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 |
|  |  |  |

# 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。

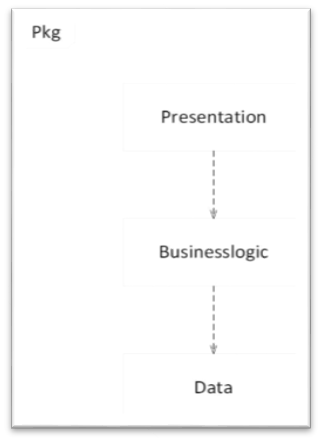


图 1

# 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

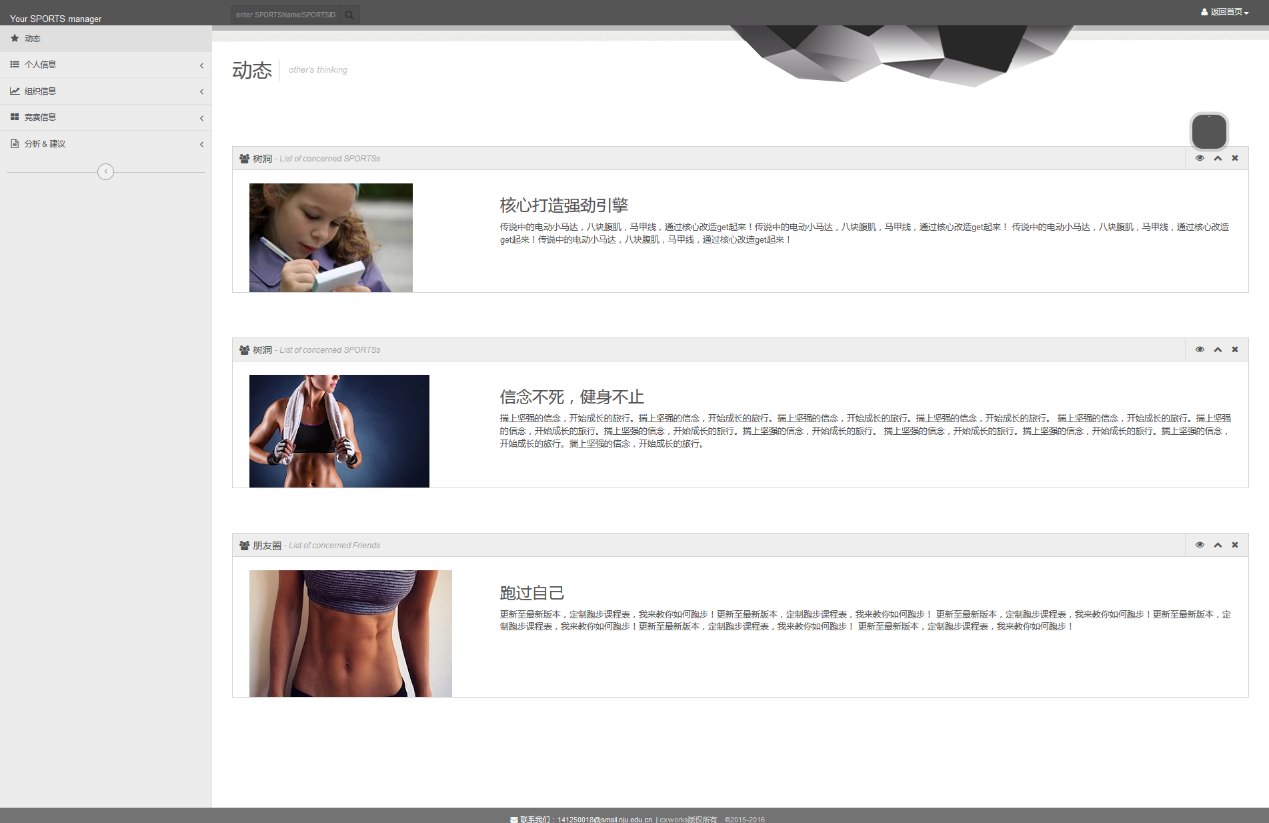


图 3



图 4

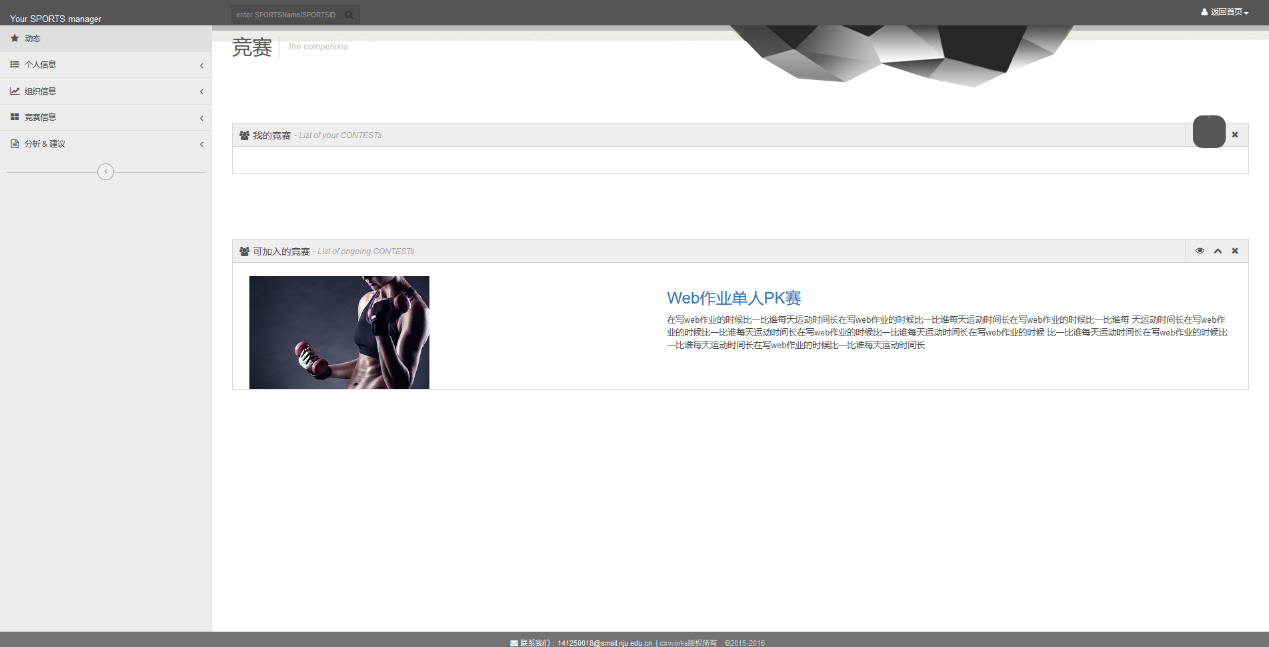


图 5

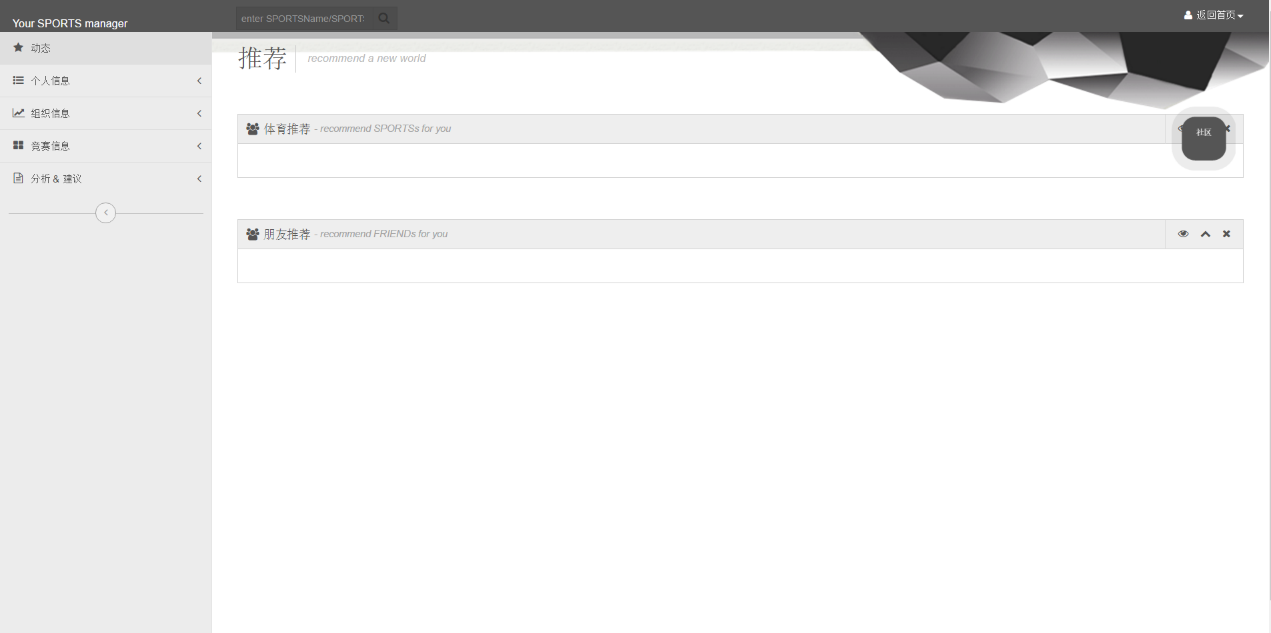


图 6

# 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：

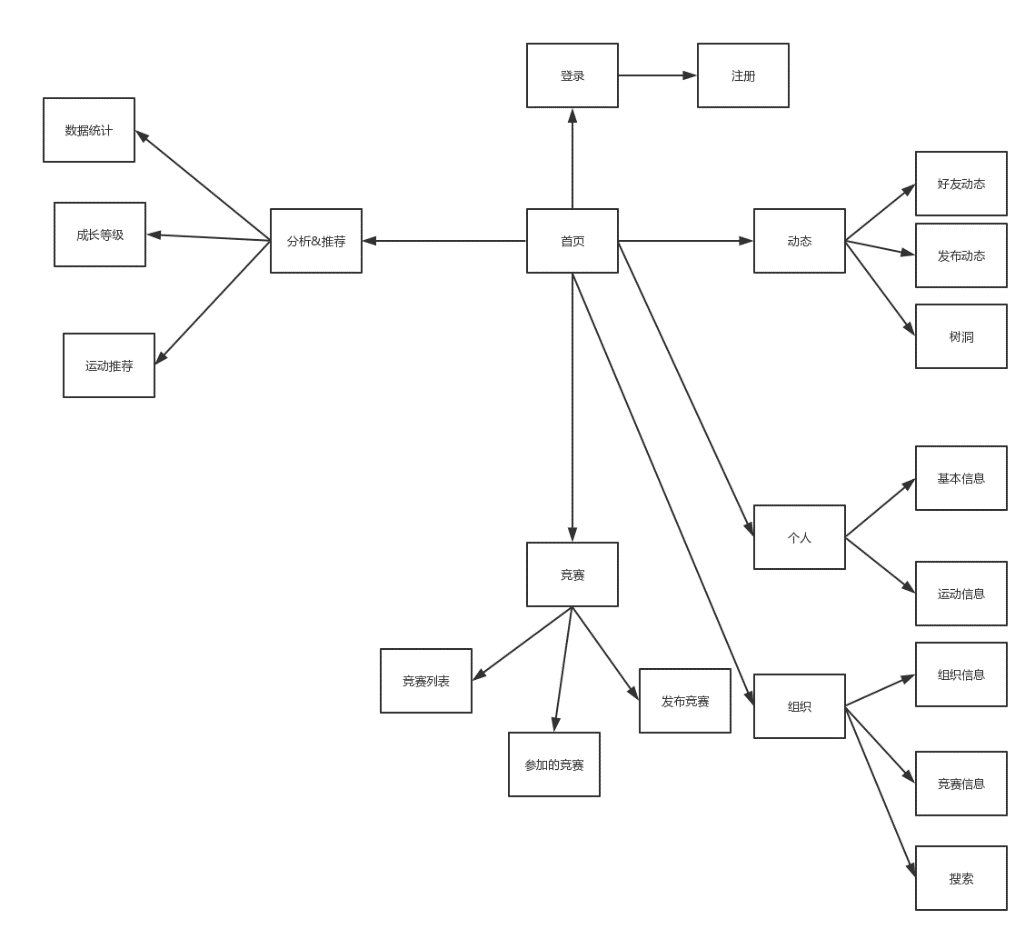


图 7

# 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。

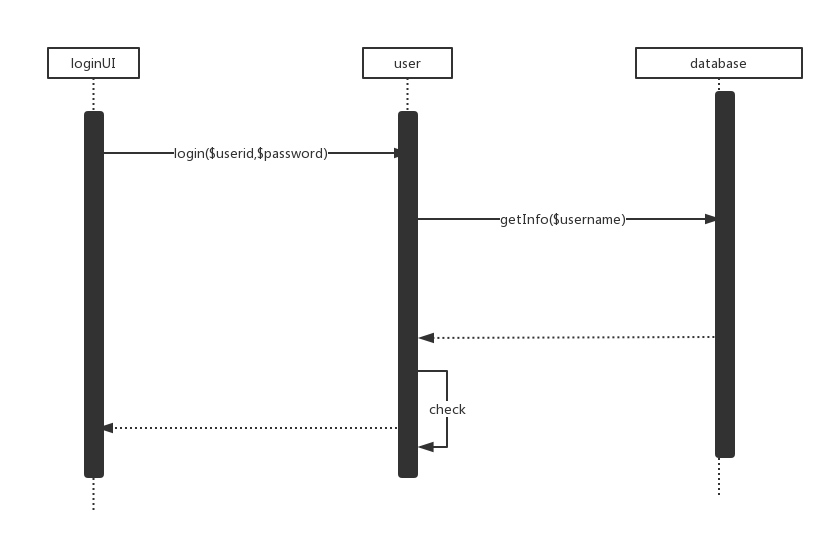


图 8

#### 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健康专家系统需求文档

* 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。

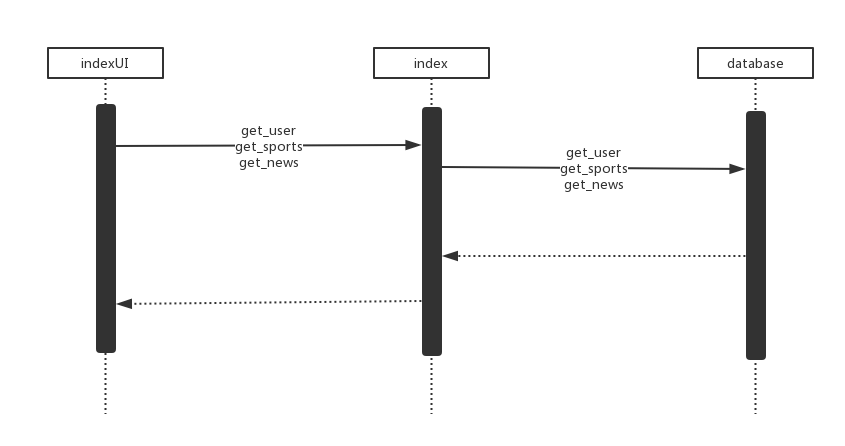


图 9

#### 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健康专家系统需求文档

* 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($contest) |
| 前置条件 | 用户成功登陆，用户输入合法 |
| 后置条件 | 记录用户发布的竞赛 |
| contest.delete | 语法 | function delete ($contest\_id) |
| 前置条件 | 要删除的比赛是此用户发起的 |
| 后置条件 | 删除用户发起的竞赛 |
| contest.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

如图表明了健康专家系统中，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。

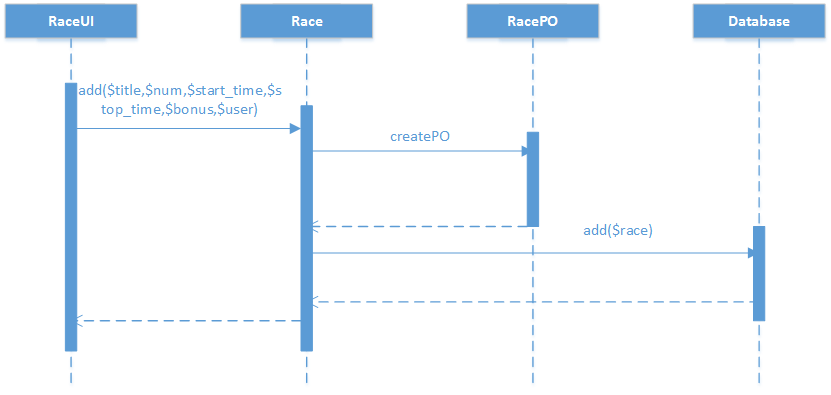


图 10