

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

Unidade Curricular: Desenvolvimento Web – Front-End

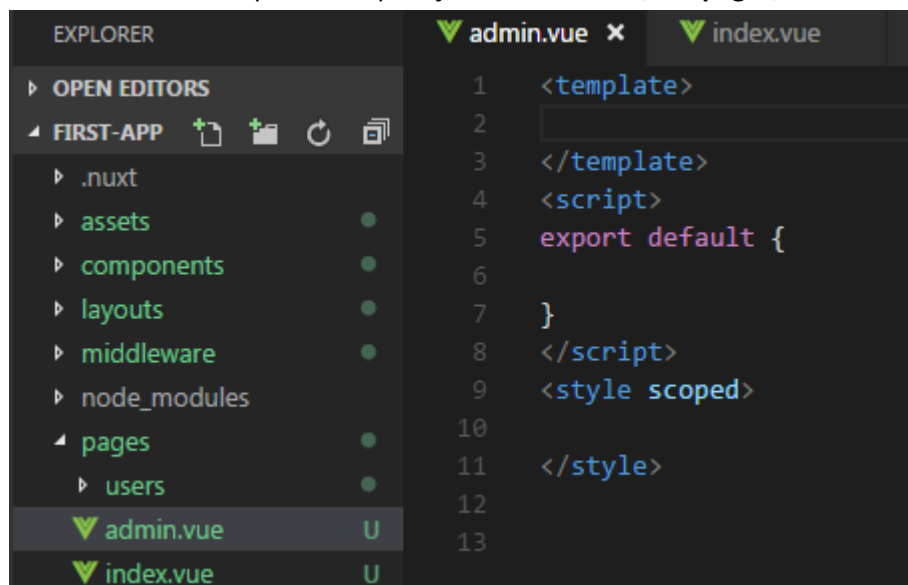
1º Ano/2º Semestre

Docente: Marco Miguel Olival Olim

Data 06/04/2018

ESTE EXERCÍCIO ABORDA FORMULÁRIOS E GRAVAÇÃO DA SUA INFORMAÇÃO EM BASES DE DADOS

Na sequência do exercício anterior, vamos acrescentar um formulário para introduzir a informação na Base de Dados de Firebase a partir da aplicação web. Criamos, em **pages**, um novo ficheiro **admin.vue**



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure with folders like .nuxt, assets, components, layouts, middleware, node_modules, and pages. The pages folder is expanded, showing subfolders users and admin.vue. The admin.vue file is selected and its content is displayed in the editor. The code in admin.vue is as follows:

```
1 <template>
2
3 </template>
4 <script>
5   export default {
6
7   }
8 </script>
9 <style scoped>
10
11 </style>
12
```

Um formulário é delimitado pela tag **<form>**. Tipicamente tem associado um atributo **action** que define a ação a realizar quando o formulário é submetido. Esta ação é desencadeada por um botão nesse formulário do tipo **submit**. A informação, propriamente dita, é submetida por tags como o **input**, as quais geralmente têm associada um **label** com o descritivo



The screenshot shows the VS Code editor with the admin.vue file open. The code in the editor is as follows:

```
1 <template>
2   <form action="">
3
4     <div class="input-control">
5       <label for="">Produto</label>
6       <input type="text">
7     </div>
8
9     <button type="submit">Enviar</button>
10
11   </form>
12 </template>
```

Cofinanciado por:

No caso do VueJS, a ação para submeter o formulário tem a designação de **@submit**. Acrescentamos a propriedade **prevent** para evitar que ao submeter o formulário seja efetuado um page reload, que é o default desta operação. O método associado ao submit é o **submeterFormulario**. Note que depois definir aqui a ação já não é necessário associar @click ao botão com o tipo **submit**.

```
1  <template>
2    <form @submit.prevent="submeterFormulario" class="form">
3
4      <div class="input-control">
5        <label for="">Produto</label>
6        <input type="text">
7      </div>
8
9      <button type="submit" class="button" >Enviar</button>
10
11    </form>
12  </template>
13
14  <script>
15    export default {
16
17    }
18  </script>
19  <style scoped>
20    .form {
21      max-width: 550px;
22      box-sizing: border-box;
23      margin: 30px;
24      margin-top: 60px;
25      padding: 30px;
26      text-align: justify;
27      box-shadow: 0 0 24px 0 rgba(0, 0, 0, 0.3);
28      width: 100%;
29    }
30    .button {
31      font: inherit;
32      cursor: pointer;
33      border-radius: 4px;
34      border: 1px solid #06c4d1;
35      background-color: white;
36      color: #06c4d1;
37      text-decoration: none;
38      padding: 10px 30px;
39    }
40    .button:hover,
41    .button:active {
42      color: #fff;
43      background-color: #06c4d1;
44    }
```

Cofinanciado por:



A informação introduzida é veiculada pelo atributo **v-model**, que neste caso irá preencher o objeto **novoProduto** definido em **data()**

```
1  <template>
2    <form @submit.prevent="submeterFormulario" class="form">
3
4      <div class="input-control">
5        <label>Produto</label>
6        <input type="text" v-model="novoProduto.produto">
7      </div>
8      <div class="input-control">
9        <label>Preço</label>
10       <input type="text" v-model="novoProduto.valor">
11     </div>
12
13     <br>
14     <button type="submit" class="button" >Enviar</button>
15
16   </form>
17 </template>
18
19 <script>
20 export default {
21   data () {
22     return {
23       novoProduto:{
24         produto:'',
25         valor:''
26       }
27     }
28   },
29   methods:{
30     submeterFormulario() {
31       //console.log(this.novoProduto)
32     }
33   }
34 }
35 </script>
```

O método **submeterFormulario** envia então um POST com o objeto **novoProduto** para o Firebase (atenção: não esquecer de importar o **axios**)

```
19 <script>
20 import axios from 'axios';
21 export default {
22   data () {
23     return {
24       novoProduto:{
25         produto:'',
26         valor:''
27       }
28     }
29   },
30   methods:{
31     submeterFormulario() {
32       return axios.post('https://umartigos.firebaseio.com/.json', this.novoProduto);
33     }
34   }
35 }
36 </script>
```

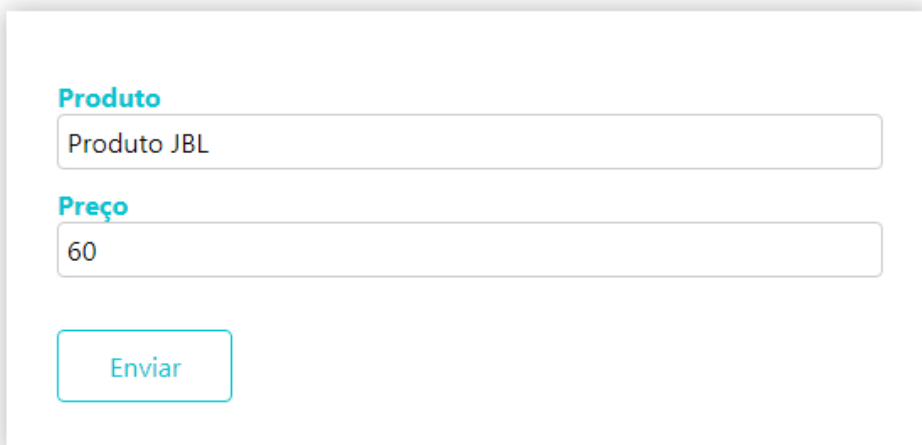
Cofinanciado por:



Antes de testar, vamos acabar de formatar o formulário, aplicando estilos à tag **input**

```
38 .input-control {
39   margin: 10px 0;
40 }
41 .input-control label {
42   display: block;
43   font-weight: bold;
44   color: #06c4d1;
45 }
46 .input-control input {
47   display: block;
48   width: 100%;
49   box-sizing: border-box;
50   font: inherit;
51   border: 1px solid #ccc;
52   border-radius: 4px;
53   padding: 5px;
54 }
55 .input-control input:focus {
56   background-color: #eee;
57   outline: none;
58 }
```

Introduzimos então um novo produto no formulário:



O formulário contém dois campos de entrada e um botão. O primeiro campo, rotulado 'Produto', contém o texto 'Produto JBL'. O segundo campo, rotulado 'Preço', contém o valor '60'. Abaixo dos campos, há um botão com o texto 'Enviar'.

No Firebase verifica-se que fica registado o novo produto. Refira-se que o Id do produto é gerado automaticamente com base num timestamp que o torna único.



Cofinanciado por:

