VaeMusic

嵩鼠音乐播放和分享 APP

摘要

本项目基于 Android 和 SpringBoot 开发，使用 okHttp+Retrofit 实现音乐、动态等数据的网络请求，使用安卓自带数据库 sqlite 保存音乐播放列表数据，实现了音乐播放、动态发布、登录、注册等多项功能，致力于打造一个简洁但是美观、功能完备的音乐播放类 APP 和允许许嵩粉丝发表帖子进行交流的社交类 APP。

10205101530 赵晗瑜

10205101452 陈黎明

2022 年 12 月 15 日

目录

一、 项目背景 3
二、 系统架构 3
1. 开发环境 3
2. 运行环境 3
3. 系统架构图 4
三、 项目重难点 4
1. 黑胶唱片相关布局 4
2. 黑胶唱片相关功能 5
3. 音乐播放相关功能 5
4. 迷你音乐播放控制器相关功能 5
四、 实现方法 6
1）实现方法总述 6
2) 项目准备工作 6
3) 实现启动界面相关功能 7
4) 实现主界面相关功能 9
5) 实现音乐播放相关功能 13
6) 实现音乐播放列表相关功能 17
7) 实现黑胶唱片相关功能 19
8) 实现我的界面相关功能 20
9) 实现用户登陆相关功能 21
10) 实现用户注册相关功能 23
11) 实现帖子列表相关功能 23
12) 实现发布帖子相关功能 26
13) 实现用户详情相关功能 28
五、 实现效果 28
六、 总结和展望 31
七、 附录 33
1. 参考文档： 33
2. 源码地址： 33

一、项目背景

1. 吸引相同的粉丝共同听老歌

许嵩早年的歌曲要么在网易云音乐上没有版权，要么在 QQ 音乐上没有版权，因此我们为许嵩早年歌曲提供一个承载平台，吸引许嵩的歌迷共同听取许嵩早年的歌曲，共同怀旧，共同交流。

2. 吸引相同的粉丝共同交流分享

用手机听音乐已变成当代年轻人的标配，许多人都有自己喜欢的音乐风格或歌唱组合等，但有时**难以找到同好**。目前，市场上不同的音乐播放软件层出不穷，但鲜少有结合音乐播放和歌迷圈交流的双重功能。本项目将整合音乐播放和社交的双重功能，以歌手许嵩为例，收集该歌手的歌曲作品；搭建一个干净、有序的社交圈，允许加入该歌手圈子的用户在同一空间发布与音乐、生活相关的感悟。

二、系统架构

本项目以安卓平台作为载体，在前期完成 MySQL 数据库本地部署和项目功能实现，中后期进一步完善 UI 设计并搭建服务器。后期进一步测试和优化软件，之后投入使用。最终呈现一款围绕许嵩的，具有听歌和发帖功能的音乐类和社交类 APP。

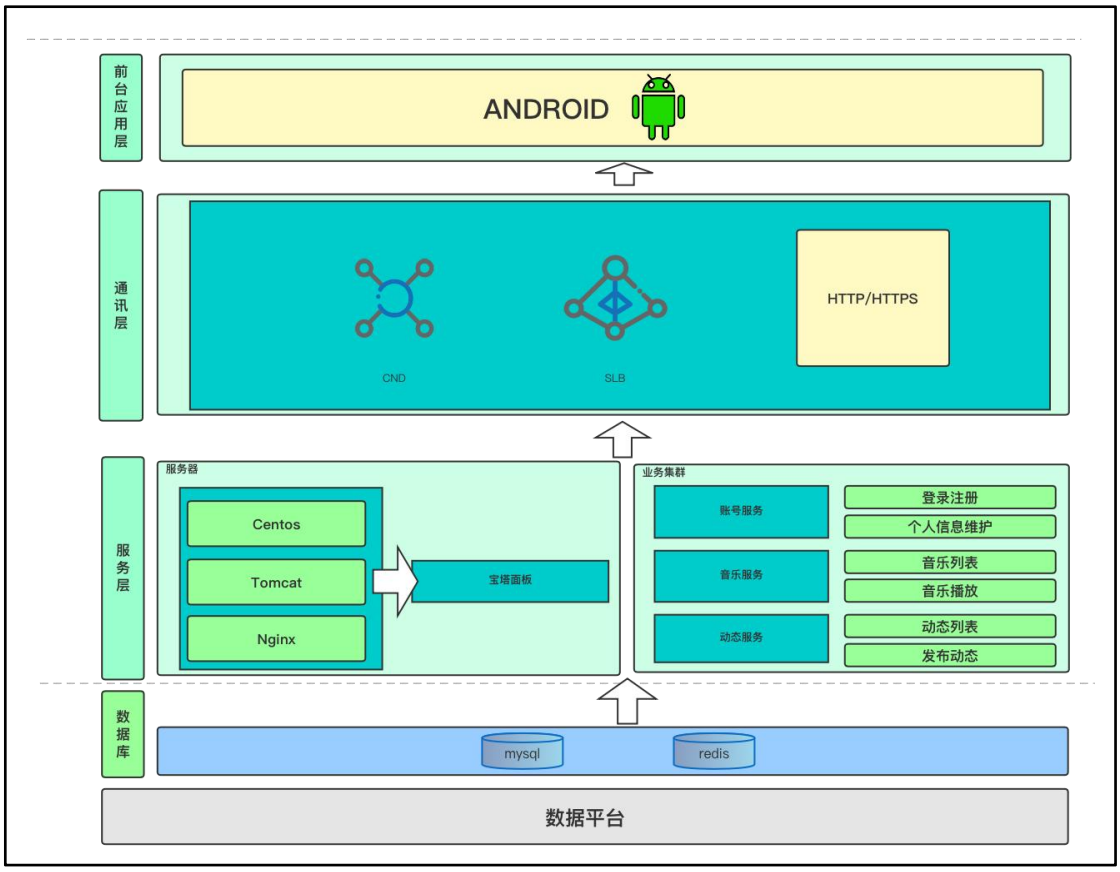
1. 开发环境

- 客户端：Android Studio 2021.2
- 服务端：IDEA 2021.3
  - JDK 1.7 + SpringBoot 2.3.7

2. 运行环境

- 客户端：Android 8.0 以上
- 服务端：CentOS 7.8

3. 系统架构图



架构图说明：

本项目的系统架构分为四层，从上到下依次为：前台应用层、通讯层、服务层和数据层。

1) **业务层**在 Android 客户端实现。
2) **通讯层**使用 HTTP/HTTPS 协议，协调服务器负载均衡和内容分发。
3) **服务层**的服务器在 Centos 环境上运行，配置 Tomcat 和 Nginx 实现 Web 服务器资源访问，使用宝塔面板操作项目业务。
4) **业务层子模块**内部职责使用 MVC 架构，项目业务包括账号服务、音乐服务和动态服务，实现用户的登录注册、个人信息维护，音乐列表及播放，动态列表及发布。
5) **数据库：**服务端使用 MySQL 和 Redis 接入服务数据，客户端使用 Sqlite 接入个人数据。

三、项目重难点

1.黑胶唱片相关布局

黑胶唱片布局实现较为困难，自定义 View 存放相关黑胶唱片组件，

该 View 的 xml 文件中使用相对布局。

2.黑胶唱片相关功能

控制黑胶唱片的旋转、音乐进度条的拖拽、音乐播放模式的切换、音乐暂停时拖拽进度条后继续自动播放。

3.音乐播放相关功能

新建**音乐播放管理器**来控制音乐的播放、暂停、播放列表的设置、删除、恢复、获取上一首和下一首音乐、更改循环模式、音乐进度的拖拽、删除播放列表管理器中的某首音乐等事件，通过这样一个管理器来控制和协调音乐播放的相关功能是我们的难点。

4. 迷你音乐播放控制器相关功能

创建一个迷你音乐播放控制器放在首页，点击可以进到相应的音乐播放详情界面，使用 EventBus 监听音乐列表的的改变，当音乐播放列表没有音乐时隐藏播放控制器。

5. 其他

① 使用 **SharedPreferences** 偏好配置类实现保存最后一次播放音乐的 id，当用户关闭应用（杀死进程）重新打开时，播放的是上次关闭时播放的最后一首音乐；同时在偏好配置中保存用户的登录状态，id，session 等信息以实现登录后不退出登录且 session 不过期一直保持登录状态；

② 使用 **EventBus** 将音乐播放列表（不是首页的音乐列表，是点击音乐播放界面右下角的列表按钮弹出的列表 dialog 中的音乐播放列表）的改变的事件通知给外界，改变的事件包括删除某首音乐，删除所有音乐，如果没有音乐播放列表（即当前没有音乐在播放），则隐藏首页的迷你音乐播放器；

③ 设置**音乐播放管理器 MusicPlayerManager** 和**音乐列表管理器 MusicListManager** 统一管理音乐播放相关事件和音乐列表改变相关事件；

④ 进行 **okHttp+Retrofit** 的封装方便后续请求网络接口以获取数据的调用；

⑤ 设置不同的 xml 进行**夜间模式**的切换；

四、实现方法

该项目实现了：

- 用户登录注册及个人信息维护；
- 获取服务器音乐资源并在线播放；
- 歌曲的停止、播放、拖拽进度、上/下一首切换；
- 音量调节；
- 播放模式切换；
- 黑胶唱片旋转播放；
- 发布多媒体动态；
- 获取动态列表；
- 内嵌 h5 访问开发团队网页；
- 转至拨号拨打客服电话。

1）使用第三方框架

- 使用 io.github.lucksiege:pictureselector 图片选择器进行发布动态时图片的选择；
- 使用 io.github.scwang90:refresh-header-classics:2.0.5 实现经典刷新头；
- 使用 io.github.scwang90:refresh-footer-classics:2.0.5 实现经典加载头；
- 使用 io.github.lucksiege:compress:v3.0.9 实现图片压缩；
- 使用 org.greenrobot:eventbus:3.3.1 跨界面通信框架；
- 使用 jp.wasabeef:glide-transformations:+实现高斯模糊效果；
- 使用 de.hdodenhof:circleimageview:+圆形 UI 控件；
- 使用 com.github.permissions-dispatcher:permissionsdispatcher:4.9.2 实现动态处理权限；
- 使用 io.github.h07000223:flycoTabLayout:3.0.0 实现 tabbar；
- 使用 com.github.CymChad:BaseRecyclerViewAdapterHelper 简化 RecyclerView 操作。

2) 项目准备工作

(一)封装相关通用逻辑

- 在 BaseActivity（所有 Activity 的父类）中将 onPostCreate 拆分为 initViews（找控件），initDatum（设置数据），initListeners（设置监听器），方便进行管理
- 在 BaseCommonActivity（继承 BaseActivity）中写可以复用的逻辑
- 在 BaseLogicActivity（继承 BaseCommonActivity）中写背景颜色，全局的迷你音乐播放控制器等相关信息
- 在 BaseTitleActivity（继承 BaseLogicActivity）中写标题相关的通用逻辑

(二)封装 ViewBinding

对 ViewBinding 进行封装，避免每次都要 setContentView

```
1. public class BaseViewModelActivity<VB extends ViewBinding> extends Bas
   eLogicActivity{
2.    protected VB binding;
3.    @Override
4.    protected void onCreate(Bundle savedInstanceState) {
5.        super.onCreate(savedInstanceState);
6.        //调用 inflate 方法，创建 viewBinding
7.        binding = ReflectUtil.newViewBinding(getLayoutInflater(),this.getClass())
   ;
8.        setContentView(binding.getRoot());
9.    }
10. }
```

3) 实现启动界面相关功能

、

① 布局：

布局实现如上图所示，使用 RelativeLayout，同时为每个 Activity 去除 ActionBar，使用 androidx.appcompat.widget.Toolbar,并对 Toolbar 相关的逻辑进行封装，方便复用。

② 沉浸式状态栏：

使用 com.qmuiteam:qmui:2.0.1 UI 框架实现:

```
1.  QMUIStatusBarHelper.translucent(this);
```

③ 动态权限处理:

使用第三方框架 permissionsdispatcher:4.9.2，在 checkPermisson 方法中让动态框架检查用户是否授权:

```
1. private void checkPermissions() {
2.    StartActivityPermissionsDispatcher.onPermissionGrantedWithPermissionCh
    eck(this);
3. }
```

如果用户授权了就进入主界面，如果用户拒绝授予权限，则告知其为什么要用到这些权限，如果用户拒绝授予权限则关闭应用。

不论用户是选择授予权限还是拒绝授予权限，系统都会调用 onRequestPermissionResult 方法，将授权的结果传递到该框架:

```
1. @Override
```

```
2. public void onRequestPermissionsResult(int requestCode, @NonNull String[]
   permissions, @NonNull int[] grantResults) {
3.     super.onRequestPermissionsResult(requestCode, permissions, grantResults);

4.     将授权结果传递到框架
5.     StartActivityPermissionsDispatcher.onRequestPermissionsResult(this, reque
   stCode, grantResults);
6. }
```

4) 实现主界面相关功能

① 底部 tabbar：

主页面的底部式 tabbar，中间是内容，使用 flycoTabLayout:3.0.3 实现类似 TabLayout 的控件。

```
1.  <com.flyco.tablayout.CommonTabLayout
2.      android:id="@+id/indicator"
3.      android:layout_width="match_parent"
4.      android:layout_height="@dimen/d65"
5.      app:tl_iconHeight="@dimen/d30"
6.      app:tl_iconWidth="@dimen/d30"
7.      app:tl_indicator_color="?attr/colorPrimary"
```

```
8.      app:tl_indicator_height="0dp"
9.      app:tl_textSelectColor="#638B4C"
10.       app:tl_textUnselectColor="?attr/colorOnSurface"
11.       app:tl_textsize="@dimen/s12"
12.       app:tl_underline_height="0dp" />
```

底部 tabbar 点击底部选项时进入该选项的内容并设置**滑动联动效果**：

```
1. binding.indicator.setOnTabSelectListener(new OnTabSelectListener() {
2.     @Override
3.     public void onTabSelect(int position) {
4.         binding.list.setCurrentItem(position);
5.     }
6. });
```

实现**手部进行滑动手势**时，底部的 toolbar 也进行切换:

```
1. binding.list.addOnPageChangeListener(new OnPageChangeListenerAdapter()
     {
2.     /**
3.      * 滚动完成
4.      * @param position
5.      */
6.     @Override
7.     public void onPageSelected(int position) {
8.         binding.indicator.setCurrentTab(position);
9.     }
10. });
```

② 从网络接口请求音乐数据

在“聆听”页面中中间的音乐列表内容的数据从调用网络接口返回的数据中获得，在调用之前，我们已经将 **okHttp+Retrofit** 进行了封装。

调用请求音乐的网络接口：

```
1. DefaultRepository.getInstance()
```

```
        .songs()
        .subscribe(new HttpObserver<ListResonse<Song>>() {
        @Override
         public void onSuccesseded(ListResonse<Song> data) {
         adapter.setNewInstance(data.getData().getData());
     }
}});
```

③ 音乐列表管理器 MusicListManager

新建一个**音乐列表管理器**来管理音乐列表相关功能，如**继续播放、更改循环模式、获取下一首音乐、拖拽音乐进度、从列表管理器中删除某一首音乐（从数据库中删除）、删除列表管理器中的所有音乐（从数据库中删除）、从数据库初始化播放列表**等。

点击音乐列表中的一首音乐便将其添加到播放列表：

```
//点击一首音乐就把所有的音乐添加到播放列表
MusicListManager.getInstance(getHostActivity()).setDatum((List<Song>) adapter.getData());
```

④ 音乐添加到 sqlite 数据库

在播放列表管理器中将音乐添加到数据库：

```
//添加新的播放列表
this.datum.addAll(datum);
//将播放列表保存至数据库
orm.saveAll(datum);
```

⑤ 从 sqlite 数据库中恢复播放列表

即从数据库中查询所有的播放列表并进行初始化。

⑥ 布局

音乐列表使用 RecyclerView，单首音乐布局使用 ConstraintLayout

为音乐列表设置线性布局管理器：

```
LinearLayoutManager layoutManager=new LinearLayoutManager(getHostActivity());
binding.list.setLayoutManager(layoutManager);
```

添加分割线：

```
1. DividerItemDecoration decoration=new DividerItemDecoration(binding.list.get
   Context(), RecyclerView.VERTICAL);
2. binding.list.addItemDecoration(decoration);
```

⑦ 迷你音乐播放器

在“聆听界面”的左下角有一个迷你音乐播放控制器，点击其可以跳转到相应的播放页面，布局使用 RelativeLayout，黑胶唱片使用 ImageView，封面使用 de.hdodenhof.circleimageview.CircleImageView 第三方组件

使用 EventBus 将播放列表改变的事件通知给外界：

```
1. /**
2.  * 发出一个通知给外界
3.  * @param i
4.  */
5. private void sendPlayListChangedEvent(int i) {
6.     EventBus.getDefault().post(new MusicPlayListChangedEvent(i));
7. }
```

当播放列表没有的时候隐藏迷你音乐播放控制器：

```
1. if(getMusicListManager().getDatum().size()>0){
2.     binding.musicControl.onResume();
3.     binding.musicControl.setVisibility(View.VISIBLE);
4. }else{
5.     binding.musicControl.setVisibility(View.GONE);
6. }
```

迷你音乐播放控制器控制其旋转：

```
1. private void showProgress() {
2.      旋转
3.     incrementRotate();
4. }
```

迷你音乐播放控制器获取当前播放的音乐并进行设置：

```
1. private void showInitData() {
2.     获取当前播放的音乐
3.     Song data= MusicListManager.getInstance(getContext()).getData();
4.     if(data==null){
5.         return;
6.     }
7.     ImageUtil.show(binding.icon,data.getPic());
8. }
```

实现音乐监听接口，当音乐准备播放时初始化音乐数据：

```
1. //音乐监听接口实现，音乐准备播放了就初始化数据
2. @Override
3. public void onPrepared(MediaPlayer mp, Song data) {
4.     showInitData();
5. }
```

如果音乐在播放中就显示播放进度并进行旋转：

```
1.         //音乐在播放中就显示进度，旋转
2. @Override
3. public void onProgress(Song data) {
4.     showProgress();
5. }
```

5)　实现音乐播放相关功能

新建一个音乐播放的管理器 **MusicPlayerManager**，管理音乐的播放、暂停、播放列表的设置、删除、恢复、获取上一首和下一首音乐、更改循环模式、音乐进度的拖拽、删除播放列表管理器中的某首音乐等事件。

在 MusicPlayerManager 中初始化播放器 player:player=new MediaPlayer();

监听音乐播放器准备完毕，音乐准备完毕后获取音乐的总时长并将音乐的总进度保存到音乐对象，并设置播放完毕播放器：

```
1. player.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
2.     @Override
3.     public void onPrepared(MediaPlayer mediaPlayer) {
4.         //获取音乐的总时长
5.         //将总进度保存到音乐对象
6.         data.setDuration(mediaPlayer.getDuration());
7.         for(MusicPlayerListener it:listeners){
8.             it.onPrepared(mediaPlayer,data);
9.         }
10.     }
11. });
```

使用系统播放器来播放音乐：

```
1. public void play(String uri, Song data) {
2.    //保存信息
3.    this.uri=uri;
4.    this.data=data;
5.    //释放播放器
6.    player.reset();
7.    playNow();
8. }
```

playNow 函数：

```
1. private void playNow() {
2.       try {
3.          if (uri.startsWith("content://")) {
4.             player.setDataSource(context, Uri.parse(uri));
5.          } else {
6.             player.setDataSource(uri);
7.          }
8.          //同步准备播放
9.          player.prepare();
10.           //开始播放
11.           player.start();
12.           /**
13.            * 回调监听器
14.            */
15.           publishPlayingStatus();
16.           startPublishProgress();
17.       } catch (IOException e) {
18.           //播放错误处理
      }
19.
```

```
20.    }
```

实现音乐进度的**拖拽**：

```
1. public void seekTo(int data) {
2.    //跳转到指定位置播放
3.    player.seekTo(data);
4. }
```

在 MusicListManager 中实现跳转到执行位置播放音乐：

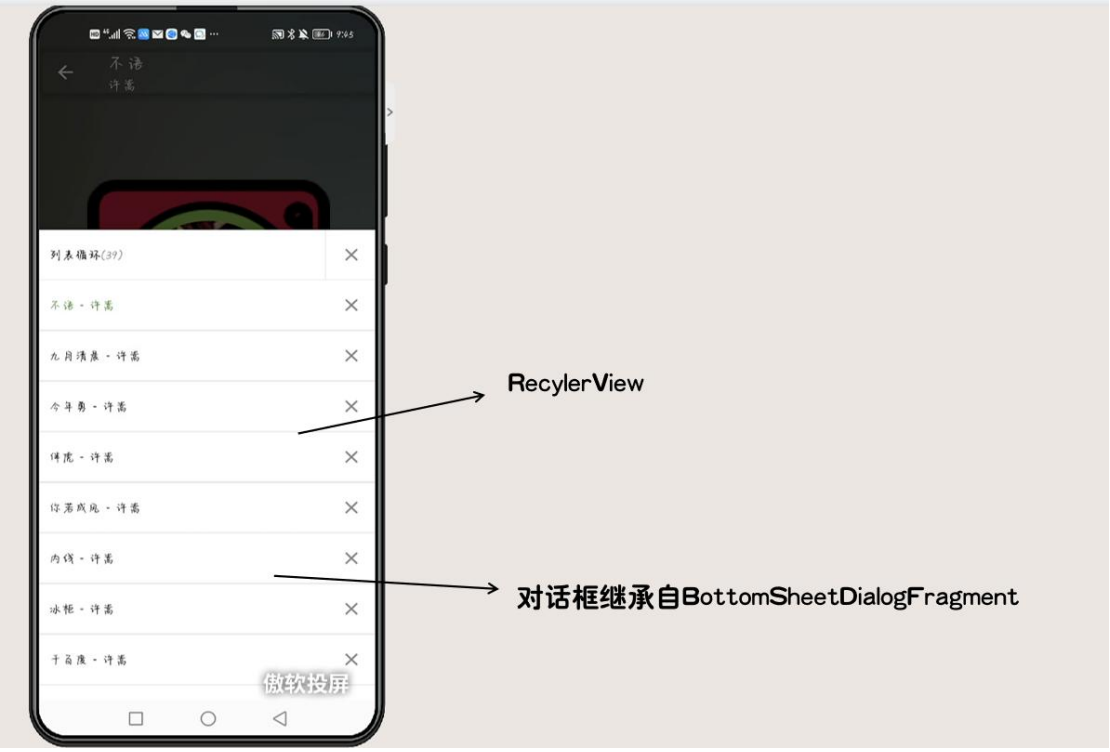
```
1. public void seekTo(int data) {
2.    if(!musicPlayerManager.isPlaying()){
3.        resume();
4.    }
5.    musicPlayerManager.seekTo(data);
6. }
```

在 MusicPlayerActivity 中**监听音乐进度拖拽**的改变，监听到则调用 seekTo 函数跳转到执行位置播放音乐：

```
1. /**
2.    * 进度改变了
3.    * @param seekBar
4.    * @param progress  当前进度
5.    * @param fromUser  是否是用户拖拽的
6.    */
7.   @Override
8.   public void onProgressChanged(SeekBar seekBar, int progress, boolean fro
     mUser) {
9.       if(fromUser){
10.          getMusicListManager().seekTo(progress);
11.      }
12.   }
13.   /**
14.    * 开始拖拽进度时
```

```
15.     * @param seekBar
16.     */
17.    @Override
18.    public void onStartTrackingTouch(SeekBar seekBar) {
19.        isSeekTracking=true;
20.    }
21.    /**
22.     * 停止拖拽
23.     * @param seekBar
24.     */
25.    @Override
26.    public void onStopTrackingTouch(SeekBar seekBar) {
27.        isSeekTracking=false;
28.    }
29.
30.    //endregion
```

6) 实现音乐播放列表相关功能



弹出框继承自 **BaseViewModelBottomSheetDialogFragment**，即从下到上弹出，并使用 RecyclerView。

监听播放列表中音乐的点击事件，获取当前的音乐播放并关闭弹窗：

```
1. adapter.setOnItemClickListener(new OnItemClickListener() {
2.     @Override
3.     public void onItemClick(@NonNull BaseQuickAdapter<?, ?> adapter, @N
    onNull View view, int position) {
4. // 获取当前音乐播放
5. // 关闭弹窗
6.     dismiss();
7.     getMusicListManager().play(getMusicListManager().getDatum().get(positio
    n));
8. }
9. });
```

删除播放列表中的某一首音乐或所有音乐即调用 MusicListManager 中的 delete 方法，在 delete 方法中删除数据库中的该音乐，从数据库恢复播放列表时即没有这首音乐。

在显示播放列表时即展示音乐数据、循环模式、一共有多少首等初始化数据：

```
@Override
protected void initDatum() {
    super.initDatum();
    //给 recyclerView 设置数据
    adapter = new MusicPlayListAdapter(R.layout.item_play_list,getMusicListManager());
    binding.list.setAdapter(adapter);
    adapter.setList(getMusicListManager().getDatum());
    showLoopModel();
    showCount();
}
```

7) 实现黑胶唱片相关功能

① 黑胶唱片布局使用自定义 View:

```
<com.zhy.view.RecordPageView
    android:id="@+id/record"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1" />
```

在 RecordPageView 中使用 ConstraintLayout，将圆形图片相对父组件的比例为 0.64，使用相对布局中的

```
app:layout_constraintWidth_percent="0.64"
```

可以使其适配不同大小的屏幕，不至于在平板屏幕太大时图片太小。为了可以让黑胶唱片的图片封面能够实现旋转功能，采用第三方控件 de.hdodenhof.circleimageview.CircleImageView。

使用如下方法控制其旋转功能：

```
/**
 * 显示音乐播放进度
 */
```

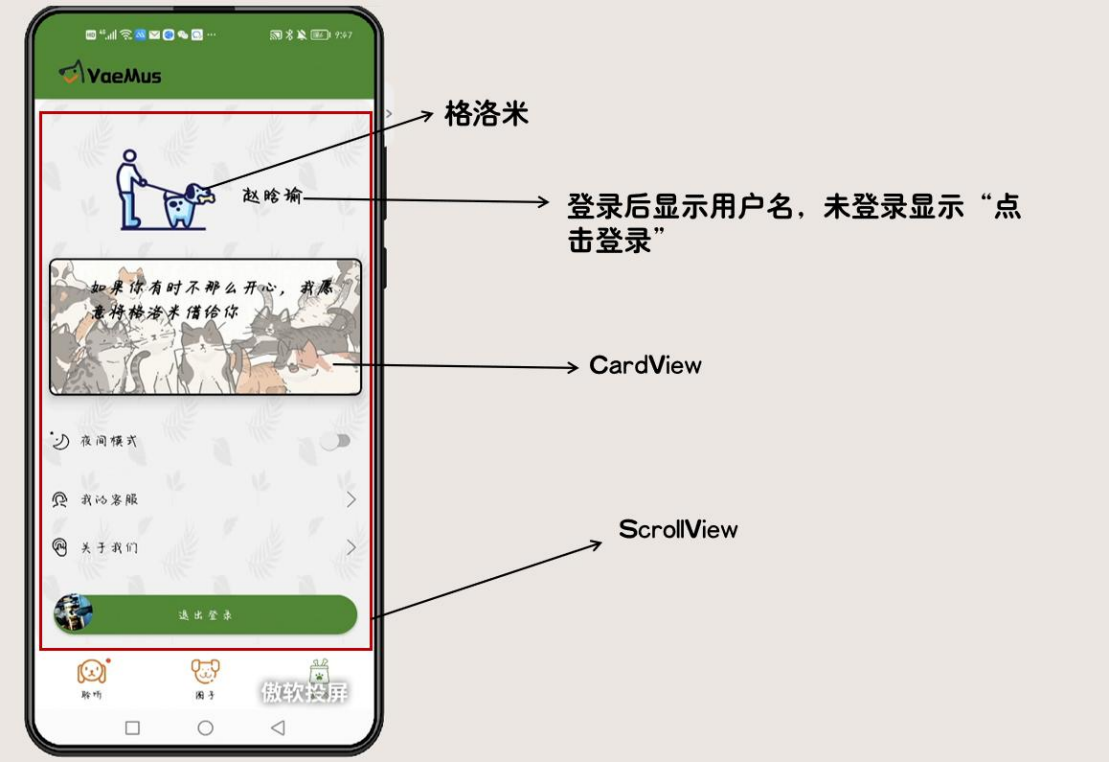
```
4. private void showProgress() {
5.     binding.record.incrementRotate();
6.     //如果当前在拖拽，不显示当前进度
7.     if(isSeekTracking){
8.         return;
9.     }
10.     //音乐的播放进度
11.     int progress=getMusicListManager().getData().getProgress();
12.      格式化进度
13.     binding.start.setText(SuperDateUtil.ms2ms(progress));
14.     //将进度设置到播放条上去
15.     binding.progress.setProgress(progress);
16. }
```

② **实现音乐播放背景高斯模糊效果**

使用第三方控件 implementation 'jp.wasabeef:glide-transformations:+'，采用如下方法实现高斯模糊效果，new BlurTransformation 中的两个参数分别是模糊半径和采样率，值越大越模糊：

```
//实现高斯模糊
RequestOptions requestOptions = RequestOptions.bitmapTransform(new BlurTransformation( radius: 50, sampling: 3));
//加载图片
requestBuilder.apply(requestOptions)
        .into(new CustomTarget<Drawable>() {
            /**
             * 图片加载完成
             * @param resource
             * @param transition
             */
            @Override
            public void onResourceReady(@NonNull Drawable resource, @Nullable Transition<? super Drawable> transition) {
                //将背景图片设置为高斯模糊后的照片
                binding.background.setImageDrawable(resource);
            }
```

8) 实现我的界面相关功能



页面整体使用 ScrollView，避免因某些手机屏幕较小看不到界面组件，中间的卡片使用 MaterialCardView，为其设置背景图片和边框。

界面上方的姓名，如果用户登录了则显示其姓名，点击会跳转到用户详情信息界面；如果用户未登录则显示“立即登录”，点击会跳转到登录界面：

```
1. View.OnClickListener userClick = view -> {
2.     if(sp.isLogin()){
3.         //已经登录了,跳转用户详情
4.         Intent intent = new Intent(getHostActivity(), UserDetailActivity.class;
5.         intent.putExtra(Constant.ID,sp.getUserId());
6.         startActivity(intent);
7.     }else{
8.         //没有登录，跳转到登录界面
9.         startActivity(LoginActivity.class);
10.     }
11. };
```

9) 实现用户登陆相关功能

① 调用登录网络接口，服务端返回 session 和 user_id:

```
1. DefaultRepository.getInstance()
2.         .login(password,phone)
3.         .subscribe(new HttpObserver<DetailResponse<Session>>() {
4.             /**
5.              * 登录成功
6.              * @param data
7.              */
8.             @Override
9.             public void onSucceeded(DetailResponse<Session> data) {
10.                 onLogin(data.getData().getId());
11.             }
12.         });
```

② 将返回的登录信息保存在 PreferenceUtil 偏好配置中：

```
    private void onLogin(String id) {
        //保存userId到偏好配置
        sp.setUserId(id);
        sp.setSession(id);
//      将登录事件代理到AppContext中
//      其他的功能可能也需要用户登录
        AppContext.getInstance().onLogin(id);
//      登陆完成后，关闭所有界面
        MyActivityManager.getInstance().finishAllLogin();

    }
}
```

每发起一个网络请求时，对网络请求进行拦截并添加请求头，value 的值为登录时服务端返回的 token：

```
builder.addNetworkInterceptor(new Interceptor() {
    @NonNull
    @Override
    public Response intercept(@NonNull Chain chain) throws IOException {
        //执行每个网络请求时，在这里可以拦截到
        PreferenceUtil sp=PreferenceUtil.getInstance(AppContext.getInstance());
        Request request=chain.request();
        if(sp.isLogin()){
            //如果登录了，获取token
            //如果没有登录，服务器端就获取不到这个请求头，网络请求会中断
            String session=sp.getSession();
            //为网络请求添加请求头
            request=request.newBuilder()
                    .addHeader("Cookie",session)
                    .build();
        }
        //继续执行网络请求
        return chain.proceed(request);
    }
});
```

10) 实现用户注册相关功能

注册功能和登录功能类似，唯一不同的是在调用注册网络接口时需要首先在客户端判断输入手机号是否符合正则表达式，两次密码输入是否一致，服务端会判断手机号是否已经被注册。

11) 实现帖子列表相关功能

帖子列表的数据从调用的网络接口返回的数据（已进行分页）中获得，每次调用请求 10 个列表：

```
1. DefaultRepository.getInstance()
2.         .feeds(param)
3.         .subscribe(new HttpObserver<ListResonse<Feed>>() {
4.         @Override
5.         public void onSucceeded(ListResonse<Feed> data) {
6.             pageMeta=data.getData();
7. //结束刷新
8.             binding.refresh.finishRefresh(2000,true,false);
9.             binding.refresh.finishLoadMore(2000,true,pageMeta.getNext()==null);

10.             if(isRefresh){
11.                 isRefresh=false;
12. //下拉刷新
13.                 adapter.setNewInstance(data.getData().getData());
14.             }else{
```

```
15.                 adapter.addData(data.getData().getData());
16.             }
17.         }
18. });
```

监听下拉刷新和上拉加载更多，当执行下拉刷新时，即请求下一页数据，当执行上拉加载更多时，则始终请求第一页数据，实现刷新效果：

```
1. //       下拉刷新监听器
2. binding.refresh.setOnRefreshListener(new OnRefreshListener() {
3.     @Override
4.     public void onRefresh(@NonNull RefreshLayout refreshLayout) {
5.         loadData();
6.     } });
7. //       上拉加载更多监听器
8. binding.refresh.setOnLoadMoreListener(new OnLoadMoreListener() {
9.     @Override
10.     public void onLoadMore(@NonNull RefreshLayout refreshLayout) {
11.         loadmore();
12.     }});
```

帖子列表的布局使用 RecyclerView，每一个帖子布局使用 LinearLayout，帖子中的九宫格图片使用 RecyclerView 并为其设置九宫格管理器：

```
1. //为 RecyclerView 设置布局管理器,九宫格
2. GridLayoutManager layoutManager = new GridLayoutManager(getContext(),
     spanCount);
3. //将布局管理器设置到控件上
4. listView.setLayoutManager(layoutManager);
5. //添加分割线
6. if(listView.getItemDecorationCount()>0){
7.     listView.removeItemDecorationAt(0);
8. }
```

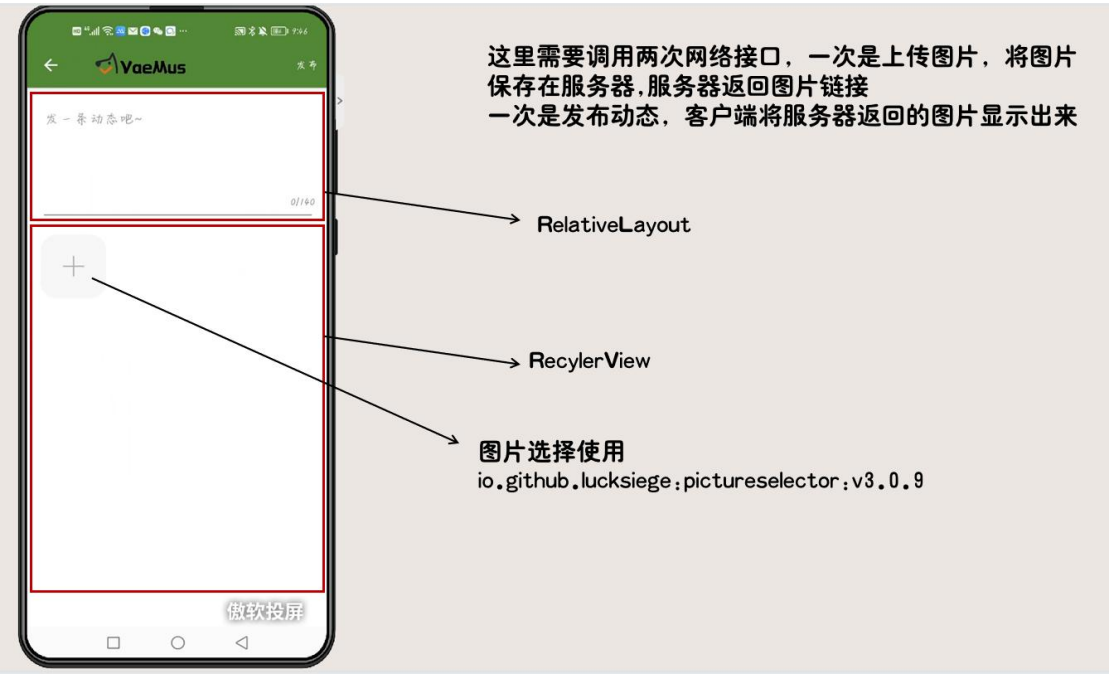
```
9. GridDividerItemDecoration divider = new GridDividerItemDecoration(getCont
    ext(), (int) DensityUtil.dip2px(getContext(), 5));
10. listView.addItemDecoration(divider);
```

图片默认为一列，当图片的数量大于 1 小于 4 时为两列，当图片的数量大于 4 小于 9 时为三列。

图片加载使用 Glide:

```
1. Glide.with(view.getContext())
2.         .load(data)
3.         .apply(options)
4.         .into(view);
```

12) 实现发布帖子相关功能



发布帖子需要调用两个网络接口，第一个是上传图片到服务器的网络接口，服务器返回图片的 url；第二个是发布动态的网络接口，服务器将返回过的图片集和动态文字保存到服务器的数据库并添加到帖子列表。

① 上传图片

上传图片即使用第三方上传多张图片的控件和压缩图片的控件：

```
1. implementation 'io.github.lucksiege:pictureselector:v3.0.9'
```

```
2. implementation 'io.github.lucksiege:compress:v3.0.9'
```

使用时即使用 pictureSelector：

```
1. private void selectImage() {
2.     PictureSelector.create(this)
3.             .openGallery(SelectMimeType.ofImage())
4.             .setImageEngine(GlideEngine.createGlideEngine())
5.             .setMaxSelectNum(9)
6.             .setMinSelectNum(1)
7.             .setImageSpanCount(3)
8.             .setSelectionMode(SelectModeConfig.MULTIPLE)
9.             .isPreviewImage(true)
10.              .isDisplayCamera(true)
11.              .setCameraImageFormat(PictureMimeType.JPEG)
12.              //压缩图片.....
```

② 调用上传图片的网络接口,即先将选择的图片创建成 MultipartBody，将每一个 MultipartBody 添加到 bodyFiles 并作为上传图片的网络接口的第一个参数：

```
private void uploadImages(List<LocalMedia> selectedImages) {
    ArrayList<MultipartBody.Part> bodyFiles = new ArrayList<>();
    for(LocalMedia it:selectedImages){
        File file = new File(it.getCompressPath());
        //创建成文件表单项
        RequestBody fileBody = RequestBody.Companion.create(file, MediaType.parse("image/*"));
        MultipartBody.Part multipartBody = MultipartBody.Part.createFormData( name: "file", file.getName(), fileBody);
        bodyFiles.add(multipartBody);

    }
    RequestBody flavorBody = RequestBody.Companion.create( $this$toRequestBody: "dev", MediaType.parse("multipart/form-data"));
//      调用接口
    DefaultRepository.getInstance().uploadFiles(bodyFiles,flavorBody)
            .subscribe((HttpObserver) (data) -> {
                saveFeed(data.getData().getData());
            });

}
```

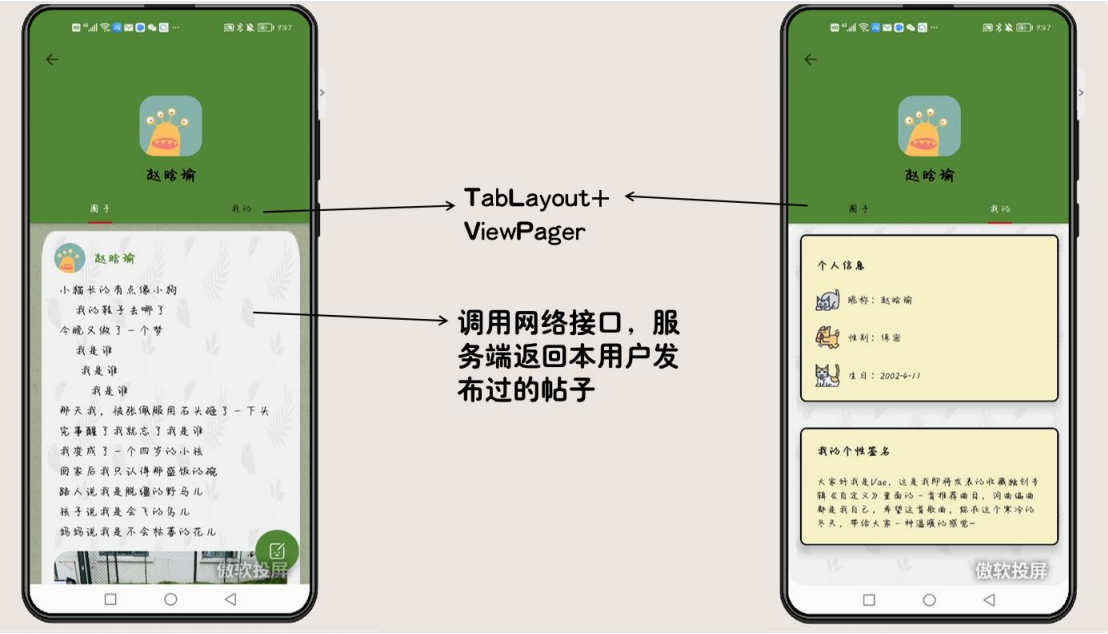
③ 调用发布动态的网络接口

用户点击发布按钮后，系统便会调用 saveFeed 方法，将图片集和图片文字作为参数传递到服务端：

```
1. DefaultRepository.getInstance()
2.         .createFeed(content,StringUtils.join(medias,","), id)
```

```
3.          .subscribe(new HttpObserver<DetailResponse<BaseId>>() {
4.              @Override
5.              public void onSucceeded(DetailResponse<BaseId> data) {
6.                  EventBus.getDefault().post(new FeedChangedEvent());
7.                  //发布动态成功
8.                  finish();
9.              }
10.         });
```

13) 实现用户详情相关功能



五、实现效果

我们实现了上文所述的所有功能，如下是手机 APP 界面截图，详情参见《使用说明书》

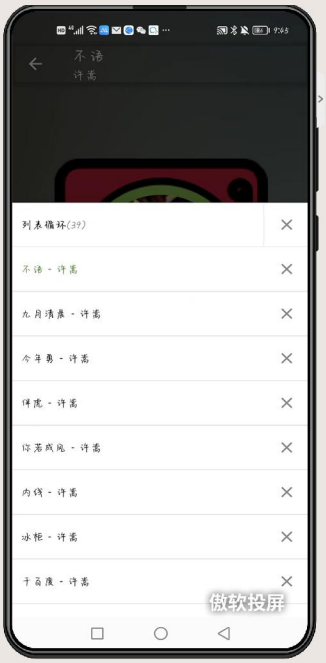
apk 下载：https://github.com/hot-zhy/VaeMusic


听一首歌 放空你的思绪
Copyright © 2022.All Rights Reserved.
傲软投屏

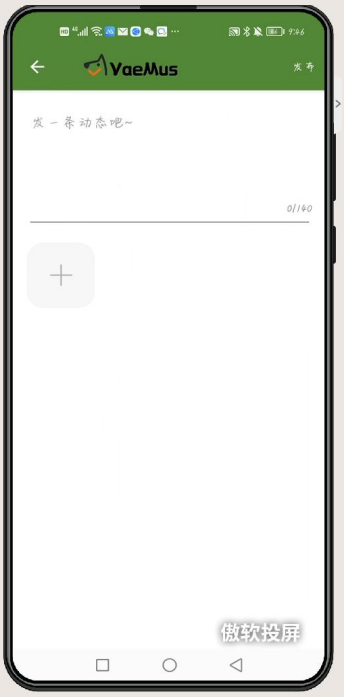

听一首歌 放空你的思绪
是否允许 "VaeMusic" 拍摄照片和录制视频?
禁止
仅使用期间允许
傲软投屏


VaeMus
不语
许嵩-专辑
九月清晨
许嵩-专辑
今年勇
许嵩-专辑
伴虎
许嵩-专辑
你若成风
许嵩-专辑
聊吧
圈子
傲软投屏

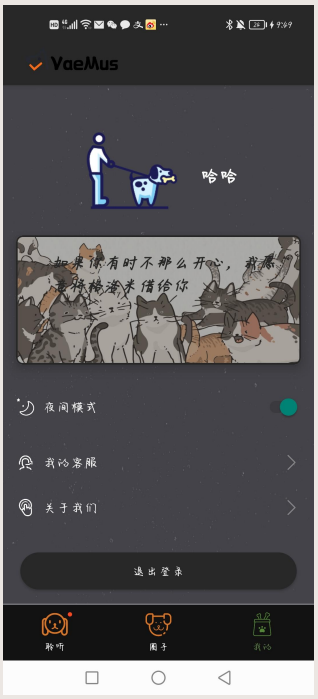

庐州月
许嵩
庐州月
许嵩
00:04
04:15
傲软投屏


不语
许嵩
列表循环(37)
不语 - 许嵩
九月清晨 - 许嵩
今年勇 - 许嵩
伴虎 - 许嵩
你若成风 - 许嵩
内伤 - 许嵩
冰轮 - 许嵩
千百度 - 许嵩
傲软投屏


VaeMus
聊圈
今天沁天真好看啊!
聊圈
特别喜欢许嵩沁歌
尤其是那一首《降温》
聊吧
圈子
傲软投屏


VaeMus
发布
发一条动态吧~
0/140
做软投屏


VaeMus
赵哈瑜
如果你有时不那么开心，我愿意将快活来借给你
夜间模式
我的客服
关于我们
退出登录
聊听
圈子
我的
做软投屏


VaeMus
哈哈
如果你有时不那么开心，我愿意将快活来借给你
夜间模式
我的客服
关于我们
退出登录
聊听
圈子
我的


VaeMus
小四
我好，大家好
1分钟前
我的鞋子去哪了
你好
聊听
圈子
我的


VaeMus
请输入您的手机号
请输入您的密码
登录
立即注册
忘记密码
第三方登录
做软投屏


VaeMus
昵称(2~10位)
0/10
请输入您的手机号
0/11
密码(6~15位)
0/15
确认密码
0/15
注册并登陆


赵晗瑜
圈子
我的
赵晗瑜
小猫长的有点像小狗
我的鞋子去哪了
今晚又做了一个梦
我是谁
我是谁
我是谁
那天我，被张佩服用石头砸了一下头
完事醒了我就忘了我是谁
我变成了一个四岁的小孩
回家后我只认得那盛饭的碗
路人说我是脱疆的野马儿
孩子说我是会飞的鸟儿
妈妈说我是不会枯萎的花儿
做软投屏
赵晗瑜
圈子
我的
个人信息
昵称：赵晗瑜
性别：保密
生日：2002-4-11
我的个性签名
大家好我是Vae，这是我即将发表的收藏独创专辑《自定义》里面的一首雅若曲目，词曲编曲都是我自己，希望这首歌曲，能在这个寒冷的冬天，带给大家一种温暖的感觉~
做软投屏
199 3315 3268
1
2
3
4
5
6
7
8
9
*
0
#
pretty-zhy.club
ECNU-zhy
你可知脆弱是一种勇敢 _
A new journey to handle the variety of data论文阅读笔记
以下是我在阅读论文的过程中画的思维导图,放大可查看详情: A New Journey to Handle the Variety of Data...
2022-12-07
非关系数据库
#非关系数据库 #论文阅读

六、总结和展望

通过此次安卓开发项目的实践，小组成员通力合作，初步开发了一个较为完整的小规模音乐播放及社交 APP。在设计和实现过程中曾遇到一些困难，如无法访问服务端静态资源、客户端为网络请求添加请求头、如何实现高斯模糊的效果、如何实现下拉刷新和上拉加载更多、如何设计一个管理器来管理音乐播放列表和音乐播放，这些问题我们在设计和实现的过程中都遇到了大大小小的问题，我们通过查阅相关文档和相关开源代码，找到了较多第三方框架帮助我们解决问题，但是利用这些第三方框架也是一个蛮难的问题，需要我们仔细阅读 readme 文档，必要时可能还会去查看源代码。我们越来越感觉到安卓开发是一个宽泛而又精细的主题，它涉及到的东西太多了，然而它在所有方面又做得很精细。

进行安卓开发和我之前利用前端 APP 框架的感受大不相同，首先感觉到的一定是语法的复杂，从标签语言到面向对象语言的大幅度转变也让我度过了很长一段时间的适应期；其次我发现安卓开发相对于一些框架开发的灵活性很大，性能很好，但是却增加了开发成本。

由于技术水平尚不成熟，时间也较为紧迫，该软件还存在些许不足。例如，在敏捷开发的过程中我们只考虑到当下的简单实现，并未为未来迭代留有足够的余地。例如没有考虑到未来项目升级迭代时，若项目规模扩大，如何整合海量曲库并进行标签分类，是否要搭建不同歌手的歌迷“圈子”？只能在线播放音乐是否有些苛刻，是否可以借鉴现有软件实现在线-缓存双播放？

但是我们想到了一条新的发展方向，市面上有一款叫做“Vae+”（市面上歌手许嵩的官方 APP），本平台实现的功能全面，能够满足用户的多方面需求，我们做的这个 APP 未来的发展方向或将朝向“Vae+”APP 看齐，提供更加多样化的功能，如：

1、汇集包括新闻、专访、图集、视频在内的所有许嵩官方新闻资讯，并置顶许嵩发帖，使用户轻松查看；

2、发布许嵩行程并提醒用户相关活动的时间安排和地点安排；

3、设计更加多样化的社交功能，如分频道/分话题讨论互动；

7、设计许嵩的卡通形象与用户进行唱歌等交互；

8、商城板块用于出售许嵩官方专辑、周边等官方正版实物商品。

但总体来说，在开发该项目的过程中，我们几乎练习并熟练掌握了移动应用开发课程中所讲的大部分中高阶知识点，通过浏览 Github 等开源网站引入了更新、更安全的组件。

安卓开发是一个既考验技术水平又考验审美布局能力的过程，日后我们也会继续努力，合作开发出更加精品的项目。

七、附录

1. 参考文档：

1) Springboot 项目部署 https://www.jb51.net/article/216616.htm
2) CentOS7 安装 Nginx https://www.likecs.com/show-522774.html
3) MySQL 远程连接解决方案
   https://blog.csdn.net/qq_39162487/article/details/121456746

2. 源码地址：

1) 服务器地址：http://119.91.198.46:9000/
2) 客户端源码地址：https://github.com/hot-zhy/cloud-music
3) 服务端源码地址：https://github.com/Accepted1226/cloud-music-server
4) apk 下载地址：https://github.com/hot-zhy/VaeMusic