

Curso Técnico Superior Profissional em Tecnologias e Programação de Sistemas de Informação

Unidade Curricular: Desenvolvimento Web – Front-End

1.º Ano/2.º Semestre

Docente: Marco Miguel Olival Olim

Data 12/04/2019

ESTE EXERCÍCIO ABORDA A UTILIZAÇÃO DO VUE CLI COM O VUETIFY

Depois de instalar o Vue CLI e gerar um projeto (com ou sem Vuetify) fica disponível uma estrutura de ficheiros a partir de **src**, já com um conjunto de componentes (“components”), páginas (em “views”, semelhante às “pages” do Nuxt), rotas de navegação (“router.js”) e gestão do estado da aplicação (“store.js”).



```
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>
```

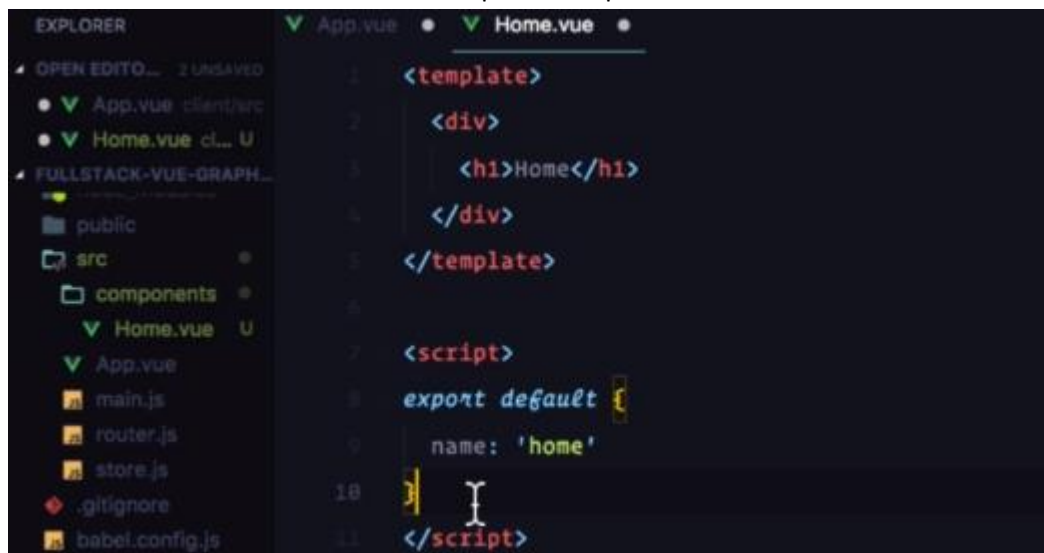
Esta estrutura inicial apenas serve para orientação. Vamos, por isso, remover estas **views**, os **assets**, o componente **Helloworld** e toda a estrutura desnecessária da página inicial do projeto, a **App.vue**



```
1 <template>
2   <div>
3     <h1>App</h1>
4     <router-view/>
5   </div>
6 </template>
7
8
```

Cofinanciado por:

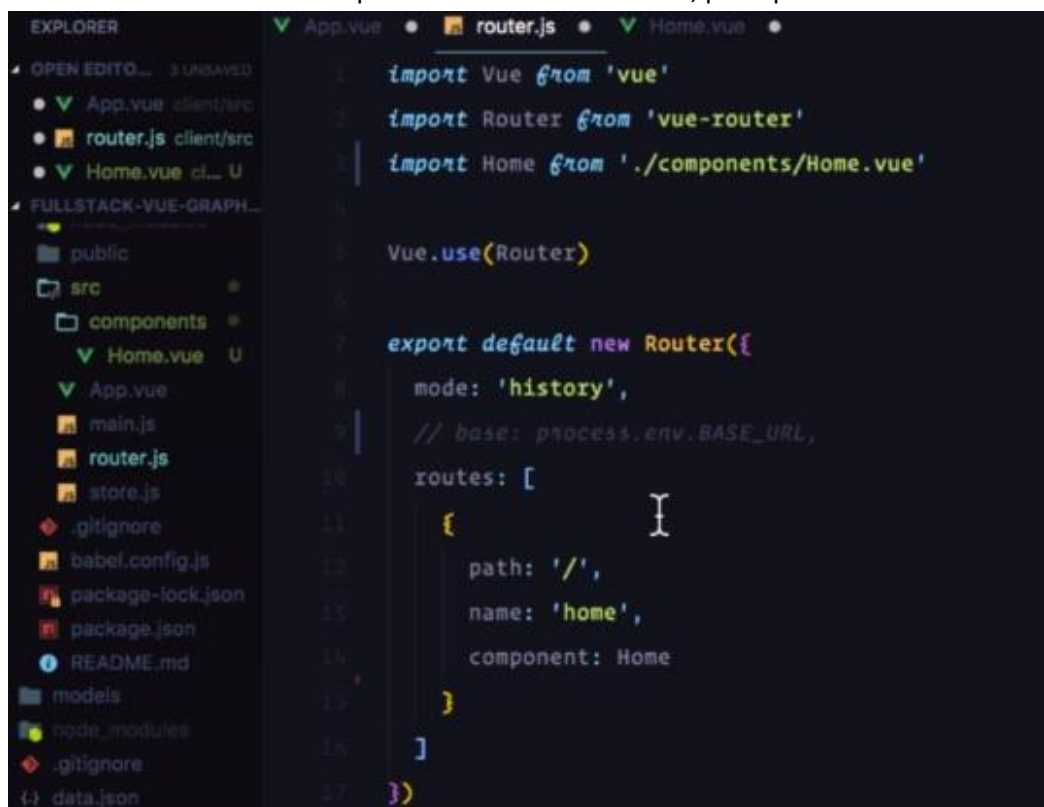
Movemos também a “view” **Home.vue** para “components” e removemos também a informação redundante



The screenshot shows the VS Code editor with the **Home.vue** file open. The Explorer sidebar on the left shows the project structure with **components/Home.vue** selected. The code in the editor is as follows:

```
1 <template>
2   <div>
3     <h1>Home</h1>
4   </div>
5 </template>
6
7 <script>
8   export default {
9     name: 'home'
10  }
11 </script>
```

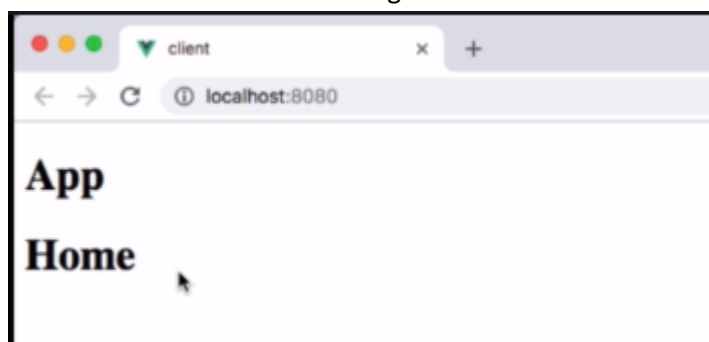
As rotas não são definidas implicitamente como no NUXT, pelo que temos de editar o ficheiro **router.js**



The screenshot shows the VS Code editor with the **router.js** file open. The Explorer sidebar on the left shows the project structure with **router.js** selected. The code in the editor is as follows:

```
1 import Vue from 'vue'
2 import Router from 'vue-router'
3 import Home from './components/Home.vue'
4
5 Vue.use(Router)
6
7 export default new Router({
8   mode: 'history',
9   // base: process.env.BASE_URL,
10  routes: [
11    {
12      path: '/',
13      name: 'home',
14      component: Home
15    }
16  ]
17 })
```

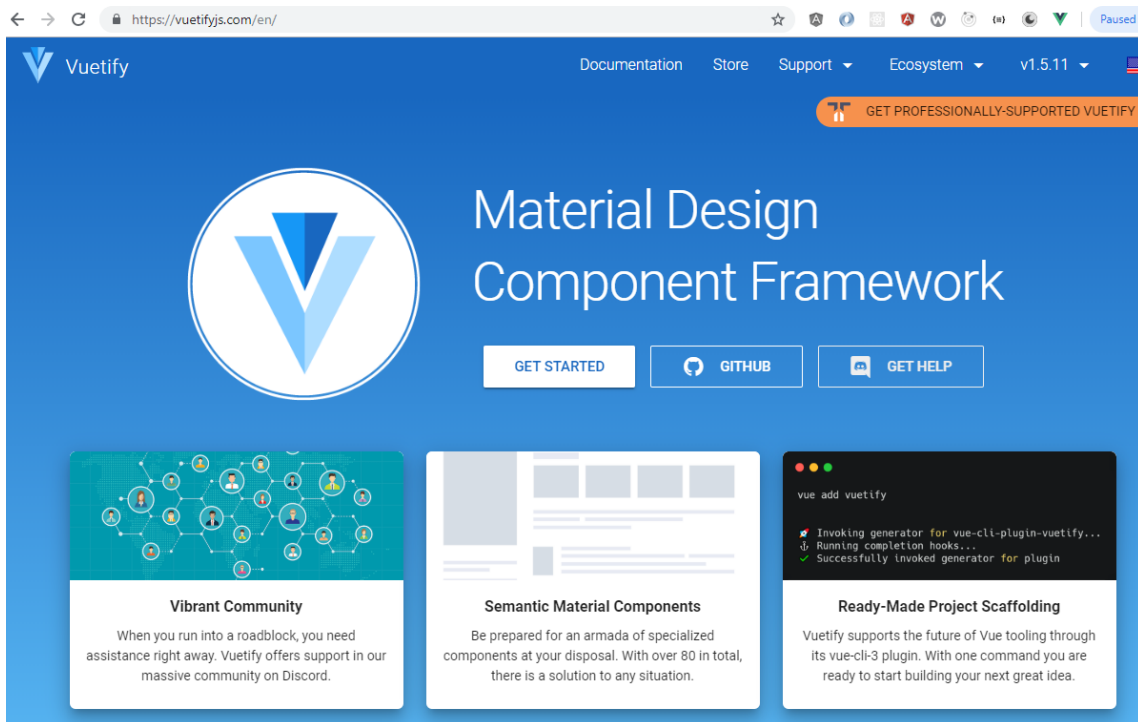
Neste momento deverá ter o seguinte resultado



Cofinanciado por:



Vamos agora incluir, na nossa aplicação, componentes da framework de User Interface: **Vuetify**



ATENÇÃO: Se ainda **não** instalou o Vuetify com a UI após a realização da ficha anterior, também poderá fazê-lo na linha de comando com **vue add vuetify**

```
OUTPUT  DEBUG CONSOLE  TERMINAL  PROBLEMS  2: node

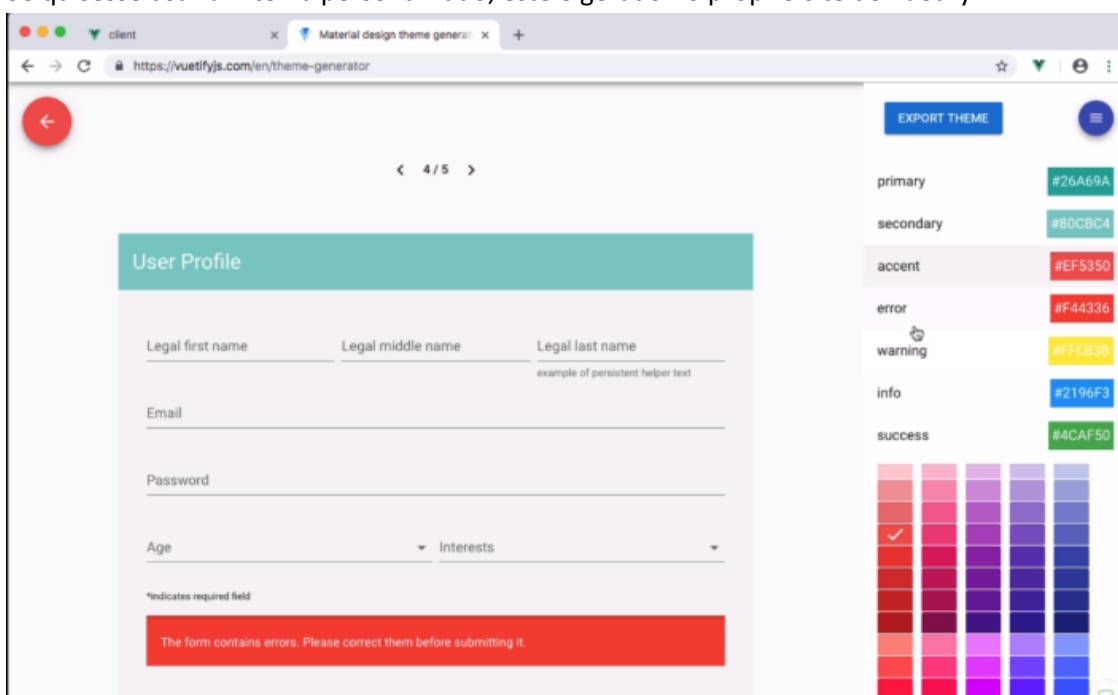
Installing vue-cli-plugin-vuetify...

+ vue-cli-plugin-vuetify@0.1.6
removed 1 package, updated 1 package and audited 11248 packages in 27.47s
found 0 vulnerabilities

✓ Successfully installed plugin: vue-cli-plugin-vuetify

? Use a pre-made template? (will replace App.vue and HelloWorld.vue) No
? Use custom theme? No
? Use a-la-carte components? No
? Use babel/polyfill? (Y/n)
```

Se quisesse usar um tema personalizado, este é gerado no próprio site do vuetify



Cofinanciado por:



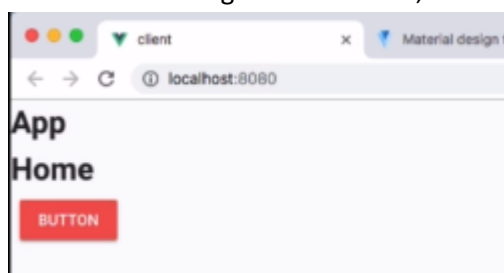
Vamos modificar o componente Home.vue acrescentando um botão que utiliza a sintaxe dos componentes do Vuetify.

```
Home.vue  vuetify.js  App.vue

<template>
  <div>
    <h1>Home</h1>
    <v-btn color="accent">Button</v-btn>
  </div>
</template>

<script>
export default {
  name: "home"
};
</script>
```

Deverá obter o seguinte resultado, com um botão típico do Material Design



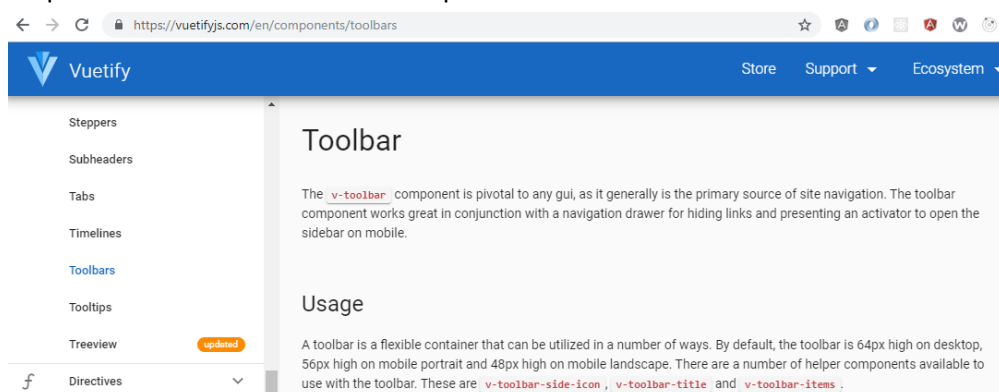
Para inicializar a aplicação com o vuetify, com o layout adequado, devemos utilizar a tag **v-app** para encapsular todos os componentes destes. Para tal vamos editar o **app.vue** e acrescentar uma **toolbar**

```
EXPLORER  Home.vue  App.vue

OPEN EDITOR... 1 UNCHANGED
  Home.vue client/...
  App.vue client/src
FULLSTACK-VUE-GRAPH...
client
  node_modules
  public
  src
    components
    Home.vue
    plugins
    vuetify.js
    App.vue
    main.js
    router.js
    store.js
    .gitignore
    babel.config.js

1 <template>
2   <v-app>
3     <!-- Horizontal Navbar -->
4     <v-toolbar fixed color="primary" dark>
5       |
6     </v-toolbar>
7
8     <!-- App Content -->
9     <main>
10      <router-view/>
11    </main>
12  </v-app>
13</template>
```

Os parâmetros a atribuir a cada componente está bem discriminado na documentação



Cofinanciado por:



No caso da toolbar são diversos os parâmetros à disposição. Neste exemplo usamos **fixed**, **color** e **dark**.

Name	Default	Type
color	undefined	string
Applies specified color to the control - it can be the name of material color (for example <code>success</code> or <code>purple</code>) or css color (<code>#033</code> or <code>rgba(255, 0, 0, 0.5)</code>)		
dark	false	boolean
Applies the dark theme variant		
dense	false	boolean
Reduces the height of the toolbar content and extension		
extended	false	boolean
Force the toolbar to generate the extension without using the slot		
extension-height	undefined	number string
Specify an explicit height for the extended slot		
fixed	false	boolean
Position the element fixed		

No caso do vuetify os elementos a inserir são representados sempre por tags em vez de classes. Neste caso inserimos um hambúrguer e um título

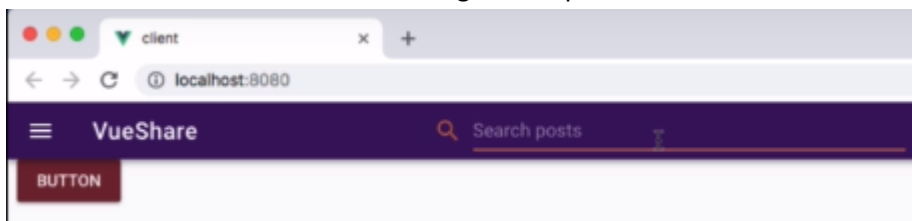
```
1 <template>
2   <v-app>
3     <!-- Horizontal Navbar -->
4     <v-toolbar fixed color="primary" dark>
5       <v-toolbar-side-icon></v-toolbar-side-icon>
6       <v-toolbar-title>
7         <router-link to="/" tag="span" style="cursor: pointer">
8           VueShare
9         </router-link>
10      </v-toolbar-title>
11    </v-toolbar>
12
13    <!-- App Content -->
14    <main>
15      <router-view/>
16    </main>
17  </v-app>
18 </template>
```

Ao lado do título será colocada uma caixa de pesquisa

```
</v-toolbar-title>
<v-spacer></v-spacer>
<!-- Search Input -->
<v-text-field flex prepend-icon="search" placeholder="Search posts" color="accent" single-line
hide-details></v-text-field>
</v-toolbar>
```

Cofinanciado por:

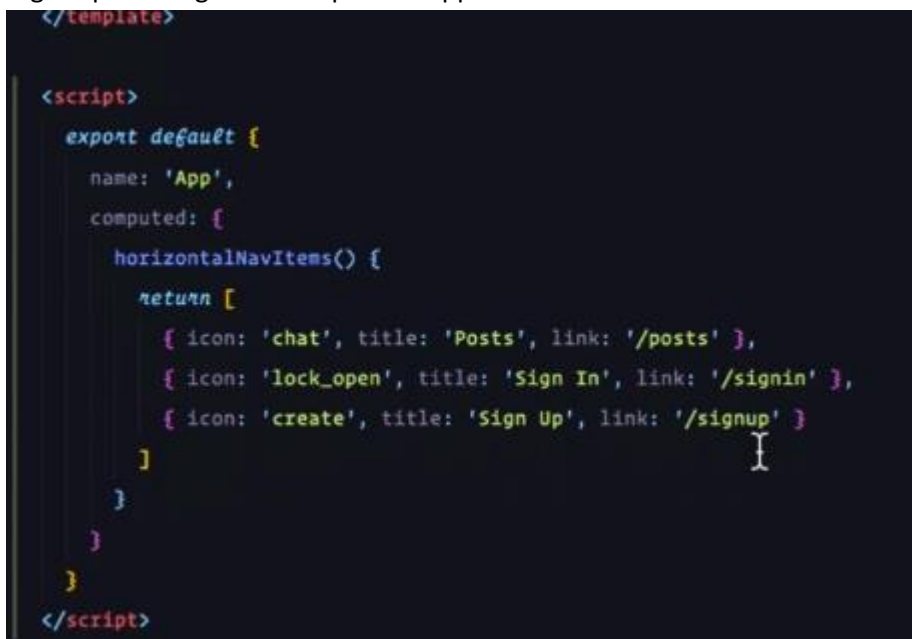
Neste momento o site deverá ter o seguinte aspeto



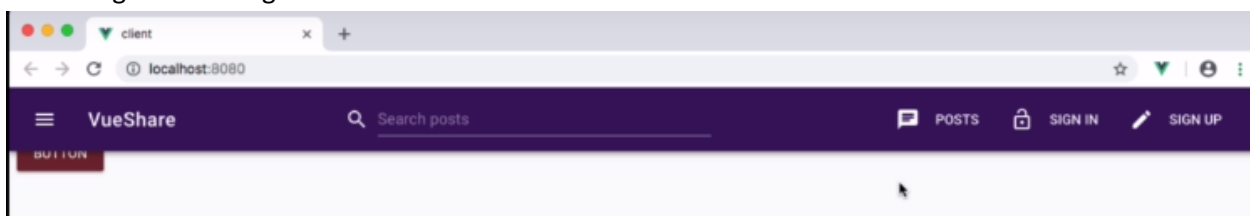
Abaixo da caixa de pesquisa adicionamos novo spacer de forma a colocar um conjunto de botões



Para disponibilizar esta lista de botões, o vuetify utiliza a seguinte sintaxe em JSON que temos de adicionar à tag script do single file component App.vue



Deverá agora ter o seguinte resultado:



Cofinanciado por:



No entanto, a barra de navegação está a ficar em cima do botão. Para corrigir, adicionamos um container com uma margem de topo

```
<main>
  <v-container class="mt-4">
    <router-view/>
  </v-container>
</main>
</v-app>
</template>
```

Também o site ainda não está muito responsivo, pelo que devemos adicionar classes, como no bootstrap, para esconder elementos em resoluções apropriadas para dispositivos móveis

```
<v-toolbar-items class="hidden-xs-only">
  <v-btn flat v-for="item in horizontalNavItems" :key="item.title" :to="item.link">
    <v-icon class="hidden-sm-only" left>{{item.icon}}</v-icon>
    {{item.title}}
  </v-btn>
</v-toolbar-items>
</v-toolbar>
```

Em dispositivos móveis costuma ser utilizada uma side nav. Acrescentamos então essa funcionalidade à nossa aplicação com o componente v-navigation-drawer

```
<template>
  <v-app>
    <!-- Side Navbar -->
    <v-navigation-drawer app temporary fixed v-model="sideNav">
      |
    </v-navigation-drawer>

    <!-- Horizontal Navbar -->
    <v-toolbar fixed color="primary" dark>
```

Para alternar o estado de visibilidade da sidenav acrescentamos uma propriedade booleana

```
<script>
export default {
  name: "App",
  data() {
    return {
      sideNav: false
    }
  },
  computed: {
    horizontalNavItems() {
      return [
        { icon: "chat", title: "Posts", link: "/posts" },
      ]
    }
  }
}
```

Cofinanciado por:

Para alterar o estado da propriedade sidenav usamos então um método para o efeito

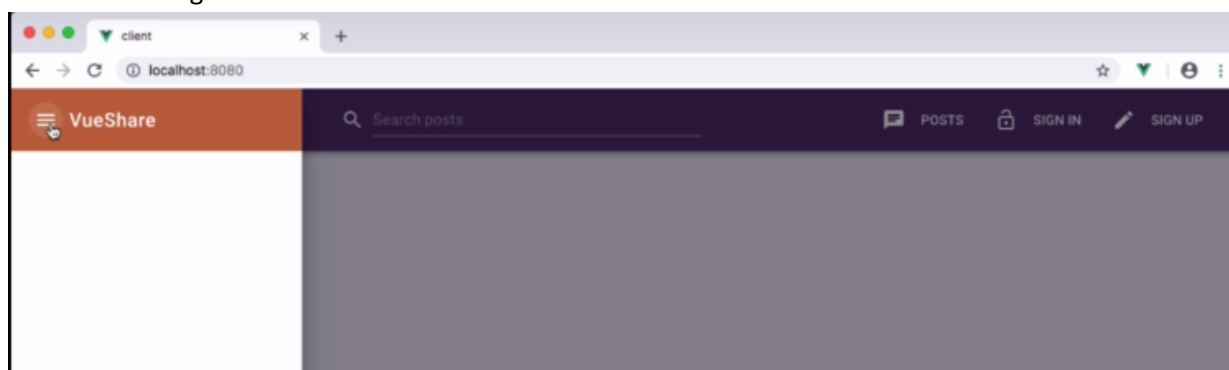
```
    { icon: "create", title: "Sign Up", link: "/signup" }
  ];
},
methods: {
  toggleSideNav() {
    this.sidenav = !this.sidenav;
  }
}
};
</script>
```

Acrescentamos agora o método ao hambúrguer e ligamos à sidenav com v-model

```
<template>
  <v-app style="background: #E3E3EE">
    <!-- Side Navbars -->
    <v-navigation-drawer app temporary fixed v-model="sidenav">
      <v-toolbar color="accent" dark flat>
        <v-toolbar-side-icon @click="toggleSideNav"></v-toolbar-side-icon>
        <router-link to="/" tag="span" style="cursor: pointer">
          <h1 class="title">VueShare</h1>
        </router-link>
      </v-toolbar>
    </v-navigation-drawer>

    <!-- Horizontal Navbar -->
    <v-toolbar fixed color="primary" dark>
      <!-- App Title -->
      <v-toolbar-side-icon @click="toggleSideNav"></v-toolbar-side-icon>
      <v-toolbar-title class="hidden-xs-only">
        <router-link to="/" tag="span" style="cursor: pointer">
          VueShare
        </router-link>
      </v-toolbar-title>
    </v-toolbar>
  </v-app>
</template>
```

Deverá ter o seguinte resultado



Cofinanciado por:



Acrescentamos agora os botões da sidenav

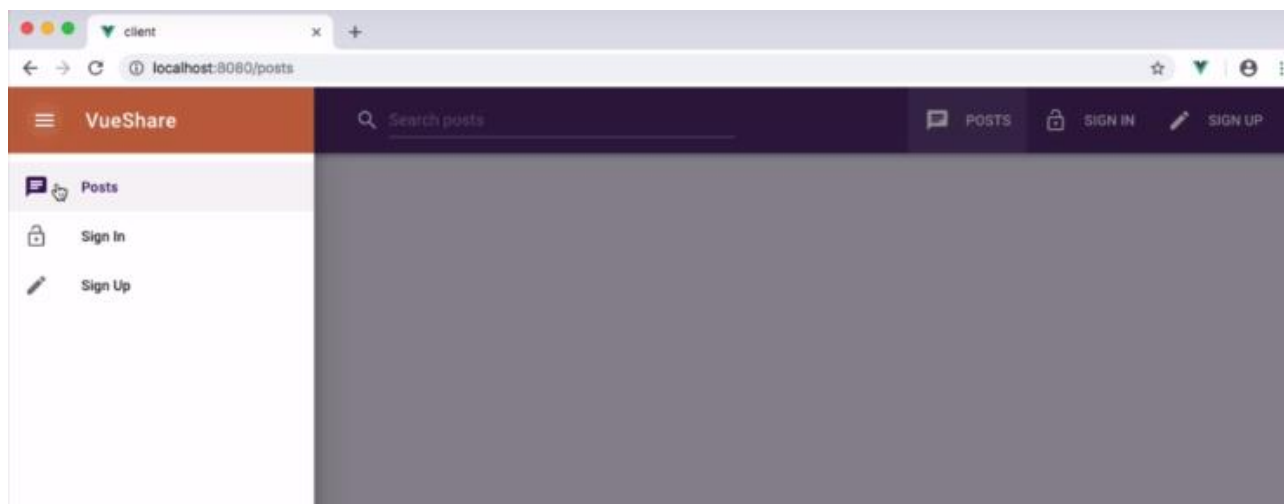
```
</v-divider></v-divider>

<!-- Side Navban Links -->
<v-list>
  <v-list-tile v-for="item in sideNavItems" :key="item.title" :to="item.link">
    <v-list-tile-action>
      <v-icon>{{item.icon}}</v-icon>
    </v-list-tile-action>
    <v-list-tile-content>
      {{item.title}}
    </v-list-tile-content>
  </v-list-tile>
</v-list>
</v-navigation-drawer>
```

Acrescentamos os itens à lista

```
computed: {
  horizontalNavItems() {
    return [
      { icon: "chat", title: "Posts", link: "/posts" },
      { icon: "lock_open", title: "Sign In", link: "/signin" },
      { icon: "create", title: "Sign Up", link: "/signup" }
    ];
  },
  sideNavItems() {
    return [
      { icon: "chat", title: "Posts", link: "/posts" },
      { icon: "lock_open", title: "Sign In", link: "/signin" },
      { icon: "create", title: "Sign Up", link: "/signup" }
    ];
  }
},
methods: {
  toggleSideNav() {
```

E teremos concluído a sidenav bar:



Cofinanciado por:

