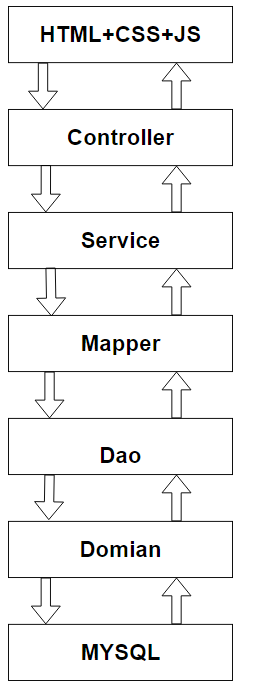
**项目设计**

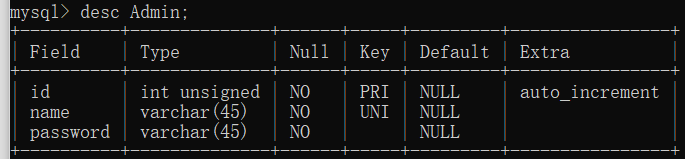
1. **分层（总体）设计**

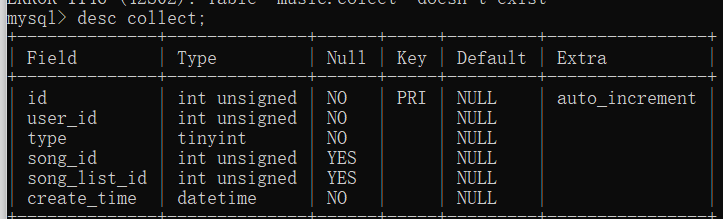
先编写domain 层定义数据库中的属性，先编写domain 层,然后在dao层写实现方法的mysql语句，然后实现service接口，继续写service的实现类，然后在controller写实现方法，然后配置前后端交互文件，然后在前端实现并返回数据库实现交互。

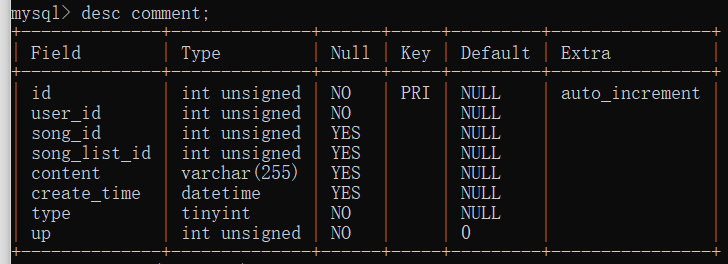


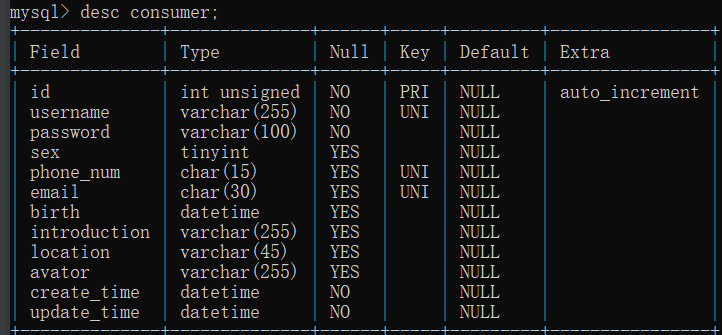
**2、数据库设计**

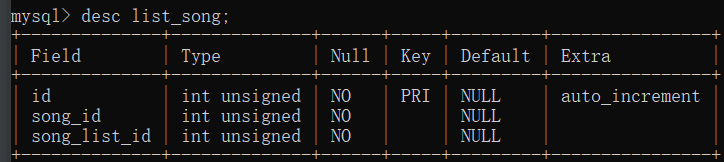
**表结构设计**

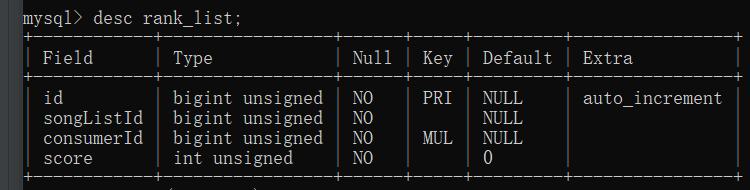
****

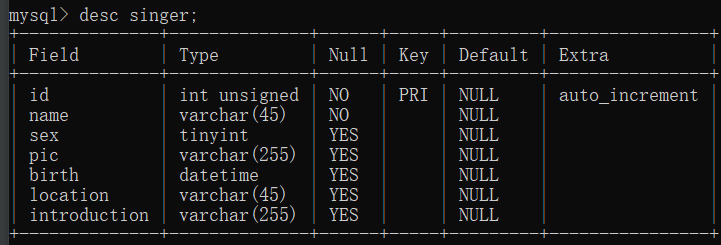
****

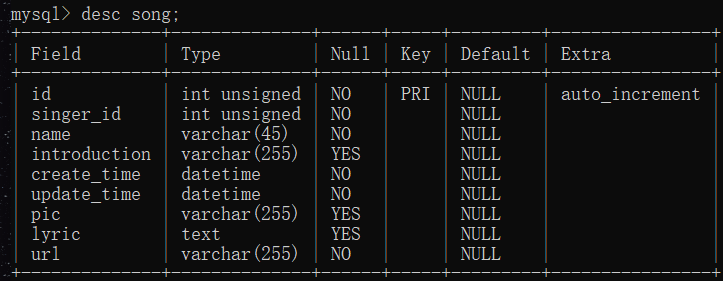
****

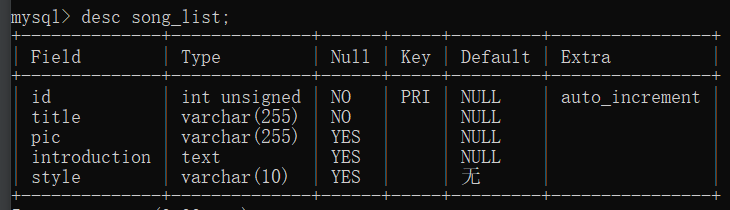
****

****

****

****

****

****

## 3、实体类设计

  此处只是显示实体类的属性。

（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

（6）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

（7）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

（8）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、项目的配置文件**

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

#端口号

server.port=8888

#热部署生效

spring.devtools.restart.enabled=true

#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

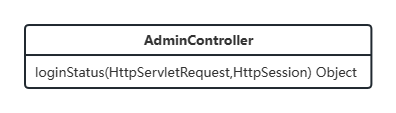
logging.level.org.apache.catalina=ERROR

## 5、前后端交互

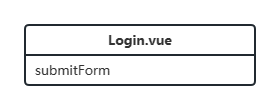
## 1.管理员系统和后端交互

（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。

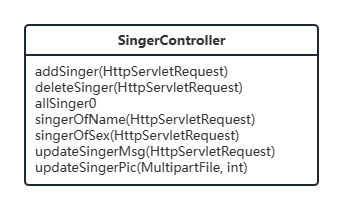


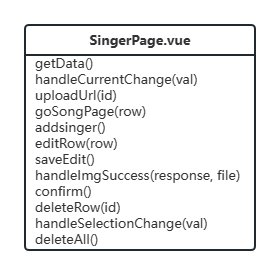
在Login.vue中使用submitForm提交表单的方法，包括发送登录请求、



（2）歌手管理

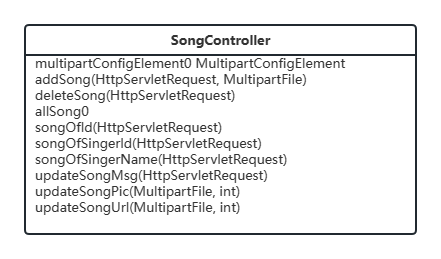
在SingerService里面定义查找歌手信息、添加歌手信息，更新歌手信息，更新头像，根据歌手ID删除歌手信息。然后在SingerPage.vue中实现，并能与数据库交互，实时更新数据库中的信息和获取数据库中的信息。

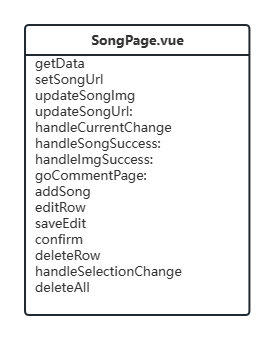




歌曲管理

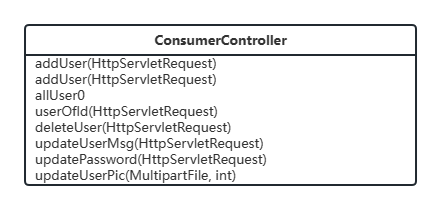
在SongController里面可以查找歌曲信息，添加歌曲，更新歌曲信息，更新图片，更新歌曲文件，删除歌曲信息。然后在SongerPage.vue中实现，并能与数据库交互，实时更新数据库中的信息和获取数据库中的信息。

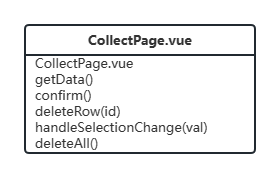


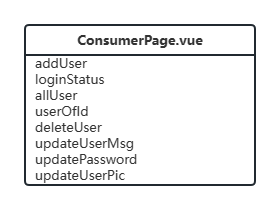


（3）用户管理

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。

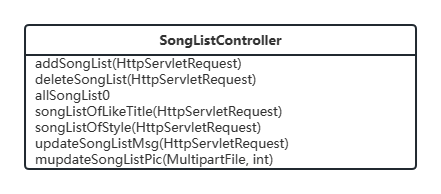


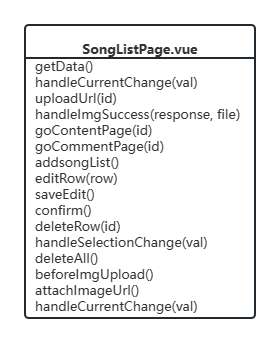




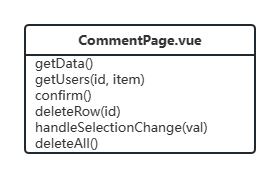
（4）歌单管理

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。



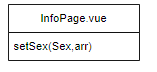


歌单评论在CommentPage.vue中实现可以获取评论数据，删除评论。



（5）系统首页

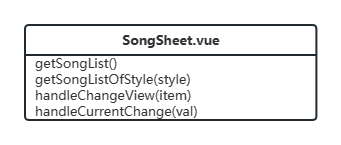
在InfoPage.vue中统计了用户数、歌曲数、歌手数、歌单数。



## 2. 普通用户和后端交互

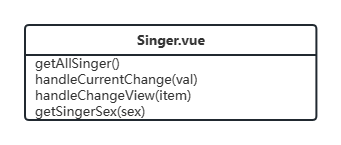
（1）查看歌单信息

在SongSheet.vue中查看全部歌单，或者查看某类歌单信息。

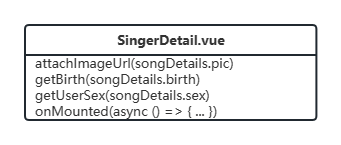


（2）歌手信息

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。



（3）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。



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。



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

