

Proyecto Vue JS en Visual Studio Code

Índice

1.- Creando un proyecto ejemplo con npm.....	1
2.- Creando un proyecto ejemplo con Vite+Vue+JS.....	4
2.- Analizando la estructura del proyecto Vue+Vite+JS.....	5
2.1.- Entrada al proyecto: index.html.....	6
2.2.- El fichero main.js.....	6
2.3.- El fichero App.vue.....	7
2.3.1.- Template de App.vue.....	7
2.3.2.- Script de App.vue.....	7
2.3.3.- Style de App.vue.....	7
2.4.- El componente HelloWorld.vue.....	8
2.4.1.- Script del componente HelloWorld.vue.....	8
2.4.2.- Template del componente HelloWorld.vue.....	9

1.- Creando un proyecto ejemplo con npm

Podemos iniciarlo usando npm o el CLI de Vue:

- npm → npm init [vue@latest](#)
 - Se nos pide el nombre del proyecto, será miPrimerProyecto
- CLI de Vue → vue create miPrimerProyecto

El proceso continúa..

En la terminal nos pide qué tipo de instalación vamos a hacer. En nuestro caso elegiremos manual:

```
? Please pick a preset:
  Default ([Vue 2] babel, eslint)
  Default (Vue 3 Preview) ([Vue 3] babel, eslint)
> Manually select features
```

Después se abre un menú para elegir los componentes. Marcaremos las opciones acordes al proyecto:

```
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to
invert selection, and <enter> to proceed)
>(*) Babel
  ( ) TypeScript
  ( ) Progressive Web App (PWA) Support
  ( ) Router
  ( ) Vuex
  ( ) CSS Pre-processors
  ( ) Linter / Formatter
  ( ) Unit Testing
  ( ) E2E Testing
```

Ahora nos pregunta qué versión queremos de Vue:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
? Check the features needed for your project: Babel
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 3.x
  2.x
```

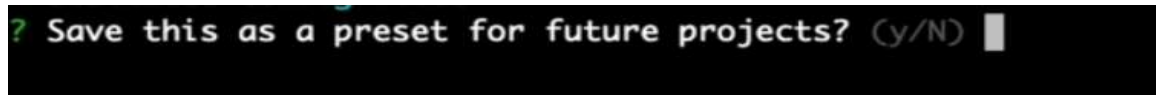
Elegimos la 3.

Ahora nos pregunta cómo queremos los ficheros de configuración. Elegimos la de ficheros dedicados:



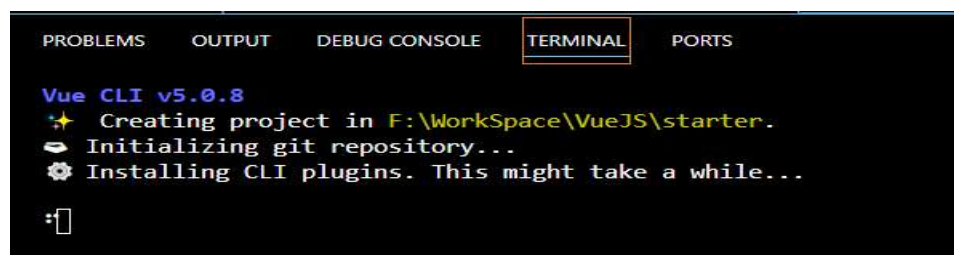
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
? Check the features needed for your project: Babel
? Choose a version of Vue.js that you want to start the project with 3.x
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

Nos pregunta si queremos guardar la configuración para futuros proyectos. Le decimos que no, porque es de prueba.



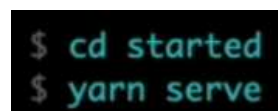
```
? Save this as a preset for future projects? (y/N)
```

Y comenzará la instalación:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Vue CLI v5.0.8
✨ Creating project in F:\Workspace\VueJS\starter.
📁 Initializing git repository...
⚙️ Installing CLI plugins. This might take a while...
📁
```

Finalmente nos indica cómo levantar la aplicación:



```
$ cd started
$ yarn serve
```

Una vez ejecutado la aplicación se levanta:

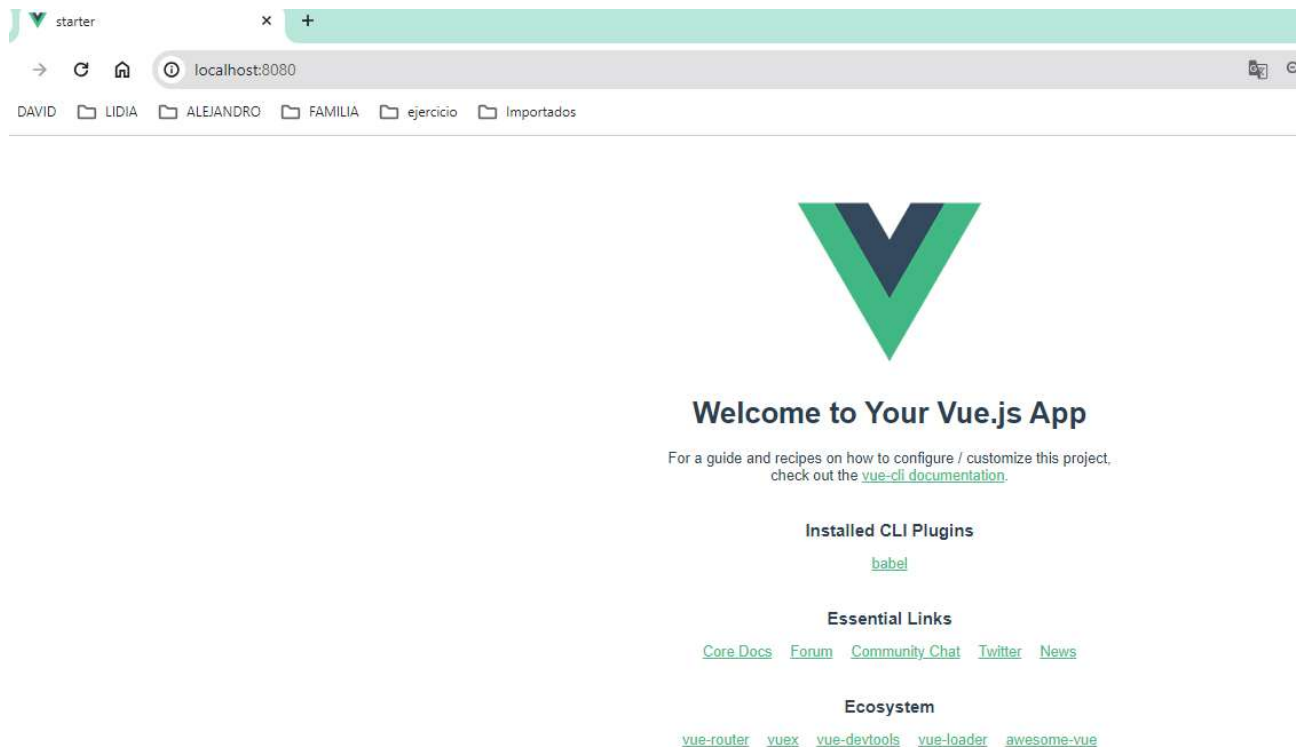
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

DONE Compiled successfully in 22896ms

App running at:
- Local:  http://localhost:8080/
- Network: unavailable

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Abrimos la URL indicada:



2.- Creando un proyecto ejemplo con Vite+Vue+JS

Nos colocamos en la carpeta donde desarrollar el proyecto Vite con Vue y Typescript-
Ejecutamos

npm create vite@latest

```
PS D:\02 IES TELESFORO BRAVO\19 MATERIALES\VueJS\00 - APUNTES\2 - Pyto base Vue con Vite> npm create vite@latest
Need to install the following packages:
create-vite@5.5.5
Ok to proceed? (y) y

> npx
> create-vite

✓ Project name: ... pytoBase_Vite_Vue_JS
✓ Package name: ... pytoBase-vite-vue-js
✓ Select a framework: » Vue
✓ Select a variant: » TypeScript

Scaffolding project in D:\02 IES TELESFORO BRAVO\19 MATERIALES\VueJS\00 - APUNTES\2 - Pyto base Vue con Vite\pytoBase_Vite_Vue_JS...

Done. Now run:

cd pytoBase_Vite_Vue_JS
npm install
npm run dev
```

2.- Analizando la estructura del proyecto Vue+Vite+JS



- **node_modules:** estan todos los ficheros de dependencia del proyecto.
- **Carpeta public:** el fichero principal de la aplicación index.html y el icono favicon
- **Carpeta src:** donde se construye la aplicación
- **App.vue :** dentro de la carpeta src.
- **main.js:** dentro de src. Es el primer fichero que se ejecuta. Le indica a Vite cómo crear el proyecto Vue en función del archivo "App.vue"
- **Carpeta src/assets:** almacena todos los archivos multimedia, documentos, etc.
- **Carpeta components:** se guardan los componentes de la aplicación.
- Por fuera tenemos los archivos de configuración: .gitignore, babel.config.js, jsconfig.json, package.json. Vue.config.js, etc.

2.1.- Entrada al proyecto: index.html

En el body se carga el script que carga el JS:

```
9 | <body>
10 | | <div id="app"></div>
11 | | <script type="module" src="/src/main.js"></script>
12 | </body>
13 </html>
```

El fichero es el main.js dentro de la carpeta src.

2.2.- El fichero main.js

El fichero main.js se encuentra dentro de src.

En el importamos el proyecto Vue, las hojas de estilo y el componente App.vue.

```
src > js main.js
1 | import { createApp } from 'vue'
2 | import './style.css'
3 | import App from './App.vue'
4 |
5 | createApp(App).mount('#app')
```

Creamos la aplicación con createApp(App).

Poniendo a la escucha el elemento con id “app”.

2.3.- El fichero App.vue

El fichero App.vue se encuentra dentro de src.

Este fichero sigue la estructura SFC:

- <template>donde está el contenido HTML.
- <script>para nuestro código Vue.
- <style>donde escribimos el estilo CSS.

Veamos cada uno

2.3.1.- Template de App.vue

Con tiene el html que se “pintará”. En este caso:

En el código se observa los enlaces de dos imágenes y un componente:

<HelloWorld msg="Vite + Vue" />

Dicho componente se encuentra en la carpeta src/components y se importará en la sección <script> de App.vue.



Vite + Vue

count is 0

[Edit components/HelloWorld.vue](#) to test HMR

Check out [create-vue](#), the official Vue + Vite starter

Learn more about IDE Support for Vue in the [Vue Docs Scaling up Guide](#).

[Click on the Vite and Vue logos to learn more](#)

2.3.2.- Script de App.vue

En este caso se importa el componente HelloWorld de src/components.

```
<script setup>
  import HelloWorld from "../components/HelloWorld.vue";
</script>
```

2.3.3.- Style de App.vue

Se define los estilos dentro de la etiqueta <style scoped>.

```
<style scoped>
.logo {
  height: 6em;
  padding: 1.5em;
  will-change: filter;
  transition: filter 300ms;
}
.logo:hover {
```

La palabra "scoped" indica que los estilos definidos dentro de esa etiqueta solo afectarán al componente en el que se declaran. No se aplicarán globalmente a otros elementos de la aplicación, lo que evita problemas de colisión de estilos y facilita el mantenimiento de tu código.

Existe el fichero general de CSS dentro de la carpeta src que define los estilos genéricos.

2.4.- El componente HelloWorld.vue

El componente HelloWorld.vue se encuentra dentro de src/components.

Por convención el nombre del componente empieza por mayúscula y con estilo Camell.

Sigue la estructura SFC.

Veamos cada parte del componente:

2.4.1.- Script del componente HelloWorld.vue

<script setup>

```
<script setup>
  import { ref } from "vue";

  defineProps({
    msg: String,
  });

  const count = ref(0);
</script>
```

En Vue 3, la etiqueta <script setup> es una forma simplificada y optimizada de definir la lógica de un componente cuando se utiliza la Composition API. Se introdujo para hacer que el código sea más conciso y fácil de escribir. Es más eficiente en términos de rendimiento porque el código se preprocesa de manera más optimizada por Vue.

Otras mejoras:

- Código más breve y con menos repeticiones
- Capacidad para declarar props y eventos emitidos usando TypeScript puro

- Mejor rendimiento en tiempo de ejecución (la plantilla se compila en una función de renderizado en el mismo ámbito, sin un proxy intermedio)
- Mejor rendimiento de inferencia de tipos IDE (menos trabajo para que el servidor de lenguaje extraiga los tipos del código)

import {ref} from “vue”;

ref es una función que convierte un valor en una referencia reactiva. Cuando se utiliza ref, Vue detecta automáticamente los cambios en el valor y actualiza la interfaz de usuario en consecuencia.

Las variables reactivas creadas con ref se pueden usar para manejar datos de manera similar a las propiedades de data en la Options API, pero con una estructura más flexible (`const count = ref(0)`).

¿Por qué se usa ref? Permite que Vue haga un seguimiento de los cambios y actualice la interfaz en tiempo real. Y es útil para definir variables reactivas dentro de la lógica de la Composition API, proporcionando una estructura más organizada para componentes complejos.

DefineProps({ msg: String, });

Es un macro del compilador que solo se pueden usar dentro de `<script setup>`

[DefineProps](#) acepta el mismo valor que la opción props.

Es una utilidad para definir y manejar las **propiedades (props)** que un componente espera recibir de su componente padre. Permite declarar y tipar las propiedades que el componente puede recibir.

Hace que el manejo de props sea más sencillo y limpio, eliminando la necesidad de definir un objeto props de forma tradicional.

¿Cómo funciona? `defineProps` recibe un objeto que define los tipos de las propiedades esperadas, similar a cómo se haría en la Options API, pero de una manera más optimizada. Cuando se utiliza `defineProps`, puedes acceder a las propiedades directamente como variables dentro del componente.

2.4.2.- Template del componente HelloWorld.vue

```
<template>
  <h1>{{ msg }}</h1>

  <div class="card">
    <button type="button" @click="count++>count is {{ count }}</button>
    <p>
      Edit
      <code>components/HelloWorld.vue</code> to test HMR
    </p>
  </div>
```


`<h1>{{ msg }}</h1>`

En contramos en primer lugar la interpolación de la variable `msg`

Esta variable **`msg`** es una variable definida como prop dentro de `defineProps`. Por lo tanto viene del padre, que es la clase **`App.vue`**. Y cuyo texto de definió en la línea `<HelloWorld msg="Vite + Vue" />` como el texto “Vite + Vue”.

En otras palabras, al definir el componente HelloWorld le pasamos por prop la variable `msg` con el texto “Vite + Vue”. Y en el componente HelloWorld se recoge al definir `defineProps({ msg: String, })`; y poder usar la variable `msg` en el template mediante el uso de interpolación.

`<button type="button" @click="count++">count is {{ count }}</button>`

Se define un botón con el evento click de Vue.

`@click` es la abreviatura de `v-on:click`.

V-on es un escuchador de eventos, véase más en la [página oficial](#).

En el evento click incremenamos la variable `count`, definida en el script.

Y en el texto del mismo botón visualizamos la variable `count` por medio de interpolación.