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

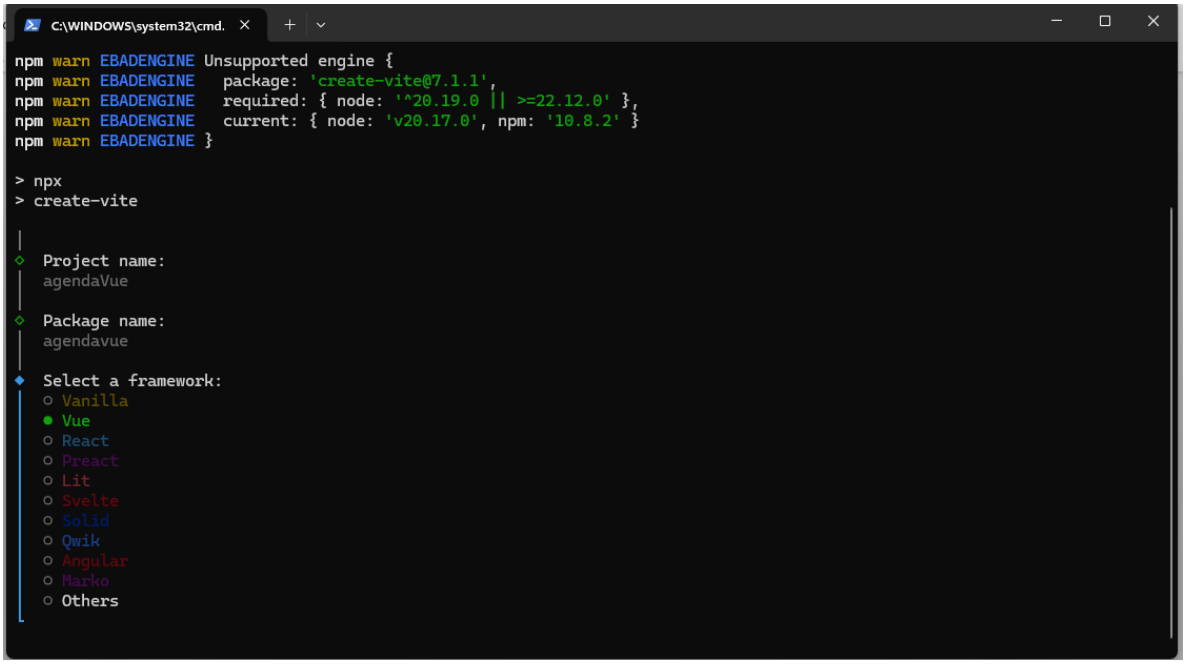
La **Composition API** es la forma moderna de trabajar con Vue 3 y se organiza principalmente en la función setup(). Todo lo que se declare dentro de setup (variables, funciones, props o estados) estará disponible para el componente.

- **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.

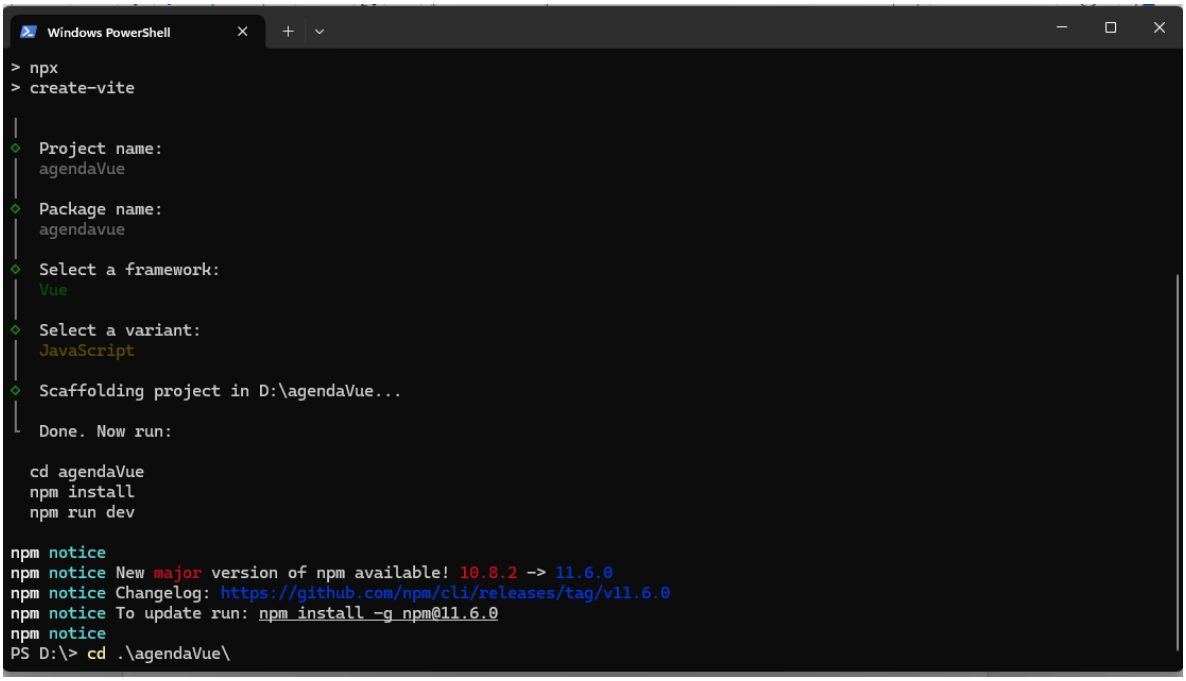
Uso de <script setup>

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.

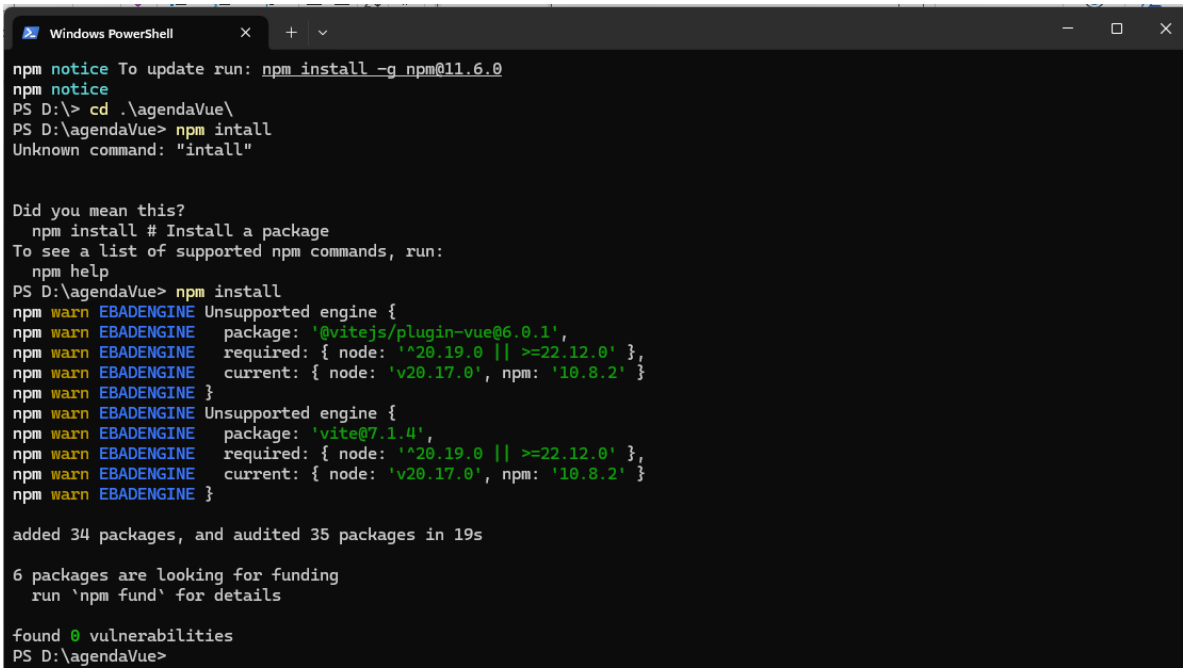
- Pasos para crear un Proyecto en Vue



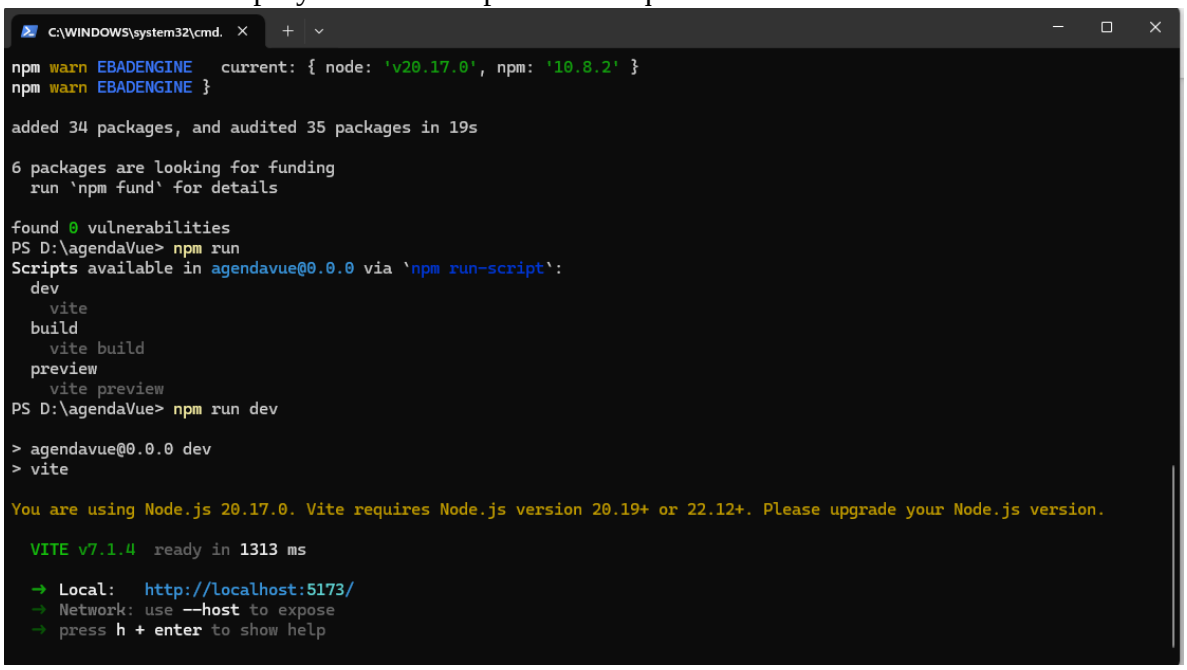
- Ya creado el Proyecto entro a la Carpeta Creada



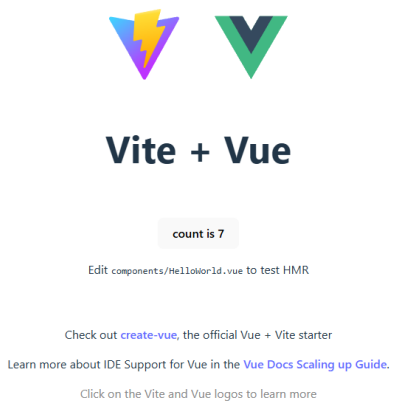
- Instalamos dependencias



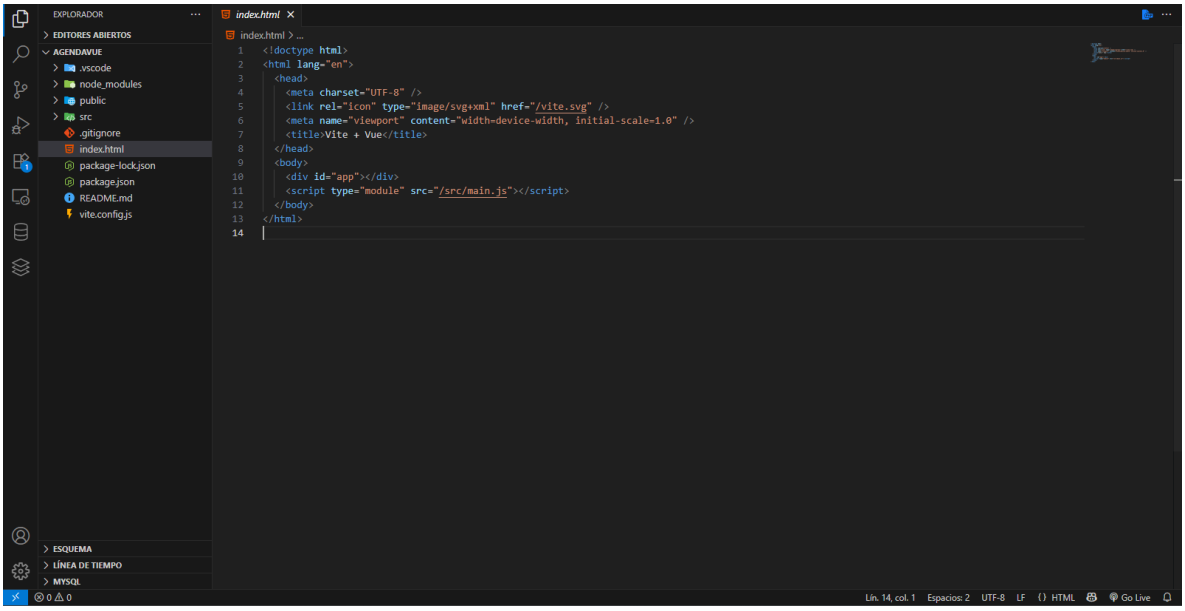
- Nos da la Url de el proyecto inicial e por defecto que seria esta



- Ejecutamos la url en el navegador y nos muestra siguiente



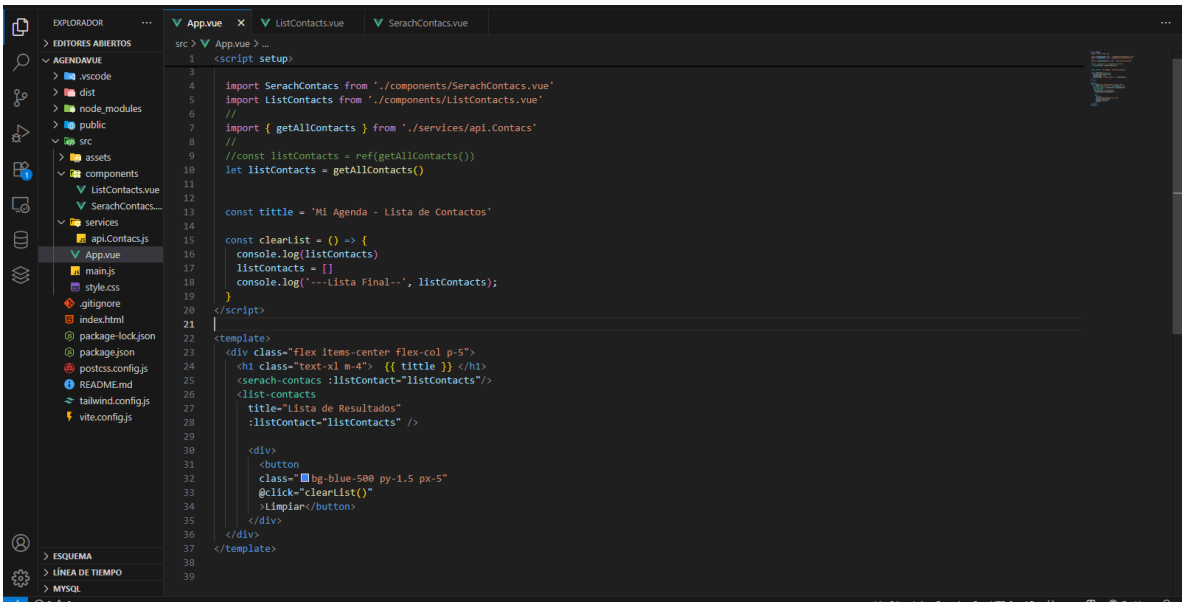
- Abrimos nuestro editor fav de código (en este abrire el proyecto creado)

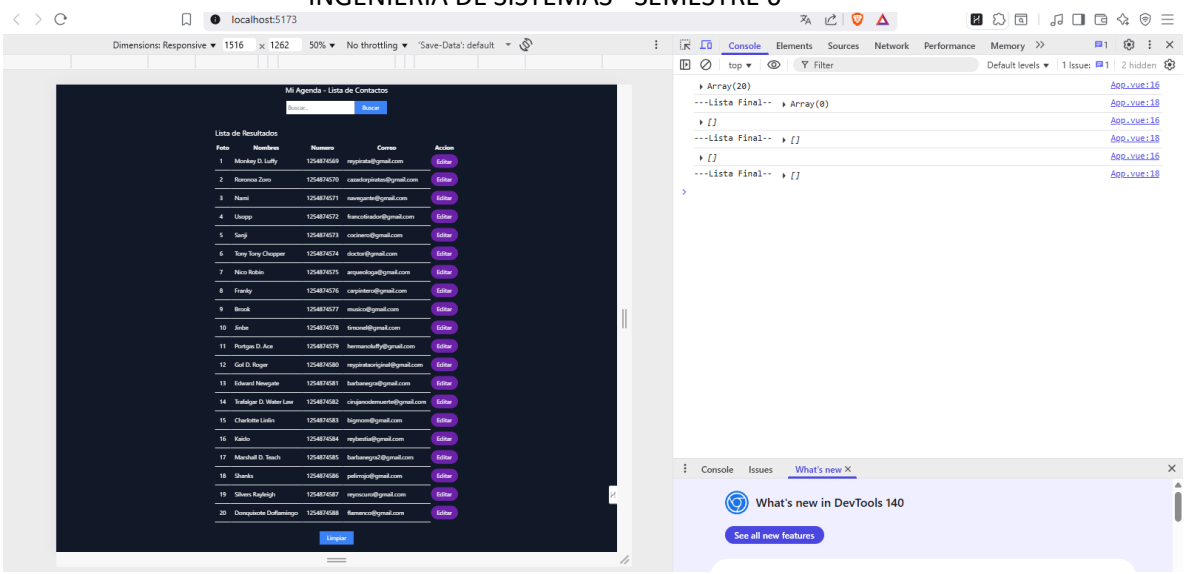


Manejo de estados

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.

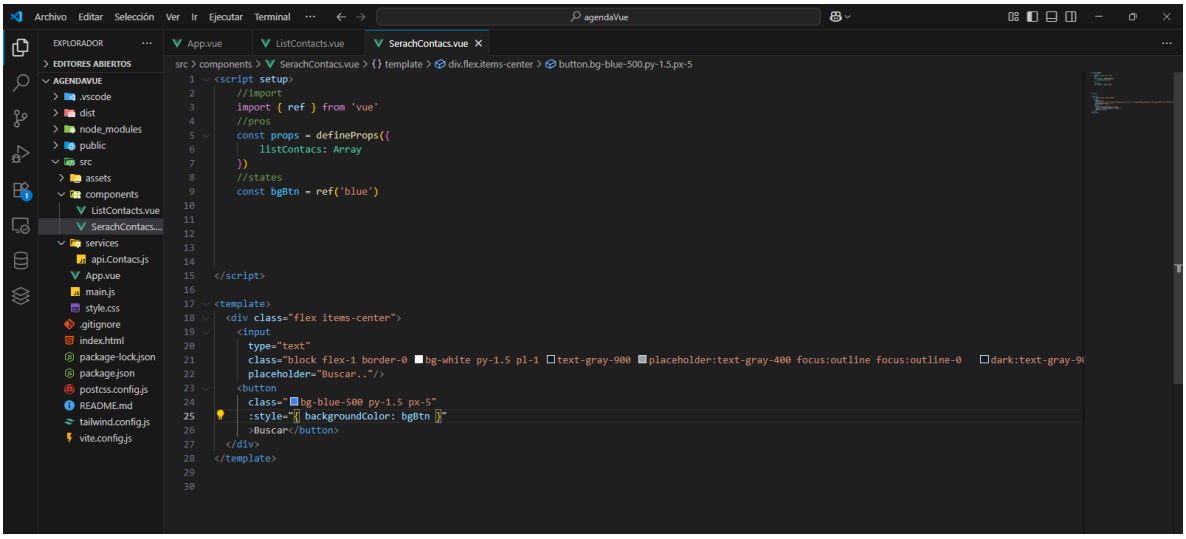


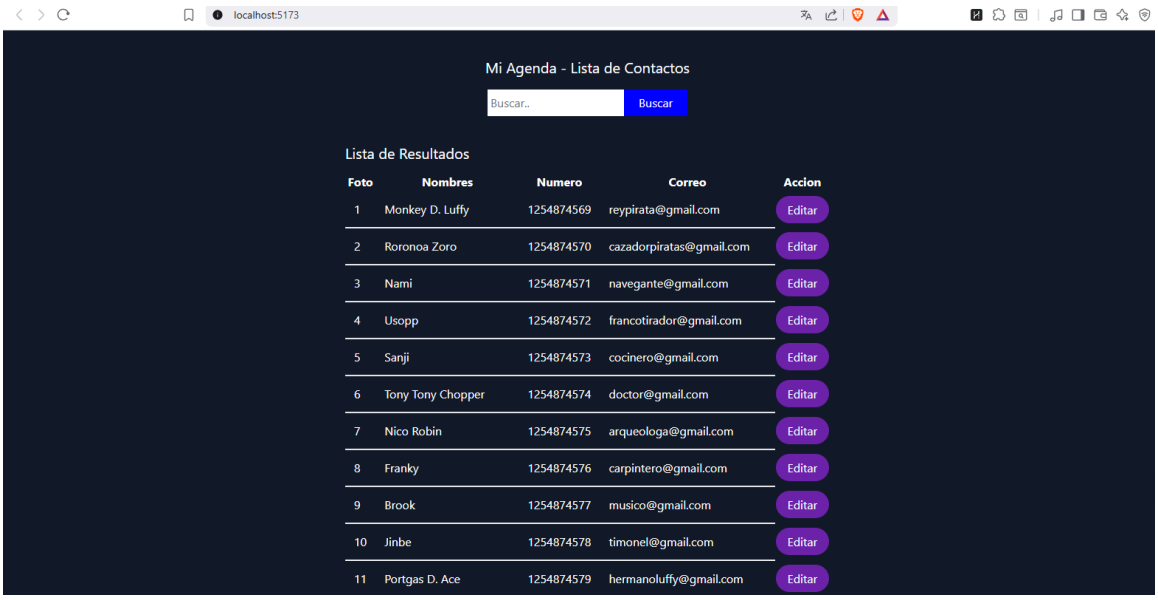


Enlace de datos (Binding)

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.

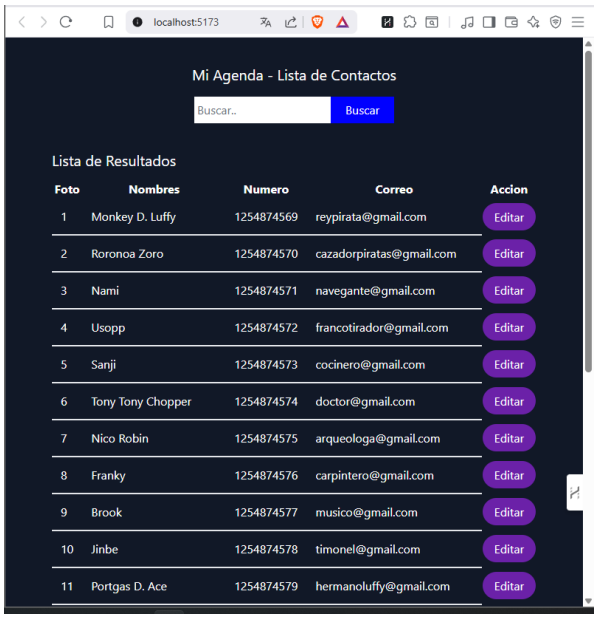


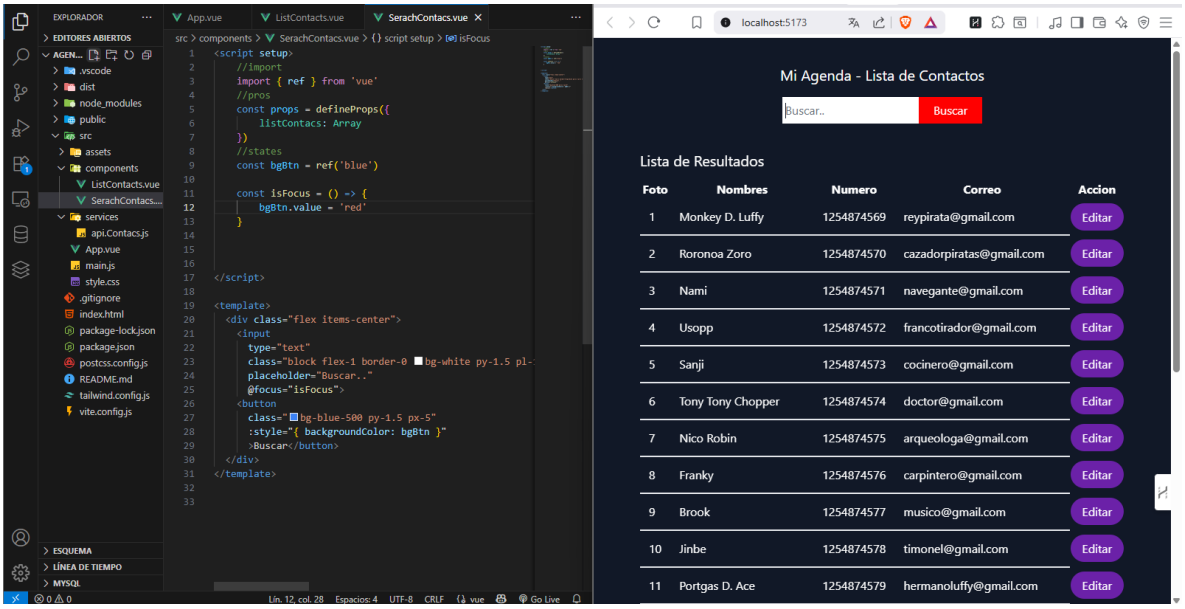


Eventos en Vue – Ejemplo con "focus"

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.

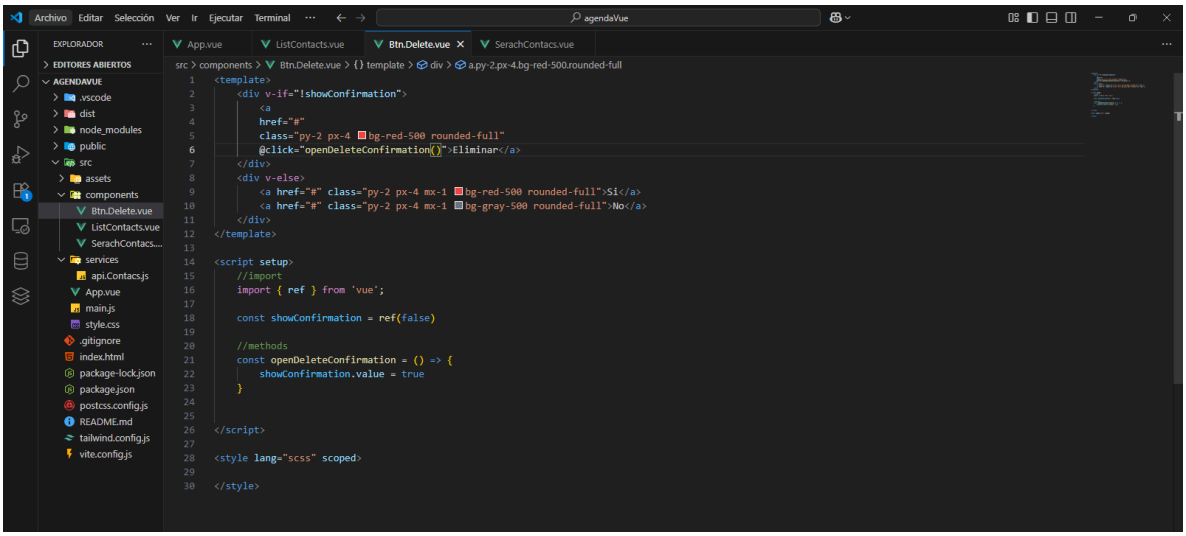
Esta parte del ejercicio permitió comprobar cómo los eventos pueden: modificar dinámicamente la apariencia o el comportamiento de un elemento, reforzando la interacción entre el usuario y la interfaz.

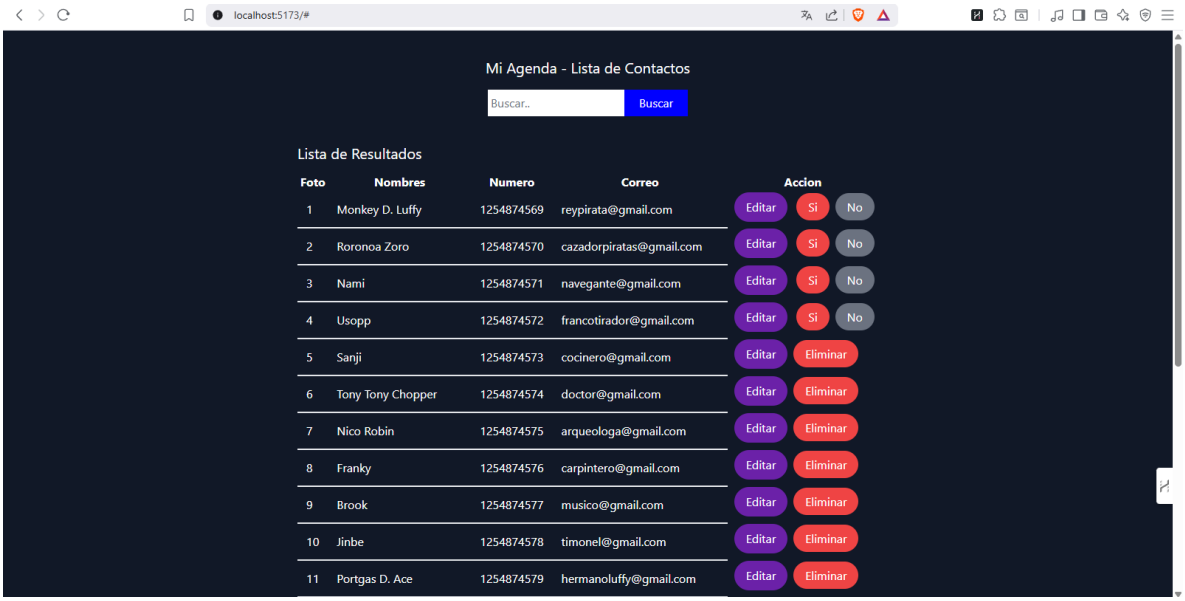




Implementación del botón de eliminar

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.

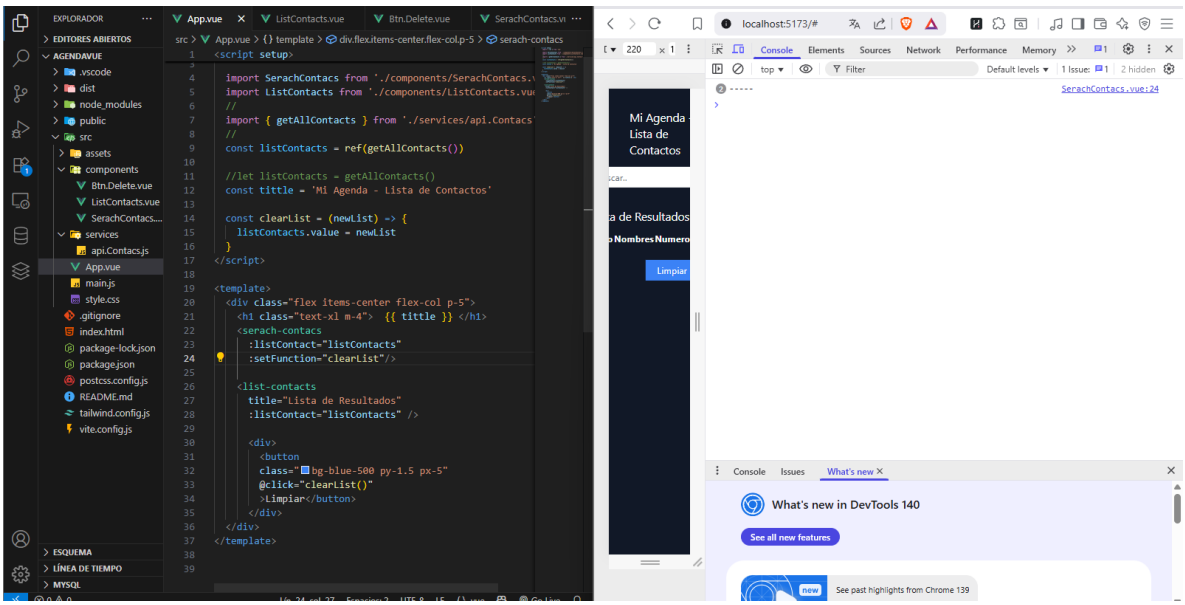


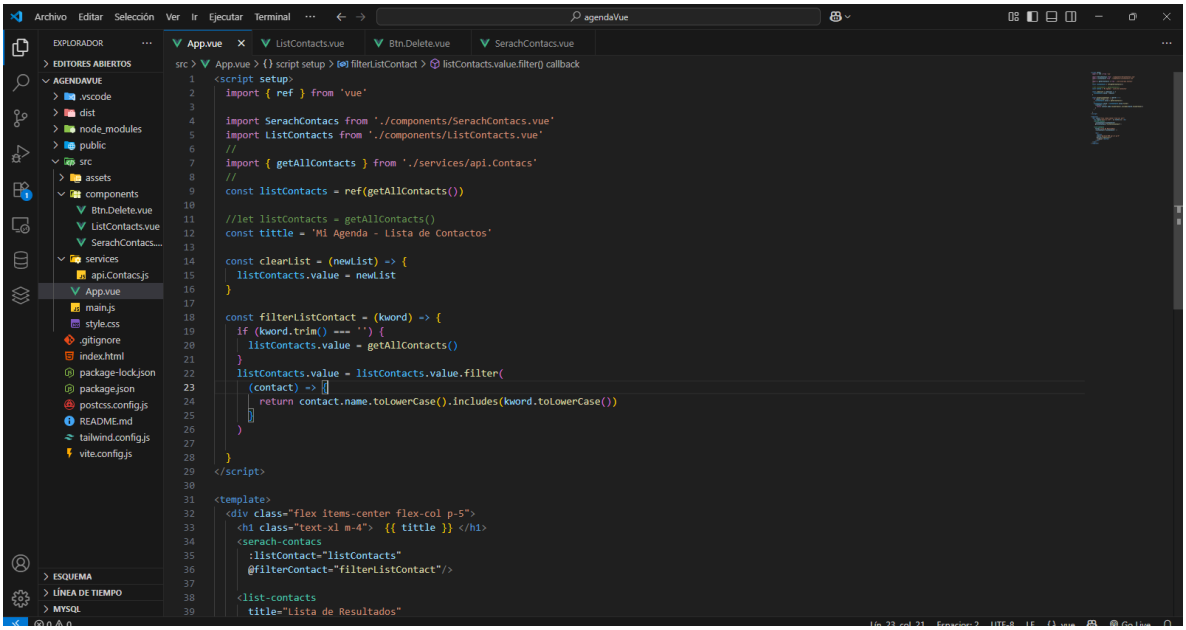


Búsqueda dinámica en la lista

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.





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.

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.

```
    }
  })
  //methods
  const totalRecompesas = () => {
      return props.listContact.reduce(
          (accumulator, contact) => accumulator + contact.recompensa, 0
      )
  }
</script>

<template>
    <div class="m-6">
      <h3 class="text-xl my-3"> {{ title }}- {{ totalRecompesas() }}</h3>
     <table class="table-fixed">
```

Mi Agenda - Lista de Contactos

Buscar.. Buscar

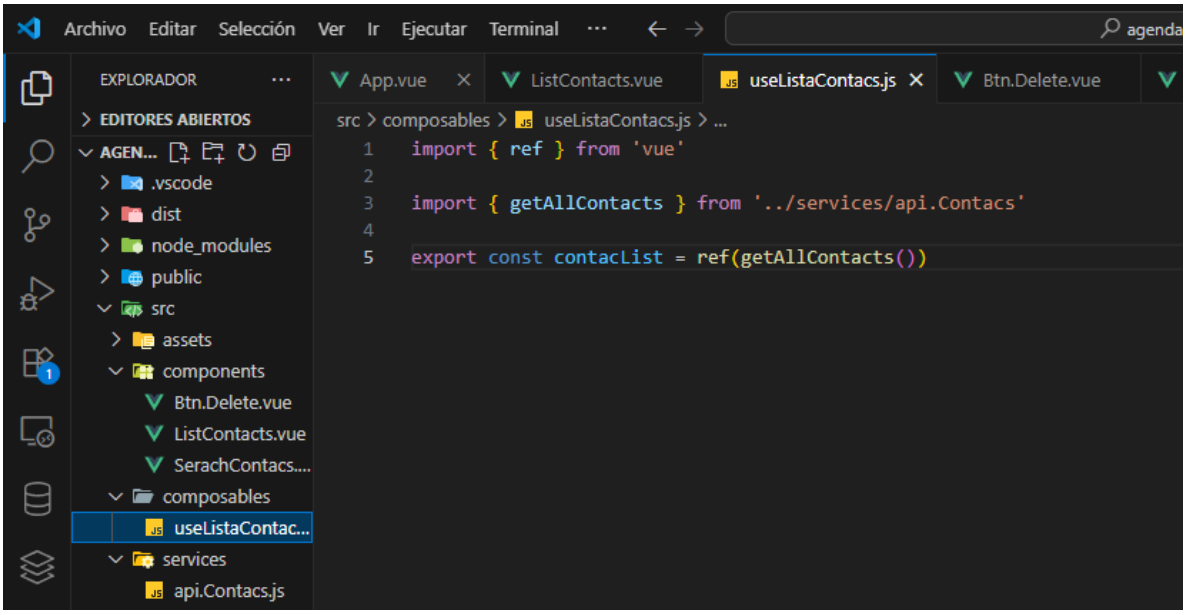
Lista de Resultados- 315.401

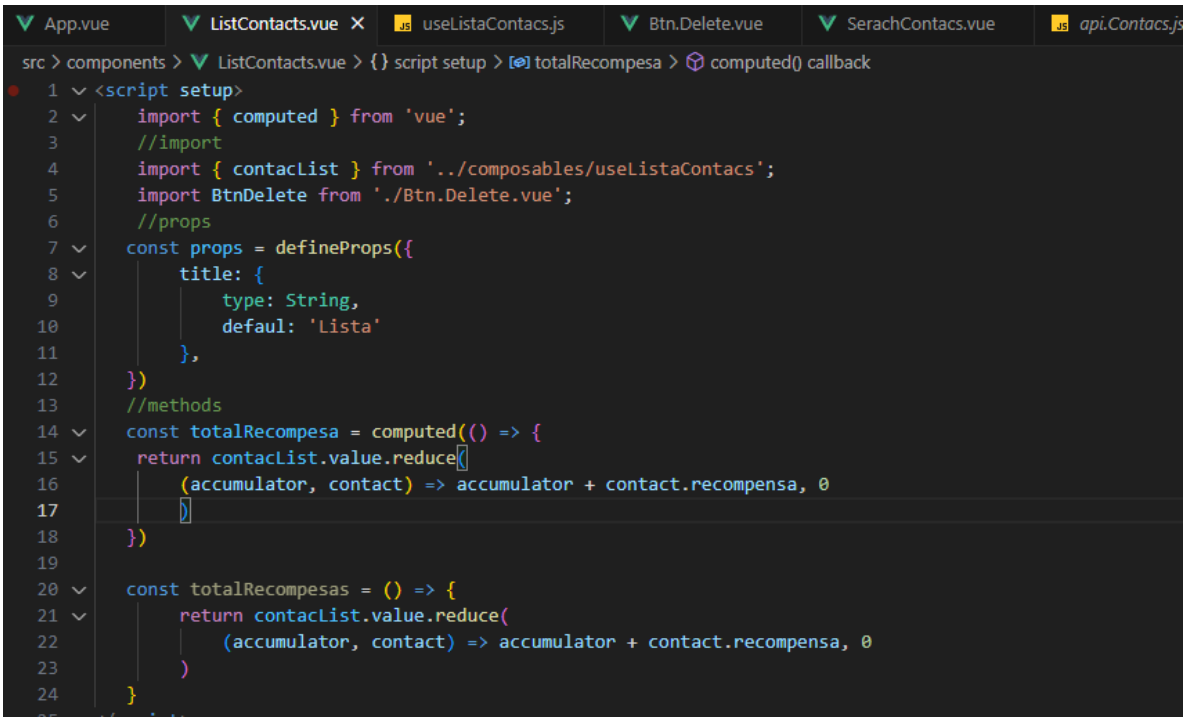
Foto	Nombres	Numero	Correo	Accion
1	Monkey D. Luffy	1254874569	reypirata@gmail.com	Editar Eliminar
2	Roronoa Zoro	1254874570	cazadorpiratas@gmail.com	Editar Eliminar
3	Nami	1254874571	navegante@gmail.com	Editar Eliminar
4	Usopp	1254874572	francotirador@gmail.com	Editar Eliminar
5	Sanji	1254874573	cocinero@gmail.com	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

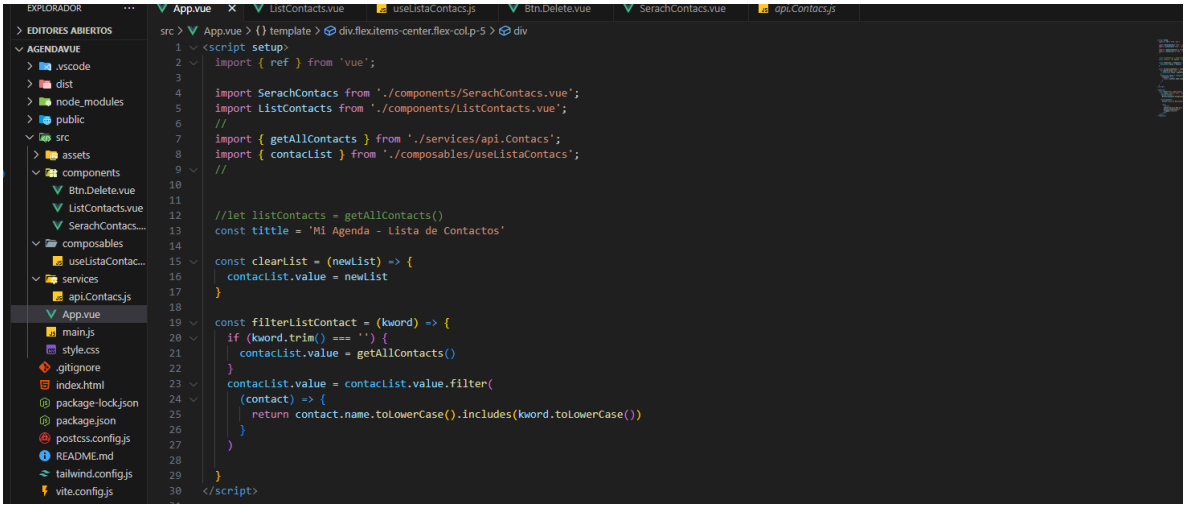
Archivos globales en el proyecto

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.



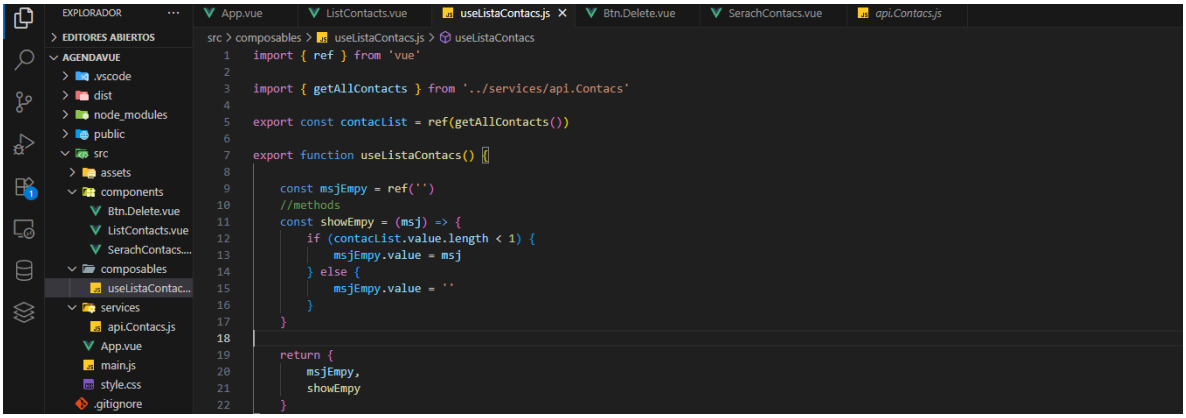


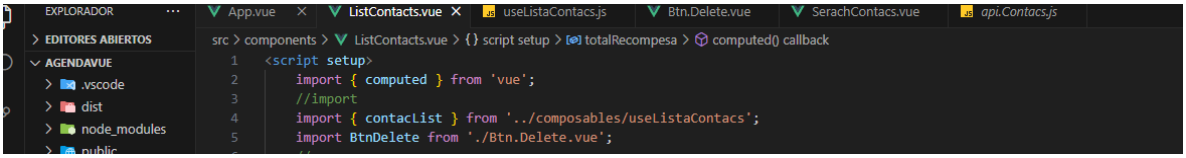


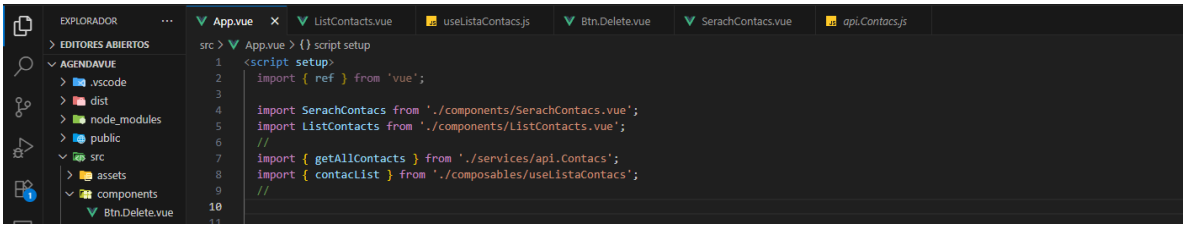
Creación de un composable para búsquedas

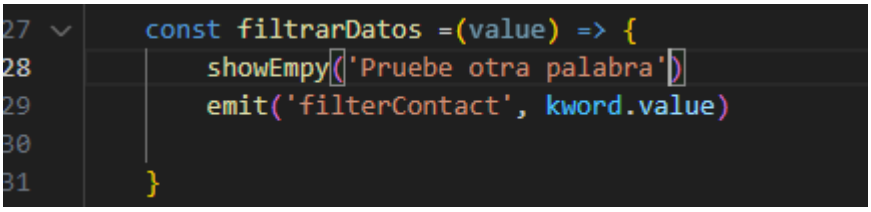
Se desarrolló un **composable** que permite gestionar el caso en el que una búsqueda no arroje resultados. Su función es detectar cuando la lista está vacía y mostrar automáticamente un mensaje al usuario, informando que no se encontraron coincidencias.

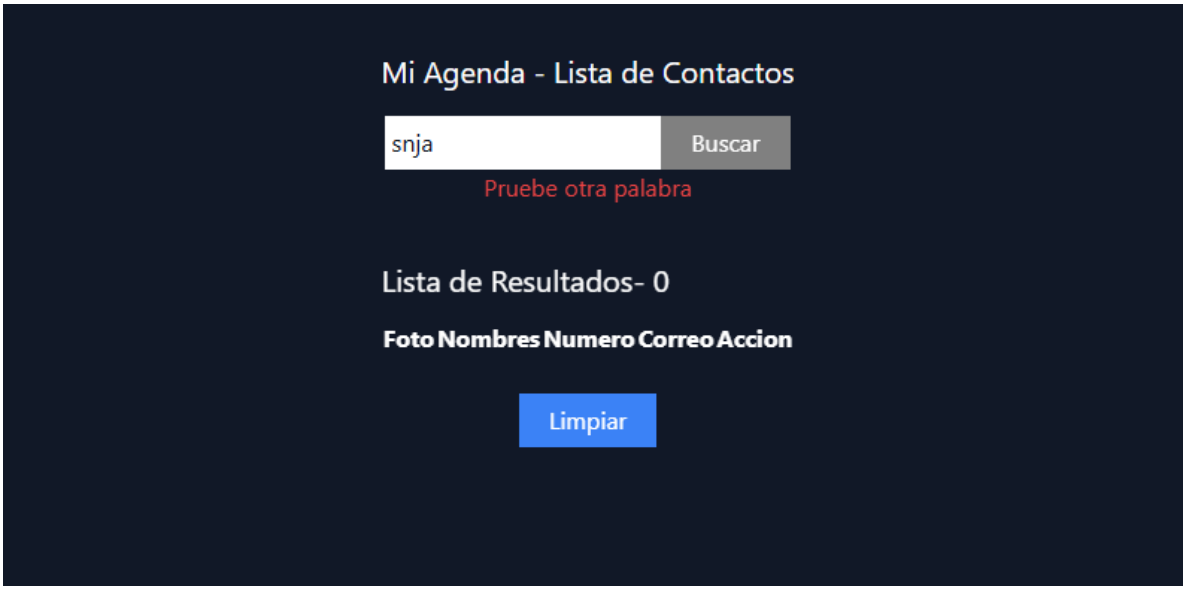
Este enfoque demuestra cómo los composables ayudan a reutilizar lógica y mantener el código más ordenado, ya que la misma funcionalidad puede aplicarse en distintos componentes sin necesidad de repetirla.



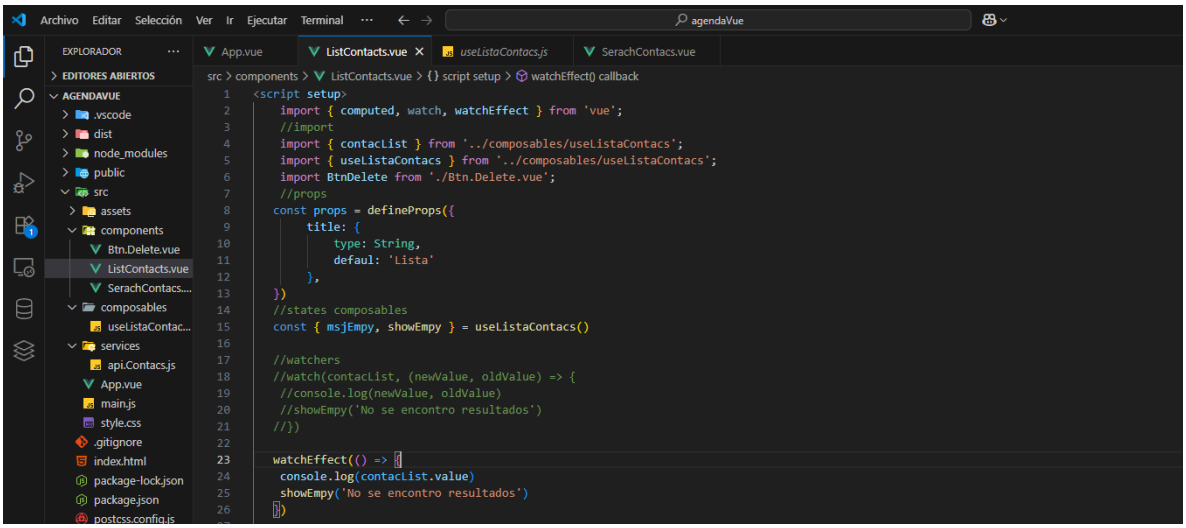


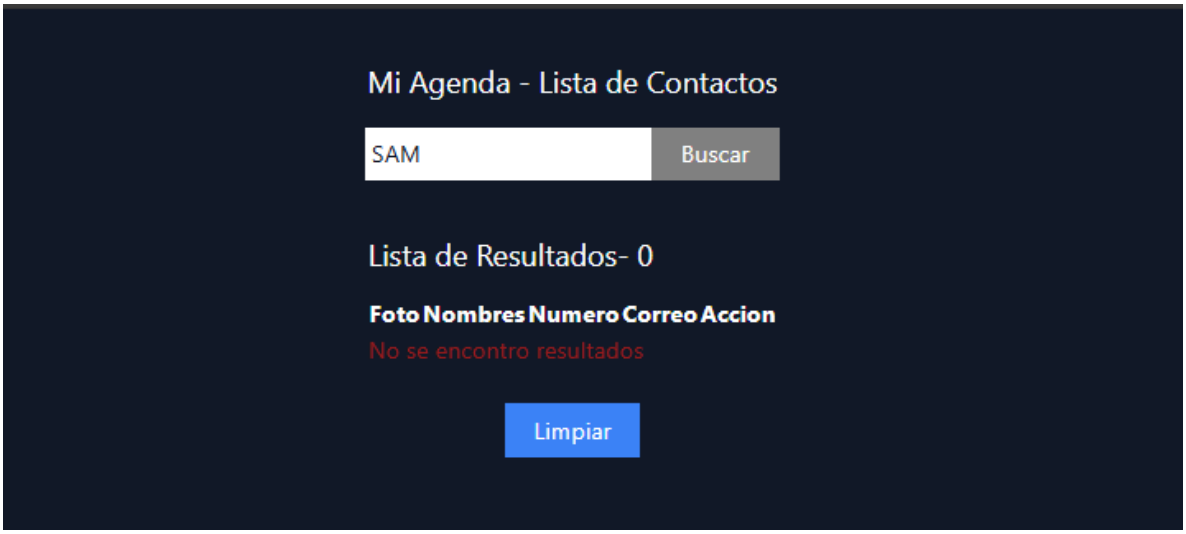






- Se hace lo mismo para listaContact




Mi Agenda - Lista de Contactos
SAM
Buscar
Lista de Resultados- 0
Foto Nombres Numero Correo Accion
No se encontro resultados
Limpiar