sustantiva

CON•SENTIDO

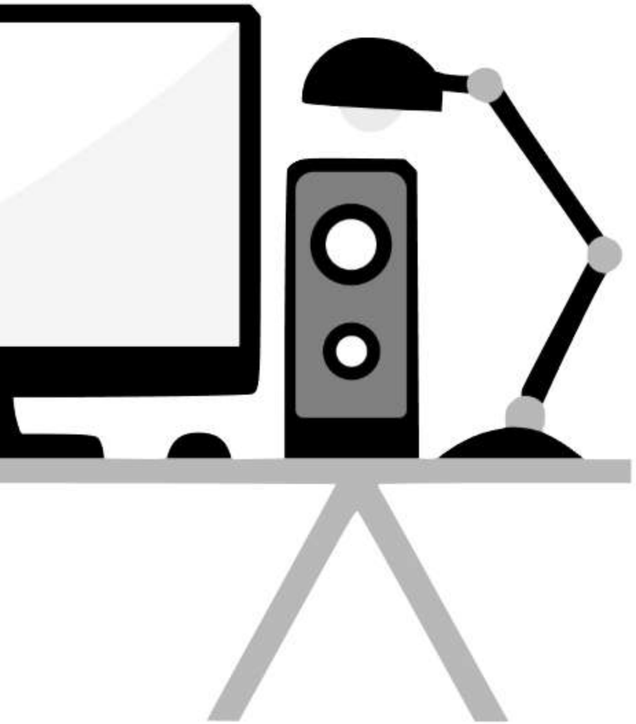
MANEJO DE RUTAS EN VUE

Lección 05

Competencia General

Implementar la navegación de una interfaz web utilizando Vue Router para dar solución a un requerimiento

Contenidos



1. Qué es Vue Router. Para qué y cuándo utilizarlo
2. Instalando Vue Router y usandolo como plugin Vue
3. Rutas estáticas y dinámicas
4. Rutas anidadas
5. Pasando props a componentes según ruta
6. Redirecciones y alias
7. Transiciones entre rutas

Qué es Vue Router

Vue Router es la librería oficial de enrutamiento para Vue.js. Se integra fácilmente con el core del framework y permite construir **Single Page Applications (SPA)**, donde los cambios de vista o página no requieren recargar el navegador.

Con Vue Router podemos asociar rutas específicas (URLs) a componentes del proyecto, permitiendo navegar entre ellos sin necesidad de recargar la página. Esto mejora el rendimiento y la experiencia del usuario. Por ejemplo, una aplicación puede tener las siguientes rutas:

- / → Página principal
- /contacto
- /orden/120
- /post/1

Vue Router

sustantiva
CON•SENTIDO

Para qué y cuándo utilizar Vue Router

Vue Router se utiliza cuando queremos desarrollar aplicaciones de una sola página (SPA) en Vue.js y necesitamos gestionar múltiples vistas, rutas o componentes de manera estructurada y eficiente.

Nos permite controlar cómo navega el usuario dentro de la aplicación, y vincular rutas específicas a componentes visuales sin perder el estado de la app ni hacer recargas innecesarias.

¿Cuándo usarlo?

- Cuando se requiere dividir la aplicación en múltiples vistas o secciones
- Si se desea que cada vista tenga su propia URL
- Para mejorar el rendimiento mediante **lazy loading** de componentes
- Para facilitar la navegación con enlaces amigables y claros

Vue Router permite:

- Control total sobre la navegación dentro de la app
- Visualizar componentes según la ruta actual Generar enlaces legibles y accesibles
- Aplicar redirecciones, alias y rutas dinámicas o anidadas
- Cargar componentes bajo demanda (lazy loading)

sustantiva
CON•SENTIDO

Instalando Vue Router y usandolo como plugin Vue

Competencias

- Instalar Vue Router mediante descarga directa (CDN).
- Instalar Vue Router utilizando NPM.
- Integrar Vue Router como plugin dentro de un proyecto Vue utilizando Vue CLI

Introducción

Luego de una breve introducción teórica a Vue Router y sus funcionalidades, es momento de pasar a la práctica. Prepararemos nuestro entorno para comenzar a trabajar con rutas dentro de una aplicación Vue moderna, entendiendo cómo integrarlas en una **SPA (Single Page Application)**.

Aplicaremos los conocimientos adquiridos en módulos anteriores y crearemos un proyecto base utilizando **Vue CLI** con los presets por defecto (Babel y ESLint). A este proyecto lo llamaremos **misitiospa**, el cual servirá como entorno para nuestros ejemplos prácticos.

Instalación de Vue Router

Vue Router puede instalarse de dos maneras, dependiendo del tipo de proyecto:

1. Descarga directa / CDN

Se puede incluir Vue Router en un archivo HTML de forma directa con etiquetas `<script>`:

```
<script src="/ruta/a/vue.js"></script>

<script src="/ruta/a/vue-router.js"></script>
```

Esto es útil para pruebas rápidas o proyectos sin configuración avanzada.

Instalación de Vue Router

2. Instalación por NPM

En caso de haber creado un proyecto con Vue CLI sin seleccionar la opción "Router", o si estás trabajando en un proyecto ya existente, es posible instalar Vue Router manualmente utilizando NPM.

Para ello, ejecuta el siguiente comando en la terminal, dentro de la carpeta del proyecto:

```
npm install vue-router
```

Una vez instalado Vue Router, verifica su presencia en el archivo package.json, dentro del bloque "dependencies". Deberías ver una línea similar a esta:

```
"vue-router": "^4.x.x"
```

Instalación de Vue Router

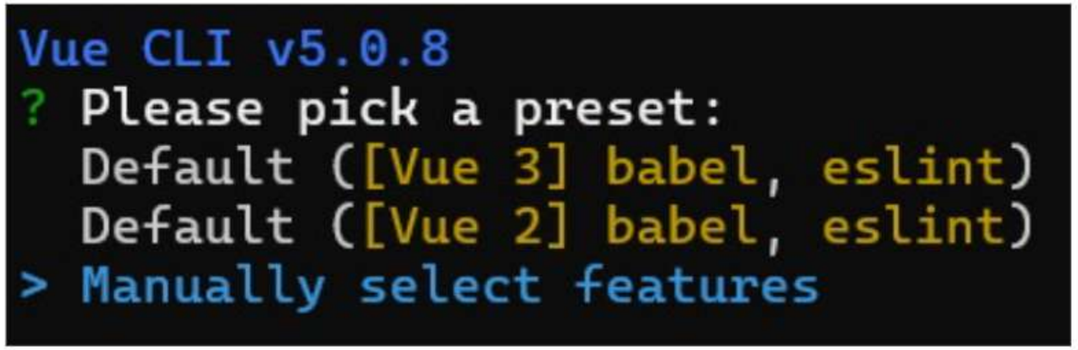
Para comenzar con un entorno limpio, vamos a crear un nuevo proyecto Vue utilizando **Vue CLI**. Esta herramienta nos permite generar la estructura base de una SPA (Single Page Application) e incluir desde el principio librerías como Vue Router o herramientas de linting.

Desde la terminal, ejecutamos el siguiente comando:

vue create misitiospa

Al ejecutar vue create, Vue CLI nos solicitará elegir una configuración o "preset". Tenemos dos opciones principales:

- **Default (Vue 3) [babel, eslint]**
- **Manually select features** ← *Seleccionamos esta opción* para tener control total sobre lo que queremos incluir, como el **Vue Router**.



Instalación de Vue Router

Aquí debemos indicar qué funcionalidades queremos incluir en nuestro proyecto. Las opciones marcadas deben ser:

- Babel
- Router
- Linter / Formatter

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? 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
```

Instalación de Vue Router

El asistente nos pedirá configurar otras opciones:

1. **Versión de Vue:** Seleccionamos Vue 3.x.
2. **History Mode para Router:** Aceptamos con Yes. Esto elimina el # de la URL.
3. **Linting Config:** Podemos dejarlo en Basic.
4. **Lint on save:** También dejamos activada esta opción.
5. **Dónde guardar la configuración:** Elegimos In package.json.

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a linter / formatter config: Basic
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.?
  In dedicated config files
> In package.json
```

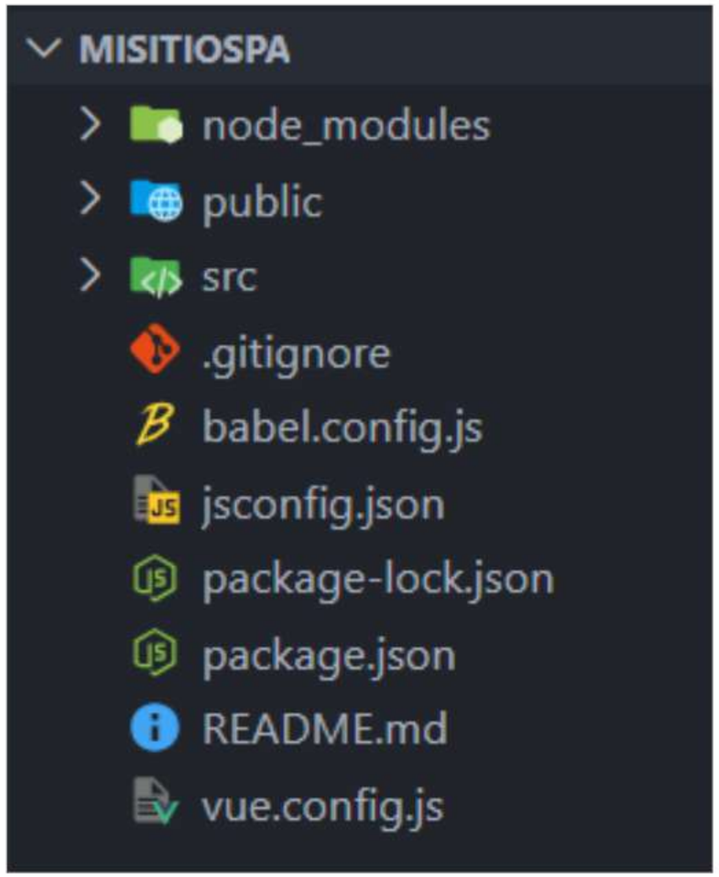
Instalación de Vue Router

Una vez completado el asistente, Vue CLI creará una carpeta con todos los archivos necesarios para comenzar a trabajar en el proyecto misitiospa.

Podemos entrar al directorio y ejecutar el servidor de desarrollo con:

```
cd misitiospa
npm run serve
```

Esto iniciará el proyecto y podremos ver la aplicación funcionando en http://localhost:8080.



Instalación de Vue Router

Cuando creamos un proyecto con Vue CLI y seleccionamos la opción de incluir Vue Router, el archivo App.vue se genera automáticamente con una estructura básica que ejemplifica el uso inicial del sistema de enrutamiento.

En esta plantilla predefinida, encontramos el componente <router-link>, que permite la navegación entre distintas rutas de forma dinámica, sin necesidad de recargar la página. Este componente reemplaza el uso tradicional de etiquetas <a> al trabajar en aplicaciones de una sola página (SPA).

También se incluye el componente <router-view>, que actúa como un marcador de posición: allí se renderizará el contenido del componente correspondiente a la ruta activa.

Finalmente, se incorporan estilos CSS básicos que definen el diseño general de la aplicación y personalizan los enlaces de navegación para mejorar la experiencia visual del usuario.

Instalación de Vue Router

Home | About

Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the vue-cli documentation.

Installed CLI Plugins

babel router eslint

Essential Links

Core Docs Forum Community Chat Twitter News

Ecosystem

vue-router vuex vue-devtools vue-loader awesome-vue

```vue
<template>
  <nav>
    <router-link to="/">Home</router-link> |
    <router-link to="/about">About</router-link>
  </nav>
  <router-view/>
</template>
<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}
nav {
  padding: 30px;
}
nav a {
  font-weight: bold;
  color: #2c3e50;
}
nav a.router-link-exact-active {
  color: #42b983;
}
</style>
```

Implementación de Vue Router cómo Plugin

Una vez creado nuestro proyecto con Vue CLI, el siguiente paso es implementar Vue Router como un plugin dentro de nuestra aplicación. Esto nos permitirá crear una SPA (Single Page Application) con navegación entre vistas sin recargar la página.

Paso 1: Modificación de App.vue

El archivo App.vue será nuestra vista principal o "máscara" de la aplicación. Aquí añadiremos enlaces de navegación mediante el componente <router-link>, que genera enlaces a las distintas rutas definidas en Vue. También incorporamos el componente <router-view>, que sirve como punto de inserción para renderizar las vistas asociadas a cada ruta.

```
<template>
  <div id="app">
    <nav>
      <router-link to="/">Inicio</router-link> |
      <router-link to="/contacto">Contacto</router-link>
    </nav>
    <router-view />
  </div>
</template>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

Implementación de Vue Router cómo Plugin

Paso 2: Crear Componentes de Vistas

Ahora agregaremos las vistas que se mostrarán cuando naveguemos por las rutas. Para ello, creamos dos archivos dentro de la carpeta src/views: VistaInicio.vue y VistaContacto.vue.

VistaInicio.vue

```
<template>
  <h1>Página de inicio</h1>
</template>
```

VistaContacto.vue

```
<template>
  <h1>Página de Contacto</h1>
</template>
```

Con esto, ya tenemos configurada la navegación básica entre dos vistas en nuestra aplicación Vue. El siguiente paso será definir las rutas dentro del archivo router/index.js, que exploraremos a continuación.

Implementación de Vue Router cómo Plugin

Paso 3: Configuración del router en el proyecto

Cuando creamos un proyecto con Vue CLI incluyendo Vue Router, se genera automáticamente una carpeta src/router/ con un archivo index.js.

No es necesario crear uno nuevo. Solo debemos modificar el contenido del archivo index.js para registrar nuestras rutas personalizadas.

```
import { createRouter, createWebHistory } from 'vue-router'
import Inicio from "../views/VistaInicio.vue"
import Contacto from '../views/VistaContacto.vue'


const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes: [
  {
    path: '/',
    name: "inicio",
    component: Inicio
  },
  {
    path: '/contacto',
    name: 'contacto',
    component: Contacto
  }
]
})

export default router
```

Implementación de Vue Router cómo Plugin

Paso 4: Integración del router en el archivo principal

Para que Vue Router funcione correctamente en nuestra aplicación, es necesario integrarlo en el archivo principal main.js. Este archivo es el punto de entrada de nuestra aplicación y es donde se crea la instancia de Vue. En este paso, importamos tanto el componente principal App.vue como el archivo de rutas que configuramos previamente (router/index.js). Luego, registramos Vue Router como un plugin mediante .use(router) antes de montar la aplicación con .mount('#app'). Esta configuración permite que las rutas definidas estén disponibles en toda la aplicación y puedan ser utilizadas mediante los componentes <router-link> y <router-view> en cualquier parte de la interfaz.

```
import { createApp } from 'vue'
import App from './views/App.vue'
import router from './router'

createApp(App).use(router).mount('#app')
```

Implementación de Vue Router cómo Plugin

Paso 5: Levantamiento y visualización del sitio

Inicio | Contacto

Página de inicio

Una vez configurado todo, ejecutamos el comando npm run serve desde la terminal para levantar nuestro sitio localmente. Vue CLI iniciará un servidor de desarrollo y nos mostrará una URL —generalmente http://localhost:8080— para acceder a la aplicación. Al ingresar en dicha dirección desde el navegador, podremos ver nuestra interfaz en funcionamiento, incluyendo el menú de navegación con las rutas que definimos, como "Inicio" y "Contacto"

Inicio | Contacto

Página de Contacto

Rutas estáticas y dinámicas

Rutas Estáticas

Las rutas estáticas son aquellas que tienen una dirección fija y se utilizan para mostrar componentes específicos sin parámetros adicionales. Son ideales para páginas como Inicio, Contacto, Nosotros, etc.

Se definen como objetos dentro del array routes, especificando al menos los campos path (la URL) y component (el componente que se debe renderizar).

Componente	Ruta definida	Acceso
`VistaInicio.vue`	/	http://localhost:8080
`VistaContacto.vue`	/contacto	http://localhost:8080/contacto

```
const routes = [
  {
    path: '/',
    name: "inicio",
    component: Inicio
  },
  {
    path: '/contacto',
    name: 'contacto',
    component: Contacto
  }
]
```

Rutas estáticas y dinámicas.

Rutas Dinámicas

Las rutas dinámicas nos permiten capturar valores variables directamente desde la URL, lo que resulta útil cuando deseamos renderizar información específica según el contexto del usuario o del contenido. Estas rutas se definen utilizando el prefijo : seguido del nombre del parámetro.

Por ejemplo, si queremos mostrar un listado de publicaciones en /posts, usaríamos una ruta estática. Pero si deseamos acceder al detalle de un post específico, como /post/1, necesitamos una ruta dinámica.

Dentro del componente Post.vue, podemos acceder al valor del parámetro dinámico id usando:

this.$route.params.id

Esto nos permite realizar acciones como cargar el post correspondiente desde un servicio o base de datos según su identificador.

```
const routes = [
  { path: '/posts', component: ListPosts },
  { path: '/post/:id', component: Post }
]
```

Rutas anidadas

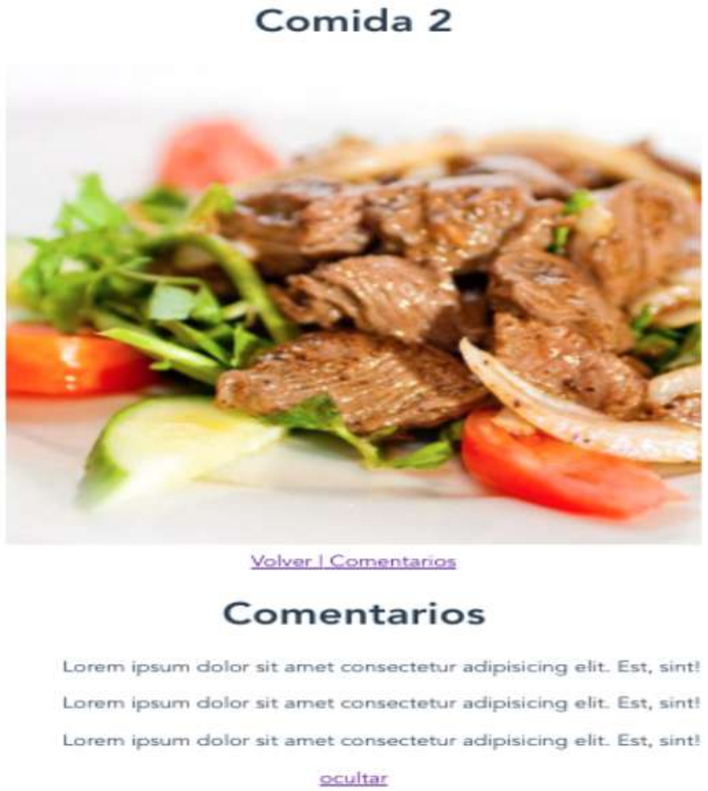
Rutas Anidadas

En algunos casos podríamos necesitar **rutas anidadas**, para hacer referencia a un elemento complementario de un componente principal. Por ejemplo, si tenemos una ruta llamada food, la cual es dinámica (/food/:id), podríamos anidar una sub ruta como comments para mostrar los comentarios asociados a esa comida. Esto permite estructurar mejor nuestras vistas y mantener la navegación limpia y jerárquica.

```
{ path: '/food/:id',
  component: Food,
  children: [
    {
      path: 'comments',
      component: FoodComments,
    }
  ]
}
```

```
  <div>
    <h1>Comida {{ id }}</h1>
    <img :src="src" alt="img"><br>

    <router-link to="/food">Volver</router-link>
    <router-link :to="`/food/${id}/comments`"
     >Comentarios</router-link>
    <router-view></router-view>
  </div>
}
```



Pasando props a componentes según ruta

Pasando props a componentes según ruta

Cuando usamos $route directamente en un componente Vue para acceder a parámetros dinámicos (como un ID desde la URL), generamos un alto acoplamiento entre la lógica del componente y la ruta.

Esto limita su reutilización, ya que solo podrá funcionar correctamente en ciertas URLs.

Para evitar este acoplamiento, Vue Router permite usar la opción props: true, lo que convierte los parámetros de la ruta en propiedades (props) del componente.

sustantiva
CON•SENTIDO

Pasando props a componentes según ruta

1. Antes – Uso acoplado con $route

```
methods: {
  getId() {
    return
this.$route.params.id;
  }
}
```

El componente depende directamente del enrutador. Si se usa fuera de una ruta con parámetro id, fallará.

2. Configurando el router con props

```
{
  path: '/food/:id',
  component: Food,
  props: true
}
```

Con props: true, indicamos que los parámetros de la URL deben convertirse en props del componente.

3. Recibiendo el parámetro en el componente

```
export default {
  name: 'FoodComponent',
  props: ['id'],
  data() {
    return {
      src: `url-api/comida/${this.id}`
    }
  }
}
```

Ahora id llega como una prop. El componente ya no necesita saber nada del router y puede reutilizarse fácilmente.

Redirecciones y alias

Una redirección ocurre cuando el usuario intenta acceder a una ruta específica (por ejemplo, /food) y el sistema automáticamente lo lleva a otra ruta diferente (como /about). Esto es útil para mantener URLs limpias, evitar duplicación de contenido o redirigir tráfico a rutas actualizadas.

```
{
  path: '/food',
  redirect: '/about'
}
```

```
{
  path: '/about',
  name: 'About',
  component: About
}
```

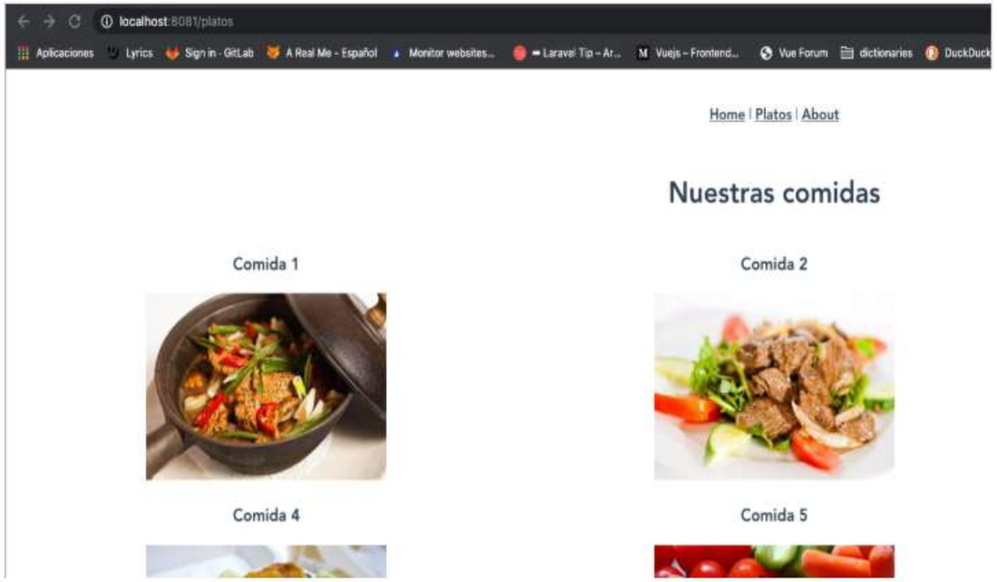
Cuando el usuario ingresa a /food, será redirigido automáticamente a /about.

Esto permite mantener una sola definición del componente y evitar repetir lógica innecesaria.

Redirecciones y alias

A diferencia de las redirecciones, los **alias** permiten que múltiples rutas apunten al mismo componente **sin cambiar la URL en el navegador**. Es decir, funcionan como sinónimos de una misma vista.

```
{
  path: '/food',
  component: Foods,
  alias: ['/comidas', '/platos']
}
```



Ya sea que el usuario acceda a /food, /comidas o /platos, siempre verá el mismo contenido, manteniendo la URL ingresada.

Transiciones entre rutas

Vue nos ofrece una forma sencilla de aplicar **efectos de transición** cuando nuestra aplicación cambia de una vista a otra. Esto mejora la experiencia del usuario al darle fluidez a la navegación.

Para lograrlo, envolvemos el componente <router-view /> con la etiqueta <transition> y definimos las clases CSS asociadas a la animación.

```
<transition name="router-anim"
mode="out-in">
  <router-view />
</transition>
```

```
.router-anim-enter-active {
  animation: coming 1s;
  animation-delay: .5s;
  opacity: 0;
}

.router-anim-leave-active {
  animation: going 1s;
}
```

sustantiva

CON•SENTIDO