

Introducción a VueJS (Parte I)

¿Qué es VueJS?

Competencias

- Identificar características de VueJS.
- Conocer e identificar elementos constituyentes del framework.
- Conocer alternativas a VueJS.
- Conocer patrón de diseño MVVM.

Introducción

En este capítulo vamos a comenzar a conocer VueJS, un framework JavaScript que ha tenido un gran crecimiento durante el último tiempo, tanto en adopción de parte de desarrolladores como en empresas que lo han tomado como parte de su stack tecnológico, en este capítulo llegaremos a conocer en mayor profundidad su composición, conocer sus partes, arquitectura, que empresas lo utilizan, parte del funcionamiento del mismo, bajo qué patrón trabaja, entre otras, empecemos.

¿Qué es VueJS?

Vue (se pronuncia 'view') fue pensado para construir interfaces de usuario, adopta parte del patrón MVVM, y está pensado para ser adoptado incrementalmente. Es decir, se puede ir incorporando progresivamente a nuestros proyectos dependiendo el alcance y las necesidades del proyecto.

Así mismo, VueJS Fue construido sólo para enfocarse en la capa de la vista (pero con su propio modelo y directivas que nos ayudarán a esta tarea), con facilidad para integrarse con otros proyectos u otras librerías. A partir de este punto, es donde VueJS saca ventaja de sus "competidores", dado que la simpleza con la que se puede lograr grandes resultados es bastante menor comparado con otros frameworks o librerías como ReactJS o AngularJS.

¿Por qué usarlo?

Una de sus principales ventajas de VueJS con respecto a otros frameworks y/o librerías para el front-End, es la curva de aprendizaje, la cual, es mucho menor en comparación con AngularJS o ReactJS (por mencionar solo algunos). Pero, eso no quiere decir que sean malas alternativas, solo que el proceso de aprendizaje es un poco diferente de VueJS.

Algunas de las características más importantes que posee son:

- Renderizado declarativo

Podemos declarar elementos dentro del DOM utilizando templates, lo que nos brinda una sintaxis bastante familiar si es que alguna vez hemos trabajado con HTML.

Para comenzar, crearemos un archivo "index.html" (puedes crearlo en tu escritorio o donde consideres pertinente), en este archivo incluiremos lo siguiente:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Iniciando con VueJS</title>
</head>
<body>
  <div id="app">
    {{message}}
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
</body>
</html>
```

En el código mostrado anteriormente, debemos poner especial atención en la llamada realizada en la etiqueta "script", la cual, nos permite utilizar VueJs en nuestra aplicación (en el siguiente capítulo entraremos en mayor detalle con esto) con la llamada a la CDN en el atributo src de la etiqueta.

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>.
```

Por lo tanto, esta sola línea de código permitirá incorporar VueJS a nuestro Proyecto desde la CDN.

Posteriormente dentro de la etiqueta "body", podremos observar la siguiente fracción del código:

```
<div id="app">
  {{message}}
</div>
```

Donde se encuentra un elemento DIV (una etiqueta de html) con un atributo ID, este id es bastante particular, debido a que en este preciso ID del elemento DIV perteneciente a HTML, se le indicará a VueJS donde "cargarse", (luego entraremos en mayor detalle con esto).

Ahora bien, dentro del elemento `<div>` se puede observar una sintaxis que pudiera resultar extraña con doble llaves o doubles moustaches `{{ }}` (los paréntesis de llaves), donde se realizará una llamada a "message" (una variable en JavaScript que mostrará el contenido que tenga asignado, ya sea una cadena de caracteres, números, verdadero o falso, u otro tipo de dato).

Justo después del cierre de la etiqueta "html", o antes del cierre de la etiqueta "body", se debe incluir lo siguiente:

```
<script>
const app = new Vue({
  el: '#app',
  data: {
    message: 'Hola Desafío Latam!!!'
  }
});
</script>
```

En el script anterior, se puede observar cómo se crea una instancia de VueJS con las palabras reservadas "new Vue" y se asigna a una constante llamada "app" (este nombre puede ser cualquier otro nombre). Ahora bien, el contenido de la instancia es un objeto que contiene elementos particulares como "el" y "data". En el caso del elemento "el", es una propiedad del objeto que tiene como valor el String '#app', es decir, referencia al id utilizado en la etiqueta `<div>` en el html, mientras que "data" posee como atributo otro objeto con la propiedad "message" y el valor 'Hola Desafío Latam!!!', en otras palabras, "data" se utilizara para inicializar las variables principales que utilizaremos en nuestro proyecto con VueJS.

Con esto se está indicando a VueJs donde "cargarse" como se indicó anteriormente, luego, la instancia de VueJS se encargará de buscar el id "app" en el html y mostrará el "message" a través de la variable definida previamente en el objeto "data".

Finalizado todos los pasos anteriores, el código final en el archivo index.html debería quedar de la siguiente manera:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Iniciando con VueJS</title>
</head>
<body>
  <div id="app">
    {{message}}
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script>
    const app = new Vue({
      el: '#app',
      data: {
        message: 'Hola Desafío Latam!!!'
      }
    });
  </script>
</body>
</html>
```

y al hacer doble clic en el archivo index.html o ejecutarlo de la manera que consideres pertinente, podrás observar lo siguiente en el navegador web de tu preferencia:



Imagen 1. Primera ejecución Vue.

Nació del aprendizaje de otro framework

Creado por Evan You (@youyuxi en Twitter), un ex trabajador de Google, quien plantea lo siguiente:

"Me imaginé que podía extraer la parte que realmente me gustaba de Angular y construir algo realmente ligero y rápido"

Componentes

Son una de las características más relevantes y poderosas que posee VueJS. Las cuales, permiten extender elementos HTML para poder encapsular código reutilizable. En otras palabras, los componentes son elementos personalizados a los que el compilador de VueJS les añade comportamiento. La definición de componentes dependerá completamente de lo que necesitemos desarrollar, un componente puede ser algo complejo como una cuenta regresiva o algo simple como un mensaje. Por consiguiente, para trabajar con componentes se deben llamar de la siguiente manera dentro:

```
<div id="app">  
<mi-componente></mi-componente>  
</div>
```

Pero previamente se debe registrar y generar la instancia para cada componente a utilizar, como se muestra a continuación:

```
<script>  
  // registro de un componente  
  Vue.component('mi-componente', {  
    template: '<div>Un componente a medida</div>'  
  });  
  // crear la instancia principal  
  new Vue({  
    el: '#app'  
  });  
</script>
```

Por lo tanto, en el script anterior se puede observar cómo se inicia un componente en VueJS, el cual, se debe realizar con las palabras reservadas `Vue.component('nombre del componente', {objetos del componente})`. Dentro de los objetos del componente, se puede apreciar la utilización de un "template", Este objeto estará conformado por la estructura html que renderiza VueJS sobre el elemento principal con la etiqueta creada para el componente en sí, en este caso: `<mi-componente></mi-componente>`. Entonces, si modificamos nuestro archivo "index.html", agregando todo lo anteriormente mencionado, quedaría de esta forma:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Iniciando con VueJS</title>
</head>
<body>
  <div id="app">
    <mi-componente></mi-componente>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script>
    // registro de un componente
    Vue.component('mi-componente', {
      template: '<div>Un componente a medida</div>'
    });
    // crear la instancia principal
    const app = new Vue({
      el: '#app'
    });
  </script>
</body>
</html>
</script>
```

Al ejecutar, veremos este resultado:



Un componente a medida
Imagen 2. Primer componente.

¿Quiénes utilizan VueJS?

La gran adopción que ha tenido VueJS en el último tiempo se muestra en la confianza que algunas compañías han puesto en este framework, tanto a nivel nacional como internacional.

Internacionalmente, tenemos algunas empresas:

- Facebook, donde se utiliza en partes de su news feed.
- Netflix, para aplicaciones internas.
- Adobe, para el producto 'Portfolio'
- Xiaomi, en el diseño de productos.
- Alibaba, en el core de su sitio, es uno de los principales sitios en adoptar Vue.
- Nintendo, en la plataforma "my nintendo"

Así como en muchas otras empresas a nivel mundial, donde la adopción de VueJS ha ido creciendo en el último tiempo, especialmente de parte de los desarrolladores, quienes han incluso llegado a superar en cantidad de estrellas "stars" a ReactJS (una librería bastante utilizada) en github (plataforma utilizada por desarrolladores para almacenar y administrar código), marcando una elección en cuanto a preferencias de desarrollo.

En el ámbito nacional no nos quedamos atrás, existen varias empresas que han ido adoptando VueJS:

- yapo.cl, donde han ido haciendo uso parcial de Vue en proyectos puntuales.
- biobio.cl, radio online que utiliza VueJs en su página principal.
- autopía.cl, también en su página principal utilizan VueJS.
- seguros falabella, también en su web principal.

Estos son sólo algunos ejemplos de empresas en que VueJS ha sido utilizado, existen muchas más donde se está utilizando y creciendo día a día.

Alternativas a VueJS

Existen muchos frameworks y librerías JavaScript, en primer lugar, para hablar de estos es necesario definir lo que es una Solicitud de Página Única o Single Page Application (SPA), brevemente, podríamos definirla como una aplicación de una única página, en la que todas las interacciones se generan sobre la misma página, como las peticiones al servidor (generalmente a una API REST), lo importante de estas, es que aportan una experiencia de usuario simple, donde no ocurrirán recargas que entorpezcan la experiencia de navegación e interacción en el sitio.

Entre las principales alternativas a Vue tenemos:

- ReactJS

Ha ganado mucho terreno en cuanto a uso, el respaldo de un grande de las tecnologías como Facebook, donde fue creado, los ayuda mucho. Bastante demandado por las empresas, es una gran librería para construir interfaces de usuario.

Permite construir SPA o incluso aplicaciones móviles (con React Native), para esto provee de un ecosistema de módulos y complementos para lograr su cometido. Además posee de varias cosas predefinidas (en desarrollo, boilerplate) que nos permite como desarrolladores hacer cosas en poco tiempo.

Uno de los principales objetivos de React, es la de construir aplicaciones web con mayor orden y prolijidad que si fueran creadas exclusivamente con JavaScript o utilizando complementos como JQuery.

- AngularJS

Es un framework creado por Google, y al igual que React, el fin principal objetivo es ayudar a los desarrolladores a construir aplicaciones web de una forma más sencilla y óptima, también nos permite construir SPA. Es uno de los precursores en cuanto a frameworks o librerías orientadas a frontend se refiere, tuvo una gran aceptación en sus inicios, pero uno de los principales reclamos que tiene, es la complejidad en su aprendizaje.

Patrón de diseño MVVM

Como se indicó previamente, VueJS implementa parcialmente el patrón Modelo-vista-modelo de vista o Model View ViewModel (MVVM), donde el primer componente es el modelo, el cual, se caracteriza por encapsular la lógica de la aplicación y el acceso a datos, en esta parte se podría ver una conexión a base de datos, a algún servicio web, validaciones o algunos procedimientos que ejecuten la lógica de negocio, para el modelo, por lo que debe ser indiferente en como se muestra la data, debido a que esta no es de su responsabilidad.

Por otra parte, tenemos la Vista, la cual podemos definirla como el producto que el usuario final ve de nuestro sistema, así mismo, dentro de sus responsabilidades se puede mencionar que la Vista es la encargada de definir la estructura (aparencia de los datos que genera nuestra aplicación), la cual, tiene relación directa con el ViewModel, por consiguiente, esta relación se produce a través de los 'data bindings' (enlaces de datos). En donde un flujo común de trabajo consiste en que la vista le informa al ViewModel de cambios que se hayan producido, para que así el ViewModel tenga que hacer algo con dicha información, y

recíprocamente el ViewModel le indica los cambios que se hayan producido en el modelo o en el ViewModel mismo. Y finalmente, el ViewModel, quien cumple la función de ser un intermediario entre Vista y Modelo, es decir, encapsula la lógica de la vista de los datos que produce el modelo y tiene tres responsabilidades:

1. Controlar las interacciones que suceden en la vista.
2. Recuperar los datos ingresados en el modelo y enlazarlos con la vista.
3. Notificar a la vista de los cambios que sucedan en él, y de sus propiedades.



Imagen 3. Diagrama de Responsabilidades del MVVM

Uno de los grandes beneficios de trabajar con este (u otro patrón), es la posibilidad de trabajar en partes separadas y complejas de forma completamente independientes.

Por una parte podríamos tener a un equipo de programadores back-ends, preocupándose por completo de trabajar en el modelo y toda la lógica de negocios, y por otra a un equipo de front-ends preocupados de trabajar en el ViewModel y parte de la vista, además del equipo de UI/UX quienes determinan la mejor forma de visualizar nuestro contenido.

El diseño de Vue está parcialmente inspirado en este patrón, como convención Vue utiliza la variable 'vm' (ViewModel) para referirse a la instancia de VueJS.

One-way data-binding y mutable data

VueJS está pensado para mutación de datos, si los datos van a ser estáticos, es mejor no utilizar VueJS, dado que esta es una de las principales fortalezas, debido q que Vue está observando siempre los cambios de datos y esperando hacer algo con eso, y eso lo hace mucho más rápido que sus competidores. Por lo tanto, VueJS piensa en la comunicación de componentes padre a hijo como forma natural, para evitar así confusiones en el diseño y manejo de data. Para tareas más complejas nos proporciona manejadores de estado como Vuex y posibilidades de distintas rutas en una sola vista con Vue Router.

Conociendo VueJS, componentes del framework

Competencias

- Identificar elementos propios de VueJS.
- Conocer opciones que nos presenta Vue CLI.
- Conocer formas de instalación de VueJS.

Introducción

Es importante conocer las formas que tenemos para trabajar con VueJS, más allá del desarrollo, tenemos distintas formas de instalar e implementar VueJS en nuestras aplicaciones, por ejemplo, Vue CLI se ha llevado gran parte de los elogios (debido a que es un sistema completo para el desarrollo rápido de aplicaciones con VueJS), pero no es la única forma de instalar e implementar Vue, ya que existe un amplio marco de opciones dado al gran crecimiento y aceptación que ha ido logrando en el último tiempo. De ahí que en este capítulo conoceremos algunos elementos que brinda este framework para resolver diversas tareas.

Formas de instalar Vue en una aplicación web

Para trabajar con VueJS tenemos, a la fecha, tres formas de instalarlo en nuestras aplicaciones:

- Primera forma - CDN: VueJS en su página web oficial mediante la Guía de Aprendizaje [VueJS.org](https://vuejs.org), facilita dos CDN para poder incluir en el proyecto a realizar. Uno de ellos es el de desarrollo. Este debería ser utilizado en todo el proyecto mientras se va construyendo o realizando. Para ello, debes incluir el siguiente enlace con la versión de desarrollo, la cual, incluye advertencias de ayuda en la consola y puede ser utilizada con la extensión [Vue.js Devtools](#) de los navegadores web, como el caso Mozilla Firefox. Nota: una CDN es un conjunto de servidores ubicados en diversas partes del mundo, estos contienen copias de contenido para utilizar, es la forma más simple de referenciar código externo, el objetivo de la CDN es de proveer el contenido para que nuestra experiencia sea más simple. Generalmente las CDN sirven contenido estático (imágenes, vídeos, scripts, css). Por consiguiente, el CDN que debes utilizar es:

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

- Posteriormente, debemos crear un documento denominado index.html, con la estructura normal de todo documento de hipertexto, como se muestra a continuación:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Iniciando con VueJS</title>
</head>
  <body>
  </body>
</html>
```

Luego, en la etiqueta **<body>** de nuestro index.html, se debe crear una estructura con el "id" que será el elemento principal de VueJS. Ejemplo:

```
<body>
  <div id="vm" class="container">
    <h1>{{message}}</h1>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
</body>
```

- Finalizado los pasos anteriores, se debe crear la instancia de VueJS haciendo referencia al id indicado en la etiqueta <div> dentro del <body> del index.html, esto debemos hacerlo con las etiquetas <script> que proporciona html, quedando de la siguiente manera:

```
<script>
  const vm = new Vue({
    el: '#vm',
    data: {
      message: 'Bienvenidos a Vue'
    }
  });
</script>
```

- Al unir todos los pasos anteriores en nuestro documento index.html y agregando un poco de estilo local mediante la etiqueta <style>, el resultado final sería el siguiente:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Iniciando con VueJS</title>
  <style>
    .container {
      width: 100%;
    }
    h1 {
      color: #a5a5a5;
    }
  </style>
</head>
<body>
  <div id="vm" class="container">
    <h1>{{message}}</h1>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script>
    const vm = new Vue({
      el: '#vm',
      data: {
        message: 'Bienvenidos a Vue'
      }
    });
  </script>
</body>
</html>
```

Si observamos la primera etiqueta `<script>` dentro del `body` del documento, se encuentra una url en el atributo `src`, donde se está referenciando a la CDN de Vue, de esta forma se puede comenzar a trabajar con Vue en nuestros archivos.

El resultado de nuestro código anterior, al ejecutar el `index.html` en un servidor local es:



Imagen 4. Bienvenidos a Vue.

- Segunda forma - Mediante gestores de paquetes/dependencias.

NPM ó Yarn: para utilizar esta forma, es necesario tener instalado NodeJS (<https://nodejs.org/en/download/>), (es bastante simple la instalación, sólo debes seguir las instrucciones que se acomodan a tu sistema operativo).

- En la actualidad desarrollando front-end es muy frecuente la necesidad de instalar alguna librería o framework para hacer algo (un cálculo matemático, alguna conexión a una API Rest, entre otras opciones), es así como surgió la necesidad de un gestor de paquetes, uno que ha tomado particular relevancia en el último tiempo es NPM (Node Package Manager), un lugar centralizado donde se encuentran muchos paquetes a descargar para incorporar en nuestros proyectos.
- Yarn es una alternativa a NPM, fue creada en colaboración entre Facebook y Google, la diferencia principal entre uno u otro, es la rapidez (Yarn es más rápido que NPM).

Para instalar Vue en nuestro proyecto, procedemos de la siguiente forma:

```
npm install vue  
#ó  
yarn add vue
```

- Tercera forma - Vue CLI

La interfaz de comandos de Vue, es una herramienta creada por el equipo de desarrolladores para ayudar a facilitar el rápido desarrollo de las apps hechas en Vue. Para instalarlo, previamente debes verificar que posees la instalación de NodeJS y el Administrador de paquetes NPM o Yarn, con los siguientes comandos por la terminal de tu computador:

```
node --version
```

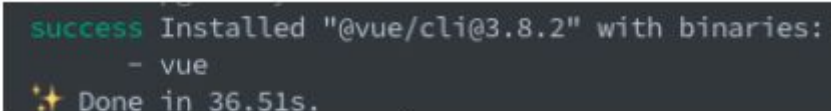
```
npm --version
```

Si ya está disponible NodeJS en tu computador, entonces también entras instalado NPM. De lo contrario, puedes descargar [NodeJS](#), quien incluye en su instalación a NPM, o si gustas puedes instalar el gestor de paquetes [Yarn](#). Todo queda a tu elección.

Una vez realizado los pasos mencionados anteriormente, es momento de instalar Vue CLI en nuestro computador mediante el uso del terminal y los siguientes comandos:

```
npm install -g @vue/cli  
#ó  
yarn global add @vue/cli
```

Al finalizar la instalación, el terminal debería mostrar un mensaje indicando la versión instalada al momento de Vue CLI, la cantidad de paquetes instalados, entre otros mensajes (no te preocupes si el resultado de tu instalación no te da exactamente como la imagen a continuación), solo debes estar seguro que la instalación fue exitosa y la terminal no arrojo ningun error o abortó el proceso de instalación.



```
success Installed "@vue/cli@3.8.2" with binaries:  
- vue  
🌟 Done in 36.51s.
```

Imagen 5. Resultado de la instalación de Vue CLI en el terminal.

Posterior a esto, debemos crear un proyecto con la siguiente instrucción por la terminal (Nota: el nombre del proyecto solo puede contener caracteres compatibles con URL, así mismo, el nombre del proyecto no puede contener letras mayúsculas, de lo contrario la instalación no podrá ejecutarse debido a errores que se mostrarán en el terminal):

```
vue create nombre_proyecto
```

Una vez realizado esto, el terminal preguntará algunas cosas, las cuales iremos detallando paso a paso.

- Setup de una aplicación con Vue CLI y sus características agregadas:

Una vez realizamos la creación de nuestro proyecto con Vue CLI, nos comenzará a preguntar varias cosas (recuerda que el número de la versión de Vue CLI depende de cuando se realizó la instalación en el computador, solo asegurate de poseer la última versión):

```
Vue CLI v3.8.2
? Please pick a preset: (Use arrow keys)
> default (babel, eslint)
  Manually select features
```

Imagen 6. Selección de características en la creación del proyecto con Vue CLI.

En primer lugar el terminal nos preguntará si queremos mantener la configuración por defecto o realizar la instalación del proyecto manualmente, esto es relativo, no existe una receta que debamos seguir siempre, simplemente depende de lo que necesitemos desarrollar en ese momento.

Si seleccionamos la primera opción, el terminal comenzará a ejecutar las instalaciones necesarias que ya vienen por defecto en la opción seleccionada, como se muestra a continuación:

```
Vue CLI v3.8.2
? Please pick a preset: default (babel, eslint)
? Pick the package manager to use when installing dependencies: (Use arrow keys)
> Use Yarn
  Use NPM
```

Imagen 7. Instalación de dependencias con la configuración por defecto de Vue CLI.

Al finalizar el proceso de instalación mediante la configuración por defecto, podremos observar los siguientes resultados en la terminal:

```
success Saved lockfile.  
✨ Done in 15.63s.  
⚓ Running completion hooks...  
  
📄 Generating README.md...  
  
🎉 Successfully created project vuecomponents.  
👉 Get started with the following commands:  
  
$ cd vuecomponents  
$ yarn serve
```

Imagen 8. Finalizada la instalación de las dependencias para el proyecto.

Genial, ya podemos ver el código que generó y desplegar nuestro proyecto, pero la verdad no hicimos mucho, se siente un poco extraño dado que no sabemos realmente que tiene nuestro proyecto.

Si navegamos a la carpeta que nos indica (cd nombre_proyecto) en el terminal, y abrimos nuestro editor de preferencia, veremos algo así:

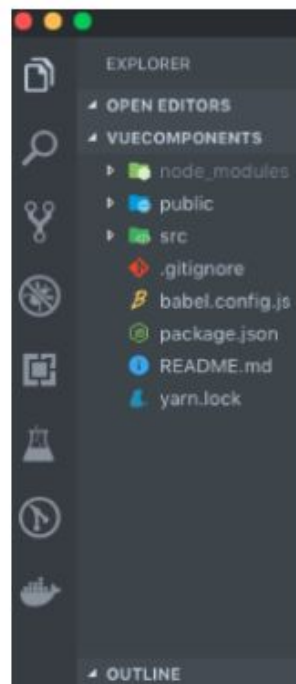


Imagen 9. Árbol de archivos utilizando Yarn para la instalación.

Donde vemos las carpetas que se instalaron y los archivos generados. Igualmente, obtendremos un resultado parecido si la instalación se realiza mediante el gestor de paquetes NPM, como se puede observar a continuación:

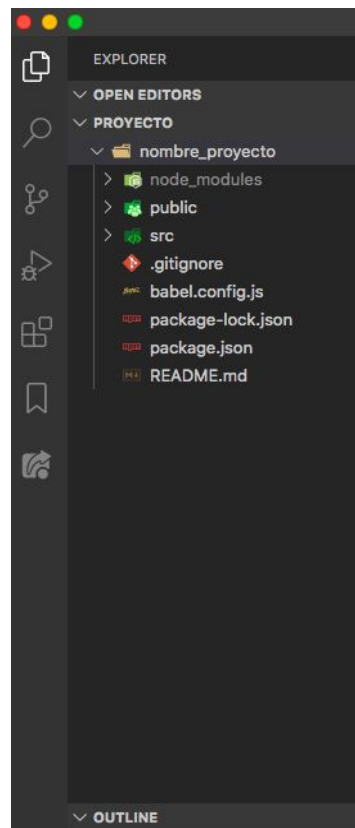


Imagen 10. Árbol de archivos utilizando NPM para la instalación.

Y si seguimos la otra instrucción que nos indica cualquiera de las instalaciones, ya sea por Yarn o NPM en el punto anterior, es decir, si ejecutamos en la terminal 'yarn serve' o 'npm run serve', nos debería aparecer algo como esto:

```
DONE Compiled successfully in 4563ms

App running at:
- Local:    http://localhost:8081/
- Network:  http://192.168.1.186:8081/

Note that the development build is not optimized.
To create a production build, run yarn build.
```

Imagen 11. Despliegue del servidor con la solución para Yarn.

```
DONE Compiled successfully in 5393ms

App running at:
- Local:    Follow Link \(Cmd + click\) :8080/
- Network:  http://192.168.1.37:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Imagen 12. Despliegue del servidor con la solución para NPM.

Donde nos indica la url local y de red donde se está desplegando nuestro proyecto, si accedemos desde nuestro navegador (recomiendo que utilices Chrome o Firefox, por las herramientas de desarrollo principalmente), veremos lo que sigue:

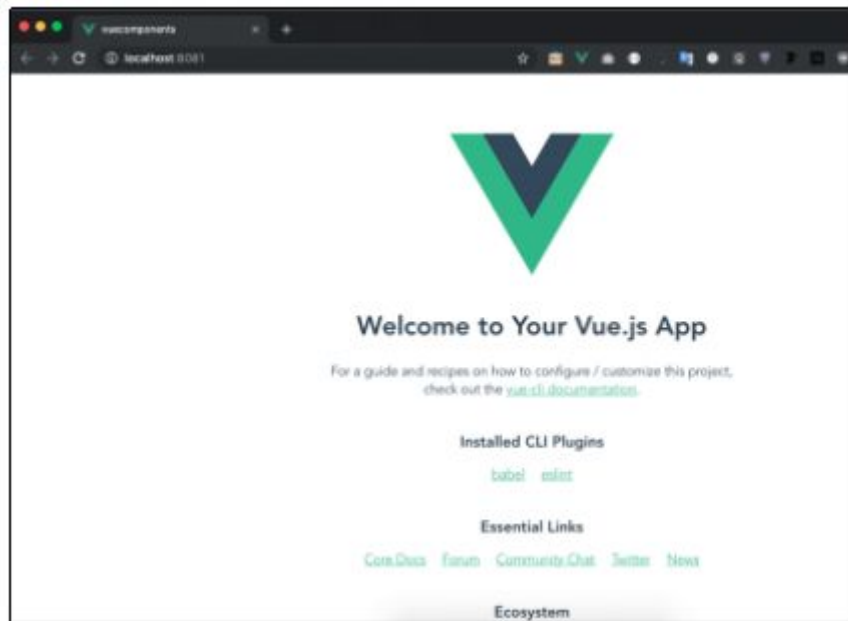


Imagen 13. Instalación por defecto de Vue con CLI.

Felicidades, ya tenemos nuestro proyecto andando y funcionando! (pero aún se siente como si no hubiéramos hecho mucho), revisemos el código.

En la carpeta 'public' veremos un archivo 'index.html', si lo abrimos y vemos dentro de este, veremos esto:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport"
content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title><%= htmlWebpackPlugin.options.title %></title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %>
doesn't work properly without JavaScript enabled. Please enable it to
continue.</strong>
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

Donde muestra por defecto el título de nuestro proyecto en la pestaña del navegador, pero en el código lo realiza mediante la línea: `<%= htmlWebpackPlugin.options.title %>`, quien se encarga de traer y mostrar el nombre de la carpeta de nuestro proyecto (esto lo puedes modificar y agregar el nombre que tu creas conveniente para tu proyecto).

También se puede apreciar un mensaje (en la etiqueta "`<noscript>`"), la cual se activará si es que el usuario de nuestro proyecto tiene desactivado JavaScript en su navegador, pero no vemos realmente el contenido que nos muestra el navegador web previamente al ejecutar el servidor.

Ahora bien, si observamos con detención el código, encontrarás una etiqueta `<div>` con el `id="app"`.

```
<div id="app"></div>
```

Y como se vio en pasos anteriores a esa sección al utilizar el CDN para implementar VueJS en nuestro proyecto, desde esa etiqueta `<div>` estaremos llamando a 'app', donde se hace referencia a lo que programemos en Vue.

Si seguimos buscando en nuestro proyecto, nos encontraremos con una carpeta llamada 'src', la cual contiene varios archivos y carpetas:

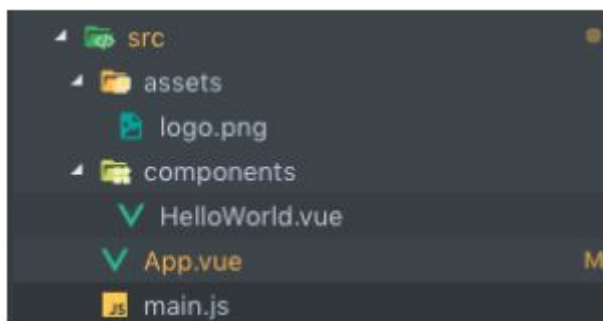


Imagen 14. Archivos del proyecto con Vue.

Por lo que se puede apreciar la carpeta 'assets' donde se manejan las imágenes, en este caso el logo, además la carpeta 'components' donde se crearán los componentes, es decir, componentes Vue, también se puede observar el archivo 'App.vue', y 'main.js', los cuales serán detallados uno por uno a continuación:

- En primer lugar revisaremos el archivo 'main.js':

```
import Vue from 'vue'
import App from './App.vue'

Vue.config.productionTip = false

new Vue({
  render: h => h(App),
}).$mount('#app')
```

En el código anterior, dispuesto en el archivo "main.js" se puede observar la importación de Vue y App ("import ..."), con las estructuras básicas para importar archivos desde las dependencias o de rutas en específico con la sintaxis de ES6 (anteriormente se utilizaba el require de javascript).

Luego, se agrega la configuración donde se indica a Vue que se estará trabajando en modo desarrollo. Posterior una nueva instancia de Vue (" new Vue({ ... }) ") donde se genera una llamada a la función 'render', y se marca el punto de montaje de la aplicación con el id app (" #app").

Por su parte, cuando se revisa en detalle el archivo "App.vue", se encontraran con lo siguiente:

```
<template>
  <div id="app">
    
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

Al analizar el código dispuesto en "App.vue" En primer lugar, se observa la etiqueta "<template>" donde se generará todos los elementos a mostrar (siendo esta una sintaxis bastante similar a html). Luego se tiene la etiqueta "<script>", donde se pueden importar elementos necesarios como otros componentes, algunas librerías específicas (como librerías de internacionalización, conexión a base de datos, entre otras), y elementos propios de Vue (props, datas, methods, lifecycle hooks, entre otros). Finalmente se tienen los estilos propios a ese componente.

Ahora modifica el `<HelloWorld msg="Welcome to Your Vue.js App"/>` cambiando el "msg" por: 'Hola Alumnos de Desafío Latam' Luego de una pequeña modificación, el proyecto se compila automáticamente mientras tengas el servidor activo en el terminal, y así podrás observar el siguiente resultado en el navegador web:

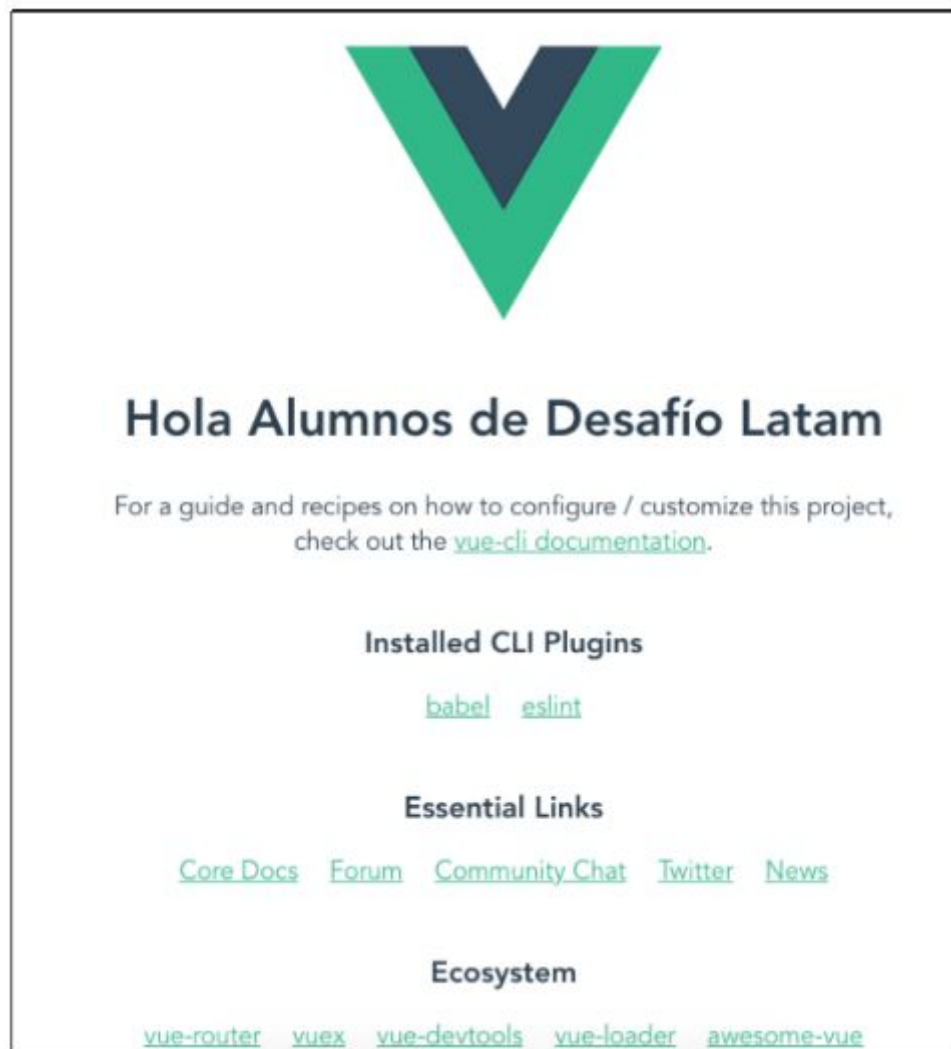


Imagen 15. Default modificado.

Perfecto, ya modificaste parcialmente el componente "App.vue" por defecto. Asimismo, podemos observar que se muestran varios links de interés para visitar en la página en el navegador web, pero si revisamos el código del archivo 'App.vue' no se encuentran codificados por ninguna parte... ¿Entonces dónde están? pista: en el otro componente que importamos.

Correcto, en el archivo 'HelloWorld.vue' se encuentran todos estos vínculos y estilos que componen nuestra aplicación y que podemos observar en el navegador web al ejecutar el servidor y mostrar la página, como se aprecia en la siguiente imagen.

```
-template-
  <div class="hello">
    <h1-{{ msg }}></h1>
    <p>
      For a guide and recipes on how to configure / customize this project, <br>
      check out the
      <a href="https://cli.vuejs.org" target="_blank" rel="noopener">vue-cli documentation</a>.
    </p>
    <h3>Installed CLI Plugins</h3>
    <ul>
      <li><a href="https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-babel"
      target="_blank" rel="noopener">babel</a></li>
      <li><a href="https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-eslint"
      target="_blank" rel="noopener">eslint</a></li>
    </ul>
    <h3>Essential Links</h3>
    <ul>
      <li><a href="https://vuejs.org" target="_blank" rel="noopener">Core Docs</a></li>
      <li><a href="https://forum.vuejs.org" target="_blank" rel="noopener">Forum</a></li>
      <li><a href="https://chat.vuejs.org" target="_blank" rel="noopener">Community Chat</a></li>
      <li><a href="https://twitter.com/vuejs" target="_blank" rel="noopener">Twitter</a></li>
      <li><a href="https://news.vuejs.org" target="_blank" rel="noopener">News</a></li>
    </ul>
    <h3>Ecosystem</h3>
    <ul>
      <li><a href="https://router.vuejs.org" target="_blank" rel="noopener">vue-router</a></li>
      <li><a href="https://vuex.vuejs.org" target="_blank" rel="noopener">vuex</a></li>
      <li><a href="https://github.com/vuejs/vue-devtools#vue-devtools" target="_blank"
      rel="noopener">vue-devtools</a></li>
      <li><a href="https://vue-loader.vuejs.org" target="_blank" rel="noopener">vue-loader</a></li>
      <li><a href="https://github.com/vuejs/awesome-vue" target="_blank" rel="noopener">awesome-vue</a>
    </li>
    </ul>
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>
h3 {
  margin: 40px 0 0;
}
ul {
  list-style-type: none;
  padding: 0;
}
li {
  display: inline-block;
  margin: 8 10px;
}
a {
  color: #42b58c;
}
</style>
```

Imagen 16. Template default de HelloWorld.vue.

En este apreciamos todos los vínculos (utilidades que nos pueden brindar una mejor experiencia de desarrollo y por supuesto la documentación oficial de Vue), además del script que define el nombre del componente y estilos (css). Con esto, se finaliza la creación de un proyecto con Vue CLI bajo la configuración por defecto.

Volviendo un poco atrás, si seleccionamos instalar nuestro proyecto con la segunda opción o manualmente:

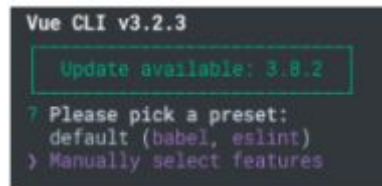


Imagen 17. Vue CLI Manual.

Se puede ir configurando el proyecto según las necesidades de la aplicación, por lo que irá preguntando en cada caso que queremos instalar (con la tecla espacio seleccionamos).



Imagen 18. Vue CLI opciones.

Nos muestra muchas opciones a elegir, veamos las más importantes:

- Babel, el compilador javascript que más se ha utilizado en el último tiempo, es el encargado de transformar nuestro código javascript codificado en el estándar ES6 (por ejemplo) y hacer que este sea compatible con algunos navegadores que no soportan este estándar (internet explorer, entre otros), por lo tanto, siempre es recomendable utilizarlo. A continuación, se podrá observar un pequeño ejemplo al implementar la compilación con Babel.

```
let a = 1;
let b = 2;

(a, b) => { console.log(a+b) }
```

En el código anterior, se puede apreciar una función escrita en ES6 (función flecha) que realiza la suma de las variables "a"+"b" y muestra el resultado por la consola web del navegador. Por lo tanto, la conversión que genera Babel para que se entienda por la mayoría de los browsers es como se muestra a continuación:

```
var a = 1;
var b = 2;
(function(a, b){
  console.log(a+b);
})
```

- **TypeScript:** el llamado super set de JavaScript, nos permite orientar a objetos Javascript, añadir tipos, por ejemplo, definir si una variable es String o no, entre muchas otras opciones, la descripción más regular que se da de TypeScript, es que es JavaScript con "Superpoderes". (no es necesario instalar esta característica para poder generar el proyecto manualmente).
- **PWA:** Progressive Web Application, no son particularmente nuevas, pero han agarrado mucha notoriedad en el último tiempo, dado que microsoft anunció que empezaría a soportarlas, pero ¿que son?, son aplicaciones web que se comportan como si fueran aplicaciones nativas en los diversos sistemas operativos, mediante el uso de services workers y otras tecnologías. Una de las grandes particularidades es que se ejecutan en segundo plano, lo que implica que no están viviendo necesariamente en el navegador web. Es posible instalarlas en nuestros teléfonos además de en nuestros equipos personales. Con estas se rompe un poco la barrera de instalación entre una aplicación nativa al sistema operativo (sea desktop o móvil), y nos brinda una gran alternativa a nosotros como desarrolladores de crear elementos transversales completamente. (no es necesario instalar esta característica para poder generar el proyecto manualmente)
- **Router:** Las aplicaciones web generalmente poseen muchas vistas, las cuales generalmente tienen un path diferente, por ejemplo la página principal de nuestro sistema pudiese ser: midominio.cl/index y una vista secundaria podría ser midominio.cl/abrochatuzapatilla, para esto necesitamos 'Router', el cual nos brinda la opción de generar un "ruteo" de nuestra aplicación web. Ok, puedes pensar, "pero eso lo hago con un tag html", si y no, sí porque sólo si quisieras enviar de una vista a otra, y no, porque con Router, podremos pasar propiedades, definir condiciones de acceso a dichas vistas, entre muchas más opciones. (Es recomendable instalar esta característica cuando ya conozcas un poco más del tema que podrás estudiar a fondo en las unidades posteriores).
- **Vuex,** el store de Vue, es el lugar centralizado donde tendremos acceso a propiedades de nuestra aplicación disponibles para su uso en cuanto sea necesario.

Puedes pensarlo como un “repositorio” de datos, esta librería (oficial), nos permite manejar el estado de nuestra aplicación. (Hay mucho más que decir respecto Vuex, pero se verá en detalle en las próximas unidades, por lo que es recomendable instalar esta característica cuando ya conozcas un poco más del tema).

- Pre procesadores CSS, extienden las funcionalidades comunes de css, nos permiten utilizar funciones, mixins, variables, reutilizar código, entre muchas otras. Los más conocidos son Sass, Less y Stylus. Este código no es CSS propiamente (pero muy parecido, en ocasiones no nos damos cuenta que no es CSS), y como tal debe ser compilado. (No es esencial para desarrollar, pero sí recomendable cuando ya tienes conocimientos sobre los pre procesadores).
- **Linters:** Son herramientas que permiten manejar buenas prácticas de desarrollo (indentación, evitar mensajes ‘console.log’, largo de líneas, buen uso de variables según el contexto, entre muchas otras). Deberías instalarlo siempre.
- **Testing:** muy importante, se tienen dos opciones con Vue CLI. Unit testing y E2E testing.

Unit testing: se refiere a los test unitarios, cada porción de código que escribimos debe ser probada, para ese fin debemos escribir test, estos deben cumplir con probar que dicho funcionamiento sea el adecuado en función de los parámetros que deseamos comprobar, por ejemplo, si tenemos una función que suma, debemos generar un test que haga que dicha función sume.

E2E (end to end) testing, es igual de importante que Unit testing, la diferencia es que en esta se prueba una aplicación o sistema como un todo, por ejemplo podemos probar el flujo de login de una aplicación ingresando datos correctos e incorrectos y comprobando que el comportamiento es el que debe ser.

Si bien no es obligatorio hacer testing, si nos suma bastante y nos ahorra bastantes dolores de cabeza si realizamos estos adecuadamente.

En conclusión, por los momentos solo debes seleccionar Babel y Linter/Formatter. Después de esta elección, se mostrarán más opciones, donde se debe seleccionar "ESLint with error prevention only", para que así nuestro editor de texto nos indique posibles errores en javascript:

```
Vue CLI v4.2.3
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Linter
? Pick a linter / formatter config: (Use arrow keys)
> ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier
```

Imagen 19. Vue CLI más opciones - ESLint with error prevention only.

Luego de esto, aparecerá otra pregunta en la terminal, en la cual, debemos seleccionar "Lint on save", esto permitirá generar advertencias cuando existan errores en el código.

```
Vue CLI v4.2.3
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Linter
? Pick a linter / formatter config: Basic
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)
> ● Lint on save
  ○ Lint and fix on commit
```

Imagen 20. Vue CLI más opciones - Lint on save.

Después de la pregunta anterior, se genera otra en la terminal, donde se solicita donde se prefiere colocar la configuración para Babel, ESLint, entre otros, para ello y mantener el proceso de instalación como el default, se debe seleccionar la segunda opción "In package.json".

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Linter
? Choose a version of Vue.js that you want to start the project with 2.x
? Pick a linter / formatter config: Basic
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.?
  In dedicated config files
> In package.json
```

Imagen 21. Vue CLI más opciones - In package.json.

Ya para finalizar todo el proceso de instalación manual, en la terminal aparecerá la última pregunta, donde se indica si se desea guardar la configuración seleccionada como un preajuste para futuros proyectos. En este caso podemos escribir la letra "N" para indicar que no guarde ninguna configuración.

```
Vue CLI v4.2.3
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection)Babel, Linter
? Pick a linter / formatter config: Basic
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? In package.json
? Save this as a preset for future projects? (y/N) N
```

Imagen 22. Vue CLI más opciones - Finalizando proceso de configuración.

Al presionar la tecla enter en la opción que se muestra en la imagen anterior, Vue CLI iniciará todo el proceso de instalación de dependencias de acuerdo a las opciones seleccionadas y aparecerá el resultado final de la instalación, donde si ejecutamos el comando 'yarn serve' o 'npm run serve' (al igual que cuando se instaló automáticamente) se mostrará el mismo resultado en el navegador web (recuerda copiar la dirección o URL proporcionada en el terminal y pegarla en la barra de direcciones del navegador web de tu preferencia).

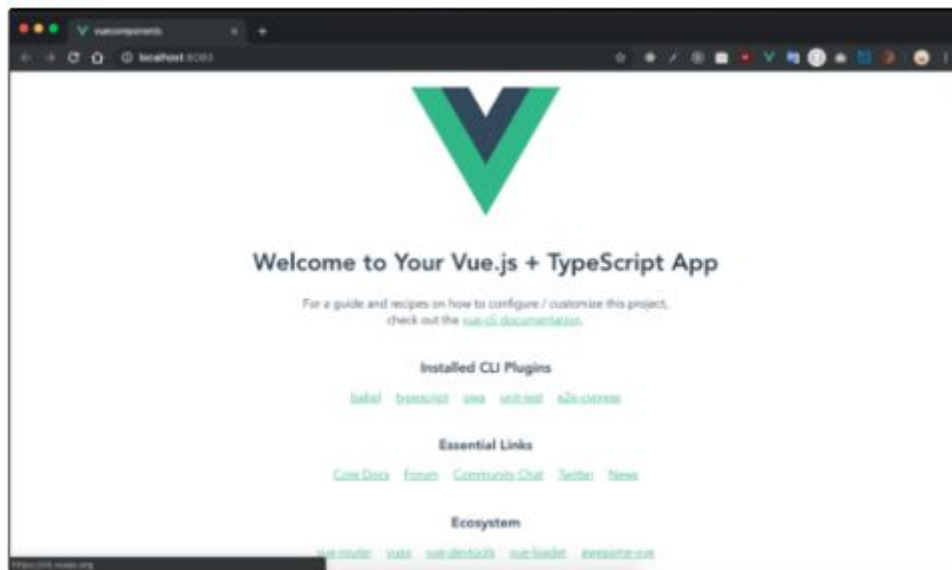


Imagen 23. Instalación full default.

Algo muy relevante: El archivo `package.json`, es muy importante, porque en él se definen las dependencias que serán utilizadas en el proyecto, es decir, si esta va a ser una dependencia de desarrollo o producción, que versión de una librería vamos a usar, entre muchas otras. Ahora, al ver en el contenido del archivo, queda un archivo que contiene varias secciones.

Al principio, el nombre del proyecto, además de la versión, y algunos scripts con los que podemos ejecutar acciones, por ejemplo 'yarn serve' o 'npm run serve', la cual levanta nuestro entorno de desarrollo para poder ser visible desde un navegador web, 'build' la cual nos permite construir (compilar) nuestros archivos para poder pasarlos a producción. 'lint' el cual nos permite verificar si estamos cumpliendo con las reglas del linter. También, pueden existir otras configuraciones en el script si se seleccionaba otras opciones, como: 'test:e2e' que levantará la ejecución del test e2e llamando a cypress, o 'test:unit', lo que correrá la suite de test unitarios.

```
{
  "name": "nuevo_proyecto",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
}
```

Luego, en el archivo veremos la sección de 'dependencies' la cual muestra lo siguiente:

```
"dependencies": {  
  "core-js": "^3.6.4",  
  "vue": "^2.6.11"  
},
```

En primer lugar se tiene el núcleo de vue y posteriormente Vue propiamente. En el caso de instalar otras dependencias como 'vue-router', 'vuex' o 'TypeScript' por ejemplo, éstas también aparecerán en esta área con el número de versión instalado.

En resumen, todo lo necesario para que nuestro proyecto funcione adecuadamente en un ambiente productivo se encontrará en esta sección del package.json.

Luego, tendremos la sección de 'devDependencies', que es algo así:

```
"devDependencies": {  
  "@vue/cli-plugin-babel": "~4.2.0",  
  "@vue/cli-plugin-eslint": "~4.2.0",  
  "@vue/cli-service": "~4.2.0",  
  "babel-eslint": "^10.0.3",  
  "eslint": "^6.7.2",  
  "eslint-plugin-vue": "^6.1.2",  
  "vue-template-compiler": "^2.6.11"  
},
```

En la sección de 'devDependencies' mostrada anteriormente, es donde se instalarán las dependencia y plugins en versión de desarrollo. Aquí se podrán encontrar dependencias como las de Babel, eslint, types (para typescript), diversos plugins, para babel, cypress, pwa, unit testing, entre otros, además de la dependencia para sass (pre procesador css) entre muchas otras. Todo depende de las opciones seleccionadas en el proceso de configuración manual, por lo tanto, varía de un proyecto a otro.

Algo relevante: En las dependencias en la parte derecha, se tiene la versión escrita de la siguiente forma '^número', lo que indica que esa versión es la mínima a utilizar y el carácter '^' indica que puede ser superior la versión realmente instalada.

Para finalizar, el archivo presenta más secciones pero que no vienen mucho al caso analizar ahora, por lo que es suficiente conocer en parte las opciones que brinda Vue CLI.