

CONSUMINDO API REST COM VUE.JS: CRUD COM LARAVEL + VUE.JS

1. API Rest

Para iniciar o CRUD com VUE.js e Laravel, é interessante desenvolvermos nossa API rest. Caso queira conferir, pode visualizar os arquivos routes > api.php e app > Http > Controllers > ProdutosController.php

2. Axios Install

Devemos instalar, apontando para a pasta do projeto (via terminal), o Axios. O Axios é a ferramenta que utilizaremos para consumir API rest. Insira o comand:

```
> npm install axios --save
```

3. Services

- * Criamos a pasta 'services' em resources > js;
- * Dentro de services (pasta recém-criada), iniciamos outros 2 arquivos -> config.js e {arquivo-relacionado-a-api}.js;
- * O conteúdo de resources > js > services > config.js deve ser:

```
import axios from 'axios'

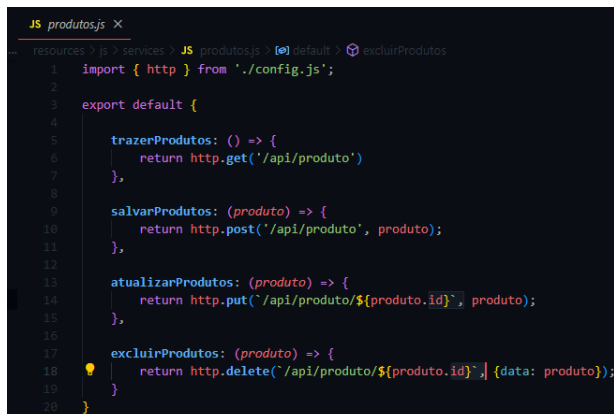
export const http = axios.create({

  baseURL: 'http://127.0.0.1:8000' /* AQUI É COLOCADA A BASE DA ROTA DA API */

})
```

4. resources > js > services > produtos.js

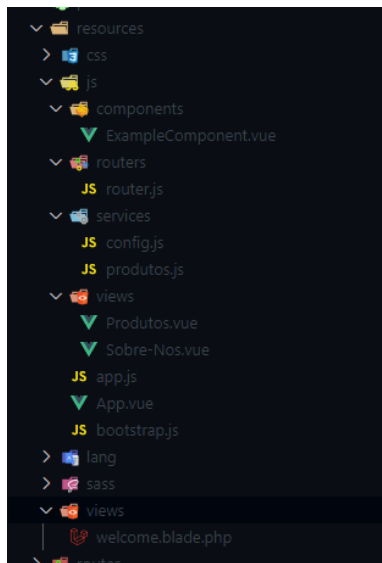
Tenha conhecimento de que o termo “produto(s)” são relacionados ao nosso atual projeto. Em seu projeto particular, deve intitular esses dados e arquivos de acordo com seu objetivo. Veja nosso arquivo com métodos para a requisição, espelhe-se nisso para gerenciar seus verbos http:



```
JS produtos.js X
resources > js > services > JS produtos.js [e] default [x] excluirProdutos
1 import { http } from './config.js';
2
3 export default {
4
5   trazerProdutos: () => {
6     return http.get('/api/produto')
7   },
8
9   salvarProdutos: (produto) => {
10    return http.post('/api/produto', produto);
11  },
12
13  atualizarProdutos: (produto) => {
14    return http.put(`/api/produto/${produto.id}`, produto);
15  },
16
17  excluirProdutos: (produto) => {
18    return http.delete(`/api/produto/${produto.id}`, {data: produto});
19  }
20 }
```

5. Estrutura de Pastas

A esta altura, é válido ressaltar a nossa atual estrutura de pastas. Verifique se a sua está de acordo:



6. Manipulação do Front-End

Observe nossas configurações em *resources > js > views > Produtos.vue*:

```
<script>
<div class="formulario">
  <form @submit.prevent="salvar" method="post" id="form">

    <div class="input-group mb-3">
      <span class="input-group-text" id="basic-addon1"><i class="bi bi-gear"></i></span>
      <input type="text" maxlength="80" v-model="produto.produto" class="form-control"
        placeholder="Produto" aria-label="Username" aria-describedby="basic-addon1">
    </div>

    <div class="input-group mb-3">
      <span class="input-group-text" id="basic-addon1"><i class="bi bi-cash-coin"></i></span>
      <input type="number" step="0.01" v-model="produto.preco"
        class="form-control" placeholder="Preço" aria-label="Username" aria-describedby="basic-addon1"
        id="preco">
    </div>

    <div class="input-group mb-3">
      <span class="input-group-text" id="basic-addon1"><i class="bi bi-receipt"></i></span>
      <input type="number" v-model="produto.cod_de_barras" class="form-control" placeholder="Código de barras"
        aria-label="Username" aria-describedby="basic-addon1">
    </div>

    <button class="btn btn-primary" type="button" id="btn-produto">SALVAR PRODUTO <i class="bi bi-id-card"></i></button>
  </div>
</script>
```

Neste exemplo de formulário, podemos notar alguns aspectos, vamos lista-los:

- @submit.prevent="salvar"
- v-model="produto.produto"

Este @submit no formulário servirá para executar a action, neste caso, denominada de salvar. O v-model atribui um valor para o input, indicando por qual dado ele será responsável (neste caso, o nome do produto).

Agora vejamos um exemplo de uma tabela:

```
<table class="table table-bordered border-dark tb-style">
  <thead>
    <tr>
      <th scope="col">ID</th>
      <th scope="col">PRODUTO</th>
      <th scope="col">PREÇO</th>
      <th scope="col">COD. BARRAS</th>
      <th scope="col">AÇÕES</th>
    </tr>
  </thead>
  <tbody>
    <tr v-for="produto of produtos" :key="produto.id">
      <th scope="row">{{ produto.id }}</th>
      <td>{{ produto.produto }}</td>
      <td>{{ produto.preco }}</td>
      <td>{{ produto.cod_de_barras }}</td>
      <td>
        <i class="bi bi-pencil color-blue" @click="editar(produto)"></i>
        <i class="bi bi-trash color-red" @click="excluir(produto)"></i>
      </td>
    </tr>
  </tbody>
</table>
```

Nota que o `<tr></tr>` da nossa tabela, encontra-se o `v-for="produto of produtos"` `:key="produto.id"`, elemento responsável por inserir os dados. Novamente, substitua o termo "produto" pelo seu termo desejado. Nos `<td></td>` inserimos dados como `{{ produto.id }}`. Também é válido ressaltar que usamos o `@click="editar(produto)"`, para editar (trocamos o método especificado para executar outras funções).

Agora, vejamos a tag `<script></script>` os métodos para manipular as informações:

```
/* importando o nosso arquivo com as requisições */
import Produtos from '../services/produtos.js';

/* Criando exportações, sendo elas as seguintes:
Dados do produto;
Mounted() -> função realizada ao iniciar o arquivo;
Methods: Métodos que usamos no formulário e na tabela > Estes fazem uso
dos métodos importados pelo componente "Produtos"

*/
export default {
  data() {
    return {
      produto: {
        id: '',
        produto: '',
        preco: '',
        cod_de_barras: ''
      },
      produtos: []
    },
  },

  mounted() {
    this.trazer();
  },

  methods: {

    trazer() {
      Produtos.trazerProdutos().then(resposta => {
        console.log(resposta.data.produtos);
        this.produtos = resposta.data.produtos;
      })
    },

    salvar() {
      if (!this.produto.id) {
        Produtos.salvarProdutos(this.produto).then(resposta => {
          Swal.fire({
            position: 'top-end',
```

```

        icon: resposta.data.icon,
        title: resposta.data.mensagem,
        showConfirmButton: false,
        timer: 2000
    });
    this.trazer();
    })
} else {
    Produtos.atualizarProdutos(this.produto).then(resposta =>
{
    Swal.fire({
        position: 'top-end',
        icon: resposta.data.icon,
        title: resposta.data.mensagem,
        showConfirmButton: false,
        timer: 2000
    });
    this.trazer();
    })
}
},

editar(produto) {
    this.produto = produto
},

excluir(produto) {
    Produtos.excluirProdutos(produto).then(resposta => {
        Swal.fire({
            position: 'top-end',
            icon: resposta.data.icon,
            title: resposta.data.mensagem,
            showConfirmButton: false,
            timer: 2000
        });
        this.trazer();
    })
}

}
}

```