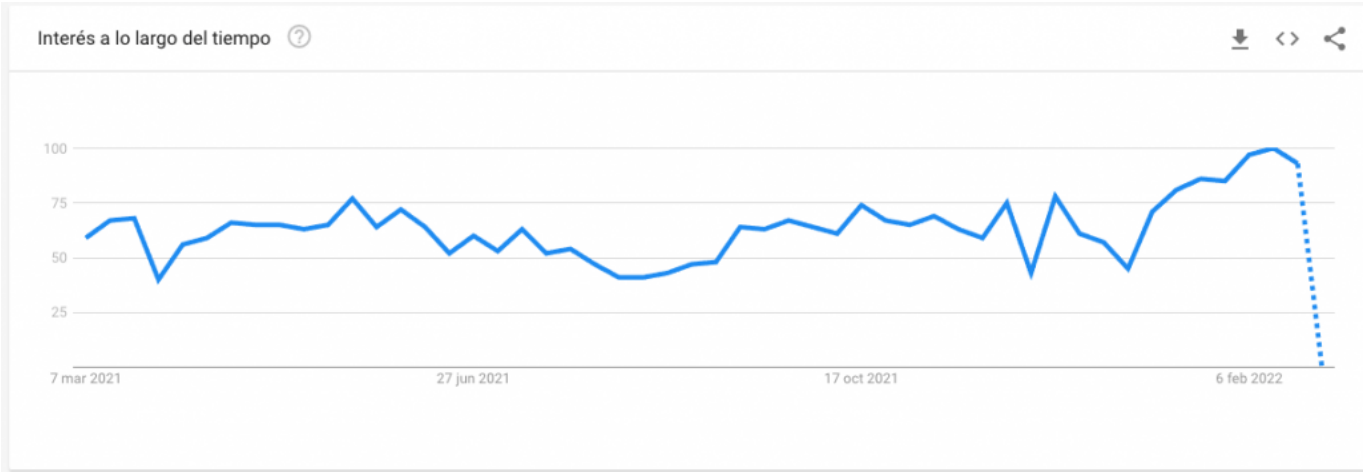
Tabla de Contenidos

¿Cómo elegir un buen Framework? . Esta es una de las preguntas más típicas en el día a día de cualquier desarrollador ya que existen muchos frameworks (demasiados) . Elegir un framework no es una tarea sencilla ya que tenemos siempre muchas opciones. La mayor parte de las veces solemos apoyarnos en que el framework es famoso o esta apoyado por una gran compañía a la hora de elegirlo como nuestro framework de referencia.

Por ejemplo si miramos Google Trends y escribimos Angular o Nest.js nos daremos cuenta que ambos Frameworks tienen un fuerte apoyo por parte de la comunidad.



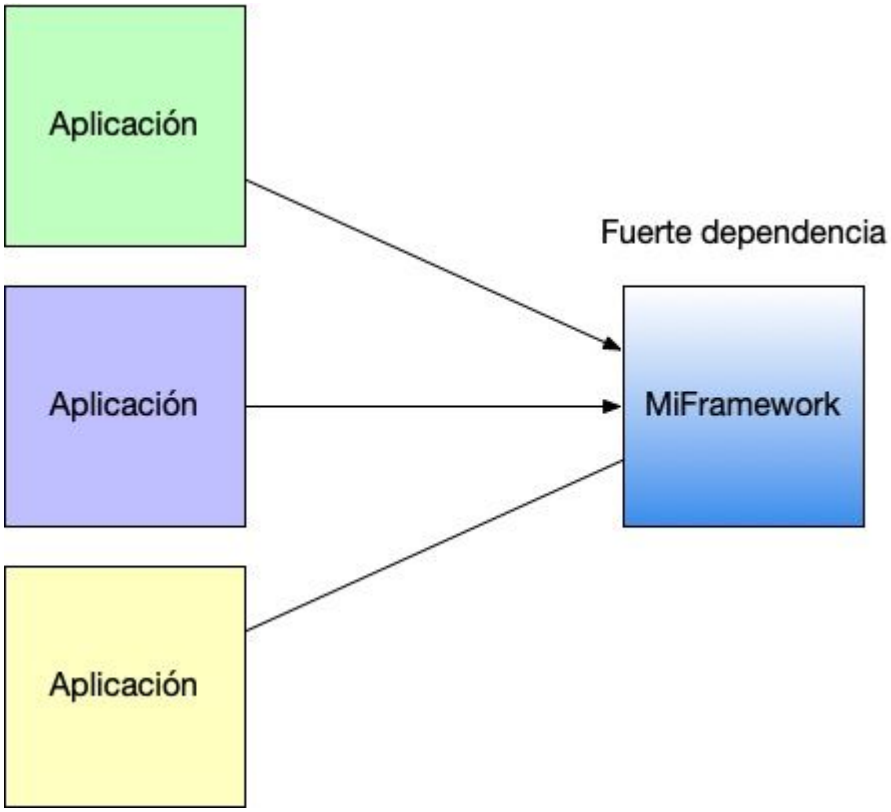
Esta es una de las bazas más importantes a la hora de tomar la decisión sin ninguna duda. Angular esta diseñado y apoyado por Google por lo tanto nos podemos quedar bastante

tranquilos sobre su futuro . Se encuentra bajo el paraguas de una gran compañía. Sin embargo las cosas no son siempre tan sencillas y en muchos casos las decisiones pueden ser delicadas. ¿Porque? . Pues porque normalmente cuando decidimos usar un framework de alguna forma nos casamos a una forma de trabajar muy muy concreta generando dependencias.

Framework y Dependencias

Cuando nosotros decidimos usar un Framework lo hacemos guiados por una serie de ventajas que tiene usarlo. Estas ventajas suelen ser muy comunes entre todos ellos.

- Aumentar la productividad
- Reforzar las buenas practicas y obligarnos a seguirlas . Un ejemplo es el patrón MVC.
- Evitar errores y malas prácticas a todos los niveles
- Generar una cultura común de trabajo entre los desarrolladores.
  Así pues todo son ventajas al usar un framework , lamentablemente hay un problema quedamos ligados al framework y dependemos de una forma muy profunda de él a nivel de desarrollo de aplicaciones .

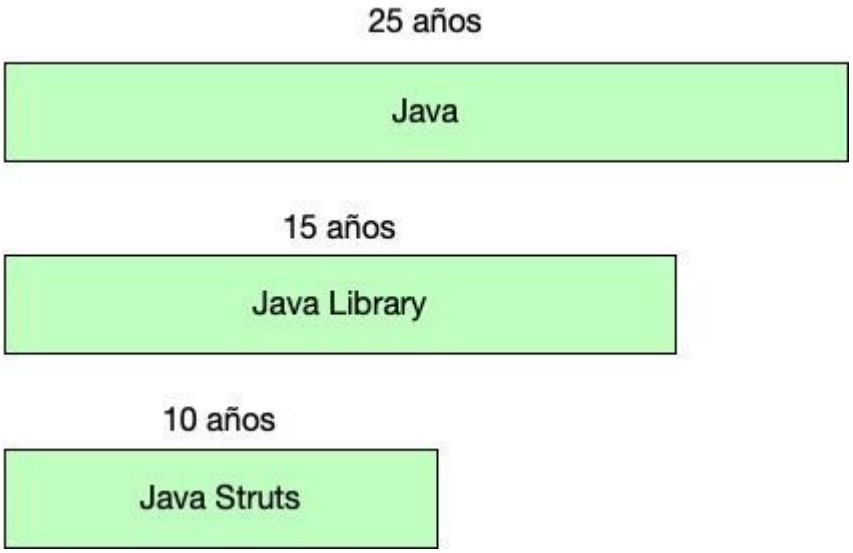


Por lo tanto nos es obligatorio acertar en la toma de decisiones sobre qué frameworks añadimos a nuestra mochila. Vamos a hablar un poco más a fondo de ello.

Frameworks y ciclo de vida

[ihc-hide-content ihc_mb_type=”show” ihc_mb_who=”4″ ihc_mb_template=”1″]

La primera pregunta que tenemos que hacernos es cual es el ciclo de vida de un Framework .Normalmente los ciclos de vida se ordenan de mayor a menor entre lenguaje , librería y framework.



Es muy difícil que una empresa decida cambiar de lenguaje de programación. Si esta contenta con el uso de una librería es posible que la siga usando durante 10 o 20 años sino hay una alternativa claramente superior. Un ejemplo puede ser JUnit . Sin embargo el framework es otra cosa , su ciclo de vida puede ser más corto y esto nos obliga a estar mucho más atentos sobre nuestras decisiones.

Frameworks de Cliente y Frameworks de Servidor

El ciclo de vida de un framework depende bastante de si es un framework de Servidor o es un Framework de Cliente (Javascript normalmente) . Los frameworks de servidor pueden tener un ciclo de vida de 10-25 años .

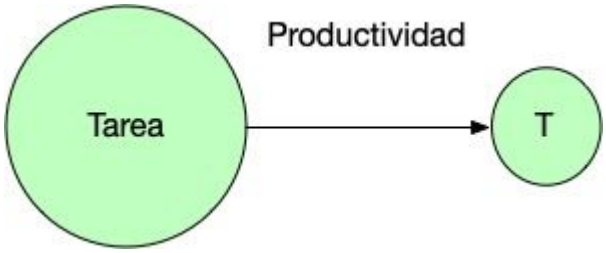
Algo que no ocurre con los frameworks de cliente ya que pueden tener un ciclo de vida más reducido de 5 a 10 años. Mi experiencia es que es más importante definir y pensar muy a detalle cuales son los frameworks de lado servidor que los frameworks de lado Cliente. Ambos son importante pero el lado servidor tiene un mayor peso.

Framework y puntos a tener en cuenta.

Normalmente la gente cuando habla de frameworks , se apoya mucho en lo que he comentado anteriormente “el framework es conocido” , lo uso y punto . Lamentablemente esta no es una buena política a la hora de tomar decisiones. Vamos a ver una serie de ideas a tener en cuenta.

Productividad vs funcionalidad

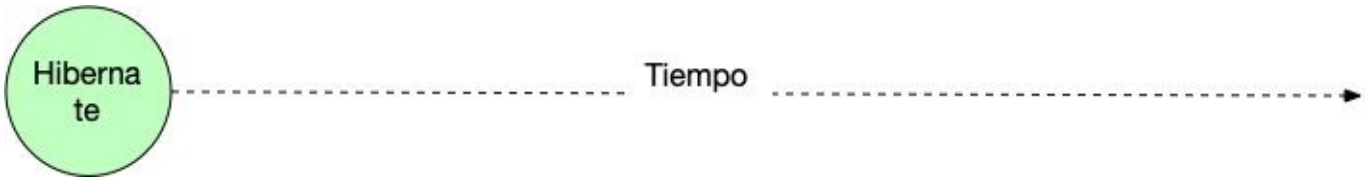
Muchas veces los desarrolladores confunden el concepto de “productividad” con el de “funcionalidad” . El concepto de productividad hace referencia a realizar una tarea en menos tiempo .



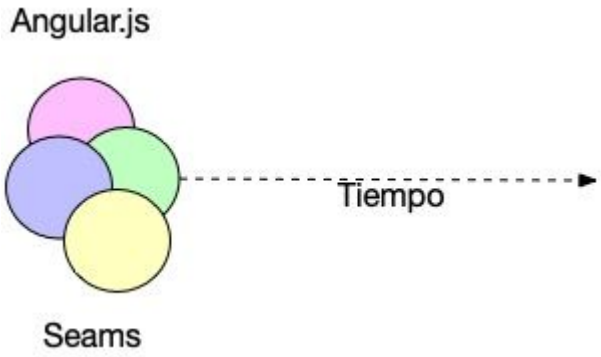
El concepto de funcionalidad hace referencia a que es capaz de realizar más cosas ambos conceptos no siempre están ligados .

Por ejemplo Hibernate es un framework que aumenta de forma considerable nuestra productividad ya que persiste de forma automática los objetos en la base de datos. Sin embargo no hace mucho más que eso , se encarga de persistir. En cambio un framework como Angular es capaz de crear una aplicación complemente funcional vistas , servicios, router etc. Tiene una gran capacidad a nivel de funcionalidad.

Cuanto menos funcionalidad aborde el framework su ciclo de vida será mayor



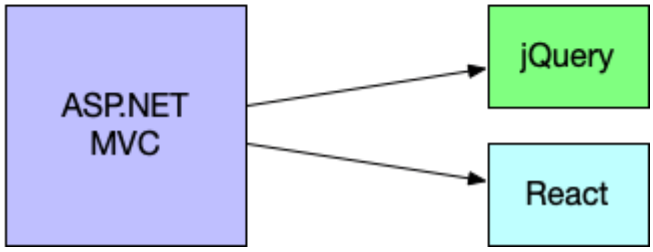
Por ejemplo Hibernate que solo aborda la parte de persistencia contra base de datos ya tiene unos 20 años de vida y esta en plena forma. Cuanto mayor funcionalidad aborde un framework y más tareas realice más probable es que aparezca una solución que lo haga de una forma más óptima en el futuro próximo.



Un paso famoso fue el de JBoss Seam parecía un framework. que sería capaz de abordar una gran funcionalidad y al final no tuvo exito.

Frameworks e Integración

Un framework debe ser capaz de soportar el trabajar en equipo con otros Frameworks de tal forma que se pueda desarrollar una funcionalidad superior. Por ejemplo Hibernate se integra con Spring Framework. Ambos se integran y generan una solución superior. Un caso de un framework que era muy dificil de integrar con otros es del de ASP.NET WebControls . Era tan cerrado que aunque aumentaba de forma fuerte la productividad al final dejo de usarse a favor de ASP.NET MVC que facilitaba de forma más clara la integración con React o JQuery.



Framework y Fabricantes

Uno de los últimos puntos a tener en cuenta es que los fabricantes apoyan a los frameworks que construyen y amplian su ciclo de vida . ¿Qué paso con Struts? . Era un framework construido por un pequeño equipo de desarrolladores cuando migraron a la versión 2 y cambiaron su estructura , dejo de tener éxito y al no estar respaldado por ningún fabricante perdió músculo y desapareció. Este punto es clave para entender el ciclo de vida tan largo que han tenido Hibernate (RedHat) o Spring (VMWare). o ASP.NET MVC (Microsoft)

[/ihc-hide-content]

Otros artículos relacionados

- Java Collections Framework y su estructura
- Framework vs Libreria dos conceptos importantes
- Nestjs un Spring Framework en TypeScript

- Curso Angular