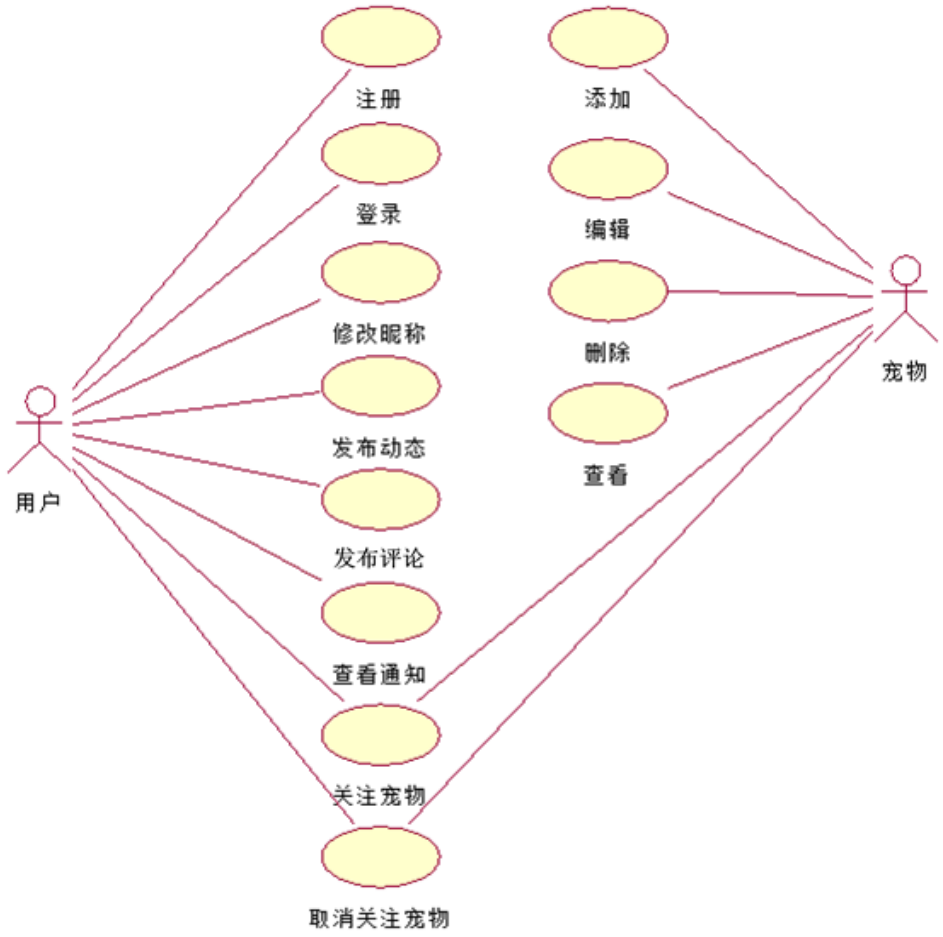
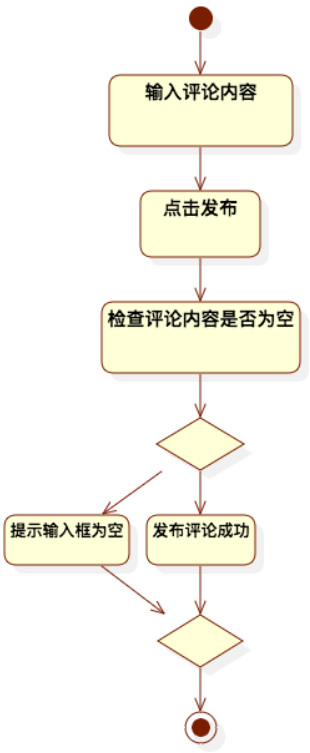
软件设计文档

一、模型图及其说明

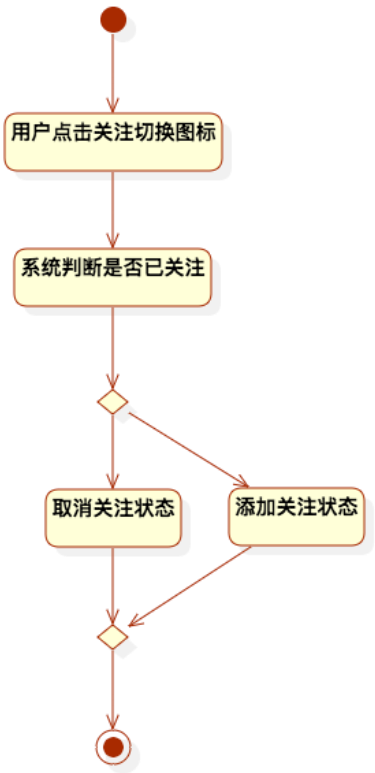
1. 用例图



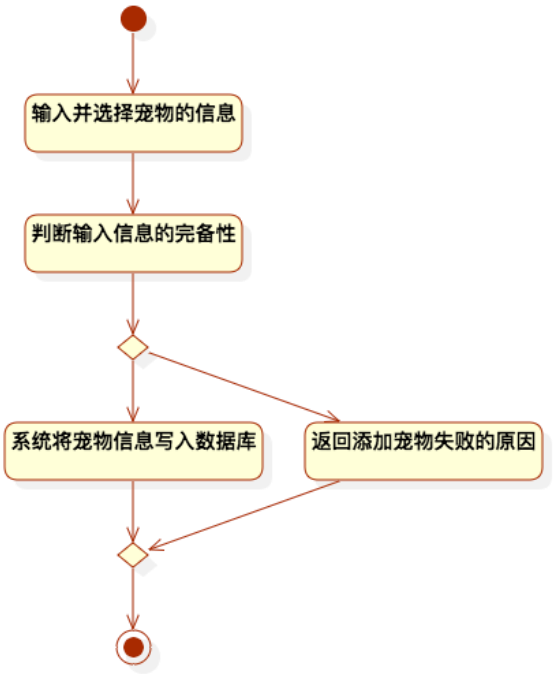
2. 活动图



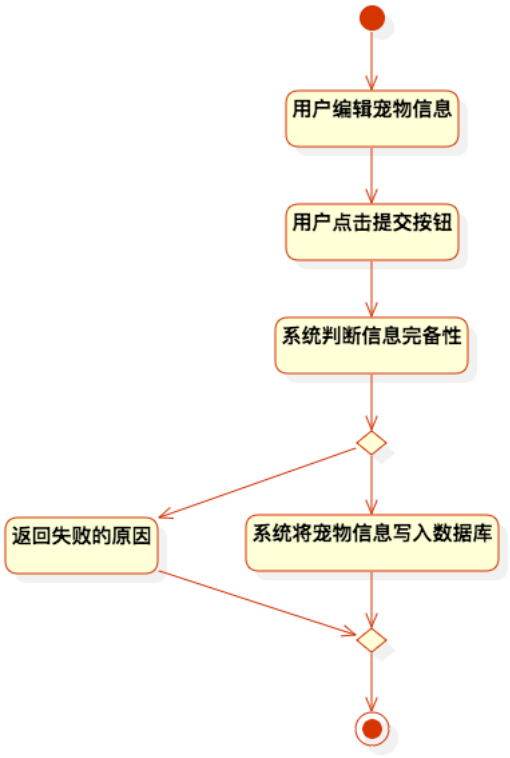
发布评论活动图



关注宠物活动图、取消关注宠物活动图

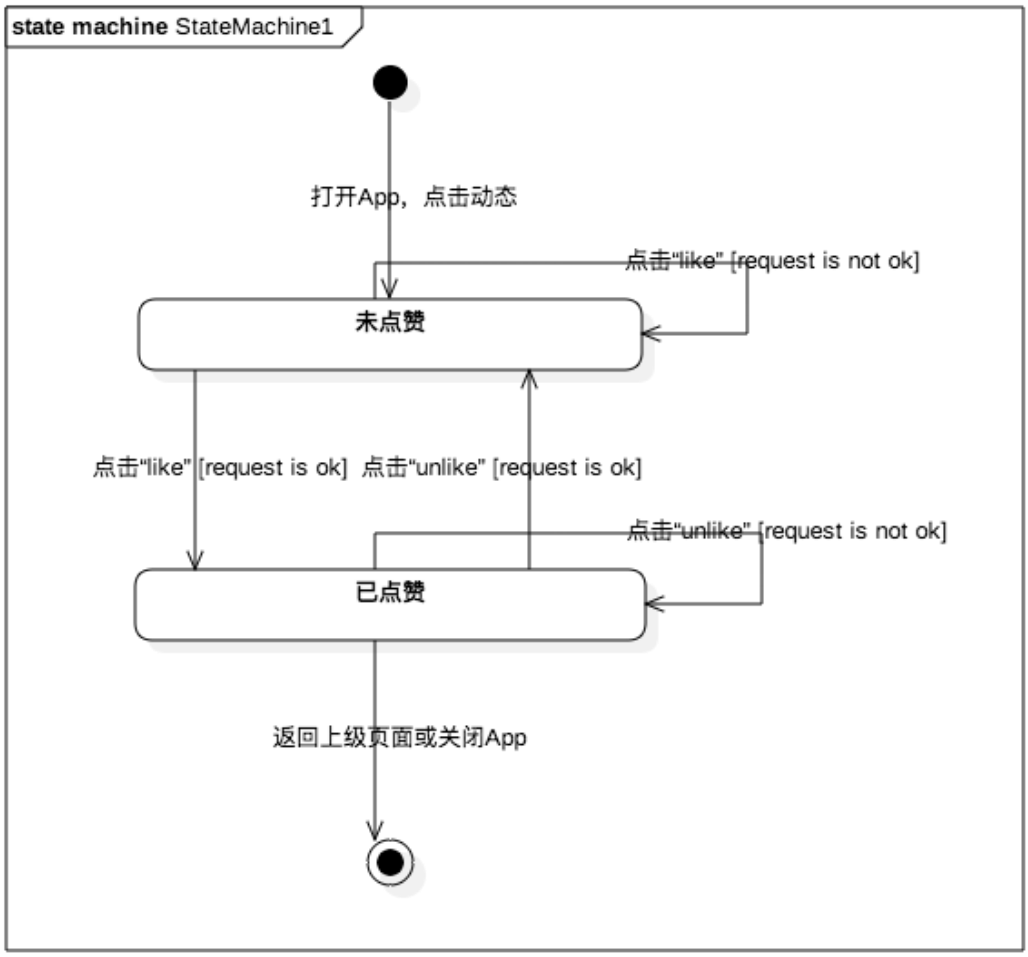


添加宠物活动图



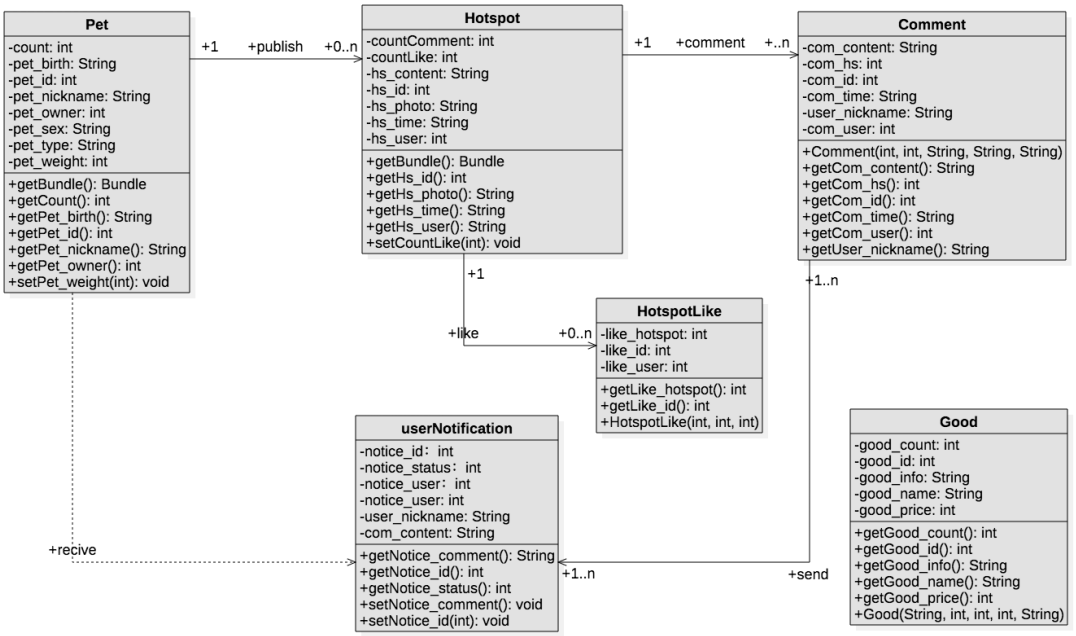
编辑宠物活动图

3. 状态图



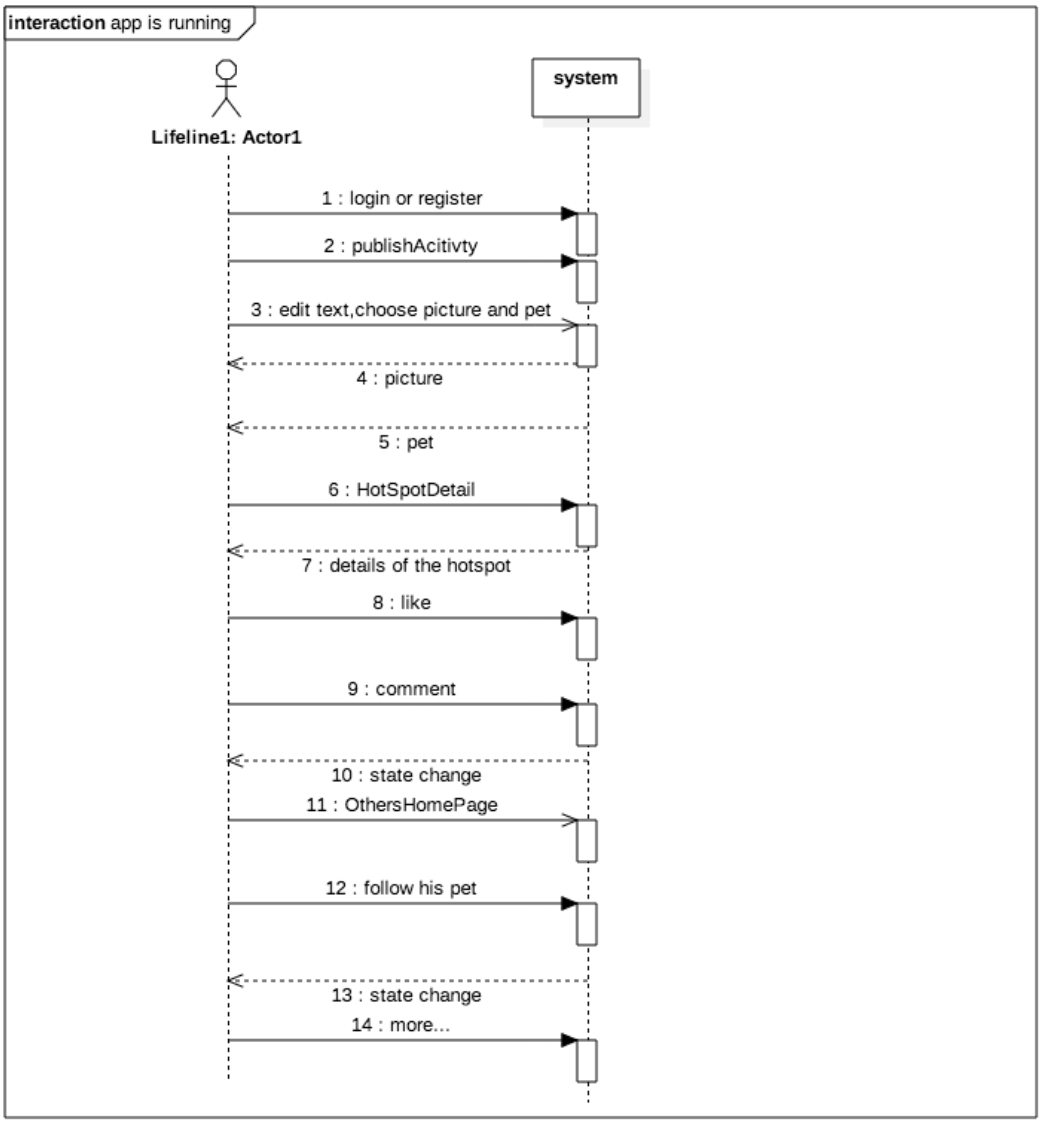
使用上图的状态图，来帮助我们描述用户对某条动态进行点赞，或取消点赞的状态的变化

4. 领域模型图



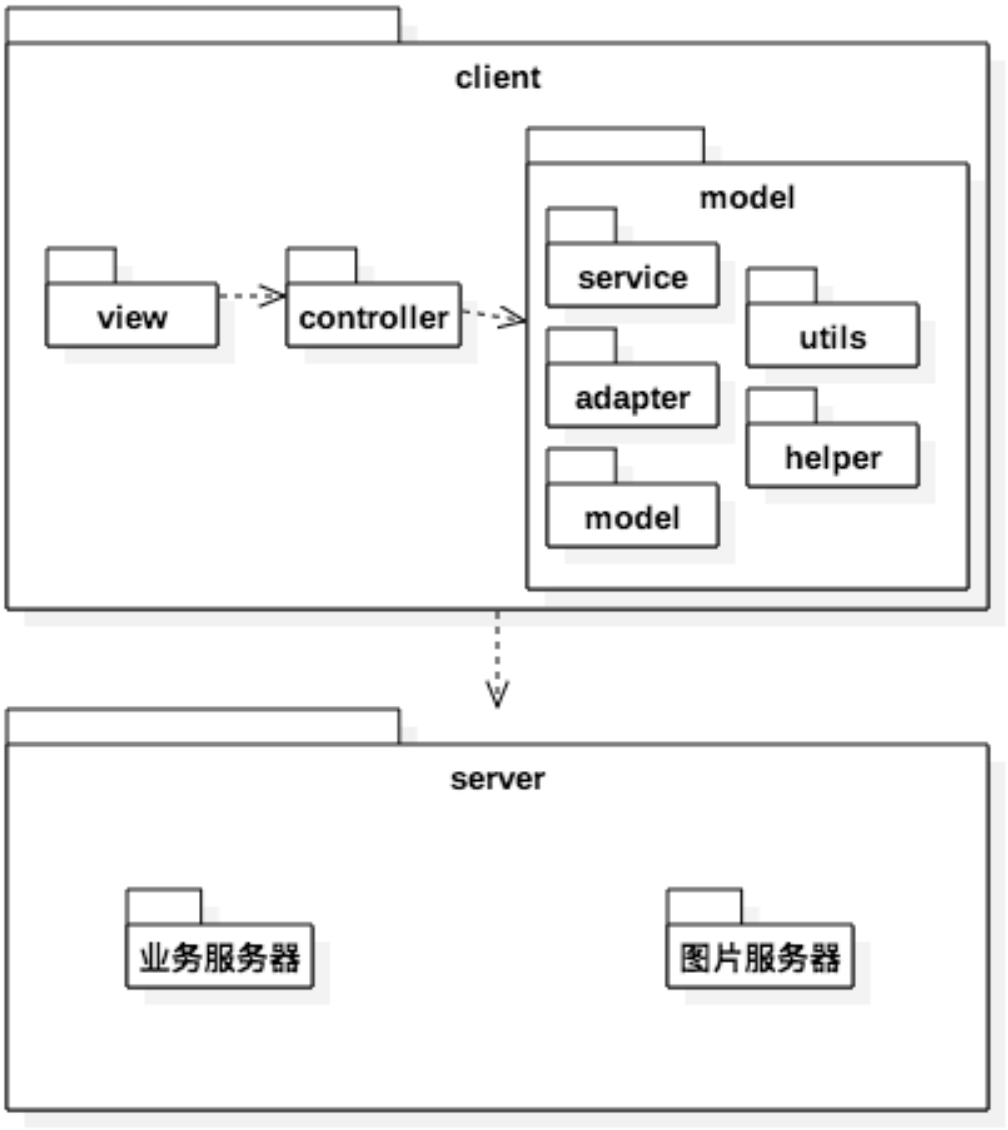
这里描述的是各个类初步的属性值和相应的联系。通过领域模型，可以理清每个类之间的联系，这将帮助我们定义他们的成员变量，也可以加速我们的开发过程。在这个图中，宠物关联动态，动态关联评论和点赞，评论则关联通知，很直观地显示了我们的设计思路。（为了更好凸显类间关系，某些类的属性和操作有所省略）

5. 系统顺序图



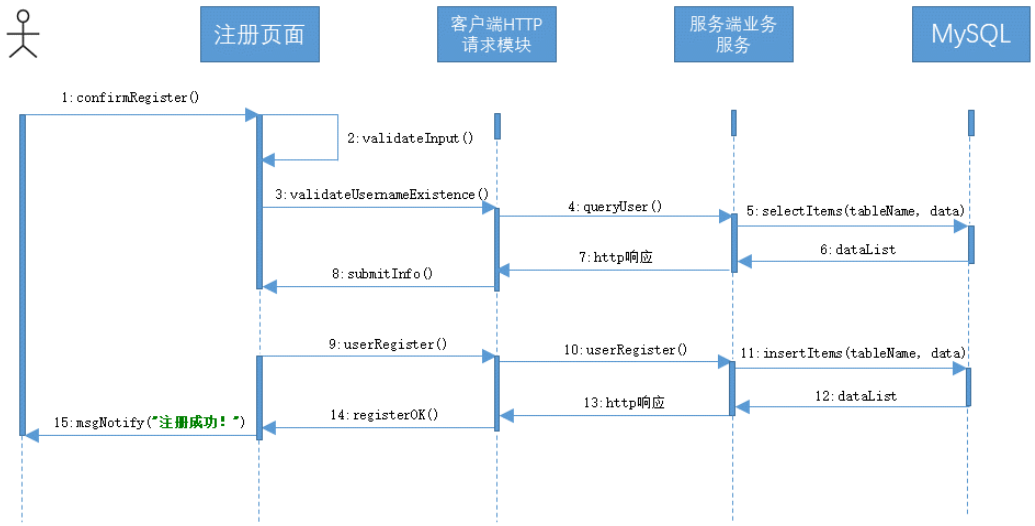
上面描述用户与系统之间交互的一系列行为，包括了发布动态，查看动态详情，点赞，评论以及到他人主页去浏览，关注他的宠物等等。

6. 包图

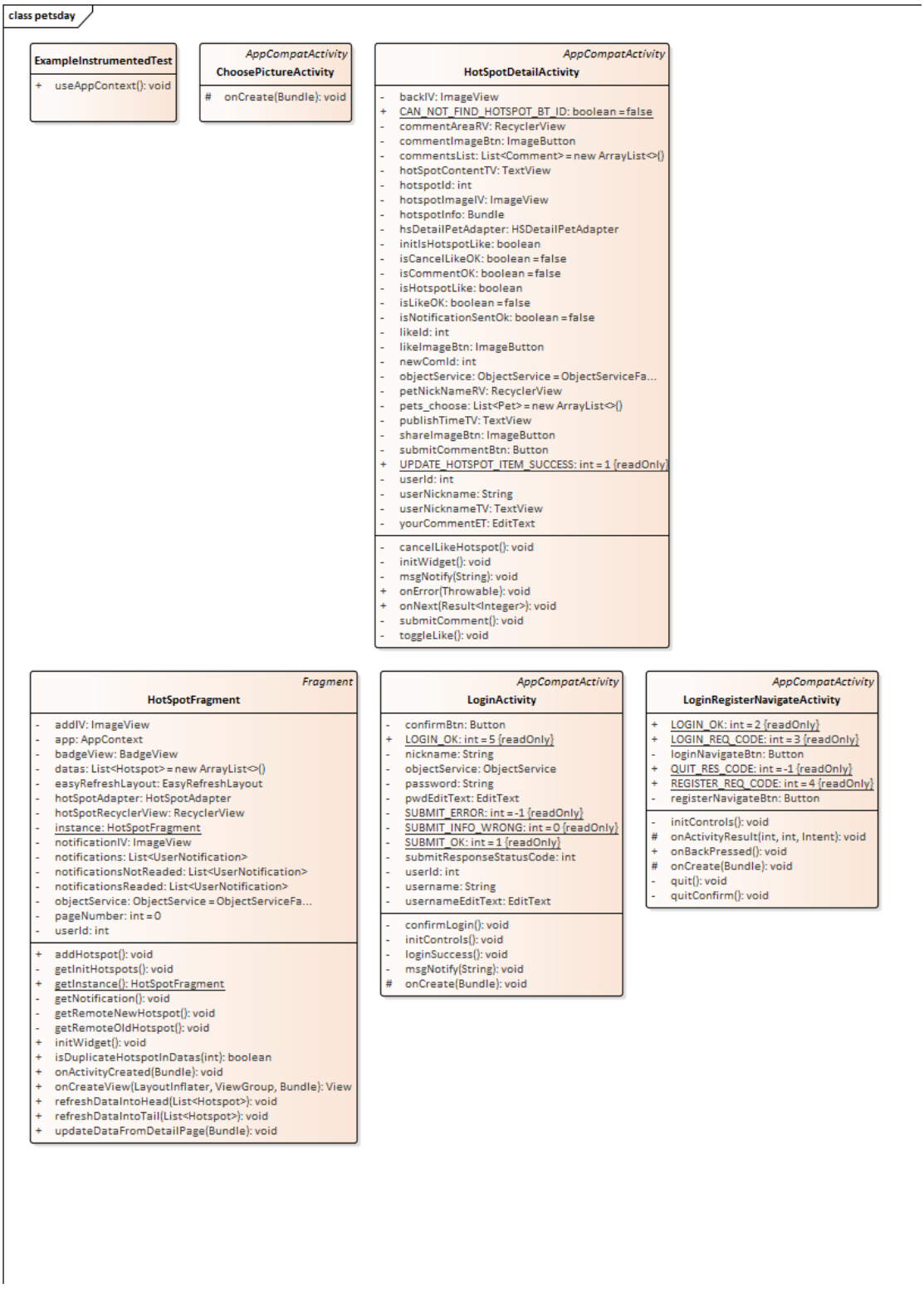


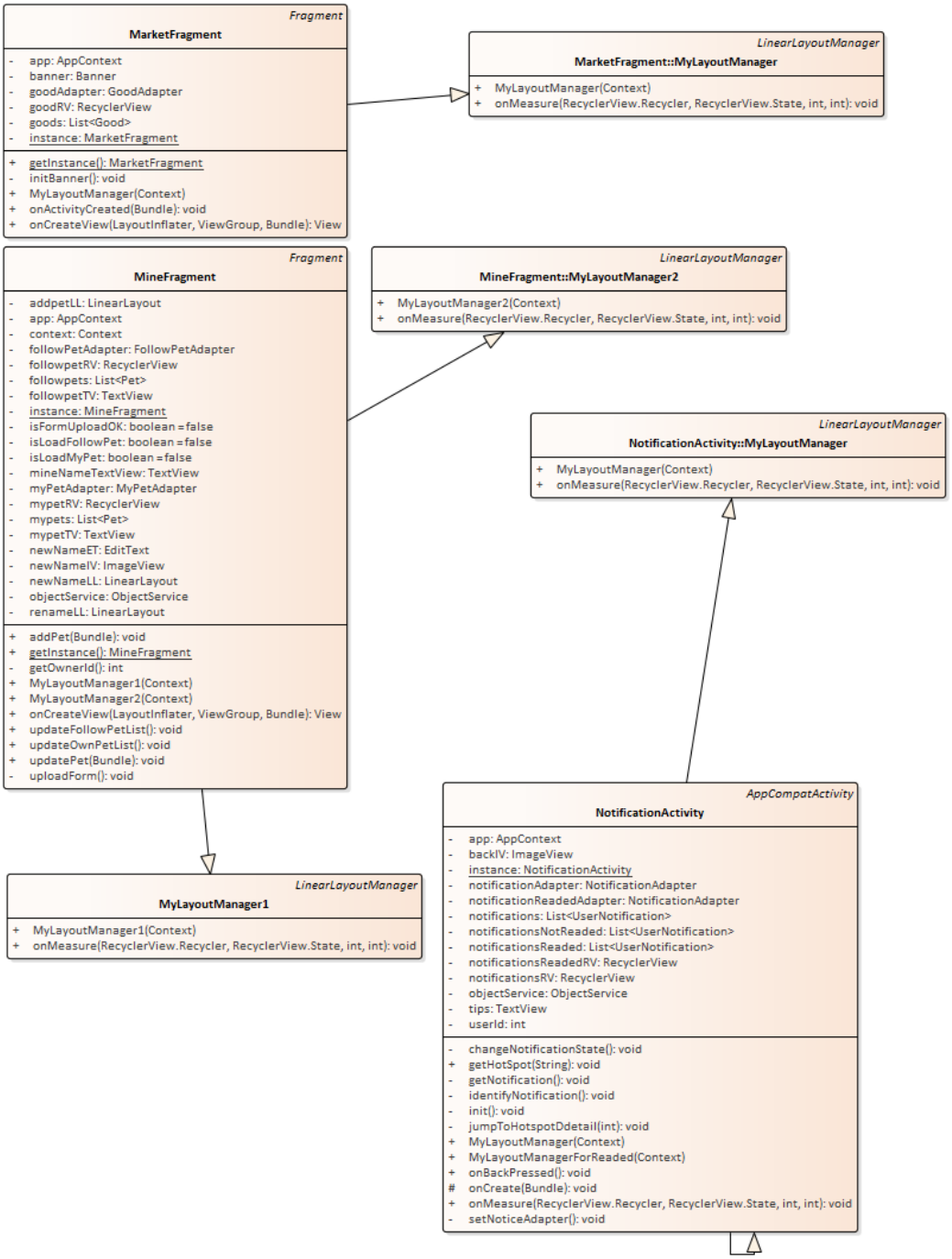
整体架构为 C/S 架构，其中客户端那边又采用了 MVC 架构。

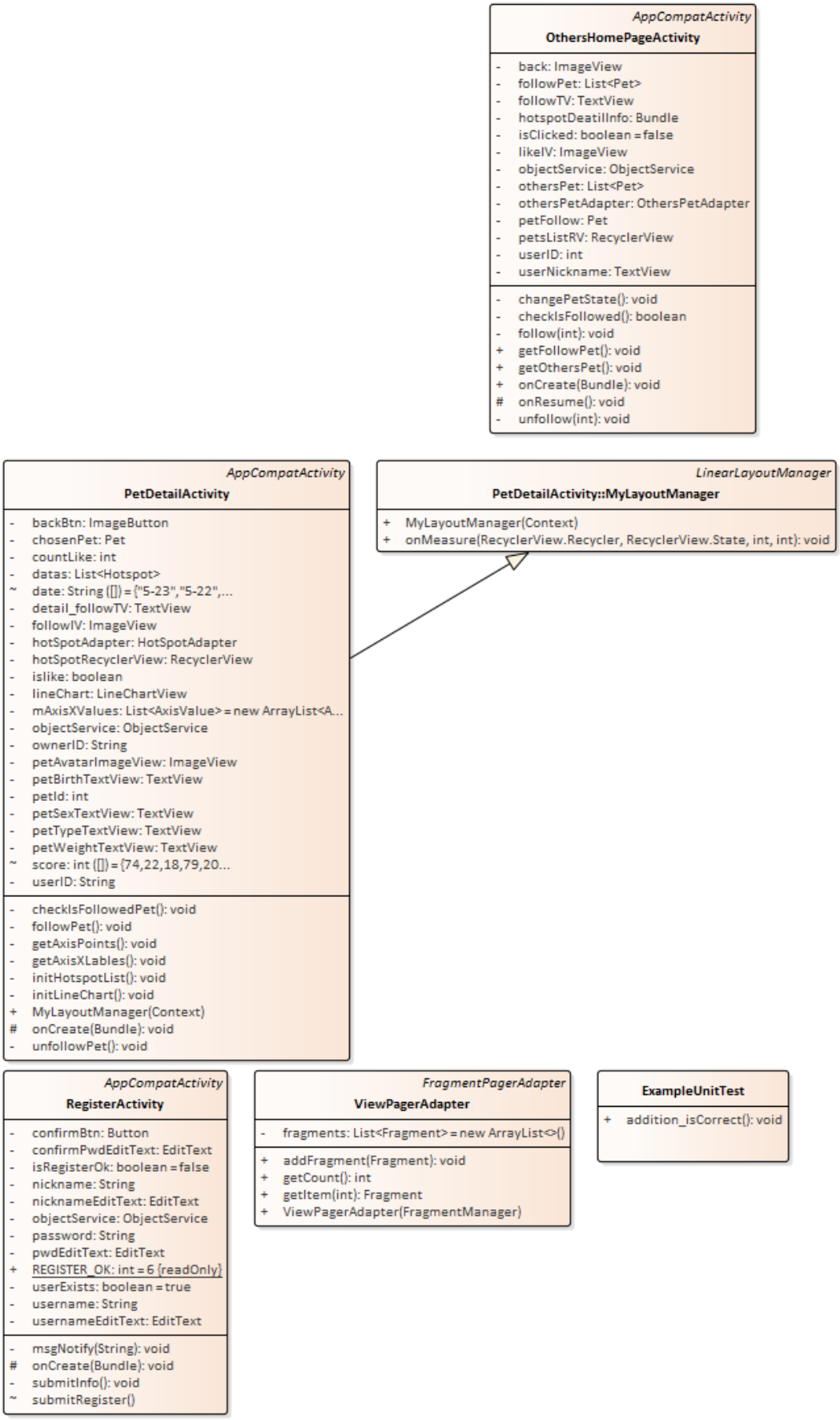
7. 顺序图

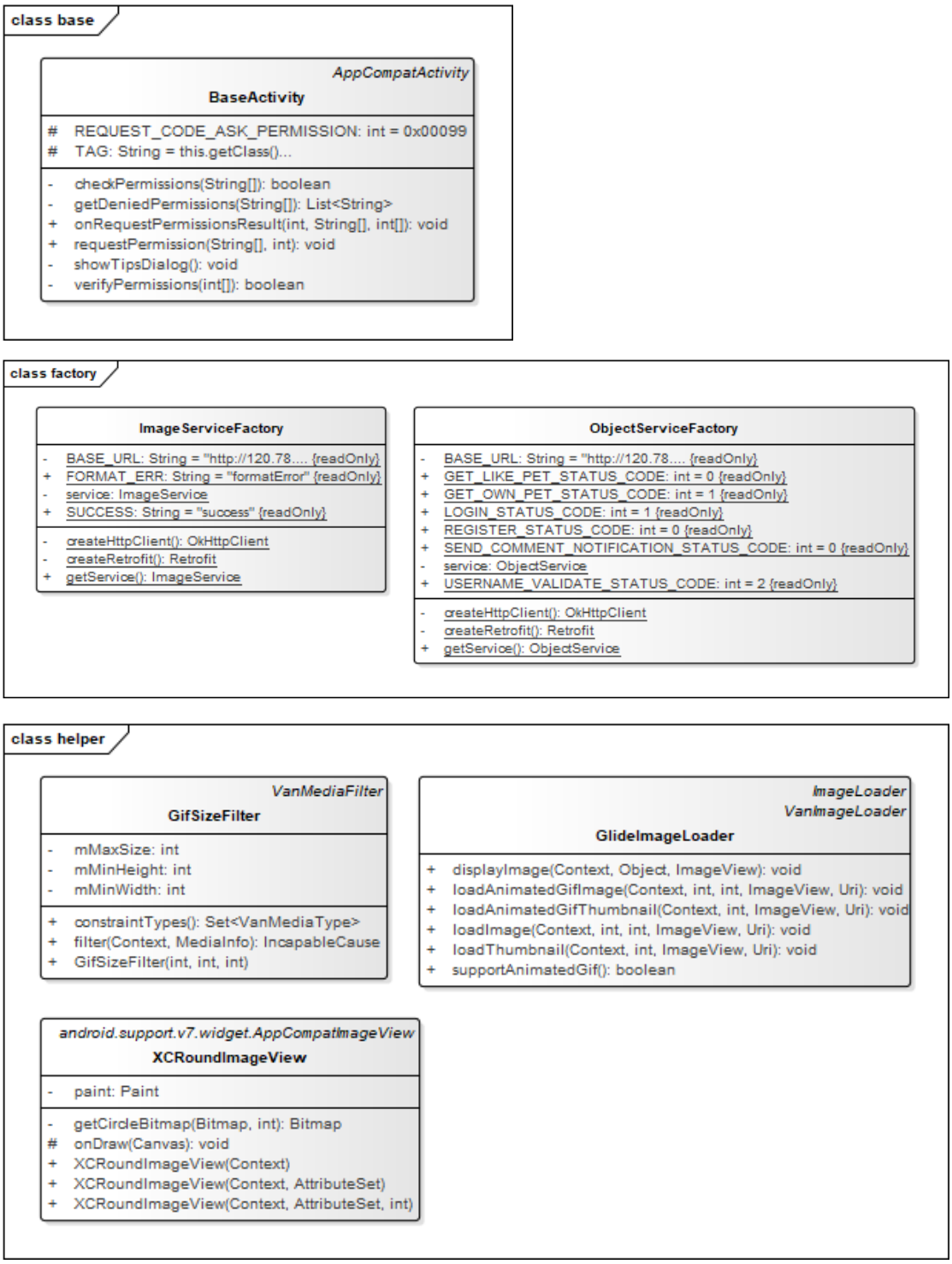

注册页面
客户端HTTP
请求模块
服务端业务
服务
MySQL
1: confirmRegister()
2: validateInput()
3: validateUsernameExistence()
4: queryUser()
5: selectItems(tableName, data)
6: dataList
7: http响应
8: submitInfo()
9: userRegister()
10: userRegister()
11: insertItems(tableName, data)
12: dataList
13: http响应
14: registerOK()
15: msgNotify("注册成功！")

8. 类图

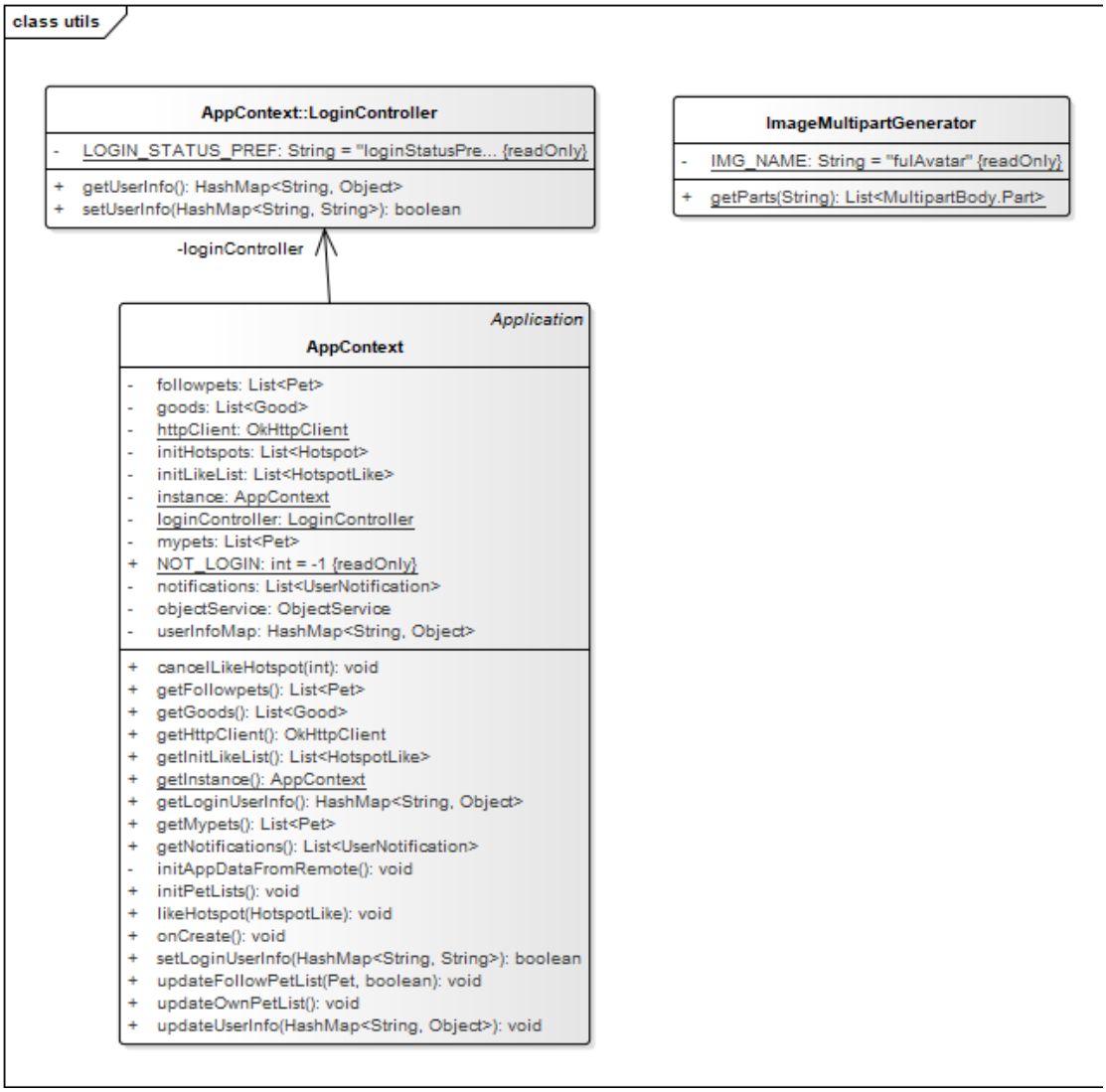

class petsday
ExampleInstrumentedTest
+ useAppContext(): void
AppCompatActivity
ChoosePictureActivity
onCreate(Bundle): void
AppCompatActivity
HotSpotDetailActivity
- backIV: ImageView
+ CAN_NOT_FIND_HOTSPOT_BT_ID: boolean = false
- commentAreaRV: RecyclerView
- commentImageBtn: ImageButton
- commentsList: List<Comment> = new ArrayList<>()
- hotSpotContentTV: TextView
- hotspotId: int
- hotspotImageIV: ImageView
- hotspotInfo: Bundle
- hsDetailPetAdapter: HSDetailPetAdapter
- initIsHotspotLike: boolean
- isCancelLikeOK: boolean = false
- isCommentOK: boolean = false
- isHotspotLike: boolean
- isLikeOK: boolean = false
- isNotificationSentOk: boolean = false
- likeId: int
- likeImageBtn: ImageButton
- newComId: int
- objectService: ObjectService = ObjectServiceFa...
- petNickNameRV: RecyclerView
- pets_choose: List<Pet> = new ArrayList<>()
- publishTimeTV: TextView
- shareImageBtn: ImageButton
- submitCommentBtn: Button
+ UPDATE_HOTSPOT_ITEM_SUCCESS: int = 1 {readOnly}
- userId: int
- userNickname: String
- userNicknameTV: TextView
- yourCommentET: EditText
- cancelLikeHotspot(): void
- initWidget(): void
- msgNotify(String): void
+ onError(Throwable): void
+ onNext(Result<Integer>): void
- submitComment(): void
- toggleLike(): void
Fragment
HotSpotFragment
- addIV: ImageView
- app: AppContext
- badgeView: BadgeView
- datas: List<Hotspot> = new ArrayList<>()
- easyRefreshLayout: EasyRefreshLayout
- hotSpotAdapter: HotSpotAdapter
- hotSpotRecyclerView: RecyclerView
- instance: HotSpotFragment
- notificationIV: ImageView
- notifications: List<UserNotification>
- notificationsNotReaded: List<UserNotification>
- notificationsReaded: List<UserNotification>
- objectService: ObjectService = ObjectServiceFa...
- pageNumber: int = 0
- userId: int
+ addHotspot(): void
- getInitHotspots(): void
+ getInstance(): HotSpotFragment
- getNotification(): void
- getRemoteNewHotspot(): void
- getRemoteOldHotspot(): void
+ initWidget(): void
+ isDuplicateHotspotInDatas(int): boolean
+ onActivityCreated(Bundle): void
+ onCreateView(LayoutInflater, ViewGroup, Bundle): View
+ refreshDataIntoHead(List<Hotspot>): void
+ refreshDataIntoTail(List<Hotspot>): void
+ updateDataFromDetailPage(Bundle): void
AppCompatActivity
LoginActivity
- confirmBtn: Button
+ LOGIN_OK: int = 5 {readOnly}
- nickname: String
- objectService: ObjectService
- password: String
- pwdEditText: EditText
- SUBMIT_ERROR: int = -1 {readOnly}
- SUBMIT_INFO_WRONG: int = 0 {readOnly}
- SUBMIT_OK: int = 1 {readOnly}
- submitResponseStatusCode: int
- userId: int
- username: String
- usernameEditText: EditText
- confirmLogin(): void
- initControls(): void
- loginSuccess(): void
- msgNotify(String): void
onCreate(Bundle): void
AppCompatActivity
LoginRegisterNavigateActivity
+ LOGIN_OK: int = 2 {readOnly}
+ LOGIN_REQ_CODE: int = 3 {readOnly}
- loginNavigateBtn: Button
+ QUIT_RES_CODE: int = -1 {readOnly}
+ REGISTER_REQ_CODE: int = 4 {readOnly}
- registerNavigateBtn: Button
- initControls(): void
onActivityResult(int, int, Intent): void
+ onBackPressed(): void
# onCreate(Bundle): void
- quit(): void
- quitConfirm(): void

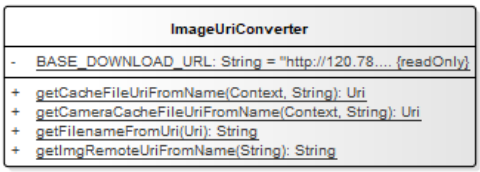

Fragment
MarketFragment
- app: AppContext
- banner: Banner
- goodAdapter: GoodAdapter
- goodRV: RecyclerView
- goods: List<Good>
- instance: MarketFragment
+ getInstance(): MarketFragment
- initBanner(): void
+ MyLayoutManager(Context)
+ onActivityCreated(Bundle): void
+ onCreateView(LayoutInflater, 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

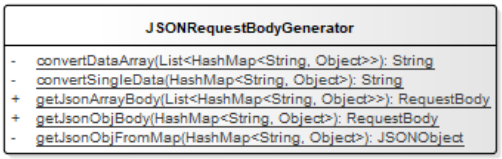

AppCompatActivity
OthersHomePageActivity
- back: ImageView
- followPet: List<Pet>
- followTV: TextView
- hotspotDeatilInfo: Bundle
- isClicked: boolean = false
- likeIV: ImageView
- objectService: ObjectService
- othersPet: List<Pet>
- othersPetAdapter: OthersPetAdapter
- petFollow: Pet
- petsListRV: 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

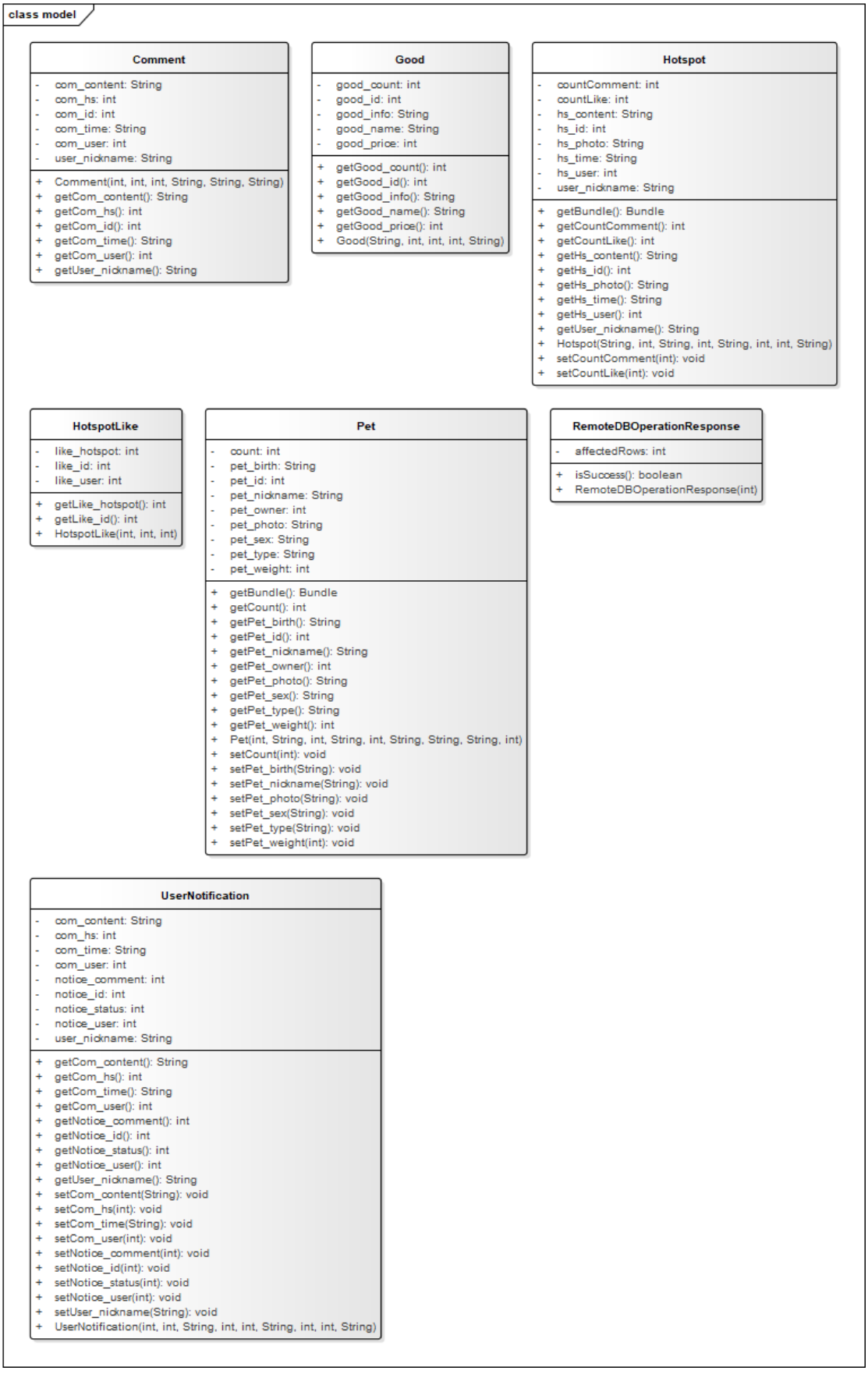

class base
AppCompatActivity
BaseActivity
REQUEST_CODE_ASK_PERMISSION: int = 0x00099
TAG: String = this.getClass()...
- checkPermissions(String[]): boolean
- getDeniedPermissions(String[]): List<String>
+ onRequestPermissionsResult(int, String[], int[]): void
+ requestPermission(String[], int): void
- showTipsDialog(): void
- verifyPermissions(int[]): boolean
class factory
ImageServiceFactory
- BASE_URL: String = "http://120.78.... {readOnly}
+ FORMAT_ERR: String = "formatError" {readOnly}
- service: ImageService
+ SUCCESS: String = "success" {readOnly}
- createHttpClient(): OkHttpClient
- createRetrofit(): Retrofit
+ getService(): ImageService
ObjectServiceFactory
- BASE_URL: String = "http://120.78.... {readOnly}
+ GET_LIKE_PET_STATUS_CODE: int = 0 {readOnly}
+ GET_OWN_PET_STATUS_CODE: int = 1 {readOnly}
+ LOGIN_STATUS_CODE: int = 1 {readOnly}
+ REGISTER_STATUS_CODE: int = 0 {readOnly}
+ SEND_COMMENT_NOTIFICATION_STATUS_CODE: int = 0 {readOnly}
- service: ObjectService
+ USERNAME_VALIDATE_STATUS_CODE: int = 2 {readOnly}
- createHttpClient(): OkHttpClient
- createRetrofit(): Retrofit
+ getService(): ObjectService
class helper
VanMediaFilter
GifSizeFilter
- mMaxSize: int
- mMinHeight: int
- mMinWidth: int
+ constraintTypes(): Set<VanMediaType>
+ filter(Context, MediaInfo): IncapableCause
+ GifSizeFilter(int, int, int)
ImageLoader
VanImageLoader
GlideImageLoader
+ displayImage(Context, Object, ImageView): void
+ loadAnimatedGifImage(Context, int, int, ImageView, Uri): void
+ loadAnimatedGifThumbnail(Context, int, ImageView, Uri): void
+ loadImage(Context, int, int, ImageView, Uri): void
+ loadThumbnail(Context, int, ImageView, Uri): void
+ supportAnimatedGif(): boolean
android.support.v7.widget.AppCompatImageView
XCRoundImageView
- paint: Paint
- getCircleBitmap(Bitmap, int): Bitmap
onDraw(Canvas): void
+ XCRoundImageView(Context)
+ XCRoundImageView(Context, AttributeSet)
+ XCRoundImageView(Context, AttributeSet, int)


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>>

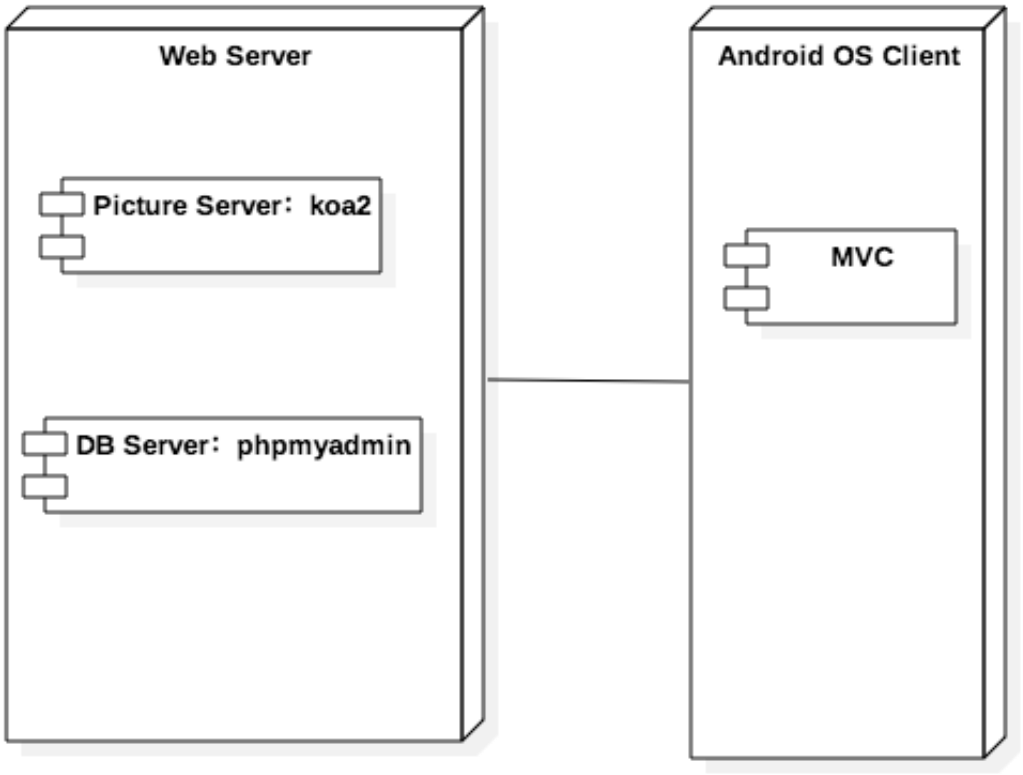

class utils
AppContext::LoginController
- LOGIN_STATUS_PREF: String = "loginStatusPre... {readOnly}
+ getUserInfo(): HashMap<String, Object>
+ setUserInfo(HashMap<String, String>): boolean
-loginController
ImageMultipartGenerator
- IMG_NAME: String = "fulAvatar" {readOnly}
+ getParts(String): List<MultipartBody.Part>
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


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

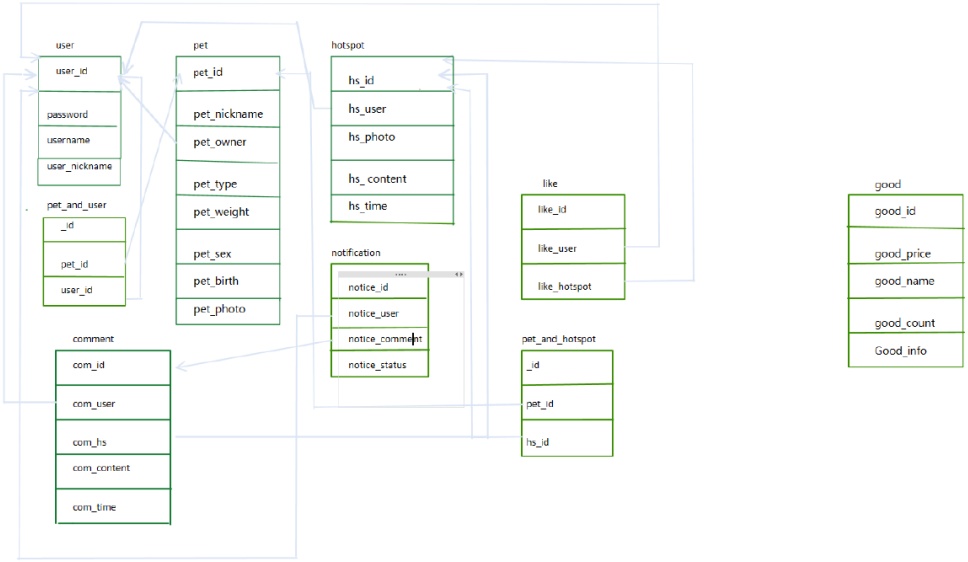

class model
Comment
- com_content: String
- com_hs: int
- com_id: int
- com_time: String
- com_user: int
- user_nickname: String
+ Comment(int, int, int, String, String, String)
+ getCom_content(): String
+ getCom_hs(): int
+ getCom_id(): int
+ getCom_time(): String
+ getCom_user(): int
+ getUser_nickname(): 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)

9. 部署图



说明了我们程序的部署方案。我们使用了远程 web 服务器，主要包含，使用 koa2 来部署的图片服务器，以及使用 phpmyadmin 来部署的数据库服务器。最右的方框显示了我们的 APP 运行在安卓系统上，使用 MVC 架构等信息。

10.数据库表格



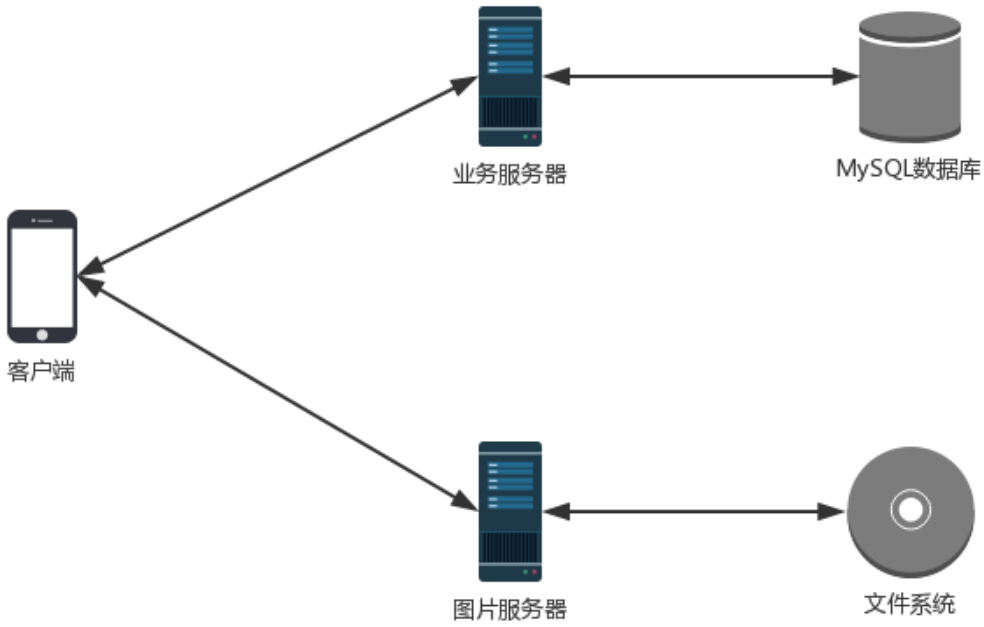
二、技术选型理由

1. Android 开发平台：平台开放性，兼容性好，开发门槛低，市场需求大。

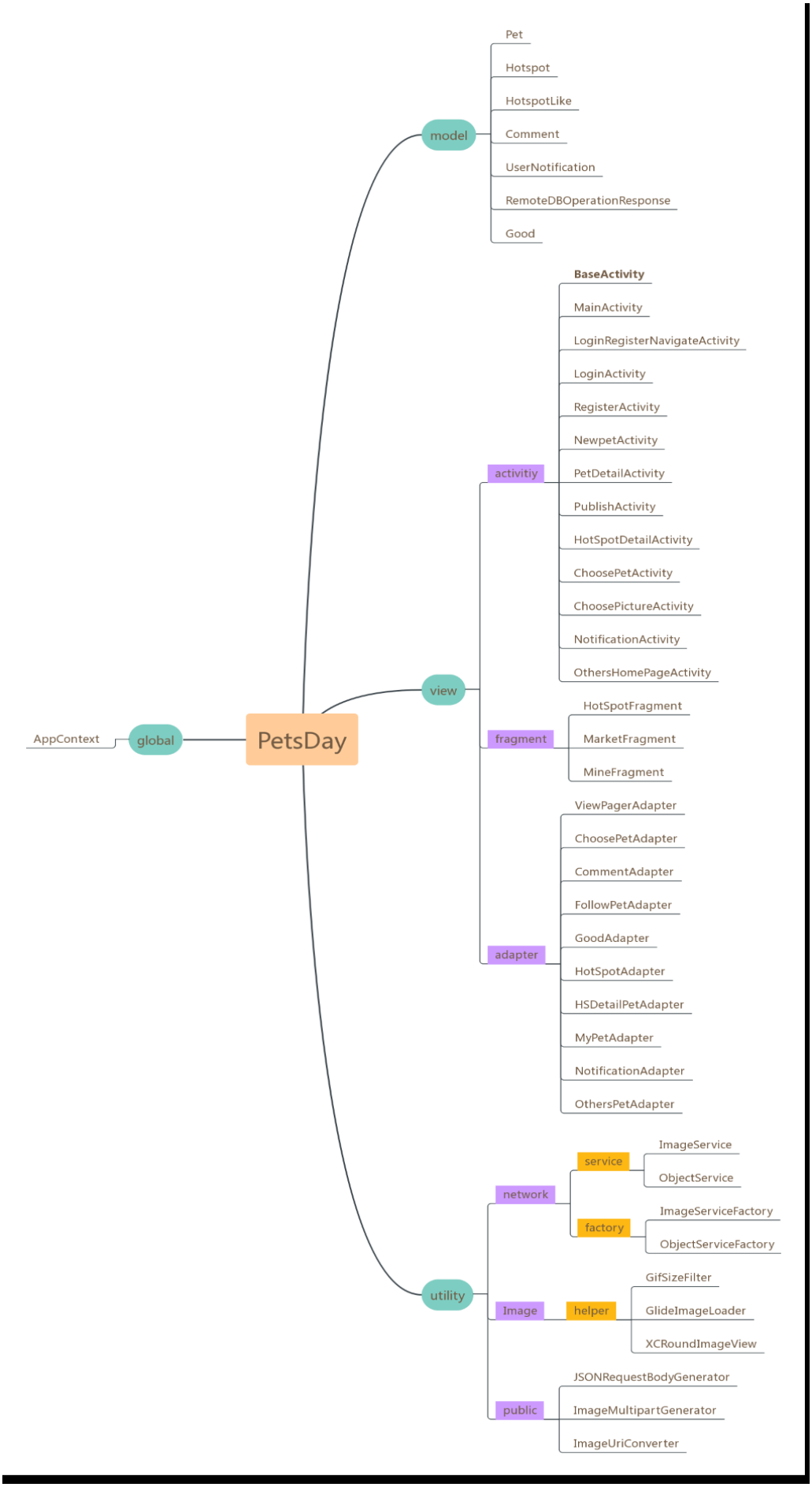
2. Mysql 数据库：云端数据库保持数据统一、易于用户管理，MySQL 是最流行的关系型数据库管理系统。

3. Golang 服务端：易于实现前后端分离，实现 restfulAPI，利于敏捷开发

三、模块划分



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。


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

本次项目当中，小组的分工为，三个人负责 Android 客户端，一个人负责后台。客户端那边划分了 model 数据层，负责与服务端交互并处理数据，view 层负责页面的切换和数据展示，还有相应的 controller 封装业务逻辑，MVC 架构提高了代码的可读性，降低后期维护的成本。后端负责提供 api 接口，并根据业务逻辑进行数据库的建表和查询，根据客户端的请求返回相应的数据。

四、技术难点以及解决方案

1. 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。

2. 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。

3. 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。

4. 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。

5. 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。

6. 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。

```
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`;
}
```