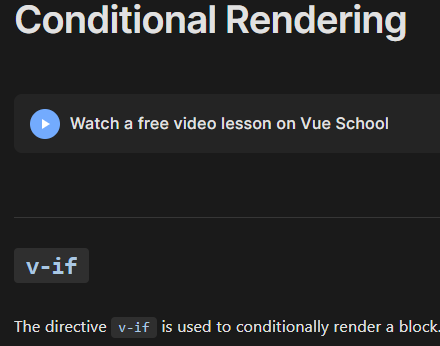
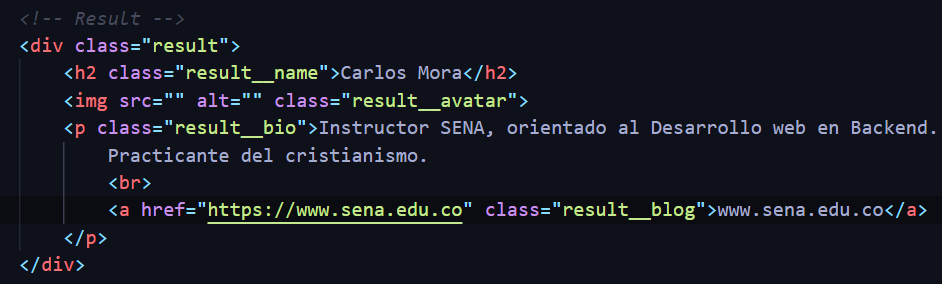
**04 - Ejercicio Práctico de Introducción a Vue: Renderizado Condicional**

Primero podemos consultar la página oficial sobre conditional rendering en <https://vuejs.org/guide/essentials/conditional.html#conditional-rendering>

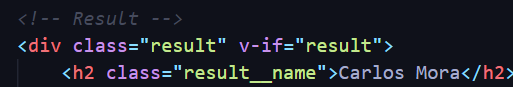


Estas directivas nos permiten decirle a vue que muestre u oculte contenido de acuerdo a una condición. En nuestro caso la condición es que obtengamos algún resultado. Siga estos pasos:

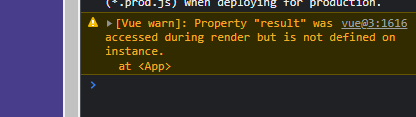
1. Digite el código de widget proporcionado en el material de apoyo.



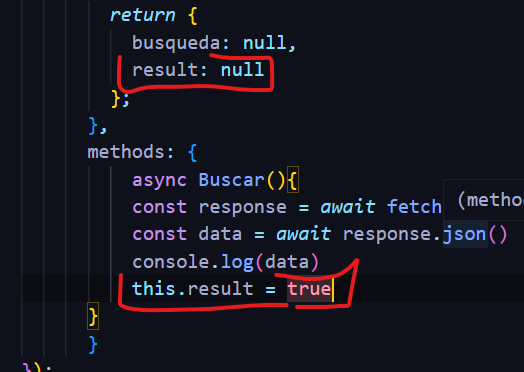
1. Luego en el div colocamos la condición con v-if:



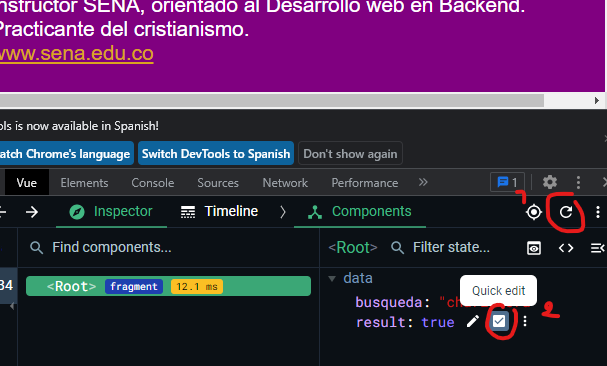
1. Veremos que sale un error en consola:



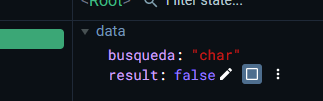
1. Si vamos a la instancia no encontramos nada de result. Por eso el error.
2. Definimos ese result en la instancia para inicie como nulo y que cuando se haga una búsqueda se vuelva verdadero. Esto iría por ahora en los métodos:

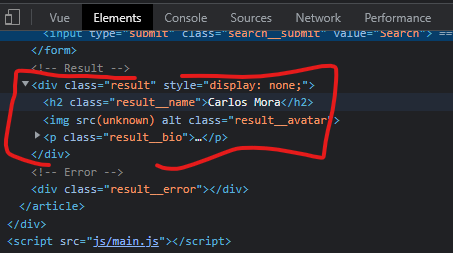


1. Podemos comprobar que funciona en el navegador y verificar la reactividad con el plugin de Vue (es necesario forece refresh). Luego en el mismo plugin podemos intercambiar entre true y falase:

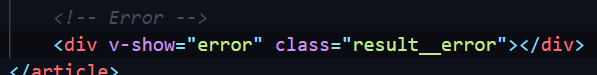


1. Es de tener en cuenta que es más recomendable usar v-show que v-if, ya que v-if consume más recursos, renderiza y luego re renderiza (nuevamente) la información. En v-show, la información que se desea mostrar y ocultar cada vez, sigue dentro del DOM cuando usamos la opción de Inspeccionar en el código, en Elementos, y vamos intercambiando entre true y false en el Vue Devtools

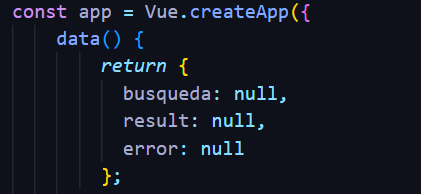




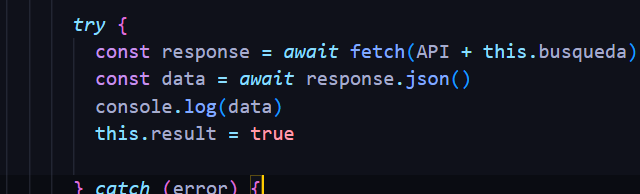
1. Ahora si nos fijamos, en la parte de debajo del widget, aparece una barra naranja para los errores. Solo debería mostrarse si, obviamente ocurren errores al hacer la búsqueda. Po ello implementamos un v-if o un v-show para indicar la condicional para que se muestre u oculte:



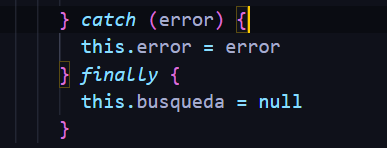
1. Se crea también el objeto error en la instancia:



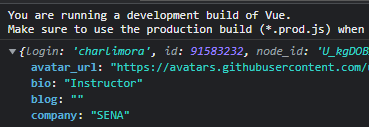
1. En los métodos, creamos un try catch (en VSC se selecciona trycatch de las sugerencias). En el try ponemos el fragmento de código del método trabajado antes:



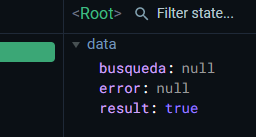
1. Luego en el catch especificamos que el objeto o atributo error de la instancia, almacenará el error que se produzca en caso de que así ocurriese. Además, con finally, indicaremos que en caso de que todo salga bien, se limpie el cuadro de búsqueda, poniendo la búsqueda en null:



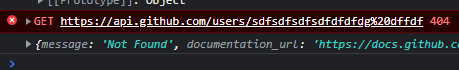
1. Hacemos las respectivas pruebas. Buscamos un usuario válido. Vemos que el resultado en consola es positivo:



1. Y en el plugin de Devtools, al hacer force refresh, el result está true:

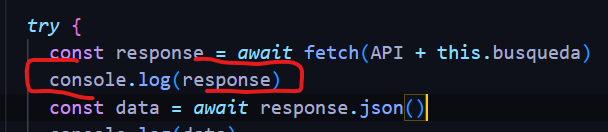


1. Pero si buscamos un usuario inexistente, sale un error 404 en consola, pero el error sigue siendo null.

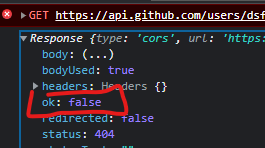


Esto sucede porque de acuerdo al navegador, el request ha funcionado bien. Simplemente, no se encontró el usuario, pero hubo respuesta.

1. Lanzaremos el error manualmente, haciendo lo siguiente:
   1. Primero mostramos en consola la respuesta:

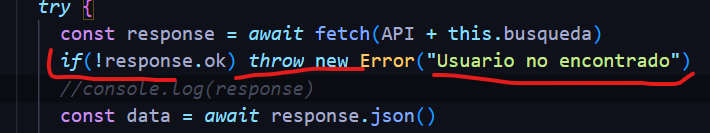


En consola se verá así:



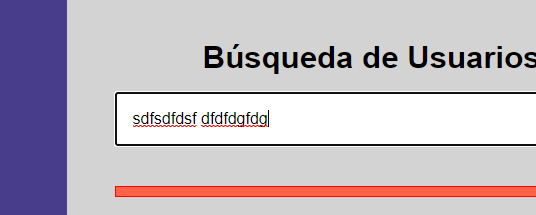
Allí se observa una propiedad llamada ok, que está en falso.

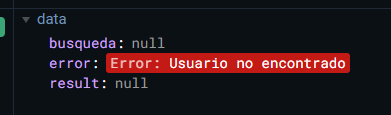
* 1. Así que en la instancia, dentro del try en el método Buscar() evaluamos qué debería pasar si el response no es ok:



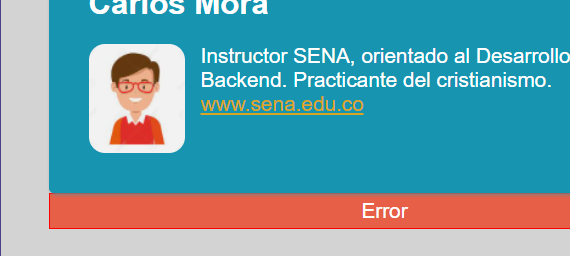
* 1. Además, ya no necesitamos que se muestre el response. Por lo cual o se comenta o se borra.

1. Ahora si hacemos la prueba, con un usuario inexistente, el div se muestra, y el error sale en consola (ver plugin Vue) en el atributo error:

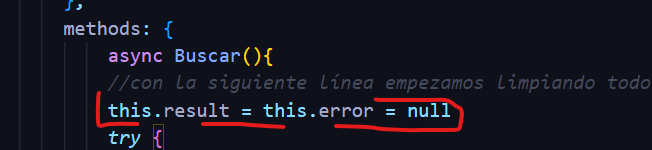




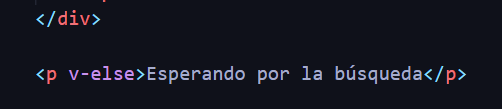
1. Es de notar que si, por ejemplo, si se hace una búsqueda correcta, se muestra el widget (con la información estática). Luego si se hace una búsqueda incorrecta se muestra el div donde debería ir el error, debajo del widget que tiene la información del usuario. Entonces, al hacerse cada operación debería desaparecer una u otra (no las dos), pero no está sucediendo:



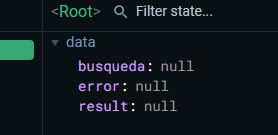
Para corregir esta situación, utilizamos la siguiente línea para iniciar todo en limpio cada vez que se inicia el método Buscar():



1. Luego comprobamos en el navegador el resultado, así como en Inspeccionar y usando Vue Deptools.
2. Ahora podemos ***implementar un mensaje que indique que se está esperando*** por una entrada de búsqueda por parte del usuario. Para eso usamos un v-else (debe ser inmediatamente después del div con el v-if).



Así que mientras result será null (que es lo mismo que falso) se ejecuta este else:

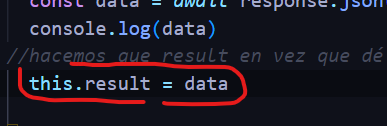


Pero, ¿cómo mostrar en el widget la información del usuario buscado y no una info estática?

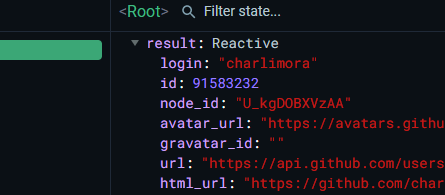
**Sintaxis de Plantilla en Vue**

Doc oficial: <https://vuejs.org/guide/essentials/template-syntax.html#template-syntax>

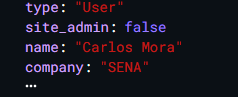
1. Lo primero que haremos es que el result en vez de que retorne un true tras una búsqueda exitosa, arroje la información que se trae de la API. Para eso cambiamos true por data:



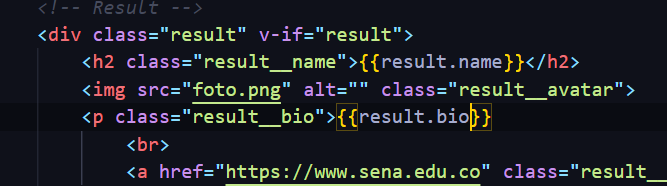
1. Luego verificamos en consola y en el complemento Devtools:



Allí de hecho podemos ver el nombre:



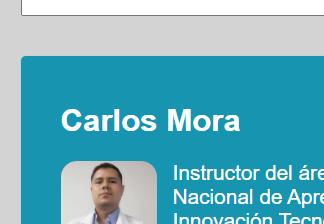
1. Este name lo podríamos traer a través de result.name, y hacemos lo mismo con la bio, en el widget:



1. Sin embargo, no funcionará para la imagen. Si lo que necesitamos es traer un atributo dinámico, como la foto se debe hacer con v-bind:



Verificamos el contenido:

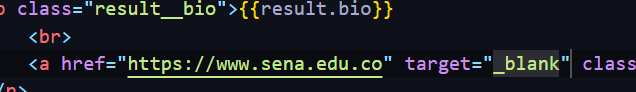


1. Incluso también se puede hacer con el atrbuto alt del img:

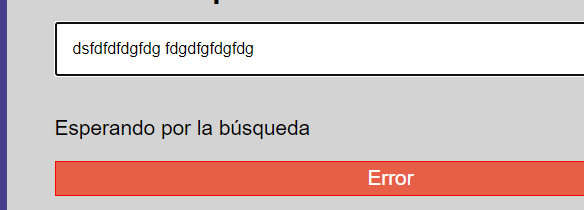


El v-bind también se puede usar para direcciones (links, a).

1. Para que la página se abra en una nueva pestaña usamos target= “\_blank”



**Corrigiendo el mensaje de error:**

Si buscamos un usuario inexistente, no saldrá el error que tenemos definido en la instancia. 

1. Para imprimir el texto almacenado en el atributo error, lo interpolamos como {{error}}, eso sí usando v-if en el div (Nota: también se pude usar v-show):

