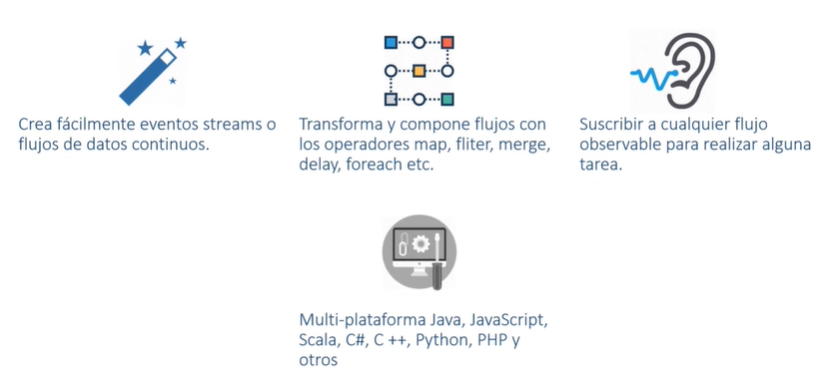
¿Qué es la programación reactiva?

Es un paradigma de programación, en el que se programa de forma encadenada a través de una estructura funcional, orientada a flujos de datos o streams, pero además también permite la propagación del cambio, es decir podemos modificar agregar operadores a nuestros datos y ahí podemos realizar alguna lógica o proceso, estos streams o flujos de datos se trabajan de forma asíncrona.

Características de la programación reactiva:

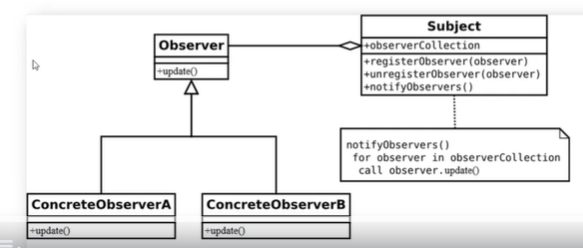
* Crea fácilmente eventos streams o flujos de datos continuos.
* Transforma y compone flujos, con los operadores map, filter, merge, delay, forEach etc.
* Subscribir a cualquier flujo observable para realizar alguna tarea.
* Multiplataforma, porque se pueden encontrar implementaciones diferentes lenguajes como Java, JavaScript, Scala, C#, C++, Python, PHP y otros.
* Trabajan con flujos de datos continuos asíncronos.
* Contrapresión o latencia (non blocking).
* Los operadores reducen notablemente las tareas de transformaciones.
* Pueden ser creados a partir de otras fuentes como stream, listas, intervalos, rangos, etc.
* Son cancelables.
* Puede ser finitos o infinitos.
* Son inmutables.
* Concurrencia hecha más simple.
* Manejo de errores asíncrono.
* Al fallar se puede volver a reintentar.



Patron Observer

La programación reactiva sigue el patrón de diseño Observer; cuando hay un cambio de estado en un objeto, los otros objetos son notificados y actualizados acorde. Por lo tanto, en lugar de sondear eventos para los cambios, los eventos se realizan de forma asíncrona para que los observadores puedan procesarlos. En este patrón se distinguen los siguientes elementos:

* **Subject**: este sujeto sería nuestro observable o Publisher y es el que se encarga de emitir, manejar y procesar el flujo de datos, la propagación del cambio y ante cualquier cambio que se lleve a cabo dentro de nuestro flujo va a notificar a los observadores. Entonces un observable es una cadena de eventos de cambios que pueden suceder a través de nuestro flujo.
* **Observer**: es el elemento que observa esas modificaciones. Cuando el dato se modifica, se notifica al componente que lo está utilizando. Normalmente este Observer necesita suscribirse cuándo quiere empezar a recibir datos, y desuscribirse cuando ya no le interesan más esos datos.



Reactive Streams

[Reactive Streams](https://www.reactive-streams.org/) es una iniciativa para proporcionar un estándar para el procesamiento asíncrono de flujos con contrapresión no bloqueante. Abarca esfuerzos dirigidos a entornos en tiempo de ejecución (JVM y JavaScript), así como a protocolos de red.

Reactor

[Reactor](https://projectreactor.io/) es una biblioteca reactiva de cuarta generación, basada en la especificación Reactive Streams, para crear aplicaciones no bloqueantes en la JVM.

Mono y Flux

**Flux**: es un es Publisher que emite 0 hasta N elementos.

**Mono**: es un Publisher igualmente pero solo emite 0 o 1 elementos.



Manejo de excepciones en un flujo reactivo con Reactor

Existen 2 categorías para el manejo de excepciones en un proyecto Reactor:

* Categoría 1: Recuperarse de una excepción.
  + **onErrorReturn**()
  + **onErrorResume**()
  + **onErrorContinue**()
* Categoría 2: Tomar una acción en una excepción y relanzar la excepción.
  + **onErrorMap()**
  + **doOnError()**