

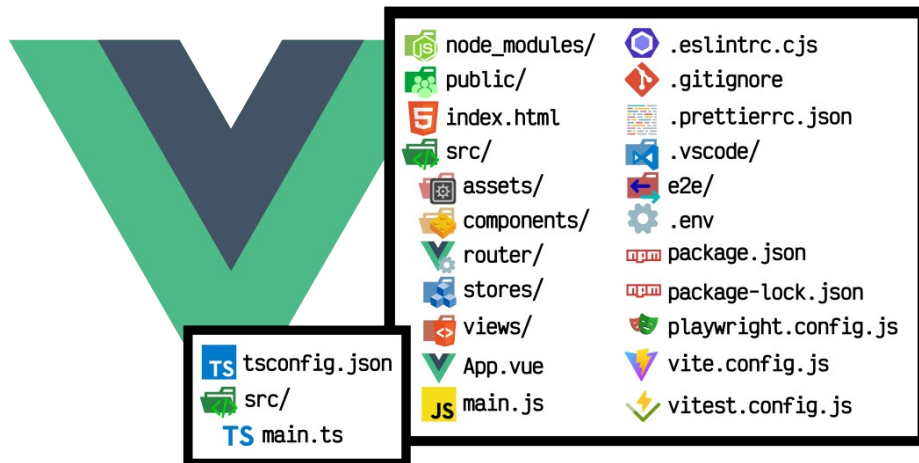
Practica VUE 03 - Estructura de Carpetas de VueJS

1 Ficheros de configuración de un proyecto Vue 3

Tanto si utilizamos el asistente de Vue por línea de comandos para crear un proyecto como si lo hacemos de forma manual, necesitaremos conocer la estructura de carpetas donde residen los diferentes archivos de nuestro proyecto.

Así, a la hora de hacer cambios, sabremos si lo estamos haciendo el sitio adecuado (*o que otras posibilidades tenemos*), de forma que el proyecto sigue siendo **fácil de mantener** por otros miembros del equipo de desarrollo o por alguien externo que quiera trabajar en el proyecto.

ESTRUCTURA DE CARPETAS



2 La carpeta raíz

En primer lugar, vamos a centrarnos en la **estructura de carpetas** de la **carpeta raíz**, profundizando en cada uno de sus ficheros, así como con que herramientas están relacionados.

El primer nivel de carpetas (*carpeta raíz del proyecto*) suele tener una pinta similar a la siguiente (*puede variar dependiendo de la elección del asistente al crear el proyecto*):

```

├── node_modules/
├── public/
├── index.html
├── src/
├── .eslintrc.js
├── .gitignore
├── .prettierrc.json
├── .vscode/
├── e2e/
├── .env
├── package.json
├── package-lock.json
├── playwright.config.js
├── vitest.config.js
└── vite.config.js

```

Las tres primeras carpetas, public, src y node_modules las abordaremos más adelante. Primero vamos a centrarnos en los ficheros, que suelen ser ficheros de configuración. Ten en cuenta que algunos archivos (*los que empiezan por punto*) son ficheros ocultos:

Fichero o carpeta	Descripción
index.html	Fichero HTML principal de la página o aplicación.
.eslintrc.cjs	Fichero de configuración del linter de Javascript ESLint .
.gitignore	Fichero que indica los archivos que Git debe ignorar al hacer el versionado.
.prettierrc.json	Fichero de configuración del formateador de código Prettier .
README.md	Fichero Markdown donde se documenta información sobre el proyecto.
.env	Fichero de variables de entorno del proyecto.
package.json	Fichero de configuración del proyecto, usando NPM . Más info en NPM.
package-lock.json	Fichero histórico de versionado de apoyo para el package.json. Más info en NPM.
playwright.config.js	Fichero de configuración de Playwright , framework de tests end-to-end.
tsconfig.json	Fichero de configuración de TypeScript , un metalenguaje de Javascript con tipos.
vitest.config.js	Fichero de configuración de Vitest , framework de tests unitarios para usar con Vite.
vite.config.js	Fichero de configuración de Vite , automatizador de aplicaciones web Javascript. Hot Module Replacement



Veamos cada uno de ellos para saber sus particularidades.

El fichero `.eslintrc.cjs`

El linter de Javascript ESLint es una magnífica herramienta que nos avisa en tiempo real (*mientras escribimos en el editor*) de, no solo posibles **errores** en nuestro código, sino también de **problemas de estilo** que pueden provocar que, a la larga, nuestro código sea menos legible o mantenible.

Con toda probabilidad tendremos un fichero con el nombre `.eslintrc.js` o `.eslintrc.cjs` (*Javascript*) o `.eslintrc.json` / `.eslintrc` (*JSON*). En dicho fichero tendremos la configuración de nuestro linter, que, si hemos utilizado el asistente create-vue, habrá configurado con reglas especiales para que nos aconseje no sólo sobre Javascript, sino como mejorar el código de nuestros componentes `.vue`.

Si quieres más información sobre **ESLint** y su configuración, puedes echar un ojo al artículo ESLint.

El fichero `.env`

EL archivo `.env` es un fichero de configuración del sistema que guarda una serie de **variables de entorno** relacionadas con el proyecto. Se suelen utilizar para establecer **variables importantes** (*nombres de usuario, contraseñas, etc...*) que se utilizarán en varias partes del proyecto y así sea mucho más sencillo modificarlas y mantenerlas.

Es especialmente útil cuando trabajamos con sistemas mixtos con **frontend y backend**, necesitamos utilizar credenciales para alguna API o cuando utilizamos contenedores de Docker, por ejemplo.

El fichero `.gitignore`

El fichero `.gitignore` es un archivo donde podemos escribir, línea por línea, los nombres de archivos (*o patrones o comodines*) que queremos que Git, nuestro sistema de control de versiones, ignore y no tenga en cuenta a la hora de guardar cambios en el repositorio.

Se trata de un simple fichero de texto que es **obligatorio** utilizar en nuestros proyectos, como mínimo para ignorar la carpeta `node_modules`. Si no sabes que ficheros y carpetas puede ser interesante añadir, puedes echar un vistazo a gitignore.io, donde te pueden aconsejar un `.gitignore` indicando el editor o sistema operativo que uses.

El fichero `.prettierrc`

El archivo `.prettierrc` es un fichero de configuración que pertenece a Prettier, un formateador de código Javascript automático, que a pesar de ser bastante **opcional**, permite modificar ciertos parámetros de comportamiento a través de este fichero y de algunas opciones.





La carpeta .vscode/

La carpeta .vscode es una carpeta que puede existir en algunos proyectos. Se trata de configuración del editor **Visual Studio Code**, pero que sólo se aplica a este proyecto.

En su interior suele tener ficheros como settings.json, donde se guarda dicha configuración, y/o el fichero extensions.json, que guarda recomendaciones de extensiones útiles, como por ejemplo Volar.

El fichero playwright.config.js

El framework de testing Playwright permite realizar **tests end-to-end (e2e)**, un tipo de tests que se realizan en un entorno de navegador, pudiendo comprobar aspectos más ligados al navegador del usuario. El fichero playwright.config.js permite indicar ciertos detalles de configuración, que en el caso de Vue, simplemente hace algunas configuraciones y referencia a la carpeta e2e/.

El fichero vitest.config.js

El framework Vitest es una herramienta para crear tests unitarios en un proyecto de forma agnóstica, y que pueden funcionar tanto en proyectos con Javascript vanilla, como con frameworks como React, Angular o Vue, que utilizan **Vite** como automatizador.

Está basada en ser compatible con la sintaxis de Jest, que es otro framework desarrollado por Facebook, ya que su API es bastante práctica y muy fácil de leer.

El fichero package.json

Los ficheros package.json y package-lock.json son los que utiliza NPM para crear y gestionar una aplicación o proyecto y sus dependencias. Si quieres más información sobre estos ficheros, echa un vistazo al artículo [El archivo package.json](#).

El fichero tsconfig.json

El fichero tsconfig.json tiene todos los detalles de configuración de TypeScript localizados en un mismo sitio. Vue lo utiliza para especificar los detalles de compilación que utilizará, así como la forma de actuar al procesar ficheros .ts o código Typescript incluido dentro de la etiqueta <script lang="ts"> que utilizaremos en los ficheros SFC .vue.

En el caso de utilizar **Typescript** en nuestro proyecto, todos los archivos con extensión .js los cambiaremos por .ts. Los ficheros con extensión .d.ts se utilizan para proporcionar información a TypeScript de la API de una librería o archivo escrito en Javascript.

El fichero vite.config.js

El fichero vite.config.js permite incluir detalles de configuración adicionales para **Vite**, el automatizador Javascript que utilizamos en una aplicación de Vue. Podemos configurar detalles como por ejemplo: rutas, plugins, alias, construcción para el despliegue, etc...



La carpeta public/

Probablemente, observarás una carpeta denominada `public/` en la carpeta raíz del proyecto. Esta carpeta se utiliza para almacenar ficheros estáticos que no serán procesados por el framework:

```
├── public/
│   ├── img/
│   ├── favicon.ico
│   ├── index.html
│   └── robots.txt
```

Vite copiará directamente el contenido de esta carpeta para utilizar en la versión definitiva de la web, por lo que es un lugar ideal para almacenar ficheros estáticos como `robots.txt`, `favicon.ico`, tipografías, imágenes o en general, recursos estáticos que necesitamos que se copien directamente.

Ten en cuenta que estos archivos siempre serán públicos y cualquier usuario podrá acceder a ellos, ya que no pasa antes por el framework.

La carpeta src/

La carpeta `src` es probablemente una de las más importantes. En ella se almacena el código fuente (*source*) de nuestro proyecto, el que estaremos modificando desde nuestro editor de texto o IDE. Es muy importante tener presente que dentro de `src` siempre vamos a encontrar los **archivos originales sin procesar**:

```
├── src/
│   ├── assets/
│   ├── components/
│   ├── router/
│   ├── stores/
│   ├── views/
│   ├── App.vue
│   └── main.js
```

Por ejemplo, en el caso de estar utilizando **TypeScript** o **Sass**, en `src/` encontrarás ficheros `.ts` y `.scss` (los cuales no son capaces de ser leídos por un navegador). Mientras, en otra carpeta fuera de `src/`, denominada comunmente `dist/` o `build/` se suelen almacenar los archivos finales procesados por **Vite**, como `.js` o `.css`, por ejemplo.



El fichero main.js o main.ts

En Vue, encontraremos siempre un fichero `main.ts` (*TypeScript*) o `main.js` (*Javascript*). Se trata del fichero principal que arranca el proyecto Vue y que se cargará desde la plantilla `index.html` que se incluye en el raíz del proyecto.

Este archivo se encargará de cargar **Vue** y todos sus plugins asociados, como iremos viendo en posteriores artículos de esta página. Generalmente, `main.js` o `main.ts` cargan el framework y sus plugins y leen el fichero **SFC** `App.vue`, donde comienza a crearse la aplicación **Vue**.

El fichero App.vue

Los ficheros `.vue` son los denominados **SFC de Vue** (*Single File Components*), se trata de un archivo especial de **Vue**, muy similar a un `.html` que incluye 3 etiquetas HTML especiales: `<template>`, `<script>` y `<style>`.

A primera vista, parecen ficheros `.html`, por lo que la curva de aprendizaje es muy sencilla, sin embargo, hay varias diferencias que veremos más adelante en el capítulo de Single File Components. De hecho, el fichero `App.vue` es un fichero **SFC** especial, el componente inicial de la aplicación, desde donde se irán cargando los demás componentes.

La carpeta assets/

En el interior de la carpeta `src` normalmente podemos encontrar una carpeta `assets`. Dicha carpeta se utiliza para guardar archivos estáticos como imágenes, audio, tipografías, video, etc... La diferencia con la carpeta `assets` o `img` que existe en la carpeta `public` es que, generalmente, los archivos estáticos que tenemos en `public` son para permitir un acceso directo a la URL.

Por otro lado, los archivos que tenemos en la carpeta `src/assets` se utilizan en nuestro código de la aplicación, importándolos, y muchas veces no queremos que se pueda acceder directamente a ellos mediante una URL concreta. Al utilizar imágenes de `src/assets`, la imagen suele ser procesada por la herramienta que utilice el framework para procesar el Javascript (*Vite en este caso*) y se suele renombrar.

La carpeta router/

Dentro de la carpeta `src` nos encontraremos una carpeta `router` si hemos elegido utilizar Vue Router en nuestra aplicación para crear rutas desde el frontend. En su interior tendremos un archivo `index.js` o `index.ts` donde podremos gestionar las rutas de la aplicación y los componentes que cargaremos.



La carpeta stores/

En el interior de src es posible encontrar una carpeta stores. Dicha carpeta existe si hemos seleccionado la opción de utilizar Pinia, un gestor o **almacén de estados** para Vue. Las aplicaciones web de tipo SPA suelen utilizar un **store** o almacén donde se guarda información centralizada para que cualquier componente o parte de la aplicación pueda acceder a ella, almacenar o recuperar información y gestionarla de forma segura.

Antiguamente, se utilizaba Vuex, sin embargo, desde **Vue 3**, la opción preferida para gestores de estado es utilizar **Pinia**, ya que tiene un enfoque mucho más simple y una filosofía más parecida a **Vue 3**. Más adelante profundizaremos en **Pinia**.

La carpeta components/

También en el interior de la carpeta src podremos encontrar components, probablemente una de las carpetas más importantes de nuestro proyecto Vue. En ella colocaremos los componentes .vue que iremos creando durante nuestro proyecto. Los componentes .vue son archivos que contienen el **HTML**, **CSS** y **Javascript** que está relacionado con una determinada parte de la página, como podría ser un botón, un panel desplegable o un comparador de imágenes.

En el caso de haber activado **Vue Router**, existirá una carpeta views donde también se guardan componentes. La diferencia respecto a la carpeta components, es que en views se guardan componentes que definen la estructura general de una página, mientras que en components se guardan partes reutilizables que podemos utilizar en múltiples lugares de nuestra web.

Más adelante veremos más sobre los **componentes de Vue**.

Es posible que nos encontremos algunos otros ficheros no mencionados en otros proyectos:

El fichero .browserslistrc

El archivo .browserslistrc es un fichero en el que podemos establecer una serie de reglas para indicar un conjunto de navegadores. Estas reglas pueden ser leídas por ciertas herramientas (*deben soportar **Browserslist**, como por ejemplo: Babel, PostCSS, etc...*), y harán lo posible para ser compatibles con los navegadores indicados.

En algunos casos, en lugar de utilizar el fichero .browserslistrc, la información se incluye en un de Javascript, dentro del campo browserslist del package.json. Tienes más información sobre como funciona en el artículo BrowsersList.

El fichero .editorconfig

En algunos casos, la estructura de carpetas puede contener un archivo .editorconfig. Este archivo es un intento de indicar características genéricas del código, para que se representen igual independientemente del editor que utilicen, ya sea **VSCode**, **Atom**, **IntelliJ IDEA** **WebStorm**, etc...

Este fichero incorpora convenciones utilizadas como el tipo de indentado (*tabuladores o espacios*),



el tamaño del indentado, si hay que eliminar los espacios en blancos del final de las líneas, añadir una línea en blanco al final, etc...

Tienes más información en la página oficial de EditorConfig.

El fichero README.md

El fichero README.md es un fichero en formato **Markdown**, que es un tipo de formato de texto que permite convertirse a HTML. Mediante **Markdown**, con una serie de símbolos, podemos formatear un documento utilizando **negritas**, *cursivas*, encabezados, imágenes, etc... Se suele incluir en los repositorios de Git, con el nombre en mayúsculas y la extensión en minúsculas: README.md. Este fichero servirá como primero punto de información sobre el repositorio.

3 Practica VUE 01 - Hola Mundo

1. ¿Enumera todos archivos de la carpeta raíz en un proyecto Vue?
2. ¿Que hace el El fichero package.json
3. ¿Enumera los archivos más importantes de La carpeta public
4. ¿Enumera los archivos más importantes de La carpeta src 5
5. ¿Que hace el fichero main.js o main.ts? Muestra su contenido aqui
6. ¿Que hace el fichero App.vue? Muestra su contenido aqui
7. ¿Que guarda la carpeta assets?
8. ¿Que guarda La carpeta router?