Contenido

Iniciar el proyecto learning-center.

Precondiciones debes tener instalado el node

Creando el proyecto en learning-center.

npm init vue@latest

npm init vue@latest

```
icrosoft Windows [Versión 10.0.19044.1826]
c) Microsoft Corporation. Todos los derechos reservados.

:\semana04>npm init vue@latest
eed to install the following packages:
  create-vue@3.3.4
k to proceed? (y)
```

learning-center

```
D:\2023-02\semana 05>npm init vue@latest
npx: installed 1 in 0.978s

Vue.js - The Progressive JavaScript Framework

√ Project name: ... learning-center
√ Add TypeScript? ... No / Yes
√ Add JSX Support? ... No / Yes
√ Add Vue Router for Single Page Application development? ... No / Yes
√ Add Pinia for state management? ... No / Yes
√ Add Vitest for Unit Testing? ... No / Yes
√ Add an End-to-End Testing Solution? » No
√ Add ESLint for code quality? ... No / Yes

Scaffolding project in D:\2023-02\semana 05\learning-center...

Done. Now run:

  cd learning-center
  npm install
  npm run dev
```

Abrimos el projecto

git init

```
MINGW64:/d/2023-02/semana 05/learning-center

Juan@DESKTOP-A8E2LPJ MINGW64 /d/2023-02/semana 05/learning-center
$ git init
Initialized empty Git repository in D:/2023-02/semana 05/learning-center/.git/
```

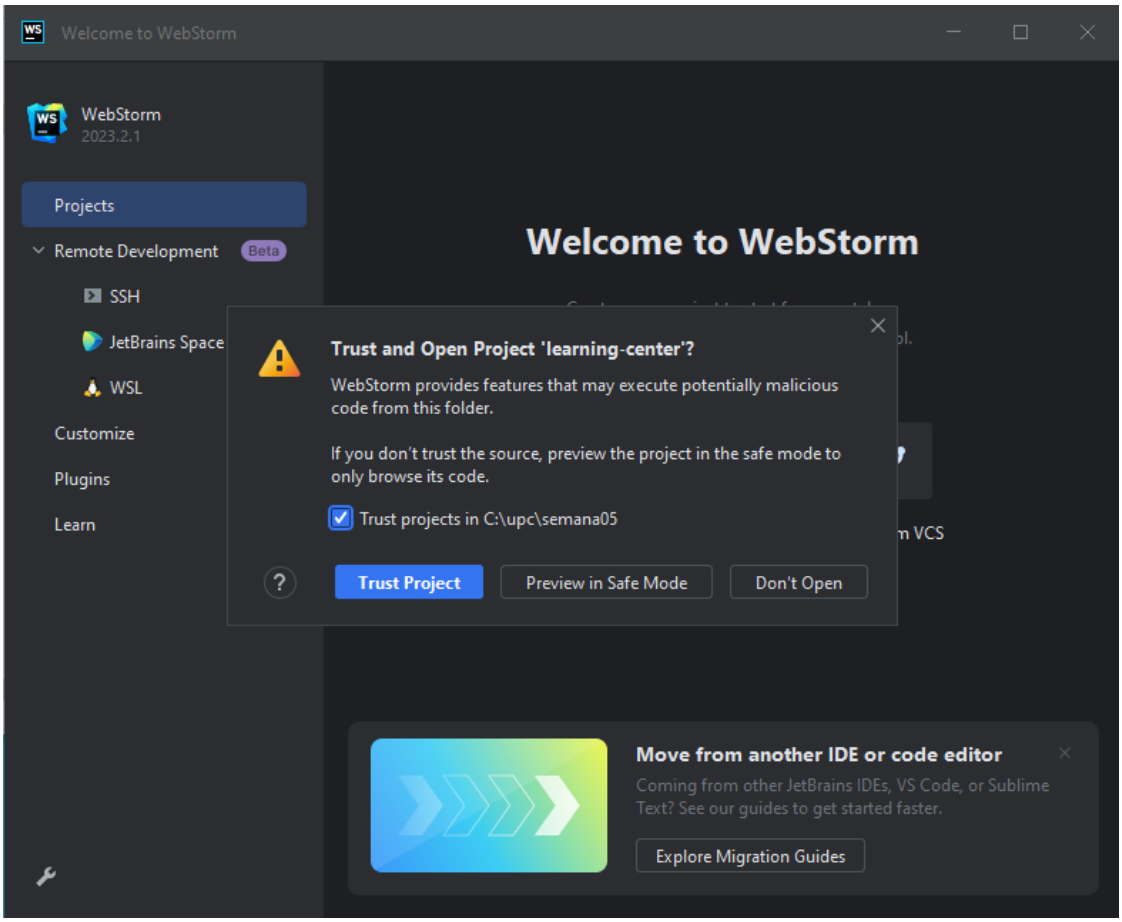
git add .

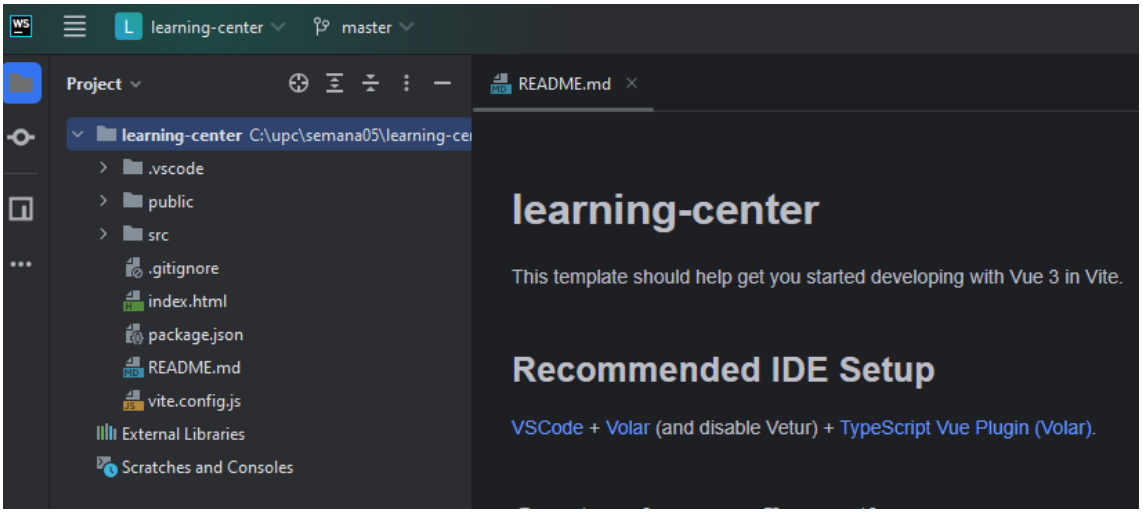
```
Juan@DESKTOP-A8E2LPJ MINGW64 /d/2023-02/semana 05/learning-center (master)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the r
warning: in the working copy of '.vscode/extensions.json', LF will be replaced
warning: in the working copy of 'README.md', LF will be replaced by CRLF the ne
warning: in the working copy of 'index.html', LF will be replaced by CRLF the r
warning: in the working copy of 'package.json', LF will be replaced by CRLF the
warning: in the working copy of 'src/App.vue', LF will be replaced by CRLF the
warning: in the working copy of 'src/assets/base.css', LF will be replaced by C
warning: in the working copy of 'src/assets/logo.svg', LF will be replaced by C
```

git commit -m “Initial commit”

```
Juan@DESKTOP-A8E2LPJ MINGW64 /d/2023-02/semana 05/learning-center (master)
$ git commit -m "Initial commit"
[master (root-commit) 400732d] Initial commit
 20 files changed, 532 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .vscode/extensions.json
 create mode 100644 README.md
 create mode 100644 index.html
 create mode 100644 package.json
 create mode 100644 public/favicon.ico
```

Realizar el commit “Initial commit”





Limpiar proyecto inicial

borrar los siguientes archivos

Path	Extension	Status
Modified Files		
src/App.vue	.vue	Modified
src/assets/base.css	.css	Missing
src/assets/logo.svg	.svg	Missing
src/assets/main.css	.css	Missing
src/components/HelloWorld.vue	.vue	Missing
src/components/TheWelcome.vue	.vue	Missing
src/components/WelcomeItem.vue	.vue	Missing
src/components/icons/IconCommunity.vue	.vue	Missing
src/components/icons/IconDocumentation.vue	.vue	Missing
src/components/icons/IconEcosystem.vue	.vue	Missing
src/components/icons/IconSupport.vue	.vue	Missing
src/components/icons/IconTooling.vue	.vue	Missing
src/main.js	.js	Modified
src/views/HomeView.vue	.vue	Modified

eliminar las lineas

src/App.vue: 2b942b42

```
1  <script setup>
2  import { RouterLink, RouterView } from 'vue-router'
3  import HelloWorld from './components/HelloWorld.vue'
4  </script>
5
6  <template>
7    <header>
8      <img alt="Vue logo" class="logo" src="@/assets/logo.svg" width="125" heig
9
10     <div class="wrapper">
11       <HelloWorld msg="You did it!" />
12
13       <nav>
14         <RouterLink to="/">Home</RouterLink>
15         <RouterLink to="/about">About</RouterLink>
16       </nav>
17     </div>
18   </header>
19
20   <RouterView />
21 </template>
```

src/main.js: 2b942b42

```
1  import './assets/main.css'
2
3  import { createApp } from 'vue'
4  import App from './App.vue'
5  import router from './router'
6
7  const app = createApp(App)
8
9  app.use(router)
10
11 app.mount('#app')
12
```

```
src/views/HomeView.vue: 2b942b42
1 <script setup>
2 import TheWelcome from '../components/TheWelcome.vue'
3 </script>
4
5 <template>
6   <main>
7     <TheWelcome />
8   </main>
9 </template>
```

Realizar el commit “chore: Clean Project”

npm install

```
Terminal   Local

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\upc\semana05\learning-center> npm install
```

```
PS C:\upc\semana05\learning-center> npm  run dev
```

npm run dev

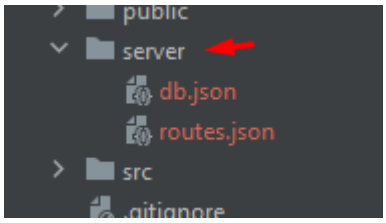
```
Terminal   Local

VITE v4.4.9  ready in 1080 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h to show help
```

Generamos los servicios

generamos los siguientes archivos



routes.json

```
{
  "/api/v1/*" : "/$1"
}
```

db.json

```
{
  "tutorials": [
    {
      "id": 1,
      "title": "The Vue Tutorials",
      "description": "The best tips and tutorials for Vue.",
      "published": false
    },
    {
      "id": 2,
      "title": "Amazing Microsoft .NET",
      "description": "Weekly tutorials about .NET and ASP.NET Core.",
      "published": false
    },
    {
      "id": 3,
      "title": "JavaScript for All",
      "description": "Tips an tricks about JavaScript from scratch.",
      "published": false
    },
    {
      "id": 4,
      "title": "Vue Unleashed",
      "description": "Vue at its best.",
      "published": false
    }
  ]
}
```

ejecutamos lo siguiente:

json-server --watch db.json --routes routes.json

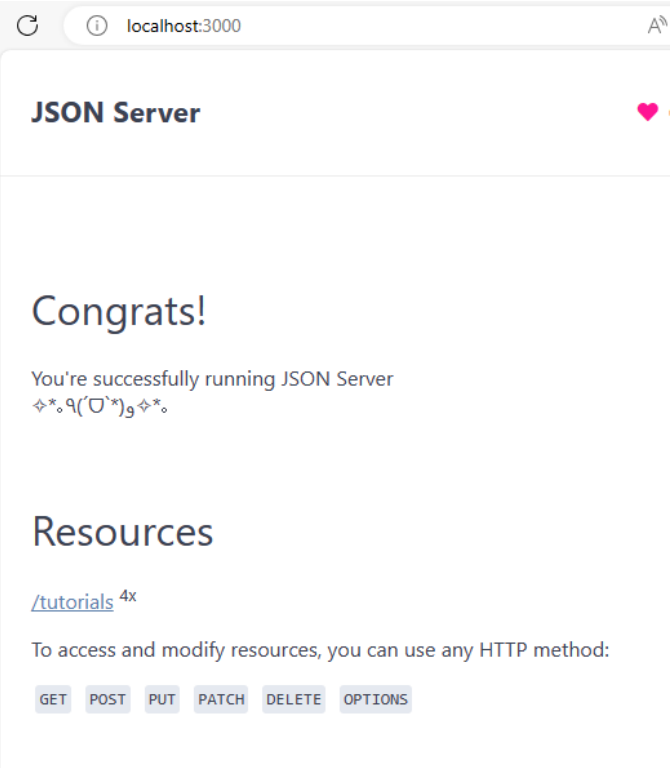
```
PS D:\2023-02\semana 05\learning-center\src> cd .\server\
PS D:\2023-02\semana 05\learning-center\src\server> json-server --watch db.json --routes routes.json

  \{^_^}/ hi!

  Loading db.json
  Loading routes.json
  Done

  Resources
  http://localhost:3000/tutorials
```

http://localhost:3000/



http://localhost:3000/tutorials

localhost:3000/tutorials


```
[
    {
        "id": 1,
        "title": "The Vue Tutorials",
        "description": "The best tips and tutorials for Vue.",
        "published": false
    },
    {
        "id": 2,
        "title": "Amazing Microsoft .NET",
        "description": "Weekly tutorials about .NET and ASP.NET Core.",
        "published": false
    },
    {
        "id": 3,
        "title": "JavaScript for All",
        "description": "Tips an tricks about JavaScript from scratch.",
        "published": false
    },
    {
        "id": 4,
        "title": "Vue Unleashed",
        "description": "Vue at its best.",
        "published": false
    }
]
```


http://localhost:3000/api/v1/tutorials

localhost:3000/api/v1/tutorials


```
[
    {
        "id": 1,
        "title": "The Vue Tutorials",
        "description": "The best tips and tutorials for Vue.",
        "published": false
    },
    {
        "id": 2,
        "title": "Amazing Microsoft .NET",
        "description": "Weekly tutorials about .NET and ASP.NET Core.",
        "published": false
    },
    {
        "id": 3,
        "title": "JavaScript for All",
        "description": "Tips an tricks about JavaScript from scratch.",
        "published": false
    },
    {
        "id": 4,
        "title": "Vue Unleashed",
        "description": "Vue at its best.",
        "published": false
    }
]
```


Realizar el commit “**feat:** implemented services”

Instalamos Dependencias

```
Instalar en el proyecto
npm install primevue@latest --save
npm install primeicons --save
npm install primeflex --save
npm install axios

npm install axios@^1.3.5
npm install primeflex@^3.3.0
npm install primeicons@^6.0.1
npm install primevue@^3.26.1
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS D:\2023-02\semana 05\learning-center\src> npm install axios@^1.3.5
[        ..........] / extract:form-data: sill extract form-data@^4.0.0 extract
```

Realizar el commit “**chore**: Install dependencies”

Generamos los servicios de consumo del Json server

Generamos el consumo de los servicios.

http-common.js

/ shared /services/http-common.js

```
import axios from 'axios';

export default axios.create({
   baseURL: 'http://localhost:3000/api/v1',
   headers: { 'Content-type': 'application/json' }
});
```

http-common.js

/learning/services/tutorials-api.service.js

```
import http from '../../shared/services/http-common';

export class TutorialsApiService {
    getAll() {
        return http.get('/tutorials');
    }
```

```
    getById(id) {
        return http.get(`/tutorials/${id}`);
    }

    create(data) {
        return http.post('/tutorials', data);
    }

    update(id, data) {
        return http.put(`/tutorials/${id}`, data);
    }

    delete(id) {
        return http.delete(`/tutorials/${id}`);
    }

    findByTitle(title) {
        return http.get(`/tutorials?title=${title}`);
    }
}
```

se logró generar los servicios que se consumirán del fake api.

Realizar el commit “**feat:** implemented consumer services”

Definición del Router vue

https://router.vuejs.org/api/

las etiquetas router-link.

Estas etiquetas son solo enlaces de anclaje elegantes. Sin embargo, a diferencia de un enlace ancla (etiqueta <a href="">), el <router-link> no recargará toda la página. Recuerda que Vue es una aplicación de una single-page. Los datos de la aplicación ya se han descargado del servidor. Cuando enrutamos a otra vista, la aplicación simplemente oculta cierta información y muestra la información solicitada. Las etiquetas de router-link tienen una propiedad que se refiere a qué página visitar. La etiqueta <router-view/> es lo que representa el componente correcto cuando se activan los enlaces de navegación.

Navegando desde la vista. Router link

Al instalar vue router, se crea una etiqueta HTML especial para poder navegar a las rutas desde el Sistema de vistas de Vue llamada router-link. Veamos un ejemplo:

```
<router-link to="/about">Link a la página de about</router-link>
```

También puedes navegar a una ruta usando su nombre (el parámetro name que has configurado antes en el array de rutas), esto es interesante porque si decides cambiar la url de la ruta, vas a poder seguir navegando correctamente porque el nombre sigue siendo el mismo.

```
<router-link :to="{ name: 'user'}">User</router-link>
```

Fíjate que he puesto los dos puntos antes de el atributo to para poder pasar un objeto de javascript.

Incluso puedes pasar una variable definida en el data o variable computada para decidir a qué ruta ir:

```
<router-link :to="user">User</router-link>
```

En este caso se creará un enlace a la ruta definida en la variable user.

Esta etiqueta renderizará una etiqueta <a> con el href ya configurado a la ruta que especifiques dentro del **to**.

Adicional

v-for

cuando queremos que los cambios se reflejen si modificamos el arreglo u objeto, es necesario proporcionar al elemento una clave. Esta clave debe ser única, y debe ser un tipo de dato primitivo. En el caso de las routes la clave podría ser el label, pues no se repite. La misma es especificada con :key

custom

```
(property) RouterLinkProps.custom?: boolean | undefined

Whether RouterLink should not wrap its content in an a tag. Useful when using v-slot to create a custom RouterLink
```

to

```
(property) RouterLinkOptions.to: RouteLocationRaw

Route Location the link should navigate to when clicked on
```

v-slot

revisar:

https://vuejs.org/guide/components/slots.html

https://github.com/vuejs/rfcs/blob/master/active-rfcs/0001-new-slot-syntax.md

https://github.com/vuejs/rfcs/blob/master/active-rfcs/0002-slot-syntax-shorthand.md

https://vuedose.tips/new-v-slot-directive-in-vue-js-2-6-0

Definición del Router vue

modificamos el **main.js**

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import ToastService from "primevue/toastservice";
import PrimeVue from "primevue/config";
// PrimeVue Material Design Theme
import "primevue/resources/themes/md-light-indigo/theme.css";
import "primevue/resources/primevue.min.css";
import "primeicons/primeicons.css";
import "primeflex/primeflex.css";
// PrimeVue Components
import DataTable from "primevue/datatable";
import Column from "primevue/column";
import Toolbar from "primevue/toolbar";
import InputText from "primevue/inputtext";
import Textarea from "primevue/textarea";
import Button from "primevue/button";
import Row from "primevue/row";
import Sidebar from "primevue/sidebar";
import Menu from "primevue/menu";
import Dialog from "primevue/dialog";
import Toast from "primevue/toast";
import Dropdown from "primevue/dropdown";
import Tag from "primevue/tag";
import Card from "primevue/card";

createApp(App)
    .use(router)
    .use(PrimeVue, { ripple: true })
    .use(ToastService)
    .component('pv-data-table', DataTable)
    .component("pv-column", Column)
    .component('pv-toolbar', Toolbar)
    .component('pv-input-text', InputText)
    .component('pv-textarea', Textarea)
    .component('pv-button', Button)
    .component('pv-row', Row)
    .component('pv-sidebar', Sidebar)
    .component('pv-menu', Menu)
    .component('pv-dialog', Dialog)
    .component('pv-toast', Toast)
    .component('pv-dropdown', Dropdown)
    .component('pv-tag', Tag)
    .component('pv-card', Card)
    .mount('#app')
```

Realizar el commit “**chore**: configure dependencies”

Implementamos sidebar y menu

Modificamos el **app.vue**

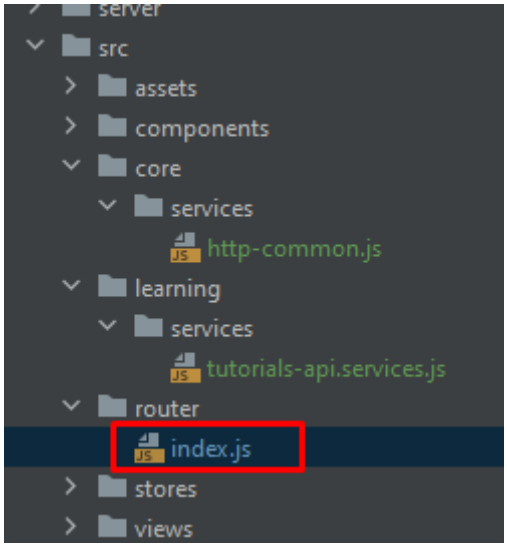
```
<script>
export default {
  data() {
    return {
      drawer: false,
      items: [
        { label: "Home", to: "/home" },
        { label: "About", to: "/about" },
      ],
    };
  },
};
</script>

<template>
  <pv-toast />
  <header>
    <pv-toolbar class="bg-primary">
      <template #start>
        <pv-button
            class="p-button-text text-white"
            icon="pi pi-bars"
            @click="drawer = !drawer"
        ></pv-button>
        <h3>ACME Learning Center</h3>
      </template>
      <template #end>
        <div class="flex-column">
          <router-link
              v-for="item in items"
              :to="item.to"
              custom
              v-slot="{ navigate, href }"
              :key="item.label"
          >
            <pv-button
                class="p-button-text text-white"
                :href="href"
                @click="navigate"
            >{{ item.label }}</pv-button
            >
          </router-link>
        </div>
      </template>
```

```
    </pv-toolbar>
  </header>
  <pv-sidebar v-model:visible="drawer"> </pv-sidebar>
  <RouterView />
</template>
```

Realizar el commit “**feat:** implemented configure menu and sidebar”

modificamos el **index.js**



aprovechando en definir components importando previamente o directamente en el routes

```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/home',
      name: 'home',
      component: HomeView,
    },
    {
      path: '/',
      redirect: 'home'
    },
    {
      path: '/about',
      name: 'about',
      // route level code-splitting
      // this generates a separate chunk (About.[hash].js) for this route
      // which is lazy-loaded when the route is visited.
```

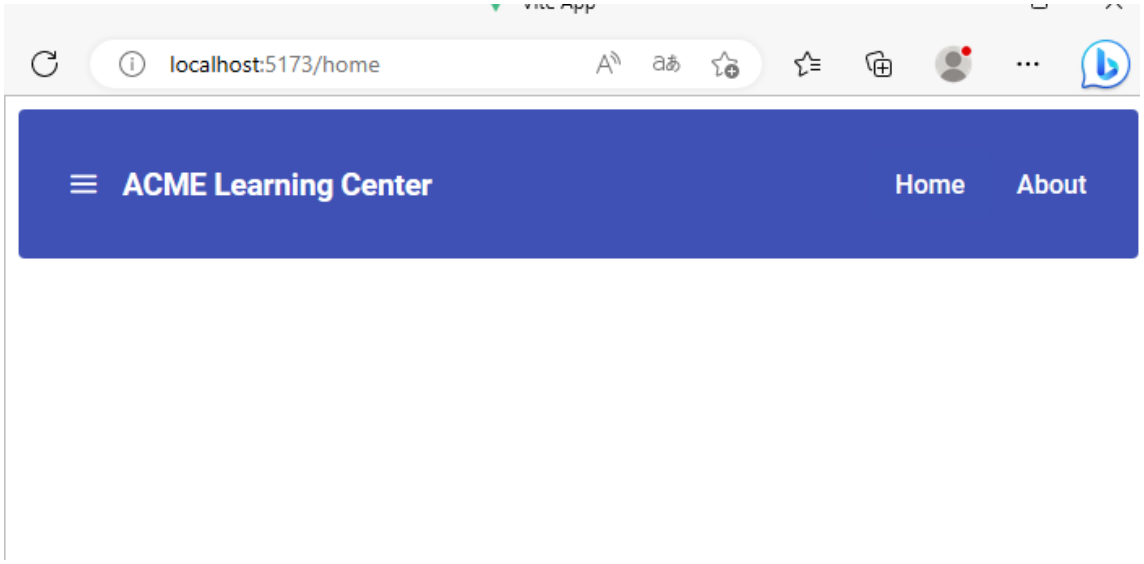
```
      component: () => import('../views/AboutView.vue')
    }
  ]
})

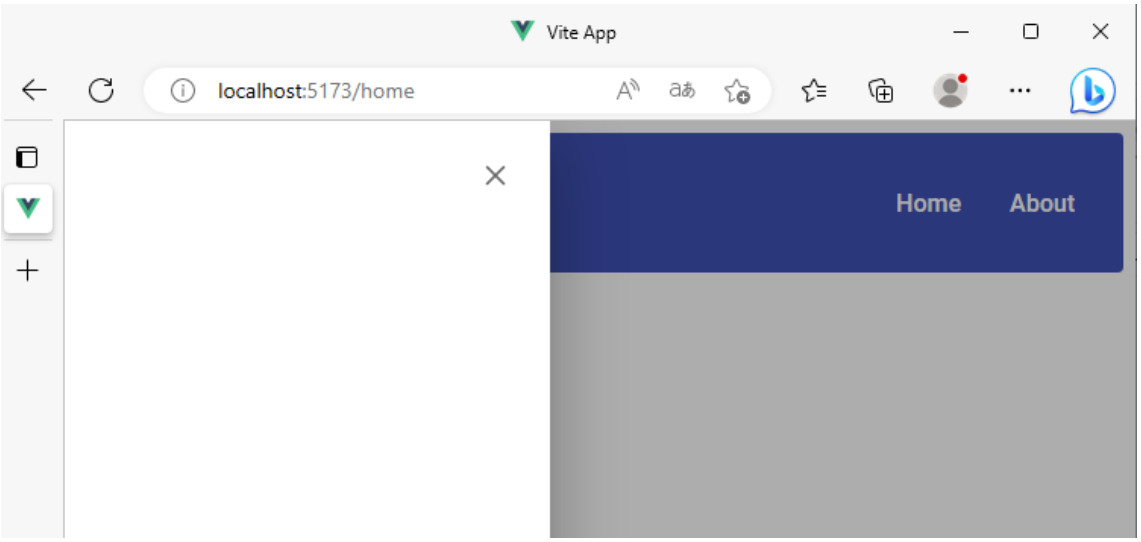
export default router
```

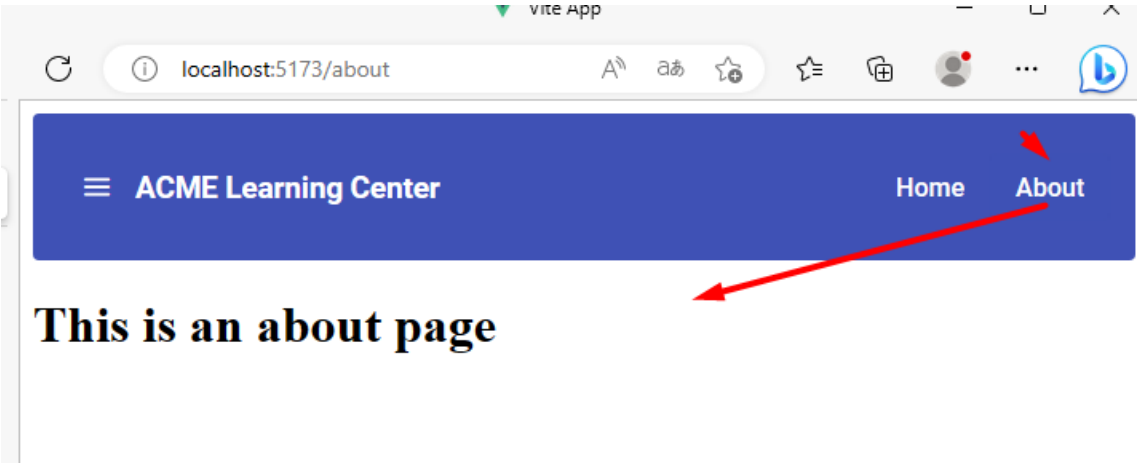
Realizar el commit “**feat:** implemented redirection to default page home”

npm run dev

se verá el sider bar y la pagina about funcionando



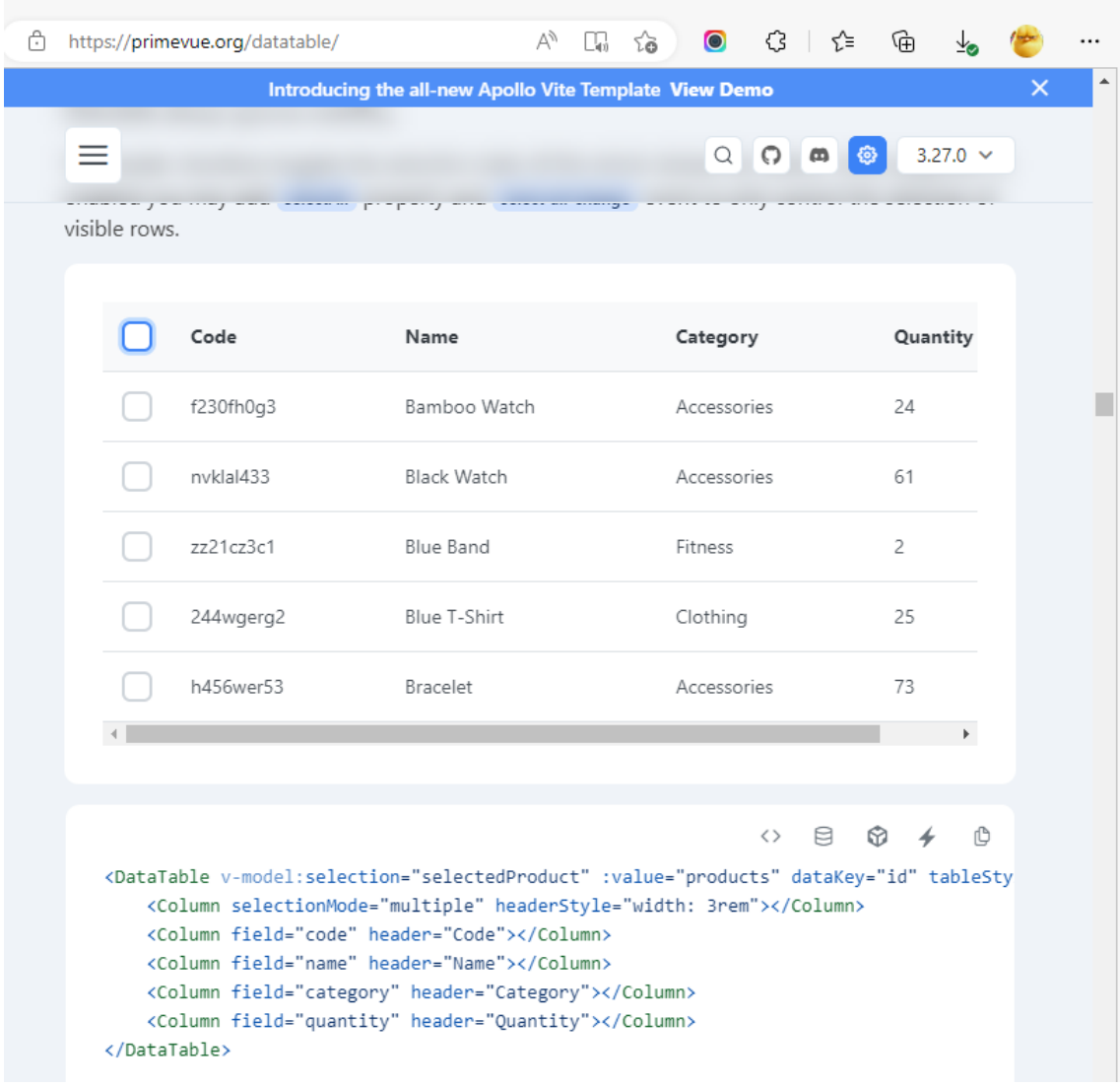




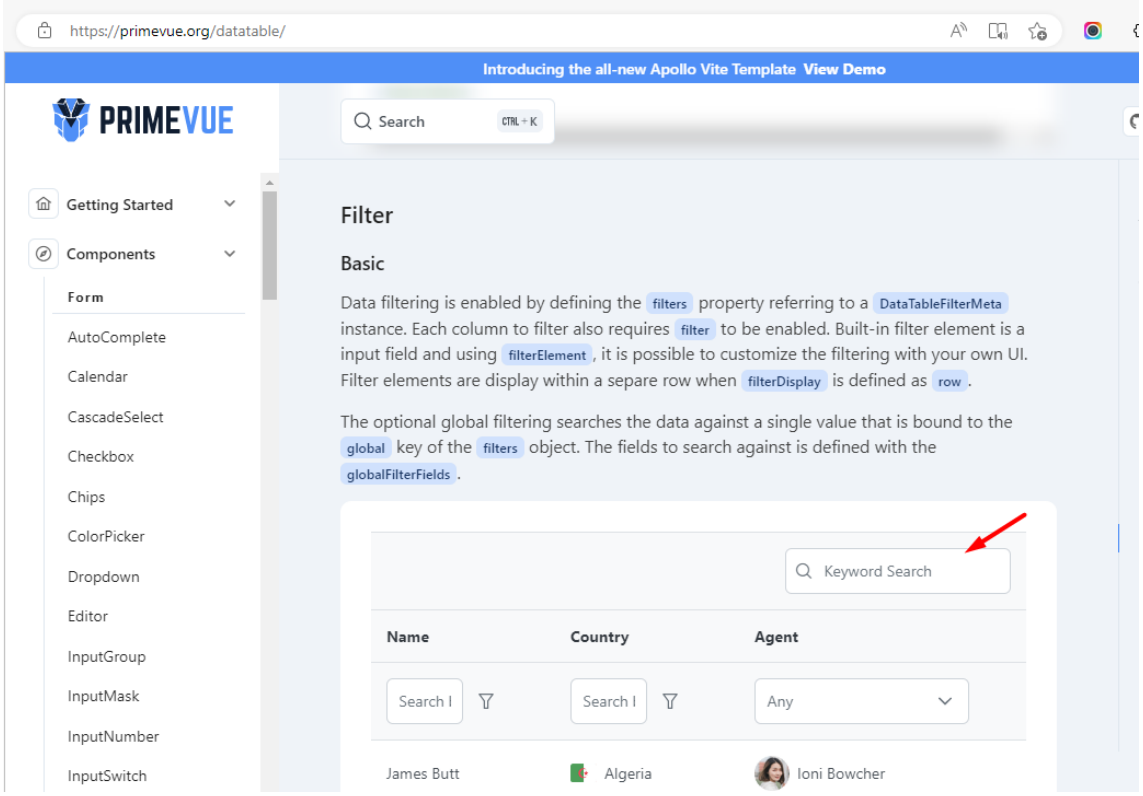
Implementar Data Table

Generamos el data table

Tomamos como ejemplo data table con select multiple



Para los filtros se toma como ejemplo



Implementar Data Table principal.

Generamos los botones crear, editar y eliminar.

Creando el componente **tutorial-list.component.vue**

```
<template>
  <div>
    <div class="card">
      <pv-toolbar class="mb-4">
        <template #start>
          <pv-button
              label="New"
              icon="pi pi-plus"
              class="p-button-success mr-2"
              @click=""
          />
          <pv-button
              label="Delete"
              icon="pi pi-trash"
              class="p-button-danger"
              @click=""
              :disabled="true"
          />
```

```
        </template>

        <template #end>
        </template>
      </pv-toolbar>

      <pv-data-table
          ref="dt"
          :value="tutorials"
          v-model:selection="selectedTutorials"
          dataKey="id"
          :paginator="true"
          :rows="10"
          paginatorTemplate="FirstPageLink PrevPageLink PageLinks
NextPageLink LastPageLink CurrentPageReport RowsPerPageDropdown"
          :rowsPerPageOptions="[5, 10, 25]"
          currentPageReportTemplate="Showing {first} to {last} of
{totalRecords} tutorials"
          responsiveLayout="scroll"
      >
        <template #header>
          <div class="table-header flex flex-column md:flex-row
md:justify-content-between">
              <h5 class="mb-2 md:m-0 p-as-md-center text-xl">Manage
Tutorials</h5>
          </div>
        </template>

        <pv-column
            selectionMode="multiple"
            style="width: 3rem"
            :exportable="false"
        ></pv-column>
        <pv-column
            field="id"
            header="Id"
            :sortable="true"
            style="min-width: 12rem"
        ></pv-column>
        <pv-column
            field="title"
            header="Title"
            :sortable="true"
            style="min-width: 16rem"
        ></pv-column>
        <pv-column
            field="description"
            header="Description"
            :sortable="true"
```

```
            style="min-width: 16rem"
        ></pv-column>
        <pv-column
            field="status"
            header="Status"
            :sortable="true"
            style="min-width: 12rem"
        >
            <template #body="slotProps">
                <pv-tag v-if="slotProps.data.status === 'Published'"
severity="success">
                    {{ slotProps.data.status }}
                </pv-tag>
                <pv-tag v-else severity="info">{{ slotProps.data.status
}}</pv-tag>
            </template>
        </pv-column>
        <pv-column :exportable="false" style="min-width: 8rem">
            <template #body="slotProps">
                <pv-button
                    icon="pi pi-pencil"
                    class="p-button-text p-button-rounded"
                    @click=""
                />
                <pv-button
                    icon="pi pi-trash"
                    class="p-button-text p-button-rounded"
                    @click=""
                />
            </template>
        </pv-column>
      </pv-data-table>
    </div>

  </div>
</template>

<script>
import { TutorialsApiService } from "../services/tutorials-api.service";

export default {
  name: "tutorial-list",
  data() {
    return {
      tutorials: [],
      tutorial: {},
      selectedTutorials: null,
      statuses: [
        { label: "Published", value: "published" },
```

```
        { label: "Unpublished", value: "unpublished" },
      ],
      tutorialsService: null,
    };
  },
  created() {
    this.tutorialsService = new TutorialsApiService();
    this.tutorialsService.getAll()
        .then((response) => {
            this.tutorials = response.data;
            console.log(this.tutorials);
            this.tutorials.forEach(
                (tutorial) => this.getDisplayableTutorial(tutorial)
            );
        console.log(this.tutorials);
    });
  },

  methods: {
    getDisplayableTutorial(tutorial) {
      tutorial.status = tutorial.published ? this.statuses[0].label :
this.statuses[1].label;
      return tutorial;
    }
  },
};
</script>

<style lang="scss" scoped>
.table-header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  @media screen and (max-width: 960px) {
    align-items: start;
  }
}

@media screen and (max-width: 960px) {
  ::v-deep(.p-toolbar) {
    flex-wrap: wrap;
    .p-button {
      margin-bottom: 0.25rem;
    }
  }
}
</style>
```

Realizar el commit “**feat:** implemented tutorial-list.component”

Actualizar routes a tutorials.

Debemos actualizar las router/index.js

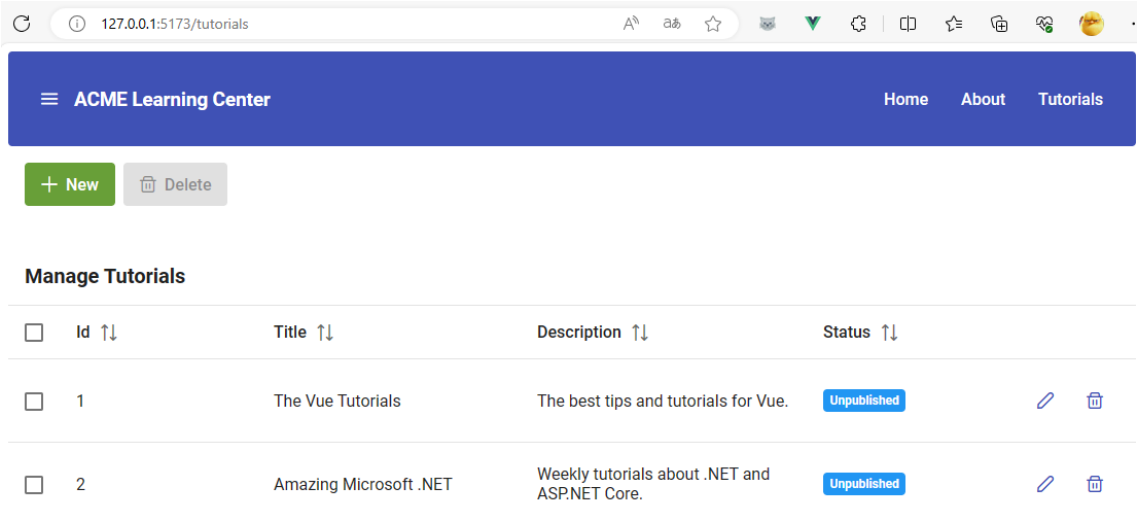
```
,
    {
      path: "/tutorials",
      name: "tutorials",
      // route level code-splitting
      // this generates a separate chunk (About.[hash].js) for this route
      // which is lazy-loaded when the route is visited.
      component: () => import("../learning/pages/tutorial-
list.component.vue"),
    },
```

Debemos actualizar las app.vue

```
          { label: "Home", to: "/home" },
          { label: "About", to: "/about" },
          { label: 'Tutorials', to: '/tutorials' }
```

Realizar el commit “**feat:** implemented direction to page tutorial”

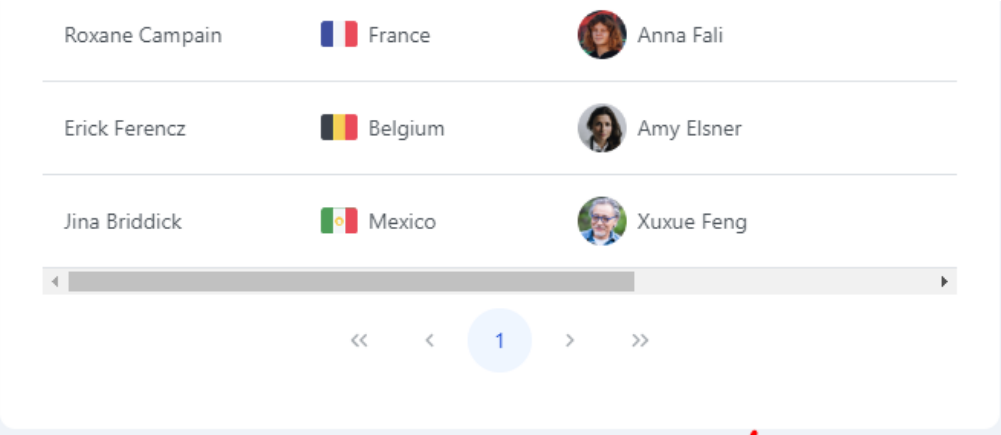
npm run dev



Implementar filtro a la bandeja (data table).

Implementamos el filter del datatable

Usando la definición de en primevue



Composition API Options API

```
<template>
    <div class="card">
        <DataTable v-model:filters="filters" :value="customers" paginator :rows="
                :globalFilterFields="['name', 'country.name', 'representative.nam
            <template #header>
                <div class="flex justify-content-end">
                    <span class="p-input-icon-left">
                        <i class="pi pi-search" />
                        <InputText v-model="filters['global'].value" placeholder=
                    </span>
                </div>
```

```
export default {
    data() {
        return {
            customers: null,
            filters: {
                global: { value: null, matchMode: FilterMatchMode.CONTAINS },
                name: { value: null, matchMode: FilterMatchMode.STARTS_WITH },
                'country.name': { value: null, matchMode: FilterMatchMode.STARTS_
                representative: { value: null, matchMode: FilterMatchMode.IN },
                status: { value: null, matchMode: FilterMatchMode.EQUALS },
                verified: { value: null, matchMode: FilterMatchMode.EQUALS }
            },
            representatives: [
```

modificar el componente **tutorial-list.component.vue**

Adicionando

```
            :filters="filters"
```

Resultado en el pv-data-table

```
        <pv-data-table
            ref="dt"
            :value="tutorials"
            v-model:selection="selectedTutorials"
            dataKey="id"
            :paginator="true"
            :rows="10"
            :filters="filters"
```

Se modifica el **`<template #header>`**

```
          <template #header>
            <div class="table-header flex flex-column md:flex-row
md:justify-content-between">
              <h5 class="mb-2 md:m-0 p-as-md-center text-xl">Manage
Tutorials</h5>
              <span class="p-input-icon-left">
                <i class="pi pi-search" />
                <pv-input-text
                    v-model="filters['global'].value"
                    placeholder="Search..."
                />
              </span>
            </div>
          </template>
```

Importamos el

```
import { FilterMatchMode } from "primevue/api";
```

resultando asi

```
<script>
import { TutorialsApiService } from "../services/tutorials-api.service";
import { FilterMatchMode } from "primevue/api";
```

adicionamos

```
filters: {},
```

quedando asi

```
export default {
  name: "tutorial-list",
  data() {
    return {
      tutorials: [],
      tutorial: {},
      selectedTutorials: null,
      filters: {},
```

modificamos el

```
created()
```

adicionando

```
this.initFilters();
```

quedando de la siguiemte manera.

```
  created() {
    this.tutorialsService = new TutorialsApiService();
    this.tutorialsService.getAll()
        .then((response) => {
            this.tutorials = response.data;
            console.log(this.tutorials);
            this.tutorials.forEach(
                (tutorial) => this.getDisplayableTutorial(tutorial)
            );
        console.log(this.tutorials);
    });
    this.initFilters();
  },
```

Geneamos un método que permita settear el valor de la búsqueda y el modo de busqueda.

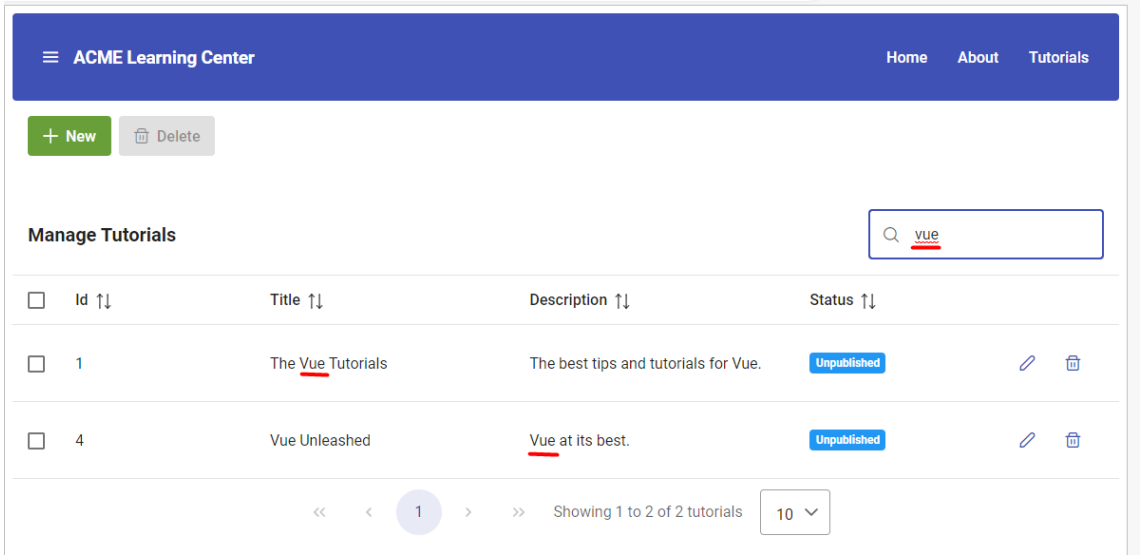
```
    initFilters() {
      this.filters = {
        global: { value: null, matchMode: FilterMatchMode.CONTAINS },
      };
    }
```

Quedando los métodos

```
  methods: {
    getDisplayableTutorial(tutorial) {
      tutorial.status = tutorial.published ? this.statuses[0].label :
this.statuses[1].label;
      return tutorial;
    },
    initFilters() {
      this.filters = {
        global: { value: null, matchMode: FilterMatchMode.CONTAINS },
      };
    }
  },
};
```

Realizar el commit “**feat:** implemented filters”

El resultado luego de ejecutar el **npm run dev** en el terminal



Implementar exportar a la bandeja.

Modificar el componente **tutorial-list.component.vue**

Implementar el exportar del datatable

Modificamos **pv-toolbar** , adicionando el botón exportar

```
        <template #end>
          <pv-button
              label="Export"
              icon="pi pi-download"
              class="p-button-help"
              @click="exportToCSV($event)"
          />
        </template>
      </pv-toolbar>
```

Debemos generar el **`exportToCSV`**

```
methods
```

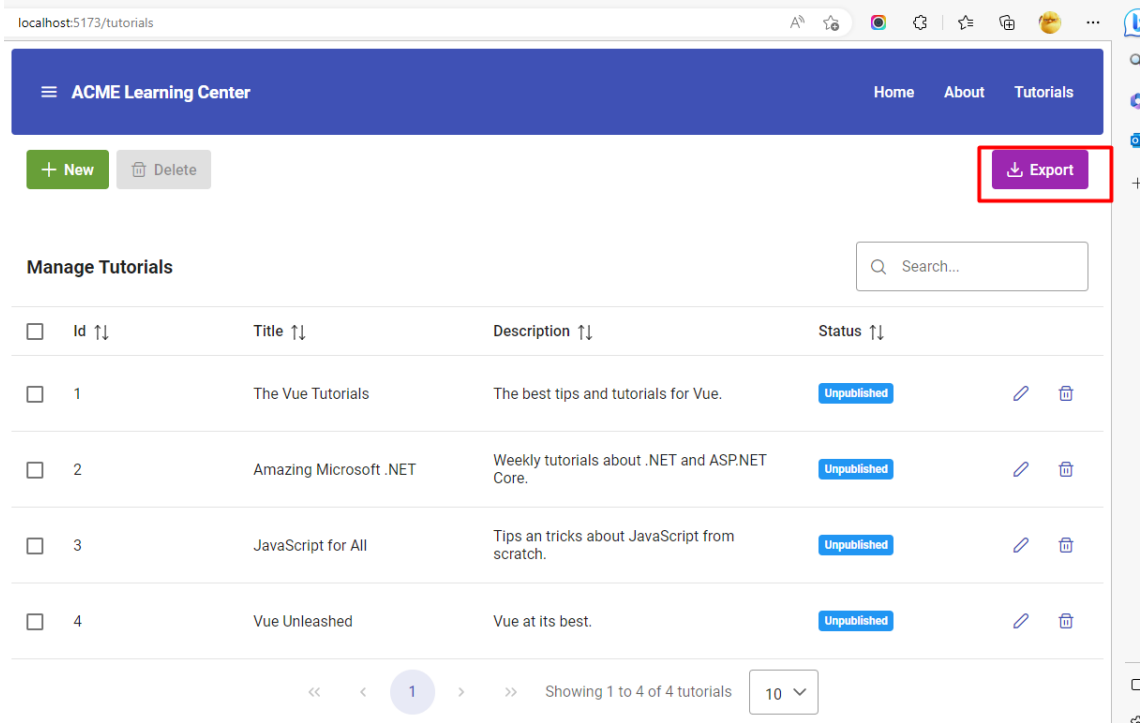
```
    exportToCSV() {
      this.$refs.dt.exportCSV();
    },
```

Quedando de la siguiente manera

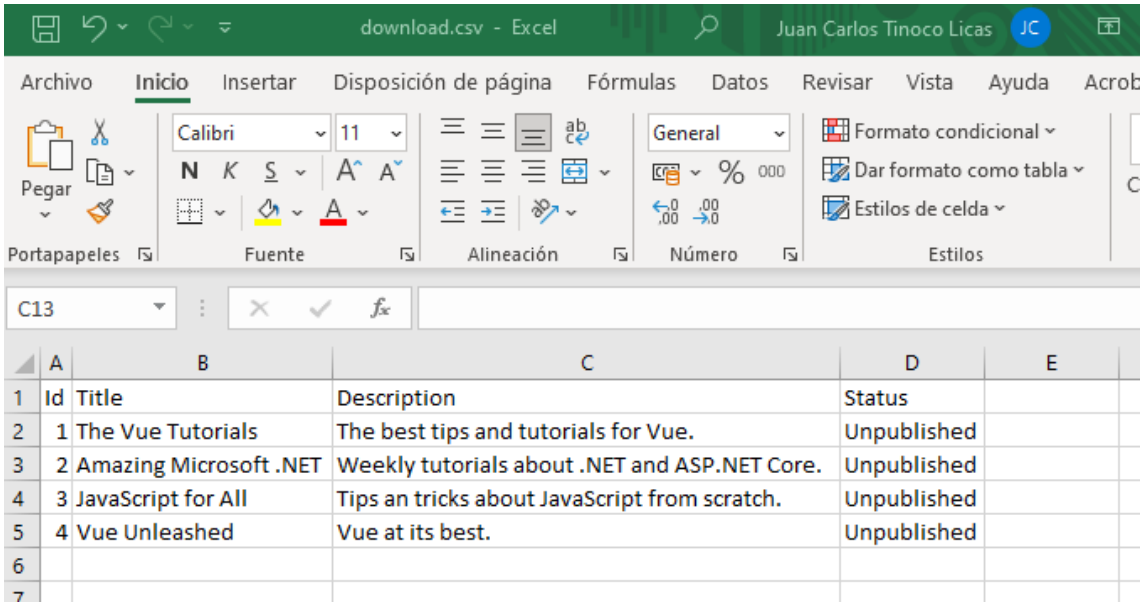
```
  methods: {
    getDisplayableTutorial(tutorial) {
      tutorial.status = tutorial.published ? this.statuses[0].label :
this.statuses[1].label;
      return tutorial;
    },
    exportToCSV() {
      this.$refs.dt.exportCSV();
    },
    initFilters() {
      this.filters = {
        global: { value: null, matchMode: FilterMatchMode.CONTAINS },
      };
    }
  },
```

Realizar el commit “**feat:** implemented export”

Quedando de la siguiente manera.



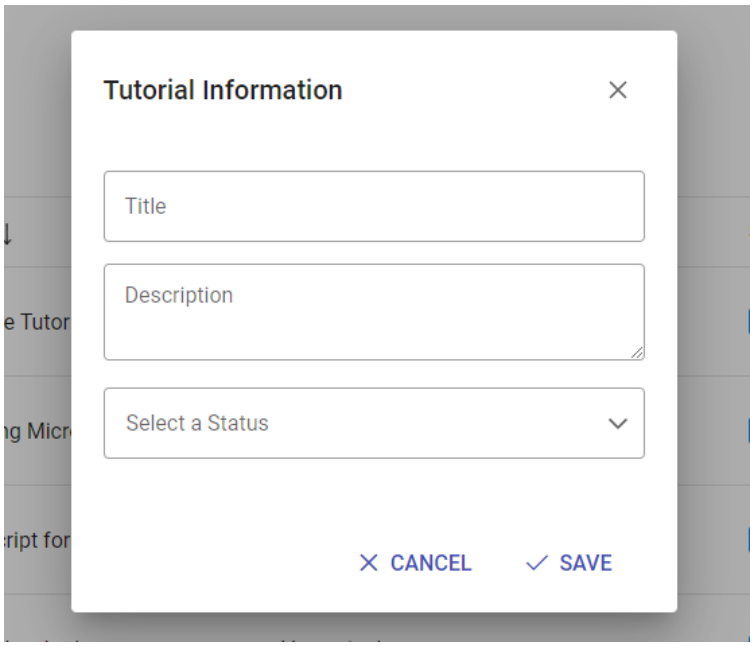
Se genera el formato



Implementar crear nuevo en la bandeja.

Implementar el botón crear nuevo elemento.

Creamos el dialog



Modificar el componente **tutorial-list.component.vue**

Debemos adicionar las variables que me permitirán manejar el dialog

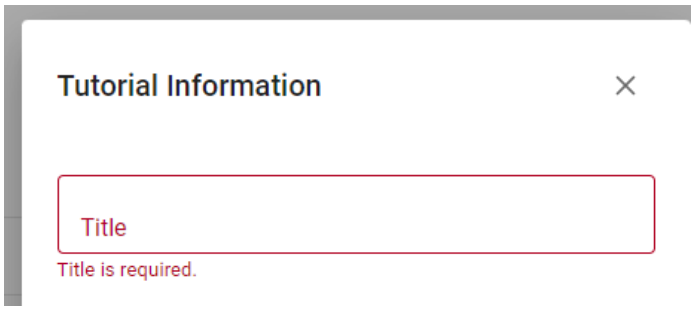
Para ellos adicionamos dos variables.

Permitirá ver o esconder el dialog de registro.

```
      tutorialDialog: false,
```

Permitira el ver el error de un título nulo al momento de iniciar el guardar el registro tutorial (tutorial: {})

```
      submitted: false,
```



finalmente quedara asi:

```
export default {
  name: "tutorial-list",
  data() {
    return {
      tutorials: [],
      tutorialDialog: false,
      tutorial: {},
      selectedTutorials: null,
      filters: {},
      submitted: false,
```

Generamos los metodos ;

- **getStorableTutorial** : Permite procesar el elemento tutorial a tutorial para visualizar.
- **openNew** : limpia la variable que tendrá a tutorial y bloqueará los botones al cambiar el valor de submitted y mostrará el dialog de registro.
- **hideDialog** : esconde el dialos
- **saveTutorial**: inicia el grabado de tutorial, si existe un title en el sabe tutorial

Se adiciona los metos en los method

```
    getStorableTutorial(displayableTutorial) {
      return {
        id: displayableTutorial.id,
        title: displayableTutorial.title,
        description: displayableTutorial.description,
        published: displayableTutorial.status.label === "Published",
      };
    },
    openNew() {
      this.tutorial = {};
      this.submitted = false;
      this.tutorialDialog = true;
    },
    hideDialog() {
      this.tutorialDialog = false;
      this.submitted = false;
    },
    saveTutorial() {
      this.submitted = true;
      if (this.tutorial.title.trim()) {
          this.tutorial.id = 0;
          console.log(this.tutorial);
          this.tutorial = this.getStorableTutorial(this.tutorial);
          this.tutorialsService
              .create(this.tutorial)
              .then((response) => {
```

```
                this.tutorial =
this.getDisplayableTutorial(response.data);
                this.tutorials.push(this.tutorial);
                this.$toast.add({
                  severity: "success",
                  summary: "Successful",
                  detail: "Tutorial Created",
                  life: 3000,
                });
                console.log(response);
              });
        this.tutorialDialog = false;
        this.tutorial = {};
      }
    },
```

Adicionamos el dialog que también servirá para modificar los registros.

Lo ubicamos luego del class=”card”

```
1   <template>
2     <div>
3 >     <div class="card">···
98      </div>
99      <pv-dialog
10        v-model:visible="tutorialDialog"
11        :style="{ width: '450px' }"
12        header="Tutorial Information"
13        :modal="true"
14        class="p-fluid"
```

```
    <pv-dialog
        v-model:visible="tutorialDialog"
        :style="{ width: '450px' }"
        header="Tutorial Information"
        :modal="true"
        class="p-fluid"
    >
      <div class="field mt-3">
        <span class="p-float-label">
          <pv-input-text
              type="text"
              id="title"
              v-model.trim="tutorial.title"
              required="true"
              autofocus
              :class="{ 'p-invalid': submitted && !tutorial.title }"
          />
```

```
          <label for="title">Title</label>
          <small class="p-error" v-if="submitted && !tutorial.title">
            Title is required.
          </small>
        </span>
      </div>

      <div class="field">
        <span class="p-float-label">
          <pv-textarea
              id="description"
              v-model="tutorial.description"
              required="false"
              rows="2"
              cols="20"
          />
          <label for="description">Description</label>
        </span>
      </div>
      <div class="field">
        <pv-dropdown
            id="published"
            v-model="tutorial.status"
            :options="statuses"
            optionLabel="label"
            placeholder="Select a Status"
        >
          <template #value="slotProps">
            <div v-if="slotProps.value && slotProps.value.value">
              <span :class="'tutorial-badge status-' +
slotProps.value.value">
                {{ slotProps.value.label}}
              </span>
            </div>
            <div v-else-if="slotProps.value && !slotProps.value.value">
              <span :class=" 'tutorial-badge status-' +
slotProps.value.toLowerCase() ">
                {{ slotProps.value }}
              </span>
            </div>
            <span v-else>
              {{ slotProps.placeholder }}
            </span>
          </template>
        </pv-dropdown>
      </div>
      <template #footer>
        <pv-button
            :label="'Cancel'.toUpperCase()"
```

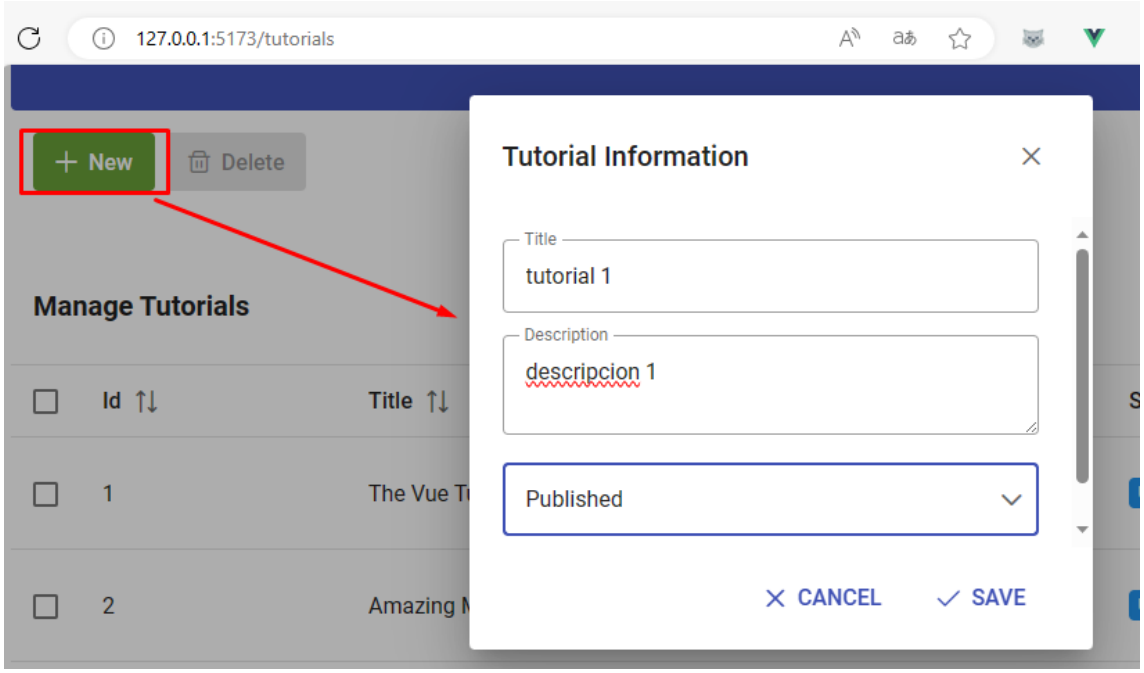
```
            icon="pi pi-times"
            class="p-button-text"
            @click="hideDialog"
        />
        <pv-button
            :label="'Save'.toUpperCase()"
            icon="pi pi-check"
            class="p-button-text"
            @click="saveTutorial"
        />
      </template>
    </pv-dialog>
```

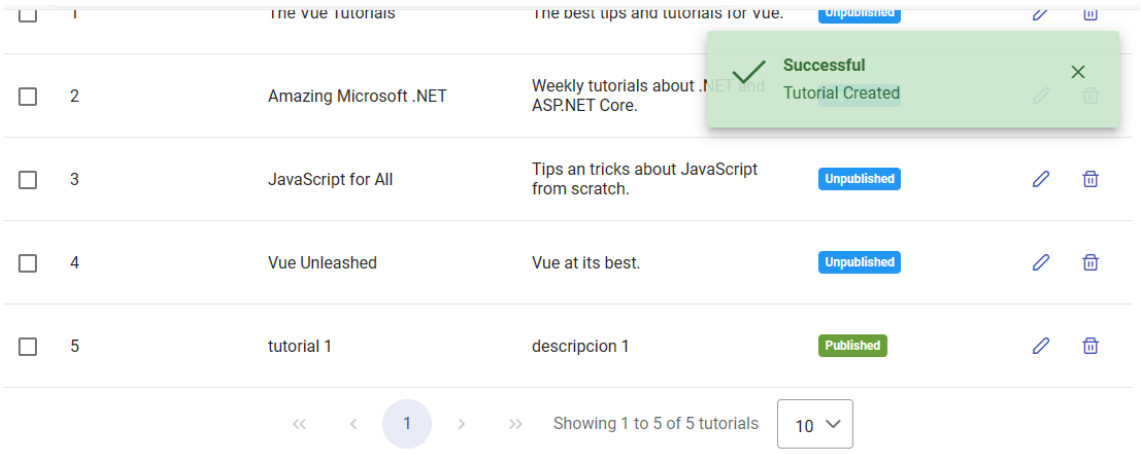
Modificamos el botón new, adicionando el método openNew, como sigue

```
        <template #start>
          <pv-button
              label="New"
              icon="pi pi-plus"
              class="p-button-success mr-2"
              @click="openNew"
          />
```

Realizar el commit "**feat:** implemented new tutorial"

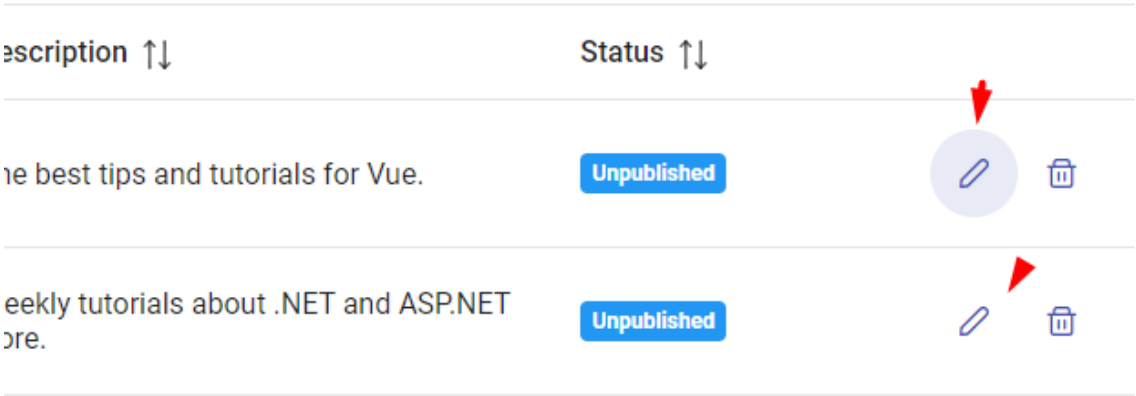
npm run dev





Implementar Modificar tutorial.

Implementamos el botón editar



Se va a reutilizar el dialog de creación, con las diferencias que se le seteara los valores del tutorial a modificar.

Debemos crear y modificar lo siguiente metodos:

- findIndexById : permitirá encontrar los valores a modificar desde el servicio web
- saveTutorial : permitirá grabar en caso sea una edición, considerando que si tiene id es una edición.
- editTutorial: se genera una nueva variable tutorial y se visualiza el dialog.

Adicionar en los method definidos.

```
    findIndexById(id) {
      console.log(`current id: ${id}`);
      return this.tutorials.findIndex((tutorial) => tutorial.id === id);
    },
```

```
    saveTutorial() {
      this.submitted = true;
      if (this.tutorial.title.trim()) {
        if (this.tutorial.id) {
          console.log(this.tutorial);
          this.tutorial = this.getStorableTutorial(this.tutorial);
          this.tutorialsService
              .update(this.tutorial.id, this.tutorial)
              .then((response) => {
                console.log(response.data.id);
                this.tutorials[this.findIndexById(response.data.id)] =
                    this.getDisplayableTutorial(response.data);
                this.$toast.add({
                  severity: "success",
                  summary: "Successful",
                  detail: "Tutorial Updated",
                  life: 3000,
                });
                console.log(response);
              });
        } else {
          this.tutorial.id = 0;
          console.log(this.tutorial);
          this.tutorial = this.getStorableTutorial(this.tutorial);
          this.tutorialsService
              .create(this.tutorial)
              .then((response) => {
                this.tutorial =
this.getDisplayableTutorial(response.data);
                this.tutorials.push(this.tutorial);
                this.$toast.add({
                  severity: "success",
                  summary: "Successful",
                  detail: "Tutorial Created",
                  life: 3000,
                });
                console.log(response);
              });
        }
        this.tutorialDialog = false;
        this.tutorial = {};
      }
    },
```

```
    editTutorial(tutorial) {
      console.log(tutorial);
      this.tutorial = { ...tutorial };
      console.log(this.tutorial);
      this.tutorialDialog = true;
    },
```

Asociamos el método editar a la fila del datatable. (icon="pi pi-pencil")

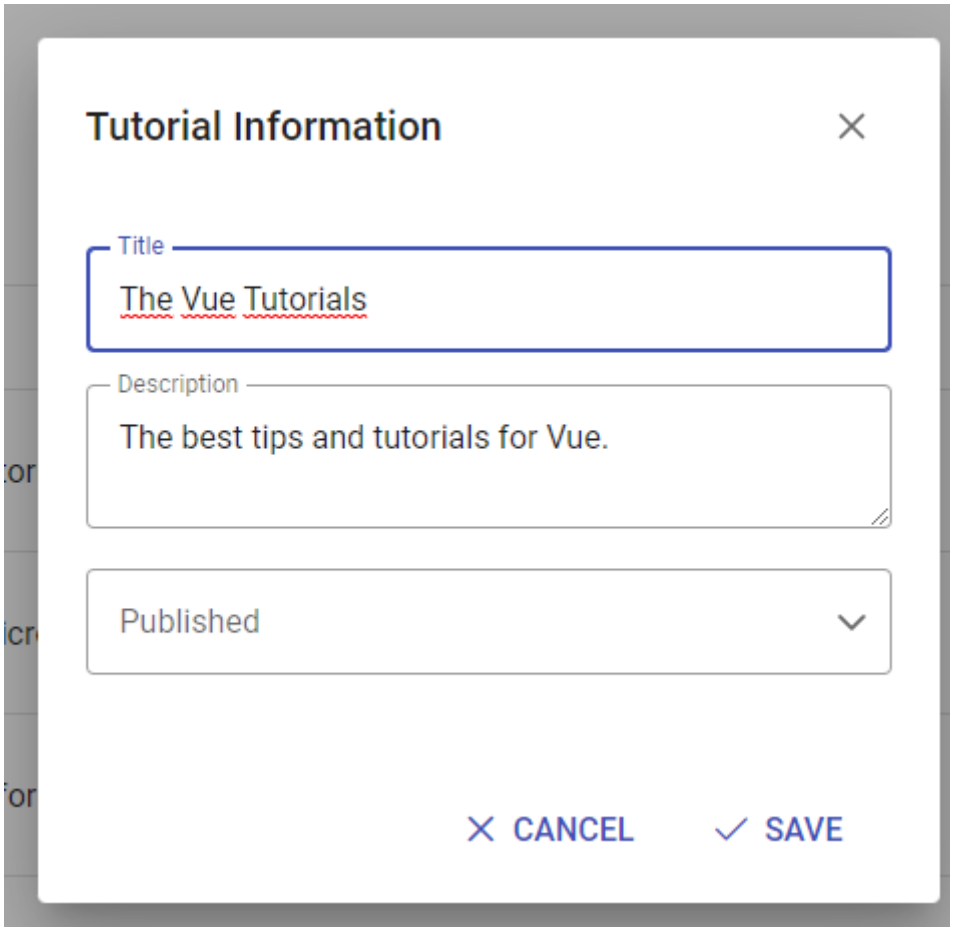
```
                    @click="editTutorial(slotProps.data)"
```

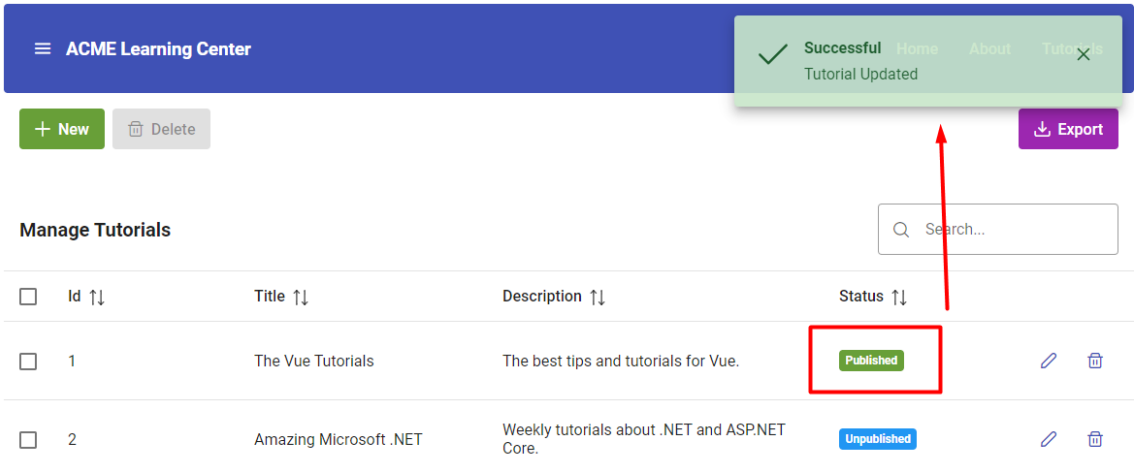
Resultando lo siguiente:

```
        <pv-column :exportable="false" style="min-width: 8rem">
            <template #body="slotProps">
                <pv-button
                    icon="pi pi-pencil"
                    class="p-button-text p-button-rounded"
                    @click="editTutorial(slotProps.data)"
                />
```

Realizar el commit “**feat:** implemented modify tutorial”

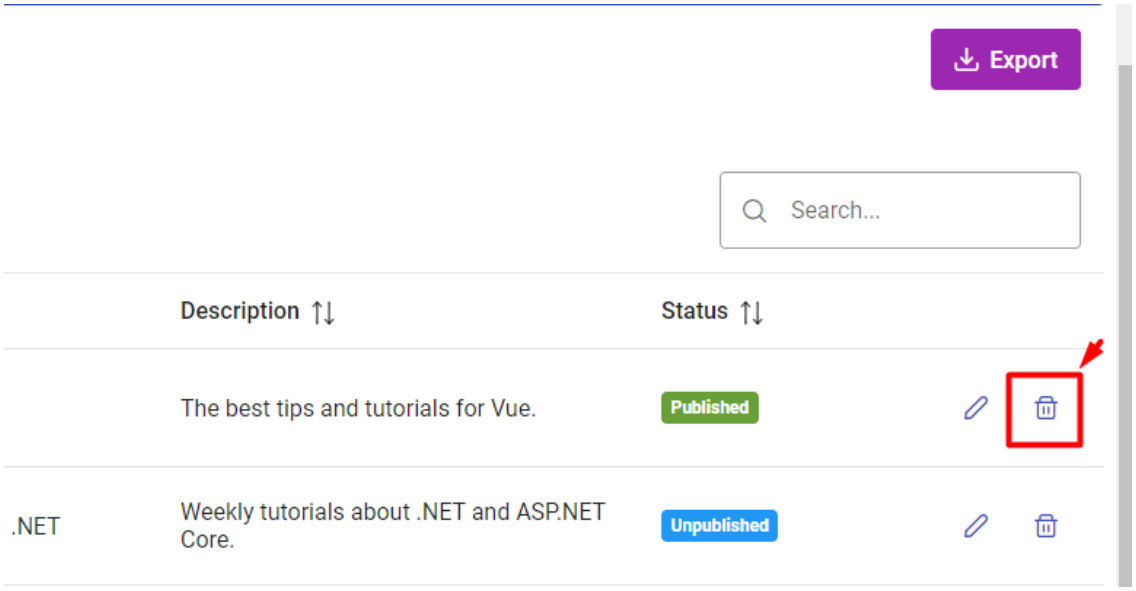
Finalmente se podra editar:



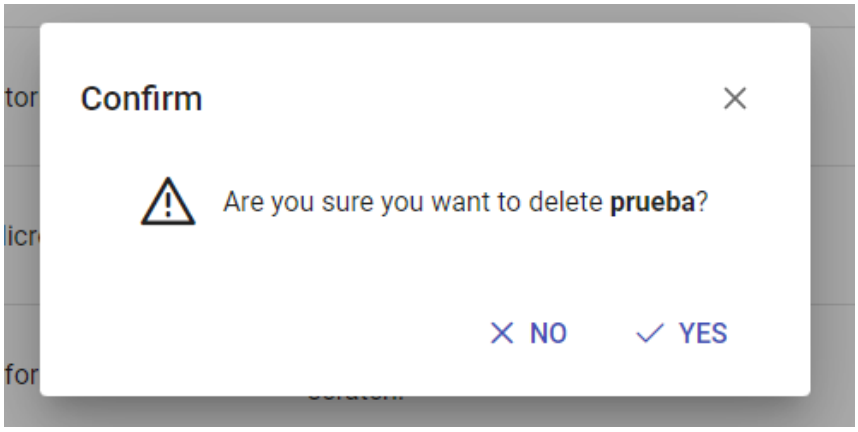


Implementar Eliminar tutorial.

Implementar el eliminar un registro tutorial.



Visualmente solo necesitamos el dialog de confirmación.



Generamos la variable que permitir controlar la visualización del dialog.

```
deleteTutorialDialog: false,
```

quedaría asi :

```
  name: "tutorial-list",
  data() {
    return {
      tutorials: [],
      tutorialDialog: false,
      deleteTutorialDialog: false,
      tutorial: {},
      selectedTutorials: null,
```

Debemos adicionar métodos en el method

- confirmDeleteTutorial : permite visualizar el dialog de confirmación.
- deleteTutorial : ejecuta el borrado y eliminar “filtra” el id eliminado.

Se adiciona los siguientes métodos:

```
    confirmDeleteTutorial(tutorial) {
      this.tutorial = tutorial;
      this.deleteTutorialDialog = true;
    },
    deleteTutorial() {
      this.tutorialsService.delete(this.tutorial.id).then((response) => {
        this.tutorials = this.tutorials.filter(
            (t) => t.id !== this.tutorial.id
        );
        this.deleteTutorialDialog = false;
        this.tutorial = {};
        this.$toast.add({
          severity: "success",
          summary: "Successful",
          detail: "Tutorial Deleted",
          life: 3000,
        });
        console.log(response);
      });
    },
```

Adicionamos el siguiente estilo

```
.confirmation-content {
  display: flex;
  align-items: center;
  justify-content: center;
}
```

Adicionamos el dialog de confirmación.

```
    <pv-dialog
        v-model:visible="deleteTutorialDialog"
        :style="{ width: '450px' }"
        header="Confirm"
        :modal="true"
    >
      <div class="confirmation-content">
        <i class="pi pi-exclamation-triangle mr-3" style="font-size:
2rem" />
        <span v-if="tutorial">
            Are you sure you want to delete <b>{{ tutorial.title }}</b>?
        </span>
      </div>
      <template #footer>
        <pv-button
            :label="'No'.toUpperCase()"
            icon="pi pi-times"
            class="p-button-text"
            @click="deleteTutorialDialog = false"
        />
        <pv-button
            :label="'Yes'.toUpperCase()"
            icon="pi pi-check"
            class="p-button-text"
            @click="deleteTutorial"
        />
      </template>
    </pv-dialog>
```

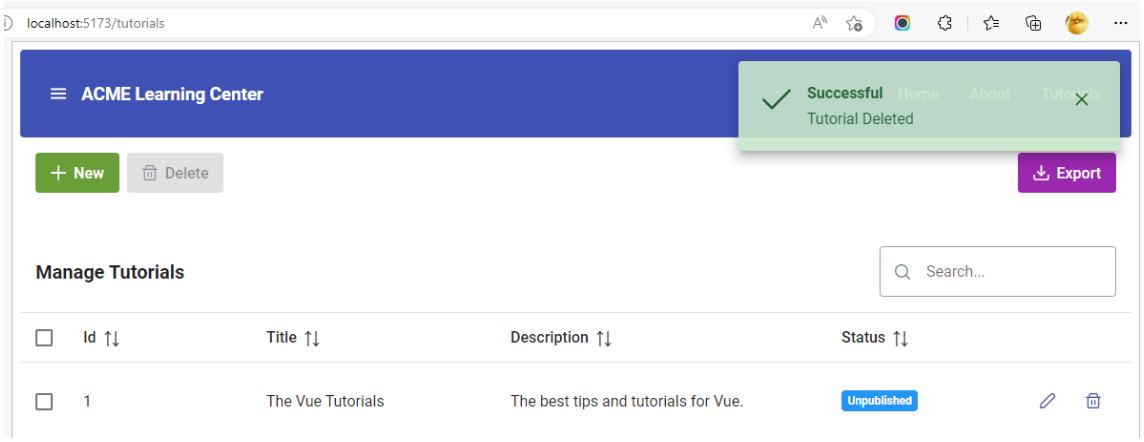
Modificamos el botón eliminar adicionando :

```
@click="confirmDeleteTutorial(slotProps.data)"
```

Quedando de la siguiente manera:

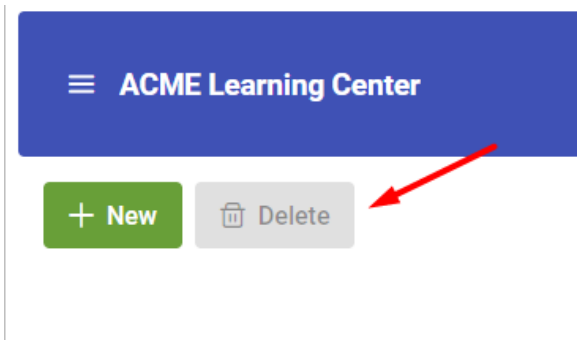
```
                <pv-button
                    icon="pi pi-trash"
                    class="p-button-text p-button-rounded"
                    @click="confirmDeleteTutorial(slotProps.data)"
                />
```

Realizar el commit “**feat:** implemented delete tutorial”

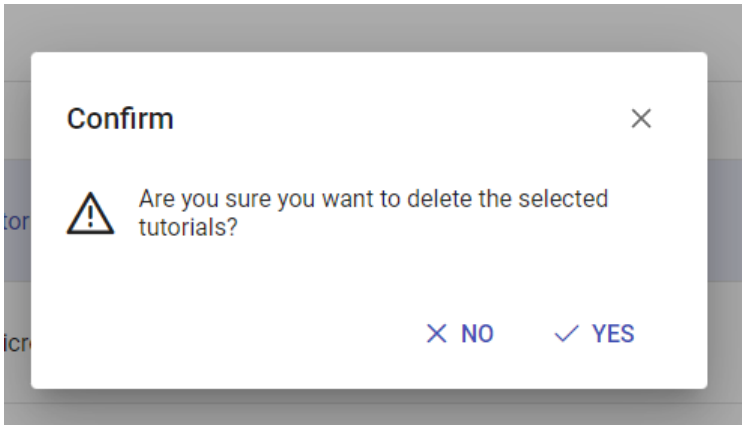


Implementar Eliminar Masivo de tutoriales.

Implementamos el borrado masivo.



Visualmente solo necesitamos el dialog de confirmación.



Generamos la variable que permitir controlar la visualización del dialog.

```
deleteTutorialsDialog: false,
```

quedaría asi :

```
  name: "tutorial-list",
  data() {
    return {
      tutorials: [],
      tutorialDialog: false,
      deleteTutorialDialog: false,
      deleteTutorialsDialog: false,
      tutorial: {},
      selectedTutorials: null,
```

Debemos adicionar métodos en el method

- confirmDeleteSelected : permite visualizar el dialog de confirmación.
- deleteSelectedTutorials : ejecuta el borrado y eliminar “filtra” el id eliminado.

Se adiciona los siguientes métodos:

```
    confirmDeleteSelected() {
      this.deleteTutorialsDialog = true;
    },
    deleteSelectedTutorials() {
      this.selectedTutorials.forEach((tutorial) => {
        this.tutorialsService.delete(tutorial.id)
          .then((response) => {
            this.tutorials = this.tutorials.filter(
              (t) => t.id !== tutorial.id
            );
            console.log(response);
          });
      });
      this.deleteTutorialsDialog = false;
    },
```

Adicionamos el dialog de confirmación.

```
    <pv-dialog
        v-model:visible="deleteTutorialsDialog"
        :style="{ width: '450px' }"
        header="Confirm"
        :modal="true"
    >
      <div class="confirmation-content">
        <i class="pi pi-exclamation-triangle mr-3" style="font-size:
2rem" />
        <span v-if="tutorial">
          Are you sure you want to delete the selected tutorials?
        </span>
      </div>
      <template #footer>
        <pv-button
            :label="'No'.toUpperCase()"
            icon="pi pi-times"
            class="p-button-text"
            @click="deleteTutorialsDialog = false"
        />
        <pv-button
            :label="'Yes'.toUpperCase()"
            icon="pi pi-check"
            class="p-button-text"
            @click="deleteSelectedTutorials"
        />
      </template>
    </pv-dialog>
```

Modificamos el botón eliminar adicionando:

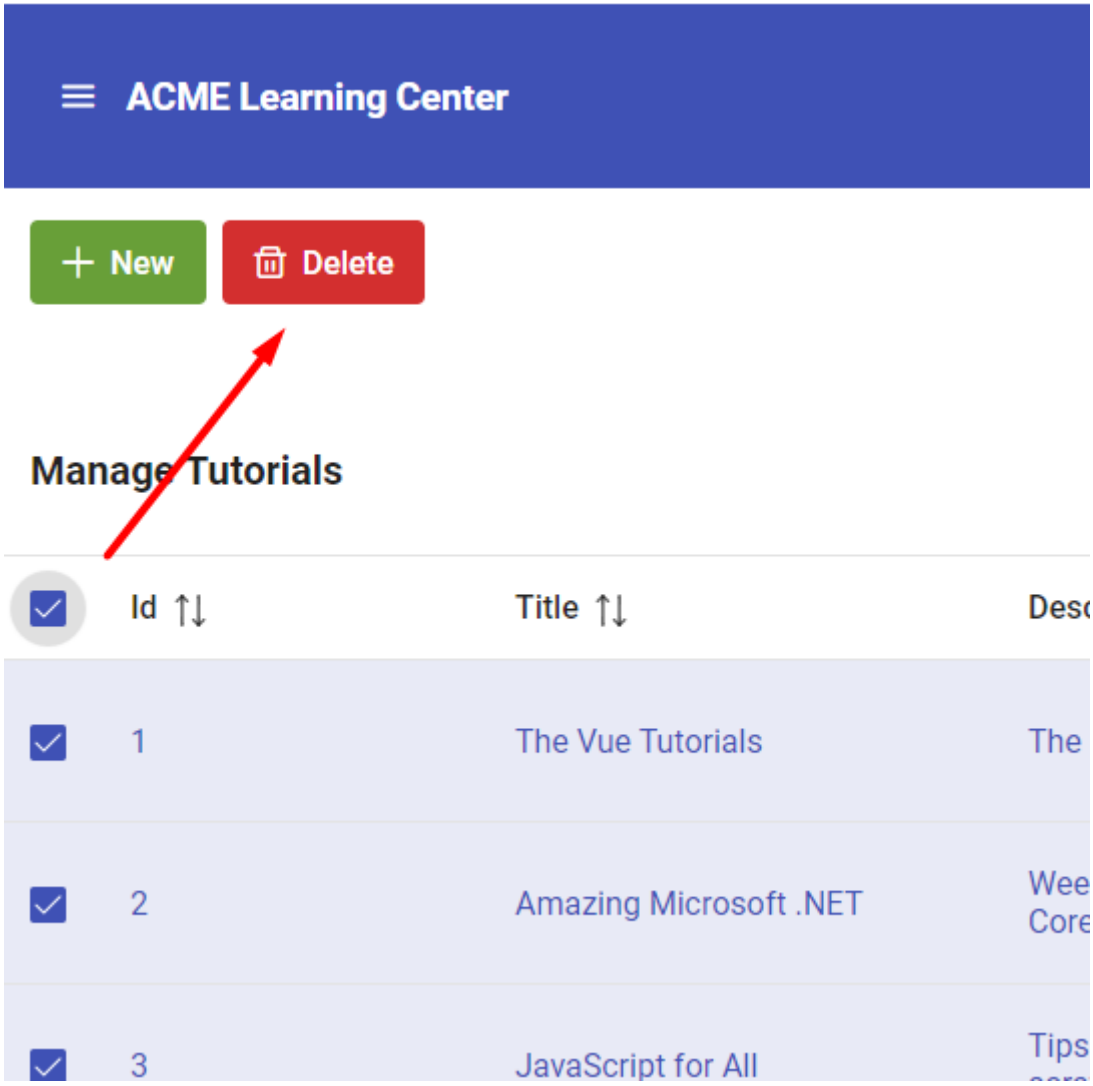
```
@click="confirmDeleteSelected"
:disabled="!selectedTutorials || !selectedTutorials.length"
```

Quedando de la siguiente manera:

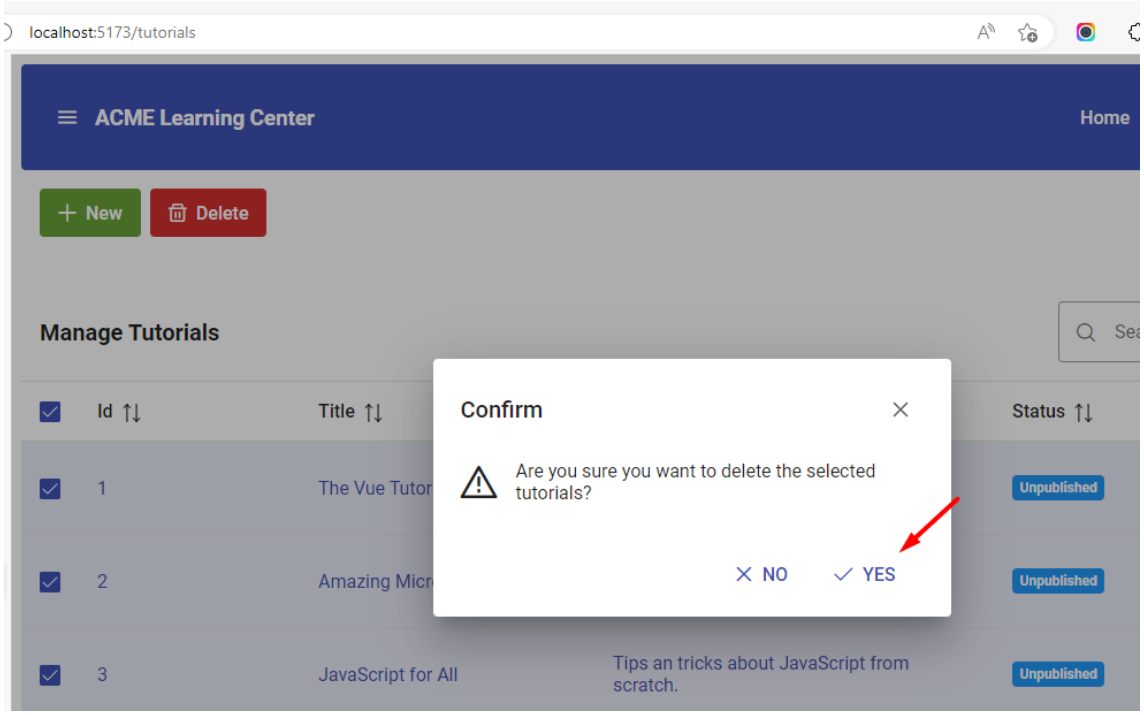
```
          <pv-button
              label="Delete"
              icon="pi pi-trash"
              class="p-button-danger"
              @click="confirmDeleteSelected"
              :disabled="!selectedTutorials || !selectedTutorials.length"
          />
```

Realizar el commit “**feat:** implemented delete tutorials”

Cambiara de color al seleccionar a todos los elementos



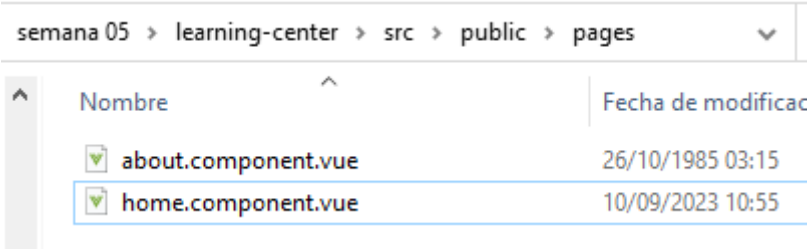
Clic en borrado



Corregir componentes de router.

Modificar el directorio view a public/pages

Actualizar los nombres del componente según estándar.



Realizar el commit “**chore:** update about and home components”

Conocimientos previos.

Precondiciones debes tener instalado el node

Defincion de la directiva v-on

v-on en Vue.js es una directiva que se utiliza para escuchar eventos DOM y vincularlos a métodos o expresiones definidas en el componente. Permite que el componente reaccione a eventos del usuario, como clics, cambios de entrada, etc. La directiva **v-on** se usa comúnmente con modificadores como **@click**, **@input**, **@submit**, entre otros, para capturar y manejar eventos específicos. Por ejemplo, **@click="handleClick"** ejecutará el método **handleClick** cuando se haga clic en un elemento HTML

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.

Estandar de commit

- **feat**: cuando se añade una nueva funcionalidad.
- **fix**: cuando se arregla un error.
- **chore**: tareas rutinarias que no sean específicas de una feature o un error como por ejemplo añadir contenido al fichero .gitignore o instalar una dependencia.
- **test**: si añadimos o arreglamos tests.
- **docs**: cuando solo se modifica documentación.
- **build**: cuando el cambio afecta al compilado del proyecto.
- **ci**: el cambio afecta a ficheros de configuración y scripts relacionados con la integración continua.
- **style**: cambios de legibilidad o formateo de código que no afecta a funcionalidad.

- **refactor**: cambio de código que no corrige errores ni añade funcionalidad, pero mejora el código.
- **perf**: usado para mejoras de rendimiento.
- **revert**: si el commit revierte un commit anterior. Debería indicarse el hash del commit que se revierte.

Componente del Prime face

Ripple

https://primefaces.org/primevue/ripple

Getting Started

Ripple

Ripple is an optional animation for the supported components such as buttons. It is disabled by default and needs to be enabled at your app's entry file (e.g. main.js) during the PrimeVue setup.

```
import {createApp} from 'vue';
import PrimeVue from 'primevue/config';
const app = createApp(App);

app.use(PrimeVue, {ripple: true});
```

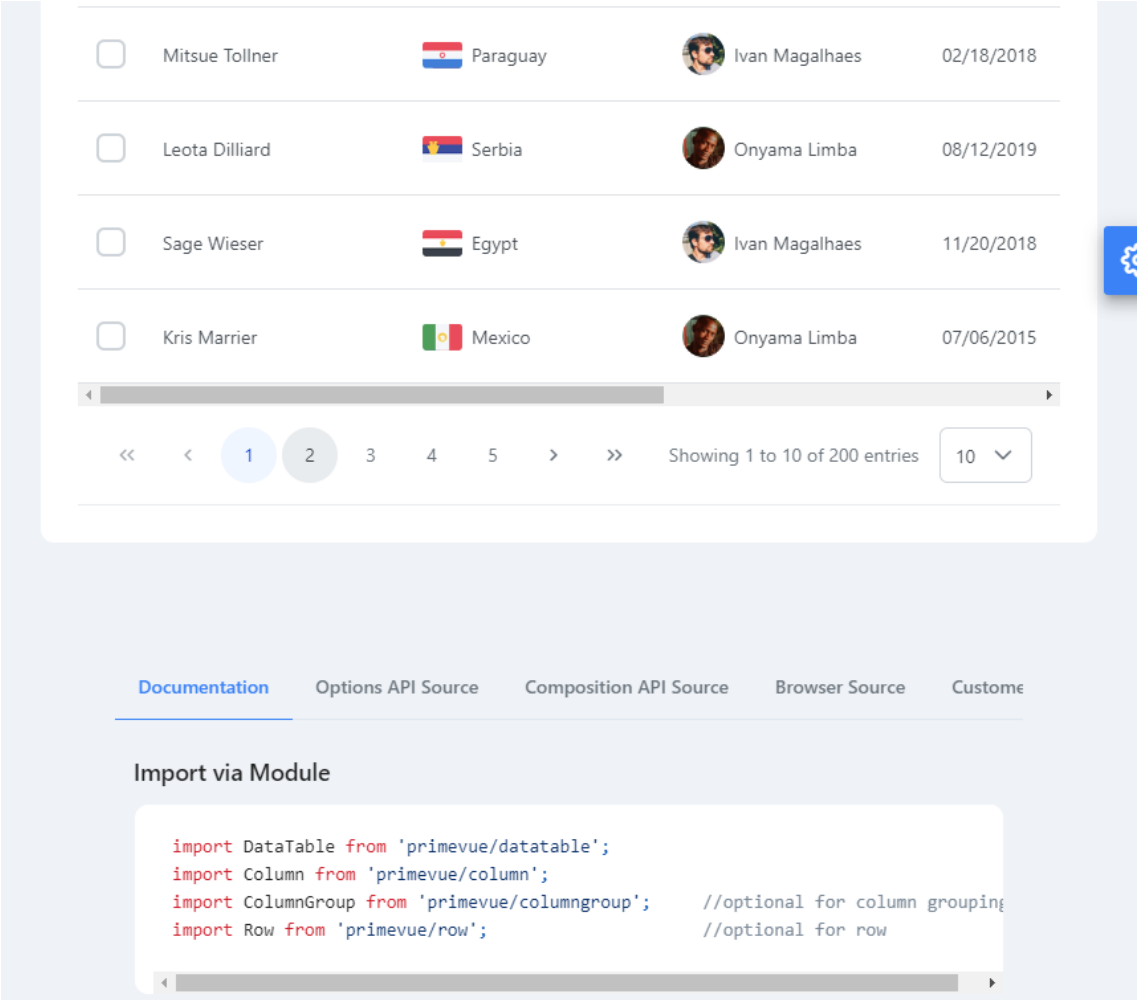
Note: That would be it to enable ripple on PrimeVue components, next section describes how to use it with your own components and standard elements.

PrimeVue is a rich set of open source native components for Vue.
PrimeFlex is a lightweight responsive CSS utility library to accompany Prime UI libraries and static webpages as well.

Datatable

import DataTable from "primevue/datatable";
import Column from "primevue/column";
import Row from "primevue/row";

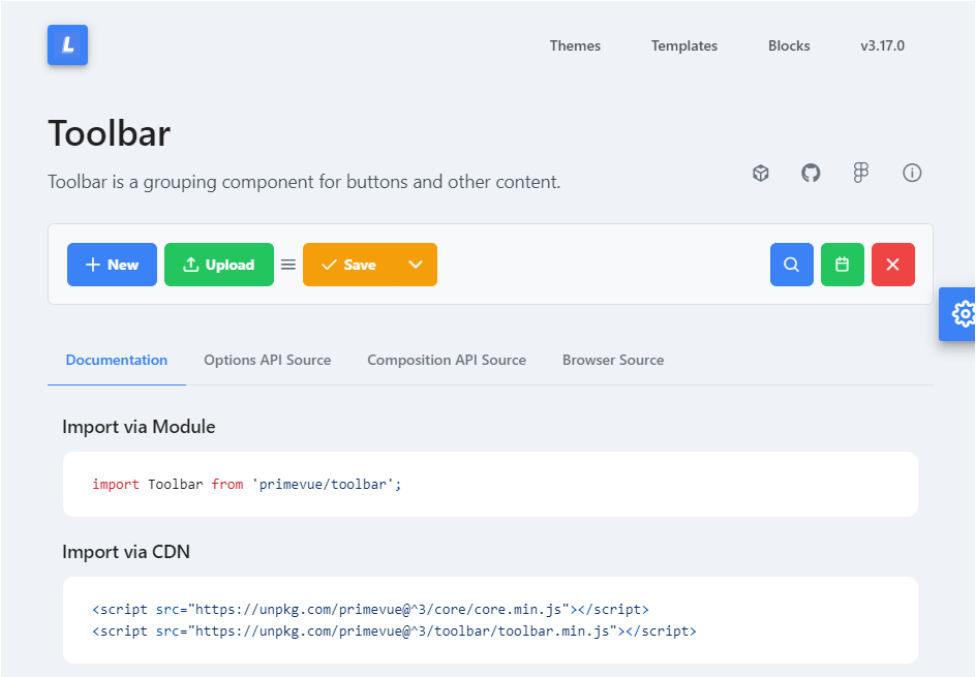
https://primefaces.org/primevue/datatable



Toobar

import Toolbar from "primevue/toolbar";

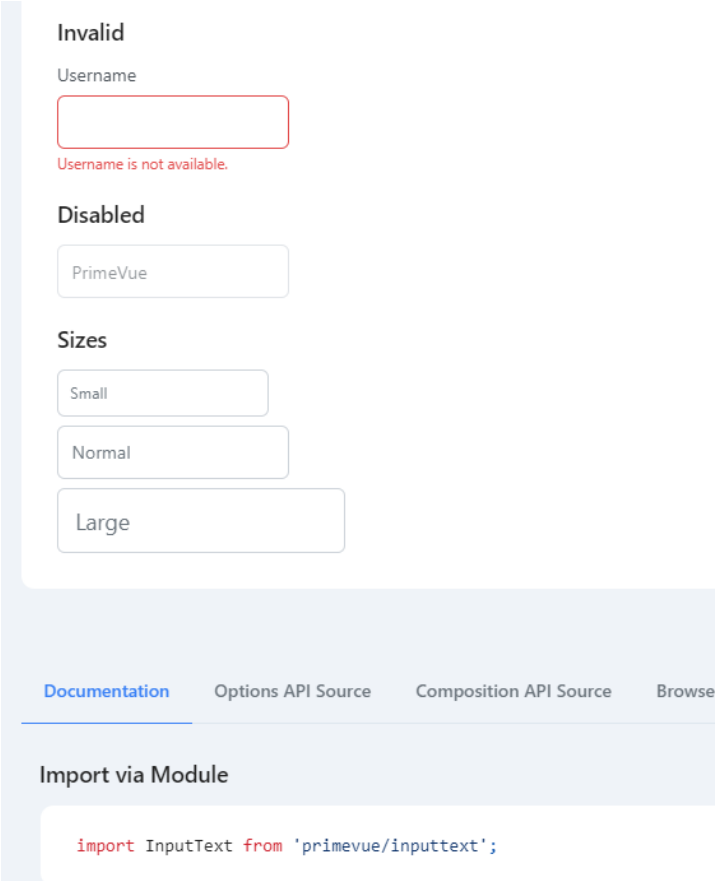
https://primefaces.org/primevue/toolbar



inputtext

import InputText from "primevue/inputtext";

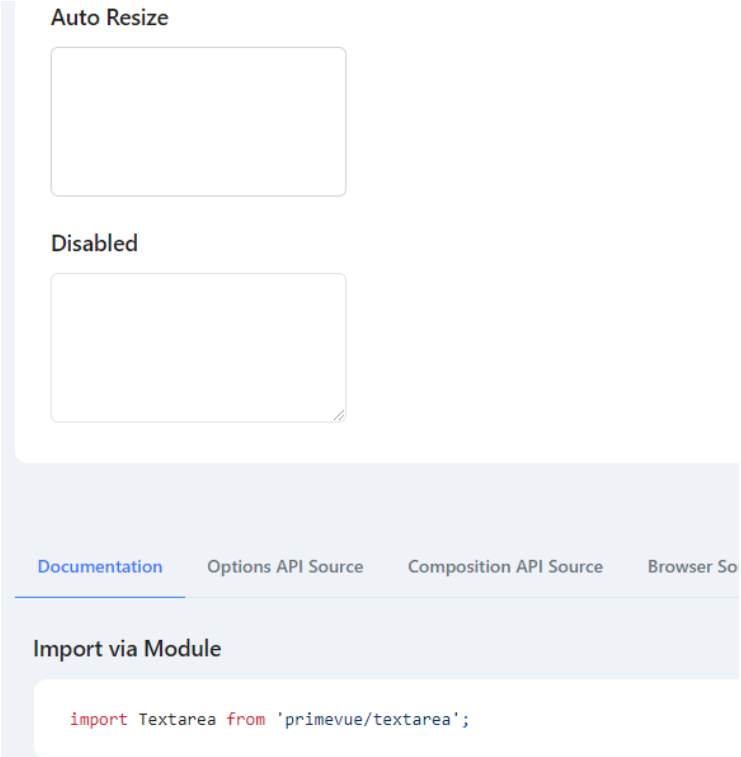
https://primefaces.org/primevue/inputtext



Textarea

import Textarea from "primevue/textarea";

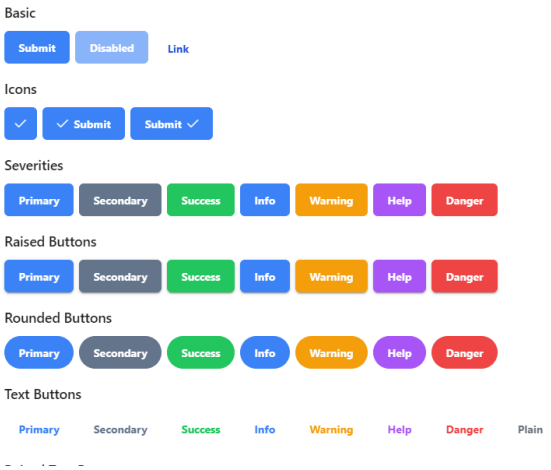
https://primefaces.org/primevue/textarea



button

import Button from "primevue/button";

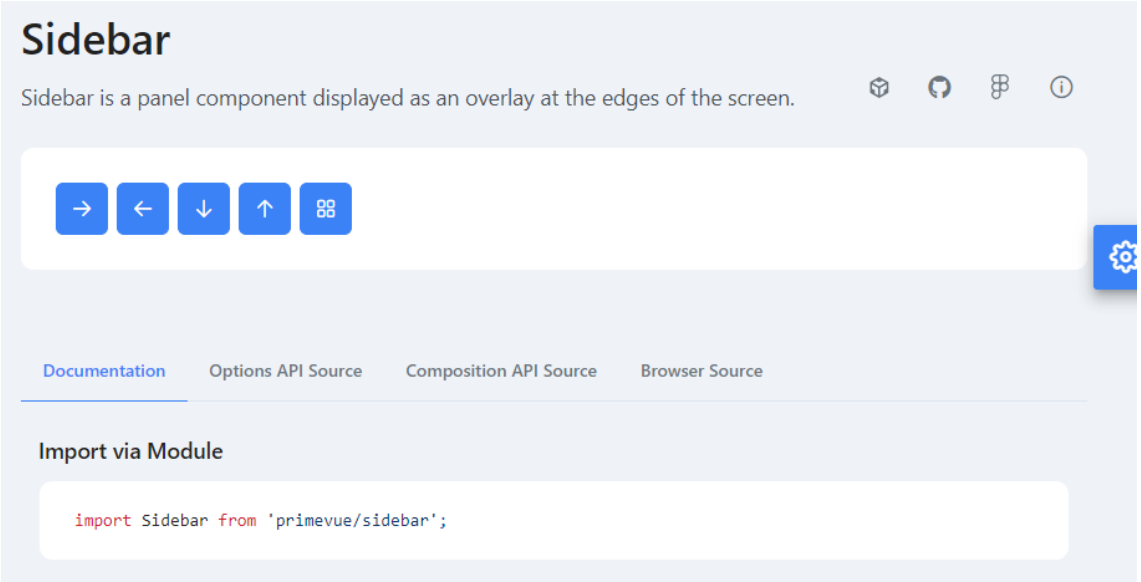
https://primefaces.org/primevue/button



siderbar

https://primefaces.org/primevue/sidebar

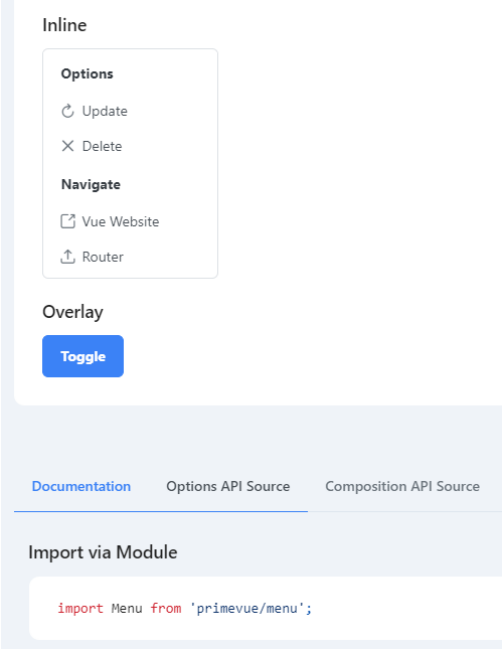
import Sidebar from "primevue/sidebar";



Menu

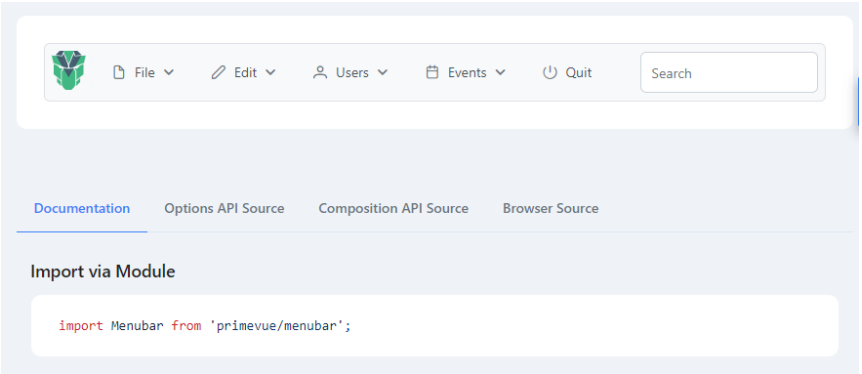
import Menu from "primevue/menu";

http://primefaces.org/primevue/menu



Menubar

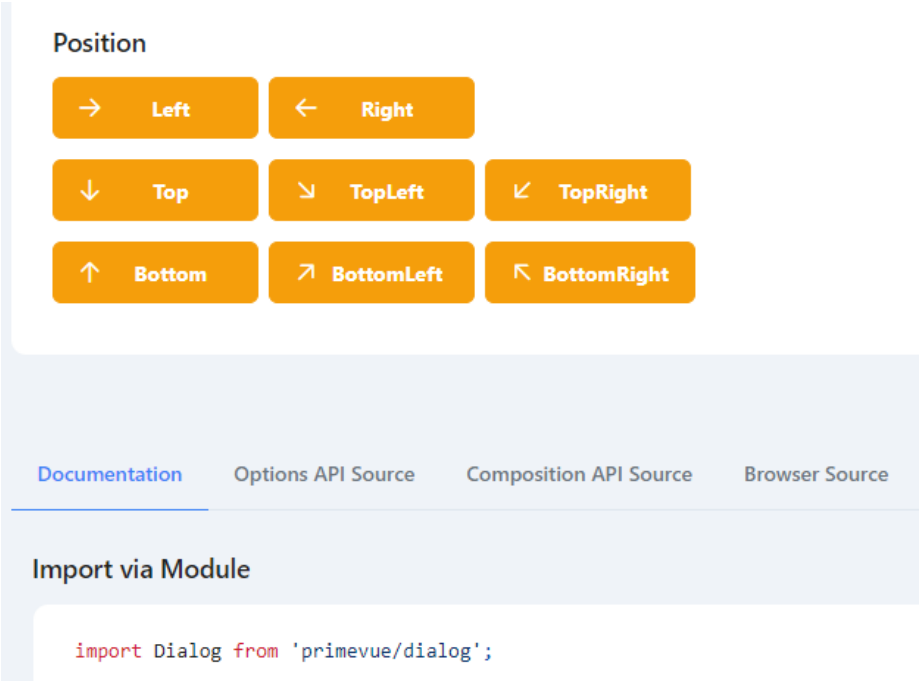
https://primefaces.org/primevue/menubar



dialog

import Dialog from "primevue/dialog";

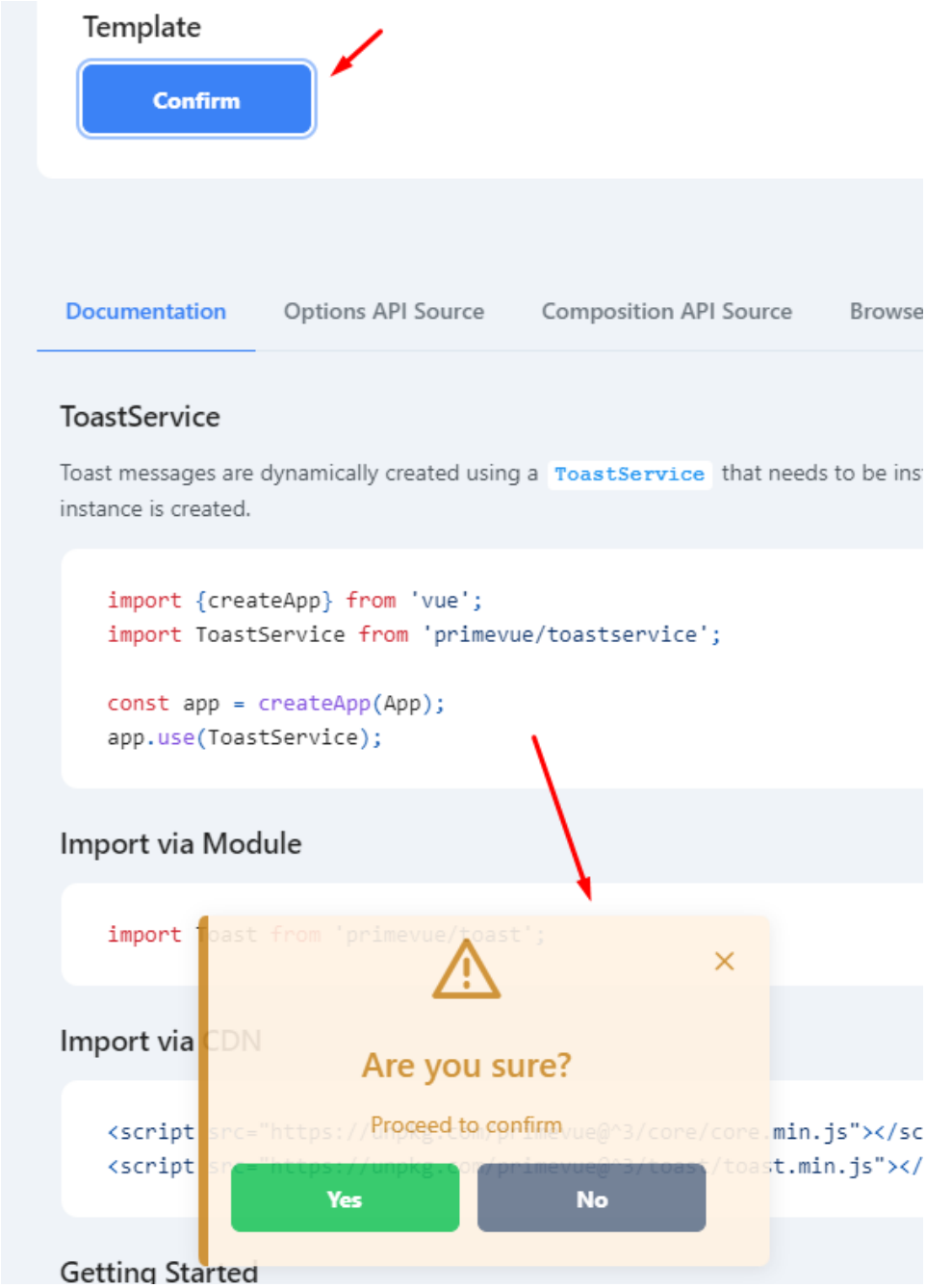
https://primefaces.org/primevue/dialog



toast

import Toast from "primevue/toast";

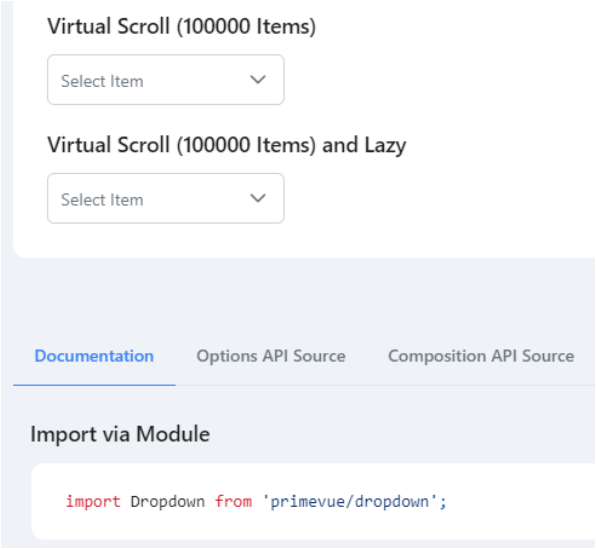
https://primefaces.org/primevue/toast



Dropdown

import Dropdown from "primevue/dropdown";

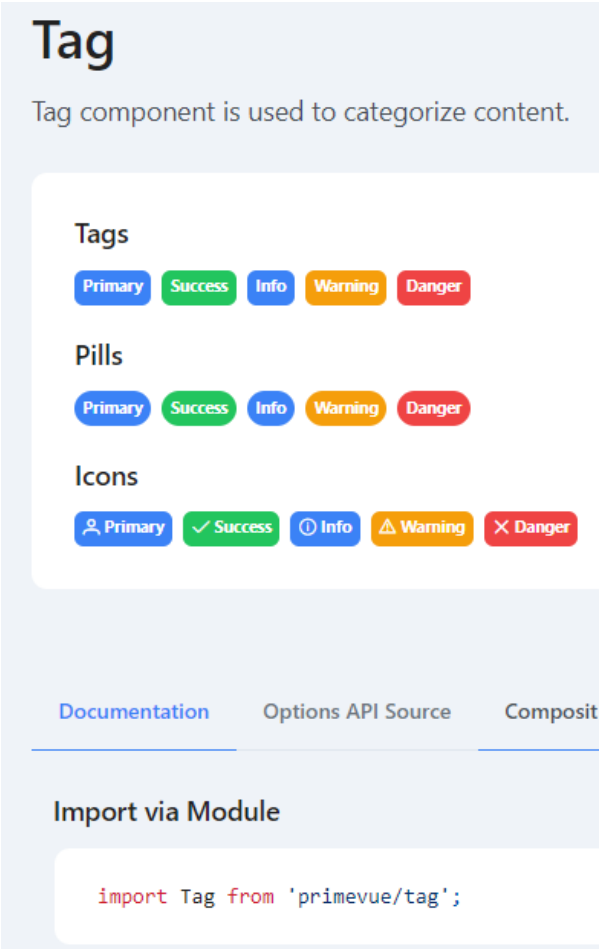
https://primefaces.org/primevue/dropdown



Tag

import Tag from "primevue/tag";

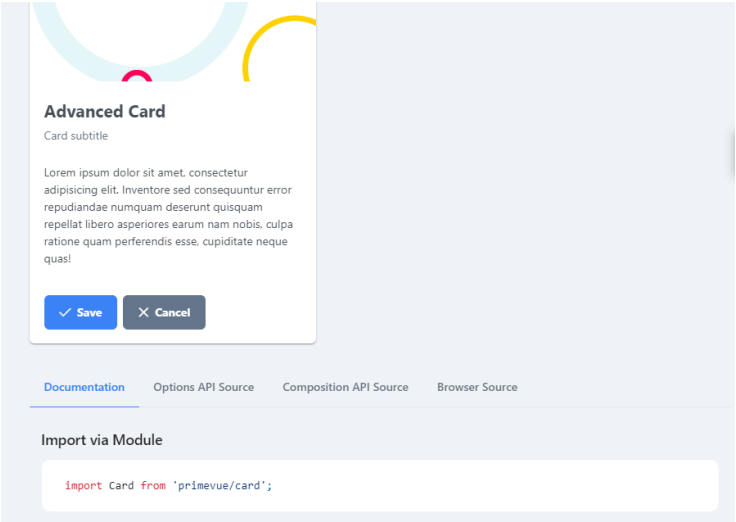
https://primefaces.org/primevue/tag



card

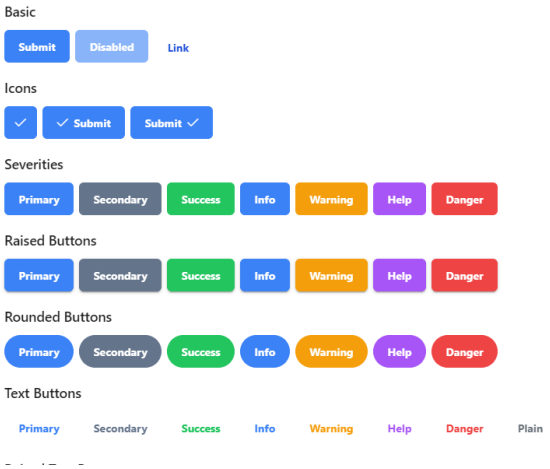
import Card from "primevue/card";

https://primefaces.org/primevue/card



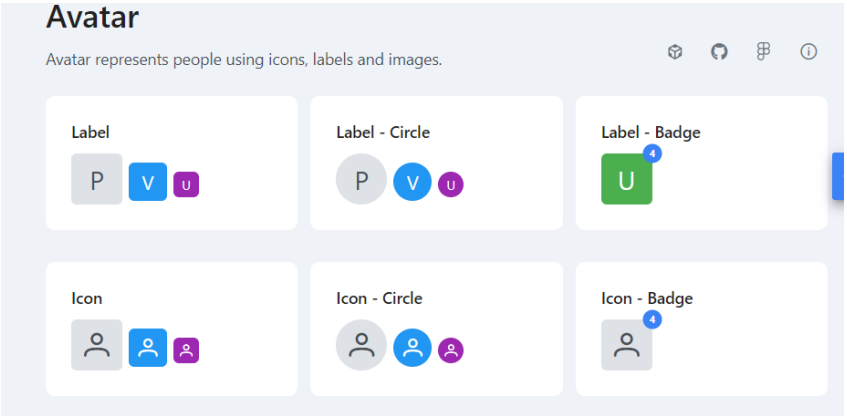
button

https://primefaces.org/primevue/button



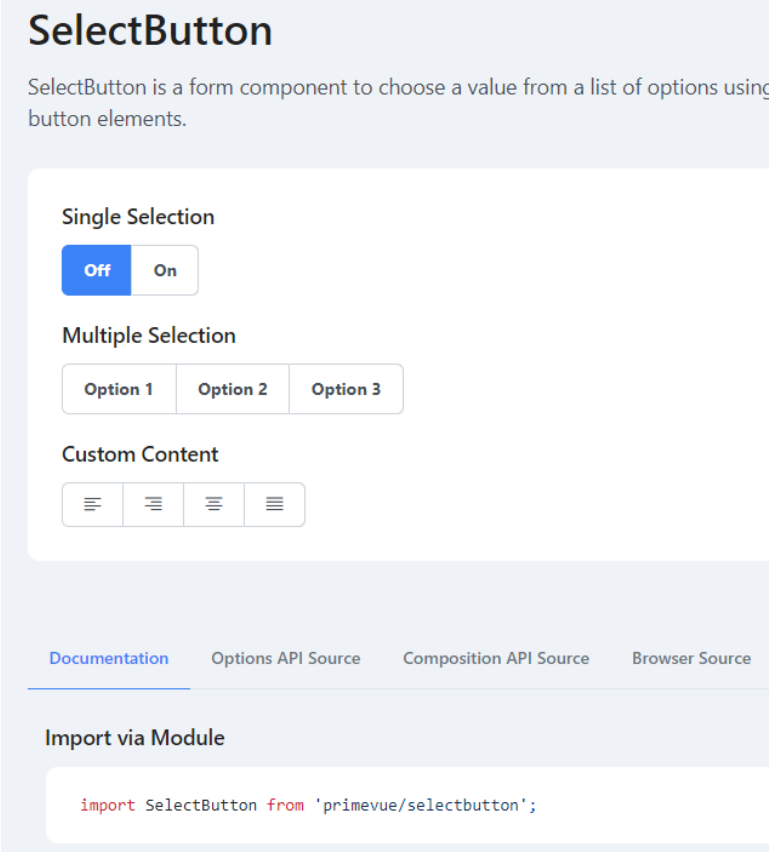
avatar

https://primefaces.org/primevue/avatar



Selectbutton

https://primefaces.org/primevue/selectbutton



Toolbar

https://primefaces.org/primevue/toolbar

Themes Templates Blocks v3.17.0

Toolbar

Toolbar is a grouping component for buttons and other content.

+ New | Upload | Save

Documentation | Options API Source | Composition API Source | Browser Source

Import via Module

```
import Toolbar from 'primevue/toolbar';
```

Import via CDN

```
<script src="https://unpkg.com/primevue@^3/core/core.min.js"></script>
<script src="https://unpkg.com/primevue@^3/toolbar/toolbar.min.js"></script>
```

galleria

https://primefaces.org/primevue/galleria

Getting Started

Galleria requires item template and a value as an array of objects.

```
<Galleria :value="images">
    <template #item="slotProps">
        <img :src="slotProps.item.itemImageSrc" :alt="slotProps.item.alt" />
    </template>
</Galleria>
```

For the rest of the documentation, sample data below would be return from an example service e.g. PhotoService.

```
{
    "data":[
        {
            "itemImageSrc": "demo/images/galleria/galleria1.jpg",
            "thumbnailImageSrc": "demo/images/galleria/galleria1s.jpg",
            "alt": "Description for Image 1",
            "title": "Title 1"
        },
        {
            "itemImageSrc": "demo/images/galleria/galleria2.jpg",
            "thumbnailImageSrc": "demo/images/galleria/galleria2s.jpg",
            "alt": "Description for Image 2",
```

CascadeSelect

https://primefaces.org/primevue/cascadeselect

emptyMessage	string	No available options	Text to be displayed when there are no options available. Defaults to value from PrimeVue locale configuration.
tabindex	number	0	Index of the element in tabbing order.
aria-label	string	null	Defines a string value that labels an interactive element.
aria-labelledby	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" />
```

img

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">
        <img :src="slotProps.item.itemImageSrc" :alt="slotProps.item.alt" style="width: 100%" />
    </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" />
```

Json Server

Instalación

Se instala JSON server

JSON Server está disponible mediante paquete NPM y la instalación puede realizarse utilizando el gestor de paquetes Node.js:

npm install -g json-server

Al añadir la opción -g nos estamos asegurando que el paquete esté instalado globalmente en nuestro sistema

```
:\2022-02\SI730 - web\semana 06\learning-center\server>npm install -g json-server

dded 182 packages, and audited 183 packages in 12s

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

 moderate severity vulnerabilities

o address all issues (including breaking changes), run:
  npm audit fix --force

un `npm audit` for details.
```

Generar servicios

Creamos los dos archives

db.json

Creamos ahora un archivo db.json que contendrá nuestros datos de ejemplo.

Este archivo contiene los datos que deben ser expuestos por la API de REST y para los objetos contenidos en la estructura JSON, los entpoints CRUD se crearán automáticamente. Unos datos de ejemplo podrían ser

```
{
  "tutorials": [
    {
      "id": 1,
      "title": "The Vue Tutorials",
      "description": "The best tips and tutorials for Vue.",
      "published": false
    },
    {
      "id": 2,
      "title": "Amazing Microsoft .NET",
      "description": "Weekly tutorials about .NET and ASP.NET Core.",
      "published": false
    },
    {
      "id": 3,
      "title": "JavaScript for All",
      "description": "Tips an tricks about JavaScript from scratch.",
      "published": false
    },
    {
      "id": 4,
      "title": "Vue Unleashed",
      "description": "Vue at its best.",
      "published": false
    }
  ],
```

```
  "shop": [
    {
      "id": 1,
      "address": "Calle Juan Martín",
      "type": "frutería",
      "nombre": "Lola Castro Frutas",
      "latitude": 37.880273,
      "longitude": -4.792098
    },
    {
      "id": 2,
      "address": "Calle Pepe Cruz",
      "type": "supermercado",
      "nombre": "Ultramarinos Lolo Castro",
      "latitude": 37.862323,
      "longitude": -4.77812
    },
    {
      "id": 3,
      "address": "Avenida de la Cruz",
      "type": "pescadería",
      "nombre": "Pescados El Boquerón",
      "latitude": 37.856273,
      "longitude": -4.776992
    }
  ]
}
```

routes.json

Para modificar y crear rutas personalizadas generamos un nuevo archivo json con la configuración que deseemos donde quedarán definidos los alias de las rutas.

```
{
  "/api/v1/*" : "/$1",
  "/shop/:type": "/shop?type=:type",
  "/shop/address/:address" :"/shop?address_like=:address"
}
```

Arrancando el JSON Server

Iniciemos el servidor JSON ejecutando el siguiente comando:

```
json-server --watch db.json --routes routes.json
```

localhost:3000/shop?type=supe

localhost:3000/shop?type=supermercado


```
[
  {
    "id": 2,
    "address": "Calle Pepe Cruz",
    "type": "supermercado",
    "nombre": "Ultramarinos Lolo Castro",
    "latitude": 37.862323,
    "longitude": -4.77812
  }
]
```


localhost:3000/shop/supermerca

localhost:3000/shop/supermercado


```
[
  {
    "id": 2,
    "address": "Calle Pepe Cruz",
    "type": "supermercado",
    "nombre": "Ultramarinos Lolo Castro",
    "latitude": 37.862323,
    "longitude": -4.77812
  }
]
```


localhost:3000/api/v1/tutorials

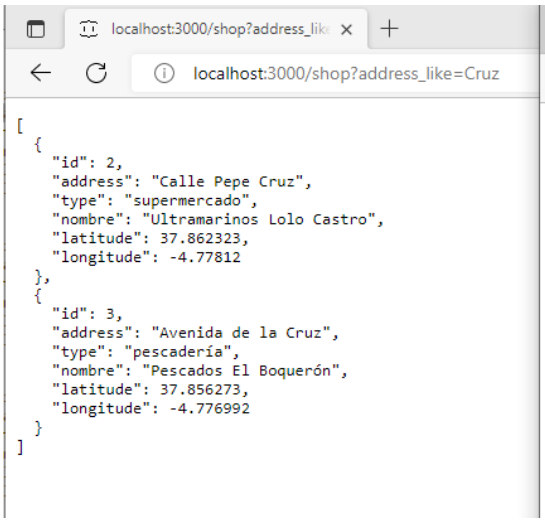
localhost:3000/api/v1/tutorials

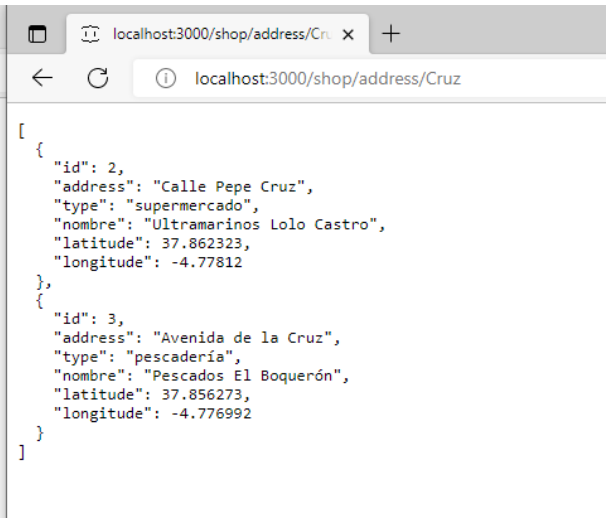

```
[
  {
    "id": 1,
    "title": "The Vue Tutorials",
    "description": "The best tips and tutorials for Vue.",
    "published": false
  },
  {
    "id": 2,
    "title": "Amazing Microsoft .NET",
    "description": "Weekly tutorials about .NET and ASP.NET Core.",
    "published": false
  },
  {
    "id": 3,
    "title": "JavaScript for All",
    "description": "Tips an tricks about JavaScript from scratch.",
    "published": false
  },
  {
    "id": 4,
    "title": "Vue Unleashed",
    "description": "Vue at its best.",
    "published": false
  }
]
```


localhost:3000/tutorials

localhost:3000/tutorials


```
[
  {
    "id": 1,
    "title": "The Vue Tutorials",
    "description": "The best tips and tutorials for Vue.",
    "published": false
  },
  {
    "id": 2,
    "title": "Amazing Microsoft .NET",
    "description": "Weekly tutorials about .NET and ASP.NET Core.",
    "published": false
  },
  {
    "id": 3,
    "title": "JavaScript for All",
    "description": "Tips an tricks about JavaScript from scratch.",
    "published": false
  },
  {
    "id": 4,
    "title": "Vue Unleashed",
    "description": "Vue at its best.",
    "published": false
  }
]
```





- http://localhost:3000/tutorials
- http://localhost:3000/api/v1/tutorials

- http://localhost:3000/shop?type=supermercado
- http://localhost:3000/shop/supermercado

- http://localhost:3000/shop?address_like=Cruz
- http://localhost:3000/shop/address/Cruz

Al usar --watch db.json nos aseguramos de que el servidor se inicie en modo de vigilancia, lo que significa que vigila los cambios de archivos y actualiza la API expuesta en consecuencia.

README.md

```
## Project Setup

```sh
npm install
```

```sh
cd server
json-server --watch db.json --routes routes.json
```

Compile and Hot-Reload for Development
```

otro ejemplo

```
json-server -H 0.0.0.0 -p 3000 --watch db.json --routes routes.json
```

```
GET    /tutorials
GET    /tutorials/{id}
```

```
POST   /tutorials
PUT    /tutorials /{id}
PATCH  /tutorials /{id}
DELETE /tutorials /{id}
```

ejemplo:

ordenar descendente

```
http://localhost:3000/shop?_sort=type&_order=desc
```

buscamos las direcciones que contienen la palabra cruz

```
http://localhost:3000/shop?address_like=Cruz
```

```
http://localhost:3000/shop/?q=lo
```

https://github.com/typicode/json-server

Aquí logramos crear el api fake.