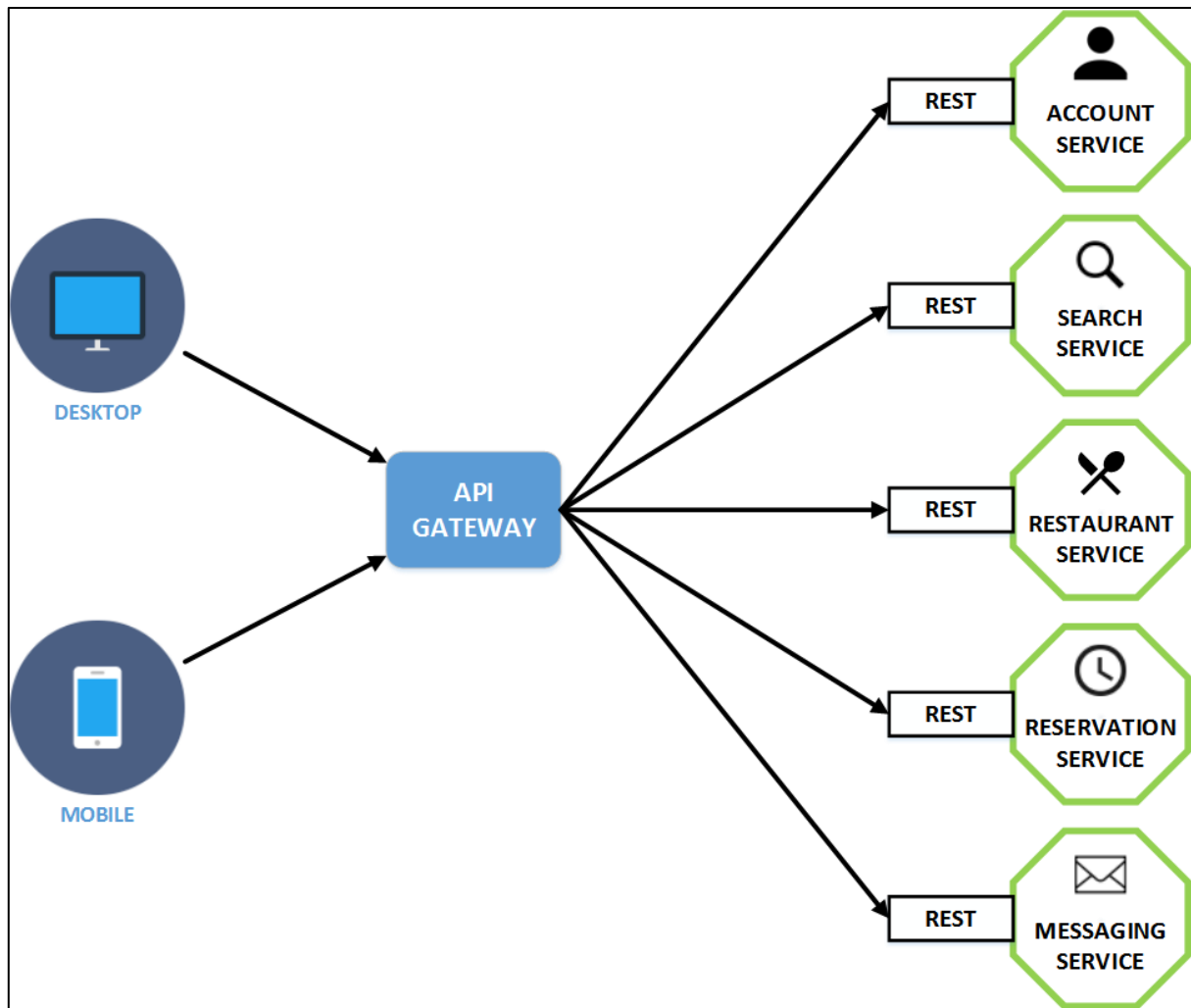


# Flutter and Flask RESTful API with Database



## Web API

คือ การให้บริการข้อมูลผ่าน Web โดยรับส่งข้อมูลผ่าน HTTP ซึ่งจะอยู่ในรูปแบบของ XML และ JSON โดยข้อมูล (ภาพ,เสียง,ข้อมูลทางธุรกิจ) จะเรียกว่า **Resource**



Client



Server



Resource

## Architectural properties

ข้อกำหนดของ REST architectural style ทั้ง 7 ประกอบไปด้วย

- Performance: ประสิทธิภาพในการโต้ตอบ
- Scalability: มีความสามารถในการปรับขยายได้ทำให้สามารถรองรับส่วนประกอบจำนวนมาก
- Simplicity: ความเรียบง่ายของอินเทอร์เฟซที่เหมือนกัน
- Modifiability: ความสามารถในการปรับเปลี่ยนส่วนประกอบเพื่อตอบสนองความต้องการที่เปลี่ยนแปลง (แม้ในขณะที่แอปพลิเคชันทำงานอยู่)
- Visibility: การมองเห็นการสื่อสารระหว่าง Component ผ่าน Service agents
- Portability: ง่ายต่อการโยกย้ายในส่วนของ Program และ Data
- Reliability: ความน่าเชื่อถือในการต้านทานความล้มเหลวในระบบ

**RESTful – RESTful Web Service(RWS)** คือ Web Service ที่ใช้สถาปัตยกรรม Rest ซึ่งเจ้าตัว RWS อนุญาต ให้ระบบ Request และเข้าถึง Resource บนเว็บโดยใช้ชุดคำสั่งที่กำหนดเอาไว้ล่วงหน้า โดยที่การโต้ตอบของระบบที่ใช้ REST จะอยู่บนพื้นฐานของ Hypertext Transfer Protocol (HTTP). Request จะส่งคำขอไปยัง URI ที่กำหนด และลัวงเอา response กลับมาเป็น Payload ในแบบ HTML, XML, JSON หรือ format อื่น ๆ โดย RESTful จะประกอบไปด้วย

**Client – ผู้ที่เข้ามาเป็น Request Resource**

**Server – ผู้ที่ให้บริการ Resource**

## Architectural constraints of RESTful API

6 ข้อกำหนดของ RESTful API ซึ่งถือเป็นสิ่งสำคัญในการสร้าง RESTful API ตามมาตรฐานซึ่งทำให้ง่ายต่อการพัฒนา และทำให้เป็นที่ยอมรับ (หากไม่ทำตามให้ครบทั้ง 6 ข้อจะไม่ถือว่าเป็น RESTful API ยกเว้น optional)

- Client-server architecture: Client ไม่จำเป็นต้องรู้อะไรเกี่ยวกับ Business logic ภายใน ไม่มีหน้าที่เกี่ยวกับการจัดเก็บข้อมูล ส่วน Server มีหน้าที่เก็บ Resource และไม่จำเป็นต้องรู้อะไรเกี่ยวกับ UI Frontend หรือสถานะของผู้เรียก
- Statelessness: ส่ง Request รับ Response จาก Server แล้วเลิก
- Cacheability: สามารถ cache response ได้ การ Response จะต้องสามารถกำหนดได้ว่าจะ Cache หรือไม่ เพื่อป้องกันไม่ให้ User หรือ Client ได้รับข้อมูลเก่า หรือท่านสามารถดูเพิ่มเติมเรื่อง idempotent ได้ที่นี่
- Layered system: ปกติ Client ไม่รู้ว่าที่ทำการเชื่อมต่อนั้น ได้เชื่อมต่อโดยตรงกับ Server ปลายทาง หรือไปยังตัวกลางอื่น ๆ ระหว่างทาง, Server ตัวกลางควรสามารถปรับปรุงความสามารถในการขยายระบบได้ โดยการใช้งานการทำ Load balance
- Code on demand (optional): Server สามารถขยายได้ชั่วคราว หรือปรับแต่งการทำงานของไคลเอนต์ได้ ตัวอย่างเช่น ทำ client-side scripts ใน JavaScript
- Uniform interface: ถือเป็นข้อสำคัญที่แยกระหว่าง REST API และ Non-REST API มันแสดงให้เห็นถึงวิธีการที่จะคุยกับ Server โดยไม่คำนึงถึงประเภทของอุปกรณ์ หรือประเภทของ application
- Uniform interface ได้แยกออกไปอีก 4 อย่าง ดังต่อไปนี้
  1. Resource-Based: เช่น API/users
  2. Manipulation of Resources Through Representations: เช่น User get user\_id หรือ Request list of users แล้วทำการ Delete หรือ Modify user
  3. Self-descriptive Messages: แต่ละ Message มีข้อมูลเพียงพอที่จะนำมาอธิบายวิธีการ Process message เพื่อให้ Server ทำการวิเคราะห์ได้ง่าย
  4. Hypermedia as the Engine of Application State (HATEOAS): จำเป็นต้องมี Links สำหรับทุก ๆ Response เพื่อให้ Client สามารถค้นหาได้ง่าย

## REST เบื้องต้น

REST ทำงานอยู่บนอยู่ใน HTTP Protocol ทำให้เวลาใช้งานจะต้องอยู่บนพื้นฐาน HTTP Method เช่น GET, POST, PUT, DELETE การที่จะใช้ Method ไหน เมื่อไร ก็ขึ้นอยู่กับว่าจะทำอะไรกับข้อมูล แต่ก็ต้องควรใช้คู่กับ Operation CRUD เช่น

เมื่อต้องการจะเรียกดูข้อมูลทั้งหมดก็ใช้ GET

เมื่อต้องการเพิ่มข้อมูลก็ใช้ POST

GET — R(etrieve) เรียกดูข้อมูล

POST — C(reate) เพิ่มข้อมูล

PUT — U(pdate) แก้ไขข้อมูล

DELETE — D(elete) ลบข้อมูล

## HTTP Method ที่สำคัญ

Get เป็นการร้องขอข้อมูลจาก resource

Post เป็นการสร้างข้อมูลใหม่ใน resource

Put เป็นการอัปเดตข้อมูลที่มีอยู่แล้ว หรือสร้างใหม่ resource

Delete เป็นการลบข้อมูลที่มีอยู่แล้วใน resource

## HTTP Response Status Code

### 2xx (success code)

- 200 OK – มาตรฐาน HTTP response success สำหรับ GET, PUT หรือ POST
- 201 Create – response สำหรับข้อมูลที่ถูกสร้างขึ้นใหม่ ใช้สำหรับ POST
- 204 No Content – response สำหรับ request ที่ดำเนินการ success แล้วไม่ return ข้อมูลกลับมา

### 3xx (Redirection)

- 304 Not Modified – บอกว่า client ได้รับการ response แล้วอยู่ใน cache และไม่จำเป็นต้องส่งข้อมูลเดิมอีกครั้ง

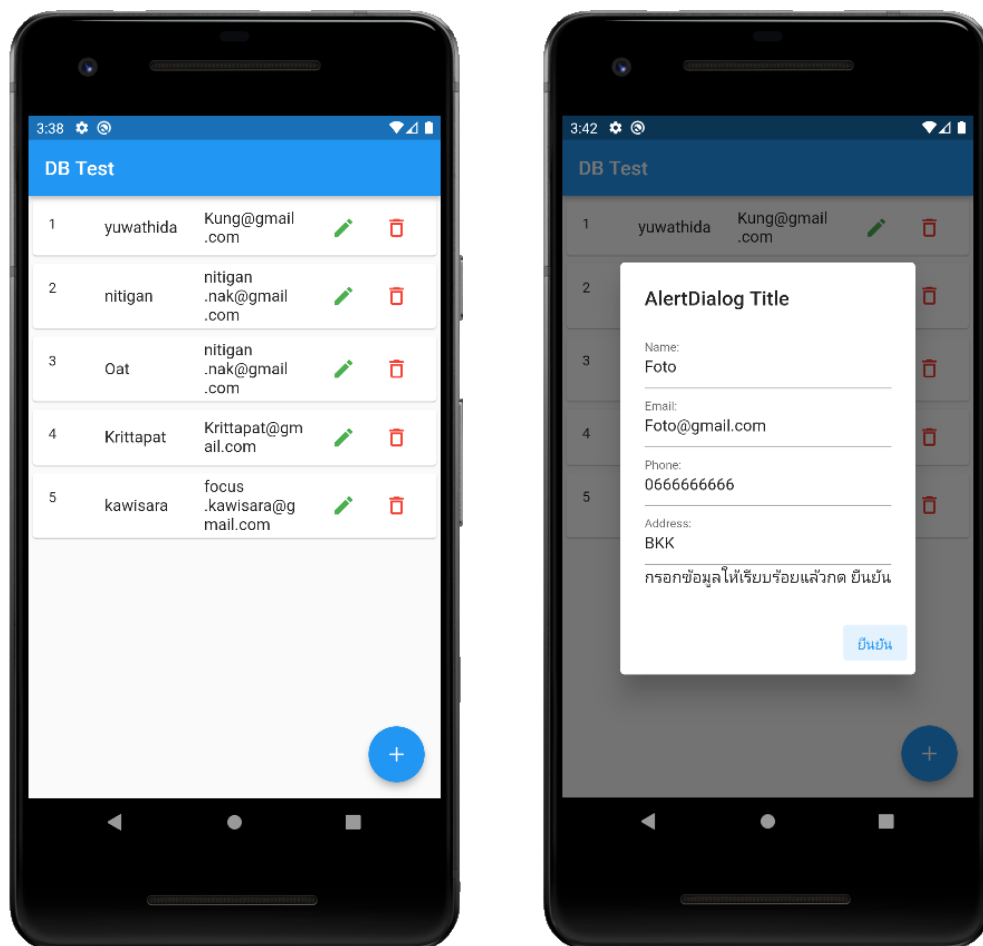
### 4xx (Client Error)

- 400 Bad Request – request ที่ส่งมาโดย client นั้นไม่ถูกดำเนินการ
- 401 Unauthorized – client ไม่ได้รับอนุญาตในการเข้าถึง resource และควรจะ request ใหม่ด้วย credential
- 403 Forbidden – บ่งบอกว่า request นั้นถูกต้องและ client ได้รับการอนุญาต แต่ client ไม่ได้รับอนุญาตให้เข้าถึง resource ด้วยเหตุผลบางประการ
- 404 Not Found – resource ที่ request มานั้นไม่ว่างใช้งานตอนนี้
- 405 Gone – resource ไม่มีอยู่แล้ว หรือถูกย้ายไปที่อื่น

### 5xx (Server Error)

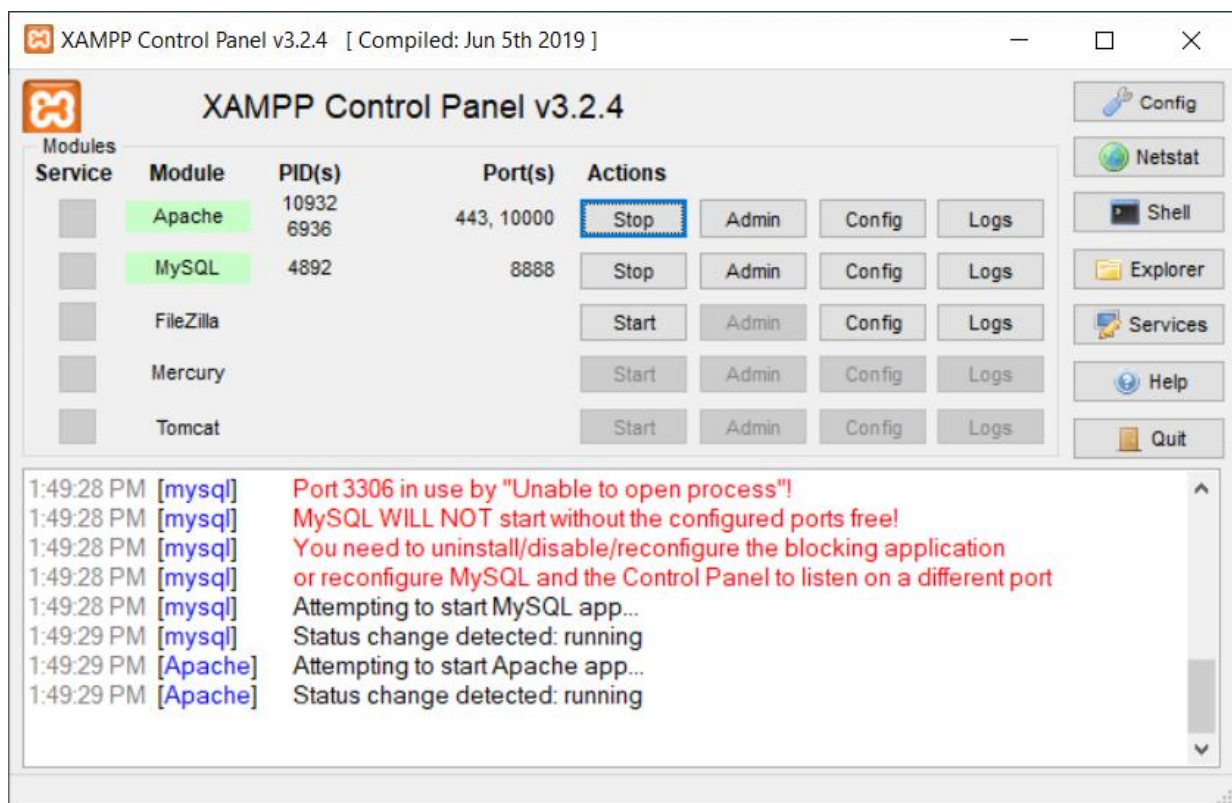
- 500 Internal Server Error – request ถูกต้อง แต่ server มีความสับสนและจะบริการด้วยเงื่อนไขที่คาดการณ์ไม่ได้
- 503 Service Unavailable – server ใช้งานไม่ได้ (โดยส่วนใหญ่ server อยู่ในช่วงบำรุงรักษา)

## สร้าง Application สำหรับการ Insert , Update , Delete ข้อมูลใน Database

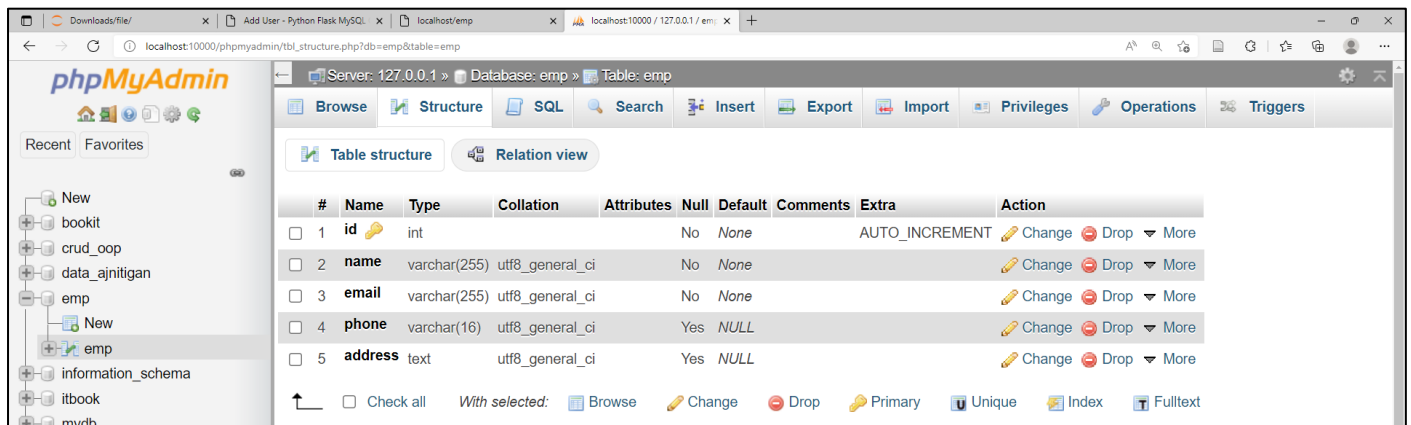


## เปิดการใช้งาน Service ของ Web Server และ MySQL

\*เปลี่ยน port ของ Apache Web Server เพื่อป้องกันหมายเลข port ชนกับ Web API



สร้างฐานข้อมูล ชื่อ emp



ใช้คำสั่ง SQL เพื่อสร้างตารางข้อมูล ชื่อ emp

```
CREATE TABLE emp (  
  id int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  name varchar(255) NOT NULL,  
  email varchar(255) NOT NULL,  
  phone varchar(16) DEFAULT NULL,  
  address text DEFAULT NULL  
)
```

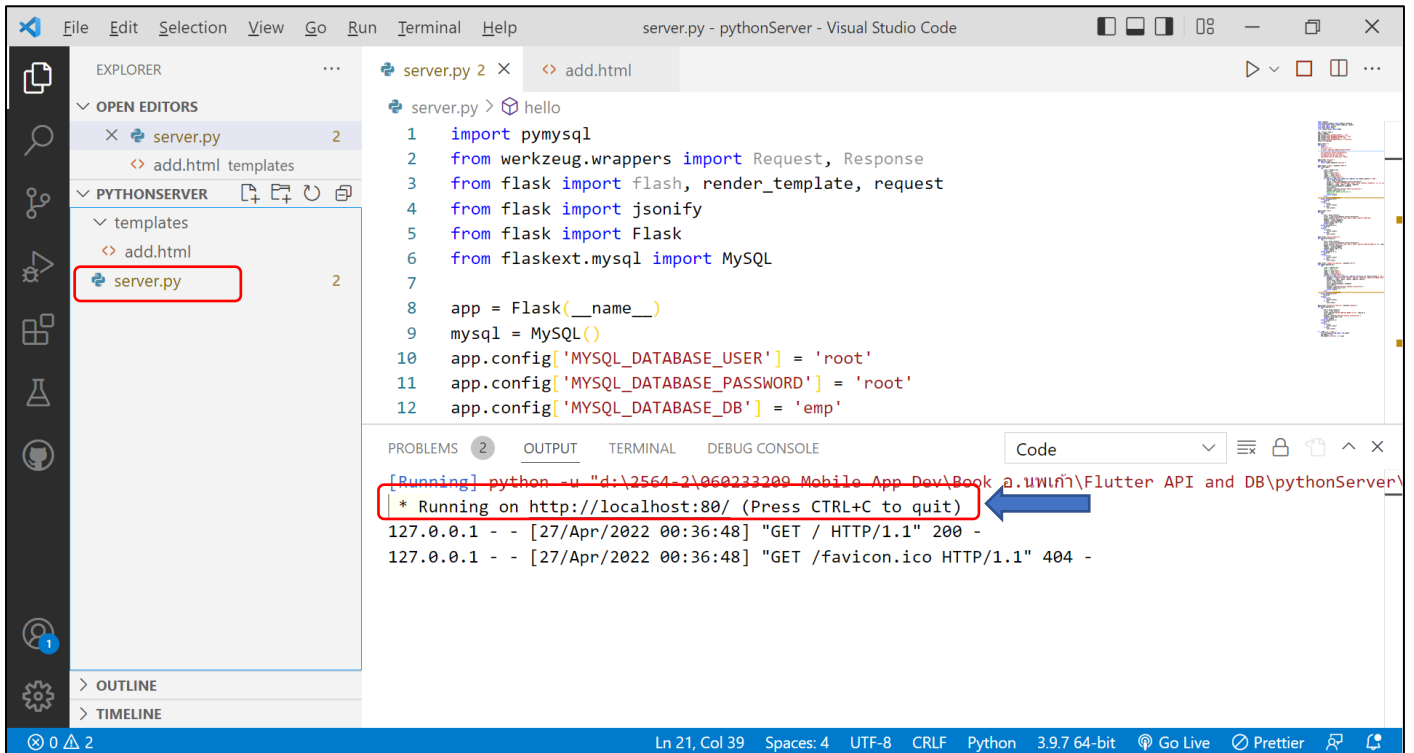
สร้าง Server สำหรับให้บริการ API โดยกำหนดให้มีโครงสร้างดังต่อไปนี้

|-templates

| — add.html

|-server.py

ติดตั้ง package  
pip install flask  
pip install flask-mysql

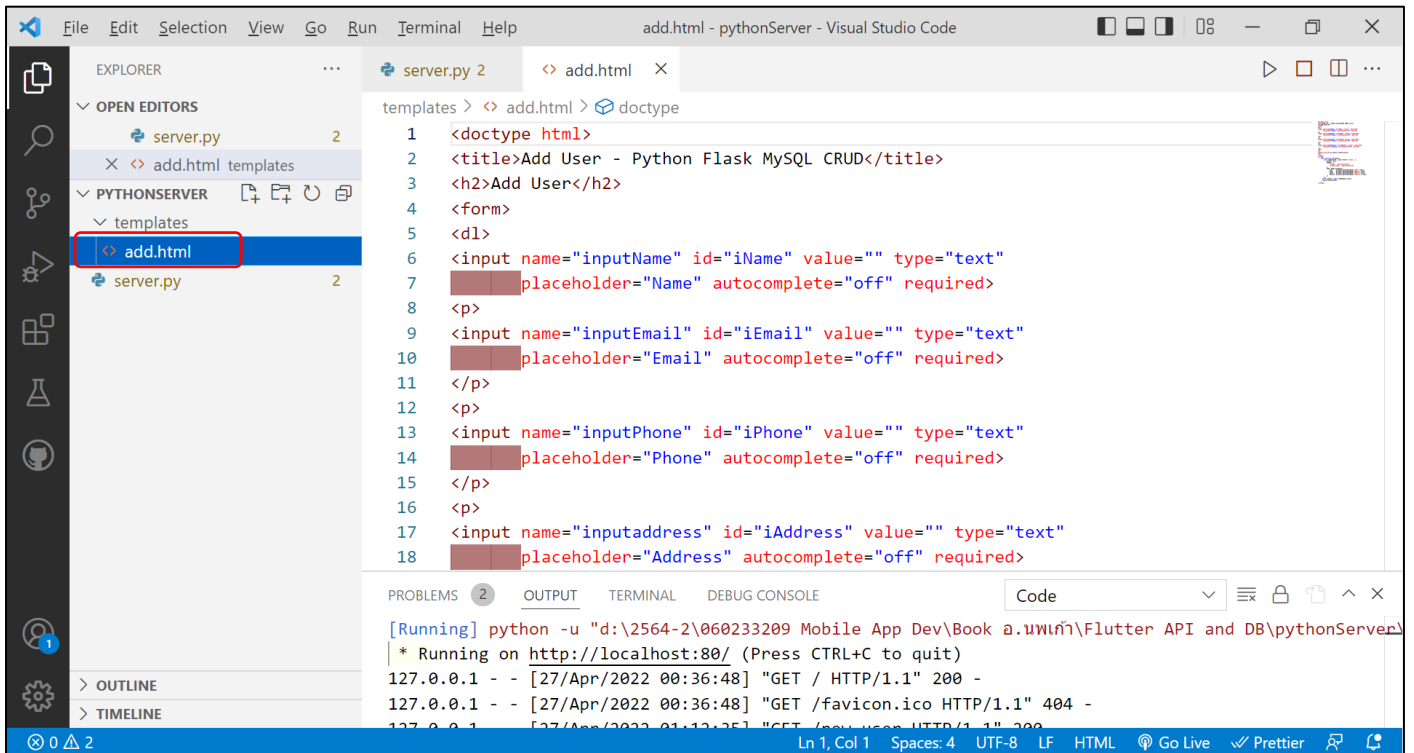


The screenshot shows the Visual Studio Code interface with the `server.py` file open in the editor. The file contains the following code:

```
1 import pymysql
2 from werkzeug.wrappers import Request, Response
3 from flask import flash, render_template, request
4 from flask import jsonify
5 from flask import Flask
6 from flaskext.mysql import MySQL
7
8 app = Flask(__name__)
9 mysql = MySQL()
10 app.config['MYSQL_DATABASE_USER'] = 'root'
11 app.config['MYSQL_DATABASE_PASSWORD'] = 'root'
12 app.config['MYSQL_DATABASE_DB'] = 'emp'
```

The terminal output shows the command `python -u "d:\2564-2\060233209 Mobile App Dev\Book a.นพเก้า\Flutter API and DB\pythonServer\server.py"` being executed, and the server running on `http://localhost:80/`. The output also shows the response for the `GET / HTTP/1.1` request, which is `200 -`.

GUI ของเว็บ สำหรับ กรอกข้อมูลและบันทึกลง Database

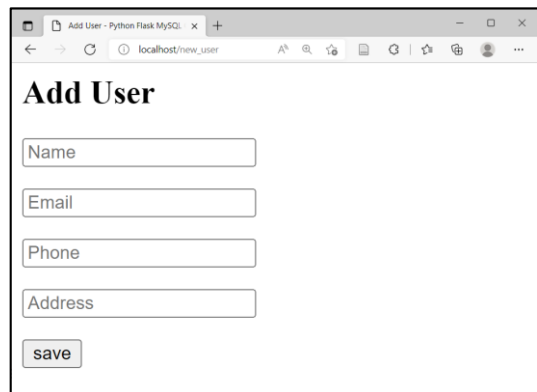
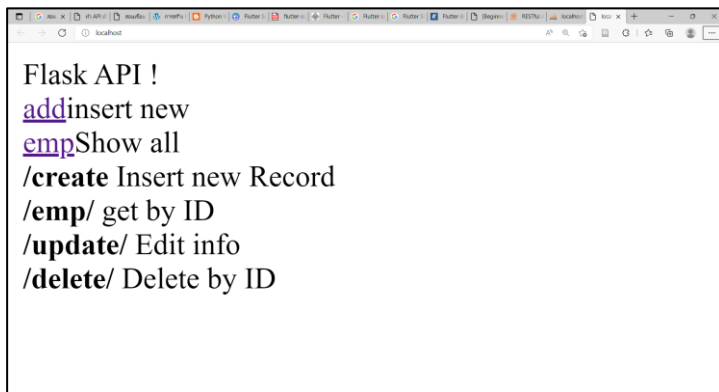


The screenshot shows the Visual Studio Code interface with the `add.html` file open in the editor. The file contains the following HTML code:

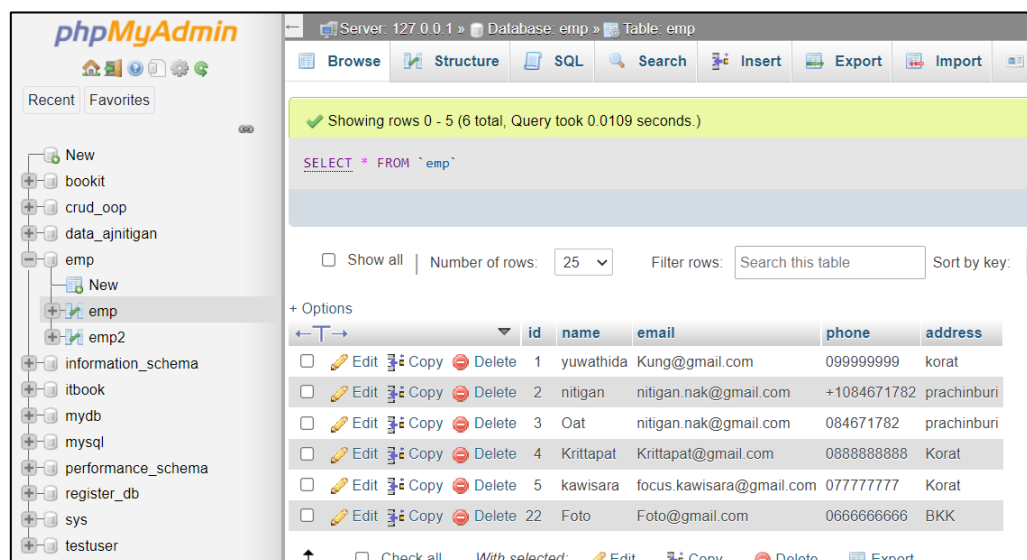
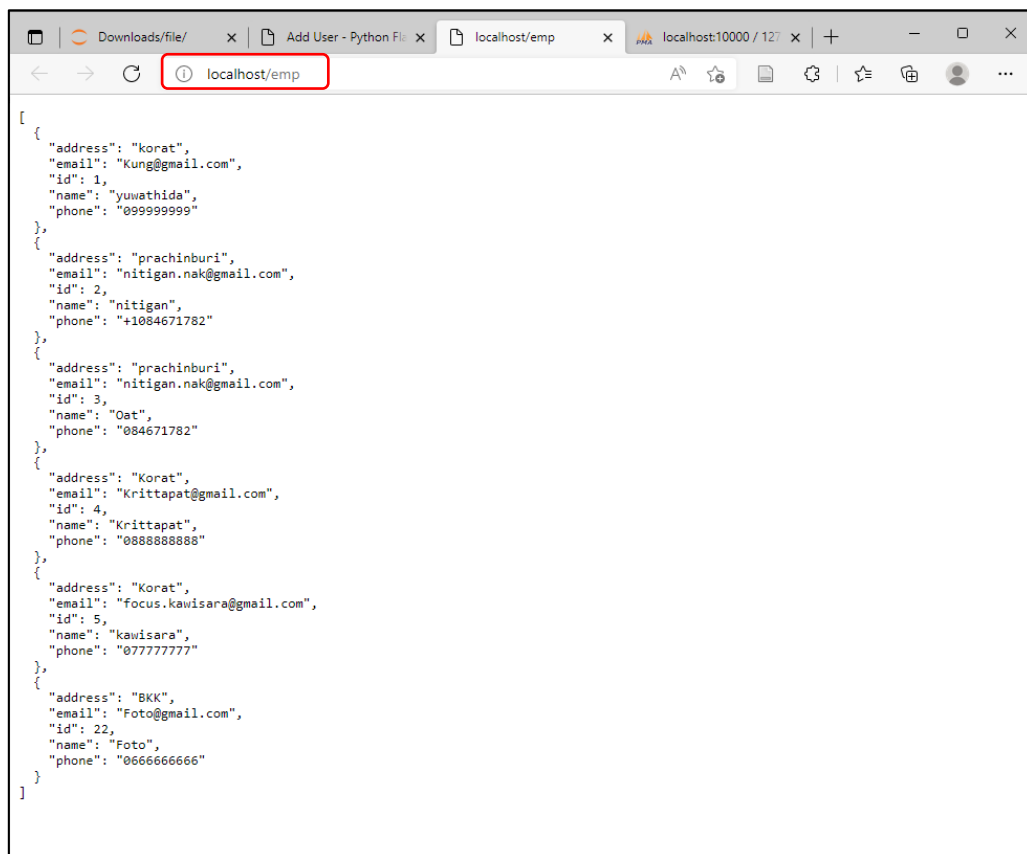
```
1 <doctype html>
2 <title>Add User - Python Flask MySQL CRUD</title>
3 <h2>Add User</h2>
4 <form>
5 <dl>
6 <input name="inputName" id="iName" value="" type="text"
7     placeholder="Name" autocomplete="off" required>
8 <p>
9 <input name="inputEmail" id="iEmail" value="" type="text"
10     placeholder="Email" autocomplete="off" required>
11 </p>
12 <p>
13 <input name="inputPhone" id="iPhone" value="" type="text"
14     placeholder="Phone" autocomplete="off" required>
15 </p>
16 <p>
17 <input name="inputaddress" id="iAddress" value="" type="text"
18     placeholder="Address" autocomplete="off" required>
```

The terminal output shows the command `python -u "d:\2564-2\060233209 Mobile App Dev\Book a.นพเก้า\Flutter API and DB\pythonServer\server.py"` being executed, and the server running on `http://localhost:80/`. The output also shows the response for the `GET / HTTP/1.1` request, which is `200 -`.

เมื่อ Run ไฟล์ server.py แล้ว ให้เรียกดูหน้าเว็บของ Server ได้ที่ localhost



เมื่อกรอกข้อมูลแล้ว หน้า Web API จะแสดงข้อมูล จาก Database ในรูปแบบ JSON



## สร้าง Application ด้วยโครงสร้างดังนี้

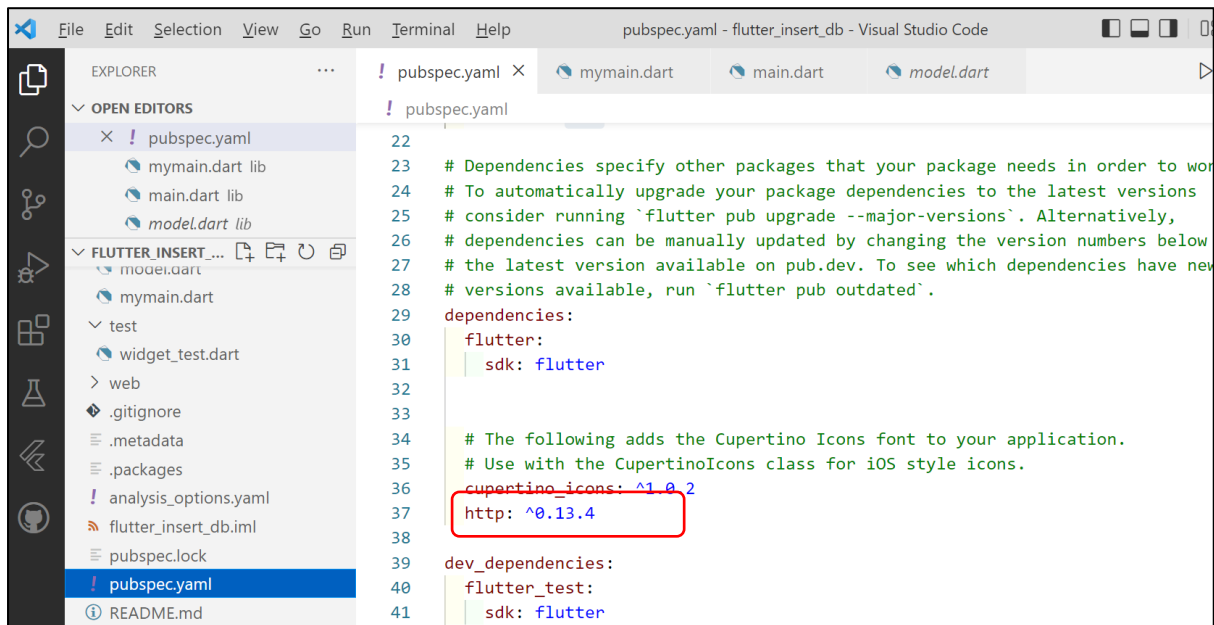
|-lib

| — main.dart

| — model.dart

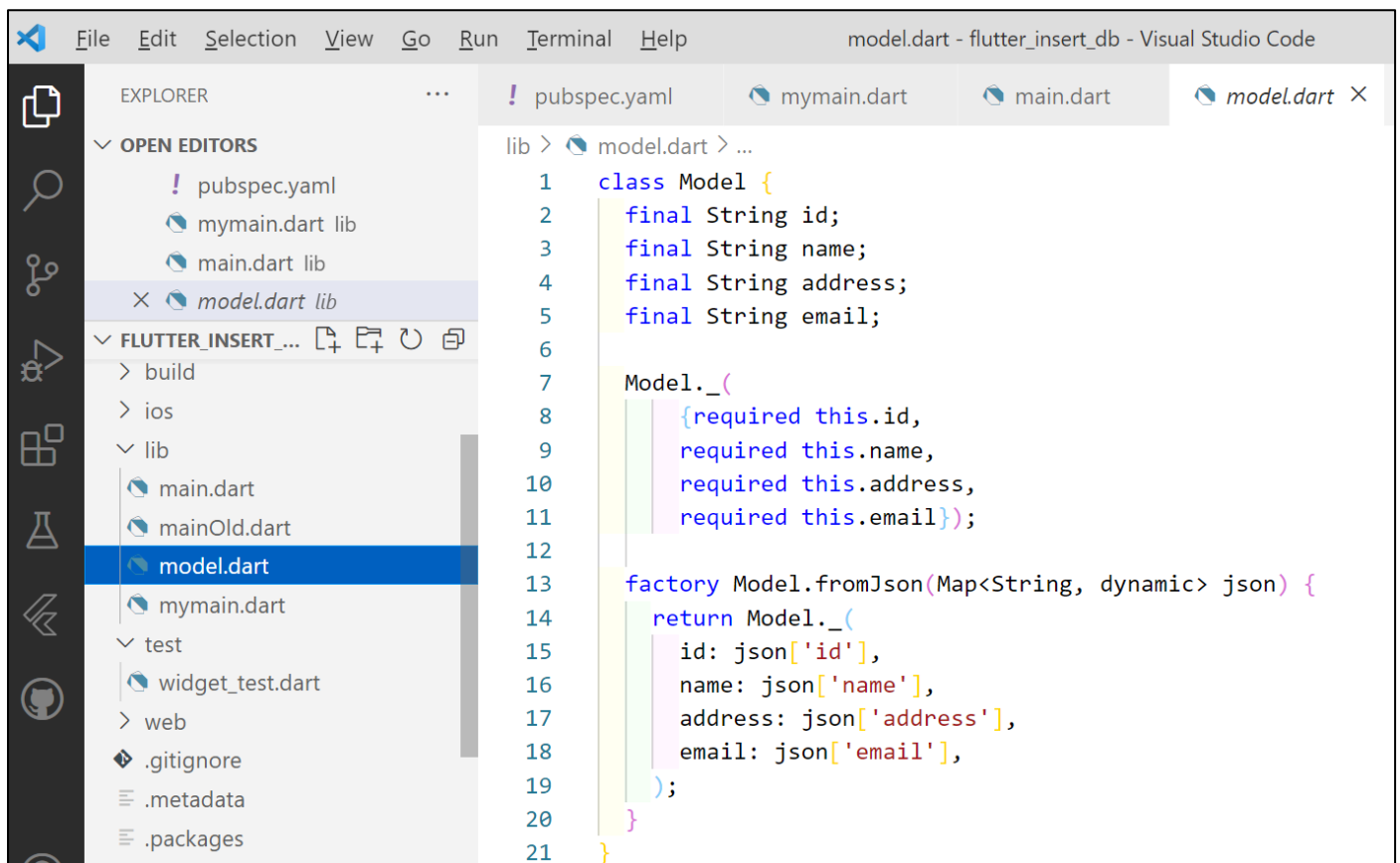
| — mymain.dart

\*ติดตั้ง Package http ที่ pubspec.yaml



```
22
23 # Dependencies specify other packages that your package needs in order to work
24 # To automatically upgrade your package dependencies to the latest versions
25 # consider running `flutter pub upgrade --major-versions`. Alternatively,
26 # dependencies can be manually updated by changing the version numbers below
27 # the latest version available on pub.dev. To see which dependencies have new
28 # versions available, run `flutter pub outdated`.
29 dependencies:
30   flutter:
31     sdk: flutter
32
33   # The following adds the Cupertino Icons font to your application.
34   # Use with the CupertinoIcons class for iOS style icons.
35   cupertino_icons: ^1.0.2
36   http: ^0.13.4
37
38 dev_dependencies:
39   flutter_test:
40     sdk: flutter
41
```

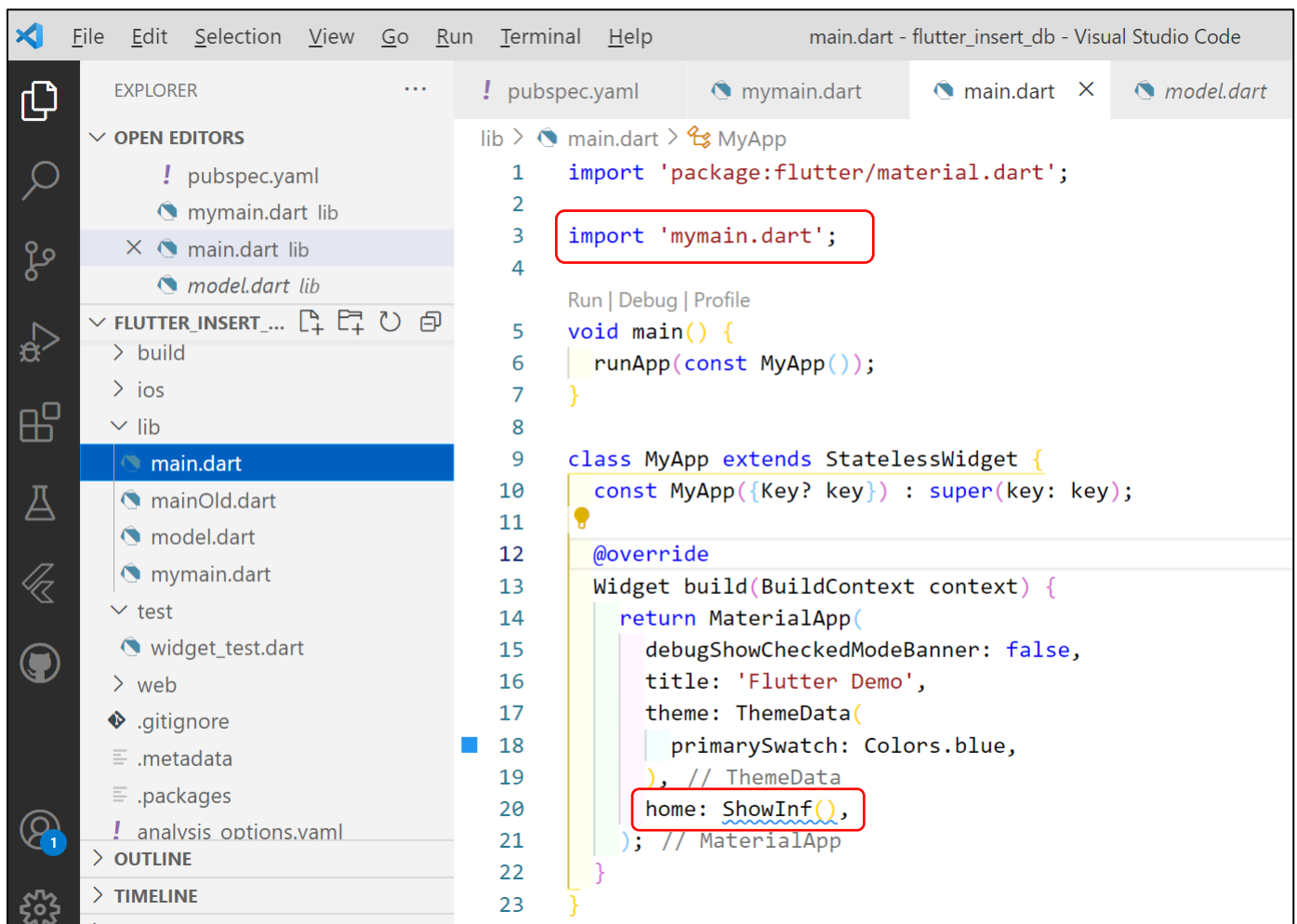
**model.dart** สำหรับกำหนดโครงสร้างข้อมูล



```
lib > model.dart > ...
1 class Model {
2   final String id;
3   final String name;
4   final String address;
5   final String email;
6
7   Model._(
8     {required this.id,
9     required this.name,
10    required this.address,
11    required this.email});
12
13   factory Model.fromJson(Map<String, dynamic> json) {
14     return Model._(
15       id: json['id'],
16       name: json['name'],
17       address: json['address'],
18       email: json['email'],
19     );
20   }
21 }
```

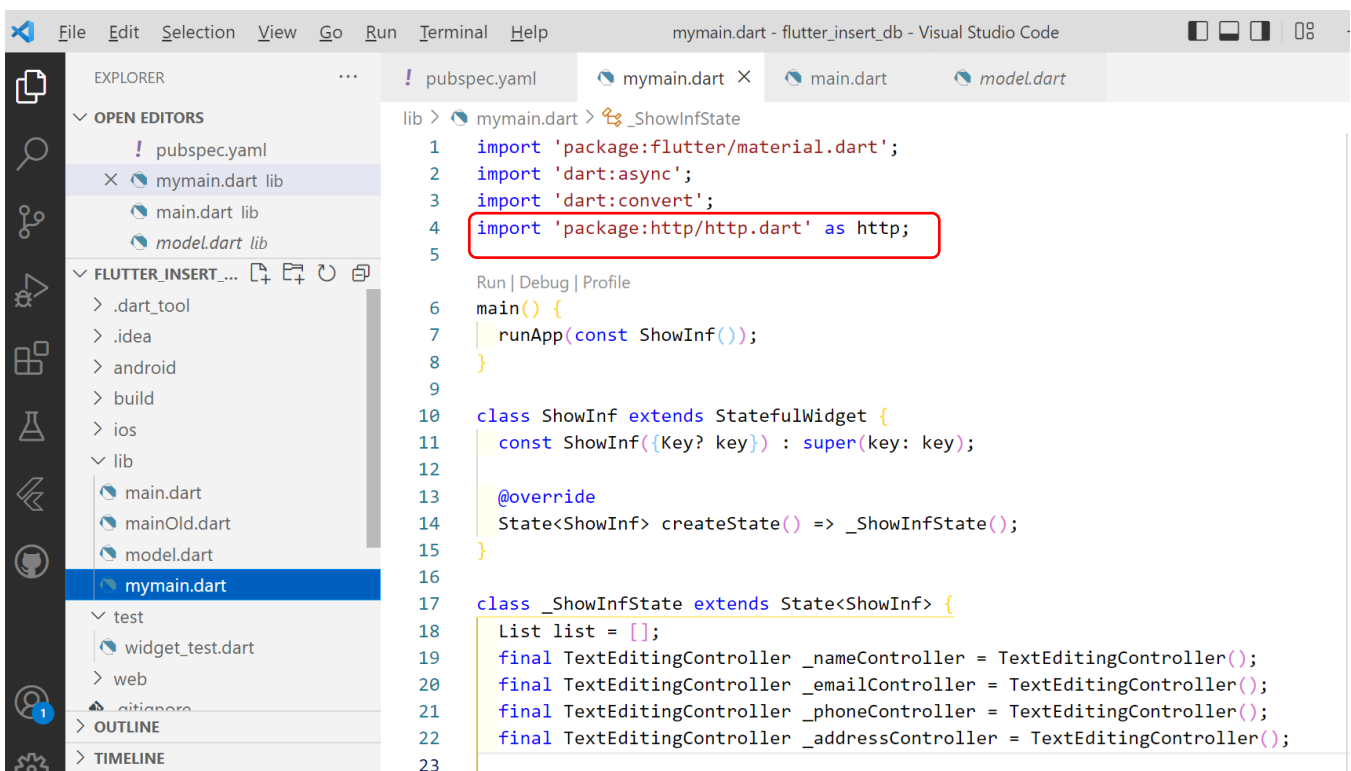


main.dart เพื่อเรียกใช้ GUI จากไฟล์ mymain.dart

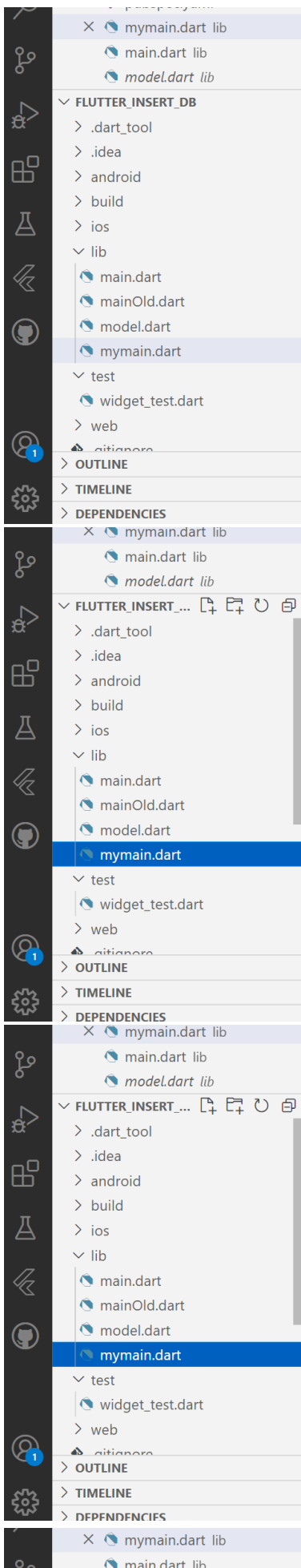


```
lib > main.dart > MyApp
1  import 'package:flutter/material.dart';
2
3  import 'mymain.dart';
4
5  void main() {
6    runApp(const MyApp());
7  }
8
9  class MyApp extends StatelessWidget {
10   const MyApp({Key? key}) : super(key: key);
11
12   @override
13   Widget build(BuildContext context) {
14     return MaterialApp(
15       debugShowCheckedModeBanner: false,
16       title: 'Flutter Demo',
17       theme: ThemeData(
18         primarySwatch: Colors.blue,
19       ), // ThemeData
20       home: ShowInf(),
21     ); // MaterialApp
22   }
23 }
```

mymain.dart



```
lib > mymain.dart > _ShowInfState
1  import 'package:flutter/material.dart';
2  import 'dart:async';
3  import 'dart:convert';
4  import 'package:http/http.dart' as http;
5
6  void main() {
7    runApp(const ShowInf());
8  }
9
10 class ShowInf extends StatefulWidget {
11   const ShowInf({Key? key}) : super(key: key);
12
13   @override
14   State<ShowInf> createState() => _ShowInfState();
15 }
16
17 class _ShowInfState extends State<ShowInf> {
18   List list = [];
19   final TextEditingController _nameController = TextEditingController();
20   final TextEditingController _emailController = TextEditingController();
21   final TextEditingController _phoneController = TextEditingController();
22   final TextEditingController _addressController = TextEditingController();
23 }
```



```

24 Future<String> listData() async {
25   var response = await http.get(Uri.http('10.0.2.2:80', 'emp'),
26     headers: {"Accept": "application/json"});
27   print('Response status: ${response.statusCode}');
28   print('Response body: ${response.body}');
29   setState(() {
30     list = jsonDecode(response.body);
31   });
32   return "Success";
33 }
34
35 @override
36 void initState() {
37   super.initState();
38   listData();
39 }
40
41 @override
42 Widget build(BuildContext context) {
43   return Scaffold(
44     appBar: AppBar(
45       title: Text("DB Test"),
46     ), // AppBar
47     body: Center(
48       child: ListView.builder(
49         itemCount: list.length,
50         itemBuilder: (BuildContext context, int index) {
51           return Card(
52             child: ListTile(
53               title: Row(
54                 children: [
55                   Expanded(child: Text(list[index]["name"])),
56                   Expanded(child: Text(list[index]["email"])),
57                 ],
58               ), // Row
59               leading: Text(list[index]["id"].toString()),
60               trailing: Wrap(
61                 spacing: 5,
62                 children: [
63                   IconButton(
64                     icon: Icon(Icons.edit, color: Colors.green),
65                     onPressed: () {
66                       Map data = {
67                         'id': list[index]['id'],
68                         'name': list[index]["name"],
69                         'email': list[index]['email'],
70                         'phone': list[index]['phone'],
71                         'address': list[index]['address'],
72                       };
73                       _showedit(data);
74                     }, // IconButton
75                   ),
76                   IconButton(
77                     icon:
78                       const Icon(Icons.delete_outline, color: Colors.red),
79                     onPressed: () => _showDel(list[index]["id"]),
80                   ), // IconButton
81                 ],
82               ), // Wrap
83             ), // ListTile
84           ); // Card
85         }, // ListView.builder
86       ), // Center
87       floatingActionButton: FloatingActionButton(
88         child: const Icon(Icons.add),
89         onPressed: () {
90           _addNewDialog();
91         },
92       ), // FloatingActionButton
93     ); // Scaffold
94   }
95

```

model.dart lib

FLUTTER\_INSERT\_...

.dart\_tool

.idea

android

build

ios

lib

main.dart

mainOld.dart

model.dart

**mymain.dart**

test

widget\_test.dart

web

citigore

OUTLINE

TIMELINE

×

mymain.dart lib

main.dart lib

model.dart lib

FLUTTER\_INSERT\_...

.dart\_tool

.idea

android

build

ios

lib

main.dart

mainOld.dart

model.dart

**mymain.dart**

test

widget\_test.dart

web

citigore

OUTLINE

TIMELINE

DEPENDENCIES

×

mymain.dart lib

main.dart lib

model.dart lib

FLUTTER\_INSERT\_...

.dart\_tool

.idea

android

build

ios

lib

main.dart

mainOld.dart

model.dart

**mymain.dart**

test

widget\_test.dart

web

citigore

OUTLINE

TIMELINE

DEPENDENCIES

×

mymain.dart lib

main.dart lib

model.dart lib

FLUTTER\_INSERT\_...

.dart\_tool

.idea

android

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

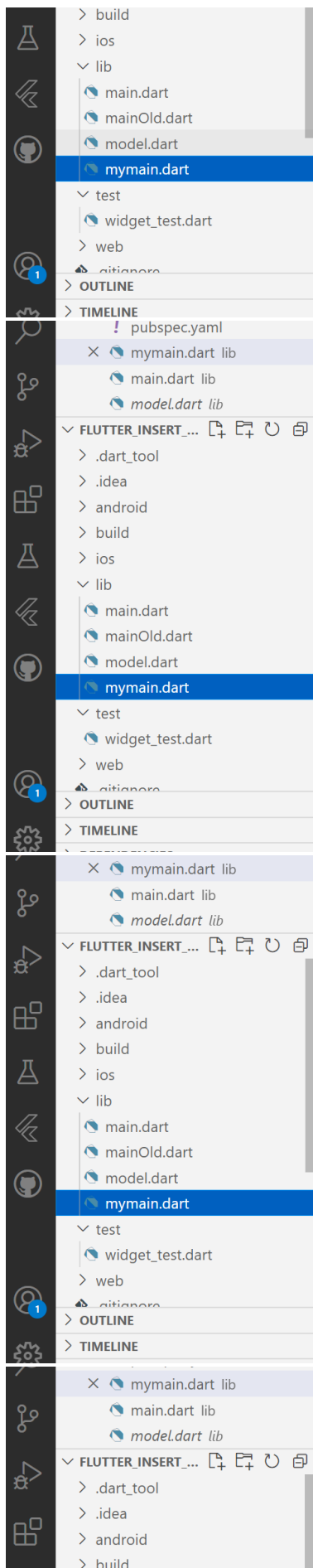
169

170

171

Future<void> \_addNewDialog() async {  
 return showDialog<void>(  
 context: context,  
 barrierDismissible: false, // user must tap button!  
 builder: (BuildContext context) {  
 return AlertDialog(  
 title: const Text('AlertDialog Title'),  
 content: SingleChildScrollView(  
 child: ListBody(  
 children: <Widget>[  
 TextField(  
 controller: \_nameController,  
 decoration: const InputDecoration(  
 hintText: "Enter emp name", labelText: 'Name:'), // InputD  
 ), // TextField  
 TextField(  
 controller: \_emailController,  
 decoration: const InputDecoration(  
 hintText: "Enter emp Email", labelText: 'Email:'), // Input  
 ), // TextField  
 TextField(  
 controller: \_phoneController,  
 decoration: const InputDecoration(  
 hintText: "Enter emp phone", labelText: 'Phone:'), // Input  
 ), // TextField  
 TextField(  
 controller: \_addressController,  
 decoration: const InputDecoration(  
 hintText: "Enter emp Address", labelText: 'Address:'), //  
 ), // TextField  
 const Text('กรอกข้อมูลให้เรียบร้อยแล้วกด ยืนยัน'),  
 ], // <Widget>[]  
 ), // ListBody  
 ), // SingleChildScrollView  
 actions: <Widget>[  
 TextButton(  
 child: const Text('ยืนยัน'),  
 onPressed: () {  
 add\_data();  
 Navigator.of(context).pop();  
 },  
 ), // TextButton  
 ], // <Widget>[]  
 ); // AlertDialog  
 },  
 );  
}

Future<void> \_showDel(int id) async {  
 return showDialog<void>(  
 context: context,  
 barrierDismissible: false, // user must tap button!  
 builder: (BuildContext context) {  
 return AlertDialog(  
 title: Text('ลบข้อมูล id'),  
 content: SingleChildScrollView(  
 child: ListBody(  
 children: const <Widget>[  
 Text('ยืนยันการลบข้อมูล กด ยืนยัน'),  
 ], // <Widget>[]  
 ), // ListBody  
 ), // SingleChildScrollView  
 actions: <Widget>[  
 TextButton(  
 child: const Text('ยืนยัน'),  
 onPressed: () {  
 del\_data(id);  
 Navigator.of(context).pop();  
 },  
 ), // TextButton  
 ], // <Widget>[]  
 ); // AlertDialog  
 },  
 );  
}

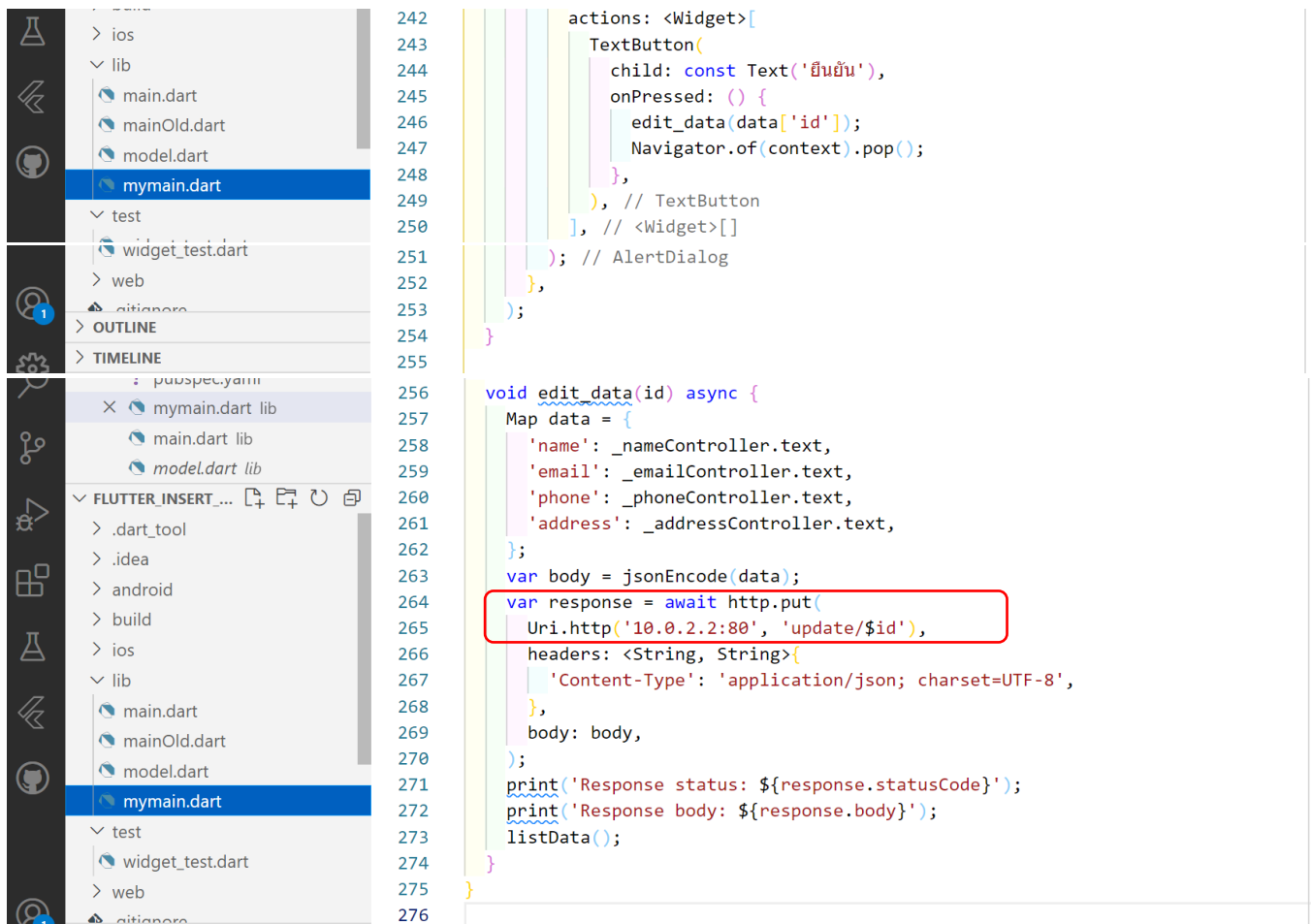


172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241

```
void add_data() async {  
  Map data = {  
    'name': _nameController.text,  
    'email': _emailController.text,  
    'phone': _phoneController.text,  
    'address': _addressController.text,  
  };  
  var body = jsonEncode(data);  
  var response = await http.post(  
    Uri.http('10.0.2.2:80', 'create'),  
    headers: {  
      "Content-Type": "application/json",  
      "Accept": "application/json"  
    },  
    body: body,  
  );  
  print('Response status: ${response.statusCode}');  
  print('Response body: ${response.body}');  
  listData();  
}  
  
void del_data(int id) async {  
  var response = await http.delete(Uri.http('10.0.2.2:80', 'delete/$id'),  
    headers: <String, String>{  
      'Content-Type': 'application/json; charset=UTF-8',  
      "Accept": "application/json"  
    });  
  print('Response status: ${response.statusCode}');  
  print('Response body: ${response.body}');  
  listData();  
}  
  
Future<void> _showedit(Map data) async {  
  _nameController.text = data['name'];  
  _emailController.text = data['email'];  
  _phoneController.text = data['phone'];  
  _addressController.text = data['address'];  
  return showDialog<void>(  
    context: context,  
    barrierDismissible: false, // user must tap button!  
    builder: (BuildContext context) {  
      return AlertDialog(  
        title: const Text('ทดสอบการ Edit'),  
        content: SingleChildScrollView(  
          child: ListBody(  
            children: <Widget>[  
              TextField(  
                controller: _nameController,  
                decoration: const InputDecoration(  
                  hintText: "Enter emp name", labelText: 'Name:'), // InputD  
              ), // TextField  
              TextField(  
                controller: _emailController,  
                decoration: const InputDecoration(  
                  hintText: "Enter emp Email", labelText: 'Email:'), // Inpu  
              ), // TextField  
              TextField(  
                controller: _phoneController,  
                decoration: const InputDecoration(  
                  hintText: "Enter emp phone", labelText: 'Phone:'), // Inpu  
              ), // TextField  
              TextField(  
                controller: _addressController,  
                decoration: const InputDecoration(  
                  hintText: "Enter emp Address", labelText: 'Address:'), //  
              ), // TextField  
              const Text('ปรับปรุงข้อมูลให้เรียบร้อยแล้วกด ยืนยัน'),  
            ], // <Widget>[]  
          ), // ListBody  
        ), // SingleChildScrollView
```

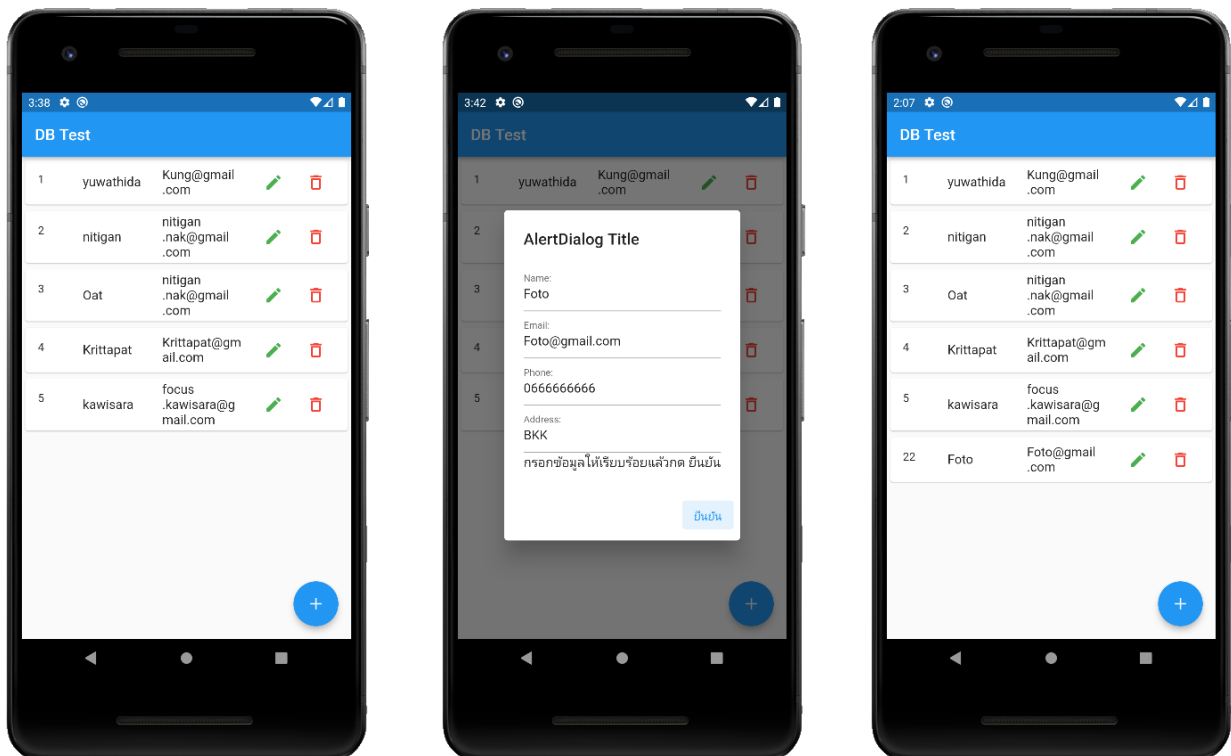
กำหนดหมายเลข port  
ให้ตรงกับ server Web API

กำหนดหมายเลข port  
ให้ตรงกับ server Web API

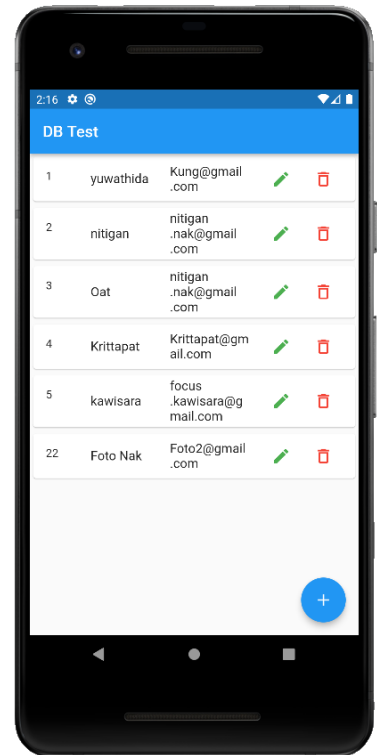
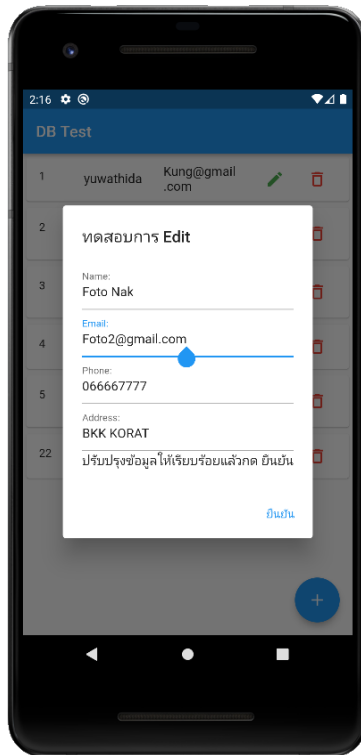
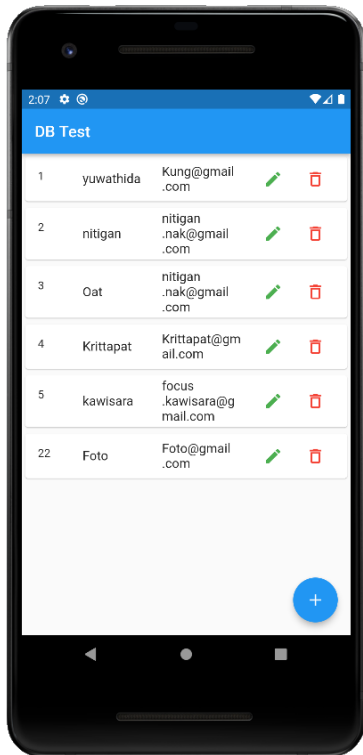


## ทดสอบการทำงานของ Application

### Insert



## Update



## Delete

