ion-icon name="radio-button-on"></ion-icon>
29.       </ion-segment-button>
30.       <ion-segment-button value="snow">
31.         <ion-icon name="snow"></ion-icon>
32.       </ion-segment-button>
33.       <ion-segment-button value="flame">
34.         <ion-icon name="flame"></ion-icon>
35.       </ion-segment-button>
36.     </ion-segment>
37.   </ion-content>
```

En el encabezado (`<ion-header></ion-header>`) simplemente tenemos dos botones para confirmar o cancelar la edición o creación de la tarea.

En el contenido principal del formulario (`<ion-content></ion-content>`) tendremos tres campos:

- `<ion-datetime><ion-datetime>` : Para indicar la fecha de la tarea. Si estamos creando una nueva tarea, en dicho valor introduciremos la fecha actual utilizando código JavaScript.
- `<ion-input></ion-input>` : Para introducir o modificar la descripción de la tarea. Estará en blanco si se trata de una tarea nueva.
- `<ion-segment></ion-segment>` : Para seleccionar la prioridad de la tarea. En nuestro ejemplo proponemos 4 opciones (`radio-button-off`

para tareas pendientes, `radio-button-on` para tarea finalizada, `snow` para una prioridad baja, y `flame` para una prioridad alta), aunque los iconos utilizados son una simple sugerencia, y se pueden cambiar fácilmente según el gusto de cada uno.

3. El código JavaScript

3.1. Accediendo a cada una de las listas

Por simplificar un poco el código, utilizaremos tres funciones para acceder y actualizar los dos elementos principales (*tabs* y *lists*):

```
1.  function getTab() {
2.      return(document.querySelector('ion-tab:not(.tab-
    hidden)'));
3.  }
4.
5.  function getList(tab = getTab()) {
6.      let list = localStorage.getItem('todo-list-'+tab.tab);
7.      return list ? JSON.parse(list) : [];
8.  }
9.
10. function saveList(tab, list) {
11.     localStorage.setItem('todo-list-'+tab.tab,
    JSON.stringify(list));
12.     printList(tab);
13. }
```

La última función (`saveList()`) nos permitirá guardar de manera persistente cada una de las listas de tareas, utilizando `localStorage` , de forma que al actualizar el contenido del navegador (o cerrar la app y volverla a abrir), podamos volver a cargar la lista de tareas.

3.2. Construyendo cada elemento de la lista de tareas

Mediante la siguiente función obtendremos todo el código necesario para cada tarea, variando simplemente el texto descriptivo de la misma, la fecha, y el icono

indicativo de la prioridad:

```
function printList(tab) {
    tab.querySelector('ion-reorder-group').innerHTML = "";

    getList(tab).forEach((item, index) => {
        tab.querySelector('ion-reorder-group').innerHTML +=
        `<ion-item-sliding>
           <ion-item onClick="addEditItem(`+index+`)">
             <ion-label text-wrap>
               <h2>`+item.text+`</h2>
               <p>`+item.date.slice(0,10)+`</p>
             </ion-label>
             <ion-icon slot="end" name="`+item.icon+`"></ion-
icon>
             <ion-reorder slot="end"></ion-reorder>
           </ion-item>
           <ion-item-options side="start">
             <ion-item-option color="danger"
onClick="deleteItem(`+index+`)">
               <ion-icon slot="icon-only" name="trash"></ion-
icon>
             </ion-item-option>
           </ion-item-options>
         </ion-item-sliding>`;
    });
}
```

3.3. Para añadir una nueva tarea o editar los detalles de una tarea existente

Utilizaremos la misma función (`addEditItem(index)`) para añadir una nueva tarea o modificar una existente. En ella, siguiendo las instrucciones indicadas en la documentación de IONIC, crearemos un cuadro de diálogo modal que mostrará el formulario para introducir o actualizar los detalles de la tarea:

```
function addEditItem(index = false) {
    closeItems();
    let list = getList();
    let item = null;

    if (index !== false) item = list[index];
```

```
    else item = { text:"", date:new Date().toISOString(), icon:"radio-button-off" };

    const modal = document.createElement('ion-modal');
    modal.component = document.createElement('div');
    modal.component.innerHTML = `
        <ion-header>
            ...
        </ion-header>
        <ion-content>
            ...
        </ion-content>`;

 modal.component.querySelector('[color="danger"]').addEventListener('click', () => {
        modal.dismiss();
    });


modal.component.querySelector('[color="primary"]').addEventListener('click', () => {
        let newDate = modal.component.querySelector('ion-datetime').value;
        let newText = modal.component.querySelector('ion-input').value;
        let newIcon = modal.component.querySelector('ion-segment').value;

        if (!newText.length) {
            error('The task cannot be empty');
        }
        else {
            let newItem = { text:newText, date:newDate, icon:newIcon };
            if (index !== false) list[index] = newItem;
            else list.unshift(newItem);
            saveList(getTab(), list);
            modal.dismiss();
        }
    });

    document.querySelector('ion-app').appendChild(modal);
    modal.present();
}
```

En la primera parte de la función, si recibimos como parámetro la tarea a modificar, recuperaremos los detalles de la misma (fecha, texto descriptivo e

icono con la prioridad):

```
if (index !== false) item = list[index];
else item = { text:"", date:new Date().toISOString(),
icon:"radio-button-off" };
```

Como se puede observar en la fecha, si en vez de modificar una tarea existente, estamos creando una nueva (parámetro de la función con valor `false`), entonces obtendremos la fecha actual.

A continuación, crearemos el formulario de edición o creación de tareas, con un encabezado con los botones correspondientes para cancelar y confirmar, y también con el contenido principal, utilizando el código HTML indicado previamente:

```
const modal = document.createElement('ion-modal');
modal.component = document.createElement('div');
modal.component.innerHTML = `
    <ion-header>
      ...
    </ion-header>
    <ion-content>
      ...
    </ion-content>`;
```

Finalmente, nos queda capturar los eventos de clic en el botón de cancelar (`color="danger"`) o confirmar (`color="primary"`). En el primer caso simplemente cerraremos el cuadro de diálogo sin hacer nada más:

```
modal.component.querySelector('[color="danger"]').addEventList
ener('click', () => {
        modal.dismiss();
    });
```

En caso de confirmar la edición o creación de una nueva tarea, primero cogeremos los valores introducidos en el formulario:

```
modal.component.querySelector('[color="primary"]').addEventLis
tener('click', () => {
        let newDate = modal.component.querySelector('ion-
datetime').value;
```

```
        let newText = modal.component.querySelector('ion-input').value;
        let newIcon = modal.component.querySelector('ion-segment').value;

    ...
});
```

En segundo lugar comprobaremos la longitud de la descripción de la tarea, para asegurarnos que no está vacía, en cuyo caso mostraríamos un error:

```
function error(message) {
  const alert = document.createElement('ion-alert');
  alert.message = message;
  alert.buttons = ['OK'];

  document.querySelector('ion-app').appendChild(alert);
  alert.present();
}

function addEditItem(index = false) {
    ...
        if (!newText.length) {
            error('The task cannot be empty');
        }
    ...
}});
```

Si estamos modificando una tarea existente, crearemos la nueva tarea con los nuevos datos del formulario, la reemplazaremos por la existente, guardaremos la lista, y cerraremos el cuadro de diálogo.

Si estamos añadiendo una nueva tarea, crearemos un nuevo elemento `<ion-item-sliding></ion-item-sliding>` que contendrá la fecha, descripción e icono introducidos en el formulario. A continuación añadiremos dicha tarea a la lista que estemos modificando, la guardaremos, y cerraremos el cuadro de diálogo:

```
let newItem = { text:newText, date:newDate, icon:newIcon };

if (index !== false) { list[index] = newItem; }
else { list.unshift(newItem); }

```

```
6.  saveList(getTab(), list);
7.  modal.dismiss();
```

3.4. Reordenando las tareas

La totalidad de la funcionalidad necesaria para reordenar las tareas de una lista ya nos la proporciona IONIC mediante el elemento `<ion-reorder></ion-reorder>` . Lo único que debemos hacer por nuestra parte, es cambiar el valor del atributo `disabled` del elemento `<ion-reorder-group></ion-reorder-group>` de la lista seleccionada:

```
1.  function toggleReorder() {
2.      closeItems();
3.      let reorder = getTab().querySelector('ion-reorder-group');
4.      reorder.disabled = !reorder.disabled;
5.  }
```

En caso de que se esté mostrando algún botón de borrado de una tarea, se ocultará para habilitar correctamente la funcionalidad de reordenar elementos.

Cuando la opción de reordenar se encuentra activa, se mostrará un nuevo icono en todas las tareas. Si lo mantenemos pulsado, nos permitirá arrastrar la tarea correspondiente para soltarla en una nueva posición.

Al volver a pulsar el botón de reordenar ubicado en el encabezado de la aplicación, la opción de reordenar se desactivará, evitando que cambiemos accidentalmente el orden de las tareas.

3.5. Borrando tareas

Para borrar todas las tareas de la lista seleccionada, pulsaremos el botón que hemos puesto en el encabezado principal de la aplicación. Y para borrar una tarea específica, pulsaremos el botón que aparece al deslizar la tarea hacia la derecha. En ambos casos, utilizaremos la misma función JavaScript (`deleteItem(index)`), pasando como parámetro el elemento HTML que

contiene la tarea a borrar, o el valor false para borrar todas las tareas de la lista seleccionada.

Para evitar borrados accidentales, pediremos confirmación al usuario utilizando un cuadro de diálogo de tipo *Alert* que ya nos proporciona IONIC:

```
function deleteItem(index = false) {
    const alert = document.createElement('ion-alert');

    alert.header = index !== false ? 'Delete item' : 'Delete all',
    alert.message = 'Are you sure?',
    alert.buttons =
            [{
                text: 'YES',
                handler: () => {
                    let list = getList();
                    if (index !== false) { list.splice(index, 1); }
                    else { list.length = 0; }
                    saveList(getTab(), list);
                }
            }, {
                text: 'NO',
                role: 'cancel'
            }]

    document.querySelector('ion-app').appendChild(alert);
    alert.present();
}
```

Como se puede observar, al confirmar el borrado, si se recibe un elemento por parámetro, sólo se borra dicho elemento. En caso contrario, se borrará toda la lista, utilizando una cadena vacía para eliminar todo el código HTML que contenía. Finalmente, guardaremos la lista correspondiente:

```
...
    let list = getList();
    if (index !== false) { list.splice(index, 1); }
    else { list.length = 0; }
    saveList(getTab(), list);
...
```

3.6. Eventos a tener en cuenta

Cada vez que reordenemos las tareas de alguna lista, deberemos volverla a guardar para que aparezcan en el orden correcto al volver a abrir la aplicación. Para ello capturaremos el evento `ionItemReorder`, tal como se indica en la documentación de IONIC, apartado de JavaScript:

```
1.  document.addEventListener('ionItemReorder', (event) => {
    moveItem(event.detail); });
2.  ...
3.  function moveItem(indexes) {
4.      let tab = getTab();
5.      let list = getList(tab);
6.      let item = list[indexes.from];
7.      list.splice(indexes.from, 1);
8.      list.splice(indexes.to, 0, item);
9.      indexes.complete();
10.     saveList(tab, list);
11. }
```

Cada vez que cerremos un cuadro de diálogo (eventos `ionModalDidDismiss` y `ionAlertDidDismiss`) cerraremos los elementos de la lista que pudieran estar deslizados hacia la derecha.

```
1.  document.addEventListener("ionModalDidDismiss", closeItems);
2.  document.addEventListener("ionAlertDidDismiss", closeItems);
3.  ...
4.  function closeItems() {
5.      getTab().querySelector('ion-list').closeSlidingItems();
6.  }
```

Y por último, cada vez que iniciemos la aplicación (evento `onload`), leeremos las listas que se encuentran guardadas en el navegador:

```
1.  function onLoad() {
2.      ...
3.      document.querySelectorAll('ion-tab').forEach(function(t) {
    printList(t); });
4.  }
```

4. En resumen...

4.1. El fichero *index.html*

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>ToDo!</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">

  <script type="module" src="https://cdn.jsdelivr.net/npm/@ionic/core@4.11.0/dist/ionic/ionic.esm.js"></script>
  <script nomodule src="https://cdn.jsdelivr.net/npm/@ionic/core@4.11.0/dist/ionic/ionic.js"></script>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@ionic/core@4.11.0/css/ionic.bundle.css"/>

  <script src="cordova.js"></script>
  <script src="todo.js"></script>
</head>

<body onload="onLoad()">
  <ion-app>
    <ion-header>
      <ion-toolbar color="primary">
        <ion-title>ToDo!</ion-title>
        <ion-buttons slot="primary">
          <ion-button onclick="deleteItem()" color="danger">
            <ion-icon slot="icon-only" name="trash"></ion-icon>
          </ion-button>
          <ion-button onclick="toggleReorder()">
            <ion-icon slot="icon-only" name="reorder"></ion-icon>
          </ion-button>
          <ion-button onclick="addEditItem()">
            <ion-icon slot="icon-only" name="add"></ion-icon>
          </ion-button>
        </ion-buttons>
```

```
      </ion-toolbar>
    </ion-header>

    <ion-content>
      <ion-tabs>
        <ion-tab tab="home">
          <ion-list lines="full">
            <ion-reorder-group></ion-reorder-group>
          </ion-list>
        </ion-tab>
        <ion-tab tab="school">
          <ion-list lines="full">
            <ion-reorder-group></ion-reorder-group>
          </ion-list>
        </ion-tab>
        <ion-tab-bar slot="bottom" color="primary">
          <ion-tab-button tab="home"><ion-icon name="home">
</ion-icon></ion-tab-button>
          <ion-tab-button tab="school"><ion-icon
name="school"></ion-icon></ion-tab-button>
        </ion-tab-bar>
      </ion-tabs>
    </ion-content>
  </ion-app>
</body>
</html>
```

4.2. El fichero *todo.js*

```
function onLoad() {
    document.addEventListener("ionModalDidDismiss",
closeItems);
    document.addEventListener("ionAlertDidDismiss",
closeItems);
    document.addEventListener("ionDidOpen", closeItems);
    document.addEventListener('ionItemReorder', (event) => {
moveItem(event.detail); });
    document.querySelectorAll('ion-tab').forEach(function(t) {
printList(t); });
}

function getTab() {
    return(document.querySelector('ion-tab:not(.tab-
hidden)'));
}
```

```
12.
13.  function getList(tab = getTab()) {
14.      let list = localStorage.getItem('todo-list-'+tab.tab);
15.      return list ? JSON.parse(list) : [];
16.  }
17.
18.  function saveList(tab, list) {
19.      localStorage.setItem('todo-list-'+tab.tab,
     JSON.stringify(list));
20.      printList(tab);
21.  }
22.
23.  function printList(tab) {
24.      tab.querySelector('ion-reorder-group').innerHTML = "";
25.
26.      getList(tab).forEach((item, index) => {
27.          tab.querySelector('ion-reorder-group').innerHTML +=
28.          `<ion-item-sliding>
29.             <ion-item onClick="addEditItem(`+index+`)">
30.               <ion-label text-wrap>
31.                 <h2>`+item.text+`</h2>
32.                 <p>`+item.date.slice(0,10)+`</p>
33.               </ion-label>
34.               <ion-icon slot="end" name="`+item.icon+`"></ion-
     icon>
35.               <ion-reorder slot="end"></ion-reorder>
36.           </ion-item>
37.           <ion-item-options side="start">
38.             <ion-item-option color="danger"
     onClick="deleteItem(`+index+`)">
39.                 <ion-icon slot="icon-only" name="trash"></ion-
     icon>
40.             </ion-item-option>
41.           </ion-item-options>
42.         </ion-item-sliding>`;
43.      });
44.  }
45.
46.  function error(message) {
47.    const alert = document.createElement('ion-alert');
48.    alert.message = message;
49.    alert.buttons = ['OK'];
50.
51.    document.querySelector('ion-app').appendChild(alert);
52.    alert.present();
53.  }
54.
55.  function toggleReorder() {
```

```
56.     closeItems();
57.     let reorder = getTab().querySelector('ion-reorder-group');
58.     reorder.disabled = !reorder.disabled;
59. }
60. 
61. function closeItems() {
62.     getTab().querySelector('ion-list').closeSlidingItems();
63. }
64. 
65. function addEditItem(index = false) {
66.     closeItems();
67.     let list = getList();
68.     let item = null;
69. 
70.     if (index !== false) item = list[index];
71.     else item = { text:"", date:new Date().toISOString(),
    icon:"radio-button-off" };
72. 
73.     const modal = document.createElement('ion-modal');
74.     modal.component = document.createElement('div');
75.     modal.component.innerHTML = `
76.         <ion-header>
77.             <ion-toolbar>
78.                 <ion-title>ToDo - `+(index !== false ? 'Edit
    task' : 'New task')+`</ion-title>
79.                 <ion-buttons slot="primary">
80.                     <ion-button color="danger"><ion-icon
    slot="icon-only" name="close"></ion-icon></ion-button>
81.                     <ion-button color="primary"><ion-icon
    slot="icon-only" name="checkmark"></ion-icon></ion-button>
82.                 </ion-buttons>
83.             </ion-toolbar>
84.         </ion-header>
85.         <ion-content>
86.             <ion-list>
87.                 <ion-item>
88.                     <ion-label position="floating">Select
    date</ion-label>
89.                     <ion-datetime display-format="D MMM YYYY"
    max="2050-12-31" value="`+item.date+`"></ion-datetime>
90.                 </ion-item>
91.                 <ion-item>
92.                     <ion-label position="floating">Enter
    task</ion-label>
93.                     <ion-input value="`+item.text+`"></ion-
    input>
94.                 </ion-item>
95.             </ion-list>
```

```
            <ion-segment value="`+item.icon+`">
                <ion-segment-button value="radio-button-off">
                    <ion-icon name="radio-button-off"></ion-icon>
                </ion-segment-button>
                <ion-segment-button value="radio-button-on">
                    <ion-icon name="radio-button-on"></ion-icon>
                </ion-segment-button>
                <ion-segment-button value="snow">
                    <ion-icon name="snow"></ion-icon>
                </ion-segment-button>
                <ion-segment-button value="flame">
                    <ion-icon name="flame"></ion-icon>
                </ion-segment-button>
            </ion-segment>
        </ion-content>`;


modal.component.querySelector('[color="danger"]').addEventListener('click', () => {
        modal.dismiss();
    });


modal.component.querySelector('[color="primary"]').addEventListener('click', () => {
        let newDate = modal.component.querySelector('ion-datetime').value;
        let newText = modal.component.querySelector('ion-input').value;
        let newIcon = modal.component.querySelector('ion-segment').value;

        if (!newText.length) {
            error('The task cannot be empty');
        }
        else {
            let newItem = { text:newText, date:newDate, icon:newIcon };
            if (index !== false) list[index] = newItem;
            else list.unshift(newItem);
            saveList(getTab(), list);
            modal.dismiss();
        }
    });

    document.querySelector('ion-app').appendChild(modal);
```

```
    modal.present();
}

function moveItem(indexes) {
    let tab = getTab();
    let list = getList(tab);
    let item = list[indexes.from];
    list.splice(indexes.from, 1);
    list.splice(indexes.to, 0, item);
    indexes.complete();
    saveList(tab, list);
}

function deleteItem(index = false) {
    const alert = document.createElement('ion-alert');

    alert.header = index !== false ? 'Delete item' : 'Delete all',
    alert.message = 'Are you sure?',
    alert.buttons =
            [{
                text: 'YES',
                handler: () => {
                    let list = getList();
                    if (index !== false) { list.splice(index, 1); }
                    else { list.length = 0; }
                    saveList(getTab(), list);
                }
            }, {
                text: 'NO',
                role: 'cancel'
            }]

    document.querySelector('ion-app').appendChild(alert);
    alert.present();
}
```

5. El resultado

Puedes hacer clic aquí para observar el aspecto que tiene la aplicación de lista de tareas y probar la funcionalidad resultante utilizando el código especificado.