时间管理app项目

软件架构文档

版本 1.0

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 8/11/2020 | 1.0 | 时间管理app软件架构文档初稿 | 崔绍杰 刘遇时 杨弈骋 周洹羽 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 5

3. 逻辑视图 6

3.1 概述 6

3.2 在构架方面具有重要意义的设计包 7

4. 进程视图 8

5. 部署视图 10

6. 实现视图 11

7. 数据视图 12

8. 质量与战术 13

软件架构文档 （简化版）

# 简介

## 目的

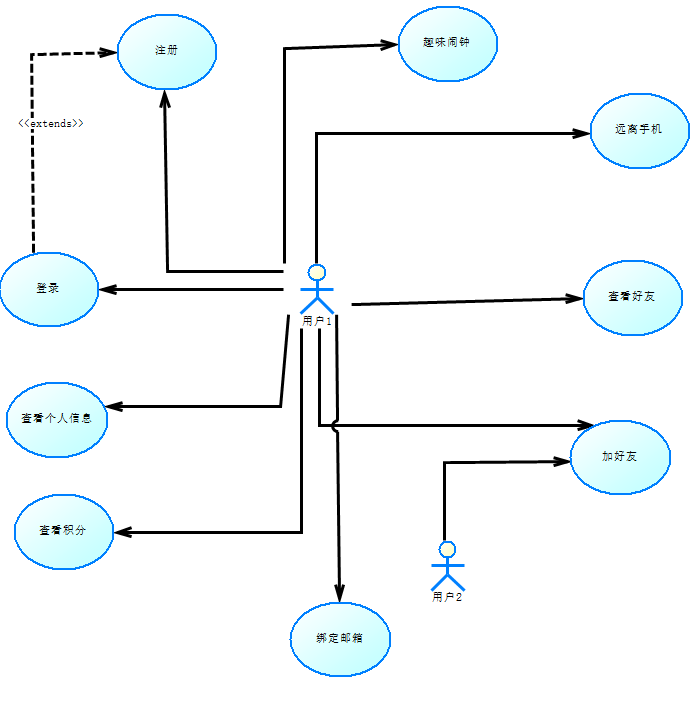
软件架构决定了一个系统的主体结构、宏观特性和具有的基本功能以及特性，复杂软件设计的成功在于软件系统有宏观层次上结构设计的正确性和合理性。因此，软件体系结构是整个软件设计成功的关键所在。

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，先是用例视图、逻辑视图，随后是进程与部署视图，最后是实现与数据视图，它们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

无

# 用例视图



本项目actor较少，除了用户本人外，只有阿里云短信服务和用户的好友。

用户和阿里云短信服务交互，可以实现“注册”和“登录”。

两个核心用例是“趣味闹钟”和“远离手机”。前者可以实现自定闹钟的时间，铃声及其响的间隔，关闭闹钟所需完成的小游戏。后者可以实现远离手机的总时长和休息间隔，软件白名单。

除此之外，用户还可以“查看个人信息”，“更换APP风格”，“绑定邮箱”。此外，还可以与别的用户交互，达成“查看好友”，“加好友”。

# 逻辑视图

## 概述

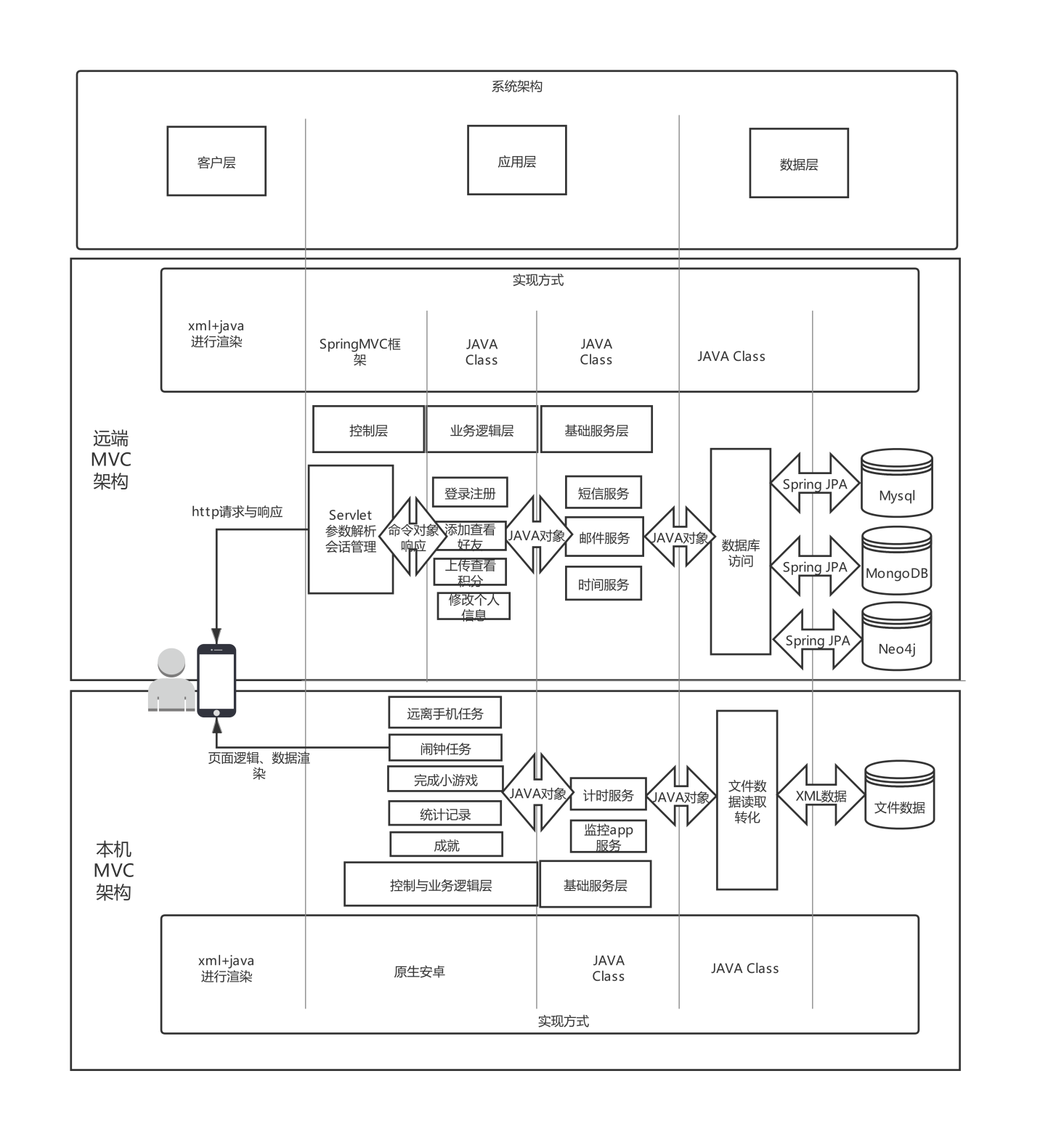
时间管理app的逻辑视图由5个主要包组成：用户界面，两个业务服务包（本机与远端）和两个业务对象包（本机与远端）。

用户界面包含与用户交互的页面渲染，以及用户与系统进行通信的表单类，它们支持登录、注册、查看个人信息、存储闹钟和远离手机任务等。

远端业务服务包含远端主要的业务逻辑，如添加好友、查看已添加好友等。

本机业务服务包含本机主要的业务逻辑，如存储与查看闹钟、小游戏逻辑等。

业务对象包括各个实体类，远端的用户实体类、积分明细实体类，和本机的任务实体类等。



## 在构架方面具有重要意义的设计包

逻辑视图分为两个MVC框架，一个是手机界面与远端交互的MVC框架，一个是手机界面和本机业务逻辑交互的MVC框架，均分为客户层，应用层和数据层。

**3.2.1远端业务对象包**

数据层进行业务对象的存储和访问，其中以mysql存储结构化数据，mongodb和neo4j存储非结构化数据。

**3.2.2 远端业务逻辑包**

包含短信验证码发送、邮箱验证码发送等服务的基础服务层

包含登录注册查看添加好友修改个人信息等主要业务逻辑的业务逻辑层

包含与手机交互，会话管理的控制层

**3.2.3本机业务对象包**

数据层同样进行数据的存储和访问，除了铃声以外的文本数据全部以xml文件形式存储

**3.2.4本机业务逻辑包**

包含计时服务等服务的基础服务层

包含执行远离手机任务，闹钟任务，小游戏等的主要逻辑层

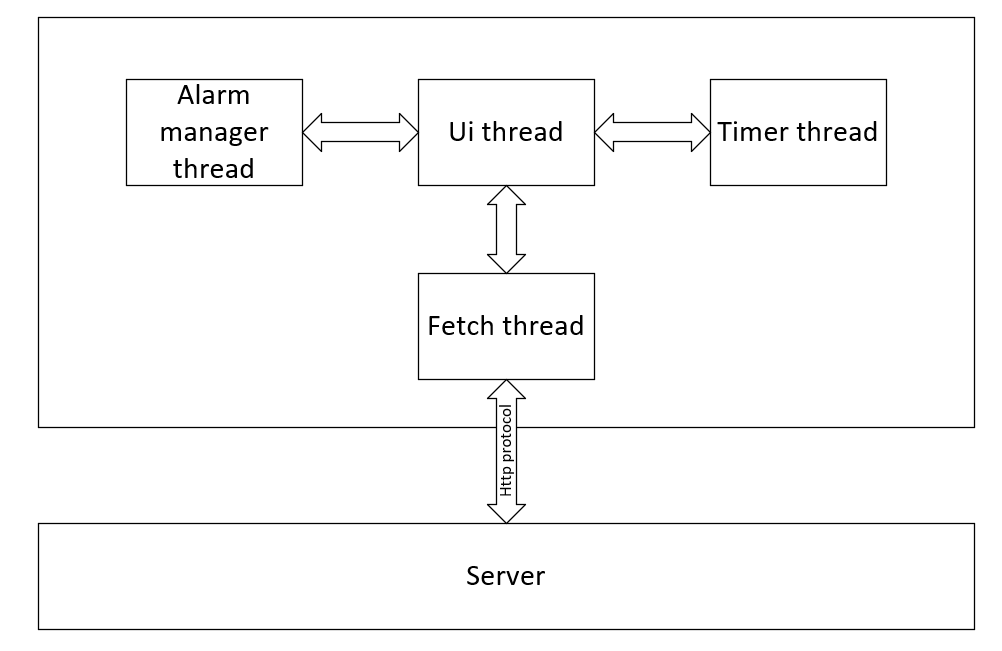
**3.2.5用户界面**

与本机直接交互，应用层处理完逻辑后，向客户层发送新的数据，客户层根据数据内容重新渲染手机页面；与后端使用http协议进行请求与响应，并根据响应内容重新渲染手机页面

# 进程视图

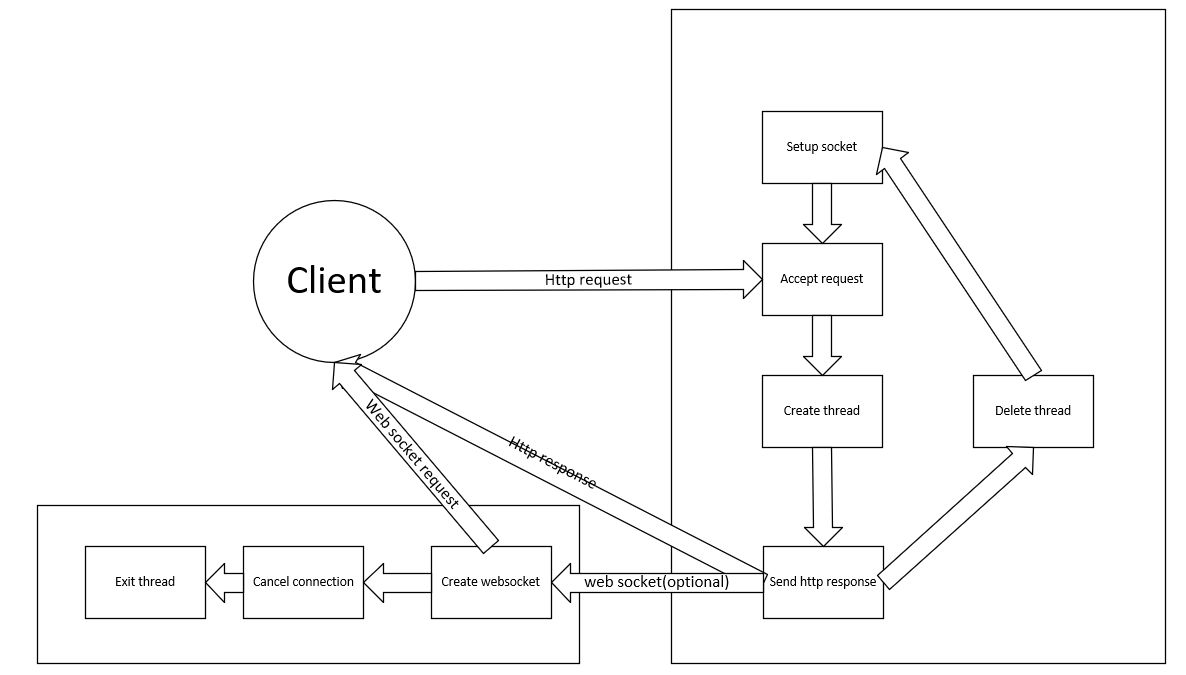
由于前后端之间只有http进行通信，耦合度很低，所以我们将两部分的进程视图分开。

前端：



前端设计4个thread，UI thread作为主线程，当用户使用相关功能时，UI thread就会会分别调度其他三个thread，而其他thread在任务完成后，会通过message,handler等方式进行回调，将用户操作产生的数据返回给UI thread，而自身就在回调之后进行释放。其中alarm manager负责闹钟的定时，timer thread负责一些定时的功能，而fetch thread则生成http request与后端进行交互。在其他thread的运行过程中，会有一个信标变量，于是UI thread可以用这一变量来对其他变量进行中段和停止

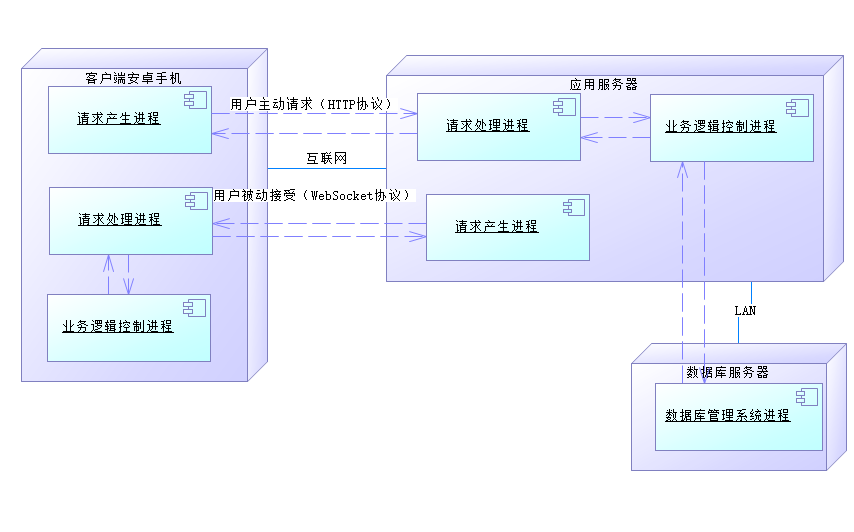
后端：



后端的进程视图又分为两部分，右侧方框为由tomcat包装的传统服务器端的结构，处理一般的http请求，而其中有一部分thread会在完成http response后再创建web socket与client进行沟通，使得后端可以自主的给前端发送消息（具体为好友请求）。当用户端不再给server端发请求，或者出现用户下线带来的连接中断，则该线程会进行消亡

# 部署视图

部署视图说明：



5.1 客户端安卓手机

该节点运行在搭载了安卓系统的移动手机上，通过互联网与应用服务器相连，运行了UI thread，Alarm manage thread, time thread, fetch thread四个前端线程。用户可以在安卓手机上下载并使用我们的app，在使用手机时APP需要做出的响应，一部分由APP自身的逻辑完成，一部分发往服务器由服务器完成。同时客户端还要处理由应用服务器主动发送的请求，比如请求添加好友。

5.2 应用服务器

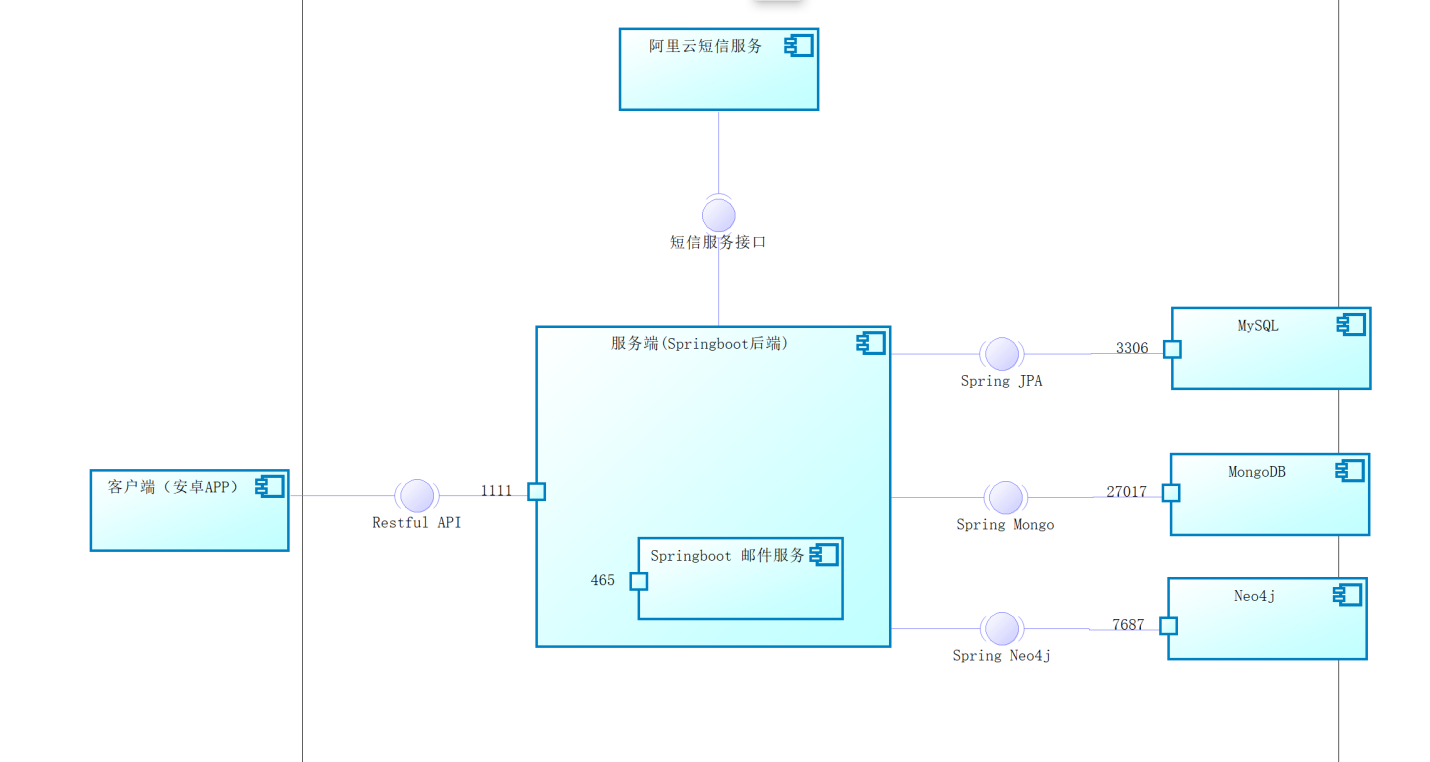
该节点运行在云服务器上，通过互联网与客户端安卓手机相连，通过LAN与数据库服务器相连，每从前端受到一个Http请求就会运行一个thread进行处理，并且会为每个正在使用的前端创建一个处理WebSocket请求的thread。应用服务器主要处理APP的联网相关功能，在收到请求后做出响应，部分请求还需要调用数据库。同时还会主动向客户端发送某些请求，比如添加好友。

5.3 数据库服务器

该节点运行在云服务器上，通过LAN与应用服务器相连。数据库服务器主要挂载了Mysql,Mongodb,neo4j数据库，在接受到应用服务器请求后作出回应。

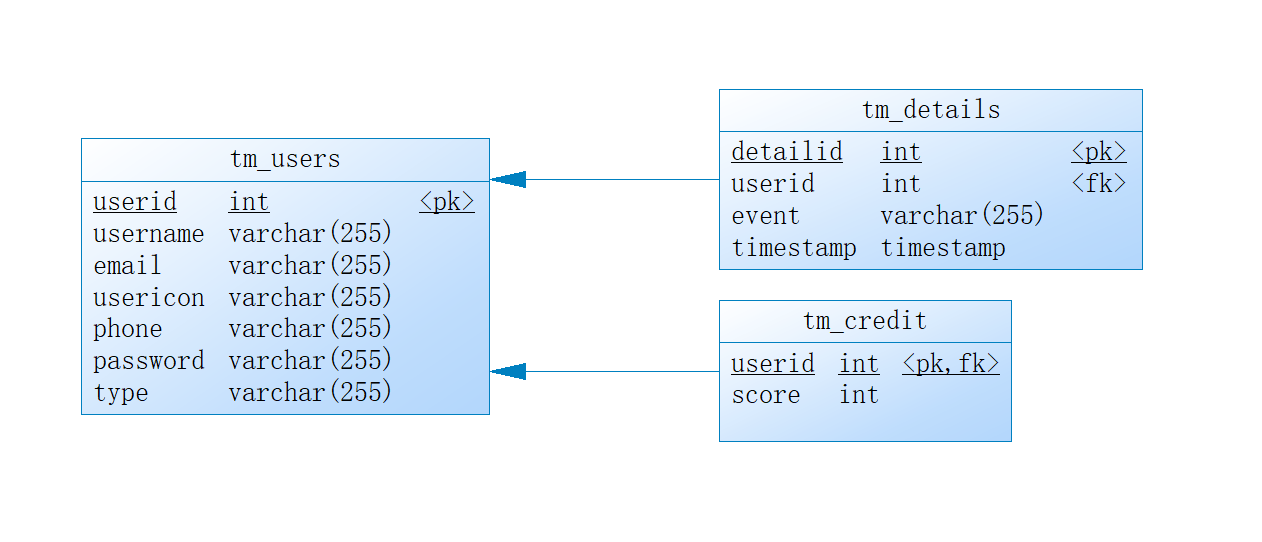
注：在开发阶段，对性能要求较低，对稳定性要求容忍度较高，出于节省成本和减轻工作量的缘故，暂时将应用服务器和数据库服务器合并

# 实现视图

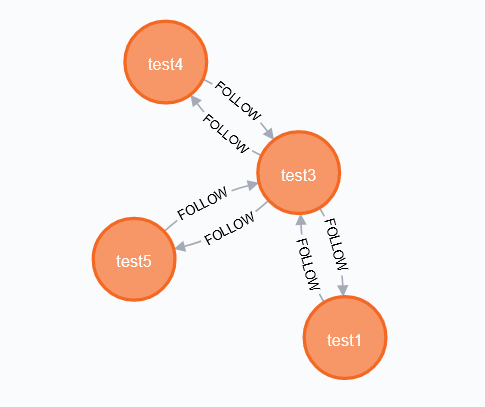


手机客户端APP通过Restful API与Springboot后端连接，服务器后端向外提供1111端口。在后端中我们采用了MySQL，MongoDB，Neo4j三种数据库，后端通过各自的接口访问数据库。Springboot后端中包含了一个能提供收发邮件的服务。为提供发送验证码功能，我们采用阿里云的短信服务，后端通过阿里云提供的短信服务接口与阿里云短信服务连接。

# 数据视图



三个数据实体：用户信息(tm\_users)，用户积分情况(tm\_credit)和用户积分明细(tm\_details)。用户信息记录了用户的用户名、邮箱、头像、手机号、密码、用户种类（用户或管理员）等信息，其中密码在数据库中加密保存。用户积分情况记录了用户当前的总积分。用户积分明细记录了用户每一次完成任务获得积分的情况。



社交系统（好友功能）在Neo4j中单独维护，上图是一个示例。好友间的FOLLOW关系代表两人已经成为好友。好友间的ASK关系代表一方向另一方发出请求。

# 质量与战术

**8.1易用性**

为用户提供适当的反馈和协助，如新手手册与引导

在任务开始前10s，为用户提供“取消”、“撤销”等命令，以免用户“手滑”。

**8.2可靠性**

错误预防：使用事务机制

错误恢复：使用log进行检查与回滚

错误检测：检测后端心跳，如后端崩溃则及时反应给客户

**8.3性能**

性能不是本app的关键质量点，故后端使用消息队列和线程池则足够

**8.4安全性**

使用安全日志，检测安全证书

**8.5可测试性**

将接口与实现分离，测试输入与输出

**8.6可维护性**

前后端均进行模块的泛化，防止连锁反应