

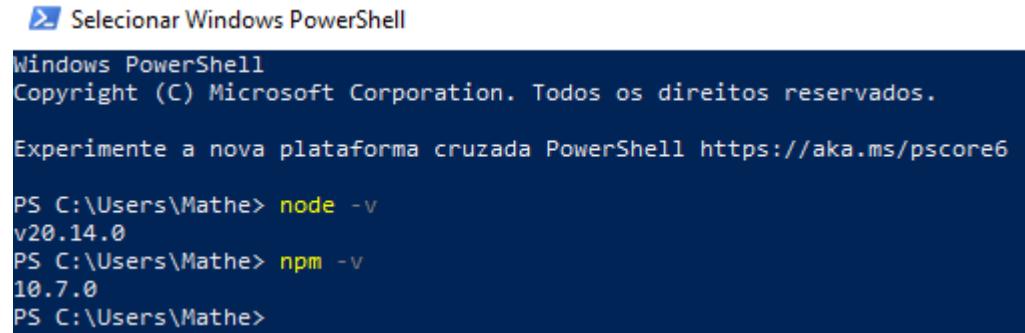
Alunos: Matheus Bueno, Vitor Mayorca Camargo

Repositório git: <https://codeberg.org/radajaaj/AP05-VueJs>

Vídeo de demonstração:

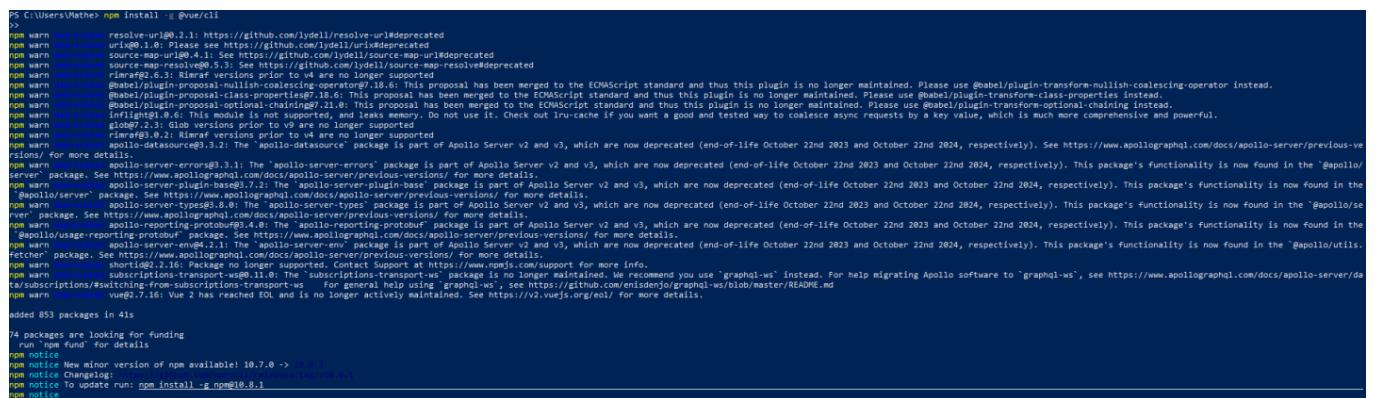
<https://codeberg.org/radajaaj/AP05-VueJs/src/branch/main/apresenta%C3%A7%C3%A3o-trab%205-%20TDS.mp4>

Inicialmente, instalamos o Node.js do site oficial (<https://nodejs.org/en/>):



```
PS C:\Users\Mathe> node -v
v20.14.0
PS C:\Users\Mathe> npm -v
10.7.0
PS C:\Users\Mathe>
```

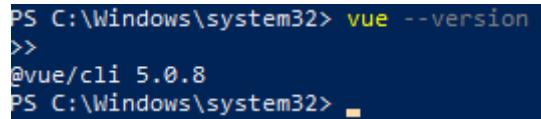
E o Vue CLI foi instalado por meio do comando “npm install -g @vue/cli”:



```
PS C:\Users\Mathe> npm install -g @vue/cli
...
npm warn deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm warn deprecated url@0.1.0: Please see https://github.com/lydell/url#deprecated
npm warn deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm warn deprecated source-map@0.5.0: https://github.com/lydell/source-map#deprecated
npm warn deprecated rimraf@2.6.3: Rimraf version prior to v4 is no longer supported
@babel/plugin-proposal-nullish-coalescing-operator@7.18.6: This proposal has been moved to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-nullish-coalescing-operator instead.
@babel/plugin-proposal-class-properties@7.18.6: This proposal has been moved to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-class-properties instead.
@babel/plugin-proposal-optional-chaining@7.11.0: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-optional-chaining instead.
@babel/plugin-proposal-inflight@0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated glob@5.0.1: Glob versions prior to v9 are no longer supported
npm warn deprecated minimatch@3.0.4: Minimatch 3.0.4 and later are no longer supported
npm warn deprecated apollo-datasource@0.3.2: The 'apollo-datasource' package is part of Apollo Server v2 and v3, which are now deprecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
...
@apollo/server-error@0.3.1: The 'apollo-server-error' package is part of Apollo Server v2 and v3, which are now deprecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the '@apollo/server' package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
@apollo/server-plugin-base@0.7.2: The 'apollo-server-plugin-base' package is part of Apollo Server v2 and v3, which are now deprecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the '@apollo/server' package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
@apollo/server-reporter@0.8.4: The 'apollo-server-reporter' package is part of Apollo Server v2 and v3, which are now deprecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the '@apollo/reporter' package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
@apollo/reporting-protobuf@0.4.0: The 'apollo-reporting-protobuf' package is part of Apollo Server v2 and v3, which are now deprecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the '@apollo/reporting' package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
...
@apollo/usage-reporter@0.1.0: The 'apollo-usage-reporter' package is part of Apollo Server v2 and v3, which are now deprecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the '@apollo/utils-fetcher' package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
...
@apollo/subscription-transport-ws@1.0.1: Subscription transport ws package is no longer supported. Contact Support: https://apollographql.com/contact-support/ for more info.
...
@apollo/subscriptions#switching-from-subscriptions-transport-ws: For general help using 'graphql-ws', see https://github.com/apollographql/graphql-ws/blob/master/README.md
...
npm warn deprecated vue@2.7.16: Vue 2 has reached EOL and is no longer actively maintained. See https://v2.vuejs.org/eol/ for more details.

added 853 packages in 41s
74 packages are looking for funding
  run `npm fund` for details
npm notice New minor version of npm available! 10.7.0 → 10.8.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.1
npm notice To update run: npm install -g npm@10.8.1
npm notice
```

Podemos ver que a instalação teve sucesso com o uso do comando vue –version:



```
PS C:\Windows\system32> vue --version
>>
@vue/cli 5.0.8
PS C:\Windows\system32>
```

Agora, dentro do VSCode, abrimos uma pasta qualquer e, dentro dela, usamos o Vue CLI para criar um novo projeto com base na template webpack:

```
PS C:\Users\Mathe\OneDrive\Desktop\site> vue init webpack site

? Project name site
? Project description trabalho tds
? Author Matheus and Victor
? Vue build standalone
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Set up unit tests No
? Setup e2e tests with Nightwatch? Yes
? Should we run `npm install` for you after the project has been created? (recommended) npm
```

Por fim, instalamos o bootstrap no projeto com o seguinte comando:

```
npm install bootstrap bootstrap-vue
```

E no arquivo ./src/main.js adicionamos as seguintes linhas:

```
// Importamos o bootstrap abaixo
import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'

import { BootstrapVue, IconsPlugin } from 'bootstrap-vue'

Vue.use(BootstrapVue)
Vue.use(IconsPlugin)
```

Agora todas as ferramentas necessárias para a execução e compilação do projeto estão instaladas e prontas para uso.

Decidimos fazer um site simples, que é a página de uma lanchonete chamada “AvaLanches”, aonde o usuário vai escolher seu pedido e obter o preço da compra.

Todas as alterações no template foram feitas dentro da pasta ./src, apenas com o uso de arquivos .vue. Inicialmente, no arquivo componente principal (./src/App.vue), inserimos um banner para a nossa loja on-line. Tanto o banner quanto a logo da lanchonete foram gerados pelo criador de imagens do Microsoft Designer (<https://www.bing.com/images/create?FORM=GENILP>).

Começamos definindo as tags template, script e style. Na aba script, importamos todas as imagens e componentes que são utilizados no site. Mais à frente veremos cada componente mais a fundo:

```
import hello from './components/HelloWorld.vue' // Importamos
import contador from './components/contador'
import compra from './components/compra'
import hamburguerCard from './assets/hamburguer_card.png'
import cocaColaCard from './assets/coca_cola_card.png'
import batataFritaCard from './assets/batata_frita_card.png'
import bannerImage from './assets/avalanches.png'
```

Dentro de style, definimos a div “app”, que conterá todos os elementos da nossa view (tela/página). Nela também inserimos o nosso banner, com a classe img-fluid do bootstrap, que permite responsividade:

```
<template>
  <div id="app">
    
  </div>
</template>
```

Em script também exportamos os componentes importados anteriormente:

```
export default {
  name: 'App',
  components: {
    hello, // Registrado no script
    contador,
    compra
  },
}
```

O primeiro componente utilizado foi o “HelloWorld.vue”. Esse componente simplesmente mostra uma mensagem na tela, com a fonte padrão do Vue.js. Ele foi criado para testar as funcionalidades básicas do Vue.js, e funciona com o uso de variáveis definidas na seção script:

```
<template>
  <div class="hello">
    <h1 :title="msg">{{ msg }}</h1>
  </div>
</template>
```

```
<script>
export default{
  name: 'hello',
  data () {
    return {
      msg: 'Boas-vindas à nossa loja! Escolha o seu pedido:'
    }
  }
</script>
```



Boas-vindas à nossa loja! Escolha o seu pedido:

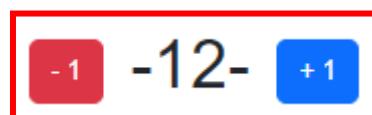
O segundo componente feito foi o contador.vue. Ele é simples: Dois botões (mais e menos) que servem para o usuário definir quanto de cada produto ele vai querer. Essa quantidade é armazenada na variável quantidade (que nunca será menor que 0). Os botões são gerados com as variantes danger e primary do bootstrap. A classe my-4 ajuda a garantir a responsividade dos componentes:

```
<template>
  <div class="d-flex justify-content-center align-items-center my-4">
    <b-button variant="danger" @click="menos">- 1</b-button>
    <h1 class="mx-3"> -{{ quantidade }}- </h1>
    <b-button variant="primary" @click="mais">+ 1</b-button>
  </div>
</template>
```

```

<script>
export default {
  props: {
    value: Number // Passa o valor de quantidade para as outras funções
  },
  data () {
    return {
      quantidade: this.value
    }
  },
  methods: {
    mais () {
      this.quantidade += 1
      this.$emit('input', this.quantidade)
    },
    menos () {
      if (this.quantidade > 0) {
        this.quantidade -= 1
        this.$emit('input', this.quantidade)
      }
    }
  }
}
</script>

```



O valor escolhido é retornado para o componente App.vue por meio do prop “value”. Dentro de App.vue, o retorno é recebido pelo comando v-model, dentro da tag do componente contador:

```
<contador v-model="item.quantity"></contador>
```

Como podemos ver, o valor é retornado numa variável de nome genérico “item.quantity”. Isso acontece pois os contadores estarão dentro do cartão de cada produto vendido, e esses cartões são gerados dinamicamente:

```

<div class="d-flex justify-content-center">
  <div class="container">
    <div class="row">
      <div class="col-12 col-md-4 mb-4" v-for="item in items" :key="item.id">
        <div class="card">
          <h1 class="card-title">{{ item.title }}</h1>
          
          <div class="card-body">
            <p class="card-text">{{ item.description }}</p>
          </div>
          <h3 class="card-title">Quantidade:</h3>
          <contador v-model="item.quantity"></contador>
        </div>
      </div>
    </div>
  </div>
</div>

```

Dentro da seção data () da tag <script> há um vetor de classes chamado items. Ele vai armazenar os dados de cada produto à venda. Se quisermos adicionar mais produtos, basta adicionar mais classes ao vetor, e preencher as variáveis:

```

items: [
  { id: 1, title: 'Hambúrguer', image: hamburguerCard, alt: 'Hamburguer saboroso', description: 'A autêntica experiência de um Hambúrguer Rústico Artesanal. Feito com um blend especial de carnes nobres, e grelhado à perfeição', quantity: 0 },
  { id: 2, title: 'Coca-Cola', image: cocaColaCard, alt: 'Coquinha gelada', description: 'Uma bebida clássica e irresistível, perfeita para acompanhar qualquer refeição. Doce, efervescente e refrescante.', quantity: 0 },
  { id: 3, title: 'Batata Frita', image: batataFritaCard, alt: 'Batata frita rústica', description: 'Cortadas em gomos generosos, temperadas com ervas finas e fritas até ficarem douradas e crocantes. Acompanha perfeitamente qualquer refeição.', quantity: 0 }
]

```

E para cada item do vetor item, será computado automaticamente o tanto de cada produto que o usuário escolheu no contador, com base na ID do item:

```

computed: {
  hamburguerQuantidade () {
    return this.items.find(item => item.id === 1).quantity
  },
  batataQuantidade () {
    return this.items.find(item => item.id === 3).quantity
  },
  cocaQuantidade () {
    return this.items.find(item => item.id === 2).quantity
  }
},

```

No site fica assim:

Hambúrguer



A autêntica experiência de um Hambúrguer Rústico Artesanal. Feito com um blend especial de carnes nobres, e grelhado à perfeição

Quantidade:

-1 -12- +1

Coca-Cola



Uma bebida clássica e irresistível, perfeita para acompanhar qualquer refeição. Doce, efervescente e refrescante.

Quantidade:

-1 -1- +1

Batata Frita



Cortadas em gomos generosos, temperadas com ervas finas e fritas até ficarem douradas e crocantes. Acompanha perfeitamente qualquer refeição.

Quantidade:

-1 -1- +1

Por fim, esses valores são inseridos no componente compra da seguinte forma:

```
<compra
    :hamburguerQuantidade="hamburguerQuantidade"
    :cocaQuantidade="cocaQuantidade"
    :batataQuantidade="batataQuantidade"
    @resetQuantities="resetQuantities"
></compra>
```

O componente compra recebe esses valores, multiplica pelo preço dos itens, e mostra o total da compra numa sidebar que fica fixa do lado direito da tela:

```
<template>
<div :class="cardClass">
  <div class="card-body">
    <h5 class="card-title">TOTAL DA COMPRA</h5>
    <hr>
    <p class="card-text">Hamburguer: {{ hamburguerQuantidade }} x R$ 19,99 = {{ hamburguerTotal.toFixed(2) }}</p>
    <p class="card-text">Batata frita: {{ batataQuantidade }} x R$ 9,99 = {{ batataTotal.toFixed(2) }}</p>
    <p class="card-text">Coca-Cola: {{ cocaQuantidade }} x R$ 5,99 = {{ cocaTotal.toFixed(2) }}</p>
    <hr>
    <p class="card-text">TOTAL: R$ {{ total.toFixed(2) }}</p>
    <hr>
    <a href="#" class="btn btn-primary" @click="notaFiscal">Finalizar compra!</a>
  </div>
</div>
</template>
```

The screenshot shows a mobile application interface for a burger restaurant. At the top, there are images of burgers and the restaurant's logo. Below that, a welcome message reads "Boas-vindas à nossa loja! Escolha o seu pedido:". The main content area contains three cards, each representing a product:

- Hambúrguer**: An image of a large, multi-layered burger. Description: "A autêntica experiência de um Hambúrguer Rústico Artesanal. Feito com um blend especial de carnes nobres, e grelhado à perfeição". Quantity selector: "-1" (red), "12-", "+1" (blue). Buttons: "-1" (red), "-12-", "+1" (blue).
- Coca-Cola**: An image of a bottle and cans of Coca-Cola. Description: "Uma bebida clássica e irresistível, perfeita para acompanhar qualquer refeição. Doce, efervescente e refrescante.". Quantity selector: "-1" (red), "-1-", "+1" (blue). Buttons: "-1" (red), "-1-", "+1" (blue).
- Batata Frita**: An image of a serving of fries. Description: "Cortadas em gomos generosos, temperadas com ervas finas e fritas até ficarem douradas e crocantes. Acompanha perfeitamente qualquer refeição.". Quantity selector: "-1" (red), "-1-", "+1" (blue). Buttons: "-1" (red), "-1-", "+1" (blue).

To the right, there is a sidebar titled "TOTAL DA COMPRA" with the following calculations:

- Hambúrguer: $12 \times R\$ 19,99 = 239,88$
- Batata frita: $1 \times R\$ 9,99 = 9,99$
- Coca-Cola: $1 \times R\$ 5,99 = 5,99$

The total is listed as "TOTAL: R\\$ 255,86". At the bottom right of the sidebar is a blue button labeled "Finalizar compra!".

Os valores são recebidos por meio de props, e são calculados da seguinte maneira:

```
cocaTotal () {
    return this.cocaQuantidade * 5.99
},
total () {
    return this.hamburguerTotal + this.batataTotal + this.cocaTotal
}
```

Porém, há um problema com esse cartão que fica grudado na sidebar: em resoluções muito pequenas, a sidebar acaba ocupando espaço demais na tela:

Hambúrguer



A autêntica experiência de um Hambúrguer Rústico Artesanal. Feito com um blend especial de carnes nobres, e

Coca-Cola



Uma bebida clássica e irresistível, perfeita para acompanhar qualquer refeição. Doce, efervescente e

Batata Frita

TOTAL DA COMPRA

Hamburguer: 0 x R\$ 19,99 = 0.00

Batata frita: 0 x R\$ 9,99 = 0.00

Coca-Cola: 0 x R\$ 5,99 = 0.00

TOTAL: R\$ 0.00

Finalizar compra!

Para resolver isso, fizemos com que a classe dessa div fosse dinâmica: se a resolução for grande, ela é um cartão fixo na tela; se a resolução for pequena, ela vira uma flexbox normal. O nome da função é cardClass:

```
<div :class="cardClass">
```

Que retorna o tipo da classe da div de acordo com o tamanho da tela:

```
cardClass () {
  return this.isSmallScreen ? 'card' : 'card fixed-card'
},
```

Se a tela for pequena, como a de um celular, a tabela de preços ficará abaixo dos produtos:



Após o término da implementação do website, ele foi inserido num repositório git na plataforma CodeBerg, com o uso dos comandos “git clone <https://codeberg.org/radajaaj/AP05-VueJs.git>”, “git add.” e “git push origin main”. Dentro do repositório também está uma cópia deste pdf, e um vídeo .mp4 demonstrando o website em funcionamento.

