**《Music**系统Node **》说明文档**

1. **学号：**
2. **姓名：**

**Music音乐播放器项目 主要通过vue+nodejs+express实现，可以提交在程序中上传自己所喜爱的音频文件，作为一个mp3仓库，同时若别的用户设置允许公开账户，那么你将获得别人的歌单。是一个有趣又富有实战意义的练手小项目。**

#### 技术框架 Vue+Express+Mysql

Vue前端部分:使用Vue + VantUi

用户之间的交互功能在前端列出，共11个请求，数据表有3个

├─build

├─config

├─dist

│ └─static

│ ├─css

│ ├─fonts

│ └─js

├─node\_modules

├─src

│ ├─assets

│ │ └─css

│ ├─components

│ ├─mock

│ ├─pages

│ ├─router

│ └─utils

└─static

### 界面功能：

登陆界面





关键代码

*<*template*>  
 <*div class="login"*>  
 <*van-nav-bar  
 title="welcome"  
 */>  
 <*van-notice-bar  
 left-icon="volume-o"  
 text="欢迎来到音乐的世界，快来添加属于你自己的音乐吧！"  
 */>  
 <*van-notice-bar scrollable text="用户xxx加入了我们" */>  
 <*van-notice-bar scrollable text="用户xxx加入了我们" */>  
 <*van-tabs type="card" v-model="active"*>  
 <*van-tab title="登录"*>  
 <*van-form @submit="onSubmit"*>  
 <*van-field  
 v-model="tel"  
 name="用户名"  
 label="用户名"  
 placeholder="用户名"  
 :rules="*[{* required: true,  
 message: '请填写用户名' ,  
 *}]*"  
 */>  
 <*van-field  
 v-model="password"  
 type="password"  
 name="密码"  
 label="密码"  
 placeholder="密码"  
 :rules="*[{* required: true, message: '请填写密码' *}]*"  
 */>  
 <*div style="margin: 16px;"*>  
 <*van-button round block type="info" native-type="submit"*>*提交*</*van-button*>  
 </*div*>  
 </*van-form*>  
 </*van-tab*>  
 <*van-tab title="注册"*>  
 <*van-form @submit="onSubmit"*>  
 <*van-field  
 v-model="username"  
 name="昵称"  
 label="用户名"  
 placeholder="用户名"  
 :rules="*[{* required: true,  
 message: '请填写用户名'  
 *}]*"  
 */>  
 <*van-field  
 v-model="tel"  
 type="tel"  
 name="手机号"  
 label="手机号"  
 placeholder="手机号"  
 :rules="*[{* required: true, message: '请填写你的手机号' *}]*"  
 */>  
 <*van-field  
 v-model="password"  
 type="password"  
 name="密码"  
 label="密码"  
 placeholder="密码"  
 :rules="*[{* required: true, message: '请填写密码' *}]*"  
 */>  
 <*van-field  
 v-model="confirmPass"  
 type="password"  
 name="确认密码"  
 label="确认密码"  
 placeholder="确认密码"  
 :rules="*[{* required: true, message: '请确认你的密码' ,validator:Validator*}]*"  
 */>  
 <*div style="margin: 16px;"*>  
 <*van-button round block type="info" native-type="submit"*>*提交*</*van-button*>  
 </*div*>  
 </*van-form*>  
 </*van-tab*>  
 </*van-tabs*>  
 </*div*>  
  
</*template*>  
  
<*script*>*import *{*register,login*}* from "../utils/query";  
  
export default *{* data*() {* return *{* username: '',  
 password: '',  
 confirmPass:'',  
 tel:'',  
 active:0  
 *}*;  
 *}*,  
 methods: *{* Validator*(){* return this.password===this.confirmPass  
 *}*,  
 click*(){* this.$router.replace*(*"/main"*)  
 }*,  
 onSubmit*(*values*) {* if*(*this.active===0*){* login*(*this.tel,this.password*)*.then*(*res=>*{* ***console***.log*(*res.code*)*;  
 if*(*res.code===0*){* this.$router.replace*(*"/main"*)  
 }  
 })  
 }*else*{* register*(*this.username,this.password,this.tel*)*.then*(*res=>*{* if*(*res.code===0*){* this.$router.replace*(*"/main"*)  
 }  
 })  
 }  
  
 }*,  
 *}*,  
 created*() {  
  
  
 }  
}*;  
*</*script*>  
  
<*style scoped*>  
</*style*>*

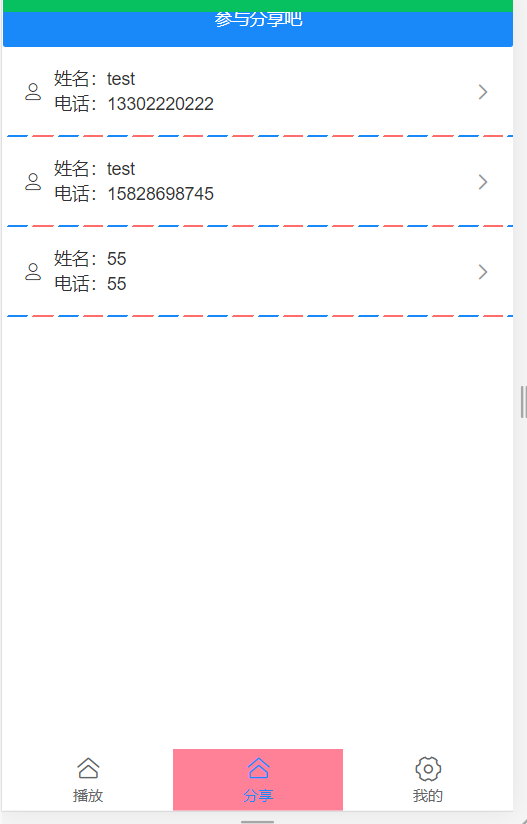
### 音乐播放主界面



关键代码

*<*template*>  
 <*div*>  
 <*van-sticky*>  
 <*van-button type="info" style="width: 100%;"*>*音乐之声*</*van-button*>  
 </*van-sticky*>  
  
 <*van-swipe class="my-swipe" :autoplay="3000" indicator-color="white"*>  
 <*van-swipe-item *>  
 <*img :src="songList*[*precentIndex*]*.imginfo" alt="" style="height: 150px"*>  
 </*van-swipe-item*>  
 </*van-swipe*>  
  
 <*van-row gutter="30" type="flex" justify="center" style="margin: 10px"*>  
 <*van-col span="8" style="text-align: right"*>  
 <*van-icon name="arrow-left" size="40" @click="lastSong"*/>  
 </*van-col*>  
 <*van-col span="8" style="text-align: center" class="switchBox"*>  
 <*van-switch v-model="checked" size="31px" @click="toPlaySong"*/>  
 <*van-icon :name="iconName" */>  
 </*van-col*>  
 <*van-col span="8"*>  
 <*van-icon name="arrow" size="40" @click="nextSong"*/>  
 </*van-col*>  
 </*van-row*>  
  
 <*van-slider v-model="defaultVal" @change="onChange" style="margin: 20px 0"*/>  
  
 <*van-collapse v-model="activeNames" style="padding-bottom: 50px"*>  
 <*van-collapse-item title="歌曲列表" name="1"*>  
 <*van-card  
 :desc="'原唱：'+item.singer"  
 :title="item.song"  
 :thumb="songList*[*index*]*.imginfo"  
 v-for="*(*item,index*)* in songList"  
 :key="index"  
 @click="clickSong*(*index*)*"  
 :class="*{*activeCard:precentIndex===index*}*"  
 */>  
 </*van-collapse-item*>  
 </*van-collapse*>  
  
 <*audio :src="precentSong" ref="audioRef"*>* 当前浏览器不支持audio  
 *</*audio*>  
 <*div class="backtome" v-if="isOther" @click="backtome"*>* 分享列表  
 *<*van-icon name="https://b.yzcdn.cn/vant/icon-demo-1126.png" size="30"*/>  
 </*div*>  
 </*div*>  
</*template*>  
<*script*>*import *{*getMySong,PublicUserSong*}* from "../utils/query";  
  
export default *{* data*() {* return *{* aa:this.active,  
 test:0,  
 active: 0,  
 checked: false,  
 defaultVal: 0,  
 isPlay:false,  
 precentIndex:0,  
 songList:*[{* url:"",  
 logoImg: "",  
 imginfo: "",  
 song:"",  
 singer: ""  
 *}]*,  
 activeNames: *[*'1'*]*,  
 isOther:false  
 *}  
 }*,  
  
 computed:*{* precentSong*(){* return this.songList*[*this.precentIndex*]*.url;  
 *}*,  
 iconName*(){* return this.checked===true?"pause":"play";  
 *}*,  
 *}*,  
 created*() {* this.userId=this.$route.params.id;  
 if*(*this.userId*){* this.isOther=true  
 PublicUserSong*(*this.userId*)*.then*(*res=>*{* this.songList=res.data;  
 *})*.catch*(*err=>*{* ***console***.log*(*err*)*;  
 *})  
 }*else*{* getMySong*()*.then*(*res=>*{* this.songList=res.data;  
 ***console***.log*(*this.songList*)  
 })*.catch*(*err=>*{* ***console***.log*(*err*)*;  
 *})  
 }  
 }*,  
  
 mounted*() {* this.audio=this.$refs.audioRef;  
  
 *}*,  
  
 methods:*{* clickSong*(*index*){* this.precentIndex=index  
 this.isPlay=false;  
 this.checked=false;  
 this.audio.currentTime=0;  
 setTimeout*(*this.toPlaySong,150*)  
 }*,  
 onChange*(*index*) {* this.checked=true;  
 let ratio=this.audio.duration\**(*index/100*)*;  
 this.audio.currentTime=ratio;  
 this.audio.play*()*;  
 this.isPlay=true;  
 *}*,  
 nextSong*(){* if*(*this.precentIndex<this.songList.length*) {* this.precentIndex++;  
 this.isPlay=false;  
 this.checked=false;  
 this.audio.currentTime=0;  
 *}  
 }*,  
 lastSong*(){* if*(*this.precentIndex>0*){* this.precentIndex--;  
 this.checked=false;  
 this.isPlay=false;  
 this.audio.currentTime=0;  
 *}  
 }*,  
 toPlaySong*(){* this.checked=!this.checked;  
 this.isPlay=!this.isPlay;  
 let intervalTime;  
 if*(*this.isPlay*){* this.audio.play*()*;  
 intervalTime=setInterval*(()*=>*{* this.defaultVal=this.audio.currentTime/this.audio.duration\*100  
 *}*,500*)*;  
 *}*else *{* this.audio.pause*()*;  
 clearInterval*(*intervalTime*)*;  
 *}  
 }*,  
 backtome*(){* this.$router.push*({* name:"share"  
 *})* // this.$router.push({  
 // name:"home"  
 // })  
 *}  
 }  
}*;  
*</*script*>  
<*style scoped*>*.todo-container li *{* border: solid 1px white;  
 background-color: beige;  
 margin-top: 1em;  
 padding: .2em;  
*}*.activeCard*{* border: #666666 1px solid;  
*}*.main*{* font-size: .7em;  
*}*.van-badge*{* font-size: 1.5em;  
*}*.my-swipe .van-swipe-item *{* color: #fff;  
 font-size: 20px;  
 line-height: 150px;  
 text-align: center;  
 background-color: #39a9ed;  
*}*.switchBox*{* display: flex;  
 align-items: center;  
 justify-content: center;  
*}*.backtome*{* position: fixed;  
 bottom: 300px;  
 right: 0;  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 font-size: 12px;  
 text-align: center;  
 border: #666666 1px solid;  
 border-radius: 50px;  
 height: 60px;  
 width: 60px;  
 background-color: #ff5777;  
 opacity: .5;  
*}  
</*style*>*

### 音乐分享界面



此处展示所有账户为公共设置的用户，并可以进入他们的歌单

关键代码：

*<*template*>  
 <*div*>  
 <*van-sticky*>  
 <*van-button type="info" style="width: 100%;"*>*参与分享吧*</*van-button*>  
 </*van-sticky*>  
  
 <*van-contact-card type="edit" :name="i.userName" :tel="i.userPhone" v-for="*(*i,index*)* in users" :key="index"  
 @click="clickuser*(*index*)*"  
 */>  
 </*div*>  
</*template*>  
  
<*script*>*import *{*getPublicUser*}* from "../utils/query";  
  
export default *{* name: "share",  
 data*(){* return*{* users:*[]  
 }  
 }*,  
 created*() {* getPublicUser*()*.then*(*res=>*{* ***console***.log*(*res*)* this.users=res.data;  
 *})  
 }*,  
 methods:*{* clickuser*(*index*){* ***console***.log*(*5*)* this.$router.push*({* name: 'home',  
 params: *{* id: this.users*[*index*]*.userId  
 *}  
 })  
 }  
 }  
}  
</*script*>  
  
<*style scoped*>  
  
</*style*>*

注册界面

关键代码：

个人信息界面：



Express 后端部分

框架 采用Express Mysql

MySql 工具使用了Mysql-Front

文件目录：

├─.idea 这是编译器的配置信息

│ └─inspectionProfiles

├─bin 这是主要执行的点

├─controller 这里是对数据库的操作

├─dao 这里是连接数据库的池

├─middles 这里放着中间件

├─node\_modules 这是项目依赖

├─public 这是打包后的Vue（run build 后的）文件

│ └─static

│ ├─css

│ ├─fonts

│ └─js

│ ─index.html

├─routes 这里是路由层配置

└─services 这里是类似与orm，对sql进行的封装

Package.json中的项目依赖

{

"name": "shop\_system",

"version": "0.0.0",

"private": true,

"scripts": {

"start": "nodemon ./bin/www"

},

"dependencies": {

"cookie-parser": "~1.4.4",

"cores": "^0.8.5",

"debug": "~2.6.9",

"ejs": "~2.6.1",

"express": "~4.16.1",

"express-session": "^1.17.2",

"http-errors": "~1.6.3",

"mockjs": "^1.1.0",

"morgan": "~1.9.1",

"multer": "^1.4.2",

"mysql2": "^2.2.5",

"vue-puzzle-vcode": "^1.1.5"

},

"devDependencies": {

"cors": "^2.8.5",

"jsonwebtoken": "^8.5.1"

}

}

后端路由：

//获取当前未完成todo

router.get('/current', current);

// 发布新TODO

router.post('/new\_todo', new\_todo);

// 更改Todo状态

router.put('/end', end\_todo);

router.put('/start', start\_todo);

// 历史Todo

router.get('/history', history\_todo);

// 分享 Todo

router.put('/share\_todo', share\_todo);

// 获取所有的分享的Todo、

router.get('/get\_share\_todo', get\_share\_todo);

//取消分享 Todo

router.delete('/del\_share\_todo', del\_share\_todo);

//注册接口

router.post('/register', register);

//登陆接口

router.post('/login', login);

//注销接口

router.all("/logout", loginout);

//获取用户信息

router.get("/userinfo", userinfo)

// 用户修改信息

router.put("/modify", modify)

数据库的连接

//加载mysql数据库连接库

//加载mysql数据库连接库

const mysql = require("mysql2");

const config = require("./config")

//调用mysql.createPool方法创建书库连接池

const pool = mysql.createPool(config);

// 中间信息查询确认,实际上线就干掉

pool.on('acquire', function (connection) {

console.log('Connection %d acquired', connection.threadId);

});

pool.on('connection', function (connection) {

connection.query('SET SESSION auto\_increment\_increment=1')

});

pool.on('enqueue', function () {

console.log('Waiting for available connection slot');

});

pool.on('release', function (connection) {

console.log('Connection %d released', connection.threadId);

});

module.exports = pool;

数据库操作部分，进行模块分离

用户的数据操作接口

//使用连接池

let pool = require('../dao/pool');

// 数据采用sha256加密

const crypto = require('crypto');

// 加入JWT ,做用户校验

const jwt = require('jsonwebtoken');

const secretKey = '6666';

const verify = (token, callback) => {

jwt.verify(

token,

secretKey,

callback);

}

// 根据用户名 获取user\_id

const getUserIdByUserPhone = (userPhone, callback) => {

pool.query(

"select userId from users where userPhone=? ",

[userPhone],

callback

)

}

/\*\*

\* @param {string} user\_account

\* @param {string} user\_password

\* @return {void}

\* \*/

//根据用户名密码获取用户信息，供登录使用

function getUserByAccountAndPassword({ userPhone, password }, callback) {

let hash = crypto.createHash('sha256');

hash.update(password)

pool.query(

"select userId from users where userPhone=? and password=?",

[userPhone, hash.digest("hex")],

callback

);

}

//添加用户信息，供注册使用

function addUser({

userName,

password,

userPhone

}, callback) {

console.log(password)

// 判断优先判断是否存在该用户

let hash = crypto.createHash('sha256');

hash.update(password)

pool.getConnection((err, connection) => {

connection.query(`insert into users(

userName,

password,

userPhone)

values(?,?,?)`,

[userName,

hash.digest("hex"),

userPhone],

(err, data) => {

callback(err, data)

console.log(20)

connection.release();

})

})

}

//根据Token 获取用户的信息

/\*\*

\* @param {string} token

\* @param {function} callback

\* @return {void}

\* \*/

function getUserInfoByToken({ token }, callback) {

verify(token, (err, { userId }) => {

// console.log(data)

pool.query(

`select

userName,userPhone

from users where userId=?`,

[userId],

callback

)

})

}

function uploadMusic({ userId, url, singer = '暂无信息', desc = '暂无描述', publishTime = '0-0-0' }, callback) {

pool.query(

`insert into music(userId,url,singer,someDesc,publishTime)values(?,?,?,?,?)`,

[userId, url, singer, desc, publishTime],

callback

)

}

function changePublic({token}, callback){

verify(token,(err,{userId})=>{

if(err){

console.log(err)

return

}

pool.query("select status from users where userId=?",[userId],(err,data)=>{

if(err){

console.log(err)

return

}

console.log(data[0].status===1)

if(data[0].status===1){

pool.query("update users set status=0 where userId=?",[userId],(err,data)=>{

console.log(data)

callback;

})

}else{

pool.query("update users set status=1 where userId=?",[userId],(err,data)=>{

console.log(data)

callback;

})

}

})

}

)

}

function getMyStatus(token, callback){

verify(token,(err,{userId})=>{

if(err){

console.log(err)

return

}

pool.query(

`

select status from users where userId=?

`,[userId],callback)

})

}

//指明可供其它模块使用的模块

module.exports = {

getUserIdByUserPhone,

addUser,

getUserInfoByToken,

uploadMusic,

getUserByAccountAndPassword,

changePublic,

getMyStatus

};

路由设置：

const {register, login, changeMyPublic,getMyPublic,userinfo,fileupload,getUserInfoByToken} = require("../controller/users");

let express = require('express');

let router = express.Router();

//注册接口

router.post('/register', register);

//登陆接口

router.post('/login', login);

//获取用户信息

router.get("/userinfo",userinfo)

router.post('/fileupload',fileupload)

router.post('/getStatus',getMyPublic)

router.post('/changeStatus',changeMyPublic)

router.post('/getProfile',getUserInfoByToken)

module.exports = router;

let express = require('express');

const {current,getPublicU,PublicUserSong} = require("../controller/main");

let router = express.Router();

/\* GET users listing. \*/

router.post('/current', current);

router.post('/getPublicU', getPublicU);

router.post('/PublicUserSong', PublicUserSong);

module.exports = router;

主要使用的中间件：

