TokApp Backend

Introduction:

TokApp backend consists of a bunch of firebase services and a call API to create a room or delete one, the services and features of our backend are as following:

Firebase Services:

- Listen to any new documents added to the “notifications” collection on Firestore database, when a document is added the listener will send the notification and then deletes the document.
- Listen to any new users or rooms added to our Firestore database and then adds them to Algolia search objects, this allows to search for users and rooms on the frontend, we’re using Algolia search service.
- Listen to changes to the realtime database (which is another database from firebase), it gives the ability to check whether a user is online or not, after we get online status of a user from it, we update firestore database (realtime database is only used for online status of users).

Call API:

There are two endpoints, one allows us to create a room and the other one is for deleting a room.

Firebase services configuration and requirements:

- Firebase library must be installed “npm install firebase”.
- The service account key is required and it can be downloaded as a json file from your firebase console, then one can import it and initialize your firebase app.
- Warning: this file must be private and you shouldn’t share it with anyone, if for some reason the file goes public or gets leaked you can delete it from firebase and create another one.
- We need the Algolia search API key and project key in order to initialize the service, these keys can simply be copied and pasted into a string in our code.
- Finally, the database URL is also required and it can be taken from firebase console.

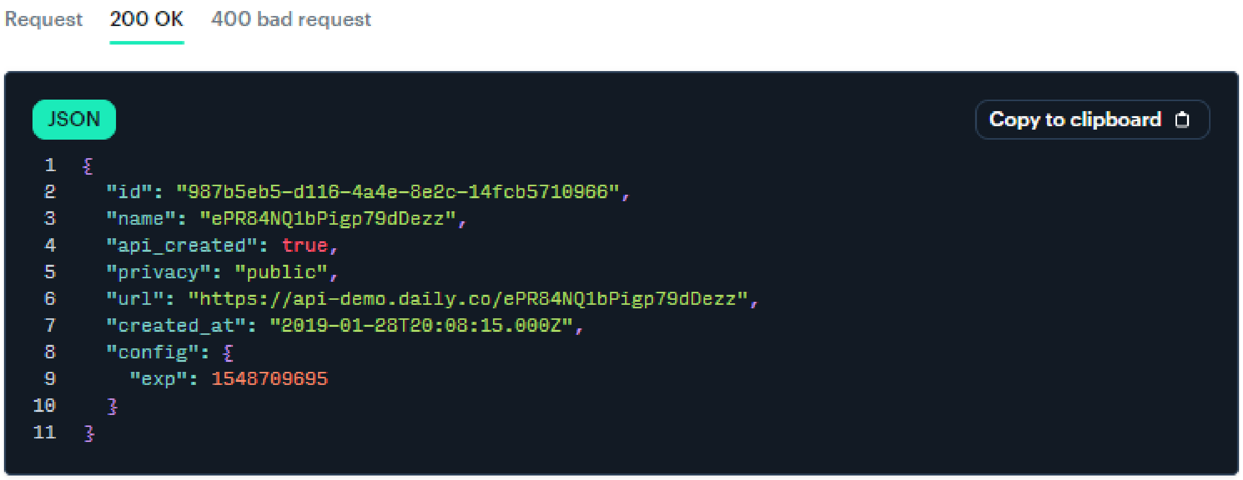
Call API configuration and requirements:

- You need the daily API key, you can get it from the console.
- Our server will need cors, they’re already set up in our code.
- We need ExpressJS to implement two endpoints one for creating a room and the other one for deleting a room.
- Our server will perform its functionalities by communicating with the daily API providing the API key.

Server configuration and requirement:

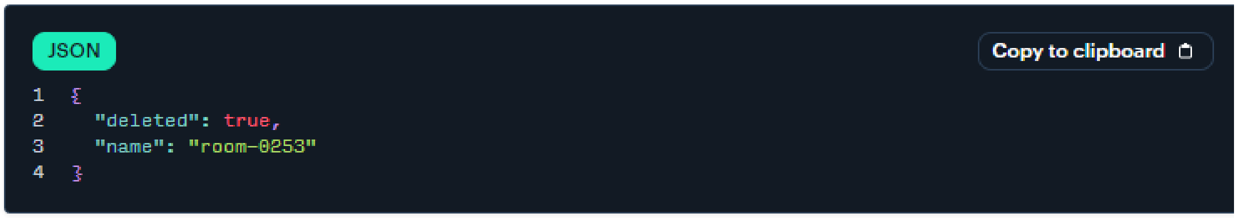
- First our environment that will run our backend needs to have NodeJS and npm installed, you can use prebuilt environment for NodeJS like Heroku, otherwise you have set it up by yourself, if you’re using google cloud as an example.

- After you've uploaded your code to your environment and prepared everything run "npm install" in order to install all the dependencies of our backend.
- After that you can run the firebase services by running the command "node index", this will run our server for firebase functionalities, it doesn't require any more configurations from you like setting up websockets or a api enpoints, it's just a worker that keeps running and listens for the changes in our firebase databases, it doesn't return anything and firebase handles all the listeners, just remember that it's a worker, it doesn't receive or returns data directly with an API for example, it receives its data from firebase listeners, firebase sdk that we installed will handle all the https requests like websockets and web hooks under the hood.
- Our second server is the Call API and this server can be run with the command "node callServer"
- First endpoint of call API is "URL/create-room", it receives POST requests and creates a room then returns the data of the new created room in the JSON format, check out the object returned by this endpoint, more info on POST /rooms (daily.co):



```json
{
  "id": "987b5eb5-d116-4a4e-8e2c-14fcb5710966",
  "name": "ePR84NQ1bPigp79dDezz",
  "api_created": true,
  "privacy": "public",
  "url": "https://api-demo.daily.co/ePR84NQ1bPigp79dDezz",
  "created_at": "2019-01-28T20:08:15.000Z",
  "config": {
    "exp": 1548709695
  }
}
```

- Second endpoint of call API is "URL/delete-room/:roomName", it receives DELETE requests where "roomName" is the name of the room to be deleted, it deletes the room then returns an object in the JSON format that has info about the deleted room and the deletion status, check out the object returned by this endpoint, more info on DELETE /rooms/:name (daily.co)



```json
{
  "deleted": true,
  "name": "room-0253"
}
```

- Finally, our backend runs two servers, one is a worker for firebase services, and the second one is an API for the call functionality that communicates with the daily API.

Both users lost connection situation??

This situation is better handled with service workers on the frontend since the backend won’t be able to communicate with the client side since there are internet problems, so we will cover this in another documentation with diagrams.