



70

## S7 | Mejorando TodoistHub

En la última sesión estuvimos trabajando juntos para crear nuestra propia réplica de Todoist.

Esta réplica cumplirá con las funcionalidades propuestas previamente, una buena maquetación, y un cambio de branding para adaptarlo a nuestro estilo personal.

En la sesión de hoy vamos a seguir trabajando con una metodología similar, pero tendremos en cuenta lo siguiente:

- El objetivo del proyecto de la última clase es el MVP que debemos conseguir. Por lo que si has conseguido llegar hasta ese punto, habrás cumplido con el objetivo propuesto 🎉 Recuerda que le habrás dado un toque personal al proyecto.
- En esta sesión propondremos la aplicación de un sistema similar a una API a través del uso de JSON y una librería adicional. De forma que podremos simular el guardado en servidor y peticiones de datos.

- Añadirás alguna funcionalidad adicional entre las propuestas o alguna que te gustaría que existiese en el proyecto.

## Consumiendo una API (fake)

Para consumir una API, no crearemos un servidor en node con una base de datos, ya que estamos en un módulo y un proyecto centrados totalmente en el Frontend, por lo que simularemos el comportamiento de una API funcional.

Esto lo conseguiremos mediante el uso de la librería `lowdb`, la cual nos permite utilizar un JSON almacenado en LocalStorage como base de datos temporal, para poder así persistir información.

```
npm i lowdb
```

Ahora podemos crear un archivo de lanzamiento de la "base de datos" que almacene la información en LocalStorage (lo llamaremos `loadDb.js`):

```
/**
 * Usaremos este archivo para configurar rápidamente una base de datos
 * en formato JSON en nuestro frontend. De esta forma podremos simular la
 * carga de información de una API sin necesidad de crear una desde cero.
 */
import low from 'lowdb';
import LocalStorage from 'lowdb/adapters/LocalStorage';

const adapter = new LocalStorage('db');
const db = low(adapter);

// Añadimos valores por defecto a nuestra nueva DB
db.defaults({ todos: [], completed: [] }).write();

export default db;
```

Si importamos este archivo en cualquier sitio tendremos acceso a la información que existe en nuestro LocalStorage 🚀

Ahora podremos cargar la información que tenemos desde la "base de datos" a través de ciclos de vida y simular un servicio.

Aquí un ejemplo de algunas funciones de `lowdb` que pueden ser de utilidad. Son extensiones de las funciones que tiene `lodash` por lo que podremos usarlo de la misma forma:

```
db.getState(); // Trae toda la base de datos  
  
db.get('todos').value() // Trae nuestra lista de todos  
  
// Altera el valor de todos en nuestra store  
db.set('todos', [{ id: '1', text: 'Soy un todo' }])  
  
// Trae los todos de la store y filtra los que tengan id '1'  
db.get('todos').filter({ id: '1' }).value()
```

Tienes más ejemplos en la documentación, pero con esto podremos comenzar a usar nuestro nuevo storage.

Aquí un ejemplo de su uso en `App` para cargar los datos cuando el componente se renderiza por primera vez:

```
componentDidMount() {  
  const { todos, completed } = db.getState();  
  this.setState({ todos, completed });  
}
```

No es necesario que usemos promesas ni `async/await` en este caso, pero recuerda que en una API real deberíamos esperar los resultados 

## Propuestas de nuevos añadidos

A continuación proponemos una serie de funcionalidades que podemos añadir a nuestra App si nos vemos bien de tiempo y queremos subir nuestro nivel:

- Crea un botón, usando un ícono de corazón o similar, que permita añadir una tarea a un array `favorites`. Esto conllevará crear una nueva ruta `/favorites` en la que mostraremos la lista de todos favoritos que podremos completar al igual que en la ruta `/`.
- Modifica los `todos` para que podamos añadir comentarios a estos justo debajo del título. Para ello tendremos que añadir un nuevo campo `description` a nuestras tareas y modificar los componentes necesarios para que se vea correctamente. Puedes añadir el input de descripción en la creación de tickets.
- Añade una funcionalidad que permita, mediante el click de un botón `Editar` o un ícono tipo lápiz de FontAwesome, cambiar el componente `Card` seleccionado por un input de edición para poder modificar los valores que queramos de una carta (texto y descripción) cuando guardemos.
- Añade un array de usuarios semilla a nuestro storage con `lowdb`. Mueve la lista de `todos` inicial que teníamos en `/` al endpoint `/all` o similar. Ahora añade un formulario de inicio de sesión en `/` y protege el resto de las rutas para que únicamente puedan acceder usuarios que se han autenticado

mediante el formulario y hemos encontrado dicho usuario en el array de la base de datos.

## Presentación del proyecto

Presentaremos el proyecto en la siguiente sesión o sesión que acordemos todos juntos. De esta forma podrás trabajarla y terminar de darle el alcance que veas adecuado a tu proyecto.

Esta actividad nos habrá permitido tener un nuevo añadido a nuestro portfolio que mejorará nuestra visibilidad laboral, nos habrá convertido en mejores profesionales y habremos asentado el conocimiento obtenido hasta este punto en el módulo de React. 