

# AMPLIACIÓN LM

## Conociendo Vue.js



Nombre del estudiante: David Romero

Asignatura: Lenguaje de Marcas

Profesor: Manuel Mauri Pajares

Título del trabajo: Ampliación LM Conociendo Vue

Documentación técnica: Instalación y puesta en marcha de un proyecto Vue.js

Documentación sobre la programación: Creación App-Recetas

Fecha de entrega: 11-06-2025

# **Índice**

## **1. Introducción**

### **1.2 ¿Por qué Vue?**

## **2. ¿Qué es Vue.js?**

### **2.1 Programación reactiva**

### **2.2 Conceptos clave en Vue**

## **3. Requisitos de instalación**

### **3.1 Software y versiones utilizadas**

### **3.2 Node.js**

### **3.3 Editor de código**

## **4. Guía de instalación**

## **5. Creación del proyecto Vue**

## **6. Estructura del proyecto**

### **6.1 Estructura de un componente**

## **7. Ejecución de la aplicación**

## **8. Documentación sobre la programación**

## **9. Conclusión**

# 1. Introducción

En este documento se detalla el proceso de instalación, configuración y ejecución de una aplicación básica utilizando el framework Vue.js, una de las herramientas modernas más populares para el desarrollo web. El objetivo es documentar todos los pasos necesarios para reproducir el entorno y entender cómo funciona una aplicación construida con esta tecnología.

## 1.2 ¿Por qué Vue?

Para este trabajo de ampliación se ofrecieron 3 posibles frameworks: Angular, React y Vue.

De React se podría decir que actualmente quizás sea el más utilizado teniendo una gran demanda en el mercado laboral, teniendo una curva media de aprendizaje.

Angular es más completo, por tanto, más complejo (como suele pasar un poco en el mundo del desarrollo de software, mayor control → más decisiones debe tomar el desarrollador). Su estructura está basada en TypeScript. Mayor control para el que ya lo domina, la menos indicada para empezar de cero y aprender conceptos básicos de programación reactiva (flujo de datos y eventos de manera asíncrona) comunes en las 3.

Por último, y la elegida Vue.js, es amigable y tiene una suave curva de aprendizaje, sintaxis clara e ideal para un primer contacto con este paradigma de programación.

## 2. ¿Qué es Vue.js?

Vue.js es un framework progresivo (lo mismo te permite hacer una pequeña parte en tu programa, qué haces una app completa usando sus diferentes herramientas: Vue Router para la navegación, Vuex para la gestión de estado) de JavaScript utilizado para construir interfaces web dinámicas. Se basa en el uso de componentes reutilizables, cada uno con su propia lógica, estructura visual y estilos. Su sistema de reactividad es uno de sus aspectos más potentes y diferenciadores. Vue es conocido por su simplicidad, facilidad de integración y curva de aprendizaje amigable para desarrolladores principiantes e intermedios.

## 2.1 Programación reactiva

Se basa en el principio de que el estado de la aplicación se actualiza automáticamente cuando los datos cambian. Esto significa que en lugar de manipular el DOM manualmente, Vue gestiona la actualización de los elementos de la interfaz de usuario de forma eficiente.

## 2.2 Conceptos clave en Vue

1. Sistema de reactividad → Detecta cambios en los datos y actualiza la interfaz en consecuencia.
  - a. Reactividad a nivel de datos, los datos de un componente están envueltos en un proxy reactivo. Cuando se accede o modifica una propiedad del estado `data`, intercepta la operación y la registra para futuras actualizaciones.
  - b. Reactividad con el objeto, los datos declarados en la función `data()` de un componente son reactivos, si se modifica un dato en el modelo, se actualiza la interfaz sin necesidad de código extra.
  - c. Refs y reactividad profunda, a partir de Vue3 el concepto de `ref()` permite crear estados reactivos incluso fuera del `data()`. También estas referencias pueden ser profundas (deep) para detectar cambios internos en objetos o arrays.
2. Directivas reactivas → Usando atributos como `v-bind`, `v-model` y `v-for`, puedes enlazar datos al DOM de manera declarativa.
3. Componentes → Son instancias reutilizables que pueden recibir y reaccionar a cambios en los datos.
4. Propiedades computadas y métodos → Las propiedades computadas (`computed`) son funciones optimizadas para la reactividad, mientras que los métodos (`methods`) permiten ejecutar lógica sin depender del sistema reactivo.
5. Watchers → Se utilizan para observar cambios en datos específicos y ejecutar funciones cuando estos cambian.
6. Estado global y Vuex/Pinia → Para gestionar el estado compartido entre múltiples componentes, Vue ofrece soluciones como Vuex (antes) y Pinia (actualmente).

### 3. Requisitos previos

Para poder seguir con el desarrollo de la tarea, es necesario tener instaladas las siguientes herramientas: Node.js y Visual Studio Code (Esta es una elección personal de desarrollador, en definitiva lo que realmente necesitas es un editor de código. Pero VS Code es bastante popular por razones de peso, ya que es potente y muy personalizable (plugins) a gusto o necesidad del desarrollador o trabajo).

#### 3.1 Software y versiones utilizadas.

Software	Versión
Sistema Operativo Windows 11 Pro	24H2
Editor de código VS Code	1.100.3
Extensiones VS Code Vue.js	ESLint 3.0.10 Prettier 11.0.0 Vue VSCode Snippets 3.2.0 Vue-Official Volar 2.2.10
Entorno de ejecución Node.js	v22.16.0
Gestor de paquetes npm	10.9.2
Vue	3
Navegador Web Firefox	139.0.1(64-bit)
Netlify	Web

#### 3.2 Node.js

Es un entorno de ejecución para JavaScript que permite ejecutar código fuera del navegador. Es fundamental en el desarrollo moderno ya que permite usar herramientas como Node Package Manager (usada en esta tarea y de ahora en adelante en este documento “npm”, ejecuta servidores locales y además gestiona dependencias y librerías (como las de Vue). A continuación, se detalla una guía de instalación de node.js:

1. Acceder a: <https://nodejs.org>.

2. Descargar la versión recomendada para la mayoría de usuarios (LTS).
3. Seguir los pasos del instalador.
4. Marcar la opción de agregar Node.js al PATH (suele venir por defecto, pero es importante asegurarse que esté marcada para poder usar node y npm desde la terminal en cualquier carpeta).

Para verificar que todo fue correcto, abrimos una terminal (CMD o PowerShell) y ejecutamos estos comandos: 'node -v' para comprobar node y

'npm -v' para comprobar npm (en la terminal van sin comilla). En ambos casos debe mostrar el número de versión si todo fue correcto.

```
PS C:\Users\david\Desktop> node -v
v22.16.0
PS C:\Users\david\Desktop> npm -v
10.9.2
PS C:\Users\david\Desktop>
```

Aunque no se programó directamente en Node.js, se utilizó como entorno necesario para instalar y ejecutar herramientas modernas del ecosistema JavaScript, como Vite y Vue. Gracias a Node.js y su gestor de paquetes npm, fue posible crear el proyecto, instalar dependencias y lanzar un servidor local para probar la aplicación.

## 3.2 Editor de código

VS Code es ligero, gratuito, multiplataforma y extensible. Además, tiene terminal integrada, resaltado de sintaxis y soporte para framework modernos. A continuación, se detalla una guía para la instalación de extensiones necesarias y recomendadas para VS Code de Vue:

(Paso 0 = Instalar VS Code Descargar desde: <https://code.visualstudio.com>, en este caso ya se cuenta con su instalación, por tanto se muestran las extensiones necesarias. Pero vamos, este paso es tan sencillo como descargar el instalador de la web oficial, ejecutar el archivo descargado y seguir los pasos de instalación.)

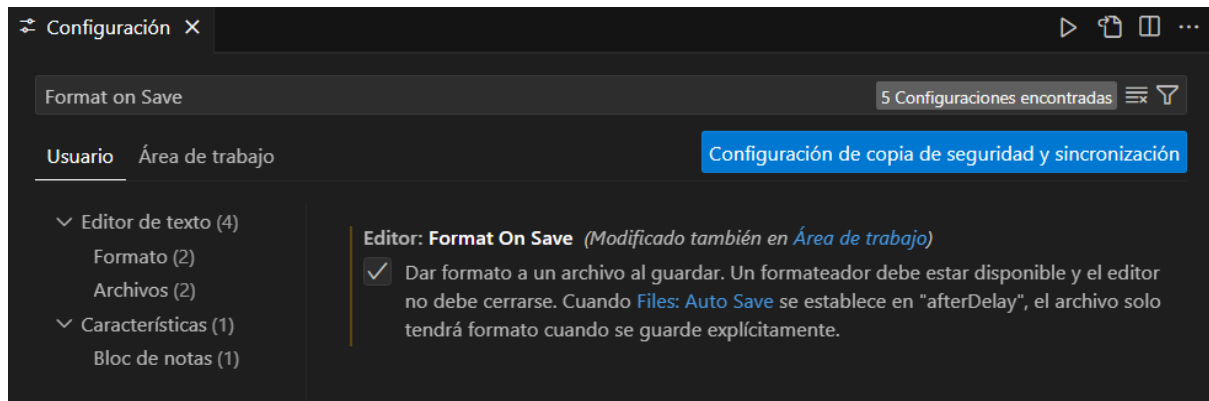
Ir al panel de extensiones de VS Code, buscar e instalar lo siguiente:

- ☒ Vue-official(Volar).
- ☒ Vue VSCode Snippets (para autocompletar componentes y directivas).
- ☒ ESLint (opcional, ayuda con errores y estilos de código).

- ☒ Prettier (opcional, para formateo automático del código: tabulaciones, cierres de llaves, corchetes, etc...).

Para activar la terminal integrada, en VS Code, ir a “Terminal” → “New Terminal”.

Para configurar formateo por defecto al guardar (con Prettier), en VS Code, ir a “Settings” → activar “Format on Save” (En la barra de búsqueda, lo escribes y te aparece como resultado de la búsqueda).



## 4. Guía de instalación

Primero creamos un nuevo proyecto desde la terminal en VS Code ejecutando lo siguiente → **npm create vite@latest prueba2** esto creará una carpeta llamada prueba2. Veámoslo desglosado:

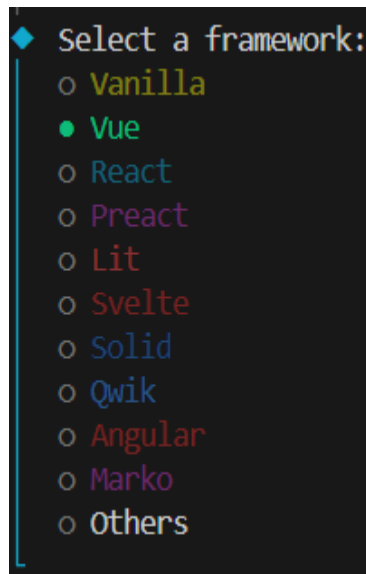
**npm** → Es el administrador de paquetes de Node.js. Se usa para instalar y gestionar herramientas en proyectos de JavaScript.

**create vite@latest** → Le dice a npm que debe ejecutar el script de instalación de Vite (una herramienta para construir aplicaciones web modernas) con la versión más reciente (@latest).

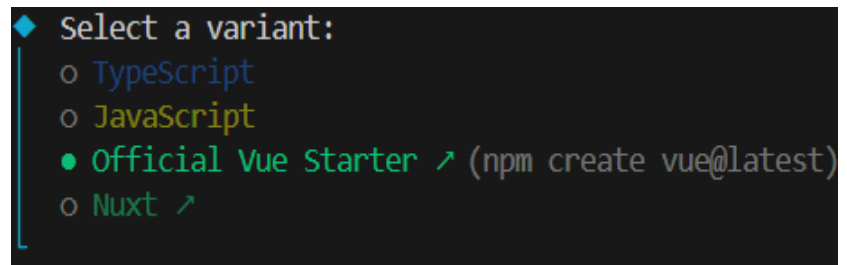
**prueba2** → Es el nombre que le estás dando a tu nuevo proyecto. Esto creará una carpeta llamada prueba2 donde se instalarán todos los archivos del proyecto.

```
PS C:\Users\david\Veú> npm create vite@latest prueba2
> npx
> create-vite prueba2
```

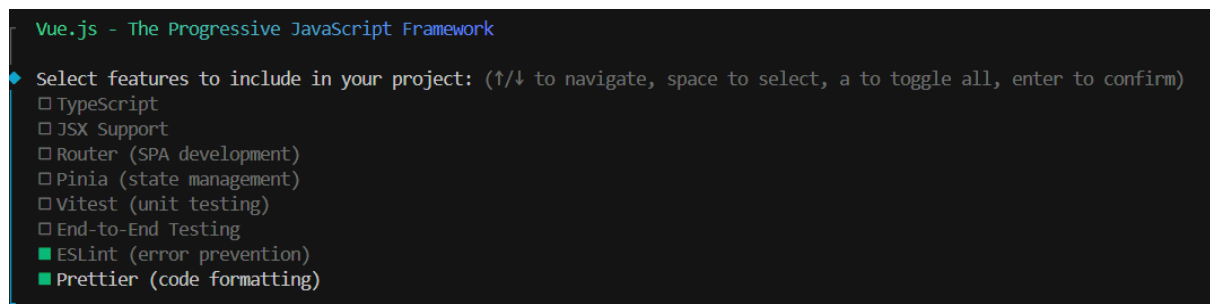
Tras pulsar Enter, comenzará el proceso de creación, el cual contiene varias preguntas:



← Seleccionamos Vue como Frameworks



☒ Seleccionamos Official Vue Starter, para comenzar con las opciones del arranque oficial de Vue. Se nos abren las opciones para elegir en la siguiente imagen.



☒ Como este es un proyecto básico, solo es necesario marcar las 2 últimas, son opcionales pero ayudan. Por defecto, Vue usa JS. El resto de opciones significan:

- ☐ TypeScript → permite configurar un proyecto Vue con soporte completo para TypeScript. Esto significa que puedes aprovechar la seguridad de tipos, la autocompletación avanzada en editores de código y una mejor detección de errores en el tiempo de desarrollo.
- ☐ JSX Support → permite habilitar el uso de JSX en Vue. JSX es una extensión de sintaxis para JavaScript que facilita la escritura de componentes de manera más declarativa y flexible, similar a React.
- ☐ Router (SPA development) → habilita **Vue Router**, la solución oficial de enrutamiento para Vue.js. . Esto es esencial para desarrollar **Single Page Applications (SPA)**, donde la navegación entre páginas ocurre sin recargar la web.



- ☐ Pinia (state management) → habilita **Pinia**, la librería oficial de gestión de estado para Vue.js. . Es una alternativa moderna y más sencilla a Vuex, diseñada para Vue 3.
- ☐ Vitest (unit testing) → habilita **Vitest**, un framework de pruebas diseñado específicamente para proyectos con **Vite** y Vue.js. . Es una alternativa moderna y rápida a Jest, optimizada para el ecosistema de Vue.
- ☐ End-To-End Testing → habilita herramientas para probar toda la aplicación simulando la interacción de un usuario real. Esto es clave para garantizar que el sistema funcione correctamente en un entorno real.
- ☒ ESLint (error prevention) → habilita **ESLint**, una herramienta para detectar y corregir errores en el código de JavaScript y Vue.js. . Su propósito es mejorar la calidad del código al aplicar reglas de estilo y evitar errores comunes.
- ☒ Prettier (code formatting) → habilita **Prettier**, una herramienta de formateo automático de código que ayuda a mantener un estilo consistente en tu proyecto Vue.

```
Scaffolding project in C:\Users\david\Ve\prueba2...
|
| Done. Now run:
|
| cd prueba2
| npm install
| npm run format
| npm run dev
|
| Optional: Initialize Git in your project directory with:
|
| git init && git add -A && git commit -m "initial commit"
|
PS C:\Users\david\Ve>
```

Una vez finalizada la creación, nos muestra los siguientes pasos a seguir, son:

1. `cd mi-app` → para colocarnos en la carpeta del proyecto Vue que se acaba de crear.

2. npm install → para instalar el proyecto en la carpeta.

```
PS C:\Users\david\Ve\prueba2> npm install

added 244 packages, and audited 245 packages in 17s

74 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

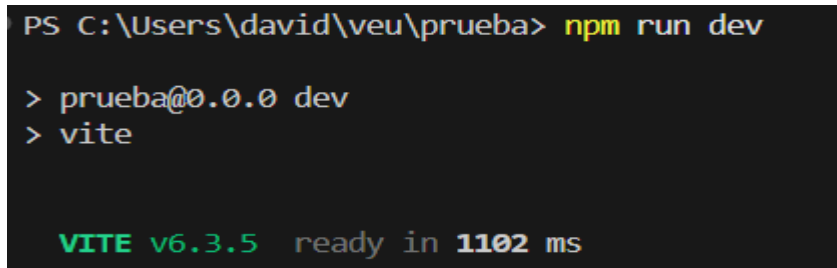
3. npm run format → para configurar y activar Prettier.

```
found 0 vulnerabilities
● PS C:\Users\david\veu\prueba> npm run format

> prueba@0.0.0 format
> prettier --write src/

src/App.vue 113ms (unchanged)
src/assets/base.css 15ms (unchanged)
src/assets/main.css 6ms (unchanged)
src/components/HelloWorld.vue 33ms (unchanged)
src/components/icons/IconCommunity.vue 4ms (unchanged)
src/components/icons/IconDocumentation.vue 2ms (unchanged)
src/components/icons/IconEcosystem.vue 5ms (unchanged)
src/components/icons/IconSupport.vue 4ms (unchanged)
src/components/icons/IconTooling.vue 3ms (unchanged)
src/components/TheWelcome.vue 29ms
src/components/WelcomeItem.vue 12ms (unchanged)
src/main.js 5ms (unchanged)
○ PS C:\Users\david\veu\prueba> 
```

4. `npm run dev` → para que se ejecute la app (tal como está, ya podemos comprobar si todo fue correcto con una pequeña demo de Vue, haciendo click en el link disponible se abre el navegador web del SO).



```
PS C:\Users\david\veu\prueba> npm run dev  
  
> prueba@0.0.0 dev  
> vite  
  
VITE v6.3.5 ready in 1102 ms
```

5. Opcionalmente nos invita a que ejecutemos Git a nuestro proyecto antes de continuar.

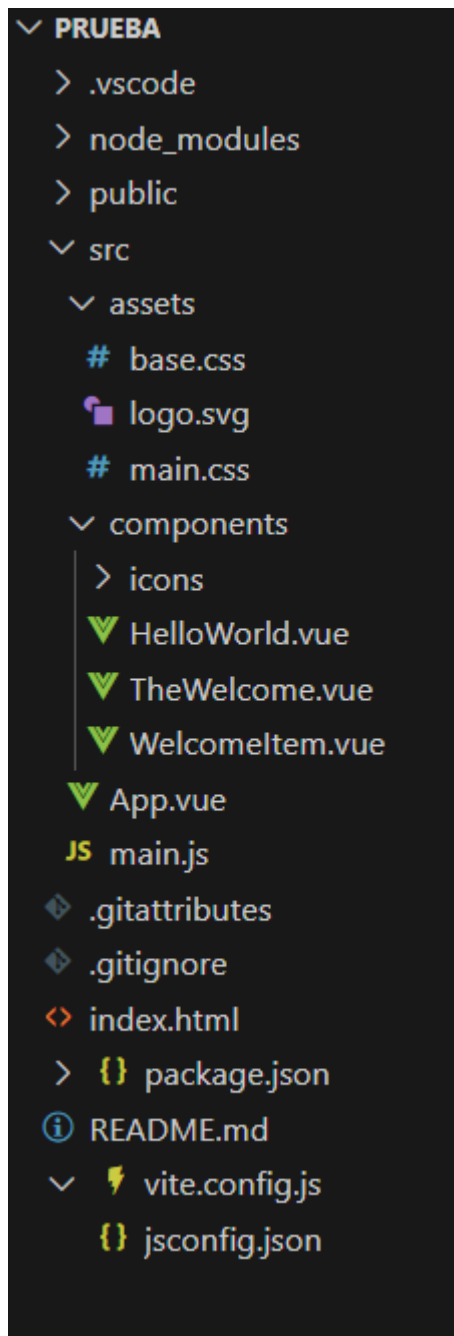
Nota Importante: Se deben tener permisos para poder crear el proyecto Vue, esto puede afectar sobre todo cuando se hace en redes corporativas o educativas y trabajas en la nube. El administrador, deberá dar permiso si no lo tuvieras.

## 5. Creación del proyecto Vue

La plantilla generada contiene lo necesario para comenzar a programar. Incluye un componente base (`App.vue`), un archivo de entrada (`main.js`) y la configuración mínima de Vite.

- **App.vue** → Es el componente base de la aplicación. En Vue, todo se organiza en componentes, y `App.vue` es el punto central desde donde se pueden incluir otros. Normalmente contiene la estructura principal de la interfaz.
- **main.js** → Es el archivo de entrada del proyecto. Aquí se inicializa Vue y se monta `App.vue` en el DOM. Es donde se configura la aplicación, se pueden importar plugins y establecer opciones globales, dicho de manera coloquial “es el puente entre el archivo Vue y el navegador”.
- Configuración mínima de **Vite** → Vite es un herramienta de desarrollo que optimiza la carga de archivos, permitiendo una experiencia más rápida. La plantilla inicial incluye la configuración básica para empezar sin preocuparte por detalles como compilación y optimización.

## 6. Estructura del proyecto



public → Archivos estáticos (favicon, imágenes, etc).

src → Código fuente de la aplicación.

assets → Recursos como imágenes y estilos globales.

components → Carpeta donde guardas tus componentes Vue.

App.vue → Componente raíz de la aplicación.

main.js → Punto de entrada y configuración de Vue.

index.html → Archivo base de la aplicación (HTML principal).

package.json → Lista de dependencias y scripts del proyecto.

vite.config.js → Configuración de Vite (opcional, por defecto ya funciona).

Archivos importantes:

index.html: Página HTML principal donde Vue se monta.

App.vue: Componente raíz con HTML, lógica y estilos.

main.js: Punto de inicio del código JavaScript.

## 6.1 Estructura de un componente

Un componente Vue tiene 3 bloques principales dentro de un archivo .vue. Si bien, ha habido un cambio bastante significativo de Vue 2 a Vue 3 siendo la parte del script la más afectada. Paso de gestionar con la API de OPCIONES (Vue 2), a gestionarse con la API de COMPOSICIÓN (Vue 3). En muchos sitios, aún se usa y se ven manuales con Vue 2, pero ya que este lo hice con Vue 3, todo estará enfocado a Vue 3:

1. `<script setup>` `</script>` Aquí defines la lógica del componente, con 5 puntos clave:
  - a. API de Composición (`script setup`)
  - b. Manejo del Estado (`ref()` y `reactive()` → Permiten actualizar automáticamente la interfaz cuando los datos cambian)
  - c. Comunicación entre Componentes (`defineProps()` y `defineEmits()` → Facilitan la interacción entre componentes sin modificar estados directamente)
  - d. Ciclo de Vida (`onMounted()` → Permite realizar acciones iniciales, como obtener datos o modificar variables)
  - e. Importaciones y Modularidad (`import` → Permite dividir el código en archivos más pequeños, mejorando su reutilización)
2. `<template>` `</template>` Es la parte visual, el HTML que se va a mostrar en la página. Aquí usas sintaxis especial de Vue para enlazar datos, mostrar listas, manejar eventos, etc. Las principales funciones son:
  - a. `v-model` → Actualiza datos y estados automáticamente.
  - b. `v-for` → Bucle para recorrer los elementos de una lista.
  - c. `v-if` y `v-else` → Mostrar o esconder contenido según condición.
  - d. `v-bind` (`:`) → Enlaza atributos (`imagen`, `href: url`, `estilos`, etc.) a variables. El paréntesis indica la forma abreviada y recomendada para su uso. [`v-bind` = `:`]
  - e. `v-on` (`@`) → Escucha eventos (`clicks`, `pasadas del ratón`, `tecla presionada`, `envío de formularios`, etc). El paréntesis indica la forma abreviada y recomendada para su uso. [`v-on` = `@`]
3. `<style>` `</style>` Opcional, sirve para poner los estilos CSS que afectarán a ese componente.

## 7. Ejecución de la aplicación

Para ejecutar la app localmente, en la terminal de VS Code escribimos (sin las comillas) y ejecutamos 'npm run dev' → esto inicia un servidor local (normalmente en http://localhost:5173) donde se puede ver y probar la app.

```
PS C:\Users\david\veu\prueba> npm run dev

> prueba@0.0.0 dev
> vite

VITE v6.3.5 ready in 1102 ms
```

Si pulsamos la vocal o + Enter se nos abre el navegador automáticamente.

O simplemente escribiendo la dirección web en el navegador. Si el acceso es desde otro dispositivo de la misma red se usa el Network http://idDelPcHuesped:5173).

Nota: Se abrirá el navegador establecido como predeterminado del sistema operativo, pero cabe reseñar que al ser una app web da igual el sistema operativo en que se ejecute, siempre que estén instalados Node.js y un navegador moderno.

### Añadido Extra:

Tras terminar la parte de programación, compile con el comando build de npm, que coje el archivo package.json del proyecto y lee el script ya preconfigurado(Con la instalación de Vite) para compilar el proyecto, y devolver una carpeta llamada **dist** en la carpeta raíz del proyecto.

```
PS C:\Users\davir\Vue> cd App-Receta
PS C:\Users\davir\Vue\AppData-Receta> npm run build

> app-receta@0.0.0 build
> vite build

vite v6.3.5 building for production...
✓ 30 modules transformed.
dist/index.html          0.43 kB | gzip: 0.28 kB
dist/assets/Gorro-DwBeLkPj.svg 4.61 kB | gzip: 1.88 kB
dist/assets/fogones-Cni31ppz.svg 6.84 kB | gzip: 1.75 kB
dist/assets/espátulas-C1cLVK48.svg 12.21 kB | gzip: 4.21 kB
dist/assets/FondoComida2-CYeK8Q56.jpg 1,689.84 kB
dist/assets/index-07Am_of7.css 3.47 kB | gzip: 1.15 kB
dist/assets/index-CTekAWMV.js 69.08 kB | gzip: 27.60 kB
✓ built in 753ms
PS C:\Users\davir\Vue\AppData-Receta>
```

Con esto, ya podemos ejecutar nuestra app de manera local allá donde tengamos una copia (y además de Node.js) sin necesidad de tener Vue3 abierto. Con el comando server de npm, que simula una conexión a un servidor mostrando la página de manera real en el navegador.

```
PS C:\Users\davir\Vue\App-Receta> npx serve dist/
```

```
- Local:    http://localhost:3000
- Network:  http://192.168.1.131:3000

Copied local address to clipboard!
```

Haciendo click en el enlace o copiando la url local en el navegador se abre la App-Receta.

Tras ello, pensando en subir el proyecto a GitHub, me topé con una herramienta web (compatible con GitHub) Netlify, que te permite subir tu app (previo registro gratuito) y la alojan en un servidor para que pueda verse en internet. Este es el enlace a [App-Receta](#)

## 8. Documentación sobre la programación

Esta es una guía informativa de apoyo sobre la estructura, sintaxis y funciones que se usan en Vue 3, ya que los archivos entregados, de la App, llevan sus respectivos comentarios cada uno.

En esta app, al usar Vite y su Official Started te crea todo el proyecto, por lo que la estructura de carpetas viene ya establecida, por lo que aparte de borrar algunos archivos demo, el tema de archivos de configuración (casi tan importante, y más difícil para el principiante, como escribir el código) no hay que tocarlos en esta ocasión. Así como, el index.html y main.js que también vienen listos para ser usados. Recalcar el uso de sintaxis de Vue 3, con cambios significativos con respecto a Vue 2.

En Vue todo gira en torno a los Componentes, que son pequeñas o grandes partes aisladas, y por tanto reutilizables, que tienen su propia configuración de código y se agrupan con otras, para formar la app. Los componentes utilizados son:

- Componentes Icon: Utilizados para contener las imágenes necesarias de en la app, menos el fondo de pantalla. En total son 4:
  1. IconCubiertos.Vue
  2. IconEspatulas.Vue
  3. IconFogones.Vue
  4. IconGorro.vue

Todos ellos son utilizados en el Componente TituloRecetas.vue

- RecetaCocina.vue es tratado como un Objeto, se define con estos 5 atributos y sus finalidades:
  1. Nombre de la receta.
  2. Descripción breve sobre la receta.
  3. Tiempo de realización.
  4. Dificultad que conlleva prepararla.
  5. Enlace que ofrece la web externa con la receta en cuestión.
- TituloRecetas.vue contiene el título y los iconos del header utilizado en index.html.
- FormularioReceta.vue es el componente que alberga el formulario que requiere los datos necesarios para añadir una nueva receta, mostrado en el index.html. Interactúa de forma reactiva con el componente principal App.vue.
- ComentariosReceta.vue este componente genera, almacena y muestra un listado de los comentarios que se realicen sobre la receta. Utiliza un array y un bucle v-for para ello. Este componente es usado en el componente ListadoRecetas.vue
- ListadoRecetas.vue componente que se encarga de mostrar tanto el componente RecetaCocina.vue con su respectivo componente ComentariosReceta.vue.
- App.vue es el componente principal, el que se ejecuta en main.js para que se inicie en el index.html.



## Descripción y ejemplos de uso de comandos de Vite:

En el <template> →

```
<input v-model="nuevoComentario" placeholder="Escribe un comentario..." />
```

v-model se encarga de entrelazar los datos en ambas direcciones haciendo que cualquier dato introducido en el input por la web se vea reflejado automáticamente en la variable “nuevoComentario”.

```
<li v-for="(comentario, index) in comentarios" :key="index">
  {{ comentario }}</li>
```

v-for es el bucle de Vue. se basa en índices para recorrer los array, se usa :key con el index de referencia también para enlazar atributos con la variable.

```
<a v-if="esUrlValida(receta.enlace)" :href="receta.enlace"
  target="_blank" class="boton-enlace">Ver receta completa</a>
<p v-else class="errorEnlace">Enlace No Válido</p>
```

v-if, v-else y v-blind o de forma abreviada y recomendada ( : )

Los condicionales, bueno funcionan un poco como el resto de condicionales if en programación(admiten operadores lógicos en la condición, v-else-if, valoran una condición y en función de si se cumple (v-if) ejecutan un código o si no se cumple (v-else) ejecutan otro. En este ejemplo, se usa la condición sobre una función que comprueba si la url existe o no y devuelve true si existe y false si no.

v-blind se encarga de enlazar atributos a variables, en el código del ejemplo vemos cómo se unen la etiqueta <a></a> para un enlace web con el atributo receta.enlace del Componente RecetaCocina.

```
<FormularioReceta @agregar-receta="agregarReceta" />
```

v-on o de forma abreviada y recomendada ( @ )

Escucha en el Dom(Vue los maneja por ti) los eventos y ejecuta las acciones pertinentes cuando ocurren, es este trozo de código vemos cómo se pasa el Componente FormularioReceta en App.vue junto con la función que envía los datos mediante el emit, y cómo se asocia el evento a dicha función agregar-receta que se activa cuando es pulsado el input correspondiente del formulario.

En el <script> →

```
<script setup>
import { ref } from "vue"
import TituloRecetas from "../components/TituloRecetas.vue"
```

setup en Vue 3 se pasa a API de Composición, lo que simplifica bastante las importaciones, creación de props, emit, estados.

import se usan para importar otros componentes del proyecto o de Vue, como puede ser ref en este ejemplo, con esto ya podemos usar ref en el componente en cuestión. También vemos como se importa un componente propio, que hemos creado para el proyecto, para ser usado por otro.

```
defineProps({ receta: Array, })
```

defineProps usado para mandar datos del Componente Padre al Componente hijo. En este trozo de código, se está indicando en el componente hijo que recibirá un array llamado receta.

```
const emit = defineEmits(["agregar-receta"]);
const agregarReceta = () => {
  if (nuevaReceta.value.nombre.trim() !== "") {
    emit("agregar-receta", { ...nuevaReceta.value })
    nuevaReceta.value = { nombre: "", descripcion: "", tiempo:
    "", dificultad: "Fácil", enlace: "" }
  }
}
```

emit y funciones.

emit permite mandar datos del Componente Hijo al Componente Padre. En este trozo de código vemos cómo creamos emit y lo definimos con una función como parámetro, con esto se está compartiendo los datos con App.vue. Una vez dentro de la función se vuelve a usar emit para asociarse con la nuevaReceta creada y de la cual se mandan los datos.

funciones se generan con const nombre de la función = () =>{ cuerpo de la función}. En este trozo de código vemos como tras un primer condicional if con una condición sobre si existe la receta (basada en si existe el nombre, usando .value y .trim para eliminar espacios en los extremos) y después si se cumple, ejecuta el emit con el valor de la nueva receta. Seguidamente limpiamos la variable nuevaReceta con valores vacíos.

## 9. Conclusión

Vue.js permite crear aplicaciones web modernas de forma clara y eficiente. Aprender su instalación y puesta en marcha, así como picar algo de código, brinda una base sólida para desarrollar proyectos reales con componentes reutilizables. Esta experiencia permitió conocer herramientas como Node.js, npm y Vite, esenciales en el desarrollo frontend actual. Además herramientas como Netlify te permiten hasta exponerlas en la Red de forma rápida y sencilla (tardas más en registrarte si aún no tienes cuenta).

No obstante, debido al plazo de entrega y otras ocupaciones, aún tiene mejoras que pulir la app, como la vista en dispositivos móviles donde falla el fondo pantalla y el Componente TituloReceta.

También comentar que es un proyecto muy muy limitado y se basa en lo que pide el enunciado, con un pequeño añadido (compilación y enlace web), y queda abierto a mejoras en verano, es decir, añadir al menos, guardar datos, conexiones a bbdd o API, un registro de usuarios y reactividad con varias páginas implicadas.