

**ACTIVIDAD DE VIDEO – VUE3
PROYECTO DE SOFTWARE III**

PRESENTADO POR:

JOAN SAMUEL ANACONA NARVAEZ

DOCENTE:

ADRIÁN IZQUIERDO

**INSTITUTO TECNOLOGICO DEL PUTUMAYO-
UNIPUTUMAYO**

MOCOA

05 – 09 -2025

CONCEPTOS BÁSICOS

Configuración del entorno y estructura de un componente. En Vue 3, la unidad fundamental de trabajo son los **SFC (Single File Components)**, archivos con extensión .vue que integran en un solo lugar la parte visual, la lógica y los estilos. Cada componente se organiza en tres bloques principales:

1. **<template>**

Aquí se define la vista o estructura visual de la aplicación, escrita en HTML. En esta sección se colocan elementos como botones, formularios, listas, entre otros.

2. **<script>**

Contiene la lógica del componente en JavaScript o TypeScript. Es el espacio donde se declaran los datos reactivos (ref, reactive), los métodos, las propiedades, y también donde se pueden importar y usar *composables* para reutilizar lógica.

3. **<style>**

Permite aplicar los estilos propios del componente, ya sea en CSS o usando preprocesadores como SCSS.

Composition API – Fundamentos

La **Composition API** es la forma moderna de trabajar con Vue 3 y se organiza principalmente en la función `setup()`. Todo lo que se declare dentro de `setup` (variables, funciones, props o estados) estará disponible para el componente.

- **Reactividad**
 - `ref()`: define variables reactivas de tipo simple (números, cadenas, booleanos).
 - `reactive()`: convierte un objeto o un arreglo completo en reactivo.
- **Propiedades derivadas**
 - `computed()`: crea valores calculados automáticamente en función de otros, actualizándose cuando cambian sus dependencias.
- **Observadores**
 - `watch()`: vigila variables específicas y ejecuta acciones cuando estas cambian.
 - `watchEffect()`: se ejecuta automáticamente, detectando las variables reactivas que se utilizan en su interior.
- **Ciclo de vida del componente**

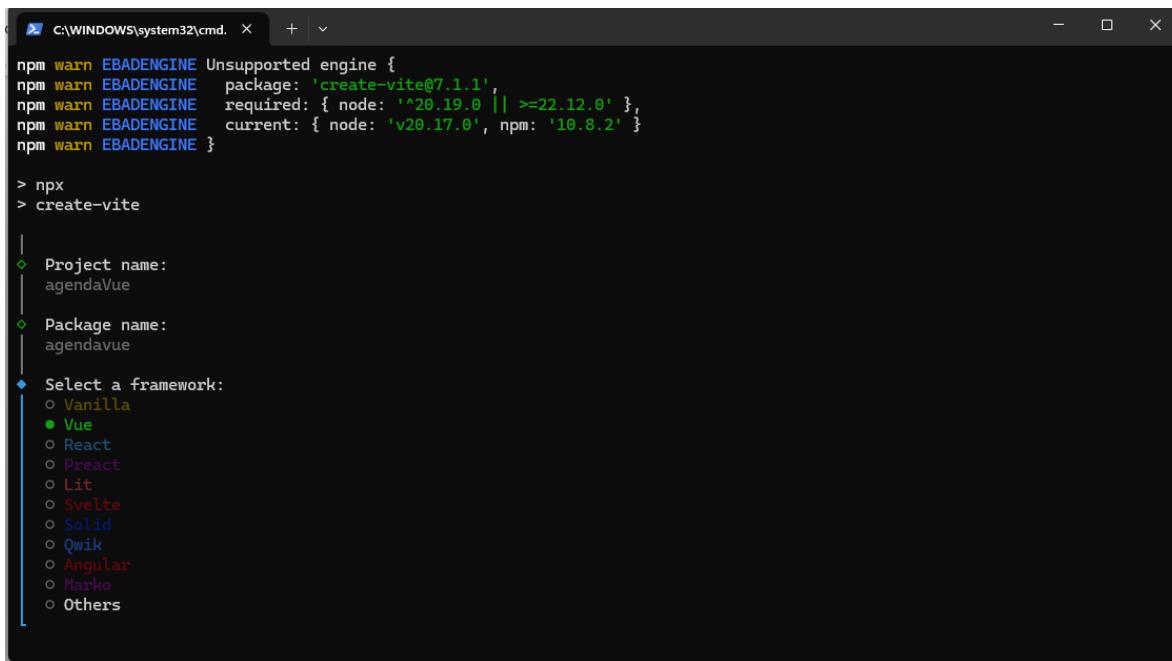
Vue proporciona funciones que se ejecutan en momentos clave del ciclo de vida:

 - `onMounted`: cuando el componente se carga en la vista.
 - `onUpdated`: cuando ocurre una actualización en el estado o en las props.
 - `onUnmounted`: cuando el componente es eliminado de la vista.

Uso de <script setup>

Vue 3 introduce la sintaxis <script setup>, que simplifica la forma de trabajar. Todo lo que se defina dentro de este bloque queda disponible en el <template> sin necesidad de retornar manualmente. Esto hace que el código sea más limpio y fácil de mantener.

- Pasos para crear un Proyecto en Vue

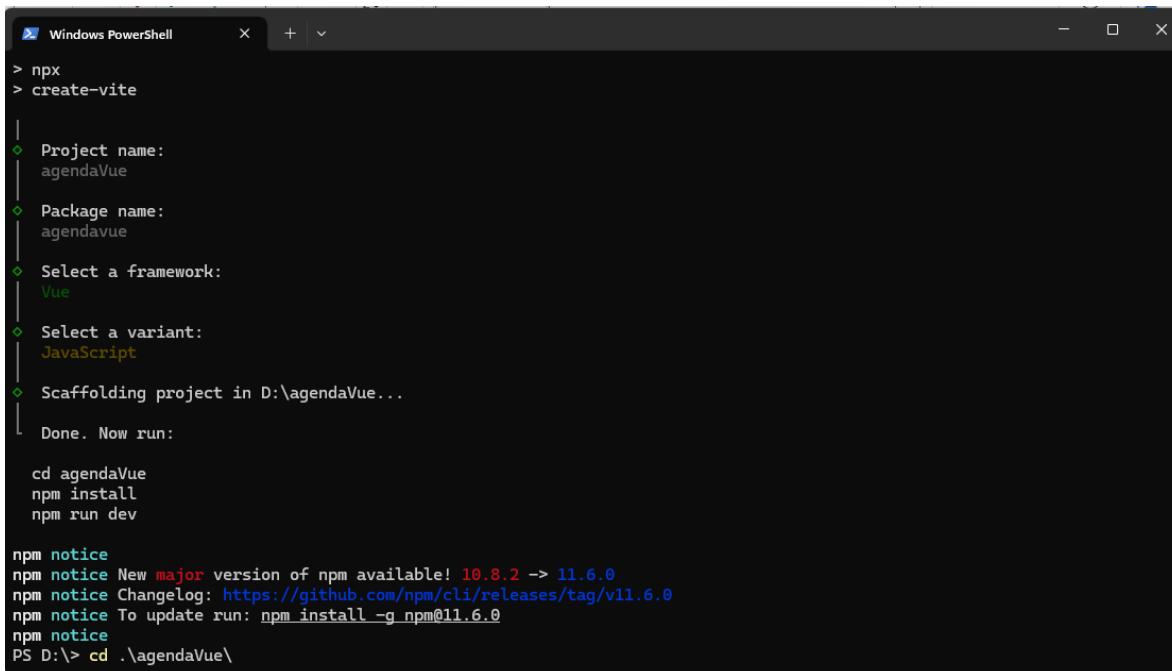


```
C:\WINDOWS\system32\cmd. x + v
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: 'create-vite@7.1.1',
npm warn EBADENGINE   required: { node: '^20.19.0 || >=22.12.0' },
npm warn EBADENGINE   current: { node: 'v20.17.0', npm: '10.8.2' }
npm warn EBADENGINE }

> npx
> create-vite

| 
| Project name: agendaVue
| 
| Package name: agendavue
| 
| Select a framework:
|   o Vanilla
|   ● Vue
|   o React
|   o Preact
|   o Lit
|   o Svelte
|   o Solid
|   o Qwik
|   o Angular
|   o Marko
|   o Others
```

- Ya creado el Proyecto entro a la Carpeta Creada



```

> npx
> create-vite

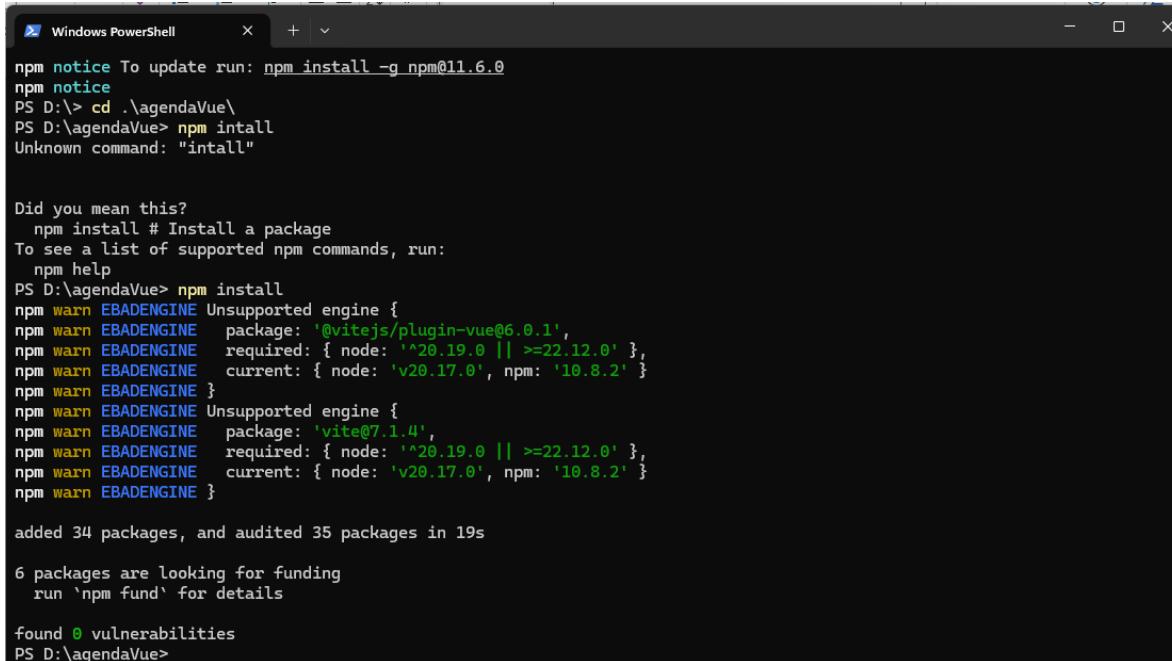
| Project name: agendaVue
| Package name: agendavue
| Select a framework: Vue
| Select a variant: JavaScript
| Scaffolding project in D:\agendaVue...

Done. Now run:

cd agendaVue
npm install
npm run dev

npm notice
npm notice New major version of npm available! 10.8.2 -> 11.6.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.6.0
npm notice To update run: npm install -g npm@11.6.0
npm notice
PS D:\> cd .\agendaVue\
    
```

- Instalamos dependencias



```

npm notice To update run: npm install -g npm@11.6.0
npm notice
PS D:\> cd .\agendaVue\
PS D:\agendaVue> npm intall
Unknown command: "intall"

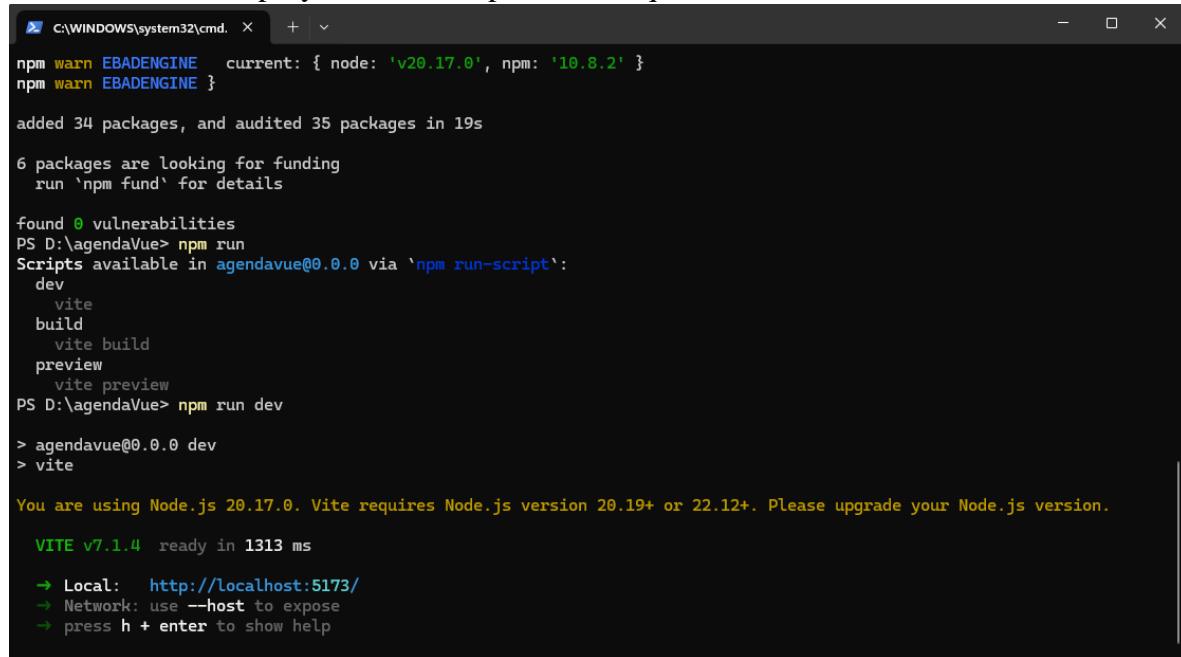
Did you mean this?
  npm install # Install a package
To see a list of supported npm commands, run:
  npm help
PS D:\agendaVue> npm install
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: '@vitejs/plugin-vue@6.0.1',
npm warn EBADENGINE   required: { node: '^20.19.0 || >=22.12.0' },
npm warn EBADENGINE   current: { node: 'v20.17.0', npm: '10.8.2' }
npm warn EBADENGINE }
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: 'vite@7.1.4',
npm warn EBADENGINE   required: { node: '^20.19.0 || >=22.12.0' },
npm warn EBADENGINE   current: { node: 'v20.17.0', npm: '10.8.2' }
npm warn EBADENGINE }

added 34 packages, and audited 35 packages in 19s

6 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
PS D:\agendaVue>
    
```

- Nos da la Url de el proyecto inicial e por defecto que seria esta



```

C:\WINDOWS\system32\cmd. x + v
npm warn EBADENGINE  current: { node: 'v20.17.0', npm: '10.8.2' }
npm warn EBADENGINE }

added 34 packages, and audited 35 packages in 19s

6 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
PS D:\agendaVue> npm run
Scripts available in agendavue@0.0.0 via 'npm run-script':
  dev
    vite
  build
    vite build
  preview
    vite preview
PS D:\agendaVue> npm run dev

> agendavue@0.0.0 dev
> vite

You are using Node.js 20.17.0. Vite requires Node.js version 20.19+ or 22.12+. Please upgrade your Node.js version.

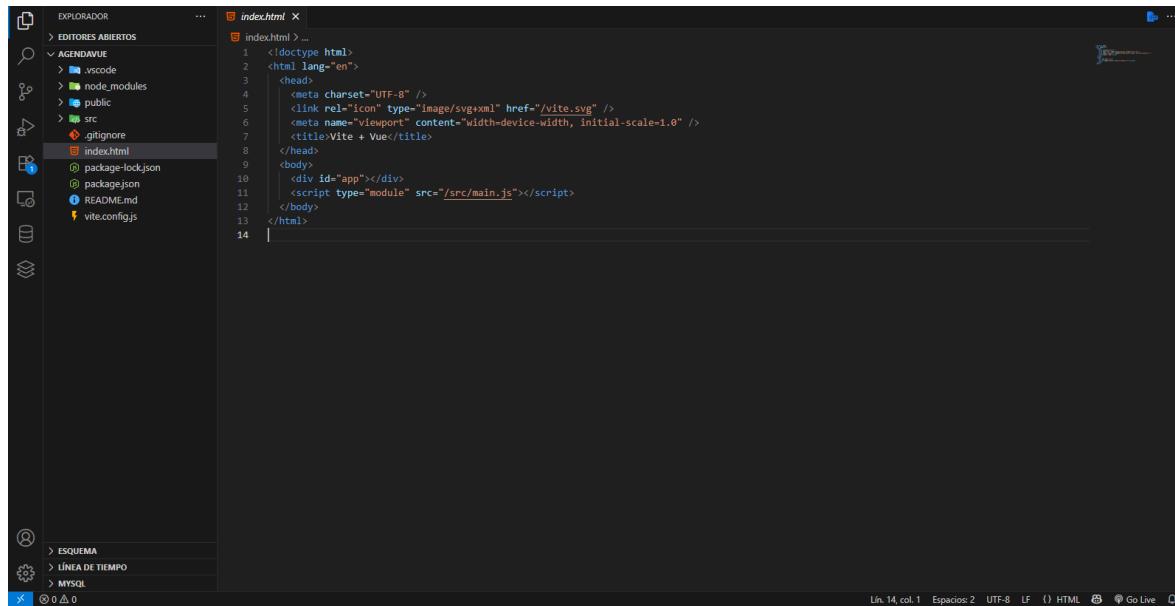
VITE v7.1.4  ready in 1313 ms

→ Local:  http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
    
```

- Ejecutamos la url en el navegador y nos muestra siguiente



- Abrimos nuestro editor fav de código (en este abre el proyecto creado)



```

index.html < ...
<!DOCTYPE html> ...
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + Vue</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>

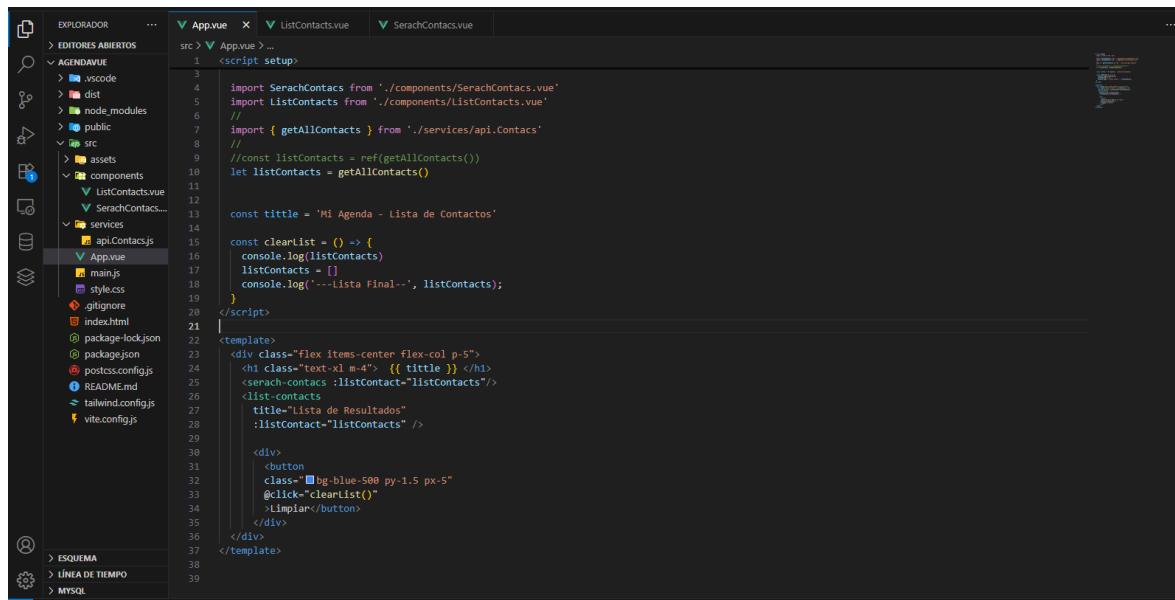
```

Lín. 14, col. 1 Espacios: 2 UTF-8 LF () HTML Go Live

Manejo de estados

Los estados en Vue 3, se implementó una función para limpiar la lista de contactos. Inicialmente, al ejecutar la acción, los datos sí se eliminaban en la consola, pero no se reflejaban en la interfaz del frontend.

Este ejercicio permitió comprender que la reactividad no solo depende de cambiar valores en el código, sino también de asegurarse de que las variables reactivas estén correctamente vinculadas con la vista para que los cambios se visualicen al instante.



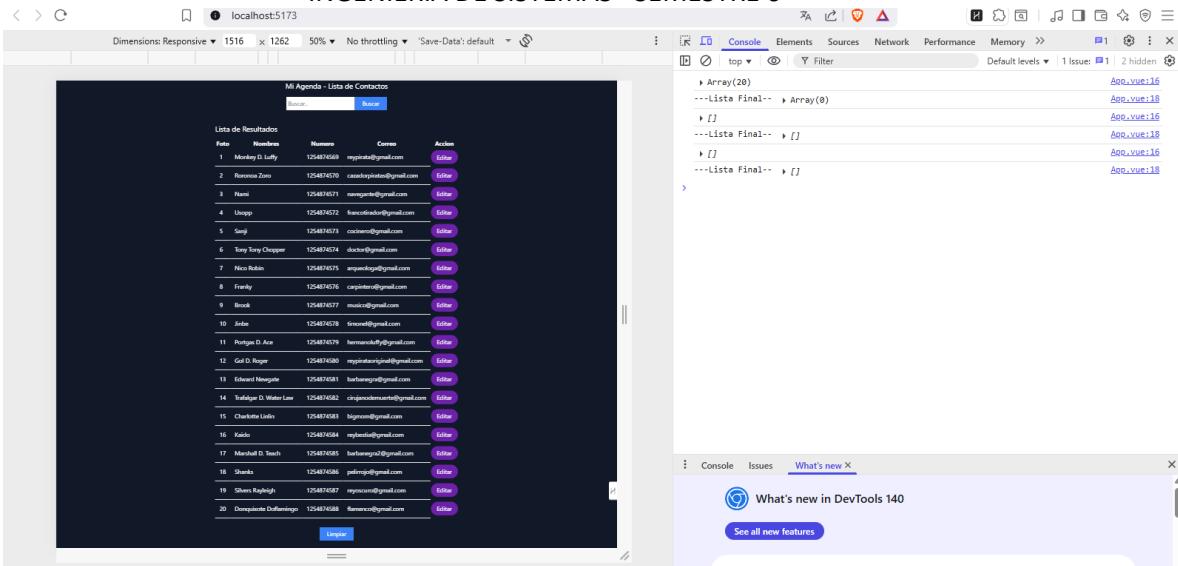
```

App.vue < ...
<script setup>
  import SerachContacs from './components/SerachContacs.vue'
  import ListContacts from './components/listContacts.vue'
  // import { getAllContacts } from './services/api.contacts'
  // const listContacts = ref(getAllContacts())
  let listContacts = getAllContacts()

  const title = 'Mi Agenda - Lista de contactos'

  const clearList = () => {
    console.log(listContacts)
    listContacts = []
    console.log('---Lista Final---', listContacts)
  }
</script>
<template>
  <div class="flex items-center flex-col p-5">
    <h1 class="text-xl m-4"> {{ title }} </h1>
    <serach-contacs :listContact="listContacts" />
    <list-contacts
      title="Lista de Resultados"
      :listContact="listContacts" />
    <div>
      <button
        class="bg-blue-500 py-1.5 px-5"
        @click="clearList()"
        >Limpiar</button>
    </div>
  </div>
</template>

```

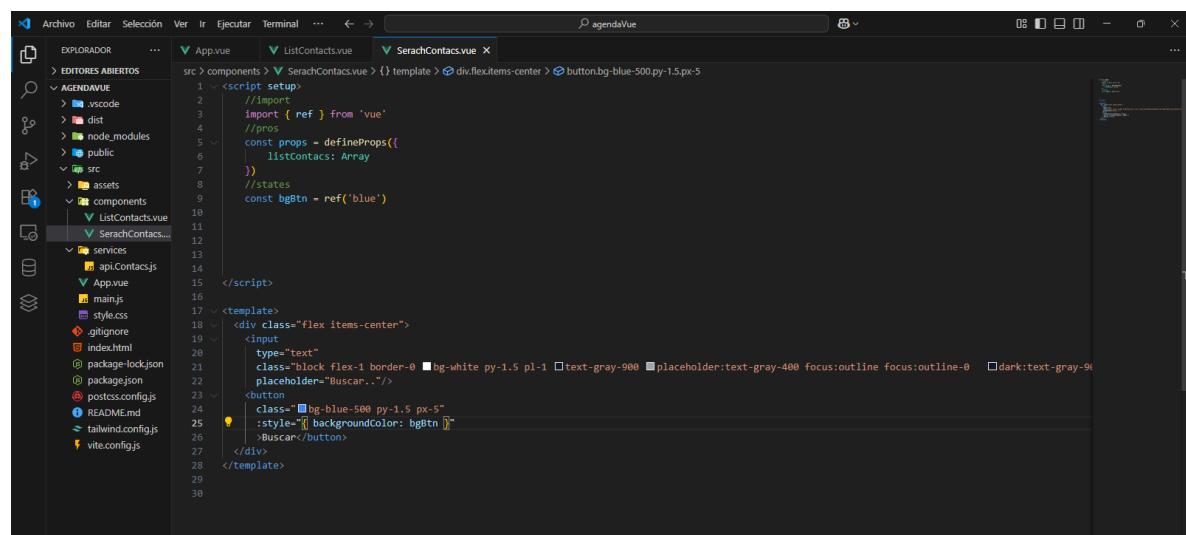


The screenshot shows a Vue.js application running locally at `localhost:5173`. The application displays a table titled "Mi Agenda - Lista de Contactos". The table has columns for Foto, Nombres, Número, Correo, and Acción. There are 20 rows of contact data. The developer tools are open, specifically the Vue DevTools. The "Console" tab shows the state of the "Lista Final" computed property, which is an array of 20 objects. The "Issues" tab shows a single warning about "What's new in DevTools 140".

Enlace de datos (Binding)

El **binding** en Vue 3 se refiere al proceso de **conectar los datos definidos en el bloque <script> con los elementos del <template>**. Gracias a este enlace automático, cualquier cambio que ocurra en las variables de JavaScript se refleja de inmediato en la vista, sin necesidad de escribir código adicional.

De esta manera, los valores que gestionamos en el estado del componente pueden mostrarse directamente en la interfaz HTML, manteniendo sincronizados los datos y la presentación.



```

src > components > SerachContacts.vue > {} template > <div>.flex.items-center <button>.bg-blue-500.py-1.5.px-5
1 <script setup>
2 //import
3 import { ref } from 'vue'
4 //props
5 const props = defineProps({
6   listContacts: Array
7 })
8 //states
9 const bgBtn = ref('blue')
10
11
12
13
14
15 </script>
16
17 <template>
18   <div class="flex items-center">
19     <input
20       type="text"
21       class="block flex-1 border-0 bg-white py-1.5 pl-1 text-gray-900 placeholder:text-gray-400 focus:outline focus:outline-0 dark:text-gray-900"
22       placeholder="Buscar..." />
23     <button
24       class="bg-blue-500 py-1.5 px-5"
25       :style="`backgroundColor: ${bgBtn}`"
26     >Buscar</button>
27   </div>
28 </template>
29
30
31
32
33
34
35
36
37
38
  
```

The screenshot shows a code editor with several files open: `App.vue`, `ListContacts.vue`, and `SerachContacts.vue`. The `SerachContacts.vue` component is the active file, displaying its code. The code uses Vue 3's composition API (`<script setup>`) and template syntax (`<template>`). It defines a prop `listContacts` and a state variable `bgBtn` using `ref`. The template includes an input field for searching and a button with a background color that changes based on the value of `bgBtn`.

localhost:5173

Mi Agenda - Lista de Contactos				
Lista de Resultados				
Foto	Nombres	Numero	Correo	Accion
1	Monkey D. Luffy	1254874569	reypirata@gmail.com	<button>Editar</button>
2	Roronoa Zoro	1254874570	cazadorpiratas@gmail.com	<button>Editar</button>
3	Nami	1254874571	navegante@gmail.com	<button>Editar</button>
4	Usopp	1254874572	francotirador@gmail.com	<button>Editar</button>
5	Sanji	1254874573	cocinero@gmail.com	<button>Editar</button>
6	Tony Tony Chopper	1254874574	doctor@gmail.com	<button>Editar</button>
7	Nico Robin	1254874575	arqueologa@gmail.com	<button>Editar</button>
8	Franky	1254874576	carpintero@gmail.com	<button>Editar</button>
9	Brook	1254874577	musico@gmail.com	<button>Editar</button>
10	Jinbe	1254874578	timonel@gmail.com	<button>Editar</button>
11	Portgas D. Ace	1254874579	hermanoluffy@gmail.com	<button>Editar</button>

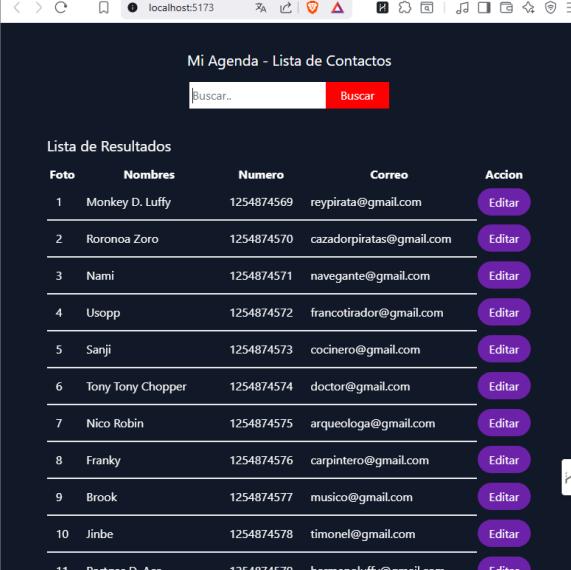
Eventos en Vue – Ejemplo con “focus”

Se probó el manejo de eventos dentro de Vue 3. Para el caso del botón de búsqueda, se configuró que, al escribir en el campo correspondiente y activar el evento focus, el botón cambiara de color automáticamente.

Esta parte del ejercicio permitió comprobar cómo los eventos pueden: modificar dinámicamente la apariencia o el comportamiento de un elemento, reforzando la interacción entre el usuario y la interfaz.

localhost:5173

Mi Agenda - Lista de Contactos				
Lista de Resultados				
Foto	Nombres	Numero	Correo	Accion
1	Monkey D. Luffy	1254874569	reypirata@gmail.com	<button>Editar</button>
2	Roronoa Zoro	1254874570	cazadorpiratas@gmail.com	<button>Editar</button>
3	Nami	1254874571	navegante@gmail.com	<button>Editar</button>
4	Usopp	1254874572	francotirador@gmail.com	<button>Editar</button>
5	Sanji	1254874573	cocinero@gmail.com	<button>Editar</button>
6	Tony Tony Chopper	1254874574	doctor@gmail.com	<button>Editar</button>
7	Nico Robin	1254874575	arqueologa@gmail.com	<button>Editar</button>
8	Franky	1254874576	carpintero@gmail.com	<button>Editar</button>
9	Brook	1254874577	musico@gmail.com	<button>Editar</button>
10	Jinbe	1254874578	timonel@gmail.com	<button>Editar</button>
11	Portgas D. Ace	1254874579	hermanoluffy@gmail.com	<button>Editar</button>

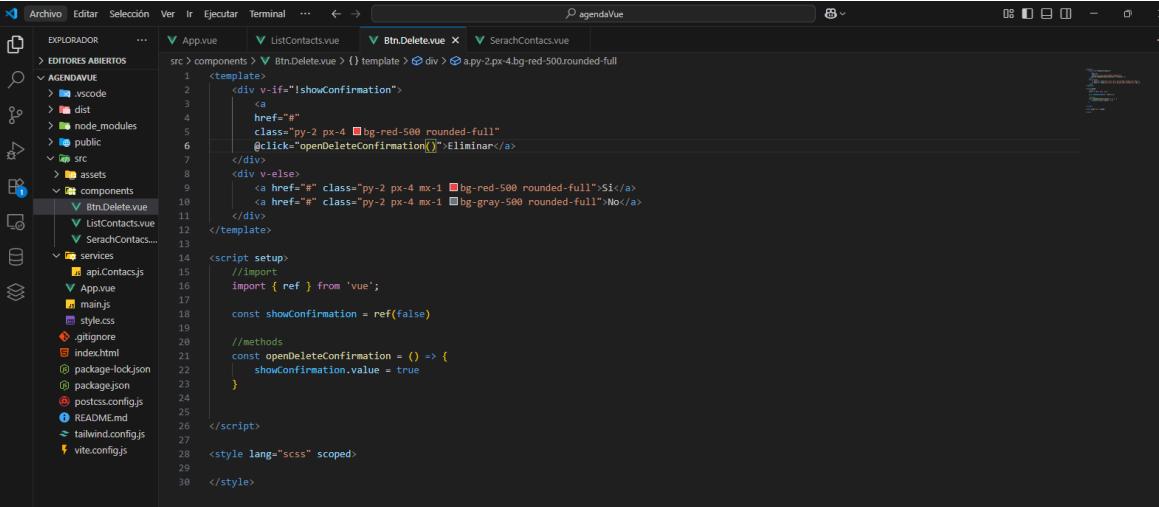


The screenshot shows a Vue.js application running in a browser. The code editor on the left shows components like App.vue, ListContacts.vue, and SerachContacs.vue. The browser window displays a contact list with 11 entries, each with a photo placeholder, name, phone number, email, and an 'Editar' button.

Foto	Nombres	Numero	Correo	Acción
1	Monkey D. Luffy	1254874569	reypirata@gmail.com	<button>Editar</button>
2	Roronoa Zoro	1254874570	cazadorpiratas@gmail.com	<button>Editar</button>
3	Nami	1254874571	navegante@gmail.com	<button>Editar</button>
4	Usopp	1254874572	francotirador@gmail.com	<button>Editar</button>
5	Sanji	1254874573	cocinero@gmail.com	<button>Editar</button>
6	Tony Tony Chopper	1254874574	doctor@gmail.com	<button>Editar</button>
7	Nico Robin	1254874575	arqueologa@gmail.com	<button>Editar</button>
8	Franky	1254874576	carpintero@gmail.com	<button>Editar</button>
9	Brook	1254874577	musico@gmail.com	<button>Editar</button>
10	Jinbe	1254874578	timonel@gmail.com	<button>Editar</button>
11	Portgas D. Ace	1254874579	hermanoluffy@gmail.com	<button>Editar</button>

Implementación del botón de eliminar

Se añadió un botón de eliminación que, al ser presionado por el usuario, muestra automáticamente una confirmación con las opciones “Sí” o “No”. De esta manera, antes de ejecutar la acción definitiva, el sistema da la oportunidad de confirmar o cancelar, evitando eliminaciones accidentales y mejorando la experiencia de uso.



The screenshot shows the code for the Btn.Delete.vue component. It contains a template with two possible states: one where it shows a confirmation dialog (v-if="!showConfirmation") and another where it shows a delete link (v-else). The component also includes a script setup with methods for opening the confirmation dialog and handling its outcome.

```

<template>
  <div v-if="!showConfirmation">
    <a href="#">
      Eliminar
    </a>
  </div>
  <div v-else>
    <a href="#">Si</a>
    <a href="#">No</a>
  </div>
</template>
<script setup>
  import { ref } from 'vue';

  const showConfirmation = ref(false);

  //methods
  const openDeleteConfirmation = () => {
    showConfirmation.value = true
  }
</script>
<style lang="scss" scoped>
</style>

```

localhost:5173/#

Mi Agenda - Lista de Contactos

Buscar.. Buscar

	Foto	Nombres	Numero	Correo	Accion
1	Monkey D. Luffy	1254874569	reypirata@gmail.com	<button>Editar</button> <button>Sí</button> <button>No</button>	
2	Roronoa Zoro	1254874570	cazadorpiratas@gmail.com	<button>Editar</button> <button>Sí</button> <button>No</button>	
3	Nami	1254874571	navegante@gmail.com	<button>Editar</button> <button>Sí</button> <button>No</button>	
4	Usopp	1254874572	francotirador@gmail.com	<button>Editar</button> <button>Sí</button> <button>No</button>	
5	Sanji	1254874573	cocinero@gmail.com	<button>Editar</button> <button>Eliminar</button>	
6	Tony Tony Chopper	1254874574	doctor@gmail.com	<button>Editar</button> <button>Eliminar</button>	
7	Nico Robin	1254874575	arqueologa@gmail.com	<button>Editar</button> <button>Eliminar</button>	
8	Franky	1254874576	carpintero@gmail.com	<button>Editar</button> <button>Eliminar</button>	
9	Brook	1254874577	musico@gmail.com	<button>Editar</button> <button>Eliminar</button>	
10	Jinbe	1254874578	timonel@gmail.com	<button>Editar</button> <button>Eliminar</button>	
11	Portgas D. Ace	1254874579	hermanoluffy@gmail.com	<button>Editar</button> <button>Eliminar</button>	

Búsqueda dinámica en la lista

Se implementó una funcionalidad que permite filtrar la lista en tiempo real. A medida que el usuario escribe en el campo de búsqueda, la lista se va actualizando automáticamente y muestra únicamente los elementos que coinciden con el texto ingresado.

Esto facilita encontrar rápidamente la información deseada y demuestra cómo Vue gestiona la reactividad para actualizar la interfaz de manera inmediata según los cambios en los datos.

localhost:5173/#

App.vue ListContacts.vue BtmDelete.vue SearchContacts.vue

```

<script setup>
  import SerachContacts from './components/SerachContacts'
  import ListContacts from './components/ListContacts.vue'
  import { getAllContacts } from './services/api.Contacts'

  const listContacts = ref(getAllContacts())
  const clearList = ( newList ) => {
    listContacts.value = newList
  }
</script>

<template>
  <div class="flex items-center flex-col p-5">
    <h1 class="text-xl m-4"> {{ title }} </h1>
    <serach-contacts
      :listContact="listContacts"
      :setFunction="clearList"
    >
    <list-contacts
      title="Lista de Resultados"
      :listContact="listContacts"
    >
    <div>
      <button
        class="bg-blue-500 py-1.5 px-5"
        @click="clearList()"
      >Limpiar</button>
    </div>
  </div>
</template>

```

EDIFICIOS ABIERTOS

- AGENDA VUE
 - .vscode
 - dist
 - node_modules
 - public
 - src
 - assets
 - components
 - BtmDelete.vue
 - ListContacts.vue
 - SerachContacts.vue
 - services
 - api.Contacts.js

ESQUEMA

LÍNEA DE TIEMPO

MySQL

Console Elements Sources Network Performance Memory

Default levels 1 issue 1 2 hidden

Mi Agenda - Lista de Contactos

Buscar..

Lista de Resultados

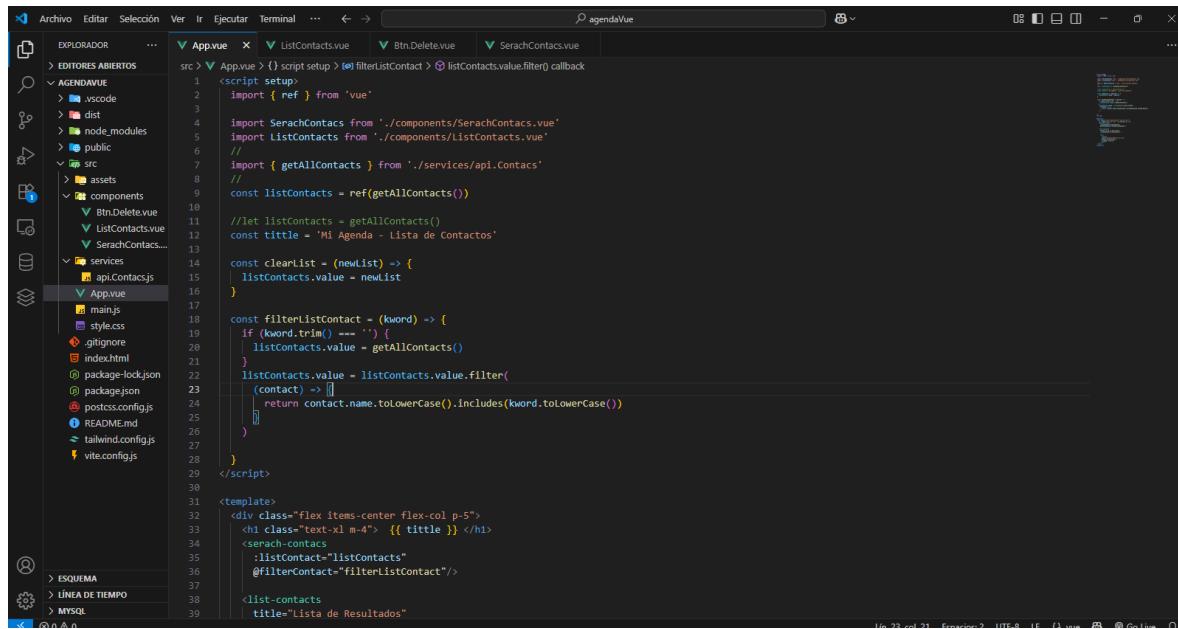
Nombres Numero Correo

Limpiar

What's new in DevTools 140

See all new features

See past highlights from Chrome 139



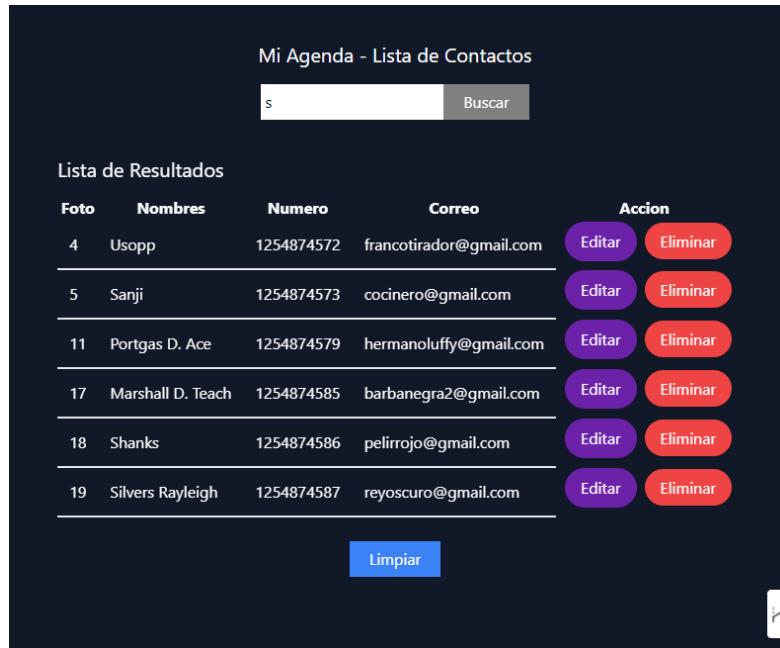
```

<script setup>
import { ref } from 'vue'
import SerachContacs from './components/SerachContacs.vue'
import ListContacts from './components/listContacts.vue'
// import { getAllContacts } from './services/api.contacts'
const listContacts = ref(getAllContacts())
// let listContacts = getAllContacts()
const title = 'Mi Agenda - Lista de Contactos'
const newList = (newList) => {
  listContacts.value = newList
}
const filterListContact = (kword) => {
  if (kword.trim() === '') {
    listContacts.value = getAllContacts()
  }
  listContacts.value = listContacts.value.filter(
    (contact) => [
      return contact.name.toLowerCase().includes(kword.toLowerCase())
    ]
  )
}
</script>
<template>
<div class="flex items-center flex-col p-5">
<h1 class="text-xl m-4"> {{ title }} </h1>
<serach-contacs
:listContact="listContacts"
@filterContact="filterListContact">
</serach-contacs>
<list-contacts
:title="Lista de Resultados"
>
</list-contacts>
</div>

```

Resultado de la búsqueda y eliminación automática

Se comprobó que al escribir un término la lista se filtra correctamente. Posteriormente, al borrar el texto del buscador, los resultados también se eliminan de manera automática, restaurando la lista completa.



Mi Agenda - Lista de Contactos

Foto	Nombres	Numero	Correo	Accion
4	Usopp	1254874572	francotirador@gmail.com	<button>Editar</button> <button>Eliminar</button>
5	Sanji	1254874573	cocinero@gmail.com	<button>Editar</button> <button>Eliminar</button>
11	Portgas D. Ace	1254874579	hermanoluffy@gmail.com	<button>Editar</button> <button>Eliminar</button>
17	Marshall D. Teach	1254874585	barbanegra2@gmail.com	<button>Editar</button> <button>Eliminar</button>
18	Shanks	1254874586	pelirrojo@gmail.com	<button>Editar</button> <button>Eliminar</button>
19	Silvers Rayleigh	1254874587	reyoscuro@gmail.com	<button>Editar</button> <button>Eliminar</button>

Limpiar

Mi Agenda - Lista de Contactos

Buscar

Lista de Resultados

Foto	Nombres	Número	Correo	Acción
1	Monkey D. Luffy	1254874569	reypirata@gmail.com	Editar Eliminar
2	Roronoa Zoro	1254874570	cazadorpiratas@gmail.com	Editar Eliminar
3	Nami	1254874571	navegante@gmail.com	Editar Eliminar
4	Usopp	1254874572	francotirador@gmail.com	Editar Eliminar
5	Sanji	1254874573	cocinero@gmail.com	Editar Eliminar
6	Tony Tony Chopper	1254874574	doctor@gmail.com	Editar Eliminar
7	Nico Robin	1254874575	arqueologa@gmail.com	Editar Eliminar
8	Franky	1254874576	carpintero@gmail.com	Editar Eliminar
9	Brook	1254874577	musico@gmail.com	Editar Eliminar
10	Jinbe	1254874578	timonel@gmail.com	Editar Eliminar

Incorporación de la propiedad “recompensa”

En la etapa final del proyecto se agregó una nueva propiedad llamada “**recompensa**” dentro del archivo apiContact.js. Esta propiedad fue calculada a partir de ciertos parámetros definidos en el código, lo que permitió asignar dinámicamente un valor específico a cada contacto.

La implementación de este método demuestra cómo se pueden extender los datos provenientes de una API, adaptándolos a las necesidades de la aplicación y enriqueciendo la información mostrada en la interfaz.

```

L3     }
L4
L5     //methods
L6     const totalRecompesas = () => {
L7         return props.listContact.reduce(
L8             (accumulator, contact) => accumulator + contact.recompensa, 0
L9         )
L10    }
L11 </script>
L12
L13 <template>
L14     <div class="m-6">
L15         <h3 class="text-xl my-3"> {{ title }}- {{ totalRecompesas() }}</h3>
L16         <table class="table-fixed">
L17
L18
L19
L20
L21
L22
L23
L24
L25
L26

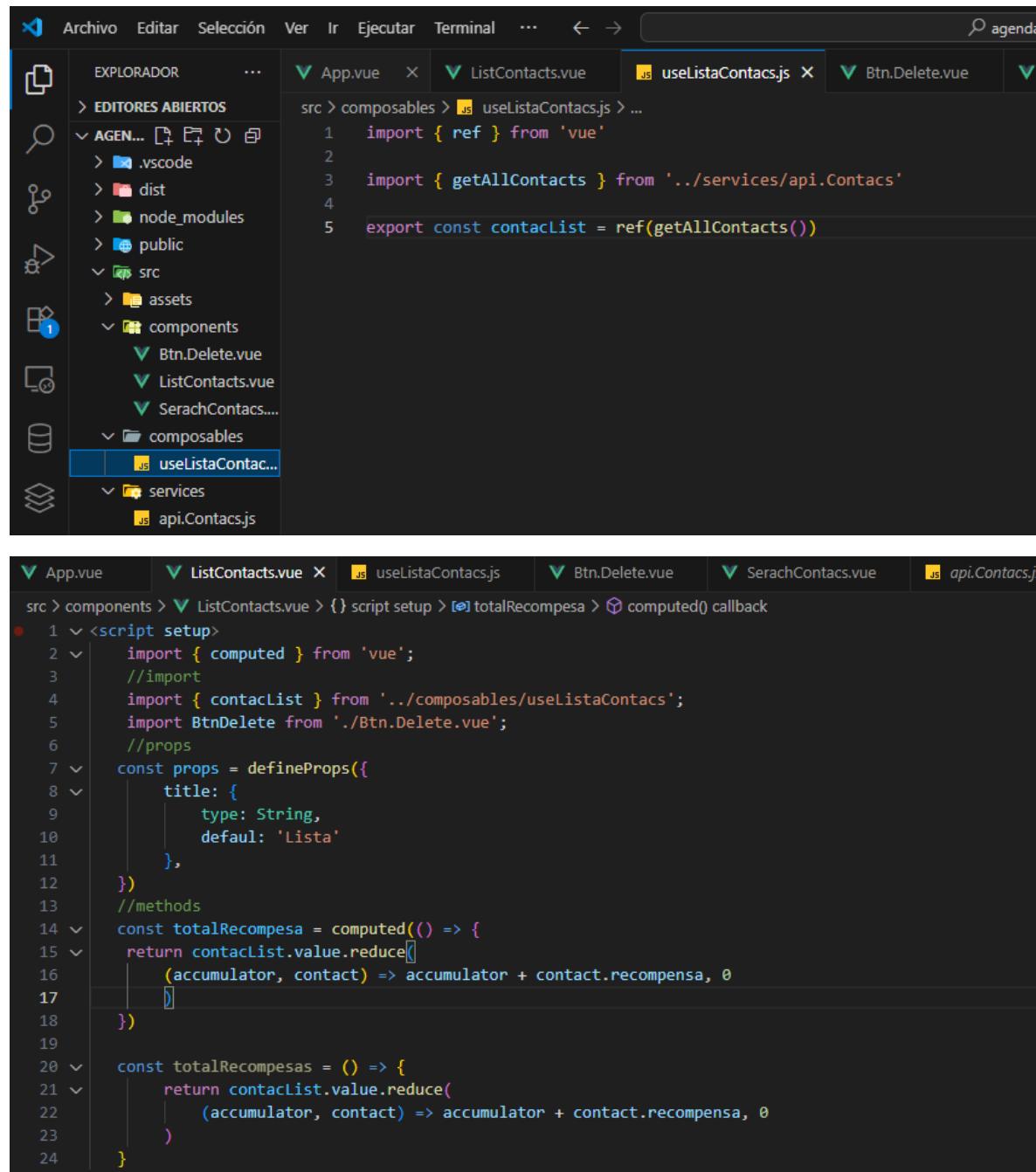
```

Mi Agenda - Lista de Contactos				
Lista de Resultados- 315.401				
Foto	Nombres	Numero	Correo	Accion
1	Monkey D. Luffy	1254874569	reypirata@gmail.com	<button>Editar</button> <button>Eliminar</button>
2	Roronoa Zoro	1254874570	cazadorpiratas@gmail.com	<button>Editar</button> <button>Eliminar</button>
3	Nami	1254874571	navegante@gmail.com	<button>Editar</button> <button>Eliminar</button>
4	Usopp	1254874572	francotirador@gmail.com	<button>Editar</button> <button>Eliminar</button>
5	Sanji	1254874573	cocinero@gmail.com	<button>Editar</button> <button>Eliminar</button>
6	Tony Tony Chopper	1254874574	doctor@gmail.com	<button>Editar</button> <button>Eliminar</button>
7	Nico Robin	1254874575	arqueologa@gmail.com	<button>Editar</button> <button>Eliminar</button>
8	Franky	1254874576	carpintero@gmail.com	<button>Editar</button> <button>Eliminar</button>
9	Brook	1254874577	musico@gmail.com	<button>Editar</button> <button>Eliminar</button>
10	Jinbe	1254874578	timonel@gmail.com	<button>Editar</button> <button>Eliminar</button>

Archivos globales en el proyecto

Los archivos globales cumplen la función de centralizar configuraciones o funcionalidades que se necesitan en diferentes partes de la aplicación. Su uso permite reducir la cantidad de **props** que se deben pasar entre componentes y evita escribir código repetitivo o demasiado extenso.

De esta manera, se logra un proyecto más organizado, fácil de mantener y con una estructura más clara.



The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure. The `src` folder contains `assets`, `components`, `composables`, and `services`. Inside `components` are `Btn.Delete.vue`, `ListContacts.vue`, and `SerachContacs....`. Inside `composables` is `useListaContac...`. Inside `services` is `api.Contacs.js`.
- Code Editor:** The `useListaContacs.js` file is open. It imports `ref` from `'vue'` and `getAllContacts` from `../services/api.Contacs'`. It exports a constant `contactList` which is a ref to `getAllContacts()`.
- Terminal:** The terminal tab is visible at the bottom.

```

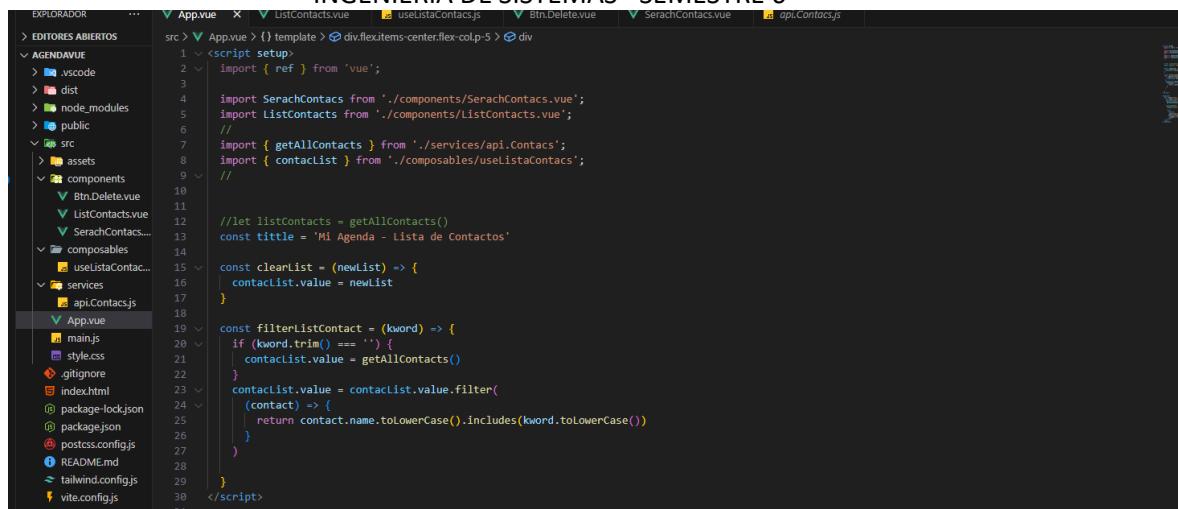
src > composables > useListaContacs.js > ...
1 import { ref } from 'vue'
2
3 import { getAllContacts } from '../services/api.Contacs'
4
5 export const contactList = ref(getAllContacts())

```

```

src > components > ListContacts.vue > {} script setup > totalRecompesa > computed() callback
● 1 <script setup>
  2 import { computed } from 'vue';
  3 //import
  4 import { contacList } from '../composables/useListaContacs';
  5 import BtnDelete from './Btn.Delete.vue';
  6 //props
  7 const props = defineProps({
  8   title: {
  9     type: String,
 10    default: 'Lista'
 11   },
 12 })
 13 //methods
 14 const totalRecompesa = computed(() => {
 15   return contacList.value.reduce(
 16     (accumulator, contact) => accumulator + contact.recompensa, 0
 17   )
 18 }
 19
 20 const totalRecompesas = () => {
 21   return contacList.value.reduce(
 22     (accumulator, contact) => accumulator + contact.recompensa, 0
 23   )
 24 }
 25

```



```

EXPLORADOR ... ▾ App.vue ▾ ListContacts.vue ▾ useListaContacs.js ▾ Btn.Delete.vue ▾ SerachContacs.vue ▾ api.Contacs.js
  ▾ EDITORES ABIERTOS
  ▾ AGENDAVUE
  > .vscode
  > dist
  > node_modules
  > public
  > src
    > assets
    > components
      ▾ Btn.Delete.vue
      ▾ ListContacts.vue
      ▾ SerachContacs...
    > composables
      ▾ useListaContac...
    > services
      ▾ api.Contacs.js
        ▾ App.vue
        main.js
        style.css
      .gitignore
      index.html
      package-lock.json
      package.json
      postcss.config.js
      README.md
      tailwind.config.js
      vite.config.js
  > vite.config.js

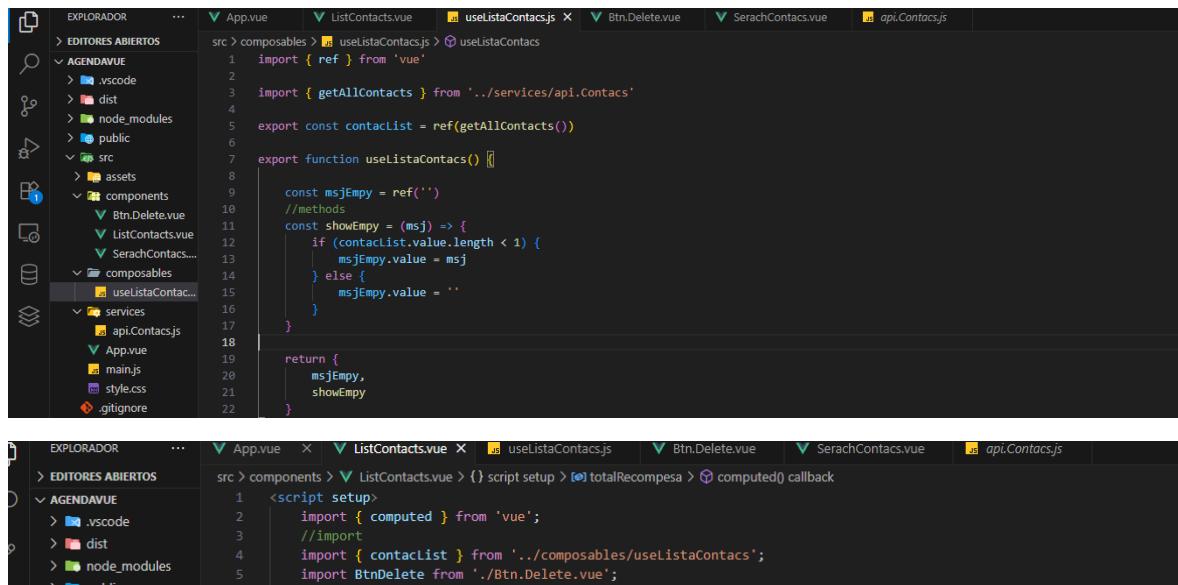
src > App.vue > {} template > ⚡ div.flex.items-center.flex-col.p-5 > ⚡ div
  1 <script setup>
  2   import { ref } from 'vue';
  3
  4   import SerachContacs from './components/SerachContacs.vue';
  5   import ListContacts from './components/ListContacts.vue';
  6   //
  7   import { getAllContacts } from './services/api.Contacs';
  8   import { contactList } from './composables/useListaContacs';
  9   //
 10
 11   //let listContacts = getAllContacts()
 12   const title = 'Mi Agenda - Lista de Contactos'
 13
 14   const clearList = (newList) => {
 15     contactList.value = newList
 16   }
 17
 18   const filterListContact = (kword) => {
 19     if (kword.trim() === '') {
 20       contactList.value = getAllContacts()
 21     }
 22
 23     contactList.value = contactList.value.filter(
 24       (contact) => {
 25         return contact.name.toLowerCase().includes(kword.toLowerCase())
 26       }
 27     )
 28   }
 29
 30   </script>

```

Creación de un composable para búsquedas

Se desarrolló un **composable** que permite gestionar el caso en el que una búsqueda no arroje resultados. Su función es detectar cuando la lista está vacía y mostrar automáticamente un mensaje al usuario, informando que no se encontraron coincidencias.

Este enfoque demuestra cómo los composable ayudan a reutilizar lógica y mantener el código más ordenado, ya que la misma funcionalidad puede aplicarse en distintos componentes sin necesidad de repetirla.



```

EXPLORADOR ... ▾ App.vue ▾ ListContacts.vue ▾ useListaContacs.js ▾ Btn.Delete.vue ▾ SerachContacs.vue ▾ api.Contacs.js
  ▾ EDITORES ABIERTOS
  ▾ AGENDAVUE
  > .vscode
  > dist
  > node_modules
  > public
  > src
    > assets
    > components
      ▾ Btn.Delete.vue
      ▾ ListContacts.vue
      ▾ SerachContacs...
    > composables
      ▾ useListaContac...
    > services
      ▾ api.Contacs.js
        ▾ App.vue
        main.js
        style.css
      .gitignore

src > composables > ▾ useListaContacs.js > ⚡ useListaContacs
  1 import { ref } from 'vue'
  2
  3 import { getAllContacts } from '../services/api.Contacs'
  4
  5 export const contactList = ref(getAllContacts())
  6
  7 export function useListaContacs() {
  8
    const msjEmpty = ref('')
    //methods
    const showEmpty = (msj) => {
      if (contactList.value.length < 1) {
        msjEmpty.value = msj
      } else {
        msjEmpty.value = ''
      }
    }
  10
  11   return {
  12     msjEmpty,
  13     showEmpty
  14   }
  15
  16 }

EXPLORADOR ... ▾ App.vue ▾ ListContacts.vue ▾ useListaContacs.js ▾ Btn.Delete.vue ▾ SerachContacs.vue ▾ api.Contacs.js
  ▾ EDITORES ABIERTOS
  ▾ AGENDAVUE
  > .vscode
  > dist
  > node_modules
  > public

src > components > ▾ ListContacts.vue > {} script setup > ⚡ totalRecompensa > ⚡ computed() callback
  1 <script setup>
  2   import { computed } from 'vue';
  3   //import
  4   import { contactList } from '../composables/useListaContacs';
  5   import BtnDelete from './Btn.Delete.vue';

```

EXPLORADOR

- > EDITORES ABIERTOS
- > AGENDAVUE
- > .vscode
- > dist
- > node_modules
- > public
- > src
 - > assets
 - > components
 - Btn.Delete.vue
 - ListContacts.vue

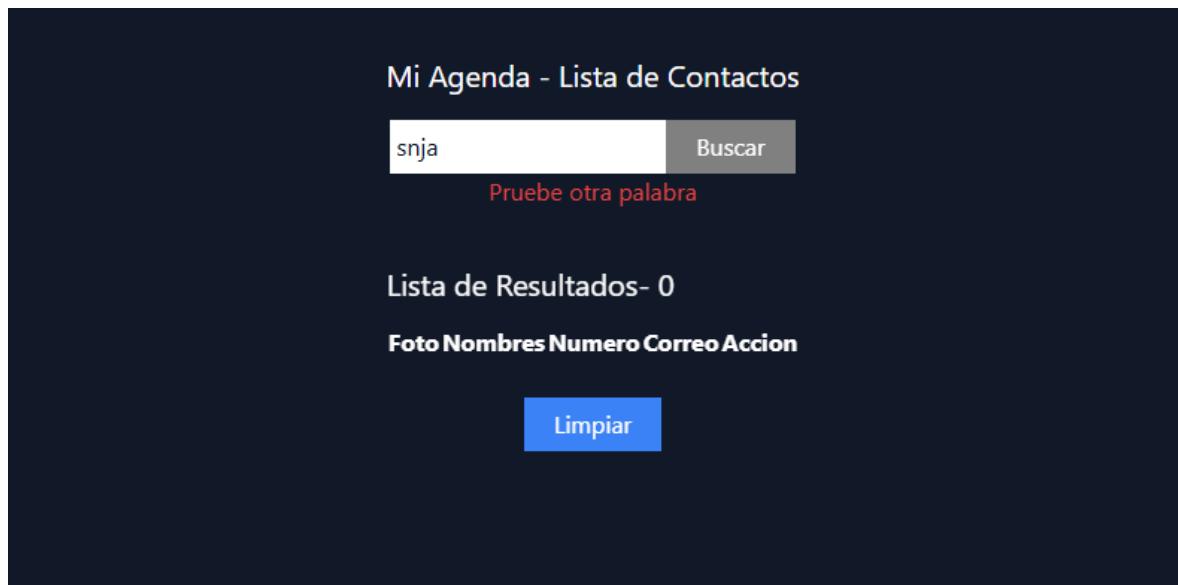
App.vue

```

src > App.vue > () script setup
  1  <script setup>
  2    import { ref } from 'vue';
  3
  4    import SerachContacs from './components/SerachContacs.vue';
  5    import ListContacts from './components/ListContacts.vue';
  6    //
  7    import { getAllContacts } from './services/api.Contacs';
  8    import { contactList } from './composables/useListaContacs';
  9    //
 10
 11  const filtrarDatos = (value) => {
 12    showEmpty('Pruebe otra palabra')
 13    emit('filterContact', keyword.value)
 14  }
 15
 16
```

```

27  const filtrarDatos = (value) => {
28    showEmpty('Pruebe otra palabra')
29    emit('filterContact', keyword.value)
30
31  }
  
```



- Se hace lo mismo para listaContact

EXPLORADOR

- > EDITORES ABIERTOS
- > AGENDAVUE
- > .vscode
- > dist
- > node_modules
- > public
- > src
 - > assets
 - > components
 - Btn.Delete.vue
 - ListContacts.vue
 - SerachContacs...
 - > composables
 - useListaContac...
 - > services
 - api.Contacs.js
 - App.vue
 - main.js
 - style.css
 - > .gitignore
 - > index.html
 - > package-lock.json
 - > package.json
 - > postcss.config.js

ListContacts.vue

```

src > components > ListContacts.vue > () script setup > watchEffect() callback
  1  <script setup>
  2    import { computed, watch, watchEffect } from 'vue';
  3    //import
  4    import { contactList } from '../composables/useListaContacs';
  5    import { useListaContacs } from '../composables/useListaContacs';
  6    import BtnDelete from './Btn.Delete.vue';
  7    //props
  8    const props = defineProps({
  9      title: {
 10        type: String,
 11        default: 'Lista'
 12      },
 13    })
 14    //states composables
 15    const { msjEmpty, showEmpty } = useListaContacs()
 16
 17    //watchers
 18    //watch(contactList, (newValue, oldValue) => {
 19    //  console.log(newValue, oldValue)
 20    //  showEmpty('No se encontró resultados')
 21    //})
 22
 23    watchEffect(() => [
 24      console.log(contactList.value),
 25      showEmpty('No se encontró resultados')
 26    ])
 27
 28
```

Mi Agenda - Lista de Contactos

SAM

Buscar

Lista de Resultados- 0

Foto Nombres Numero Correo Accion

No se encontro resultados

Limpiar