NextStep运动健康管理系统

软件体系结构设计文档 v1.0

南京大学软件学院

陈沐恩

2016.10.29

变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 陈沐恩 | 2016.10.29 | 最初框架 | v0.1 |
| 陈沐恩 | 2016.11.4 | 补全详细设计 | v1.0 |

目录

[1. 引言 6](#_Toc466119765)

[1.1. 编制目的和范围 6](#_Toc466119766)

[1.2. 术语表 6](#_Toc466119767)

[1.3. 参考资料 6](#_Toc466119768)

[2. 产品描述 6](#_Toc466119769)

[3. 总体设计 6](#_Toc466119770)

[3.1. 运行环境 6](#_Toc466119771)

[3.2. 设计模式 6](#_Toc466119772)

[3.3. 软件架构 7](#_Toc466119773)

[3.4. 模块结构 8](#_Toc466119774)

[4. 数据库设计 8](#_Toc466119775)

[4.1. 关系模型 8](#_Toc466119776)

[5. 模块/接口设计 9](#_Toc466119777)

[5.1. 登录注册 9](#_Toc466119778)

[5.1.1. 提供的接口 9](#_Toc466119779)

[5.1.2. 依赖的接口 9](#_Toc466119780)

[5.2. 用户管理 10](#_Toc466119781)

[5.2.1. 提供的接口 10](#_Toc466119782)

[5.2.2. 依赖的接口 11](#_Toc466119783)

[5.3. 个人设备管理 11](#_Toc466119784)

[5.3.1. 提供的接口 11](#_Toc466119785)

[5.3.2. 依赖的接口 12](#_Toc466119786)

[5.4. 查看健康数据 12](#_Toc466119787)

[5.4.1. 提供的接口 13](#_Toc466119788)

[5.4.2. 依赖的接口 14](#_Toc466119789)

[5.5. 查看健康统计数据 15](#_Toc466119790)

[5.5.1. 提供的接口 15](#_Toc466119791)

[5.5.2. 依赖的接口 16](#_Toc466119792)

[5.6. 关注管理 17](#_Toc466119793)

[5.6.1. 提供的接口 17](#_Toc466119794)

[5.6.2. 依赖的接口 18](#_Toc466119795)

[5.7. 活动管理 19](#_Toc466119796)

[5.7.1. 提供的接口 19](#_Toc466119797)

[5.7.2. 依赖的接口 23](#_Toc466119798)

[5.8. 圈子管理 23](#_Toc466119799)

[5.8.1. 提供的接口 23](#_Toc466119800)

[5.8.2. 依赖的接口 26](#_Toc466119801)

[5.9. 权限管理 27](#_Toc466119802)

[5.9.1. 提供的接口 27](#_Toc466119803)

[5.9.2. 依赖的接口 27](#_Toc466119804)

[6. 设备数据格式定义 27](#_Toc466119805)

[6.1. 步数数据 28](#_Toc466119806)

[6.2. 睡眠数据 28](#_Toc466119807)

[6.3. 心率数据 28](#_Toc466119808)

[6.4. 体重数据 29](#_Toc466119809)

[7. 交互设计 29](#_Toc466119810)

[8. 安全保密设计 29](#_Toc466119811)

[9. 性能优化 29](#_Toc466119812)

[9.1. 尽量减少HTTP请求 29](#_Toc466119813)

[9.2. 使用浏览器缓存 29](#_Toc466119814)

[9.3. 图片、JS文件等预载入 29](#_Toc466119815)

[9.4. 合理安排脚本文件、样式文件的位置 29](#_Toc466119816)

[9.5. JS、CSS文件均使用外联方式 30](#_Toc466119817)

[9.6. 精简各类文件 30](#_Toc466119818)

[9.7. 其他 30](#_Toc466119819)

1. 引言
   1. 编制目的和范围

本报告详细完成对NextStep运动健康管理系统的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

* 1. 术语表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| NextStep | 该系统名称 |  |
| Laraval | 一种PHP开发框架 |  |
| RESTful | 一种软件架构风格 |  |
| BMI | 体质指数 |  |

* 1. 参考资料

1. IEEE标准
2. NextStep运动健康管理系统需求文档v1.1
3. 产品描述

参考NextStep软件需求规格说明文档中对产品的概括描述。

1. 总体设计
   1. 运行环境

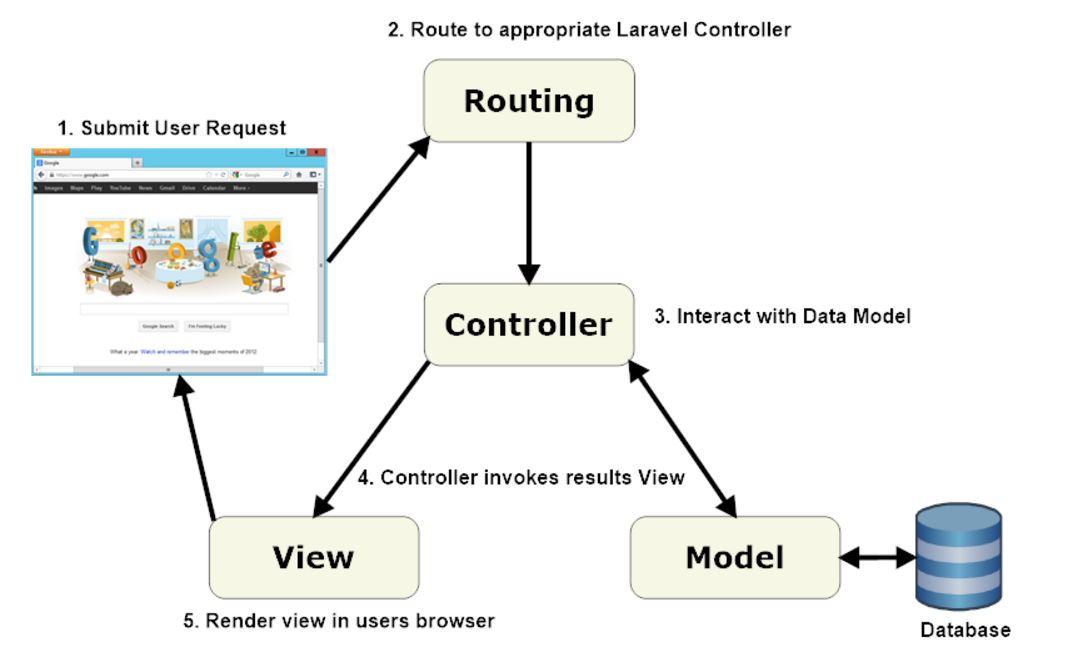
|  |  |
| --- | --- |
| 服务端 | |
|  |  |
| WEB服务器 | Apache |
| 服务端语言 | PHP5.5.7 |
| 数据库 | SQLite 3 |
| 网络协议 | TCP/IP协议 |
| 客户端 | |
| 操作系统 | Windows, macOS, Linux |
| 网络协议 | TCP/IP协议 |
| 浏览器 | Internet Explorer 8.0以上版本 |

* 1. 设计模式

因为NextStep系统服务端使用的是Lavaral框架，所以基本设计模式为MVC设计模式。

当与服务器交互时，浏览器发送一个请求，web服务器接收到请求并且传给Laravel路由引擎。Laravel路由接收到请求，然后重定向给基于路由的URL模式的合适的控制器类方法。

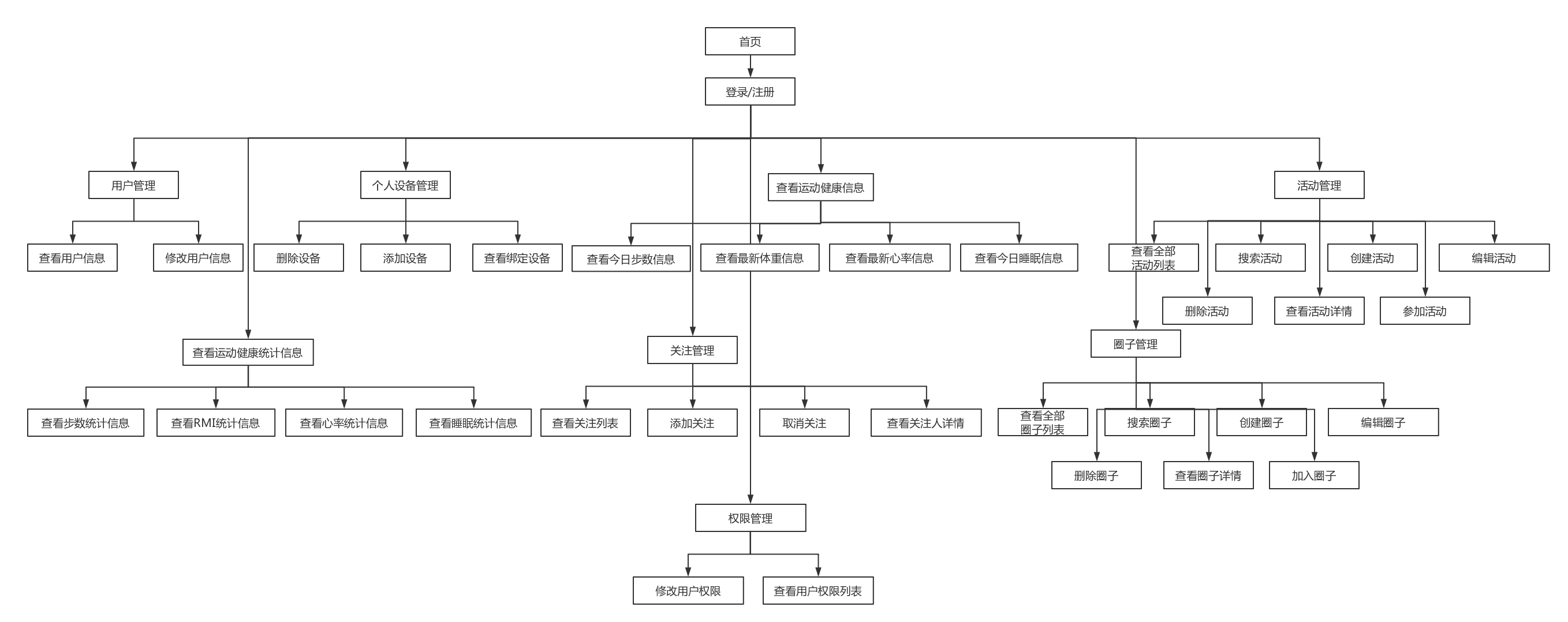
然后控制器类接管。在某种情况下，控制器会立即呈现出一个视图，它是一个被转换成HTML并送回浏览器的模版。调用模型后，控制器则呈现最终视图（ HTML，CSS和图像），并返回完整的网页到用户的浏览器。



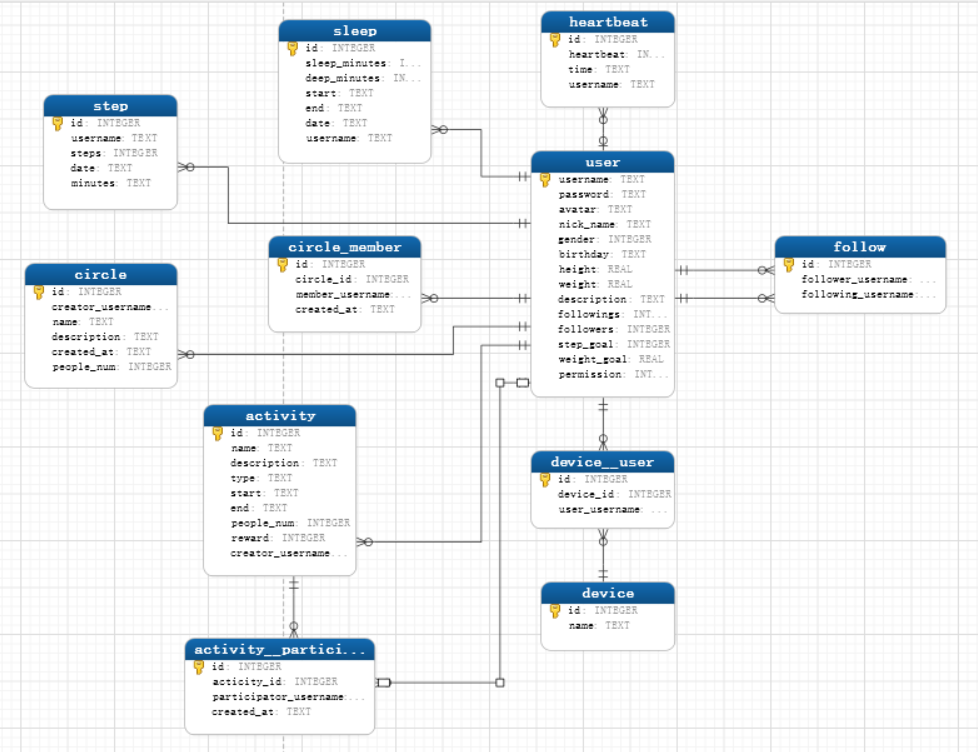
* 1. 软件架构

本系统采用RESTful软件架构，具体体现为：

1. 每一个URI代表一种资源；
2. 客户端和服务器之间，传递这种资源的某种表现层；
3. 客户端通过四个HTTP动词，对服务器端资源进行操作，实现"表现层状态转化"。
   1. 模块结构



1. 数据库设计
   1. 关系模型



1. 模块/接口设计
   1. 登录注册
      1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 登录 | URL | /login |
| 获取类型 | GET |
| 参数 | username – 用户名  password – 密码 |
| 依赖控制器 | UserController |
| 方法 | login(username, password) |
| 前置条件 | 用户已注册，并正确输入用户名、密码 |
| 后置条件 | 跳转至用户首页 |
| 注册 | URL | /register |
| 获取类型 | POST |
| 参数 | username – 用户名  password – 密码 |
| 依赖控制器 | UserController |
| 方法 | register(username, password) |
| 前置条件 | 无 |
| 后置条件 | 新增账号，跳转至登录界面 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 服务 | 说明 |
| User::login | 登录 |
| User::register | 注册 |

* 1. 用户管理
     1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 查看个人信息 | URL | /user/info |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | UserController |
| 方法 | getUserInfo (username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 更改个人信息 | URL | /user/info |
| 获取类型 | POST |
| 参数 | username – 昵称  age – 年龄  height – 升高  weight – 体重  steps\_goal – 每日目标步数  weight\_goal – 体重目标 |
| 依赖控制器 | UserController |
| 方法 | updateUserInfo(username) |
| 前置条件 | 用户已注册，并正确输入用户名、密码 |
| 后置条件 | 无 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| User::getInfo | 获取用户信息 |
| User::updateInfo | 更新用户信息 |

* 1. 个人设备管理
     1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 查看当前设备 | URL | /device |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | DeviceController |
| 方法 | getDevice(username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 添加设备 | URL | /device |
| 获取类型 | POST |
| 参数 | device\_id – 设备识别号 |
| 依赖控制器 | DeviceController |
| 方法 | addDevice(username, device\_id) |
| 前置条件 | 无 |
| 后置条件 | 跳转至查看设备界面 |
| 删除个人设备 | URL | /device |
| 获取类型 | Delete |
| 参数 | device\_id – 设备识别号 |
| 依赖控制器 | DeviceController |
| 方法 | deleteDevice(username, device\_id) |
| 前置条件 | 用户已登录，并绑定该设备 |
| 后置条件 | 跳转至查看设备界面 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| Device::getUserDevice | 获取用户绑定设备信息 |
| Device::addUserDevice | 新增用户绑定设备 |
| Device::deleteUserDevice | 删除用户绑定设备 |

* 1. 查看健康数据
     1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 查看今日步行数据 | URL | /health\_data/today/steps |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | HealthController |
| 方法 | getTodayStep(username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看今日睡眠数据 | URL | /health\_data/today/sleep |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | HealthController |
| 方法 | getTodaySleep(username, device\_id) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看最近心跳 | URL | /health\_data/heart |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | HealthController |
| 方法 | getLastestHeart(username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看最近体重 | URL | /health\_data/weight |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | HealthController |
| 方法 | getLastestWeight(username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看今日步数排名 | URL | /health\_data/today/step\_rank |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | HealthController |
| 方法 | getTodayStepRank(username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| Health::getStepsData | 获取步数数据 |
| Health::getSleepData | 获取睡眠数据 |
| Health::getHeartbeatsData | 获取心跳数据 |
| Health::getWeightData | 获取体重数据 |
| User::getUserInfo | 获取用户数据 |

* 1. 查看健康统计数据
     1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 查看步行统计数据 | URL | /statistics/step |
| 获取类型 | GET |
| 参数 | time\_type[‘week’, ‘month’, ‘year’] – 统计时间单位  time – 统计时间 |
| 依赖控制器 | StatisticsController |
| 方法 | getSleepStatistics(username, timeType, time) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看睡眠统计数据 | URL | /statistics/sleep |
| 获取类型 | GET |
| 参数 | time\_type[‘week’, ‘month’, ‘year’] – 统计时间单位  time – 统计时间 |
| 依赖控制器 | StatisticsController |
| 方法 | getSleepStatistics(username, timeType, time) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看心率统计数据 | URL | /statistics/heart |
| 获取类型 | GET |
| 参数 | time – 统计时间 |
| 依赖控制器 | StatisticsController |
| 方法 | getHeartStatistics(username, time) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看指标统计数据 | URL | /statistics/BMI |
| 获取类型 | GET |
| 参数 | time – 统计时间 |
| 依赖控制器 | StatisticsController |
| 方法 | getBMIStatistics(username, time) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| Health::getStepsData | 获取步数数据 |
| Health::getSleepData | 获取睡眠数据 |
| Health::getHeartbeatsData | 获取心跳数据 |
| Health::getWeightData | 获取体重数据 |
| User::getUserInfo | 获取用户数据 |
| Statistics::getStepsStat | 获取步数统计数据 |
| Statistics::getSleepStat | 获取睡眠统计数据 |
| Statistics::getHeartbeatsStat | 获取心跳统计数据 |
| Statistics::getBMIStat | 获取BMI统计数据 |

* 1. 关注管理
     1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 查看好友列表 | URL | /following\_list |
| 获取类型 | GET |
| 参数 | page – 页数 |
| 依赖控制器 | FollowController |
| 方法 | getFollowingList(username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 添加好友 | URL | /follow |
| 获取类型 | POST |
| 参数 | following\_id – 关注人的用户名 |
| 依赖控制器 | FollowingController |
| 方法 | addFollow (username, following\_id) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 删除好友 | URL | /follow |
| 获取类型 | DETELE |
| 参数 | following\_id – 关注人的用户名 |
| 依赖控制器 | FollowController |
| 方法 | deleteFollow (username, following\_id) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看好友详情 | URL | /follow |
| 获取类型 | GET |
| 参数 | following\_id – 关注人的用户名 |
| 依赖控制器 | FollowController |
| 方法 | deleteFollow (username, following\_id) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| Follow::getMyFollow | 获取当前关注用户 |
| Follow::addFollow | 关注用户 |
| Follow::deleteFollow | 取消关注 |
| User::getInfo | 获取用户详细信息 |

* 1. 活动管理
     1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 查看所有活动列表 | URL | /activity\_list/all |
| 获取类型 | GET |
| 参数 | sort\_type – 排序方式  page – 页数 |
| 依赖控制器 | ActivityController |
| 方法 | getAllActivity(sort\_type, page) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 搜索活动 | URL | /activity\_list/search |
| 获取类型 | GET |
| 参数 | keyword – 关键词  sort\_type – 排序方式  page – 页数 |
| 依赖控制器 | ActivityController |
| 方法 | searchActivity(keyword, sort\_type, page) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看活动详情 | URL | /activity |
| 获取类型 | GET |
| 参数 | id – 活动ID |
| 依赖控制器 | ActivityController |
| 方法 | getActivityDetail(activity\_id) |
| 前置条件 | 用户已登录，该活动已被创建 |
| 后置条件 | 无 |
| 查看用户创建活动 | URL | /activity\_list/created |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | ActivityController |
| 方法 | getCreatedActivity(username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 跳转至我创建的活动界面 |
| 查看用户参与活动 | URL | /activity\_list/joined |
| 获取类型 | GET |
| 参数 | 无 |
| 依赖控制器 | ActivityController |
| 方法 | getJoinedActivity(username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 跳转至我创建的活动界面 |
| 创建活动 | URL | /activity |
| 获取类型 | POST |
| 参数 | name – 活动名  start – 活动开始时间  end – 活动结束时间  people\_num[0表示无限制] – 人数限制  type[goal, competition] – 活动类型  reward – 奖励  description – 活动描述 |
| 依赖控制器 | ActivityController |
| 方法 | addActivity(name, start, end, people\_num, type, reward, description) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 编辑活动 | URL | /activity |
| 获取类型 | UPDATE |
| 参数 | id – 活动ID  name – 活动名  start – 活动开始时间  end – 活动结束时间  people\_num[0表示无限制] – 人数限制  type[goal, competition] – 活动类型  reward – 奖励  description – 活动描述 |
| 依赖控制器 | ActivityController |
| 方法 | updateActivity(id, name, start, end, people\_num, type, reward, description) |
| 前置条件 | 用户已登录，且为该活动的创建者（或为管理员） |
| 后置条件 | 无 |
| 删除活动 | URL | /activity |
| 获取类型 | DELETE |
| 参数 | id – 活动ID |
| 依赖控制器 | ActivityController |
| 方法 | deleteActivity(id) |
| 前置条件 | 用户已登录，且为该活动的创建者（或为管理员） |
| 后置条件 | 无 |
| 参与活动 | URL | /activity/join |
| 获取类型 | POST |
| 参数 | id – 活动ID |
| 依赖控制器 | ActivityController |
| 方法 | joinActivity(id, username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| Activity::getActList | 获取活动列表 |
| Activity::getJoinedActList | 获取参与的活动列表 |
| Activity::getCreatedActList | 获取创建的活动列表 |
| Activity::getInfo | 获取活动详情 |
| Activity::create | 创建活动 |
| Activity::update | 更新活动 |
| Activity::delete | 删除活动 |
| Activity::join | 参加活动 |

* 1. 圈子管理
     1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 查看圈子列表 | URL | /circle\_list/all |
| 获取类型 | GET |
| 参数 | sort\_type – 排序方式  page – 页数 |
| 依赖控制器 | CircleController |
| 方法 | getAllCircle(sort\_type, page) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 搜索圈子 | URL | /circle\_list/search |
| 获取类型 | GET |
| 参数 | keyword – 关键词  sort\_type – 排序方式  page – 页数 |
| 依赖控制器 | CircleController |
| 方法 | searchCircle(keyword, sort\_type, page) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 查看圈子详情 | URL | /circle |
| 获取类型 | GET |
| 参数 | id – 圈子ID |
| 依赖控制器 | CircleController |
| 方法 | getCircleDetail(circle\_id) |
| 前置条件 | 用户已登录，该圈子已被创建 |
| 后置条件 | 无 |
| 创建圈子 | URL | /circle |
| 获取类型 | POST |
| 参数 | name – 圈子名  people\_num[0表示无限制] – 人数限制  description – 圈子描述 |
| 依赖控制器 | CircleController |
| 方法 | addCircle(name, people\_num, type, description) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |
| 编辑圈子 | URL | /circle |
| 获取类型 | UPDATE |
| 参数 | id – 圈子ID  name – 圈子名  start – 圈子开始时间  end – 圈子结束时间  people\_num[0表示无限制] – 人数限制  type[goal, competition] – 圈子类型  reward – 奖励  description – 圈子描述 |
| 依赖控制器 | CircleController |
| 方法 | updateCircle() |
| 前置条件 | 用户已登录，且为该圈子的创建者 |
| 后置条件 | 无 |
| 删除圈子 | URL | /circle |
| 获取类型 | DELETE |
| 参数 | id – 圈子ID |
| 依赖控制器 | CircleController |
| 方法 | deleteCircle(id) |
| 前置条件 | 用户已登录，且为该圈子的创建者 |
| 后置条件 | 无 |
| 加入圈子 | URL | /circle/join |
| 获取类型 | POST |
| 参数 | id – 圈子ID |
| 依赖控制器 | CircleController |
| 方法 | joinCircle(id, username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 无 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| Circle::getCircleList | 获取圈子列表 |
| Circle::getJoinedCircleList | 获取加入的圈子列表 |
| Circle::getCreatedCircleList | 获取创建的圈子列表 |
| Circle::getInfo | 获取圈子详情 |
| Circle::create | 创建圈子 |
| Circle::update | 更新圈子 |
| Circle::delete | 删除圈子 |
| Circle::join | 加入圈子 |

* 1. 权限管理
     1. 提供的接口

|  |  |  |
| --- | --- | --- |
| 修改用户权限 | URL | /permission |
| 获取类型 | UPDATE |
| 参数 | username – 修改的用户名  level – 更新的等级 |
| 依赖控制器 | PermissionController |
| 方法 | updatePermission(username, level) |
| 前置条件 | 用户已登录，且为管理员 |
| 后置条件 | 无 |

* + 1. 依赖的接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| Permission::update | 修改用户权限 |

1. 设备数据格式定义
   1. 步数数据

{

"steps": 0, // 步数

"date": 2016-11-5 // 日期

"calories": 0, // 消耗卡路里

"meters": 0, // 运动距离

"minutes": 0 // 运动时长

}

* 1. 睡眠数据

{

"total\_minutes": 80, //总睡眠时间

"date": "2016-11-05", //日期

"start": "2016-11-04 23:29:12", //入睡时间

"end": "2016-11-05 08:29:12", //醒来时间

"deep\_minutes":60, //深睡时间

"light\_minutes": 20, //浅睡时间

}

* 1. 心率数据

{

"heartbeat\_rate": 90 // 心率（次/分）

"time": "2016-11-5 12:23:31" // 测量时间

"status": "normal"["normal", "fast", "slow"] // 心跳状况

}

* 1. 体重数据

{

"weight": 70 // 体重（kg）

"time": "2016-12-12 21:34:21" // 测量时间

"status": "normal" //体重状况

}

1. 交互设计

参考NextStep需求规格文档v1.1 用户界面部分

1. 安全保密设计
2. 用户通过账号密码登录系统
3. 用户密码使用MD5加密后存于数据库
4. 设计权限表，用户根据权限不同享有不同级别的功能
5. 性能优化
   1. 尽量减少HTTP请求

合并脚本和样式表，CSS Sprites，拆分初始化负载

* 1. 使用浏览器缓存

尽量强制浏览器缓存文件，并设置过期时间

* 1. 图片、JS文件等预载入
  2. 合理安排脚本文件、样式文件的位置
* 脚本文件置于底部
* 样式文件置于顶部
  1. JS、CSS文件均使用外联方式

使用外部 JavaScript 和 CSS

* 1. 精简各类文件
* 移除重复脚本
* 复用样式表
  1. 其他
* 使用 CDN
* 添加Expires 头减少 DNS 查找
* 配置 ETag，使AJAX可缓存