[Introducció 1](#_Toc99351011)

[Conceptes a analitzar 1](#_Toc99351012)

[Conclusions 2](#_Toc99351013)

# Introducció

Abans de començar amb la part més tècnica de VUE i ara que ja sabem de Front, farem un treball de cerca i anàlisis. Intentarem conceptualitzar i entendre els termes bàsics amb els que VUE treballa. D’aquesta manera quan ens fiquem en la part tècnica serà tot més fàcil d’aplicar. És molt important que feu aquesta part, sinó serà molt fàcil que us perdeu. Amb els frameworks front entrem en un nou paradigma

# Conceptes a analitzar

Escriu una petita descripció per cadascun d’aquest conceptes i si s’escau, afegeix algun link que hagis trobat d’interès:

* ¿Qué es un Frameworks?

Un **framework** es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software. Incluyen elementos que optimizan la eficacia en el desarrollo de aplicaciones.

* ¿Qué es VUE?

Vue.js es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página.

* Indica una diferencia de Vue con otros Frameworks.

En cuanto a la curva de aprendizaje con otro frameworks se refiere, Angular es el más difícil de aprender. Ya que necesitas aprender Typescript y el tema de inyección de dependencias, añade cierta complejidad a su aprendizaje.

React es más fácil de aprender, pero su sintaxis puede resultar compleja para ciertas personas con conocimientos básicos o medios.

Vue, es más sencillo de aprender, su sintaxis no resulta compleja y tiene una curva de aprendizaje muy asequible, por lo que para adentrarnos en el mundo del desarrollo frontend, es uno de los recomendados por multitud de expertos.

Por otro lado, tanto Angular como React, cuentan con el apoyo de grandes compañías, Google en el caso de Angular y React en el caso de Facebook.

VueJS no tiene apoyo de grandes compañías, solo de la comunidad, pero sí que hay empresas de reputado prestigio como sponsors, siendo su comunidad desarrolladores una de las más activas.

Por otro lado, existen frameworks dentro del propio Vue.js, frameworks dentro de frameworks, que permiten añadir o alcanzar nuevas funcionalidades, existen unos cuantos e incluso derivaciones del propio Vue.js (forks).

* Templates

**Vue**. js utiliza una sintaxis de **template** basada en HTML que le permite vincular de forma declarativa el DOM renderizado a los datos de la instancia de **Vue** subyacente. Todas las **templates** de **Vue**. js son HTML válidas que pueden analizarse mediante navegadores compatibles con especificaciones y analizadores de HTML.

* Renderizar por pantalla.

Un nodo DOM virtual le dice a **Vue** qué **renderizar** en la **pantalla**, que se convertirá en un nodo DOM real. El **DOM** es una estructura en forma de árbol que se crea una vez que el navegador parsea una archivo HTML, contiene todos los elementos que deben ser renderizados en la página

* Reactivitat de los datos.

Los sistemas reactivos son aquellos que mantienen una interacción constante con su entorno. Son sistemas que permiten el cambio de estado interno por medio de eventos (externos o internos) que manejan acciones cuando son accionados. Su propia palabra lo dice: son sistemas que reaccionan al cambio.

* Data – Binding.

En un proceso de vinculación de datos, cada vez que se modifican los datos, se refleja automáticamente en los elementos vinculados a los datos. El término enlace de datos también se utiliza en los casos en que cambia una representación externa de datos en un elemento y los datos subyacentes se actualizan automáticamente para reflejar este cambio.

* Componentes.

Un **componente** en **Vue**. js es la forma de reutilizar una instancia de **Vue** tantas veces como sea necesario. Mediante los **componentes** podemos crear nuestros propios elementos html, cada uno con sus propiedades y métodos particulares que fácilmente podemos reutilizar en toda la aplicación.

* Hooks - Ciclo de vida de los Componentes.

En código, cada una de estas fases se manifiesta como una especial llamada **hook** de ciclo de vida. Esta función es autoejecutada por **Vue** cuando el componente se encuentra en la correspondiente fase, realizando todas las tareas que tengamos en su interior.

* Routing.

En **Vue**, **vue**-**router** es la biblioteca de enrutamiento oficial del lado del cliente que proporciona las herramientas necesarias para asignar los componentes de una aplicación a diferentes rutas de URL del navegador.

* Interpolación

Las **interpolaciones** y directivas en **VueJs** 2 son la manera de trabajar con las variables de nuestros componentes reactivos. En primer lugar la **interpolación** usa la sintaxis «mustache», lo que quiere decir que encierra entre dobles llaves el código que evalua y concatena.

* Directivas

En **Vue**. js, una **directiva** es el término usado para referirse a algunos atributos especiales, identificados con v- , que le indican a **Vue**. js que debe realizar ciertos cambios en un elemento del DOM, cada vez que la expresión asociada con dicha **directiva** cambie.

* Eventos

Los **eventos** o events son la contraparte de los props, es decir, sirven para pasar información desde el componete hijo al padre. El ejemplo típico de uso es de cuando queremos notificar de un suceso en el componente hijo.

* Computed properties, Watchers

Las propiedades computadas o **computed** properties son una característica muy interesante de **Vue**. Básicamente una computada es una variable, la diferencia con las Variables de **Vue** es que las computadas normalmente transforman la variable o hacen algún tipo de cálculo antes de devolverla.

* Filters

Un filtro en **Vue** es una función que se va a ejecutar ANTES de pintar un dato en una vista en **Vue**, es decir, normalmente se usan para formatear datos.

* Props

Los **props** no solo sirven para para pasar información desde un componente padre a un componente hijo, también puedes pasar **props** a una ruta cualquiera siempre y cuando estén activados. Con eso ya le decimos al **vue** router que esa ruta puede recibir **props** desde fuera.

* $emit

$**emit**() emites el evento. Dentro de la función tienes que poner el nombre que le quieres dar al evento, eVue JS ≫ s decir, si por ejemplo pones this. $**emit**("hover"); luego para usarlo tienes que llamar al evento @hover

* Slots

Los **slots** son un mecanismo de **Vue** JS que sirve para insertar contenido HTML dentro de los componentes. Es decir, con los props puedes pasar objetos y variables javascript a los componentes y con los **slots** puedes insertar contenido HTML dentro de otros componentes.

* SPA - (Single Page Application)

Una **single**-**page application** (**SPA**), o aplicación de página única, es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios, como si fuera una aplicación de escritorio.

* VUE-CLI

**Vue CLI** es una librería que permite usar comandos en la consola del ordenador para automatizar ciertas tareas. Muchos frameworks como Angular o React también tiene este tipo de interfaces para consola.

* AXIOS

**Axios** es un cliente HTTP basado en promesas que funciona tanto en el cliente (navegador) como en el servidor (por ejemplo, Node. js).

# Conclusions

Que és el que més t’ha sorprès/agradat de VUE?

Que comparant amb react i angular, vue ha estudiat les característiques més importants de cada framework e implementar-les.

Que és el que menys t’ha agradat?

Que no conec la majoria de mètodes que utilitza.

Pren nota d’alguna curiositat sobre VUE, que hagis après en el teu estudi.

Que vue ha realitzat una labor de depuració, reduint el codi innecessari. Això fa que el pes de la aplicació es redueixi i sigui més fàcil d’aprendre que els altres dos. Angular pesa 500+kb, react 100kb i vue 80kb.