

## EXERCISES QUE TRABAJAREMOS EN LA CUE

- EXERCISE 1: COMENZANDO CON QUASAR CLI.
- EXERCISE 2: DESARROLLO CON QUASAR CLI PARTE 1.
- EXERCISE 3: DESARROLLO CON QUASAR CLI PARTE 2.
- EXERCISE 4: DESARROLLO CON QUASAR CLI PARTE 3.
- EXERCISE 5: DESARROLLO CON QUASAR CLI PARTE 4.

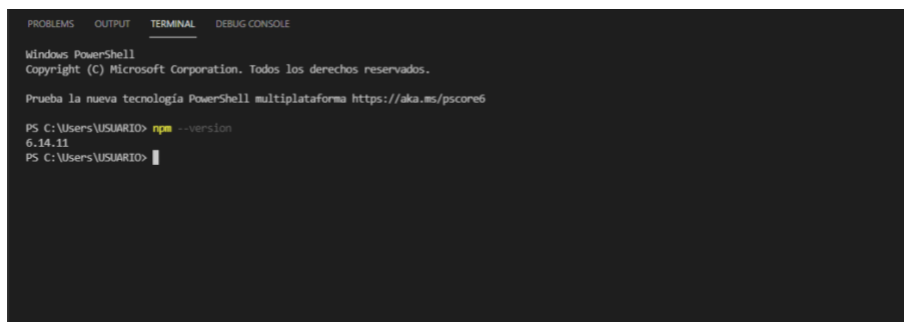
## EXERCISE 1: COMENZANDO CON QUASAR CLI

### INTRODUCCIÓN

En el siguiente ejercicio revisaremos cómo instalar la **CLI** de **Quasar**, e iniciar un proyecto usándola. Como vimos en la lectura, necesitamos tener **NPM** y **Node JS** instalados, y con versiones estables para que sea compatible con **Quasar**, así que primero, comprobaremos cuales tenemos instaladas.

### COMPROBANDO Y ACTUALIZANDO VERSIONES

Abrimos nuestro editor de texto y una nueva terminal, escribimos **"npm --version"**, y presionamos "enter".



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO> npm --version
6.14.11
PS C:\Users\USUARIO> 
```

Vemos que la versión instalada localmente corresponde a 6.14.11, que no es compatible con lo requerido por la documentación de **Quasar**, así que haremos una actualización.

Para eso, vamos a escribir el comando `"npm install npm@latest -g"`, y presionamos "enter".

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS54\Exercises\cli-unit-test> npm install npm@latest -g
```

Empieza la instalación, y vemos que se actualiza. Confirmamos escribiendo `"npm --version"`, y presionamos "enter".

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS54\Exercises\cli-unit-test> npm install npm@latest -g
C:\Users\USUARIO\AppData\Roaming\npm\npx -> C:\Users\USUARIO\AppData\Roaming\npm\node_modules\npm\bin\npx-cli.js
C:\Users\USUARIO\AppData\Roaming\npm\npx -> C:\Users\USUARIO\AppData\Roaming\npm\node_modules\npm\bin\npx-cli.js
+ npm@7.18.1
added 255 packages from 146 contributors in 7.566s
PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS54\Exercises\cli-unit-test> npm --version
```

Nos damos cuenta de que cambió la versión, ahora tenemos la 7.18.1. Con esto actualizado, podemos continuar revisando **Node.js**. Ahora, escribimos en la terminal el comando: `"node --version"`, y presionamos "enter".

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS518\Exercises\ejerc-18-quasar> node --version
v14.16.0
PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS518\Exercises\ejerc-18-quasar>
```

## INSTALANDO CLI DE QUASAR

Podemos ver que la versión que tenemos localmente es la 14.16.0, la cual es estable y superior a lo solicitado por **Quasar**, por lo tanto, debería funcionar; así que continuaremos con la instalación, escribiendo **"npm install -g @quasar/cli"** y presionamos "enter".

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/powershell

PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test> npm install npm@latest -g
C:\Users\USUARIO\AppData\Roaming\npm\npx -> C:\Users\USUARIO\AppData\Roaming\npm\node_modules\npm\bin\npx-cli.js
C:\Users\USUARIO\AppData\Roaming\npm\npx -> C:\Users\USUARIO\AppData\Roaming\npm\node_modules\npm\bin\npx-cli.js
+ npm@7.18.1
added 255 packages from 146 contributors in 7.566s
PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test> npm --version
7.18.1
PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test> node --version
v14.16.0
PS C:\Users\USUARIO\Desktop\Edutecno\VS-frontend-vue\VS-Front-end-Vue\VS54\Exercises\ci-unit-test> npm install -g @quasar/cli
[ ] ..... - idealTree:is-object: 511 fetch manifest has-symbol-support-x@1.4.1
  
```

Aquí empieza la instalación. Una vez que termina, confirmamos escribiendo **"quasar"**, y presionamos "enter".

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\USUARIO> quasar

  _____
 /  _  _  _  \
|  _||_||_||_|
|  _||_||_||_|
 \  _  _  _  /
  \_||_||_||_

Running @quasar/cli v1.2.1

Example usage
$ quasar <command> <options>

Help for a command
$ quasar <command> --help
$ quasar <command> -h

Options
--version, -v Print Quasar CLI version

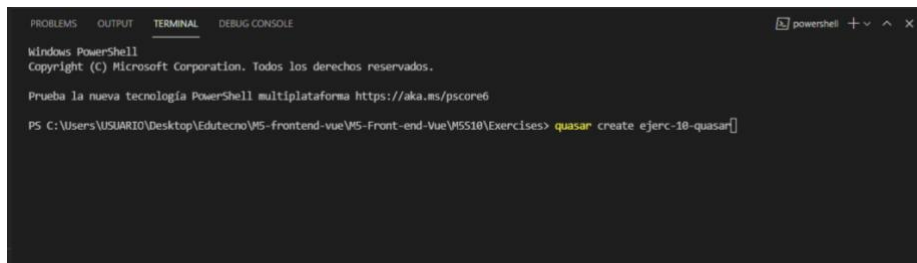
Commands
create  Create a project folder
info    Display info about your machine
        (and your App if in a project folder)
upgrade Check (and optionally) upgrade Quasar packages
        from a Quasar project folder
serve   Create an ad-hoc server on App's distributables
help, -h Displays this message

-----
  
```

Y vemos que aparece nuestra versión de **Quasar CLI**, que es la 1.2.1.

## CREANDO UN PROYECTO CON LA CLI DE QUASAR

Con esto, ya estamos listos para crear un proyecto con nuestro editor de texto. Nos ubicamos en la carpeta donde queremos desarrollarlo, y la abrimos. Aquí escribiremos el comando: **“quasar create (nombre proyecto)”**. En este caso, le pondremos “ejerc-10-quasar”, y presionamos “enter”.



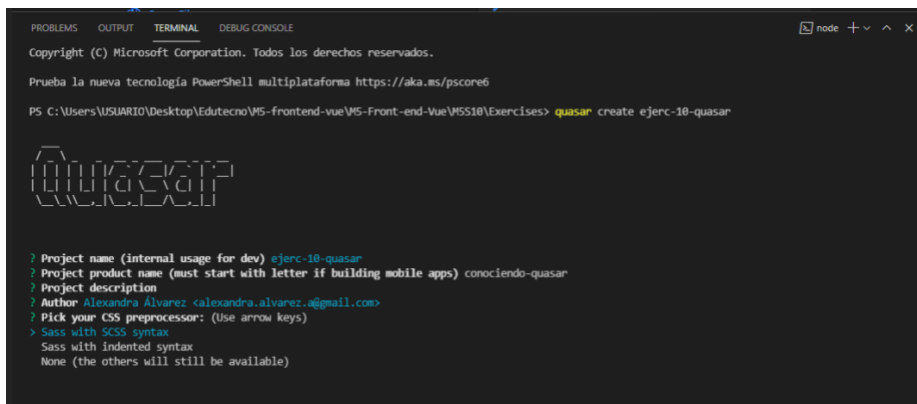
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\M5S10\Exercises> quasar create ejerc-10-quasar
  
```

Empezará a hacernos algunas preguntas de configuración. La primera es, ¿cuál es el nombre del proyecto?: por defecto, nos sugiere el que le dimos a la carpeta, y presionaremos “enter”. La siguiente pregunta tiene que ver con el nombre del producto del proyecto, le pondremos **“quasar app”**, que es la que viene por defecto, y “enter”. Luego, nos pide una descripción de éste, podemos dejarlo vacío y nuevamente, “enter”. A continuación, nos pregunta por el nombre del autor y nos da una sugerencia, en este caso la tomaremos, y “enter”.



```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\M5S10\Exercises> quasar create ejerc-10-quasar

  [Quasar CLI]

? Project name (internal usage for dev) ejerc-10-quasar
? Project product name (must start with letter if building mobile apps) conociendo-quasar
? Project description
? Author Alexandra Álvarez <alexandra.alvarez.a@gmail.com>
? Pick your CSS preprocessor: (Use arrow keys)
  > Sass with SCSS syntax
    Sass with indented syntax
    None (the others will still be available)
  
```

Ahora, nos pregunta por la elección de un preprocesador de **Sass**, podemos tomar la sugerencia que es: **“Sass with SCSS syntax”**, y tecla “enter”. Continúa con las características que necesitará nuestro proyecto, quitaremos **ESlint** presionando la tecla “espaciado”, dejaremos **Vuex** y **Axios**, e ignoramos **Vue-i18n**; este último es un plugin para crear aplicaciones internacionales, que se pueden adaptar a diversos idiomas utilizando **Vue**. Con esto seleccionado, oprimimos “enter”.

The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
```

Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/powershell>

PS C:\Users\USUARIO\Desktop\EduTecno\VS-frontent-vue\VS-Front-end-Vue\VS510\Exercises> quasar create ejerc-10-quasar

A stylized ASCII art logo consisting of vertical bars of varying heights arranged in a grid-like pattern.

```
> Project name (internal usage for dev) ejerc-10-quasar
> Project product name (must start with letter if building mobile apps) conociendo-quasar
> Project description
> Author Alexandra Álvarez <alexandra.alvarez.a@gmail.com>
> Pick your CSS preprocessor: SCSS
> Check the features needed for your project:
( ) ESLint (recommended)
( ) TypeScript
(*) Vuex
>(*) Axios
( ) Vue-i18n
```

De inmediato, nos pregunta si queremos instalar dependencias después de que el proyecto haya sido creado. Seleccionamos la alternativa **“Yes, use NPM”**, y presionamos “enter”.

[illegible]

Veremos cómo se comienza a crear el proyecto.

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Copyright (c) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO\Desktop\Edutecno\MS-frontend-vue\MS-Front-end-Vue\MS10\Exercises> quasar create ejerc-10-quasar

[Quasar CLI]
[?] Project name (internal usage for dev) ejerc-10-quasar
[?] Project product name (must start with letter if building mobile apps) conociendo-quasar
[?] Project description
[?] Author Alexandra Álvarez <alexandra.alvarez.a@gmail.com>
[?] Pick your CSS preprocessor: SCSS
[?] Check the features needed for your project: Vuex, Axios
[?] Continue to install project dependencies after the project has been created? (recommended) NPM

Quasar CLI - Generated "ejerc-10-quasar".

[*] Installing project dependencies ...
[.....] - IdealTree:zlib: 5.11 fetch manifest wilddcard@2.0.0

```

Una vez que termina, nos muestra dos comandos para comenzar; el primero es `cd proyecto-quasar`, que sirve para poder entrar a la carpeta del proyecto; y el segundo es `quasar dev`, que es para levantar el servidor. Entonces, utilizaremos la primera opción, escribimos su comando, y presionamos “enter”. Ahora que ya estamos ahí, utilizaremos el segundo, y nuevamente “enter”.

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS C:\Users\USUARIO\Desktop\Edutecno\MS-frontend-vue\MS-Front-end-Vue\MS10\Exercises\ejerc-10-quasar> quasar dev

Dev mode..... spa
Pkg quasar..... v2.0.1
Pkg @quasar/app... v3.0.1
Pkg webpack..... v5
Debugging..... enabled

Configured browser support (>= 87.66% of global marketshare):
- Chrome for Android >= 91
- Firefox for Android >= 89
- Android >= 91
- Chrome >= 81
- Edge >= 88
- Firefox >= 80
- iOS >= 11.0-11.2
- Opera >= 73
- Safari >= 11.1

```

Vemos que muestra un error, esto es porque no permite levantar el proyecto en el **puerto 8080**.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

- Edge >= 88
- Firefox >= 80
- iOS >= 11.0-11.2
- Opera >= 73
- Safari >= 11.1

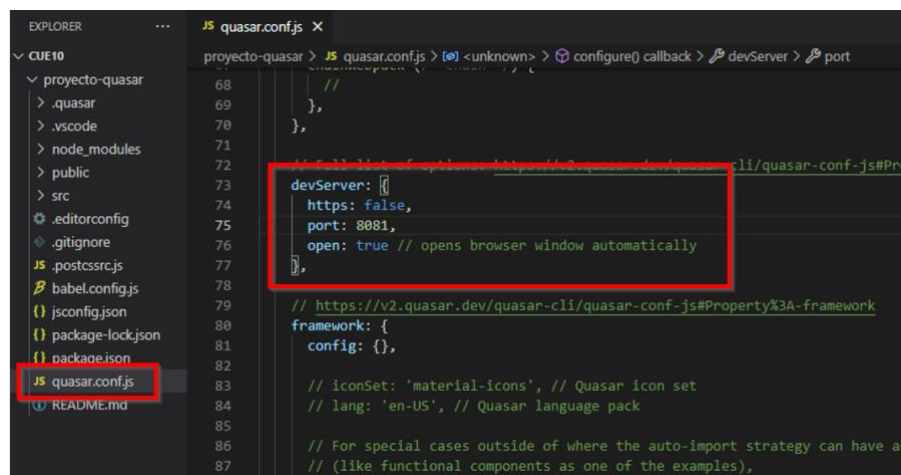
App • ⚠ Unknown network error occurred
[ Error: permission denied 0.0.0.0:8080

- task_queues.js:81 processTicksAndRejections
  internal/process/task_queues.js:81:21

] {
  code: 'EACCES',
  errno: -4092,
  syscall: 'listen',
  address: '0.0.0.0',
  port: 8080
}

```

Entonces, lo que haremos será ir al archivo `“quasar.config.js”`, buscamos `“devServer”`, que es donde está el puerto, y lo cambiaremos por el 8081. Guardamos los cambios, volvemos a escribir el comando `“quasar dev”`, y presionamos “enter”.



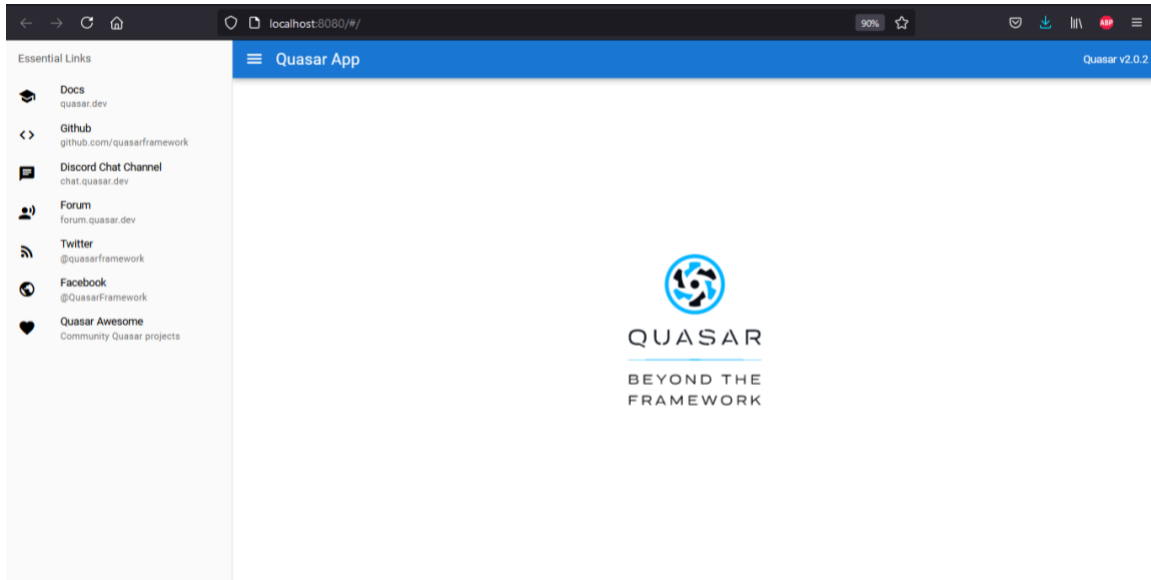
```

EXPLORER    ...    JS quasar.config.js X
CUE10
  proyecto-quasar
    .quasar
    .vscode
    node_modules
    public
    src
    .editorconfig
    .gitignore
    .postcssrc.js
    babel.config.js
    jsconfig.json
    package-lock.json
    package.json
    JS quasar.config.js
    README.md

68 //
69 },
70 },
71 },
72 //
73 //
74 //
75 devServer: {
76   https: false,
77   port: 8081,
78   open: true // opens browser window automatically
79 },
80 // https://v2.quasar.dev/quasar-cli/quasar-conf-js#Property%3A-framework
81 framework: {
82   config: {},
83   // iconSet: 'material-icons', // Quasar icon set
84   // lang: 'en-US', // Quasar language pack
85   // For special cases outside of where the auto-import strategy can have an
86   // (like functional components as one of the examples),
87

```

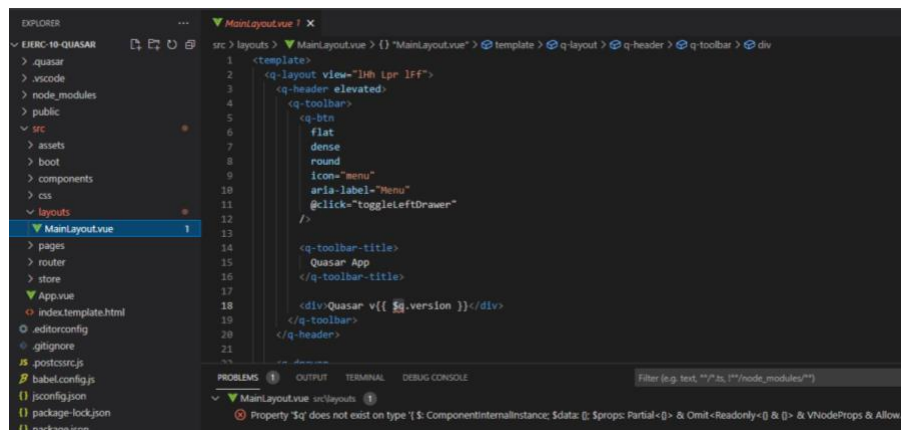
Automáticamente nos abre la ventana del proyecto. Podemos ver que todo está en orden para comenzar a desarrollar.



## EXERCISE 2: DESARROLLO CON QUASAR CLI PARTE 1

### INTRODUCCIÓN

Ya que hemos creado el proyecto exitosamente, es hora de empezar a desarrollarlo, por lo que aprenderemos a generar un **layout**. Este componente actúa como una plantilla base de las páginas del proyecto. Permite compartir elementos, tales como: una barra de navegación, menú, footer, entre otros, facilitando la reutilización del código. Su uso no es obligatorio, pero ayuda a simplificar el diseño de los sitios web o aplicaciones creados con **Quasar**. Revisaremos la carpeta **src>layout** de nuestro proyecto, y encontraremos un archivo llamado **MainLayout.vue**.



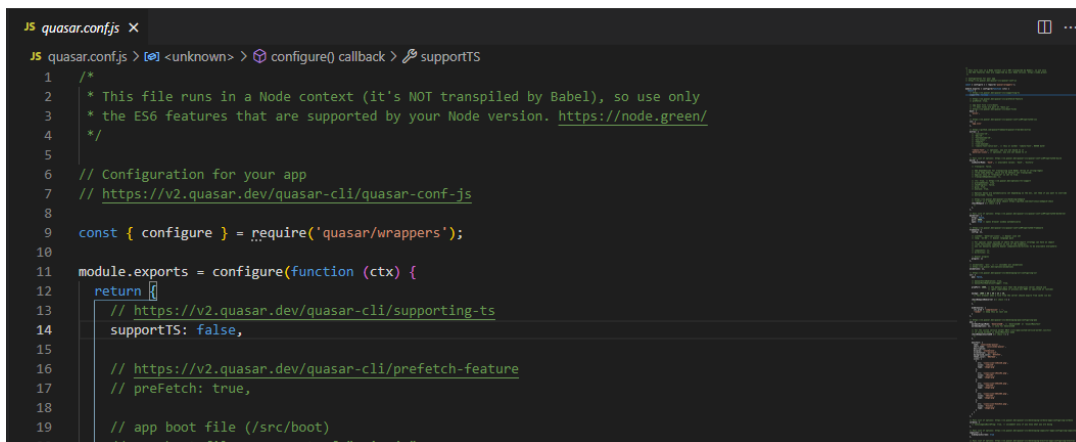


Esto es el **layout** base que **Quasar** nos brinda por defecto, trae un **header** y un menú lateral con enlaces de utilidad.

## AGREGANDO SOPORTE A TYPESCRIPT

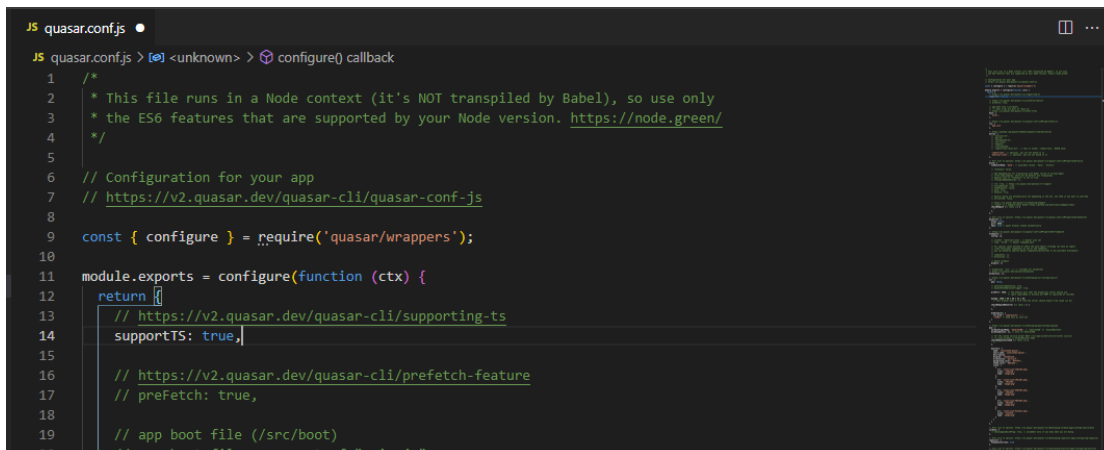
En algunos casos, se puede ver en la pestaña de problemas de la terminal, un error que se soluciona con los siguientes pasos:

Iremos al archivo **“/quasar.conf.js”**.



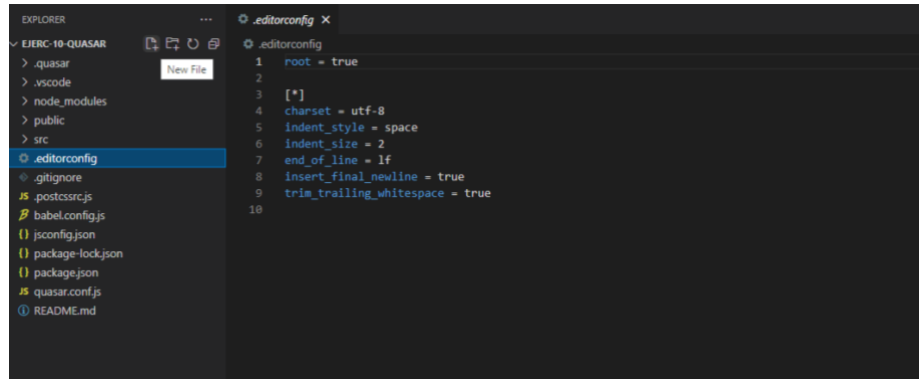
```
JS quasar.conf.js X
JS quasar.conf.js > [?] <unknown> > [?] configure() callback > [?] supportTS
1  /*
2   * This file runs in a Node context (it's NOT transpiled by Babel), so use only
3   * the ES6 features that are supported by your Node version. https://node.green/
4   */
5
6   // Configuration for your app
7   // https://v2.quasar.dev/quasar-cli/quasar-conf.js
8
9   const { configure } = require('quasar/wrappers');
10
11   module.exports = configure(function (ctx) {
12     return {
13       // https://v2.quasar.dev/quasar-cli/supporting-ts
14       supportTS: false,
15
16       // https://v2.quasar.dev/quasar-cli/prefetch-feature
17       // preFetch: true,
18
19       // app boot file (/src/boot)
20       // ... boot files are part of "main.js"
```

El error consiste en que **TypeScript** no tiene soporte. Notaremos que en el código dice: **“supportTS: false”**, y lo vamos a cambiar a **“true”**.

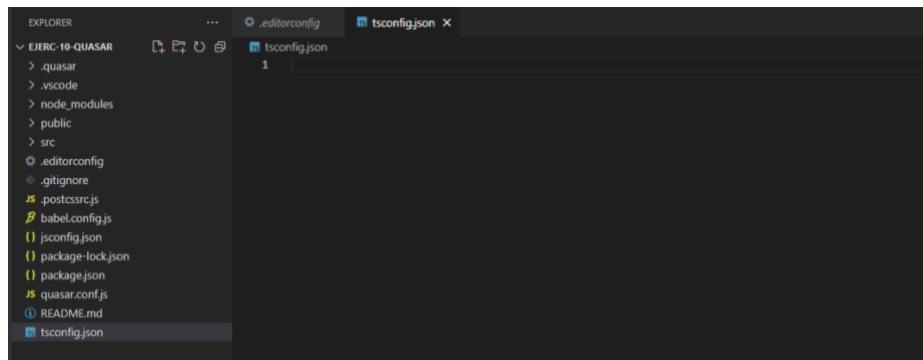


```
JS quasar.conf.js ●
JS quasar.conf.js > [?] <unknown> > [?] configure() callback
1  /*
2   * This file runs in a Node context (it's NOT transpiled by Babel), so use only
3   * the ES6 features that are supported by your Node version. https://node.green/
4   */
5
6   // Configuration for your app
7   // https://v2.quasar.dev/quasar-cli/quasar-conf.js
8
9   const { configure } = require('quasar/wrappers');
10
11   module.exports = configure(function (ctx) {
12     return {
13       // https://v2.quasar.dev/quasar-cli/supporting-ts
14       supportTS: true,
15
16       // https://v2.quasar.dev/quasar-cli/prefetch-feature
17       // preFetch: true,
18
19       // app boot file (/src/boot)
20       // ... boot files are part of "main.js"
```

Guardamos, y de inmediato vamos a ir a la raíz del proyecto a crear un nuevo archivo.



Éste es llamado: **"tsconfig.json"**.



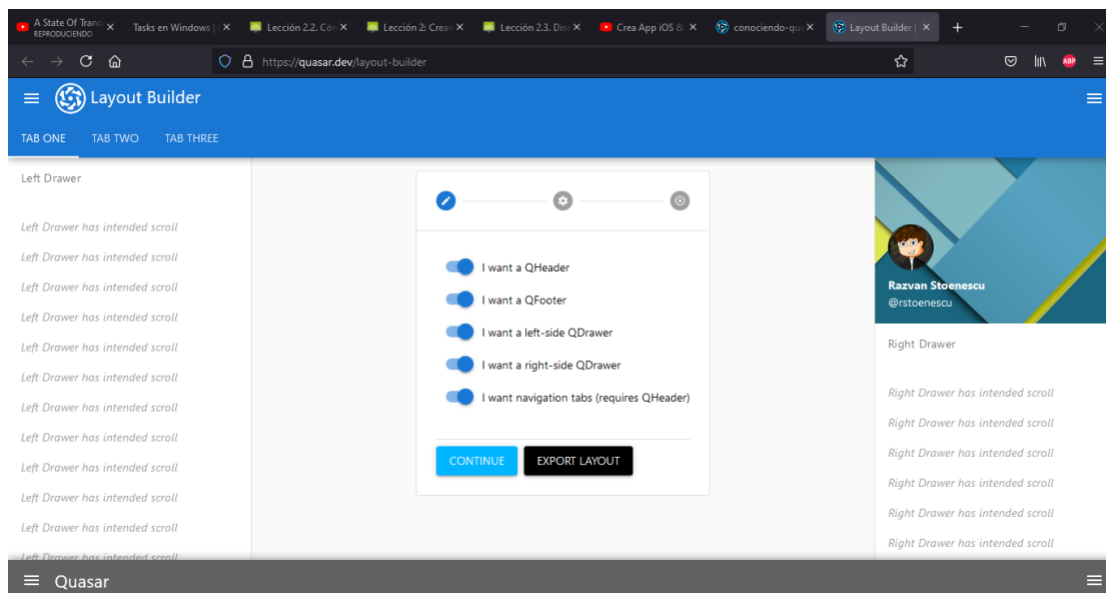
Ahora, le copiamos y pegamos el siguiente código. Guardamos.

```
1 {
2   "extends": "@quasar/app/tsconfig-preset",
3   "compilerOptions": {
4     "baseUrl": "."
5   }
6 }
```

Para que los cambios sean reconocidos, cerramos el editor de texto, y lo abrimos de nuevo. Volvemos al archivo **“MainLayout.vue”**. Allí veremos que el error ha desaparecido, y podemos continuar.

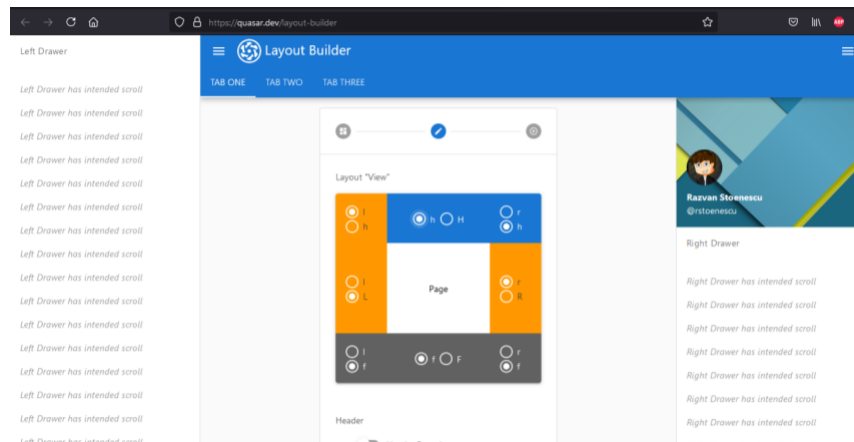
## LAYOUT BUILDER

Si vamos a la dirección: <https://quasar.dev/layout-builder>, encontramos una utilidad que permite crear **layouts** a la medida. Ahí podemos ver una página web, con una serie de componentes como: un **header**, una barra lateral izquierda, y una derecha, un **footer**, y el contenido del centro de la página, como se puede ver en la imagen.

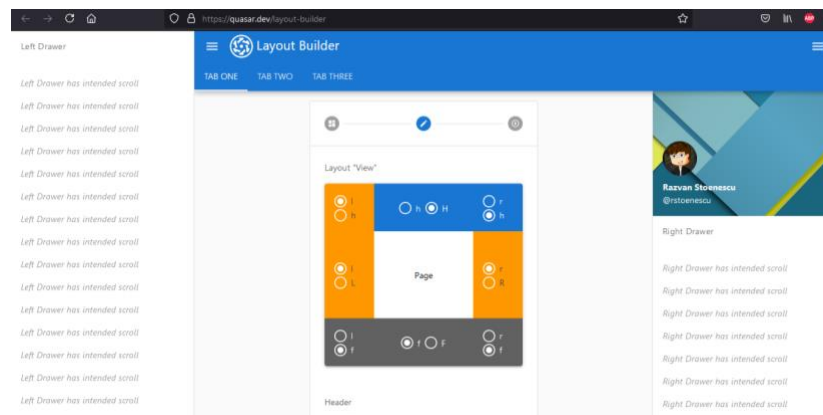


En el centro, encontramos un asistente que nos ayudará a construir nuestro **layout**. Podemos ir marcando y desmarcando las opciones, para ver cómo se va modificando. Una vez que tengamos clara la selección, en nuestro caso, dejaremos todo por defecto, y vamos al botón “Continue”.

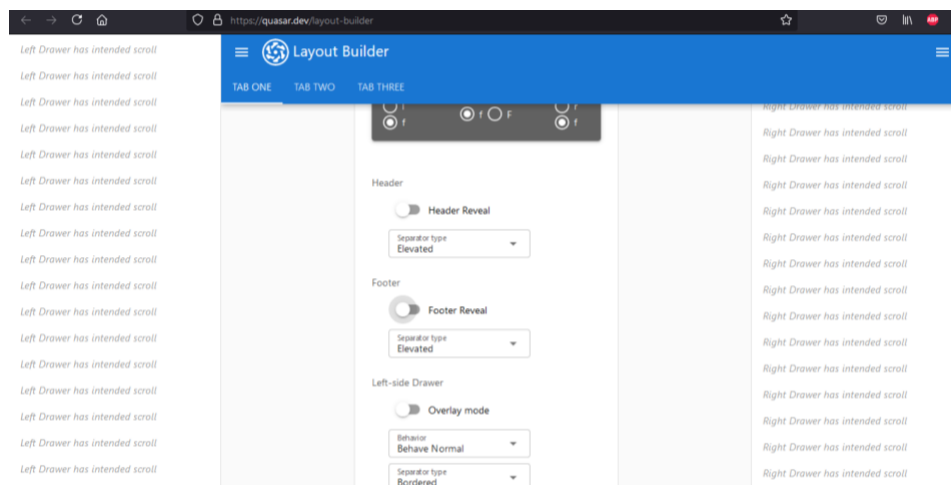
En el siguiente paso, vamos a definir como se comportarán los diferentes componentes que elegimos. Para entender sus funciones, la idea es que se pueda jugar y probar las diferentes opciones. Así descubrimos, por ejemplo, que la **(1)** de la barra lateral izquierda, más **(1)** del **header**, permiten establecer que dicha barra lateral izquierda esté por encima del menú superior.



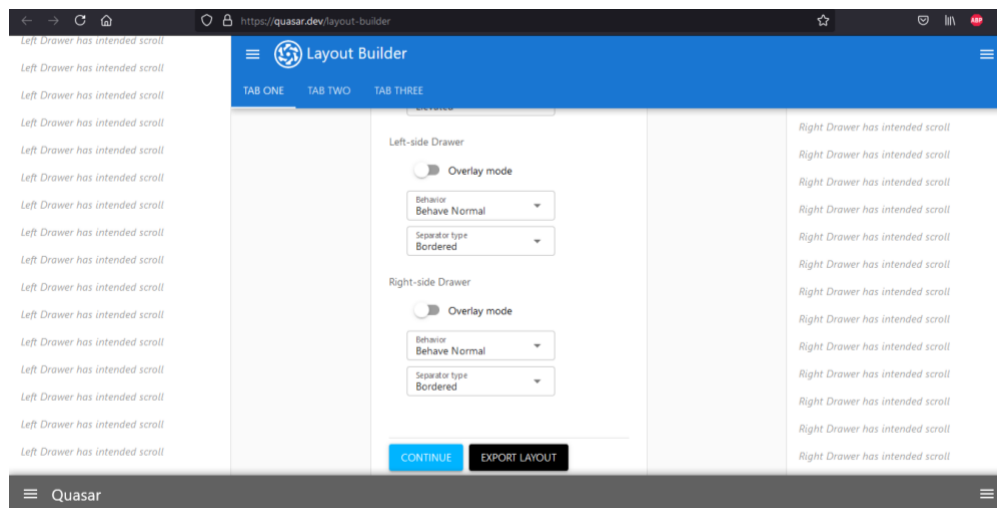
En este caso, por un tema de preferencias, se dejarán establecidos: para la barra de navegación **(H)**, que es **header fixed**; y **(h)** para que quede sobre el menú lateral derecho; al mismo menú derecho **(r)**, para que esté fijo y sin **scroll** interno; para el **footer**, **(f)** para que quede sobre el menú derecho, **(f)** para que sea visible solo al bajar la página, y **(f)** para que quede también sobre el menú izquierdo; y, para el menú izquierdo, **(1)** para que quede sin **scroll** interno, y **(1)** para que quede sobre el **header**.



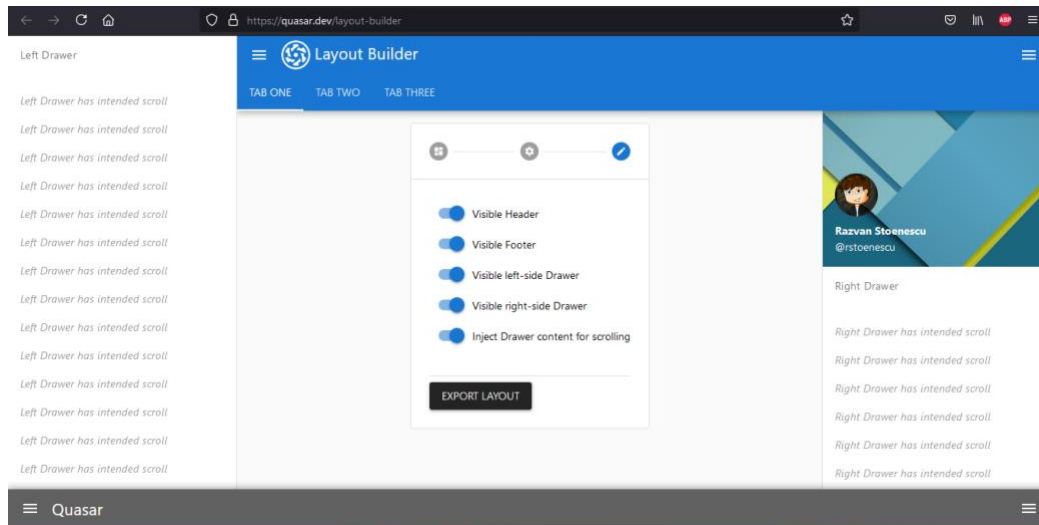
Si bajamos un poco en la página, encontramos otras opciones para personalizar. Tanto para el **header**, como para el **footer**, tenemos la posibilidad de darle un borde elevado o un borde sombreado. Los botones que dicen **header** y **footer**, nos permiten dejar visibles ambos elementos, aun cuando no estén fijos, son solo para facilitar la selección. Para ambos casos, lo dejaremos como elevado.



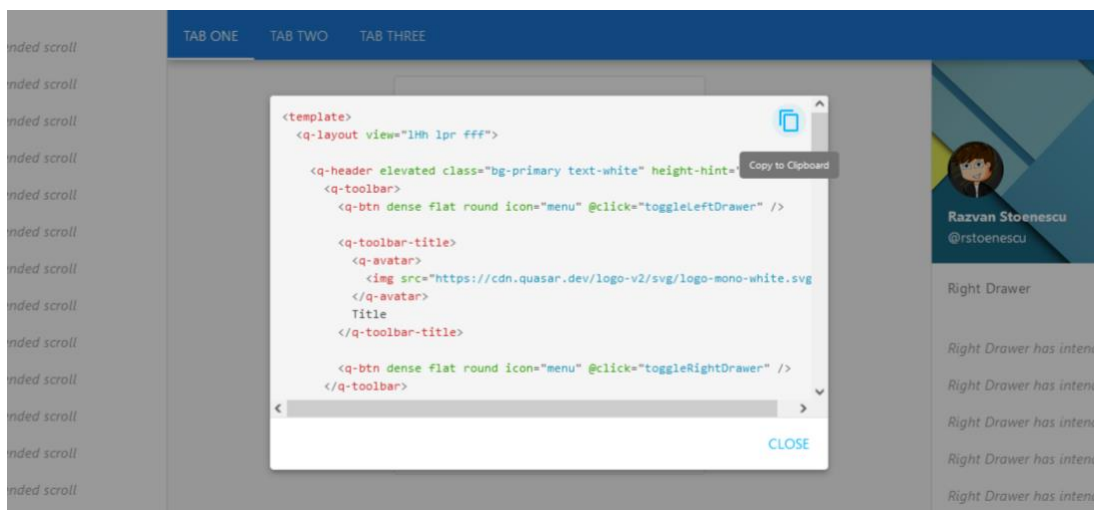
En la siguiente sección de configuración, tenemos la posibilidad de seleccionar el comportamiento de los menús laterales. Vamos a comenzar por **“Behave normal”** para ambos menús, y luego, el tipo de separador en **“Border”**, y presionamos el botón **“Continue”**.



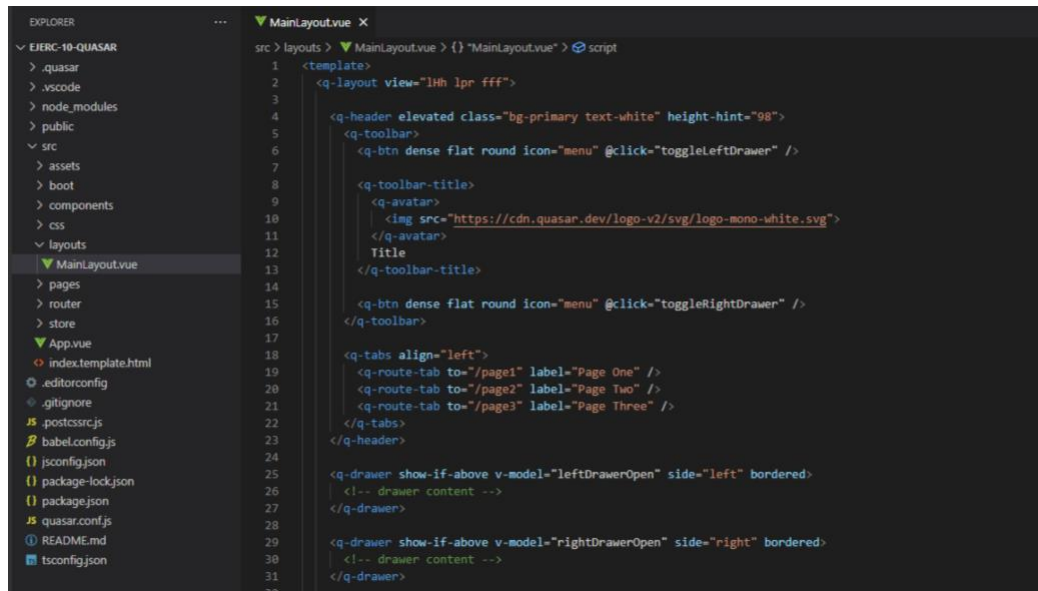
En la última vista de configuración del **layout**, nos permite confirmar lo que hemos elegido. Presionaremos el botón **“Export layout”**.



Aparece una ventana modal con el código de nuestro **layout**. Le damos clic al botón copiar que se encuentra en la parte superior derecha.



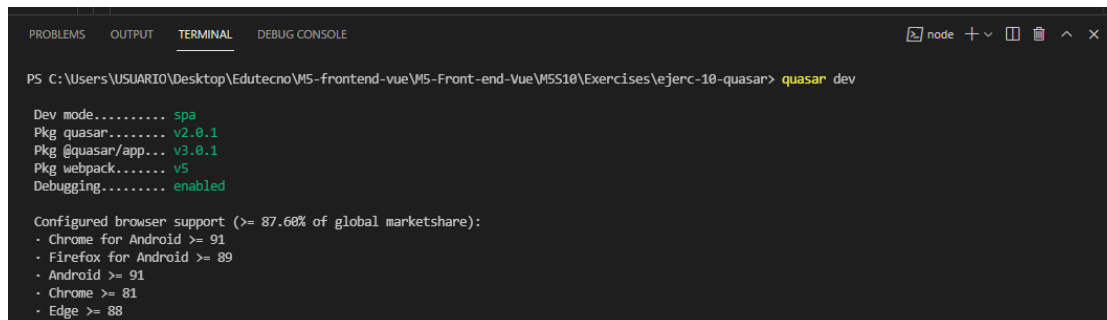
Y volvemos a nuestro editor. Borramos todo el contenido de **MainLayout.vue**, insertamos el código y guardamos.



```

src > layouts > MainLayout.vue > {} "MainLayout.vue" > script
1  <template>
2  <q-layout view="lHh lpr fff">
3
4  <q-header elevated class="bg-primary text-white" height-hint="98">
5    <q-toolbar>
6      <q-btn dense flat round icon="menu" @click="toggleLeftDrawer" />
7
8      <q-toolbar-title>
9        <q-avatar>
10         
11        </q-avatar>
12        Title
13      </q-toolbar-title>
14
15      <q-btn dense flat round icon="menu" @click="toggleRightDrawer" />
16    </q-toolbar>
17
18    <q-tabs align="left">
19      <q-route-tab to="/page1" label="Page One" />
20      <q-route-tab to="/page2" label="Page Two" />
21      <q-route-tab to="/page3" label="Page Three" />
22    </q-tabs>
23  </q-header>
24
25  <q-drawer show-if-above v-model="leftDrawerOpen" side="left" bordered>
26    <!-- drawer content -->
27  </q-drawer>
28
29  <q-drawer show-if-above v-model="rightDrawerOpen" side="right" bordered>
30    <!-- drawer content -->
31  </q-drawer>
32
33
  
```

Revisaremos cómo quedó nuestro **layout**, ejecutando el comando **"Quasar dev"** en la terminal.



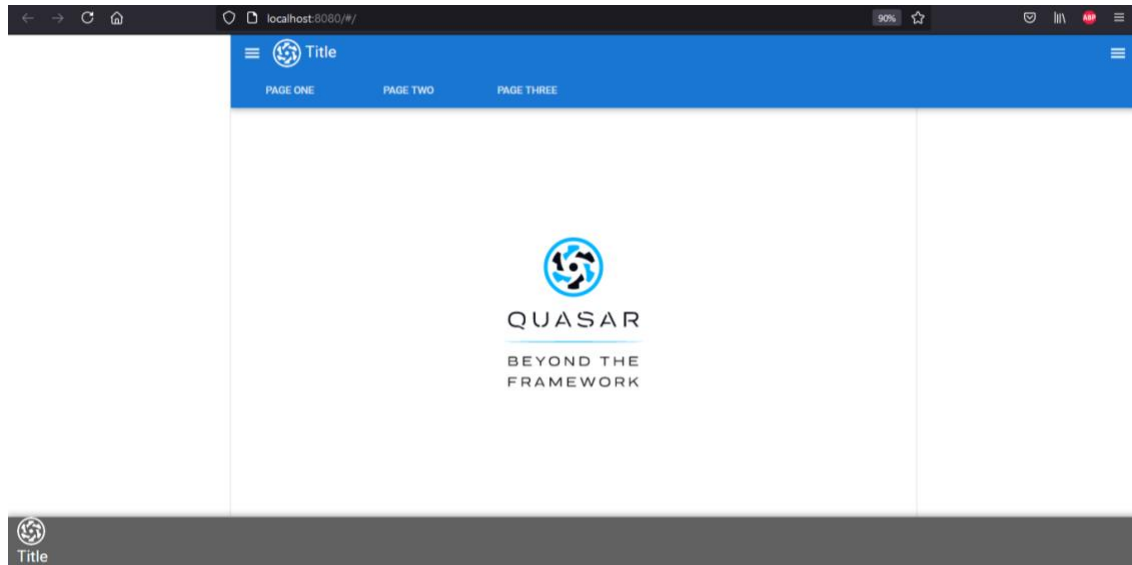
```

PS C:\Users\USUARIO\Desktop\Edutecno\W5-front-end-Vue\W5-Front-end-Vue\W5S10\Exercises\ejerc-10-quasar> quasar dev

Dev mode..... spa
Pkg quasar..... v2.0.1
Pkg @quasar/app... v3.0.1
Pkg webpack..... v5
Debugging..... enabled

Configured browser support (>= 87.60% of global marketshare):
- Chrome for Android >= 91
- Firefox for Android >= 89
- Android >= 91
- Chrome >= 81
- Edge >= 88
  
```

Cuando se abre la ventana, encontramos que nuestro código ha funcionado correctamente. Si bien los menús no son visibles, pues no tienen contenido, al presionar los botones es posible apreciar la configuración que le dimos.



## EXERCISE 3: DESARROLLO CON QUASAR CLI PARTE 2

### INTRODUCCIÓN

Para continuar, vamos a hacer un par de modificaciones al `layout`.

### MODIFICANDO EL LAYOUT

Dentro de `"MainLayout.vue"`, buscamos y localizamos el componente `"q-toolbar"`, casi en el inicio, y más adentro ubicamos `"q-toolbar-title"`. Allí tiene dos hijos, `"q-avatar"` y el texto "Title".



```

MainLayout.vue X
src > layouts > MainLayout.vue > {} MainLayout.vue > @template > @q-layout > @q-header.bg-primary.text-white > @q-toolbar > @q-toolbar title
1 <template>
2 <q-layout view="1dh 1pr fff">
3
4   <q-header elevated class="bg-primary text-white" height="58">
5     <q-toolbar>
6       <q-btn dense flat round icon="menu" @click="toggleLeftDrawer" />
7     <q-toolbar-title>
8       <q-avatar>
9         
10      </q-avatar>
11      <h1>
12      </h1>
13    </q-toolbar-title>
14    <q-btn dense flat round icon="menu" @click="toggleRightDrawer" />
15  </q-header>
16
17  <q-tabs align="left">
18    <q-route-tab to="/pages" label="Page One" />
19    <q-route-tab to="/pages" label="Page Two" />
20    <q-route-tab to="/pages" label="Page Three" />
21  </q-tabs>
22 </q-layout>
23 </template>
24

```

Modificaremos el título por **“Notaviso”**, y seguidamente vamos a **“q-drawer”**, al que dice **side:”**  
**left”**, y en su interior agregamos el siguiente código:

```

1 <q-drawer show-if-above v-model="leftDrawerOpen" side="left" bordered>
2 <q-list>
3   <q-item-label header>Menú de notas</q-item-label>
4   <q-item clickable to="/">
5     <q-item-section avatar>
6       <q-icon name="school" />
7     </q-item-section>
8     <q-item-section>
9       <q-item-label>Notas</q-item-label>
10    </q-item-section>
11  </q-item>
12 </q-list>
13 <!-- drawer content -->
14 </q-drawer>

```

Luego, vamos a **“q-drawer”**, al que dice **side:”right”**, y en su interior agregamos el siguiente código:

```

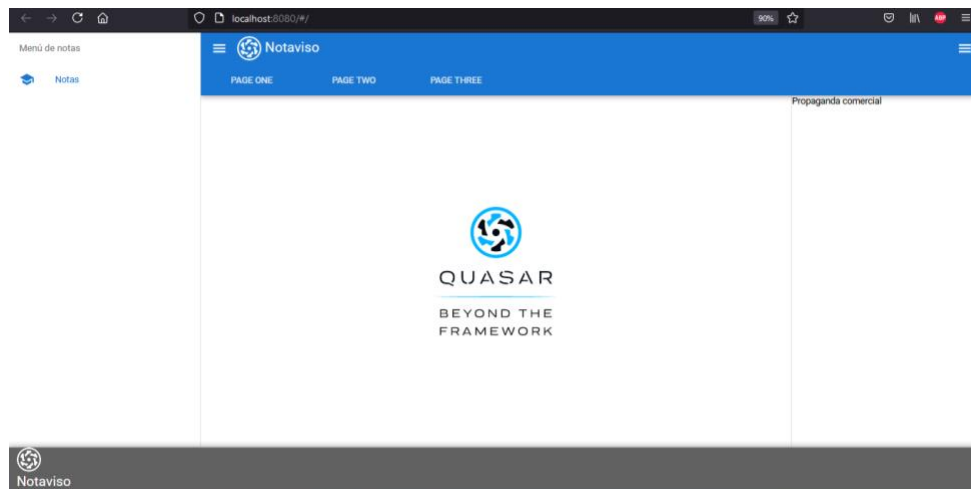
1 <q-drawer show-if-above v-model="rightDrawerOpen" side="right"
2 bordered>
3   <q-item-section>
4     <q-item-label>Propaganda comercial</q-item-label>
5   </q-item-section>
6 <!-- drawer content -->
7 </q-drawer>

```

Por último, cambiamos el título que se encuentra al interior del **“q- footer”**, en el **“q-toolbar-**  
**title”** del **“div”** hijo. Le escribimos: **“Notaviso”**.

```
1 <q-footer elevated class="bg-grey-8 text-white">
2   <q-toolbar>
3     <q-toolbar-title>
4       <q-avatar>
5         
7       </q-avatar>
8       <div>Notavisio</div>
9     </q-toolbar-title>
10  </q-toolbar>
11 </q-footer>
```

Como ya hemos realizado varios cambios pequeños, revisaremos en el navegador, y confirmamos que están funcionando.



## MODIFICANDO ESTILOS

Para continuar, haremos unos pequeños cambios en los estilos del proyecto. Ya que al crear el proyecto seleccionamos **SCSS**, iremos a **"src-css"**, al archivo **"quasar.variables.scss"**, y lo abrimos.

```

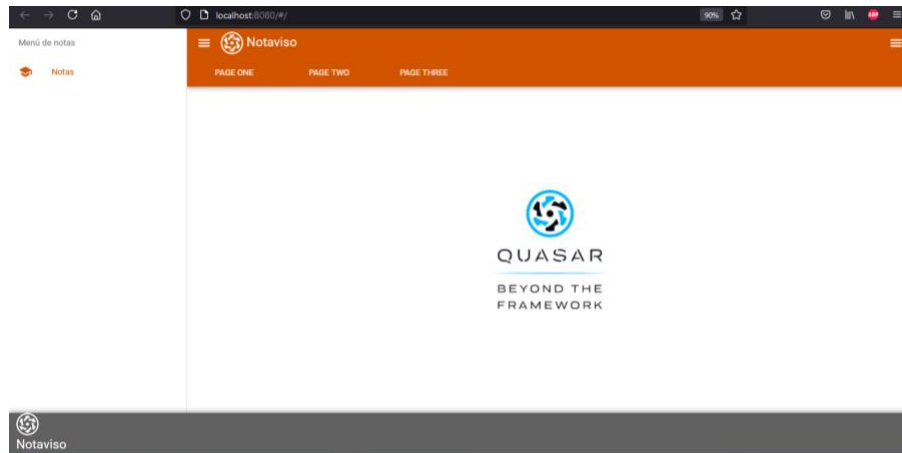
1 // Quasar SCSS (& Sass) Variables
2 // -----
3 // To customize the look and feel of this app, you can override
4 // the Sass/SCSS variables found in Quasar's source Sass/SCSS files.
5
6 // Check documentation for full list of Quasar variables
7
8 // Your own variables (that are declared here) and Quasar's own
9 // ones will be available out of the box in your .vue/.scss/.sass files
10
11 // It's highly recommended to change the default colors
12 // to match your app's branding.
13 // Tip: Use the "Theme Builder" on Quasar's documentation website.
14
15 $primary : #1976D2;
16 $secondary : #26A69A;
17 $accent : #9C27B0;
18
19 $dark : #1D1D1D;
20
21 $positive : #21BA45;
22 $negative : #C0392B;
23 $info : #31CCEC;
24 $warning : #F2C037;
25
  
```

Allí podemos encontrar rápidamente las variables de los colores, y podemos usar los de nuestra preferencia. Conservaremos: `$positive`, `$negative` y `$warning`. Para `$primary` utilizaremos `#d35400`; para `$secondary`, `#f2f3f4`; para `$accent`, `#884ea0`; para `$dark`, `#34495e`; y para `$info`, `#035afc`. Guardamos, y debería verse así:

```

1 // Quasar SCSS (& Sass) Variables
2 // -----
3 // To customize the look and feel of this app, you can override
4 // the Sass/SCSS variables found in Quasar's source Sass/SCSS files.
5
6 // Check documentation for full list of Quasar variables
7
8 // Your own variables (that are declared here) and Quasar's own
9 // ones will be available out of the box in your .vue/.scss/.sass files
10
11 // It's highly recommended to change the default colors
12 // to match your app's branding.
13 // Tip: Use the "Theme Builder" on Quasar's documentation website.
14
15 $primary : #d35400;
16 $secondary : #f2f3f4;
17 $accent : #884ea0;
18
19 $dark : #34495e;
20
21 $positive : #21BA45;
22 $negative : #C0392B;
23 $info : #035afc;
24 $warning : #F2C037;
25
  
```

Confirmamos los cambios en nuestra página.



Ya podemos ver nuestros cambios. Continuaremos con la creación de un componente.

## EXERCISE 4: DESARROLLO CON QUASAR CLI PARTE 3

### INTRODUCCIÓN

Ahora que tenemos listo el **layout**, y hemos aprendido a modificar los estilos, vamos a crear un componente simple.

### CREANDO UN COMPONENTE

Para esto, iremos a la documentación de **quasar**, que podemos encontrarla en el siguiente enlace:

<https://quasar.dev/vue-components/editor#Examples>

Search Quasar v2...

QUASAR v2.0.2

- Card
- Carousel
- Chat Message
- Chip
- Circular Progress
- Color Picker
- Dialog
- Editor - WYSIWYG
- Expansion Item
- Floating Action Button
- Form Components
  - Input Textfield
  - Select
  - File picker
  - Form
  - Field (wrapper)
  - Radio
  - Checkbox

## Editor (WYSIWYG)

The QEditor component is a WYSIWYG ("what you see is what you get") editor component that enables the user to write and even paste HTML. It uses the so-called Design Mode and the cross-browser `contentEditable` interface. Here are some go-to reference pages from the MDN webdocs with more detailed information about the underlying technology:

- [Making content editable](#)
- [Design Mode](#)
- [execCommand\(\) reference](#)
- [contentEditable spec](#)

### QEditor API

QEditor	Vue Component
Props 29	Events 1
Methods 7	Filter...
Behavior 3	fullscreen : Boolean
Description	

Bajaremos un poco, y encontramos un cuadro de editor de texto, que le serviría al usuario para escribir mensajes y editar sus estilos básicos. Para poder usarlo, necesitamos presionar el botón `<<`, `>>`, o `view source`.

Search Quasar v2...

QUASAR v2.0.2

- Card
- Carousel
- Chat Message
- Chip
- Circular Progress
- Color Picker
- Dialog
- Editor - WYSIWYG
- Expansion Item
- Floating Action Button
- Form Components
  - Input Textfield
  - Select
  - File picker
  - Form
  - Field (wrapper)
  - Radio
  - Checkbox

## Examples

Default editor

What you see is **what** you get.

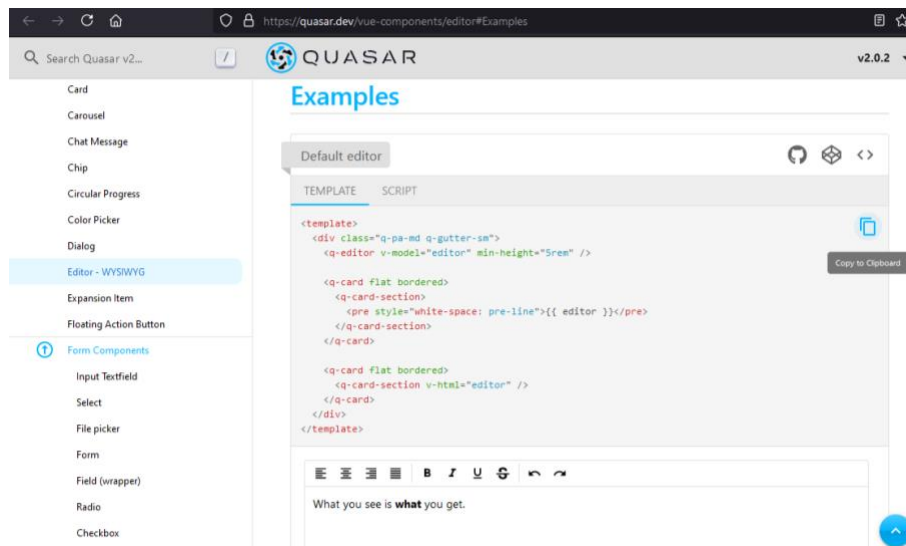
What you see is `<b>what</b>` you get.

What you see is **what** you get.

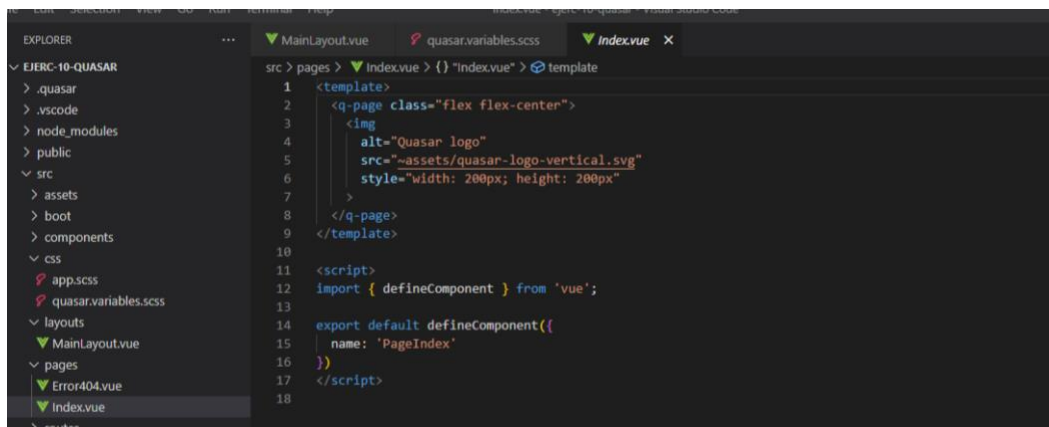
**WARNING**

In this first example, there are two cards below the editor. The first shows the unparsed html using the double-moustache, whereas the second shows the rendered version using `v-html="editor"`. Using `v-html` this way renders your users vulnerable to Cross Site Scripting attacks. If the content is user generated, be sure to sanitize it either on render or server side (or both).

Como podemos notar, acá nos muestra el código del elemento. Si presionamos el botón **“copy to clipboard”**.



Ya tenemos el código listo para insertarlo en el proyecto. Por eso, volvemos a nuestro editor de texto, nos dirigimos a la carpeta **“pages”**, y aquí nos encontramos con dos archivos: **“Error404.vue”** y **“Index.vue”**. Trabajaremos con el segundo.



Reemplazaremos el **“template”** del archivo, por el que recién copiamos en la página de **Quasar**. En algunos casos, puede aparecer un error, marcado en rojo. Si así sucede, podemos levantar la terminal, y ahí nos indica que éste es porque no encuentra la propiedad, o el método **“editor”**.

```

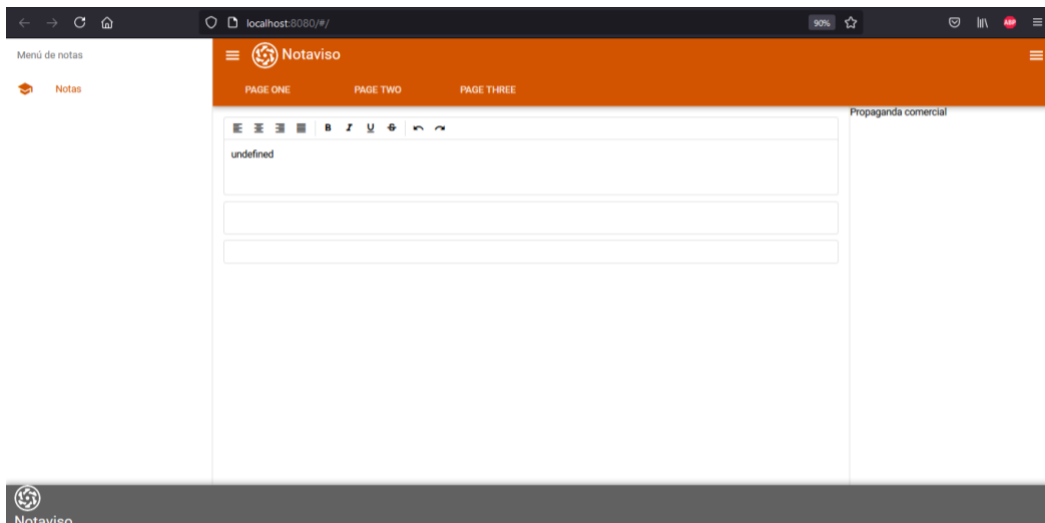
1 <template>
2   <div class="q-pa-md q-gutter-sm">
3     <q-editor v-model="editor" min-height="5rem" />
4
5     <q-card flat bordered>
6       <q-card-section>
7         <pre style="white-space: pre-line">{{ editor }}</pre>
8       </q-card-section>
9     </q-card>
10
11     <q-card flat bordered>
12       <q-card-section v-html="editor" />
13     </q-card>
14   </div>
15 </template>
16
17 <script>
18   import { defineComponent } from 'vue';
19
20   export default defineComponent({
21     name: 'PageIndex'
22   })
23 </script>
24

```

**PROBLEMS** 3

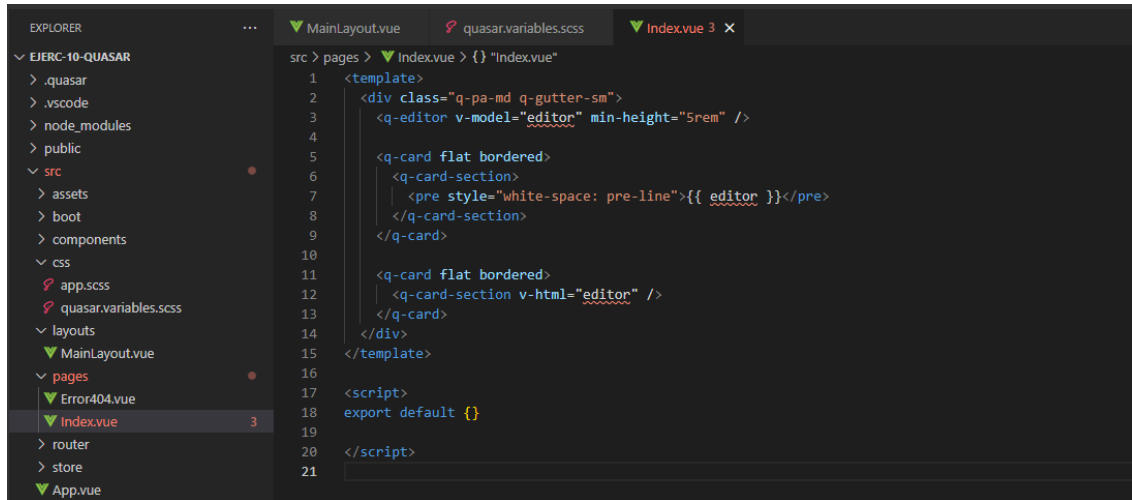
- Property 'editor' does not exist on type '{ \$: ComponentInternalInstance; \$data: {}; \$props: Partial<{}> & Omit<Readonly<{}> & {}> & VNodeProps & All...'
- Property 'editor' does not exist on type '{ \$: ComponentInternalInstance; \$data: {}; \$props: Partial<{}> & Omit<Readonly<{}> & {}> & VNodeProps & All...'
- Property 'editor' does not exist on type '{ \$: ComponentInternalInstance; \$data: {}; \$props: Partial<{}> & Omit<Readonly<{}> & {}> & VNodeProps & All...'

De igual forma guardamos, y vemos nuestro proyecto en el navegador.



Ya tenemos nuestro cuadro editor de texto, así que, en el caso de que salga el error, iremos a solucionarlo. Tanto si éste aparece, como si no, revisaremos dentro del **“script”** donde tenemos el

llamado al componente que venía por defecto. Entonces, vamos a borrar todo, excepto `“export default”`.



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows the project structure with 'Index.vue' selected under the 'pages' folder. The main editor displays the content of 'Index.vue' with line numbers 1 through 21. The code includes a template section with two cards: one containing a QEditor component and another containing a pre-formatted text block. The script section contains an 'export default' statement with an empty object.

```

1 <template>
2   <div class="q-pa-md q-gutter-sm">
3     <q-editor v-model="editor" min-height="5rem" />
4
5     <q-card flat bordered>
6       <q-card-section>
7         <pre style="white-space: pre-line">{{ editor }}</pre>
8       </q-card-section>
9     </q-card>
10
11     <q-card flat bordered>
12       <q-card-section v-html="editor" />
13     </q-card-section>
14   </div>
15 </template>
16
17 <script>
18   export default {}
19
20 </script>
21

```

Y, vamos a escribir en su interior:



The screenshot shows a close-up of the script section of the 'Index.vue' file. It displays the 'export default' statement with a data function that returns an object containing an 'editor' property initialized as an empty string.

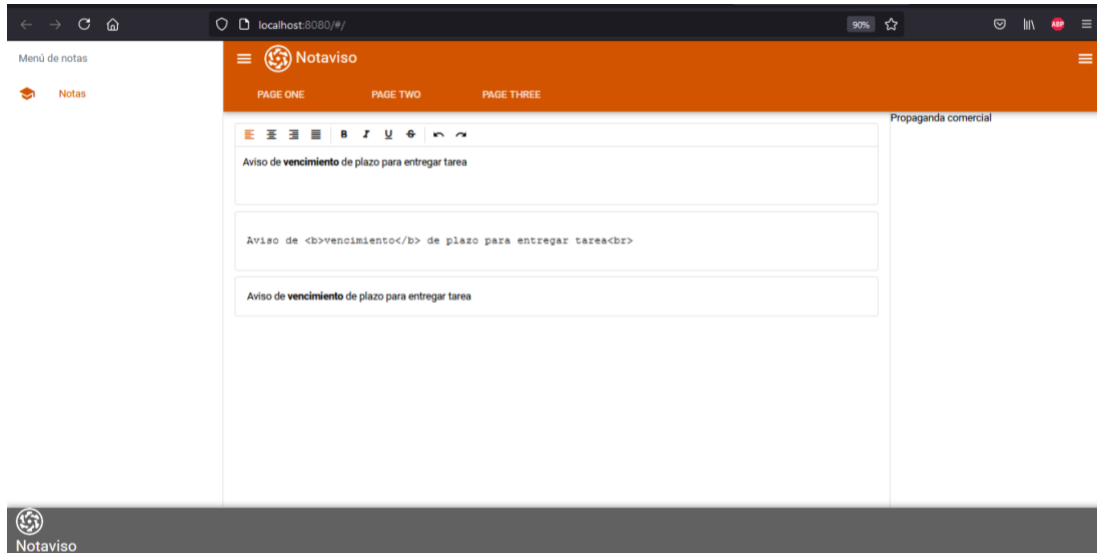
```

1 <script>
2 export default {
3   data() {
4     return {
5       editor: ''
6     }
7   },
8 }
9 </script>

```

Guardamos, y vemos que los problemas desaparecen. Ahora, vamos a ir a nuestro navegador, y probaremos. Se escribirá la frase: “Aviso de **vencimiento** de plazo para entregar tarea”. Notaremos que se comienza a mostrar en las 3 ventanitas.





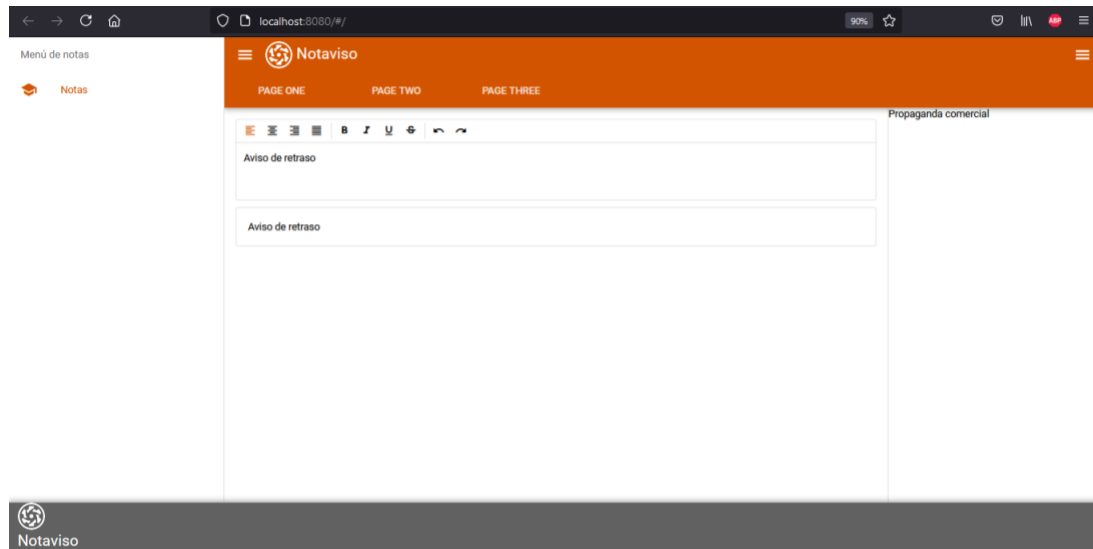
Pero nosotros no queremos que sea vea el texto plano, y sin estilos de la segunda ventana, así que la borraremos. Volvemos al editor de texto, la encontramos en nuestro código, dentro del **“template”**, y la borramos. Quedando de la siguiente forma:

```

1 <template>
2   <div class="q-pa-md q-gutter-sm">
3     <q-editor v-model="editor" min-height="5rem" />
4
5     <q-card flat bordered>
6
7     </q-card>
8
9     <q-card flat bordered>
10      <q-card-section v-html="editor" />
11    </q-card>
12  </div>
13</template>

```

Guardamos, revisamos en el navegador, y vemos que el segundo cuadro de texto ha desaparecido.



Finalmente, todo funciona correctamente.

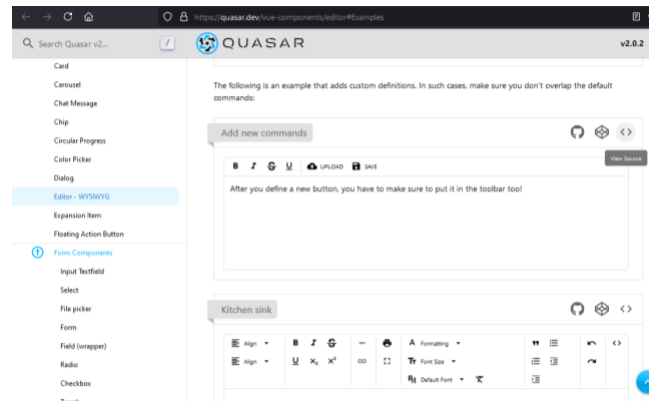
## EXERCISE 5: DESARROLLO CON QUASAR CLI PARTE 4

### INTRODUCCIÓN

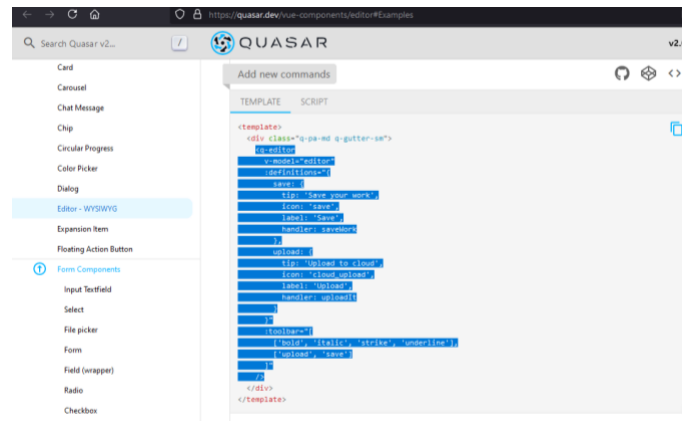
Ya tenemos lo principal del componente. Ahora, haremos un par de cambios que nos permitan darle funcionalidad, asociada a un plugin.

### MODIFICANDO EL EDITOR DE TEXTO

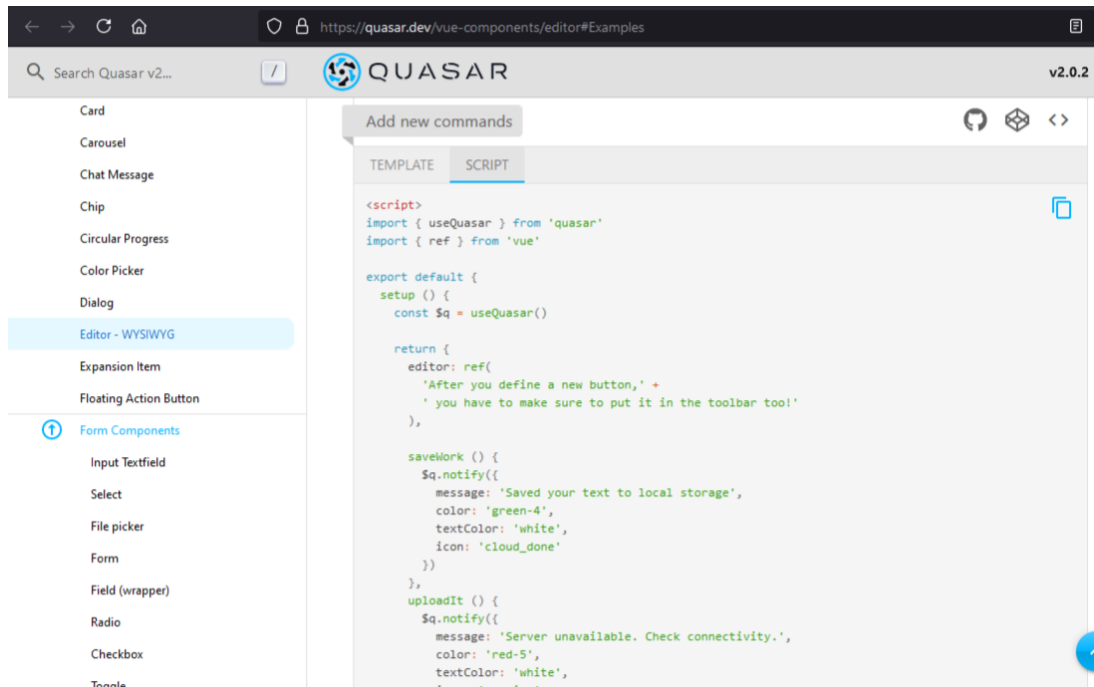
Lo siguiente que trabajaremos será conseguir un botón para guardar. Vamos a ir a la página de [Quasar](#), un poco más abajo, hasta donde encontremos un editor de texto con el botón guardar, y le damos clic a **“View source”**.



Vemos el código, identificamos el **“q-editor”**, y copiamos todo lo que contenga.



Regresamos a nuestro editor, y reemplazamos nuestro **“q-editor”** con lo que vamos a insertar ahora. En algunos casos, pueden aparecer problemas, marcados con rojo. Vamos a hacer las modificaciones para que quede correctamente. Comenzamos borrando el código que corresponde al botón **“upload”**, porque no lo utilizaremos. En el botón **“save”**, tenemos el **“handler”** **“savework”**, que no existe en nuestro código, pero, sí existe en la documentación de **Quasar**. Entonces volvemos a su página, y presionamos el botón **“script”**.



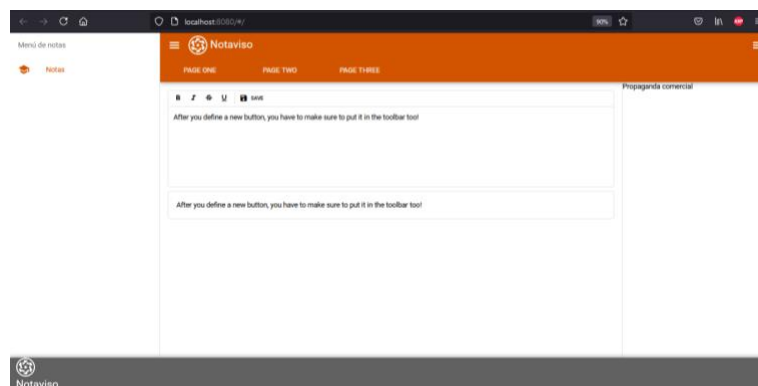
Aquí podemos encontrar nuestro método **“saveWork”**. Vamos a copiar todo lo que está en el **“script”**, lo llevamos a nuestro editor de texto, y reemplazamos acá en el **“script”** del proyecto. Borramos el **“Uploadit”**.

```

1 <script>
2 import { useQuasar } from 'quasar'
3 import { ref } from 'vue'
4
5 export default {
6   setup () {
7     const $q = useQuasar()
8     return {
9       editor: ref(
10        'After you define a new button,' +
11        ' you have to make sure to put it in the toolbar too!'
12      ),
13       saveWork () {
14         $q.notify({
15           message: 'Saved your text to local storage',
16           color: 'green-4',
17           textColor: 'white',
18           icon: 'cloud_done'
19         })
20       },
21     }
22   }
23 }
24 </script>

```

Veremos que desaparecen todos los problemas que nos marcaba la consola. Guardamos, y vamos a revisar al navegador para confirmar nuestros cambios.



Como todo está en orden, continuaremos modificando un par de detalles. Le cambiaremos el texto al **label** del botón “Save”, le pondremos “Guardar”, y al tip por “Guardar tu notaviso”.

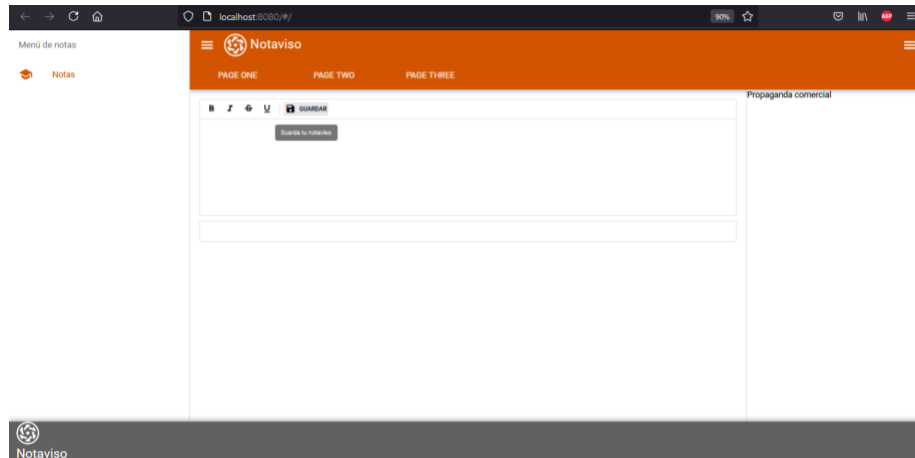
```
1 <q-editor
2   v-model="editor"
3   :definitions="{
4     save: {
5       tip: 'Guarda tu notaviso',
6       icon: 'save',
7       label: 'Guardar',
8       handler: saveWork
9     },
10  }"
11  :toolbar="[
12    ['bold', 'italic', 'strike', 'underline'],
13    ['upload', 'save']
14  ]"
15 />
```

En el **"script"**, vamos a borrar el contenido del "editor", y le vamos a colocar el **string** vacío.

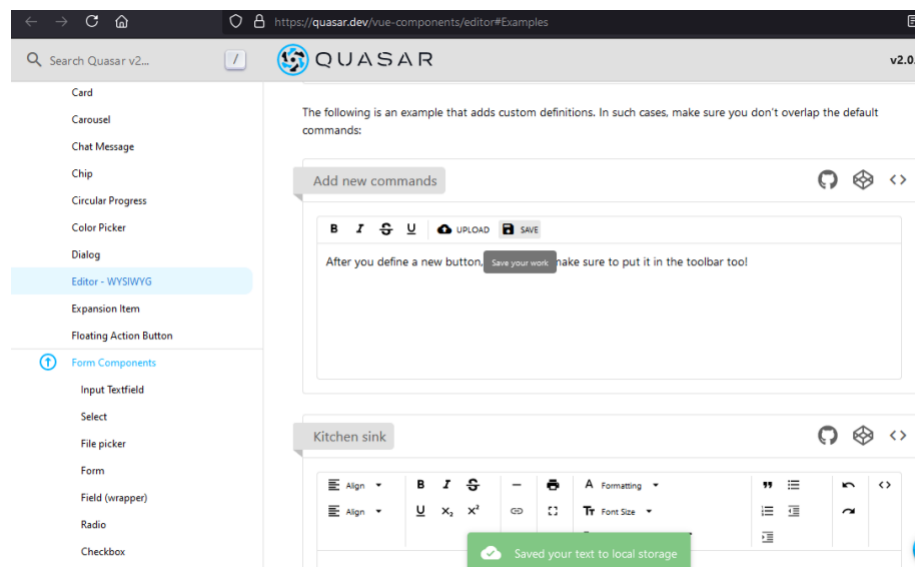
Guardamos, y confirmamos en el navegador.

```
1 export default {
2   setup () {
3     const $q = useQuasar()
4
5     return {
6       editor: '',
7
8       saveWork () {
9         $q.notify({
10           message: 'Saved your text to local storage',
11           color: 'green-4',
12           textColor: 'white',
13           icon: 'cloud_done'
14         })
15       },
16     }
17   }
18 }
```

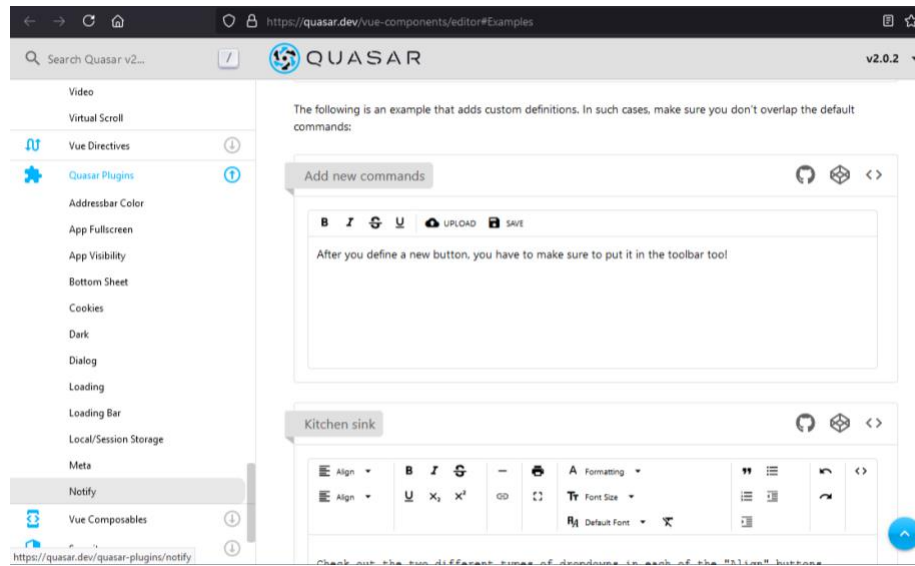
Guardamos y confirmamos en el navegador. Ahora, si presionamos el botón "Guardar", vemos que sale nuestro tip "Guardar tu notaviso", pero ningún mensaje que confirme la acción.



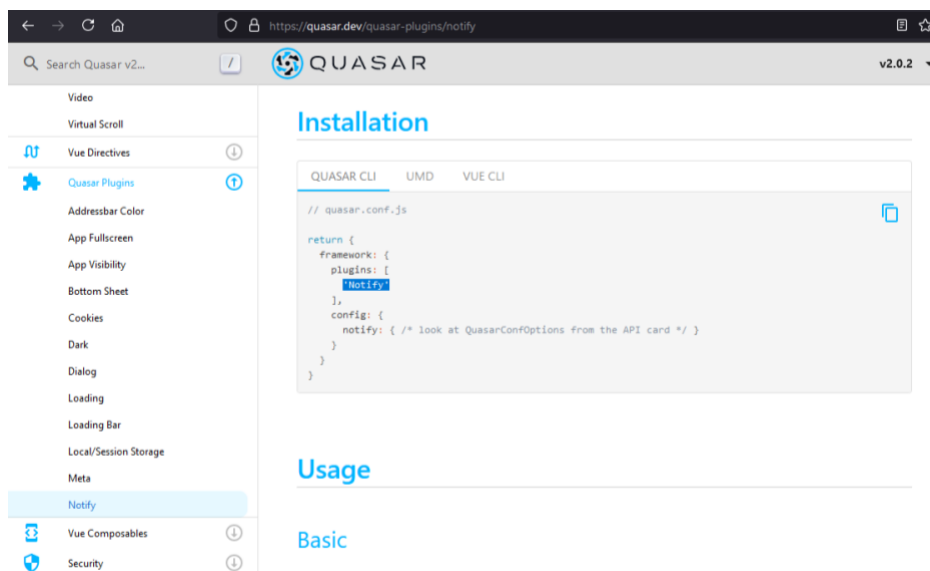
Sin embargo, si lo hacemos en la página de **Quasar**, vemos que sale un mensaje en verde. Eso sucede porque tiene integrado un plugin que nosotros no hemos integrado aún.



Lo iremos a buscar al menú lateral, donde dice **“Quasar-plugins”**, vamos a buscar **“Notify”**, y lo seleccionamos.

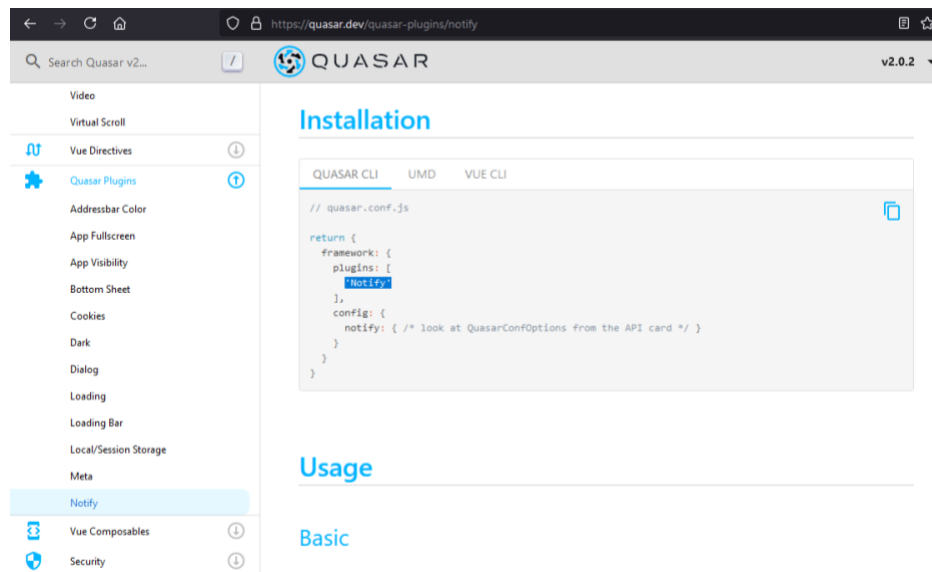


Revisamos el apartado de **“Installation”**, y copiamos el nombre del plugin.

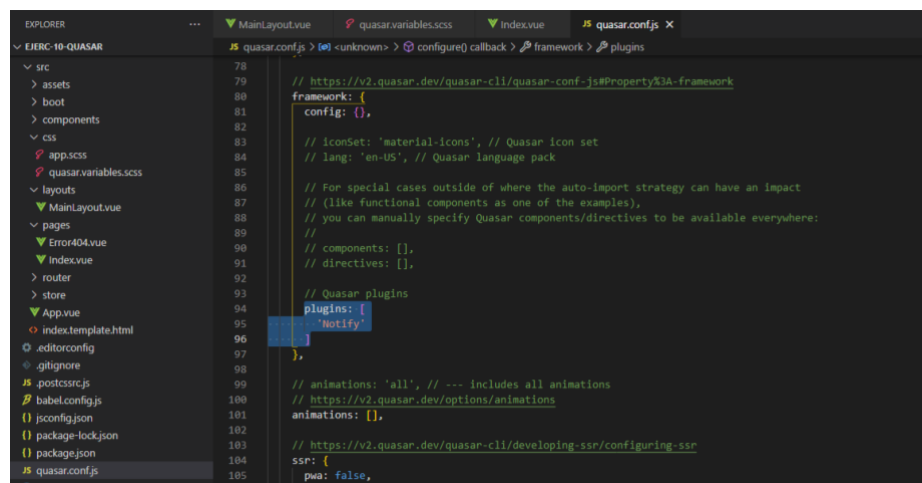




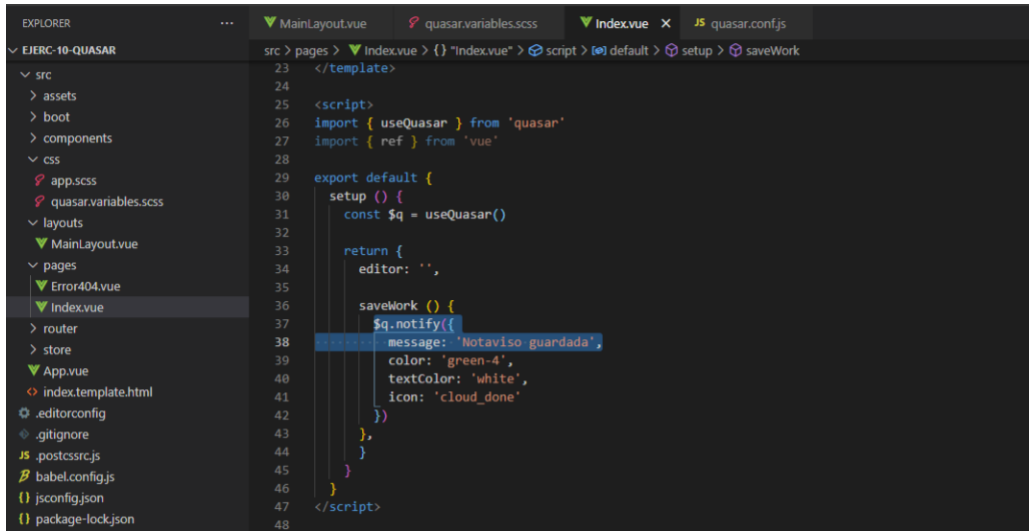
Ahora vamos a nuestro proyecto, abrimos el archivo **“quasar.config.js”**, y buscamos el apartado de **plugins**.



Dentro del paréntesis, insertamos el nombre del plugin que copiamos desde **Quasar**, y guardamos.



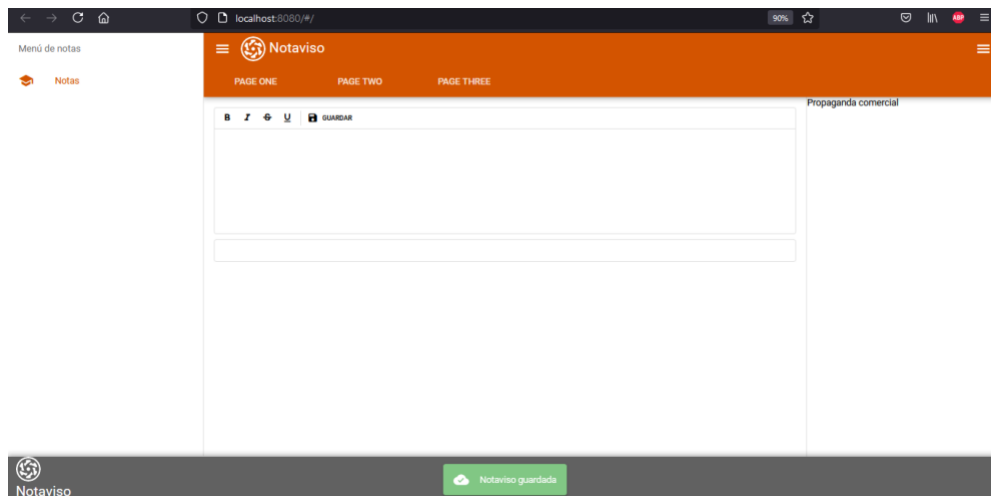
Si volvemos a **"Index.vue"**, nos encontramos que en el script de **"saveWork"** llamamos a **"q.notify"**, y tenemos todo configurado para emitir el mensaje. Vamos a modificarlo para que quede en español. Donde dice "message", ponemos "Notaviso guardada", y seleccionamos guardar.



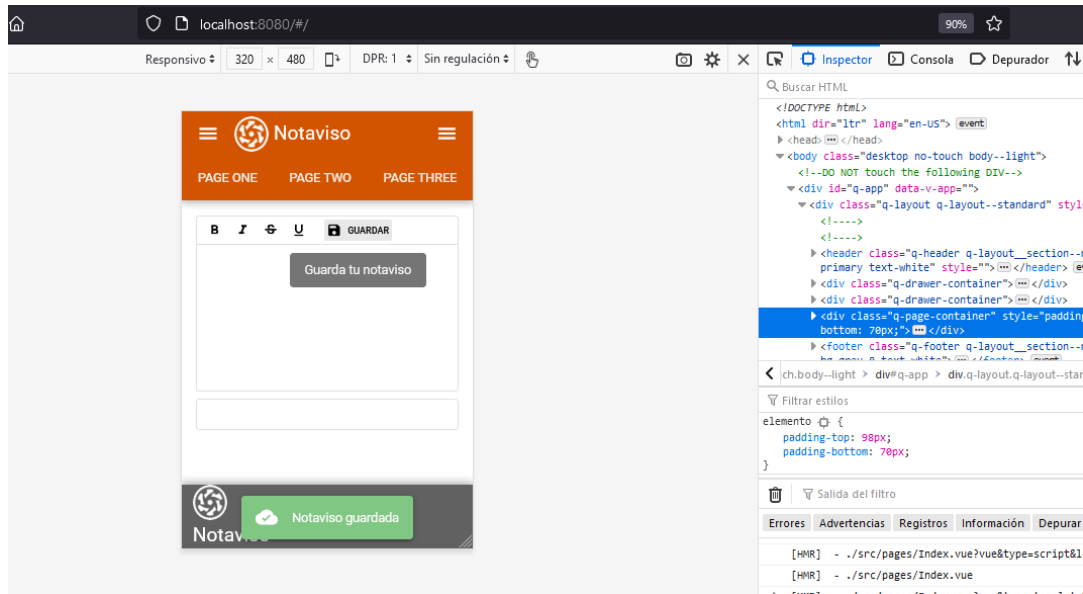
```

23 </template>
24
25 <script>
26 import { useQuasar } from 'quasar'
27 import { ref } from 'vue'
28
29 export default {
30   setup () {
31     const $q = useQuasar()
32
33     return {
34       editor: '',
35
36       saveWork () {
37         $q.notify({
38           message: 'Notaviso guardada',
39           color: 'green-4',
40           textColor: 'white',
41           icon: 'cloud_done'
42         })
43       },
44     }
45   },
46 }
47 </script>
48
  
```

La **"q-card"** del **template**, donde estaba el segundo cuadro, también lo podemos borrar. Guardamos, y así, con agregar una simple línea de código donde pusimos el plugin de **"Notify"**, ya tenemos todo listo y configurado para poder visualizar la notificación. Revisamos en el navegador, presionamos el botón "Guardar", y vemos el mensaje.



Una última confirmación de cómo se ve en formato **mobile**. Vamos a abrir el inspector, y presionamos el “modo de diseño reactivo”. Volvemos a presionar el botón “Guardar”, y confirmamos que efectivamente está funcionando bien.



Aunque faltan algunos detalles para terminar la funcionalidad, lo dejaremos hasta ahí. Con esto, damos por terminado el ejercicio de Quasar.