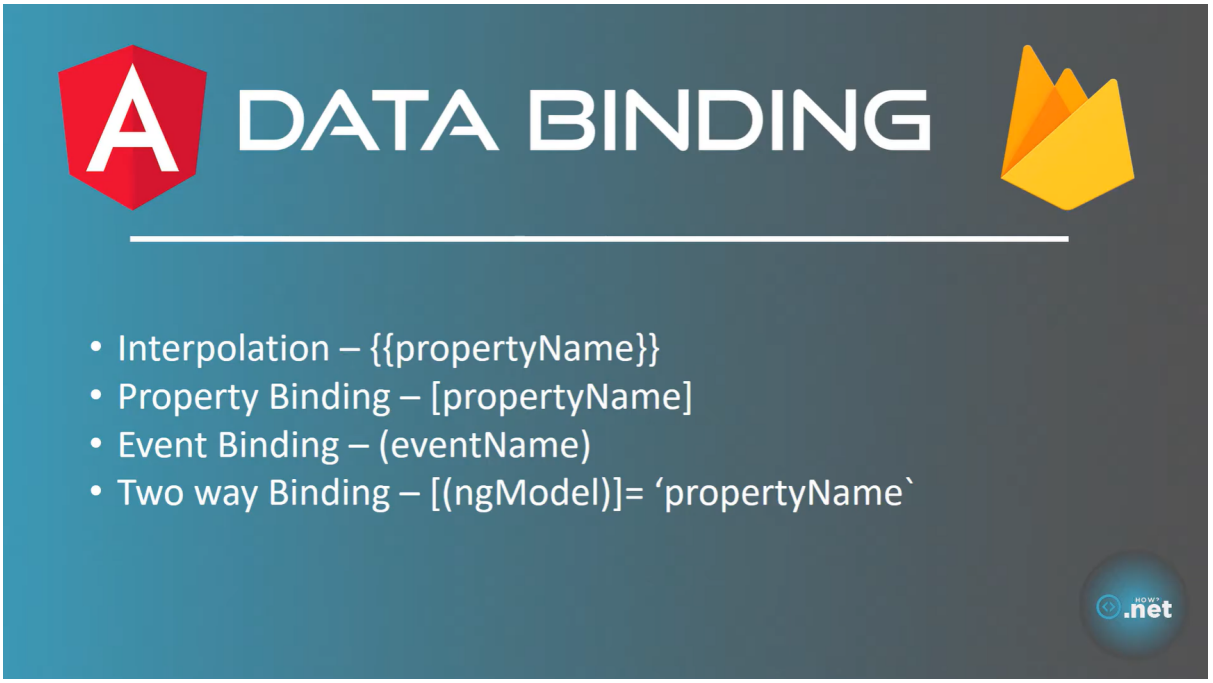
1. DATA BINDINGS



- 2.1. En el home.component.html vamos hacer una prueba de Interpolation, Event Binding y Two Way Binding

```
<!-- <p>home works!</p> -->

<h1>Couse title is: {{ courseTitle }}</h1>

<h2 (click)="clicked()">
    Click here!
</h2>

<input [(ngModel)]="courseTitle">
```

- 2.2. En el home.component.ts …

```
export class HomeComponent implements OnInit {

  courseTitle = "Library";

  constructor() { }

  clicked() {
    console.log("h2 clicked!");
  }
```

```
  ngOnInit(): void {
  }

}
```

- 2.3. Y para que funcione el [(ngModel)], en el app.module.ts …

```
imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule
  ],
```

2. ALL COMPONENTS

- 3.1. ng g c components/home
- 3.2. ng g c components/book-list
- 3.3. ng g c components/book-details
- 3.4. ng g c components/add-book
- 3.5. ng g c components/edit-book
- 3.6. ng g c components/delete-book
- 3.7. ng g c components/navbar
- 3.8. ng g c components/footer

3. ROUTES CONFIGURATION

En el app-routing.module.ts tenemos que definir las siguientes rutas:

```
const routes: Routes = [
  {
    path: '',
    component: HomeComponent
  },
  {
    path: 'book-list',
    component: BookListComponent
  },
  {
    path: 'book-details/:id',
    component: BookDetailsComponent
  },
  {
    path: 'add-book',
    component: AddBookComponent
  },
  {
    path: 'edit-book/:id',
    component: EditBookComponent
  },
  {
    path: 'delete-book/:id',
    component: DeleteBookComponent
  }
];
```

4. NAVIGATION BAR

- 4.1. En el app.component.html debemos borrar todo lo que ya viene por defecto y poner en su lugar

```
<app-navbar></app-navbar>

<router-outlet></router-outlet>
```

- 4.2. Después, en el index.html, antes de que acabe el </head>, enrutamos el siguiente archivo, el cual podemos conseguir guardando este raw: https://bootswatch.com/3/readable/bootstrap.min.css

```
<link rel="stylesheet" href="./assets/bootstrap-readable.min.css">
```

- 4.1. En el navbar.component.html creamos esta estructura

```
<!-- <p>navbar works!</p> -->

<nav class="navbar navbar-default">
    <div class="container-fluid">

        <div>
            <a class="navbar-brand" [routerLink]="['/']">Library</a>
        </div>

        <ul class="nav navbar-nav">
            <li><a [routerLink]="['/']"
routerLinkActive="router-link-active">Home</a></li>
            <li><a [routerLink]="['/book-list']"
routerLinkActive="router-link-active">Book List</a></li>
            <li><a [routerLink]="['/add-book']"
routerLinkActive="router-link-active">Add Book</a></li>
        </ul>

    </div>
</nav>
```

- 4.2. Por otro lado, en su css …

```
li {
    list-style: none;
}
```

Library Home Book List Add Book

Couse title is: Library

Click here!

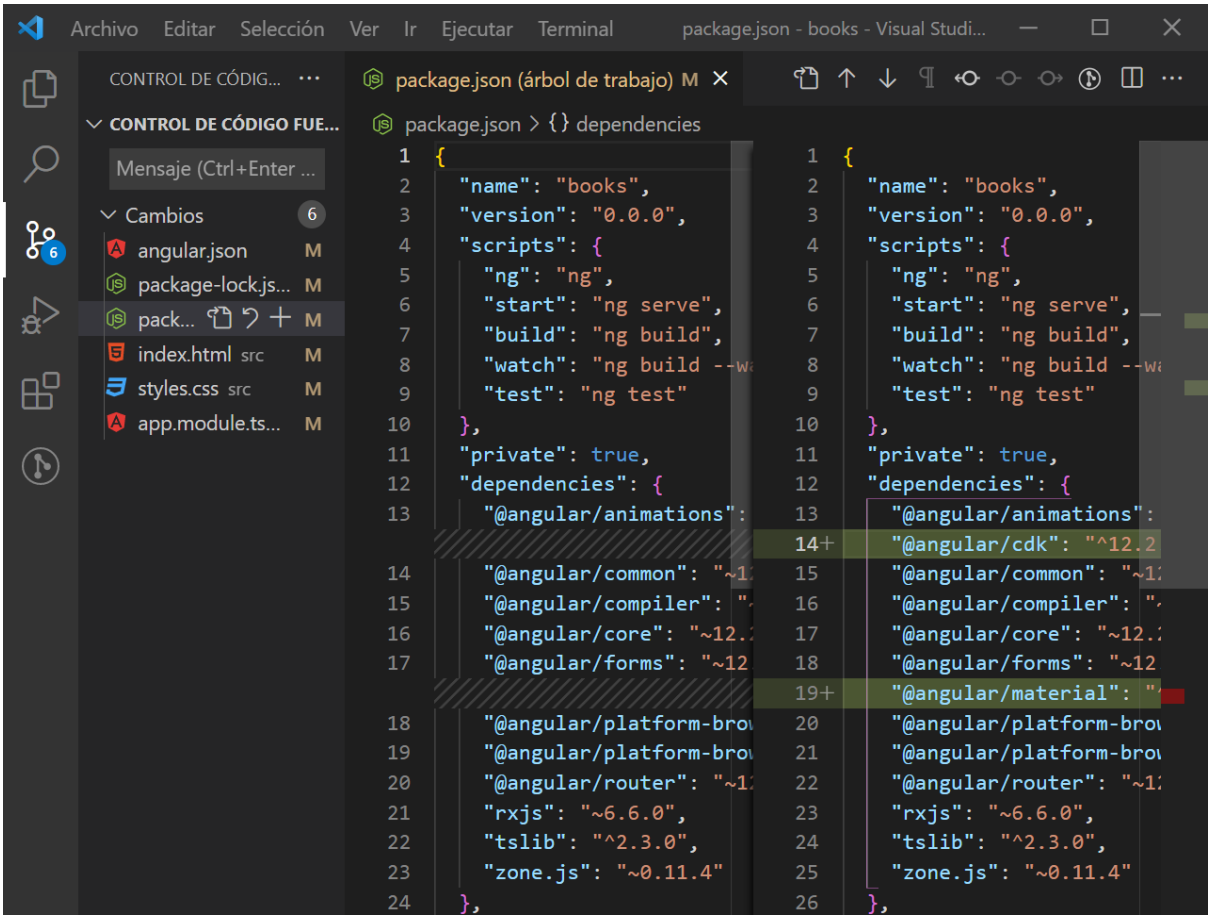
Library

Library Home Book List Add Book

book-list works!

5. MATERIAL DESIGN SETUP

- 5.1. ng add @angular/material



- 5.2. Ahora vamos a crear un nuevo módulo que será el propio de AngularMaterial, el cual importaremos en el app.module.ts

  ng g m modules/angular-material --flat

  Ejemplo:

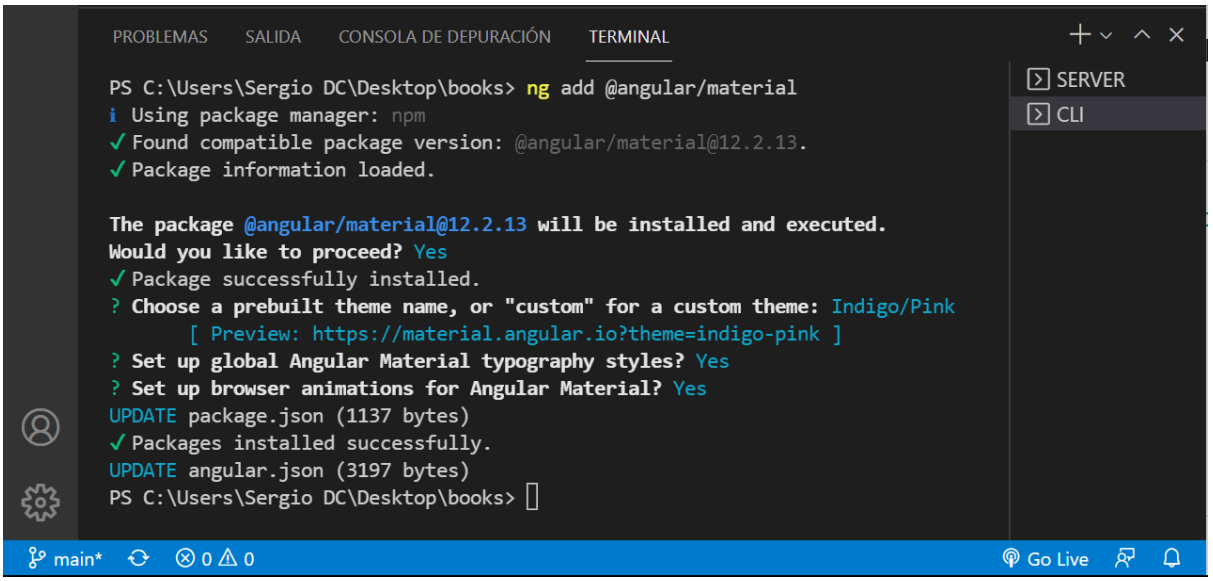
```
import { NgModule } from '@angular/core';
import { MatButtonModule } from '@angular/material/button';

@NgModule({
  declarations: [],
  imports: [MatButtonModule],
  exports: [MatButtonModule]
})

export class AngularMaterialModule { }
```

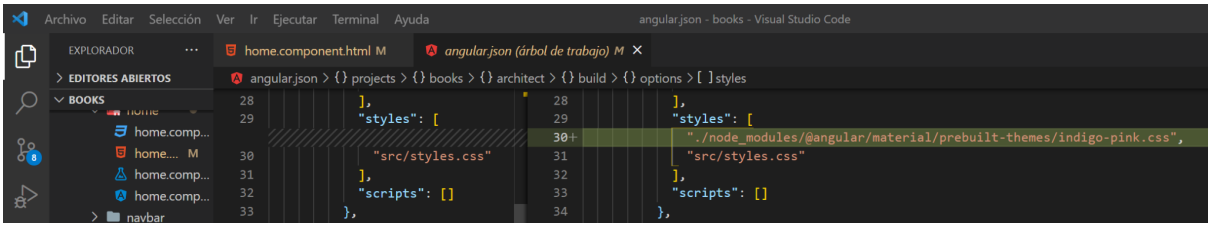
ATENCIÓN:
Si cuando hagamos esta primera prueba de Angular Material, vemos que el componente que estuvieras probando de Angular Material no funcionase, y que la consola del inspeccionar nos diese un error tal que:

Could not find Angular Material core theme. Most Material components may not work as expected.

Esto se soluciona reiniciando el servidor localhost:4200 que proyecta nuestra app.

Nota:
Cuando instalamos @angular/material, nos pide que elijamos un Theme por defecto, el cual, normalmente se suele elegir el primero de la lista, el “indigo-pink” (azul), y debemos saber que para cambiar este a otro de los que vienen por defecto en la lista, tan sólo debemos cambiar su ruta de acceso en el archivo de angular.json en la parte del array de styles:[ ]





- 5.3. Para ver si funciona, vamos al home.component.html, y borramos lo que habíamos puesto desde antes (test) y lo reemplazamos por:

```
<button mat-raised-button>Basic</button>
```

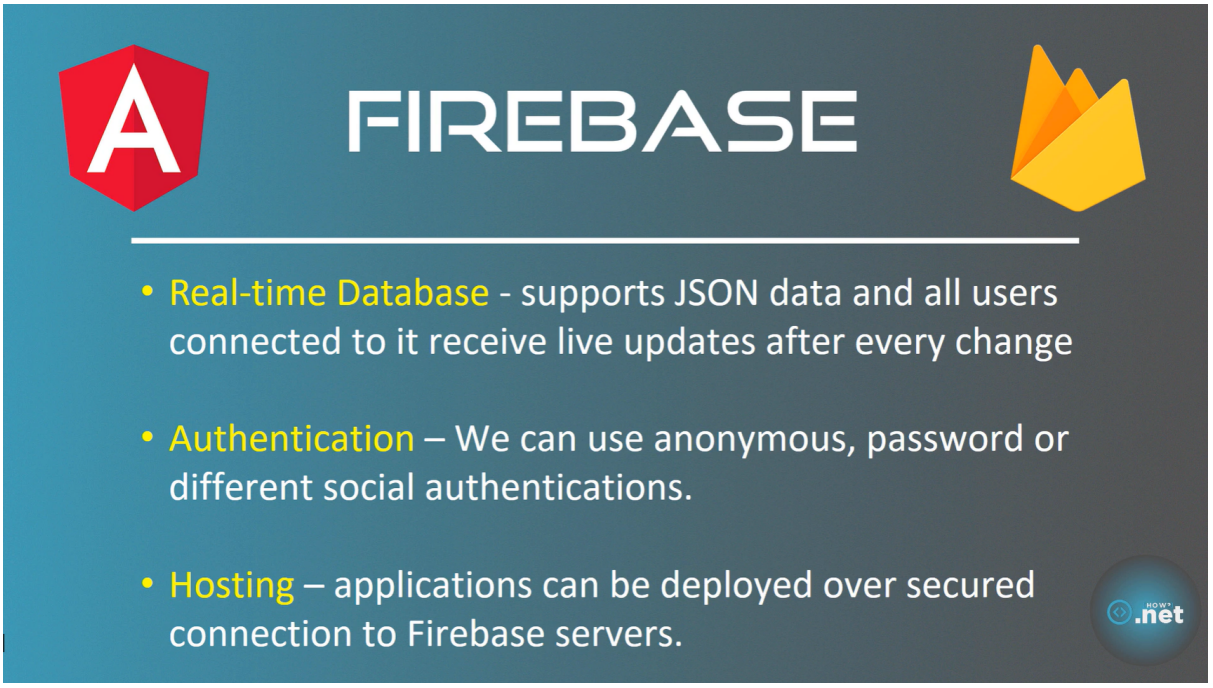
Y si vamos a nuestro navegador, vemos que AngularMaterial funciona perfectamente

- 5.4. De AngularMaterial ya sólo nos falta agregar el CDN hacia sus Icons, así que antes de acabar el tag </head> del index.html, agregamos …

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
```

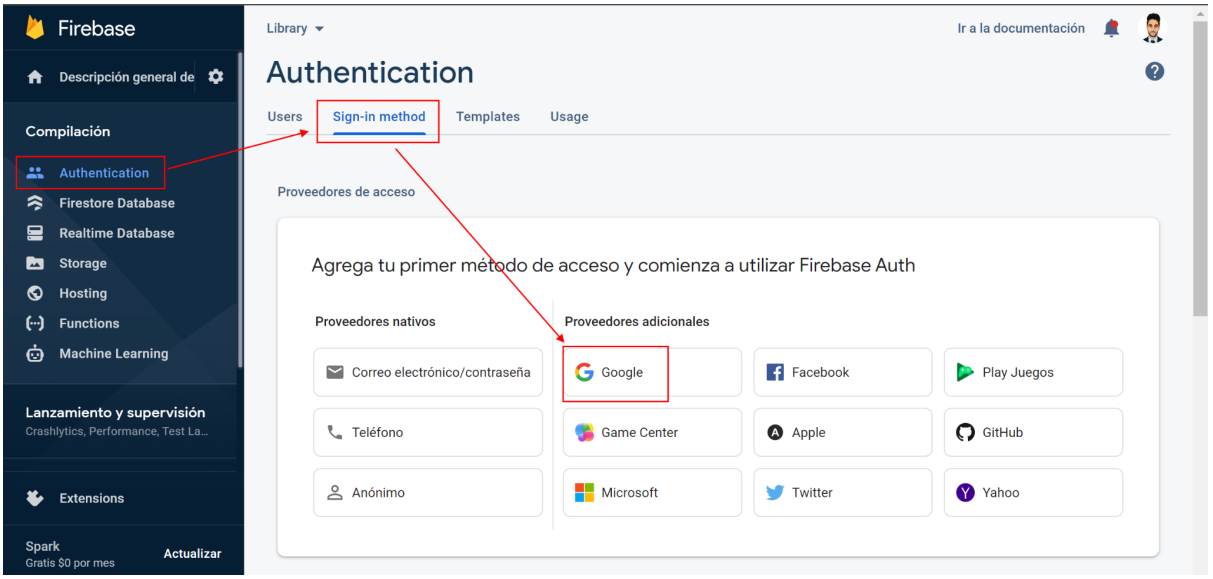
6. CREATE FIREBASE APPLICATION

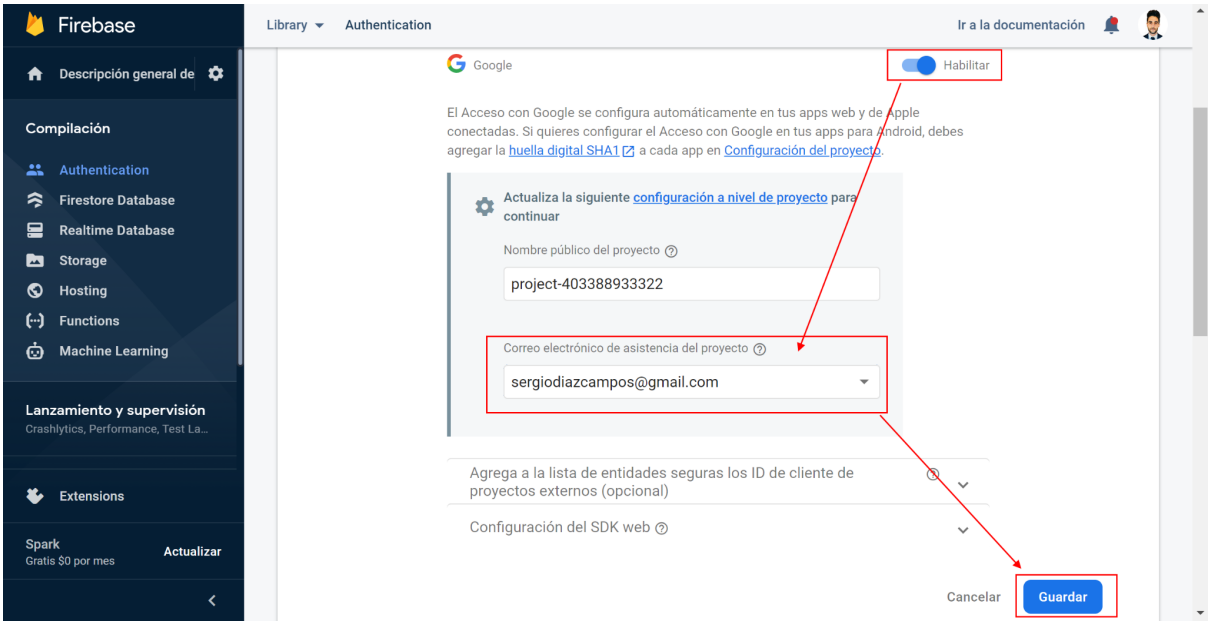
En Firebase vamos a contar con:



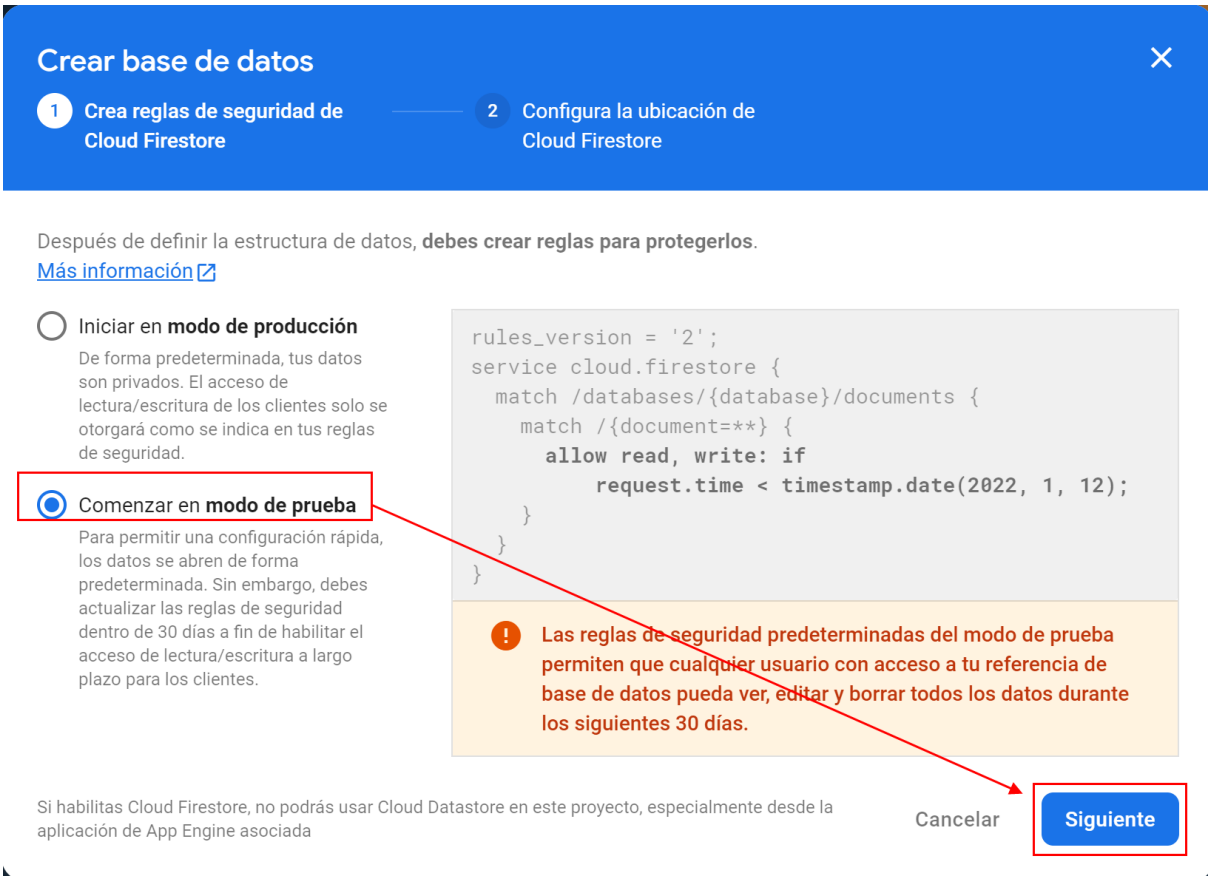
Una vez que estemos con nuestra cuenta de Google en Firebase, le damos a “Crear nuevo Proyecto”, y escribimos el nombre de nuestro proyecto (“library”) y le decimos que no queremos los Google Analytics.

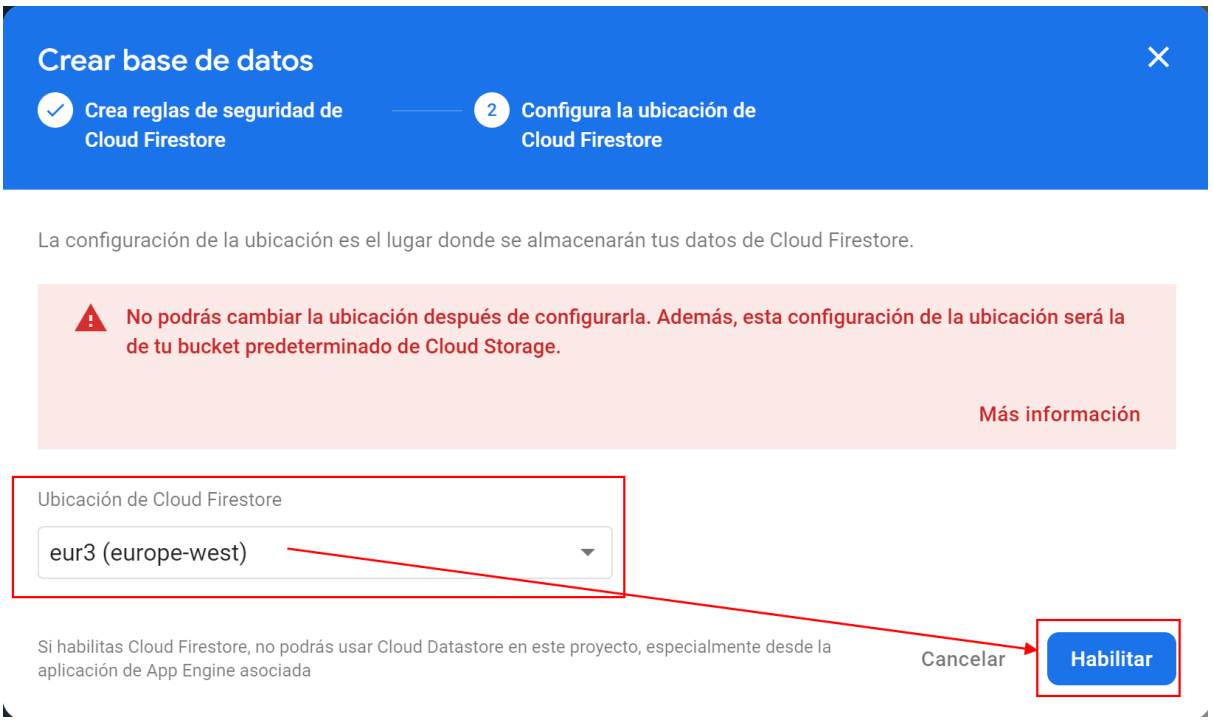
- 6.1. Lo primero que vamos a hacer es preparar la Autentificación con Google



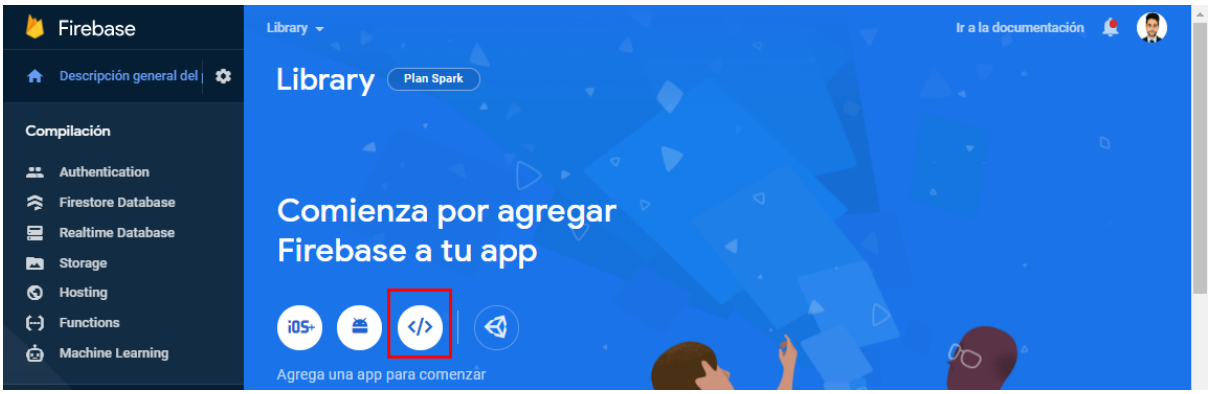


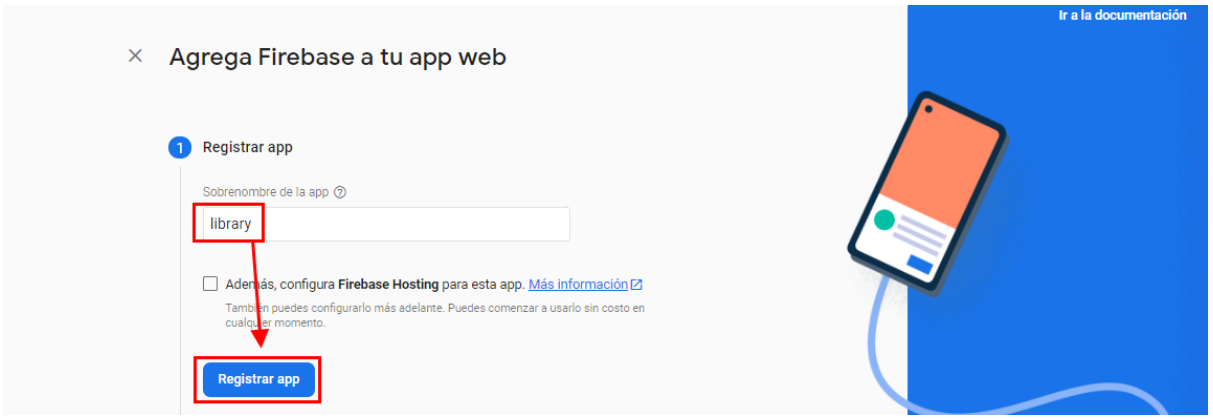
- 6.2. Lo siguiente será la BBDD de Firestore





- 6.3. Ahora vamos a configurar el proyecto para que Firebase sepa que estamos desarrollando una aplicación web





En el paso #2 de “Configurar el SDK” … NO HACER NADA… simplemente darle abajo a “aceptar”

- 6.4. Vamos al archivo de `src/app/environments/environment.ts`

```
// This file can be replaced during build by using the
`fileReplacements` array.
// `ng build` replaces `environment.ts` with `environment.prod.ts`.
// The list of file replacements can be found in `angular.json`.

// export const environment = {
//   production: false
// };

export const environment = {
  production: false,
  firebase: {
    apiKey: "yourApiKey",
    authDomain: "yourAuthDomain",
    projectId: "yourProjectId",
    storageBucket: "yourStorageBucket",
    messagingSenderId: "yourMessagingSenderId",
    appId: "yourAppId"
  }
};
```

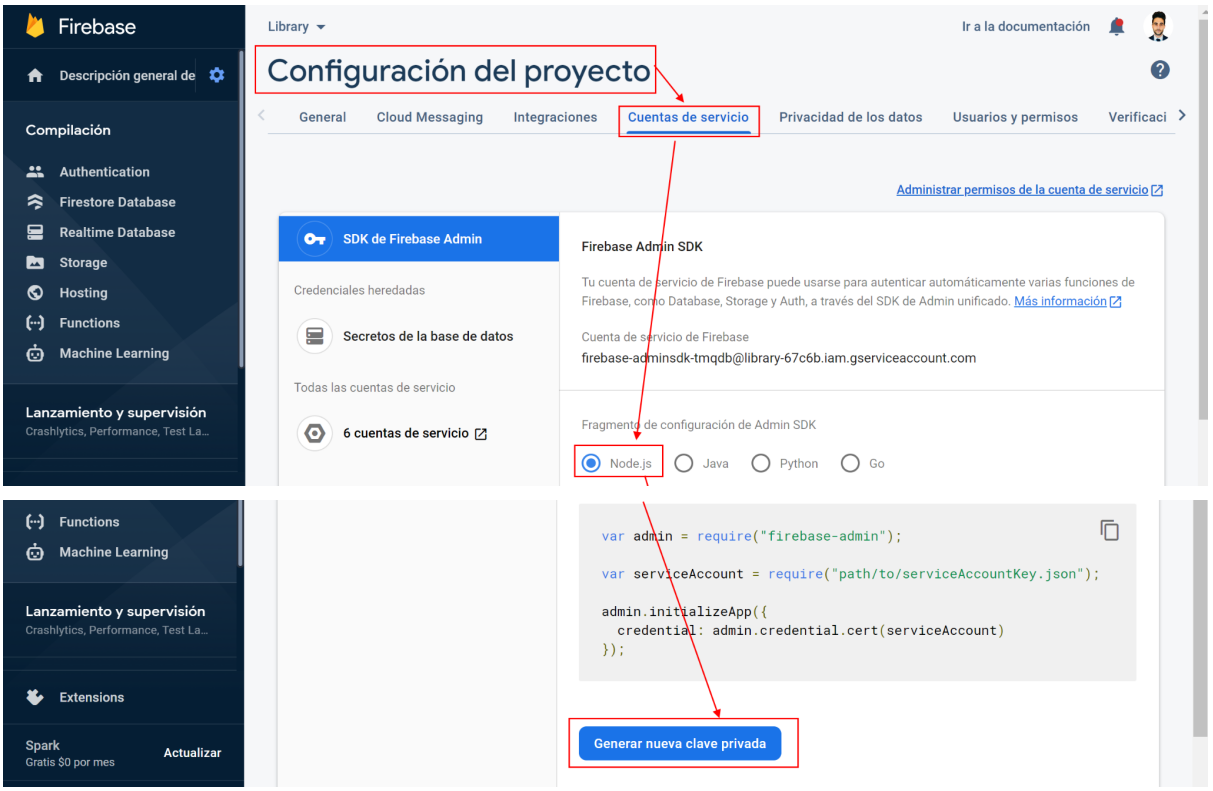
- 6.5. `npm install firebase @angular/fire --save`

- 6.6. modificaremos el archivo `src/app/app.module.ts` y agregaremos dos nuevos imports

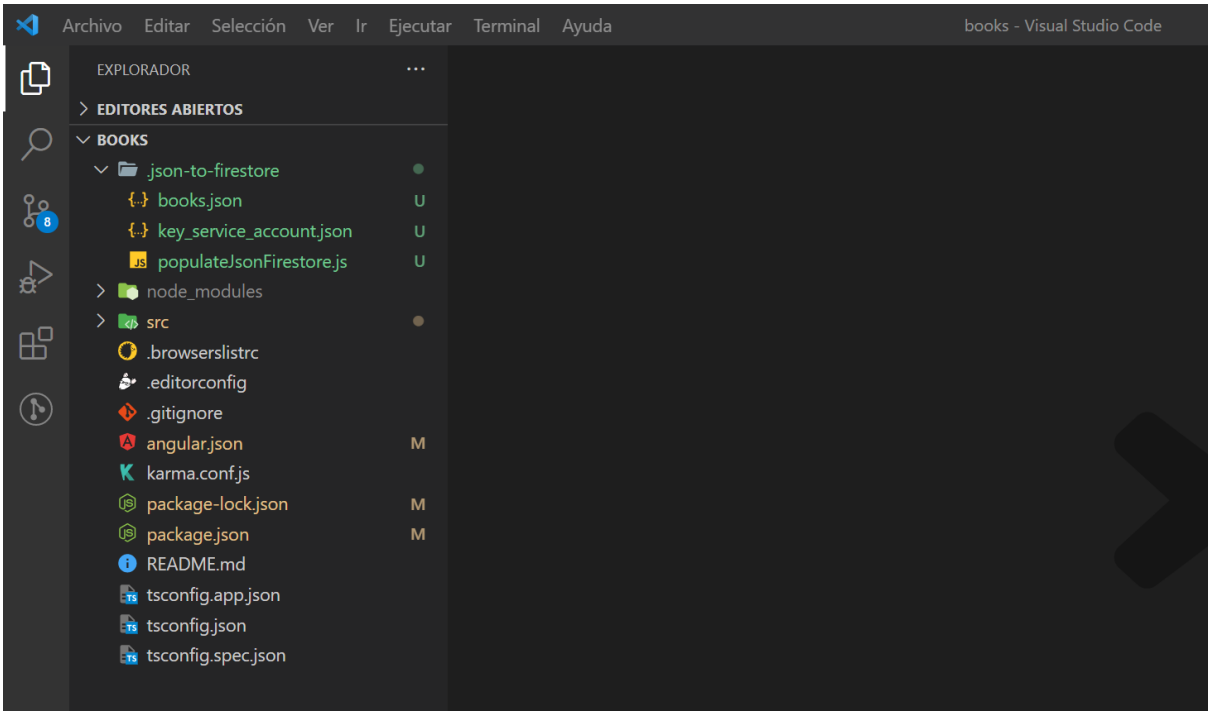
```
import { AngularFireModule } from '@angular/fire/compat';
import { environment } from 'src/environments/environment';
```

```
imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    BrowserAnimationsModule,
    AngularMaterialModule,
    AngularFireModule.initializeApp(environment.firebase)
  ],
```

- 6.EXTRA… Vamos a subir un archivo .json a Firestore con toda nuestra data ya preparada



- Este archivo que acabamos de descargar lo renombramos como “key_service_account”.
- Creamos una carpeta en el RootFolder del proyecto, y esta vez no hablamos de la carpeta “app”, si no de en el primer nivel superior nada más abrir literalmente la carpeta, en el mismo nivel que los “node_modules”, por ejemplo.
- En ella crearemos a su vez 3 archivos, siendo el “key_service_account” uno de ellos:



- En el books.json tenemos los siguientes datos de ejemplo:


```
{
  "0": {
    "id": "0",
    "author": "Grant Cardone",
    "dateadded": "2017-8-9",
    "dateread": "2017-8-27",
    "description": "The secret to extraordinary success is to put in 10
times the relevant effort than most people, and to condition your mind
for the success. You also have to recognize that with the increased
efforts, increased obstacles will confront you, and you have to work
your way around and through them.\n\nMega success is not possible
without overexposure, because you have to be everywhere at the same
time, so people recognize your name, brand and logo.\n\nAccording to
The 10X Rule by Grant Cardone, the secret to extraordinary success is
to put in 10 times the relevant effort than most people, and to
condition your mind for the success. You also have to recognize that
with the increased efforts, increased obstacles will confront you, and
you have to work your way around and through them.\n\nThe book gives
you ideas on how to grow a business fast.",
    "imageUrl":
"http://cdn2.btrstatic.com/pics/showpics/large/261663_pXyAOnzs.jpg",
    "price": 16.96,
    "rate": 3.9,
    "title": "The 10X Rule"
```

```
  },
  "1": {
    "id": "1",
    "author": "Peter Thiel",
    "dateadded": "2017-8-22",
    "imageUrl":
"https://aydenjacobdotcom4.files.wordpress.com/2016/02/zero_to_one.jpg?
w=400",
    "price": 18.64,
    "title": "Zero to One"
  },
  "2": {
    "id": "2",
    "author": "Ashlee Vance",
    "dateadded": "2017-8-24",
    "dateread": "2017-8-25",
    "description": "In the spirit of Steve Jobs and Moneyball, Elon
Musk is both an illuminating and authorized look at the extraordinary
life of one of Silicon Valley's most exciting, unpredictable, and
ambitious entrepreneurs--a real-life Tony Stark--and a fascinating
exploration of the renewal of American invention and its new
\"makers.\"\nElon Musk spotlights the technology and vision of Elon
Musk, the renowned entrepreneur and innovator behind SpaceX, Tesla, and
SolarCity, who sold one of his Internet companies, PayPal, for $1.5
billion. Ashlee Vance captures the full spectacle and arc of the
genius's life and work, from his tumultuous upbringing in South Africa
and flight to the United States to his dramatic technical innovations
and entrepreneurial pursuits.\nVance uses Musk's story to explore one
of the pressing questions of our age: can the nation of inventors and
creators who led the modern world for a century still compete in an age
of fierce global competition? He argues that Musk--one of the most
unusual and striking figures in American business history--is a
contemporary, visionary amalgam of legendary inventors and
industrialists including Thomas Edison, Henry Ford, Howard Hughes, and
Steve Jobs. More than any other entrepreneur today, Musk has dedicated
his energies and his own vast fortune to inventing a future that is as
rich and far-reaching as the visionaries of the golden age of
science-fiction fantasy.\nThorough and insightful, Elon Musk brings to
life a technology industry that is rapidly and dramatically changing by
examining the life of one of its most powerful and influential
titans.",
    "imageUrl":
"https://d.allegroimg.com/s400/014968/a4d959604fbbab9d1807eae5d57d",
```

```
    "price": 11.55,
    "rate": 4.8,
    "title": "Elon Musk"
  },
  "3": {
    "id": "3",
    "author": "Eric Schmidt & Jonathan Rosenberg",
    "dateadded": "2017-8-24",
    "imageUrl":
"https://static.businessinsider.com/image/551d80c4ecad049b13c2b3a5-400/
image.jpg",
    "price": 15.67,
    "title": "How google works"
  },
  "4": {
    "id": "4",
    "author": "Grant Cardone",
    "dateadded": "2017-8-1",
    "dateread": "2017-8-25",
    "description": "Sell or Be Sold is based on the premise that every
day we are selling: the child who wants a piece of candy and needs to
convince his/her parents to comply, the teacher to the student on why
learning trigonometry is important and the sales professional who
closes the deal with a prospect. With this in mind, the ability to
communicate, persuade, negotiate and close a deal is important
regardless of your profession. Grant takes you from amateur to
professional by explaining the key differences at each stage, how to
make the transition from average to great and provides perspectives
needed to help you be successful at “selling” in your chosen endeavor",
    "imageUrl":
"http://cdn2.btrstatic.com/pics/showpics/large/261663_nqOVVNea.png",
    "price": 19.96,
    "rate": 4.9,
    "title": "Sell or Be Sold"
  },
  "5": {
    "id": "5",
    "author": "Carol Dweck",
    "dateadded": "2017-8-27",
    "imageUrl":
"http://cdn2.btrstatic.com/pics/showpics/large/388609_6U6DDHpQ.jpg",
    "price": 11.99,
    "title": "Mindset"
```

```
  }
}
```

- Y en el archivo de populateJsonFirestore.js ponemos lo siguiente:

```
const admin = require('firebase-admin');
const serviceAccount = require("./key_service_account.json");
const data = require("./books.json");
const collectionKey = "book-list"; //Name of the collection

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount)
});

const firestore = admin.firestore();
const settings = {timestampsInSnapshots: true};

firestore.settings(settings);

if (data && (typeof data === "object")) {
  Object.keys(data).forEach(docKey => {

firestore.collection(collectionKey).doc(docKey).set(data[docKey]).then(
(res) => {
      console.log("Document " + docKey + " successfully written!");

    }).catch((error) => {
      console.error("Error writing document: ", error);
    });

  });
}
```

Nota:
Para evitar algún posible error en paso final con el comando de la subida de datos, vamos a instalar de antemano el “firebase-admin”

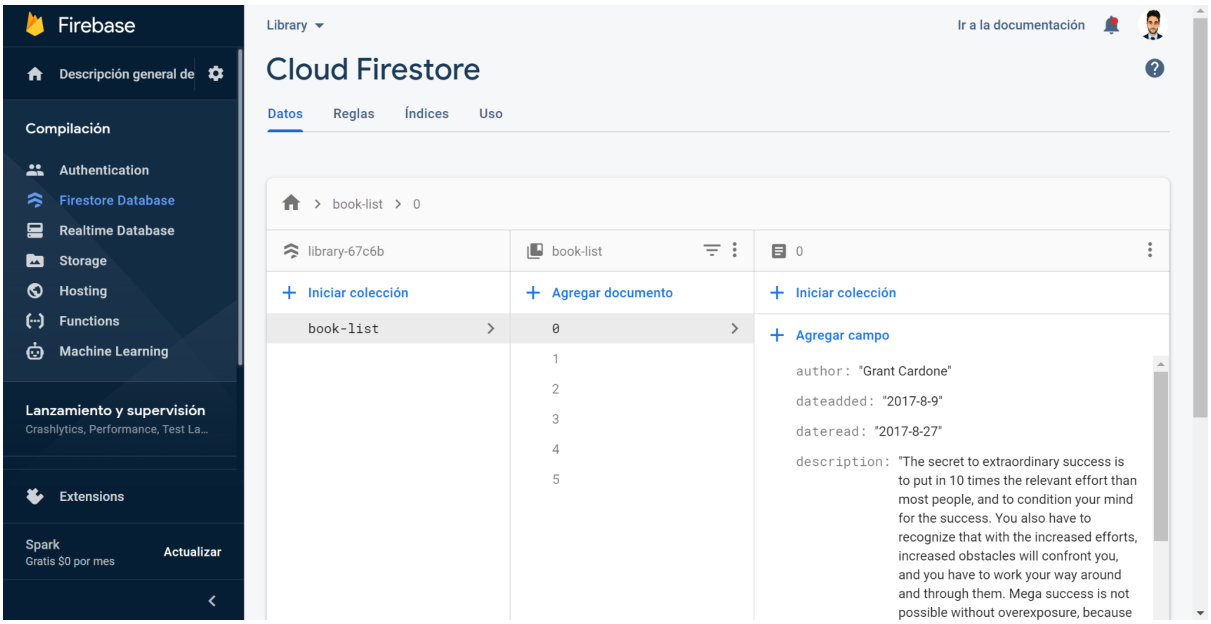
npm install --save firebase-admin

- Por último, para subir finalmente el json a Firestore, abrimos una nueva terminal, con la que primero accedemos a la carpeta de “.json-to-firestore”, y ejecutamos el comando

  node populateJsonFirestore.js

- Podemos observar que los datos del Json se han subido satisfactoriamente, y que en Firestore ya aparecen

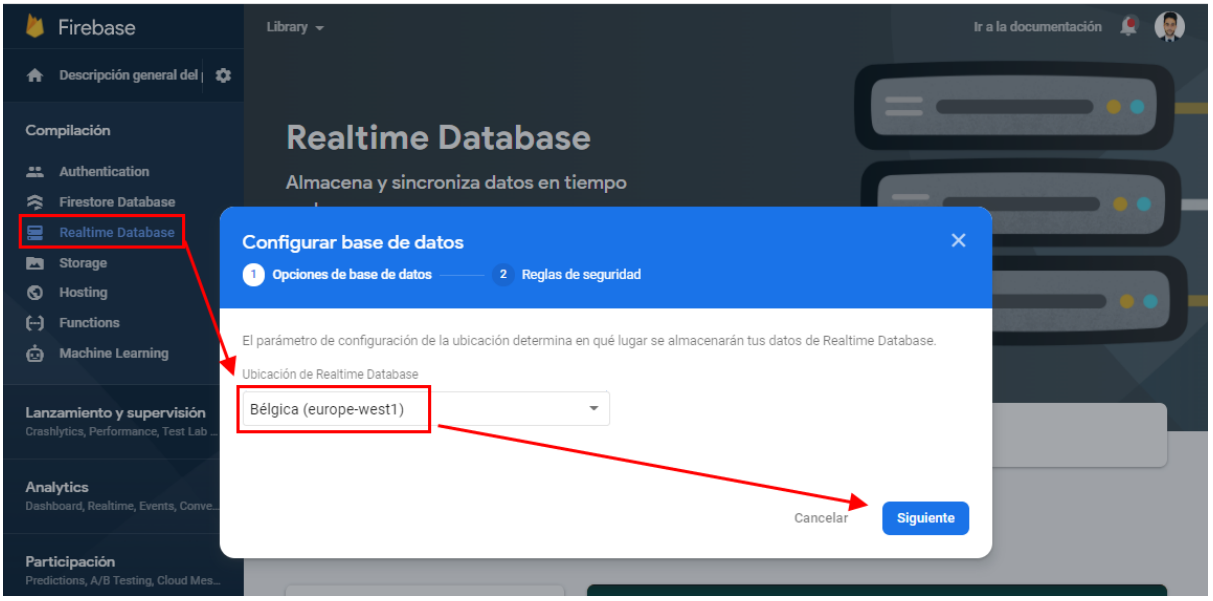
```
PS C:\Users\Sergio DC\Desktop\books> cd .json-to-firestore
PS C:\Users\Sergio DC\Desktop\books\.json-to-firestore> node populateJsonFirestore_v2.js
Document 0 successfully written!
Document 2 successfully written!
Document 4 successfully written!
Document 3 successfully written!
Document 5 successfully written!
Document 1 successfully written!
PS C:\Users\Sergio DC\Desktop\books\.json-to-firestore>
```

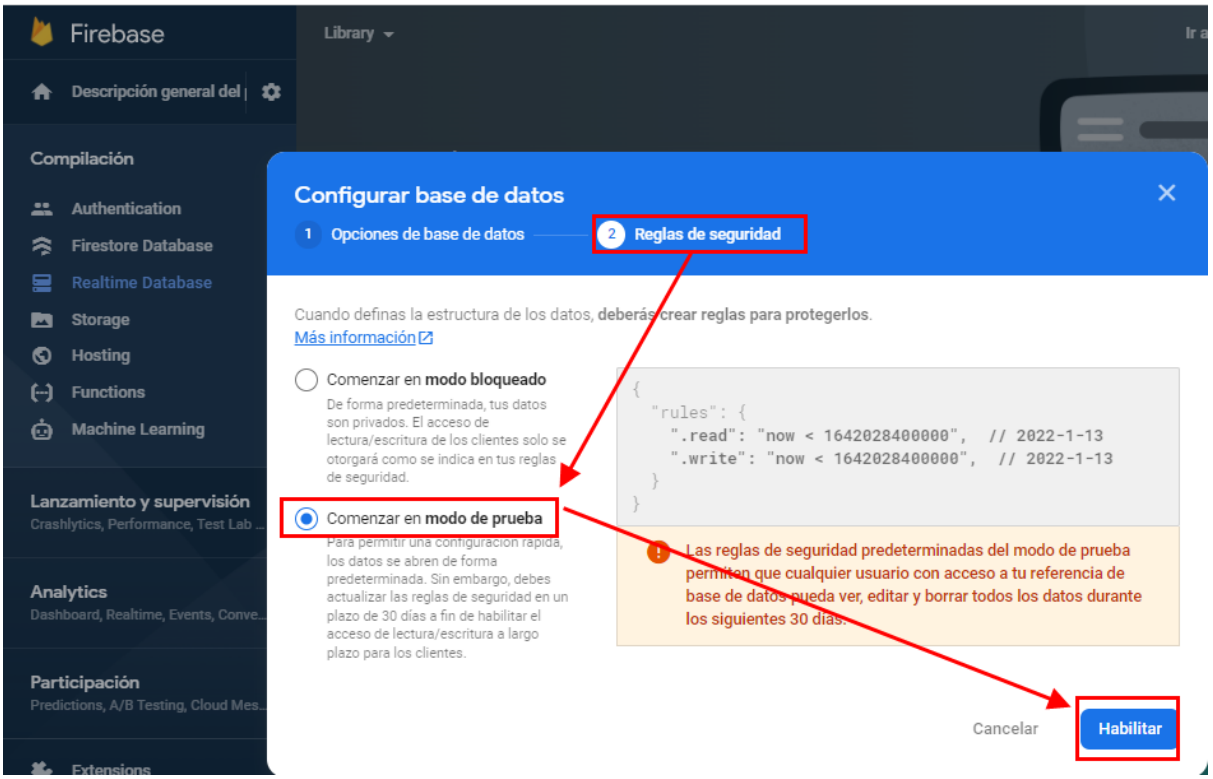


- 6.7. Añadir el módulo de Firestore (y de la database, por si acaso) al app.module.ts

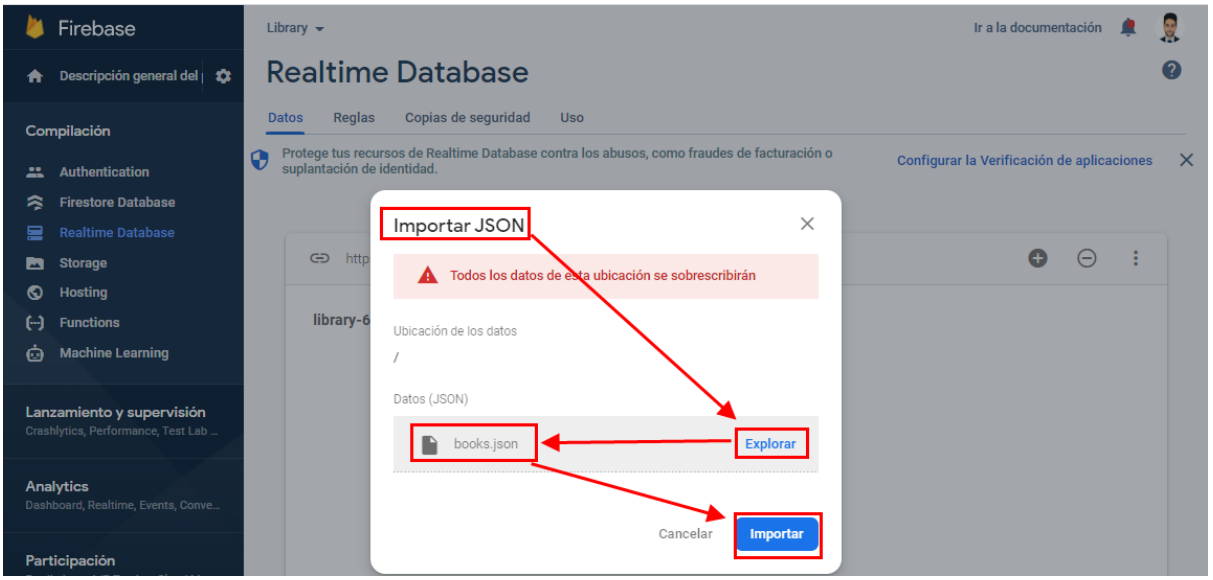
```
import { AngularFireModule } from '@angular/fire/compat';
import { environment } from 'src/environments/environment';
import { AngularFirestoreModule } from
'@angular/fire/compat/firestore';
imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    BrowserAnimationsModule,
    AngularMaterialModule,
    AngularFireModule.initializeApp(environment.firebase, 'library'),
// optionally provide a custom firebase application name
    AngularFirestoreModule,
  ],
```

- 6.8. Ahora, vamos a preparar un segundo camino para almacenar nuestros libros, utilizando este vez la RealTime DataBase





- 6.9. Subimos nuestro books.json a la RealTime DataBase



- 6.9. Mejor vamos a importar el books.json ORIGINAL

https://raw.githubusercontent.com/etrupja/BookNotes/master/books.json

- 6.10. No olvidemos volver al archivo de enviroments.ts para añadir la databaseURL

```
export const environment = {
  production: false,
  firebase: {
    apiKey: "yourApiKey",
    authDomain: "yourAuthDomain",
    databaseURL: "yourDatabaseURL",
    projectId: "yourProjectId",
    storageBucket: "yourStorageBucket",
    messagingSenderId: "yourMessagingSenderId",
    appId: "yourAppId"
  }
};
```

- 6.11. Ahora vamos a volver al app.module.ts para añadir los imports de la RealTime DataBase

```
import { AngularFireModule } from '@angular/fire/compat';
import { environment } from 'src/environments/environment';

import { AngularFirestoreModule } from
'@angular/fire/compat/firestore';

import { AngularFireDatabaseModule } from
'@angular/fire/compat/database';
```

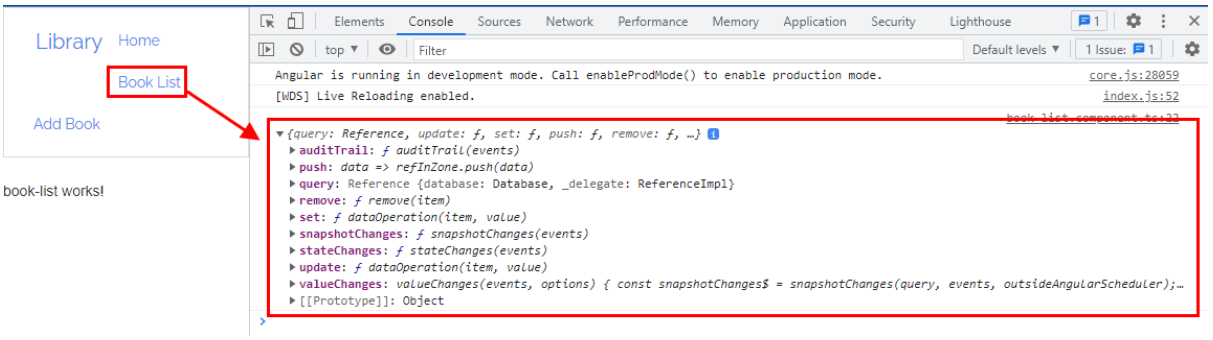
```
imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    BrowserAnimationsModule,
    AngularMaterialModule,
    AngularFireModule.initializeApp(environment.firebase, 'library'),
// optionally provide a custom firebase application name
    AngularFirestoreModule,
    AngularFireDatabaseModule
  ],
```

7. LIST of books FROM FIREBASE

- 7.1. Vamos al book-list.component.ts para…

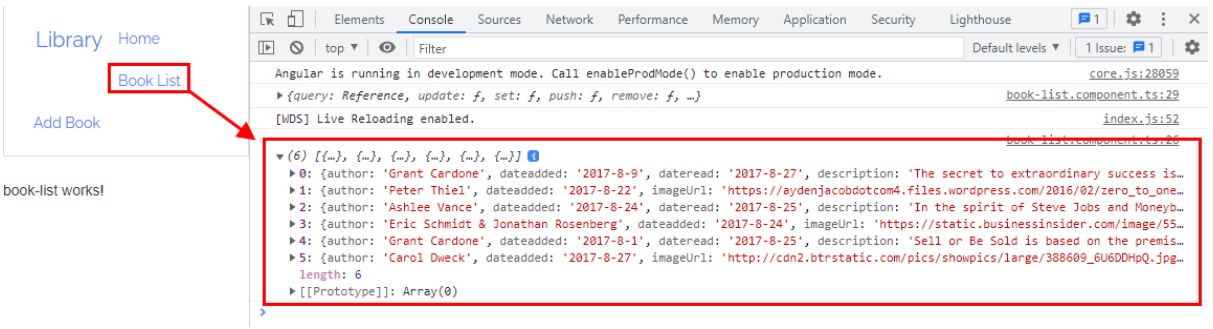
```
import { Component, OnInit } from '@angular/core';

import { AngularFirestore, AngularFirestoreCollection,
AngularFirestoreCollectionGroup, AngularFirestoreDocument } from
'@angular/fire/compat/firestore';

import { AngularFireDatabase, AngularFireList, AngularFireObject,
AngularFireAction } from '@angular/fire/compat/database';

@Component({
  selector: 'app-book-list',
  templateUrl: './book-list.component.html',
  styleUrls: ['./book-list.component.css']
})
export class BookListComponent implements OnInit {

  books: AngularFireList<any[]> | undefined;

  constructor(
    private angularFirestore: AngularFirestore,
    private angularFireDatabase: AngularFireDatabase
  ) {
    this.books = angularFireDatabase.list('/books');

    console.log(this.books);
```

```
  }

  ngOnInit(): void {

  }

}
```



- 7.2. Pero la cosa es que queremos llegar a ver el array con los libros, y cada libro con sus propiedades…

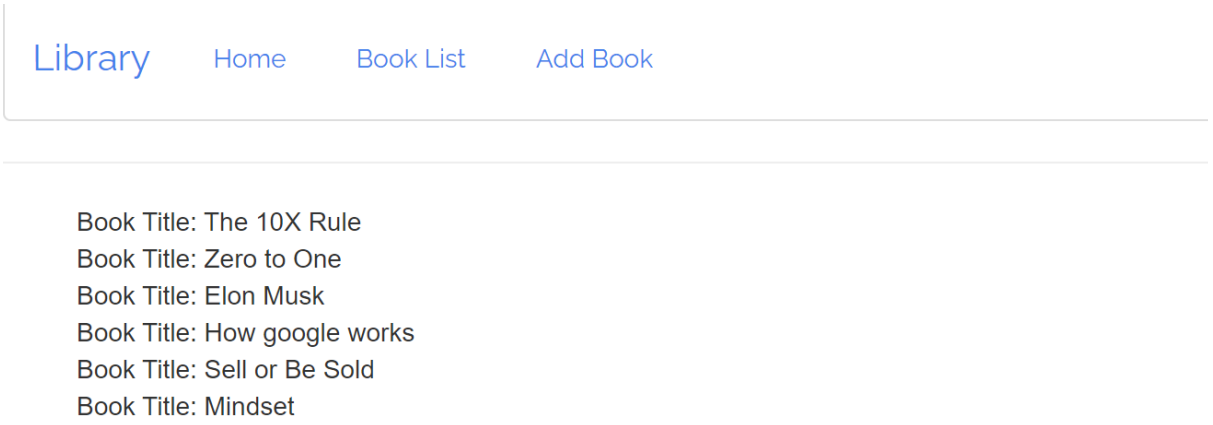
```
export class BookListComponent implements OnInit {

  books: AngularFireList<any[]> | undefined;
  allBooks: any;

  constructor(
    private angularFirestore: AngularFirestore,
    private angularFireDatabase: AngularFireDatabase
  ) {
    this.books = angularFireDatabase.list('/books');

    this.books.valueChanges().subscribe((books: any) => {
      this.allBooks = books;
      console.log(this.allBooks);
    })

    console.log(this.books);
  }

  ngOnInit(): void { }

}
```


Library
Home
Book List
Add Book
book-list works!
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
top
Filter
Default levels
1 Issue: 1
Angular is running in development mode. Call enableProdMode() to enable production mode.
core.js:28059
{query: Reference, update: ƒ, set: ƒ, push: ƒ, remove: ƒ, …}
book-list.component.ts:29
[WDS] Live Reloading enabled.
index.js:52
(6) [{…}, {…}, {…}, {…}, {…}, {…}]
0: {author: 'Grant Cardone', dateadded: '2017-8-9', dateread: '2017-8-27', description: 'The secret to extraordinary success is…
1: {author: 'Peter Thiel', dateadded: '2017-8-22', imageUrl: 'https://aydenjacobdotcom4.files.wordpress.com/2016/02/zero_to_one…
2: {author: 'Ashlee Vance', dateadded: '2017-8-24', dateread: '2017-8-25', description: 'In the spirit of Steve Jobs and Moneyb…
3: {author: 'Eric Schmidt & Jonathan Rosenberg', dateadded: '2017-8-24', imageUrl: 'https://static.businessinsider.com/image/55…
4: {author: 'Grant Cardone', dateadded: '2017-8-1', dateread: '2017-8-25', description: 'Sell or Be Sold is based on the premis…
5: {author: 'Carol Dweck', dateadded: '2017-8-27', imageUrl: 'http://cdn2.btrstatic.com/pics/showpics/large/388609_6U6DDHpQ.jpg…
length: 6
[[Prototype]]: Array(0)

8. ANGULAR SERVICE

Para comprobar que lo que llevamos de momento funciona correctamente, hacemos una pequeña prueba en el book-list.component.html

```
<!-- <p>book-list works!</p> -->

<ul>
    <li *ngFor="let book of allBooks">Book Title: {{ book.title }}</li>
</ul>
```



- 8.1. Creamos una carpeta para los servicios y creamos dentro de ella el propio servicio

  ng g s services/realTimeDB

- 8.2. Vamos al app.module.ts para importar como Provider nuestro nuevo servicio

```
providers: [
    RealTimeDBService
  ],
```

- 8.3. Ahora en nuestro realTimeDBService.ts

```
import { Injectable } from '@angular/core';
import { AngularFireDatabase, AngularFireList, AngularFireObject,
AngularFireAction } from '@angular/fire/compat/database';

@Injectable({
  providedIn: 'root'
})
```

```
export class RealTimeDBService {

  books: AngularFireList<any[]> | undefined; // from firebase

  constructor(
    private angularFireDatabse: AngularFireDatabase
  ) { }

  getBooks() {
    this.books = this.angularFireDatabse.list('/books') as
AngularFireList<any[]>;

    return this.books;
  }
}
```

- 8.4. Volviendo a book-list.component.ts, ya podemos eliminar la propiedad de books, las inyecciones en el constructor, así como todo su contenido, para sustituirlo por:

```
export class BookListComponent implements OnInit {

  allBooks: any;

  constructor(
    private realTimeDBService: RealTimeDBService
  ) { }

  ngOnInit(): void {
    this.realTimeDBService.getBooks().valueChanges()
      .subscribe((books) => {
        this.allBooks = books;
      })
  }

}
```

- 8.5. Si volvemos al navegador, podemos comprobar que con el <li *ngFor> que teníamos ya en el html, se siguen viendo los títulos de los libros

9. FAVORITE BOOKS

En esta parte, vamos a empezar a construir el Home page, el cual tendrá principalmente dos secciones: los libros favoritos, y los libros no leídos. En esta parte trataremos los libros favoritos.

- 9.1. Vamos al RealTimeDatabaseService.ts para añadir una propiedad para los favouriteBooks, y un método que obtenga los libros favoritos:

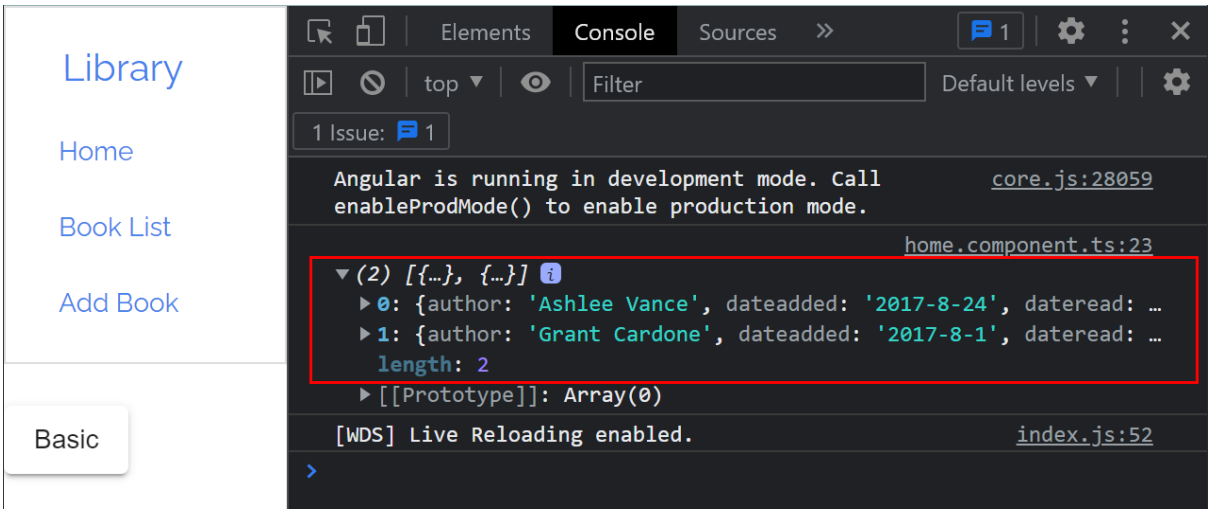
```
import { AngularFireDatabase, AngularFireList, AngularFireObject,
AngularFireAction } from '@angular/fire/compat/database';
import { Observable } from 'rxjs/';
import { map } from 'rxjs/operators';
```

```
export class RealTimeDBService {

  books: AngularFireList<any[]> | undefined; // from firebase
  favouriteBooks: Observable<any> | undefined;

  constructor(
    private angularFireDatabse: AngularFireDatabase
  ) { }

  getBooks() {
    this.books = this.angularFireDatabse.list('/books') as
AngularFireList<any[]>;

    return this.books;
  }

  getFavouriteBooks() {
    this.favouriteBooks = this.angularFireDatabse.list('/books')
      .valueChanges().pipe(map((books) => {
        const topRatedBooks = books.filter((item: any) => item.rate >
4);

        return topRatedBooks;
      }))

      return this.favouriteBooks;
  }
}
```

- 9.2. Y ahora, vamos al home.component.ts para inyectar nuestro servicio, y creamos aquí también la propiedad de favouriteBooks, para posteriormente, dentro del ngOnInit(), subscribirnos al método de obtener los libros favoritos de nuestro servicio:

```
export class HomeComponent implements OnInit {

  // favourite books
  favouriteBooks: any;

  constructor(
    private realTimeDatabaseService: RealTimeDBService
  ) { }

  ngOnInit(): void {
    this.realTimeDatabaseService.getFavouriteBooks()
      .subscribe((favBooks) => {
        this.favouriteBooks = favBooks;
        console.log(favBooks);
      })
  }

}
```

- 9.3. Si vamos al navegador y vamos al Home, a través del inspeccionar podemos confirmar que se han obtenido los dos libros con una Rate superior a 4 puntos



10. UNREAD BOOKS

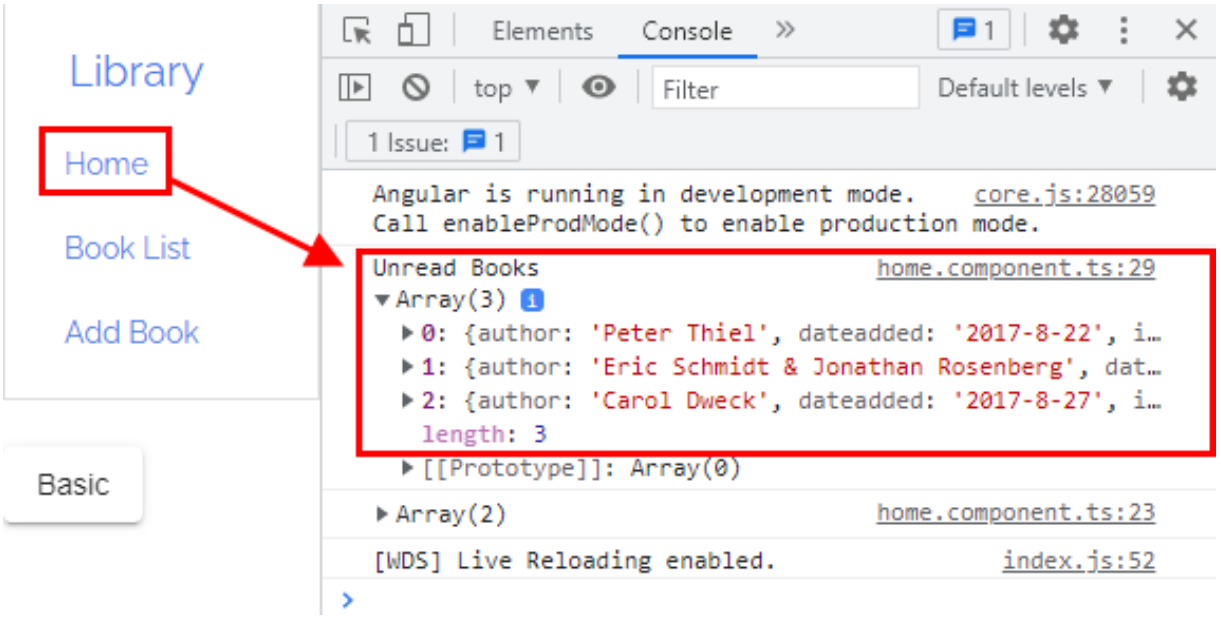
- 10.1. Vamos al RealTimeDBService para repetir los mismos pasos que para obtener los libros favoritos

```
export class RealTimeDBService {

  books: AngularFireList<any[]> | undefined; // from firebase
  favouriteBooks: Observable<any> | undefined;
  unreadBooks: Observable<any> | undefined;

  constructor(
    private angularFireDatabse: AngularFireDatabase
  ) { }

  getBooks() {
    this.books = this.angularFireDatabse.list('/books') as
AngularFireList<any[]>;
    return this.books;
  }

  getFavouriteBooks() {
    this.favouriteBooks = this.angularFireDatabse.list('/books')
      .valueChanges().pipe(map((books) => {
        const topRatedBooks = books.filter((item: any) => item.rate >
4);
        return topRatedBooks;
      }))
      return this.favouriteBooks;
  }

  getUnreadBooks() {
    this.unreadBooks = this.angularFireDatabse.list('/books')
      .valueChanges().pipe(map((books) => {
        const unreadBooks = books.filter((item: any) => item.dateread
== null);
        return unreadBooks;
      }))
      return this.unreadBooks;
  }
}
```

- 10.2. Volviendo al home.component.ts …

```
export class HomeComponent implements OnInit {

  favouriteBooks: any;
  unreadBooks: any;

  constructor(
    private realTimeDatabaseService: RealTimeDBService
  ) { }

  ngOnInit(): void {
    this.realTimeDatabaseService.getFavouriteBooks()
      .subscribe((favBooks) => {
        this.favouriteBooks = favBooks;
        console.log(this.favouriteBooks);
      })

    this.realTimeDatabaseService.getUnreadBooks()
      .subscribe((noReadedBooks) => {
        this.unreadBooks = noReadedBooks;
        console.log('Unread Books', this.unreadBooks);
      })
  }
}
```

- 10.3. Comprobamos en el inspeccionar del navegador que ha funcionado



11. HOME VIEW

- 11.1. Vamos a ir importando en nuestro angular-material.module.ts, todos los componentes de Angular Material que prevemos que vamos a usar.

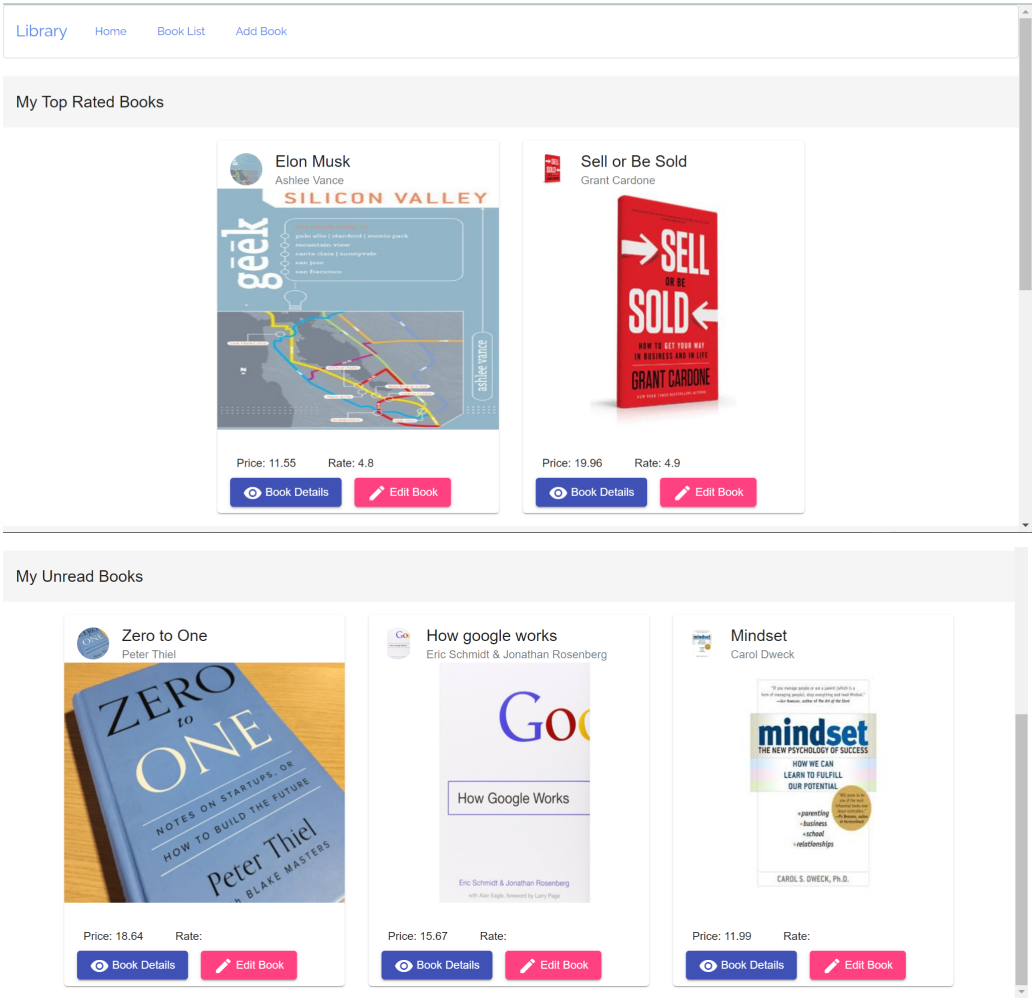
  https://material.angular.io/components/categories

```
import { MatButtonModule } from '@angular/material/button';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatCardModule } from '@angular/material/card';
import { MatGridListModule } from '@angular/material/grid-list';
import { MatInputModule } from '@angular/material/input';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatNativeDateModule } from '@angular/material/core';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatProgressSpinnerModule } from
'@angular/material/progress-spinner';
import { MatTabsModule } from '@angular/material/tabs';
import { MatListModule } from '@angular/material/list';
import { MatIconModule } from '@angular/material/icon';

@NgModule({
  declarations: [],
  imports: [MatButtonModule, MatCheckboxModule, MatCardModule,
MatGridListModule, MatInputModule, MatDatepickerModule,
MatNativeDateModule, MatToolbarModule, MatProgressSpinnerModule,
MatTabsModule, MatListModule, MatIconModule],
  exports: [MatButtonModule, MatCheckboxModule, MatCardModule,
MatGridListModule, MatInputModule, MatDatepickerModule,
MatNativeDateModule, MatToolbarModule, MatProgressSpinnerModule,
MatTabsModule, MatListModule, MatIconModule]
})
```

- 11.2. Vamos al home.component.html para desarrollar la vista con Angular Material

```
<!-- <p>home works!</p> -->

<mat-toolbar>My Top Rated Books | RealTimeDB</mat-toolbar>
<div class="superContainer" *ngIf=" favouriteBooks != null ||
favouriteBooks != undefined">
    <mat-card class="example-card" *ngFor="let book of favouriteBooks">
      <mat-card-header>
        <img mat-card-avatar src="{{ book.imageUrl }}">
        <mat-card-title>{{ book.title }}</mat-card-title>
        <mat-card-subtitle>{{ book.author }}</mat-card-subtitle>
      </mat-card-header>

      <a [routerLink]="['/book-details/'+book.$key]">
        <img class="book-img" mat-card-image src="{{ book.imageUrl }}">
      </a>

      <mat-card-actions>
          <button mat-button>Price: {{ book.price }}</button>
          <button mat-button>Rate: {{ book.rate }}</button>
          <button mat-raised-button color="primary"
[routerLink]="['/book-details/'+book.$key]">
            <i class="material-icons">visibility</i> Book Details
          </button>
          <button mat-raised-button color="accent"
[routerLink]="['/edit-book/'+book.$key]">
              <i class="material-icons">mode_edit</i> Edit Book
          </button>
      </mat-card-actions>
    </mat-card>
</div>

<mat-toolbar>My Unread Books | RealTimeDB</mat-toolbar>
<div class="superContainer" *ngIf=" unreadBooks != null || unreadBooks
!= undefined">
    <mat-card class="example-card" *ngFor="let book of unreadBooks">
      <mat-card-header>
        <img mat-card-avatar src="{{ book.imageUrl }}">
        <mat-card-title>{{ book.title }}</mat-card-title>
        <mat-card-subtitle>{{ book.author }}</mat-card-subtitle>
      </mat-card-header>
```

```
      <a [routerLink]="['/book-details/'+book.$key]">
        <img class="book-img" mat-card-image src="{{ book.imageUrl }}">
      </a>

      <mat-card-actions>
          <button mat-button>Price: {{ book.price }}</button>
          <button mat-button>Rate: {{ book.rate }}</button>
          <button mat-raised-button color="primary"
[routerLink]="['/book-details/'+book.$key]">
            <i class="material-icons">visibility</i> Book Details
        </button>
        <button mat-raised-button color="accent"
[routerLink]="['/edit-book/'+book.$key]">
            <i class="material-icons">mode_edit</i> Edit Book
        </button>
      </mat-card-actions>
    </mat-card>
</div>
```



Ahora vamos a volver a recibir estos datos, pero esta vez, desde Firestore

- 11.3. Vamos a crear un servicio aparte para Firestore

  ng g s services/firestore

- 11.4. Creamos una interfaz para el libro

  ng g interface interfaces/book

```
export interface Book {
    $key?: string;
    author?: string;
    title?: string;
    price?: number;
    dateadded?:  Date;
    // isRead?:boolean;
    dateread?: Date;
    description?: string;
    rate?: number;
    imageUrl?: string;
}
```

- 11.5. Una vez en nuestro firestore.service.ts

```
import { Injectable } from '@angular/core';
import { AngularFirestore, AngularFirestoreDocument,
AngularFirestoreCollection, AngularFirestoreCollectionGroup } from
'@angular/fire/compat/firestore';
import { Book } from '../interfaces/book';

@Injectable({
  providedIn: 'root'
})

export class FirestoreService {

  constructor(
    private angularFirestore: AngularFirestore
  ) {
  }

  getBooks() {
    return
this.angularFirestore.collection('book-list').snapshotChanges();
  }
```

```
  addBook(payload: Book) {
    return this.angularFirestore.collection('book-list').add(payload);
  }

  updateBook(bookId: string, payload: Book) {
    return this.angularFirestore.doc('book-list/' +
bookId).update(payload);
  }

  deleteBook(bookId: string) {
    return this.angularFirestore.doc('book-list/' + bookId).delete();
  }

}
```

- 11.6. También pasamos por el app.module.ts ya que no debemos olvidar comprobar que las importaciones están correctas

```
import { AngularFireModule } from '@angular/fire/compat';
import { environment } from 'src/environments/environment';

import { AngularFirestoreModule } from
'@angular/fire/compat/firestore';

import { AngularFireDatabaseModule } from
'@angular/fire/compat/database';

import { FirestoreService } from './services/firestore.service';
```

Nota: (estas importaciones se añaden a las que ya teníamos)

```
imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    BrowserAnimationsModule,
    AngularMaterialModule,
    AngularFireModule.initializeApp(environment.firebase, 'library'),
// optionally provide a custom firebase application name
    AngularFirestoreModule,
    AngularFireDatabaseModule
  ],
```

```
  providers: [
    RealTimeDBService,
    FirestoreService
  ],
```

- 11.7. Y ahora, en el home.component.ts

```
import { FirestoreService } from '../../services/firestore.service';
import { Book } from 'src/app/interfaces/book';
import { FirestoreService } from '../../services/firestore.service';
```

```
export class HomeComponent implements OnInit {

  favouriteBooks: any;
  unreadBooks: any;

  public bookList: Book[] = [];

  constructor(
    private realTimeDatabaseService: RealTimeDBService,
    private firestoreService: FirestoreService
  ) { }

  ngOnInit(): void {
    this.realTimeDatabaseService.getFavouriteBooks()
      .subscribe((favBooks) => {
        this.favouriteBooks = favBooks;
        console.log(this.favouriteBooks);
      })

    this.realTimeDatabaseService.getUnreadBooks()
      .subscribe((noReadedBooks) => {
        this.unreadBooks = noReadedBooks;
        console.log('Unread Books', this.unreadBooks);
      })

    this.getBooks();
  }

  getBooks(): void {
    this.firestoreService.getBooks()
      .subscribe((res) => {
```

```
        this.bookList = res.map((book: any) => {
          return {
            ...book.payload.doc.data(),
            id: book.payload.doc.id
          } as Book;
        });
      });
  }
}
```

- 11.7. Si vamos al home.component.html y copiamos y pegamos una de nuestras estructuras que ya teníamos, y cambiamos el *ngIf y el *ngFor, vemos que los libros se vuelven a obtener y desplegar perfectamente, pero esta vez, desde Firestore.

```
<!--
-------------------------------------------------------------------------
------ -->

<mat-toolbar>My Books | Firestore</mat-toolbar>
<div class="superContainer" *ngIf=" bookList != null || bookList !=
undefined">
    <mat-card class="example-card" *ngFor="let book of bookList">
      <mat-card-header>
        <img mat-card-avatar src="{{ book.imageUrl }}">
        <mat-card-title>{{ book.title }}</mat-card-title>
        <mat-card-subtitle>{{ book.author }}</mat-card-subtitle>
      </mat-card-header>

      <a [routerLink]="['/book-details/'+book.$key]">
        <img class="book-img" mat-card-image src="{{ book.imageUrl }}">
      </a>

      <mat-card-actions>
          <button mat-button>Price: {{ book.price }}</button>
          <button mat-button>Rate: {{ book.rate }}</button>
          <button mat-raised-button color="primary"
[routerLink]="['/book-details/'+book.$key]">
            <i class="material-icons">visibility</i> Book Details
          </button>
          <button mat-raised-button color="accent"
[routerLink]="['/edit-book/'+book.$key]">
              <i class="material-icons">mode_edit</i> Edit Book
          </button>
```

```
      </mat-card-actions>
    </mat-card>
</div>
```

Nota: Al final de este proyecto, este componente queda obsoleto, ya que cuando hagamos el BookDetails, éste Home carecerá de sentido, y al final lo transformaremos completamente en un home tipo intro de recibimiento al usuario y presentación de la web.

12. ALL BOOKS VIEW (BOOK-LIST)

- 12.1. Primero vamos a ambos servicios, tanto el de Firestore como el de la realTimeDB para crear los respectivos métodos de origen de getBooks()

```
import { Injectable } from '@angular/core';
import { AngularFirestore, AngularFirestoreDocument,
AngularFirestoreCollection, AngularFirestoreCollectionGroup }
from '@angular/fire/compat/firestore';
import { Book } from '../interfaces/book';

@Injectable({
  providedIn: 'root'
})

export class FirestoreService {

  constructor(
    private angularFirestore: AngularFirestore
  ) {
  }

  public getBooks() {
    return
this.angularFirestore.collection('book-list').snapshotChanges();
  }

}
```

```
import { Injectable } from '@angular/core';
import { AngularFireDatabase, AngularFireList, AngularFireObject,
AngularFireAction } from '@angular/fire/compat/database';
import { Observable } from 'rxjs/';
import { map } from 'rxjs/operators';
import { Book } from '../interfaces/book';
import BookModel from '../models/book-model';

@Injectable({
  providedIn: 'root'
})
export class RealTimeDBService {
```

```
  private dbPath = '/books';

  books: AngularFireList<any[]> | undefined; // from firebase
  booksRef: AngularFireList<BookModel>;
  favouriteBooks: Observable<any> | undefined;
  unreadBooks: Observable<any> | undefined;
  bookDetails: AngularFireObject<any> | undefined; // from
firebase

  constructor(
    private angularFireDatabse: AngularFireDatabase
  ) {
    this.booksRef = angularFireDatabse.list(this.dbPath);
  }

  getBooks_v2(): AngularFireList<BookModel> {
    return this.booksRef;
  }

}
```

- 12.2. Vamos al book-list.component.ts para crear los métodos que nos permitirán recibir los objetos de libro tanto de Firestore como de la RealTimeDB, así también como las funciones para los botones que nos llevarán hacia el componente de BookDetails o del EditBook, para los cuales, necesitamos enviarles el objeto (libro) en cuestión desde esta lista de libros …

```
import { Component, OnInit } from '@angular/core';
import { NavigationExtras, Router } from '@angular/router';

import { map } from 'rxjs/operators';
import { Book } from 'src/app/interfaces/book';
import BookModel from 'src/app/models/book-model';

import { FirestoreService } from 'src/app/services/firestore.service';
import { RealTimeDBService } from
'src/app/services/real-time-db.service';
```

```
@Component({
  selector: 'app-book-list',
  templateUrl: './book-list.component.html',
  styleUrls: ['./book-list.component.css']
})
```

```
export class BookListComponent implements OnInit {

  allBooksRealTimeDB: any;
  allBooksRealTimeDB_v2?: BookModel[];
  allBooksFirestore: any;

  navigationExtras:NavigationExtras={
    state:{
      value:null
    }}

  constructor(
    private realTimeDBService: RealTimeDBService,
    private firestoreService: FirestoreService,
    private router: Router
  ) { }

  ngOnInit(): void {
    this.getAllBooks_v2(); // para obtener los libros de RealTimeDB
    this.getAllBooks(); // para obtener los libros de Firestore
  }

  getAllBooks(): void {
    this.firestoreService.getBooks()
      .subscribe((res) => {
        this.allBooksFirestore = res.map((book: any) => {
          return {
            data: book.payload.doc.data(),
            id: book.payload.doc.id
          } as Book;
        });
      });
      //console.log(this.allBooksFirestore);
  }

  getAllBooks_v2(): void {
    this.realTimeDBService.getBooks_v2().snapshotChanges().pipe(
      map((changes) =>
        changes.map((c) =>
          ({ key: c.payload.key, ...c.payload.val() })
        )
      )
    ).subscribe((data) => {
```

```
      this.allBooksRealTimeDB_v2 = data;
      //console.log(this.allBooksRealTimeDB_v2);
    });
    //console.log(this.allBooksRealTimeDB_v2);
  }

  getBookDetails(book: any){
    this.navigationExtras.state = book;
    console.log(book);
    this.router.navigate(["book-details"], this.navigationExtras)
  }

  getBookDetailsRTDB(book: any){
    this.navigationExtras.state = book;
    console.log(book);
    this.router.navigate(["book-details-rtdb"], this.navigationExtras)
  }

  getBookEdit(book: any){
    this.navigationExtras.state = book;
    console.log(book);
    this.router.navigate(["edit-book-fire"], this.navigationExtras)
  }

  getBookEditRTDB(book: any){
    this.navigationExtras.state = book;
    console.log(book);
    this.router.navigate(["edit-book-rtdb"], this.navigationExtras)
  }
}
```

- 12.3. Ahora, en el book-list.component.html, eliminamos el *ngFor que teníamos de antes a modo de prueba para poner en su lugar …

```
<!-- <p>book-list works!</p> -->

<mat-toolbar>My Books | Firestore</mat-toolbar>
<div class="superContainer" *ngIf="allBooksFirestore != null ||
allBooksFirestore != undefined">
    <mat-card class="example-card" *ngFor="let book of
allBooksFirestore">
      <mat-card-header>
        <img mat-card-avatar src="{{ book.data.imageUrl }}">
```

```
        <mat-card-title>{{ book.data.title }}</mat-card-title>
        <mat-card-subtitle>{{ book.data.author }}</mat-card-subtitle>
      </mat-card-header>

      <a (click)="getBookDetails(book)">
        <img class="book-img" mat-card-image src="{{ book.data.imageUrl
}}">
      </a>

      <mat-card-actions class="mat-card-actions">
          <button mat-button>Price: {{ book.data.price }}</button>
          <button mat-button>Rate: {{ book.data.rate }}</button>

          <button mat-raised-button color="primary"
(click)="getBookDetails(book)">
            <i class="material-icons">visibility</i> Book Details
          </button>
          <button mat-raised-button color="accent"
(click)="getBookEdit(book)">
              <i class="material-icons">mode_edit</i> Edit Book
          </button>
      </mat-card-actions>
    </mat-card>
</div>
<!--
------------------------------------------------------------------------
------ -->
<mat-toolbar>All Books || RealTimeDB</mat-toolbar>

<div class="superContainer">
    <!-- <mat-card class="example-card" *ngFor="let book of
allBooksRealTimeDB"> -->
      <mat-card class="example-card" *ngFor="let book of
allBooksRealTimeDB_v2">
      <mat-card-header>
        <img mat-card-avatar src="{{ book.imageUrl }}">
        <mat-card-title>{{ book.title }}</mat-card-title>
        <mat-card-subtitle>{{ book.author }}</mat-card-subtitle>
      </mat-card-header>

      <a (click)="getBookDetailsRTDB(book)">
        <img class="book-img" mat-card-image src="{{ book.imageUrl }}">
      </a>
```

```
      <mat-card-actions>
          <button mat-button>Price: {{ book.price }}</button>
          <button mat-button>Rate: {{ book.rate }}</button>

          <button mat-raised-button color="primary"
(click)="getBookDetailsRTDB(book)">
            <i class="material-icons">visibility</i> Book Details
          </button>
          <button mat-raised-button color="accent"
(click)="getBookEditRTDB(book)">
              <i class="material-icons">mode_edit</i> Edit Book
          </button>
        </mat-card-actions>
    </mat-card>
</div>
```

13. BOOK DETAILS

Comenzamos el desarrollo del controlador para los detalles del libro seleccionado.
En este momento, debemos de pasar todos los datos de un objeto a través de la ruta que conecta el componente del BookList con el componente del BookDetails.
Lo haremos gracias a NavigationExtras.

- 13.1. Empezamos por el book-details.component.ts para recibir tal objeto, y también crearemos los métodos que nos lleven a los componentes de EditBook y DeleteBook, con lo que tendremos que volver a pasar el mismo objeto que ya hemos recibido aquí en los detalles del libro …

```
import { Component, OnInit } from '@angular/core';
import { RealTimeDBService } from
'src/app/services/real-time-db.service';
import { ActivatedRoute, NavigationExtras, Router } from
'@angular/router';
import { FirestoreService } from 'src/app/services/firestore.service';

@Component({
  selector: 'app-book-details',
  templateUrl: './book-details.component.html',
  styleUrls: ['./book-details.component.css']
})
export class BookDetailsComponent implements OnInit {

  book: any | undefined;
  navigationExtras:NavigationExtras={
    state:{
      value:null
    }}
  currentNavigate: any;

  constructor(
    private realTimeDBService: RealTimeDBService,
    private firestoreService: FirestoreService,
    private router: Router,
    private activatedRoute: ActivatedRoute
  ) {
    //hay que hacer esto en el constructor porque si se hiciera todo en
el ngOnInit(), obtendríamos el objetivo null debido a que la navegación
muere al crearse
```

```
//https://stackoverflow.com/questions/54891110/router-getcurrentnavigation-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.book = navigation?.extras?.state;

    this.currentNavigate = navigation?.finalUrl?.root.children.primary.segments[0].path;
  }

  ngOnInit(): void {

    if (this.book == undefined){
      //asi controlo que nadie se intente colar
      this.router.navigate(["book-list"]);
    }

    console.log(this.book);
  }

  esFirestore() {
    if (this.currentNavigate == "book-details") {
      return true;
    }
    else if (this.currentNavigate == "book-details-rtdb") {
      return false;
    }

    // por si acaso
    return null;
  }

  getBookEditFire(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["edit-book-fire"], this.navigationExtras)
  }

  getBookEditRTDB(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["edit-book-rtdb"], this.navigationExtras)
  }

  getBookDeleteFire(book: any){
```

```
    this.navigationExtras.state = book;
    this.router.navigate(["delete-book-fire"], this.navigationExtras)
  }

  getBookDeleteRTDB(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["delete-book"], this.navigationExtras)
  }
}
```

- 12.2. Luego añadimos nuestros métodos en una estructura en el book-details.html

```
<!-- <p>book-details works!</p> -->

<div class="superContainer" *ngIf="esFirestore()">
    <mat-toolbar> Book Details | Firestore</mat-toolbar>

    <div class="book-details" *ngIf="book">
        <mat-card>
            <mat-card-title>
                <span class="mat-headline">{{book.data.title}}</span>
            </mat-card-title>
            <mat-card-content>
                <div class="left">
                    <img class="book-imageUrl"
src="{{book.data.imageUrl}}" />
                </div>
                <div class="right">
                    <mat-list>
                        <mat-list-item> <i
class="material-icons">person_pin</i> Author:  <b>
{{book.data.author}}</b>
                        </mat-list-item>
                        <mat-list-item> <i
class="material-icons">date_range</i> Date Added:  <b>
                                {{book.data.dateadded}}</b>
</mat-list-item>
                        <mat-list-item *ngIf="book.data.dateread"> <i
class="material-icons">date_range</i> Date
```

```
                            Read:  <b>
{{book.data.dateread}}</b> </mat-list-item>
                        <mat-list-item> <i
class="material-icons">attach_money</i> Price:  <b>
{{book.data.price}}</b>
                        </mat-list-item>
                        <mat-list-item *ngIf="book.data.rate"> <i
class="material-icons">star</i>

Rate:  <b>{{book.data.rate}}</b></mat-list-item>
                    </mat-list>
                </div>
                <hr>
                <mat-toolbar>Description</mat-toolbar>
                {{book.data.description}}
                <p *ngIf="!book.data.description"> No description for
this book! </p>
            </mat-card-content>
            <mat-card-actions>
                <button [routerLink]="['/book-list']"
mat-raised-button> <i class="material-icons">view_module</i> All
Books</button>
                <button mat-raised-button class="editButton"
(click)="getBookEditFire(book)"> <i
class="material-icons">mode_edit</i> Edit </button>
                <button mat-raised-button color="warn"
(click)="getBookDeleteFire(book)"> <i
class="material-icons">delete_sweep</i> Delete In Firestore</button>
                <button mat-raised-button color="warn"
(click)="getBookDeleteRTDB(book)"> <i
class="material-icons">delete_sweep</i> Delete In RealTimeDB</button>
            </mat-card-actions>
        </mat-card>
    </div>
</div>
<!--
-------------------------------------------------------------------------
------------------------------------------------------------ -->
<div class="superContainer" *ngIf="!esFirestore()">
    <mat-toolbar> Book Details | RealTimeDB</mat-toolbar>

    <div class="book-details" *ngIf="book">
        <mat-card>
```

```
            <mat-card-title>
                <span class="mat-headline">{{book.title}}</span>
            </mat-card-title>
            <mat-card-content>
                <div class="left">
                    <img class="book-imageUrl" src="{{book.imageUrl}}"
/>
                </div>
                <div class="right">
                    <mat-list>
                        <mat-list-item> <i
class="material-icons">person_pin</i> Author:  <b>
{{book.author}}</b>
                        </mat-list-item>
                        <mat-list-item> <i
class="material-icons">date_range</i> Date Added:  <b>
                                {{book.dateadded}}</b> </mat-list-item>
                        <mat-list-item *ngIf="book.dateread"> <i
class="material-icons">date_range</i> Date
                            Read:  <b> {{book.dateread}}</b>
</mat-list-item>
                        <mat-list-item> <i
class="material-icons">attach_money</i> Price:  <b>
{{book.price}}</b>
                        </mat-list-item>
                        <mat-list-item *ngIf="book.rate"> <i
class="material-icons">star</i>

Rate:  <b>{{book.rate}}</b></mat-list-item>
                    </mat-list>
                </div>
                <hr>
                <mat-toolbar>Description</mat-toolbar>
                {{book.description}}
                <p *ngIf="!book.description"> No description for this
book! </p>
            </mat-card-content>
            <mat-card-actions>
                <button [routerLink]="['/book-list']"
mat-raised-button> <i class="material-icons">view_module</i> All
Books</button>
```

```
                <button mat-raised-button class="editButton"
(click)="getBookEditRTDB(book)"> <i
class="material-icons">mode_edit</i> Edit </button>
                <button mat-raised-button color="warn"
(click)="getBookDeleteFire(book)"> <i
class="material-icons">delete_sweep</i> Delete In Firestore</button>
                <button mat-raised-button color="warn"
(click)="getBookDeleteRTDB(book)"> <i
class="material-icons">delete_sweep</i> Delete In RealTimeDB</button>
            </mat-card-actions>
        </mat-card>
    </div>
</div>
```

- 12.3. Y para su css definimos los siguientes estilos …

```
.left{float: left}
.right{float: left; padding: 25px;}

.superContainer {
    padding: 10px;
    padding-left: 30px;
    margin: 10px;
    display: flex;
    flex-wrap: wrap;
}

.book-details{
    margin: 0 auto;
    width: 60%;
    height: 100vh
}

.bookTile{padding:25px auto;}
.editButton{
    color :white !important;
    background-color: rgba(255, 222, 121, 0.96);
}

.detailsButton{
    color :white !important;
    background-color: rgb(114, 168, 255)
```

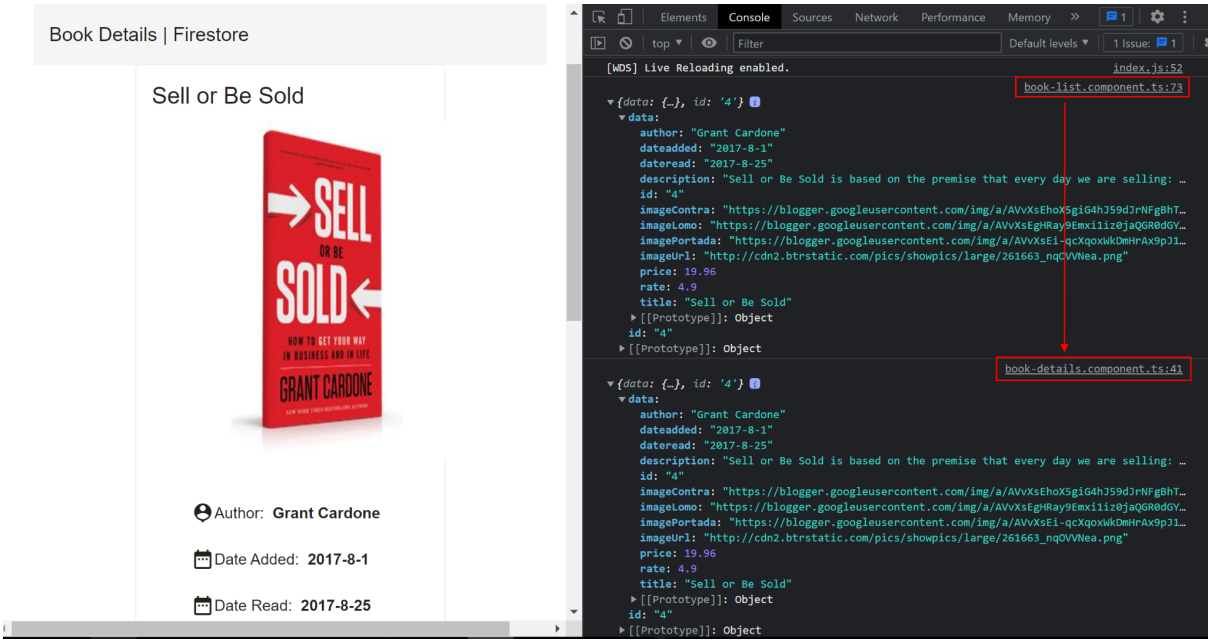
```
}

.book-imageUrl {
    width: 345px;
    height: 350px;
}
```

- 12.4. Y ya sólo tendríamos que asegurarnos de las rutas en el app-routing.module

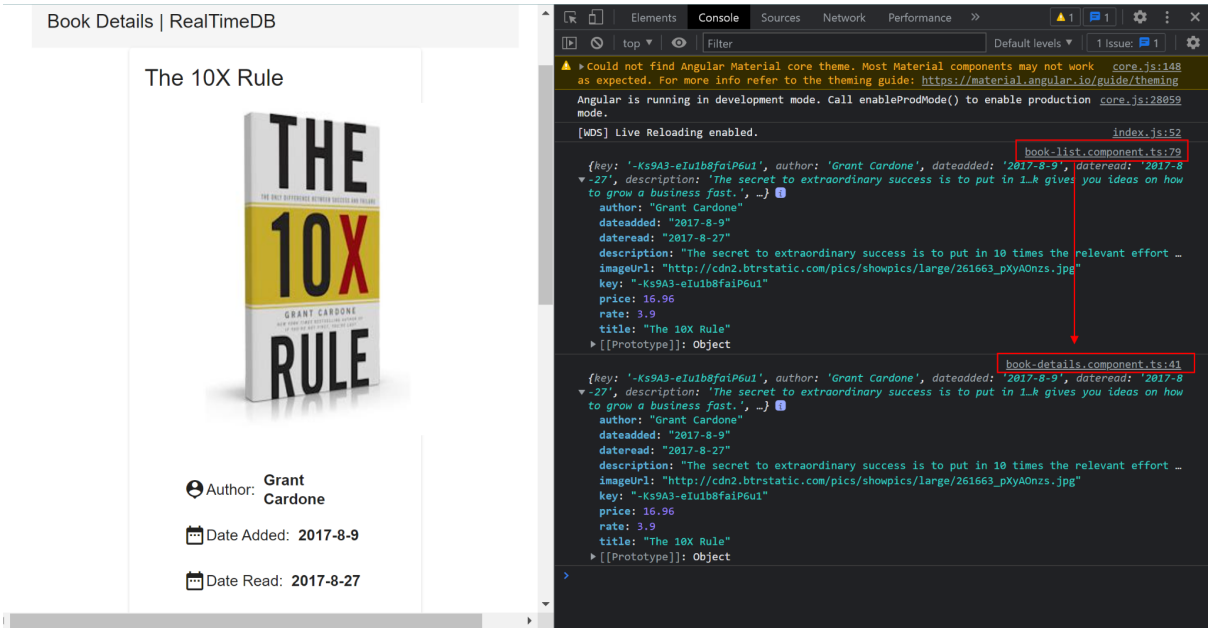
```
const routes: Routes = [
  {
    path: '', component: HomeComponent
  },
  {
    path: 'book-list', component: BookListComponent
  },
  {
    // path: 'book-details/:id',
    path: 'book-details', component: BookDetailsComponent
  },
  {
    // path: 'book-details/:id',
    path: 'book-details-rtdb', component: BookDetailsComponent
  },
  {
    // path: 'edit-book-rtdb/:id',
    path: 'edit-book-rtdb', component: EditBookRtdbComponent
  },
  {
    path: 'edit-book-fire', component: EditBookFireComponent
  },
  {
    // path: 'delete-book/:id',
    path: 'delete-book', component: DeleteBookComponent
  },
  {
    path: 'delete-book-fire', component: DeleteBookFireComponent
  },
];
```

Nota: Hemos omitido la transmisión del id del objeto para que no sea visible en la ruta cuando accedamos al mismo, ya que sería una vulnerabilidad para los datos en Firestore que el usuario pudiera conocer los IDs.

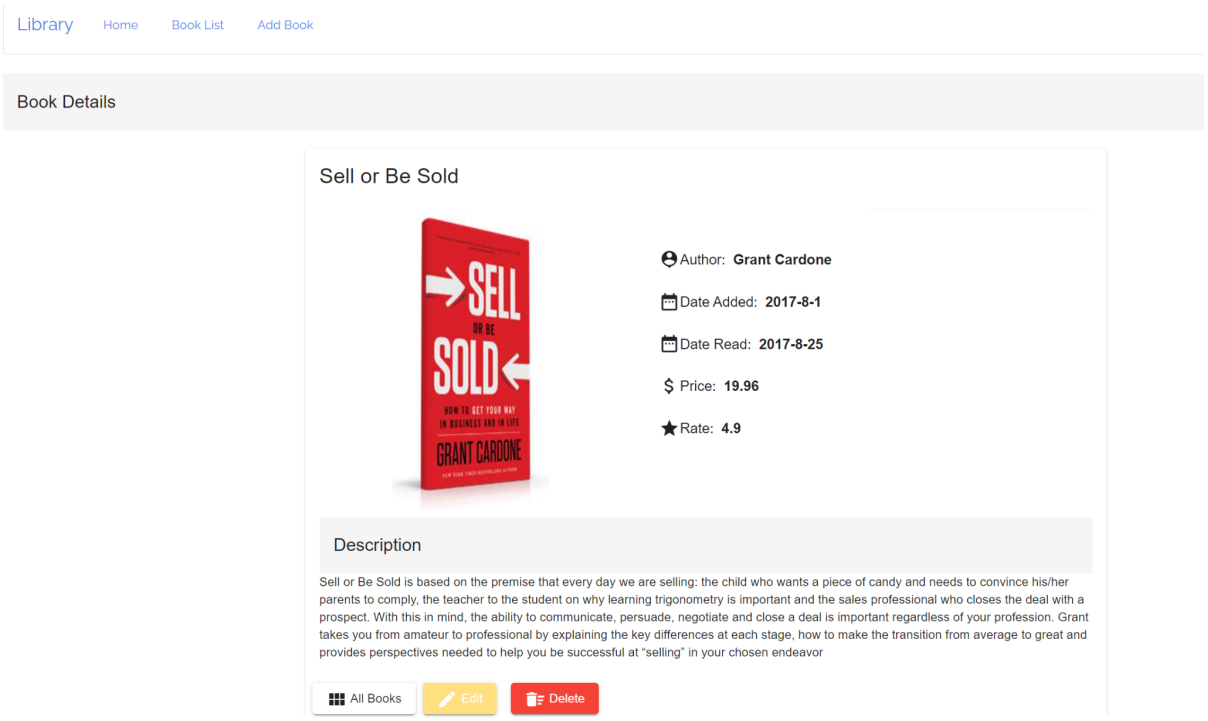
- 12.5. Si ahora vamos al inspeccionar en el navegador Chrome, podremos apreciar cómo obtenemos el objeto en cuestión en el componente de BookDetails y todos sus datos se muestran correctamente en nuestra estructura:



Objeto Libro de Firestore



Objeto Libro de RealTimeDatabase



Captura tomada al zoom 80% de la vista del BookDetails

14. ADD BOOK VIEW

Vamos a empezar desarrollando este componente, con orientación a añadir un nuevo libro a la RealTimeDB

Nota: Para un mejor funcionamiento de nuestra web, hemos creado dos nuevos componentes para separar el add-book en, add-book-fire y add-book-rtdb

- 15.1. Vamos al add-book-rtdb.component.ts para establecer las propiedades que recogerá un formulario en la vista, para crear con ellas un nuevo objeto, que será el nuevo libro que se añade a la RealTimeDB

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import BookModel from 'src/app/models/book-model';
import { RealTimeDBService } from
'src/app/services/real-time-db.service';

@Component({
  selector: 'app-add-book-rtdb',
  templateUrl: './add-book-rtdb.component.html',
  styleUrls: ['./add-book-rtdb.component.css']
})
export class AddBookRtdbComponent implements OnInit {

  book: BookModel = new BookModel();
  submitted = false;

  constructor(
    private realTimeDBService: RealTimeDBService,
    private router: Router,
  ) {  }

  ngOnInit(): void {
  }

  saveBook(): void {
    this.realTimeDBService.addBook_v2(this.book).then(() => {
      console.log('Created new item successfully!');
      this.submitted = true;
      console.log(this.book);
    });
    this.router.navigate(['book-list']);
```

```
  }

  newBook(): void {
    this.submitted = false;
    this.book = new BookModel();
    console.log(this.book);
  }
}
```

- 15.2. Para que el método de submitAdd() no nos de error, necesitamos crearlo en el servicio de RealTimeDBService

```
import { Injectable } from '@angular/core';
import { AngularFireDatabase, AngularFireList, AngularFireObject,
AngularFireAction } from '@angular/fire/compat/database';
import { Observable } from 'rxjs/';
import { map } from 'rxjs/operators';
import { Book } from '../interfaces/book';
import BookModel from '../models/book-model';

@Injectable({
  providedIn: 'root'
})
export class RealTimeDBService {

  private dbPath = '/books';

  books: AngularFireList<any[]> | undefined; // from firebase
  booksRef: AngularFireList<BookModel>;
  favouriteBooks: Observable<any> | undefined;
  unreadBooks: Observable<any> | undefined;
  bookDetails: AngularFireObject<any> | undefined; // from firebase

  constructor(
    private angularFireDatabse: AngularFireDatabase
  ) {
    this.booksRef = angularFireDatabse.list(this.dbPath);
  }

  getBooks_v2(): AngularFireList<BookModel> {
    return this.booksRef;
  }
```

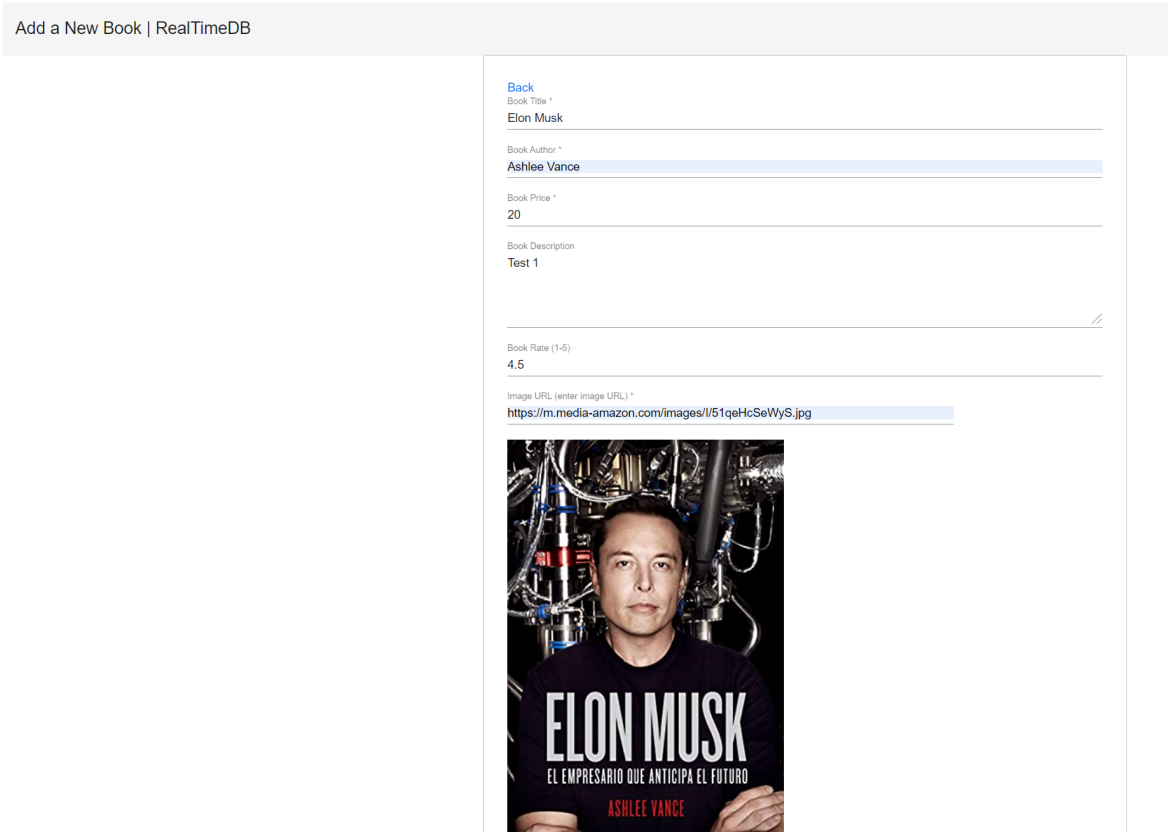
```
  getFavouriteBooks() {
    this.favouriteBooks = this.angularFireDatabse.list('/books')
      .valueChanges().pipe(map((books) => {
        const topRatedBooks = books.filter((item: any) => item.rate >
4);

        return topRatedBooks;
      }))

      return this.favouriteBooks;
  }

  getUnreadBooks() {
    this.unreadBooks = this.angularFireDatabse.list('/books')
      .valueChanges().pipe(map((books) => {
        const unreadBooks = books.filter((item: any) => item.dateread
== null);

        return unreadBooks;
      }))

      return this.unreadBooks;
  }

  getBookDetails(id: any) {
    this.bookDetails = this.angularFireDatabse.object('/books/' + id)
as AngularFireObject<Book>;

    return this.bookDetails;
  }

  addBook_v2(book: Book): any {
    return this.booksRef.push(book);
  }
```

- 15.3. En tercer lugar, vamos a su vista html para crear la siguiente estructura de formulario en el que añadiremos los correspondientes métodos

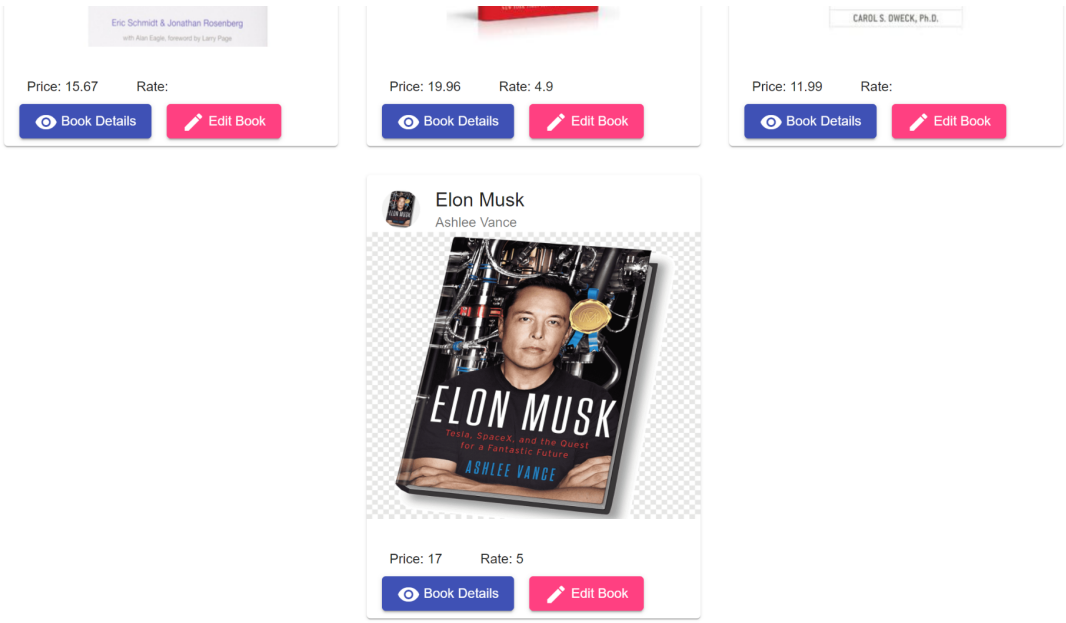
```
<!-- <p>add-book-rtdb works!</p> -->

<mat-toolbar>Add a New Book | RealTimeDB</mat-toolbar>

<div class="add-book-form" *ngIf="!submitted">
    <a [routerLink]="['/book-list']">Back</a>

    <form #myForm="ngForm">
        <mat-form-field class="full-width">
            <input matInput placeholder="Book Title" name="title"
id="title"  required [(ngModel)]="book.title">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Book Author" name="author"
id="author"  required [(ngModel)]="book.author">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Book Price" type="number"
name="price" id="price"  required [(ngModel)]="book.price">
        </mat-form-field>

        <div>
            <mat-form-field class="full-width">
                <textarea matInput name="description" id="description"
rows="5" placeholder="Book Description"
[(ngModel)]="book.description"></textarea>
            </mat-form-field>

            <mat-form-field class="full-width">
                <input matInput placeholder="Book Rate (1-5)" min="1"
max="5" step="0.1" type="number" name="rate" id="rate"
[(ngModel)]="book.rate">
            </mat-form-field>
        </div>

        <mat-form-field class="third-width">
```

```
            <input matInput placeholder="Image URL (enter image URL)"
type="url" name="imageUrl" id="imageUrl"  required
[(ngModel)]="book.imageUrl">
        </mat-form-field>
        <br>

        <img src="{{book.imageUrl}}">
        <br>
        <hr>

        <button class="a" (click)="saveBook()"
[disabled]="!myForm.form.valid">Save Book</button>

        <br><br>

        <button mat-raised-button color="warn" value="submit"
[disabled]="!myForm.form.valid">
            Create Book
        </button>

    </form>
</div>
```

- 15.4. Si vamos a nuestro navegador, podemos ver que el libro se añade después de rellenar los campos y pulsar el botón de Submit.



Captura de pantalla tomada al 70% del zoom del navegador


Price: 15.67
Rate:
Book Details
Edit Book
Price: 19.96
Rate: 4.9
Book Details
Edit Book
CAROL S. DWECK, Ph.D.
Price: 11.99
Rate:
Book Details
Edit Book
Elon Musk
Ashlee Vance
ELON MUSK
Tesla, SpaceX, and the Quest for a Fantastic Future
ASHLEE VANCE
Price: 17
Rate: 5
Book Details
Edit Book


Authentication
Firestore Database
Realtime Database
Storage
Hosting
Functions
Machine Learning
Lanzamiento y supervisión
Crashlytics, Performance, Test La...
Extensions
Spark
Gratis $0 por mes
Actualizar
https://library-67c6b-default-rtdb.europe-west1.firebasedatabase.app/
books
-Ks9A3-elu1b8faiP6u1
-Ks9DMa1mociWYhuU3N3
-KsNa2IqF0HtVBo4KIXl
-KsNa6Nmu0M67D8rZh_G
-KsZ42HAqB-18oPs61Ar
-KsZ4ugfbbPPxbb91v01
-MrhAlap7JOStKkpQASo
author: "Ashlee Vance"
description: "Esta es la fascinante historia de la tumultuosa..."
imageUrl: "https://e7.pngegg.com/pngimages/14/562/png-clip..."
price: 17
rate: 5
title: "Elon Musk"
Ubicación de la base de datos: Bélgica (europe-west1)

Ahora, vamos a hacer algo similar, pero con algunos cambios en ciertas sentencias para desarrollar el mismo componente pero esta vez para añadir el nuevo libro a Firestore.

- 15.5. Vamos al FirestoreService para crear el método de origen …

```
import { Injectable } from '@angular/core';
import { AngularFirestore, AngularFirestoreDocument,
AngularFirestoreCollection, AngularFirestoreCollectionGroup } from
'@angular/fire/compat/firestore';
import { Book } from '../interfaces/book';

@Injectable({
  providedIn: 'root'
})

export class FirestoreService {

  constructor(
    private angularFirestore: AngularFirestore
  ) {
  }

  public getBooks() {
    return
this.angularFirestore.collection('book-list').snapshotChanges();
  }

  public createBook(book: Book) {
    return this.angularFirestore.collection('book-list').add(book);
  }
}
```

- 15.6. Vamos al add-book-fire.component.ts para …

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

import { FirestoreService } from 'src/app/services/firestore.service';

import {FormControl, FormGroup, Validators} from "@angular/forms";
import { Book } from 'src/app/interfaces/book';
```

```
@Component({
  selector: 'app-add-book-fire',
  templateUrl: './add-book-fire.component.html',
  styleUrls: ['./add-book-fire.component.css']
})
export class AddBookFireComponent implements OnInit {

  // Properties for Firestore
  book: Book | undefined;
  public bookForm= new FormGroup({
    title: new FormControl('', Validators.required),
    author: new FormControl('', Validators.required),
    price: new FormControl('', Validators.required),
    description:new FormControl(''),
    rate: new FormControl(''),
    imageUrl: new FormControl('', Validators.required)
  });


  constructor(
    private firestoreService: FirestoreService,
    private router: Router,
  ) {  }

  ngOnInit(): void {
  }

  createBook() {
    console.log(this.bookForm.valid);

    if (this.bookForm.valid) {
      this.firestoreService.createBook(this.bookForm.value);

      this.router.navigate(['book-list']);

    } else {
      alert("recuerda que hay que rellenar los campos, si dejas alguno
vacio no valdrá");
    }
  }

}
```

- 15.7. Por último, vamos al add-book-fire.component.html para …

```
<!-- <p>add-book-fire works!</p> -->

<mat-toolbar>Add a New Book | Firestore</mat-toolbar>

<div class="add-book-form">
    <a [routerLink]="['/book-list']">Back</a>

    <form #myForm="ngForm" [formGroup]="bookForm"
(ngSubmit)="createBook()">
        <mat-form-field class="full-width">
            <input matInput placeholder="Book Title" name="title"
id="title" required formControlName="title">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Book Author" name="author"
id="author" required formControlName="author">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Book Price" type="number"
name="price" id="price" required formControlName="price">
        </mat-form-field>

        <table class="full-width" cellspacing="10px">
            <tr>
                <td>
                    <mat-form-field class="full-width">
                        <input matInput [matDatepicker]="dateadded"
placeholder="Date Added" type="date" name="dateadded">
                        <mat-datepicker-toggle matSuffix
[for]="dateadded"></mat-datepicker-toggle>
                    </mat-form-field>
                    <mat-datepicker #dateadded></mat-datepicker>
                </td>

                <td>
                    <mat-form-field class="full-width">
                        <input matInput [matDatepicker]="dateread"
placeholder="Date Read" type="date" name="dateread">
```

```
                        <mat-datepicker-toggle matSuffix
[for]="dateread"></mat-datepicker-toggle>
                    </mat-form-field>
                    <mat-datepicker #dateread></mat-datepicker>
                </td>
            </tr>
        </table>

        <div>
            <mat-form-field class="full-width">
                <textarea matInput name="description" id="description"
rows="5" placeholder="Book Description"
formControlName="description"></textarea>
            </mat-form-field>

            <mat-form-field class="full-width">
                <input matInput placeholder="Book Rate (1-5)" min="1"
max="5" step="0.1" type="number" name="rate" id="rate"
formControlName="rate">
            </mat-form-field>
        </div>

        <mat-form-field class="third-width">
            <input matInput placeholder="Image URL (enter image URL)"
type="url" name="imageUrl" id="imageUrl" required
formControlName="imageUrl">
        </mat-form-field>

        <br>
        <hr>

        <button style="margin-right: 30px" mat-raised-button
color="warn" value="ngSubmit" [disabled]="!myForm.form.valid">
            Create Book
        </button>

    </form>
</div>
```

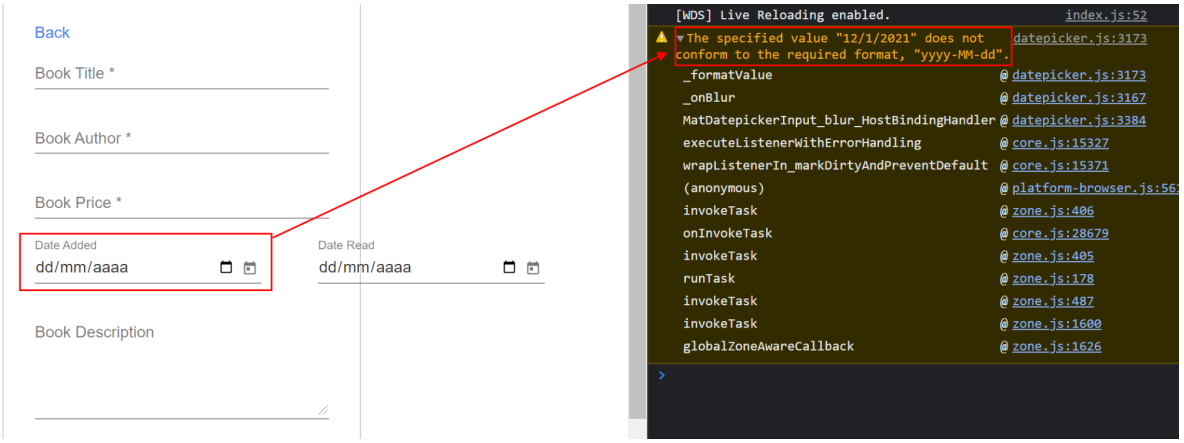
Nota: En este componente he intentado implementar el <mat-datepicker> pero …

Estoy experimentando un error (warning) en el input para seleccionar la fecha de DateAdded y DateRead.

Este warning se produce porque el <input [MatDatepicker]> predeterminadamente está preparado para recibir una fecha en formato “YYYY-MM-DD”, y el caso es que el <mat-datepicker> predeterminadamente está preparado para selecciona una fecha en formato “DD-MM-YYYY”

He intentado, aún sin éxito, modificar el formato de fecha que se recoge del mat-datepicker para que pase de exigir YYYY-MM-DD, a lo que permite seleccionar, DD-MM-YYYY.
Me he documentado en estas guías para los pasos a seguir

1) https://www.freakyjolly.com/angular-material-datepicker-change-format-of-selected-date/
2) https://www.concretepage.com/angular-material/angular-material-datepicker-format
3)https://stackoverflow.com/questions/53359598/how-to-change-angular-material-datepicker-format



Library Home Book List Add Book

Add a New Book

Back

Book Title *

Elon Musk

Book Author *

Ashlee Vance

Book Price *

99.99

Date Added

dd/mm/aaaa

Date Read

dd/mm/aaaa

Book Description

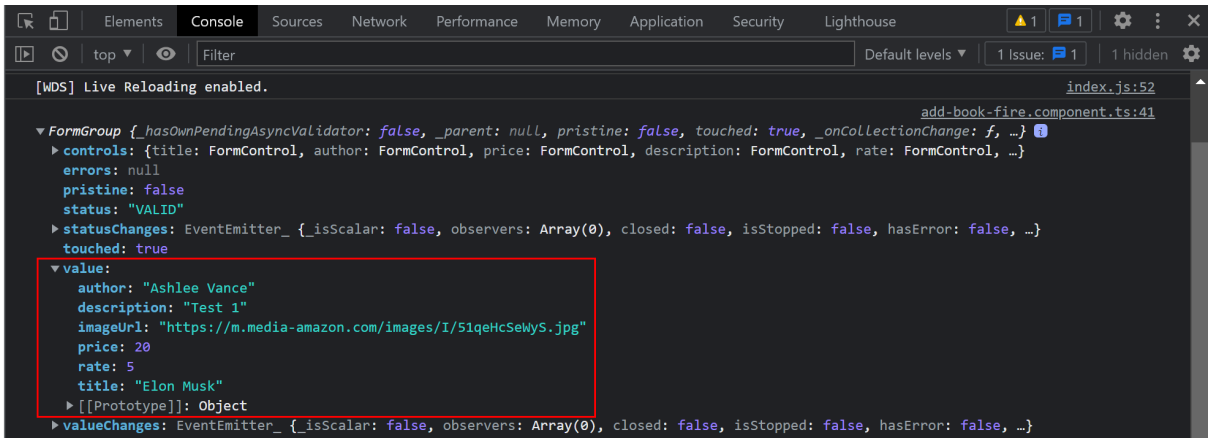
test creating new book in RealTimeDB and Firestore

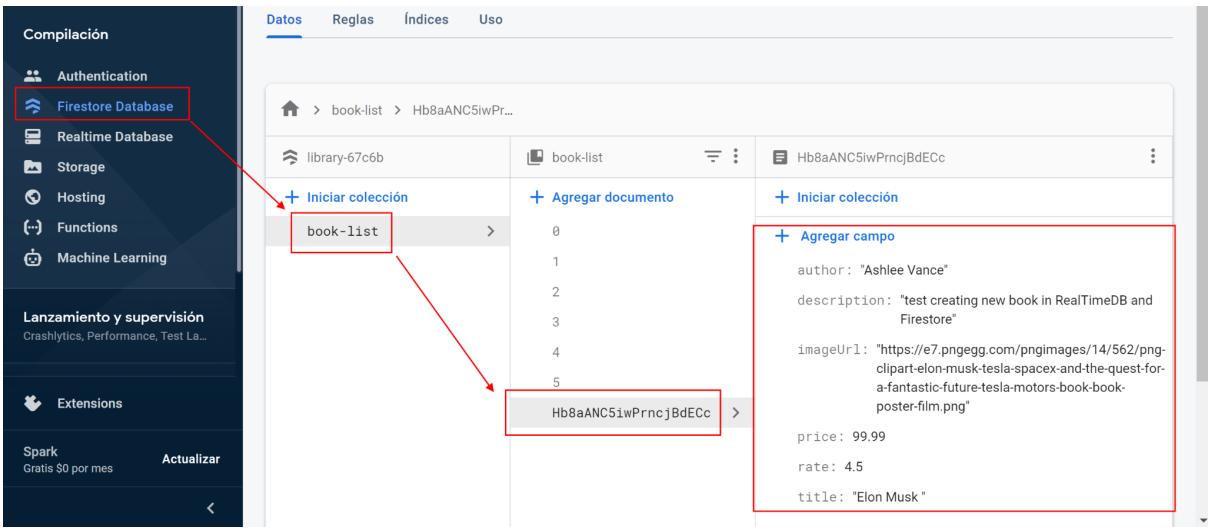
Book Rate (1-5)

4.5

Image URL (enter image URL) *

https://e7.pngegg.com/pngimages/14/562/png-clipart-e





Price: 15.67 Rate:

Book Details Edit Book

Price: 19.96 Rate: 4.9

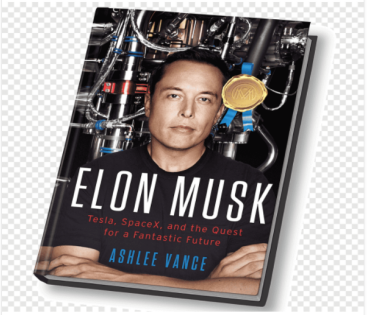
Book Details Edit Book

Price: 11.99 Rate:

Book Details Edit Book

Elon Musk

Ashlee Vance



Price: 99.99 Rate: 4.5

Book Details Edit Book

All Books || RealTimeDB

Price: 15.67 Rate:

Book Details Edit Book

Price: 19.96 Rate: 4.9

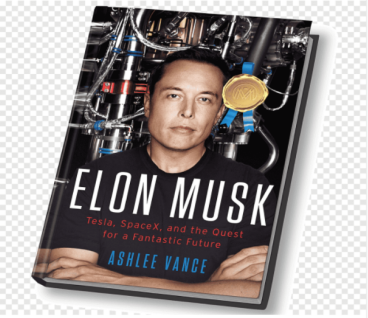
Book Details Edit Book

Price: 11.99 Rate:

Book Details Edit Book

Elon Musk

Ashlee Vance



Price: 99.99 Rate: 4.5

Book Details Edit Book

15. EDIT BOOK

- 15.1. Para una mejor organización, eliminamos el componente de “edit-book” que ya teníamos preparado desde el principio, para crear mejor en su lugar:

  ng g c components/edit-book-fire
  ng g c components/edit-book-rtdb

- 15.2. Incluímos sus paths en el app-routing.module.ts

```
  {
    // path: 'edit-book-rtdb/:id',
    path: 'edit-book-rtdb',
    component: EditBookRtdbComponent
  },
  {
    path: 'edit-book-fire',
    component: EditBookFireComponent
  },
```

- 15.3. Vamos a los services a crear los métodos originales de tipo update tanto para el FirestoreService como para el RealTimeDBService

Para Firestore

```
public updateBook(id: string, book: Book) {
    return
this.angularFirestore.collection('book-list').doc(id).set(book);
  }
```

Para RealTimeDB

```
updateBook_v2(key: string, value: any): Promise<void> {
    return this.booksRef.update(key, value);
  }
```

Nota: recordar que a veces no requiero poner todo el código de todo el archivo en cuestión, y que por ejemplo en este caso del real-time-db.service.ts, la propiedad ya declarada de antemano de booksRef ya viene haciendo referencia a la dirección /books de la RealTimeDB en Firebase … sólo recordar que para ver mejor detalles así, recomiendo ir siguiendo esta guía con mi GitHub paralelamente.

- 15.4. Ahora debemos crear los respectivos métodos en los controladores de cada componente de edit-book, y colocarlos en una estructura similar a la de AddBook.

Para Firestore

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { NavigationExtras, Router } from '@angular/router';
import { FirestoreService } from 'src/app/services/firestore.service';

@Component({
  selector: 'app-edit-book-fire',
  templateUrl: './edit-book-fire.component.html',
  styleUrls: ['./edit-book-fire.component.css']
})
export class EditBookFireComponent implements OnInit {

  book: any;
  navigationExtras: NavigationExtras = {
    state: {
      value: null
    }}
  bookForm: any;

  constructor(
    private router: Router,
    private firestoreService: FirestoreService
  ) {
    //hay que hacer esto en el constructor porque si se hiciera todo en
el ngOnInit(), obtendríamos el objetivo null debido a que la navegación
muere al crearse

//https://stackoverflow.com/questions/54891110/router-getcurrentnavigat
ion-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.book = navigation?.extras?.state;

    // establecemos que el bookForm tenga estos valores por defecto
    this.bookForm = new FormGroup({
      title: new FormControl(this.book.data.title,
Validators.required),
      author: new FormControl(this.book.data.author,
Validators.required),
```

```
      price: new FormControl(this.book.data.price,
Validators.required),
      description: new FormControl(this.book.data.description),
      rate: new FormControl(this.book.data.rate),
      imageUrl: new FormControl(this.book.data.imageUrl,
Validators.required)
    })
  }

  ngOnInit(): void {
    console.log(this.book);
    console.log(this.bookForm);

    if (this.book == null) {
      //asi controlo que nadie se intente colar
      this.router.navigate(["book-list"]);
    }
  }

  updateBook() {
    console.log(this.bookForm.valid);

    if (this.bookForm.valid) {
      this.firestoreService.updateBook(this.book.id,
this.bookForm.value);

      this.router.navigate(["book-list"]);

    } else {
      alert("no se rellenaron todos los campos del formulario revisalo
bien")
    }
  }

  isValid(field: string){
    const fieldValidate = this.bookForm.get(field);

    return (!fieldValidate.valid && fieldValidate.touched)
      ? 'is-invalid'
      : fieldValidate.touched
        ? "is-valid"
        : "";
  }
```

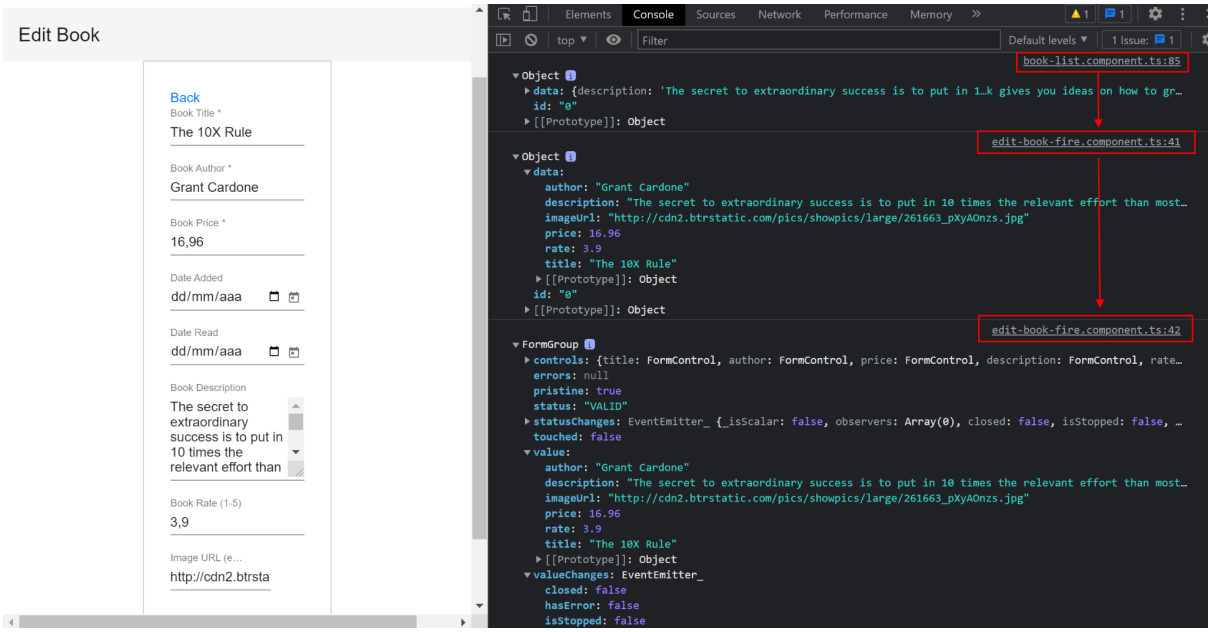
```
}
```

y para su vista (edit-book-fire.component.html)

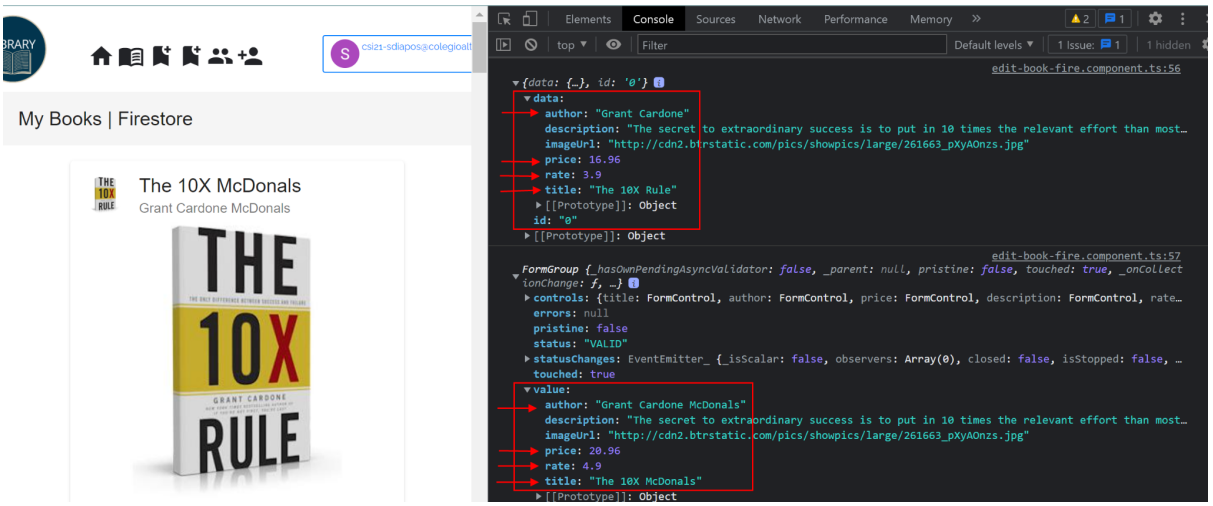
```
<!-- <p>edit-book-fire works!</p> -->

<mat-toolbar>Edit Book</mat-toolbar>

<div class="edit-book-form">
    <a [routerLink]="['/book-list']">Back</a>

    <form #values="ngForm" [formGroup]="bookForm"
(ngSubmit)="updateBook()">
      <mat-form-field class="full-width">
        <input matInput placeholder="Book Title" name="title"
id="title" formControlName="title" [class]="isValid('title')" required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Book Author" name="author"
id="author" formControlName="author" [class]="isValid('author')"
required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Book Price" type="number"
name="price" id="price" formControlName="price"
[class]="isValid('price')" required>
      </mat-form-field>

      <!-- los datepickers están desactivados porque aún no hemos hecho
el formateo de fecha de <mat-datepicker> -->
      <mat-form-field class="full-width" >
          <input matInput [matDatepicker]="dateadded" placeholder="Date
Added" type="date" name="dateadded">
          <mat-datepicker-toggle matSuffix
[for]="dateadded"></mat-datepicker-toggle>
      </mat-form-field>
      <mat-datepicker #dateadded></mat-datepicker>

      <mat-form-field class="full-width" >
          <input matInput [matDatepicker]="dateread" placeholder="Date
Read" type="date" name="dateread">
```

```
          <mat-datepicker-toggle matSuffix
[for]="dateread"></mat-datepicker-toggle>
      </mat-form-field>
      <mat-datepicker #dateread></mat-datepicker>


      <mat-form-field class="full-width">
        <textarea matInput name="description" id="description" rows="5"
placeholder="Book Description"
formControlName="description"></textarea>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Book Rate (1-5)" min="1" max="5"
step="0.1" type="number" name="rate" id="rate" formControlName="rate">
      </mat-form-field>


      <mat-form-field class="third-width">
        <input matInput placeholder="Image URL (enter image URL)"
type="url" name="imageUrl" id="imageUrl" formControlName="imageUrl"
[class]="isValid('imageUrl')" required>
      </mat-form-field>

      <hr>

      <button style="margin-right: 30px" mat-raised-button color="warn"
(click)="updateBook()" value="updateBook">
        Update Book
      </button>

    </form>
</div>
```

Vamos a hacer una prueba cambiando los datos del primer libro de Firestore “The 10X Rule”



Se envía el libro desde el BookDetails y se recibe en el EditBook, y se vuelca en el form para rellenar sus campos



Cambiamos algunos campos, y al pulsar en el botón de Actualizar y volver al BookList, vemos en la consola que efectivamente se han cambiado los datos de los campos

Para RealTimeDatabase

```
import { Component, OnInit, OnChanges } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { ActivatedRoute, NavigationExtras, Router } from
'@angular/router';
import BookModel from 'src/app/models/book-model';
import { RealTimeDBService } from
'src/app/services/real-time-db.service';

@Component({
  selector: 'app-edit-book-rtdb',
  templateUrl: './edit-book-rtdb.component.html',
  styleUrls: ['./edit-book-rtdb.component.css']
})
export class EditBookRtdbComponent implements OnInit, OnChanges {

  book: any;
  navigationExtras: NavigationExtras = { // andrés, por qué no tienes
esto en tu ActualizarComponent?
    state: {
      value: null
    }}
  bookForm: any;
  message = '';

  constructor(
    private realTimeDBService: RealTimeDBService,
    private router: Router,
    private activatedRoute: ActivatedRoute
  ) {
    //hay que hacer esto en el constructor porque si se hiciera todo en
el ngOnInit(), obtendríamos el objetivo null debido a que la navegación
muere al crearse

//https://stackoverflow.com/questions/54891110/router-getcurrentnavigat
ion-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.book = navigation?.extras?.state;

    // establecemos que el bookForm tenga estos valores por defecto
    this.bookForm = new FormGroup({
      title: new FormControl(this.book.title, Validators.required),
      author: new FormControl(this.book.author, Validators.required),
```

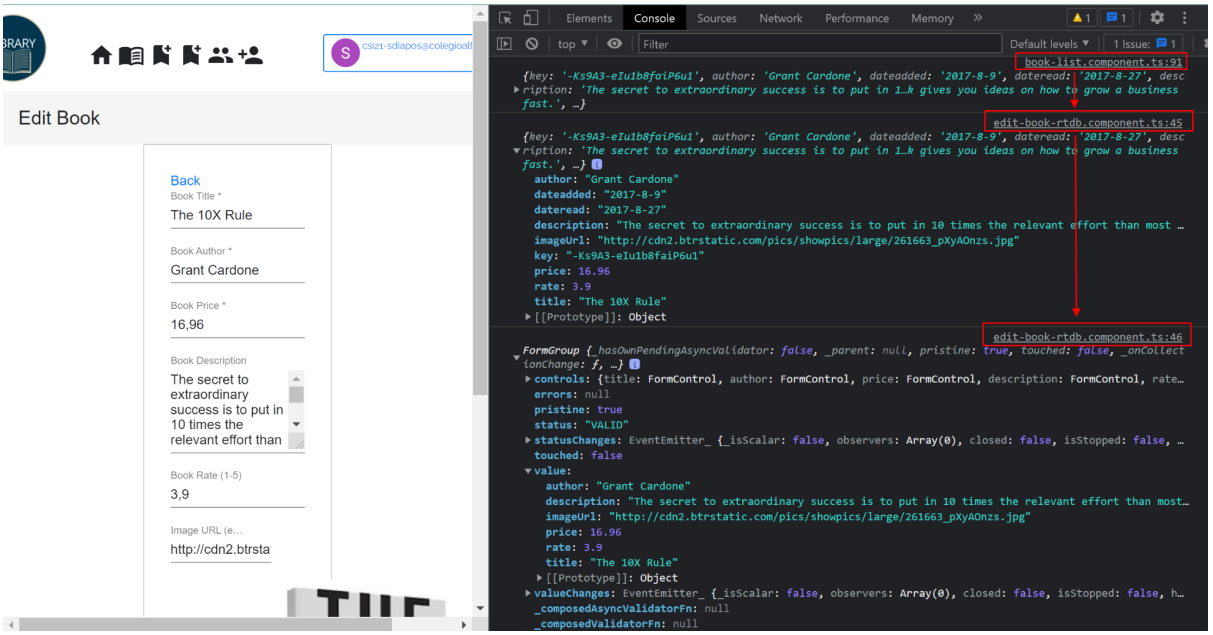
```
      price: new FormControl(this.book.price, Validators.required),
      description: new FormControl(this.book.description),
      rate: new FormControl(this.book.rate),
      imageUrl: new FormControl(this.book.imageUrl,
Validators.required)
    })
  }

  ngOnInit(): void {
    console.log(this.book);
    console.log(this.bookForm);

    if (this.book == null) {
      //asi controlo que nadie se intente colar
      this.router.navigate(["book-list"]);
    }
    this.message = '';
  }

  ngOnChanges(): void {
    this.message = '';
    this.bookForm = { ...this.book };
  }


  updateBook_v2(): void {

    if (this.bookForm.valid) {
      if (this.book.key) {
        this.realTimeDBService.updateBook_v2(this.book.key,
this.bookForm.value)
          .then(() => this.message = 'The tutorial was updated
successfully!')
          .catch(err => console.log(err));
        console.log(this.book);
        console.log(this.bookForm);
      }

      this.router.navigate(["book-list"]);

    } else {
      alert("no se rellenaron todos los campos del formulario revisalo
bien")
```

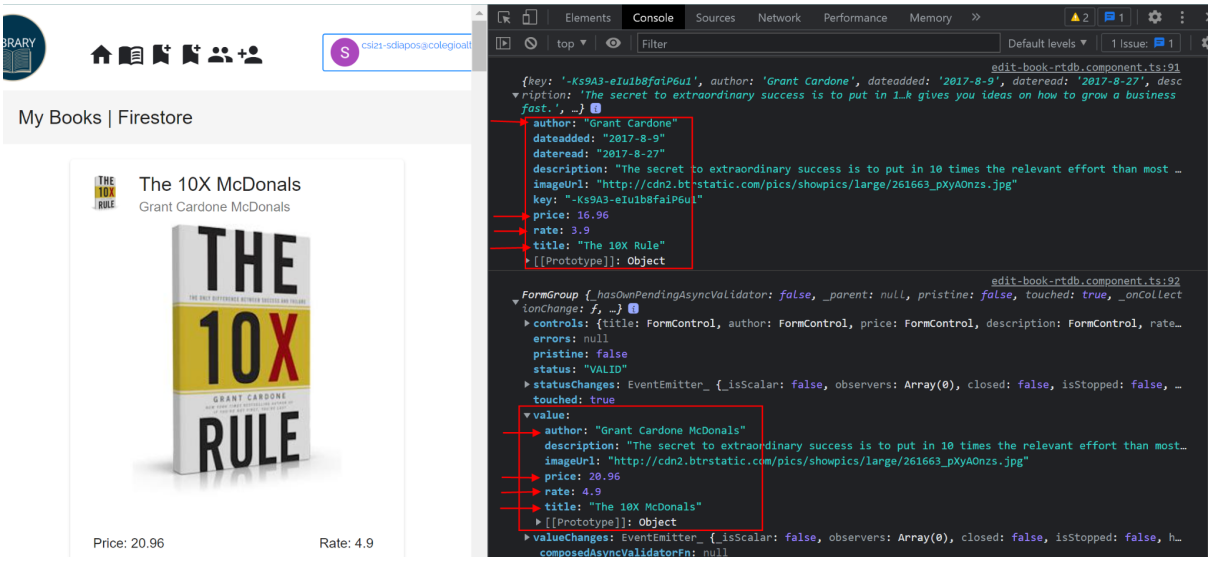
```
    }

  }

  isValid(field: string){
    const fieldValidate = this.bookForm.get(field);

    return (!fieldValidate.valid && fieldValidate.touched)
      ? 'is-invalid'
      : fieldValidate.touched
        ? "is-valid"
        : "";
  }
}
```

y para su vista (edit-book-rtdb.component.html)

```
<!-- <p>edit-book-rtdb works!</p> -->

<mat-toolbar>Edit Book</mat-toolbar>

<div class="edit-book-form" *ngIf="book">
    <a [routerLink]="['/book-list']">Back</a>

    <form #values="ngForm" [formGroup]="bookForm">
      <mat-form-field class="full-width">
        <input matInput placeholder="Book Title" name="title"
id="title" formControlName="title" [class]="isValid('title')" required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Book Author" name="author"
id="author" formControlName="author" [class]="isValid('author')"
required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Book Price" type="number"
name="price" id="price" formControlName="price"
[class]="isValid('price')" required>
      </mat-form-field>

      <mat-form-field class="full-width">
```

```
        <textarea matInput name="description" id="description" rows="5"
placeholder="Book Description" formControlName="description"
[class]="isValid('description')"></textarea>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Book Rate (1-5)" min="1" max="5"
step="0.1" type="number" name="rate" id="rate" formControlName="rate"
[class]="isValid('rate')">
      </mat-form-field>


      <mat-form-field class="third-width">
        <input matInput placeholder="Image URL (enter image URL)"
type="url" name="imageUrl" id="imageUrl" formControlName="imageUrl"
[class]="isValid('imageUrl')" required>
      </mat-form-field>

      <img src="{{book.imageUrl}}">

      <hr>

      <button class="a" (click)="updateBook_v2()" value="updateBook">
            Update Book
      </button>

    </form>
</div>
```


Edit Book
Back
Book Title *
The 10X Rule
Book Author *
Grant Cardone
Book Price *
16,96
Book Description
The secret to extraordinary success is to put in 10 times the relevant effort than
Book Rate (1-5)
3,9
Image URL (e...
http://cdn2.btrsta
Elements
Console
Sources
Network
Performance
Memory
book-list.component.ts:91
edit-book-rtdb.component.ts:45
edit-book-rtdb.component.ts:46
author: "Grant Cardone"
dateadded: "2017-8-9"
dateread: "2017-8-27"
description: "The secret to extraordinary success is to put in 10 times the relevant effort than most …
imageUrl: "http://cdn2.btrstatic.com/pics/showpics/large/261663_pXyAOnzs.jpg"
key: "-Ks9A3-eIu1b8faiP6u1"
price: 16.96
rate: 3.9
title: "The 10X Rule"


My Books | Firestore
The 10X McDonals
Grant Cardone McDonals
Price: 20.96
Rate: 4.9
Elements
Console
Sources
Network
Performance
Memory
edit-book-rtdb.component.ts:91
edit-book-rtdb.component.ts:92
author: "Grant Cardone"
dateadded: "2017-8-9"
dateread: "2017-8-27"
description: "The secret to extraordinary success is to put in 10 times the relevant effort than most …
imageUrl: "http://cdn2.btrstatic.com/pics/showpics/large/261663_pXyAOnzs.jpg"
key: "-Ks9A3-eIu1b8faiP6u1"
price: 16.96
rate: 3.9
title: "The 10X Rule"
author: "Grant Cardone McDonals"
description: "The secret to extraordinary success is to put in 10 times the relevant effort than most…
imageUrl: "http://cdn2.btrstatic.com/pics/showpics/large/261663_pXyAOnzs.jpg"
price: 20.96
rate: 4.9
title: "The 10X McDonals"

16. DELETE BOOK

Empezaremos eliminando un libro de Firestore.

- 16.1. Primero necesitamos ir al FirestoreService para el método de eliminación.

```
public deleteBook(id: string){
    return
this.angularFirestore.collection('book-list').doc(id).delete();
  }
```

- 16.2. En segundo lugar también vamos a crear un método para el book-details.component.ts para que cuando pulsemos en Eliminar, que nos lleve al componente de DeleteBookFirestore pasando el objeto de ese libro por la ruta.

```
getBookDeleteFire(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["delete-book-fire"], this.navigationExtras)
  }
```

- 16.3. Y en el book-details.component.html añadimos el siguiente botón

```
<button mat-raised-button color="warn"
(click)="getBookDeleteFire(book)">
<i class="material-icons">delete_sweep</i> Delete In Firestore</button>
```

- 16.4. Ahora en el delete-book-fire.component.ts, debemos recoger el objeto que el controlador de book-details le está enviando, para después poder eliminarlo definitivamente con el último método de deleteBook()

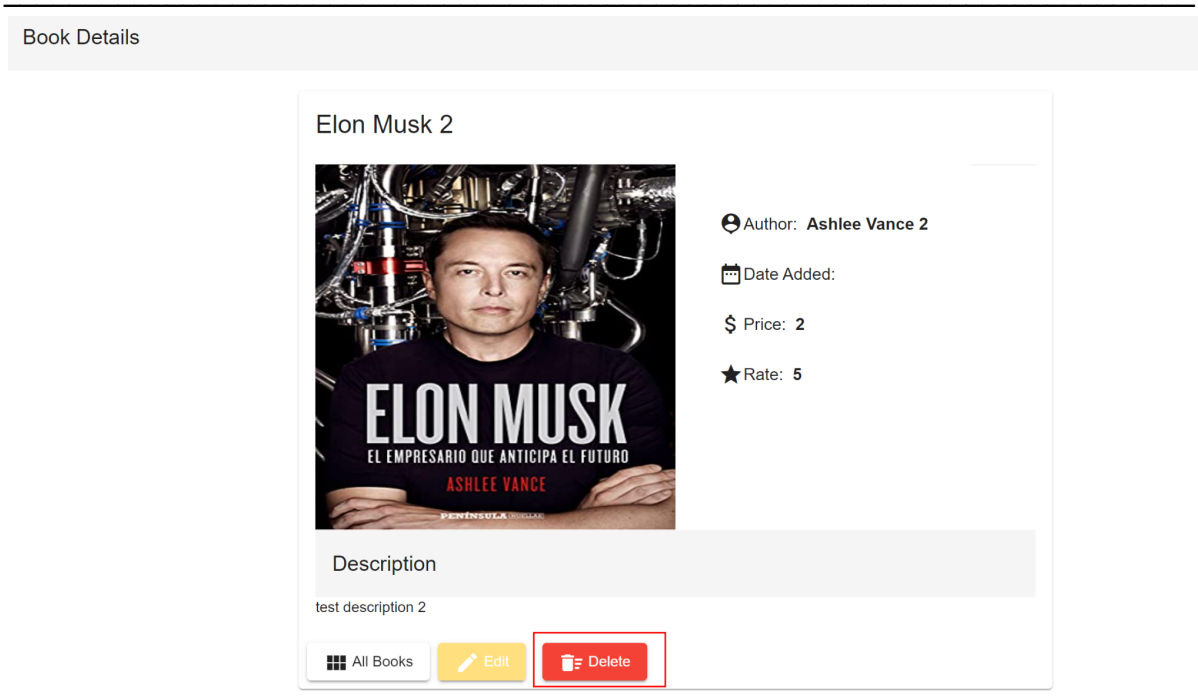
```
export class DeleteBookFireComponent implements OnInit {

  book: any = null;
  navigationExtras: NavigationExtras = {
    state:{
      value: null
    }}

  constructor(
    private router: Router,
    private firestoreService: FirestoreService
```

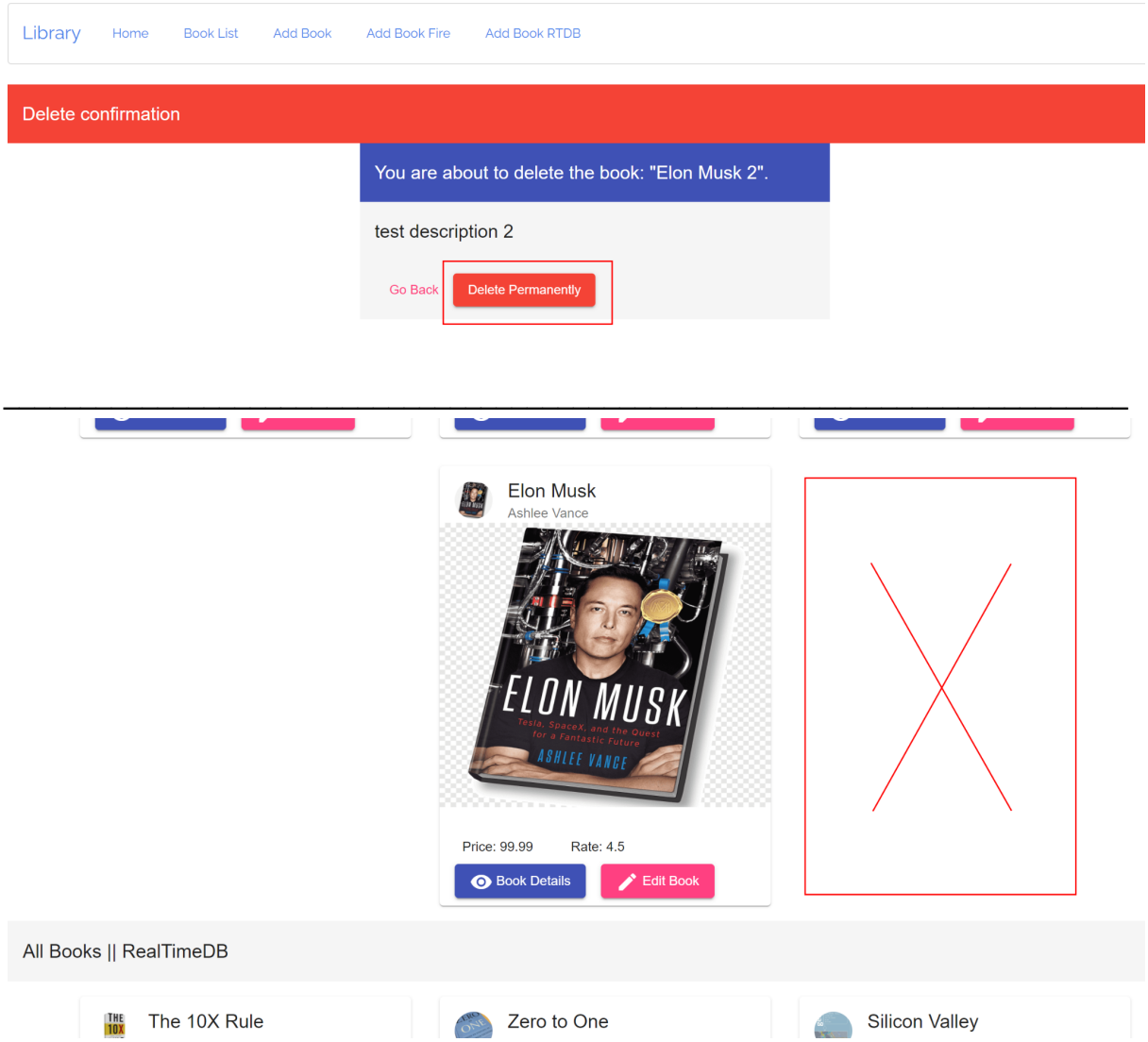
```
  ) {
    //tengo que hacerlo en el constructor porque si lo hago en init me
da null debido a que la navegacion muere al crearse

//https://stackoverflow.com/questions/54891110/router-getcurrentnavigat
ion-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.book = navigation?.extras?.state;
  }

  ngOnInit(): void {
    if (this.book == null){
      //asi controlo que nadie se intente colar
      this.router.navigate(["book-list"]);
    }
  }

  deleteBook(book: any){
    this.firestoreService.deleteBook(book.id);
    console.log(book);
    this.router.navigate(["book-list"]);
  }

}
```

- 16.5. Por último, en la vista html creamos la siguiente estructura a modo de confirmación …

```
<!-- <p>delete-book-fire works!</p> -->


<mat-toolbar color="warn"> Delete confirmation </mat-toolbar>

<div class="delete-confirm" *ngIf="book">

    <mat-toolbar color="primary">You are about to delete the book:
"{{book.title}}".</mat-toolbar>

    <mat-toolbar>
        <mat-toolbar-row>
            <span>{{book.description}}</span>
        </mat-toolbar-row>
```

```
        <mat-toolbar-row>
            <button [routerLink]="['book-details']" mat-button
color="accent"> Go Back </button>
            <button (click)="deleteBook(book)" mat-raised-button
color="warn">  Delete Permanently </button>
          </mat-toolbar-row>
    </mat-toolbar>
</div>
```




Library
Home
Book List
Add Book
Add Book Fire
Add Book RTDB
Delete confirmation
You are about to delete the book: "Elon Musk 2".
test description 2
Go Back
Delete Permanently
Elon Musk
Ashlee Vance
ELON MUSK
Tesla, SpaceX, and the Quest for a Fantastic Future
ASHLEE VANCE
Price: 99.99
Rate: 4.5
Book Details
Edit Book
All Books || RealTimeDB
The 10X Rule
Zero to One
Silicon Valley

Ahora vamos a volver a conseguir eliminar el libro, pero esta vez para la RealTimeDB

- 16.6. Vamos hacia nuestro realTimeDBService …

```
updateBook_v2(key: string, value: any): Promise<void> {
    return this.booksRef.update(key, value);
  }
```

- 16.7. En segundo lugar también vamos a crear un método para el book-details.component.ts para que cuando pulsemos en Eliminar RTDB, que nos lleve al componente de DeleteBookRTDB pasando el objeto de ese libro por la ruta.

```
getBookDeleteRTDB(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["delete-book"], this.navigationExtras)
  }
```

- 16.8. Añadimos dicho método a la vista del BookDetails en su botón correspondiente

```
<button mat-raised-button color="warn"
(click)="getBookDeleteRTDB(book)"> <i
class="material-icons">delete_sweep</i>
     Delete In RealTimeDB
</button>
```

- 16.9. Ahora en el delete-book-rtdb.component.ts, debemos recoger el objeto que el controlador de book-details le está enviando, para después poder eliminarlo definitivamente con el último método de deleteBook()

```
import { Component, OnInit, Output, EventEmitter } from
'@angular/core';
import { ActivatedRoute, NavigationExtras, Router } from
'@angular/router';
import BookModel from 'src/app/models/book-model';
import { RealTimeDBService } from
'src/app/services/real-time-db.service';

@Component({
  selector: 'app-delete-book',
  templateUrl: './delete-book.component.html',
  styleUrls: ['./delete-book.component.css']
})
export class DeleteBookComponent implements OnInit {

  book: any = null;
```

```
  navigationExtras: NavigationExtras = {
    state:{
      value: null
    }}
  @Output() refreshList: EventEmitter<any> = new EventEmitter();
  message = '';

  constructor(
    private realTimeDBService: RealTimeDBService,
    private router: Router,
  ) {
    //tengo q hacerlo en el constructor porque si lo hago en init me da
null debido a que la navegacion muere al crearse

//https://stackoverflow.com/questions/54891110/router-getcurrentnavigat
ion-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.book = navigation?.extras?.state;
  }

  ngOnInit(): void {
    console.log(this.book);

    if (this.book == null){
      //asi controlo que nadie se intente colar
      this.router.navigate(["book-list"]);
    }
    this.message = '';
  }

  deleteBook(): void {
    if (this.book.key) {
      this.realTimeDBService.deleteBook_v2(this.book.key)
        .then(() => {
          this.refreshList.emit();
          this.message = 'The Book was updated successfully!';
          console.log(this.book);
        })
        .catch(err => console.log(err));
    }
    this.router.navigate(["book-list"]);
  }
}
```

- 16.10. Por último, en la vista html creamos la siguiente estructura a modo de confirmación …

```
<!-- <p>delete-book works!</p> -->

<mat-toolbar color="warn"> Delete confirmation </mat-toolbar>

<div class="delete-confirm" *ngIf="book">

    <mat-toolbar color="primary">You are about to delete the book:
"{{book.title}}".</mat-toolbar>

    <mat-toolbar>
        <mat-toolbar-row>
            <span>{{book.description}}</span>
        </mat-toolbar-row>

        <mat-toolbar-row>
            <button [routerLink]="['book-details']" mat-button
color="accent"> Go Back </button>

            <button mat-raised-button color="warn">
                    Delete Permanently
            </button>
          </mat-toolbar-row>
    </mat-toolbar>

    <div *ngIf="!book">
        <br>
        <p>Cannot access this Book...</p>
    </div>
</div>
```

17. Clientes CRUD

Ahora toca hacer el mismo CRUD que hemos logrado con los libros, pero esta vez, para crear la parte de Clientes.
El CRUD en sí va a ser una copia del de libros, pero cambiaremos el diseño de la vista.

- 17.1. ng g c components/clientes
- 17.2. Añadimos su path al app-routing.module.ts y su routerLink en el navbar
- 17.3. ng g interface interfaces/cliente

```
export interface Cliente {
    id?: string;
    name?: string;
    email?: string;
    postalCode?: string;
    address?: string;
    age?: string;
    job?: string;
    url?: string;
    notes?: string;
}
```

- 17.4. ng g s services/clientes (será clientes-v2 porque la v1 no me funcionó)

```
import { Injectable } from '@angular/core';
import { AngularFirestore, AngularFirestoreDocument,
AngularFirestoreCollection, AngularFirestoreCollectionGroup } from
'@angular/fire/compat/firestore';
import { Cliente } from '../interfaces/cliente';

@Injectable({
  providedIn: 'root'
})
export class ClienteV2Service {

  constructor(
    private angularFirestore: AngularFirestore
  ) { }

  public getCustomers() {
    return
this.angularFirestore.collection('clientes').snapshotChanges();
  }
```

```
  public createCustomer(customer: Cliente) {
    return this.angularFirestore.collection('clientes').add(customer);
  }

  public updateCustomer(id: string, customer: Cliente) {
    return
this.angularFirestore.collection('clientes').doc(id).set(customer);
  }

  public deleteCustomer(id: string){
    return
this.angularFirestore.collection('clientes').doc(id).delete();
  }
}
```

Nota: Este CRUD ya lo voy a enfocar sólo hacia Firestore

- 17.5. Y ahora para el clientes-v2.component.ts vamos a copiar al book-list

```
import { Component, OnInit } from '@angular/core';
import { NavigationExtras, Router } from '@angular/router';
import { Cliente } from 'src/app/interfaces/cliente';
import { ClienteV2Service } from 'src/app/services/cliente-v2.service';

@Component({
  selector: 'app-clientes-v2',
  templateUrl: './clientes-v2.component.html',
  styleUrls: ['./clientes-v2.component.css']
})
export class ClientesV2Component implements OnInit {

  allCustomersFirestore: any;

  navigationExtras: NavigationExtras = {
    state: {
      value: null
    }}

  constructor(
    private clientesV2Service: ClienteV2Service,
    private router: Router
  ) { }
```

```
  ngOnInit(): void {
    this.getAllCustomers();
  }

  getAllCustomers(): void {
    this.clientesV2Service.getCustomers()
      .subscribe((res) => {
        this.allCustomersFirestore = res.map((customer: any) => {
          return {
            data: customer.payload.doc.data(),
            id: customer.payload.doc.id
          } as Cliente;
        });
      });
      console.log(this.allCustomersFirestore);
  }

  getCustomerDetails(customer: any){
    this.navigationExtras.state = customer;
    console.log(customer);
    this.router.navigate(["customer-details"], this.navigationExtras)
  }

  getCustomerEdit(customer: any){
    this.navigationExtras.state = customer;
    console.log(customer);
    this.router.navigate(["edit-customer"], this.navigationExtras)
  }

  getCustomerDelete(customer: any){
    this.navigationExtras.state = customer;
    this.router.navigate(["delete-customer"], this.navigationExtras)
  }

}
```

- 17.5. Y ahora para la estructura, vamos a reutilizar la estructura que hicimos para los jugadores de mi primer proyecto “Videojuegos” (hay un repositorio en mi GitHub)

```
<p>clientes-v2 works!</p>

<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css"
integrity="sha512-5A8nwdMOWrSz20fDsjczgUidUBR8liPYU+WymTZP1lmY9G6Oc7HlZv156XqnsgNUzTyMefFTcsFH/tnJE/+xBg==" crossorigin="anonymous"
referrerpolicy="no-referrer" />

<div class="body">
    <div class="container" *ngIf="allCustomersFirestore != null ||
allCustomersFirestore != undefined">

        <div class="cards" *ngFor="let customer of
allCustomersFirestore">
            <div class="imgBx">
                <a [routerLink]="['customer-details']">
                    <img src="{{ customer.data.url }}">
                </a>
            </div>

            <div class="content">
                <div class="details">
                    <h2>
                        {{ customer.data.job }}
                        <br>
                        <span>{{ customer.data.email }}</span>
                        <br>
                        {{ customer.data.name }},
                        {{ customer.data.age }}
                    </h2>

                    <ul class="social_icons">
                        <li><a [routerLink]="['/add-customer']"
><i class="fa fa-plus" aria-hidden="true"></i></a></li>
                        <li (click)="getCustomerDetails(customer)"><a
href="#"><i class="fa fa-info" aria-hidden="true"></i></a></li>
                        <li (click)="getCustomerEdit(customer)"><a
href="#"><i class="fa fa-pencil" aria-hidden="true"></i></a></li>
```

```
                        <li (click)="getCustomerDelete(customer)"><a
href="#"><i class="fa fa-trash-o" aria-hidden="true"></i></a></li>
                    </ul>
                </div>
            </div>
        </div>

    </div>
</div>
```

- 17.6. Añadimos el resto de componentes para este CRUD

  ng g c components/add-customer
  ng g c components/customer-details
  ng g c components/edit-customer
  ng g c components/delete-customer

Nota: para los estilos, ver mi GitHub

- 17.7. Añadimos sus respectivas rutas al app-routing.module.ts

```
{
    path: 'clientes-v2',
    component: ClientesV2Component
  },
  {
    path: 'add-customer',
    component: AddCustomerComponent
  },
  {
    path: 'customer-details',
    component: CustomerDetailsComponent
  },
  {
    path: 'edit-customer',
    component: EditCustomerComponent
  },
  {
    path: 'delete-customer',
    component: DeleteCustomerComponent
  }
```

- 17.8. Para el customer-details.component.ts …

```
import { Component, OnInit } from '@angular/core';
import { NavigationExtras, Router } from '@angular/router';

@Component({
  selector: 'app-customer-details',
  templateUrl: './customer-details.component.html',
  styleUrls: ['./customer-details.component.css']
})
export class CustomerDetailsComponent implements OnInit {

  customer: any | undefined;
  navigationExtras: NavigationExtras = {
    state: {
      value: null
    }}
  currentNavigate: any;

  constructor(
    private router: Router,
  ) {
    //hay que hacer esto en el constructor porque si se hiciera todo en
el ngOnInit(), obtendríamos el objetivo null debido a que la navegación
muere al crearse

//https://stackoverflow.com/questions/54891110/router-getcurrentnavigat
ion-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.customer = navigation?.extras?.state;

    this.currentNavigate =
navigation?.finalUrl?.root.children.primary.segments[0].path;
  }

  ngOnInit(): void {
    if (this.customer == undefined){
      //asi controlo que nadie se intente colar
      this.router.navigate(["clientes-v2"]);
    }

    console.log(this.customer);
  }
```

```
  isRouteOk() {
    if (this.currentNavigate == "customer-details") {
      return true;
    }
    else if (this.currentNavigate != "customer-details") {
      return false;
    }

    // por si acaso
    return null;
  }

  getCustomerEdit(customer: any){
    this.navigationExtras.state = customer;
    this.router.navigate(["edit-customer"], this.navigationExtras)
  }

  getCustomerDelete(customer: any){
    this.navigationExtras.state = customer;
    this.router.navigate(["delete-customer"], this.navigationExtras)
  }

}
```

- 17.9. y para su vista html …

```
<!-- <p>customer-details works!</p> -->

<mat-toolbar> Customer Details </mat-toolbar>

<div class="superContainer" *ngIf="isRouteOk()">

    <div class="customer-details" *ngIf="customer">
        <mat-card>
            <mat-card-title>
                <span
class="mat-headline">{{customer.data.name}}</span>
            </mat-card-title>

            <mat-card-content>
                <div class="left">
```

```
                    <img class="customer-url"
src="{{customer.data.url}}" />
                </div>

                <div class="right">
                    <mat-list>
                        <mat-list-item>
                            <i
class="material-icons">person_pin</i>  Name:  <b>
{{customer.data.name}}</b>
                        </mat-list-item>
                        <mat-list-item>
                            <i
class="material-icons">cake</i>  Age:  <b>
{{customer.data.age}}</b>
                        </mat-list-item>
                        <mat-list-item>
                            <i
class="material-icons">email</i>  Email:  <b>
{{customer.data.email}}</b>
                        </mat-list-item>
                        <mat-list-item>
                            <i
class="material-icons">home</i>  Address:  <b>
{{customer.data.address}}</b>
                        </mat-list-item>
                        <mat-list-item>
                            <i
class="material-icons">location_on</i>  Postal
Code:  <b> {{customer.data.postalCode}}</b>
                        </mat-list-item>
                        <mat-list-item>
                            <i
class="material-icons">work</i>  Job:  <b>
{{customer.data.job}}</b>
                        </mat-list-item>
                    </mat-list>
                </div>

                <hr>

                <mat-toolbar>Notes</mat-toolbar>
                {{customer.data.notes}}
```

```
                <p *ngIf="!customer.data.notes"> No notes yet for this
customer </p>
            </mat-card-content>

            <mat-card-actions>
                <button [routerLink]="['/clientes-v2']"
mat-raised-button> <i class="material-icons">view_module</i> All
Customers</button>

                <button mat-raised-button class="editButton"
(click)="getCustomerEdit(customer)"> <i
class="material-icons">mode_edit</i> Edit </button>

                <button mat-raised-button color="warn"
(click)="getCustomerDelete(customer)"> <i
class="material-icons">delete_sweep</i> Delete</button>

            </mat-card-actions>
        </mat-card>
    </div>

</div>
```

- 17.10. Para el add-customer.component.ts …

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { Cliente } from 'src/app/interfaces/cliente';
import { ClienteV2Service } from 'src/app/services/cliente-v2.service';

@Component({
  selector: 'app-add-customer',
  templateUrl: './add-customer.component.html',
  styleUrls: ['./add-customer.component.css']
})
export class AddCustomerComponent implements OnInit {

  // Properties for Firestore
  customer: Cliente | undefined;

  public customerForm = new FormGroup({
```

```
    name: new FormControl('', Validators.required),
    age: new FormControl('', Validators.required),
    email: new FormControl('', Validators.required),
    postalCode: new FormControl('', Validators.required),
    address: new FormControl('', Validators.required),
    job: new FormControl('', Validators.required),
    url: new FormControl('', Validators.required),
    notes: new FormControl(''),
  });

  constructor(
    private clientesV2Service: ClienteV2Service,
    private router: Router,
  ) { }

  ngOnInit(): void {
  }

  createCustomer() {
    console.log(this.customerForm.valid);

    if (this.customerForm.valid) {
      this.clientesV2Service.createCustomer(this.customerForm.value);

      this.router.navigate(['clientes-v2']);

    } else {
      alert("recuerda que hay que rellenar los campos, si dejas alguno
vacio no valdrá");
    }
  }

}
```

- 17.11. Para el add-customer.component.html …

```
<!-- <p>add-customer works!</p> -->

<mat-toolbar>Add a New Customer</mat-toolbar>

<div class="add-book-form">
    <a [routerLink]="['/clientes-v2']">Back</a>

    <form #myForm="ngForm" [formGroup]="customerForm"
(ngSubmit)="createCustomer()">
        <mat-form-field class="full-width">
            <input matInput placeholder="Customer Name" name="name"
id="name" required formControlName="name">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Customer Age" name="age"
id="age" required formControlName="age">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Customer Email" type="email"
name="email" id="email" required formControlName="email">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Customer Postal Code"
name="postalCode" id="postalCode" required
formControlName="postalCode">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Customer Address"
name="address" id="address" required formControlName="address">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Customer Job" name="job"
id="job" required formControlName="job">
        </mat-form-field>

        <mat-form-field class="third-width">
```

```
            <input matInput placeholder="Image URL (enter image URL)"
type="url" name="url" id="url" required formControlName="url">
        </mat-form-field>

        <mat-form-field class="full-width">
            <input matInput placeholder="Customer Notes" name="notes"
id="notes" formControlName="notes">
        </mat-form-field>

        <br>
        <hr>

        <button style="margin-right: 30px" mat-raised-button
color="warn" value="ngSubmit" [disabled]="!myForm.form.valid">
            Create Customer
        </button>

    </form>
</div>
```

- 17.12. Para el edit-customer.component.ts …

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { NavigationExtras, Router } from '@angular/router';
import { ClientesService } from 'src/app/services/clientes.service';

@Component({
  selector: 'app-edit-customer',
  templateUrl: './edit-customer.component.html',
  styleUrls: ['./edit-customer.component.css']
})
export class EditCustomerComponent implements OnInit {

  customer: any;
  navigationExtras: NavigationExtras = {
    state: {
      value: null
    }}
  customerForm: any;
```

```
  constructor(
    private router: Router,
    private clientesService: ClientesService
  ) {
    //hay que hacer esto en el constructor porque si se hiciera todo en
el ngOnInit(), obtendríamos el objetivo null debido a que la navegación
muere al crearse

//https://stackoverflow.com/questions/54891110/router-getcurrentnavigat
ion-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.customer = navigation?.extras?.state;

    // establecemos que el customerForm tenga estos valores por defecto
    this.customerForm = new FormGroup({
      name: new FormControl(this.customer.data.name,
Validators.required),
      age: new FormControl(this.customer.data.age,
Validators.required),
      email: new FormControl(this.customer.data.email,
Validators.required),
      postalCode: new FormControl(this.customer.data.postalCode,
Validators.required),
      address: new FormControl(this.customer.data.address,
Validators.required),
      job: new FormControl(this.customer.data.job,
Validators.required),
      url: new FormControl(this.customer.data.url,
Validators.required),
      notes: new FormControl(this.customer.data.notes,
Validators.required)
    })
  }

  ngOnInit(): void {
    console.log(this.customer);
    console.log(this.customerForm);

    if (this.customer == null) {
      //asi controlo que nadie se intente colar
      this.router.navigate(["clientes-v2"]);
    }
  }
```

```
  updateCustomer() {
    console.log(this.customerForm.valid);

    if (this.customerForm.valid) {
      this.clientesService.updateCustomer(this.customer.id,
this.customerForm.value);

      this.router.navigate(["clientes-v2"]);

    } else {
      alert("no se rellenaron todos los campos del formulario revisalo
bien")
    }
  }

  isValid(field: string){
    const fieldValidate = this.customerForm.get(field);

    return (!fieldValidate.valid && fieldValidate.touched)
      ? 'is-invalid'
      : fieldValidate.touched
        ? "is-valid"
        : "";
  }

}
```

- 17.13. Y en su vista edit-customer.component.html …

```
<!-- <p>edit-customer works!</p> -->

<mat-toolbar>Edit Customer</mat-toolbar>

<div class="edit-customer-form">
    <a [routerLink]="['/clientes-v2']">Back</a>

    <form #values="ngForm" [formGroup]="customerForm"
(ngSubmit)="updateCustomer()">
```

```
      <mat-form-field class="full-width">
        <input matInput placeholder="Customer Name" name="name"
id="name" formControlName="name" [class]="isValid('name')" required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Customer Age" name="age" id="age"
formControlName="age" [class]="isValid('age')" required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Customer email" type="email"
name="email" id="email" formControlName="email"
[class]="isValid('email')" required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Postal Code" name="postalCode"
id="postalCode" formControlName="postalCode"
[class]="isValid('postalCode')" required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <input matInput placeholder="Customer Address" name="address"
id="address" formControlName="address" [class]="isValid('address')"
required>
      </mat-form-field>

      <mat-form-field class="third-width">
        <input matInput placeholder="Image URL (enter image URL)"
type="url" name="url" id="url" formControlName="url"
[class]="isValid('url')" required>
      </mat-form-field>

      <mat-form-field class="full-width">
        <textarea matInput placeholder="Customer Notes" name="notes"
id="notes" rows="5" formControlName="notes"></textarea>
      </mat-form-field>

      <hr>

      <button style="margin-right: 30px" mat-raised-button color="warn"
(click)="updateCustomer()" value="updateCustomer">
```

```
        Update Customer
      </button>

    </form>
</div>
```

Nota: me gustaría repetir, que para los estilos, miren mi GitHub, aunque si sigue sin quedarte bien el estilo del form, puede ser porque sus clases de estilo vienen definidas en el archivo de styles.css del directorio src, o quizás porque algún campo tenga mal alguna configuración en el html.

- 17.14. Para terminar el CRUD, vamos hacia el delete-customer.component.ts …

```
import { Component, OnInit } from '@angular/core';
import { NavigationExtras, Router } from '@angular/router';
import { ClientesService } from 'src/app/services/clientes.service';

@Component({
  selector: 'app-delete-customer',
  templateUrl: './delete-customer.component.html',
  styleUrls: ['./delete-customer.component.css']
})
export class DeleteCustomerComponent implements OnInit {

  customer: any = null;
  navigationExtras: NavigationExtras = {
    state:{
      value: null
    }}

  constructor(
    private router: Router,
    private clientesService: ClientesService
  ) {
    //tengo que hacerlo en el constructor porque si lo hago en init me
da null debido a que la navegacion muere al crearse

//https://stackoverflow.com/questions/54891110/router-getcurrentnavigat
ion-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.customer = navigation?.extras?.state;
  }
```

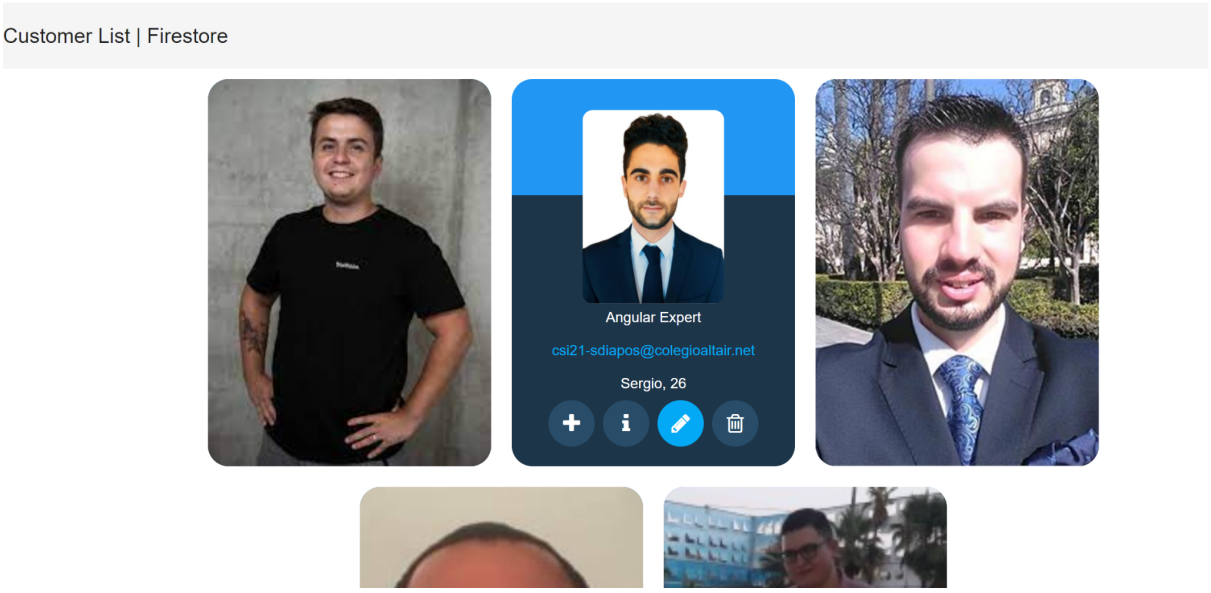
```
  ngOnInit(): void {
    if (this.customer == null){
      //asi controlo que nadie se intente colar
      this.router.navigate(["clientes-v2"]);
    }
  }

  deleteCustomer(customer: any){
    this.clientesService.deleteCustomer(customer.id);
    console.log(customer);
    this.router.navigate(["clientes-v2"]);
  }
}
```

- 17.15. Y para su vista el delete-customer.component.html …

```
<!-- <p>delete-customer works!</p> -->

<mat-toolbar color="warn"> Delete confirmation </mat-toolbar>

<div class="delete-confirm" *ngIf="customer">
    <mat-toolbar color="primary">You are about to delete the customer:
"{{customer.data.name}}".</mat-toolbar>

    <mat-toolbar>
        <mat-toolbar-row>
            <span>Name: {{customer.data.name}}</span>
        </mat-toolbar-row>
        <mat-toolbar-row>
            <span>Age: {{customer.data.age}}</span>
        </mat-toolbar-row>
        <mat-toolbar-row>
            <span>Email: {{customer.data.email}}</span>
        </mat-toolbar-row>
        <mat-toolbar-row>
            <span>Postal Code: {{customer.data.postalCode}}</span>
        </mat-toolbar-row>
        <mat-toolbar-row>
            <span>Address: {{customer.data.address}}</span>
        </mat-toolbar-row>
        <mat-toolbar-row>
            <span>Notes: {{customer.data.notes}}</span>
        </mat-toolbar-row>
```

```
        <mat-toolbar-row>
            <button [routerLink]="['/clientes-v2']" mat-button
color="accent"> Go Back </button>
            <button (click)="deleteCustomer(customer)"
mat-raised-button color="warn">  Delete Permanently </button>
          </mat-toolbar-row>
    </mat-toolbar>
</div>
```

- 17.16. Y con esto, ya hemos terminado también el CRUD de los clientes.



18. Login Logic Functionality and Register

Como ya hicimos al principio del todo con la activación del sign-up-method de Google en el apartado de Firebase de Authentication, ahora también vamos a activar el sign-up-method de Email/Contraseña.

Comenzamos creando el servicio correspondiente para esto

- 18.1. ng g s services/auth

```
import { Injectable } from '@angular/core';
import { AngularFireAuth } from '@angular/fire/compat/auth';
import firebase from 'firebase/compat/app';
```

```
@Injectable({
  providedIn: 'root'
})
export class AuthService {

  constructor(
    private angularFireAuth: AngularFireAuth
  ) { }

  async register(email: string, password: string) {
    try {
      return await
this.angularFireAuth.createUserWithEmailAndPassword(email, password);
    } catch (err) {
      console.log("error en login: ", err);
      return null;
    }
  }

  async login(email: string, password: string) {
    try {
      return await
this.angularFireAuth.signInWithEmailAndPassword(email, password);
    } catch (err) {
      console.log("error en login: ", err);
      return null;
    }
  }
```

```
  async loginWithGoogle(email: string, password: string) {
    try {
      return await this.angularFireAuth.signInWithPopup(new
firebase.auth.GoogleAuthProvider())
    } catch (err) {
      console.log("error en login con Google: ", err);
      return null;
    }
  }

  getUserInfo() {
    return this.angularFireAuth.authState;
  }

  logout() {
    this.angularFireAuth.signOut();
  }
}
```

Como podemos observar, en este servicio disponemos de 5 métodos de los cuáles verdaderamente 3 son la esencia base de la lógica del Login/Register, que son los métodos de register(), login() y logout().

Es muy típico que ya a día de hoy, para facilitar el login y crear confianza en los usuarios, se suele establecer un login provisto por Google, que es el método de Firebase de “signInWithPopup()”.

Por otra parte también he puesto un método para poder obtener cualquier dato del usuario usando el authState.

- 18.2. Ahora vamos a crear un par de componentes, que serán el Login y Register

  ng g c components/login
  ng g c components/register

- 18.3. Primero, en el register.component.ts …

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/services/auth.service';
import { RegisterService } from 'src/app/services/register.service';
```

```
@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent implements OnInit {

  users: any;
  user = {
    email: '',
    password: '',
    name: ''
  }

  constructor(
    private authService: AuthService,
    private registerService: RegisterService,
    private router: Router
  ) { }

  ngOnInit(): void {
    this.registerService.obtenerTodos("users").subscribe((usuariosRef)
=> {
      console.log("usuariosRef: ", usuariosRef);
      this.users = usuariosRef.map(userRef => {
        let usuario: any = userRef.payload.doc.data();
        usuario['id'] = userRef.payload.doc.id;
        return usuario;
      });
      console.log(this.users);
    })
  }
```

```
  register() {
    const { email, password } = this.user;

    this.authService.register(email, password).then(user => {
      console.log("se registro: ", user);

      let list = [...this.users];
      let existe = list.find(user => user.email == email);

      if (!existe) {
        console.log("USUARIO NUEVO CREADO");

        this.registerService.crear('users', this.user);
      };

      this.router.navigate(['/book-list']);

    }).catch(err => {
      console.log(err)
    })
  }
}
```

- 18.4. Y en su vista html …

```
<!-- <p>register works!</p> -->

<div class="container">
    <div class="card text-center">

      <div class="card-header">
        <b> REGISTRO</b>
      </div>

      <div class="card-body p-4">
        <label style="float: left;">Nombre</label>
        <input class="form-control m-2" type="email"
placeholder="ingrese su nombre" [(ngModel)]="user.name">

        <label style="float: left;">Email</label>
        <input class="form-control m-2" type="email"
placeholder="ingrese su email" [(ngModel)]="user.email">
```

```
        <label style="float: left;">Password</label>
        <input class="form-control m-2" type="password"
placeholder="ingrese su password" [(ngModel)]="user.password">

        <button class="btn btn-primary btn-block m-2"
(click)="register()">Registrar</button>
        <label>Ya tenes cuenta? <a
routerLink="/login">Ingresar</a></label>
      </div>

    </div>
</div>
```

Nota: aunque no documente el paso, no olvidar enrutar los nuevos componentes en el app-routing.module.ts …

- 18.5. En el otro punto de vista, para el login.component.ts …

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/services/auth.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  user = {
    email: '',
    password: ''
  }

  constructor(
    private authService: AuthService,
    private router: Router
  ) { }

  ngOnInit(): void {
  }
```

```
  login() {
    console.log(this.user);

    const { email, password } = this.user; // destructuring

    this.authService.login(email, password).then((res) => {
      console.log("Se registró: ", res);
    })

    this.router.navigate(['/book-list']);
  }

  loginWithGoogle() {
    console.log(this.user);

    const { email, password } = this.user; // destructuring

    this.authService.loginWithGoogle(email, password).then((res) => {
      console.log("Se registró: ", res);
    })

    this.router.navigate(['/book-list']);
  }

  getUserInfo() {
    this.authService.getUserInfo()
      .subscribe((res) => {
        console.log(res?.email);
      });
  }

  logout() {
    this.authService.logout();
  }
}
```

- 18.6. Y en su estructura html …

```
<!-- <p>login works!</p> -->

<div class="container">
    <div class="card text-center">
        <div class="card-header">
          <b>Login</b>
        </div>

        <div class="card-body p-4">

            <label style="float: left;">Name</label>
            <input class="form-control m-2" type="email"
placeholder="Introduzca su email" [(ngModel)]="user.email">

            <label style="float: left;">Password</label>
            <input class="form-control m-2" type="password"
placeholder="Ingtroduzca su contraseña" [(ngModel)]="user.password">

            <button class="btn btn-primary btn-block m-2"
(click)="login()">Ingresar</button>
            <button class="btn btn-info btn-block m-2"
(click)="loginWithGoogle()">Google</button>
            <label>¿Aún no tienes cuenta?<a
routerLink="/register">Registrarse</a></label>

            <div>
                <button class="btn btn-info btn-block m2"
(click)="getUserInfo()">Usuario Logeado</button>
                <button class="btn btn-info btn-block m2"
(click)="logout()">Logout</button>
            </div>
        </div>

    </div>
</div>
```

- 18.7. El última detalle para que verdaderamente quede bien el Login, es hacer ver al usuario que, efectivamente si no está loggeado, no puede ver el contenido de nuestra web, así que conseguiremos este detalle mediante el navbar

Primero, en el navbar.component.ts

```
export class NavbarComponent implements OnInit {

  userInfo = this.authService.getUserInfo();

  constructor(
    private authService: AuthService
  ) { }

  ngOnInit(): void {
  }

  logout(){
    this.authService.logout();
  }

}
```

Luego, en el navbar.component.html

```
<!-- <p>navbar works!</p> -->

    <div class="navigation">

        <img [routerLink]="['/']" src="../../../assets/library.jpg"
class="logo">

        <div class="spacer1"></div>

        <ul>
            <li class="list" *ngIf="userInfo | async as user">
                <a [routerLink]="['/']">
                    <span class="icon material-icons">
                        home
                    </span>
                    <span class="text">Home</span>
                </a>
            </li>

            <li class="list" *ngIf="userInfo | async as user">
                <a [routerLink]="['/book-list']">
```

```
                <span class="icon material-icons">
                    menu_book
                </span>
                <span class="text">Books</span>
            </a>
        </li>

        <li class="list" *ngIf="userInfo | async as user">
            <a [routerLink]="['/add-book-fire']">
                <span class="icon material-icons">
                    bookmark_add
                </span>
                <span class="text">Add Book Fire</span>
            </a>
        </li>

        <li class="list" *ngIf="userInfo | async as user">
            <a [routerLink]="['/add-book-rtdb']">
                <span class="icon material-icons">
                    bookmark_add
                </span>
                <span class="text">Add Book RTDB</span>
            </a>
        </li>

        <li class="list" *ngIf="userInfo | async as user">
            <a [routerLink]="['/clientes-v2']">
                <span class="icon material-icons">
                    people
                </span>
                <span class="text">Customers</span>
            </a>
        </li>

        <li class="list" *ngIf="userInfo | async as user">
            <a [routerLink]="['/add-customer']">
                <span class="icon material-icons">
                    person_add
                </span>
                <span class="text">Add Customer</span>
            </a>
        </li>
```

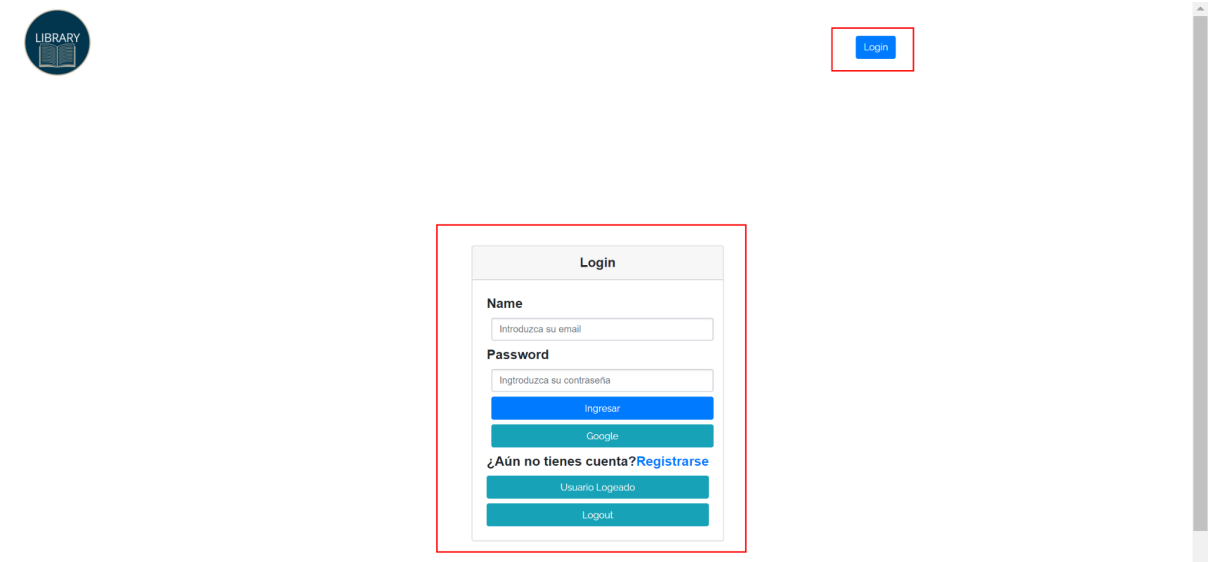
```
            <li class="list" *ngIf="userInfo | async as user">
                <a [routerLink]="['/lista-prestamos']">
                    <span class="icon material-icons">
                        handshake
                    </span>
                    <span class="text">Préstamos</span>
                </a>
            </li>

            <div class="spacer2"></div>

            <div ngbDropdown class="d-inline-block" *ngIf="userInfo |
async as user; else LoginButton">
                <button class="btn btn-outline-primary"
id="dropdownBasic1" ngbDropdownToggle>
                    <img style="border-radius: 100%;"
*ngIf="user.photoURL" src="{{user.photoURL}}" width="30" height="30"
                        class="d-inline-block align-top" alt="">
                    {{user.email}}
                </button>
                <div ngbDropdownMenu aria-labelledby="dropdownBasic1">
                    <button ngbDropdownItem
routerLink="/panel-control">Panel de control</button>
                    <button ngbDropdownItem routerLink="/login"
(click)="logout()">Salir</button>
                </div>
            </div>

            <ng-template #LoginButton>
                <button class="btn btn-primary" routerLink="/login">
Login</button>
            </ng-template>

        </ul>
    </div>
```

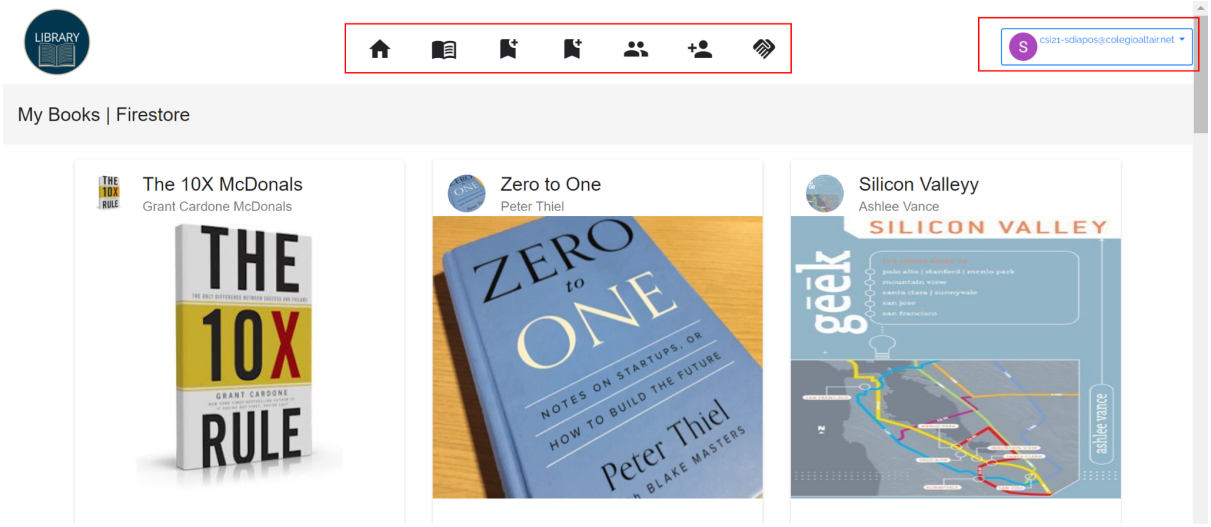
El truco visual está, por un lado, en los dos div que conforman el botón de Login (cuando el, usuario no está loggeado) y el botón con el email y photoUrl del usuario (que indica que el usuario sí está loggeado), los cuales pueden ir alternándose sin coinciddir los dos a la vez en el frontend, gracias a un uso algo más avanzado de la directiva *ngIf del div del botón de el email y photoUrl del usuario loggeado …

```
*ngIf="userInfo | async as user; else LoginButton"
```

Cómo los métodos originales del AuthService son asíncronos (async), podemos usar este tipo especial de pipe, en la que si la propiedad del navbar.component.ts “userInfo” está rellena con los datos que le han podido llegar, se activará tal botón que tenga esta condición, y en este caso concreto, contamos con un else, el cual al no estar userInfo relleno, se mostraría en su lugar el botón de Login ya que se entendería que al no estar relleno userInfo, el usuario no estaría loggeado (y porque el botón de Login cuenta con un id (#) que funciona como “ancla”).

Así que he puesto este pipe en los demás <li> que conforman el navbar, para que el propio navbar (que es el componente que da acceso a todo el contenido de la web) sólo se muestre si el usuario está loggeado.





19. Listar Préstamos y Devolver Préstamos

Para que el bibliotecario que usase nuestra web pudiera llevar un control (log) de los libros que presta a sus clientes …

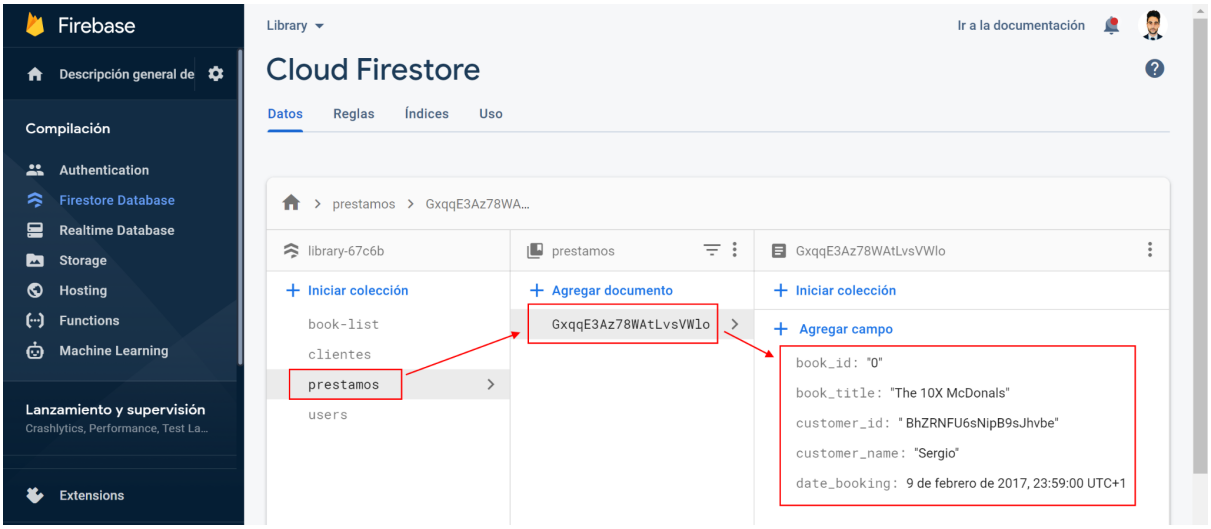
- 19.1. Creamos un servicio para ello, el cual será un CRUD normal de Firestore

  ng g s services/prestamos

```
import { Injectable } from '@angular/core';
import { AngularFirestore } from '@angular/fire/compat/firestore';

@Injectable({
  providedIn: 'root'
})
export class PrestamosService {

  constructor(
    private angularFirestore: AngularFirestore
  ) { }

  public createPrestamo(prestamo: any) {
    return this.angularFirestore.collection('prestamos').add(prestamo);
  }

  public getPrestamos() {
    return
this.angularFirestore.collection('prestamos').snapshotChanges();
  }

  public updatePrestamo(Id: string, prestamo: any) {
    return
this.angularFirestore.collection('prestamos').doc(Id).set(prestamo);
  }

  public deletePrestamo(Id: string) {
    return
this.angularFirestore.collection('prestamos').doc(Id).delete();
  }
```

```
  public getPrestamo(Id: string) {
    return
this.angularFirestore.collection('prestamos').doc(Id).snapshotChanges()
;
  }
}
```

- 19.2. Vamos a pasar por la Firestore para crear la colección llamada “prestamos” y dentro de ella, vamos a crear un objeto a modo de prueba con los siguientes campos



- 19.3. Necesitamos crear dos componentes para esto, y el primero será la típica lista

  ng g c components/lista-prestamos

```
import { Component, OnInit } from '@angular/core';
import { NavigationExtras, Router } from '@angular/router';
// import { Timestamp } from 'rxjs/internal/operators/timestamp'; //
este precísamente es el que NO nos sirve para esto
import { PrestamosService } from 'src/app/services/prestamos.service';
import firebase from 'firebase/compat/app';
import Timestamp = firebase.firestore.Timestamp;

@Component({
  selector: 'app-lista-prestamos',
  templateUrl: './lista-prestamos.component.html',
  styleUrls: ['./lista-prestamos.component.css']
})
export class ListaPrestamosComponent implements OnInit {
```

```
  prestamos: any = [];
  navigationExtras: NavigationExtras = {
    state: {
      value: null
    }};

  constructor(
    private router: Router,
    private prestamosServices: PrestamosService
  ) { }

  ngOnInit(): void {
    this.prestamosServices.getPrestamos()
      .subscribe((bookings) => {
        this.prestamos = [];

        bookings.forEach((bookingData: any) => {
          this.prestamos.push({
            data: bookingData.payload.doc.data(),
            id: bookingData.payload.doc.id
          });
        })
        console.log(this.prestamos);
      })
  }

  finalizarPrestamo(prestamo: any) {
    prestamo.data.date_end_booking = Timestamp.now();

    this.prestamosServices.updatePrestamo(prestamo.id, prestamo.data);
    this.router.navigate(['lista-prestamos']); // nos quedamos en el
mismo componente pero resfrescándolo
  }
}
```

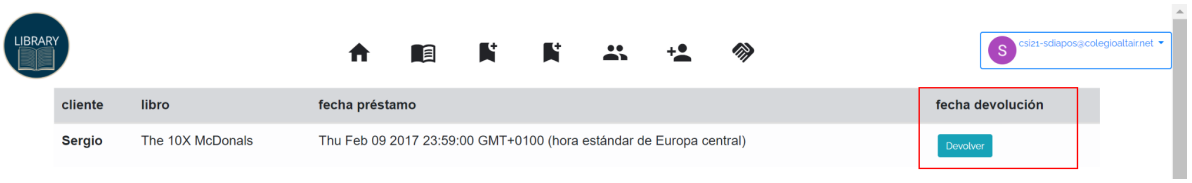
La lógica de todo esto reside en que, cuando estemos en el BookDetails, veremos un botón más como los que ya pusimos en su momento, que dirá “Prestar Libro”, y nos lleve al componente que desarrollaremos después, que será el “crear-prestamo”, en el cual estaremos recibiendo el objeto del libro en cuestión, y ya sólo tendremos que seleccionar un cliente de los ya registrados para crear el objeto del préstamo …

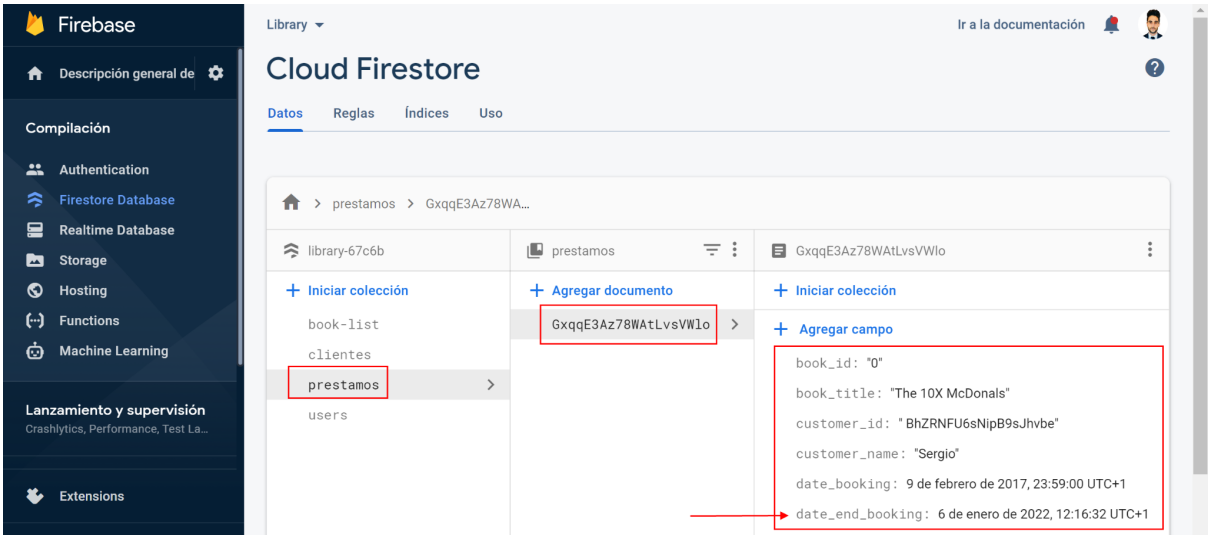
Y tanto para devolver el libro (fin del préstamo) como para ver el log (listado) de los préstamos realizados, lo hacemos desde este componente, el “lista-prestamos”, el cual obtiene los objetos de la colección “prestamos” a través de los métodos del lista-prestamos.component.ts y los lista en la siguiente estructura …

```
<!-- <p>lista-prestamos works!</p> -->

<div class="container">
  <table class="table table-hover">

    <thead>
      <tr class="table-secondary">
        <th scope="col">cliente</th>
        <th scope="col">libro</th>
        <th scope="col">fecha préstamo</th>
        <th scope="col">fecha devolución</th>
      </tr>
    </thead>

    <tbody>
      <tr *ngFor="let prestamo of prestamos" class="table-light">
        <th scope="row">{{prestamo.data.customer_name}}</th>

        <td>{{prestamo.data.book_title}}</td>

        <td>{{prestamo.data.date_booking.toDate()}}</td>

        <div *ngIf="prestamo.data.date_end_booking != null">
          <td>{{prestamo.data.date_end_booking.toDate()}}</td>
        </div>

        <div *ngIf="prestamo.data.date_end_booking == null">
          <td><button type="button"
(click)="finalizarPrestamo(prestamo)" class="btn btn-info
m-1">Devolver</button></td>
        </div>
      </tr>
    </tbody>

  </table>
</div>
```

En este momento, al ver este componente, te habrás dado cuenta de que tanto en el método de finalizarPrestamo() como en un <td> de la lista, hay un campo del objeto préstamo que, a priori, no existe cuando se crea el objeto.

La funcionalidad del método de devolverPrestamo() hace que al pulsar el botón de "devolver" de un préstamo de la lista, se añada a ese objeto un campo con la fecha en que se pulsa tal botón (se entiende que se pulsa cuando el cliente le devuelve el libro prestado al bibliotecario).





Antes de proseguir con el componente que nos queda de CrearPrestamo, vamos a resolver la falla de lógica de nuestra web de que, si un libro está prestado, no debería de poderse editar los datos de tal libro.

Nota: recordar que desde el CRUD de clientes, ya estamos haciendo todo lo demás con Firestore

- 19.4. Empezaremos por quitar la posibilidad de editar un libro desde el BookList

```
<a class="a" (click)="getBookDetails(book)">
            <span class="span">Book Details</span>
            <span class="span">Click Here</span>
          </a>
          <!-- <a class="b" (click)="getBookEdit(book)">
            <span class="span">Edit Book</span>
            <span class="span">Click Here</span>
          </a> -->
```

Y en su css, le damos más width al primer botón y acabará rellenando todo el espacio de la card

- 19.5. Para los botones del BookDetails aplicamos los siguientes cambios:

```
<button class="button-details" [routerLink]="['/book-list']"><a
routerLinkActive="router-link-active"
style="--clr:#4da9ff;--i:0;"><span>All Books</span></a></button>
                <button class="button-details"
(click)="getBookEditFire(book)" *ngIf="book.prestamo == null; else
elseBlock"><a routerLinkActive="router-link-active"
style="--clr:#d6cd52;--i:0;"><span>Edit Book</span></a></button>
                <ng-template #elseBlock>
                    <button class="button-details" disabled
(click)="getBookEditFire(book)"><a
routerLinkActive="router-link-active"
style="--clr:#d6cd52;--i:0;"><span>Edit Book</span></a></button>
                </ng-template>

                <button class="button-details"
(click)="getBookDeleteFire(book)"><a
routerLinkActive="router-link-active"
style="--clr:#f86464;--i:0;"><span>Delete Fire</span></a></button>
                <button class="button-details"
(click)="getBookDeleteRTDB(book)"><a
routerLinkActive="router-link-active"
style="--clr:#ff8f44;--i:0;"><span>Delete RTDB</span></a></button>
```

```
                <button class="button-details"
(click)="getBookBookingFire(book)" *ngIf="book.prestamo == null; else
elseBlock2"><a routerLinkActive="router-link-active"
style="--clr:#d793ff;--i:0;"><span>Booking Fire</span></a></button>
                <ng-template #elseBlock2>
                    <button class="button-details"
(click)="finalizarPrestamo(book.prestamo)"><a
routerLinkActive="router-link-active"
style="--clr:#d793ff;--i:0;"><span>Devolver Libro</span></a></button>
                </ng-template>
```

- 19.6. Y el ts de BookDetails quedaría finalmente así, añadiendo ciertas novedades…

```
import { Component, OnInit } from '@angular/core';
import { RealTimeDBService } from
'src/app/services/real-time-db.service';
import { ActivatedRoute, NavigationExtras, Router } from
'@angular/router';
import { FirestoreService } from 'src/app/services/firestore.service';
import { PrestamosService } from 'src/app/services/prestamos.service';
import firebase from 'firebase/compat/app';
import Timestamp = firebase.firestore.Timestamp;

@Component({
  selector: 'app-book-details',
  templateUrl: './book-details.component.html',
  styleUrls: ['./book-details.component.css']
})
export class BookDetailsComponent implements OnInit {

  book: any | undefined;
  prestamos: any | undefined;
  navigationExtras:NavigationExtras={
    state:{
      value:null
    }}
  currentNavigate: any;

  constructor(
    private realTimeDBService: RealTimeDBService,
    private firestoreService: FirestoreService,
```

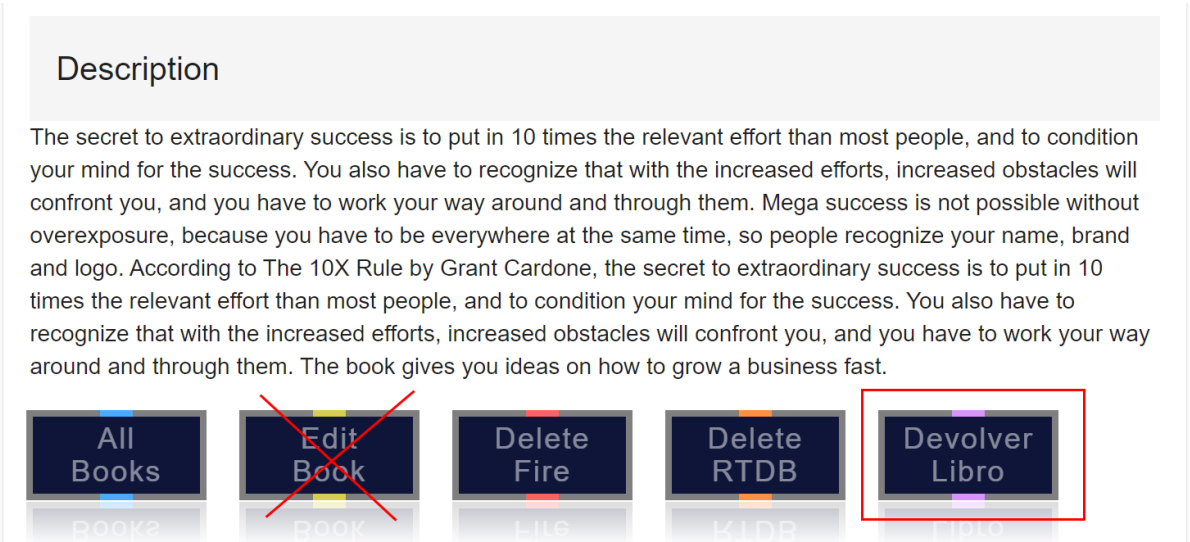
```
    private router: Router,
    private activatedRoute: ActivatedRoute,
    private prestamosService: PrestamosService
  ) {
    //hay que hacer esto en el constructor porque si se hiciera todo en
el ngOnInit(), obtendríamos el objetivo null debido a que la navegación
muere al crearse

//https://stackoverflow.com/questions/54891110/router-getcurrentnavigat
ion-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.book = navigation?.extras?.state;

    this.currentNavigate =
navigation?.finalUrl?.root.children.primary.segments[0].path;
  }

  ngOnInit(): void {

    if (this.book == undefined){
      //asi controlo que nadie se intente colar
      this.router.navigate(["book-list"]);
    }

    console.log(this.book);

    this.prestamosService.getPrestamos().subscribe((prestamosSnapshot)
=> {
      this.prestamos = [];
      prestamosSnapshot.forEach((prestamoData: any) => {
        this.prestamos.push({
          data: prestamoData.payload.doc.data(),
          id: prestamoData.payload.doc.id
        });
      })
      this.fillBookBooking();
    })
  }

  esFirestore() {
    if (this.currentNavigate == "book-details") {
      return true;
    }
```

```
    else if (this.currentNavigate == "book-details-rtdb") {
      return false;
    }
    // por si acaso
    return null;
  }

  getBookEditFire(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["edit-book-fire"], this.navigationExtras)
  }

  getBookEditRTDB(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["edit-book-rtdb"], this.navigationExtras)
  }

  getBookDeleteFire(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["delete-book-fire"], this.navigationExtras)
  }

  getBookDeleteRTDB(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["delete-book"], this.navigationExtras)
  }

  getBookBookingFire(book: any){
    this.navigationExtras.state = book;
    this.router.navigate(["crear-prestamo-fire"],
this.navigationExtras)

    console.log(this.navigationExtras.state);
  }

  fillBookBooking() {
    for(let j = 0; j < this.prestamos.length; j++) {
      if(this.book.id == this.prestamos[j].data.book_id &&
this.prestamos[j].data.date_end_booking == null) {
        this.book.prestamo = this.prestamos[j];
      }
    }
  }
```

```
  finalizarPrestamo(prestamo: any) {
    prestamo.data.date_end_booking = Timestamp.now();

    this.prestamosService.updatePrestamo(prestamo.id, prestamo.data);
    this.router.navigate(['lista-prestamos']); // nos quedamos en el
mismo componente pero resfrescándolo
  }
}
```

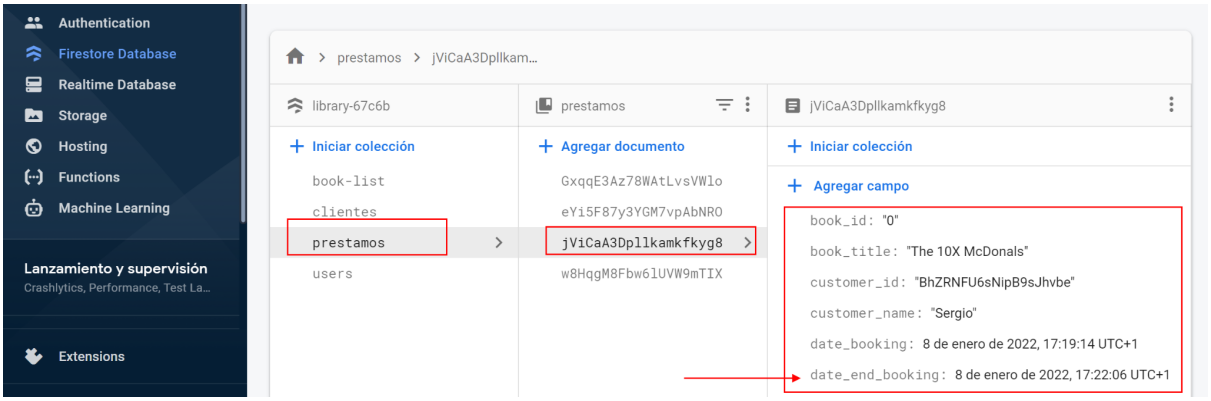
Como podemos ver, en función de si el libro al que accedemos a sus detalles, está prestado o no, se mostrará un botón para devolver el préstamo u otro para crear el préstamo.

Y ahora, con el objeto de prueba que ya teníamos de la colección Préstamos, podemos ver que el control de los botones del BookDetails funciona perfectamente, y que cuando el libro está prestado, aparecerá el botón de devolver, el cual devolverá el préstamo y nos llevará al componente de ListaPréstamos; y que por el contrario, si el libro no estaba ya prestado, nos llevará al componente de CrearPrestamo para poder crear el dicho préstamo, componente que desarrollaremos ahora.

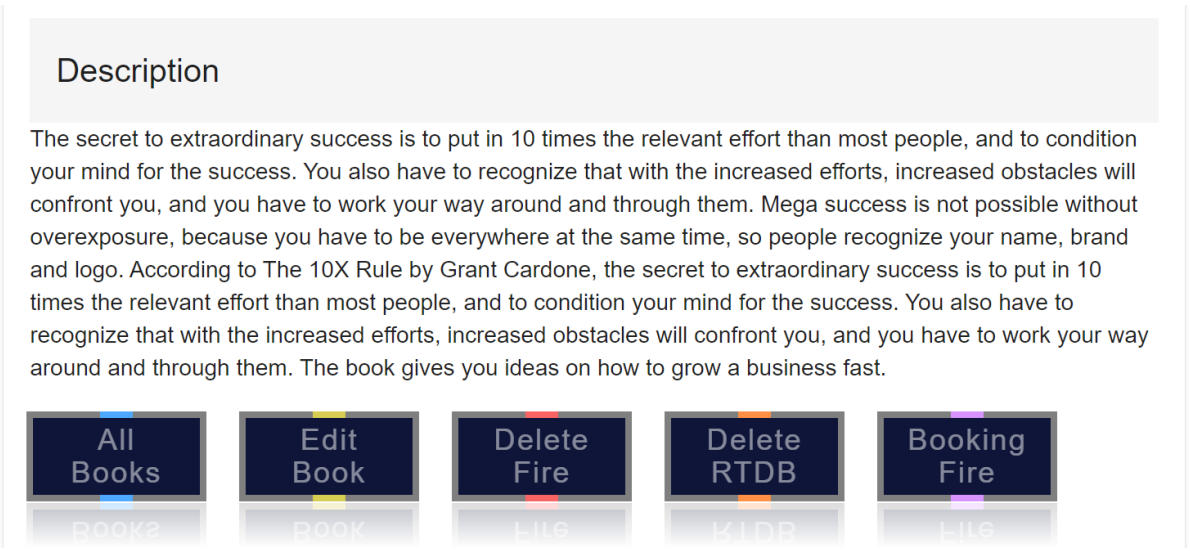


Vemos que si el libro aún está prestado, el botón de EditBook se deshabilita, y aparece el de DevolverLibro.

Nota: recordar que para las variaciones de estilos, consultar mi GitHub.



Comprobamos en el ListaPrestamos que efectivamente se ha devuelto porque se ha generado la fecha de devolución, y enel BookDetails se ha vuelto a habilitar el botón de EditBook y el DevolverLibro se ha cambiado por el de BookingFire.



Y si ahora pulsamos el botón de BookingFire nos llevará a nosotros y a tal objeto de libro hacia el componente de …

20. Crear Préstamo

- 20.1. Para el crear-prestamo.component.ts, recibimos el libro por la ruta, recolectamos todos los clientes de firestore, y creamos el objeto del préstamo …

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { NavigationExtras, Router } from '@angular/router';
import { FirestoreService } from 'src/app/services/firestore.service';
import { map } from 'rxjs/operators';
import { Cliente } from 'src/app/interfaces/cliente';
import { ClienteV2Service } from 'src/app/services/cliente-v2.service';
import firebase from 'firebase/compat/app';
import Timestamp = firebase.firestore.Timestamp;
import { PrestamosService } from 'src/app/services/prestamos.service';

@Component({
  selector: 'app-crear-prestamo-fire',
  templateUrl: './crear-prestamo-fire.component.html',
  styleUrls: ['./crear-prestamo-fire.component.css']
})
export class CrearPrestamoFireComponent implements OnInit {

  book: any;
  navigationExtras: NavigationExtras = {
    state: {
      value: null
    }}

  public bookingForm: any;
  customers: any;

  constructor(
    private router: Router,
    private firestoreService: FirestoreService,
    private clientesV2Service: ClienteV2Service,
    private prestamoService: PrestamosService
  ) {
    //hay que hacer esto en el constructor porque si se hiciera todo en el ngOnInit(), obtendríamos el objetivo null debido a que la navegación muere al crearse
```

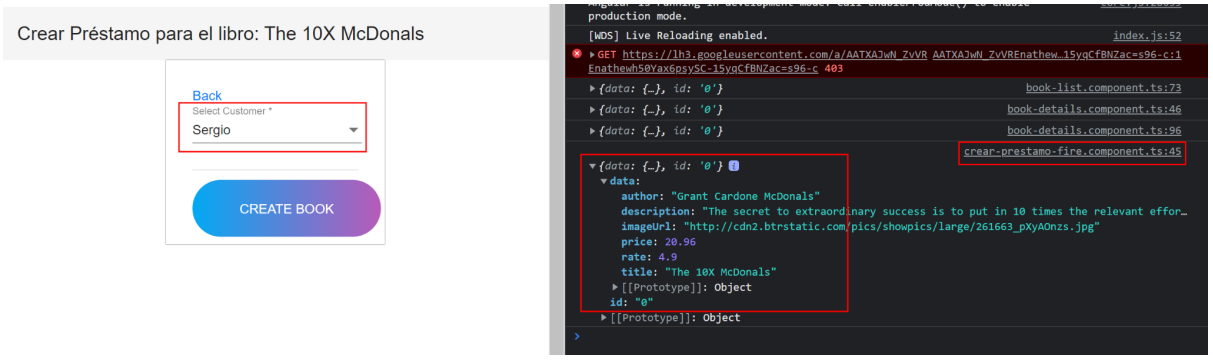
```
//https://stackoverflow.com/questions/54891110/router-getcurrentnavigation-always-returns-null
    const navigation = this.router.getCurrentNavigation();
    this.book = navigation?.extras?.state;

    this.bookingForm = new FormGroup({
      customer: new FormControl(null, Validators.required),
    });
  }

  ngOnInit(): void {
    console.log(this.book);

    if (this.book == null) {
      //asi controlo que nadie se intente colar
      this.router.navigate(["lista-prestamos"]);
    }

    this.getAllCustomers();


  }

  getAllCustomers(): void {
    this.clientesV2Service.getCustomers()
      .subscribe((res) => {
        this.customers = res.map((customer: any) => {
          return {
            data: customer.payload.doc.data(),
            id: customer.payload.doc.id
          } as Cliente;
        });
        //console.log(this.customers);
      });
  }
```

```
  generateBooking() {
    let prestamo: any = {};
    let formData: any = null;

    if(this.bookingForm.valid) {
      formData = this.bookingForm.value.customer;
      prestamo.customer_id = formData.id;
      prestamo.customer_name = formData.data.name;
      prestamo.book_id = this.book.id;
      prestamo.book_title = this.book.data.title;
      prestamo.date_booking = Timestamp.now();

      this.prestamoService.createPrestamo(prestamo);
      this.router.navigate(["lista-prestamos"]);

      console.log(prestamo);
    }
  }
}
```

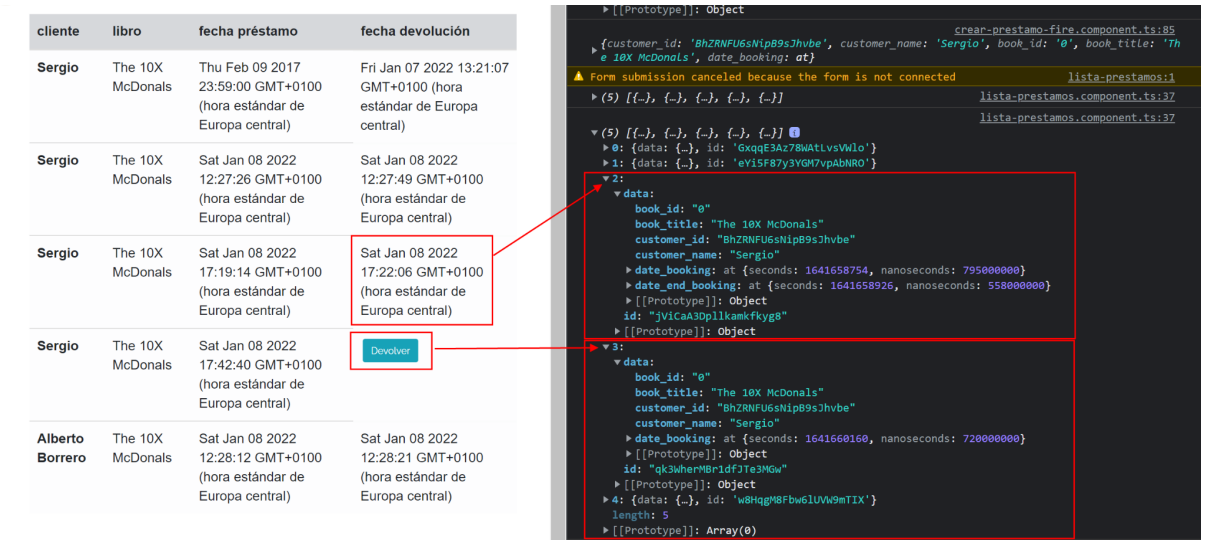
- 20.2. Y para su estructura html un form muy sencillo con un <select> <option> …

```
<!-- <p>crear-prestamo-fire works!</p> -->

<mat-toolbar>Crear Préstamo para el libro: {{ book.data.title
}}</mat-toolbar>

<div class="add-book-form" *ngIf="customers != null || customers !=
undefined">
    <a [routerLink]="['/book-list']">Back</a>

    <form #myForm="ngForm" [formGroup]="bookingForm">

        <mat-form-field>
            <mat-select placeholder="Select Customer" name="customer"
id="customer" required formControlName="customer">
                <ng-container>
                    <mat-option *ngFor="let customer of customers"
[value]="customer">{{ customer.data.name }}</mat-option>
                </ng-container>
            </mat-select>
        </mat-form-field>
```

```
        <br>
        <hr>

        <button style="margin-right: 30px" mat-raised-button
color="warn" (click)="generateBooking()"
[disabled]="!myForm.form.valid">
            Create Book
        </button>
        <br>

    </form>
</div>
```



Haciendo la prueba, vemos que al llegar a CrearPrestamo, se recibe el objeto del libro en cuestión, y vuelvo a elegir a Sergio de entre todos los clientes, y al pulsar el botón de Booking Book …



… nos lleva a la ListaPrestamos y vemos como el objeto del préstamo se ha creado perfectamente.

21. What’s Next ??

Con todo esto, concluímos y damos por terminado este proyecto de Biblioteca, pero…
¿ qué sería lo siguiente que se pudiera seguir desarrollando en él ?

En mi opinión, como esta web está orientada y enfocada para ser usada por el bibliotecario o cualquier otro cargo correspondiente a las tareas de administración de la recepción de una biblioteca, podríamos seguir desarrollando la web en base a diferentes roles, es decir, se podría llegar a diferenciar entre el rol de empleado y el rol de un cliente.

Para el rol del empleado, es todo lo que hemos hecho en este proyecto, pero se podría hacer un rol de cliente con acceso a nuevos componentes que podríamos desarrollar para que el cliente pudiera informarse acerca de nuevo libros que entrasen, eventos de la biblioteca, y demás temas propios del interés de un lector, mientras que le restringiríamos el acceso a los componentes propios de CRUD, a los que el rol de empleado sí tendría acceso