<运动健身app>

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <30/6/2022> | <1.0> | <第一版> | <宋雨骅> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 技术视图 5

8. 数据视图（可选） 5

9. 核心算法设计（可选） 5

10. 质量属性的设计 5

软件架构文档

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

## 参考资料

[1] 沈备军, 陈昊鹏, 陈雨亭. 软件工程原理[M]. 高等教育出版社, 2013.

[2] 张宏升. 软件架构的非功能性需求指标和区域化支持[J]. 电脑知识与技术，2011，7（09）：2085.

[3] 侯霞,王杨,张磊.一种支持公式的在线文档编辑器的设计[J].电脑知识与技术,2016,12(13):1807.

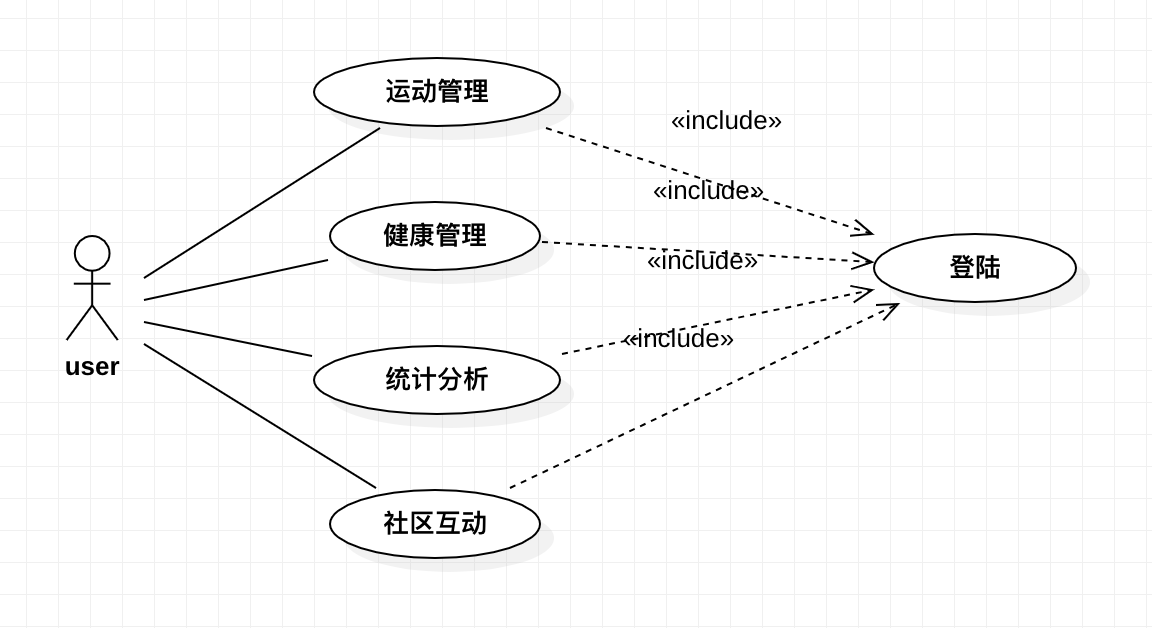
[4] 王文,陈志伟,王树锋.基于中间件的文档在线编辑管理技术的实现[J].中国制造业信息 化,2012,41(23):89.

[5] 朱正扬. 基于Web的协同编辑中资源共享子系统的设计与实现[D].北京邮电大学,2016.

[6] 陆远蓉. 基于 UML 的“4+1”视图软件体系结构描述研究[J]. 现代计算机（专业版），2011

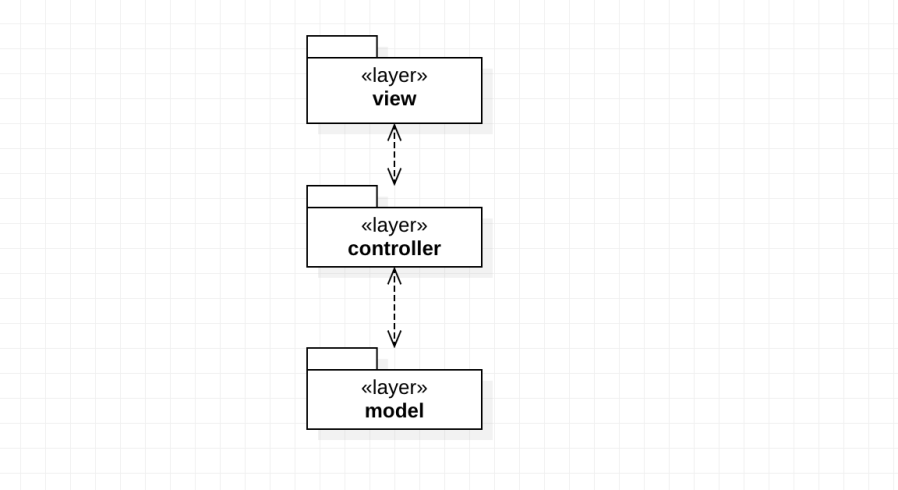
（04）：27-30.

# 用例视图



# 逻辑视图

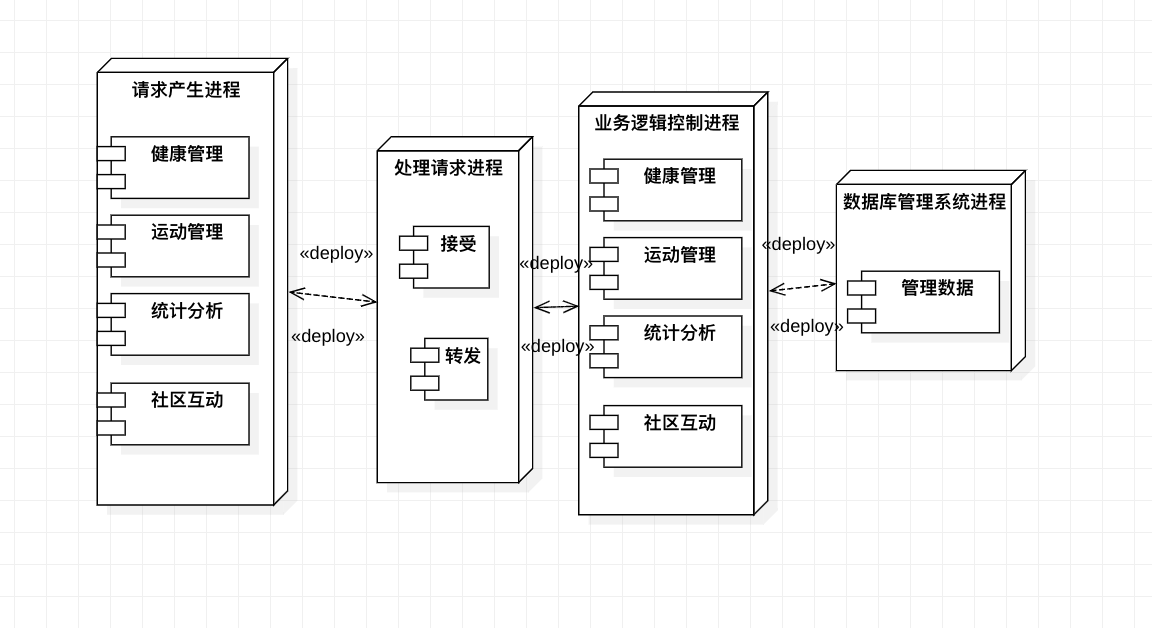
## 概述



采用层次架构（layered architecture）设计。应用层提供了用例图中的各项功能。

## 在构架方面具有重要意义的设计包

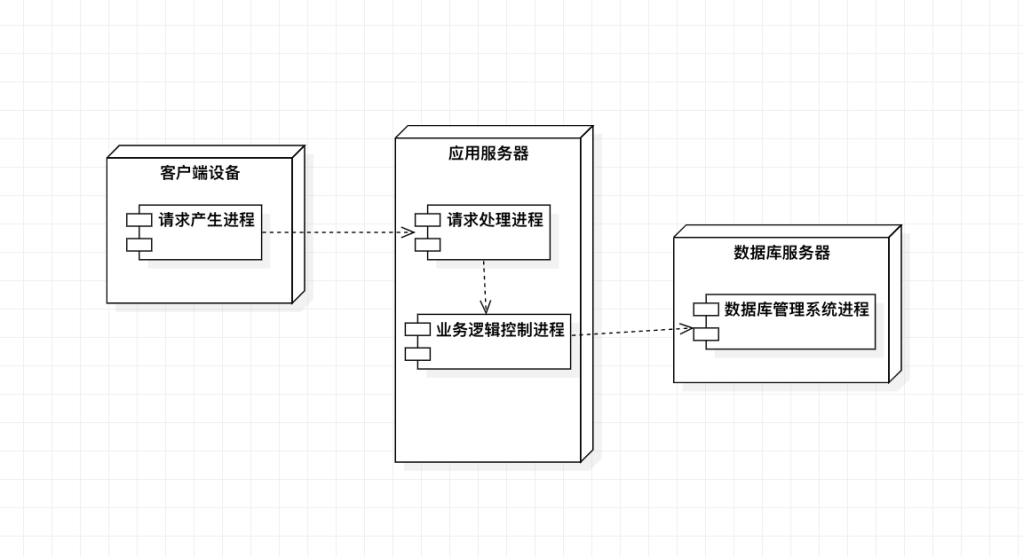
# 进程视图



app端进程与服务器通过异步 I/O 通信，请求处理进程和业务逻辑处理进程通信在服务器端内核中

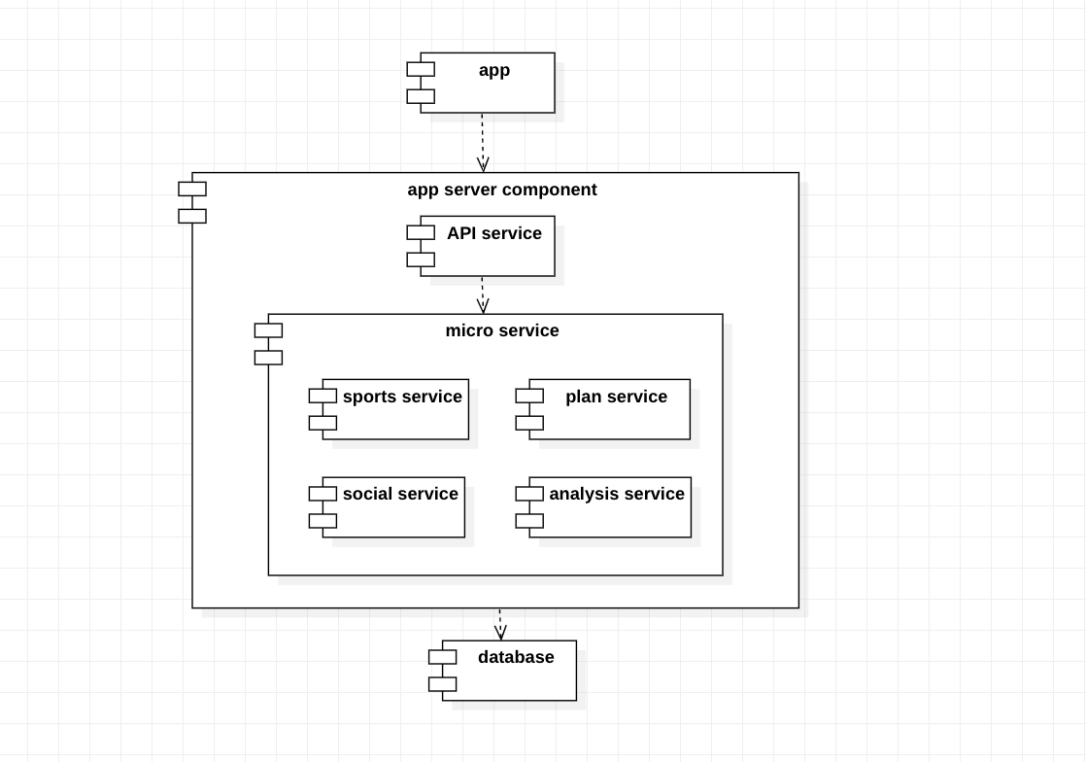
完成，与数据库管理系统进程通过阻塞 I/O 通信。

# 部署视图



多名安卓系统用户在设备上通过app访问服务器，app server 部署在一台服务器上，由于并发用户设置有上线，一台服务器可以在控制成本同时实现需求，app server 连接着数据库服务器，数据库 sport DB 部署在数据库服务器上。

# 实现视图



# 技术视图

**7.1 操作系统：安卓系统**

**7.2 数据库：SQLite**

**7.3 技术栈**

7.3.1 编程语言：Java JavaScript Java

7.3.3 开发工具：Android Studio等

# 质量属性的设计

**8.1 安全性**

8.1.1 原因

本系统的控制台功能只能为有权限的已登录用户提供服务，如果系统安全性不高，易被外界破解， 从事一些非法操作，会对运动会的秩序产生影响。

8.1.2 实现

访问本系统的控制台前，用户必须先进行登录，服务器会验证用户的账号、密码是否在数据库中，

若数据库中没有与之相匹配的信息，则用户将登录失败。同时，未登录用户强行访问控制台时，系

统将发出警告。

**8.2 可修改性**

8.2.1 原因

在系统开发过程中或在最后交付用户使用时，开发人员可能会根据开发进度和用户需求对系统进行

修改；此外，在系统上线后的使用过程中，开发人员也可以增加、删除、修改系统功能使之随着市

场的改变更具有可用性。

8.2.2 实现

本系统前后端的代码架构分层清晰、不通模块之间相互独立，模块内部聚合度高，模块之间耦合度

低，需要对某一功能做出修改时，只需要修改某一部分代码，且不会影响到剩余部分代码。

**8.3 易用性**

8.3.1 原因

易用性关注于如何让用户能够轻易地操作本系统。通过提升易用性，用户可以快速、高效、精准地

使用本系统的功能，同时，减少用户进行系统不支持的错误操作的可能性。

8.3.2 实现

本系统采用 B/S（Browser/ Server）方式实现，并且实现 MVC 分离，将用户操作接口与其余部分离开、耦合度低，单独维护用户接口代码把变更局部化在某个地方。同时，本系统采用中英文设计UI 交互界面，便于各类用户理解。

**8.4 性能**

8.4.1 原因

本系统必须保证其性能，才能为用户提供正常、可靠的服务。如果服务器无法及时对用户操作产生

正确的反馈，这将会导致用户使用本系统体验差，严重影响系统的质量与外部评价。本系统应支持

5名并发用户同时使用系统，系统的响应时间不超过 3 秒。

8.4.2 实现

本系统试图通过优化算法，降低了单一事件处理时的资源开销，提高了计算效率。

# 结构设计

**9.1软件应用系统在纵向结构方面进行分层和隔离**

9.1.1纵向分层和隔离

“层（Layer）”在面向对象的系统设计中是指对软件应用系统中的功能模块和类、或子系统等粗粒度的分组机制，每一层具有相对独立和内聚的职责——各层之间的依赖关系应该是逐层向下而不能跨层产生依赖关系；而且只能是上层依赖于下层、而不能让下层反过来依赖于上层。

9.1.2合理地对软件应用系统在纵向方向进行分层隔离设计

通过合理地对软件应用系统在纵向方向进行分层隔离设计——如目前的C/S和B/S等架构模式中的各个分层策略，将允许设计人员将复杂的软件应用系统中所涉及的各个方面的问题分解成多个不同层次的实现。每一层最多只影响其相关联的上、下两层，同时只要给相邻的上、下层提供接口或者实现接口，从而也就能够允许每层使用不同的方法包括不同的技术来实现，因此为软件应用系统的可重用提供了强大的结构支持。

9.1.3层与层之间应该要相互分离

在软件应用系统的多层架构设计中，一般都倡导将软件应用系统中的界面显示功能、业务逻辑处理功能和数据访问功能完全分离（但按照MVC架构模式的划分规则，在软件应用系统的系统架构设计中的业务逻辑层以及数据访问层中的各个组件都属于MVC架构模式中的模型组件），而不要在软件应用系统的界面组件（一般处于系统的表示层）中加入应用逻辑功能实现，从而避免了在应用系统的需求发生变化时（功能方面或者非功能方面），而产生牵一发而动全身的连锁修改软件应用系统的设计方案和功能实现的代码，可以实现软件应用系统的松耦合和良好的系统可维护性。

**9.2软件应用系统在横向结构方面需要拆分为不同的功能模块**

9.2.1横向拆分为不同的功能模块

常用的设计策略是遵守面向对象设计中所倡导的依赖倒置原则、并应用控制反转和依赖注入技术（IOC，Inversion of Control）。

9.2.2模块之间依赖关系应该要尽可能达到低藕合的模块设计目标

应用面向对象设计中所倡导的依赖倒置原则实现将依赖关系倒置为依赖接口，从而在很大程度上阻止了“由于需求发生变化而导致代码修改”的变化波及范围的扩大，有效地隔离了“变化”和有助于增强软件应用系统的可重用性和可扩展性。

利用控制反转技术能够消解框架系统和软件应用系统之间的依赖关系——因为利用“控制反转”技术能够减少对象的请求者对服务提供者的特定功能实现逻辑的依赖，此时系统中的各个组件类不再需要自己去查找或者实例化它们所依赖的其它目标组件类的对象实例。