

系统详细设计文档

健康专家



2015-11-9

更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 陈云龙 | 2015-11-9 | 项目系统详细设计文档 | V1.0 |
| 陈云龙 | 2015-11-10 | 完善文档 | V2.0 |
|  |  |  |  |

目录

[1. 引言 3](#_Toc434958440)

[1.1. 编制目的 3](#_Toc434958441)

[1.2. 参考资料 3](#_Toc434958442)

[2. 产品描述 3](#_Toc434958443)

[3. 分层体系结构 3](#_Toc434958444)

[4. 导航设计 4](#_Toc434958445)

[4.1. 导航内容 4](#_Toc434958446)

[4.2. UI原型界面 4](#_Toc434958447)

[5. 人机交互设计 7](#_Toc434958448)

[5.1. 人机交互界面 7](#_Toc434958449)

[5.2. 人机交互界面转换 7](#_Toc434958450)

[6. 接口设计 8](#_Toc434958451)

[6.1. 业务逻辑层的分解 8](#_Toc434958452)

[6.1.1. 业务逻辑层模块职责 8](#_Toc434958453)

[6.1.2. 业务逻辑层模块的接口规范 8](#_Toc434958454)

[6.2. 数据层的分解 15](#_Toc434958455)

[6.2.1. 数据层模块的职责 15](#_Toc434958456)

[6.2.2. 数据层模块的接口规范 16](#_Toc434958457)

[7. 架构与组件设计 18](#_Toc434958458)

[7.1. 系统架构 18](#_Toc434958459)

[7.2. 组件设计 20](#_Toc434958460)

[7.2.1. 图标 20](#_Toc434958461)

[7.2.2. 导航栏 20](#_Toc434958462)

[7.2.3. 按钮 20](#_Toc434958463)

[7.2.4. 侧边栏 21](#_Toc434958464)

[7.2.5. 进度条 21](#_Toc434958465)

[7.2.6. 输入框 21](#_Toc434958466)

[7.2.7. 分页 22](#_Toc434958467)

[7.2.8. 搜索框 22](#_Toc434958468)

# 引言

## 编制目的

本文档详细完成对健康专家系统的概要设计，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》
* 《健康专家系统需求文档》

# 产品描述

参考健康专家系统需求文档对产品的概述描述

# 分层体系结构

健康专家系统中，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。

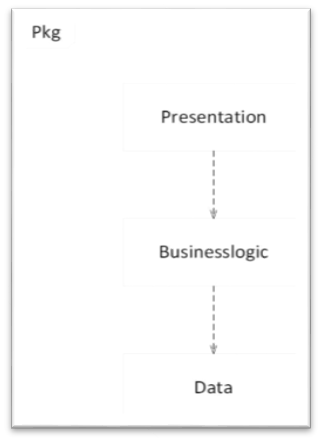


图 1

# 导航设计

## 导航内容

|  |  |
| --- | --- |
| 导航栏 | 侧边栏 |
| 首页 | 疯狂运动、朋友圈、我的排名、我的健康 |
| 竞赛 | 全部竞赛、发布竞赛、我参与的、历史竞赛 |
| 活动 | 活动推荐、全部活动、我参与的 |
| 兴趣组 | 兴趣组推荐、全部兴趣组、我参与的 |
| 个人 | 我的动态、个人资料、我的好友、我的账户、消息盒子 |

## 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21个界面，分别是登陆界面、注册界面、首页（疯狂运动、朋友圈、我的排名、我的健康）、竞赛（全部竞赛、我参与的、发布竞赛、历史竞赛）、活动（全部活动、我参与的、推荐活动）、个人（我的动态、个人资料、我的好友、我的账户、消息盒子）、兴趣组（全部兴趣组、我参与的、推荐兴趣组）

## 人机交互界面转换

人机交互界面转换如下图：

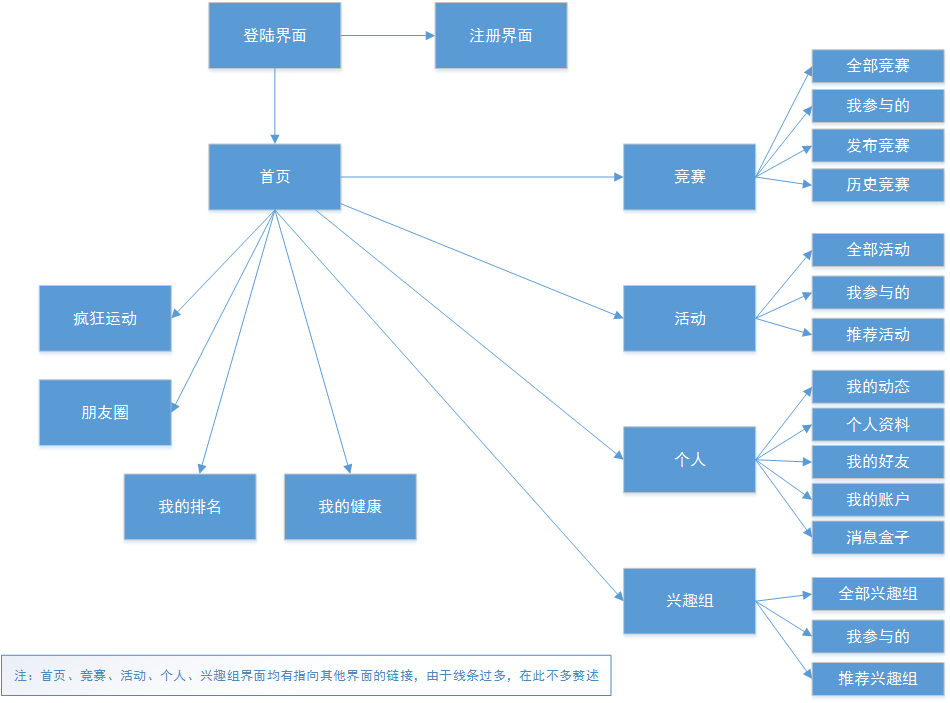


图 7

# 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 |
| race | 负责竞赛界面所需要的服务 |
| activity | 负责活动界面所需要的服务 |
| user | 负责个人信息所需要的服务 |
| interest | 负责兴趣组界面所需要的服务 |

### 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健康专家系统需求文档。

* 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, $account\_id) |
| 前置条件 | 用户id必须存在 |
| 后置条件 | 获得这个用户的账户 |
| user.get\_notify | 语法 | function get\_notify ($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通知消息 |
| 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 ($user\_id) |
| 前置条件 | 用户id必须存在 |
| 后置条件 | 删除这个用户的一个好友 |
| user.update\_account | 语法 | function update\_account($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。

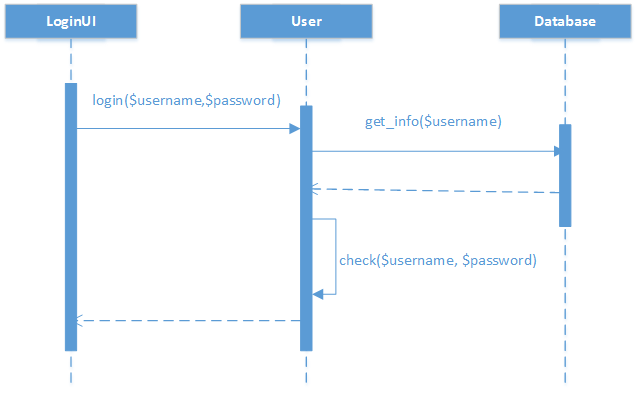


图 8

#### index模块

* 模块概述

提供首页中显示全部动态、获得排名、获得健康信息，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\_news | 语法 | function get\_news() |
| 前置条件 | 无 |
| 后置条件 | 读取用户的动态，显示在朋友圈内 |
| index.get\_ranking | 语法 | function get\_ranking() |
| 前置条件 | 无 |
| 后置条件 | 获取当前用户的排名 |
| index.get\_sports\_time | 语法 | function get\_sports\_time() |
| 前置条件 | 无 |
| 后置条件 | 读取用户的运动时间 |
| index.get\_sports\_distance | 语法 | function get\_sports\_distance() |
| 前置条件 | 无 |
| 后置条件 | 读取用户的运动距离 |
| index.get\_health | 语法 | function get\_health() |
| 前置条件 | 无 |
| 后置条件 | 读取用户的健康数据（身高，体重，心率，血压） |
| index.set\_health | 语法 | function set\_health($health) |
| 前置条件 | 用户输入的健康数据（身高，体重，心率，血压）格式正确 |
| 后置条件 | 插入用户的健康数据（之前的作为历史数据保留） |
| index.add\_news | 语法 | function add\_news() |
| 前置条件 | 用户输入的动态不为空 |
| 后置条件 | 记录用户新增的动态 |

* 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。

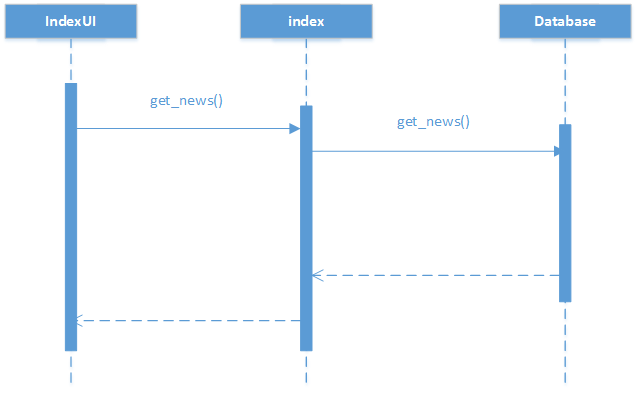


图 9

#### race模块

* 模块概述

race模块提供显示全部竞赛、发起竞赛、删除竞赛，搜索竞赛、历史竞赛、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

* race接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| race.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的竞赛 |
| race.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 |
| race.add | 语法 | function add($race) |
| 前置条件 | 用户成功登陆，用户输入合法 |
| 后置条件 | 记录用户发布的竞赛 |
| race.delete | 语法 | function delete ($race\_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 |
| race.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 |
| race.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。

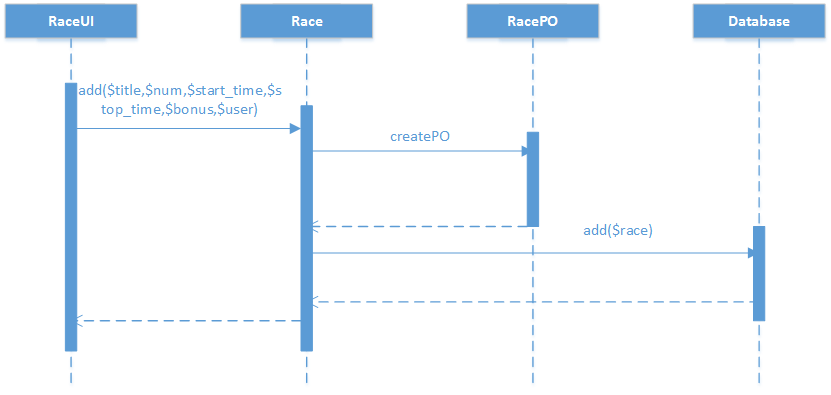


图 10

#### activity模块

* 模块概述

activity模块提供显示全部活动、发起活动、删除活动，搜索活动、历史活动、我的活动功能，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

* activity接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| activity.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的活动 |
| activity.my\_ participate | 语法 | function my\_ participate () |
| 前置条件 | 用户成功登陆 |
| 后置条件 | 获得用户参与的活动 |
| activity.add | 语法 | function add ($activity) |
| 前置条件 | 管理员成功登陆，用户输入合法 |
| 后置条件 | 记录管理员发布的活动 |
| activity.delete | 语法 | function delete ($activity\_id) |
| 前置条件 | 管理员成功登陆 |
| 后置条件 | 删除管理员发起的竞赛 |
| activity.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活动 |
| activity.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查找活动 |

* 业务逻辑层的动态模型

如图表明了健康专家系统中，当添加一次活动的时候，业务逻辑处理的相关对象之间的协作。

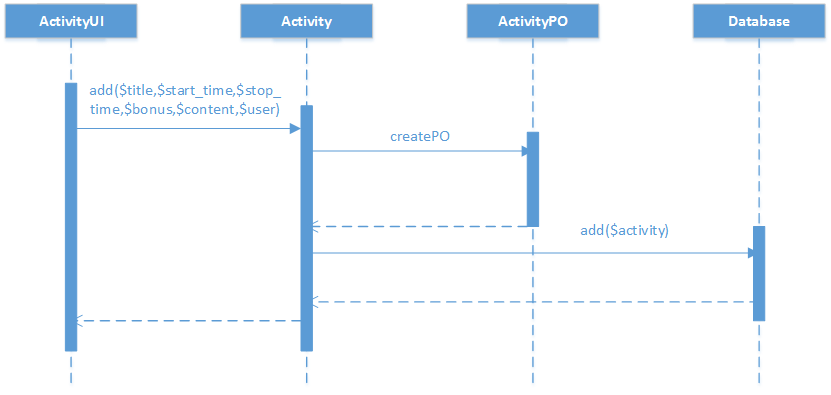


图 11

#### interest模块

* 模块概述

interest模块提供显示全部兴趣组、发起兴趣组、删除兴趣组，搜索兴趣组、我的兴趣组功能，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

* interest接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| interest.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的兴趣组 |
| interest.my\_ participate | 语法 | function my\_ participate () |
| 前置条件 | 用户成功登陆 |
| 后置条件 | 获得用户参与的兴趣组 |
| interest.add | 语法 | function add ($interest) |
| 前置条件 | 用户成功登陆，用户输入合法 |
| 后置条件 | 记录用户发布的兴趣组 |
| interest.delete | 语法 | function delete ($interest \_id) |
| 前置条件 | 要删除的兴趣组是此用户发起的 |
| 后置条件 | 删除用户发起的兴趣组 |
| interest.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查找兴趣组 |
| interest.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 |

* 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。

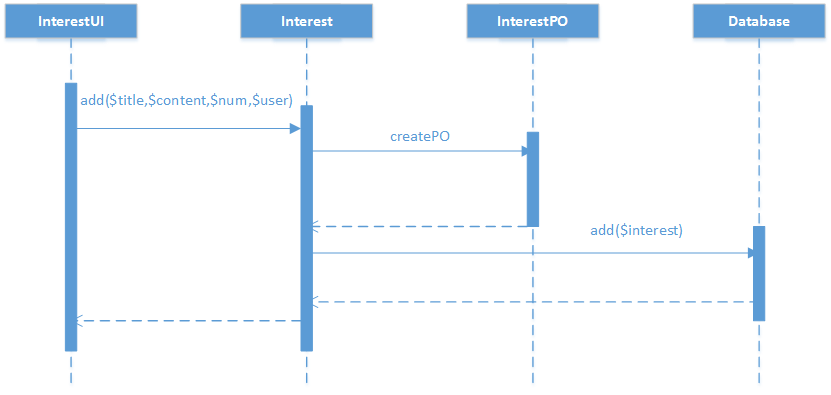


图 12

## 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 |
| interestdata | 负责读取全部兴趣组，提供增删改查功能 |
| activitydata | 负责读取全部活动，提供增删改查功能 |
| racedata | 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 |

#### interestdata模块

* **interestdata接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| interestdata.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的兴趣组 |
| interestdata.find | 语法 | function find($id) |
| 前置条件 | id输入合法 |
| 后置条件 | 根据id查找兴趣组 |
| interestdata.add | 语法 | function add($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 |
| interestdata.delete | 语法 | function delete($id) |
| 前置条件 | 删除的兴趣组为此用户发起的 |
| 后置条件 | 根据id删除兴趣组 |
| interestdata.update | 语法 | function update($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 |

#### activitydata模块

* **activitydata接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| activitydata.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的活动 |
| activitydata.find | 语法 | function find($id) |
| 前置条件 | id输入合法 |
| 后置条件 | 根据id查找活动 |
| activitydata.add | 语法 | function add($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 |
| activitydata.delete | 语法 | function delete($id) |
| 前置条件 | 删除的活动为此用户发起的 |
| 后置条件 | 根据id删除活动 |
| activitydata.update | 语法 | function update($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 |

#### racedata模块

* **racedata接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| racedata.show | 语法 | function show() |
| 前置条件 | 无 |
| 后置条件 | 提供全部的竞赛 |
| racedata.find | 语法 | function find($id) |
| 前置条件 | id输入合法 |
| 后置条件 | 根据id查找竞赛 |
| racedata.add | 语法 | function add($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 |
| racedata.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：

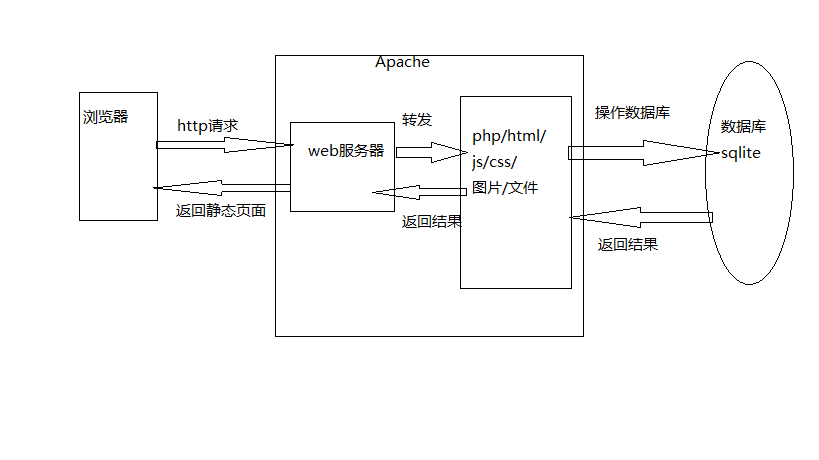


图 13

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

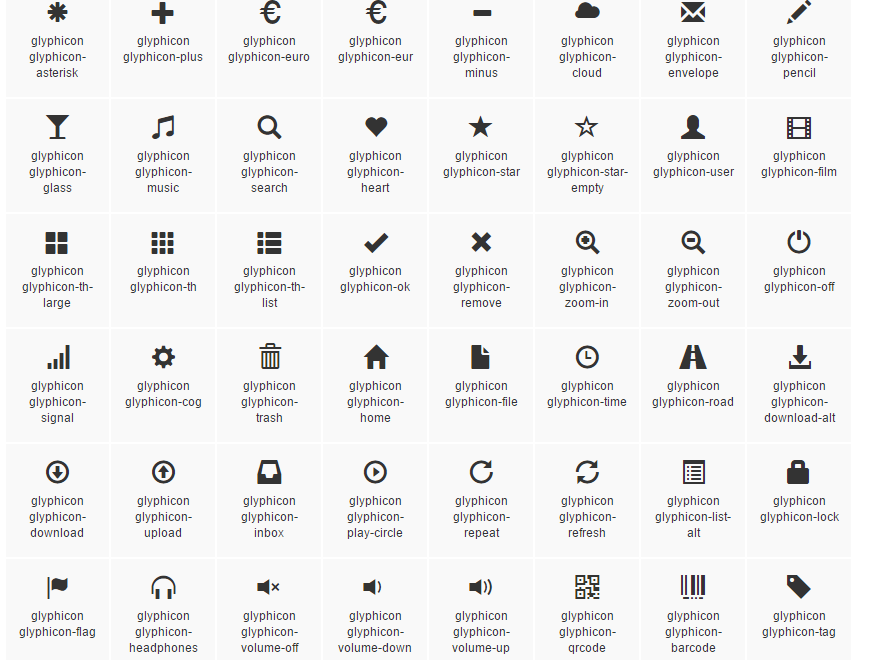


图 14

### 导航栏



图 15

### 按钮



图 16

### 侧边栏

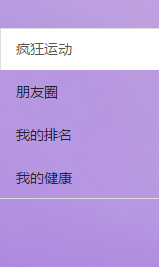


图 17

### 进度条

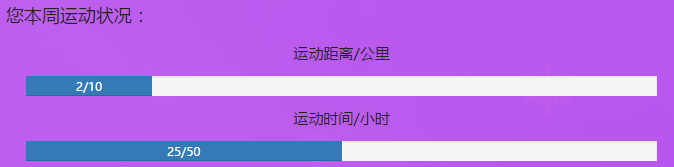


图 18

### 输入框

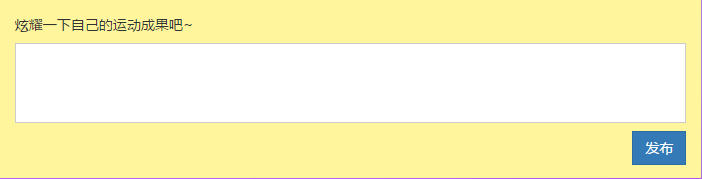


图 19

### 分页



图 20

### 搜索框



图 21