

**Android课程设计报告**

**——仿今日头条**

**设计题目: 仿今日头条的新闻APP**

**班 级: 计算机161 计算机162**

**组 员: 靳珂 王钰婷**

**指导老师: 邓飞**

目录

[1可行性分析 3](#_Toc8738587)

[1.1经济可行性 3](#_Toc8738588)

[1.2技术可行性 3](#_Toc8738589)

[2需求分析 3](#_Toc8738590)

[2.1功能需求 4](#_Toc8738591)

[2.2性能需求 4](#_Toc8738592)

[3开发环境搭建 4](#_Toc8738593)

[3.1 Java环境配置 4](#_Toc8738594)

[3.2 Android Studio的安装 5](#_Toc8738595)

[4系统设计 6](#_Toc8738596)

[4.1功能设计 6](#_Toc8738597)

[4.2用户界面设计 7](#_Toc8738598)

[4.2.1主界面 7](#_Toc8738599)

[4.2.2频道分类编辑 10](#_Toc8738600)

[4.2.3新闻内容页 12](#_Toc8738601)

[4.2.5视频新闻 14](#_Toc8738602)

[4.2.6微头条 15](#_Toc8738603)

[4.2.7我的用户界面 15](#_Toc8738604)

[4.2.8登录注册 16](#_Toc8738605)

[4.3数据库搭建 17](#_Toc8738606)

[4.4后台服务器搭建 17](#_Toc8738607)

[5具体功能实现的代码 18](#_Toc8738608)

[5.1后台服务器连接 18](#_Toc8738609)

[5.2下拉刷新以及首页与刷新图标的切换 18](#_Toc8738610)

[5.3获取新闻记录 19](#_Toc8738611)

[5.4和时间相关的工具类的封装 23](#_Toc8738612)

[5.5视频解析加密 28](#_Toc8738613)

[5.6Activity的基类 32](#_Toc8738614)

[5.7Fragment的基类 39](#_Toc8738615)

[5.8频道编辑操作的实现 43](#_Toc8738616)

[6总结 54](#_Toc8738617)

# 

# 1可行性分析

开发任何一个基于手机的应用程序，都会受到资源和时间上的限制。因此，在接收任何一个应用程序发任务之前，必须根据用户可能提供的资源和时间条件进行可行性分析，旨在减少项目开发风险，避免人力、财力和物力的浪费。风险分析与可行性分析在很多方面是相互关联的，项目的风险越大，开发质量高的软件的可行性的几率就越小。

## 1.1经济可行性

采用的手机数据流量或者无线WIFI作为通信手段，当今通信技术的迅速发展， 4G网络已经基本普及世界各地，网速高速发展，之前的较慢的3G网络已基本被完全取代，并且5G技术即将开始投入市场，数据流量的资费也越来越低，速度越来越快，而且现在很多手机都可连接无线wifi， 实现数据流量0资费，大大加强了本新闻阅读软件的经济可行性。

## 1.2技术可行性

开发一个新闻客户端软件，用到的技术不会十分复杂，使用Fragment碎片技术实现界面的UI操作设计，再利用Fidder来抓取今日头条的新闻视频资源，再通过正则匹配进行解析数据，通过联网的往事实现实时刷新，通过相关资料以及实例可以尝试实现。

# 2需求分析

随着智能手机世界范围内的迅速发展，智能手机的操作系统之间的竞争也是越来越激烈。不过安卓手机以其开源性，以及众多手机制造商的原因，占据了广大市场，价格上也更加社和大众，拥有了大量的智能手机用户后，人们在这上面就有了更多的生活、工作、娱乐、休闲的方式。手机看新闻也成为了人们足不出户便可观之天下大事的方式，不在局限于每天只有通过报纸或者电脑上网才能了解时事。所以新闻手机app的设计是非常有必要的，具有非常广阔的发展前景。

## 2.1功能需求

启动APP后便可进入主界面进行看新闻，上方有导航条可以选择新闻的分类，点击操作按钮可以进行分类频道的编辑，包括添加，删除，移动等。下拉新闻可进行实时刷新新闻，点击可进入查看新闻详情，下方菜单栏可以选择首页新闻、视频、发头条、我的等界面

1. 底部菜单栏实现碎片布局分类
2. 新闻页通过导航条实现分类
3. 对分类频道进行添加 删除 移动排序等编辑操作
4. 新闻页面以及新闻评论显示
5. 点击图片可查看大图并可保存图片至本地
6. 下拉可实时刷新新闻以及视频
7. 播放视频，全屏播放，自动屏幕旋转等
8. 实时抓取今日头条的新闻、视频数据

9.我的界面进行用户注册登录等功能

## 2.2性能需求

从用户的角度出发，安卓手机的普及基本实现，近几年安卓的手机无论是从硬件还是系统软件都在快速发展，用户对app的需求以及要求也快速提高，程序响应速度必须快、信息处理速度必须快、安全性必须高。并且功能必须多样化，不仅仅要只能看文字新闻，图片和文字必须相互搭配，随着4G网络的发展以及流量速度提高以及资费的降低，视频新闻也更加具备用户市场。另外为了受众面更广，根据目前的android设备系统版本的分布情况，android6.0以上的版本占到设备总量的90%以上，所以做开发的时候选择的android系统最低版本也必须是android6.0以上来保证用户量。

# 3开发环境搭建

本款手机新闻客户端软件是采用java语言编写，使用Android Studio集成开发环境，Android Studio是一款谷歌官方推荐的优秀集成环境。客户端通过抓取解析今日头条的相关新闻，生成客户端数据，并搭建了第三方服务器BMOB来实现用户的相关信息操作。

## 3.1 Java环境配置

采用jdk的版本是jdk1.7，目前jdk版本最高的是jdk1.8，但是为了程序的稳定性，还是采用已经稳定成熟的版本作为开发工具。

  从Oracle公司下载 jdk1.7之后安装，由于本机的开发系统是win7的64位操作系统，选择下载64为jdk,然后按照安装步骤一步一步next，jdk就安装完成了，然后配置环境java的环境变量，整个jdk的过程就完成了。

## 3.2 Android Studio的安装

Android Studio是Google于2013 I/O大会针对Android开发推出的新的开发工具，目前很多开源项目都已经在采用，Google的更新速度也很快，明显能感觉到这是Android开发的未来，其优点如下

**1、Google推出的**

毫无疑问，这个是它的最大优势，Android Stuido是Google推出，专门为Android“量身订做”的，是Google大力支持的一款基于IntelliJ IDEA改造的IDE

**2、速度更快**

Eclipse的启动速度、响应速度、内存占用一直被诟病，相信大家这点应该深有体会，而且经常遇到卡死状态。Studio不管哪一个方面都全面领先Eclipse

**3、UI更漂亮**

I/O上演示的那款黑色主题真是太棒了，极客范，Stuido自带的Darcula主题的炫酷黑界面实在是高大上，相比而言Eclipse下的黑色主题太low了

**4、更加智能**

提示补全对于开发来说意义重大， Studio则更加智能，智能保存，从此再也不用每次都 Ctrl + S了。熟悉Studio以后效率会大大提升。

**5、整合了Gradle构建工具**

Gradle是一个新的构建工具，自Studio亮相之处就支持Gradle，可以说Gradle集合了Ant和Maven的优点，不管是配置、编译、打包都非常棒。

**6、强大的UI编辑器**

Android Studio的编辑器非常的智能，除了吸收Eclipse+ADT的优点之外，还自带了多设备的实时预览，相对这对Android开发者来说简直是神器啊。

**7、内置终端**

Studio内置终端，这对于习惯命令行操作的人来说简直是福音啊，再也不用来回切换了，一个Studio全部搞定。

**8、更完善的插件系统**

Studio下支持各种插件，如Git、Markdown、Gradle等等，你想要什么插件，直接搜索下载。

**9、完美整合版本控制系统**

安装的时候就自带了如GitHub, Git, SVN等流行的版本控制系统，可以直接check out你的项目。

从谷歌官网下载最新版本，一直下一步完成安装。安装完成后运行Android Studio完成配置即可。



# 4系统设计

## 4.1功能设计

仿今日头条新闻网客户端功能结构图如下

## 4.2用户界面设计

### 4.2.1主界面

底部菜单栏使用BottomBarLayout(轻量级底部导航栏)第三方库实现首页、视频、发头条、我的 四个部分，首页顶部采用频道分类的方式实现，使用BaseRecyclerViewAdapterHelper

(ReclerView万能适配器)进行下拉刷新等操作，整体界面采用多个Fragment碎片实现相互隐藏或显示来实现多个界面。







### 4.2.2频道分类编辑

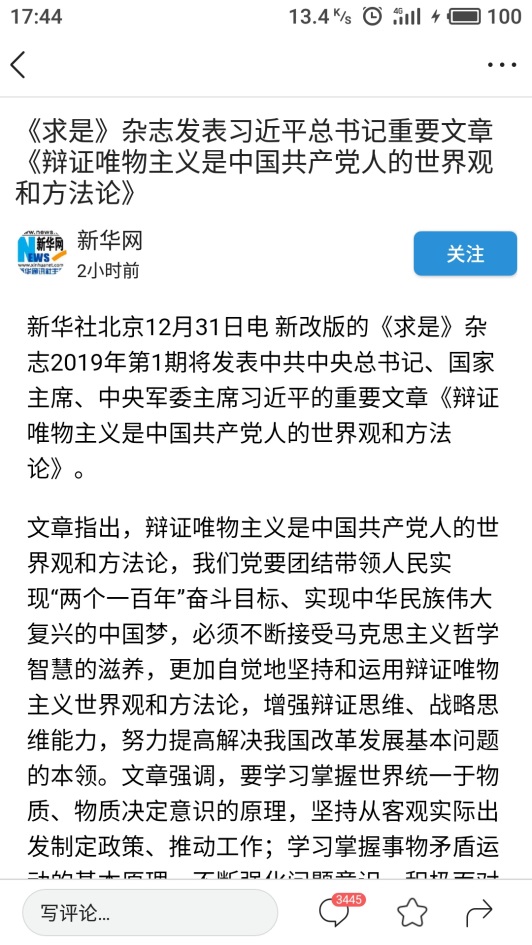
点击主页面频道分类编辑按钮可进入频道分类编辑界面，点击编辑，后进入编辑模式，可点击叉将频道从我的频道删除至频道推荐栏，长按一个频道并拖动可移动排序，点击频道推荐栏的频道可将其添加至我的频道

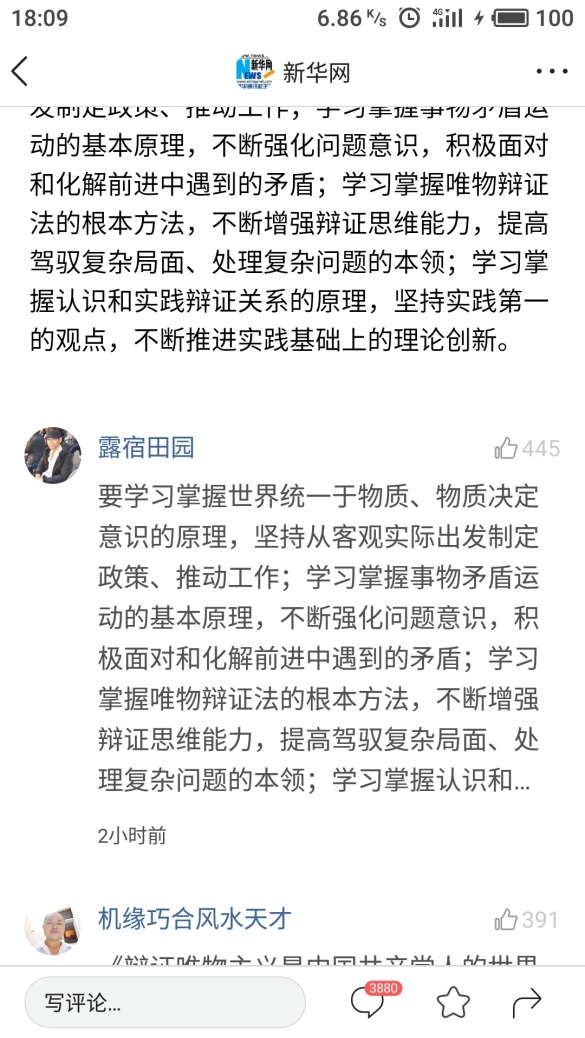




### 4.2.3新闻内容页

点击主页面对应的新闻打开新闻详细页，新闻详细页显示解析后的页面数据，包括新闻的发送人，内容详情页面，文字图片等，下方还有评论发言等





### 4.2.5视频新闻

在视频列表点击视频可直接播放，当视频滑过顶部时自动暂停，也可点击进入详情不放，可进入全屏播放，可调整音量，亮度，自动进行屏幕旋转等，下方可显示评论。







### 4.2.6微头条

微头条可发布、查看短幅新闻，目前暂未实现具体功能

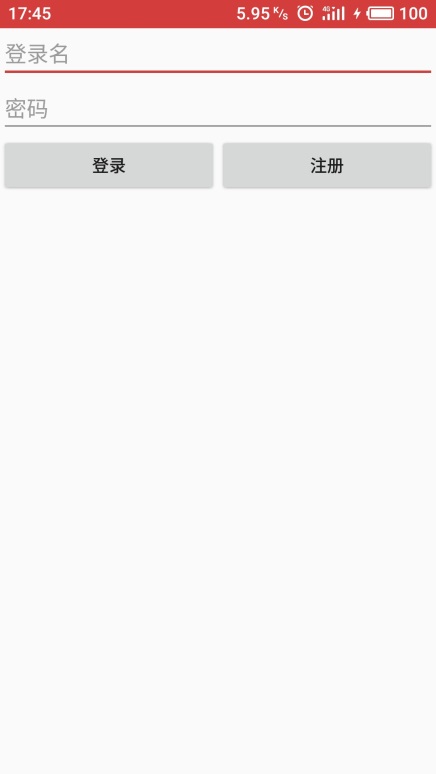


### 4.2.7我的用户界面

在该界面可现实用户信息，并进行用户的登录注册等相关功能



### 4.2.8登录注册

点击进入登录注册页面，通过填写相关信息进行注册，数据会上传至后台服务器

## 4.3数据库搭建

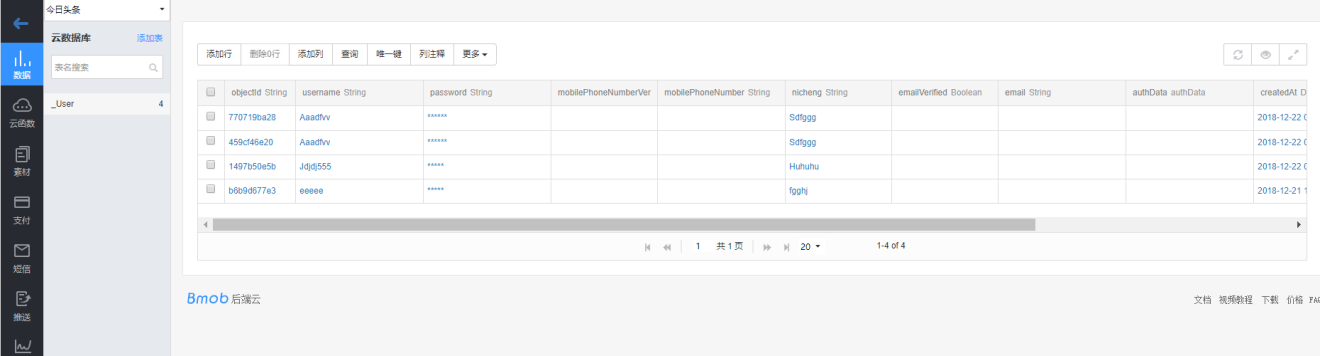
主要需要存储抓取的新闻或者视频的相关信息，采用Android内置的Sqlite数据库。

新闻表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性** |  | **数据类型** |  | **说明** |
| **id** |  | int |  | 主键 |
| **author** |  | varchar |  | 新闻作者 |
| **click** |  | int |  | 新闻点击次数 |
| **time** |  | datetime |  | 新闻发布时间 |
| **title** |  | varchar |  | 新闻标题 |
| **url** |  | varchar |  | 新闻url地址 |

## 4.4后台服务器搭建

由于本app采用抓取新闻方式，因此新闻视频等都通过解析后存储在本地数据库，不需要进行存储在网络服务器，网络服务器只需要存储用户的账号密码，头像昵称等相关信息即可，所以我们采用了建议的网络第三方云数据库BMOB数据库



# 5具体功能实现的代码

## 5.1后台服务器连接

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_denglu\_);

Bmob.initialize(this, "fbf3514decb00037fc3a1d8c10301453");

initialize();

initView();

## 5.2下拉刷新以及首页与刷新图标的切换

public void initListener() {  
 mTabAdapter = new MainTabAdapter(mFragments, getSupportFragmentManager());  
 mVpContent.setAdapter(mTabAdapter);  
 mVpContent.setOffscreenPageLimit(mFragments.size());  
 mBottomBarLayout.setViewPager(mVpContent);  
 //设置条目点击的监听  
 mBottomBarLayout.setOnItemSelectedListener(new BottomBarLayout.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(BottomBarItem bottomBarItem, int position) {  
 setStatusBarColor(position);//设置状态栏颜色  
  
 Jzvd.releaseAllVideos();//底部页签切换或者是下拉刷新，释放资源  
  
 if (position == 0 || position == 1) {  
 //如果点击的是首页  
 if (mBottomBarLayout.getCurrentItem() == position) {  
 //如果当前页码和点击的页码一致,进行下拉刷新  
 String channelCode = "";  
 if (position == 0) {  
 channelCode = ((HomeFragment) mFragments.get(0)).getCurrentChannelCode();//获取到首页当前显示的fragment的频道  
 } else {  
 channelCode = ((VideoFragment) mFragments.get(1)).getCurrentChannelCode();//获取到视频当前显示的fragment的频道  
 }  
 postTabRefreshEvent(bottomBarItem, position, channelCode);//发送下拉刷新的事件  
 }  
 return;  
 }  
  
 //如果点击了其他条目  
 BottomBarItem bottomItem = mBottomBarLayout.getBottomItem(0);  
 bottomItem.setIconSelectedResourceId(R.mipmap.tab\_home\_selected);//更换为原来的图标  
  
 cancelTabLoading(bottomItem);//停止旋转动画  
 }  
 });  
}

## 5.3获取新闻记录

public class NewsRecordHelper {

private static Gson mGson = new Gson();

/\*\*

\* 获取数据库保存的某个频道的最后一条记录

\*

\* @param channelCode 频道

\* @return

\*/

public static NewsRecord getLastNewsRecord(String channelCode) {

return DataSupport.where("channelCode=?", channelCode).findLast(NewsRecord.class);

}

/\*\*

\* 获取某个频道上一组新闻记录

\*

\* @param channelCode 频道

\* @param page 页码

\* @return

\*/

public static NewsRecord getPreNewsRecord(String channelCode, int page) {

List<NewsRecord> newsRecords = selectNewsRecords(channelCode, page - 1);

if (ListUtils.isEmpty(newsRecords)) {

return null;

}

return newsRecords.get(0);

}

/\*\*

\* 保存新闻记录

\*

\* @param channelCode

\* @param json

\*/

public static void save(String channelCode, String json) {

int page = 1;

NewsRecord lastNewsRecord = getLastNewsRecord(channelCode);

if (lastNewsRecord != null) {

//如果有记录

page = lastNewsRecord.getPage() + 1;//页码为最后一条的页码加1

}

//保存新的记录

NewsRecord newsRecord = new NewsRecord(channelCode, page, json, System.currentTimeMillis());

newsRecord.saveOrUpdate("channelCode = ? and page = ?", channelCode, String.valueOf(page));

}

/\*\*

\* 根据频道码和页码查询新闻记录

\* @param channelCode

\* @param page

\* @return

\*/

private static List<NewsRecord> selectNewsRecords(String channelCode, int page) {

return DataSupport

.where("channelCode = ? and page = ?", channelCode, String.valueOf(page))

.find(NewsRecord.class);

}

/\*\*

\* 将json转换成新闻集合

\*

\* @param json

\* @return

\*/

public static List<News> convertToNewsList(String json) {

return mGson.fromJson(json, new TypeToken<List<News>>() {

}.getType());

}

}

## 5.4和时间相关的工具类的封装

public class TimeUtils {

public static final long ONE\_MINUTE\_MILLIONS = 60 \* 1000;

public static final long ONE\_HOUR\_MILLIONS = 60 \* ONE\_MINUTE\_MILLIONS;

public static final long ONE\_DAY\_MILLIONS = 24 \* ONE\_HOUR\_MILLIONS;

/\*\*

\* 获取短时间格式

\*

\* @return

\*/

public static String getShortTime(long millis) {

Date date = new Date(millis);

Date curDate = new Date();

String str = "";

long durTime = curDate.getTime() - date.getTime();

int dayStatus = calculateDayStatus(date, new Date());

if (durTime <= 10 \* ONE\_MINUTE\_MILLIONS) {

str = "刚刚";

} else if (durTime < ONE\_HOUR\_MILLIONS) {

str = durTime / ONE\_MINUTE\_MILLIONS + "分钟前";

} else if (dayStatus == 0) {

str = durTime / ONE\_HOUR\_MILLIONS + "小时前";

} else if (dayStatus == -1) {

str = "昨天" + DateFormat.format("HH:mm", date);

} else if (isSameYear(date, curDate) && dayStatus < -1) {

str = DateFormat.format("MM-dd", date).toString();

} else {

str = DateFormat.format("yyyy-MM", date).toString();

}

return str;

}

/\*\*

\* 判断是否是同一年

\* @param targetTime

\* @param compareTime

\* @return

\*/

public static boolean isSameYear(Date targetTime, Date compareTime) {

Calendar tarCalendar = Calendar.getInstance();

tarCalendar.setTime(targetTime);

int tarYear = tarCalendar.get(Calendar.YEAR);

Calendar compareCalendar = Calendar.getInstance();

compareCalendar.setTime(compareTime);

int comYear = compareCalendar.get(Calendar.YEAR);

return tarYear == comYear;

}

/\*\*

\* 判断是否处于今天还是昨天，0表示今天，-1表示昨天，小于-1则是昨天以前

\* @param targetTime

\* @param compareTime

\* @return

\*/

public static int calculateDayStatus(Date targetTime, Date compareTime) {

Calendar tarCalendar = Calendar.getInstance();

tarCalendar.setTime(targetTime);

int tarDayOfYear = tarCalendar.get(Calendar.DAY\_OF\_YEAR);

Calendar compareCalendar = Calendar.getInstance();

compareCalendar.setTime(compareTime);

int comDayOfYear = compareCalendar.get(Calendar.DAY\_OF\_YEAR);

return tarDayOfYear - comDayOfYear;

}

/\*\*

\* 将秒数转换成00:00的字符串，如 118秒 -> 01:58

\* @param time

\* @return

\*/

public static String secToTime(int time) {

String timeStr = null;

int hour = 0;

int minute = 0;

int second = 0;

if (time <= 0)

return "00:00";

else {

minute = time / 60;

if (minute < 60) {

second = time % 60;

timeStr = unitFormat(minute) + ":" + unitFormat(second);

} else {

hour = minute / 60;

if (hour > 99)

return "99:59:59";

minute = minute % 60;

second = time - hour \* 3600 - minute \* 60;

timeStr = unitFormat(hour) + ":" + unitFormat(minute) + ":"

+ unitFormat(second);

}

}

return timeStr;

}

public static String unitFormat(int i) {

String retStr = null;

if (i >= 0 && i < 10)

retStr = "0" + Integer.toString(i);

else

retStr = "" + i;

return retStr;

}

}

## 5.5视频解析加密

public abstract class VideoPathDecoder {

public static final String TAG = VideoPathDecoder.class.getSimpleName();

public void decodePath(String srcUrl) {

KLog.e(TAG,"srcUrl: " + srcUrl);

ApiRetrofit.getInstance().getApiService().getVideoHtml(srcUrl)

.flatMap(new Func1<String, Observable<ResultResponse<VideoModel>>>() {

@Override

public Observable<ResultResponse<VideoModel>> call(String response) {

Pattern pattern = Pattern.compile("videoId: '(.+)'");

Matcher matcher = pattern.matcher(response);

if (matcher.find()) {

String videoId = matcher.group(1);

KLog.e(TAG,"videoId: " + videoId);

//1.将/video/urls/v/1/toutiao/mp4/{videoid}?r={Math.random()}，进行crc32加密。

String r = getRandom();

KLog.e(TAG,"r: " + r);

CRC32 crc32 = new CRC32();

String s = String.format(ApiConstant.URL\_VIDEO, videoId, r);

//进行crc32加密。

crc32.update(s.getBytes());

String crcString = crc32.getValue() + "";

//2.访问http://i.snssdk.com/video/urls/v/1/toutiao/mp4/{videoid}?r={Math.random()}&s={crc32值}

String url = ApiConstant.HOST\_VIDEO + s + "&s=" + crcString;

KLog.e(TAG,"s: " + crcString);

Log.i(TAG,url);

return ApiRetrofit.getInstance().getApiService().getVideoData(url);

}

return null;

}

})

.map(new Func1<ResultResponse<VideoModel>, Video>() {

@Override

public Video call(ResultResponse<VideoModel> videoModelResultResponse) {

VideoModel.VideoListBean data = videoModelResultResponse.data.video\_list;

if (data.video\_3 != null) {

return updateVideo(data.video\_3);

}

if (data.video\_2 != null) {

return updateVideo(data.video\_2);

}

if (data.video\_1 != null) {

return updateVideo(data.video\_1);

}

return null;

}

private String getRealPath(String base64) {

return new String(Base64.decode(base64.getBytes(), Base64.DEFAULT));

}

private Video updateVideo(Video video) {

//base64解码

video.main\_url = getRealPath(video.main\_url);

return video;

}

})

.subscribeOn(Schedulers.io())

.observeOn(AndroidSchedulers.mainThread())

.subscribe(new Subscriber<Video>() {

@Override

public void onCompleted() {

}

@Override

public void onError(Throwable e) {

e.printStackTrace();

onDecodeError(UIUtils.getString(R.string.video\_parse\_error));

}

@Override

public void onNext(Video video) {

KLog.e(TAG,"视频地址：" + video.main\_url);

onSuccess(video.main\_url);

}

});

}

private String getRandom() {

Random random = new Random();

StringBuilder result = new StringBuilder();

for (int i = 0; i < 16; i++) {

result.append(random.nextInt(10));

}

return result.toString();

}

public abstract void onSuccess(String url);

public abstract void onDecodeError(String errorMsg);

}

## 5.6Activity的基类

@Override

public final void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

if (enableSlideClose()) {

ParallaxBackLayout layout = ParallaxHelper.getParallaxBackLayout(this, true);

layout.setEdgeMode(EDGE\_MODE\_DEFAULT);//边缘滑动

layout.setEdgeFlag(getEdgeDirection());

layout.setLayoutType(getSlideLayoutType(),null);

layout.setSlideCallback(new ParallaxBackLayout.ParallaxSlideCallback() {

@Override

public void onStateChanged(int state) {

//收起软键盘

UIUtils.hideInput(getWindow().getDecorView());

}

@Override

public void onPositionChanged(float percent) {

}

});

}

this.savedInstanceState = savedInstanceState;

//初始化的时候将其添加到集合中

synchronized (mActivities) {

mActivities.add(this);

}

mPresenter = createPresenter();

//子类不再需要设置布局ID，也不再需要使用ButterKnife.bind()

setContentView(provideContentViewId());

ButterKnife.bind(this);

initView();

initData();

initListener();

}

public boolean enableSlideClose() {

return true;

}

/\*\*

\* 默认为左滑，子类可重写返回对应的方向

\* @return

\*/

public int getEdgeDirection(){

return EDGE\_LEFT;

}

/\*\*

\* 默认为覆盖滑动关闭效果，子类可重写

\* @return

\*/

public int getSlideLayoutType() {

return LAYOUT\_COVER;

}

public void initView() {

}

public void initData() {

}

/\*\*

\* 退出应用的方法

\*/

public static void exitApp() {

ListIterator<Activity> iterator = mActivities.listIterator();

while (iterator.hasNext()) {

Activity next = iterator.next();

next.finish();

}

}

public static Activity getCurrentActivity() {

return mCurrentActivity;

}

/\*\*

\* 统一退出控制

\*/

@Override

public void onBackPressed() {

if (mCurrentActivity instanceof MainActivity){

//如果是主页面

if (System.currentTimeMillis() - mPreTime > 2000) {// 两次点击间隔大于2秒

// UIUtils.showToast("再按一次，退出应用");

mPreTime = System.currentTimeMillis();

return;

}

}

super.onBackPressed();// finish()

}

public boolean isEventBusRegisted(Object subscribe) {

return EventBus.getDefault().isRegistered(subscribe);

}

public void registerEventBus(Object subscribe) {

if (!isEventBusRegisted(subscribe)) {

EventBus.getDefault().register(subscribe);

}

}

public void unregisterEventBus(Object subscribe) {

if (isEventBusRegisted(subscribe)) {

EventBus.getDefault().unregister(subscribe);

}

}

/\*\*

\* 申请运行时权限

\*/

public void requestRuntimePermission(String[] permissions, PermissionListener permissionListener) {

mPermissionListener = permissionListener;

List<String> permissionList = new ArrayList<>();

for (String permission : permissions) {

if (ContextCompat.checkSelfPermission(this, permission) != PackageManager.PERMISSION\_GRANTED) {

permissionList.add(permission);

}

}

if (!permissionList.isEmpty()) {

ActivityCompat.requestPermissions(this, permissionList.toArray(new String[permissionList.size()]), 1);

} else {

permissionListener.onGranted();

}

}

## 5.7Fragment的基类

public abstract class BaseFragment<T extends BasePresenter> extends LazyLoadFragment {

protected T mPresenter;

private View rootView;

protected StateView mStateView;//用于显示加载中、网络异常，空布局、内容布局

protected Activity mActivity;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

mPresenter = createPresenter();

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

if (rootView == null) {

rootView = inflater.inflate(provideContentViewId(),container,false);

ButterKnife.bind(this, rootView);

mStateView = StateView.inject(getStateViewRoot());

if (mStateView != null){

mStateView.setLoadingResource(R.layout.page\_loading);

mStateView.setRetryResource(R.layout.page\_net\_error);

}

initView(rootView);

initData();

initListener();

} else {

ViewGroup parent = (ViewGroup) rootView.getParent();

if (parent != null) {

parent.removeView(rootView);

}

}

return rootView;

}

/\*\*StateView的根布局，默认是整个界面，如果需要变换可以重写此方法\*/

public View getStateViewRoot() {

return rootView;

}

@Override

protected void onFragmentFirstVisible() {

//当第一次可见的时候，加载数据

loadData();

}

//用于创建Presenter和判断是否使用MVP模式(由子类实现)

protected abstract T createPresenter();

//得到当前界面的布局文件id(由子类实现)

protected abstract int provideContentViewId();

//加载数据

protected abstract void loadData();

@Override

public void onDestroy() {

super.onDestroy();

if (mPresenter != null) {

mPresenter.detachView();

mPresenter = null;

}

rootView = null;

}

public boolean isEventBusRegisted(Object subscribe) {

return EventBus.getDefault().isRegistered(subscribe);

}

public void registerEventBus(Object subscribe) {

if (!isEventBusRegisted(subscribe)) {

EventBus.getDefault().register(subscribe);

}

}

public void unregisterEventBus(Object subscribe) {

if (isEventBusRegisted(subscribe)) {

EventBus.getDefault().unregister(subscribe);

}

}

}

## 5.8频道编辑操作的实现

helper.setText(R.id.tvTitle, channel.title)

.setOnClickListener(R.id.tvEdit, new View.OnClickListener() {

@Override

public void onClick(View v) {

if (!mIsEdit) {

startEditMode(true);

helper.setText(R.id.tvEdit, "完成");

} else {

startEditMode(false);

helper.setText(R.id.tvEdit, "编辑");

}

}

});

break;

case Channel.TYPE\_OTHER:

//频道推荐

helper.setText(R.id.tvTitle, channel.title)

.setGone(R.id.tvEdit, false);

break;

case Channel.TYPE\_MY\_CHANNEL:

//我的频道列表

helper

.setVisible(R.id.ivDelete, mIsEdit && !(channel.title.equals("推荐")))//编辑模式就显示删除按钮

.setOnLongClickListener(R.id.rlItemView, new View.OnLongClickListener() {

@Override

public boolean onLongClick(View v) {

if (!mIsEdit) {

//开启编辑模式

startEditMode(true);

mEditViewHolder.setText(R.id.tvEdit, "完成");

}

if (onChannelDragListener != null)

onChannelDragListener.onStarDrag(helper);

return true;

}

}).setOnTouchListener(R.id.tvChannel, new View.OnTouchListener() { @Override

public void onClick(View v) {

//执行删除，移动到推荐频道列表

if (mIsEdit) {

int otherFirstPosition = getOtherFirstPosition();

int currentPosition = helper.getAdapterPosition() - getHeaderLayoutCount();

//获取到目标View

View targetView = mRecyclerView.getLayoutManager().findViewByPosition(otherFirstPosition);

//获取当前需要移动的View

View currentView = mRecyclerView.getLayoutManager().findViewByPosition(currentPosition);

// 如果targetView不在屏幕内,则indexOfChild为-1 此时不需要添加动画,因为此时notifyItemMoved自带一个向目标移动的动画

// 如果在屏幕内,则添加一个位移动画

if (mRecyclerView.indexOfChild(targetView) >= 0 && otherFirstPosition != -1) {

RecyclerView.LayoutManager manager = mRecyclerView.getLayoutManager();

int spanCount = ((GridLayoutManager) manager).getSpanCount();

int targetX = targetView.getLeft();

int targetY = targetView.getTop();

int myChannelSize = getMyChannelSize();

if (myChannelSize % spanCount == 1) {

//我的频道最后一行 之后一个，移动后

targetY -= targetView.getHeight();

}

//我的频道 移动到 推荐频道的第一个

channel.setItemType(Channel.TYPE\_OTHER\_CHANNEL);//改为推荐频道类型

if (onChannelDragListener != null)

onChannelDragListener.onMoveToOtherChannel(currentPosition, otherFirstPosition - 1);

startAnimation(currentView, targetX, targetY);

} else {

channel.setItemType(Channel.TYPE\_OTHER\_CHANNEL);//改为推荐频道类型

if (otherFirstPosition == -1) otherFirstPosition = mData.size();

if (onChannelDragListener != null)

onChannelDragListener.onMoveToOtherChannel(currentPosition, otherFirstPosition - 1);

}

// GlobalParams.mRemovedChannels.add(channel);

}

}

});

break;

case Channel.TYPE\_OTHER\_CHANNEL:

//频道推荐列表

helper.setText(R.id.tvChannel, channel.title).setVisible(R.id.ivDelete, false)

.setOnClickListener(R.id.tvChannel, new View.OnClickListener() {

@Override

public void onClick(View v) {

int myLastPosition = getMyLastPosition();

int currentPosition = helper.getAdapterPosition() - getHeaderLayoutCount();

//获取到目标View

View targetView = mRecyclerView.getLayoutManager().findViewByPosition(myLastPosition);

//获取当前需要移动的View

View currentView = mRecyclerView.getLayoutManager().findViewByPosition(currentPosition);

// 如果targetView不在屏幕内,则indexOfChild为-1 此时不需要添加动画,因为此时notifyItemMoved自带一个向目标移动的动画

// 如果在屏幕内,则添加一个位移动画

if (mRecyclerView.indexOfChild(targetView) >= 0 && myLastPosition != -1) {

RecyclerView.LayoutManager manager = mRecyclerView.getLayoutManager();

int spanCount = ((GridLayoutManager) manager).getSpanCount();

int targetX = targetView.getLeft() + targetView.getWidth();

int targetY = targetView.getTop();

int myChannelSize = getMyChannelSize();//这里我是为了偷懒 ，算出来我的频道的大小

if (myChannelSize % spanCount == 0) {

//添加到我的频道后会换行，所以找到倒数第4个的位置

View lastFourthView = mRecyclerView.getLayoutManager().findViewByPosition(getMyLastPosition() - 3);

// View lastFourthView = mRecyclerView.getChildAt(getMyLastPosition() - 3);

targetX = lastFourthView.getLeft();

targetY = lastFourthView.getTop() + lastFourthView.getHeight();

}

@Override

public void onAnimationStart(Animation animation) {

}

@Override

public void onAnimationEnd(Animation animation) {

parent.removeView(mirrorView);//删除添加的镜像View

if (currentView.getVisibility() == View.INVISIBLE) {

currentView.setVisibility(View.VISIBLE);//显示隐藏的View

}

}

@Override

public void onAnimationRepeat(Animation animation) {

}

});

}

/\*\*

\* 添加需要移动的 镜像View

\*/

private ImageView addMirrorView(ViewGroup parent, View view) {

view.destroyDrawingCache();

//首先开启Cache图片 ，然后调用view.getDrawingCache()就可以获取Cache图片

view.setDrawingCacheEnabled(true);

ImageView mirrorView = new ImageView(view.getContext());

//获取该view的Cache图片

Bitmap bitmap = Bitmap.createBitmap(view.getDrawingCache());

mirrorView.setImageBitmap(bitmap);

//销毁掉cache图片

view.setDrawingCacheEnabled(false);

int[] locations = new int[2];

view.getLocationOnScreen(locations);//获取当前View的坐标

int[] parenLocations = new int[2];

mRecyclerView.getLocationOnScreen(parenLocations);//获取RecyclerView所在坐标

FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(bitmap.getWidth(), bitmap.getHeight());

params.setMargins(locations[0], locations[1] - parenLocations[1], 0, 0);

parent.addView(mirrorView, params);//在RecyclerView的Parent添加我们的镜像View，parent要是FrameLayout这样才可以放到那个坐标点

return mirrorView;

}

private int ANIM\_TIME = 360;

/\*\*

\* 获取位移动画

\*/

private TranslateAnimation getTranslateAnimator(float targetX, float targetY) {

TranslateAnimation translateAnimation = new TranslateAnimation(

Animation.RELATIVE\_TO\_SELF, 0f,

Animation.ABSOLUTE, targetX,

Animation.RELATIVE\_TO\_SELF, 0f,

Animation.ABSOLUTE, targetY);

// RecyclerView默认移动动画250ms 这里设置360ms 是为了防止在位移动画结束后 remove(view)过早 导致闪烁

translateAnimation.setDuration(ANIM\_TIME);

translateAnimation.setFillAfter(true);

return translateAnimation;

}

/\*\*

\* 获取推荐频道列表的第一个position

\*

\* @return

\*/

private int getOtherFirstPosition() {

//之前找到了第一个pos直接返回

// if (mOtherFirstPosition != 0) return mOtherFirstPosition;

for (int i = 0; i < mData.size(); i++) {

Channel channel = (Channel) mData.get(i);

if (Channel.TYPE\_OTHER\_CHANNEL == channel.getItemType()) {

//找到第一个直接返回

return i;

}

}

return -1;

}

/\*\*

\* 我的频道最后一个的position

\*

\* @return

\*/

private int getMyLastPosition() {

for (int i = mData.size() - 1; i > -1; i--) {

Channel channel = (Channel) mData.get(i);

if (Channel.TYPE\_MY\_CHANNEL == channel.getItemType()) {

//找到第一个直接返回

return i;

}

}

return -1;

}

/\*\*

\* 开启编辑模式

\*/

private void startEditMode(boolean isEdit) {

mIsEdit = isEdit;

int visibleChildCount = mRecyclerView.getChildCount();

for (int i = 0; i < visibleChildCount; i++) {

View view = mRecyclerView.getChildAt(i);

ImageView imgEdit = (ImageView) view.findViewById(R.id.ivDelete);

if (imgEdit != null) {

boolean isVis = imgEdit.getTag() == null ? false : (boolean) imgEdit.getTag();

imgEdit.setVisibility(isVis && isEdit && !mData.get(i).title.equals("推荐") ? View.VISIBLE : View.INVISIBLE);

}

}

}

}

# 6总结

经过长达近一个月的的不懈努力，Android仿今日头条的软件终于完成了，使用用户可以通过我们的款新闻客户端查看实时新闻，还可以分类查看，设置自己喜欢的频道类型，还能够观看视频，小品，娱乐等众多分类，具有极佳的使用体验。

在这段实践设计过程中，发现Android的app开发需要用到许多的Java的基础知识，在这方面Java语言带给我了很多的方便，但是，由于之前并不是非常系统的学习java语言，导致在这过程中我们也遇到了许多的问题，不过最终通过我和队友的探讨以及查阅资料，我们还是克服了这些。

在之前的课程当中，我们并没有详细的讲解有关Fragment的知识，在应用中也出现了碎片的显示隐藏的相关问题，最终通过网上的一篇文章，通过学习搞懂了碎片的知识，然后发现这样用碎片做成的ui界面真的非常方便简洁。fragments 的设计不需要你来亲自管理view hierarchy 的复杂变化，通过将Activity 的布局分散到frament 中，可以在运行时修改activity 的外观，并且由activity 管理的back stack 中保存些变化。当一个片段指定了自身的布局时，它能和其他片段配置成不同的组合，在活动中为不同的屏幕尺寸修改布局配置。

同时，在这次的实验中，我们运用了许多第三方的库以及插件，比如BottomBarLayout(轻量级底部导航栏)、BaseRecyclerViewAdapterHelper(ReclerView万能适配器)、StateView(用于几种状态布局的切换)、JieCaoVideoPlayer，改名 JiaoZiVideoPlayer (视频播放)、BGARefreshLayout-Android(下拉刷新)、Eyes(修改状态栏颜色)、KLog(log管理)等等。通过对这些第三方库的学习，并不断地进行尝试，我们最终将这些库的功能在我们的软件中实现。发现在以后的开发中，很有必要善于运用第三方库，这样不仅方便自己，而且能让自己的程序变的更加方便整洁，运行速度更加快速，不出现卡顿情况。

总之，在这段时间里，我们小组遇到多很多困难，但是通过团队各种方式，这些困难问题都被一一解决，并且我们每个人都从中学到了许多许多，不仅仅是把整个学期的课堂内容贯穿在一起，还融入了大量的新鲜的东西，都受益匪浅。这也极大地鼓舞了我们每个人，让我们在以后的IT之路上更加自信，更加努力！