**Ejercicio Práctico de Introducción a Vue**

Recursos cortesía y propiedad de Escuela Vue (Juan Andrés Núñez), link GitHub <https://github.com/escuelavue/curso-vue-3-desde-cero/tree/L2_start>

1. Abrir la carpeta con los recursos iniciales descargados del link anterior
2. Allí notamos que tenemos la estructura básica de un html, así como un css y un js. Este último (main.js) tiene el acceso a la api de github (no requiere logueo).
3. Primero en el index copiamos el cdn (desde la pag. Oficial) para acceder a Vue:

Pantalla negra con letras blancas

Descripción generada automáticamente con confianza media

1. En main.js copiamos el código de inicio rápido que está en la página oficial de Vue. Ya no se trabaja con una instancia como en Vue 2 sino con el método createApp. En este caso se usará un objeto apodado message:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

1. Sin embargo, es necesario indicarle a Vue de manera explícita en este js, la parte o sección donde se va a montar (mount) o acoplar en el html. Para el caso particular aquí abordado, será dentro del div con el id ‘app’, como se aprecia en esta captura del index.html:

Texto

Descripción generada automáticamente

La forma de indicar esto en el js es:

Imagen que contiene Logotipo

Descripción generada automáticamente como lo muestra la documentación oficial.

1. Para mostrar el mensaje contenido en el objeto message, en nuestra página web principal, solo tendríamos que interpolar dicho objeto por nombre en nuestro index. Lo podríamos hacer en una etiqueta de párrafo por ejemplo:

Texto

Descripción generada automáticamente

1. Ahora haremos una prueba con la reactividad que se puede lograr con Vue:
   1. Cortamos la línea de código del js referente a mount(‘#app’), sin el punto, y la pegamos en el index, dentro de un nuevo script, asignando esa línea a una variable que llamaremos mountedApp:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

\*\*se uso app. para indicar que es heredada de la constante app creada en el js.

* 1. Ahora desde la consola del navegador sí podemos tener acceso a mountedApp con la información traída de Vue.
     1. Damos clic en inspeccionar, en el navegador, y luego en consola:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Vemos que trae el mensaje de ese objeto

* + 1. Incluso desde la misma consola se puede cambiar el valor de message y el navegador lo da por válido:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Usando las Vue DepTools**

**Es un conjunto de herramientas** que permiten depurar y averiguar lo que sucede dentro de un proyecto Vue. Cómo implementarlo:

1. Ir a la página oficial <https://devtools.vuejs.org/> y luego install now y buscar el requerido para el navegador usado.
2. Se da clic en inspeccionar, se busca la opción Vue y se ubica al principio. También es bueno, acoplar el inspeccionar en la parte inferior:

**Captura de pantalla de un celular

Descripción generada automáticamente**

1. Al dar clic en la parte de <Root> fragment, veremos el modelo y el objeto que estamos trabajando:

**Interfaz de usuario gráfica

Descripción generada automáticamente**

1. El mensaje en rojo se puede cambiar y también veremos el cambio en el navegador:

**Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente**

**TRABAJANDO CON LA API DE GITHUB**

1. Es necesario ir a la API oficial: <https://docs.github.com/es/rest/users/users#get-a-user>
2. Vamos a Usuarios->usuarios->get a user

Texto

Descripción generada automáticamente

1. Allí vemos lo siguiente en la sección de get a user:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Vemos que se requiere la ruta /users (que ya la tenemos en una constante en nuestro main), y luego el parámetro {username}

1. En nuestro archivo main.js creamos una función con las siguientes líneas. Es de tener en cuenta que la respuesta recibida desde el servidor es asíncrona (muy diferente el caso si estuviésemos trabajando en local). En este caso usaremos la palabra reservada await, que si recordamos, únicamente puede ser usada dentro de una función de tipo async:

Texto

Descripción generada automáticamente

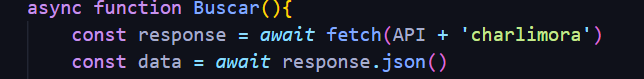
Queremos ver qué se imprime con ese response.

1. Vamos a la consola del navegador e imprimimos el Buscar():

Captura de pantalla de computadora

Descripción generada automáticamente

1. La información arrojada por el response es de tipo json, por lo cual es necesario interpretarla como tal. Para eso usamos json() colocándolo con punto junto a response. Definiremos una nueva constante llamada data para poder almacenar la información obtenida.



1. Por último en vez de mostrar en la consola, la impresión de response, mostraremos la de data. Entonces, borramos response del console.log y ponemos data:

Texto

Descripción generada automáticamente

1. Luego llamamos la función desde la consola:

Texto

Descripción generada automáticamente

Precisamente se obtiene la información propia de GitHub para el usuario charlimora.