# NgRx

**Web:** https://ngrx.io/

Framework para el desarrollo de aplicaciones reactivas en Angular, que proporciona una administración del estado implementando el patrón Redux que a su vez está inspirado por el patrón Flux.

NgRx está construido con los principios de RxJS, Subject y Observer.

**NgRx se construye a partir de las siguientes partes**

**Store:** Contiene el estado de la aplicación mantenida en memoria, y es inmutable.

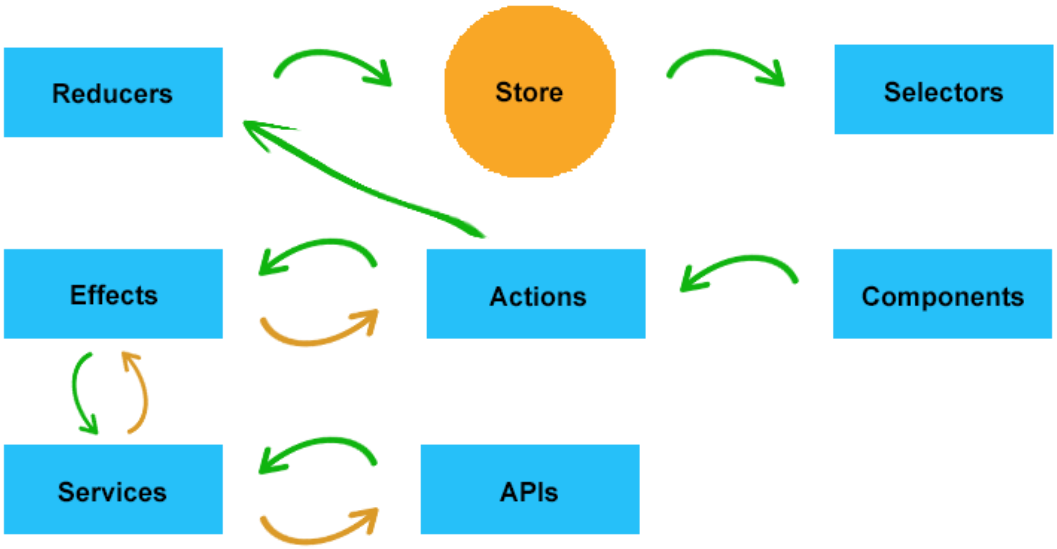
**Components:** Partes de la aplicación que se subscriben al store y obtienen actualizaciones del estado a través de los selectors.

**Reducers:** Toman el estado actual de la aplicación, responden a una acción y crean un nuevo estado inmutable que devolver al store con las nuevas modificaciones aportadas.

**Actions:** Eventos que producen los componentes y servicios, que describen como debe de cambiar el estado del store a través de los reducers.

**Selectors:** Funciones que obtienen porciones de estado del store, a través de estos, podemos también mutar el estado de un componente.

**Effects:** Escuchan las acciones despachadas por un observable, que procesan la respuesta del servidor y devuelven las nuevas acciones al reducer.



## ¿Qué es Flux?

**Web:** https://facebook.github.io/flux/

Arquitectura usada por Facebook para crear aplicaciones web del lado del cliente haciendo uso de un flujo de datos unidireccional, los nodos son independientes.

## ¿Qué es Redux?

**Web:** https://redux.js.org/

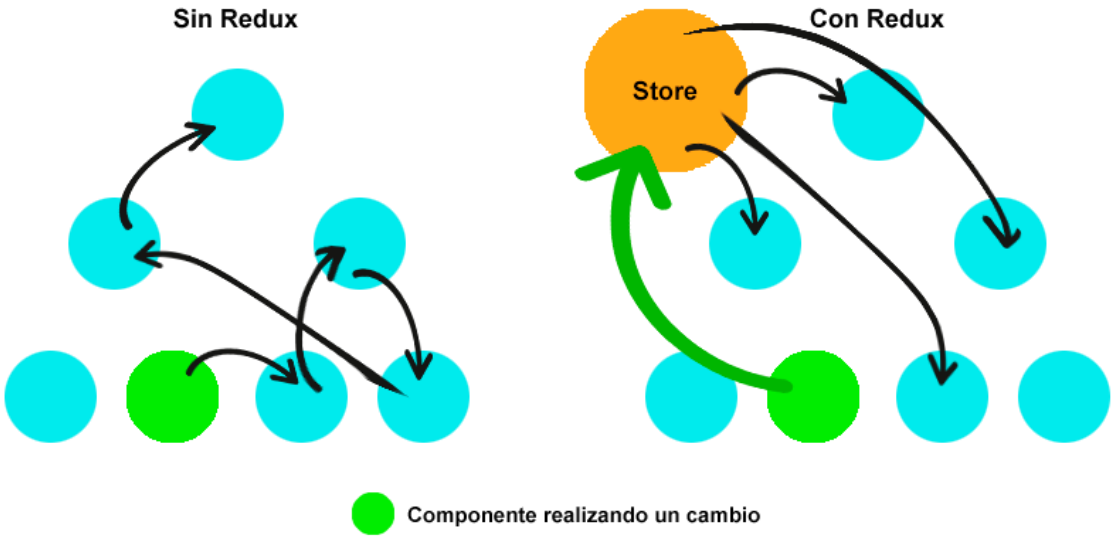
Redux es un contenedor de estado predictivo que nos hace posible utilizar una administración centralizada de estado y lógica de nuestra aplicación.

Aportándonos una arquitectura escalable, con un mayor control en el flujo de datos, gracias a la gestión que Redux tiene de los estados de la aplicación.

Mediante el uso de Redux podemos hacer predecibles los cambios de estado de nuestra aplicación lo que nos facilita el tratamiento de errores mediante herramientas de debugging.

**Principales características:**

**Fuente única de verdad:** Utiliza un único store para almacenar el estado de toda la aplicación.




**Inmutabilidad:** El estado es de solo lectura por lo que no se puede modificar directamente, para modificarlo es necesario realizar los cambios a través de acciones.

**Funciones puras "Reducers":** Usadas para realizar los cambios en los estados, estas funciones reciben dos parámetros, el estado inicial y una acción y dependiendo del tipo de acción realizará una operación u otra en el estado.

## ¿Qué es RxJS?

**Web:** https://rxjs-dev.firebaseapp.com/

RxJS es una librería de JavaScript para programación reactiva que facilita el trabajo con flujos de datos asíncronos bajo el uso de secuencias observables.

## Programación reactiva

Programación orientada al manejo de streams de datos asíncronos y su propagación de cambio en otras palabras el funcionamiento de RxJS es tratar todos los datos como streams (ligados a operaciones I/O como lectura/escritura de ficheros) ya sean eventos, arrays, rangos, promesas, etc… Los streams están representados por secuencias observables y un consumidor llamado observer, el observer no devolverá ningún valor hasta que se active la comunicación entre ambas partes, para activar la comunicación es necesaria establecer una subscripción, el objeto observer está compuesto por tres métodos next, error y complete.

**next:** notifica un valor del stream emitido por el Observable.

**error:** se ejecuta cuando se produce un error.

**complete:** se emite solo cuando el stream de datos ha finalizado.

## observer

Podemos denominar al objeto observer como un sujeto que posee estado. Si el estado cambia, notifica a sus suscriptores el cambio con lo que los subscriptores no tienen qué preocuparse de cuándo se produzca un cambio de estado, puesto que el observer ya se encarga de informar a aquellos objetos que estén subscriptos.

## subjects

Son un tipo especial de observable que permite que los valores sean difundidos a muchos observadores, RxJS dispone de diferentes tipos de Subjects que vienen a cubrir distintas necesidades, la diferencias entre ellos es la manera en la que responden a una subscripción.

#### Tipos de Subjects

**Subject:** Al subscribirnos a un Subject recibimos los siguientes eventos que el Subject emite después de haberse iniciado la suscripción y para finalizar el evento complete.

**BehaviorSubject:** Al suscribirnos recibimos el ultimo evento que ocurrió antes de haberse iniciado la suscripción, igualmente recibimos los siguientes eventos que vayan ocurriendo y para finalizar el evento complete.

**ReplaySubject:** Una vez subscriptos recibimos todos los eventos ya pasados y seguidamente recibimos los siguientes eventos que van ocurriendo durante la subscripción y para finalizar el evento complete.

**AsyncSubject:** Da igual cuando nos suscribamos el esperara a completarse para emitir el ultimo evento y para finalizar el evento complete.

## Operadores

Métodos que se aplican a los observables para modificar su naturaleza o su respuesta, entre otros podemos encontrar create, from, of, map, filter

**create:** crea un observable que recibe una función callback donde definimos el next, el error y el complete.

**from:** crea un observable a partir de un array, un objeto iterable, una promesa...

**of:** Crea una secuencia observable a partir de una lista de argumentos.

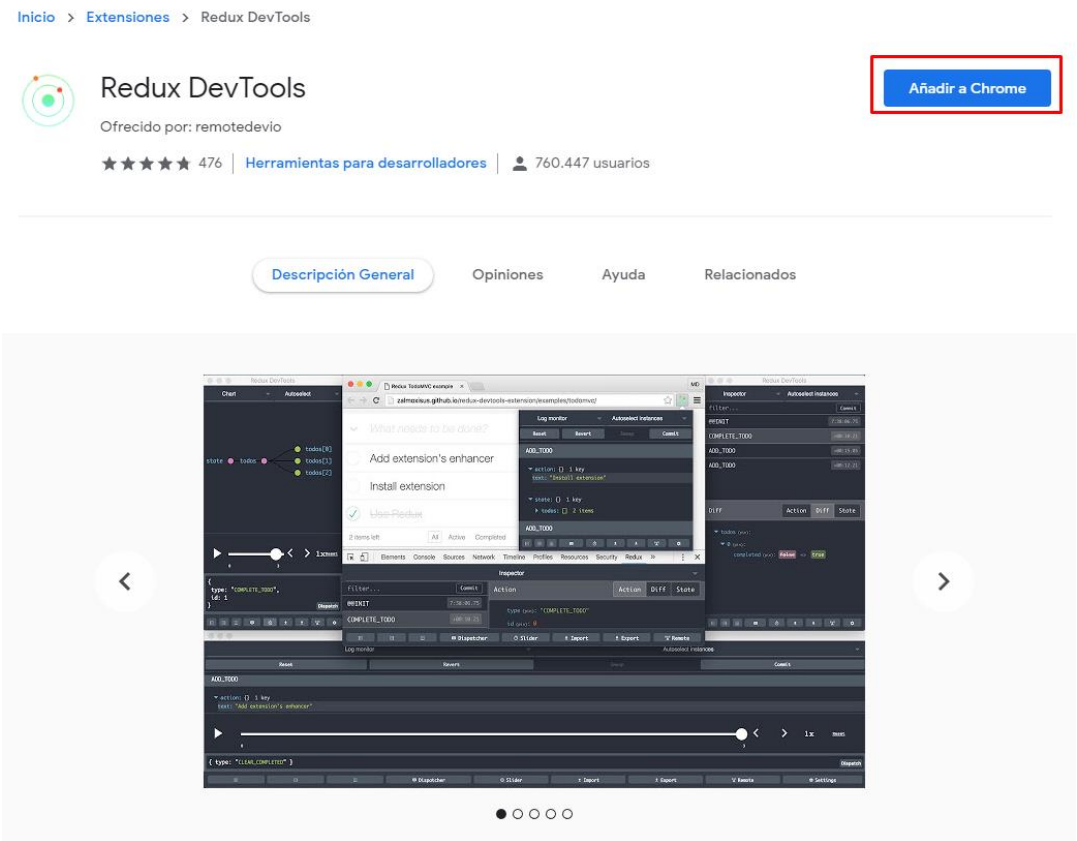
**filter:** filtrar los resultados para que solo se devuelvan los que cumplen la condición indicada.

**map:** Transforma cada elemento de la secuencia Observable. Se puede considerar similar a la función de mapa de Array.

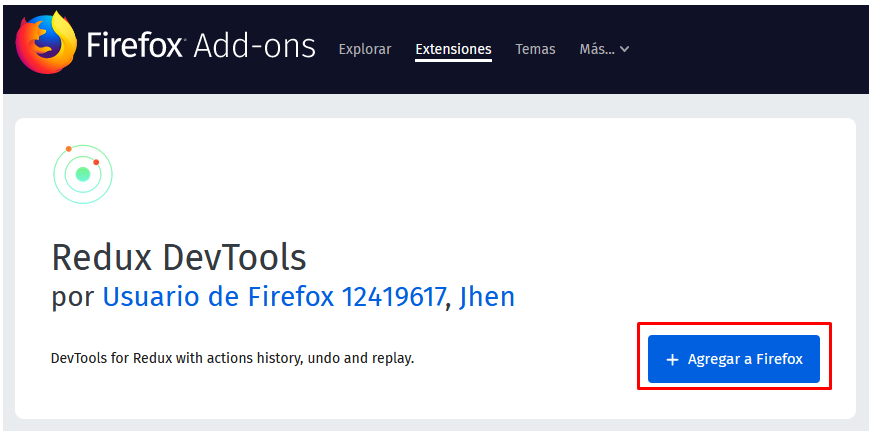
## Instalación de herramientas de debugging y desarrollo

Para el desarrollo de aplicaciones basadas en Redux tenemos la siguiente extensión para Chrome y Firefox que puede sernos de gran ayuda a la hora de depurar nuestras aplicaciones.

https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd

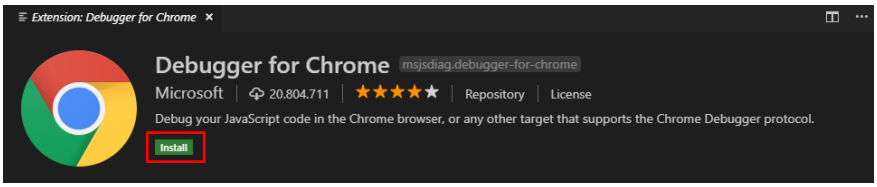


https://addons.mozilla.org/es/firefox/addon/reduxdevtools/

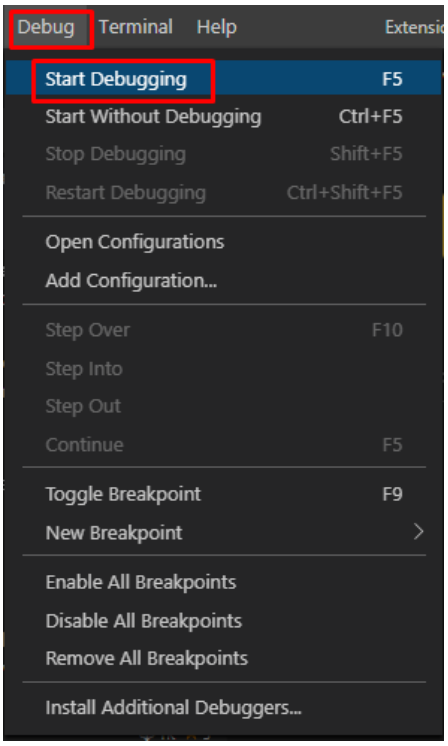



Para debuggear en VsCode tambien podemos instalar el siguiente plugin

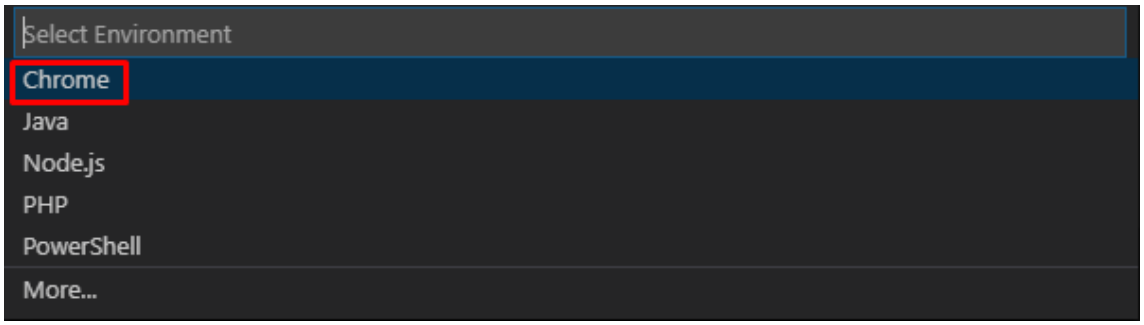
https://marketplace.visualstudio.com/items?itemName=msjsdiag.debugger-for-chrome



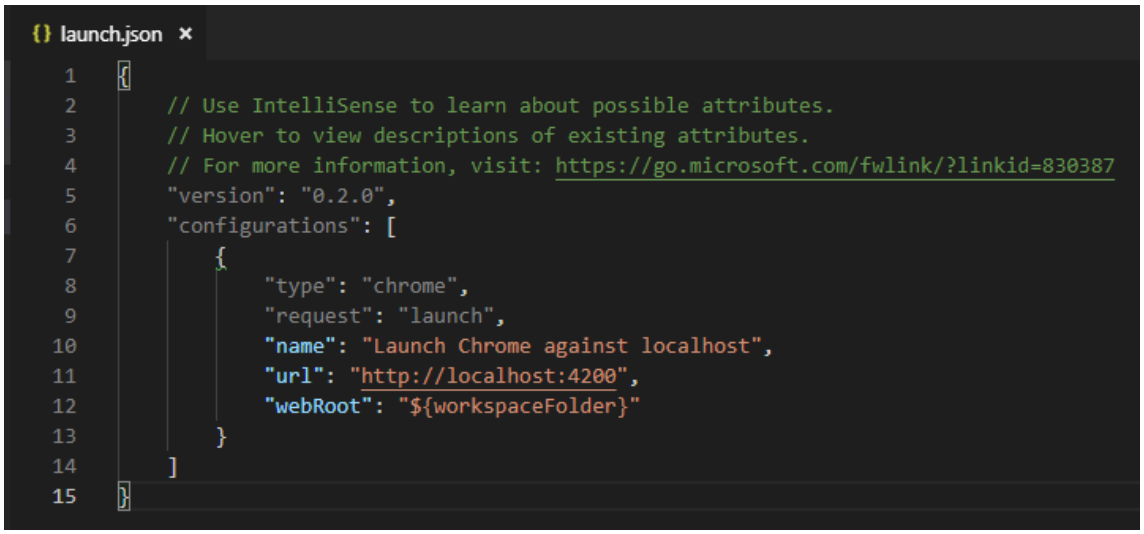

Una vez instalada para iniciarla haremos click en debug y seguidamente Start Debugging.



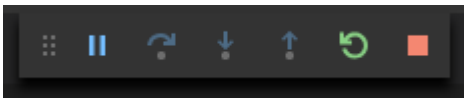
Esperaremos un poco y nos dira con que queremos debuggear.



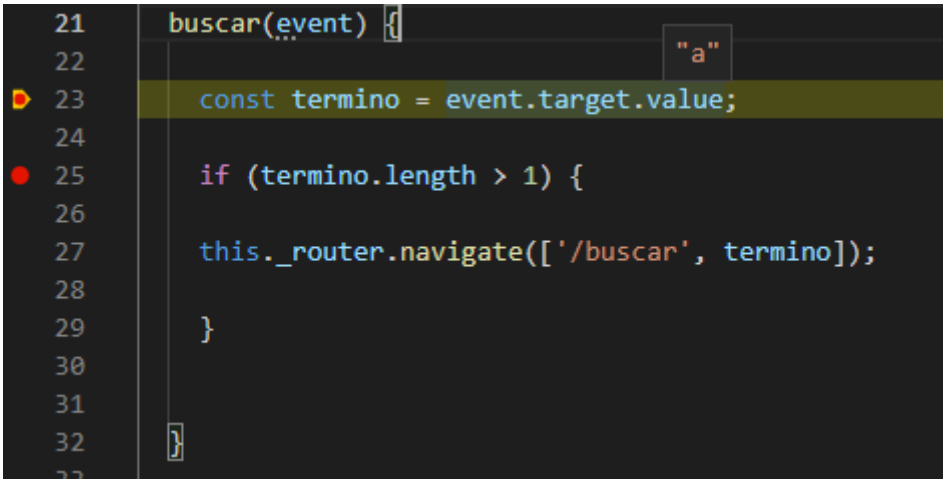

Nos creara un fichero de configuración donde cambiaremos el puerto con el que estemos desarrollamos nuestra aplicación en Angular por defecto es el 4200.




Iniciaremos nuestra aplicación con npm start, y volveremos a arrancar el debbuging y nos apareceran los controles para poder movernos durante la depuración.



Con lo que ya podremos poner puntos de ruptura en los puntos que necesitemos.



Ademas del siguiente plugin para Vs Code que nos ayudara a la hora del desarrollos basados en ngRx y RxJS

https://marketplace.visualstudio.com/items?itemName=Mikael.Angular-BeastCode