

SportsZone运动社交

详细设计文档



2016-11-1

南京大学软件学院

张文玘

学号：141250192

| 修改人员 | 修改目的 | 更新日期 | 版本号 |
| --- | --- | --- | --- |
| 张文玘 | 完成文档初稿 | 2015-11-01 | V1.0 |

目录

[1引言 3](#_Toc465962174)

[1.1 编制目的 3](#_Toc465962175)

[1.2 词汇表 3](#_Toc465962176)

[1.3参考资料 3](#_Toc465962177)

[2.总体设计 3](#_Toc465962178)

[2.1开发环境 3](#_Toc465962179)

[2.2模块结构 4](#_Toc465962180)

[2.3系统设计模式 4](#_Toc465962181)

[2.4软件架构 5](#_Toc465962182)

[3导航设计 5](#_Toc465962183)

[3.1整体导航设计 5](#_Toc465962184)

[4.详细设计 5](#_Toc465962185)

[5.交互设计 5](#_Toc465962186)

# 1引言

## 1.1 编制目的

本文档是SportsZone运动社交系统的详细设计文档，用于指导后续的软件开发，实现与测试及用户的沟通。

本文档说明的内容可能在项目实施过程中发生变更，但是必须由项目执行者仔细分析最终决定，建立持续有效的版本控制。

## 1.2 词汇表

无

## 1.3参考资料

1）IEEE标准

2）SportsZone运动社交系统需求文档

# 2.总体设计

## 2.1开发环境

本系统采用php作为主要开发语言，服务端主要使用php+sqlite+Apache，客户端使用html+css+js。用Apache作为服务器，采用sqlite作为后台数据管理系统。

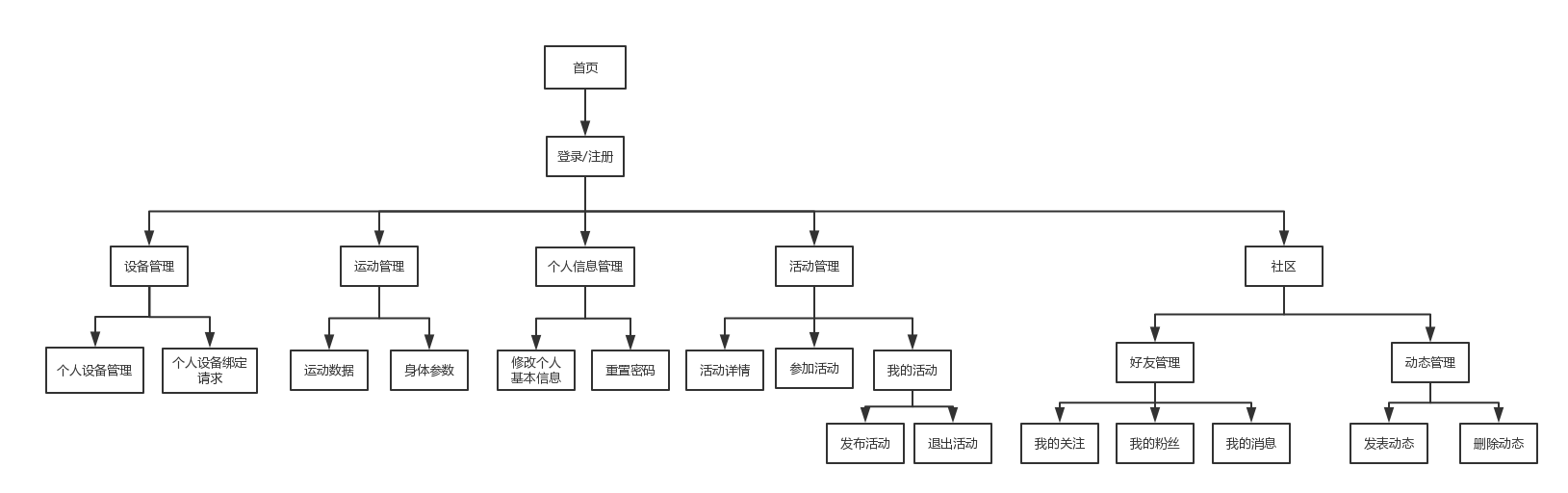
开发环境：windows 10

开发工具：phpstorm

测试浏览器：chorme

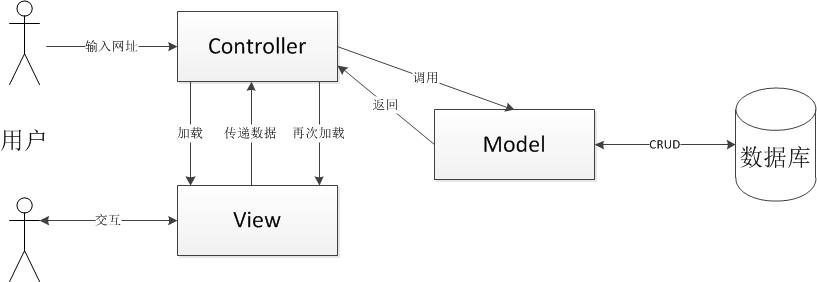
服务器：Apache2.4

## 2.2模块结构

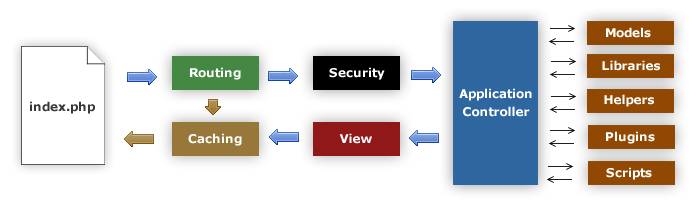


## 2.3系统设计模式

本系统采用CodeIgnitor框架进行开发，基于MVC设计模式



CI基本框架：



1. index.php 文件作为前端控制器，初始化运行CodeIgniter 所需的基本资源；
2. Router 检查 HTTP 请求，以确定如何处理该请求；
3. 如果存在缓存文件，将直接输出到浏览器，不用走下面正常的系统流程；
4. 在加载应用程序控制器之前，对 HTTP 请求以及任何用户提交的数据进行安全检查；
5. 控制器加载模型、核心类库、辅助函数以及其他所有处理请求所需的资源；
6. 最后一步，渲染视图并发送至浏览器，如果开启了缓存，视图被会先缓存起来用于后续的请求。

## 2.4软件架构

Restful架构

# 3导航设计

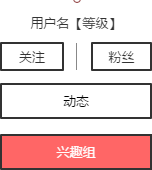
## 3.1整体导航设计



## 3.2 活动导航设计



## 3.3 社区导航设计



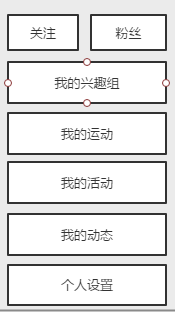
#### 3.3.1 兴趣组导航



## 3.4 运动导航设计



## 3.5 个人中心导航设计



#### 3.5.1 个人设置导航设计



# 4.详细设计

## 4.1登录/注册

#### 4.1.1提供的接口设计

|  |  |  |
| --- | --- | --- |
| 登录请求 | url | /Login |
| 方法类型 | Post |
| 控制器 | AuthController |
| 方法 | Login(Request$request) |
| 参数 | 用户名，密码 |
| 前置条件 | 用户未登录 |
| 后置条件 | 返回登录信息，失败返回失败提示信息 |
| 注册请求 | url | /SignUp |
| 方法类型 | Post |
| 控制器 | AuthController |
| 方法 | Register(Request$request) |
| 参数 | 邮箱，用户名，密码 |
| 前置条件 | 用户未登录 |
| 后置条件 | 返回注册结果 |

#### 4.1.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::create | 创建用户并加入数据库 |
| User::find | 查找用户 |
| User::save | 保存用户信息并更新数据库 |

## 4.2 活动管理

#### 4.2.1 提供的接口设计

|  |  |  |
| --- | --- | --- |
| 活动界面 | url | /Activity |
| 方法类型 | Get |
| 控制器 | ActivityController |
| 方法 | getActivityList() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回活动界面 |
| 活动详情 | url | /Activity |
| 方法类型 | Get |
| 控制器 | ActivityController |
| 方法 | getActivity(Request$request) |
| 参数 | 活动id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回活动详情 |
| 发布活动 | url | /Activity |
| 方法类型 | Post |
| 控制器 | ActivityController |
| 方法 | createActivity(Request$request) |
| 参数 | 活动信息（包括名称、简介、时间、类型） |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回发布结果 |
| 参与活动 | url | /Activity |
| 方法类型 | Post |
| 控制器 | ActivityController |
| 方法 | ParticipateIn(Request$request) |
| 参数 | 活动id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回参与结果 |

#### 4.2.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Activity::getActivityList | 获取活动列表 |
| Activity::getActivity | 获取对应活动的活动详情 |
| Activity::createActivity | 新建活动并将该活动信息更新到数据库中 |
| Activity::participateIn | 参与活动，更新活动参与者列表与用户个人活动列表 |

## 4.3 社区

#### 4.3.1 提供的接口设计

|  |  |  |
| --- | --- | --- |
| 动态界面 | url | /trend |
| 方法类型 | Get |
| 控制器 | TrendController |
| 方法 | getTrendsList() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回该用户已关注用户和互相关注的用户的动态 |
| 点赞动态 | url | /Trend |
| 方法类型 | Post |
| 控制器 | TrendController |
| 方法 | Register(Request$request) |
| 参数 | 动态id |
| 前置条件 | 用户已登录且给某条动态点赞 |
| 后置条件 | 系统更新动态信息 |

#### 4.3.2 需要的接口

## 4.4 运动管理

#### 4.4.1 提供的接口设计

#### 4.4.2 需要的接口

## 4.5 个人管理

#### 4.5.1 提供的接口设计

#### 4.5.2 需要的接口

## 4.6 好友管理

#### 4.6.1 提供的接口设计

#### 4.6.2 需要的接口

## 4.7 兴趣组管理

#### 4.7.1 提供的接口设计

#### 4.7.2 需要的接口

# 5.交互设计