

**《软件工程》项目报告**

**题目： 小星球APP**

**课程名称： 软件工程**

**专业班级： ACM1901**

**学 号： U201915128**

**姓 名： 蓝冰瑛**

**同组成员： 张弘弢**

**邹雅**

**指导教师： 万胜刚**

**报告日期： 2021.12.29**

**计算机科学与技术学院**

**任 务 书**

**一 总体要求**

1. 综合运用软件工程的思想，协同完成一个软件项目的开发，掌软件工程相关的技术和方法；

2. 组成小组进行选题，通过调研完成项目的需求分析，并详细说明小组成员的分工、项目的时间管理等方面。

3. 根据需求分析进行总体设计、详细设计、编码与测试等。

**二 基本内容**

根据给出的题目任选一题，自行组队，设计与开发中软件过程必须包括：

**1. 问题概述、需求分析：**正确使用相关工具和方法说明所开发软件的问题定义和需求分析，比如NABCD模型，Microsoft Visio，StarUML等工具 (20%)；

**2. 原型系统设计、概要设计、详细设计**：主要说明所开发软件的架构、数据结构及主要算法设计，比如墨刀等工具（35%）；

**3. 编码与测试**：编码规范，运用码云等平台进行版本管理，设计测试计划和测试用例（30%）；

**4．功能创新**：与众不同、特别吸引用户的创新（10%）；

**5. 用户反馈**：包括用户的使用记录，照片，视频等（5%）。

目录

[**1 问题定义 1**](#_Toc91712151)

[1.1项目背景与意义 1](#_Toc91712152)

[1.1.1项目背景 1](#_Toc91712153)

[1.1.2项目意义 1](#_Toc91712154)

[1.2.1 需求(Need) 1](#_Toc91712155)

[1.2.2 方法(Approach) 1](#_Toc91712156)

[1.2.3 获益(Benefit) 2](#_Toc91712157)

[1.2.4 竞争产品分析(Competition) 2](#_Toc91712158)

[1.2.5 发布(Delivery) 2](#_Toc91712159)

[1.3 可行性分析 2](#_Toc91712160)

[1.3.1 市场需求 2](#_Toc91712161)

[1.3.2 技术支持 3](#_Toc91712162)

[1.3.3 数据安全 3](#_Toc91712163)

[1.4人员管理和项目进度管理 3](#_Toc91712164)

[1.4.1 任务主体安排 3](#_Toc91712165)

[1.4.2 项目进度管理 3](#_Toc91712166)

[**2 需求分析 6**](#_Toc91712167)

[2.1 E-R图 6](#_Toc91712168)

[2.2 用例图 6](#_Toc91712169)

[2.3 原型系统设计 7](#_Toc91712170)

[2.3.1 设计工具 7](#_Toc91712171)

[2.3.2 设计思路 7](#_Toc91712172)

[2.3.3 设计成果 7](#_Toc91712173)

[**3 概要设计和详细设计 14**](#_Toc91712174)

[3.1 系统结构 14](#_Toc91712175)

[3.1.1 功能模块 14](#_Toc91712176)

[3.1.2 接口设计 20](#_Toc91712177)

[3.1.3 整体系统结构 28](#_Toc91712178)

[3.2 关键数据结构定义 28](#_Toc91712179)

[3.2.1 后端数据结构 29](#_Toc91712180)

[3.2.2 客户端数据结构 31](#_Toc91712181)

[3.2.3音乐推荐系统数据结构： 33](#_Toc91712182)

[3.3 关键算法设计 33](#_Toc91712183)

[3.4 数据管理说明 34](#_Toc91712184)

[3.4.1 客户端数据管理 34](#_Toc91712185)

[3.4.2 后端和音乐推荐系统数据管理 35](#_Toc91712186)

[**4 实现与测试 36**](#_Toc91712187)

[4.1实现环境与代码管理 36](#_Toc91712188)

[4.1.1 实现环境 36](#_Toc91712189)

[4.1.2 代码管理 36](#_Toc91712190)

[4.2 关键函数说明 37](#_Toc91712191)

[4.2.1 客户端关键函数 37](#_Toc91712192)

[4.2.2 后端关键函数 38](#_Toc91712193)

[4.2.3 音乐推荐关键函数 41](#_Toc91712194)

[4.3 测试计划和测试用例 42](#_Toc91712195)

[4.3.1 常用测试方法 42](#_Toc91712196)

[4.3.2 APP测试 43](#_Toc91712197)

[4.4 结果分析 47](#_Toc91712198)

[**5 总结 48**](#_Toc91712199)

[5.1 用户反馈 48](#_Toc91712200)

[5.1.1 用户反馈图文 48](#_Toc91712201)

[5.1.2 用户反馈总结 49](#_Toc91712202)

[5.2 全文总结 49](#_Toc91712203)

[**6 体会 51**](#_Toc91712204)

# 1 问题定义

## 1.1项目背景与意义

### 1.1.1项目背景

在现今社交媒体软件层出不穷的时代，例如微博，抖音，豆瓣等等，只要打开一个软件，就能够看到无数的资讯。当人们习惯了日常生活被无尽的，而又不那么重要的信息充满后，便不会再习惯自身的信息产出，和一个人独处的独处生活。在现实中，的确有更多人，在无所事事的空闲时间，抑或是夜晚无法入眠的时候，会下意识地会打开手机，持续刷着无意义的信息。在这样的恶性循环中，人们将逐渐忘记应该如何思考，应该如何面对自己的情绪，而仅仅将社交媒体作为自己的情感依托。

### 1.1.2项目意义

我们小组希望能做出一个让用户专注自身的APP，创造一个只有用户的空间，去安静的思考和记录生活。使用户逐渐减少过高的社交媒体使用频率，鼓励用户树立自己独立思考的能力，不会被社交媒体上的风向带偏。同时，我们希望这个项目能够让用户有意识地摆脱长时间看手机的习惯，而是去专注于现实生活，做更有意义的事。

**1.2 项目基本目标**

下面，将使用NABCD模型来说明我们小组对于这个APP的期待，从实现功能，为用户带来什么，以及到最后的发布，制订一系列明确的目标。

### 1.2.1 需求(Need)

APP具有三大基本功能：为用户推荐音乐，记录用户心情，提交日记功能。

### 1.2.2 方法(Approach)

* 用户可以编写日志，记录每天的心情。
* APP每天会在固定时间提醒用户撰写日记，督促用户养成记日记的好习惯。
* 通过用户选择的心情以及喜欢的音乐列表，并将其他同样心情的用户喜欢了什么音乐作为参考，来为用户推荐音乐。
* 每天爬取最新发布的音乐，保证推荐音乐歌单的实时性。

### 1.2.3 获益(Benefit)

* 让用户拥有简洁，高效的平台记录每日心情。培养用户思考和记录日记的习惯。
* 让用户听到更多他可能感兴趣的音乐。
* 保证了每位用户的隐私，用户不能看到其他用户的任何数据。

### 竞争产品分析(Competition)

* 对于常用音乐软件，例如网易云、QQ音乐等，它们隐私性较差。这两个APP都加入了用户社区和评论功能，看评论听歌容易被别人的评论带偏。同时，它们商业气息浓重，每次打开都有很多广告，放不设防。并且，常用音乐软件都缺少日记记录功能。同时，它们并不会考虑到用户当前的心情去推荐音乐，并且推荐的歌曲很有可能是公司购买的广告，并不是基于用户的喜好来推荐的。我们的APP相较于常用音乐软件的劣势也非常明显，
* 像其他音乐推荐软件，例如Fly Music，缺少了相关联的推荐歌曲功能，并不能根据用户喜爱歌曲的列表去推荐歌曲，缺少了用户的个性定制。然而，由于音乐版权的原因，我们的APP无法直接播放音乐，只能使用外链让用户跳转到其他页面去收听，因此有潜在流失用户的风险。
* 与其他日记软件相比，我们APP的日记记录逻辑会简单很多，但我们后续会将其功能与听歌和心情部分结合起来，使日记功能的利用率达到最大化。

### 1.2.5 发布(Delivery)

* 完成所有需求后会上架到苹果商店，供iPad和iPhone用户使用。在上架之前，可以使用Mac进行源代码编译，在iPhone上进行尝鲜体验。
* 继续进行下一步完善，添加日志、音乐共享，增加用户与用户之间的交互。持续推送新的版本。

## 1.3 可行性分析

### 1.3.1 市场需求

我们小组认为，大部分社交媒体用户实际上对于APP中过多的广告，以及没有营养的信息输入已经疲倦，他们缺少的只是没有定位明确的APP来取代现有社交媒体。因此我们觉得我们的APP会令用户耳目一新，使产品脱颖而出。

### 1.3.2 技术支持

老师在我们设计项目时曾经提到过有关冷启动的问题，实际上我们不仅仅会基于APP中用户喜爱的音乐作为推荐音乐的原始数据，在APP推广的初期阶段，我们采用的是爬取各大音乐评价网站上的数据，基于那些网站上的用户评分和音乐分类进行基本的推荐。

为保证数据同步，需要购买云服务器。同时，为了保证后续推荐系统的稳定性，我们也需要单独购买云数据库服务，保证训练的数据集不会因为体积突然增大服务器无法容纳。

### 1.3.3 数据安全

在建立数据库时可以通过对数据库的设计用户名和密码进行保密，以及在建立数据库以后可以对数据进行压缩等技术，保证数据的安全，使数据库具有安全保障。

## 1.4人员管理和项目进度管理

### 1.4.1 任务主体安排

页面设计：张弘弢，邹雅，蓝冰瑛

IOS前端设计：邹雅

后端支持：张弘弢

推荐算法：蓝冰瑛

### 1.4.2 项目进度管理

为了更好地进行项目进度管理，我们将使用甘特图将项目任务进行细化，将任务具体分配给每个人。考虑到我们的项目有三大模块构成，在后续的联合调试过程中会耗费很多时间，因此在开发的测试阶段会安排充足的时间。

* 总体设计

表1-1 人员分配

|  |  |  |  |
| --- | --- | --- | --- |
| 负责人 | 任务 | 计划开始日 | 计划天数 |
| 蓝冰瑛、邹雅、张弘弢 | 立项 | 2021-11-12 | 7 |
| 蓝冰瑛、邹雅、张弘弢 | 需求分析 | 2021-11-19 | 3 |
| 蓝冰瑛、邹雅、张弘弢 | 原型开发 | 2021-11-19 | 7 |
| 蓝冰瑛、邹雅、张弘弢 | 系统设计 | 2021-11-24 | 3 |

* 开发

APP部分：

表1-2 APP任务

|  |  |  |  |
| --- | --- | --- | --- |
| 负责人 | 任务 | 计划开始日 | 计划天数 |
| 邹雅 | 登陆、注册和忘记密码页面构建 | 2021-11-24 | 2 |
| 邹雅 | 个人中心页面构建 | 2021-11-26 | 2 |
| 邹雅 | 音乐推荐页面构建 | 2021-11-28 | 1 |
| 邹雅 | 首页页面页面构建 | 2021-11-29 | 2 |
| 邹雅 | 选择心情页面构建 | 2021-11-30 | 2 |
| 邹雅 | 天气页面页面构建 | 2021-12-01 | 2 |
| 邹雅 | 日记页面页面构建 | 2021-12-02 | 1 |
| 邹雅 | 搜索页面页面构建 | 2021-12-03 | 1 |
| 邹雅 | 实现用户与APP的交互 | 2021-12-04 | 3 |
| 邹雅 | 给后端提供客户端数据 | 2021-12-05 | 3 |

后端部分：

表1-3 后端任务

|  |  |  |  |
| --- | --- | --- | --- |
| 负责人 | 任务 | 计划开始日 | 计划天数 |
| 张弘弢 | 完成mysql和redis的配置 | 2021-11-28 | 1 |
| 张弘弢 | 实现用户登陆注册功能 | 2021-11-30 | 1 |
| 张弘弢 | 实现用户的心情选择 | 2021-12-02 | 3 |
| 张弘弢 | 实现天气，地理位置的接口 | 2021-12-04 | 5 |
| 张弘弢 | 实现用户的所有日记功能 | 2021-12-05 | 4 |
| 张弘弢 | 实现音乐推荐功能 | 2021-12-06 | 4 |
| 张弘弢 | 完成与推荐功能的交互 | 2021-12-07 | 3 |
| 张弘弢 | 完成与app的交互 | 2021-12-08 | 3 |

AI部分：

**表1-4 AI任务**

|  |  |  |  |
| --- | --- | --- | --- |
| 负责人 | 任务 | 计划开始日 | 计划天数 |
| 蓝冰瑛 | 数据挖掘 音乐信息 | 2021-11-26 | 4 |
| 蓝冰瑛 | 数据挖掘 天气信息 | 2021-11-26 | 3 |
| 蓝冰瑛 | 数据分析 心情解析 | 2021-11-30 | 4 |
| 蓝冰瑛 | 数据分析 推荐音乐 | 2021-12-01 | 3 |
| 蓝冰瑛 | 数据分析 用户聚类 | 2021-12-03 | 3 |
| 蓝冰瑛 | 数据分析 用户画像 | 2021-12-04 | 2 |

集成测试：

表1-5 集成测试任务

|  |  |  |  |
| --- | --- | --- | --- |
| 负责人 | 任务 | 计划开始日 | 计划天数 |
| 蓝冰瑛、邹雅、张弘弢 | 服务器部署 | 2021-12-02 | 1 |
| 蓝冰瑛、邹雅、张弘弢 | 域名解析 | 2021-12-02 | 1 |
| 蓝冰瑛、邹雅、张弘弢 | 服务器维护 | 2021-12-02 | 1 |
| 蓝冰瑛、邹雅、张弘弢 | 内网管理端口 | 2021-12-02 | 1 |

* 发布

由于将APP提交给Apple Store的时间周期较长，因此发布时间不可预估。

我们小组的总体甘特图如下：

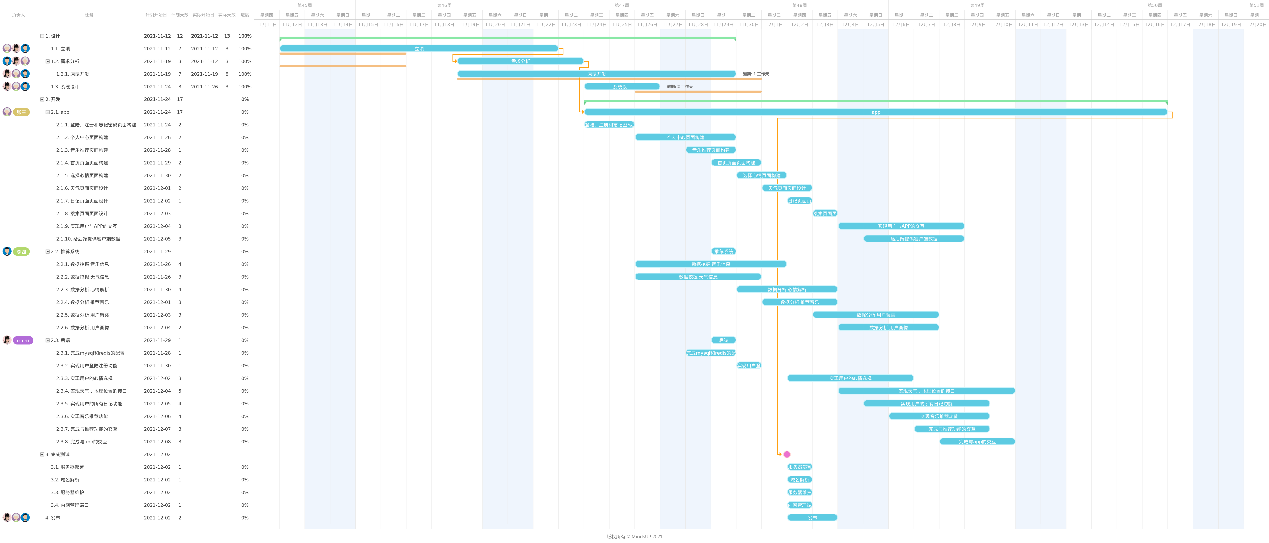


图1-1 任务进度甘特图

# 2 需求分析

## 2.1 E-R图

我们小组在前期通过一系列的调研和分析，最后的系统E-R图和数据流图如下图2-1和2-2所示：

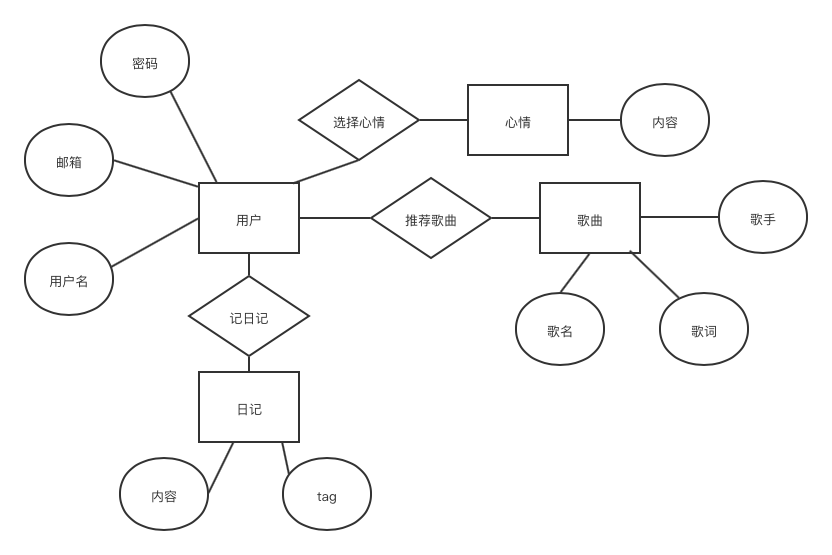


图2-1 系统E-R图

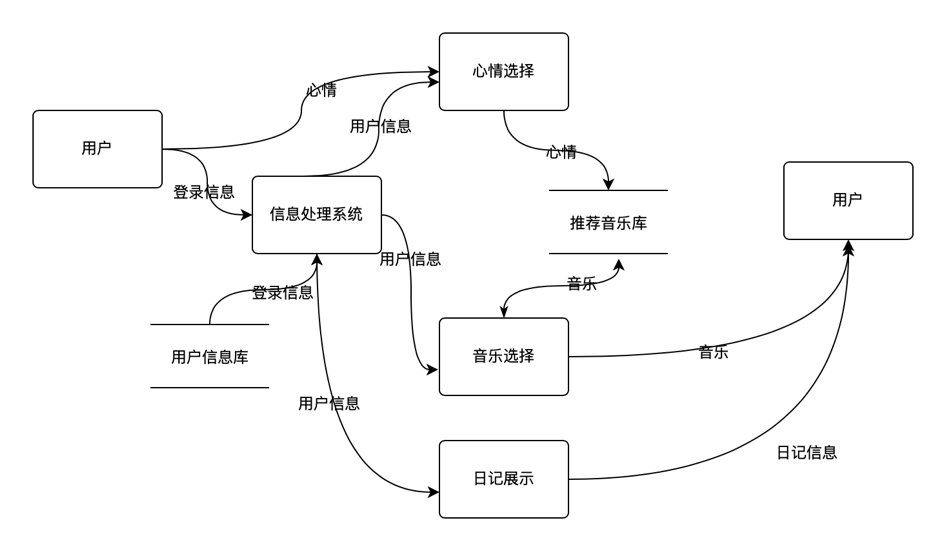


图2-2 数据流图

## 2.2 用例图

系统用例图如下图2-3所示：

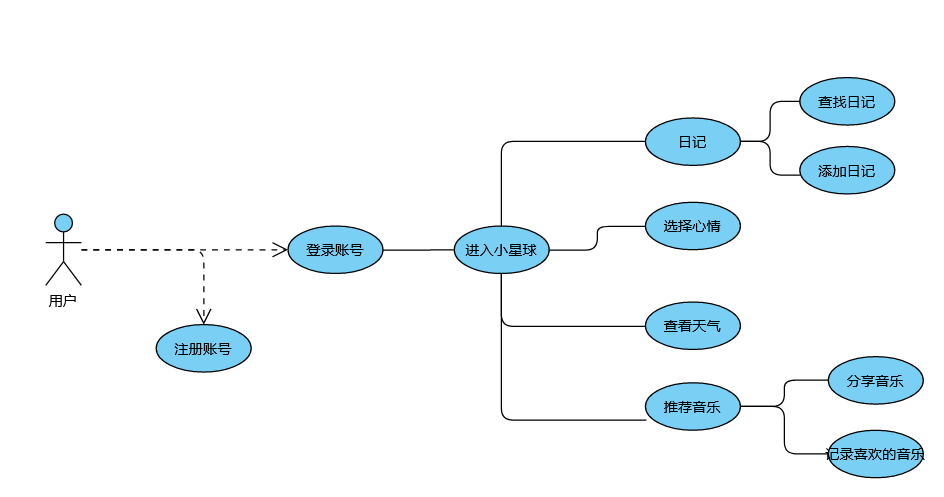


图2-3 系统用例图

## 2.3 原型系统设计

### 2.3.1 设计工具

墨刀企业版。

### 2.3.2 设计思路

基于APP开发，因此初步设计11个页面。根据功能将页面分为四个部分来描述：用户登录部分、小星球（主页）部分、日记部分、音乐推荐部分。

### 2.3.3 设计成果

原型系统效果详见网页：

https://modao.cc/app/kwhdg7yr090x2g#screen=skvxfmnzx2bv7lq

1. 用户登录

登录部分有APP的图标以及常规的账号密码，其中包括了忘记密码和注册账号，每个输入框有相应的提示信息；注册部分也与登录部分类似，主要是增加了注册所需要的信息，忘记密码则是基于注册部分将头像换成APP图标。具体效果如图所示。



图2-4 注册界面



图2-5 登录界面



图2-6 忘记密码

1. 小星球（主页）

进入APP首先看到的是一颗属于自己的小星球，在星球上面可以选择进入日记、音乐、天气板块，所以也在星球上有对应的按钮，星球下方的按钮可以进入心情选择，左上方的问候语也可以进入用户信息。心情选择是若干按钮组成的界面，用户选择一个心情后会自动回到小星球。用户信息除了包括常规信息以外还包括了喜欢的音乐。



图2-7 小星球主页



图2-8 用户信息

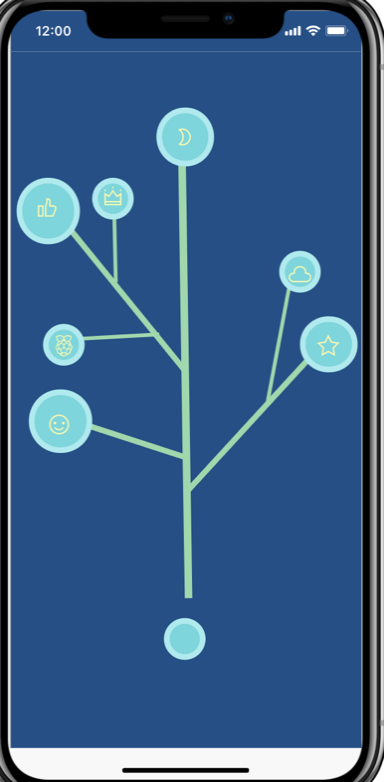


图2-9 心情选择



图2-10 喜欢的音乐列表

1. 日记

从主页进入日记部分首先是之前的日记，上方的搜索可以进入搜索功能，点击每个日记的tag同样可以进入搜索。创建日记的则是一个简洁的输入内容部分，右上方点击发送则可以添加到自己的日记记录中了。

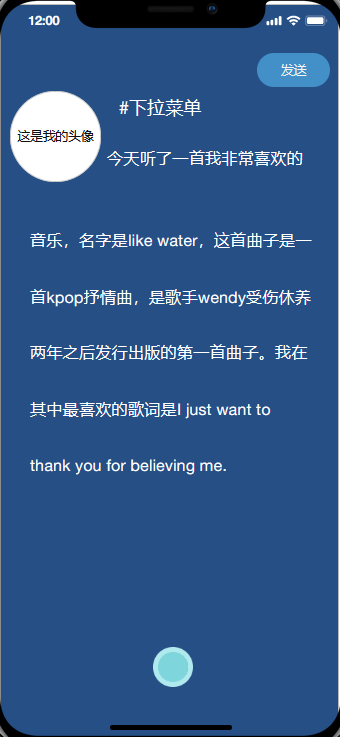


图2-11 创建日记

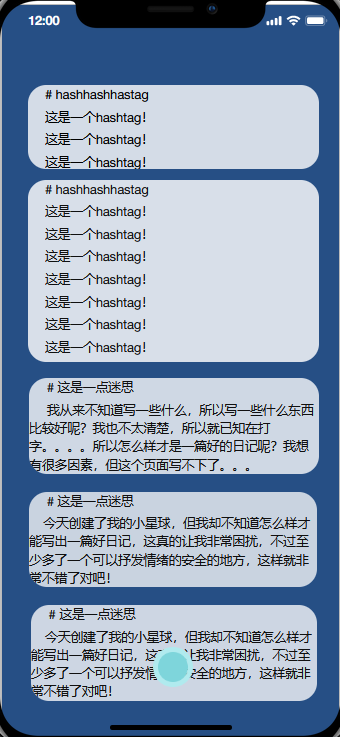


图2-12 全部日记

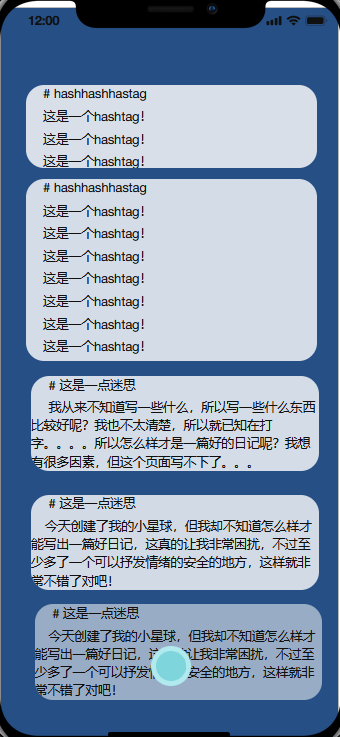


图2-13 tag查找日记

1. 音乐推荐

音乐推荐只有一个页面，背景是从数据库中获得的专辑图片或相关图片，主题是这首歌的歌词，双击可以将这首歌添加到喜欢的音乐中，右上方的下拉框可以选择分享或将歌曲添加到日记中(进入创建日记页面)。



图2-14 推荐的界面

# 3 概要设计和详细设计

## 3.1 系统结构

项目整体分为三个部分：客户端、后端和音乐推荐系统。下图为完整的系统结构设计图，本节的说明也依据此图展开。

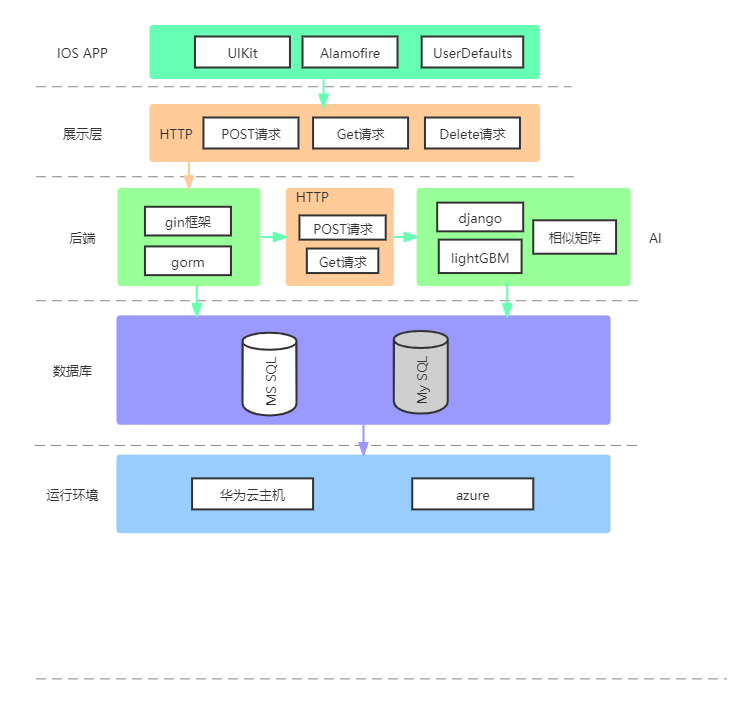


图3-1 系统结构图

### 3.1.1 功能模块

* 客户端模块

客户端界面采用MVC架构模式进行构建。视图(View)代表用户交互界面，提供给用户的操作界面，是程序的外壳。模型(Model)就是业务流程/状态的处理以及业务规则的制定，是程序需要操作的数据或信息。控制(Controller)可以理解为从用户接收请求, 将模型与视图匹配在一起，共同完成用户的请求；它负责根据用户从"视图层"输入的指令，选取"数据层"中的数据，然后对其进行相应的操作，产生最终结果。这三层是紧密联系在一起的，但又是互相独立的，每一层内部的变化不影响其他层。

1. ViewController

Controller可以分为11个模块，分别如下所述：

1. MainViewController

主界面控制器，展示小星球，以及控制跳转至个人信息、日记、推荐音乐、选择心情的界面。实现通过本地化的数据判断是否登录，若未完成登录，则日记、推荐音乐、选择心情模块都不会显示。

1. SelectMoodViewController

选择心情界面。控制选择心情后的动画，以及完成心情选择退出后对后端发起请求通知该用户当前的心情。

1. InfoViewController

展现用户的信息界面，从该界面可以退出登录，也可以跳转展现用户所有喜欢音乐的界面。

1. RegisterViewController

用户注册界面。在该界面用户可以选择头像，输入用户名、邮箱、密码、密码确认，并向后端发起注册请求。

1. LoginViewController

登录页面。在该界面用户可以输入邮箱和密码，向后端发起登录请求。

1. ForgetPasswordViewController

忘记密码页面。在该页面可以输入邮箱和新密码，并向后端发起新密码的请求。

1. RecommendMusicViewController

推荐音乐页面。在该页面向后端发起get请求，获得推荐的音乐歌曲。当获得一定数量的歌曲后，直接显示在页面上，同时创建新的线程，发起异步的请求去请求更多的推荐歌曲，完成推荐歌曲的不断推荐。同时实现了识别滑动手势并且同步让卡片右滑的动画。

1. FavoriteMusicViewController

展示所有喜爱歌曲的页面。在页面通过用户ID发起get请求，获得该用户喜欢的全部歌曲，保存下歌曲ID和歌名。其中歌名用于显示歌曲列表，歌曲列表用UICollectionView来实现。

1. CreateDiaryViewController

创建日记的页面。在该页面用户可以创建一条新的日记，并向后端发起post请求，让后端保存下日记数据。同时在该页面还可以跳转到显示所有日记的页面。

1. AllDiaryViewController

显示所有日记的页面。在该页面通过向后端请求得到该用户所有的日记并用UITableView来实现。同时在该页面有一个输入框，可以输入tag来查找所有拥有该tag的日记。

1. MusicInfoViewController

显示音乐详细信息的页面。音乐的详细信息也通过向后端发起请求来得到。

1. View

* CardView

用于显示日记详细信息的卡片UIView。其中有如下这些属性用于保存音乐相关的信息：

var id:Int = 0

var title:String = ""

var singer:String = ""

var lyrics:String = ""

var cover = UIImage()

var coverStr:String = ""

var type:String = ""

var isFavorite:Bool = false

在基础的UIView上增加如下的控件属性来实现界面布局：

**var** backgroundCard = UIImageView()

**var** titleLabel = UILabel()

**var** singerLabel = UILabel()

**var** lyricsLabel:UITextView!

**var** coverView = UIImageView()

**var** favoriteBtn:UIButton!

在这个类内部封装了两个函数——sendFavorite()和sendNotFavorite()，主要是在favoriteBtn被点击时根据当前是否喜欢来触发的。

* BubbleButton

继承自UIButton，增加一个isChoosed属性。

* FavoriteButton

继承自UIButton，增加一个isFavorite属性。

* DiaryCollectionView

用于展示日记的Cell，有一个UILabel属性展现日记的文字部分。

* MusicTableView

用于展示音乐列表的Cell，有一个UILabel属性展现音乐的歌名。

1. Model

这一部分的大部分定义如下3.3.3的APP数据结构所述。在此补充一个如下所示：

**class** ScaleAnimation:CABasicAnimation{

override init() {

**super**.init()

keyPath = "transform.scale"

duration = 0.6

repeatCount = .infinity

autoreverses = **true**

fromValue = 1.0

toValue = 1.3

}

}

这个类继承自CABasicAnimation，是用来实现选择心情中泡泡缩放的动画的。

1. NetWork

这一部分主要是封装了一系列网络请求的结构体和函数来完成对后端的网络请求以及解析得到的结果。

**struct** WebAPI {

**var** path: String

**var** method: HTTPMethod

**var** parameter: [String]?

}

该结构体封装了网络请求相关的数据，path代表URL路径，method是网络请求的method，而paremeter传入一系列参数。

NetworkMgr有一个单例对象shared，在这个类中封装了发起请求的一系列函数。NetworkRequst中封装了一系列函数用于解析网络请求的响应，便于判断响应状态和解析数据。

* 后端功能模块

后端的程序结构图如下所示：

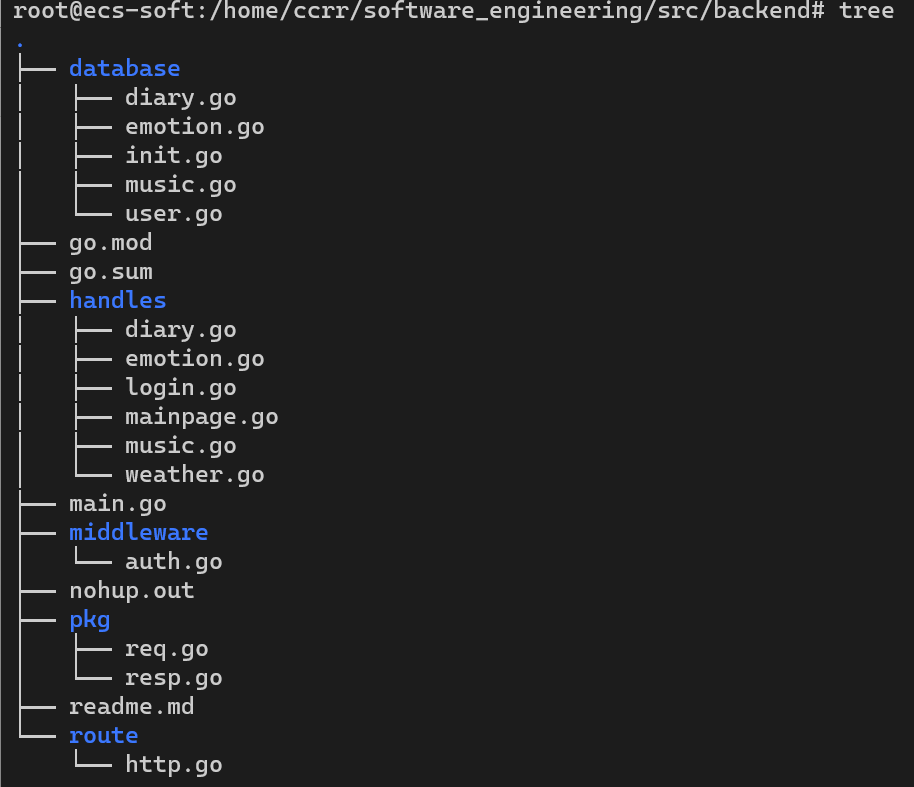


图3-2 后端程序结构图

后端主要由5个模块组成，在代码中主要以分包来体现。分别是database模块，用于定义mysql的表项，并提供直接操作数据库的接口；Handlers用于处理并相应HTTP请求，直接调用database模块提供的接口对数据进行操作，操作完毕后会向上层返回正确信息或错误提示；pkg块定义了每一个HTTP的请求和回答数据结构，并规范定义了请求体中json文件的字段名称，保证请求传输准确无误。Route是基本的路由配置模块，在该模块中会使后端程序运行在服务器的指定端口上，并对每一个路径的请求做出相应响应，即调用handlers中已经实现的方法；Middleware主要用于对用户的权限做验证，也是被Route调用，保证用户向特定路径发送请求时，不会有权限管理出错的地方。可以用下图来表示后端的工作流程：

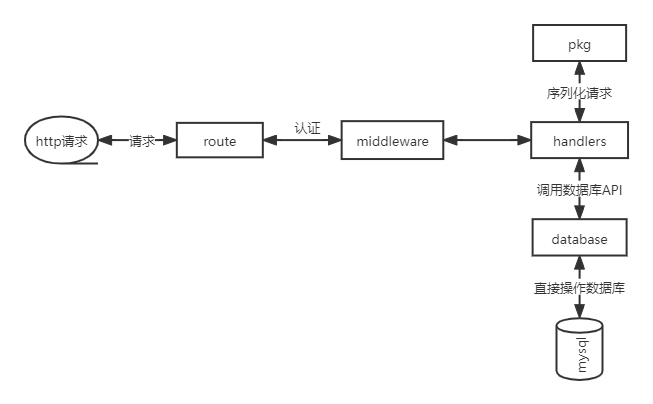


图3-3 后端工作流程

* 音乐推荐功能模块

音乐推荐模块主要分为四个部分：爬取音乐，交付信息给数据库，推荐音乐和与后端通信。这四个部分都用python实现，交付信息用python的sqlalchemy和pymysql库。推荐音乐需要单独创建一个AI项目，这个项目还要负责与后端进行通信。程序结构图如下：

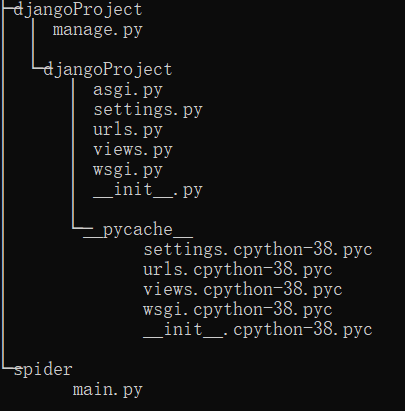


图3-4 音乐推荐程序结构

### 3.1.2 接口设计

由于上述功能模块的设计中，后端需要与前端和音乐推荐系统进行通信，因此后端要给出前端使用的接口，音乐推荐系统需要给出后端使用的接口。

1. 后端接口
2. 检查连通性：status为true说明连接正确 目前，所有的正确返回码都为200，错误码为400(Bad Requets)。

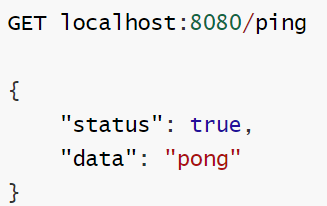


图 3-5 检查连通性接口

1. 注册登录
2. 注册：ID一律用用户邮箱代替。



图3-6 注册接口

1. 登录

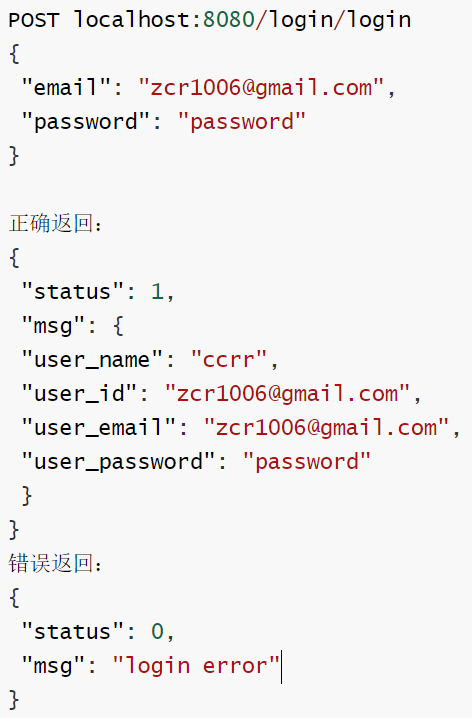


图3-7 登录接口

1. 改密码



图3-8 改密码接口

1. 心情选择
2. 获得所有心情



图3-9 获得所有心情接口

1. 设置心情

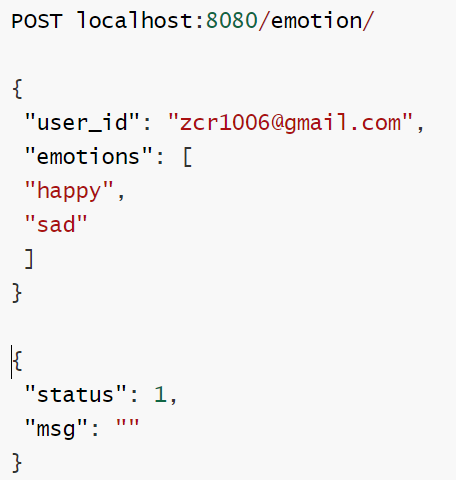


图3-10 设置心情接口

1. 音乐推荐
2. 获取喜爱音乐列表



图3-11 获得喜爱音乐

1. 喜欢一首音乐



图3-12 喜欢一首音乐

1. 删除一首喜欢音乐



图3-13 删除一首喜欢音乐

1. 获得推荐音乐

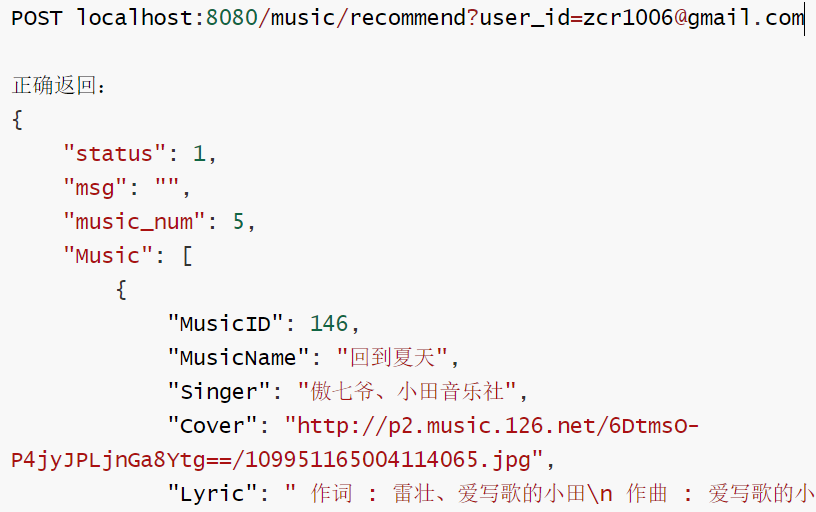


图3-14 获得推荐音乐

1. 获取一首音乐的详细信息



图3-15 获得音乐详细信息

1. 日记
2. 展示用户所有日记



图3-16 用户所有日记

1. 按照tag搜索日记



图3-17 按tag搜索日记

1. 创建日记

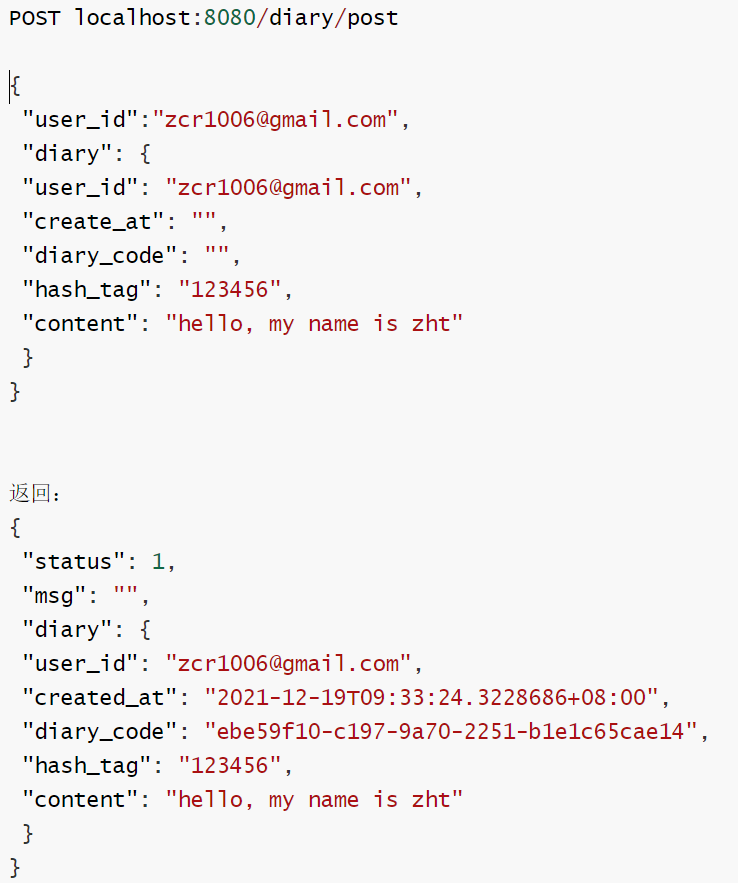


图3-18 创建日记

1. 删除日记



图3-19 删除日记

1. 音乐推荐接口
2. 获取一首音乐的具体信息

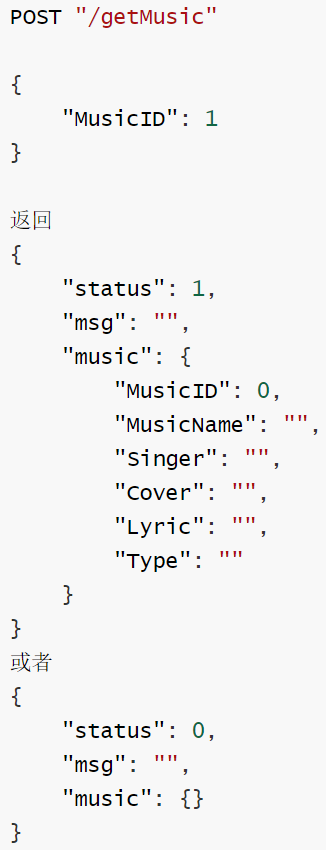
****

图3-20 获取具体信息

1. 获取一首推荐音乐

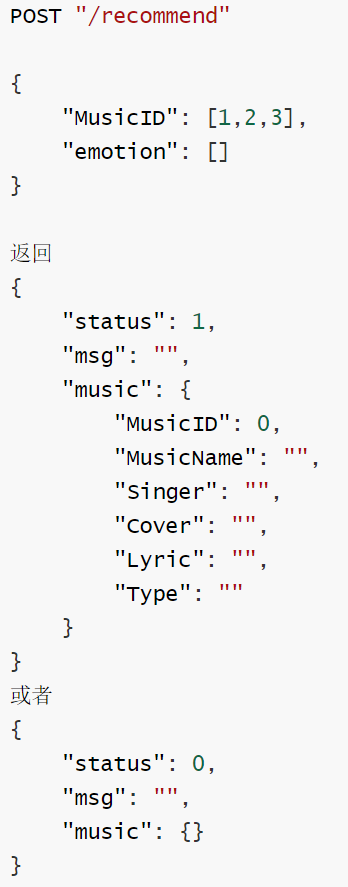


图3-21 推荐音乐

### 3.1.3 整体系统结构

项目整体系统结构如图3-16所示。

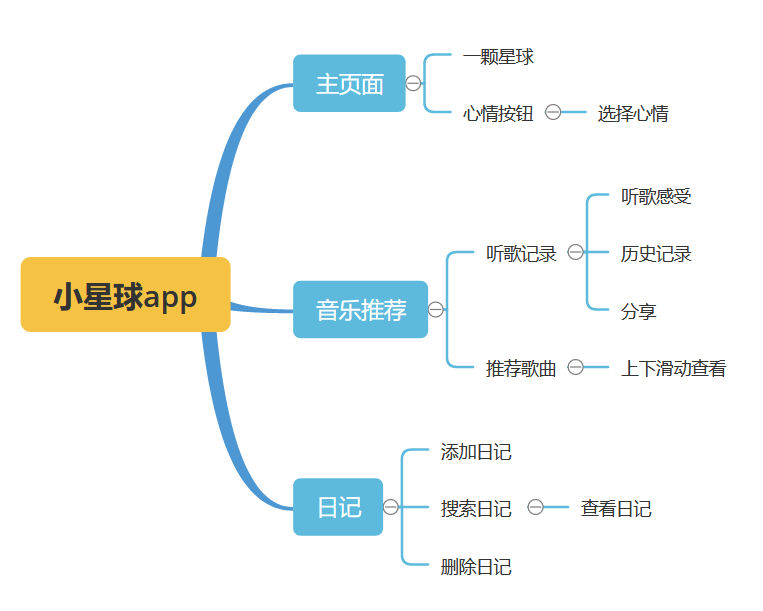


图3-22 整体系统结构图

## 3.2 关键数据结构定义

### 3.2.1 后端数据结构

后端的数据结构设计主要分为两个部分：HTTP通信数据结构以及操作数据库的数据结构。

* HTTP通信数据结构

HTTP通信数据结构有两种类型，一是接收前端发送来的请求，并将结果返回给前端，这样的一次返回就需要定义两个数据结构。同时，后端还要向AI请求推荐音乐内容，因此也要设计推荐音乐的请求结构体和接收信息的结构体。

由于后端定义的HTTP通信数据结构较多，并且定义思路大致相同，这里仅展示日记模块的数据结构。

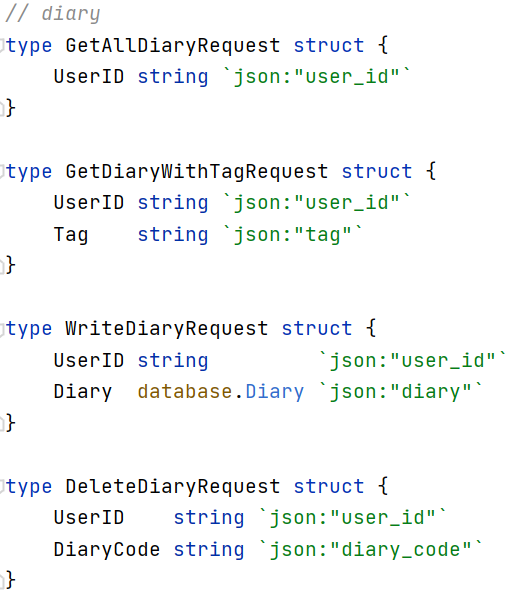


图3-23 日记模块

同时，每个请求也对应着一个返回值：



图3-24 返回值

* 调用数据库数据结构

项目使用mysql作为存储APP所有数据内容的数据库。数据库主要存储四个内容：日记信息，用户心情信息，音乐信息。每一个数据结构对应了mysql数据库的一张表。

其中日记信息表结构设计如下，column字段表示的即是对应列名。



图3-25 日记信息

用户心情定义如下所示：

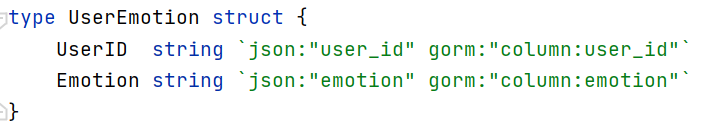


图3-26 用户心情

音乐数据定义如下图所示：



图3-27 音乐

用户基本信息定义如下图所示：

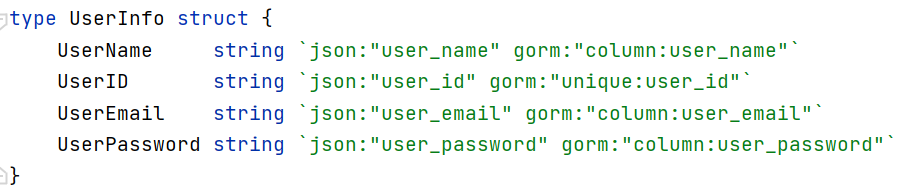


图3-28 用户心情

用户音乐定义：

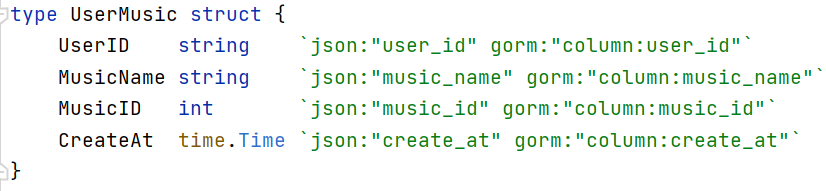


图3-29 用户音乐

### 3.2.2 客户端数据结构

1. UserInfo：保存用户相关的信息，便于交互

**struct** UserInfo{

**var** userName:String

**var** email:String

**var** password:String

}

1. Emotion：保存用户当前的心情

**struct** Emotion:Codable{

**var** user\_id:String

**var** emotions:[String]

}

1. Music：保存一首音乐相关的所有信息，包括歌曲ID、歌名、歌手、封面、歌词以及歌曲的类型

**struct** Music:Codable{

**var** MusicID:Int

**var** MusicName:String

**var** Singer:String

**var** Cover:String

**var** Lyric:String

**var** \_Type:String

}

1. PostMusic：用于喜欢某一首歌对后端发起网络请求时的数据结构

**struct** PostMusic:Codable{

**var** user\_id:String

**var** music:Music

}

1. DeleteMusic：用于取消喜欢某一首歌对后端发起网络请求时的数据结构

**struct** DeleteMusic:Codable{

**var** user\_id:String

**var** music\_code:Int

}

1. FavoriteMusic：用于保存所有用户喜爱歌曲的数据结构

**struct** FavoriteMusic:Codable{

**var** music\_name:String

**var** music\_id:Int

}

1. Diary：用于保存用户信息以及一条日记相关的数据结构

**struct** Diary:Codable{

**var** user\_id:String

**struct** diary\_struct:Codable{

**var** user\_id:String

**var** create\_at:String = ""

**var** diary\_code:String = ""

**var** hash\_tag:String

**var** content:String

}

**var** diary:diary\_struct

}

### 3.2.3音乐推荐系统数据结构：

1. 每首音乐具体信息的数据结构

class Music(base):

\_\_tablename\_\_ = 'Music' # 表名

MusicID = Column(Integer, primary\_key=True)

MusicName = Column(String(255))

Singer = Column(String(255))

Cover = Column(String(255))

Lyric = Column(Text(65535))

Type = Column(String(255))

1. 从后端获得推荐音乐相关信息的数据结构：

id = r['MusicID']

emotion = r['emotion']

## 3.3 关键算法设计

在此比较关键的就是如何进行音乐推荐。

音乐推荐算法，就是针对音乐自身的内容特征以及用户的听歌行为，为广大用户提供可能符合他们兴趣爱好的歌曲的算法。而基于大数据的个性化音乐推荐算法，能够通过历史数据，别的用户的历史数据分析出潜在的喜好相似性，为用户更准确地挖掘出潜在的喜欢的音乐。主要通过以下几点实现：

1. **协同过滤**，寻找爱好兴趣相似的用户，然后给一方推荐给另一方的歌单上的歌曲；
2. **NLP**，爬取网上关于音乐的信息，并分析用户讨论的具体的艺术家或者歌曲内容，比如用的什么形容词或者哪些词用的最多，经过统计分析得到文化向量和高频短语，通过文化向量与高频短语，就能大概率找到风格近似的音乐。
3. **心情**，通过分析歌词文本得到一首歌的情感倾向，推荐符合心情的音乐。

算法流程图如图3-18所示。

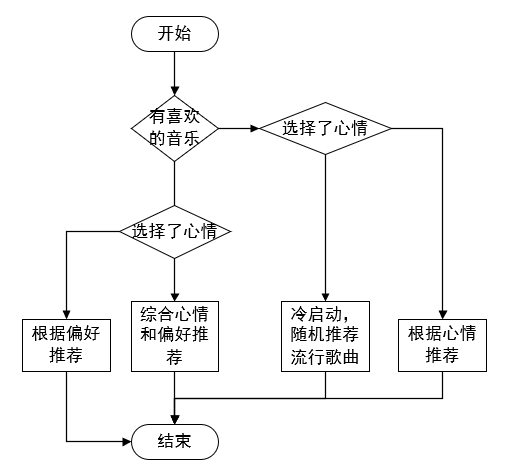


图3-30 推荐音乐流程图

## 3.4 数据管理说明

### 3.4.1 客户端数据管理

在APP中，有部分数据需要本地化保存：用户名、邮箱、用户头像。本地化保存通过UserDefaults来实现。UserDefaults是iOS开发本地化保存数据的一种方式，用UserDefaults保存的数据会保存在沙盒Library/Preferences目录下，文件格式是plist。UserDefaults支持的数据格式有：NSNumber（Integer、Float、Double），NSString，NSDate，NSArray，NSDictionary，BOOL，NSData类型等等。这个类使用起来也很简单，使用它的单例UserDefaults.standard即可，示例代码如下：

//保存String类型的用户名

UserDefaults.standard.set(userName, forKey: "userName")

//获取Key为"userName"的String数据

UserDefaults.standard.string(forKey: "userName")

### 3.4.2 后端和音乐推荐系统数据管理

项目为了便于数据的存取，使用最常用的 Mysql数据库做数据存储。考虑到后端数据关系的复杂性，直接使用sql语句对数据库进行操作虽然效率高，但容易出错，故在项目中使用了对象关系映射器技术（Object Relational Mappers，ORM），开源gorm框架对于不同的关系型数据库的支持都很不错，故选用gorm来管理数据库。

* 数据库

Mysql MySQL 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

由于后端代码需要在服务器上运行，数据库部分也应放在云端。项目后期会将数据库迁移到华为云的RDS云数据库中，以保证数据的安全性，可靠性，以及随时都可以访问该数据。

* 数据库管理框架

gorm是Golang的ORM框架，对众多关系型数据库有良好的支持，包括支持关联(包含一个，包含多个，属于，多对多，多种包含)，回调，预加载，事务，复合主键，自动迁移，日志等功能。同时，使用gorm社区也十分活跃，遇到框架的bug也能提出issue及时解决。

# 4 实现与测试

## 4.1实现环境与代码管理

### 4.1.1 实现环境

* 音乐推荐系统：

IDEA：pycharm2021.3

语言：python

* 客户端：

硬件环境：MacBook Pro 2.3 GHz Quad-Core Intel Core i5

操作系统环境：macOS BigSur 11.3.1

真机运行环境：iPhoneXR iOS15

IDE：XCode 13.0

编程语言：Swift

* 后端

语言: GoLang

IDEA：Goland 2020.3.2

接口调试工具：Postman

真机运行环境：Windows 11 Intel i7-8565U CPU @ 1.80GHz 1.99 GHz

服务器环境：Ubuntu 18.04

数据库版本：Mysql 5.7.36

### 4.1.2 代码管理

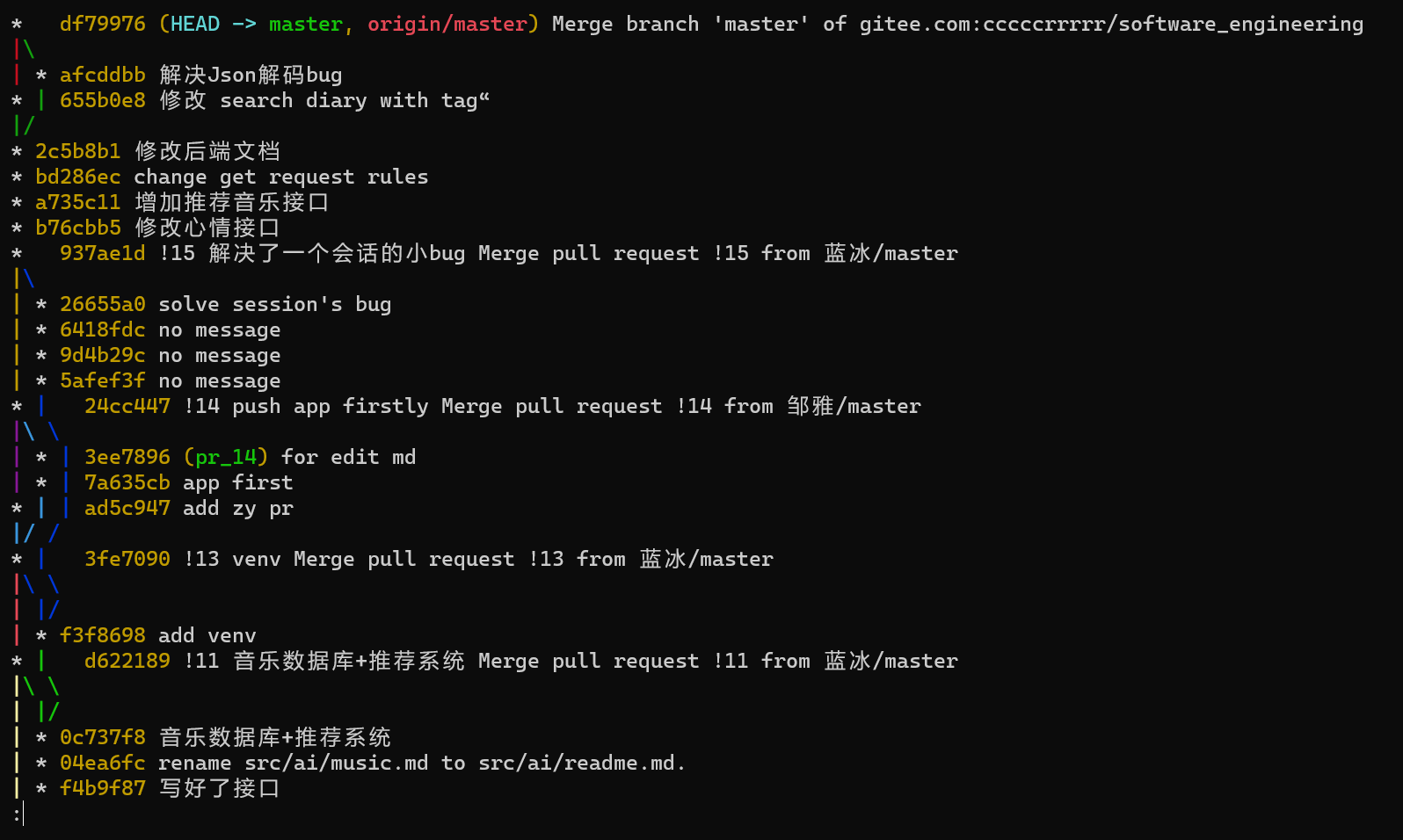


图4-1 代码管理

本项目使用gitee进行代码管理和版本控制，项目成员会fork源仓库，在完成部分功能后提出pull Request到主仓库，仓库所有者在完成审查之后会合并该代码。如上图所示，是该项目从立项开始，到最后代码完成的提交、合并记录图形化展示，在gitee记录中，我们的项目共有76次commit记录，17个pull request请求，接收审查最终合并的个数为13个。

## 4.2 关键函数说明

### 4.2.1 客户端关键函数

APP中界面都是通过类来实现的，而与后端的交互通过一系列函数来实现，在这里将这些函数列出如下：

1. **func** sendMoodToNetwork(strArray:[String])

将心情选择的结果发送给后端

1. **func** confirmToRegster()

将用户注册的信息——用户名、邮箱和密码发送给后端

1. **func** confirmToLogin()

将用户登录的信息——邮箱和密码发送给后端

1. **func** confirmToChangePassword()

将用户修改密码的信息——邮箱和新密码发送给后端

1. **func** getAllRecommendMusic()

通过用户ID获得推荐的音乐

1. **func** getAllFavorite()

通过用户ID获得用户所有喜欢的音乐

1. **func** sendDiary()

发送用户ID、当前日记和日记tag给后端保存至数据库

1. **func** searchForTag()

发送用户ID、日记tag来查找相关的日记

1. **func** getAllDiaries()

发送用户ID来得到该用户的所有日记

1. **func** getInfo(id:Int)

发送音乐ID来获得音乐的详细信息，包括歌名、歌手和歌词

1. **func** sendFavorite()

发送用户ID和音乐ID表示当前该用户喜欢了该歌曲

1. **func** sendNotFavorite()

发送用户ID和音乐ID表示当前该用户取消了喜欢该歌曲

这些接口函数本质上都是使用了网络请求来进行的，向对应的后端URL发起get、post或者delete请求，再解析返回的结果来得到相应的结果并返回给APP来使用或展示。

### 4.2.2 后端关键函数

* 数据库接口

1. func GetAllDiaryWithID(id string) (\*[]Diary, error)

根据用户ID查询所有日记。

1. func GetAllDiaryWithIDAndTag(id string, tag string) (\*[]Diary, error)

根据用户ID返回所有特定标签的日记。

1. func InsertDiaryWithID(id string, tag string, content string) (\*Diary, error)

生成一个唯一的ID作为日记的唯一标识符，加入一篇日记。

1. func DeleteDiaryWithID(id string, code string) error

根据日记的unique ID删除一篇日记。

1. func GetEmotionByID(id string) ([]string, error)

根据用户ID获取用户当前的所有心情。

1. func UpdateEmotion(id string, emotions []string) error

更新用户的所有心情。该函数逻辑是把用户的之前的心情全部删除，再添加所有当前的心情字段。

1. func AddMusicListByUserID(id string, music UserMusic) ([]UserMusic, error)

向数据中插入用户喜欢的一首曲子。考虑到项目运行到后期，爬取的音乐数量和越来越大，因此后端数据库中并不存储音乐的具体信息，只存储一个ID。

1. func DeleteMusicListByUserID(id string, musicID int) (\*UserMusic, error)

删除用户中的一首喜爱曲子。

1. func GetMusicListByUserID(id string) ([]UserMusic, error)

通过用户ID获取该用户喜爱的所有音乐。

1. func ChangePasswordByEmail(email string, password string) (bool, string)

通过邮件名更改密码。

1. func GetUserByEmail(email string) \*UserInfo

通过邮箱名获取用户的所有信息。

1. func AddUser(name string, email string, password string) \*UserInfo

将一条用户信息添加到数据库中。

* 用户注册登录模块

1. func Register(ctx \*gin.Context)

接收APP端发送的注册请求，如果邮箱没有注册，返回注册成功的信息，否则返回注册失败信息。

1. func Login(ctx \*gin.Context)

接收APP端发送的登录请求，如果用户ID和密码正确，返回登陆成功信息，否则返回登陆失败信息。

1. func ChangePassword(ctx \*gin.Context)

接收APP端发送的修改密码请求。这一接口实现的逻辑较为简单，只有用户ID正确给出，直接更新其密码。有关二次输入密码校验的工作由APP部分完成。

* 日记模块

1. func GetDiary(ctx \*gin.Context)

接收APP发送的获取日记请求，首先会检查这一用户是否存在，如果不存在会返回错误提示信息。

1. func DeleteDiary(ctx \*gin.Context)

接收APP发送的删除日记请求，请求中只会有用户ID和日记ID，如果与数据库中的数据相匹配，就成功删除，并返回正确信息。

1. func GetDiaryWithTag(ctx \*gin.Context)

接收APP发送的获取日记和Tag的请求，同样也会先检查这一用户是否存在，不存在直接返回错误值。如果搜索的日记结果为空，不会有报错产生。

1. func PostDiary(ctx \*gin.Context)

接收APP发送的创建一篇日记的请求，后端会生成唯一的ID表示一篇日记，并调用下层的数据库接口将日记内容存入数据库中。

* 心情模块

1. func GetEmotion(ctx \*gin.Context)

接收APP发送的获取心情请求，返回当前用户的所有心情。

1. func SetEmotion(ctx \*gin.Context)

接收APP发送的设置心情请求，调用下层数据库接口，更新存储用户的心情。

* 音乐推荐模块

1. func GetMusicList(ctx \*gin.Context)

接收APP发送的获取用户喜爱音乐列表请求，调用下层接口，获取喜爱音乐名称和序号，一并返回给APP。

1. func LikeMusic(ctx \*gin.Context)

接收APP发送的喜爱一首音乐的请求，在数据库中添加喜爱音乐信息。

1. func DeleteLikeMusic(ctx \*gin.Context)

接收APP发送的取消喜爱一首音乐的请求，在数据库中删除喜爱音乐信息。

1. func GetMusicWithDetail(ctx \*gin.Context)

接收APP发送的获取一首歌曲的详细信息，转而向AI端发送请求，获得歌曲的详细信息后返回给APP端。

1. func RecommendMusic(ctx \*gin.Context)

接收APP发送的推荐请求，后端会从数据库中取出用户喜爱音乐的列表和当前的心情状态，向AI端发起请求，并将获得结果返回给APP端。

### 4.2.3 音乐推荐关键函数

1. 爬取音乐：
2. 爬取一个歌单里面的歌

def get\_url(url, Session)

1. 保存音乐信息到数据库中

def savaMusicToDB(id, name, img, singer, lyric, type)

1. 音乐推荐系统：
2. 推荐音乐有关函数

def word2vecModel(con\_list, model\_path='my.model', word2vec=None)

参数解释：

1. sentences：可以是一个List，对于大语料集，建议使用BrownCorpus,Text8Corpus或·ineSentence构建。
2. sg： 用于设置训练算法，默认为0，对应CBOW算法；sg=1则采用skip-gram算法。
3. size：是指输出的词的向量维数，默认为100。大的size需要更多的训练数据,但是效果会更好. 推荐值为几十到几百。
4. window：为训练的窗口大小，8表示每个词考虑前8个词与后8个词（实际代码中还有一个随机选窗口的过程，窗口大小<=5)，默认值为5。
5. 获得音乐信息有关函数：根据request中的json信息返回音乐信息，具体返回值与接口定义一致。

def getMusic(request)

## 4.3 测试计划和测试用例

### 4.3.1 常用测试方法

测试分为白盒测试和黑盒测试。白盒测试的常见方法包括：逻辑覆盖法、路径测试法。黑盒测试的内容包括Alpha/Beta测试、菜单、帮助测试、发行测试、回归测试、效能测试、负载/压力测试。黑盒测试法是根据被测程序功能来进行测试，也称为功能测试，有4种常用技术：等价分类法、边界值分析法、错误猜测法、因果图法。测试计划分为对流程的测试和对功能的测试。

### 4.3.2 APP测试

APP的测试主要是看界面实现的效果是否达到了最初的UI设计，以及和后端交互后返回的结果。

1. 界面设计效果以及动画效果



图4-2 主界面完成测试效果

图4-3 4-4 选择心情界面动画效果测试

图4-5 4-6 注册成功、注册失败测试

图4-7 4-8 登录成功、登录失败测试



图4-9 修改密码测试

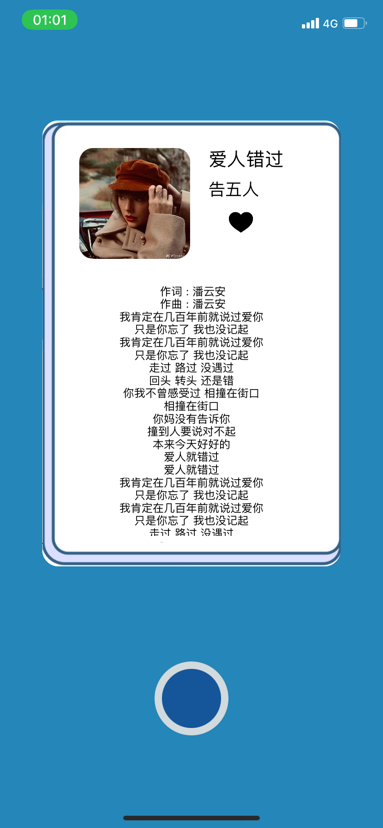
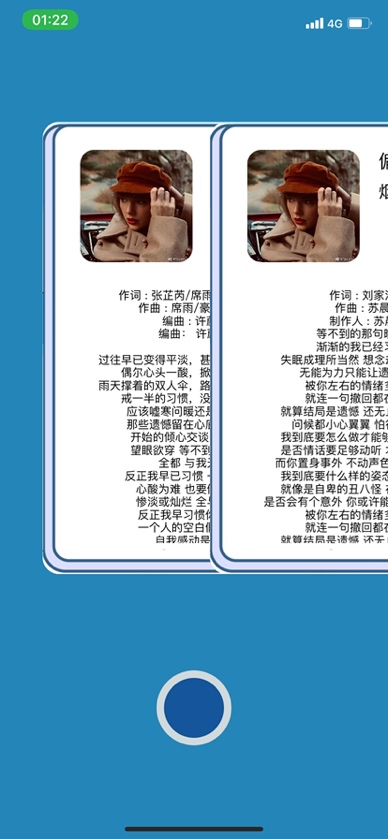
 

图4-10 4-11 获取推荐音乐、推荐音乐滑动测试

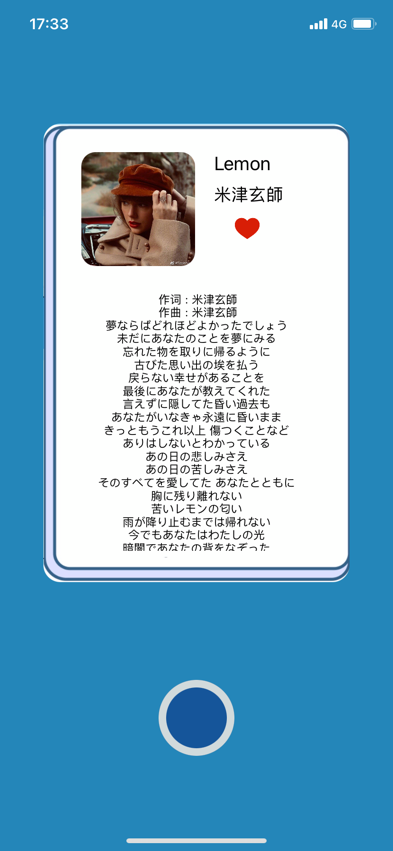
 

图4-12 4-13 喜欢一首音乐、获取所有喜欢的音乐列表测试



图4-14 发送日记测试

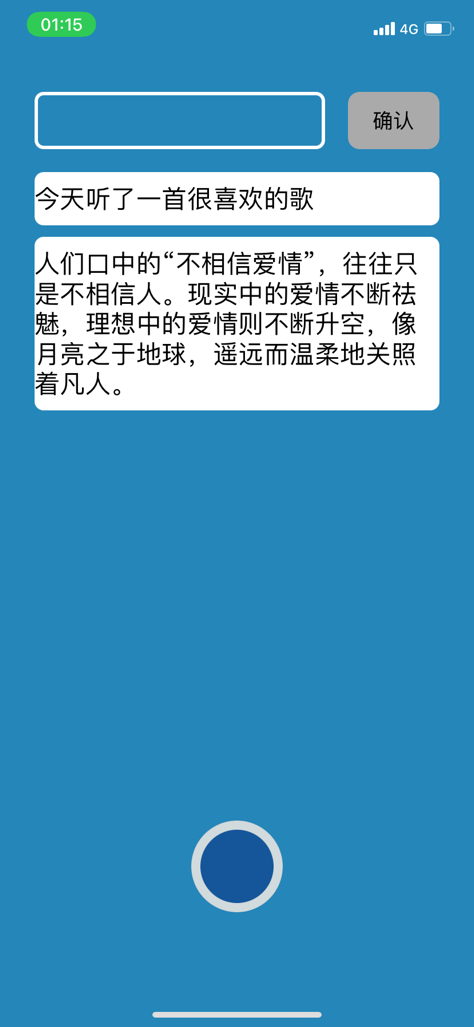
 

图4-15 4-16 查看全部日记、按tag筛选日记测试

## 4.4 结果分析

从界面显示、操作流程来进行分析，可以看到所有界面都能正常展示，所有动画都能流畅使用。从功能角度来看，与后端交互正常，能在不同的接口中正确提供相应的接口数据，同时也能正常解析数据进行使用。

# 5 总结

## 5.1 用户反馈

### 5.1.1 用户反馈图文

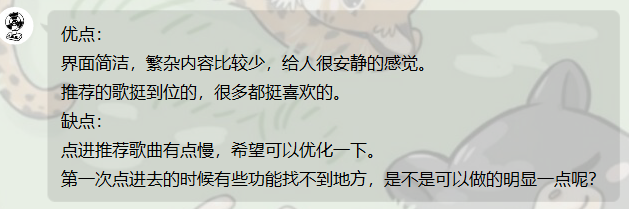


图5-1 用户反馈1

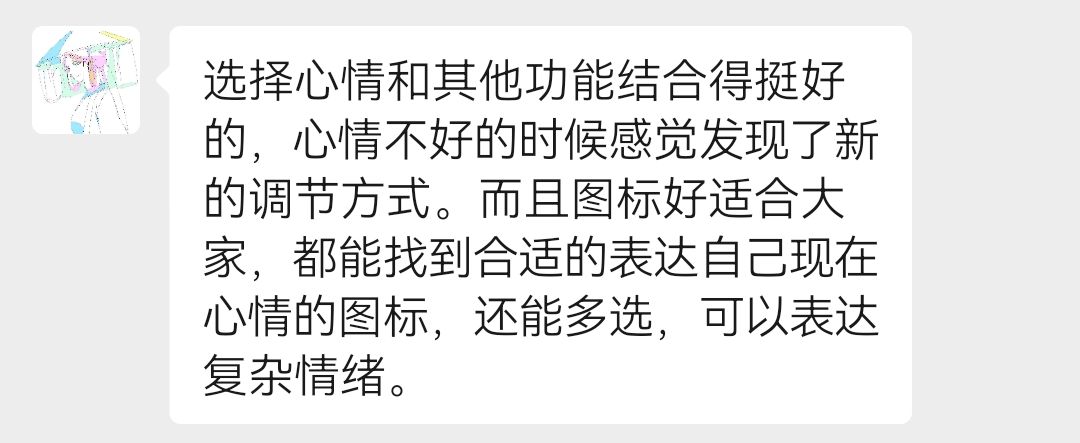


图5-2 用户反馈2

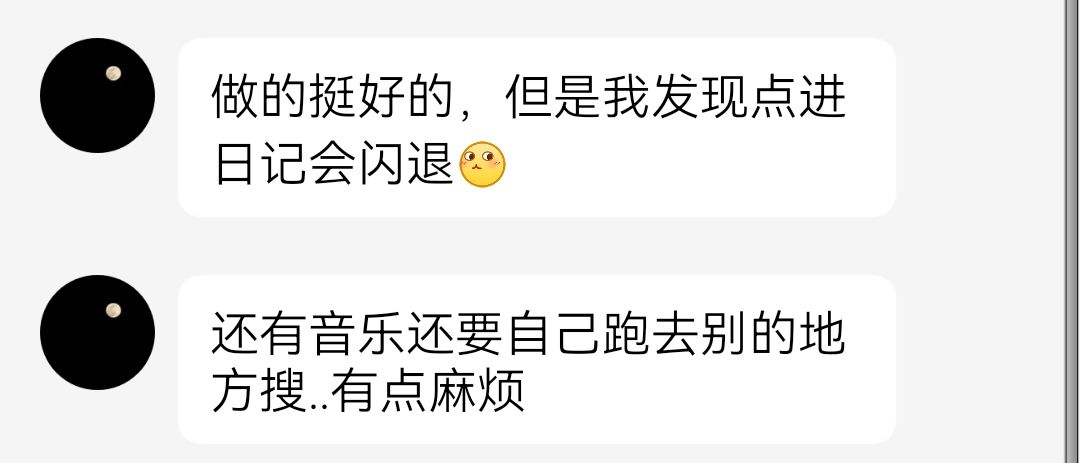


图5-3 用户反馈3

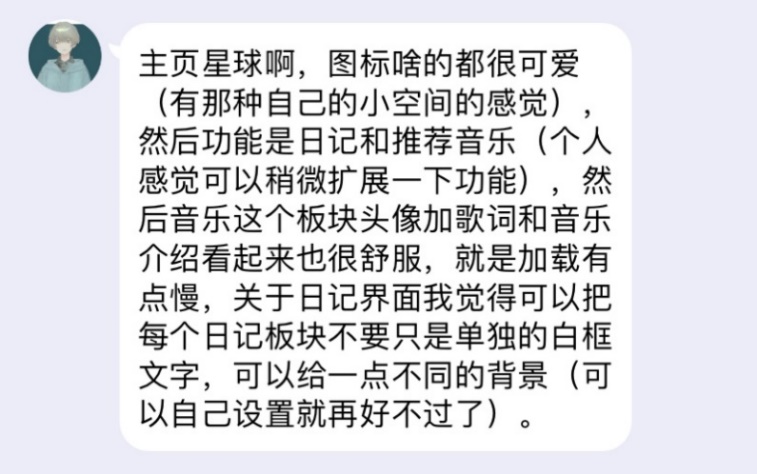


图5-4 用户反馈4

### 5.1.2 用户反馈总结

* 优点：

1. 界面简洁，繁杂内容比较少，给人很安静的感觉。
2. 推荐的歌挺到位的，很多都挺喜欢的。
3. 日记的tag功能很方便，可以把日记分类之后随时查看不同心情的日记。
4. 选择心情和其他功能结合得挺好的，心情不好的时候感觉发现了新的调节方式。
5. 选择心情的图标好适合大家，都能找到合适的表达自己现在心情的图标。
6. 心情还能多选，可以表达复杂情绪。

* 建议：

1. 点进推荐歌曲有点慢，希望可以优化一下。
2. 第一次点进去的时候有些功能找不到地方，是不是可以做的明显一点呢？
3. 发现一个小bug，点进日记有时候会闪退。
4. 日记是不是可以结合天气啥的呢，可不可以加点贴纸信纸什么的，做成手账那个样子会很不错。
5. 音乐可以做成在APP里面播放的吗，还要自己搜有点麻烦。

## 5.2 全文总结

* APP 客户端主要工作如下：

1. 使用 Swift ，搭建各个页面模块，完成 iOS 客户端应用
2. 通过网络请求与后端进行连接，完成数据交换与数据存储

* 后端系统主要工作如下：

1. 使用gin框架搭建项目后端，响应客户端发来的请求。
2. 使用mysql存储用户数据，并用gorm框架处理数据库数据。
3. 配置云服务器环境，将代码迁移到服务器上运行。

* 音乐推荐系统主要工作如下：

1. 使用python，定期爬取近期热度较高的各种音乐放入数据库以供后端和推荐音乐使用。
2. 使用django框架搭建小型web服务器框架用于与后端通信。
3. 使用LightGBM模型对推荐音乐模型进行训练，另外通过文本向量化对歌曲情感色彩分类。
4. 对不同用户使用相似矩阵进行相似度分析，描绘用户画像。

# 6 体会

* 邹雅

这是第一次和同学组队从头做一个全新的项目。之前做项目基本上都是基于一个现有的想法或者灵感，然后照着设计图在写。所以一开始在立项的时候，大家都在努力的发挥创意；而在开发初期，大家画的原型图并不能够像真正的设计图那样按着写UI即可。在界面搭建的中间，我花了比较多的时间找小星球的logo、按钮icon，以及调整页面布局等等。最终得到的界面也算是符合我们自己的审美了。

从零开始搭建一个APP所需要的工作量比我预期想象得要多得多。从早期先搭建好页面，因为用的是相对布局，没有考虑清楚的情况下调整布局的时候总会出现各种奇怪的现象。接下来写控制逻辑，各种页面的跳转逻辑在需要认真考虑。记得比较清楚就是在写是否喜欢一首歌两种状态分别点击时应该对应怎样的交互逻辑的时候，没有考虑清楚，导致调了很久。然后是完成与后端的连接，之前虽然我也写过一些别的iOS项目，但是和后端连接我是第一次写。一开始我不明白这个连接是什么意思，后来知道了就是通过网络请求来完成的。中间有一个记忆比较深刻的是，我发送get请求，携带的数据在URL中，而后端定义的接口是把数据放在了HTTPBody中；在我和后端的同学都没意识到这一点的时候不停地请求失败找不到原因，卡了挺长时间，后来才恍然大悟。

除了这些bug以外，APP中还比较花费时间的就是大量的细节。比如需要判断当前用户是否登录来决定日记、音乐功能是否可以使用，比如注册、登录这些请求如果失败了应该要做怎么样的页面显示，比如键盘弹起之后回收的逻辑，比如判断两次密码输入是否正确来判断是否可以完成注册……所以这个APP实际写的时间比预期的要长一些。

我们的APP在功能上，可能受限于大家都没有产品经理的经验，所以提出的需求在用户层面上可能不是很够。而且受限于工期时间，很多东西都是做了一个基本吧，没有做到很完善。之后我们还会继续进行优化，比如说完成歌曲封面图片的拉取，比如优化推荐音乐的时延。然后结合初期用户的一些意见，增加一些能增强体验感的功能。等这些问题解决了，我应该会着手把它上架到App Store里。

* 张弘弢

这次项目带给我的是第一次完整的项目开发经历，也是本学期挑战最大的实验。我觉得软件工程中最有意义的地方在于从选题，设计界面，到最后的设计都是一个小组去思考，讨论，修改，最后达成统一。学CS的同学工作一般都会选择开发岗，会依据设计的稿子和pm的逻辑去实现，基本不会要求对产品本身有什么思考。因此，我也觉得这次的软件工程的意义就在于此。

我负责项目的后端部分，后端有两大工作，一是完成供客户端调用的接口，二是配置好数据库以及云服务器环境。这两项工作说起来简单，由于缺少后端开发的经验，在具体实现时总会有琐碎的细节没有考虑到。对于第一项任务，为了能和客户端同学更好的对接，我必须先对所有接口进行本地测试，写好完整的接口文档。由于我对于swift请求的规则不是特别了解，在客户端开发的过程中逐渐发现了许多接口设计不合理的地方。例如，在最开始的时候，我要求所有的GET请求必须带上body，但是IOS端只要带上body程序直接报错，最后我把所有的GET请求逻辑都改成了Query的形式两端才能正确通信。同样的，由于我和AI同学使用的不是同一个语言，无法直接调用她的API，也考虑到后端和AI模块不应该耦合太多，因此额外让AI同学搭建了Web框架以便两个模块的通信。

在配置云服务器的过程中，我体验了三个不同版本的Ubuntu服务器，以及不同的mysql使用方法，例如在Ubuntu 20.04LTS中，无法用root用户直接登入mysql，必须先配置无密码登录或者通过debain用户才能登陆进去。同时，由于需要存储很多中文字符，如果使用默认的创表方式，在插入歌词时会出现编码报错，需要将对应的数据库编码更新为utf8mb4才可以正确存储数据。最后，由于Ubuntu包管理器并不能获取最新版本的某些程序，例如GoLang，通过apt-get install获取的go版本很低，无法正确运行我的程序。最后我到官方网站下载最新版本Go语言完成环境配置后，问题才得以解决。

总而言之，软件工程课程比我之前的工作量预想的大很多，同时也有趣很多。我体验到了如何与队友沟通去完善这个项目的设计逻辑，如何将一个简单的，不成熟的idea变为一个实实在在的项目。

最后感谢老师对我们小组项目的关注，督促了项目的完成。我们小组也认真思考了老师提出的问题，进一步完善改进了项目。非常感谢老师的付出。

* 蓝冰瑛

这次开发是算比较完整的开发了，从立项到最后写出来都是大家亲力亲为的，在老师和同学们的身上都学到了很多东西。在机器学习的课程上学到的知识能够马上应用到实际开发中感觉还是很奇妙的。

在开发中也遇到很多困难。首先以前没有学过用python和数据库进行交互，于是自己走了挺多弯路，学习了比较原始的Mysql语句并且用Mysql语句来创建表，然后和后端的同学交流之后发现可以用ORM不用这么麻烦……AI也是用python写的但是一开始没有明确与后端通信的方式并且没有在python上搭建过web服务器，同学推荐django之后由于找不到合适的学习材料也导致进度停滞了一两天。和后端同学确定需求以及调试接口也花费了较长的时间。并且服务器上没有对应的python环境，配置环境中因为过于老旧的python版本导致无法更新，卸载老版本时不知道出了什么差错把数据库的内容也弄丢了，最后和同学商量之后换了一台服务器才解决问题，实际上也不算解决了问题因为并没有弄明白为什么会误把数据库内容删掉，但也算增加了debug经验。

不过最后做出成果还是很有成就感的。在这次软工项目里不仅学到了除了码代码以外的知识，扩宽了对于今后开发项目的长远见识，也获得了与同学们合作的经验。