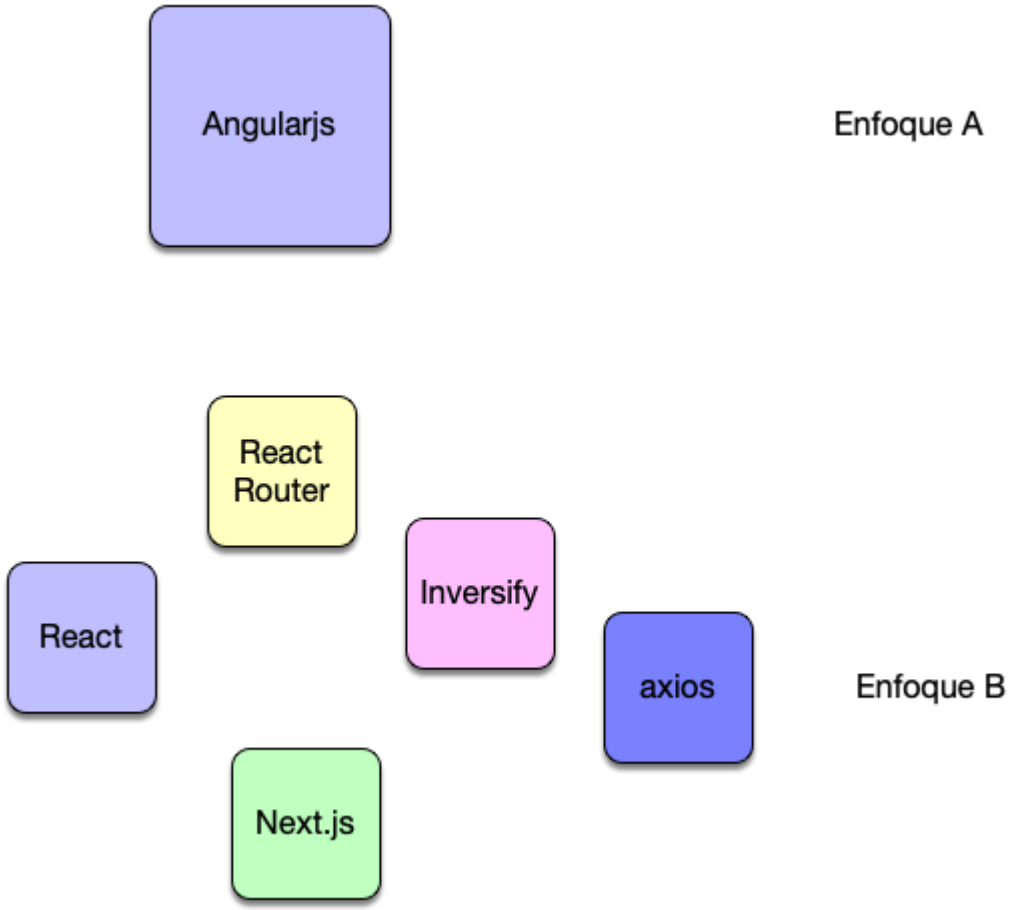
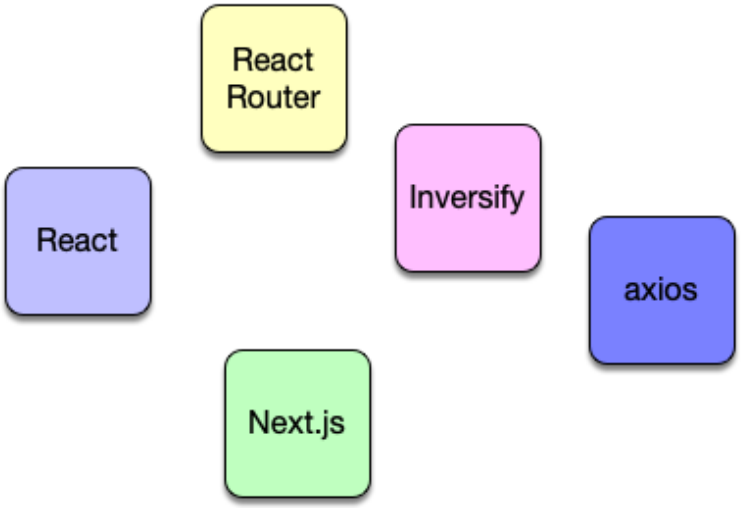
¿Arquitecturas FrameworkLess? . Hoy en día todos usamos frameworks cuando desarrollamos aplicaciones . Unos usamos Spring otros usan Angular , otros ASP.NET MVC y otros Struts etc. Los framework facilitan sobremanera la forma de trabajar . Ahora bien no nos olvidemos que cuando usamos un framework de alguna manera asumimos su forma de trabajar y no nos podemos salir de ahí. Esto genera de alguna forma una dependencia y no se trata de una dependencia sin importancia.

La clave aquí es cuanto tiempo nos aguantará ese framework vivo y sobre todo cuantos años es el ciclo de vida de nuestra aplicación en el mercado. Esta también es una pregunta importante. Para responder a estas preguntas aparecen enfoques que intentan minimizar el uso de frameworks . Esto es a lo que hace referencia una Arquitectura FrameworkLess. Por ejemplo nosotros podemos usar como framework de cliente AngularJS. O podemos combinar varias tecnologías para hacer algo similar . Por ejemplo usar React ,ReactRouter , InversifyJs y Axios. Ambos enfoques son muy diferentes.



Frameworkless y Tiempo

En principio nos puede parecer mejor usar un Framework sobre todo si necesitamos la mayoría de sus capacidades. La gran pregunta que nos tenemos que hacer es si este framework aguantará el paso del tiempo . Muchas veces pensamos que sí , pero nos podemos encontrar con sorpresas. Por ejemplo Angular.js ya ha muerto y ha sido substituido por Angular como framework de referencia basado en TypeScript. Parecía algo imposible pero la realidad es tozuda y hoy por hoy poca gente quedará que desarrolle sobre AngularJS. Eso sí quedan las aplicaciones… muchas aplicaciones y muchos mantenimientos. Algo que si tenemos que migrar a nuevas tecnologías habrá casi que empezar de cero.



¿Cómo elegir una opción?

El uso de librerías normalmente no nos genera una dependencia a la hora de trabajar tan fuerte como la que genera un framework . Por ejemplo cambiar una librería que gestiona la inversión de control por otra suele ser sencillo de abordar. Así pues el uso de librerías nos puede ayudar en situaciones en las cuales sepamos que el ciclo de vida del producto es muy largo y aceptemos que vamos hacia una arquitectura FrameworkLess. O por el contrario podemos seguir queriendo usar un Framework en vez de tener que tener que integrar todas y cada una de las librerías. ¿Cómo podemos elegir?. Muchas veces se nos dice que lo mejor es elegir un framework que tenga una gran aceptación en la comunidad. Puede parecer la mejor opción pero hay que darse cuenta que el software también se mueve por modas y ha

veces algo que tiene una tracción muy fuerte al principio puede desinflarse en poco tiempo. ¿Qué cosas son importantes a la hora de elegir un framework? . Para mi las más importantes son:

1. Genera un cambio de paradigma : Es decir pasamos de programar de una forma a otra en la cual la productividad , o la extensibilidad se ven mejoradas.
2. La comunidad lo apoya: Muchos programadores lo usan y el proyecto se encuentra muy activo.
3. Es un framework ya maduro: Quizá no le hemos utilizado pero ya tiene un tiempo en el mercado y esta más que probado
4. Esta apoyado por un fabricante: Es decir un fabricante o conjunto de fabricantes lo apoya de esto pueden ser ejemplos Hibernate (RedHad) ASP.NET MVC (Microsoft) Spring (pivotal) , Java EE (Oracle).
5. ¿Es de cliente o de Servidor? : Esta es una pregunta también importante ya que por ejemplo el mundo de JavaScript es mucho mas dinámico que Java o .NET.
   Siempre me ha parecido curioso que la mayoría de las personas no tienen ningún reparo en valorar el punto 1 y el 2 y todos se dan cuenta de ello . Sin embargo muchas veces nos olvidamos del resto de puntos a valorar que son muy importantes y también deben de entrar en la balanza de nuestra decisión. Por ejemplo en .NET los WebForms tienen mas de 15 años y en Java Spring o Hibernate parecido. Usemos todas estas medidas de ponderación para elegir lo mejor posible . Asumiento siempre eso sí que nada esta exento de riesgo.

Otros artículos relacionados

1. Los Frameworks y su lado oscuro
2. Framework vs Libreria dos conceptos importantes
3. React vs Angular 2 , frameworks vs librerias
4. Framework concepto