Android 移动应用开发期末项目报告

Pet's Day

组号：第 32 组

小组成员

郭瑶佳 15331093

宋思婷 14970011

谢梓丰 15331329

许基骏 15331338

目录

1 概述

小组项目的名称叫 PetsDay，是一款专注于宠物社交的 App。

在一些社交媒体和视频网站上，有很多类的用户。其中的一类，他们喜欢小动物，爱好通过网络来吸宠。基于这样的情况，一个可以专门用来吸宠的 App，PetsDay，就出现在了我们的 idea 中。

打开我们的 App，将会是一个以宠物为主打的社交圈。所以我们舍弃了用户的一切信息，

2 背景

PetsDay 的设计概念就是宠物社交，基本的设计是“用户关联宠物，宠物关联动态”。

相关语言：Java、Xml、SQL

实现平台：Android Studio、phpMyAdmin

App 的特性：

1. 实用性：这款应用定位在宠物社交，我们主要实现了社交的基础功能，发布动态、点赞、评论、通知以及关注等等。当然相比于成熟的社交 App，我们的功能仍然不完善的，比如我们还无法对评论进行回复。

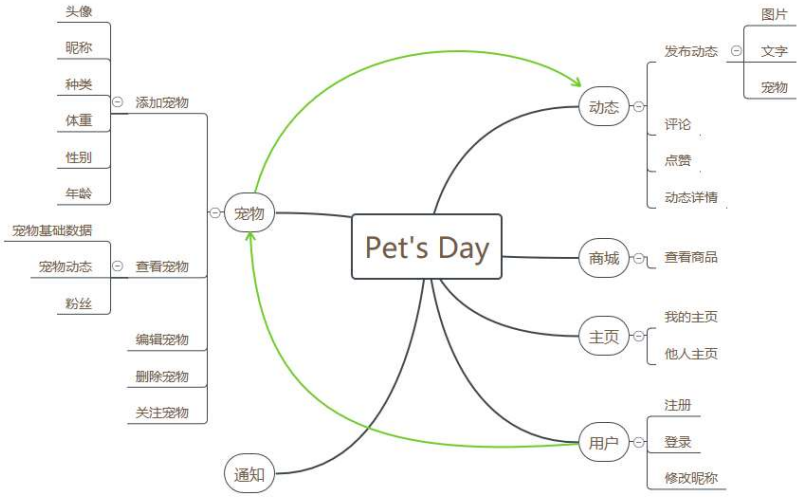
2. 创新性：市场上有挺多吸宠的 App，与他们的 App 相比，我们比较独特的地方是我们的社交主体是宠物，动态是宠物的动态，粉丝也是宠物的粉丝。这可能是我们在设想中一个比较新奇的地方。

3. 用户体验：我们在设计交互的时候，是模仿了一些比较成熟的社交媒体的交互方式和 UI 风格。此外我们还做了图片的缓存和列表的动画。

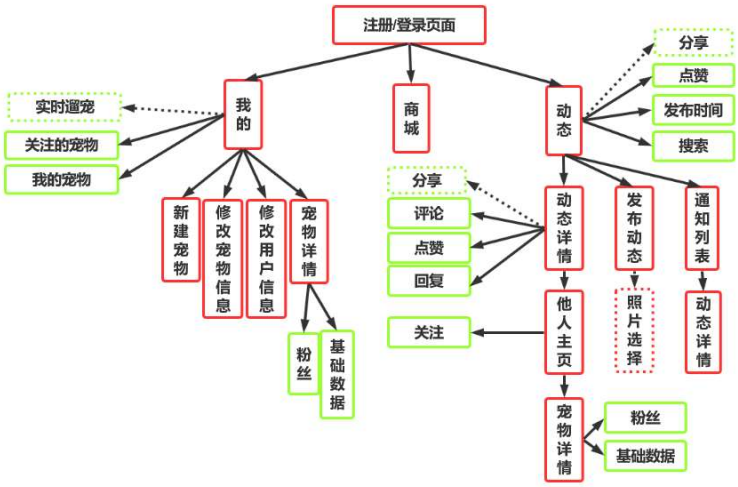
3 项目说明（重点）

3.1 功能模块

以下是我们的思维导图（图一）和功能模块图（图二）



图一 思维导图



图二 功能模块图

主要的功能模块说明：

①. 宠物

i. 添加宠物

ii. 查看宠物

iii. 关注宠物

iv. 修改宠物信息

②. 用户

i. 注册

ii. 登录

iii. 修改昵称

③. 动态

i. 发布动态

ii. 点赞动态

④. 评论

i. 评论动态

⑤. 通知

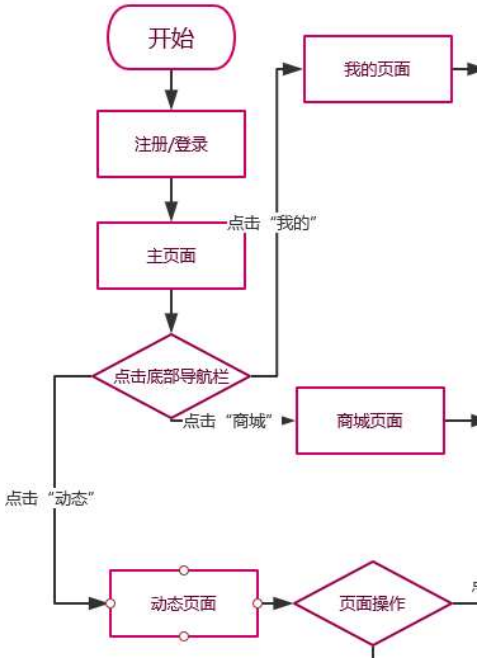
查看通知（包括已读和未读）

3.2 逻辑流程

流程图中，红色代表点击之后跳转的页面，绿色代表功能。

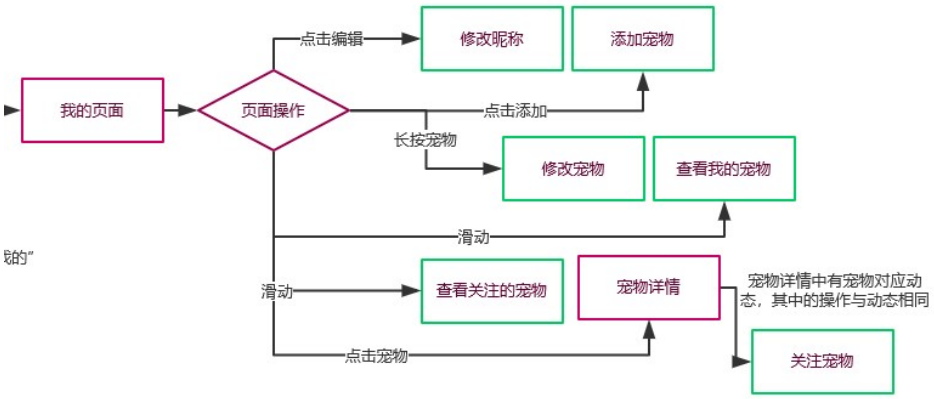
逻辑流程图（图三）

用户打开 APP，需要注册或者登陆，之后会记住状态，无需再登陆，然后点击导航栏，可分别跳转的动态、我的和商城页面



图三 逻辑流程图一

逻辑流程图（图四）说明了我的页面的交互和功能

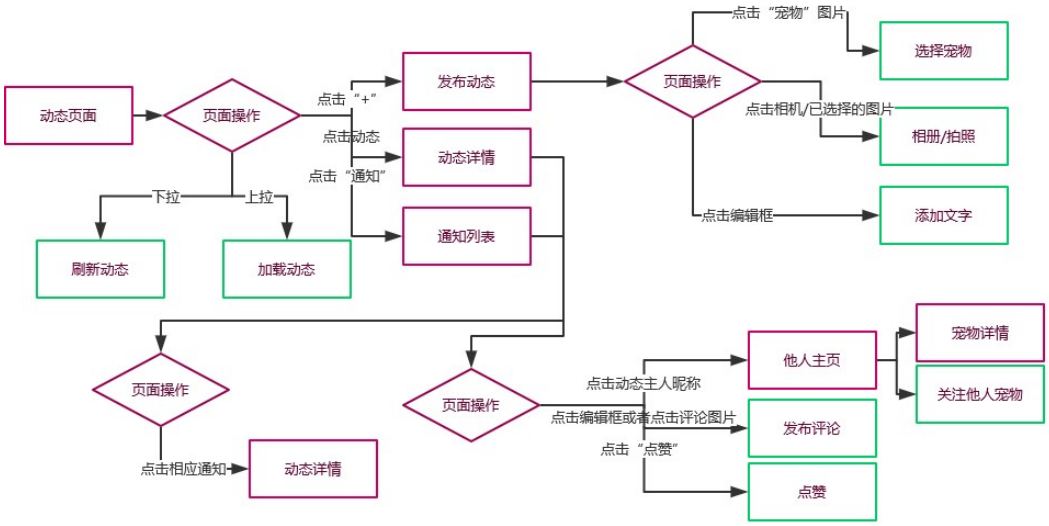


图四 我的页面的逻辑流程

逻辑流程图（图五）说明了商城的功能

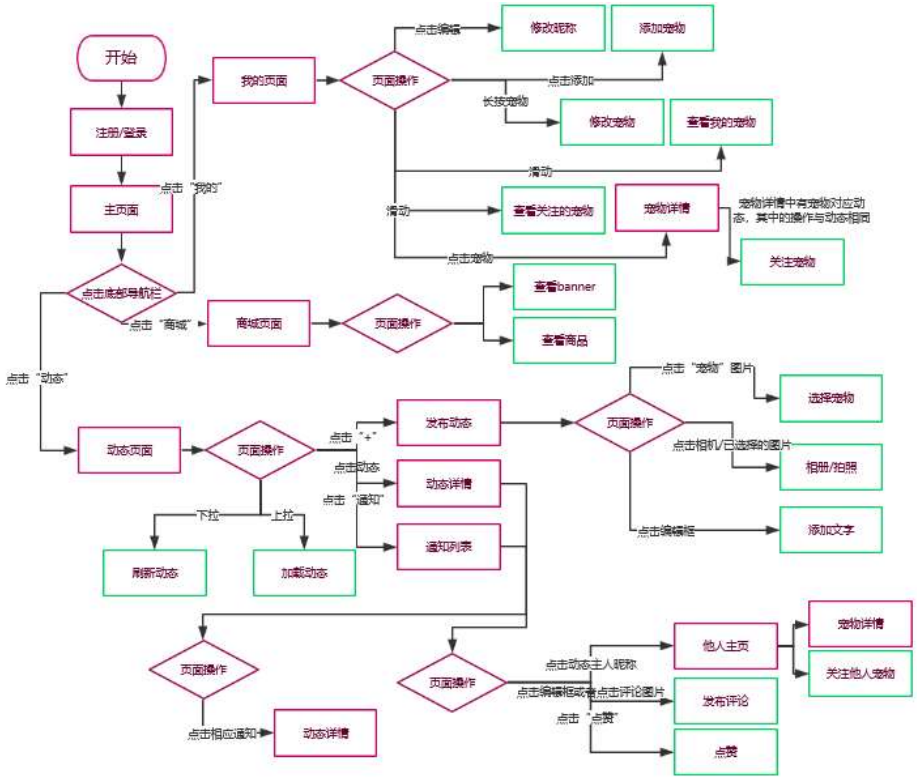
图五 商城页面的逻辑流程

逻辑流程图（图六）说明了动态页面的功能



图六 动态页面的逻辑流程图

逻辑流程图（图七）展示了所有的流程



图七 联结所有流程

3.3 技术说明 (重点)

详细说明应用使用到的技术：数据库、云服务器、高级界面等等，最好画一个技术结构图说明。总体概述各部分的功能，接着详细说明具体的实现方法，重点突出小组中使用到的特别的技术、游戏算法等。

1、 服务端

① 搭建图片服务器：使用 node.js 和 express 框架快速搭建轻量级的 HTTP 服务器，使用在 NPM 托管的文件上传处理辅助组件 formidable 来对上传的图片进行解析，并存储到服务器文件目录中；提供图片下载接口以供客户端调用。

客户端则使用 Android 中生成 HTTP 请求体的相关接口来生成图像上传请求体，二进制的图像数据会被转换成表单数据 multiPart。

② 搭建网络请求的服务端：使用 Koa2 的 Node.js 框架搭建服务端，搭建在阿里云的 centos 服务器，开放 3000 端口。

这是目前 web 领域当中社区比较活跃的 Node.js 框架，学习成本较低，搭建服务端比较方便。使用的 node 版本是 v8 以上，能够使用 await 和 async 解决数据请求当中的异步问题，而且相应的各个中间件之间也能做到同步管理，与使用的数据库 MySQL 集成较好。因此在项目当中打算使用 Koa2 搭建的服务端当作客户端与数据库之间交互的中间接口，处理用户的请求并相应对数据库进行操作。

以下代码当是服务端当中针对客户端请求的不同方法进行处理，对应相关的增删改查的操作。（图八）

```
async function handleRequest(ctx) {

  if (ctx.request.method === 'GET') {
    ctx.body = await selectItems(ctx.request.url.split('?')[0].slice(1),
    ctx.request.query);

  } else if (ctx.request.method === 'POST') {
  console.log(ctx.request);
    ctx.body = await insertItems(ctx.request.url.slice(1),
    ctx.request.body);

  } else if (ctx.request.method === 'PUT') {
    ctx.body = await updateItems(ctx.request.url.slice(1),
    ctx.request.body);

  } else if (ctx.request.method === 'DELETE') {
    ctx.body = await deleteItems(ctx.request.url.split('?')[0].slice(1),
    ctx.request.query);
  }
}
```

图八 网络请求服务端操作

2、 数据库

数据库使用的 MySQL 这个关系型数据库，并使用 phpMyAdmin 进行数据库可视化操作的。

①. 云服务器

使用 phpMyadmin 云服务器搭建数据库，辅助开发。

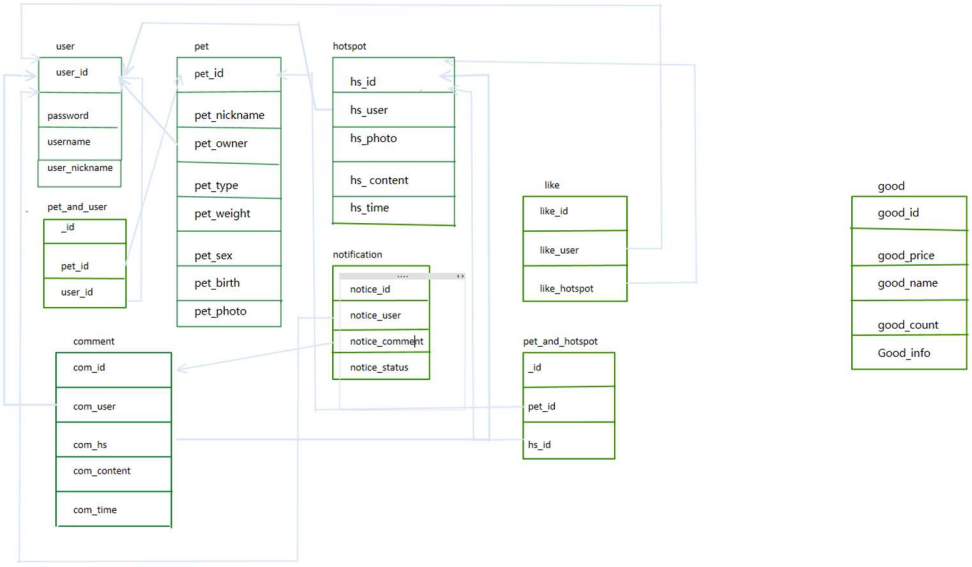
本次项目当中，使用的是一个线上部署的数据库，在开发过程当中并没有使用本地的数据库，而是直接使用线上的数据库，因为某些功能模块耦合程度较大，开发过程中使用相同的数据也是比较重要的能够避免很多错误。同时，客户端在进行数据库操作的时候，能够通过这个平台查看到数据库相关的操作结果，同时也能够对服务端返回的结果进行判断，同时还能够通过图形化界面对数据库进行操作，插入所需要的实际数据，方便调试。界面如下（图九）

图九 phpMyAdmin 界面

搭建服务器的数据库系统和 apache 服务器处理网络请求，之后搭建 phpMyAdmin 所需要的环境，包括 PHP 相应的版本，使其能够连接上服务器本地的数据库，并进行相应的防火墙设置，使得用户能够 ip/phpMyAdmin 的方式通过浏览器访问数据库。

②. MySQL

本次项目涉及到的主体有人物，宠物，评论，通知等等，相关关系是比较复杂的，因此采用关系型数据库进行数据的存储是比较合理的，相关的数据库的表和字段，以及外键约束关系见下图（图十）。



图十 关系表

对于相关的表中的数据，需要进行数据库表格初始化并插入初始化数据，同时完善相关表格相应的增删改查的操作。同时根据项目当中的要求，通过各种拼表和聚合函数实现复杂的查询请求。

以下是根据功能点实现的相关的数据库查询的请求（图十一）。

查看动态详情里面的相关评论

```
curl -X GET 120.78.169.206:3000/comment\?hs_id=2
```

根据用户id查看拥有的宠物(**增加了count字段表示粉丝数**)

- 查看用户拥有的宠物(status=1)
- 查看用户关注的宠物(status=0)

```
curl -X GET 120.78.169.206:3000/pet\?user_id=1&&status=1
curl -X GET 120.78.169.206:3000/pet\?user_id=1&&status=0
```

查看相关动态关联的宠物(**增加了count字段表示粉丝数**)

```
curl -X GET 120.78.169.206:3000/pet\?hs_id=1
```

看看用户通知(相关评论内容也可以看到)

```
curl -X GET 120.78.169.206:3000/notification\?user_id=1
```

查看宠物的粉丝

```
curl -X GET 120.78.169.206:3000/pet\?pet_id=1
```

查看相关用户点赞的条目

```
curl -X GET 120.78.169.206:3000/like\?user_id=1
```

图十一 相关数据查询入口

3 项目说明（重点）

我们实现的内容如下。

1) 用户

①. 注册

②. 登录，可记住登录状态

③. 修改昵称

2) 动态

①. 增：发布动态

②. 获取：根据用户不同操作，获取动态

i. 打开 app 获取最新动态

ii. 下拉刷新获取最新动态

iii. 上拉加载获取之前的动态

iv. 查看宠物详情同时获取其对应的所有动态

3) 点赞

①. 增：用户进入某条动态的时候，可以进行点赞和取消点赞

4) 评论：

①. 增：用户可以在他人动态详情下发表评论

②. 获取：每当用户进入某一条动态时，会根据动态来获取对应评论

5) 通知：

①. 获取：当用户发布的动态收到评论时，他会收到通知。查看后，通知的状态由未读变成已读。

6) 宠物：

①. 增：新建宠物（插入到宠物表中）

②. 删：删除宠物（并且删除对应的动态和评论，忘记有没有保留了。）

③. 改：修改宠物的信息

④. 获取：

i. 查看他人主页时，根据用户名获取宠物

ii. 查看动态详情，根据动态 id 获取对应的宠物

7) 商城

①. 查：获取商品

3.4 代码架构

Global：

AppContext 类：这是继承于 Application 类的一个应用全局类，类包含一些全局使用的数据还有接口，比如各个页面的列表数据，请求后端服务的全局 Service 单例，还有一些更新全局数据的接口。

Utility

1. 网络操作服务类：用于与后端进行数据交互，包括表单上传，对象的 CRUD 还有图片的上传。这里使用 Observable 来进行异步的请求。
2. 网络操作服务类工厂类：用于生成网络操作服务类的实例，这里使用 OkHTTP 和 Retrofit 来生成网络操作类实例。
3. 生成图像上传请求体操作类：使用 Android 中生成 HTTP 请求体的相关接口来生成图像上传请求体，二进制的图像数据会被转换成表单数据 multiPart。
4. 添加网络操作返回状态类：这个类可以作为网络操作的返回数据，并且将返回的 json 数据含义进行了分析，提供了获取返回状态的接口，等于说将分析相同结构的返回内容的逻辑集中到此处，实现充分的解耦。

Model

1. Model：实现了宠物、动态、点赞、评论、通知等的 Model 类。

View

1. Adapter：共实现了 10 个 Adapter 类，其中一部分继承了 BaseQuickAdapter 这个类，减少了代码量。主要是用于绑定数据和 UI，添加监听或子控件的监听。
2. Acitivty：实现多个页面的逻辑。

页面逻辑：

i. 登录状态检查：对存储在本地的登录状态信息进行检查，如果没有登录则跳转到登录界面

ii. 登录注册逻辑：分为登录和注册 2 个页面。页面上提交填写的表单的时候实现内容合法性检测；将表单上传给后端服务器之后根据返回的信息进行判断，操作失败会向用户提示错误原因，操作成功则进

行下一步：登录成功则存储登录信息，进入应用主页面，加载主页面信息；注册成功则返回到登录页面。

iii. 用户添加新宠物页面：页面为一个表单，包含宠物的各项信息：头像、名字、类型、性别、体重和生日，同样的用户提交表单的时候会进行合法性检查，对服务器返回的操作结果进行解析，如果添加成功会返回主页面，操作失败会提示错误原因。

注：用户修改宠物信息也是同样的界面和逻辑。

iv. 新建动态页面逻辑：项目中的“动态”；类似于微博，每一条动态包含文字、图片还有关联的宠物。新建动态的时候用户需要填写文字内容，选择图片，并且选择动态关联的宠物，提交的时候会进行合法性检测，提交成功后返回主页，并刷新主页。

v. 动态详情页逻辑：点击主页面的动态条目会跳转到动态详情页面，动态详情页面会显示动态的文字内容、图片、关联宠物、宠物主人信息、动态发布时间。并且我们的动态具有评论功能，在动态详情页可以添加对动态的评论内容，添加后实时刷新评论列表数据。同时还可以为动态点赞、取消点赞，每一次进入动态详情页面会显示之前的点赞还有评论信息。

vi. 他人主页的逻辑：他人主页上主要展示了对应用户的昵称及其宠物。在宠物条目上的子控件添加了监听，点击宠物条目上的关注，可以对宠物进行关注和取消关注。另外是，点击宠物条目可以跳转进宠物详情。

vii. 通知页面的逻辑：当其他用户评论你的动态时，你便会收到通知。每次当动态刷新的时候便或者最新的通知，设为未读。并且将未读的通知数目显示在动态主页的通知 icon 上。通知条目主要展现的是其他用户的昵称、评论内容和评论时间。当用户阅读之后，通知的状态设为已读，并且点击通知条目可以跳转到相应的动态详情进行查看。

viii. 宠物详情页逻辑：宠物详情页中包含宠物的基础数据，包括其头像、体重、昵称和出生年月等等数据；还包含其粉丝数。其他用户可以对其进行关注和取消关注，获取对应的宠物动态，当然，显示出来的宠物动态也是可以进行点赞和评论的。

3. Fragment：实现主页面中的三个 Fragment，分别是 HotSpotFragment，MarketFragment 和 MineFragment。

i. 主页动态展示最新的动态，属于热点区，动态展示了发布动态的用户的昵称，发布时间，该条动态的点赞数和评论数。同时，主页动态存在多个功能入口，如点击动态条目进入动态详情，点击通知可以查看最新通知，点击发布可以发布动态（前提是拥有宠物）等等。

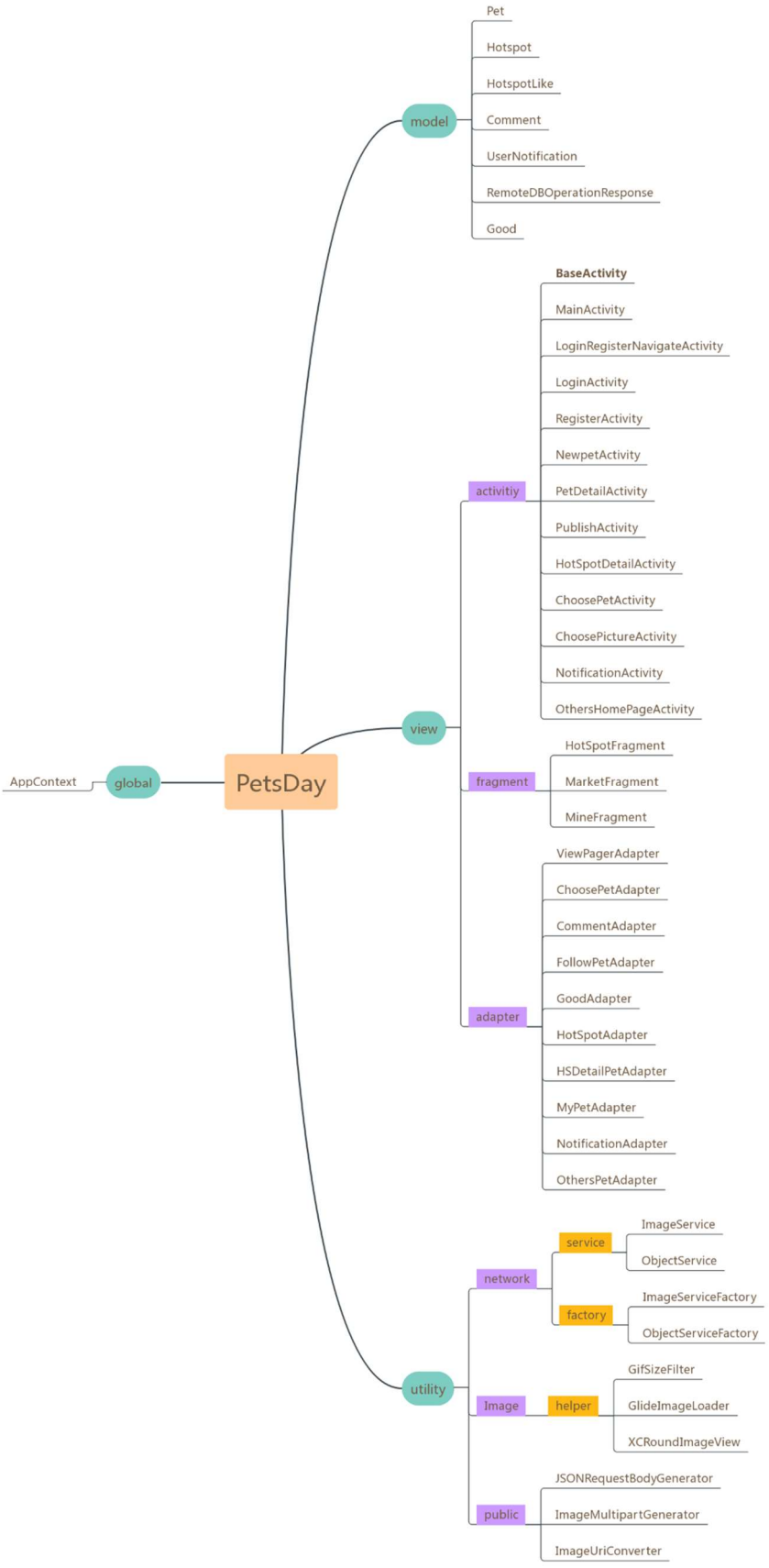
ii. 我的页面的逻辑。属于用户自己的信息页面。主要分用户的宠物和其关注的宠物两个列表，宠物条目包含其头像、昵称、种类以及关注人数等信息。点击可以查看宠物，进行关注；还可以进行用户昵称的修改、添加宠物等等操作。

优化处理：

①. 对图片的加载做简单的缓存处理：项目中大量使用了第三方图片加载工具 Glide，可以从本地还有服务器获取图片数据显示在页面指定的 ImageView 上。为了减少网络请求的次数，提升使用流畅性，图片加载后会进行缓存操作，并且每次重新加载图片的时候会先判断手机本地是否已经存在了所要的图片，如果有则优先加载本地图片，否则再从服务器获取。

②. 拉取开源控件，包括 RecyclerView 的下拉刷新控件、通知的提示小红点控件、图片选择器的控件来完善 App 功能。同时使用了 BaseQuickAdapter 来自定义 RecyclerView 的 Adapter，在减少代码量的同时，优化了视觉效果，添加了流畅了动画。另外是使用 LineChartView、ScrollBar 等控件来改善 UI 布局。

③. 用户的操作有文字说明或者操作之后出现相应的 Toast 提示，提高 App 的易用性。


PetsDay
global
AppContext
model
Pet
Hotspot
HotspotLike
Comment
UserNotification
RemoteDBOperationResponse
Good
view
activitiy
BaseActivity
MainActivity
LoginRegisterNavigateActivity
LoginActivity
RegisterActivity
NewpetActivity
PetDetailActivity
PublishActivity
HotSpotDetailActivity
ChoosePetActivity
ChoosePictureActivity
NotificationActivity
OthersHomePageActivity
fragment
HotSpotFragment
MarketFragment
MineFragment
adapter
ViewPagerAdapter
ChoosePetAdapter
CommentAdapter
FollowPetAdapter
GoodAdapter
HotSpotAdapter
HSDetailPetAdapter
MyPetAdapter
NotificationAdapter
OthersPetAdapter
utility
network
service
ImageService
ObjectService
factory
ImageServiceFactory
ObjectServiceFactory
Image
helper
GifSizeFilter
GlideImageLoader
XCRoundImageView
public
JSONRequestBodyGenerator
ImageMultipartGenerator
ImageUriConverter

4 应用效果

1) 首先是用户的登录和注册。注册页面对用户名有查重的检测

登录/注册的选择页

注册时用户名查重

注册页

登录

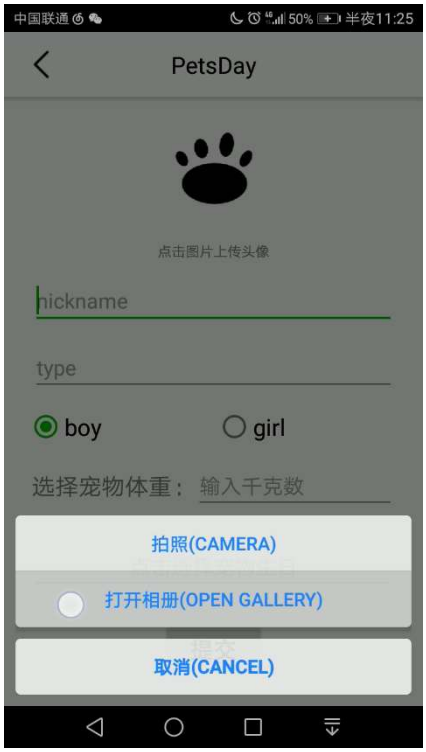
2） 三个主要页面

动态

商城

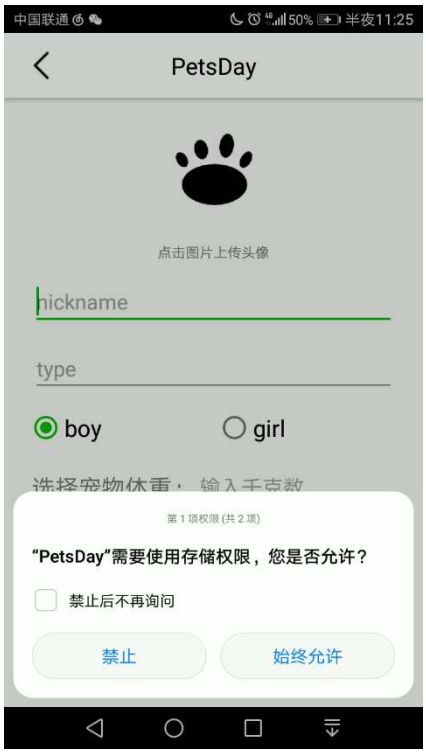
我的

3） 给我的页面添加宠物，最后会实时更新在我的宠物中。

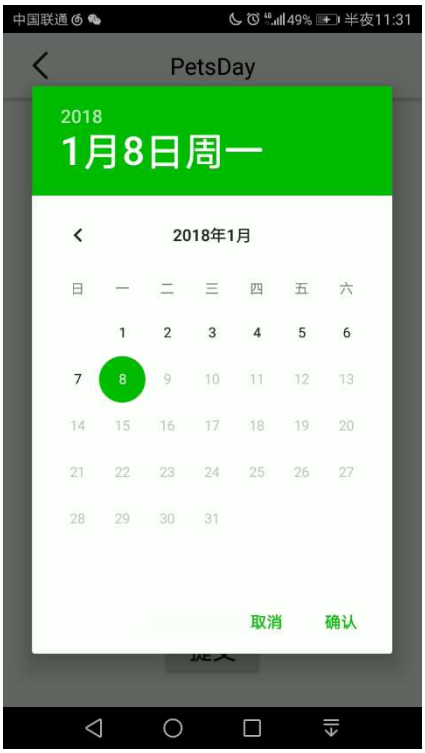


点击头像可以选择相册或者拍照

选中图片，然后进行裁剪



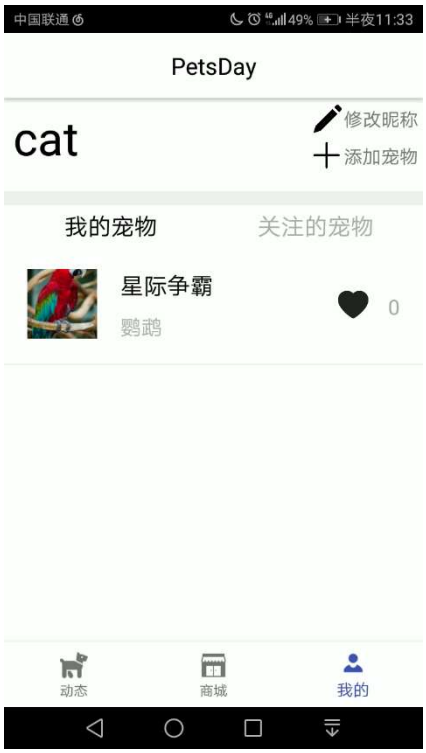
申请权限



选择宠物的出生年月日

添加宠物的页面有合法性检测

点击宠物，进去宠物详情

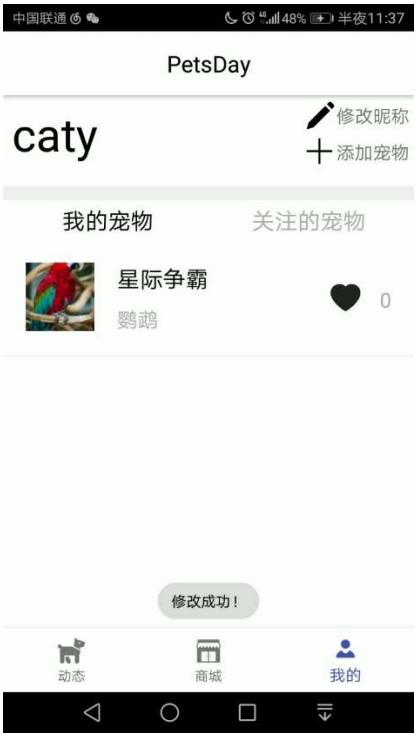


实时更新在我的页面中

宠物详情中包含宠物的基础数据，以及它的粉丝数和对应的动态，因为是新的宠物，所以还没有粉丝和动态。（中间部分是遛宠的功能，但没有真正实现）

4） 修改用户的昵称

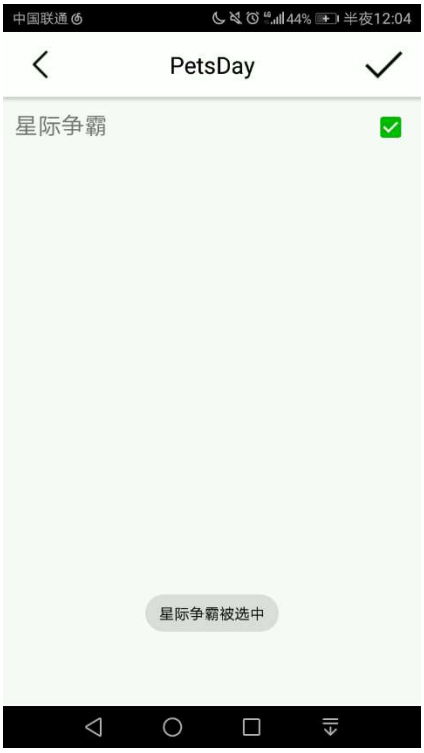
修改用户昵称



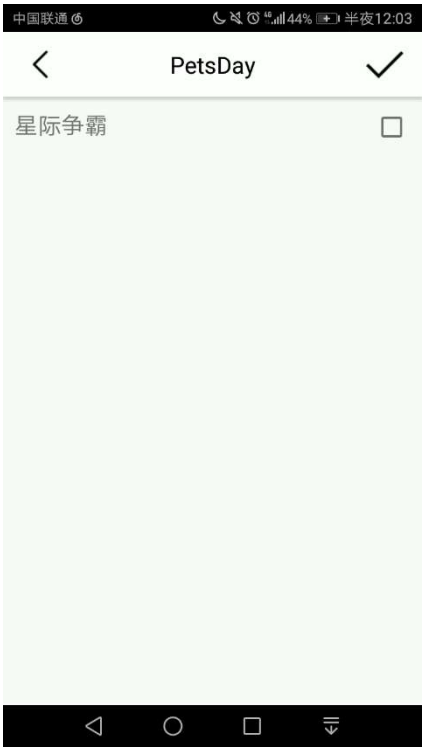
修改成功

5） 动态页面中，发布动态

发布界面

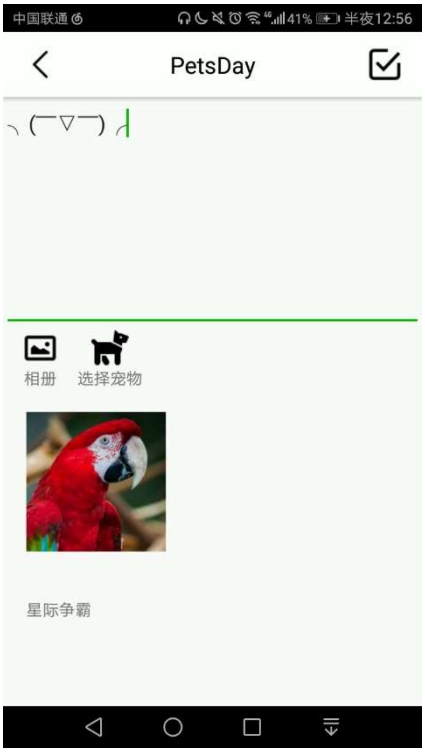


选中宠物，这里可以单选可以多选



选择宠物界面

点击发布 跳转回到动态详情并实时更新



发布动态的页面

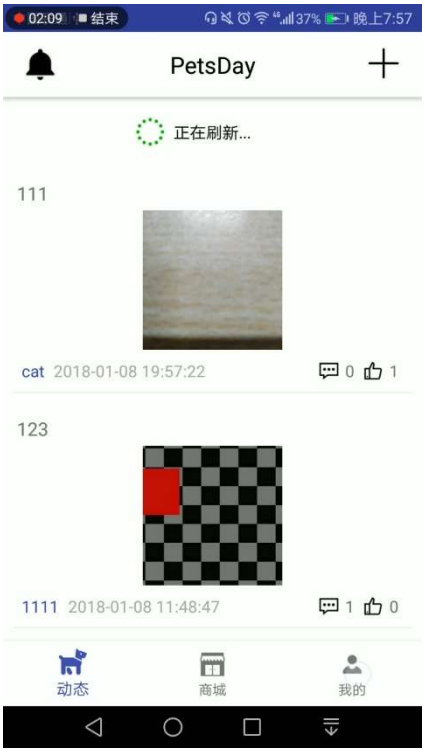
6） 点赞

点击动态项进入动态详情

实时更新在动态列表中，显示点赞数为 1

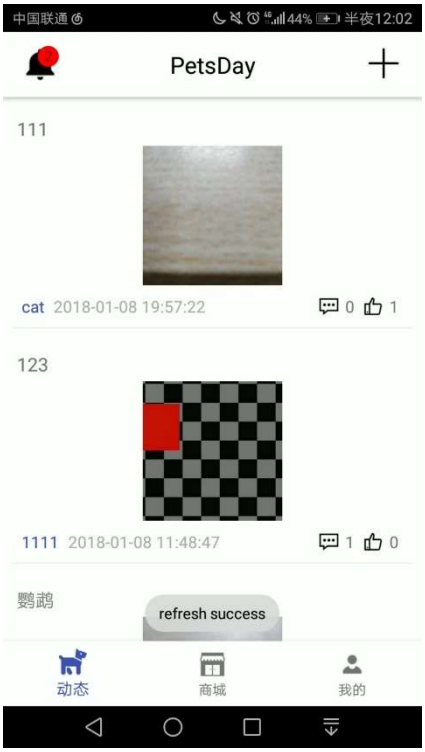
然后点赞

7） 动态实现下拉刷新



下拉刷新

点击通知进行查看，查看之后，通知变已读



刷新完毕，没有新的动态但有新的两条通知

点击通知可进入相应动态详情查看

8） 上拉加载 && 发布

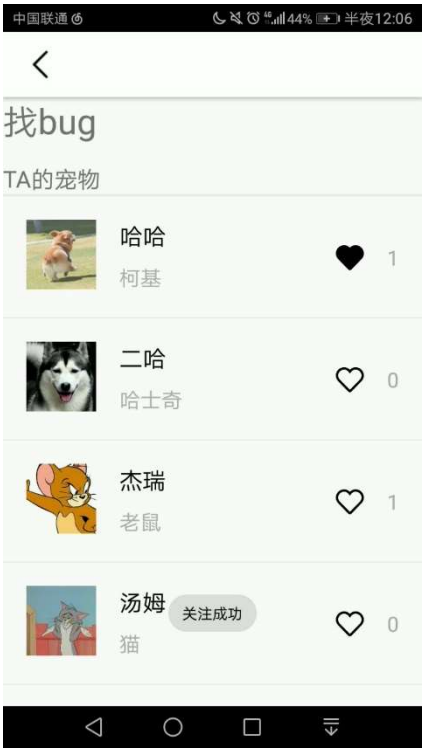
加载

发布评论

9） 点击动态详情中的用户昵称跳转至他人主页，并且可以关注和取消关注宠物

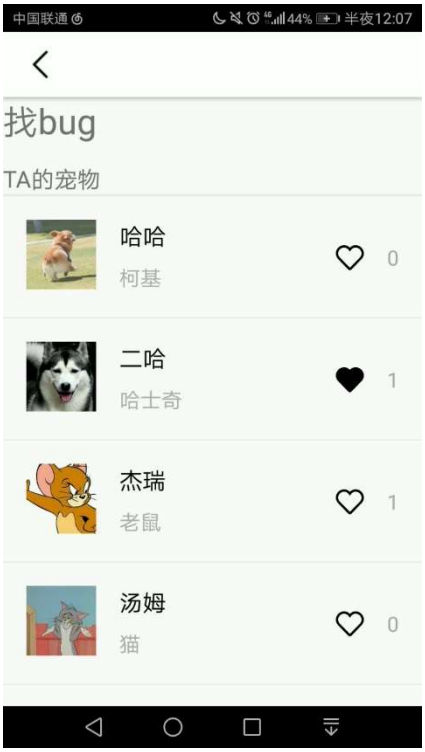
他人主页，实心说明已关注，空心是未关注

点击宠物进其动态详情，因为已经关注该宠物，显示实心的 Icon，更新人数



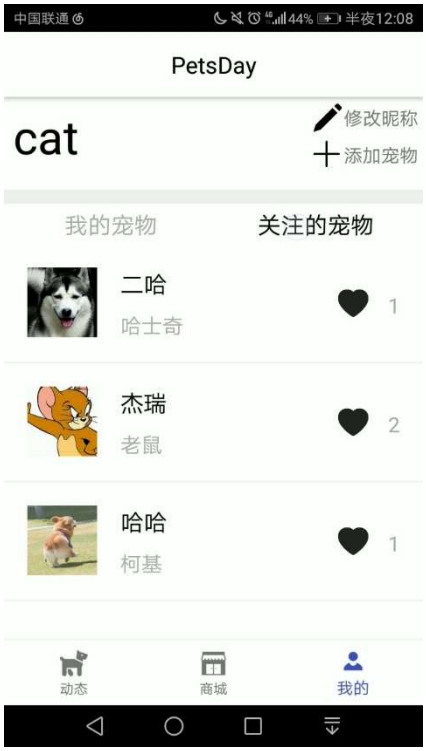
点击关注

在宠物详情页可以取消关注

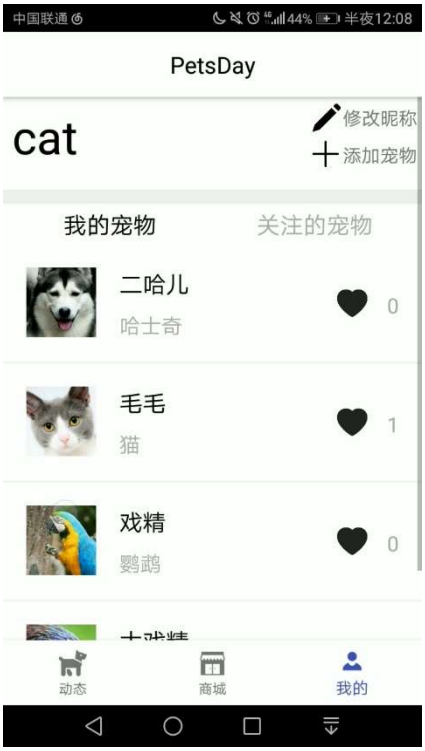


返回的时候，已经取消关注了，第二项是实心，是因为之前点击过关注

然后补充下我的页面中，我的宠物和关注宠物的切换



关注的宠物



我自己的宠物

5 开发过程中遇到的问题及解决办法

- 问题一：首次使用 fragment，其机制我们比较陌生。

  解决方案：在 fragment 中写代码逻辑的时候，一些能在 Activity 中使用的方法，会不能直接使用。因为 fragment 是 activity 的一部分，会有所不同。同时不能使用 OnActivityResult 函数，因为返回的请求码很奇怪。

- 问题二：异步操作

- Adapter 中加载网络图片无法显示

  解决方案：由于网络图片是异步加载，通常加载结束时已经完成了列表渲染，因此最初的写法导致列表中所有图片都无法显示，采用 Glide 加载库解决了问题，同时可以在图片加载未完成和加载失败时分别显示默认图片，用户体验更好

- 网络请求后，列表没有出现预想的更新

  解决方案：①在这次项目中大量使用了网络访问，主页上的动态列表，个人页面中的宠物列表还有通知列表都需要向服务端请求数据，而使用观察者模式本身就是异步操作的一种。许多对页面的更新操作需要在网络请求的回调中进行。而我们平时接触异步编程还是相对较少，项目中有不少地方一开始为了预留接口而采取了同步的编写方式，后面将网络请求接上去之后就频频出现奇怪的问题，最常见的就是页面没有出现预想的更新。经过一番 debug 之后才发现在先前的设计中没有考虑异步的问题。所以在发现问题之后进行后续的开发过程中才特别注意到这个问题，预留接口的时候会考虑设计成网络接口调用与请求回调函数结合的设计方式。②页面设计复杂，单步调试后发现，已经获取到数据，但是没有预想的更新，原来是 RecyclerView 被遮挡了，可以使用 Dom 树查看结点是否添加了，然后对布局进行调整，使 RecyclerView 有一定的空间。

- 问题三：部分页面（比如宠物详情）可能有多个入口导向

  解决方案：由于最初页面间逻辑设计不到位，页面之间采用 intent 传递数据不统一，反复报错跳出，修改跳转页面之间的 intent 和 bundle 解决。但这个问题警示我们，今后做项目设计时应该首先考虑完备需求，设计各个页面的功能和跳转逻辑，之后才能有条不紊的进行开发，事半功倍。

- 问题四：图片上传

解决方案：基本没有接触到文件上传下载的知识。查阅了很多相关的资料，才了解清楚服务端逻辑的设计，进行了服务端还有浏览器端的测试后才搞定了服务端的文件上传逻辑。这里面的关键是要将图片转成 form 表单的内容进行上传，服务端再使用第三方的文件上传处理组件来处理。安卓客户端也是同理，需要将图片转成 multiPart 填入 form 表单中，再将数据封装到 HTTP 请求报文中进行上传。

- 问题五：原生的 RecyclerView 高度无法自适应

  解决方案：我们这次的 APP 使用了大量的 RecyclerView，又因为页面设计比较复杂，有些 页面设计出来之后，RecyclerView 就被限制在一个比较尴尬的高度。 我就去查了一下是什么原因造成的，发现是原生的 LayoutManager 高度设置是一个 列表项的高度，根据查到的解决方法，自定义了一个派生的 LayoutManager 类。然后将整个布局封装在 ScrollBar 中，改善了布局。

- 问题六：数据库的复杂查询

  解决方案：项目当中相关的表较多，基本是通过 id 去设定外键和约束关系，为了服务端返回的数据字段的要求，常常需要进行几张表复杂的 join 操作。例如查询一条动态，需要根据动态的`id`拼上查询评论表和点赞表，并使用聚合函数`count`计算总数，如果根据宠物信息查找动态，还要根据宠物的 pet_id 拼上宠物表，并返回相应信息。从数据库规范来讲这种操作是比较合理的，在数据量较大的时候仍然能够维持较快的查询速度。但相关的 sql 比较复杂，需要花时间去构思这些语句，也复习了一遍数据库相关开发规范和查询优化的内容。

  下面是相关数据库查询当中比较复杂的 sql 语句。（见图十二）

```
if (data.page != null) {
  sql = `SELECT hs_time, hs_user, user_nickname, hs_content, hs_photo,
  hs_id, ifnull(countLike, 0) AS countLike, ifnull(countComment, 0) AS
  countComment FROM \`hotspot\` LEFT JOIN (SELECT like_hotspot,
  COUNT(like_hotspot) AS countLike FROM \`like\` GROUP BY like_hotspot)
  AS count_table ON count_table.like_hotspot=hotspot.hs_id LEFT JOIN
  (SELECT com_hs, COUNT(com_hs) AS countComment FROM \`comment\` GROUP
  BY com_hs) AS count_comment_table ON
  count_comment_table.com_hs=hotspot.hs_id LEFT JOIN user ON
  user.user_id=hs_user WHERE hs_id <= (SELECT COUNT(*) FROM
  \`hotspot\`)- ${data.page}*20 ORDER BY hs_id DESC LIMIT 20`;
} else if (data.pet_id != null) {
  sql = `SELECT hs_time, hs_user, user_nickname, hs_content, hs_photo,
  hotspot.hs_id, pet_id, ifnull(countLike, 0) AS countLike,
  ifnull(countComment, 0) AS countComment FROM \`hotspot\` LEFT JOIN
  pet_and_hotspot ON hotspot.hs_id=pet_and_hotspot.hs_id  LEFT JOIN
  (SELECT like_hotspot, COUNT(like_hotspot) AS countLike FROM \`like\`
  GROUP BY like_hotspot) AS count_table ON
  count_table.like_hotspot=hotspot.hs_id LEFT JOIN (SELECT com_hs,
  COUNT(com_hs) AS countComment FROM \`comment\` GROUP BY com_hs) AS
  count_comment_table ON count_comment_table.com_hs=hotspot.hs_id LEFT
  JOIN user ON user.user_id=hs_user WHERE pet_id = ${data.pet_id}`;
} else if (data.hs_id != null) {
  sql = `SELECT hs_time, hs_user, user_nickname, hs_content, hs_photo,
  hotspot.hs_id, ifnull(countLike, 0) AS countLike ,
  ifnull(countComment, 0) AS countComment FROM \`hotspot\`  LEFT JOIN
  (SELECT like_hotspot, COUNT(like_hotspot) AS countLike FROM \`like\`
  GROUP BY like_hotspot) AS count_table ON
  count_table.like_hotspot=hotspot.hs_id LEFT JOIN (SELECT com_hs,
  COUNT(com_hs) AS countComment FROM \`comment\` GROUP BY com_hs) AS
  count_comment_table ON count_comment_table.com_hs=hotspot.hs_id LEFT
  JOIN user ON user.user_id=hs_user WHERE hs_id = ${data.hs_id}`;
} else {
  sql = `SELECT hs_time, hs_user, user_nickname, hs_content, hs_photo,
  hs_id, ifnull(countLike, 0) AS countLike , ifnull(countComment, 0) AS
  countComment FROM \`hotspot\` LEFT JOIN (SELECT like_hotspot,
  COUNT(like_hotspot) AS countLike FROM \`like\` GROUP BY like_hotspot)
  AS count_table ON count_table.like_hotspot=hotspot.hs_id LEFT JOIN
  (SELECT com_hs, COUNT(com_hs) AS countComment FROM \`comment\` GROUP
  BY com_hs) AS count_comment_table ON
  count_comment_table.com_hs=hotspot.hs_id LEFT JOIN user ON
  user.user_id=hs_user ORDER BY \`hs_id\` DESC LIMIT 20`;
}
```

图十二 复杂的 sql 语句

- 问题七：图片上传与下载

解决方案：一开始是在 github 上查找有关图片上传的的模块，不过因为服务端代码比较复杂，php 和 python 以及 java 都有，本地的客户端在连接过程中经常与服务端的交互出现异常。后面的想法是考虑使用图床功能，将图片上传到七牛云图床，之后可以应用图床对图片尺寸进行统一的裁剪以及方便图片进行管理。不过客户端上传图片的时候，需要搭建一个业务服务器去根据相应的密钥进行 as 加密，之后客户端携带颁发的 token 才能将图片上传至服务器。后面考虑到项目开发的时间原因以及项目规模，远没有很大的用户量而进行图片管理就砍掉了这个功能。而是采用了另一个组员手动实现的图片上传功能，将图片放置于同一个文件夹当中设定权限，数据库保存文件名进行图片文件的查找。客户端根据 url 异步加载图片。

6 思考及感想

我们小组从 14 周就开始准备这个项目，从最初的构思到设计到真正的开发，一直到 19 周，很长的一个开发周期。

在这次开发过程中，我们吸收了我们在开发期中项目的时候，得到的教训。我们组在开发期中项目的时候出现过以下问题，分工不均；在开发前没有预先做好相关的准备，比如确定界面的交互方式，统一数据库接口；组员之间的工作翻车了；使用 Github 进行开发的时候发生冲突导致未提交代码丢失的情况。

所以在这次项目开发之前，我们开会确定了下开发的项目内容，并且设计了 UI 界面图和程序流程图，帮助构建项目思路，同时确定了每个页面的具体功能和交互方式，并且统一了数据库所用的表和接口。在后续开发的过程中，这些准备带来很大的便利。 当然开发前的设想仍然有些不完善，所以我们每周开一次会，汇报进度；保持沟通，跟踪当前开发的进度，改善我们之前设计的不足，然后分配下周任务。

以上是我们这次改进的一些地方。

这次项目中，我们也出现了一些问题。

首先是我们在开发前做的那些准备，交互的确定和数据库的表，因为经验的不足，仍然存在一些缺陷。我们小组成员在开发的过程中，花费了许多时间在讨论底层的逻辑和修改交互方式上的缺点。如动态与点赞之间的关系；评论与通知之间的关系；不了解原生的 RecyclerView 高度不能设置，多个 RecyclerView 在一个页面内，会有非常尴尬的高度；页面跳转逻辑最初设计的时候不够完善，导致 intent 的传入和传出有 bug 等等问题。

然后是在项目管理上还是有所不足。开发前期搭建框架，进度较慢，大部分核心逻辑是从后两周才完成的，同时面临着期末考试，我们不得不砍掉部分的功能，以及放弃掉对 UI 的美化。

我们小组每个成员也同时在进步。通过这次项目，我们对 Android 中 Java 类有了一些新的认识，例如接口类，例如全局类。我们认识到如果能够熟悉

Java 这门语言的话，我们的项目能减少许多代码量，例如小组成员中有建议说我们使用全局类来存放数据，通过这样的方式，我们减少了在写修改，添加和页面跳转这些逻辑的代码量。

然后我们使用了服务器进行数据持久化，简单地接触了一下 curl 语法。小组成员通过查阅资料来熟悉原本不了解的知识。然后成员之间有时候会交流一些各种各样的技巧，例如使用 Dom 树查看 Xml 结点等等。

7 小组分工

以表格形式说明小组各成员的分工，并以百分比的形式注明。

表 1: 小组分工

组员	学号	负责内容	贡献百分比（%）
郭瑶佳	15331093	界面设计，构建 UI 框架（包括布局文件和自定义 Adapter），编写简单的页面逻辑；开发后续的工作，如整合实验报告	25%
宋思亭	14970011	构建 UI 框架（包括布局文件和自定义 Adapter），编写部分页面逻辑；完成部分与数据库的网络交互	23%
谢梓丰	15331329	搭建数据库（phpMyAdmin）以及查询插入（MySQL）等操作，负责网络请求的服务端逻辑。	23%
许基骏	15331338	搭建图片服务器，编写主要的页面逻辑，包括登录、注册、获取动态列表等逻辑。提供网络请求的客户端接口。	29%

如果使用 word，则其他关于表格、图标、字体大小等方面的要求，请参见《中山大学本科生毕业论文（设计）写作与印制规范》。使用 latex 或者 markdown 另当别论。

文档命名格式为：组号 _ 应用名称 _ 说明文档。提交 pdf 格式。绝对不允许抄袭，如发现抄袭，按 0 分处理。