

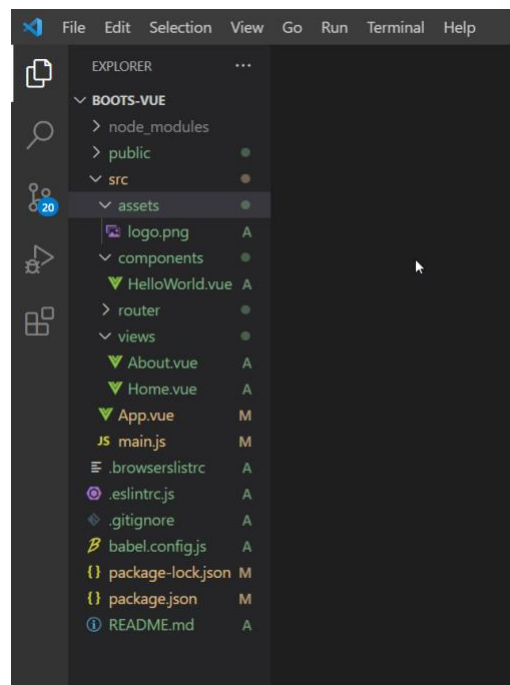
EXERCISES QUE TRABAJAREMOS EN LA CUE

- EXERCISE 1: INSTALANDO BOOTSTRAP VUE.
- EXERCISE 2: UTILIZANDO BOOTSTRAP VUE.
- EXERCISE 3: INSTALANDO VUETIFY.
- EXERCISE 4: UTILIZANDO VUETIFY.

EXERCISE 1: INSTALANDO BOOTSTRAP VUE

Bootstrap-Vue es una librería especializada para VUE, trabajado con Bootstrap. No sólo es compatible con los componentes de Bootstrap y el sistema de rejilla, sino que también incluye soporte para las **Vue.js Directives**, lo que nos proporciona el conjunto completo de características del ecosistema **Vue.js**.

Lo primero que se debe hacer, es crear un proyecto con **vue-cli** en Visual Studio Code.

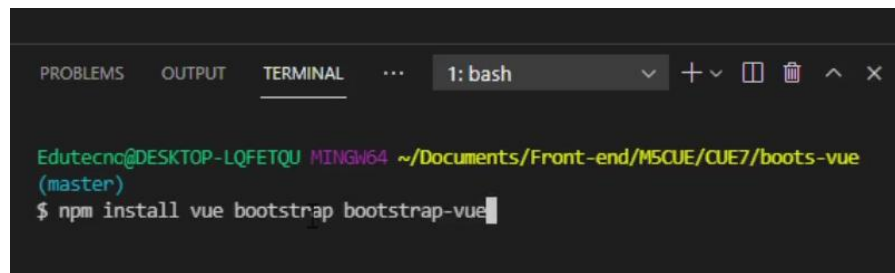


Después, iremos a la página oficial de [BootstrapVue](#), y presionamos get started.

En el apartado using module bundlers, lo primero que nos aparecerá será el comando para instalar **Bootstrap-Vue** en nuestro proyecto, puede ser con **npm** o con **yarn**.

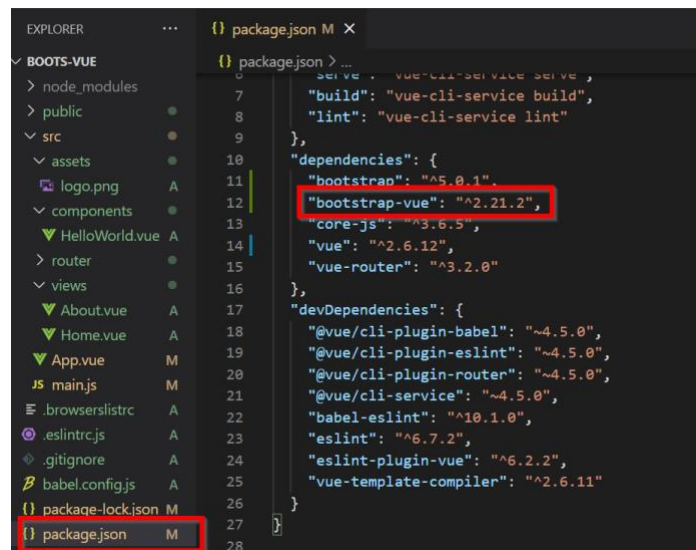
```
# With npm
npm install vue bootstrap bootstrap-vue

# With yarn
yarn add vue bootstrap bootstrap-vue
```



```
Edutecno@DESKTOP-LQFETQU MINGW64 ~/Documents/Front-end/M5CUE/CUE7/boots-vue
(master)
$ npm install vue bootstrap bootstrap-vue
```

Una vez instalado **Bootstrap-Vue**, podemos corroborar si todo está bien en el archivo **package.json**, en donde observaremos qué dependencias está implementando **Bootstrap-Vue** en nuestro proyecto.



```
EXPLORER    ...    {} package.json M X
BOOTS-VUE
  > node_modules
  > public
  > src
    > assets
    > components
    > HelloWorld.vue A
    > router
    > views
      > About.vue A
      > Home.vue A
      > App.vue M
    JS main.js M
    .browserslistrc A
    .eslintrc.js A
    .gitignore A
    babel.config.js A
    {} package-lock.json M
    {} package.json M
    {} package.json M X
      {} package.json > ...
        7 serve: "vue-cli-service serve",
        8 "build": "vue-cli-service build",
        9 "lint": "vue-cli-service lint"
      },
      10 "dependencies": {
      11   "bootstrap": "^5.0.1",
      12   "bootstrap-vue": "^2.21.2",
      13   "core-js": "^3.6.5",
      14   "vue": "^2.6.12",
      15   "vue-router": "^3.2.0"
      16 },
      17 "devDependencies": {
      18   "@vue/cli-plugin-babel": "~4.5.0",
      19   "@vue/cli-plugin-eslint": "~4.5.0",
      20   "@vue/cli-plugin-router": "~4.5.0",
      21   "@vue/cli-service": "~4.5.0",
      22   "babel-eslint": "^10.1.0",
      23   "eslint": "^6.7.2",
      24   "eslint-plugin-vue": "^6.2.2",
      25   "vue-template-compiler": "^2.6.11"
      26 }
      27
      28
```

Si seguimos revisando la página de **Bootstrap-Vue**, notaremos que debemos importar la librería de Bootstrap en el archivo **main.js**.

Then, register BootstrapVue in your app entry point (typically **app.js** or **main.js**):

```
import Vue from 'vue'
import { BootstrapVue, IconsPlugin } from 'bootstrap-vue'

// Import Bootstrap and BootstrapVue CSS files (order is important)
import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'

// Make BootstrapVue available throughout your project
Vue.use(BootstrapVue)
// Optionally install the BootstrapVue icon components plugin
Vue.use(IconsPlugin)
```

Copiamos las líneas que necesitamos, y las pegamos en nuestro código. También importaremos los estilos **css** propios de esta librería (**bootstrap.css**, **Bootstrap-vue.css**).

JS main.js M X

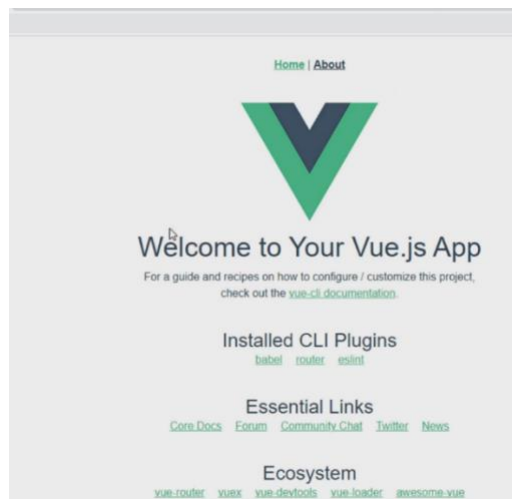
src > JS main.js > ...

```
1 import Vue from 'vue'
2 import App from './App.vue'
3 import router from './router'
4
5 import { BootstrapVue, IconsPlugin } from 'bootstrap-vue'
6 import 'bootstrap/dist/css/bootstrap.css'
7 import 'bootstrap-vue/dist/bootstrap-vue.css'
```

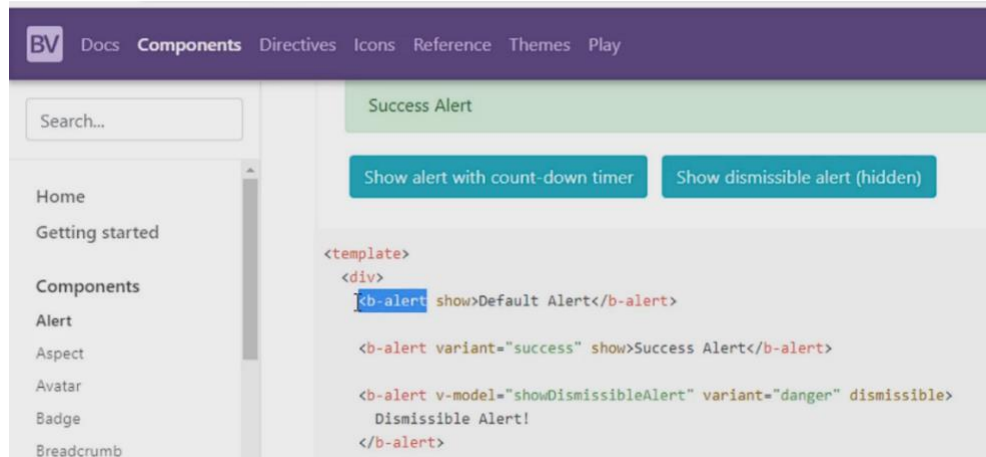
Por último, utilizamos Bootstrap copiando: `Vue.use(BootstrapVue) Vue.use(IconsPlugin)`.
Esto para que Bootstrap pueda estar disponible en todo nuestro proyecto.

```
JS main.js M X
src > JS main.js > ...
1  import Vue from 'vue'
2  import App from './App.vue'
3  import router from './router'
4
5  import { BootstrapVue, IconsPlugin } from 'bootstrap-vue'
6  import 'bootstrap/dist/css/bootstrap.css'
7  import 'bootstrap-vue/dist/bootstrap-vue.css'
8
9  Vue.use(BootstrapVue);
10 Vue.use(IconsPlugin);
11
12
13 Vue.config.productionTip = false
14
15 new Vue({
16   router,
17   render: h => h(App)
18 }).$mount('#app')
```

Guardamos el archivo que modificamos, y ahora levantamos nuestro proyecto para comprobar que todo esté instalado correctamente. Escribimos el comando `npm run serve`, nos dirigiremos al navegador, y se puede ver que los estilos ya están cambiados.



En la página de **Bootstrap-Vue**, se visualizará en la pestaña **components** lo que podemos utilizar con las etiquetas de Bootstrap, ahorrándonos la acción de escribir muchas clases de Bootstrap, pues ya están todas configuradas.



EXERCISE 2: UTILIZANDO BOOTSTRAP VUE

En este ejercicio, utilizaremos **Bootstrap-Vue** y lo implementaremos en nuestro proyecto. Para ello, seguiremos trabajando con creado en el anterior Exercise: "Instalando Bootstrap-Vue".

En primer lugar, se creará un **navbar**, ya que es uno de los componentes más utilizados en los sitios web. Si levantamos nuestro proyecto con el comando **npm run serve**, presionamos enter, y nos dirigimos al navegador, podemos ver que tiene un **navbar** por defecto.

```
$ npm run serve  
  
> boots-vue@0.1.0 serve C:\Use  
-vue  
> vue-cli-service serve
```



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](#) [vuex](#) [eslint](#)

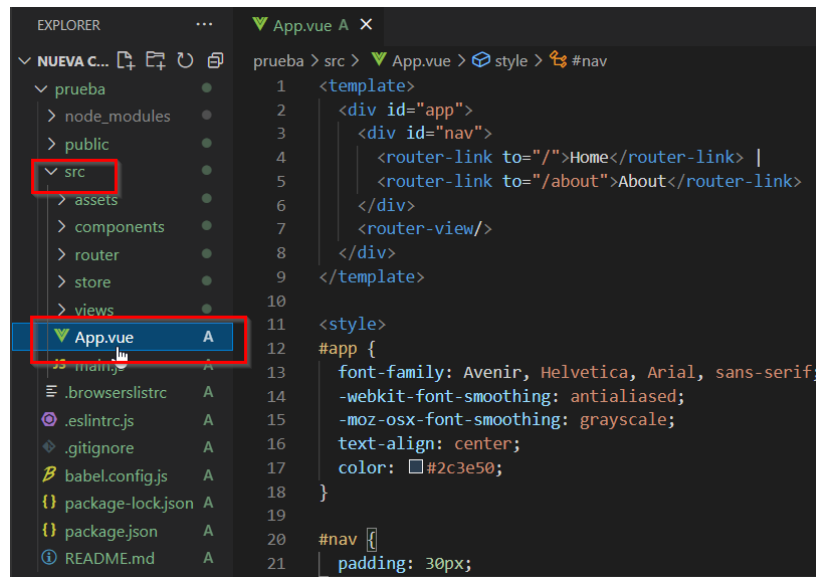
Essential Links

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

Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

Ahora, en el proyecto que estamos utilizando en Visual Studio Code, iremos al archivo **app.vue** que está dentro de la carpeta **src**, y veremos el código que viene por defecto para el **navbar** junto con sus estilos.

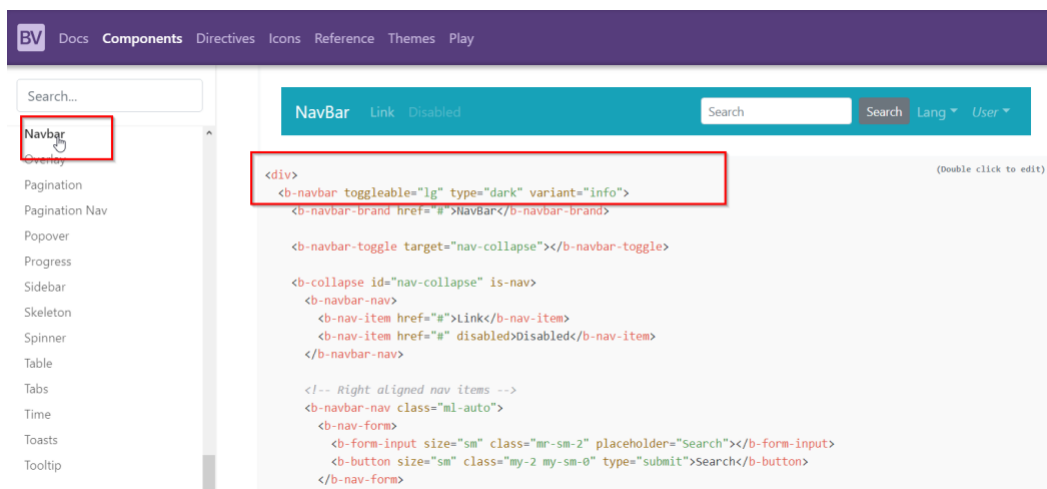


```
EXPLORER
└─ NUEVA C...
  └─ prueba
    └─ node_modules
      └─ public
        └─ src
          └─ assets
            └─ components
              └─ router
                └─ store
                  └─ views
                    └─ App.vue A
                      └─ main.js
                        └─ .browserslistrc
                          └─ .eslintrc.js
                            └─ .gitignore
                              └─ babel.config.js
                                └─ package-lock.json
                                  └─ package.json
                                    └─ README.md

App.vue A
prueba > src > App.vue > style > #nav
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>
10
11 <style>
12 #app {
13   font-family: Avenir, Helvetica, Arial, sans-serif;
14   -webkit-font-smoothing: antialiased;
15   -moz-osx-font-smoothing: grayscale;
16   text-align: center;
17   color: #2c3e50;
18 }
19
20 #nav {
21   padding: 30px;
```

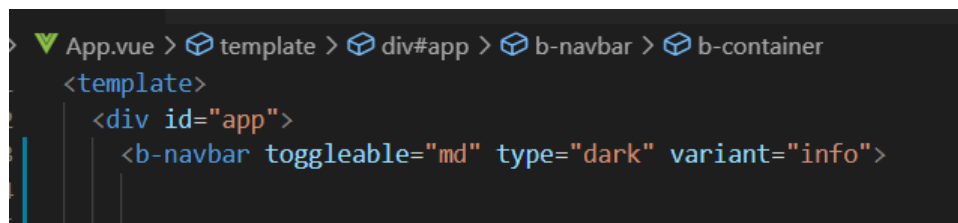
Como el **navbar** lo crearemos con **Bootstrap-Vue**, eliminaremos los estilos que vienen predeterminados. y también todo el **div** con **id= "nav"**.

Iremos a la página de **Bootstrap-Vue**, y buscamos el componente **navbar**. Aquí se puede revisar un ejemplo: lo primero que vemos es que en la etiqueta **<b-navbar>**, tenemos esta propiedad **toggleable = "lg"**.

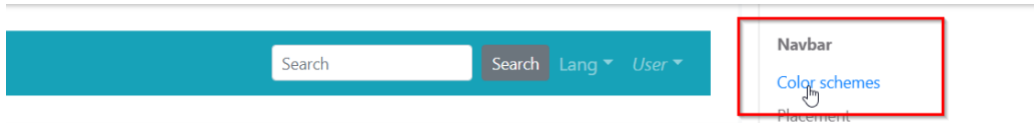


Copiaremos una parte del código, y lo pegaremos dentro del **div** con **id= "app"**; cambiaremos **toggleable = "lg"** por **"md"**, para que nuestro **navbar** se ajuste en dispositivos medianos y quede responsivo. La etiqueta **<b-navbar>** es reconocida por VUE, ésta viene con todas las configuraciones y clases en específico.

También podemos ver que tenemos **type= "dark"**, que hace que las letras queden en color blanco, además, está la variante **variant= "info"**, estos son los colores que vienen por defecto.



Y los podemos revisar en los colores (colores schema) de la página de **Bootstrap-Vue**, donde veremos: **primary, info, danger** entre otros. En este caso, utilizaremos dark para que veamos la diferencia; cambiamos info por dark,



Color schemes

`<b-navbar>` supports the standard Bootstrap v4 available background color variants. Set the variant prop to one of the following values to change the background color: primary, success, info, warning, danger, dark, or light.

```

App.vue M X
src > App.vue > ...
1 <template>
2   <div id="app">
3     <b-navbar toggleable="md" type="dark" variant="dark">
4
5
  
```

Ahora, crearemos una etiqueta container dentro de nuestra etiqueta `<b-navbar>`, y escribiremos `<b-container>`. Esta etiqueta nos generará un div con la clase container, y dentro de éstas, colocaremos todo lo referente al **navbar** para que se vea más ordenado.

```

1 <template>
2   <div id="app">
3     <b-navbar toggleable="md" type="dark" variant="dark">
4       <b-container>
5
6       </b-container>
7     </b-navbar>
8     <router-view/>
9   </div>
10 </template>
  
```


Si seguimos revisando el **navbar** ejemplo de la página de **Bootstrap-Vue**, del que nos estamos guiando, veremos la etiqueta **<b-navbar-brand>**; ésta nos permite agregar un logo que nos puede dirigir a alguna parte de la página con el href.

```
<div>
  <b-navbar toggleable="lg" type="dark" variant="info">
    <b-navbar-brand href="#">NavBar</b-navbar-brand>
```

Lo dejaremos como está, copiamos, y lo pegamos en nuestro código.

```
1 <template>
2   <div id="app">
3     <b-navbar toggleable="md" type="dark" variant="dark">
4       <b-container>
5         <b-navbar-brand href="#">NavBar</b-navbar-brand>
6       </b-container>
7     </b-navbar>
8   </div>
9   <router-view/>
10 </template>
```

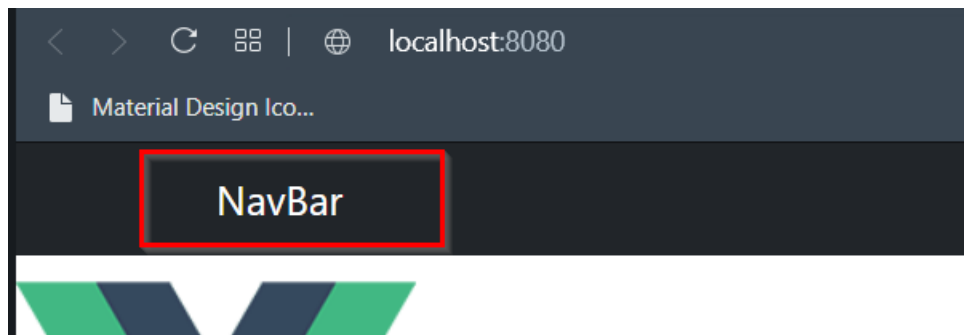
Después agregaremos la etiqueta que contendrá el botón **toggle <b-navbar-toggle>**, copiamos, y lo pegamos en nuestro código.

```
<div>
  <b-navbar toggleable="lg" type="dark" variant="info">
    <b-navbar-brand href="#">NavBar</b-navbar-brand>

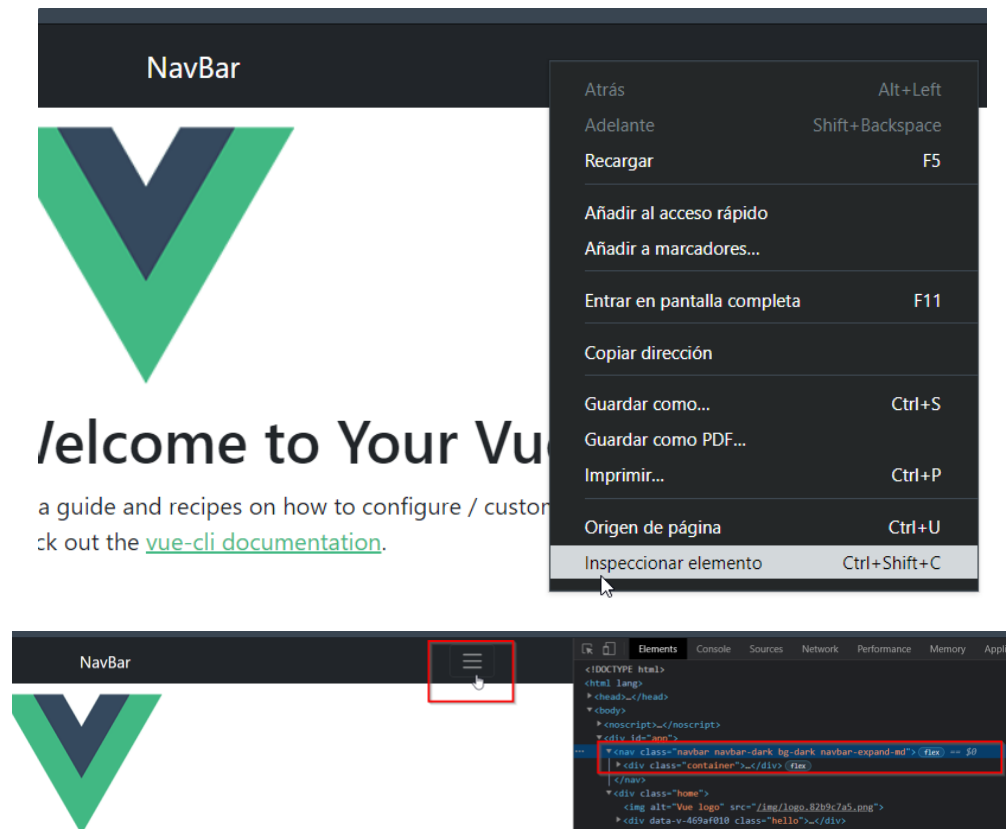
    <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>
```

```
1 <template>
2   <div id="app">
3
4     <b-navbar toggleable="md" type="dark" variant="dark">
5       <b-container>
6         <b-navbar-brand href="#">NavBar</b-navbar-brand>
7         <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>
8
9       </b-container>
10    </b-navbar>
11
12    <router-view/>
13  </div>
14</template>
```

Si visualizamos nuestro sitio, veremos el **navbar** con el container, si no apareciera, estaría la palabra **NavBar**, cercana al extremo izquierdo de la página.



Si inspeccionamos, podemos ver que efectivamente tenemos el botón que es responsivo, y en el **html**, estarán todas las clases que se utilizaban con **Bootstrap**; con esto nos damos cuenta que al utilizar **Bootstrap** con **Vue**, nos ahorramos muchas líneas de código.



Si seguimos revisando los componentes del **navbar**, también está el elemento **<b-collapsed>**. ¿Qué es este elemento? Son todos aquellos que van a estar colapsados, es decir, todo lo que esté dentro estará oculto en dispositivos medianos (md); esta etiqueta va a recibir propiedades como el **Id = "nav-collapse"**, y éste lo tendremos relacionado en el botón **<b-navbar-toggle>** en el **target= "nav-collapse"**, por lo que cada vez que este botón se active, se abrirá todo lo que esté dentro de los ítems (**b-nav-item**).

```
<div>
  <b-navbar toggleable="lg" type="dark" variant="info">
    <b-navbar-brand href="#">NavBar</b-navbar-brand>

    <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>
    <b-collapse id="nav-collapse" is-nav>
      <b-navbar-nav>
        <b-nav-item href="#">Link</b-nav-item>
        <b-nav-item href="#" disabled>Disabled</b-nav-item>
      </b-navbar-nav>
    </b-collapse>
  </b-navbar>
</div>
```

Copiamos, y lo pegamos en nuestro código; ordenamos y agregamos el cierre de la etiqueta: **</b-collapse>**.

```
1 <template>
2   <div id="app">
3     <b-navbar toggleable="md" type="dark" variant="dark">
4       <b-container>
5         <b-navbar-brand href="#">NavBar</b-navbar-brand>
6         <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>
7         <b-collapse id="nav-collapse" is-nav>
8           <b-navbar-nav>
9             <b-nav-item href="#">Link</b-nav-item>
10            <b-nav-item href="#" disabled>Disabled</b-nav-item>
11          </b-navbar-nav>
12        </b-collapse>
13      </b-container>
14    </b-navbar>
15    <router-view />
16  </div>
17 </template>
```

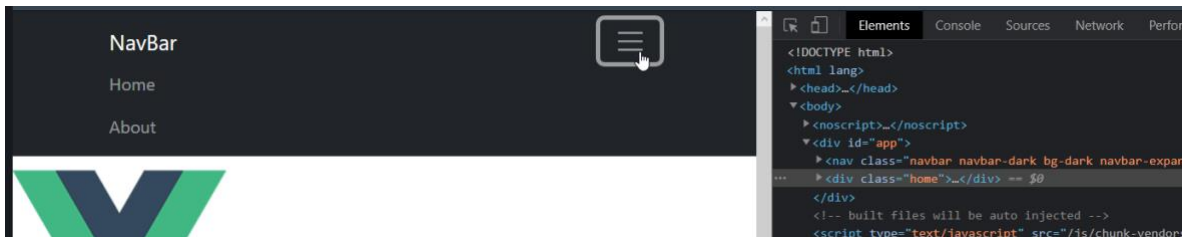
Cambiamos Link por home, y disable por About. También eliminaremos el disabled de la etiqueta.

```

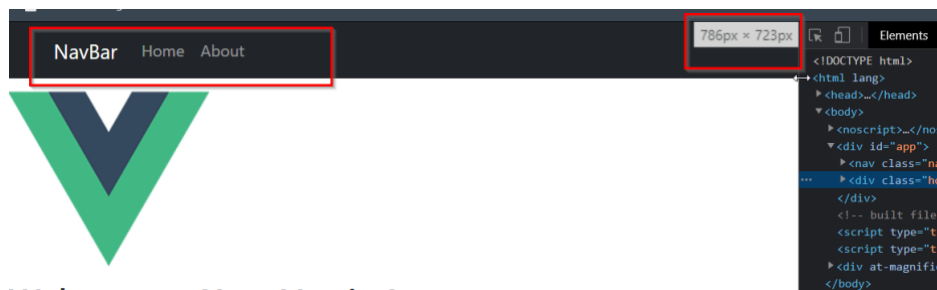
1 <template>
2   <div id="app">
3     <b-navbar toggleable="md" type="dark" variant="dark">
4       <b-container>
5         <b-navbar-brand href="#">NavBar</b-navbar-brand>
6         <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>
7         <b-collapse id="nav-collapse" is-nav>
8           <b-navbar-nav>
9             <b-nav-item href="#">Home</b-nav-item>
10            <b-nav-item href="#">About</b-nav-item>
11          </b-navbar-nav>
12        </b-collapse>
13      </b-container>
14    </b-navbar>
15    <router-view />
16  </div>
17</template>

```

Guardamos todos los cambios, vamos al navegador, y visualizamos que efectivamente todo está funcionando.



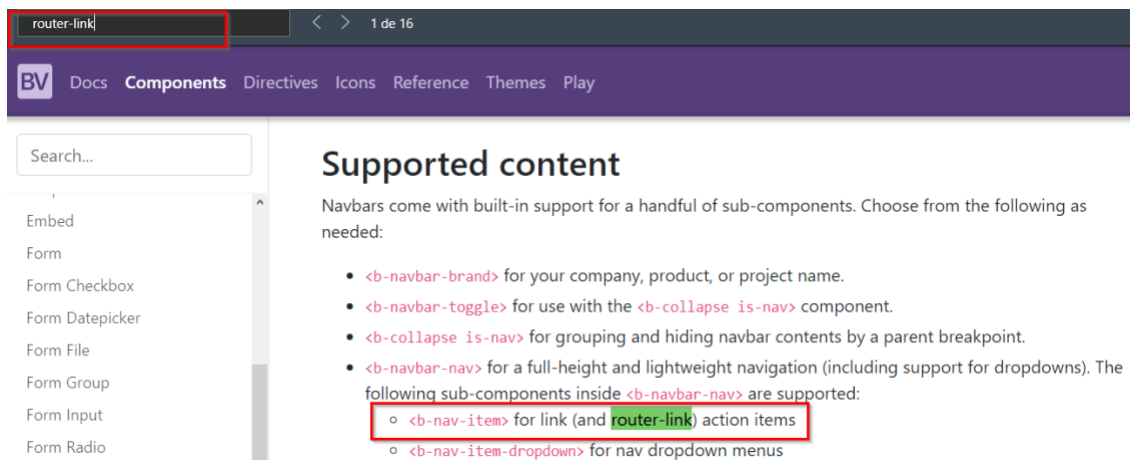
Si vamos agrandando nuestra página, notaremos que cuando sobrepasa a una pantalla de tamaño md, aparecen los botones.



En **Vue**, lo que se utilizaba en reemplazo de **href** era el **router-link**, pero aquí tenemos un **<b-nav-item>**.

```
<b-collapse id="nav-collapse" is-nav>
  <b-navbar-nav>
    <b-nav-item href="#">link</b-nav-item>
    <b-nav-item href="#" disabled>Disabled</b-nav-item>
  </b-navbar-nav>
```

Si buscamos en la documentación **"router link"**, veremos que la etiqueta **b-nav-item** efectivamente tiene configurado el **router-link**.



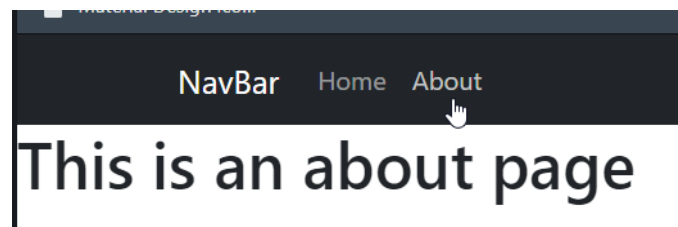
The screenshot shows the BootstrapVue documentation page. The search bar at the top contains 'router-link'. The left sidebar lists various components like 'Form', 'Form Input', etc. The main content area is titled 'Supported content' and lists sub-components for navbars. A red box highlights the following sub-components inside **<b-navbar-nav>** are supported:

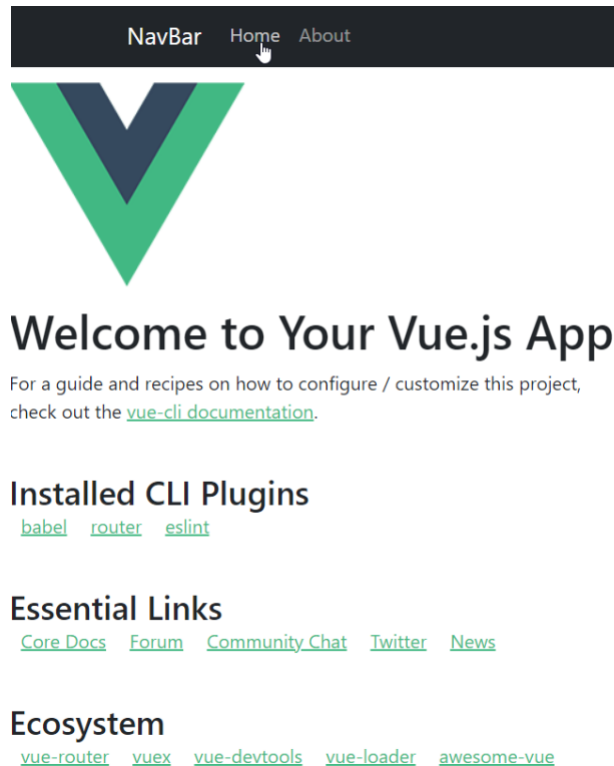
- **<b-nav-item>** for link (and **router-link**) action items
- **<b-nav-item-dropdown>** for nav dropdown menus

Por lo tanto, lo vamos a reemplazar, borramos el **href**, y agregamos: **:to= "{name:'Home' }"**. Copiamos, y haremos lo mismo con about,

```
1 <template>
2   <div id="app">
3     <b-navbar toggleable="md" type="dark" variant="dark">
4       <b-container>
5         <b-navbar-brand href="#">NavBar</b-navbar-brand>
6         <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>
7         <b-collapse id="nav-collapse" is-nav>
8           <b-navbar-nav>
9             <b-nav-item :to="{ name: 'Home' }">Home</b-nav-item>
10            <b-nav-item :to="{ name: 'About' }">About</b-nav-item>
11          </b-navbar-nav>
12        </b-collapse>
13      </b-container>
14    </b-navbar>
15    <router-view/>
16  </div>
17 </template>
```

Guardamos todos los cambios y visualizamos, presionamos about, y nos daremos cuenta de que todo funciona correctamente, y los botones dirigen a la vista correspondiente.





Con todo lo que hicimos, logramos tener un menú responsivo para nuestro sitio web, utilizando **Bootstrap-Vue**.

EXERCISE 3: INSTALANDO VUETIFY

En este ejercicio, veremos **Vuetify**. Éste es un framework que combina **VueJs**, con la estética de **Material Design**. Permite acelerar el desarrollo de aplicaciones web complejas, incorporando una gran cantidad de componentes "listos para usar".

Vuetify se basa en el sistema tipo "grid", para ordenar los layout de las páginas. Dispone de una enorme librería de componentes, que incluye desde los elementos de formulario sencillos, como: **botones, combobox, inputs, sliders**; y puede pasar a componentes más avanzados, típicos en aplicaciones Android, como: **"cards" o "snackbars"**.

Para empezar a utilizarlo, primero crearemos un proyecto con **vue cli**, iremos a Visual Studio Code, y en la terminal escribiremos **vue create** y el nombre del proyecto, en este caso, le pondremos **vue-vuetify** y presionamos enter, escogeremos la configuración manual para definir todos los elementos que necesitamos.

```
$ vue create vue-vuetify

Vue CLI v4.5.11

New version available 4.5.11 → 4.5.13
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
```

Tendremos: **babel, router, vuex y linter**. Presionamos enter.

```
? 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
```

Escogemos la versión estable de **Vue**, que es **2.x**, y presionamos enter. A continuación, escribimos **"Y"**, o sólo presionamos enter, ya que por defecto, se tomará la opción que esté en mayúscula. En la siguiente, dejamos la opción que viene por defecto, y nuevamente enter.

```
? 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
```

En todas las siguientes configuraciones, presionaremos enter, pues dejaremos los que vienen por defecto.

```
? Pick a linter / formatter config: (Use arrow keys)
> ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier
```

```
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)
>(*) Lint on save
( ) Lint and fix on commit
```

```
? 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 generado el proyecto con todas las opciones escogidas, ingresamos a éste con el comando `cd nombre proyecto`; en este caso `cd vue-vuetify`, presionamos enter.

```
$ cd vue-vuetify
```

Levantaremos nuestro proyecto con el comando `npm run serve`, para comprobar que esté todo bien, ingresamos a la URL, y veremos en nuestro navegador que se ha instalado correctamente.

```
$ npm run serve

> vue-vuetify@0.1.0 serve C:
> vue-cli-service serve

INFO Starting development
```



localhost:8080

[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](#) [vuex](#) [eslint](#)

Essential Links

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

Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

Ahora, nos dirigiremos a la página oficial de **Vuetify**: <https://vuetifyjs.com/en/>, vamos a **get started**, y seguiremos las instrucciones.

vuetifyjs.com/en/

Search Vuetify (press /)

[Learn](#) [Support](#)



Material Design Framework

Vuetify is a Vue UI Library with beautifully handcrafted Material Components. No design skills required — everything you need to create amazing applications is at your fingertips.

[GET STARTED](#)

[WHY VUETIFY?](#)

[GITHUB](#)

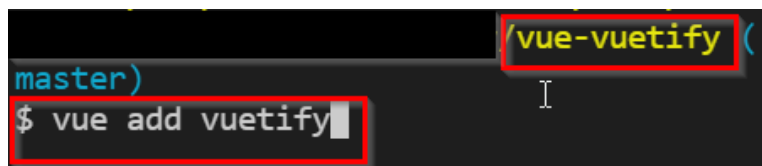
Como se puede observar, en nuestro proyecto tenemos que escribir o copiar este comando: `vue add vuetify` en nuestra terminal, y presionamos enter.

If you have not already created a new Vue.js project using **Vue CLI**, you can do so by typing:

```
vue create my-app
# navigate to new project directory
cd my-app
```

Now that you have an instantiated project, you can add the Vuetify [Vue CLI package](#) using the cli.

```
vue add vuetify
```



```
master)
$ vue add vuetify
```

Seleccionaremos la opción por defecto: **Default**.

```
📦 Installing vue-cli-plugin-vuetify...

+ vue-cli-plugin-vuetify@2.4.1
updated 1 package and audited 1406 packages in 8.849s

88 packages are looking for funding
  run `npm fund` for details

found 120 vulnerabilities (113 moderate, 7 high)
  run `npm audit fix` to fix them, or `npm audit` for details
✓ Successfully installed plugin: vue-cli-plugin-vuetify

? Choose a preset: (Use arrow keys)
> Default (recommended)
  Preview (Vuetify 3 + Vite)
  Prototype (rapid development)
  V3 (alpha)
  Configure (advanced)
```

Esperamos a que todo se instale.

```
? Choose a preset: Default (recommended)

🚀 Invoking generator for vue-cli-plugin-vuetify...
⚓ Running completion hooks...

✓ Successfully invoked generator for plugin: vue-cli-plugin-vuetify
vuetify Discord community: https://community.vuetifyjs.com
vuetify Github: https://github.com/vuetifyjs/vuetify
vuetify Support Vuetify: https://github.com/sponsors/johnleider
```

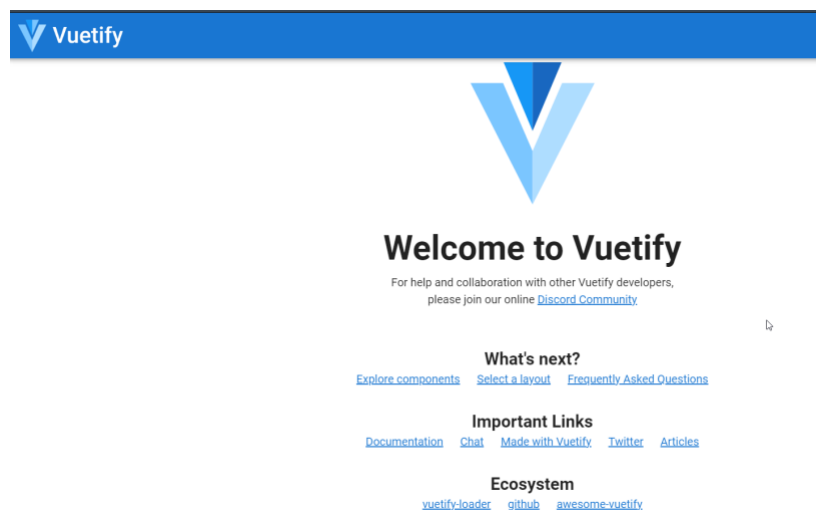
Y volvemos a levantar nuestro proyecto, con el comando: `npm run serve`. Presionamos enter.

```
$ npm run serve

> vue-vuetify@0.1.0 serve C:\Users\jocel\Documents\vue-vuetify
> vue-cli-service serve

INFO Starting development server...
40% building 21/32 modules 11 active ...ents\...
```

Nos dirigimos al navegador, y veremos que **Vuetify** ya está integrado.

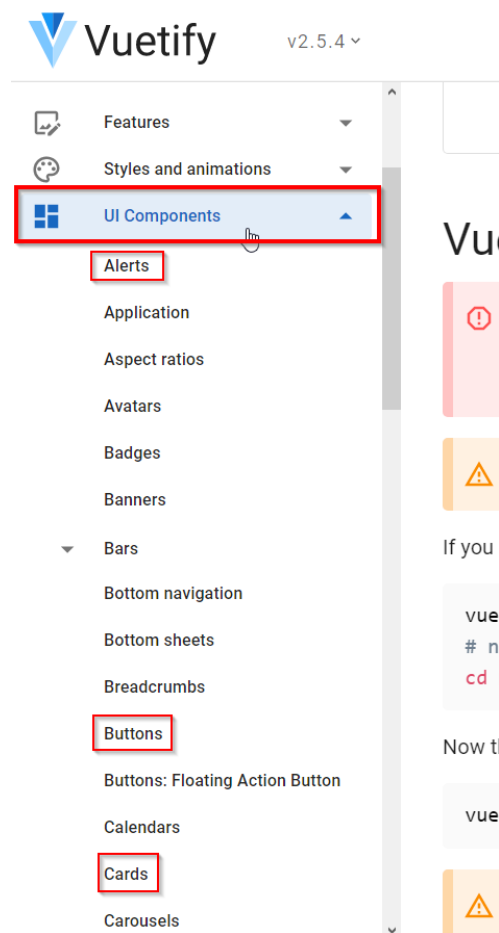


EXERCISE 4: UTILIZANDO VUETIFY

En este ejercicio, aprenderemos a utilizar **Vuetify**, y a reutilizar componentes. Esto nos permitirá agilizar la construcción de nuestro proyecto.

Veremos dos componentes que son necesarios en la mayoría de los proyectos, y que se pueden reutilizar. El primero es el **navbar** o barra de navegación, y el segundo es el **footer**. Seguiremos trabajando con el proyecto anterior "Instalando Vuetify".

Si vamos a la página de **Vuetify**, en la pestaña **UI Components**, veremos el listado de los componentes como: **los alerts, buttons, cards**, entre otros.



Si nos dirigimos a los **grids**, podemos ver que se manejan de una forma similar a Bootstrap, y que la estructura correcta a utilizar es: **v-container**, un **v-row** y un **v-col**.

Expansion panels

Footers

▼ Form inputs & controls

Grid system

▼ Groups

Hover

Icons

Images

Lazy


Lists






Menus

Navigation drawers

Overlays

Pagination

 Watch a free video on **Vuetify Grids**

| Material Design Breakpoints | | | |
|---|-----------|------------------------|--------------------|
| Device | Code | Type | Range |
|  Extra small | xs | Small to large phone | < 600px |
|  Small | sm | Small to medium tablet | 600px > < 960px |
|  Medium | md | Large tablet to laptop | 960px > < 1264px* |
|  Large | lg | Desktop | 1264px > < 1904px* |
|  Extra large | xl | 4k and ultra-wide | > 1904px* |

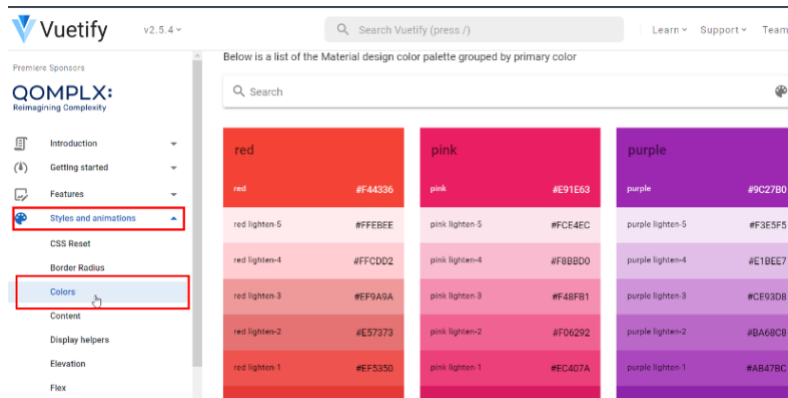
* -16px on desktop for browser scrollbar

API

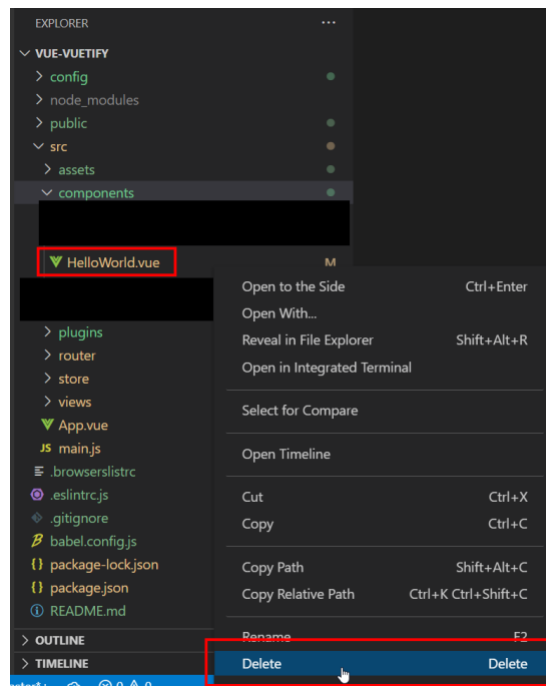
- **v-container** 
- **v-row** 
- **v-col** 
- **v-spacer** 

Sub-componentes

También podemos encontrar una gran gama de colores. Para esto nos dirigimos a **styles and animation**, ingresamos a colors, y veremos todas las opciones disponibles para elegir.



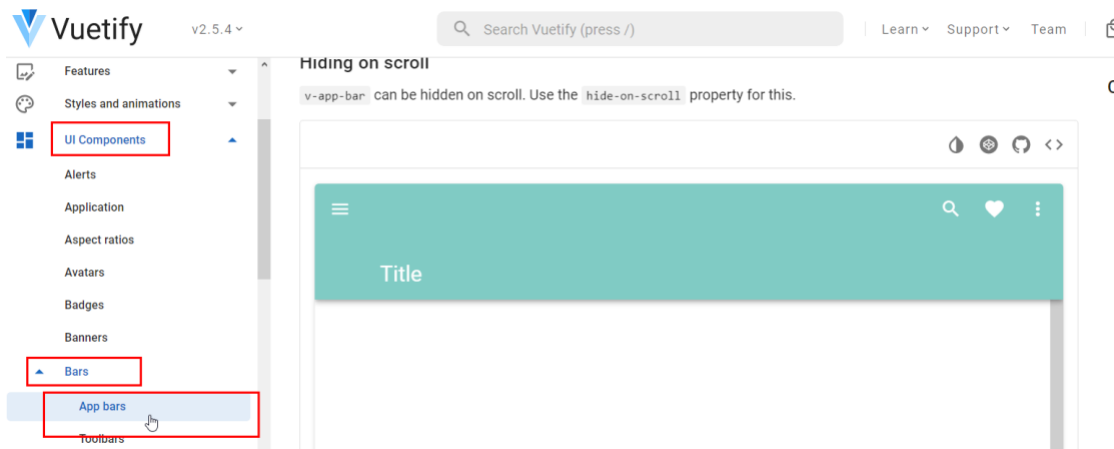
Ahora, nos dirigiremos a nuestro proyecto creado en el Exercise anterior, en Visual Studio Code. Comenzaremos eliminando el componente **HelloWorld.vue**, y en la carpeta **views** del archivo **Home.vue**, eliminaremos las referencias a éste.



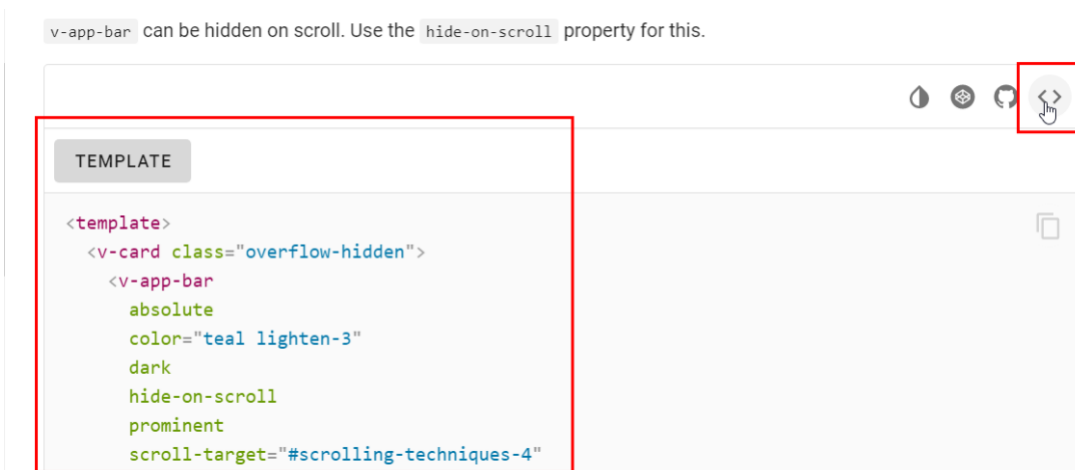
Crearemos un título dentro de etiquetas `<div>`, con un `h1` que diga: "Bienvenido al Home", y guardamos.

Ahora, crearemos dos componentes: uno llamado `NavBar.vue`, y el otro `Footer.vue`; en cada uno crearemos la estructura base con un `template`.

Si volvemos a la página de **Vuetify**, nos dirigimos a los componentes, y buscamos **bars** y luego **app bars**, podemos ver muchos ejemplos para nuestro **navbar**,



En esta pestaña se puede ver el código del **navbar** que escojamos.



Volvemos a Visual Studio Code, y empezaremos a crear un **navbar** básico.

El primer elemento será un `div`, que contendrá todo nuestro `navbar`. A continuación, le agregaremos `<v-app-bar>` y un color que escojamos `"teal darken-4"`, e indicaremos que el estilo será `"dark"`.

```

▼ Footer.vue U    ▼ NavBar.vue U X
src > components > ▼ NavBar.vue > {} "NavBar.vue" > script
1  <template>
2    <div>
3      <v-app-bar color="teal darken-4" dark>
4
5
6
7      </v-app-bar>
8    </div>
9  </template>

```

También utilizaremos un `container`, para seguir la estructura recomendada en la página de `Vuetify`, y además agregaremos un `v-row`.

Dentro, colocaremos dos botones del tipo `text`: uno para que nos dirija al home, y el otro que nos dirija a la página about.

```

1  <template>
2    <div>
3      <v-app-bar color="teal darken-4" dark>
4        <v-container>
5          <v-row>
6            <v-btn text> Home</v-btn>
7            <v-btn text> About</v-btn>
8          </v-row>
9        </v-container>
10     </v-app-bar>
11   </div>
12 </template>

```

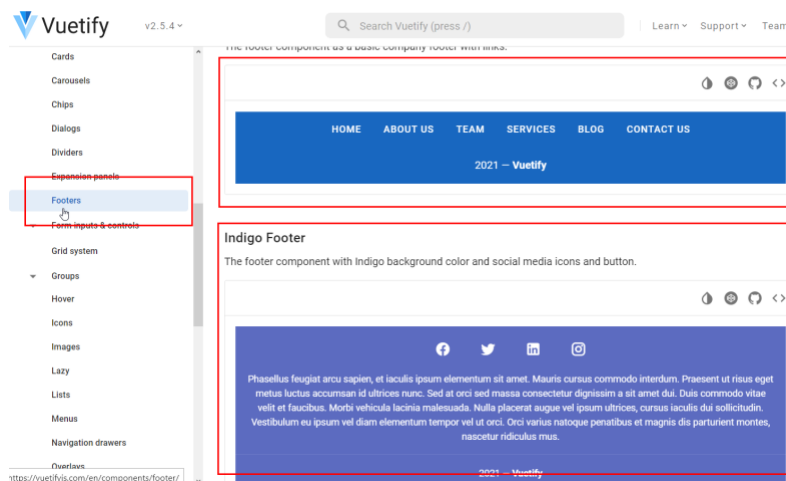
Lo haremos más amigable visualmente, agregando íconos.

Para agregarlos, usaremos las etiquetas `v-icon`, y a cada una de éstas, una clase `margin right 1 (mr-1)`; al home le incluiremos el ícono `mdi-home`, y al about un ícono `mdi-account-multiple`. También agregaremos la dirección a donde tienen que dirigirse, y para eso escribiremos `to= "/"` para el home, y `to= "/about"` para que nos dirija a la página about.

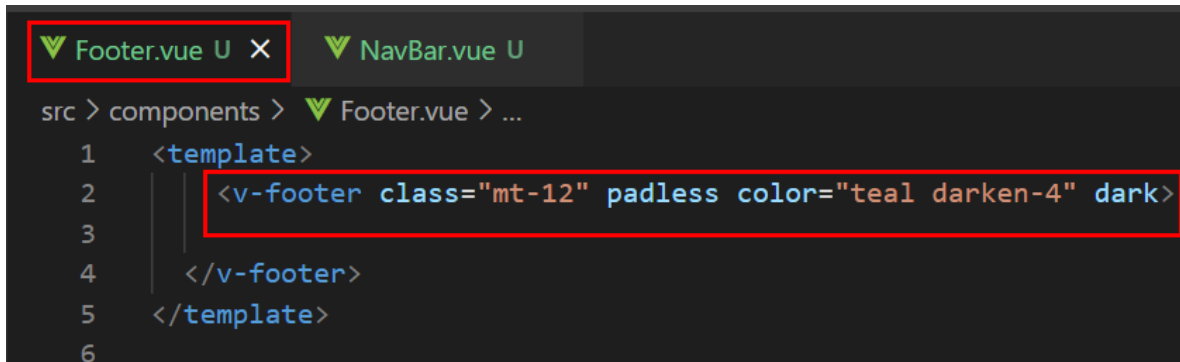
```

1 <template>
2   <div>
3     <v-app-bar color="teal darken-4" dark>
4       <v-container>
5         <v-row>
6           <v-btn text to="/">
7             <v-icon class="mr-1">mdi-home</v-icon>Home</v-btn>
8           <v-btn text to="/about">
9             <v-icon class="mr-1">mdi-account-multiple</v-icon>
10            About
11          </v-btn>
12        </v-row>
13      </v-container>
14    </v-app-bar>
15  </div>
16 </template>
  
```

Ahora iremos al componente `Footer.vue`, y aquí agregaremos uno de tipo básico. Si volvemos a la página de `Vuetify`, y buscamos footer, notaremos que existen varios estilos.



Volvemos a nuestro proyecto, y en el componente **footer**, lo primero que agregaremos será la etiqueta que lo identifica: **<v-footer>**, luego la clase **mt-12**, que quiere decir que le daremos un margin-top de 12, y le incluiremos algunas clases y un color, que será el mismo que le dimos al **navbar** (**teal darken-4**) y de estilo dark.

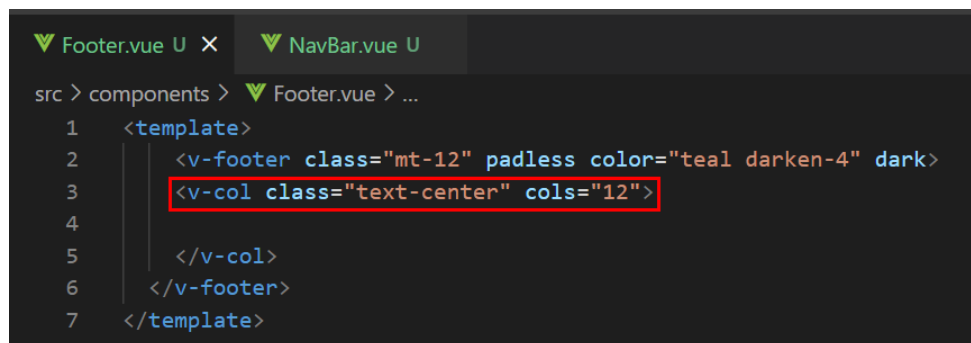


```

src > components > Footer.vue > ...
1  <template>
2    <v-footer class="mt-12" padless color="teal darken-4" dark>
3
4    </v-footer>
5  </template>
6

```

Después agregaremos la etiqueta **v-col**, y la clase **text center**, para que el texto quede centrado, también columnas de 12, para que se vea en las 12 columnas de la pantalla.

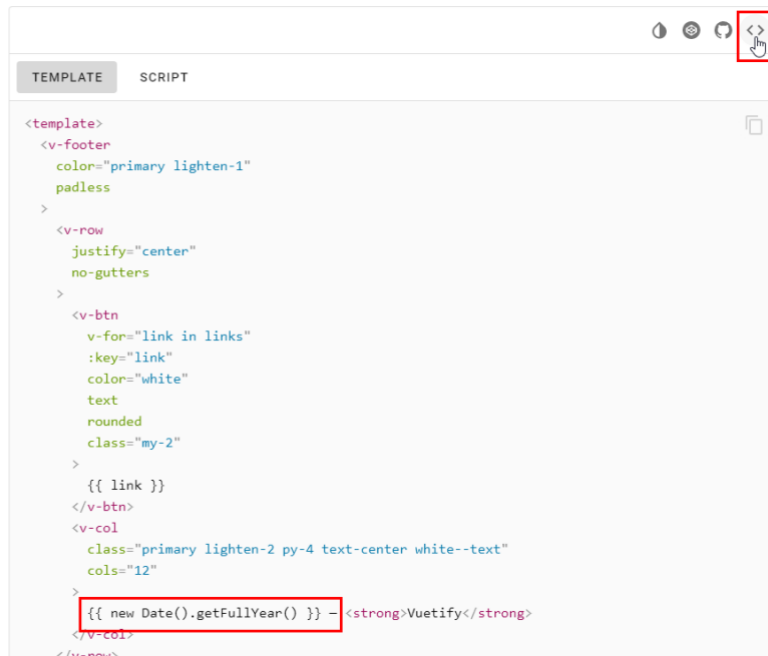


```

src > components > Footer.vue > ...
1  <template>
2    <v-footer class="mt-12" padless color="teal darken-4" dark>
3    <v-col class="text-center" cols="12">
4
5    </v-col>
6    </v-footer>
7  </template>
8

```

Y por último, agregaremos un texto con las etiquetas ****, en este caso pondremos: "Bienvenido a Vuetify", y a continuación, colocaremos una función que muestre el año en el que estamos, ésta se puede encontrar en los componentes footer de la página de Vuetify, copiamos y lo pegamos.



```

1 <template>
2   <v-footer class="mt-12" padless color="teal darken-4" dark>
3     <v-col class="text-center" cols="12">
4       <strong>Bienvenido a vuetify</strong>
5       {{new Date().getFullYear()}} -
6     </v-col>
7   </v-footer>
8 </template>

```

Una vez terminado de modificar el **footer**, guardamos, nos dirigimos a **app.vue**, y eliminaremos todo lo que está dentro de las etiquetas **<v-app-bar>**. Ahora, importaremos nuestros dos componentes en el **script**, escribimos: **import NavBarComp from './components/NavBar'**, y haremos lo mismo con el **footer**. Después, agregaremos el objeto **components** en **export default**, y llamaremos a nuestros componentes.

Por último, llamaremos a nuestros componentes dentro de la etiqueta **<v-app>**; de esta manera, estamos reutilizando los dos componentes creados, llamándolos en el componente principal **app.vue**.

El orden en que llamemos a nuestros componentes es importante, porque es como se visualizará. Por ejemplo, los **navbar** tienen que ir primero, luego colocaremos el componente como etiqueta, entremedio irá **router-view**, recordemos que esta etiqueta nos permite que se observen todas nuestras vistas, y por último irá el **footer**.

```
1 <template>
2   <v-app>
3     <navBarComp></navBarComp>
4     <v-main>
5       <router-view/>
6     </v-main>
7     <footer-comp></footer-comp>
8   </v-app>
9 </template>
10 <script>
11 import navBarComp from './components/NavBar.vue'
12 import footerComp from './components/Footer.vue'
13 export default {
14   name: 'App',
15   components: {
16     navBarComp, footerComp
17   }
18 }
19 </script>
```

Guardamos los cambios que hemos hecho, y levantamos el proyecto con el comando: **npm run serve**. Nos dirigimos al navegador, y aquí podemos ver nuestros componentes **navbar** y **footer**.

```
$ npm run serve

> vue-vuetify@0.1.0 serve C:\Users\
> vue-cli-service serve

INFO Starting development server.
40% building 22/25 modules 3 active
```

[HOME](#) [ABOUT](#)

Bienvenido al Home

Bienvenido a vuetify 2021 -

← → ↻ ⓘ localhost:8081/about

[HOME](#) [ABOUT](#)

This is an about page

Bienvenido a vuetify 2021 -

De esta manera, aprendimos a reutilizar dos componentes en una vista.