学号：201910121213

**移动互联网开发课程设计**

题 目：一种基于Android的音乐播放器的设计与开发

班 级： 网络191

姓 名： 徐艺芸

指导老师： 刘翌昊

完成日期： 2022年6月1日

上海海事大学信息工程学院

**2022 年 6 月**

**目录**

[一、设计题目 3](#_Toc104994330)

[二、设计目的 3](#_Toc104994331)

[三、设计原理和方案 5](#_Toc104994332)

[1 Android的系统架构 5](#_Toc104994333)

[2 使用的软件工具和环境 8](#_Toc104994334)

[3 需求分析与概要设计 8](#_Toc104994335)

[3.1用户分析 8](#_Toc104994336)

[3.2 设计目标 9](#_Toc104994337)

[3.3 流程图 10](#_Toc104994338)

[3.4目录结构 12](#_Toc104994339)

[4 数据库设计 13](#_Toc104994340)

[5 详细设计和关键问题 14](#_Toc104994341)

[四、页面布局设计 23](#_Toc104994342)

[五、总结 26](#_Toc104994343)

[5.1 创新点 26](#_Toc104994344)

[5.2 总结与反思 27](#_Toc104994345)

[**参考文献** 28](#_Toc104994346)

# 一、设计题目

基于Android的音乐播放器设计，使用Android studio作为开发软件,使用的主要编程语言是Java。

# 二、设计目的

随着音乐的普及，音乐成为了现代人们生活的必需品，它能够缓解人们焦躁的心情，调节现代化生活的节奏，让人们舒缓身心。一个好的音乐播放器，应该能够降低用户获取想要听的音乐的时间成本，快速让用户找到自己想要听的歌曲，并且能够让用户更加轻松掌控使用音乐，达到舒缓压力的需求。

再来随着人们物质生活水平的提高，人们越来越注重精神生活的满足，热衷于在消费中寻求快乐和娱乐体验。音乐作为人类娱乐生活的重要组成元素，一直以来都备受关注。而人类进入工业社会以来，将音乐播放与工业产品结合发展出了一系列的音乐播放产品，并随着技术的革新和消费者需求的变化而不断更新，为人类的娱乐生活提供了时尚便利的道具。

特别是市场呈现红海，满足“喜好者”的需要是很常见的一些功能，音乐播放器、音乐曲库、音乐下载、音乐铃声制作等。这些基本功能的技术门槛并不 高，于是，成败的关键问题就变成了软件的用户体验感是否达到要求。

然而市面上的大多数音乐播放器为了引流，增加平台收益，添加越来越多的大多数对用户帮助不大的功能，如收听电台，开放音乐广场社区等，事实上这些功能虽然能够满足部分用户的需求，但对于那些只是想简单收听音乐的用户们成为负担，为了收听音乐不得不接受额外的信息冲击，不仅占用手机内存，让听歌的打开方式变得越来越复杂，还本末导致，降低了用户听音乐时对音乐的掌控感和仪式感。而且不同平台也存在相互竞争，通过争夺音乐版权，垄断市场，对手机的本地音乐层层加码，原本存在在用户手机的音乐很难被用户找到，或者找到了也无法使用该平台进行播放，增加其用户的使用成本，由于音乐app的权限设置，让识别手机中本地音乐并非其考虑的重点，大多数用户难以发现和找到自己手机本地上本该存在的音频文件，使得自己在手机上录取的音频和曾经下载过的音乐很难被发现和利用。

为此，针对这一现象笔者也对市场进行相关调查研究，发现大多数用户对简单界面清爽的音乐播放器有需求，这也充分说明了本设计的现实意义和可用性。基于这样的研究背景和存在问题，设计一款用户体验度高，功能完善，界面优化与高交互性的音乐播放器具有重大的价值与现实意义。

笔者便考虑设计这样一款轻量级音乐app ,能够快速检索到本地的音乐并清晰的呈现给用户，通过设计更加简洁优雅的UI界面，让用户可以通过我们的软件轻松操纵音乐，增加仪式感，考虑到用户对个性化的需求，我们将设计一个更改播放列表背景一功能，让用户打开本app 能够在收听自己喜爱音乐的同时还能够欣赏自己喜爱的图片，进一步增加用户对音乐的掌控感，进一步辅助舒缓用户心情，达到解压的目的。

# 三、设计原理和方案

## 1 Android的系统架构

Android的系统架构和其操作系统一样，采用了分层的架构。从图3.1中可以看出，Android分为四个层，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和Linux核心层。

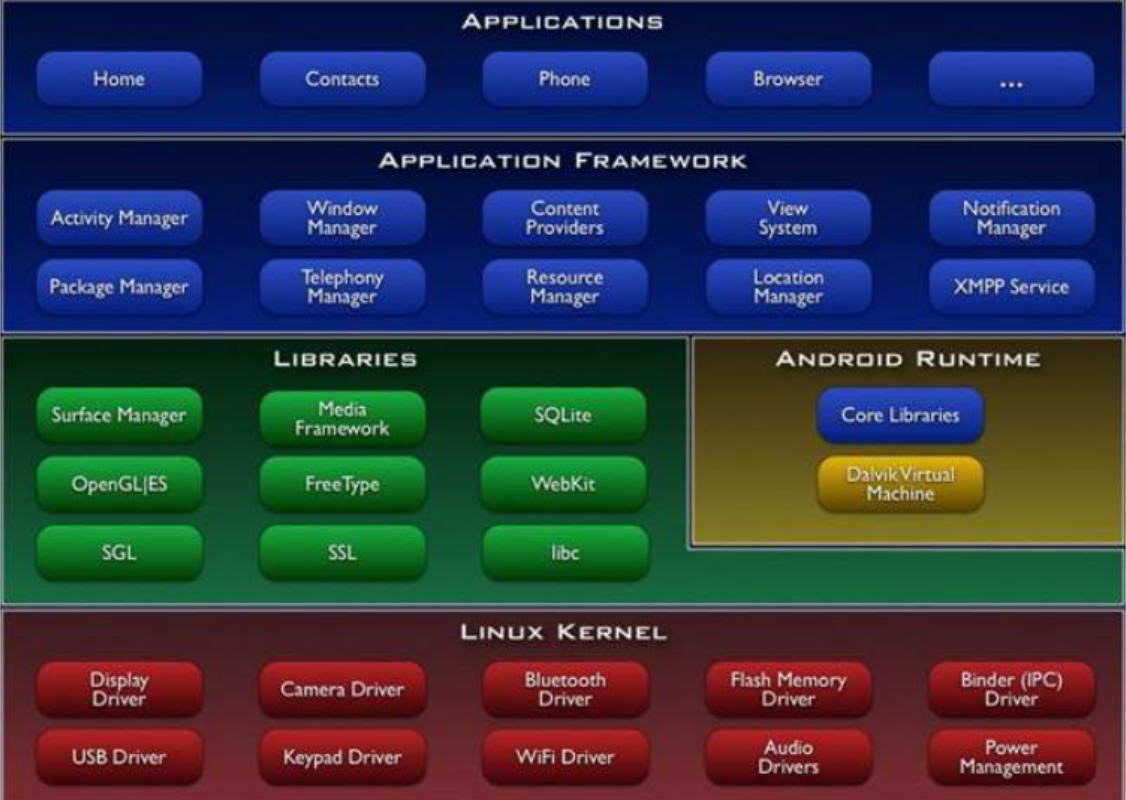


图3.1 Android的系统架构图

（1）应用程序

Android会同一系列核心应用程序包一起发布，该应用程序包包括Email客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用JAVA语言编写的。

（2）应用程序框架

开发人员也可以完全访问核心应用程序所使用的API框架。该应用程序的架构设计简化了组件的重用；任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块（不过得遵循框架的安全性限制）。同样，该应用程序重用机制也使用户可以方便的替换程序组件。

隐藏在每个应用后面的是一系列的服务和系统，其中包括：

丰富而又可扩展的视图（Views）：可以用来构建应用程序，它包括列表1ists），网格(grids),文本框(text boxes),按钮(buttons),甚至可嵌入的web浏览器。

内容提供器（Content Providers）：使得应用程序可以访问另一个应用程序的数据（如联系人数据库），或者共享它们自己的数据。

资源管理器（Resource Manager）：提供非代码资源的访问，如本地字符串，图形，和布局文件(layout files)。

通知管理器（Notification Manager）：使得应用程序可以在状态栏中显示自定义的提示信息。

活动管理器（Activity Manager）：用来管理应用程序生命周期并提供常用的导航回退功能。

（3）系统运行库

1）程序库

Android包含一些C/C++库，这些库能被Android系统中不同的组件使用。它们通过Android应用程序框架为开发者提供服务。以下是一些核心库：

系统C库：一个从BSD继承来的标准C系统函数库（1ibc），它是专门为基于embedded 1inux的设备定制的。

媒体库：基于PacketVideo OpenCORE，该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括MPEG4，H.264,MP3,AAC,AMR,JPG，PNG。

Surface Manager：对显示子系统的管理，并且为多个应用程序提供了2D和3D图层的无缝融合。

LibWebCore：一个最新web浏览器引篫用,支持Android浏览器和一个可嵌入的web视图。

SGL：底层的2D图形引擎。

3D 1ibraries：基于OpenGL ES 1.0APIs实现：该库可以使用硬件3D加速（如果可用）或者使用高度优化的3D软加速。

FreeType：位图（bitmap)和矢量（vector)字体显示。

SQLite：一个对于所有应用程序可用，功能强劲的轻型关系型数据库引擎。

2)Android 运行库

Android包括了一个核心库，该核心库提供了JAVA编程语言核心库的大多数功能。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。Dalvik被设计成一个设备可以同时高效地运行多个虚拟系统。Dalvik虚拟机执行（.dex）的Dalvik可执行文件，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有的类都经由JAVA编译器编译，然后通过SDK的"dx"工具转化成.dex格式由虚拟机执行。

Dalvik虚拟机依赖于Linux内核的一些功能，比如线程机制和底层内存管理机制。

3)Linux内核

Android的核心系统服务依赖于Linux2.6内核，如安全性，内存管理，进程管理，网络协议栈和驱动模型。Linux内核也同时作为硬件和软件栈之间的抽象层。

## 2 使用的软件工具和环境

本项目使用的开发工具是Android Studio ，它是谷歌推出的一个Android集成开发工具，基于[IntelliJ IDEA](https://baike.baidu.com/item/IntelliJ%20IDEA). 类似 [Eclipse](https://baike.baidu.com/item/Eclipse/61703) [ADT](https://baike.baidu.com/item/ADT)，Android Studio 提供了集成的 Android 开发工具用于开发和调试。该软件在JetBrains Intellij [IDEA](https://baike.baidu.com/item/IDEA/1671803)的基础上，提供：

* 基于[Gradle](https://baike.baidu.com/item/Gradle)的构建支持
* Android 专属的重构和快速修复
* 提示工具以捕获性能、可用性、版本兼容性等问题
* 支持[ProGuard](https://baike.baidu.com/item/ProGuard) 和应用签名
* 基于模板的向导来生成常用的 Android 应用设计和组件
* 功能强大的布局编辑器，可以让你拖拉 UI 控件并进行效果预览

Android Studio软件与市面上的其他开发工具相比提供的工具更多，功能更加强大，且开源，基于这样的优势便选此作为项目的开发工具。

## 3 需求分析与概要设计

### 3.1用户分析

首先，我们对使用软件的群体进行了分析，可划分为5类人，但因为社会环境、音乐素养和音乐市场的差异，各类人群在不同国家的占比有些差异，这也就导致同一类型的音乐播放软件，在国内外的境遇不同。详细说明请见下表3.3.3.1

|  |  |  |  |
| --- | --- | --- | --- |
| 大类 | 说明 | 国际市场 | 中国大陆市场 |
| 喜好者 | 喜好音乐,把音乐作为自己生活和工作的背景和陪伴,倾向于音乐播放器的功能 | 大众需求 | 大众需求 |
| 痴迷者 | 痴迷于某一类或几类细分类型音乐、或是某个歌星乐队,追求音乐播放器的极致音乐品质。 | 小众需求 | 小众需求,但未被 很好的满足 |
| 娱乐者 | 将音乐作为娱乐方式,比如卡拉OK、各类音乐游戏，热衷于播放器的效果与界面设计 | 大众需求 | 大众需求,但未被 很好的满足 |
| 玩票者 | 业余爱好者中的专业音乐人,讲究专业水准，对歌曲添加、收藏、导出等相关便捷程度要求高 | 大众需求 | 人数很少,还未到 达小众需求 |
| 专业者 | 专业音乐工作者，需要查看歌曲的频谱等内容 | \ | \ |

表3.3.1 用户分析表

分析了上述使用群体后，可以从音乐播放器的使用者的角度，概括出用户的需求主要有以下5类：

* **播放歌曲**
* **导入歌曲**
* **歌词展示**
* **歌曲收藏**
* **模式切换**

### 3.2 设计目标

基于本次设计的初衷，为用户提供更加简单纯粹的音乐播放器，以及结合用户的需求，本系统设计最终决定的设计目标如下：

1. 界面简洁追求精美,UI完善且交互性高。
2. 能够快速检索本地音乐并加载到app的页面中、并且能够自动扫描系统文件夹的中的音乐。
3. 支持本地播放的音乐格式有：.mp3 文件、.wav 文件、.aac 文件。
4. 能够支持解析音乐的相关属性（专辑图片、时长、歌曲名称等）。
5. 支持歌单列表控制，音乐支持后台播放，每一首播放的音乐都有独立的界面进行控制。
6. 支持用户登录与注册功能，支持用户信息填写。
7. 个性化用户选择，用户可以根据个人喜好，自定义头像。
8. 播放的歌曲可以拉取进度条快进回退，灵活切换下一首，上一首。

### 3.3 流程图

本次设计的流程图如图3.3.1所示

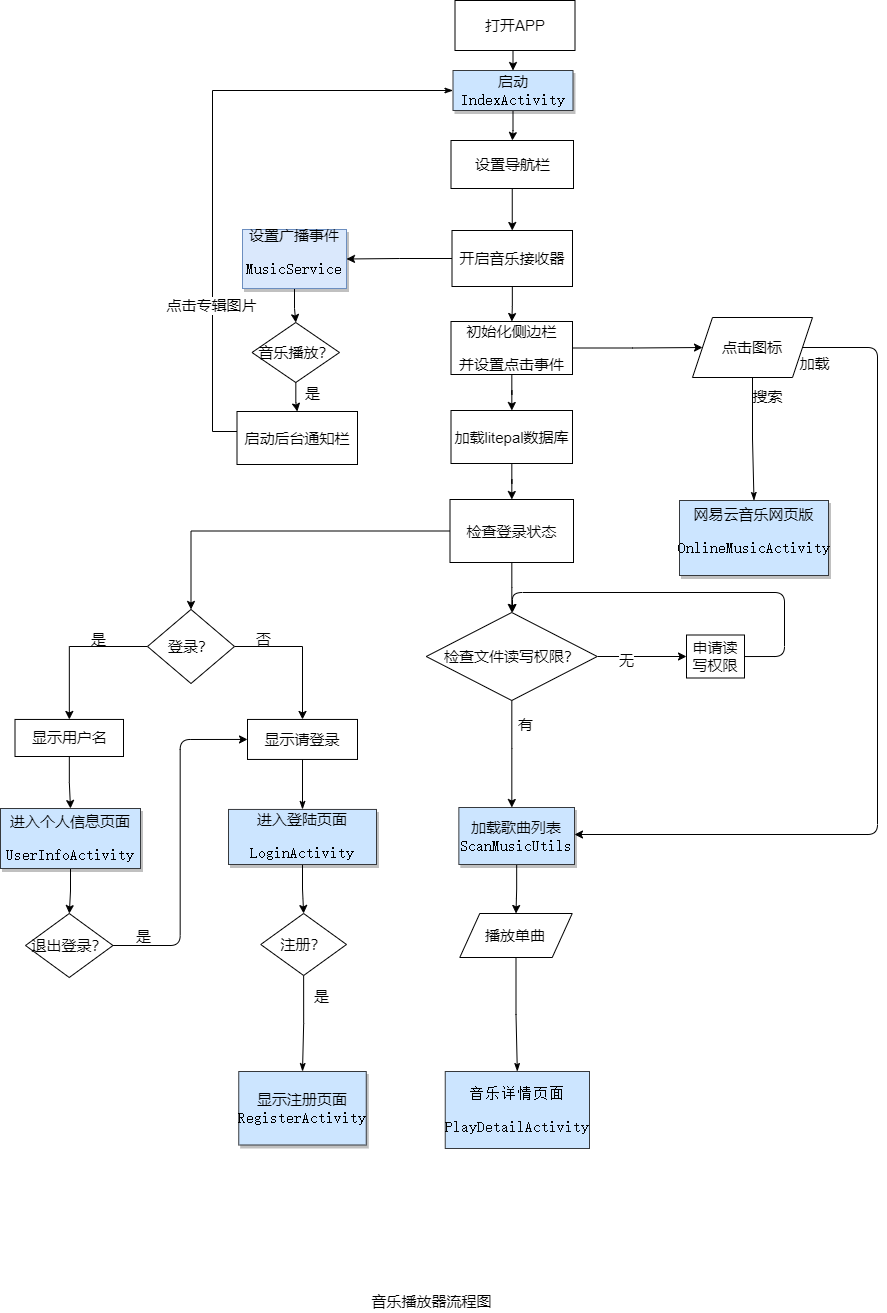


图3.3.1 音乐播放器实现流程图

### 3.4目录结构

1）源代码目录结构如图3.3.2 目录结构图

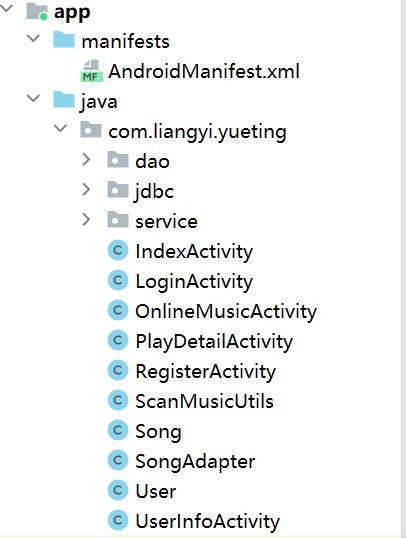
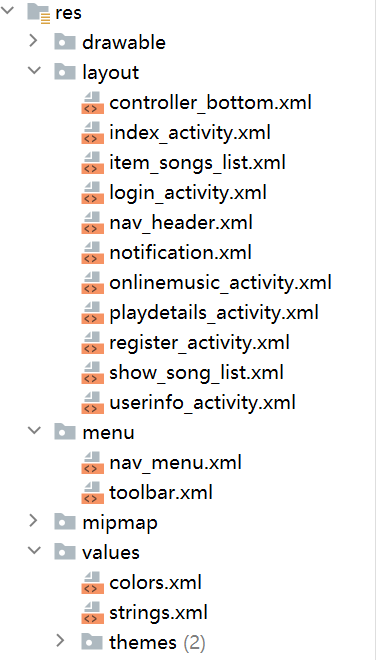


图3.3.2 目录结构图

* 1. java文件用途说明请见下表

|  |  |  |
| --- | --- | --- |
| 包名称 | 文件名 | 说明 |
| package com.liangyi.yueting | indexActivity | App首页，本地音乐列表的Activity |
| LoginActivity | 登录页面的Activity |
| OnlineActivity | 网页云音乐网页的Activity |
| PlayDetailActivity | 音乐播放详情页面的Activity |
| RegisterActivity | 注册页面的Activity |
| ScanMusicUtils | 本地音乐扫描获取模块 |
| Song | 音乐属性定义模块 |
| SongAdapter | 音乐适配器模块 |
| User | 用户属性定义模块 |
| UserInfoActivity | 用户信息页面的Activity |
| package com.liangyi.yueting.service | MusicService | 后台服务，让播放的音乐在通知栏显示 |

表3.3.1 java文件用途表

3）资源文件名称与用途请见下表

|  |  |  |
| --- | --- | --- |
| 资源目录 | 文件 | 说明 |
| Layout | Controller\_botton.xml | 音乐播放底部控制栏布局 |
| Index\_activity.xml | App首页页面布局 |
| Item\_songs\_list.xml | 单项歌曲控制布局 |
| Login\_activity.xml | 登录页面布局 |
| Nav\_header.xml | 主页侧边栏布局 |
| Notification.xml | 通知栏布局 |
| Onlinemusic\_activity.xml | 在线音乐页面布局 |
| Playdetails\_activity.xml | 播放详情页面布局 |
| Register\_activity.xml | 注册页面布局 |
| Show\_song\_list.xml | 歌曲列表布局 |
| Userinfo\_activity.xml | 用户信息页面布局 |
| Menu | Nav\_menu.xml | 导航菜单布局 |
| Toolbar.xml | 工具栏布局 |
| Values | colors.xml | 颜色资源 |
| strings.xml | 字符串资源 |
| themes.xml | 项目主题资源 |

表3.3.2 资源文件名称与用途表

## 4 数据库设计

本项目使用的数据库是LitePal，LitePal是一款开源的Android数据库框架，采用了[对象关系映射](https://baike.baidu.com/item/%E5%AF%B9%E8%B1%A1%E5%85%B3%E7%B3%BB%E6%98%A0%E5%B0%84/311152)(ORM)的模式，将平时开发时最常用的一些数据库功能进行了封装，使得开发者不用编写一行SQL语句就可以完成各种建表、増删改查的操作。

1）实现登录注册功能需要实现数据库的搭建，以便管理用户的信息，以下便是管理用户信息设计的表，如表3.4.1所示：

|  |  |  |
| --- | --- | --- |
| 属性 | 数据类型 | 说明 |
| userName | String | 用户姓名 |
| userNumber | String | 用户ID |
| userPassword | String | 用户密码 |
| userSex | String | 用户性别 |
| userStatus | String | 用户状态（登录为1，未登录为0） |
| userEmail | String | 用户邮箱 |

表3.4.1 用户数据表

管理歌曲内容所需要的表，如表4.2 所示

|  |  |  |
| --- | --- | --- |
| 属性 | 数据类型 | 说明 |
| name | String | 歌曲名 |
| singer | String | 歌手 |
| size | Long | 歌曲所占空间大小 |
| duration | String | 歌曲时间长度 |
| path | String | 歌曲地址 |
| albumId | long | 专辑图片ID |
| album | String | 专辑名称 |

表3.4.2歌曲信息表

## 详细设计和关键问题

#### 1）注册音乐服务管理

为了让软件在关闭播放器界面后，仍能够保持音乐持续播放，需要用到Service服务组件，它不用直接与用户交互，长期在后台运行，本项目在Java文件中创建MusicService类，它继承于service，通过该服务完成音乐在后台的播放，还能够用户在退出音乐app界面时，让手机的通知栏显示音乐播放状态栏。

首先要在AndroidManifest.xml文件注册服务

|  |
| --- |
| <service  android:name=".service.MusicService"  android:enabled="true"  android:exported="true" /> |

然后新建一个MusicService类

|  |
| --- |
| public class MusicService extends Service {…} |

#### 2）后台通知栏控制音乐播放

让手机通知栏显示后台播放的音乐，主要引用android.app.NotificationManager包，获取NotificationManager的实例需要调用Context的getSystemService()方法，getSystemService()方法接收一个字符串参数用于确定获取系统那一个服务，本项目在MusicService中由initNotificationBar（）来实现：

|  |
| --- |
| private void initNotificationBar(){  *//检查当前Android版本，8.0以上需设置通知渠道* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*);  CharSequence name = "通知栏播放";  String description = "notification description";  int importance = NotificationManager.*IMPORTANCE\_MIN*;  NotificationChannel mChannel = new NotificationChannel(notificationChannelId, name, importance);  mChannel.setDescription(description);  mChannel.setLightColor(Color.*RED*);  mChannel.setVibrationPattern(new long[]{100, 200, 300, 400, 500, 400, 300, 200, 400});  mNotificationManager.createNotificationChannel(mChannel);  }  mBuilder=new NotificationCompat.Builder(this,notificationChannelId);  *//获取当前歌曲信息* Log.*d*("通知栏播放页init", "updateNotification: 更新状态栏歌曲新息");  String songName=songlist.get(MusicService.*mPosition*).getName();  String songSinger=songlist.get(MusicService.*mPosition*).getSinger();  *//获取通知栏歌曲控制器* remoteView=new RemoteViews(getPackageName(),R.layout.*notification*);  remoteView.setTextViewText(R.id.*notification\_title*,songName+" - "+songSinger);  loadingCover(songlist.get(MusicService.*mPosition*).getPath());  */////控制按钮   //设置通知栏样式* mBuilder.setWhen(System.*currentTimeMillis*())  .setContentTitle("PongPong~")  .setContent(remoteView)  .setSmallIcon(R.drawable.*icon\_music*)  .setLargeIcon(BitmapFactory.*decodeResource*(getResources(),R.drawable.*icon\_music*));  notification=mBuilder.build();  notification.flags = Notification.*FLAG\_ONGOING\_EVENT*;*//设置通知点击或滑动时不被清除* NotificationManager manager=(NotificationManager)getSystemService(Service.*NOTIFICATION\_SERVICE*);  manager.notify(notifyId,notification);  updateNotification();  } |

通过android.os.Ibinder包，实现接口Ibinder的onbind()方法，使得开启服务的类能够操纵和管理服务。

|  |
| --- |
| public IBinder onBind(Intent intent) {  *// TODO: Return the communication channel to the service.* return new musicBinder(); } |

#### 3）实现操纵音乐的方法

|  |
| --- |
| public class musicBinder extends Binder{  public boolean isPlaying(){  return player.isPlaying();*//判断当前歌曲是否正在播放* }  public void play(){  if(player.isPlaying())  player.pause();  else{  player.start();  }  }  *//播放下一首歌* public void next(int type){  *mPosition*+=type;  *mPosition*=(*mPosition*+songlist.size())%songlist.size();  song=songlist.get(*mPosition*);  prepare();*//切换音乐的具体实现*  }  *//Returns the length of the music in milliseconds* public int getDuration(){…}  public int getPosition(){return *mPosition*;}  *//Return the name of the music* public String getName(){…}  public String getPath(){…}  public String getSinger(){…}  *//Returns the current progress of the music in milliseconds* public int getCurrenPostion(){…}  public long getAlbumId(){…}  *//Set the progress of music playback in milliseconds* public void seekTo(int mesc){…} } |

#### 4）启动页面的设计

设置菜单

|  |
| --- |
| public boolean onCreateOptionsMenu(Menu menu) {getMenuInflater().inflate(R.menu.*toolbar*,menu);  return true; } |

Oncreate()函数实现如下,思路就是按照流程图的思路一步步实现所需要的方法，分别调用以下功能：设置导航栏，开启音乐广播接收器，初始化侧边栏，加载litepal数据库、检查用户是否登录、检查文件读写权限并加载歌曲列表。

|  |
| --- |
| protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);*//保存现场* setContentView(R.layout.*index\_activity*);*//将index\_activity设置为主页* Toolbar toolbar=findViewById(R.id.*toolbar*);  setSupportActionBar(toolbar);*//设置导航栏* drawerLayout=findViewById(R.id.*drawer\_layout*);  ActionBar actionBar=getSupportActionBar();  if(actionBar!=null){  actionBar.setDisplayHomeAsUpEnabled(true);  actionBar.setHomeAsUpIndicator(R.drawable.*ic\_menu*);  }  mReceiver = new MusicReceiver(new Handler());*//开启音乐接收器* IntentFilter itFilter = new IntentFilter();  itFilter.addAction(MusicService.*MAIN\_UPDATE\_UI*);  registerReceiver(mReceiver, itFilter);*//注册音乐接收器  //初始化侧边栏，并设置点击事件* navigationView=(NavigationView)findViewById(R.id.*nav\_view*);  initNavMenu();  *//加载litepal数据库  /\*=================litepal数据库=====================\*/* LitePal.*initialize*(this);  db = LitePal.*getDatabase*();*//获取数据库实例  //检查登录状态* checkUserLogin();*//检查用户是否登录* checkPermission();*//检查文件读写权限并加载歌曲列表* } |

#### 5）实现更改用户的头像功能

实现更改用户的头像具体思路是，是用户在点击更改头像选项时,使用intent,它是一种消息传递机制，用于组件之间数据交换和发送广播信息，在本项目中通过它调用获取手机系统文件的接口，从而使得用户可以通过点击事件打开手机系统图像。

|  |
| --- |
| Intent intent=new Intent("android.intent.action.GET\_CONTENT");  *//隐式调用，调用手机文件* intent.setType("image/\*"); startActivityForResult(intent,*CHOOSE\_PHOTO*);*//打开相册* |

#### 6）实现获取本地所有音乐方法

获取本地所有音乐主要是实现一个ScanMusicUtils类中getMusicData（）方法，该方法的返回类型是List<Song>即自定义的歌曲信息列表结构，由song.Java文件实现，通过调用context.getContentResolver().query（）函数，获取设备中的所有歌曲信息。

|  |
| --- |
| *// 媒体库查询语句（写一个工具类MusicUtils）* Cursor cursor = context.getContentResolver().query(  MediaStore.Audio.Media.*EXTERNAL\_CONTENT\_URI*, null, null,  null, MediaStore.Audio.AudioColumns.*IS\_MUSIC* ); |

#### 7）加载所用音乐到主页上

将获取的所有音乐加载在主页上使用RecyclerView组件，它高效替代了最初的ListView等列表组件。使用上实现了视图代码解耦，功能上具备强大的item复用机制，并且提供默认多种LayoutMananger用来处理多种布局，下面它在本项目的具体应用：

在布局index\_activity.xml中

|  |
| --- |
| <androidx.recyclerview.widget.RecyclerView  android:id="@+id/mRecyclerView"  android:layout\_width="match\_parent"  android:layout\_marginBottom="30dp"  android:layout\_height="match\_parent" /> |

在布局show\_song\_list.xml中

|  |
| --- |
| <LinearLayout  android:id="@+id/linearRecyclerView"  android:layout\_width="match\_parent"  android:layout\_height="0dp"  android:layout\_marginBottom="30dp"  android:layout\_weight="1"  android:orientation="vertical">   <androidx.recyclerview.widget.RecyclerView  android:id="@+id/mRecyclerView"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" />  </LinearLayout> |

SongAdapter适配器

|  |
| --- |
| public class SongAdapter extends RecyclerView.Adapter<SongAdapter.ViewHolder> {…} |

在IndexActivity添加代码

|  |
| --- |
| songList=ScanMusicUtils.*getMusicData*(IndexActivity.this);*//初始化获取歌曲信息* SongAdapter songAdapter=new SongAdapter(songList);*//歌曲适配器* RecyclerView recyclerView=(RecyclerView)findViewById(R.id.*mRecyclerView*); layoutManager=new LinearLayoutManager(IndexActivity.this); recyclerView.setLayoutManager(layoutManager); recyclerView.setAdapter(songAdapter);*//加载歌曲列表* |

#### 8）管理音乐数据和用户数据

管理音乐数据结构，需单独创建Song类，即音乐信息类，下面列出主要的数据类型成员有：

|  |
| --- |
| public class Song implements Parcelable {  public String name;*//歌曲名* public String singer;*//歌手* public long size;*//歌曲所占空间大小* public int duration;*//歌曲时间长度* public String path;*//歌曲地址* public long albumId;*//图片id* public String album;*//专辑名称*  …..  } |

管理用户信息，需单独创建User类，即用户信息类，主要的数据成员如下：

|  |
| --- |
| public class User extends LitePalSupport {  private String userName;  private String userNumber;  private String userPassword;  private String userSex;  private String userStatus;  private String userEmail;  public User()  ……  } |

#### 9）实现登录注册功能

本项目的另一亮点是实现了登录注册功能，实现此功能主要思路是：用户信息类继承LitePal Android数据库[框架](https://so.csdn.net/so/search?q=%E6%A1%86%E6%9E%B6&spm=1001.2101.3001.7020)，使用该框架的好处是它对一些数据库功能进行了封装，使得开发者不用编写一行SQL语句就可以完成各种建表、増删改查的操作。并且LitePal很“轻”，jar包大小不到100k，符合开发需求，在用户登录时，输入的用户名和密码查询验证如下：

|  |
| --- |
| *//根据手机号查找用户* List<User> findUser= LitePal.*where*("userNumber=?",number)  .find(User.class); |

可以看出使用LitePal非常的简洁和高效。

#### 10）实现播放音乐详情页面

点击音乐列表中的音乐或者点击音乐播放状态栏的专辑头像亦或是点击通知栏的专辑头像都会进入播放音乐详情页面，在这个页面中实现对单曲的控制，如暂停播放，启动播放，下一曲，上一曲，进度条拉取

实现音乐详情加载页面功能在项目中是PlayDetailActivity，重写的oncreate()方法中可以看出其页面加载时需要具体实现的方法函数：

|  |
| --- |
| @Override protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*playdetails\_activity*);  songList=ScanMusicUtils.*getMusicData*(this);  *//显示启动音乐服务* Intent intent=new Intent(this,MusicService.class);  Bundle bundle=getIntent().getExtras();  intent.putExtras(bundle);  *//连接音乐服务* conn=new MusicConnection();  *//启动服务* startService(intent);  Log.*d*("进入播放页", "onCreate: Intentd传来的position"+bundle.getInt("position"));  *//绑定服务* bindService(intent,conn,*BIND\_AUTO\_CREATE*);  mReceiver=new MusicReceiver(new Handler());  IntentFilter intentFilter=new IntentFilter();  intentFilter.addAction(MusicService.*MAIN\_UPDATE\_UI*);  getApplicationContext().registerReceiver(mReceiver,intentFilter);  bindView(); } |

# 四、页面布局设计

本项目为了让用户有更好的使用体验，页面布局主题基调为静谧蓝，其用意是希望用户在打开app,心情能够平静，沉浸式享受音乐，舒缓身心压力，UI设计追求简洁，没有多余的花里胡哨的设计。

本项目根据需求设计了11个页面布局，两个菜单布局。并使用华为荣耀20手机进行软件测试，部分布局显示效果如下：

1. app首页，加载所有本地音乐

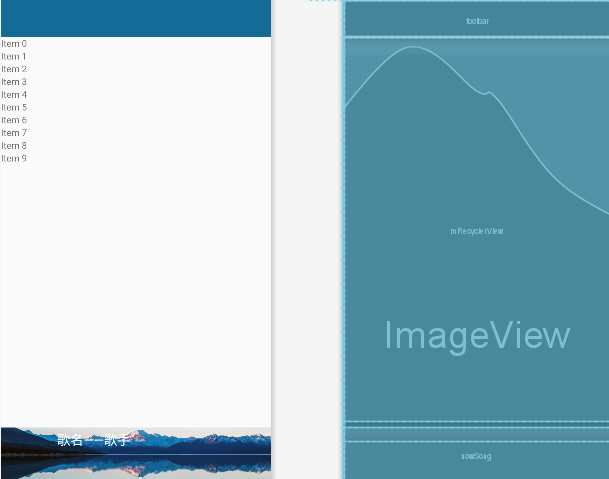
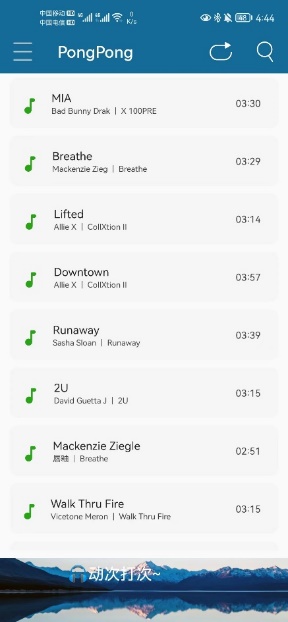
 

图4.1 index\_activity.xml布局效果

1. 音乐详情页面，操纵单曲

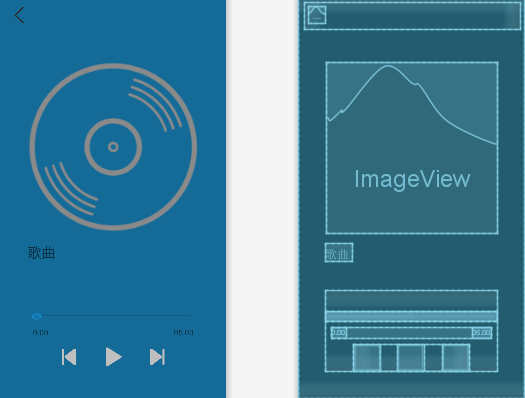
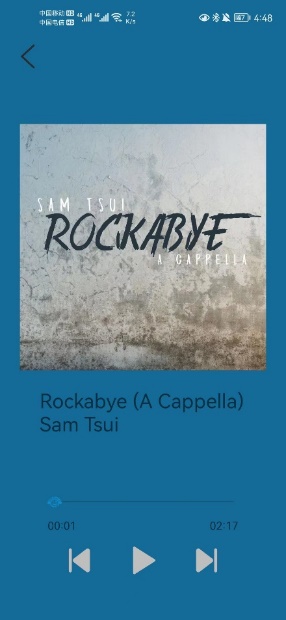
 

图4.2 playdetails\_activity.xml布局效果

1. 导航栏，显示个人信息，更改用户头像

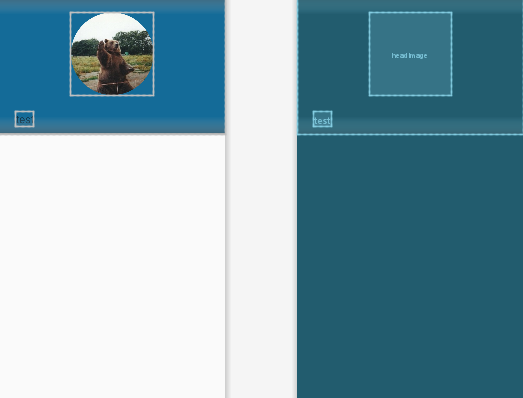
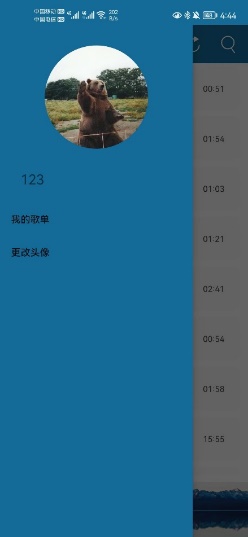
 

图4.3 nav\_header.xml布局效果

1. 通知栏设计，显示后台播放音乐，通过该页面进入app.

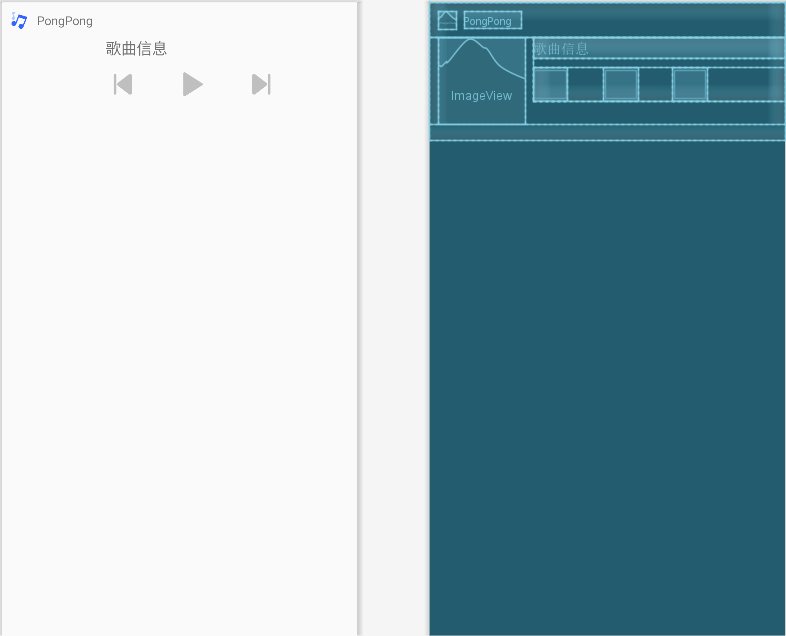
 **

图4.4 notification.xml布局效果

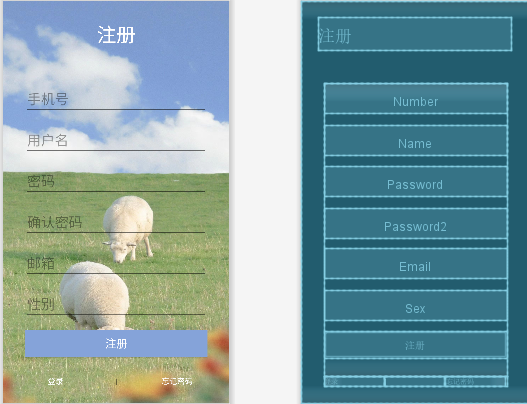
5)登录、注册页面，显示用户信息或者更改用户信息 

图4.5 register\_activity.xml布局效果

图4.6 login\_activity.xml布局效果

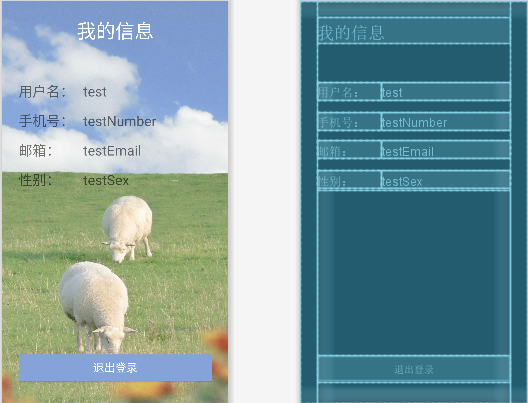
 

图4.7 userinfo\_activity.xml布局效果

# 五、总结

## 5.1 创新点

本项目是基于Android开发音乐app,相比于传统的音乐播放器而言，更加轻量，只有不到10M，却能实现登录注册功能，加载本地音乐，更换用户头像，自由切换单曲等功能，满足了绝大多数用户的需求，数据库采用LitePal，使得实现数据库查询时，不需要用sql语句，让程序的运行更加轻量和快速，页面布局设计精美，优雅，能给用户更好的使用体验，继承了传统音乐软件的功能优势，可替代性弱；对于播放的音乐，用户手机还可以在通知栏显示正在播放的音乐，也可以通过通知栏进入app，升级用户体验感。

## 5.2 总结与反思

在做写程序实现功能的过程中，界面和基本功能的设计需要较强的逻辑思维，一不小心就容易出错，在开发完成后我明白了在具体的功能设计和实现的过程中要尽可能用最简单的语句实现基本的播放器功能, 所以我自认为的不足之处存在于：我觉得可以设计更多的菜单及在相应菜单项目实现更多的有趣功能，比如：收藏音乐，删除音乐，查询音乐，显示歌词等功能，或者UI的设计可以更加丰富一点，让用户打开app的体验更好，这样感觉整个音乐播放器更具有灵活性，媒体感更强烈一些，界面会更加丰富和生动一些。

通过自己做课程设计，自己设计界面，功能和设计功能的代码实现，将理论知识运用到实际操作中，加深了对理论知识的理解和记忆，在做程序设计的过程中，对很多代码的多次操作和重复使用，使得对很多基本的代码和相关的操作能够牢固的记，但发现自己在代码整洁方法还有待提到，书写的代码体系较为混乱，后续开发会消耗较多的时间。

此外，在设计过程中一直追求UI的设计，没有太注意内存优化和模块化，大多数对象与监听器采用了复用。在运行时的内存占用率一直较高，这也是后续有待优化之处。

**参考文献**

**[1]王向辉, 张国印, 沈洁. Android应用程序开发[M]. 3. 清华大学出版社, 2016-05-01.**

**[2] 郑蕾."一种高效率的软件开发方法——以用户为中心的软件开发方法（UCD）" (2006).**

**[3] 耿凌艳. “浅析人机交互界面“显性”版式设计” (2011).**

**[4] Paul Krill. JavaFX 14 enhances API, mobile support[J]. InfoWorld.com,2020:**

**[5] 叶佳骏 et al. “基于“云”技术的音乐播放器设计” (2012).**

**[6] Jim Clarke et al. “Javafx: Developing Rich Internet Applications” (2009).**

**[7] Kishori Sharan. “Understanding Concurrency in JavaFX” (2015).**

**[8] 余湉 and 付强. “面向用户体验的软件界面设计——以Pro/Engineer界面为例” (2015).**