

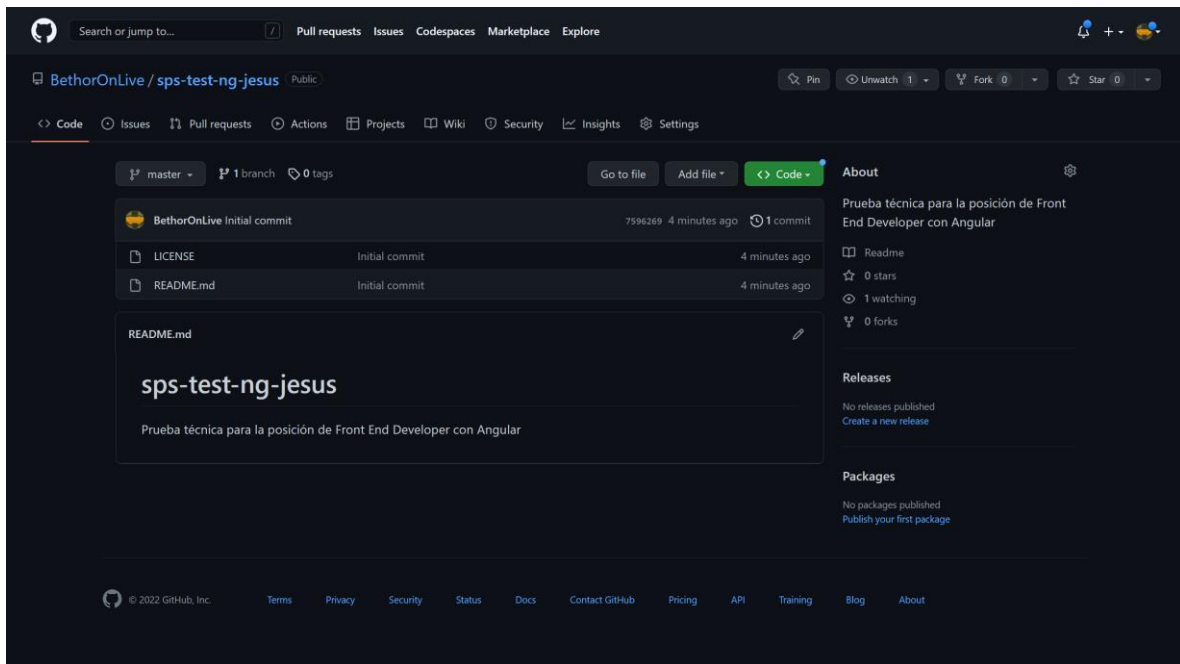


# Front End Developer Test

Angular

12/12/2022

Se crea un repositorio público en GitHub para subir el proyecto.



Se clona el repositorio de GitHub a local.

```
computadora@DESKTOP-8H2VSGO MINGW64 ~  
$ git clone https://github.com/BethorOnLive/sps-test-ng-jesus.git  
Cloning into 'sps-test-ng-jesus'..  
remote: Enumerating objects: 4, done.  
remote: Counting objects: 100% (4/4), done.  
remote: Compressing objects: 100% (4/4), done.  
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0  
Receiving objects: 100% (4/4), done.
```

Se crea un nuevo proyecto Angular.

```
MINGW64/c:/Users/computadora/sps-test-ng-jesus  
computadora@DESKTOP-8H2VSGO MINGW64 ~/sps-test-ng-jesus (master)  
$ ng new shopee-store  
CREATE shopee-store/angular.json (2952 bytes)  
CREATE shopee-store/package.json (1043 bytes)  
CREATE shopee-store/README.md (1065 bytes)  
CREATE shopee-store/tsconfig.json (863 bytes)  
CREATE shopee-store/.editorconfig (274 bytes)  
CREATE shopee-store/.gitignore (548 bytes)  
CREATE shopee-store/.browserslistrc (600 bytes)  
CREATE shopee-store/karma.conf.js (1429 bytes)
```

Se hace el primer commit.

```
computadora@DESKTOP-8H2VSGO MINGW64 ~/sps-test-ng-jesus/shopee-store (master)
$ git commit -am "Initial commit"
[master c6fca50] Initial commit
28 files changed, 21359 insertions(+)
create mode 100644 shopee-store/.browserslistrc
create mode 100644 shopee-store/.editorconfig
create mode 100644 shopee-store/.gitignore
create mode 100644 shopee-store/.vscode/extensions.json
create mode 100644 shopee-store/.vscode/launch.json
```

```
computadora@DESKTOP-8H2VSGO MINGW64 ~/sps-test-ng-jesus/shopee-store (master)
$ git pull origin master
From https://github.com/BethorOnLive/sps-test-ng-jesus
* branch      master      -> FETCH_HEAD
Already up to date.

computadora@DESKTOP-8H2VSGO MINGW64 ~/sps-test-ng-jesus/shopee-store (master)
$ git push origin master
Enumerating objects: 36, done.
Counting objects: 100% (36/36), done.
```

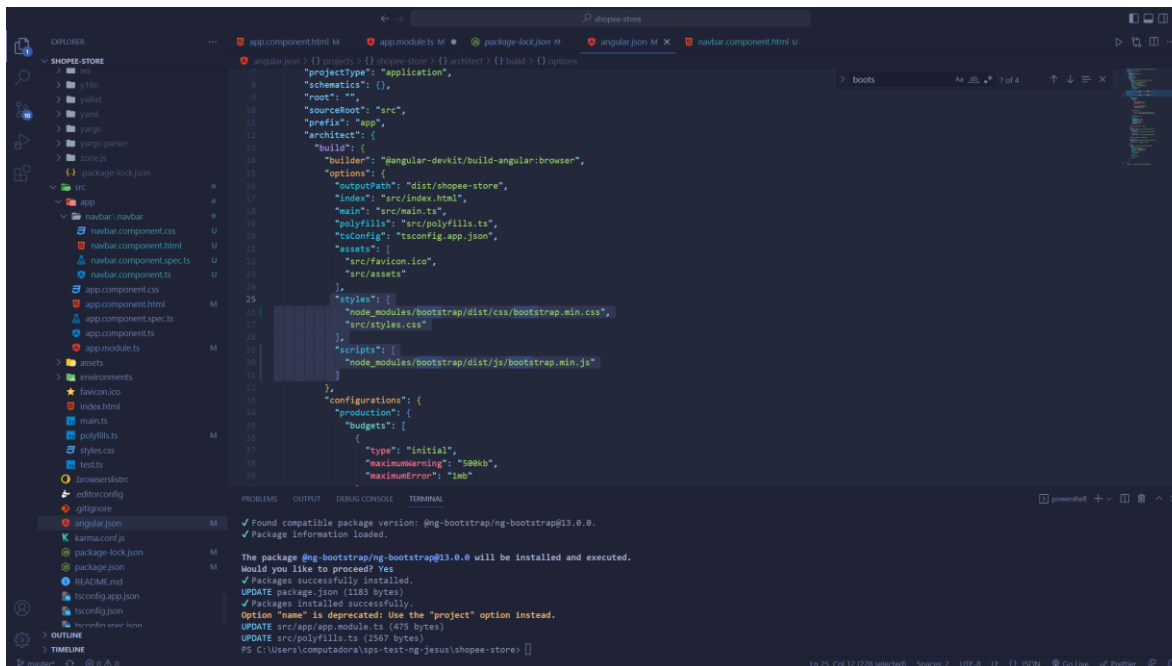
Se corre el proyecto en local host.

The screenshot displays the Angular CLI web interface, which provides resources and next steps for a new project. A terminal window is overlaid on the interface, showing the command `ng serve --o` being executed. The terminal output indicates that the browser application bundle generation is complete and provides a table of initial chunk files.

Initial Chunk Files	Names	Raw Size
vendor.js	vendor	1.77 MB
polyfills.js	polyfills	328.06 kB
styles.css, styles.js	styles	210.09 kB
main.js	main	48.05 kB
runtime.js	runtime	6.52 kB

The terminal also shows the initial total size of 2.34 MB and the build time of 12663ms. It concludes with the message "Compiled successfully."

Se instala ng-bootstrap para los componentes UI.



<https://ng-bootstrap.github.io/#/getting-started>

Se cargan algunas fuentes para darle estilo a la tipografía.



<https://fonts.google.com/specimen/Basic?query=basic>

Se instala Font Awesome para usar iconos.

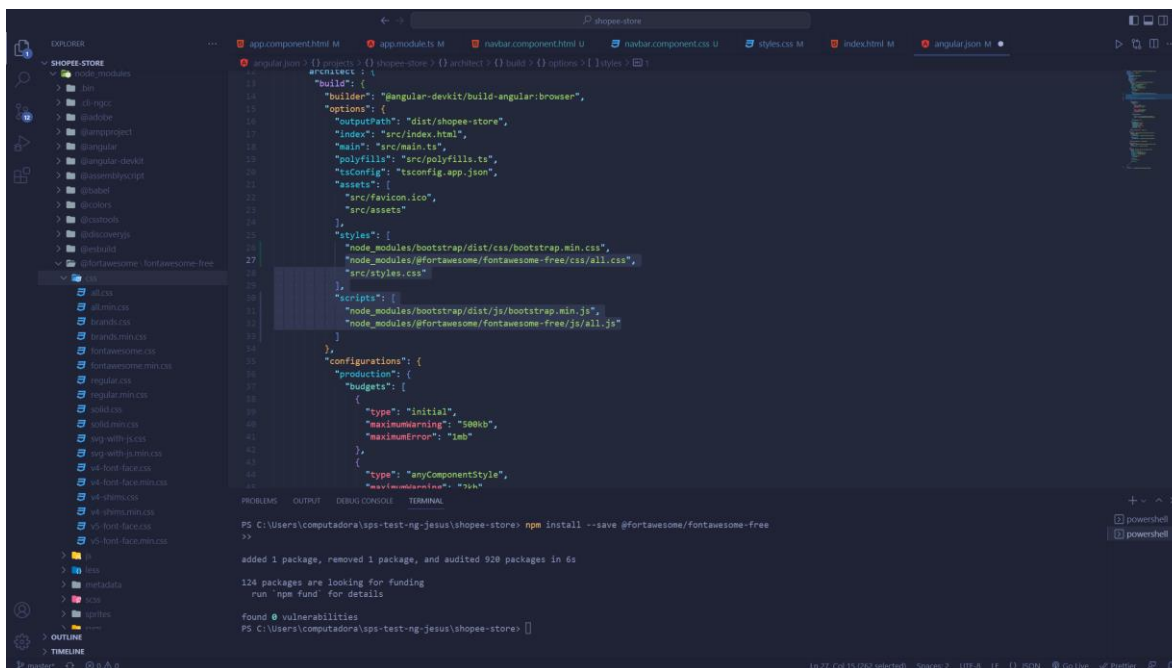
## The Free Package

If you need just the Free icons, we have a slimmed-down free-only package for that. Install with this command:

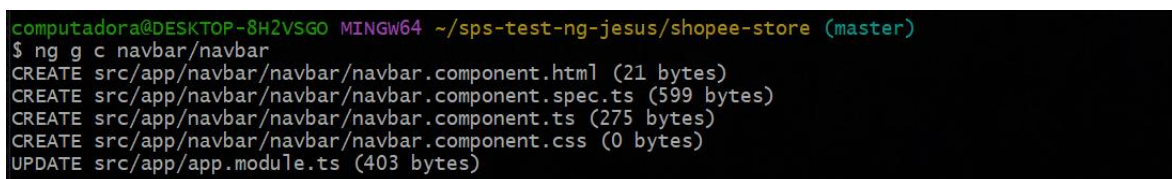
```
npm yarn
```

```
npm install --save @fortawesome/fontawesome-free
```

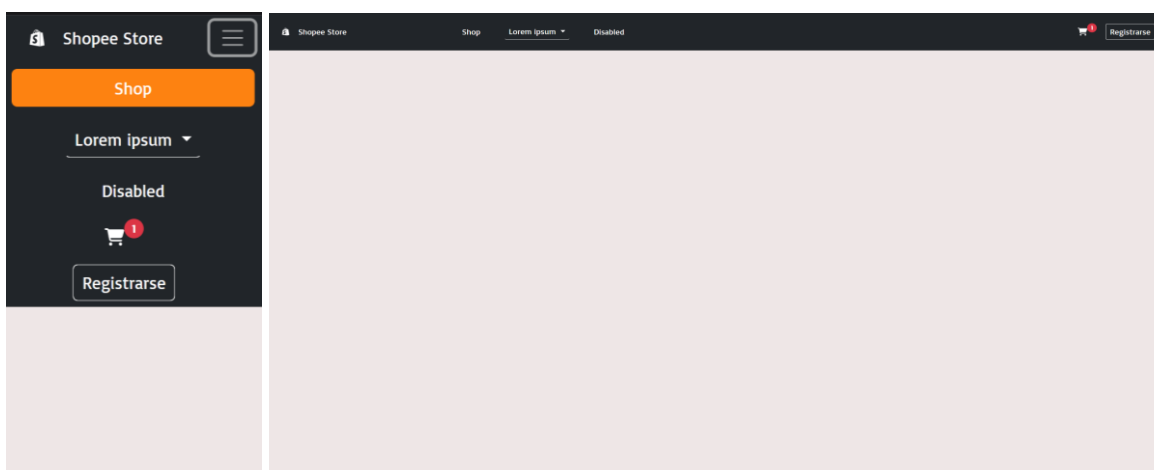




Se crea una carpeta y un componente para el navbar.



Y se agrega una navbar de los componentes de Bootstrap y posteriormente se personaliza con estilos CSS propios y se optimiza el responsive para visualizar en dispositivos móviles.



Se agrega un nuevo componente llamado “register” el cual sustituye el botón “Registrarse” en el navbar.

```
computadora@DESKTOP-8H2VSGO MINGW64 ~/sps-test-ng-jesus/shopee-store (master)
$ ng g c register
CREATE src/app/register/register.component.html (23 bytes)
CREATE src/app/register/register.component.spec.ts (613 bytes)
CREATE src/app/register/register.component.ts (283 bytes)
CREATE src/app/register/register.component.css (0 bytes)
UPDATE src/app/app.module.ts (564 bytes)
```

Al dar click en el botón se abre un modal tamaño lg con la ayuda del componente de bootstrap para modales, el cual se importa en el componente donde se va a utilizar:

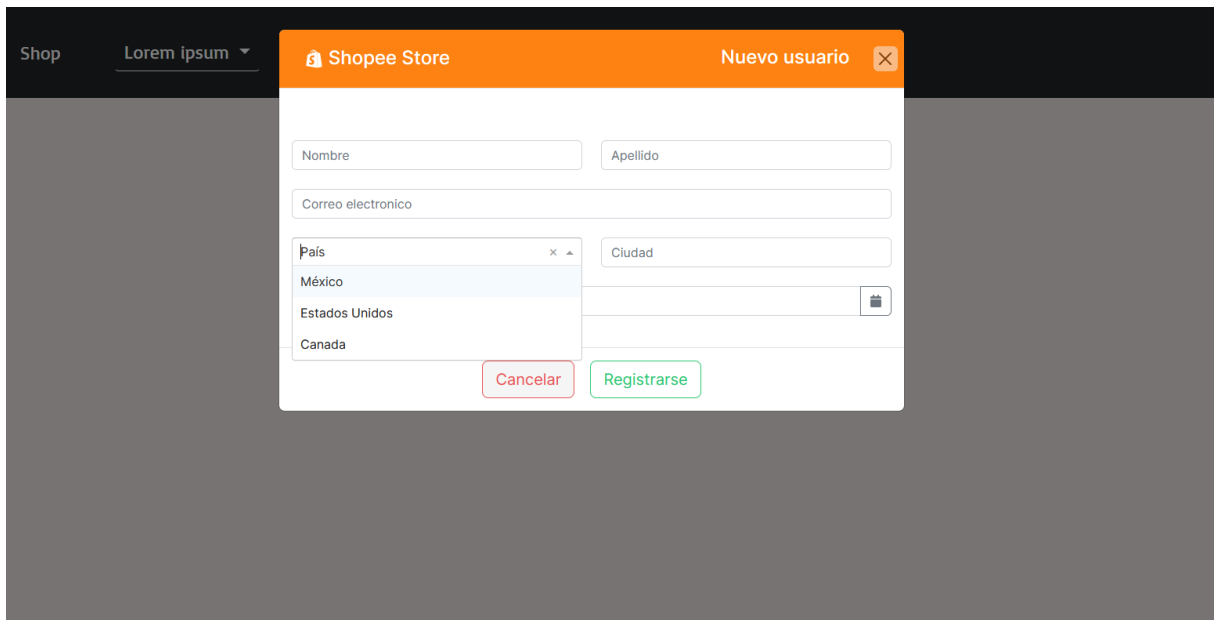
```
1 import { NgbModal } from '@ng-bootstrap/ng-bootstrap';
```

El objetivo del formulario es simular el registro de los datos de un usuario nuevo, haciendo uso de **formularios reactivos**, los campos se validan y devuelven un mensaje de error en caso de no cumplir con los requerimientos.

The screenshot shows a web application interface with a dark sidebar on the left containing a 'Lorem ipsum' dropdown. The main content area features a modal window titled 'Shopee Store' with a sub-header 'Nuevo usuario' and a close button. The modal contains a registration form with the following fields: 'Nombre' (Name), 'Apellido' (Last Name), 'Correo electronico' (Email), 'Pais' (Country), 'Ciudad' (City), and a date field labeled 'yyyy-mm-dd' with a calendar icon. At the bottom of the modal are two buttons: 'Cancelar' (Cancel) and 'Registrarse' (Register).

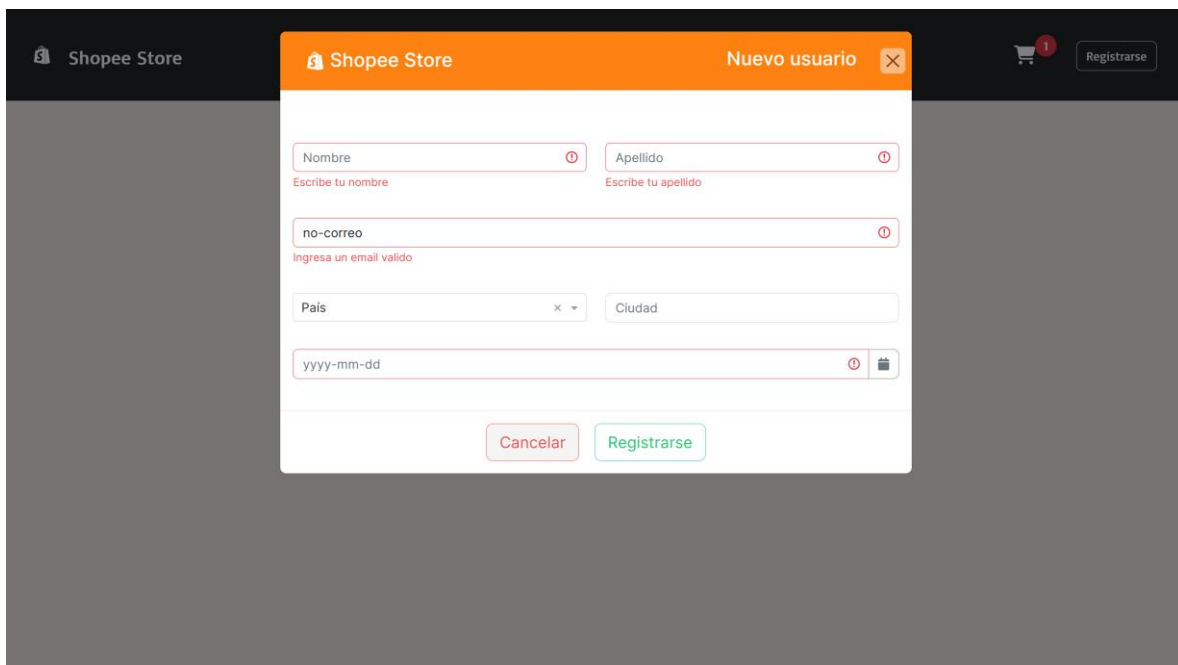
Se agregó un ng-select en campo países.

<https://github.com/ng-select/ng-select>



The screenshot shows a modal window titled 'Nuevo usuario' for the 'Shopee Store'. It contains a registration form with the following fields: 'Nombre', 'Apellido', 'Correo electronico', 'País' (a dropdown menu with 'México', 'Estados Unidos', and 'Canada' selected), and 'Ciudad'. There are also 'Cancelar' and 'Registrarse' buttons at the bottom of the form.

Después se personalizan estilos para los mensajes de validación del formulario que son disparados a través de una condicional **ngClass y class.active**.

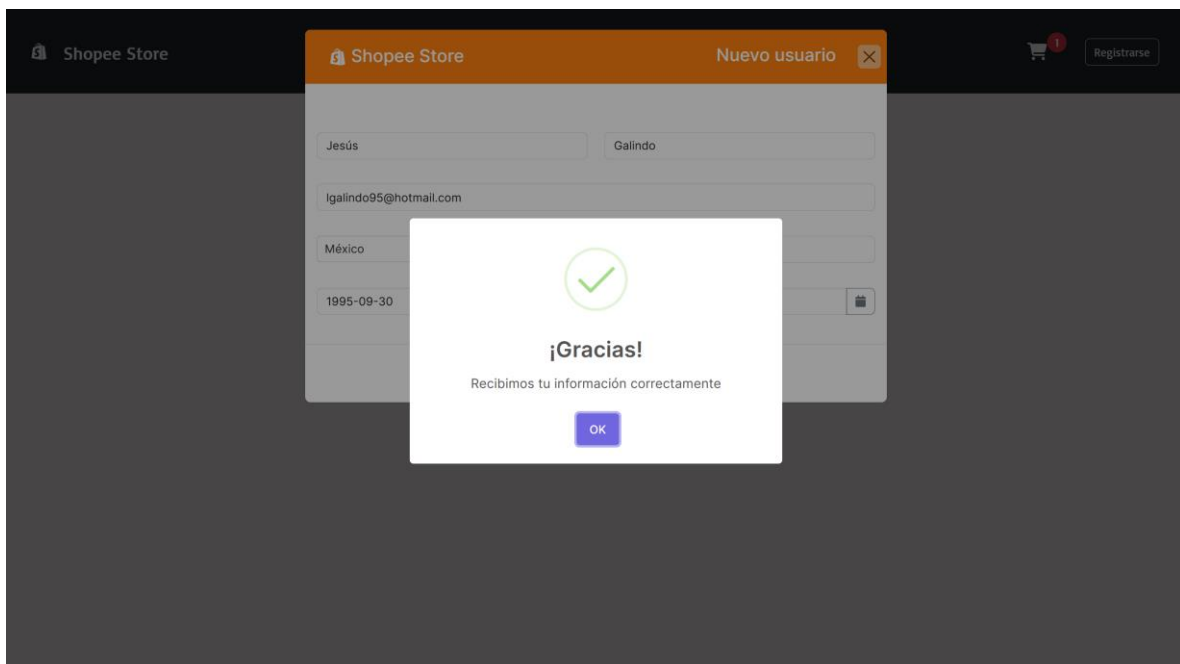
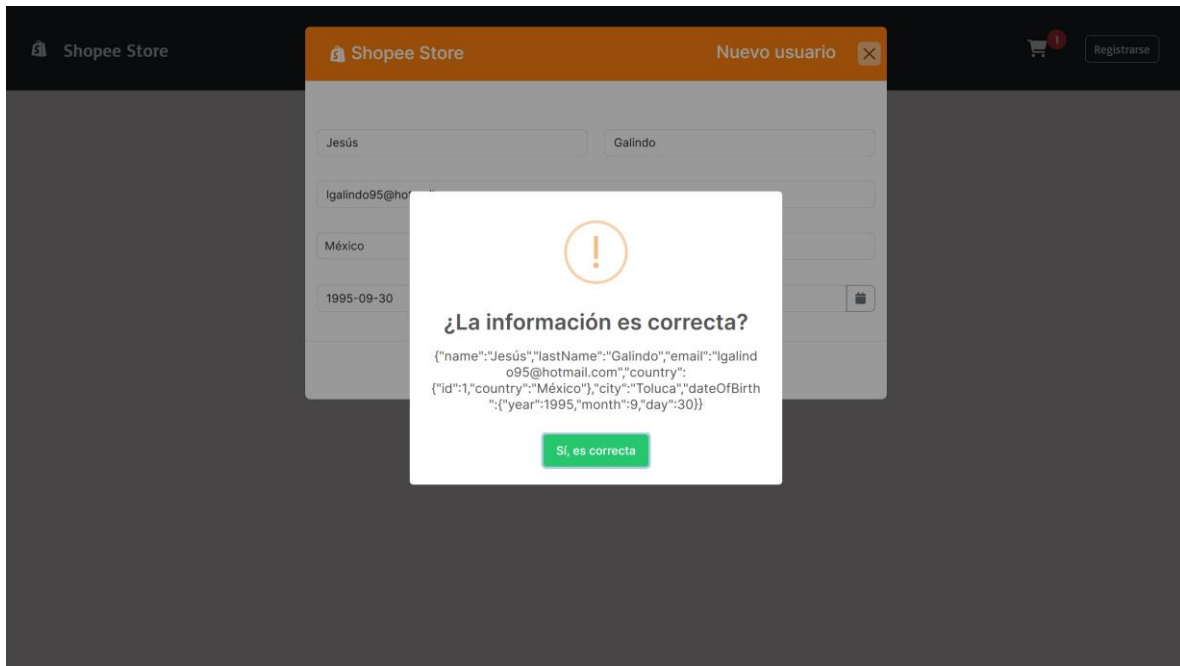


The screenshot shows the same 'Nuevo usuario' modal window, but now with validation messages. The 'Nombre' field has a red border and the message 'Escribe tu nombre'. The 'Apellido' field has a red border and the message 'Escribe tu apellido'. The 'Correo electronico' field has a red border and the message 'Ingresa un email valido'. The 'País' field is now a standard dropdown menu. There are also 'Cancelar' and 'Registrarse' buttons at the bottom of the form.

Para esta simulación se utiliza una alerta de confirmación de envío de datos, mostrando los mismos en formato **JSON** a manera de demostración del request al hacer submit.

Para lo anterior se requiere alguna librería como Sweet Alert:

<https://github.com/sweetalert2/ngx-sweetalert2>





Se agrega un componente nuevo que contendrá el banner de fondo, la barra de búsqueda y un select para hacer filtros.

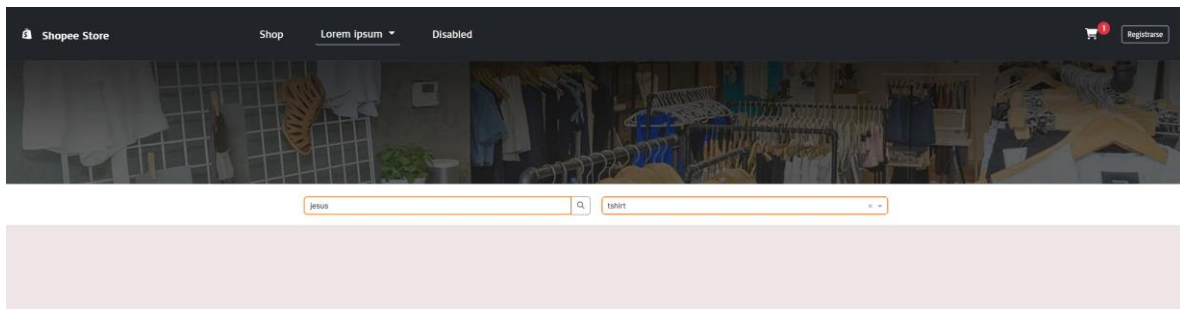
Para esta barra de búsqueda se agregaron controles reactivos uno para la búsqueda libre y otro para un select que filtrará por tipo de producto. Cabe mencionar que hasta este punto aun no se hacen los filtros puesto que no se han agregado servicios, sin embargo, se dejan los controles listos.

```
1 import { Component, OnInit, ViewEncapsulation } from '@angular/core';
2 import { FormControl } from '@angular/forms';
3 import { debounceTime } from 'rxjs/operators';
```

**debounceTime** es un operador de **RxJS** parecido al método `setTimeout()` que nos va ayudar a devolver la respuesta de los controles en un intervalo de tiempo determinado.

```
1 searchProduct = new FormControl();
2 searchCategory = new FormControl();
```

```
1 ngOnInit(): void {
2
3   this.searchProduct.valueChanges
4     .pipe(debounceTime(500))
5     .subscribe(value => {
6       console.log('searchProduct has changed:', value)
7     });
8
9   this.searchCategory.valueChanges
10    .pipe(debounceTime(500))
11    .subscribe(value => {
12      console.log('searchCategory has changed:', value)
13    });
14 }
```

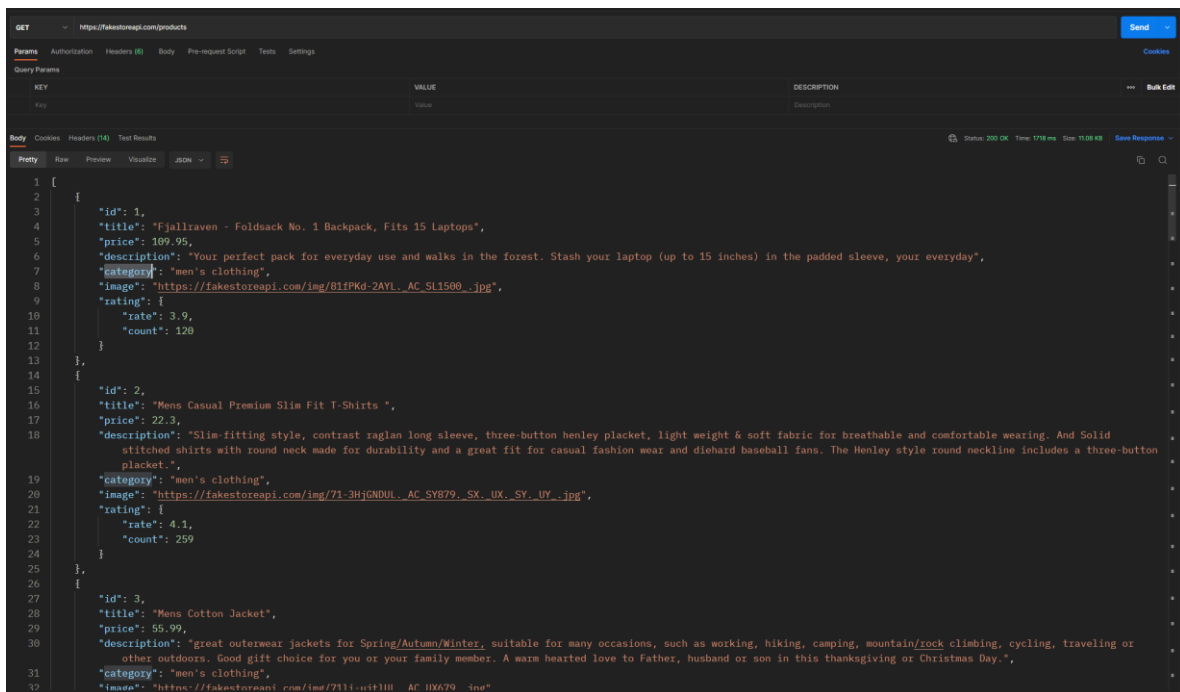


Para la sección principal donde se mostrarán las tarjetas se crea un nuevo componente y un servicio para empezar a integrar el consumo del api, para este proyecto se utiliza la siguiente:

<https://fakestoreapi.com/>

Es necesario agregar HttpClientModule en nuestro modulo.

Probamos en Postman:



```
import { HttpClientModule } from '@angular/common/http';
```

Importar nuestro servicio:

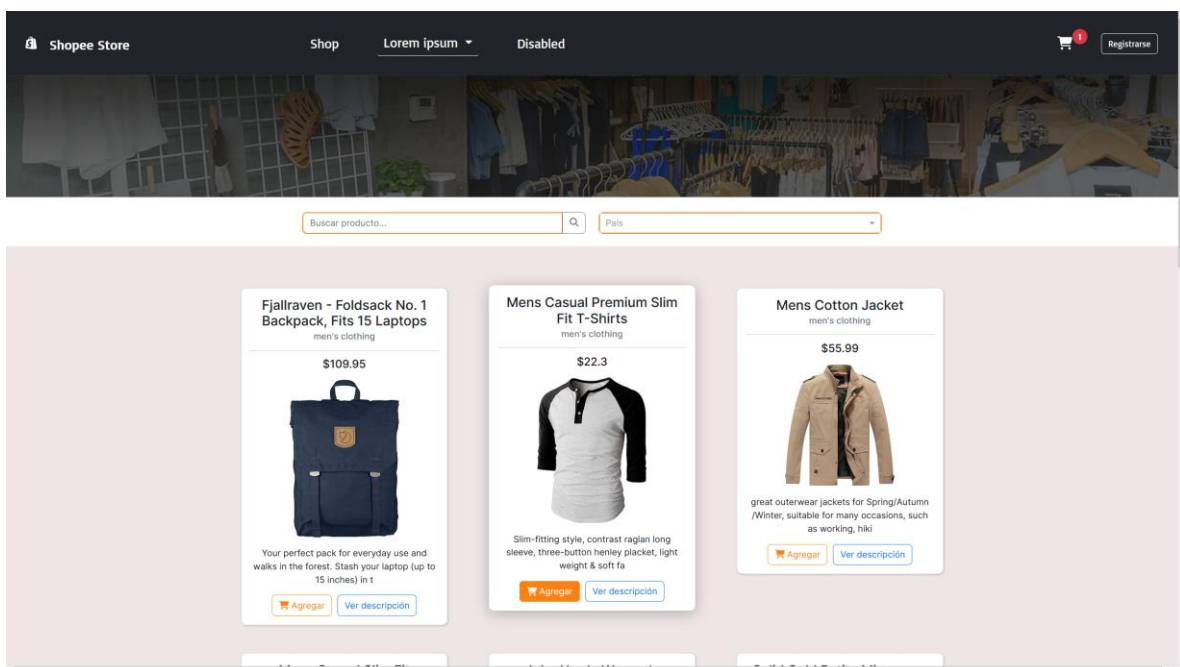
```
import { StoreServiceService } from '../main-content/store-service.service';
```

y agregarlo en providers:

```
providers: [HttpClientModule, StoreServiceService],
```

```
1 import { Component, OnInit } from '@angular/core';
2 import { StoreServiceService } from './store-service.service'
3
4 @Component({
5   selector: 'app-main-content',
6   templateUrl: './main-content.component.html',
7   styleUrls: ['./main-content.component.css']
8 })
9 export class MainContentComponent implements OnInit {
10
11   constructor(
12     private _api: StoreServiceService
13   ) { }
14
15   ngOnInit(): void {
16   }
17
18 }
19
```

Finalmente después de integrar servicios y estilar un poco las cards el resultado es:



Para darle continuidad a los campos de búsqueda que se definieron como **formControl**, se implementó una función para hacer el filtro, pero antes hay que resolver el problema de que los campos de búsqueda se encuentran en un componente distinto al componente donde se renderizan las cards con los productos.

Para solucionar el problema es necesario comunicar los datos entre 2 componentes distintos, para lo cual se utilizó **RXJS Subject**.

Se crea un servicio que será compartido por ambos componentes y por medio del cual uno enviara datos y el otro recibirá datos utilizando **subscribe**.

### Servicio

```
1 import { Injectable } from '@angular/core';
2 import { Observable, Subject } from 'rxjs';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class SharedServiceService {
8
9   private _send_data = new Subject<string>();
10
11   constructor() {}
12
13   setData(value:string){
14     this._send_data.next(value);
15   }
16
17   getData():Observable<string>{
18     return this._send_data.asObservable();
19   }
20 }
21
```

### Agregar en modulo

```
1 providers: [StoreServiceService, SharedServiceService],
```

### Componente que envía datos

```
1 this.searchProduct.valueChanges
2   .pipe(debounceTime(500))
3   .subscribe(value => {
4     //Con esta función enviamos el valor que recupera el input hacia el componente donde se renderizan las cards
5     this.sharedService.setData(value);
6   });
```

## Componente que recibe datos

```
1  ngOnInit(): void {  
2    this.getProducts();  
3  
4    this.sharedService.getData().subscribe(result =>{  
5      console.log("result: ", result);  
6  
7      const temporal = this._dataProducts.filter(item => item.title.toLowerCase().includes(result.toLocaleLowerCase()));  
8  
9      console.log("temporal: ", temporal);  
10     this.dataProductsTemp = temporal;  
11   })  
12 }
```

Al tiempo que se reciben los datos (que es lo que el usuario escribió en el campo de texto) se hace el filtro utilizando algunos métodos de **EcmaScript 6** como **filter** e **includes**.

```
1  this.searchCategory.valueChanges  
2    .pipe(debounceTime(500))  
3    .subscribe(value => {  
4  
5      console.log("Value category:", value.category);  
6  
7      //Aquí tuve dificultad para hacer el filtro por categoria :(  
8  
9    });
```

Nota: Aquí tuve dificultad para hacer el filtro por categoria :(