软件（PetsDay）需求规格说明书

目录

第 1 章 引言

1.1 编写目的

编写此文档是为了进一步定制软件开发的细节问题，希望能使本软件开发工作更具体。为了使用户、软件开发者及分析和测试人员对该软件的初始规定有一个共同的理解，它说明了本软件的各项功能需求、性能需求和数据需求，明确标识各项功能的具体含义，阐述实用背景及范围，提供客户解决问题或达到目标所需要的条件或权能，提供一个度量和遵循的基准。具体而言，编写软件需求规格说明的目的是为所开发的软件提出：

a) 软件确认测试的依据。
b) 软件设计总体要求，作为软件开发人员、软件测试人员相互了解的基础；
c) 功能、性能要求，数据结构和采集要求，重要的接口要求，作为软件设计人员进行概要设计的依据；

1.2 编写依据

a) 《使用需求》
b) 《软件设计文档》

1.3 术语与缩略语

缩写、术语及符号	解释
AS	即 Android Studio，类似 Eclipse ADT，是一个提供集成 Andorid 开发工具的开发平台
APK	APK 是 AndroidPackage 的缩写，即 Android 安装包。APK 是类似 Symbian Sis 或 Sisx 的文件格式。通过将 APK 文件直接传到 Android 模拟器或 Android 手机中执行即可安装
更新动态	包括两部分：发布动态和点赞动态
查看评论	可以看到主页上的评论动态
查看通知	可以查看已读通知和未读通知
关注宠物	用户可以关注其他用户的宠物
添加	用户可以自由添加宠物
编辑	用户可以编辑宠物
删除	用户可以删除宠物
查看宠物	用户可以查看宠物的信息，包括：宠物基础数据、动态、粉丝

第 2 章 软件概要

2.1 软件总体描述

在很多社交媒体和视频网站上面，有一类用户通过分享自己的宠物及其日常，为自己带来很高的人气。所以我们打算设计一款专门用于吸宠的社交 App。和如今众多社交 App 不同的是，我们的 App 为了建立以宠物为主打的社交圈，舍弃关于用户的一切社交功能和信息，让这款 App 在内容上关注度更加的集中，所有的功能都是基于宠物这个主体进行开发的。

实用性上，这款 App 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，原则上还是一款社交软件。创新性上，相比于其他社交类软件或者吸宠类软件来说，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，弱化了用户在这款软件当中的功能特性。在交互设计上，我们模仿比较成熟的社交媒体的 UI 风格和交互方式，操作过程当中增加相应的过渡动画，让整个交互更加的合理和友好。

2.2 软件技术选型

a) 客户端：选择移动终端开发。本项目参考了大三上学期 Android 移动应用开发期末项目，一方面作为对一门课程学习成果的实践反馈，另一方面这款 App 定位在宠物社交，模仿借鉴了比较成熟的社交媒体的交互方式。开发平台 AS 专为安卓开发打造，集成 Gradle 自动化构建，可以减少搭建开发环境的开销，同时可以方便地引用开源库的内容来丰富自己的应用。

b) 服务端：使用目前 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。

c) 数据库：使用 MySQL 这个关系型数据库。本次项目使用了线上部署的数据库 phpMyAdmin，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

2.2 软件设计约束及有关说明

a) 遵循的规范

客户端：软件的设计和开发过程需要严格按照《软件设计文档》要求，根据软件的设计方案来进行。软件开发过程遵循 Google Java 编程风格规范，对过程和版本进行管理和控制。

服务端：采用 MySQL 数据库管理数据，使用 Node.js 环境和 Express、Koa 框架来搭建服务端

b) 测试环境： Android 移动终端，Android 5.0.2 系统

c) 软件交付形式： APK 文件

d) 软件交付日期： 201.7.15

2.3 开发与运行环境

a) 软件开发环境

开发平台：AS 3.0.1

操作系统：Windows 10/Mac OSX

开发语言：Java

b) 软件运行环境：Android 4.0 以上。

c) 接口

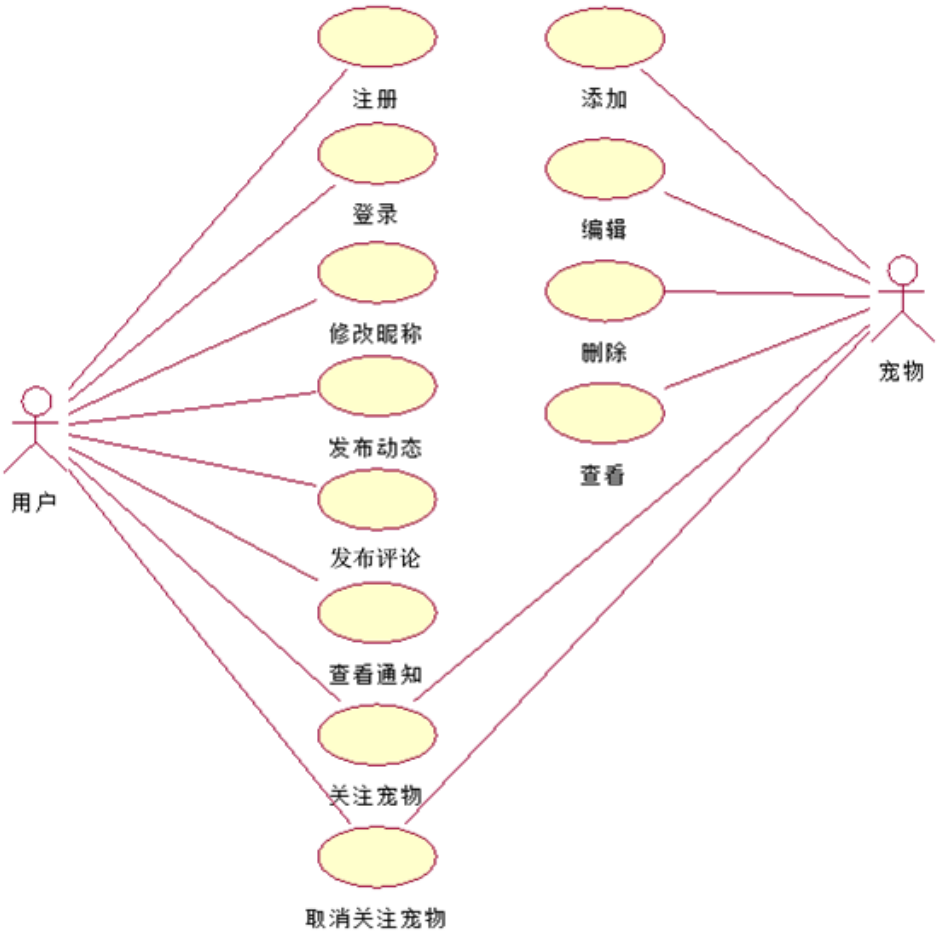
外部接口：用户界面部分按 Android 应用软件用户界面的规范来设计，界面设计风格与 Android 环境保持一致，采用 Android 常规页面作为用户界面，便于用户使用。

软件接口：本软件运行于 Android 4.0 及其以上版本的操作系统之上。

d) 控制与操作：本软件的最终交付形式为 APK 文件，软件的控制操作方法和一般手机 APP 相同。

第 3 章 功能需求

3.1 用例模型



用例图

3.2 用例规约

3.2.1 用户注册

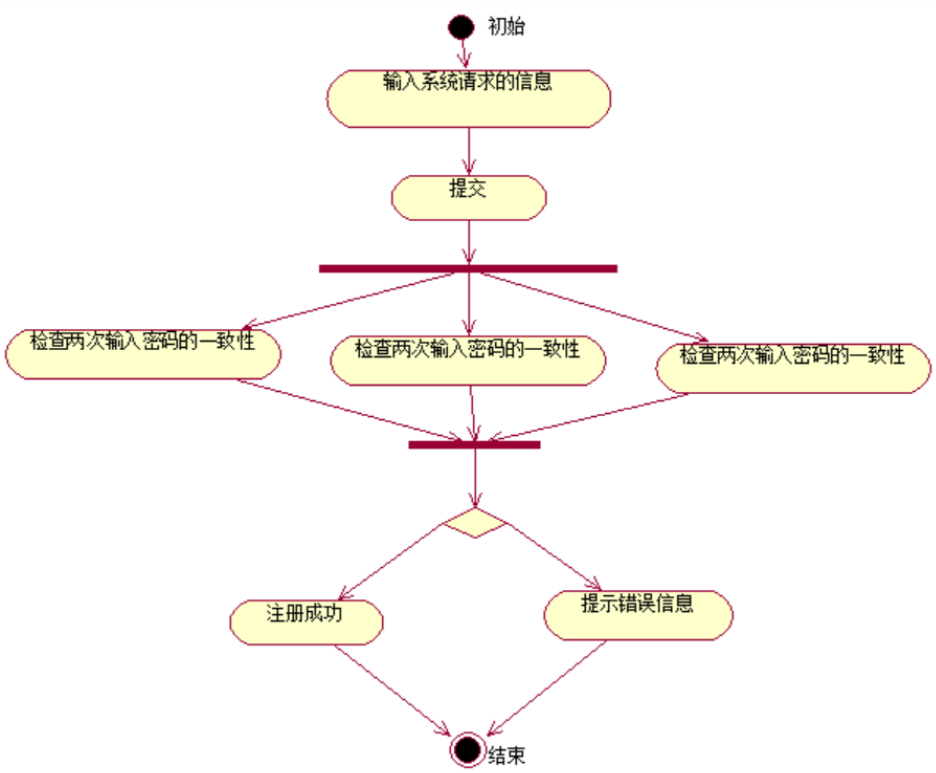
1） **简要说明**

本用例描述用户如何在应用上进行注册

2） **参与者**

注册用户

3） **事件流**



用户注册活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中注册。

i. 系统请求注册用户输入昵称

ii. 系统请求注册用户输入两次密码

iii. 用户提交上述输入内容，系统对输入内容进行检查

A1. 有输入框为空

A2. 用户名已经存在

A3. 两次输入的密码不一样

iv. 系统检查未发现问题，系统把注册用户的信息写入数据库

b） **后备事件流**

A1. 有输入框为空

1. 用户界面提示有输入框为空
2. 返回基本事件流的第一步

A2. 用户名已经存在

3. 用户界面显示用户已存在
4. 返回基本事件流第一步

A3. 两次密码不一样

5. 用户界面显示两次输入的密码不一样
6. 返回基本事件流第一步

4） **特殊需求**

密码输入框中内容不能明文呈现

5） **前置条件**

本用例开始前用户打开注册页面

6） **后置条件**

如果用例成功，用户注册成功，数据库将增加一条注册用户的记录。否则系统状态不改变

3.2.2 用户登录

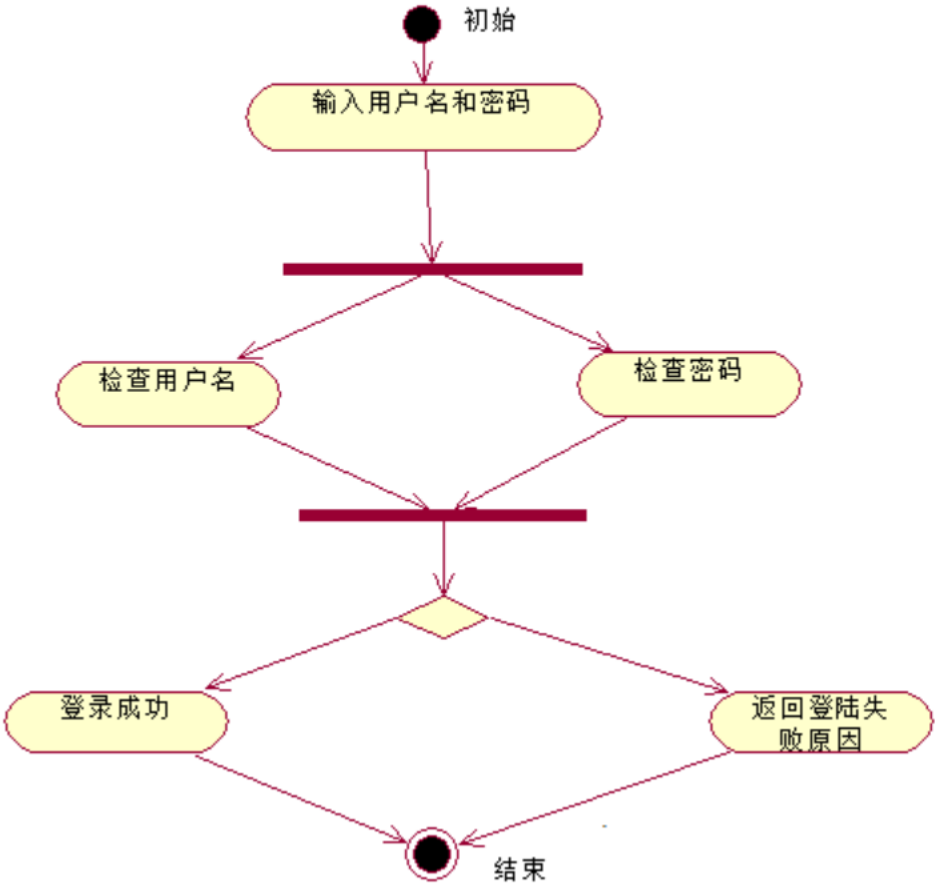
1） **简要说明**

本用例描述注册用户如何登录到系统

2） **参与者**

注册用户

3） **事件流**



用户登录活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中登录。

i. 系统请求注册用户输入用户名

ii. 系统请求注册用户输入密码

iii. 用户提交上述输入内容，系统对输入内容进行检查

A1. 有输入框为空

A2. 用户名不存在

A3. 密码错误

系统检查未发现问题，注册用户登录成功，进入应用主界面

b） **后备事件流**

A1. 有输入框为空

1. 用户界面提示有输入框为空
2. 返回基本事件流的第一步

A2. 用户名不存在

3. 用户界面显示用户不存在
4. 返回基本事件流第一步

A3. 两次密码不一样

5. 用户界面显示两次输入的密码不一样
6. 返回基本事件流第一步

4） **特殊需求**

密码输入框中内容不能明文呈现

5） **前置条件**

本用例开始前用户打开客户端并且客户端检测出用户未登录

6） **后置条件**

如果用例成功，用户将登录成功，进入主界面，并且客户端将保存登录状态。否则系统状态不改变

3.2.3 修改用户昵称

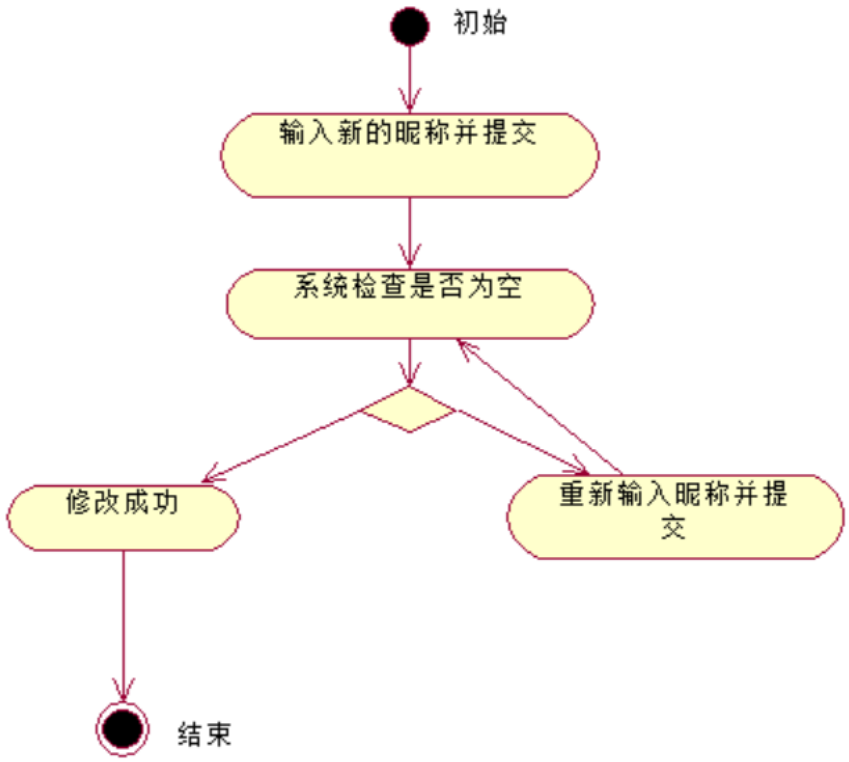
1） **简要说明**

本用例描述已登录用户如何修改昵称

2） **参与者**

已登录注册用户

3） **事件流**



用户修改昵称活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中修改用户昵称。

i. 系统请求用户输入新的昵称

ii. 用户提交输入，客户端检查输入内容

A1. 新昵称输入框为空

iii. 系统检查未发现问题，用户修改昵称成功，新的昵称写入到数据库中该用户对应条目

b） **后备事件流**

A1. 新昵称输入框为空

1. 用户界面提示输入框为空错误信息
2. 回到基本事件流第一步

4） **特殊需求**

无

5） **前置条件**

已登录的注册用户在个人资料页面点击修改昵称

6） **后置条件**

若用例成功，用户将成功修改昵称，数据库将相应更新用户资料条目。否则系统状态不改变

3.2.4 发布动态

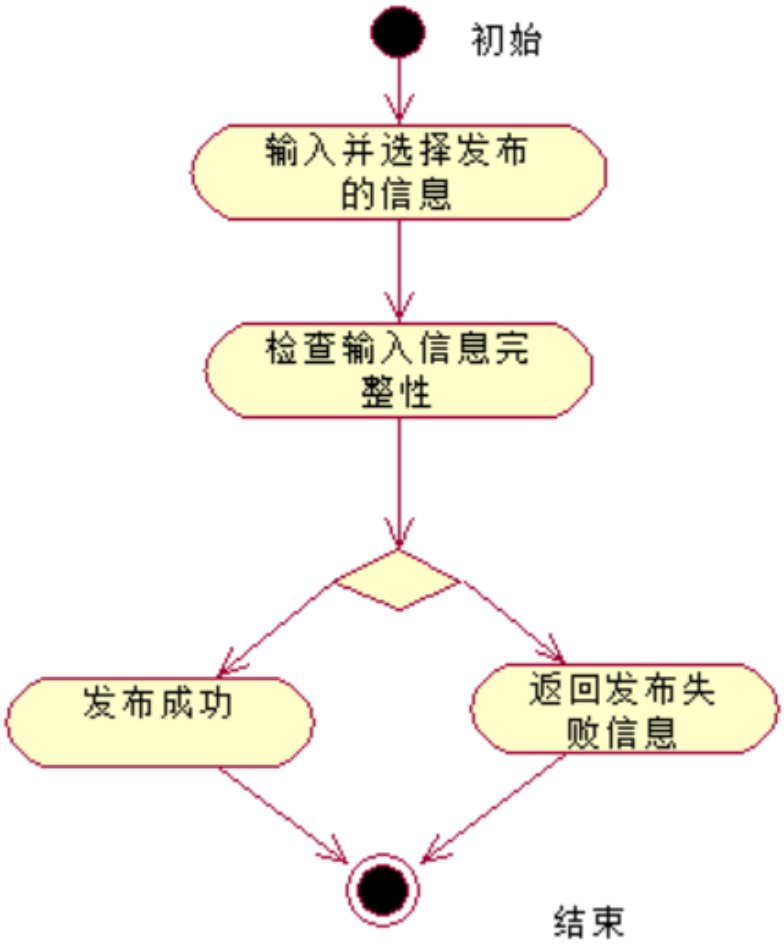
1） **简要说明**

本用例描述用户如何发布动态

2） **参与者**

已经登录的注册用户

3） **事件流**



用户发布动态活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中发布动态。

i. 客户端要求用户输入要发布动态的文字内容

ii. 客户端要求用户选择要发布动态的对应图片

1. 在相册中选择图片并裁剪

2. 调用摄像头拍摄图片并裁剪

iii. 选择对应的宠物

iv. 用户提交上述输入，客户端对输入内容进行检查

A1. 未输入文字内容

A2. 未选择图片

A3. 未选择对应宠物

v. 客户端检查未发现问题，动态发布成功，系统将新动态写入数据库

b） **后备事件流**

A1. 未输入文字内容

1. 用户界面提示未输入文字内容
2. 返回基本事件流第一步

A2. 未选择图片

3. 用户界面提示未选择图片
4. 返回基本事件流第一步

A3. 未选择对应宠物

5. 用户界面提示未选择对应宠物
6. 返回基本事件流第一步

4） **特殊需求**

用户操作前应授予应用访问存储以及调用相机的权限

5） **前置条件**

已经登录的注册用户点击发布新动态

6） **后置条件**

若用例成功，用户将成功发布动态，系统将新的动态写入数据库。否则系统状态不改变

3.2.5 发布评论

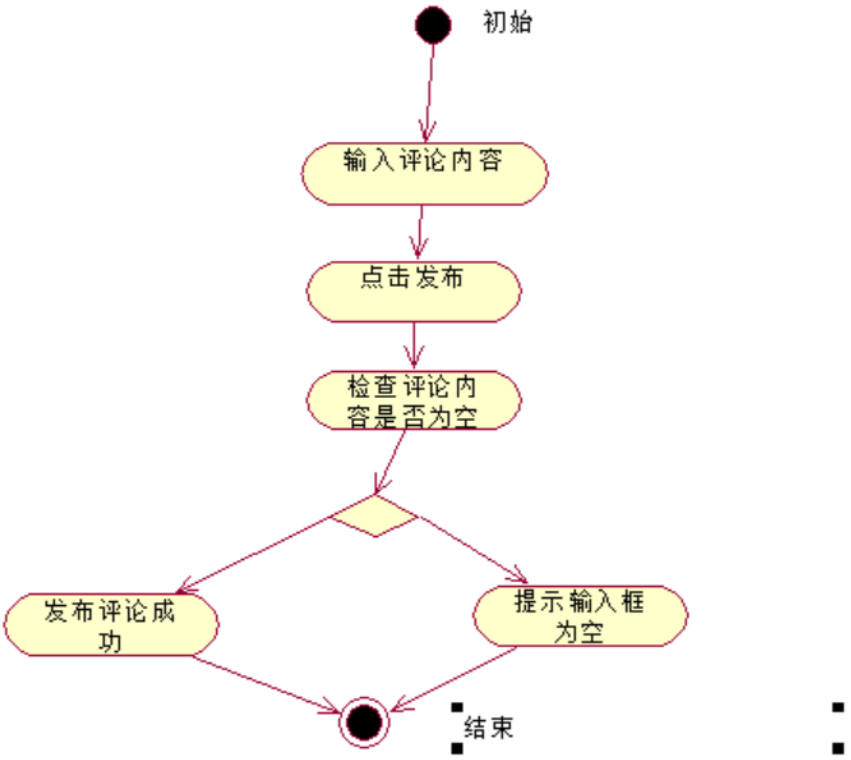
1） **简要说明**

本用例描述已经登录的用户如何发布对动态的评论

2） **参与者**

已经登录的注册用户

3） **事件流**



发布评论活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中对动态发布评论。

i. 客户端请求用户输入评论内容

ii. 用户点击发布评论，客户端检查评论内容

A1. 未输入评论内容

iii. 客户端检查未发现问题，系统将评论写入数据库，动态详情页显示用户发布的评论；服务端为动态发布者生成评论通知，在动态发布者使用应用的时候进行下发

b） **后备事件流**

A1. 未输入评论内容

1. 用户界面提示输入框为空
2. 返回基本事件流第一步

4） **特殊需求**

无

5） **前置条件**

已经登录的注册用户在主页上有动态，点击任意一条动态进入动态详情页，在页面下方点击发布评论

6） **后置条件**

若用例成功，用户将成功发布评论，系统将评论写入数据库，用户界面上对应动态详情页将显示用户发布的评论；服务端将生成对应的评论通知下发给动态发布者。否则系统状态不改变

3.2.6 查看通知

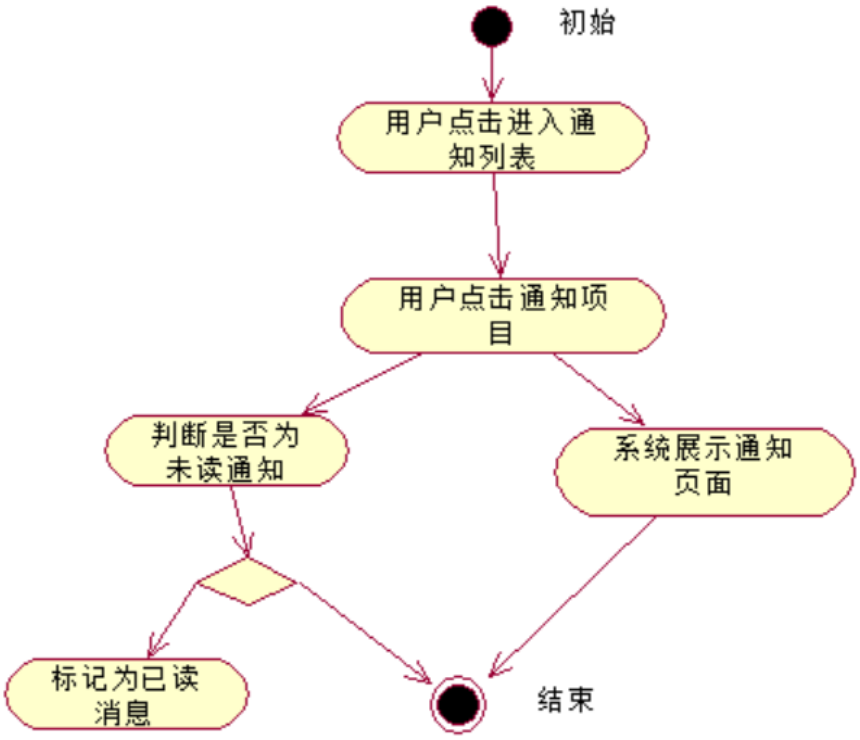
1） **简要说明**

本用例描述已登录的注册用户如何查看通知

2） **参与者**

已登录的注册用户

3） **事件流**



查看通知活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中查看通知。

i. 用户主界面显示当前未读通知数目

ii. 用户点击进入通知列表

iii. 用户点击通知项目，跳转到对应的页面

iv. 若用户点击的是未读通知项，系统将会把数据库中该通知条目标记为已读

b） **后备事件流**

无

4） **特殊需求**

无

5） **前置条件**

用户的通知列表中有通知

6） **后置条件**

若用例成功，用户将成功查看通知，如果查看的是未读通知，系统将会把数据库中该通

知条目标记为已读。否则系统状态不变

3.2.7 关注宠物

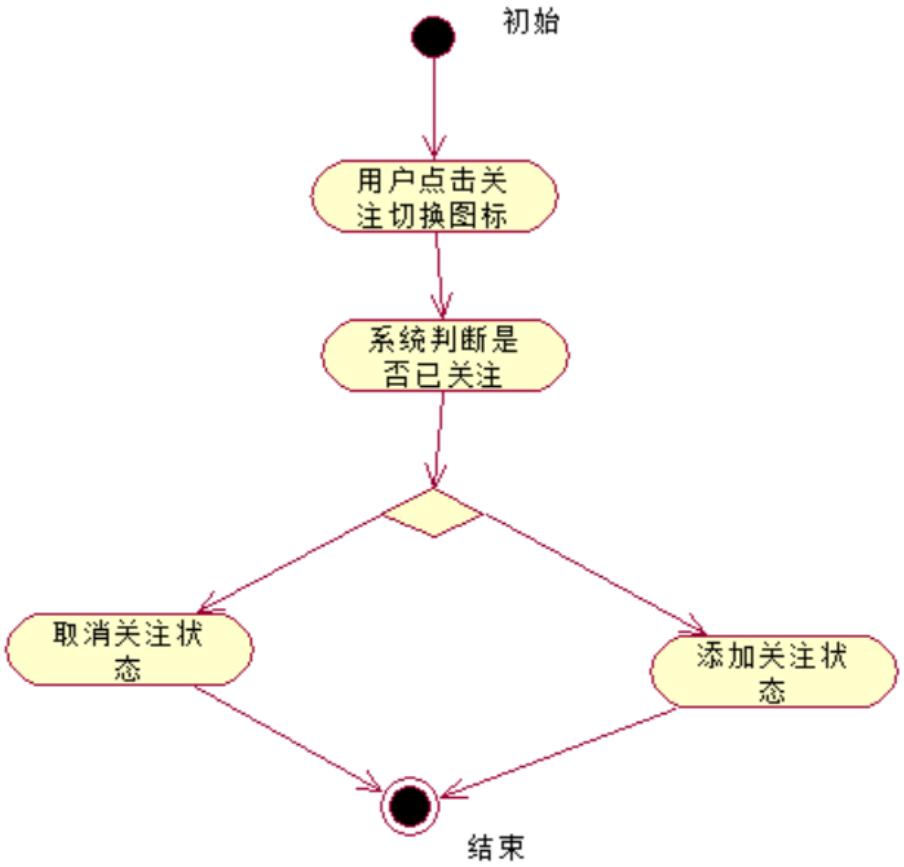
1） **简要说明**

本用例描述已经登录的注册用户如何关注宠物

2） **参与者**

已经登录的注册用户

3） **事件流**



关注宠物活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中关注宠物。

i. 用户点击切换关注状态图标，系统将把关注状态写入数据库

b） **后备事件流**

无

4） **特殊需求**

无

5） **前置条件**

用户需要进入其他用户的资料页，并且对方用户拥有自己创建的宠物且该本方用户未关

注对应宠物才能进行关注

6） **后置条件**

若用例成功，用户将成功关注宠物，关注状态将被系统写入数据库。否则系统状态不变

3.2.8 取消关注宠物

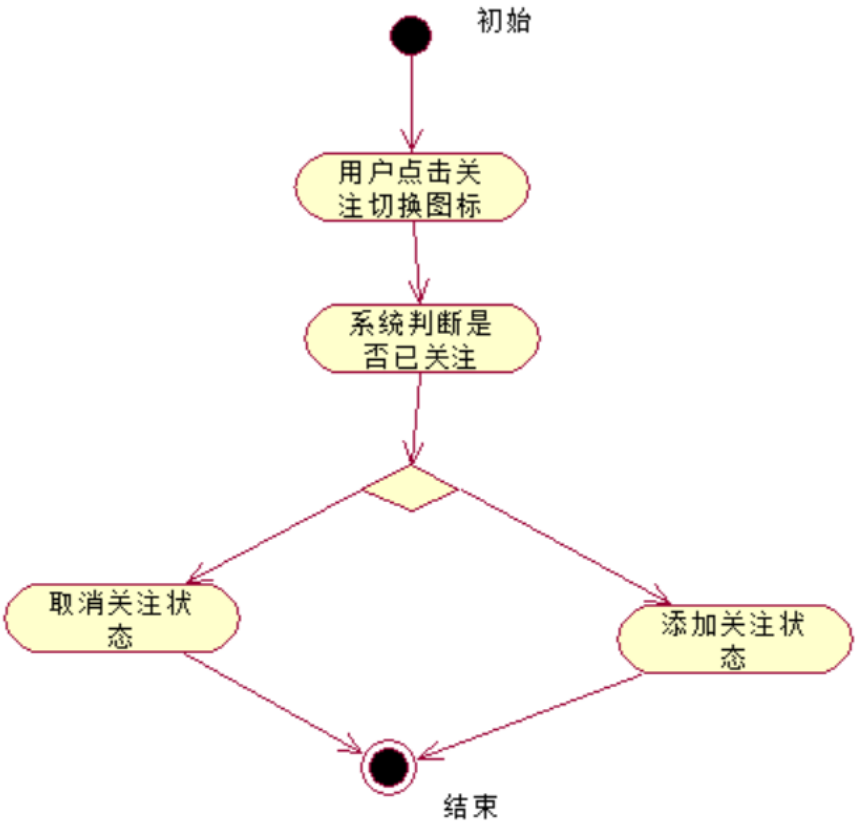
1） **简要说明**

本用例描述已经登录的注册用户如何取消关注宠物

2） **参与者**

已经登录的注册用户

3） **事件流**



取消关注宠物活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中取消关注宠物。

i. 用户点击关注图标，系统将把取消关注状态写入数据库

b） **后备事件流**

无

4） **特殊需求**

无

5） **前置条件**

用户需要在自己的资料页中对自己关注的宠物进行取消关注操作，或者进入其他用户的资料页，并且对方用户拥有自己创建的宠物且该本方用户已关注对应宠物才能进行取消关注

6） **后置条件**

若用例成功，用户将成功取消关注宠物，关注状态将被系统写入数据库。否则系统状态不变

3.2.9 添加宠物

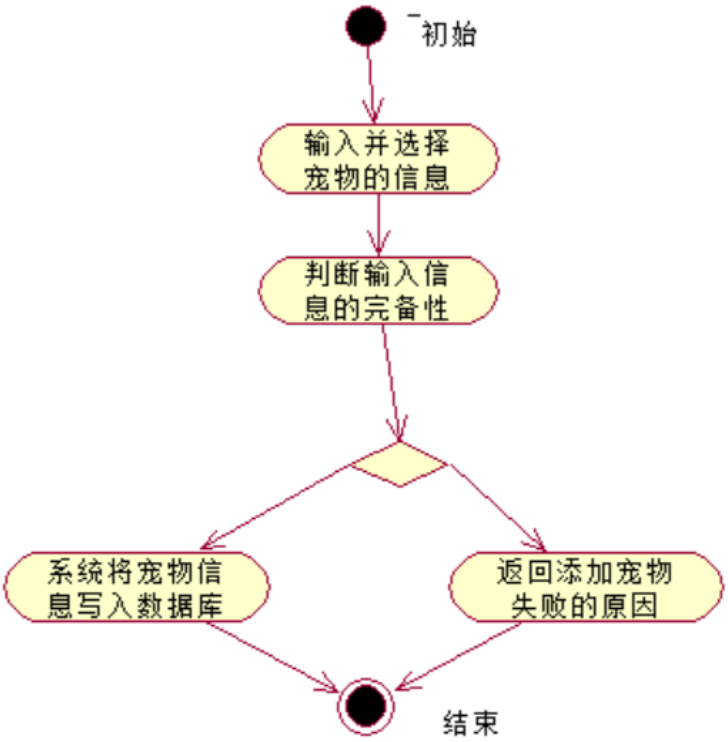
1） **简要说明**

本用例描述已登录的注册用户如何添加宠物

2） **参与者**

已登录的注册用户

3） **事件流**



添加宠物活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中添加宠物。

i. 系统请求用户选择宠物的头像
   1. 在相册中选择图片并裁剪
   2. 调用摄像头拍摄图片并裁剪

ii. 系统请求用户输入宠物的昵称

iii. 系统请求用户输入宠物的类型

iv. 系统请求用户选择宠物性别，默认为雄性
v. 系统请求用户输入宠物体重
vi. 系统请求用户输入宠物的生日
vii. 用户点击提交输入，客户端对上述输入项进行检查
A1. 未选择宠物头像
A2. 存在未输入的项目
viii. 客户端检查未发现问题，系统将宠物信息写入数据库

b） **后备事件流**

A1. 未选择宠物头像
1. 用户界面提示未选择宠物头像
2. 返回基本事件流第一步
A2. 存在未输入的项目
3. 用户界面提示未输入的项目
4. 返回基本事件流第一步

4） **特殊需求**

a) 用户操作前应授予应用访问存储以及调用相机的权限
b) 客户端需要限制用户在选择宠物生日时只能选择当前日期及之前的日期

5） **前置条件**

已登录的注册用户在自己的资料卡中点击添加宠物

6） **后置条件**

若用例成功，用户将成功添加宠物，系统将宠物信息写入数据库。否则系统状态不改变

3.2.10 编辑宠物

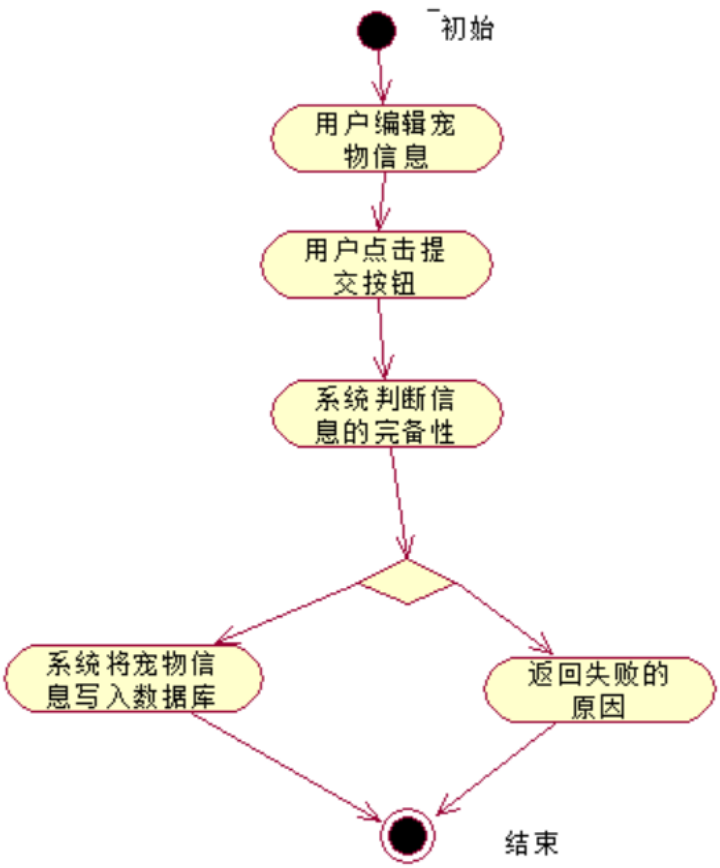
1） **简要说明**

本用例描述已登录的注册用户如何编辑宠物信息

2） **参与者**

已登录且拥有自己添加的宠物的注册用户

3） **事件流**



编辑宠物活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中编辑宠物。

i. 系统请求用户选择宠物的头像
   1. 在相册中选择图片并裁剪
   2. 调用摄像头拍摄图片并裁剪
ii. 系统请求用户输入宠物的昵称
iii. 系统请求用户输入宠物的类型
iv. 系统请求用户选择宠物性别，默认为雄性
v. 系统请求用户输入宠物体重
vi. 系统请求用户输入宠物的生日
vii. 用户点击提交输入，客户端对上述输入项进行检查
   A1. 存在未输入的项目
viii. 客户端检查未发现问题，系统将宠物信息写入数据库

b） **后备事件流**

A1. 存在未输入的项目
1. 用户界面提示未输入的项目
2. 返回基本事件流第一步

4） **特殊需求**

a) 用户操作前应授予应用访问存储以及调用相机的权限

b) 客户端需要限制用户在选择宠物生日时只能选择当前日期及之前的日期

5） **前置条件**

已登录的且拥有自己添加的宠物注册用户在自己的资料卡中点击编辑宠物信息，进入编辑页面后，页面中各个输入框和选择框会填入当前宠物的信息

6） **后置条件**

若用例成功，用户将成功编辑宠物，系统将会把编辑后的宠物信息写入数据库。否则系统状态不改变

3.2.11 查看宠物

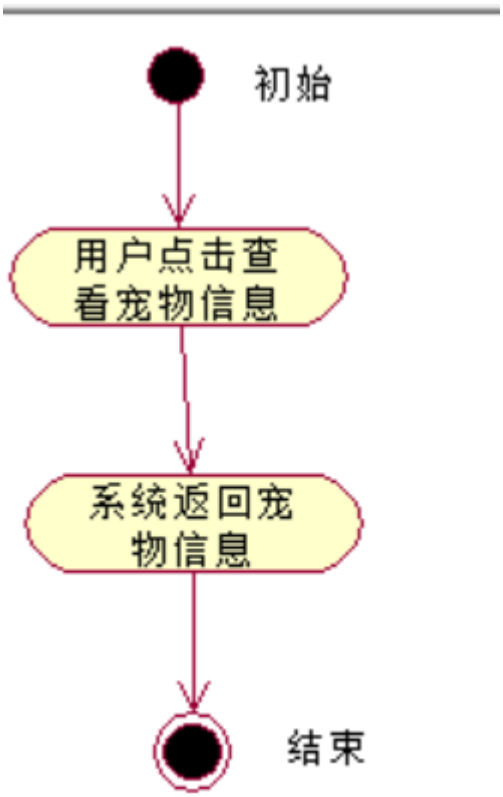
1） **简要说明**

本用例描述已经登录的注册用户如何查看宠物

2） **参与者**

已经登录的且拥有关注的宠物或者自己添加的宠物的注册用户

3） **事件流**



查看宠物活动图

a） **基本事件流**

本用例开始于用户希望在 PetsDay 中查看宠物。

i. 注册用户查看自己关注的或者添加的宠物

b） **后备事件流**

无

4） 特殊需求

无

5） 前置条件

已登录的且拥有自己添加的宠物注册用户在自己的资料卡中点击查看宠物信息

6） 后置条件

无

3.3 补充规约

本部分为该系统的整体非功能性补充要求

1）目标

本小节的目的是定义 PetsDay 中的整体非功能性补充要求。补充规约和用力模型一桶定义关于系统的一整套需求。

2）范围

本规约定义了 PetsDay 的非功能性需求，例如：兼容性、可靠性、性能、易用性、安全性、设计约束。（功能性需求在用例规约中定义。）

3）兼容性

应用可支持 Android 操作系统。

4）可靠性

应用要保证联系人的信息、宠物的信息、动态、主页、通知内容完整无差错，且可以永久保存。

5）性能

应用可支持 5000 个在线用户。

用户进行关注/取关、添加/删除宠物时，应用的响应时间应小于 8 秒。

用户进入主页、获取动态时，应用的响应时间应小于 58 秒。

6）易用性

应用充分考虑经常使用的功能和很少使用的功能在页面上的分布，对于大部分常用的功能，直接提供快捷按键便于用户操作，例如：修改昵称、添加宠物等。

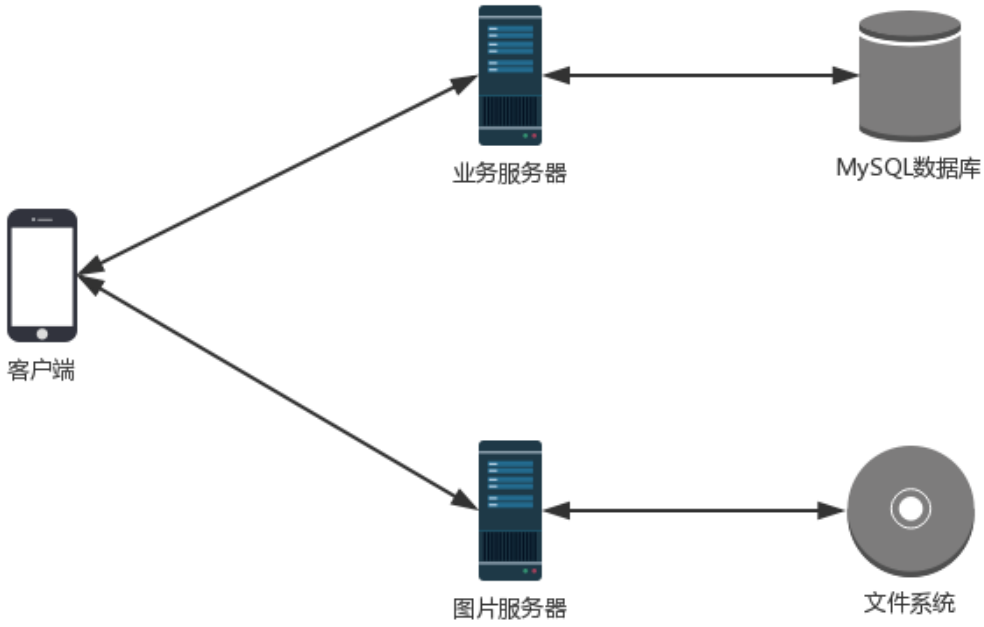
7）安全性

应用要求对用户私密信息，例如：密码、通知等，进行操作权限管理，保证用户只能在各自允许的权限范围内查看个人私密信息。

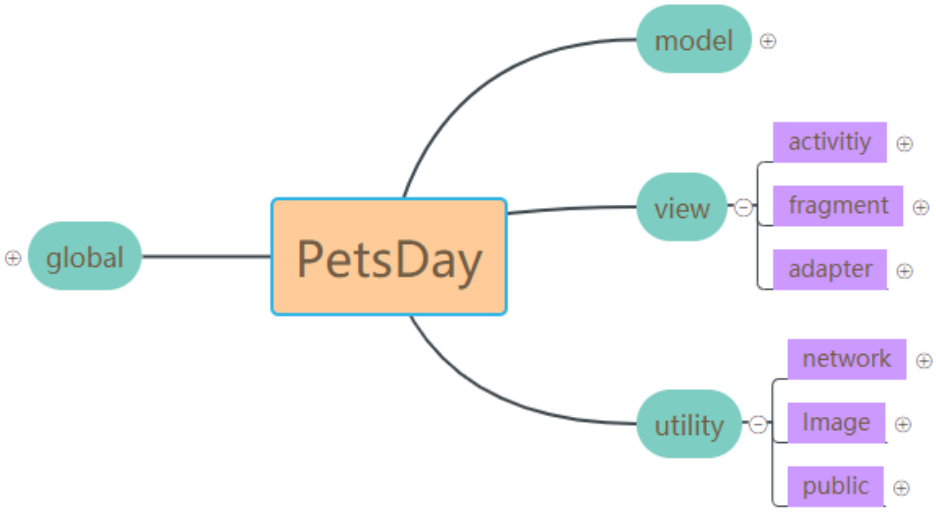
第 4 章 架构分析与设计

4.1 描述架构

本系统基于 C/S 架构，客户端用 JDK、Android SDK 开发实现，服务端用 Node.js 环境开发实现。客户端与服务端之间采用 RESTful API 交互。



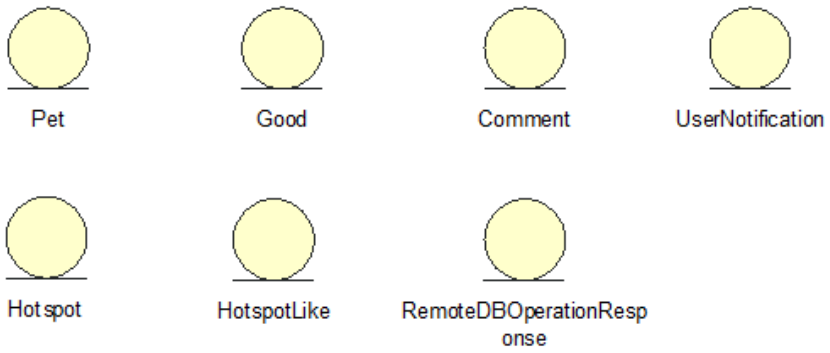
总架构图



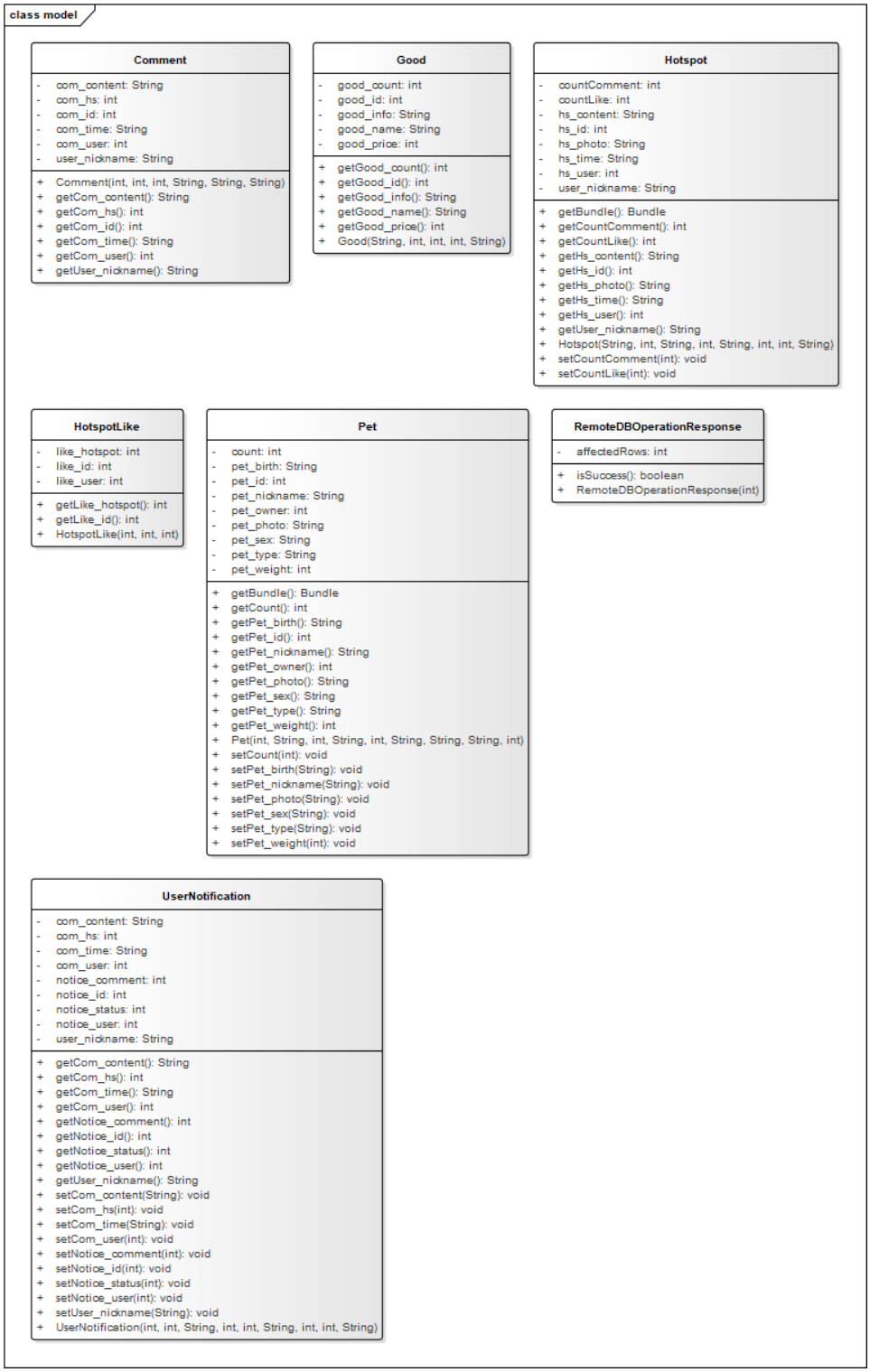
客户端架构图

4.2 关键抽象

系统关键抽象即系统实体类图，系统实体类描述了系统中的类及其相互之间的各种关系，它反映了系统中包含的各种对象的类型以及对象之间的各种静态关系。主要描述了系统实体层中各实体类的属性及其相互的关系。是对实体层中的各模块的描述。



关键抽象图



关键抽象类图

4.3 用例分析

由于系统的用例较多，无法一一列举，这里只选择用户注册这个用例进行详细分析，其他的用例与这个用例相似。其他的用例分析与这两例相似。

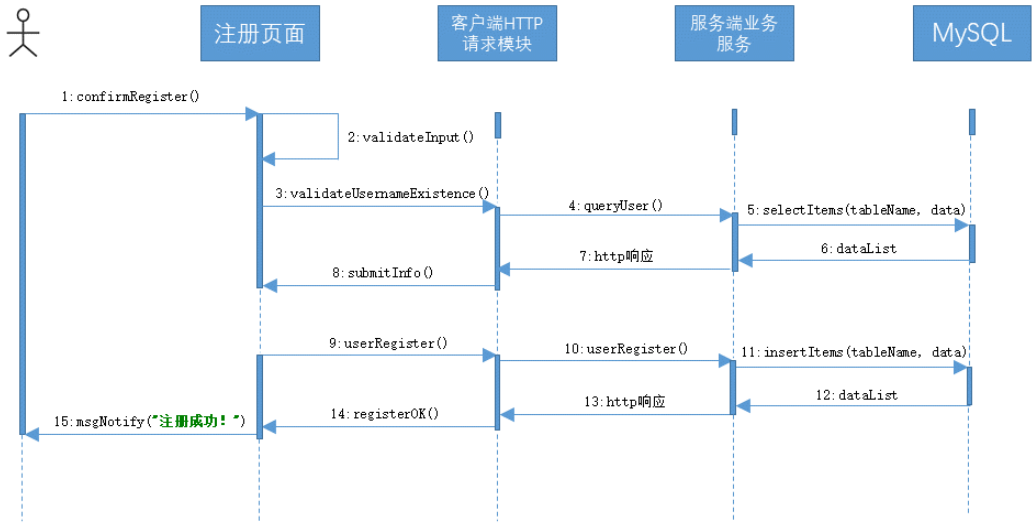
4.3.1 添加宠物用例分析

4.3.2 用例功能描述

用户可以利用这一功能进行注册

4.3.3 用例交互过程

1. 用户进入客户端注册页面，填写相关信息，点击提交
2. 客户端对用户填写内容进行检查，若发现问题则予以提示，未发现问题则提交到服务端
3. 服务端检查用户名是否存在，不存在则可以成功注册，服务端将数据写入数据库，返回确认信息。若用户名存在，则返回用户名已经存在的信息，客户端再显示到用户界面上



用例交互过程

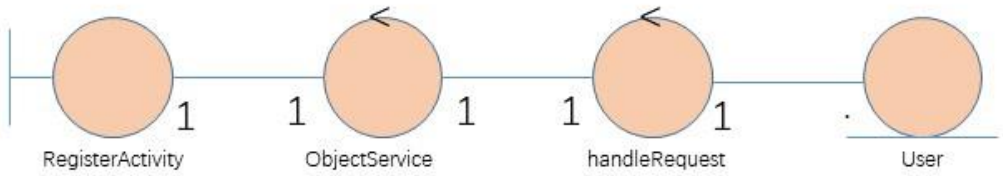
4.3.4 用例的类分析和设计

1. 边界类：用例中，边界类为用户注册界面 RegisterActivity.java。该页面以表单为主，用于获取用户的注册信息并显示注册提示信息。
2. 控制类：用例中，客户端控制类为 RegisterActivity.java 和 ObjectService.java，服务端控制类为 handleRequest。客户端的控制类负责注册信息本地校验以及向服务端发送请求。服务端控制类负责解析客户端请求并操作数据库。
3. 实体类：用户实体类，即为数据库中的 User，存储了用户的各种信息。

4.3.5 分析类关联关系

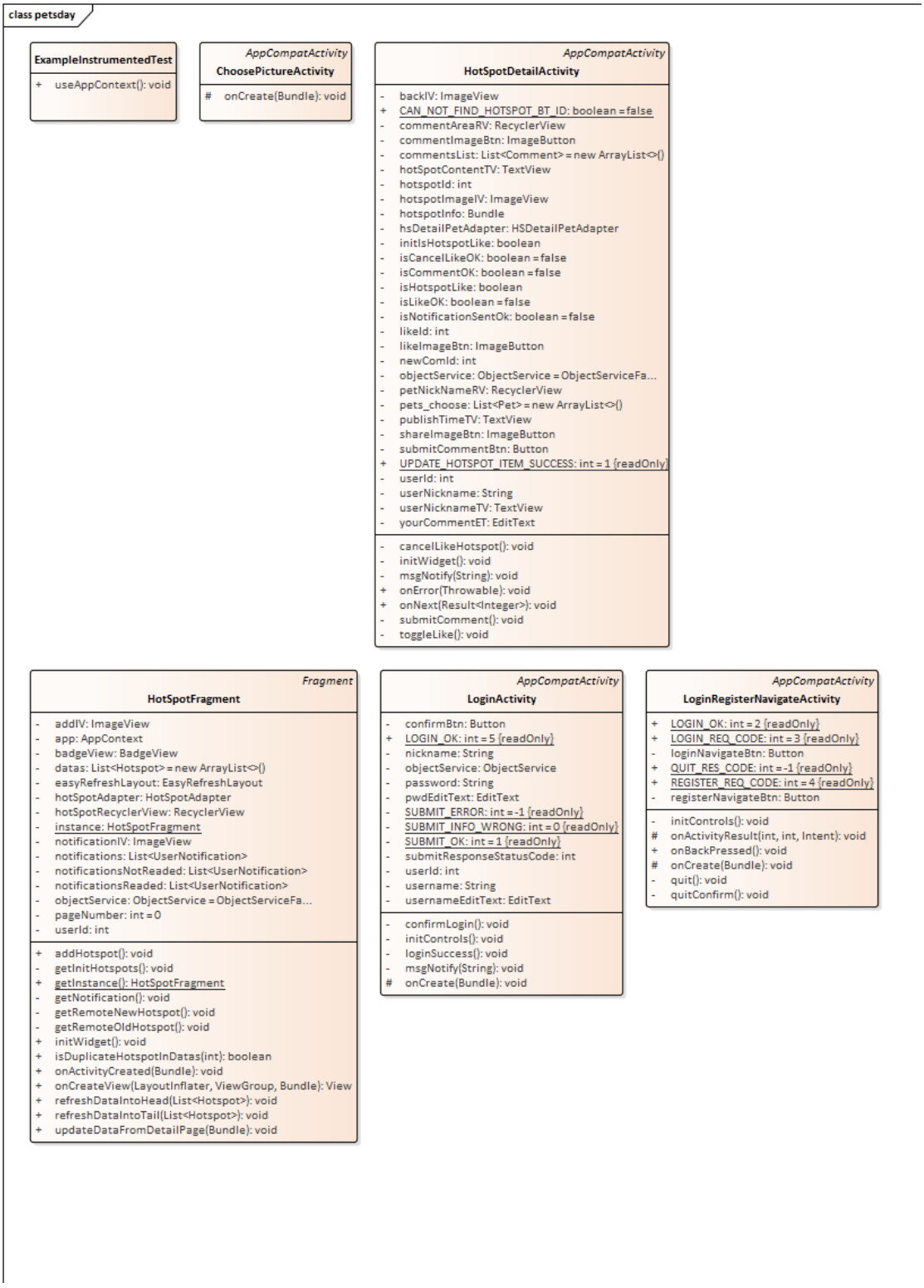
RegistActivity 注册类调用 ObjectService 对象服务类，对象服务类调用请求处理类

HandleRequest，对象处理类调用对象实体 User。

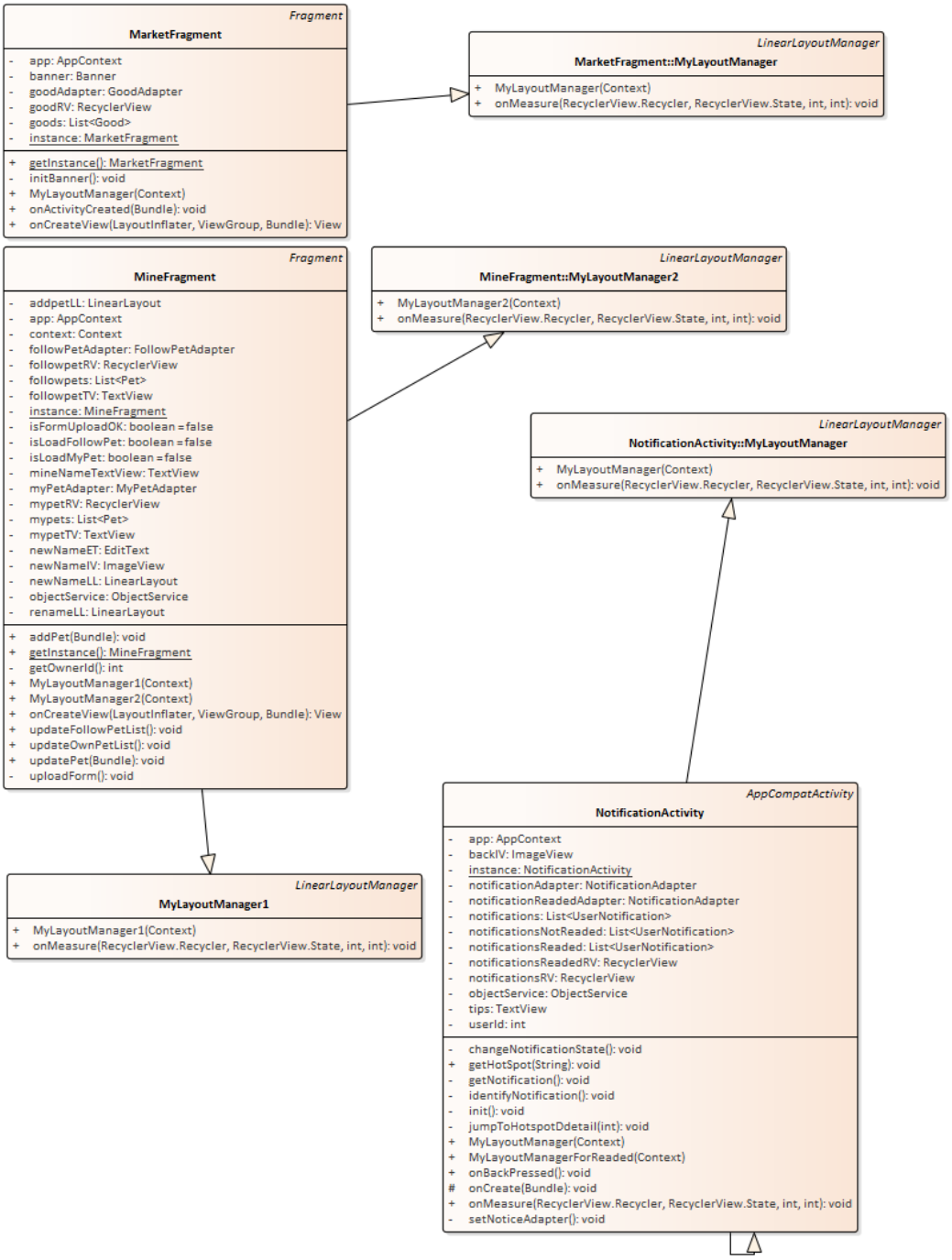


类关联关系

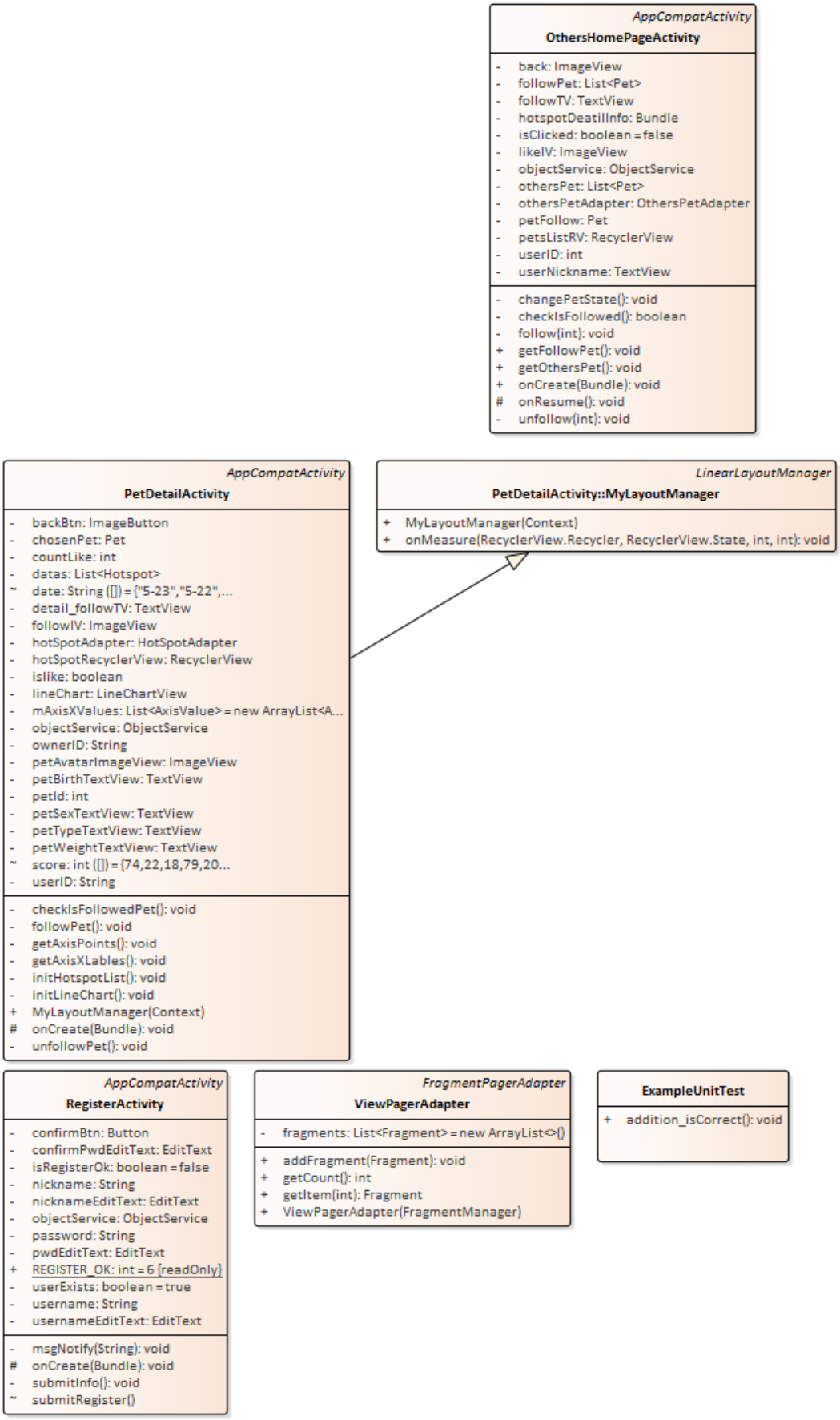
4.4 系统类图



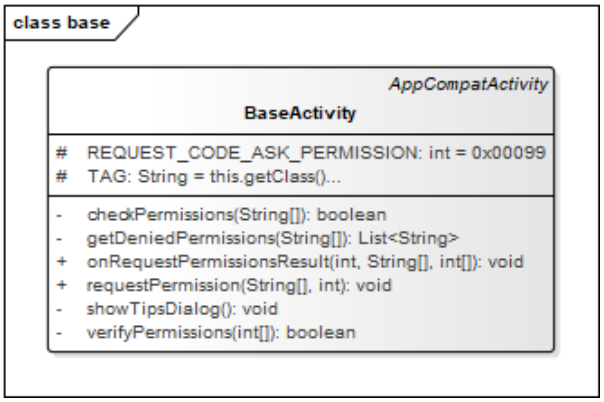
系统类图_1


Fragment
MarketFragment
- app: AppContext
- banner: Banner
- goodAdapter: GoodAdapter
- goodRV: RecyclerView
- goods: List<Good>
- instance: MarketFragment
+ getInstance(): MarketFragment
- initBanner(): void
+ MyLayoutManager(Context)
+ onActivityCreated(Bundle): void
+ onCreateView(LayoutInflater, ViewGroup, Bundle): View
LinearLayoutManager
MarketFragment::MyLayoutManager
+ MyLayoutManager(Context)
+ onMeasure(RecyclerView.Recycler, RecyclerView.State, int, int): void
Fragment
MineFragment
- addpetLL: LinearLayout
- app: AppContext
- context: Context
- followPetAdapter: FollowPetAdapter
- followpetRV: RecyclerView
- followpets: List<Pet>
- followpetTV: TextView
- instance: MineFragment
- isFormUploadOK: boolean = false
- isLoadFollowPet: boolean = false
- isLoadMyPet: boolean = false
- mineNameTextView: TextView
- myPetAdapter: MyPetAdapter
- mypetRV: RecyclerView
- mypets: List<Pet>
- mypetTV: TextView
- newNameET: EditText
- newNameIV: ImageView
- newNameLL: LinearLayout
- objectService: ObjectService
- renameLL: LinearLayout
+ addPet(Bundle): void
+ getInstance(): MineFragment
- getOwnerId(): int
+ MyLayoutManager1(Context)
+ MyLayoutManager2(Context)
+ onCreateView(LayoutInflater, ViewGroup, Bundle): View
+ updateFollowPetList(): void
+ updateOwnPetList(): void
+ updatePet(Bundle): void
- uploadForm(): void
LinearLayoutManager
MineFragment::MyLayoutManager2
+ MyLayoutManager2(Context)
+ onMeasure(RecyclerView.Recycler, RecyclerView.State, int, int): void
LinearLayoutManager
NotificationActivity::MyLayoutManager
+ MyLayoutManager(Context)
+ onMeasure(RecyclerView.Recycler, RecyclerView.State, int, int): void
LinearLayoutManager
MyLayoutManager1
+ MyLayoutManager1(Context)
+ onMeasure(RecyclerView.Recycler, RecyclerView.State, int, int): void
AppCompatActivity
NotificationActivity
- app: AppContext
- backIV: ImageView
- instance: NotificationActivity
- notificationAdapter: NotificationAdapter
- notificationReadedAdapter: NotificationAdapter
- notifications: List<UserNotification>
- notificationsNotReaded: List<UserNotification>
- notificationsReaded: List<UserNotification>
- notificationsReadedRV: RecyclerView
- notificationsRV: RecyclerView
- objectService: ObjectService
- tips: TextView
- userId: int
- changeNotificationState(): void
+ getHotSpot(String): void
- getNotification(): void
- identifyNotification(): void
- init(): void
- jumpToHotspotDdetail(int): void
+ MyLayoutManager(Context)
+ MyLayoutManagerForReaded(Context)
+ onBackPressed(): void
# onCreate(Bundle): void
+ onMeasure(RecyclerView.Recycler, RecyclerView.State, int, int): void
- setNoticeAdapter(): void

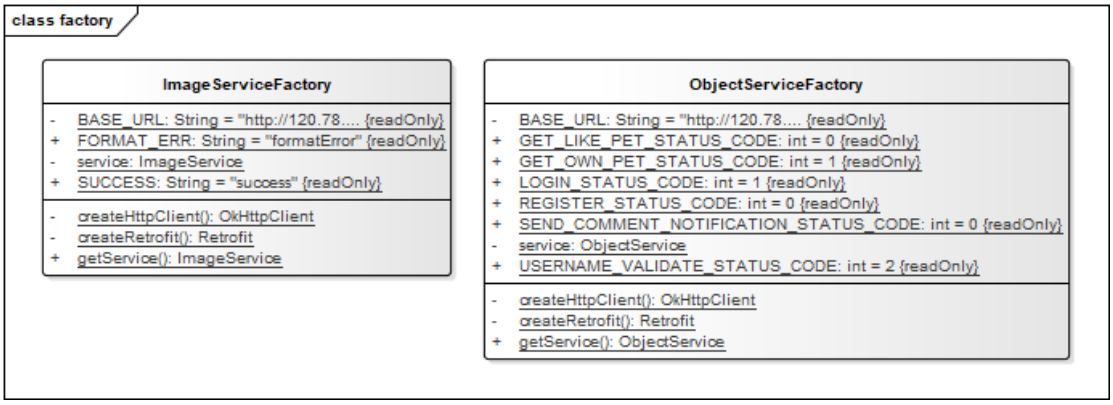
系统类图_2


AppCompatActivity
OthersHomePageActivity
- back: ImageView
- followPet: List<Pet>
- followTV: TextView
- hotspotDeatilInfo: Bundle
- isClicked: boolean = false
- likeIV: ImageView
- objectService: ObjectService
- othersPet: List<Pet>
- othersPetAdapter: OthersPetAdapter
- petFollow: Pet
- petsListRV: RecyclerView
- userID: int
- userNickname: TextView
- changePetState(): void
- checkIsFollowed(): boolean
- follow(int): void
+ getFollowPet(): void
+ getOthersPet(): void
+ onCreate(Bundle): void
# onResume(): void
- unfollow(int): void
AppCompatActivity
PetDetailActivity
- backBtn: ImageButton
- chosenPet: Pet
- countLike: int
- datas: List<Hotspot>
~ date: String ([]) = {"5-23","5-22",...
- detail_followTV: TextView
- followIV: ImageView
- hotSpotAdapter: HotSpotAdapter
- hotSpotRecyclerView: RecyclerView
- islike: boolean
- lineChart: LineChartView
- mAxisXValues: List<AxisValue> = new ArrayList<A...
- objectService: ObjectService
- ownerID: String
- petAvatarImageView: ImageView
- petBirthTextView: TextView
- petId: int
- petSexTextView: TextView
- petTypeTextView: TextView
- petWeightTextView: TextView
~ score: int ([]) = {74,22,18,79,20...
- userID: String
- checkIsFollowedPet(): void
- followPet(): void
- getAxisPoints(): void
- getAxisXLables(): void
- initHotspotList(): void
- initLineChart(): void
+ MyLayoutManager(Context)
# onCreate(Bundle): void
- unfollowPet(): void
LinearLayoutManager
PetDetailActivity::MyLayoutManager
+ MyLayoutManager(Context)
+ onMeasure(RecyclerView.Recycler, RecyclerView.State, int, int): void
AppCompatActivity
RegisterActivity
- confirmBtn: Button
- confirmPwdEditText: EditText
- isRegisterOk: boolean = false
- nickname: String
- nicknameEditText: EditText
- objectService: ObjectService
- password: String
- pwdEditText: EditText
+ REGISTER_OK: int = 6 {readOnly}
- userExists: boolean = true
- username: String
- usernameEditText: EditText
- msgNotify(String): void
onCreate(Bundle): void
- submitInfo(): void
~ submitRegister()
FragmentPagerAdapter
ViewPagerAdapter
- fragments: List<Fragment> = new ArrayList<>()
+ addFragment(Fragment): void
+ getCount(): int
+ getItem(int): Fragment
+ ViewPagerAdapter(FragmentManager)
ExampleUnitTest
+ addition_isCorrect(): void

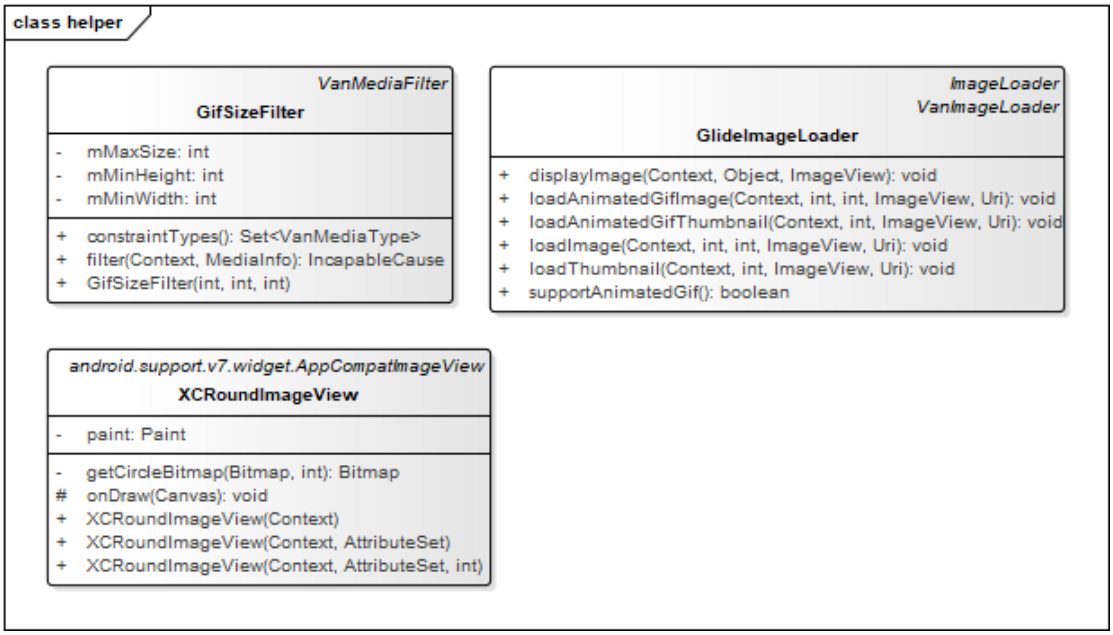
系统类图_3



系统类图_4



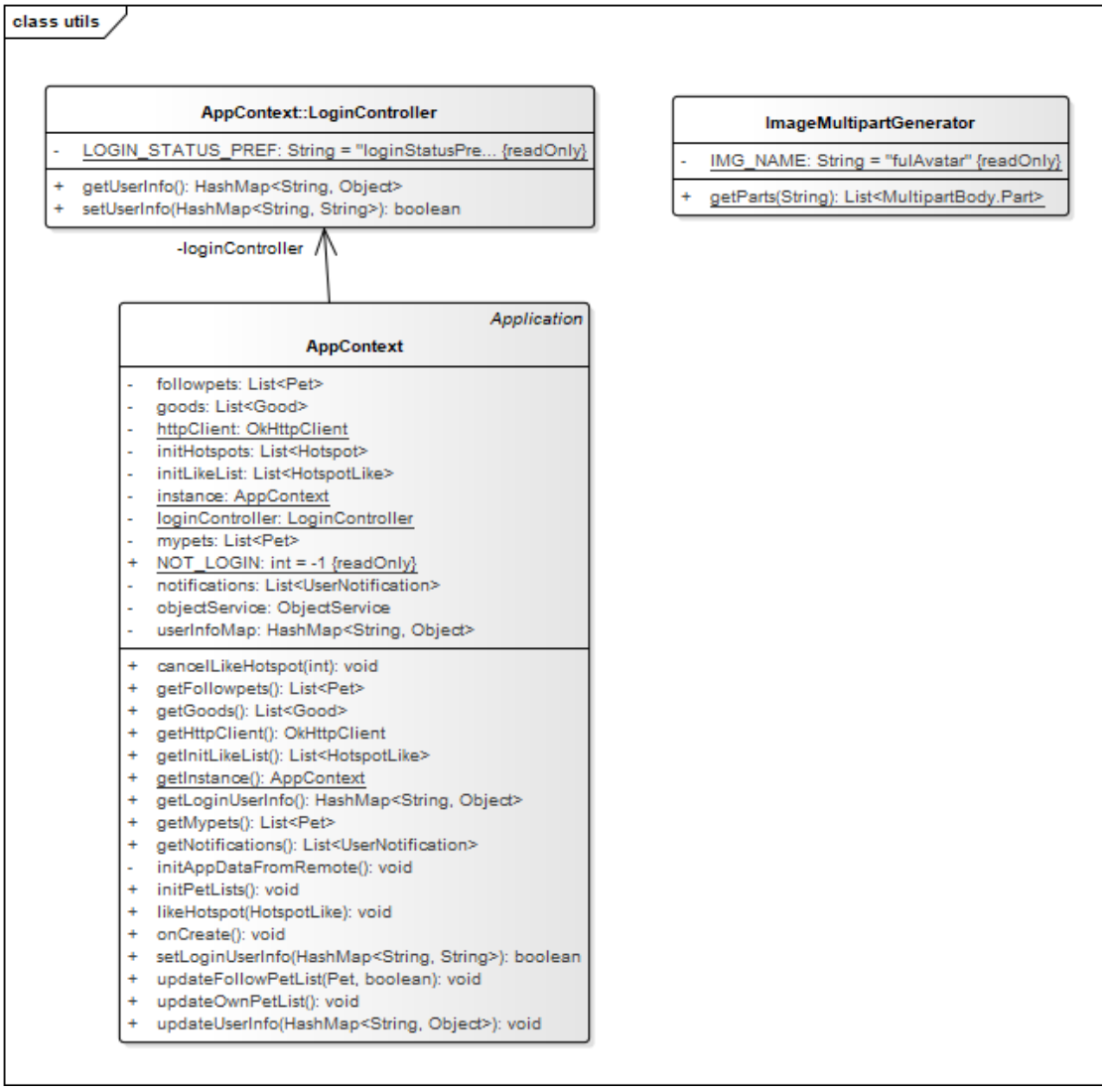
系统类图_5



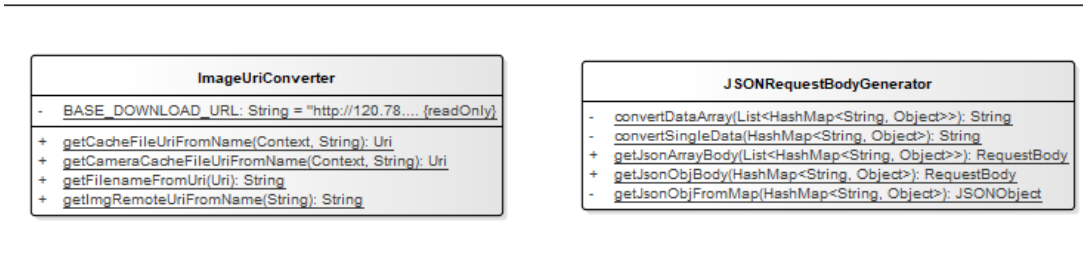
系统类图_6


class service
«interface»
ImageService
+ uploadAvatar(List<MultipartBody.Part>): Observable<Result<String>>
«interface»
ObjectService
+ cancelLike(String): Observable<RemoteDBOperationResponse>
+ cancelUserFansPet(String, String): Observable<RemoteDBOperationResponse>
+ combinePetAndHotspot(RequestBody): Observable<Result<Integer>>
+ deleteCommentById(String): Observable<RemoteDBOperationResponse>
+ getCommentListByHotspot(String): Observable<List<Comment>>
+ getHotspotById(String): Observable<List<Hotspot>>
+ getHotspotByPetId(String): Observable<List<Hotspot>>
+ getHotspotListByPageNumber(String): Observable<List<Hotspot>>
+ getLikeListByUserId(String): Observable<List<HotspotLike>>
+ getNewestHotspotList(): Observable<List<Hotspot>>
+ getNotificationByUserId(String): Observable<List<UserNotification>>
+ getPetListForHotspot(String): Observable<List<Pet>>
+ getPetListForUser(String, String): Observable<List<Pet>>
+ haveReadNotification(RequestBody): Observable<RemoteDBOperationResponse>
+ insertComment(RequestBody): Observable<Result<Integer>>
+ insertHotspot(RequestBody): Observable<Result<Integer>>
+ insertPet(RequestBody): Observable<Result<Integer>>
+ likeHotspot(RequestBody): Observable<Result<Integer>>
+ postNotification(RequestBody): Observable<Result<Integer>>
+ publishGood(RequestBody): Observable<Result<Integer>>
+ queryUsername(RequestBody): Observable<ResponseBody>
+ updatePet(RequestBody): Observable<RemoteDBOperationResponse>
+ updateUser(RequestBody): Observable<RemoteDBOperationResponse>
+ userFansPet(RequestBody): Observable<Result<Integer>>
+ userLogin(RequestBody): Observable<ResponseBody>
+ userRegister(RequestBody): Observable<Result<Integer>>

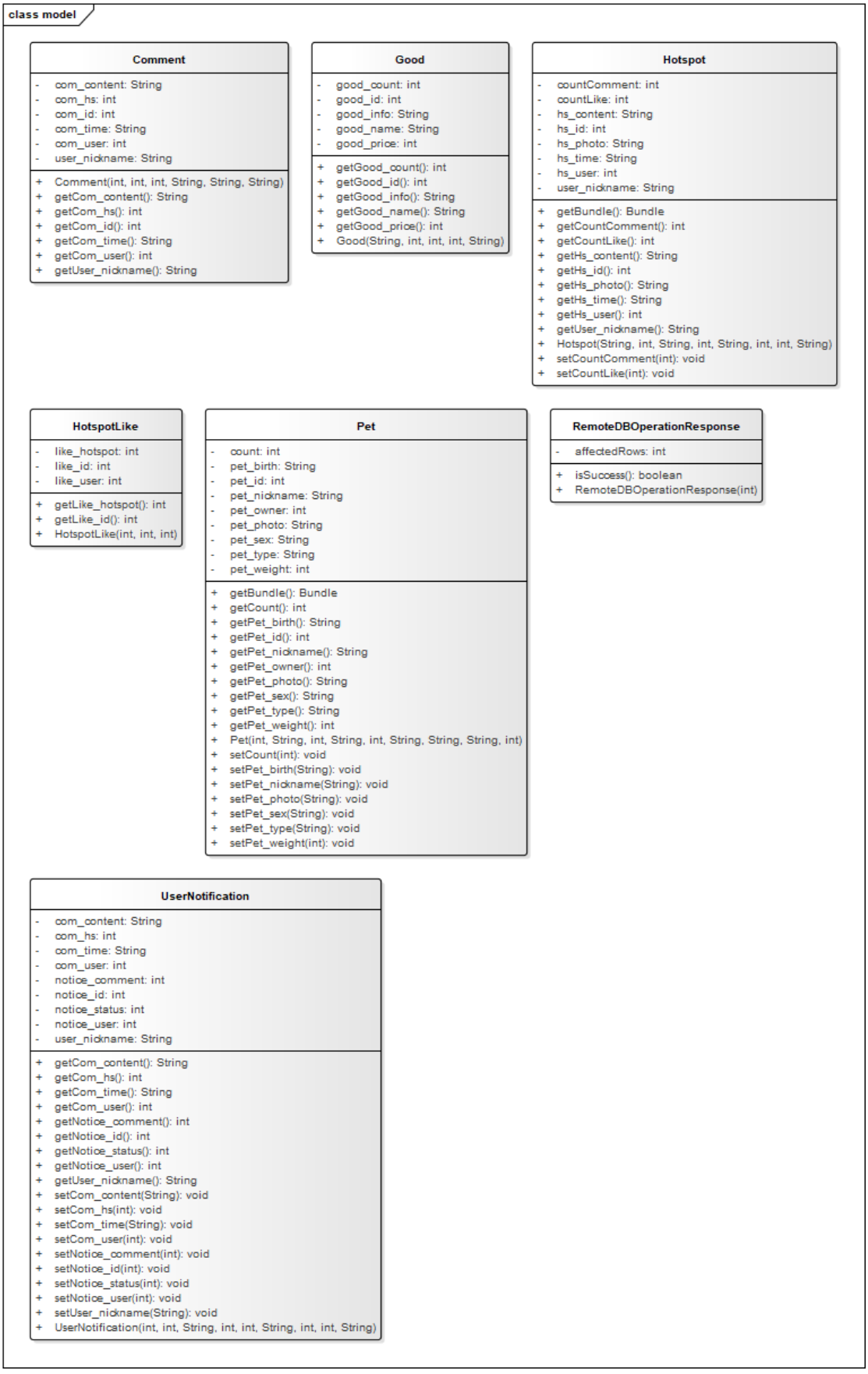
系统类图_7


class utils
AppContext::LoginController
- LOGIN_STATUS_PREF: String = "loginStatusPre... {readOnly}
+ getUserInfo(): HashMap<String, Object>
+ setUserInfo(HashMap<String, String>): boolean
-loginController
ImageMultipartGenerator
- IMG_NAME: String = "fulAvatar" {readOnly}
+ getParts(String): List<MultipartBody.Part>
Application
AppContext
- followpets: List<Pet>
- goods: List<Good>
- httpClient: OkHttpClient
- initHotspots: List<Hotspot>
- initLikeList: List<HotspotLike>
- instance: AppContext
- loginController: LoginController
- mypets: List<Pet>
+ NOT_LOGIN: int = -1 {readOnly}
- notifications: List<UserNotification>
- objectService: ObjectService
- userInfoMap: HashMap<String, Object>
+ cancelLikeHotspot(int): void
+ getFollowpets(): List<Pet>
+ getGoods(): List<Good>
+ getHttpClient(): OkHttpClient
+ getInitLikeList(): List<HotspotLike>
+ getInstance(): AppContext
+ getLoginUserInfo(): HashMap<String, Object>
+ getMypets(): List<Pet>
+ getNotifications(): List<UserNotification>
- initAppDataFromRemote(): void
+ initPetLists(): void
+ likeHotspot(HotspotLike): void
+ onCreate(): void
+ setLoginUserInfo(HashMap<String, String>): boolean
+ updateFollowPetList(Pet, boolean): void
+ updateOwnPetList(): void
+ updateUserInfo(HashMap<String, Object>): void

系统类图_8


ImageUriConverter
- BASE_DOWNLOAD_URL: String = "http://120.78.... {readOnly}
+ getCacheFileUriFromName(Context, String): Uri
+ getCameraCacheFileUriFromName(Context, String): Uri
+ getFilenameFromUri(Uri): String
+ getImgRemoteUriFromName(String): String
JSONRequestBodyGenerator
- convertDataArray(List<HashMap<String, Object>>): String
- convertSingleData(HashMap<String, Object>): String
+ getJsonArrayBody(List<HashMap<String, Object>>): RequestBody
+ getJsonObjBody(HashMap<String, Object>): RequestBody
- getJsonObjFromMap(HashMap<String, Object>): JSONObject

系统类图_9


class model
Comment
- com_content: String
- com_hs: int
- com_id: int
- com_time: String
- com_user: int
- user_nickname: String
+ Comment(int, int, int, String, String, String)
+ getCom_content(): String
+ getCom_hs(): int
+ getCom_id(): int
+ getCom_time(): String
+ getCom_user(): int
+ getUser_nickname(): String
Good
- good_count: int
- good_id: int
- good_info: String
- good_name: String
- good_price: int
+ getGood_count(): int
+ getGood_id(): int
+ getGood_info(): String
+ getGood_name(): String
+ getGood_price(): int
+ Good(String, int, int, int, String)
Hotspot
- countComment: int
- countLike: int
- hs_content: String
- hs_id: int
- hs_photo: String
- hs_time: String
- hs_user: int
- user_nickname: String
+ getBundle(): Bundle
+ getCountComment(): int
+ getCountLike(): int
+ getHs_content(): String
+ getHs_id(): int
+ getHs_photo(): String
+ getHs_time(): String
+ getHs_user(): int
+ getUser_nickname(): String
+ Hotspot(String, int, String, int, String, int, int, String)
+ setCountComment(int): void
+ setCountLike(int): void
HotspotLike
- like_hotspot: int
- like_id: int
- like_user: int
+ getLike_hotspot(): int
+ getLike_id(): int
+ HotspotLike(int, int, int)
Pet
- count: int
- pet_birth: String
- pet_id: int
- pet_nickname: String
- pet_owner: int
- pet_photo: String
- pet_sex: String
- pet_type: String
- pet_weight: int
+ getBundle(): Bundle
+ getCount(): int
+ getPet_birth(): String
+ getPet_id(): int
+ getPet_nickname(): String
+ getPet_owner(): int
+ getPet_photo(): String
+ getPet_sex(): String
+ getPet_type(): String
+ getPet_weight(): int
+ Pet(int, String, int, String, int, String, String, String, int)
+ setCount(int): void
+ setPet_birth(String): void
+ setPet_nickname(String): void
+ setPet_photo(String): void
+ setPet_sex(String): void
+ setPet_type(String): void
+ setPet_weight(int): void
RemoteDBOperationResponse
- affectedRows: int
+ isSuccess(): boolean
+ RemoteDBOperationResponse(int)
UserNotification
- com_content: String
- com_hs: int
- com_time: String
- com_user: int
- notice_comment: int
- notice_id: int
- notice_status: int
- notice_user: int
- user_nickname: String
+ getCom_content(): String
+ getCom_hs(): int
+ getCom_time(): String
+ getCom_user(): int
+ getNotice_comment(): int
+ getNotice_id(): int
+ getNotice_status(): int
+ getNotice_user(): int
+ getUser_nickname(): String
+ setCom_content(String): void
+ setCom_hs(int): void
+ setCom_time(String): void
+ setCom_user(int): void
+ setNotice_comment(int): void
+ setNotice_id(int): void
+ setNotice_status(int): void
+ setNotice_user(int): void
+ setUser_nickname(String): void
+ UserNotification(int, int, String, int, int, String, int, int, String)

系统类图_10

5. 使用技术

5.1 结构化编程 Structure Programming

5.1.1 技术描述

> *结构化编程是一种编程模式，旨在通过广泛使用结构化控制流程结构的选择（if / then / else）和重复（while 和 for），块结构来提高计算机程序的清晰度、质量和开发时间。和子程序相比，使用简单的测试和跳转，如去声明，但这可能导致可能难以遵循和维护的“ 意大利面代码 ”。*
>
> ——引用自结构化编程 - 维基百科

5.1.2 示例代码

模块：客户端模块
代码位置：.\app\src\main\java\com\example\gypc\petsday\MainActivity.java
说明：用于判断从哪个页面点击返回跳转而来，从而显示对应的页面

```
@Override
protected void onActivityResult(int reqCode, int resCode, Intent dataIntent) {
    if (reqCode == LOGIN_REQ_CODE) {
        if (resCode == LoginRegisterNavigateActivity.LOGIN_OK) {
            init();
        } else if (resCode == LoginRegisterNavigateActivity.QUIT_RES_CODE) {
            finish();
        }
    } else if (reqCode == ADD_PET_REQ_CODE) {
        if (resCode == NewpetActivity.SUCCESS_RES_CODE) {
            MineFragment.getInstance().addPet(dataIntent.getExtras());
        }
    } else if (reqCode == EDIT_PET_CODE) {
        if (resCode == NewpetActivity.SUCCESS_RES_CODE) {
            MineFragment.getInstance().updatePet(dataIntent.getExtras());
        }
    } else if (reqCode == NEW_HOTSPOT_REQ_CODE) {
        if (resCode == PublishActivity.PUBLISH_SUCCESS) {
            HotSpotFragment.getInstance().addHotspot();
        }
    } else if (reqCode == HOTSPOT_DETAIL_REQ_CODE) {
        if (resCode == HotSpotDetailActivity.UPDATE_HOTSPOT_ITEM_SUCCESS) {
            HotSpotFragment.getInstance().updateDataFromDetailPage(dataIntent.getExtras());
        }
    }
}
```

5.2 面向对象的编程 Object-Oriented Programming

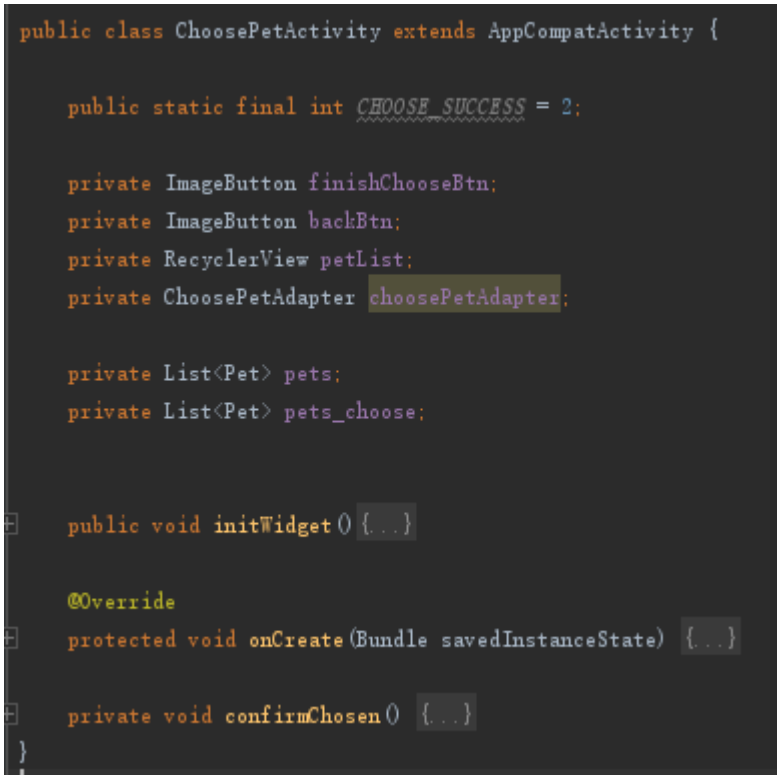
5.2.1 技术描述

> *面向对象编程（OOP）是一种基于“对象”概念的编程范例，它可能包含数据，以字段的形式，通常称为属性; 和程序形式的代码，通常称为方法。对象的一个特征是对象的过程可以访问并经常修改与它们相关联的对象的数据字段（对象具有“this”或“self”的概念）。在 OOP 中，计算机程序是通过使它们脱离彼此交互的对象而设计的。有 OOP 语言的多样性显著，但最流行的是基于类的，这意味着对象实例的类，它通常也决定了他们的类型。*
>
> ——引用自面向对象的编程 - 维基百科

5.2.2 示例代码

模块：客户端模块
代码位置：.\app\src\main\java\com\example\gypc\petsday\ChoosePetActivity.java

说明：用于客户端开发的 java 语言是重要的面向对象语言，因此几乎所有类均遵循面向对象编程技术



```
public class ChoosePetActivity extends AppCompatActivity {

    public static final int CHOOSE_SUCCESS = 2;

    private ImageButton finishChooseBtn;
    private ImageButton backBtn;
    private RecyclerView petList;
    private ChoosePetAdapter choosePetAdapter;

    private List<Pet> pets;
    private List<Pet> pets_choose;


    public void initWidget() {...}

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    private void confirmChosen() {...}
}
```