|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | 版 本： | V1.0 | |  |
|  | |  | |  | 密 级： | 内部 | |  |
|  | |  | |  | 总页数： | 38 | |  |
|  | | | | | | | | |
| **大数据应用工程实践课程设计文档**  文档控制号：  在线音乐播放网站软件  概要设计说明 | | | | | | | | |
|  | | | | | | | | |
|  | **单 位** | | 20及数据科学与大数据技术班 | | | |  | |
|  | **编 写** | | 耿晓雪 | | | |  | |
|  | **参与者** | | 张金聪 李光华 耿晓雪 刘光蕊 郭彦泽 毛欣竹 徐艳华 | | | |  | |
|  | **校 对** | | 耿晓雪 | | | |  | |
|  | **审 核** | |  | | | |  | |
|  | **批 准** | |  | | | |  | |
|  | | | | | | | | |
|  | | | | | | | | |
|  | **楚雄师范学院数计学院** | | | | | |  | |
|  | | | | | | | | |

目录

[一． 项目简介 3](#_Toc24430)

[1.1 项目简介 3](#_Toc28144)

[1.2 访问yin音乐网的状态： 3](#_Toc6855)

[二． 项目分析 3](#_Toc19543)

[1.3 功能实现 3](#_Toc3035)

[三． 总体设计 4](#_Toc7015)

[3.1音乐网站组成 4](#_Toc25197)

[3.2音乐网站技术架构 5](#_Toc19595)

[四． 概要设计 14](#_Toc12303)

[4.1分层（总体）设计 14](#_Toc16797)

[4.2数据库设计 20](#_Toc13204)

[4.3实体类设计 24](#_Toc6393)

[4.4项目的配置文件 27](#_Toc26974)

[4.5前后端交互 28](#_Toc7974)

[五． 项目结果 33](#_Toc20180)

# 项目简介

1. 项目简介

Yin音乐网站设计主要是为了更好的直接的更改音乐网站里面的信息，修改歌手歌曲信息等。它的特点是前端和后端可以相互交互，更改歌曲信息歌手信息的同时数据库也会更新。

该音乐网站的设计范围主要包括前端和后端的交互，创建属于自己的仓库，并把数据、文件等传入自己所建的仓库保存。Yin音乐网站设计可以通过npm run serve命令代开音乐网站，并展示yin音乐的具体信息。

1. 访问yin音乐网的状态：

1.管理员：

（1）未登录状态：只显示一个登录页面，不能查看任何信息。

（2）登录状态：可以实现未登录状态下的所有操作，同时可以有更多的操作：查看歌单歌手及用户的详细信息，还可以查看用户数量，歌曲数量，歌手数量及歌单数量，用户和歌手的性别比例，歌曲类型和歌手国际等，还可以直接对歌曲普通用户及歌手进行编辑，修改歌单照片等。

2.普通用户：

（1）未登录状态：可以查看歌单、歌手的详细信息及分类，还可以搜索歌曲、歌手。

（2）登录状态：可以收藏歌曲、评论歌曲等。

# 项目分析

1. 功能实现
   1. 用户注册和登录功能：

（1）提供用户注册页面，收集用户基本信息，并进行验证和存储。

（2）提供用户登录页面，验证用户身份，以便后续操作和个性化功能。

1. 2.音乐搜索和播放功能：

（1）提供搜索框和搜索按钮，允许用户输入关键词搜索音乐。

（2）根据用户输入的关键词，在音乐库中进行检索，并显示相匹配的音乐结果。

（3）用户点击播放按钮后，开始播放所选音乐，并显示音乐播放进度和控制按钮。

1. 3.歌曲上传和管理功能：

（1）提供上传页面，管理员可以上传音乐文件和相关信息。

（2）在后台进行音乐文件的存储和管理，并为每首歌曲分配唯一的标识符。

（3）提供管理界面，允许管理员编辑和删除已上传的歌曲信息。

1. 4.创建和管理歌单功能：

（1）允许用户创建自己的歌单，将喜欢的音乐添加到歌单中。

（2）提供管理界面，允许用户对歌单进行编辑和删除。

1. 5.评论功能：

（1）允许用户对歌曲进行评论，表达自己的感受和观点。

1. 6、用户个性化设置：

（1）允许用户自定义个人资料，包括头像、昵称和个性签名等。

1. 7、音乐排行：

（1）提供热门音乐排行榜，展示当前热门的歌曲和艺术家。

1. 8、歌词展示：

（1）对于支持歌词的音乐文件，提供歌词展示功能，帮助用户更好地理解和欣赏音乐。

# 总体设计

3.1音乐网站组成

1. 组成部分

1.前端（Front-end）：

用户界面（UI）：包括网站的主页、播放器、搜索功能、歌曲排行榜等。

用户体验（UX）：设计用户友好的界面，确保良好的用户交互和体验。

HTML/CSS/JavaScript：构建和呈现前端页面的技术基础。

2.后端（Back-end）：

服务器端技术：处理请求、处理逻辑、与数据库交互等。

数据库管理系统：用于存储和管理音乐数据、用户信息以及其他相关数据的数据库系统。

API（Application Programming Interface）：用于与前端页面交互和数据传输的接口。

3.用户管理系统：

用户注册和登录功能：让用户创建账户并登录，以便于他们收藏、评论和与其他用户互动。

用户个人资料管理：用户个人信息、收藏的歌曲、歌单等。

4.音乐库和数据管理：

音乐上传和存储：允许音乐供应商、艺术家或用户上传音乐，并在服务器上存储。

元数据管理：用于上传和关联音乐文件的元数据，如歌曲标题、艺术家、专辑、封面等信息。

音乐分类和标签：对音乐进行分类、标记和组织，以方便用户浏览和发现音乐。

5.播放：

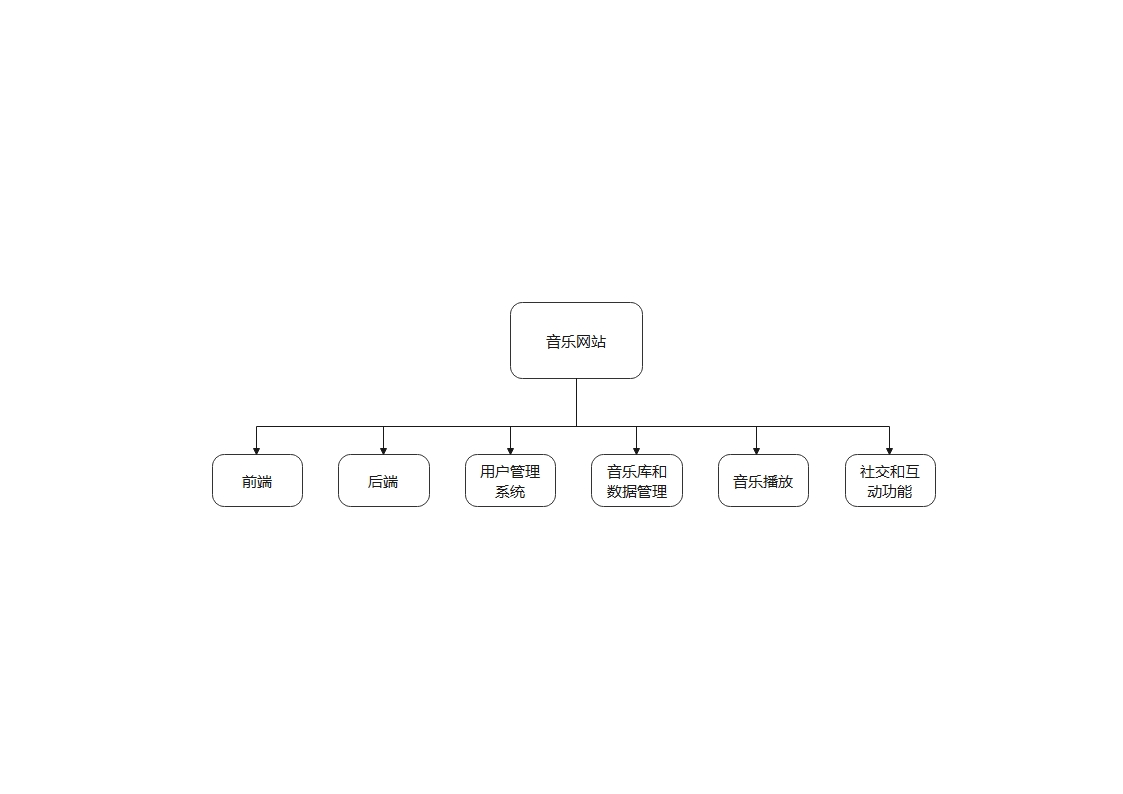
音乐播放器：提供基本的音乐播放功能，如播放、暂停、上一首、下一首等。

6.社交和互动功能：

评论和评分：用户可以在音乐上留下评论和评分，与其他用户交流。

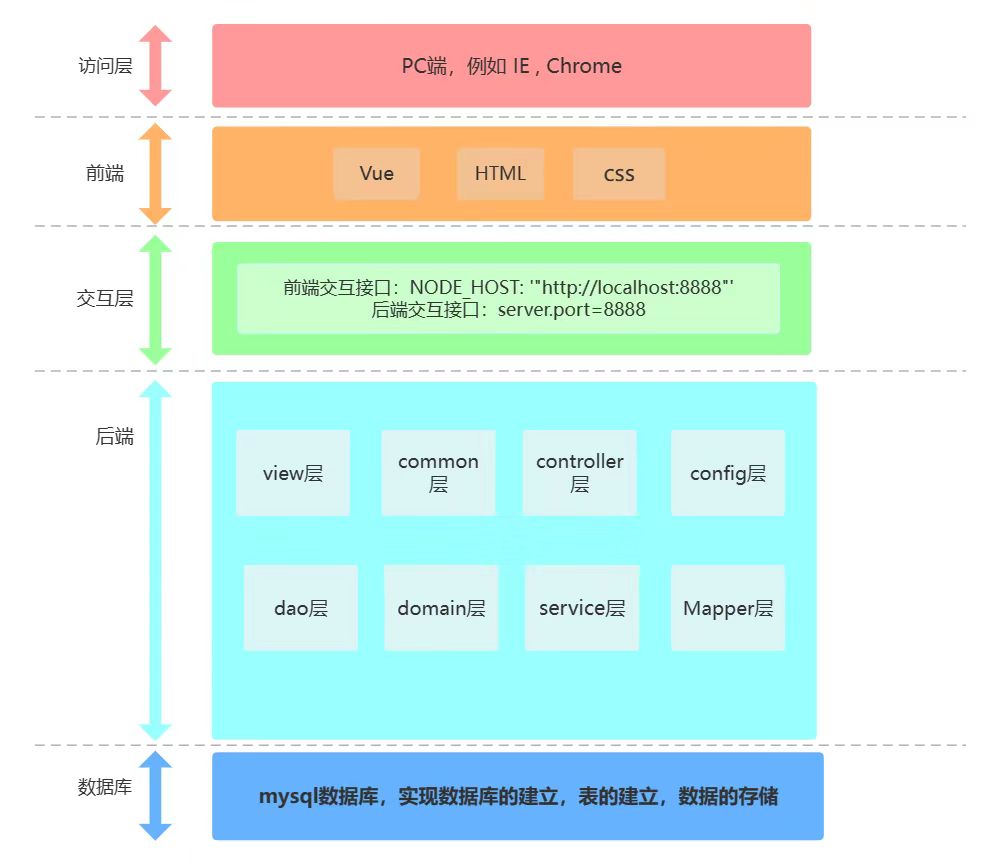
分享和推荐：用户可以与好友分享自己喜欢的歌曲、歌单或推荐给其他用户。

社交集成：集成第三方社交媒体平台，让用户更方便地与其他社交圈子分享音乐。

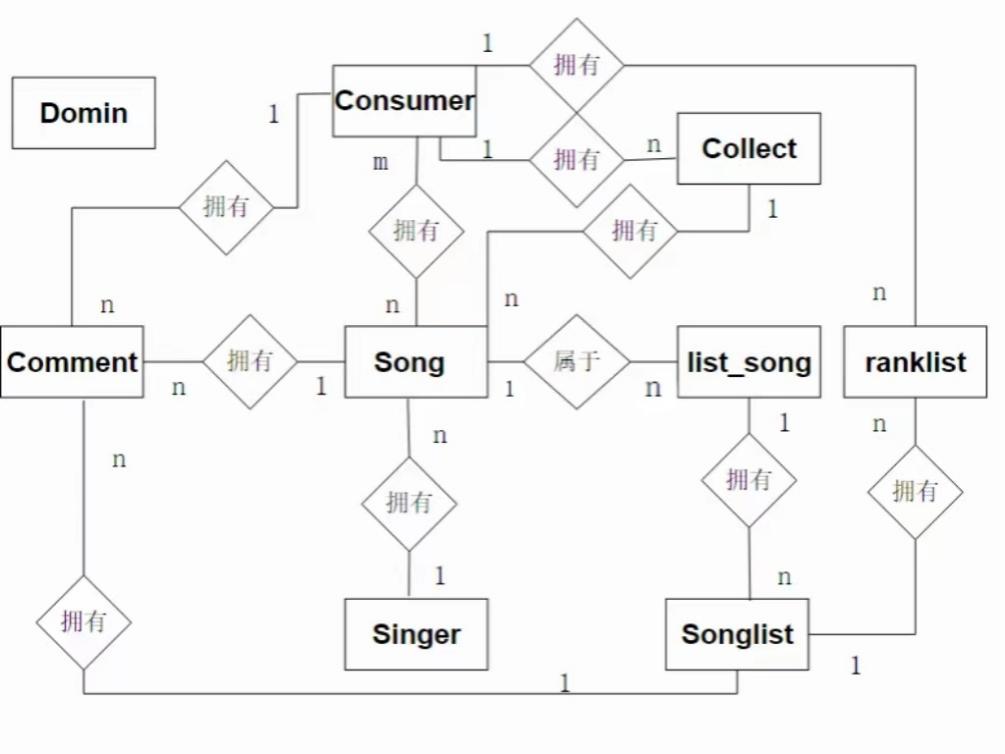


3.2音乐网站技术架构

1. 架构图展示



1. ER图展示

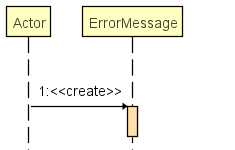


1. 顺序图

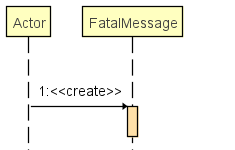
（1）顺序图是软件开发过程中一种常用的交互建模工具，它主要用于描述系统中对象之间的时序关系和交互流程。顺序图作为一种交互建模工具，可以揭示对象间的交互流程、强调时序关系、支持系统设计和分析、促进交流与共享理解，以及支持测试和验证。它是软件开发过程中重要的工具之一，可以帮助开发团队更好地理解系统的行为和交互方式。

（2）图形及其解释

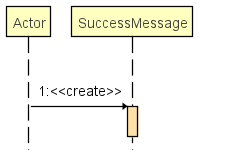
1. 信息提示板块：



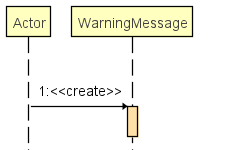
在创建参与者时可能遇到的错误信息

、

在创建参与者时可能遇到的致命错误消息



在创建参与者时得到的成功消息



在创建参与者时得到的警告消息

1. 输入密码



**Actor**：该角色代表与系统交互的用户或外部系统。

**AdminController：**该角色是一个控制器类，在代码中被定义为class AdminController。它处理管理员相关的请求，并调用相应的服务类完成业务逻辑。其中，loginStatus()方法是该控制器类的一个方法，用于判断管理员登录状态。

**AdminServiceImpl：**该角色是一个服务实现类，代码中定义为class AdminServiceImpl。它实现了与管理员相关的业务逻辑，并被控制器类(AdminController)调用。其中，veritypasswd()方法是该服务实现类的一个方法，用于验证管理员的用户名和密码是否匹配。

**AdminMapper：**该角色可能是一个数据访问对象或映射器，负责与数据库进行数据交互。在代码中没有具体的定义和使用，所以无法得知该角色的具体功能和行为。

**SuccessMessage：**该角色可能是一个自定义类，用于封装成功消息的对象。它包含了一个成功消息的内容，并提供了获取消息的方法(getMessage())。在代码中，当管理员登录成功时，会创建一个SuccessMessage对象，并通过调用getMessage()方法获取成功消息。

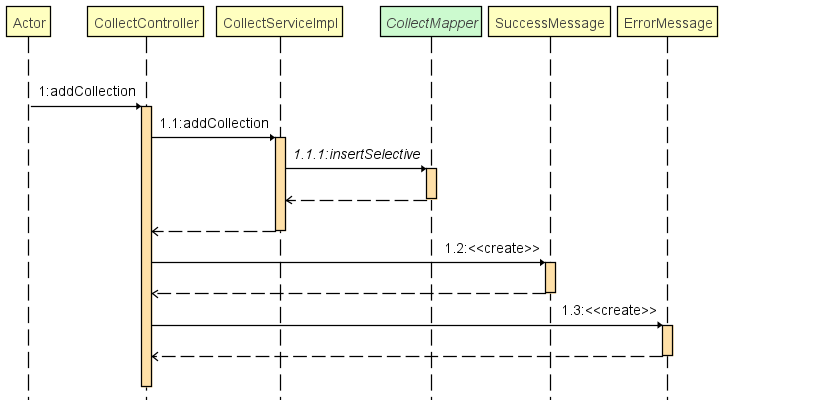
**ErrorMessage：**该角色可能是一个自定义类，用于封装错误消息的对象。它包含了一个错误消息的内容，并提供了获取消息的方法(getMessage())。在代码中，当管理员登录失败时，会创建一个ErrorMessage对象，并通过调用getMessage()方法获取错误消息。

**loginStatus：**该对象表示顺序图中的一个消息，指示从Actor角色向AdminController角色发送了一个名为loginStatus的请求消息。

**veritypasswd：**该对象表示顺序图中的一个消息，指示从AdminController角色向AdminServiceImpl角色发送了一个名为veritypasswd的方法调用消息。

**<ecreate>>：**该对象表示一个创建消息，通常用于表示对象的实例化。在这个顺序图中，2'<ecreate>>暗示了在某个时刻创建了一个未具体说明的对象。

1. 添加收藏的歌曲，取消收藏歌曲



**CollectController：**该角色是一个控制器类，在代码中被定义为class CollectController。它处理与收藏功能相关的请求，并调用相应的服务类完成业务逻辑。在这个顺序图中，addCollection()方法是该控制器类的一个方法，用于添加收藏。

**CollectServiceImpl：**该角色是一个服务实现类，代表收藏功能的具体实现。在代码中定义为class CollectServiceImpl。它实现了与收藏相关的业务逻辑，并被控制器类(CollectController)调用。在这个顺序图中，insertSelective表示调用CollectServiceImpl中的insertSelective()方法。**CollectMapper：**该角色可能是一个数据访问对象或映射器，负责与数据库进行数据insertSelective表示调用CollectMapper中的insertSelective()方法，该方法用于插入收藏相关的数据到数据库中。

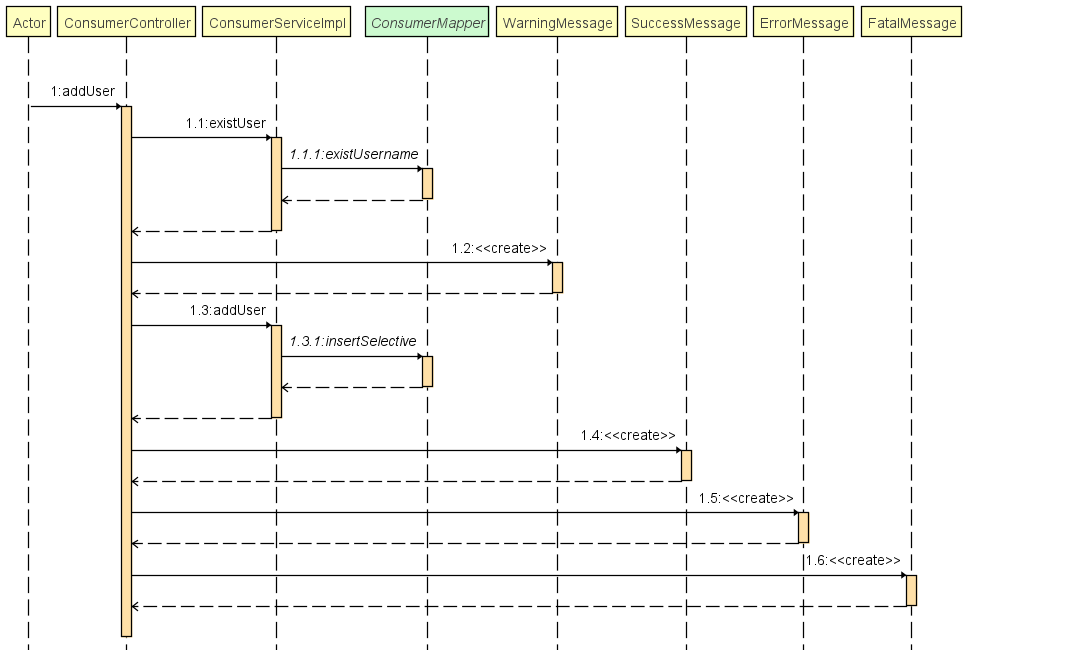
1. 评论并对评论点赞



**CommentController：**该角色是一个控制器类，负责处理与评论功能相关的请求。在代码中定义为CommentController类。它接收来自用户或外部系统的请求，并调用相应的服务类完成业务逻辑。在顺序图中，addComment()表示CommentController类的一个方法，用于添加评论。

**CommentServiceImpl：**该角色是一个服务实现类，表示评论功能的具体实现。在代码中定义为CommentServiceImpl类。它实现了与评论相关的业务逻辑，并由控制器类(CommentController)调用

1. 用户注册

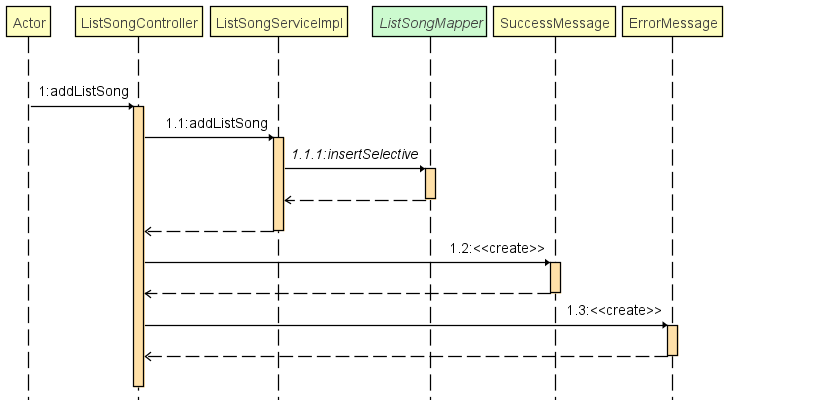


**CommentController：**该角色是一个控制器类，负责处理与评论相关的请求。它接收来自用户或外部系统的请求，并调用相应的服务类来完成业务逻辑。

**CommentServiceImpl：**该角色是评论功能的服务实现类。它实现了与评论相关的业务逻辑，并由控制器类(CommentController)调用。

**CommentMapper：**该角色可能是一个数据访问对象或映射器，用于与数据库交互。

1. 添加歌单

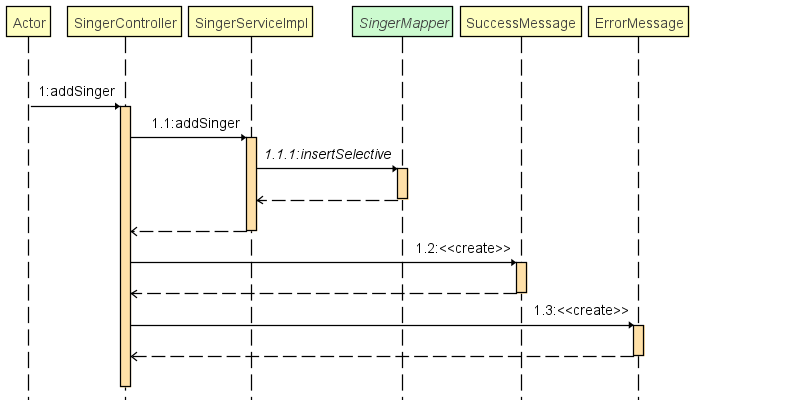


**ListSongController:**是一个控制器（Controller）类，它负责接收来自用户或外部实体的请求，并将其转发给服务层进行处理。

**ListSongServiceImpl：**ListSongServiceImpl是一个服务层（Service）类，它包含了对歌曲列表进行操作的业务逻辑。

**ListSongMapper：**ListSongMapper是一个数据访问层（DAO）类，它负责与数据库进行交互，执行CRUD（创建、读取、更新、删除）操作。

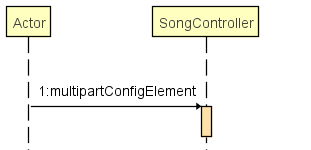
1. 添加歌手



**SingerController：**SingerController是一个控制器（Controller）类，它负责接收来自用户或外部实体的请求，并将其转发给服务层进行处理。

**SingerServiceImpl：**SingerServiceImpl是一个服务层（Service）类，它包含对歌手进行操作的业务逻辑。它接收来自SingerController的请求，并调用相应的方法来处理请求。

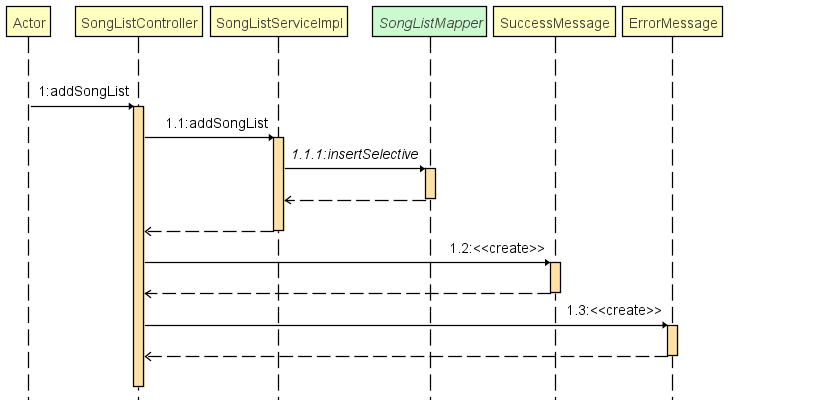
1. 添加歌曲



**SongController：**是一个控制器（Controller）类，它负责接收来自用户或外部实体的请求，并将其转发给相应的处理逻辑。

**multipartConfigElement：**这是在顺序图中表示的消息或事件。它在这个上下文中表示一个配置元素，用于配置控制器或系统以支持分段上传（multipart）的功能。

1. 添加歌单



**SongListController：**SongListController是一个控制器（Controller）类，它负责接收来自用户或外部实体的请求，并将其转发给服务层进行处理。

**SongListServiceImpl：**SongListServiceImpl是一个服务层（Service）类，它包含了对歌单进行操作的业务逻辑。它接收来自SongListController的请求，然后调用相应的方法来处理请求。

1. 部署图
2. 部署图是一种用于可视化系统或应用程序架构的图形表示方法。展示了系统中不同组件、服务或模块的物理或逻辑部署位置以及它们之间的关系和通信方式。硬件环境和软件环境共同构成了音乐网站的部署基础。通过合理的硬件配置和适当的软件环境选择，给用户提供良好的体验。
3. 运行环境
4. 硬件环境

硬件环境包括实际的物理设备，如服务器、存储设备和网络设备。在音乐网站中，硬件环境扮演着以下几个角色和起到以下作用：

①用户端设备：用户通过个人电脑、手机或平板电脑等用户端设备来访问音乐网站。硬件环境的性能和配置将直接影响用户体验，包括页面加载速度、内容传输速度等。

②前端服务器：前端服务器负责处理用户界面的渲染和用户请求的处理。前端服务器的硬件环境可以根据需求进行配置，包括处理器、内存、存储等。

③后端服务器：后端服务器处理用户请求、执行业务逻辑和访问数据库等操作。硬件环境对后端服务器的性能和可扩展性有重要影响。

④数据库服务器：数据库服务器用于存储音乐元数据、用户信息等数据。硬件环境的选择对数据库的性能和可靠性至关重要。

1. 软件环境

软件环境包括操作系统、中间件和应用程序等软件组件。在音乐网站中，软件环境扮演着以下几个角色和起到以下作用：

①操作系统：操作系统提供底层的硬件资源管理和进程调度等功能。

②中间件：中间件层提供各种支持服务，如Web服务器、应用服务器等，它们可以帮助前端和后端服务器与数据库和其他服务进行通信和协作。

③应用程序：应用程序是实际的音乐网站程序，包括前端服务器代码、后端服务器代码和数据库访问代码等。应用程序通过软件环境可实现用户界面渲染、业务逻辑处理和数据操作等功能。

1. 音乐网站的部署图概要设计如下
   1. 前端服务器

承担用户界面的渲染和交互逻辑。使用SpringBoot等前端框架来实现。前端服务器与后端服务器通过API进行通信。前端服务器负责接收用户的请求，渲染用户界面并处理用户交互。它可以使用前端框架来构建用户界面，并向后端服务器请求数据。前端服务器将用户输入的搜索关键字、播放列表等信息发送给后端服务器，并展示搜索结果、音乐播放器等界面给用户。

* 1. 后端服务器

处理用户请求、业务逻辑和数据操作。后端服务器处理前端服务器发送的请求，并进行相应的业务逻辑处理和数据操作。例如，当用户搜索音乐时，后端服务器会通过数据库查询匹配的音乐，并将结果返回给前端服务器。

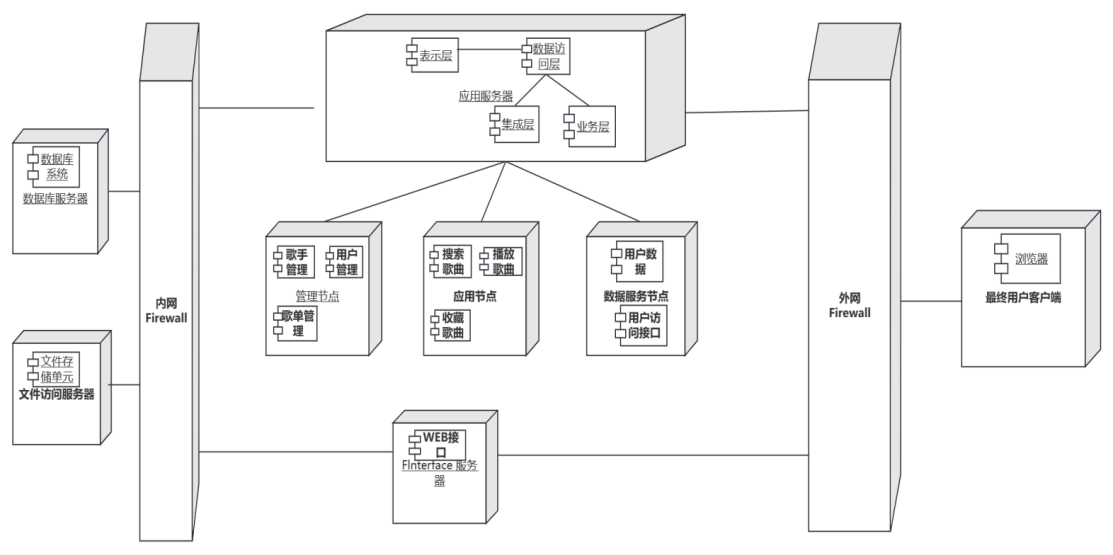
* 1. 数据库服务器

存储音乐元数据、用户信息、播放列表等数据。它可以使用关系数据库（如MySQL8.0）来存储数据。音乐元数据包括歌曲名称、歌手、歌曲等信息。用户信息包括用户名、密码、个人信息等。播放列表是用户创建的包含多个音乐的列表，与音乐元数据建立关联关系。

④ 文件存储服务

存储用户上传的音乐文件和封面图片。用户上传的音乐文件和封面图片可以使用文件存储服务进行存储。这样可以确保音乐文件和封面图片的可访问性和可靠性。

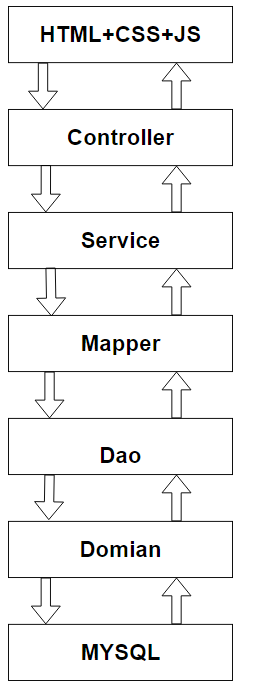
部署图如下：



# 概要设计

4.1分层（总体）设计

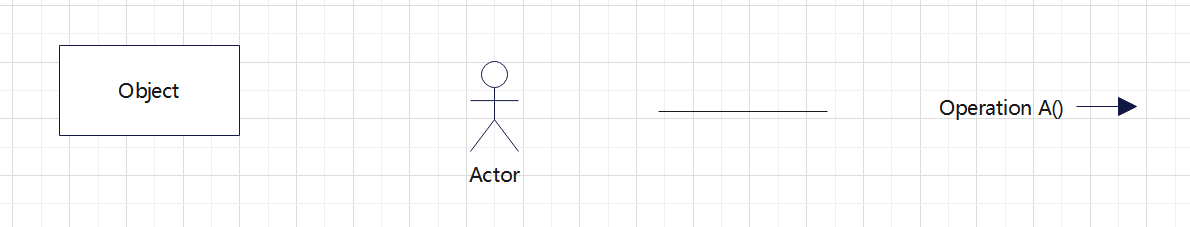
（一）先编写domain 层定义数据库中的属性，先编写domain 层,然后在dao层写实现方法的mysql语句，然后实现service接口，继续写service的实现类，然后在controller写实现方法，然后配置前后端交互文件，然后在前端实现并返回数据库实现交互。

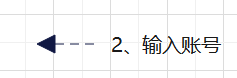


(二)协作图

1. 协作图是用于描述系统中的对象如何一起协作以完成某个功能或实现某个用例的图形化表示。协作图也被称为协作图（Collaboration diagram）或通信图（Communication diagram）。它展示了对象之间的消息传递和交互关系，以及它们之间的协作方式。

主要有如图12的四个部分构成：



信息的返回用如下标表示：

1. 协作图及其分析

**图13表示管理员查找音乐：**

1. 首先用户登录系统 然后返回输入账号，查找音乐后返回输入音乐名，然后在系统窗口提交信息
2. 通过输入的音乐名查询是否有相同的音乐，若显示“否”，则提示不存在，否则就获取相应信息，并显示所查询的信息。

图13



**图14表示管理员添加音乐：**

1. 首先用户登录系统 然后返回输入账号，添加音乐后返回输入音乐信息，然后在系统窗口提交信息
2. 通过输入的音乐信息查询是否有音乐重复，若显示“否”，则提示重复，否则就新增音乐条目，然后再音乐列表更新音乐。

图14

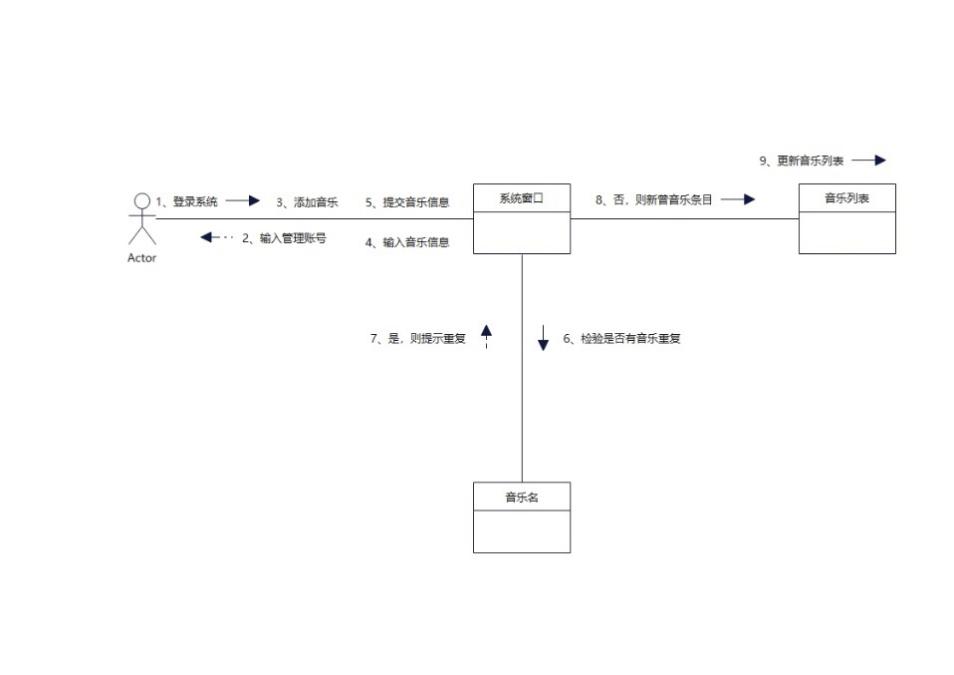


图15表示管理员修改音乐：

首先用户登录系统 然后返回输入账号，查找所有音乐后获取所有音乐，然后返回系统窗口显示所有信息，用户选择并修改音乐后返回提交修改信息，然后更新列表信息，提示修改成功即可。

图15

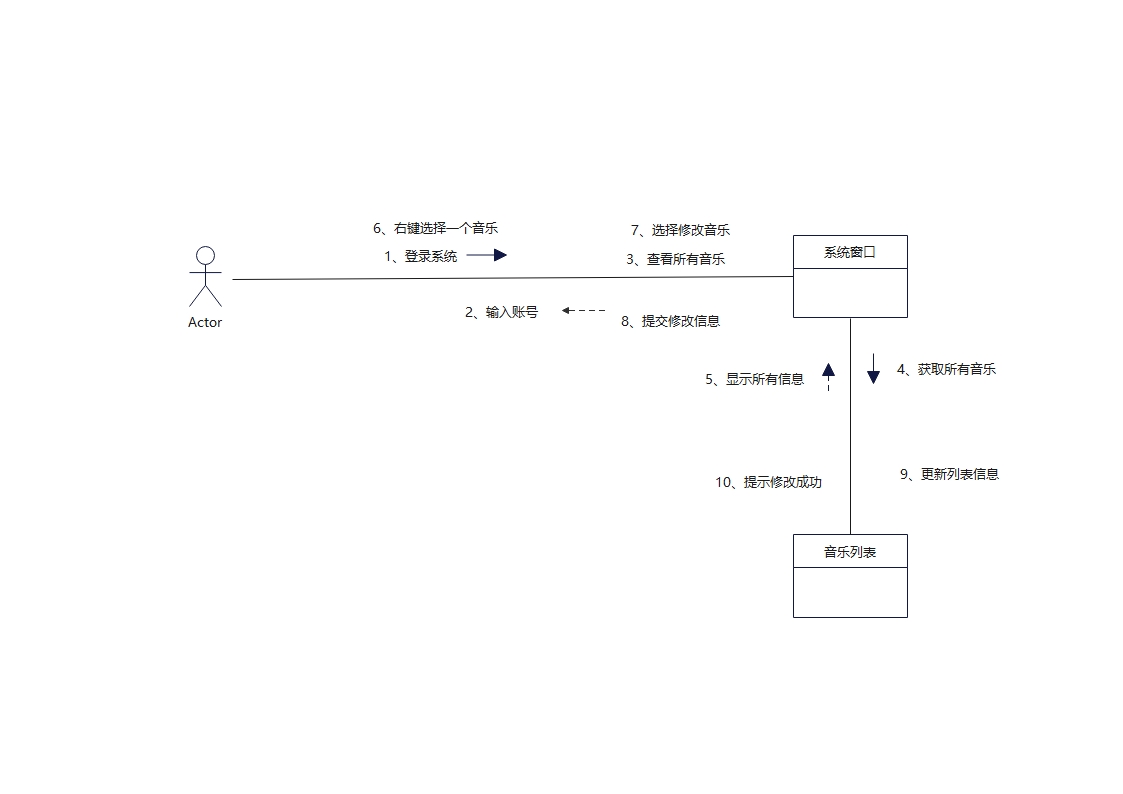


图16表示管理员删除音乐：

用户首先查看音乐，在音乐列表获取所有音乐信息，然后系统窗口返回音乐列表显示所有音乐信息，选择其中一个音乐后返回用户显示右键菜单，然后删除音乐，最后更新音乐列表，提示“删除成功”即可。

图16

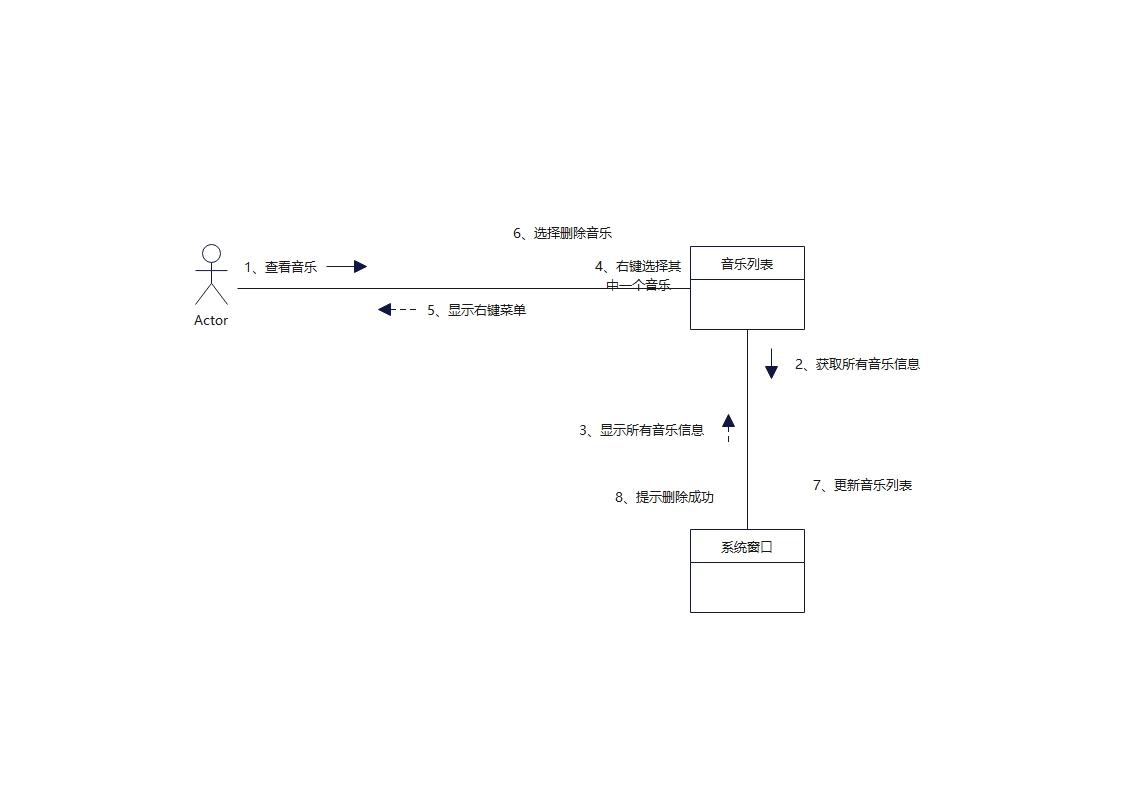


图17表示管理员修改用户：

1. 首先查看用户，获取所有用户信息，然后返回显示所有用户信息，继续返回选择用户
2. 显示右键菜单然后点击修改，显示用户信息后点击确定，更新用户列表即提示修改成功。

图17

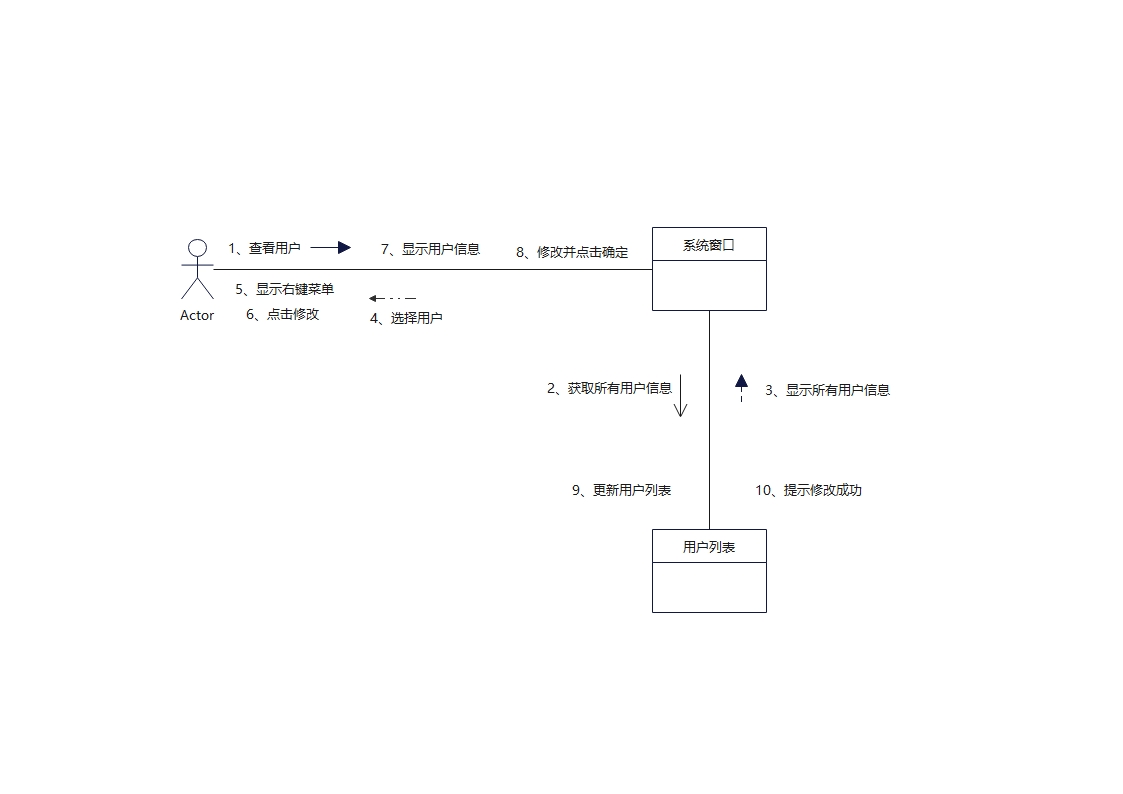


图18表示管理员删除用户：

1. 首先查看用户，获取所有用户信息，然后返回显示所有用户信息，右键选择用户在返回显示右键菜单，然后删除用户。
2. 更新用户信息，提示删除成功即可。

图18

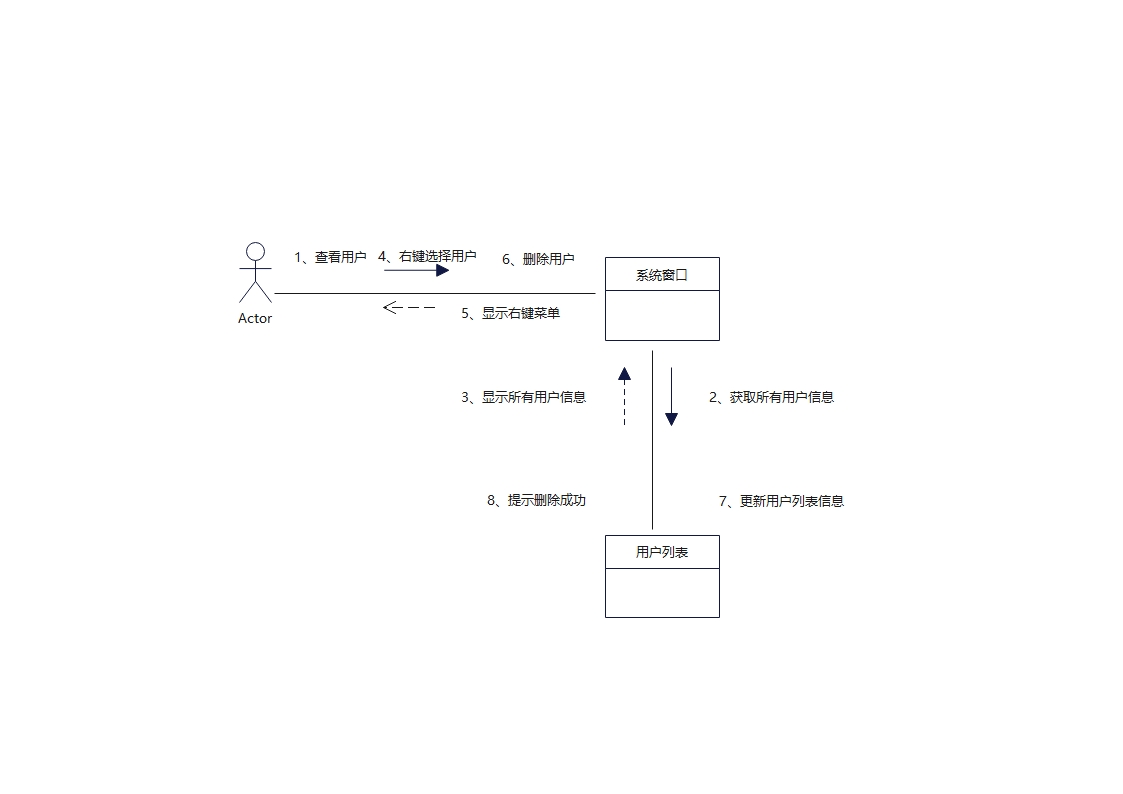


图19表示用户收藏音乐：

1. 用户打开音乐，获取所有用户信息并显示出来，选择其中一首音乐并下载
2. 更新音乐列表即可显示音乐下载成功。

图19

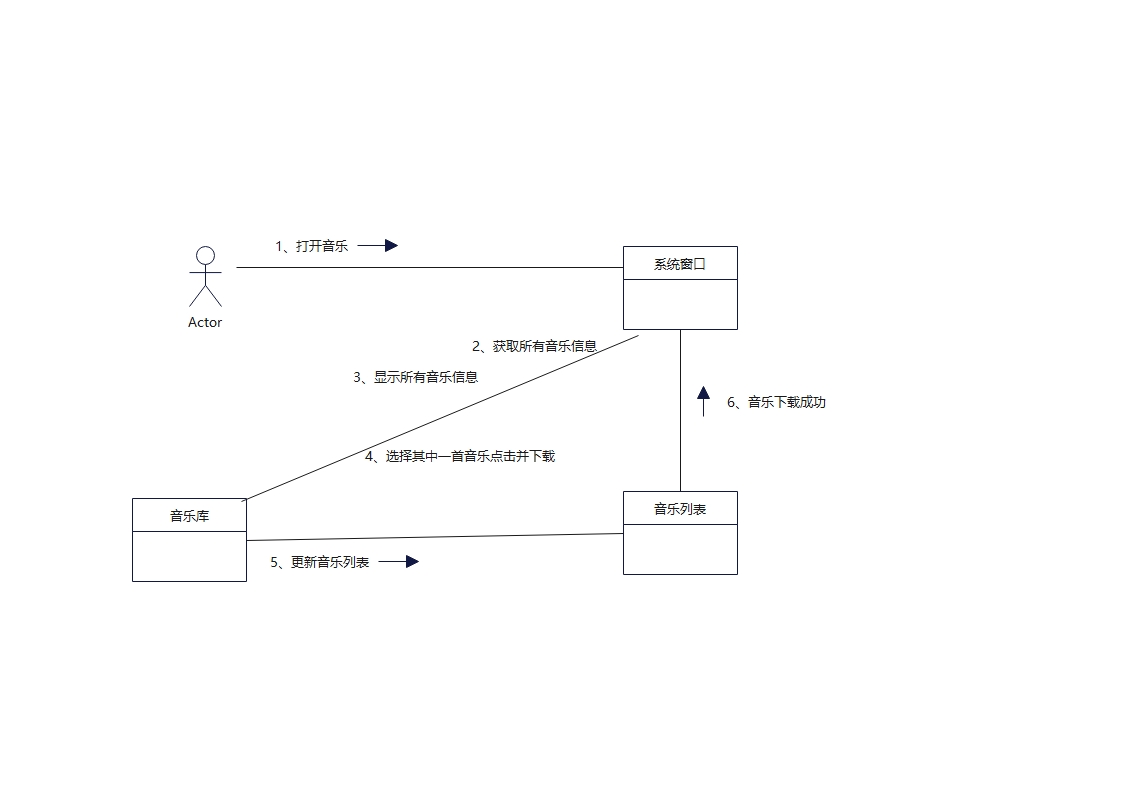
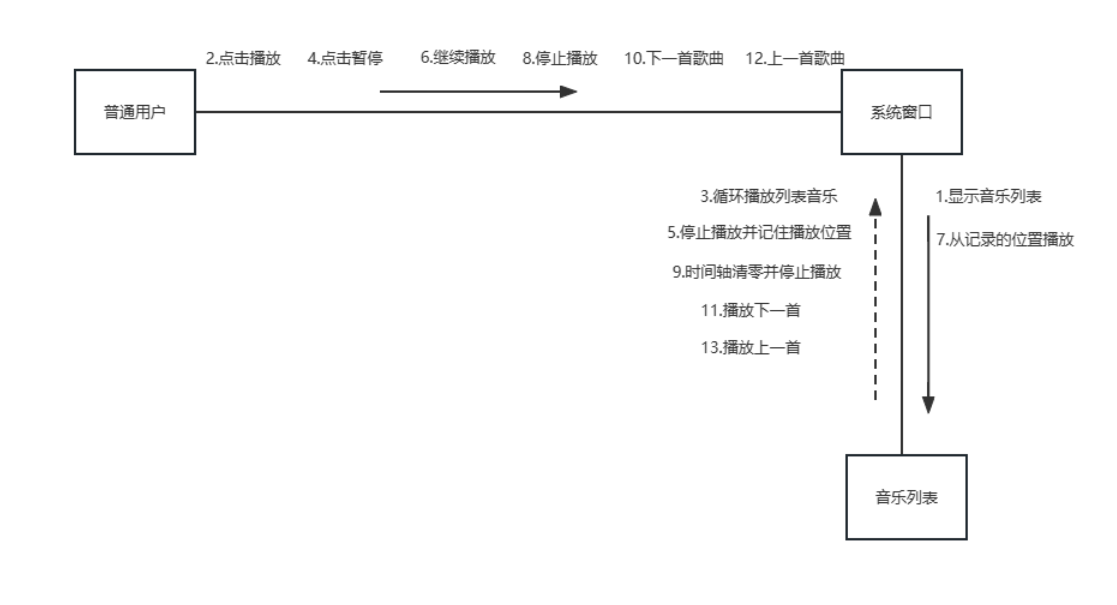


图20表示用户播放音乐：

1. 在系统窗口显示音乐列表，然后点击播放，循环播放列表，点击暂停然后停止播放并记住播放的位置
2. 继续播放（从记录的位置播放），停止播放后时间轴清零
3. “下一首歌曲”播放下一首，“上一首歌曲”播放上一首歌曲。

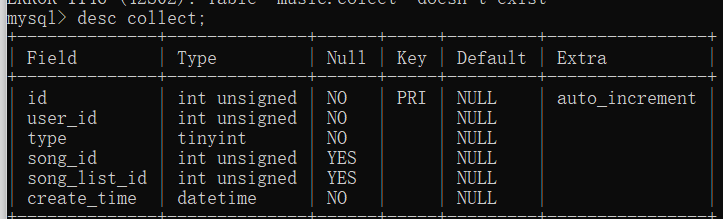
图20

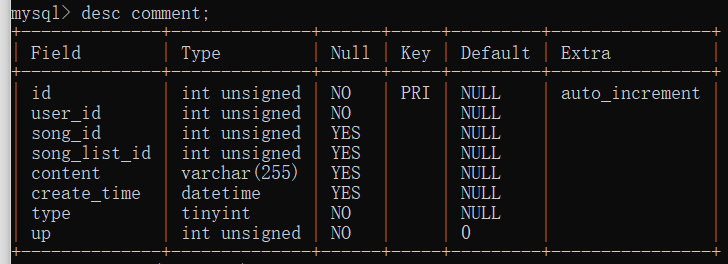


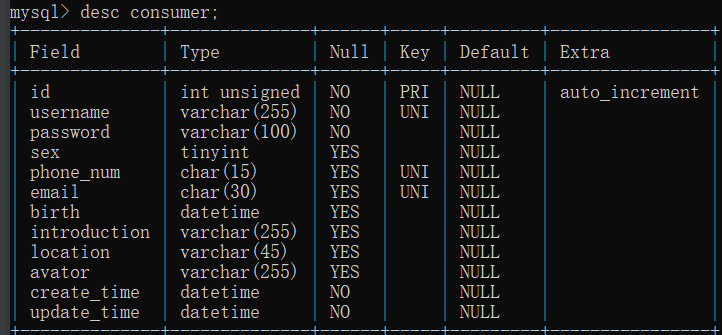
4.2数据库设计

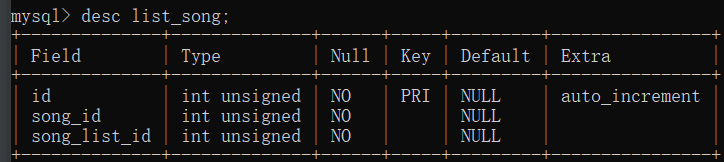
（一）表结构设计

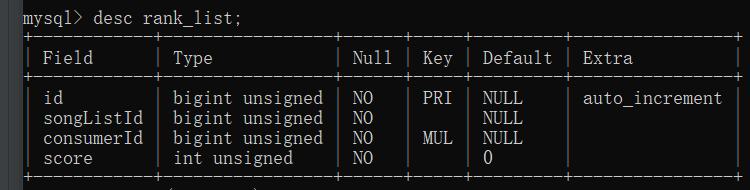
1.

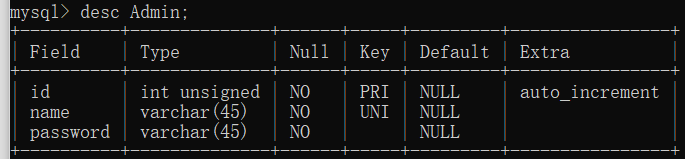


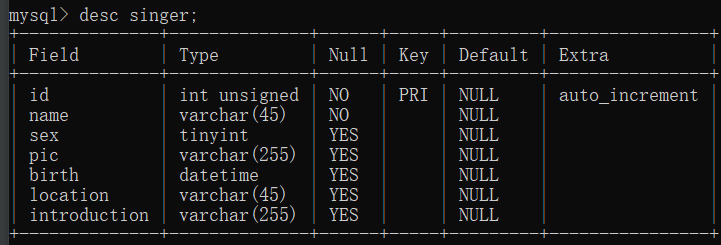


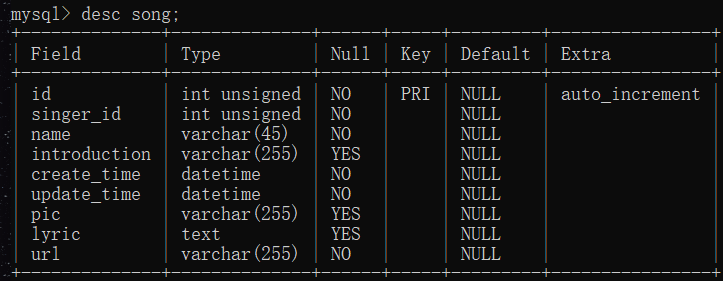


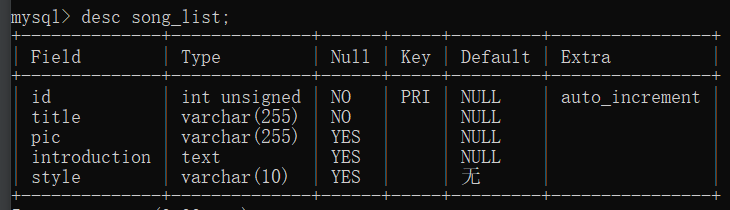












（二）类图

为了更好的描述系统中的类、接口、属性和方法等元素之间的关系。可以帮助我们更好地理解系统的架构和设计，类图中的元素之间可以存在多种关系，如继承、实现、关联、聚合等。继承表示一个类从另一个类继承属性和方法；实现表示一个类实现了多个接口；关联表示两个类之间存在某种联系；聚合表示多个类组合成一个整体。

1.图形基础

将整个系统划分为View层，Controller层，Service层，DAO层四层，使用Spring MVC负责请求的转发和视图管理，Spring实现业务对象管理，Mybatis作为数据对象的持久化引擎。

（1）后端：

**Controller层**：Controller层负责具体的业务模块流程的控制。

**DAO层：**DAO层主要是做数据持久层的工作，负责与数据库进行联络的一些任务都封装在此。

**Domain层:**Domain层负责定义私有属性和公有方法，链接数据库。

**Service层：**Service层主要负责业务模块的逻辑应用设计。

**mapper：**存放XML文件,有关数据交换和配置文件 。

**application.properties** ：用于存放程序的各种依赖模块的配置信息，比如 服务端口，数据库连接配置等。

**gitignore：**使用版本控制工具 git 的时候，设置一些忽略提交的内容。

**YinMusicApplication.java：**SpringBoot 程序执行的入口，执行该程序中的 main 方法，启动当前SpringBoot项目。

（2）前端：

**View层：**View层与控制层结合比较紧密，需要二者结合起来协同工发。View层主要负责前台jsp页面的表示。

**dist和assets：**存放静态资源。如图片、CSS、JavaScript 等 。

2.图形解释

（1）类图是静态图的一种，我们为yin音乐网站设计了九个类，分别为admin,comment,ranklist,song,songlist,comsumer，singer,list\_song,collect。其中song与comment，songlist与comment,comsumer与songlist,singer与song为一对多的关系，song与list\_song，songlist与list\_song为多对一的关系，ranklist依赖于song,collect依赖于list\_song。

（2）-： private +： public

冒号前是方法名/变量名（根据有无括号区分），冒号后是返回参数/变量类型（根据有无括号区分），如果没有冒号的话表示方法返回空（也有人通过：void表示返空）。

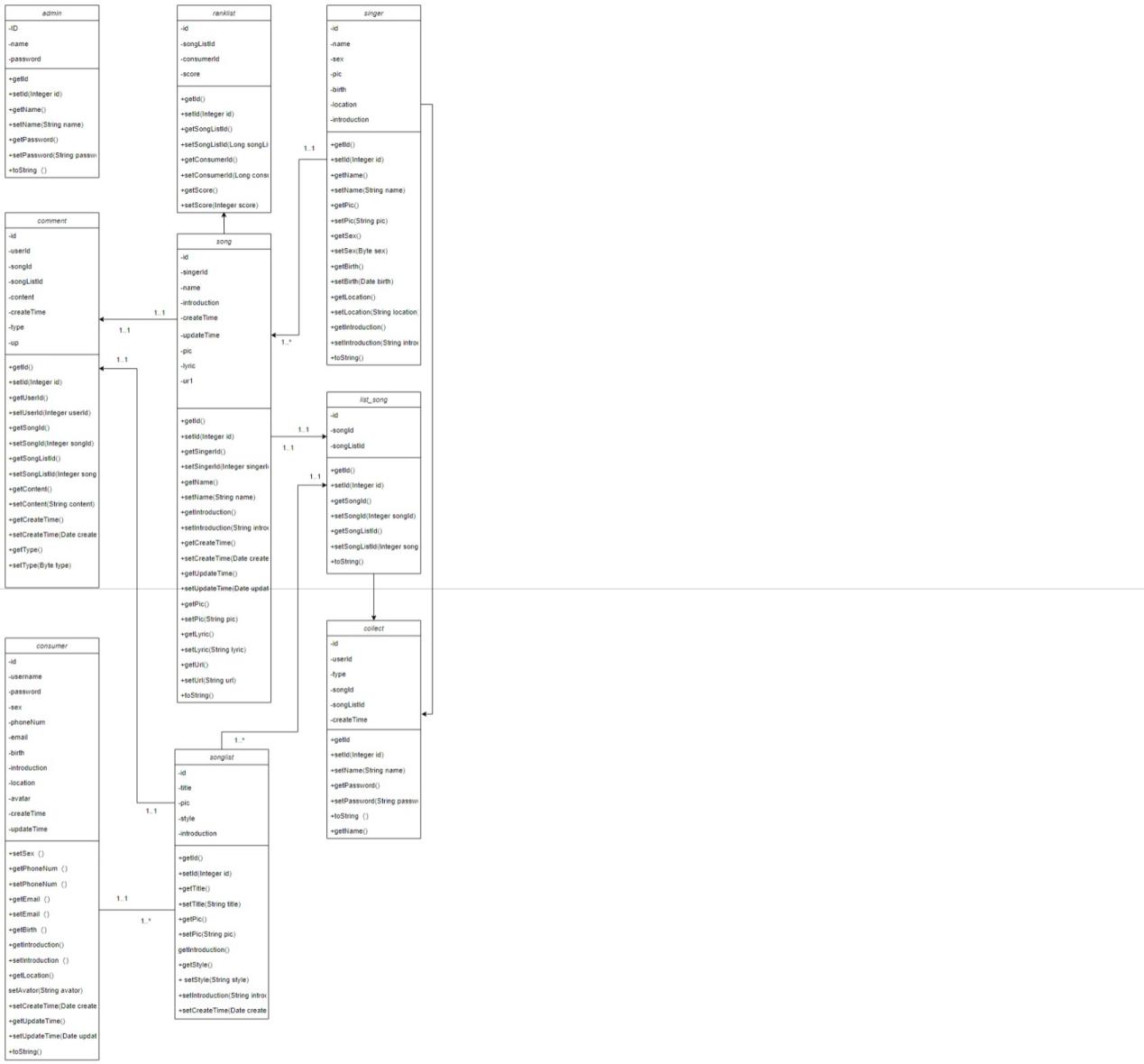
（3）1:表示一个集合中的一个对象对应另一个集合中一个对象。

0..\*:表示一个集合中的一个对象对应另一个集合中的0个或多个对象。(可以不对应)

1..:表示一个集合中的一个对象对应另一个集合中的一个或多个对象。(至少对应一个)

\*:表示一个集合中的一个对象对应另一个集合中的多个的对象。

类图如图所示：



4.3实体类设计

（一） 显示实体类的属性。

1.Admin实体类

private Integer id; //定义管理员ID

private String name; //定义管理员的名字

private String password; //定义管理员的密码

2.Collect实体类

id（Integer类型）： //表示收藏记录的唯一标识符。

userId（Integer类型）： //表示收藏记录所属的用户的唯一标识符。

type（Byte类型）： //表示收藏的类型。

songId（Integer类型）： //表示被收藏的歌曲的唯一标识符。

songListId（Integer类型）：//表示被收藏的歌单的唯一标识符。

createTime（Date类型）： //表示收藏记录的创建时间。

3.Commentt实体类

private Integer id; //评论的唯一标识符（主键）

private Integer userId; // 评论的发表者的用户ID。

private Integer songId; // 评论所针对的歌曲的ID

private Integer songListId; // 评论所属的歌单的ID

private String content; // 评论的内容

private Date createTime; // 评论的创建时间

private Byte type; // 评论的类型，

private Integer up; // 评论的类型，可能是一个表示评论类型的字节值

4.Consumer实体类

private Integer id; // 普通用户的唯一标识符（主键）

private String username; // 普通用户的用户名

private String password; // 普通用户的密码

private Byte sex; // 普通用户的性别，

private String phoneNum; // 普通用户的手机号码

private String email; // 普通用户的电子邮件地址

private Date birth; // 普通用户的出生日期

private String introduction; // 普通用户的简介或个人介绍

private String location; // 普通用户的所在地

private String avator;···· // 普通用户的头像文件路径或URL

private Date createTime; // 普通用户的创建时间

private Date updateTime; // 普通用户的更新时间

5.ListSong实体类

private Integer id; // 歌单的唯一标识符（主键）

private Integer songId; // 歌曲的唯一标识符

private Integer songListId; // 关联的歌单的ID。

6.RankList实体类

private Long id; // RankList 的唯一标识符（主键），

private Long songListId; //歌单的ID

private Long consumerId; //普通用户的ID

private Integer score; //普通用户的评分

7.Singer实体类

private Integer id; // 歌手的唯一标识符（主键）

private String name; // 歌手的姓名

private Byte sex; // 歌手的性别

private String pic; // 歌手的照片路径或URL，

private Date birth; // 歌手的出生日期

private String location; // 歌手的所在地

private String introduction; // 歌手的简介或介绍

8.Song实体类

private Integer id; // 歌曲的唯一标识符（主键）

private Integer singerId; // 歌曲所属的歌手的ID。

private String name; // 歌曲的名称

private String introduction ;// 歌曲的简介或介绍

private Date createTime; // 歌曲的创建时间

private Date updateTime; // 歌曲的更新时间

private String pic; // 歌曲的封面图片路径或URL

private String lyric; // 歌曲的歌词

private String url; // 歌曲的音频文件路径或URL

9.SongList实体类

private Integer id; // SongList 的唯一标识符（主键）

private String title; // 歌单的标题或名称

private String pic; // 歌单的封面图片路径或URL

private String style; // 歌单的风格或类型

private String introduction; // 歌单的简介或介绍

4.4项目的配置文件

（一）配置文件

spring.datasource.url=jdbc:mysql://localhost:3306/music?serverTimezone=Asia/Shanghai&characterEncoding=utf8

spring.datasource.username=root

spring.datasource.password=password

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

mybatis.typeAliasesPackage=com.example.yin.domain

mybatis.mapperLocations=classpath:mapper/\*.xml

spring.jackson.date-format=yyyy-MM-dd HH:mm:ss

spring.jackson.time-zone=GMT+8

（二）端口号

server.port=8888

（三）热部署生效

spring.devtools.restart.enabled=true

（四）设置重启的目录

spring.devtools.restart.additional-paths=src/main/java

#classpath目录下的WEB-INF文件夹内容修改不重启

spring.devtools.restart.exclude=WEB-INF/\*\*

# 关闭CONDITIONS EVALUATION REPORT及自动配置内容向控制台的输出

logging.level.org.springframework.boot.autoconfigure=ERROR

logging.level.org.apache.tomcat=ERROR

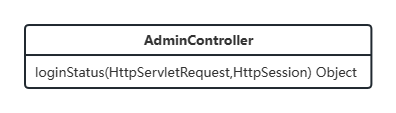
logging.level.org.apache.catalina=ERROR

4.5前后端交互

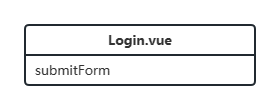
（一）管理员系统和后端交互

1.管理员登录

登录管理员用户，验证登录，根据用户输入得到的用户名和密码查询数据库，验证该用户是否存在与数据库中。

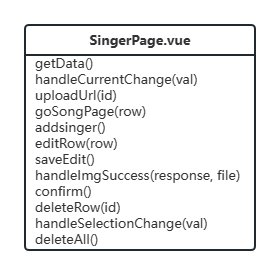
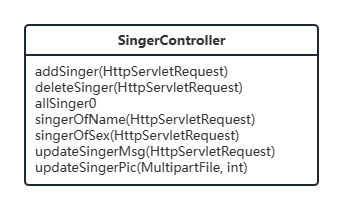


在Login.vue中使用submitForm提交表单的方法，包括发送登录请求、



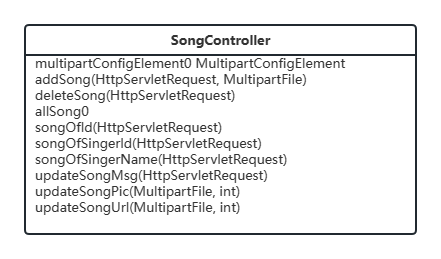
2.歌手管理

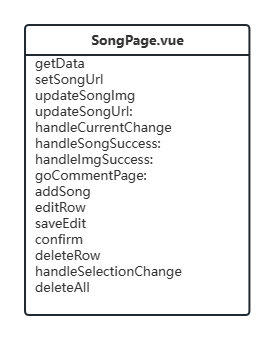
在SingerService里面定义查找歌手信息、添加歌手信息，更新歌手信息，更新头像，根据歌手ID删除歌手信息。然后在SingerPage.vue中实现，并能与数据库交互，实时更新数据库中的信息和获取数据库中的信息。



3.歌曲管理

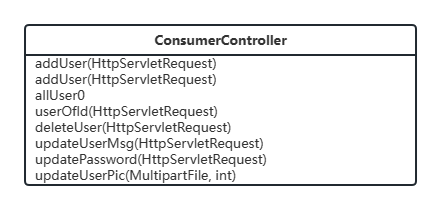
在SongController里面可以查找歌曲信息，添加歌曲，更新歌曲信息，更新图片，更新歌曲文件，删除歌曲信息。然后在SongerPage.vue中实现，并能与数据库交互，实时更新数据库中的信息和获取数据库中的信息。

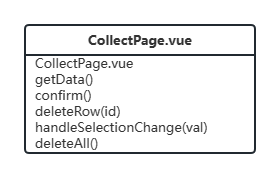


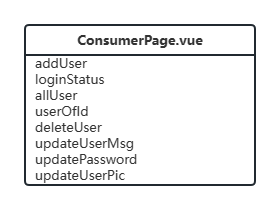


4.用户管理

通过COnsumerController里面课更新用户信息，查找用户信息，添加用户信息，更新用户头像，更新用户信息，删除用户信息，收藏歌曲信息。然后在CollectPage.vue和ConsumerController中实现，并能与数据库交互，实时更新数据库中的信息和获取数据库中的信息。

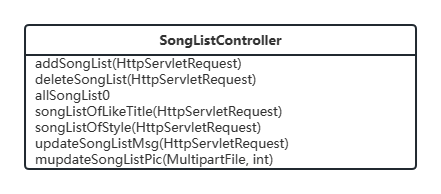


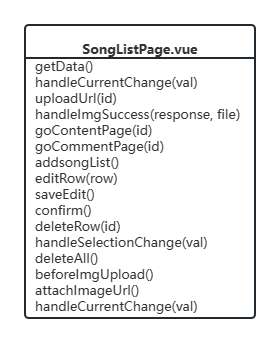




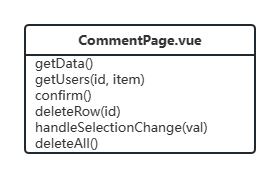
5.歌单管理

在SongListController中查找歌单信息，依据查找到的歌单信息添加歌单信息，更新歌单头像，更新歌单信息，删除歌单信息。然后在CollectPage.vue和ConsumerController中实现，并能与数据库交互，实时更新数据库中的信息和获取数据库中的信息。



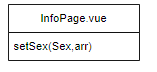


歌单评论在CommentPage.vue中实现可以获取评论数据，删除评论。



6.系统首页

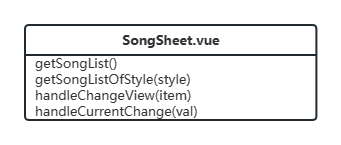
在InfoPage.vue中统计了用户数、歌曲数、歌手数、歌单数。



（二）普通用户和后端交互

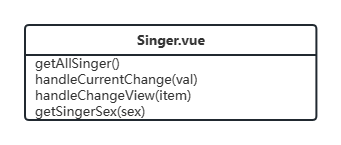
1.查看歌单信息

在SongSheet.vue中查看全部歌单，或者查看某类歌单信息。

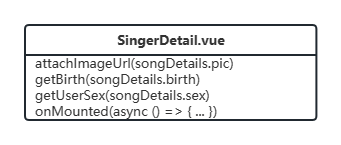


2.歌手信息

在Singer.vue中实现查看所有歌手信息、查看某类歌手信息



在SingerDetail.vue中实现获取歌手信息：头像、生日、性别…



3.排行榜

在RankListController中查看歌单信息，查看歌曲信息，歌单评分，在数据库中查询，查看是否交互成功。



4.用户操作：播放歌曲，在song.ts中实现循环播放、上一首、暂停、下一首、音量、收藏、下载，点击歌曲头像查看歌词。



5.收藏操作，在controller中收藏歌曲，取消收藏，是否收藏，更新收藏的歌曲，在数据库中查看是否取消收藏成功。



6.评论操作，在CommentController中显示所有的评论信息，添加评论，删除自己的评论



# 项目结果

（一）结果展示





