电影资讯apps概要设计规约

[电影资讯app概要设计规约 1](#_Toc534141526)

[1引言 2](#_Toc534141527)

[1.1概要设计依据 2](#_Toc534141528)

[1.2 参考资料 3](#_Toc534141529)

[1.3假定与约束 3](#_Toc534141530)

[2 概要设计 3](#_Toc534141531)

[2.1系统总体架构设计 3](#_Toc534141532)

[2.2系统软件结构设计 4](#_Toc534141533)

[2.2.1登录功能的类调用关系图 4](#_Toc534141534)

[2.2.2注册功能的类调用关系图 5](#_Toc534141535)

[2.2.3 找片Tab的类调用关系图 5](#_Toc534141536)

[2.2.4 热映Tab 的类调用关系图 6](#_Toc534141537)

[2.2.5 “我的信息”Tab类调用关系图 6](#_Toc534141538)

[2.2.6电影列表类调用关系图 6](#_Toc534141539)

[2.2.7 电影详情类调用关系图 7](#_Toc534141540)

[2.2.8 电影关键词搜索类调用关系图 8](#_Toc534141541)

[2.2.9电影类型搜索类调用关系图 8](#_Toc534141542)

[2.2.10电影信息统计类调用关系图 9](#_Toc534141543)

[2.2.11 用户信息统计类调用关系图 9](#_Toc534141544)

[2.2.12 服务端软件体系结构图 10](#_Toc534141545)

[2.3接口设计 10](#_Toc534141546)

[2.3.1 Activity类之间的接口 10](#_Toc534141547)

[2.3.2 Presenter, View, Model之间绑定的接口 10](#_Toc534141548)

[2.3.3 为RecyclerView(列表控件)绑定Adapter(数据适配器)的接口 10](#_Toc534141549)

[2.3.4 网络访问组件Retrofit的接口 11](#_Toc534141550)

[2.3.5 本地持续化存储访问接口 11](#_Toc534141551)

[2.3.6 APP客户端与服务端的接口 11](#_Toc534141552)

[2.3.6.1 登录接口 11](#_Toc534141553)

[2.3.6.2 注册接口 11](#_Toc534141554)

[2.3.6.3 收藏接口 11](#_Toc534141555)

[2.3.6.4 取消收藏接口 12](#_Toc534141556)

[2.3.6.5 电影收藏与否接口 12](#_Toc534141557)

[2.3.6.6 用户收藏列表 12](#_Toc534141558)

[2.3.6.7 增加浏览记录接口 12](#_Toc534141559)

[2.3.6.8 删除浏览记录接口 12](#_Toc534141560)

[2.3.6.9 用户浏览记录列表接口 13](#_Toc534141561)

[2.3.6.10 用户个人喜好推荐接口 13](#_Toc534141562)

[2.3.6.11 电影信息统计接口 13](#_Toc534141563)

[2.3.6.12 用户信息统计接口 13](#_Toc534141564)

[2.3.6.13 用户头像设置接口 13](#_Toc534141565)

[2.4界面设计 14](#_Toc534141566)

[2.4.1界面设计思路 14](#_Toc534141567)

[2.4.2 界面设计展示 14](#_Toc534141568)

[2.5数据库设计 16](#_Toc534141569)

[2.5.1服务端数据库 16](#_Toc534141570)

[2.5.1.1用户信息表users 16](#_Toc534141571)

[2.5.1.2用户收藏表collect 17](#_Toc534141572)

[2.5.1.3用户浏览记录表history 18](#_Toc534141573)

[2.5.1.4 电影数据表moviedata 18](#_Toc534141574)

[2.5.1.5用户画像表 typicalUser 18](#_Toc534141575)

[2.5.2 客户端本地持久性存储 19](#_Toc534141576)

[2.6系统出错处理设计 19](#_Toc534141577)

[2.6.1出错信息 19](#_Toc534141578)

[2.6.1.1客户端网络连接出错 19](#_Toc534141579)

[2.6.1.2登录注册时用户账号密码格式出错 20](#_Toc534141580)

[2.6.1.3 用户未登录时进行了需要登录的操作 20](#_Toc534141581)

[2.6.1.4 用户未登录情况下点击退出登录按钮 20](#_Toc534141582)

[2.6.1.5 用户使用管理员功能 20](#_Toc534141583)

[2.6.2 补救措施 20](#_Toc534141584)

[2.6.2.1 客户端访问网络失败 20](#_Toc534141585)

[2.6.2.2 登录注册时用户账号密码格式出错 20](#_Toc534141586)

[2.6.1.3 用户未登录时进行了需要登录的操作 20](#_Toc534141587)

[2.6.3系统维护设计 20](#_Toc534141588)

## 1引言

#### 1.1概要设计依据

a. 电影资讯app需求分析规约。

b. google官方mvp架构示例项目-todoapp。

#### 1.2 参考资料

a. 《软件工程-实践者的研究方法（原书第八版）》-罗杰S·普莱斯曼，机械工业出版社

b. 《第一行代码-Android》人民邮电出版社

#### 1.3假定与约束

a. 系统开发时间期限截止于2019年1月3日

b. iteration1：Android APP端使用java语言进行开发，使用SQLite进行本地数据库存储。

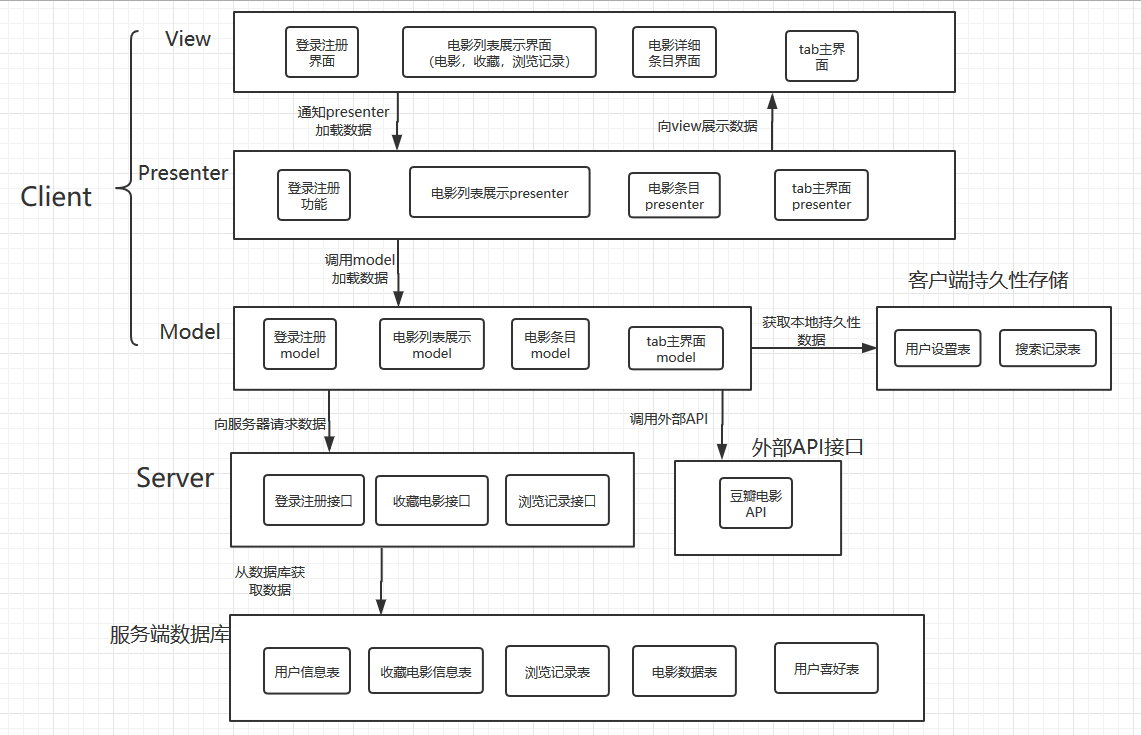
Iteration2: Android APP端使用java语言进行开发，使用sharedPreferences进行本地数据持久性存储。

c. 服务端使用Node. Js的Express框架进行开发，使用MongoDB非关系型数据库进行数据存储。

d. 开发的Android端APP需要适用于SDK版本android 4.4，API级别19及以上的android设备。

## 2 概要设计

#### 2.1系统总体架构设计



总体架构图

系统主要是使用了CS架构，并且在Client端使用了MVP架构模式将数据与UI界面进行分离。

其中Client端主要分为三层，最上层View层进行UI界面展示；第二层Presenter层则负责数据的绑定，调用model层请求数据，将获取到的数据展示在View层之上；第三层Model层进行各种数据的获取，包括请求服务端数据，外部API的调用，以及本地客户端数据库中数据获取。图中就举例显示了登录注册，列表展示等几组主要的MVP组件。

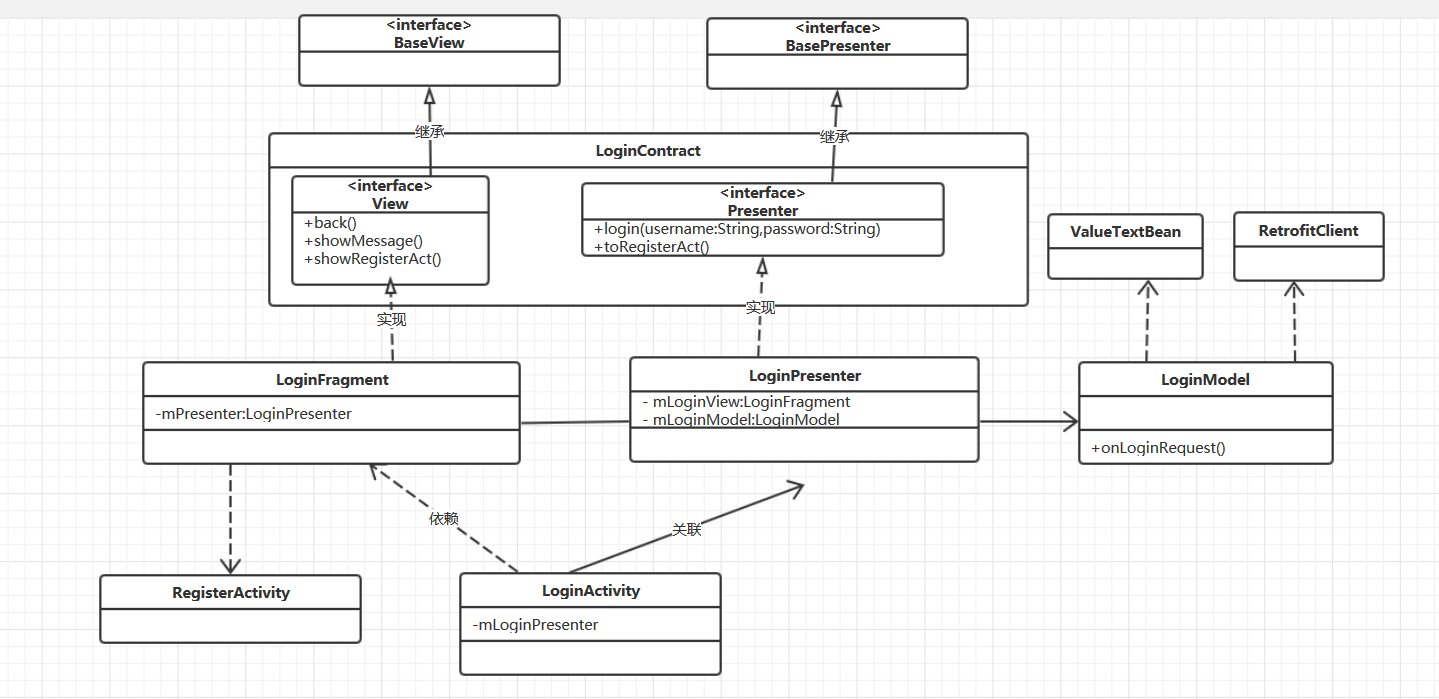
在Client端还有一个小型的sharedPreferences，用来进行数据的本地化持续性存储，用来存储用户信息等一系列本地数据。

Server端主要向客户端提供接口，响应客户端请求，对服务端数据库进行增删改查一系列操作，并将相关的数据返回给客户端。Server端主要对用户账号信息，用户收藏电影信息,用户浏览记录进行维护。

系统中还调用了一个豆瓣电影的外部API来进行电影信息的获取。

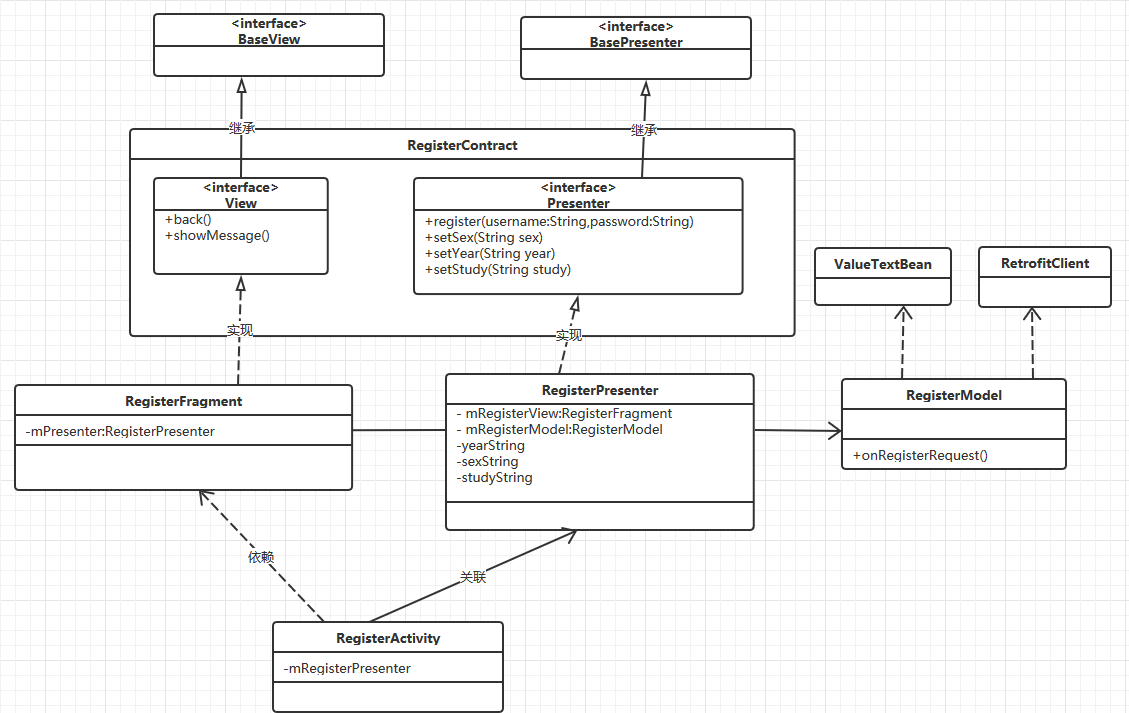
###### 2.2系统软件结构设计

###### 2.2.1登录功能的类调用关系图



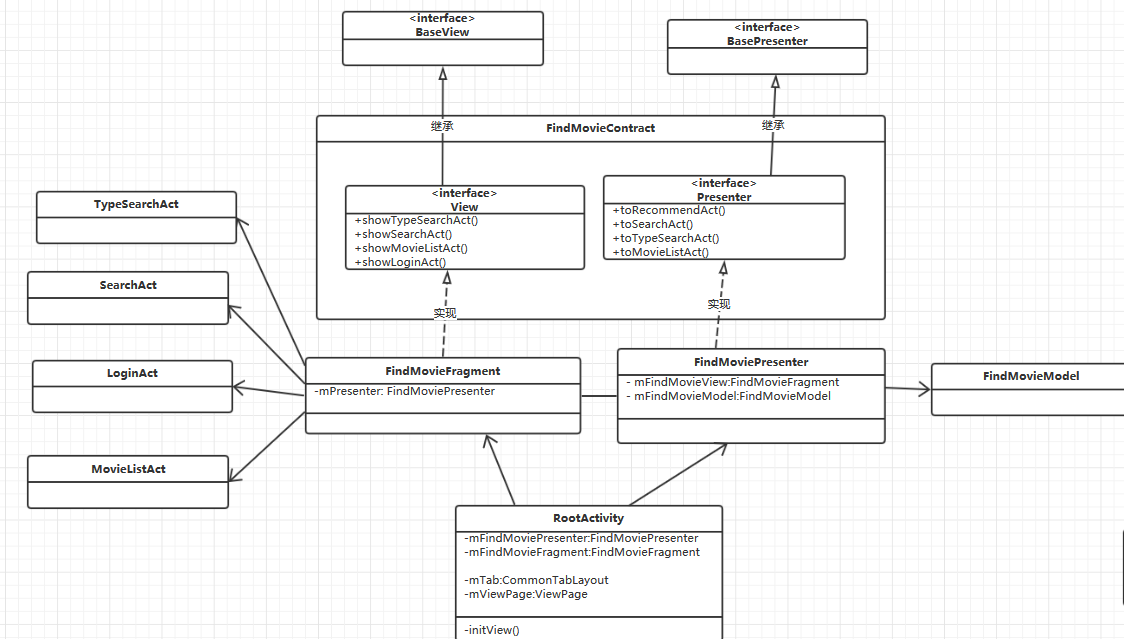
本系统中的登录功能，主要是在LoginActivity中进行MVP三部分的初始化与关系组织，当用户输入相应信息后点击登录饥按钮，会调用Presenter的login函数，对用户名和密码基本格式进行判断（包括是否为空，长度过长等），接着调用Model的onLoginRequest函数，生成RetrofitClient访问服务器，用ValueTextBean接受登录信息，再由presenter调用view来显示对应信息。

###### 2.2.2注册功能的类调用关系图



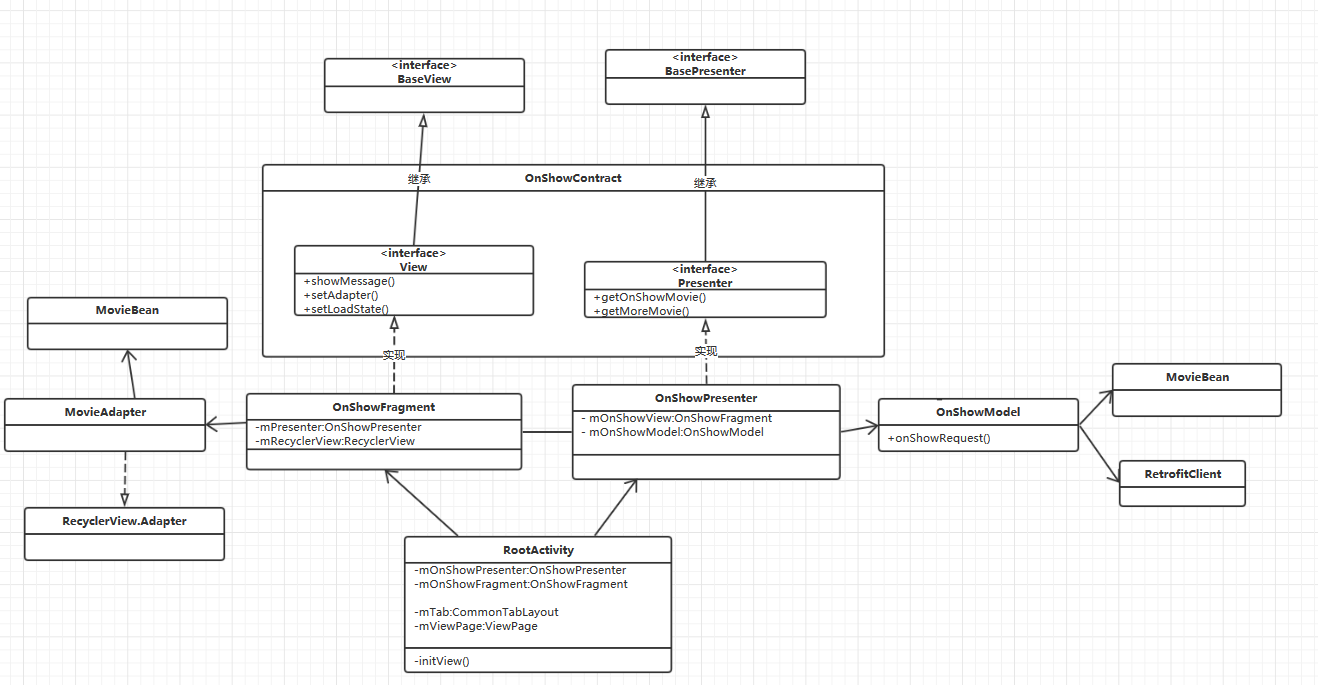
本系统的注册功能，是在RegisterActivity中进行MVP三部分的初始化与关系组织，由用户对View上的下拉框控件进行操作，选择用户的性别，年龄，学历等，在输入用户名密码后，再调用Presenter函数中的register函数进行注册，register中调用了Model的onRegisterRequest函数访问后端服务器，完成用户信息表的插入。

###### 2.2.3 找片Tab的类调用关系图



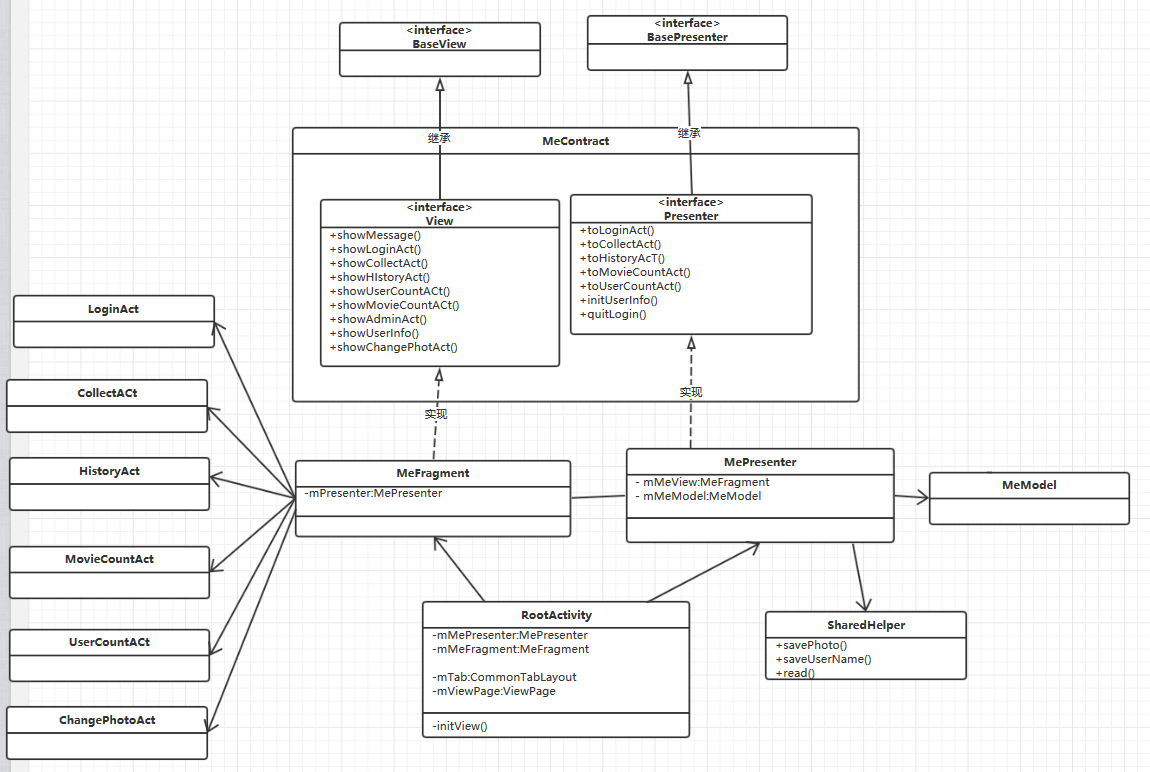
这是打开APP首先展示的View, RootActivity的第一个Tab项，主要是一些导航性质的功能，能够跳转到其他功能的Activity中。包括LoginActivity, MovieListActivity, SearchActivity, TypeSearchActivity。

###### 2.2.4 热映Tab 的类调用关系图



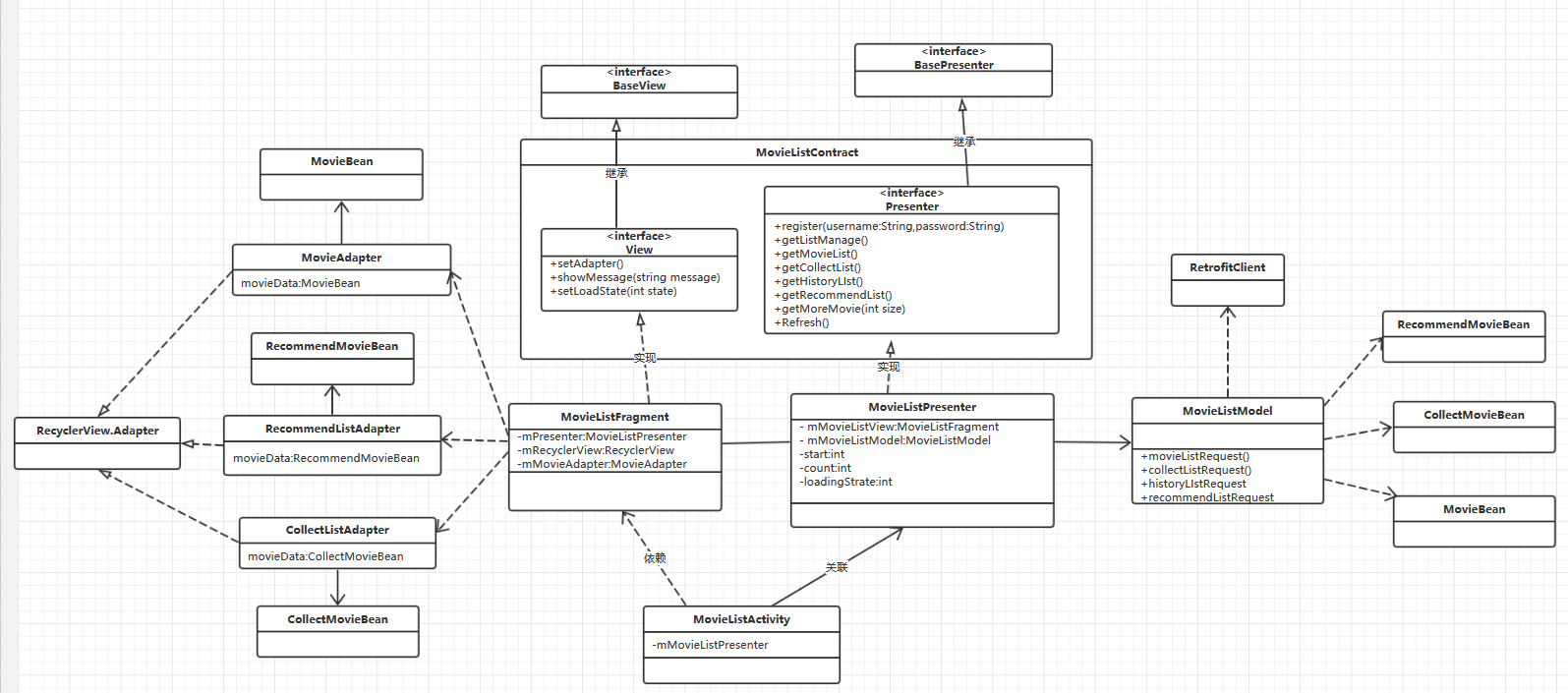
这是RootActivity的第二个Tab,主要是通过Presenter的getOnShowMovie函数，调用OnshowModel的onShowRequest函数访问豆瓣API,获取电影信息，返回后转化成MovieBean，由MovieAdapter类绑定到视图上。

###### 2.2.5 “我的信息”Tab类调用关系图



这是RootActivity的第三个Tab界面，主要也是一些导航性质的功能，跳转到登录界面，查看收藏界面，查看浏览记录界面，查看电影数据统计界面，查看用户数据统计界面，更换头像界面。还有一个退出登录的功能，通过sharedHelper类来对SharePrefence进行读写，更改本地保存的用户信息。

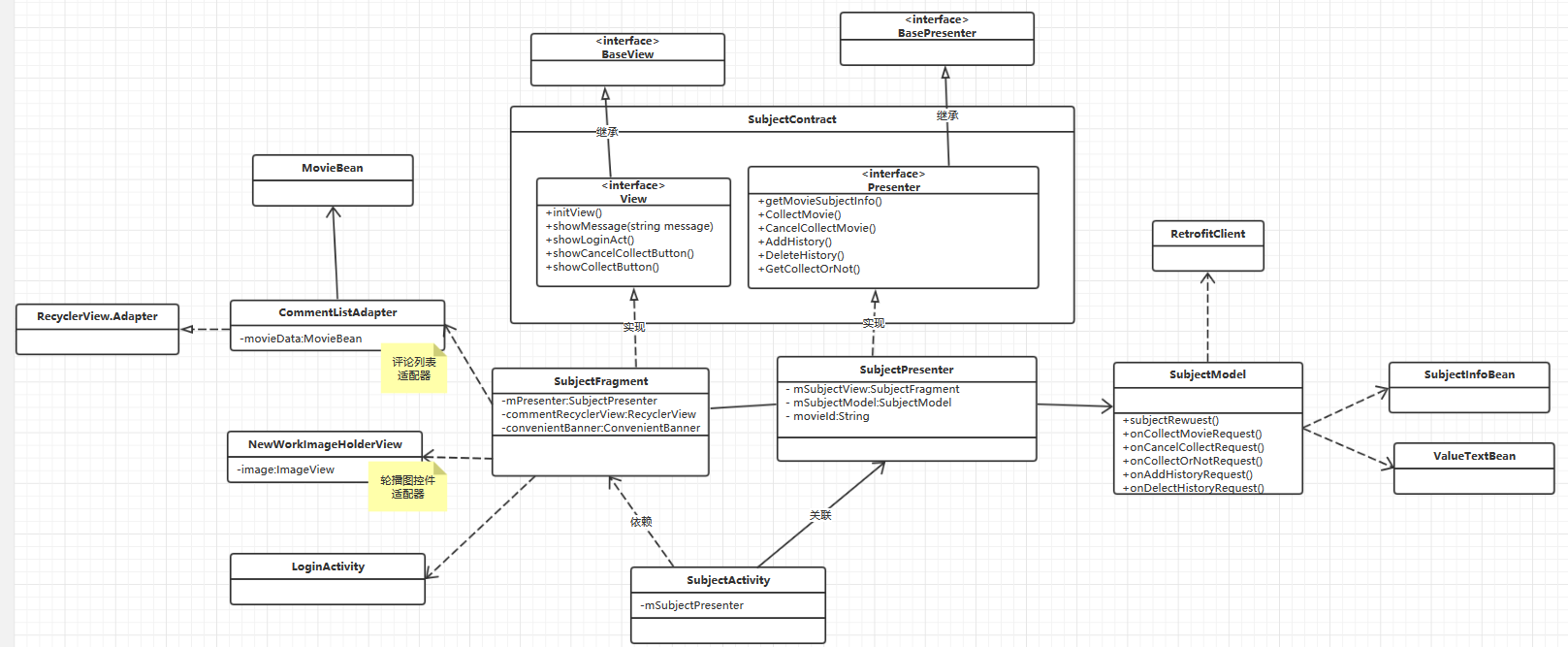
###### 2.2.6电影列表类调用关系图



电影列表类，主要功能是负责获取并展示各种列表信息，包括电影列表，收藏列表，推荐列表，历史记录列表。其中电影列表包括热映电影列表，关键词搜索结果的电影列表，新片榜电影列表等等。

主要流程是通过Presenter的getListManage()函数，判断是需要生产何种类型的列表，然后通过相关的getXXXList函数调用MovieListModel中的XXXListRequest来访问服务器获取列表信息。获得到的信息，通过Fragment中的Adapter类来对数据进行绑定，展示在RecycleView即列表控件上。

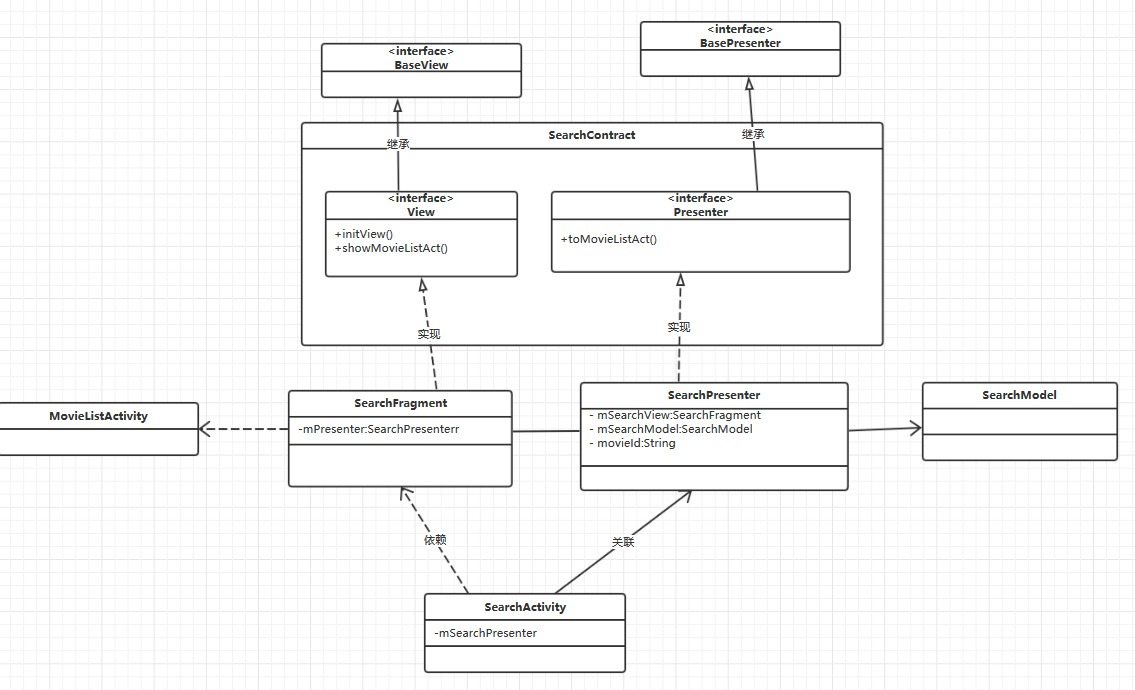
###### 2.2.7 电影详情类调用关系图



电影详情类，主要是在用户点击电影列表中某个项后进入电影详情界面，对该电影的详细信息进行展示，增加用户浏览记录，提供收藏电影，取消收藏，删除浏览记录的功能。如果用户未登录时进行收藏电影等操作，会跳转到登录界面。

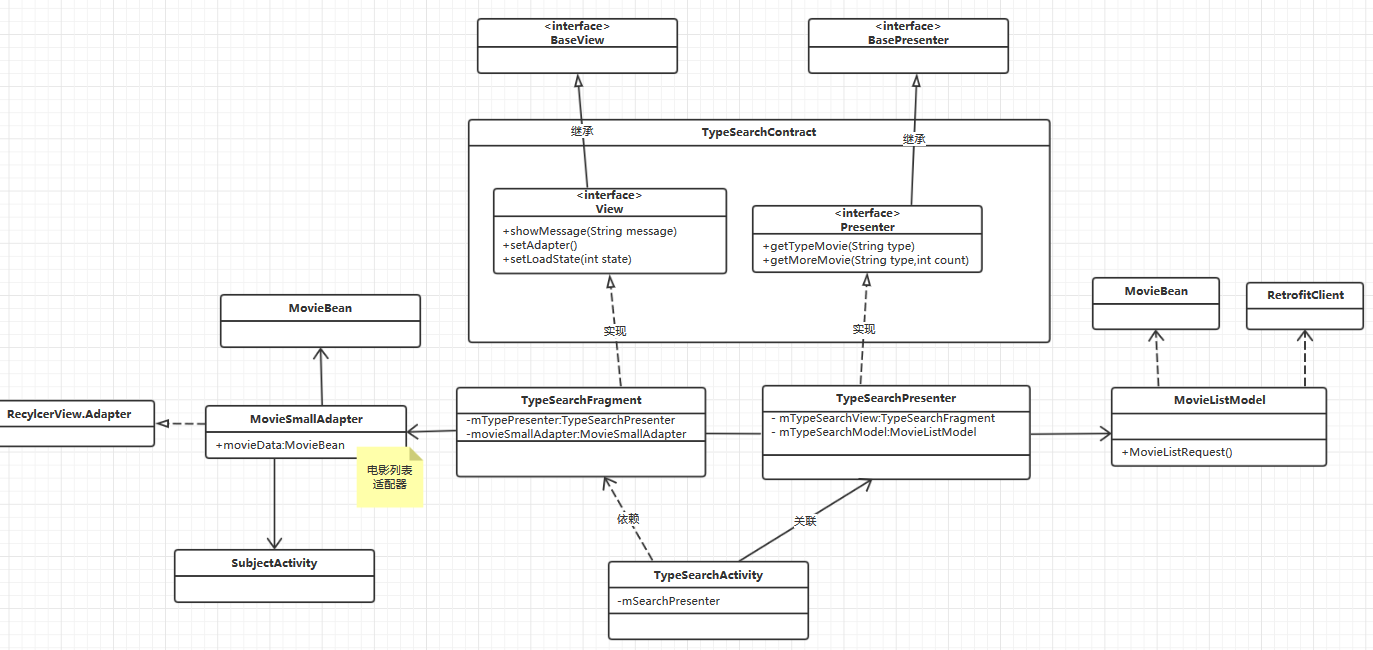
Fragment中initView函数用于初始化界面信息，showLoginAct用于跳转到登录界面，showCancelCollectButton与showCollectButton根据用户的收藏与否情况，对收藏按钮进行设置。Presenter中getMovieSubjecInfo()对应model的subjectRequest(),用于访问豆瓣API获取电影详情信息，collectMovie()用于收藏电影，cancelCollect()取消收藏，addHistory()增加浏览记录，deleteHistory()删除浏览记录。

###### 2.2.8 电影关键词搜索类调用关系图



这是一个比较简单的类，引用了一个外部开源库的SearchView控件，会对用户的搜索历史进行保存并且持久性存储在本地数据库中。在用户输入关键词后，会跳转到MovieListActivity获取关键词的搜索结果，并对搜索结果进行展示。

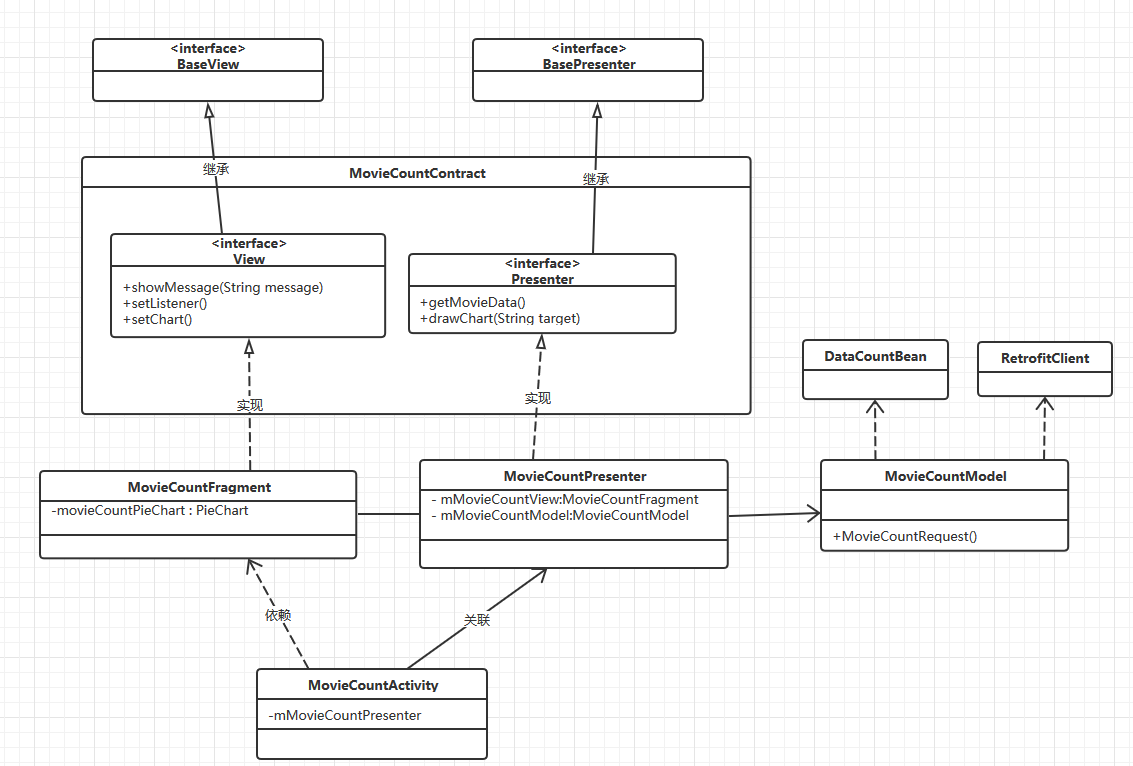
###### 2.2.9电影类型搜索类调用关系图



电影分类搜索类，主要功能是调用豆瓣API获得各类型电影数据，在使用RecyclerView进行展示。

其中使用presenter来调用MovieListModel的MovieListRequest()来访问豆瓣API,返回的数据存在MovieBean类中，交由MovieSmallAdapter进行数据绑定，并且点击各个项会跳转到对应的电影详情界面。

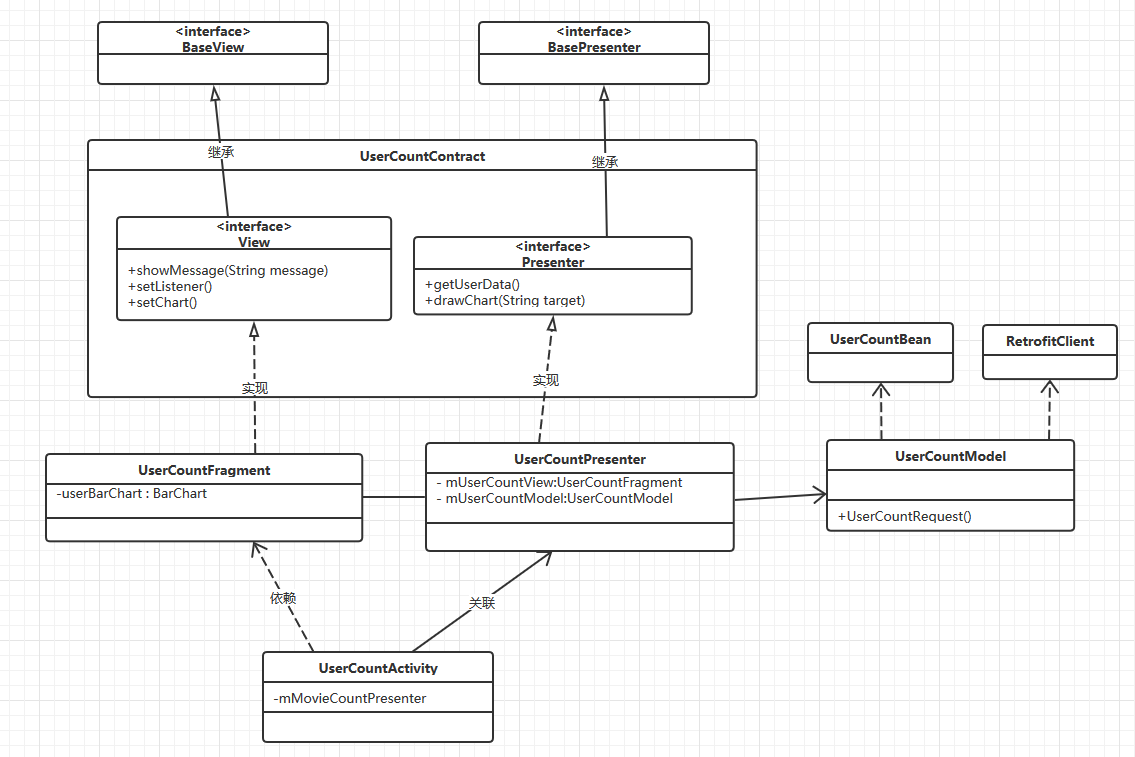
###### 2.2.10电影信息统计类调用关系图



电影信息统计类，能够根据用户所选的指标，访问服务器获得对应电影统计数据，以图形化的形式展示给用户。

主要是通过Presenter的getMovieData函数，调用Model的MovieCountRequest方法，访问服务器获得数据存再DataCountBean中，通过Fragment的setChart函数完成图表的绘制。

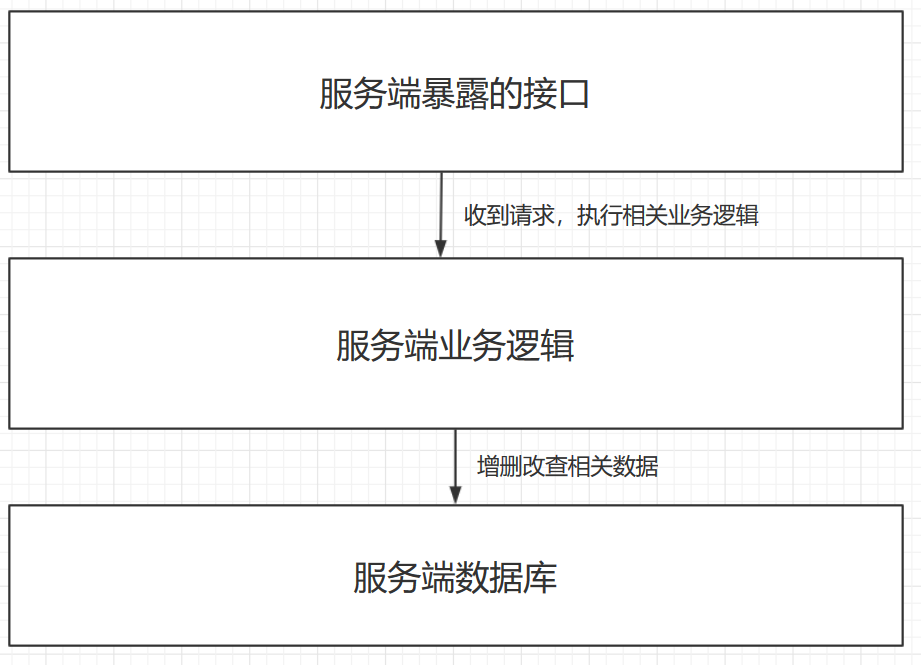
###### 2.2.11 用户信息统计类调用关系图



用户信息统计类，能够根据用户所选的指标，访问服务器获得对应用户信息统计数据，以图形化的形式展示给用户。

主要是通过Presenter的getUserData函数，调用Model的UserCountRequest方法，访问服务器获得数据存再UserCountBean中，通过Fragment的setChart函数完成图表的绘制。

###### 2.2.12 服务端软件体系结构图



服务端主要是采用node.js 下的express框架，为客户端提供了功能接口，具体接口信息详见2.3.6接口设计。

服务端的业务逻辑主要是对数据库进行相应的增删改查，向客户端返回对应数据。较为复杂的逻辑主要是用户喜好特征表的计算，根据基于内容的推荐算法，与基于用户画像的协同过滤推荐算法来给用户推荐他可能喜欢的电影。具体算法逻辑见详细设计规约。

#### 2.3接口设计

###### 2.3.1 Activity类之间的接口

各个Activity类之间的调用，需要使用Activity自带的startActivity 函数进行其他Activity的生成于跳转，使用Intent进行Activity之间的参数传递。例如生成Register活动样例如下：

Intent intent=new Intent(getContext(),RegisterActivity.class);

starActivity(intent);

###### 2.3.2 Presenter, View, Model之间绑定的接口

先在Activity中实例化一个View，一个Model,在实例化Presenter的构造函数中传入View与Model,初始化Presenter中的View与Model参数，同时调用View.SetPresenter（）函数，初始化对View中Presenter参数，至此完成了MVP模式的绑定。

###### 2.3.3 为RecyclerView(列表控件)绑定Adapter(数据适配器)的接口

需要实现一个RecyclerView.Adapter类, 在其中完成以下三个函数的实现

Void OnBindViewHolder (RecyclerView.Holder holder, int position),

Int getItemCount()

RecyclerView.Holder onCreateViewHolder (ViewGroup parent, int state)

接着实例化一个RecyclerView.Adapter类，调用RecyclerView的setAdapter（）函数进行适配器的绑定。

###### 2.3.4 网络访问组件Retrofit的接口

在系统中设计了一个RetrofitClient类，有getUserService() 和getMovieService()两个函数，分别返回访问服务器的网络Retrofit组件与访问豆瓣API的Retrofit组件，一般是在model中被调用。

###### 2.3.5 本地持续化存储访问接口

系统是使用SharedPreference进行本地持久性存储的，主要存储一些用户偏好设置信息。在系统中设计了一个ShareHelper类，有savePhoto(string photo),saveUsername(String username)两个接口来进行用户名和头像的本地存储。还有个read()接口进行以上信息的读取。

###### 2.3.6 APP客户端与服务端的接口

客户端使用Retrofit网络组件异步通信的方式访问客户端。

服务端提供了一系列用户功能的接口，是通过json形式返回的相关的数据。

服务端总端口 http:192.168.3.6:3500

2.3.6.1 登录接口

路径：/login

方式：Get

参数：用户名 username : string ，密码 password : string

返回值：{value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

2.3.6.2 注册接口

路径：/register

方式：Get

参数：用户名 username: string , 密码 password: string, 性别 sex: string, 年龄段 year :string , 学历 study: string

返回值：{value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

2.3.6.3 收藏接口

路径：/collect

方式：Get

参数：用户名 username: string, 电影ID movieId: string, 电影名movieTitle: string，电影类型 movieType: string，电影导演 movieDirector: string，电影年代 movieYear : string，电影语言 movieLanguage: string，用户收藏时间time:string.

返回值：{value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

2.3.6.4 取消收藏接口

路径：/cancel\_collect

方式：Get

参数：用户名 username: string, 电影ID movieId: string

返回值：{value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

2.3.6.5 电影收藏与否接口

路径：/collect\_or\_not

方式：Get

参数：用户名 username: string, 电影ID movieId: string

返回值：{value：string, Text : string}, 其中value值true表示该电影已经收藏,false为未收藏，Text表示收藏情况信息。

2.3.6.6 用户收藏列表

路径：/collect\_display

方式：Get

参数：用户名 username: string

返回值：{subject: Array[{用户名username: string, 电影ID movieID: string , 电影名 movieTitle: string , 收藏时间 time:string}]}

2.3.6.7 增加浏览记录接口

路径：/add\_history

方式：Get

参数：用户名 username: string, 电影ID movieId: string, 电影名movieTitle: string，用户浏览时间time:string.

返回值：{value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

2.3.6.8 删除浏览记录接口

路径：/delete\_history

方式：Get

参数：用户名 username: string, 电影ID movieId: string

返回值：{value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

2.3.6.9 用户浏览记录列表接口

路径：/history\_display

方式：Get

参数：用户名 username: string

返回值：{subject: Array[{用户名username: string, 电影ID movieID: string , 电影名 movieTitle: string , 浏览时间 time:string}]}

2.3.6.10 用户个人喜好推荐接口

路径：/guess\_you\_like

方式：Get

参数：用户名 username: string

返回值：{subject: Array[{电影ID movieId: string, 电影名movieTitle: string，电影类型 movieType: string，电影导演 movieDirector: string，电影年代 movieYear : string，电影语言 movieLanguage: string, 推荐理由reason:string }]}

2.3.6.11 电影信息统计接口

路径：/movie\_count

方式：Get

参数：无

返回值：{TypeLike: Array[{类型名 name:string，该类型电影收藏百分比 count:float}],

LanguageLike:Array[{语种名 name:string，该语种电影收藏百分比 count:float}]

YearLike:Array[{年代名 name:string，该年代电影收藏百分比 count:float}]}

2.3.6.12 用户信息统计接口

路径：/user\_count

方式：Get

参数：无

返回值：{manYear: Array[{年龄段名 name:string，该年龄段男性用户人数 count:float}],

womanYear: Array[{年龄段名 name:string，该年龄段女性用户人数 count:float}],

manStudy: Array[{学历名 name:string，该学历段男性用户人数 count:float}],

womanYear: Array[{学历名 name:string，该学历段女性用户人数 count:float}],}

2.3.6.13 用户头像设置接口

路径：/update\_photo

方式：Get

参数：用户名 username: string, 头像图片photo: string

返回值：{value：string, Text : string}, 其中value表示成功或失败，Text表示成功信息或失败信息。

###### 2.3.7 APP客户端与豆瓣API的接口

具体API说明文档如下：

<https://github.com/jokermonn/-Api/blob/master/DoubanMovie.md#introduce>

2.3.7.1 当前热映电影接口

url：<https://api.douban.com/v2/movie/in_theaters>

拼接参数：

 apikey：固定值0b2bdeda43b5688921839c8ecb20399b

 city：所在城市，例如北京、上海等

 start：分页使用，表示第几页

 count：分页使用，表示数量

 client：客户端信息。可为空

 udid：用户 id。可为空

返回值：movieBean格式

* count：返回数量
* start：分页量
* total：数据库总数量
* title：返回数据相关信息
* subjects：具体电影信息
  + rating：排名信息
    - max：最高分
    - average：该电影得分
    - details：?
    - stars：星数
    - min：最低分
  + genres：电影分类
  + title：电影名
  + casts：原型
    - avatars：该演员剧照
    - name\_en：该演员英文名
    - name：该演员中文名
    - alt：网页链接
    - id：演员 id
  + duration：电影时长
  + collect\_count：多少人看过
  + mainland\_pubdate：大陆上映时间
  + has\_video：是否有资源
  + original\_title：电影原名
  + subtype：固定值 movie
  + directors：导演信息
    - avatars：该导演剧照
    - name\_en：该导演英文名
    - name：该导演中文名
    - alt：网页链接
    - id：导演 id
  + pubdates：各地上映日期
  + year：上映年
  + images：剧照
  + alt：网页链接
  + id：电影 id，用于[电影介绍](https://github.com/jokermonn/-Api/blob/master/DoubanMovie.md" \l "introduce)

2.3.7.2 电影详情介绍接口

url：<http://api.douban.com/v2/movie/subject/> + 电影 id（来源[上映](https://github.com/jokermonn/-Api/blob/master/DoubanMovie.md" \l "playing)中的 subjects 中的 id 字段）

拼接参数：

* apikey：略，同[上映](https://github.com/jokermonn/-Api/blob/master/DoubanMovie.md" \l "playing)
* city：略，同[上映](https://github.com/jokermonn/-Api/blob/master/DoubanMovie.md" \l "playing)
* client：略，同[上映](https://github.com/jokermonn/-Api/blob/master/DoubanMovie.md" \l "playing)
* udid：略，同[上映](https://github.com/jokermonn/-Api/blob/master/DoubanMovie.md" \l "playing)

返回值: SubjectBean格式

* videos：片源
* wish\_count：多少人想看
* id：电影 id
* mobile\_url：移动端 url
* languages：电影语言
* writers：编剧
* trailers：原型
  + title：预告片名
  + resource\_url：预告片源
  + id：预告片 id
* has\_ticket：是否有票
* photos：剧照列表
* popular\_comments：热门短评
  + rating：评分
    - max：最高分
    - min：最低分
    - value：该评论给分
  + useful\_count：有用点赞量
  + author：评论者信息
    - uid：用户 id
    - avatar：用户头像
    - signature：用户个签
    - alt：用户主页链接
    - id：用户 id
    - name：用户名
  + subject\_id：电影 id
  + content：评论内容
  + create\_at：评论发表时间
  + id：评论 id
* comments\_count：短评总量
* summary：电影介绍
* popular\_reviews：热门影评
* reviews\_count：影评总量

2.3.7.3 电影搜索接口

url：<http://api.douban.com/v2/movie/search>

拼接参数：

tag : 搜索的电影标签

q: 搜索关键字

apikey：固定值0b2bdeda43b5688921839c8ecb20399b

start：分页使用，表示第几页

count：分页使用，表示数量

返回值：movieBean格式，与2.3.7.1返回值格式相同

2.3.7.4 豆瓣电影top250接口

url：<http://api.douban.com/v2/movie/top250>

拼接参数：

apikey：固定值0b2bdeda43b5688921839c8ecb20399b

start：分页使用，表示第几页

count：分页使用，表示数量

返回值：movieBean格式，与2.3.7.1返回值格式相同

2.3.7.5 新片榜接口

url：http://api.douban.com/v2/movie/new\_movies

拼接参数：

apikey：固定值0b2bdeda43b5688921839c8ecb20399b

start：分页使用，表示第几页

count：分页使用，表示数量

返回值：movieBean格式，与2.3.7.1返回值格式相同

#### 2.4界面设计

###### 2.4.1界面设计思路

采用传统的APP的设计方式，主界面下方布置三个TAB进行UI界面的导航。

第一个TAB为“找片”，主要提供一些电影列表信息的入口，方便用户找到自己想看的电影。

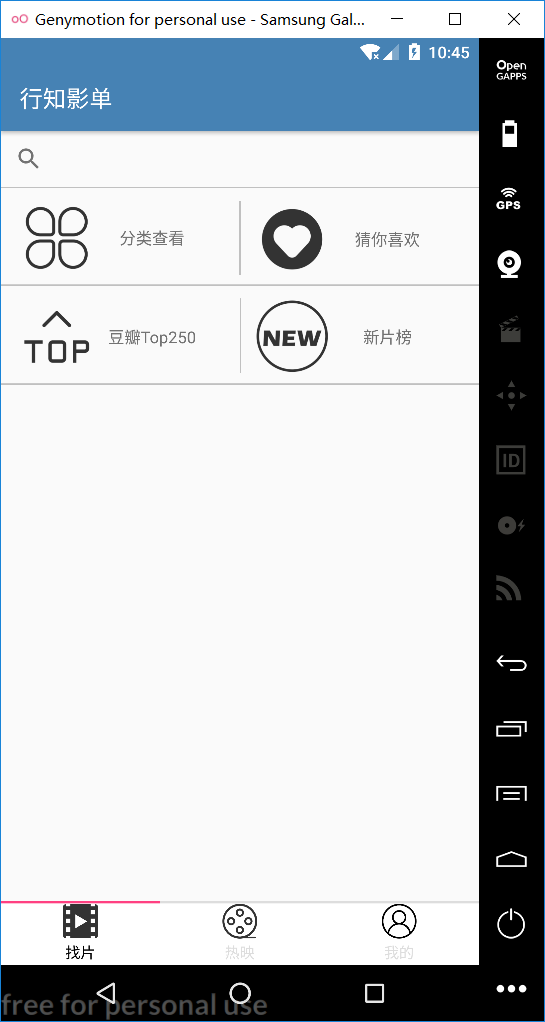
第二个TAB为“热映”，展示当前正在院线热映的电影列表，由于帮助用户了解当前院线电影，安排自己的观影计划本APP核心需求之一，因此把这个热映电影列表直接占据一个TAB。

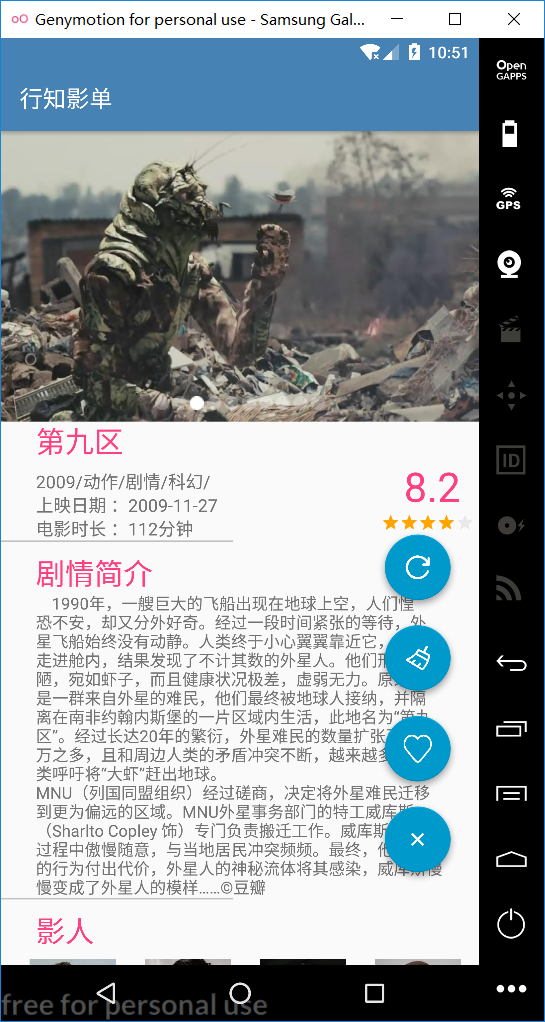
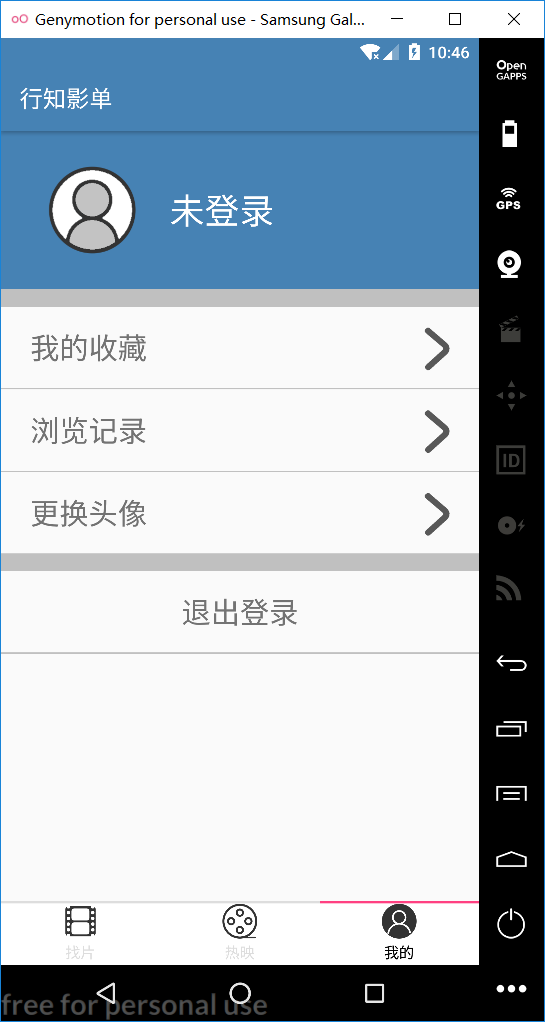
第三个TAB为“我的”，主要是一些需要登录注册后才能使用的用户功能，比如收藏列表，浏览记录，更换头像等等。

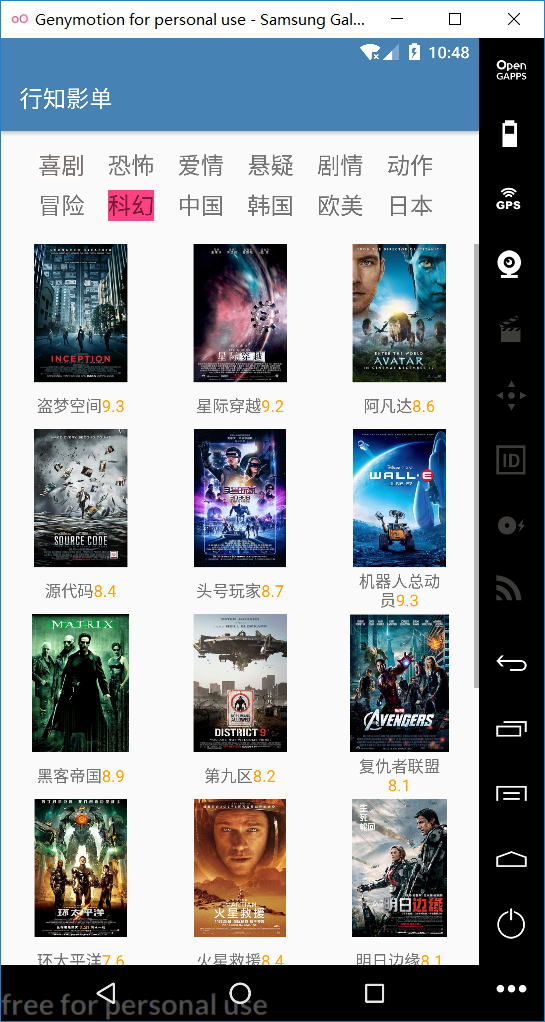
颜色主要以蓝，白，粉，灰四个颜色为主，保持风格的一致。

为了减少用户的学习成本，各个界面间的结构应该尽量统一，从起始页面到达各个界面的点击次数应该小于5次。

###### 2.4.2 界面设计展示







#### 2.5数据库设计

###### 2.5.1服务端数据库

本系统的服务端数据是使用MongoDB数据库进行数据存储。MongoDB 是一种基于分布式文件存储的数据库，它是一种非关系型的数据库，因此它支持的数据结构非常松散，是类似[json](https://baike.baidu.com/item/json)的[bson](https://baike.baidu.com/item/bson" \t "_blank)格式，可以存储比较复杂的数据类型。

它总共有users, collect, history, moviedata, typicaluser五张表，用来存储五种不同的类型与用途的数据。

2.5.1.1用户信息表users

用户信息表主要用于存储用户的信息，它有\_id, username, password, year, study, sex, photo, userLike等数据项。

\_ id ：ObjectID类型，表示该记录的id，相当于表的主码。

username : string 类型，表示用户名，长度应大于0，小于15。

password : string 类型，表示账户的密码，长度应大于0，小于15。

year : string类型，表示用户的年龄段，有0-15，16-20，21-25，26-30，31-35，36-40，41-50，50+等选项。

study : string 类型，表示用户的学历情况，有高中及以下，大专，本科，研究生及以上等选项。

sex : string类型，表示用户的性别，有男，女两个选项。

photo :string类型，表示用户的头像信息。

userLike : object类型，object类型下有三个数组

{

typeLike :[{movieType: string , movieTypeCount : float }],

languageLike: [{movieLanguage: string , movieLanguageCount : float }],

yearLike:[{movieYear: {start:int , end :int} , movieYearCount : float }]

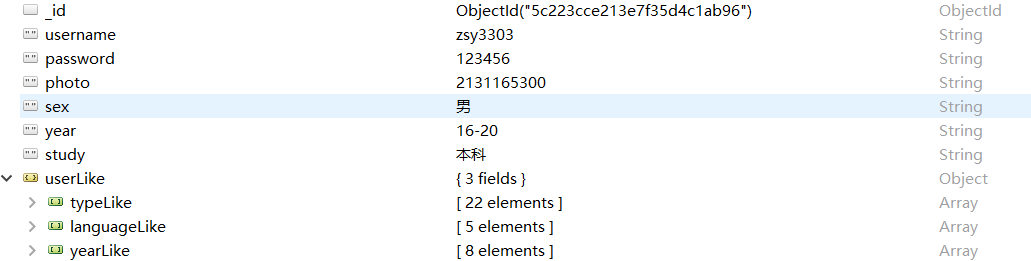
}

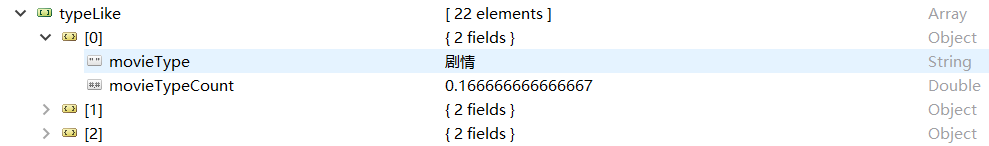
typeLike为类型喜好表，name项对应类型名（包括剧情，爱情，奇幻，悬疑，恐怖，惊悚，动作，喜剧，科幻，古装，武侠，家庭，传记，运动，动画，音乐，犯罪，冒险，战争，歌舞，儿童，同性等共二十一种），movieTypeCount项对应该用户对该类型电影喜好程度，为一个介于0-1的小数。

LanguageLike为语种喜好表，name项对应电影的语种（包括汉语普通话，英语，日语，韩语，粤语共五种），movieLanguageCount项对应该用户对该语种电影喜好程度，为一个介于0-1的小数。

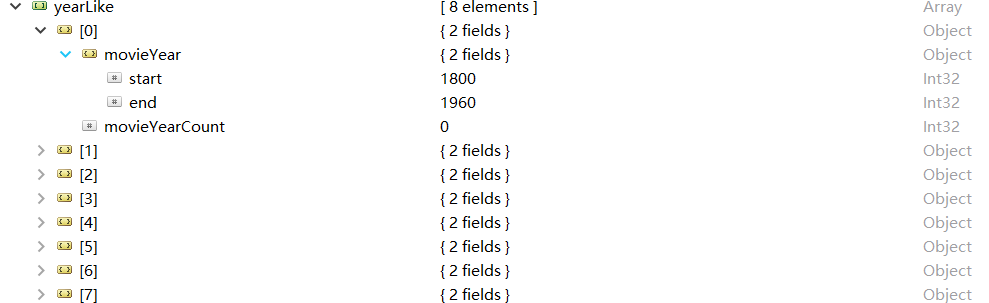
YearLike为年代喜好表，movieYear是一个object类型数据， 对应着电影的年代，有start, end两个int类型参数（包括{start:1800,end:1960} , {start:1961,end:1970} , {start:1971,end:1980} , {start:1981,end:1990} , {start:1991,end:2000} , {start:2001,end:2010} , {start:2011,end:2015} , {start:2016,end:2017} , {start:2018,end:2019}共八个项 ）。movieYearCount项对应该用户对该年代电影的喜好程度。

具体结构如图所示









2.5.1.2用户收藏表collect

用户收藏表主要用户存储用户的收藏信息。它的具体数据项包括：

\_id ：ObjectID类型，表示该记录的id，相当于表的主码。

username : string 类型，表示用户名，长度应大于0，小于15。

movieID : string 类型，表示电影的ID。

movieType : string类型，表示电影的类型，一部电影可能包含多种电影类型，用“/”隔开。

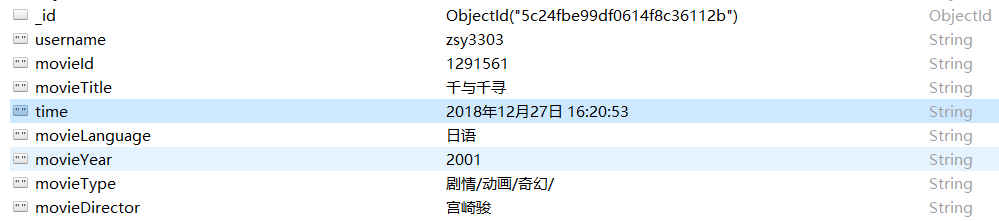
movieTitle : string 类型，表示电影的名字。

movieLanguage : string类型，表示电影的语种。

movieYear :int类型，表示电影的拍摄年份。

movieDirector : string 类型，表示电影导演的名字。

time : string 类型，表示用户收藏该电影的时间 格式为xxxx年xx月xx日 xx:xx:xx。



2.5.1.3用户浏览记录表history

用户浏览记录表主要用户存储用户的浏览记录信息。它的具体数据项包括：

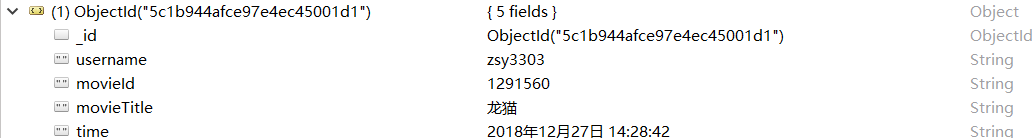
\_id ：ObjectID类型，表示该记录的id，相当于表的主码。

username : string 类型，表示用户名，长度应大于0，小于15。

movieID : string 类型，表示电影的ID。

movieTitle : string 类型，表示电影的名字。

time : string 类型，表示用户收藏该电影的时间 格式为xxxx年xx月xx日 xx:xx:xx。



2.5.1.4 电影数据表moviedata

电影数据表，主要存储待推荐给用户的电影的信息，它的具体数据项包括：

\_id ：ObjectID类型，表示该记录的id，相当于表的主码。

movieID : string 类型，表示电影的ID。

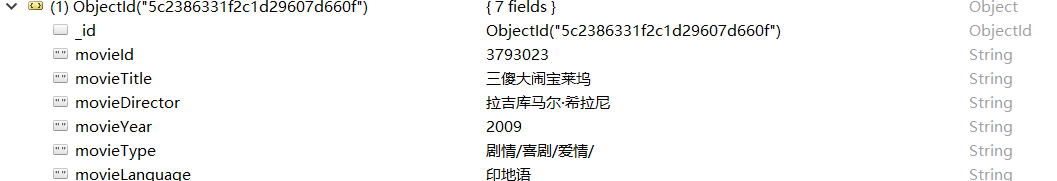
movieType : string类型，表示电影的类型，一部电影可能包含多种电影类型，用“/”隔开。

movieTitle : string 类型，表示电影的名字。

movieLanguage : string类型，表示电影的语种。

movieYear :int类型，表示电影的拍摄年份。

movieDirector : string 类型，表示电影导演的名字。



2.5.1.5用户画像表 typicalUser

用户画像表是主要用于刻画某类（年龄，性别，学历）用户的具体画像（主要是用户的电影喜好情况）。

它的数据结构和user表基本相同，它主要是用于给用户推荐电影时候，能够根据所有该类用户的电影喜好数据，模拟出一个典型的该类用户代表，再根据这个模拟出的用户代表给特定的用户推荐电影。这个表，就是用于存储某类用户的画像，而不用每次都去重新统计计算。它的结构为

\_ id ：ObjectID类型，表示该记录的id，相当于表的主码。

year : string类型，表示用户的年龄段，有0-15，16-20，21-25，26-30，31-35，36-40，41-50，50+等选项。

study : string 类型，表示用户的学历情况，有高中及以下，大专，本科，研究生及以上等选项。

sex : string类型，表示用户的性别，有男，女两个选项。

userLike : object类型，object类型下有三个数组

{

typeLike :[{movieType: string , movieTypeCount : float }],

languageLike: [{movieLanguage: string , movieLanguageCount : float }],

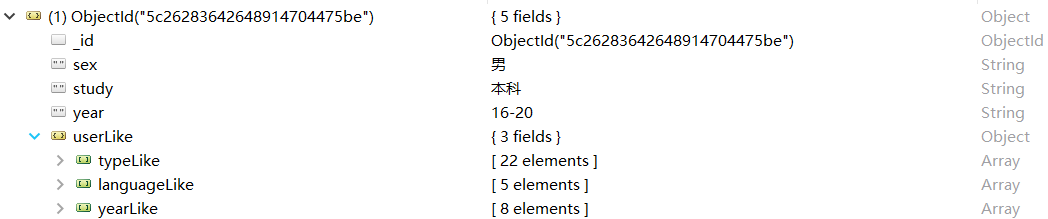
yearLike:[{movieYear: {start:int , end :int} , movieYearCount : float }]

}

typeLike为类型喜好表，name项对应类型名（包括剧情，爱情，奇幻，悬疑，恐怖，惊悚，动作，喜剧，科幻，古装，武侠，家庭，传记，运动，动画，音乐，犯罪，冒险，战争，歌舞，儿童，同性等共二十一种），movieTypeCount项对应该用户对该类型电影喜好程度，为一个介于0-1的小数。

LanguageLike为语种喜好表，name项对应电影的语种（包括汉语普通话，英语，日语，韩语，粤语共五种），movieLanguageCount项对应该用户对该语种电影喜好程度，为一个介于0-1的小数。

YearLike为年代喜好表，movieYear是一个object类型数据， 对应着电影的年代，有start, end两个int类型参数（包括{start:1800,end:1960} , {start:1961,end:1970} , {start:1971,end:1980} , {start:1981,end:1990} , {start:1991,end:2000} , {start:2001,end:2010} , {start:2011,end:2015} , {start:2016,end:2017} , {start:2018,end:2019}共八个项 ）。movieYearCount项对应该用户对该年代电影的喜好程度。



###### 2.5.2 客户端本地持久性存储

客户端使用SharedPreference进行本地的持久性存储，SharedPreferences是Android平台上一个轻量级的存储类，用来保存应用的一些常用配置。在我们的系统中，主要是用它来存储用户名和用户头像，这样当用户关闭APP后重新打开，就可以免去登陆的步骤。

username :string 用户名

userPhoto :string 用户头像

#### 2.6系统出错处理设计

###### 2.6.1出错信息

2.6.1.1客户端网络连接出错

当前界面会出现一个消息提示框（Toast）提示网络连接出错的原因，例如“time out”,“failed to connect to 192.168.3.6:3500”等

2.6.1.2登录注册时用户账号密码格式出错

当用户在注册或者登录时，输入了错误格式的账号密码信息时，系统会有消息提示框来提示用户。

2.6.1.3 用户未登录时进行了需要登录的操作

当用户在为登录时进行了需要登录的操作，系统会提示用户，还未登录。并且跳转到登录界面，引导用户登录。

2.6.1.4 用户未登录情况下点击退出登录按钮

当用户在未登录情况下点击退出登录，系统会提示当前用户还未登录。

2.6.1.5 用户使用管理员功能

理论上只有检测到该账号为管理员账号时，系统才会暴露出关系操作的接口，才能进行管理官操作。

###### 2.6.2 补救措施

2.6.2.1 客户端访问网络失败

当用户在电影详情界面获取数据失败时，可以点击浮动按钮中的刷新按钮进行界面的刷新。

当用户在某个Activity中获取数据超时时候，建议退出该Activity后重新加载。

2.6.2.2 登录注册时用户账号密码格式出错

此时直接在原先界面接着输入正确的账号密码即可。

2.6.1.3 用户未登录时进行了需要登录的操作

此时系统会直接跳转到登录界面引导用户登录。

###### 2.6.3系统维护设计

暂无。