

**软件架构课程实验报告**

题 目  **跑操管理系统 —— Destination**

**软 件 学 院** 院（系） **软 件 工 程** 专业

学 号  **71116、71Y16、71117**

学生姓名  **龚呈 奉捷 金杰 刘瑞琦**

**沈梦烨 曲正锟 孙浩然 张治隆**

指导教师  **廖 力**

起止日期 **2018.3.7 - 2018.4.15**

设计地点 **南 京**

目 录

[第一章 课题概述 3](#_Toc509392184)

[1.1 选题背景及意义 3](#_Toc509392185)

[1.2 实验的主要工作 3](#_Toc509392186)

[1.3 实验报告的组织结构 3](#_Toc509392187)

[第二章 需求分析 5](#_Toc509392188)

[2.1 系统总体需求 5](#_Toc509392189)

[2.1.1 学生模块需求 5](#_Toc509392190)

[2.1.2 老师模块需求 9](#_Toc509392191)

[2.2 本章小结 1](#_Toc509392193)2

[第三章 系统设计 1](#_Toc509392194)3

[3.1 总体系统功能设计 1](#_Toc509392195)3

[3.1.1 学生模块 1](#_Toc509392195)3

[3.1.2 教师模块 1](#_Toc509392195)4

[3.1.3 服务端模块 1](#_Toc509392195)4

[3.2 体系结构设计 1](#_Toc509392196)4

[3.3 接口设计 1](#_Toc509392197)5

[3.4 本章小结 1](#_Toc509392198)8

[第四章 系统的模块设计与实现 19](#_Toc509392199)

[4.1 数据库的设计 1](#_Toc509392200)9

[4.2 学生模块的实现 2](#_Toc509392201)2

[4.2.1 学生互动模块 2](#_Toc509392201)2

[4.2.2 学生跑步模块 2](#_Toc509392201)4

[4.2.3 天气提醒 2](#_Toc509392201)6

[4.2.4 学生跑操记录查询模块 2](#_Toc509392201)7

[4.3 老师模块的实现 3](#_Toc509392202)1

[4.4 服务端模块的实现 3](#_Toc509392203)5

[4.5 本章小结 3](#_Toc509392204)9

[第五章 系统测试 4](#_Toc509392205)0

[5.1 环境 4](#_Toc509392206)1

[5.2 各模块测试 4](#_Toc509392207)2

[5.2.1 登录注册 4](#_Toc509392207)2

[5.2.2 现有功能 4](#_Toc509392207)3

[第六章 实验总结 4](#_Toc509392208)4

第一章 课题概述

# 1.1 选题背景及意义

软件系统设计与体系结构小组作业，选题通过小组成员投票确定。

本早操管理系统旨在为同学们设计一个更加贴近东大校园生活的运动App，为班主任与体育老师提供更加方便快捷的信息发布与查询通道。

此项目可丰富组员Android/Java开发经验，提高编程能力与沟通交流能力，并可作为课后的长期项目。

# 1.2 实验的主要工作

本文就跑操管理系统，分析界面、网络、数据库、具体功能等，讲解跑操管理系统的设计与实现。

1.跑操管理系统的需求分析。从三大角色介绍系统的业务需求，并说明接口需求以及该接口的目的。

2.跑操管理系统的设计。先描述系统的总体模块设计，然后从三个模块入手，分别介绍各个模块的设计。

3.跑操管理系统的实现。从各个模块入手，介绍该模块的实现，包括用到的技术等。使用MySQL完成对数据的存储，查询等，用json传输数据。使用struts 2技术，完成对系统后端的开发。使用XML完成前端页面的展示。

4.跑操管理系统的运行。根据不同的角色，按照各个角色的功能，介绍运行的结果。

# 1.3 实验报告的组织结构

本文主要分为以下几个部分：

第一章是绪论，主要介绍跑操管理系统研究意义，研究背景，主要工作，章节安排等。

第二章是跑操管理系统的需求分析，按总分总结构介绍系统的需求。

第三章是跑操管理系统的设计，主要是对各个模块设计的分析。

第四章是跑操管理系统的实现，先描述数据库的设计，接着对各个模块的实现进行介绍，主要包括类设计和其中的主要技术。

第五章为系统测试，主要是职跑操管理系统整个流程步骤的展示和执行结果。**第二章 需求分析**

# 2.1 系统总体需求

本课题的目的是为同学们开发一个兼具校内规定跑操与日常体育锻炼功能的跑操管理系统，为班主任与体育老师提供更加方便快捷的信息发布与查询通道。包括学生早操日期和时间设定，运动范围设定，个性化路线定制等功能，支持各类人员的不同运动量标准，为学生、班主任和体育老师出具各类统计报表。

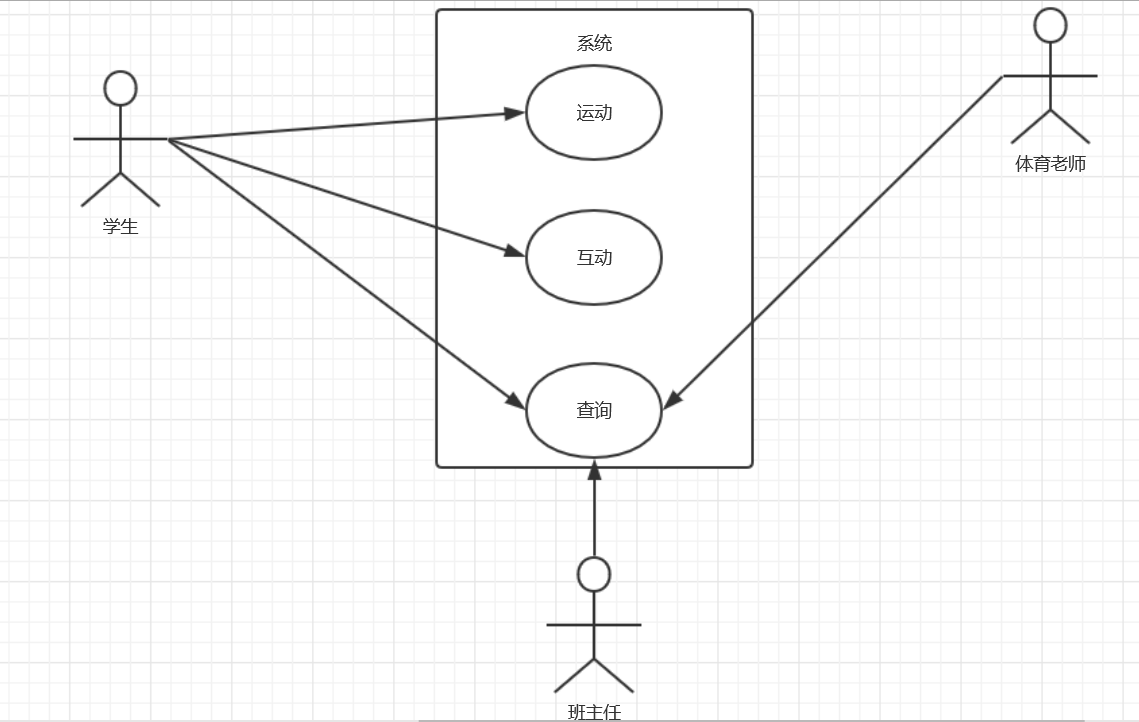
本系统的总体use case图如下：

图2-1系统用例图

# 2.1.1 学生模块需求

在该模块，涉及的主要角色是学生角色。

具体用例图如下：

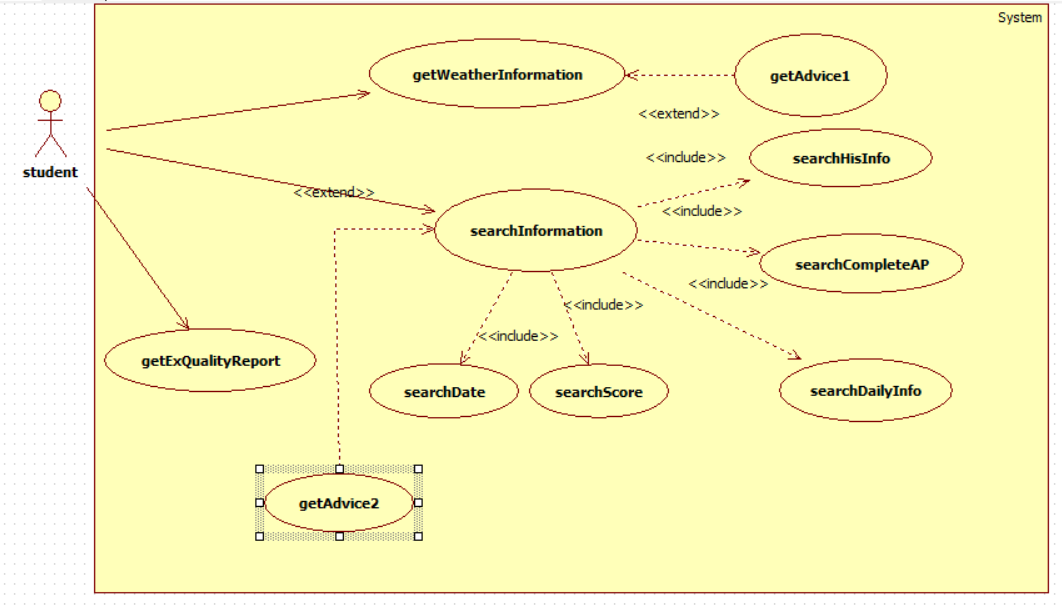


图2-2学生用例图

对于学生角色，具体的用例说明如下表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 角色 | 类别 | 用例 | 功能 | 实现函数 |
| 学生 | Weather | GetWeatherInfo | 获取当天实时天气信息 |  |
| GetAdvice1  （跑操） | 根据天气信息，给予相应的跑操建议 |  |
| InfomationSearch  (related to exercise information) | SearchHisInfo | 查询自己过去跑操的详细数据 |  |
| SearchCompletementAP | 查看早晚操的完成情况 |  |
| SearchDailyInfo | 查看当天的早晚操情况 |  |
| SearchScore | 查询已经获得的分数/未获得的分数 |  |
| SearchDate | 查询已经经过的日期/剩余日期 |  |
| GetAdvice2  （日程） | 根据学生跑操的完成度，结合日程，天气给予日程规划 |  |
| Report(Exercise quality) | GetExQualityReport | 分析学生的锻炼质量，给予反馈 |  |

表2-1学生用例图说明

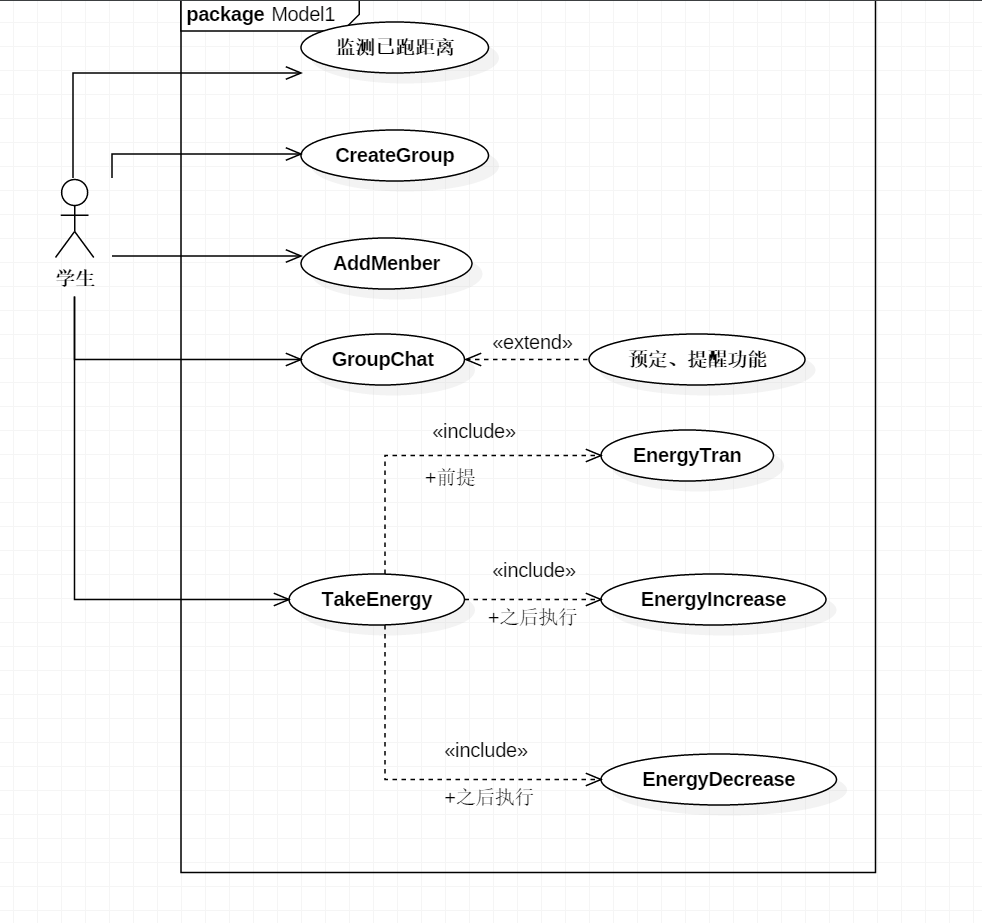


图2-3学生用例图

|  |  |  |
| --- | --- | --- |
| 角色 | 用例 | 功能 |
| 学生 | 监测已跑距离 | 在跑操时能够实时检测已跑距离，速度，并根据个人身体素质给予跑速建议，最后在时间将要到达时，给予提醒。 |
| CreateGroup | 创建一个群组，能够在里面聊天，或者预约高邮一起跑操 |
| AddMenber | 群组添加群成员 |
| GroupChat | 在群组里进行通信，与群成员进行文字但不限于文字的聊天（看后续开发情况） |
| 预定、提醒功能 | 约好友跑操、并在app中创建一个提醒 |
| TakeEnergy | 好友互动功能，收取好友通过锻炼而获取的能量，为自己的树+1s，从而提升自己的排名 |
| EnergyTran | 收取能量的前提，根据运动量来进行相应转换，生成一定数量的能量 |
| EnergyIncrease | 收取能量之后的操作，增加收取能量用户的能量数值 |
| EnergyDecrease | 收取能量之后的操作，减少被收取用户的尚未收取的所剩能量数值 |

表2-2学生用例图说明

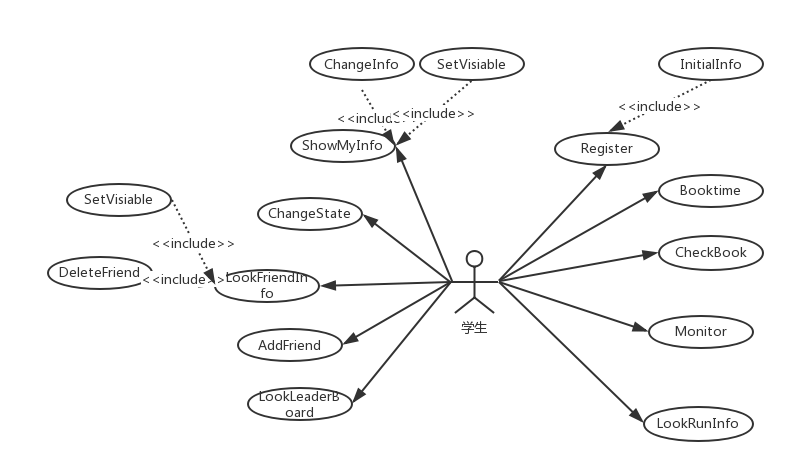


图2-4学生用例图

|  |  |  |  |
| --- | --- | --- | --- |
| 角色 | 用例 | 功能 | 实现函数 |
| Student(学生) | Register | 通过学号/一卡通进行注册，在注册成功后自动调用InitialInfo初始化个人信息。 |  |
| InitialInfo | 初始化个人信息，包括院系，年纪，班级，性别，学号，姓名（或者昵称）。 |  |
| BookTime | 提前预定跑操时间，允许进行多次预定。 |  |
| CheckBook | 查看已有的预定，并能进行修改时间，删除预订等操作。 |  |
| Monitor | 监控系统时间，当时间符合预定时间时进行提醒。 |  |
| ShowMyInfo | 拥有自己的个人主页，包含各类个人信息（各类app中的“我”，“我的”这个栏目）。 |  |
| ChangeInfo | 修改个人信息。 |  |
| SetVisiable | 再查看个人信息时可以设置对全体好友的可见与否，当查看某好友信息时设置对其单独的可见与否。 |  |
| ChangeState | 更改个人状态，例如空闲忙碌跑操中等等。 |  |
| AddFriend | 通过学号，一卡通号添加好友。 |  |
| LookFriendInfo | 查看好友信息，如被屏蔽则显示“该用户信息不可见”。 |  |
| DeleteFriend | 删除好友。 |  |
| LookLeaderBoard | 查看排行榜（次数、分数、消耗卡路里等），分为校内排行和好友排行。 |  |

表2-3学生用例图说明

上表详细说明了在学生模块中，用户的需求，并正对用例图，详细说明了每个用例的功能。

# 2.1.2 老师模块需求

在该模块，涉及的主要角色是Manager(体育老师)。

具体用例图如下：

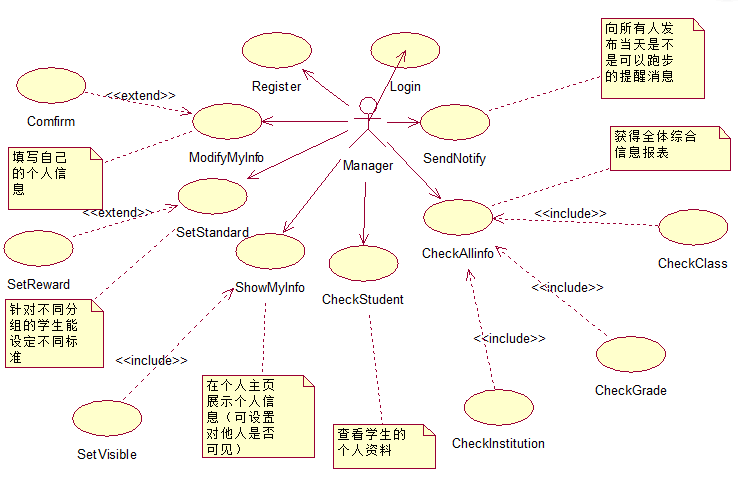


图2-5 Manager用例图

在此模块，涉及了登录注册功能，信息的填写及展示功能，设置标准，发布通知以及查询信息的功能，具体的用例说明如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| 角色 | 用例 | 功能 | 实现函数 |
| Manager（体育老师） | Register | 提供用户名快捷注册方式。统一通过验证码方式验证。 |  |
| Login | 通过注册的账号密码登录系统。 |  |
| ModifyMyInfo | 修改基本信息、工作经历、学习经历、通讯信息、职称信息等信息。 |  |
| SetStandard | 针对不同分组的学生能设定不同标准，包括跑操次数，时间要求，距离要求，运动成绩的达标值等。 |  |
| SetReward | 冬季长跑？阳光长跑？ |  |
| ShowMyInfo | 拥有自己的个人主页，包含各类个人信息（各类app中的“我”，“我的”这个栏目。 |  |
| CheckStudent | 查看所有学生的个人资料，包括姓名，学号，性别，身体素质状况等。能够查询学生历史跑操记录，包括时间，地点，距离，平均速度，消耗的卡路里等运动信息。 |  |
| CheckAllinfo | 能够获得全体综合信息报表（对数据的总结）和整体学生的运动状况，包括每个班/年级/学院平均运动次数，平均运动时间，达标人数，不达标人数，各个成绩段的分布情况等（包括全体学生的任务完成进度：已完成人数，未完成人数，风险人数，不可能完成人数）。 |  |
| SendNotify | 能向所有人发布当天是不是可以跑步的提醒消息。 |  |
| SetVisible | 能对他人设置是否可见。 |  |

表2-4Manager用例图说明

另一主要角色是Teacher(辅导员/班主任)。

具体用例图如下：

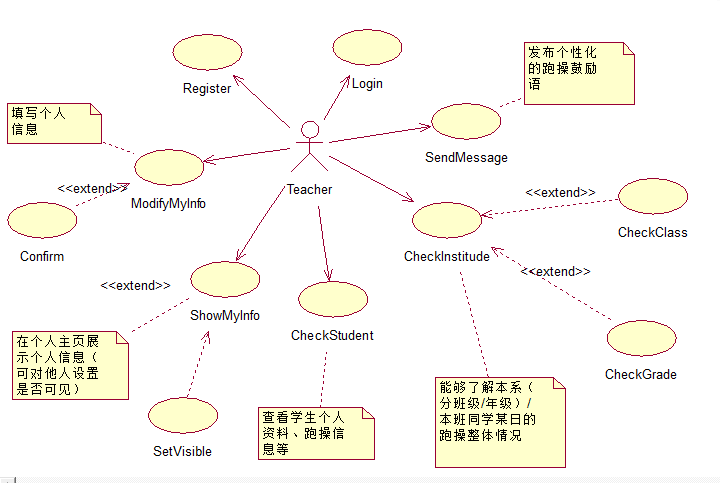


图2-5 Teacher用例图

在此模块，涉及了登录注册功能，信息的填写及展示功能，发布通知以及查询信息的功能，具体的用例说明如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| 角色 | 用例 | 功能 | 实现函数 |
| Teacher（班主任/辅导员） | Register | 提供用户名快捷注册方式。统一通过验证码方式验证。 |  |
| Login | 通过注册的账号密码登录系统。 |  |
| ModifyMyInfo | 修改基本信息、工作经历、学习经历、通讯信息、职称信息等信息。 |  |
| ShowMyInfo | 拥有自己的个人主页，包含各类个人信息（各类app中的“我”，“我的”这个栏目。 |  |
| CheckStudent | 查看所有学生的个人资料，包括姓名，学号，性别，身体素质状况等。能够查询学生历史跑操记录，包括时间，地点，距离，平均速度，消耗的卡路里等运动信息。 |  |
| CheckInstitude | 能够了解本系（分班级、年级）/本班同学某日的跑操整体情况，以便对本系别（不同班级）/本班级同学进行表扬，批评，鼓励。 |  |
| SendMessage | 发布个性化的跑操鼓励语。 |  |
| SetVisible | 能对他人设置是否可见。 |  |

表2-5Manager用例图说明

# 2.2 本章小结

本章主要介绍了整个系统的需求分析，包括系统功能性需求，并通过使用用例图介绍了两个模块的具体功能，通过表格详细分析了用例图。

第三章 系统设计

# 3.1 总体系统功能设计

本系统是基于Android应用框架开发的学生跑操管理系统，主要分成学生模块，教师模块，服务端模块。在学生模块中，学生通过登录模块注册并登陆后，在跑操信息管理模块中可以实时查看个人的跑操数据，可以在跑步过程中查看本次跑步相关的数据，同时可以在个人信息管理模块中查看修改个人信息，最后可以通过社交模块进行添加好友，建立群聊，进行约跑等行为。教师模块中，教师可以通过学生数据查询模块查询所有学生的个人资料，跑操情况等信息，并通过信息发布模块发出天气信息，鼓励评语等。服务端模块主要和网络数据库配套，用于存储个人信息和每个学生的跑操信息三个模块构成了整个学生跑操管理系统的流程。本系统的总体设计如图3-1所示。

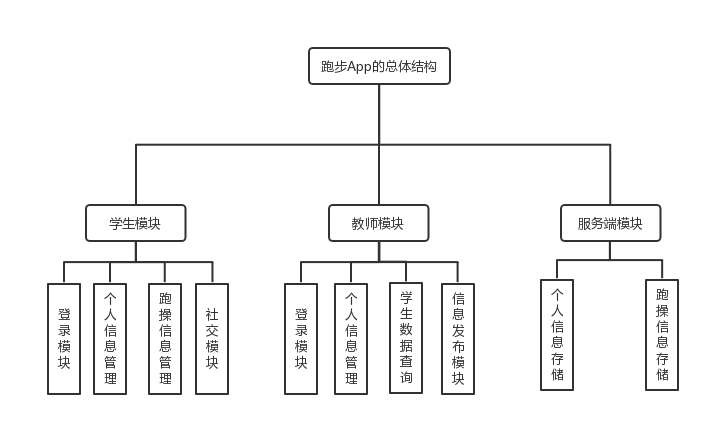


图3-1 学生跑操管理系统模块设计图

**3.1.1 学生模块**

登陆模块：负责学生的注册以及登陆。

个人信息管理：学生可以在该模块中查看个人信息，修改个人信息（如昵称等），并且设置个人信息是否对他人可见。

跑操信息管理：在该模块中，学生可以查看自己跑操的分数，每天完成的情况，并得到相应的建议。在跑步过程中，还可以实时检测已跑距离，速度，并根据个人身体素质给予跑速建议，最后在快到达终点时会进行提醒。除此之外，学生还可以预定每天的跑步时间，确保自己不会错过当天的跑操。

社交模块：学生可以添加/删除好友，查看好友的跑步情况，个人资料，进行能量的赠与、传递，还可以建立群组进行群聊，约跑等活动。

**3.1.2 教师模块**

教师模块中的登录模块与个人信息管理模块与学生模块基本相同，主要介绍后两个模块。

学生数据查询：教师登陆之后可以通过该模块查看所有学生的个人资料，包括姓名，学号，性别，身体素质状况等。能够查询学生历史跑操记录，包括时间，地点，距离，平均速度等运动信息。还能够获得全体综合信息报表（对数据的总结）和整体学生的运动状况，包括每个班/年级/学院平均运动次数，平均运动时间，达标人数，不达标人数，各个成绩段的分布情况等（包括全体学生的任务完成进度：已完成人数，未完成人数，风险人数，不可能完成人数）。能够了解本系（分班级、年级）/本班同学某日的跑操整体情况，以便对本系别（不同班级）/本班级同学进行表扬，批评，鼓励。

信息发布模块：可以根据不同同学的跑操情况发布不同的标语，同时可以发布天气的预告（能否跑步）。

**3.1.3 服务端模块**

本模块主要用来存储数据，和网络数据库配套，个人信息存储主要用来存储每位用户的个人信息，包括姓名，院系，学号，电话等，跑操信息存储主要用于存储每个学生的跑操得分。每天的跑操情况等，便于教师进行查询。

# 3.2 体系结构设计

Destination跑操管理系统因为功能简单，并发量适中（每校数千人），采用了三层C/S结构风格设计。客户端包含表示层与功能层，服务端则只包含数据层。

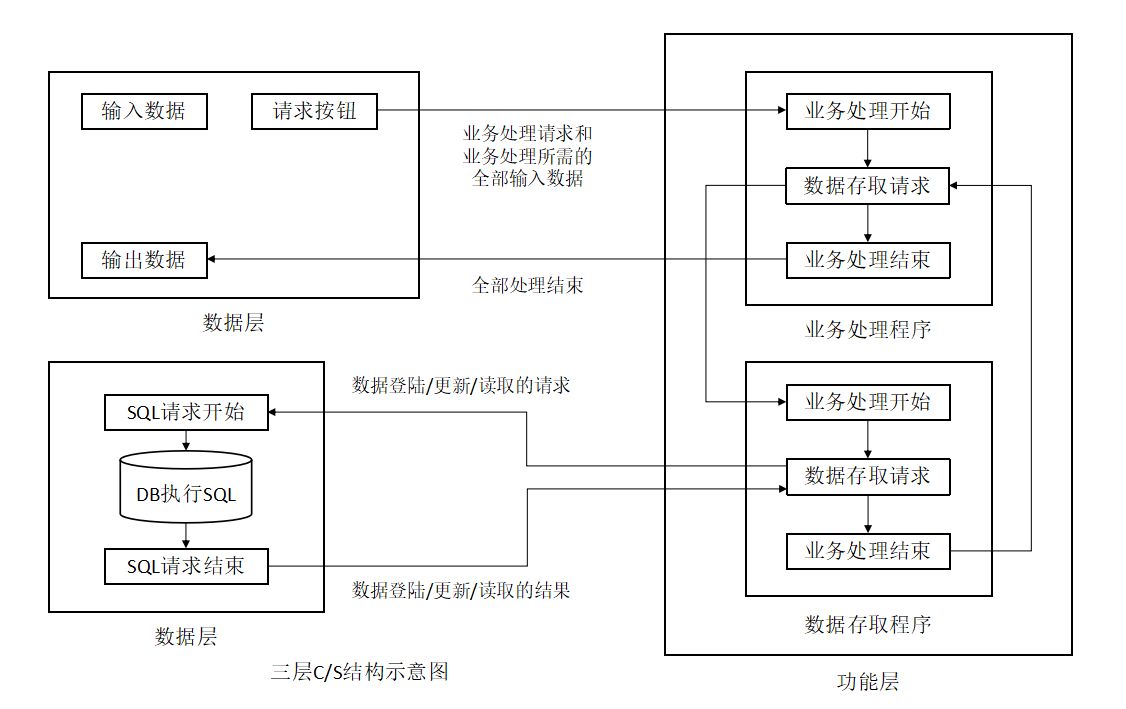


图3-2 C/S结构示意图

# 3.3 接口设计

客户端网络连接接口说明：

在接口中，用户直接调用HTTPClient，ReadObject()，HttpClient，CreateObject()， HttpClient，UpdateObject()，HttpClient，DeleteObject()，HttpClient，ReadWithFilterConditions()。这些方法组成了逻辑上的命令包装模块。调用这些方法并传入相关数据Map即可。

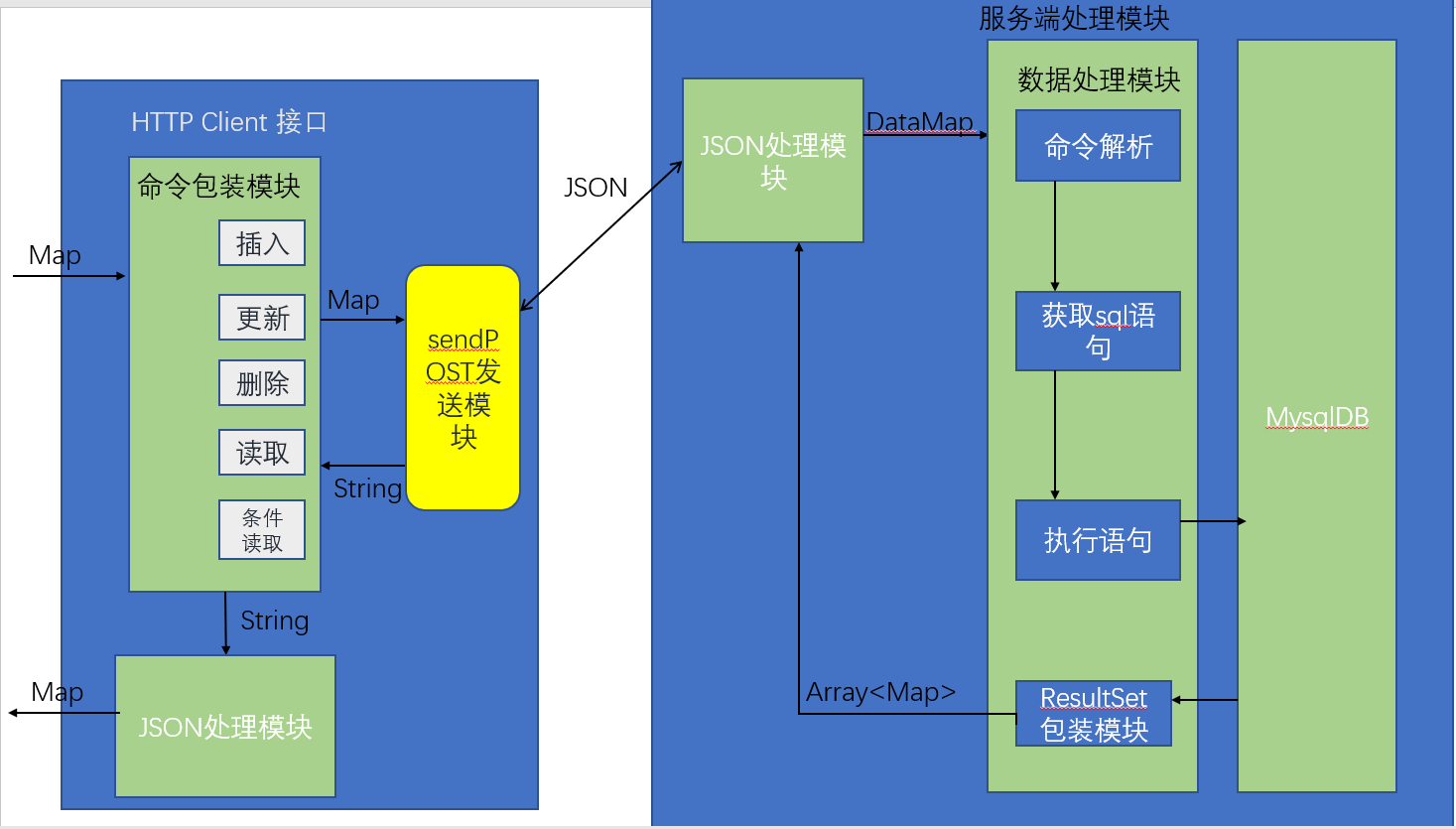
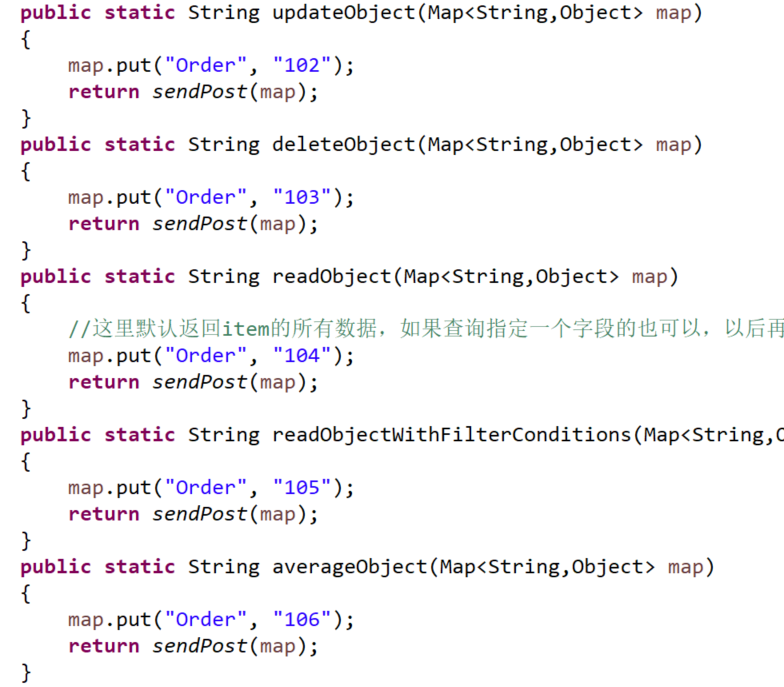


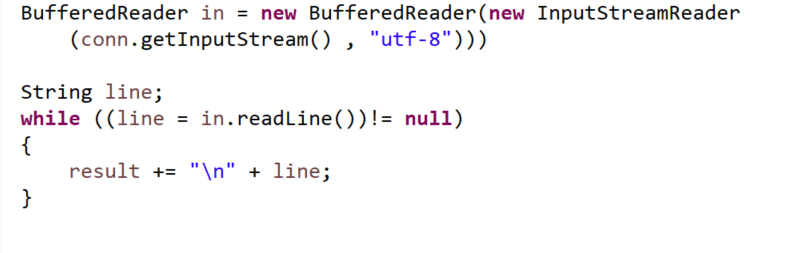
图3-3 客户端网络连接接口与服务端模块示意图



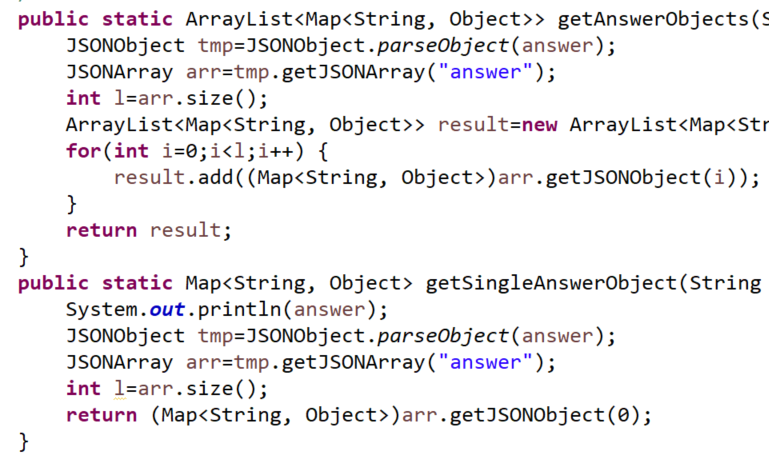
经过上述模块，Map别转发到发送模块，由sendPOST(Map)转化为json，并以字节流发送到服务器。



将返回值读取到String中。



用户可以通过调用getSingleAnswerObject(),getAnswerObjects()两个方法得到返回的数据数组。



# 3.4 本章小结

本章主要介绍了系统的设计。

首先从总体出发，描述系统的总体及各个模块。从三大模块分别详细分析各个模块的内容。

然后介绍整个系统的体系结构设计。

最后介绍服务端与客户端的接口设计，提供了网络状态参数表。

第四章 系统的模块设计与实现

# 4.1 数据库的设计

数据库名： Destination

|  |  |  |
| --- | --- | --- |
| 序号 | 表名 | 说明 |
| 1 | AccountInfo | 账号信息 |
| 2 | AccountSportsRecord | 账号运动记录 |
| 3 | Friendship | 朋友 |
| 4 | Groupship | 群组 |
| 5 | StudentInfo | 学生信息 |
| 6 | TeacherInfo | 教师信息 |
| 7 | UniversityRule | 学校规则 |
| 8 | StudentSportsRecord | 学生运动记录 |

数据表名： AccountInfo

说明： 账号信息

数据列：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | id | VARCHAR | 序号 |
| 2 | a\_type | INT | 账号类型 |
| 3 | a\_id | INT | 账号ID |
| 4 | a\_name | CHAR | 账号名 |
| 5 | a\_password | CHAR | 密码 |
| 6 | nickname | CHAR | 昵称 |
| 7 | user\_type | INT | 用户类型 |
| 8 | user\_id | INT | 用户ID |
| 9 | gender | INT | 性别 |
| 10 | birthday | INT | 生日 |
| 11 | height | INT | 身高 |
| 12 | weight | INT | 体重 |
| 13 | scheduled\_time | datetime | 预定跑操的时间 |

数据表名： AccountSportsRecord

说明： 账号运动记录

数据列：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | id | VARCHAR | 序号 |
| 2 | a\_id | INT | 账号ID |
| 3 | date | CHAR | 日期 |
| 4 | time | CHAR | 时间 |
| 5 | location | CHAR | 位置 |
| 6 | distance | INT | 距离 |
| 7 | cost | INT | 消耗热量 |

数据表名： Friendship

说明： 朋友

数据列：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | id | VARCHAR | 序号 |
| 2 | a\_id\_1 | INT | 用户1ID |
| 3 | a\_id\_2 | INT | 用户2ID |

数据表名： Groupship

说明： 群组

数据列：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | id | VARCHAR | 序号 |
| 2 | group\_id | INT | 群组ID |
| 3 | group\_name | CHAR | 群组名称 |
| 4 | member\_id | INT | 成员ID |
| 5 | member\_type | INT | 成员类型 |

数据表名： StudentInfo

说明： 学生信息

数据列：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | id | VARCHAR | 序号 |
| 2 | stu\_id | INT | 学生ID |
| 3 | school\_id | INT | 学校ID |
| 4 | stu\_name | CHAR | 学生姓名 |
| 5 | stu\_gender | INT | 学生性别 |
| 6 | phone\_number | CHAR | 电话 |
| 7 | email | CHAR | 电子邮箱 |
| 8 | score | INT | 分数 |
| 9 | institution | varchar(32) | 学院 |
| 10 | grade | varchar(32) | 年级 |
| 11 | class | varchar(32) | 班级 |

数据表名： TeacherInfo

说明： 教师信息

数据列：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | id | VARCHAR | 序号 |
| 2 | tea\_id | INT | 教师ID |
| 3 | school\_id | INT | 学校ID |
| 4 | tea\_name | CHAR | 教师姓名 |
| 5 | tea\_type | INT | 教师类别 |
| 6 | phone\_number | CHAR | 电话 |
| 7 | email | CHAR | 电子邮箱 |

数据表名： UniversityRule

说明： 学校规则

数据列：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | id | VARCHAR | 序号 |
| 2 | school\_id | INT | 学校ID |
| 3 | school\_name | CHAR | 学校名称 |
| 4 | auther\_id | INT | 管理员ID |
| 5 | period | CHAR | 规则适用时期 |
| 6 | time | CHAR | 运动时间 |
| 7 | run\_time | CHAR | 跑操时间 |
| 8 | pass\_score | INT | 及格分数 |
| 9 | score\_set | CHAR | 分数设置 |
| 10 | run\_limit | CHAR | 跑步得分上限 |
| 11 | standard | CHAR | 运动标准 |
| 12 | run\_standard | CHAR | 跑操标准 |

数据表名： StudentSportsRecord

说明： 学生运动记录（仅得分记录）

数据列：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 数据类型 | 说明 |
| 1 | id | VARCHAR | 序号 |
| 2 | school\_id | INT | 学校ID |
| 3 | stu\_id | INT | 学生ID |
| 4 | type | CHAR | 运动类型 |
| 5 | date | INT | 日期 |
| 6 | time | CHAR | 时间 |

# 4.2 学生模块的实现

**4.2.1 学生互动模块**

学生模块中一小块功能即为互动功能，即通过锻炼产生能量，然后可以用能量来种树，同时会有一个排行榜，可以收取其他好友的能量，从而达到一定的趣味性，增加用户粘性，具体实现如下：

一个表示新生成能量的类，其对象作为用户的一个成员属性：

**public class** Energy {  
 **private double energy**; //新生成能量  
 **private double LossEnergy**; //失去的能量  
 **private double ExpirationTime**; //能量过期时间  
 HashMap **WhoSteal** = **new** HashMap();//储存收过我能量的用户，以此实现不可重复收取  
 **public** Energy(){ //构造函数  
 **energy** = 0;  
 **LossEnergy** = 0;  
 **ExpirationTime** = 0;

}

// 其他为set、get函数

}

1. 学生类：

//增加能量

**private void** EnergyIncrease(**double** take) {  
 **TotalEnergy** += take;  
}

**/**/转换能量  
**private void** EnergyTran() {  
 **if**(**ExerciseDistance** < 1000){  
 **newenergy**.setEnergy(Math.*random*()\*40 + 30);  
 **newenergy**.setExpirationTime(86400);  
 }  
 **else if**(**ExerciseDistance** < 3000){  
 **newenergy**.setEnergy(**ExerciseDistance**/3000\*256);  
 **newenergy**.setExpirationTime(86400);  
 }  
 **else** {  
 **newenergy**.setEnergy(256);  
 **newenergy**.setExpirationTime(86400);  
 }  
}

//收取自己的能量  
**public void** TakeEnergy() {  
 **double** temp = **newenergy**.getEnergy();  
 **newenergy**.setEnergy(0);  
 **newenergy**.setExpirationTime(0);  
 EnergyIncrease(temp);  
}

//收取他人的能量  
**public double** TakeOthersEnergy(Student OtherUser) {  
 **if**(OtherUser.**newenergy**.**WhoSteal**.containsKey(**UserID**)){   
 **return** -1;  
 } //这个判断语句判定用户这一能量周期内是否收取过我的能量，防止重复收取 **if**(OtherUser.**newenergy**.getLossEnergy() / (OtherUser.**newenergy**.getEnergy() + OtherUser.**newenergy**.getLossEnergy()) >= 0.4) {  
 **return** -2;  
 } //这个判断语句用于判定用户失去的能量是否达到一定数值，超过这个数值不再让其他用户收取

**else**{  
 **double** temp = Math.*random*()\*((OtherUser.**newenergy**.getEnergy() + OtherUser.**newenergy**.getLossEnergy())\*0.4) + 1;  
 OtherUser.**newenergy**.setEnergy(OtherUser.**newenergy**.getEnergy() - temp); //更改被收取能量用户剩余能量  
OtherUser.**newenergy**.setLossEnergy(OtherUser.**newenergy**.getLossEnergy() + temp); //记录用户被收取能量，用于判定失去能量的比例，达到超过比例便不再让别人收取的目的  
 OtherUser.**newenergy**.**WhoSteal**.put(**UserID**, temp);  
 EnergyIncrease(temp); //记录收取过我能量的用户，防止二次收取  
 **return** temp;  
 }  
}

**4.2.2 学生跑步模块**

这里主要是根据坐标转换距离、以及获取坐标问题

*//输入两个经纬度信息计算距离*

**public static double** getDistance(**double** lng1, **double** lat1, **double** lng2, **double** lat2){

**final double** EARTH\_RADIUS = 6378137;  
 **double** radLat1 = *rad*(lat1);  
 **double** radLat2 = *rad*(lat2);  
 **double** a = radLat1 - radLat2;  
 **double** b = *rad*(lng1) - *rad*(lng2);  
 **double** s = 2 \* Math.*asin*(Math.*sqrt*(Math.*pow*(Math.*sin*(a/2),2)+

Math.*cos*(radLat1)\*Math.*cos*(radLat2)\*Math.*pow*(Math.*sin*(b/2),2)));  
 s = s \* EARTH\_RADIUS;  
 s = Math.*round*(s \* 10000) / 10000;  
 **return** s;  
}

//发送post请求，可以用于获取定位

**public static** String sendGet(String url, String param) {  
 String result = **""**;  
 BufferedReader in = **null**;  
 **try** {  
 String urlNameString = url + **"?"** + param;  
 URL realUrl = **new** URL(urlNameString);  
 URLConnection connection = realUrl.openConnection();  
 connection.setRequestProperty(**"accept"**, **"\*/\*"**);  
 connection.setRequestProperty(**"connection"**, **"Keep-Alive"**);  
 connection.setRequestProperty(**"user-agent"**,  
 **"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;SV1)"**);  
 connection.connect();  
 Map<String, List<String>> map = connection.getHeaderFields();  
 */\*for (String key : map.keySet()) {  
 System.out.println(key + "--->" + map.get(key));  
 }\*/* in = **new** BufferedReader(**new** InputStreamReader(  
 connection.getInputStream()));  
 String line;  
 **while** ((line = in.readLine()) != **null**) {  
 result += line;  
 }  
 } **catch** (Exception e) {  
 System.***out***.println(**"GET Failed! "** + e);  
 e.printStackTrace();  
 }  
 **finally** {  
 **try** {  
 **if** (in != **null**) {  
 in.close();  
 }  
 } **catch** (Exception e2) {  
 e2.printStackTrace();  
 }  
 }  
 **return** result;  
}

//发送get请求，可以用于获取定位

**public static** String sendPost(String url, String param) {  
 PrintWriter out = **null**;  
 BufferedReader in = **null**;  
 String result = **""**;  
 **try** {  
 URL realUrl = **new** URL(url);  
 URLConnection conn = realUrl.openConnection();  
 conn.setRequestProperty(**"accept"**, **"\*/\*"**);  
 conn.setRequestProperty(**"connection"**, **"Keep-Alive"**);  
 conn.setRequestProperty(**"user-agent"**,  
 **"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;SV1)"**);  
 conn.setDoOutput(**true**);  
 conn.setDoInput(**true**);  
 out = **new** PrintWriter(conn.getOutputStream());  
 out.print(param);  
 out.flush();  
 in = **new** BufferedReader(  
 **new** InputStreamReader(conn.getInputStream()));  
 String line;  
 **while** ((line = in.readLine()) != **null**) {  
 result += line;  
 }  
 } **catch** (Exception e) {  
 System.***out***.println(**"POST Failed! "**+e);  
 e.printStackTrace();  
 }  
 **finally**{  
 **try**{  
 **if**(out!=**null**){  
 out.close();  
 }  
 **if**(in!=**null**){  
 in.close();  
 }  
 }  
 **catch**(IOException ex){  
 ex.printStackTrace();  
 }  
 }  
 **return** result;  
}

**4.2.3 天气提醒**

通过调用中央气象台的api，获取实时的天气信息，并根据对应情形做出提醒。相关代码如下：

/\*\*

\*

\* 获取实时天气2<br>

\* 方 法 名： getTodayWeather <br>

\*

\* **@param** Cityid

\* 城市编码

\*/

**public** **static** Map<String, Object> getTodayWeather2(String Cityid)

**throws** IOException, NullPointerException {

// 连接中央气象台的API

URL url = **new** URL("http://www.weather.com.cn/data/cityinfo/" + Cityid

+ ".html");

URLConnection connectionData = url.openConnection();

connectionData.setConnectTimeout(1000);

Map<String, Object> map = **new** HashMap<String, Object>();

**try** {

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(

connectionData.getInputStream(), "UTF-8"));

StringBuilder sb = **new** StringBuilder();

String line = **null**;

**while** ((line = br.readLine()) != **null**)

sb.append(line);

String datas = sb.toString();

System.***out***.println(datas);

JSONObject jsonData = JSONObject.*fromObject*(datas);

JSONObject info = jsonData.getJSONObject("weatherinfo");

map.put("city", info.getString("city").toString());// 城市

map.put("temp1", info.getString("temp1").toString());// 最高温度

map.put("temp2", info.getString("temp2").toString());// 最低温度

map.put("weather", info.getString("weather").toString());//天气

map.put("ptime", info.getString("ptime").toString());// 发布时间

} **catch** (SocketTimeoutException e) {

System.***out***.println("连接超时");

} **catch** (FileNotFoundException e) {

System.***out***.println("加载文件出错");

}

**return** map;

}

**4.2.4 学生跑操记录查询模块**

对于学生各类跑操信息的获取及给予建议（包括1，过去跑操的详细数据;2,早晚操的完成情况；3，当天的早晚操情况；4，已经获得的分数/未获得的分数，已经经过的日期/剩余日期）

核心操作：通过点击相应的标签向服务器发送请求，接受服务器回传的数据并处理，最后呈现在跳转的界面上

相应代码如下：

（1）：日历界面：通过点击日期返回当天跑操信息

**for**(**int** i = 0;i < 42;i++)

{

**int** now=i;

day[i].setText("");

day[i].setForeground(Color.***red***);//设置字体颜色

day[i].setFont(**new** Font("Jokerman",Font.***BOLD***,20));

day[i].setBorder(**null**);

day[i].addMouseListener(**new** MouseAdapter()

{ String choice;

**public** **void** mouseClicked(MouseEvent e)

{

JFrame newFrame=**new** JFrame("Destination-Daily");

JPanel pan1=**new** JPanel(**new** FlowLayout());

JTextField item=**new** JTextField("这是你今天的跑步情况",20);

item.setBackground(Color.***green***);

item.setEditable(**false**);

pan1.add(item);

JPanel pan2=**new** JPanel(**new** GridLayout(5,2));

JLabel lab1=**new** JLabel("早操");

JLabel lab2=**new** JLabel("晚操");

JLabel lab3=**new** JLabel("跑步距离");

JLabel lab4=**new** JLabel("平均速度");

JLabel lab5=**new** JLabel("得分");

JTextField jtf1=**new** JTextField("0",10);

JTextField jtf2=**new** JTextField("0",10);

JTextField jtf3=**new** JTextField("0",10);

JTextField jtf4=**new** JTextField("0",10);

JTextField jtf5=**new** JTextField("0",10);

pan2.add(lab1);

pan2.add(jtf1);

pan2.add(lab2);

pan2.add(jtf2);

pan2.add(lab3);

pan2.add(jtf3);

pan2.add(lab4);

pan2.add(jtf4);

pan2.add(lab5);

pan2.add(jtf5);

newFrame.add(pan1,BorderLayout.***NORTH***);

newFrame.add(pan2,BorderLayout.***CENTER***);

newFrame.setSize(50, 200);

newFrame.setVisible(**true**);

String nowDay=day[now].getText();

Integer nowMonth=month\_list.getSelectedIndex()+1;

String nowYear=t.getText();

String selectedDate=nowYear+"-"+nowMonth.toString()+"-"+nowDay;

System.***out***.println(selectedDate);

Map item=**new** HashMap();

}

（2）：通过定时(24小时)向服务器发送请求,获得跑操进度（ 已经获得的分数/未获得的分数，已经经过的日期/剩余日期）

相关代码如下：

/\*功能描述：获得已跑次数/未跑次数；已经过过日期/剩余日期；

\*

\*/

package package1;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.util.HashMap;

import java.util.Map;

public class Procession extends JFrame {

private JLabel la1,la2,la3,la4;

private JTextField jtf1,jtf2,jtf3,jtf4;

private String score;

public Procession()

{

Container cp=getContentPane();

cp.setLayout(new FlowLayout());

la1=new JLabel("已获得分数");

la2=new JLabel("达标分数");

la3=new JLabel("已经过日程");

la4=new JLabel("总日程");

jtf1=new JTextField("0",10);

jtf2=new JTextField("0",10);

jtf3=new JTextField("0",10);

jtf4=new JTextField("0",10);

cp.add(la1);

cp.add(jtf1);

cp.add(la2);

cp.add(jtf2);

cp.add(la3);

cp.add(jtf3);

cp.add(la4);

cp.add(jtf4);

score=new NewTimer().getScore();

int total;

String start,end;

String now;

jtf1.setText(score);//获取已获得分数

jtf2.setText("total");//目标分数

jtf3.setText("now-start");//已过日程

jtf4.setText("end-start");//总日程

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(150,280);

setVisible(true);

setTitle("Destination-Information");

}

public static void main(String[] args)

{

new Procession();

}

}

/\*

\* 功能：定时器，定时向服务器发送请求，获取最新的分数和日程进度

\*/

package package1;

import java.util.Calendar;

import java.util.Date;

import java.util.Timer;

import java.util.TimerTask;

import java.util.Map;

import java.util.HashMap;

public class NewTimer {

private static String score;

public String getScore()

{

return score;

}

public static void timer4() {

Calendar calendar = Calendar.getInstance();

calendar.set(Calendar.HOUR\_OF\_DAY, 12); // 控制时

calendar.set(Calendar.MINUTE, 0); // 控制分

calendar.set(Calendar.SECOND, 0); // 控制秒

Date time = calendar.getTime(); // 得出执行任务的时间,此处为今天的12：00：00

Timer timer = new Timer();

timer.scheduleAtFixedRate(new TimerTask() {

public void run() {

Map item=new HashMap();

item.put("id", "score");

score=new HttpClient().createObject(item);

}

}, time, 1000 \* 60 \* 60 \* 24);// 这里设定将延时每天固定执行

}

}

# 4.3 老师模块的实现

发布通告（体育老师发布跑操公告/班主任（辅导员）发送鼓励信息）

相应代码如下：

**public** **class** MainFrame **extends** JFrame {

**private** JButton but;

**private** JTextField jtf;

**private** String mess;

**private** Scanner sc;

**public** MainFrame()

{

Container cp=getContentPane();

cp.setLayout(**new** FlowLayout());

but=**new** JButton("跑操信息发布");

cp.add(but);

jtf=**new** JTextField("0",10);

cp.add(jtf);

but.addActionListener(**new** ActionListener()

{

**public** **void** actionPerformed(ActionEvent e)

{

JFrame newFrame=**new** JFrame("Destination-Report");

JPanel pan1=**new** JPanel(**new** FlowLayout());

JLabel la=**new** JLabel("请输入发布内容");

JTextField jtf1=**new** JTextField("0",10);

//BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

// try {

// mess = br.readLine();

// } catch (IOException e1) {

// e1.printStackTrace();

// }\*/

jtf1.setText("今天不跑操");

mess=jtf1.getText();

JButton but1=**new** JButton("确认发布");

but1.addActionListener(**new** ActionListener()

{

**public** **void** actionPerformed(ActionEvent e1)

{

**if**(mess==**null**)

JOptionPane.*showMessageDialog*(**null**, "发布内容为空，请重新输入","Destination-Attention",JOptionPane.***WARNING\_MESSAGE***);

**else**

jtf1.setText("0");

System.***out***.println(mess);

JOptionPane.*showMessageDialog*(**null**, "公告已发布","Destination-Message",JOptionPane.***INFORMATION\_MESSAGE***);

Map item=**new** HashMap();

item.put("id", mess);

jtf.setText(**new** HttpClient().*createObject*(item));

}

});

pan1.add(la);

pan1.add(jtf1);

newFrame.add(pan1,BorderLayout.***CENTER***);

newFrame.add(but1,BorderLayout.***SOUTH***);

newFrame.setSize(200, 400);

newFrame.setVisible(**true**);

}

});

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setSize(300,400);

setVisible(**true**);

setTitle("Destination-Report");

}

查询

班主任/辅导员能够了解本系（分班级）/本班同学某日的跑操整体情况，以便对本系别（不同班级）/本班级同学进行表扬，批评，鼓励。同时也能够查询某个同学的跑操记录和成绩，以便了解该同学的日常作息。

**public** String CheckStudentInfo(String stuid)

{

String res="";

java.util.Map<String, Object> selfmap=**new** HashMap<String, Object>();

**try**{

selfmap.put("table\_name","StudentInfo");

selfmap.put("stu\_id", stuid);

res=HttpClient.*readObject*(selfmap);//调用HttpClient的函数

}**catch**(Exception e){

e.printStackTrace();

}

**return** res;

}

**public** String CheckStudentRecord(String aid)

{

String res="";

java.util.Map<String, Object> selfmap=**new** HashMap<String, Object>();

**try**{

selfmap.put("table\_name","AccountSportsRecord");

selfmap.put("a\_id", aid);

res=HttpClient.*readObject*(selfmap);//调用HttpClient的函数

}**catch**(Exception e){

e.printStackTrace();

}

**return** res;

}

**public** String CheckClass(String banji)

{

String res="";

java.util.Map<String, Object> selfmap=**new** HashMap<String, Object>();

**try**{

selfmap.put("table\_name","StudentInfo");

selfmap.put("banji", banji);

res=HttpClient.*readObject*(selfmap);//调用HttpClient的函数

}**catch**(Exception e){

e.printStackTrace();

}

**return** res;

}

public String CheckGrade(String grade)

1）体育老师可以获得全体综合信息报表（对数据的总结）和整体学生的运动状况，包括每个班平均运动次数，平均运动时间，达标人数，不达标人数，各个成绩段的分布情况等（包括全体学生的任务完成进度：已完成人数，未完成人数，风险人数，不可能完成人数）。

**public** String CheckAllinfo()

{

String res="";

java.util.Map<String, Object> selfmap=**new** HashMap<String, Object>();

**try**{

selfmap.put("table\_name","StudentInfo");

res=HttpClient.*readObject*(selfmap);//调用HttpClient的函数

res+=HttpClient.*averageObject*(selfmap);

}**catch**(Exception e){

e.printStackTrace();

}

**return** res;

}

**public** String CheckClass(String banji)

**public** String CheckGrade(String grade)

**public** String CheckInstitution(String institution)

2）能查看所有学生的个人资料，包括姓名，学号，性别，身体素质状况等。能够查询学生历史跑操记录，包括时间，地点，距离，平均速度，消耗的卡路里等运动信息。

**public** String CheckStudentInfo(String stuid)

{

String res="";

java.util.Map<String, Object> selfmap=**new** HashMap<String, Object>();

**try**{

selfmap.put("table\_name","StudentInfo");

selfmap.put("stu\_id", stuid);

res=HttpClient.*readObject*(selfmap);//调用HttpClient的函数

}**catch**(Exception e){

e.printStackTrace();

}

**return** res;

}

**public** String CheckStudentRecord(String aid)

{

String res="";

java.util.Map<String, Object> selfmap=**new** HashMap<String, Object>();

**try**{

selfmap.put("table\_name","AccountSportsRecord");

selfmap.put("a\_id", aid);

res=HttpClient.*readObject*(selfmap);//调用HttpClient的函数

}**catch**(Exception e){

e.printStackTrace();

}

**return** res;

}

3）设置标准

体育老师能设定跑操标准，包括跑操次数，时间要求，距离要求，运动成绩的达标值等。

**public** **boolean** setStandard(java.util.Map<String,Object> map)

{

**try**{

HttpClient.*createObject*(map);//调用HttpClient的函数

}**catch**(Exception e){

e.printStackTrace();

**return** **false**;

}

**return** **true**;

}

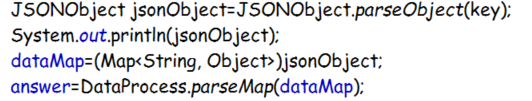
# 4.4 服务端模块的实现

首先服务端webservice使用Struts2框架。

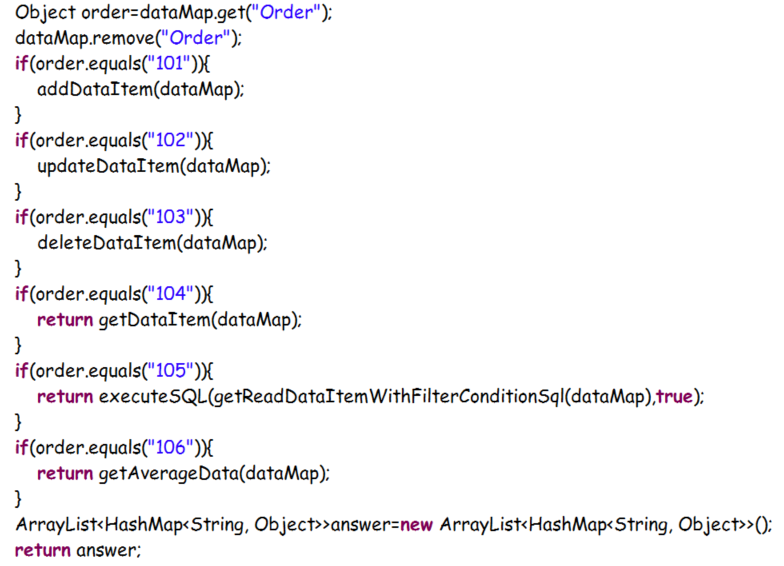
在JSON处理模块中，通过ActionContext对象获取到json值，



将json转化为Map，传入解析模块。

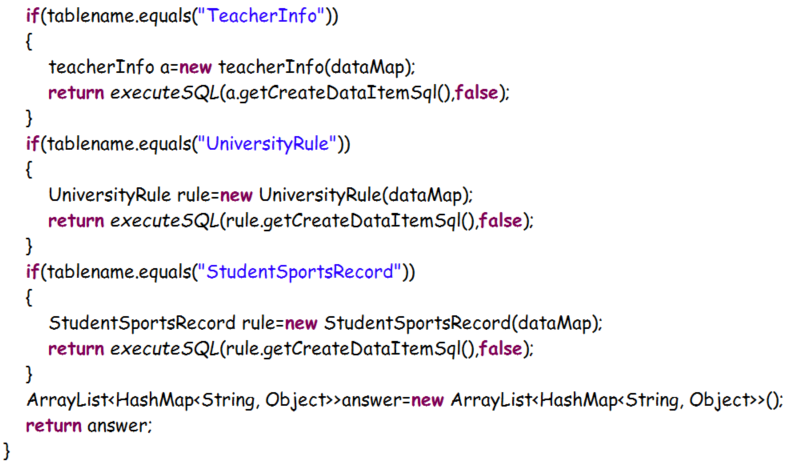


在数据处理模块，先解析命令，调用对应函数。



对应函数中，解析表名，通过建立不同的表的对象，获得sql语句。

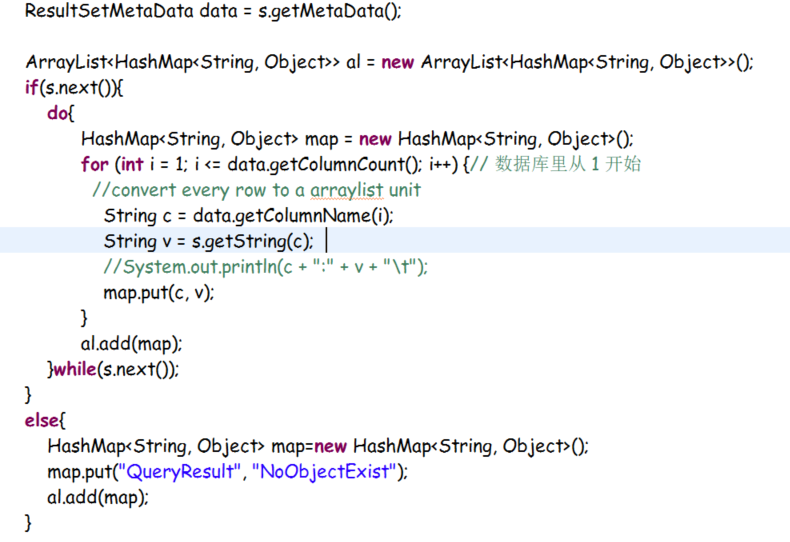




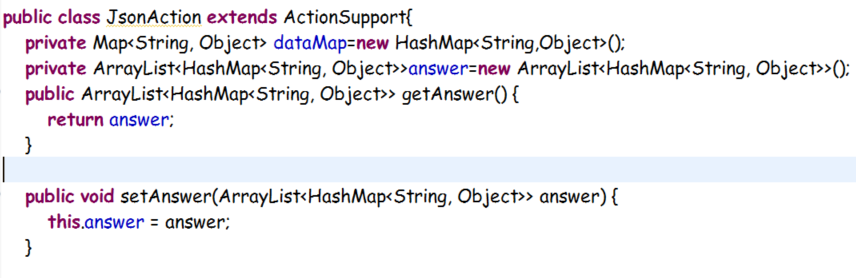
再调用执行函数。



将返回的ResultSet处理



返回ArrayList<Map>给JSON处理模块，返回给客户端。



# 4.5 本章小结

本章主要介绍了系统的实现。介绍了实现时使用的关键性技术，并贴出了大部分代码，以便理解。

在介绍实现时，配合一定的文字说明，与含有充足注释的实现代码相结合，更易于理解。

第五章 系统测试

# 5.1 环境

IDE及环境：

Android Studio 3.1

Build #AI-173.4670197, built on March 22, 2018

JRE: 1.8.0\_152-release-1024-b02 amd64

JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

Windows 10 10.0

安卓虚拟机：

Android Emulator-Nexux\_5X\_API\_27\_x86(Android 8.1.0)

实机：

SAMSUNG Galaxy S8(Android 7.0.0)

# 5.2 各模块测试

# 5.2.1 登陆注册

# 5.2.2 现有功能

# Screenshot_20180415-131226 Screenshot_20180415-131534

# Screenshot_20180415-131249 Screenshot_20180415-131256

1. 实验总结

**一、客户端功能部分**

**二、网络及服务端数据库部分**

客户端部分的设计问题：

第一．应该给一个可选择的将javaObject向map转化的方法，这样可以免去程序员向数据map里手动添加数据的麻烦。

第二．因为没有考虑到Android4以后网络连接是多线程的，出现了线程问题。

第三．返回数据的解析根本没有必要放到发送模块外面让程序员调用，直接在sendPOST方法里更好。这个问题是因为设计的时候没有考虑的很全面，使这个接口变得复杂。

服务端部分的设计问题：

第一．最大的问题是DAO层编写的太麻烦，创建了相关表的类，这样使得DB很庞大。

第二．如果将业务逻辑全部放到后端，而不是这样手机端写一部分，服务器端写一部分，就可以利用到struts2的过滤器，发挥更大的作用。

第三．Webservice与DAO耦合性太高？有这样的怀疑。

展望

这一部分的话，使用json传输数据，有一些问题出现。如果可以使用xml的方式，说不定会得以解决这些问题。

传输方式是字节流，可以尝试一下字符流。

后续应当增加一些UDP方式连接服务器的方法，以传输用户的聊天内容等。

**三、客户端前端部分**

前端选用Android系统平台，想把项目做成一个App。在实际开发过程中，发现了很多Android开发需要注意的问题，同时也意识到，Android项目的开发需要极高的高内聚低耦合性，功能开发人员也应该具有一定的Android相关知识。

项目成功的最重要因素就是设计阶段。良好的设计可以使得开发过程事半功倍，也十分有利于后期的更新迭代及维护。

**四、总结**

本次实验收获颇丰，我们第一次系统地用软件架构的思想进行程序的开发，熟悉了软件开发的整体思路与顺序，实践了课堂中学习的知识，加深了印象。我们为将来的工作或者深造也积累了软件开发的经验，并更加深切地感受到沟通交流和团队合作的重要性。

但同时我们也发现了不少不足之处。究其原因，既有由于缺乏开发经验造成的设计错误，也有缺乏沟通交流导致整合时的进度缓慢。当然，时间不够充足与开发任务困难繁重也是一大原因。

希望本次试验的成果可以继续开发完善下去，给大家在课余时间提供一个锻炼的平台。