

EXERCISES QUE TRABAJAREMOS EN LA CUE

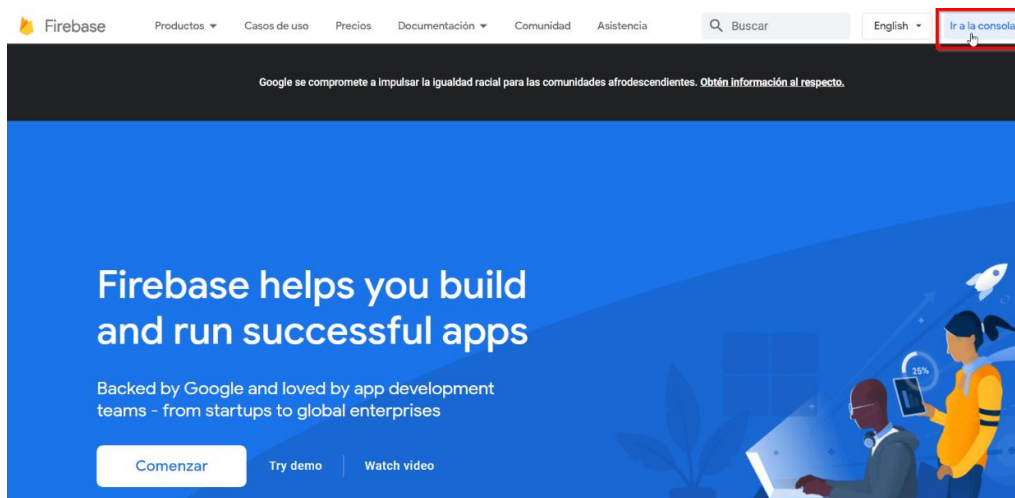
- EXERCISE 1: CONFIGURAR PROYECTO CON FIREBASE.
- EXERCISE 2: USO PRÁCTICO DE FIREBASE, AUTENTICACIÓN PARTE 1.
- EXERCISE 3: USO PRÁCTICO DE FIREBASE, AUTENTICACIÓN PARTE 2.
- EXERCISE 4: USO PRÁCTICO DE FIREBASE, AUTENTICACIÓN PARTE 3.
- EXERCISE 5: CONFIGURACIÓN REALTIME DATABASE.
- EXERCISE 6: USO PRÁCTICO REALTIME DATABASE.

EXERCISE 1: CONFIGURAR PROYECTO CON FIREBASE

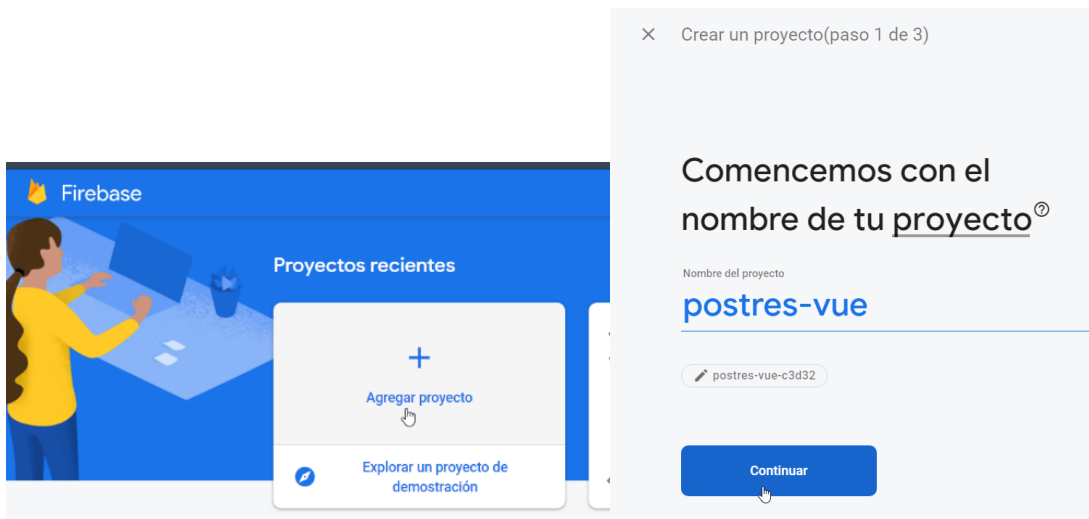
En el ejercicio de hoy, aprenderemos a usar **Firebase**, una plataforma ubicada en la nube, que nos facilitará la integración de nuestro proyecto para autenticaciones y base de datos.

Crearemos un proyecto, donde el usuario podrá ver los postres con stock sólo si inició sesión, si no lo hizo, sólo le aparecerán aquellos que están sin stock.

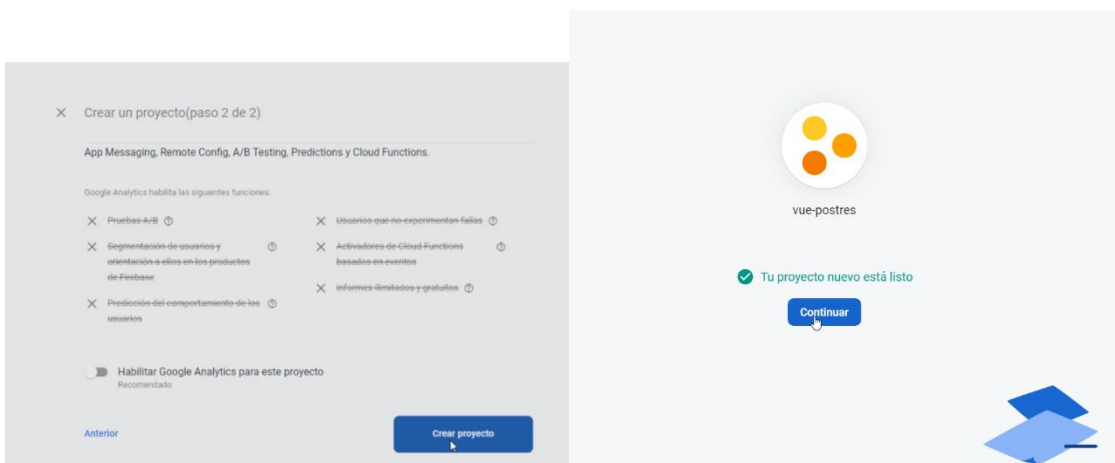
Lo primero que haremos es ir a la página web de **Firebase**: <https://firebase.google.com>, podremos acceder con cualquier cuenta de Gmail que tengamos; una vez iniciada la sesión, nos dirigimos al botón "Ir a la consola".



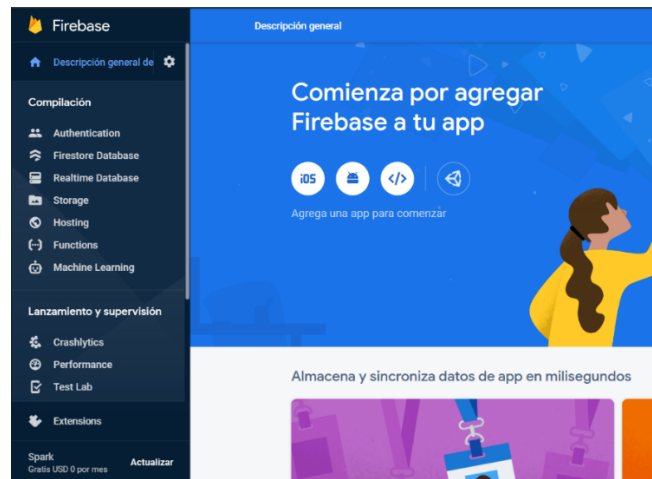
Nos aparecerá la opción de agregar un proyecto, presionamos, e ingresaremos el nombre, en este caso le pondremos “postres-vue”, para identificar el proyecto con el que trabajaremos, y presionamos Continuar.



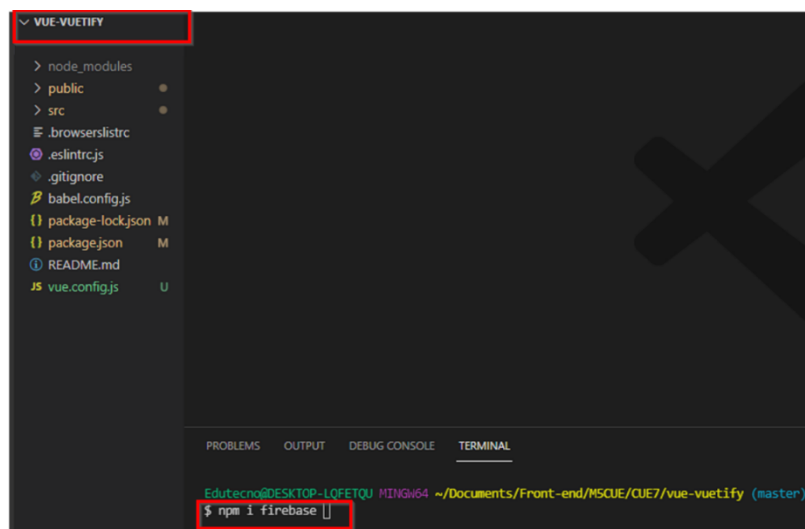
En la siguiente pantalla, deshabilitaremos Google Analytics, y presionaremos el botón “crear proyecto”, esperamos unos minutos a que termine el proceso, y clic a Continuar.



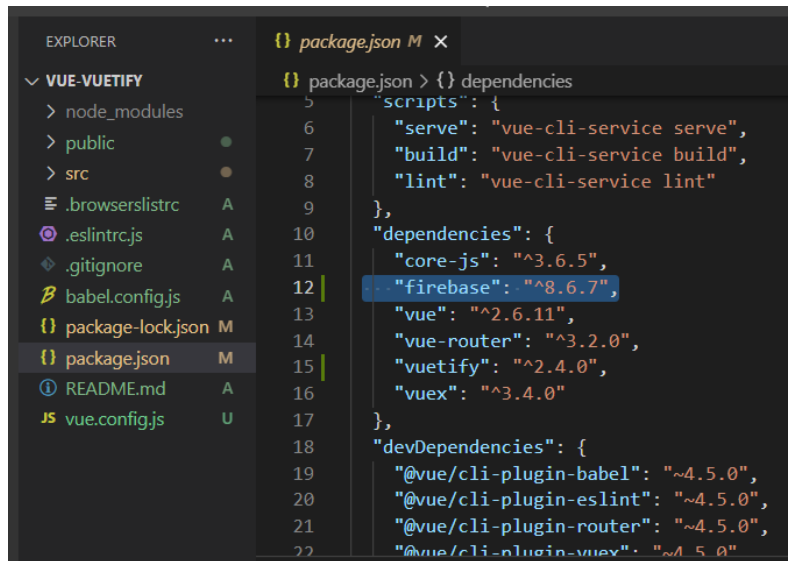
Inmediatamente, nos dirigirá a nuestro panel de funciones, y lo primero que nos indica es que debemos agregar **Firebase** a nuestra app.



Iremos a VSC, y abriremos el proyecto creado en el CUE anterior “vue-vuetify”. Nos dirigiremos a la consola, y verificaremos que nos encontramos en el proyecto que vamos a utilizar, una vez hecho, escribiremos el comando: `npm i firebase`, para integrar **Firebase**, y presionamos enter.



Una vez terminada la instalación, verificaremos que **Firebase** está en las dependencias de nuestro proyecto, para eso, nos dirigiremos al archivo **package.json**, y en dependencias deberíamos ver **Firebase** junto a su versión.

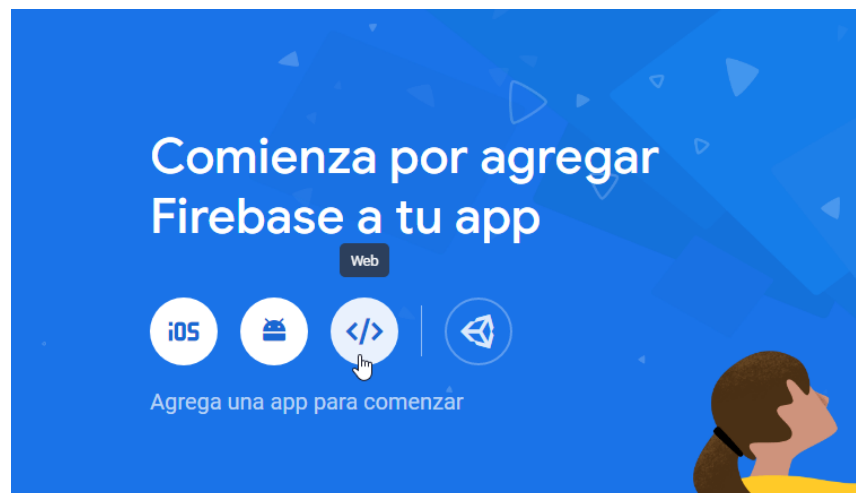


```

package.json > {} dependencies
5  "scripts": {
6    "serve": "vue-cli-service serve",
7    "build": "vue-cli-service build",
8    "lint": "vue-cli-service lint"
9  },
10 "dependencies": {
11   "core-js": "^3.6.5",
12   "firebase": "^8.6.7",
13   "vue": "^2.6.11",
14   "vue-router": "^3.2.0",
15   "vuetify": "^2.4.0",
16   "vuex": "^3.4.0"
17 },
18 "devDependencies": {
19   "@vue/cli-plugin-babel": "~4.5.0",
20   "@vue/cli-plugin-eslint": "~4.5.0",
21   "@vue/cli-plugin-router": "~4.5.0",
22   "@vue/cli-plugin-vuex": "~4.5.0"

```

Volveremos al panel de funciones de **Firebase**, y presionaremos el botón de código que dice “Web”.



Nos pedirá registrar una app, pondremos postres (esto para la ejemplificación), y le daremos clic al botón “Registrar app”.

×

Agrega Firebase a tu aplicación web

1

Registrar app

Sobrenombre de la app ⓘ

postres

☐

Además, configura **Firebase Hosting** para esta app. [Más información](#)

También puedes configurarlo más adelante. Puedes comenzar a usarlo sin costo en cualquier momento.

Registrar app

2

Agrega el SDK de Firebase

Después, nos mostrará una secuencia de comandos, el cual debemos copiar y conservar en algún archivo txt o algún documento, para utilizarlo más adelante, y presionaremos el botón “Ir a la consola”.

3

Agrega el SDK de Firebase

Copia y pega estas secuencias de comandos en la parte inferior de la etiqueta <body> antes de usar cualquier servicio de Firebase:

```

<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.6.7/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyA0J5Uw06opwy34MhG31fTKcP0r5Kxf-0w",
    authDomain: "vue-postres.firebaseio.com",
    projectId: "vue-postres",
    storageBucket: "vue-postres.appspot.com",
    messagingSenderId: "126982263758",
    appId: "1:126982263758:web:b42b454186b46c5a5f519f"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>

```

Obtén más información sobre Firebase para la Web con estos recursos: [Primeros pasos](#), [Referencia de API del SDK web](#) y [Muestras](#)

Copia y pega estas secuencias de comandos en la parte inferior de la etiqueta <body> antes de usar cualquier servicio de Firebase:

```

<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.6.7/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyA0J5Uw06opwy34MhG31fTKcP0r5Kxf-0w",
    authDomain: "vue-postres.firebaseio.com",
    projectId: "vue-postres",
    storageBucket: "vue-postres.appspot.com",
    messagingSenderId: "126982263758",
    appId: "1:126982263758:web:b42b454186b46c5a5f519f"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>

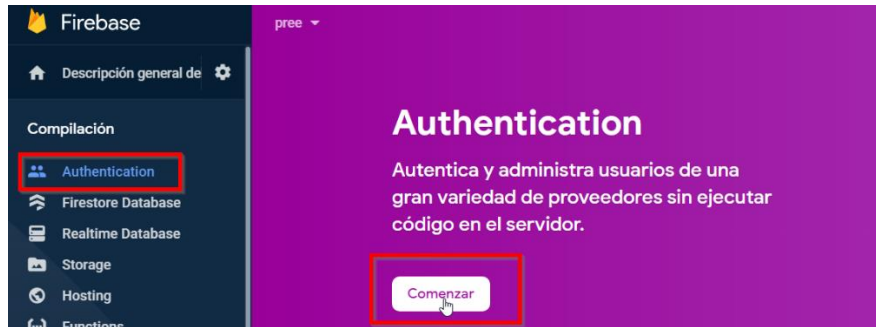
```

Obtén más información sobre Firebase para la Web con estos recursos: [Primeros pasos](#), [Referencia de API del SDK web](#) y [Muestras](#)

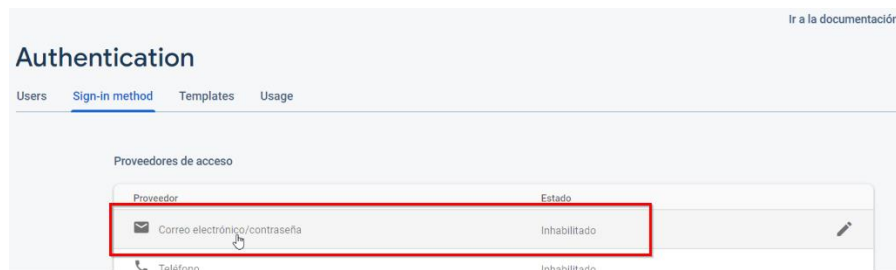
¿Usas npm o un agrupador como Webpack o Rollup? Consulta el [SDK modular](#) (actualmente en versión beta).

Ir a la consola

Lo siguiente que haremos será ir a authentication, y le daremos clic a Comenzar.



Elegiremos correo electrónico/contraseña.



Presionamos Habilitar, y guardamos.

Proveedores de acceso

Proveedor	Estado
Correo electrónico/contraseña	<div> <div></div> <div>Habilitar</div> </div> <p>Permite que los usuarios se registren con su dirección de correo electrónico y contraseña. Nuestros SDK también proporcionan verificación de la dirección de correo electrónico, recuperación de contraseñas y primitivas de cambio de dirección de correo. Más información</p> <div> <div></div> <div>Vínculo del correo electrónico (acceso sin contraseña)</div> </div> <div> <div></div> <div>Habilitar</div> </div>

Cancelar Guardar

Iremos a Users, y agregaremos un usuario. Aquí crearemos un correo de prueba y una contraseña, que utilizaremos más tarde para poder iniciar sesión en nuestra aplicación, escribiremos usuario@postres.cl y contraseña 123456, y presionaremos Agregar al usuario.

Authentication [Ir a la documentación](#)

Users | Sign-in method | Templates | Usage

➤ Crea prototipos y haz pruebas de extremo a extremo con Local Emulator Suite, que ahora es compatible con Firebase Authentication. [Comenzar](#)

Agregar usuario

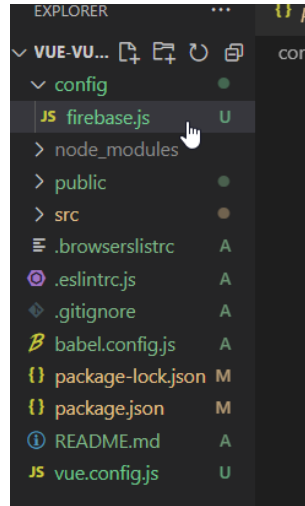
Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
Este proyecto todavía no tiene usuarios				

Agregar usuario

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
<p>Agrega un usuario con correo electrónico y contraseña</p> <div> <div>Correo electrónico</div> <div>Contraseña</div> </div> <div> <div>usuario@postres.cl</div> <div>123456</div> </div>				

Cancelar **Agregar usuario**

Ahora, iremos a nuestro proyecto en VSC, y haremos algunas modificaciones. Lo primero será crear la carpeta config en la raíz de nuestro proyecto, y dentro de ésta, el archivo **Firestore.js**.



En este archivo pegaremos los comandos que **Firebase** nos pidió que copiáramos.

```

config > JS firebase.js
1  <script>
2  // Your web app's Firebase configuration
3  var firebaseConfig = {
4    apiKey: "AIzaSyAoJ5Uw06opwy34MaH31fTKcP0r5KwF-0w",
5    authDomain: "vue-postres.firebaseio.com",
6    projectId: "vue-postres",
7    storageBucket: "vue-postres.appspot.com",
8    messagingSenderId: "126982263758",
9    appId: "1:126982263758:web:b42b454106b46c5a5f519f"
10  };
11
12  firebase.initializeApp(firebaseConfig);
13 </script>

```

Lo ordenaremos, y reemplazaremos **var FirebaseConfig** por **export default**, borrarémos los comentarios y las etiquetas **script**, y volvemos a ordenar.

Por último, cortaremos la línea de código: **firebase.initializeApp(firebaseConfig)** para utilizarla más tarde, y guardamos.


```
{ } package.json M JS firebase.js U X
config > JS firebase.js > ...
1   export default{
2     apiKey: "AIzaSyAoJ5Uw06opwy34MaH31fTKcP0r5Kwf-0w",
3     authDomain: "vue-postres.firebaseio.com",
4     projectId: "vue-postres",
5     storageBucket: "vue-postres.appspot.com",
6     messagingSenderId: "126982263758",
7     appId: "1:126982263758:web:b42b454106b46c5a5f519f"
8   };
9
10
```

Ahora, iremos a la carpeta `src`, y en el archivo `main.js` importaremos `Firebase`, escribiremos: `import Firebase from 'Firebase'`. También importaremos las configuraciones de `Firebase`, escribiendo: `import FirebaseConfig` desde la carpeta que creamos.

En la siguiente línea pegaremos el código que cortamos: `Firebase.initializeApp(FirebaseConfig)`, y agregaremos una variable `let app`, debajo de la configuración de `VUE`.

```
1 import Vue from 'vue'
2 import App from './App.vue'
3 import router from './router'
4 import store from './store'
5 import Firebase from 'Firebase'
6 import FirebaseConfig from '../config/Firebase'
7 import vuetify from './plugins/vuetify'
8
9 Firebase.initializeApp(FirebaseConfig)
10
11 Vue.config.productionTip = false
12 let app;
```

Después escribiremos: `Firebase.auth().onAuthStateChanged(() => {})`, y dentro de esta función colocaremos un: `if(!app){ app = new Vue}`.

Con esto, ya tenemos los primeros pasos de configuración de nuestro proyecto con **Firebase**.

```
1 import Vue from 'vue'
2 import App from './App.vue'
3 import router from './router'
4 import store from './store'
5 import Firebase from 'Firebase'
6 import FirebaseConfig from '../config/Firebase'
7 import vuetify from './plugins/vuetify'
8
9 Firebase.initializeApp(FirebaseConfig)
10
11 Vue.config.productionTip = false
12 let app;
13
14 Firebase.auth().onAuthStateChanged(() => {
15   if(!app){
16     app = new Vue({
17       router,
18       store,
19       vuetify,
20       render: h => h(App)
21     }).$mount('#app')
22   }
23 })
```

EXERCISE 2: USO PRÁCTICO DE FIREBASE, AUTENTICACIÓN PARTE 1

Continuaremos configurando nuestro proyecto. Ahora, en la carpeta store abriremos el archivo `index.js`, y crearemos dos variables: la primera será `currentUser`, que utilizaremos para la autenticación del usuario, será indefinida; y la segunda será un arreglo de objetos, que se llamará: `listaPostres`.

```
1 import Vue from 'vue'
2 import Vuex from 'vuex'
3 Vue.use(Vuex)
4 export default new Vuex.Store({
5   state: {
6     currentUser: undefined,
7     listaPostres: [
8       {id: 1, nombre: "Torta de Chocolate", stock: 0,
9        img: 'https://cdn.colombia.com/gastronomia/2015/03/24/tarta-de-
10 chocolate-3088.jpg'},
11       {id: 2, nombre: "Arroz con leche", stock: 0,
12        img: 'https://d1e3z2jco40k3v.cloudfront.net/-/media/mccormick-
13 us/recipes/espanol/800x800/arroz_con_leche_cremoso_recipes_800x800.jp
14 g'},
15       {id: 3, nombre: "Pie de limón", stock: 0, img: 'https://encrypted-
16 tbn0.gstatic.com/images?q=tbn:ANd9GcQgkhEE84sai-
17 LMvjG3LXEINCq6RcNBbZZVcA&usqp=CAU'},
18       {id: 4, nombre: "Duraznos con crema", stock: 4,
19        img: 'https://i.ytimg.com/vi/-q4eu_ieQZ0/sddefault.jpg'},
20       {id: 5, nombre: "Mousse de frambuesa", stock: 2,
21        img: 'https://3.bp.blogspot.com/-
22 90EvVPzqKk/WVsuA2O4NXI/AAAAAAAAAks/YiY3yQOw4DQsyGh_hHWdTbX1kF5I_L4gA
23 CLcBGAs/s1600/Pudding-de-chias-y-chocolate-y-mousse-de-frambuesa-
24 4.png'},
25       {id: 6, nombre: "Leche asada", stock: 10, img: 'https://img-
26 global.cpcdn.com/recipes/2521df3c8b4df6b7/751x532cq70/leche-asada-
27 foto-principal.jpg'},
28       {id: 7, nombre: "Cheesecake maracuyá", stock: 14,
29        img: 'https://mui.kitchen/__export/1622926268148/sites/muikitchen/img/
30 2021/06/05/gisela-kretzschar-a_hnk7r6zua-
31 unsplash.jpg_65302158.jpg'}]
32   })
```

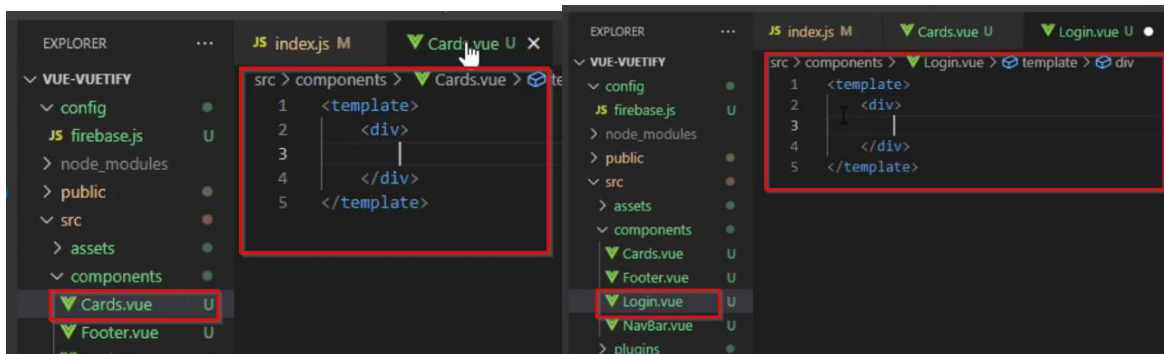
Crearemos una **mutation** para el usuario, y escribiremos en mayúscula: **SET_USER**.

Ahora, crearemos la **action** que tendrá el nombre de la **mutation**, pero en minúscula: **setUser**, y aquí invocamos a la **mutation** que queremos utilizar.

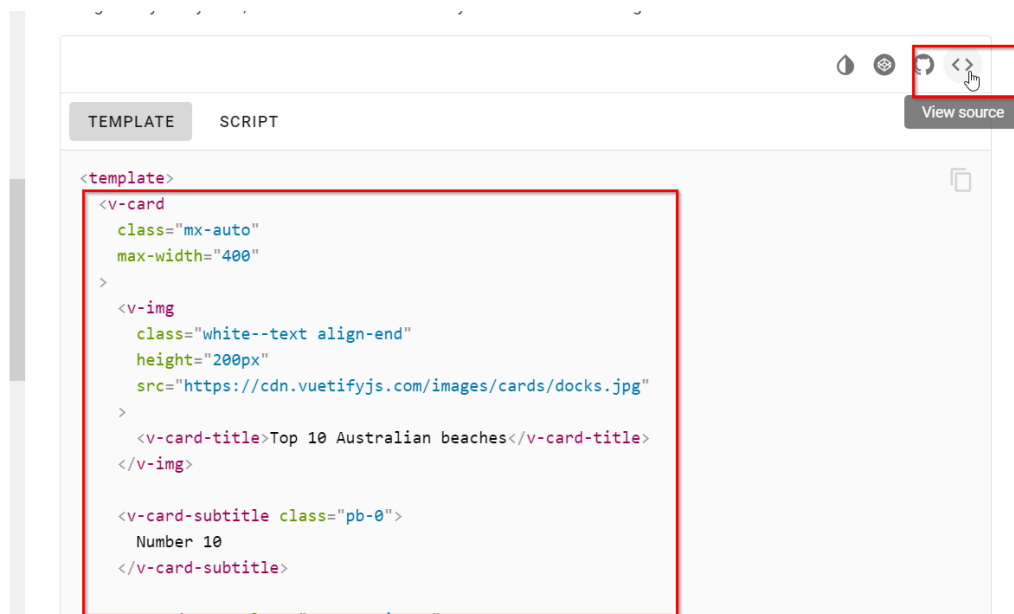
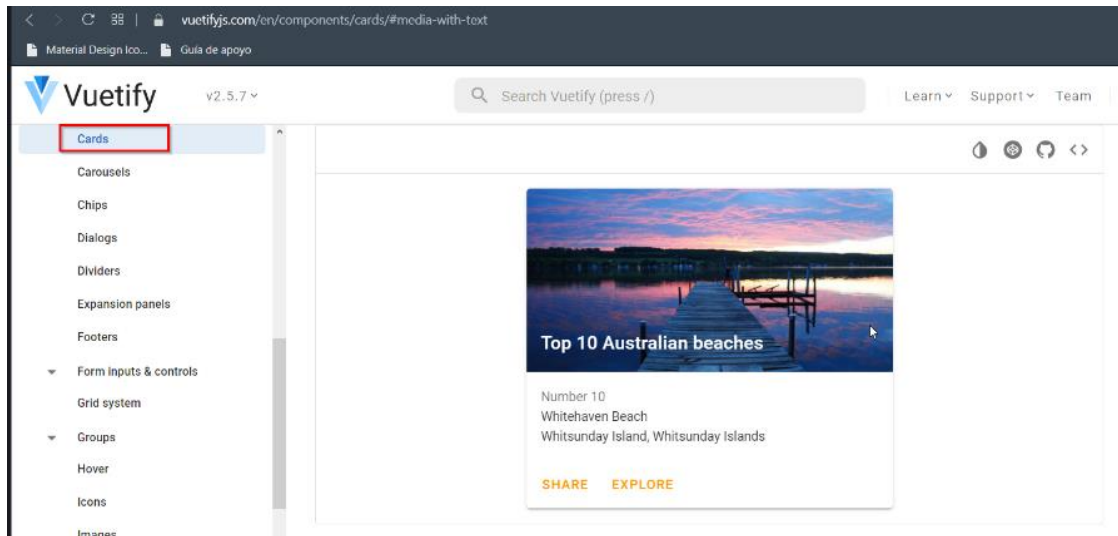
```

1 mutations: {
2   SET_USER (state, user) {
3     state.currentUser = user
4   }
5 },
6 actions: {
7   setUser ({commit}, user) {
8     commit ('SET_USER', user)
9   }
10 }
  
```

Crearemos dos componentes nuevos que reutilizaremos, el primero será: **Cards.vue** y **Login.vue**, y a cada uno le agregaremos el **template** y un elemento raíz **<div>**.



Empezaremos agregando elementos al componente **cards**, iremos a la página de **Vuetify**, y en sus componentes buscaremos **cards**, utilizaremos una básica, copiaremos todo lo de las etiquetas **v-card**.



Pegamos en nuestro componente, dentro de las etiquetas `<v-col>`, borraremos `<v-card-actions>`, y ordenamos nuestro código, dejando sólo un `<div>`, al que le agregaremos el texto `lorem ipsum`.

```

1 <template>
2   <div class="container">
3     <v-row>
4       <v-col>
5         <v-card class="mx-auto" max-width="400">
6           <v-img class="white--text align-end" height="200px">
7             <v-card-title> </v-card-title>
8           </v-img>
9           <v-card-subtitle class="pb-0">
10            </v-card-subtitle class="pb-0">
11            </v-card-subtitle>
12            <v-card-text class="text--primary">
13              <div>
14                Lorem ipsum dolor, sit amet consectetur adipisicing elit.
15                Esse,elit assumenda neque quis accusamus soluta facilis sapiente.
16              </div>
17            </v-card-text>
18          </v-card>
19        </v-col>
20      </v-row>
21    </div>
22  </template>

```

Ahora, lo que debemos hacer, es mostrar la información de los postres que tengan stock en cada `cards`, en éstas se mostrarán: el nombre, la imagen y cuántos quedan de cada uno.

Esa información la tenemos que sacar de nuestro arreglo `"listaPostres"`, que se encuentra en el archivo `index.js` de la carpeta `store`.

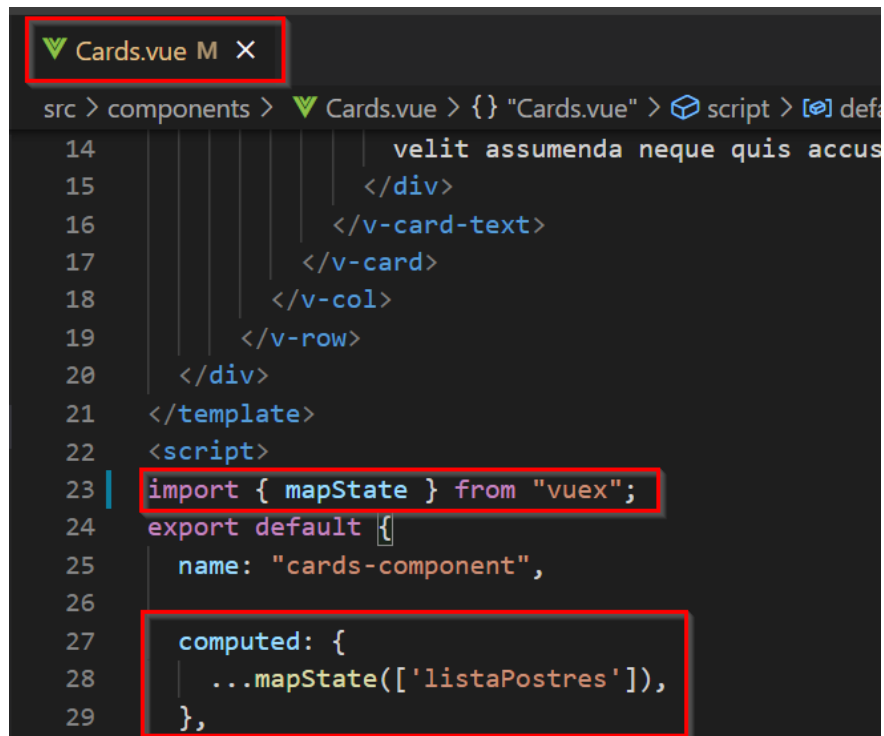
```

listaPostres: [
  {id: 1, nombre: "Torta de Chocolate", stock: 0, img: 'https://cdn.colombia.com/gastronomia/2015/03/24/tarta-de-chocol
  {id: 2, nombre: "Arroz con leche", stock: 0, img: 'https://d1e3z2jco40k3v.cloudfront.net/-/media/mccormick-us/recipes/e
  {id: 3, nombre: "Pie de limón", stock: 0, img: 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQgkhEE84sai-LMyj
  {id: 5, nombre: "Mousse de frambuesa", stock: 2, img: 'https://3.bp.blogspot.com/-90EvVPzqKk/WVsuA204NXI/AAAAAAAAAks/Y
  {id: 6, nombre: "Leche asada", stock: 10, img: 'https://img-global.cpcdn.com/recipes/2521df3c8b4df6b7/751x532cq70/lech
  {id: 7, nombre: "Cheesecake maracuyá", stock: 14, img: 'https://mui.kitchen/_export/1622926268148/sites/muikitchen/im
],

```

Para acceder a él, tenemos que importar el `state`, que es donde se encuentra y escribiremos en las etiquetas: `script import {mapState} from 'vuex'`.

Ahora crearemos el objeto `computed`, para indicar la variable que necesitamos del `state` `...mapState(['listaPostres'])`.



```
src > components > ▼ Cards.vue > {} "Cards.vue" > script > [🔗] default
14 |         velit assumenda neque quis accus
15 |     </div>
16 | </v-card-text>
17 | </v-card>
18 | </v-col>
19 | </v-row>
20 | </div>
21 | </template>
22 | <script>
23 | import { mapState } from "vuex";
24 | export default {
25 |     name: "cards-component",
26 |
27 |     computed: {
28 |         ...mapState(['listaPostres']),
29 |     },

```

En la etiqueta **v-col**, haremos el recorrido del arreglo y utilizaremos la directiva **v-for**, donde un postre recorrerá la lista de postres. En la etiqueta **v-card** agregaremos la directiva **v-if**, esto para que sólo traiga los postres con stock mayor a 0, en la imagen colocaremos **:src "postre.img"**, como título de la **card** pondremos el nombre, y finalmente en el subtítulo escribiremos "stock".

```
1 <template>
2   <div class="container">
3     <v-row>
4       <v-col v-for="postre in listaPostres" :key="postre.id">
5         <v-card v-if="postre.stock>0" class="mx-auto" max-
6 width="400">
7           <v-img class="white--text align-end" height="200px"
8 :src="postre.img">
9           <v-card-title> {{ postre.nombre }} </v-card-title>
10          </v-img>
11          <v-card-subtitle class="pb-0">Stock {{postre.stock }}
12          </v-card-subtitle>
13          <v-card-text class="text--primary">
14            <div>Lorem ipsum dolor, sit amet consectetur adipisicing
15 elit. Esse, velit assumenda neque quis accusamus soluta facilis
16 sapiente.</div>
17          </v-card-text>
18          </v-card>
19        </v-col>
20      </v-row>
21    </div>
22  </template>
```


Ahora crearemos la vista `Postres.vue`, escribiremos el `template`, un elemento base `div`, y dentro colocaremos una etiqueta `v-main`; dentro de éstas generaremos un `v-container`, que contendrá un título `<h1>` "Bienvenido", y una etiqueta `<h2>` que diga: "Listado de postres con stock".

```
1 <template>
2   <div>
3     <v-main>
4       <v-container>
5         <h1>Bienvenido</h1>
6         <h2>Listado de postres con stock</h2>
7       </v-container>
8     </v-main>
9   </div>
10 </template>
```

Importaremos el componente `cards`, lo agregaremos al objeto `components`, y por último lo invocaremos en el `template` como etiqueta.

```
1 <template>
2   <div>
3     <v-main>
4       <v-container>
5         <h1>Bienvenido</h1>
6         <h2>Listado de postres con stock</h2>
7         <cards-comp></cards-comp>
8       </v-container>
9     </v-main>
10  </div>
11 </template>
12 <script>
13
14 import CardsComp from "../components/Cards.vue";
15 export default {
16   components: {
17     CardsComp
18   }
19 };
20 </script>
```

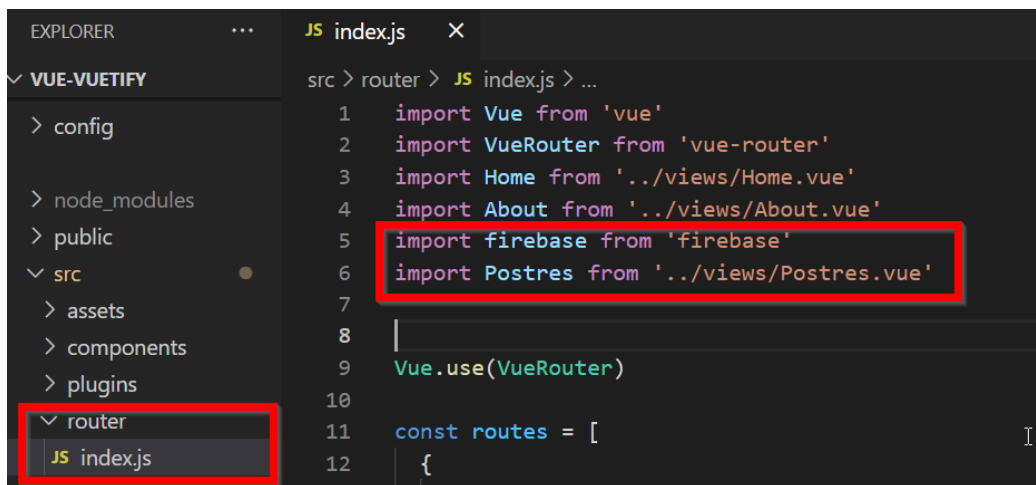
También invocaremos el email del usuario, para que el saludo sea más personalizado de acuerdo con quién inició sesión. Para eso, debemos llegar al usuario creado en el **state**, importamos el **mapstate**, que nos da acceso a todas las variables que tengamos, y ahora lo invocamos en el objeto **computed** y al lado de la palabra bienvenidos, agregaremos: **{{currentUser.email}}**.

```
1 <template>
2   <div>
3     <v-main>
4       <v-container>
5         <h1>Bienvenido {{currentUser.email}} </h1>
6         <h2>Listado de postres con stock</h2>
7         <cards-comp></cards-comp>
8       </v-container>
9     </v-main>
10  </div>
11 </template>
12 <script>
13 import {mapState} from 'vuex'
14 import CardsComp from "../components/Cards.vue";
15 export default {
16   data() {
17     return {}
18   },
19   Components:{
20     CardsComp
21   },
22   computed:{
23     ...mapState (["currentUser"])
24   },
25 };
26 </script>
```

Con eso tenemos creado nuestra vista **“postres”**, y la **card** que traerá la información de todos los postres que tengamos.

EXERCISE 3: USO PRÁCTICO DE FIREBASE, AUTENTICACIÓN PARTE 2

Ahora crearemos la ruta para linkear nuestra vista **"postres"**. Para ello, abriremos el archivo **index.js** de la carpeta **router**, y agregaremos: **import Postres from '../views/Postres.vue'**, también importaremos: **firebase** **import firebase from 'firebase'**.



```

EXPLORER
  VUE-VUETIFY
    > config
    > node_modules
    > public
    > src
      > assets
      > components
      > plugins
      > router
        JS index.js
  JS index.js
src > router > JS index.js > ...
1  import Vue from 'vue'
2  import VueRouter from 'vue-router'
3  import Home from '../views/Home.vue'
4  import About from '../views/About.vue'
5  import firebase from 'firebase'
6  import Postres from '../views/Postres.vue'
7
8
9  Vue.use(VueRouter)
10
11 const routes = [
12   {

```

Agregaremos un nuevo objeto, colocamos: **path: '/postres'**, **name: 'Postres'** y **component: Postres**, pero lo distinto que tendrá esta ruta será que sólo se podrá ver si el usuario está logueado, para esto agregaremos: **meta: { login: true }**.



```

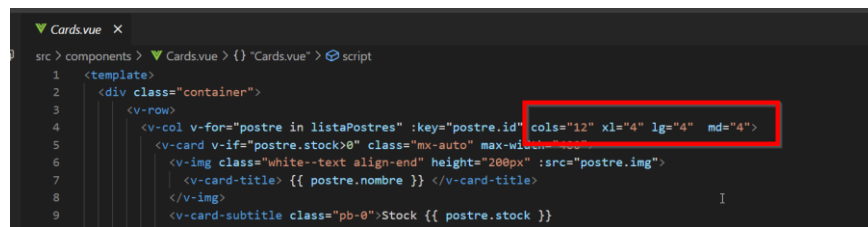
22
23
24   {
25     path: '/postres',
26     name: 'Postres',
27     component: Postres,
28     meta: {
29       login: true
30     }
31   },

```

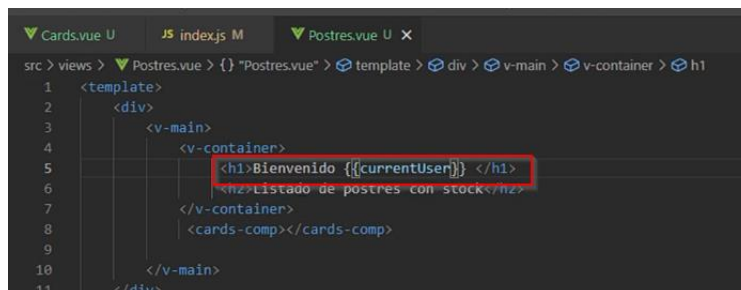
Además, agregaremos una función que se ejecutará cada vez que haya un usuario autenticado; entonces, antes del `export default router`, escribiremos lo siguiente:

```
1 router.beforeEach((to, from, next) => {
2   let user = firebase.auth().currentUser
3   let authRequired = to.matched.some(route => route.meta.login)
4   if (!user && authRequired) {
5     next('/')
6   } else {
7     next()
8   }
9 })
```

En el componente `cards`, agregaremos algunas clases para hacer responsiva las vistas de los postres, en diferentes tamaños de pantalla, y agregaremos en la etiqueta `v-col` `cols="12" lg="4" md="4"`.



Iremos a la vista postres, y eliminaremos `.email` de la variable `currentUser`, solo para que no nos dé error, y poder revisarla para ver el resultado; después, lo volveremos a colocar.



Y levantamos el proyecto con el comando `npm run serve`.

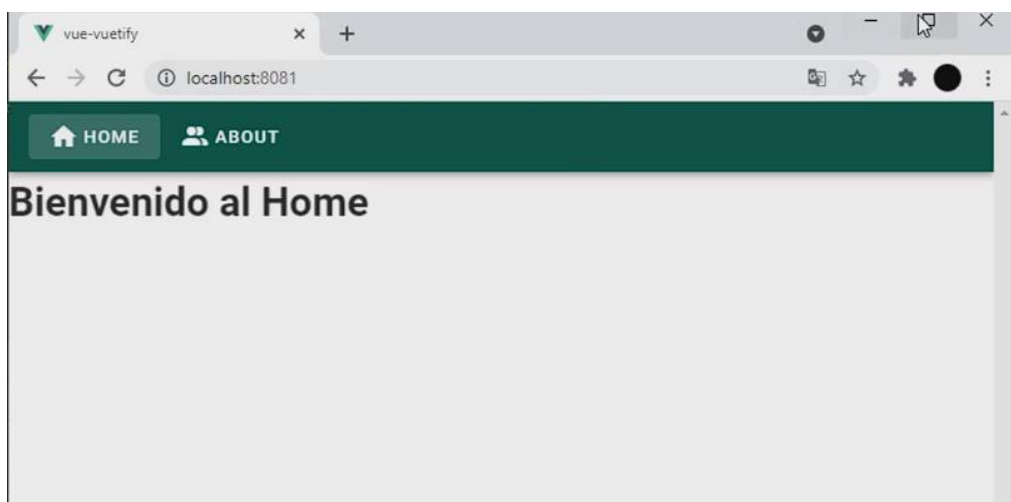
```
EduTecno@DESKTOP-LOEFTOU MINGW64 ~/Documents/Front-end/M5CUE/CUE7/vue-vuetify (master)
$ npm run serve
> vue-vuetify@0.1.0 serve C:\Users\EduTecno\Documents\Front-end\M5CUE\CUE7\vue-vuetify
> vue-cli-service serve
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
DONE Compiled successfully in 16129ms

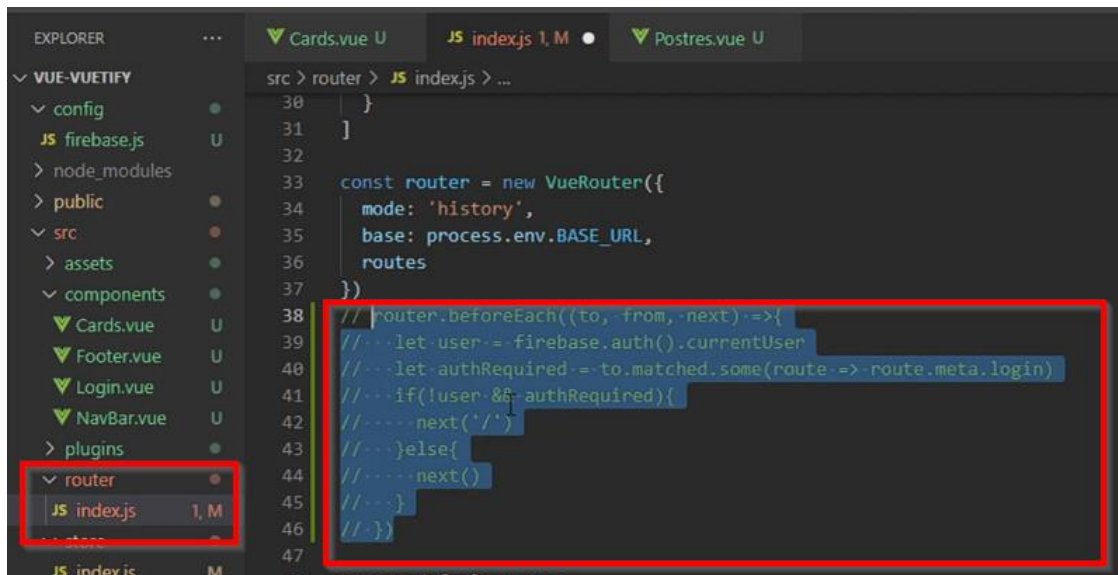
App running at:
- Local:  http://localhost:8081/
  Network: http://192.168.43.210:8081/

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

Abrimos el navegador, y vemos que está funcionando.



Para poder visualizar la vista poste, iremos al archivo `index.js`, y vamos a comentar la función que hicimos.

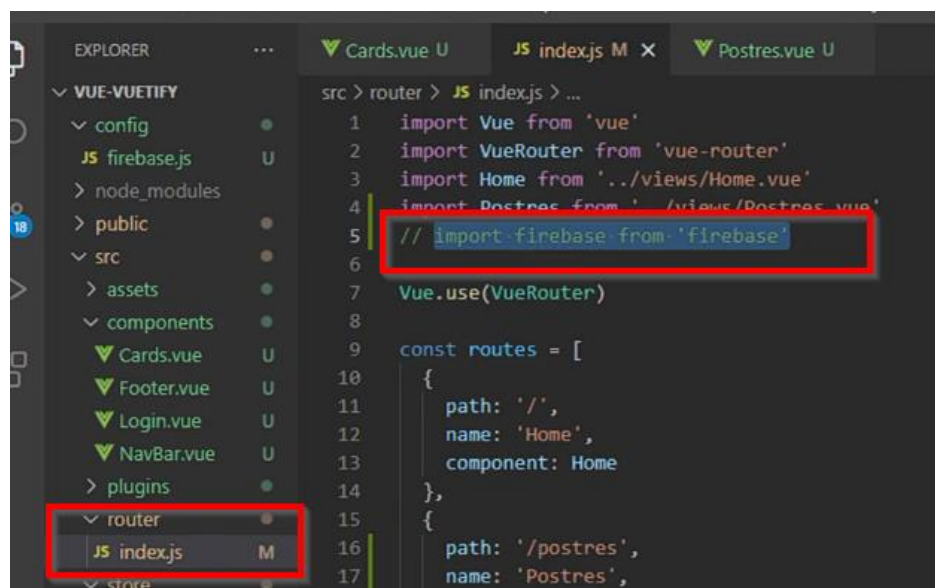


```

EXPLORER
  VUE-VUETIFY
    config
    JS firebase.js
    node_modules
    public
    src
      assets
      components
        Cards.vue
        Footer.vue
        Login.vue
        NavBar.vue
      plugins
        router
          JS index.js
        store
          JS index.js

src > router > JS index.js > ...
30   }
31   ]
32
33   const router = new VueRouter({
34     mode: 'history',
35     base: process.env.BASE_URL,
36     routes
37   })
38
39   // router.beforeEach((to, from, next) => {
40   //   ...let user = firebase.auth().currentUser
41   //   ...let authRequired = to.matched.some(route => route.meta.login)
42   //   ...if (!user && authRequired) {
43   //     ...next('/')
44   //   } else {
45   //     ...next()
46   //   }
47   // })
  
```

También comentamos el `import de firebase`.

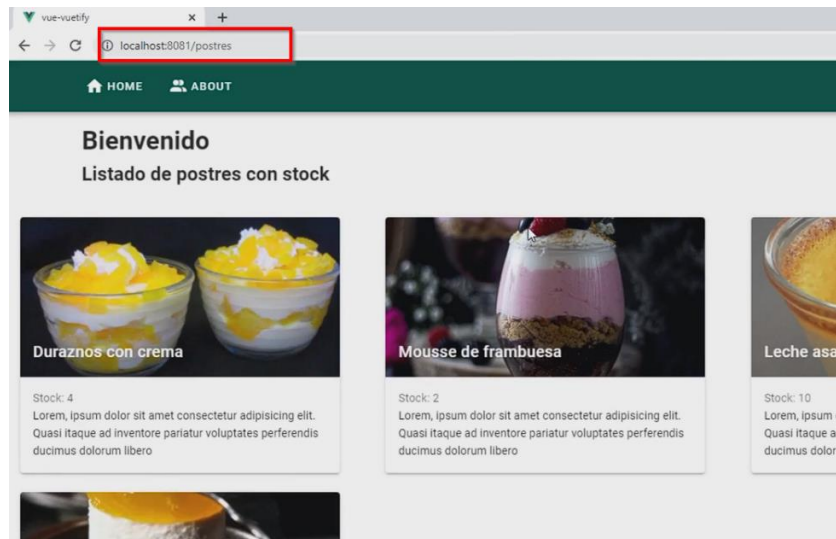


```

EXPLORER
  VUE-VUETIFY
    config
    JS firebase.js
    node_modules
    public
    src
      assets
      components
        Cards.vue
        Footer.vue
        Login.vue
        NavBar.vue
      plugins
        router
          JS index.js
        store
          JS index.js

src > router > JS index.js > ...
1   import Vue from 'vue'
2   import VueRouter from 'vue-router'
3   import Home from '../views/Home.vue'
4   import Postres from '../views/Postres.vue'
5   // import firebase from 'firebase'
6
7   Vue.use(VueRouter)
8
9   const routes = [
10    {
11      path: '/',
12      name: 'Home',
13      component: Home
14    },
15    {
16      path: '/postres',
17      name: 'Postres',
  
```

Volvemos al navegador, y escribimos `/postres`. Como podemos ver, nos está trayendo toda la información, y también permite observar las `cards`.



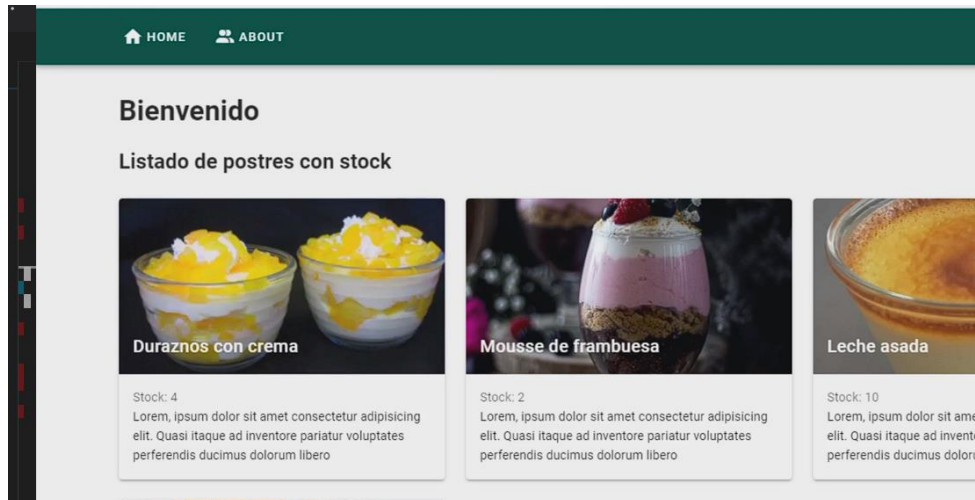
Agregaremos una clase a las etiquetas `h1` y `h2`, y pondremos `class = "mt-4"`, esto quiere decir que tendrá un margen top de 4.

```

1 <template>
2   <div>
3     <v-main>
4       <v-container>
5         <h1 class="mt-4">Bienvenido {{currentUser}} </h1>
6         <h2 class="mt-4">Listado de postres con stock</h2>
7         <cards-comp></cards-comp>
8       </v-container>
9     </v-main>
10  </div>
11 </template>

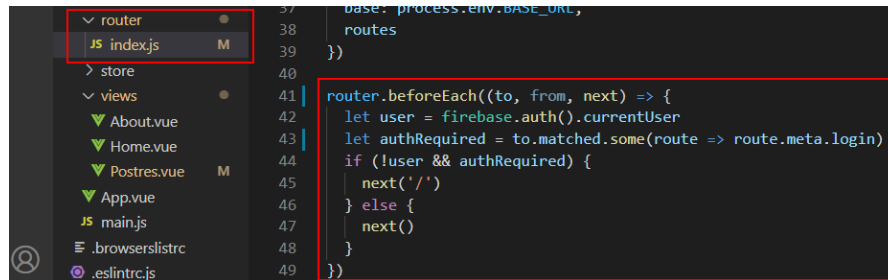
```

Volvemos a revisar en el navegador.



Regresamos a nuestro código al archivo `index.js`, de la carpeta `router`, y descomentamos el `import de Firebase` y nuestra función.

```
JS index.js M X
src > router > JS index.js > [e] routes
1  import Vue from 'vue'
2  import VueRouter from 'vue-router'
3  import Home from '../views/Home.vue'
4  import About from '../views/About.vue'
5  import firebase from 'firebase'
6  import Postres from '../views/Postres.vue'
```

Ahora, crearemos el contenido para el login, donde se pedirá un email y la contraseña, y utilizaremos un **template** que ya está estructurado.

En la primera etiqueta **<v-text-field>**, crearemos un **v-model** que será el que reciba el email; y en la segunda etiqueta **<v-text-field>**, el **v-model** que recibirá la password. La variable **showPassword** es para que no se muestre la contraseña, y la declararemos en la data más adelante. En el botón colocaremos un evento **@click.prevent**, y será igual a login, que es la función que crearemos posteriormente para autenticar al usuario, y el nombre del botón será Login.

```

1 <template>
2   <div>
3     <v-card width="400px" class="mx-auto mt-5" >
4       <v-card-title primary-title>
5         <h1 class="mx-auto">Login</h1>
6       </v-card-title>
7       <v-card-text>
8         <v-form>
9           <v-text-field v-model="email" name="email" label="email"
10 prepend-icon="mdi-account-circle">
11         </v-text-field>
12         <v-text-field v-model="password" @click:append= "showPassword
13 = !showPassword" :type="showPassword ? 'text' : 'password'"
14 label="password" prepend-icon="mdi-lock" :append-
15 icon="showPassword ? 'mdi-eye' : 'mdi-eye-off'">
16         </v-text-field>
17       </v-form>
18     </v-card-text>
19     <v-card-actions>
20       <v-spacer></v-spacer>
21       <v-btn @click.prevent="login" color="info">Login</v-btn>
22     </v-card-actions>
23   </v-card>
24 </div>
25 </template>
  
```

Lo siguiente que tenemos que hacer, es importar `firebase` para la autenticación, y también `mapActions` para utilizar `setUser`.

```
24 <script>
25 import firebase from 'firebase'
26 import { mapActions } from 'vuex'
27 export default {
```

Crearemos la data, donde declararemos `showPassword false`, y las variables de los `v-model` que generamos en el `template` anteriormente.

```
27 export default {
28   data: () => ({
29     showPassword: false,
30     email: '',
31     password: ''
32   })),
```

Agregaremos el objeto `methods`, donde llamaremos el `mapAction` que queremos utilizar, y aquí será donde crearemos la función Login.

```
1  methods: {  
2    ...mapActions(['setUser']),  
3    login() {  
4      firebase.auth().signInWithEmailAndPassword(this.email,  
5 this.password).  
6      then(() => {  
7        this.setUser({email: this.email})  
8        this.$router.push('/postres')  
9      }).catch(() => {  
10       alert('Usuario erroneo')  
11     })  
12   }  
13 },
```

`firebase.auth().signInWithEmailAndPassword(this.email, this.password)` es el método de autenticación que elegimos al momento de crear al usuario en `firebase`, es decir, con un email y una contraseña.

EXERCISE 4: USO PRÁCTICO DE FIREBASE, AUTENTICACIÓN PARTE 3

Continuaremos con la vista `Home.vue`, y le haremos algunas modificaciones. Aquí mostraremos los postres sin stock, y es donde el usuario se deberá loguear correctamente para poder ver la lista de los postres que si están disponibles.

Lo que haremos, será crear una etiqueta v-container dentro de `<div>`. Aquí colocaremos la etiqueta `h1`, que tendrá la clase `mt-5` con el título: "Postres favoritos"; y a continuación, crearemos la etiqueta `h3`, con la clase `mb-5` que contenga la frase: "Temporalmente sin stock".

```
1 <template>
2   <div>
3     <v-container>
4       <h1 class="mt-5">Postres favoritos</h1>
5       <h3 class="mb-5">Temporalmente sin stock</h3>
6     </v-container>
7   </div>
8 </template>
```

A continuación, agregaremos un `v-row` y dentro un `v-col`. En esta etiqueta haremos la iteración de los postres con la directiva `v-for`, y también incluiremos una etiqueta `div`, en la que colocaremos un `v-if` para que nos muestre los postres con stock 0.

```
1 <template>
2   <div>
3     <v-container>
4       <h1 class="mt-5">Postres favoritos</h1>
5       <h3 class="mb-5">Temporalemente sin stock</h3>
6       <v-row>
7         <v-col v-for="postre in listaPostres" :key="postre.id"
8         cols="12" lg="4" md="6">
9           <div v-if="postre.stock==0">
10            </div>
11          </v-col>
12        </v-row>
13      </v-container>
14    </div>
15 </template>
```

Pondremos una etiqueta `v-img`, que tendrá la clase `white--text align-end`, un `height de 200px` y un `width de 100%`, y llamaremos a la imagen del postre con `:src="postre.img"`. Por último, tendremos una etiqueta `v-card-text`, donde pondremos el nombre de los postres con `{{postre.nombre}}`.

```
1 <template>
2   <div>
3     <v-container>
4       <h1 class="mt-5">Postres favoritos</h1>
5       <h3 class="mb-5">Temporalemente sin stock</h3>
6       <v-row>
7         <v-col v-for="postre in listaPostres" :key="postre.id"
8 cols="12" lg="4" md="6">
9         <div v-if="postre.stock==0">
10          <v-img class="white--text align-end" height="200px"
11 width="100%"
12          :src="postre.img">
13          <v-card-text> {{ postre.nombre }} </v-card-text>
14        </v-img>
15      </div>
16    </v-col>
17  </v-row>
18 </v-container>
19 </div>
20 </template>
```

Ahora crearemos otra etiqueta `v-row`, y una `v-col` dentro. Aquí colocaremos un `h2`, que tendrá una clase `text-center` y dirá: "Para ver nuestros postres disponibles, inicia sesión". Importaremos el componente Login, y también el `mapstate`; crearemos el objeto `components`, donde incluiremos a Login; en el objeto `computed`, invocaremos la variable `listaPostres`; y en el `template`, debajo del `h2`, llamaremos al Login como etiqueta.

```
1 <template>
2   <div>
3     <v-container>
4       <h1 class="mt-5">Postres favoritos</h1>
5       <h3 class="mb-5">Temporalmente sin stock</h3>
6       <v-row>
7         <v-col v-for="postre in listaPostres" :key="postre.id"
8 cols="12" lg="4" md="6">
9           <div v-if="postre.stock==0">
10             <v-img class="white--text align-end"
11 height="200px" width="100%" :src="postre.img">
12               <v-card-text> {{postre.nombre }}</v-card-text>
13             </v-img>
14           </div>
15         </v-col>
16       </v-row>
17       <v-row>
18         <v-col>
19           <h2 class="text-center">Para ver nuestros postres
20 disponibles, inicia sesión</h2>
21           <login></login>
22         </v-col>
23       </v-row>
24     </v-container>
25   </div>
26 </template>
27
28 <script>
29 import Login from "@/components/Login.vue";
30 import {mapState} from "vuex";
31 export default {
32   name: "Home",
33   components: {
34     Login,
35   },
36   computed: {
37     ...mapState(["listaPostres"]);
38   }
39 </script>
```

Iremos a nuestro componente **NavBar.vue**, y reemplazaremos el botón about (no lo utilizaremos en este ejemplo), por el botón para los postres.

```
1 <template>
2   <div>
3     <v-app-bar color="teal darken-4" dark>
4       <v-container>
5         <v-row>
6           <v-btn text to="/"><v-icon class="mr-1">mdi-home</v-
7 icon>Home</v-btn>
8           <v-btn text to="/postres"><v-icon class="mr-1">mdi-cake</v-
9 icon>Postres</v-btn>
10        </v-row>
11      </v-container>
12    </v-app-bar>
13  </div>
14</template>
```

Crearemos un botón para salir, una vez autenticado y visto los postres con stock, escribiremos la etiqueta **<v-btn>**, colocaremos **v-if="currentUser"** que será de tipo text, y le agregaremos el evento **@click.prevent**, que estará ligado a una función llamada "logout", y como nombre tendrá "Salir". Le pondremos un ícono con **v-icon** y una clase **"mr-1"**.

```
1 <template>
2   <div>
3     <v-app-bar color="teal darken-4" dark>
4       <v-container>
5         <v-row>
6           <v-btn text to="/"><v-icon class="mr-1">mdi-home</v-
7 icon>Home</v-btn>
8           <v-btn text to="/postres">
9             <v-icon class="mr-1">mdi-cake</v-icon>Postres
10          </v-btn>
11          <v-btn v-if="currentUser" text
12 @click.prevent="logout">
13            <v-icon class="mr-1">mdi-logout</v-icon>Salir
14          </v-btn>
15        </v-row>
16      </v-container>
17    </v-app-bar>
18  </div>
19</template>
```

Aquí le estamos indicando que si el usuario, en este caso `currentUser`, es el correcto, tendrá la opción de visualizar el botón para salir de su sesión.

Al botón que nos dirige al Home, le agregaremos la directiva `v-if`, y le indicaremos que será distinto de `currentUser`, esto quiere decir que, si es diferente al usuario logueado puede ingresar al Home, si no, puede ingresar a la vista postres y visualizar los que tienen stock.

```

▼ NavBar.vue M X
src > components > ▼ NavBar.vue > {} "NavBar.vue" > script > default
1  <template>
2    <div>
3      <v-app-bar color="teal darken-4" dark>
4        <v-container>
5          <v-row>
6            <v-btn v-if="!currentUser" text to="/"><v-icon class="mr-1">mdi-home</v-icon>Home</v-btn>
7            <v-btn text to="/postres">
8              <v-icon class="mr-1">mdi-cake</v-icon>Postres
9            </v-btn>

```

En la etiqueta script, importaremos: `Firebase`, `mapAction` y `mapState`, crearemos el objeto `methods` y `computed`; en el primero, invocamos a `setUser`, y aquí generaremos la función `logout` para cerrar sesión, y que al hacerlo nos dirija a la vista home; y en el segundo, invocaremos `currentUser`.

```

1  import firebase from 'firebase'
2  import { mapActions, mapState } from 'vuex'
3  export default {
4    data() {
5      return {};
6    },
7    methods: {
8      ...mapActions(['setUser']),
9
10     logout() {
11       firebase.auth().signOut().then(() => {
12         this.$router.push('/')
13         this.setUser(undefined)
14       })
15     },
16   },
17   computed: {
18     ...mapState(['currentUser'])
19   }
20 };

```

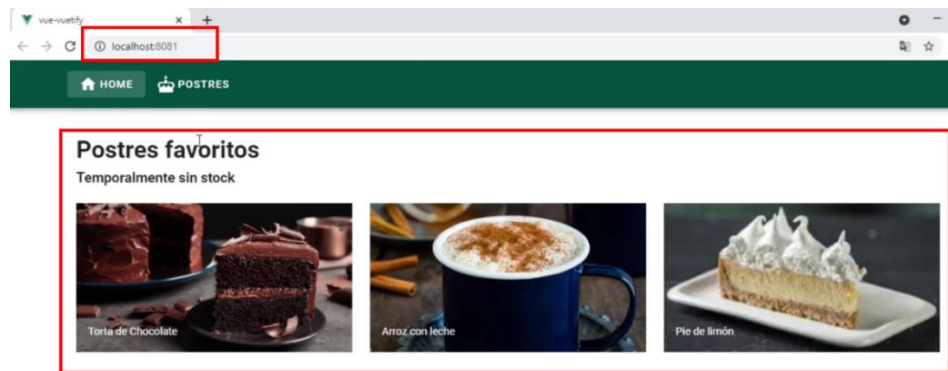

Levantamos nuestro proyecto con `npm run serve`, y visualizamos en el navegador.

```
Edutecno@DESKTOP-LQFETQJ MINGW64 ~/Documents/Front-end/M5CUE/CUE7/vue-vuetify (master)
$ npm run serve

> vue-vuetify@0.1.0 serve C:\Users\Edutecno\Documents\Front-end\M5CUE\CUE7\vue-vuetify
> vue-cli-service serve

INFO Starting development server...
```

Podemos ver en el Home, las imágenes de los postres sin stock.



Si presionamos el botón “Postres”, notaremos que no se puede ingresar, así que iniciaremos sesión, escribiremos el email que registramos en **Firestore**, y la contraseña; pondremos usuario@postres.cl y como contraseña 123456, y presionamos el botón Login.

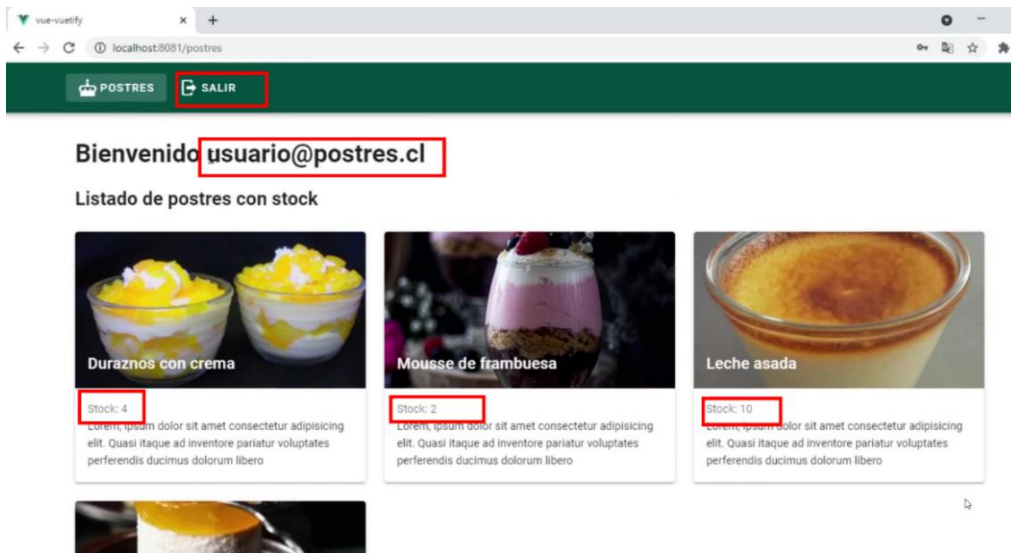
Para ver nuestros postres disponibles, inicia sesión

Login

email

password

Ahora que iniciamos sesión, nos dice "Bienvenido -el mail que registramos-", y nos muestra los postres con stock, así como el botón Salir.

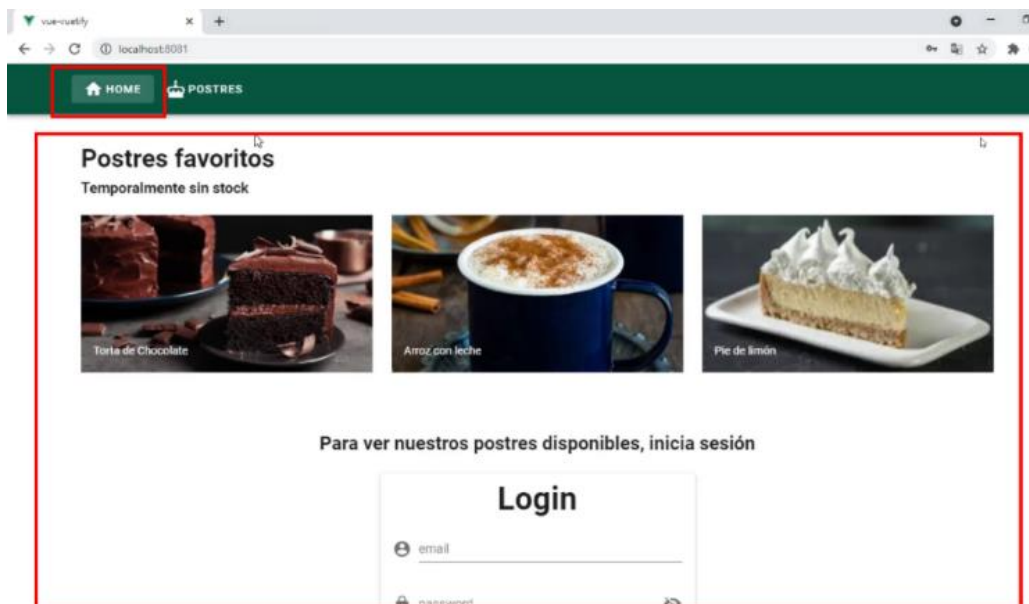


Si presionamos el botón Salir, nuestra sesión se cierra.



Y nos redirige al home.

De esta manera, ya podemos hacer una autenticación con **Firestore**.

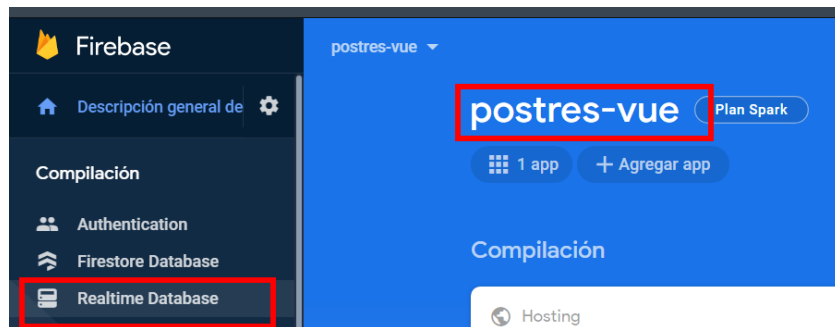


EXERCISE 5: CONFIGURACIÓN REALTIME DATABASE

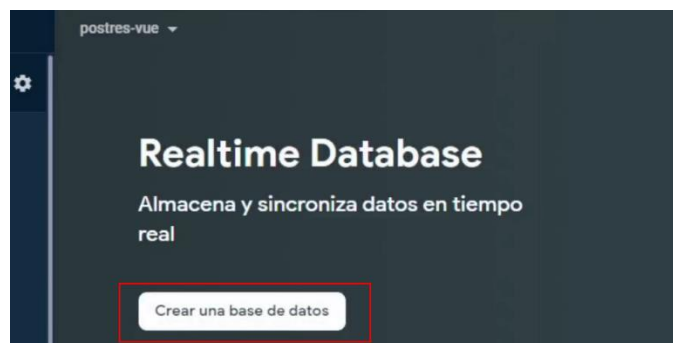
En el siguiente ejercicio, utilizaremos **Realtime Database**, una base de datos alojada en la nube de **Firebase**. Los datos se almacenan en formato **JSON**, y se sincronizan en tiempo real con cada cliente conectado, recibiendo las actualizaciones automáticamente con los datos más recientes.

En esta base de datos, guardaremos toda la información de los postres (nombre, stock y la imagen), y el id no lo guardaremos porque se crearán automáticamente. De esta manera, no será necesario tener un arreglo en el state.

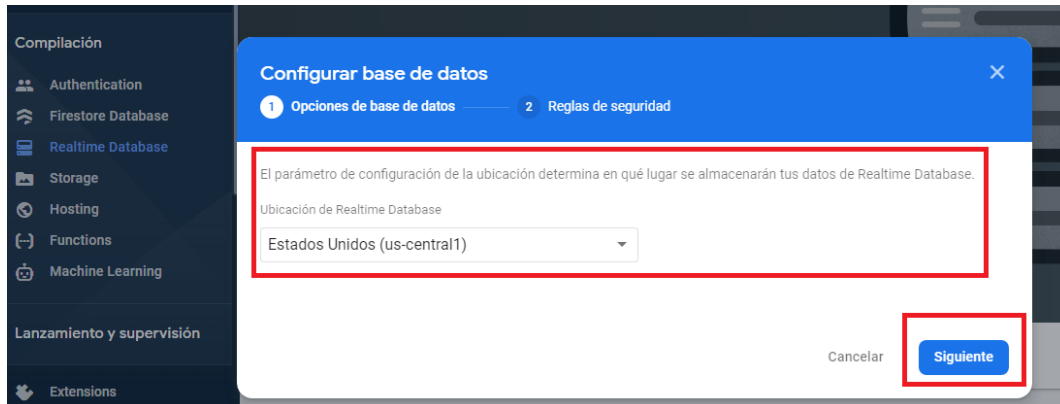
Abriremos el proyecto que creamos en **Firebase** "postres-vue" del Exercise anterior, y nos iremos a la pestaña de **Realtime Database**.



Nos aparecerá la opción de crear una base de datos, y le damos clic.



Dejamos la ubicación que viene por defecto, y presionamos Siguiente.



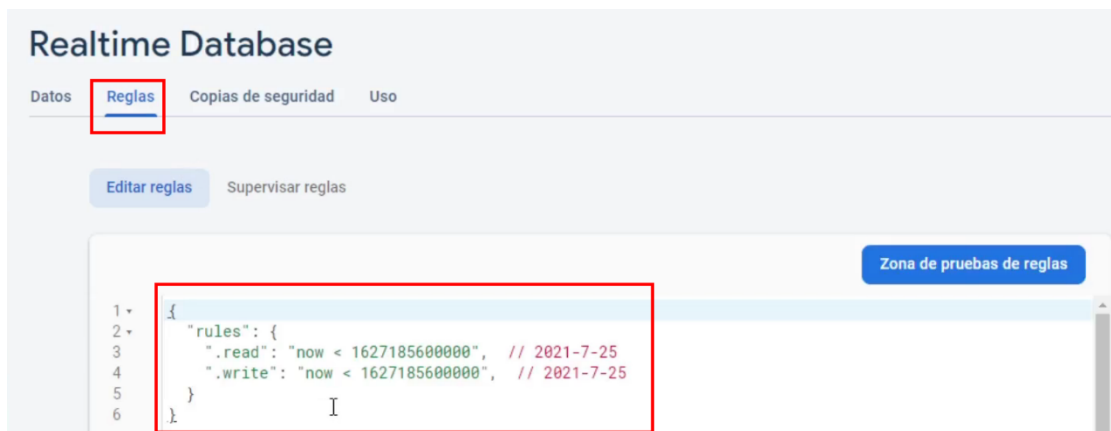
En esta pantalla, “reglas de seguridad”, pondremos la opción: “Comenzar en modo de prueba”, y presionamos Habilitar.



De esta manera, nos creará automáticamente una **URL**.



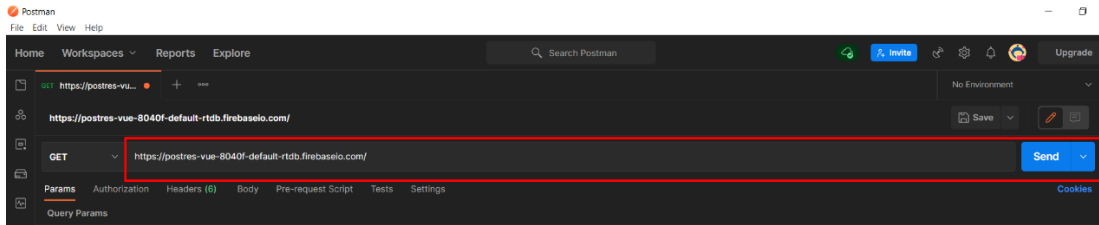
Ésta ya se puede utilizar como **API**, a la que cualquier persona puede acceder, ya que no hemos creado reglas de seguridad, y para este ejercicio de ejemplificación no será necesario, por lo que dejaremos las que vienen por defecto.



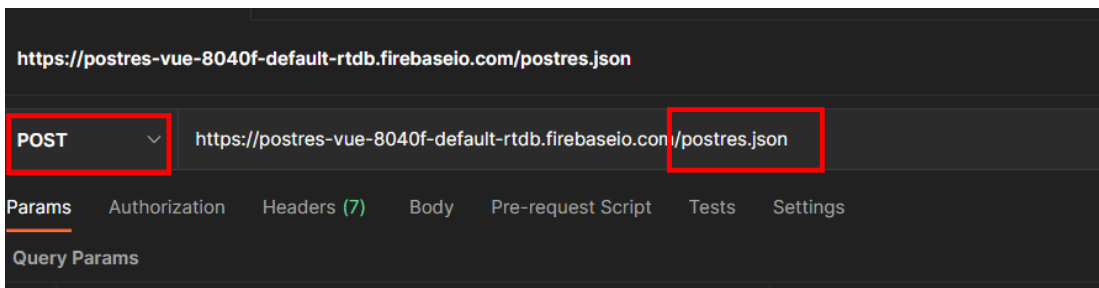
Ahora necesitamos crear los postres, y guardarlos en nuestra base de datos con toda su información.

EXERCISE 6: USO PRÁCTICO REALTIME DATABASE

Para esto, copiaremos la **URL**, nos iremos al programa **Postman**, y la pegamos.

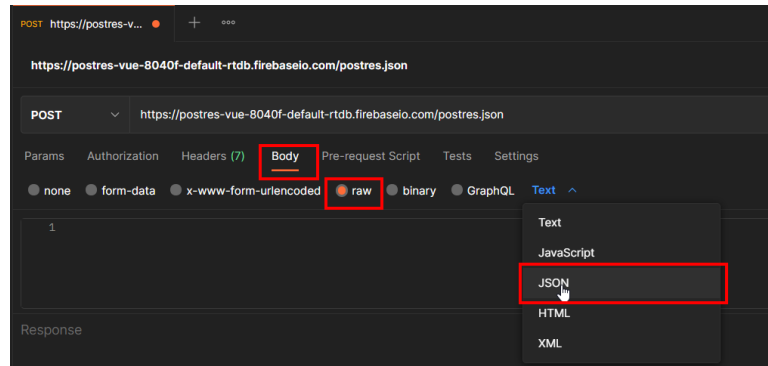


Aquí ya podemos agregar los postres, pero, ¿cómo lo haremos?: con la petición **post**, y además, aquí hay una regla importante; nosotros tenemos que crear un **Array** que va a almacenar todos los postres en un repositorio o colección, por lo que a nuestra **URL** le agregaremos el nombre de ese repositorio que los contendrá, y se llamará: **postres.json**.

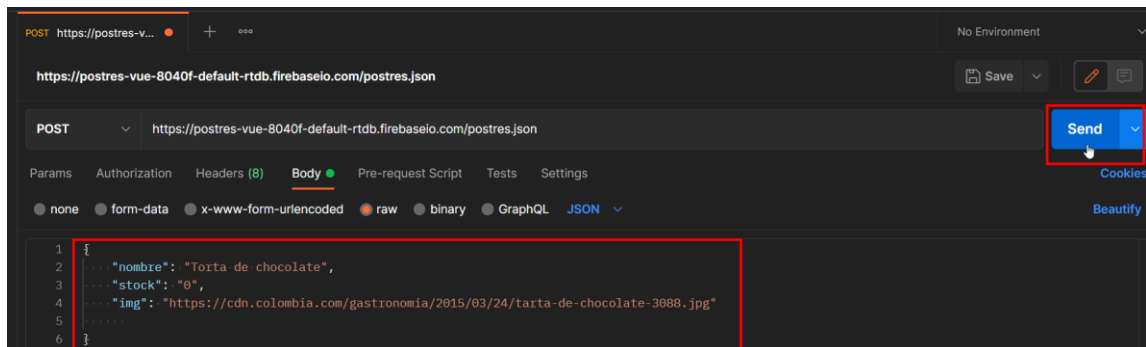


Recordemos que **Realtime Database** trabaja con **JSON**.

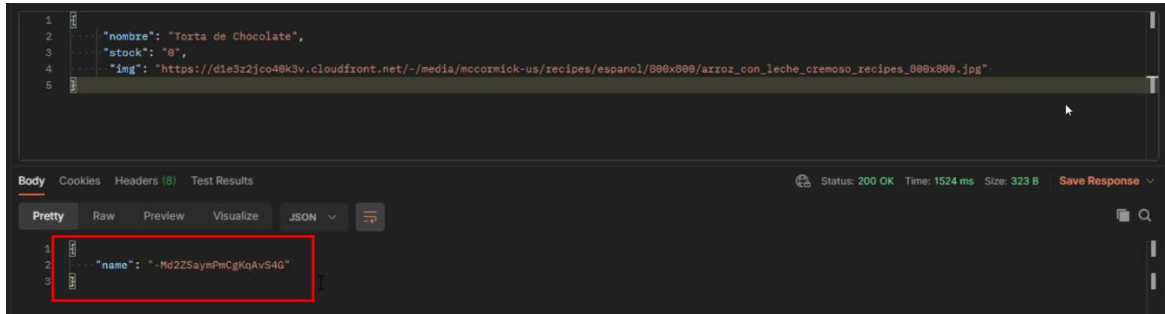
Lo siguiente que haremos es ir a la pestaña **body**, cliqueamos la opción **raw**, y colocamos **JSON**.



Y entre corchetes {}, colocaremos toda la información de los postres entre comillas; una vez hecho esto, presionamos Send.



Y podemos ver una respuesta en donde aparece un **ID**.

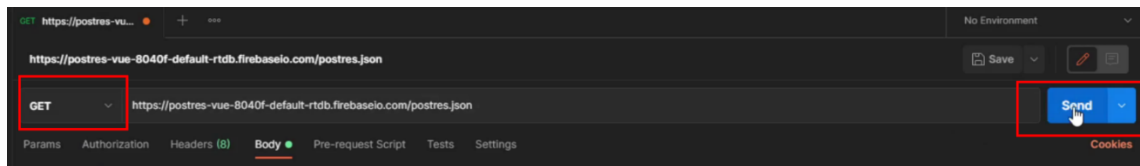


Si volvemos para revisar nuestra base de datos, notaremos que tenemos creado nuestro primer postre, con un **id** generado automáticamente, y con sus datos.



Haremos esto con cada uno de los postres que tenemos en nuestro **state**: arroz con leche, duraznos con crema, mousse de frambuesa, leche asada, y por último, cheesecake de maracuyá.

Ahora comprobaremos que tenemos acceso a los postres, cambiaremos el tipo de petición a **GET**, y presionamos Send.



Y finalmente podemos ver el listado de todos los postres que agregamos.



Revisamos nuestra base de datos en **Firebase**, y también podemos ver que están nuestros postres agregados.



```

-Mf-gaorUBsD6k-7yR36
  img: "https://i.ytimg.com/vi/-q4eu_ieQZ0/sddefault.jpg"
  nombre: "Duraznos con crema"
  stock: "6"
-Mf-gdGc7yfDhCmAiQ49
  img: "https://3.bp.blogspot.com/-90EvVPzqKk/WVsua204..."
  nombre: "Mousse de frambuesa"
  stock: "8"
-Mf-gfyZ2JbwcgN61e_o
  img: "https://img-global.cpcdn.com/recipes/2521df3c8b..."
  nombre: "Leche asada"
  stock: "15"
-Mf-giEVf4A42Eo25yTu
  img: "https://mui.kitchen/_export/1622926268148/site..."
  nombre: "Cheesecake maracuya"
  
```

Una vez ingresados todos nuestros postres, abrimos nuestro proyecto en VSC, iremos al archivo **index.js** de la carpeta store, y eliminaremos los datos de nuestro arreglo, dejándolo vacío.

```

EXPLORER
  VUE-VUETIFY
    > config
    > node_modules
    > public
    > src
      > assets
      > components
      > plugins
      > router
      > store
        JS index.js M
  > views

JS index.js M X
src > store > JS index.js > [🔗] default
1  import Vue from 'vue'
2  import Vuex from 'vuex'
3
4
5  Vue.use(Vuex)
6
7  export default new Vuex.Store({
8    state: {
9      currentUser: undefined,
10     listaPostres: [],
11   },
12   mutations: {
13
14   }
15 })
  
```

Lo que necesitamos ahora es obtener nuestros postres desde la base de datos, para eso, emplearemos **Axios**, la librería que nos permitirá hacer las peticiones. Abrimos la terminal de VSC para instalarlo en nuestro proyecto, y escribiremos el comando: **npm install axios**.

```
s/ejercicio/VueVuetify/vue-vuetify (main)
$ npm install axios
loadExtraneous: sill removeObsoleteDep removing axios@0.21.1 from the tree as its been replaced by a newer version
```

Ahora que ya se instaló, lo importaremos, escribiendo: **import axios from 'axios'**.

```
JS index.js  X
src > store > JS index.js > [⌘] default
1  import Vue from 'vue'
2  import Vuex from 'vuex'
3  import Axios from 'axios'
4
```

Crearemos una nueva **mutation**, que se llamará: **"SET_POSTRES"**.

```
mutations: {
  SET_USER(state, user) {
    state.currentUser = user
  },
  SET_POSTRES(state, postres){
    state.listaPostres= postres
  },
}
```

También crearemos la **actions** que utilizará esta **mutation setPostres**, y empleando **Axios** con la función **GET** para obtener nuestros postres, escribiremos entre comillas la **URL** con la colección que creamos.

```
1  actions: {
2    setUser ({commit}, user) {
3      commit ('SET_USER', user)
4    },
5    setPostres ({commit}) {
6      Axios.get('https://vue-postres-default-
7 rtdb.firebaseio.com/postres.json/').
8        then((response)=> {
9          commit ('SET_POSTRES', response.data)
10         })
11     },
12 }
```

Haremos unas pequeñas modificaciones en el componente **cards**, importaremos en el **script** el **mapActions**, crearemos el objeto **methods**, para llamar la **actions setPostres**, y por último, haremos una función **created**, que se ejecutará cuando el componente esté listo.

```
1  import { mapActions, mapState } from "vuex";
2
3  export default {
4    name: "cards-component",
5
6    computed: {
7      ...mapState(['listaPostres']),
8    },
9    methods: {
10      ...mapActions(['setPostres'])
11    },
12
13    created() {
14      this.setPostres()
15    }
16  };
```

Ahora nos iremos a la vista Home, entre las etiquetas `script`, importaremos `mapActions`, lo llamaremos en el objeto `methods`, y crearemos la función `created()`.

```
1 import Login from "@/components/Login.vue";
2
3 import { mapActions, mapState } from "vuex";
4
5 export default {
6   name: "Home",
7
8   components: {
9     Login,
10  },
11  computed: {
12    ...mapState(["listaPostres"]),
13  },
14  methods: {
15    ...mapActions(["setPostres"]),
16  },
17
18  created() {
19    this.setPostres();
20  },
21 };
```

Y en el `v-if` del `template` lo dejaremos con doble = que es más flexible. Cuando se pone con triple =, se le está pidiendo al sistema que vea a los que cumplen al 100 con la condición.

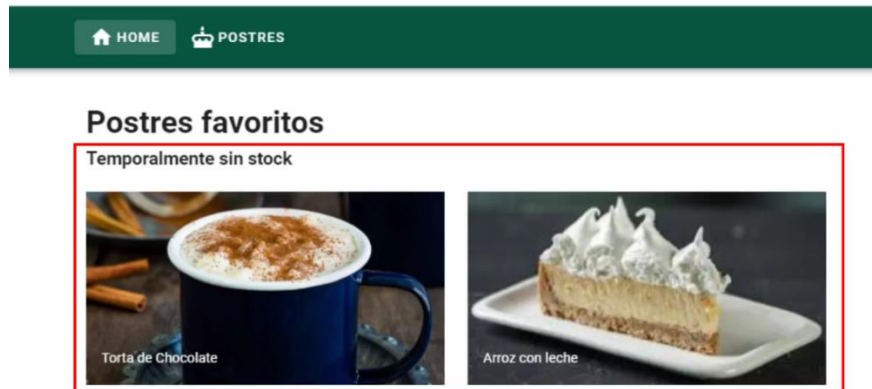


```
Home.vue M X
src > views > Home.vue > {} "Home.vue" > script
1 <template>
2 <div>
3   <v-container>
4     <h1 class="mt-5">Postres favoritos</h1>
5     <h3 class="mb-5">Temporalemente sin stock</h3>
6   <v-row>
7     <v-col v-for="postre in listaPostres" :key="postre.id" cols="12" lg="4" md="6">
8       <div v-if="postre.stock==0">
```

Guardamos todos los cambios, y levantamos nuestro proyecto con `npm run serve`.

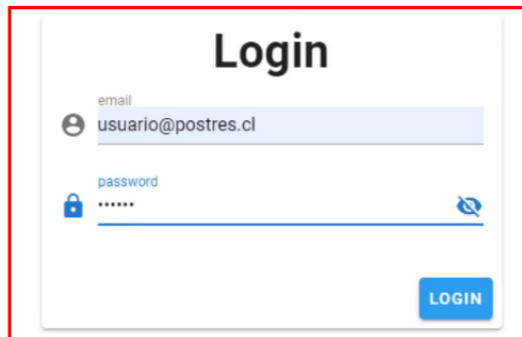
```
$ npm run serve  
vue-vuetify@0.1.0 serve C  
vue-cli-service serve
```

Revisamos el navegador, y vemos que nos muestra los postres sin stock.



Luego, iniciamos sesión.

Para ver nuestros postres disponibles, inicia sesión



Login

email
usuario@postres.cl

password

LOGIN

Y veremos los postres con stock, todo esto con la información ingresada a la base de datos de **Firestore**.

