健康之友app架构设计说明书

**华中科技大学软件学院**

|  |  |  |
| --- | --- | --- |
| 李浩翔 | U201217485 | 软工1205 |
| 石成山 | U201217493 | 软工1205 |
| 李歆 | U201217488 | 软工1205 |
| 杨逸雷 | U201217445 | 软工1203 |

目录

[一.文档简介 4](#_Toc417757528)

[二.系统架构设计 4](#_Toc417757529)

[三.架构设计目标与约束 5](#_Toc417757530)

[1. 概述 5](#_Toc417757531)

[2.关键功能需求 5](#_Toc417757532)

[3.质量场景 6](#_Toc417757533)

[四.用例视图 8](#_Toc417757534)

[1.概述 8](#_Toc417757535)

[2.关键用例 8](#_Toc417757536)

[五.逻辑视图 9](#_Toc417757537)

[1.概述 9](#_Toc417757538)

[2.类图 9](#_Toc417757539)

[六.部署视图 9](#_Toc417757540)

[1.概述 9](#_Toc417757541)

[2.部署模型视图 10](#_Toc417757542)

# 文档简介

编写目的：

本文档全面与系统地表述了健康软件系统的构架，并通过使用多种视图来从不同角度描述本系统的各个主要方面，以满足健康软件系统的相关涉众（客户、设计人员等）对本系统不同的关注焦点和需求。本文档记录并表述了系统架构的设计人员对系统构架方面做出的重要决策。

本文档的预期阅读人员为程序设计人员、测试人员和其他有关的工作人员。

# 系统架构设计

整个软件架构方案采用分层结构，明确地分离了表现层和业务逻辑，能够保证应用服务逻辑的一致性和稳定性、结构的开放性、功能的可扩展性和可维护性、开发的可并行性，同时采用一些开源的框架，兼顾了经济性。框架是一种特殊的软件，它为软件开发带来了高度的重用性，是无数软件开发人员的多年项目开发经验的总结。在一个优秀的框架上开发应用，而不是从零开始，可以大量缩短项目的开发周期、降低开发风险、增强应用系统的稳定性。

数据访问层 Dao hibernate

表示层 Android sdk

业务层 service spring

数据库MySQL

Android手机

从架构图中可以看出系统分为三层：

·表示层：借助Android sdk实现

·业务层：借助Spring 进行业务组件的组装关联。

·数据持久层：借助Hibernate实现

通过成熟的开源产品实现各层，同自己编写代码实现，相比之下能缩短开发周期，且架构所用到的开源产品均有很广泛的用户群，经受过实践的考验，质量和性能更有保障。

·层与层之间松散耦合，增加代码重用率。

·各层分工明确，这样也利于团队的明确分工。

系统的总体架构从结构上分为表示层、业务层、数据访问层以及在层间传递数据的数据传输对象。下面针对各层加以描述。

1) 表示层

表示层与app客户端对应，采用浏览器与用户交互，把来自系统的信息显示给用户，主要控制页面外观，产生页面逻辑以及对用户输入的数据进行合法性验证。

2) 业务层

业务层处理应用的核心业务逻辑。业务逻辑对象把业务规则、约束、活动和数据结合在一起，Spring 负责对这些业务对象的管理。

3) 数据访问层

数据访问对象(Data Access Object)把底层的数据访问操作和上层的商务逻辑分开。

4) 数据传输对象（Data Transfer Object）

数据传输对象通常作为各业务实体的JAVABEAN对象，负责层与层之间数据的传输。

# 架构设计目标与约束

## 概述

描述构架设计最主要目标就是满足关键系统功能需求和质量约束，这些功能需求和质量要求对软件构架有重大的影响，并决定了构架的设计。本节同时还列明影响构架的其他相关因素，如软件的复用策略、使用商业构件、设计与实施的策略等。

## 关键功能需求

按照需求分析文档的规格要求，本学分制下选课系统的设计分成了选课，退选和查询，查看已修课程模块为主的三大模块进行开发。系统的开发主要是为了使选课的管理工作更加规范化、系统化和程序化，提高信息处理的速度和准确性，提高读者对求知的用户体验。其功能模块如下所示：

健康管理：随时记录新的身高、体重等信息，并保存历史信息

营养管理：记录摄入能量和消耗的能量，并生成饮食建议

运动管理：运动时做出相应提醒，记录本次运动信息；查看以往运动信息

社区管理：供用户发帖交流，并可发布健身活动信息

推广功能：盈利手段，推广App，推送健身器材、营养产品等广告

个人中心：显示用户个人信息；注册；登陆；Vip增值服务



系统组件图

## 质量场景

|  |  |  |
| --- | --- | --- |
| 目标 | 实现方法 | 采用战术 |
| 性能 | 系统能对用户的请求在规定的时间内做出正确的响应 | 限制访问队列的大小  抓取策略  二级缓存 |
| 安全性 | 可授权检查的安全机制，数据的正确性，可靠的封装机制 | 身份验证，授权验证，验证码验证，数据加密 |
| 可用性 | 当系统遇到错误操作时进行报错 | 错误检测 |
| 模块性 | 根据功能进行模块划分 | 模块划分 |
| 可维护性 | 选课系统在规定时间内完成测试和部署的变更 | 布局优化，防止连锁反应 |
| 可测试性 | 在系统发生变更后可以对系统进行测试 | 输入、输出测试 |

1）性能场景：保证用户的要求能在5s内被响应

**制品：**

系统

**刺激：**提交请求

**响应：**请求被正确处理

**环境：**在正常操 作下

**响应度量：**平均等待时间在5秒内

源：用户

2）安全性场景：只允许用户查看、变更个人的基本信息，系统管理员不能随意查看用户信息，杜绝非法访问、修改数据库的行为，屏蔽无意义的访问，在并发用户多的情况下，系统能保证数据的完整性。

**制品：**

系统

**刺激：**试图非法操作信息

**响应：**系统维持审核踪迹

**环境：**在正常操 作下

**响应度量：**半天内恢复校正数据

源：

用户

# 用例视图

## 概述

用例视图从用户使用的角度描述系统构架的基本外部行为特性，通常包含业务用例模型与系统用例模型。业务用例模型不适用于本系统，这里只关注系统用例。这里选取了用例模型中对系统构架的内容产生重大影响的应用场景与用例集合，这些用例代表了系统主要的核心功能，决定了系统构架的基本组成元素。有些用例强调或决定了构架的某些具体然而重要的细节，通常也可以列在本节内，总之所列的用例集合应基本覆盖系统构架的主要方面。

## 关键用例

* 关键的系统参与者

Health Info

Diet Info

Exercise Info

User Info

Activity Info

# 逻辑视图

## 概述

逻辑视图（Logical View，又叫设计视图（Design View））是为了便于理解系统设计的结构与组织，在“分析设计”工作流程中使用了名为逻辑视图的构架视图。系统只有一个逻辑视图，该视图以图形方式说明关键的用例实现、子系统、包和类，它们包含了在构架方面具有重要意义的行为。逻辑视图在每次迭代过程中都会加以改进。

## 类图



# 部署视图

## 概述

部署视图是为了便于理解系统在一组处理节点上的物理分布，在“分析设计”工作流程中使用了名为部署视图的构架视图。系统只有一个部署视图，它以图形方式说明了处理活动在系统中各节点的分布，包括进程和线程的物理分布。部署视图在每次迭代过程中都会加以改进。

## 部署模型视图

DATABASE

DAO

管理员

Servlet