日常记录及信息分享服务软件

软件代码清单说明



**课 程**：\_\_\_\_ \_ 软件工程 \_\_ \_\_\_\_ \_\_\_

**题 目**：\_ 基于App的类博客式分享交流软件设计与开发 \_

**专业班级**：\_\_\_\_ \_ \_\_软件工程 1802\_\_\_\_\_\_ \_\_\_

**组 长**：\_\_\_\_\_ \_ \_ G13-陈玲曦-31801349\_ \_\_\_\_ \_\_\_

**组 员**：\_\_\_\_ \_ \_\_ G13-童峻涛-31801341\_\_ \_\_ \_\_\_\_

**组 员**：\_\_\_\_ \_ \_ G13-刘书宇-31801323\_\_ \_ \_ \_\_

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [√] 正在修改 | 文件标识： | SE2020-G13-软件代码清单 |
| 当前版本： | 0.1.1 |
| 作者： | 陈玲曦、童峻涛、刘书宇 |
| 完成日期： | 2020-12-25 |

软件代码清单说明

说明：

《软件代码清单》 描述了软件在开发结束后对所涉及的代码进行清单罗列以及检查相关规范。该文档可能对《软件测试报告》和《软件代码走查说明》以软件维护过程的阶段资料有所补充。

**版本记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 发布日期 | 备注 |
| 01 | 2020-12-25 | 0.1.1 | 童峻涛 | 2020-12-25 | 报告的初始版本制作 |
|  |  |  |  |  |  |

目录

[软件代码清单说明 2](#_Toc59800846)

[1引言 4](#_Toc59800847)

[1.1标识 4](#_Toc59800848)

[1.2系统概述 4](#_Toc59800849)

[1.3文档概述 4](#_Toc59800850)

[1.4基线 5](#_Toc59800851)

[2引用文件 5](#_Toc59800852)

[3前端代码 5](#_Toc59800853)

[3.1模块分解说明 5](#_Toc59800854)

[3.2主要源代码清单 7](#_Toc59800855)

[3.2.1抽象化组件components 7](#_Toc59800856)

[3.2.2全局变量global 22](#_Toc59800857)

[3.2.3抽象类models 24](#_Toc59800858)

[3.2.4页面类pages 28](#_Toc59800859)

[3.2.4.1统计奖励awards 28](#_Toc59800860)

[3.2.4.2动态消息dynamic 35](#_Toc59800861)

[3.2.4.3个人信息personal 48](#_Toc59800862)

[3.2.4.4每日记录record 64](#_Toc59800863)

[3.2.5功能函数utils 79](#_Toc59800864)

[3.2.6统一页面 81](#_Toc59800865)

[4服务器端代码 88](#_Toc59800866)

[4.1模块分解说明 88](#_Toc59800867)

[4.2主要源代码清单 90](#_Toc59800868)

[4.2.1打卡模块Clock 90](#_Toc59800869)

[4.2.2动态模块Dynamic 98](#_Toc59800870)

[4.2.3全局模块Global 103](#_Toc59800871)

[4.2.4三餐模块Meal 107](#_Toc59800872)

[4.2.5用户模块User 109](#_Toc59800873)

[5注解 112](#_Toc59800874)

[附录 112](#_Toc59800875)

**根据《GB8567－06计算机软件产品开发文件编制指南》中项目测试阶段结合实际情况调整后的《软件代码清单说明》内容如下：**

# 1引言

## 1.1标识

标题：《软件代码清单说明》

版本号：0.1.1

说明：本文档支持Office、WPS等word文档阅读软件

## 1.2系统概述

1. **软件概述**

我们小组开发的记录分享软件（Day）能够在手机上记录生活的点点滴滴，同时可以发布分享自己的遇到的有趣的新鲜事，收藏别人的动态。考虑到用户使用的积极性与粘度，我们计划加入打卡模块，通过设置属于自己的专属任务打卡，之后会根据相关情况提供专属奖励。

1. **项目概述**
2. 项目名称：移动端记录分享服务型App应用程序（Day）
3. 项目用途：在学生繁忙的生活中提供一些便利与休闲娱乐
4. 任务提出者：杨枨老师
5. 项目开发者：陈玲曦、刘书宇、童峻涛
6. 用户：浙大城市学院学生
7. 课程名称：《软件工程》
8. 承办小组：G13小组
9. 相关文档：项目的提出和介绍；软件项目计划；软件可行性分析报告；软件需求分析报告；软件总体设计报告；软件详细设计报告；软件测试报告；项目总结报告。

## 1.3文档概述

本文档的内容包括前端代码和服务器端代码等并对其进行详细分析。

本文档的使用除了项目开发团队的内部使用外，还需提交用户和客户组织负责人审查批准。

本文档的使用应遵守国家先关法律法规，未经允许不得对外公开，需要有一定的保密性和私密性。

## 1.4基线

基线是软件生存期各个开发阶段的工作成果。

重要的检查点是里程碑，重要的需要客户确认的里程碑是基线。

主要基线如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 基线名称 | 基线缩写 | 基线标识 | 计划建立日期 | 计划完成时间 |
| 软件需求基线 | SRBL | SR | 2020/11/07 | 2020/11/18 |
| 概要设计基线 | PDBL | PD | 2020/11/19 | 2020/11/26 |
| 详细设计基线 | DDBL | DD | 2020/11/27 | 2020/12/03 |
| 代码基线 | SCBL | SC | 2020/12/04 | 2020/12/20 |
| 测试基线 | STBL | ST | 2020/12/21 | 2020/12/30 |
| 运行基线 | PRBL | PR | 2020/12/31 | 2021/01/13 |

# 2引用文件

[1]《GB T-8567-2006计算机软件文档编制规范》

[2]张海藩，牟永敏.软件工程导论(第6版)[M]北京：清华大学出版社

[3]《Flutter中文网开发者代码规范》. [EB/OL].<https://flutterchina.club/> --2020/12/16

[4]《Google Python Style Guides》

[5]《Android studio 开发者使用手册》

# 3前端代码

## 3.1模块分解说明

|  |  |  |
| --- | --- | --- |
| 序号 | 文件名称 | 功能 |
| 1 | addPics | 添加图片 |
| 2 | circleBedge | 奖章组件 |
| 3 | ClockinItem | 打卡记录组件：新建打卡记录、显示一条打卡记录 |
| 4 | collectDialog | 复选框组件 |
| 5 | customizedToast | 特殊的阴影 |
| 6 | diet | 展示单张图片、没图片则点击+号上传、展示当天吃饭图片 |
| 7 | expandableText | 动态的扩展性 |
| 8 | menu | 我的菜单栏 |
| 9 | nineGrid | 九宫格显示图片 |
| 10 | percentBar | 打卡进度条 |
| 11 | returnButton | 返回按钮组件 |
| 12 | todayCard | 显示时间与格言 |
| 13 | host | 访问服务器的地址端口 |
| 14 | iconTheme | 打卡图标的选择 |
| 15 | info | 使用者信息的保存 |
| 16 | mottos | 库存的格言 |
| 17 | achieve | 成就类 |
| 18 | clockRecord | 打卡记录类 |
| 19 | collection | 个人收藏类 |
| 20 | dynamic | 动态类 |
| 21 | httpRes | 接口调用类 |
| 22 | iconThemeAttribute | 图标主题类 |
| 23 | meal | 一日三餐类 |
| 24 | personInfo | 个人信息类 |
| 25 | weekViewModel | 周视图类 |
| 26 | yearViewModel | 年视图类 |
| 27 | monthView | 月视图页面 |
| 28 | views | 总体布局 |
| 29 | weekView | 周视图页面 |
| 30 | yearView | 年视图页面 |
| 31 | dynamicMain | 动态主页 |
| 32 | oneWidget | 单个动态页面 |
| 33 | publish | 发表动态 |
| 34 | collectionDetail | 收藏夹中的详情页面 |
| 35 | myAchieve | 我的成就页面 |
| 36 | myCollection | 我的收藏页面 |
| 37 | myDynamic | 我的动态的页面 |
| 38 | recentMeal | 最近的饮食页面 |
| 39 | login | 登录页面 |
| 40 | personlPage | 个人信息页面 |
| 41 | register | 注册页面 |
| 42 | setPersonInfo | 设置个人信息页面 |
| 43 | chooseIconTheme | 选择主题图标 |
| 44 | createTask | 创建打卡任务 |
| 45 | recordPage | 每日记录主页面 |
| 46 | setTask | 设置任务的页面 |
| 47 | utils | 功能性函数 |
| 48 | aboutPage | 开发者相关信息 |
| 49 | main | 判断登录是否过期 |
| 50 | mainPage | 主界面的设置 |
| 51 | widget\_test | 测试类 |

## 3.2主要源代码清单

### 3.2.1抽象化组件components

**1. addPics.dart**

1. **import** 'dart:io';
3. **import** 'package:flutter/material.dart';
4. **import** 'package:image\_pickers/image\_pickers.dart';


8. **class** AddPicBox **extends** StatelessWidget
9. {
10. **final** ValueChanged<List<File> > chooseImgCallback;
12. AddPicBox({**this**.chooseImgCallback});
14. @override
15. Widget build(BuildContext context)
16. {
17. **return** GestureDetector(
18. child:
19. Container(
20. padding: EdgeInsets.only(left:5.0, right:5.0, top:10.0,bottom: 10.0),
21. child:
22. SizedBox(
23. width: 100,
24. height: 100,
25. child: DecoratedBox(
26. decoration: BoxDecoration(
27. borderRadius: BorderRadius.circular(3.0),
28. color:Colors.grey[200],
29. ),
30. child: Icon(Icons.add,size: 55,color:Colors.grey)
31. ),
32. )),
33. onTap: () async{
34. print("点击了增加图片");
35. List<Media> \_listImagePaths = await ImagePickers.pickerPaths(
36. galleryMode: GalleryMode.image,
37. selectCount: 6,
38. showGif: **false**,
39. showCamera: **true**,
40. compressSize: 200,
41. uiConfig: UIConfig(uiThemeColor: Colors.lightBlue),
42. cropConfig: CropConfig(enableCrop: **false**, width: 2, height: 1)
43. );
45. List<File> files = **new** List<File>();
46. **for**(**int** i=0;i<\_listImagePaths.length;i++){
47. File file = **new** File(\_listImagePaths[i].path);
48. files.add(file);
49. }
51. chooseImgCallback(files);
53. },
54. );
55. }
56. }

**2. diet.dart**

1. **import** 'dart:convert';
2. **import** 'dart:typed\_data';
3. **import** 'package:cached\_network\_image/cached\_network\_image.dart';
4. **import** 'package:frontend/global/host.dart';
5. **import** 'package:frontend/global/info.dart';
6. **import** 'package:frontend/models/httpRes.dart';
7. **import** 'package:http/http.dart' as http;
8. **import** 'package:flutter/material.dart';
9. **import** 'package:frontend/models/meal.dart';
10. **import** 'package:frontend/utils/utils.dart';
11. **import** 'package:image\_picker/image\_picker.dart';
12. **import** 'package:shared\_preferences/shared\_preferences.dart';
14. /\*
15. 三个组件
16. @ShowMeal -> 展示单张饮食图片
17. @DietBox  -> 有+按钮的方形，点击后选择图片上传
18. @Diet     -> 展示当日饮食图片的组件, 如果三餐都已上传，展示3个ShowMeal；否则展示已上传的ShowMeal组件和一个DietBox组件
19. \*/

22. // ignore: must\_be\_immutable
23. **class** ShowMeal **extends** StatelessWidget
24. {
25. Meal meal;
26. ShowMeal({**this**.meal});


30. Widget build(BuildContext context){
31. **return**
32. Container(
33. margin: EdgeInsets.only(left:5.0, right:5.0, top:10.0,bottom: 10.0),
34. child: Align(
35. alignment: Alignment.bottomLeft,
36. child:Container(
37. height:30.0,
38. width: 100.0,
39. child:Row(
40. mainAxisAlignment: MainAxisAlignment.spaceBetween,
41. children: [
42. Text(meal.type, style:TextStyle( color: Colors.white, fontSize: 14.0)),
43. //Text(meal.timestamp,style:TextStyle(color: Colors.white, fontSize: 10.0))
44. ]
45. ),
46. decoration: BoxDecoration(
47. color: Colors.grey[400].withOpacity(0.8),
48. ),
49. )
50. ),
51. width: 100,
52. height: 100,
53. decoration: BoxDecoration(
54. image: DecorationImage(
55. image: CachedNetworkImageProvider(
56. myHost +  "/meal/show?path="+meal.fileName
57. ),
58. fit: BoxFit.contain,),
59. ),
60. );
61. }
63. }
65. // ignore: must\_be\_immutable
66. **class** DietBox **extends** StatefulWidget
67. {
68. Meal meal;
69. **final** ValueChanged<String > chooseImgCallback;
70. DietBox({**this**.meal, **this**.chooseImgCallback,});
72. @override
73. \_DietBox createState() => \_DietBox(meal: meal, chooseImgCallback: chooseImgCallback);
74. }
76. **class** \_DietBox **extends** State<DietBox>
77. {
78. **final** Meal meal;
79. **final** ValueChanged<String > chooseImgCallback;
80. \_DietBox({ **this**.meal, **this**.chooseImgCallback});

83. **final** picker = ImagePicker();
85. // 如果没有图片，展示“+”按钮，点击后上传图片
86. Widget showNull(){
87. **return** GestureDetector(
88. child:Container(
89. padding: EdgeInsets.only(left:5.0, right:5.0, top:10.0,bottom: 10.0),
90. child:
91. SizedBox(
92. width: 100,
93. height: 100,
94. child: DecoratedBox(
95. decoration: BoxDecoration(
96. borderRadius: BorderRadius.circular(3.0),
97. color:Colors.grey[200],
98. ),
99. child: Icon(Icons.add,size: 55,color:Colors.grey)
100. ),
101. )),
102. onTap: () async{
103. // TODO: 上传图片至服务端
104. **final** pickedFile = await picker.getImage(source: ImageSource.gallery);
105. **if**(pickedFile != **null**){
106. Uint8List imgStr = await pickedFile.readAsBytes();
108. String base64Image = base64Encode(imgStr);
109. String fileName = pickedFile.path.toString().split("/").last;
110. print(fileName);
112. // 使用HTTP协议上传到服务端
113. SharedPreferences \_prefs = await SharedPreferences.getInstance();
115. **int** cnt = \_prefs.getInt(key());
116. **if**(cnt==**null**)
117. {
118. cnt=0;
119. \_prefs.setInt(key(), cnt);
120. }**else**{
121. cnt++;
122. \_prefs.setInt(key(), cnt);
123. }
125. http.Response res = await http.post(myHost+"/meal/upload",
126. body: {
127. "mobile": userMobile.toString(),
128. "fileName": fileName,
129. "image": base64Image,
130. "index": cnt.toString(),
131. }
132. );
133. HttpRes \_res = HttpRes.fromJson(json.decode(res.body.toString()));
134. String imgPath = \_res.type;
136. // 缓存每一餐的信息 e.g "2020-11-2618933928018-0"早餐
137. print("index:"+cnt.toString());
138. print("返回结果"+\_res.type);
139. \_prefs.setString(key()+"-"+cnt.toString(), imgPath);
141. **new** Future.delayed(**new** Duration(microseconds: 1000), ()=>chooseImgCallback(imgPath));
142. }**else**{
143. print('No image selected');
144. }
145. },
146. );
147. }
149. @override
150. Widget build(BuildContext context)
151. {
152. **return** showNull();
153. }
154. }

157. **class** Diet **extends** StatefulWidget
158. {
159. @override
160. \_Diet createState() => \_Diet();
161. }
163. **class** \_Diet **extends** State<StatefulWidget>
164. {
165. // 当日饮食图片
166. List<Meal> images=**new** List();


170. @override
171. **void** initState()
172. {
173. **super**.initState();
175. fetchMeal();
177. }
179. fetchMeal() async
180. {
181. SharedPreferences \_prefs = await SharedPreferences.getInstance();
183. String breakfast = \_prefs.getString(key()+"-0");
184. String lunch = \_prefs.getString(key()+"-1");
185. String dinner = \_prefs.getString(key()+"-2");
187. **if**(breakfast!=**null**){
188. print("早餐"+ breakfast);
189. images.add(Meal(fileName: breakfast, timestamp: "", type: "早餐"));
190. }
191. **if**(lunch!=**null**){
192. print("中餐"+ lunch);
193. images.add(Meal(fileName: lunch, timestamp: "", type: "中餐"));
194. }
195. **if**(dinner!=**null**){
196. print("晚餐"+ dinner);
197. images.add(Meal(fileName: dinner, timestamp: "", type: "晚餐"));
198. }
200. /\* \_prefs.remove(key());
201. \_prefs.remove(key()+"-0");
202. \_prefs.remove(key()+"-1");
203. \_prefs.remove(key()+"-2");\*/


207. setState((){
208. images = images;
210. });

213. }

216. // 子组件上传图片触发
217. choseImageCallback(String file){
219. Meal newMeal = **new** Meal( fileName: file, type:"",timestamp: formateTime());
221. images.add(newMeal);
223. **this**.setState(() {
224. images = images;
225. });
226. }
228. // 展示三餐列表组件
229. Widget showMeals(){
230. List<Widget> mealWidget = **new** List();
231. List<String> types = ["早餐","中餐","晚餐"];
232. **if**(images!=**null**&& images.length>=1){
233. **for**(**int** i=0;i<images.length;i++){
234. **if**(images[i]!=**null**){
235. mealWidget.add(**new** ShowMeal(meal: Meal(fileName: images[i].fileName, timestamp: images[i].timestamp, type: types[i])));
236. }
237. }}
238. **if**(images.length<3&&mealWidget.length>=0){
239. //if(mealWidget.length==0 ||(  mealWidget.length>0&&mealWidget.last.toString()!="DietBox"))
240. mealWidget.add(**new** DietBox(meal: Meal(fileName: **null**,timestamp: **null**,type:"中餐"),chooseImgCallback: choseImageCallback,));
241. }
243. **return** Row(children: mealWidget);
244. }

247. @override
248. Widget build(BuildContext context){
249. **return** Column(
250. crossAxisAlignment: CrossAxisAlignment.start,
251. children: [
252. // “记饮食”
253. Row(
254. mainAxisAlignment: MainAxisAlignment.spaceBetween,
255. children: [
256. Text(
257. "记饮食",
258. style: TextStyle(color:Theme.of(context).primaryColor, fontSize:20.0 , fontWeight: FontWeight.w600),
259. ),
260. images.length==3?Icon(Icons.check,color:Theme.of(context).primaryColor):Text('')
261. ],
262. ),
264. // 至多三个方格展示当日饮食
265. showMeals(),
266. Divider(),
268. // “记打卡”
269. Row(
270. mainAxisAlignment: MainAxisAlignment.spaceBetween,
271. children: [
272. Text(
273. "记打卡",
274. style: TextStyle(color:Theme.of(context).primaryColor, fontSize:20.0 , fontWeight: FontWeight.w600),
275. )
276. ],
277. ),

280. ],);
281. }
283. }

**3. menu.dart**

1. **import** 'package:flutter/material.dart';
2. **import** 'package:frontend/pages/personal/personalFunc/myAchieve.dart';
3. **import** 'package:frontend/pages/personal/personalFunc/myCollection.dart';
4. **import** 'package:frontend/pages/personal/personalFunc/myDynamic.dart';
5. **import** 'package:frontend/pages/personal/personalFunc/recentMeal.dart';
7. **class** MainMenu **extends** StatelessWidget
8. {
10. @override
11. Widget build(BuildContext context)
12. {
13. **return** **new** Container(
14. margin:**const** EdgeInsets.only(left: 10.0, right: 10.0, bottom: 5.0, top:5.0),
15. constraints: **const** BoxConstraints(maxHeight: 250.0),
16. child: **new** ListView(
17. padding: **const** EdgeInsets.only(left:5.0),
18. children: [
19. \_buildListItem("最近饮食", Icons.restaurant,
20. (){
21. Navigator.push(context, MaterialPageRoute(builder: (context)=> **new** RecentMealPage()));
22. }),
23. // \_buildListItem("体重报告", Icons.line\_weight, (){}),
24. \_buildListItem("我的发布", Icons.camera,
25. (){
26. Navigator.push(context, MaterialPageRoute(builder: (context)=>**new** MyDynamics()));
27. })
28. ,
29. \_buildListItem("我的收藏", Icons.favorite, (){
30. Navigator.push(context, MaterialPageRoute(builder: (context)=>**new** MyCollectionsPage()));
31. }),
32. \_buildListItem("我的成就", Icons.military\_tech, (){
33. Navigator.push(context, MaterialPageRoute(builder: (context)=>**new** MyAchievePage()));
34. }),
35. ],
36. )
37. );
38. }


42. Widget \_buildListItem(String title, IconData iconData, VoidCallback action)
43. {
44. **final** textStyle = **new** TextStyle(color:Colors.black54, fontSize:18.0, fontWeight: FontWeight.w600);
46. **return** **new** InkWell(
47. onTap: action,
48. child: **new** Padding(
49. padding: **const** EdgeInsets.only(left: 10.0, right: 10.0, bottom: 5.0, top:5.0),
50. child: **new** Row(
51. mainAxisAlignment: MainAxisAlignment.start,
52. crossAxisAlignment: CrossAxisAlignment.center,
53. children: [
54. **new** Container(
55. width: 35.0,
56. height: 35.0,
57. margin: **const** EdgeInsets.only(right: 10.0),
58. decoration: **new** BoxDecoration(color:Colors.blue, borderRadius: **new** BorderRadius.circular(5.0)),
59. alignment: Alignment.center,
60. child:**new** Icon(iconData, color:Colors.white, size:24.0),
61. ),
62. **new** Text(title, style: textStyle),
63. **new** Expanded(child: **new** Container()),
64. **new** IconButton(
65. icon: **new** Icon(Icons.chevron\_right, color:Colors.black26),
66. onPressed: action,
67. )
68. ],
69. ),
70. )
71. );
72. }
74. }

**4. nineGrid.dart**

1. **import** 'package:cached\_network\_image/cached\_network\_image.dart';
2. **import** 'package:flutter/cupertino.dart';
3. **import** 'package:flutter/material.dart';
4. **import** 'package:frontend/global/host.dart';
6. // ignore: must\_be\_immutable
7. **class** OnePics **extends** StatefulWidget
8. {
9. String path;
10. **double** width;
11. OnePics({**this**.path, **this**.width});
12. @override
13. \_OnePics createState() => \_OnePics(path: path, width:width);
14. }
16. **class** \_OnePics **extends** State<OnePics>
17. {
18. String path;
19. **double** width;
20. \_OnePics({**this**.path, **this**.width});
21. CachedNetworkImage \_img;
23. @override
24. **void** initState(){
25. **super**.initState();
26. loadImg();
27. setState((){
28. path = path;
29. });
30. }
32. loadImg()async{
33. //\_img = await ExtendedImage.network(myHost+"/dynamic/showpic/"+path, cache: false,);
34. \_img = await CachedNetworkImage(
35. imageUrl: myHost+"/dynamic/showpic/"+path, width: width,height: width,fit:BoxFit.cover,
36. errorWidget: (context, str, dy){
37. **return** Center(child:Text("发生错误"));
38. },
39. );
40. setState((){
41. \_img = \_img;
42. });
43. }

46. Widget build(BuildContext context){
47. **return** **this**.path==**null**||**this**.path.length==0 || \_img==**null**?
48. Container(height: width, width: width,
49. decoration: BoxDecoration(color:Colors.grey),
50. ):
51. Container(
52. margin: EdgeInsets.only(left:5.0, right:5.0, top:10.0,bottom: 10.0,),
53. child: \_img,
54. width: width,
55. height: width,
56. );
57. }
59. }

62. // ignore: must\_be\_immutable
63. **class** NineGrid **extends** StatelessWidget
64. {
65. List<String> paths;
66. NineGrid({**this**.paths});

69. buildWidget(){
70. List<Widget> \_widget = **new** List<Widget>();
71. **if**(paths.length<=3){
72. // 1\*n
73. **for**(**int** i=0;i<paths.length;i++){
74. \_widget.add(
75. OnePics(path: paths[i], width: 100,),
76. );
77. }
78. **return** Row(
79. children: \_widget,
80. );
81. }
82. **if**(paths.length==4){
83. // 2\*2
84. \_widget = [OnePics(path: paths[0], width: 140,), OnePics(path: paths[1], width: 140,)];
85. List<Widget> \_widget2 = **new** List<Widget>();
86. \_widget2 = [OnePics(path: paths[2], width: 140,), OnePics(path: paths[3], width: 140,)];
87. **return** Column(children: [
88. Row(children: \_widget,),
89. Row(children: \_widget2,)
90. ],);
91. }
92. // 2\*3
93. \_widget = [OnePics(path: paths[0], width: 100,), OnePics(path: paths[1], width: 100,), OnePics(path:paths[2],width: 100,)];
94. List<Widget> \_widget2 = **new** List<Widget>();
95. **for**(**int** i=3;i<paths.length;i++){
96. \_widget2.add(OnePics(path:paths[i], width: 100,));
97. }
98. **return** Column(children: [
99. Row(children: \_widget),
100. Row(children: \_widget2),
101. ],) ;
102. }
104. @override
105. Widget build(BuildContext context)
106. {

109. **return** buildWidget();
111. }

114. }

**5. returnButton.dart**

1. **import** 'package:flutter/material.dart';

4. **class** ReturnButton **extends** StatelessWidget
5. {
6. @override
7. Widget build(BuildContext context){
8. **return** IconButton(icon:Icon(Icons.chevron\_left, size:28.0),
9. onPressed: (){
10. Navigator.pop(context);
11. },
12. );
13. }
14. }

**6. todayCard.dart**

1. **import** 'package:flutter/material.dart';
2. **import** 'package:frontend/utils/utils.dart';
4. /\*
5. CalendarCard - 显示当日时间卡片
6. 主页上方展示
7. \*/
8. // ignore: must\_be\_immutable
9. **class** CalendarCard **extends** StatelessWidget {
11. // 定时切换motto
12. String motto;
13. CalendarCard({**this**.motto});
15. **final** Map<String, String> cardInfo = {
16. 'title': formatDate(),
17. 'time': weekdayInfo(),
18. 'image': 'assets/images/dailyCard.jpg',
19. };
21. @override
22. Widget build(BuildContext context) {
23. Size size = MediaQuery.of(context).size;
24. **return** Container(
25. child: Align(
26. alignment: Alignment.center,
27. child: Container(
28. height: 200.0,
29. width: 400,
30. child: Column(
31. mainAxisAlignment: MainAxisAlignment.center,
32. children: <Widget>[
33. Text(
34. **this**.cardInfo['title']+" "+**this**.cardInfo['time'],
35. style: TextStyle(color: Colors.white, fontSize: 32.0,fontWeight: FontWeight.w900,
36. ),
37. ),
38. Divider(),
39. Text(
40. motto,
41. //"\"天行健，君子以自强不息\"",
42. //"\"每一个不曾起舞的日子，都是对生命的辜负\"",
43. style: TextStyle(color: Colors.white,fontSize:14.0)
44. )
45. ],
46. ),
47. padding: EdgeInsets.all(20.0),
48. decoration: BoxDecoration(
49. color: Colors.grey[400].withOpacity(0.8),
50. borderRadius: BorderRadius.all( Radius.circular(20.0),),
51. ),
52. ),
53. ),
54. width: size.width, // - 40
55. height: (size.width - 100) / 2,
56. decoration: BoxDecoration(
57. image: DecorationImage(image: AssetImage(**this**.cardInfo['image']),fit: BoxFit.fill,),
58. borderRadius: BorderRadius.all( Radius.circular(20.0),),
59. color: Colors.white70,
60. boxShadow: [
61. BoxShadow( color: Colors.black38, blurRadius: 25.0, offset: Offset(8.0, 8.0), ),
62. ],
63. ),
64. );
65. }
66. }

### 3.2.2全局变量global

**1. iconTheme.dart**

1. **import** 'package:flutter/cupertino.dart';
2. **import** 'package:flutter/material.dart';
3. **import** 'package:flutter\_icons/flutter\_icons.dart';
5. List<IconData> icons = [
6. MaterialIcons.star,
7. MaterialIcons.today,
8. MaterialIcons.sentiment\_satisfied,
9. MaterialIcons.weekend,
10. MaterialIcons.battery\_charging\_full,
11. MaterialIcons.watch,
12. Icons.music\_note,
13. Icons.color\_lens,
14. Icons.emoji\_people,
16. Icons.directions\_run,
17. Icons.directions\_bike,
18. Icons.train,
19. Icons.self\_improvement,
20. Icons.sports\_golf,
21. Icons.sports\_esports,
22. Icons.sports\_cricket,
23. Icons.sports\_handball,
24. Icons.sports\_basketball,
25. Icons.sports\_tennis,
26. Icons.sports\_soccer,
27. Icons.accessible,
28. Icons.alarm\_on,
30. Icons.two\_wheeler,
31. Icons.edit\_location,
32. Icons.fastfood,
33. Icons.local\_cafe,
34. Icons.local\_dining,
35. Icons.cake,
36. Icons.military\_tech , // 勋章
38. Icons.restaurant,
39. Icons.wine\_bar,
40. Icons.room\_service,
41. Icons.store,
43. Icons.local\_grocery\_store,
44. Icons.local\_offer,
45. ];

**2. info.dart**

1. String userMobile = "";
2. String myPwd = 'pwd';
4. **int** totalLen = 0;
6. var myToken = "";

**3. mottos.dart**

1. List<String> mottos = [
2. "\"天行健,君子以自强不息;地势坤,君子以厚德载物\"",
3. "\"每一个不曾起舞的日子,都是对生命的辜负\"",
4. "\"穷且益坚,不坠青云之志\"",
5. "\"但愿每次回忆,对生活都不感到负疚\"",
6. "\"耐心和恒心总会得到报酬的\"",
7. "\"贵有恒,何必三更起五更眠\"",
8. "\"不积跬步,无以至千里\"",
9. "\"功崇惟志,业广惟勤\"",
10. "\"谁有历经千辛万苦的意志,谁就能达到任何目的\"",
11. "\"有志者事竟成\""
12. ];

### 3.2.3抽象类models

**1. dynamic.dart**

1. **class** Dynamic
2. {
3. String hisAvatar;
4. String hisNickname;
5. String hisMobile;
7. String id;
8. DateTime timestamp;
9. String txtContent;
10. String imageList;
11. List<String> paths;
12. String type;
14. Dynamic({
15. **this**.hisAvatar,
16. **this**.hisNickname,
17. **this**.hisMobile,
18. **this**.id,
19. **this**.timestamp,
20. **this**.txtContent,
21. **this**.imageList,
22. **this**.paths,
23. **this**.type,
24. });
26. factory Dynamic.fromJson(Map<String, dynamic> json){
27. **return** Dynamic(
28. hisAvatar: json["user\_avatar"],
29. hisNickname: json["user\_nickname"],
30. hisMobile: json["user\_mobile"].toString(),
31. id: json["id"],
32. timestamp: DateTime.parse(json["timestamp"]),
33. txtContent: json["txt\_content"],
34. imageList: json["image\_list"],
35. paths: **new** List<String>(),
36. type: json["type"]
37. );
38. }
39. }

42. **class** Dynamics {
43. List<Dynamic> dys;
45. Dynamics({**this**.dys});
47. factory Dynamics.fromJson(List<dynamic> json)
48. {
49. List<Dynamic> dys;
50. **return** Dynamics(dys: dys=json.map((i)=>Dynamic.fromJson(i)).toList());
51. }
53. }

**2. clockRecord.dart**

1. **class** ClockRecord {
2. **int** id;
3. String title;
4. String startDate;
5. String endDate;
6. **int** freq;
7. String icon;
8. String color;
9. String cTime;
10. **int** hasTodayTag;
11. **int** hasWeekTaskFinish;
13. ClockRecord(
14. {**this**.id,
15. **this**.title,
16. **this**.startDate,
17. **this**.endDate,
18. **this**.freq,
19. **this**.icon,
20. **this**.color,
21. **this**.cTime,
22. **this**.hasTodayTag,
23. **this**.hasWeekTaskFinish
24. });
26. factory ClockRecord.fromJson(Map<String, dynamic> json) {
27. **return** ClockRecord(
28. id:json['id'],
29. title:json['title'],
30. startDate:json['startDate'],
31. endDate:json['endDate'],
32. freq:json['freq'],
33. icon:json['icon'],
34. color:json['color'],
35. cTime:json['cTime'],
36. hasTodayTag:json['hasTodayTag'],
37. hasWeekTaskFinish: json['hasWeekTaskFinish']
38. );
39. }
41. }

44. **class** ClockRecords{
45. List<ClockRecord> records;
46. ClockRecords({**this**.records});
48. factory ClockRecords.fromJson(List<dynamic> json){
49. List<ClockRecord> records = **new** List<ClockRecord>();
50. **return** **new** ClockRecords(records: records = json.map((i)=>ClockRecord.fromJson(i)).toList());
51. }

54. }

**3. meal.dart**

1. **import** 'dart:typed\_data';
3. **class** Meal
4. {
5. **final** String fileName;
6. **final** String timestamp;
7. **final** String type;
9. Meal({**this**.fileName, **this**.timestamp, **this**.type});
10. }

13. **class** DayMeal
14. {
15. **final** DateTime day;
16. **final** String breakfast;
17. **final** String lunch;
18. **final** String dinner;
20. DayMeal({**this**.day, **this**.breakfast, **this**.lunch, **this**.dinner});
22. factory DayMeal.fromJson(Map<String, dynamic> json)
23. {
24. **return** DayMeal(
25. day: DateTime.parse(json["day"]),
26. breakfast: json["breakfast"],
27. lunch: json["lunch"],
28. dinner: json["dinner"],
29. );
30. }
32. }

35. **class** DayMealList
36. {
37. **final** List<DayMeal> meals;
39. DayMealList({**this**.meals});
41. factory DayMealList.fromJson(List<dynamic> json)
42. {
43. List<DayMeal> meals = **new** List<DayMeal>();
44. **return** DayMealList(meals: meals=json.map((i)=>DayMeal.fromJson(i)).toList());
45. }
47. }

**4. persomInfo.dart**

1. **class** PersonInfo
2. {
3. String mobile;
4. String nickname;
5. String profile;
6. String password;
7. String cTime;
8. String gender;
10. PersonInfo({
11. **this**.mobile,
12. **this**.nickname,
13. **this**.profile,
14. **this**.password,
15. **this**.cTime,
16. **this**.gender
17. });
19. factory PersonInfo.fromJson(Map<String, dynamic> json){
20. **return** PersonInfo(
21. mobile: json["mobile"].toString(),
22. nickname: json["nickname"],
23. profile: json["profile"],
24. password: json["password"],
25. cTime: json["cTime"],
26. gender: json["gender"]
27. );
28. }
30. }

### 3.2.4页面类pages

#### 3.2.4.1统计奖励awards

**1. view.dart**

1. **import** 'package:flutter/material.dart';
2. **import** 'package:frontend/pages/awards/monthView.dart';
3. // import 'package:bubble\_tab\_indicator/bubble\_tab\_indicator.dart';
4. **import** 'package:frontend/pages/awards/weekView.dart';
5. **import** 'package:frontend/pages/awards/yearView.dart';
7. // ignore: must\_be\_immutable
8. **class** AllViewScreen **extends** StatefulWidget
9. {
10. TabController controller;
11. AllViewScreen({**this**.controller});
13. @override
14. \_AllViewScreen createState() => \_AllViewScreen(controller: controller);
15. }
17. **class** \_AllViewScreen **extends** State<AllViewScreen>
18. {
19. TabController controller;
20. \_AllViewScreen({**this**.controller});
22. @override
23. **void** initState(){
24. **super**.initState();
25. }
27. @override
28. Widget build(BuildContext context){
29. **return**
30. TabBarView(
31. controller: controller,
32. children: [ WeekView(),MonthView(), YearView(),],
33. );
34. }

37. }

**2. weekView.dart**

1. **import** 'dart:convert';
2. **import** 'package:dio/dio.dart';
3. **import** 'package:flutter/material.dart';
4. **import** 'package:flutter/rendering.dart';
5. **import** 'package:frontend/components/circleBedge.dart';
6. **import** 'package:frontend/components/percentBar.dart';
7. **import** 'package:frontend/global/host.dart';
8. **import** 'package:frontend/global/iconTheme.dart';
9. **import** 'package:frontend/global/info.dart';
10. **import** 'package:frontend/models/weekViewModel.dart';
11. **import** 'package:frontend/utils/utils.dart';
13. **import** 'package:flutter\_calendar\_week/flutter\_calendar\_week.dart';
15. /\*
16. 周视图
17. \*/
18. **class** WeekView **extends** StatefulWidget
19. {
20. @override
21. \_WeekView createState() => \_WeekView();
22. }
24. **class** \_WeekView **extends** State<WeekView>
25. {
27. **final** CalendarWeekController \_controller = CalendarWeekController();

30. bool isLoading = **true**; // 是否正在加载
31. DateTime selectedDate = DateTime.now();
33. DateTime startDate=DateTime.now(), endDate=DateTime.now();

36. // 有标注的日期 [打卡日]
37. List<DateTime> markedDates =**new** List<DateTime>();
38. List<DayRecordDetail> lists=**new** List<DayRecordDetail>();
40. // 加载的每日打卡记录
41. Map<DateTime, List<DayRecordDetail> > mp=**new** Map<DateTime, List<DayRecordDetail> >();
43. // 每个日期的装饰
44. List<DecorationItem> \_items=**new** List<DecorationItem>();
46. @override
47. **void** initState(){
48. **super**.initState();
49. dataLoad();
50. }
52. // 加载打卡记录数据
53. dataLoad() async{
54. Dio \_dio = **new** Dio();
55. \_dio.options.responseType = ResponseType.plain;
56. Response \_res = await \_dio.get(myHost+"/clockin/week/view", queryParameters: {"mobile":userMobile,}, options:  Options(headers: {'Authorization':myToken}));
58. DayRecords recs = DayRecords.fromJson(json.decode(\_res.data.toString()));
59. List<DayRecordDetail> \_records = recs.records;
61. // 进行按时间的整理归档
62. **for**(var record in \_records){
63. var dayStr = record.date.toString().substring(0,10); // 用日期的前十位，形如2020-11-16作为key
64. var key = DateTime.parse(dayStr);
65. **if**(mp.containsKey(key)){
66. mp[key].add(record);
67. }**else**{
68. mp[key] = [record];
69. markedDates.add(key);
70. }
71. }
73. selectedDate = DateTime.now();
74. **if**(mp.containsKey(DateTime.parse(selectedDate.toString().substring(0,10)))==**true**){
75. lists = mp[ DateTime.parse(selectedDate.toString().substring(0,10))];
76. }
78. // 将时间限制在本周
79. **while**(startDate.weekday != 1){
80. startDate = startDate.subtract(Duration(days:1));
81. }
83. **while**(endDate.weekday != 7){
84. endDate = endDate.add(Duration(days: 1));
85. }
87. mp.forEach((key, value) {
89. List<Widget> \_row=**new** List<Widget>();
90. **for**(var detail in value){
91. \_row.add(Icon(icons[**int**.parse(detail.iconTheme)], color:HexColor(detail.colorTheme)));
92. }
93. DecorationItem \_item = **new** DecorationItem(date: key, decorationAlignment: value.length<=3? FractionalOffset.bottomCenter: FractionalOffset.bottomLeft,decoration: Row(children: \_row));
94. \_items.add(\_item);
95. });
97. setState(() {
98. markedDates =  markedDates;
99. mp = mp;
100. selectedDate = DateTime.now();
101. lists = lists;
102. isLoading = **false**;
103. startDate = startDate;
104. endDate = endDate;
105. \_items = \_items;
106. });
107. }
109. onDayPress(DateTime time){
110. setState((){
111. selectedDate = time;
112. });
113. }

116. // 奖章与成就widget
117. achieve(DateTime select){
118. // 如果被点击的当日无打卡行为, 返回为空
119. var key = DateTime.parse(select.toString().substring(0,10));
120. **if**(mp.containsKey(key)==**false**){
121. **return** Container(
122. margin: EdgeInsets.fromLTRB(25.0, 50, 25.0, 40),
123. child:Center(
124. child: Column(children: [
125. Image.asset("assets/images/non.png"),
126. Container(height:30.0),
127. Text("当日无打卡行为哦", )
128. ],)
129. )
130. );
131. }**else**{
133. // 完成的打卡形成一行成就
135. // 需要根据条目不同来判断
136. Widget \_son;
137. **if**(mp[key].length<=3){
138. List<Widget> achieves = **new** List<Widget>();
139. **for**(var detail in mp[key]){
140. achieves.add(
141. CircleBadge( color:HexColor(detail.colorTheme),title: detail.title, subtitle: detail.detail,)
142. );
143. \_son = Row(mainAxisAlignment: MainAxisAlignment.spaceBetween, children: achieves,);
144. }
145. }**else**{
146. List<Widget> row1 = **new** List<Widget>(), row2 = **new** List<Widget>();
147. **int** mid = mp[key].length==4?2:3;
148. **for**(**int** i=0;i<mid;i++){
149. var detail = mp[key][i];
150. row1.add(CircleBadge(color:HexColor(detail.colorTheme), title: detail.title, subtitle: detail.detail,));
151. }
152. **for**(**int** i=mid;i<mp[key].length;i++){
153. var detail = mp[key][i];
154. row2.add(CircleBadge(color:HexColor(detail.colorTheme), title: detail.title, subtitle: detail.detail,));
155. }
156. \_son = Column(children: [
157. Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly, children: row1),
158. Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly, children: row2),
159. ],);
160. }


164. **return**
165. Container(
166. margin: EdgeInsets.only(top:20.0),
167. //decoration: BoxDecoration(  color: Colors.grey[200]),
168. padding:  EdgeInsets.fromLTRB(5.0, 25.0, 5.0, 15.0),
169. decoration: BoxDecoration(color: Colors.grey[200], boxShadow: [BoxShadow(color: Theme.of(context).primaryColor.withOpacity(0.2), blurRadius: 10.0,spreadRadius: 1)]),
170. child:
172. Column(
174. crossAxisAlignment: CrossAxisAlignment.start,
175. children: [
176. Container(padding:EdgeInsets.symmetric(horizontal:10.0, vertical:0),child:Text('成就', style: TextStyle(fontSize: 18.0, color:Theme.of(context).primaryColor,fontWeight: FontWeight.w600,),),) ,
177. Container(
178. margin: EdgeInsets.only(bottom: 30.0),
179. child: \_son
180. ),
181. PercentBar(len: mp[key].length, totalLen: totalLen),
182. ],));
183. }}


187. @override
188. Widget build(BuildContext context){
189. **return**
190. isLoading==**true**?
191. Center(child:CircularProgressIndicator()):
193. ListView(
194. children: [
195. // 周日历视图
196. Container(
197. decoration: BoxDecoration(boxShadow: [BoxShadow(color:Theme.of(context).primaryColor.withOpacity(0.2),blurRadius: 10, spreadRadius: 1)]),
198. child: CalendarWeek(
199. controller:  \_controller,
200. height: 100,
201. maxDate:endDate,
202. minDate:startDate,
203. onDatePressed: onDayPress,
204. onDateLongPressed: onDayPress,
206. onWeekChanged: () {},
207. weekendsStyle: TextStyle(color: Theme.of(context).primaryColor, fontWeight: FontWeight.w600),
208. dayOfWeekStyle: TextStyle(color: Theme.of(context).primaryColorLight, fontWeight: FontWeight.w600),
209. dayOfWeekAlignment: FractionalOffset.bottomCenter,
210. dateStyle: TextStyle(color: Theme.of(context).primaryColorLight, fontWeight: FontWeight.w400),
211. dateAlignment: FractionalOffset.topCenter,
212. todayDateStyle:TextStyle(color: Colors.orange, fontWeight: FontWeight.w400),
213. todayBackgroundColor: Colors.black.withOpacity(0.20),
214. pressedDateBackgroundColor: Theme.of(context).primaryColor,
215. pressedDateStyle: TextStyle(color: Colors.white, fontWeight: FontWeight.w400),
216. dateBackgroundColor: Colors.transparent,
217. backgroundColor: Colors.white,
218. dayOfWeek: ['周一', '周二', '周三', '周四', '周五', '周六', '周天'],
219. showMonth:**false**,
220. spaceBetweenLabelAndDate: 0,
221. dayShapeBorder: CircleBorder(),
222. decorations: \_items,
223. ),
224. ),
226. // 当日成就
227. achieve(selectedDate),
229. ],);
230. }

233. }

#### 3.2.4.2动态消息dynamic

**1. dynamicMain.dart**

1. **import** 'dart:convert';
3. **import** 'package:dio/dio.dart';
4. **import** 'package:flutter/cupertino.dart';
5. **import** 'package:flutter/material.dart';
6. **import** 'package:flutter\_slidable/flutter\_slidable.dart';
7. **import** 'package:fluttertoast/fluttertoast.dart';
8. **import** 'package:frontend/components/collectDialog.dart';
9. **import** 'package:frontend/components/customizedToast.dart';
10. **import** 'package:frontend/global/host.dart';
11. **import** 'package:frontend/global/info.dart';
12. **import** 'package:frontend/models/dynamic.dart';
13. **import** 'package:frontend/models/httpRes.dart';
14. **import** 'package:frontend/pages/dynamic/oneWidget.dart';
15. **import** 'package:nine\_grid\_view/nine\_grid\_view.dart';
16. **import** 'package:pull\_to\_refresh/pull\_to\_refresh.dart';

19. **class** DynamicMain **extends** StatefulWidget
20. {
21. @override
22. \_DynamicMain createState() => \_DynamicMain();
23. }

26. **class** \_DynamicMain **extends** State<DynamicMain>
27. {
28. bool isLoading = **true**;
29. **int** startPos=0;
31. List<Dynamic> shows = **new** List<Dynamic>();
33. bool hasMore = **true**;
35. // 显示Toast
36. FToast \_fToast;
38. @override
39. **void** initState(){
41. **super**.initState();
42. load(0);
43. \_fToast = FToast();
44. \_fToast.init(context);
45. // showToast(\_fToast, Icons.star, "右滑可以收藏哦");
46. }
48. load(**int** pos) async{
49. //showToast(\_fToast, Icons.star, "右滑可以收藏哦");
50. print("fecth data, start:"+startPos.toString());
51. Dio \_dio = **new** Dio();
52. \_dio.options.responseType = ResponseType.plain;
53. Response \_res = await \_dio.get(myHost+"/dynamic/load/1",queryParameters: {'mobile':userMobile, "start": pos},options: Options(headers: {'Authorization':myToken}));
55. Dynamics dys = Dynamics.fromJson(json.decode(\_res.data.toString()));
56. shows.addAll(dys.dys);
58. **if**(dys.dys==**null** || dys.dys.length==0){
59. setState(() {
60. hasMore = **false**;
61. });
62. }
64. setState((){
65. isLoading = **false**;
66. startPos=pos+3;
67. shows = shows;
68. });
70. }
72. RefreshController \_refreshController = RefreshController(initialRefresh: **false**);
74. **void** \_onRefresh() async{
75. shows = **new** List<Dynamic>();
76. setState(() {
77. shows = shows;
78. });
79. load(0);
80. \_refreshController.refreshCompleted();
81. }
83. **void** \_onLoading() async{
84. load(startPos);
85. \_refreshController.loadComplete();
86. }
88. @override
89. Widget build(BuildContext context)
90. {
92. **return**
93. isLoading==**null** || isLoading==**true**?
94. Center(child:CircularProgressIndicator()):
96. SmartRefresher(
97. enablePullDown: **true**,
98. enablePullUp: **true**,
99. header: WaterDropHeader(),
100. footer: CustomFooter(
101. builder: (BuildContext context, LoadStatus mode){
102. Widget body;
103. **if**(mode==LoadStatus.idle){
104. body = Text("下拉刷新", style: TextStyle(color: Theme.of(context).primaryColor),);
105. }**else** **if**(mode==LoadStatus.loading){
106. body = CupertinoActivityIndicator();
107. }**else** **if**(mode==LoadStatus.failed){
108. body = Text("加载失败，请重试", style: TextStyle(color: Theme.of(context).primaryColor),);
109. }**else** **if**(mode==LoadStatus.canLoading){
110. body=Text("下拉更多",style: TextStyle(color: Theme.of(context).primaryColor),);
111. }**else** **if**(mode==LoadStatus.noMore){
112. body = Text("没有更多数据了",style: TextStyle(color: Theme.of(context).primaryColor),);
113. }
114. **return** Container(height: 50.0, child:Center(child:body));
115. },
116. ),
118. controller: \_refreshController,
119. onRefresh: \_onRefresh,
120. onLoading: \_onLoading,
121. child:
122. ListView.builder(
123. itemCount: shows.length,
124. itemBuilder:(context, index){
125. **return**
126. Slidable(
127. actionPane: SlidableDrawerActionPane(),
128. actionExtentRatio: 0.25,
129. child:  OneWidget(dy: shows[index]),
130. actions: [
131. IconSlideAction(
132. iconWidget: Icon(Icons.star, color:Colors.white),
133. caption: "收藏",
134. color:Colors.red[100],
135. onTap: ()async{
136. print("tap delete");
137. bool \_default=**false**, \_sport=**false**, \_food=**false**, \_daily= **false**;
138. //弹出收藏
139. List<bool> res=await showDialog<List<bool> >(
140. context: context,
141. builder: (context){
142. **return** AlertDialog(
143. title: Text("加入收藏夹"),
144. content: Column(
145. crossAxisAlignment: CrossAxisAlignment.start,
146. mainAxisSize: MainAxisSize.min,
147. children: [
148. Column(children: [
149. Row( children: [
150. Text("默认收藏夹"),DialogCheckbox(  value: \_default, onChanged: (bool value){ \_default = !\_default; },
151. ) ],
152. ),
153. Row(children: [
154. Text("运动"), DialogCheckbox(value: \_sport, onChanged: (bool value){\_sport=!\_sport;})
155. ],),
156. Row(children: [
157. Text("经验"), DialogCheckbox(value: \_daily, onChanged: (bool value){\_daily=!\_daily;})
158. ],),
159. Row(children: [
160. Text("饮食"), DialogCheckbox(value: \_food, onChanged: (bool value){\_food=!\_food;})
161. ],),
162. ],)
163. ],
164. ),
165. actions: [
166. FlatButton(child: Text("取消"),onPressed: ()=>Navigator.of(context).pop(),),
167. FlatButton(child:Text("确定"),
168. onPressed: () async{
169. print("确定");
170. Navigator.of(context).pop([\_default, \_sport, \_daily, \_food]);
172. Dio \_dio = **new** Dio();
173. String collectName="";
174. **if**(\_default){
175. collectName="默认收藏夹";
176. }**else** **if**(\_sport){
177. collectName="运动";
178. }**else** **if**(\_daily){
179. collectName="日常";
180. }**else** {
181. collectName="饮食";
182. }
183. \_dio.options.responseType = ResponseType.plain;
184. var queryParam = {
185. "mobile": userMobile,
186. "collection\_name": collectName,
187. "bind\_dynamic": shows[index].id,
188. };
189. Response \_res = await \_dio.get(myHost+"/dynamic/collect",queryParameters: queryParam,options: Options(headers: {'Authorization':myToken}));
190. HttpRes \_httpRes = HttpRes.fromJson(json.decode(\_res.data.toString()));
191. **if**(\_httpRes.status=='ok'){
192. showToast(\_fToast, Icons.star, "收藏成功");
193. }**else**{
194. showToast(\_fToast, Icons.error, "收藏失败，请重试");
195. }
197. },
198. )
199. ],
200. );
201. }
202. );
203. print(res);
204. },
205. ),
206. ],
208. );
210. }
211. )
212. );
214. }
215. }

**2. publish.dart**

1. **import** 'dart:convert';
2. **import** 'dart:io';
3. **import** 'package:http/http.dart' as http;
4. **import** 'package:dio/dio.dart';
5. **import** 'package:flutter/material.dart';
6. **import** 'package:fluttertoast/fluttertoast.dart';
7. **import** 'package:frontend/components/addPics.dart';
8. **import** 'package:frontend/components/customizedToast.dart';
9. **import** 'package:frontend/components/returnButton.dart';
10. **import** 'package:frontend/global/host.dart';
11. **import** 'package:frontend/global/info.dart';
12. **import** 'package:frontend/models/httpRes.dart';

15. // 用户发布动态界面
16. **class** PublishPage **extends** StatefulWidget
17. {
18. @override
19. \_PublishPage createState() => \_PublishPage();
20. }

23. **class** \_PublishPage **extends** State<PublishPage>
24. {
25. // 用户文本输入
26. TextEditingController \_txtController = **new** TextEditingController();
27. List<File> images = **new** List();
29. // 类型选择
30. String selectedType = "";
31. String selectedSecondType = "";
32. List<Widget> widgets = [];
33. List<String> choices = ["健身", "饮食", "经验"];
34. List<String> descriptions = ["跑步、有氧、无氧运动等相关🏊‍♀️", "日常三餐，减肥食谱，低卡美食, 😋", "日常打卡，成果分享，心路历程🌈"];
35. List<String> formatChoices = ["sport", "food", "experience"];
37. // 是否正在提交
38. bool isSubmitting = **false**;
39. **final** \_formKey = GlobalKey<FormState>();
41. // 是否发布成功
42. bool isSubmitSucc = **false**;
44. // 显示Toast
45. FToast \_fToast;
47. @override
48. **void** initState()
49. {
50. **super**.initState();
51. \_fToast = FToast();
52. \_fToast.init(context);
53. }

56. @override
57. Widget build(BuildContext context)
58. {
59. **return** Scaffold(
60. appBar: AppBar(
61. leading: ReturnButton(),
62. title: Text("发布动态"),centerTitle: **true**,
63. ),
64. body:
65. isSubmitSucc==**true**?
66. Container(
67. alignment: Alignment.center,
68. child: Column(
69. children: [
70. Container(height: 200.0,),
71. Icon(Icons.check, color:Theme.of(context).primaryColor, size:80.0),
72. Text("发布成功!", style:TextStyle(color:Colors.black54, fontWeight: FontWeight.w600, fontSize: 24.0)),
73. ],),
74. ) :
75. Form(
76. key: \_formKey,
77. child:
78. ListView(
79. padding: EdgeInsets.all(15.0),
80. children: [
81. // 多行文本输入文字内容
82. TextFormField(
83. controller: \_txtController,
84. autofocus: **false**,
85. maxLines: 4,
86. decoration: InputDecoration(labelText: "内容", hintText:"记录下你想要分享的动态吧"),
87. // ignore: missing\_return
88. validator: (String value){
89. **if**(value.isEmpty){
90. **return** "发布内容不能为空！";
91. }
92. },
93. ),
95. // 图片选择与展示
96. showAllPics(),
98. // type / secondary\_type选择
99. Text("选择类型", style: TextStyle(color:Colors.black45, fontWeight: FontWeight.w600, fontSize: 20.0)),
100. **for**(var i=0;i<choices.length;i++)
101. RadioListTile(
102. value: formatChoices[i],
103. groupValue: selectedType,
104. title: Text(choices[i]),
105. subtitle: Text(descriptions[i]),
106. onChanged: (currentType){
107. setState((){
108. selectedType = currentType;
109. });
110. },
111. selected: selectedType==formatChoices[i],
112. activeColor: Colors.pink,
113. ),

116. Container(height:50.0),
118. // 发布按钮 => 防止用户点击两次
119. submit(context),
120. ],
121. )
122. )
123. );
125. }

128. callback(List<File> imgs){
129. images.addAll(imgs);
130. setState((){
131. images = images;
132. });
133. }


137. // 展示一张图片 - 图片内容+右上角叉号删除
138. Widget showPic(File img)
139. {
140. **return** Container(
141. margin: EdgeInsets.only(left:5.0, right:5.0, top:10.0,bottom: 10.0),
142. child: Image.file(img, width: 100, height: 100, fit:BoxFit.contain),
143. width: 100,
144. height: 100,
145. );
146. }
148. // 图片Widget
149. Widget showAllPics()
150. {
151. // 根据目前的imgList实时渲染
152. **if**(images.length==0)
153. {
154. print("render null");
155. **return** Row(
156. children: [**new** AddPicBox(chooseImgCallback: callback,)],
157. );
158. }
160. print("render pics");
162. List<Widget> picsWidget = **new** List<Widget>();
164. **for**(**int** i=0;i<images.length;i++)
165. {
166. picsWidget.add(showPic(images[i]));
167. }
169. **if**(images.length<6){
170. picsWidget.add(**new** AddPicBox(chooseImgCallback: callback,));
171. }
173. // 根据当前的picsWidget分行展示widget
174. **if**(picsWidget.length<=3)
175. {
176. **return** Row(
177. children: picsWidget,
178. );
179. }**else**{
180. Widget row1 = **new** Row(children: [picsWidget[0],picsWidget[1], picsWidget[2] ]);
182. List<Widget> append = **new** List<Widget>();
183. **for**(var i=3;i<picsWidget.length;i++){
184. append.add(picsWidget[i]);
185. }
186. Widget row2 = **new** Row(children: append);
187. **return** Column(
188. children: [
189. row1,
190. row2,
191. ],
192. );
193. }
195. }

198. // 发布按钮
199. Align submit(BuildContext context)
200. {
201. **return** Align(
202. child: SizedBox(
203. height: 45.0,
204. width: 270.0,
205. child: FlatButton(
206. child: Text("发布",  style:TextStyle(fontSize:24.0, color:Colors.white, fontWeight: FontWeight.bold)),
207. color:Colors.blue,
208. onPressed: () async{
209. **if**(\_formKey.currentState.validate())
210. {
211. \_formKey.currentState.save();
212. }
214. **if**(images.length==0){
215. showToast(\_fToast,Icons.error, "请选择图片");
216. **return** ;
217. }
218. **if**(selectedType==""){
219. showToast(\_fToast,Icons.error, "请选择类型");
220. **return**;
221. }
223. **if**(isSubmitting){
224. showToast(\_fToast, Icons.error, "正在提交，请勿重复");
225. **return** ;
226. }
228. print("发布～");
229. var queryParams = {
230. "mobile": userMobile==""?"13588631227":userMobile,
231. "txt\_content": \_txtController.text,
232. "type": selectedType,
233. "image\_len": images.length.toString(),
234. };
235. setState((){
236. isSubmitting=**true**;
237. });
238. **for**(**int** i=0;i<images.length;i++)
239. {
240. var key="image\_content"+i.toString();
241. List<**int**> imageBytes = await images[i].readAsBytesSync();
242. queryParams[key]=base64Encode(imageBytes);
244. var nameKey = "image\_name"+i.toString();
245. String name = images[i].toString().split("/").last.toLowerCase();
246. **if**(name[name.length-1]=='\''){
247. name = name.substring(0,name.length-1);
248. print(name);
249. }
250. print(name);
251. queryParams[nameKey]=name;
252. }
254. http.Response res = await http.post(myHost+"/dynamic/publish",
255. body: queryParams
256. );
258. setState(() {
259. isSubmitting = **false**;
260. });
262. HttpRes \_res = HttpRes.fromJson(json.decode(res.body.toString()));
263. **if**(\_res.status=='ok'){
264. //showToast(\_fToast, Icons.info, "发布成功，请等待审核");
265. setState((){
266. isSubmitSucc = **true**;
267. });
268. }**else**{
269. showToast(\_fToast, Icons.error, "发布失败，请重试");
270. }
272. },
273. )
275. )
276. );
277. }
279. }

#### 3.2.4.3个人信息personal

**1. myDanamic.dart**

1. **import** 'dart:convert';
3. **import** 'package:dio/dio.dart';
4. **import** 'package:flutter/material.dart';
5. **import** 'package:flutter\_slidable/flutter\_slidable.dart';
6. **import** 'package:fluttertoast/fluttertoast.dart';
7. **import** 'package:frontend/components/customizedToast.dart';
8. **import** 'package:frontend/components/returnButton.dart';
9. **import** 'package:frontend/global/host.dart';
10. **import** 'package:frontend/global/info.dart';
11. **import** 'package:frontend/models/dynamic.dart';
12. **import** 'package:frontend/models/httpRes.dart';
13. **import** 'package:frontend/pages/dynamic/oneWidget.dart';

16. // 我的动态界面
18. **class** MyDynamics **extends** StatefulWidget
19. {
20. @override
21. \_MyDynamics createState() => \_MyDynamics();
22. }

25. **class** \_MyDynamics **extends** State<MyDynamics>
26. {
27. bool isLoading = **true**;
29. List<Dynamic> shows = **new** List<Dynamic>();
31. // 显示Toast
32. FToast \_fToast;
34. @override
35. **void** initState()
36. {
37. **super**.initState();
38. load();
39. \_fToast = FToast();
40. \_fToast.init(context);
41. }
43. load() async{
45. Dio \_dio = **new** Dio();
46. \_dio.options.responseType = ResponseType.plain;
47. Response \_res = await \_dio.get(myHost+"/dynamic/load/2",queryParameters: {'mobile':userMobile, },options: Options(headers: {'Authorization':myToken}));
49. Dynamics dys = Dynamics.fromJson(json.decode(\_res.data.toString()));
50. shows.addAll(dys.dys);
52. setState((){
53. isLoading = **false**;
54. shows = shows;
55. });
56. showToast(\_fToast, Icons.star, "右滑可以删除哦");
57. }
59. delete(**int** index) async{
60. Dynamic \_del = shows[index];
62. Dio \_dio = **new** Dio();
63. \_dio.options.responseType = ResponseType.plain;
64. var queryParam = {
65. "mobile": userMobile,
66. "dynamic\_id": \_del.id,
67. };
68. Response \_res = await \_dio.get(myHost+"/dynamic/delete",queryParameters: queryParam,options: Options(headers: {'Authorization':myToken}));
69. HttpRes \_httpRes = HttpRes.fromJson(json.decode(\_res.data.toString()));
70. **if**(\_httpRes.status == 'ok'){
71. showToast(\_fToast, Icons.check, "删除成功");
72. shows.removeAt(index);
73. setState((){
74. shows = shows;
75. });
76. }**else** {
77. showToast(\_fToast, Icons.error, "删除失败，请重试");
78. }
80. }

83. @override
84. Widget build(BuildContext context)
85. {
86. **return** Scaffold(
87. appBar: AppBar(title:Text("我的发布"),centerTitle: **true**,
88. leading: ReturnButton(),
89. ),
91. body:
92. isLoading==**true**?
93. Center(child:CircularProgressIndicator())
94. :
96. shows.length==0?
97. (
98. Container(alignment: Alignment.center,
99. child: Column(
100. children: [
101. Container(height: 80.0,),
102. Image.asset("assets/images/non.png"),
103. Container(height: 20.0,),
104. Text("暂时没有发布任何动态哦")
105. ],
106. ),
107. )
108. )
109. :
110. ListView.builder(
111. itemCount: shows.length,
112. itemBuilder:(context, index){
114. // 提供侧滑删除功能
115. **return**
117. Slidable(
118. actionPane: SlidableDrawerActionPane(),
119. actionExtentRatio: 0.25,
120. child:  OneWidget(dy: shows[index]),
121. actions: [
122. IconSlideAction(
123. caption: "删除",
124. color:Colors.red,
125. icon:Icons.delete,
126. onTap: (){
127. print("tap delete");
128. delete(index);
129. // TODO：删除功能
130. // Navigator.push(context, MaterialPageRoute(builder: (context)=> new SetTask(record: item)));
132. },
133. ),
134. ],
135. );

138. }
139. )
140. );

143. }
144. }

**2. recentMeal.dart**

1. **import** 'dart:convert';
3. **import** 'package:flutter/material.dart';
4. **import** 'package:dio/dio.dart';
5. **import** 'package:frontend/components/diet.dart';
6. **import** 'package:frontend/components/returnButton.dart';
7. **import** 'package:frontend/global/host.dart';
8. **import** 'package:frontend/global/info.dart';
9. **import** 'package:frontend/models/meal.dart';
10. **import** 'package:frontend/utils/utils.dart';

13. // 最近饮食界面

16. **class** RecentMealPage **extends** StatefulWidget
17. {
18. @override
19. \_RecentMealPage createState() => \_RecentMealPage();
20. }

23. **class** \_RecentMealPage **extends** State<RecentMealPage>
24. {
25. bool isLoading = **true**;
27. DayMealList \_list;
29. Map<DateTime, DayMeal > mps = **new** Map<DateTime, DayMeal >();


33. @override
34. **void** initState()
35. {
36. **super**.initState();
37. fetchMeals();
38. }
40. fetchMeals() async
41. {
42. Dio \_dio = **new** Dio();
43. \_dio.options.responseType = ResponseType.plain;
44. Response \_res = await \_dio.get(myHost+"/meal/recent", queryParameters: {"mobile":userMobile,}, options:  Options(headers: {'Authorization':myToken}));
46. **try**{
47. \_list = DayMealList.fromJson(json.decode(\_res.data.toString()));
49. setState((){
50. isLoading = **false**;
51. \_list = \_list;
52. });
54. }**catch**(e){
55. print(e);
56. }
58. }
60. @override
61. Widget build(BuildContext context)
62. {
63. **return** Scaffold(
64. appBar: AppBar(leading: ReturnButton(),
65. title: Text("最近饮食"),centerTitle: **true**,
66. ),
67. body:
68. isLoading==**true**?
69. Center(child:CircularProgressIndicator())
70. :
71. ListView.builder(
72. itemCount: \_list.meals.length,
73. itemBuilder: (context, index){
74. **return**
75. Container(
76. margin: EdgeInsets.symmetric(horizontal:20.0, vertical: 15.0),
77. child:
78. Row(
79. mainAxisAlignment: MainAxisAlignment.start,
80. children: [
82. Column(
83. mainAxisAlignment: MainAxisAlignment.start,
84. crossAxisAlignment: CrossAxisAlignment.start,
85. children: [
86. Text( \_list.meals[index].day.toString().substring(0,10)+"  "+fetchWeekDay(\_list.meals[index].day),
87. style: TextStyle(color: Colors.blue, fontSize:20.0, fontWeight: FontWeight.w500),
88. ),
89. Row(
90. children: [
91. ShowMeal(meal:Meal(fileName: \_list.meals[index].breakfast, type:"早餐", timestamp: "")),
92. ShowMeal(meal:Meal(fileName:  \_list.meals[index].lunch, type:"午餐", timestamp: "")),
93. ShowMeal(meal: Meal(fileName: \_list.meals[index].dinner, type:"晚餐", timestamp: ""),)
94. ],
95. ),
96. ],
97. )],
98. ));
99. },
100. )
101. );
103. }
105. }

**3. login.dart**

1. **import** 'dart:convert';
3. **import** 'package:dio/dio.dart';
4. **import** 'package:flutter/material.dart';
5. **import** 'package:fluttertoast/fluttertoast.dart';
6. **import** 'package:frontend/components/customizedToast.dart';
7. **import** 'package:frontend/global/host.dart';
8. **import** 'package:frontend/global/info.dart';
9. **import** 'package:frontend/mainPage.dart';
10. **import** 'package:frontend/models/httpRes.dart';
11. **import** 'package:frontend/pages/personal/register.dart';
12. **import** 'package:shared\_preferences/shared\_preferences.dart';

15. /\*
16. 登陆界面
17. \*/


21. **class** LoginPage **extends** StatefulWidget
22. {
23. @override
24. \_LoginPage createState() => \_LoginPage();
25. }

28. **class** \_LoginPage **extends** State<LoginPage>
29. {
30. TextEditingController \_userController = **new** TextEditingController();
31. TextEditingController \_pwdController = **new** TextEditingController();
32. bool \_isObscure = **true**; //默认显示'\*'
33. Color \_eyeColor=Colors.grey;
35. **final** \_formkey = GlobalKey<FormState>(); //表单控制
36. FToast \_fToast;
38. @override
39. **void** initState(){
40. **super**.initState();
41. \_fToast = FToast();
42. \_fToast.init(context);
43. }

46. @override
47. Widget build(BuildContext context)
48. {
49. **return** Scaffold(
50. body: Form(
51. key: \_formkey,
52. child : ListView(
53. padding: EdgeInsets.symmetric(horizontal: 22.0),
54. children: <Widget>[
55. SizedBox(
56. height: kToolbarHeight,
57. ),
58. //            buildTitle(),
59. //            buildTitleLine(),
60. SizedBox(height:50.0),
61. logoImageArea,
62. SizedBox(height:20.0),
63. buildUserTextFiled(),
64. SizedBox(height:20.0),
65. buildPasswordTextField(context),
66. buildForgetPasswordText(context),
67. SizedBox(height: 25.0),
68. buildLoginButton(context),
69. SizedBox(height:25.0),
70. buildRegisterText(context),
71. ],
72. )
73. )
75. );
76. }
77. // logo 图片区域
78. Widget logoImageArea = **new** Container(
79. alignment: Alignment.topCenter,
81. child: Image.asset(
82. "assets/images/logo.jpg",
83. height: 135,
84. width: 135,
85. fit: BoxFit.cover,
87. ),
88. );
90. //左上方标题栏
91. Padding buildTitle(){
92. **return** Padding(
93. padding: EdgeInsets.all(8.0),
94. child: Text(
95. 'Days',
96. style: TextStyle(fontSize:42.0,color:Colors.blue),
97. ));}
99. //标题下方的横线
100. Padding buildTitleLine(){
101. **return** Padding(
102. padding: EdgeInsets.only(left:12.0, top:4.0),
103. child: Align(
104. alignment: Alignment.bottomLeft,
105. child: Container(
106. color: Colors.blue,
107. width: 80.0,
108. height: 2.0,
109. )
110. )
111. );
112. }
114. //用户名输入框
115. //这里指定输入为手机号码（正则表达式检验)
116. TextFormField buildUserTextFiled(){
117. **return** TextFormField(
118. autofocus: **true**,
119. controller: \_userController,
120. decoration: InputDecoration(
121. labelText: '手机号码',
122. ),
123. validator: (String value){
124. //检查手机号码为11位
125. var userReg = RegExp(r"^1[3456789]\d{9}$");
126. **if**(!userReg.hasMatch(value)){
127. **return** '请输入正确的手机号码';
128. }
129. },
130. );
131. }
133. //密码栏
134. //检验密码不为空，同时设置用户可以切换是否明文看密码
135. TextFormField buildPasswordTextField(BuildContext context)
136. {
137. **return** TextFormField(
138. controller: \_pwdController,
139. obscureText: \_isObscure, //展示为‘\*’
140. validator: (String value){
141. **if** (value.isEmpty){
142. **return** '请输入密码';
143. }}, //检验非空
144. decoration: InputDecoration( //调整密码是否为可见
145. labelText:'密码',
146. suffixIcon: IconButton(
147. icon: Icon(
148. Icons.remove\_red\_eye,
149. color: \_eyeColor,
150. ),
151. onPressed:(){
152. setState((){
153. \_isObscure = !\_isObscure;
154. \_eyeColor = \_isObscure
155. ? Colors.grey : Theme.of(context).primaryColor;
156. });
157. })
158. ),
159. );
160. }
162. //忘记密码一栏，点击即切入忘记密码界面
163. Padding buildForgetPasswordText(BuildContext context)
164. {
165. **return** Padding(
166. padding: **const** EdgeInsets.only(top:8.0),
167. child: Align(
168. alignment: Alignment.centerRight,
169. child: FlatButton(
170. child: Text('忘记密码',style: TextStyle(fontSize:14.0,color:Colors.grey),),
171. onPressed:(){
172. //Navigator.push(context,new MaterialPageRoute(builder: (context)=> ForgetPwd1()));
173. }
174. )
175. )
176. );
177. }
179. //注册新用户入口
180. Align buildRegisterText(BuildContext context) {
181. **return** Align(
182. alignment: Alignment.center,
183. child: Padding(
184. padding: EdgeInsets.only(top: 10.0),
185. child: Row(
186. mainAxisAlignment: MainAxisAlignment.center,
187. children: <Widget>[
188. Text('没有账号？'),
189. GestureDetector(
190. child: Text(
191. '点击注册',
192. style: TextStyle(color: Theme.of(context).primaryColor)
193. ),
194. onTap: () {
195. Navigator.push(context, **new** MaterialPageRoute(builder:(context) => **new** RegisterPage()));
196. },
197. ),
198. ],
199. ),
200. ),
201. );
202. }
204. @override
205. **void** dispose(){
206. \_userController.dispose();
207. \_pwdController.dispose();
208. **super**.dispose();
209. }
211. Align buildLoginButton(BuildContext context)
212. {
213. **return** Align(
214. child: SizedBox(
215. height: 45.0,
216. width: 270.0,
217. child:FlatButton(
218. child: Text('登陆', style:TextStyle(fontSize:24.0, color:Colors.white, fontWeight: FontWeight.bold)),
219. color:Colors.blue,
220. shape: **new** RoundedRectangleBorder(borderRadius: **new** BorderRadius.circular(45.0)),
221. onPressed: () async{
222. **if**(\_formkey.currentState.validate()){
223. \_formkey.currentState.save();
225. Dio \_dio = **new** Dio();
226. var queryParams = {
227. "mobile": \_userController.text,
228. "pwd": \_pwdController.text,
229. };
230. Response \_res = await \_dio.get(myHost+"/user/login",queryParameters: queryParams, );
231. HttpRes \_result = HttpRes.fromJson(json.decode(\_res.data.toString()));
233. **if**(\_result.status=='ok'){
234. print("创建成功");
235. print(\_result.type);
236. String token = \_result.type;
237. myToken = "token "+token;
238. userMobile = \_userController.text;
240. // 将token缓存到本地
241. SharedPreferences \_prefs = await SharedPreferences.getInstance();
242. await \_prefs.setString("token", token);
244. Navigator.push(context, MaterialPageRoute(builder: (context)=>**new** MainPage()));
246. }**else** {
247. showToast(\_fToast,Icons.error, "登陆失败，检查用户名或密码");
248. }
250. }
251. },
252. )
253. ),
254. );
255. }
257. }

**4. setPersonInfo.dart**

1. //import 'dart:convert';
3. //import 'package:dio/dio.dart';
4. **import** 'package:flutter/material.dart';
5. **import** 'package:frontend/components/returnButton.dart';
6. **import** 'package:frontend/global/host.dart';
7. **import** 'package:frontend/models/personInfo.dart';
8. **import** 'package:dio/dio.dart';
9. **import** 'package:frontend/models/httpRes.dart';
10. **import** 'package:dio/dio.dart';
11. **import** 'package:flutter/material.dart';
12. **import** 'package:flutter\_colorpicker/flutter\_colorpicker.dart';
13. **import** 'package:fluttertoast/fluttertoast.dart';
14. **import** 'package:frontend/components/customizedToast.dart';
15. **import** 'package:frontend/components/percentBar.dart';
16. **import** 'package:frontend/components/returnButton.dart';
17. **import** 'package:frontend/global/host.dart';
18. **import** 'package:frontend/global/iconTheme.dart';
19. **import** 'package:frontend/global/info.dart';
20. **import** 'package:frontend/models/clockRecord.dart';
21. **import** 'package:frontend/models/httpRes.dart';
22. **import** 'package:frontend/utils/utils.dart';
23. **import** 'dart:convert';
25. // ignore: must\_be\_immutable
26. **class** SetPersonInfoPage **extends** StatefulWidget
27. {
28. PersonInfo myInfo;
29. SetPersonInfoPage({**this**.myInfo});
31. @override
32. \_SetPersonInfoPage createState() => \_SetPersonInfoPage(myInfo: myInfo);
33. }


37. **class** \_SetPersonInfoPage **extends** State<SetPersonInfoPage>
38. {
39. FToast \_fToast;
40. PersonInfo myInfo;
41. \_SetPersonInfoPage({**this**.myInfo});
42. @override
43. **void** initState(){
44. **super**.initState();
45. \_fToast = FToast();
46. \_fToast.init(context);
48. //fetch();
49. }
50. @override
51. Widget build(BuildContext context)
52. {
53. var change;
54. **return** Scaffold(
55. appBar: AppBar(
56. leading: ReturnButton(),
57. title: Text("修改个人信息"),centerTitle: **true**,
58. ),
59. body:
60. ListView(
61. padding: EdgeInsets.all(20.0),
62. children: [
63. ListTile(leading: Text("头像"), title:
64. ClipOval(child: Image.network(myHost+"/user/profile?path="+myInfo.profile, height: 50.0, width: 50.0)), trailing: Icon(Icons.chevron\_right),
65. onTap:(){
66. print("修改头像");
67. }
68. ),
70. Divider(),
72. ListTile(leading: Text("昵称"), title: Text(myInfo.nickname, style: TextStyle(color:Colors.grey,fontSize:16.0),),
73. trailing: Icon(Icons.chevron\_right),
74. onTap: ()async{
75. showDialog(
76. context: context,
77. builder: (BuildContext context){
78. **return** AlertDialog(
79. title: Text("修改昵称",
80. style: TextStyle(
81. //height: 1.6,
82. color: Colors.blue,
83. //fontSize: 16.0,
84. ),),
85. content: Theme(
86. data: Theme.of(context).copyWith(
87. primaryColor:
88. Colors.blue),
89. child: TextFormField(
90. //controller: \_userName,
91. decoration: InputDecoration(
92. labelText: "昵称",
93. hintText: "输入新的昵称",
94. ),
95. onChanged: (value) {
96. setState(() {
97. change = value;
98. });
99. },
100. //validator:validateUserName
101. )),
102. actions: [
103. FlatButton(child:Text("取消",style:TextStyle(color: Colors.grey)),onPressed: (){Navigator.pop(context);},),
104. FlatButton(child:Text("确定",style:TextStyle(color: Theme.of(context).primaryColor)),onPressed: () async{
105. setState((){
106. myInfo.nickname = change.toString();
107. });
109. // 发起服务端请求
110. Dio \_dio = **new** Dio();
111. \_dio.options.responseType = ResponseType.plain;
112. var queryParams = {
113. "mobile": myInfo.mobile,
114. "key": 'nickname',
115. "val": myInfo.nickname
116. };
117. Response \_res = await \_dio.get(myHost+"/user/change",queryParameters: queryParams,options: Options(headers: {'Authorization':myToken}));
119. HttpRes \_httpRes = HttpRes.fromJson(json.decode(\_res.data.toString()));
120. **if**(\_httpRes.status=='ok'){
121. print("succ");
122. showToast(\_fToast,Icons.check, "修改成功");
123. }**else**{
124. showToast(\_fToast, Icons.error, "修改失败,请重试");
125. }
127. Navigator.pop(context);
128. },),
129. ],
130. );
131. }
132. );
133. },
134. ),
136. Divider(),
138. ListTile(leading:Text("手机"),title: Text(myInfo.mobile, style:TextStyle(color:Colors.grey,fontSize:16.0)),),
140. Divider(),
142. ListTile(leading: Text('修改密码'),
143. trailing: Icon(Icons.chevron\_right),
144. onTap:(){
145. print("修改密码");
146. }
147. ),
149. Divider(),

152. ],
153. )
154. );
155. }
157. }
159. /\*\*
160. \* 验证用户名
161. \*/
162. String validateUserName(value) {
163. **if** (value.isEmpty) {
164. **return** "用户名不能为空";
165. } **else** **if** (value.trim().length < 4 || value.trim().length > 18) {
166. **return** "用户名长度不正确";
167. }
168. **return** **null**;
169. }

#### 3.2.4.4每日记录record

**1. createTask.dart**

1. **import** 'dart:convert';
3. **import** 'package:dio/dio.dart';
4. **import** 'package:flutter/material.dart';
5. **import** 'package:flutter\_icons/flutter\_icons.dart';
6. **import** 'package:frontend/components/customizedToast.dart';
7. **import** 'package:frontend/components/returnButton.dart';
8. **import** 'package:frontend/global/host.dart';
9. **import** 'package:frontend/global/iconTheme.dart';
10. **import** 'package:frontend/global/info.dart';
11. **import** 'package:frontend/models/clockRecord.dart';
12. **import** 'package:frontend/models/httpRes.dart';
13. **import** 'package:frontend/pages/record/chooseIconTheme.dart';
14. **import** 'package:date\_range\_picker/date\_range\_picker.dart' as DateRangePicker;
15. **import** 'package:flutter\_colorpicker/flutter\_colorpicker.dart';
16. **import** 'package:fluttertoast/fluttertoast.dart';
17. **import** 'package:frontend/utils/utils.dart';
19. /\*
20. 新建一个打卡记录界面
21. \*/
23. **class** Tag **extends** StatelessWidget
24. {
25. **final** String content;
26. **final** ValueChanged<String> onChooseTag;
27. Tag({**this**.content, **this**.onChooseTag});

30. @override
31. Widget build(BuildContext context){
32. **return** GestureDetector(
33. onTap:(){
34. onChooseTag(content);
35. },
36. child:
37. Container(
38. padding: EdgeInsets.only(left:5.0, right:5.0, top:10.0,bottom: 10.0),
39. alignment: Alignment.center,
40. width:  (content.length+1).toDouble()\*20,
41. child:
42. SizedBox(
43. height: 30,
44. child: DecoratedBox(
45. decoration: BoxDecoration(borderRadius: BorderRadius.circular(10.0),color:Colors.grey[200],),
46. child:Align(child: Text("#"+content+"#"))
47. )
48. )));
49. }
50. }
52. **class** CreateTask **extends** StatefulWidget
53. {
54. @override
55. \_CreateTask createState() => \_CreateTask();
56. }
58. **class** \_CreateTask **extends** State<CreateTask>
59. {
60. // 新的打卡任务属性
61. IconData \_iconTheme;
62. List<DateTime> \_date;
63. **double** \_freq=1;
64. Color \_colorTheme = Colors.lightBlueAccent;
65. String \_showDate;
66. **int** \_iconIndex=0;
68. // 表单控制
69. TextEditingController \_titleController = **new** TextEditingController();
70. GlobalKey \_taskKey =**new** GlobalKey<FormState>();
72. // 显示Toast
73. FToast \_fToast;
75. @override
76. **void** initState(){
77. **super**.initState();
78. \_iconTheme = **null**;
79. \_fToast = FToast();
80. \_fToast.init(context);
81. }
83. **void** changeColor(Color color) => setState(() => \_colorTheme = color);
84. **void** onTagChosenCallback(String content){
85. setState((){
86. \_titleController.text = content;
87. });
88. }
90. @override
91. Widget build(BuildContext context){
92. **return** Scaffold(
93. appBar: AppBar(title: Text("新建打卡"),centerTitle: **true**,leading: ReturnButton(),),
94. body: Form(
95. key: \_taskKey,
96. child:
97. ListView(
98. padding: EdgeInsets.only( top: 5.0,left: 20.0,right: 20.0,),
99. children: [
100. Container(
101. child: TextField(
102. autofocus: **true**,
103. controller: \_titleController,
104. decoration: InputDecoration(labelText: "任务名",hintText: "开启打卡的任务名",icon:Icon(Icons.alarm\_on),border: InputBorder.none),
105. ),
106. decoration: BoxDecoration(border: Border(bottom: BorderSide(color: Colors.grey[200],width: 1.0)),)
107. ),
109. // 展示热门的tag
110. Container(
111. padding: EdgeInsets.all(10.0),
112. child:Column(
113. children: [
114. Row(children: [
115. Icon(FontAwesome.fire,color:Theme.of(context).primaryColor),Text("热门",style: TextStyle(color:Theme.of(context).primaryColor),)
116. ],),
117. Row(children: [
118. Tag(content: "跑步", onChooseTag: onTagChosenCallback,),
119. Tag(content:"Keep运动",onChooseTag: onTagChosenCallback,),
120. Tag(content:"不喝奶茶",onChooseTag: onTagChosenCallback,),
121. ],),
122. Row(children: [
123. Tag(content:"不挑食",onChooseTag: onTagChosenCallback),
124. Tag(content:"早期",onChooseTag: onTagChosenCallback),
125. Tag(content:"软工作业",onChooseTag: onTagChosenCallback,),
126. Tag(content:"早睡",onChooseTag: onTagChosenCallback,),
127. ],
128. )
129. ],
130. ),
131. decoration: BoxDecoration(borderRadius: BorderRadius.circular(10.0)),
132. ),
133. Divider(),
135. ListTile(title:Text("选择时间区间",),
136. trailing: \_showDate==**null**? Icon(Icons.chevron\_right):Text(\_showDate),
137. onTap:() async{
138. **final** List<DateTime> picked = await DateRangePicker.showDatePicker(
139. context: context,
140. initialFirstDate: **new** DateTime.now(),
141. initialLastDate: (**new** DateTime.now()).add(**new** Duration(days:7)),
142. firstDate: (**new** DateTime.now()).subtract(**new** Duration(days:20)),
143. lastDate: (**new** DateTime.now()).add(**new** Duration(days:360)),
144. );
145. **if** (picked != **null** && picked.length == 2 && picked[1].compareTo(DateTime.now())>0 ) {
146. print(picked);
147. String show = picked[0].toString().substring(0,10) + "至" + picked[1].toString().substring(0,10);
148. setState((){
149. \_date = picked;
150. \_showDate = show;
151. });
152. }**else**{
153. showToast(\_fToast, Icons.error, "请选择合法时间");
154. }
155. }
156. ),
157. Divider(),
159. // 选择频率
160. Row(
161. children: [
162. Container(width: 15.0,),
163. Text("选择频率",style:TextStyle(fontSize: 18.0)),
164. Slider(value: \_freq, min:1,max:7,divisions: 6,label: "每周"+\_freq.round().toString()+"次",
165. onChanged: (**double** value){
166. setState((){
167. \_freq = value;
168. });
169. },
170. ),
171. Text( "每周"+\_freq.round().toString()+"次",style:TextStyle(fontSize: 12.0))
172. ],
173. ),
175. Divider(),
177. ListTile(title:Text("选择主题"),trailing: Icon( \_iconTheme==**null**? Icons.chevron\_right: \_iconTheme, color: \_iconTheme==**null**?  Colors.grey: Theme.of(context).primaryColor),
178. onTap: ()async {
179. // 等待前一界面返回结果
180. **final** \_icon = await  Navigator.push(context, MaterialPageRoute(builder: (context)=>**new** IconsPage()));
181. **if**(\_icon>0){
182. setState(() {
183. \_iconTheme = icons[\_icon];
184. \_iconIndex = \_icon;
185. });
186. }
187. },
188. ),
189. Divider(),

192. ListTile(title:Text("选择主题色"),trailing: Icon( Icons.color\_lens,color:\_colorTheme),
193. onTap: (){
194. showDialog(
195. context: context,
196. builder: (BuildContext context){
197. **return** AlertDialog(
198. titlePadding: **const** EdgeInsets.all(0.0),
199. contentPadding: **const** EdgeInsets.all(0.0),
200. content: SingleChildScrollView(
201. child:
202. MaterialPicker(
203. pickerColor: \_colorTheme,
204. onColorChanged: changeColor,
205. enableLabel: **true**,
206. ),
207. ),
208. actions: [
209. FlatButton(child:Text("取消",style:TextStyle(color: Colors.grey)),onPressed: (){Navigator.pop(context);},),
210. FlatButton(child:Text("确定",style:TextStyle(color: Theme.of(context).primaryColor)),onPressed: (){Navigator.pop(context);},),
211. ],
212. );
213. }
214. );
215. },
216. ),
218. Divider(),
220. Container(height: 20.0,),
222. FloatingActionButton.extended(
223. onPressed: ()async{

226. print(\_titleController.text);
227. print(\_freq.toString());
229. **if**(\_titleController.text=="" || \_freq.toInt()==0 || \_iconTheme==**null** || \_date.length!=2){
230. showToast(\_fToast,Icons.error, "请检查表单");
231. **return** ;
232. }

235. var queryParams = {
236. "mobile":userMobile,
237. "startDate":\_date[0].toString().substring(0,10),
238. "endDate":\_date[1].toString().substring(0,10),
239. "color": encodeColor(\_colorTheme.toString()), //提取color对应的int
240. "icon": \_iconIndex, //存储icon对应的index
241. "title": \_titleController.text,
242. "freq":\_freq.toInt(),
243. };
244. print(queryParams);
246. Dio \_dio = **new** Dio();
247. \_dio.options.responseType = ResponseType.plain;
248. Response \_res = await \_dio.get(myHost+"/clockin/create",queryParameters: queryParams,options: Options(headers: {'Authorization':myToken}));
250. HttpRes \_httpRes = HttpRes.fromJson(json.decode(\_res.data.toString()));
251. // 将新对象传递给前页
252. **if**(\_httpRes.status=='ok'){
253. showToast(\_fToast,Icons.check, "创建成功");
254. ClockRecord newRecord = ClockRecord(
255. id:**int**.parse( \_httpRes.type),
256. title: queryParams["title"],
257. startDate: queryParams["startDate"],
258. endDate: queryParams["endDate"],
259. freq: queryParams["freq"],
260. icon: queryParams["icon"].toString(),
261. color: queryParams["color"],
262. cTime: DateTime.now().toString().substring(0,10),
263. hasTodayTag: 0,
264. );
265. // 如果今天也在打卡日子中
266. **if**(\_date[0].compareTo(DateTime.now())<=0 && \_date[1].compareTo(DateTime.now())>=0){
267. Navigator.pop(context, newRecord);
268. }**else**{
269. Navigator.pop(context);
270. }
272. }**else** **if**(\_httpRes.status=='error' && \_httpRes.type=='over length'){
273. showToast(\_fToast,Icons.error, "超过最多打卡数目限制!");
274. }**else** {
275. showToast(\_fToast,Icons.error, "创建失败");
276. }

279. },
280. backgroundColor: Theme.of(context).primaryColor,
281. label: Row(children: [
282. Icon(Icons.send),
283. Text("新建任务")
284. ],)
285. )
286. ],)
287. ));
288. }
290. /\*\_showToast(IconData icon, String hintText){
291. \_fToast.showToast(child: CusToast(icon:icon, hintText:hintText),
292. gravity: ToastGravity.BOTTOM,
293. toastDuration: Duration(seconds:2),
294. );
295. }\*/
297. }

**2. recordPage.dart**

1. **import** 'dart:async';
2. **import** 'dart:convert';
3. **import** 'dart:math';
4. **import** 'package:flutter/cupertino.dart';
5. **import** 'package:pull\_to\_refresh/pull\_to\_refresh.dart';
6. **import** 'package:dio/dio.dart';
7. **import** 'package:flutter/material.dart';
8. **import** 'package:frontend/components/clockinItem.dart';
9. **import** 'package:frontend/components/diet.dart';
10. **import** 'package:frontend/components/todayCard.dart';
11. **import** 'package:frontend/global/host.dart';
12. **import** 'package:frontend/global/info.dart';
13. **import** 'package:frontend/global/mottos.dart';
14. **import** 'package:frontend/models/clockRecord.dart';
16. /\*
17. 每日记录主页
18. \*/
20. **class** Record **extends** StatefulWidget
21. {
22. @override
23. \_Record createState() => \_Record();
24. }
26. **class** \_Record **extends** State<Record>  with TickerProviderStateMixin
27. {
28. bool allFinish = **false**;
29. List<ClockRecord> \_clocks;
31. Timer \_timer;
32. String motto=mottos[0]; // 每隔5min切换一次motto
34. // 按照待打卡任务是否已经完成本周任务进行区分
35. List<ClockRecord> \_finishWeekList=**new** List<ClockRecord>();
36. List<ClockRecord> \_notFinishWeekList=**new** List<ClockRecord>();
37. bool \_show = **true**;
39. RefreshController \_refreshController = RefreshController(initialRefresh: **false**);
41. @override
42. **void** initState(){
43. **super**.initState();
45. //mockLogin()
46. \_loadItems();
48. // 设置定时切换motto的任务
49. \_timer = Timer.periodic(Duration(seconds: 30), (timer) {
50. String newMotto = mottos[Random().nextInt(8)];
51. setState((){
52. motto = newMotto;
53. });
54. });

57. }
59. // 模拟登陆
60. mockLogin() async{
61. Dio \_dio = **new** Dio();
62. Response \_res = await \_dio.get(myHost+"/user/login",queryParameters: {'mobile':userMobile,'pwd':myPwd});
63. // 赋值给token
64. myToken = \_res.toString();
65. }

68. // 加载今天需要打卡的条目
69. \_loadItems() async{
70. Dio \_dio = **new** Dio();
71. \_dio.options.responseType = ResponseType.plain;
72. Response \_res = await \_dio.get(myHost+"/clockin/load",queryParameters: {'mobile':userMobile},options: Options(headers: {'Authorization':myToken}));
73. var \_result = \_res.data.toString();
74. var \_jsonResult = json.decode(\_result);
75. List<ClockRecord> \_c = ClockRecords.fromJson(\_jsonResult).records;
77. setState(() {
78. \_clocks = \_c;
79. totalLen = \_clocks.length;
80. });
81. split();
82. }
84. // 下拉刷新
85. **void** \_onRefresh() async{
86. \_loadItems();
87. \_refreshController.refreshCompleted();
88. }
90. // 下拉加载
91. **void** \_onLoading() async{
92. await Future.delayed(Duration(microseconds: 1000));
93. \_refreshController.loadComplete();
94. }
96. // 用户创建了新的打卡任务时回调
97. onCreatedNew(ClockRecord newRecord){
98. \_clocks.add(newRecord);
99. totalLen = \_clocks.length;
100. print("new Record:"+newRecord.title);
101. split();
102. }

105. split(){
106. \_finishWeekList=**new** List<ClockRecord>();
107. \_notFinishWeekList=**new** List<ClockRecord>();
108. **for**(**int** i=0;i<\_clocks.length;i++){
109. **if**(\_clocks[i].hasWeekTaskFinish==1){
110. \_finishWeekList.add(\_clocks[i]);
111. }**else**{
112. \_notFinishWeekList.add(\_clocks[i]);
113. }
114. }
116. setState((){
117. \_finishWeekList = \_finishWeekList;
118. \_notFinishWeekList = \_notFinishWeekList;
119. });
121. }
123. // 用户完成今日打卡回调
124. onFinish(String title){
125. **for**(**int** i=0;i<\_notFinishWeekList.length;i++){
126. **if**(\_notFinishWeekList[i].title==title){
127. \_notFinishWeekList[i].hasTodayTag = 1;
128. setState(() {
129. \_notFinishWeekList = \_notFinishWeekList;
130. });
131. **return** ;
132. }
133. }
134. **for**(**int** i=0;i<\_finishWeekList.length;i++){
135. **if**(\_finishWeekList[i].title==title){
136. \_finishWeekList[i].hasTodayTag = 1;
137. setState(() {
138. \_finishWeekList = \_finishWeekList;
139. });
140. **return** ;
141. }
142. }
143. }
145. onDelete(String title){
146. **for**(**int** i=0;i<\_clocks.length;i++){
147. **if**(\_clocks[i].title==title){
148. \_clocks.removeAt(i);
149. **break**;
150. }
151. }
153. **for**(**int** i=0;i<\_notFinishWeekList.length;i++){
155. **if**(\_notFinishWeekList[i].title==title){
156. \_notFinishWeekList.removeAt(i);
157. print("found and delete");
158. setState(() {
159. totalLen = \_clocks.length;
160. \_clocks = \_clocks;
161. \_notFinishWeekList = \_notFinishWeekList;
162. });
163. **return** ;
164. }
165. }
166. **for**(**int** i=0;i<\_finishWeekList.length;i++){
167. **if**(\_finishWeekList[i].title==title){
168. \_finishWeekList.removeAt(i);
169. setState(() {
170. totalLen = \_clocks.length;
171. \_clocks = \_clocks;
172. \_finishWeekList = \_finishWeekList;
173. });
174. **return** ;
175. }
176. }
178. }

181. @override
182. Widget build(BuildContext context){
183. **return**
184. SmartRefresher(
185. enablePullDown: **true**,
186. enablePullUp: **true**,
187. header: WaterDropHeader(),
188. footer:
189. CustomFooter(
190. builder: (BuildContext context, LoadStatus mode){
191. Widget body;
192. **if**(mode==LoadStatus.idle){
193. body = Text("下拉刷新", style: TextStyle(color: Theme.of(context).primaryColor),);
194. }**else** **if**(mode==LoadStatus.loading){
195. body = CupertinoActivityIndicator();
196. }**else** **if**(mode==LoadStatus.failed){
197. body = Text("加载失败，请重试", style: TextStyle(color: Theme.of(context).primaryColor),);
198. }**else** **if**(mode==LoadStatus.canLoading){
199. body=Text("下拉更多",style: TextStyle(color: Theme.of(context).primaryColor),);
200. }**else** {
201. body = Text("没有更多数据了",style: TextStyle(color: Theme.of(context).primaryColor),);
202. }
203. **return** Container(height: 50.0, child:Center(child:body));
204. },
205. ),
206. controller: \_refreshController,
207. onRefresh: \_onRefresh,
208. onLoading: \_onLoading,
209. child: ListView(
210. padding: EdgeInsets.only( top: 20.0,left: 20.0,right: 20.0,),
211. children:<Widget>[
213. CalendarCard(motto: motto,),
214. Container(height:20.0),
215. Diet(),
216. \_clocks ==**null**?  Center( child:CircularProgressIndicator(strokeWidth: 4,)) :
217. // ClockIns(list: \_clocks,),
219. Container(
220. child: Column(children: [
221. **for**(**int** i=0;i<\_notFinishWeekList.length;i++)
222. ClockinItem(item: \_notFinishWeekList[i], index:i, onFinish: onFinish,onDelete: onDelete,),
223. EmptyClockinItem(onCreate: onCreatedNew,),
225. \_finishWeekList.length>0?
226. Column(
227. children: [
228. GestureDetector(child:
229. Row(
230. children: [
231. Expanded(child: Divider(color: Theme.of(context).primaryColor),),
232. Text(" 点击查看本周已完成的任务 ",style: TextStyle(color: Theme.of(context).primaryColor),),
233. Expanded(child:Divider(color: Theme.of(context).primaryColor)),
234. ],
235. ),
236. onTap:(){
237. setState((){
238. \_show = !\_show;
239. });
240. }
241. ),
242. Offstage(
243. offstage: \_show,
244. child: Column(children: [
245. **for**(**int** i=0;i<\_finishWeekList.length; i++)
246. ClockinItem(item: \_finishWeekList[i], index:i, onFinish: onFinish,onDelete: onDelete,),
247. ],)
248. )
249. ],
250. ):Container()
252. ],),
253. )


257. ])
258. );
259. }





266. }

### 3.2.5功能函数utils

**1. utils.dart**

1. **import** 'package:flutter/material.dart';
2. **import** 'package:frontend/global/info.dart';
3. **import** 'package:shared\_preferences/shared\_preferences.dart';
5. String formatDate (){
6. var date = DateTime.now();
7. **return** date.year.toString()+"-"+date.month.toString()+"-"+date.day.toString();
8. }
10. String formateTime() {
11. var hour = DateTime.now().hour.toString();
12. **if**(hour.length==1){
13. hour = "0"+hour;
14. }
15. var minute = DateTime.now().minute.toString();
16. **if**(minute.length==1){
17. minute = "0"+minute;
18. }
20. **return** hour+":"+minute;
21. }
23. String weekdayInfo(){
24. List weekdayList = ['','周一','周二','周三','周四','周五','周六','周末'];
26. **return** weekdayList[DateTime.now().weekday];
27. }
29. String fetchWeekDay(DateTime time)
30. {
31. List weekdayList = ['','周一','周二','周三','周四','周五','周六','周末'];
33. **return** weekdayList[time.weekday];
34. }

37. // 色彩提取 遍历找到第一个出现0x...的
38. String encodeColor(String color){
39. **for**(**int** i=0;i<color.length;i++){
40. **if**( color[i]=='0' && i+1<color.length && color[i+1]=='x')
41. {
42. **for**(**int** j=i+1;j<color.length+1;j++){
43. **if**(color[j]==')'){
44. **return** color.substring(i+2,j);
45. }
46. }
47. }
48. }
49. **return** "0xffeeff41"; // 默认颜色
50. }
52. // 解析color
53. **class** HexColor **extends** Color {
54. **static** **int** \_getColorFromHex(String hexColor) {
55. hexColor = hexColor.toUpperCase().replaceAll("#", "");
56. **if** (hexColor.length == 6) {
57. hexColor = "FF" + hexColor;
58. }
59. **return** **int**.parse(hexColor, radix: 16);
60. }
62. HexColor(**final** String hexColor) : **super**(\_getColorFromHex(hexColor));
63. }

66. Future<**void**> reset() async{
67. userMobile = "";
68. myToken = "";
69. SharedPreferences \_prefs =  await SharedPreferences.getInstance();
70. \_prefs.setString("token", **null**);
71. }

74. String key()
75. {
76. **return** DateTime.now().toString().substring(0,10)+userMobile;
77. }

### 3.2.6统一页面

**1. aboutPage.dart**

1. **import** 'package:flutter/material.dart';
2. **import** 'package:frontend/components/returnButton.dart';

5. **class** AboutDay **extends** StatelessWidget {
6. @override
7. Widget build(BuildContext context) {
8. **return** MaterialApp(
9. title: 'AboutDay',
10. // 创建登陆界面对象
11. color: Colors.grey[300],
12. home: AboutDayPage(),
13. );
14. }
15. }
17. **class** AboutDayPage **extends** StatefulWidget {
18. @override
19. AboutDayPageState createState() => AboutDayPageState();
20. }
22. **class** AboutDayPageState **extends** State<AboutDayPage> {
24. Widget build(BuildContext context) {
25. // logo 图片区域
26. Widget logoImageArea = **new** Container(
27. alignment: Alignment.topCenter,
28. // 设置图片为圆形
30. child: Image.asset(
31. "assets/images/logo.jpg",
32. height: 125,
33. width: 125,
34. fit: BoxFit.cover,
35. ),
36. );



41. **return** Scaffold(
42. appBar: AppBar(title:Text("关于我们"),centerTitle: **true**,leading: ReturnButton(),),
43. body: **new** GestureDetector(
44. onTap: () {
45. // 点击空白区域，回收键盘
46. print("点击了空白区域");
47. },
49. child: **new** ListView(
50. children: <Widget>[
51. **new** SizedBox(height: 10.0),
52. **new** SizedBox(height: 20.0),
53. logoImageArea,
54. //wordSayArea,
55. **new** Text ('Days の 小小日记',textAlign: TextAlign.center,
56. style: TextStyle(
57. height: 4.0,
58. color: Colors.blue[400],
59. fontSize: 20.0,
60. ),),
61. **new** Text ('\n作者：二C、Anchovy\n版本信息：v1.0\n发布时间：2020-12-20\n',textAlign: TextAlign.center,
62. style: TextStyle(
63. //height: 4.0,
64. color: Colors.blueGrey,
65. fontSize: 17.0,
66. ),),
67. **new** Text( '有关该应用的功能信息方面，有待后续完善.\n希望大家可以持续关注，并提出您宝贵的意见与建议！',textAlign: TextAlign.center,
68. style: TextStyle(
69. //height: 4.0,
70. color: Colors.black,
71. fontSize: 14.0,
72. ),),
73. **new** Text( '邮箱：31801341@stu.zucc.edu.cn',textAlign: TextAlign.center,
74. style: TextStyle(
75. //height: 4.0,
76. color: Colors.blueAccent,
77. fontSize: 14.0,
78. ),),
79. ],
80. ),
81. ),
82. );
83. }
85. }

**2. main.dart**

1. **import** 'package:flutter/material.dart';
2. **import** 'package:flutter\_localizations/flutter\_localizations.dart';
3. **import** 'package:frontend/global/info.dart';
4. **import** 'package:frontend/mainPage.dart';
5. **import** 'package:frontend/pages/personal/login.dart';
6. **import** 'package:jwt\_decoder/jwt\_decoder.dart';
7. **import** 'package:shared\_preferences/shared\_preferences.dart';
9. **void** main() {
10. runApp(MyApp());
11. }
13. **class** MyApp **extends** StatelessWidget {
14. @override
15. Widget build(BuildContext context) {
16. **return** MaterialApp(
17. localizationsDelegates: [
18. GlobalMaterialLocalizations.delegate,
19. GlobalWidgetsLocalizations.delegate,
20. ],
21. supportedLocales: [
22. **const** Locale('zh','CH'),
23. **const** Locale('en','US')
24. ],
25. locale: Locale('zh'), // 选择中文环境
26. title: 'Day',
27. theme: ThemeData(
28. primarySwatch: Colors.lightBlue,
29. unselectedWidgetColor: Colors.grey,
30. visualDensity: VisualDensity.adaptivePlatformDensity,
31. ),
32. home:
33. StartApp(),
34. debugShowCheckedModeBanner: **false**,
35. );
36. }
37. }

40. // 根据token是否在有效期判断是登陆 / 直接进入用户主页
41. **class** StartApp **extends** StatefulWidget
42. {
43. @override
44. \_StartApp createState() => \_StartApp();
45. }
47. **class** \_StartApp **extends** State<StartApp>
48. {
49. bool loginValid;
51. @override
52. **void** initState(){
53. **super**.initState();
55. check();
56. }
58. check() async{
59. SharedPreferences \_prefs = await SharedPreferences.getInstance();
60. bool tmp = **false**;
62. String token = \_prefs.getString("token");
63. **if**(token!=**null**){
64. **if**(JwtDecoder.isExpired(token)==**false**){
65. tmp=**true**;
66. myToken = "token " + token;
68. Map<String, dynamic> decodedToken = JwtDecoder.decode(token);
69. userMobile = decodedToken["data"]["mobile"].toString();
70. }
71. }
72. print("登陆是否有效:"+tmp.toString());
73. setState((){
74. loginValid = tmp;
75. });
76. }
78. @override
79. Widget build(BuildContext context){
80. **return** //LoginPage();
81. loginValid==**null**?
82. Center(child:CircularProgressIndicator())
83. :
84. (loginValid==**true**?
85. MainPage():LoginPage()
86. );
88. }
89. }

**3. mainPage.dart**

1. **import** 'dart:convert';
3. **import** 'package:bubble\_tab\_indicator/bubble\_tab\_indicator.dart';
4. **import** 'package:flutter/material.dart';
5. **import** 'package:frontend/components/circleBedge.dart';
6. **import** 'package:frontend/global/info.dart';
7. **import** 'package:frontend/pages/awards/views.dart';
8. **import** 'package:frontend/pages/dynamic/dynamicMain.dart';
9. **import** 'package:frontend/pages/dynamic/publish.dart';
10. **import** 'package:frontend/pages/personal/personalPage.dart';
11. **import** 'package:frontend/pages/record/recordPage.dart';
12. **import** 'package:web\_socket\_channel/io.dart';
13. **import** 'package:frontend/aboutPage.dart';
15. **class** MainPage **extends** StatefulWidget
16. {
17. @override
18. \_MainPage createState() => \_MainPage();
19. }
21. **class** \_MainPage **extends** State<MainPage> with SingleTickerProviderStateMixin
22. {
23. **int** \_selectedIndex = 0;
24. List<Widget> \_bottomNavPages = List();
25. TabController \_tabController;
27. **void** \_onItemTapped(**int** index){
28. setState(() {
29. \_selectedIndex = index;
30. });
31. }
33. @override
34. **void** initState(){
35. **super**.initState();
36. \_tabController = **new**  TabController(vsync: **this**, length: 3);
37. \_bottomNavPages..add(Record())..add(AllViewScreen(controller: \_tabController,))..add(DynamicMain())..add(PersonalPage());
39. //开启完成本周打卡任务的socket监听
40. listen();
42. }
44. listen(){
45. var channel = IOWebSocketChannel.connect("ws://121.196.168.214:8000/ws/message/"+userMobile+"/");
46. channel.sink.add(jsonEncode({"message":"hello world"}));
47. channel.stream.listen((event) async{
48. print(event);
49. // 已完成打卡的名字
50. String title = event;

53. // 展示对话框提示
54. showDialog(
55. context: context,
56. builder: (BuildContext context){
57. **return** AlertDialog(
58. title: Text("系统提示"),
59. content:
60. Column(
61. mainAxisSize: MainAxisSize.min,
62. children: [
63. Text("恭喜你，完成了本周"+title+"打卡任务!"),
64. CircleBadge(color:Theme.of(context).primaryColor, title:title, subtitle:"继续努力!")
65. ],
66. ),
67. actions: [
68. FlatButton(child:Text("确定"), onPressed: ()=>{Navigator.of(context).pop()},)
69. ],
70. );
71. }
72. );
74. });
75. }
77. @override
78. **void** dispose(){
79. \_tabController.dispose();
80. **super**.dispose();
81. }
83. @override
84. Widget build(BuildContext context){
85. **return** Scaffold(
86. appBar: AppBar(
87. leading: IconButton(icon: Icon(Icons.help\_outline),onPressed: (){
88. Navigator.push(context, MaterialPageRoute(builder: (context)=>**new** AboutDayPage()));
89. },),
90. actions: \_selectedIndex==2?
91. [
92. IconButton(icon: Icon(Icons.add), onPressed: (){
93. Navigator.push(context, MaterialPageRoute(builder: (context)=>**new** PublishPage()));
94. },)
95. ]:
96. [
97. ]
98. ,
99. //title:Text("轻记"),
100. bottom: \_selectedIndex==1? **new** TabBar(isScrollable: **true**,
101. unselectedLabelColor: Colors.black87,
102. labelColor: Colors.white,
103. indicatorSize: TabBarIndicatorSize.tab,
104. indicator: **new** BubbleTabIndicator(indicatorHeight:25.0, indicatorColor:Theme.of(context).primaryColor, tabBarIndicatorSize: TabBarIndicatorSize.tab),
105. tabs: [**new** Tab(text:"周视图"), **new** Tab(text:"月视图"), **new** Tab(text:"年视图")],
106. controller: \_tabController,
107. ):**null**,
108. ),
109. body: \_bottomNavPages[\_selectedIndex],
110. bottomNavigationBar: BottomNavigationBar(
111. items: <BottomNavigationBarItem>[
112. BottomNavigationBarItem(icon: Icon(Icons.home), label:"主页"),
113. BottomNavigationBarItem(icon: Icon(Icons.list), label:"统计"),
114. BottomNavigationBarItem(icon: Icon(Icons.camera),label:"发现"),
115. BottomNavigationBarItem(icon: Icon(Icons.person),label: "我的")
116. ],
117. currentIndex: \_selectedIndex,
118. selectedItemColor: Theme.of(context).primaryColor,
119. unselectedItemColor: Theme.of(context).unselectedWidgetColor,
120. selectedLabelStyle: TextStyle(),
121. onTap: \_onItemTapped,
123. ),
124. );
125. }
127. }

# 4服务器端代码

## 4.1模块分解说明

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 文件名称 | 所属模块 | 功能 |
| 1 | admin | Clock | 管理员网页设置 |
| 2 | apps | Clock | App基类 |
| 3 | consumers | Clock | 网页集成 |
| 4 | models | Clock | 创建模型 |
| 5 | routing | Clock | 配置路由 |
| 6 | tasks | Clock | 检查用户是否完成本周的打卡任务 |
| 7 | tests | Clock | 测试类 |
| 8 | urls | Clock | 路由地址 |
| 9 | views | Clock | 交互接口：创建新的打卡、加载打卡内容、完成打卡、一周内打卡查询、周视图、月视图、年试图、修改打卡主题色、修改打卡结束时间、删除打卡、打卡结果实时统计、我的成就 |
| 10 | admin | Dynamic | 管理员网页设置 |
| 11 | apps | Dynamic | App基类 |
| 12 | models | Dynamic | 创建模型 |
| 13 | tests | Dynamic | 测试类 |
| 14 | urls | Dynamic | 路由地址 |
| 15 | views | Dynamic | 交互接口：发布新动态、加载动态、删除发布的动态、添加到收藏夹、从收藏夹移除、加载用户的收藏夹、加载某个收藏夹中的动态 |
| 16 | asgi | Global | 部署配置 |
| 17 | celery | Global | 集成配置 |
| 18 | routing | Global | 路由信息 |
| 19 | settings | Global | 配置 |
| 20 | urls | Global | 路由配置 |
| 21 | wsgi | Global | 部署配置 |
| 22 | admin | Meal | 管理员网页设置 |
| 23 | apps | Meal | App基类 |
| 24 | models | Meal | 创建模型 |
| 25 | tests | Meal | 测试类 |
| 26 | urls | Meal | 路由地址 |
| 27 | views | Meal | 交互接口：上传一餐图片、加载一餐图片、最近十天饮食图片、 |
| 28 | dynamic | Static | 存放动态图片 |
| 29 | meal | Static | 存放三餐图片 |
| 30 | profile | Static | 存放头像图片 |
| 31 | admin | User | 管理员网页设置 |
| 32 | apps | User | App基类 |
| 33 | models | User | 创建模型 |
| 34 | tests | User | 测试类 |
| 35 | urls | User | 路由地址 |
| 36 | views | User | 交互接口：用户登录、新用户注册、加载用户的个人信息、修改用户的个人信息、加载用户头像 |
| 37 | manage |  | 运行后台管理程序 |
| 38 | utils |  | 功能性函数 |

## 4.2主要源代码清单

### 4.2.1打卡模块Clock

1. **import** json
2. **from** datetime **import** datetime as dt
3. **import** time as t
4. **from** django.http **import** HttpResponse, JsonResponse
5. **from** django.shortcuts **import** render
6. **from** .models **import** ClockInItem, ClockRecord
7. **from** User.models **import** User
8. **from** utils **import** mySerializer, random\_str, now, day\_now, login\_required, get\_current\_week, get\_year\_start, \
9. get\_year\_end, get\_current\_month, get\_week\_num
10. **from** .tasks **import** \*

13. # Create your views here.
14. @login\_required()
15. **def** create(request):
16. """
17. 用户创建新的打卡
18. :param request: 客户端请求
19. :return: {'status':状态, 'type':错误/成功类型}
20. """
21. mobile = request.GET["mobile"]
22. title = request.GET["title"]
23. start\_date = request.GET["startDate"]
24. end\_date = request.GET["endDate"]
25. color = request.GET["color"]
26. icon\_theme = request.GET["icon"]
27. freq = request.GET["freq"]
29. # 先查用户是否有重名的
30. user = User.objects.get(mobile=mobile)
31. sets = ClockInItem.objects.filter(user=user, title=title)
32. **if** sets:
33. **return** HttpResponse(json.dumps({'status': 'error', 'type': 'duplicate'}))
35. # 进行用户已经设定的长度判断
36. sets = ClockInItem.objects.filter(user=user, status=True)
37. **if** sets.count() >= 4:
38. **return** HttpResponse(json.dumps({'status': 'error', 'type': 'over length'}))
40. **try**:
41. # 执行逻辑
42. new\_item = ClockInItem(user=user, title=title, start\_date=start\_date, end\_date=end\_date, frequency=freq,
43. icon\_theme=icon\_theme, color\_theme=color, cTime=now(),
44. \_id=str(int(t.time())) + random\_str(12))
45. new\_item.save()
46. **return** HttpResponse(json.dumps({'status': 'ok', 'type': new\_item.id}))
47. **except** Exception as e:
48. **return** HttpResponse(json.dumps({'status': 'error', 'type': str(e)}))

51. @login\_required()
52. **def** load(request):
53. """
54. 加载用户待打卡内容
55. :param request:
56. :return:
57. """
58. mobile = request.GET["mobile"]
59. user = User.objects.get(mobile=mobile)
60. **print**(day\_now())
61. sets = ClockInItem.objects.filter(user=user, status=True, end\_date\_\_gte=day\_now(), start\_date\_\_lte=day\_now())
63. # 对数据打tag -> 今日是否已打卡
64. res\_list = []
65. **for** item **in** sets:
66. res\_list.append({
67. 'id': item.id,
68. 'title': item.title,
69. 'startDate': str(item.start\_date),
70. 'endDate': str(item.end\_date),
71. 'freq': item.frequency,
72. 'icon': item.icon\_theme,
73. 'color': item.color\_theme,
74. 'cTime': str(item.cTime),
75. # 今日是否已经打卡
76. 'hasTodayTag': ClockRecord.objects.filter(user=user, bind\_item=item, timestamp=day\_now()).count(),
77. # 是否完成该任务本周的打卡计划
78. 'hasWeekTaskFinish': check\_succ(mobile, item.title)
79. })
80. **print**(res\_list)
81. **return** HttpResponse(json.dumps(res\_list), content\_type='application/json')

84. @login\_required()
85. **def** record(request):
86. """
87. 用户完成一项打卡
88. :param request:
89. :return:
90. """
91. mobile = request.GET["mobile"]
92. user = User.objects.get(mobile=mobile)
93. detail = request.GET["detail"]
94. clock\_item = ClockInItem.objects.filter(title=request.GET['title'], user=user)
95. # 检查是否存在或者过期
96. **if** **not** clock\_item.count():
97. **return** HttpResponse(json.dumps({'status': 'error', 'type': 'not exist'}))
98. **else**:
99. clock\_item = clock\_item[0]
100. **if** **not** clock\_item.status:
101. **return** HttpResponse(json.dumps({'status': 'error', 'type': 'expired'}))
102. # 检查是否同一天多次打卡
103. **if** ClockRecord.objects.filter(user=user, bind\_item=clock\_item, timestamp=day\_now()).count():
104. **return** HttpResponse(json.dumps({'status': 'error', 'type': 'duplicate '}))
106. **try**:
107. new\_record = ClockRecord(user=user, bind\_item=clock\_item, timestamp=day\_now(), detail=detail)
108. new\_record.save()
109. check\_succ(mobile, request.GET['title'], 1)
110. **return** HttpResponse(json.dumps({'status': 'ok', 'type': 'create success'}))
111. **except** Exception as e:
112. **return** HttpResponse(json.dumps({'status': 'error', 'type': str(e)}))

115. **def** start\_end\_records(mobile, start\_day, end\_day):
116. """
117. 功能函数，查询任意两天间的打卡记录.MonthView和WeekView函数中被调用
118. :return:
119. """
120. result\_list = []
122. user = User.objects.get(mobile=mobile)
123. records = ClockRecord.objects.filter(user=user, timestamp\_\_lte=end\_day, timestamp\_\_gte=start\_day)
125. **for** rec **in** records:
126. result\_list.append({
127. "date": str(rec.timestamp),
128. "title": rec.bind\_item.title,
129. "icon\_theme": rec.bind\_item.icon\_theme,
130. "color\_theme": rec.bind\_item.color\_theme,
131. "detail": rec.detail,
132. })
133. **return** result\_list

136. @login\_required()
137. **def** week\_view(request):
138. """
139. 周视图
140. :param request:
141. :return:
142. """
143. mobile = request.GET["mobile"]
145. start\_day, end\_day = get\_current\_week()
146. result\_list = start\_end\_records(mobile, start\_day, end\_day)
148. **return** HttpResponse(json.dumps(result\_list))

151. @login\_required()
152. **def** month\_view(request):
153. """
154. 统计月视图，格式与周视图一致
155. :param request:
156. :return:
157. """
158. mobile = request.GET["mobile"]
160. start\_day, end\_day = get\_current\_month()
161. **print**(start\_day, end\_day)
162. result\_list = start\_end\_records(mobile, start\_day, end\_day)
163. **return** HttpResponse(json.dumps(result\_list))

166. @login\_required()
167. **def** year\_view(request):
168. """
169. 统计该用户年度的各个打卡记录
170. :param request:
171. :return:
172. """
173. year = dt.now().year
174. start, end = str(year) + "-01-01", str(year) + "-12-31"
176. user = User.objects.get(mobile=request.GET["mobile"])
177. records = ClockRecord.objects.filter(user=user, timestamp\_\_lte=end, timestamp\_\_gte=start)
179. # 返回的数据格式
180. # {
181. #   "跑步": [ "2020-11-07", "2020-09-30", ],
182. #   ...
183. # }
184. mps, title2color = {}, {}
186. **for** rec **in** records:
187. **if** rec.bind\_item.title **not** **in** mps.keys():
188. mps[rec.bind\_item.title] = str(rec.timestamp)
189. title2color[rec.bind\_item.title] = rec.bind\_item.color\_theme
190. **else**:
191. mps[rec.bind\_item.title] += "\r\n" + str(rec.timestamp)
192. res = []
193. **for** (key, val) **in** mps.items():
194. res.append({"title": key, "times": val, "color": title2color[key]})
196. **return** HttpResponse(json.dumps(res), content\_type="application/json")

199. **def** valid\_check(mobile, title):
200. user = User.objects.get(mobile=mobile)
201. clock = ClockInItem.objects.filter(user=user, title=title)
202. **if** clock.count() == 0 **or** clock.count() > 1:
203. **return** {'error': 'length'}
204. clock = clock[0]
205. **if** **not** clock.status:
206. **return** {'error': 'status'}
207. **return** {'obj': clock}


211. @login\_required()
212. **def** set\_color(request):
213. """
214. 修改打卡的主题色
215. :param request:
216. :return:
217. """
218. res = valid\_check(request.GET["mobile"], request.GET["title"])
219. **if** res **and** 'error' **in** res.keys():
220. **return** HttpResponse(json.dumps({'status': 'error', 'type': res["error"]}))
221. user = User.objects.get(mobile=request.GET["mobile"])
222. **try**:
223. ClockInItem.objects.filter(user=user, title=request.GET["title"]).update(color\_theme=request.GET["color"])
224. **return** HttpResponse(json.dumps({'status': 'ok', 'type': 'update success'}))
225. **except** Exception as e:
226. **return** HttpResponse(json.dumps({"status": 'error', 'type': str(e)}))

229. @login\_required()
230. **def** set\_end\_date(request):
231. """
232. 修改打卡结束日期
233. :param request:
234. :return:
235. """
236. res = valid\_check(request.GET["mobile"], request.GET["title"])
237. **if** res **and** 'error' **in** res.keys():
238. **return** HttpResponse(json.dumps({'status': 'error', 'type': res["error"]}))
239. user = User.objects.get(mobile=request.GET["mobile"])
240. **try**:
241. ClockInItem.objects.filter(user=user, title=request.GET["title"]).update(end\_date=request.GET["end\_date"])
242. **return** HttpResponse(json.dumps({'status': 'ok', 'type': 'update success'}))
243. **except** Exception as e:
244. **return** HttpResponse(json.dumps({"status": 'error', 'type': str(e)}))

247. @login\_required()
248. **def** delete(request):
249. """
250. 删除打卡
251. :param request:
252. :return:
253. """
254. user = User.objects.get(mobile=request.GET["mobile"])
255. res = valid\_check(request.GET["mobile"], request.GET["title"])
256. **if** res **and** 'error' **in** res.keys():
257. **return** HttpResponse(json.dumps({'status': 'error', 'type': res["error"]}))
259. user = User.objects.get(mobile=request.GET["mobile"])
260. **try**:
261. ClockInItem.objects.filter(user=user, title=request.GET["title"]).delete()
262. **return** HttpResponse(json.dumps({'status': 'ok', 'type': 'delete succ'}))
263. **except** Exception as e:
264. **return** HttpResponse(json.dumps({"status": 'error', 'type': str(e)}))

267. @login\_required()
268. **def** analysis(request):
269. """
270. 对打卡结果实时统计
271. :param request:
272. :return:
273. """
274. user = User.objects.get(mobile=request.GET["mobile"])
275. task = ClockInItem.objects.filter(user=user, title=request.GET["title"])[0]
277. # 共打卡多少次
278. nums = ClockRecord.objects.filter(user=user, bind\_item=task).count()
280. # 到现在已经执行了几周
281. weeks = get\_week\_num(str(task.start\_date), str(dt.now())[:10])+1
282. # 有几周是完成任务的
283. finish\_nums = WeekRecord.objects.filter(user=user, bind\_item=task).count()
285. **print**(nums, weeks, finish\_nums)
286. res = str(nums)+"\r\n"+str(weeks)+"\r\n"+str(finish\_nums)
287. **return** HttpResponse(res)

290. @login\_required()
291. **def** achieve(request):
292. """
293. 我的成就
294. :param request:
295. :return:
296. """
297. user = User.objects.get(mobile=request.GET["mobile"])
298. records = WeekRecord.objects.filter(user=user).order\_by('-timestamp')
299. res\_list = []
300. **for** rec **in** records:
301. res\_list.append(
302. {
303. "task\_title": rec.bind\_item.title,
304. "task\_icon": rec.bind\_item.icon\_theme,
305. "task\_color": rec.bind\_item.color\_theme,
306. "timestamp": rec.timestamp,
307. "week\_num": str(rec.week\_num)
308. }
309. )
310. **return** JsonResponse(res\_list, safe=False)

### 4.2.2动态模块Dynamic

1. **import** base64
2. **import** json
3. **import** os
5. **from** django.http **import** HttpResponse
6. **from** django.views.decorators.csrf **import** csrf\_exempt
7. **from** .models **import** \*
8. **from** utils **import** \*

11. # Create your views here.
12. @csrf\_exempt
13. **def** publish(request):
14. """
15. 用户发布新的动态
16. :param request:
17. :return:
18. """
19. mobile = request.POST["mobile"]
20. user = User.objects.get(mobile=mobile)
21. txt\_content = request.POST["txt\_content"]
22. image\_len = request.POST["image\_len"]
23. type\_ = request.POST["type"]
25. len\_ = eval(image\_len)
26. **print**(mobile, txt\_content, len\_, type\_)
27. paths = ""
28. **for** i **in** range(len\_):
29. **try**:
30. content\_key = "image\_content" + str(i)
31. image\_content = request.POST[content\_key]
32. **print**(image\_content)
33. """
34. folder = "./static/dynamic/" + str(mobile) + "/" + day\_now()+"/"
35. print(folder)
36. if not os.path.exists(folder):
37. os.makedirs(folder)
38. name\_key = "image\_name" + str(i)
39. name = request.POST[name\_key]
40. with open(folder+name, 'wb+') as f:
41. f.write(base64.b64decode(image\_content))
42. """
43. folder = "./static/dynamic/"
44. # if not os.path.exists(folder):
45. #     os.makedirs(folder)
46. name\_key = "image\_name" + str(i)
47. # name = request.POST[name\_key]
48. img\_title = str(mobile)+str(random.randint(10000, 99999))
49. img\_name = img\_title + ".jpg"
50. paths += img\_title
51. with open(folder + img\_name, 'wb+') as f:
52. f.write(base64.b64decode(image\_content))
53. paths += ","
54. **except** Exception as e:
55. **print**(e)
56. **return** HttpResponse(json.dumps({'status': 'error', 'type': str(e)}))
58. **print**(paths)
59. dynamic = UserDynamic(user=user, timestamp=now(), id=mobile + str(datetime.now()),
60. txt\_content=txt\_content, image\_list=paths, type=type\_
61. )
62. dynamic.save()
63. **return** HttpResponse(json.dumps({'status': 'ok', 'type': 'succ'}))

66. **def** showpic(request, name):
67. **print**(name)
68. with open("./static/dynamic/"+str(name)+".jpg", 'rb') as f:
69. profile\_data = f.read()
70. **return** HttpResponse(profile\_data, content\_type='image/jpg')

73. **def** load(request, type\_):
74. """
75. 加载动态， 根据type\_决定是查看动态的时候加载 还是 加载自己的发布动态
76. :param type\_:
77. :param request:
78. :return:
79. """
80. **def** searize(query\_set):
81. res\_list = []
82. **for** query **in** query\_set:
83. res\_list.append(
84. {
85. "user\_avatar": query.user.profile,
86. "user\_nickname": query.user.nickname,
87. "user\_mobile": query.user.mobile,
88. "id": query.id,
89. "timestamp": query.timestamp,
90. "txt\_content": query.txt\_content,
91. "image\_list": query.image\_list,
92. "type": query.type,
93. }
94. )
95. **return** res\_list
97. **if** type\_ == 1:
98. start\_pos = eval(request.GET["start"])
99. end\_pos = start\_pos+3
100. query\_set = UserDynamic.objects.order\_by('-timestamp')[start\_pos:end\_pos]
101. **print**(query\_set)
102. **elif** type\_ == 2:
103. user = User.objects.get(mobile=request.GET["mobile"])
104. query\_set = UserDynamic.objects.filter(user=user).order\_by('-timestamp')
106. **return** JsonResponse(searize(query\_set), safe=False)

109. @login\_required()
110. **def** delete(request):
111. """
112. 用户删除已经发布的动态
113. :param request:
114. :return:
115. """
116. mobile = request.GET["mobile"]
117. dynamic\_id = request.GET["dynamic\_id"]
118. **try**:
119. user = User.objects.get(mobile=mobile)
120. UserDynamic.objects.filter(user=user, id=dynamic\_id).delete()
121. **return** JsonResponse({'status': 'ok', 'type': 'del succ'})
122. **except** Exception as e:
123. **return** JsonResponse({'status': 'error', 'type': str(e)})

126. @login\_required()
127. **def** collect(request):
128. """
129. 添加到收藏夹
130. :param request:
131. :return:
132. """
133. mobile = request.GET["mobile"]
134. collection\_name = request.GET["collection\_name"]
135. # is\_new = eval(request.GET["is\_new"])
136. bind\_dynamic = request.GET["bind\_dynamic"]
138. user = User.objects.get(mobile=mobile)
139. dynamic = UserDynamic.objects.get(id=bind\_dynamic)
141. is\_exist = Collections.objects.filter(user=user, collection\_name=collection\_name)
142. **if** **not** is\_exist.count():
143. id\_ = mobile+"\_"+random\_str(16)
144. new\_collection = Collections(id=id\_, user=user, collection\_name=collection\_name, create\_time=now())
145. new\_collection.save()
147. bind\_collection = Collections.objects.filter(user=user, collection\_name=collection\_name)[0]
148. image = dynamic.image\_list.split(",")[0]
149. **try**:
150. new\_record = UserCollectionRecord(user=user, bind\_collection=bind\_collection, bind\_dynamic=dynamic,
151. preview\_img=image, add\_time=now())
152. new\_record.save()
153. **return** JsonResponse({'status': 'ok', 'type': 'add succ'})
154. **except** Exception as e:
155. **return** JsonResponse({'status': 'error', 'type': str(e)})

158. @login\_required()
159. **def** remove(request):
160. """
161. 从收藏夹移除
162. :param request:
163. :return:
164. """
165. mobile = request.GET["mobile"]
166. collection\_id = request.GET["collection\_id"]
167. bind\_dynamic = request.GET["bind\_dynamic"]
169. **try**:
170. user = User.objects.get(mobile=mobile)
171. dynamic = UserDynamic.objects.get(id=bind\_dynamic)
172. collection = Collections.objects.get(id=collection\_id)
173. UserCollectionRecord.objects.filter(user=user, bind\_collection=collection, bind\_dynamic=dynamic).delete()
174. **return** JsonResponse({'status': 'ok', 'type': 'del succ'})
175. **except** Exception as e:
176. **return** JsonResponse({'status': 'error', 'type': str(e)})

179. @login\_required()
180. **def** get\_collection(request):
181. """
182. 加载用户的收藏夹
183. :param request:
184. :return:
185. """
186. mobile = request.GET["mobile"]
187. user = User.objects.get(mobile=mobile)
189. collections = Collections.objects.filter(user=user)
190. res\_list = []
191. **for** collection **in** collections:
192. records = UserCollectionRecord.objects.filter(user=user, bind\_collection=collection)
193. num = records.count()
194. image = records[0].preview\_img
195. res\_list.append(
196. {
197. "name": collection.collection\_name,
198. "id": collection.id,
199. "num": str(num),
200. "image": image,
201. #"create\_time": collection.create\_time
202. }
203. )
205. **return** JsonResponse(res\_list, safe=False)

208. @login\_required()
209. **def** one\_collection(request):
210. """
211. 加载某个收藏夹中的动态
212. :param request:
213. :return:
214. """
215. mobile = request.GET["mobile"]
216. user = User.objects.get(mobile=mobile)
218. collect\_id = request.GET["collection\_id"]
219. collection = Collections.objects.get(id=collect\_id)
221. query\_set = UserCollectionRecord.objects.filter(user=user, bind\_collection=collection)
222. res\_list = []
223. **for** query **in** query\_set:
224. res\_list.append({
225. "user\_nickname": query.bind\_dynamic.user.nickname,
226. "user\_avatar": query.bind\_dynamic.user.profile,
227. "user\_mobile": query.bind\_dynamic.user.mobile,
228. "id": query.bind\_dynamic.id,
229. "timestamp": query.bind\_dynamic.timestamp,
230. "txt\_content": query.bind\_dynamic.txt\_content,
231. "image\_list": query.bind\_dynamic.image\_list,
232. "type": query.bind\_dynamic.type,
233. })
234. **return** JsonResponse(res\_list, safe=False)

### 4.2.3全局模块Global

1. """
2. Django settings for LoseWeightBackend project.
3. Generated by 'django-admin startproject' using Django 3.1.3.
4. For more information on this file, see
5. https://docs.djangoproject.com/en/3.1/topics/settings/
6. For the full list of settings and their values, see
7. https://docs.djangoproject.com/en/3.1/ref/settings/
8. """
9. **import** os
10. **from** pathlib **import** Path
12. # Build paths inside the project like this: BASE\_DIR / 'subdir'.

15. BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent
17. MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')
18. MEDIA\_URL = '/media/'
19. # Quick-start development settings - unsuitable for production
20. # See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/
22. # SECURITY WARNING: keep the secret key used in production secret!
23. SECRET\_KEY = '2jcxtyk1)mypxu$g935$-7xu++o3$+sdg&q+ni$7n-9mc5v\*(y'
25. # SECURITY WARNING: don't run with debug turned on in production!
26. DEBUG = True
28. ALLOWED\_HOSTS = ['10.66.10.220']

31. # Application definition
33. INSTALLED\_APPS = [
34. 'django.contrib.admin',
35. 'django.contrib.auth',
36. 'django.contrib.contenttypes',
37. 'django.contrib.sessions',
38. 'django.contrib.messages',
39. 'django.contrib.staticfiles',
41. 'channels',
42. 'redis',
43. 'User',
44. 'clock',
45. 'Meal',
46. 'Dynamic'
47. ]

50. MIDDLEWARE = [
51. 'django.middleware.security.SecurityMiddleware',
52. 'django.contrib.sessions.middleware.SessionMiddleware',
53. 'django.middleware.common.CommonMiddleware',
54. 'django.middleware.csrf.CsrfViewMiddleware',
55. 'django.contrib.auth.middleware.AuthenticationMiddleware',
56. 'django.contrib.messages.middleware.MessageMiddleware',
57. 'django.middleware.clickjacking.XFrameOptionsMiddleware',
58. ]
60. ROOT\_URLCONF = 'LoseWeightBackend.urls'
62. TEMPLATES = [
63. {
64. 'BACKEND': 'django.template.backends.django.DjangoTemplates',
65. 'DIRS': [BASE\_DIR / 'templates']
66. ,
67. 'APP\_DIRS': True,
68. 'OPTIONS': {
69. 'context\_processors': [
70. 'django.template.context\_processors.debug',
71. 'django.template.context\_processors.request',
72. 'django.contrib.auth.context\_processors.auth',
73. 'django.contrib.messages.context\_processors.messages',
74. ],
75. },
76. },
77. ]
79. WSGI\_APPLICATION = 'LoseWeightBackend.wsgi.application'
80. ASGI\_APPLICATION = 'LoseWeightBackend.routing.application'
81. SESSION\_ENGINE = 'django.contrib.sessions.backends.cached\_db'        # 引擎
82. # Database
83. # https://docs.djangoproject.com/en/3.1/ref/settings/#databases
85. DATABASES = {
86. 'default': {
87. 'ENGINE': 'django.db.backends.sqlite3',
88. 'NAME': BASE\_DIR / 'db.sqlite3',
89. }
90. }
92. CHANNEL\_LAYERS = {
93. 'default': {
94. 'BACKEND': 'channels\_redis.core.RedisChannelLayer',
95. 'CONFIG': {
96. "hosts": [('127.0.0.1', 6379)],
97. },
98. },
99. }
101. # Password validation
102. # https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators
104. AUTH\_PASSWORD\_VALIDATORS = [
105. {
106. 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',
107. },
108. {
109. 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',
110. },
111. {
112. 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',
113. },
114. {
115. 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',
116. },
117. ]

120. # Internationalization
121. # https://docs.djangoproject.com/en/3.1/topics/i18n/
123. LANGUAGE\_CODE = 'zh-hans'
125. TIME\_ZONE = 'Asia/Shanghai'
127. USE\_I18N = True
129. USE\_L10N = True
131. USE\_TZ = True

134. # Static files (CSS, JavaScript, Images)
135. # https://docs.djangoproject.com/en/3.1/howto/static-files/
137. STATIC\_URL = '/static/'

### 4.2.4三餐模块Meal

1. **import** base64
2. **import** json
3. **import** os
5. **from** django.http **import** HttpResponse
6. **from** django.shortcuts **import** render
7. **from** django.views.decorators.csrf **import** csrf\_exempt
9. **from** Meal.models **import** UserMeal
10. **from** User.models **import** User
11. **import** datetime
12. **from** utils **import** \*

15. # Create your views here.
16. @csrf\_exempt
17. **def** upload(request):
18. """
19. 上传一餐图片
20. :param request:
21. :return:
22. """
23. mobile = request.POST["mobile"]
24. file\_name = request.POST["fileName"]
25. image\_content = request.POST["image"]
26. index = request.POST["index"]
28. user = User.objects.get(mobile=mobile)
29. folder = "./static/meal/" + str(mobile) + "/" + day\_now()
30. **print**(folder)
31. **if** **not** os.path.exists(folder):
32. os.makedirs(folder)
33. path = "/" + index + '.' + file\_name.split(".")[-1]
34. **print**(folder + path)
36. with open(folder + path, 'wb+') as f:
37. f.write(base64.b64decode(image\_content))
39. **try**:
40. **if** index == "0":
41. entry = UserMeal(user=user, day=day\_now(), breakfast=folder + path,)
42. entry.save()
43. **elif** index == "1":
44. UserMeal.objects.filter(user=user, day=day\_now()).update(lunch=folder + path)
45. **elif** index == "2":
46. UserMeal.objects.filter(user=user, day=day\_now()).update(dinner=folder + path)
47. **return** HttpResponse(json.dumps({'status': 'ok', 'type': folder + path}))
48. **except** Exception as e:
49. **return** HttpResponse(json.dumps({'status': 'error', 'type': str(e)}))

52. **def** show(request):
53. """
54. 加载一餐的图片
55. :param request:
56. :return:
57. """
58. path = request.GET["path"]
59. **print**("request:" + path)
60. with open(path, 'rb') as f:
61. profile\_data = f.read()
63. **return** HttpResponse(profile\_data, content\_type='image/jpg')

66. @login\_required()
67. **def** recent(request):
68. """
69. 最近十天饮食图片
70. :param request:
71. :return:
72. """
73. today = datetime.now()
74. day\_begin = today - timedelta(days=10)
76. user = User.objects.get(mobile=request.GET["mobile"])
77. meals = UserMeal.objects.filter(user=user, day\_\_gt=day\_begin)
78. res\_list = []
80. day\_list = []
81. **for** meal **in** meals:
82. day\_list.append(meal.day.strftime("%Y-%m-%d"))
84. default = "./static/meal/non.png"
86. **while** today != day\_begin:
87. today\_str = today.strftime("%Y-%m-%d")
88. **if** today\_str **not** **in** day\_list:
89. res\_list.append({
90. "day": today\_str,
91. "breakfast": default,
92. "lunch": default,
93. "dinner": default,
94. })
95. today -= timedelta(days=1)
97. **for** meal **in** meals:
98. res\_list.append(
99. {
100. "day": meal.day.strftime("%Y-%m-%d"),
101. "breakfast": meal.breakfast,
102. "lunch": meal.lunch,
103. "dinner": meal.dinner,
104. }
105. )
107. res\_list.sort(key=**lambda** meal\_: meal\_["day"], reverse=True)
108. **return** HttpResponse(json.dumps(res\_list))

### 4.2.5用户模块User

1. **import** json
3. **from** django.http **import** HttpResponse
4. **from** utils **import** mySerializer, md5, day\_now, login\_required
5. **from** .models **import** User

8. # Create your views here.
9. **def** login(request):
10. """
11. 用户登陆
12. :param request:
13. :return:
14. """
15. **try**:
16. mobile = request.GET["mobile"]
17. u = User.objects.get(mobile=mobile)
18. **print**(u.password)
19. **print**(request.GET["pwd"])
20. **if** u.password == request.GET["pwd"]:
21. **return** HttpResponse(json.dumps({'status': 'ok', 'type': u.token}))
22. **else**:
23. **return** HttpResponse(json.dumps({'status': 'error', 'type': 'not match'}))
24. **except** Exception as e:
25. **return** HttpResponse(json.dumps({'status': 'error', 'type': str(e)}))

28. **def** register(request):
29. """
30. 新用户注册
31. :param request:
32. :return:
33. """
34. **try**:
35. mobile = request.GET["mobile"]
36. pwd = request.GET["password"]
37. nickname = request.GET["nickname"]
39. user = User.objects.filter(mobile=mobile)
40. **if** user.count():
41. **return** HttpResponse(json.dumps({'status': 'error', 'type': 'duplicate'}))
43. c\_time = day\_now()
44. new\_user = User(mobile=mobile, password=md5(pwd), nickname=nickname, cTime=c\_time)
45. new\_user.save()
46. **return** HttpResponse(json.dumps({'status': 'ok', 'type': new\_user.token}))
47. **except** Exception as e:
48. **return** HttpResponse(json.dumps({'status': 'error', 'type': str(e)}))

51. @login\_required()
52. **def** load(request):
53. """
54. 加载用户的个人信息
55. :param request:
56. :return:
57. """
58. user = User.objects.get(mobile=request.GET["mobile"])
59. result = {
60. "mobile": user.mobile,
61. "nickname": user.nickname,
62. "profile": user.profile,
63. "password": user.password,
64. "cTime": str(user.cTime),
65. "gender": user.gender
66. }
67. **return** HttpResponse(json.dumps(result))

70. @login\_required()
71. **def** change(request):
72. """
73. 修改用户的个人信息
74. :param request:
75. :return:
76. """
77. key = request.GET["key"]
78. val = request.GET["val"]
79. **try**:
80. **if** key == 'nickname':
81. User.objects.filter(mobile=request.GET["mobile"]).update(nickname=val)
82. **elif** key == 'password':
83. User.objects.filter(mobile=request.GET["mobile"]).update(password=md5(val))
84. **elif** key == 'gender':
85. User.objects.filter(mobile=request.GET["mobile"]).update(gender=val)
86. **return** HttpResponse(json.dumps({'status': 'ok', 'type': 'update succ'}))
87. **except** Exception as e:
88. **return** HttpResponse(json.dumps({'status': 'error', 'type': str(e)}))

91. **def** profile(request):
92. """
93. 加载用户头像
94. :param request:
95. :return:
96. """
97. path = request.GET["path"]
98. with open(path, 'rb') as f:
99. profile\_data = f.read()
100. suffix = path.split('.')[-1]
101. **return** HttpResponse(profile\_data, content\_type='image/'+suffix)

# 5注解

Flutter： Google 开源的 UI 工具包，帮助开发者通过一套代码库高效构建多平台精美应用，支持移动、Web、桌面和嵌入式平台。

Django：一个由 Python 编写的一个开放源代码的 Web 应用框架。使用 Django，只要很少的代码，Python 的程序开发人员就可以轻松地完成一个正式网站所需要的大部分内容，并进一步开发出全功能的 Web 服务 Django 本身基于 MVC 模型，即 Model（模型）+ View（视图）+ Controller（控制器）设计模式，MVC 模式使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。

# 附录

[1] Flutter [EB/OL].https://flutterchina.club/ -2020/11/04

[2] Django 简介 [EB/OL]. https://www.runoob.com/django/django-intro -2020/12/05