



## Clase 10

# Diseño y Programación Web

**Materia:**  
Aplicaciones para  
Dispositivos Móviles

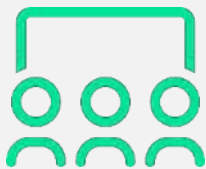
**Docente contenidista:** GARCIA, Mabel

**Revisión:** Coordinación

# Contenido

Instalación Vue-router .....	04
Estructura base del proyecto.....	06
HomeView .....	06
AboutView .....	08
App.....	08
Main.js .....	09
Configuración de rutas .....	10
Carpeta router/index.js .....	10
Paso por parámetros entre vistas (\$route.params) .....	13
Bibliografía .....	20

# Clase 10



iTe damos la bienvenida a la materia  
**Aplicaciones para Dispositivos Móviles!**

**En esta clase vamos a ver los siguientes temas:**

- Instalación de Vue-router.
- Estructura base del proyecto.
- Configuración de rutas.
- Paso por parámetros entre vistas (`$route.params`).

# Instalación Vue-router

Anteriormente, en el apunte 8, comenzamos a familiarizarnos con Vue-Router utilizando el CDN y desde la perspectiva de JavaScript. En esta sección, nos adentraremos en cómo trabajar con este sistema desde el entorno de consola para poder crear un proyecto del tipo SPA (Single Page Application) y profundizar en Vue-router.

Cuando creamos un proyecto desde las opciones de configuración por consola tenemos la posibilidad de instalarlo desde ahí.

Recordemos que depende de la ruta en donde estamos ubicados en la consola se generará el nuevo proyecto. Para esto necesitamos el comando:

**npm create vue@latest.**



A este nuevo proyecto le daremos el nombre de “**enrutamiento**” y seleccionamos por medio del cursor la opción **YES** para “agregar Vue Router para el desarrollo de aplicaciones de una sola página”, además configuramos con la opción YES que vamos a crear un proyecto con Eslint y Prettier.



Luego nos indicará que comandos tenemos que escribir para poder correr este nuevo proyecto:

```
Done. Now run:

cd enrutamiento
npm install
npm run format
npm run dev
```

Una vez que ejecutemos los comandos necesarios, nos dirá el puerto en el cual tendremos el servidor ejecutándose con nuestro proyecto en funcionamiento.

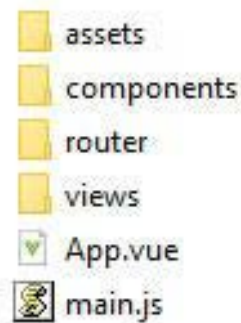
Por defecto: **localhost://5173**

Si abrimos el navegador y vamos a esta dirección, vemos que a la interfaz se le **ha agregado una navegación** de ejemplo con dos links: **Home** y **About**.



# Estructura base del proyecto

Esta nueva funcionalidad provoca una adición de carpetas y configuraciones que se añaden al proyecto base. Si exploramos la estructura de archivos y carpetas generados, observamos que se han creado **dos nuevos directorios: views y router dentro de la carpeta src**



Dentro de la carpeta **views** encontraremos las vistas creadas por defecto (HomeView y About). Las vistas son un concepto fundamental y representan los elementos visuales de nuestra aplicación.

Se implementan habitualmente usando templates HTML y componentes. Éstas son responsables de mostrar datos y responder a las interacciones del usuario.

Cada vista suele estar asociada a una ruta específica y puede contener componentes, datos y lógica relacionada con esa parte de la interfaz.

## HomeView

El archivo **HomeView**, por ejemplo, incorpora un componente dentro de su template desde la carpeta **components** en donde se encuentra definido. Recordemos que por defecto está usando sintaxis de la Api composition:

```
HomeView.vue

<script setup>
import TheWelcome from '../components/TheWelcome.vue'
</script>

<template>
  <main>
    <TheWelcome />
  </main>
</template>
```

Pasemos este código a sintaxis de la Api Options:

```
HomeView.vue

<script>
import TheWelcome from '../components/TheWelcome.vue'

export default {
  name: 'HomeView',
  components: {
    TheWelcome
  }
}
</script>

<template>
  <main>
    <TheWelcome />
  </main>
</template>
```

Repasemos la sintaxis:

Importa un componente Vue llamado **TheWelcome** desde un archivo **TheWelcome.vue** ubicado en la carpeta **components**.

Define un objeto de configuración para este componente:

- Establece el nombre del componente como 'HomeView'.
- Registra el componente **TheWelcome** importado como un componente hijo.

En el template:

- Renderiza una sección <main>.
- Dentro de <main>, renderiza una instancia del componente **TheWelcome**.

## AboutView

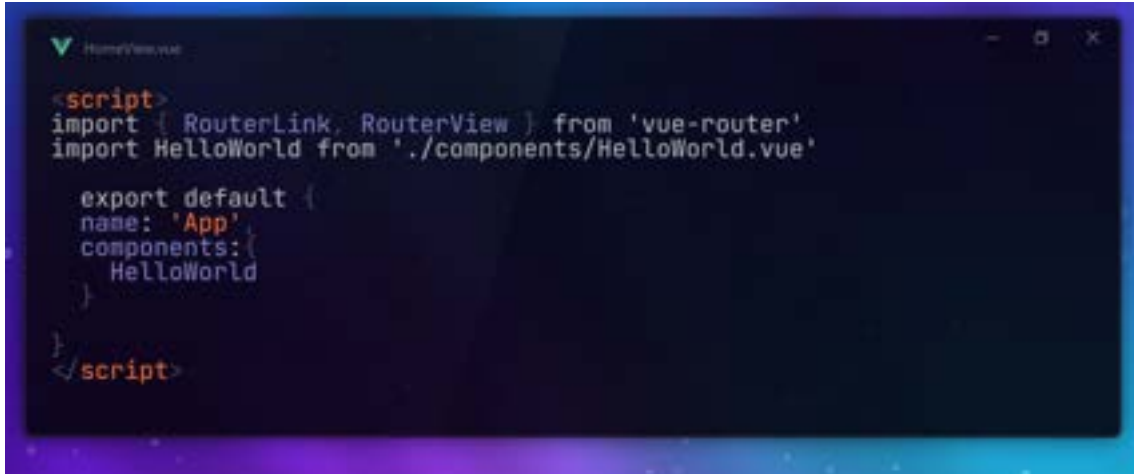
En cambio la vista **AboutView** no incorpora ningún componente. Deberíamos declarar la definición de este componente.



```
<script>
export default {
  name: 'AboutView'
}
</script>
```

## App

El archivo **App.vue** deberíamos modificarlo:



```
<script>
import { RouterLink, RouterView } from 'vue-router'
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',
  components: {
    HelloWorld
  }
}
</script>
```

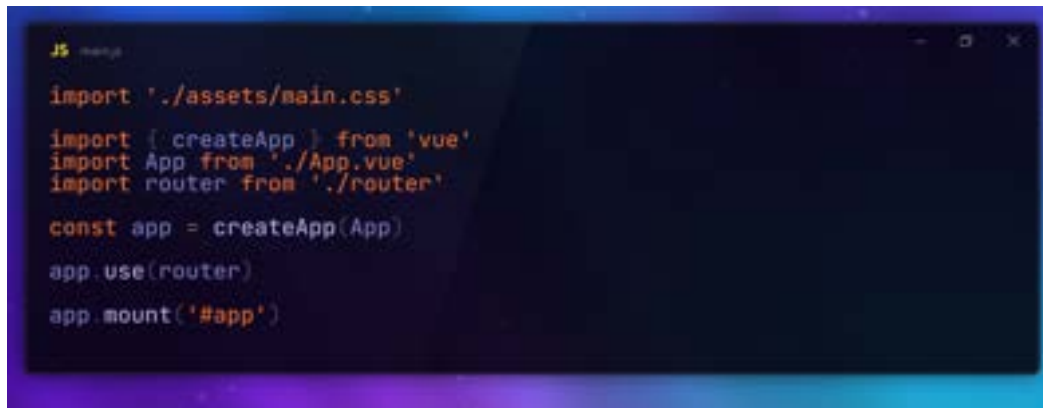
Los componentes **RouterLink** y **RouterView** se importan desde Vue Router y son globales.

Por otro lado, el componente **HelloWorld** debe ser registrado en el objeto **components**, porque es un componente personalizado que hemos creado y debe estar disponible para App.vue.



# Main.js

Nuestro archivo **main.js** también se ve modificado por la incorporación de vue-router:



```
import './assets/main.css'

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

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

Analicemos el código:

- Importa el archivo **main.css** ubicado en la **carpeta assets**. Éste contiene los estilos aplicados a toda la aplicación Vue.js.
- Importa la **función createApp** desde la carpeta **node\_modules**. Esta función permite **crear una nueva instancia**.
- Importa el **componente raíz desde el archivo App.vue**. Este componente es el **punto de entrada de la aplicación y contendrá los demás componentes y la lógica principal**.
- Se importa el **enrutador de Vue Router** desde la carpeta router. Este enrutador se encargará de manejar la navegación y las rutas de la aplicación.
- **Crea la instancia root con la función createApp y le pasa el componente raíz App como argumento.**
- **Se registra el enrutador** importado anteriormente en la instancia con **el método use**. Esto permite que use el enrutador para manejar la navegación y las rutas.
- **Monta la aplicación en un elemento HTML con el ID #app.** Todo el contenido y la lógica se renderizará dentro de este elemento HTML en el documento.

# Configuración de rutas

## Carpeta router/index.js

Por otro lado, la carpeta **router** al crear el proyecto genera un archivo llamado **index.js**, el cual define las rutas de nuestra aplicación y cómo se relacionan con las vistas que acabamos de mencionar.

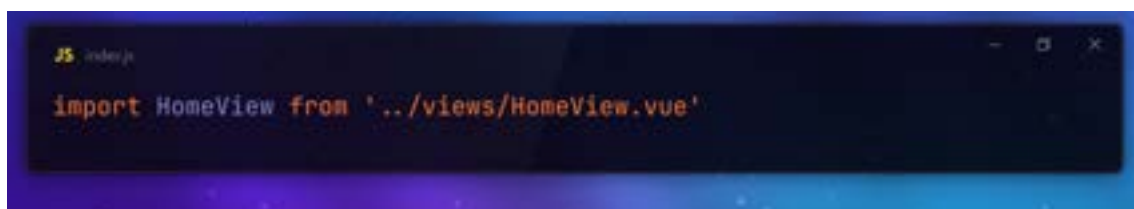
En este archivo, podemos establecer rutas dinámicas, rutas protegidas que requieran autenticación y realizar otras configuraciones avanzadas para el enrutamiento.

Analicemos el código por defecto del archivo index.js:



```
import { createRouter, createWebHistory } from 'vue-router'
```

- Importa dos funciones desde el paquete vue-router que se encuentra en la carpeta node\_modules. La función createRouter se utiliza para crear una nueva instancia del enrutador, mientras que createWebHistory se usa para crear un historial de navegación web.



```
import HomeView from '../views/HomeView.vue'
```

- Importa el componente Vue llamado HomeView desde el archivo HomeView.vue ubicado en la carpeta views. Este componente se utilizará como la vista principal (home) de la aplicación.



```

JS index.js

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    //definición de rutas
  ]
})

```

- Crea una nueva instancia del enrutador usando la función `createRouter`. Dentro del objeto de configuración, se establece el history utilizando `createWebHistory` (`import.meta.env.BASE_URL`). Esto configura el enrutador para utilizar el historial de navegación web y tener en cuenta la ruta base de la aplicación (útil cuando se despliega en un subdirectorio).



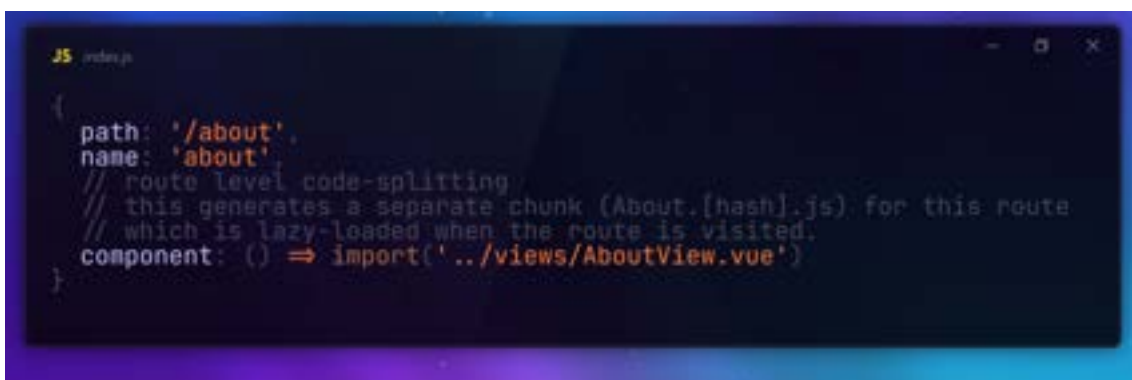
```

JS index.js

{
  path: '/',
  name: 'home',
  component: HomeView
}

```

- Dentro del array `routes` tenemos la primera definición de ruta. La propiedad `path` especifica la ruta URL (`/` representa la ruta raíz). La propiedad `name` asigna un nombre a la ruta (en este caso, `'home'`), lo cual puede ser útil para hacer referencia a ella más adelante. La propiedad `component` especifica el componente Vue que se renderizará cuando se visite esta ruta (en este caso, `HomeView`).



```

JS index.js

{
  path: '/about',
  name: 'about',
  // route level code-splitting
  // this generates a separate chunk (About.[hash].js) for this route
  // which is lazy-loaded when the route is visited.
  component: () => import('../views/AboutView.vue')
}

```

- Esta es otra definición de ruta. En este caso, la ruta es **/about**. La propiedad **component** utiliza una función de flecha que importa dinámicamente el componente **AboutView.vue** de la carpeta **views**. Esto permite que el código de este componente se divida (code-splitting) en un archivo JavaScript separado que se cargará de forma perezosa (lazy-loaded) cuando se visite esta ruta, lo que puede mejorar el rendimiento inicial de la aplicación.

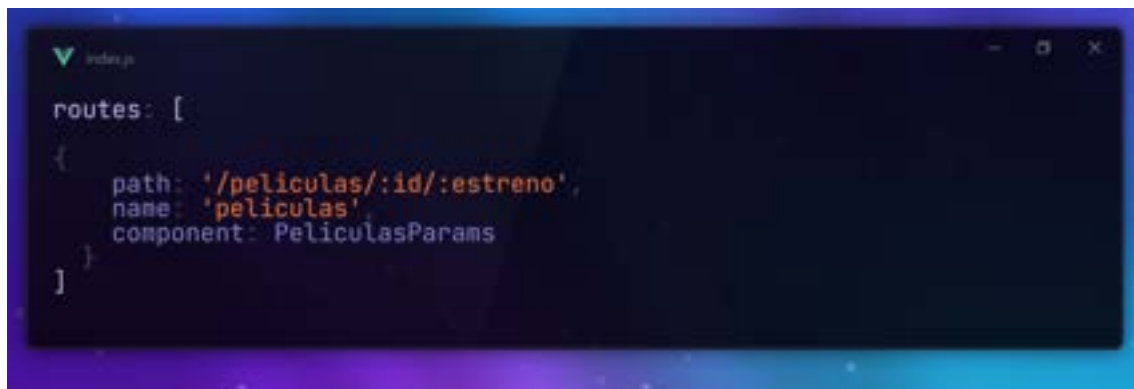


- Por último, exporta la instancia del enrutador como el módulo predeterminado. Esto permite que otros archivos importen y utilicen el enrutador en la aplicación Vue.js.

## Paso por parámetros entre vistas (\$route.params)

Vue.js, al igual que otros frameworks modernos, facilita el paso de parámetros entre diferentes vistas al definir rutas. Utilizando vue-router, podemos incorporar **segmentos dinámicos** en las rutas para lograr este objetivo. Un segmento dinámico se indica con dos puntos (:), y su valor se hace accesible como **this.\$route.params** dentro de cada componente cuando se coincide con una ruta específica.

Es importante destacar que las rutas que requieren parámetros deben estar correctamente declaradas en la configuración de rutas de la aplicación, dentro del archivo **index.js**.



```
index.js
routes: [
  {
    path: '/peliculas/:id/:estreno',
    name: 'peliculas',
    component: PeliculasParams
  }
]
```

En el código del ejemplo anterior, estamos pasando dos parámetros en la ruta: **películas**.

De esta forma listaremos, dinámicamente, un conjunto de películas y creamos enlaces que permiten navegar a vistas detalladas usando parámetros dinámicos en las rutas.

Consideremos el siguiente escenario como ejemplo: tenemos un componente que lista ciertos datos, mostrando únicamente un título al usuario. Junto a cada título, se incluye un componente **RouterLink** que permite al usuario navegar a una vista detallada.

Al hacer click en este enlace, se redirige al usuario a otra vista, pasando el id del título como parámetro en la URL. Como resultado, se evalúa el **id** recibido a través de la URL y se presenta al usuario el template correspondiente con los detalles asociados a ese **id**.

Ejemplifiquemos:

```
PelículasComponent.js

<template>
  <div class="detalles">
    <ol reversed>
      <li v-for="item in pelis" :key="item.id">
        {{ item.titulo }}

        <router-link :to="'/peliculas/' + item.id + '/' +
item.estreno"> Ver Detalle </router-link>
      </li>
    </ol>
  </div>
</template>

<script>
export default {
  name: 'PelículasComponent',
  data: function () {
    return {
      pelis: [
        { id: 1, titulo: 'The Avengers', estreno: 2012 },
        { id: 2, titulo: 'Los juegos del hambre', estreno: 2012 },
        { id: 3, titulo: 'Avatar', estreno: 2009 }
      ]
    }
  }
}
```

El código anterior, estamos definiendo un componente Vue llamado **PelículasComponent**. Dentro del template del componente, se encuentra una lista ordenada (<ol>), la cual se muestra en orden inverso gracias al atributo reversed. La lista se genera dinámicamente usando la directiva **v-for**, que itera sobre el array **pelis**, definido en la sección de datos del componente.

Por cada elemento en el array **pelis**, se crea una etiqueta <li>. Dentro de cada <li>, se muestra el título de la película por medio de una interpolación. Además, tenemos un componente **router-link**, que genera un enlace para cada película. Este enlace utiliza la ruta **/peliculas/** seguida del id y el estreno de la película como parámetros, permitiendo la navegación a una vista detallada específica para cada película al hacer click en "Ver Detalle".

En la **sección de script del componente**, se define un objeto que contiene el **nombre del componente** y una **función data** que **retorna** un **objeto**. Este objeto contiene el array **pelis**, el cual incluye varios objetos, cada uno representando una película con sus propiedades id, título y estreno.

En el navegador, veríamos el siguiente template del componente **PelículasComponent**:

III. The Avengers [Ver Detalle](#)

II. Los juegos del hambre [Ver Detalle](#)

I. Avatar [Ver Detalle](#)

Cuando el usuario haga click en el vínculo "Ver Detalle", pasaremos por parámetro su id y fecha de estreno.  
Por ejemplo, si el usuario elige **Ver detalles** de **The Avengers**, iremos a la **ruta previamente configurada en el archivo index.js** que coincide con la url pasada por medio del componente **RouterLink**:

<http://localhost:5173/peliculas/1/2012>

Esto nos llevará a una vista creada por nosotros que recibirá estos parámetros y evaluará qué contenido debe mostrar acorde al id que recibe. Estos valores los captura con **this.\$route.params**. En este caso **podemos asignar estos parámetros a variables para el componente o trabajarlos directamente con esa sintaxis por medio del enlace de datos**.

Los parámetros: **1- 2012**

### The Avengers



Cuando un enemigo inesperado amenaza la seguridad del planeta y de sus habitantes, Nick Fury (Samuel L. Jackson), director de SHIELD, monta un dispositivo con todos los hombres capaces de preservar a la humanidad del caos. El enemigo es tan poderoso, que necesita que todos los superhéroes luchen juntos y formen un equipo compacto. Ellos serán: el Capitán América (Chris Evans), Thor (Chris Hemsworth), Iron Man (Robert Downey Jr.), Hulk (Mark Ruffalo), Ojo de Halcón (Jeremy Renner) y la Viuda Negra (Scarlett Johansson).

Basado en los cómics de Marvel, el film está dirigido por Joss Whedon (Buffy Cazavampiros, Angel), que ya practicó el género en su primera película,

Serenity. Completan el reparto Gwyneth Paltrow (que repite su papel en Iron Man), Paul Bettany (que pone la voz a Jarvis), Cobie Smulders (Cómo conocí a vuestra madre), como Maria Hill, Stellan Skarsgård (Rompiendo las olas), como el profesor Erik Selvig y Tom Hiddleston en el papel que interpretó en Thor.

Fecha de Estreno: **2012**



```

V PeliculasParamsView.js
<template>
  <div v-if="i === '1'">
    <!--Imagen y texto -->
  </div>
  <div v-else-if="i === '2'">
    <!--Imagen y texto -->
  </div>
  <div v-else>
    <!--Imagen y texto -->
  </div>
</template>

```

**PeliculasParamsView** será la vista que recibirá estos parámetros y evaluará qué contenido debe mostrar según el id que recibe.

En el script del componente, creamos el manejo de estos valores:

```

V PeliculasParamsView.js
export default {
  name: 'PeliculasParamsView',
  data: function () {
    return {
      i: '',
      e: ''
    }
  },
  mounted: function () {
    this.recuperarPeli()
  },
  methods: {
    recuperarPeli: function () {
      this.i = this.$route.params.id
      this.e = this.$route.params.estreno
    }
  }
}

```

En la función data, definimos el estado local del componente con dos variables **"i"** y **"e"** inicializadas como cadenas vacías que guardarán los valores de los parámetros recibidos de la URL.

El methods, creamos la función **recuperarPeli** que recuperará y mostrará los parámetros de la URL. Asignamos **el parámetro id a la variable "i"**, y el **parámetro estreno a la variable "e"**.

Por último, en el hook mounted se llama al método **"recuperarPeli"** después que el componente esté listo.



Cabe aclarar que hay distintos escenarios que pueden surgir en los desarrollos para satisfacer diferentes necesidades. Es importante tener presente que si definimos este tipo de rutas y falta algún parámetro, Vue no sabrá qué mostrar dentro de **RouterView** porque la ruta no coincidirá con la que hemos establecido en la configuración del **index.js**.

Podría ser el caso de un formulario cuyos campos no hemos validado correctamente y estamos pasando sus valores por parámetros para editarlos en otra vista. Esto podría provocar errores de visualización.

Por tal motivo, es importante validar el ingreso de datos y, en la edición de los mismos, no dejar elementos vacíos que puedan perjudicar la usabilidad del sistema.



<p style="text-align: center;"><b>Datos guardados</b></p> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px;"> <p><b>Nombre:</b> Avatar: la leyenda de Aang</p> <p><b>Comentario:</b> Excelente serie animada</p> <p><b>Temática:</b></p> <p>Animacion</p> <p>Ciencia Ficción</p> <p><b>Estreno:</b></p> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <span style="border: 1px solid #ccc; padding: 5px 10px;">Editar</span> <span style="border: 1px solid #ccc; padding: 5px 10px;">Borrar</span> </div> </div>	<p>Si se guarda en localStorage y <b>falta el dato de la fecha de estreno</b>, cuando el usuario quiera <b>editar</b> esta información y pasarla por parámetros en la URL, no veremos la pantalla del formulario para editar.</p>
--	---

localhost:5173/editar/Avatar:%20la%20leyenda%20de%20Aang/Excelente%20serie%20animada/Animacion/Ciencia%20Ficción/177449270591

Al faltar el parámetro de la fecha de estreno no coincide con la ruta y no vemos el listado de películas.

### Enrutamiento con Parámetros

Home Ingresar Ver

→ Todos los derechos reservados © Javier García

*¡Estamos preparados para practicar con Router-view y paso de parámetros!*



Hemos llegado así al final de esta clase en la que vimos:

- Instalación de Vue-router.
- Estructura base del proyecto.
- Configuración de rutas.
- Paso por parámetros entre vistas (\$route.params).



Te esperamos en la **clase en vivo** de esta semana.  
No olvides realizar el **desafío semanal**.

**¡Hasta la próxima clase!**

# Bibliografía

Router Vue.js. (s.f.). *Documentación Oficial: Introducción a Router*. Recuperado de <https://router.vuejs.org/introduction.html>

Router Vue.js. (s.f.). *Documentación Oficial: Rutas con Parámetros*. Recuperado de <https://router.vuejs.org/guide/essentials/dynamic-matching.html>