足迹APP

软件架构视图说明

版本 1.0

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 4/5/2022 | 1.0 | 添加逻辑视图、部署视图 | 陆浩旗、赵浩如 |
| 4/25/2022 | 2.0 | 添加剩余视图 | 陆浩旗 |
|  |  |  |  |
|  |  |  |  |

**目录**

[1. 简介 4](#_Toc101894445)

[1.1 目的 4](#_Toc101894446)

[2. 用例视图 5](#_Toc101894447)

[3. 逻辑视图 6](#_Toc101894448)

[4. 进程视图 8](#_Toc101894449)

[5. 部署视图 8](#_Toc101894450)

[6. 实现视图 9](#_Toc101894451)

[7. 技术视图 9](#_Toc101894452)

[8. 质量属性的设计 10](#_Toc101894453)

软件架构文档

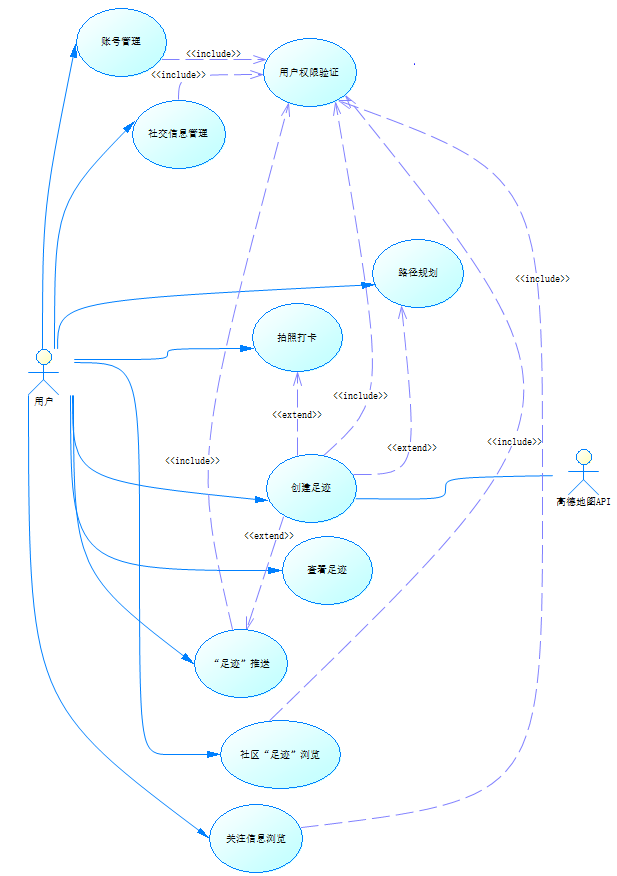
# 简介

## 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。

该文档的主要目的是为了让开发人员能够更好地理解项目整体的架构，方便他们进行开发，以及用例视图能够让玩家更好的理解项目的使用方法。

# 用例视图



足迹APP的用例和用例实现，具体详见之前的作业.

# 逻辑视图

足迹APP在架构设计时，采用了三层架构（3 Tires Application），即为了“高内聚，低耦合”的思想进行架构设计。主要分为应用层，业务层与中间件。

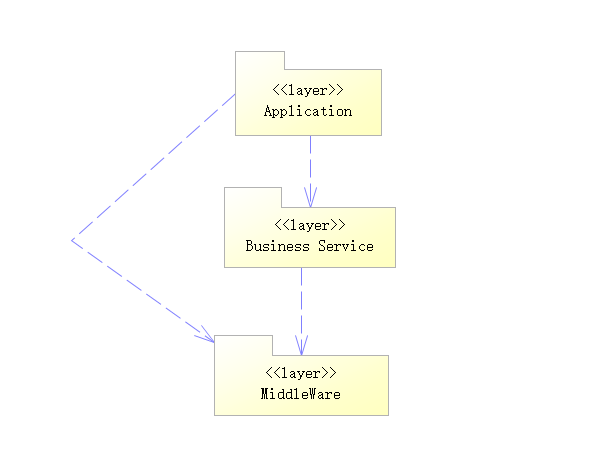


图3.1 主要层次架构

首先是应用层，客户层为了两类客户端，一个是用户所使用的移动APP客户端，使用React Native框架进行设计；另一个是管理员使用的Web网站管理客户端。客户层通过http请求和基于http的XML请求与服务器（业务层）进行通讯。应用层主要由登录与注册两个功能，经过用户权限验证后进入GUI框架，至此用户可以选择丰富的APP功能。

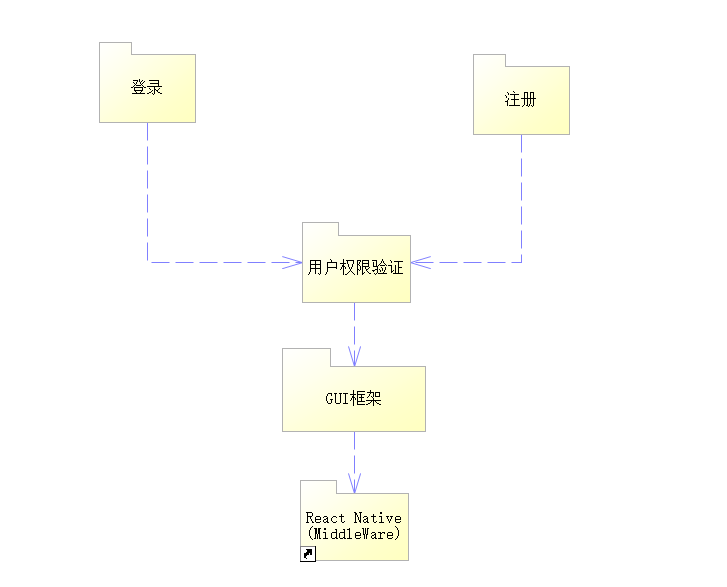


图3.2 Application模块

然后是业务层，当业务层接收到http请求后，通过编写的路由进入到对应的业务组件中。目前业务层主要分为路径规划业务、个人足迹业务与社区交流业务。路径规划业务实现路径规划主功能；个人足迹业务实现创建足迹、拍照打卡、查看足迹功能；社区交流业务实现足迹推送、社区足迹浏览、个人足迹浏览与点赞评论等功能。

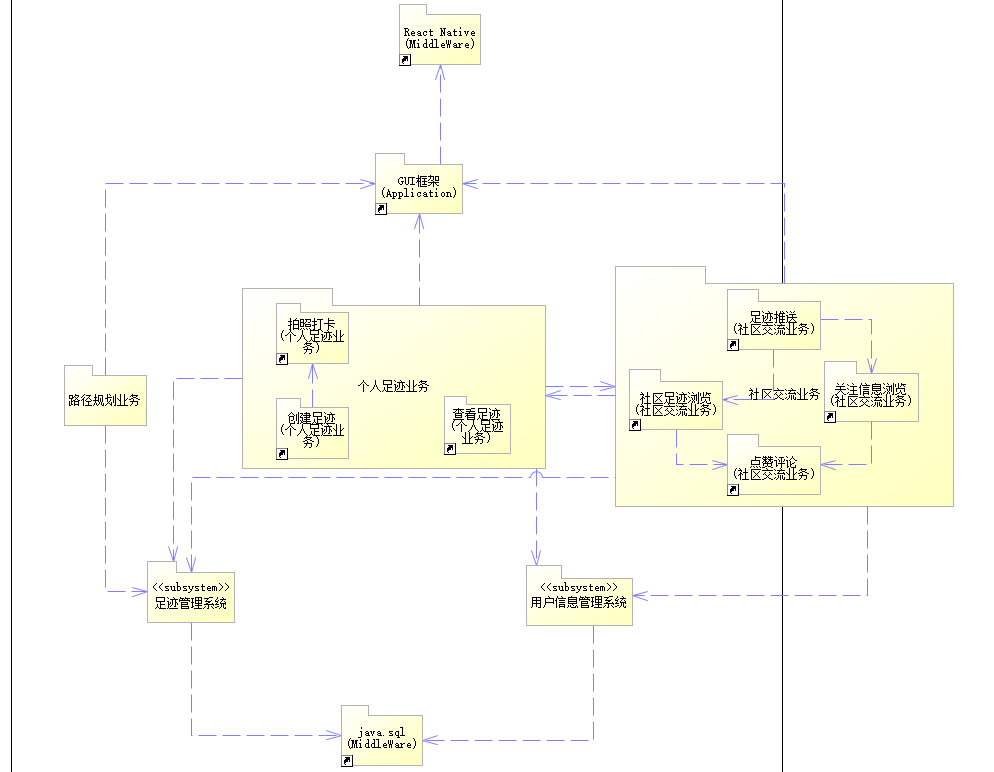


图3.3 Business Service模块

中间件部分中最主要的是用于搭建主体框架的相关中间件（如java.lang 、java.io、 react native 等）以及与数据库进行通信的java.sql ，实现对应的接口。

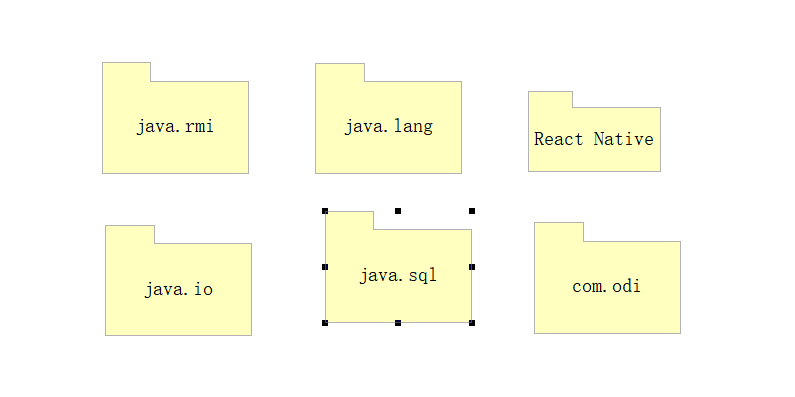


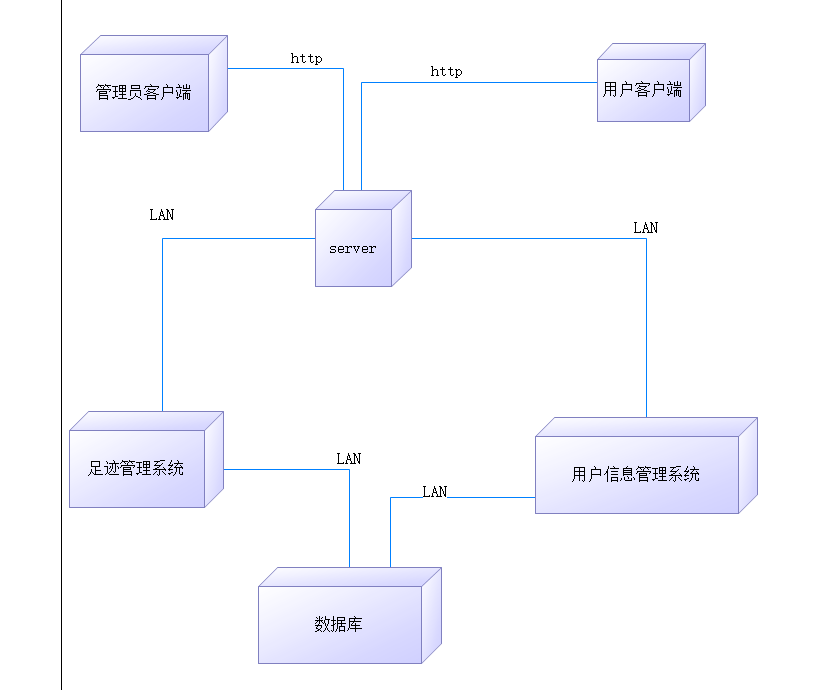
图3.4 MiddleWare模块

# 进程视图

本项目前端采用React Native框架，后端采用Spring框架，两者通过HTTP请求互相通信。

所有的进程调换由框架进行实现，在开发者看来前后端各自可以看做一个进程，而不用在内部具体管理更加细致的各类资源分配的操作，因为技术所限以及涉及的用户规模量不大，内部也没有细分线程去提高效率。因此此处只做简单介绍，重点在于其他视图。

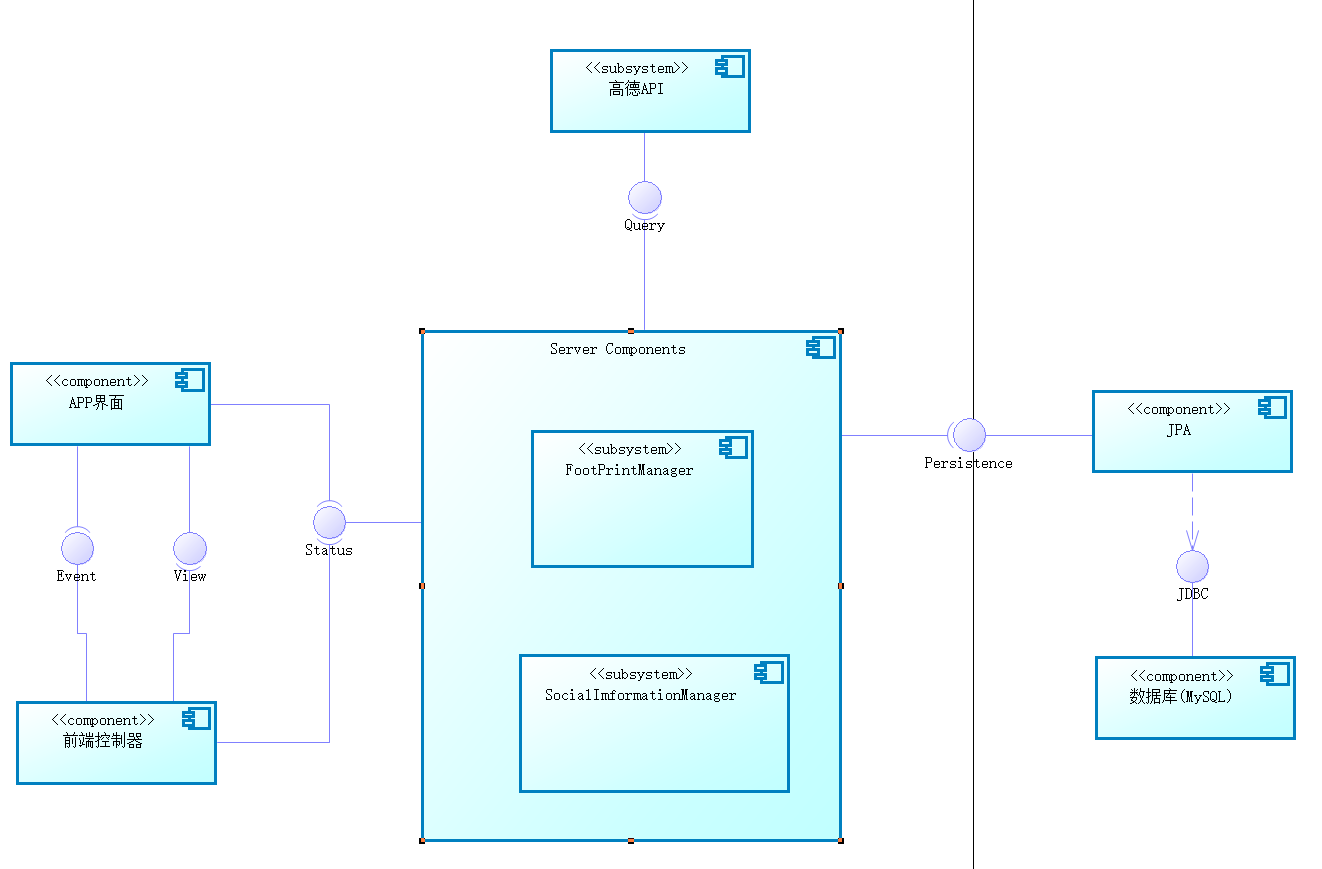
# 部署视图



足迹APP的部署视图主要可以分为六个节点，两个客户端（一个用户客户端和一个管理员客户端），一个总服务器server、两个管理子系统（足迹管理系统、用户信息管理系统）和一个MySQL数据库。

客户端和服务器使用互联网的http协议进行通信。服务器和子系统之间、子系统和数据库之间，使用局域网进行通信。

# 实现视图



主要分为前端与后端，两者通过HTTP通信。前端采用React Native+amap3d等包； Spring框架，通过JDBC向数据库查询、增加信息实现持久化。

# 技术视图

|  |  |  |  |
| --- | --- | --- | --- |
|  | 前端 | 后端 | 数据库 |
| 编程语言 | JavaScript | Java | SQL |
| 操作系统 | Android OS | Linux | Lnux |
| 框架 | React Native | Spring | Spring Data JPA  MySQL |

附加说明：

前端还采用了amap3d，React-image-picker等中间件，帮助我们能够直接用JS编写React Native代码其实现一个安卓APP。

# 质量属性的设计

可维护性：

逻辑视图中选用了三层架构，这样允许开发人员可以只关注整个结构中的其中某一层；可以很容易的用新的实现来替换原有层次的实现；可以降低层与层之间的依赖；有利于标准化；利于各层逻辑的复用，这些优点都提升了软件的可维护性。

易用性：

会设计简单的使用说明（新手教程），帮助新用户使用我们的应用。

可用性：

1. 错误预防，会在release之前进行内测来确保release版本的稳定性。
2. 错误修复，我们会在release后实时跟进并修改用户在使用过程中遇到的错误，并及时出新版本等来修复错误。