2016-11-5

程翔 141250018

南京大学软件学院

SPORTS

详细设计文档

## 更新历史

|  |  |  |
| --- | --- | --- |
| 时间 | 修改人 | 备注 |
| 2016-10-24 | 程翔 | 创建文档 |
| 2016-10-25 | 程翔 | 完善文档 |
|  |  |  |

目录

[更新历史 1](#_Toc465203892)

[1. 引言 3](#_Toc465203893)

[1.1. 编制目的 3](#_Toc465203894)

[1.2. 参考资料 3](#_Toc465203895)

[2. 产品描述 3](#_Toc465203896)

[3. 分层体系结构 3](#_Toc465203897)

[4. 导航设计 4](#_Toc465203898)

[4.1. 导航内容 4](#_Toc465203899)

[4.2. UI原型界面 4](#_Toc465203900)

[5. 人机交互设计 7](#_Toc465203901)

[5.1. 人机交互界面 7](#_Toc465203902)

[5.2. 人机交互界面转换 8](#_Toc465203903)

[6. 接口设计 8](#_Toc465203904)

[6.1. 业务逻辑层的分解 8](#_Toc465203905)

[6.1.1. 业务逻辑层模块职责 8](#_Toc465203906)

[6.1.2. 业务逻辑层模块的接口规范 9](#_Toc465203907)

[6.2. 数据层的分解 15](#_Toc465203908)

[6.2.1. 数据层模块的职责 15](#_Toc465203909)

[6.2.2. 数据层模块的接口规范 15](#_Toc465203910)

[7. 架构与组件设计 18](#_Toc465203911)

[7.1. 系统架构 18](#_Toc465203912)

[7.2. 组件设计 19](#_Toc465203913)

[7.2.1. 图标（采用font awesome图标系统） 19](#_Toc465203914)

[7.2.2. 动态 19](#_Toc465203915)

[7.2.3. 按钮 19](#_Toc465203916)

[7.2.4. 侧边栏 20](#_Toc465203917)

[7.2.5. 图表 20](#_Toc465203918)

[7.2.6. 竞赛 20](#_Toc465203919)

[7.2.7. 分页 21](#_Toc465203920)

[7.2.8. 搜索框 21](#_Toc465203921)

# 引言

## 编制目的

本文档详细完成对SPORTS的概要设计，达到指导开发的目的，同时实现测试人员及用户的沟通。

本文档面向开发人员，测试人员及最终用户编写，是了解系统的导航。

## 参考资料

* 《Web程序设计 第8版》
* 《SPORTS需求文档》

# 产品描述

参考SPORTS需求文档对产品的概述描述

# 分层体系结构

SPORTS中，选择了分层体系结构的风格，将系统分为三层（展示层、业务逻辑层、数据层，参见5中人机交互的界面，参见6中接口设计的业务逻辑层和数据层的分解）能够很好的示意整个高层抽象。展示包括GUI界面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的读取。分层体系结构的逻辑视角如图1所示。

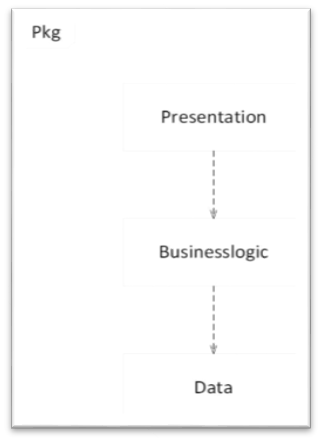


图 1

# 导航设计

## 导航内容

|  |  |
| --- | --- |
| 导航栏 | 侧边栏 |
| 首页 | 个人运动、竞赛信息、统计数据、我的健康 |
| 动态 | 无 |
| 竞赛 | 全部竞赛、发布竞赛 |
| 分析&建议 | 数据分析、运动推荐 |
| 组织 | 组织信息 |
| 个人 | 个人基本信息、个人运动信息 |

## UI原型界面

UI界面原型图如下图2-图6所示



图 2

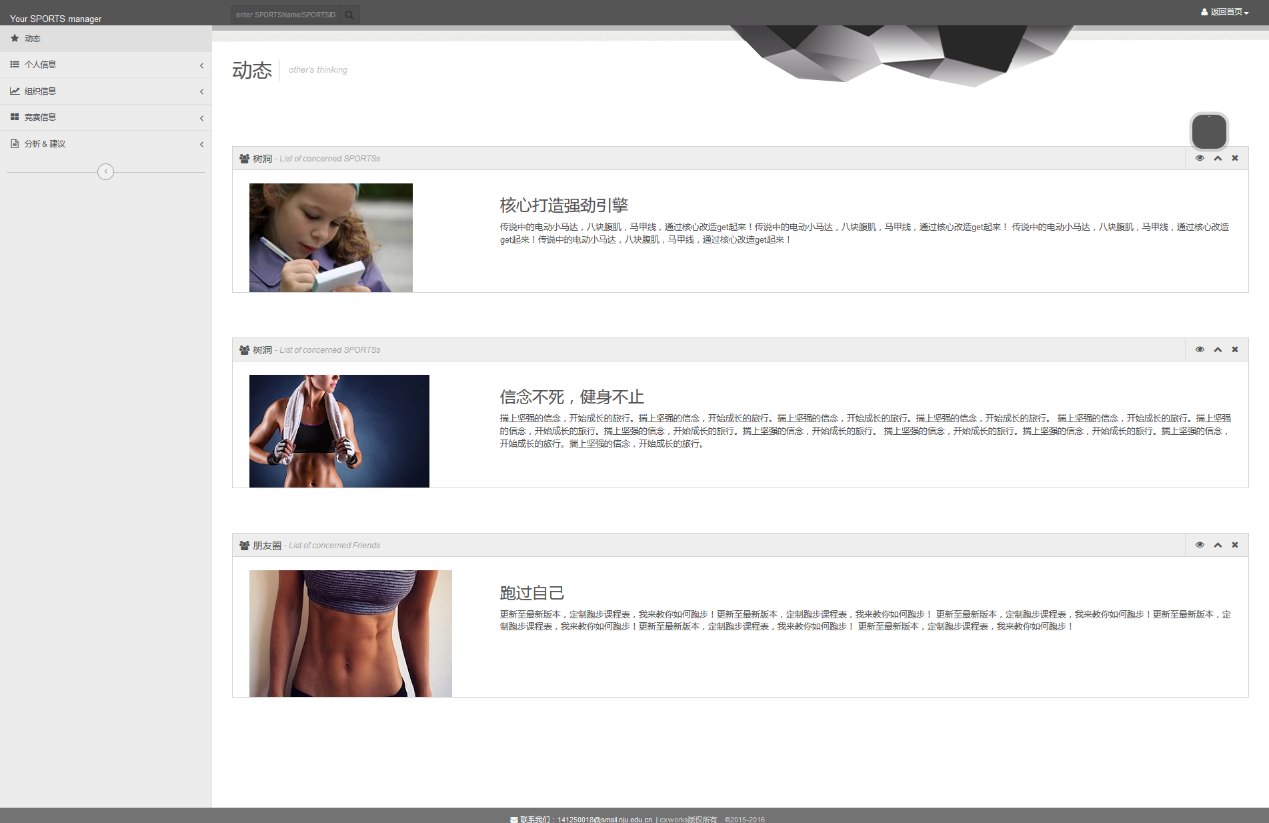


图 3



图 4

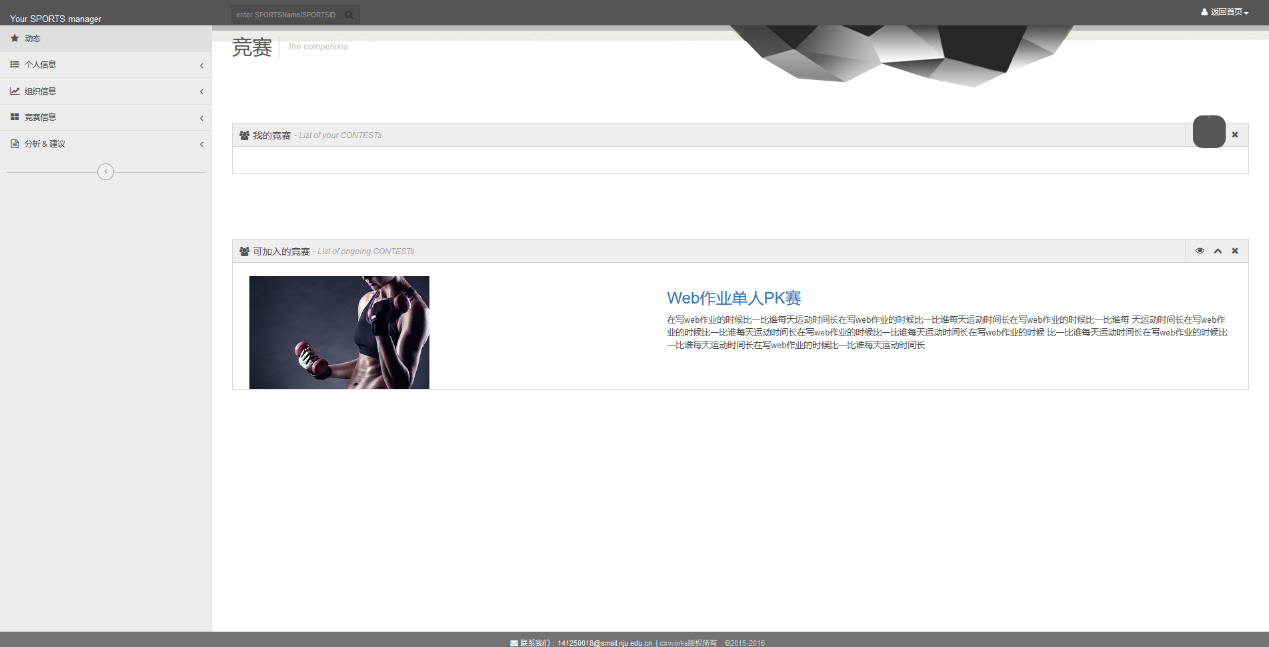


图 5

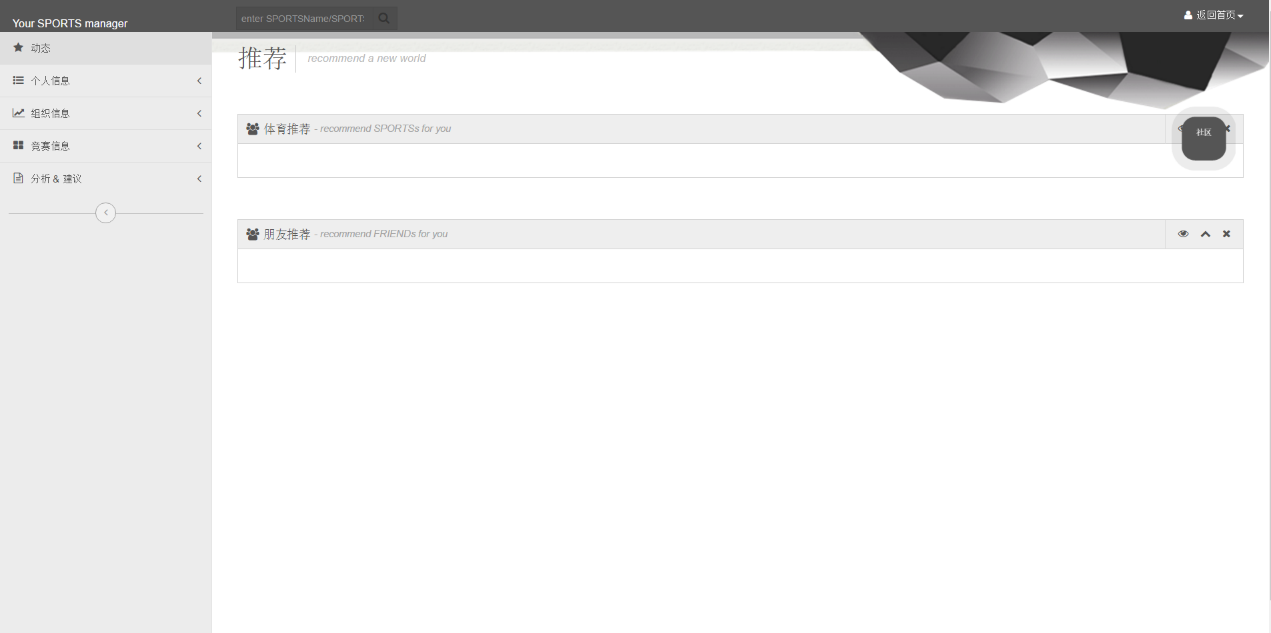


图 6

# 人机交互设计

## 人机交互界面

存在16个界面，分别是登陆、注册界面、首页、竞赛（全部竞赛、我参与的、发布竞赛）、动态（好友动态、发布动态、树洞）、个人（基本信息、运动信息）、分析&推荐（数据分析、成长等级、运动推荐）

## 人机交互界面转换

人机交互界面转换如下图：

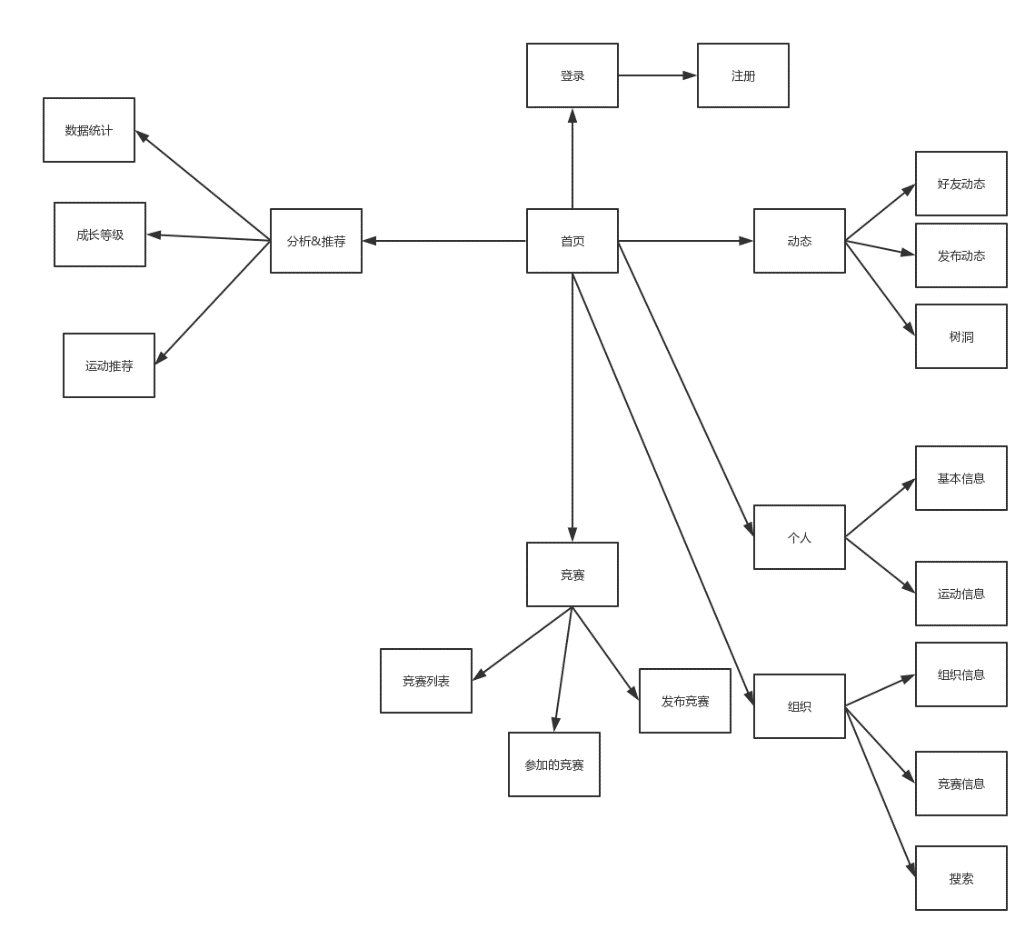


图 7

# 接口设计

## 业务逻辑层的分解

### 业务逻辑层模块职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| index | 负责首页界面所需要的服务 |
| contest | 负责竞赛界面所需要的服务 |
| news | 负责动态界面所需要的服务 |
| user | 负责个人信息所需要的服务 |
| dataanalyze | 负责分析&推荐界面所需要的服务 |

### 业务逻辑层模块的接口规范

#### user模块

* 模块概述

user模块负责用户的登陆注册，设置个人资料，读取用户账户好友动态，对账户动态好友进行更改。具体功能需求和非功能需求课参见SPORTS需求文档。

* user接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| user.login | 语法 | function login($username, $password) |
| 前置条件 | username和password不为空 |
| 后置条件 | 验证用户名和密码是否匹配 |
| user.register | 语法 | function register($user) |
| 前置条件 | 用户填写的必要信息不为空，且符合语法规范 |
| 后置条件 | 记录新用户的数据 |
| user.get\_my\_news | 语法 | function get\_my\_news ($user\_id) |
| 前置条件 | 用户id必须存在 |
| 后置条件 | 获得这个用户的全部动态 |
| user.get\_info | 语法 | function get\_info ($user\_id) |
| 前置条件 | 用户id必须存在 |
| 后置条件 | 获得这个用户的全部基本资料 |
| user.get\_friends | 语法 | function get\_friends ($user\_id) |
| 前置条件 | 用户id和account\_id必须存在 |
| 后置条件 | 获得这个用户的全部好友 |
| user.get\_account | 语法 | function get\_account($user\_id, $account\_id) |
| 前置条件 | 用户id必须存在 |
| 后置条件 | 获得这个用户的账户 |
| user.update\_info | 语法 | function update\_info ($user\_id) |
| 前置条件 | 用户id必须存在 |
| 后置条件 | 更新这个用户的基本资料 |
| user.add\_friends | 语法 | function add\_friends ($user\_id) |
| 前置条件 | 用户id必须存在 |
| 后置条件 | 用户添加一个好友 |
| user.delete\_friends | 语法 | function delete\_friends ($user\_id) |
| 前置条件 | 用户id必须存在 |
| 后置条件 | 删除这个用户的一个好友 |
| user.get\_all\_news | 语法 | function get\_all\_news () |
| 前置条件 | 无 |
| 后置条件 | 所有动态信息 |
| user.\_friends\_news | 语法 | function get\_friends\_news ($user\_id) |
| 前置条件 | 用户已经登录 |
| 后置条件 | 用户好友的动态信息 |
| user.update\_account | 语法 | function update\_account($user\_id, $account\_id) |
| 前置条件 | 用户id和account\_id必须存在 |
| 后置条件 | 更新此用户的account |

* 业务逻辑层的动态模型

如下图显示在用户登录的时候用户逻辑处理的相关对象之间的协作。

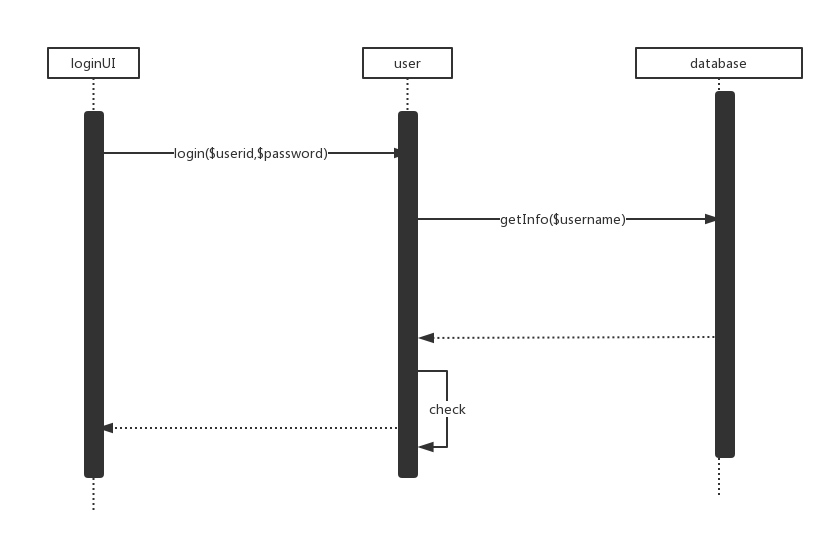


图 8

#### index模块

* 模块概述

提供首页中显示运动信息、统计信息、动态信息，具体功能需求和非功能需求课参见SPORTS需求文档

* index接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| index.get\_user | 语法 | function get\_user() |
| 前置条件 | 无 |
| 后置条件 | 获得注册用户人数 |
| index.get\_sports | 语法 | function get\_sports() |
| 前置条件 | 无 |
| 后置条件 | 获取当前运动记录数 |
| index.get\_news | 语法 | function get\_news() |
| 前置条件 | 无 |
| 后置条件 | 获取动态总数 |

* 业务逻辑层的动态模型

如下图显示了获得全部动态的时候，业务逻辑处理的相关对象之间的协作。

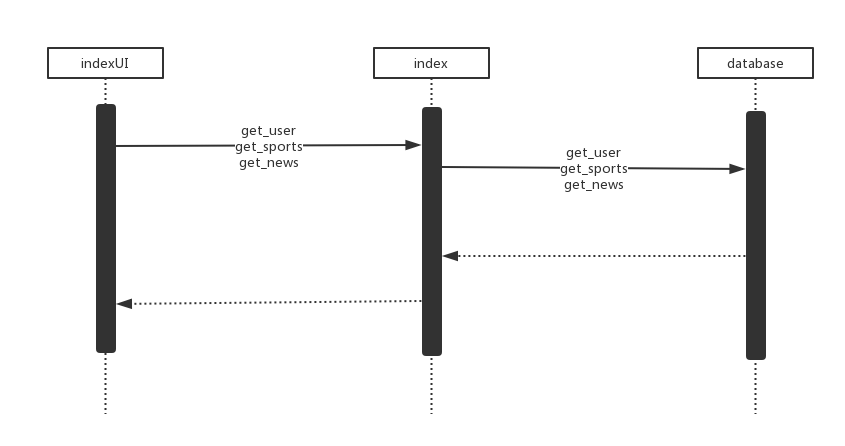


图 9

#### contest模块

* 模块概述

contest模块提供显示全部竞赛、发起竞赛、我的竞赛功能，具体功能需求和非功能需求课参见SPORTS需求文档

* contest接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| contest.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的竞赛 |
| contest.my\_ participate | 语法 | function my\_ participate () |
| 前置条件 | 用户成功登陆 |
| 后置条件 | 获得用户参与的竞赛 |
| contest.add | 语法 | function add($contest) |
| 前置条件 | 用户成功登陆，用户输入合法 |
| 后置条件 | 记录组织发布的竞赛 |
| contest.get\_history | 语法 | function get\_history () |
| 前置条件 | 无 |
| 后置条件 | 获得已经的历史竞赛 |
| contest.find | 语法 | function find($key，$id) |
| 前置条件 | 用户输入的key符合条件 |
| 后置条件 | 根据用户输入或者准确id查找比赛 |

* 业务逻辑层的动态模型

如图表明了SPORTS中，当添加一次竞赛的时候，业务逻辑处理的相关对象之间的协作。

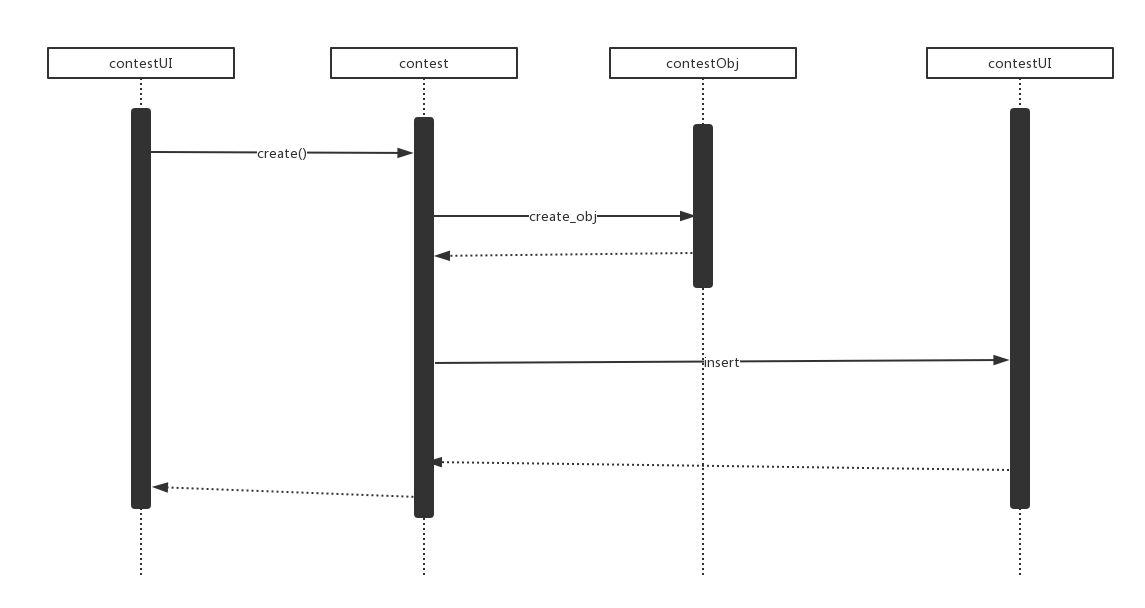


图 10

#### news模块

* 模块概述

news模块提供显示全部动态、发起动态、删除动态，搜索动态、朋友动态、我的动态功能，具体功能需求和非功能需求课参见SPORTS需求文档

* news接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| news.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的动态 |
| news.my\_ participate | 语法 | function my\_ participate () |
| 前置条件 | 用户成功登陆 |
| 后置条件 | 获得用户参与的动态 |
| news.add | 语法 | function add ($news) |
| 前置条件 | 管理员成功登陆，用户输入合法 |
| 后置条件 | 记录用户发布的动态 |
| news.delete | 语法 | function delete ($news\_id) |
| 前置条件 | 用户成功登陆 |
| 后置条件 | 删除用户发起的竞赛 |
| news.get\_history | 语法 | function get\_history () |
| 前置条件 | 无 |
| 后置条件 | 获得已经过期的历史动态 |
| news.find | 语法 | function find($key，$id) |
| 前置条件 | 用户输入的key符合条件 |
| 后置条件 | 根据用户输入或者准确id查找动态 |
| news.friends | 语法 | function friends($username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 显示用户好友的所有动态 |
|  |  |  |

* 业务逻辑层的动态模型

如图表明了SPORTS中，当添加一次动态的时候，业务逻辑处理的相关对象之间的协作。

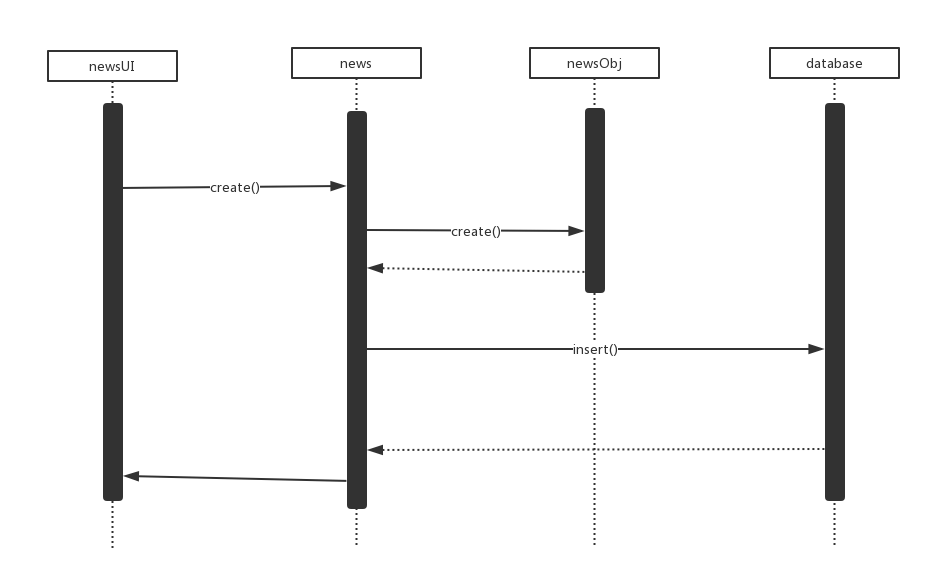


图 11

#### dataanalyze模块

* 模块概述

dataanalyze模块提供显示全部数据记录、数据分析、成长轨迹、运动推荐等功能，具体功能需求和非功能需求课参见SPORTS需求文档

* dataanalyze接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| dataanalyze.my\_history | 语法 | function my\_history ($username) |
| 前置条件 | 用户已经登录 |
| 后置条件 | 提供全部的运动数据 |
| dataanalyze.my\_ participate | 语法 | function my\_ participate ($username) |
| 前置条件 | 用户成功登陆 |
| 后置条件 | 获得用户参与的竞赛 |
| dataanalyze.my\_level | 语法 | function my\_level ($username) |
| 前置条件 | 用户成功登陆 |
| 后置条件 | 记录用户的经验记录 |
| dataanalyze.analyze | 语法 | function analyze ($username) |
| 前置条件 | 用户成功登陆 |
| 后置条件 | 返回基于用户数据的分析 |
| dataanalyze.recommend | 语法 | function recommend($username) |
| 前置条件 | 用户已经登录 |
| 后置条件 | 根据用户根据用户数据的推荐内容 |

* 业务逻辑层的动态模型

如图表明了SPORTS中，当添加一次兴趣组的时候，业务逻辑处理的相关对象之间的协作。

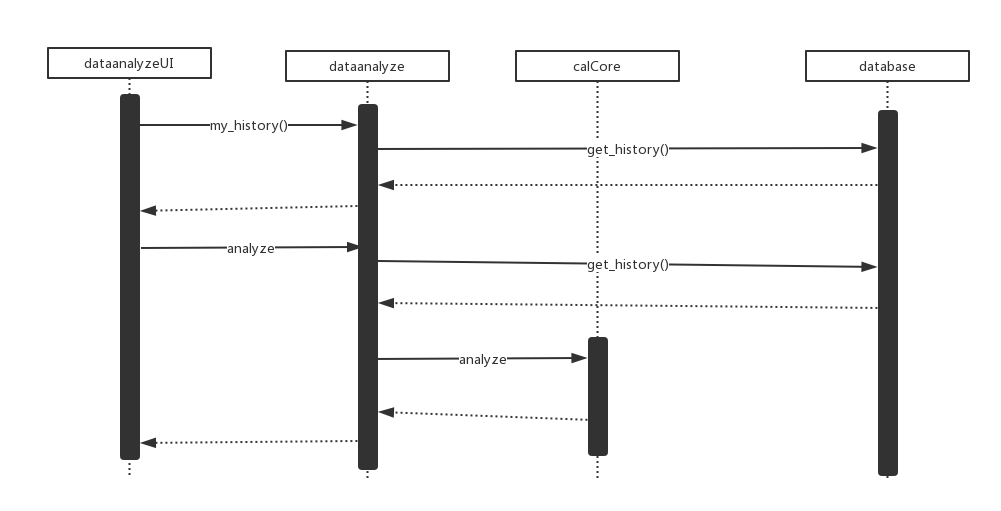


图 12

## 数据层的分解

### 数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| userdata | 负责读取用户个人资料，好友，账户，通知，动态 |
| sportsdata | 负责读取全部运动记录 |
| newsdata | 负责读取全部动态，提供增删改查功能 |
| contestdata | 负责读取全部竞赛，提供增删改查功能 |

### 数据层模块的接口规范

#### userdata模块

* **userdata接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| userdata.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的信息 |
| userdata.get\_info | 语法 | function get\_info($userid) |
| 前置条件 | user\_id存在 |
| 后置条件 | 提供用户的基本信息 |
| userdata.set\_info | 语法 | function set\_info($user\_id, $user) |
| 前置条件 | 用户成功登陆，输入信息格式规范 |
| 后置条件 | 修改用户的信息 |
| userdata.get\_friends | 语法 | function get\_friends($user\_id) |
| 前置条件 | user\_id存在 |
| 后置条件 | 获得用户的好友 |
| userdata.add\_friends | 语法 | function add\_friends($user\_id, $friend\_id) |
| 前置条件 | user\_id和friend\_id存在 |
| 后置条件 | 添加好友 |
| userdata.delete\_friends | 语法 | function delete\_friends($user\_id, $friend\_id) |
| 前置条件 | user\_id和friend\_id存在 |
| 后置条件 | 删除好友 |
| userdata.get\_news | 语法 | function get\_news($user\_id) |
| 前置条件 | user\_id存在 |
| 后置条件 | 获得用户的动态 |
| userdata.get\_notify | 语法 | function get\_notify($user\_id) |
| 前置条件 | user\_id存在 |
| 后置条件 | 获得用户的通知 |

#### sportsdata模块

* **sportsdata接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| sportsdata.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的兴趣组 |
| sportsdata.find | 语法 | function find($id) |
| 前置条件 | id输入合法 |
| 后置条件 | 根据id查找兴趣组 |
| sportsdata.add | 语法 | function add($interest) |
| 前置条件 | 用户输入格式正确 |
| 后置条件 | 插入一个兴趣组到数据库 |
| sportsdata.delete | 语法 | function delete($id) |
| 前置条件 | 删除的兴趣组为此用户发起的 |
| 后置条件 | 根据id删除兴趣组 |
| sportsdata.update | 语法 | function update($id) |
| 前置条件 | 更改的兴趣组为此用户发起的 |
| 后置条件 | 根据id更新兴趣组 |

#### newsdata模块

* **newsdata接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| newsdata.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的活动 |
| newsdata.find | 语法 | function find($id) |
| 前置条件 | id输入合法 |
| 后置条件 | 根据id查找活动 |
| newsdata.add | 语法 | function add($interest) |
| 前置条件 | 用户输入格式正确 |
| 后置条件 | 插入一个活动到数据库 |
| newsdata.delete | 语法 | function delete($id) |
| 前置条件 | 删除的活动为此用户发起的 |
| 后置条件 | 根据id删除活动 |
| newsdata.update | 语法 | function update($id) |
| 前置条件 | 更改的活动为此用户发起的 |
| 后置条件 | 根据id更新活动 |

#### contestdata模块

* **contestdata接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| contestdata.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的竞赛 |
| contestdata.find | 语法 | function find($id) |
| 前置条件 | id输入合法 |
| 后置条件 | 根据id查找竞赛 |
| contestdata.add | 语法 | function add($interest) |
| 前置条件 | 用户输入格式正确 |
| 后置条件 | 插入一个竞赛到数据库 |
| contestdata.delete | 语法 | function delete($id) |
| 前置条件 | 删除的竞赛为此用户发起的 |
| 后置条件 | 根据id删除活动 |

# 架构与组件设计

## 系统架构

本系统为动态网站，采取BS模式，利用apache或nginx做服务器，由php操纵sqlite数据库，获得所需数据。具体工作流程如下：

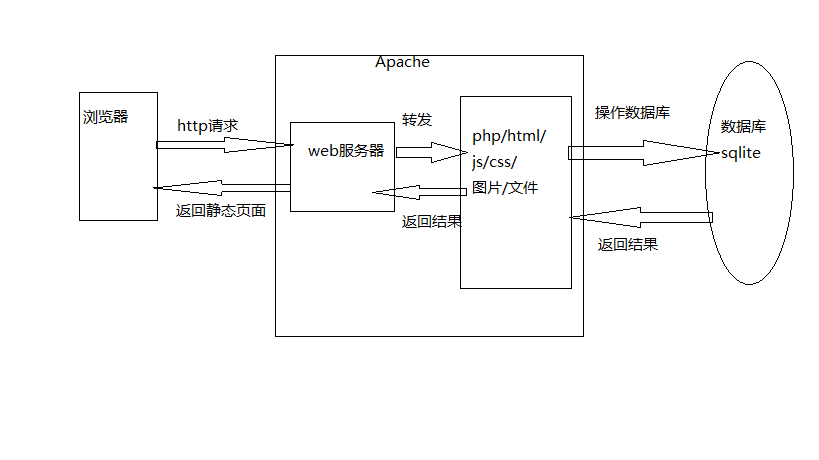


图 13

组件样式采用bootstrap中的组件设计，并对其中部分组件进行更改，符合系统的风格特性。

## 组件设计

### 图标（采用font awesome图标系统）

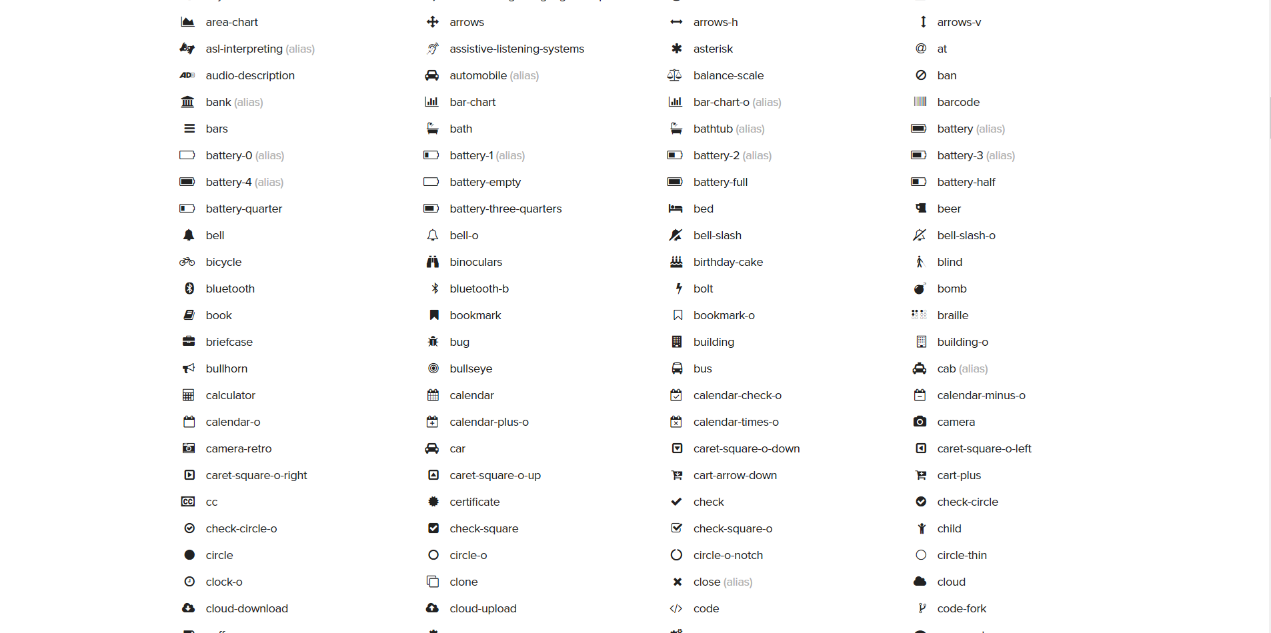


图 14

### 动态

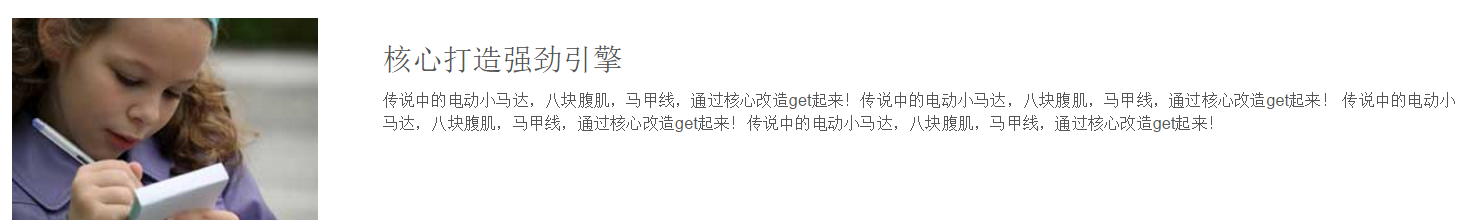


图 15

### 按钮

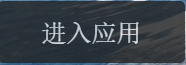


图 16

### 侧边栏



图 17

### 图表

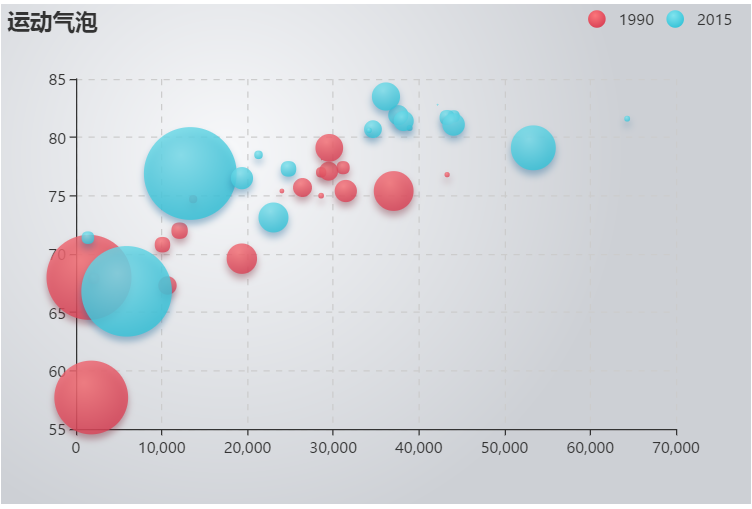


图 18

### 竞赛



图 19

### 分页



图 20

### 搜索框



图 21