重庆大学本科学生毕业设计（论文）

个人习惯监测软件开发

学　　生：　张子玄

学　　号：　20164338

指导教师：　杨瑞龙

专　　业：　计算机科学与技术

重庆大学计算机学院

2020 年 6 月

Graduation Design(Thesis) of Chongqing University

Design and Development of Personal Habits Monitoring Software

Undergraduate: Zhang Zixuan

Supervisor: Prof. Yang Ruilong

Major: Computer Science and Technology

College of Computer Science

Chongqing University

June 2020

摘　要

青少年时期是习惯养成的关键期。如何辅助个人养成一个良好的习惯成为了一个重要的问题。本文设计、实现一个个人习惯监测软件，通过该软件，青少年可以简单方便的培养自己的生活习惯、学习习惯、运动习惯以及等等。

本课题的设计与实现主要基于 Flutter、Spring Boot、Docker 三大技术设计、开发与部署。结合市场现状，分析需求、并进行了详细的界面设计与功能实现。

在功能方面，除了核心的监测记录功能外还实现了主题、国际化、社区、商城、数据同步等功能。其中社区功能应用了综合激励模型，用户记录数据可获得奖励，奖励可用于排名与商城使用。

在技术方面本设计有两大创新点：一是设计并实现了一套基于 Flutter 的解耦方案，在 Flutter 应用开发中利用自定义状态管理拆分了业务逻辑与 UI 界面，提升了运行效率、缩短了开发周期、提升了代码的复用率；二是设计并实现了一个简单的基于 Flutter 的 ORM 框架，提高了对移动端数据库 IO 的便捷度。

本文详细描述了个人习惯监测软件的设计与开发过程，包括需求分析、架构设计、系统设计与实现。并通过测试和部署最终稳定高效的运行在安卓与 iOS 设备中。

关键词：习惯监测，Flutter，Spring Boot，Docker，解耦框架

ABSTRACT

Adolescence is the key period to form good habits. How to help individuals develop a good habits has become an important issue. This article designs and implements a personal habits monitoring software. Through this software, teenagers can easily cultivate their own living habits, learning habits, sports habits and so on.

The design and implementation of this project is mainly based on the design, development and deployment of three major technologies: flutter, spring boot and docker. Combined with the current situation of the market, the demand is analyzed, and the detailed interface design and function realization are carried out.

In terms of function, in addition to the core monitoring and recording function, it also realizes the functions of theme, internationalization, community, shopping mall, data synchronization, etc. Among them, the community function applies a comprehensive incentive model, users can get rewards by recording data, and rewards can be used for ranking and shopping.

In terms of technology, this design has two major innovations: one is to design and implement a set of decoupling scheme based on flutter. In the application development of flutter, the user-defined state management is used to split the business logic and UI interface, which improves the operation efficiency, shortens the development cycle, and improves the reuse rate of code generation; the other is to design and implement a simple ORM framework based on flutter, which improved the development efficiency of the mobile database IO.

This article describes the design and development process of personal habits monitoring software in detail, including requirement analysis, architecture design, system design and implementation. Finally, it runs stably and efficiently in Android and iOS devices through testing and deployment.

Key words： Habits Monitoring, Flutter, Spring Boot, Docker, Decoupling Framework

目　录

1 绪论

课题背景的意义以及国内外研究的现状在本章有所介绍，为设计和开发个人习惯监测软件提出做了背景的概述，以及技术选型的简单分析。

1.1 选题背景

在高校中，智能手机虽然使高校学生的生活更加方便，但是也随之也使得部分高校学生群体养成诸多的坏习惯。个人习惯监测软件可对青少年的习惯进行一定程度的干预，促进学生舍弃坏的习惯、养成良好的习惯。

在技术背景方面。工信部公布的数据显示，近年来我国移动互联网接入流量有所提高[2]。Android 系统和 iOS 系统得到消费者和开发商的青睐[1]。在我国手机已迅速占领人们生活的各个领域[3]。

在社会背景方面。缓解大学生的手机成瘾问题，是学校和教育工作者的当务之急[3]。本课题从个人的习惯方面入手，设计并开发一款干预手机使用习惯的软件。其旨在为大学生、青少年群体提供一个养成及监测个人习惯的手机软件，从手机的本身出发，改善手机的成瘾情况。并且，良好的习惯是健康点基础，养成好习惯无疑对于青少年的身体健康、心理健康都有益处。

1.2 研究现状

1.2.1 市场现状

在市场方面而言，在同类型或近似类型的市场方面，存在着一定量优秀出众的同类软件。这类软件或存在着不错的创新，或有着高度的可定制化，在市场上取得了不错的成绩。本课题结合了同类软件的优点，完善了同类软件的不足，最终决定将应用社区功能作为激励核心。

市面上有形形色色的习惯监测软件，现针对以下几款具有代表性的软件进行研究（统计数据截止 2020 年 3 月 10 日）：

① “番茄 To Do”：这款软件是一款基于番茄工作法的习惯监测养成软件。该软件在 iOS 端效率类 App 排名第 7，评分 4.9，评分数 22.2 万。

其主要针对代办事项，依赖于番茄工作法来养成用户习惯，使用计时器来量化每日习惯，使得用户的工作、学习效率更高。这种番茄工作法也存在一定的问题，它的计时方式在如饮食、运动等场景则无法得到应用。

② “小日常”：是一款以日常打卡驱动的习惯养成软件。该软件在 iOS 端效率类 App 排名第 9，评分 4.8，评分数 17.9 万。

软件主要依靠于每日的打卡签到来实现习惯的监督，软件会显示用户连续签

到的天数。但是无法具体记录到工作量、运动量、摄入量等信息。并且过高的自定义还带来了更高的用户使用成本。

③ “达目标”：它用押金提高专注，执行力的手机软件。该软件在 iOS 端效率类 App 排名 89，评分 5.0，评分数 18.4 万。

该软件是通过缴纳押金、社区监督分红方式来监督用户保持习惯。这种方式虽然起到了非常好的监督效果，但是无疑提高了用户的入门门槛。

同类软件对比如下表 1.1 所示。

表 1.1 典型习惯监测软件对比

应用名	核心驱动方式	激励坚持手段
“番茄 To Do”	番茄工作法	工作、学习时长排名，利用攀比心理。
“小日常”	打卡签到	显示坚持的天数，让用户有成就感。
“达目标”	押金监督	失败会扣除押金，迫使用户坚持。

1.2.2 技术现状

对于在技术方面而言。目前，占据中国市场优势的移动端操作系统是 Android 与 iOS[4]。虑到二者都有不少的受众量，故采用跨平台开发方案。跨平台解决方案众多，在利弊的权衡下，本课题最终决定使用 Flutter 作为移动端的开发框架。

根据其原理的不同，目前的跨平台的解决方案主要分为如下表 1.2 所示三类：

表 1.2 目前跨平台技术对比

技术类型	UI 渲染方式	性能	开发效率	框架代表
H5+原生	WebView	一般	高	Cordova、Ionic
JavaScript+原生渲染	原生控件	高	中	RN、Weex
自绘 UI+原生	调用系统 API	高	Flutter 高 QT 低	QT、Flutter

① H5+原生：

这类框架是使用原生的 WebView 来渲染界面，开发者实际开发的是 Web 应用。

Ionic 使用 Web 技术开发跨平台移动应用[7]。

Cordova 包含了一组设备 API，开发者通过使用 JavaScript 语言来调用这些

API，来实现原生功能[6]。

微信小程序基于 Web 技术，与传统的 Web 开发有很多相似之处[10]。

H5+原生混合应用的优点是从事 Web 开发的人员可以快速上手开发，社区及资源丰富，缺点是性能不好。

② JavaScript 开发+原生渲染：

这种解决方案的实现是将布局信息发送给安卓或 iOS 系统，安卓或 iOS 系统渲染原生界面。

Weex 致力于使开发者能基于通用跨平台的 Web 开发语言和开发经验，来构建 Android、iOS 和 Web 应用[8]。

React Native 使用 JavaScript 语言，若熟悉 React 开发则可以以很低的成本开始进行 React Native 的开发。

快应用标准组织由国内众多手机厂商联合发起，如华为、小米、OPPO、vivo 等厂商，实现开发者一次开发，10 家厂商同时上线[9]。

这种实现方式在一些情况下会因为频繁的通信而导致卡顿。

③ 自绘 UI+原生：

这种实现方式，实现一个渲染引擎，这个渲染引擎有着一致的接口。程序只调用自绘引擎来实现界面显示。

QT 使用 C++开发，C++开发需要考虑内存等问题，开发效率太低，逐渐被淘汰出了市场。

1.2.3 选型分析

在软件设计方面，针对同类市场存在的问题。在软件功能方面，本课题重点在于生活、学习、体育锻炼三个方面的监测、记录与展示，用户无需进行繁琐的设置，上手速度快，降低了用户使用成本。在保留用户方面，本课题拟设计了一套基于社区的综合激励模型，激励用户完成每日打卡。

在技术方面，从上一章的比较来看，目前最好的选择是 Flutter。它在 2019 年上半年高频率的发布新版本，由此可以看出 Google 在 Flutter 方面下了较大的精力。Flutter 支持热重载，并且会保存当前状态，这使得开发周期变得更短。

Flutter 保持了 RN 的优点，有着简洁的语法[18]。并且在业务逻辑与 UI 上均提供了高性能且易开发的开发方法[20]。它的底层实现原理及它引入的快速开发周期，使得它成为了一个比原生开发更具竞争力的选择[19]。

因此，最终确定移动端开发技术栈选择使用 Flutter。

1.3 研究意义

青少年时期是人生成长的关键期，也是好习惯养成的关键期。个人习惯监测

软件的发展可以帮助青少年纠正自己不良生活习惯，改善自己日常生活中的一些不良小举动，让个人素养得到极大的提升。通过个人习惯监测软件，青少年可以简单方便的培养自己的生活习惯、学习习惯、运动习惯以及等等。

本课题主要目标是开发一个移动端个人习惯监测软件。在功能方面为青少年养成良好的生活、学习习惯进行辅助。在技术方面，本课题探索、实践了 Flutter 在移动端开发的可行性。

2 相关技术与理论

本章从移动端开发技术、服务端开发技术、数据库技术以及部署技术四个方面介绍、分析了课题项目设计实现的相关技术和理论。

2.1 设计风格

交互界面是应用程序与用户沟通的窗口，不友好的交互就意味着差劲的用户体验。所以交互界面的设计尤为重要，本课题使用 Material Design 风格设计界面。Material Design 可以让用户获得更好的体验[11]。虽然 Flutter 中提供了完整的 Material Design 风格的 Widget 可供开发者使用，但是在本课题中存在着一定量的自绘组件，如图表等。

2.2 移动端开发技术

本课题移动端开发采用了 Flutter 作为开发的框架，本小节概述了 Flutter 框架的基本原理与概念，以及本课题使用到的第三方依赖包。

2.2.1 Flutter 框架

Flutter 由 Google 团队打造，在 Android 与 iOS 系统上运行的 Flutter 程序有着较好的性能。在上一章中，本文已经分析了 Flutter 的优势。Flutter 通过相同的渲染器、框架和同一组 Widget，达到同时构建 Android 和 iOS 应用的目的[5]。

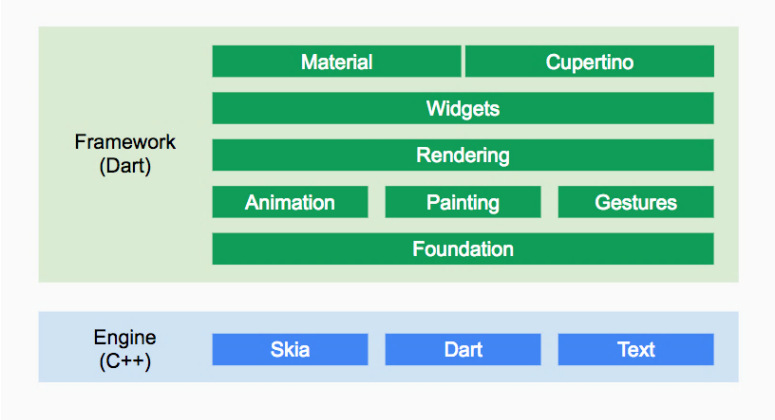

图 2.1 Flutter 架构

Flutter 将做为谷歌正在开发的最新操作系统 Fuchsia 的开发框架[16]。Flutter 可与流行的开发工具集成在一起[17]。Flutter 的架构如上图 2.1 所示。

Flutter 的架构分为 Framework 与 Engine 两部分。

其中 Framework 部分实现了一系列基础的组件。它是一个用 Dart 语言实现的软件开发工具包。

Engine 层实现了不同平台的相同 UI 绘制，是由 C++实现的，效率较高。

本项目使用的 Flutter 版本为 1.12.13+hotfix.8。

2.2.3 Flutter 第三方依赖包

在软件的开发中，我们可以直接集成第三方的模块来辅助我们的开发。以此，便可大大提高开发的效率。Flutter 的仓库是 Pub（https://pub.dev/），它类似于 maven 仓库，开发者可以在 Pub 中查找与发布第三方包与插件。

本项目主要用到了如下表 2.1 的第三方包和插件。

表 2.1 第三方包和插件

名称	作用
sqflite	SQLite 数据库引擎
flutter_local_notifications	原生通知推送
shared_preferences	键值对数据持久化
dio	http 客户端
fl_chart	图表 UI
image_picker	原生相机、相册的图片选择器
image_crop	图片裁剪
flutter_image_compress	图片压缩
provider	状态管理
flutter_markdown	Markdown 预览

2.3 服务端开发技术

本课题服务端开发采用了 Spring Boot 作为开发框架，本小节概述了 Spring Boot 框架的基本原理与概念，以及本课题使用到的其他 Maven 依赖包。

2.3.1 Spring Boot 框架

Spring 是一个开源的 Java EE 框架，其主要作用是程序解耦，增加代码的复用。Spring 通过控制反转与依赖注入，解决了耦合度问题。使得开发人员在开发过程中能够更专注于业务逻辑，从而提高开发效率。

Pivotal 团队提供了一个全新框架 Spring Boot，它对于 Java 的快速开发是有帮助的，它简化了开发过程、配置过程、部署过程和监控过程[15]。

2.3.2 其他依赖

本课题使用到了许多优秀的 Maven 依赖，具体的 Maven 依赖如下表 2.2 所示。

表 2.2 Maven 依赖

名称	作用
spring-boot-starter-mail	发送邮件
mysql-connector-java	MySQL 连接接口
spring-boot-starter-data-mongodb	MongoDB ORM 框架
spring-boot-starter-data-jpa	MySQL ORM 框架
spring-boot-starter-web	Spring Web

2.4 数据库技术

2.4.1 Sqflite 数据库

Sqflite 是 Flutter 的一个第三方依赖插件，它类似 SQLite。目前该插件已支持到了 iOS, Android 和 MacOS 设备（版本 1.3.0+1）。

目前 Sqflite 支持四种类型的 SQLite 数据分别是：INTEGER、REAL、TEXT、BLOB。分别对应了 dart 中的 int、num、String、Unit8List 类型。

虽然插件内置了基础的数据库 IO API，但是不支持实体类映射。关于实体类映射，在第四章第二小节 ORM 框架的设计中将给出解决方案。

2.4.2 MySQL 数据库

MySQL 是一个关系型 DBMS，它高效率、高可靠。本项目采用的 MySQL 版本为 8.0.19。

2.4.3 MongoDB 数据库

MongoDB 介于非关系型数据库与关系型数据库之间。其性能要高于 MySQL，本项目采用的 MongoDB 版本为 3.4.3。

2.5 部署技术

2.5.1 Docker 容器化

Docker 容器化技术属于操作系统层面的虚拟化技术[13]。应用 Docker 可以使得部署变得更加简单高效，本项目使用的 Docker 版本为 19.03.8。

2.5.2 Docker Compose 集群快速编排

Docker Compose 负责实现对 Docker 容器集群的快速编排[13]。它可以快速启动一个容器集群，并可对该集群进行管理。

2.6 小结

本章介绍了本课题涉及到的所有技术如下表 2.3 所示。

表 2.3 相关技术

名称	说明
Dart	面向对象的编程语言（解释型）
Flutter	跨平台开发框架
Java	面向对象编程语言
Spring	Java EE 框架
MySQL	关系型数据库
MongoDB	非关系型数据库
Docker	容器化技术

3 需求分析

本章从技术、经济、时间进行了可行性与需求分析。

3.1 可行性分析

3.1.1 技术可行性

本课题中移动端使用 Flutter 开发，Flutter 目前的生态比较完善，有着非常丰富的插件社区。但是 Flutter 目前还处于起步阶段，开发的过程中必定会遇到很多挑战，不过得益于生态的活跃，大部分插件在 issues 上提出问题或 bug 都能很快的得到作者的回复。

服务端使用 Spring Boot 开发，Spring 有着成熟的社区与生态，并且有着众多优秀的项目可以作为参考。

3.1.2 经济可行性

关于经济可行性，本课题采用了阿里云轻量级服务器作为部署的环境，学生价格有一定的优惠，一个轻量级服务器已经可以满足项目的基本需求，支出约为十元每月。

3.1.3 时间可行性

本课题任务明确，需求清晰，在开发计划的周期内可以如期完成。

3.2 需求分析

个人习惯监测软件可对青少年的习惯进行一定程度的干预，促进学生舍弃坏习惯、养成良好的习惯。

本课题结合了任务书以及同类软件的考察情况，最终确定了以下八个需求，分别为：

① 个人信息、生活状态、学习状态、体育锻炼的记录功能；

② 个人信息、生活状态、学习状态、体育锻炼的展示功能；

③ 监督提醒功能；

④ 数据的同步功能；

⑤ 设置主题、语言、头像等个性化主题功能；

⑥ 社区功能；

⑦ 商城功能;

⑧ 国际化。

根据需求分析可将功能划分为 8 部分，分别为基本信息记录展示模块、日常生活记录展示模块、体育锻炼记录展示模块、课程学习记录展示模块、用户模块、设置模块、商城模块、社区模块。用例图如下图 3.1 所示。

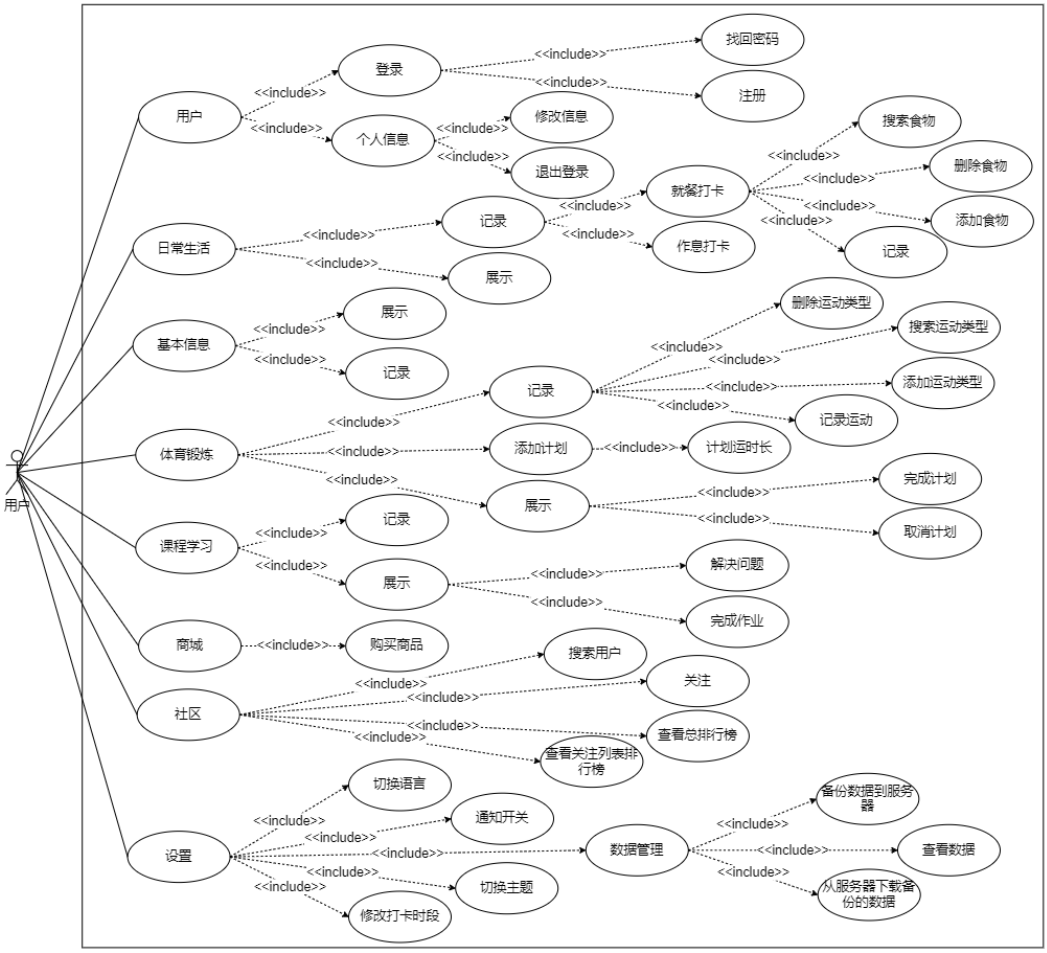

图 3.1 个人习惯监测软件用例图

4 系统设计

本章主要包括了 Flutter 解耦架构的设计、Flutter ORM 框架的设计、数据库设计、接口设计四个方面。

4.1 Flutter 解耦架构的设计

4.1.1 问题分析

Flutter 应用的开发与传统的 Web 应用开发、移动端原生开发都有很大的区别。Google 设计 Flutter 时借鉴了 React 的思想，通过一个 UI 树来渲染界面，diff 算法更新界面。

Flutter 构建 UI 是通过 Widget 类来构建的，如果想要构建动态的界面（界面中的元素可以发生改变）则需要继承 Flutter 提供的 StatefullWidget 类，每一个 StatefullWidget 都有一个所需要的维护的状态，称之为 State。

当所需要的维护的状态被改变时，调用 setState 方法可以达到更新 UI 的目的。

简单来说如果要想改变界面中的元素，则需要进行一下三步操作：

① 在 State 中定义一个变量并初始化；

② 在 build 方法返回的组件中包含该变量，如返回一个 Text(“${你的变量}”)；

③ 修改变量的值，并调用 setState。

这种更新方法虽然简单快捷但是同样也存在着一些问题，其中最明显是以下两点：

① 效率问题：

虽然 Flutter 在调用 build 方法更新界面的时候应用了 diff 算法，但是还是会去遍历当前节点树下的所有子节点，找到需要更新的元素，之后更新数据发生变化的元素。在实际开发中，一个组件可能会有非常多子组件，如果使用 setState 方法来更新组件，每次更新都要访问一遍每个子节点，无疑降低了程序运行效率。

② 编码耦合度问题：

一个界面一般会有很多的变量对应的动态元素，如果将这些变量及其更新的方法都定义在 State 中会使得一个类过于繁重，不易于维护；

4.1.2 解耦方案设计

针对效率问题，Flutter 提供另一种更新组件的方式，就是使用 InheritedWidget。当这个组件中的数据发生改变时，会更新在其节点下的子组件。利用这个特性可精确的更新某个组件的界面显示。但是 Flutter 中想要使用 InheritedWidget 是一件不那么容易的事情，它至少涉及了 4 个类，构建起来十分的复杂。

这种方式有点类似 MVVM 的思想。MVVM 是 Model-View-ViewModel 的简称，设计原理是借鉴 MVC 的思想改造的设计模式，它的最大特点就是开发者不需要直接改变 View，即无须直接操作 DOM[14]。

Provider 是 Flutter 的一个第三方依赖包，它简化了 InheritedWidget 的使用，相当于一个对 InheritedWidget 的包装。Flutter 官方推荐使用 Provider 来进行状态管理，而不是使用 InheritedWidget。开发时就可以通过直接使用 Provider 来进行界面的状态管理。

当使用了 Provider 后，就可以不再使用动态的 StatefullWidget 而是使用静态的 StatelessWidget 来构建界面，将需要改变的元素对应的变量定义在 ChangeNotefier 中，通过 Provider 精确的更新组件。下图 4.1 是引用了 Provider 之后架构原理图。

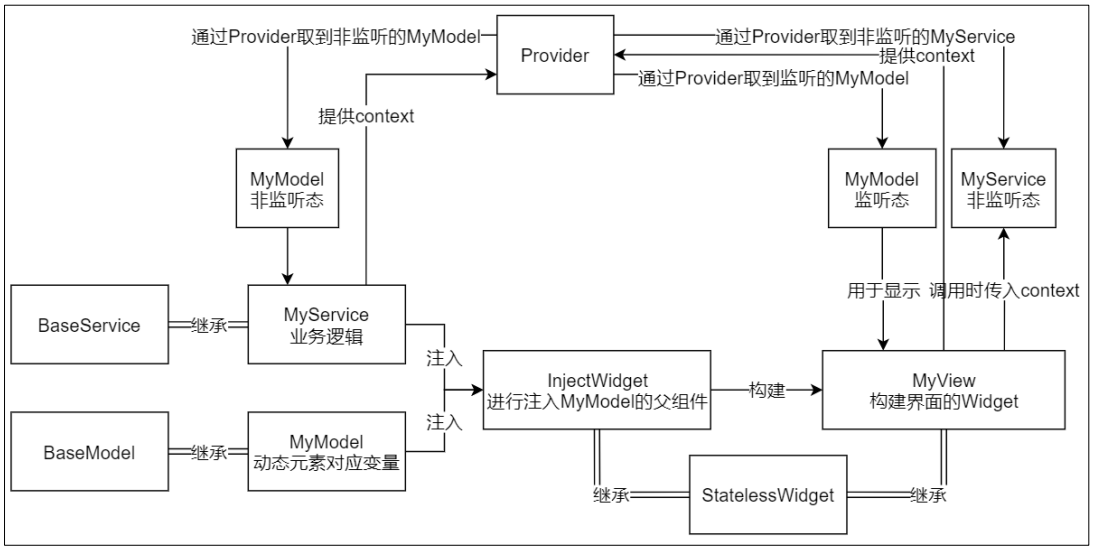

图 4.1 引入 Provider 后架构原理图

要想使用 Provider 需要以下几步：

① 定义 Model 类用来存储界面中需要变化的数据，Model 是继承自 ChangeNotifier 的，如图中的 MyModel。

② 定义 InjectWidget 进行 ChangeNotifier 的注入，InjectWidget 是继承自 StatelessWidget 的，如图中的 InjectWidget。ChangeNotifier 必须在子 Widget 的父 Widget 中注入，才可以在子 Widget 中取得到 ChangeNotifier 对象，这是由于 ChangeNotifier 是注入在 context 之中的，Provider 必须通过 context 才能获得 ChangeNotifier，直接在父 Widget 中获取 ChangeNotifier 是无法获取到的，因为父 Widget 的 context 并没有注入所需要的 ChangeNotifier。因此需要一个注入的节点。

③ 定义 View 类来构建界面显示，View 也是继承自 StatelessWidget 的。在需要用到动态元素的时候则通过 Provider 从 context 中以监听的方式取到第二步注入的 Model，使用 Model 中的变量构建界面。这样当 Model 中的数据发生变化，且收到更新的消息后就会精确的修改界面中的对应元素。

④ 在 View 中编写业务逻辑，在业务逻辑中通过 Provider 从 context 中以非监听的方式取到第二步注入的 Model，根据场景需要修改 Model 中的变量，并在修改后通知 Model 去更新界面。

这样就解决了效率问题以及一部分耦合度的问题。这里之所以说是一部分，是因为业务逻辑部分并没有与界面构建逻辑分开。解耦的最终目的应该是将业务逻辑部分与界面构建逻辑部分完全分开，使得这两部分可以相互独立。

为了拆分业务逻辑与界面构建逻辑，解耦架构可以在上图 4.1 的架构的基础上增加一个继承自 ChangeNotifier 的 Service 类，该 Service 类专门用于处理业务逻辑，类似于 Spring 框架中的 Service 层的功能。这里 Service 与 Model 均是继承自 ChangeNotifier，所以应该对 ChangeNotifier 进行一次封装。最终定义三个基础的类 BasicModel、BasicService 与 BasicDependency，其中 BasicDependency 对开发者是不可感知的，类图如下图 4.2 所示

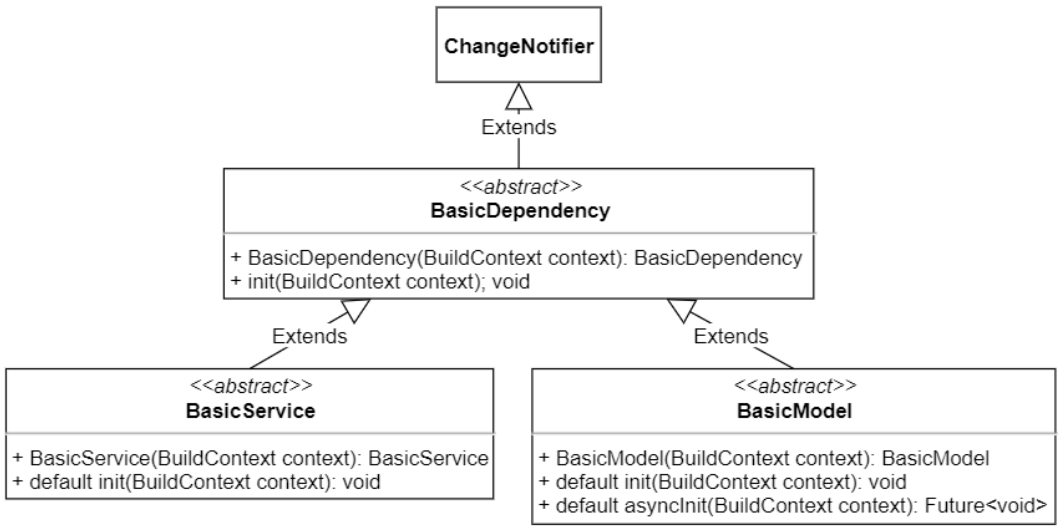

图 4.2 解耦架构中部分类的类图 I

如果应用了这两个基础的抽象类作为依赖，那么就有如下图 4.3 所示的架构原理。

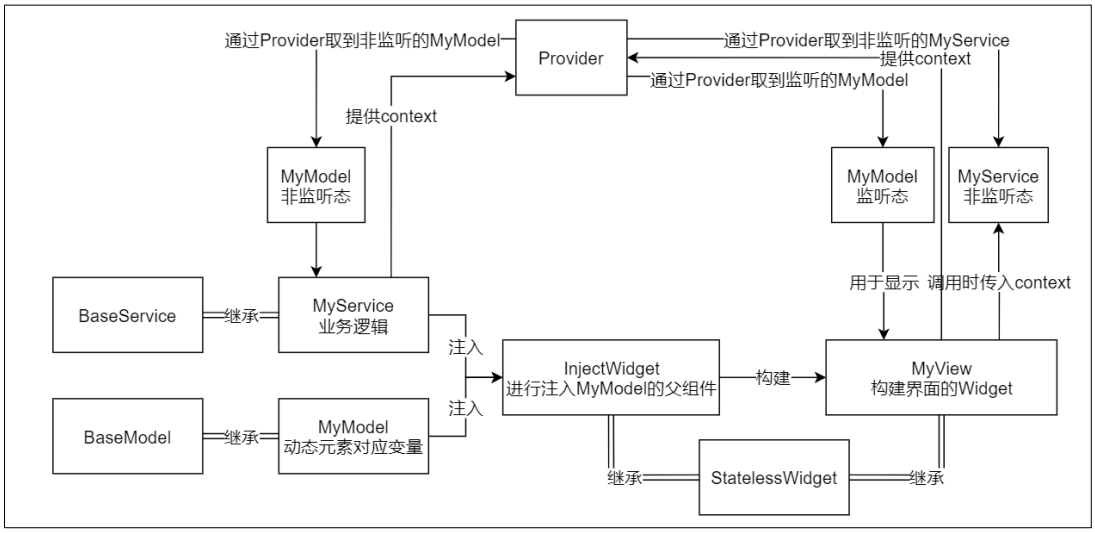

图 4.3 引入基础架构后架构原理图

通过上图可以看出来，这样设计成功的做到了拆分业务逻辑与界面构建逻辑，已经可以应用这个架构来构建强壮的应用了。但是仔细观察可以发现，这样设计使得一个页面至少需要构建 4 个类，分别是 Service、Model、InjectWidget、View。这里的 InjectWidget 只有一个注入的作用，与开发中的逻辑毫无关系。需要找到一种方法省去 InjectWidget 直接在一个类中同时完成依赖注入与界面构建，将原来的 4 个类简化为 Service、Model、View 这 3 个类。

针对上述问题，由于依赖是必须要注入的，所以无论怎样都无法越过 InjectWidget。但是可以从开发的角度下手，使得开发的时候感觉不到中间 InjectWidget 的存在。在这里应用到了 Dart 语言的一个特性，实现了一个 DependencyInjectedWidget 的抽象类，开发过程中使用该抽象类时必须重写其中的 dependencyInjection 方法来进行依赖注入，重写 buildView 方法来进行界面构建。这个类会实例化一个 ViewWidget 类，ViewWidget 开发人员是不可见的。DependencyInjectedWidget 将 buildView 方法传入 ViewWidget 中，ViewWidget 的 build 方法调用传入的 buildView 方法。这样就实现了在一个类中同时实现了依赖注入与界面构建，类图如下图 4.4 所示。

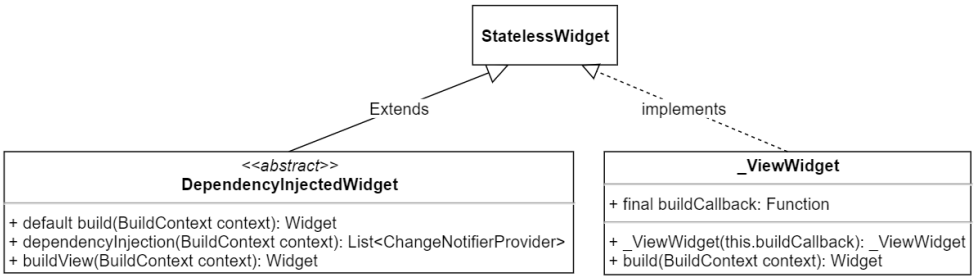

图 4.4 解耦架构中部分类的类图 II

使用上文提到的 DependencyInjectedWidget 后最终架构原理图如下图 4.5 所示。

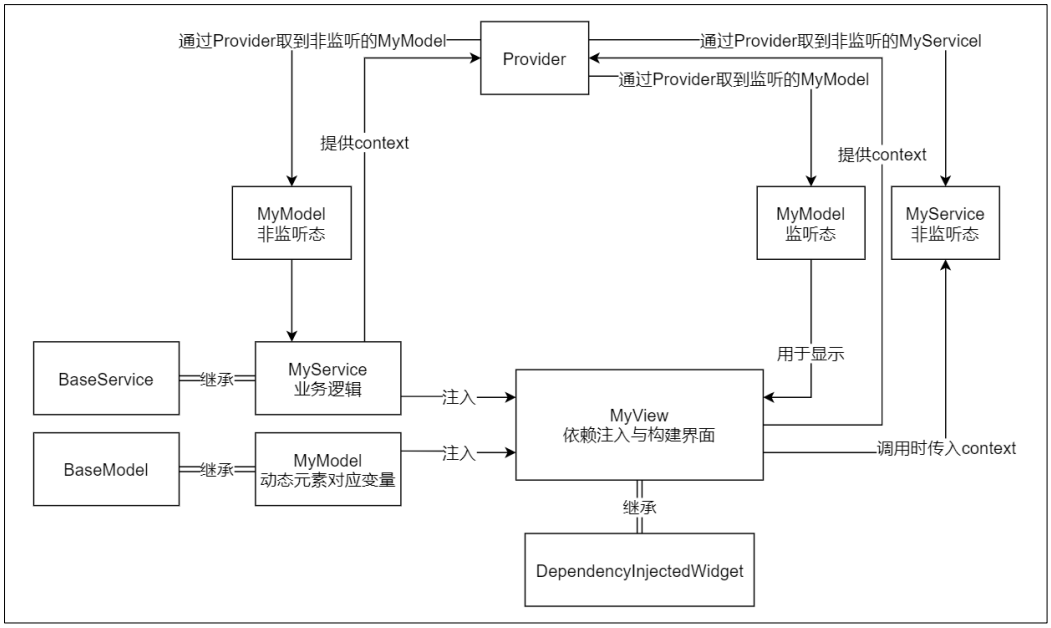

图 4.5 最终架构原理图

通过这种架构，可以说是比较完美的实现了解耦。它拆分了业务逻辑与界面构建逻辑，并且提高了运行的效率。开发过程中只需要关注与页面构建、业务逻辑以及动态数据定义即可，并且在后期维护的过程中，只需要在对应的层进行相应的修改即可。

总结上述内容，本文设计的解耦框架使用方法为以下三步：

① 定义 Model 类存储页面动态信息；

② 定义 Service 类编写业务逻辑；

③ 定义 View 进行依赖注入，构建界面 UI，绑定 Model 中的变量，绑定 Service 中的方法。

通过应用这个解耦框架，可以让 Flutter 项目拥有更低的耦合度，增加了代码的复用性与扩展性，提高了程序的运行效率。

4.2 Flutter ORM 框架的设计

4.2.1 问题分析

Flutter 应用的数据库是 Sqflite。Sqflite 提供的 IO 方法不能将数据映射到实体类上。在代码编写中 Sqflite 自带的 IO 方法传入传出的参数均为 Map 类型。

在开发过程中如果不包装出实体类，则很容易在操作数据库的时候出现参数

的错误，给开发带来了一定的难度，一是没有代码联想功能，读写数据时可能会将参数写错。二是没有编译检查，无法提前发现参数的错误。这些问题降低了开发的效率。

4.2.2 Flutter ORM 框架设计

针对上述问题，社区也存在一些 ORM 的插件，但是这些插件的 GitHub 仓库 Star 都很少，也没有什么知名的企业发布 ORM 插件。如果盲目使用这些插件可能会拖慢开发进度。因此本课题决定自己造轮子，自己写一个简单的 ORM 框架。

首先是关于实体类的设计。在 Flutter 的开发中反射相关的库是不被允许使用的。这就使得在开发时如果想要构建实体类，必须编写对应数据库表与实体类之间的映射函数。即遍历 Map 中的每一个 key-value，一个一个赋值给对应变量名的变量，这个过程无疑是不可能用一个通用的函数来实现。也就是说每一个实体类都需要开发者手动写一个映射函数，这大大降低了开发效率。

出于上述的原因，本课题考虑到将实体类的代码通过代码来生成。设想的需求是：输入表结构的信息，就可以生产对应的实体类的 Dart 代码。最终生成的实体类如下图 4.6 所示。

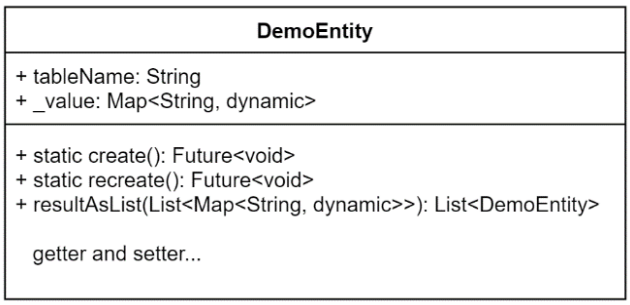

图 4.6 数据库实体类的结构

由于在 Sqflite 数据库的 IO 操作时表名也是一个必要的参数，因此实体类需要有一个 tableName 属性记录该实体类的表名。

通过代码来生成不会出现字母的编写错误，因此可以直接将数据库中查出的 Map 的值作为实体类中的一个私有变量_value。在实体类的 getter 与 setter 方法中，直接对实体类中的_value 进行操作，这样在实体类在生成的时候就减少了一次遍历时间，这种方式可以增加实体类的生成速度。

关于数据库表创建与删除的 SQL 语句也需要通过实体类生成器生成。创建与删除方法作为实体类的静态方法，通过调用这些方法可以创建或删除实体类对应的数据库表。如应用在第一次运行时，需要执行。

在 Sqflite 数据库查询时，查询结果的一行会被返回为 Map<String, dynamic>类型的数据。但是 List<Map<String, dynamic>不能直接转换为 List<实体类>类型，因此需要一个转换函数 resultAsList，用来将 List<Map<String, dynamic>转换为 List<实体类>。

一个 ORM 框架最重要的是其中的映射方法，也就是 Mapper 类。本课题设计了一个通用的 Mapper，支持基础的数据库操作，也支持自定义 SQL 操作。下图 4.7 是 CommonMapper 类的结构图。

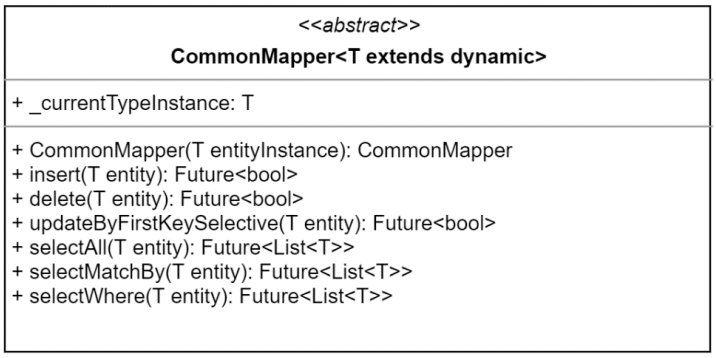

图 4.7 CommonMapper 类的结构

CommonMapper 提供了一组通用的增删改查的方法，定义一个 Mapper 并继承 CommonMapper，实现一个构造函数即可使用 CommonMapper 增删改查功能，推荐将自定义的 Mapper 写为单例模式。下图 4.8 是一个单例 Mapper 的实例。

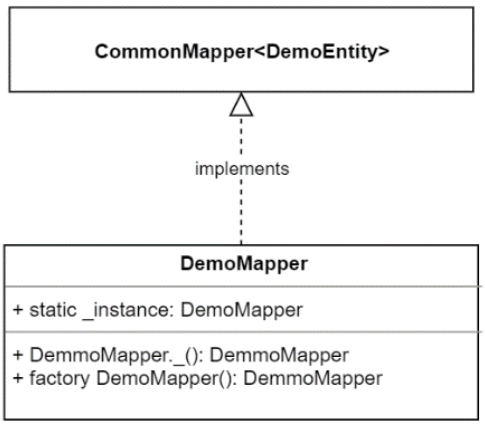

图 4.8 一个自定义 Mapper 的实例

通过上述的封装，可以得到了一个简单 ORM 框架，麻雀虽小，五脏俱全，它目前已经可以满足整个项目的全部使用场景。

4.3 移动端数据库设计

移动端数据库采用了 Sqflite 数据库，主要为个人习惯数据服务。其数据库 E-R 图如下图 4.9 所示。

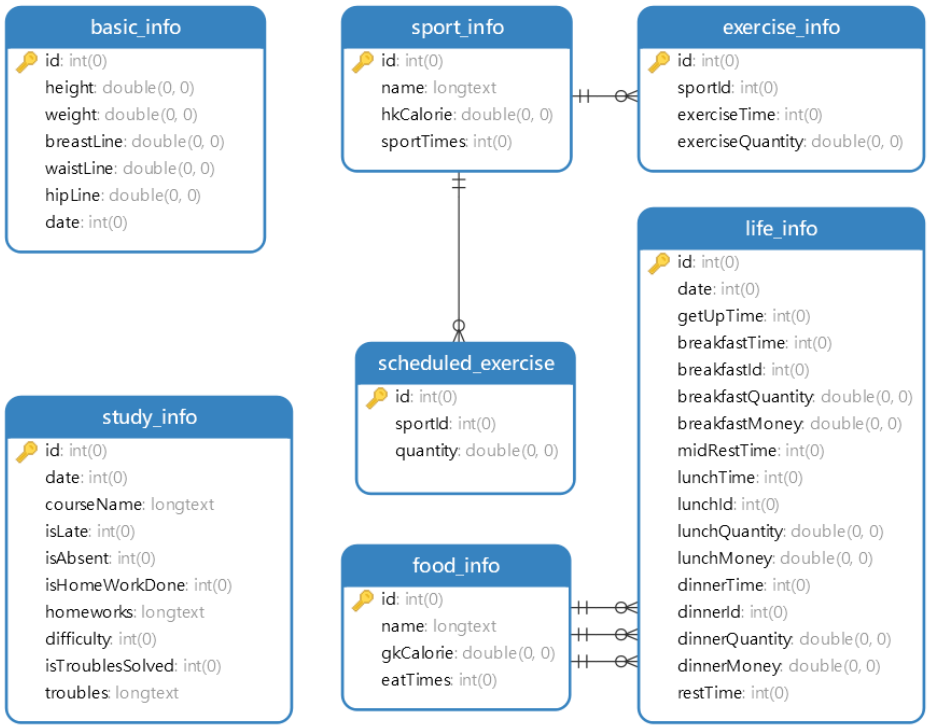

图 4.9 移动端数据库 E-R 图

其中各个表的字段具体说明如下表 4.1 至表 4.7 所示。

表 4.1 基本信息 basic_info 表

字段	数据类型	注释
id	int	自增主键
height	double	身高
weight	double	体重
breastLine	double	胸围
waistLine	double	腰围
hipLine	double	臀围
date	int	记录时间

表 4.2 食物类别 food_info 表

字段	数据类型	注释
id	int	自增主键
name	longtext	食物名称
gkCalorie	double	每百克每千卡路里
eatTimes	int	食用次数

表 4.3 日常生活信息 life_info 表

字段	数据类型	注释
id	int	自增主键
date	int	记录时间
getUpTime	int	起床时间
breakfastTime	int	早餐时间
breakfastId	int	早餐 id（food_info.id）
breakfastQuantity	double	早餐进食量
breakfastMoney	double	早餐花费
midRestTime	int	午休时间
lunchTime	int	午餐时间
lunchId	int	午餐 id（food_info.id）
lunchQuantity	double	午餐进食量
lunchMoney	double	午餐花费
dinnerTime	int	晚餐时间
dinnerId	int	晚餐 id（food_info.id）
dinnerQuantity	double	晚餐进食量
dinnerMoney	double	晚餐花费
restTime	int	晚安时间

表 4.4 运动类别 sport_info 表

字段	数据类型	注释
id	int	自增主键
name	longtext	运动名称
hkCalorie	double	每小时每千卡路里
sportTimes	int	运动次数

表 4.5 体育锻炼信息 exercise_info 表

字段	数据类型	注释
id	int	自增主键
sportId	int	类别 id（sport_info.id）
exerciseTime	int	运动时间
exerciseQuantity	double	运动量

表 4.6 运动计划 scheduled_exercise 表

字段	数据类型	注释
id	int	自增主键
sportId	int	类别 id（sport_info.id）
quantity	double	计划运动量

表 4.7 课程学习信息 study_info 表

字段	数据类型	注释
id	int	自增主键

date	int	记录时间
courseName	longtext	课程名称
isLate	int	是否迟到
isAbsent	int	是否缺席
isHomeWorkDone	int	作业是否完成
homeworks	longtext	作业
difficulty	int	困难度
isTroublesSolved	int	困难是否解决
troubles	longtext	遇到的困难

4.4 服务端数据库设计

服务端数据库主要为账号、社区、商城系统服务，其中 MySQL 数据库的 E-R 图如下图 4.10 所示。

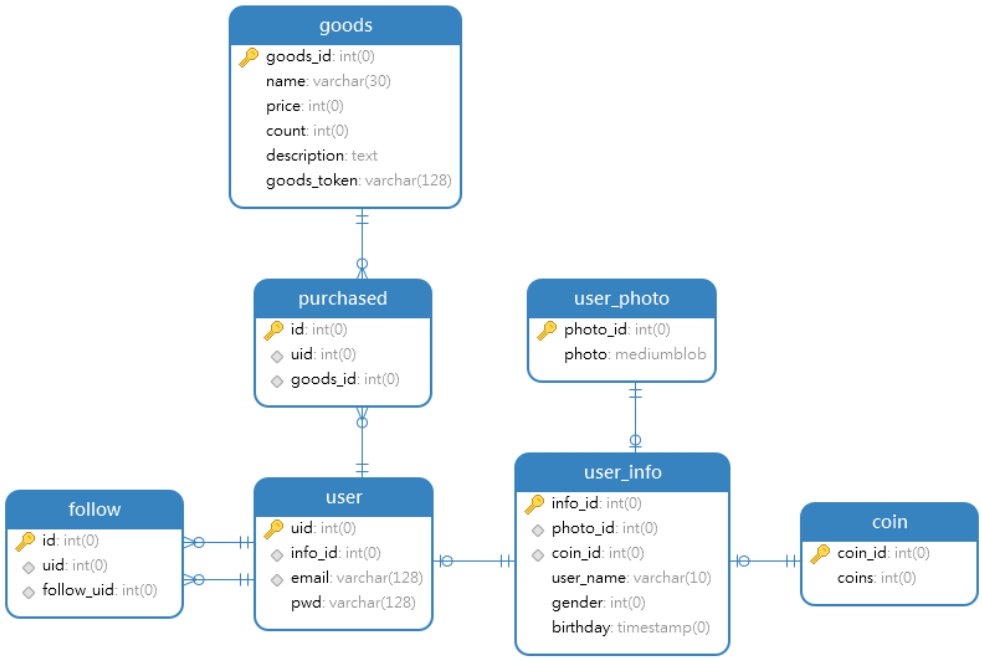

图 4.10 MySQL 数据库 E-R 图

其中各个表的字段具体说明如下表 4.8 至表 4.14 所示。表中简单解释了各个字段的作用。

表 4.8 用户金币 coin 表

字段	数据类型	注释
coin_id	int	自增主键
coins	int	用户金币数

表 4.9 用户头像 user_photo 表

字段	数据类型	注释
photo_id	int	自增主键
photo	mediumblob	用户头像

表 4.10 用户信息 user_info 表

字段	数据类型	注释
info_id	int	自增主键
photo_id	int	头像 id（user_photo. photo_id）
coin_id	int	金币 id（coin. coin_id）
user_name	varchar	用户名
gender	varchar	性别
birthday	varchar	生日

表 4.11 用户账号 user 表

字段	数据类型	注释
uid	int	自增主键
info_id	int	类别 id（user_info. info_id）
email	varchar	邮箱
pwd	varchar	密码

表 4.12 关注 follow 表

字段	数据类型	注释
id	int	自增主键
uid	int	账号 id（user.uid）
follow_uid	int	被关注账号 id（user.uid）

表 4. 13 商品 goods 表

字段	数据类型	注释
goods_id	int	自增主键
name	varchar	商品名称
price	int	商品价格
count	int	商品库存
description	text	商品描述
goods_token	varchar	商品兑换码

表 4.14 购买记录 purchased 表

字段	数据类型	注释
id	int	自增主键
uid	int	账号 id（user.uid）
goods_id	int	商品 id（goods. goods_id）

本项目还使用到了 MongoDB，其主要作用是防作弊、储存登录 token、储存

注册验证码、储存找回密码验证码、储存用户备份资料。MongoDB 的集合设计如下表 4.15 至表 4.18 所示。

表 4.15 令牌 token 集合

字段	数据类型	注释
uid	Integer	用户 uid
token	String	token 串
createDate	Timestamp	创建时间 1 个月时效

表 4.16 验证码 auth_code 集合

字段	数据类型	注释
email	String	用户邮箱
authCode	String	六位验证码
purpose	String	验证码使用目的
createDate	Timestamp	创建时间 5 分钟时效

表 4.17 反作弊 increase_coin_times 集合

字段	数据类型	注释
uid	Integer	用户 uid
times	String	签到次数
createDate	Timestamp	创建时间 12 小时时效

表 4.18 用户数据备份 user_data 集合

字段	数据类型	注释
uid	Integer	用户 uid
data	BSON	用户备份数据
createDate	Timestamp	创建时间

4.5 服务端响应接口设计

服务端采用了 RESTful 风格的接口，其具体接口设计如下表 4.19 所示。

表 4.19 服务端接口设计

接口	方式	注释
/api/user/?nameLike	GET	搜索用户
/api/user/{uid}/info	GET	获取 uid 用户的信息
/api/user/{uid}/photo	GET	获取 uid 用户的头像
/api/user/{uid}/coin	GET	获取 uid 用户的金币
/api/user/{uid}/data	GET	获取 uid 用户的备份数据
/api/user	POST	注册用户
/api/user/{uid}	PUT	修改 uid 用户的密码
/api/user/{uid}/info	PUT	修改 uid 用户的信息
/api/user/{uid}/photo	PUT	修改 uid 用户的头像

/api/user/{uid}/coin	PUT	增加 uid 用户的金币
/api/user/{uid}/data	PUT	备份 uid 用户的数据
/api/permission/token?uid&token	GET	获取登录状态
/api/permission/token	POST	用户登录
/api/permission/authCode	POST	请求发送验证码到邮箱
/api/shopping/goods/	GET	获取所有商品
/api/shopping/goods	POST	新建商品
/api/shopping/goods/{goodsId}	POST	购买商品
/api/shopping/goods/{goodsId}	PUT	修改商品
/api/shopping/goods/{goodsId}	DEL	删除商品
/api/community/top	GET	获取排行榜
/api/community/follow/?uid	GET	获取关注列表
/api/community/follow?uid&followUid	GET	获取关注状态
/api/community/follow	POST	关注用户
/api/community/follow/{uid}/{followUid}	DEL	取消关注用户

4.6 整体架构设计

项目整体由服务端与移动端构成。移动端业务逻辑处理交互响应后修改页面数据更新。服务端接到请求在拦截器进行身份效验，控制器定义响应接口，业务逻辑对数据库进行操作，最终服务端返回响应数据。具体架构设计如下图 4.11 所示。

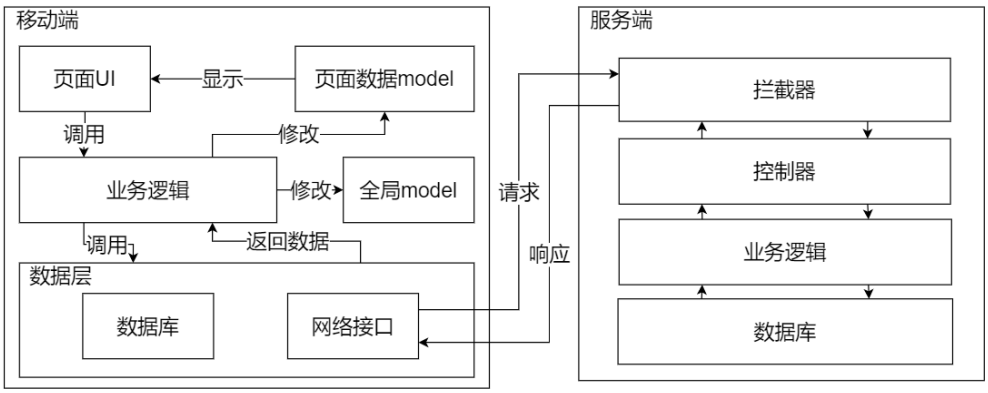

图 4.11 项目整体架构

5 系统实现

本章主要介绍了项目的整体结构以及具体实现。

5.1 项目结构

5.1.1 移动端项目包结构

移动端项目结构如下图 5.1 所示。

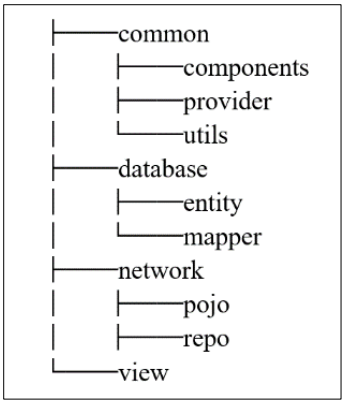

图 5.1 移动端项目结构

其中 common 包是用于存放通用的类，如上一章提到的解耦架构与 ORM 框架、工具类、通用 UI 组件、基础 provider、国际化组件等。database 包用于存放移动端数据库有关的类，如数据库实体类与数据库映射类。network 包用于存放于网络接口相关的类，如 HTTP 客户端、接口信息、pojo 等。view 包用于存放界面相关的类。

5.1.2 服务端项目包结构

服务端项目结构如下图 5.2 所示，这是一个常见的标准的三层架构 Spring 应用的包结构。

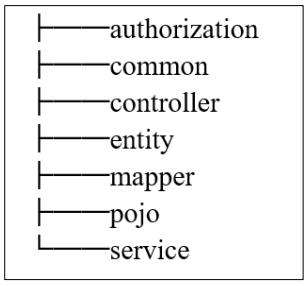

图 5.2 服务端项目结构

其中 authorization 用于存放与权限效验有关的类，如拦截器、拦截器配置类、权限注解。包 common 用于存放通用的类，如响应状态码的封装、工具类等。四个包 controller、service、mapper、entity 分别对应了控制层、业务逻辑层、数据访问层与实体类。包 pojo 用于存放请求响应体。

5.2 功能实现

5.2.1 基本信息展示与记录

基本信息包括了身高、体重、胸围、臀围、腰围的数据。程序实现了三张卡片来展示这些数据，基本信息卡片显示目前最新的基本信息数据。体重信息卡片与三维信息卡片中用户可以调整显示最近 7、30、90 天信息折线图。实现效果图如下图 5.3 所示。

图 5.3 基本信息效果图

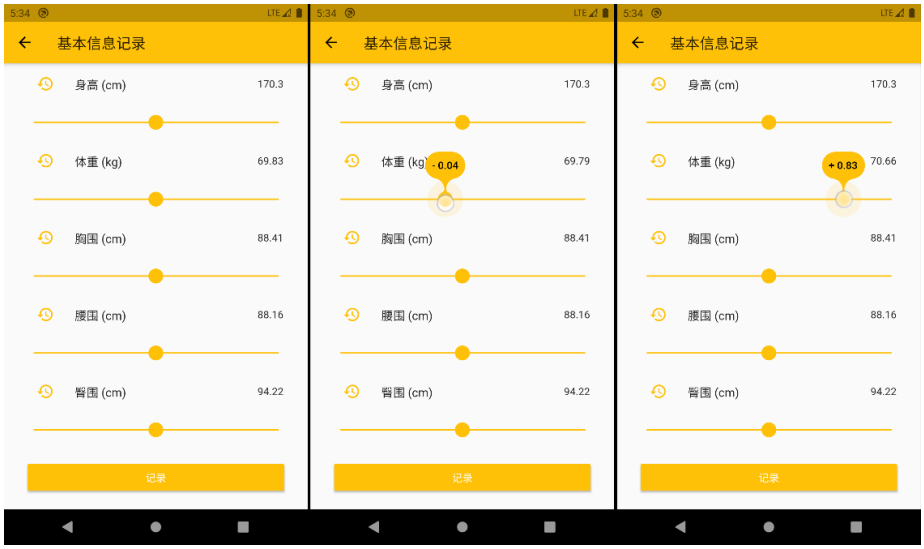

图 5.4 基本信息记录效果图

由于每日的身高、体重、胸围、臀围、腰围数据均只在一个很小的范围波动，因此本课题设计了一个拖动条，用户可在上次记录数据的基础上直接通过拖动调整数据。这个拖动条的拖动距离与变化的值为次方关系，也就是说拖动的越近，数据变动越小，越远则变动越大。通过这种方式避免了输入框的使用，用户可以方便快捷的记录数据变化。具体实现如上图 5.4 所示。

5.2.2 日常生活展示与记录

日常生活信息包括了起床信息、早饭信息、午饭信息、午休信息、晚餐信息、睡眠信息。其中有关就餐信息的记录均包含就餐时间、就餐内容、进食量、消费金额。程序实现了今日信息、每日睡眠时长、每日摄入卡路里、每日花费金额、每日睡眠时间点、每日吃饭时间点以及每日打卡完成度七张卡片的数据展示功能。实现效果如下图 5.5 所示。

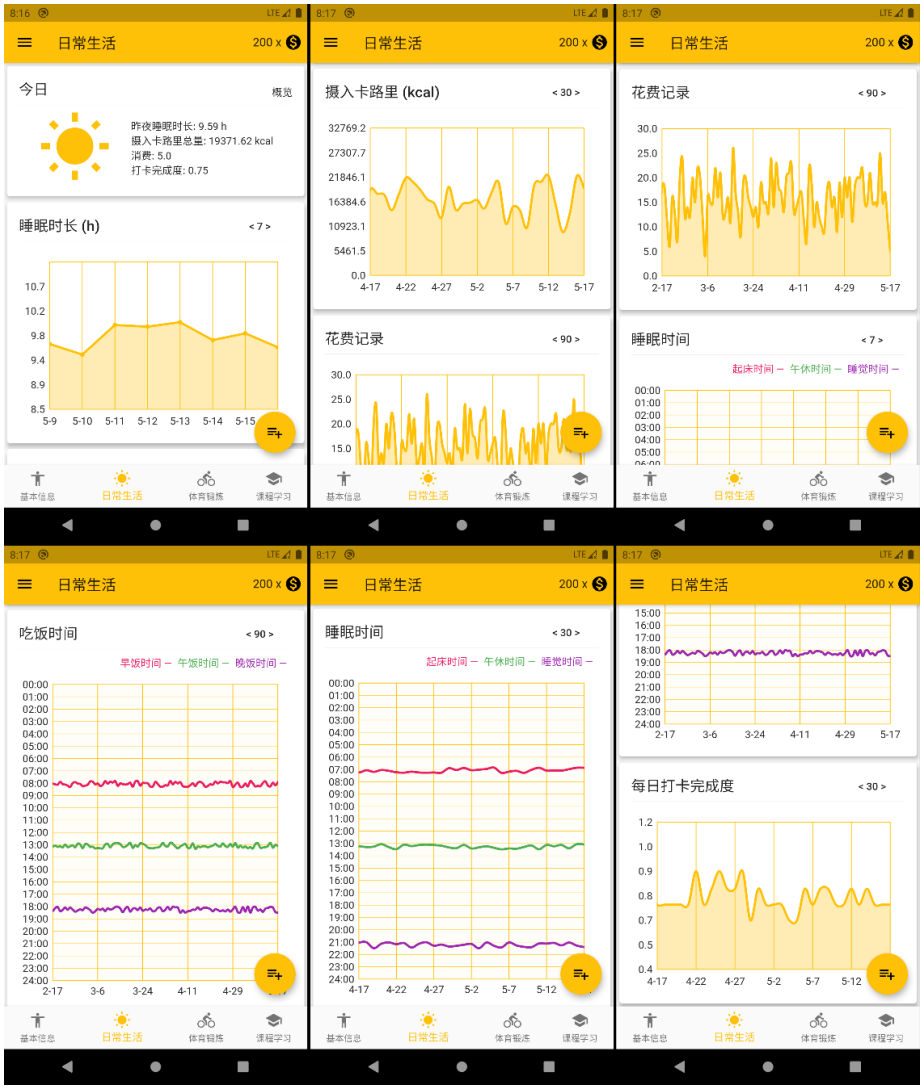

图 5.5 日常生活数据展示效果图

上图中的数据需要记录后才会对应显示。在记录数据时，用户可针对需要打卡的项目进行打卡。如果用户需要进行有关就餐的打卡，则需要先添加食物，设置该的每百克每千卡路里，食物添加一次后，每次就餐时都可以直接选择之前添加的食物。选择食物即可进行就餐打卡，打卡时需要输入进食量与消费金额。实现效果如下图 5.6 所示。

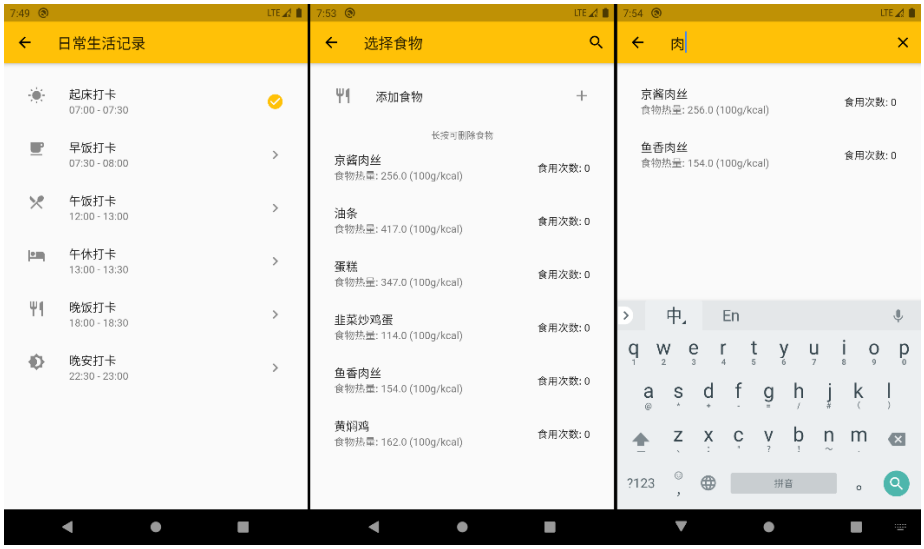

图 5.6 日常生活数据记录效果图

5.2.3 体育锻炼展示与记录

体育锻炼信息包括了类别、时长、以及消耗量。体育锻炼信息的展示与记录实现效果如下图 5.7 所示。

图 5.7 体育锻炼数据展示与记录效果图

这里记录体育锻炼的方式与记录就餐信息方式类似，需要用户添加自己的运动，之后记录该运动。

用户可进行体育锻炼的计划任务，提前添加一条计划以及目标。完成后直接进行记录。实现效果如下图 5.8 所示。

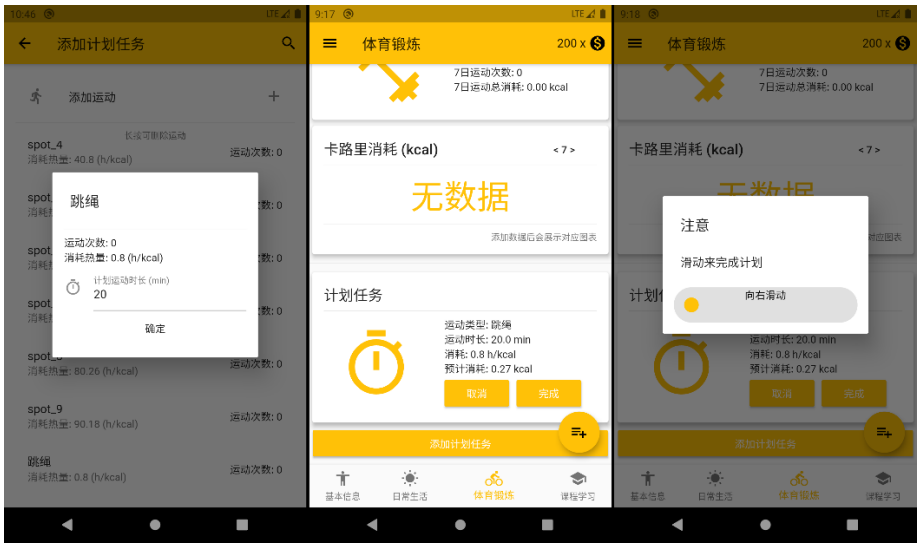

图 5.8 体育锻炼数据计划任务效果图

5.2.4 课程学习展示与记录

课程学习信息包括了学生是否迟到、是否缺席、作业是否完成、课程难度、未完成的作业以及遇到的困难。课程学习信息的展示与记录实现效果如下图 5.9 所示。

图 5.9 课程学习信息的展示与记录效果图

在用户记录自己的课程时若选择了作业未完成，用户可输入未完成的作业。

若选择了课程难度为困难，用户可输入该课程遇到的困难。在遇到的困难输入框中用户可编写 Markdown 文档，用户可以插入自己的图片，对文字排版等。若当前存在未完成的作业或未解决的问题时，在课程学习的信息展示页面将列出一系列未解决的问题。用户可浏览这节课程的详细信息，并把未完成的作业或未解决的问题标记为已完成或已解决。已解决问题且已完成的作业的课程将不再显示在课程学习的信息展示页面。实现效果如下图 5.10 所示。

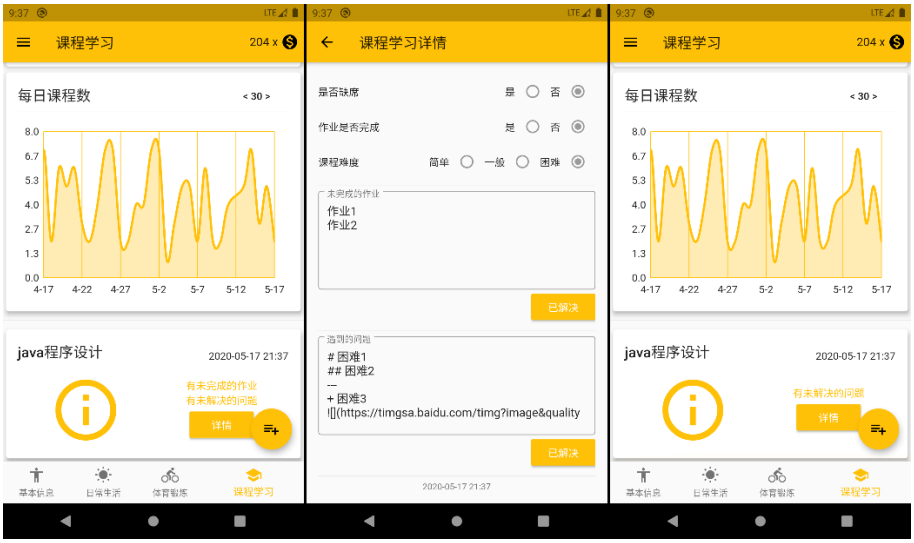

图 5.10 完成未完成的作业效果图

5.2.5 各类信息展示与记录实现原理

上文中提到的基本信息、日常生活、体育锻炼、课程学习均涉及到了跨组件的状态管理，因此均是由 Provider 实现的。本课题定义了一个 DataProvider 来专门管理这四个状态。DataProvider 有四个更新方法，每个更新方法对应一种类别信息的获取。当用户记录各数据时，会调用对应的更新方法。更新方法首先会在数据库中查出对应的数据，之后将这些数据转换为 fl_chart 插件所弄接受的点的信息 FlSport 对象。上文中所有的图表中的数据通过这些 FlSport 对象所构成的。

DataProvider 在整个程序的根节点进行注入，在加载页面进行初始化，加载四个部分的数据，并转换为对应的 FlSport。

这些 FlSport 用于构建三种折线图，分别是 DateTimeMultiLineChart、DateValueMultiLineChart、DateValueSingleLineChart，分别对应了日期时间多线折线图、日期数值多线折线图、日期数值单线带阴影折线图。这些折现图是对 fl_chart 组件的一个封装，修改了其中显示的颜色，将颜色改为主题使用的颜色，以此来实现适配不同的主题。在使用时只需要传入 List<FlSport>数据即可快速构建出设计风格一致的折线图。

5.2.6 用户信息管理

用户信息管理功能包含了登录、注册、重置密码、修改信息的功能。

用户注册、登录、重设密码时，传输的密码都是加盐后通过 MD5 加密传输的。在用户登录时，服务端会生成一条 token 并存在 MongoDB 中。之后服务端向移动端返回这条生成的 token。Token 串是根据用户 id、登录时间戳加盐后通过 MD5 加密所生成的。token 就是用户的身份令牌，移动端调用所有涉及到需要登录后才能使用的接口时，需要将 token 写入请求头中一起请求。服务端的拦截器通过接收到的 token 来确认该请求的身份是否合法，以此来决定该请求是否放行。

用户注册与登录实现了邮箱发送验证码的功能，当用户点击获取验证码按钮时，会向服务端发送一条请求。服务端接收到该请求后，会生成一个六位的数字验证码存到 MongoDB 中，并且向用户输入的邮箱发送该验证码。用户必须需要使用该验证码才能继续注册或重置密码操作。

下图 5.11 展示了登录、注册、找回密码实现效果。

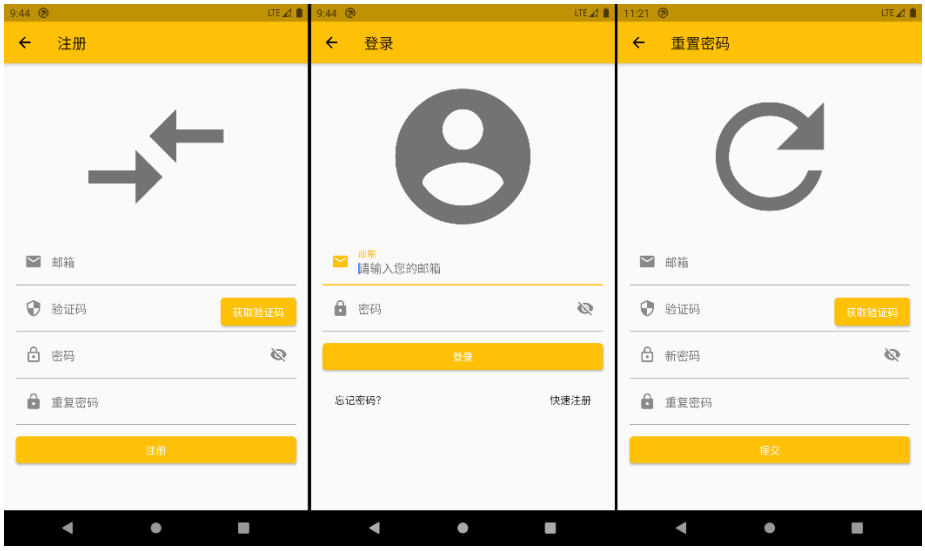

图 5.11 登录、注册、找回密码效果图

在用户修改用户信息中的头像时，可从相册选择图片作为头像，或直接使用相机拍照作为头像。

程序实现了图片的裁剪与压缩功能，这些功能涉及到三个第三方插件，分别为 image_picker 用于图片的选取与拍照、image_crop 用于图片的裁剪、flutter_image_compress 用于图片的压缩。

信息展示、修改、裁剪头像实现效果如下图 5.12 所示。

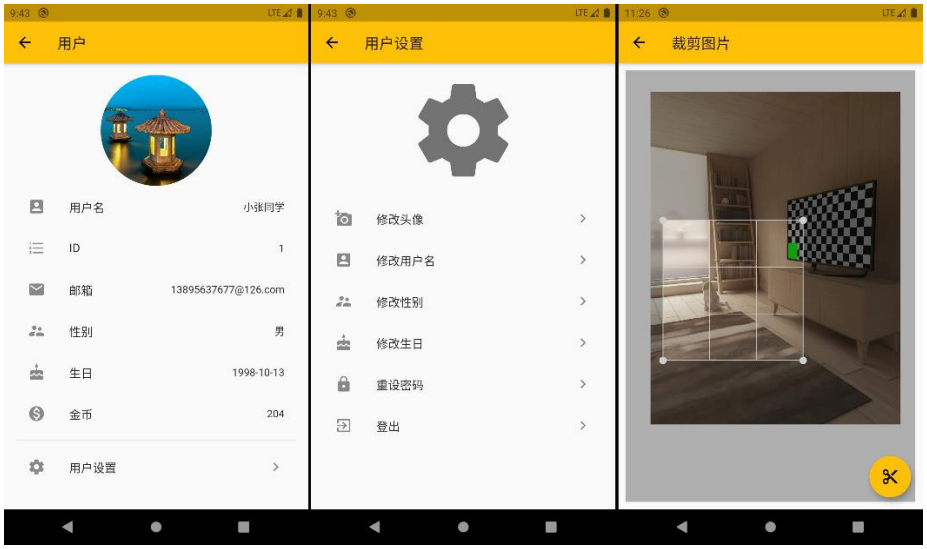

图 5.12 信息展示、修改、裁剪头像效果图

5.2.7 社区与商城

社区与商城功能主要包括了金币管理模块、排行榜模块、我的关注模块、用户搜索模块、商城模块。

本程序中实现了一套打卡签到领金币的系统，在每日第一次记录基本信息、每一条日常生活信息、体育锻炼信息、课程学习信息时，系统会为用户随机增加 7-14 枚金币。其中日常生活信息必须在各项规定时间范围内完成签到才能获得金币。用户获得的金币用于排行榜的排行、关注列表的排行、商城的购物三个方面。

排行榜、关注列表、用户搜索实现效果如下图 5.13 所示。

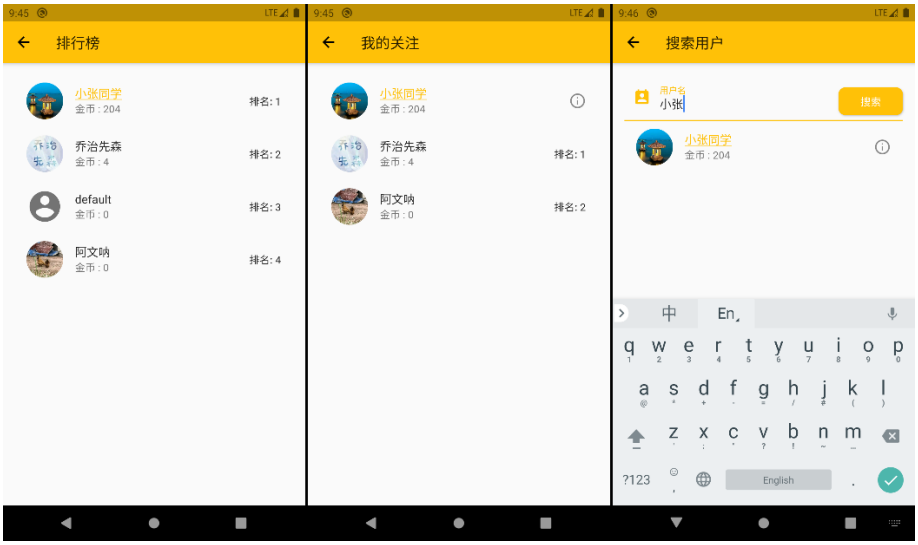

图 5.13 排行榜、关注列表、用户搜索效果图

这个打卡系统是根据波特—劳勒综合激励模型实现的[12]。排行榜会显示所

有用户中金币排名前 20 的用户以及当前登录用户的总排名，

用户可在用户搜索模块通过用户名搜索用户，可以对搜索到的用户进行关注，用户也可对排行榜中的用户进行关注操作。用户关注的所有用户将出现在关注列表中。

关注列表会按照金币数对关注的好友进行排行，使得朋友之间互相监督打卡情况。综合了排行榜与关注列表排行榜，使得用户之间产生攀比，从而激励完成每日的打卡来提升自己的排名与金币积累。

在商城中，用户可进行购物，购物实际上是购买商品的优惠券。购买成功会向用户的邮箱中发送该用户所购买的商品的兑换码。用户可根据商品描述提供的兑换码与用户自己的账号 id 向相应商家索取优惠。

兑换码是通过用户 id 与商家上架商品时提供的特征码组合后，再经过 MD5 算法加密生成的。这保证了兑换码的唯一性以及安全性，商家使用相同的算法即可得知用户提供的兑换码是否为正确的兑换码。

下图 5.14 展示了商城的实现效果。

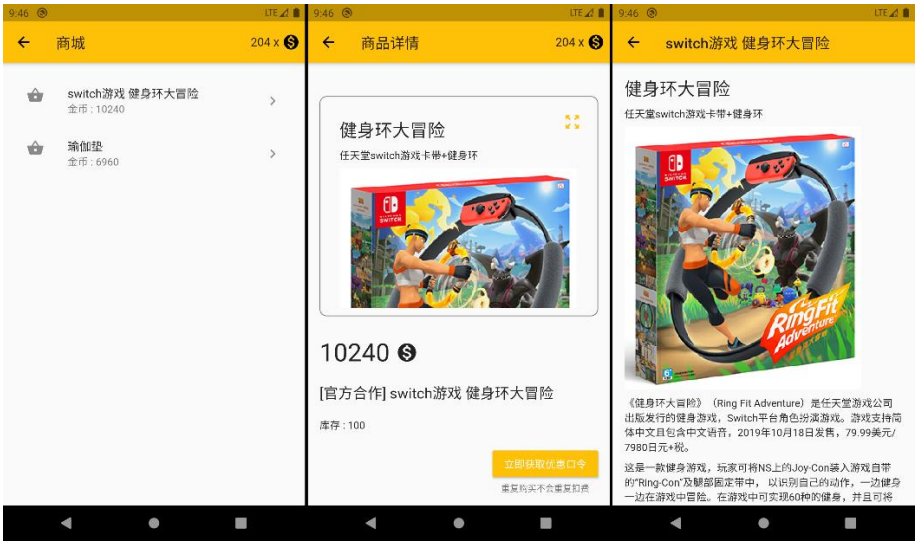

图 5.14 商城效果图

关于安全性，本课题还实现了针对恶意刷金币操作的封禁处理。移动端每次发起增加金币的请求时，服务端会先在 MongoDB 的反作弊表中查询该用户的增加金币记录，如果没有则创建一条增加金币的记录。这条记录记录了用户增加金币的次数，并且设置自动过期时间为 12 小时。反作弊表中有一个 times 字段，初始值为 0。每次用户增加金币时，均会使 times 的值加一，当 times 的值达到 100 时，将会封禁该用户。

换言之反作弊系统会封禁每 12 小时内增加了 100 次金币的用户，这种用户

被视为非正常操作的恶意用户，正常使用本软件不会达到每 12 小时内增加了 100 次金币。因此出现这种情况只会是用户的恶意行为，作为惩罚被封禁用户的金币会被置为负数，这样该用户的排名会位于好友列表的最底端。并且该用户再也无法获取金币，每次获取金币时均会提示账户封禁警告字样。

5.2.8 通知提醒

通知提醒包括两部分，一是通知的开关，二是通知时段的设置。通知提醒主要通过第三方插件包 flutter_local_notifications 实现，这个第三方插件封装好了与原生交互的功能。在使用时只需要调用其接口即可实现安卓与 iOS 的原生消息推送。用户在设置中调整日常生活打卡的通知开关。用户也可以对日常生活的通知打卡时间进行设置。

在用户对通知进行相关设置后，程序会取消先前的计划任务，重新开启计划任务。

5.2.9 主题切换

多主题的实现效果如下图 5.15 所示。

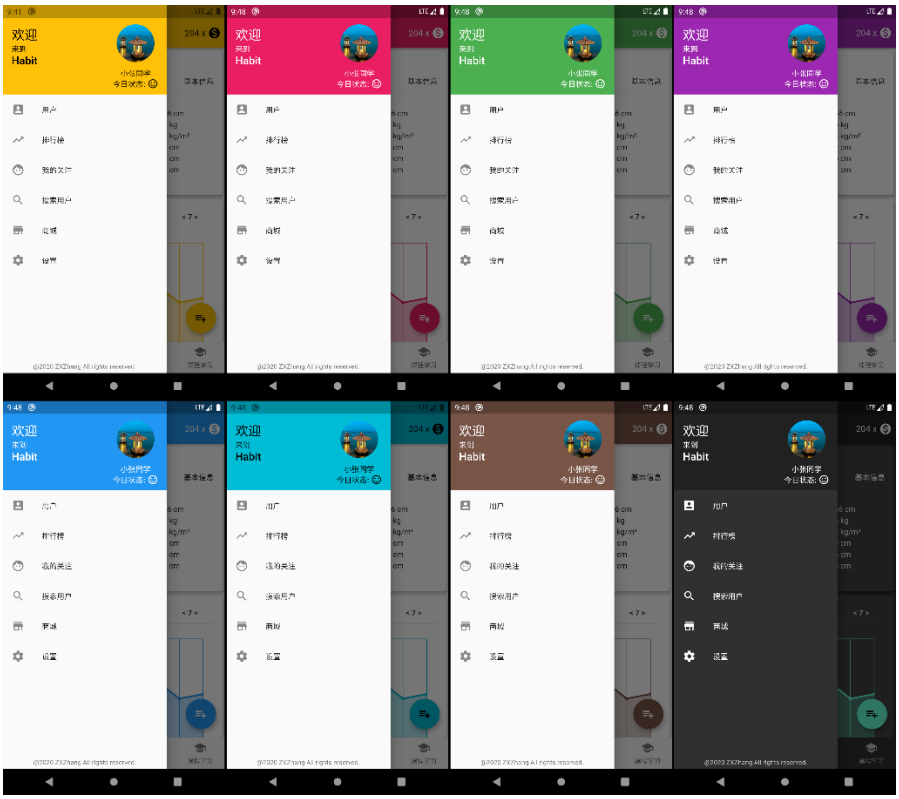

图 5.15 多主题效果图

本课题实现了八种可选主题的切换，要实现主题切换则必须实现以下两点要求：

① 在开发的过程中，所有涉及到样式的定义时，均使用 Theme.of(context)取

到应用主题后，从主题信息中获取对应的样式，如颜色、字体等。

② 必须找到办法修改根节点 MaterialApp 中的 theme 中的属性。这一点可以通过使用 Provider 来实现。通过 Provider 在根节点注入一个主题的 Model，在定义 primarySwatch 与 brightness 属性时，使用 Model 中的变量。这样想要修改主题，只需要修改 Model 中的主题信息，并通知更新即可实现主题的切换。

5.2.10 国际化

国际化可以不使用 Provider 来实现，这是因为整个项目中文字占到了大部分的内容。使用 Provider 精确更新整个程序中的文字内容，与直接 setState 更新整个组件树在效率上并没有什么太大的区别，因此使用比较简单的 setState 方式即可实现国际化。

实现国际化需要定义一个 Map<String, Map<String, String>>的结构作为国际化的字典，这个字典里定义了不同语言下不同的显示内容。将字典封装到 I18N 类中，在所有使用到文字的地方均使用 I18N 类的静态方法来获取当前语言对应的字符串即可。

5.2.11 数据同步

数据图同步主要同步用户的设置数据、以及用户每日记录的所有信息。移动端会将用户的设置数据以及记录信息打包成为一个二进制文件，发送给服务端进行储存。当数据恢复时，从服务器请求到之前上传的备份数据，解包后将本地设置数据以及记录信息恢复为之前备份的数据。这样用户只要数据存在备份，在更换设备时也可以继承之前设备中的数据。

5.2.12 今日状态评估

在主页的抽屉菜单中有一个表示今日状态的图标，该图标是一个表情符号，它有三种状态，分别是开心、不开心、难过。状态的评定主要与每日的打卡完成度有关，打卡完成度最大值为 14 点，其中基本信息、体育锻炼、课程学习各占 1 点，日常生活占 12 点。所有的打卡操作都能得到对应的 1 点打卡完成度，其中日常生活中剩余的 6 点完成度，必须在规定的时间范围内打卡才能拿到。即规定时间内打卡能拿到 2 点完成度，否则只能拿到 1 点完成度。

评定今日状态是在 DataProvider 中实现的，在 DataProvider 的初始化过程中，会先加载主界面各个信息的数据，之后根据当天的数据，评估出当天的打卡状态。在之后的数据更新时，只会直接查询全部的当天数据来评估今日状态。

6 项目部署

本章主要介绍了项目服务端的自动化部署。

6.1 部署配置

本项目是使用 Docker Compose 进行部署的，部署配置由三部分组成 Java 容器部分、MySQL 容器部分以及 MongoDB 容器部分。其中 MySQL 容器与 MongoDB 容器均不暴露端口，保证了安全性。并且 MySQL 容器与 MongoDB 容器均进行了持久化卷映射，这保证了当数据库容器宕机时，数据会得到保留。Java 容器暴露了 80 端口与 465 端口，分别用于服务端访问与邮件发送。Java 容器通过 Docker 内置网关连接到 MySQL 与 MongoDB 中。

6.2 构建部署流程

项目主要需要部署 Java 容器、MySQL 容器以及 MongoDB 容器。具体构建流程如下图 6.1 所示。

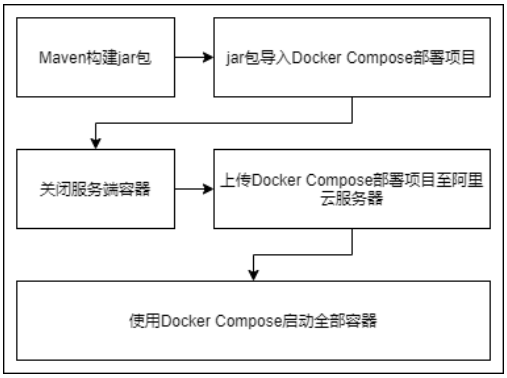

图 6.1 项目构建流程

首先在开发端中，开发者先执行打包上传脚本。该脚本会打包 Spring 项目为 jar 包，之后将 jar 包拷贝至 Docker Compose 项目下。拷贝完成后，会将整个 Docker Compose 项目上传至云服务器。开发者在脚本执行完成后，登录云服务器，在云服务器中关闭正在运行的容器，并启动新的容器。

这种构建方法是本地构建后上传至云端，上传的过程逻辑比较复杂，涉及到云服务器的连接，项目一旦开源，开发者的疏忽可能会使得云服务器密码暴露。

6.3 自动化构建部署

针对上一小节的问题，可将部署方式改为云服务器构编译。因此需要用到

GitHub 来存储代码，云服务器从 GitHub 拉下代码后，在服务端直接构建并运行。改进后的构建流程如下图 6.2 所示。

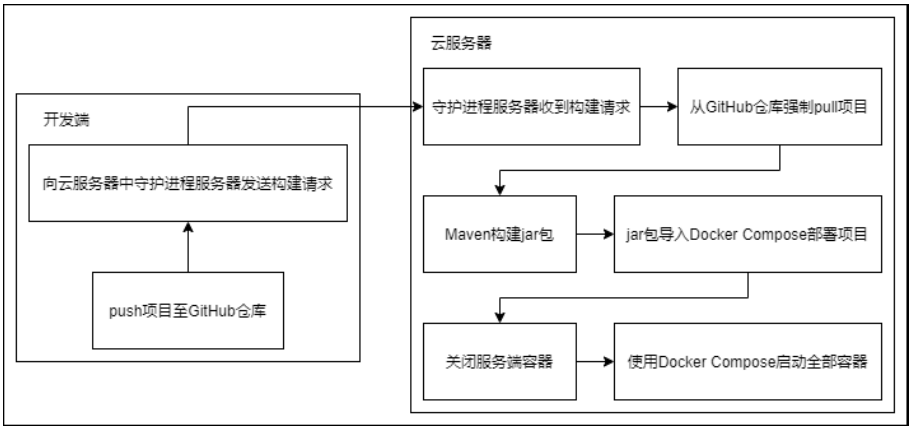

图 6.2 引入 GitHub 后的项目构建流程

在这种构建部署流程中，作为开发者。首先只需将项目推至 GitHub 仓库中，之后调用云服务器中守护进程的构建接口即可构建部署项目。云服务器中的守护进程是一个简单的 Spring 后端程序，是专门为项目部署服务的。守护进程在收到构建请求的时候会从 GitHub 上拉下最新的代码库并进行项目构建与重启容器。

通过使用上述这种构建流程，即可在开发的过程中快速的自动化部署项目，使得开发与测试效率变得更高。

7 总结与展望

本章是对本文的一个总结与展望。总结概括了本文的研究成果，并对存在改进空间的设计提出了展望。

7.1 总结

在当今开发技术快速迭代的背景下，Flutter 框架一经推出，就受到了移动端开发行业的广泛关注。目前为止，从发布 Flutter 1.0 版本到今天，只经过了一年半的时间，但是其目前的热度已经远超其他跨平台开发框架。类似腾讯、阿里巴巴等大厂均在 Flutter 上进行着一定的探索与实践。其中阿里巴巴的闲鱼团队已经将 Flutter 落实在了闲鱼 app 的开发上。

本课题实现的个人习惯监测软件是基于 Flutter 的，这也是对最新技术的一种探索。为了探索 Flutter 应用的能力，一般移动应用所需要的功能在本项目中都有体现，如原生推送、原生的图片选择器、移动端数据库、Http 客户端等。

本项目不只在基础功能使用实现方面进行了探索，还在架构方面进行了一定的探索。通过分析现有的解决方案，自主设计了一套解耦的框架，提高了程序运行效率，降低了开发与维护的成本。在部署方面，本课题应用了容器化的技术，并在云服务器上实现了一个简单的守护进程服务端，用于一键自动化部署项目。

本课题在针对性能与框架中存在的一些问题提出了自己的看法，并给出了一些解决方案，这些方案具有一定的参考价值。但是由于笔者自身能力有限，这其中也可能存在着一些不足之处，还请读者多多包涵。

7.2 展望

结合系统设计与系统实现分析，个人习惯监测软件的改进还可从以下方面入手：

①数据记录的方式：

手动录入数据的用户成本比较高，用户每一种数据的记录都需要自己完成。后续应该联动智能家居设备，如智能体重秤，智能手环等设备，实现数据的自动记录。

②社区功能：

社区目前的功能还比较少，要想继续加深用户之间的互联，后续应该加入实时聊天、点赞、留言板等互动功能。这些功能可以提高用户的参与度，使得用户坚持打卡。

③商业目的：

任何应用程序都需要盈利，目前情况来看，收入来源只有可能来自与第三方商家的合作来获取收入。后续可以考虑更多的盈利手段，如充值、广告位等。

致谢

时间过得飞快，我也经历了属于我自己的大学四年时光，一路走来感触良多。

首先，感谢我的母校重庆大学给予我完成学业的机会，母校的平台给予了我更宽广的视野，让我见识到了更广的天空！

其次，感谢指导我研究本课题的杨瑞龙老师。杨老师认真负责，在我毕业设计的过程中指出了许多我忽略的问题，并加以指点，万分感谢！

此外，还要感谢所有授我以业的老师，感谢多年来的悉心培养，如若没有这几年的知识积累与技术沉淀，我也不会完成本次毕业设计，更没有勇气挑战最新的技术！

最后，感谢在我大学期间给予我无私帮助的同学，以及对我学业支持的家人！

谢谢各位！

参考文献

[1] 于志浩.基于 Android 和网络爬虫的课外阅读系统设计与实现[19].山东大学,2018.

[2] 工信部运行监测协调局.工信部发布 2018 年通信业统计公报[EB/OL].[2019-02-25].

[3] 李勇,王静丽.运动干预对大学生手机成瘾程度的影响[J].智库时代,2019(47):204-205.

[4] 邓皓瀚.基于 Flutter 的跨平台移动 APP 开发前景研究[J].信息与电脑,2019,0(15):197-199.

[5] 周勇,程子清.Flutter 的原理深度剖析[J].电脑编程技巧与维护,2018,0(11):19-21.

[6] 周玉轩,杨絮,鲍富成等.Cordova-NodeJS 混合式物联网信息服务系统[J].计算机工程与应用,2019,55(6):209-217.

[7] Gao,Xingjian,Hua 等.基于 Ionic 的混合移动应用的研究与实现[J].计算机时代,2018,0(3):31-34.

[8] 李睿.基于 Weex 的小程序系统设计[J].中国科技信息,2019,0(21):64-65.

[9] 华为、小米、OPPO、vivo 等十大手机厂商联合发布快应用标准[J].微型计算机,2018,0(10):35-35.

[10] 姚静.微信小程序页面路由原理[J].电脑知识与技术：学术版,2019,15(10):54-55.

[11] 胡昆,奚斌.基于 Material Design 设计语言的隐喻设计研究[J].设计,2017,0(15):136-137.

[12] 莫申容.基于综合激励模型的基层中青年干部激励机制研究——以重庆 S 区为例[J].领导科学,2019,0(18):106-109.

[13] 王海霞.云计算场景中 Docker-Compose 安装部署 及 Nginx 应用构建实践[J].河北北方学院学报：自然科学版,2019,35(7):14-19.

[14] 邓成,孙书会.MVVM 设计模式的前端应用[J].电脑知识与技术：学术版,2019,15(29):249-250.

[15] 王丹,孙晓宇,杨路斌等.基于 SpringBoot 的软件统计分析系统设计与实现[J].软件工程,2019,22(3):40-42.

[16] Vijayan,,Jaikumar.Google Begins Development of Fuchsia Open-Source OS[J].eWEEK,2016:3-3.

[17] Vijayan,,Jaikumar.Google Releases Beta Version of Flutter Mobile App Development Tool[J].eWEEK,2018:1-1.

[18] Wu W. React Native vs Flutter, Cross-platforms mobile application frameworks[J]. 2018.

[19] Dagne L. Flutter for cross-platform App and SDK development[J]. 2019.

[20] Fayzullaev J. Native-like cross-platform mobile development: Multi-os engine & kotlin native vs flutter[J]. 2018.