# Open Stack Development

Week 2 Worksheet – Serverless functions implementation of Post, Get, Delete

## Learning Outcomes

- Implement  serverless functions for a Books database using
  - Post, Get, Delete
- Using Postman
- Angular application connecting to these serverless functions

## Get, Post, Delete, Put

- **GET** retrieves the representation of the resource at a specified URI
- **POST** creates a new resource
- **DELETE** deletes a resource at a specified URI
- **PUT** updates a resource at a specified URI

- We use POSTMAN as an API client to test the serverless functions we create.

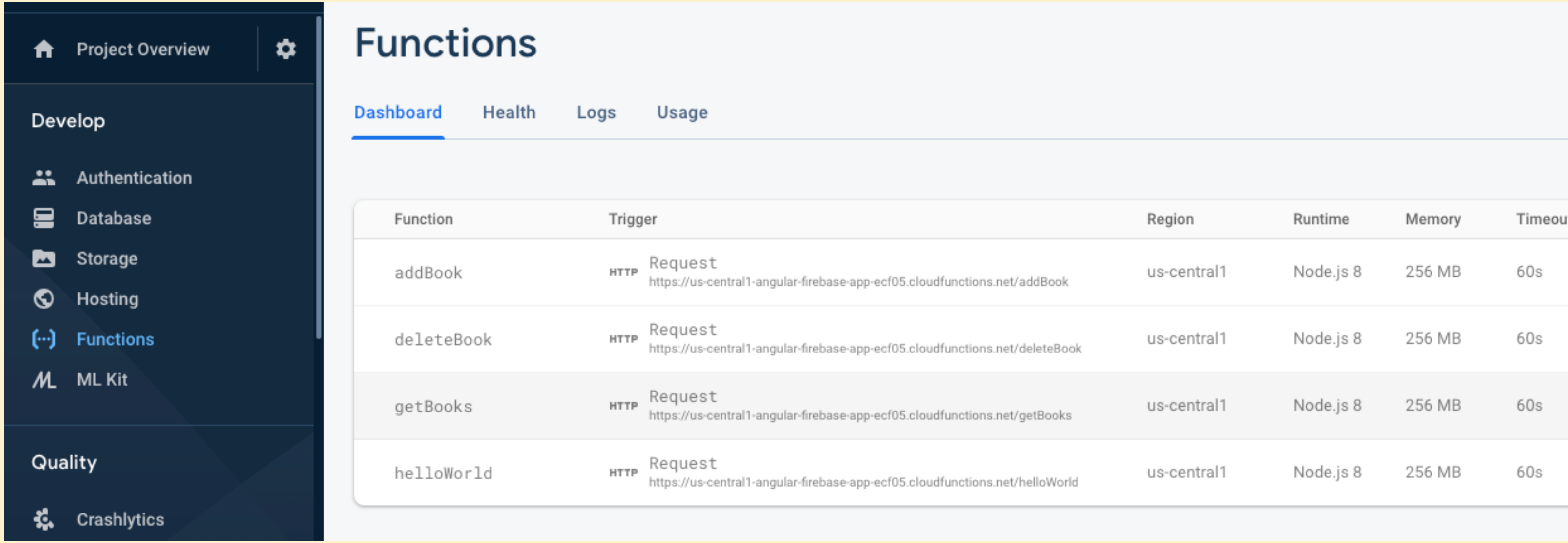
# Firebase Serverless Functions

- Create a new angular project firebase-sf-app and then from CLI do the following
  - `npm install firebase @angular/fire`
  - `npm install firebase-tools`
  - `firebase login` (for authentication)
  - `firebase init functions`
    - You will select an existing project you have setup in Firebase console
    - When asked what language you want to use cloud functions – this time choose Javascript rather than Typescript
    - Select the defaults for the remainder of the initialization

# Firebase Serverless Functions

- Replace /functions/index.js with index.js from Week  in Moodle – saves a lot of typing ☺
- From the CLI
  - `firebase deploy` (initial deployment) <u>OR</u>
  - `firebase deploy --only functions` <u>OR</u>
  - `firebase deploy --only functions:<functionname>` (only redeploys named function)

# Check Firebase >> your project >> functions

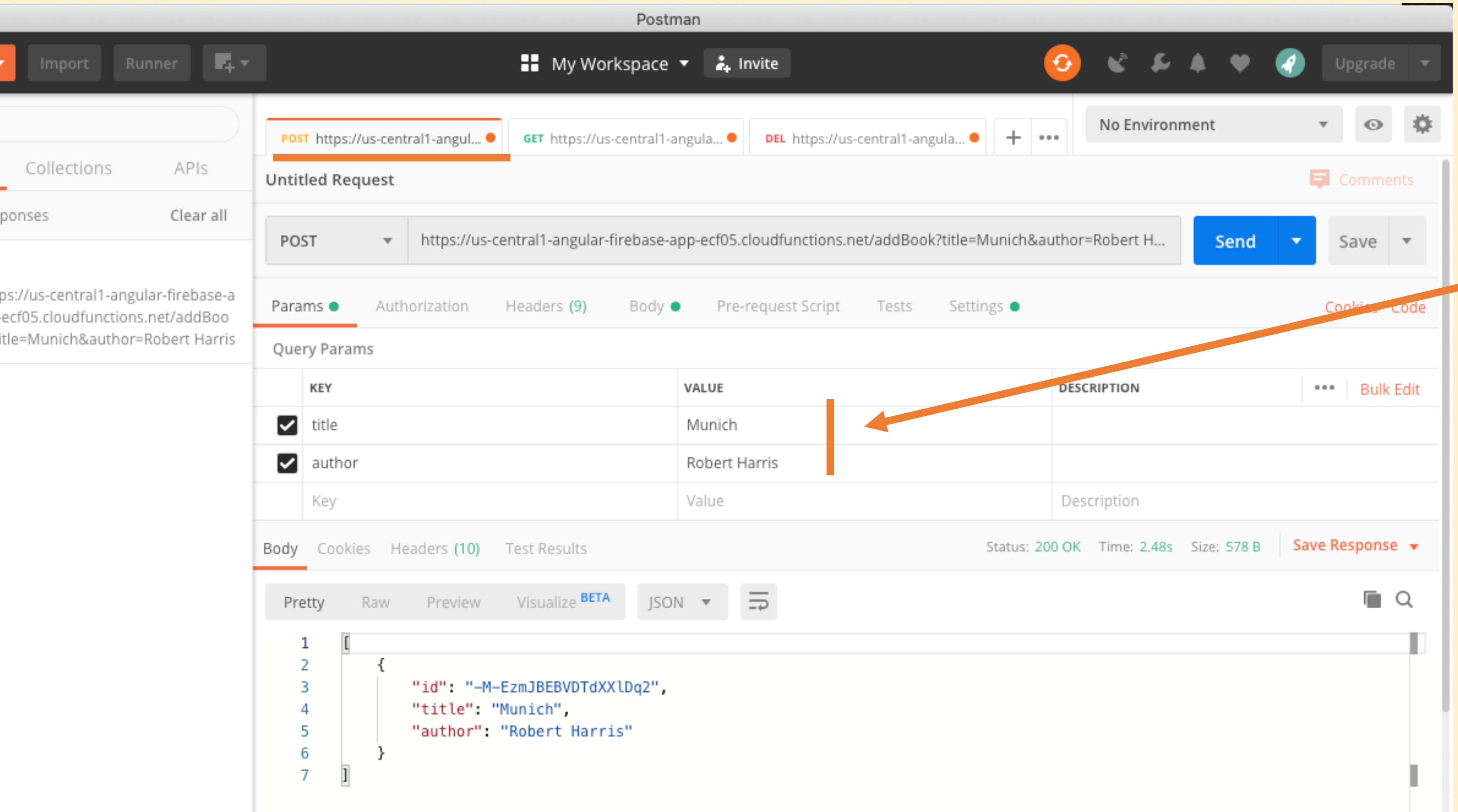


## Functions

Dashboard Health Logs Usage

| Function | Trigger | Region | Runtime | Memory | Timeou |
|---|---|---|---|---|---|
| addBook | HTTP Request https://us-central1-angular-firebase-app-ecf05.cloudfunctions.net/addBook | us-central1 | Node.js 8 | 256 MB | 60s |
| deleteBook | HTTP Request https://us-central1-angular-firebase-app-ecf05.cloudfunctions.net/deleteBook | us-central1 | Node.js 8 | 256 MB | 60s |
| getBooks | HTTP Request https://us-central1-angular-firebase-app-ecf05.cloudfunctions.net/getBooks | us-central1 | Node.js 8 | 256 MB | 60s |
| helloWorld | HTTP Request https://us-central1-angular-firebase-app-ecf05.cloudfunctions.net/helloWorld | us-central1 | Node.js 8 | 256 MB | 60s |

# Postman – Post, use No Auth, 2 parameters



https://us-central1-angular-firebase-app-ecf05.cloudfunctions.net/addBook?title=Munich&author=Robert H...

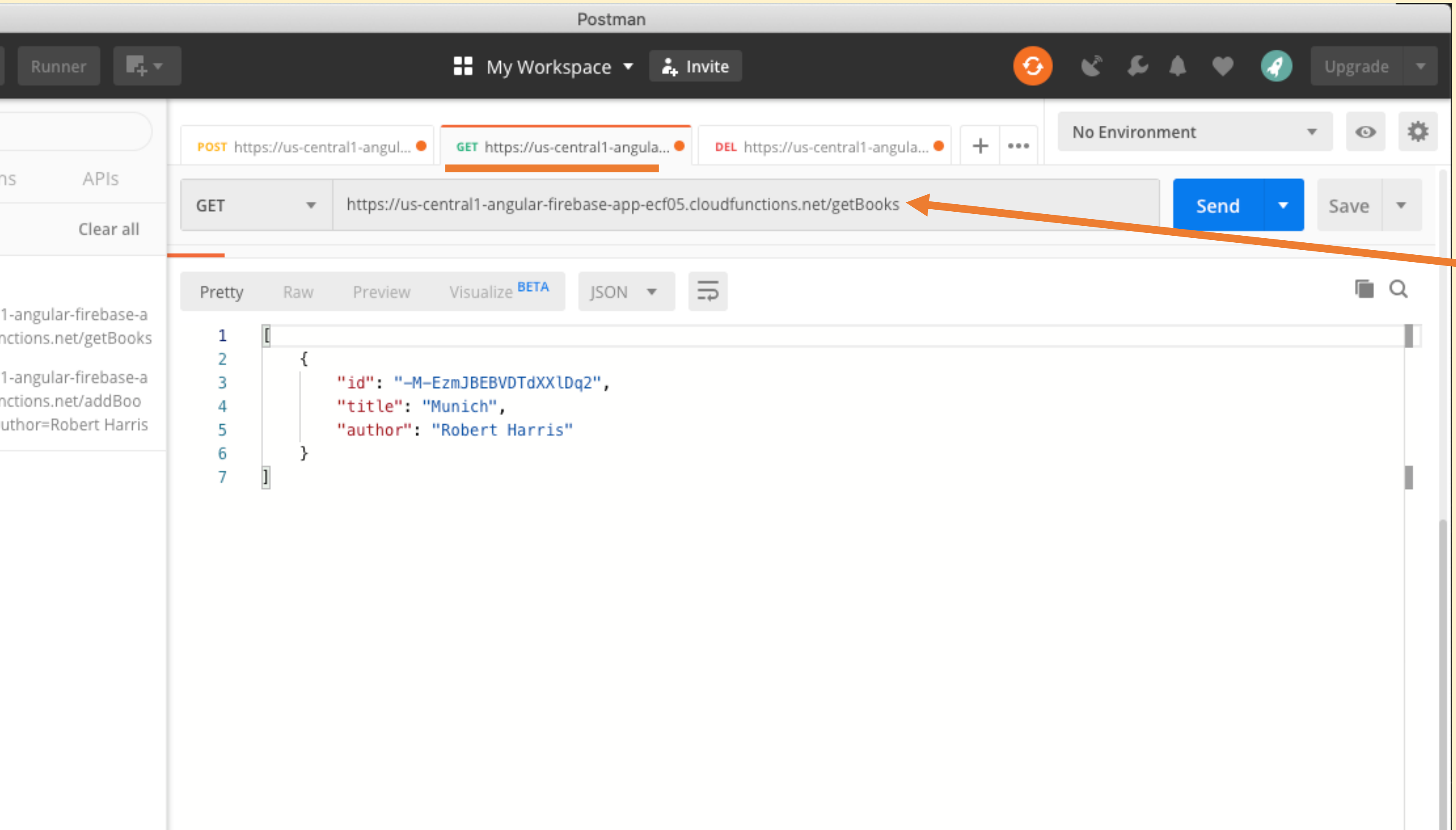
| | KEY | VALUE | DESCRIPTION |
|---|---|---|---|
| ✓ | title | Munich | |
| ✓ | author | Robert Harris | |
| | Key | Value | Description |

Status: 200 OK Time: 2.48s Size: 578 B

```
[
    {
        "id": "-M-EzmJBEBVDTdXXlDq2",
        "title": "Munich",
        "author": "Robert Harris"
    }
]
```

addBook
Adding author and title

# Postman – Get, use No Auth, no parameters

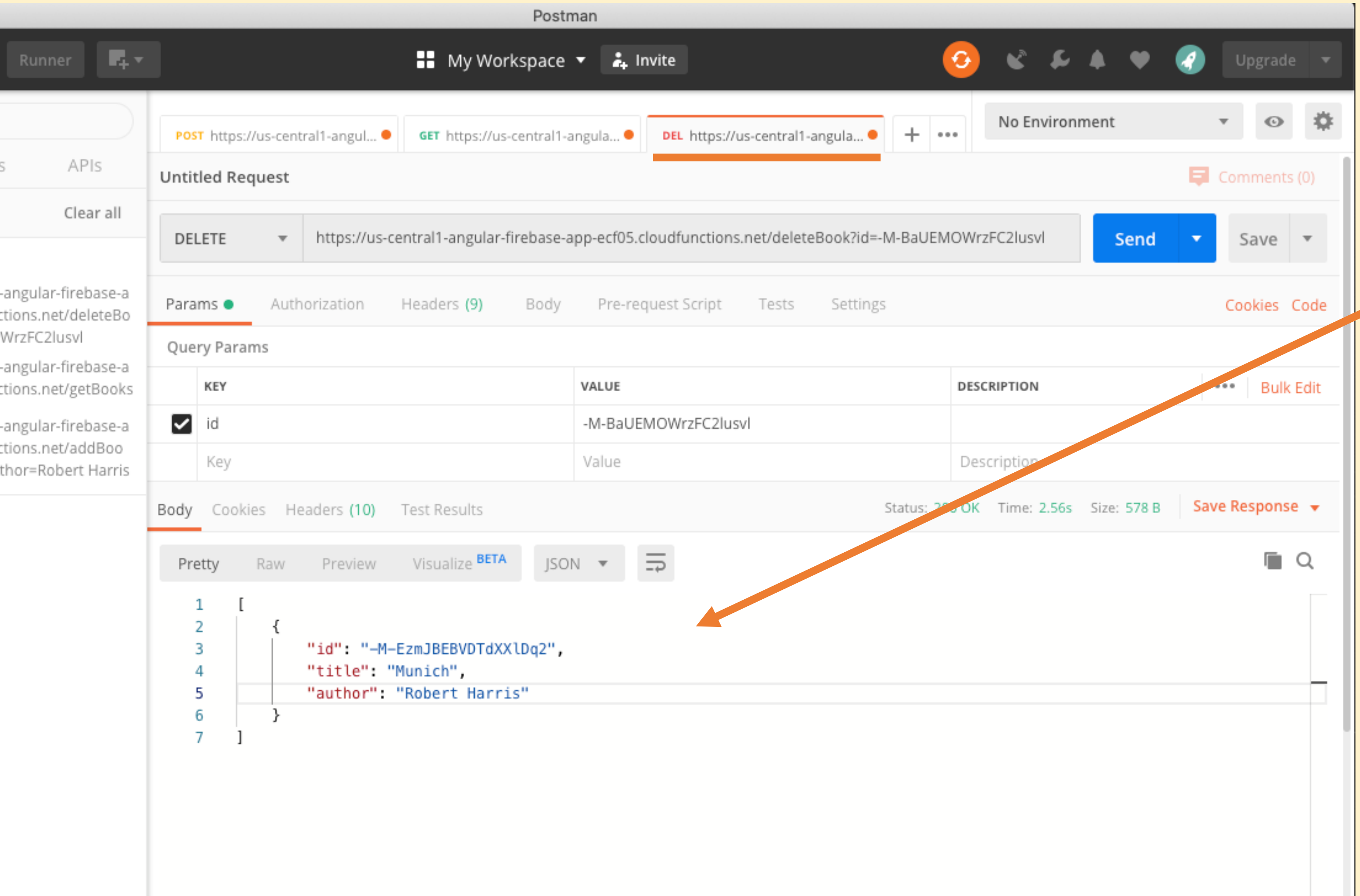


GET https://us-central1-angular-firebase-app-ecf05.cloudfunctions.net/getBooks

```
[
    {
        "id": "-M-EzmJBEBVDTdXXlDq2",
        "title": "Munich",
        "author": "Robert Harris"
    }
]
```

getBooks
No parameters, returns all details

# Postman – Del, use No Auth, 1 parameter



DELETE https://us-central1-angular-firebase-app-ecf05.cloudfunctions.net/deleteBook?id=-M-BaUEMOWrzFC2lusvl

| KEY | VALUE | DESCRIPTION |
|---|---|---|
| ✓ id | -M-BaUEMOWrzFC2lusvl | |
| Key | Value | Description |

Status: 200 OK Time: 2.56s Size: 578 B

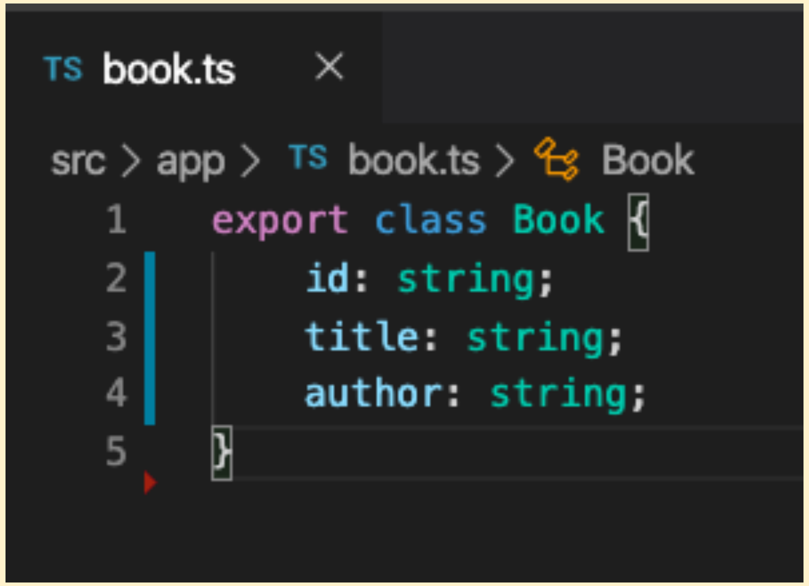
```
[
    {
        "id": "-M-EzmJBEBVDTdXXlDq2",
        "title": "Munich",
        "author": "Robert Harris"
    }
]
```

-angular-firebase-a
ctions.net/deleteBo
WrzFC2lusvl
-angular-firebase-a
ctions.net/getBooks
-angular-firebase-a
ctions.net/addBoo
thor=Robert Harris

deleteBook
Id as parameter, returns remaining books

# Building a simple Angular App

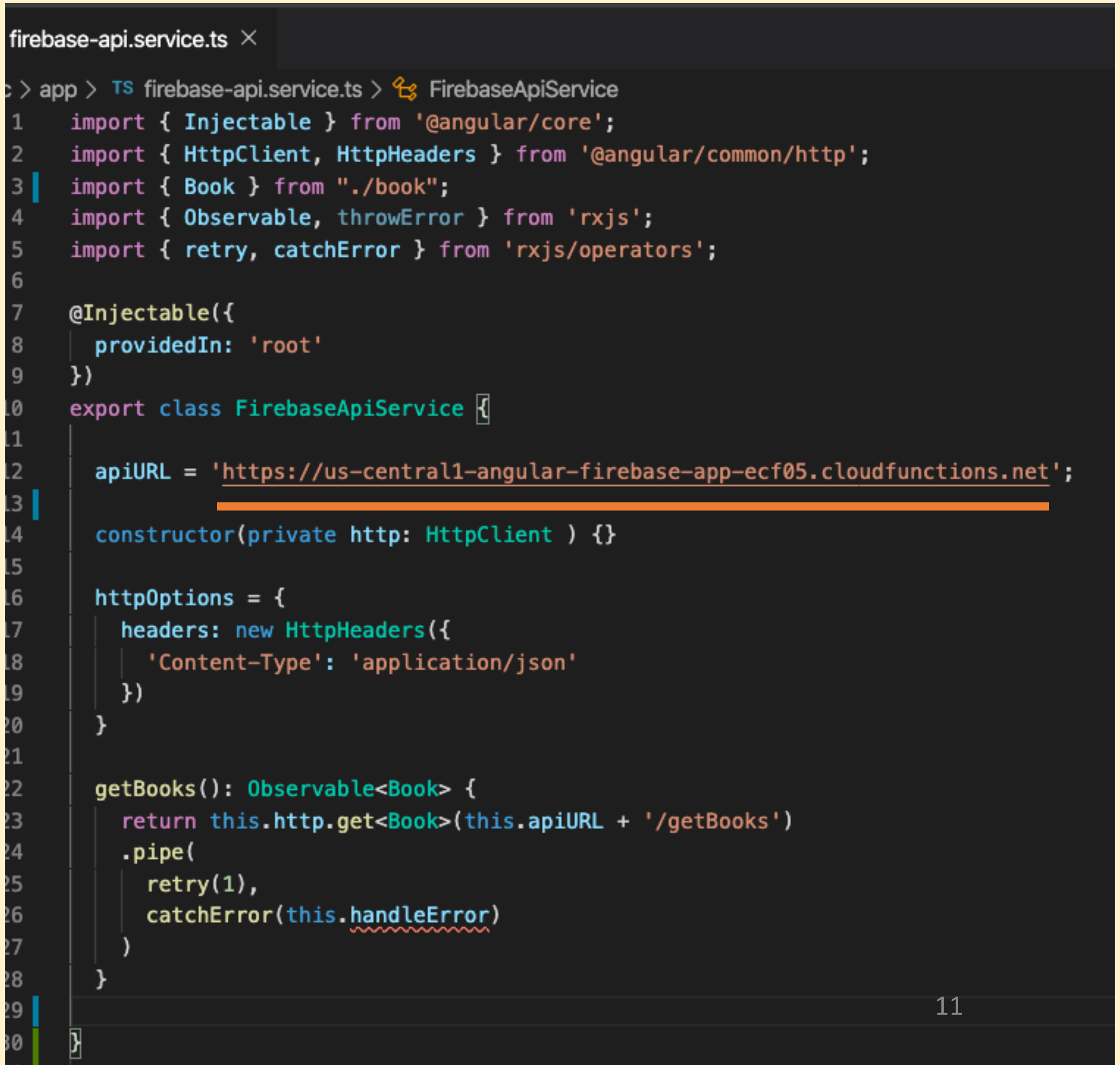
- Create a new angular app
- Create a class for a book in book.ts



```
export class Book {
    id: string;
    title: string;
    author: string;
}
```

# firebase-api.service.ts

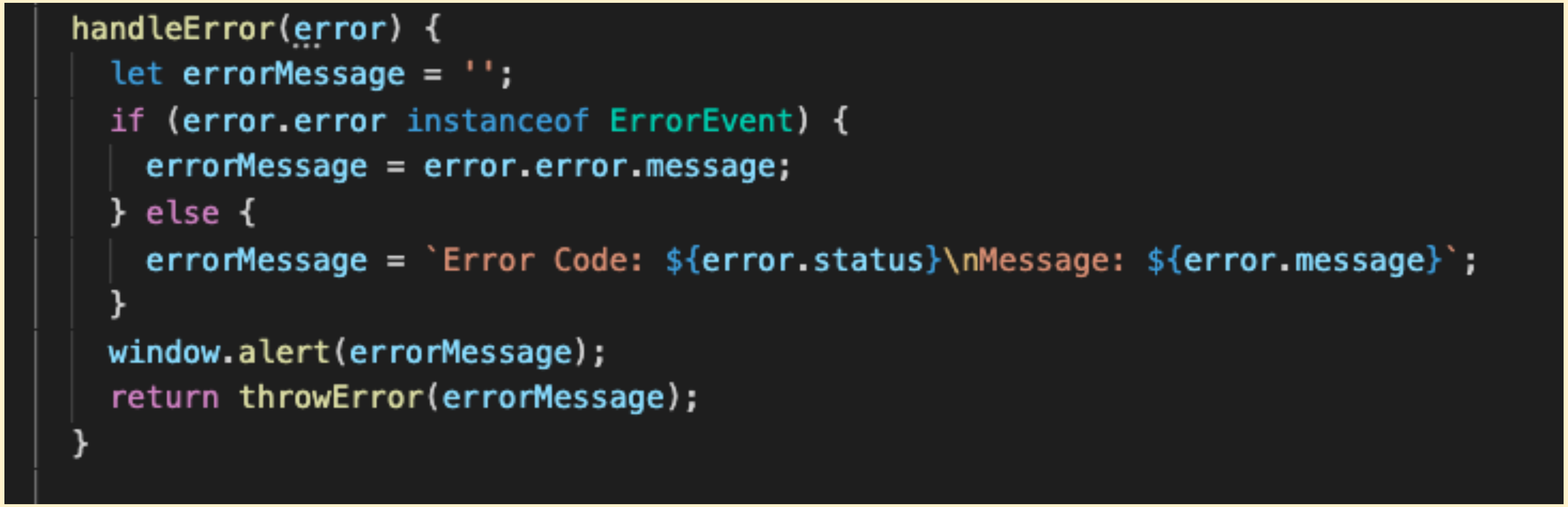
- Create a service for Firebase and the http calls we are going to make to our serverless functions
  - `ng generate service firebase-api`
- Replace apiURL with your own instance



```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Book } from "./book";
import { Observable, throwError } from 'rxjs';
import { retry, catchError } from 'rxjs/operators';

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

  apiURL = 'https://us-central1-angular-firebase-app-ecf05.cloudfunctions.net';

  constructor(private http: HttpClient ) {}

  httpOptions = {
    headers: new HttpHeaders({
      'Content-Type': 'application/json'
    })
  }

  getBooks(): Observable<Book> {
    return this.http.get<Book>(this.apiURL + '/getBooks')
    .pipe(
      retry(1),
      catchError(this.handleError)
    )
  }

}
```

# firebase-api.service.ts

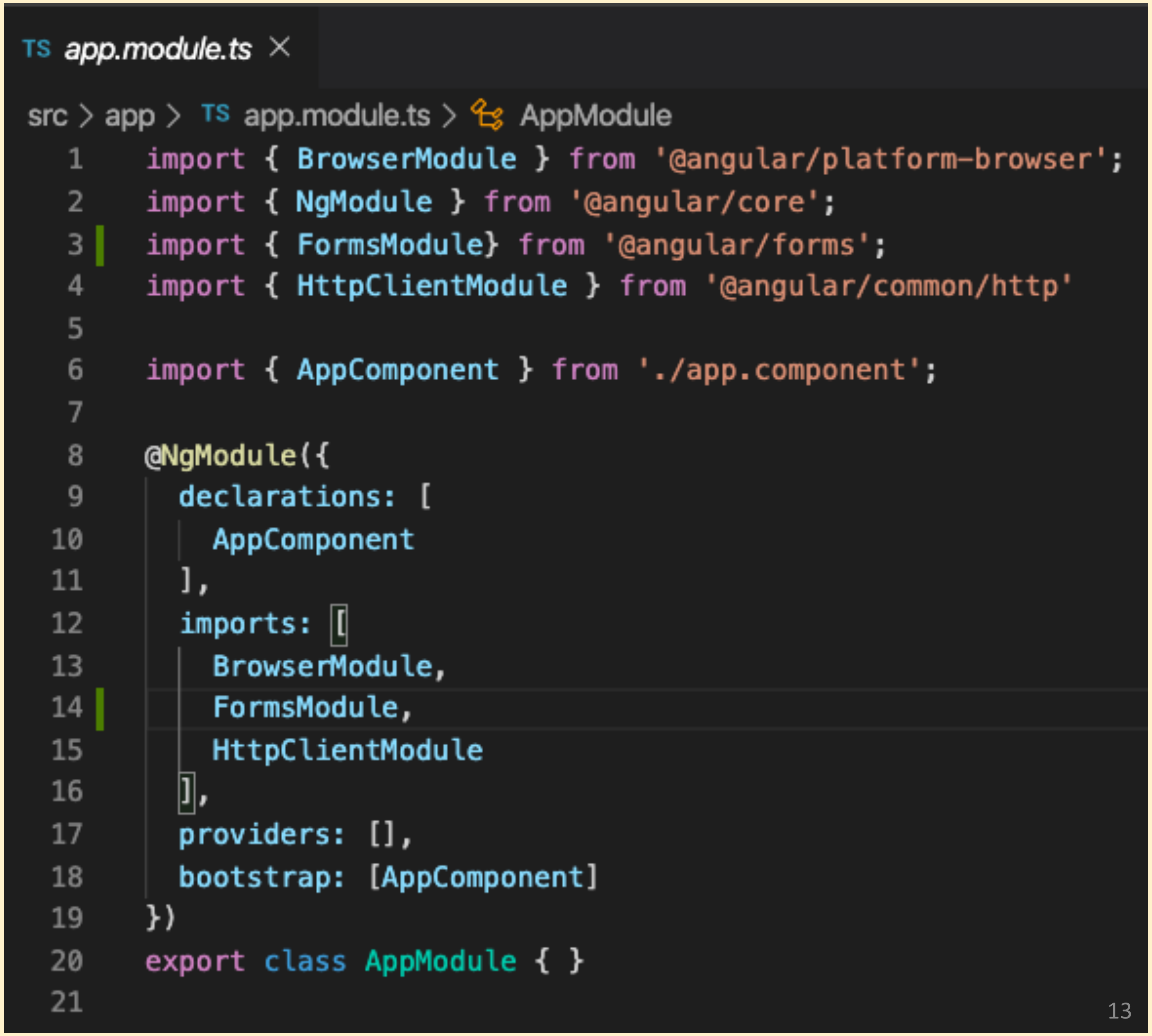
- Add this to handle any errors that may be returned..



```
handleError(error) {
  let errorMessage = '';
  if (error.error instanceof ErrorEvent) {
    errorMessage = error.error.message;
  } else {
    errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;
  }
  window.alert(errorMessage);
  return throwError(errorMessage);
}
```

# App.module.ts

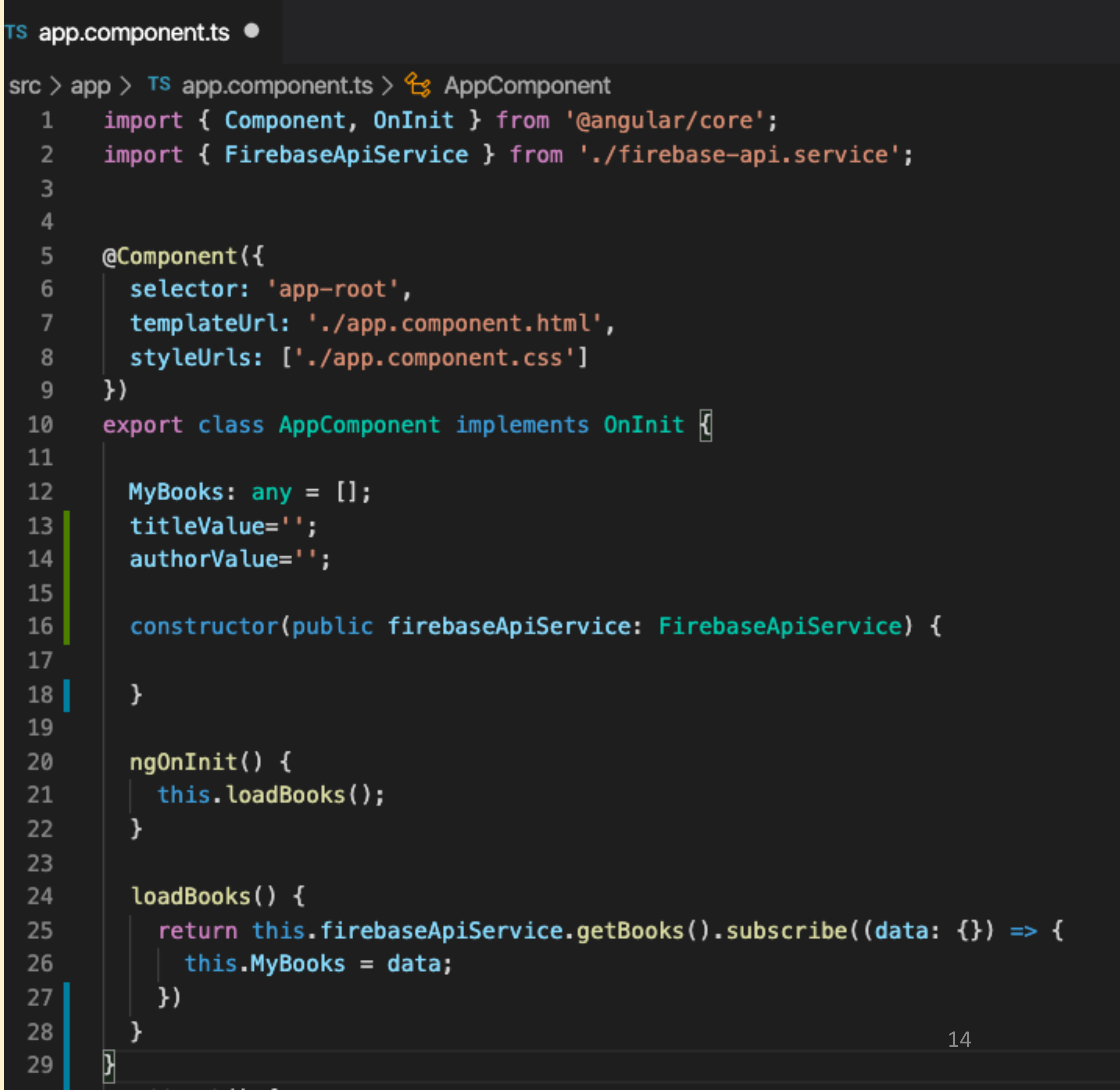
- We need
  - FormsModule
  - HttpClientModule



```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule} from '@angular/forms';
import { HttpClientModule } from '@angular/common/http'

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# app.component.ts

- We inject our FirebaseAPiService in the constructor
- ngOnInit then calls loadBooks
- Mybooks will be given the details from database via subscription in loadBooks()
- titleValue and authorValue are inputs for a form we will add later

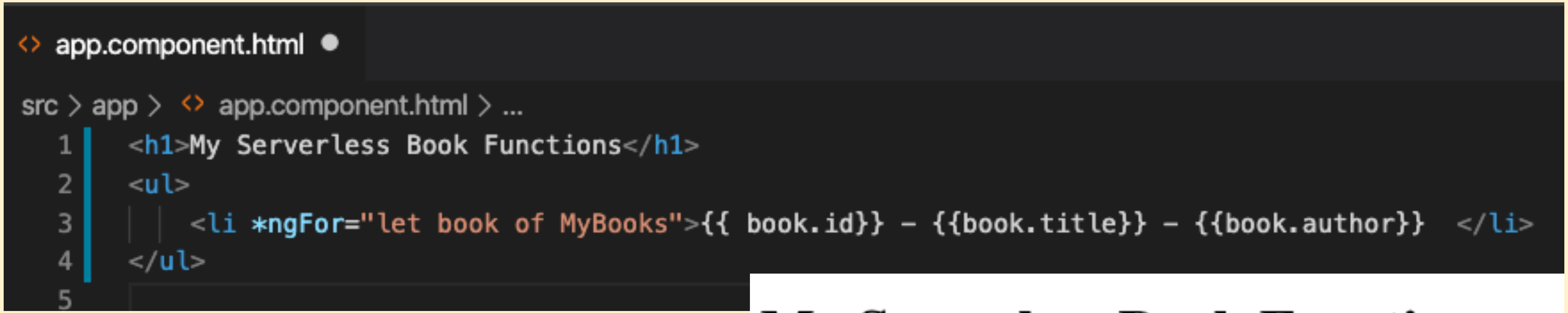


app.component.ts

src > app > app.component.ts > AppComponent

```
import { Component, OnInit } from '@angular/core';
import { FirebaseApiService } from './firebase-api.service';


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

  MyBooks: any = [];
  titleValue='';
  authorValue='';

  constructor(public firebaseApiService: FirebaseApiService) {

  }

  ngOnInit() {
    this.loadBooks();
  }

  loadBooks() {
    return this.firebaseApiService.getBooks().subscribe((data: {}) => {
      this.MyBooks = data;
    })
  }
}
```

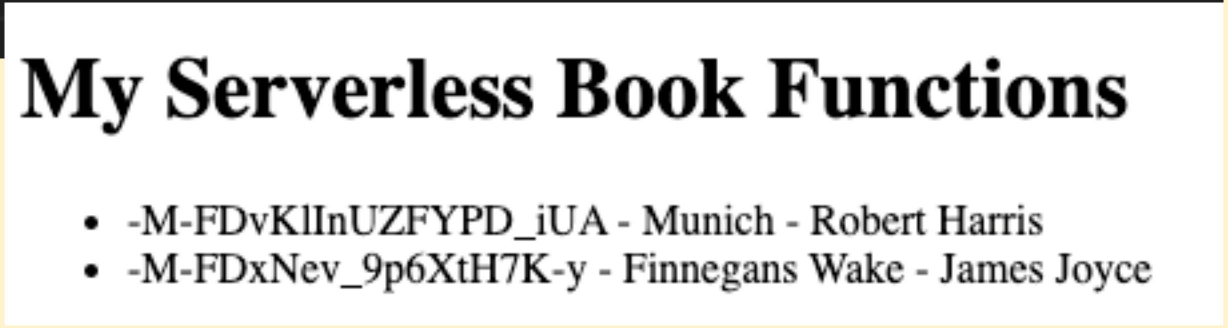
## app.component.html


app.component.html

src > app > app.component.html > ...

```
<h1>My Serverless Book Functions</h1>
<ul>
    <li *ngFor="let book of MyBooks">{{ book.id}} - {{book.title}} - {{book.author}}  </li>
</ul>
```

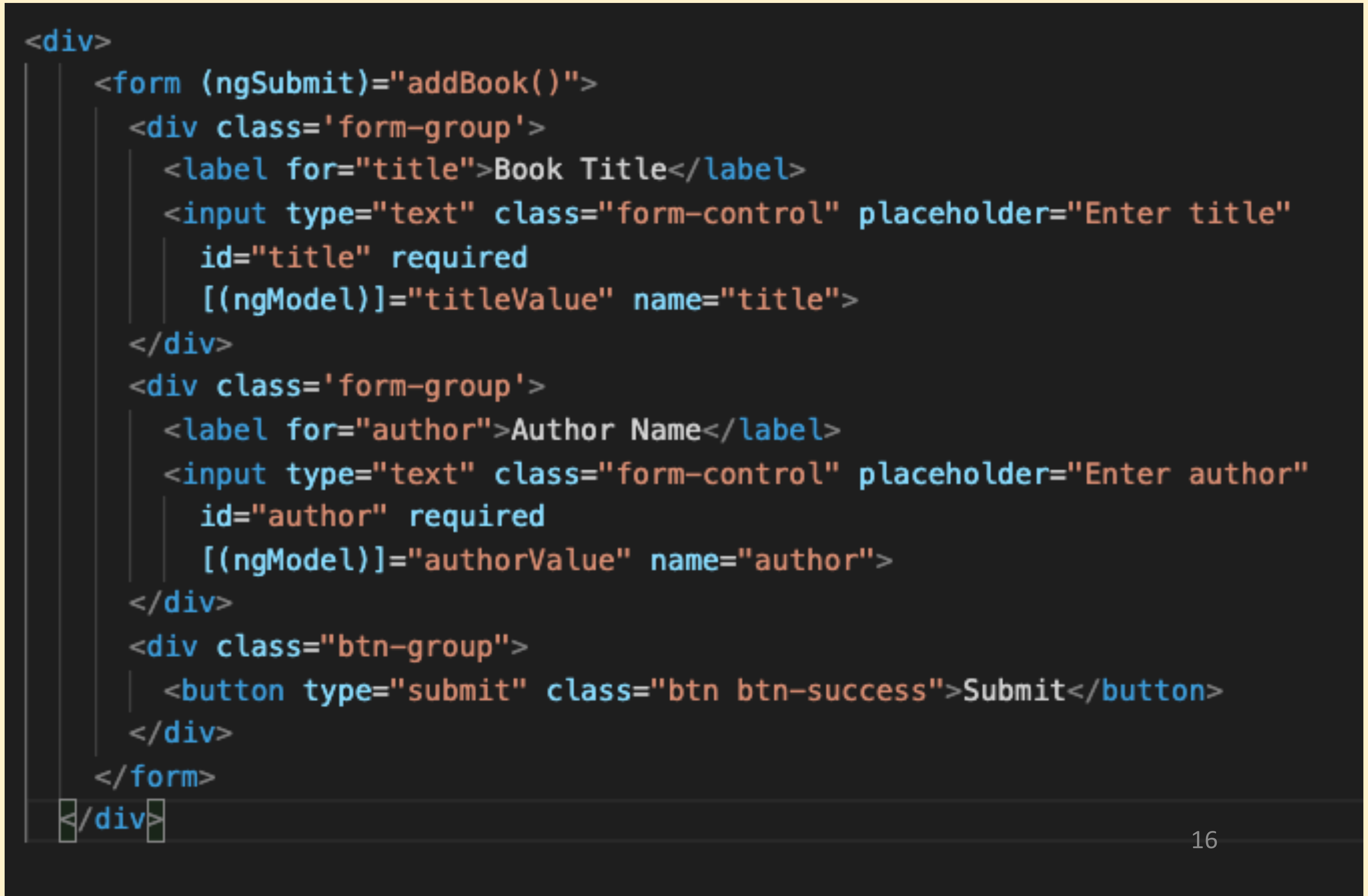
- Now check that you can see the books you added via Postman


My Serverless Book Functions

- -M-FDvKlInUZFYPD_iUA - Munich - Robert Harris
- -M-FDxNev_9p6XtH7K-y - Finnegans Wake - James Joyce

# Add Book (through our Angular app)

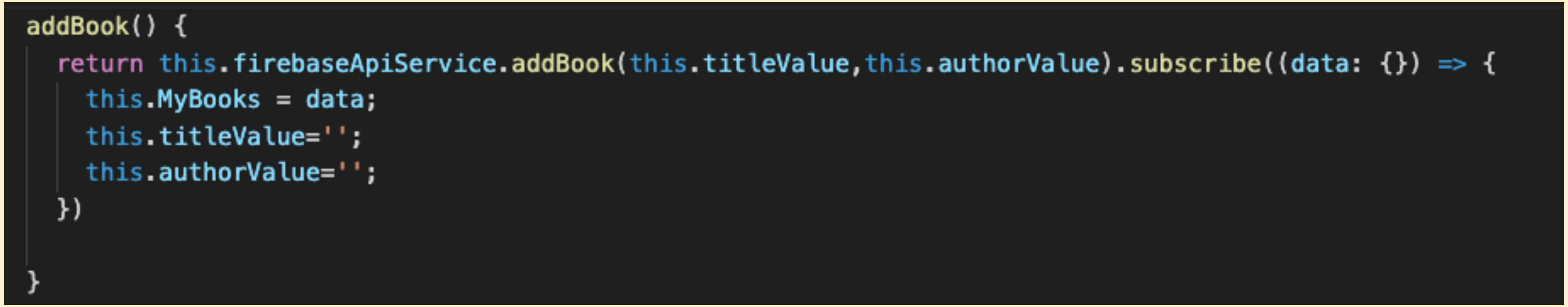
- app.component.html



```
<div>
    <form (ngSubmit)="addBook()">
      <div class='form-group'>
        <label for="title">Book Title</label>
        <input type="text" class="form-control" placeholder="Enter title"
          id="title" required
          [(ngModel)]="titleValue" name="title">
      </div>
      <div class='form-group'>
        <label for="author">Author Name</label>
        <input type="text" class="form-control" placeholder="Enter author"
          id="author" required
          [(ngModel)]="authorValue" name="author">
      </div>
      <div class="btn-group">
        <button type="submit" class="btn btn-success">Submit</button>
      </div>
    </form>
  </div>
```

# Add Book

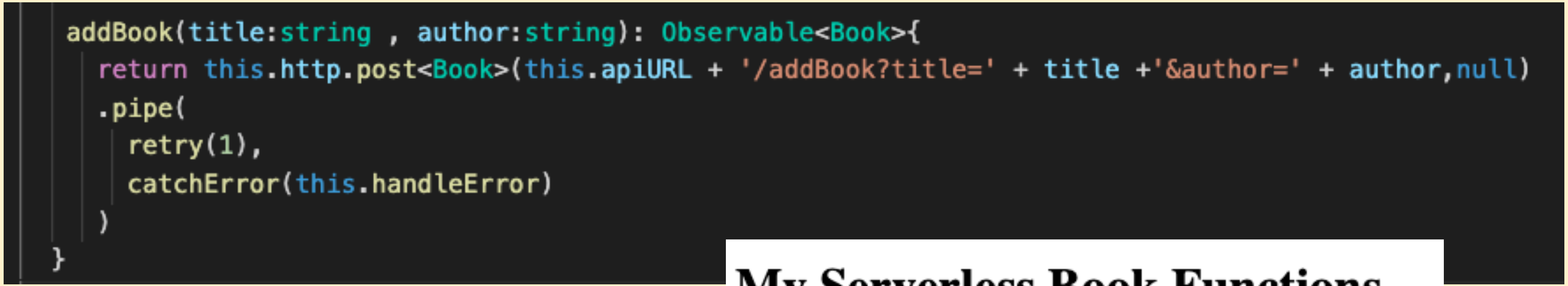
- app.component.ts



```
addBook() {
  return this.firebaseApiService.addBook(this.titleValue,this.authorValue).subscribe((data: {}) => {
    this.MyBooks = data;
    this.titleValue='';
    this.authorValue='';
  })

}
```

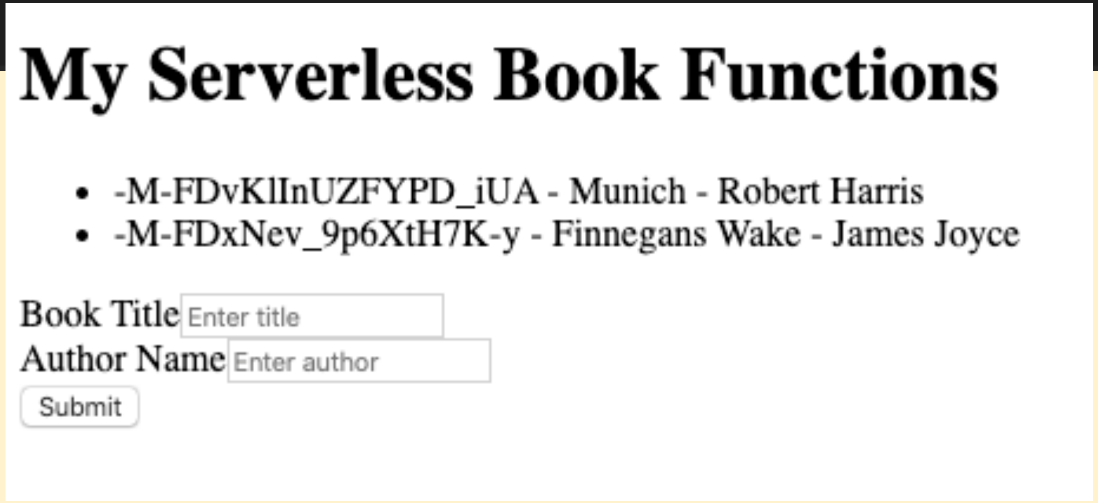
## Add Book

- firebase-api.service.ts



```
addBook(title:string , author:string): Observable<Book>{
  return this.http.post<Book>(this.apiURL + '/addBook?title=' + title +'&author=' + author,null)
  .pipe(
    retry(1),
    catchError(this.handleError)
  )
}
```

- Test it – can you add a book?



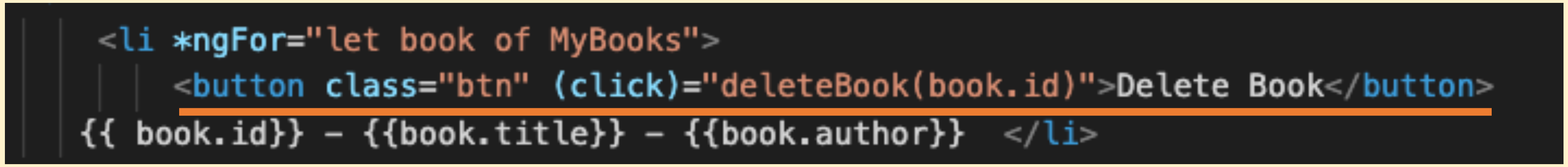
# My Serverless Book Functions

- -M-FDvKlInUZFYPD_iUA - Munich - Robert Harris
- -M-FDxNev_9p6XtH7K-y - Finnegans Wake - James Joyce

Book Title Enter title
Author Name Enter author
Submit

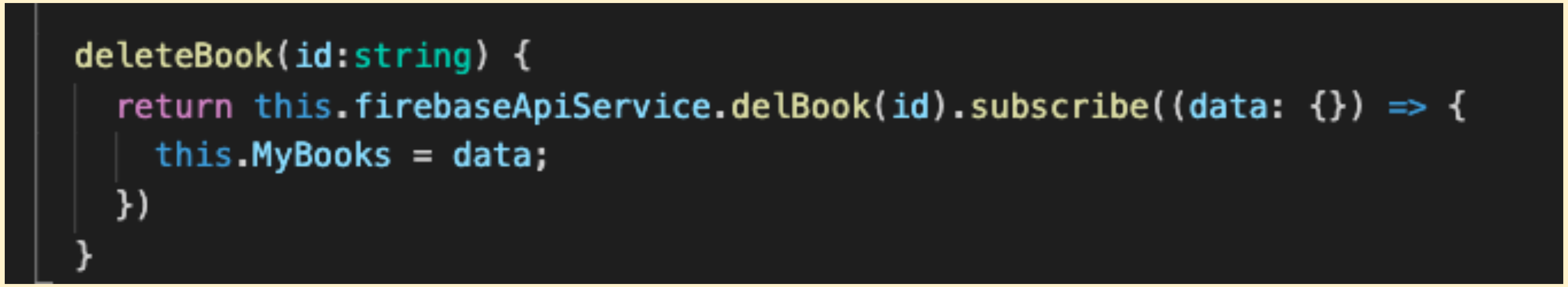
## Delete Book (through the Angular App)

- app.component.html



```
<li *ngFor="let book of MyBooks">
    <button class="btn" (click)="deleteBook(book.id)">Delete Book</button>
{{ book.id}} - {{book.title}} - {{book.author}}  </li>
```

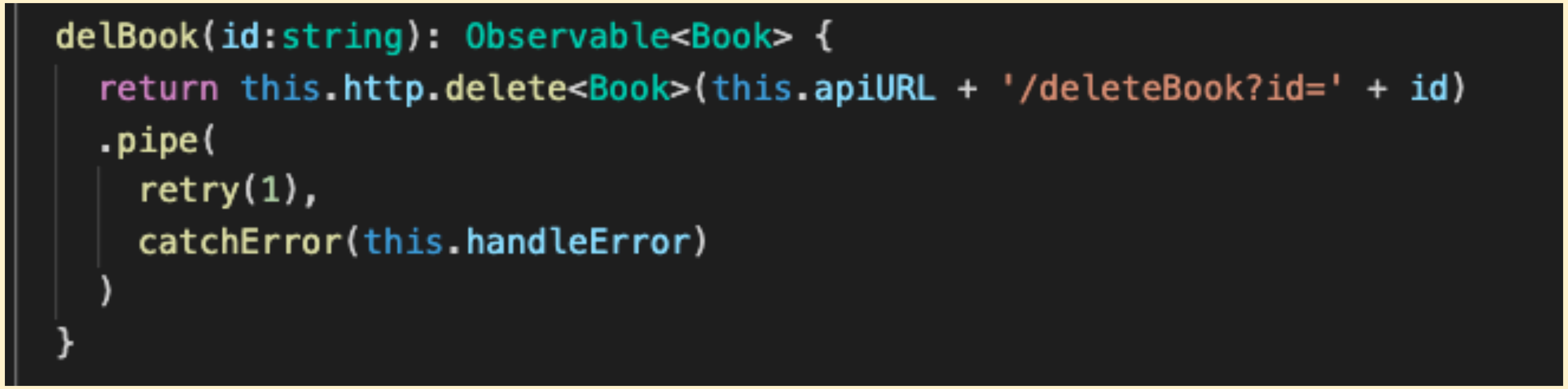
- app.component.ts



```
deleteBook(id:string) {
  return this.firebaseApiService.delBook(id).subscribe((data: {}) => {
    this.MyBooks = data;
  })
}
```

# Delete Book

- firebase-api.service.ts



```
delBook(id:string): Observable<Book> {
  return this.http.delete<Book>(this.apiURL + '/deleteBook?id=' + id)
  .pipe(
    retry(1),
    catchError(this.handleError)
  )
}
```

- Test it – does it delete a book??