Redux-Thunk es una librería desarrollada por el creador de ReduxJS. **Esta librería nos permite definir acciones de Redux que ejecuten código asíncrono**.

Se utiliza como un middleware. **Los middleware son librerías de Redux que reciben las acciones antes de que lleguen a los reducers**. Estas pueden modificar la acción, pararla o lanzar otras.

Primero necesitamos instalar la librería. Esta se distribuye como un paquete `npm`.

```
npm install --save redux-thunk
```

El siguiente paso es agregar el middleware a Redux. Vamos a utilizar el método `applyMiddleware` de ReduxJS. Agregar middleware es muy sencillo, solo necesitamos modificar el fichero `src/store.js`:

```
// Importamos el método para crear el store de redux
import { createStore, applyMiddleware } from 'redux';

// Importamos la librería
import thunk from 'redux-thunk';

// Importamos el reducer de nuestra aplicación
import Reducer from './reducers/reducer';

// Creamos el reducer
const store = createStore(
  Reducer,
  // Aplicamos el middleware
  applyMiddleware(thunk)
);
export default store;
```

Ahora vamos a definir una acción que utilice el middleware. Estas acciones no retornan un objeto, si no que retorna una función que recibe como parámetro el método `dispatch`. Dentro de la acción podemos realizar llamadas asíncronas y ejecutar `dispatch` cuando sea necesario. Definimos la acción `search` en el fichero `src/actions/actions.js`:

```
// Comenzamos una nueva búsqueda
/* export const startSearch = ... */

// Retornamos los resultados
/* export const successSearch = ... */

export const search = value =>
  dispatch => {
    // Lanzamos la acción startSearch
    dispatch(startSearch(value));
```

```
    // Realizamos la búsqueda
    fetch(`https://api.github.com/search/repositories?q=${ value }`)
      .then(res => {
        return res.json()
      })
      .then(res => {
        // Almacenamos el resultado en redux
        dispatch(successSearch(res.items));
      })
      .catch(err => {
        console.log(err);
      });
  }
```

Esta acción se encarga de realizar las llamadas y el `dispatch` de las acciones por lo que el método `onSubmit` de `SearchContainer.js` se reduce a:

```
/**
 * Este método actua como callback del evento onSubmit del formulario.
 * Recibe como parámetro el campo que debe de buscar.
 */
onSubmit = value => {
  // Lanzamos la accion!
  this.props.dispatch(search(value));
}
```

Tenéis todo el código disponible en la rama redux-thunk del proyecto.