天津工业大学计算机科学与技术学院

实训总结报告



　项目名称:

　合作培养企业：天津市融创软通科技股份有限公司

　专业方向：大数据方向

　项目负责人：何晶

　专业班级: 软件z1802

姓名：刘博

　实训起止日期：2020.6.15-2020.7.15

**1 引言**

1.1项目概述

在当下这个信息爆炸的时代，互联网作为人们工作、生活、学习都无法脱离的平台，每天都会产生海量数据，音乐、电影、书籍、评论种种信息虽然琐碎，担当大量获取分析后将会得出许多有用的结论。

本项目采用了前后端分离的开发方式，前段采用Vue框架进行开发，后端使用SpringBoot框架开发，通过scrapy爬虫机制针对豆瓣电影、音乐、图书和当当图书、B站动漫评论进行数据收集并分析，结果保存与远程mariaDB数据库中。提供简洁明了的数据查询服务，便于查询所需条件的数据，也提供关于数据内容的分析服务，并将其以图表的方式简洁直观的展示出来，方便用户观看。用户可根据分析所获结果，快捷的寻找满意的电影、音乐、书籍等资源，此外团队远程开发利用代码托管平台gitee和代码版本控制器git。

为便于完成项目，本次实训中企业为我们提供了scrapy爬虫基础教学、SSM服务器短开发框架，前端技术以及所需要的模版、软件

1.2 需求分析

随着社会发展，人们的生活质量越来越高，伴随着各种娱乐产业的新增，每天都有海量数据生成。单单只看其中一些数据，是毫无意义的，担当将当量数据集中金雄分析，就能从中取得许多信息，这也是大数据的意义所在。

该项目将爬取豆瓣电影、音乐、图书和当当图书、B站动漫评论的数据，然后所获得的信息进行分析，得到诸如最畅销的书籍、最受欢迎的导演等等数据，之后以表格的形式将分析直观的展示出来，以此为客户挑选电影、音乐、动漫、书籍时提供便捷。

1.3 运行环境

硬件环境

1.笔记本1台

2.内存：4GB以上。

4.局域网畅通

软件环境

1. 操作系统：Windows 10。

2. 文档编写工具：Microsoft office 2016（word excel powerpoint visio）。

3. 开发工具：IDEA2019，PyCharm2019，MySQL5.7.x，WebStorm2019、Navicat、ECharts。

4. 版本控制工具：Git/码云。

**2 项目设计**

2.1 设计思路

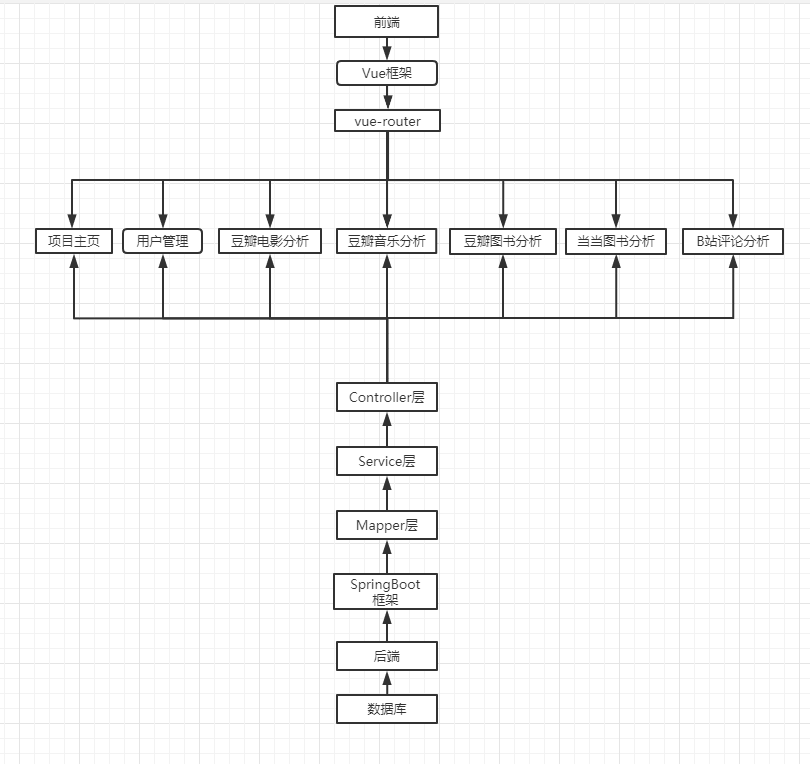


Figure 1设计思路

系统大致可以分为三层：Vue前端、SSM后端、以及MySql数据库三层。Mysql数据库存储通过scrapy爬取到的数据，用以进行后端分析和前端展示；SSM后端主要有三层：Controller层（控制层，负责具体模块的业务流程控制，需要调用service逻辑设计层的接口来控制业务流程）、Mapper层（对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的，主要实现一些增删改查操作）、Service层（业务service层，给controller层的类提供接口进行调用。），由此向前端提供接口，完成交互；Vue前端通过接口获得相应数据，完成展示

2.2 模块功能介绍

模块一：系统基础管理模块

1) 系统用户管理（包括平台管理人员的建立维护以及平台外部注册用户的基础信息管理等功能）。

2) 平台主界面设计及开发。

3) 管理首页设计与开发，图形化当前系统情况，包括已注册用户数量、已贮存影片（书籍、评论、音乐）数量、已分析的评论数、各分类的检索前十名汇总等信息；

模块二：1) 平台允许通过相关页面进行个人用户的注册。

2) 能够对个人信息进行更新。

3) 允许重置个人密码。

模块三：豆瓣图书模块

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展

模块四：当当图书模块

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展出

模块五：B站评论模块

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展出

模块六：豆瓣音乐模块

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展出

模块七：豆瓣电影模块

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展出

2.3 模块结构图

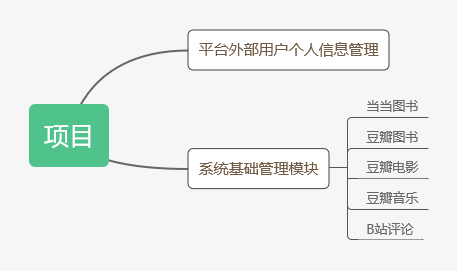
****

Figure 2模块结构图

项目大致可分为七个模块，包含两个主模块，五个小模块。主模块包括平台外部用户个人信息管理，主要负责用户注册，修改信息；系统基础管理模块负责平台页面框架的搭建，其中包含着当当图书、豆瓣图书、豆瓣电影、豆瓣音乐、B站评论五个模块的内容

2.4 程序流程图

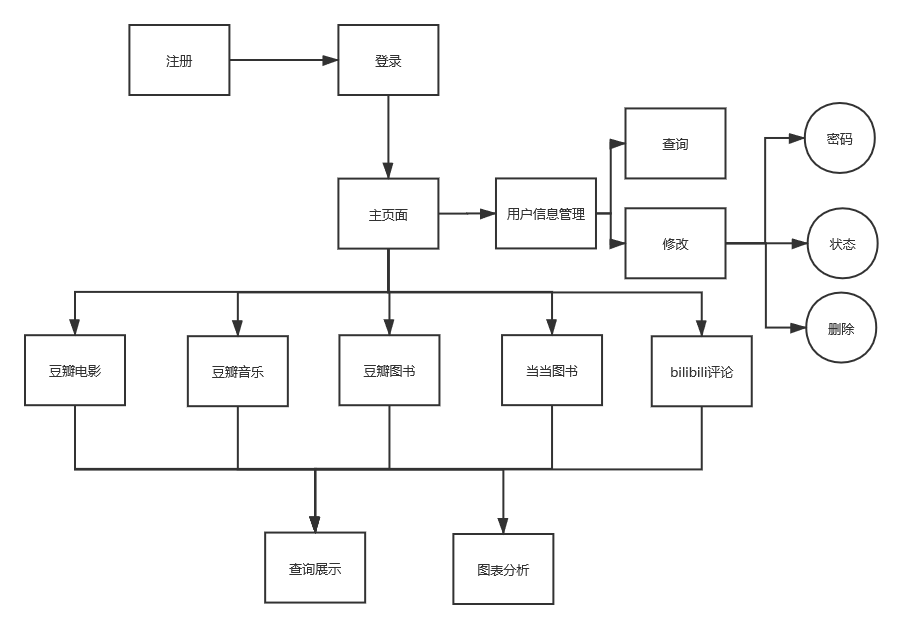


Figure 3程序流程图

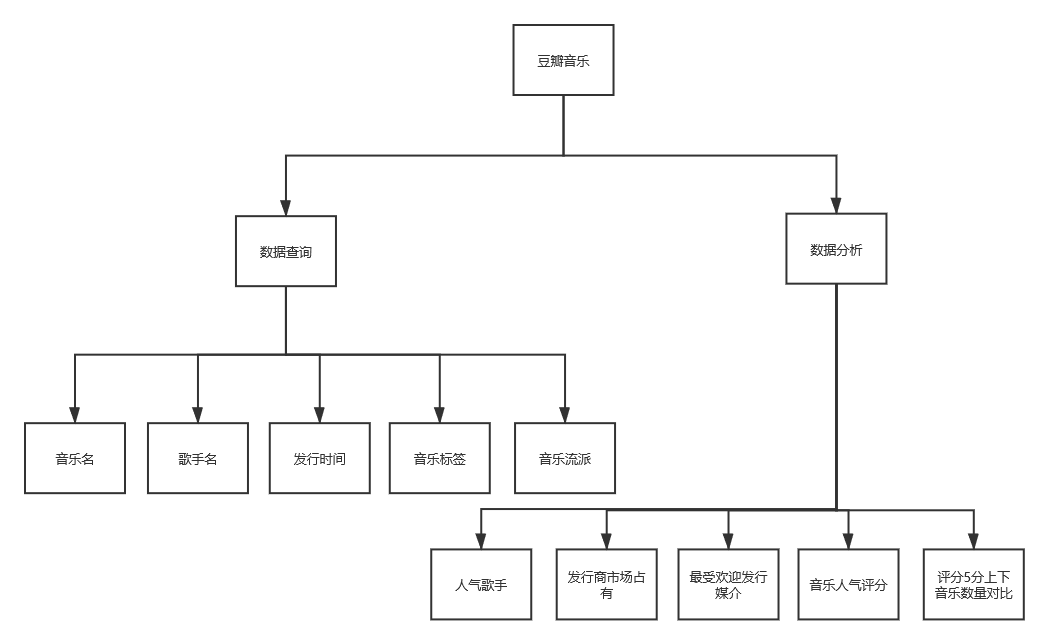


Figure 4豆瓣音乐——程序流程

2.5 功能设计分工

数据爬取：豆瓣电影、豆瓣音乐、豆瓣图书、当当图书、B站评论

前端框架：主界面搭建、登录、注册、用户管理、豆瓣电影、豆瓣音乐、豆瓣图书、当当图书、B站评论

后端：登录、注册、用户管理、豆瓣电影、豆瓣音乐、豆瓣图书、当当图书、B站评论

**3 详细设计**

远程Mariadb数据库，由scrapy爬取数据后存入其中，主要负责：

user表：用户表，包含用户的用户名、密码、昵称、邮箱、地址、年龄、性别、状态

spt\_music表：音乐数据表，包含音乐名、歌手名、发行商、音乐种类、流派、发行媒介、发行时间、音乐简介、歌曲推荐、包含歌曲等

scrapy爬虫，包括：

Scheduler(调度器)：它负责接受引擎发送过来的Request请求，并按照一定的方式进行整理排列，入队，当引擎需要时，交还给引擎。

Downloader（下载器）：负责下载Scrapy Engine(引擎)发送的所有Requests请求，并将其获取到的Responses交还给Scrapy Engine(引擎)，由引擎交给Spider来处理。

Spider（爬虫）：它负责处理所有Responses,从中分析提取数据，获取Item字段需要的数据，并将需要跟进的URL提交给引擎，再次进入Scheduler(调度器)。

Item Pipeline(管道)：它负责处理Spider中获取到的Item，并进行后期处理（详细分析、过滤、存储等）的地方。

Downloader Middlewares（下载中间件）：一个可以自定义扩展下载功能的组件。

Spider Middlewares（Spider中间件）：一个可以自定扩展和操作引擎和Spider中间通信的功能组件。

SSM后端：

Controller层（控制层，负责具体模块的业务流程控制，需要调用service逻辑设计层的接口来控制业务流程）

Mapper层（对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的，主要实现一些增删改查操作）

Service层（业务service层，给controller层的类提供接口进行调用。）

主要包括：用户管理所需接口，包括登录、注册、修改用户信息；豆瓣音乐信息，包括带有关键字（音乐名、歌手、时间、标签、流派）的查询，可视化数据分析

VUE前端：

前端框架，主要负责豆瓣音乐关键字查询、可视化图表分析界面

**4 实现方案**

SSM后端：

一、用户管理

Controller层：负责用户管理模块的业务流程，为前端提供接口，调用UserService逻辑设计层的接口来控制业务流程，包括登录、注册、修改信息、数据展示

@Controller  
@RequestMapping(**"/logic/user"**)  
**public class** UserController {  
  
 @Autowired  
 **private** UserService **service**;  
  
@RequestMapping(**"/currentUser"**)  
 @ResponseBody  
 **public** OneData currentUser(String uid){  
 OneData data = **new** OneData();  
 data.setCode(2000);  
 data.setMsg(**"登录状态异常"**);  
  
 User result = **service**.getCurrentUser(uid);  
 **if** (result != **null**){  
 data.setData(result);  
 data.setCode(1000);  
 data.setMsg(**"登陆状态成功"**);  
 }  
  
 **return** data;  
 }  
  
 @RequestMapping(**"/login"**)  
 @ResponseBody  
 **public** TableData login(String uid, String pwd, Model model){  
 User currentUser =**service**.login(uid, pwd);  
 TableData data =**new** TableData();  
 **if** (currentUser != **null**){  
*// model.addAttribute("currentUser", currentUser);* data.setData(**null**);  
 data.setCode(1000);  
 data.setMsg(**"登录成功"**);  
 data.setCount(0);  
 }  
 **else** {  
 data.setData(**null**);  
 data.setCode(2000);  
 data.setMsg(**"登陆失败"**);  
 data.setCount(0);}  
  
 **return** data;  
 }  
  
  
  
 @RequestMapping(value = **"/add"**)  
 @ResponseBody  
 **public** TableData register(User user){  
 TableData data = **service**.register(user);  
 **return** data;  
 }  
  
 @RequestMapping(value=**"/search"**)  
 @ResponseBody  
 **public** TableData search(String uid,String nickName,**int** state , **int** page, **int** limit){  
 TableData data = **new** TableData();  
 List<User> res =**service**.search(uid, nickName, state, page, limit);  
 data.setCode(1000);  
 data.setMsg(**""**);  
 data.setData(res);  
 data.setCount(**service**.searchCount(uid, nickName, state));  
 **return** data;  
 }  
  
 @RequestMapping(value = **"/pwdreset"**)  
 @ResponseBody  
 **public** OneData pwdReset(String uid, String newPwd){  
 OneData data = **new** OneData();  
  
 data.setCode(2000);  
 data.setMsg(**"修改失败"**);  
 data.setData(**null**);  
  
 **if** (**service**.pwdReset(uid,newPwd)==**true**){  
 data.setCode(1000);  
 data.setMsg(**"修改成功"**);  
 }  
 **return** data;  
 }  
  
 @RequestMapping(value = **"/staterest"**)  
 @ResponseBody  
 **public** OneData stateReset(String uid, Integer state){  
 OneData data = **new** OneData();  
  
 data.setCode(2000);  
 data.setMsg(**"修改失败"**);  
 data.setData(**null**);  
  
 **if** (**service**.stateReset(uid, state) == **true**){  
 data.setCode(1000);  
 data.setMsg(**"修改成功"**);  
 }  
 **return** data;  
 }  
  
 @RequestMapping(value = **"/deleteuser"**)  
 @ResponseBody  
 **public** OneData deleteUser(String uid){  
 OneData data = **new** OneData();  
  
 data.setCode(2000);  
 data.setMsg(**"删除失败"**);  
 data.setData(**null**);  
  
 **if** (**service**.deleteUser(uid)){  
 data.setCode(1000);  
 data.setMsg(**"删除成功"**);  
 }  
 **return** data;  
 }  
  
}

Service层：

给UserController层的类提供接口进行调用，通过调用UserMapper中的sql语句返回数据信息

@Service  
**public class** UserServiceImpl **implements** UserService {  
  
 @Autowired  
 **private** UserMapper **mapper**;  
  
 @Override  
 **public** User login(String uid, String pwd) {  
  
 User user = **mapper**.login(uid,pwd);  
 **if** (user == **null** || user.getState()==0)  
 user=**null**;  
  
 **return** user;  
 }  
  
 @Override  
 **public** TableData register(User user) {  
  
 TableData data = **new** TableData();  
 **if** (**mapper**.selectByUid(user.getUid()) != **null**){  
 data.setCode(2000);  
 data.setData(**null**);  
 data.setMsg(**"用户名已存在"**);  
 **return** data;  
 }  
user.setState(1);  
 **mapper**.register(user);  
 data.setMsg(**"注册成功"**);  
 data.setCode(1000);  
 data.setData(**null**);  
 **return** data;  
  
 }  
  
 @Override  
 **public** List<User> search(String uid, String nickName,Integer state, Integer page, Integer limit) {  
  
 **if** (uid!=**null** && !**""**.equals(uid.trim()))  
 uid = **"%"**+uid+**"%"**;  
  
 **if** (nickName!=**null** && !**""**.equals(nickName.trim()))  
 nickName = **"%"**+nickName+**"%"**;  
  
 **if** (page>0 && limit>0)  
 **return mapper**.selectByWhere(uid, nickName, state, (page-1)\*limit, limit);  
  
 **return mapper**.selectByWhere(uid ,nickName ,state, **null**, **null**);  
 }  
  
 @Override  
 **public boolean** pwdReset(String uid,String newPwd) {  
  
 **if** (**mapper**.updatePwd(uid,newPwd)==1)  
 **return true**;  
  
 **return false**;  
 }  
  
 @Override  
 **public int** searchCount(String uid, String nickName, **int** state) {  
 **return mapper**.countSeletcByWhere(uid, nickName, state);  
 }  
  
 @Override  
 **public boolean** stateReset(String uid, Integer state) {  
 **if** (**mapper**.stateReset(uid, state) == 1)  
 **return true**;  
 **return false**;  
 }  
  
 @Override  
 **public boolean** deleteUser(String uid) {  
 **if** (**mapper**.deleteUser(uid) == 1){  
 **return true**;  
 }  
  
 **return false**;  
 }  
  
 @Override  
 **public** User getCurrentUser(String uid) {  
  
 User currentUser = **mapper**.selectByUid(uid);  
  
 **if** (currentUser != **null**){  
 **return** currentUser;  
 }  
  
 **return null**;  
 }  
  
}

Mapper层：

对数据库进行数据持久化操作，他的方法语句是直接针对用户数据的操作，包括查询用户、新增用户、删除用户、修改用户

@Repository  
@Mapper  
**public interface** UserMapper {  
  
 @Select(**"select \* from user where uid = #{uid} and pwd = #{pwd}"**)  
 **public** User login(@Param(**"uid"**) String uid, @Param(**"pwd"**) String pwd);  
  
 @Select(**"select \* from user where uid = #{uid}"**)  
 **public** User selectByUid(@Param(**"uid"**) String uid);  
  
 @Insert(**"insert into user( uid,pwd,nick\_name,email,address,sex,age,state) "**+  
 **"values(#{user.uid},#{user.pwd},#{user.nickName},#{user.email},#{user.address},#{user.sex},#{user.age},#{user.state})"**)  
 @Options(useGeneratedKeys = **true**,keyProperty = **"id"**)  
 **public int** register(@Param(**"user"**) User user);  
  
 @Update(**"update user set pwd = #{newPwd} where uid = #{uid}"**)  
 **int** updatePwd(@Param(**"uid"**) String uid, @Param(**"newPwd"**) String newPwd);  
  
 @Select(**"<script>"** +  
 **"select \* "** +  
 **"from user"** +  
 **"<where>"** +  
 **" <if test='uid !=null and uid.length > 0'>"** +  
 **" and uid like #{uid}"** +  
 **" </if>"**+  
 **" <if test='nickName !=null and nickName.length > 0'>"** +  
 **" and nick\_name like #{nickName}"** +  
 **" </if>"**+  
 **" <if test='state !=null and state != -1'>"** +  
 **" and state = #{state}"** +  
 **" </if>"**+  
 **"</where>"** +  
 **"<if test='start != null and limit != null'>"** +  
 **"limit #{start}, #{limit}"** +  
 **"</if>"**+  
 **"</script>"**)  
 **public** List<User> selectByWhere(@Param(**"uid"**) String uid,@Param(**"nickName"**) String nickName,@Param(**"state"**) Integer state,@Param(**"start"**) Integer start,@Param(**"limit"**) Integer limit);  
  
 @Select(**"<script>"** +  
 **"select count(1)"** +  
 **"from user"** +  
 **"<where>"** +  
 **" <if test='uid !=null and uid.length > 0'>"** +  
 **" and uid like #{uid}"** +  
 **" </if>"**+  
 **" <if test='nickName !=null and nickName.length > 0'>"** +  
 **" and nick\_name like #{nickName}"** +  
 **" </if>"**+  
 **" <if test='state !=null and state != -1'>"** +  
 **" and state = #{state}"** +  
 **" </if>"**+  
 **"</where>"** +  
 **"</script>"**)  
 **public int** countSeletcByWhere(@Param(**"uid"**) String uid,@Param(**"nickName"**) String nickName,@Param(**"state"**) Integer state);  
  
 @Update(**"update user set state = #{state} where uid = #{uid}"**)  
 **int** stateReset(@Param(**"uid"**) String uid, @Param(**"state"**) Integer state);  
  
 @Delete(**"delete from user where uid = #{uid}"**)  
 **int** deleteUser(@Param(**"uid"**) String uid);  
}

豆瓣音乐

Controller层：

Controller层：负责豆瓣音乐模块的业务流程，为前端提供接口，调用MusicService逻辑设计层的接口来控制业务流程，包括查询各种条件的音乐信息

@Controller  
@RequestMapping(**"/music"**)  
**public class** MusicController {  
  
 @Autowired  
 **private** MusicService **service**;  
  
 @RequestMapping(**"/search"**)  
 @ResponseBody  
 **public** TableData search(String name,String singer,String time , String tag,String kind, **int** page, **int** limit){  
 TableData data = **new** TableData();  
 List<Music> res =**service**.search(name, singer, time, tag, kind, page, limit);  
 data.setCode(1000);  
 data.setMsg(**""**);  
 data.setData(res);  
 data.setCount(**service**.searchCount(name, singer, time, tag, kind));  
 **return** data;  
 }  
  
 @RequestMapping(**"/analysis"**)  
 @ResponseBody  
 **public** TwoData analysis(String type){  
 TwoData data = **new** TwoData();  
 data = **service**.analysis(type);  
 **return** data;  
 }  
}

Service层：

给MusicController层的类提供接口进行调用，通过调用UserMapper中的sql语句返回数据信息

@Service  
**public class** MusicServiceImpl **implements** MusicService {  
  
 @Autowired  
 **private** MusicMapper **mapper**;  
  
 @Override  
 **public** List<Music> search(String name, String singer, String time , String tag, String kind, **int** page, **int** limit) {  
  
 **if** (name!=**null** && !**""**.equals(name.trim()))  
 name = **"%"**+name+**"%"**;  
 **else** name=**null**;  
  
 **if** (singer!=**null** && !**""**.equals(singer.trim()))  
 singer = **"%"**+singer+**"%"**;  
 **else** singer=**null**;  
  
 **if** (time!=**null** && !**""**.equals(time.trim()))  
 time = **"%"**+time+**"%"**;  
 **else** time=**null**;  
  
 **if** (tag!=**null** && !**""**.equals(tag.trim()))  
 tag = **"%"**+tag+**"%"**;  
 **else** tag=**null**;  
  
 **if** (kind!=**null** && !**""**.equals(kind.trim())){  
 **switch**(Integer.*valueOf*(kind)){  
 **case** -1 :  
 kind = **null**;  
 **break**;  
 **case** 0 :  
 kind = **"%古典%"**;  
 **break**;  
 **case** 1 :  
 kind = **"%摇滚%"**;  
 **break**;  
 **case** 2 :  
 kind = **"%放克/灵歌/R&amp;B%"**;  
 **break**;  
 **case** 3 :  
 kind = **"%流行%"**;  
 **break**;  
 **case** 4 :  
 kind = **"%说唱%"**;  
 **break**;  
 **case** 5 :  
 kind = **"%民谣%"**;  
 **break**;  
 **case** 6 :  
 kind = **"%轻音乐%"**;  
 **break**;  
 **case** 7 :  
 kind = **"%原声%"**;  
 **break**;  
 **case** 8 :  
 kind = **"%电子%"**;  
 **break**;  
 **case** 9 :  
 kind = **"%拉丁%"**;  
 **break**;  
 **case** 10 :  
 kind = **"%爵士%"**;  
 **break**;  
 **case** 11 :  
 kind = **"%世界音乐%"**;  
 **break**;  
 **case** 12 :  
 kind = **"%布鲁斯%"**;  
 **break**;  
 **case** 13 :  
 kind = **"%Electronic電子%"**;  
 **break**;  
 **case** 14 :  
 kind = **"%Funk/Soul/R&amp;B%"**;  
 **break**;  
 **case** 15 :  
 kind = **"%雷鬼%"**;  
 **break**;  
 **case** 16 :  
 kind = **"%Soundtrack原聲%"**;  
 **break**;  
 **default**:  
 kind = **null**;  
 }  
 }  
  
 **if** (page>0 && limit>0)  
 **return mapper**.selectByWhere(name, singer, time, tag, kind, (page-1)\*limit, limit);  
  
 **return mapper**.selectByWhere(name, singer, time, tag, kind, **null**, **null**);  
 }  
  
 @Override  
 **public int** searchCount(String name, String singer, String time, String tag, String kind) {  
  
 **if** (name!=**null** && !**""**.equals(name.trim()))  
 name = **"%"**+name+**"%"**;  
 **else** name=**null**;  
  
 **if** (singer!=**null** && !**""**.equals(singer.trim()))  
 singer = **"%"**+singer+**"%"**;  
 **else** singer=**null**;  
  
 **if** (time!=**null** && !**""**.equals(time.trim()))  
 time = **"%"**+time+**"%"**;  
 **else** time=**null**;  
  
 **if** (tag!=**null** && !**""**.equals(tag.trim()))  
 tag = **"%"**+tag+**"%"**;  
 **else** tag=**null**;  
  
 **if** (kind!=**null** && !**""**.equals(kind.trim())){  
 **switch**(Integer.*valueOf*(kind)){  
 **case** -1 :  
 kind = **null**;  
 **break**;  
 **case** 0 :  
 kind = **"%古典%"**;  
 **break**;  
 **case** 1 :  
 kind = **"%摇滚%"**;  
 **break**;  
 **case** 2 :  
 kind = **"%放克/灵歌/R&amp;B%"**;  
 **break**;  
 **case** 3 :  
 kind = **"%流行%"**;  
 **break**;  
 **case** 4 :  
 kind = **"%说唱%"**;  
 **break**;  
 **case** 5 :  
 kind = **"%民谣%"**;  
 **break**;  
 **case** 6 :  
 kind = **"%轻音乐%"**;  
 **break**;  
 **case** 7 :  
 kind = **"%原声%"**;  
 **break**;  
 **case** 8 :  
 kind = **"%电子%"**;  
 **break**;  
 **case** 9 :  
 kind = **"%拉丁%"**;  
 **break**;  
 **case** 10 :  
 kind = **"%爵士%"**;  
 **break**;  
 **case** 11 :  
 kind = **"%世界音乐%"**;  
 **break**;  
 **case** 12 :  
 kind = **"%布鲁斯%"**;  
 **break**;  
 **case** 13 :  
 kind = **"%Electronic電子%"**;  
 **break**;  
 **case** 14 :  
 kind = **"%Funk/Soul/R&amp;B%"**;  
 **break**;  
 **case** 15 :  
 kind = **"%雷鬼%"**;  
 **break**;  
 **case** 16 :  
 kind = **"%Soundtrack原聲%"**;  
 **break**;  
 **default**:  
 kind = **null**;  
 }  
 }  
 **return mapper**.countSeletcByWhere(name, singer, time, tag, kind);  
 }  
  
 @Override  
 **public** TwoData analysis(String type) {  
 TwoData data = **new** TwoData();  
 **switch** (type){  
 **case "compare"**:  
 data.setData1(**mapper**.compare1());  
 data.setData2(**mapper**.compare2());  
 **break**;  
 **case "publisher"**:  
 data.setData1(**mapper**.publisher());  
 **break**;  
 **case "singer"**:  
 data.setData1(**mapper**.singer());  
 **break**;  
 **case "medium"**:  
 data.setData1(**mapper**.medium());  
 **break**;  
 **case "kind"**:  
 data.setData1(**mapper**.kind1());  
 data.setData2(**mapper**.kind2());  
 **break**;  
 }  
  
 data.setMsg(**null**);  
 data.setCode(1000);  
 **return** data;  
 }  
}

Mapper层：

对数据库进行数据持久化操作，他的方法语句是直接针对音乐数据的操作，主要是按照不同条件的音乐数据。返还给前端

@Repository  
@Mapper  
**public interface** MusicMapper {  
  
 @Select(**"<script>"** +  
 **"select music\_name as name, music\_singer as singer, music\_time as time, music\_rating as rate,music\_tags as tag,music\_kind as kind,music\_recommend as recommend"** +  
 **" from spt\_music"** +  
 **"<where>"** +  
 **" <if test='name !=null and name.length > 0'>"** +  
 **" and music\_name like #{name}"** +  
 **" </if>"**+  
 **" <if test='singer !=null and singer.length > 0'>"** +  
 **" and music\_singer like #{singer}"** +  
 **" </if>"**+  
 **" <if test='time !=null and time.length > 0'>"** +  
 **" and music\_time like #{time}"** +  
 **" </if>"**+  
 **" <if test='tag !=null and tag.length > 0'>"** +  
 **" and music\_tags like #{tag}"** +  
 **" </if>"**+  
 **" <if test='kind !=null and kind.length > 0'>"** +  
 **" and music\_kind like #{kind}"** +  
 **" </if>"**+  
 **"</where>"** +  
*// "<if test='start != null and limit != null'>" +  
// " limit #{start}, #{limit}" +  
// "</if>"+* **"</script>"**)  
 **public** List<Music> selectByWhere(@Param(**"name"**) String name, @Param(**"singer"**) String singer, @Param(**"time"**) String time, @Param(**"tag"**) String tag, @Param(**"kind"**) String kind, @Param(**"start"**) Integer start, @Param(**"limit"**) Integer limit);  
  
 @Select(**"<script>"** +  
 **"select count(1)"** +  
 **"from spt\_music"** +  
 **"<where>"** +  
 **" <if test='name !=null and name.length > 0'>"** +  
 **" and music\_name like #{name}"** +  
 **" </if>"**+  
 **" <if test='singer !=null and singer.length > 0'>"** +  
 **" and music\_singer like #{singer}"** +  
 **" </if>"**+  
 **" <if test='time !=null and time.length > 0'>"** +  
 **" and music\_time like #{time}"** +  
 **" </if>"**+  
 **" <if test='tag !=null and tag.length > 0'> "** +  
 **" and music\_tags like #{tag}"** +  
 **" </if>"**+  
 **" <if test='kind !=null and kind.length > 0 '>"** +  
 **" and music\_kind like #{kind}"** +  
 **" </if>"**+  
 **"</where>"** +  
 **"</script>"**)  
 **public int** countSeletcByWhere(String name, String singer, String time, String tag, String kind);  
  
 @Select(**"select music\_kind as type, count(\*) as count from spt\_music where music\_rating+0 > 5 group by music\_kind order by music\_kind"**)  
 **public** List<MusicData> compare1();  
  
 @Select(**"select music\_kind as type, count(music\_kind) as count from spt\_music where music\_rating-4<=5 group by music\_kind order by music\_kind"**)  
 **public** List<MusicData> compare2();  
  
 @Select(**"select music\_publisher as type, count(\*) as count from spt\_music group by music\_publisher order by count desc limit 0,40"**)  
 **public** List<MusicData> publisher();  
  
 @Select(**"select music\_singer as type, sum(music\_votes) as count from spt\_music group by music\_singer "**)  
 **public** List<MusicData> singer();  
  
 @Select(**"select music\_medium as type, count(\*) as count from spt\_music group by music\_medium order by count desc limit 0,20"**)  
 **public** List<MusicData> medium();  
  
 @Select(**"select music\_kind as type, sum(music\_rating+0)/count(\*) as count from spt\_music group by music\_kind order by music\_kind"**)  
 **public** List<MusicData> kind1();  
  
 @Select(**"select music\_kind as type, sum(music\_votes) as count from spt\_music group by music\_kind order by music\_kind"**)  
 **public** List<MusicData> kind2();  
}

VUE前端

音乐查询界面：

可根据音乐名、歌手、时间、标签、类型寻找出想要的音乐

<**template**>  
 <**div**>  
  
 <**a-page-header style="border-bottom: 1px solid rgb(235, 237, 240)"  
 title="音乐展示"**>  
 </**a-page-header**>  
  
 <**a-form layout="inline"**>  
  
 <**a-form-item  
 lable="乐名"  
 :label-col="formItemLayout**.**labelCol"  
 :wrapper-col="formItemLayout**.**wrapperCol"** >  
 <**a-input placeholder="请输入音乐名或关键字" v-model="name"**></**a-input**>  
 </**a-form-item**>  
  
 <**a-form-item  
 lable="歌手"  
 :label-col="formItemLayout**.**labelCol"  
 :wrapper-col="formItemLayout**.**wrapperCol"** >  
 <**a-input placeholder="请输入歌手名或关键字" v-model="singer"**></**a-input**>  
 </**a-form-item**>  
  
 <**a-form-item  
 lable="时间"  
 :label-col="formItemLayout**.**labelCol"  
 :wrapper-col="formItemLayout**.**wrapperCol"** >  
 <**a-input placeholder="请输入年份" v-model="time"**></**a-input**>  
 </**a-form-item**>  
  
 <**a-form-item  
 lable="标签"  
 :label-col="formItemLayout**.**labelCol"  
 :wrapper-col="formItemLayout**.**wrapperCol"** >  
 <**a-input placeholder="请输入标签" v-model="tag"**></**a-input**>  
 </**a-form-item**>  
  
 <**a-form-item  
 lable="种类"  
 :label-col="formItemLayout**.**labelCol"  
 :wrapper-col="formItemLayout**.**wrapperCol"** >  
 <**a-select default-value="全部" style="width: 120px" @change="**kindChange**"**>  
 <**a-select-option value="-1"**>  
 全部查询  
 </**a-select-option**>  
 <**a-select-option value="0"**>  
 古典  
 </**a-select-option**>  
 <**a-select-option value="1"**>  
 摇滚  
 </**a-select-option**>  
 <**a-select-option value="2"**>  
 放克/灵歌/R**&amp;**B  
 </**a-select-option**>  
 <**a-select-option value="3"**>  
 流行  
 </**a-select-option**>  
 <**a-select-option value="4"**>  
 说唱  
 </**a-select-option**>  
 <**a-select-option value="5"**>  
 民谣  
 </**a-select-option**>  
 <**a-select-option value="6"**>  
 轻音乐  
 </**a-select-option**>  
 <**a-select-option value="7"**>  
 原声  
 </**a-select-option**>  
 <**a-select-option value="8"**>  
 电子  
 </**a-select-option**>  
 <**a-select-option value="9"**>  
 拉丁  
 </**a-select-option**>  
 <**a-select-option value="10"**>  
 爵士  
 </**a-select-option**>  
 <**a-select-option value="11"**>  
 世界音乐  
 </**a-select-option**>  
 <**a-select-option value="12"**>  
 布鲁斯  
 </**a-select-option**>  
 <**a-select-option value="13"**>  
 Electronic電子  
 </**a-select-option**>  
 <**a-select-option value="14"**>  
 Funk/Soul/R**&amp;**B  
 </**a-select-option**>  
 <**a-select-option value="15"**>  
 雷鬼  
 </**a-select-option**>  
 <**a-select-option value="16"**>  
 Soundtrack原聲  
 </**a-select-option**>  
 </**a-select**>  
 </**a-form-item**>  
  
 <**a-form-item :wrapper-col="buttonItemLayout**.**wrapperCol"**>  
 <**a-button type="primary" v-on:click="**handleSubmit**"**>  
 立即查询  
 </**a-button**>  
 </**a-form-item**>  
 </**a-form**>  
  
 <**a-table :columns="*columns*" :data-source="data" :loading="loading" v-on:change="**nextPage**" :pagination = "pagination"**>  
 <**a slot="name" slot-scope="**text**"**>{{text}}</**a**>  
 <**span slot="customTitle"**><**a-icon type="smile-o"**>Name</**a-icon**></**span**>  
 <**span slot="tags" slot-scope="**tags**"**>  
 <**a-tag**>  
 {{tags}}  
 </**a-tag**>  
 </**span**>  
 </**a-table**>  
 </**div**>  
</**template**>  
  
<**script**>  
 **import qs from "qs"**;  
 **import *axios* from "axios"  
  
 const *columns*** = [  
 {  
 **title**:**'音乐名'**,  
 **dataIndex**:**'name'**,  
 **key**:**'name'**,  
 },  
 {  
 **title**:**'歌手'**,  
 **dataIndex**:**'singer'**,  
 **key**:**'singer'**,  
 },  
 {  
 **title**:**'发布时间'**,  
 **dataIndex**:**'time'**,  
 **key**:**'time'**,  
 },  
 {  
 **title**:**'评分'**,  
 **dataIndex**:**'rate'**,  
 **key**:**'rate'**,  
 },  
 {  
 **title**:**'标签'**,  
 **dataIndex**:**'tag'**,  
 **key**:**'tag'**,  
 },  
 {  
 **title**:**'种类'**,  
 **dataIndex**:**'kind'**,  
 **key**:**'kind'**,  
 },  
 {  
 **title**:**'同类歌曲推荐'**,  
 **dataIndex**:**'recommend'**,  
 **key**:**'recommend'**,  
 },  
 ];  
 **export default** {  
 **name**: **"dataDisplay"**,  
  
 data(){  
 **return**{  
 **name**: **null**,  
 **singer**: **null**,  
 **time**: **null**,  
 **rate**: **null**,  
 **tag**: **null**,  
 **kind**: -1,  
 **recommend**: **null**,  
  
 **data**: [],  
 ***columns***,  
 **loading**: **false**,  
 **page**: 1,  
 **limit**: 10,  
 **pagination**:{  
 **total**: 0,  
 }  
 }  
 },  
 **computed**:{  
 formItemLayout() {  
 **const** { formLayout } = **this**;  
 **return** formLayout === **'horizontal'** ? {  
 **labelCol**: { **span**: 4 },  
 **wrapperCol**: { **span**: 14 },  
 }  
 : {};  
 },  
 buttonItemLayout() {  
 **const** { formLayout } = **this**;  
 **return** formLayout === **'horizontal'** ? {  
 **wrapperCol**: { **span**: 14, **offset**: 4 },  
 }  
 : {};  
 }  
 },  
 **methods**:{  
 kindChange(e){  
 **this**.**kind** = e;  
 },  
 handleSubmit(){  
 **this**.**loading** = **true**;  
  
 **const** data = **qs**.*stringify*({  
 **name**: **this**.**name**,  
 **singer**: **this**.**singer**,  
 **time**: **this**.**time**,  
 **tag**: **this**.**tag**,  
 **kind**: **this**.**kind**,  
 **page**: **this**.**page**,  
 **limit**: **this**.**limit**,  
 **rate**: **this**.**rate**,  
 **recommend**: **this**.**recommend** })  
  
 **let** array = [];  
 ***axios***.post(**"http://localhost:8080/music/search"**,data,{**headers**:{**'Content-Type'**:**'application/x-www-form-urlencoded'**}})  
 .then(response => {  
  
 **let** data = response.**data**.**data**;  
 **for** (**var** index **in** data){  
 **let** music= data[index]  
 array = array.concat({  
 **key**: music.**id**,  
 **name**: music.**name**,  
 **singer**: music.**singer**,  
 **time**: music.**time**,  
 **tag**: music.**tag**,  
 **kind**: music.**kind**,  
 **rate**: music.**rate**,  
 **recommend**: music.**recommend**,  
 })  
 }  
  
 **this**.**pagination**.**total** = response.**data**.**count  
 this**.**data** = array  
 **this**.**loading** = **false** }).catch(error =>{  
 ***console***.log(error)  
 **this**.**loading** = **false** })  
 },  
 }  
 }  
</**script**>  
  
<**style scoped**>  
  
</**style**>

数据分析图表组件：

人气歌手：返回给歌手投票的总人数作为人气指数，以折线图展出

<**script**>  
 **import *axios* from "axios"**;  
 **import qs from "qs"**;  
 **export default** {  
 **name**: **"EchartsSinger"**,  
 data(){  
 **return**{  
 **datalist**: [],  
 **singerlist**: []  
 }  
 },  
 **methods**:{  
 myCharts(){  
 **var** myChart = **this**.**$echarts**.init(***document***.getElementById(**'singer'**));  
  
 **var** option = {  
 **tooltip**: {  
 **trigger**: **'axis'**,  
 position: **function** (pt) {  
 **return** [pt[0], **'10%'**];  
 }  
 },  
 **title**: {  
 **left**: **'center'**,  
 **text**: **'歌手人气榜'**,  
 },  
 **toolbox**: {  
 **feature**: {  
 **dataZoom**: {  
 **yAxisIndex**: **'none'** },  
 **restore**: {},  
 **saveAsImage**: {}  
 }  
 },  
 **xAxis**: {  
 **type**: **'category'**,  
 **boundaryGap**: **false**,  
 **data**: **this**.**singerlist** },  
 **yAxis**: {  
 **type**: **'value'**,  
 **boundaryGap**: [0, **'100%'**]  
 },  
 **dataZoom**: [{  
 **type**: **'inside'**,  
 **start**: 0,  
 **end**: 10  
 }, {  
 **start**: 0,  
 **end**: 10,  
 **handleIcon**: **'M10.7,11.9v-1.3H9.3v1.3c-4.9,0.3-8.8,4.4-8.8,9.4c0,5,3.9,9.1,8.8,9.4v1.3h1.3v-1.3c4.9-0.3,8.8-4.4,8.8-9.4C19.5,16.3,15.6,12.2,10.7,11.9z M13.3,24.4H6.7V23h6.6V24.4z M13.3,19.6H6.7v-1.4h6.6V19.6z'**,  
 **handleSize**: **'80%'**,  
 **handleStyle**: {  
 **color**: **'#fff'**,  
 **shadowBlur**: 3,  
 **shadowColor**: **'rgba(0, 0, 0, 0.6)'**,  
 **shadowOffsetX**: 2,  
 **shadowOffsetY**: 2  
 }  
 }],  
 **series**: [  
 {  
 **name**: **'模拟数据'**,  
 **type**: **'line'**,  
 **smooth**: **true**,  
 **symbol**: **'none'**,  
 **sampling**: **'average'**,  
 **itemStyle**: {  
 **color**: **'rgb(255, 70, 131)'** },  
 **data**: **this**.**datalist** }  
 ]  
 };  
 myChart.setOption(option);  
  
 }  
  
  
 },  
 mounted() {  
 **let** datarr = [];  
 **let** singerarr = [];  
  
 ***axios***.post(**"http://localhost:8080/music/analysis"**, **qs**.*stringify*({  
 **type**: **'singer'** }, {**headers**:{**'Content-Type'**:**'application/x-www-form-urlencoded'**}}))  
 .then(response =>{  
 **const** data = response.**data**.data1;  
 **for** (**let** index **in** data){  
 **let** item = data[index];  
 datarr = datarr.concat(item.**count**);  
 singerarr = singerarr.concat(item.**type**);  
 }  
  
 **this**.**datalist** = datarr;  
 **this**.**singerlist** = singerarr;  
 **this**.myCharts();  
 }).catch(error=>{  
 ***console***.log(error)  
 })  
 }  
 }  
</**script**>

发行商市场占有率：

返回发行商市场占有比率，以饼状图展示

<**script**>  
 **import *axios* from "axios"**;  
 **import qs from "qs"**;  
 **export default** {  
 **name**: **"EchartsCompare"**,  
 data(){  
 **return**{  
 **datalist**:[],  
 **publisherlist**:[],  
 **dictlist**:[]  
 }  
 },  
 **methods**:{  
 myCharts(){  
 **var** myChart = **this**.**$echarts**.init(***document***.getElementById(**'publisher'**));  
  
 **var** option = {  
 **title**: {  
 **text**: **'发行商市场占有率'**,  
 **left**: **'center'** },  
 **tooltip**: {  
 **trigger**: **'item'**,  
 **formatter**: **'{a} <br/>{b} : {c} ({d}%)'** },  
 **legend**: {  
 **type**: **'scroll'**,  
 **orient**: **'vertical'**,  
 **right**: -50,  
 **top**: 20,  
 **bottom**: 20,  
 **data**: **this**.**publisherlist**,  
  
 **selected**: **this**.**dictlist** },  
 **series**: [  
 {  
 **name**: **'姓名'**,  
 **type**: **'pie'**,  
 **radius**: **'55%'**,  
 **center**: [**'40%'**, **'50%'**],  
 **data**: **this**.**datalist**,  
 **emphasis**: {  
 **itemStyle**: {  
 **shadowBlur**: 10,  
 **shadowOffsetX**: 0,  
 **shadowColor**: **'rgba(0, 0, 0, 0.5)'** }  
 }  
 }  
 ]  
 };  
  
 myChart.setOption(option);  
  
 }  
  
  
 },  
 mounted() {  
 **let** datarr = [];  
 **let** publisharr = [];  
 **let** dictarr = {};  
  
 ***axios***.post(**"http://localhost:8080/music/analysis"**, **qs**.*stringify*({  
 **type**: **'publisher'** }, {**headers**:{**'Content-Type'**:**'application/x-www-form-urlencoded'**}}))  
 .then(response =>{  
 **const** data = response.**data**.data1;  
 **let** i = 0;  
 **for** (**let** index **in** data){  
 **let** item = data[index];  
 datarr = datarr.concat({**name**:item.**type**,**value**:item.**count**})  
 publisharr = publisharr.concat(item.**type**)  
 dictarr[item.**type**] = i<10;  
 i++;  
 }  
 **this**.**datalist** = datarr;  
 **this**.**publisherlist** = publisharr;  
 **this**.**dictlist** = dictarr;  
 **this**.myCharts();  
 }).catch(error=>{  
 ***console***.log(error)  
 })  
 }  
 }  
</**script**>

媒介人气：查看不同梅姐的歌曲的评分

<**script**>  
 **import *axios* from "axios"**;  
 **import qs from "qs"**;  
 **export default** {  
 **name**: **"EchartsMedium"**,  
 data(){  
 **return**{  
 **datalist**:[],  
 **mediumlist**:[]  
 }  
 },  
 **methods**:{  
 myCharts(){  
 **var** myChart = **this**.**$echarts**.init(***document***.getElementById(**'medium'**));  
  
 **var** option = {  
 **tooltip**: {  
 **trigger**: **'item'**,  
 **formatter**: **'{a} <br/>{b}: {c} ({d}%)'** },  
 **legend**: {  
 **orient**: **'vertical'**,  
 **left**: 10,  
 **data**: **this**.***mediumaquamarine*** },  
 **title**: {  
 **left**: **'center'**,  
 **text**: **'最受欢迎的发行媒介'**,  
 },  
 **series**: [  
 {  
 **name**: **'访问来源'**,  
 **type**: **'pie'**,  
 **radius**: [**'50%'**, **'70%'**],  
 **avoidLabelOverlap**: **false**,  
 **label**: {  
 **show**: **false**,  
 **position**: **'center'** },  
 **emphasis**: {  
 **label**: {  
 **show**: **true**,  
 **fontSize**: **'30'**,  
 **fontWeight**: **'bold'** }  
 },  
 **labelLine**: {  
 **show**: **false** },  
 **data**: **this**.**datalist** }  
 ]  
 };  
  
  
 myChart.setOption(option);  
  
 }  
  
  
 },  
 mounted() {  
 **let** datarr = [];  
 **let** mediumarr = [];  
 ***axios***.post(**"http://localhost:8080/music/analysis"**, **qs**.*stringify*({  
 **type**: **'medium'** }, {**headers**:{**'Content-Type'**:**'application/x-www-form-urlencoded'**}}))  
 .then(response =>{  
 **const** data = response.**data**.data1;  
 **for** (**let** index **in** data){  
 **let** item = data[index];  
 datarr = datarr.concat({**value**:item.**count**,**name**:item.**type**})  
 mediumarr = mediumarr.concat(item.**type**)  
 }  
 **this**.**datalist** = datarr;  
 **this**.**mediumlist** = mediumarr;  
 **this**.myCharts();  
 }).catch(error=>{  
 ***console***.log(error)  
 })  
 }  
 }  
</**script**>

不同类型音乐5分上下数量

<**script**>  
 **import *axios* from "axios"**;  
 **import qs from "qs"**;  
 **export default** {  
 **name**: **"EchartsCompare"**,  
 data(){  
 **return**{  
 **datalist1**: [],  
 **datalist2**: [],  
 **kindlist**: [],  
 }  
 },  
 **methods**:{  
 myCharts(){  
 **var** myChart = **this**.**$echarts**.init(***document***.getElementById(**'compare'**));  
  
 **var** option = {  
 **tooltip**: {  
 **trigger**: **'axis'**,  
 **axisPointer**: { *// 坐标轴指示器，坐标轴触发有效* **type**: **'shadow'** *// 默认为直线，可选为：'line' | 'shadow'* }  
 },  
 **title**: {  
 **right**: 0,  
 **text**: **'5分上下比较'** },  
 **legend**: {  
 **data**: [ **'5分及以下'**, **'5分以上'**]  
 },  
 **grid**: {  
 **left**: **'3%'**,  
 **right**: **'4%'**,  
 **bottom**: **'3%'**,  
 **containLabel**: **true** },  
 **xAxis**: [  
 {  
 **type**: **'value'** }  
 ],  
 **yAxis**: [  
 {  
 **type**: **'category'**,  
 **axisTick**: {  
 **show**: **false** },  
 **data**:**this**.**kindlist** }  
 ],  
 **series**: [  
 {  
 **name**: **'5分以上'**,  
 **type**: **'bar'**,  
 **stack**: **'总量'**,  
 **label**: {  
 **show**: **true**,  
 **postion**: **"right"** },  
 **data**: **this**.**datalist1** },  
 {  
 **name**: **'5分以下'**,  
 **type**: **'bar'**,  
 **stack**: **'总量'**,  
 **label**: {  
 **show**: **true**,  
  
 },  
 **data**:**this**.**datalist2** }  
 ]  
 };  
  
 myChart.setOption(option);  
  
 }  
  
  
 },  
 mounted() {  
 **let** data1arr = [];  
 **let** data2arr = [];  
 **let** kindarr = [];  
  
 ***axios***.post(**"http://localhost:8080/music/analysis"**, **qs**.*stringify*({  
 **type**: **'compare'** }, {**headers**:{**'Content-Type'**:**'application/x-www-form-urlencoded'**}}))  
 .then(response =>{  
 **const** data1 = response.**data**.data1;  
 **for** (**let** index **in** data1){  
 **let** item = data1[index];  
 data1arr = data1arr.concat(item.**count**);  
 kindarr = kindarr.concat(item.**type**);  
 }  
 **const** data2 = response.**data**.data2;  
 **for** (**let** index **in** data2){  
 **let** item = data2[index];  
 data2arr = data2arr.concat(-(item.**count**))  
 }  
  
 **this**.**datalist1** = data1arr;  
 **this**.**datalist2** = data2arr;  
 **this**.**kindlist** = kindarr;  
 **this**.myCharts();  
 }).catch(error=>{  
 ***console***.log(error)  
 })  
 }  
 }  
</**script**>

不同类型音乐评分人气

<**script**>  
 **import *axios* from "axios"**;  
 **import qs from "qs"**;  
 **export default** {  
 **name**: **"EchartsKind"**,  
 data(){  
 **return**{  
 **datalist1**: [],  
 **datalist2**: [],  
 **kindlist**: []  
 }  
 },  
 **methods**:{  
 myCharts(){  
 **var** myChart = **this**.**$echarts**.init(***document***.getElementById(**'kind'**));  
  
 **var** option = {  
 **title**: {  
 **text**: **'音乐类型人气&评分'** },  
 **legend**: {  
 **data**: [**'总评分'**, **'总投票'**]  
 },  
 **toolbox**: {  
 **y**: **'bottom'**,  
 **feature**: {  
 **magicType**: {  
 **type**: [**'stack'**, **'tiled'**]  
 },  
 **dataView**: {},  
 **saveAsImage**: {  
 **pixelRatio**: 2  
 }  
 }  
 },  
 **tooltip**: {},  
 **xAxis**: {  
 **data**: **this**.**kindlist**,  
 **splitLine**: {  
 **show**: **false** }  
 },  
 **yAxis**: {  
 },  
 **series**: [{  
 **name**: **'人气'**,  
 **type**: **'bar'**,  
 **data**: **this**.**datalist1**,  
 animationDelay: **function** (idx) {  
 **return** idx \* 10;  
 }  
 }, {  
 **name**: **'评分'**,  
 **type**: **'bar'**,  
 **data**: **this**.**datalist2**,  
 animationDelay: **function** (idx) {  
 **return** idx \* 10 + 100;  
 }  
 }],  
 **animationEasing**: **'elasticOut'**,  
 animationDelayUpdate: **function** (idx) {  
 **return** idx \* 5;  
 }  
 };  
 myChart.setOption(option);  
  
 }  
  
  
 },  
 mounted() {  
 **let** data1arr = [];  
 **let** data2arr = [];  
 **let** kindarr = [];  
  
 ***axios***.post(**"http://localhost:8080/music/analysis"**, **qs**.*stringify*({  
 **type**: **'kind'** }, {**headers**:{**'Content-Type'**:**'application/x-www-form-urlencoded'**}}))  
 .then(response =>{  
 **const** data1 = response.**data**.data1;  
 **for** (**let** index **in** data1){  
 **let** item = data1[index];  
 data1arr = data1arr.concat(item.**count**);  
 kindarr = kindarr.concat(item.**type**);  
 }  
 **const** data2 = response.**data**.data2;  
 **for** (**let** index **in** data2){  
 **let** item = data2[index];  
 data2arr = data2arr.concat(item.**count**)  
 }  
  
 **this**.**datalist1** = data1arr;  
 **this**.**datalist2** = data2arr;  
 **this**.**kindlist** = kindarr;  
 **this**.myCharts();  
 }).catch(error=>{  
 ***console***.log(error)  
 })  
 }  
 }  
</**script**>

数据可视化

<**template**>  
 <**div**>  
 <**a-row**>  
 <**EchartsSinger** />  
 </**a-row**>  
 <**a-row**>  
 <**Echartspublisher** />  
 </**a-row**>  
 <**a-row**>  
 <**EchartsMedium** />  
 </**a-row**>  
 <**a-row**>  
 <**a-col :span="**12**"**>  
 <**EchartsKind** />  
 </**a-col**>  
 <**a-col :span="**12**"**>  
 <**EchartsCompare** />  
 </**a-col**>  
 </**a-row**>  
 </**div**>  
</**template**>  
  
<**script**>  
 **import** EchartsSinger **from "../../components/music/EchartsSinger"**;  
 **import** EchartsCompare **from "../../components/music/EchartsCompare"**;  
 **import** Echartspublisher **from "../../components/music/Echartspublisher"**;  
 **import** EchartsMedium **from "../../components/music/EchartsMedium"**;  
 **import** EchartsKind **from "../../components/music/EchartsKind"**;  
 **export default** {  
 **name**:**"dataDisplay"**,  
 **components**:{  
 EchartsSinger,  
 EchartsCompare,  
 Echartspublisher,  
 EchartsMedium,  
 EchartsKind  
 }  
 }  
</**script**>  
  
<**style scoped**>  
  
</**style**>

**Spider（爬虫）**：

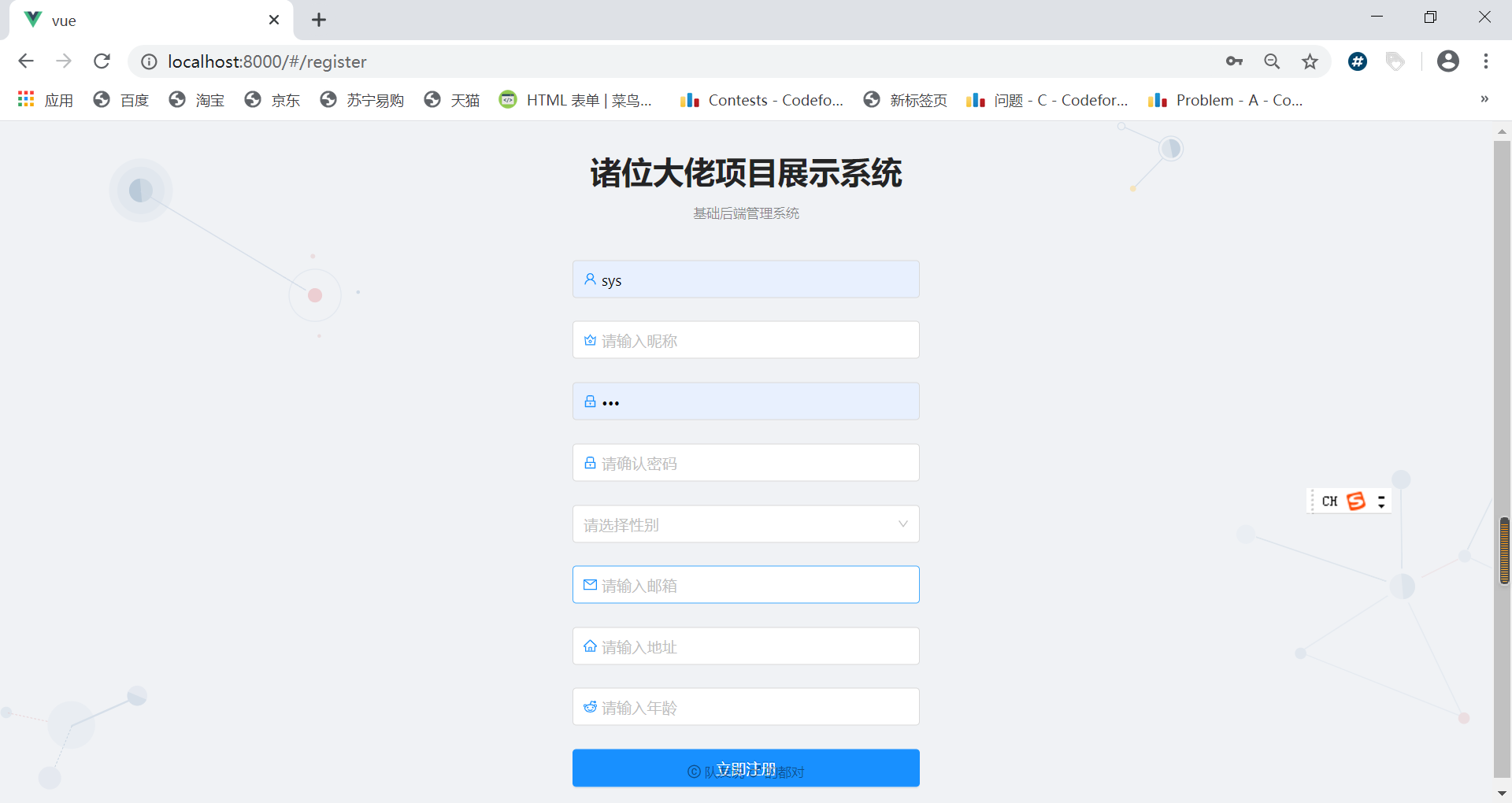
包括爬取网站，爬取标签，爬取元素

class DoubanscrapySpider(CrawlSpider):  
  
 name = **'doubanscrapy'** allowed\_domains = [**'music.douban.com'**]  
 start\_urls = [**'https://music.douban.com/tag/瑞典'**]  
  
 rules = (  
  
 Rule(LinkExtractor(allow=**r"/subject/\d+/$"**), callback=**'parse\_music'**, follow=True),  
 Rule(LinkExtractor(allow=**r"/comments/$"**),callback=**'parse\_review'**, follow=True)  
 )  
  
 def parse\_music(self, response):  
 item = MusicItem()  
 pic = PicItem()  
  
 pic[**'src'**] = [response.xpath(**'//\*[@class="nbg"]/img/@src'**).extract\_first()]  
 try:  
 item[**'music\_name'**] = response.xpath(**'//\*[@id="wrapper"]/h1/span/text()'**).extract()[0]  
 content = **""**.join(response.xpath(**'//\*[@id="info"]'**).extract())  
 info = response.xpath(**'//\*[@id="info"]/span'**).extract()  
 item[**'music\_alias'**] = **""** for i in range(0, len(info)):  
 if **"又名"** in info[i]:  
 if i == 0:  
 item[**'music\_alias'**] = response.xpath(**'//\*[@id="info"]/text()'**).extract()[1]\  
 .replace(**"**\xa0**"**, **""**).replace(**"**\n**"**, **""**).rstrip()  
 if i == 1:  
 item[**'music\_alias'**] = response.xpath(**'//\*[@id="info"]/text()'**).extract()[2]\  
 .replace(**"**\xa0**"**, **""**).replace(**"**\n**"**, **""**).rstrip()  
 if i == 2:  
 item[**'music\_alias'**] = response.xpath(**'//\*[@id="info"]/text()'**).extract()[3]\  
 .replace(**"**\xa0**"**, **""**).replace(**"**\n**"**, **""**).rstrip()  
  
 if **"表演者"** in info[i]:  
 if i == 0:  
 item[**'music\_singer'**] = **"|"**.join(  
 response.xpath(**'//\*[@id="info"]/span[1]/span/a/text()'**).extract()  
 )  
 elif i == 1:  
 item[**'music\_singer'**] = **"|"**.join(  
 response.xpath(**'//\*[@id="info"]/span[2]/span/a/text()'**).extract()  
 )  
 elif i == 2:  
 item[**'music\_singer'**] = **"|"**.join(  
 response.xpath(**'//\*[@id="info"]/span[3]/span/a/text()'**).extract()  
 )  
 else:  
 item[**'music\_singer'**] = **""** if **"发行时间"** in info[i]:  
 nbsp = re.findall(**r"<span class=\"pl\">发行时间:</span>(.\*?)<br>"**,content,re.S)  
 item[**'music\_time'**] = **""**.join(nbsp).replace(**"**\xa0**"**, **""**).replace(**"**\n**"**, **""**).replace(**" "**, **""**)  
 if **"流派"** in info[i]:  
 nbsp = re.findall(**r"<span class=\"pl\">流派:</span>(.\*?)<br>"**, content, re.S)  
 item[**'music\_kind'**] = **""**.join(nbsp).replace(**"**\xa0**"**, **""**).replace(**"**\n**"**, **""**).replace(**" "**, **""**)  
 if **"专辑类型"** in info[i]:  
 nbsp = re.findall(**r"<span class=\"pl\">专辑类型:</span>(.\*?)<br>"**, content, re.S)  
 item[**'music\_type'**] = **""**.join(nbsp).replace(**"**\xa0**"**, **""**).replace(**"**\n**"**, **""**).replace(**" "**, **""**)  
 if **"介质"** in info[i]:  
 nbsp = re.findall(**r"<span class=\"pl\">介质:</span>(.\*?)<br>"**, content, re.S)  
 item[**'music\_medium'**] = **""**.join(nbsp).replace(**"**\xa0**"**, **""**).replace(**"**\n**"**, **""**).replace(**" "**, **""**)  
 if **"出版者"** in info[i]:  
 nbsp = re.findall(**r"<span class=\"pl\">出版者:</span>(.\*?)<br>"**, content, re.S)  
 item[**'music\_publisher'**] = **""**.join(nbsp).replace(**"**\xa0**"**, **""**).replace(**"**\n**"**, **""**).replace(**" "**, **""**)  
  
 try:  
 item[**'music\_rating'**] = **""**.join(response.xpath(  
 **'//\*[@class="rating\_self clearfix"]/strong/text()'**).extract())  
 item[**'music\_votes'**] = **""**.join(response.xpath(  
 **'//\*[@class="rating\_self clearfix"]/div/div[@class="rating\_sum"]/a/span/text()'**).extract())  
 except Exception as error:  
 item[**'music\_rating'**] = **'0'** item[**'music\_votes'**] = **'0'** print(error)  
  
 item[**'music\_recommend'**] = **"|"**.join(response.xpath(**'//\*[@id="db-rec-section"]/div/dl/dd/a/text()'**).extract())  
 item[**'music\_tags'**] = **"|"**.join(response.xpath(**'//\*[@id="db-tags-section"]/div/a/text()'**).extract())  
 item[**'music\_url'**] = response.url  
 item[**'music\_song'**] = **""**.join(response.xpath(**'//\*[@class="track-list"]/div/div/text()'**).extract())\  
 .replace(**"**\n**"**, **""**).replace(**"**\t**"**, **""**).replace(**" "**, **""**)  
 intr = response.xpath(**'//\*[@class="all hidden"]/text()'**).extract()  
 item[**'music\_introduction'**] = **""**.join(intr).replace(**"**\u3000**"**, **""**).replace(**"**\\**'"**, **""**)  
 item[**'pic\_hash'**] = **'full/%s.jpg'** % (hashlib.sha1(to\_bytes(pic[**'src'**][0])).hexdigest())  
  
 yield pic  
 yield item  
 except Exception as err:  
 print(**"spyider"** + err)

pipelines：

将spider中爬去的数据存入数据库中

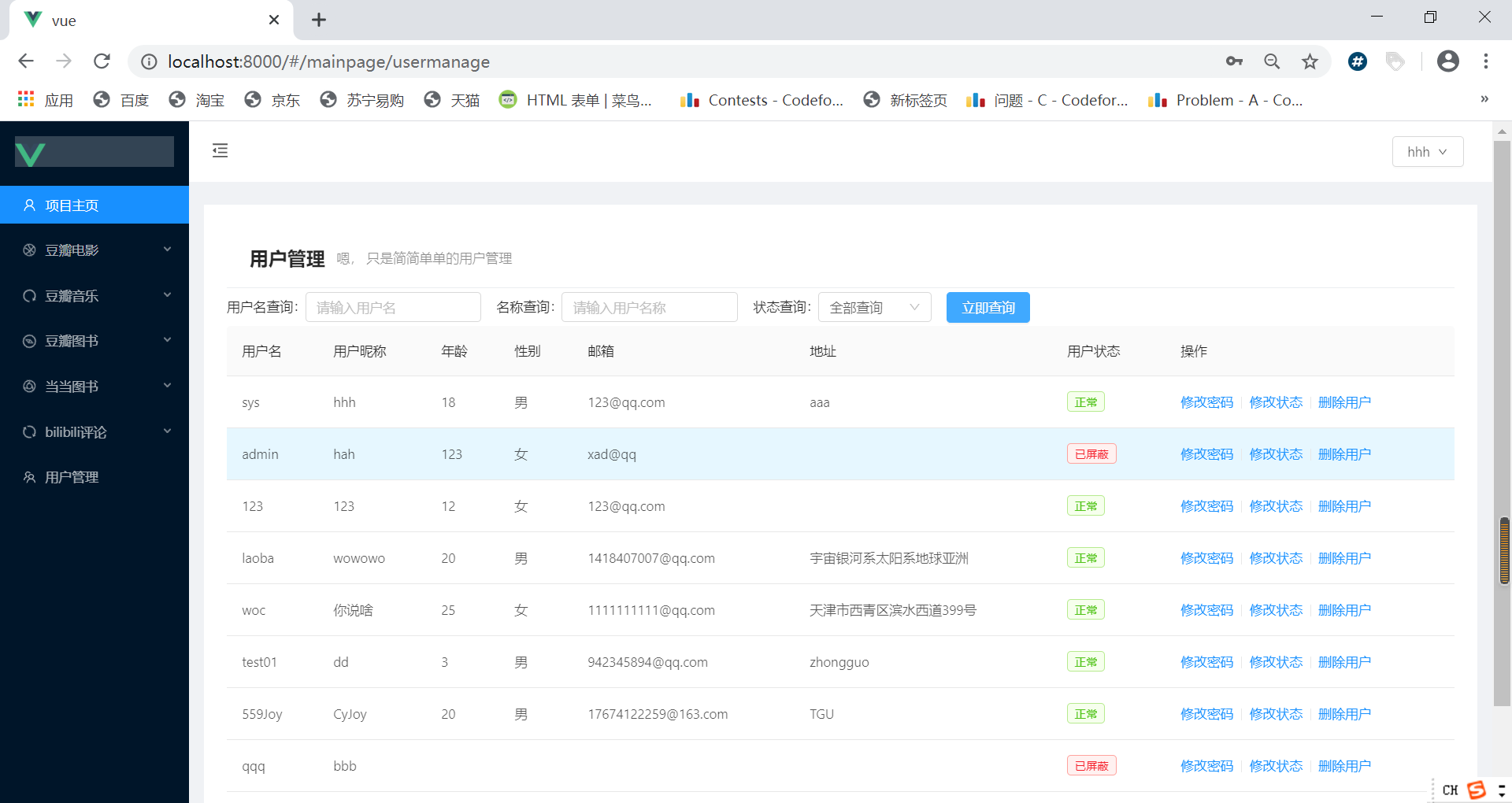
def process\_music(self, item):  
 try:  
 self.connect.ping(reconnect=True)  
 self.cursor.execute(**'''  
 select \* from spt\_music where music\_name=%s   
 '''**,(item[**'music\_name'**],))  
 music = self.cursor.fetchone()  
 if music == None:  
 self.cursor.execute(**'''  
 insert into spt\_music(music\_name, music\_alias, music\_singer, music\_time,   
 music\_rating,music\_votes, music\_tags, music\_url, music\_medium, music\_kind,  
 music\_type, music\_publisher, music\_recommend, music\_song, music\_introduction, pic\_hash)  
 value (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)  
 '''**,  
 (item[**'music\_name'**],  
 item[**'music\_alias'**],  
 item[**'music\_singer'**],  
 item[**'music\_time'**],  
 item[**'music\_rating'**],  
 item[**'music\_votes'**],  
 item[**'music\_tags'**],  
 item[**'music\_url'**],  
 item[**'music\_medium'**],  
 item[**'music\_kind'**],  
 item[**'music\_type'**],  
 item[**'music\_publisher'**],  
 item[**'music\_recommend'**],  
 item[**'music\_song'**],  
 item[**'music\_introduction'**],  
 item[**'pic\_hash'**]))  
 else:  
 self.cursor.execute(**'''  
 update spt\_music  
 set music\_alias = %s,   
 music\_singer = %s,   
 music\_time = %s,   
 music\_rating = %s,  
 music\_votes = %s,   
 music\_tags = %s,   
 music\_url = %s,  
 music\_medium = %s,  
 music\_kind = %s,  
 music\_type = %s,  
 music\_publisher = %s,  
 music\_recommend = %s,  
 music\_song = %s,  
 music\_introduction = %s,  
 pic\_hash = %s  
 where music\_name=%s  
 '''**,  
 (item[**'music\_alias'**],  
 item[**'music\_singer'**],  
 item[**'music\_time'**],  
 item[**'music\_rating'**],  
 item[**'music\_votes'**],  
 item[**'music\_tags'**],  
 item[**'music\_url'**],  
 item[**'music\_medium'**],  
 item[**'music\_kind'**],  
 item[**'music\_type'**],  
 item[**'music\_publisher'**],  
 item[**'music\_recommend'**],  
 item[**'music\_song'**],  
 item[**'music\_introduction'**],  
 item[**'pic\_hash'**],  
 item[**'music\_name'**]))  
 self.connect.commit()  
 except Exception as err:  
 print(**"pipelines"** + err)



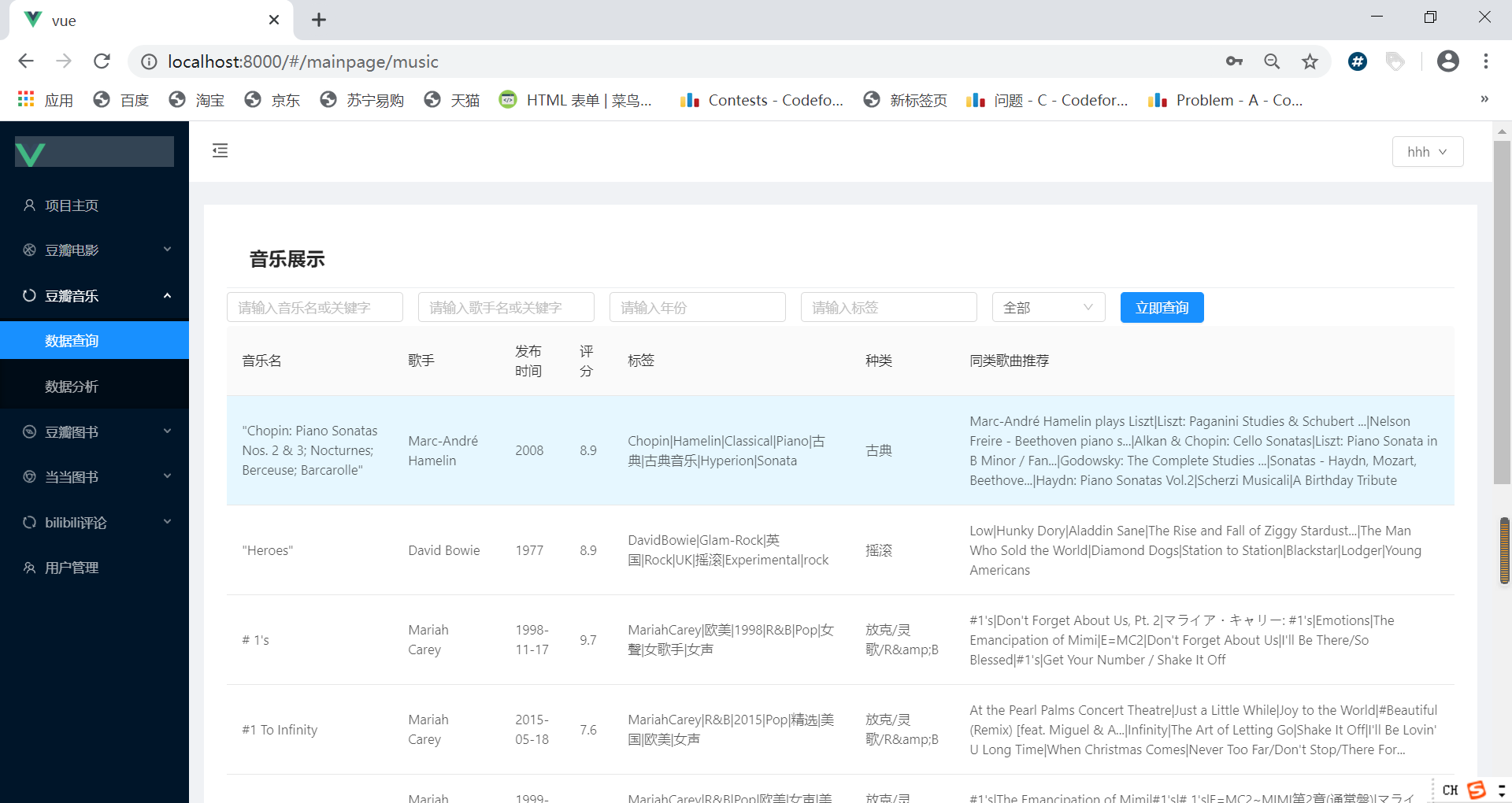
用户注册：系统自动判断条件，若符合规则则将用户添加如数据库中

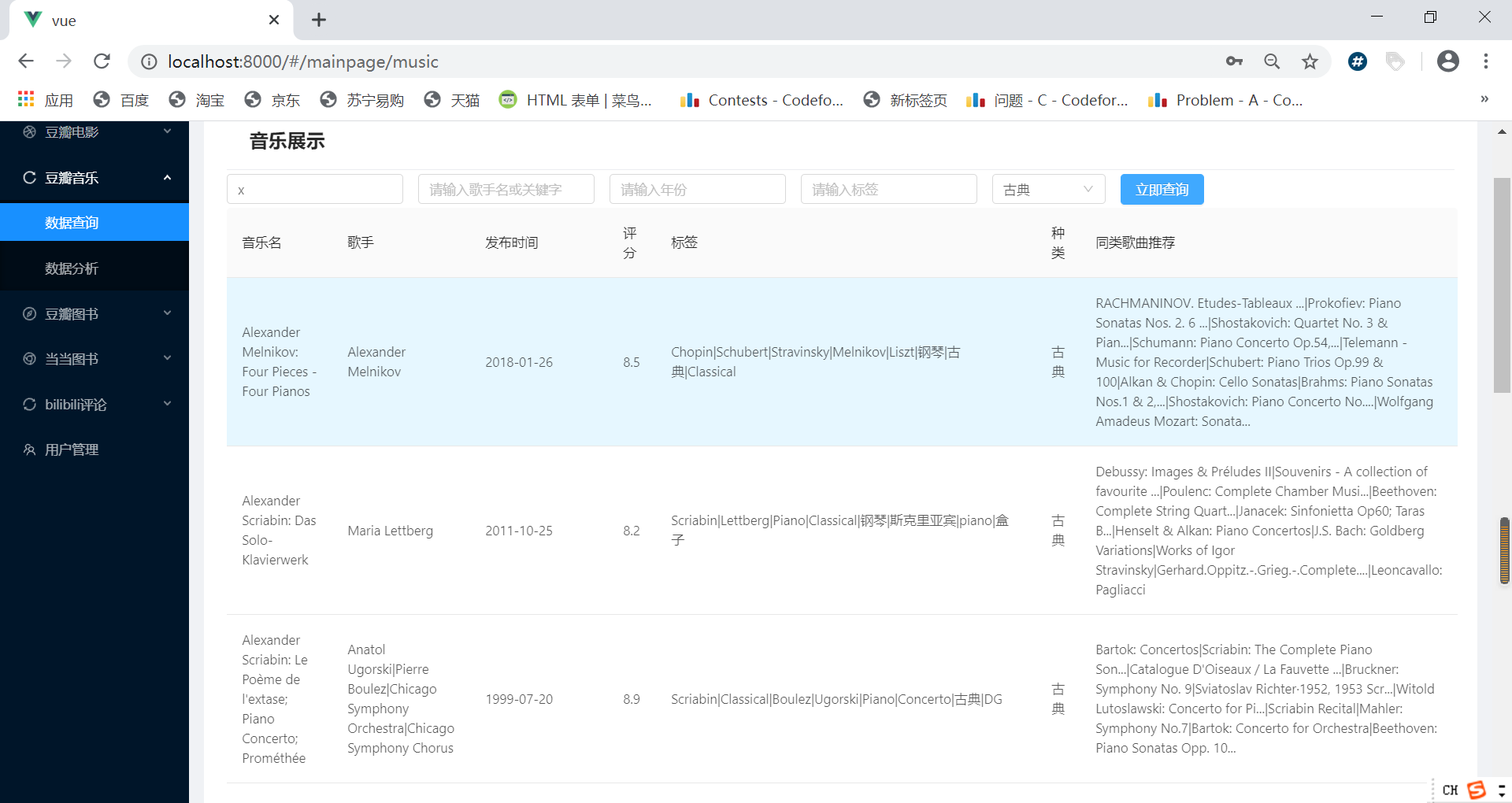


用户登录：传给给后端用户名，判断用户名密码存在时则登录成功，否则失败

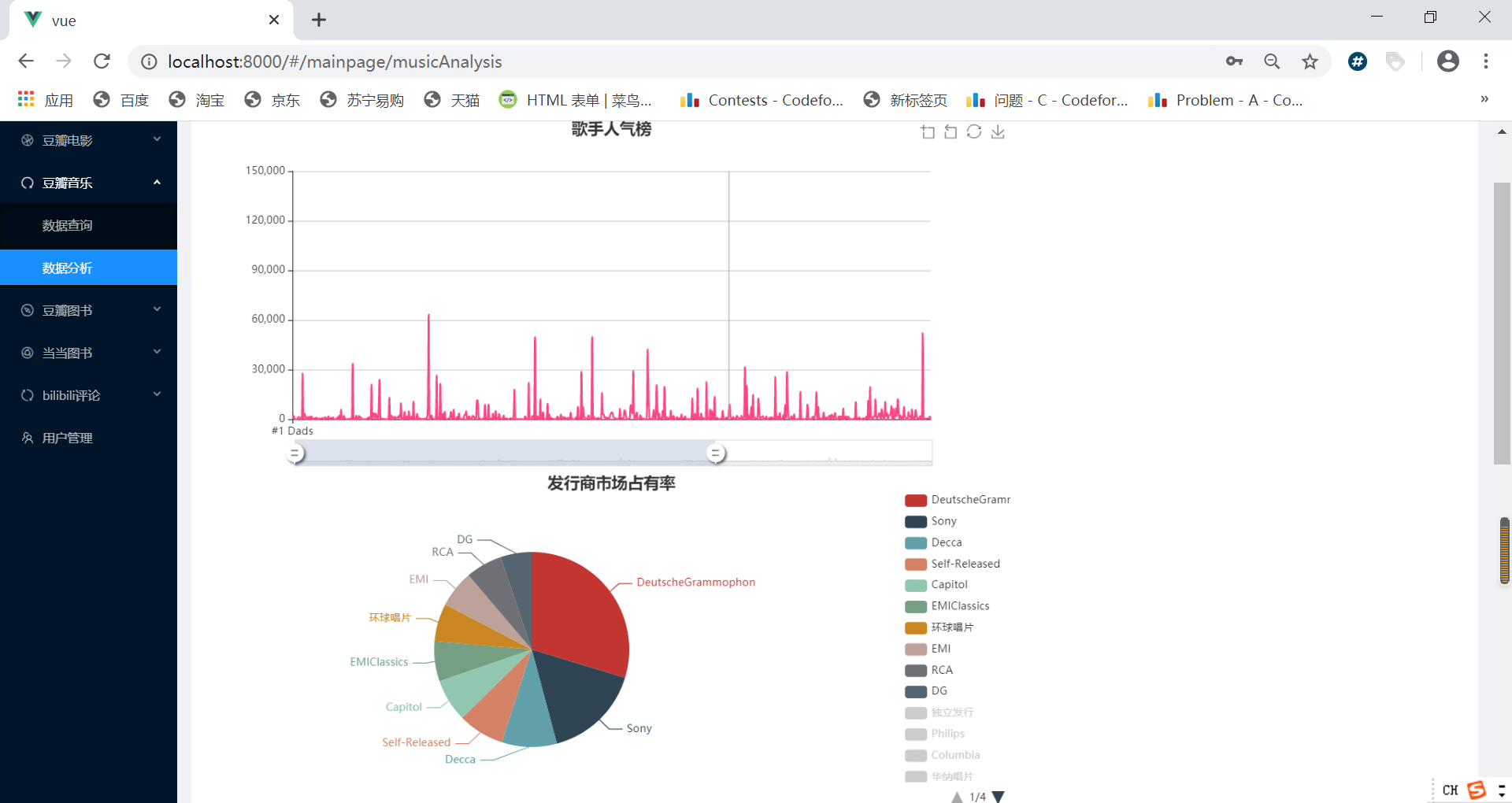


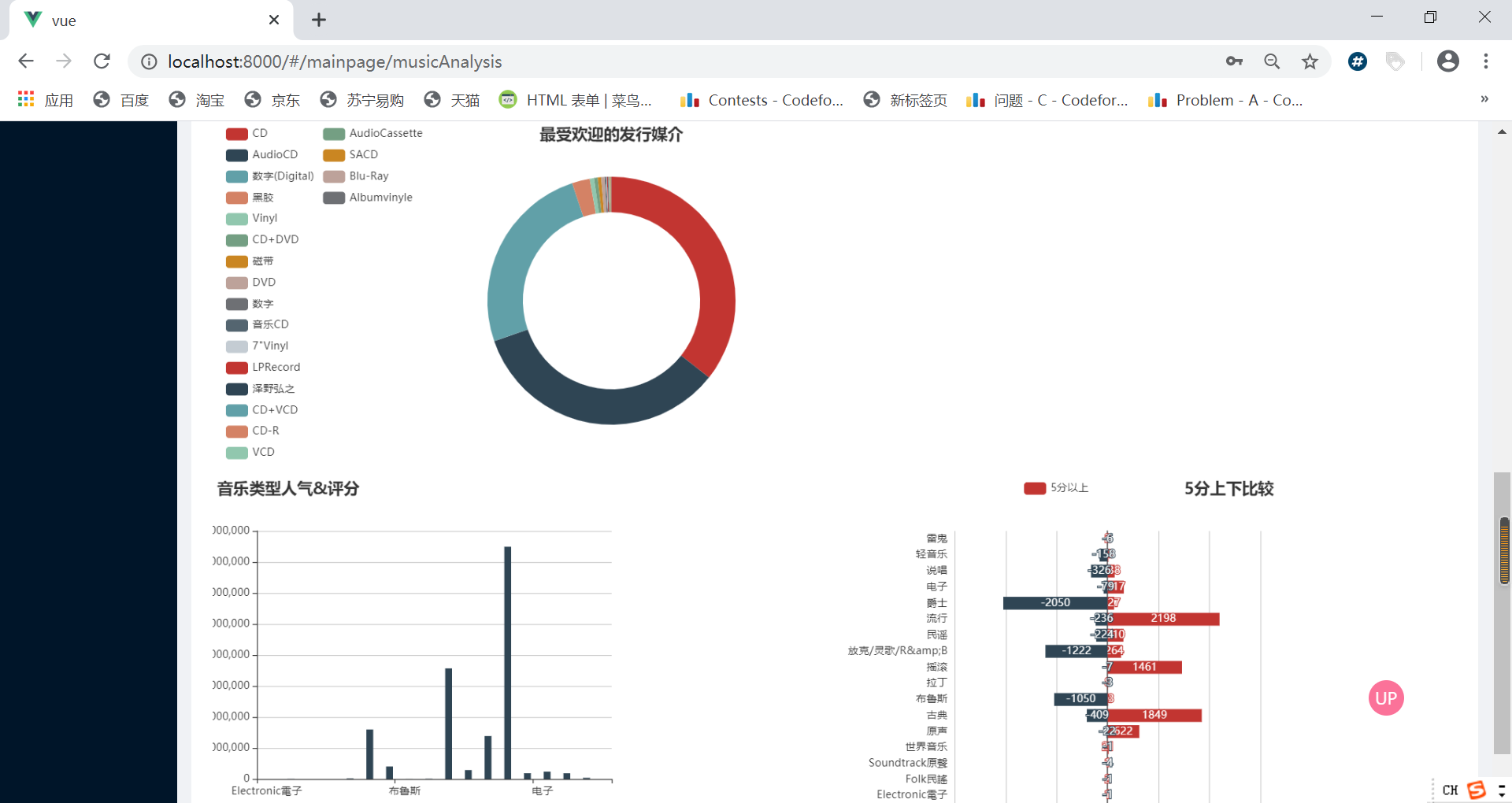
用户管理：可以在界面中查询所有用户，还可以修改用户状态、密码、删除用户





豆瓣音乐查询：可以根据该关键字查询所需的音乐





豆瓣音乐分析：以图表方式展现出相关音乐数据分析

**5 系统测试**

测试方法：通过Postman测试接口返回值是否正常，前端登录页面测试，修改室主要依据idea、webstorm自带报错机制。