Flutter Networking

GV: Nguyễn Thủy An

* http = HyperText Transfer Protocol -> (world wide web) www : truyền dữ liệu dưới dạng văn bản, hình ảnh, video, âm thanh...từ web server <-> trình duyệt web
    * TCP: Transmission Control Protocol/ IP: Internet Protocol
* https = HyperText Transfer Protocol Secure
    * SSL(secure sockets layer)->
    * TLS(transport layer security)
    * PKI

* synchronous: xử lí đồng bộ, theo thứ tự
  * không có lỗi
  * bộ nhớ tràn, trạng thái chờ, block UI,
* asynchronous: xử lí bất đồng bộ, không theo thứ tự
  * tận dụng tài nguyên
  * dễ phát sinh vấn đề, quản lí
  * database, api, mem, disk

```
Future<int> getFive() {
  return Future.delayed(Duration(seconds: 4), () => 5);
}

void main() {
    var x = getFive();
    print(x);
    print("this is test");
}
```

```
void main() async {
  var x = await getFive();
  print(x);
  print("this is test");
}

Future<int> getFive() {
   return Future.delayed(Duration(seconds: 4), () => 5);
}
```

kq = ?

```
Future<T> functionName() async {
  await bieu_thuc;
  return T;
}
```

Future:

- Uncompleted -> chờ cho các hoạt động không đồng bộ của hàm kết thúc và trả về error
- Completed: hoàn thành với giá trị

await: trong thân hàm async , chờ lấy kết quả hàm bất đồng bộ

thực thi code phải tạm dừng khi async dcd thực thi ,
xong-> Future object

- api = application programming interface
  - phương thức kết nối đến thư viện và ứng dụng
- rest api = representational state transfer
  - chuyển đổi cấu trúc dữ liệu (ứng dụng kết nối)
  - dùng http để giao tiếp giữa các máy (client, server), gửi iu cầu đến url xử lí
    - get , post, delete, put
    - status code: 200 (ok), 400 (bad request), 404(not found), 403(forbidden)

* get: return resources
* post: create resource
* put: update information for resource
* delete: delete resource
    * postman

* setup json server
  * run json-server
* Fetch data from the server
  * http package
  * convert response into custom Dart object
  * fetch data and display

```
body: FutureBuilder<List<Photo>>(
  future: fetchPhotos(http.Client()),
  builder: (context, snapshot) {
    if(snapshot.hasError) {
      return const Center(
        child: Text('An error...'),
      );
    } else if(snapshot.hasData) {
      return PhotoList(photos: snapshot.data!);
    } else {
      return const Center(
        child: CircularProgressIndicator(),
      );
    }
  },
),
```

```dart
List<Photo> parsePhotos(String responseBody) {
  final parsed = jsonDecode(responseBody).cast<Map<String, dynamic>>();
  return parsed.map<Photo>((json) => Photo.fromJson(json)).toList();
}
```

```dart
Future<List<Photo>> fetchPhotos(http.Client client) async {
  final response = await client
      .get(Uri.parse('https://jsonplaceholder.typicode.com/photos'));
  return compute(parsePhotos, response.body);
}
```

```dart
factory Photo.fromJson(Map<String, dynamic> json) {
  return Photo(
    albumId: json['albumId'] as int,
    id: json['id'] as int,
    title: json['title'] as String,
    url: json['url'] as String,
    thumbnailUrl: json['thumbnailUrl'] as String,
  );
}
```