

一 管理端用户模块

模块时长：240min

学习目标

1. 软件工程
2. 数据库创建
3. 项目环境搭建
4. 用户模块功能

1 软件工程

1.1 简介

是研究和应用如何以系统性的、规范化的、可量化的过程化方法去开发和维护软件，以及如何把正确的管理技术和当前最好的技术方法结合起来的一门学科。

1.2 实施

1.2.1 步骤

1. 制定计划

确定要开发软件系统的总目标。给出功能、性能、可靠性以及接口等方面的要求，完成该任务的可行性研究，估计可利用的资源。成果有<<可行性分析报告>> <<软件计划说明书>>

2. 需求分析

对用户提出的需求进行分析并给出详细的定义。编写软件需求说明书或系统功能说明书及初步的系统用户手册，提交管理机构评审。成果有<<软件需求说明书>>

3. 软件设计

把各项需求转换成软件的体系结构，对每个模块要完成的工作进行具体的描述，为源程序编写打下基础。成果有<<系统设计说明书>> <<详细设计说明书>>

4. 程序编码

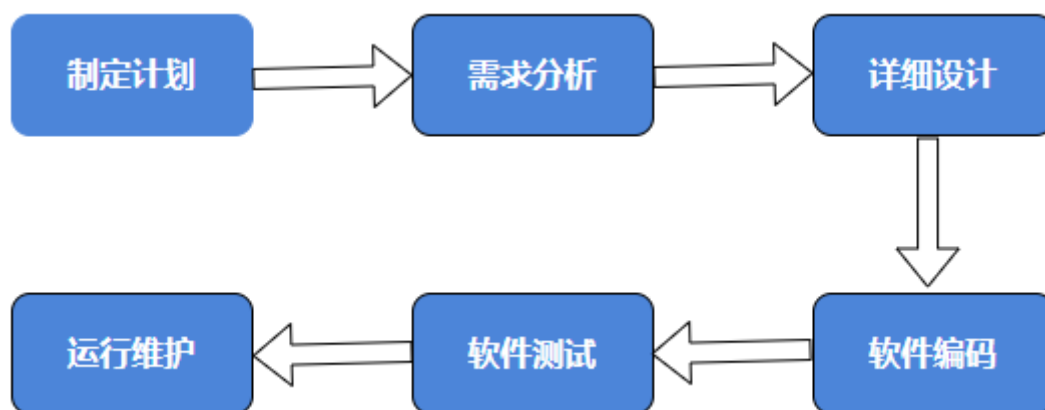
把程序设计转换成计算机程序代码，即写成以某一种特定程序设计语言表示的源程序。

5. 软件测试

查找各模块在功能和结构上存在的问题并加以纠正。将已测试过的模块按一定顺序组装起来测试。成果有<<测试计划说明书>> <<测试分析报告>>

6. 运行维护

运行中发现了软件中的错误需要修正。变化了的软件工作环境，需做是适当的变更。成果有<<用户操作手册>> <<软件维护手册>>



2 数据库创建

2.1 分析

2.1.1 数据库创建

1. 根据需求分析和详细设计，创建数据库概念模型。

管理员信息表	
编号	Integer
姓名	Variable characters (100)
密码	Variable characters (100)

用户表	
编号	Integer
用户名	Variable characters (100)
密码	Variable characters (100)
性别	Characters (2)
电话号码	Characters (15)
邮箱	Variable characters (30)
生日	Date & Time
简介	Variable characters (200)
薪资	Variable characters (100)
图像	Variable characters (200)
创建时间	Date & Time
更新日期	Date & Time

2. 根据具体数据库类型，创建物理模型。生成sql语句

管理员信息表		
编号	int	<pk>
姓名	varchar(100)	
密码	varchar(100)	

用户表		
编号	int	<pk>
用户名	varchar(100)	
密码	varchar(100)	
性别	char(2)	
电话号码	char(15)	
邮箱	varchar(30)	
生日	datetime	
简介	varchar(200)	
薪资	varchar(100)	
图像	varchar(200)	
创建时间	datetime	
更新日期	datetime	

3. 初始化系统数据

备注 此处先在MySQL中创建OnlineMusic数据库，然后导入提供的OnlineMusic.sql语句

3 项目环境搭建

3.1 后端环境搭建

3.1.1 后台框架搭建

1. 打开IDEA, 创建SpringBoot项目

Server URL: start.spring.io

Name:

Location:
Project will be created in: F:\ideaWorkspace\OnlineMusic-Server

☐ Create Git repository

Language: ☐ Java ☐ Kotlin ☐ Groovy

Type: ☐ Gradle - Groovy ☐ Gradle - Kotlin ☐ Maven

Group:

Artifact:

Package name:

JDK:

Java:

2. 添加依赖 pom.xml

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.2</version>
    <relativePath/>
</parent>
.....
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
.....
<!--阿里云OSS服务-->
<dependency>
    <groupId>com.aliyun.oss</groupId>
    <artifactId>aliyun-sdk-oss</artifactId>
    <version>3.10.2</version>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-logging</artifactId>
```

```

        </exclusion>
    </exclusions>
</dependency>

<!--引入测试-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<!--mysql-connector-java就是帮助java程序操作mysql的驱动程序。通过与mysql服务端建立连接，发送sql语句并且获取结果集-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.16</version>
</dependency>

<!--mybatis用来和数据库进行交互-->

<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-starter</artifactId>
    <version>3.5.1</version>
</dependency>

<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>2.1.4</version>
</dependency>

<!--这里面有点小工具啥的-->
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.8.1</version>
</dependency>

<!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.47</version>
</dependency>

<!-- Log4j -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-log4j</artifactId>
    <version>1.3.8.RELEASE</version>

```

```
</dependency>

<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-to-slf4j</artifactId>
    <version>2.8.2</version>
</dependency>

<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.1</version>
</dependency>

<!-- 热部署模块 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
</dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>

<!-- redis -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
    <version>2.6.6</version>
</dependency>
<!--<spring2.X集成redis所需common-pool2-->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-pool2</artifactId>
    <version>2.11.1</version>
</dependency>
<!--阿里云短信服务依赖包-->
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>dysmsapi20170525</artifactId>
    <version>2.0.23</version>
</dependency>
<!--支付宝登录SDK-->
<dependency>
    <groupId>com.alipay.sdk</groupId>
    <artifactId>alipay-sdk-java</artifactId>
    <version>4.35.37.ALL</version>
</dependency>
<!--各种第三方登录-->
<dependency>
    <groupId>me.zhyd.oauth</groupId>
    <artifactId>JustAuth</artifactId>
    <version>1.16.5</version>
```

```

</dependency>
<!-- 邮件-->
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>javax.mail-api</artifactId>
    <version>1.6.2</version>
</dependency>
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>1.4.7</version>
</dependency>
<!-- 百度人脸识别-->
<dependency>
    <groupId>com.baidu.aip</groupId>
    <artifactId>java-sdk</artifactId>
    <version>4.16.16</version>
</dependency>
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>4.10.0</version>
</dependency>
<!-- JWT依赖-->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>

```

3. 添加跨域拦截器 config/CorsInterceptor.java

```

public class CorsInterceptor extends HandlerInterceptorAdapter {

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler) {
        response.setHeader("Access-Control-Allow-Origin",
request.getHeader("Origin"));
        response.setHeader("Access-Control-Allow-Methods", "POST, GET,
OPTIONS, DELETE");
        response.setHeader("Access-Control-Max-Age", "3600");
        response.setHeader("Access-Control-Allow-Headers",
"x_requested_with,x-requested-with,Authorization,Content-Type,token");
        response.setHeader("Access-Control-Allow-Credentials", "true");
        return true;
    }
}

```

4. 添加配置类 方便跨域访问

跨域指的是浏览器不能执行其他网站的脚本，从一个域名的网页去请求另一个域名的资源时，域名、端口、协议任一不同，都是跨域。跨域是由浏览器的同源策略造成的，是浏览器施加的安全限制。

config/WebMvcConfig.java

```
/**
 * SpringBoot解决跨域方案有多种，重写WebMvcConfigurer是全局解决跨域方案。
 */
@Configuration
public class WebMvcConfig implements WebMvcConfigurer {

    @Bean
    public CorsInterceptor corsInterceptor() {
        return new CorsInterceptor();
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(corsInterceptor())
            .addPathPatterns("/**");
    }

}
```

5. 拷贝资源文件

拷贝图片img文件夹 音乐song文件夹等到后台项目的根目录下

6. 拷贝日志配置文件log4j.properties到resources目录下

7. 添加图片 音乐等映射处理文件

config/WebPicConfig.java

```
@Configuration
public class WebPicConfig implements WebMvcConfigurer {

    //TODO 这个配置类的目的是什么 就是注册了一个类似于拦截器吧 看到对应的资源 会将其修
    改成相应的地址
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/img/avatarImages/**")
            .addResourceLocations(Constants.AVATOR_IMAGES_PATH);
        registry.addResourceHandler("/img/singerPic/**")
            .addResourceLocations(Constants.SINGER_PIC_PATH);
        registry.addResourceHandler("/img/songPic/**")
            .addResourceLocations(Constants.SONG_PIC_PATH);
        registry.addResourceHandler("/song/**")
            .addResourceLocations(Constants.SONG_PATH);
        registry.addResourceHandler("/img/songListPic/**")
            .addResourceLocations(Constants.SONGLIST_PIC_PATH);
        registry.addResourceHandler("/img/swiper/**")
            .addResourceLocations(Constants.BANNER_PIC_PATH);
    }

}
```

constant/Constants.java

```
public class Constants {
```

```

/* 歌曲图片，歌手图片，歌曲文件，歌单图片等文件的存放路径 */
public static String ASSETS_PATH = System.getProperty("user.dir");

public static String AVATOR_IMAGES_PATH = "file:" + ASSETS_PATH +
"/img/avatorImages/";
public static String SONGLIST_PIC_PATH = "file:" + ASSETS_PATH +
"/img/songListPic/";
public static String SONG_PIC_PATH = "file:" + ASSETS_PATH +
"/img/songPic/";
public static String SONG_PATH = "file:" + ASSETS_PATH + "/song/";
public static String SINGER_PIC_PATH = "file:" + ASSETS_PATH +
"/img/singerPic/";
public static String BANNER_PIC_PATH = "file:" + ASSETS_PATH +
"/img/swiper/";

/* 盐值加密 */
public static String SALT = "xsdfsdfujiwjerwek1231yz";
}

```

8. 修改端口 application.properties

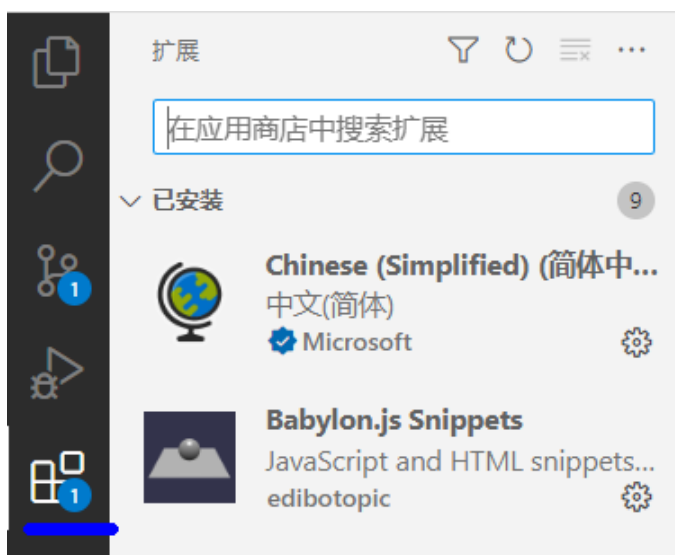
```
server.port=8888
```

3.2 前端项目搭建

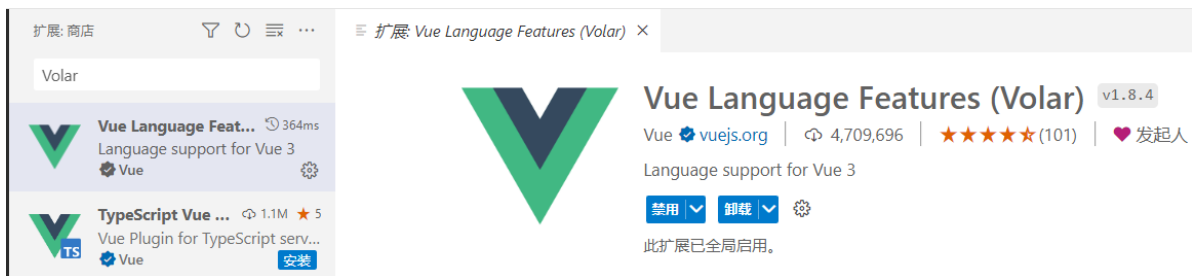
3.2.1 插件安装

3.2.1.1 Volar插件

打开Visual Studio Code，选择左侧菜单栏的扩展按钮



搜索框中输入Volar，然后选择第一个，选中安装即可。



3.2.2 前端框架搭建

1. 在Visual Studio Code中打开源代码文件夹，右键选择在集成终端中打开。cmd进入DOS命令界面。

```
问题 输出 终端 调试控制台

PS F:\VueWorkspace\code> cmd
Microsoft Windows [版本 10.0.19045.3086]
(c) Microsoft Corporation。保留所有权利。

F:\VueWorkspace\code>
```

2. 创建Vue项目 项目目必须都是小写

```
问题 输出 终端 调试控制台

F:\VueWorkspace\code>vue create onlinemusic-admin
```

3. 选择手工方式

```
? Please pick a preset:
  Default ([Vue 2] babel, eslint)
  Default (Vue 3) ([Vue 3] babel, eslint)
> Manually select features
```

4. 使用下箭头移动，空格键选中。Vue version Babel Typescript Router Vuex eslint

```
? Please pick a preset: Manually select features
? Check the features needed for your project:
> (*) Choose Vue version
  (*) Babel
  (*) TypeScript
  ( ) Progressive Web App (PWA) Support
  (*) Router
  (*) Vuex
  ( ) CSS Pre-processors
  (*) Linter / Formatter
  ( ) Unit Testing
  ( ) E2E Testing
```

5. 回车键后选择Vue3

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Linter
? Choose a version of Vue.js that you want to start the project with
  2.x
> 3.x
```

6. 选择N

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? (y/N) N
```

7. 选择n

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JS): Y/n) n
```

8. Router选择

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX)? No
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) n
```

9. Eslint

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JS): No
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a linter / formatter config: (Use arrow keys)
> ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier
  TSLint (deprecated)
```

10. 保存选择默认

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX)? No
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a linter / formatter config: Basic
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to invert selection)
>(*) Lint on save
  ( ) Lint and fix on commit
```


11. 保存单独文件



```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, Router, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JS): No
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a linter / formatter config: Basic
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

12. 是否为后续保存该设置 选择默认

13. 进入项目根目录，执行启动命令测试

问题 输出 终端 调试控制台

 Generating README.md...

 Successfully created project **onlinemusic-admin**.
 Get started with the following commands:

```
$ cd onlinemusic-admin
$ npm run serve
```

```
F:\VueWorkspace\code>cd onlinemusic-admin
```

```
F:\VueWorkspace\code\onlinemusic-admin>npm run serve
```

14. 按照终端提示在浏览器输入 <http://localhost:8080/>

参考 https://blog.csdn.net/weixin_47529373/article/details/119052636

15. 增加配置文件 根目录创建vue.config.js

```
const { defineConfig } = require('@vue/cli-service')

module.exports = defineConfig({
  lintOnSave: false,
  transpileDependencies: true,
  chainWebpack: config => {
    config.plugin('define').tap(definitions => {
      Object.assign(definitions[0]['process.env'], {
        //配置变量NODE_HOST方便访问远程服务器
        NODE_HOST: '"http://localhost:8888"',
      });
      return definitions;
    });
  }
})
```

16. 修改配置 在tsconfig.json 添加

```
"sourceMap": true,
//compilerOptions的sourceMap属性后面即可
"noImplicitAny": false,
"strictNullChecks": false,
```

17. 添加css assets/css/main.css

18. 添加font assets/icons/iconfont.js

19. 添加背景图片 assets拷贝images文件夹

20. 引入 css和font main.ts

```
import "../assets/css/main.css";
import "../assets/icons/iconfont.js";
```

21. 修改 main.ts

```
declare module "@vue/runtime-core" {
  interface State {
    count: number;
  }

  interface ComponentCustomProperties {
    $store: Store<State>;
  }
}
```

22. 添加模块和类型声明 shims-vue.d.ts文件中 为 Vue的单文件组件和其他非 TypeScript 模块提供类型信息

```
declare module "vue/types/vue" {
  import VueRouter, { Route } from "vue-router";

  interface Vue {
    $router: VueRouter;
    $route: Route;
  }
}
interface ResponseBody {
  code: string;
  success: boolean;
  message: string;
  type: string;
  data?: any;
}
```

3.2.3 安装相关依赖

3.2.3.1 Vue Router路由

在单页面应用(Single-page application) 中, 客户端的 JavaScript 需要拦截页面的跳转请求, 动态获取新的数据, 然后在无需重新加载的情况下更新当前页面。需要使用路由技术, 实质就是页面跳转管理。Vue Router 是Vue 的官方路由。

1. 安装

```
npm install vue-router@4
```

备注 上面3.2.2已安装的话, 这个步骤可忽略

2. 添加页面 src/views/Login.vue

```

<template>
  <div>登录页面</div>
</template>
<script>
</script>
<style scoped></style>

```

3. 修改配置文件 src/router/index.ts

```

import { createRouter, createWebHistory, RouteRecordRaw } from 'vue-router'
import Login from '../views/Login.vue'
//使用RouteRecordRaw声明的对象会被当做路由对象，放入到路由对象池里
const routes: Array<RouteRecordRaw> = [
  {
    path: '/',
    name: 'Login',
    component: Login
  },
]
//createRouter创建路由;
const router = createRouter({
  //createWebHistory代表路由使用 HTML5 模式，也是推荐使用的模式;
  history: createWebHistory(process.env.BASE_URL),
  //routes: routes的缩写
  routes
})
//导出以便其他页面使用
export default router

```

4. 启用路由 main.ts中引入，上面已经配置过了则忽略

```

import router from './router'
.....
createApp(App).use(store).use(router).mount('#app')

```

3.2.3.2 axios

是一个基于Promise的HTTP 库，可以发送get、post等请求。主要用于向后台发送请求。

安装axios 项目正在运行的话需要Ctrl+C停止

npm install axios

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

F:\VueWorkspace\code\onlinemusic-admin>npm install axios

added 3 packages in 33s

F:\VueWorkspace\code\onlinemusic-admin>

```

创建工具类以便复用 src/api/request.ts

```
import axios from 'axios'
import router from '../router'
const BASE_URL = process.env.NODE_HOST
```

```
axios.defaults.timeout = 5000 // 超时时间设置
axios.defaults.withCredentials = true // true允许跨域
axios.defaults.baseURL = BASE_URL
// Content-Type 响应头
axios.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded;charset=UTF-8'
```

//请求拦截器 在发请求之前 对请求进行处理。一般把本地token添加到请求头里面执行登录认证，这样在发请求时就不用考虑并处理token了

```
axios.interceptors.request.use(
  //在发送请求前动作
  config => {
    //是否存在token 存在则header都加上token,不用每次请求手动添加
    if (localStorage.getItem('token')) {
      config.headers.token = localStorage.getItem('token');
    }
    return config;
  },
  //在请求错误后处理
  error => {
    //可添加错误处理
    return Promise.reject(error);
  }
)
```

// 响应拦截器

```
axios.interceptors.response.use(
  response => {
    // 如果返回的状态码为200，说明接口请求成功，可以正常拿到数据
    // 否则的话抛出错误
    if (response.status === 200) {
      return Promise.resolve(response)
    } else {
      return Promise.reject(response)
    }
  },
  // 服务器状态码不是2开头的的情况
  error => {
    console.log(error)
    if (error.response.status) {
      switch (error.response.status) {
        // 401: 未登录
        case 401:
          router.replace({
            path: "/",
            query: {
              // redirect: router.currentRoute.fullPath
            },
          });
          break;
      }
    }
  }
)
```

的页面

```
        case 403:
            // console.log('管理员权限已修改请重新登录')
            // 跳转登录页面，并将要浏览的页面fullPath传过去，登录成功后跳转需要访问
            的页面

            setTimeout(() => {
                router.replace({
                    path: "/",
                    query: {
                        // redirect: router.currentRoute.fullPath
                    },
                });
            }, 1000);
            break;

            // 404请求不存在
            case 404:
                // console.log('请求页面飞到火星去了')
                break;
        }
        return Promise.reject(error.response);
    }
}

export function getBaseURL() {
    return BASE_URL;
}

/**
 * 封装get方法
 * @param url
 * @param data
 * @returns {Promise}
 */
export function get(url, params?: object) {
    return new Promise((resolve, reject) => {
        axios.get(url, params).then(
            response => resolve(response.data),
            error => reject(error)
        )
    });
}

/**
 * 封装post请求
 * @param url
 * @param data
 * @returns {Promise}
 */
export function post(url, data = {}) {
    return new Promise((resolve, reject) => {
        axios.post(url, data).then(
            response => resolve(response.data),
            error => reject(error)
        );
    });
}
```

```

    });
  }

  /**
   * 封装delete请求
   * @param url
   * @param data
   * @returns {Promise}
   */
  export function deletes(url, data = {}) {
    return new Promise((resolve, reject) => {
      axios.delete(url, data).then(
        response => resolve(response.data),
        error => reject(error)
      );
    });
  }

  /**
   * 封装put请求
   * @param url
   * @param data
   * @returns {Promise}
   */
  export function put(url, data = {}) {
    return new Promise((resolve, reject) => {
      axios.put(url, data).then(
        response => resolve(response.data),
        error => reject(error)
      );
    });
  }
}

```

3.2.3.3 Element Plus

是一个基于 Vue 3 的高质量 UI 组件库。它包含了丰富的组件和扩展功能，例如表格、表单、按钮、导航、通知等，让开发者能够快速构建高质量的 Web 应用。<https://element-plus.org/zh-CN/guide/installation.html>

1. 安装

```
npm install element-plus --save
```

2. 配置 main.ts

```

import ElementPlus from "element-plus";
import "element-plus/dist/index.css";
createApp(App).use(store).use(router).use(ElementPlus).mount('#app')

```

3.2.3.4 Icon 图标

Element Plus 提供了一套常用的图标集合。

安装

```
npm install @element-plus/icons-vue
```


3.2.3.5 Echarts

方便在Vue3中使用Echarts的图表组件。

安装

```
npm install echarts --save
```

3.2.3.5 Mitt

实现全局事件的发布和订阅与取消订阅，也就是跨组件通讯。

1. 安装

```
npm i -S mitt
```

2. 配置 utils/emitter.ts

```
import mitt from "mitt";

const emitter = mitt();

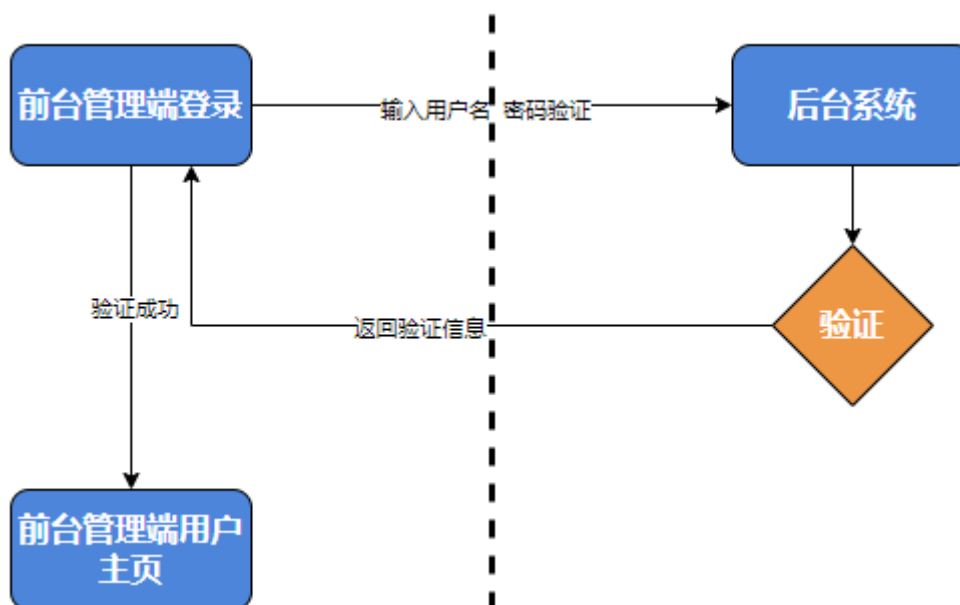
export default emitter;
```

4 管理端后台用户登录模块

4.1 需求分析

4.1.1 业务流程

1. 用户输入用户名和密码，发送给后端和数据库比对验证，用户是否存在，密码是否正确
2. 服务端验证用户名密码是否正确。正确的话返回登录成功信息，否则返回失败信息。
3. 管理端接收到服务端返回信息后，给用户提示，同时跳转到首页页面



4.2 创建数据库表及PO类型

4.2.1 创建数据库表

```
CREATE TABLE `admin` (  
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT '管理员编号',  
  `name` varchar(45) NOT NULL COMMENT '姓名',  
  `password` varchar(45) NOT NULL COMMENT '密码',  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `name_UNIQUE` (`name`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;  
  
INSERT INTO `admin` VALUES (1,'admin','123'),(2,'admin1','565');  
  
CREATE TABLE `consumer`  
(  
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT '用户编号',  
  `username` varchar(255) NOT NULL COMMENT '用户名',  
  `password` varchar(100) NOT NULL COMMENT '密码',  
  `sex` tinyint(4) DEFAULT NULL COMMENT '性别',  
  `phone_num` char(15) DEFAULT NULL COMMENT '电话',  
  `email` char(30) DEFAULT NULL COMMENT '电子邮件',  
  `birth` datetime DEFAULT NULL COMMENT '生日',  
  `introduction` varchar(255) DEFAULT NULL COMMENT '简介',  
  `location` varchar(45) DEFAULT NULL COMMENT '位置',  
  `avator` varchar(255) DEFAULT NULL COMMENT '图像',  
  `create_time` datetime NOT NULL COMMENT '创建时间',  
  `update_time` datetime NOT NULL COMMENT '更新时间',  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `username_UNIQUE` (`username`),  
  UNIQUE KEY `phone_num_UNIQUE` (`phone_num`),  
  UNIQUE KEY `email_UNIQUE` (`email`)  
) ENGINE=InnoDB AUTO_INCREMENT=64 DEFAULT CHARSET=utf8;  
  
//初始数据  
INSERT INTO `consumer`  
VALUES (1, 'Yin33', '123', 0, '13776412237', 'yoona@qq.com', '2019-05-21  
00:00:00', '好好吃饭', '山西',  
  '/img/avatorImages/1547477117223tou.jpg', '2019-01-04 21:42:24', '2022-  
04-18 01:19:12');  
INSERT INTO `consumer`  
VALUES (2, '012', '012', 0, '13754803255', 'love@gmail.com', '2019-04-24  
00:00:00', '我就喜欢吃', '北京',  
  '/img/avatorImages/1547477117223tou.jpg', '2019-01-05 15:02:45', '2020-  
03-23 01:24:59');  
INSERT INTO `consumer`  
VALUES (5, '789', '789', 0, '13634377258', '666@126.com', '2019-01-08 00:00:00',  
'今天很开心啊', '山西',  
  '/img/avatorImages/1547477117223tou.jpg', '2019-01-07 16:16:42', '2022-  
04-10 01:59:13');  
INSERT INTO `consumer`  
VALUES (8, 'tawuhen', '123', 0, '', '192673541@qq.com', '2019-04-25 18:58:39',  
'你好', '北京',  
  '/img/avatorImages/1547477117223tou.jpg', '2019-04-25 00:28:58', '2019-  
04-25 18:58:39');  
INSERT INTO `consumer`
```

```

VALUES (12, 'yoona', '123', 0, '13854173655', '1236795@qq.com', '2019-04-25
00:00:00', '好好吃饭', '北京',
'/img/avatorImages/1547477117223tou.jpg', '2019-04-25 10:56:54', '2020-
04-29 01:28:37');
INSERT INTO `consumer`
VALUES (16, '1234321', '123', 1, '13754803257', '123@qq.com', '2020-03-08
17:25:45', '', '',
'/img/avatorImages/user.jpg', '2020-03-08 17:25:45', '2020-03-08
17:25:45');
INSERT INTO `consumer`
VALUES (24, 'yoonaAA', '123', 1, NULL, NULL, '2020-03-04 00:00:00', '', '',
'/img/avatorImages/user.jpg',
'2020-03-21 22:20:27', '2020-03-21 22:20:27');
INSERT INTO `consumer`
VALUES (25, 'yoonaAB', '123', 1, NULL, NULL, '2020-03-02 00:00:00', '', '',
'/img/avatorImages/user.jpg',
'2020-03-21 22:21:50', '2020-03-21 22:21:50');
INSERT INTO `consumer`
VALUES (26, 'yoonaAC', '123', 1, 'null', 'null', '2020-03-11 00:00:00', '', '',
'/img/avatorImages/user.jpg',
'2020-03-21 22:23:43', '2020-05-14 21:12:56');

```

```

CREATE TABLE `collect`
(
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT '主键',
  `user_id` int(10) unsigned NOT NULL COMMENT '收藏者id',
  `type` tinyint(4) NOT NULL COMMENT '收藏类型',
  `song_id` int(10) unsigned DEFAULT NULL COMMENT '歌曲编号',
  `song_list_id` int(10) unsigned DEFAULT NULL COMMENT '歌单编号',
  `create_time` datetime NOT NULL COMMENT '创建时间',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=105 DEFAULT CHARSET=utf8;

```

```

INSERT INTO `collect`
VALUES (4, 94, 0, 23, NULL, '2019-01-07 16:41:44');
INSERT INTO `collect`
VALUES (10, 94, 0, 3, NULL, '2019-01-07 16:58:59');
INSERT INTO `collect`
VALUES (16, 94, 0, 24, NULL, '2019-01-07 17:34:07');
INSERT INTO `collect`
VALUES (21, 5, 0, 24, NULL, '2019-01-08 15:18:33');
INSERT INTO `collect`
VALUES (24, 5, 0, 8, NULL, '2019-01-08 16:07:57');
INSERT INTO `collect`
VALUES (43, 5, 0, 7, NULL, '2019-04-26 01:06:20');
INSERT INTO `collect`
VALUES (45, 26, 0, 44, NULL, '2020-03-21 22:26:37');
INSERT INTO `collect`
VALUES (46, 26, 0, 36, NULL, '2020-03-21 22:28:24');
INSERT INTO `collect`
VALUES (47, 26, 0, 69, NULL, '2020-03-22 01:56:54');
INSERT INTO `collect`
VALUES (48, 26, 0, 45, NULL, '2020-03-22 02:08:36');

```

```
INSERT INTO `collect`  
VALUES (50, 26, 0, 100, NULL, '2020-03-22 03:41:14');
```

4.2.2 生成pojo类

- Admin.java model/domain下

```
@TableName(value = "admin")  
@Data  
public class Admin implements Serializable {  
    @TableId(type = IdType.AUTO)  
    private Integer id;  
  
    private String name;  
  
    private String password;  
}
```

- Consumer.java

```
@TableName(value = "consumer")  
@Data  
public class Consumer {  
    @TableId(type = IdType.AUTO)  
    private Integer id;  
  
    private String username;  
  
    private String password;  
  
    private Byte sex;  
  
    private String phoneNum;  
  
    private String email;  
  
    private Date birth;  
  
    private String introduction;  
  
    private String location;  
  
    private String avator;  
  
    @TableField(fill = FieldFill.INSERT)  
    private Date createTime;  
  
    @TableField(fill = FieldFill.INSERT_UPDATE)  
    private Date updateTime;  
  
    @Override  
    public String toString() {  
        return ToStringBuilder.reflectionToString(this);  
    }  
}
```

```
}
```

- Collect.java

```
@TableName(value = "collect")
@Data
public class collect {
    @TableId(type = IdType.AUTO)
    private Integer id;

    private Integer userId;

    private Byte type;

    private Integer songId;

    private Integer songListId;

    @TableField(fill = FieldFill.INSERT)
    private Date createTime;

    @Override
    public String toString() {
        return ToStringBuilder.reflectionToString(this);
    }
}
```

4.3 设计接口

4.3.1 接口设计分析

1. 获取前端用户输入的用户名和密码，传递到底层数据库进行校验。通过的话返回成功信息，否则返回错误信息。
2. 获取所有的客户端用户信息，显示在当前页面。
3. 获取指定用户收藏信息，显示在当前页面

4.3.3 定义接口

- controller/AdminController.java

```

@RestController
public class AdminController {
    @Autowired
    private AdminService adminService;

    // 判断是否登录成功
    @PostMapping("/admin/login/status")
    public R loginStatus(@RequestBody AdminRequest adminRequest, HttpSession session) {
        R result = adminService.verityPasswd(adminRequest, session);
        return result;
    }
}

```

- controller/ConsumerController.java

```

@RestController
public class ConsumerController {
    @Autowired
    private ConsumerService consumerService;
    /**
     *管理界面调用：返回所有用户
     */
    @GetMapping("/user")
    public R allUser() {
        return consumerService.allUser();
    }
    /**
     *删除用户
     */
    @GetMapping("/user/delete")
    public R deleteUser(@RequestParam int id) {
        return consumerService.deleteUser(id);
    }
}

```

- CollectController.java

```

@RestController
public class CollectController {
    @Autowired
    private CollectService collectService;
    /**
     * 返回的指定用户 ID 收藏的列表
     */
    @GetMapping("/collection/detail")
    public R collectionOfUser(@RequestParam Integer userId) {
        return collectService.collectionOfUser(userId);
    }
}

```

启动类添加Mapper扫描

```
@MapperScan("com.niit.onlinemusicserver.mapper")
@SpringBootApplication
public class OnlineMusicServerApplication {}
```

4.3.4 辅助类 common/R.java

```
@Data
public class R {

    private int code;

    private String message;

    private String type;

    private Boolean success;

    private Object data;

    public static R ok() {
        R r = new R();
        r.setCode(200);
        r.setSuccess(true);
        r.setType("success");
        r.setData(null);
        return r;
    }

    public static R success(String message) {
        R r = new R();
        r.setCode(200);
        r.setMessage(message);
        r.setSuccess(true);
        r.setType("success");
        r.setData(null);
        return r;
    }

    public static R success(String message, Object data) {
        R r = success(message);
        r.setData(data);
        return r;
    }

    public static R warning(String message) {
        R r = error(message);
        r.setType("warning");
        return r;
    }

    public static R error(String message) {
        R r = success(message);
        r.setSuccess(false);
        r.setType("error");
        return r;
    }
}
```

```
    }

    public static R fatal(String message) {
        R r = error(message);
        r.setCode(500);
        return r;
    }
}
```

model/request/AdminRequest.java

```
@Data
public class AdminRequest {
    private Integer id;

    private String username;

    private String password;
}
```

4.4 生成接口文档

4.4.1 用户登录

通过传入的用户名和密码，调用adminService的verityPasswd()进行验证

请求

```
POST /admin/login/status
```

请求参数

参数名	参数位置	参数类型	是否必须	描述
username	查询参数	字符串	是	用户名
password	查询参数	字符串	是	密码
响应				

标志位	类型	描述
true	登录成功	
false	登录	
示例		

```
POST /admin/login/status    请求体数据{"username": "admin","password": "123"}
```


4.4.2 获取用户信息

获取所有用户信息，调用consumerService的allUser()

请求

```
GET /user
```

请求参数

无

状态码	类型	描述
200	查询成功	
404	查询失败	
示例		

```
GET /user
```

4.4.3 删除用户

删除指定用户信息，调用consumerService的deleteUser()方法

请求

```
GET /user/delete
```

请求参数

参数名	参数位置	参数类型	是否必须	描述
id	查询参数	字符串	是	用户编号

状态码	类型	描述
true	删除成功	
false	删除失败	
示例		

```
GET /user/delete?id=1
```

4.5 开发持久层

4.5.1 生成mapper

- mapper/AdminMapper.java

```
public interface AdminMapper extends BaseMapper<Admin> {  
}
```

- 映射文件 AdminMapper.xml mapper文件夹下

```
<!DOCTYPE mapper  
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.example.online.music.mapper.AdminMapper">  
    <resultMap id="BaseResultMap"  
        type="com.example.online.music.model.domain.Admin">  
        <id column="id" jdbcType="INTEGER" property="id" />  
        <result column="name" jdbcType="VARCHAR" property="name" />  
        <result column="password" jdbcType="VARCHAR" property="password" />  
    </resultMap>  
    <sql id="Base_Column_List">  
        id, name, password  
    </sql>  
</mapper>
```

- ConsumerMapper.java

```
public interface ConsumerMapper extends BaseMapper<Consumer> {  
}
```

映射文件 ConsumerMapper.xml mapper文件夹下

```
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.example.online.music.mapper.ConsumerMapper">  
    <resultMap id="BaseResultMap"  
        type="com.example.online.music.model.domain.Consumer">  
        <result column="id" property="id" jdbcType="INTEGER" />  
        <result column="username" property="username" jdbcType="VARCHAR" />  
        <result column="password" property="password" jdbcType="VARCHAR" />  
        <result column="sex" property="sex" jdbcType="TINYINT" />  
        <result column="phone_num" property="phoneNum" jdbcType="CHAR" />  
        <result column="email" property="email" jdbcType="CHAR" />  
        <result column="birth" property="birth" jdbcType="TIMESTAMP" />  
        <result column="introduction" property="introduction" jdbcType="VARCHAR" />  
        <result column="location" property="location" jdbcType="VARCHAR" />  
        <result column="avator" property="avator" jdbcType="VARCHAR" />  
        <result column="create_time" property="createTime" jdbcType="TIMESTAMP" />  
        <result column="update_time" property="updateTime" jdbcType="TIMESTAMP" />  
    </resultMap>
```

```

    <sql id="Base_Column_List">
        id, username, password, sex, phone_num, email, birth, introduction,
        location, avator, create_time, update_time
    </sql>
</mapper>

```

- CollectMapper

```

public interface CollectMapper extends BaseMapper<Collect> {
}

```

映射文件 CollectMapper.xml mapper文件夹下

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.online.music.mapper.CollectMapper">
    <resultMap id="BaseResultMap"
        type="com.example.online.music.model.domain.Collect">
        <id column="id" property="id" jdbcType="INTEGER" />
        <result column="user_id" property="userId" jdbcType="INTEGER" />
        <result column="type" property="type" jdbcType="TINYINT" />
        <result column="song_id" property="songId" jdbcType="INTEGER" />
        <result column="song_list_id" property="songListId" jdbcType="INTEGER"
        />
        <result column="create_time" property="createTime" jdbcType="TIMESTAMP"
        />
    </resultMap>
    <sql id="Base_Column_List">
        id, user_id, `type`, song_id, song_list_id, create_time
    </sql>
</mapper>

```

添加数据库配置 application.properties

```

spring.datasource.url=jdbc:mysql://localhost:3306/OnlineMusic?
allowPublicKeyRetrieval=true&serverTimezone=Asia/Shanghai&useSSL=false
spring.datasource.username=root
spring.datasource.password=123456
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

```

4.5.2 测试mapper

- AdminMapperTest.java

```

@SpringBootTest
public class AdminMapperTest {
    @Autowired
    private AdminMapper adminMapper;

    /**
     * 测试登录使用的方法
     */
}

```

```

@Test
public void testSelectCount(){
    QueryWrapper<Admin> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("name", "admin");
    queryWrapper.eq("password", "123");
    Long num = adminMapper.selectCount(queryWrapper);
}
}

```

- ConsumerMapperTest.java

```

@SpringBootTest
public class ConsumerMapperTest {
    @Autowired
    private ConsumerMapper consumerMapper;
    @Test
    public void testDeleteByIds(){
        String ids = "26";
        consumerMapper.deleteByIds(ids);
    }
}

```

- CollectMapperTest.java

```

@SpringBootTest
public class TestCollectMapper {
    @Autowired
    private CollectMapper collectMapper;
    @Test
    public void testSelectList(){
        QueryWrapper<Collect> queryWrapper = new QueryWrapper();
        queryWrapper.eq("user_id", 1);
        collectMapper.selectList(queryWrapper);
    }
}

```

4.6 开发业务层

4.6.1 编写Service

- service/AdminService.java

```

public interface AdminService extends IService<Admin> {
    R verifyPasswd(AdminRequest adminRequest, HttpSession session);
}

```

- service/ConsumerService.java

```

public interface ConsumerService extends IService<Consumer> {
    R allUser();
    R deleteUser(Integer id);
}

```

- service/CollectService.java

```
public interface CollectService extends IService<Collect> {
    R collectionOfUser(Integer userId);
}
```

- service/impl/AdminServiceImpl.java

```
@Service
public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin>
    implements AdminService{
    @Autowired
    private AdminMapper adminMapper;
    @Override
    public R verityPasswd(AdminRequest adminRequest, HttpSession session) {
        QueryWrapper<Admin> queryWrapper = new QueryWrapper<>();
        queryWrapper.eq("name",adminRequest.getUsername());
        queryWrapper.eq("password",adminRequest.getPassword());
        if (adminMapper.selectCount(queryWrapper) > 0) {
            session.setAttribute("name", adminRequest.getUsername());
            return R.success("登录成功");
        } else {
            return R.error("用户名或密码错误");
        }
    }
}
```

- service/impl/ConsumerServiceImpl.java

```
@Service
public class ConsumerServiceImpl extends ServiceImpl<ConsumerMapper,
Consumer>
    implements ConsumerService{
    @Autowired
    private ConsumerMapper consumerMapper;
    @Override
    public R allUser() {
        return R.success("查询成功",consumerMapper.selectList(null));
    }

    /**
     * 删除用户
     */
    @Override
    public R deleteUser(Integer id) {
        if (consumerMapper.deleteById(id) > 0) {
            return R.success("删除成功");
        } else {
            return R.error("删除失败");
        }
    }
}
```

- service/impl/CollectServiceImpl.java

```

@Service
public class CollectServiceImpl extends ServiceImpl<CollectMapper, Collect>
implements CollectService {
    @Autowired
    private CollectMapper collectMapper;
    @Override
    public R collectionOfUser(Integer userId) {
        QueryWrapper<Collect> queryWrapper = new QueryWrapper();
        queryWrapper.eq("user_id",userId);
        return R.success("用户收藏", collectMapper.selectList(queryWrapper));
    }
}

```

- 添加辅助文件 config/WebPicConfig.java

```

@Configuration
public class WebPicConfig implements WebMvcConfigurer {

    //这个配置类的目的 就是注册了一个类似于拦截器吧 看到对应的资源 会将其修改成相应的地址
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/img/avatorImages/**")
            .addResourceLocations(Constants.AVATOR_IMAGES_PATH);
        registry.addResourceHandler("/img/singerPic/**")
            .addResourceLocations(Constants.SINGER_PIC_PATH);
        registry.addResourceHandler("/img/songPic/**")
            .addResourceLocations(Constants.SONG_PIC_PATH);
        registry.addResourceHandler("/song/**")
            .addResourceLocations(Constants.SONG_PATH);
        registry.addResourceHandler("/img/songListPic/**")
            .addResourceLocations(Constants.SONGLIST_PIC_PATH);
        registry.addResourceHandler("/img/swiper/**")
            .addResourceLocations(Constants.BANNER_PIC_PATH);
    }
}

```

- constant/Constants.java

```

public class Constants {
    /* 歌曲图片，歌手图片，歌曲文件，歌单图片等文件的存放路径 */
    public static String ASSETS_PATH = System.getProperty("user.dir");

    public static String AVATOR_IMAGES_PATH = "file:" + ASSETS_PATH +
"/img/avatorImages/";
    public static String SONGLIST_PIC_PATH = "file:" + ASSETS_PATH +
"/img/songListPic/";
    public static String SONG_PIC_PATH = "file:" + ASSETS_PATH +
"/img/songPic/";
    public static String SONG_PATH = "file:" + ASSETS_PATH + "/song/";
    public static String SINGER_PIC_PATH = "file:" + ASSETS_PATH +
"/img/singerPic/";
}

```

```

        public static String BANNER_PIC_PATH = "file:" + ASSETS_PATH +
        "/img/swiper/";

        /* 盐值加密 */
        public static String SALT = "xsdfsdfujiwjerwek1231yz";
    }

```

4.6.2 测试service

```

@SpringBootTest
public class AdminServiceTest {
    @Autowired
    private AdminService adminService;
    private static MockHttpSession sessionPub;

    @BeforeAll
    public static void setupMockMvc() {
        sessionPub = new MockHttpSession();
        sessionPub.setAttribute("username", "admin");
    }

    /**
     * 测试VerityPasswd()
     */
    @Test
    public void testVerityPasswd() {
        AdminRequest ar = new AdminRequest();
        ar.setUsername("admin");
        ar.setPassword("123");
        adminService.verityPasswd(ar, sessionPub);
    }
}

```

```

@SpringBootTest
public class ConsumersServiceTest {
    @Autowired
    private ConsumerService consumerService;

    @Test
    public void testAllUser(){
        consumerService.allUser();
    }
    @Test
    public void testDeleteUser(){
        consumerService.deleteUser(62);
    }

    @Test
    public void testDeleteUsers(){
        consumerService.deleteUsers(new String[]{"61", "62"});
    }
}

```

```

@SpringBootTest
public class CollectServiceTest {
    @Autowired
    private CollectService collectService;
    @Test
    public void testCollectionOfUser(){
        collectService.collectionOfUser(1);
    }
}

```

4.7 接口测试

```

@SpringBootTest
public class AdminControllerTest {
    @Autowired
    private AdminController adminController;

    private static MockHttpSession sessionPub;

    @BeforeAll
    public static void setupMockMvc() {
        sessionPub = new MockHttpSession();
    }
    @Test
    public void testLoginStatus(){
        AdminRequest ar = new AdminRequest();
        ar.setUsername("admin");
        ar.setPassword("123");
        adminController.loginStatus(ar,sessionPub);
    }
}

```

```

@SpringBootTest
@Rollback
public class ConsumerControllerTest {
    @Autowired
    private ConsumerController consumerController;

    @Test
    public void testAllUser(){
        consumerController.allUser();
    }
    @Test
    public void testDeleteUser(){
        consumerController.deleteUser(62);
    }
    @Test
    public void testDeleteUsers(){
        consumerController.deleteUsers(new String[]{"61","59"});
    }
}

```



```

@SpringBootTest
public class CollectControllerTest {
    @Autowired
    private CollectController collectController;

    @Test
    public void testCollectionOfUser(){
        collectController.collectionOfUser(1);
    }
}

```

4.8 前后端联调

4.8.1 管理端onlinemusic-admin 登录实现

1. 封装访问后台系统方法 src/api/index.ts 方便统一管理和维护，方便调用，需要参考后台的API文档

```

import { deletes, get, getBaseURL, post } from './request'
const HttpManager = {
    // 获取图片信息
    attachImageUrl: (url) => `${getBaseURL()}/${url}`,
    // =====> 管理员 API 完成
    // 是否登录成功
    // 是否登录成功
    getLoginStatus: ({ username, password }) => post(`admin/login/status`, {
        username, password }),

    // =====> 用户 API 完成
    // 返回所有用户
    getAllUser: () => get(`user`),
    // 返回指定ID的用户
    getUserOfId: (id) => get(`user/detail?id=${id}`),
    // 删除用户
    deleteUser: (id) => get(`user/delete?id=${id}`),

    // =====> 收藏列表 API 完成
    // 返回的指定用户ID收藏列表
    getCollectionOfUser: (userId) => get(`collection/detail?
        userId=${userId}`),
    // 删除收藏的歌曲
    deleteCollection: (userId, songId) => deletes(`collection/delete?
        userId=${userId}&&songId=${songId}`),

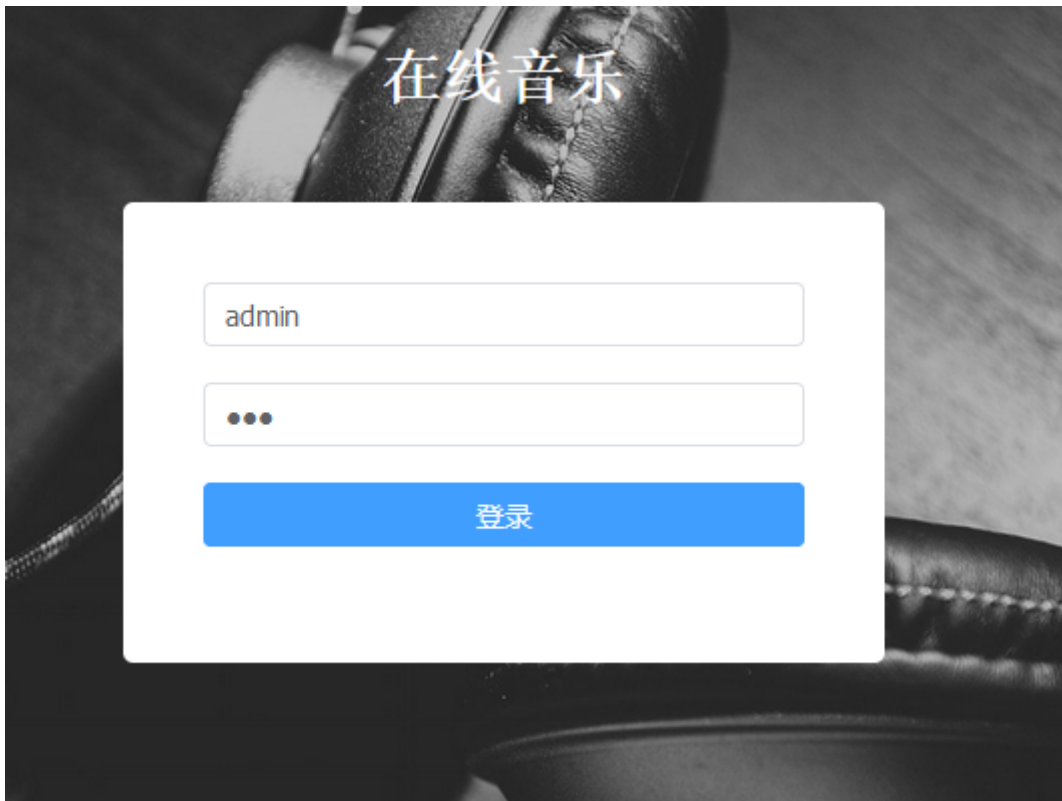
    // =====> 评论列表 API 完成
    // 获得指定歌曲ID的评论列表
    getCommentOfSongId: (songId) => get(`comment/song/detail?
        songId=${songId}`),
    // 获得指定歌单ID的评论列表
    getCommentOfSongListId: (songListId) => get(`comment/songList/detail?
        songListId=${songListId}`),
    // 删除评论
    deleteComment: (id) => get(`comment/delete?id=${id}`),
}

```

```
export { HttpManager }
```

2. 添加登录页面views/Login.vue及相关页面src/mixins/mixin.ts和enums文件夹中所有Login.vue中部分逻辑代码修改，暂时不要token代码

```
<script>
export default defineComponent({
  setup() {
    .....
    /*
    console.log(localStorage.getItem('token'))
    //从请求头中获取authorization的值即token
    const token = result.data.authorization;
    localStorage.setItem('token',token);
    */
    .....
  }
})
</script>
```



3. 添加登录成功后跳转相关页面

components/layouts/YinHeader.vue

components/layouts/YinAudio.vue

components/layouts/YinAside.vue

views\Home.vue

views/InfoPage.vue

备注：

YinAside.vue中添加：import {PieChart,User,Mic,Document} from '@element-plus/icons-vue';

InfoPage.vue中暂时注释下面代码：

```

<script>
.....
// const userSexChart = echarts.init(proxy.$refs.userSex);
const userSexChart = echarts.init(document.getElementById("userSex"));
userSexChart.setOption(userSex);
});
/*
HttpManager.getAllSong().then((res) => {
    songCount.value = (res as ResponseBody).data.length;
});
HttpManager.getSongList().then((res) => {
    const result = res as ResponseBody;
    songListCount.value = result.data.length;
    for (let item of result.data) {
        for (let i = 0; i < songStyle.xAxis.data.length; i++) {
            if (item.style.includes(songStyle.xAxis.data[i])) {
                songStyle.series[0].data[i]++;
            }
        }
    }
}
// const songStyleChart = echarts.init(proxy.$refs.songStyle);
const songStyleChart = echarts.init(document.getElementById("songStyle"));
songStyleChart.setOption(songStyle);
});

HttpManager.getAllSinger().then((res) => {
    const result = res as ResponseBody;
    singerCount.value = result.data.length;
    singerSex.series[0].data.push(setSex(0, result.data));
    singerSex.series[0].data.push(setSex(1, result.data));
    const singerSexChart = echarts.init(document.getElementById("singerSex"));
    singerSexChart.setOption(singerSex);

    for (let item of result.data) {
        for (let i = 0; i < country.xAxis.data.length; i++) {
            if (item.location.includes(country.xAxis.data[i])) {
                country.series[0].data[i]++;
            }
        }
    }
}
const countryChart = echarts.init(document.getElementById("country"));
countryChart.setOption(country);
});*/
</script>

```

4. 添加用户管理相关页面

utils\index.ts

components/dialog/YinDeIDialog.vue

views\ConsumerPage.vue

系统首页

用户管理

歌手管理

歌单管理

批量删除

筛选用户

<input type="checkbox"/>	ID	用户头像	用户名	性别	手机号码	邮箱	生日	签名	地区	收藏	操作
<input type="checkbox"/>	1		Yin33	女	13776412237	yocna@qq.com	2019-05-21	好好吃饭	山西	<div>收藏</div>	<div>删除</div>
<input type="checkbox"/>	2		012	女	13754803255	love@gmail.com	2019-04-24	我就喜欢吃	北京	<div>收藏</div>	<div>删除</div>
<input type="checkbox"/>	5		789	女	13634377258	666@126.com	2019-01-08	今天很开心啊	山西	<div>收藏</div>	<div>删除</div>
<input type="checkbox"/>	8		taruhen	女		192673541@qq.com	2019-04-25	你好	北京	<div>收藏</div>	<div>删除</div>

5. 添加收藏功能相关页面

views/CollectPage.vue 注意需要暂时注释下面代码，歌曲模块完成后再测试使用

```
// 通过用户 ID 获取用户收藏的歌曲 ID
async function getData() {
  tableData.value = [];
  tempDate.value = [];
  /*
  const result = (await
  HttpManager.getCollectionOfUser(proxy.$route.query.id)) as any;
  for (let item of result.data) {
    const result = await HttpManager.getSongOfId(item.songId) as any;
    tableData.value.push(result.data[0]);
    tempDate.value.push(result.data[0]);
  }*/
}
```

6. 更新路由 router/index.ts

```
const routes: Array<RouteRecordRaw> = [
  {
    path: '/',
    name: 'Login',
    component: Login
  },
  {
    path: '/Home',
    component: () => import('@views/Home.vue'),
    meta: { title: '自述文件' },
    children: [
      {
        path: '/Info',
        component: () => import('@views/InfoPage.vue'),
        meta: { title: 'Info' }
      },
      {
        path: '/Consumer',
        component: () => import('@views/ConsumerPage.vue'),
        meta: { title: 'Consumer' }
      },
      {
        path: '/Collect',
        component: () => import('@views/CollectPage.vue'),
```

```
    meta: { title: 'collect' }  
  },  
]  
,  
]
```