### 三、Kruchten “4+1”视图模型

#### 3.1 Kruchten “4+1”视图模型概述

由于不同涉众对于同一个软件系统的构成的关注点不同，因此我们可以针对不同涉众展现不同的视图，凸显与他们相关的方面而隐藏与他们无关的细节。

Kruchten的“4+1”视图就是一种将系统架构以不同视图，展现给不同涉众的视图模型，其将结构模型、框架模型、动态模型、过程模型和功能模型有机地统一在一个完整的模型中来刻画软件体系结构。

逻辑视图

进程视图

部署视图

开发视图

**图3-1 Kruchten “4+1”视图**

#### 3.2逻辑视图

##### 3.2.1逻辑视图概述

逻辑视图面向系统分析和设计人员，关注系统提供给最终用户的服务，关注客户的功能需求。以本购票系统为例，逻辑视图就是要向客户展示本购票系统的功能。在面向对象技术中心，通过抽象、封装和继承，可以利用对象模型来代表逻辑视图。可以采用类图、包图来描述软件的静态组成，用序列图来描述软件的动态结构。

##### 3.2.2类图描述

本文采用类图描述本音乐管理系统中每个类的主要操作以及类之间的关系：

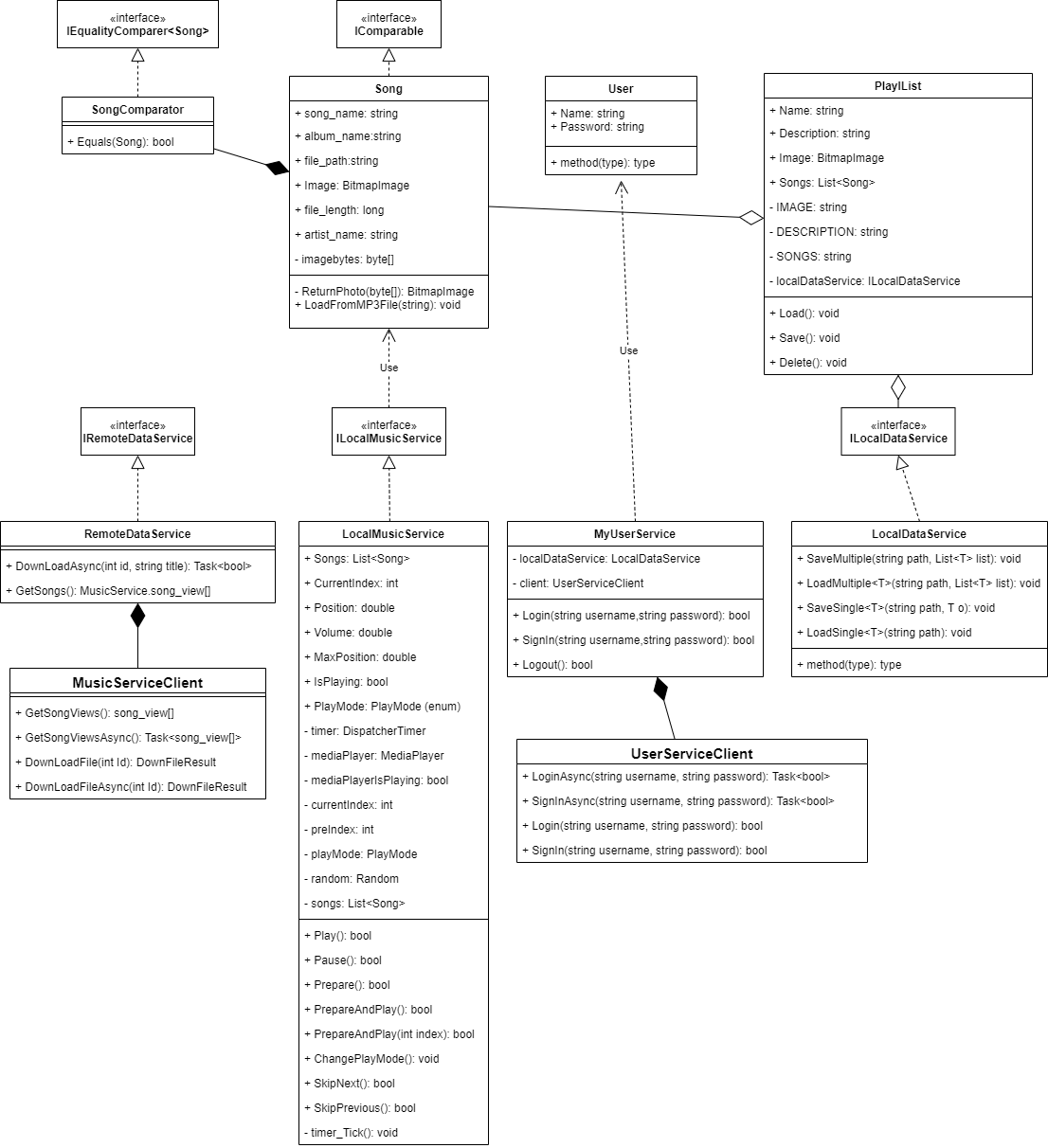


图3-2 音乐管理系统类图1

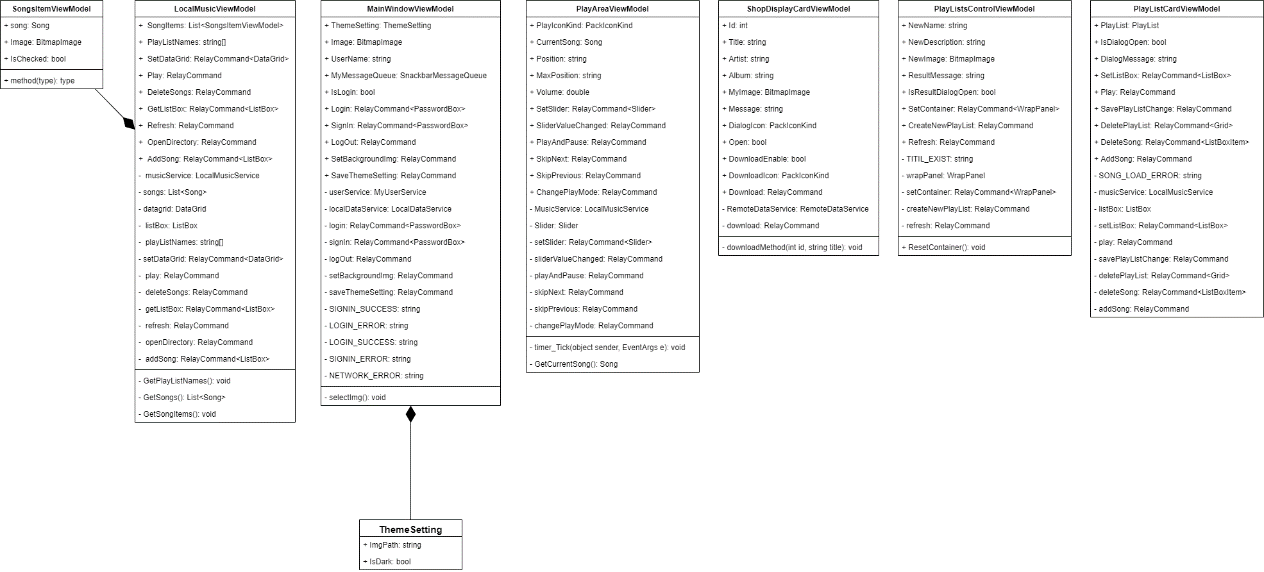


图3-3 音乐管理系统类图2

本系统主要包括用户类、活动类、购物车类、订单类、地址类、活动分类类、订单详情类、管理员类和会员类共9大类。其中，管理员类和会员类继承用户类而来。

##### 3.2.3包图

包图能够将各个类通过包分类管理，每个包包含相同功能的类，能表达出系统的层次结构

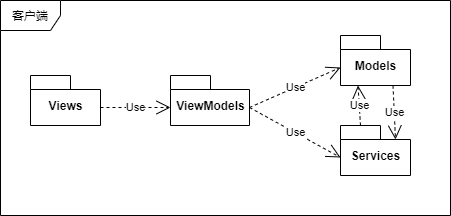


图3-4 音乐管理系统包图

##### 3.2.4序列图

序列图通过绘制用户、对象在时间序列上的消息传递，能够描述出系统中各组成部分的动态交互，展示系统的动态结构和功能。

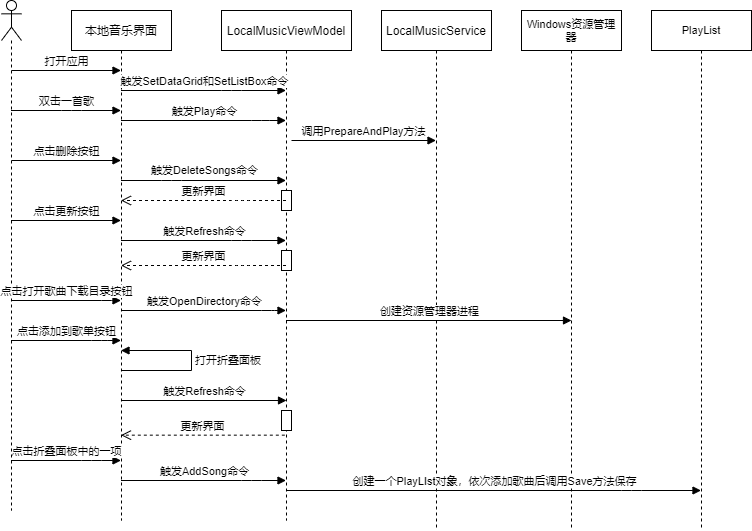


图3-5 音乐管理系统序列图1

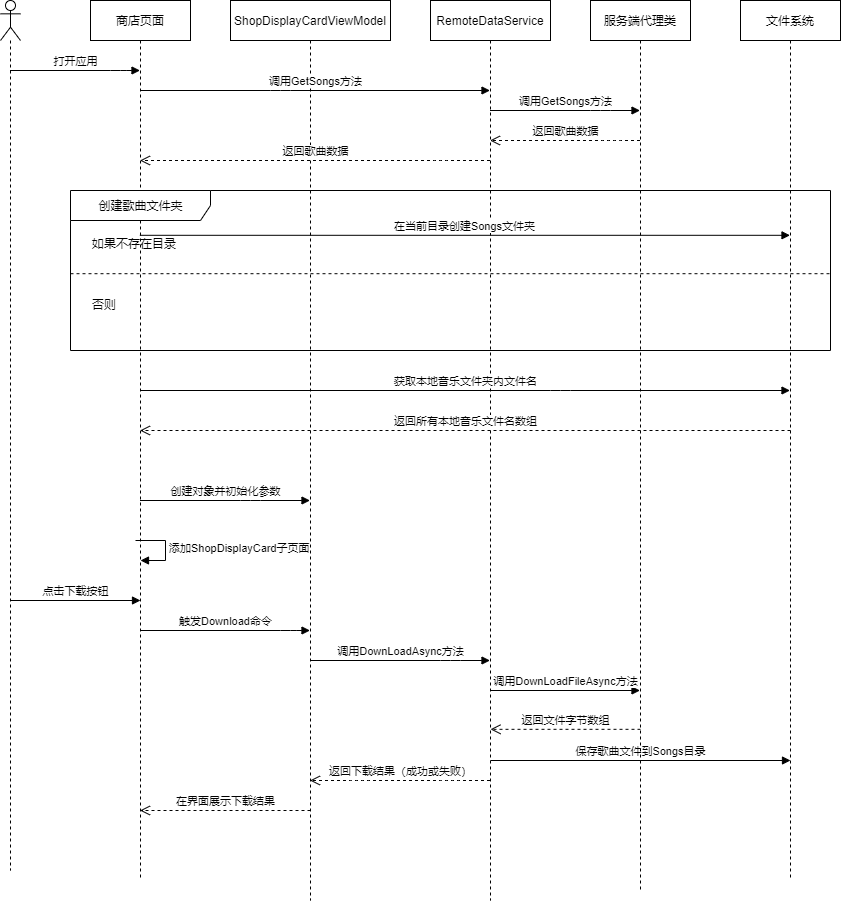


图3-6 音乐管理系统序列图2

#### 3.3 开发视图

##### 3.3.1开发视图概述

开发视图面向编程人员，侧重软件模块的组织与管理。以本购票系统为例，开发视图就需要考虑软件内部结构、构造，比如本系统开发的容易性、本系统软件的重用性和软件的通用性以及由于不同开发工具而带来的局限性。那么，编程人员是对系统的内部最熟悉的，开发视图就需要画出系统的软件模块是如何组织、管理的，这是小粒度描述；大粒度描述还可以画出源程序文件等之间有什么关系。

与逻辑视图相比，开发视图表现了由小组开发的工件的物理层次，组件图通常用来表达开发视图。

##### 3.3.2组件图描述

在UML中，组件图表示组件是如何互相组织以构建更大的组件或是软件系统的。他们被用来标示系统的结构，显示了不同组件、可用的端口以及在开发环境中的依赖关系，包括提供的接口和需要的接口等。

（1）系统组件图

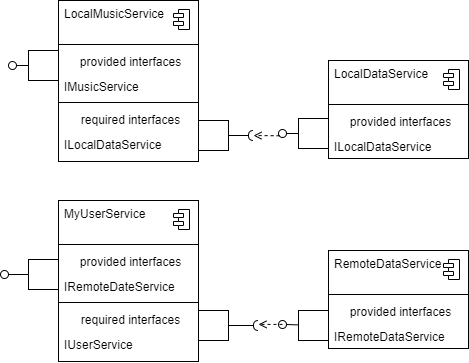


图3-7 系统组件图

#### 3.4 进程视图

##### 3.4.1进程视图概述

进程视图面向系统集成人员，侧重系统运行时的特性。以本购票系统为例，当系统初步实现编码后，已经可以初步运行时。则进程视图就是要把系统运行过程中的性能表现出来，比如能同时访问的用户数量、系统购票延迟等。所以，进程视图主要表现系统运行时的特性。可从静态与动态两方面画图：静态方面关注软件系统运行时的单元结构；动态方面关注运行时的单元之间的交互机制。

##### 3.4.2 活动图描述

活动图与流程图较为相似，在不同的视图中有广泛的应用，在进程视图中，它们可以用来描述程序流和复杂的业务逻辑，包括行为、决策点、分支、合并以及并行处理等。

（1）用户登录注册活动图

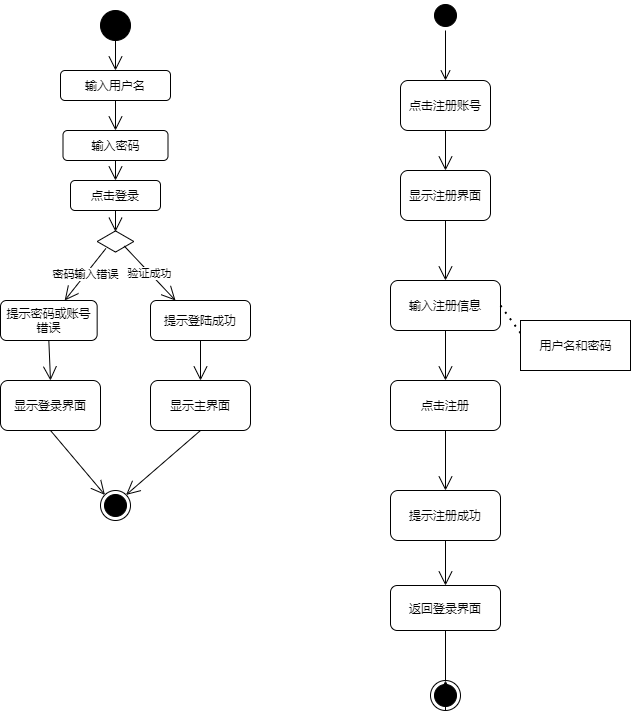


图3-7活动图1

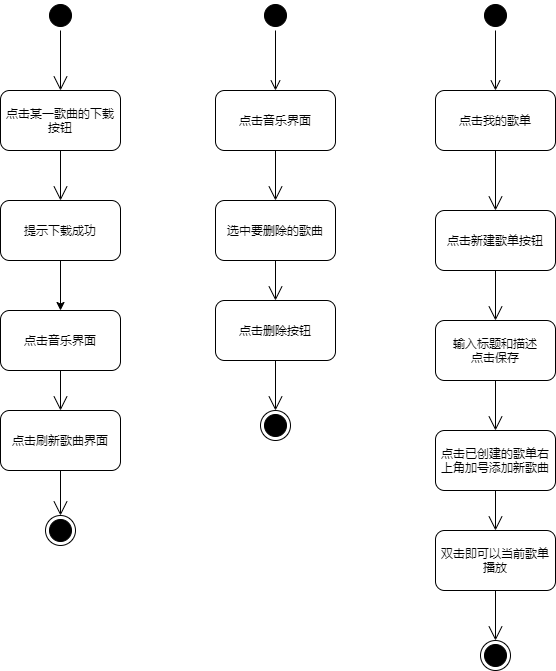
（2）歌曲下载、删除、新建歌单活动图

图3-8 歌曲下载、删除、新建歌单活动图

（4）修改歌单信息、账号登出活动图

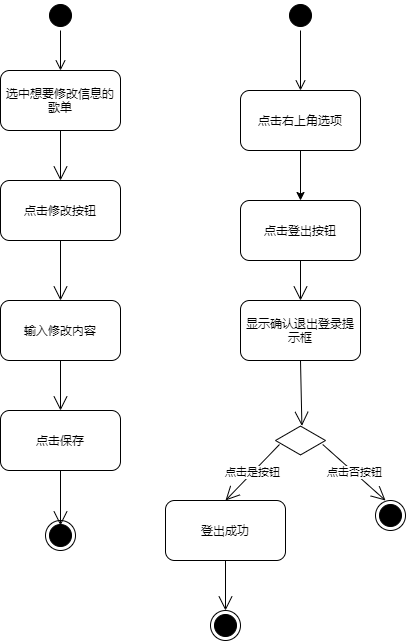


图3-9 修改歌单信息、账号登出活动图

#### 3.5 部署视图

##### 3.5.1部署视图概述

部署视图面向系统工程师、系统部署人员，关注系统、网络、服务器等基础设施，着重考虑安装与部署需求。解决系统拓扑结构、安装、通信等问题。

##### 3.5.2 部署图描述

部署图显示了现实世界中工件的物理位置，UML提供了构造型来表达节点，以及这些设备的依赖关系等。用户PC和服务器通过WCF的远程过程调用RPC通信，服务器端部署音乐数据库和数据库辅助软件。

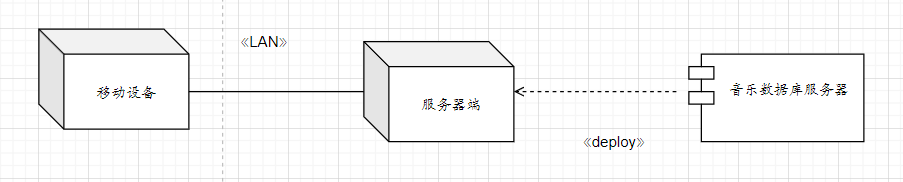


图3-10 音乐管理系统部署图

#### 3.6 用例视图

##### 3.6.1用例视图描述

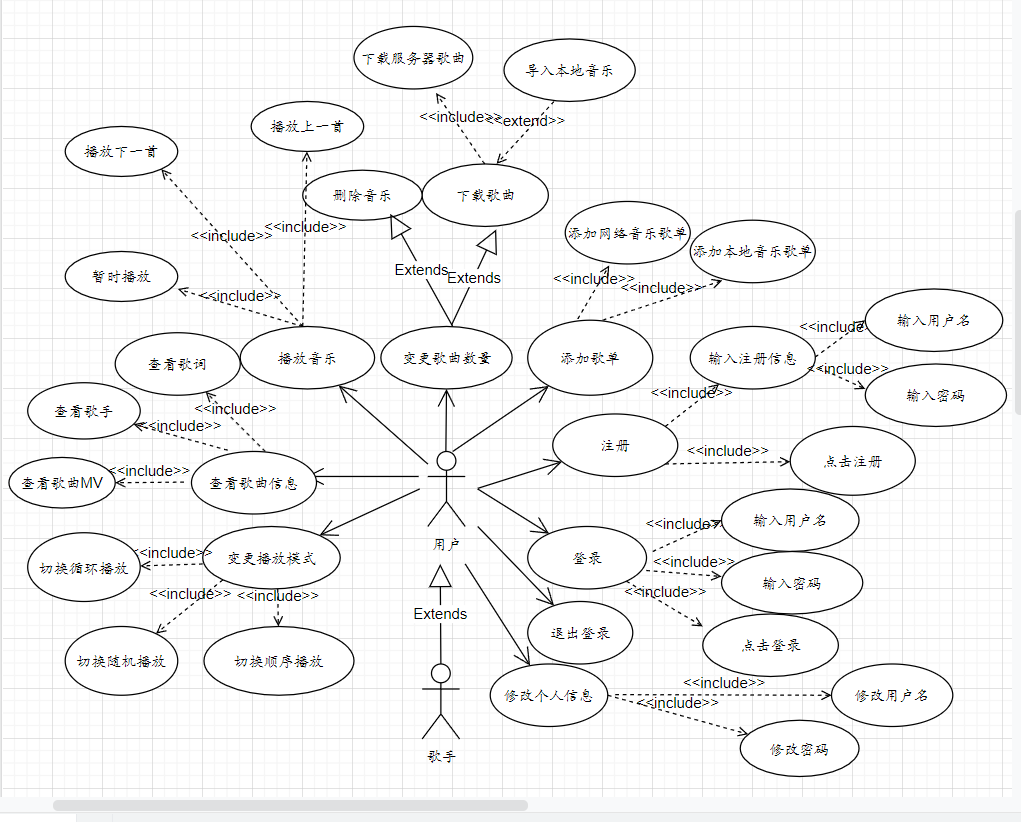
用例视图面向用户，要展示外部用户所能观察到的功能。用例视图也称场景，是重要系统活动的抽象。开发体系结构时，用例视图可以帮助架构师找到体系结构的构件及其关系。

用例视图将其它4个视图有机联系起来，从某种意义上说，用例视图是最重要的需求对象，在软件体系结构开发中，它可以帮助设计者找到体系结构的构件和它们之间的关系作用。

##### 3.6.2用例图描述

UML提供了用例图表达用例视图，用例图由用例和参与者构成，它们与详细的场景描述紧密联系，作为架构视图，本文只描述对需要建模的重要用例。

1.普通用户



名称：音乐管理系统

用例标识号：001

参与者：用户

简要说明：用户是音乐播放器的使用者，可以注册、登录、退出、修改各人信息和播放歌曲、添加歌单、变更歌曲数量以及查看个人信息。音乐人则是普通用户的一种。

前置条件：注册并登录音乐播放器

基本事件流：

1. 新用户点击“注册”按钮，填写用户名和密码后，单击“登录”按钮，填写正确的用户名和密码后，可进入主页面。

2. 点击个人中心修改信息。

3. 点击歌单可以添加本地音乐歌单和网络音乐歌单。

4. 点击删除音乐和下载音乐变更歌曲数量。

5. 可以分别单击“上一首”“下一首”“暂停”，来播放音乐。

6. 点击查看歌词信息可以查看歌词、歌手、歌曲MV。

7. 分别点击顺序播放、循环播放、随机播放更换播放模式。

其他事件流：

A1：用户不注册或者退出后仍然可以播放音乐，但不能使用个人中心。

A2：用户非登录期间保存的歌曲登录后依然存在。

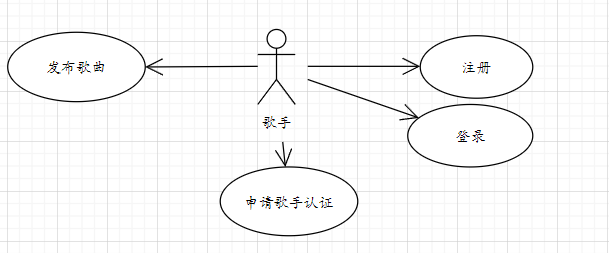
异常事件流：

1. 音乐播放器遇到异常退出。

后置条件：歌曲可下载变更。

注释：无

2.歌手



用例标识号：002

参与者：歌手

简要说明：歌手是普通用户的一种。歌手可以注册登录进播放器后，进行歌手认证，认证完成后可以上传歌曲，审核完成后可以发布。

前置条件：歌手注册并登录音乐播放器

基本事件流：

1.歌手点击“注册”按钮，填写用户名和密码后，单击“登录”按钮，填写正确的用户名和密码后，可进入主页面。

2.点击个人中心修改信息

3.申请歌手认证

4.发布歌曲

其他事件流：

A1：歌手不用申请歌手认证也可以发布歌曲

异常事件流：

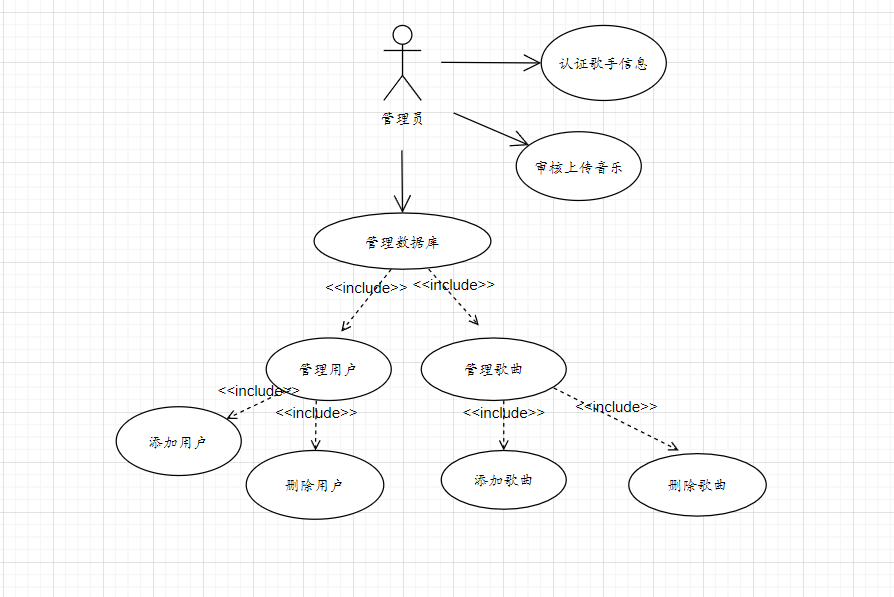
1. 音乐播放器遇到异常退出。

2. 审核失败

后置条件：无

注释：无

3服务器管理员



用例标识号：003

参与者：管理员

简要说明：管理员是后台管理人员，可以审核上传音乐，认证歌手，管理用户和歌曲。

前置条件：管理员用后台管理的权限。

基本事件流：

1.管理员可以··认证歌手

2.管理员审核上传音乐

3.管理用户和音乐

其他事件流：

A1：管理员可以审核用户的违规行为

异常事件流：

服务器出现故障

后置条件：拥有管理权限

注释：无