



El futuro digital  
es de todos

MinTIC

```
te( "name" );  
"type" );
```

```
( type == "sprite" )
```

```
std::string item_name = item->Attribute( "name" );  
std::string spritename = item->Attribute( "spritename" );  
float x = boost::lexical_cast<float>( item->Attribute( "x" ) );  
float y = boost::lexical_cast<float>( item->Attribute( "y" ) );  
float offset = boost::lexical_cast<float>( item->Attribute( "offset" ) );  
  
SpriteDescList::iterator sp = sprite_descs.begin();  
for( ; sp != sprite_descs.end(); ++sp )  
    if ( sp->name_ == spritename )  
        break;
```

## Ciclo 3:

Desarrollo de Software



**Misión  
TIC2022**

VERSIÓN 1.0

Unidad de educación  
continua y permanente  
Facultad de Ingeniería



Unidad Camilo Torres  
Calle 44 # 45-67  
Bloque 85 piso 1



(57) + 314 5000  
uec\_ibog@unaleduco

# Componente de Presentación

## (CORS e Introducción a Vue)

### Actividad Práctica

Para el desarrollo del componente de presentación es necesario configurar los CORS en el componente lógico. Una vez se realice la configuración, se creará el proyecto que contendrá al componente de presentación del caso de estudio.

### Configuración de CORS

Para la configuración de los CORS en el componente lógico, se debe abrir el proyecto de Django en Visual Studio Code, y su terminal con el entorno virtual de Python activado ([env\Scripts\activate](#) en Windows, o [source env/bin/activate](#) si se utiliza Linux o macOS). Posteriormente, se debe instalar el paquete [django-cors-headers](#) con el comando [pip install django-cors-headers](#):

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE

(env) D:\bank_be>pip install django-cors-headers
Collecting django-cors-headers
  Downloading django_cors_headers-3.9.0-py3-none-any.whl (12 kB)
Requirement already satisfied: Django>=2.2 in d:\bank_be\env\lib\site-packages (from django-cors-headers) (3.2.7)
Requirement already satisfied: asgiref<4,>=3.3.2 in d:\bank_be\env\lib\site-packages (from Django>=2.2->django-cors-headers) (3.4.1)
Requirement already satisfied: sqlparse>=0.2.2 in d:\bank_be\env\lib\site-packages (from Django>=2.2->django-cors-headers) (0.4.2)
Requirement already satisfied: pytz in d:\bank_be\env\lib\site-packages (from Django>=2.2->django-cors-headers) (2021.1)
Installing collected packages: django-cors-headers
Successfully installed django-cors-headers-3.9.0
```

Una vez se ha instalado, se debe añadir este paquete en el archivo [requirements.txt](#):

```
django==3.2.7
django-rest-framework==3.12.4
django-rest-framework-simplejwt==4.8.0
psycopg2==2.9.1
gunicorn==20.1.0
django-heroku==0.3.1
django-cors-headers==3.9.0
```

Ahora, se debe realizar la configuración del servidor del componente lógico para aceptar peticiones de los dominios que se requieren. Para esto, en el archivo [authProject/settings.py](#) se deben realizar una serie de cambios. El primero de ellos es añadir [corsheaders](#) a la lista de aplicaciones instaladas en el proyecto, de tal

forma que la variable `INSTALLED_APPS` se vea así:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'rest_framework',  
    'authApp',  
    'corsheaders',  
]
```

Luego, en el mismo archivo se debe añadir `corsheaders.middleware.CorsMiddleware` a la lista de middlewares del servidor. De forma que la variable `MIDDLEWARE` se vea así:

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
    "corsheaders.middleware.CorsMiddleware",  
]
```

Lo último que se debe hacer es configurar los orígenes desde los cuales el servidor aceptará peticiones. Esto se puede realizar de dos formas, la primera es declarando la variable `CORS_ALLOWED_ORIGINS` que contendrá una lista de cadenas con los nombres de los orígenes que aceptará. Un ejemplo podría ser:

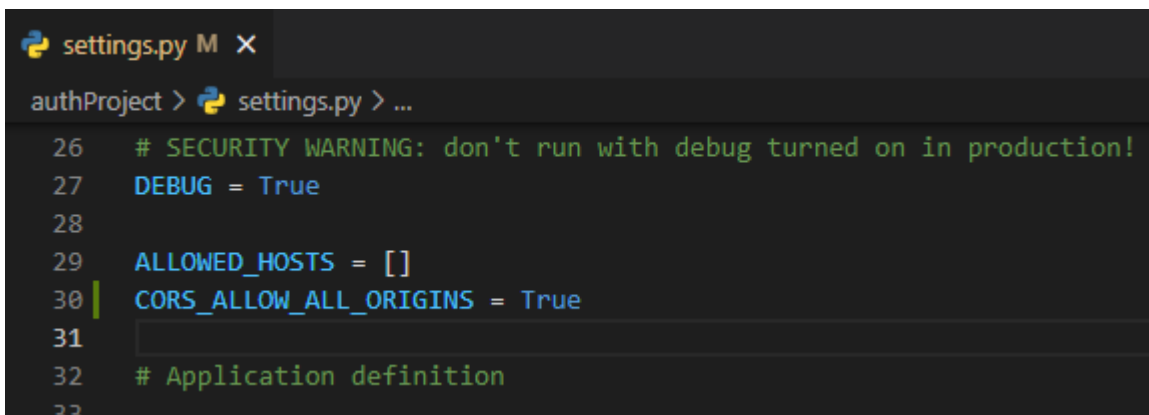
```
CORS_ALLOWED_ORIGINS = [  
    "https://example.com",  
    "https://sub.example.com",  
    "http://localhost:8080",  
]
```

```
"http://127.0.0.1:9000",  
]
```

La segunda forma es indicarle a Django que acepte las todas peticiones sin importar su origen. Para ello, se declara la variable `CORS_ALLOW_ALL_ORIGINS` de la siguiente manera:

```
CORS_ALLOW_ALL_ORIGINS = True
```

Para evitar inconvenientes futuros se utilizará esta última opción. Esta variable se puede ubicar en cualquier parte del archivo, sin embargo, para mantener el orden de las configuraciones, se ubicará bajo la variable `ALLOWED_HOSTS`:



```
settings.py M X  
authProject > settings.py > ...  
26 # SECURITY WARNING: don't run with debug turned on in production!  
27 DEBUG = True  
28  
29 ALLOWED_HOSTS = []  
30 CORS_ALLOW_ALL_ORIGINS = True  
31  
32 # Application definition  
33
```

Una vez se han realizado las respectivas configuraciones, se debe hacer de nuevo el despliegue. Para ello se utilizan los mismos comandos que se utilizaron para hacer el despliegue anteriormente:

```
heroku login  
git add .  
git commit -am "deployment changes"  
git push heroku master
```

**Nota:** de ser necesario, en el material de la clase se encuentra un archivo `C3.AP.14. bank_be.rar`, con todos los avances en el desarrollo del componente realizado en esta guía.

## Uso de Vue.js

### Creación del proyecto

Existen múltiples formas de crear un proyecto de [Vue.js](#). Para facilitar este proceso, a continuación, se utilizará [Vue CLI](#). Para ello se debe abrir una consola en la carpeta en la cual se creará el proyecto de Vue, y allí se debe utilizar el comando `vue create appName`, donde `appName` se debe reemplazar por el nombre que se le desea dar al proyecto. Dependiendo de los recursos del computador, este comando se puede demorar un poco en ejecutar:

```
D:\>vue create bank_fe

Vue CLI v4.5.13
? Please pick a preset: (Use arrow keys)
> Default ([Vue 2] babel, eslint)
  Default (Vue 3) ([Vue 3] babel, eslint)
  Manually select features
```

Allí se muestra una interfaz por la cual se puede navegar con las siguientes teclas:

- Flechas de arriba y abajo: para moverse entre las opciones que se presentan. La opción seleccionada es aquella que tiene a la izquierda un mayor que (>) y se ve de color azul.
- Enter: para confirmar la selección realizada.
- Space: para seleccionar y deseleccionar un opción. Esta tecla solo es útil cuando se pueden seleccionar una o más opciones.

En la interfaz que se muestra, se debe seleccionar la opción [Manually select features](#), pues se configurarán las características que tendrá el proyecto. Luego se confirma la selección con [Enter](#).

```
Vue CLI v4.5.13
? Please pick a preset:
  Default ([Vue 2] babel, eslint)
  Default (Vue 3) ([Vue 3] babel, eslint)
> Manually select features
```

En las siguientes opciones que se muestran, se deben seleccionar con [Space](#) únicamente tres: [Choose Vue versión](#), [Babel](#), [Router](#). Luego se confirma la selección con [Enter](#):

```
Vue CLI v4.5.13
? Please pick a preset: Manually select features
? Check the features needed for your project:
  (*) Choose Vue version
  (*) Babel
  ( ) TypeScript
  ( ) Progressive Web App (PWA) Support
> (*) Router
  ( ) Vuex
  ( ) CSS Pre-processors
  ( ) Linter / Formatter
  ( ) Unit Testing
  ( ) E2E Testing
```

Las últimas cuatro opciones que se deben seleccionar/ingresar en la interfaz son:

- La opción **3.x**:

```
Vue CLI v4.5.13
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router
? Choose a version of Vue.js that you want to start the project with
  2.x
> 3.x
```

- La opción **n**:

```
Vue CLI v4.5.13
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) n
```

- La opción **In package.json**:

```
Vue CLI v4.5.13
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Where do you prefer placing config for Babel, ESLint, etc.?
  In dedicated config files
> In package.json
```

- La opción **n**:

```
Vue CLI v4.5.13
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Where do you prefer placing config for Babel, ESLint, etc.? In package.json
? Save this as a preset for future projects? (y/N) n
```

Una vez se ingresen todas estas opciones, se crea el proyecto:

```
Vue CLI v4.5.13
✨ Creating project in D:\bank_fe.
🚚 Initializing git repository...
⚙ Installing CLI plugins. This might take a while...

[██████████] ..... - extract:postcss-calc: sill extract upper-case@1.1.3
```

Cuando Vue CLI termine, se puede comprobar que el proyecto se creó correctamente abriendo una ventana de comandos en su carpeta y ejecutando el servidor con el comando `npm run serve`. Dependiendo de los recursos del computador, este comando se puede demorar un poco en ejecutar:

```
D:\bank_fe>npm run serve

> bank_fe@0.1.0 serve D:\bank_fe
> vue-cli-service serve
```

Si no hay ningún problema con la ejecución del proyecto, en la consola se muestra lo siguiente:

```
DONE Compiled successfully in 1787ms
```

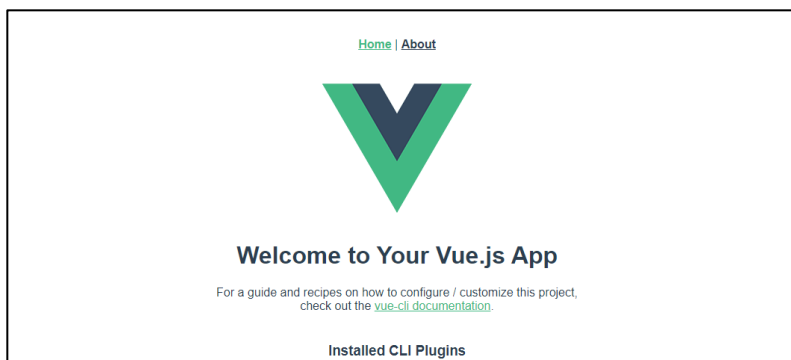
App running at:

```
- Local: http://localhost:8080/
- Network: http://192.168.1.59:8080/
```

Note that the development build is not optimized.  
To create a production build, run `npm run build`.

Si se abre la url <http://localhost:8080/> se debe ver lo siguiente:

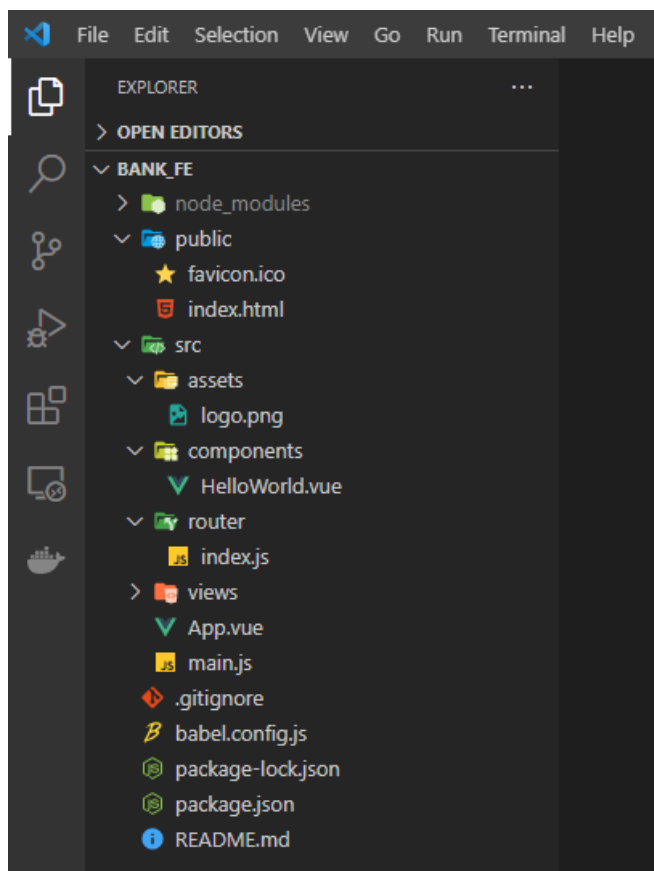




Para detener la ejecución del servidor del componente de presentación, se debe utilizar la combinación de teclas: **CTRL+C**.

### Configuración inicial del proyecto

Inicialmente, la estructura del proyecto de Vue se debe ver así:





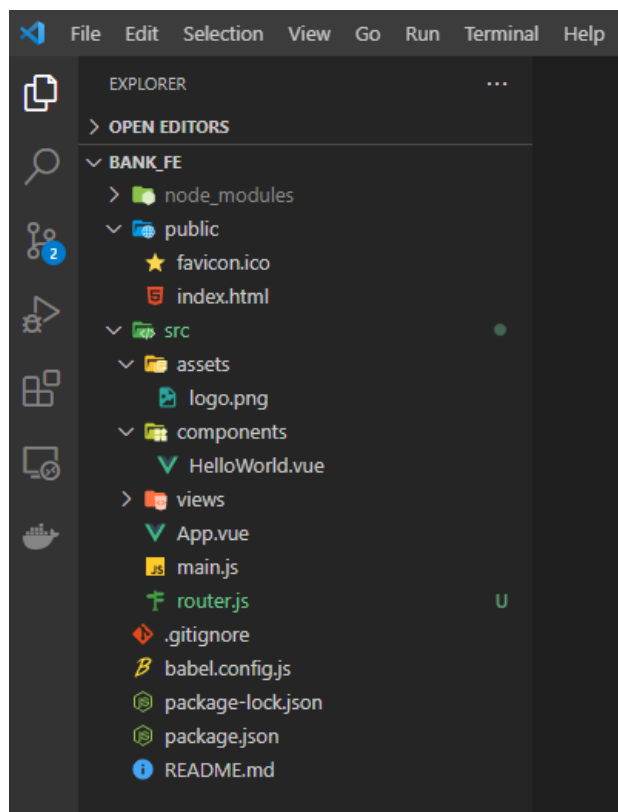
Esta estructura está conformada por las carpetas:

- `node_modules`: donde se almacenan las librerías externas del proyecto. Esta carpeta no debe ser modificada por el desarrollador.
- `public`: donde se almacenan los archivos base de la aplicación web.
- `src`: donde se almacena el código fuente de la aplicación web. Este contiene a su vez múltiples carpetas para organizar el código.

Durante el desarrollo del componente de presentación únicamente se modificarán los archivos que se encuentran dentro de la carpeta `src`. Por esta razón, para organizar su estructura de acuerdo con las necesidades del proyecto, se realizarán algunas modificaciones a su estructura.

## Router

Ya que solo se utilizará un archivo JavaScript para indicar las rutas de la página web, la primera modificación que se hará es mover el archivo `src/router/index.js` a la carpeta `src`, es decir que se va a sacar el archivo `index.js` de la carpeta `router`. Una vez se ha movido el archivo, se debe eliminar la carpeta `router` y se debe cambiar el nombre del archivo `src/index.js` por un nombre más diciente, como `router.js`. Una vez se han realizado estos cambios, la estructura del proyecto es la siguiente:



Dentro del archivo [src/router.js](#) se debe reemplazar todo el código por lo siguiente:

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

const routes = [{
  path: '/',
  name: 'root',
  component: App
}]

const router = createRouter({
  history: createWebHistory(),
  routes
})

export default router
```

## Main.js

El archivo [src/main.js](#) es aquel en el que se realizan las configuraciones del servidor, incluyendo el uso de un router, y la asignación de un componente inicial a la página web. Por esta razón, para evitar inconvenientes futuros, se debe asegurar que la importación del componente inicial y del router se hace correctamente. Luego de hacer los cambios al archivo [src/router.js](#), el código del archivo [src/main.js](#) debe ser el siguiente:

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'

createApp(App).use(router).mount('#app')
```

## App.vue

Por ahora, el único componente que se utilizará en la página web, es el componente principal [App.vue](#). En este se declara el código HTML, JavaScript y CSS de la página inicial de la página web. Por esto, el código que tiene este archivo se debe reemplazar por lo siguiente:

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

    <div class="header">
      <h1>Banco Misión TIC</h1>
    </div>

  </div>
</template>

<script>
export default {
  name: 'App',

  data: function(){
  },

  methods:{
  },

  created: function(){
  }
}
</script>

<style>
  body{
    margin: 0 0 0 0;
  }

  .header{
    margin: 0;
    padding: 0;
    width: 100%;
    height: 10vh;
    min-height: 100px;

    background-color: #283747 ;
    color:#E5E7E9 ;
  }
</style>
```

```
display: flex;
justify-content: space-between;
align-items: center;
}

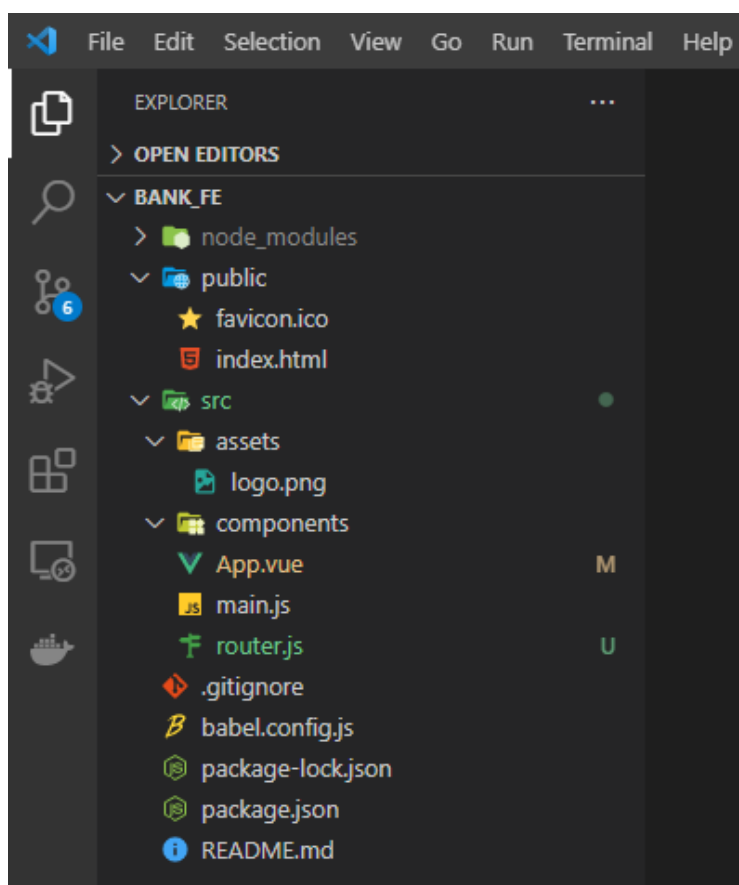
.header h1{
width: 20%;
text-align: center;
}
</style>
```

En el código que se encuentra en la etiqueta `script`, se exporta el objeto que define por medio de llaves/valores, los atributos y métodos del componente. La sintaxis para la definición de cada llave/valor, y su propósito varía de acuerdo con la llave. En el código utilizado, las llaves asignadas son las siguientes:

- **name**: a esta llave se asigna una cadena que indica el nombre del componente. Este nombre comúnmente es utilizado por otros componentes del proyecto para referenciar al componente.
- **data**: a esta llave se asigna una función que retorna un objeto, cuyas llaves son el nombre de la variable, y los valores son el valor que se da a cada una de ellas. Estas variables pueden ser utilizadas en cualquier parte del componente, ya sea en sus funciones, o en su código HTML.
- **methods**: a esta llave se asigna un objeto. Las llaves de este objeto representan el nombre de las funciones, y sus valores contienen la definición de cada función del componente.
- **created**: a esta llave se asigna la función que debe **ejecutar** el componente **cada vez** que se renderice. Esta no debe retornar nada, y se utiliza para realizar alguna operación necesaria tras renderizar el componente en la página web.

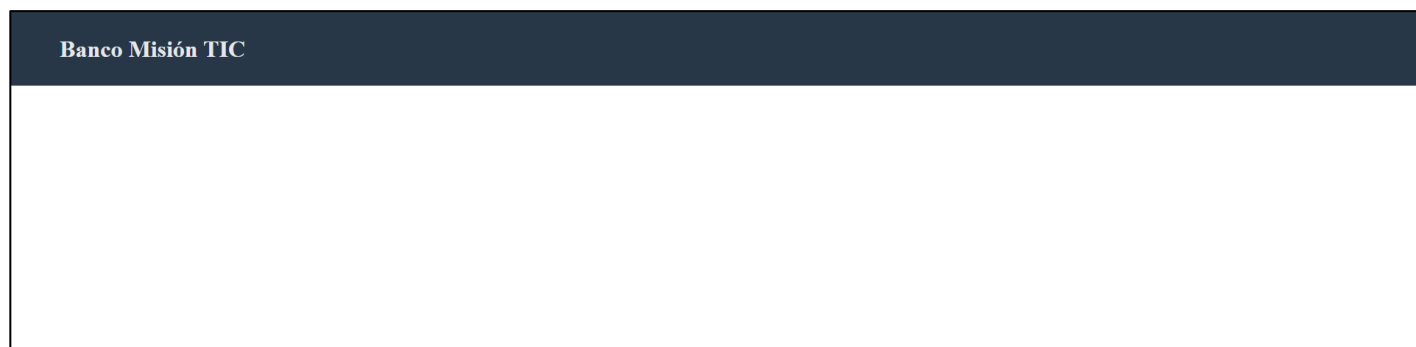
## Views y Components

El concepto de vistas no se utilizará en el desarrollo del componente de presentación, por ello se debe eliminar la carpeta `src/views`. Por otro lado, aún no se utilizará ningún componente diferente al principal, por lo que se deben eliminar todos los archivos que contiene la carpeta `src/components` (sin borrar la carpeta). Al realizar estos cambios la estructura del proyecto es la siguiente:



## App.vue

Una vez se han realizado todos los cambios, si se ejecuta el servidor y se abre su página principal se debe ver lo siguiente:



**Nota:** de ser necesario, en el material de la clase se encuentra un archivo *C3.AP.14. bank\_fe.rar*, con todos los avances en el desarrollo del componente realizado en esta guía.