Ex 5 : Http Request

Starting date: @2020년 10월 26일

Deadline: @2020년 11월 6일

Instructions

For this exercise you will create a real-time database on Firebase then add data, remove data, modify data in it though the app. When you create the realtime database, start in testMode so anybody have the right to write or read the database. (Of course for a real project usually you have already rules so only the people who are allowed can read or write in the database). You will learn in the next exercise how to do it.

Before adding data, think how you gonna link the data in the database.

Add the request in the **services** folder. For that create some classes that you can call in other files.

For each request you will handle the error by showing it in the console but also add show an error message with a SnackBar.

Objectives

- How to use Firebase
- Add Loader
- Add refresh (pull / scroll bottom when you are already at the top) ⇒ SmartRefresher
- How to create a real time database
- Add / remove / modify data in the database
- Learn promise, future, await, async

- Use SnackBar

Documentation

You don't have to read all the documentation (because it gonna takes a long time) but you can refers to it when you don't know how to do something or when you have issues.

SnackBar

- https://flutter.dev/docs/cookbook/design/snackbars
- https://www.youtube.com/watch?v=zpO6n_oZWw0&ab_channel=Flutter

Promise / Async (you should read that if you don't know about asynchronous functions and promise - if you don't understand after reading it, you can ask me)

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Promises
- https://medium.com/flutter-community/a-guide-to-using-futures-in-flutter-for-beginners-ebeddfbfb967

Real time database

- https://firebase.google.com/docs/database
- https://firebase.google.com/products/realtime-database

Flutter Firebase :

- https://firebase.flutter.dev/docs/overview/
- https://pub.dev/packages/firebase_core (package) OR https://pub.dev/packages/http (package - this one is the one that we use on the other projects since we don't use firebase for the database)
- https://www.youtube.com/watch?v=DqJ_KjFzL9I&ab_channel=GoogleDevelopers

HTTP Request

For that you have to create Future (Promise). Do asynchronous function that gonna return the result of your request after getting it and checking if that correct. If it doesn't get any data or there is an error as a response of the request, this error should be handle.

For example :

```
class AddressService extends GlobalService {

  // get address
  Future<Map<String, dynamic>> getAddress(String address, {int page}) async {
    try {
      Map<String, dynamic> responseData;
      Map<String, dynamic> receiveData = {
        'list': [],
        'error': ''
      };

      Map<String, dynamic> requestData = {
        'keyword': address,
        'confmKey': 'devU01TX0FVVEgyMDIwMDIyNjE2MjIyMDEwOTQ5ODQ=',
        'resultType': 'json'
      };

      if(page != null) {
        requestData['currentPage'] = page.toString();
      }

      http.Response response = await http.post(
        'http://www.juso.go.kr/addrlink/addrLinkApiJsonp.do',
        body: requestData
      );

      if(response.statusCode == 200) {
        // remove the () at the begining and at the end to have a proper json
        String responseString = response.body.replaceFirst('(', '');
        responseString = responseString.substring(0, responseString.length - 1);

        responseData = json.decode(responseString);

        receiveData['list'] = responseData['results']['juso'];
        receiveData['error'] = responseData['results']['common']['errorMessage'];
        receiveData['totalPage'] = responseData['results']['common']['countPerPage'];
        return receiveData;
      } else {
        print(response.body);
```

```
        return null;
      }

    } catch (error) {
      print('error');
      print(error);
      return null;
    }
  }

}
```