TSZ Musicplayer 项目文档

https://github.com/tsz-musicplayer/TSZ_Musicplayer.git

技术小组 第 八 小组

小组成员 周敏、田益兰、苏雪莲

指导教师 龚伟

起止时间 2019 年 2 月 ～2019 年 6 月

重庆师范大学软件工程系

摘要

在互联网的时代中，随着手机、电脑的普及，4G 网络的成熟，各种音频资源在网络上愈加丰富。当今人们更需要各种各样的娱乐软件来帮助提升生活品质，音乐播放器在其中占据了不低的地位，如今许多播放器的功能过于繁杂，对于大多数用户来说不仅已形同虚设，还会占用内存，为避免这些弊端，选择开发一款以播放音乐为主的播放器，具有音乐播放器的常用功能，满足不需要繁杂功能用户的需求。

而在如今这个紧张且忙碌的社会环境中，听音乐是一种很好的解压方式，虽然，现如今的已具有很多成熟的音乐播放器，它们功能多样，但是，想拥有一个纯粹的音乐播放器却很是不易，MSZ Musicplayer 就是这样一款可以播放音乐的播放器，便于人们闲暇时欣赏音乐，它具有控制播放器播放、暂停、上一曲、下一曲，还具有播放器播放、下载、分类管理歌曲等基本功能且界面简单、操作简洁、界面美观。

目录

第1章 愿景文档

1.1. 问题陈述

1. 问题一

要素	描述
问题	无法处理短时间不想听但不想删除的某歌曲
影响	听众的体验
结果	可以设置固定时间某歌曲不再出现在播放列表中
优点	新系统有以下有点： ● 增加了歌曲某一段时间不再出现在播放列表中功能 ● 获取更好的听众体验

1.2. 涉众与用户

1. 涉众

涉众	涉众类型	简要描述
歌友	用户	使用系统听歌曲，交流歌曲的人
音乐人	用户	使用系统上传歌曲的人
运营人员	运营人员	维护系统运行
愿景者	发起人	提出愿景的人
客户	客户	系统完成后，为系统付出金钱的人
项目经理	开发团队	对项目的成功策划和成功执行负总责
软件架构师	开发团队	主导系统全局分析设计和实施，负责软件架构和关键技术决策的人
开发人员	开发团队	系统开发的专业人员
运维人员	维护人员	系统交互后的维护工作
测试人员	开发团队	系统测试的专业人员
美工团队	开发团队	设计系统的用户界面
法律专家	权威人士	具有音乐法律专业的知识

音乐顾问	权威人士	提供判定音乐合格准则
投资人	执行资助人	给项目提供资金

2. 用户

用户	用户类型	简要描述
歌友	歌迷、发烧友、歌友、路人	系统的直接使用者，系统面向的大众用户
音乐人	网络歌手、职业歌手	系统的直接使用者，系统面向的具有一定特殊权限的用户
审核员	音乐鉴别师、审核人员	系统的直接使用者，负责维护系统的正常运作
管理员	反馈收集员	系统的直接使用者，负责系统运作中的管理工作
小编	小编	系统的直接使用者，负责系统运作中文案编辑的工作

1.3. 关键涉众和用户需求

1.从听众的角度

- 音乐播放器能提供优秀的音乐播放功能（支持.mp3.mp4 等格式的文件播放）
- 操作方便
- 界面简洁、直观、不夸张
- 在有需要听歌曲时，可以快速的获取歌曲并播放
- 可以按自己的喜好分类歌曲
- 如果不喜欢的歌曲可以自由的删除
- 方便上网搜索自己喜欢的歌曲
- 可以自由下载喜欢的歌曲
- 如果没有网络，可以播放本地的歌曲

2.从管理人员的角度

- 操作方便
- 界面简洁
- 有一套完善的管理体系
- 有很强的可兼容性
- 安全
- 管理方法容易学习

1.4. 产品概述

1. 产品定位陈述

for	大众
who	讨厌功能复杂的人播放、下载音乐
the	是属于系统软件
That	能够满足各个年龄段的听众对音乐喜爱的人群集结

2. 完整的产品概述

音乐播放器是一款华丽的音乐播放软件，操作简单，包含本地音乐和在线音乐两层功能，极佳的音质音效、支持随时随地播放、搜索、下载歌曲，并且能自动匹配歌词和专辑图片。

音乐播放器且提供方便的控制歌曲播放、暂停、停止、上一曲、下一曲功能。

一大功能是将线上歌曲下载到本地，就能在无网络时享受优美的音乐

另一大特点是文件夹模式的歌曲播放功能，还可以让你很轻松的编辑管理歌曲列表，可以轻松地管理你的歌曲，音乐播放器可以轻松找到手机中的所有歌曲

1.5. 特性

1. 系统必须支持播放本地音乐
2. 系统必须支持播放在线音乐
3. 系统必须支持暂停播放
4. 系统必须支持下载音乐
5. 系统应该支持选择播放模式
6. 系统应该支持定时停止播放
7. 系统应该支持歌曲进度条的显示
8. 系统应该支持歌词实时显示
9. 系统应该能够控制音乐音量的大小
10. 系统应该支持根据自己喜好创建各种音乐清单
11. 系统可以自定义设置背景

1.6. 其他产品需求

1. **易用性**：操作简单，界面整洁，可提供多种播放模式和舒适的平台
2. **性能**：需要足够的存储空间。
3. **可支持性**：需要有播放设备。
4. 其他需求

1.1. 适应的标准

(1)歌曲符合相关版权保护法，上传歌曲有一定质量，否则将会被屏蔽

(3)禁止使用敏感词汇

(4)只有在保证网络良好的情况下才能进行通信

1.2. 系统需求

(1)某些功能必须在网络环境良好的情况下才能正常使用，比如在线播放音乐。

(2)播放音乐时需要有音响等相应的播放设备

1.3. 许可、安全性与安装

(1)所涉及的歌曲符合国家相关法律规定，禁止反社会，反共，传播黄赌毒等其他不良信息的音乐。

(2)安装允许读写硬盘权限。

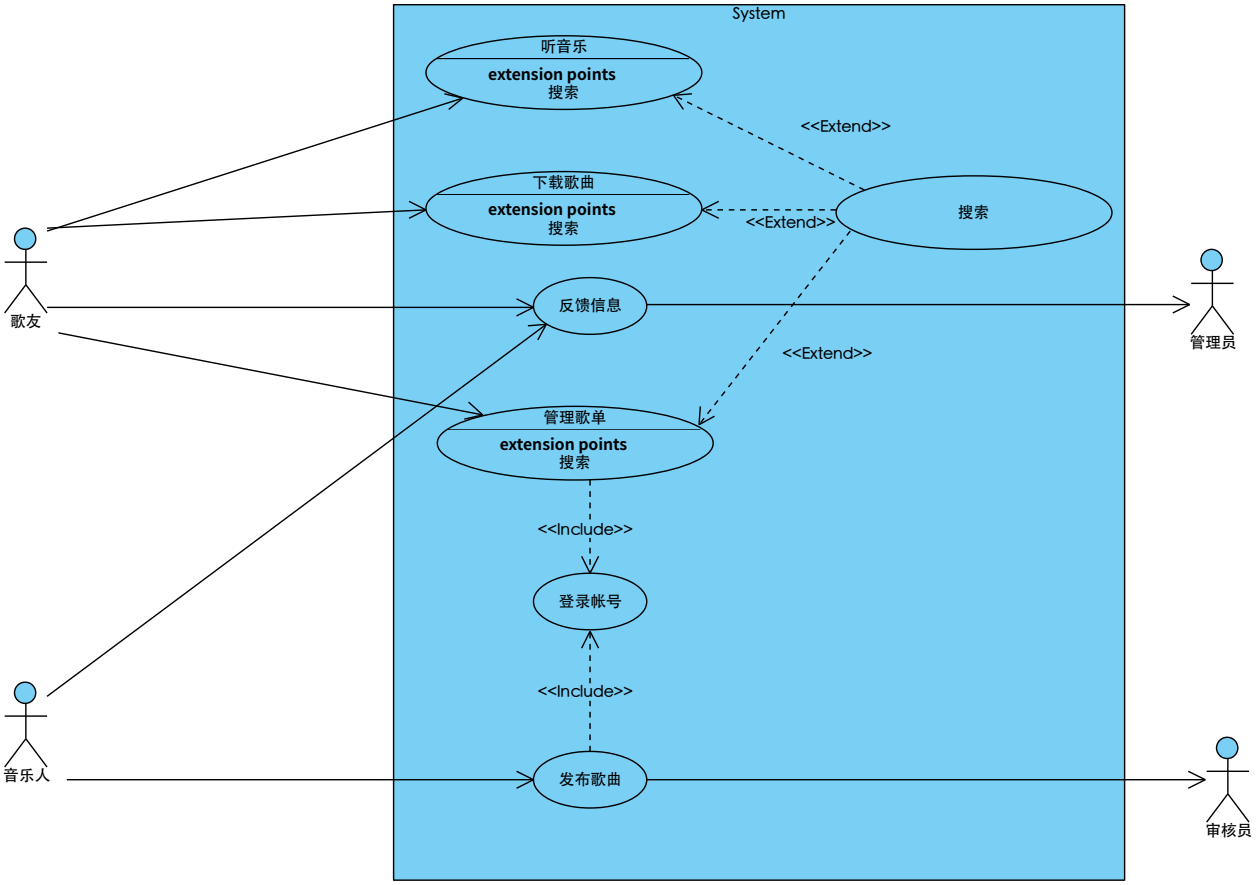
第2章 用况模型

2.1. 术语表

名称	定义
歌友	使用系统听音乐的人
目标歌单	被歌友选中的歌单
目标歌曲	被歌友选中的歌曲
在线歌单	保存在系统中的歌曲列表
搜索歌单	系统搜索生成的临时歌单
我喜欢的音乐	一个歌单，用来保存被标记为我喜欢的音乐的歌曲列表
歌曲信息	歌曲信息包括: 1. 音乐标题 2. 歌手 3. 歌曲专辑 4. 歌曲时长 5. 歌曲是否已下载 6. 歌曲是否已加入我喜欢的音乐
播放列表	一个歌曲列表，系统以某种播放模式播放该列表内的音乐
播放模式	播放音乐的顺序规则，播放模式包括: 1. 单曲循环：当正在播放的该歌曲播放结束后，重新播放该歌曲 2. 顺序播放：当正在播放的该歌曲播放结束后，选择播放播放列表的该歌曲的下一首歌曲 3. 随机播放：当正在播放的该歌曲播放结束后，随意选择播放列表内的除该歌曲以外的歌曲。
播放进度	音乐已经播放的时间长度。
关键词	输入的目标字段
继续播放	继续接着上一次播放的停止的地方继续播放

暂停播放	停止播放
播放上一曲	播放当前歌曲的上一曲
播放下一曲	播放当前歌曲的下一曲
临时歌单	临时生成显示，不会存储在系统中
实时歌词	当前正在播放的那一句歌词

2.2. TSZ Musicplayer 的主要用况

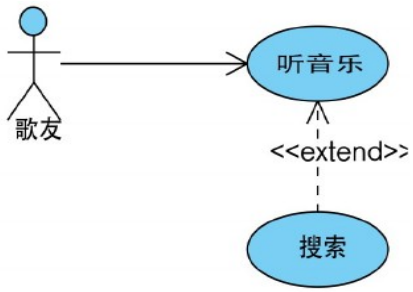


2.3. 用况描述——听音乐

1. 简要描述

歌友使用系统播放自己想要听的歌曲

2. 用况图



3. 前置条件

网络连接成功

4. 基本流

1. 参与者**歌友**进入系统后，开始执行用况

2. 系统读取并显示**在线歌单、播放列表**

{选择歌单}

3. 系统执行子流***选择歌单***

{选择歌曲}

4. 执行子流***选择歌曲***

{加载歌曲}

5. 执行子流***加载歌曲***

{播放歌曲}

6. 执行子流***播放歌曲***

7. 一首歌曲播放完毕后，系统按照**播放模式**（默认为按照列表顺序播放）选择下一首歌曲设置为**目标歌曲**

8. 用况回到**{加载歌曲}**处

9. 执行子流***停止听音乐***

{用况结束}

10. 用况结束

5. 备选流

5.1 错误处理

5.1.1 *网络未连接*

在**{选择歌单}**处，若网络未连接，则

系统提醒歌友网络未连接

用况恢复到**{用况结束}**

5.1.1 *歌曲不存在*

在**{选择歌曲}**处，若**目标歌单**为空，则

系统提醒歌友**目标歌单**为空，歌曲不存在

用况恢复到**{用况结束}**

6. 子流

6.1 *选择歌单*

1. 如果**歌友**选择了一个**歌单，**则：

a. 系统将该歌单标记为**目标歌单**（被选中的歌单）

b. 系统读取**目标歌单**（被**歌友**选中的**歌单**）的歌曲列表及**歌曲信息**（歌名、歌手、歌曲所属专辑、歌曲时长、歌曲是否下载、歌曲是非被标记为**我喜欢的音乐**）

c. 系统显示**目标歌单**及**歌曲信息**

2. 如果**歌友**选择**搜索歌曲**

a. 系统执行被扩展用况***搜索***

d. 系统将该**搜索歌单**标记为**目标歌单**

e. 系统读取**目标歌单**（被**歌友**选中的**歌单**）的歌曲列表及**歌曲信息**（歌名、歌手、歌曲所属专辑、歌曲时长、歌曲是否下载、歌曲是非被标记为**我喜欢的音乐**）

f. 系统显示**目标歌单**及**歌曲信息**

3. 如果**歌友**未选择任何**歌单**，则系统默认**歌友**选择的歌单**播放列表**

6.2 *选择歌曲*

1. 若**歌友**选择在系统显示的**目标歌单**中选择歌曲，则

a. 歌友在**目标歌单**中选择歌曲

b. 系统将被选择歌曲标记为**目标歌曲**

c. 若**目标歌单**为**临时歌单**，则系统将**目标歌曲**加入到**播放列表**

d. 若**目标歌单**不是**临时歌单**，则系统清空**播放列表**，将该歌单内的所有歌曲写入**播放列表**

e. 系统更新**播放列表**

2. 若**歌友**选择**继续播放**

a. 系统读取存储的上一次最后播放的歌曲

b. 系统将该歌曲设置为**目标歌曲**

c. 系统读取存储的歌曲、歌曲播放进度

3. 若**歌友**选择**播放上一曲，**则系统根据**播放列表**，找到正在播放歌曲的上一曲歌曲文件，将其设置为**目标歌曲**

4. 若**歌友**选择**播放下一曲，**则系统根据**播放列表**，找到正在播放歌曲的下一曲歌曲文件，将其设置为**目标歌曲**

6.3 *加载歌曲*

1. 系统读取**目标歌曲**（歌友选中播放的歌曲）音源文件、歌曲歌词文件、歌曲封面文件

2. 系统解码**目标歌曲**

6.4 *播放歌曲*

1. 系统播放**目标歌曲**

2. 系统高亮显示当前正在播放的歌曲，系统显示歌曲时长、播放歌曲的进度（以秒为单位，1s 更新一次）

3. 若**歌友**选择播放详情，则系统显示歌曲封面、显示**实时歌词**（当前音乐正在播放的那一句歌词）

4. 若歌友选择**暂停播放**：

a. 若**歌友**选择**继续播放**，则

① 系统读取存储的上一次最后播放的歌曲

② 系统将该歌曲设置为**目标歌曲**

③ 系统读取存储的**播放进度**及**歌曲相关信息**

b. 若**歌友**选择**播放上一曲**，则系统根据**播放列表**，找到正在播放歌曲的上一曲歌曲文件，将其设置为**目标歌曲**

c. 若**歌友**选择**播放下一曲**，则系统根据**播放列表**，找到正在播放歌曲的下一曲歌曲文件，将其设置为**目标歌曲**

6.5 *停止听音乐*

1. **歌友**选择**暂停播放**或者选择**退出系统**或者选择重新**选择歌单**播放歌曲
2. 若**歌友**选择退出系统，则系统保存**播放列表**
3. 系统将当前正在播放的歌曲、歌曲的播放进度保存记录

7. 后置条件

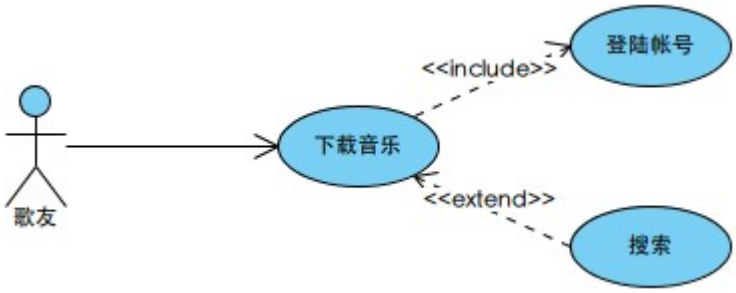
无

2.4. 用况描述——下载音乐

1. 简要描述

歌友使用系统下载自己想要听的歌曲，用于无网络的时候播放

2. 用况图



3. 前置条件

网络连接成功

4. 基本流

1. 参与者**歌友**进入系统后

{显示在线歌单}

2. 系统读取并显示**在线歌单**

{选择歌单}

3. **歌友**执行子流***选择歌单***

{选择歌曲}

4. **歌友**在**目标歌单**中选择歌曲
5. 系统将被选择歌曲标记为**目标歌曲**

{下载歌曲}

6. 系统执行包含用况***登录帐号***
7. 执行子流***下载歌曲***
8. 歌曲下载完毕，系统将所下载的歌曲放入**下载歌单列表**

{用况结束}

9. 用况结束

5. 备选流

1.1. 错误流

1.1.1. 网络未连接

在**{选择歌单}**处，若网络未连接，则

系统提醒歌友网络未连接

用况恢复到基本流的**{用况结束}**

1.1.2. 歌曲无下载权限

在**{下载歌曲}**处，若该歌曲无下载权限，则

系统提醒歌友无该音乐的下载权限

用况恢复到基本流的**{用况结束}**

在**{下载歌曲}**处，若该帐号没登陆，则

系统提醒歌友需要注册登录帐号

执行被包含用况**登录帐号**

用况恢复到**{下载歌曲}**

6. 子流

1.2. ***选择歌单***

1. 如果**歌友**选择了一个**在线歌单，**则：
   a. 系统将该歌单标记为**目标歌单**（被选中的歌单）
   b. 系统读取**目标歌单**（被**歌友**选中的**歌单**）的歌曲列表及**歌曲信息**（歌名、歌手、歌曲所属专辑、歌曲时长、歌曲是否下载、歌曲是非被标记为**我喜欢的音乐**）
   c. 系统显示**目标歌单**及**歌曲信息**

2. 如果**歌友**选择**搜索歌曲**
   a. 系统执行扩展用况***搜索***
   b. 系统将符合条件的歌曲记录下来，生成**临时歌单**（临时生成，不会永久存在）
   c. 系统将该**临时歌单**标记为**目标歌单**
   d. 系统读取**目标歌单**（被**歌友**选中的**歌单**）的歌曲列表及**歌曲信息**（歌名、歌手、歌曲所属专辑、歌曲时长、歌曲是否下载、歌曲是非被标记为**我喜欢的音乐**）
   e. 系统显示**目标歌单**及**歌曲信息**

3. 如果**歌友**未选择任何**歌单**，则系统显示默认的**歌单列表**

1.3. 下载歌曲

1. 系统下载**目标歌曲**

2. 系统生成临时的**下载列表**，显示当前正在下载的歌曲，系统显示歌曲下载的进度（以秒为单位，1s 更新一次）
   a. 若**歌友**选择**取消下载**
      1 系统读取当前存储的歌曲
      2 系统将该歌曲设置为**目标歌曲**
      3 系统将歌曲从**下载列表**中清除
      4 系统删除存储中歌曲已下载部分的内容
   b. 若**歌友**选择**暂停下载**，则
      1 系统存储的正在下载的歌曲
      2 系统将该歌曲设置为**目标歌曲**
      1 系统存储的歌曲歌曲信息（已下载的歌曲内容、歌曲下载进度）
   c. 若**歌友**选择**继续下载**，则
      2 系统读取存储的上一次正在下载的歌曲
      3 系统将该歌曲设置为**目标歌曲**
      4 系统读取存储的歌曲信息（已下载的歌曲内容、歌曲下载进度）
      5 用况恢复到基本流的**{下载歌曲}**处

7. 后置条件

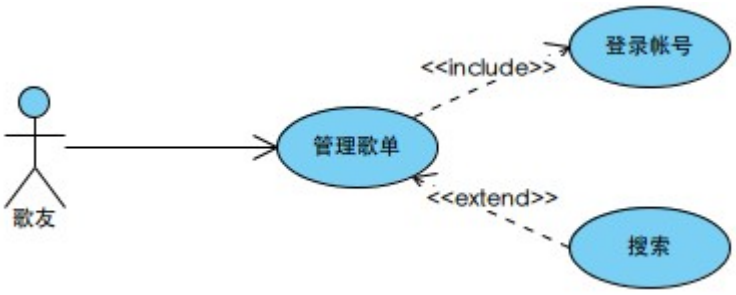
无

2.5. 用况描述——管理歌单

1. 简要描述

歌友使用系统分类管理自己喜欢的歌曲

2. 用况图



3. 前置条件

网络已连接

4. 基本流

1. 参与者**歌友**进入系统后，开始执行用况

{选择歌单}

2. 系统执行子流***选择歌单***

{操作歌单}

3. 系统执行包含用况***登录帐号***
4. 执行子流***操作歌单***

{用况结束}

5. 用况结束

5. 备选流

1.1. 错误处理

5.1.1 *歌曲不存在*

在**{选择歌单}**处

a. 若**目标歌曲**不存在，则
系统提醒歌友，该歌曲不存在
用况恢复到基本流的**{用况结束}**

6. 子流

1.2. 选择歌单

1. **歌友**需要选择一个**歌单**
   a. 若**歌单**未创建，则：
      1) 若**歌友**选择创建新歌单，则
         1 **歌友**输入新歌单名称
         2 **歌单**创建成功，系统给出提示
         3 系统将该歌单标记为**目标歌单**
      2) 若**歌友**选择不创建歌单，用况恢复到**{用况结束}**

b. 若**歌单**存在，则：

1 系统将该歌单标记为**目标歌单**（被选中的歌单）

2 系统读取**目标歌单**（被**歌友**选中的**歌单**）的歌曲列表及**歌曲信息**（歌名、歌手、歌曲所属专辑、歌曲时长、歌曲是否下载、歌曲是非被标记为**我喜欢的音乐**）

3 系统显示**目标歌单**及**歌曲信息**

1.3. 操作歌单

1. **歌友**管理**目标歌单**：

a. 若**歌友**选择**删除歌单**，则

1 系统询问歌友是否确定删除

2 歌友确认删除**目标歌单**

3 系统清除**目标歌单**和**目标歌单中的歌曲**

4 系统给出删除成功提示信息

b. 若**歌友**选择**修改歌单**，则

a) 若**歌友**选择修改歌单内容，则

(1) 若歌友选择搜索需要修改的歌曲，则

1 系统执行包含用况***搜索***

2 歌友选择中**搜索列表**中的歌曲

3 系统将该歌曲设置为**目标歌曲**（被选中的歌曲）

(2) 若歌友直接在**歌单列表**选择歌曲，则系统将该歌曲作为**目标歌曲**（被选中歌曲）

(3) 修改**目标歌曲**

1) 若歌友选择删除**目标歌曲**，则

1 系统给出提示确认歌友删除歌曲

2 系统删除歌曲，并给出删除成功提示信息

2) 若歌友需要添加**目标歌曲**，则

1 歌友选择需要添加歌曲的歌单

2 系统给出提示询问**歌友**确认添加信息

3 系统将其设置为**目标歌单**

4 系统将**目标歌曲**添加到**目标歌单**

5 系统更新**歌单列表**

6 系统给出添加成功提示信息

b) 若**歌友**选择修改歌单信息，则

(1) 系统获取**歌友**需要的**歌单信息**（歌单的名称）

(2) 系统修改**歌单信息**

(3) 系统更新**歌单列表**

(4) 系统给出修改成功提示信息

7. 后置条件

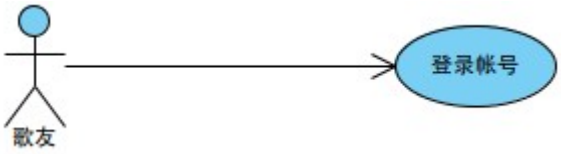
无

2.6. 用况描述——登录帐号

1. 简要描述

下载音乐、管理歌单用况包含了此用况，本用况用来验证歌友使用帐号、密码登录系统

2. 用况图



3. 前置条件

网络连接成功

4. 基本流

1．参与者**歌友**进入登录入口，开始执行用况

{登录帐号}

2．执行子流***登录帐号***

{用况结束}

3．用况终止

5. 备选流

1.1. 错误流

1.1.1．网络未连接

在**{信息验证}**处，若网络未连接，则

系统提醒歌友网络连接失败

用况恢复到**{用况结束}**

1.1.2. 在**{登录帐号}**处，未输入正确的帐号和密码

1. 若帐号和密码有误，则
   a. 系统通知该**歌友**所输入的帐号和密码有误
   b. 系统通知**歌友**重新输入
   c. 恢复到基本流的**{用况结束}**处
2. 若没帐号未注册，则
   a. 系统通知该**歌友**所输入的帐号未注册
   b. 系统提醒**歌友**需要注册帐号
   c. 恢复到基本流的**{用况结束}**处

6. 子流

1.2. 登录帐号

1．若**歌友**帐号已登录，则系统提示用户帐号已登录。

2．若**歌友**已注册帐号，则

1 系统提示**歌友**输入账号和密码

2 **歌友**输入账号和密码

3 系统检查输入的账号和密码与系统中存储的信息一致

4 用况恢复到{用况结束}

3．若**歌友**未注册帐号，则

(1) 系统提示歌友未注册帐号，询问是否需要注册新帐号

1) 若歌友注册新帐号，则

a. 系统提醒**歌友**输入个人信息（昵称、性别、邮箱和手机号（获取验证码）等）和密码

b. **歌友**按要求填写个人信息和密码

c. **歌友**完成注册

d. 系统给**歌友**提供**帐号**

e. 用况跳转到**{登录帐号}**

2) 若歌友不注册新帐号，则用况跳转到**{用况结束}**

7. 后置条件

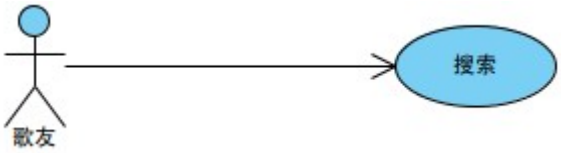
无

2.7. 用况描述——搜索

1. 简要描述

扩展下载音乐、管理歌单用况，该用况描述了歌友如何通过关键字收搜歌曲相关信息。

2. 用况图



3. 前置条件

4. 基本流

1. **歌友**进入搜索栏，开始执行用况

{输入关键字}

4. **歌友**输入想要搜索内容的**关键字**，一个关键字可以搜索出多种不同类别的信息（歌名、歌手、专辑、歌词、mv等）

{查找关键字}

2. 系统搜索符合**关键字**的歌曲

3. 系统根据**歌友**输入的**关键字**在系统中搜索与符合**关键字**相关的信息

4. 系统列表显示与关键字相关的信息

{用况结束}

5. 用况终止

5. 备选流

1.1. 错误处理

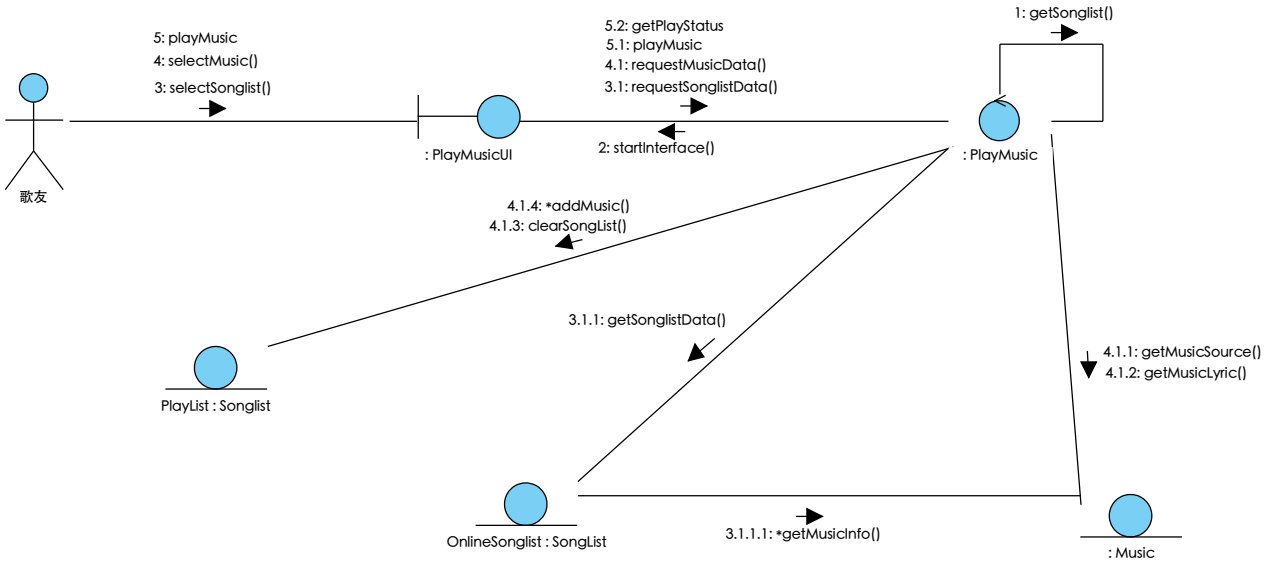
5.1.1 **歌曲***不存在*
在**{查找关键字}**处
b. 若系统中没有包含关键字的歌曲信息，则
系统提醒歌友，该没有与关键词相符的歌曲
用况恢复到基本流的**{输入关键字}**

6. 后置条件

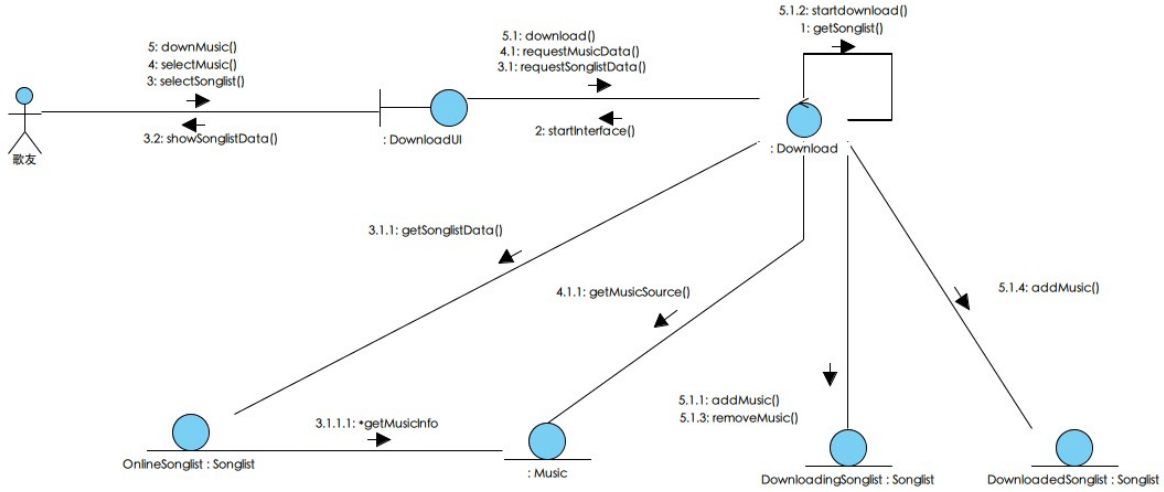
无

第3章 健壮性分析

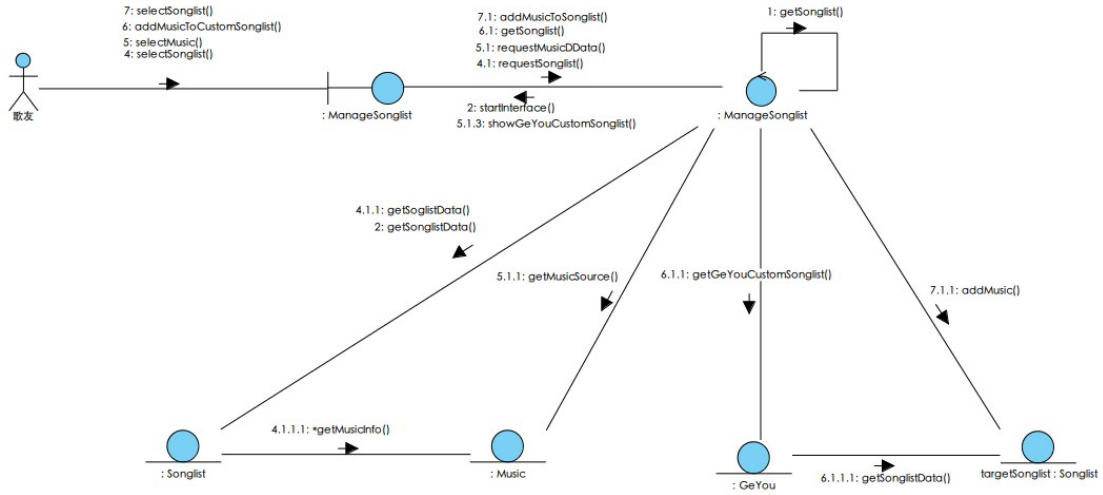
3.1. 通讯图-听音乐



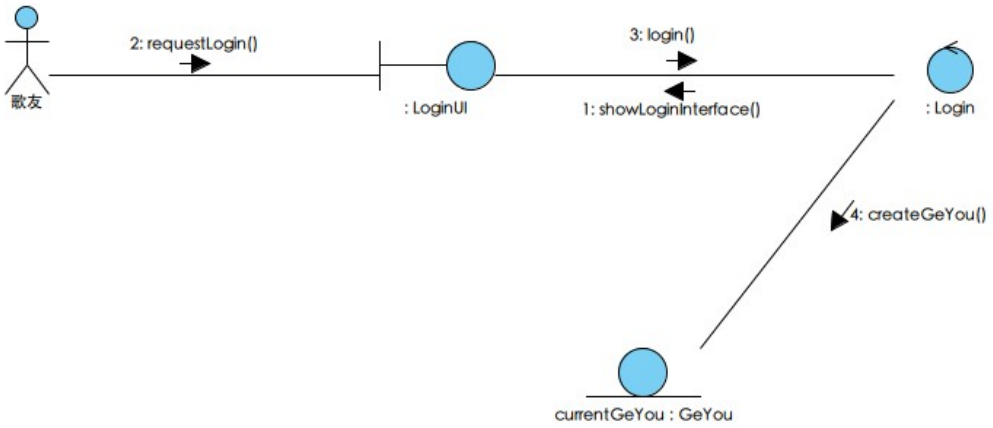
3.2. 通讯图-下载音乐



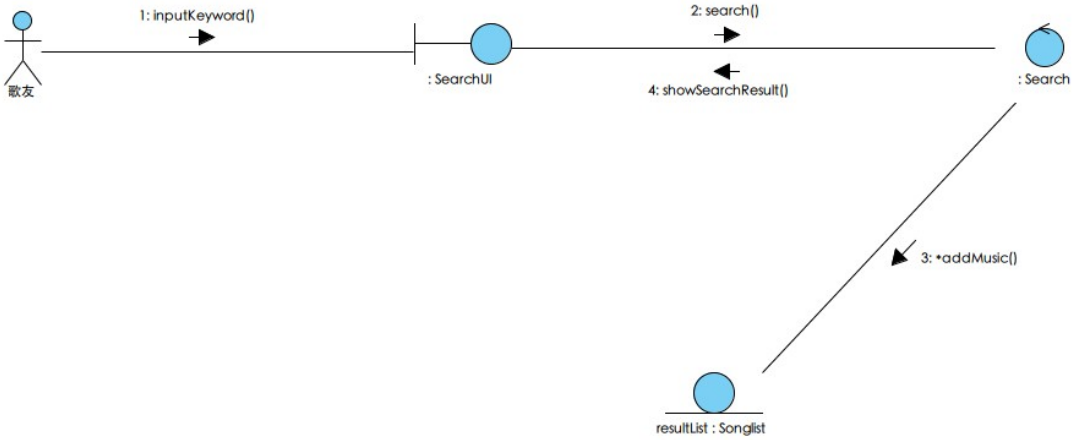
3.3. 通讯图-管理歌单



3.4. 通讯图-登录

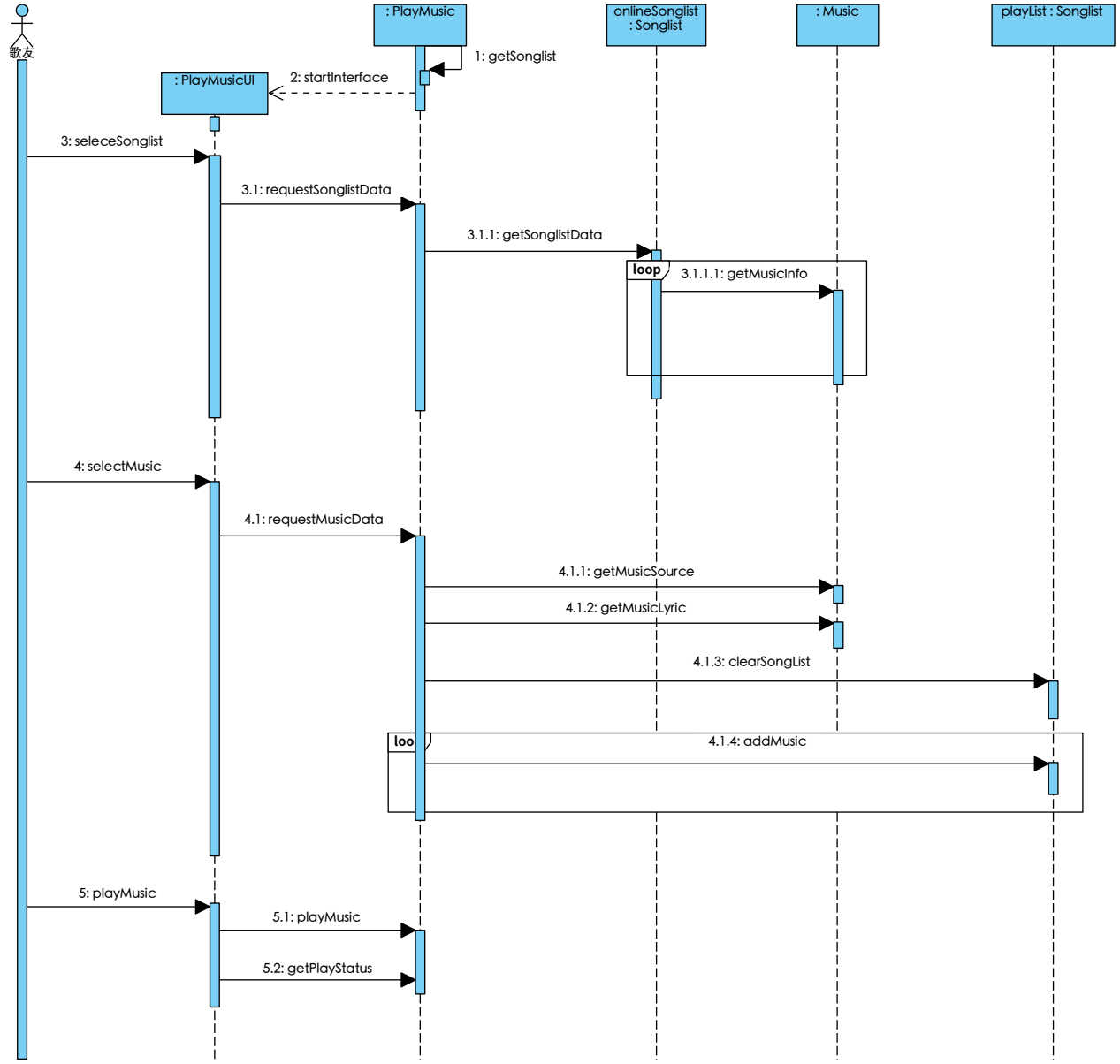


3.5. 通讯图-搜索

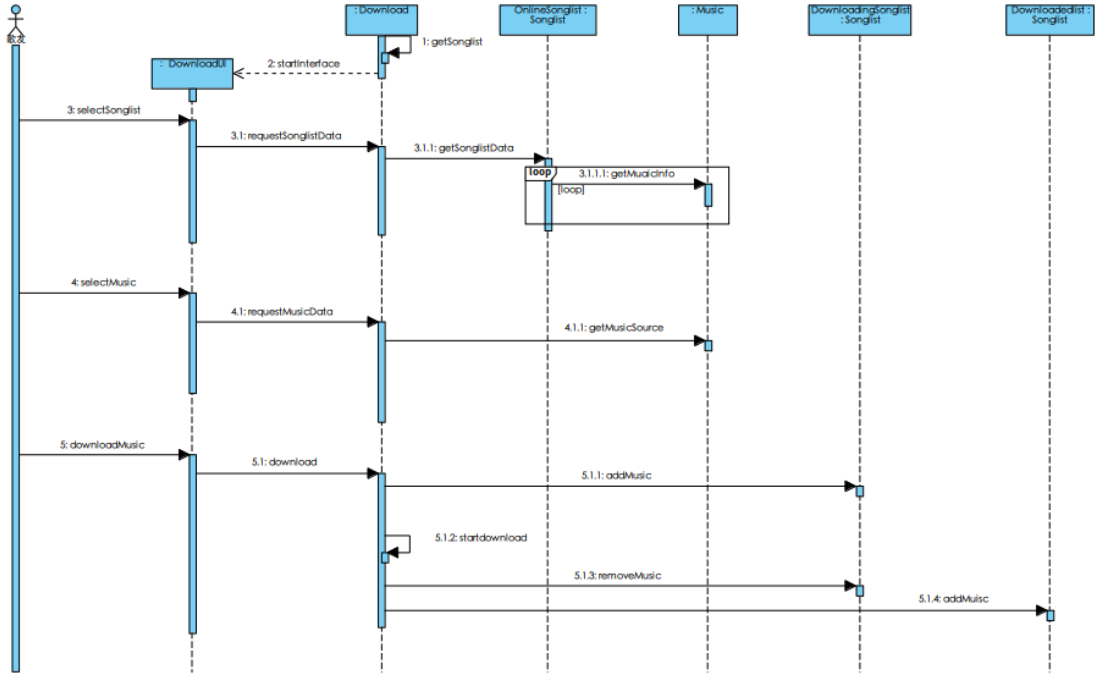


第4章 交互模型分析

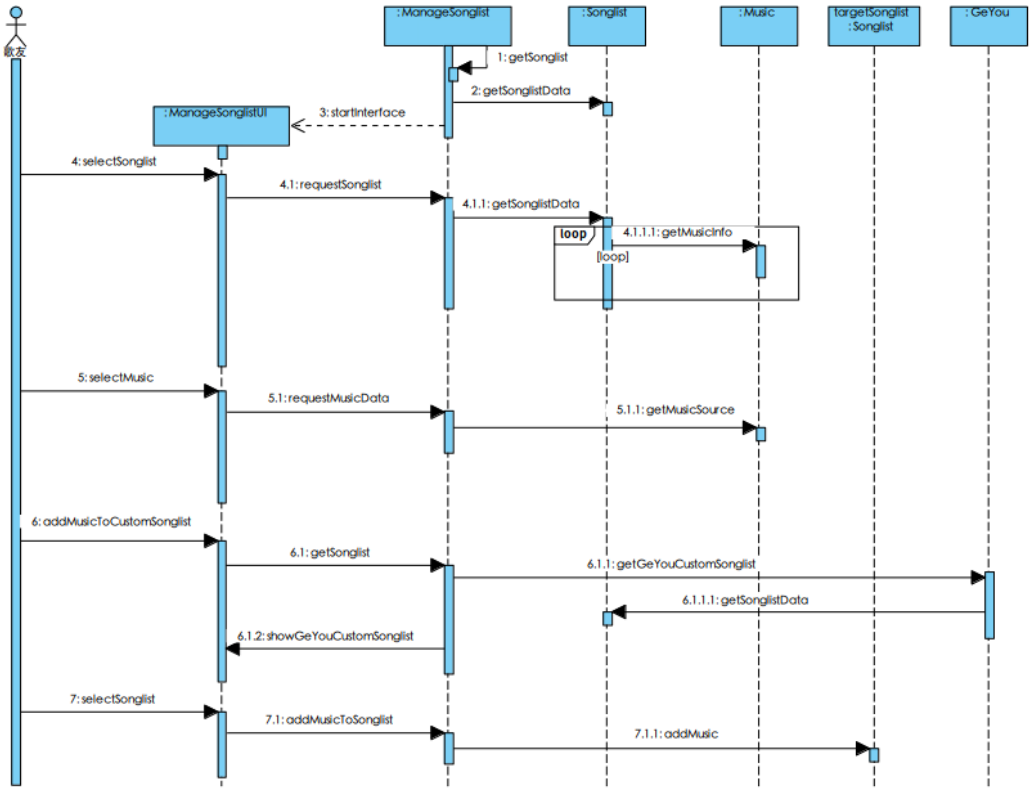
4.1. 顺序图-听音乐



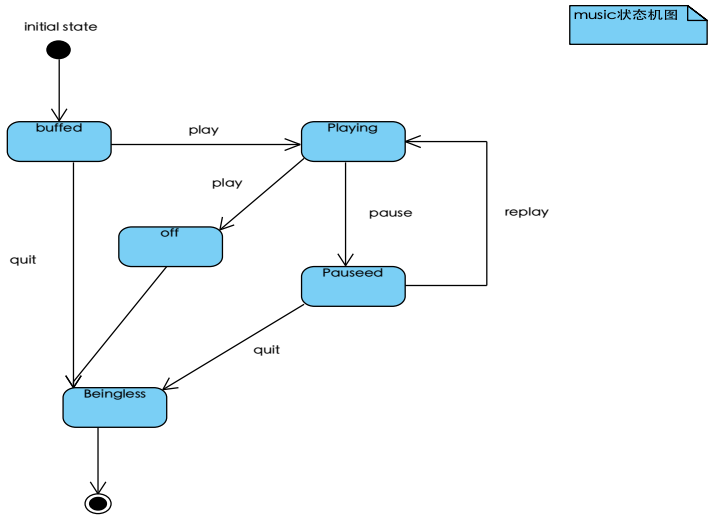
4.2. 顺序图-下载音乐

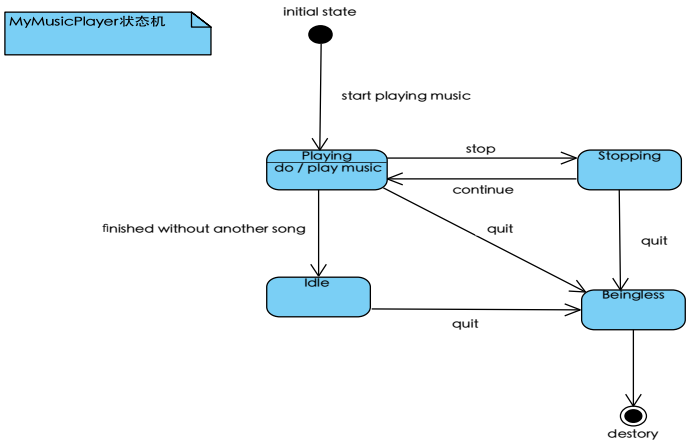


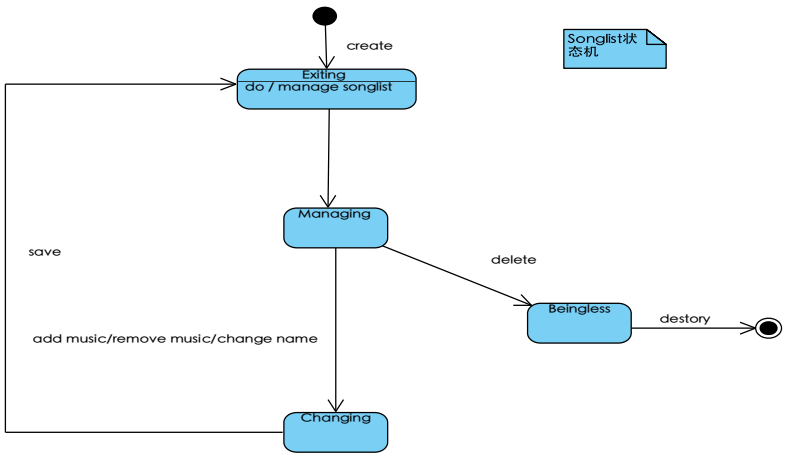
4.3. 顺序图-管理歌单



第5章 状态机分析

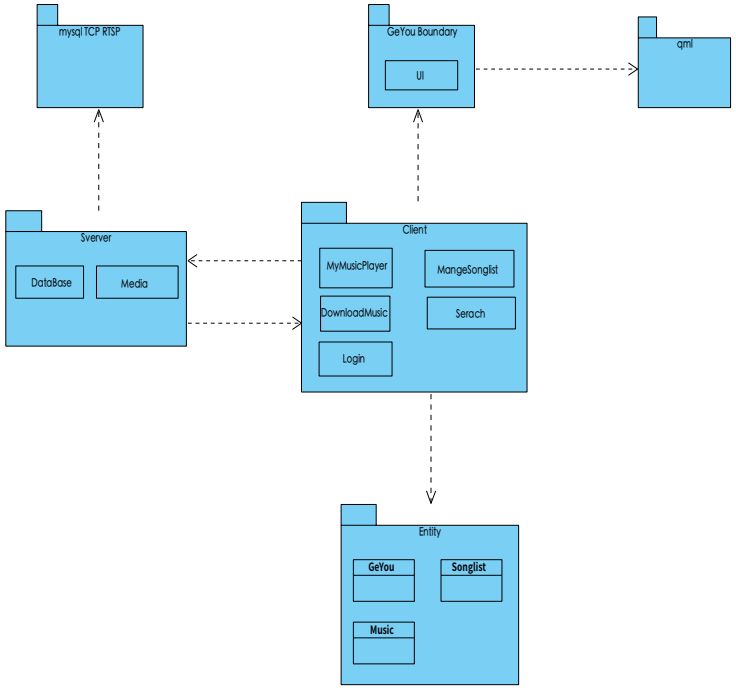

music状态机图
initial state
buffed
play
Playing
play
pause
replay
off
quit
Pauseed
quit
Beingless

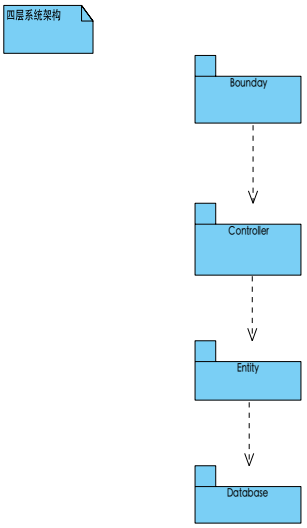

MyMusicPlayer状态机
initial state
start playing music
Playing
do / play music
stop
Stopping
continue
finished without another song
quit
quit
Idle
Beingless
quit
destory


create
Songlist状态机
Exiting
do / manage songlist
Managing
save
delete
Beingless
destory
add music/remove music/change name
Changing

第6章 系统结构设计

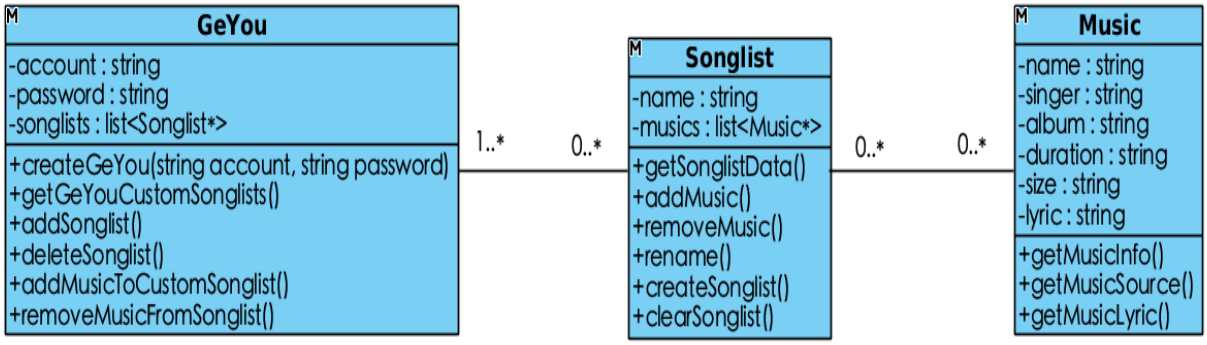
6.1. 系统架构包图



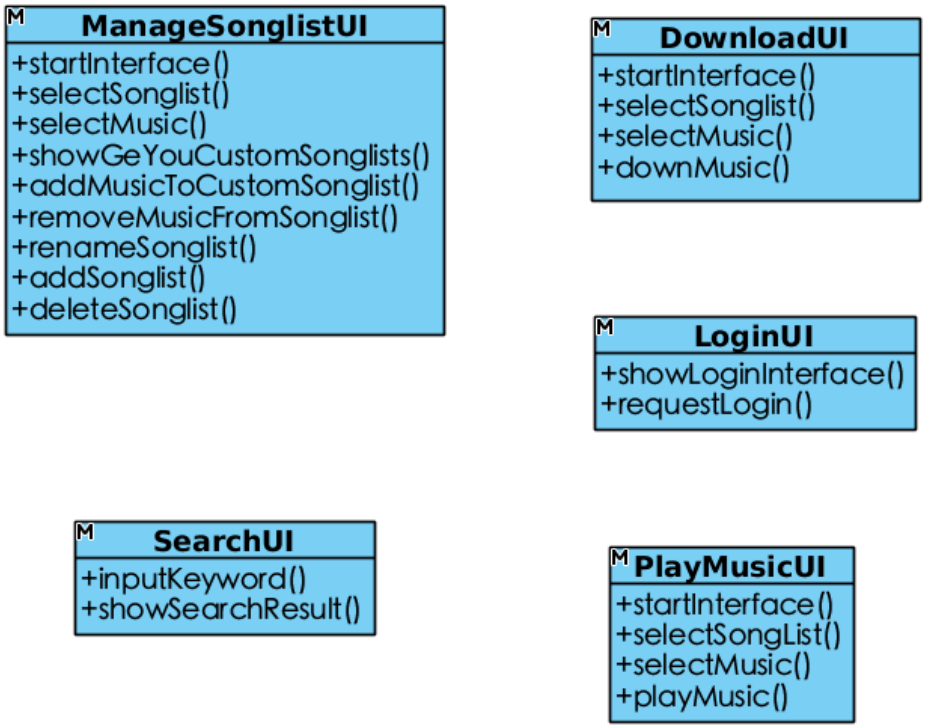


第7章 详细设计

7.1. 系统类模型



7.2. 边界类设计



第8章 数据库实现

8.1. 数据库表：

1. 系统歌单列表

存储了系统所有的歌曲的信息，歌曲的 path 是唯一的

Field	Type	Null	Key	Default	Extra
path	varchar(20)	YES		NULL	
name	char(20)	YES		NULL	
album	varchar(20)	YES		NULL	
duretion	char(20)	YES		NULL	
lyric	varchar(20)	YES		NULL	
size	varchar(20)	YES		NULL	

mysql代码：

```
create table Music(
       path varchar(20),
       name char(20),
       album varchar(20),
       duretion char(20),
       lyric varchar(20),
       size varchar(20)
);
```

2. 用户列表

存储了系统中所有注册账户的信息，账户 ID 是唯一的

Field	Type	Null	Key	Default	Extra
user_id	varchar(20)	YES		NULL	
password	char(20)	YES		NULL	

mysql代码：

```
create table Userinfo(
          user_id varchar(20),
          password char(20)
);
```

3. 歌单列表

存储了系统所有的歌曲的信息，歌单的列表名是唯一的

```
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| songlist | varchar(20) | YES  |     | NULL    |       |
+----------+------------+------+-----+---------+-------+
```

mysql代码：

```
INSERT INTO songListLab_name_lab
VALUES(
        'songList_name_list'
        );
```

4. 歌曲列表

存储了系统所有的歌曲的信息，歌曲的path是唯一的

```
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| path  | varchar(100) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

mysql代码：

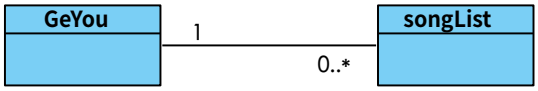
```
INSERT INTO songList_name_list
VALUES(
        'music_path'
);
```

8.2. 关联属性

一对多关联实现：

1. 一个歌友有多个歌单

歌单与用户：一个用户可以创建多个歌单



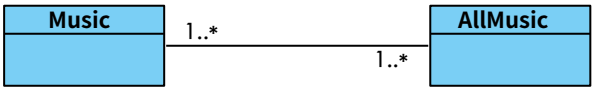
关联表

```
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| user_id  | varchar(20) | YES  |     | NULL    |       |
| password | char(20)   | YES  |     | NULL    |       |
+----------+------------+------+-----+---------+-------+
```

多对多关联实现：

2. 多个歌单对应多个在线歌曲

歌曲与歌单：一个歌曲可以被多个歌单收藏

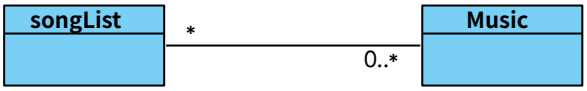


关联属性（music_path）

Field	Type	Null	Key	Default	Extra
path	varchar(20)	YES		NULL	
name	char(20)	YES		NULL	
album	varchar(20)	YES		NULL	
duretion	char(20)	YES		NULL	
lyric	varchar(20)	YES		NULL	
size	varchar(20)	YES		NULL	

3. 多个歌单有多个歌曲

歌曲与歌单：一个歌曲可以属于多个歌单



关联属性（songList）

Field	Type	Null	Key	Default	Extra
path	varchar(100)	YES		NULL	

第9章 实现

TSZ_MusicPlayer 使用了 C/S 架构,使用 QML 实现界面,C++实现逻辑,mysql 数据库实现数据的存储,使用 live555 流媒体服务器实现在线播放音乐,客户端 Client 将用户请求发送给服务器,服务器 Server 接收到消息后,判断消息请求,提供相关服务。

9.1. Json 数据传输

网络传输使用 json 文件交换数据

1、解析 json

解析使用 open()函数打开一个.json 文件,调用 Json::Read 对字符串进行解析,读取 Json::Value 将对象读取出来存储到 vecter 中。

```
//解析json
std::vector<stru_music> MyJson::analysisSonglistJson(const std::string file)
{
    std::cout<<"读取json文件: " << file <<std::endl;
    std::ifstream ifs;
    ifs.open(file);

    stru_music music;
    std::vector<stru_music> songlist;

    Json::Value jsonRoot;
    Json::Reader readerinfo;                                    'Reader'
    Json::Value root;
    if(!ifs.is_open()){
        std::cout << "open file error" << std::endl;
    }
    if (readerinfo.parse(ifs, jsonRoot)) {
        if (jsonRoot.isArray()) {
            int nArraySize = jsonRoot.size();    implicit conversion changes
            for (int i=0; i<nArraySize; i++) {
                music.name = jsonRoot[i]["name"].asString();
                music.singer = jsonRoot[i]["singer"].asString();
                music.album = jsonRoot[i]["album"].asString();
                music.duration = jsonRoot[i]["duration"].asString();
                music.size = jsonRoot[i]["size"].asString();
                music.source = jsonRoot[i]["source"].asString();
                music.lyric = jsonRoot[i]["lyric"].asString();
                std::cout << music.source << std::endl;
                songlist.push_back(music);
            }
        }
    }
    return songlist;
}
```

2、封装 json

Json::Value 可以存储 json 格式的数据,将传输的 data 数据转换为 json 格式的数据存储之后,调用 open()打开文件,将 json 数据存储到文件中,返回封装完成的文件名称。

```
//封装json
std::string MyJson::packageSonglistJson(const std::vector<std::string> data, const std::string file)
{
    Json::Value arrayObj;
    int n = 5;
    Json::Value new_item;

    std::vector<stru_music> songlist;
    stru_music music;

    for(auto &t : data){ ...}

    // 构建对象
    for (int i = 0; i < songlist.size() ; i++) { ...}

    std::string filename = ".json";
    filename = path + file + filename;
    std::ofstream ofs;
    ofs.open(filename);
    ofs<< arrayObj.toStyledString();
    ofs.close();

    return filename;
}
```

9.2. FFmpeg 解码.mp3 文件

AVFormat 是一个贯穿始终的数据结构,很多函数都要用到它做参数。音频的元数据(metadata)信息可以通过 AVDictionary 获取。元数据存储在 AVDictionaryEntry 结构体中。

每一条元数据分别位 key 和 value 两个属性,在 FFmpeg 中通过 av_dict_get()函数获得音频的元数据。

```
//解析.mp3文件获取音乐信息
AVFormatContext *fmt_ctx = NULL;
AVDictionaryEntry *tag = NULL;

av_register_all();

fmt_ctx = avformat_alloc_context();

//时长
 auto d = fmt_ctx->duration;
 duration = formatTime(d);
 cout <<"时长:  " << duration.toStdString() <<endl;

 //大小
 double tmp = (fileinfo.size()/1024.0)/1024.0;
 size = QString::number(tmp, 'f',1);
 cout << "大小:  " << size.toStdString() <<endl;


char*  ch;
QByteArray ba = fileinfo.absoluteFilePath().toLatin1(); // must
ch=ba.data();
cout << ch << endl;

int ret;
if(ret = avformat_open_input(&fmt_ctx, ch, NULL, NULL)){
    cout << "Fail to open file" << endl;
}


cout << "+++++++" << endl;
//读取metadata中所有的tag
while ((tag = av_dict_get(fmt_ctx->metadata, "", tag, AV_DICT_IGNORE_SUFFIX))){
    cout << "tags:" << endl;
    cout << tag->key << + ": " << tag->value << endl;
    string tmp = tag->key;
    if(tmp== "album") album = tag->value;
    if(tmp == "artist") singer = tag->value;
    if(tmp == "title") name = tag->value;

}
```

9.3. 服务器交互

Server 管理 Session 类,等待客户端连接,接收到连接后,创建一个 Session 类提供服务,使用 async_accepte()接收一个链接,连接成功后生成一个 Session 类处理客户端请求服务。

```
void Server::start_accept()
{
  Session::Pointer session = Session::create(acceptor_.get_io_service());
  acceptor_.async_accept(session->socket(),
                         boost::bind(&Server::handle_accept, this, session,
asio::placeholders::error));
}

void Server::handle_accept(Session::Pointer session, const Server::Error &error)
{
  if (error) return print_asio_error(error);
  session->receive_service();
  start_accept();
}
```

服务器到客户端后,判断客户端请求服务,调用相关函数向客户端提供服务

```
void Session::do_service(const Error& error)
{
    std::cout << "message: " << buffer_ << std::endl;
    if(strcmp(buffer_, "login") == 0){
        login();
    } else if(strcmp(buffer_, "registe") == 0){
        registe();
    } else if ((strcmp(buffer_, "upload") == 0)) {
        receive();
    } else if ((strcmp(buffer_, "download") == 0)) {
        send();
    } else if(strcmp(buffer_, "online") == 0){
        onlineSonglist();
    } else if(strcmp(buffer_, "songlist") == 0){
        sendSonglist();
    } else if(strcmp(buffer_, "custom") == 0){
        //接收用户自定义歌单
    }

    receive_service();
}
```

9.4. 客户端交互

连接服务器,向服务器请求服务,使用 connect()连接服务器。

```
void Client::connetct(const char *ip_address, unsigned port)
{
    socket_.connect(TCP::endpoint(asio::ip::address_v4::from_string(ip_address), port));
}
```

连接成功调用相关函数,向服务器请求相关服务。

```
//向服务器发送请求
socket_.send(asio::buffer(service, message_max_size));
```

第10章　集成测试

10.1. 主界面

主界面分为三个模块：播放模块、在线歌单模块、功能模块

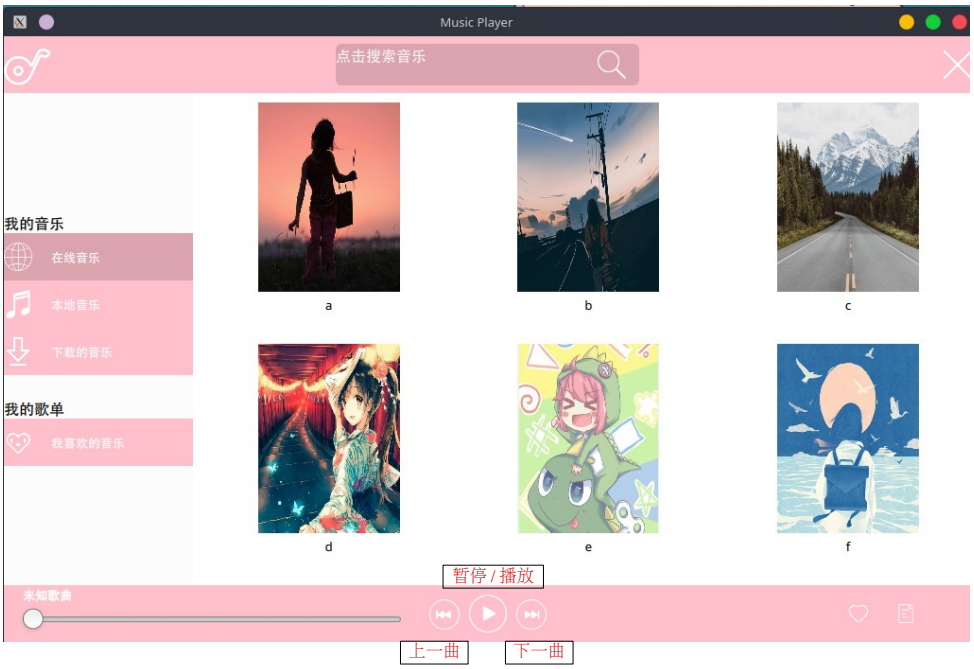


图 10.1　主界面

10.2. 歌曲列表界面

歌曲列表显示音乐的信息（序号、标题、歌手、专辑、时长、大小）

图 10.2　歌曲列表界面

10.3. 歌词界面

图 10.2　歌词界面

10.4. 登录注册界面

若用户已注册，则输入用户名和密码则可的登录成功。

图 10.2　登录/注册界面

后记

参考文献

- 用况建模 Use Case Modeling [美] Kurt Bittner/Lan Spence 著.
- 《Object-Oriented Systems Analysis and Design Using UML.》
- 软件需求管理 用例方法（第二版）[美] Dean Leffingwell/Don Widrig 著. Ed Yourdon 序.
- 掌握需求过程（第三版）[英] Suzanne Robertson James Robertson 著.
- 敏捷软件需求　团队、项目群与企业级的精英需求实践 [美] Dean Leffingwell 著