西安交通大學
XI'AN JIAOTONG UNIVERSITY

关于“你知”——问答系统 APP 开发项目的软件设计说明

团队自评分：

姓名	学号	班级	自评分
温中镇	2171411500	计算机 74	90
马凯	2176314126	计算机 74	90
敬羽戈	2175112030	计算机 74	90
陈宏宇	2175112019	计算机 74	90
李玥	2176122622	物联网 71	90

报告发布日期：2020/4/19

目录

1、引言

1.1、标识

文件状态：终稿

当前版本：2.1

1.2、概述

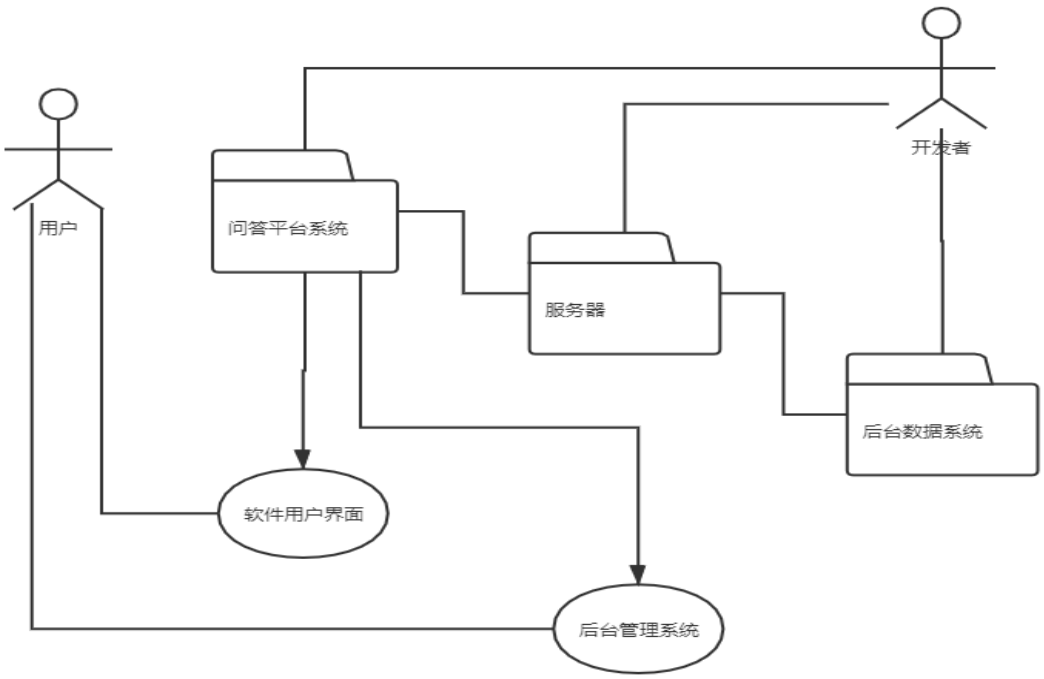
名称：“你知”——问答系统 APP

用途：服务于在校大学生的问答平台

功能：平台用户注册账户、注销账户、修改账户信息等基本账户功能，用户提出问题、回答问题、查询问题、筛选信息等问答平台功能，用户修改问题、删除问题、删除答案、修改答案、查询答案等基于基本用例的功能。

性能：待测试

上下文关系：



开发者：温中镇、马凯、敬羽戈、陈宏宇、李玥

保密性：后台数据保密、用户信息保密

2、引用文件

无

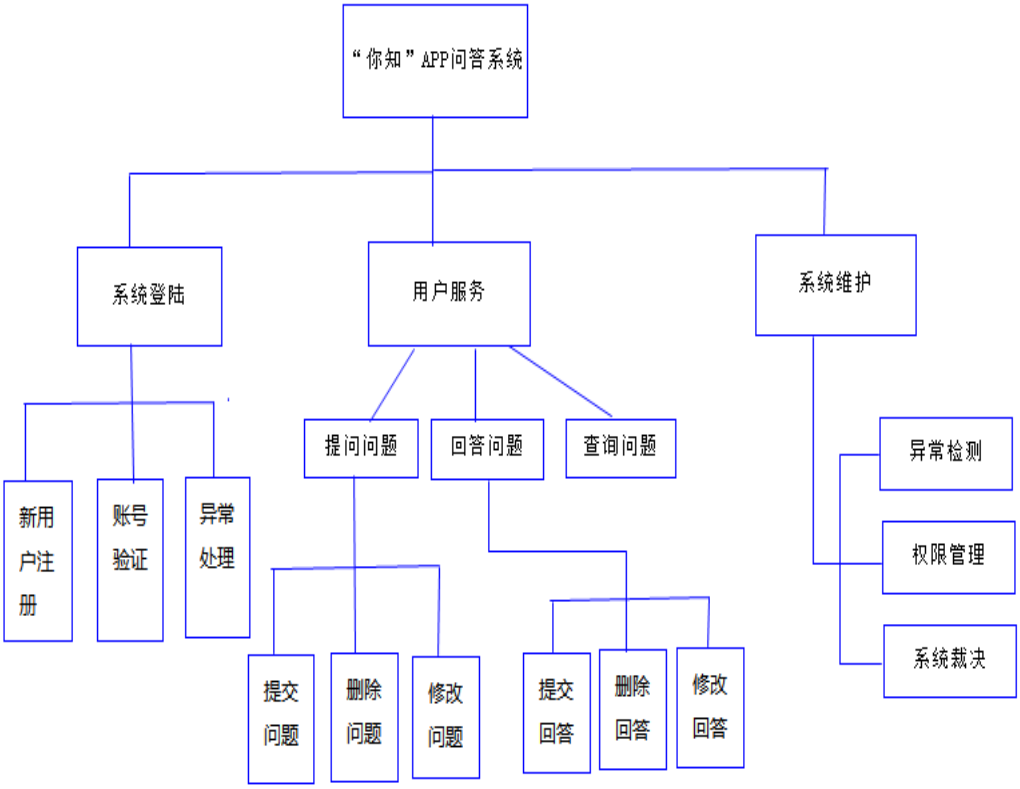
3、概要设计

3.1 总体结构

3.1.1 需求概述

该项目目标在于开发一款基于问答系统的APP，能成功实现一个问答平台基本的功能，通过提问，回答和检索的问答形式为用户提供高质量的信息，做到页面精简，功能清晰，精准搜索。实现平台用户注册账户、注销账户、修改账户信息等基本账户功能，用户提出问题、回答问题、查询问题、筛选信息等问答平台功能，用户修改问题、删除问题、删除答案、修改答案、查询答案等基于基本用例的功能。

3.1.2 软件结构概述



系统结构概述图

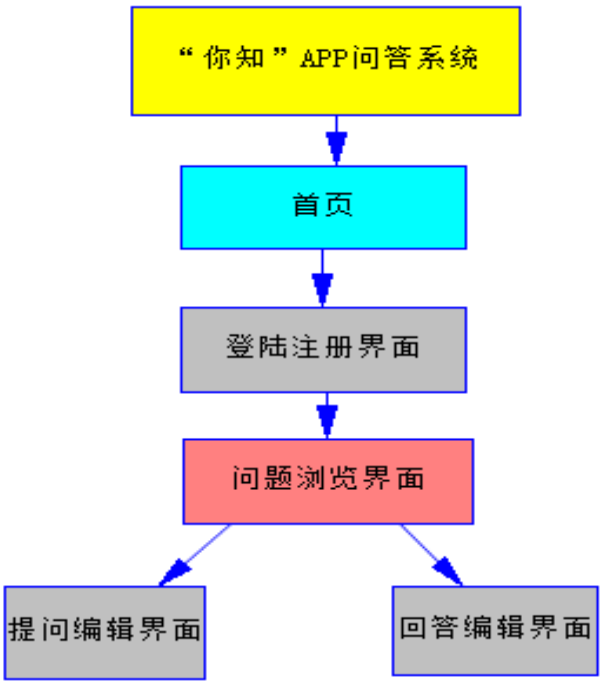
3.2 系统接口设计

3.2.1 用户界面设计

根据系统的具体功能，页面结构分为首页，登陆注册，浏览，编辑页面四个大模块，其中问题浏览界面是重点模块，包括问题内容，问题查看，内容结构等。

界面中文字体选用 Microsoft Yahei，西文字体选用 San Francisco；字体大小根据组件和页面的布局选择适宜浏览的大小。

系统整体选取暗系色彩风格，营造一种简约，规整的实用感，网页的背景色选用 16 进制色号#2c343c；其他组件配色多接近暗色色系。

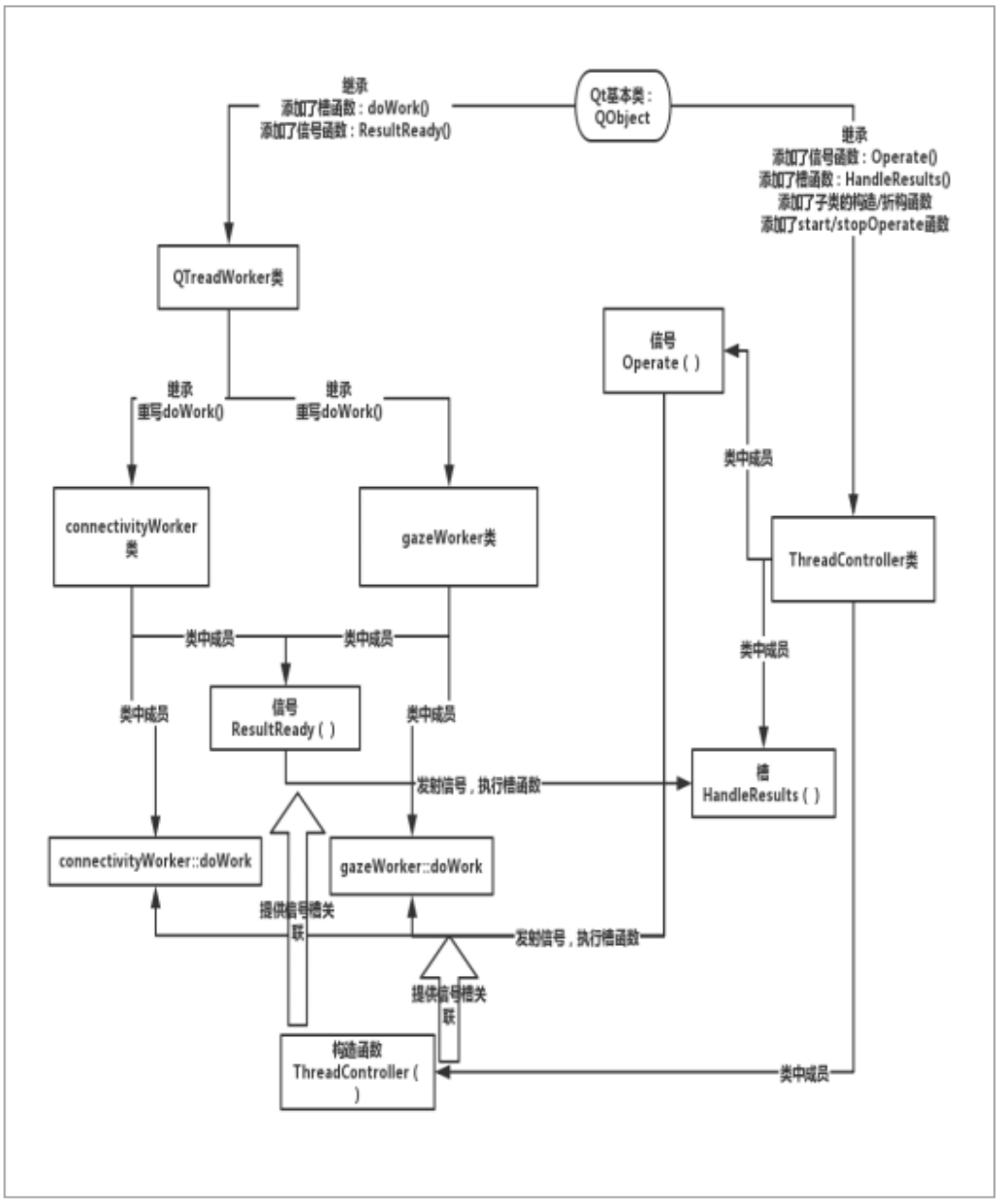


系统页面结构图

首页效果图（借鉴知乎）

3.2.2 子系统接口设计

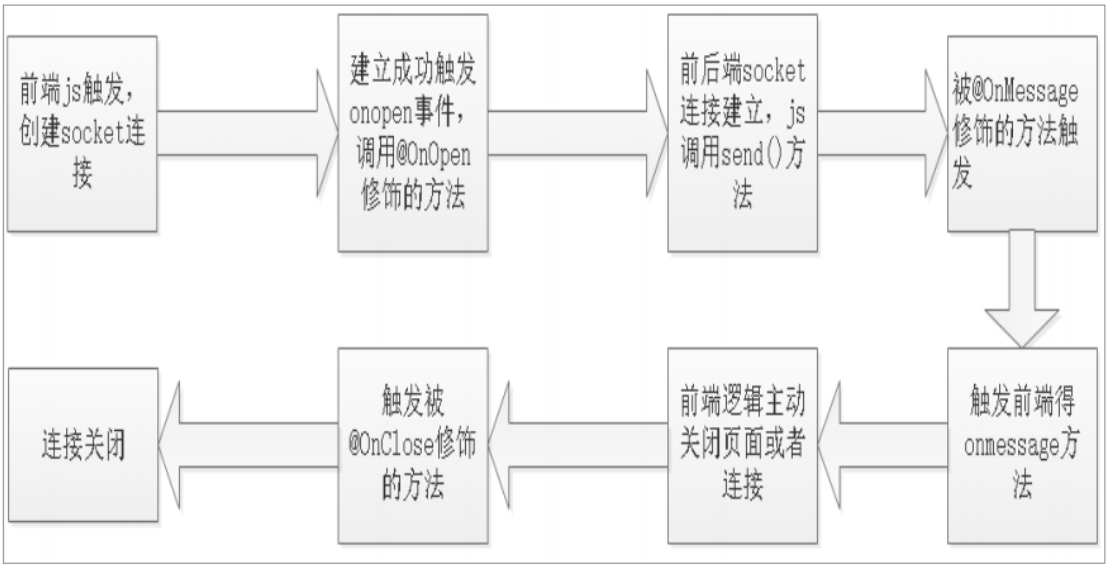
在设计子系统和接口的时候，因为客户程序与抽象类的实现部分之间存在着很大的依赖性。所以我们引入 facade 将这个子系统与客户以及其他的子系统分离，提高子系统的独立性和可移植性。当构建一个层次结构的子系统时，使用 facade 模式定义子系统中每层的入口点。对于相互依赖的子系统，可以让它们仅通过 facade 进行通讯，从而简化了它们 之间的依赖关系。



子系统重要类之间的通信

3.2.3 通信接口设计

在对于整个 web 交互来说，ws 成为 html5 的标准协议。对于前端来说，创建连接的方式变得简单，只需要 new 出来 websocket 的对象即可。在创建完对象之后，合理使用四种触动方法，就能完成实时通信，对于后端来说，道理是相通的，同样主要依靠四种触动方法来书写逻辑，首先我们必不可少的是需要显式添加配类，使得 springboot 支持 ws 的路由，然后由下图我们可以了解连接建立的基本通信流程。

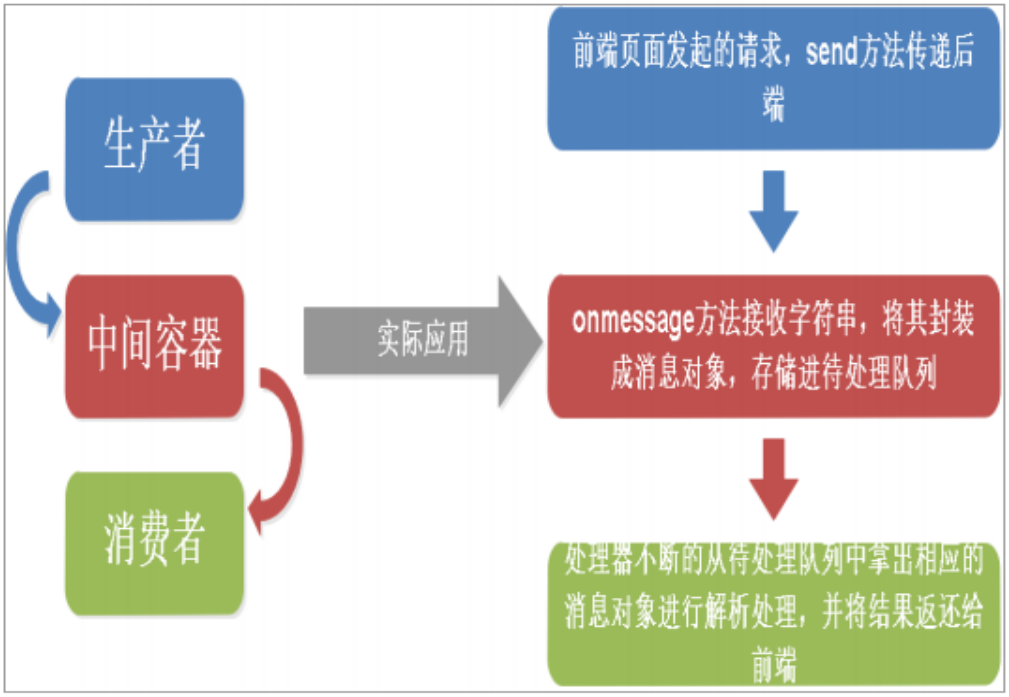


websocket 一般连接建立流程

在书写通信接口的时候，我们并没有去向往常一样 new 实例化出来，而是使用了@Autowired 进行了自动装配，虽然自动装配的过程与实例化的过程是基本相同的，但是主动权是不一样的，我们主动去 new 实例化了对象，而自动装配是 spring 实例化了对象，那么什么时候销毁，什么时候具体实例化，则是由 spring 去控制的。这就是我们常说的控制反转，将控制权交由 spring 去控制，对于同一工程体系下的类来说，被 spring 注解方法或者类，去 new 或调用一个普通类，是可以行的通的。反过来，在一个普通类中去调用一个 spring 托管的方法或者类,是行不通的,会报空指针错误。@OnOpen，@OnMessage，@OnClose，@OnError，作为原生 java 包里面的注解，很容易给人一种误会，当有消息来的时候，我们只能在方法里面使用普通的类或方法，不能调用有关 spring 托管的类或方法。

接着，我们回顾了生产者与消费者模型，是通过一个容器来解决生产者和消费者的强耦合问题。通俗的讲,就是生产者在不断的生产，消费者也在不断的消费，可是消费者消费的产品是生产者生产的，这就必然存在一个中间容器，其实就是一个阻塞队列，生产者生产的产品不直接给消费者消费，而是仍给阻塞队列，这个阻塞队列就是来解决生产者消费者的强耦合的。采用中间容器的最大的好处是，可以

避免由原生态注解和 spring 注解不相容的矛盾,只需要选择或者创造好合适的数据结构，问题就会变得十分容易。采用中间容器的方式，也有利于线程之间的相互通信，使得曾经可能需要数据库进行通信的线程之间，提供了性能上的进步。

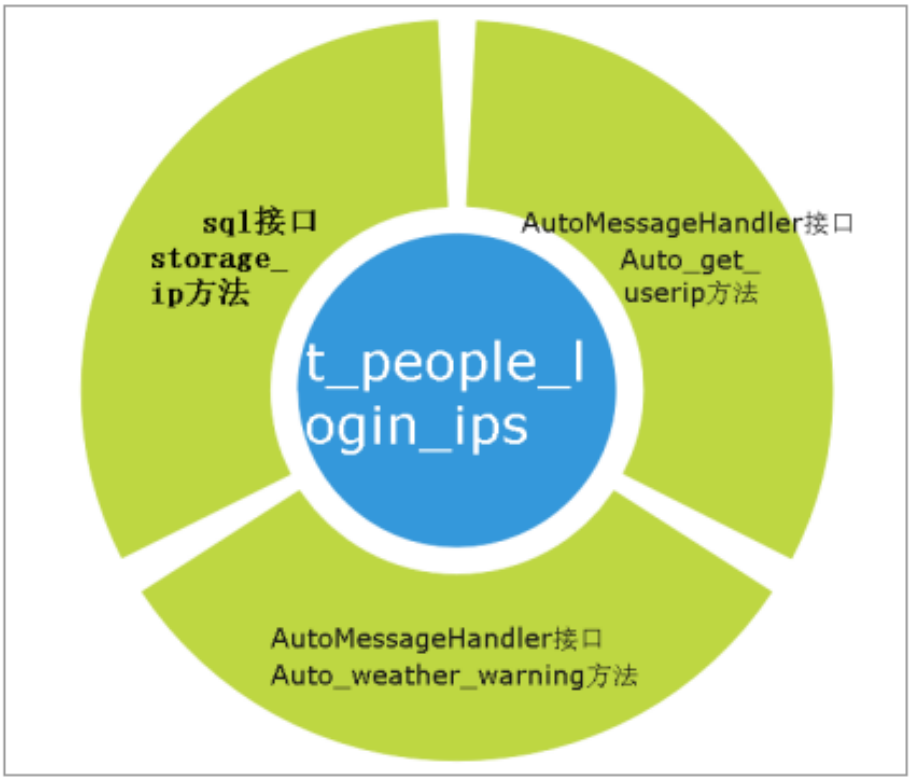


消费者模型实际应用

3.2.4 数据库接口设计

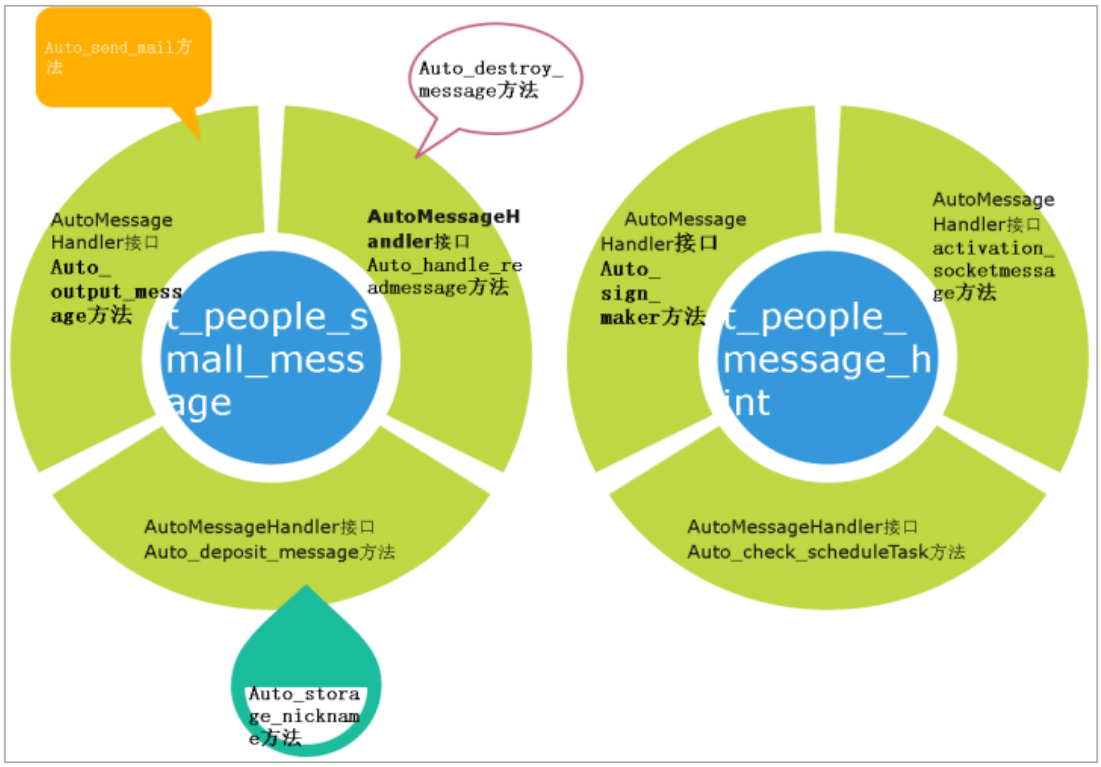
t_people_right_static 与内部接口耦合图

t_people_right_static 作为用户基础属性数据表，负责管理用户的基础权限，如用户名、密码、昵称、头像。通过耦合图可以看出来，此表主要与 Repository 层关联紧密，与业务层有关联较少，确保了一定程度上的分层。业务层调用 DAO 层的接口，即可实现相应功能。对此表只能进行查询、更新和插入的操作，无删除操作，作为改变可能极少的表，无冗余属性，在后端的数据库中扮演着地基的角色。



t_people_login_ips 与内部接口耦合图

从耦合图中我们可以看出，login_ips 只是在系统中实现一个特定组件的特定功能。IP 数据永久存储在数据库，短暂存储在 log 文件，可用于用户行为的聚类分析，以及异常检测。



t_people_small_message,t_people_message_hint 与内部接口耦合图

message_hint 数据表。当用户每一次使用 cookie 登入功能界面时，会触发激活，与其相关的消息制造器将会开始工作。当消息制造器制造完毕之后，触发数据库将其权限关闭，而由消息制造器触发之后产生的消息，则会存储在 small_message 里面。消息发送器，则会使用 small_message 里面的数据，它记录的数据，是有关定时任务的消息，用来向用户展示。在耦合程度上，small_message 与 socket 实时通信系统耦合紧密，与其他系统无耦合。message_hint 与网关层系统的登入逻辑有耦合，与实时通信系统有耦合，与其他系统无关。

3.3 全局数据说明

3.3.1 常量和变量

变量及函数命名：采用小驼峰式命名法，英文名具有实际含义。

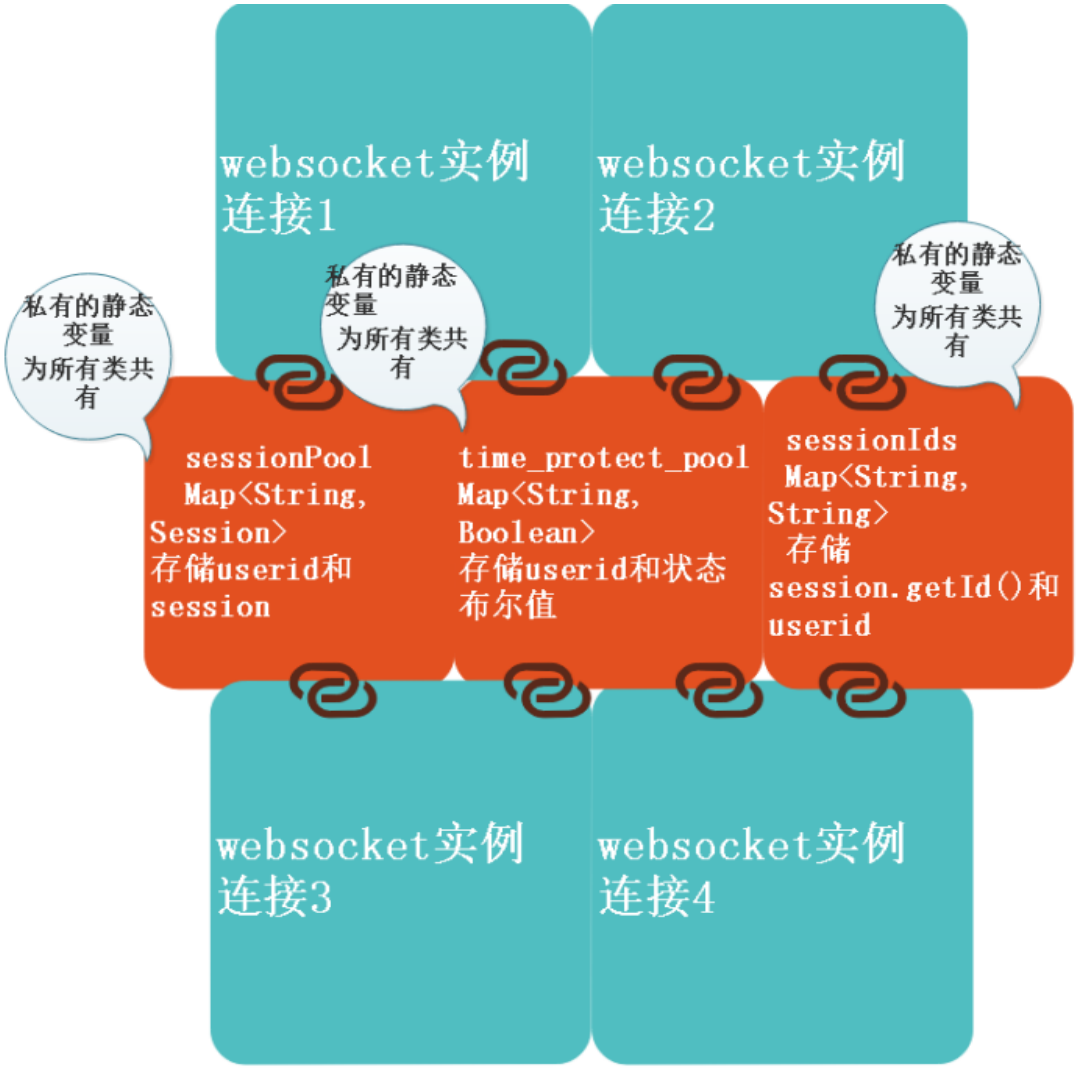
部分常量和变量说明：

参数	名称	类型	说明
常量	response_index1=1	int	问题内容标识符
常量	response_return=1	int	返回标识符
变量	source_id	int	问题索引id
变量	chapter_id	int	页面id
变量	title	str	标题
变量	content	str	问题正文
变量	comment	str	回答评论
变量	req_time_stamp	int	相应请求时间戳
变量	target	int	关系目标问题索引

3.3.2 数据结构

a. ConcurrentHashMap 的使用

使用线程安全型哈希 map，较高的稳定于多线程环境，在连接建立的时候，将 session 和 cookie 放入 sessionpool 池子中保存，将 cookie 和连接号放入 sessionids 池子中保存，在连接断开的时候，将 cookie 和 true 放到时间保护池子中保存，书写静态方法，便于其他普通类或者 spring 托管的方法 websocket 进行监控或操作。

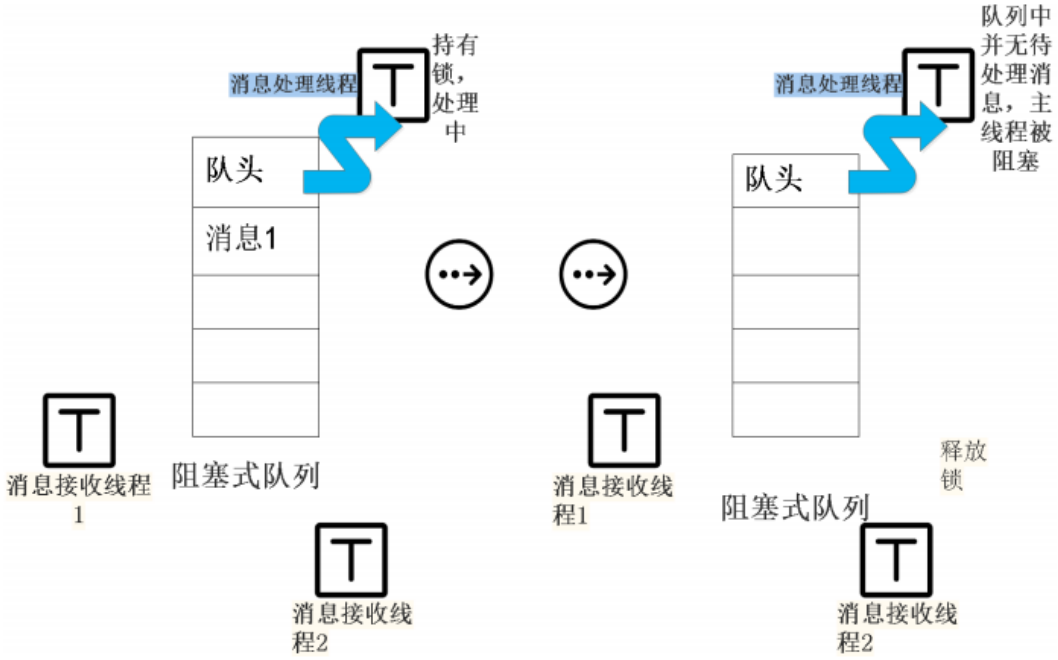


websocket 主要管理池

b.泛型阻塞队列的设计

使用 java 原生的关键字 synchronized 和 wait()方法和 notify()方法，初次的考虑是此队列可能只为一种业务服务，后来将其改成泛型数据队列，使其支持多种数据的阻塞队列，之后广泛用于本系统，为各种定时器之间的通信提供了方便，系统整体性能得到了提高。独立出来一个线程，不停的去处理前端发来的请求，我们把它定义成，消息处理器。设计一个中间容器让消息处理器，和 onmessage 方式怎么去通信，只不过这个容器我们要考虑到多线程的操作安全性，对于读来说，对于线程均安全，一个对象允许多个线程去读取，但是不允许多个线程对其进行写操作，那么这个容器我们设定的更加严格，每次只允许一种操作进行，所以我们给 pop 和 push 两种方法加上

了互斥锁，并且合理利用阻塞，当整个队列为空的时候，我们使其wait()，这样，消息处理器就会阻塞住,当有新的消息来时，消息处理器再次工作。

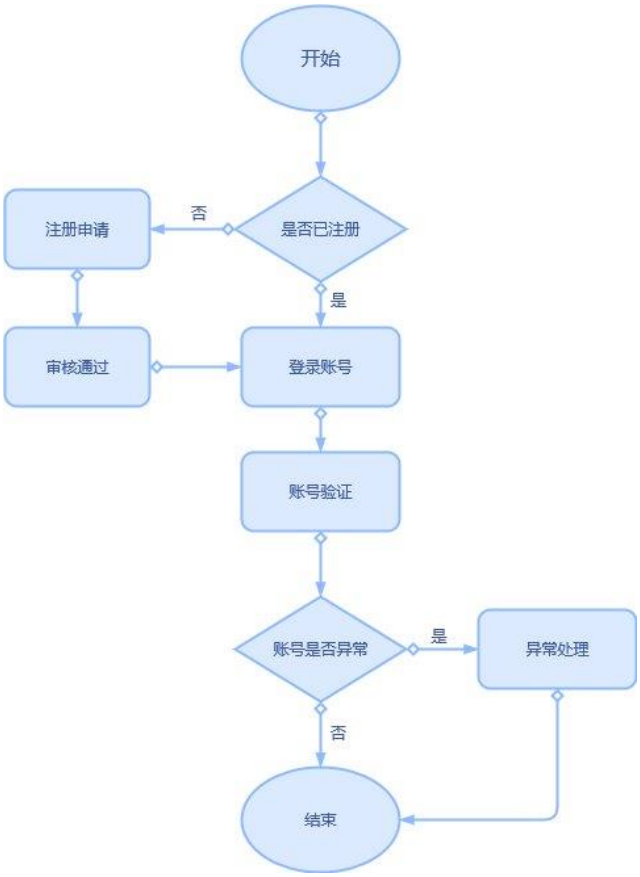


处理过程

3.4 部件（子系统）

3.4.1 登陆管理系统

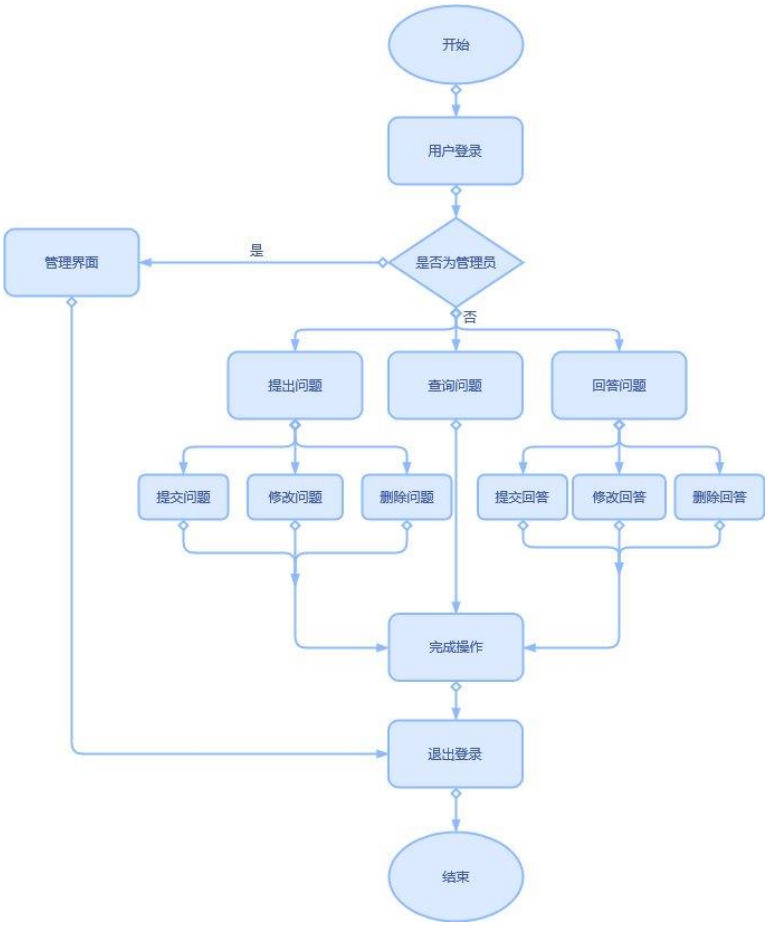
登陆管理系统主要实现用户注册登陆，账号验证和异常处理的功能。新用户通过登陆管理系统进行注册；老用户则直接进行登陆，系统对其账号进行验证并确认是否有异常。其具体流程如下图：



登录管理系统

3.4.2 问答系统

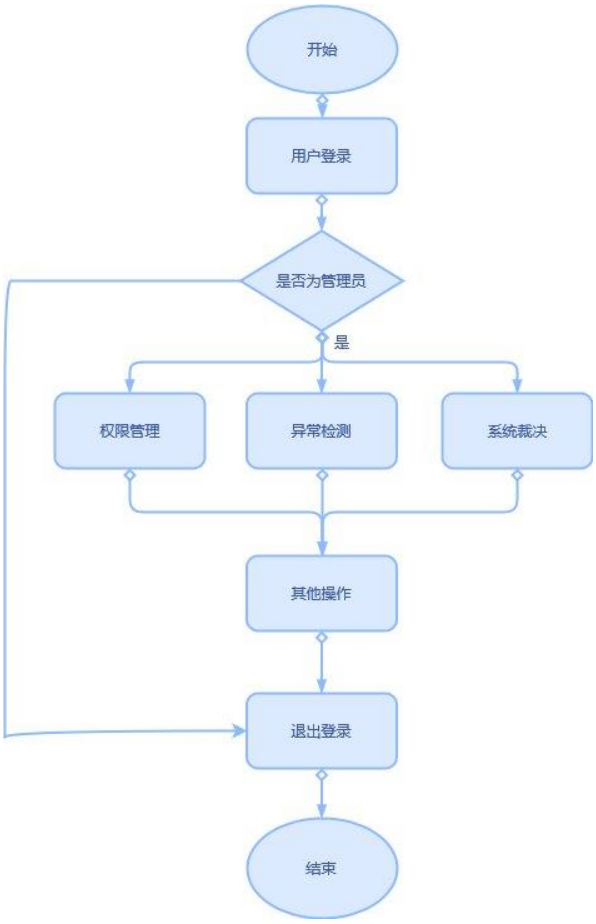
问答系统主要提供给用户实现实时问答交互，包括：提出问题，回答问题和查询问题三个主要功能。对于提出问题和回答问题，用户均可对提问或者回答进行查询，删除和修改，其具体流程如下图：



问答系统

3.4.3 维护系统

该子系统主要供软件管理员对软件进行维护、问题筛查等操作，包括异常检测，权限管理和系统裁决三个方面，其具体流程如下：



维护系统

3.5 运行行为

“你知”APP 在服务器端开多线程，将主要的提问功能、回答功能通过并行设计接受请求，使用线程安全型 hashmap 以确保实时交互的正确实现。线程安全型 hashmap 有较高的稳定于多线程环境，使用其便于其他普通类或者 spring 托管的方法 websocket 进行监控或操作。

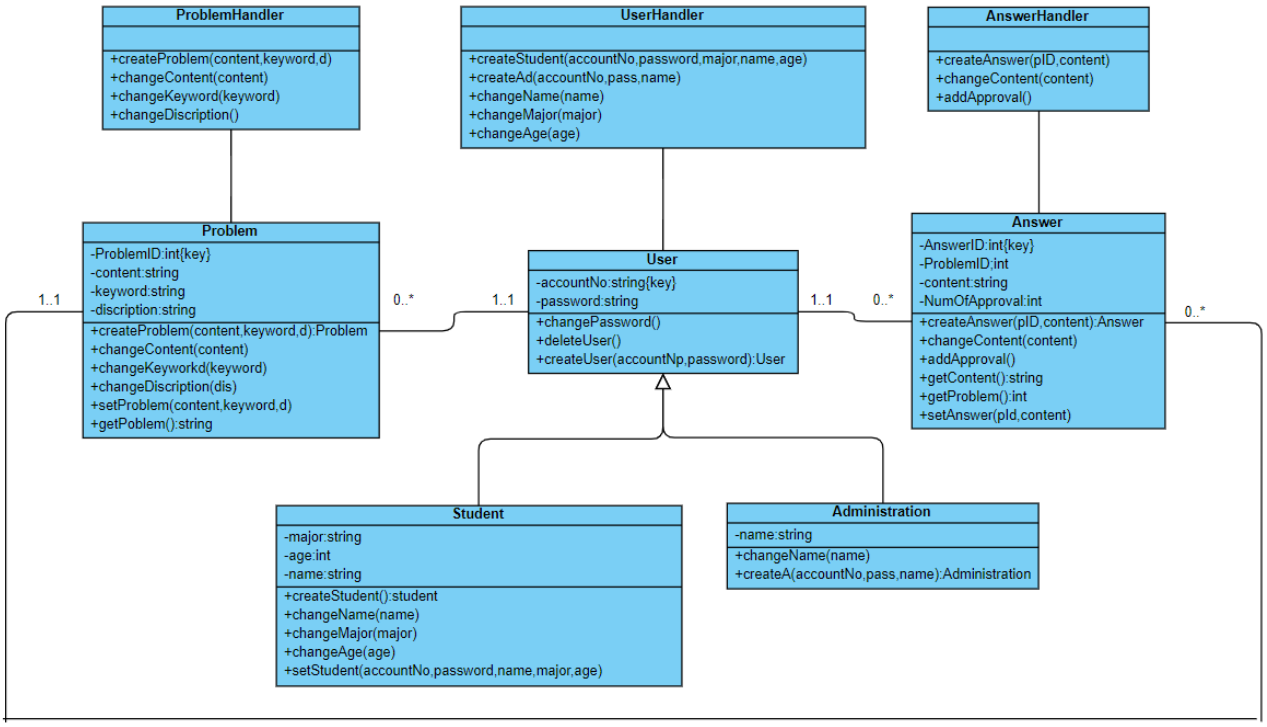
同时，我们也采用了泛型阻塞队列，提高了系统的整体性能。泛型阻塞队列的优势在于能支持多种数据，为各种定时器之间的通信提供了方便，也大大提高了容错率。为了确保用户请求读写操作时不出现拥堵，我们给 pop 和 push 两种方法加上了互斥锁，并且合理利用阻塞，当整个队列为空的时候，我们使其 wait()，这样，消息处理器就会阻塞住，当有新的消息来时，消息处理器再次工作。对于系统的负载问题，我们有专门的动态分配算法完成负载均衡，降低出错概率。系统通过用户的响应反馈进行状态转换，将提问（或者回答）

实时提供给用户，确保用户体验，异常控制则通过操作系统和硬件进行实现。

4、详细设计

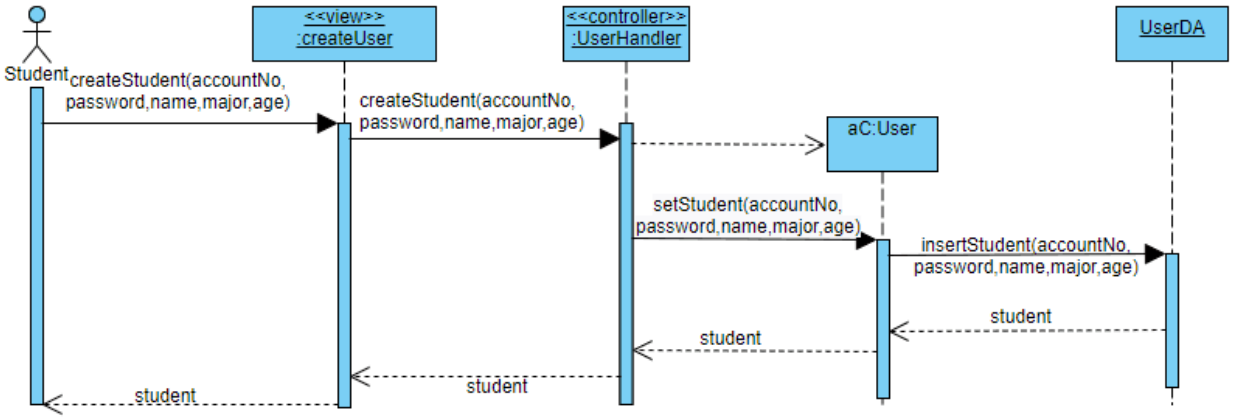
4.1 系统类图

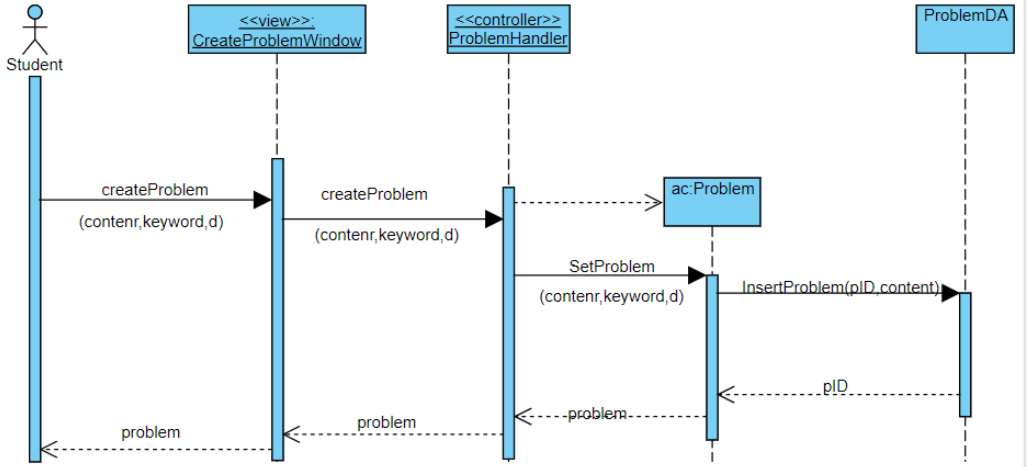
首先，我们设计了整个系统的类图，不仅表现了每个类的属性，方法，而且表示了类之间的关系。

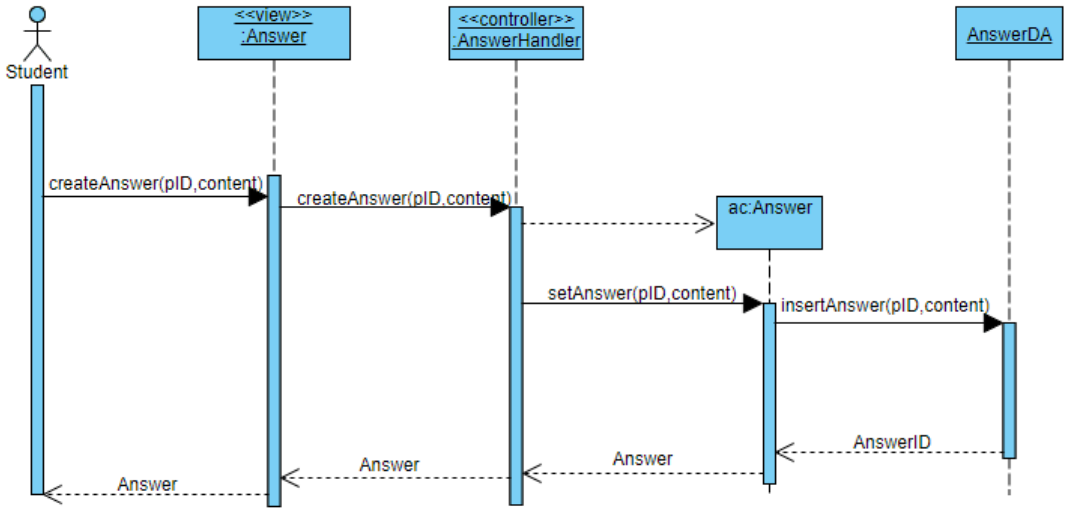


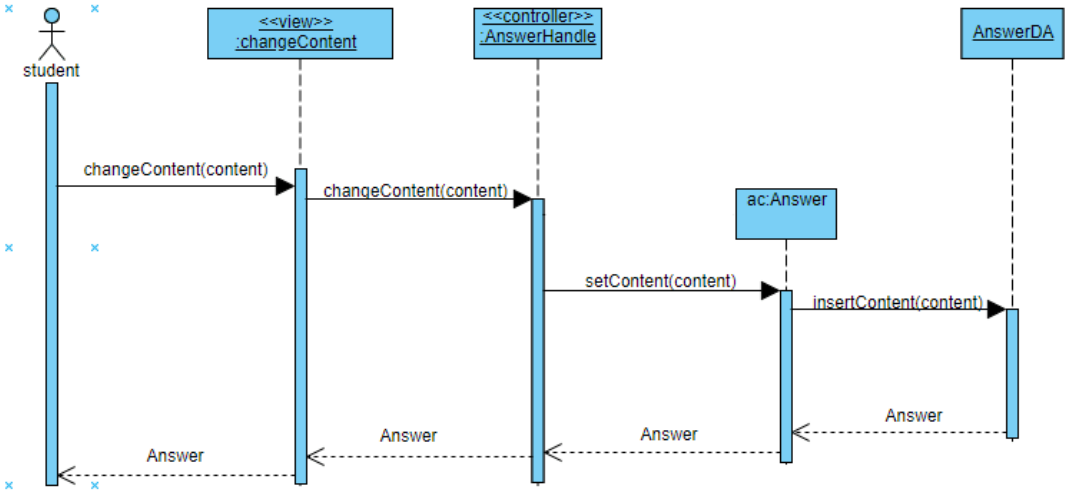
4.2 用例顺序图

我们设计了创建账户，添加问题，添加回答，修改回答四个用例的顺序图。


Student
<<view>>
:createUser
<<controller>>
:UserHandler
UserDA
aC:User
createStudent(accountNo,
password,name,major,age)
createStudent(accountNo,
password,name,major,age)
setStudent(accountNo,
password,name,major,age)
insertStudent(accountNo,
password,name,major,age)
student
student
student
student


Student
<<view>>:
CreateProblemWindow
<<controller>>
ProblemHandler
ProblemDA
ac:Problem
createProblem
(contenr,keyword,d)
createProblem
(contenr,keyword,d)
SetProblem
(contenr,keyword,d)
InsertProblem(pID,content)
pID
problem
problem
problem
problem


Student
<<view>>
:Answer
<<controller>>
:AnswerHandler
AnswerDA
ac:Answer
createAnswer(pID,content)
createAnswer(pID,content)
setAnswer(pID,content)
insertAnswer(pID,content)
AnswerID
Answer
Answer
Answer



5、需求的优先级

功能需求优先级最高，其中按用例的重要性，将功能需求分为三级：

- 注册账户、提问问题、回答问题（第一级）。
- 查询问题、筛选信息、修改问题、删除问题、删除答案、修改答案、查询答案（第二级）。
- 注销账户、修改账户（第三级）。

非功能需求之中，可靠性的需求最高，其次是可用性，安全和性能。

6、合格性规定

① 为确保上述需求得到满足，拟采用以下合格性验证方法进行保证：

② 演示：运行依赖于可见的功能操作的软件配置,不需要使用仪器、专用测试设备或进行事后分析；

③ 测试：在软硬件及系统设计方面，可通过进行模拟测试进行检验，使用仪器或其他专用测试设备运行系统代码,以便采集数据供事后分析使用；

④ 分析：对从其他合格性方法中获得的积累数据进行处理；

⑤ 同时，应保证此次项目的开发过程及最终成果符合国家相关的法律法规规定。

7、需求可追踪性

需求可追踪性高，可以根据用户新的需求进行软件的更新与优化，并在开发过程中对需求细节进行进一步细化和更改，但保持基本功能点不变。