FRAMEWORK Vue.js

2º DWEC DAW

IES Ingeniero de la Cierva

INTRODUCCIÓN

- Vue.js es un framework que se enfoca en la construcción de las interfaces de usuario.

- Sólo trabaja con la capa de vista de la aplicación, es decir todo lo relacionado con el usuario. No hace referencia al modelo o base de datos al menos por si sólo.
- Como su propia definición indica, es un **framework progresivo,** es decir, se diseño desde su base para que se adopte incrementalmente
- Diseñado para que sea fácilmente integrado con otros proyectos o librerías pero al mismo tiempo capaz de crear aplicaciones muy potentes que se actualizan al mismo tiempo que el usuario interacciona con la aplicación.

1.- INSTALACIÓN

- La forma en la que empezaremos la configuración de Vue.js es a traves de su CLI.
- 1.- Para ello es necesario instalar node.js con anterioridad, lo podemos hacer desde su página web.
- 2.- Una vez instalado, abrimos una terminal y ejecutamos las siguientes instrucciones
- npm install -g vue-cli
- La primera instrucción instala la interfaz de comandos de vue mediante el gestor de paquetes de node.
- 3.- Una vez instalado el cli, nos dirigimos al directorio donde queremos tener el proyecto e iniciamos un proyecto vue mediante la siguiente instrucción
- $ vue init webpack myapp
- .

1.- INSTALACIÓN

- 4.- La instalación realiza una serie de preguntas, en ellas podemos dejar la opción por defecto y contestar que no a la instalación de componentes adicionales para una instalación mas simple.
- 5.- Entramos en el directorio que se ha creado en la anterior instrucción
- cd myapp
- 6.- Instalamos todos los paquetes node necesarios, y que se encuentra en el archivo package.json, medinate la siguiente instrucción:
- npm install
- 7.- Finalmente ejecutamos el servidor y una vez en funcionamiento se abrirá un navegador automáticamente en el que visualizaremos la página de inicio de Vue
- npm run dev

2.- PRIMEROS PASOS

En el proyecto creado hay que tener en cuenta 3 archivos principales:

- **src/main.js**: Creación del componente Vue y su configuración. Observaremos que importa y carga el componente principal **App.vue**
- **src/App.vue**: Componente principal del proyecto que carga el sub-componente Hello.vue.
- **src/components/Hello.vue**: Primer componente del proyecto.

Cuando hacemos referencia al término **componente**, se entiende como una parte especifica de una aplicación que realiza una funcionalidad determinada y que se puede extraer de la composición general. Los componentes nos ayudan a dividir una aplicación relativamente grande en trozos más pequeños para facilitar el trabajo de desarrollo.

2.- PRIMEROS PASOS

Estudiando un componente como App.vue nos damos cuenta de que se compone de 3 etiquetas diferenciadas y que componen cada componente que vayamos a crear en un futuro:

- <template></template>
- <script></script>
- <style></style>

2.- PRIMERO PASOS

- Lo primero que haremos será eliminar todo aquello que no haga falta en el componente principal App.vue.
- Además crearemos un nuevo componente en src/components que llamaremos Animales.vue y que cargaremos dentro de App.vue.
- Finalmente el App.vue debe tener este aspecto:

2.- PRIMEROS PASOS

- El primer paso en este tutorial es eliminar todo aquello que no haga falta en el componente principal App.vue. Además crearemos un nuevo componente en **src/components** que llamaremos **Animales.vue** y que cargaremos dentro de App.vue. Finalmente el App.vue debe tener este aspecto

```
<template>
 <div id="app">
 <animales></animales>
 </div>
</template>
 <script>
import Animales from './components/Animales'
export default {
 name: 'app',
  components: {
   Animales
  }
}
</script>
<style></style>
```

2.- PRIMEROS PASOS

Si abrimos la consola del inspector del navegador al cargar el proyecto observamos que hay un error que nos dice que falla al tratar de cargar el componente, en este caso Animales.vue.
Esto es debido a que Animales.vue está vacío.
Arreglamos esto añadiendo los 3 elementos principales y una configuración básica, por lo que Animales.vue queda así:

```
<template>
 <div class="animales">
 {{titulo}}
 </div>
</template>

<script>
export default {
 name: 'animales',
 data () {
  return {
   titulo: "Animales",
  }
 }
}
</script>

<style></style>
```

2.- PRIMEROS PASOS – enlace de datos

En Animales.vue observamos una de las principales características de Vue.js.
Dentro de el componente *template*, hay un trozo de html básico y el elemento *{{titulo}}*, este elemento esta asociado con la parte del script del componente en el que si nos fijamos en la linea '*titulo: "Animales",'* comprenderemos que al recargar el proyecto en el navegador ahora aparecerá el texto *Animales*.
Intuimos así que se ha establecido un enlace entre {{titulo}} y la asociación de la variable con la palabra Animales en el momento de renderizar la página. De esta manera podemos asociar valores dinámicamente en una pagina web mediante el poder de javascript.
Si ampliamos nuestra plantilla de Animales.vue, con el siguiente código:

```
<template>
 <div class="animales">
  {{titulo}}
 <br>
  <input type="text" v-model="animal">
 <br>
  {{animal}}
 </div>
</template>

<script>
export default {
 name: 'animales',
 data () {
  return {
   titulo: "Animales",
   animal: "Perro",
  }
 }
}
</script>

<style></style>
```

2.- PRIMEROS PASOS – BUCLES

Ahora observamos que hay un elemento nuevo elemento en elemento script y que devuelve una nueva variable, *animal: "Perro"*.
De esta manera en la parte de *template*, se enlazara con *{{animal}}*.
Aparte, tenemos un nuevo elemento html input, que contiene un atributo llamado **v-model** y que esta enlazado a *animal* (*v-model="animal"*), este atributo es propio de Vue.js e indica que cualquier dato que insertemos en el **input** se ligará automáticamente a la variable correspondiente mostrando dinámicamente el cambio en el html en caso de que la variable *{{animal}}* esté presente.

En la parte de script también podemos crear variables complejas como un array de elementos clave elemento que nos pueden ayudar a ordenar nuestros datos. Para este ejemplo crearemos el siguiente array de animales:

```
animales: [
 {nombre: "Gato", tipo:"Terrestre"},
 {nombre: "Águila", tipo:"Aereo"},
 {nombre: "Carpa", tipo:"Acuático"}
]
```

Para representar estos datos en la plantilla haremos uso del atributo **v-for** de Vue.
Añadiremos una lista html debajo de los datos anteriores:

```
<ul>
 <li v-for="elementoAnimal in animales">
  Nombre: {{elementoAnimal.nombre}}, Tipo: {{elementoAnimal.tipo}}
 </li>
</ul>
```

3.- ALGUNAS CARACTERÍSTICAS

Vue.js cuenta con su propia extensión **.vue** y es justo en esos archivos donde podemos escribir tanto el código javascript como el html, simplemente debemos exportar un objeto con la lógica que necesitemos y un trozo de código html que esté dentro de una etiqueta template.

```
<template>
    <!-- código html -->
</template>


<script>
export default {
    //lógica del componente
}
</script>
```

3.- ALGUNAS CARACTERÍSTICAS

Aunque sea muy sencillo de utilizar, también es muy potente, dentro de cada componente tendremos disponibles una serie de funcionalidades común en todos ellos, paso a detallar algunas.

- **data**: devuelve un objeto con los datos disponibles al inicializarse un componente, aunque éstos podrán ser alterados mdiante **ounted.**
- **mounted**: da acceso al ciclo de vida de un componente que nos permite por ejemplo hacer llamadas a una api para inicializar datos.
- **methods**: ofrece las funciones que estarán disponibles para el componente en cuestión.
- **props**: útil para añadir datos de entrada a un componente y así poder reutilizarlo.
- **watch**: para poder estar actualizando en tiempo real datos de nuestro componente.
- **computed**: si queremos crear campos calculados, por ejemplo el nombre y el apellido del usuario, desde aquí podemos definir una nueva propiedad la cual será una variable nueva en nuestra plantilla

4.- DESARROLLAR UN COMPONENTE CON Vue

Cómo puedes ver, **Vue.js** está enfocado a trabajar con **componentes**, una cosa muy interesante es que también tenemos la posibilidad de utilizar **two-way data bindings** gracias a v-model, que es una directiva que se encarga de alterar los datos de ida y de vuelta, algo muy útil cuando trabajamos con campos formularios, por ejemplo.

Algo muy importante también a tener en cuenta es que Laravel lo incorpora de base en la creación de un proyecto y es muy recomendable utilizarlo en determinados casos para generar componentes que después podamos utilizar en blade.

4.- DESARROLLAR UN COMPONENTE con Vue

- Crea un directorio dentro de src llamado **components** y dentro un archivo llamado **HolaMundo.vue** el cual será nuestro primer componente, así que vamos a añadir el siguiente código.

```
<template>
    <div class="hola-mundo">
        <button v-on:click="toggleText()">ToggleText</button>
        <p v-if="showText">{{ text }}</p>
        <p>{{ reverseText }}</p>
    </div>
</template>
<script>
    export default {
        data() {
            return {
                text: 'Hola Mundo',
                showText: false
            }
        },
        methods: {
            toggleText() {
                this.showText = !this.showText;
            }
        }
    }
</script>
```

4.- DESARROLLAR UN COMPONENTE con Vue

Este fichero está formado por un trozo de html dentro de unaetiuqueta template y un trozo de javascript que exporta un objeto, con **data** inicializamos los datos del componente y con **methods** definimos métodos del componente.

Para poder utilizar nuestro componente, vamos a abrir el archivo src/App.vue y vamos a modificarlo para que quede cómo el siguiente código.

```
<template>
<div id="app">
 <img src="./assets/logo.png">
<h1>{{ msg }}</h1>
<hola-mundo></hola-mundo> <!-- utilizamos nuestro componente -->
</div>
 </template>
```

```
<script>
 import HolaMundo from './components/HolaMundo.vue'; // <---- importamos nuestro componente
export default {
name: 'app',
components: { HolaMundo },
data () { return {
msg: 'Welcome to Your Vue.js App'
} } }
 </script>

<style>
 #app {
 font-family: 'Avenir', Helvetica, Arial, sans-serif;
text-align: center;
color: #2c3e50;
margin-top: 60px;
}
h1, h2 {
 font-weight: normal;
}
a {
color: #42b983;
}
 </style>
```

4.- DESARROLLAR UN COMPONENTE con Vue

Así de sencillo es utilizar los componentes que vayamos desarrollando, simplemente son etiquetas html.
Aunque este ejemplo es el más sencillo que vamos a ver, vamos a mejorarlo: Para ello vamos a abrir de nuevo el componente **HolaMundo.vue** y vamos a modficarlo.

```
<template>
<div class="hola-mundo">
    <button v-on:click="toggleText()">ToggleText</button>
    <p v-if="showText">{{ text }}</p>

    <hr />
    <ul>
        <li v-for="user in users" v-on:click="showUser()"> {{ user }} </li>
    </ul>

</div>
</template>
```

```
<script>
    export default {
        data() {
            return { text: '',
                    showText: false,
                    users: ['juan', 'andrés', 'luís', 'pedro']
                   }
        },
        methods: {
            toggleText() {
                this.showText = !this.showText;
            },
            showUser(user) {
                console.log(user);
            }
        }
    }
</script>
```

Así de sencillo es recorrer datos en Vue.js 2, inicializamos un array de usuarios y con v-for los recorremos, cómo puedes ver también añadimos un evento click para que al pulsar en cada usuario podemos ver en la consola la información del usuario pulsado.

4.- DESARROLLAR UN COMPONENTE con Vue

Imagina que en lugar de querer tener los usuarios inicializados con data quieres obtener la información de una api, en ese caso debemos utilizar **mounted** que es el punto de acceso para hacer cosas cuando el componente se inicializa. (HolaMundo.vue)

```
<template>
    <div class="hola-mundo">
        <button v-on:click="toggleText()">ToggleText</button>
        <p v-if="showText">{{ text }}</p>

        <hr />
        <ul>
            <li v-for="user in users" v-on:click="showUser(user)">
                {{ user }}
            </li>
        </ul>
    </div>
</template>
```

.

```
<script>
    export default {
        data() {
            return {
                text: '',
                showText: false,
                users: []
            }
        },
        methods: {
            toggleText() {
                this.showText = !this.showText;
            },
            showUser(user) {
                console.log(user);
            }
        },
        mounted() {
            this.users = ['juan', 'andrés', 'luís', 'pedro'];
        }
    }
</script>
```

4.- DESARROLLAR UN COMPONENTE con Vue

Para alterar los datos en los dos sentidos (two way binding) vamos a añadir un input haciendo uso de v-model en nuestra template.
Añadir a HolaMundo.Vue

```
<input v-model="text" />
```

De esta forma, todo lo que escribamos en esa caja de texto modificará el valor de text, y ese dato será alterado tanto en la template como en el componente, que es lo importante.