|  |
| --- |
| 拜运动社交web应用 |
| 体系结构设计文档 |
|  |

|  |
| --- |
| 朱宇翔 141250216  2016.11 |

目录

[1. 引言 2](#_Toc466127674)

[1.1编制目的 2](#_Toc466127675)

[1.2词汇表 2](#_Toc466127676)

[1.3框架概述 2](#_Toc466127677)

[2. 整体逻辑结构概述 2](#_Toc466127678)

[3. 视图分解描述 3](#_Toc466127679)

[3.1视图之间的分解和跳转 3](#_Toc466127680)

[4. 控制器的模块分解 4](#_Toc466127681)

[4.1 控制器的各模块职责 4](#_Toc466127682)

[4.2 业务逻辑层模块的接口规范 5](#_Toc466127683)

[4.2.1 authenticateController的接口规范 5](#_Toc466127684)

[4.2.2 activityController的接口规范 5](#_Toc466127685)

[4.2.3 exerciseController的接口规范 6](#_Toc466127686)

[4.2.4 friendController的接口规范 8](#_Toc466127687)

[4.2.5 blogController的接口规范 8](#_Toc466127688)

[4.2.6 userController的接口规范 9](#_Toc466127689)

[4.2.7 searchController的接口规范 9](#_Toc466127690)

[4.2.8 mainPageController的接口规范 10](#_Toc466127691)

[5．模型（数据库）分解 10](#_Toc466127692)

[5.1 模型的各模块职责 10](#_Toc466127693)

[5.2 模型部分的详细分解 10](#_Toc466127694)

[5.2.1 活动 – ActivityModel 10](#_Toc466127695)

[5.2.2 单次健身数据（以及评论） - ExerciseModel 11](#_Toc466127696)

[5.2.3 统计数据列表 - StatisticsModel 12](#_Toc466127697)

[5.2.4 好友关系 - FriendshipModel 12](#_Toc466127698)

[5.2.5 聊天记录 - DialogModel 12](#_Toc466127699)

[5.2.6 用户 - UserModel 13](#_Toc466127700)

[6. 数据格式定义 14](#_Toc466127701)

# 1. 引言

## 1.1编制目的

本报告大略完成对拜运动社交web应用的概要设计，达到指引详细设计和开发的目的，同时帮助设计人员和测试人员，用户的沟通。

本报告面向开发人员，测试人员和最终用户而编写，是了解系统的导航。

## 1.2词汇表

|  |  |
| --- | --- |
| 词汇名称 | 词汇含义 |
| Laravel | Laravel是一套PHP Web开发框架(PHP Web Framework)。最低支持php版本为5.6.4。 |
| MVC | 视图-模型-控制器体系结构风格 |
| Route | 路由，负责分发http请求 |

## 1.3框架概述

本项目基于laravel框架进行开发。

# 2. 整体逻辑结构概述

在本系统中，选择了经典的MVC的体系结构风格，将系统分为三模块（controller, model and view）能够很好地示意整个高层抽象。模型模块主要包括对数据库的访问，控制模块包含业务逻辑处理的实现，是模型与视图的桥梁，视图模块则包括模板和对网页的渲染。MVC体系结构的逻辑视角如下图所示。



基于laravel的主要开发模块及职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| 路由 | 负责将http请求分发到不同的控制器上。 |
| 视图 | 基于窗口的快递物流管理系统客户端用户界面。 |
| 模块 | 负责管理数据库，建立基本的数据存储，读取，更改规则。 |
| 控制器 | 包含业务逻辑，负责读取各模块的信息，并结合信息调用视图。 |

# 3. 视图分解描述

## 3.1视图之间的分解和跳转



# 4. 控制器的模块分解

## 4.1 控制器的各模块职责

|  |  |
| --- | --- |
| 开发模块 | 功能 |
| activityController | 处理活动模块的业务逻辑。包括对活动的增删改查。 |
| exerciseController | 对运动健身数据处理的业务逻辑，包括管理用户运动数据和统计数据，并统筹用户评论，点赞等社交相关的功能。 |
| friendController | 好友控制器。管理好友列表以及好友聊天记录。 |
| userController | 用户控制器，管理个人账户信息和用户列表。 |
| authenticateController | 处理用户的注册，登录和登出。 |
| searchController | 处理搜索请求 |
| blogController | 处理自己与好友**主页**的控制器 |
| mainPageController | 显示**主页**的控制器，包括近期内健身信息，以及统计数据 |

## 4.2 业务逻辑层模块的接口规范

### 4.2.1 authenticateController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| authenticateController.index | 语法 | | public function index(); |
| 前置条件 | | 无 |
| 后置条件 | | 显示登录页面 |
| authenticateController.register | 语法 | | public function register(Request $request); |
| 前置条件 | | 已经通过laravel的request机制验证过用户名和密码符合格式规范 |
| 后置条件 | | POST方法绑定。如果用户名没有被占用，就存储注册信息并重定向到欢迎页面。否则返回注册页面并声明用户名被占用。 |
| authenticateController.login | 语法 | | public function login(Request $request); |
| 前置条件 | | 无 |
| 后置条件 | | POST方法绑定。若用户名与密码匹配，则登入系统并进入主页。否则返回登录页面并提示错误。 |
| 需要的服务（接口） | | | |
| 服务名 | | **服务** | |
| UserModel.\* | | 验证匹配用户名与密码，插入注册信息 | |

### 4.2.2 activityController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| activityController.index | 语法 | public function index(); |
| 前置条件 | 无 |
| 后置条件 | 显示活动列表页面 |
| activityController.create | 语法 | public function create(); |
| 前置条件 | 无 |
| 后置条件 | 进入发起活动表单填写页面 |
| activityController.createSubmit | 语法 | public function createSubmit(Request $request); |
| 前置条件 | 已经通过laravel的request机制验证表单的各项符合格式规范 |
| 后置条件 | 如果符合要求，就在模型中创建这项活动，否则就返回创建活动页面并提示错误 |
| activityController.show | 语法 | public function show($id); |
| 前置条件 | 无 |
| 后置条件 | 显示id对应的活动详细信息 |
| activityController.join | 语法 | public function join($id); |
| 前置条件 | 无 |
| 后置条件 | POST方法绑定。若用户不符合加入条件（比如人数已满），就返回活动信息详细页面并提示错误。否则在模型中将此用户加入到这个活动中，并显示加入成功！ |
| activityController.leave | 语法 | public function leave($id); |
| 前置条件 | 若用户为创建者，则显示“关闭活动”选项。若用户为加入者，则显示“退出活动”选项。若用户不为这两种情况之一，不显示以上选项。 |
| 后置条件 | 若用户为创建者，则关闭此活动。若用户为加入者，则退出此活动。 |
| 需要的服务（接口） | | |
| 服务名 | | **服务** |
| UserModel.getPuissance | | 验证权限（管理员可关闭任意活动） |
| ActivityModel.\* | | 管理活动模型 |

### 4.2.3 exerciseController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| exerciseController.index | 语法 | public function index(); |
| 前置条件 | 无 |
| 后置条件 | 显示健身数据列表页面 |
| exerciseController.create | 语法 | public function create(); |
| 前置条件 | 无 |
| 后置条件 | 进入记录健身表单填写页面 |
| exerciseController.createSubmit | 语法 | public function createSubmit(Request $request); |
| 前置条件 | 已经通过laravel的request机制验证表单的各项符合格式规范 |
| 后置条件 | 如果符合要求，就在模型中记录这项健身，否则就返回记录健身表单填写页面并提示错误 |
| exerciseController.edit | 语法 | public function edit(); |
| 前置条件 | 若用户为创建者，则显示“编辑”选项。若用户不为编辑者，不显示编辑选项。 |
| 后置条件 | 进入编辑健身表单填写页面 |
| exerciseController.editSubmit | 语法 | public function editSubmit(Request $request); |
| 前置条件 | 已经通过laravel的request机制验证表单的各项符合格式规范 |
| 后置条件 | 如果符合要求并且**再次验证修改的用户为创建者**（防止利用URL绕过第一次验证），就在模型中修改这项健身，否则就返回记录健身表单填写页面并提示错误 |
| exerciseController.show | 语法 | public function show($id); |
| 前置条件 | 无 |
| 后置条件 | 显示id对应的健身详细信息 |
| exerciseController.like | 语法 | public function like($id); |
| 前置条件 | 若用户已经点赞，点赞按钮设为不可点击模式。 |
| 后置条件 | 若用户尚未点赞，将此次点赞行为加入到模型中去。 |
| exerciseController.delete | 语法 | public function delete($id); |
| 前置条件 | 若用户为创建者，则显示“删除”选项。若用户不为创建者，不显示删除选项。 |
| 后置条件 | 若用户为创建者，则在模型中删除此健身。 |
| 需要的服务（接口） | | |
| 服务名 | | **服务** |
| UserModel.\* | | 验证用户权限 |
| ExerciseModel.\* | | 管理健身数据模型 |

### 4.2.4 friendController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| friendController.chatMain | 语法 | | public function chatMain (); |
| 前置条件 | | 无 |
| 后置条件 | | 显示好友聊天主界面，左侧为好友列表，右侧为聊天页面 |
| friendController.delete | 语法 | | public function delete ($id); |
| 前置条件 | | 两人已经是好友关系 |
| 后置条件 | | 在已登录用户的好友列表内删除id标示的好友。并返回好友聊天主页面。 |
| 需要的服务（接口） | | | |
| 服务名 | | **服务** | |
| FriendshipModel.\* | | 管理好友列表 | |
| DialogModel.\* | | 管理聊天记录 | |

### 4.2.5 blogController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| blogController.index | 语法 | | public function index(); |
| 前置条件 | | 无 |
| 后置条件 | | 显示个人主页，包括健身数据，个人信息，个人统计数据等信息。 |
| blogController.show | 语法 | | public function show($id); |
| 前置条件 | | 无 |
| 后置条件 | | 显示id所对应用户的个人主页 |
| 需要的服务（接口） | | | |
| 服务名 | | **服务** | |
| ExerciseModel.\* | | 显示此人的健身记录 | |
| StatisticsModel.\* | | 显示此人的统计数据 | |
| FriendshipModel.\* | | 判断此人与当前登录用户是否为好友，若为好友，显示删除好友选项，否则显示加为好友选项 | |
| UserModel.\* | | 显示此人的个人信息 | |

### 4.2.6 userController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| userController.index | 语法 | | public function index (); |
| 前置条件 | | 无 |
| 后置条件 | | 显示编辑个人信息主页面 |
| userController.store | 语法 | | public function store ($id); |
| 前置条件 | | 已经通过laravel的request机制验证表单的各项符合格式规范 |
| 后置条件 | | 将用户提交的个人信息编辑存储到模型中 |
| 需要的服务（接口） | | | |
| 服务名 | | **服务** | |
| UserModel.\* | | 查看，编辑个人信息 | |

### 4.2.7 searchController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| searchController.search | 语法 | | public function search ($content); |
| 前置条件 | | 无 |
| 后置条件 | | 将content中的内容，作为关键字搜索：好友动态标题和内容；活动标题和内容；用户名。如果content中的内容与某标签匹配，那么搜索标签 |
| 需要的服务（接口） | | | |
| 服务名 | | **服务** | |
| UserModel.\* | | 查看个人信息 | |
| ActivityModel.\* | | 查看活动信息 | |
| ExerciseModel.\* | | 查看自己及好友健身信息 | |

### 4.2.8 mainPageController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| mainPageController.index | 语法 | public function index (); |
| 前置条件 | 无 |
| 后置条件 | 显示用户的各项统计数据（包括等阶），以及最近的三次健身 |
| 需要的服务（接口） | | |
| 服务名 | | **服务** |
| ExerciseModel.\* | | 管理健身数据模型 |
| StatisticsModel.\* | | 管理统计数据模型 |

# 5．模型（数据库）分解

## 5.1 模型的各模块职责

|  |  |
| --- | --- |
| 开发模块 | 功能 |
| ActivityModel | 记录活动（比如比赛） |
| ExerciseModel | 记录用户单次健身数据，以及每条数据附属的评论 |
| StatisticsModel | 记录每个用户对应的统计数据 |
| FriendshipModel | 记录用户之间的相互好友关系 |
| DialogModel | 保存**好友**之间的聊天记录 |
| UserModel | 记录用户的各项基本信息，也保存密码用于验证用户身份 |

## 5.**2** 模型部分的详细分解

### 5.**2.**1 活动 – ActivityModel

活动模型记录活动（比如比赛）的各项数据：

|  |  |
| --- | --- |
| 属性 | 说明 |
| 标题 | String类型，有最大长度限制。 |
| 开始时间 | 时间类型 |
| 结束时间 | 时间类型 |
| 已参加人数 | InstitutionPO类包含机构的地点，id。 |
| 最大参加人数 | InstitutionPO类包含机构的地点，id。 |
| 活动描述 | 活动属性的核心，为text类型 |
| 发起人 | 外键：用户的id |
| 浏览热度 | 浏览次数，int |
| 已参加者 | 以用户列表的形式体现，即用户ID的list |
| 状态 | 结束，未开始，进行中，关闭（只能由创建用户或管理员关闭） |

### 5.2.2 单次健身数据（以及评论） - ExerciseModel

单次健身数据模型记录单次健身数据（比如一次爬山）的各项数据：

|  |  |
| --- | --- |
| 属性 | 说明 |
| 标题 | String类型，有最大长度限制。 |
| 开始时间 | 时间类型 |
| 运动时长 | 时间段类型，存储小时，分钟。 |
| 运动地点 | String类型 |
| 运动标签 | 便于用户归类 |
| 运动描述 | Text型，长度较大。用于记录心得 |
| 记录人 | 外键：用户的id |
| 预估运动量 | 整形，以卡路里为单位 |
| 评论 | 外键的列表，链接到评论表的id，并且根据时间先后顺序排列 |
| 标签（和上面的运动标签合并） | 外键的列表，链接到标签表的id |
| 点赞人 | 以用户列表的形式体现，即用户ID的list |

链接的评论表：

|  |  |
| --- | --- |
| 属性 | 说明 |
| 内容 | 回复的内容 |
| User\_id | 评论人，外键，链接到用户 |
| 评论时间 | 时间戳 |

链接的标签表：

|  |  |
| --- | --- |
| 属性 | 说明 |
| content | 标签的内容 |

### 5.2.3 统计数据列表 - StatisticsModel

根据用户的健身数据算出，在用户每一次插入健身数据后，由controller计算并更新。

|  |  |
| --- | --- |
| 属性 | 说明 |
| User\_id | 外键，链接到用户 |
| Exercise\_hour | 时间类型，单位为分钟，但只显示小时 |
| Continuous\_exercise\_day | 天数，若有一天用户没有锻炼，则重设为0 |
| Overall\_exercise\_day | 用户总共锻炼的天数 |
| calories | 由用户预估的卡路里计算并统计 |
| Grade | int类型，另外在常量里定义数值与特殊字符串（比如“小试牛刀”等阶）的对应关系 |
| Update\_at | 时间戳，记录最后一次运动的时间，有助于统计连续锻炼天数 |

### 5.**2.**4 好友关系 - FriendshipModel

好友列表记录用户之间的相互关联：

|  |  |
| --- | --- |
| 属性 | 说明 |
| User\_id\_1 | 甲方，数值更小 |
| User\_id\_2 | 乙方，数值更大 |

注：这两个属性的顺序无关紧要，在逻辑上是等价的。我们规定，两者的ID，在**数值**上更小的一方被分为甲方，在**数值**上更大的一方被分为乙方（不存在数值相等的情况）。

### 5.2.5 聊天记录 - DialogModel

两位好友之间的聊天记录，需要存储比较长的时间（可以在一段时间后删除，但本系统由于不考虑对硬盘的占用情况，因此不进行删除）。每一条数据表示一条信息。

|  |  |
| --- | --- |
| 属性 | 说明 |
| Friendship\_id | 外键，链接到好友列表中的一对好友 |
| Time | 时间戳 |
| Sender\_id | 外键，链接到用户列表。为两位好友的一位，为了与接收者相区分。 |
| content | Text类型，长度不定 |

### 5.2.6 用户 - UserModel

包括用户权限和个人信息两个部分

|  |  |
| --- | --- |
| 属性 | 说明 |
| Name | String类型,昵称 |
| Email | **登录依据**，应当经过格式检查 |
| Phone | String类型 |
| Qq | String类型 |
| Signature | 个性签名，Text格式，长度不定 |
| Image | Img格式， |
| Password | **登录依据**，经过加密 |
| clout | 分为用户和管理员，String类型 |

# 6. 数据格式定义

注入运动数据：

<status>传输状态，ok 或是 fail </status>

<data>

<title>这是标题</title>

<location>这是运动地点</ location >

<startTime>健身开始时间，如果传入值不合法或为空，系统自动打上传入时的时间戳</ startTime >

<exerciseMin>运动时长，单位为分钟，比如1小时32分钟，就输入92</ exerciseMin >

<calories>这是运动消耗的卡路里</ calories >

<kind>运动类型</kind>

<userid>用户id</userid>

</data>