**武汉大学计算机学院**

**本科生课程设计报告**

**“Moss”App的设计与实现**

专 业 名 称 ：软件工程（卓越工程师）

课 程 名 称 ：软件工程

团 队 名 称 ：Moss小组

指 导 教 师 一：李晓剑 教授

团 队 成 员 一:程浩宇(2016302580002)

团 队 成 员 二:戴侃(2016302580013)

团 队 成 员 三:朱昱睿(2016302580059)

团 队 成 员 四:杜会远(2016302580130)

二○一九年五月

**郑 重 声 明**

本团队呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

团队成员签名：程浩宇 朱昱睿 戴侃 杜会远 日期：2019/5/20

**目录**

**1产品定义**

**2创意来源**

**3背景调研**

3.1市场综述（分析实用性）

3.2竞争对手分析

**4需求分析文档**

4.1 核心功能

4.2 非功能性需求分析

4.2.1 可用性

4.2.2 兼容与性能

4.2.3 开发配置

**5程序总体设计**

5.1 架构层次分析

5.1.1 界面层

5.1.2 接口层

5.1.3 数据模型/处理层

5.2 数据库设计

5.2.1 ER图

5.3 UI设计

5.3.1 登陆注册界面

5.3.2 聊天界面

5.3.3 时间创建和打卡界面

5.3.4 事件总览界面

5.3.5 个人信息界面

5.4 用户流程设计

5.4.1 用户流程分析

5.4.2 时序图

5.5处理层及接口层设计

5.5.1数据接口模块

5.5.2实体类定义模块

5.5.3Service模块

5.5.4配置模块

5.5.5 Controller模块

**6 测试文档**

**7 项目管理**

7.1 项目规划表

7.2 实际进度表

7.3 团队成员分工及感想

**结论** …………………………………………………………………………………… 57

**参考文献 ………………………………………………………………………………** 59

**1□产品定义**

我们的目标是做一个基于聊天机器人的备忘录的软件应用。 特色点在于机器人的聊天风格是恋爱风格，吸引15-30岁的年轻目标用户对象。

本团队经调研后针对15-30岁的年轻用户，打造一款陪伴风聊天互动式的移动端（安卓）备忘录力求以趣味性聊天对话回复，协助用户进行时间管理

备忘录意指任何一种能够帮助记忆,简单说明主题与相关事件的图片、文字或语音资料。 聊天机器人是经由对话或文字进行交谈的计算机程序。能够用于产生日常对话，模拟用户在生活中的社交关系联结。

在恋爱类风格应用中，恋爱的本质为“白日梦”。“白日梦”具有社会表演的虚幻叙事特质，其源于玩家对虚拟理想自我的建构需求、对理想恋爱对象的期待和现实生活中对话逻辑的渗透与传播。

**2创意来源**

软件的创意来源于对手机自带和目前主流备忘录以及打卡软件的功能的整合和补充。目前主流的打卡软件功能较为单一无趣，容易造成用户无法坚持使用的情况。同时，备忘录和打卡软件在功能上有较多的共同之处，完全可以作为整合后再同一个app内使用。

同时，参考软件“叨叨记账”的聊天机器人参与记账过程，提供反馈，以此增加用户体验，确保用户得到积极反馈，增加用户的使用动力的思路，在备忘录里也引入聊天对象和聊天机器人的设计，用来度备忘和打卡事件的提醒和反馈。

**3□背景调研**

**3.1**□**市场综述（分析实用性）**

1.有实际存在的一群消费者 —— 备忘录为手机基础应用，受众广泛，需求强烈。

2.这群客户有明确的需求 —— 因为人类的记忆力有限，对备忘录有刚性需求。

3.是否有一系列产品来满足需求 ——

现在的备忘录软件主要分为两类：

• 系统自带，简约但基本不具备提醒功能。

• 市面上大部分app，主要是记录待做事项、重要事件的时间、日程安排等，且获得手机授权后，具有一定的提醒功能

但是缺少趣味性、且主要靠使用者自己整理。

4.在决策购买时，市场中消费者相互参考 —— 对日常生活工具类应用易产生使用依赖，通过趣味性进行产品营销传播。

**3.2竞争对手分析**

目前聊天机器人广泛运用于即时通讯平台，例如脸书 Messenger，WeChat，LINE和 Kik，以娱乐、零售行销、以及客服为目的。此外，即时通讯平台提供易于整合的webhook，使得第三方开发商易于可通用于不同通讯平台之聊天机器人。这些软件机器人以客服的身份出现或是成为团体聊天的一员。

目前Siri等手机语音助手可以通过聊天功能完成日常事务的记录与提醒，但是对话过程较为枯燥乏味，且无法完成在聊天机器人中删除备忘录的功能。

在日常工具类应用中，很少有使用趣味聊天机器人为主要交互类型的应用。本应用正是为了填补这一空白市场而提出的。

**4核心功能需求分析**

**4.1核心功能**

* **用户管理**
* 实现用户的注册、用户登录退出、忘记密码、找回密码等功能
* **Tips：**本地和远程备份相协调。（备忘等事件储存在本地，用户可以选择是否云端备份，用户名密码信息存在云端）
* **用户聊天对象设置**
* 设置聊天对象的头像，称呼，姓名等。用户自定义聊天界面的背景。
* **Tips：**前期可以是简单的机械式回答，后期可以做成复杂的语料库来丰富聊天内容的匹配程度，增加趣味性。
* **备忘事件功能**
* 基本的备忘录功能。即创建备忘（设定备忘时间，是否提醒，是否每天提醒，备忘标题和备忘内容等），编辑备忘，删除备忘，标记完成备忘
* **Tips：**被标记为完成的备忘在程序内保留一定时间（例24h）后删除。创建备忘的必选项为备忘标题，其余皆有默认值。
* **打卡事件功能**
* 基本的打卡事件功能。包括打卡事件的创建（打卡事件标题，开始及结束日期，打卡周期（每天，每三天，每周等）），打卡事件的编辑，删除和标记完成。以及打卡事件的标记打卡。
* **Tips：**主要功能和备忘录功能相似。打卡事件的有总体完成和每周期完成的区分。备忘录和打卡时间的创建可作为相同弹窗不同标签页
* **聊天交互功能**
* 聊天界面应该包含的接口有，备忘录和打卡事件的创建接口，打卡的每周期完成标记，和备忘以及打卡事件的完成接口以及所有接口相应的聊天内容创建和反馈。
* **Tips：**这个页面可以作为首页呈现。
* **事件管理功能**
* 展示所有的备忘事件和打卡事件（包含打卡事件当前进度（已创建多久和打卡完成情况）），以及筛选功能（展示特定时间，或特定类型（备忘or打卡）的时间）
* **提醒功能**
* 对于用户标记的需要提醒的备忘和设定周期的打卡，用应用提醒的方式提醒，或者发送邮件，发送短信的方式强行提醒。
* **Tips：**后期可以实现，生成当日事件的图片（可以作为锁屏壁纸的那种）
* **日历式展示功能**
* 日历界面展示所有事件。
* **适配**
* 包含设备本身适配，不同安卓系统的适配（深度定制化安卓系统太多了），以及手机端和WEB端的适配

**4.2非功能性需求分析**

**4．2．1** **可用性**

充分测试，出错处理并充分考虑程序崩溃无响应的出口，界面逻辑清晰，操作方便友好等等。

**4．2．2** **兼容与性能**

上一节适配中也提到了，兼容就是适配，然后在性能要肯定要运行流畅，不会出现闪退、ANR（？？）、内存泄漏（buffer overflow）等问题，主要体现在数据流、图片缓存、列表优化、数据库操作等等方面。

**4．2．3** **开发配置**

预设应用场景：2000用户，可承受100-200并发访问不卡顿

• 服务端：Android端，Web端可构造网页进行导流，然后后台管理界面就写成web端

• 服务器：基本是文本传输，因为目前还不涉及云同步问题，只有Android端，只是涉及更新语料库问题，100-200并发访问量下，压力比较小。4核8G 5M带宽已经是高配了，

当然后面还有更多技术，如果真做大的话，什么集群，负载均衡，异地缓存什么的

• 数据库：暂定Mysql Community Server 充分学习优化技术 必要时可以考虑升级 Mysql Enterprise Server

连接池技术，最大连接数

• 短信服务：阿里云短信服务SMS，用以发送验证码，短信收费，一定是必要验证时发送

• 身份认证：需要一定的计算性能，但是4核应该能Cover这个认证又不是并发的。

**5程序总体设计**

**5. 1实验原理**

前端：web官网 HTML+JS+CSS

Android前端：基本layout+OKHttp+Retrofit

后端：SpringBoot+Redis+Mysql

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。用我的话来理解，就是 Spring Boot 其实不是什么新的框架，它默认配置了很多框架的使用方式，就像 Maven 整合了所有的 Jar 包，Spring Boot 整合了所有的框架。

Spring Boot有这么几个特性：创建独立的Spring应用程序，嵌入的Tomcat，无需部署WAR文件，简化Maven配置，自动配置Spring，提供生产就绪型功能，如指标，健康检查和外部配置，绝对没有代码生成并且对XML也没有配置要求。

Redis(Remote Dictionary Server，远程数据服务)是一个开源缓存数据库，其中包含五大数据结构，是基于键值的存储服务系统，并且高性能、功能丰富。项目中主要使用了List这一数据结构，用Redis的list构造了一个聊天记录的队列，每次用lrange去list中得到相应的聊天记录。

**5. 2 架构层次分析**

**5. 2. 1** **界面层**

• 登录

• 注册

• 聊天界面

– 聊天主界面

– 所有类型备忘录预览

– 打卡快速写入界面

– 普通备忘录书写界面

• 我的Moss

– 个人信息界面

– 设置界面

– 分享界面

• 总览

– 日历界面

– 查看所有备忘录界面

– 专门打卡界面

**5. 2. 2** **接口层**

• 接口层封装了服务器提供的API，定义一个请求引擎类，对请求的发送响应结果进行处理，方便给核心层调用。

• 搜索 用关键词找到商家

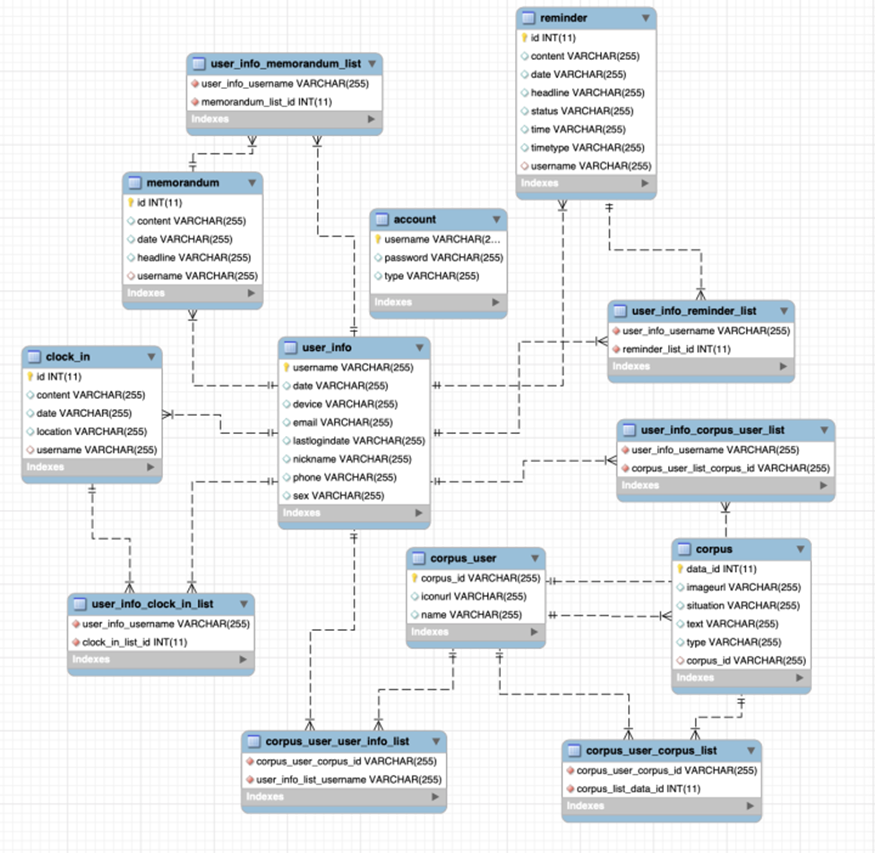
• 首页信息 要么抓取，要么自己自动生成新闻（这就涉及到机器学习的技术了，但是很明显我们的机器是跑不成机器学习任务的，算力不够，做实验可以

**5. 2. 3** **数据模型/处理层**

• 根据不同板块的需求，存取数据。

**5. 3数据库设计**

**5. 3. 1 ER图**



其中，较为重要的数据表的属性和功能说明如下：

• **用户列表**

users\_list

(

userID(PK),//唯一标识，系统自动生成

username(FK) REFERENCES users(username)//username 可以是邮箱或者手机号码

)

• **用户信息表**

users

(

username(PK),

vipID(FK) REFERENCES vip(vipID)

sex,

nickName,

phone,

email,

registerDate,

lastLoginDate,

lastLoginDevice,

lastLoginLocation,//安全保护

QRcode //要内嵌分享按钮 或者二维码生成程序

)

• **登录表**

account

(

username(PK),

type, //common users or DBM or Admin

password //extremely important 传输加密

)

• **会员表**

vip

(

vipID(PK),//暂定，其实这个版本用不到

vipStartTime,

vipEndTime,

vipRestTime,

vipScore,

vipDiscount,

vipLevel,

autoRenew,

)

• **备忘录列表**

memorandumList

(

memorandumID(PK),//全局检索id

Headline，//标题

username,

Content,//限制300字

date

)

• **提醒表**

reminder(

reminderID(PK),

Headline,

Type,

username,

content,

Remind,//boolean True/False

Time,//几点

Date,//特定日期

Status //是否已打卡

)

• **打卡**

clockin(

clockID(PK),

username,

content,

location,

date

)

• **用户-语料库表**

corpusUser

(

Id(PK),//全局检索id

username(FK),

corpusId(FK),指定类型语料

name，//聊天角色名 可由用户自己定义

iconURL,//用户自行上传图片作头像

)

• **语料表**

corpus(

dataId(PK)//全局检索id

corpusId,

type,//image or text

situation,

originname,//

imageURL,

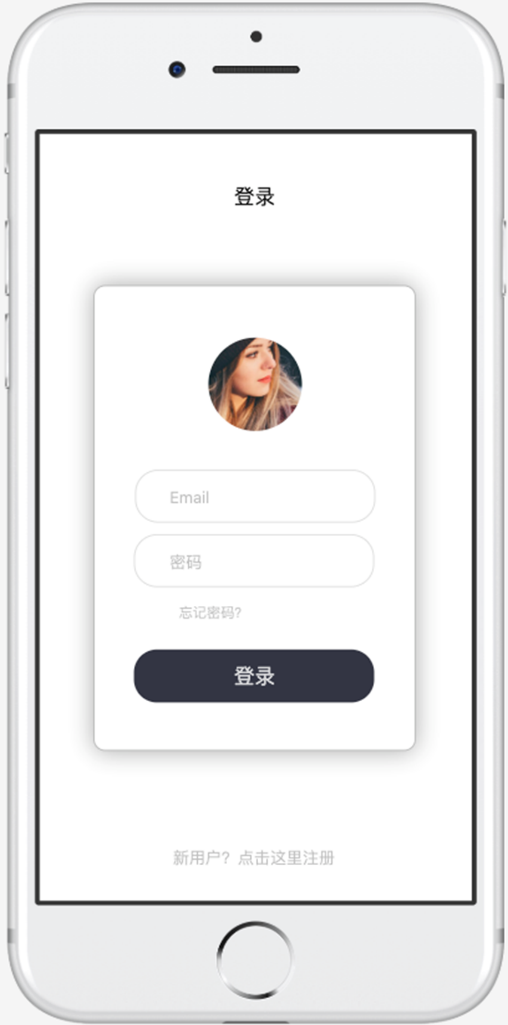
)

**5. 4 UI设计**

采用原型设计工具modao完成了app较为完整的原型设计，确定了各个页面的组成，功能实现，美术设计和跳转关系。

具体界面和功能如下：

**5. 4. 1** **登陆注册界面**

登录注册界面包括基本的用户注册，用户登录，密码查询，以及两个界面之前的相互跳转功能。

**5. 4. 2** **聊天界面**



聊天界面为软件的主界面，包含创建备忘和打卡事件，完成打卡并得到语料库反馈等功能，同时拥有跳转到备忘录和打卡事件总览，跳转个人中心等功能。还可以有自定义头像，自定义聊天背景等辅助功能。

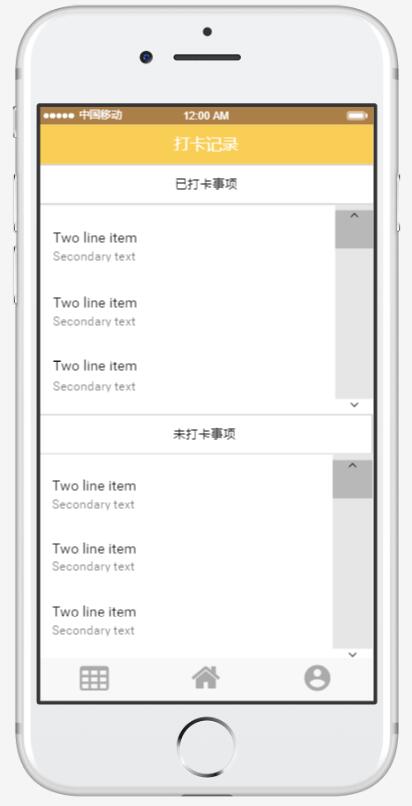
**5. 4. 3** **事件创建和打卡界面**

额外弹窗方式创建备忘录和打卡事件，便于多个不同界面的统一调用，增加复用性。

打卡完成界面基于聊天界面创建，完成后获得鼓励反馈。

**5. 4. 4** **事件总览界面**

备忘事件和打卡事件的总览页面，查看用户创建的所有事件及其状态.

**5. 4. 5** **个人信息界面**



应支持头像昵称等的自定义，非主要功能。

可增加的页面还有：

• 设置界面

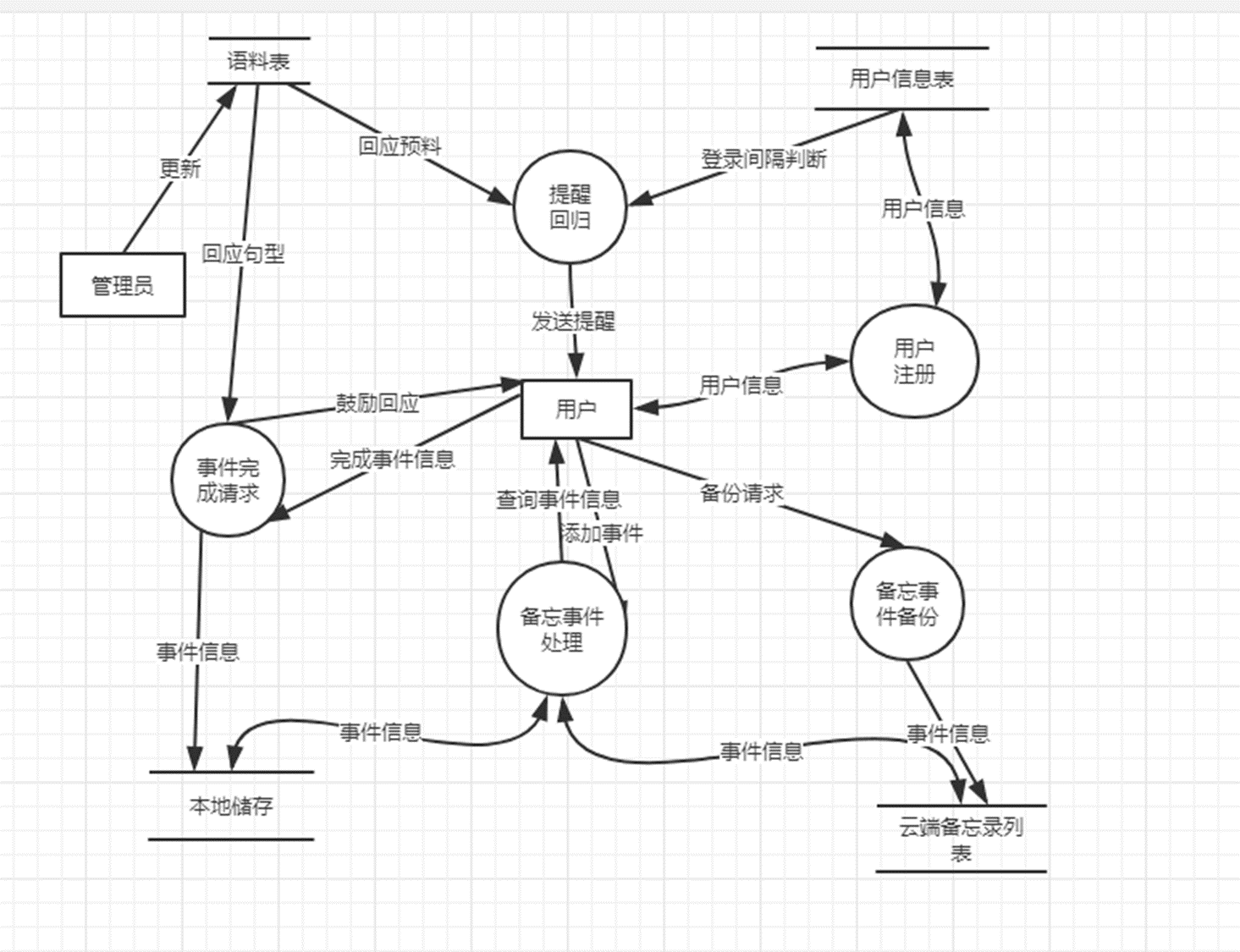
• 日历方式浏览备忘和打卡事件界面

• 提醒模式设置界面。

**5. 5 用户流程设计**

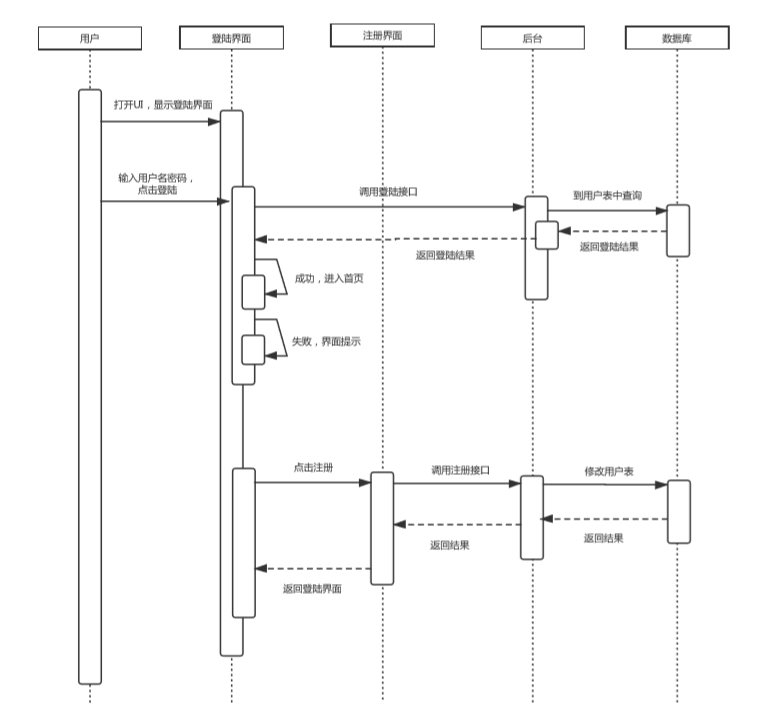
**5. 5. 1** **用户流程分析**

数据流图如下：

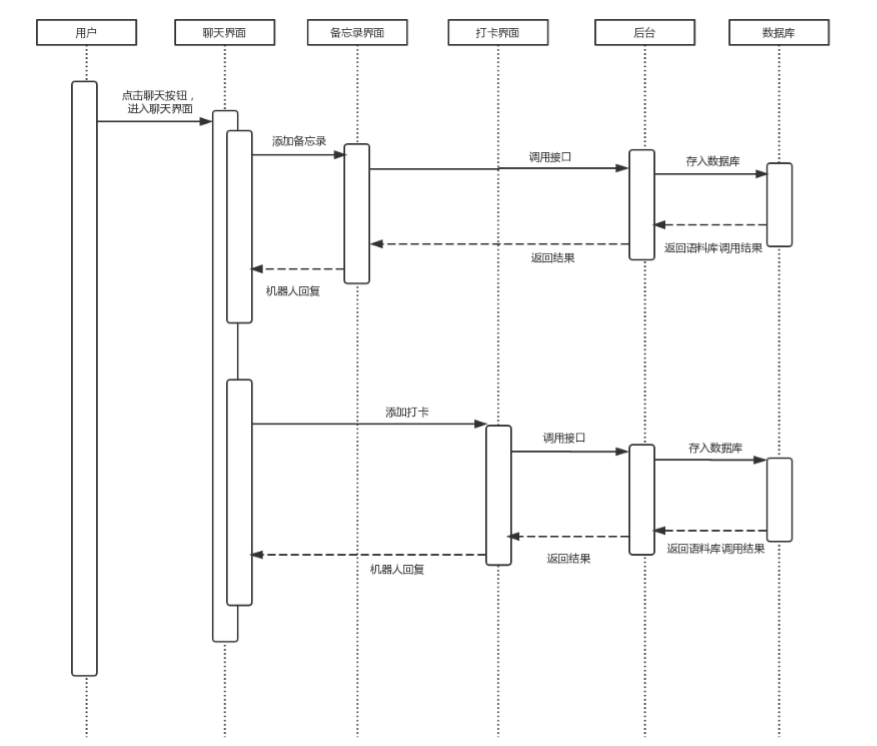


**5. 5. 2时序图**

用户登陆注册时序图：



聊天界面时序图：



**5.5处理层及接口层设计**

该部分的大致结构为：

• Service模块

• 数据接口模块

• 实体类定义模块

• 配置模块

• 控制器模块

处理层的类图结构如图所示：

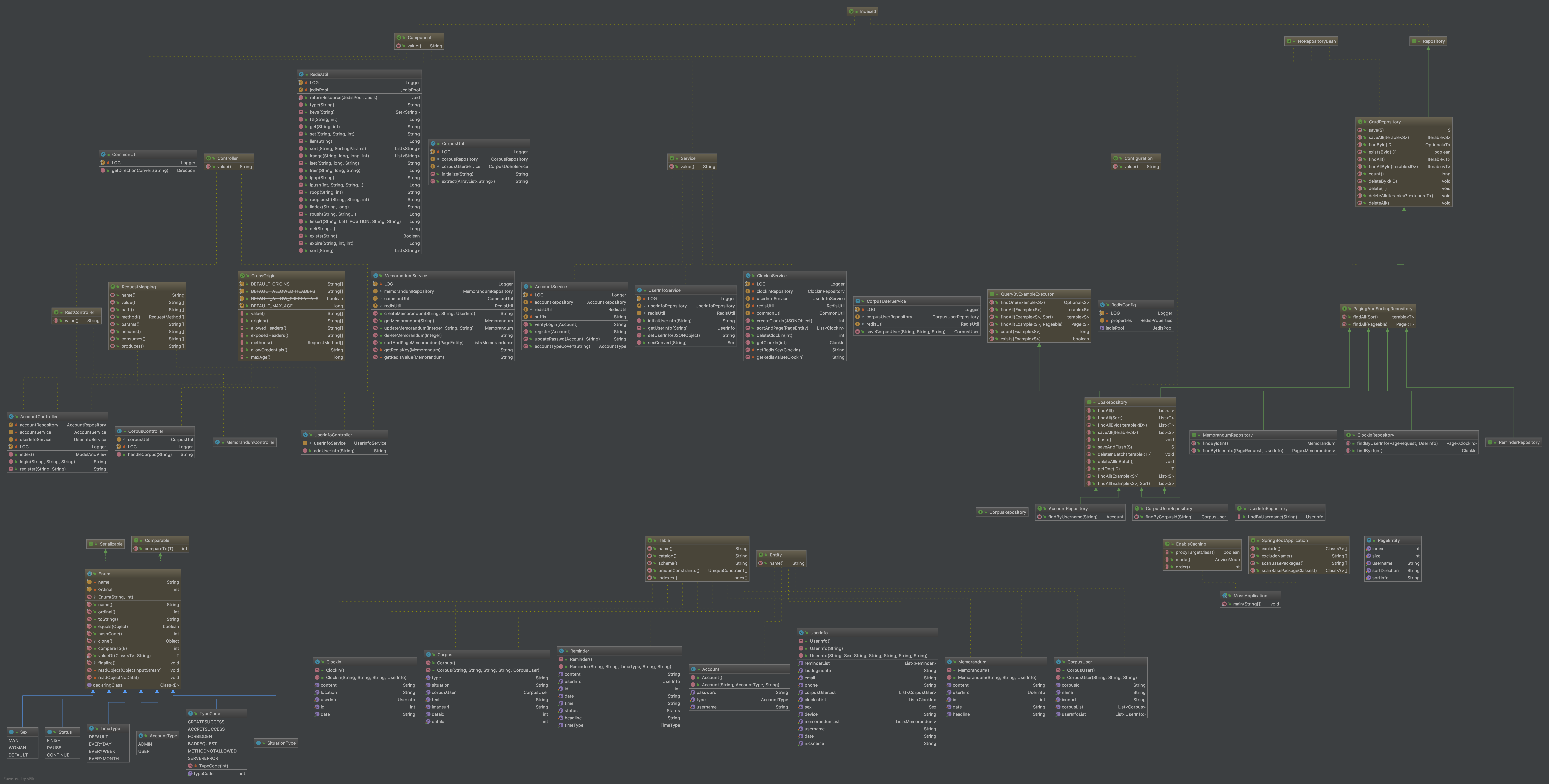


图5.5:总的类结构图

**5．5．1** **数据接口模块**

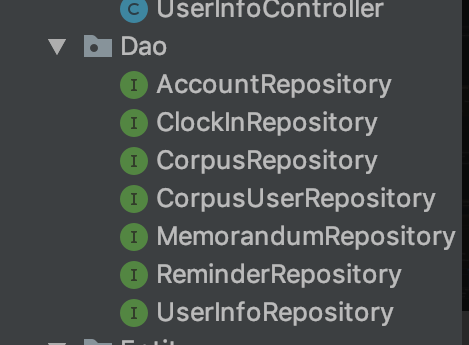


图5.5.1-1:dao层接口类

Spring相关的一共有两种ORM框架，一种是比较灵活的Mybatis框架，需要自己手写sql语句并注入相关属性，另一种就是hibernate框架，用户完全对repository实例进行操作，通过调用相关方法名，生成sql语句，sql语句在开发中是不可见的，运行时可打印在后台。而本项目所使用的Spring Data JPA就是对hibernate的进一步封装。 只需要调用几个方法就可完成对数据库的CRUD的操作

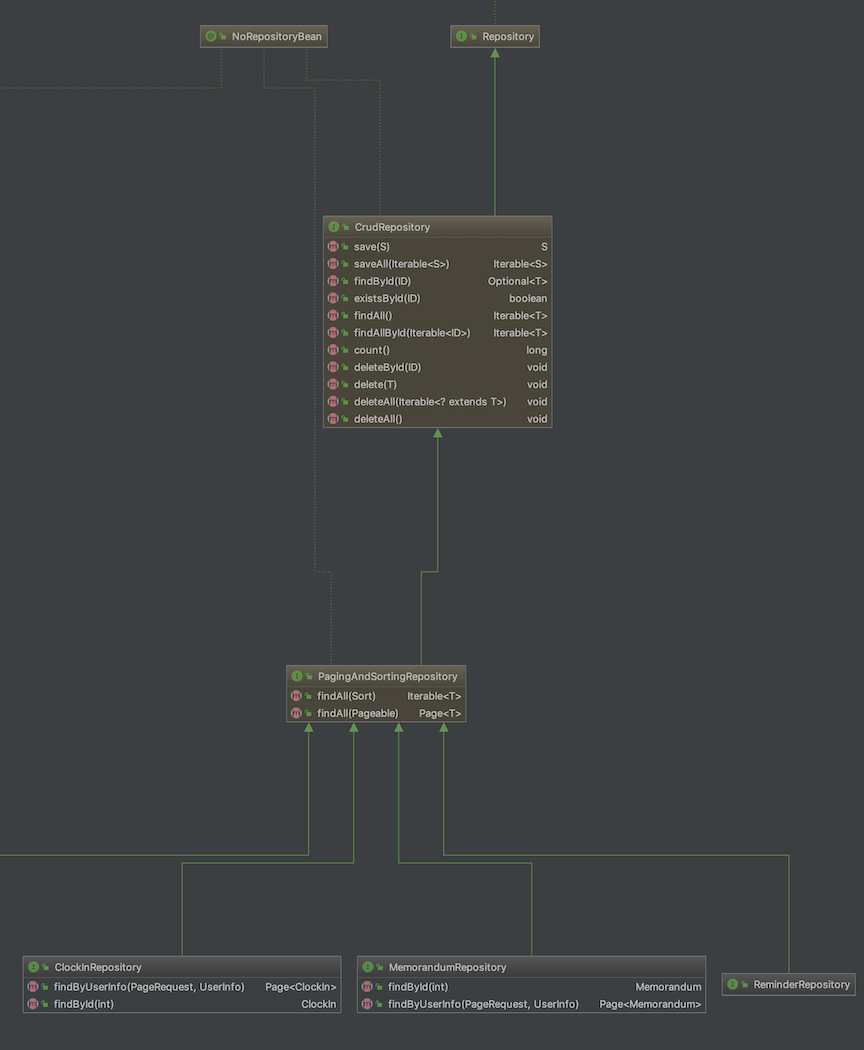


图5.5.1-2: Repository接口类之间的继承关系

从这张图可以看到，在Spring体系中从数据库角度看，bean分为两类，NoRepositoryBean和Repository 两大接口，CrudRepository继承了Repository接口，并给出了基本的数据库操作方法比如 save，find，delete，count（聚合函数），exist等。

而PageAndSortRepository又继承了CrudRepository并在此基础上进一步增加了find（Sort） 和 find（Pageable）两大模版函数，其实从SQL语句角度很容易理解，无非就是在最终Hibernate生成SQL语句中加入了 order by 和 limit 限定。

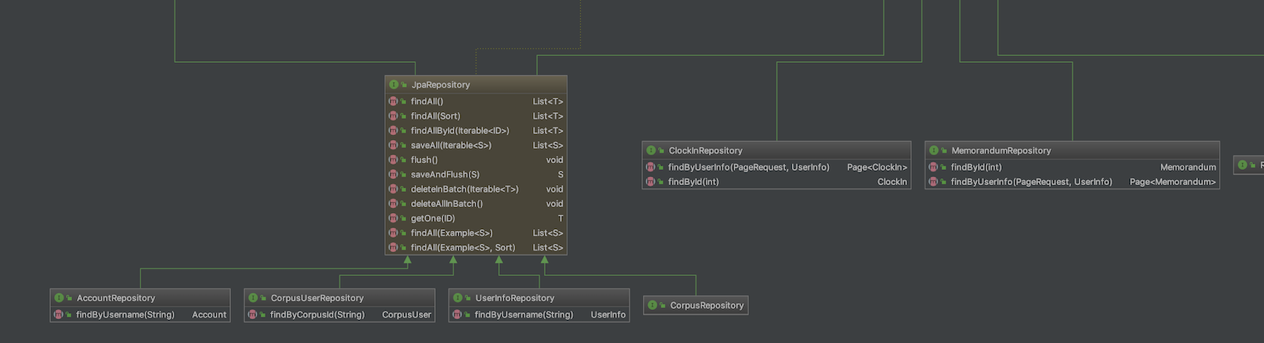


图5.5.1-3: 最终的JPA接口

根据5.2的数据库设计，总共是ClockIn，Corpus，CorpusUser，Memorandum，Reminder，Account，UserInfo 7个主表，其中，ClockIn Memorandum以及Reminder三个表继承SortAndPageRepository做了分页排序的功能，其余四张表的接口则是继承了JpaRepository这个通用大类。

**5．5．2** □**实体类定义模块**

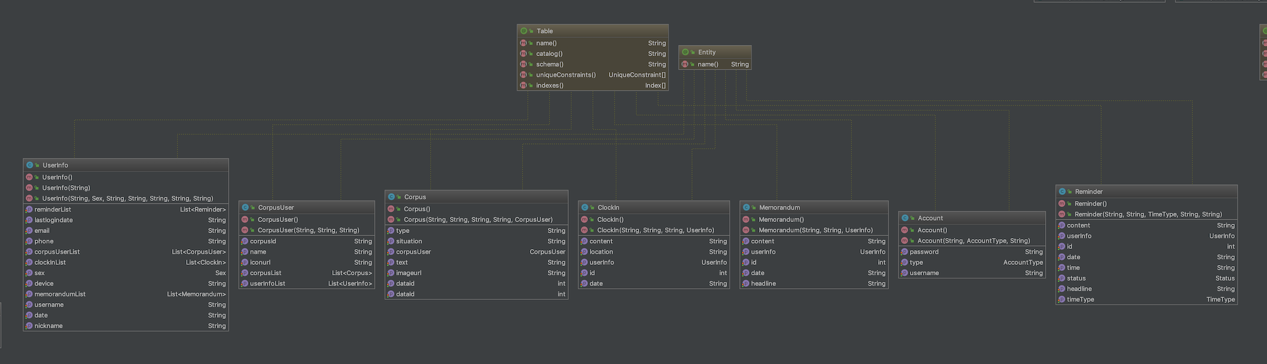
****

图5.5.2-1: 所有Table表

有了Spring Data JPA 建表就非常轻松简单，只需要引入javax.persistece.\*中的各色永久层注解，设定相应的Entity名，Table名，以及Table总各字段的Column名即可。

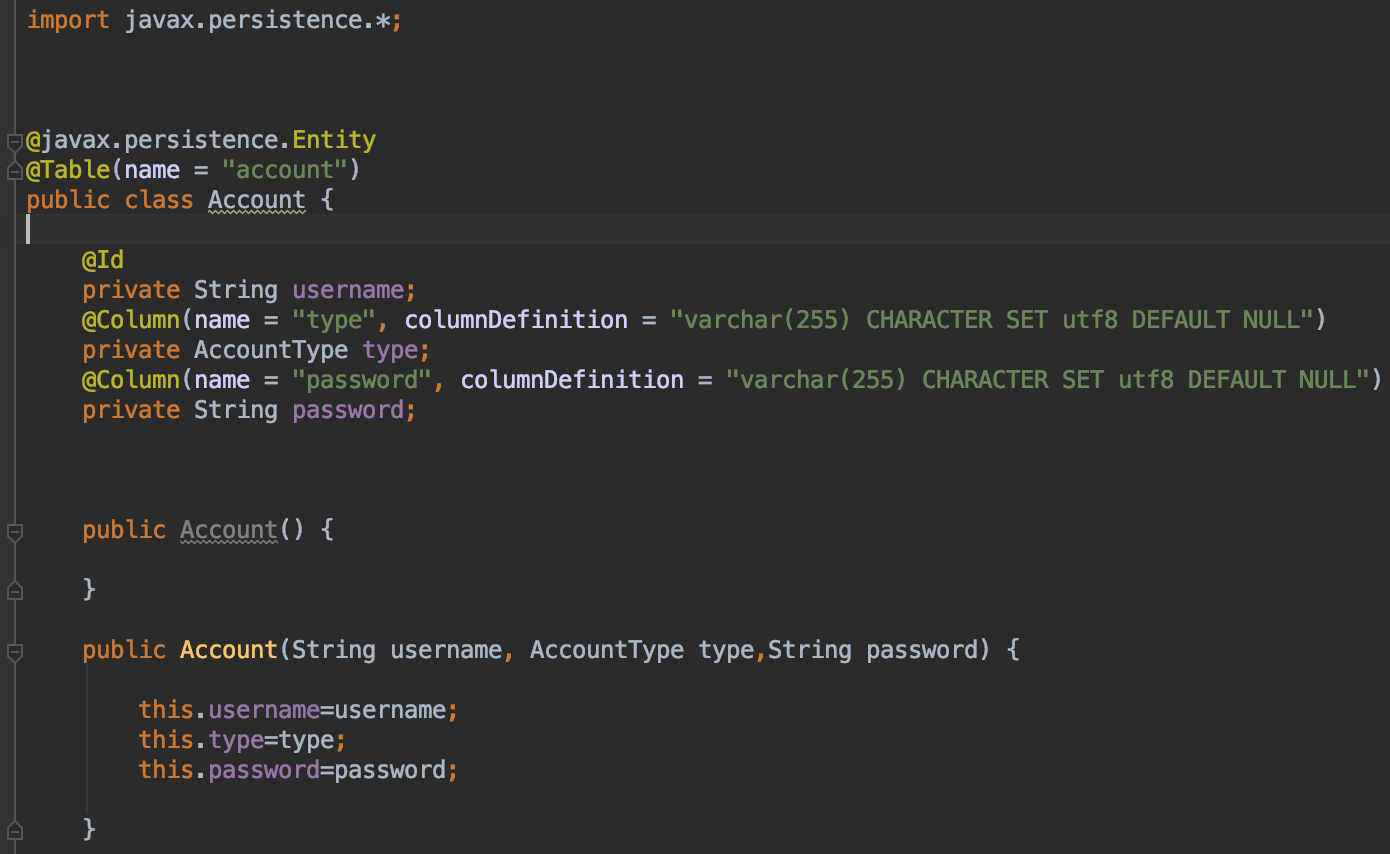


图5.5.2-2: Entity类举例

Spring最出名的特性就是DI，DI是IOC的主要实现手段，这里Column注解配置，注解属性输入，就是靠的DI，装配一个bean，有好多种实现方式，可以用xml配置，也可以用java配置类，还有一种方式，就是通过注解，Spring容器会对所以文件进行扫描，将对bean的定义构造为bean definition 封装在bean类里，由beanfactory进行管理。

ClockIn，Corpus，CorpusUser，Memorandum，Reminder，Account，UserInfo

**5．5．3** **Service模块**

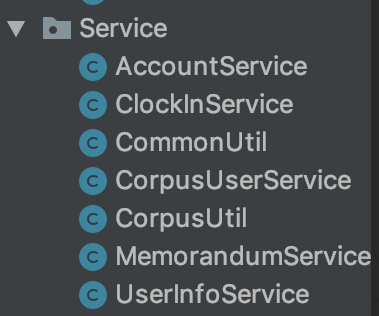


图5.5.3-1: Service模块

Service是主要的核心功能类，对应相应的Table和Controller，主要实现业务逻辑上的功能，比如AccountService类，AccountService类引入了slf4j.LoggerFactory服务端日志，方便检错，同时@Resource RedisUtil 告诉Spring容器需要这个bean类。

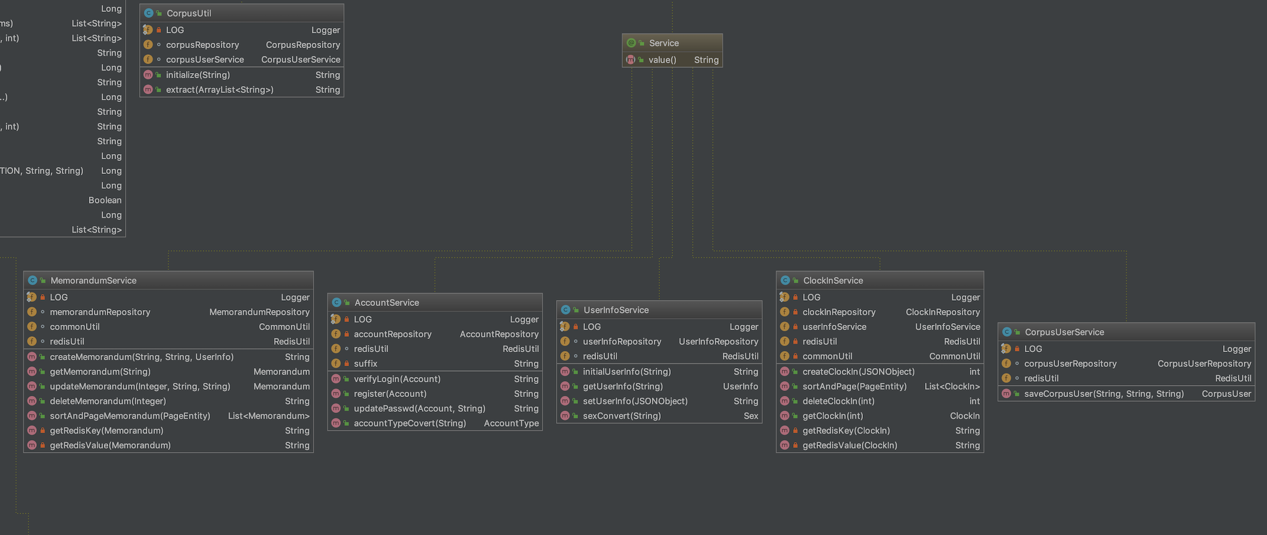


图5.5.3-2: Service UML图

这里引入了Redis，主要功能就是在注册时，将用户的account存入缓存中，并设置过期时间，当需要验证用户的登陆状态时，只需要以用户名称作key，到Redis中去取，进而实现单点登陆，同时以用户的username新建一个list类的redis对象，用来放用户的来聊天数据

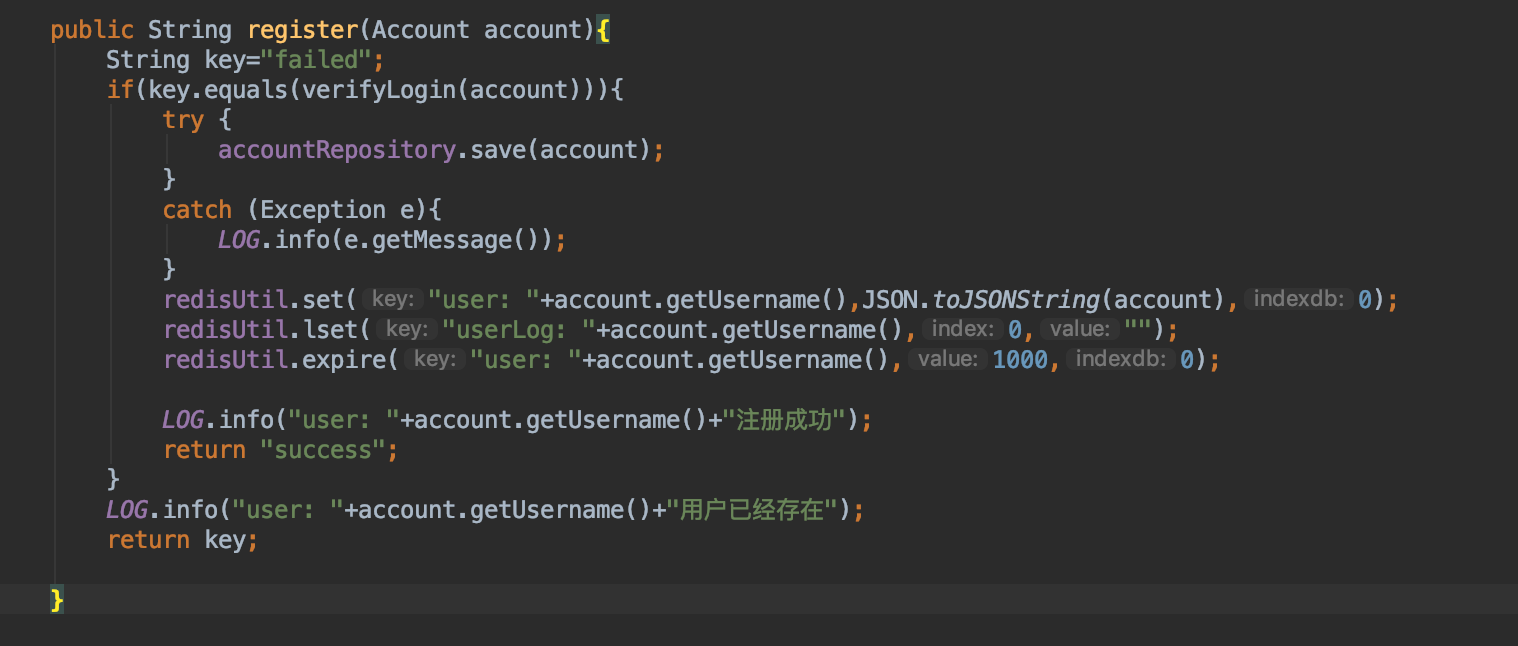


图5.5.3-3: Register方法举例

用户每创建一个Memorandum，ClockIn或者Reminder，就会从Redis中通过用户名取出该UserLog，并将新创建对象的主键存进去，redisUtil.rpush在队列尾添加当前Memorandum的id。

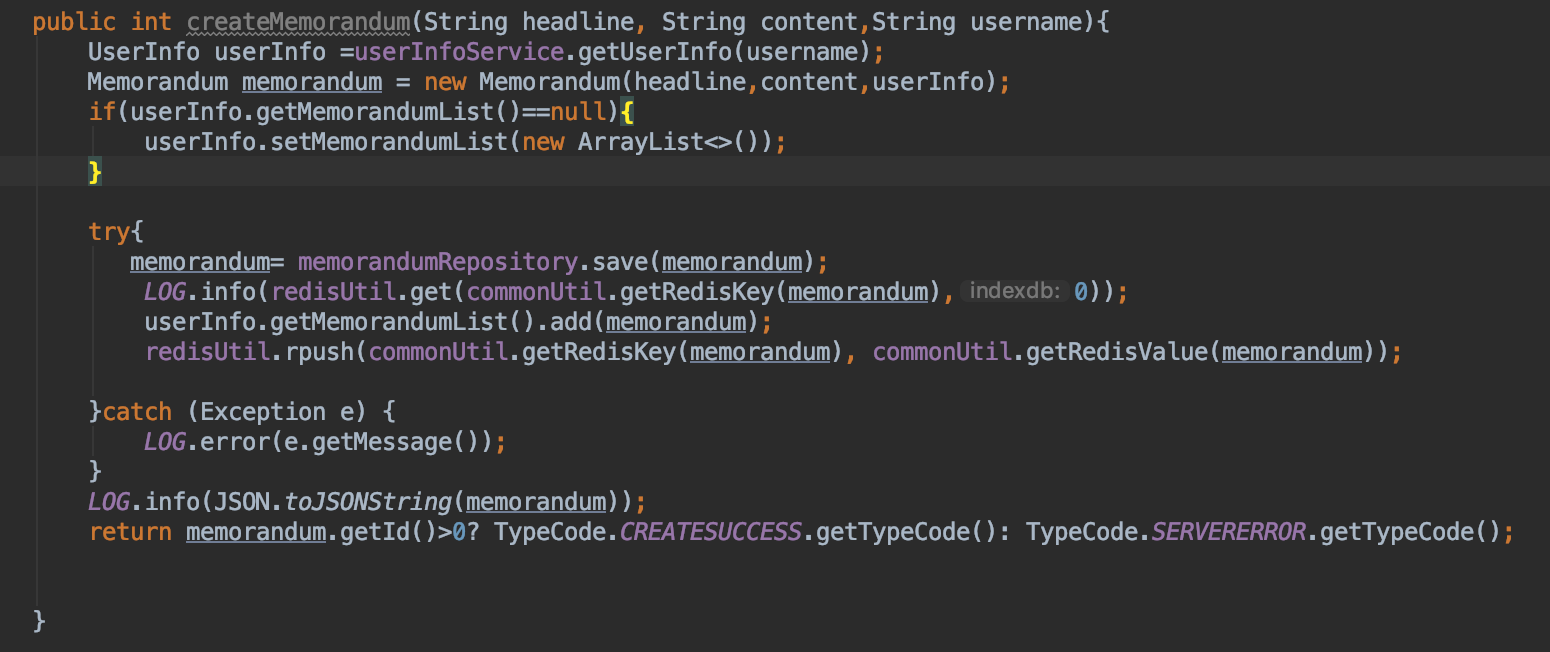


图5.5.3-3: 创建Memorandum

这里Spring Bean设置的是单例的，但是单例也可以有很好的并发性能，比如redis，所以用Spring用threadlocal来解决线程安全问题，每一个线程都保留一个资源的副本，这样就减少了创建销毁资源的时间，同时也解决了竞争共享资源的问题，现在人人都有了，比如每一个threadlocal都保存一个。

由于Redis也是单线程的，所有的请求都遵循FIFO原则，所以也不会存在线程安全问题。



图5.5.3-4: 获取聊天记录

并在用户调用查询以往聊天记录的时候，调用getCacheChat函数，通过用户名和当前页号，从UserLog中把记录（JsonObject格式）取出来，并根据键的值判断到底属于哪一种记录类型，然后到相应的数据库中查找出来，将内容封装在JSONString里返回至前端页面。

**5．5．4** **配置模块**

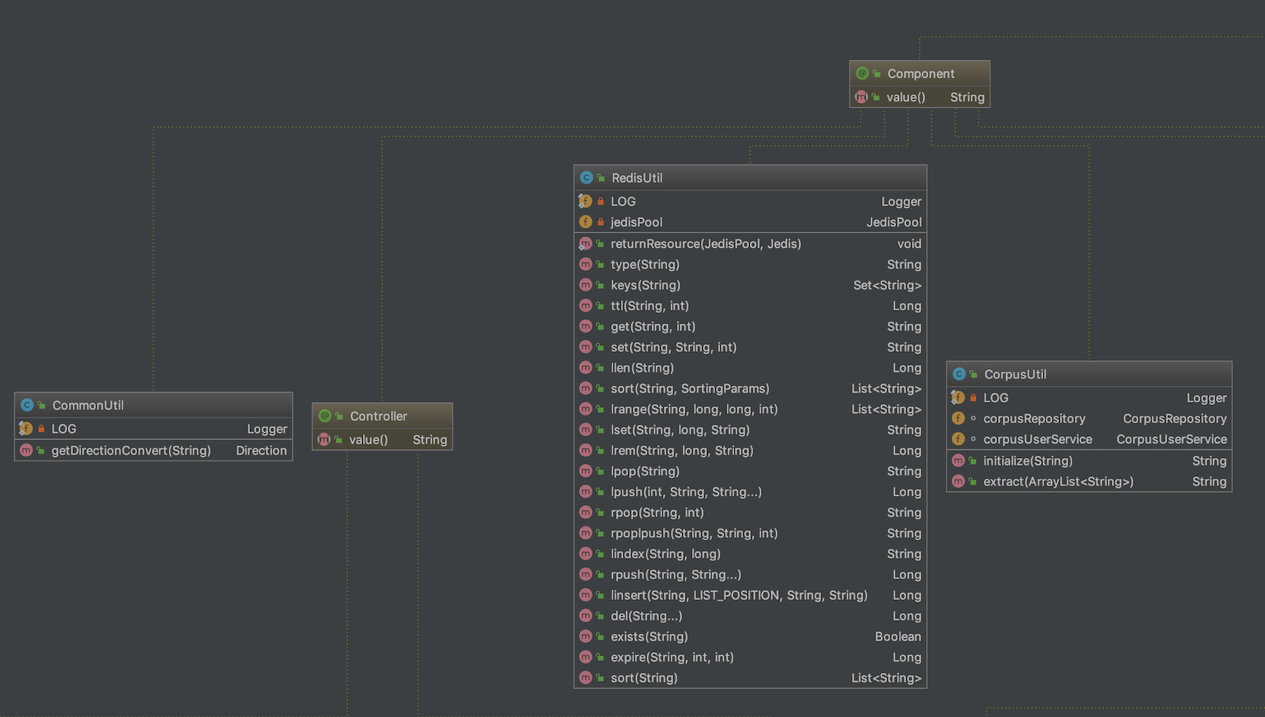
****

图5.5.4-1: 所有Util类

所有Util类都打上了Component注解表明，这些类不是Service也不是Controller，但是依旧希望Spring 容器进行托管，那么这时候就需要给这些类打上Controller的父注解Component表明这些类虽然无法细分，但是依旧是不可或缺的组件。

比如RedisConfig这里，就是将Jedis进行二次封装，形成一个个的接口

该模块用于配置登陆注册所用的redis接口和配置信息。

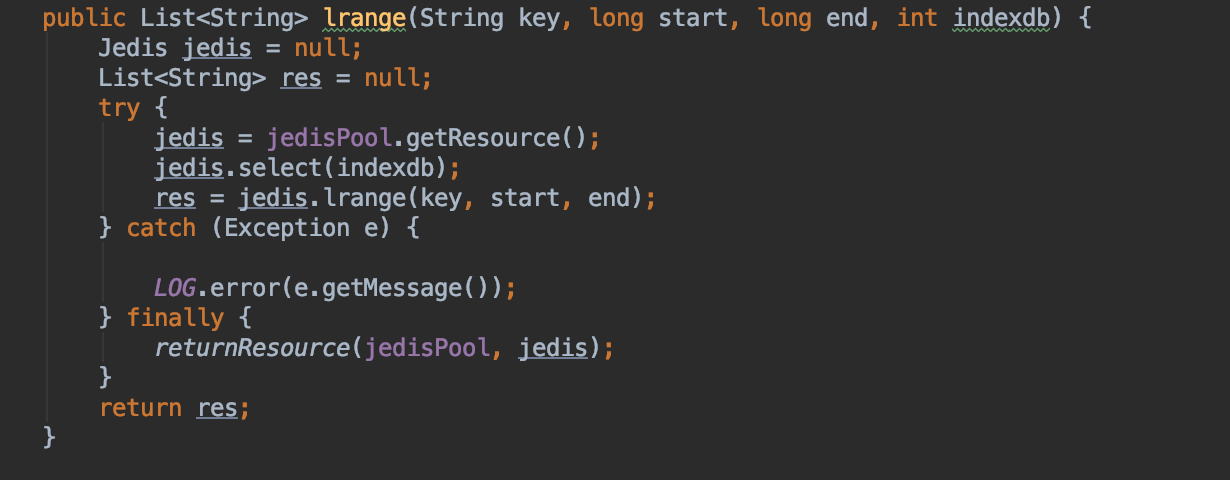


图5.5.4-2: RedisUtil中封装的函数

**5．5．5** **Controller模块**

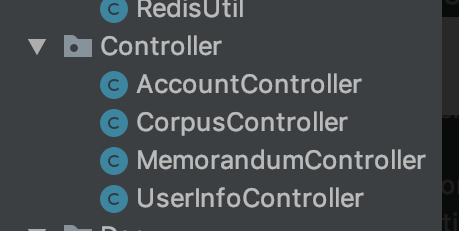


图5.5.5-1: 所有Controller类

截止到文档提交前已经写了4个Controller类，用户管理的AccountController和UserInfoController，和备忘录创建删除查询相关的MemorandumController，还有和抽取语料，查询聊天记录相关的CorpusController。

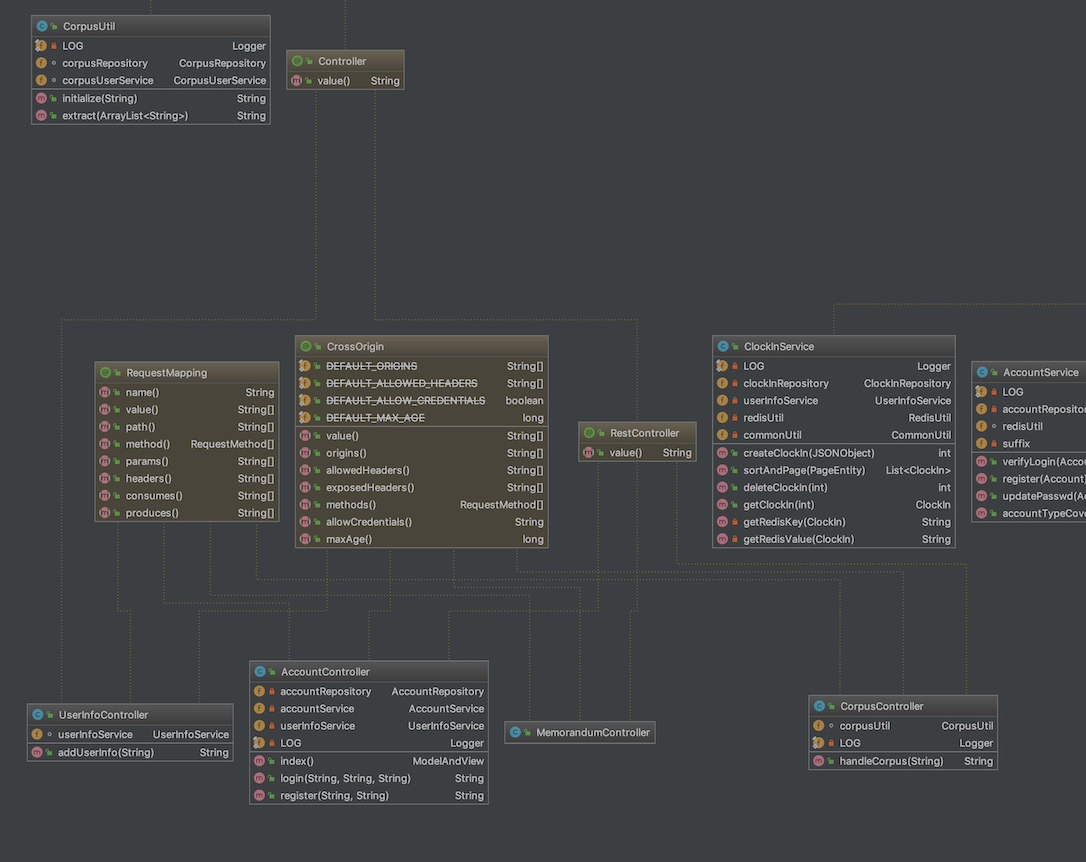


图5.5.5-1: Controller类UML结构

上面就是Controller类的UML图，可以看到，@Controller注解也是继承自@Component

**5.6 安卓端设计与实现**

**5.7 Web官网设计与实现**

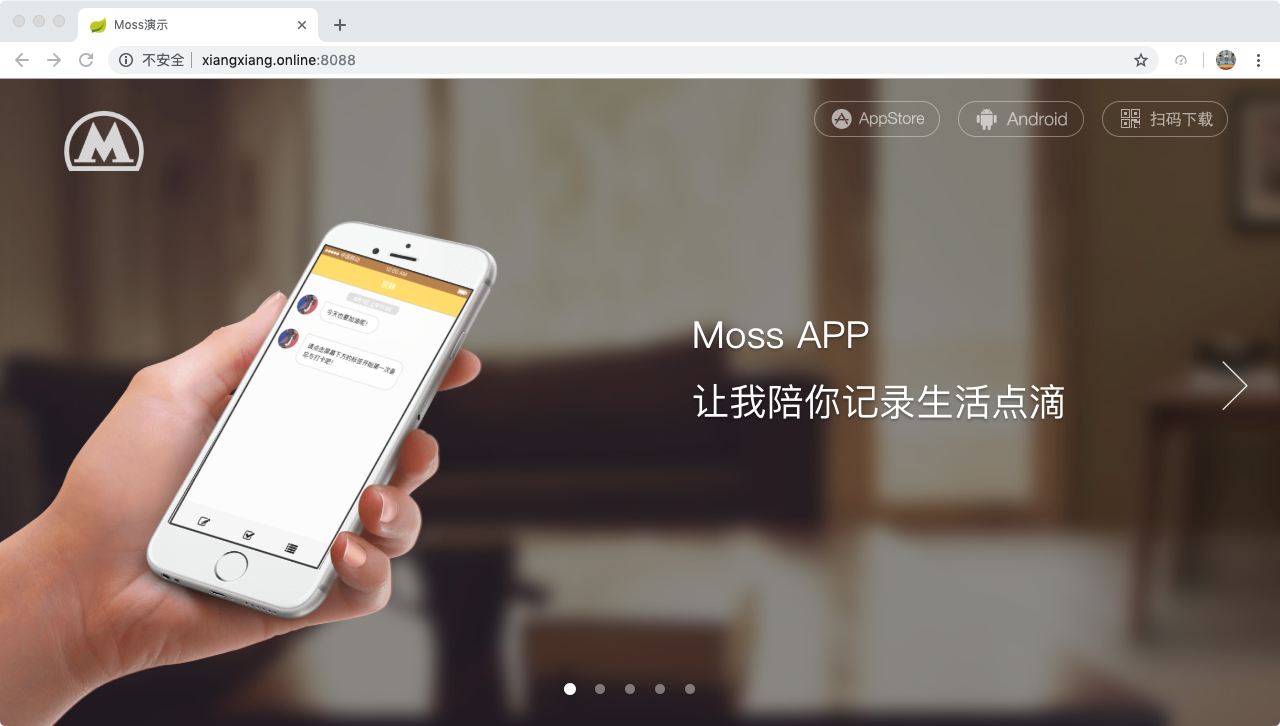
****

图5.7-1: Moss官网

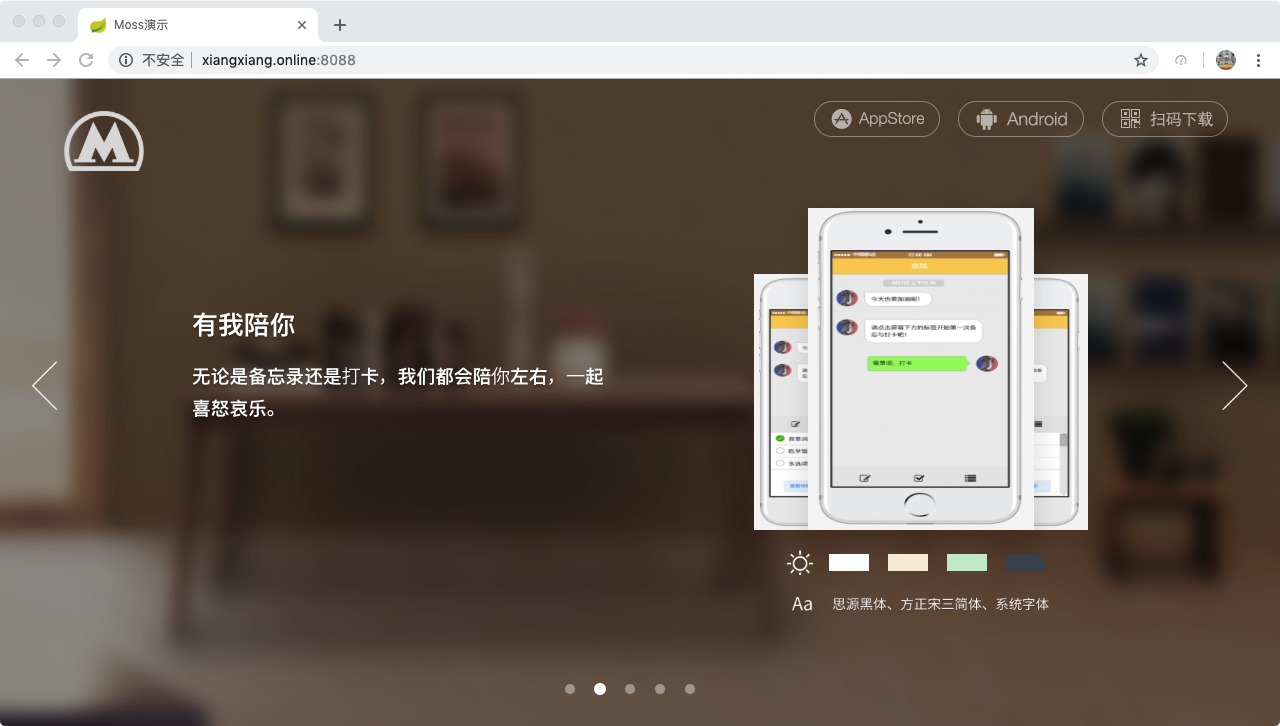


图5.7-2: 陪伴式鼓励

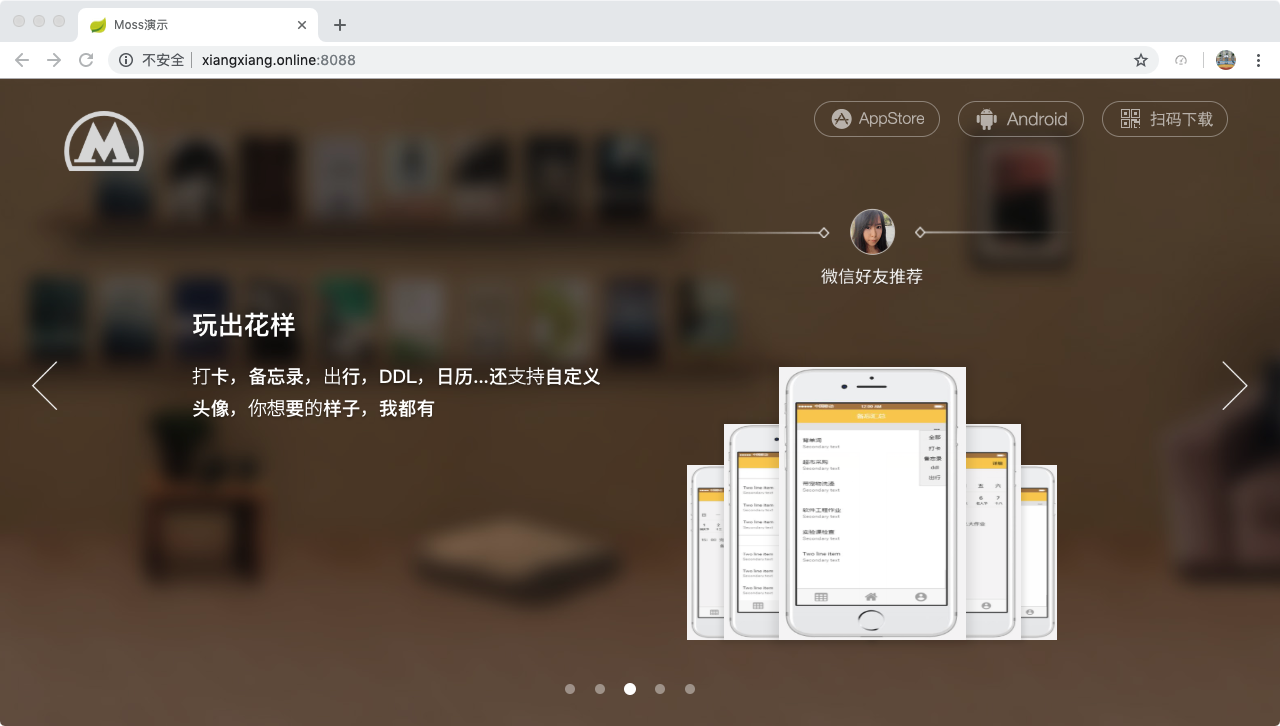


图5.7-3: 功能展示

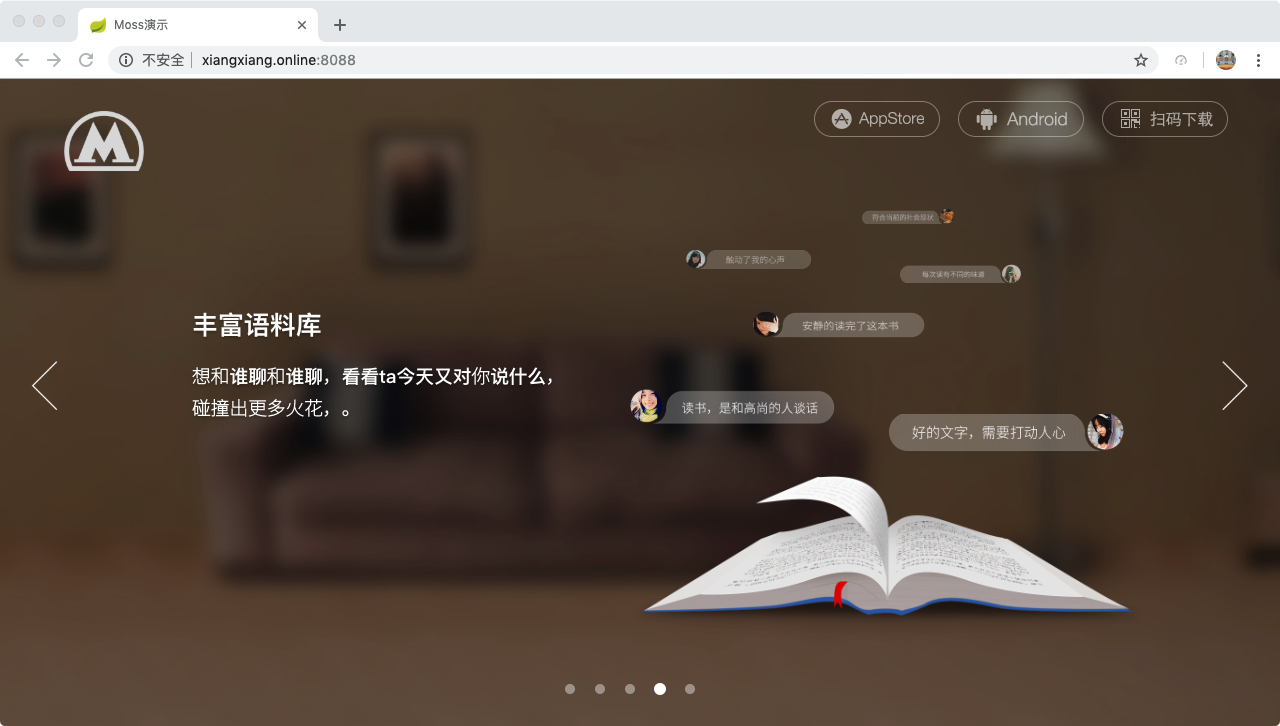


图5.7-4: 丰富语料库



图5.7-5: 好友互动

**6测试文档**

**6.1 测试概要**

**6.1.1测试目的**

本测试文档为Moss项目的测试文档，目的在于测试Moss项目中在安卓手机客户端是否达到预定需求，以及对后台的压力测试。

**6.1.2 测试环境**

**安卓端**：

安卓版本：MIUI 10.2.25.0

手机配置：最高2.84GHz 骁龙855 八核 6GB运行内存

**6.2 测试流程图**

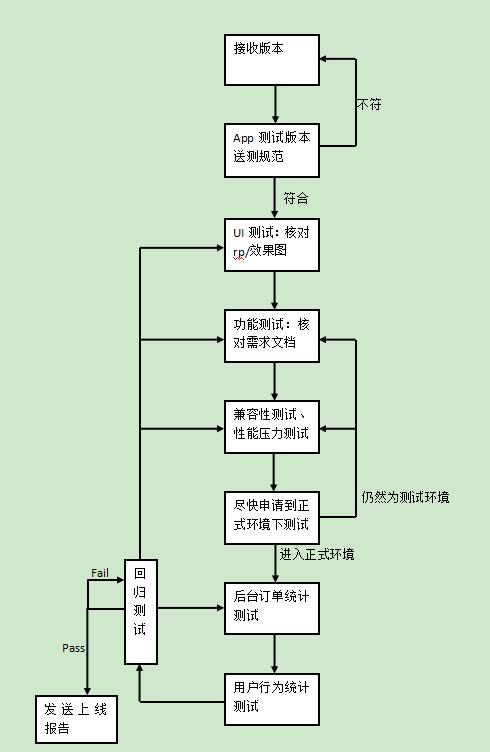


图6.2-1: 测试流程图

**6.3非功能性测试**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试内容 | 测试流程、方法 | 预期结果 | 测试情况 |
| 硬件连接 | 手机开机前检查手机元件是否齐全、有无异常，各硬件之间连接线路是否完好无损； | 无异常，手机元件齐全、无异常，连接线路完好无损； | 通过 |
| 网络连接 | 连接网络观察手机是否能打开其他网页查询、下载数据； | 手机可以打开其他网页并且查询下载数据； | 通过 |
| 硬件测试 | 检查手机是否具有启动系统、正常使用的能力； | 手机可以成功打开系统，跳至登录界面； | 通过 |

**6.4功能性测试**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试内容 | 测试流程、方法 | 预期结果 | 测试情况 |
| 基本UI测试 | 核对档中的UI效果图， | 页面齐全，在各色尺寸的设备上无兼容问题，单指，多指触碰响应区无异常， | 通过 |
| 用户注册登陆功能 | 检查系统是否能够成功注册、登录，如果出现用户名、密码不符合的情况会不会报错并不允许登录。 | 系统能够成功注册、登录，如果出现用户名、密码不符合的情况会及时报错并不允许用户登录。测试通过。 | 通过 |
| 备忘录板块检测 | 检查系统能否正确的创建，更新，删除备忘录，打卡，提醒事项等板块；并且能够在相应页面正确展示备忘录 | 系统能够正确的增删改，展示备忘录 | 通过 |

|  |  |  |  |
| --- | --- | --- | --- |
| 聊天板块检测 | 系统能否根据用户创建的备忘录类型和用户的完整状态，从语料库中抽取返回语句 |  |  |

**7项目管理**

**6.1项目规划表**

项目的进度规划表如下：

|  |  |
| --- | --- |
| 日期 | 规划进度 |
| 3.17-3.23 | 第一阶段，需求阶段，该阶段任务已圆满完成。 1.产品上，对产品 APP 整体功能体系进行再次规划。 参见本策划书项目产品模块，着重在产品重定义。 2.研发上，对整个软件架构规划进一步明晰，明确学习任务及分工，多线程并行。 参见完整架构设计板块。 |
| 3.24-3.30 | 第二阶段，编码筹备阶段，该阶段任务接近完成。 1.储备学习Android相关技术，整理安卓技术学习笔记。 2.做完Android的所有UI设计工作 2.开始编写Web官网 |
| 3.31-4.6 | 第三阶段，研发阶段，1. 确定分工，开始讨论技术路线和框架使用，初步搭建前端和后台的项目架构 |
| 4.7-4.13 | 第三阶段，研发阶段。1.完成应用的登陆和注册部分，搭建储存用户数据的数据库，完成登陆，注册组件的界面的代码实现。 |
| 4.14-4.20 | 第三阶段，研发阶段。1.语料库的建立。 完成语料库数据库架构，确定数据表属性和其连接关系。2. 全组收集语料信息，收集短语并加以分类，导入数据库。3. 测试语料库。 |
| 4.21-4.27 | 第三阶段，研发阶段。1.实现备忘录功能。即备忘录的创建、查看、编辑、删除等功能的接口和具体实现。2. 完成备忘录的相关界面和前端实现 。 3. 完成聊天功能的界面和前端部分 |
| 4.28-5.6 | 第三阶段，研发阶段。1.完成打卡事件功能的实现。即打卡时间的创建，提醒，修改，标记完成和删除功能。2.基本完成打卡相关界面的创建 3. 完成聊天功能的实现，包括对话创建，语料库反馈，聊天记录查询等 |
| 5.7-5.13 | 第三阶段，研发阶段。1. 完成个人中心功能，即用户的个人信息设置，聊天角色设置，密码设置，个性化设置（更改头像，壁纸，配色等）2. 优化ui界面交互和细节，增加美观性 |
| 5.14-5.16 | 第四阶段，测试阶段。1. 测试软件的各个功能的实现情况，用户友好程度，容错率。2. 优化反馈机制，第一次使用提示等辅助功能 3. 完善补充开发设计文档 |

**6.2实际进度表**

实际进度和完成情况的反馈和总结如下：

|  |  |
| --- | --- |
| 日期 | 完成情况 |
| 3.17-3.23 | 第一阶段，需求阶段，该阶段任务已圆满完成。 |
| 3.24-3.30 | 第二阶段，编码筹备阶段，该阶段任务已圆满完成。 |
| 3.31-4.6 | 第三阶段，研发阶段。初步完成框架搭建，在项目搭建的同时进一步完善之前的文档和设计工作 |
| 4.7-4.13 | 第三阶段，研发阶段，通过学习和研究，确定采用redis完成登录注册功能。完成了前端的界面和实现。 |
| 4.14-4.20 | 语料库建立完成，语料收集初步完成。因为聊天功能为实现所以没有进行完全测试。同时语料库内语料仍需继续收集 |
| 4.21-4.27 | 1. 初步完成备忘录功能的实现和接口 2. 基本实现备忘录界面 3. 备忘录相关功能的前后端连接未如期完成 4. 完成语料库语料收集 5. 聊天界面尚未完成 |
| 4.28-5.6 | 1. 初步完成打卡功能的实现和接口 2. 集成备忘录和打卡界面，细节适配尚存在一些问题 3. 打卡功能的前后端 连接未如期完成（发现接口不够规范，出现问题） 4. 聊天功能的界面完成，实现代码尚未全部完成。 |
| 5.7-5.13 | 1. 个人中心的实现，接口和界面初步完成。 2. 尝试进行项目整体的前后端的调用和测试，初步完成，仍需调试和优化 3. 聊天功能的主体已经完成，小部分仍存在问题未解决。 |
| 5.14-5.16 | 1. 完成开发设计文档的完善工作，完成项目架构和界面，模块等方面的更新。 2. 调试和测试软件 |

**6.3最终成员分工**

|  |  |  |
| --- | --- | --- |
| **姓名** | **项目角色** | **完成情况** |
| 程浩宇 | 项目组长，总负责人，后端程序员；项目管理方面，组织规划产品定义、设计、编码和进度计划。其中主要负责数据库模块设计与编码，负责处理层以及接口层设计、编码和文档工作 |  |
| 杜会远 |  |  |
| 朱昱睿 |  |  |
|  |  |  |

**7 团队成员分工、感想**

**8 参考文献**

[1]美)沙赫.软件工程：面向对象和传统的方法[M].机械工业出版社:北京,2007:35.

[2]米格尔•格林贝格.Web Development Based On Flask[M].人民邮电出版社:北京,2014:18.

[3]陶辉.Understanding Nginx[M].机械工业出版社:北京,2016-2:05.

**教师评语评分**

评语：

评分：

评阅人： 年 月 日

（备注：对该实验报告给予优点和不足的评价，并给出百分制评分。）