TALLER 1

LEYDER ANDRES CERON SALAZAR

ADRIAN IZQUIERDO

UNIVERSIDAD DEL PUTUMAYO

SEXTO SEMESTRE

MOCOA - PUTUMAYO

2025

**Entorno y herramientas**

* Vue 3: Es un framework de javascript para construir interfaces de usuarios
* Vite.js: generador de entornos frontend rápido y moderno.
* Node.js: es requisito para usar vuejs ya que al instalar node.js se instala este
* Project IDX de Google: alternativa online para crear proyectos Vue.js

**Creación del proyecto**

* npm create vite@latest → inicializar proyecto.
* Selección de framework: Vue.
* Lenguaje: JavaScript.
* npm install este instala dependencias.
* npm run dev para correr en el servidor local.

**Estructura del proyecto**

* index.html
* main.js este monta la app Vue.
* App.vue componente raíz de todo
* Carpeta src/ es la lógica y componentes principales.
* Uso de <script>, <template>, <style>.
* Script lógica js
* Template el html
* Style el css

**Interacción básica**

* Variables en <script> renderizadas con {{ }}.
* Eventos con @click="funcion".
* Ejemplo: buscarContacto().

**Tailwind CSS**

* Instalación de tailwind, postcss, autoprefixer.
* Configuración en tailwind.config.js.
* Uso de flex: p-5, text-2xl, bg-gray-900)
* Dark Mode: class="dark" en <body>.

**Construcción de la agenda**

* Input + botón de búsqueda.
* Tablas y listas con v-for.

**Diseño con Tailwind**

* Clases de tablas table-fixed, border-b-2.
* Botones py-2 px-4, bg-purple-800, rounded-full.

**Problema de código**

* Todo en App.vue es poco mantenible.
* Y su solución es: **dividir en componentes**.

**Creación de componentes**

* Carpeta components/.
* Ejemplo: SearchContacts.vue, ListContacts.vue.
* Estructura básica: <script setup>, <template>, <style>.

**Convenciones de nombres**

* PascalCase en archivos.
* kebab-case al usarlos <list-contacts>.

**Reutilización de componentes**

* Un mismo componente en distintas vistas.

**Comunicación entre componentes**

* **Props** son datos del padre al hijo “defineProps”.
* Props con tipo y valor por defecto.

**Binding (:)**

* Enlazar valores JS a atributos HTML.
* Ejemplo: :style="{ background: bgButton }".

**Eventos (@)**

* @click, @focus, etc. → disparan funciones.

**Confirmación de eliminación**

* v-if / v-else → mostrar botón o confirmación (Sí/No).

**Observación de estados**

* Watch: observar estado específico.
* watchEffect: observa automáticamente lo que se usa en el callback.

**Buenas prácticas**

* Usar composables para organizar y reutilizar lógica.
* No abusar de estados globales sueltos.
* Usar Pinia para proyectos medianos/grandes.