

EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: INSTALANDO VUE-ROUTER.
- EXERCISE 2: RUTAS ESTÁTICAS.
- EXERCISE 3: RUTAS DINÁMICAS.

EXERCISE 1: INSTALANDO VUE-ROUTER.

Para realizar los ejercicios con vue-router, revisaremos como instalarlo cuando iniciemos un nuevo proyecto con vue/cli.

INSTALANDO:

Abriremos Visual Studio Code, e iremos al terminal. Aquí escribiremos el comando: `vue create nombre_proyecto`.

```
alejandros-MacBook-Pro:cue8 bonilla$ vue create cue8
```

Luego, vamos a seleccionar: **"Manually select features"**.

Vue CLI v4.5.6

New version available 4.5.6 → 4.5.9
Run `npm i -g @vue/cli` to update!

? Please pick a preset:

Default ([Vue 2] babel, eslint)

Default (Vue 3 Preview) ([Vue 3] babel, eslint)

> Manually select features

Aparte de las predefinidas, babel y linter; vamos a seleccionar router.

```
Vue CLI v4.5.6

New version available 4.5.6 → 4.5.9
Run npm i -g @vue/cli to update!

? Please pick a preset: Manually select features
? Check the features needed for your project:
  ● Choose Vue version
  ● Babel
  ○ TypeScript
  ○ Progressive Web App (PWA) Support
  > ● Router
  ○ Vuex
  ○ CSS Pre-processors
  ● Linter / Formatter
  ○ Unit Testing
  ○ E2E Testing
```

Seleccionaremos la versión 2.X.

```
Vue CLI v4.5.6

New version available 4.5.6 → 4.5.9
Run npm i -g @vue/cli to update!

? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Linter
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
  > 2.x
  3.x (Preview)
```

Usaremos history mode, y seleccionaremos “y”.

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Linter
? Choose a version of Vue.js that you want to start the project with 2.x
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) y
```

Luego, seleccionaremos las opciones por default.

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Linter
? Choose a version of Vue.js that you want to start the project with 2.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a linter / formatter config: (Use arrow keys)
> ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier
```

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Linter
? Choose a version of Vue.js that you want to start the project with 2.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: (Press <space> to select, <a> to toggle all, <i> to invert selection)
> ☒ Lint on save
  ☐ Lint and fix on commit
```

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Linter
? Choose a version of Vue.js that you want to start the project with 2.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.? (Use arrow keys)
> In dedicated config files
  In package.json
```

Por último, cuando nos pregunte si queremos guardar esta configuración de proyecto, colocaremos “N”.

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Linter
? Choose a version of Vue.js that you want to start the project with 2.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
? Save this as a preset for future projects? (y/N) N
```

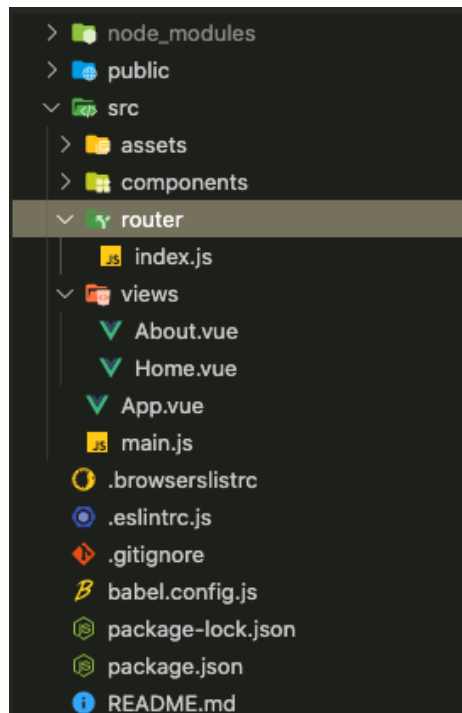
Una vez instalado, visualizaremos:

```
🎉 Successfully created project cue8.
👉 Get started with the following commands:

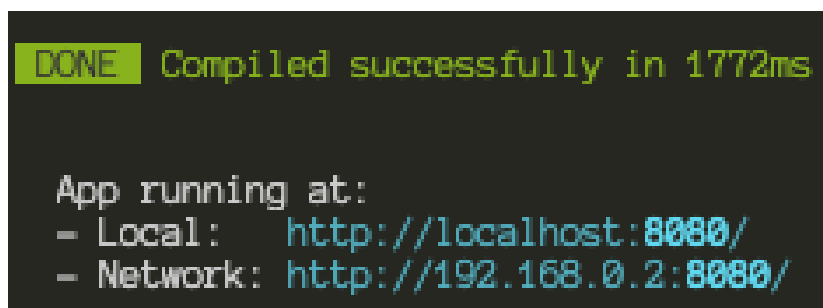
$ cd cue8
$ npm run serve
```

ESTRUCTURA DE PROYECTO:

Al abrir el proyecto recién creado desde Visual Studio Code, veremos que hay 2 carpetas nuevas: “router” y “views”.



Si levantamos el proyecto con “npm run serve”, observaremos:





En la parte superior de la página, se encuentran 2 links, uno a Home y otro a About. Si hacemos clic en About, veremos que la pagina cambia de ruta y nos muestra el componente.

[Home](#) | [About](#)



Welcome to Your Vue.js App

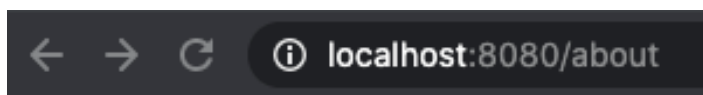
For a guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#) [router](#) [eslint](#)

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)



[Home](#) | [About](#)

This is an about page

EXERCISE 2: RUTAS ESTÁTICAS.

Ahora, vamos a continuar trabajando en el proyecto creado anteriormente. En este ejercicio estudiaremos qué son las rutas estáticas.

La mayoría de los sitios web por su tamaño, poseen más de una ruta para mostrar sus diferentes contenidos, es por esto que instalaremos vue-router. Es el momento de revisar las diferentes rutas con las cuales podemos trabajar, en el caso de este ejercicio, veremos las **estáticas**.

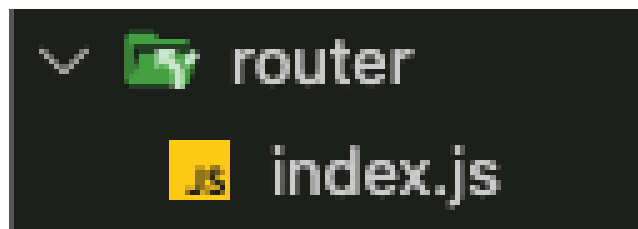
RUTAS ESTÁTICAS:

Como su nombre lo indica, son aquellas que siempre tendrán el mismo path(camino), para representar un contenido.

Componente	Ruta definida	Acceso
Home.vue	/	http://localhost:8080
About.vue	/about	http://localhost:8080/about
Contact.vue	/contact	http://localhost:8080/contact

ARCHIVO DE RUTAS:

Antes de comenzar, vamos a estudiar el archivo index.js, que se encuentra dentro de la carpeta router. Es el encargado de manejar todas las rutas existentes en nuestra aplicación.



```
1 import Vue from 'vue'
2 import VueRouter from 'vue-router'
3 import Home from '../views/Home.vue'
4
5 Vue.use(VueRouter)
6
7 const routes = [
8   {
9     path: '/',
10    name: 'Home',
11    component: Home
12  },
13  {
14    path: '/about',
15    name: 'About',
16    // route level code-splitting
17    // this generates a separate chunk (about.[hash].js) for
18    // this route
19    // which is lazy-loaded when the route is visited.
20    component: () => import(/* webpackChunkName: "about" */
21      '../views/About.vue')
22  }
23 ]
24 const router = new VueRouter({
25   mode: 'history',
26   base: process.env.BASE_URL,
27   routes
28 })
29 export default router
```

Si nos fijamos, cada ruta está declarada dentro de un objeto, y cada objeto se compone de:

- **path:** ruta url.
- **name:** nombre de ruta.
- **component:** componente a mostrar cuando se visite el path declarado.

ROUTER-VIEW Y ROUTER-LINK

Si vamos al archivo App.vue, nos daremos cuenta de que tiene la siguiente estructura:

```
1 <template>
2   <div id="app">
3     <div id="nav">
4       <router-link to="/">Home</router-link> |
5       <router-link to="/about">About</router-link>
6     </div>
7     <router-view/>
8   </div>
9 </template>
```

Tanto la etiqueta `<router-link>`, como `<router-view>`, pertenecen a vue-router. Éstas nos permiten:

- `<router-link>`: generar un enlace a ruta definida.
- `<router-view>`: muestra el componente definido para la ruta visitada; home cuando visitamos "/", o about si visitamos "/about".

CARPETA VIEWS:

En esta carpeta deben ir las vistas principales. Si necesitamos crear componentes para estas vistas, debemos dejarlas en la carpeta "components".

Vamos a crear la vista Main.vue, con el siguiente contenido:

```
1 <template>
2   <h1> Vista principal</h1>
3 </template>
4
5 <script>
6 export default {
7   name: 'Main-component',
8   // props: {},
9   data: function() {
10    return {}
11  },
12  // computed: {},
13  methods: {
14    // -- Metodos
```

```
15   },
16   // components: {},
17 }
18 </script>
19
20 <style scoped>
21
22 </style>
```

Una vez guardado el Main.vue, creamos otro archivo llamado "Contact.vue", con el siguiente contenido:

```
1 <template>
2   <h1> Vista de contacto</h1>
3 </template>
4
5 <script>
6 export default {
7   name: 'Contact-Component',
8   // props: {},
9   data: function() {
10     return {}
11   },
12   // computed: {},
13   methods: {
14     // -- Metodos
15   },
16   // components: {},
17 }
18 </script>
19 <style scoped></style>
```

CARPETA ROUTER.

Lo siguiente, será vincular el archivo Main.vue a la ruta "/", que se encuentra en el archivo index.js de la carpeta router.

```
1 {
2   path: '/',
3   name: 'Home',
4   component: Home
5 },
```

Y lo reemplazaremos por:



```
1 {  
2   path: '/',  
3   name: 'Main',  
4   component: Main  
5 },
```

No olvidemos importar la vista Main.vue, y eliminar el import de Home.vue.

```
1 import Main from '@views/Main.vue'
```

Luego, crearemos otra ruta llamada “/contact”, la cual nos mostrará el componente Contact.vue.

```
1 {  
2   path: '/contact',  
3   name: 'Contact',  
4   component: Contact,  
5 },
```

No olvidemos importar la vista Contact.vue.

```
1 import Main from '@views/Main.vue'  
2 import Contact from '@views/Contact.vue'
```

Si levantamos la aplicación con el comando “npm run serve”, se visualizará así:

[Home](#) | [Contact](#) | [About](#)

Vista principal



Y si hacemos clic en Contact, seremos redirigidos a “/contact”, y podremos ver el contenido de Contact.vue.

[Home](#) | [Contact](#) | [About](#)

Vista de contacto

EXERCISE 3: RUTAS DINÁMICAS.

INDICACIONES:

Para realizar este ejercicio, continuaremos trabajando con el proyecto que hemos creado anteriormente.

En algunos casos, cuando trabajamos con rutas, es necesario definir algunas de ellas como rutas dinámicas, para así poder tener una vista genérica que muestre diferentes datos, según el dato dinámico de la ruta.

Componente	Ruta dinámica definida	Acceso
Foto.vue	/fotos/:id	Localhost:8080/fotos/2
		Localhost:8080/ fotos /300
		Localhost:8080/ fotos /1

En primer lugar, crearemos una vista llamada fotos.vue, dentro de la carpeta “views”. Ésta será asociada a una ruta estática, que nos servirá para mostrar todas las fotos disponibles.

El archivo quedará de esta forma:

```

1 <template>
2   <div>
3     <h1>Álbum</h1>
4     <div class="grid_foto">
5       <div v-for="i in 6" :key="i">
6         <h3>Foto {{i}}</h3>
7         <a @click="redirect(i)">
8           
9         </a>
10      </div>
11    </div>
12  </div>
13 </template>
14 <script>
15 export default {
16   name: 'fotos-component',
  
```

```

17 // props: {},
18 data: function(){
19   return {}
20 },
21 // computed: {},
22 methods: {
23   srcImage(id) {
24     return `https://picsum.photos/200/300?grayscale/${id}`
25   },
26   redirect(id) {
27     this.$router.push(`/fotos/${id}`)
28   }
29 },
30 // components: {},
31 }
32 </script>
33 <style scoped>
34   .grid_foto{
35     display:grid;
36     grid-template-columns: 1fr 1fr 1fr;
37   }
38 </style>
  
```

Una vez creado el archivo, lo debemos vincular al archivo de rutas.

```

1 {
2   path: '/fotos',
3   name: 'Fotos',
4   component: Fotos,
5 },
  
```

Luego en el archivo App.vue, generaremos un link, a través de <router-link>

```

1 <template>
2   <div id="app">
3     <div id="nav">
4       <router-link to="/">Home</router-link> |
5       <router-link to="/fotos">Fotos</router-link> |
6       <router-link to="/contact">Contact</router-link> |
7       <router-link to="/about">About</router-link>
8     </div>
9     <router-view/>
10   </div>
11 </template>
  
```

Si guardamos y levantamos la aplicación, veremos:

[Home](#) | [Fotos](#) | [Contact](#) | [About](#)

Vista principal

Si hacemos clic sobre Fotos:

[Home](#) | [Fotos](#) | [Contact](#) | [About](#)

Álbum

Foto 1



Foto 2



Foto 3



Foto 4



Foto 5



Foto 6



Si quisiéramos acceder, por ejemplo, a la foto 5 o a la 4, es donde debemos ocupar **rutas dinámicas**.

RUTA DINÁMICA:

Para poder acceder a una foto en específico, debemos generar un nuevo componente en views, llamado "Foto.vue", el cual estará enlazado a una ruta dinámica.

COMPONENTE FOTO.VUE:

Creando el archivo Foto.vue dentro de views, el archivo debe quedar así:

```
1 <template>
2   <div>
3     <h1>Foto {{id}}</h1>
4     
5     <router-link to="/fotos">Volver</router-link>
6   </div>
7 </template>
8
9 <script>
10 export default {
11   name: 'foto-component',
12   // props: {},
13   data: function() {
14     return {
15       src: '',
16     }
17   },
18   computed: {
19     id() {
20       return this.$route.params.id
21     }
22   },
23   methods: {
24     // -- Metodos
25   },
26   // components: {},
27   created() {
28     this.src =
29     https://picsum.photos/200/300?grayscale/${this.id}`
30   }
31 }
32 </script>
33
34 <style scoped></style>
35
```

En este componente, se utilizan las propiedades de router para obtener la id **"this.\$route.params.id"**.

RUTEANDO FOTO.VUE:

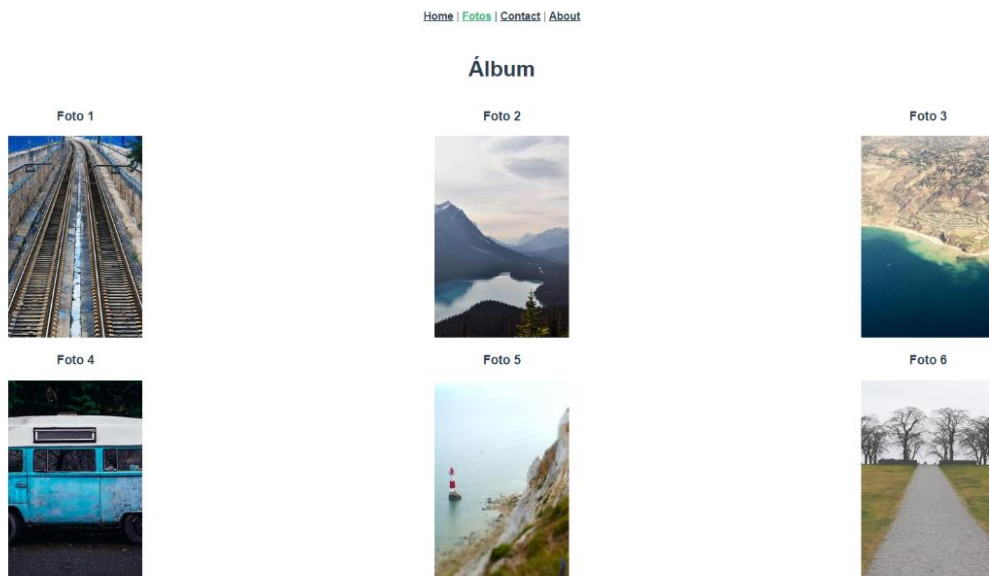
```
1 import Foto from '@views/Foto.vue'
```

Luego de importar el componente Foto.vue, debemos rutiarlo. Aquí agregaremos la ruta dinámica en el path, acompañado de dos puntos “:id”.


```
1 {
2   path: '/fotos/:id',
3   name: 'Foto',
4   component: Foto,
5 },
```

Al indicar “:id”, vue sabe que esa parte de la ruta irá cambiando.

Ahora, si hacemos clic en alguna imagen de la ruta “/fotos”, seremos redireccionados a una vista específica de la foto.

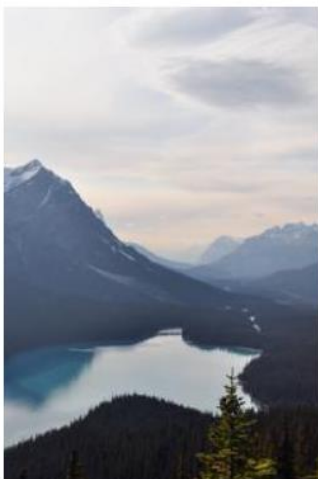


Por ejemplo, si hacemos clic en la foto 2, seremos redirigidos a:

 localhost:8081/fotos/2

[Home](#) | [Fotos](#) | [Contact](#) | [About](#)

Foto 2



[Volver](#)