基于SpringBoot的宠物社交软件的设计与实现

Design and Implementation of Pet Social Software Based on SpringBoot

|  |  |
| --- | --- |
| 学生姓名： | 韩超 |
| 学生学号： | 1507030123 |
| 专业名称： | 网络工程 |
| 指导教师： | 李志圣 |

计算机与信息工程学院

2019年5月20日

独创性声明

本人声明所呈交的毕业设计（论文）是本人在指导教师指导下进行的研究工作和取得的研究成果，除了文中特别加以引用标注之处外，论文中不包含其他人已经发表或撰写过的研究成果，没有伪造数据的行为。

毕业设计（论文）作者签名： 签字日期：2019年6月6日

毕业设计（论文）版权使用授权书

本毕业设计（论文）作者完全了解学校有关保留、使用论文的规定。同意学校保留并向有关管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权天津城建大学可以将本论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本论文。

（保密的毕业设计（论文）在解密后适用本授权说明）

毕业设计（论文）作者签名： 指导教师签名：

签字日期：2019年6月6日 签字日期： 2019年6月6日

摘 要

在如今这样一个信息发达的时代，越来越多的人喜欢养宠物，在这一行业驱动下，越来越多的宠物店应运而生，但是如今这样一个电子商务的大背景下，各大传统行业纷纷转投互联网的怀抱，而宠物行业然仍是以线下售卖为主。

本系统基于以上的现状，以Spring-Boot为基础Web服务框架，ElasticSearch作为数据存储产品，Apache Kafka作为应用之间的消息中间件，WebSocket技术提供聊天功能支持，Framework7前端框架配合Apache Cordova技术构建手机APP。采取前后端分离的方式，为宠物店和养宠物用户提供一个互联网交易平台，同时也给热爱宠物的人提供一个经验交流的社交平台。

本系统主要功能包含：宠物订单的交易，商家个性化销售信息发布，用户基于地图API获取附近宠物店信息，用户之间的养宠经验分享

关键词：Spring-Boot；社交；电子商务；WebSocket

**ABSTRACT**

In today's era of information development, more and more people like to raise pets. Driven by this industry, more and more pet shops have emerged, but nowadays, under the background of e-commerce, major Traditional industries have turned to the embrace of the Internet, while the pet industry is still dominated by offline sales.

Based on the above situation, this system uses Spring-Boot as the basic Web service framework, ElasticSearch as the data storage product, Apache Kafka as the message middleware between applications, WebSocket technology to provide chat function support, Framework7 front-end framework and Apache Cordova technology to build mobile APP. The separation of front and rear ends provides an Internet trading platform for pet shops and pet users, and also provides a social platform for people who love pets to exchange experiences.

The main functions of the system include: pet order transactions, merchant personalized sales information release, users based on the map API to obtain information on nearby pet stores, sharing experience between users

**Key words：**Spring-Boot；Social；E-commerce；WebSocket

目 录

[基于SpringBoot的宠物社交软件的设计与实现 1](#_Toc9786423)

[计算机与信息工程学院 1](#_Toc9786424)

[第1章 绪论 1](#_Toc9786425)

[1.1开发背景和现状 1](#_Toc9786426)

[1.2开发意义 1](#_Toc9786427)

[第2章 系统可行性分析 2](#_Toc9786428)

[2.1可行性分析 2](#_Toc9786429)

[2.1.1 技术可行性 2](#_Toc9786430)

[2.1.2 社会可行性 2](#_Toc9786431)

[2.2开发环境以及技术 2](#_Toc9786432)

[2.3设计目标 3](#_Toc9786433)

[第3章 需求分析 4](#_Toc9786434)

[3.1 角色划分 4](#_Toc9786435)

[3.1.1 商家角色 4](#_Toc9786436)

[3.1.2 用户角色 4](#_Toc9786437)

[3.2 需求概述 5](#_Toc9786438)

[3.3 系统运行环境 5](#_Toc9786439)

[3.3.1 服务器端 5](#_Toc9786440)

[3.3.2 客户端 5](#_Toc9786441)

[第4章 功能设计 6](#_Toc9786442)

[4.1 公共功能 6](#_Toc9786443)

[4.1.1 注册和登录 6](#_Toc9786444)

[4.1.2 操作认证 6](#_Toc9786445)

[4.1.3 文件资源上传，查看和管理 6](#_Toc9786446)

[4.1.4 信息修改 6](#_Toc9786447)

[4.2 商家角色操作 7](#_Toc9786448)

[4.2.1 销售信息发布 7](#_Toc9786449)

[4.2.2 订单流程 8](#_Toc9786450)

[4.3 用户角色操作 8](#_Toc9786451)

[4.3.1 订单发起 8](#_Toc9786452)

[4.3.2 动态发布 9](#_Toc9786453)

[4.4 聊天功能设计 9](#_Toc9786454)

[第5章 系统设计 11](#_Toc9786455)

[5.1 实体设计 11](#_Toc9786456)

[5.1.1 资源实体 11](#_Toc9786457)

[5.1.2 商家实体 11](#_Toc9786458)

[5.1.3 用户实体 12](#_Toc9786459)

[5.1.4 库存实体 12](#_Toc9786460)

[5.1.5 订单实体 13](#_Toc9786461)

[5.1.6 动态消息实体 13](#_Toc9786462)

[5.1.7 评论实体 14](#_Toc9786463)

[5.1.8 会话实体 14](#_Toc9786464)

[5.2 系统整体架构 15](#_Toc9786465)

[5.2.1 用户端架构 15](#_Toc9786466)

[5.2.2 商家端架构 16](#_Toc9786467)

[5.2.3 整体架构 17](#_Toc9786468)

[第6章 系统实现 18](#_Toc9786469)

[6.1 商家端实现 18](#_Toc9786470)

[6.1.1 登录，注册和个人信息修改页面 18](#_Toc9786471)

[6.1.2 销售信息发布 20](#_Toc9786472)

[6.2 用户端实现 20](#_Toc9786473)

[6.2.1 登录，注册和个人信息修改页面 20](#_Toc9786474)

[6.2.2 销售信息查看 22](#_Toc9786475)

[6.2.3 动态信息发布，查看和评论 22](#_Toc9786476)

[6.3 下单流程实现 23](#_Toc9786477)

[6.3.1 下单 23](#_Toc9786478)

[6.3.2 接单和拣货 24](#_Toc9786479)

[6.3.3 确认 25](#_Toc9786480)

[6.4 聊天功能实现 25](#_Toc9786481)

[**总 结** 26](#_Toc9786482)

[致 谢 27](#_Toc9786483)

[参考文献 28](#_Toc9786484)

# 第1章 绪论

## 1.1开发背景和现状

国民生活水平的提高，对精神生活的更高追求，越来越国际化的社会环境等，使宠物行业初具雏形。 随着改革开放的浪潮翻涌，大量"海归"回归并带入了国际上各种包括宠物在内的新知概念，加之国际化宠物产品品牌陆续在中国建厂并通过不同的渠道进入人们的视线，就此引发了国内全新的宠物概念。

虽然国内宠物市场相比于国际宠物市场起步较晚，在过去很长一段时间处于初级阶段，但随着我国社会经济的发展、城市化进程的加速、人均收入水平的提高以及老龄化步伐的加快，饲养宠物的家庭日益增多，宠物产业也随着逐渐发展壮大。

2017年，中国宠物行业市场规模达到1340亿元，同期美国宠物市场规模为695.1亿美元，约是我国的3.5倍。2004-2017年宠物行业市场规模复合增长率为38.18%，行业保持高速发展；年增速有放缓趋势，逐渐向稳定与成熟期过渡。

根据《2017年中国宠物行业白皮书》数据显示，中国养宠家庭正在增加，2017年为5912万，中国养宠家庭比例较去年增长了2%，渗透率达到17%。仍有较大增长空间。在养宠用户中，养狗家庭比例最高，达到60.3%，养猫比例接近40%，猫和狗是最主要的宠物类别。宠物狗的数量约4990万只，宠物猫的数量约3756万只。但也有一部分养宠用户会选择其他种类的宠物，包括水族、爬行类、龋齿类等。

伴随着宠物行业的发展,越来越多的宠物店开始在城市兴起,这些宠物店大多以线下销售为主,但是随着今年来电商行业的火爆,传统线下销售行业都受到了不同程度的冲击,为此一些宠物店做起了微商,或者以微信群的方式进行线上的销售,国内暂时没有比较好的宠物交易线上平台

## 1.2开发意义

旨在为宠物店主和喜欢养宠物的人构建一个交流的平台，让宠物行业也可以享受到互联网的的益处，同时也为更多的年轻人提供一个养宠经验交流的平台

# 第2章 系统可行性分析

## 2.1可行性分析

### 2.1.1 技术可行性

本系统属于B2C电商类项目，目前市面上针对电商类项目的开源产品很丰富，相关解决方案也日趋成熟。在社交方面，由于有websocket的支持，构建聊天系统也十分快捷

### 2.1.2 社会可行性

自从电商开始在中国兴起，越来越多的人选择足不出户，在家用手机进行购物，但是针对宠物行业，市面上并没有相关电商产品，这也使得大多数人不得不去线下购买，此系统可以为宠物爱好者和宠物店主之间提供一个交易平台，是可以迎合很多人的需求的。

同时，针对年轻人喜欢分享养宠经验的心理需求，此系统也提供了分享动态功能，大大增加了受众人群。

## 2.2开发环境以及技术

开发语言使用Java1.8版本，开发工具为IntelliJ IDEA Community Edition

服务端开发基于Spring-Boot框架，Spring-Boot是由Pivotal公司推出的一款全新创建，用于简化Spring应用的搭建和开发。使用Spring-Boot可以很容易的创建出一个以Spring为核心的独立应用，不需要依赖于任何外部环境。

数据存储方面，考虑到此类电商系统往往需要大量组合条件查询，同时考虑到存储的数据类型可能存储集合这种复杂类型，传统的关系型数据库在这方面的兼容性做的并不好，因此，我考虑使用新型的数据存储产品ElasticSearch，该产品不但支持多种存储类型，还在多条件查询方面做了很多优化，非常适合该项目

在系统交互方面，由于需要设计两套系统（商家和用户），因此必然会涉及到系统之间的数据交互，传统的方式是才有Http请求的方式去调用相应api，但是这样做系统的耦合度会很高，因此我考虑使用消息队列的方式去解耦合，同时消息队列还能增加系统可用性

用户界面方面我考虑以手机APP的方式为主，考虑到我在Android和iOS开发方面的知识欠缺，因此准备使用WebApp的方式去搭建，前端框架使用Framework7，这是一款能快速构建原生App的前端框架，Apache Cordova用来将写到的Web页面转换成对应的Android APP和iOS APP

聊天应用方面，现有市场大多使用长连接或者Http轮询的方式进行构建，这两种方式对服务器的开销特别大，幸运的是，RFC 6455所定义的WebSocket标准完美解决了这一问题。WebSocket是一种全双工的连接方式，基于Http协议的同时也支持持久化连接

所需开源产品和技术清单如下：

表2-1 开源产品&技术依赖

|  |  |  |
| --- | --- | --- |
| **名称** | **版本&标准** | **描述** |
| Java | 1.8 | 程序运行环境 |
| Spring-Boot | 1.5.9.FINAL | 创建可以独立运行的Spring程序 |
| Elasticsearch | 2.4.6 | 方便快速的数据存储开源产品 |
| Apache Zookeeper | 3.4.10 | 开源的分布式应用程序协调服务 |
| Apache Kafka | 0.10.1.1 | 高吞吐率的分布式消息队列 |
| Scala | 2.11 | Apache Kafka运行依赖库 |
| WebSocket | RFC 6455 | 全双工通信协议 |
| Apache Cordova | 9.0.1 | 搭建移动应用程序 |
| Framework7 | 4.1.1 | 用于构建iOS&Android页面的HTML框架 |

## 2.3设计目标

1)基本功能

对于宠物店主来说:

1.发布销售信息以吸引消费者购买

2.订单管理

对于消费者来说:

1.发布宠物动态,评论好友的动态

2.根据自身地理位置结合地图服务获取附近宠物店的销售信息和其他人的宠物动态

# 第3章 需求分析

## 3.1 角色划分

系统大致分为两大角色：用户和商家

### 3.1.1 商家角色

商家角色主要承担交易关系中的卖方，提供销售信息

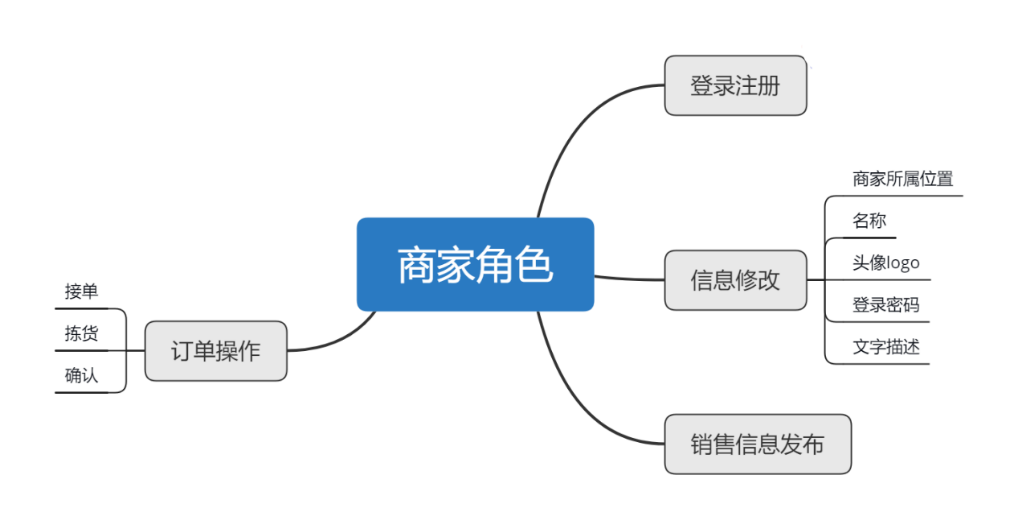


图3-1 商家角色

### 3.1.2 用户角色

用户角色主要承担交易关系用的买方，发起订单和发布动态，是社交的关键一环

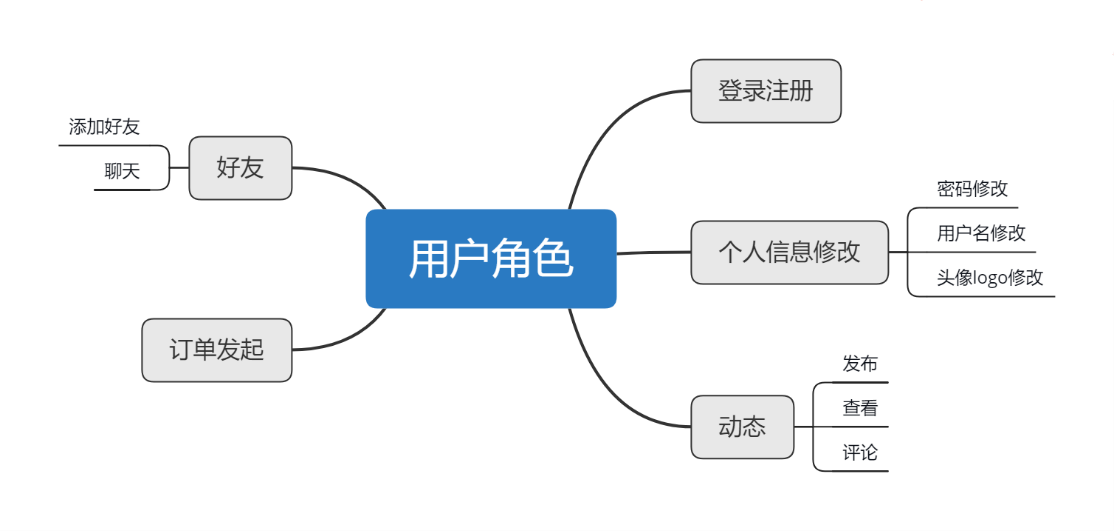


图3-2 用户角色

## 3.2 需求概述

登录注册：使用系统必须拥有账号进行登录方可进行后续操作，没有账号的用户或者商家均可快速注册一个账号

个人信息修改：提供修改密码，个人信息等功能。对于商家，支持修改头像logo，店家名称和地址。对于普通用户，支持修改头像和用户名

操作安全机制：系统应设计一套安全认证机制，无论是商家还是用户都必须通过认证方可进行后续操作，认证失败则不允许操作并且会退出当前登录状态

销售信息发布和浏览：由商家发布销售信息，用户查看后可以进行购买操作，同时考虑到宠物类等活物的特殊性，应该有条件的进行筛选

订单操作：用户选择好需要购买的商品后，可以进行下单操作，下单之后会第一时间通知到商家，商家开始进行下单后的后续操作，同时会同步订单状态到用户，用户可以第一时间跟踪到订单状态

动态信息发布和评论：用户可以凭据自己喜欢发布动态，改动态可以被改用户的好友查看，同时支持评论功能

聊天功能：聊天功能方便用户和商家进行交流，同时也方便用户与好友之间进行经验分享。

## 3.3 系统运行环境

### 3.3.1 服务器端

内存：4GB以上

Java环境：1.8以上

Scala版本：2.11

ElasticSearch版本：2.4.6

Apache Kafka版本：0.10.1.1

Apache Zookeeper版本：3.4.10

### 3.3.2 客户端

安卓版本：必须支持WebView

内存：4GB以上

所需安卓权限：文件浏览读写，拍照

# 第4章 功能设计

考虑到两个角色（商家和客户）都有一些类似的功能，因此先提炼出公共的功能进行设计，再针对单个系统进行设计

## 4.1 公共功能

### 4.1.1 注册和登录

系统提供常规注册和登录功能。

商家注册时需要使用邮箱注册，普通用户可以使用自定义的用户名进行注册。用户名和邮箱应保证唯一性。

注册时会将注册的密码进行MD5加密并存储，登录时也会将用户输入的密码进行MD5加密，并进行比对

### 4.1.2 操作认证

无论是商家还是用户，操作都应该处于认证状态下，否则会引发系统安全为题，认证方式如下：

1. 商家和用户进行登录，登录完成后服务端会根据相应的登录信息返回一个token返回给客户端，token和用户信息一并存入Redis中，默认存储7天
2. 客户端收到token后，将其存储在本地，后续有任何操作需要认证时，只需在发送请求的同时将本地存储的token一并发送给服务端去验证
3. 服务器端一旦接收到客户端传过来的token，首先验证该token对应在Redis中的用户信息是否一致，一旦不一致或者token以过期，服务端停止请求响应，并要求客户端的用户进行重新登录

### 4.1.3 文件资源上传，查看和管理

支持多媒体文件上传，使用统一的资源模型，存储在服务器指定位置，每一个资源都拥有一个标识，用于标记该资源是否正在被引用。

同时提供定时删除功能，删除不再被引用的资源文件，减少服务器资源占用，定时周期为一周

### 4.1.4 信息修改

提供信息修改功能，无论是商家还是用户，都可以在信息页面修改自己信息，如头像，文字描述，用户名等等。头像上传目前只支持图片的方式，如需要上传动态头像，应该先转换为gif格式，否则系统不允许上传

特别的，针对商家还提供了位置定位的功能，通过腾讯地图API的方式方便快捷的定位

## 4.2 商家角色操作

对于商家来说，最核心的业务点在于销售信息的发布和订单的操作

### 4.2.1 销售信息发布

销售信息用于展示商家售卖产品，支持多种多媒体格式，如视频，图片，文字等等，具体流程如下所示：

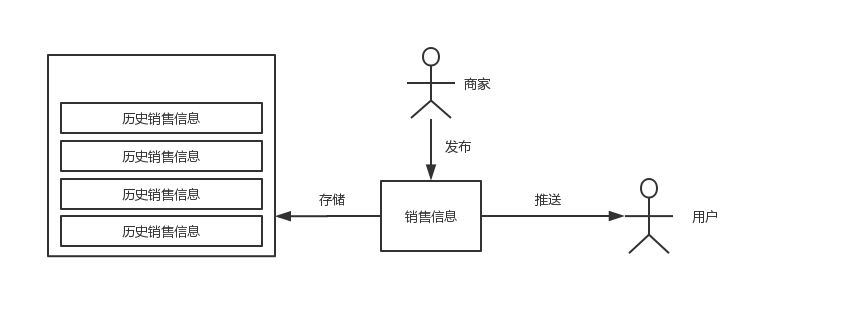


图4-1 商家销售信息流程

商家在发布完成销售信息后，用户会第一时间查看到该销售信息，同时考虑到宠物这类活物无法远距离寄送的特点，发布信息之后只会推送给附近的用户。

地理位置范围的考量标准以经纬度正负差值为准，默认值为0.5，如用户所处位置的经度为118.61°，纬度为31.425°，那么默认情况下只会推荐纬度范围为118.11°～119.11°，经度范围为30.925°～31.925°的商家的销售信息进行推送，这个正负值用户可以自行调整。

为了方便用户查看，在用户端的界面上会以地图的形式呈现给用户，用户可以方便快捷的定位到大致位置。地图服务使用腾讯地图

在推送时，在用户界面里一个商家只会显示一条销售信息，这条销售信息为商家最近创建的销售信息，用户进入某一销售信息后，可以选择去查看该商家发布的所有历史销售信息。

发布完成后，该销售信息便会存储到商家的历史销售信息列表之中，以便后续查看

### 4.2.2 订单流程

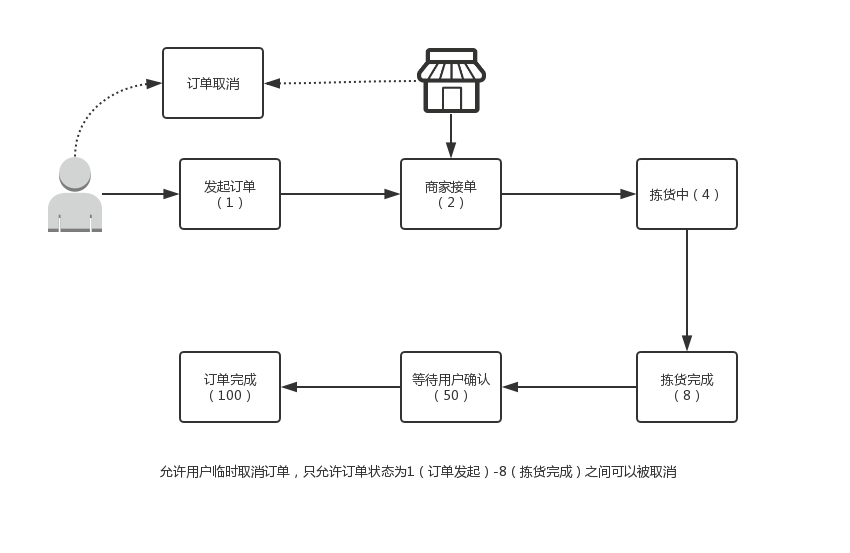


图4-2 订单流程

各种订单状态的说明：

1. 订单创建：由用户发起订单，并实时推送到商家
2. 商家接单：商家查看到用户发起的订单后，可以根据实际情况进行接单操作
3. 拣货中：商家接单后，开始进行拣货操作
4. 拣货完成：此时商家已经拣货完成
5. 等待用户确认：商家拣货完成后，系统会发送消息给对应用户，提示用户已经拣货完成，并等待用户前往门店取货
6. 订单完成：在用户取完货之后，商家可以将该订单关闭，代表订单已经完成

同时，系统允许用户临时取消订单，但是只能在订单状态在创建和拣货完成之间时取消，一旦商家拣货完成，用户则无法取消

## 4.3 用户角色操作

对于用户而言，常用的操作主要为订单发起和动态发布

### 4.3.1 订单发起

考虑到寄送成本的问题，推荐给用户的销售信息应该考虑地理信息，即用户只会接受到自己所处位置附件的销售信息

用户选择好要购买的东西后，确认完数量，即可进行下单操作

下单完成后，可以在个人信息->所有订单界面实时查看订单状态，同时可以向对应的商家发起聊天，以确认订单的详细需求

### 4.3.2 动态发布

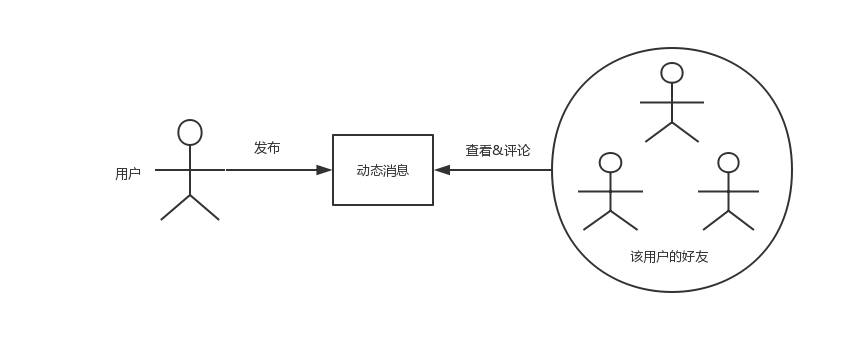


图4-3 用户动态发布

用户可以根据自己喜好发布动态信息，动态信息可以包含图文，视频等文件。发布后，用户和用户的好友即可立即查看，同时支持评论，增加社交性

## 4.4 聊天功能设计

聊天功能是社交环节的重要部分，聊天的类型大致可以分为两种，用户对商家和用户对用户

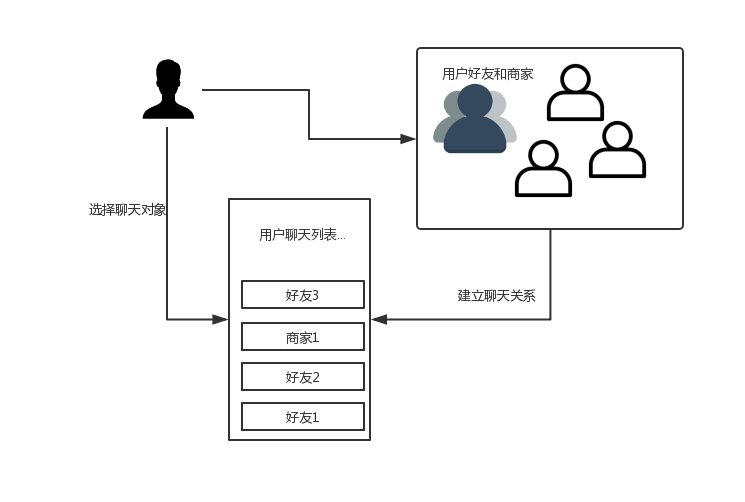


图4-4 聊天功能

聊天的发起者为用户

用户可以选择一个角色去建立聊天关系，这个角色可以是用户的好友，也可以是商家，每一个用户都会单独维护一个聊天列表

每个角色都对应一个Session，这个Session伴随从角色登录伴随到角色注销，用户选择要进行聊天的角色后，将对应角色的Session和该用户进行配对，即将Session添加到对应用户的聊天列表中去。配对完成之后，下次用户便可以直接从聊天列表中选取对应的聊天对象进行聊天。

聊天的消息可以是文字也可以是图片视频，图片视频会以资源的形式上传至服务器，上传完成后立即标记该资源处于未被使用状态，等待系统进行删除回收。

# 第5章 系统设计

## 5.1 实体设计

### 5.1.1 资源实体

为了简化设计，将所有资源文件类型用统一实体模型表示

表5-1 资源实体

|  |  |  |
| --- | --- | --- |
| **属性** | **数据类型** | **描述** |
| id | String | 资源id，唯一标识符 |
| path | String | 存储路径，相对于classpath的相对路径 |
| timestamp | Long | 创建时间戳 |
| type | String | 资源类型，比如image/jpeg |
| hasUsed | Boolean | 该资源是否被使用，不被使用的资源可以被删除 |

### 5.1.2 商家实体

表5-2 商家实体

|  |  |  |
| --- | --- | --- |
| **属性** | **数据类型** | **描述** |
| id | String | 商家id，唯一标识符 |
| name | String | 店名称 |
| description | String | 商家描述 |
| authenticated | Boolean | 是否为已认证商家 |
| email | String | 邮箱，登录账户 |
| password | String | 登录密码，存储应该使用MD5加密 |
| logo | String | 商家logo.存储为对应资源id |
| longitude | Double | 经度 |
| latitude | Double | 纬度 |
| address | String | 商家详细地址 |

### 5.1.3 用户实体

表5-3 用户实体

|  |  |  |
| --- | --- | --- |
| **属性** | **数据类型** | **描述** |
| id | String | 用户id，唯一标识符 |
| username | String | 用户名，登录使用 |
| password | String | 登录密码 |
| logo | String | 用户头像，对应资源id |
| friends | Set<String> | 好友列表，集合类型，每个元素为用户id |
| shopkeepers | Set<String> | 关注的商家列表，集合类型，每个元素为商家id |

### 5.1.4 库存实体

库存为商家所拥有的货品，可以根据一个库存对应一个销售信息，用于向用户展示

表5-4 库存实体

|  |  |  |
| --- | --- | --- |
| **属性** | **数据类型** | **描述** |
| id | String | 库存id，唯一标识符 |
| resources | List<String> | 关于货品的资源描述，视频图片都可以 |
| title | String | 货品名称 |
| logo | String | 库存展示图片，对应资源id |
| text | String | 文字秒速 |
| shopkeeperId | String | 所属商家id |
| last | Integer | 剩余数量 |
| price | Float | 售卖单价，单位为RMB |
| timestamp | Long | 创建时间戳 |

### 5.1.5 订单实体

考虑到需要远程传输，需要实现java.io.Serializable接口，同时约定serialVersionUID为123456L

表5-5 订单实体

|  |  |  |
| --- | --- | --- |
| **属性** | **数据类型** | **描述** |
| id | String | 订单id，唯一标识符 |
| userId | String | 买方id |
| shopkeeperId | String | 卖方id |
| createTimeStamp | Long | 创建时间，时间戳表示 |
| shockId | String | 购买商品，对应库存id |
| count | Integer | 购买数量 |
| status | Integer | 订单状态 |

### 5.1.6 动态消息实体

表5-6 动态消息实体

|  |  |  |
| --- | --- | --- |
| **属性** | **数据类型** | **描述** |
| id | String | 动态消息id，唯一标识符 |
| title | String | 标题 |
| content | String | 文字内容 |
| user | String | 创建用户id |
| imgs | List<String> | 资源描述，集合，每个元素为资源id |
| timestamp | Long | 创建时间戳 |

### 5.1.7 评论实体

表5-7 评论实体

|  |  |  |
| --- | --- | --- |
| **属性** | **数据类型** | **描述** |
| id | String | 评论id，唯一标识符 |
| content | String | 评论内容 |
| parent | String | 父级评论id |
| userId | String | 评论者id |
| message | String | 所属消息id |
| timestamp | Long | 创建时间戳 |

### 5.1.8 会话实体

会话实体为接口，针对商家和用户分别有两种对应实现，用于保存聊天会话

表5-8 评论实体

|  |  |  |
| --- | --- | --- |
| **属性** | **数据类型** | **描述** |
| id | String | 用户/商家id |
| Session | javax.websocket.Session | 对应websocket会话 |
| open | Boolean | 该会话是否关闭 |
| type | Integer | 类型，1为用户会话，2为商家会话 |
| lastTimeStamp | Long | 上次活跃时间 |

## 5.2 系统整体架构

### 5.2.1 用户端架构

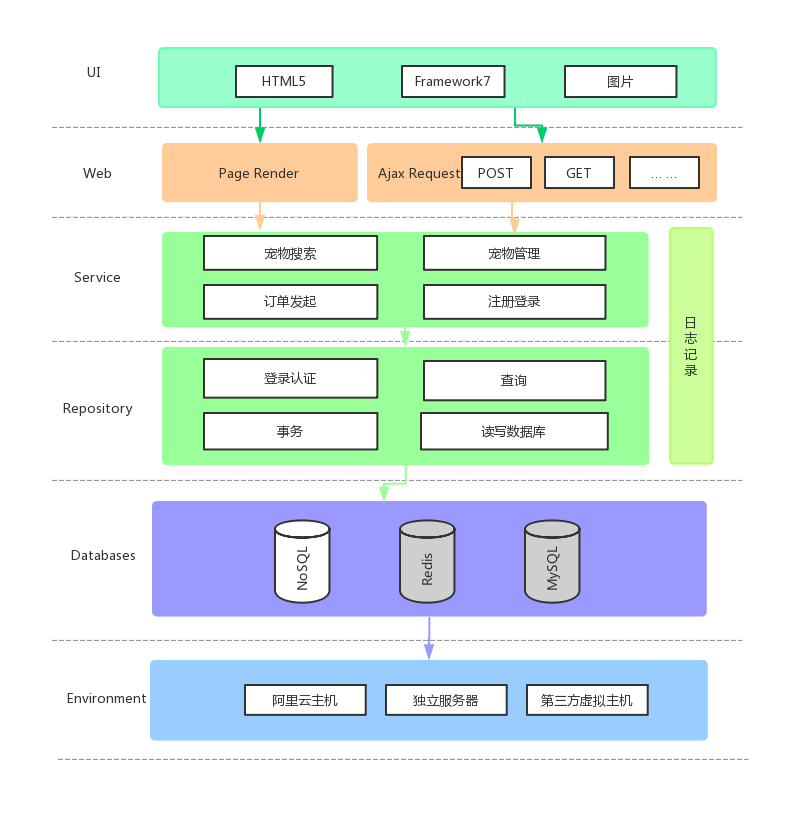


图5-1 用户端服务架构

针对用户端，总共分为6层，Databases层和Environment层提供底层基础服务。

Repository层用于提供数据查询基础服务，比如基本实体类的增删改错等。

Service层则主要提供业务逻辑处理，比如销售信息查询，用户登录注册，动态消息发布以及评论等业务操作。

Web和UI层主要提供用户操作接口和操作界面，采用前后端分离的方式。前端页面通过Ajax技术发送Http请求到Web层接口，获取相应数据，然后在前端通过页面渲染技术展示给用户

### 5.2.2 商家端架构

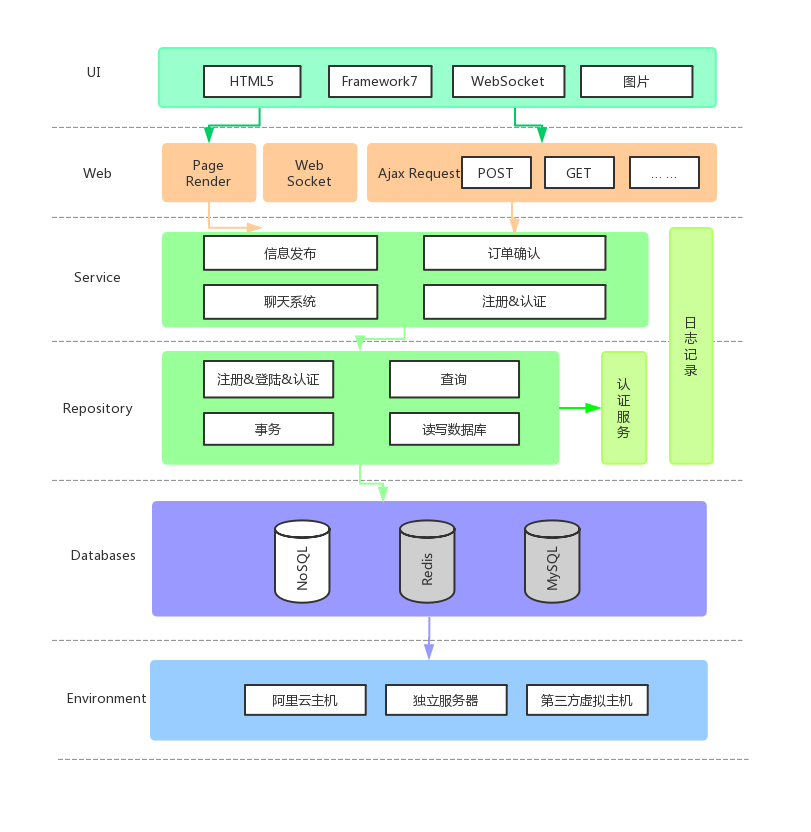


图5-2 商家端架构

类似于用户端，商家端同样分为6层，唯一的不同点在于Service层，Service实现了商家用户需要用到的业务逻辑，同时增加了商家认证服务

### 5.2.3 整体架构

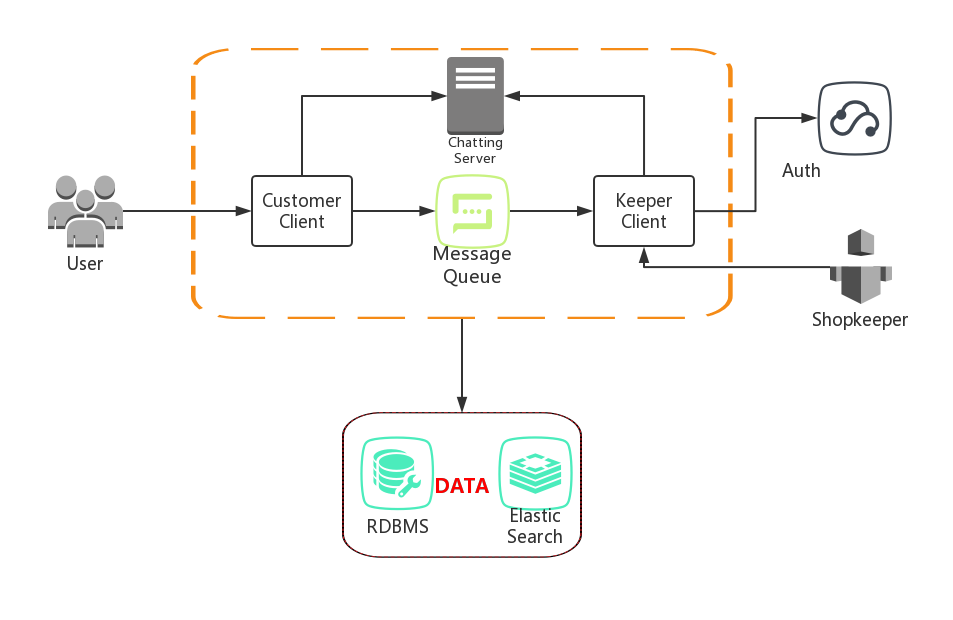


图5-3 整体架构

由RDBMS和ElasticSearch提供数据存储服务，分配对应用户端和商家端的Databases层，用户端和商家端使用同一数据存储服务

商家端和用户端直接的订单传递使用消息队列进行传递，这样做的好处可以减小系统藕合度，同时可以增加系统可用性，比如订单量很大时，可以通过消息队列对流量进行削峰处理，降低系统并发数

同时为了能够提供聊天服务给商家和用户，单独提供了一台聊天服务器用于进行聊天信息的交换

整套系统提供两套用户界面，分别用于商家和普通用户。UI相关资源以App的形式提供给用户和商家，服务端以Spring-Boot应用的方式建立服务，执行方式为Jar包

# 第6章 系统实现

## 6.1 商家端实现

### 6.1.1 登录，注册和个人信息修改页面

图6-1页面分别为登录，注册页面，可以进行登录注册操作

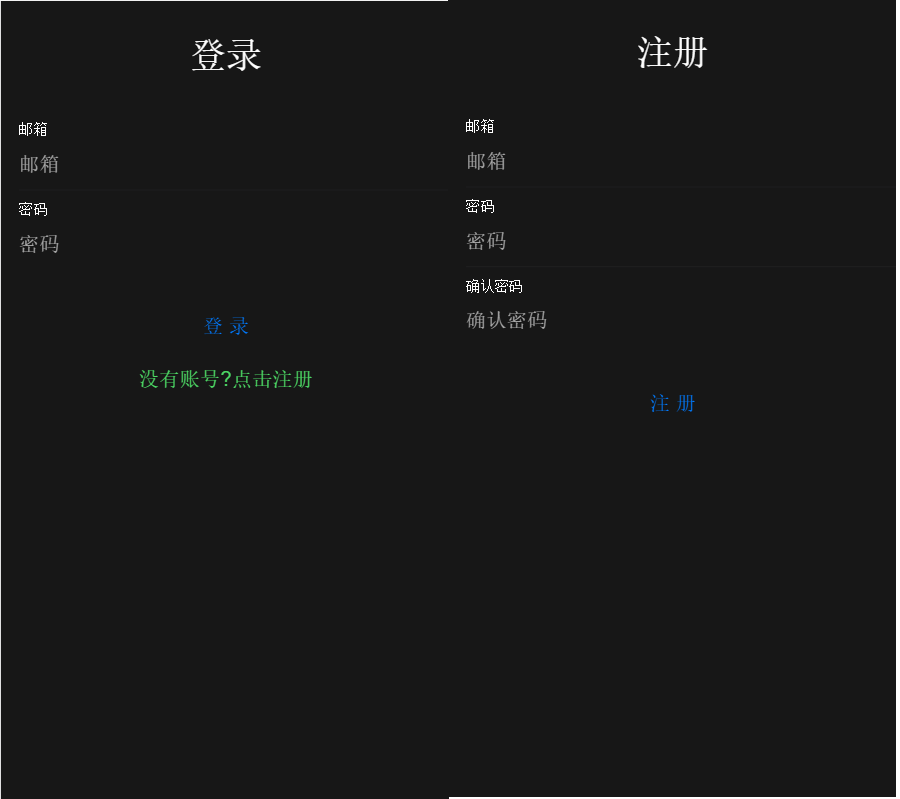


图6-1 商家注册和登录

图6-2展示的是个人信息修改页面，可以修改店铺名称，头像和文字描述

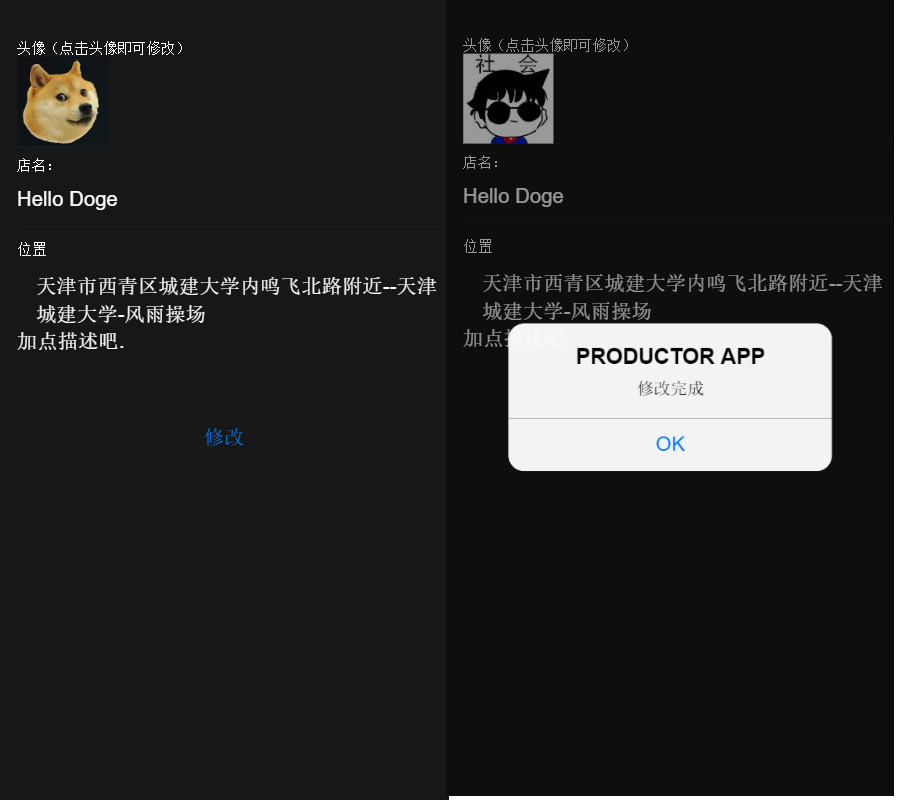


图6-2 商家个人信息修改

图6-3展示的是修改店铺地理位置，使用腾讯地图API进行地位

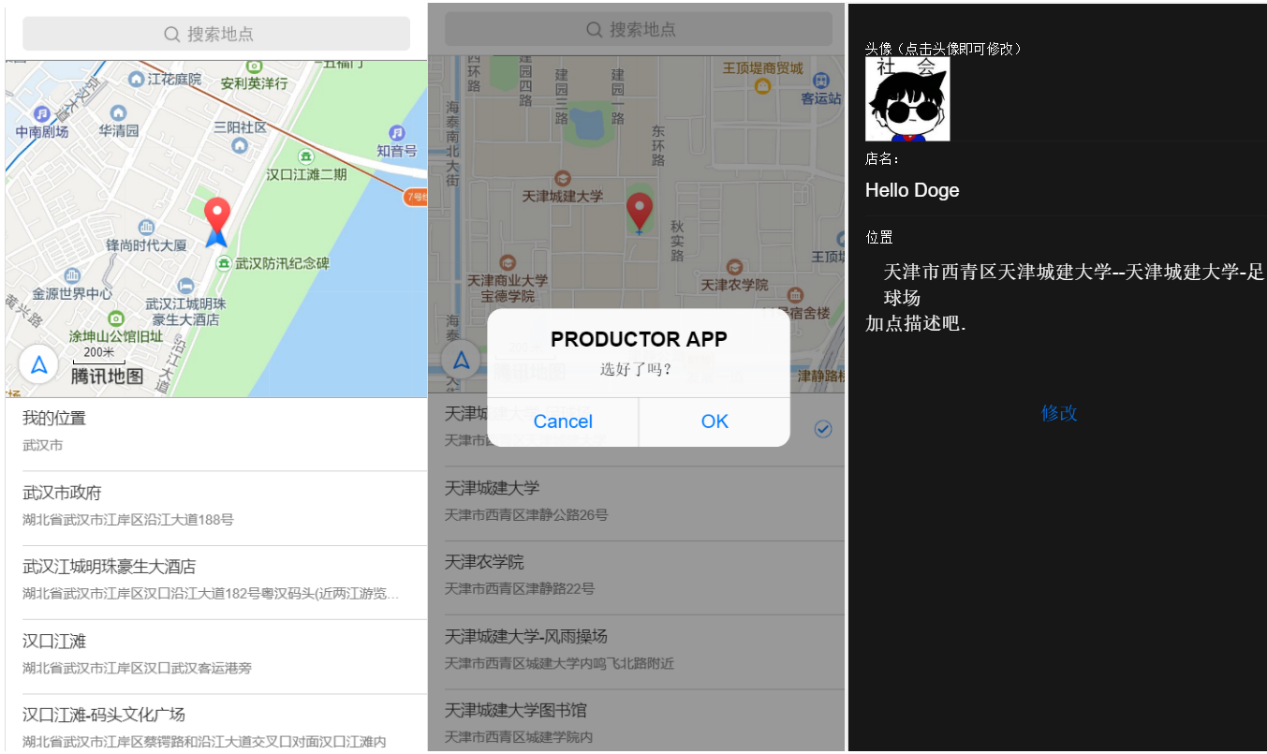


图6-3 商家修改地理位置

### 6.1.2 销售信息发布

图6-4展示商家如何发布消息，商家上传完相应文件，填写完成相应信息即可发布

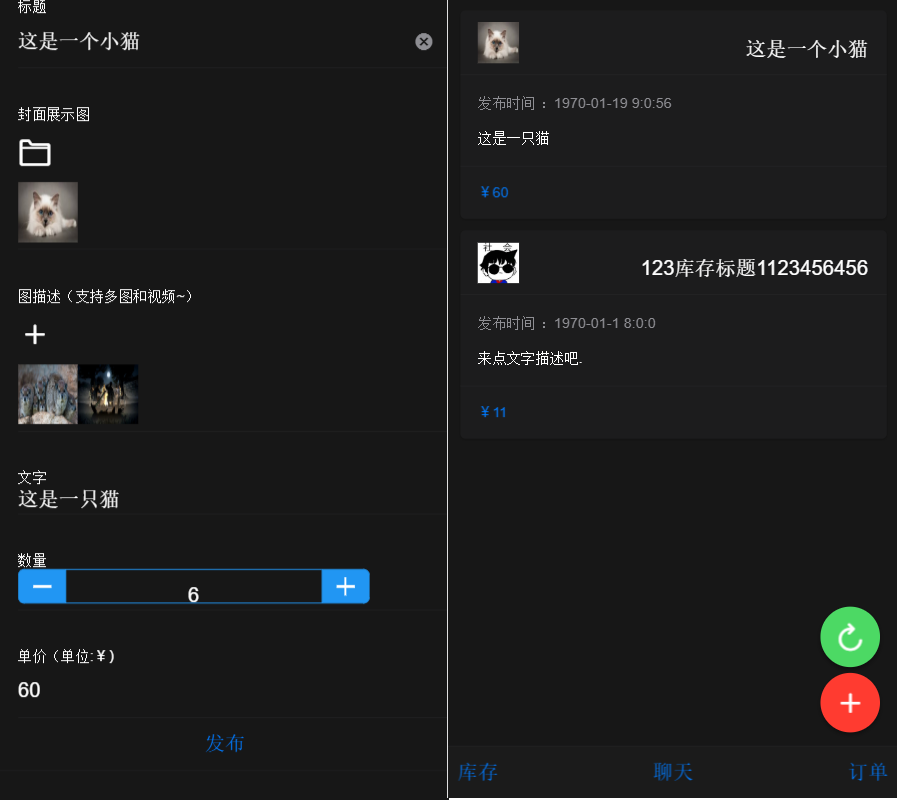


图6-4 商家销售信息发布

## 6.2 用户端实现

### 6.2.1 登录，注册和个人信息修改页面

图6-5页面分别为登录，注册页面，可以进行登录注册操作



图6-5 用户登录和注册

图6-6展示的是个人信息修改页面，可以修改用户名和头像

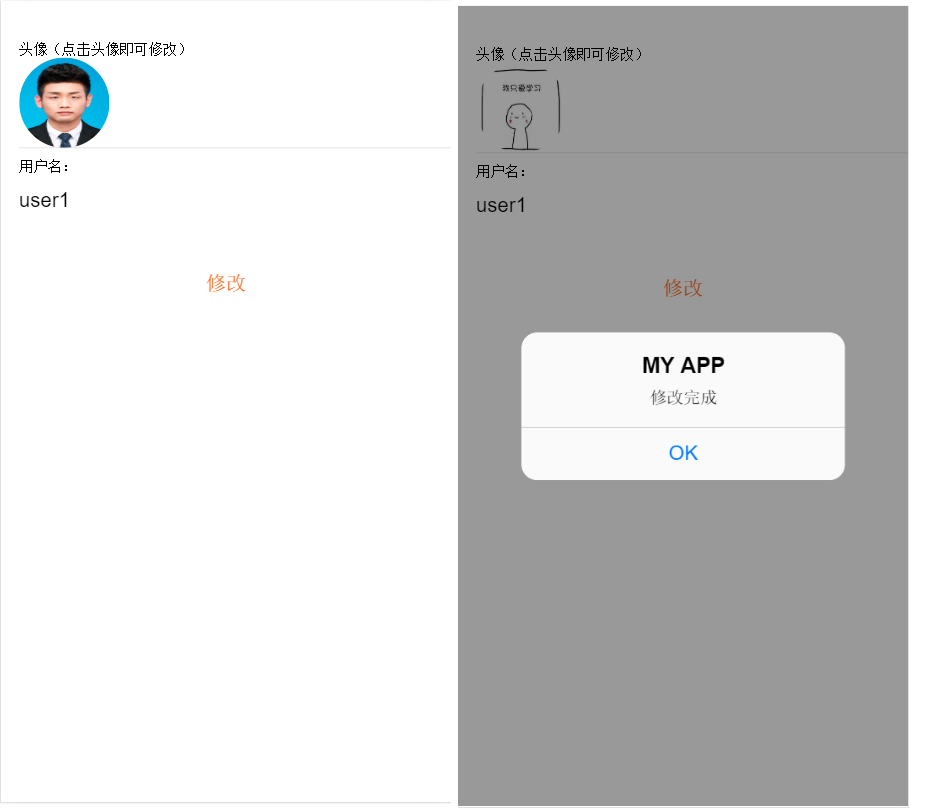


图6-6 用户个人信息修改

### 6.2.2 销售信息查看

图6-7为用户端查看到的销售信息，显示在地图上，点击地图上的任何一销售信息，可以进入对应的详情页，点击上方商家名称，可以看出该商家历史销售信息

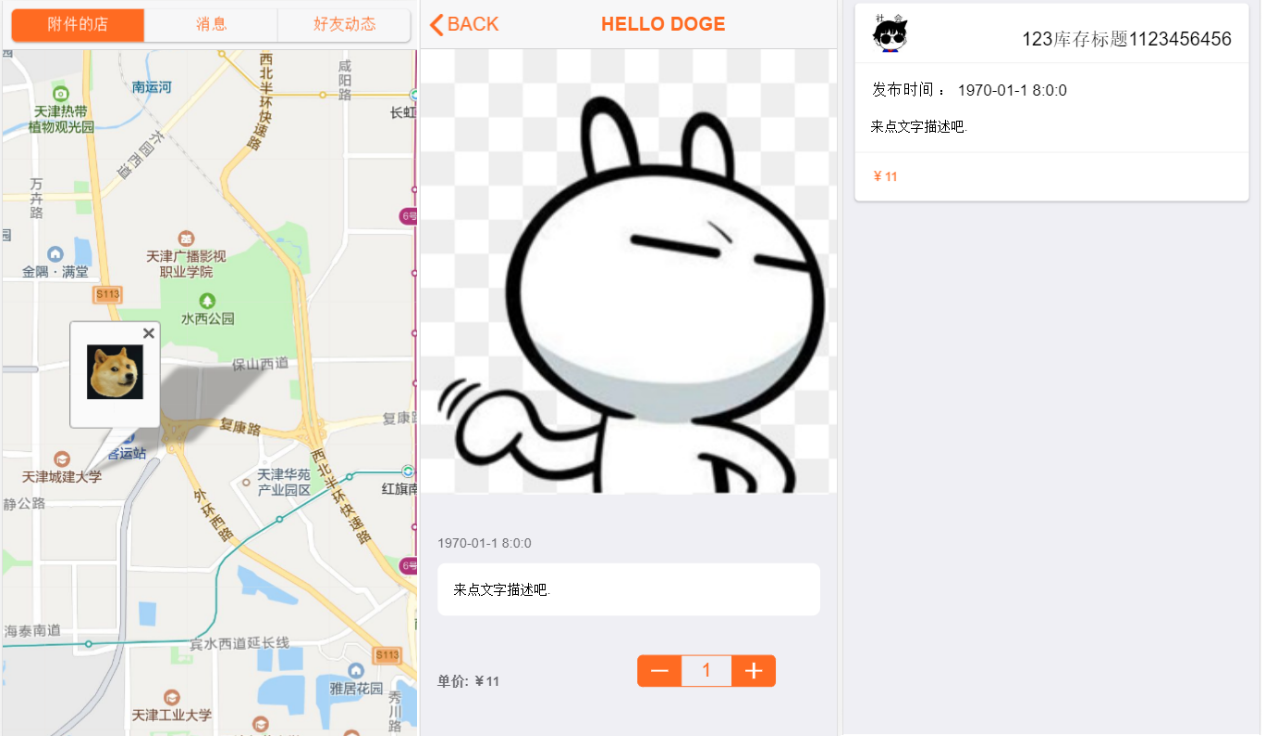


图6-7 用户销售信息查看

### 6.2.3 动态信息发布，查看和评论

图6-9展示的是动态的发布，查看和评论

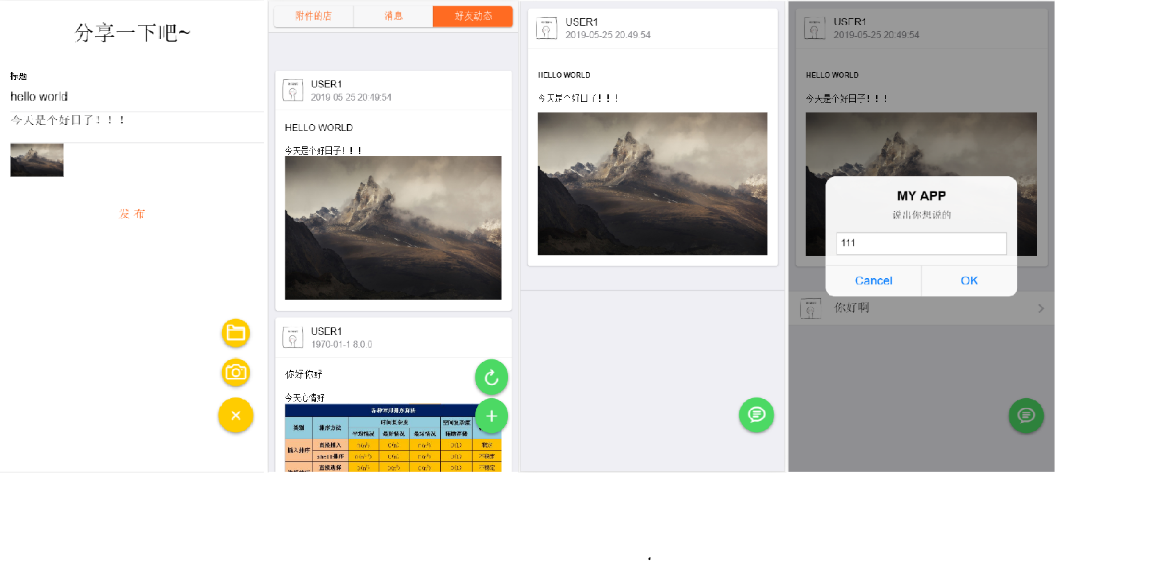


图6-9 用户动态的发布，查看和评论

## 6.3 下单流程实现

### 6.3.1 下单

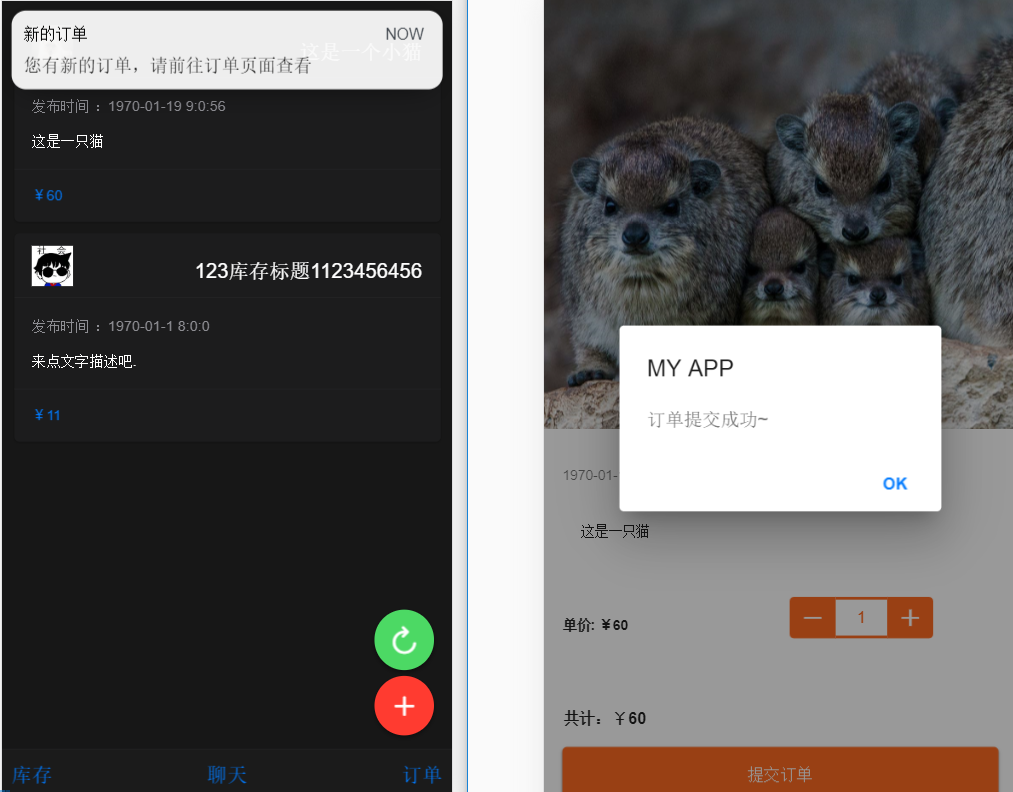


图6-10 下单

### 6.3.2 接单和拣货

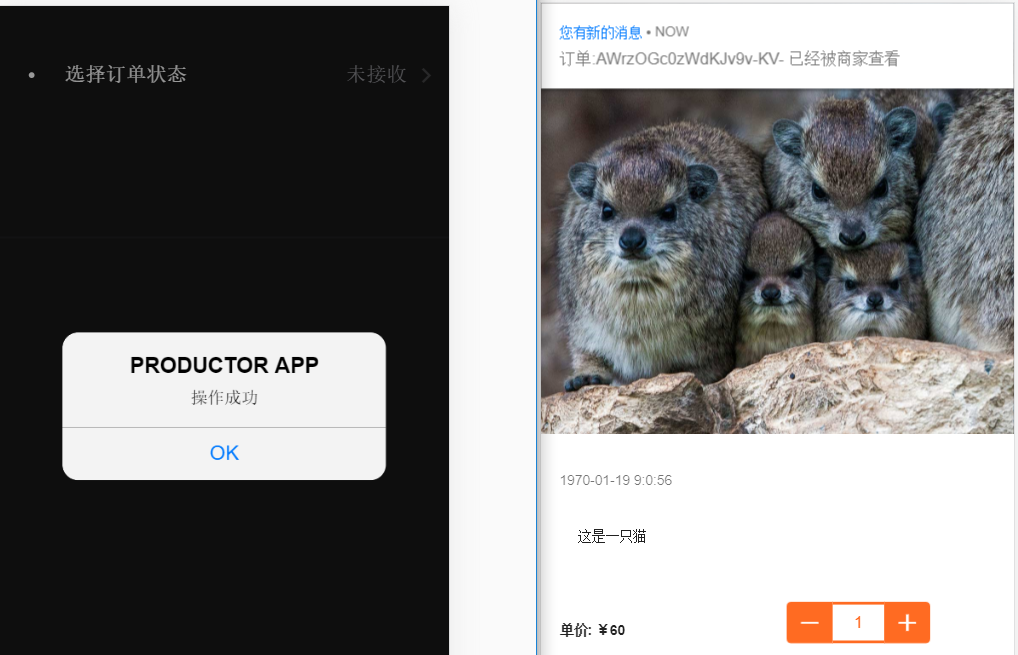


图6-11 接单

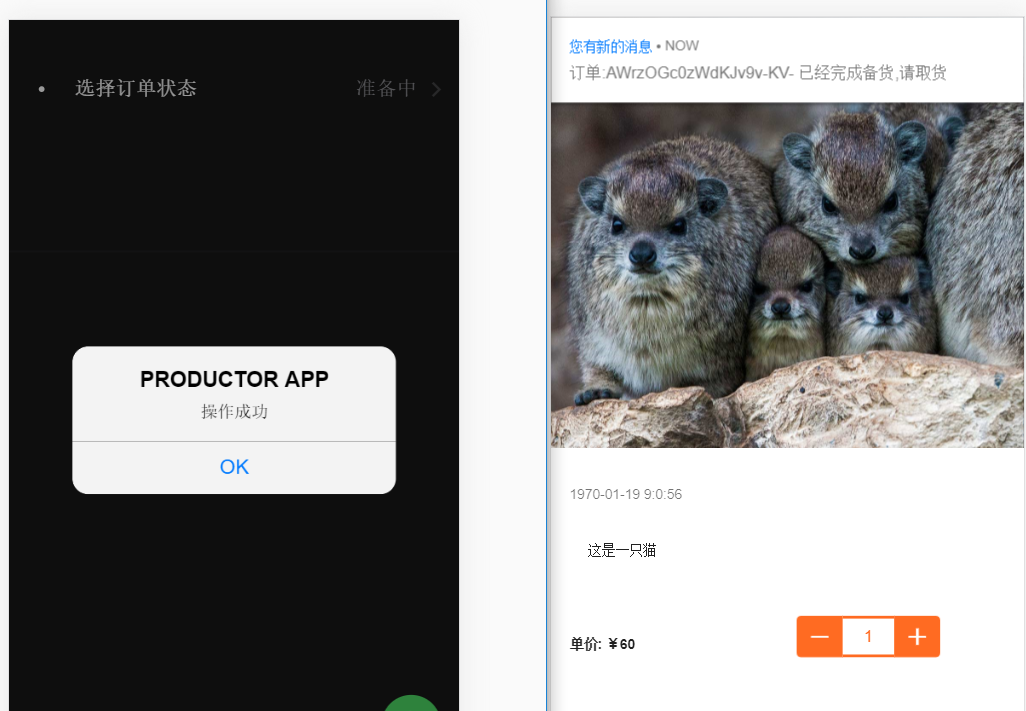
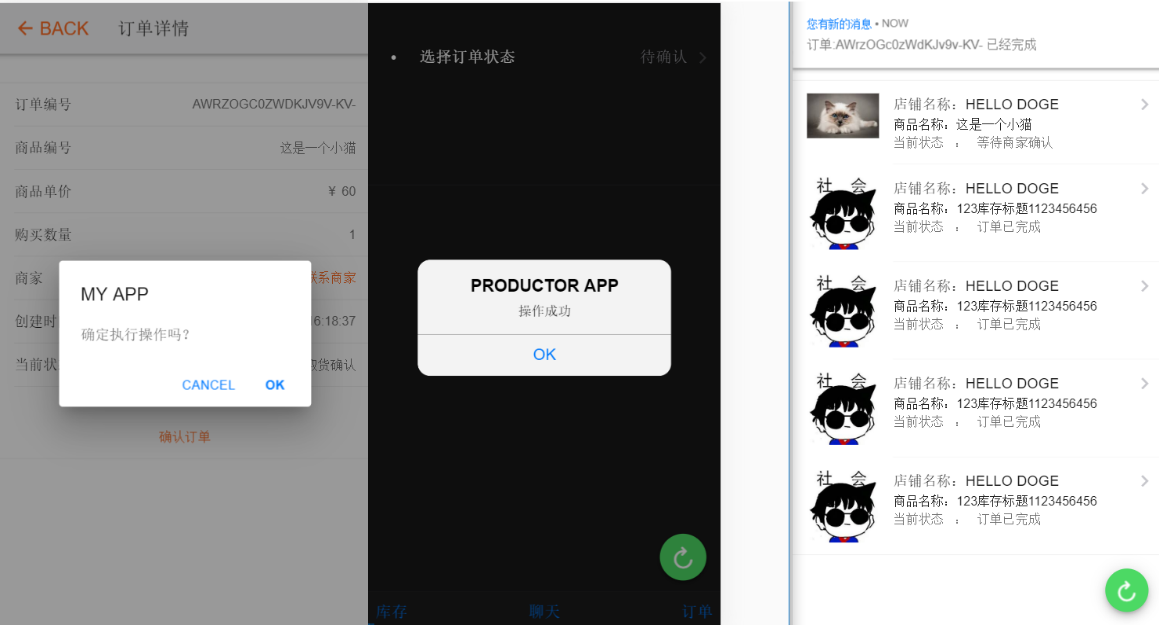


图6-12 拣货完成

### 6.3.3 确认



6-13 订单完成

订单完成需要用户确认，然后商家才可以进行确认操作

## 6.4 聊天功能实现

用户和商家均可在订单详情界面联系对方

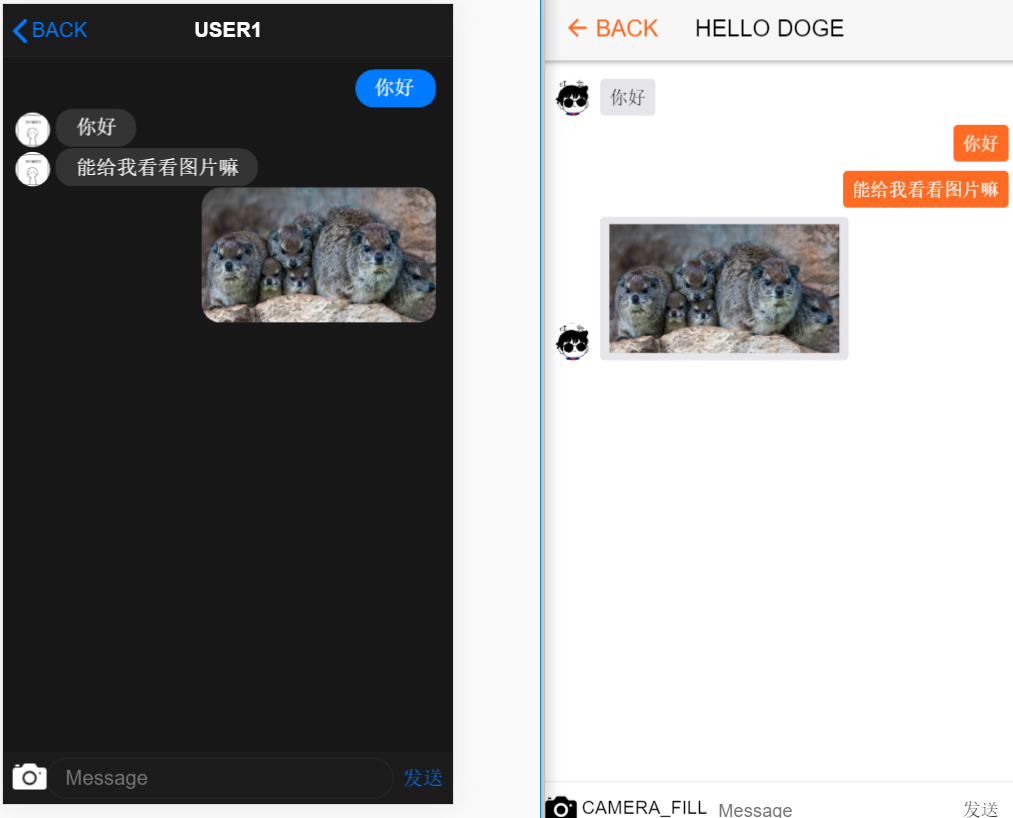


图6-14 聊天页面

总 结

系统的设计终于快告一段落了，通过老师的悉心指导和自己的努力，宠物社交APP的设计与实现完成了它应有的功能。

在商家端，完成了销售信息的发布，利用腾讯地图API进行定位，同时在商家端实现了完整的订单流程，使得商家可以正常的进行订单操作，在商家操作订单的同时也会实时的通知客户

在用户端，可以利用腾讯地图技术获取附件的宠物店销售信息，同时也可以快速的进行下单操作，在商家操作订单时，客户也可以与商家建立聊天关系，方便确定详细需求。同时支持用户实时发布动态，分享自己养宠经验，也支持他人进行评论。

系统的主要难点在于订单在用户系统和商家系统之间的传递，由于Apache Kafka采用流的方式进行传输，而Java对实体类的序列化需要用到包名和序列化id，后来也是查阅了很多资料才定位到问题

毕业设计写完了，其中有很多磕磕绊绊，但自己并没有放弃，同时也暴露了自己知识上的欠缺，自己定会在以后的日子中继续学习，不断完善自己的知识面。

# 致 谢

时间飞逝，不知不觉之中毕业设计也已经完成，大学四年的生活也落下的帷幕。

在这里，我要先感谢我的导师。从开始选题到最终的完成，都离不开导师的悉心指导。在整个过程中，导师不仅在知识层面上给予我莫大帮助，更在为人处世上教会我许多道理，无论是严谨的治学态度还是精益求精的工作作风，都在无时无刻的影响着我。在此，谨向导师表示崇高的敬意和衷心的感谢!

其次，我要感谢我的父母，是他们给了我无限的关爱，是他们不辞劳苦的教导和抚育,大学四年的成长也离不开他们的支持。

最后，感谢和我一起经历大学四年的同窗好友，感谢他们愿意指出我在大学生活中不足，也感谢他们在平时日常生活中给予给我的帮助，同时也感谢四年来我的所有任课老师，正是他们教会我许多专业知识，让我的知识面更加宽广。也正是因为有了他们，我的大学生活才变得充实有意义。

# 参考文献

[1] Mark Pollack. Spring Data实战[M] 北京：人民邮电出版社，2014

[2] 赵新江. 消费升级引爆宠物产业[J]. 理财：市场版, 2018.

[3] 陈雄华. 精通Spring 4.x [M] 电子工业出版社，2017

[4] Fielding, Roy Thomas. Chapter 5: Representational State Transfer (REST). Architectural Styles and the Design of Network-based Software Architectures (Ph.D.)[D]. University of California, Irvine. 2000.