Human Computer Interaction

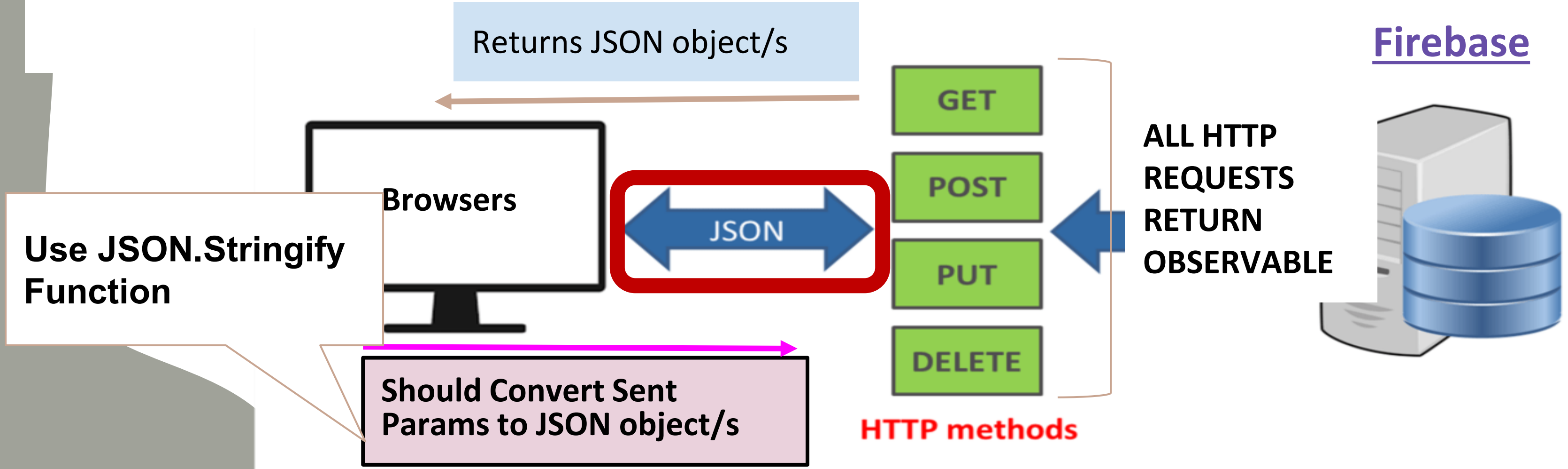
Lab 8 – Angular

Agenda

- HTTP Requests
- Firebase

HTTP Client

- The front-end of applications communicate with back-end services to get or send the data over HTTP protocol (For Example: API Request).
- This communication is done in Angular with the help of **HttpClient (Built-in Service), So** you need to import **HttpClientModule** in your app module.
- Inject the **HttpClient service in the constructor** of your service (**where you need to communicate with DB or API**).


Returns JSON object/s
Firebase
Browsers
JSON
GET
POST
PUT
DELETE
ALL HTTP REQUESTS RETURN OBSERVABLE
Use JSON.Stringify Function
Should Convert Sent Params to JSON object/s
HTTP methods

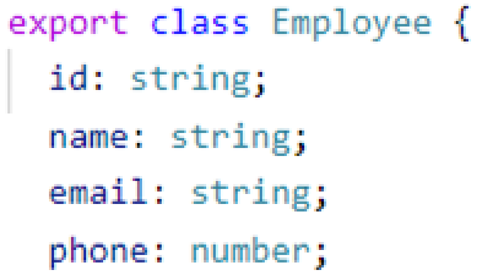
HTTP Requests

```
export class Employee {
      string;
    e: string;
  email: string;
  phone: number;
}
```

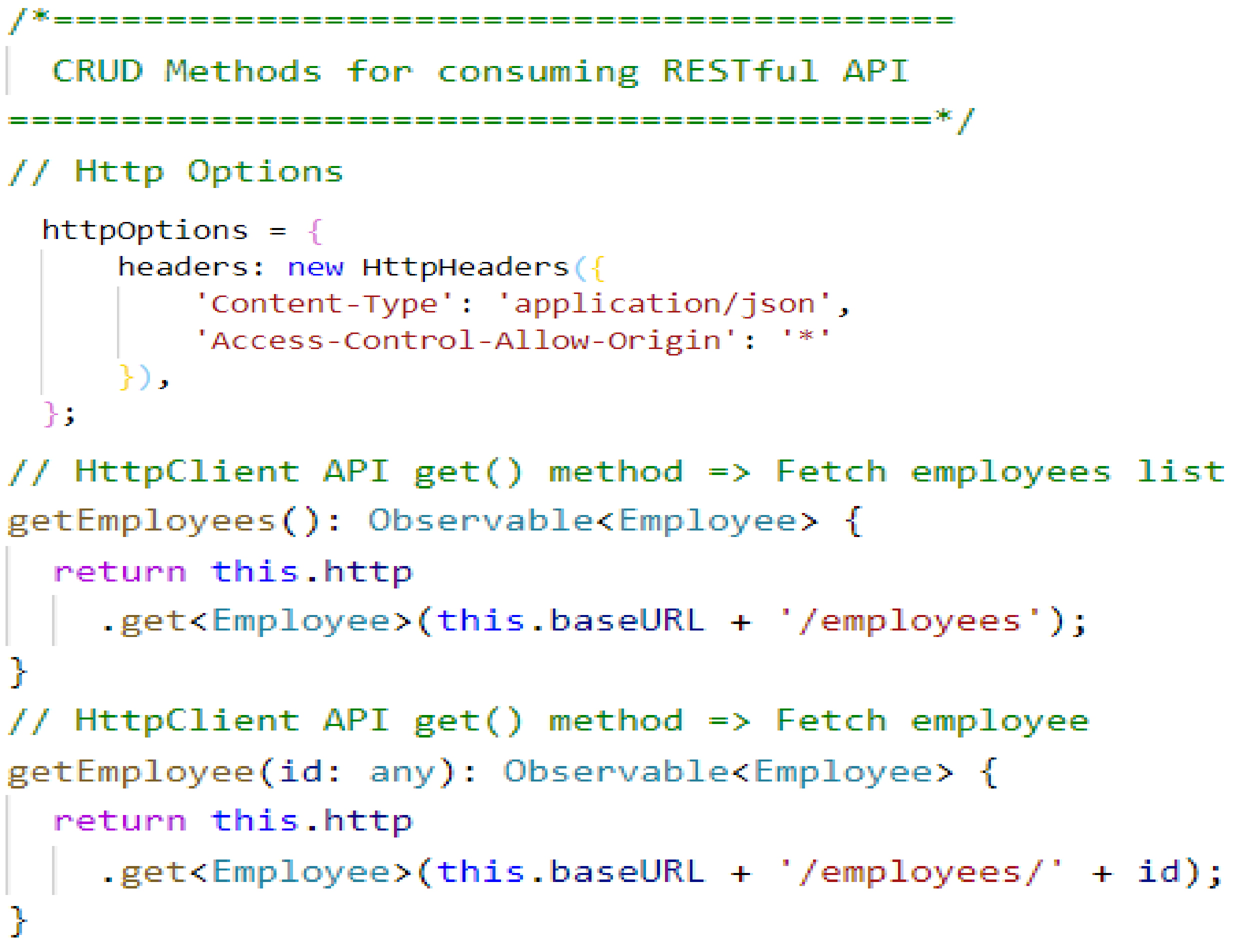
Can be replaced with Interface

```
import { Injectable } from '@angular/core';
import {
  HttpClient,
  HttpRequest,
  HttpEvent,
  HttpEventType
} from '@angular/common/http';

@Injectable()

export class RestApiService {
  // Define DB/API URL
  baseURL = 'https://katowulf-examples.firebaseio.com/incid/';
  constructor(private http: HttpClient) {}
```

HTTP Requests URLs

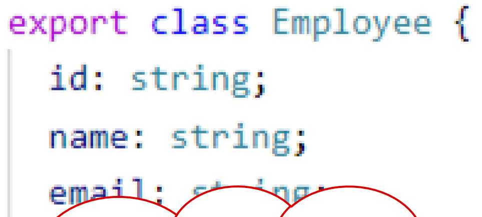


```
export class Employee {
  id: string;
  name: string;
  email: string;
  phone: number;
```



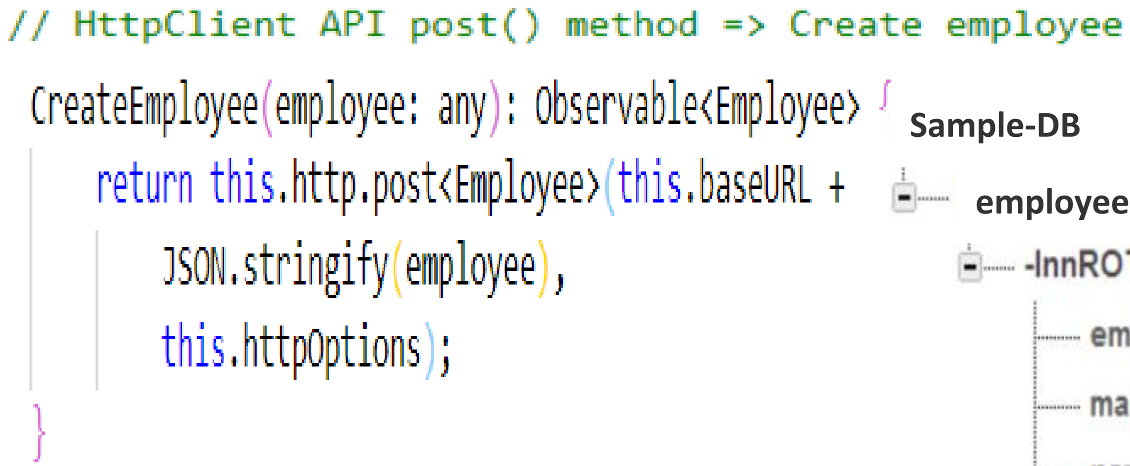
```
/*===================================
  CRUD Methods for consuming RESTful API
===================================*/
// Http Options

  httpOptions = {
    headers: new HttpHeaders({
        'Content-Type': 'application/json',
        'Access-Control-Allow-Origin': '*'
    }),
  };
// HttpClient API get() method => Fetch employees list
getEmployees(): Observable<Employee> {
  return this.http
    .get<Employee>(this.baseURL + '/employees');
}
// HttpClient API get() method => Fetch employee
getEmployee(id: any): Observable<Employee> {
  return this.http
    .get<Employee>(this.baseURL + '/employees/' + id);
}
```

HTTP Requests URLs

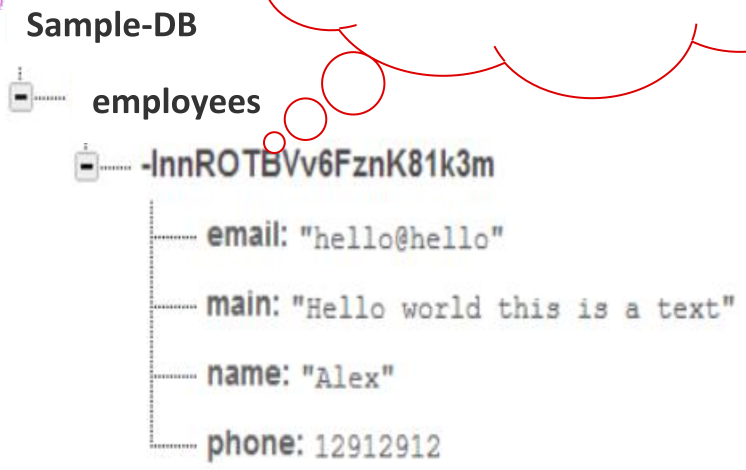


```
export class Employee {
  id: string;
  name: string;
  email: string;
```

Auto- Generated ID & is returned by default after insertion to calling function

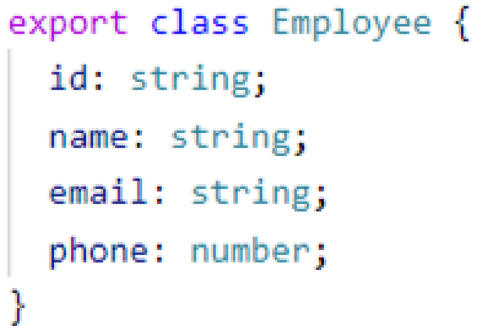


```
// HttpClient API post() method => Create employee
CreateEmployee(employee: any): Observable<Employee> {
    return this.http.post<Employee>(this.baseURL +
        JSON.stringify(employee),
        this.httpOptions);
}
```

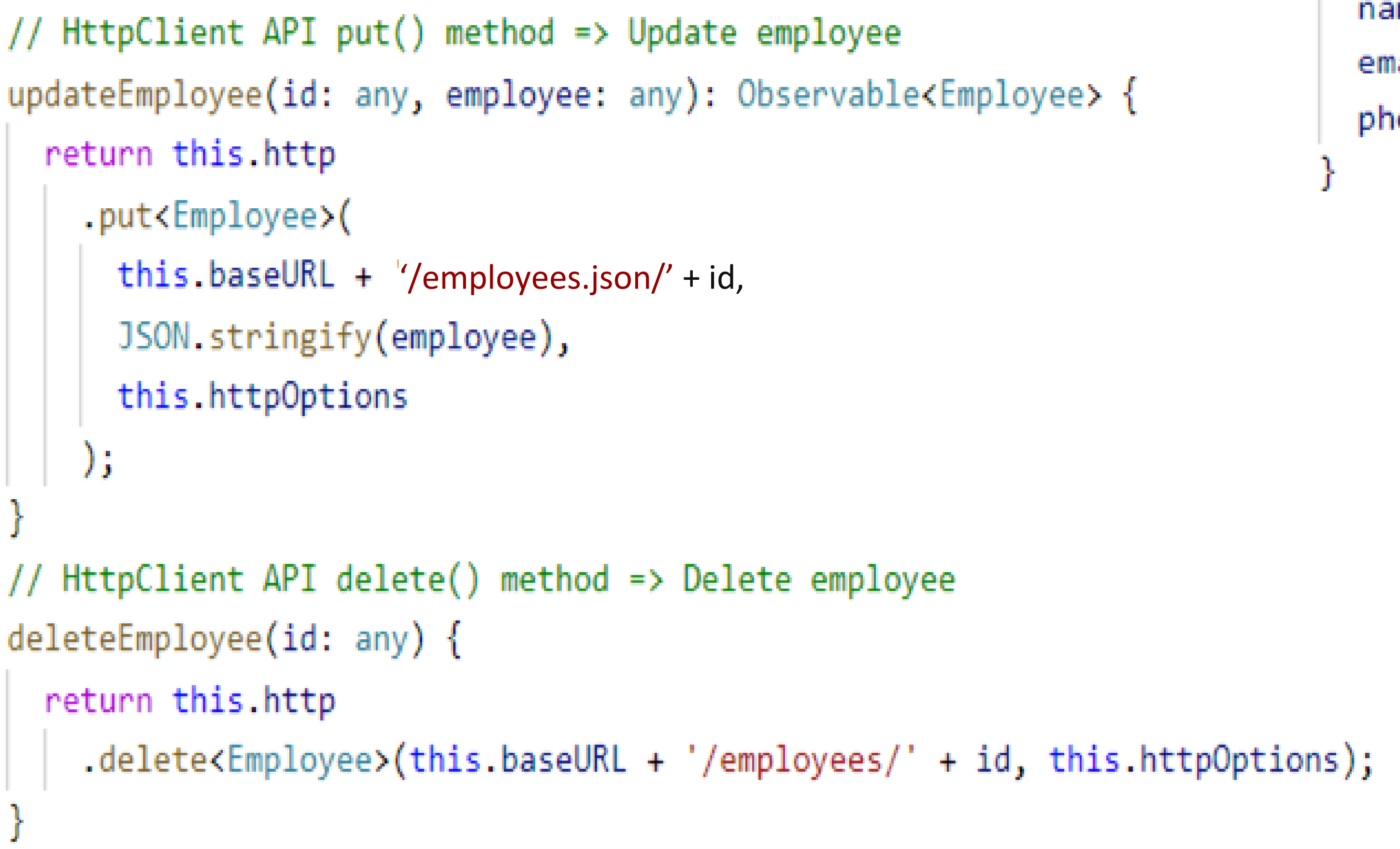


```
Sample-DB
 ⊟ employees
    ⊟ -InnROTBVv6FznK81k3m
         email: "hello@hello"
         main: "Hello world this is a text"
         name: "Alex"
         phone: 12912912
```

HTTP Requests URLs



```
export class Employee {
  id: string;
  name: string;
  email: string;
  phone: number;
}
```



```
// HttpClient API put() method => Update employee
updateEmployee(id: any, employee: any): Observable<Employee> {
  return this.http
    .put<Employee>(
      this.baseURL + '/employees.json/' + id,
      JSON.stringify(employee),
      this.httpOptions
    );
}
// HttpClient API delete() method => Delete employee
deleteEmployee(id: any) {
  return this.http
    .delete<Employee>(this.baseURL + '/employees/' + id, this.httpOptions);
}
```

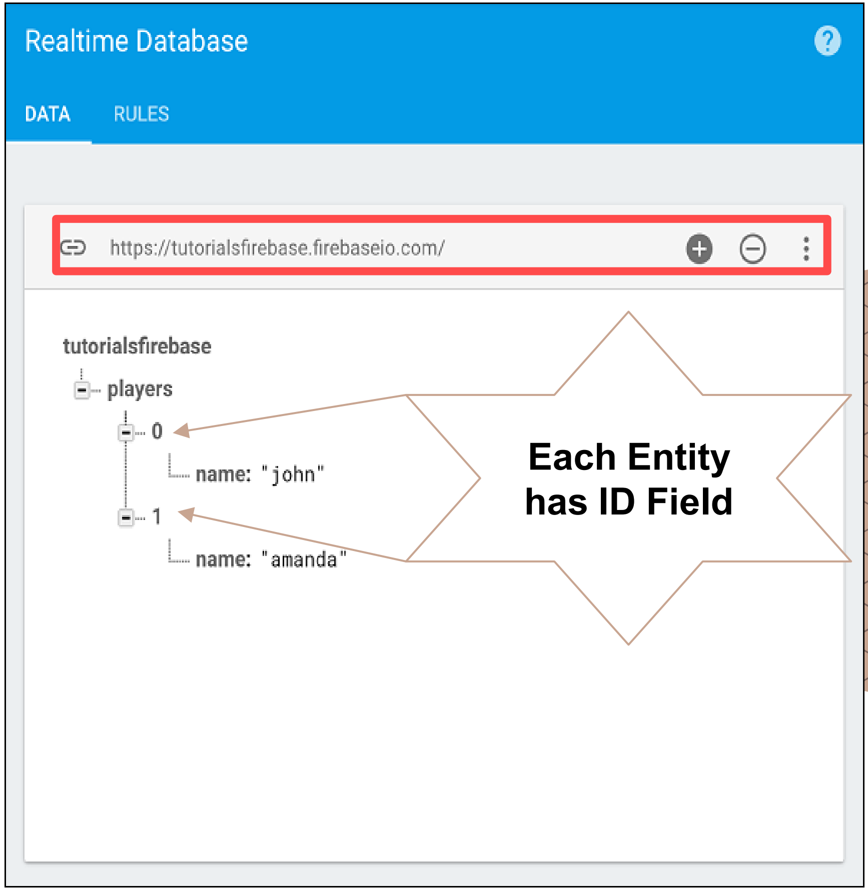
Firebase

- Real-time Database – Firebase supports **JSON data** and all users connected to it receive live updates after every change.

- Authentication – We can use anonymous, password or different social authentications.

CMD installation:

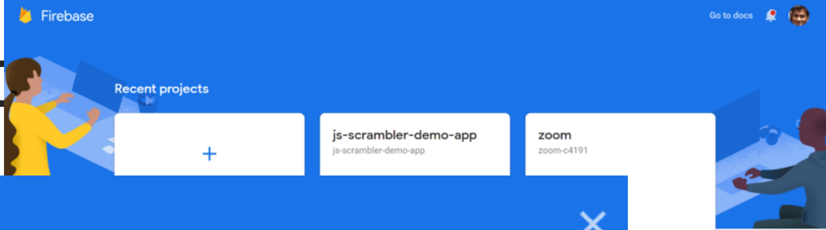
```
npm i firebase@5.10.1
npm i angularfire2@5.1.2
```

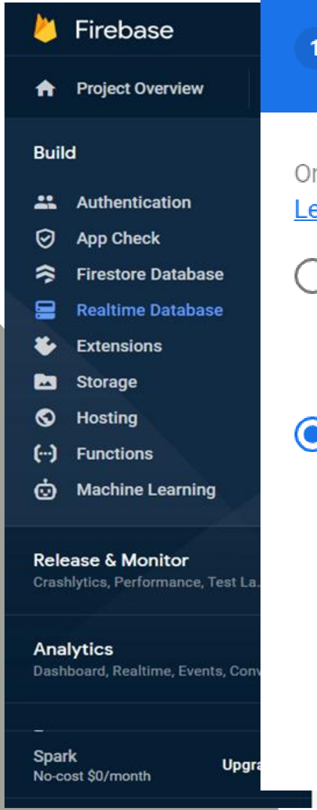


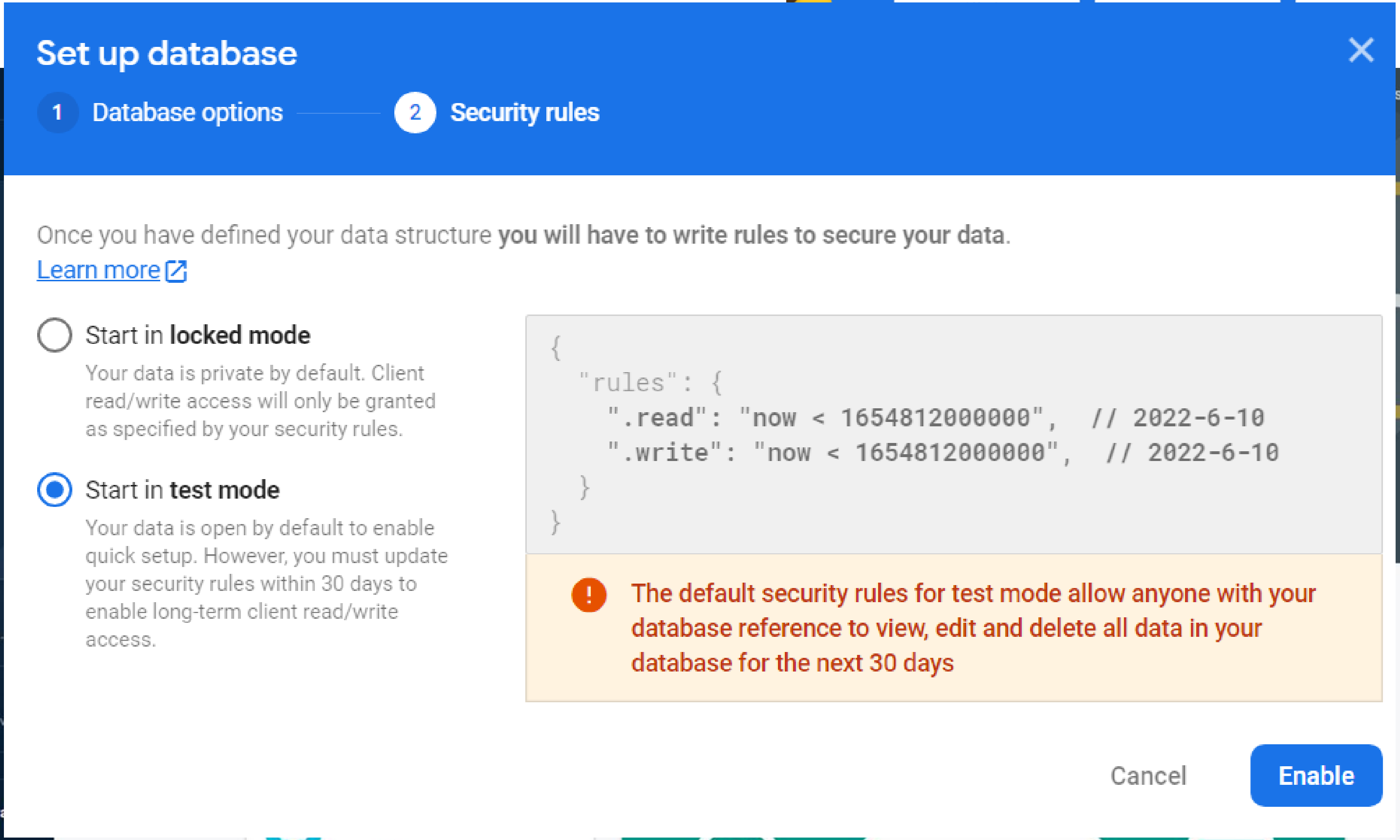
```
tutorialsfirebase
└─ players
   ├─ 0
   │  └─ name: "john"
   └─ 1
      └─ name: "amanda"
```

Firebase

Firebase Realtime Creatio







Once you have defined your data structure **you will have to write rules to secure your data.**

Start in **locked mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.

[✓ Start in **test mode**]

Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
{
  "rules": {
    ".read": "now < 1654812000000",  // 2022-6-10
    ".write": "now < 1654812000000",  // 2022-6-10
  }
}
```

The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Hands-On

Use the angular HTTPClient to make a get request to the Fake Store Api to fetch all products from this url:

https://fakestoreapi.com/products

Then display all the products in a list

Thank you

Any question?