



Doc - Frontend Zodíaco - Vue

Repositorio

GitHub - SZDohmen/lab4-examen-vue-zodiaco


Contribute to SZDohmen/lab4-examen-vue-zodiaco development by creating an account on GitHub.

<https://github.com/SZDohmen/lab4-examen-vue-zodiaco>

men/**lab4-examen-zodiaco**

0 Issues 0 Stars 0 Forks

JSON url

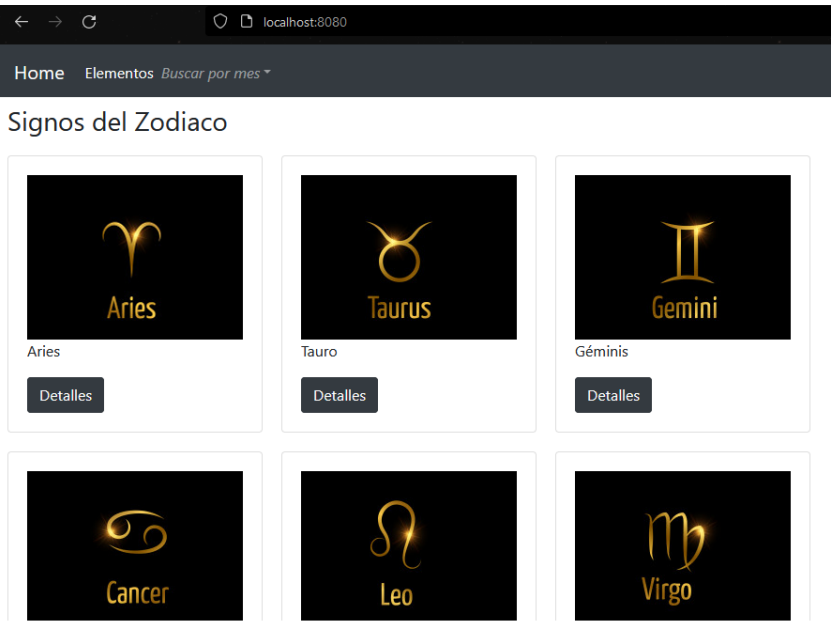
 <http://179.43.113.170:8082/test/tb/zodiaco.json>

Estructura del proyecto

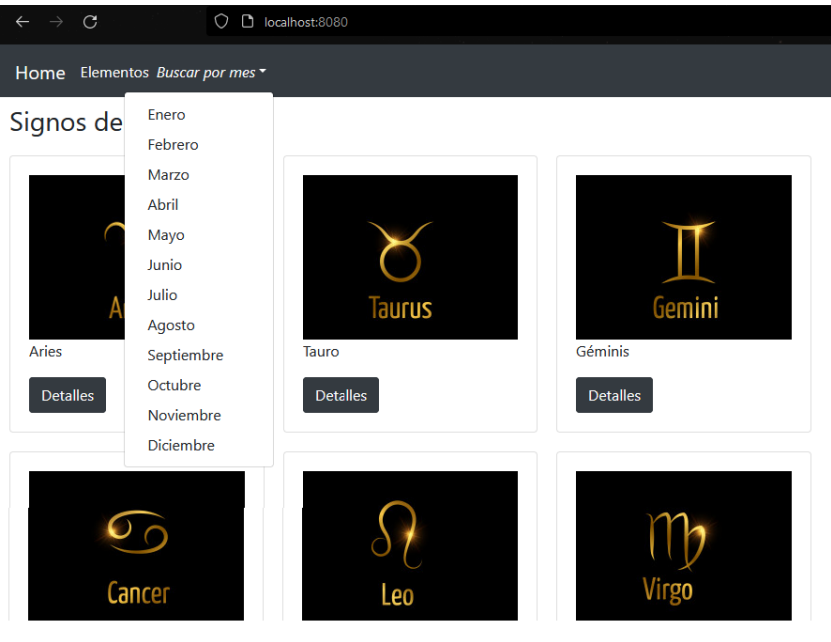
Navbar superior de opciones siempre presente en todas las páginas. Corresponde al Componente **Menu** con elemento:

- *Home* → redirecciona al la url localhost:8080
- *Elementos* → redirecciona al Componente **Elementos** enviando datos recuperados del JSON.
- *Buscar por mes* → scroll sin funcionalidad.

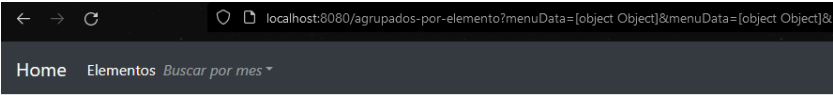
Componente **Home** muestra un conjunto de cartas con los datos traídos del JSON. Dentro de cada carta contiene un botón *Detalles* que redirecciona al Componente **Detalle** junto con los datos extraídos del JSON para esa carta.



Componente **Elementos** con los datos enviados desde **Home**



Componente **Detalle** con los datos obtenidos de la carta.



Agrupados por elementos

ELEMENTO	SIGNOS
F u e g o	Aries Leo Sagitario
T i e r r a	Tauro Virgo Capricornio
A i r e	Géminis Libra Acuario
A g u a	Cáncer Escorpio Piscis

Volver



Géminis

Descripción: Los gemelos Cástor y Pólux. Pólux era inmortal, no así su hermano Cástor. Cuando Cástor murió, Pólux ofreció su inmortalidad por salvar a su hermano.

Elemento: Aire

Símbolo: Los Gemelos

Meses:

- 5
- 6

Volver

Creación del proyecto

Crear carpeta **proyecto-vue-6** con:

- Carpeta **public**

→ Copiar **index.html** y **favicon.ico** (en caso de tenerlo).

→ Si no, código del index.html:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <!-- <link rel="icon" href="<%= BASE_URL %>favicon.ico"> -->
    <title><%= htmlWebpackPlugin.options.title %></title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %>
        doesn't work properly without JavaScript enabled. Please enable
        it to continue.</strong>
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

- Carpeta **src**:

→ Carpeta plugins con:

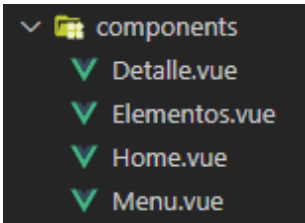
- Archivo **bootstrap-vue.js**

```
import Vue from 'vue'

import BootstrapVue from 'bootstrap-vue'
import 'bootstrap/dist/css/bootstrap.min.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'
import 'leaflet/dist/leaflet.css'

Vue.use(BootstrapVue)
```

→ Carpeta **components** (vacíos para empezar).



- Archivo **package.json**:

```
{
  "name": "proyecto-vue-6",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "axios": "^0.21.1",
    "bootstrap-vue": "^2.1.0",
    "core-js": "^3.6.5",
    "leaflet": "^1.7.1",
    "vue": "^2.6.11",
    "vue-axios": "^3.2.5",
    "vue-axios-cors": "^1.0.1",
    "vue-router": "^3.2.0",
    "vue2-leaflet": "^2.7.0",
    "vuex": "^3.4.0"
  },
  "devDependencies": {
    "@babel/polyfill": "^7.7.0",
    "@vue/cli-plugin-babel": "~4.5.0",
    "@vue/cli-plugin-eslint": "~4.5.0",
    "@vue/cli-plugin-router": "~4.5.0",
    "@vue/cli-plugin-vuex": "~4.5.0",
    "@vue/cli-service": "~4.5.0",
    "babel-eslint": "^10.1.0",
    "bootstrap": "^4.3.1",
    "eslint": "^6.7.2",
    "eslint-plugin-vue": "^6.2.2",
    "mutationobserver-shim": "^0.3.3",
    "popper.js": "^1.16.0",
    "portal-vue": "^2.1.6",
    "sass": "^1.19.0",
    "sass-loader": "^8.0.0",
    "vue-cli-plugin-bootstrap": "~0.4.0",
    "vue-template-compiler": "^2.6.11"
  },
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true
    },
    "extends": [
      "plugin:vue/essential",
      "eslint:recommended"
    ],
    "parserOptions": {
      "parser": "babel-eslint"
    },
    "rules": {}
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead"
  ]
}
```

→ Archivo **main.js** (base para empezar).

```
import './plugins/bootstrap-vue'
import Vue from 'vue'
import App from './App.vue'

new Vue({
  render: h => h(App)
}).$mount('#app')
```

```
]
}
```

→ Archivo **App.vue** (base para empezar).

```
<template>
  <div>
    <h3> App.vue </h3>
  </div>
</template>

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

- Instalación de carpeta **node_modules** → una vez creado todos los archivos anteriores.

```
npm i
```

Crea el archivo **package-lock.json**

!! Una vez hecho todo lo anterior, podemos ejecutar el comando para levantar la aplicación con:
→ `npm run serve`

Configuración de Archivos

vue.config.js

Creamos archivo **vue.config.js** en la raíz del proyecto. Agregamos el proxy.

```
vue.config.js > ...
1  module.exports = {
2    configureWebpack: {
3      devServer: {
4        proxy: "http://179.43.113.170:8082/"
5      }
6    }
7  }
```

main.js

```
Vue.config.productionTip = false;
```

Variable que por consola indica si estamos usando Vue en producción.
Con *false* la desactivamos.

```
Vue.use(VueRouter);
const routes = [
  { path: '/', component: Home },
  { path: '/detalle/:id', component: Detalle },
  { path: '/agrupados-por-elemento', component: Elementos },
];
const router = new VueRouter( {
  mode: 'history',
  base: process.env.BASE_URL,
  routes
});
```

VueRouter para el enrutado de los componentes . En *routes* definimos las rutas y la integramos al *router*. La *base* de este corresponde a la opción pública (proxy) del archivo *vue.config.js*

```
new Vue({
  router:router,
  render: h => h(App)
}).$mount('#app')
```

Creamos nuevo *Vue()* como núcleo o ambiente de trabajo. Con *render* crea una estructura HTML para la aplicación principal (App) y *mount* indica dónde mostraremos al App.

```
src > main.js > ...
1  import './plugins/bootstrap-vue'
2  import Vue from 'vue'
3  import App from './App.vue'
4  import VueRouter from 'vue-router'
5  import axios from 'axios'
6  import VueAxios from 'vue-axios'
7  import Home from './components/Home.vue'
8  import Detalle from './components/Detalle.vue'
9  import Elementos from './components/Elementos.vue'
10
11  Vue.config.productionTip = false;
12
13  Vue.use(VueRouter);
14  const routes = [
15    { path: '/', component: Home },
16    { path: '/detalle/:id', component: Detalle },
17    { path: '/agrupados-por-elemento', component: Elementos },
18  ];
19  const router = new VueRouter( {
20    mode: 'history',
21    base: process.env.BASE_URL,
22    routes
23  } );
24
25  new Vue({
26    router:router,
27    render: h => h(App)
28  }).$mount('#app')
29
30  Vue.use(VueAxios, axios)
```

```
Vue.use(VueAxios, axios)
```

Con *use()* encadenamos los plugins de *axios* (para operaciones como cliente HTTP) y *VueAxios* (para permisos del navegador).

App.vue

Archivos vue subdivididos en tres partes: *templates*, *scripts* y *style* (no usado dentro de esta Api). Etiqueta *template* contiene la vista de la página. Etiqueta *script* contiene el código general y la lógica escrito en JavaScript (con elementos propios de Vue Framework).

Sección <template>

```
<div>
  <Menu v-bind:datosArray="datosArray" />
  <router-view v-bind:datosArray="datosArray" />
</div>
```

Etiqueta del componente Menu integra una props *datosArray* que contiene los datos de una variable definida en App.vue también llamada *datosArray*. Etiqueta *router-view* renderiza el VueRouter del main.js

```
src > App.vue > {} "App.vue"
1  <template>
2    <div>
3      <Menu v-bind:datosArray="datosArray" />
4      <router-view v-bind:datosArray="datosArray" />
5    </div>
6  </template>
7
```

Sección <script>

```
data() {
  return { datosArray: [], };
},
```

Dentro de *data()*, creamos y retornamos variable *datosArray* vacía de tipo arreglo

```
methods: {
  async getData() {
```

```
    try {
      let dataJson = await axios.get(
        "http://localhost:8080/test/tb/zodiaco.json"
      );
      this.datosArray = dataJson.data;
      console.log(this.datosArray);
    } catch (error) {
      console.error('Error al obtener los datos: ' + error);
    }
  },
},
```

En *methods: {}* declaramos los métodos a usar. Función *getData()* trae los datos del JSON con petición *get* del *axios*, guardando la respuesta en variable local *dataJson*. Guardamos unicamente la *data* dentro de la variable ya definida en *data()* → *datosArray*

```
src > App.vue > {} "App.vue"
8   <script>
9   import axios from 'axios';
10  import Menu from './components/Menu.vue'
11  export default{
12    name:'App',
13    components:{
14      Menu
15    },
16    data() {
17      return { datosArray: [], };
18    },
19    methods: {
20      async getData() {
21        try {
22          let dataJson = await axios.get(
23            "http://localhost:8080/test/tb/zodiaco.json"
24          );
25          this.datosArray = dataJson.data;
26          console.log(this.datosArray);
27        } catch (error) {
28          console.error('Error al obtener los datos: ' + error);
29        }
30      },
31    },
32    mounted() {
33      this.getData();
34    },
35  }
36  </script>
```

```
mounted() {
  this.getData();
},
```

Manda a llamar las funciones dentro de *methods*

Menu.vue

```
<script>
  export default {
    name:'Menu',
    props: {
      datosArray:Array
    },
  }
</script>
```

Recibe la *props* enviada desde App.vue en variable local que debe tener el MISMO NOMBRE *datosArray* y la declaramos como arreglo.

El nombre de la props se define en el componente que la recibe mientras que el componente que la manda debe 'llamarla' para enviar los datos.

⇒ El *template* contiene un *router-link* que redirige al componente Elementos a través de su url dentro de *path* y le envía dentro *query* los datos guardados en de una props (*menuData*) definida en el componente Elementos.

```
<template>
<div>
  <div>
    <b-navbar toggleable="lg" type="dark" variant="dark">
      <b-navbar-brand href="/">Home</b-navbar-brand>

      <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>

    <b-collapse id="nav-collapse" is-nav>
      <b-navbar-nav>
        <b-nav-text>
          <router-link :to="{
            path: '/agrupados-por-elemento',
            query: { menuData:this.datosArray }
          }"> Elementos </router-link>
        </b-nav-text>
      </b-navbar-nav>

      <b-navbar-nav>
        <b-nav-item-dropdown right>
          <template #button-content>
            <em>Buscar por mes</em>
          </template>
          <b-dropdown-item href="#">Enero</b-dropdown-item>
          <b-dropdown-item href="#">Febrero</b-dropdown-item>
          <b-dropdown-item href="#">Marzo</b-dropdown-item>
          <b-dropdown-item href="#">Abril</b-dropdown-item>
          <b-dropdown-item href="#">Mayo</b-dropdown-item>
          <b-dropdown-item href="#">Junio</b-dropdown-item>
          <b-dropdown-item href="#">Julio</b-dropdown-item>
          <b-dropdown-item href="#">Agosto</b-dropdown-item>
          <b-dropdown-item href="#">Septiembre</b-dropdown-item>
          <b-dropdown-item href="#">Octubre</b-dropdown-item>
          <b-dropdown-item href="#">Noviembre</b-dropdown-item>
          <b-dropdown-item href="#">Diciembre</b-dropdown-item>
        </b-nav-item-dropdown>
      </b-navbar-nav>
    </b-collapse>
  </b-navbar>
</div>
</div>
</template>
```

Elementos.vue

```
<template>
<div>
  <h3 style="margin: 10px"> Agrupados por elementos </h3>

  <b-card style="max-width: 800px; margin: 10px">
    <table class="table mx-auto">
      <thead>
        <tr>
          <th>ELEMENTO</th>
          <th>SIGNOS</th>
        </tr>
      </thead>
      <tbody>
        <tr v-for="(signos, index) in array" :key="index">
          <td v-for="(elemento, indice) in signos" :key="indice">
            <b v-for="(i, c) in elemento" :key="c">{{elemento[c]}} </b>
          </td>
        </tr>
      </tbody>
    </table>

    <router-link class="btn btn-dark" v-bind:to="`/${`">
      Volver
    </router-link>
  </b-card>
</div>
</template>
```

```
<script>
export default {

  name: 'Elementos',
  props: {
    menuData:[]
  },

  data() {
    console.log(this.$route.query.menuData);
    return { array:new Map(), }
  },

  methods:{
    filterByElement(){
      let dataFromHome = this.$route.query.menuData;
      let tempArray = new Map();

      dataFromHome.map( data => {
        if(tempArray.get(data.elemento) === undefined )
          tempArray.set(data.elemento, []);
        }
        tempArray.get(data.elemento).push(data.signo);
      });

      this.array = tempArray;
    }
  },

  mounted(){
    this.filterByElement()
  }

}
</script>
```

En el *templete* tenemos tres if anidados que se encargan de mostrar el nombre del elemento y los signos que pertenecen a él en conjunto con lo extraído dentro del *script*. Está horrible pero funciona. No es de mi autoría...

Home.vue

```
<script>
export default{
  name:'Home',
  props: {
    datosArray:Array
  }
}
</script>
```

Con la props *datosArray* recibida desde App.vue, dentro del *template*, en su segundo *div* recorre el arreglo de datos con un for (v-*for* propio de Vue Framework)

Etiqueta *router-link* envía los datos filtrados por este for para enviarlos a la url correspondiente para Detalle dentro de una props *detalleItem*

```
<template>
<div>
  <h3 style="margin:10px"> Signos del Zodiaco </h3>
  <div class="d-flex flex-wrap" style="width: auto;">
    <div v-for="(item, i) in datosArray" :key="i" style="margin:10px">
      <b-card style="max-width: 20rem;">
        
        <b-card-text> {{ item.signo }} </b-card-text>
        <router-link class="btn btn-dark" :to="{
          path: '/detalle/'+item.id,
          query: { detalleItem:item }
        }" > Detalles </router-link>
      </b-card>
    </div>
  </div>
</div>
</template>
```

Detalle.vue

```
<script>
export default {
  name: "Detalle",
  data() {
    console.log(this.$route.query);
    return {
      dataRecibida:this.$route.query.detalleItem
    }
  }
}
```

```
<template>
<div>
  <b-card style="max-width: 800px; margin: 10px">
    <div>
      

      <b-card-text>
```

```
    },  
  }  
</script>
```

En el *script* usamos los datos obtenidos desde la ruta url (no use la props acá porque no sabía como hacerlo aún).

Obtenemos *detalleItem* de la query de la url y la guardamos en variable local *dataRecibida*.

```
<h5> {{ dataRecibida.signo }} </h5>  
</b-card-text>  
  
<b-card-text>  
  Descripción: {{ dataRecibida.descripcion }}  
</b-card-text>  
  
<b-card-text>  
  Elemento: {{ dataRecibida.elemento }}  
</b-card-text>  
  
<b-card-text>  
  Símbolo: {{ dataRecibida.simbolo }}  
</b-card-text>  
  
Meses:  
<ul v-for="(item, i) in dataRecibida.meses" :key="i" >  
  <li> {{ item }} </li>  
</ul>  
  
<router-link class="btn btn-dark" v-bind:to="`/${`>  
  Volver  
</router-link>  
</div>  
</b-card>  
</div>  
</template>
```