Práctica Angular y Firebase

Introducción

La práctica consiste en la creación de una aplicación tipo CRUD - Create, Read, Update, Delete- con las siguientes características:

- Angular 8
- *Firebase* como base de datos en tiempo real desde la que leer y guardar información sobre productos
- Bootstrap 4 como framework CSS
- Icons de Font Awesome 5
- Módulo Toaster de Angular para notificaciones

Se guía al alumno en el proceso de creación de la primera parte de la aplicación, para finalmente solicitarle que, a partir de todo lo visto termine por si mismo la parte restante.

Pasos a seguir

1) Creación de proyecto Angular mediante Angular CLI

```
ng new crud-firebase
```

Una vez generado el proyecto, lo abrimos e instalamos dos módulos necesarios para el trabajo con Firebase:

```
npm install firebase @angular/fire
```

El primero (firebase) es necesario para Node y el segundo (@angular/fire) para Angular.

2) Añadir al fichero index.html las CDN's de Bootstrap 4 y Font Awesome

Bootstrap

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
```

Font Awesome

```
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.3.1/css/all.css"
integrity="sha384-mzrmE5qonljUremFsqc01SB46JvROS7bZs3IO2EmfFsd15uHvIt+Y8vEf7N7fWAU"
crossorigin="anonymous">
```

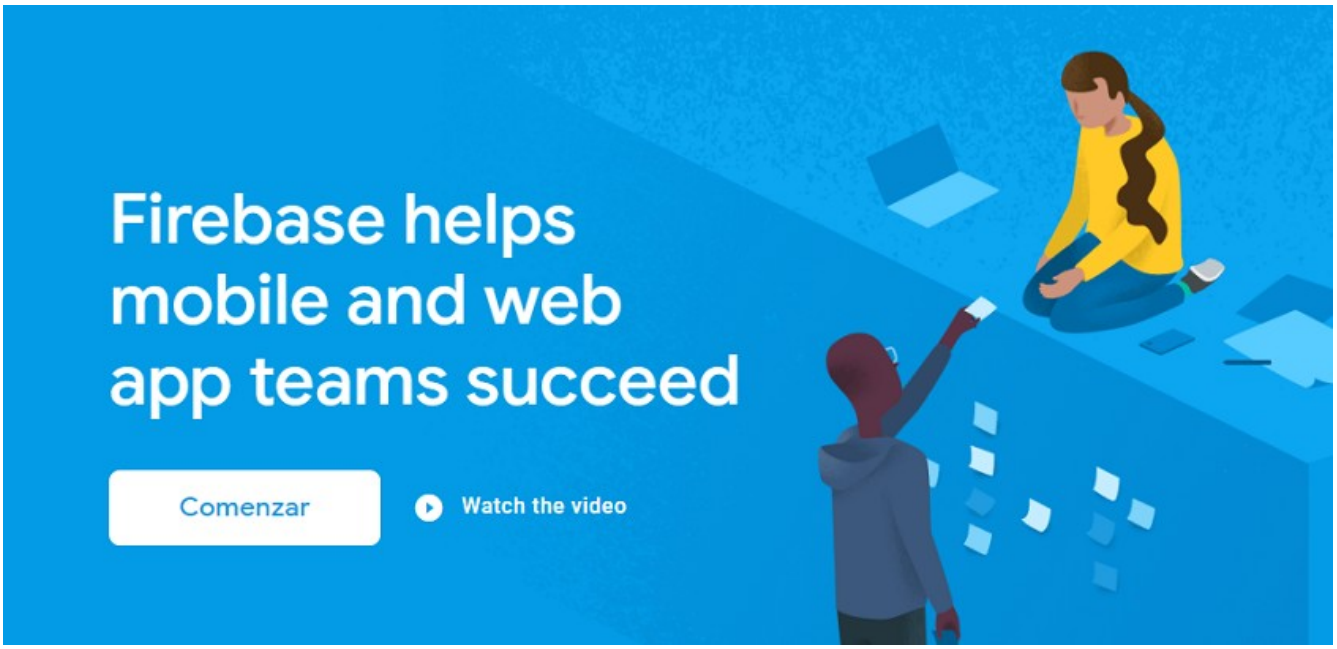
Pegamos el código anterior en la sección <head> del *index.html*.

3) Creación del proyecto Firebase

https://firebase.google.com/

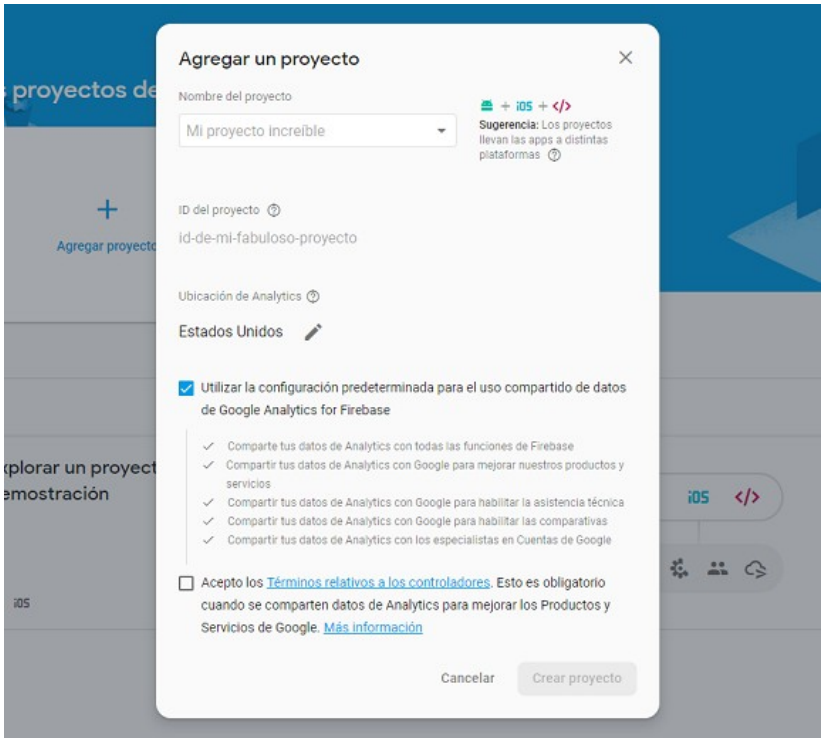
Nota: necesitamos una cuenta de Gmail para trabar con Firebase.

- Pulsamos sobre el botón ‘Comenzar’:

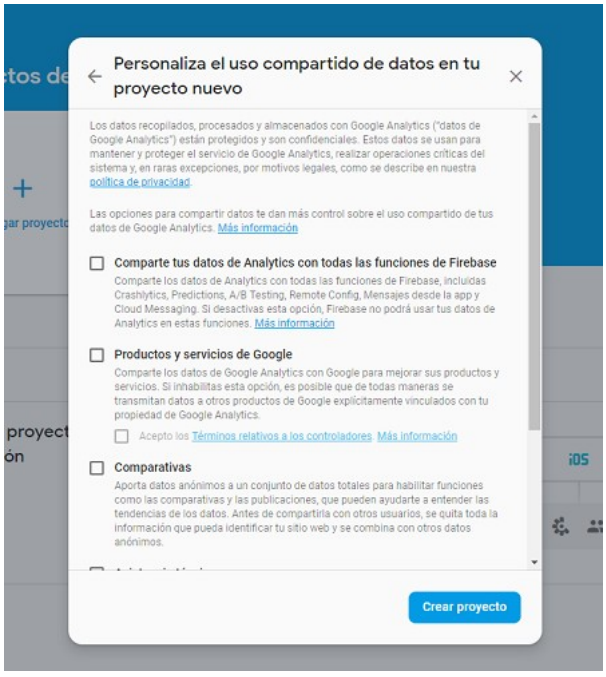


- En la nueva página pulsamos el botón ‘Añadir proyecto’:

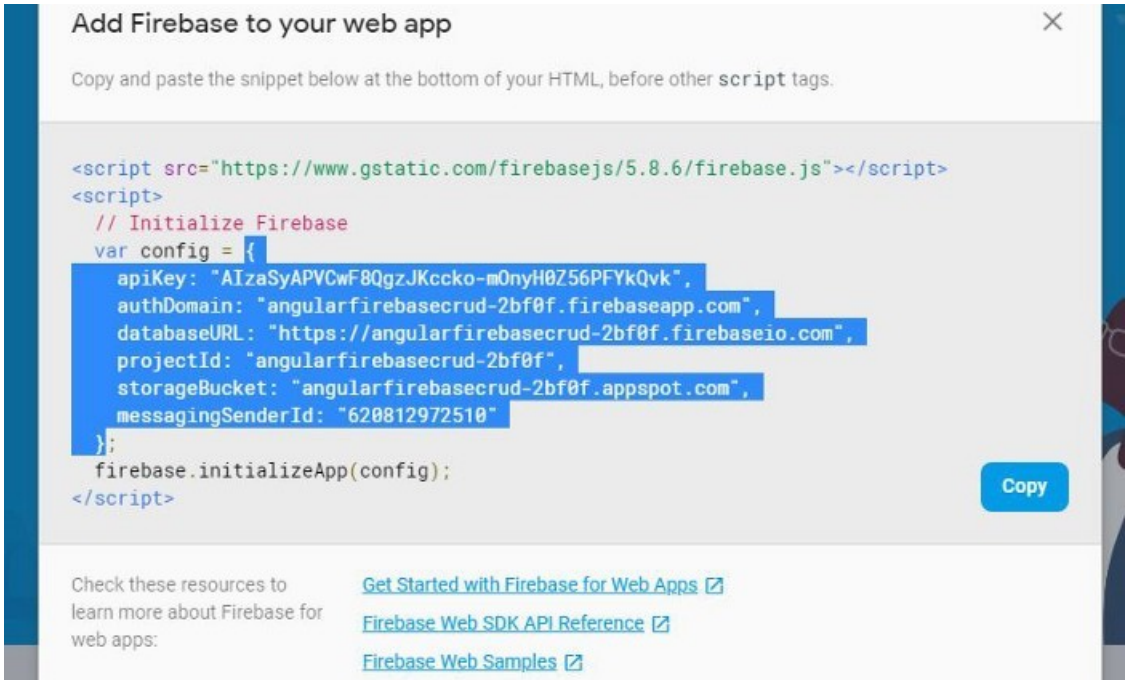
- En el cuadro de diálogo que aparece:



  - Proporcionamos un nombre para la base de datos, por ejemplo `crud-firebase`, lo cual genera un **ID** para el proyecto.
  - Establecemos la ubicación a España (afecta para monedas, fechas, etc).
  - Pulsamos el botón 'Continuar'. Aparecerá una nueva ventana con varios checkboxs. Podemos ignorarlos y pulsar el botón 'Crear Proyecto'.



- En breve nos indicará que el proyecto está listo. Pulsamos el botón 'Continuar', lo cual nos redirige a la pantalla principal de Firebase.
- Ahora hemos de indicar si queremos trabajar con Android, iOS o para aplicaciones web. Pulsamos en 'aplicaciones web'.
- En la nueva página que aparece veremos un código Javascript. Este código lo necesitaremos en nuestra aplicación Angular.



- Volvemos a nuestro IDE.

4) Incorporación de los módulos de Firebase

Abrimos el fichero "app.module.ts". En la sección "imports" añadimos los módulos AngularFireModule y AngularFireDatabaseModule (con sus *imports* correspondientes).

Notad que AngularFireModule, que es el módulo principal, invoca a una función inicializadora y le pasa el código de nuestro proyecto de Firebase (ojo! Solo el contenido del objeto "config").

Debe quedar así el fichero:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

// Firebase
import { AngularFireModule } from 'angularfire2';
import { AngularFireDatabaseModule } from 'angularfire2/database';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
       AngularFireModule.initializeApp({
       apiKey: 'AIzaSyDxF2ddLeViR9NJk434343KstjKbwFuOfA',
       authDomain: 'crud-firebase-1334d.firebaseapp.com',
       databaseURL: 'https://crud-firebase-1094d.firebaseio.com',
       projectId: 'crud-firebase-1134d',
       storageBucket: 'crud-firebase-10534d.appspot.com',
       messagingSenderId: '3813794610'
    }),
    AngularFireDatabaseModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

5) Configurar la base de datos en Firebase

En Firebase, cerramos la ventana que nos mostraba el código Javascript. Hacemos clic en “Desarrollo”, ubicado en la izquierda de la pantalla. A continuación, seleccionamos ”Database”.

En la pantalla que aparece bajamos hasta el apartado que dice ”Realtime Database” y pulsamos el botón “Crear base de datos”.

En el cuadro de diálogo que aparece seleccionamos “Empezar con el modo de prueba”. Aceptamos.

Ya hemos terminado con Firebase. Volvemos a la aplicación Angular.

6) Creación del modelo

Comenzamos por crear el modelo de la aplicación, que para esta práctica tan básica consiste en una clase llamada Producto:

```
ng g cl modelos/producto
```

```
export class Producto {
    $key: string;
    descripcion: string;
    precio: number;
}
```

7) Creación del servicio

El servicio va a proporcionar a los diferentes componentes de la aplicación las operaciones CRUD relativas a productos. De esta manera, el servicio encapsulará el acceso a Firebase.

Creamos el servicio:

```
ng g s servicios/producto
```

Y lo añadimos en la sección *providers* del fichero *app.module.ts*.

Procedamos a la implementación del servicio:

Necesitaremos un objeto *AngularFireList* para recibir de Firebase la lista de productos. Creamos la lista como atributo de la clase:

```
listaProd: AngularFireList<any>;
```

También necesitaremos inyectar en el constructor un objeto *AngularFireDatabase*. Este objeto nos permitirá conectar con Firebase.

```
constructor(private firebase: AngularFireDatabase) { }
```

Además, hemos de crear un método que denominaremos *getProductos* que recibirá de Firebase la lista de productos y los devolverá. Los productos se alojarán en Firebase en una colección que identificaremos por el nombre 'productos'.

```
getProductos() {
    return this.listaProd = this.firebase.list('productos');
}
```

Finalmente hemos de crear el resto de os métodos para el mantenimiento de productos:

```
nuevoProducto(producto: Producto) {
     this.listaProd.push({
       descripcion: producto.descripcion,
       precio: producto.precio
     });
}

actualizarProducto(producto: Producto) {
     this.listaProd.update(producto.$key, {
       descripcion: producto.descripcion,
       precio: producto.precio
     });
}

eliminarProducto($key: string) {
     this.listaProd.remove($key);
}
```

8) Creación de componentes

Para la aplicación que queremos necesitaremos tres componentes, pues se trata de crear una vista que mostrará un formulario para dar de alta productos y a su vez permitirá visualizar la lista de productos existentes (sincronizada con Firebase). Los productos de la lista se podrán editar o eliminar. El tercer componente es el producto como tal.

Creamos los componentes:

```
ng g c componentes/productos
ng g c componentes/productos/lista-productos
ng g c componentes/productos/producto
```

9) HTML del componente app.component

Eliminamos todo el contenido del fichero y pegamos el siguiente código:

```
<nav class="navbar navbar-dark bg-dark">
    <a href="/" class="navbar-brand">
    Gestión de productos
  </a>
</nav>
<app-productos></app-productos>
```

Notad que `<app-productos></app-productos>` es el selector del componente base que contendrá el formulario de alta de producto y la lista de productos.

Arrancamos el servidor. Deberíamos ver la barra de navegación y justo debajo el mensaje "productos works!".

10) HTML del componente productos.component

Necesitamos establecer el HTML del componente base que contendrá el formulario de alta de producto y la lista. Básicamente vamos a añadir los selectores correspondientes a tales componentes.

Eliminamos el contenido del fichero y pegamos el siguiente código:

```
<div class="container">
    <div class="row">
        <div class="col-md-5">
            <app-producto></app-producto>
        </div>
        <div class="col-md-5">
```

```
            <app-lista-productos></app-lista-productos>
        </div>
    </div>
</div>
```

11) HTML del componente producto.component

Se trata del formulario para crear y editar productos. Eliminamos el contenido del fichero y pegamos el siguiente código:

```
<div class="card mt-5">
<div class="card-body">
<form #productForm="ngForm" (ngSubmit)="onSubmit(productForm)">
     <input type="hidden" name="$key"
     #$key="ngModel"
     [(ngModel)]="productoService.productoSeleccionado.$key">

     <div class="form-group">
          <input type="text" class="form-control"
          name="descripcion" #descripcion="ngModel"

     [(ngModel)]="productoService.productoSeleccionado.descripcion"
          placeholder="Descripción">
     </div>
     <div class="form-group">
          <div class="input-group">
               <div class="input-group-prepend">
                    <span class="input-group-text">€</span>
               </div>
               <input type="number" class="form-control"
               name="precio" #precio="ngModel"
```

```
      [(ngModel)]="productoService.productoSeleccionado.precio"
                placeholder="Precio">
          </div>
      </div>

      <div class="form-group">
          <button class="btn btn-primary" type="submit">
          <i class="fas fa-plus-circle"></i> Crear producto
      </button>
          <button class="btn btn-secondary" type="reset" (
          click)="resetForm(productForm)">
          <i class="fas fa-sync-alt"></i> Reiniciar
      </button>
      </div>
</form>
</div>
</div>
```

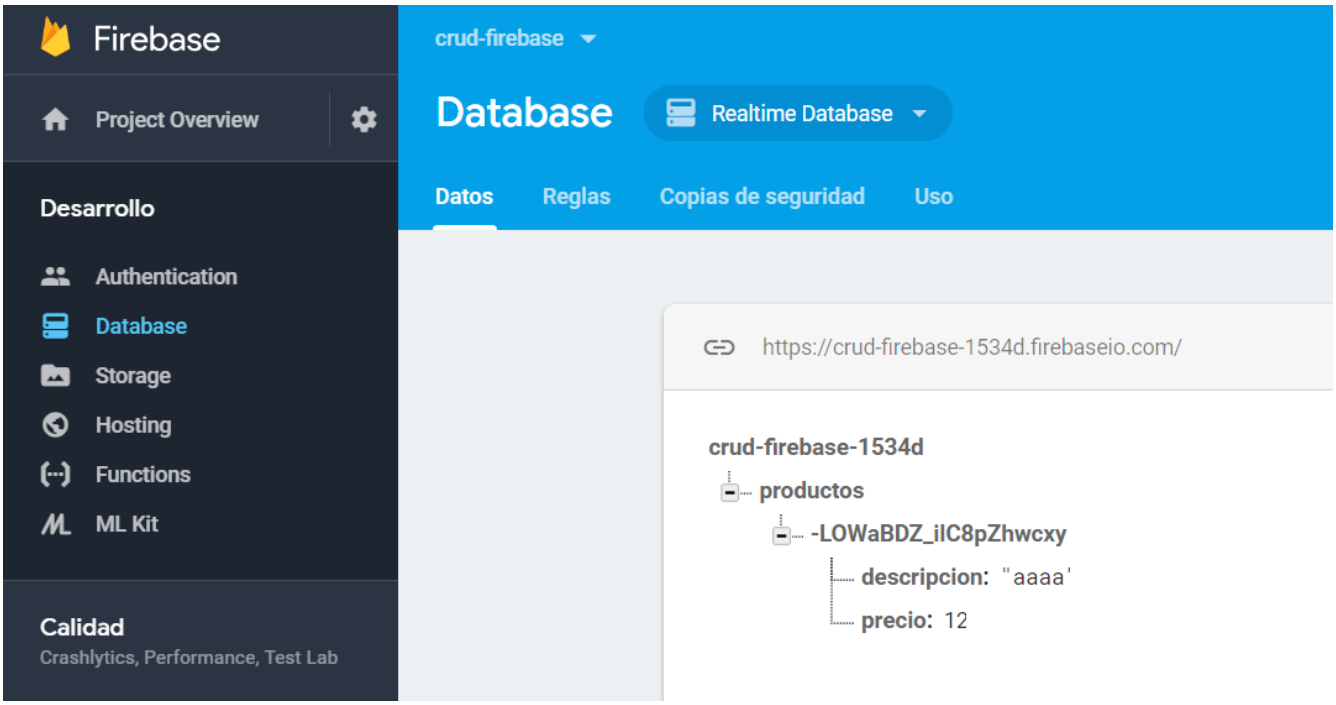
11) El typescript del componente producto.component

Necesitareis el siguiente código:

```
export class ProductoComponent implements OnInit {

  constructor(private productoService: ProductoService
  ) { }

  ngOnInit() {
    this.productoService.getProductos();
    this.resetForm();
  }
```

```
  onSubmit(productForm: NgForm) {
    if (productForm.value.$key == null) {
      this.productoService.nuevoProducto(productForm.value);
    } else {
      this.productoService.actualizarProducto(productForm.value);
    }
    this.resetForm(productForm);
   }

  resetForm(productForm?: NgForm) {
    if (productForm != null) {
      productForm.reset();
      this.productoService.productoSeleccionado = new Producto();
    }
  }
}
```

Si ahora vamos a Firebase deberíamos ver almacenado nuestro producto:



12) Proporcionar feedback al usuario mediante notificaciones toastr

Instalamos el paquete ***toastr***:

```
npm install ngx-toastr
```

Ahora abrimos el fichero *angular.json* y en la sección "styles", bajo el valor "styles.css" indicamos la ruta hasta el fichero css de toastr:

```
...
 "styles": [
  "src/styles.css",
  "./node_modules/ngx-toastr/toastr.css"
],
"scripts": []
...
```

Ahora tenemos que habilitar las animaciones de Angular, pues son una dependencia de Toastr:

```
npm install @angular/animations
```

Y en el fichero *app.module.ts* importamos BrowserAnimationsModule:

```
import { BrowserAnimationsModule } from
'@angular/platform-browser/animations';
```

y lo añadimos en la sección *imports*.

Y también hacemos lo propio para toaster:

```
import { ToastrModule } from 'ngx-toastr';
```

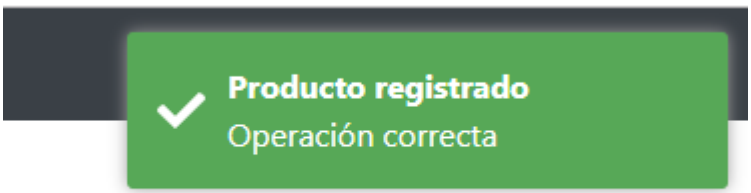
Por último hemos de inyectar el servicio ToastrService que trae de serie Toastr e inyectarlo en el constructor de product.component:

constructor(private productoService: ProductoService**, private toastrService: ToastrService**
){}

Y en el método *onSubmit*:

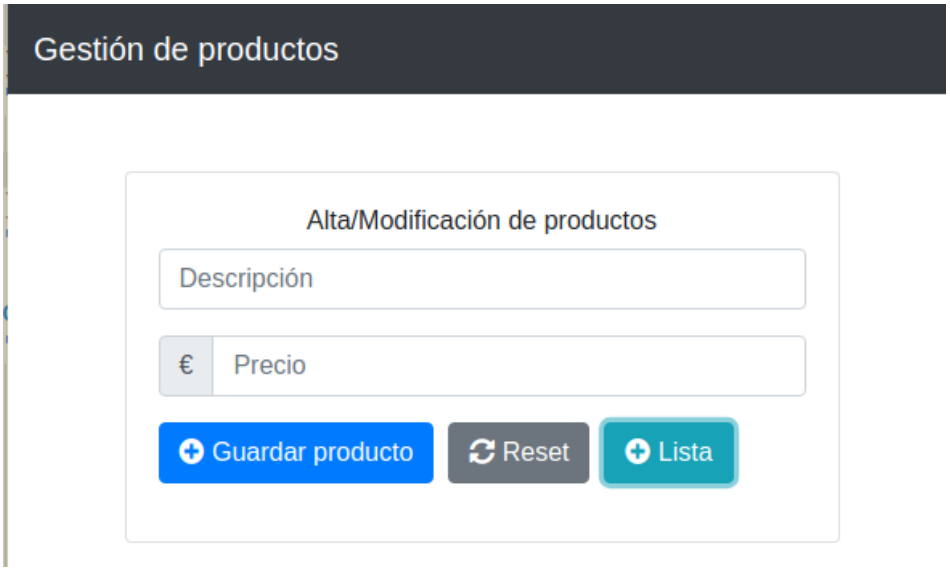
```
this.toastrService.success('Operación correcta', 'Producto registrado');
```

Ahora si grabamos un producto deberíamos ver una notificación:



Se pide

Realizar la parte de la práctica restante.
Se requiere implementar el componente que lista los productos. En la lista deben aparecer los productos existentes en Firebase. Se mostrarán en una tabla y por cada registro se debe poder editar/eliminar. La edición consistirá en mostrar el producto en el componente de Alta/Modificación de productos, por lo que el botón “Guardar” puede actuar tanto para crear nuevos productos como para modificar un productos existente.
La tabla que muestra los productos estará oculta por defecto. El usuario pulsando un botón podrá mostrarla u ocultarla.



Aplicación con la lista de productos visibles: