

```
npm install primevue@latest --save
```

```
npm install primeicons --save
```

```
npm install primeflex --save
```

```
npm install vue-i18n@9
```

```
npm install axios
```

Conceptos

v-on

<https://es.vuejs.org/v2/guide/events.html>

v-on se utiliza para adjuntar escuchadores de eventos a elementos HTML. El atributo **v-on** permite la ejecución de expresiones JS en respuesta a eventos del DOM.

El atributo **v-on** tiene la siguiente sintaxis básica:

```
v-on:eventName="handler"
```

Donde **eventName** es el nombre del evento en el que se está interesado, como **click**, **submit**, **input**, etc. Y **handler** es el método que se ejecutará cuando se dispare el evento.

Por ejemplo, si desea llamar al método **showAlert** cuando se hace clic en un botón, puede hacerlo de la siguiente manera:

```
<button v-on:click="showAlert">Mostrar alerta</button>
```

En este ejemplo, **v-on:click** establece el escuchador de eventos para el evento de clic, y **showAlert** es el método que se ejecutará cuando se dispare el evento.

También se pueden utilizar modificadores con **v-on**. Por ejemplo, el modificador **.prevent** se utiliza para prevenir el comportamiento predeterminado de un evento, como enviar un formulario o navegar a una nueva página. La sintaxis para esto es:

```
v-on:eventName.prevent="handler"
```


Hay varios otros modificadores, como **.stop**, **.capture**, **.self**, **.once**, etc.

sidebar

<https://primefaces.org/primevue/sidebar>

Sidebar

Sidebar is a panel component displayed as an overlay at the edges of the screen.



The image shows a sidebar component with a white background and a light blue border. It contains five blue square buttons with white icons: a right arrow, a left arrow, a down arrow, an up arrow, and a grid icon. To the right of the sidebar is a blue square button with a white gear icon.

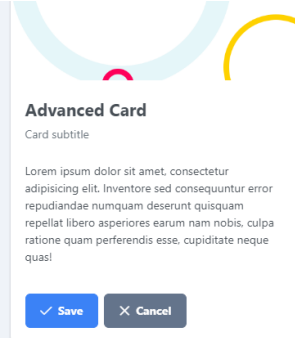
[Documentation](#) [Options API Source](#) [Composition API Source](#) [Browser Source](#)

Import via Module

```
import Sidebar from 'primevue/sidebar';
```

Elemento del prime

<https://primefaces.org/primevue/card>



The image shows a card component with a white background and a light blue border. It has a header with the title "Advanced Card" and a subtitle "Card subtitle". The body contains a paragraph of Lorem Ipsum text. The footer has two buttons: "Save" and "Cancel".

[Documentation](#) [Options API Source](#) [Composition API Source](#) [Browser Source](#)

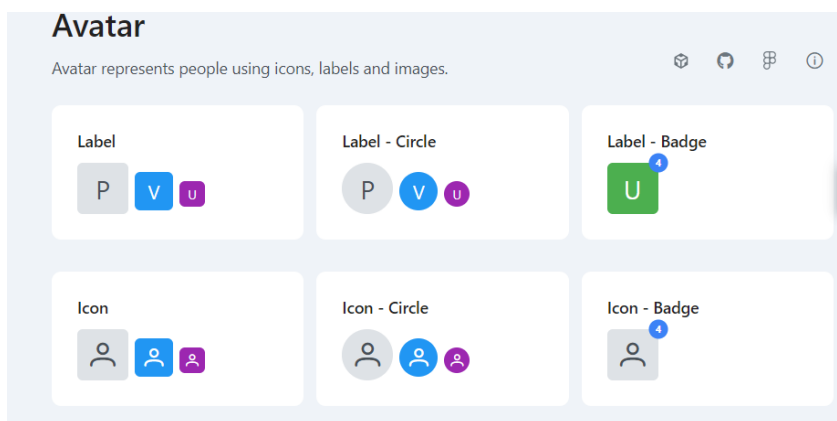
Import via Module

```
import Card from 'primevue/card';
```

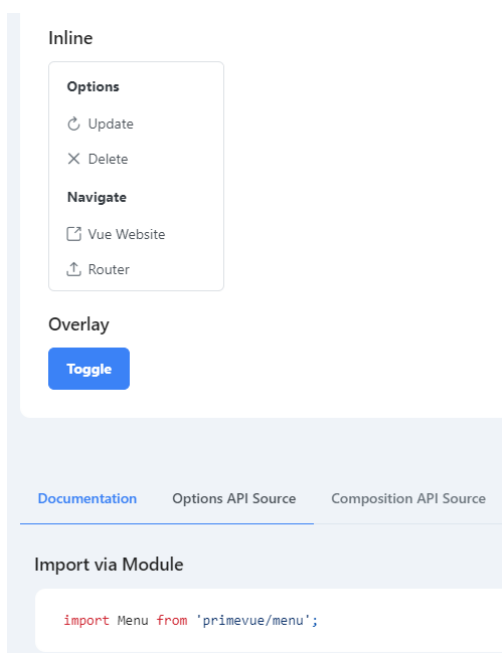
<https://primefaces.org/primevue/button>



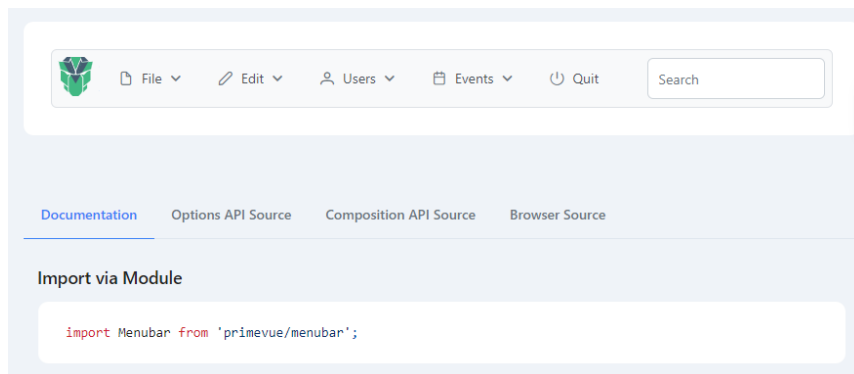
<https://primefaces.org/primevue/avatar>



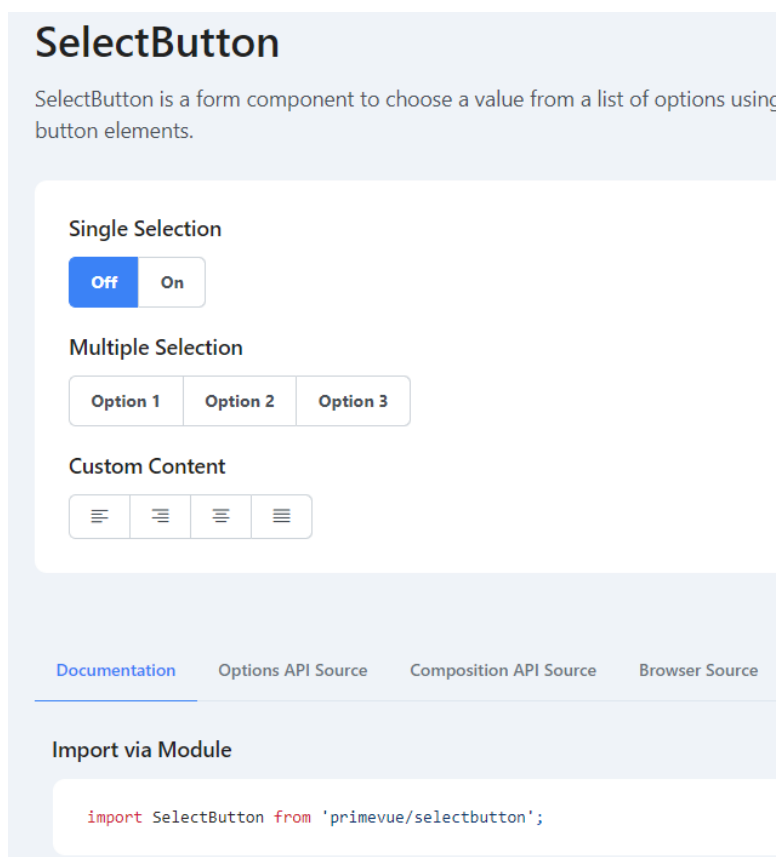
<http://primefaces.org/primevue/menu>



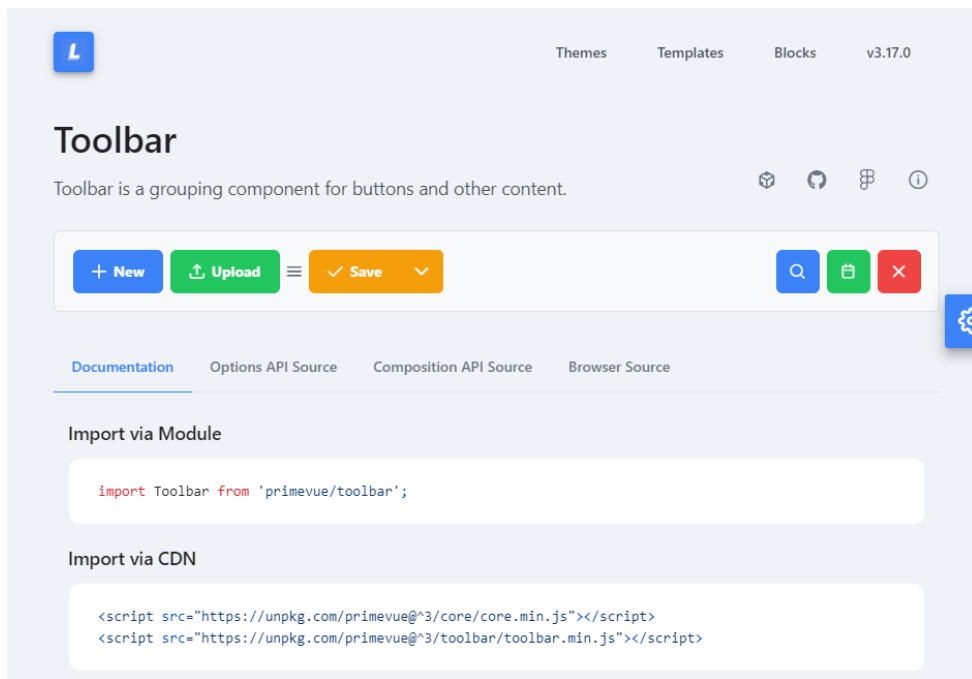
<https://primefaces.org/primevue/menubar>



<https://primefaces.org/primevue/selectbutton>



<https://primefaces.org/primevue/toolbar>



<https://primefaces.org/primevue/galleria>



<https://primefaces.org/primevue/cascadeselect>

<code>emptyMessage</code>	string	No available options	Text to be displayed when there are no options available. Defaults to value from PrimeVue locale configuration.
<code>tabindex</code>	number	0	Index of the element in tabbing order.
<code>aria-label</code>	string	null	Defines a string value that labels an interactive element.
<code>aria-labelledby</code>	string	null	Establishes relationships between the component and label(s) where its value should be one or more element IDs.

Accessibility

Screen Reader

Value to describe the component can either be provided with `aria-labelledby` or `aria-label` props. The `CascadeSelect` element has a `combobox` role in addition to `aria-haspopup` and `aria-expanded` attributes. The relation between the combobox and the popup is created with `aria-controls` that refers to the id of the popup.

The popup list has an id that refers to the `aria-controls` attribute of the `combobox` element and uses `tree` as the role. Each list item has a `treeitem` role along with `aria-label`, `aria-selected` and `aria-expanded` attributes. The container element of a treenode has the `group` role. The `aria-setsize`, `aria-posinset` and `aria-level` attributes are calculated implicitly and added to each treeitem.

```
<span id="dd1">Options</span>
<CascadeSelect aria-labelledby="dd1" />

<CascadeSelect aria-label="Options" />
```

Iniciando el proyecto

Index.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite App</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

Importamos todo el elemento necesario

Main.js

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

import './assets/main.css';
//import i18n from "@/i18n";
import PrimeVue from "primevue/config";
```

```
// App Theme
import 'primevue/resources/themes/md-light-indigo/theme.css';
import 'primevue/resources/primevue.min.css';
import 'primeicons/primeicons.css';

// Add PrimeFlex
import 'primeflex/primeflex.css';

// Add Components
import Card from "primevue/card";
import Button from "primevue/button";
import Sidebar from "primevue/sidebar";
import Avatar from "primevue/avatar";
import Menu from "primevue/menu";
import Menubar from "primevue/menubar";
import SelectButton from "primevue/selectbutton";
import Toolbar from "primevue/toolbar";

createApp(App)
  .use(PrimeVue, { ripple: true })
  // .use(i18n)
  .component('pv-card', Card)
  .component('pv-button', Button)
  .component('pv-select-button', SelectButton)
  .component('pv-sidebar', Sidebar)
  .component('pv-avatar', Avatar)
  .component('pv-menu', Menu)
  .component('pv-menubar', Menubar)
  .component('pv-toolbar', Toolbar)
  .mount('#app');
```

App.vue

```
<template>
  <div class="w-full">
    <div>
      <pv-menubar class="sticky bg-primary">
        <template #start>
          <pv-button label="CatchUp-inicio" icon="pi pi-bars"
@click="toggleSidebar"/>
        </template>
        <template #end>
          <pv-button label="CatchUp-end" icon="pi pi-bars"
@click="toggleSidebar"/>
        </template>
      </pv-menubar>
    </div>
    <div>
      </div>
    </div>
  </div>
</template>

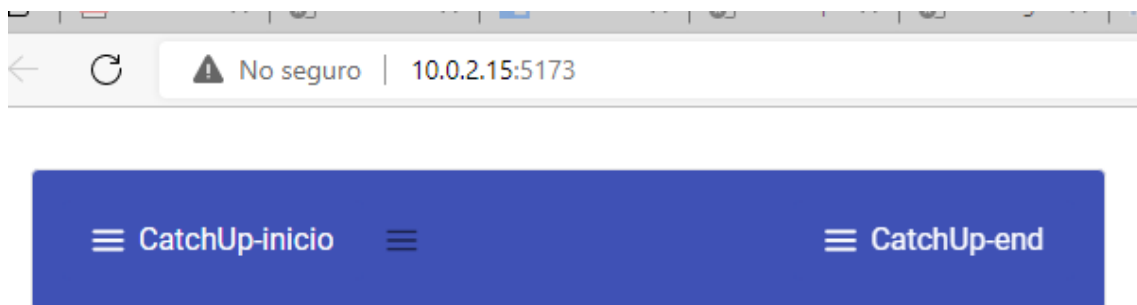
<script>
export default {
  name: 'App',
```

```

data() {
  return {
    sidebarVisible: false
  }
},
methods: {
  toggleSidebar() {
    this.sidebarVisible = !this.sidebarVisible;
  }
}
}
</script>

```

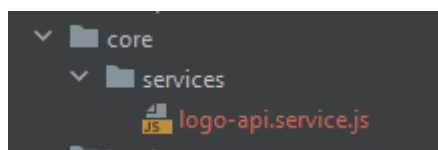
Compilamos y vemos el resultado



Generamos los Servicios que pensamos consumir para este Proyecto.

Para consumir el servicio de logo

Creamos el news/services/logo-api.service.js

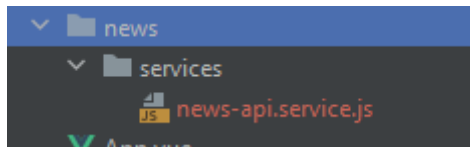


```

export class LogoApiService {
  getUrlToLogo(source) {
    return `https://logo.clearbit.com/${ new URL(source.url).host
  };
}
}

```

Para consumer



new/services/news-api.service.js

deben usar el token de sus cuentas.

```
import axios from "axios";
import { LogoApiService } from "@core/services/logo-api.service";

const http = axios.create({
  baseURL: 'https://newsapi.org/v2/'
});

export class NewsApiService {
  apiKey = 'xxxxxxxxxxxxxxxx';
  logoApi = new LogoApiService();

  getSources() {
    return http.get(`top-headlines/sources?apiKey=${this.apiKey}`);
  }

  getArticlesForSource(sourceId) {
    return http.get(`top-headlines/sources=${sourceId}&apiKey=${this.apiKey}`);
  }

  getUrlToLogo(source) {
    return this.logoApi.getUrlToLogo(source);
  }
}
```

adicionamos los Servicios a vue.app

```
<template>
  <div class="w-full">
    <div>
      <pv-menubar class="sticky bg-primary">
        <template #start>
          <pv-button label="CatchUp-inicio" icon="pi pi-bars"
            @click="toggleSidebar"/>
        </template>
        <template #end>
          <pv-button label="CatchUp-end" icon="pi pi-bars"
            @click="toggleSidebar"/>
        </template>
      </pv-menubar>
    </div>
    <div>
    </div>
  </div>
</template>
```

```

<script>
import {NewsApiService} from "@news/services/news-api.service";

export default {
  name: 'App',
  data() {
    return {
      sidebarVisible: false,
      articles: [],
      errors: [],
      newsApi: new NewsApiService()
    }
  },

  created() {
    console.log('created');
    this.getArticlesForSource('bbc-news');
  },

  methods: {

    // Fetch Articles for selected Source

    getArticlesForSource(sourceId) {
      this.newsApi.getArticlesForSource(sourceId)
        .then(response => {
          this.articles = response.data.articles;
          console.log(response.data);
        })
        .catch(e => {
          this.errors.push(e);
        });
    },

    // Fetch Articles for selected Source with Logo URL

    getArticlesForSourceWithUrl(source) {
      this.newsApi.getArticlesForSource(source.id)
        .then(response => {
          this.articles = response.data.articles;
          this.articles.map(article => article.source.urlToLogo =
source.urlToLogo);
          console.log(response.data);
        })
        .catch(e => {
          this.errors.push();
        });
    },

    // On Source selected

    setSource(source) {
      this.getArticlesForSourceWithUrl(source);
      this.sidebarVisible = !this.sidebarVisible;
    },

    toggleSidebar() {
      this.sidebarVisible = !this.sidebarVisible;
    }
  }
}

```

```
}  
</script>
```

probamos y todo compila.

<http://localhost:5173/>

iniciamos los componentes para ver la información obtenida

ejemplo de cómo se genera el img

```
<Galleria :value="images" :responsiveOptions="responsiveOptions" :numVisible="5" containerStyle="max-wid  
  <template #header>  
    <h1>Header</h1>  
  </template>  
  <template #item="slotProps">  
      
  </template>  
  <template #footer>  
    <h1>Footer</h1>  
  </template>  
</Galleria>
```

como se obtiene usando el componente avatar de primevue

Properties of Avatar

Any property as style and class are passed to the main container element. Following are the additional properties component.

Name	Type	Default	Description
label	string	null	Defines the text to display.
icon	string	null	Defines the icon to display.
image	string	null	Defines the image to display.
size	string	null	Size of the element, valid options are "large" and "xlarge".
shape	string	square	Shape of the element, valid options are "square" and "circle".

Accessibility

Screen Reader

Value to describe the component can either be provided with `aria-labelledby` or `aria-label` props. The `cascadeselect` element has a `combobox` role in addition to `aria-haspopup` and `aria-expanded` attributes. The relation between the combobox and the popup is created with `aria-controls` that refers to the id of the popup.

The popup list has an id that refers to the `aria-controls` attribute of the `combobox` element and uses `tree` as the role. Each list item has a `treeitem` role along with `aria-label`, `aria-selected` and `aria-expanded` attributes. The container element of a treenode has the `group` role. The `aria-setsize`, `aria-posinset` and `aria-level` attributes are calculated implicitly and added to each `treeitem`.

```
<span id="dd1">Options</span>
<CascadeSelect aria-labelledby="dd1" />

<CascadeSelect aria-label="Options" />
```

creamos el componente `/components/article-card.component.vue`

```
<template>
  <pv-card class="m-2">
    <template #header>
      
    </template>
    <template #title>
      <p>{{ article.title }}</p>
    </template>
    <template #content>
      <p class="flex align-content-start flex-wrap">
        <span class="flex align-items-center justify-content-center mr-2">
          <pv-avatar :image="article.source.urlToLogo" :aria-label="article.source.name" shape="circle"/>
        </span>
        <span class="flex align-items-center justify-content-center">
          {{ article.source.name }}
        </span>
      </p>
      <p class="flex align-content-start flex-wrap mt-4">
        {{ article.description }}
      </p>
    </template>
    <template #footer>
      <template>
    </template>
  </pv-card>
</template>

<script>
export default {
  name: "article-card",
  props: {
    article: null,
  }
}
</script>
```

```
<style scoped>

</style>
```

/components/main-content.component.vue

```
<template>
  <div v-for="article in articles">
    <article-card :article="article"></article-card>
  </div>
</template>

<script>
import ArticleCard from "@components/article-card.component.vue";
export default {
  name: "main-content",
  components: {ArticleCard},
  props: {
    articles: Array
  }
}
</script>

<style scoped>

</style>
```

/components/unavailable-content.component.vue

```
<template>
  <div class="grid align-content-start">
    <div class="col-12"><h4>News Service is unavailable now</h4></div>
    <div class="col-12" v-for="error in errors"><h6>{{ error
  }}</h6></div>
  </div>
</template>

<script>
export default {
  name: "unavailable-content",
  props: {
    errors: null,
  }
}
</script>

<style scoped>

</style>
```

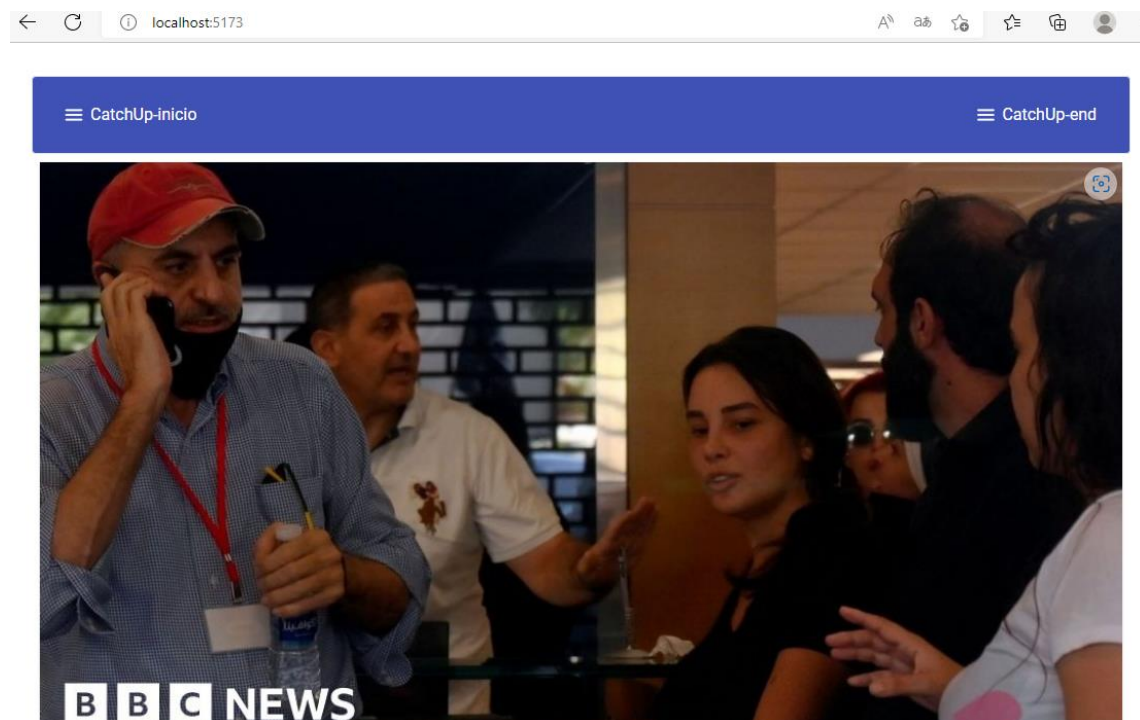
En app.vue adicionamos los componentes

```
import MainContent from "@/components/main-content.component.vue";
import UnavailableContent from "@/components/unavailable-
content.component.vue";
```

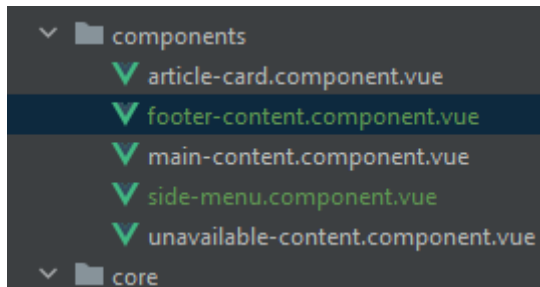
```
</pv-menubar>
</div>
<div>
  <main-content v-if="errors" :articles="articles"></main-content>
  <unavailable-content v-else :errors="errors"></unavailable-content>
</div>
```

```
export default {
  name: 'App',
  components: {UnavailableContent, MainContent},
```

compilando y mostramos



generamos el footer



creamos el /components/footer-content.component.vue

```
<template>
  <div class="grid bg-primary mt-4 p-4 align-content-start">
    <div class="col-12 ml-3 align-items-center justify-content-center">
      <p>Copyright &copy; 2022. ACME Studios</p>
    </div>
    <div class="col-12 ml-3 mt-3 align-items-center justify-content-center">
      <p>
        Made with <i class="pi pi-heart text-red-50"/>
        using <a class="text-white"
href="https://www.primefaces.org/primevue"
target="_blank">PrimeVue</a>
        Developer Team
      </p>
    </div>
  </div>
</template>

<script>
export default {
  name: "footer-content"
}
</script>

<style scoped>
</style>
```

app.vue

```
<div>
  <main-content v-if="errors" :articles="articles"></main-content>
  <unavailable-content v-else :errors="errors"></unavailable-content>
</div>
<footer-content/>
</div>
```

```
import UnavailableContent from "@/components/unavailable-content.component.vue";
import FooterContent from "@/components/footer-content.component.vue";
```

```
name: 'App',
components: { FooterContent, UnavailableContent, MainContent },
```

al final , luego de compilar tendríamos lo siguiente

China's Xi to meet Putin in first foreign trip since pandemic



BBC News

He will meet Russian leader Vladimir Putin on the sidelines of a regional summit in Uzbekistan.

Copyright © 2022. ACME Studios

Made with ❤️ using PrimeVue Developer Team

adicionamos el siderbar

components/side-menu.component.vue"

```
components
  ✓ article-card.component.vue
  ✓ footer-content.component.vue
  ✓ main-content.component.vue
  ✓ side-menu.component.vue
  ✓ unavailable-content.component.vue
```

creamos el sider bar con los Servicios que permitan cargar informacion

```
<template>
  <pv-sidebar v-model:visible="visible">
    <div v-for="source in sources" class="m-4">
      <div @click="onSourceSelected(source)" class="flex align-
content-start flex-wrap">
        <span class="flex align-items-center justify-content-center
mr-2"><pv-avatar :image="source.urlToLogo" shape="circle" :aria-
label="source.name"></pv-avatar></span>
        <span class="flex align-items-center justify-content-
```



```

center">{{ source.name }}</span>
    </div>
  </div>
</pv-sidebar>
</template>

<script>
import { NewsApiService } from "@/news/services/news-api.service";

export default {
  name: "side-menu",
  props: {
    visible: Boolean
  },
  data() {
    return {
      sources: [],
      errors: [],
      newsApi: new NewsApiService()
    };
  },
  created() {
    this.newsApi.getSources()
      .then(response => {
        this.sources = response.data.sources;
        this.sources.map(source => source.urlToLogo =
this.newsApi.getUrlToLogo(source));
        console.log('data: ');
        console.log(response.data.sources);
      })
      .catch(e => {
        this.errors.push(e);
      })
  },
  methods: {
    onSourceSelected(source) {
      this.$emit('source-selected', source);
    }
  }
}
</script>

<style scoped>

</style>

```

adicionamos en app.vue

```

<template #start>
  <pv-button label="CatchUp" icon="pi pi-bars"
@click="toggleSidebar"/>
  <side-menu v-model:visible="sidebarVisible" v-on:source-
selected="setSource"></side-menu>
</template>

```

```

import UnavailableContent from "@/components/unavailable-

```

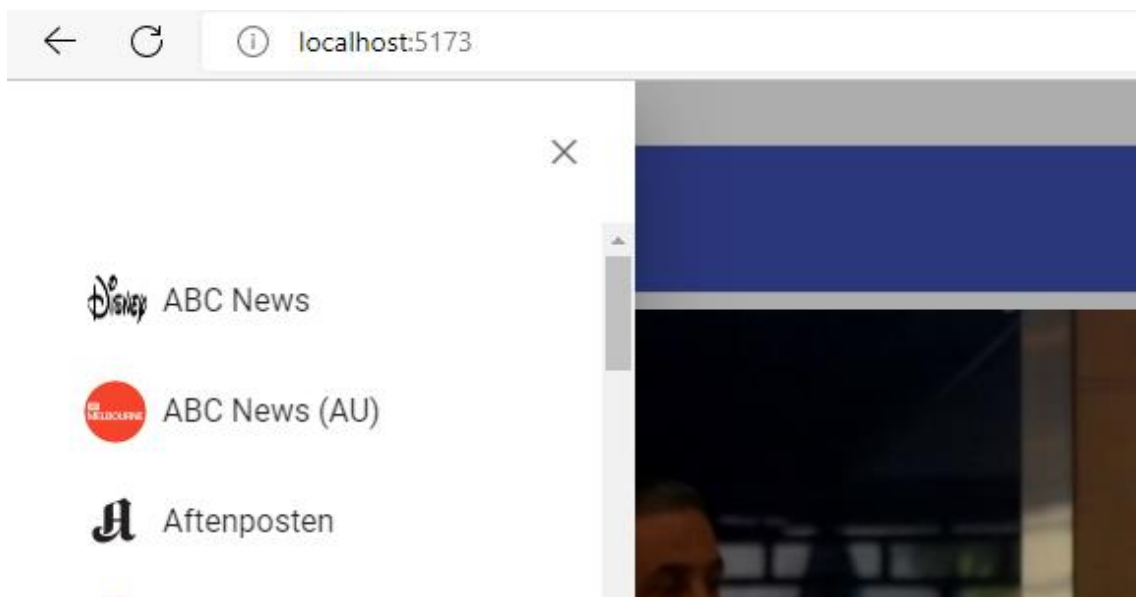
```
content.component.vue";
import SideMenu from "@/components/side-menu.component.vue";
```

```
export default {
  name: 'App',
  components: { FooterContent, SideMenu, UnavailableContent,
MainContent},
```

compilando obtenemos que el

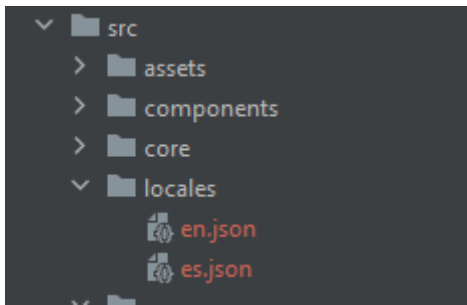


permita el mostrar el menubar



incluimos el boton para soporte de i18n

creamos primero los objetos json.



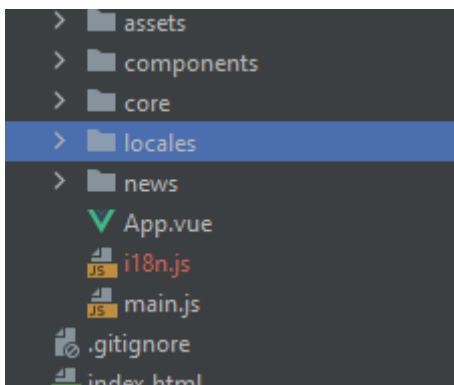
en.json

```
{
  "read-more": "Read More",
  "unavailable-news": "News Service is unavailable now",
  "authoring-phrase": {
    "intro": "Made with",
    "use": "using",
    "author": "by {brand} Developer Team"
  }
}
```

es.json

```
{
  "read-more": "Ver más",
  "unavailable-news": "Servicio de Noticias no disponible en este momento",
  "authoring-phrase": {
    "intro": "Hecho con",
    "use": "utilizando",
    "author": "por el Equipo de Desarrollo de {brand}"
  }
}
```

generamos i18n.js



```
import {createI18n} from "vue-i18n";
import en from "../locales/en.json";
import es from "../locales/es.json";

const i18n = createI18n({
  legacy: false,
  locale: 'en',
  globalInjection: true,
  messages: { en, es }
});

export default i18n;
```

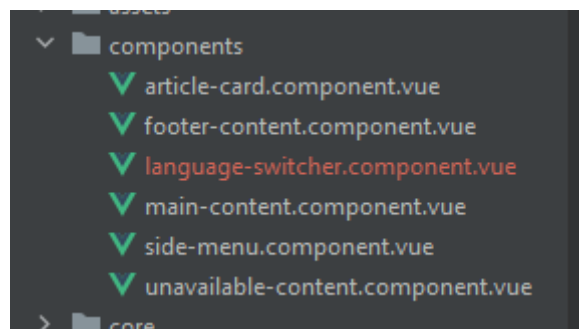
incluimos en el main.js

```
import './assets/main.css';
import i18n from "@i18n";
import PrimeVue from "primevue/config";
```

```
createApp(App)
  .use(PrimeVue, { ripple: true })
  .use(i18n)
  .component('pv-card', Card)
  .component('pv-button', Button)
  .component('pv-select-button', SelectButton)
  .component('pv-sidebar', Sidebar)
  .component('pv-avatar', Avatar)
  .component('pv-menu', Menu)
  .component('pv-menubar', Menubar)
  .component('pv-toolbar', Toolbar)
  .mount('#app');
```

creamos un objeto

/components/language-switcher.component.vue



```

<template>
  <pv-select-button v-model="$18n.locale" :options="languages"
class="bg-primary uppercase"></pv-select-button>
</template>

<script>
export default {
  name: "language-switcher",
  data() {
    return {
      languages: ['en', 'es'],
      language: 'en',
    }
  }
}
</script>

```

adicionamos en app.vue el componente `<language-switcher/>`

```

<pv-menubar class="sticky bg-primary">
  <template #start>
    <pv-button label="CatchUp-inicio" icon="pi pi-bars"
@click="toggleSidebar"/>
    <side-menu v-model:visible="sidebarVisible" v-on:source-
selected="setSource"></side-menu>
  </template>
  <template #end>
    <language-switcher/>
  </template>
</pv-menubar>

```

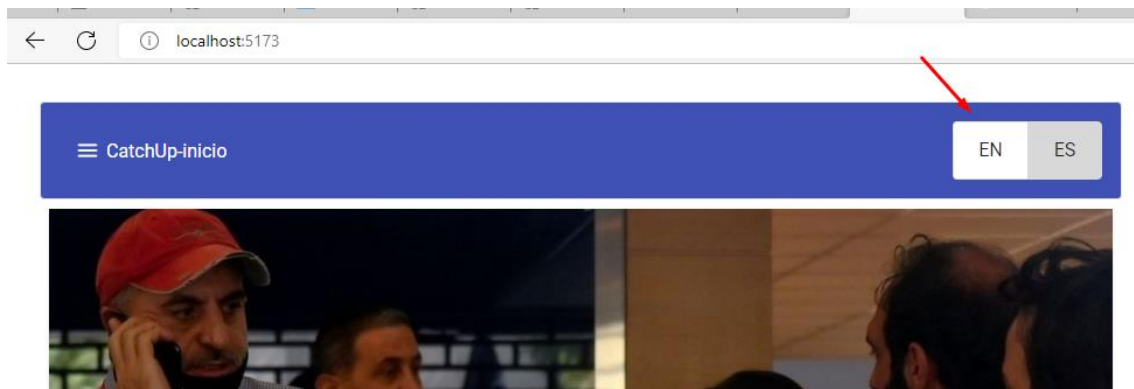
```

import FooterContent from "@components/footer-content.component.vue";
import LanguageSwitcher from "@components/language-
switcher.component.vue";

export default {
  name: 'App',
  components: {LanguageSwitcher, FooterContent, SideMenu,
UnavailableContent, MainContent},

```

si compilamos vemos que



se genera el switch

Actualizamos el footer y el unavailable

```
<div class="col-12 ml-3 mt-3 align-items-center justify-content-center">
  <p>
    {{ $t('authoring-phrase.intro') }} <i class="pi pi-heart text-red-50"/>
    {{ $t('authoring-phrase.use') }} <a class="text-white"
href="https://www.primefaces.org/primevue"
target="_blank">PrimeVue</a>
    {{ $t('authoring-phrase.author', { brand: "ACME" }) }}
  </p>
</div>
```

```
<div class="grid align-content-start">
  <div class="col-12"><h4>{{ $t('unavailable-news') }}</h4></div>
  <div class="col-12" v-for="error in errors"><h6>{{ error
}}</h6></div>
</div>
```

Copyright © 2022. ACME Studios

Made with ♥ using PrimeVue by ACME Developer Team

Corrección final

<Sidebar v-model:visible="visible"> no se está usando correctamente la sintaxis del modificador **v-model** en Vue 3.

En Vue 3, el modificador **v-model** ha cambiado. En lugar de usar la sintaxis antigua **v-model:propName**, ahora se debe utilizar la directiva **v-bind** para enlazar la propiedad **propName** y la directiva **v-on** para enlazar el evento **update:propName**.

Por lo tanto, para corregir el error, se debería cambiar la línea de código por:

```
<Sidebar v-bind:visible="visible" v-on:update:visible="val => visible = val">
```

En este caso, estamos enlazando la propiedad **visible** al valor de la variable **visible**, y también enlazando el evento **update:visible** para que actualice el valor de **visible** cuando cambie el estado del componente **Sidebar**.