Gaming shop project documentation

Summary:

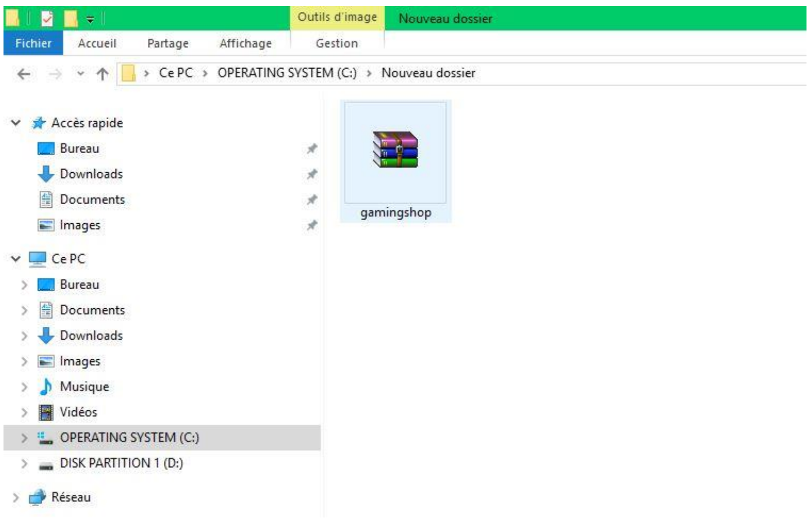
1-Introduction:

This project is front-end part of an e-commerce app, entirely build with technologies including Html, Css and JavaScript. This App is powered by Vue.js Framework.

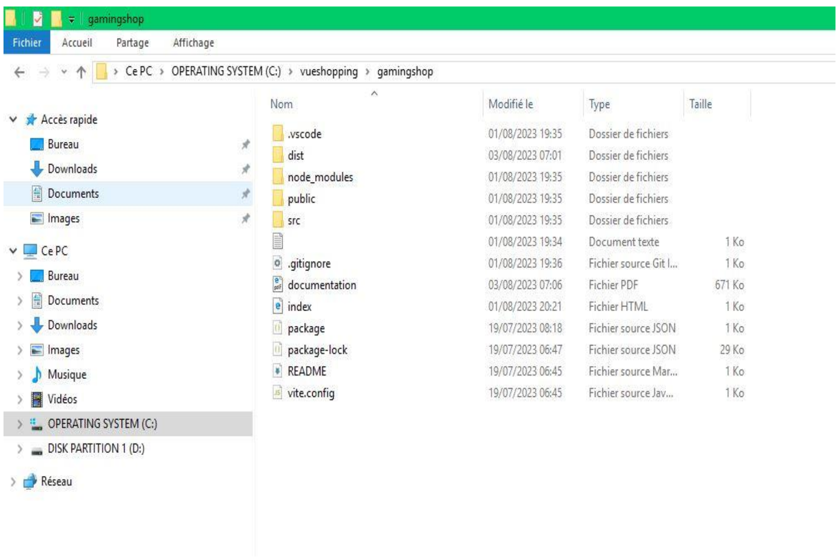
All the images contained in this project are provided by unsplash.com with free license. These are used for product image examples all over this project.

We will see all the components detail and functionalities bellow.

2-Direction for use:



To use this aplication, first, extract the archive named gamingshop.



Nom	Modifié le	Type	Taille
.vscode	01/08/2023 19:35	Dossier de fichiers	
dist	03/08/2023 07:01	Dossier de fichiers	
node_modules	01/08/2023 19:35	Dossier de fichiers	
public	01/08/2023 19:35	Dossier de fichiers	
src	01/08/2023 19:35	Dossier de fichiers	
	01/08/2023 19:34	Document texte	1 Ko
.gitignore	01/08/2023 19:36	Fichier source Git I...	1 Ko
documentation	03/08/2023 07:06	Fichier PDF	671 Ko
index	01/08/2023 20:21	Fichier HTML	1 Ko
package	19/07/2023 08:18	Fichier source JSON	1 Ko
package-lock	19/07/2023 06:47	Fichier source JSON	29 Ko
README	19/07/2023 06:45	Fichier source Mar...	1 Ko
vite.config	19/07/2023 06:45	Fichier source Jav...	1 Ko

After that, these are the contents of the archive:

.vscode

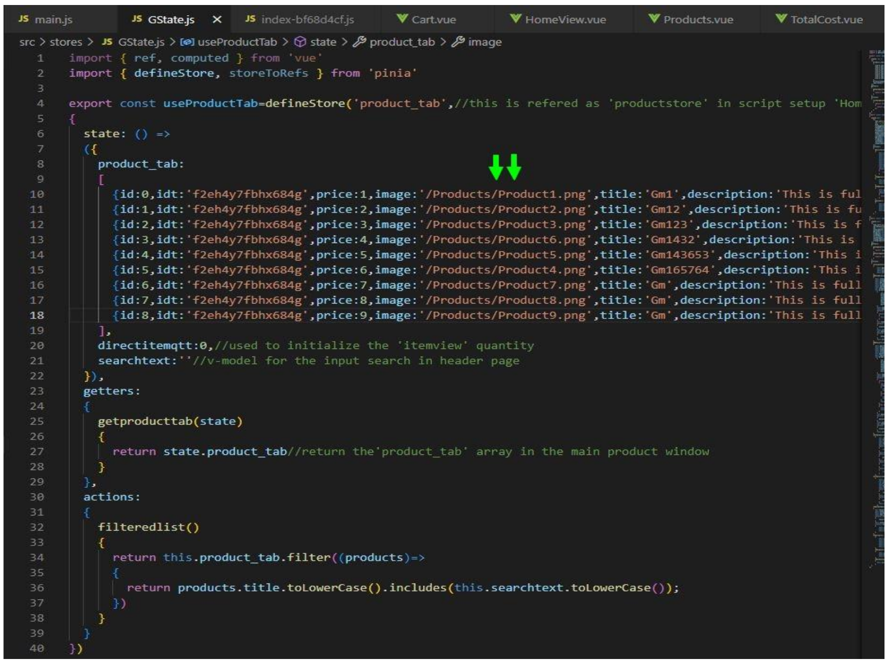
dist: the minified file for production.

node_modules: all the files used for development

public: contain files like icons, product images, ...

src: directory that contains assets, components, router, stores, views directory, App.vue file and the main.js file

Before the user want to make any change in this project, make sure to have Node.js installed, Vue/Cli, and any code editor.



```
1   import { ref, computed } from 'vue'
2   import { defineStore, storeToRefs } from 'pinia'
3
4   export const useProductTab=defineStore('product_tab',//this is refered as 'productstore' in script setup 'Hom
5   {
6     state: () =>
7     ({
8       product_tab:
9       [
10        {id:0,idt:'f2eh4y7fbhx684g',price:1,image:'/Products/Product1.png',title:'Gm1',description:'This is ful
11        {id:1,idt:'f2eh4y7fbhx684g',price:2,image:'/Products/Product2.png',title:'Gm12',description:'This is fu
12        {id:2,idt:'f2eh4y7fbhx684g',price:3,image:'/Products/Product3.png',title:'Gm123',description:'This is f
13        {id:3,idt:'f2eh4y7fbhx684g',price:4,image:'/Products/Product6.png',title:'Gm1432',description:'This is
14        {id:4,idt:'f2eh4y7fbhx684g',price:5,image:'/Products/Product5.png',title:'Gm143653',description:'This i
15        {id:5,idt:'f2eh4y7fbhx684g',price:6,image:'/Products/Product4.png',title:'Gm165764',description:'This i
16        {id:6,idt:'f2eh4y7fbhx684g',price:7,image:'/Products/Product7.png',title:'Gm',description:'This is full
17        {id:7,idt:'f2eh4y7fbhx684g',price:8,image:'/Products/Product8.png',title:'Gm',description:'This is full
18        {id:8,idt:'f2eh4y7fbhx684g',price:9,image:'/Products/Product9.png',title:'Gm',description:'This is full
19      ],
20      directitemqtt:0,//used to initialize the 'itemview' quantity
21      searchtext:''//v-model for the input search in header page
22    }),
23    getters:
24    {
25      getproducttab(state)
26      {
27        return state.product_tab//return the'product_tab' array in the main product window
28      }
29    },
30    actions:
31    {
32      filteredlist()
33      {
34        return this.product_tab.filter((products)=>
35        {
36          return products.title.toLowerCase().includes(this.searchtext.toLowerCase());
37        })
38      }
39    }
40  })
```

The user can modify all the content of this "product_tab" array in the store.

It is better that we can fetch all data via an API. This application can do that because Axios is already installed with all the necessary dependencies.

As we can see, at the line 6 to 18, this is how to declare a state. In this case we use an array of object and the "Products.vue" in the "HomeView" component loop over this array to fetch all data.

Thereafter, all keys and value of the array are passed via props to be used by another component that we will see later.

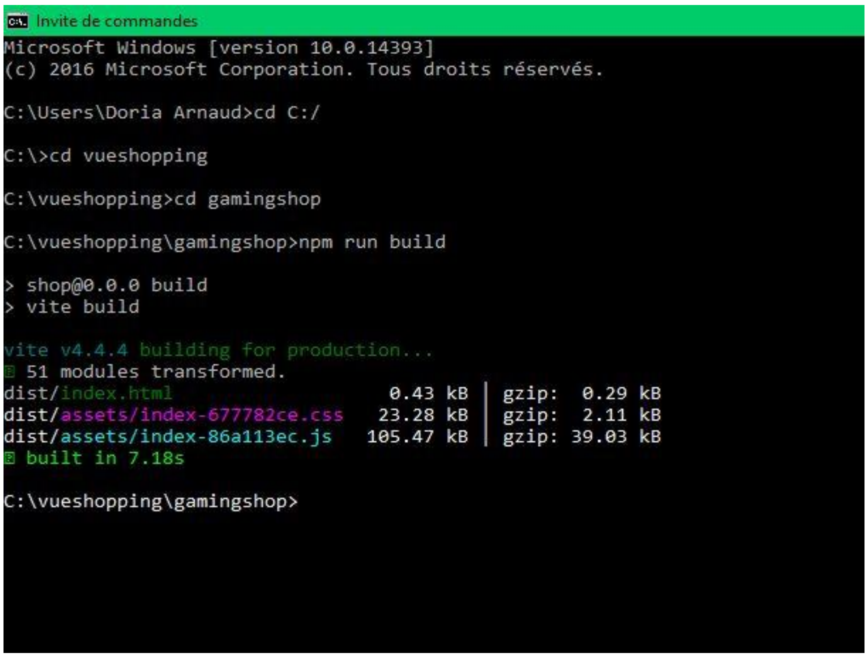
In this case, all the images path in this project are "/Products/Product1.png",… In the project tree in development, these can be find in the "public" folder. So the user could add his own images at this folder and delete all the old images, run the npm command "***npm run build***" and obtain a new distributable project file named "dist" on the same path of this entire project.

Please note: don't make a mistake about the image path. The user could put directly her own product images at the "public" folder, so the path in "product_tab" (marked with arrows) become: **"/<image_name>.<extension>"** or in a folder: **"/<image_folder>/<image_name>.<extension>"**

At the line 20, « directitemqtt » variable is used to initialize each item quantity appearing on the ItemView component

The seachtext variable is the v-model used by the search input field.

The "filteredlist" is the function returning the result matched with the input search field.

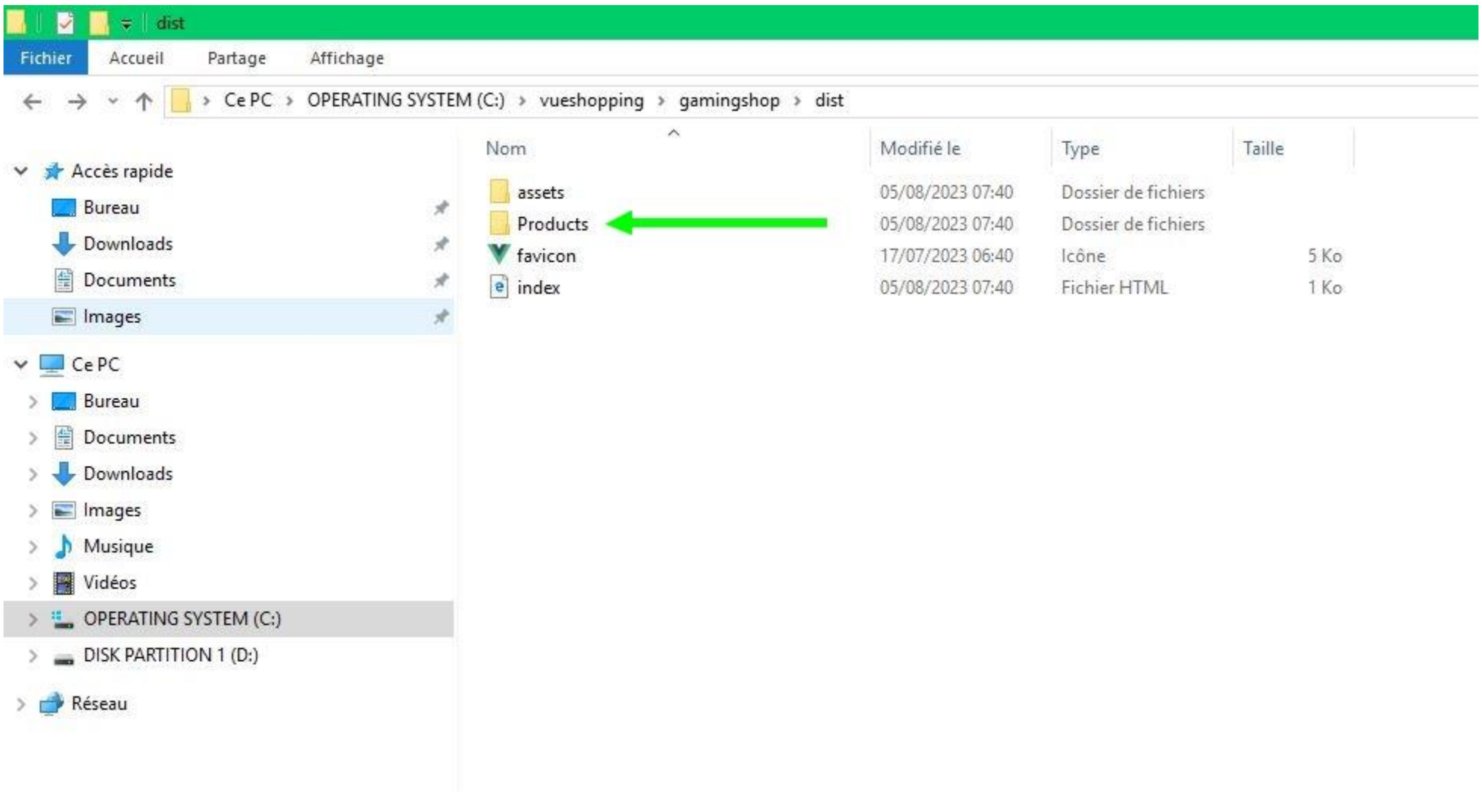


```
Microsoft Windows [version 10.0.14393]
(c) 2016 Microsoft Corporation. Tous droits réservés.

C:\Users\Doria Arnaud>cd C:/

C:\>cd vueshopping

C:\vueshopping>cd gamingshop

C:\vueshopping\gamingshop>npm run build

> shop@0.0.0 build
> vite build

vite v4.4.4 building for production...
✓ 51 modules transformed.
dist/index.html                   0.43 kB │ gzip:  0.29 kB
dist/assets/index-677782ce.css   23.28 kB │ gzip:  2.11 kB
dist/assets/index-86a113ec.js   105.47 kB │ gzip: 39.03 kB
✓ built in 7.18s

C:\vueshopping\gamingshop>
```

After all modifications made, the user can run again "***npm run build***" command to build the project. After, we can find the project minified at the same path that contains the entire project, this build directory is named "dist".

In this project, we had used npm package by Node.js and Vue/Cli. But we can use another like yarn,… for example.

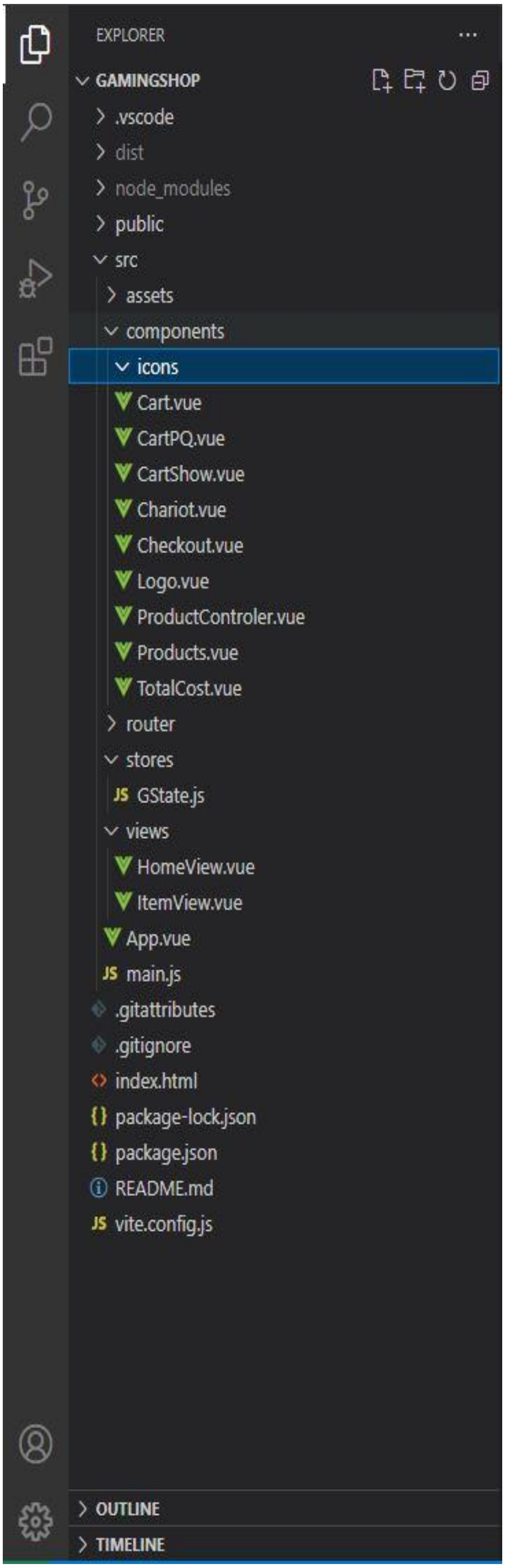
Make sure that we had installed these tools before.



Nom	Modifié le	Type	Taille
assets	05/08/2023 07:40	Dossier de fichiers	
Products	05/08/2023 07:40	Dossier de fichiers	
favicon	17/07/2023 06:40	Icône	5 Ko
index	05/08/2023 07:40	Fichier HTML	1 Ko

The contents of "dist" folder look like this after the user made any changes and ran "***npm run build***" in the terminal. All the product images are in "Products" after that. He can freely change all the images in this folder and accordingly make some changes in the path in the code. The project is now ready to deploy.

3-The Project tree:



```
EXPLORER
∨ GAMINGSHOP
  > .vscode
  > dist
  > node_modules
  > public
  ∨ src
    > assets
    ∨ components
      ∨ icons
      Cart.vue
      CartPQ.vue
      CartShow.vue
      Chariot.vue
      Checkout.vue
      Logo.vue
      ProductControler.vue
      Products.vue
      TotalCost.vue
    > router
    ∨ stores
      GState.js
    ∨ views
      HomeView.vue
      ItemView.vue
    App.vue
    main.js
  .gitattributes
  .gitignore
  index.html
  package-lock.json
  package.json
  README.md
  vite.config.js
> OUTLINE
> TIMELINE
```

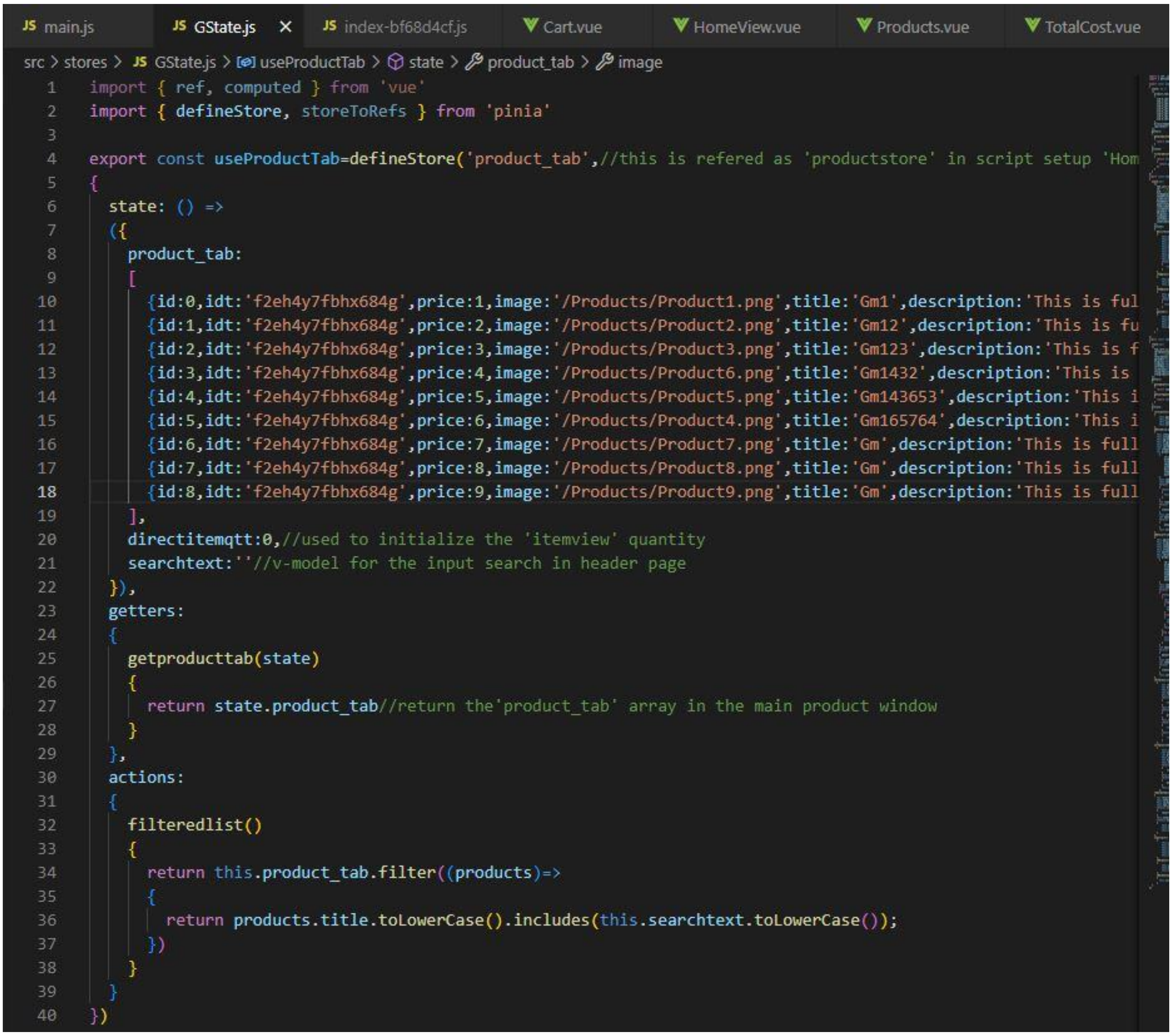
This is the representation of the project tree in Visual Studio code.

This project already contains a distributable file for production, build with npm package.

As we can see, all the components are stored in the “components” directory to build each components in the App, the router is in “router” for routing but this project is in Single Page Application, so the router target is just the HomeView component that we will see below. And the store is in “stores” directory for state management. In this case, we used Pinia, the official state manager for Vuejs application. Views are in the “views” directory.

4-Principal components:

- **The Store component : GState.js**



```
 1  import { ref, computed } from 'vue'
 2  import { defineStore, storeToRefs } from 'pinia'
 3  
 4  export const useProductTab=defineStore('product_tab',//this is refered as 'productstore' in script setup 'Hom
 5  {
 6    state: () =>
 7    ({
 8      product_tab:
 9      [
10        {id:0,idt:'f2eh4y7fbhx684g',price:1,image:'/Products/Product1.png',title:'Gm1',description:'This is ful
11        {id:1,idt:'f2eh4y7fbhx684g',price:2,image:'/Products/Product2.png',title:'Gm12',description:'This is fu
12        {id:2,idt:'f2eh4y7fbhx684g',price:3,image:'/Products/Product3.png',title:'Gm123',description:'This is f
13        {id:3,idt:'f2eh4y7fbhx684g',price:4,image:'/Products/Product6.png',title:'Gm1432',description:'This is
14        {id:4,idt:'f2eh4y7fbhx684g',price:5,image:'/Products/Product5.png',title:'Gm143653',description:'This i
15        {id:5,idt:'f2eh4y7fbhx684g',price:6,image:'/Products/Product4.png',title:'Gm165764',description:'This i
16        {id:6,idt:'f2eh4y7fbhx684g',price:7,image:'/Products/Product7.png',title:'Gm',description:'This is full
17        {id:7,idt:'f2eh4y7fbhx684g',price:8,image:'/Products/Product8.png',title:'Gm',description:'This is full
18        {id:8,idt:'f2eh4y7fbhx684g',price:9,image:'/Products/Product9.png',title:'Gm',description:'This is full
19      ],
20      directitemqtt:0,//used to initialize the 'itemview' quantity
21      searchtext:''//v-model for the input search in header page
22    }),
23    getters:
24    {
25      getproducttab(state)
26      {
27        return state.product_tab//return the'product_tab' array in the main product window
28      }
29    },
30    actions:
31    {
32      filteredlist()
33      {
34        return this.product_tab.filter((products)=>
35        {
36          return products.title.toLowerCase().includes(this.searchtext.toLowerCase());
37        })
38      }
39    }
40  })
```

This is the “GState.js” component. This is the fundamental component on this project. All data and method used by this application are there.

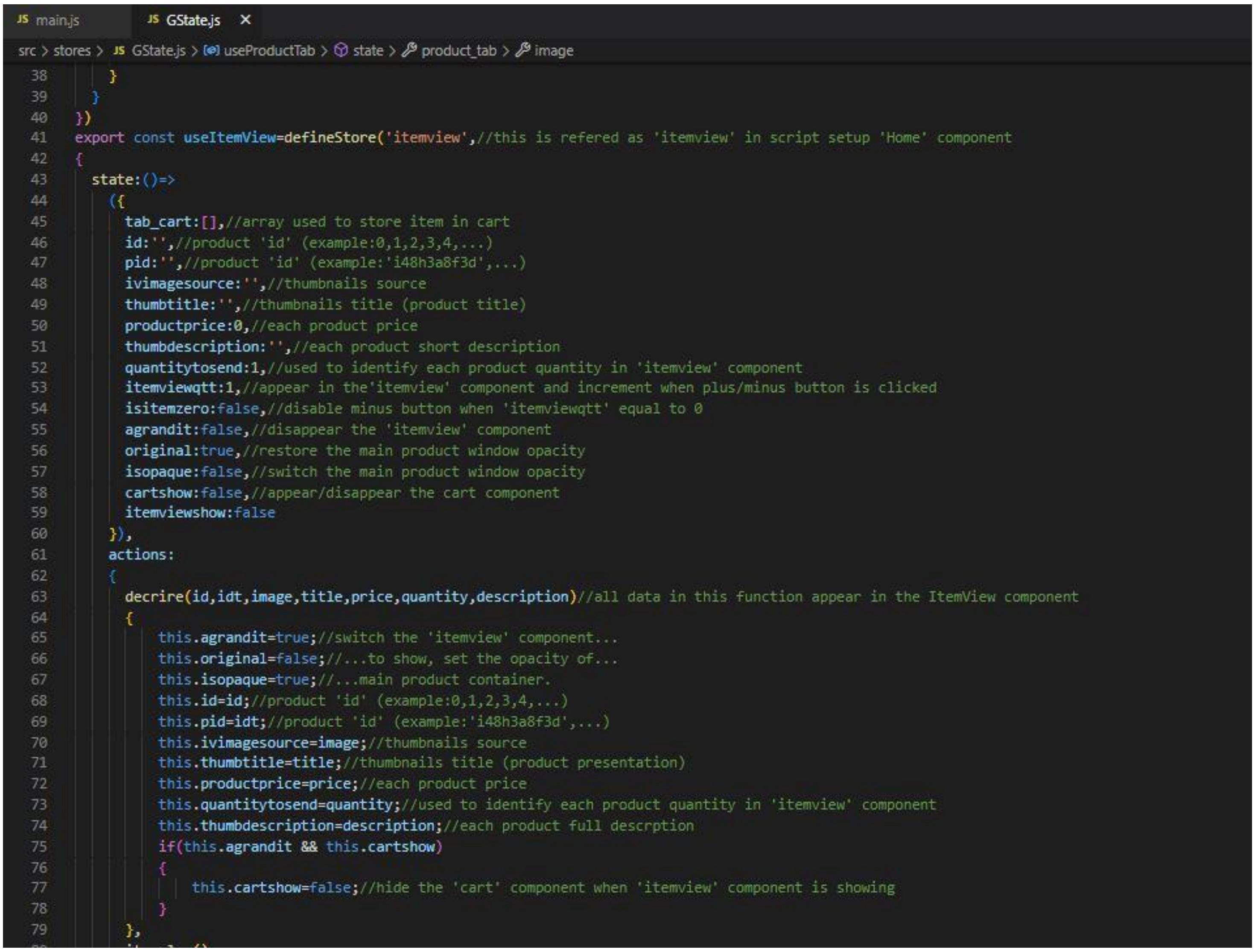
As seen on the picture, the state used here is an array declared as “product_tab”. The data could be fetched via an API too.

At the line 20, the “directitemqtt” will used by the “ItemView” component as item quantity.

“searchtext” : v-model of the input text field on the “HomeView” component.

“getproducttab”: function that return the product_tab array for the invoker.

“filteredlist”: function returning the matched result with “searchtext”.


main.js
GState.js
src > stores > GState.js > useProductTab > state > product_tab > image

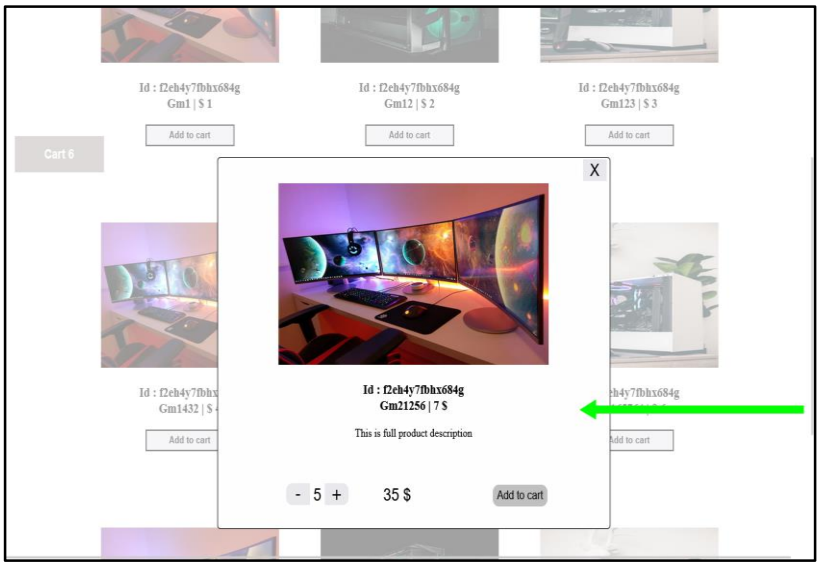
```javascript
    }
  }
})
export const useItemView=defineStore('itemview',//this is refered as 'itemview' in script setup 'Home' component
{
  state:()=>
    ({
      tab_cart:[],//array used to store item in cart
      id:'',//product 'id' (example:0,1,2,3,4,...)
      pid:'',//product 'id' (example:'i48h3a8f3d',...)
      ivimagesource:'',//thumbnails source
      thumbtitle:'',//thumbnails title (product title)
      productprice:0,//each product price
      thumbdescription:'',//each product short description
      quantitytosend:1,//used to identify each product quantity in 'itemview' component
      itemviewqtt:1,//appear in the'itemview' component and increment when plus/minus button is clicked
      isitemzero:false,//disable minus button when 'itemviewqtt' equal to 0
      agrandit:false,//disappear the 'itemview' component
      original:true,//restore the main product window opacity
      isopaque:false,//switch the main product window opacity
      cartshow:false,//appear/disappear the cart component
      itemviewshow:false
    }),
    actions:
    {
      decrire(id,idt,image,title,price,quantity,description)//all data in this function appear in the ItemView component
      {
          this.agrandit=true;//switch the 'itemview' component...
          this.original=false;//...to show, set the opacity of...
          this.isopaque=true;//...main product container.
          this.id=id;//product 'id' (example:0,1,2,3,4,...)
          this.pid=idt;//product 'id' (example:'i48h3a8f3d',...)
          this.ivimagesource=image;//thumbnails source
          this.thumbtitle=title;//thumbnails title (product presentation)
          this.productprice=price;//each product price
          this.quantitytosend=quantity;//used to identify each product quantity in 'itemview' component
          this.thumbdescription=description;//each product full descrption
          if(this.agrandit && this.cartshow)
          {
              this.cartshow=false;//hide the 'cart' component when 'itemview' component is showing
          }
      },
```

useItemView States:

- **"tab_cart"**: Array used as cart.
- **"id"**: each item identification (ex:0,1,2,3,4,...)
- **"pid"**: each item identification typed String. (ex: 34y793h78c0ff,..)
- **"ivimagesource"**: used for each image source of img element in template.
- **"thumbtitle"**: each image name typed String.
- **"productprice"**: the productprice. Type Number and could be a decimal
- **"thumbdescription"**: the full product description typed String.
- **"quantitytosend"**: a Number that used by the "ItemView" component that incremented/decremented by its plus/minus buttons.
- **"itemviewqtt"**: similar to "quantitytosend" but it is the Number appearing "visually" on the "ItemView" component, "quantitytosend" variable does the real work calculated by the program.
- **"isitemzero"**: Boolean used to switch minus button on "ItemView" component to be enabled/disabled.
- **"agrandit"**: Boolean for controlling the "ItemView" component to appear/disappear.
- **"original"**: Boolean for controlling the main product window opacity.

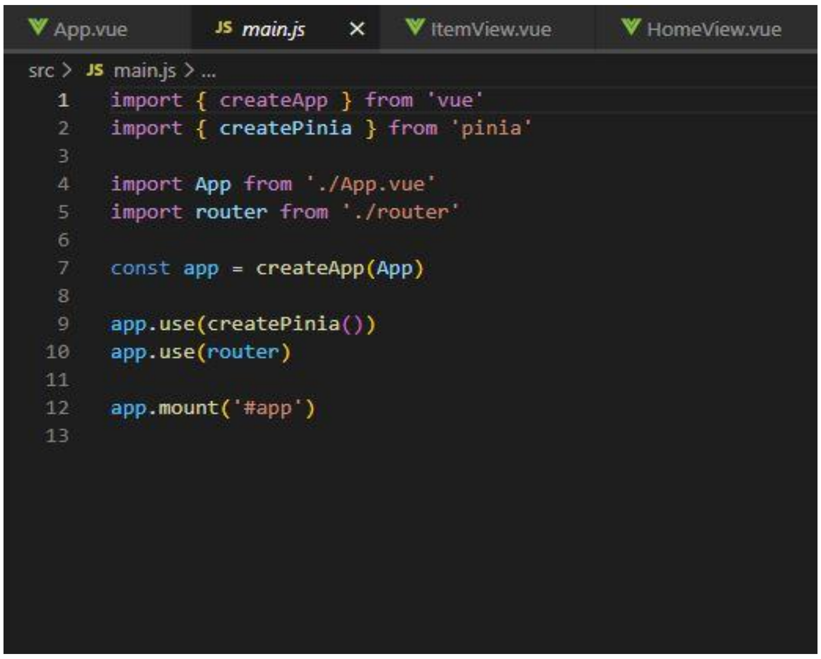
- **"isopaque"**: the same role as "original".
- **"cartshow"**: Boolean for controlling the cart component to appear/disappear.

All the arguments of “decrire” function are fetched directly in the “Products.vue” at the “HomeView” component. After, there are passed to these sate variables by the “decrire” function.



This is the « ItemView » Component

- **The Main.js component**



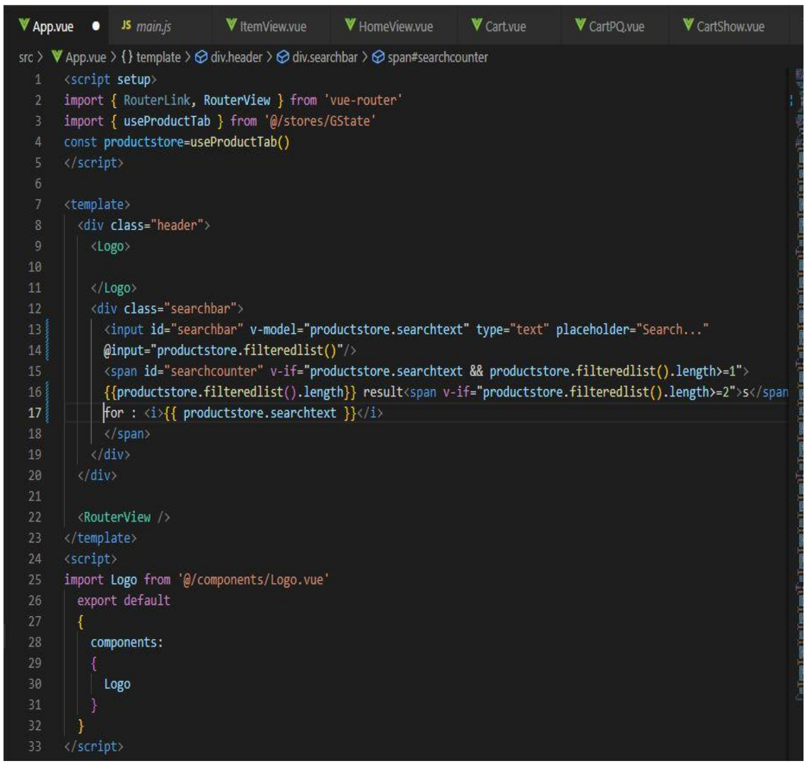
```
import { createApp } from 'vue'
import { createPinia } from 'pinia'

import App from './App.vue'
import router from './router'

const app = createApp(App)

app.use(createPinia())
app.use(router)

app.mount('#app')
```

This is the main.js component.

Used to create a Vue instance. Generally, this is the Application root.

As we can see, at the top line , the importation of “CreateApp” function to create a Vue instance and the “CreatePinia” to create an instance of Pinia that we use for the entire Project. This is the state manager that contains the data source, all the functions and methods used in the app.

- **The App.vue component**



```
App.vue  main.js  ItemView.vue  HomeView.vue  Cart.vue  CartPQ.vue  CartShow.vue
src > App.vue > {} template > div.header > div.searchbar > span#searchcounter
1   <script setup>
2   import { RouterLink, RouterView } from 'vue-router'
3   import { useProductTab } from '@/stores/GState'
4   const productstore=useProductTab()
5   </script>
6
7   <template>
8     <div class="header">
9       <Logo>
10
11      </Logo>
12      <div class="searchbar">
13        <input id="searchbar" v-model="productstore.searchtext" type="text" placeholder="Search..."
14        @input="productstore.filteredlist()"/>
15        <span id="searchcounter" v-if="productstore.searchtext && productstore.filteredlist().length>=1">
16        {{productstore.filteredlist().length}} result<span v-if="productstore.filteredlist().length>=2">s</span
17        for : <i>{{ productstore.searchtext }}</i>
18        </span>
19      </div>
20    </div>
21
22    <RouterView />
23  </template>
24  <script>
25  import Logo from '@/components/Logo.vue'
26    export default
27    {
28      components:
29      {
30        Logo
31      }
32    }
33  </script>
```

This is the App.vue component, appear in the main page. From line 1 to 5, we import the “RouterLink” and the ”RouterView” from “vue-router” to build the route within the app. At the line 3 and 4, this is the importation of “UseProductTab”, a reference in store to use all over the app once assigned to a constant. Here we use “producstore” for that.

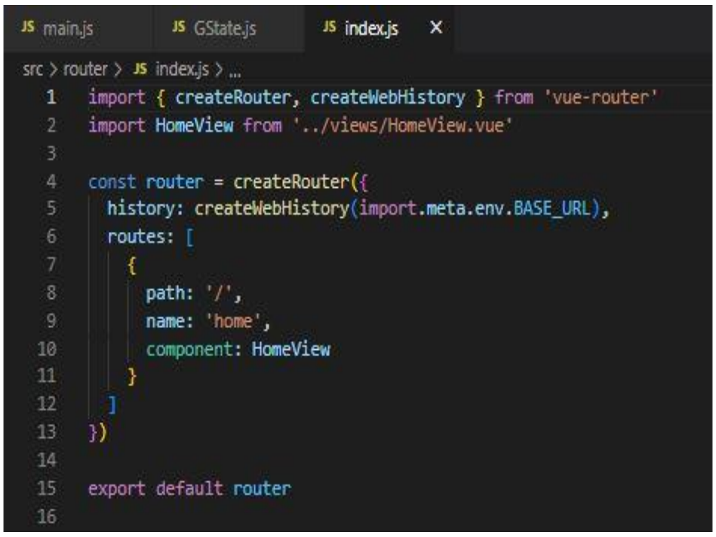
From the line 7 is beginning of the App.vue template. The “Logo” component is imported at the line 25 as local component.

After the “Logo” component to line 17, we use an input element for the search field.

The v-model directive of this input field use the “productstore” constant. The target is “searchtext”, a variable declared in the store referred by “useProductTab” state.

After that, we use a <span> element to show the result number once a text is entered in the input field and match an item in the “product_tab” array at the store. “filteredlist()” is the function called by the input event. The span element appear when searchtext variable return “true” and filteredlist length is more than 1.

- **The router component**



```
main.js  GState.js  index.js  ×
src > router > index.js > ...
1   import { createRouter, createWebHistory } from 'vue-router'
2   import HomeView from '../views/HomeView.vue'
3
4   const router = createRouter({
5     history: createWebHistory(import.meta.env.BASE_URL),
6     routes: [
7       {
8         path: '/',
9         name: 'home',
10        component: HomeView
11      }
12    ]
13  })
14
15  export default router
16
```

This is the router component that make link all over the application.

We don’t need many routes here because only the “HomeView” component is the target.

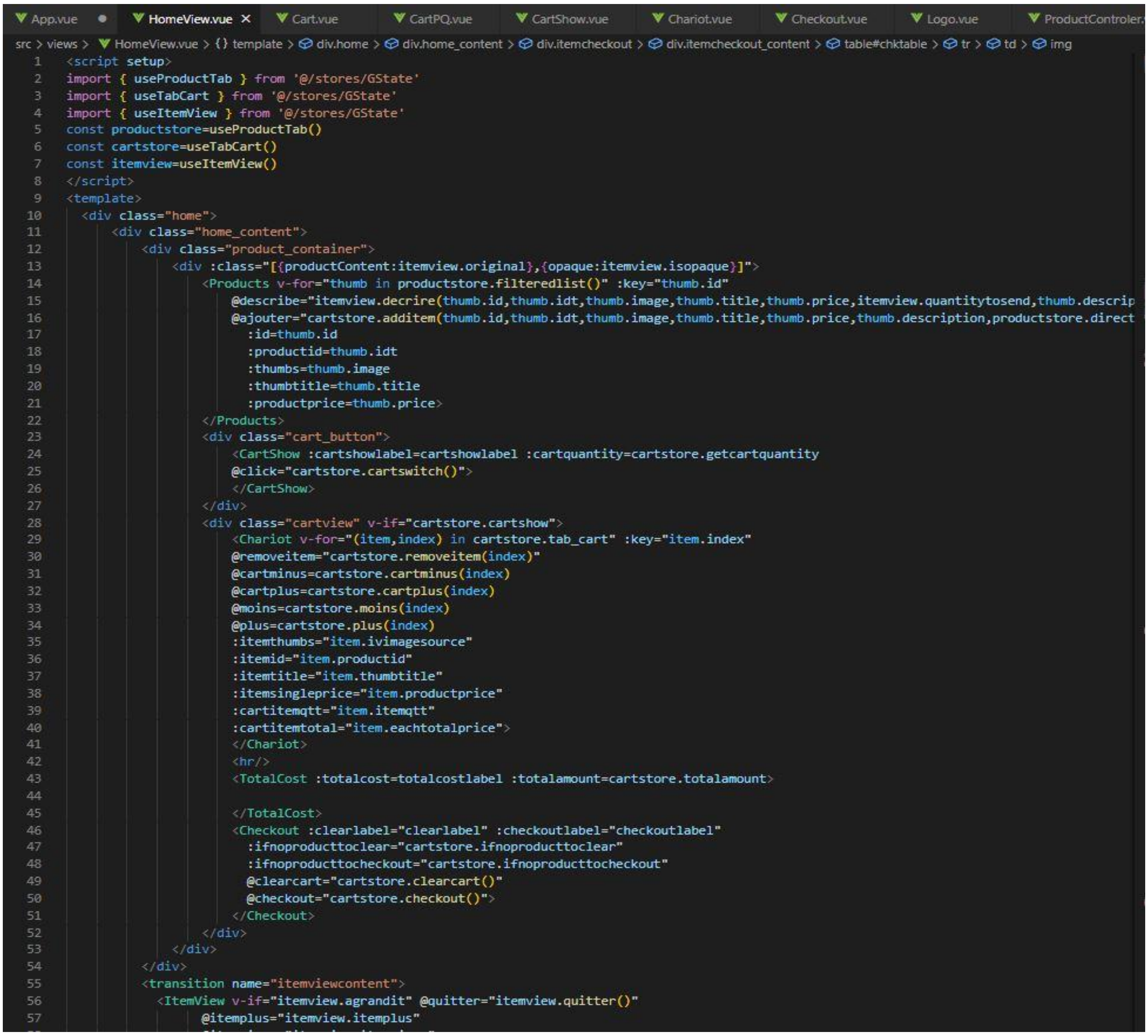
At the line 1, this is the importation of “CreateRouter” and the “createWebHistory” function to create routes.

At the second line, we import the “HomeView” component from Views to match the route.

At the line 4, this is how to create an instance of vue router. The “createRouter” function need history and the route path. At the line 8, our path is the main page as “/”, name “home” and use “HomeView” as local component.

5-Views components:

- **HomeView.vue Component:**



```
1  <script setup>
2  import { useProductTab } from '@/stores/GState'
3  import { useTabCart } from '@/stores/GState'
4  import { useItemView } from '@/stores/GState'
5  const productstore=useProductTab()
6  const cartstore=useTabCart()
7  const itemview=useItemView()
8  </script>
9  <template>
10   <div class="home">
11       <div class="home_content">
12           <div class="product_container">
13               <div :class="[{productContent:itemview.original},{opaque:itemview.isopaque}]">
14                   <Products v-for="thumb in productstore.filteredlist()" :key="thumb.id"
15                       @describe="itemview.decrire(thumb.id,thumb.idt,thumb.image,thumb.title,thumb.price,itemview.quantitytosend,thumb.descrip
16                       @ajouter="cartstore.additem(thumb.id,thumb.idt,thumb.image,thumb.title,thumb.price,thumb.description,productstore.direct
17                         :id=thumb.id
18                         :productid=thumb.idt
19                         :thumbs=thumb.image
20                         :thumbtitle=thumb.title
21                         :productprice=thumb.price>
22                   </Products>
23                   <div class="cart_button">
24                       <CartShow :cartshowlabel=cartshowlabel :cartquantity=cartstore.getcartquantity
25                       @click="cartstore.cartswitch()">
26                       </CartShow>
27                   </div>
28                   <div class="cartview" v-if="cartstore.cartshow">
29                       <Chariot v-for="(item,index) in cartstore.tab_cart" :key="item.index"
30                       @removeitem="cartstore.removeitem(index)"
31                       @cartminus=cartstore.cartminus(index)
32                       @cartplus=cartstore.cartplus(index)
33                       @moins=cartstore.moins(index)
34                       @plus=cartstore.plus(index)
35                       :itemthumbs="item.ivimagesource"
36                       :itemid="item.productid"
37                       :itemtitle="item.thumbtitle"
38                       :itemsingleprice="item.productprice"
39                       :cartitemqtt="item.itemqtt"
40                       :cartitemtotal="item.eachtotalprice">
41                       </Chariot>
42                       <hr/>
43                       <TotalCost :totalcost=totalcostlabel :totalamount=cartstore.totalamount>
44 
45                       </TotalCost>
46                       <Checkout :clearlabel="clearlabel" :checkoutlabel="checkoutlabel"
47                         :ifnoproducttoclear="cartstore.ifnoproducttoclear"
48                         :ifnoproducttocheckout="cartstore.ifnoproducttocheckout"
49                         @clearcart="cartstore.clearcart()"
50                         @checkout="cartstore.checkout()">
51                       </Checkout>
52                   </div>
53               </div>
54           </div>
55           <transition name="itemviewcontent">
56             <ItemView v-if="itemview.agrandit" @quitter="itemview.quitter()"
57                 @itemplus="itemview.itemplus"
```

This “HomeView” component is the main component of the project.

At the line 1 to 8, the script setup that contains importation of “useProductTab” from the store, that contains the array of object. The importation of “useTabCart”, and the useItemView”, all from the store named “GState”. After importation, we assigned these function into a constant and we can access these anywhere all over the application template.

At the line 13, these are classes switched by the two Boolean “original” and “isopaque” in the itemview store.

After that line, “Products” is the local component looping the product_tab array in the store and call the filteredlist function if a text is typed in input text field (the search input).

The “@describe” function receive many arguments that we use in “decrire” function at the store. “thumb” is the alias of “product_tab” array.

All the keys of this array are used by the “decrire” function and past these via props for the “ItemView” component.

The “@ajouter” function too is the same principle as “@describe” but it call the “additem” function in cartstore state.

The following are the props of Products component: (data binding)

- **id** : the “id” of each product in aray (0,1,2,3,….)
- **productid** : an example of “product identification” like (1n3e5sd7efg876,…)
- **thumbs** : the thumbnails used by the <img src=””> element in this component.
- **thumbtitle** : the name of each product
- **productprice** : each product price (we can use a decimal number)

This <Products></Products> component is the responsible of appearance of each products in the “Homeview” component.

After the products component is the the <CartShow> component. This appear as a button on the main window.

It take “cartshowlabel” and “cartquantity” as props, respectively the “Cart” label could be changed to cart icon and the quantity number of the cart as number. This quantity is fetched in cartstore state.

At the line 29 to 41, this is the Chariot component as cart. It loops over the tab_cart array in the cartsore state and fetch all the data. The item and index are respectively the alias of tab_cart aray and index of each item.

The removeitem function use the index as argument to react with each remove button action. This function is in the cartstore state.

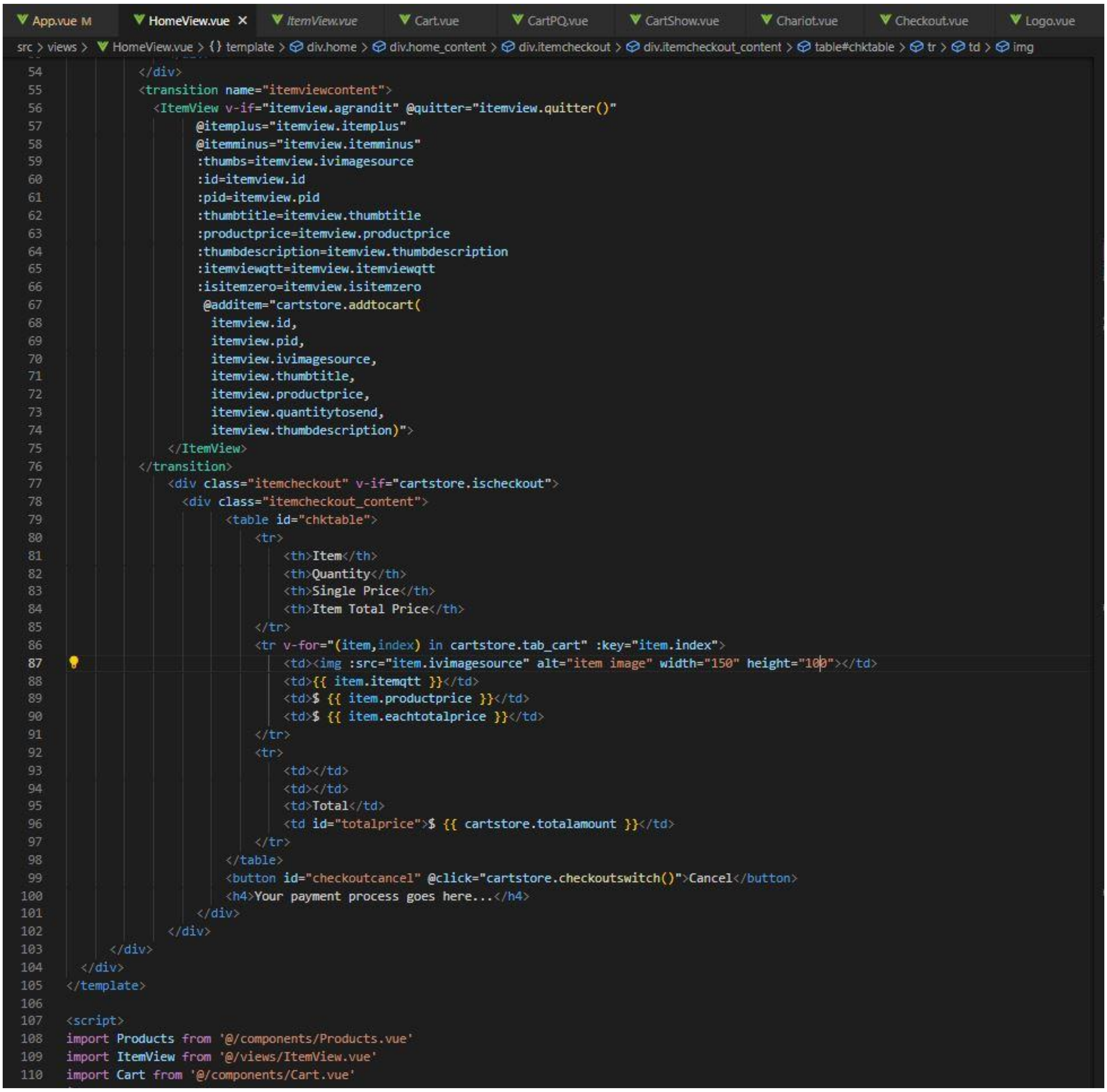
The cartplus, cartminus, itemplus and itemminus functions react with the plus and minus buttons on each item in cart.

All these functions use the useTabCart state that we can find in GState.js

At the line 43, we have the totalcost component that appear in the bottom of cart component. It takes totalcostlabel and totalamount props respectively icon and the total amount of all items in cart as number, fetched in the useTabCart state.

At the line 46, this is the checkout component. It takes clearlabel, checkoutlabel as labels of the clear cart button and the checkout button. Next, two Boolean ifnoprducttoclear and ifnoproducttocheckout for disabling these button if no product are in the cart.

The @clearcart and @checkout functions call the clearcart and checkout functions referenced by cartstore in the useTabCart state respectively to clear the cart and appear the checkout component.


App.vue M
HomeView.vue
ItemView.vue
Cart.vue
CartPQ.vue
CartShow.vue
Chariot.vue
Checkout.vue
Logo.vue
src > views > HomeView.vue > {} template > div.home > div.home_content > div.itemcheckout > div.itemcheckout_content > table#chktable > tr > td > img

```
 54          </div>
 55          <transition name="itemviewcontent">
 56            <ItemView v-if="itemview.agrandit" @quitter="itemview.quitter()"
 57                  @itemplus="itemview.itemplus"
 58                  @itemminus="itemview.itemminus"
 59                  :thumbs=itemview.ivimagesource
 60                  :id=itemview.id
 61                  :pid=itemview.pid
 62                  :thumbtitle=itemview.thumbtitle
 63                  :productprice=itemview.productprice
 64                  :thumbdescription=itemview.thumbdescription
 65                  :itemviewqtt=itemview.itemviewqtt
 66                  :isitemzero=itemview.isitemzero
 67                   @additem="cartstore.addtocart(
 68                    itemview.id,
 69                    itemview.pid,
 70                    itemview.ivimagesource,
 71                    itemview.thumbtitle,
 72                    itemview.productprice,
 73                    itemview.quantitytosend,
 74                    itemview.thumbdescription)">
 75              </ItemView>
 76          </transition>
 77              <div class="itemcheckout" v-if="cartstore.ischeckout">
 78                <div class="itemcheckout_content">
 79                      <table id="chktable">
 80                          <tr>
 81                              <th>Item</th>
 82                              <th>Quantity</th>
 83                              <th>Single Price</th>
 84                              <th>Item Total Price</th>
 85                          </tr>
 86                          <tr v-for="(item,index) in cartstore.tab_cart" :key="item.index">
 87                              <td><img :src="item.ivimagesource" alt="item image" width="150" height="100"></td>
 88                              <td>{{ item.itemqtt }}</td>
 89                              <td>$ {{ item.productprice }}</td>
 90                              <td>$ {{ item.eachtotalprice }}</td>
 91                          </tr>
 92                          <tr>
 93                              <td></td>
 94                              <td></td>
 95                              <td>Total</td>
 96                              <td id="totalprice">$ {{ cartstore.totalamount }}</td>
 97                          </tr>
 98                      </table>
 99                      <button id="checkoutcancel" @click="cartstore.checkoutswitch()">Cancel</button>
100                      <h4>Your payment process goes here...</h4>
101                  </div>
102              </div>
103      </div>
104    </div>
105  </template>
106
107  <script>
108  import Products from '@/components/Products.vue'
109  import ItemView from '@/views/ItemView.vue'
110  import Cart from '@/components/Cart.vue'
```

This portion of code follow the code above.

From the line 55 to 76, this is the ItemView component. At the top line, this is a sort of animation provided by vuejs.

This component appear if the ”agrandit” Boolean in useItemView store returns true. The close button calls a method named “quitter” in this same store.

The @itemplus and @itemminus at line 57, 58 calls the methods to increment and decrement quantity shown between the plus and minus buttons on this same component.

Following are the props used by “ItemView” component of each product details:

- id: Number representing each product id fetched in the array of product (0,1,2,3,4,…)
- pid: String that used as product identification (ex: 45n673t2542e7,…)
- thumbtitle: String used as an image source of img element in this component

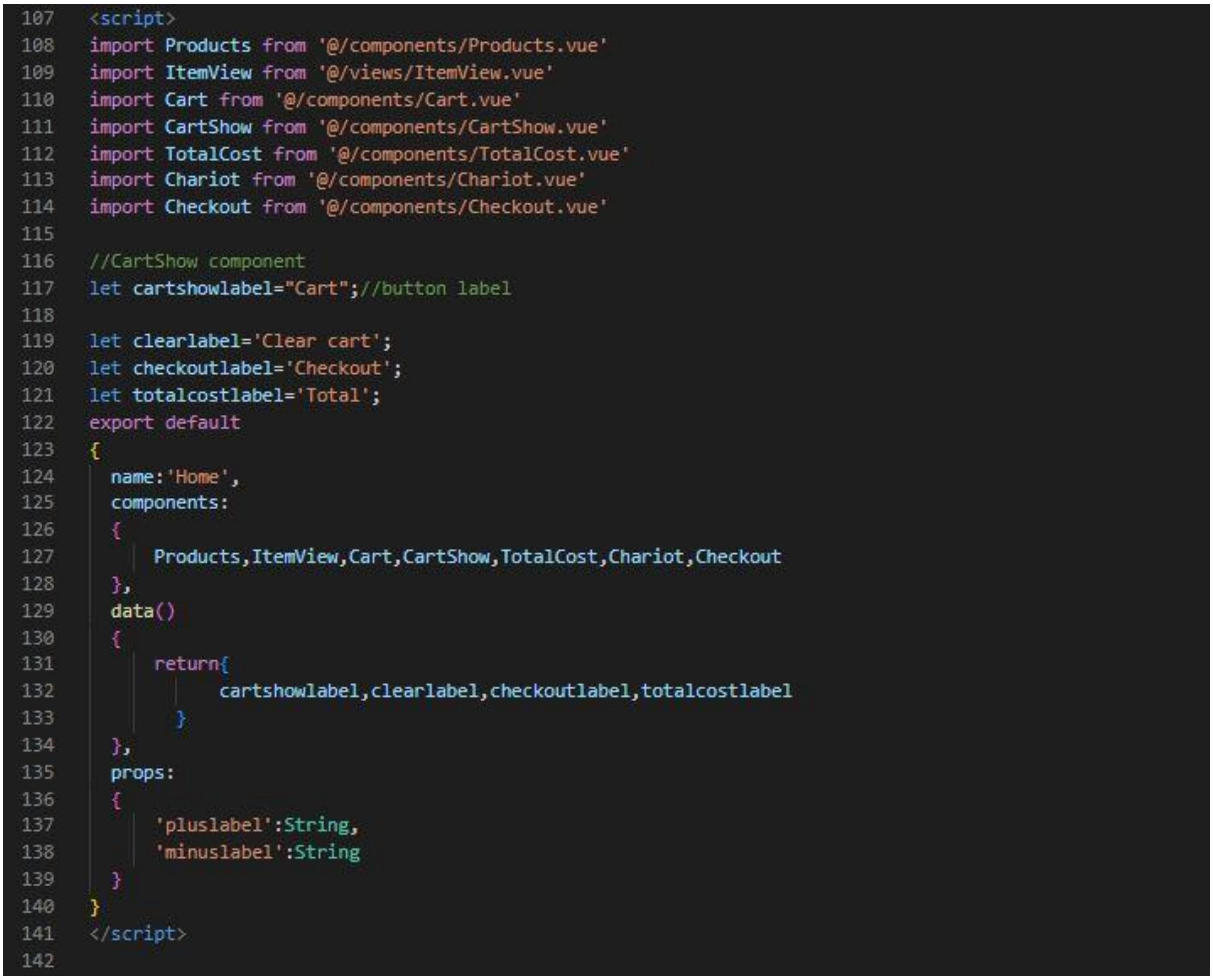
- productprice: Number that represent each product price. Note that a decimal number can be used too.
- thumbdescription: String representing the full product description.
- Itemviewqtt: Number used as each product quantity. This is sent by the argument of "decrire" method at the "useTabCart" store and after that, this component take the control of this quantity and resend this.
- isitemzero: Boolean used to switch the minus button to enable/disable if item quantity is more than 1 or equal to 0. This Boolean is passed from here to the children that will see later.

The @additem at the line 67 to 74 calls the "addtocart" method in "useProductTab" store. All the arguments of this function are these props seen below.

This method adds each product appearing with all parameters in the "tab_cart" array in the store once the "Add to cart" button is clicked.

From the line 77 to 102 is the "itemcheckout" component. This appear once the Checkout button is clicked. His role is looping over the tab_cart array in the store and fetch all data of each product and calculate the total amount of these products.

Now, let's talk about the rest of the "HomeView" script code.

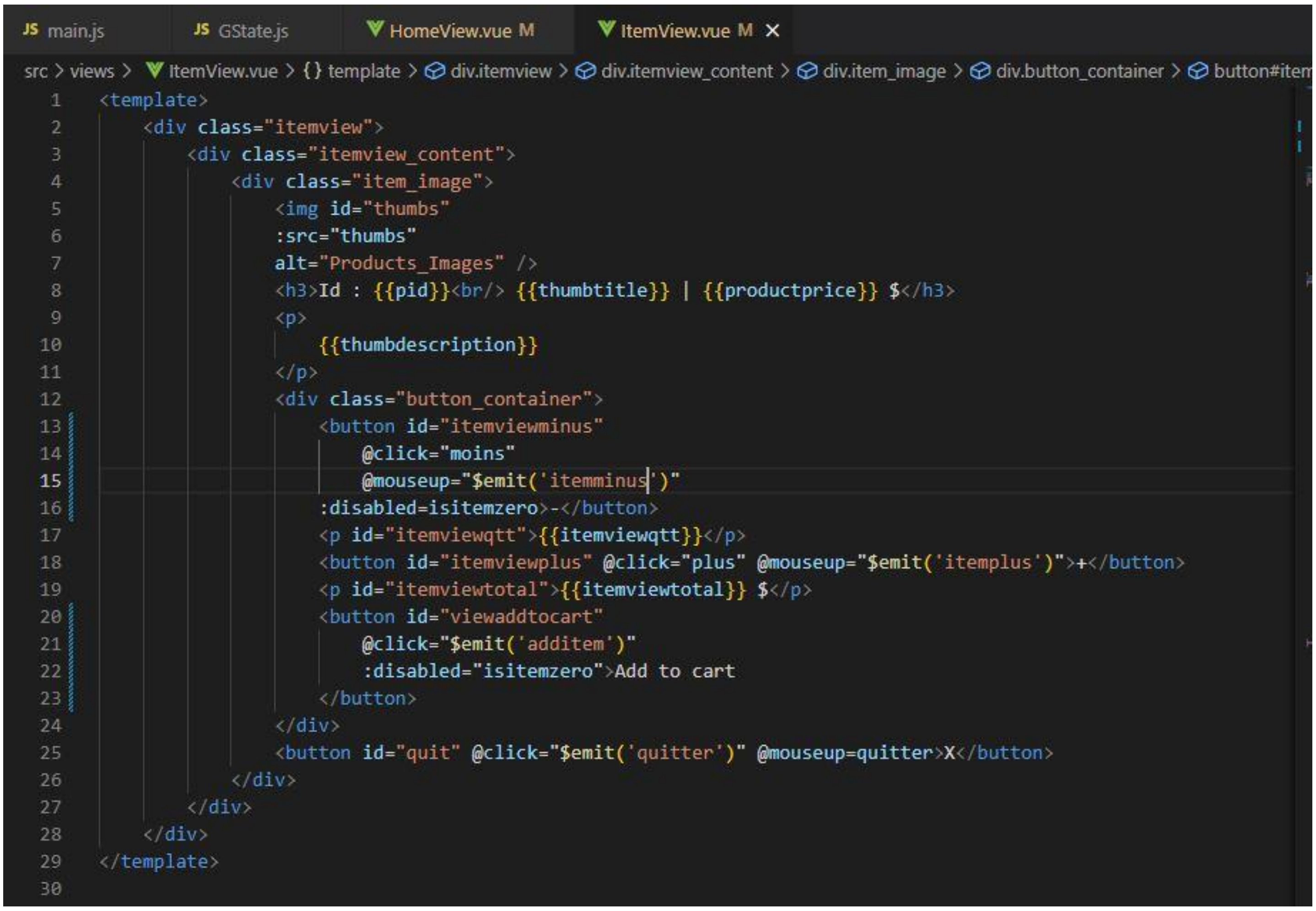


```
107   <script>
108   import Products from '@/components/Products.vue'
109   import ItemView from '@/views/ItemView.vue'
110   import Cart from '@/components/Cart.vue'
111   import CartShow from '@/components/CartShow.vue'
112   import TotalCost from '@/components/TotalCost.vue'
113   import Chariot from '@/components/Chariot.vue'
114   import Checkout from '@/components/Checkout.vue'
115
116   //CartShow component
117   let cartshowlabel="Cart";//button label
118
119   let clearlabel='Clear cart';
120   let checkoutlabel='Checkout';
121   let totalcostlabel='Total';
122   export default
123   {
124     name:'Home',
125     components:
126     {
127         Products,ItemView,Cart,CartShow,TotalCost,Chariot,Checkout
128     },
129     data()
130     {
131         return{
132               cartshowlabel,clearlabel,checkoutlabel,totalcostlabel
133           }
134     },
135     props:
136     {
137         'pluslabel':String,
138         'minuslabel':String
139     }
140   }
141   </script>
142
```

From the line 108 to 114 are the importation of all components used by the HomeView. The "cartshowlabel" is the label of the cart button. Note that this can be modified to a cart icon. The clearlabel, checkoutlabel, totalcostlabel are the labels of buttons.

At the line 127 is the local component registration. These components are used by the parent which is the "HomeView" here.

- **ItemView.vue Component:**


main.js
GState.js
HomeView.vue M
ItemView.vue M
src > views > ItemView.vue > {} template > div.itemview > div.itemview_content > div.item_image > div.button_container > button#item

```
1   <template>
2       <div class="itemview">
3           <div class="itemview_content">
4               <div class="item_image">
5                   <img id="thumbs"
6                   :src="thumbs"
7                   alt="Products_Images" />
8                   <h3>Id : {{pid}}<br/> {{thumbtitle}} | {{productprice}} $</h3>
9                   <p>
10                      {{thumbdescription}}
11                  </p>
12                  <div class="button_container">
13                      <button id="itemviewminus"
14                          @click="moins"
15                          @mouseup="$emit('itemminus')"
16                      :disabled=isitemzero>-</button>
17                      <p id="itemviewqtt">{{itemviewqtt}}</p>
18                      <button id="itemviewplus" @click="plus" @mouseup="$emit('itemplus')">+</button>
19                      <p id="itemviewtotal">{{itemviewtotal}} $</p>
20                      <button id="viewaddtocart"
21                          @click="$emit('additem')"
22                          :disabled="isitemzero">Add to cart
23                      </button>
24                  </div>
25                  <button id="quit" @click="$emit('quitter')" @mouseup=quitter>X</button>
26              </div>
27          </div>
28      </div>
29  </template>
30
```

This is the itemview component. From the line 4 to 7 is the iamge element that contains src binded with thumbs props see in the explication below. At the line 8, this is a h3 element that takes pid, thumbtitle, productprice and one paragraph element below this, had thumbdecription as props.

From the line 12 to 16 is the minus button that calls "moins" method and have one more attribute which is the @mouseup event. It emit an event called "itemminus" to his parent "HomeView". And this "HomeView" component calls the "itemminus" method in the store.

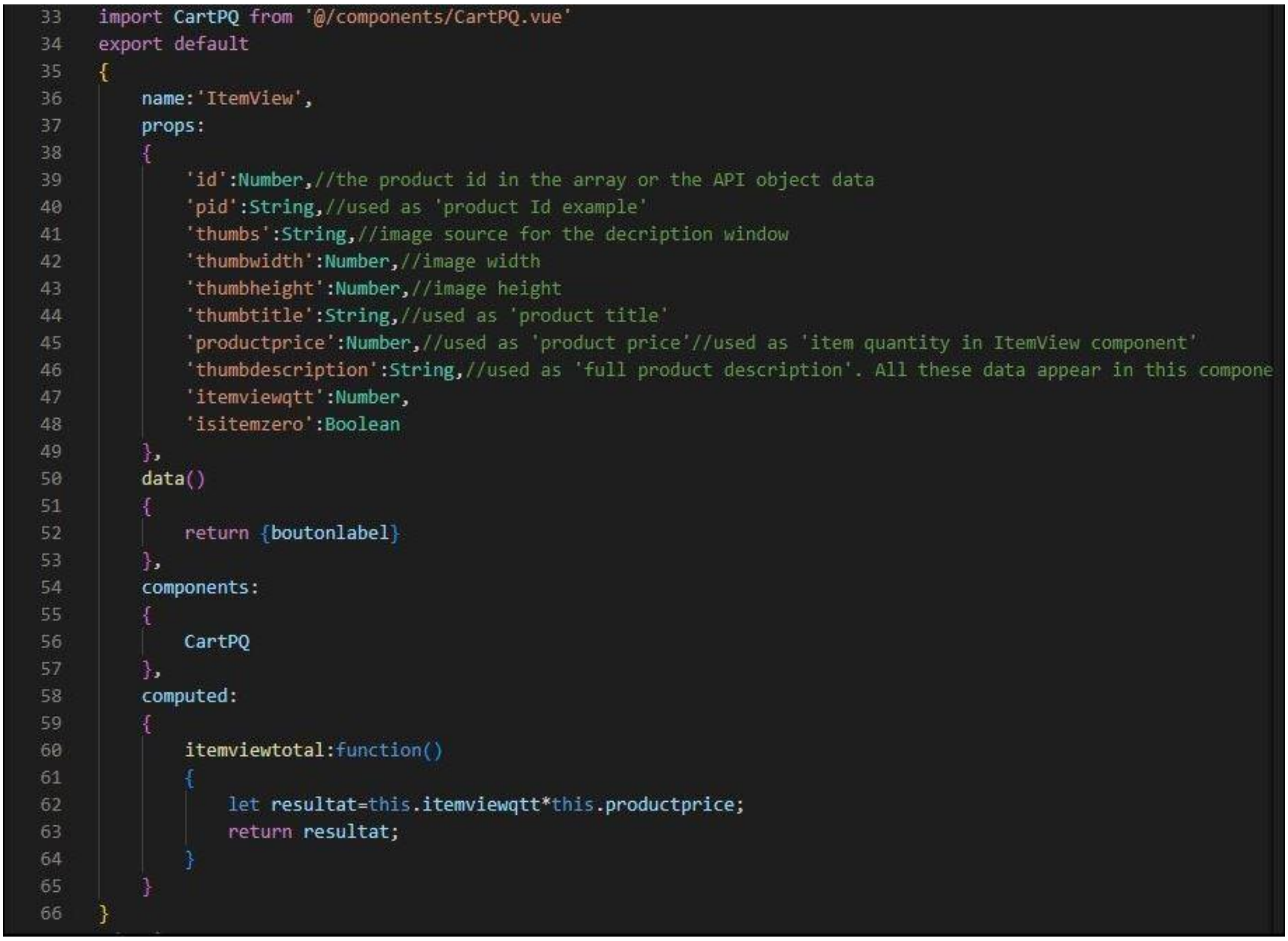
The line 17 is a paragraph element that show item quantity when plus or minus button is clicked. Next, this quantity is used to be added as wach product quantity in the "tab_cart" array at the store.

The line 18 is the plus button that has the same functionality as the minus button on line 12 above.

After that, at the line 19, this is a paragraph element that showing the total price of each item in this "ItemView" component in real time.

The button element on the line 20 is the "add to cart" button. When this is clicked, it emits the "additem" event to the parent that is the "HomeView" component.

At the line 25, this is the "quit" button. His role is to disappear the itemview component when clicked. The parent is always "HomeView", listening the "quitter" event.



```
33  import CartPQ from '@/components/CartPQ.vue'
34  export default
35  {
36      name:'ItemView',
37      props:
38      {
39          'id':Number,//the product id in the array or the API object data
40          'pid':String,//used as 'product Id example'
41          'thumbs':String,//image source for the decription window
42          'thumbwidth':Number,//image width
43          'thumbheight':Number,//image height
44          'thumbtitle':String,//used as 'product title'
45          'productprice':Number,//used as 'product price'//used as 'item quantity in ItemView component'
46          'thumbdescription':String,//used as 'full product description'. All these data appear in this compone
47          'itemviewqtt':Number,
48          'isitemzero':Boolean
49      },
50      data()
51      {
52          return {boutonlabel}
53      },
54      components:
55      {
56          CartPQ
57      },
58      computed:
59      {
60          itemviewtotal:function()
61          {
62              let resultat=this.itemviewqtt*this.productprice;
63              return resultat;
64          }
65      }
66  }
```

This portion of code follow the code above. At the line 33, this is the importation of "CartPQ" component from the components directory.

Following are the props used by the itemview component:

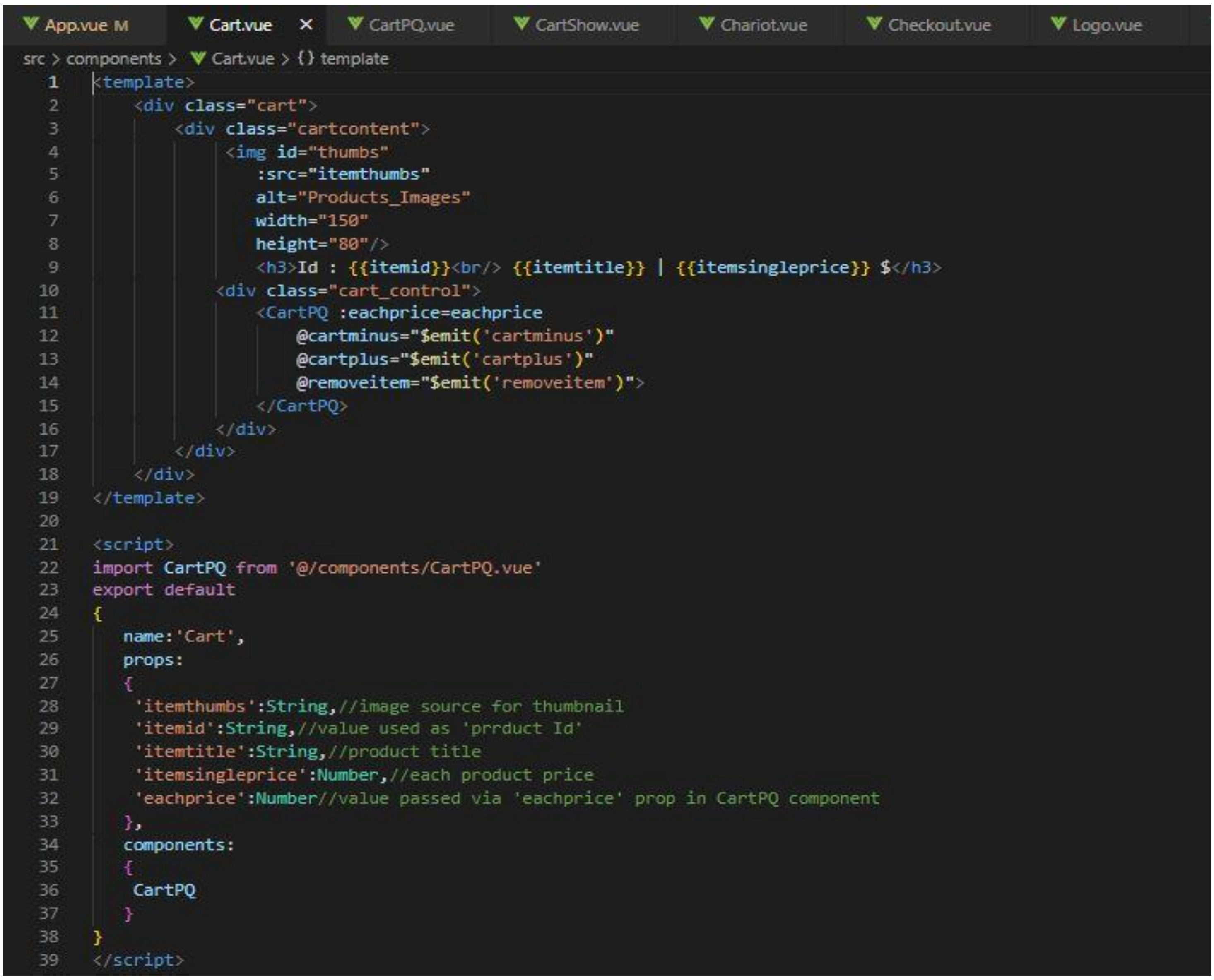
- **id**: Number received from the "decrire" method seen earlier (ex : 0,1,2,3,4,…)
- **pid**: String that represent an example of item identification (ex: 34xl4f7e8c9g,…)
- **thumbwidth**: Number that represent the width of the image (ex: 100,200,300,…) used by the width attribute of image element.
- **thumbheight**: Number that represent the height of the image (ex: 100,200,300,…) used by the height attribute of image element.
- **thumbtitle**: String used as the name of each product (ex: PC Ram 8gigs, Gpu,…). This is an element returned by the search input field but could be changed with another array keys of "product_tab" in the store, may be the id, the description.
- **productprice**: Number used as each product price. Could be a decimal number.
- **thumbdescription**: String representing each product description.
- **itemviewqtt**: Number used as each item quantity. This will be the item quantity argument for the "additem" function.
- **isitemzero**: Boolean that is used to enable/disable the "add to cart" button when itemviewqtt equal to 0,and to avoid negative value of this quantity.

At the line 60, this is a computed property containing itemviewtotal. This function calculate each item total price in real time on this component.

All these props seen above are used by the "additem" method as argument to be added to the "tab_cart" array in the store.

Let’s talk about all child component in this application.

- **Cart.vue Component:**



```
 1  <template>
 2      <div class="cart">
 3          <div class="cartcontent">
 4              <img id="thumbs"
 5                 :src="itemthumbs"
 6                 alt="Products_Images"
 7                 width="150"
 8                 height="80"/>
 9                 <h3>Id : {{itemid}}<br/> {{itemtitle}} | {{itemsingleprice}} $</h3>
10             <div class="cart_control">
11                 <CartPQ :eachprice=eachprice
12                     @cartminus="$emit('cartminus')"
13                     @cartplus="$emit('cartplus')"
14                     @removeitem="$emit('removeitem')">
15                 </CartPQ>
16             </div>
17         </div>
18     </div>
19 </template>
20
21 <script>
22 import CartPQ from '@/components/CartPQ.vue'
23 export default
24 {
25    name:'Cart',
26    props:
27    {
28     'itemthumbs':String,//image source for thumbnail
29     'itemid':String,//value used as 'prrduct Id'
30     'itemtitle':String,//product title
31     'itemsingleprice':Number,//each product price
32     'eachprice':Number//value passed via 'eachprice' prop in CartPQ component
33    },
34    components:
35    {
36     CartPQ
37    }
38 }
39 </script>
```

This is the Cart.vue component. From the line 4 to 8 is the image element, followed by an h3 that take itemid, itemtitle and itemsingleprice as props. All these props functionalities will see below. At the line 11 to 14 is the CartPQ component that has eachprice as props, @cartminus emitting cartminus event, @cartplus emitting cartplus event and @removeitem emitting removeitem event. All these are listened by the parent component which is the “HomeView” and this parent send there to each methods in the store.

At the line 22 is the importation of cartPQ component. This is the child component of cart.vue here.

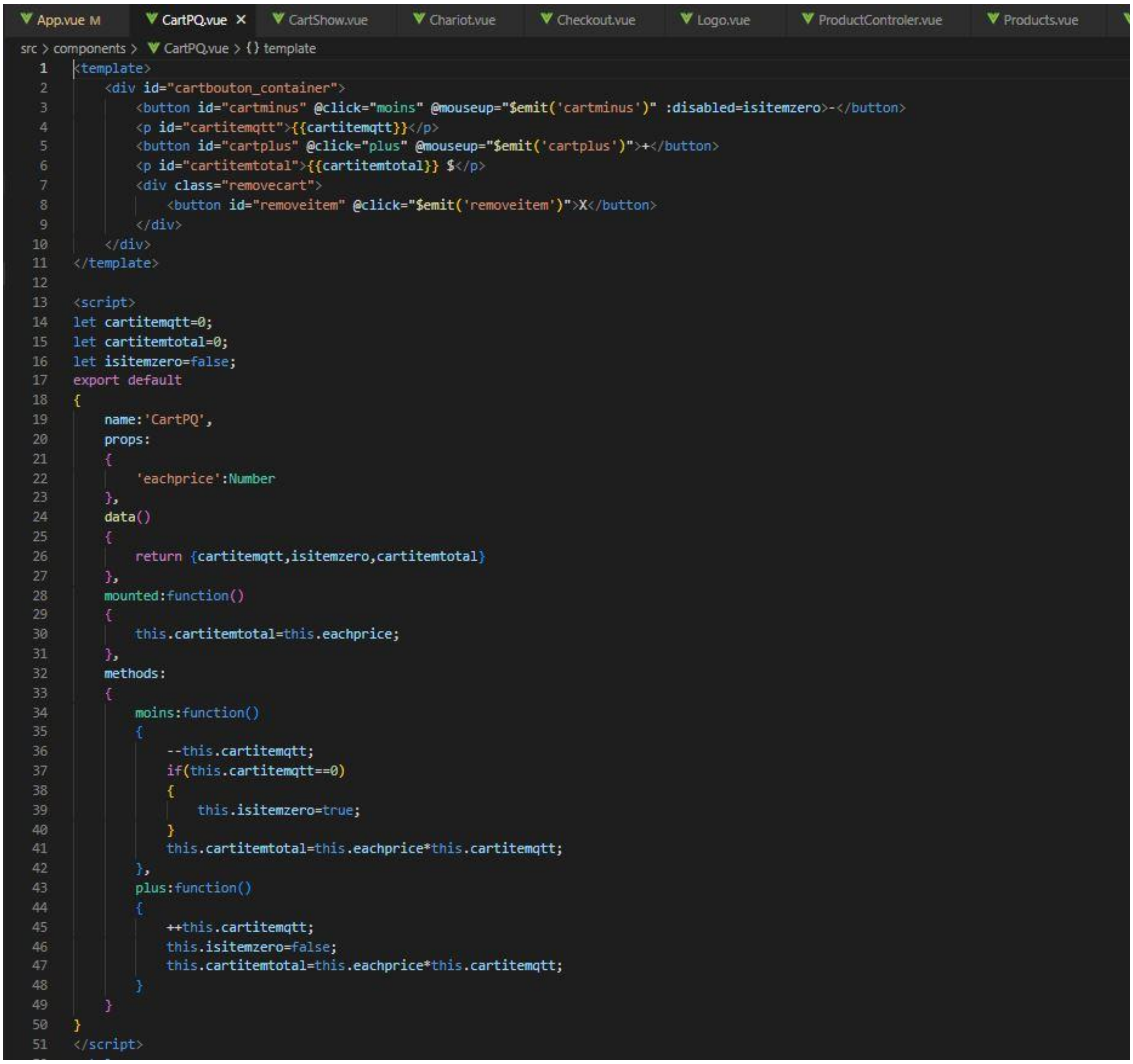
Following are the props and each functionalities:

- **src**: String used as the image source of img element. This is fetched within the arguments of “additem” and “addtocart” functions in the store.
- **itemid**: String that represent item identification (ex: 34h7j8g90hk,…)
- **itemtitle**: String used as the name of each product (ex: PC ram, GPU, CPU,…)
- **itemsingleprice**: Number used as each item price (ex: 4.33, 5, 7, 1.233,…)

@cartminus is the event emitted by CartPQ component but there, it was re-emit by Cart.vue component to be listened by the parent which is “Homeview”. @cartplus and @removeitem have the same functionalities as @cartminus.

All these props are fetched by using the arguments of “additem” and “Addtocart” functions in the store.

- **CartPQ.vue Component:**



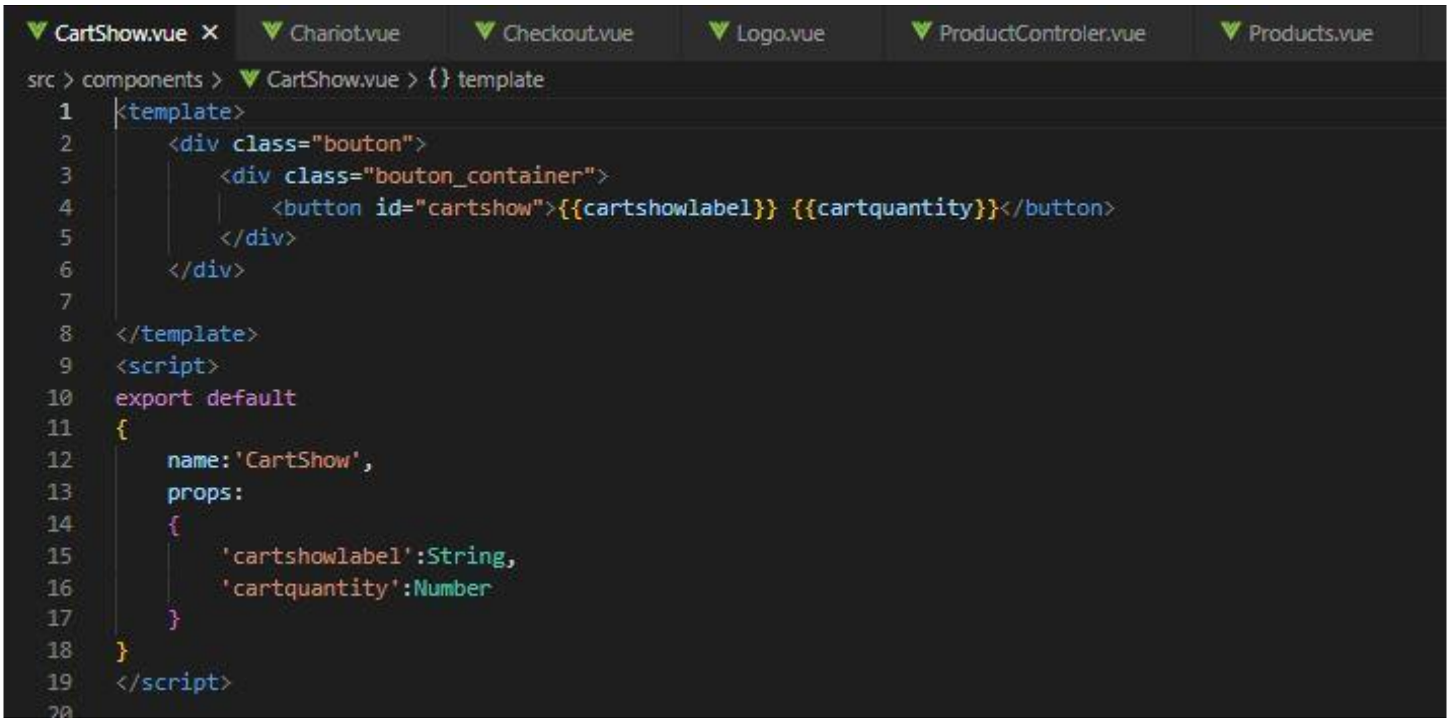
```
<template>
    <div id="cartbouton_container">
        <button id="cartminus" @click="moins" @mouseup="$emit('cartminus')" :disabled=isitemzero>-</button>
        <p id="cartitemqtt">{{cartitemqtt}}</p>
        <button id="cartplus" @click="plus" @mouseup="$emit('cartplus')">+</button>
        <p id="cartitemtotal">{{cartitemtotal}} $</p>
        <div class="removecart">
            <button id="removeitem" @click="$emit('removeitem')">X</button>
        </div>
    </div>
</template>

<script>
let cartitemqtt=0;
let cartitemtotal=0;
let isitemzero=false;
export default
{
    name:'CartPQ',
    props:
    {
        'eachprice':Number
    },
    data()
    {
        return {cartitemqtt,isitemzero,cartitemtotal}
    },
    mounted:function()
    {
        this.cartitemtotal=this.eachprice;
    },
    methods:
    {
        moins:function()
        {
            --this.cartitemqtt;
            if(this.cartitemqtt==0)
            {
                this.isitemzero=true;
            }
            this.cartitemtotal=this.eachprice*this.cartitemqtt;
        },
        plus:function()
        {
            ++this.cartitemqtt;
            this.isitemzero=false;
            this.cartitemtotal=this.eachprice*this.cartitemqtt;
        }
    }
}
</script>
```

The CartPQ component is a child component of Cart.vue.

- **@click=”moins”**: calls the moins method for itemqtt decrementing. In another word this method update itemqtt which is a number used as item quantity.
- **@mouseup=”$emit(‘cartminus’)”**: emit cartminus event to the parent component “HomeView” and calls the cartminus method in the store.
- **cartitemqtt props**: this is a props that used as item quantity for each product in cart.

- **@click=”plus”** and **@mouseup=”$emit(‘cartplus’)”**: respectively calls the plus function to increment “cartitemqtt” and the second is emitting cartplus event to be listened by the parent “HomeView”.
- **cartitemtotal**: Number used as each item total. This is obtained by multiplying itemqtt above by each item single price.
- **@click=”$emit(‘removitem’)”**: emit removeitem event and calls the removeitem function in the store passing through “HomeView”.
- **isitemzero**: Boolean to switch the minus button to enable/disable when itemqtt equal to 0 to avoid negative value.

- **CartShow.vue Component:**



```
<template>
    <div class="bouton">
        <div class="bouton_container">
            <button id="cartshow">{{cartshowlabel}} {{cartquantity}}</button>
        </div>
    </div>

</template>
<script>
export default
{
    name:'CartShow',
    props:
    {
        'cartshowlabel':String,
        'cartquantity':Number
    }
}
</script>
```

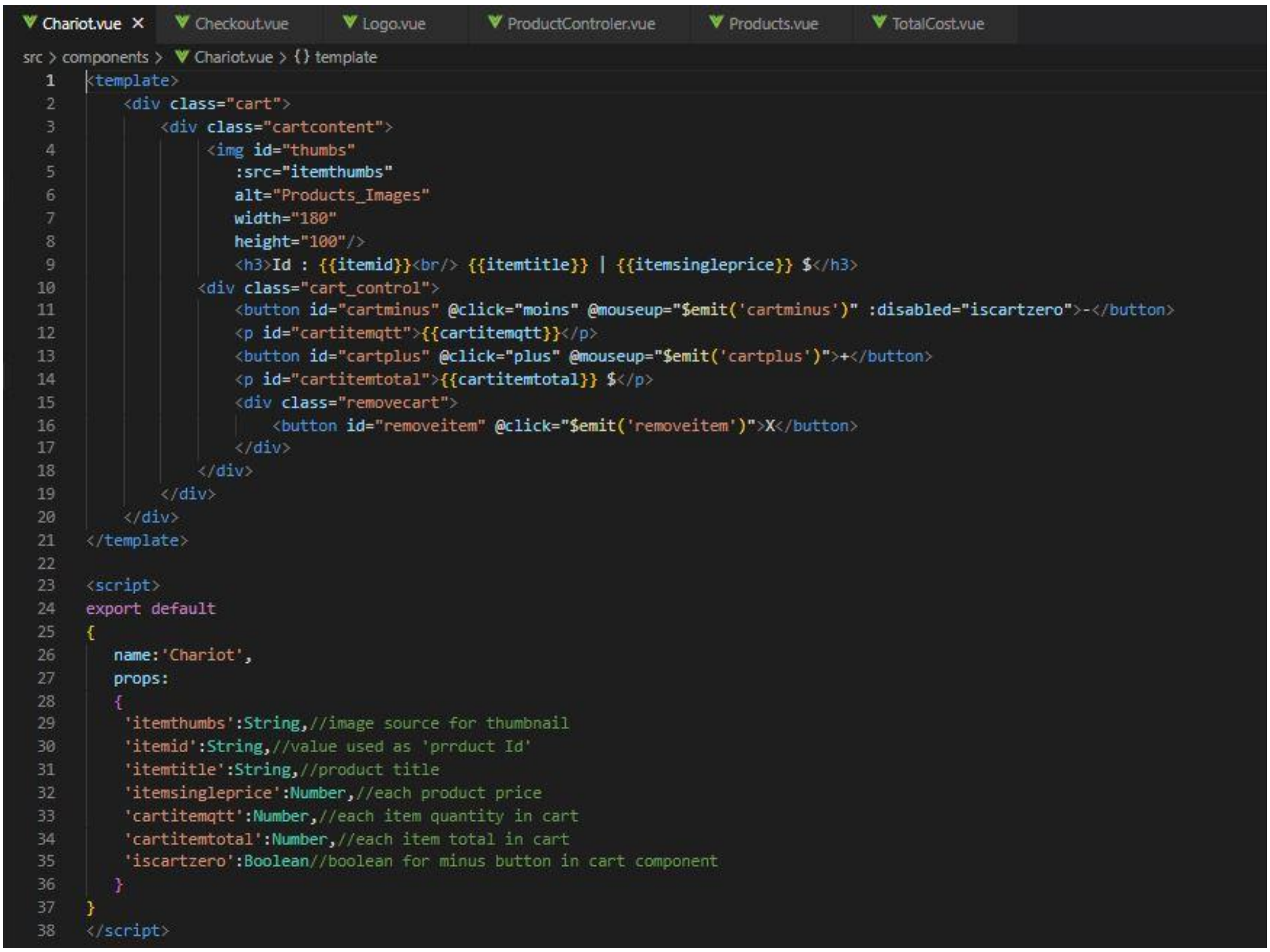
CartShow is the component showing the cart quantity.

Props:

cartsholabel: String props used as label of this component. Could be changed into cart image icon.

cartquantity: Number that represent the global cart quantity.

- **Chariot.vue Component:**

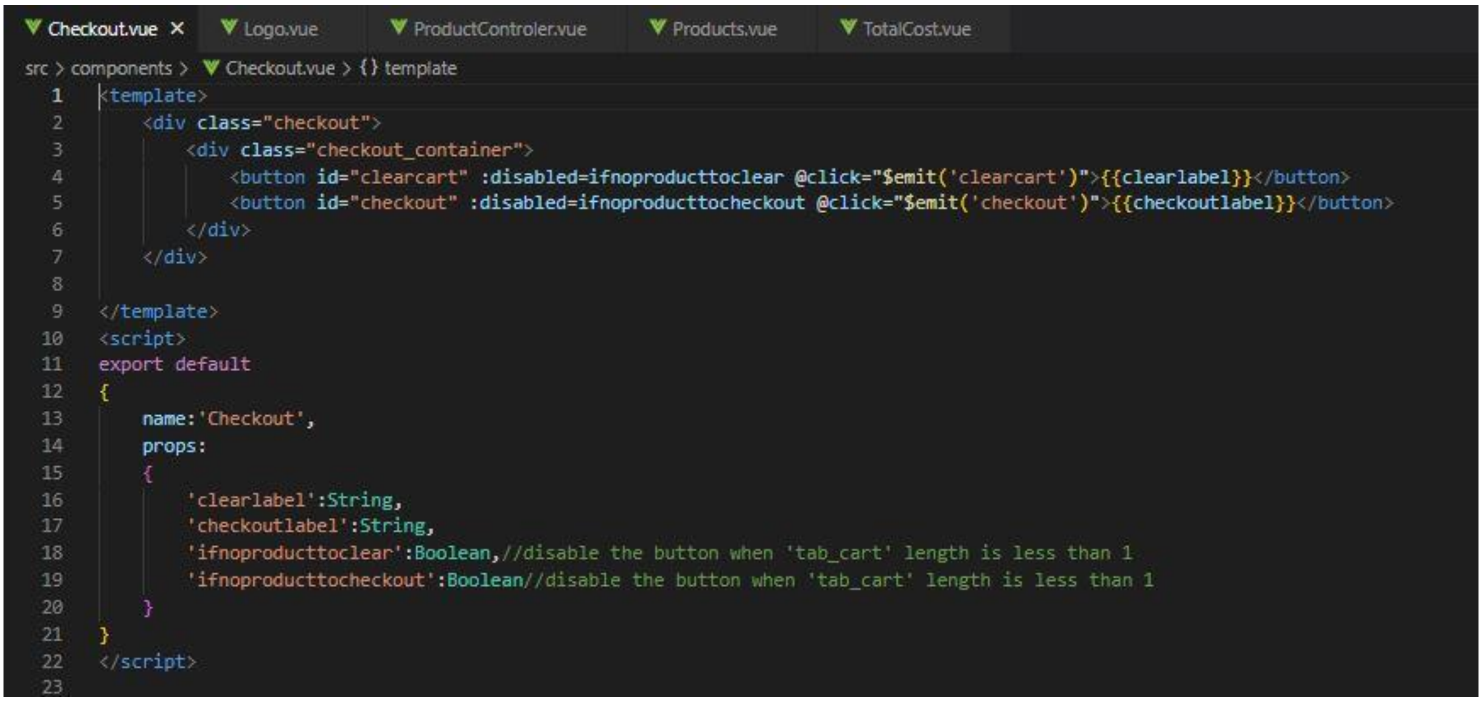


```
<template>
    <div class="cart">
        <div class="cartcontent">
            <img id="thumbs"
              :src="itemthumbs"
              alt="Products_Images"
              width="180"
              height="100"/>
              <h3>Id : {{itemid}}<br/> {{itemtitle}} | {{itemsingleprice}} $</h3>
          <div class="cart_control">
              <button id="cartminus" @click="moins" @mouseup="$emit('cartminus')" :disabled="iscartzero">-</button>
              <p id="cartitemqtt">{{cartitemqtt}}</p>
              <button id="cartplus" @click="plus" @mouseup="$emit('cartplus')">+</button>
              <p id="cartitemtotal">{{cartitemtotal}} $</p>
              <div class="removecart">
                  <button id="removeitem" @click="$emit('removeitem')">X</button>
              </div>
          </div>
        </div>
    </div>
</template>

<script>
export default
{
   name:'Chariot',
   props:
   {
    'itemthumbs':String,//image source for thumbnail
    'itemid':String,//value used as 'prrduct Id'
    'itemtitle':String,//product title
    'itemsingleprice':Number,//each product price
    'cartitemqtt':Number,//each item quantity in cart
    'cartitemtotal':Number,//each item total in cart
    'iscartzero':Boolean//boolean for minus button in cart component
   }
}
</script>
```

This is the Chariot.vue, used as a cart in this project.

- **src="itemthumbs"**: "itemthumbs" is the props for image source of img element.
- **itemid**: String that represent as an example of item identification (ex: 45ef782g9j,…)
- **itemtitle**: String used by the name of each product.
- **itemsingleprice**: Number used as each item price.
- **@click="moins"** and **@mouseup="$emit('cartminus')"**: respectively calls the "moins" method to decrement "cartitemqtt" and emit "cartminus" event to the parent component.
- **@click="plus"** and **@mouseup="$emit('cartminus')"** : respectively calls the "plus" method to decrement "cartitemqtt" and emit "cartplus" event to the parent component.
- **@click="$emit('removeitem')"** : emit "removeitem" event to the parent component and calls the "removeitem" function in the store.
- **iscartzero** : Boolean used to enable/disable the minus button to avoid negative value.

- **Chariot.vue Component:**



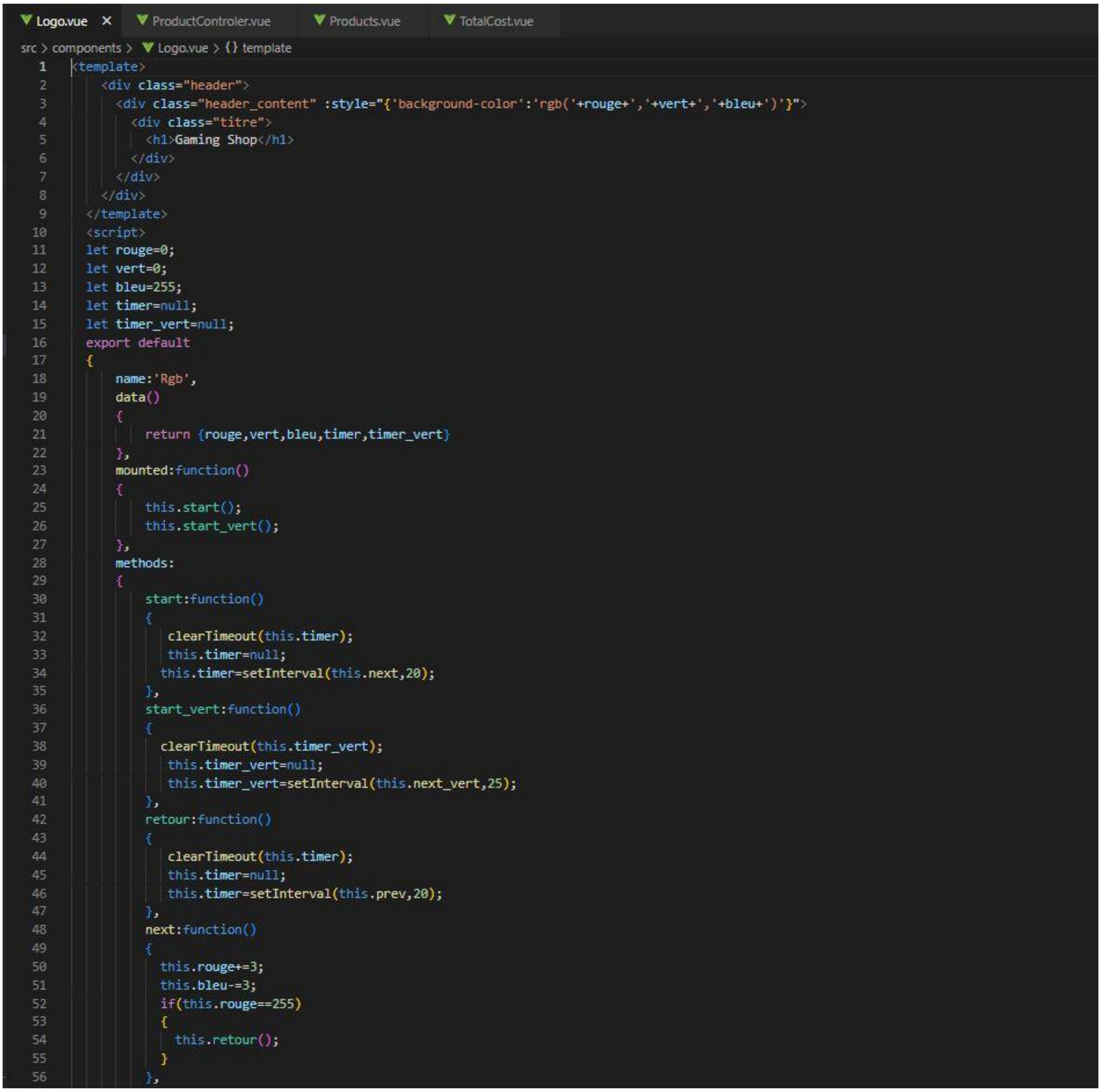
```
<template>
    <div class="checkout">
        <div class="checkout_container">
            <button id="clearcart" :disabled=ifnoproducttoclear @click="$emit('clearcart')">{{clearlabel}}</button>
            <button id="checkout" :disabled=ifnoproducttocheckout @click="$emit('checkout')">{{checkoutlabel}}</button>
        </div>
    </div>

</template>
<script>
export default
{
    name:'Checkout',
    props:
    {
        'clearlabel':String,
        'checkoutlabel':String,
        'ifnoproducttoclear':Boolean,//disable the button when 'tab_cart' length is less than 1
        'ifnoproducttocheckout':Boolean//disable the button when 'tab_cart' length is less than 1
    }
}
</script>

```

This is the “Checkout” component.

Props and attributes:

- **ifnoproducttoclear**: Boolean that used to disable the clear cart button if there is no product in the cart.
- **@click=”$emit(‘clearcart’)”**: attribute that emit “clearcart” event to be listened by “HomeView” and calls the “clearcart” function in the store.
- **ifnoproducttocheckout**: Boolean that used to disable the clear cart button if there is no product in the cart.
- **@click=”$emit(‘checkout’)”**: attribute that emit “checkout” event to be listened by “HomeView” and calls the “checkout” function in the store.
- **clearlabel**: used as button label of clear cart button. Could be changed as clear icon.
- **checkoutlabel**: used as button label of checkout button.

- **Logo.vue Component:**



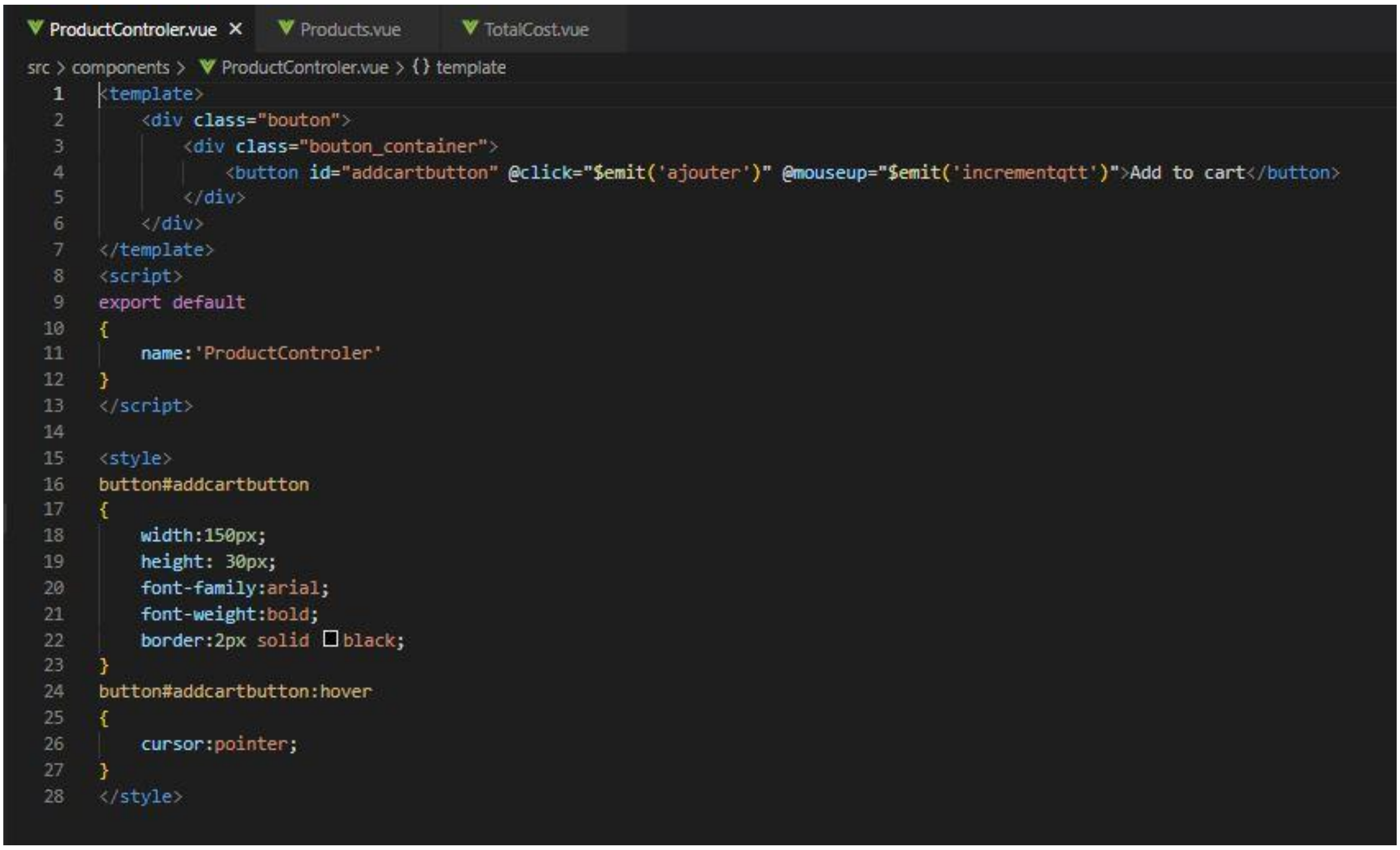
```
<template>
    <div class="header">
      <div class="header_content" :style="{'background-color':'rgb('+rouge+','+vert+','+bleu+')'}">
        <div class="titre">
          <h1>Gaming Shop</h1>
        </div>
      </div>
    </div>
  </template>
  <script>
  let rouge=0;
  let vert=0;
  let bleu=255;
  let timer=null;
  let timer_vert=null;
  export default
  {
      name:'Rgb',
      data()
      {
          return {rouge,vert,bleu,timer,timer_vert}
      },
      mounted:function()
      {
          this.start();
          this.start_vert();
      },
      methods:
      {
          start:function()
          {
             clearTimeout(this.timer);
             this.timer=null;
            this.timer=setInterval(this.next,20);
          },
          start_vert:function()
          {
            clearTimeout(this.timer_vert);
             this.timer_vert=null;
             this.timer_vert=setInterval(this.next_vert,25);
          },
          retour:function()
          {
             clearTimeout(this.timer);
             this.timer=null;
             this.timer=setInterval(this.prev,20);
          },
          next:function()
          {
            this.rouge+=3;
            this.bleu-=3;
            if(this.rouge==255)
            {
              this.retour();
            }
          },
```

This is the "Logo.vue" component.

The style attribute is binded here with a mixing of red, blue and green color.

Once this component is mounted, the "start" and "start_vert" method are called and invoke each other to vary the intensity of red, blue and green color.

- **ProductControler.vue Component:**

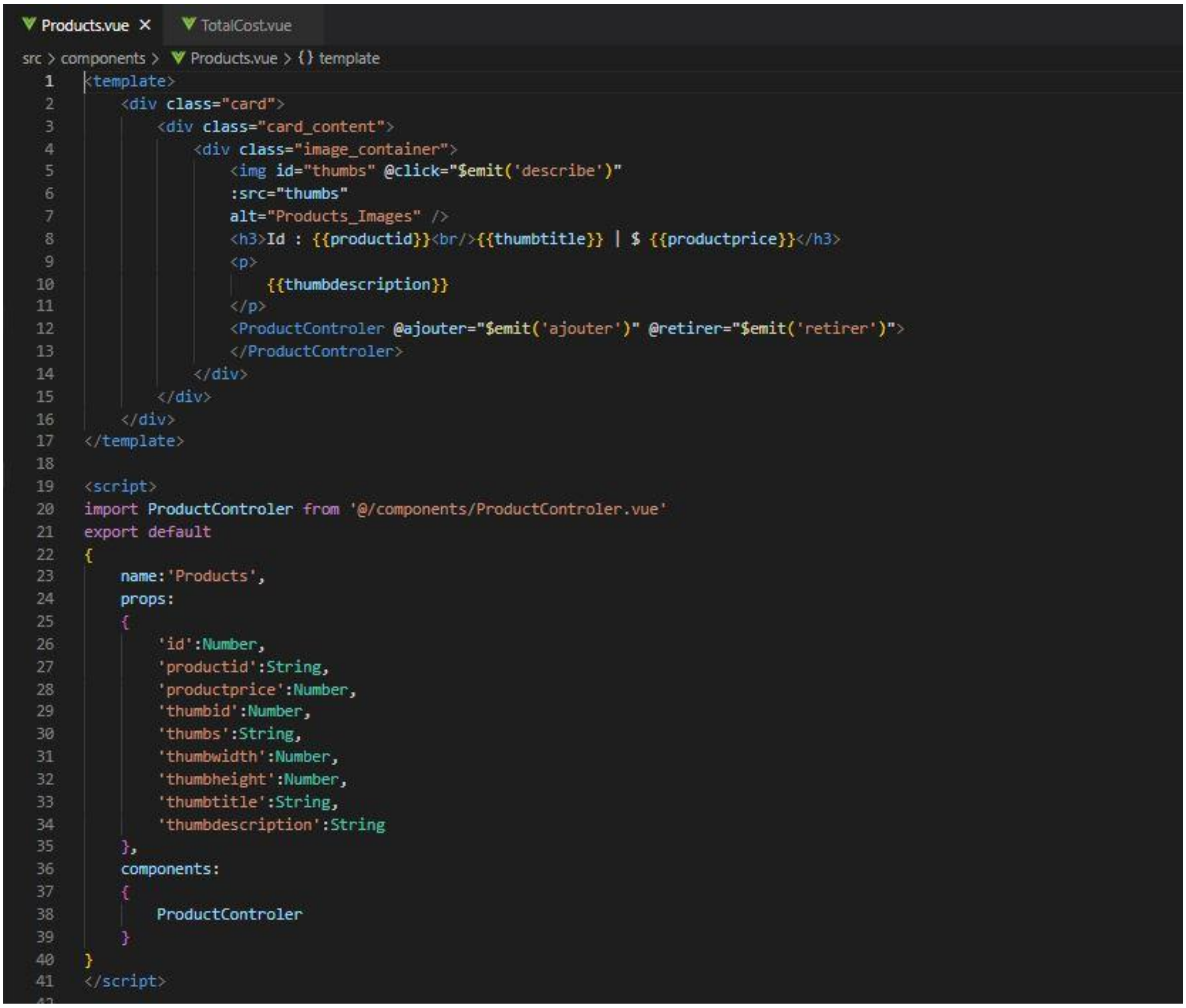


```
<template>
    <div class="bouton">
        <div class="bouton_container">
            <button id="addcartbutton" @click="$emit('ajouter')" @mouseup="$emit('incrementqtt')">Add to cart</button>
        </div>
    </div>
</template>
<script>
export default
{
    name:'ProductControler'
}
</script>

<style>
button#addcartbutton
{
    width:150px;
    height: 30px;
    font-family:arial;
    font-weight:bold;
    border:2px solid black;
}
button#addcartbutton:hover
{
    cursor:pointer;
}
</style>
```

This is the "ProductControler.vue" component.

@click="$emit('ajouter')"** and **@click="$emit('incrementqtt')" : these attributes are respectively emit "ajouter" event to add item in cart and "incrementqt" event to increment quantity in the parent that invoke this component.

- **Products.vue Component:**



```
<template>
    <div class="card">
        <div class="card_content">
            <div class="image_container">
                <img id="thumbs" @click="$emit('describe')"
                :src="thumbs"
                alt="Products_Images" />
                <h3>Id : {{productid}}<br/>{{thumbtitle}} | $ {{productprice}}</h3>
                <p>
                    {{thumbdescription}}
                </p>
                <ProductControler @ajouter="$emit('ajouter')" @retirer="$emit('retirer')">
                </ProductControler>
            </div>
        </div>
    </div>
</template>

<script>
import ProductControler from '@/components/ProductControler.vue'
export default
{
    name:'Products',
    props:
    {
        'id':Number,
        'productid':String,
        'productprice':Number,
        'thumbid':Number,
        'thumbs':String,
        'thumbwidth':Number,
        'thumbheight':Number,
        'thumbtitle':String,
        'thumbdescription':String
    },
    components:
    {
        ProductControler
    }
}
</script>
```

This is the “Products.vue” component.

@click=”$emit(‘describe’)” and “src=thumbs” : respectively emit “describe” event to be listened by the parent which is the “HomeView” and after, calls the “decrire” methods in the store. All the arguments of “decrire” function are fetched by the props of this component after lopping over the “tab_cart” array on the “HomeView” as his parent.

The “ProductControler” here is a child component of “Products.vue”. It have two attributes:

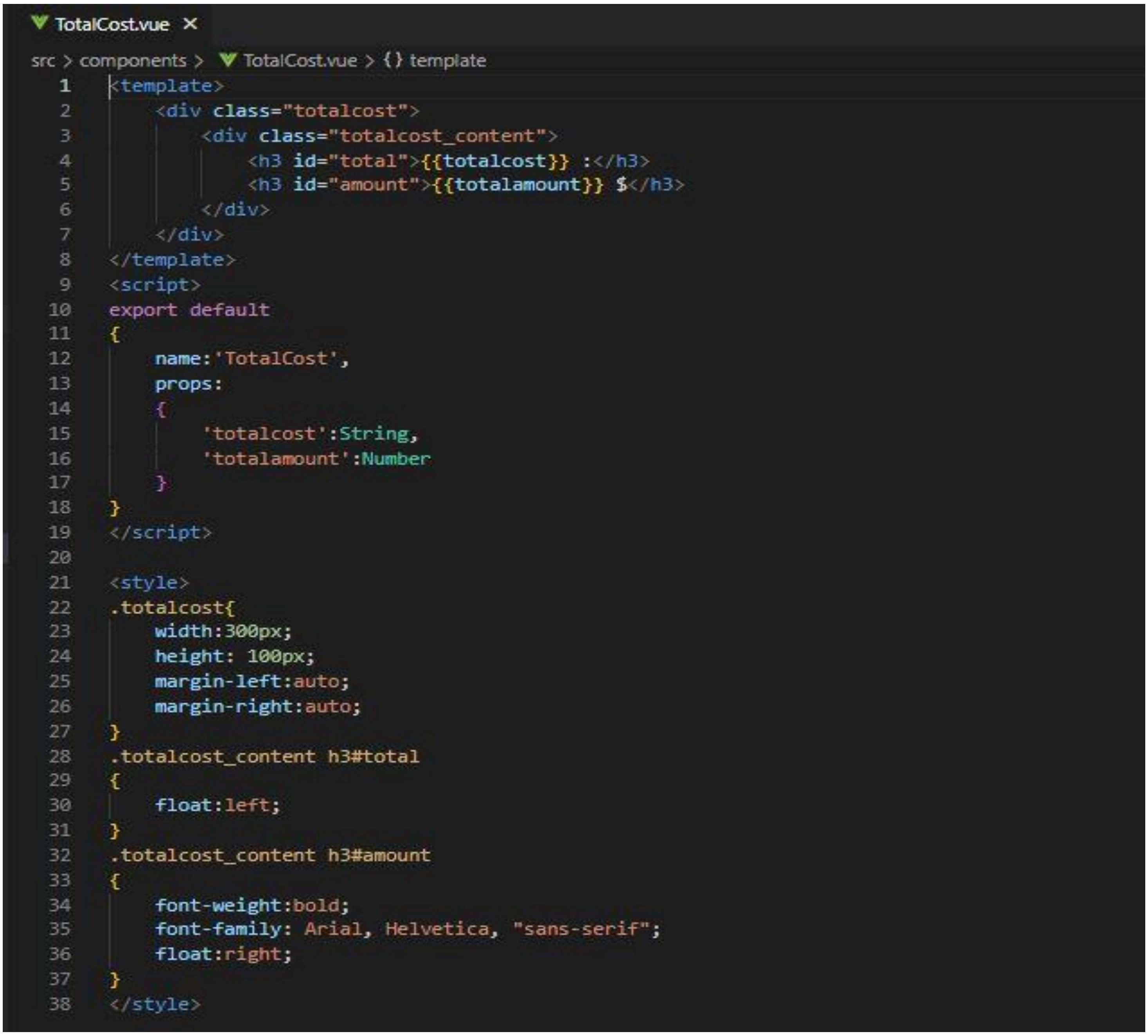
@ajouter=”$emit(‘ajouter’)”** and **@retirer=”$emit(‘retirer’)” : respectively re-emit “ajouter” event for cart adding and “retirer” event for cart removing item. All these event are received by the “HomeView” and this calls the “additem” method in the store.

Props:

- **id**: the product identification in the array like an index. Type Number (0, 1, 2, 3, 4 …)
- **productid**: String used as product identification (ex: 7g9j876x6hb,…)
- **productprice**: each product price type Number (could be a decimal number)

- **thumbid**: used as each product identification. Type Number (0, 1, 2, 3 …)
- **thumbtitle**: the name of each product type String.
- **thumbs**: used as image source for the img element.
- **thumbwidth**: the width of the thumbnail. Type Number (100, 200, 300, …)
- **thumbheight**: the height of the thumbnail. Type Number (100, 200, 300, …)
- **thumbtitle**: the name of each product. Type String
- **thumbdescription**: the product full description. Type String

- **TotalCost.vue Component:**



```
<template>
    <div class="totalcost">
        <div class="totalcost_content">
            <h3 id="total">{{totalcost}} :</h3>
            <h3 id="amount">{{totalamount}} $</h3>
        </div>
    </div>
</template>
<script>
export default
{
    name:'TotalCost',
    props:
    {
        'totalcost':String,
        'totalamount':Number
    }
}
</script>

<style>
.totalcost{
    width:300px;
    height: 100px;
    margin-left:auto;
    margin-right:auto;
}
.totalcost_content h3#total
{
    float:left;
}
.totalcost_content h3#amount
{
    font-weight:bold;
    font-family: Arial, Helvetica, "sans-serif";
    float:right;
}
</style>
```

This is the "TotalCost.vue" component. It's used to display the total sot of items in cart.

Props:

totalcost: label used for the h3 element. Type String. This is obtained by passing this type via the parent "HomeView"

totalamount: the total amount of items in cart. Type Number and could be a decimal number. This is obtained by passing the value via this props.