目 录

# 绪论

## 研究背景与现状

### 标题3

## 国内外现状

## 研究内容

## 论文结构

# 用到的技术

## 标题2

### 标题3

公式按章重新编号：

 （2-1）

公式（2-1）说明，…………（公式在正文中的引用）

图题注：

图2-1 XXXXXX

# 需求分析

## 需求陈述

O2O即Online To Offline，是指将线下的商务机会与互联网结合，让互联网成为线下交易的前台。对本地商家来说，O2O模式要求消费者网站支付，支付信息会成为商家了解消费者购物信息的渠道，方便商家对消费者购买数据的搜集，进而达成精准营销的目的，更好地维护并拓展客户。对消费者而言，O2O提供丰富、全面、及时的商家折扣信息，能够快捷筛选并订购适宜的商品或服务，且价格实惠。当前市面上存在的各种O2O APP如美团、大众点评、糯米等虽然都提供了店铺浏览、在线选单、在线支付、店铺评价等功能，但是仍然缺乏一个良好的信息交流与分析的途径，而本项目就是针对这个问题，为用户提供一个更加便捷的信息分享与交流平台，使得用户可以即时地发表自己对于某次就餐、聚会、旅游、理发等服务的使用体验，也可以在本项目提供的平台上浏览其他用户的个人体验并进行交流，从而获得更好的服务。

系统具体要实现的功能如下：

1. 浏览不同用户发布的帖子：在社区中心浏览所有帖子，也可以通过城市、消费类别、距离等进行筛选，尽可能浏览自己感兴趣的帖子
2. 发布个人体验：用户将自己的就餐体验等各种生活服务体验通过图文的形式发布出来供其他用户参考
3. 与发表体验的人进行交流：用户可以进入帖子详情页面浏览完整的帖子信息，并对自己感兴趣的地方以跟帖或者评论的形式发表看法，不同的用户通过这种形式进行深入地交流
4. 处理消息中心的各项通知：其他用户的评论、回复、点赞、收藏、跟帖等操作都会在消息中心罗列出来，用户可以从消息中心跳转到具体的帖子详情页面查看具体的评论等
5. 在个人中心管理自己发布的帖子：用户在个人中心“我发布的”版块可以浏览自己以往发布的帖子，或者对不合适的帖子进行修改，或者删除自己不再想保留的帖子
6. 在个人中心查看收藏的帖子：用户在个人中心“我收藏的”版块，查看自己曾收藏的感兴趣的帖子

## 用例模型

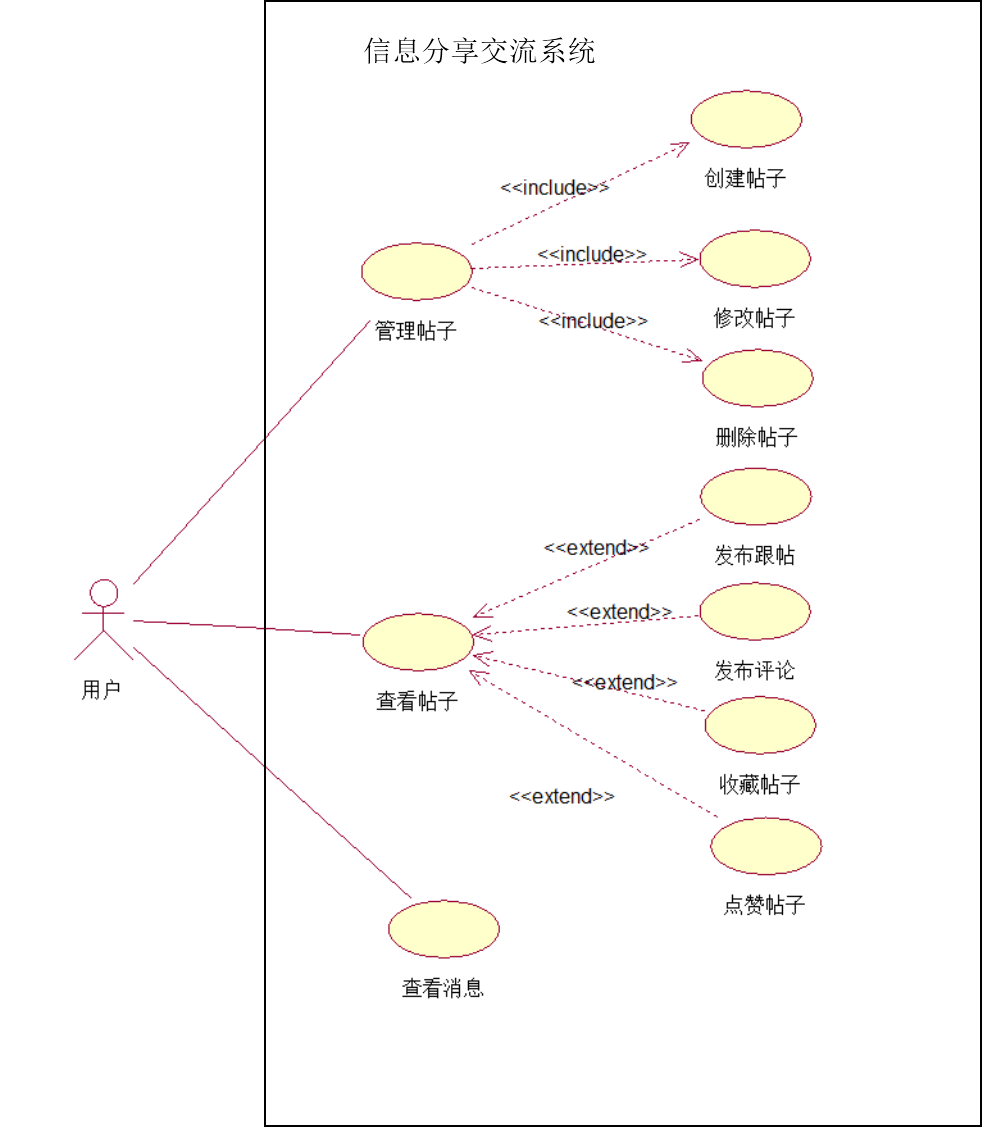


图3-1 系统用例

表1：“创建帖子”用例说明

|  |  |
| --- | --- |
| **用例名称** | 创建帖子 |
| **用例标识号** | uc-post-1 |
| **执行者** | 用户 |
| **描述** | 用户可以通过创建帖子发布自己的某次消费体验 |
| **前置条件** | 无 |
| **基本事件流** | 1. 开始创建帖子 2. 输入帖子标题 3. 输入帖子内容 4. 选择要上传的图片 5. 输入完成，发布帖子 |
| **结束状况** | 系统提示成功创建帖子 |
| **备选事件流** | 1. 开始创建帖子 2. 输入帖子标题 3. 输入完成发布帖子 4. 系统提示用户输入帖子内容 |
| **后置条件** | 无 |
| **说明** | 无 |

表2：“发布评论”用例说明

|  |  |
| --- | --- |
| **用例名称** | 发布评论 |
| **用例标识号** | uc-post-2 |
| **执行者** | 用户 |
| **描述** | 用户可以通过“发布评论”来描述自己对于某个帖子的看法 |
| **前置条件** | 无 |
| **基本事件流** | 1. 进入帖子详情页面 2. 浏览到某一跟帖 3. 针对该跟帖发布评论 |
| **结束状况** | 系统提示成功发布评论 |
| **后置条件** | 无 |
| **说明** | 无 |

表3：“收藏帖子”用例说明

|  |  |
| --- | --- |
| **用例名称** | 收藏帖子 |
| **用例标识号** | uc-post-3 |
| **执行者** | 用户 |
| **描述** | 用户可以通过“收藏帖子”操作收藏自己感兴趣的帖子，在个人中心查看 |
| **前置条件** | 已进入某具体的帖子 |
| **基本事件流** | 1. 进入帖子详情 2. 收藏帖子 |
| **结束状况** | 系统提示成功收藏帖子 |
| **异常事件流** | 1. 进入帖子详情 2. 收藏帖子 3. 无网络，提示用户“网络异常” |
| **后置条件** | 无 |
| **说明** | 无 |

表4：“查看消息”用例说明

|  |  |
| --- | --- |
| **用例名称** | 查看消息 |
| **用例标识号** | uc-post-4 |
| **执行者** | 用户 |
| **描述** | 用户可以通过“查看消息”操作查看本应用的关于评论、回复、跟帖等的通知 |
| **前置条件** | 无 |
| **基本事件流** | 1. 进入“消息中心” 2. 查看关于评论、回复、跟帖等的通知 |
| **结束状况** | 无 |
| **异常事件流** | 1. 进入“消息中心”查看消息 2. 无网络，提示用户“网络异常” |
| **后置条件** | 无 |
| **说明** | 无 |

表5：“管理我发布的帖子”用例说明

|  |  |
| --- | --- |
| **用例名称** | 管理我发布的帖子 |
| **用例标识号** | uc-post-5 |
| **执行者** | 用户 |
| **描述** | 用户可以通过“管理我发布的帖子”操作查看查看、删除或者编辑自己发布过的帖子 |
| **前置条件** | 无 |
| **基本事件流** | 1. 用户进入个人中心； 2. 选择个人中心“我发布的”版块； 3. 浏览发布过的帖子； 4. 点击某个帖子浏览帖子详情 |
| **结束状况** | 无 |
| **备选事件流1** | 1. 用户进入个人中心； 2. 选择个人中心“我发布的”版块； 3. 长按某个帖子，选择“删除”该贴子； 4. 操作成功，提醒用户“成功删除帖子” |
| **备选事件流2** | 1. 用户进入个人中心； 2. 选择个人中心“我发布的”版块； 3. 选择某个帖子进入帖子详情 4. 选择“编辑”帖子，进入帖子编辑界面 5. 修改帖子标题 6. 修改帖子内容 7. 选择要上传的图片 8. 提交对帖子的修改 |
| **结束状况** | 提示用户“成功修改帖子” |
| **异常事件流** | 1. 用户进入个人中心； 2. 选择个人中心“我发布的”版块； 3. 长按某个帖子，选择“删除”该帖子 4. 操作失败，提醒用户“删除帖子失败 |
| **后置条件** | 无 |
| **说明** | 无 |

## 功能模块

### 帖子模块

1. 创建帖子



图-2：“创建帖子”活动图

1. 开始节点：用户开始创建帖子
2. 结束节点：成功创建帖子，帖子上传失败，帖子信息不完整
3. 活动状态：进入创建帖子页面，输入帖子标题，输入帖子内容，上传帖子图片，提交
4. 修改帖子

功能描述：修改帖子的标题或者内容或者图片

1. 删除帖子

功能描述：删除某个帖子

1. 浏览帖子

功能描述：浏览其他用户发布的所有帖子，也可以通过帖子类别、距离等进行筛选

1. 查看帖子详情

功能描述：

1. 获取帖子的详细信息，包括帖子的完整内容、帖子关联的跟帖
2. 可以在帖子详情页面添加跟帖



图-3：“发布评论”活动图

1. 开始节点：进入帖子列表
2. 结束节点：评论发布成功，评论发布失败
3. 活动状态：进入帖子详情页面，浏览某一跟帖，在跟帖下输入评论内容
4. 可以针对某一跟帖发表评论和回复
5. 可以收藏用户感兴趣的帖子



图-3 “收藏帖子”活动图

1. 开始节点：进入帖子列表
2. 结束节点：成功收藏帖子，收藏帖子失败
3. 活动状态：进入帖子详情，收藏帖子

### 消息模块

1. 查看消息列表

功能描述：进入“消息中心”，查看所有属于该用户消息提醒，包括其他用户对该用户的评论、回复、对该用户发布帖子的跟帖

1. 查看消息详情

功能描述：点击”消息中心“的某一项，查看具体的评论、回复或者跟帖内容

### 个人中心

1. 管理用户发布的帖子



图-4：“修改帖子”活动图

1. 开始节点：进入软件
2. 结束节点：帖子信息不完整，帖子修改失败
3. 活动状态：进入个人中心，选择“我的帖子”，选择要修改的帖子，进入帖子修改页面，修改帖子
4. 管理用户收藏的帖子

# 设计阶段

## 概要设计

### 服务器端系统架构

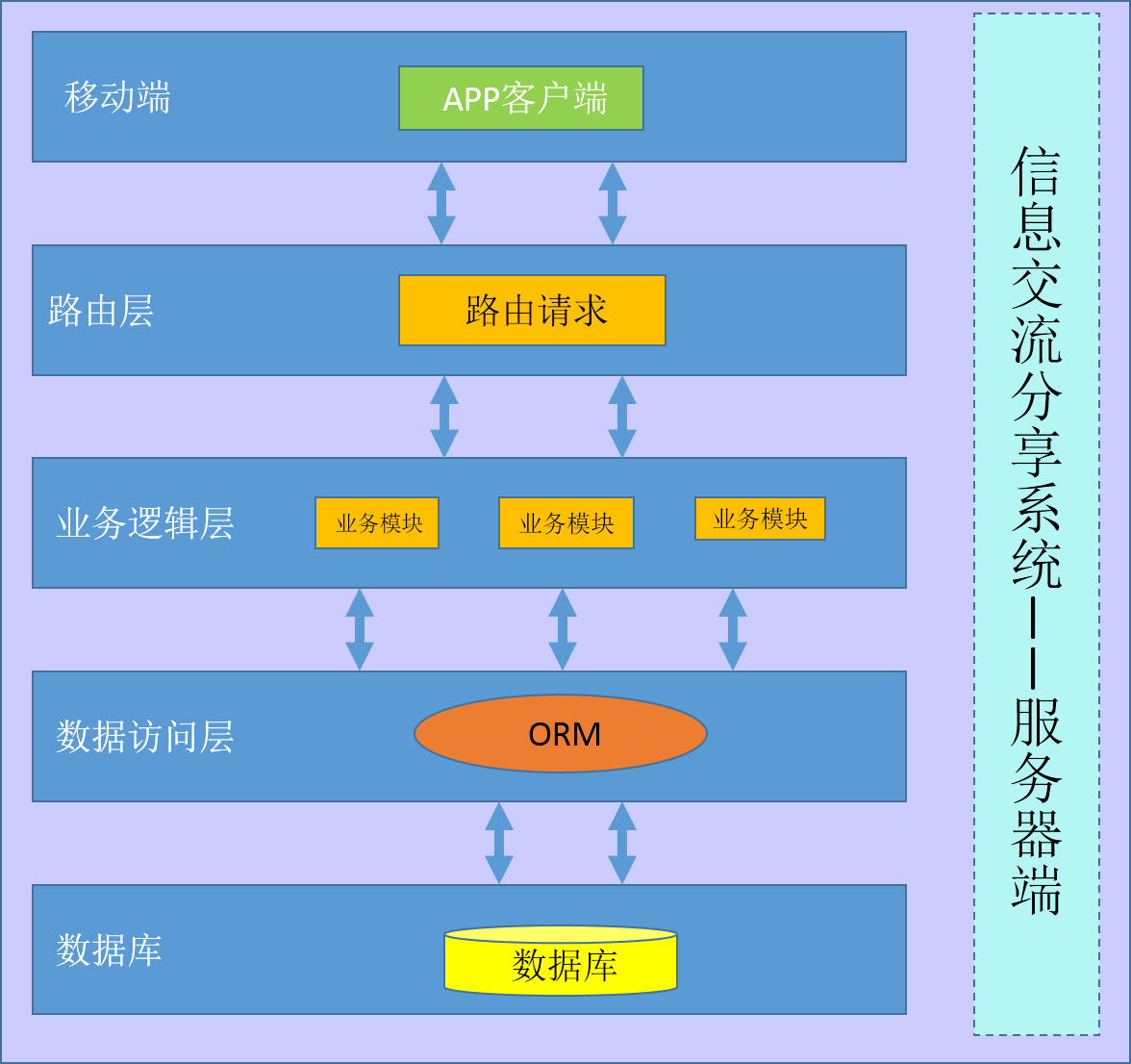


图 -1：系统服务器架构

服务器端系统架构说明：

1. 数据库：保存项目的各项数据；
2. 数据访问层：负责对数据库进行操作，包括对数据库中各张表进行增加、删除、查找、更新等操作；
3. 业务逻辑层：负责具体的业务逻辑，此部分为项目的重点；
4. 路由层：通过对请求路径进行不同的映射，调用可以响应该请求的的业务层的某个方法；
5. 移动端：负责向服务器发出请求，然后使用服务器的响应使用户完成各项操作

### 客户端系统架构

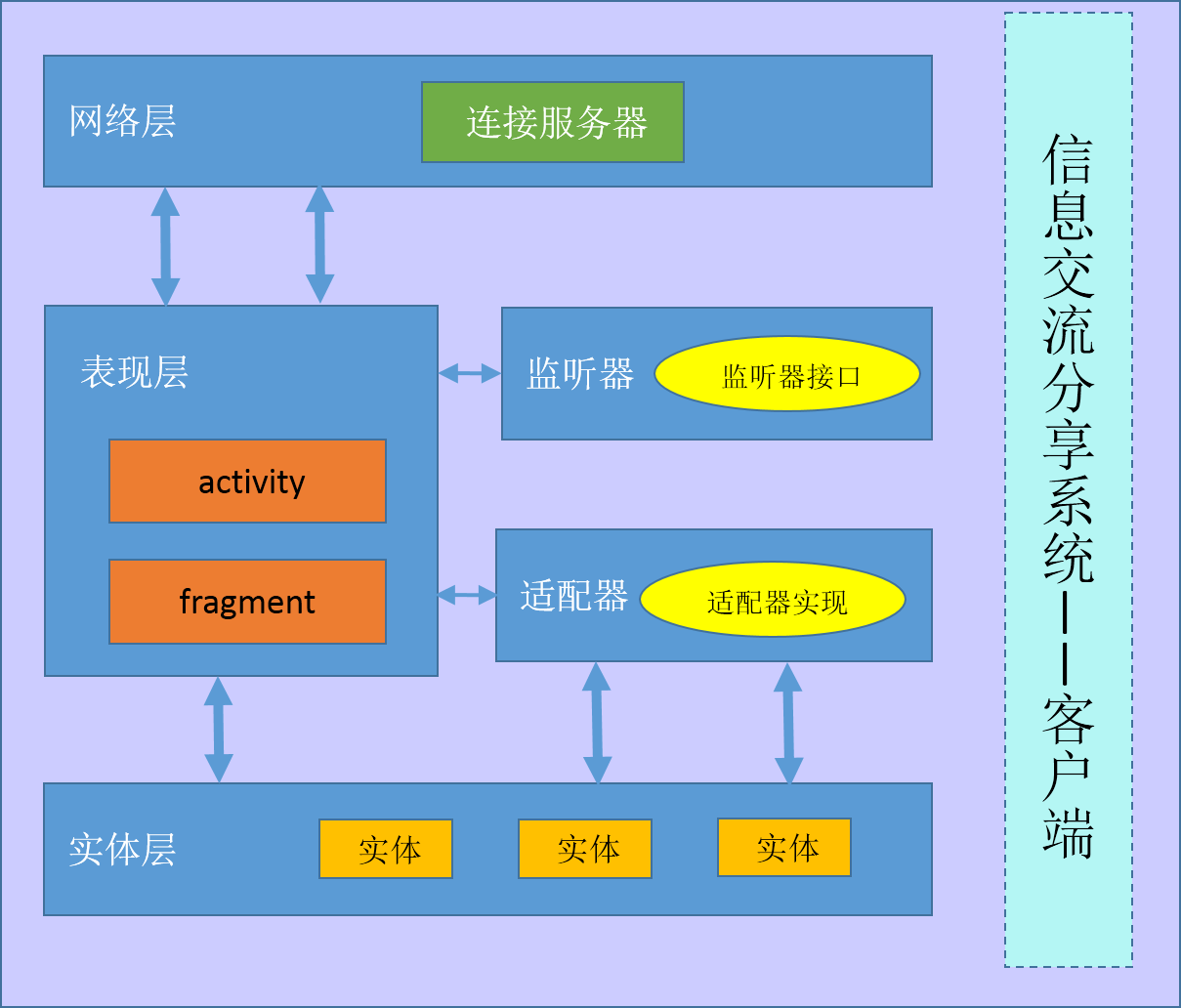


图4-2 客户端系统架构

Android端系统架构说明：

1. 实体层：基本的java类，只有简单的属性以及对属性进行获取和设置的getter、setter方法；
2. 适配器：android系统中，每个列表RecyclerView（android新的列表类）或者ListView（android旧的列表类）等都需要一个适配器adapter来为列表设定列表中每一个项目的布局layout以及为每个项目填充的数据
3. 监听器：接听器接口Listener，通过实现这些监听器接口来指定点击某个组件后应当进行怎样的响应
4. 表现层：表现层分为activity（活动）和fragment（碎片），一个activity（活动）可以包含多个fragment（碎片），activity（活动）和fragment（碎片）负责加载布局，显示界面，并响应用户在界面上的操作。
5. 网络层：负责客户端端与服务器端进行操作，向服务器发起请求request，根据该请求的不同响应response完成用户的各项操作

## 详细设计

### 帖子模块

1. 创建帖子



图4-3：“创建帖子”时序图

“创建帖子”时序图实例分析：

1. 序号1.0：用户进入CreateActivity界面，输入Post的标题、内容、图片，然后点击“发布”帖子按钮；
2. 序号1.1：客户端向服务器发出创建帖子请求，以POST的方式传入用户输入的内容；
3. 序号1.2：路由模块根据请求的路径，判断出由业务逻辑模块的PostCotroller来处理这个请求，然后调用PostController中的add( )方法来进行具体的业务操作；
4. 序号1.3：业务逻辑模块的PostController中的add( )方法调用数据访问层中PostModel中的create( )方法来完成向数据库中增加一个帖子Post的操作；
5. 序号1.4：数据访问层中PostModel的create( )方法返回在数据库中创建帖子的操作结果；
6. 序号1.5：业务逻辑层的PostController中的add( )方法，根据PostModel中的create( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
7. 序号1.6：客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户创建帖子的操作结果
8. 修改帖子

“修改帖子”程序流程分析：

1. 用户进入“MineActivity”界面，选择“我发布的”标签，显示出用户发布的帖子列表；
2. 用户选择某一个帖子，进入该帖子的DetailActivity界面，点击“修改”按钮，进入CreateActivity界面；
3. 修改Post的标题或者内容或者图片，然后点击“发布”帖子按钮；
4. 客户端向服务器发出修改帖子请求，以POST的方式传入用户输入的帖子的内容；
5. 路由模块根据请求的路径，判断出由业务逻辑模块的PostCotroller来处理这个请求，然后调用PostController中的update( )方法来进行具体的业务操作；
6. 业务逻辑模块的PostController中的update( )方法调用数据访问层中PostModel中的update( )方法来完成修改数据库中某个帖子Post的操作；
7. 数据访问层中PostModel的update( )方法返回在数据库中修改帖子的操作结果；
8. 业务逻辑层的PostController中的update( )方法，根据PostModel中的update( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
9. 客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户修改帖子的操作结果
10. 删除帖子

“删除帖子”程序流程分析：

1. 用户进入“MineActivity”界面，选择“我发布的”标签，显示出用户发布的帖子列表；
2. 用户选择某一个帖子，长按弹出提示用户删除帖子的对话框
3. 用户点击“确定”按钮，选择删除帖子
4. 客户端向服务器发出删除帖子请求，在URL中携带要删除的帖子的id
5. 路由模块根据请求的路径，判断出由业务逻辑模块的PostCotroller来处理这个请求，然后调用PostController中的delete( )方法来进行具体的业务操作；
6. 业务逻辑模块的PostController中的delete( )方法调用数据访问层中PostModel中的destroy( )方法来完成向数据库中删除一个帖子的操作；
7. 数据访问层中PostModel的destroy( )方法返回在数据库中删除帖子的操作结果；
8. 业务逻辑层的PostController中的delete( )方法，根据PostModel中的的destroy( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
9. 客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户删除帖子的操作结果
10. 浏览帖子
    1. 浏览所有帖子

“浏览所有帖子”程序流程分析：

1. 用户进入CenterActivity界面
2. 客户端向服务器发出获取帖子列表的请求
3. 路由模块根据请求的路径，判断出由业务逻辑模块的PostCotroller来处理这个请求，然后调用PostController中的get( )方法来进行具体的业务操作；
4. 业务逻辑模块的PostController中的get( )方法调用数据访问层中PostModel中的findAll( )方法来完成向数据库中获取所有用户发布的帖子的操作；
5. 数据访问层中PostModel的findAll( )方法返回在数据库中获取所有用户发布的帖子的操作结果；
6. 业务逻辑层的PostController中的get( )方法，根据PostModel中的findAll( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
7. 客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户获取所有帖子的操作结果
   1. 获取帖子详情

“获取帖子详情”程序流程分析：

1. 用户进入CenterActivity界面，浏览所有帖子
2. 用户选中某一个帖子，进入DetailActivity界面
3. 客户端像服务器发起获取某个帖子详情的请求，在请求URL中传入该帖子的id
4. 路由模块根据请求的路径，判断出由业务逻辑模块的PostCotroller来处理这个请求，然后调用PostController中的get( )方法来进行具体的业务操作；
5. 业务逻辑模块的PostController中的get( )方法调用数据访问层中PostModel中的getById( )方法来完成向数据库中获取对应id的帖子详情的操作；
6. 数据访问层中PostModel的getById( )方法返回在数据库中获取对应id的帖子详情的操作结果；
7. 业务逻辑层的PostController中的get( )方法，根据PostModel中的getById( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
8. 客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户获取对应id的帖子详情的操作结果
   1. 评论或者回复跟帖



图4-4：“添加评论”时序图

“评论或者回复跟帖”时序图实例分析：

1. 序号1.0：用户进入DetailActivity界面，浏览帖子的详情，包括帖子标题、内容、图片和帖子的跟帖等；
2. 序号1.1：用户选择某一跟帖，在该跟帖下输入评论内容，点击“发送”按钮，客户端向服务器发出创建评论请求，以POST的方式传入用户输入的评论内容；
3. 序号1.2：路由模块根据请求的路径，判断出由业务逻辑模块的PostCommentController来处理这个请求，然后调用PostCommentController中的add( )方法来进行具体的业务操作；
4. 序号1.3：业务逻辑模块的PostCommentController中的add( )方法调用数据访问层中PostCommentModel中的create( )方法来完成向数据库中增加一个评论或者回复的操作；
5. 序号1.4：数据访问层中PostCommentModel的create( )方法返回在数据库中创建评论的操作结果；
6. 序号1.5：业务逻辑层的PostCommentController中的add( )方法，根据PostCommentModel中的create( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
7. 序号1.6：客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户创建评论的操作结果
   1. 收藏帖子



图4-5：“收藏帖子”时序图

“创建帖子”时序图实例分析：

1. 序号1.0：用户进入DetailActivity界面，点击“收藏”按钮；
2. 序号1.1：客户端向服务器发出收藏帖子请求，在URL中传入用户id和要被收藏的帖子id；
3. 序号1.2：路由模块根据请求的路径，判断出由业务逻辑模块的PostCollectionController来处理这个请求，然后调用PostCollectionController中的collect( )方法来进行具体的业务操作；
4. 序号1.3：业务逻辑模块的PostCollectionController中的collect( )方法调用数据访问层中PostCollectionModel中的create( )方法来完成向数据库中增加一条评论的操作；
5. 序号1.4：数据访问层中PostCollectionModel的create( )方法返回在数据库中创建一条评论的操作结果；
6. 序号1.5：业务逻辑层的PostCollectionController中的collect( )方法，根据PostCollectionModel中的create( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
7. 序号1.6：客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户发布一条评论的操作结果

### 消息模块

1. 查看消息列表

“查看消息列表”程序流程分析：

1. 用户进入MessageActivity界面
2. 客户端向服务器发出获取消息列表的请求
3. 路由模块根据请求的路径，判断出由业务逻辑模块的MessageController来处理这个请求，然后调用MessageController中的get( )方法来进行具体的业务操作；
4. 业务逻辑模块的MessageController中的get( )方法调用数据访问层中MessageModel中的findAllByUserId( )方法来完成向数据库中获取针对该用户的所有消息的操作；
5. 数据访问层中MessageModel的findAllByUserId( )方法返回在数据库中获取针对该用户的所有消息的操作结果；
6. 业务逻辑层的MessageController中的get( )方法，根据MessageModel中的findAllByUserId( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
7. 客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户获取所有消息的操作结果
8. 查看消息详情

“查看消息详情”程序流程分析：

1. 用户进入MessageActivity界面，点击某一条消息查看消息的具体内容
2. 客户端向服务器发出获取消息详情的请求，在URL中传入消息的id
3. 路由模块根据请求的路径，判断出由业务逻辑模块的MessageController来处理这个请求，然后调用MessageController中的read( )方法来进行具体的业务操作；
4. 业务逻辑模块的MessageController中的read( )方法调用数据访问层中MessageModel中的update( )方法来完成修改数据库中消息的已读状态的操作，并通过消息的类型和消息的实际id在数据库中获得消息的详情；
5. 数据访问层中MessageModel的update( )方法返回在数据库中获取消息详情的操作结果；
6. 业务逻辑层的MessageController中的read( )方法，根据MessageModel中的update( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
7. 客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户获取帖子详情的操作结果

### 个人中心

1. 管理用户发布的帖子
2. 管理用户收藏的帖子



图4-6：“查看用户收藏的帖子”时序图

“查看用户收藏的帖子”时序图实例分析：

1. 序号1.0：用户进入MineActivity界面，选择“我收藏的”标签；
2. 序号1.1：客户端向服务器发出获取我收藏的帖子请求，在URL中传入用户的id；
3. 序号1.2：路由模块根据请求的路径，判断出由业务逻辑模块的PostCollectionController来处理这个请求，然后调用PostCollectionController中的getCollection( )方法来进行具体的业务操作；
4. 序号1.3：业务逻辑模块的PostCollectionController中的getCollection( )方法调用数据访问层中PostCollectionModel中的findAll( )方法来完成向数据库中获取对应用户收藏的所有帖子的操作；
5. 序号1.4：数据访问层中PostCollectionModel的getCollection( )方法返回在数据库中获取对应用户收藏的所有帖子的操作结果；
6. 序号1.5：业务逻辑层的PostCollectionController中的getCollection( )方法，根据PostCollectionModel中的findAll( )方法的返回结果，生成返回对象Response，然后通过http协议将服务器端的操作结果返回给客户端；
7. 序号1.6：客户端获取http的返回，根据返回的状态码和返回的data来响应用户的操作，通知用户创建帖子的操作结果

### 数据库设计

1. ER图



图4-7：ER图

1. 数据库表设计说明

表posts定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **类型** | **是否可为空** | **说明** | **备注** |
| Id | Integer | 否 | 帖子id | 自动生成 |
| UserId | Integer | 否 | 发帖人id |  |
| Type | Integer | 是 | 帖子类型 | 1：美食，2：KTV，3：丽人 |
| Title | Varchar | 否 | 帖子标题 |  |
| Content | Varchar | 否 | 帖子内容 |  |
|  |  |  |  |  |
| Time | Date | 否 | 发帖时间 |  |
| Praise | Integer | 否 | 点赞人数 |  |
| Follow | Integer | 否 | 跟帖数 |  |
| Good | Boolean | 否 | 是否为优质贴 |  |
| Collect | Integer | 否 | 收藏数 |  |

表postfollows定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **类型** | **是否为空** | **说明** | **备注** |
| Id | Integer | 否 | 跟帖id | 自动生成 |
| UserId | Integer | 否 | 发帖人id |  |
| PostId | Integer | 否 | 跟帖所属帖子id |  |
| Content | Varchar | 否 | 跟帖内容 |  |
| Time | Date | 否 | 发布跟帖的时间 |  |
| Praise | Integer | 否 | 点赞人数 |  |
| Reply | Integer | 否 | 评论和回复总数 |  |

表postcollections定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **类型** | **是否为空** | **说明** | **备注** |
| Id | Integer | 否 | 标识id | 自动生成 |
| UserId | Integer | 否 | 收藏帖子的用户id |  |
| PostId | Integer | 否 | 被收藏的帖子id |  |

表postcomments定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **类型** | **是否为空** | **说明** | **备注** |
| Id | Integer | 否 | 评论id | 自动生成 |
| PostFollowId | Integer | 否 | 评论所属跟帖id |  |
| UserId | Integer | 否 | 发布评论的用户id |  |
| toUserId | Integer | 否 | 用户2的id |  |
| Content | Varchar | 否 | 评论内容 |  |
| Time | Date | 否 | 发布评论的时间 |  |

表messages定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **类型** | **是否为空** | **说明** | **备注** |
| Id | Integer | 否 | 消息id | 自动生成 |
| SenderId | Integer | 否 | 发送消息的用户的id |  |
| ReceiverId | Integer | 否 | 接收消息的用户的id |  |
| Type | Integer | 否 | 消息类型 | 1：有跟帖，2：其他用户进行评论，3：其他用户进行回复，4：其他用户收藏了当前用户的帖子，5：其他用户点赞了当前用户的帖子 |
| ContentId | Integer | 否 | 真实的id |  |
| Read | Integer | 否 | 消息的已读未读状态 | 0：未读，1：已读 |

表users定义如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **类型** | **是否为空** | **说明** | **备注** |
| Id | Integer | 否 | 用户id |  |
| Name | Varchar | 否 | 用户姓名 |  |
| PicUrl | Varchar | 是 | 用户头像 | 存储获取用户头像的URL |

# 实现

## 开发环境

1. 服务器集成开发环境IDE：JetBrains WebStorm 2016.1.2（64）
2. 客户端集成开发环境：Android Studio 2.0（64）
3. 编译器：java SE Development Kit 8u91，android SDK
4. Android项目构建工具：gradle 1.5.0
5. 数据库：mysql 5.6.25
6. 操作系统：Windows 7 旗舰版

## 实现部分

### 帖子模块

1. 创建帖子

客户端CreateActivity布局代码：



图5-1：CreateActivity布局代码

布局代码分析：

1. 包含两个EditText分别为标题输入框和内容输入框，一个ImageView用于用户上传的图片；
2. 布局文件toolbar\_create为CreateActivity的标题栏，包含该activity的标题和“发布“按钮，点击”发布“按钮后触发向服务器发起创建请求的操作

客户端部分核心代码：



图5-2：客户端“创建帖子”核心代码

代码流程简单分析：

1. 客户端新建一个AsynHttpClient类型的对象client，用于上传图片；
2. 调用client对象的post方法上传图片，服务器返回访问图片的URL；
3. 客户端新建一个Post对象，设置它的title属性、content属性、picUrl属性；
4. 客户端新建一个StringRequest对象，设置参数为json字符串化的post实例，向服务器传输post；
5. 如果服务器返回创建成功，使用android系统的Toast提示用户成功创建

服务器端核心代码：

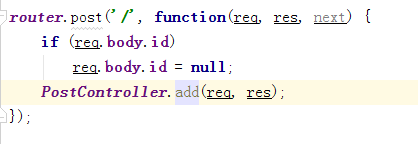


图5-3：服务器端router代码

router代码说明：通过router，根据请求路径调用PostController的add方法。

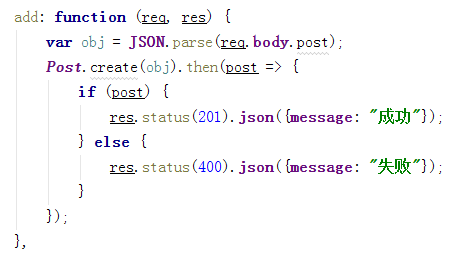


图5-4：PostController创建帖子的核心代码

PostController的add方法说明：add方法调用model层中Post的create方法向数据库中增加一个post。

1. 修改帖子

客户端布局：同CreateActivity布局文件

1. 删除帖子
2. 浏览帖子
   1. 浏览所有帖子

CenterActivity布局文件：



图5-5：CenterActivity布局文件

布局代码分析：

* + 1. Toolbar\_pool：用于显示CenterActivity的标题和“创建”按钮，点击：创建按钮几个进行帖子创建操作
    2. SwipeRefreshLayout：用于下拉刷新的空间
    3. RecyclerView：列表控件，用于包含每一个单独的子项



图5-6：列表每一个项目的布局文件

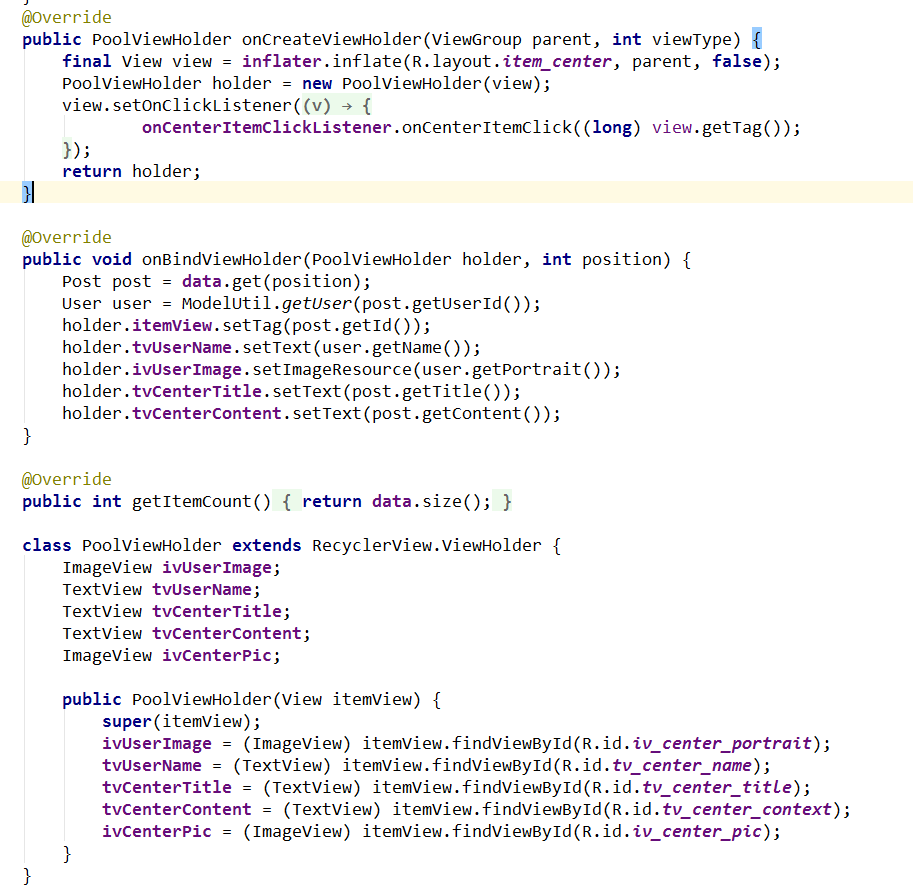


图5-7：列表的adapter的核心代码

RecyclerView的适配器PoolListAdapter的核心代码分析：

1. OnCreateViewHolder：指定每一项用到的布局，使用该布局填充view
2. OnBindViewHolder：每显示一个item，就会调用该方法，为该item填充数据
3. PoolViewHolder：保存每个item所含控件的类

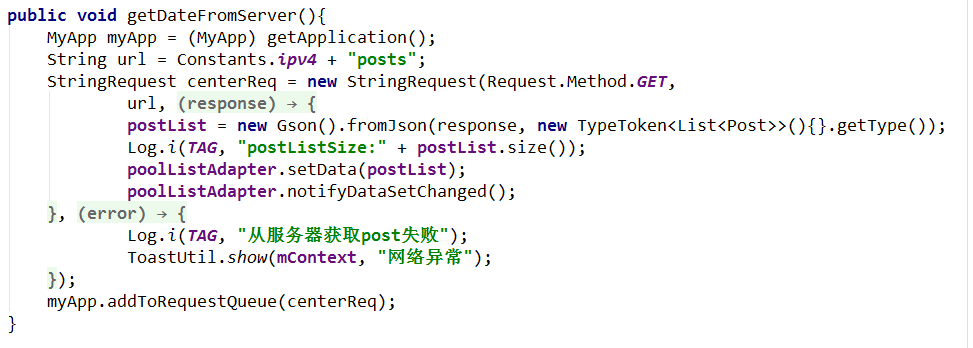


图5-8：CenterActivity核心代码

代码说明：

1. 创建一个StringRequest对象，传入请求路径，请求成功和请求失败是的处理方法
2. 将该请求添加到volley的请求列表中
   1. 查看帖子详情



图5-9：DetailActivity布局代码

* 1. 添加跟帖
  2. 评论回复
  3. 收藏帖子

### 消息模块

1. 查看消息列表
2. 查看消息详情

### 个人中心

1. 管理用户发布的帖子
2. 管理用户收藏的帖子

## 测试部分

## 运行结果

### 帖子模块

1. 创建帖子
2. 修改帖子
3. 删除帖子
4. 浏览帖子
   1. 查看帖子详情
   2. 添加跟帖
   3. 评论回复
   4. 收藏帖子

### 消息模块

1. 查看消息列表
2. 查看消息详情

### 个人中心

1. 管理用户发布的帖子
2. 管理用户收藏的帖子

# 结论与展望

## 结论

### 标题3

## 展望

图题注：