FLUX, MOBX Y REDUX

Jose Juan Navarro Giner

Flux

Fig. 1 - Logo de Flux

Origen

Flux se trata de una arquitectura para el **manejo y flujo de los datos** en una aplicación web en el **Front-End**. Fue ideada por **Facebook,** con el fin de sustituir al patrón **MVC,** debido a un problema que se les presentó al tener una comunicación bidireccional entre los modelos y los controladores, haciéndoles muy difícil poder depurar y rastrear errores.

Objetivo

El patrón Flux es un **patrón de arquitectura de software**, utilizado principalmente para la construcción de Single-Page Applications, es decir, **aplicaciones web de una sóla página**. Se enfoca principalmente en el manejo del flujo de datos **a través de la aplicación,** separando la lógica de negocio de la interfaz del usuario.

El principal **objetivo** de Flux es proponer una arquitectura en la que el **flujo** de datos es **unidireccional**, en el cual los datos viajan por medio de **acciones** y llegan desde un ***Store*** desde el cual se actualizará la vista de nuevo.

Funcionamiento

El patrón Flux consta de **cuatro elementos principales:**

1. **Acciones:** son objetos que describen algo que ocurre en ese momento en la aplicación. Un ejemplo puede ser la entrada del usuario o la respuesta del servidor.
2. **Dispatcher:** se trata del objeto que recibe las acciones y se encarga de enviarlas a sus correspondientes Stores.
3. **Stores:** son aquellos objetos que contienen el estado de la aplicación y la lógica de negocio.
4. **Vista:** es la interfaz del usuario de la aplicación.

En el diagrama de la Fig. 2 podemos observar un diagrama que resume este funcionamiento.

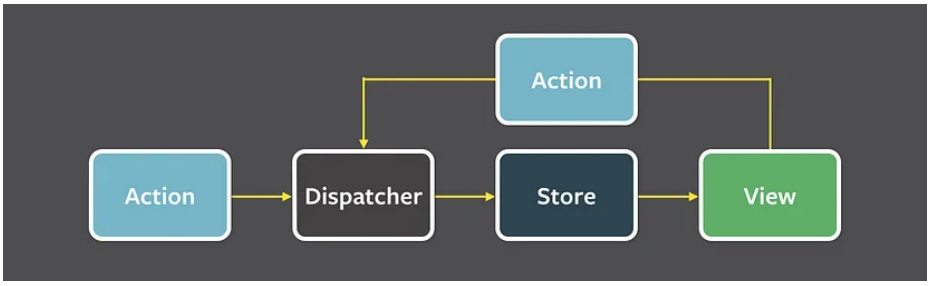

Fig. 2 - Funcionamiento de Flux

Redux

Fig. 3 - Logo de Redux

Origen

La librería **Redux** de JavaScript es una librería de código abierto que se encarga principalmente del **estado de las aplicaciones**, comúnmente utilizada de forma conjunta con **React** o **Angular**. Fue desarrollada por **Dan Abramov** y **Andrew Clark.**

Mientras Abramov se encontraba desarrollando la primera versión de Redux, Abramov se quedó impresionado con el patrón de **Flux** con la operación **reducer**. Por ese motivo Abramov se puso en contacto con Clark (autor de la implementación de Flux) para colaborar juntos en el desarrollo de Redux. Finalmente, sería en el año 2015 cuando Redux viera la luz.

Objetivo

Como he mencionado anteriormente, Redux rescata las ideas principales de Flux, modelando. El **objetivo principal de Redux** es el de ser capaz de **administrar el estado de la aplicación,** aportando una gran trazabilidad completa a los datos que nuestra aplicación está manejando mediante el *Store.*

Funcionamiento

Los **principios fundamentales** de Redux son los siguientes:

1. **Store**, que como hemos dicho anteriormente, funciona como el pilar de todo.
2. **State,** el cual simplemente es **de lectura**, el cual contiene los datos que se van a manipular en la aplicación.
3. **Funciones que realizan los cambios,** que podemos utilizar mediante el uso de **reducers**, que determina cómo cambia el estado a la aplicación en respuesta a las aplicaciones enviadas al *Store*

Mobx

Fig. 4 - Logo de Mobx

Origen

Mobx se trata de otra de las bibliotecas más extendidas que permiten almacenar el estado de las aplicaciones, así como también lo era **Redux**. Esta nueva biblioteca fue desarrollada por **Michael Westrate** en el año 2015.

Objetivo

La librería de Mobs se implementa en los **principios de la programación reactiva funcional (FRP)**, la cual permite envolver los estados en objetos observables, lo cual facilita la reacción ante cualquier estado en nuestra aplicación.

El **objetivo** principal de Mobx es, al igual que Redux, administrar el **estado de la aplicación**, con la cual podemos desarrollar aplicaciones de React **más eficientes** y poder mantener un código **más limpio y eficiente**. Además, es muy sencillo de integrar con los componentes de React.

Funcionamiento

A pesar de tener **objetivos similares**, la principal diferencia que encontramos en Mobx con respecto a Redux es que **no necesita acciones ni reductores**, ya que estos se pueden dar de una forma más directa y sencilla. De la misma manera, el **estado** de nuestra aplicación puede estar **distribuido en diversos objetos observables**, al contrario que Redux, el cual estaba **centralizado**.

Bibliografía

https://carlosazaustre.es/como-funciona-flux

https://www.linkedin.com/pulse/qu%C3%A9-es-el-patr%C3%B3n-de-dise%C3%B1o-flux-y-c%C3%B3mo-funciona-jorge-arias-arg%C3%BCelles/?originalSubdomain=es

https://es.wikipedia.org/wiki/Redux_(JavaScript)

https://es.survivejs.com/react/implementing-kanban/react-and-flux/

https://digital55.com/blog/cuando-por-que-debo-usar-redux-proyectos-frontend/

https://www.arsys.es/blog/redux-datos-aplicaciones#Patron_de_arquitectura_de_datos_Redux

https://redwerk.es/blog/mobx-vs-redux/#:~:text=La%20alternativa%20m%C3%A1s%20extendida%20es,de%20estado%20de%20la%20aplicaci%C3%B3n.

https://www.linkedin.com/pulse/gesti%C3%B3n-de-estado-en-react-con-mobx-pedro-pe%C3%B1a-garc%C3%ADa/?originalSubdomain=es

https://keepcoding.io/blog/aprende-a-usar-mobx-paso-a-paso/#:~:text=Es%20una%20biblioteca%20para%20el%20estado%20de%20las%20aplicaciones%20React.&text=Con%20MobX%2C%20puedes%20crear%20aplicaciones,integraci%C3%B3n%20con%20componentes%20de%20React.