摘 要

本项目为社交化音乐播放器，旨在在手机客户端实现音乐播放，在本地情况下可以进行音乐播放控制，本地音乐扫描等基础功能。

网络方面可以实现丰富的在线功能，如好友系统，在线评论，点赞等操作，并可根据用户的特点推荐用户偏向的歌曲。

后台方面可以方便地进行用户信息管理，可以由用户本人进行，也可以由管理员统一管理。本播放器具有强大的本地与网络功能，结构清晰，界面优美，是一个成功的项目。

**关键词：播放器，本地功能，网络功能，后台管理，界面优化**

# 

# 目 录

[摘 要 2](#_Toc27826)

[目 录 3](#_Toc16092)

[第一章 针对复杂工程问题的方案设计与实现 5](#_Toc27010)

[1.1针对复杂工程问题的方案设计 5](#_Toc6353)

[1.1.1本系统适用范围 5](#_Toc1296)

[1.1.2功能摘要 5](#_Toc31230)

[1.1.3运行环境约束 5](#_Toc13729)

[1.1.4系统分工 6](#_Toc7453)

[1.1.5具体需求 6](#_Toc6767)

[1.1.6系统功能的具体实现及关系. 8](#_Toc15963)

[1.1.7 ER图 8](#_Toc6491)

[1.1.8条件与限制 9](#_Toc9406)

[1.1.9具体功能描述 9](#_Toc7005)

[1.2针对复杂工程问题的推理分析 14](#_Toc25434)

[1.2.1后台交互 14](#_Toc25929)

[1.2.2歌词显示 14](#_Toc31097)

[1.2.3歌曲推荐 14](#_Toc14304)

[1.2.4社交化 15](#_Toc28918)

[1.3针对复杂工程问题的方案实现 16](#_Toc31047)

[1.3.1前端总结 16](#_Toc21575)

[1.3.2后端总结 24](#_Toc31580)

[第二章 系统测试 48](#_Toc25107)

[1.1编写目的 48](#_Toc19835)

[1.2背景说明 48](#_Toc21067)

[1.3定义 48](#_Toc15798)

[1.4 参考资料 49](#_Toc24850)

[1.4.1．测试 49](#_Toc3219)

[1.4.2测试结果及发现 49](#_Toc13830)

[1.4.3.缺陷和限制 50](#_Toc12218)

[第三章 知识技能学习情况 51](#_Toc12963)

[3.1.几大类的学习与掌握 51](#_Toc9037)

[3.1.1.Activity  51](#_Toc18638)

[3.1.2.Service 51](#_Toc1722)

[3.1.3.Broadcast Receiver 51](#_Toc17114)

[3.1.4.Content Provider 51](#_Toc26941)

[3.1.5.Intent 52](#_Toc28796)

[3.2.xml的学习与掌握 52](#_Toc2936)

[第四章 分工协作与交流情况 54](#_Toc2182)

[参考文献 55](#_Toc18123)

[致谢 55](#_Toc19814)

# 

# 第一章 针对复杂工程问题的方案设计与实现

## 1.1针对复杂工程问题的方案设计

### 1.1.1本系统适用范围

音乐播放器主要为手机提供播放音频的功能，适合所有喜爱音乐的人群使用。

### 1.1.2功能摘要

（1）使用音乐播放器播放歌曲；

（2）浏览歌曲列表；

（3）同步显示歌曲歌词

（4）拖动进度条调节播放进度

（6）后台播放歌曲

（7）上一曲/下一曲，播放/暂停等基本功能

（8）登录注册、找会密码（在线功能）

（9）好友系统（点赞、评论）

（10）数据库维护（包括歌曲的增删改查）、用户账号维护（包括修改用户信息，删除用户信息等）

### 1.1.3运行环境约束

操作系统：android2.2——4.2

数据库：QLite

软件平台：ADT

### 1.1.4系统分工

付鼎：后台功能的实现（包括登录，注册，推荐/在线播放歌曲）

耿睿：UI界面的制作与优化

龚子成：在线功能的实现（包括歌曲评论，点赞等）

程弈茗：聊天界面（分享歌曲的实现，好友系统），滚动歌词

### 1.1.5具体需求

（1）后台播放音频文件

描述：对指定音频文件进行后台播放，当程序被切换后台依旧可以播放。

加工：用户可在播放界面进行播放、暂停、下一曲/上一曲、播放模式切换、

进入播放列表、拖动播放进度操作。允许对本模块执行查询功能。

  查寻功能：点击播放按钮，获取自定义类Music，该类存储歌曲的信息。

 （2）显示歌曲歌词

  描述：对播放的歌曲的歌词进行同步显示。

  加工：当音乐播放时，自动搜寻对应的歌词并添加到播放界面，如果没

有歌词则不显示。允许对本模块执行查询功能。

  查询功能：查询歌曲对应的LRC歌词文件。

 （3）自动获取手机内音频文件

  描述：对手机内的歌曲进行自动加载。

  加工：打开软件进入到欢迎界面，欢迎界面两秒后自动跳转播放主界面，

在该界面完成自动遍历手机内的音频文件，并将数据保存到数据库

的表中，允许对本模块执行新增功能。

（4）显示歌曲列表

  描述：对获取的歌曲使用列表显示出来。

  加工：点击列表内的歌曲可以播放该曲目。允许对本模块执行查询功能。

  查询功能：查询数据库里歌曲的相关信息，添加到列表里；

  输出：歌曲名称与歌唱者显示在列表里。

（5）进度条显示歌曲播放进度，并可以拖动调整播放进度

 描述：实时更新歌曲的播放进度。

  操作：拖动进度条可调整歌曲播放进度。允许对本模块执行修改功能。

修改功能：拖动进度条修改音乐播放的进度。

  输出：歌曲播放进度被改变

（6）可在线点赞评论

描述：听歌的时候可以在评论区编写并发表评论，点赞。

操作：在评论输入框中编写评论，在点赞按钮上点赞。

输出：评论自动调整对齐

（7）登录/密码找回

描述：在主界面上可以登录进入本地用户，若忘记密码可以找回

操作：输入账号密码登录，若密码错误则拒绝登录。点击忘记密码会通过

邮箱发送随机八位密码进行重置。

### 1.1.6系统功能的具体实现及关系.

见下图1-1-6.

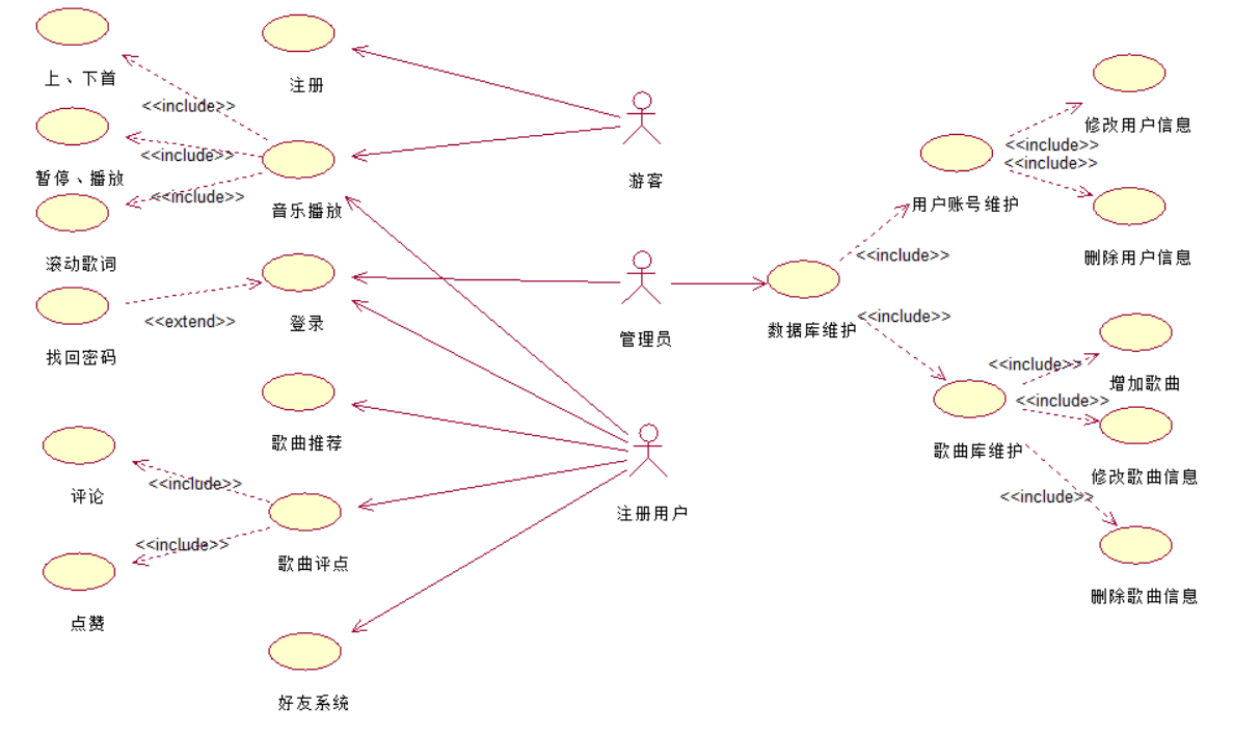


图1-1-6

### 1.1.7 ER图

见下图1-1-7.

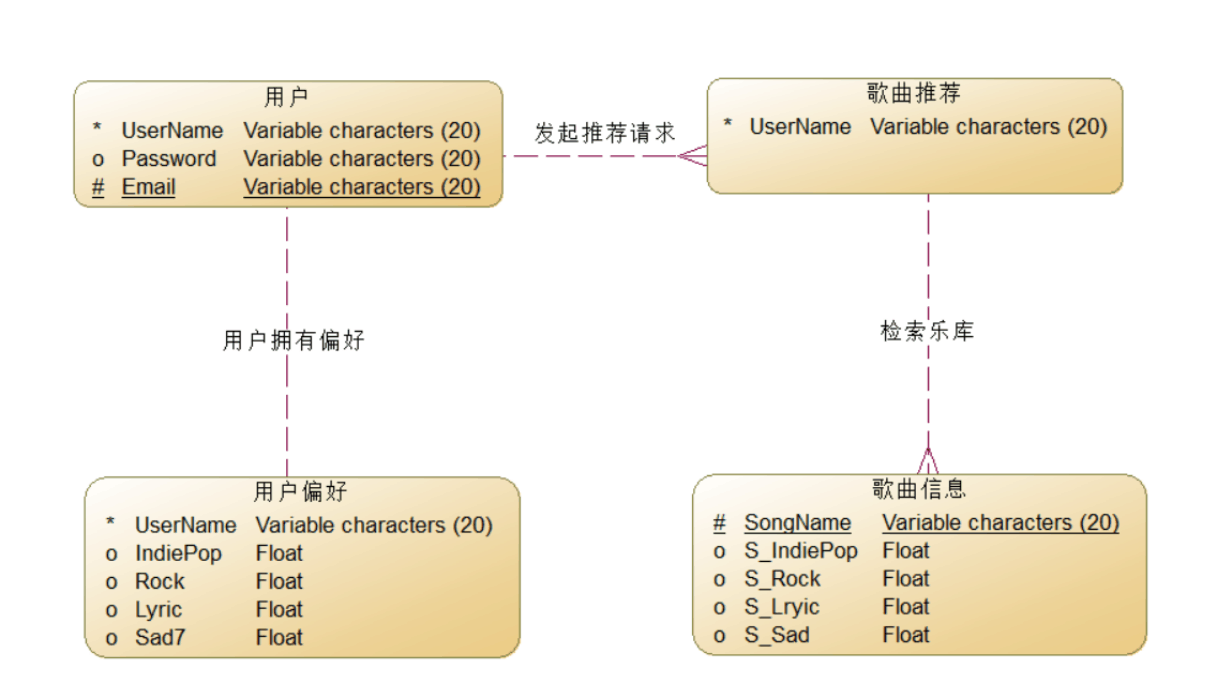


图1-1-7

### 1.1.8条件与限制

（1）开发人员开发经验不足；

（2）编程人员对开发工具不熟悉，编程经验非常有限；

### 1.1.9具体功能描述

1. 主界面，见图1-1-9-1.

基础功能：网络（在线搜索，听歌）

本地下载（从网上下载歌曲到本地）

用户信息（登录后可查看本地歌曲等）

列表：全部歌曲（显示本机储存的歌曲）

歌手（本地音乐按照歌手名字分类）

专辑（本地音乐按照专辑分类）

我的好友（可显示添加的好友）

播放列表（正在播放的歌曲所属列表）

我最爱听（本地收藏的喜爱歌曲）

最近播放（按照播放的时间顺序排序最近听过的歌曲）

歌曲扫描（扫描本地音乐并添加到全部歌曲中）

歌曲推荐（根据用户收藏的歌曲推荐用户可能喜欢的歌曲）



图1-1-9-1

（2）音乐播放界面,见图1-1-9-2.

回退按钮：退回主界面

音量键：滑动调整音量大小，范围为0到100

进度条：通过滑动控制播放进度

播放/暂停：控制音乐播放的停止/启动

菜单键：弹出菜单（收藏歌曲，标记为喜爱歌曲，下载等）

上一首/下一首：控制音乐切换上一首/下一首

循环切换：控制播放模式（单曲循环，列表循环等）



图1-1-9-2

（3）全部歌曲界面，见图1-1-9-3.

列出本地所有歌曲，用户可随意选择想要播放的歌曲，歌曲信息显示在位

面下方的控制台中。包括歌名，歌曲时长，播放暂停控制。



图1-1-9-3

1. 歌手页面，见图1-1-9-4.

所有歌曲可按照歌手名字分类，用户可以通过点击歌手名字找到相应的歌曲



图1-1-9-4

1. 用户喜爱歌曲页面，见图1-1-9-5.

在我最爱听列表中列出了用户标记的喜爱歌曲，用户可以在听歌时随时标记喜爱，并在此列表中查看。



图1-1-9-5

1. 歌曲扫描页面，见图1-1-9-6和图1-1-9-7

在本机中列出文件夹并扫描其中的歌曲，并将扫描到的歌曲添加进入全部歌

曲列表中



图1-1-9-6



图1-1-9-7

## 1.2针对复杂工程问题的推理分析

### 1.2.1后台交互

原理：是基于HttpClient，它是 Apache Jakarta Common 下的子项目，可以用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包。

步骤：

1. 创建HttpClient对象。

2. 创建请求方法的实例，并指定请求URL。

3. 调用setParams方法来添加请求参数

4. 调用HttpClient对象的execute(HttpUriRequest request)发送请求，该方法返回一个HttpResponse。

5. 调用HttpResponse的getHeaders(String name)方法可获取服务器的响应头；该对象包装了服务器的响应内容。程序可通过该对象获取服务器的响应内容。

6. 释放连接。无论执行方法是否成功，都必须释放连接

### 1.2.2歌词显示

　歌词同步的实现思路很简单：获取歌词文件LRC中的时间和歌词内容，然后在指定的时间内播放相应的内容。获取不难，难就在于如何在手机屏幕上实现歌词的滚动。通过MotionEvent事件来判断用户对歌词的操作，以实现进度同步和歌词大小调整。

### 1.2.3歌曲推荐

原理：潜在因子算法，这种算法是在NetFlix（美国知名流媒体提供商）的推荐算法竞赛中获奖的算法每个用户（user）都有自己的偏好，比如A喜欢带有小清新的、吉他伴奏的、王菲等元素（latent factor），如果一首歌（item）带有这些元素，那么就将这首歌推荐给该用户，也就是用元素去连接用户和音乐。每个人对不同的元素偏好不同，而每首歌包含的元素也不一样。每个用户对每首歌都这样计算可以得到不同用户对不同歌曲的评分矩阵。从而实现歌曲推荐功能。

### 1.2.4社交化

原理：我们知道，传统的HTTP协议是无状态的，每次请求（request）都要由客户端（如 浏览器）主动发起，服务端进行处理后返回response结果，而服务端很难主动向客户端发送数据；而对于涉及实时信息的Web应用却带来了很大的不便，WebSocket的工作流程是这 样的：浏览器通过JavaScript向服务端发出建立WebSocket连接的请求，在WebSocket连接建立成功后，客户端和服务端就可以通过 TCP连接传输数据。因为WebSocket连接本质上是TCP连接，不需要每次传输都带上重复的头部数据，所以它的数据传输量很小。

1.3针对复杂工程问题的方案实现

### 1.3.1前端总结

1.3.1.1歌词方案设计与实现

自定义View来实现解析lrc歌词同步滚动、上下拖动、缩放歌词等功能

关于歌词有以下几个功能：

1. 实现歌词同步滚动的功能，即歌曲播放到哪句歌词，就高亮地显示出正在播放的这个歌词；  
   2）实现上下拖动歌词时候，可以拖动播放器的进度。即可以不停地上下拖动歌词，当手指离开屏幕时候 即从当前拖动到的歌词位置播放。  
   3）实现歌词的字体大小可以进行缩放的功能。即双指在屏幕进行缩放操作时，歌词的字体大小也进行相应的缩放操作。

1.3.1.2歌词显示原理

LRC歌词文件：lrc是英文lyric（歌词）的缩写，被用做歌词文件的扩展名。以lrc为扩展名的歌词文件可以在各类数码播放器中同步显示。

LRC歌词文件的格式：lrc歌词文本中含有两类标签：一是标识标签 ，二是时间标签。

1）标识标签

标识标签，其格式为“[标识名:值]”，主要包含以下预定义的标签：

[ar:歌手名] [ti:歌曲名] [al:专辑名] [by:编辑者(指lrc歌词的制作人)] [offset:时间补偿值] （其单位是毫秒，正值表示整体提前，负值相反。这是用于总体调整显示快慢的，但多数的MP3可能不会支持这种标签）。

2）时间标签

时间标签，形式为“[mm:ss]”或“[mm:ss.ff]”(分钟数:秒数.毫秒数)，数字须为非负整数，

比如”[12:34.50]”是有效的，而”[0x0C:-34.50]”无效。

时间标签需位于某行歌词中的句首部分，一行歌词可以包含多个时间标签

(比如歌词中的迭句部分)。当歌曲播放到达某一时间点时，MP3就会寻找对应的时间标签并显示标签后面的歌词文本，这样就完成了“歌词同步”的功能。

解析lrc歌词文件：从assets目录下读取test.lrc歌词文件内容，则可以调用上面的getFromAssets(String fileName)方法得到歌词的文本内容。

1.3.1.3创建实体类LrcRow

首先封装一个表示每行歌词内容的实体类LrcRow，该类由三个属性，分别为：  
strTime、time、content。例如一行歌词内容为：[02:34.14]当你我不小心又想起她 , 解析该行歌词后的实体类LrcRow的属性如下所示：

strTime表示该行歌词要开始播放的时间，格式如下：[02:34.14]

time表示将strTime转换为long型之后的数值

例如将strTime为[02:34.14]格式转换154014（154014=02 \* 60 \* 1000 + 34 \* 1000+14）content表示该行歌词的内容，如：当你我不小心又想起她

LrcRow中存在一个List createRows(String standardLrcLine)方法 ，该方法将循环地一行一行的去读取歌词的内容。然后对每一行的歌词进行解析，每解析出一个时间标签[XX:XX.XX]则new出一个LrcRow对象，然后加入到歌词行List集合中去。

该LrcRow类实现Comparable接口，用来进行解析之后的排序操作，排序按时间从小到大排序。

1.3.1.4.ILrcBuilder接口和DefaultLrcBuilder歌词解析构造器

定义一个ILrcBuilder接口，接口有一个List getLrcRows(String rawLrc)方法，该方法用来解析歌词，得到LrcRow的集合，DefaultLrcBuilder实现ILrcBuilder接口，List getLrcRows(String rawLrc)方法会循环地读取歌词的每一行，然后调用LrcRow类的List createRows(String standardLrcLine)方法，得到解析每一行歌词之后的LrcRow集合，再将每一行得到LrcRow集合中得到的LrcRow实体加入一个总 的到LrcRow集合rows中去，然后将rows集合根据歌词行的时间排序，得到排序后的LrcRow集合，该集合就是最终的解析歌词后的内容了。

1.3.1.5.ILrcViewListener接口（在ILrcView中使用）

ILrcViewListener接口，该接口定义了一个onLrcSeeked方法用来监听用户上下拖动歌词的动作定义了一个方法onLrcSeeked(int newPosition, LrcRow row)

1.3.1.6定义一个实体类ILrcView和ILrcView接口

自定义一个LrcView，该LrcView继承android.view.View对象，实现了ILrcView接口。该自定义LrcView可以实现了同步显示歌词，拖动歌词，缩放歌词等功能。

1.3.1.2同步显示歌词功能

首先来说说显示歌词的实现思路，要显示歌词即把歌词的内容绘制出来，可以分以下三步来绘制歌词：

第1步：高亮地画出正在播放的那句歌词

第2步：画出正在播放的那句歌词的上面可以展示出来的歌词

第3步：画出正在播放的那句歌词的下面的可以展示出来的歌词

重写onDraw(Canvas canvas)方法，在方法中按照上面的思路来绘制者三部分的歌词。

ILrcView接口定义了三个方法：

setLrc(List lrcRows);调用该方法设置要展示的歌词行集合;seekLrcToTime(long time)

音乐播放的时候调用该方法滚动歌词，高亮正在播放的那句歌词

setListener(ILrcViewListener l),调用该方法设设置歌词拖动时候的监听类，用以回调ILrcViewListener的onLrcSeeked(int newPosition, LrcRow row)方法

1.3.1.3.详细功能：

1.3.1.3.1同步显示歌词功能

为了实现同步显示功能的功能，则需要不停地将自定义的LrcView进行重绘。首先当MediaPlayer开始播放的时候，同步的启动一个TimerTask来进行歌词的滚动操作。即当MediaPlayer开始播放的时候，启动一个定时器Timer，然后通过这个定时器每隔mPalyTimerDuration时间来执行一次LrcTask任务。

|  |
| --- |
| mPlayer.setOnPreparedListener(new OnPreparedListener() {   //准备完毕   public void onPrepared(MediaPlayer mp) {   mp.start();   if(mTimer == null){             mTimer = new Timer();             mTask = new LrcTask();             mTimer.scheduleAtFixedRate(mTask, 0, mPalyTimerDuration);             }   }  });  /\*\*  \* 展示歌曲的定时任务  \*/      class LrcTask extends TimerTask{          @Override          public void run() {              //获取歌曲播放的位置              final long timePassed = mPlayer.getCurrentPosition();              MainActivity.this.runOnUiThread(new Runnable() {                  public void run() {                      //滚动歌词                      mLrcView.seekLrcToTime(timePassed);                  }              });            }      }; |

首先获取MediaPlayer的播放进度值，然后调用了LrcView的seekLrcToTime(long time)方法进行歌词同步滚动，首先通过传入进来的MediaPlayer的播放进度值，来判断需要高亮地歌词行LrcRow是哪一行，然后调用seekLrc(int position, boolean cb)方法来进行歌词重绘操作。

|  |
| --- |
| public void seekLrc(int position, boolean cb) {     if (mLrcRows == null || position < 0 || position > mLrcRows.size()) {          return;     }     LrcRow lrcRow = mLrcRows.get(position);     mHignlightRow = position;     invalidate();     //如果是手指拖动歌词后     if (mLrcViewListener != null && cb) {     //回调onLrcSeeked方法，将音乐播放器播放的位置移动到高亮歌词的位置         mLrcViewListener.onLrcSeeked(position, lrcRow);     }  } |

1.3.1.3.2.拖动歌词的功能

当歌词进行拖动的时候，回调ILrcViewListener接口的onLrcSeeked(int newPosition, LrcRow row)方法。

判断手指在屏幕上的操作，来进行歌词滚动的操作。重写onTouchEvent(MotionEvent event)方法，来判断手指的操作是拖动歌词还是缩放歌词，当一个手指移动的时候，则调用doSeek(MotionEvent event)方法来进行拖动歌词的操作。当一个手指不停的在屏幕上移动时，将会不停地调用doSeek(MotionEvent event)方法来进行LrcView的重绘操作，从而实现了歌词拖动的效果。当手指离开屏幕的时候，即MotionEvent 为MotionEvent.ACTION\_UP的时候，会调用seekLrc(int position, boolean cb)方法，从而回调ILrcViewListener接口的onLrcSeeked方法，来拖动MediaPlayer的播放进度值，从而达到了拖动歌词后从最终高亮的歌词开始重新播放歌词的功能。

|  |
| --- |
| public void seekLrc(int position, boolean cb) {      if (mLrcRows == null || position < 0 || position > mLrcRows.size()) {          return;       }       LrcRow lrcRow = mLrcRows.get(position);       mHignlightRow = position;       invalidate();       //如果是手指拖动歌词后       if (mLrcViewListener != null && cb) {       //回调onLrcSeeked方法，将音乐播放器播放的位置移动到高亮歌词的位置            mLrcViewListener.onLrcSeeked(position, lrcRow);       }  } |

1.3.1.3.3.缩放歌词的功能

当两个手指第一次放在屏幕上时候，调用setTwoPointerLocation(MotionEvent event)方法来记录两个手指的x坐标和y坐标。当两个手指在屏幕上移动的时候，调用getScale(MotionEvent event)方法来对比两个手指前后两次的x坐标和y坐标，从而得到要缩放的比例scaleSize。当通过getScale(MotionEvent event)方法获得了缩放比scaleSize后，调用setNewFontSize(int scaleSize)来设置歌词的新的字体大小，然后重绘LrcView，从而实现了缩放歌词的功能。

|  |
| --- |
| /\*\*  \* 处理双指在屏幕移动时的，歌词大小缩放  \*/      private void doScale(MotionEvent event) {      //如果歌词的模式为：拖动歌词模式      if (mDisplayMode == DISPLAY\_MODE\_SEEK) {      //如果是单指按下，在进行歌词上下滚动，然后按下另外一个手指，则把歌词模式从 拖动歌词模式 变为 缩放歌词模式          mDisplayMode = DISPLAY\_MODE\_SCALE;          Log.d(TAG, "change mode from DISPLAY\_MODE\_SEEK to DISPLAY\_MODE\_SCALE");          return;       }       // two pointer mode , scale font       if (mIsFirstMove) {           mDisplayMode = DISPLAY\_MODE\_SCALE;           invalidate();           mIsFirstMove = false;           //两个手指的x坐标和y坐标           setTwoPointerLocation(event);       }       //获取歌词大小要缩放的比例       int scaleSize = getScale(event);       Log.d(TAG, "scaleSize:" + scaleSize);       //如果缩放大小不等于0，进行缩放，重绘LrcView        if (scaleSize != 0) {            setNewFontSize(scaleSize);              invalidate();        }        setTwoPointerLocation(event);  }  /\*\*  \* 设置当前两个手指的x坐标和y坐标  \*/      private void setTwoPointerLocation(MotionEvent event) {          mPointerOneLastMotion.x = event.getX(0);          mPointerOneLastMotion.y = event.getY(0);          mPointerTwoLastMotion.x = event.getX(1);          mPointerTwoLastMotion.y = event.getY(1);      }  private int getScale(MotionEvent event) {      Log.d(TAG, "scaleSize getScale");      float x0 = event.getX(0);      float y0 = event.getY(0);      float x1 = event.getX(1);      float y1 = event.getY(1);    float maxOffset = 0;  // max offset between x or y axis,used to decide scale size        boolean zoomin = false;      //第一次双指之间的x坐标的差距      float oldXOffset = Math.abs(mPointerOneLastMotion.x - mPointerTwoLastMotion.x);      //第二次双指之间的x坐标的差距      float newXoffset = Math.abs(x1 - x0);      //第一次双指之间的y坐标的差距      float oldYOffset = Math.abs(mPointerOneLastMotion.y - mPointerTwoLastMotion.y);      //第二次双指之间的y坐标的差距      float newYoffset = Math.abs(y1 - y0);      //双指移动之后，判断双指之间移动的最大差距      maxOffset = Math.max(Math.abs(newXoffset - oldXOffset), Math.abs(newYoffset - oldYOffset));      //如果x坐标移动的多一些      if (maxOffset == Math.abs(newXoffset - oldXOffset)) {       //如果第二次双指之间的x坐标的差距大于第一次双指之间的x坐标的差距则是放大，反之则缩小      zoomin = newXoffset > oldXOffset ? true : false;      }      //如果y坐标移动的多一些      else {      //如果第二次双指之间的y坐标的差距大于第一次双指之间的y坐标的差距则是放大，反之则缩小      zoomin = newYoffset > oldYOffset ? true : false;      }      Log.d(TAG, "scaleSize maxOffset:" + maxOffset);      if (zoomin) {           return (int) (maxOffset / 10);//放大双指之间移动的最大差距的1/10      } else {           return -(int) (maxOffset / 10);//缩小双指之间移动的最大差距的1/10       }  } |

### 1.3.2后端总结

安卓端与服务器进行交互，是基于HttpClient，它 是 Apache Jakarta Common 下的子项目，可以用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包。

服务器端的步骤：

1. 创建HttpClient对象。

2. 创建请求方法的实例，并指定请求URL。如果需要发送GET请求，创建HttpGet对象；如果需要发送POST请求，创建HttpPost对象。

3. 如果需要发送请求参数，可调用HttpGet、HttpPost共同的setParams(HetpParams params)方法来添加请求参数；对于HttpPost对象而言，也可调用setEntity(HttpEntity entity)方法来设置请求参数。

4. 调用HttpClient对象的execute(HttpUriRequest request)发送请求，该方法返回一个HttpResponse。

5. 调用HttpResponse的getAllHeaders()、getHeaders(String name)等方法可获取服务器的响应头；调用HttpResponse的getEntity()方法可获取HttpEntity对象，该对象包装了服务器的响应内容。程序可通过该对象获取服务器的响应内容。

6. 释放连接。无论执行方法是否成功，都必须释放连接

代码：

|  |
| --- |
| *//设置URL地址*  **private static** String *ProcessURL*=**"http://210.41.99.178:8080/MusicService/finds.action?"**;  *//创建一个HttpClient对象* HttpClient httpclient = **new** DefaultHttpClient(); *//远程登录URL* processURL=*ProcessURL*+**"userName="**+userName+**"&email="**+password; Log.*d*(**"远程URL"**, processURL); *//创建HttpGet对象* HttpGet request=**new** HttpGet(processURL); *//请求信息类型MIME每种响应类型的输出（普通文本、html 和 XML，json）。允许的响应类型应当匹配资源类中生成的 MIME 类型* request.addHeader(**"Accept"**,**"text/json"**); *//获取响应的结果* HttpResponse response =httpclient.execute(request); *//获取HttpEntity* HttpEntity entity=response.getEntity(); *//获取响应的结果信息* String json = EntityUtils.*toString*(entity, **"UTF-8"**); *//JSON的解析过程* **if**(json!=**null**){  JSONObject jsonObject=**new** JSONObject(json);  result=jsonObject.get(**"message"**).toString();  } **if**(result==**null**){  json=**"登录失败请重新登录"**; } *//创建提示框提醒是否登录成功* AlertDialog.Builder builder=**new** AlertDialog.Builder(find.**this**); builder.setTitle(**"提示"**)  .setMessage(result)  .setPositiveButton(**"确定"**, **new** DialogInterface.OnClickListener() {   @Override  **public void** onClick(DialogInterface dialog, **int** which) {  dialog.dismiss();  }  }).create().show(); |

后台登陆代码：

|  |
| --- |
| //设置编码格式  **this**.response.setContentType("text/json;charset=utf-8");  **this**.response.setCharacterEncoding("UTF-8");  //初始化JSON  Map<String,String> json=**new** HashMap<String,String>();    **try**{  //初始化会话  SessionFactory sf = **new** Configuration().configure().buildSessionFactory();  Session session = sf.openSession();  //创建事务  Transaction tx = session.beginTransaction();  //创建SQL语句  String sqlQuery = "select u from UserInfo u where u.username='"+ userName + "'";  //执行SQL语句  Query lQuery = session.createQuery(sqlQuery);  //取出结果  ArrayList userList = (ArrayList) lQuery.list();  //关闭事务、会话  tx.commit();  session.close();  //新建User对象  UserInfo user = **new** UserInfo();  //判断返回结果是否为空  **if** ((**null** != userList) && (userList.size() > 0)) {  //取出返回的结果  user = (UserInfo) userList.get(0);  //放入登陆结果信息  **if** (user.getPassword().equals(password)) {  json.put("message", "登录成功!");  }**else**{  json.put("message", "账户或密码错误!");  }  }  }**catch**(HibernateException e){  e.printStackTrace();  }  //写入HttpServletResponse  **byte**[] jsonBytes = json.toString().getBytes("utf-8");  response.setContentLength(jsonBytes.length);  response.getOutputStream().write(jsonBytes);  response.getOutputStream().flush();  response.getOutputStream().close(); |

找回密码：

|  |
| --- |
| UserInfo user = **new** UserInfo();  **if** ((**null** != userList) && (userList.size() > 0)) {  user = (UserInfo) userList.get(0);  **if** (user.getEmai().equals(email)) {  String temppasswod=**null**;  //随机初始化8位密码  TempPassword temp=**new** TempPassword();  //发送邮件  Email e=**new** Email();  temppasswod=temp.getpassword();  e.send(email,temppasswod);  user.setPassword(temppasswod);  //将新密码保存到数据库中  session.save(user);  json.put("message", "重置密码成功！请在邮箱中查询！");  }**else**{  json.put("message", "NO！");  }  } |

其中发送邮箱是基于java.mail包实现，其大致流程如下：

1. 构建一个继承自javax.mail.Authenticator的具体类，并重写里面的getPasswordAuthentication()方法。此类是用作登录校验的，以确保你对该邮箱有发送邮件的权利。  
   2、构建一个properties文件，该文件中存放SMTP服务器地址等参数。  
   3、通过构建的properties文件和javax.mail.Authenticator具体类来创建一个javax.mail.Session。Session的创建，就相当于登录邮箱一样。剩下的自然就是新建邮件。  
   4、构建邮件内容，一般是javax.mail.internet.MimeMessage对象，并指定发送人，收信人，主题，内容等等。  
   5、使用javax.mail.Transport工具类发送邮件。

代码如下：

|  |
| --- |
| **final** String from = "fd159951@sina.com";  **final** String frompassword="fd159951";  // 指定发送邮件的主机为 smtp.sina.com  String host = "smtp.sina.com"; //sina 邮件服务器  Properties properties = System.*getProperties*();  // 设置邮件服务器  properties.setProperty("mail.smtp.host", host);  properties.put("mail.smtp.auth", "true");  // 获取默认session对象  Session session = Session.*getDefaultInstance*(properties,**new** Authenticator(){  **public** PasswordAuthentication getPasswordAuthentication()  {  **return** **new** PasswordAuthentication(from, frompassword); //发件人邮件用户名、密码  } });  **try**{  // 创建默认的 MimeMessage 对象  MimeMessage message = **new** MimeMessage(session);  // Set From: 头部头字段  message.setFrom(**new** InternetAddress(from));  // Set To: 头部头字段  message.addRecipient(Message.RecipientType.***TO***,**new**  InternetAddress(to));  // Set Subject: 头部头字段  message.setSubject("重置用户密码");    // 设置消息体  message.setText("你的新密码是："+temp\_password);  // 发送消息  Transport.*send*(message);  System.***out***.println("Sent message successfully");  }**catch** (MessagingException mex) {  mex.printStackTrace();  } |

推荐歌曲算法：潜在因子算法，每个用户（user）都有自己的偏好，比如A喜欢带有小清新的、吉他伴奏的、王菲等元素（latent factor），如果一首歌（item）带有这些元素，那么就将这首歌推荐给该用户，也就是用元素去连接用户和音乐。每个人对不同的元素偏好不同，而每首歌包含的元素也不一样。每个用户对每首歌都这样计算可以得到不同用户对不同歌曲的评分矩阵。

代码：

|  |
| --- |
| //获取歌曲信息  SongInfo song = **new** SongInfo();  //计算歌曲列表长度  **float**[] sum = **new** **float**[songlist.size()];  **for**(**int** p=0;p<songlist.size();p++){  res[p] = p;  }  //矩阵求和  **for**(**int** i=0;i<songlist.size();i++){  song = (SongInfo)songlist.get(i);  sum[i]=up.getIndiePop()\*song.getIndiepop()+up.getLyric()\*song.getLryic()+up.getRock()+song.getRock()+up.getSad()\*song.getSad();  }  //选择排序  **for**(**int** j=0;j<sum.length;j++){  **for**(**int** k=j;k<sum.length;k++){  **float** temp= 0;  **int** res\_temp = 0;  **if**(sum[j]<sum[k]){  temp = sum[j];sum[j] = sum[k];sum[k] = temp;  res\_temp = res[j];res[j] = res[k];res[k] = res\_temp;  }  }  } |

安卓端：

|  |
| --- |
| ibtn\_list\_content\_icon=(ImageButton)rl\_list\_content.findViewById(R.id.ibtn\_list\_content\_icon);  ibtn\_list\_content\_do\_icon=(ImageButton)rl\_list\_content.findViewById(R.id.ibtn\_list\_content\_do\_icon);  tv\_list\_content\_title=(TextView)rl\_list\_content.findViewById(R.id.tv\_list\_content\_title);  lv\_list\_change\_content=(ListView)rl\_list\_content.findViewById(R.id.lv\_list\_change\_content);  ibtn\_list\_content\_icon.setOnClickListener(imageButton\_listener);  ibtn\_list\_content\_do\_icon.setOnClickListener(imageButton\_listener);  lv\_list\_change\_content.setOnItemClickListener(list\_change\_content\_listener);  lv\_list\_change\_content.setOnItemLongClickListener(list\_change\_content\_looglistener);  btn\_list\_random\_music2=(Button)rl\_list\_content.findViewById(R.id.btn\_list\_random\_music2);  btn\_list\_random\_music2.setOnClickListener(btn\_randomPlayerListener); |

注释：本地音乐和下载管理的二三级布局-公共 标题和内容区域

|  |
| --- |
| Timer mTimer=null;  TimerTask mTask;  class LrcTask extends TimerTask {  @Override  public void run() {  //获取歌曲播放的位置  final int timePassed = mediaPlayerManager.getPlayerProgress();  runOnUiThread(new Runnable() {  public void run() {  //滚动歌词  mLrcView.seekLrcToTime(timePassed);  }  });  }  }  if(mTimer == null){  mTimer = new Timer();  mTask = new LrcTask();  mTimer.scheduleAtFixedRate(mTask, 0, 1000);  }  } |

注释：使用TimerTask用来监听播放歌曲的进度，用来控制拖动条的进度和滚动歌词的进度

|  |
| --- |
| private class MediaPlayerBroadcastReceiver extends BroadcastReceiver{  @Override  public void onReceive(Context context, Intent intent) {  int flag=intent.getIntExtra("flag", -1);  if(flag==MediaPlayerManager.FLAG\_CHANGED){  if(!isSeekDrag){  int currentPosition=intent.getIntExtra("currentPosition", 0);  int duration=intent.getIntExtra("duration", 0);  tv\_player\_playing\_time.setText(Common.formatSecondTime(currentPosition));  tv\_player\_playering\_duration.setText(Common.formatSecondTime(duration));  sb\_player\_playprogress.setProgress(currentPosition);  sb\_player\_playprogress.setMax(duration);  }  }else if(flag==MediaPlayerManager.FLAG\_PREPARE){  String albumPic=intent.getStringExtra("albumPic");  tv\_player\_song\_info.setText(intent.getStringExtra("title"));  l.setTitle(intent.getStringExtra("title"));  mLrcView.setLrc(l.lyrics(l.getTitle()));  if(TextUtils.isEmpty(albumPic)){  setAlbum(R.drawable.default\_album);  }else{  Bitmap bitmap=BitmapFactory.decodeFile(albumPic);  //判断SD图片是否存在  if(bitmap!=null){  setAlbum(bitmap);  }else{  setAlbum(R.drawable.default\_album);  }  }  int duration=intent.getIntExtra("duration", 0);  int currentPosition=intent.getIntExtra("currentPosition", 0);  tv\_player\_playing\_time.setText(Common.formatSecondTime(currentPosition));  tv\_player\_playering\_duration.setText(Common.formatSecondTime(duration));  sb\_player\_playprogress.setMax(duration);  sb\_player\_playprogress.setProgress(currentPosition);  sb\_player\_playprogress.setSecondaryProgress(0);  }else if(flag==MediaPlayerManager.FLAG\_INIT){//初始化播放信息  int currentPosition=intent.getIntExtra("currentPosition", 0);  int duration=intent.getIntExtra("duration", 0);  int playerMode=intent.getIntExtra("playerMode", 0);  int playerState=intent.getIntExtra("playerState", 0);  if(playerState==MediaPlayerManager.STATE\_PLAYER||playerState==MediaPlayerManager.STATE\_PREPARE){//播放  ibtn\_player\_control\_play.setBackgroundResource(R.drawable.player\_btn\_player\_pause);  }else{  ibtn\_player\_control\_play.setBackgroundResource(R.drawable.player\_btn\_player\_play);  }  ibtn\_player\_control\_mode  .setBackgroundResource(MODE\_DRAWABLE\_ID[playerMode]);  sb\_player\_playprogress.setMax(duration);  sb\_player\_playprogress.setProgress(currentPosition);  tv\_player\_playing\_time.setText(Common.formatSecondTime(currentPosition));  tv\_player\_playering\_duration.setText(Common.formatSecondTime(duration));  tv\_player\_song\_info.setText(intent.getStringExtra("title"));  l.setTitle(intent.getStringExtra("title"));  mLrcView.setLrc(l.lyrics(l.getTitle()));  System.out.println(l.getTitle());  String albumPic=intent.getStringExtra("albumPic");  if(TextUtils.isEmpty(albumPic)){  setAlbum(R.drawable.default\_album);  }else{  Bitmap bitmap=BitmapFactory.decodeFile(albumPic);  //判断SD图片是否存在  if(bitmap!=null){  setAlbum(bitmap);  }else{  setAlbum(R.drawable.default\_album);  }  }  }else if(flag==MediaPlayerManager.FLAG\_BUFFERING){  int percent=intent.getIntExtra("percent", 0);  percent=(int)(sb\_player\_playprogress.getMax()/100f)\*percent;  sb\_player\_playprogress.setSecondaryProgress(percent);  }else if(flag==MediaPlayerManager.FLAG\_LIST){  int state=mediaPlayerManager.getPlayerState();  if(state==MediaPlayerManager.STATE\_PLAYER||state==MediaPlayerManager.STATE\_PREPARE){//播放  ibtn\_player\_control\_play.setBackgroundResource(R.drawable.player\_btn\_player\_pause);  }else{  ibtn\_player\_control\_play.setBackgroundResource(R.drawable.player\_btn\_player\_play);  }  }  }  } |

注释：使用广播的方法实现歌曲的后台播放，当播放状态改变（比如拖动进度条）会激活广播监听器，使歌曲的播放状态做出相应的改变。

|  |
| --- |
| //下一首动作  PendingIntent nextPendingIntent=PendingIntent.getService(this, 1, new Intent(MediaPlayerManager.SERVICE\_ACTION)  .putExtra("flag", MediaPlayerManager.SERVICE\_MUSIC\_NEXT)  , PendingIntent.FLAG\_UPDATE\_CURRENT);  remoteViews.setOnClickPendingIntent(R.id.btn\_widget\_next, nextPendingIntent);  //播放/暂停动作  PendingIntent playerPendingIntent=PendingIntent.getService(this, 2, new Intent(MediaPlayerManager.SERVICE\_ACTION)  .putExtra("flag", MediaPlayerManager.SERVICE\_MUSIC\_PLAYERORPAUSE)  , PendingIntent.FLAG\_UPDATE\_CURRENT);  remoteViews.setOnClickPendingIntent(R.id.btn\_widget\_player, playerPendingIntent);  remoteViews.setOnClickPendingIntent(R.id.btn\_widget\_pause, playerPendingIntent);  //上一首  PendingIntent prevPendingIntent=PendingIntent.getService(this, 3, new Intent(MediaPlayerManager.SERVICE\_ACTION)  .putExtra("flag", MediaPlayerManager.SERVICE\_MUSIC\_PREV)  , PendingIntent.FLAG\_UPDATE\_CURRENT);  remoteViews.setOnClickPendingIntent(R.id.btn\_widget\_prev, prevPendingIntent); |

注释：通过服务的绑定，当用户发出上一首、下一首、播放、暂停的动作，会发出相应的广播信息，当广播监听器接收到相应的意图信息，将会做出相应的处理，从而实现歌曲后台播放，广播监听器全局监听，动作激活改变播放状态。

|  |
| --- |
| public class ScanDirectoryActivity extends SettingActivity {  private ListView lv\_scan\_music\_list;  private File[] files;  private List<ScanData> data;  private ScanMusicFilterFile myFilterFile;//过滤文件  private File currrentFile;//当前文件的路径  private ScanListAdapter adapter;  private String rootFilePath;//父级文件的路径  private String rs="";//返回结果  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.scan\_music\_adddirectory);  resultCode=ScanMusicActivity.SCAN\_MUSIC\_CANCEL;  setBackButton();  setTopTitle(getResources().getString(R.string.scan\_directory\_title));  //检测SD卡  if(!Common.isExistSdCard()){  Toast.makeText(this, "请先插入SD卡", Toast.LENGTH\_SHORT).show();  setResult(RESULT\_CANCELED);  this.finish();  }  lv\_scan\_music\_list=(ListView)this.findViewById(R.id.lv\_scan\_music\_list);  ((Button)this.findViewById(R.id.btn\_scan\_add)).setOnClickListener(listener);  ((Button)this.findViewById(R.id.btn\_scan\_back)).setOnClickListener(listener);  ((Button)this.findViewById(R.id.btn\_scan\_directory\_goup)).setOnClickListener(listener);  //获取默认音乐目录的选取结果  rs=getIntent().getStringExtra("rs");  //读取mnt根目录  myFilterFile=new ScanMusicFilterFile();  lv\_scan\_music\_list=(ListView)this.findViewById(R.id.lv\_scan\_music\_list);  currrentFile=Environment.getExternalStorageDirectory().getParentFile();//获取sd卡路径的父目录文件  rootFilePath=currrentFile.getPath().toLowerCase();//设置根目录的路径  data=new ArrayList<ScanData>();  getFilePath(currrentFile);//遍历目录  adapter=new ScanListAdapter(this,data);  adapter.setCheckFilePath(rs);  lv\_scan\_music\_list.setAdapter(adapter);  lv\_scan\_music\_list.setOnItemClickListener(itemClickListener);  }  private OnClickListener listener=new OnClickListener() {  public void onClick(View v) {  switch (v.getId()) {  case R.id.btn\_scan\_add:  //添加扫描目录  Intent addit=new Intent(ScanDirectoryActivity.this,ScanMusicActivity.class);  addit.putExtra("rs", adapter.getCheckFilePath());  setResult(ScanMusicActivity.SCAN\_MUSIC\_OK, addit);  finish();  break;  case R.id.btn\_scan\_directory\_goup:  //判断是否是父级目录  if(!currrentFile.getPath().toLowerCase().equals(rootFilePath)){  currrentFile=currrentFile.getParentFile();  getFilePath(currrentFile);  adapter.notifyDataSetChanged();  }  break;  case R.id.btn\_scan\_back:  setResult(-1);  finish();  break;  default:  break;  }  }  };  private OnItemClickListener itemClickListener=new OnItemClickListener() {  public void onItemClick(AdapterView<?> parent, View view, int position,  long id) {  //进入下一级目录  currrentFile=files[position];  rs=adapter.getCheckFilePath();  getFilePath(currrentFile);  adapter.notifyDataSetChanged();  }  };  private void getFilePath(File parent){  data.clear();  files=parent.listFiles(myFilterFile);  for(File file:files){  String fp=file.getPath().toLowerCase();  //判断是否是选中的  ScanData d=new ScanData(fp+"/",false);  if(rs.contains("$"+fp+"/$")){  d.setChecked(true);  }  data.add(d);  }  }  }  注释：通过遍历查找实现指定目录下的歌曲扫描。  //当没有歌词的时候  if (mLrcRows == null || mLrcRows.size() == 0) {  if (mLoadingLrcTip != null) {  // draw tip when no lrc.  mPaint.setColor(mHignlightRowColor);  mPaint.setTextSize(mLrcFontSize);  mPaint.setTextAlign(Align.CENTER);  canvas.drawText(mLoadingLrcTip, width / 2, height / 2 - mLrcFontSize, mPaint);  }  return;  }  int rowY = 0; // vertical point of each row.  final int rowX = width / 2;  int rowNum = 0;    String highlightText = mLrcRows.get(mHignlightRow).content;  int highlightRowY = height / 2 - mLrcFontSize;  mPaint.setColor(mHignlightRowColor);  mPaint.setTextSize(mLrcFontSize);  mPaint.setTextAlign(Align.CENTER);  canvas.drawText(highlightText, rowX, highlightRowY, mPaint);  // 上下拖动歌词的时候 画出拖动要高亮的那句歌词的时间 和 高亮的那句歌词下面的一条直线  if (mDisplayMode == DISPLAY\_MODE\_SEEK) {  // 画出高亮的那句歌词下面的一条直线  mPaint.setColor(mSeekLineColor);  //该直线的x坐标从0到屏幕宽度 y坐标为高亮歌词和下一行歌词中间  canvas.drawLine(mSeekLinePaddingX, highlightRowY + mPaddingY, width - mSeekLinePaddingX, highlightRowY + mPaddingY, mPaint);  // 画出高亮的那句歌词的时间  mPaint.setColor(mSeekLineTextColor);  mPaint.setTextSize(mSeekLineTextSize);  mPaint.setTextAlign(Align.LEFT);  canvas.drawText(mLrcRows.get(mHignlightRow).strTime, 0, highlightRowY, mPaint);  }  // 2、画出正在播放的那句歌词的上面可以展示出来的歌词  mPaint.setColor(mNormalRowColor);  mPaint.setTextSize(mLrcFontSize);  mPaint.setTextAlign(Align.CENTER);  rowNum = mHignlightRow - 1;  rowY = highlightRowY - mPaddingY - mLrcFontSize;  //只画出正在播放的那句歌词的上一句歌词  // if (rowY > -mLrcFontSize && rowNum >= 0) {  // String text = mLrcRows.get(rowNum).content;  // canvas.drawText(text, rowX, rowY, mPaint);  // }  //画出正在播放的那句歌词的上面所有的歌词  while( rowY > -mLrcFontSize && rowNum >= 0){  String text = mLrcRows.get(rowNum).content;  canvas.drawText(text, rowX, rowY, mPaint);  rowY -= (mPaddingY + mLrcFontSize);  rowNum --;  }  // 3、画出正在播放的那句歌词的下面的可以展示出来的歌词  rowNum = mHignlightRow + 1;  rowY = highlightRowY + mPaddingY + mLrcFontSize;  //只画出正在播放的那句歌词的下一句歌词  // if (rowY < height && rowNum < mLrcRows.size()) {  // String text2 = mLrcRows.get(rowNum).content;  // canvas.drawText(text2, rowX, rowY, mPaint);  // }  //画出正在播放的那句歌词的所有下面的可以展示出来的歌词  while( rowY < height && rowNum < mLrcRows.size()){  String text = mLrcRows.get(rowNum).content;  canvas.drawText(text, rowX, rowY, mPaint);  rowY += (mPaddingY + mLrcFontSize);  rowNum ++;  }  }  /\*\*  \* 设置要高亮的歌词为第几行歌词  \*  \* @param position 要高亮的歌词行数  \* @param cb 是否是手指拖动后要高亮的歌词  \*/  public void seekLrc(int position, boolean cb) {  if (mLrcRows == null || position < 0 || position > mLrcRows.size()) {  return;  }  LrcRow lrcRow = mLrcRows.get(position);  mHignlightRow = position;  invalidate();  //如果是手指拖动歌词后  if (mLrcViewListener != null && cb) {  //回调onLrcSeeked方法，将音乐播放器播放的位置移动到高亮歌词的位置  mLrcViewListener.onLrcSeeked(position, lrcRow);  }  }  private float mLastMotionY;  /\*\*  \* 第一个手指的坐标  \*\*/  private PointF mPointerOneLastMotion = new PointF();  /\*\*  \* 第二个手指的坐标  \*\*/  private PointF mPointerTwoLastMotion = new PointF();  /\*\*  \* 是否是第一次移动，当一个手指按下后开始移动的时候，设置为true,  \* 当第二个手指按下的时候，即两个手指同时移动的时候，设置为false  \*/  private boolean mIsFirstMove = false;  @Override  public boolean onTouchEvent(MotionEvent event) {  if (mLrcRows == null || mLrcRows.size() == 0) {  return super.onTouchEvent(event);  }  switch (event.getAction()) {  //手指按下  case MotionEvent.ACTION\_DOWN:  Log.d(TAG, "down,mLastMotionY:" + mLastMotionY);  mLastMotionY = event.getY();  mIsFirstMove = true;  invalidate();  break;  //手指移动  case MotionEvent.ACTION\_MOVE:  if (event.getPointerCount() == 2) {  Log.d(TAG, "two move");  doScale(event);  return true;  }  Log.d(TAG, "one move");  // single pointer mode ,seek  //如果是双指同时按下，进行歌词大小缩放，抬起其中一个手指，另外一个手指不离开屏幕地移动的话，不做任何处理  if (mDisplayMode == DISPLAY\_MODE\_SCALE) {  //if scaling but pointer become not two ,do nothing.  return true;  }  //如果一个手指按下，在屏幕上移动的话，拖动歌词上下  doSeek(event);  break;  case MotionEvent.ACTION\_CANCEL:  //手指抬起  case MotionEvent.ACTION\_UP:  if (mDisplayMode == DISPLAY\_MODE\_SEEK) {  //高亮手指抬起时的歌词并播放从该句歌词开始播放  seekLrc(mHignlightRow, true);  }  mDisplayMode = DISPLAY\_MODE\_NORMAL;  invalidate();  break;  }  return true;  }  /\*\*  \* 处理双指在屏幕移动时的，歌词大小缩放  \*/  private void doScale(MotionEvent event) {  //如果歌词的模式为：拖动歌词模式  if (mDisplayMode == DISPLAY\_MODE\_SEEK) {  //如果是单指按下，在进行歌词上下滚动，然后按下另外一个手指，则把歌词模式从 拖动歌词模式 变为 缩放歌词模式  mDisplayMode = DISPLAY\_MODE\_SCALE;  Log.d(TAG, "change mode from DISPLAY\_MODE\_SEEK to DISPLAY\_MODE\_SCALE");  return;  }  // two pointer mode , scale font  if (mIsFirstMove) {  mDisplayMode = DISPLAY\_MODE\_SCALE;  invalidate();  mIsFirstMove = false;  //两个手指的x坐标和y坐标  setTwoPointerLocation(event);  }  //获取歌词大小要缩放的比例  int scaleSize = getScale(event);  Log.d(TAG, "scaleSize:" + scaleSize);  //如果缩放大小不等于0，进行缩放，重绘LrcView  if (scaleSize != 0) {  setNewFontSize(scaleSize);  invalidate();  }  setTwoPointerLocation(event);  }  /\*\*  \* 处理单指在屏幕移动时，歌词上下滚动  \*/  private void doSeek(MotionEvent event) {  float y = event.getY();//手指当前位置的y坐标  float offsetY = y - mLastMotionY; //第一次按下的y坐标和目前移动手指位置的y坐标之差  //如果移动距离小于10，不做任何处理  if (Math.abs(offsetY) < mMinSeekFiredOffset) {  return;  }  //将模式设置为拖动歌词模式  mDisplayMode = DISPLAY\_MODE\_SEEK;  int rowOffset = Math.abs((int) offsetY / mLrcFontSize); //歌词要滚动的行数  Log.d(TAG, "move to new hightlightrow : " + mHignlightRow + " offsetY: " + offsetY + " rowOffset:" + rowOffset);  if (offsetY < 0) {  //手指向上移动，歌词向下滚动  mHignlightRow += rowOffset;//设置要高亮的歌词为 当前高亮歌词 向下滚动rowOffset行后的歌词  } else if (offsetY > 0) {  //手指向下移动，歌词向上滚动  mHignlightRow -= rowOffset;//设置要高亮的歌词为 当前高亮歌词 向上滚动rowOffset行后的歌词  }  //设置要高亮的歌词为0和mHignlightRow中的较大值，即如果mHignlightRow < 0，mHignlightRow=0  mHignlightRow = Math.max(0, mHignlightRow);  //设置要高亮的歌词为0和mHignlightRow中的较小值，即如果mHignlight > RowmLrcRows.size()-1，mHignlightRow=mLrcRows.size()-1  mHignlightRow = Math.min(mHignlightRow, mLrcRows.size() - 1);  //如果歌词要滚动的行数大于0，则重画LrcView  if (rowOffset > 0) {  mLastMotionY = y;  invalidate();  }  }  /\*\*  \* 设置当前两个手指的x坐标和y坐标  \*/  private void setTwoPointerLocation(MotionEvent event) {  mPointerOneLastMotion.x = event.getX(0);  mPointerOneLastMotion.y = event.getY(0);  mPointerTwoLastMotion.x = event.getX(1);  mPointerTwoLastMotion.y = event.getY(1);  }  /\*\*  \* 设置缩放后的字体大小  \*/  private void setNewFontSize(int scaleSize) {  //设置歌词缩放后的的最新字体大小  mLrcFontSize += scaleSize;  mLrcFontSize = Math.max(mLrcFontSize, mMinLrcFontSize);  mLrcFontSize = Math.min(mLrcFontSize, mMaxLrcFontSize);  //设置显示高亮的那句歌词的时间最新字体大小  mSeekLineTextSize += scaleSize;  mSeekLineTextSize = Math.max(mSeekLineTextSize, mMinSeekLineTextSize);  mSeekLineTextSize = Math.min(mSeekLineTextSize, mMaxSeekLineTextSize);  }  /\*\*  \* 获取歌词大小要缩放的比例  \*/  private int getScale(MotionEvent event) {  Log.d(TAG, "scaleSize getScale");  float x0 = event.getX(0);  float y0 = event.getY(0);  float x1 = event.getX(1);  float y1 = event.getY(1);  float maxOffset = 0; // max offset between x or y axis,used to decide scale size  boolean zoomin = false;  //第一次双指之间的x坐标的差距  float oldXOffset = Math.abs(mPointerOneLastMotion.x - mPointerTwoLastMotion.x);  //第二次双指之间的x坐标的差距  float newXoffset = Math.abs(x1 - x0);  //第一次双指之间的y坐标的差距  float oldYOffset = Math.abs(mPointerOneLastMotion.y - mPointerTwoLastMotion.y);  //第二次双指之间的y坐标的差距  float newYoffset = Math.abs(y1 - y0);  //双指移动之后，判断双指之间移动的最大差距  maxOffset = Math.max(Math.abs(newXoffset - oldXOffset), Math.abs(newYoffset - oldYOffset));  //如果x坐标移动的多一些  if (maxOffset == Math.abs(newXoffset - oldXOffset)) {  //如果第二次双指之间的x坐标的差距大于第一次双指之间的x坐标的差距则是放大，反之则缩小  zoomin = newXoffset > oldXOffset ? true : false;  }  //如果y坐标移动的多一些  else {  //如果第二次双指之间的y坐标的差距大于第一次双指之间的y坐标的差距则是放大，反之则缩小  zoomin = newYoffset > oldYOffset ? true : false;  }  Log.d(TAG, "scaleSize maxOffset:" + maxOffset);  if (zoomin) {  return (int) (maxOffset / 10);//放大双指之间移动的最大差距的1/10  } else {  return -(int) (maxOffset / 10);//缩小双指之间移动的最大差距的1/10  }  }  /\*\*  \* 设置歌词行集合  \* @param lrcRows  \*/  public void setLrc(List lrcRows) {  mLrcRows = lrcRows;  invalidate();  }  /\*\*  \* 播放的时候调用该方法滚动歌词，高亮正在播放的那句歌词  \* @param time  \*/  public void seekLrcToTime(long time) {  if (mLrcRows == null || mLrcRows.size() == 0) {  return;  }  if (mDisplayMode != DISPLAY\_MODE\_NORMAL) {  return;  }  Log.d(TAG, "seekLrcToTime:" + time);  for (int i = 0; i < mLrcRows.size(); i++) {  LrcRow current = mLrcRows.get(i);  LrcRow next = i + 1 == mLrcRows.size() ? null : mLrcRows.get(i + 1);  /\*\*  \* 正在播放的时间大于current行的歌词的时间而小于next行歌词的时间， 设置要高亮的行为current行  \* 正在播放的时间大于current行的歌词，而current行为最后一句歌词时，设置要高亮的行为current行  \*/  if ((time >= current.time && next != null && time < next.time)  || (time > current.time && next == null)){  seekLrc(i, false);  return;  }  }  }  } |

注释：

分以下三步来绘制歌词

第1步：高亮地画出正在播放的那句歌词

第2步：画出正在播放的那句歌词的上面可以展示出来的歌词

第3步：画出正在播放的那句歌词的下面的可以展示出来的歌词

高亮地画出正在要高亮的的那句歌词

|  |
| --- |
| String name = getName(R.string.myDisplayName);  String date = getDate();  String msgText = getText();  int RIdA = R.layout.chat\_me;  ChatMsg newMsg = new ChatMsg(name, date, msgText, RIdA);  list.add(newMsg);  int RIdB = R.layout.chat\_he;  String othername = getName(R.string.otherDisplayName);  ChatMsg backMsg = new ChatMsg(othername, date, "自动回复(hello!)", RIdB);  list.add(backMsg);  talkView.setAdapter(new ChatAdapter(ChatActivity.this, list));  messageText.setText("");  }  });  }  private String getName(int id) {  return getResources().getString(id);  }  private String getDate() {  SimpleDateFormat sdf = new SimpleDateFormat("MM-dd HH:mm");  Date d = new Date();  return sdf.format(d);  }  private String getText() {  return messageText.getText().toString();  } |

# 

# 

# 第二章 系统测试

## 2.1编写目的

编写该测试报告主要由以下几个目的

1.通过对测试结果的分析，得到对软件质量的评价

2.分析测试的过程，产品，资源，信息，为以后制定测试计划提供参考

3.评估测试执行和测试计划是否符合

4.分析系统存在的缺陷，为修复和预防bug提供建议

## 2.2背景说明

说明：

1.被测试软件系统的名称：android音乐播放器

2.该软件的任务提出者：文军老师。

### 1.3定义

严重bug：出现以下缺陷，测试定义为严重bug

系统无响应，处于死机状态，需要其他人工修复系统才可复原。

点击某个按钮后出现“Unexpect error，the application has been stopped”或者返回异常错误。

进行某个操作后，出现“Unexpect error，the application has been stopped”或者返回异常错误。

当切换音乐时，出现”Unexpect error，the application has been stopped”或者返回异常错误。

## 1.4 参考资料

列出要用到的参考资料，如：

1．《android需求和实际和说明书》

2．《android项目数据字典》

3．《android后台管理系统测试计划》

4.《android项目计划》

5.《android程序设计基础》

### 1.4.1测试

测试概要Android音乐播放器系统测试从2014年5月25日开始到2014年6月1日结共持续6天，测试功能点6个，执行10个测试用例，平均每个功能点执行测试用例2个，测试共发现5个bug，其中严重级别的1个，无效1个，平均每个测试功能点1个bug。

### 1.4.2测试结果及发现

1.4.2.1.测试1（功能键测试）

  在本次测试中对各个功能键进行了相关的测试，并把各个功能键该有的功能给体现出来。最后的测试结果是，各个功能键基本符合预想的要求，但是在测试中间，不时会出现一些系统错误。

1.4.2.2.测试2（音乐清单测试）

在对音乐清单模块进行测试时，先了解音乐清单的具体功能的体现与要求。音乐清单模块具备自动扫描功能，自动更新，删除重复，删除错误功能。测试过程比较繁琐，不停更换音乐，增加重复音乐，增加错误来对该项进行测试。对音乐清单界面转变，字体等还需改进。

### 1.4.3.缺陷和限制

1．缺陷描述：音乐清单有乱码，音乐无名称，查看不方便

  缺陷影响：其他音乐都有名称，音乐无名称，查看不方便

  推迟原因：目前的日志为了调试方便，显示了很多其它信息，在项目正式发布时会统一处理的。

2．缺陷描述：数据字典种类修改，默认值设置后，在调用该数据字典种类的数据字典，默认值无显示

 缺陷影响：数据字典种类的默认值设置后，不能显示设置的默认值，相当于数据字典种类默认值设置功能未实现

推迟原因：该功能暂时不好实现，需要和和系统的默认语种一起处理。

3.缺陷描述：多媒体添加，文件上传功能未实现

 缺陷影响：文件上传功能未实现

推迟原因：该功能暂时不好完成，在下个版本中完成

4.建议

  在项目开始的时候应该制定编码标准，数据库标准，需求变更标准，开发和测试人员都严格按照标准进行，可以在后期减少因为开发，测试不一致而导致的问题同时也可以降低沟通成本。

发布版本的时候，正确布置测试环境，减少因为测试环境，测试数据库数据的

问题而出现的无效bug。开发人员解决bug的时候，填写bug原因以及解决方式，方便bug的跟踪。开发人员在开发版本上发现bug，可以通知测试人员，因为开发人员发现的bug，很有可能在测试版本上出现，而测试人员和开发人员的思路不同，有可能测试人员没有发现该bug，而且，这样可以保证发现的bug都能够被跟踪。

5．评价

  本软件经测试，可以在任何android设备上运行，安全性得到了保证，可以交付使用。

# 第三章 知识技能学习情况

## 3.1.几大类的学习与掌握

3.1.1.Activity —— 应用表示层（基类Activity）

应用程序中的每个屏幕都是通过继承和扩展基类Activity来实现的。同一应用中的每个Activity是相互独立的。程序启动后显示的第一幅画面是应用程序的第一个Activity（默认窗口），而后可以根据需要从这个Activity启动另一个新的Activity。

Activity利用View来实现应用中的GUI（用户直接通过GUI和应用程序做交互）。Activity窗口内的可见内容通过基类View提供。使用Activity.setContentView()方法设置当前Activity中的View对象。

3.1.2.Service —— 没有可见的用户界面，但能够长时间运行于后台（基类Service）

运行于应用程序进程的主线程中，因此Service不会阻塞其他组件和用户界面。 Service是不能自己启动的，必须通过Context对象（如一个Activity）调用startService或bindService方法来启动（用这两种方法启动的Service的生命周期不同）。

3.1.3.Broadcast Receiver —— 用户接收广播通知的组件（基类BroadcastReceiver）

Android中的广播要么来自于系统，要么来自普通应用程序。 很多事件都可能导致系统广播，如手机所在时区发生变化，电池电量低，用户改变系统语言设置等.

3.1.4.Content Provider —— 为解决应用程序间数据通信、共享的问题（基类ContentProvider）

在Android中，每个应用程序都是用自己的用户ID并在自己的进程中运行。这样的好处是，可以有效地保护系统及应用程序，避免被其他不正常德应用程序所影响，每个进程都拥有独立的进程地址空间和虚拟空间。

Content Provider可以将应用程序特定的数据提供给另一个应用程序使用。其数据存储方式可以是Android文件系统、SQLite数据库或者其他合理的方式。

当数据需要在应用程序间共享时，我们就可以利用ContentProvider为数据定义一个URI。之后，其他应用程序对数据进行查询或者修改时，只需要从当前上下文对象获得一个ContentResolver, 然后传入响应的URI就可以了。

3.1.5.Intent —— 连接组件的纽带

以上4种基本组件中，除了Content Provider是通过Content Resolver激活外，其他3种组件Activity、Service和Broadcast Receiver都是由Intent异步消息激活的。

Intent在不同的组件之间传递消息，将一个组件的请求意图传给另一个组件。因此，Intent是包含具体请求信息的对象。

Intent是一种运行时绑定（runtime binding）机制，它能够在程序运行的过程中连接两个不同的组件。通过Intent，你的程序可以向Android表到某种请求或者意愿，Android会根据意愿的内容选择适当的组件来处理请求。

## 3.2.xml的学习与掌握

在安卓工程中，大量使用了XML文件，比如界面布局文件，字符资源文件，风格style文件，简单图形文件等。

比如在界面布局文件中，通过指定控件的排列方式、间隔像素、对齐方式等，达到对页面进行布局的目的，同时使用特定编辑器，实时解析布局文件，到达所见即所得的目的。

在字符资源中，通过关键字编辑字符内容，这样可以在不修改代码的情况下，修改程序内容。同时，考虑多国语言的使用，更加容易替换字符内容，所以我们建议在代码的编写过程中，尽量少地在程序中使用特定语言字符，而是在字符资源文件中编写。

在风格文件中，通过特定关键字，可以使界面呈现统一风格，同时减少重复代码的编写，减轻工作量。图形XML文件中可以实现一些矩形、椭圆形等图形文件，可以指定颜色和渐变颜色范围。

# 

# 第四章 分工协作与交流情况

付鼎：功能设计：登录，注册，推荐/在线播放歌曲

ppt制作

后台服务器端总结：技能学习情况，方案设计与实现

耿睿：UI界面制作，真机适配，功能跳转逻辑实现，界面测试及优化

安卓端界面总结：技能学习情况，方案设计与实现

龚子成:功能设计：常用功能，歌曲评论，歌曲点赞

后台安卓端总结：技能学习情况，方案设计与实现

程弈茗：功能设计：聊天界面（分享歌曲），好友系统

安卓服务器端总结：技能学习情况，方案设计与实现

付鼎与龚子成负责后端服务器与数据库开发，程弈茗与耿睿实现前端功能与界面。

# 

# 参考文献

1. Lesaek A.Maciaszek（澳）.需求分析与系统设计: 44-76
2. 李刚.疯狂web讲义. （北京）：230-364

# 致谢

本报告的工作是在我的指导教师文军老师的悉心指导下完成的，感谢文军老师在项目过程中的帮助以及对数据库详细耐心的讲解。