音乐吧 online 网站概要设计

V5.0

三峡大学 数字媒体技术专业

评审日期： 2019 年 11 月 24 日

目录

1. 导言

1.1 编写目的

该文档根据音乐吧 Oline 网站的功能和性能，阐述了音乐吧 Oline 网站的概要设计，包括框架设计，功能模块设计，数据库设计，界面设计等部分。

本文档的预期读者包括：

- 设计开发人员
- 项目管理人员
- 测试人员
- 用户

1.2 范围

该文档的目的是解决整个项目系统怎么“怎么做的问题”。在这里，主要是根据用户提出的需求进行全面设计。

1.3 引用标准

[1]《SPM 课程网站概要设计》文档　北京邮电大学

1.4 参考资料

[1]　软件工程：实践者的研究方法（原书第八版·本科教学版）/（美）罗杰 S.普莱斯曼(Roger S.Pressman)，(美)布鲁斯 R.马克西姆(Bruce R.Maxim)著；郑仁杰等译.—北京：机械工业出版社；

[2]　UML2 面向对象的分析与设计/谭火彬著.—北京：清华大学出版社；

[3]　人月神话（40 周年中文纪念表）/(美)布鲁克斯（Brooks,F.P.）著；UML China 翻译组，汪颖译.—北京：清华大学出版社。

1.5 版本更新信息

本文档更新记录如表 1-5 1 版本更新表所示：

修改编号	修改日期	修改后版本	修改位置	修改内容概述
001	2019.11.3	1.0	全部	初始发布版本
002	2019.11.16	2.0	全部	修改、增加
003	2019.11.17	3.0	全部	修改
004	2019.11.19	4.0	局部	修改
005		5.0	局部	修改

1-5 1 版本更新表

2. 项目设计原则简介

音乐吧 Oline 应用程序是根据热爱听音乐，热爱创作音乐的用户的需求进行设计的音乐分享交流平台。该平台具有功能的多样性，不局限于听歌，歌曲下载，还融合了线上交友功能，可以互相关注，进行歌曲评论。也给独立创作的音乐人提供一个上传作品的平台，建立爱听音乐的人与爱创作音乐的人之间的联系。为网民提供一个轻松快捷的线上音乐分享交流平台。

在整个系统设计的过程中遵循以下设计原则：

1.实用性：实用性是系统的主要设计原则，系统设计必须最大可能地满足用户的需求，做到操作方便、界面友好、可即时更新，能适应不同层次用户的需求。

2.先进性：信息技术发展迅速，系统设计尽可能采用先进的技术标准和技术方法。

3.以用户为中心的处理：个性化服务充分体现了这一点，根据用户当前偏向爱好，配置页面功能布局及展现内容，贴合用户操作。

4.使用便捷。系统要有设计良好的人机交互界面，即使系统的操作界面简单易用，又能具有较强的适用性，满足不同计算机使用水平的用户使用。

5.灵活和易维护：采用开放的体系架构，基于开放源代码的技术框架和数据库系统，使用高效率的开源和免费开发工具，具备完整的文档说明。在维护方面，主要考虑两个层面，一是对于开发人员来讲，系统编码容易调整，可适应需求的

变化和调整；二是对于系统管理维护人员来说，能够对系统进行便捷的维护和管理。

6.安全可靠：选择安全可靠的软硬件运行平台，并在系统设计和实现的时候关注系统的安全控制和执行效率，提供相应的安全防护功能，保证系统具有较高的安全性和可靠性。安全性方面，要考虑系统的安全、数据管理的安全、网络安全。保证用户权限、数据安全和系统的稳定性。

7.单一职责原则：我们系统在面向对象设计部分采取单一职责原则，其核心思想为：一个类，最好只做一件事，只有一个引起它的变化。单一职责原则可以看做是低耦合、高内聚在面向对象原则上的引申，将职责定义为引起变化的原因，以提高内聚性来减少引起变化的原因。从而最终提高我们系统的可修改性和可维护性。

3. 结构体系设计

音乐吧 online 系统本着软件开发的设计原则，采用浏览器/服务器（B/S）的体系结构。为了满足网站后台的增删改查和平台用户创建信息与信息收发的基本操作，在软件架构上，采用五层体系结构：控制层，表现层，业务逻辑层，数据持久层，域模型层；在设计实现上，我们采用 MVC 的设计模式：Model 模型层、View 视图层、Controller 控制层；在体系架构上，音乐吧 online 系统选择用 Vue 架构。

Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。简单来说，Aue 是为了实现前后端分离的开发理念，开发前端 SPA（single page web application）项目，实现数据绑定，路由配置，项目编译打包等一系列工作的技术框架。Vue.js 不支持 IE8 及其以下版本，因为 Vue.js 使用了 IE8 不能模拟的 ECMAScript 5 特性。 Vue.js 支持所有兼容 ECMAScript 5 的浏览器。

Vue 框架关系如图 3-1 1 Vue 框架关系示意图所示：

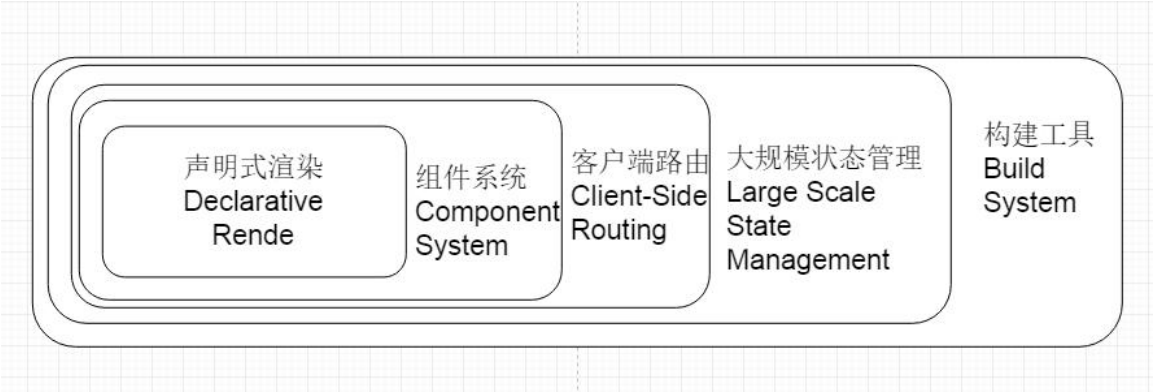

图 3-1 1 Vue 框架关系示意图

如果要实现一个功能，具体流程如下：

1.安装相关软件，配置相关运行环境，创建新项目。

2.安装相关依赖包,安装之后,查看浏览器 localhost:8080,是否有 welcome to You Vue.js App 字样，有就代表环境配置以及项目创建成功了。

3. Vue 是单页面应用，一个项目只能创建一个 new Vue，Vue 能够自动刷新。vue 项目创建成功之后，测试 npm run dev ，查看 localhost 8080 能否查看，测试成功之后，用 pycharm 打开项目目录。各个目录详细如图 3-1 2 项目目录详细说明所示：

目录/文件	说明
build	最终发布的代码存放位置。
config	配置目录，包括端口号等。我们初学可以使用默认的。
node_modules	npm 加载的项目依赖模块
src	这里是我们要开发的目录，基本上要做的事情都在这个目录里。里面包含了几个目录及文件： • assets: 放置一些图片，如logo等。 • components: 目录里面放了一个组件文件，可以不用。 • App.vue: 项目入口文件，我们也可以直接将组件写这里，而不使用 components 目录。 • main.js: 项目的核心文件。
static	静态资源目录，如图片、字体等。
test	初始测试目录，可删除
.xxxx文件	这些是一些配置文件，包括语法配置，git配置等。
index.html	首页入口文件，你可以添加一些 meta 信息或同统计代码啥的。
package.json	项目配置文件。
README.md	项目的说明文档，markdown 格式

图 3-1 2 项目目录详细说明

4. 项目目录内，node_modules 目录一般是放依赖包，安装的依赖包去这里查看是否安装成功。src 一般放项目需要的程序以及组件等等。Vue 项目的一般逻辑是:用户直接访问的是 index.html,index.html 下面是 App.vue 和 main.js 通过路由的方式(index.js) 连接组件 components ，目录中的组件，渲染具体内容。所以编写思路就是：1 创建组件（Vue 文件）；2 配置路由；3 编写路径

（router-link）。然后是代码：在 src 目录下的 components 目录下创建组件：duanzi.vue

5.写 Service.java（ServiceImpl.java），为控制层提供服务，接受控制层的参数，完成相应的功能，并返回给控制层,注意这里用到注解@Service 以及@Autowired 进行 service 层定义以及 DAO 层类注入。

6. 在创建另一个组件 HelloWorld.vue，代码道理和 duanzi.vue 一样，只是输出内容不一样，实现两个页面。

7. 在 App.vue 中配置具体渲染位置。写完代码，我们可以通过 npm start（npm run dev 也可以)在 cmd 中启动 Vue 项目,然后通过 http://localhost:8080 访问内容。

整体来看，在 Aue 架构里，核心特点主要是响应的数据绑定、可组合的视图组件、虚拟 DOM、MVVM 模式、声明式渲染。分层结构图如图 3-1 3Vue 生命周期示意图所示：

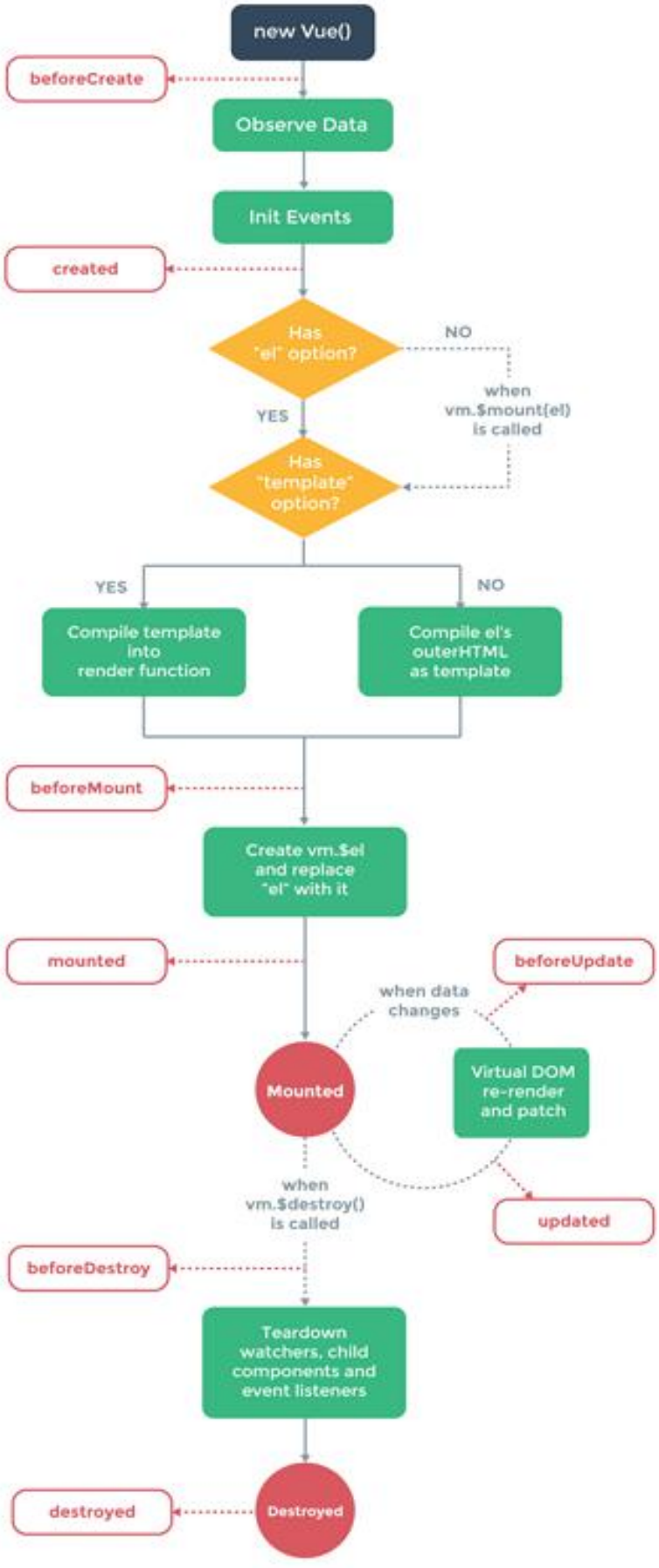

图 3-1 3Vue 生命周期示意图

3.1 vue.js 的核心特点-响应的数据绑定

传统的 js 操作页面：在以前使用 js 操作页面的时候是这样的，需要操作某个 html 元素的数据，就的使用 js 代码获取元素然后在处理业务逻辑。Vue 数据双向绑定是通过数据劫持结合发布者-订阅者模式的方式来实现的。首先要对数据进行劫持监听，所以我们需要设置一个监听器 Observer，用来监听所有属性。如果属性发上变化了，就需要告诉订阅者 Watcher 看是否需要更新。因为订阅者是有很多个，所以我们需要有一个消息订阅器 Dep 来专门收集这些订阅者，然后在监听器 Observer 和订阅者 Watcher 之间进行统一管理的。我们还需要有一个指令解析器 Compile，对每个节点元素进行扫描和解析，将相关指令（如 v-model，v-on）对应初始化成一个订阅者 Watcher，并替换模板数据或者绑定相应的函数，此时当订阅者 Watcher 接收到相应属性的变化，就会执行对应的更新 函数，从而更新视图。如图 3-1 4 传统的 js 操作页面数据绑定示意图所示：

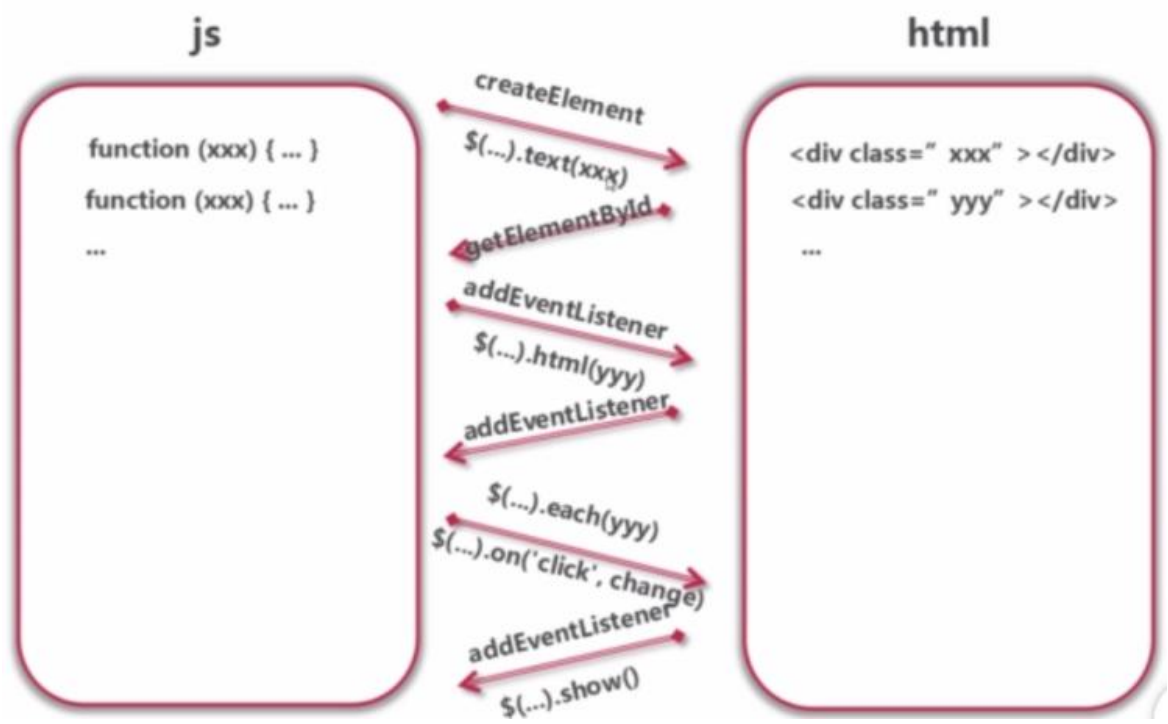

图 3-1 4 传统的 js 操作页面数据绑定示意图

响应式数据绑定的方式操作页面，可以直接使用像下面代码那样的写法就可以将数据填充到页面中。如图图 3-1 5 新版 Vue 架构响应式数据绑定流程图　3-1 6 响应式原理解析图 图 3-1 7 js 代码示意图所示：

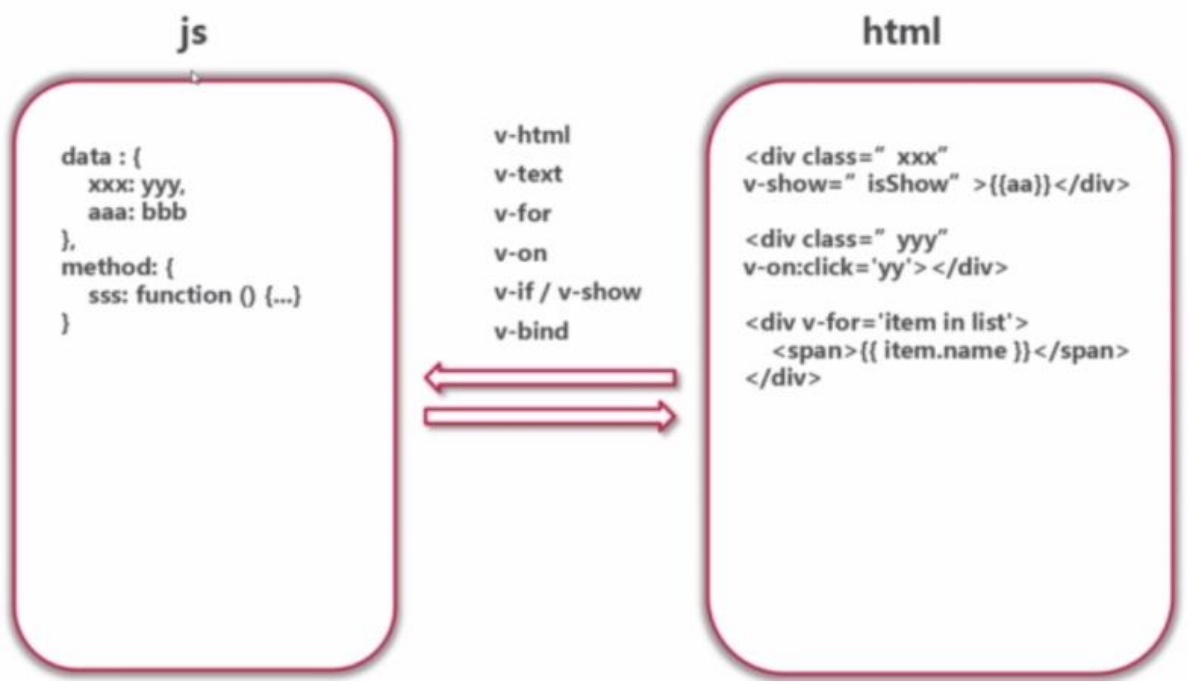

图 3-1 5 新版 Vue 架构响应式数据绑定流程

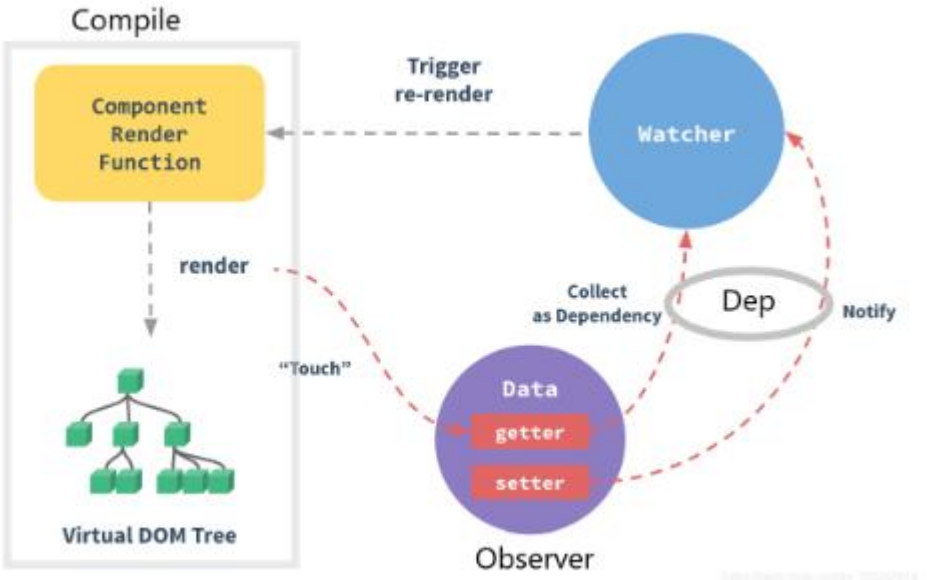

图 3-1 6 响应式原理解析图

```
<template>
  <div id="app">
    {{ message }}
  </div>
</template>

<script>
export default {
  name: 'app',
  data () {
    return {
      message: 'Welcome to Your Vue.js App'
    }
  }
}
</script>

<style>
</style>
```

图 3-1 7　js 代码示意图

3.2 vue.js 的核心特点-可组合的视图组件

每用一次组件，就会有一个它的新实例被创建。一个组件的 data 选项必须是一个函数。可以调用内建的$emit 方法，并传入事件的名字，来向父级组件触发一个事件。全局注册和局部注册，全局注册的弊端在于：如果你使用 webpack 这样的构建系统，全局注册意味着即便你不再使用一个组件，它仍然会被包含在你最终的构建结果中。这造成了用户下载的 JavaScript 的无谓的增加。全局注册的组件可以用在其被注册之后的任何（通过 new Vue）新创建的 Vue 根实例，也包括其组件树中的所有子组件的模板中。如图 3-2 1 组件关系图　图 3-2 2 父子组件示意图 所示：

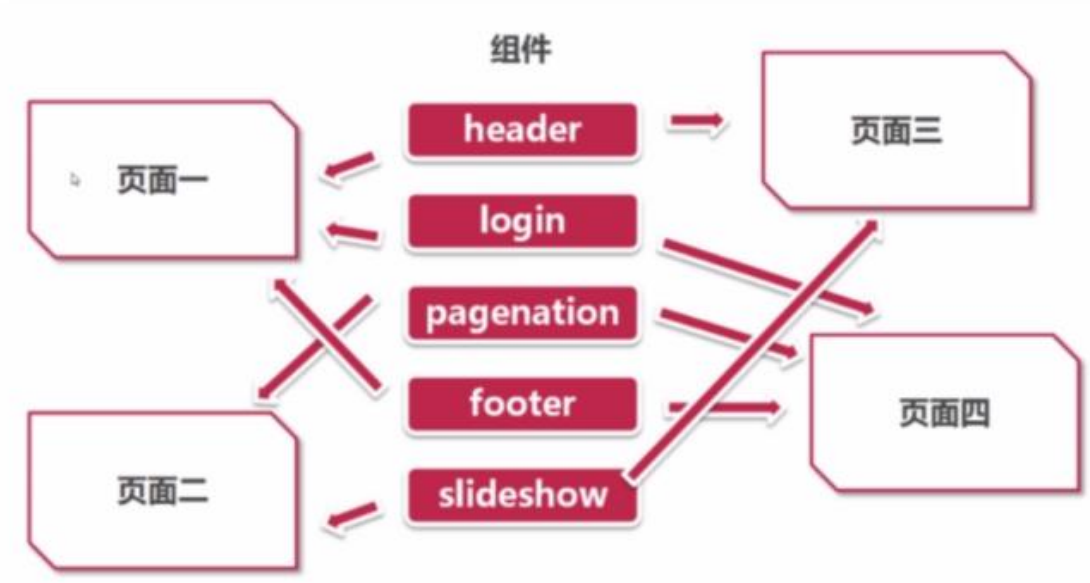

图 3-2 1 组件关系图

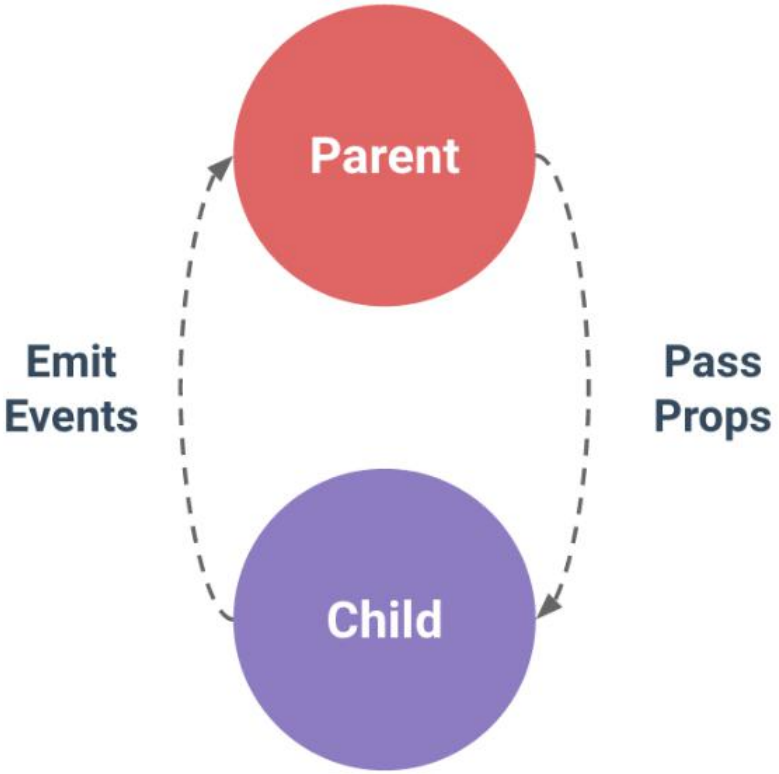

图 3-2 2 父子组件示意图

组件 A 在它的模板中使用了组件 B，它们之间必然需要相互通信，父组件可能要给子组件下发数据，子组件则可能要将它内部发生的事情告知父组件，父子组件通信主要是使用 prop 和自定义事件，父组件通过 prop 给子组件下发数据，子组件通过事件给父组件发送消息。

组件+构建工具==> 单文件组件概念

在同一个 Vue 文件里，可以同时写 template, script 和 style，三个东西放在一个里面。这样的好处是有了一个构建的机会，可以对这些单文件组件做更多的分析处，在每一个语言块里可以单独使用不同的处理器

3.3 vue.js 的核心特点-虚拟 DOM

运行的 js 速度是很快的，大量的操作 DOM 就会很慢，时常在更新数据后会重新渲染页面，这样造成在没有改变数据的地方也重新渲染了 DOM 节点，这样就造成了很大程度上的资源浪费。

利用在内存中生成与真实 DOM 与之对应的数据结构，这个在内存中生成的结构称之为虚拟 DOM，当数据发生变化时，能够智能地计算出重新渲染组件的最小代价并应用到 DOM 操作上。构建 DOM 数是一个渐进过程，为达到更好用户体验，

渲染引擎会尽快将内容显示在屏幕上。它不必等到整个 HTML 文档解析完毕之后才开始构建 render 数和布局。

vdom 的真正意义是为了实现跨平台，服务端渲染，以及提供一个性能还算不错 Dom 更新策略。vdom 让整个 mvvm 框架灵活了起来。

Diff 算法只是为了虚拟 DOM 比较替换效率更高，通过 Diff 算法得到 diff 算法结果数据表(需要进行哪些操作记录表)。原本要操作的 DOM 在 vue 这边还是要操作的，只不过用到了 js 的 DOM fragment 来操作 dom（统一计算出所有变化后统一更新一次 DOM）进行浏览器 DOM 一次性更新。如图 3-3 1 虚拟 Dom 原理图 图 3-3 2 虚拟 Dom 流程图所示：

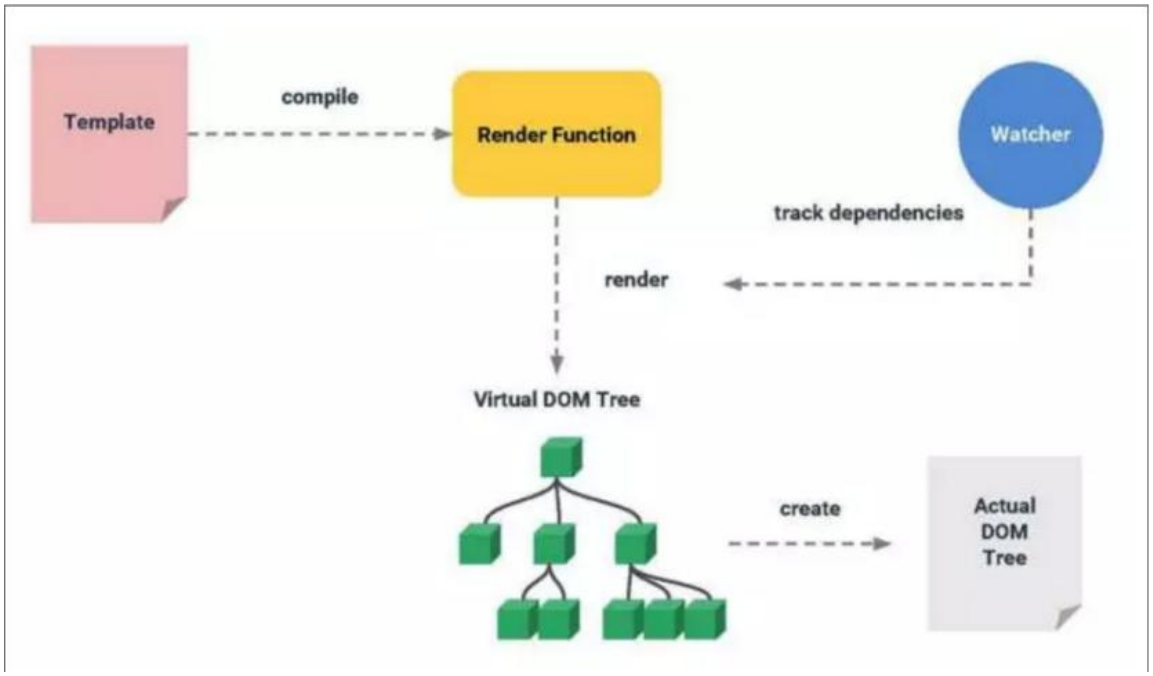

图 3-3 1 虚拟 Dom 原理图

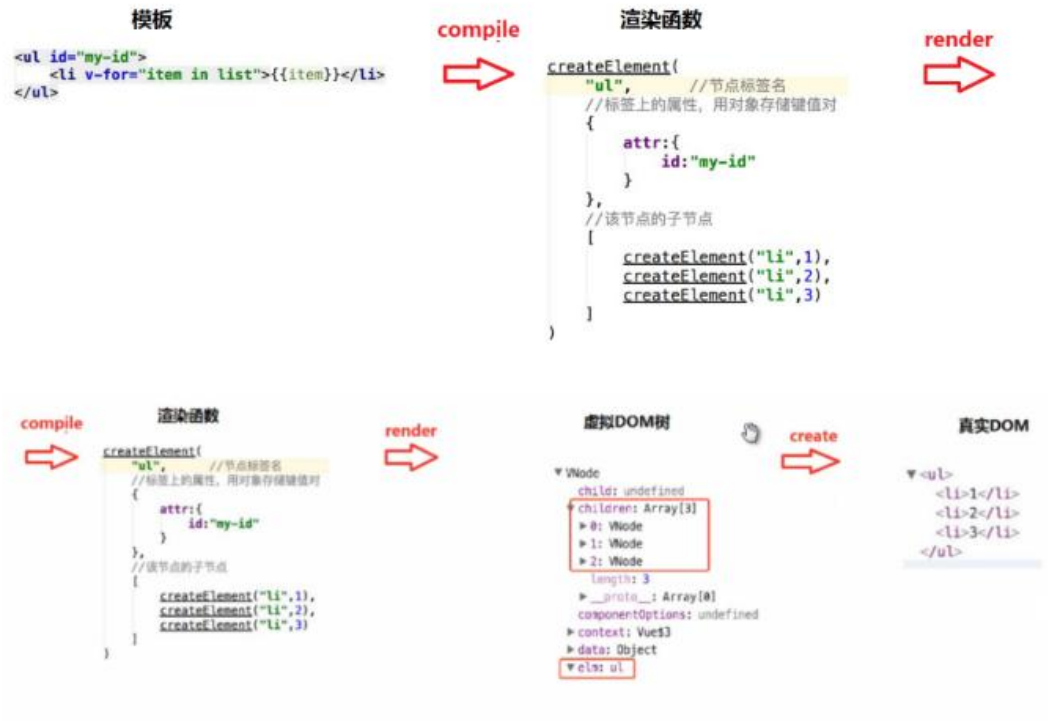

图 3-3 2 虚拟 Dom 流程图

3.4 vue.js 的核心特点-MVVM 模式

MVVM 概述：M：Model 数据模型，V：view 视图模板，vm：view-Model：视图模型，MVVM 将数据双向绑定（data-binding）作为核心思想，View 和 Model 之间没有联系，它们通过 ViewModel 这个桥梁进行交互。Model 和 ViewModel 之间的交互是双向的，因此 View 的变化会自动同步到 Model，而 Model 的变化也会立即反映到 View 上显示。当用户操作 View，ViewModel 感知到变化，然后通知 Model 发生相应改变；反之当 Model 发生改变，ViewModel 也能感知到变化，使 View 作出相应更新。

MVVM 是一种设计思想。Model 层代表数据模型，也可以在 Model 中定义数据修改和操作的业务逻辑；View 代表 UI 组件，它负责将数据模型转化成 UI 展现出来，ViewModel 是一个同步 View 和 Model 的对象。

在 MVVM 架构下，View 和 Model 之间并没有直接的联系，而是通过 ViewModel 进行交互，Model 和 ViewModel 之间的交互是双向的， 因此 View 数据的变化会同步到 Model 中，而 Model 数据的变化也会立即反应到 View 上。

ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而 View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作 DOM，不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。如图 3-4 1MVVM 模式中三者关系 图 3-4 2MVVM 概述所示：

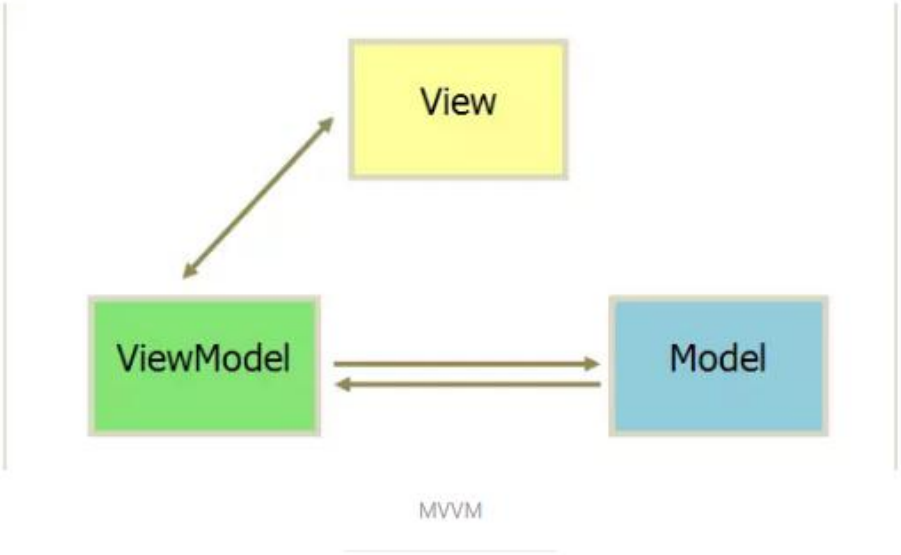

图 3-4 1MVVM 模式中三者关系

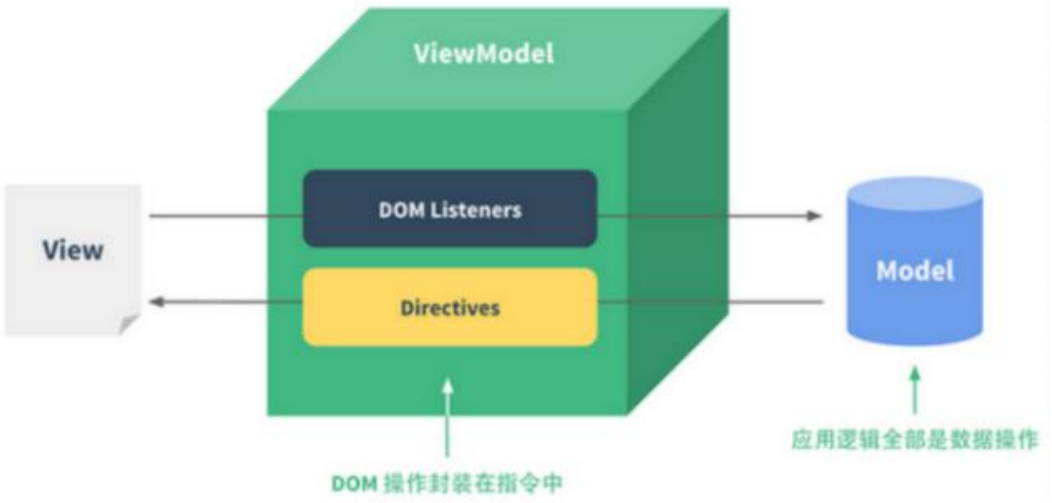

图 3-4 2MVVM 概述

3.5 vue.js 的核心特点-声明式渲染

Vue.js 的核心是一个允许采用简洁的模板语法来声明式的将数据渲染进 DOM，初始化根实例，vue 自动将数据绑定在 DOM 模板上，声明式渲染与命令式渲染区别：声明式渲染：所谓声明式渲染只需要声明在哪里，做什么，而无需关心如何实现；命令式渲染：需要具体代码表达在哪里，做什么，如何实践，需求：求数组中每一项的倍数，放在另一个数组中实例。

$mount 方法就是整个渲染过程的起始点

在渲染过程中，提供了三种渲染模式，自定义 Render 函数、template、el 均可以渲染页面，也就是对应我们使用 Vue 时，三种写法。

核心流程：

1、new Vue，执行初始化

2、挂载$mount 方法，通过自定义 Render 方法、template、el 等生成 Render 函数

3、通过 Watcher 监听数据的变化

4、当数据发生变化时，Render 函数执行生成 VNode 对象

5、通过 patch 方法，对比新旧 VNode 对象，通过 DOM Diff 算法，添加、修改、删除真正的 DOM 元素

至此，整个 new Vue 的渲染过程完毕。

render 流程:

1、创建虚拟 DOM

2、真实 DOM 连接 虚拟 DOM

3、视图更新

4、计算 [新虚拟 DOM] 和 [旧虚拟 DOM] 的差异 (diff)

5、根据计算的 差异，更新真实 DOM (patch)

如图 3-5 1 两种渲染方法代码示例 图 3-5 2 两种渲染方法的输出示例所示：

```
<script>
    // 命令式渲染
    // 关心每步，关心流程，用命令去实现
    var newArr = [];
    var arr = [1,2,3,4,5];
    for(var i = 0; i < arr.length;i++){
        newArr.push(arr[i]*2);
    }
    console.log(newArr);

    // 声明式渲染，不用关心中间流程，只需要关心结果和实现的条件

    var arr1 = arr.map(function(item){
        return item*2
    })
    console.log(arr1)
</script>
```

图 3-5 1 两种渲染方法代码示例

图 3-5 2 两种渲染方法的输出示例

3.6 vue.js 的核心特点-生命周期

每个 Vue 实例在被创建之前都要经过一系列的初始化过程。例如，实例需要：

配置数据观测(data observer)；

编译模版、挂载实例到 DOM

在数据变化时更新 DOM

在这个过程中，实例也会调用一些生命周期钩子 ，这就给我们提供了执行自定义逻辑的机会。从 Vue 实例创建、运行、到销毁期间，会伴随着触发各种各样的事件，这些事件，统称为生命周期。

生命周期钩子 = 生命周期函数 = 生命周期事件　　如图 3-6 1 生命周期示意图所示：

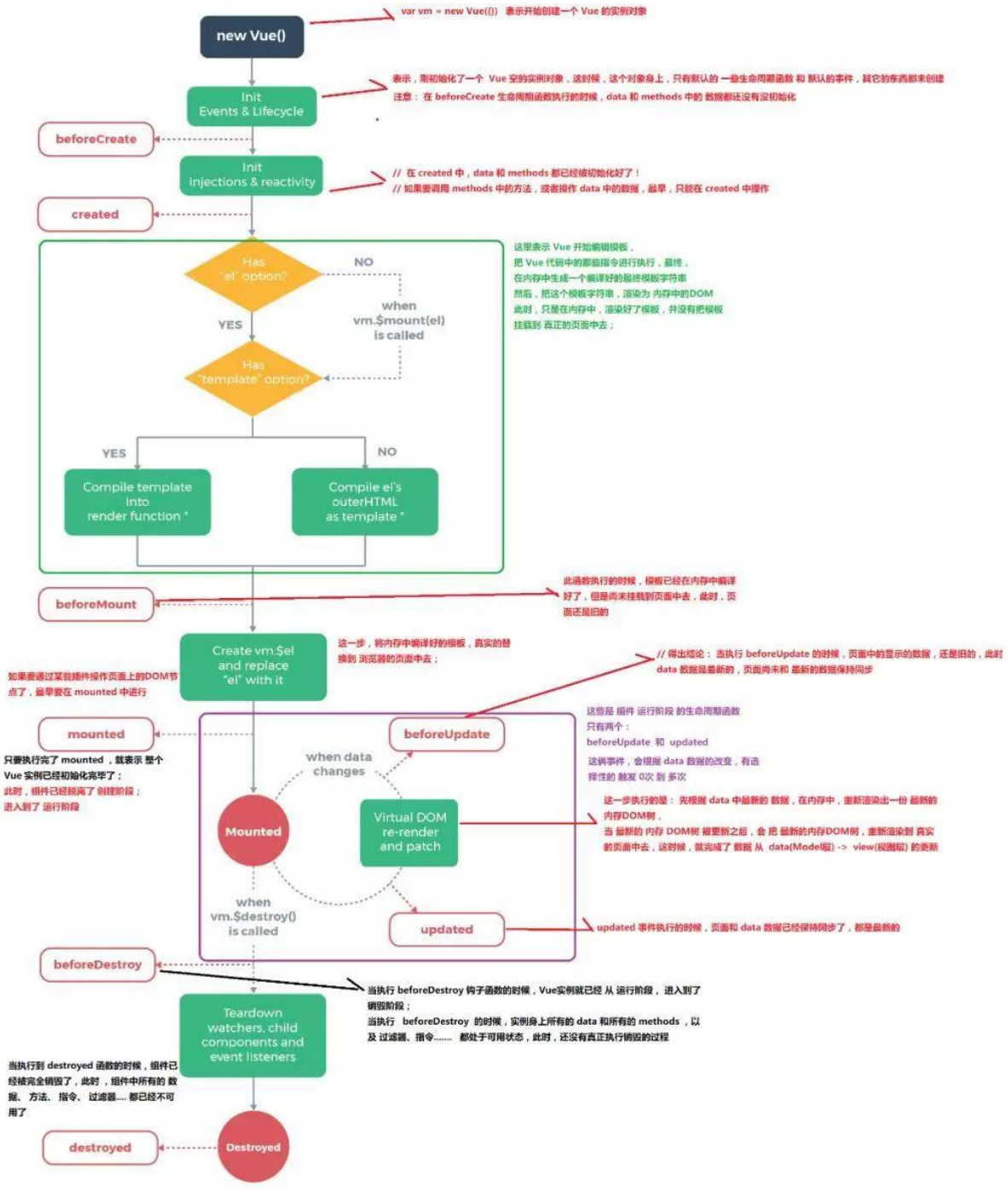

图 3-6 1 生命周期示意图

生命周期中事件划分:

1) 创建期间的生命周期函数

beforeCreate

实例刚在内存中被创建出来，此时，还没有初始化好 data 和 methods 属性;

created

实例已经在内存中创建完毕，此时 data 和 methods 已经创建完毕，此时还没有开始编译模板;

beforeMount

此时已经完成了模板的编译，但是还没有挂载到页面中;

mounted

此时已经将编译好的模板，挂载到了页面指定的容器中显示;

2) 运行期间的生命周期函数

beforeUpdate

状态更新之前执行此函数， 此时 data 中的状态值是最新的，但是界面上显示的 数据还是旧的，因为此时还没有开始重新渲染 DOM 节点;

updated

实例更新完毕之后调用此函数，此时 data 中的状态值 和 界面上显示的数据，都已经完成了更新，界面已经被重新渲染好了!

3) 销毁期间的生命周期函数

beforeDestroy

实例被销毁之前调用，在该函数中，实例仍然完全可用。

destroyed

Vue 实例销毁后调用，调用该函数后，Vue 实例指示的所有东西都会解绑定，所有的事件监听器会被移除，所有的子实例也会被销毁。

4. 后台原理

4.1 跨站请求伪造（CSRF）

跨站请求伪造，是一种挟制用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。跟跨网站脚本（XSS）相比，XSS 利用的是用户对指定网站的信任，CSRF 利用的是网站对用户网页浏览器的信任。

跨站请求攻击，简单地说，是攻击者通过一些技术手段欺骗用户的浏览器去访问一个自己曾经认证过的网站并运行一些操作（如发邮件，发消息，甚至财产操作如转账和购买商品）。由于浏览器曾经认证过，所以被访问的网站会认为是真正的用户操作而去运行。这利用了 web 中用户身份验证的一个漏洞：简单的身份验证只能保证请求发自某个用户的浏览器，却不能保证请求本身是用户自愿发出的。

预防措施有：检查 Referer 字段和添加校验 token

4.2 伪造请求头

在项目开发中，系统需要调其它系统接口，并且该系统需要获取客户端（浏览器访问端）的 IP 地址。正常流程：浏览器---访问 PC 系统----PC 系统需要调第三方系统，此时默认情况下，PC 发起的 request 请求 IP 地址是 PC 所在服务器的 IP 地址，而不是请求浏览器端的 IP 地址

可以在 http 请求头里，追加一个头信息（名称：x-forwarded-for），它会位于原始 IP 地址之前，所以当第三方系统获取地址时，就获取到了真实的浏览器访问地址 IP

4.3 调用官方 API

Nodejs 调用外部 API 需要先安装依赖：npm install request，实现代码如图 4-3 1 安装依赖图所示：

```
var request = require('request');

request('https://XXXXXXXX', function (error, response, body) {

    if (!error && response.statusCode == 200) {

        body = JSON.parse(body);

    }
})
```

图 4-3 1 安装依赖图

5. 功能模块设计

5.1 功能模块设计总述

本系统为用户交互式管理平台，主要包括得模块为两部分：前台基本业务模块和后台管理模块，如图 5-1 1 音乐吧 online 网站主要模块图 图 5-1 2 音乐吧 online 网站前台基本业务模块图 图 5-1 3 音乐吧 online 网站后台管理模块图所示是后台管理模块图：

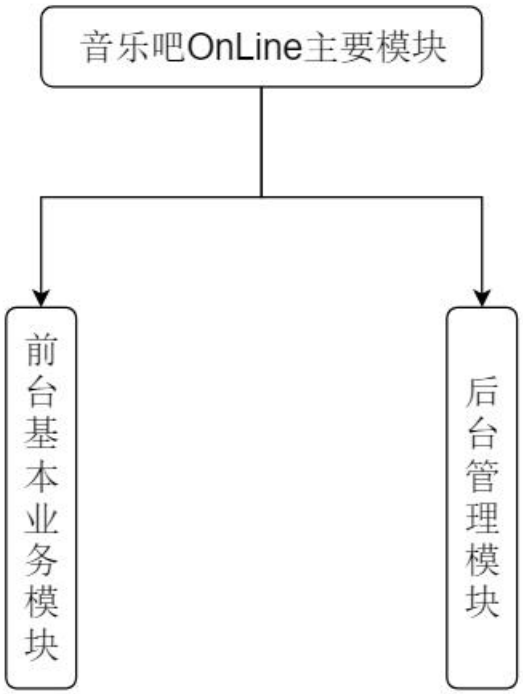

图 5-1 1 音乐吧 online 网站主要模块图

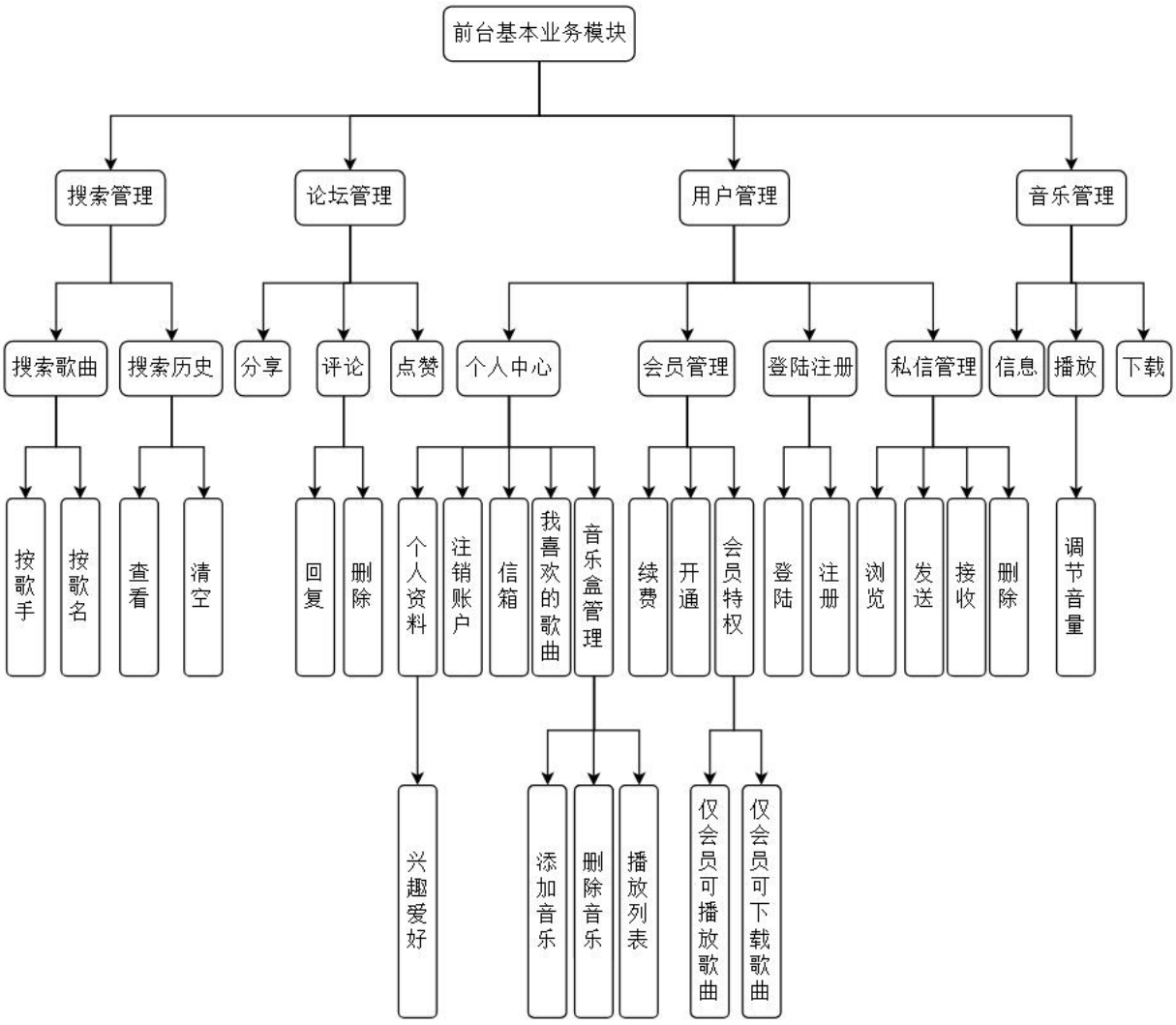

图 5-1 2 音乐吧 online 网站前台基本业务模块图

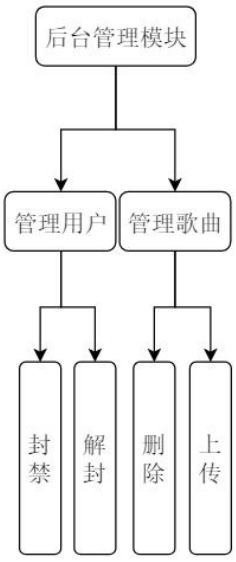

图 5-1 3 音乐吧 online 网站后台管理模块图

5.2 前台基本业务模块设计

5.2.1 模块 CM1：登录系统

编号：CM1；

模块名称：登录系统；

功能简介：本模块为系统登录模块，即用户登录系统的入口。在此模块中，用户输入自己的注册邮箱和密码，系统在后台数据库进行查询操作后，返回布尔值，表示该输入是否正确，输入正确则进入系统，错误则对用户进行相应提示；

输入：注册邮箱，密码；

输出：用户是否登录成功；

操作流程：如图 5-2 1 登录系统操作流程所示。

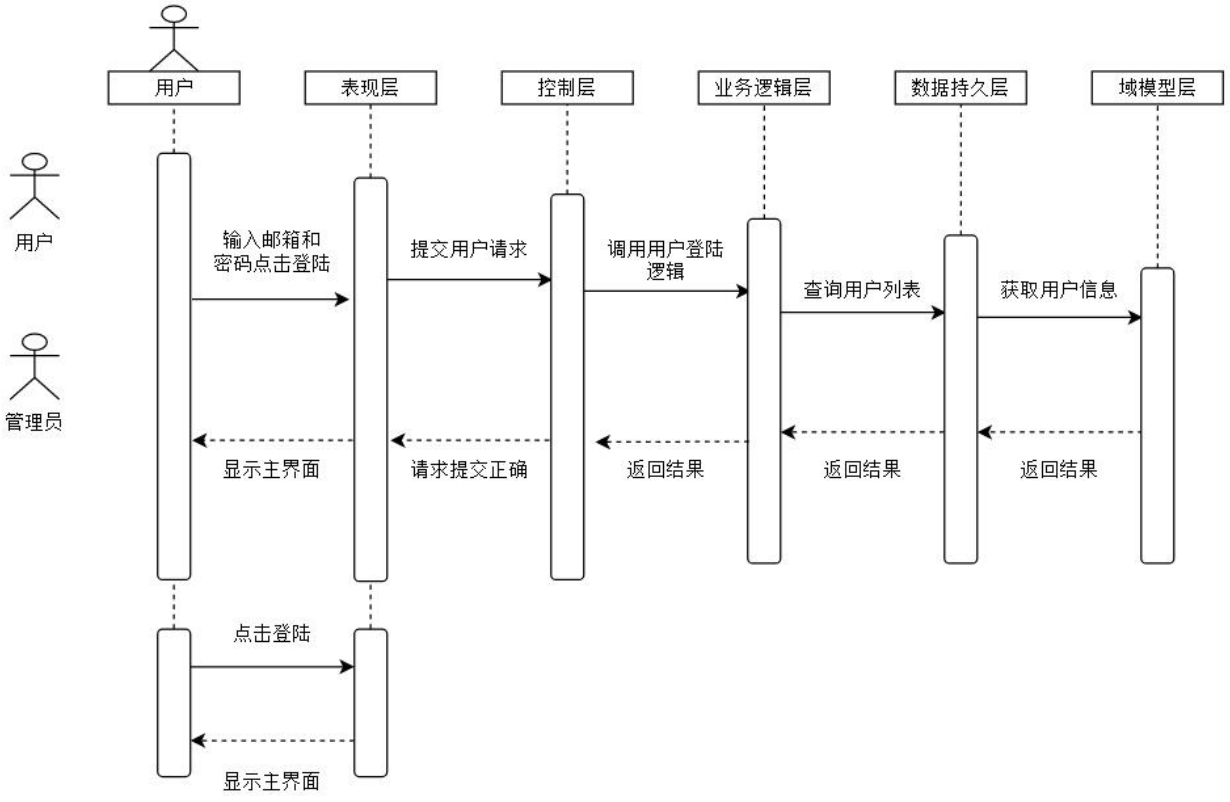

图 5-2 1 登录系统操作流程

5.2.2 模块 CM2：注册模块

编号：CM2；

模块名称：注册系统；

功能简介：本模块为系统注册模块。用户首次进入系统时。可通过本模块进行注册。在此模块中，系统显示注册界面，用户输入相关必要的身份信息，单击确定。若注册成功，系统将以注册邮箱作为账号，用户设定密码作为密码存入后台数据库。注册完成后，用户可使用注册成功的账号和密码登录系统；

输入：注册邮箱，密码，验证码等相关信息；

输出：用户是否注册成功；

备注：对于未登录系统的游客用户，系统将自动限制一部分功能的显示；

操作流程：如图 5-2 2 注册系统操作流程所示。

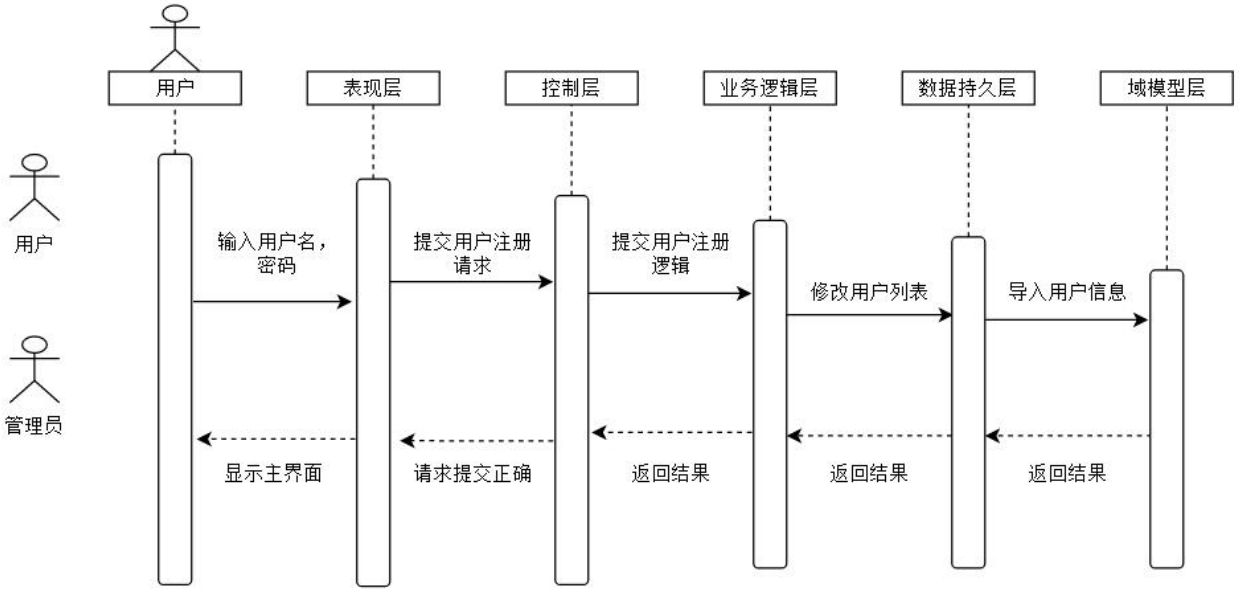

图 5-2 2 注册系统操作流程

5.2.3 模块 CM3：发表动态

编号：CM3；

模块名称：发表动态；

功能简介：用户点击网页的论坛板块，网页按时间排序呈现出用户（包括自己）最近发布的动态，每个动态可进行大致浏览和点击阅读；用户点击发表动态按钮，跳转发表动态页面，用户上传写好的动态内容，完毕后点击发布动态，系统更新动态列表，更新成功后动态成功显示在论坛页面供其他用户阅读；用户点击我的动态按钮，系统刷新，网页按时间排序呈现出用户自己最近发布的动态；

输入：文本信息；

输出：页面刷新；

操作流程：如图 5-2 3 论坛系统操作流程所示。

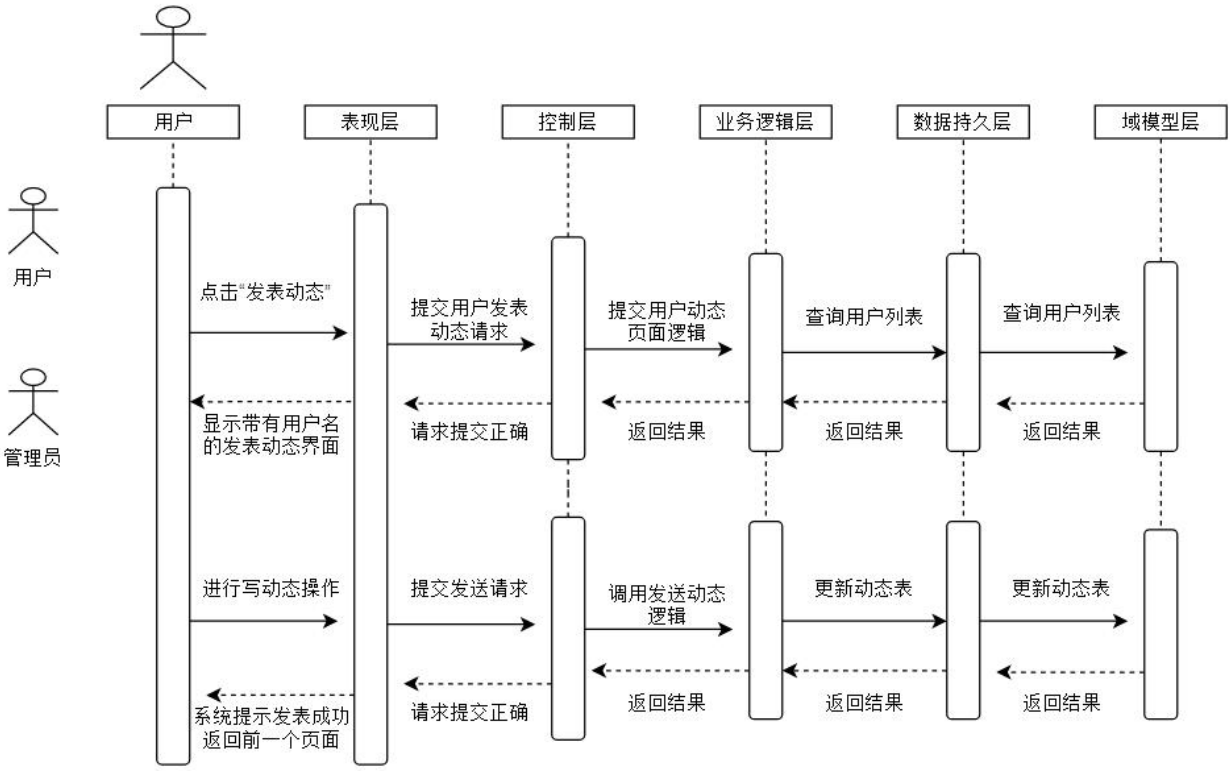

图 5-2 3 论坛系统操作流程

5.2.4 模块 CM4：评论系统

编号：CM4；

模块名称：评论系统；

功能简介：用户进入论坛界面，可以看到其他用户发布的动态以及动态下的评论，用户可以对其他用户发布的动态进行评论，也可以对评论进行回复，系统更新评论列表，更新成功后评论显示在论坛页面可供其他用户查看；

输入：文本信息；

输出：页面刷新；

备注：自己发表的评论可以进行修改和删除操作，但是不能操作他人评论

操作流程：如图 5-2 4 论坛系统操作流程所示。

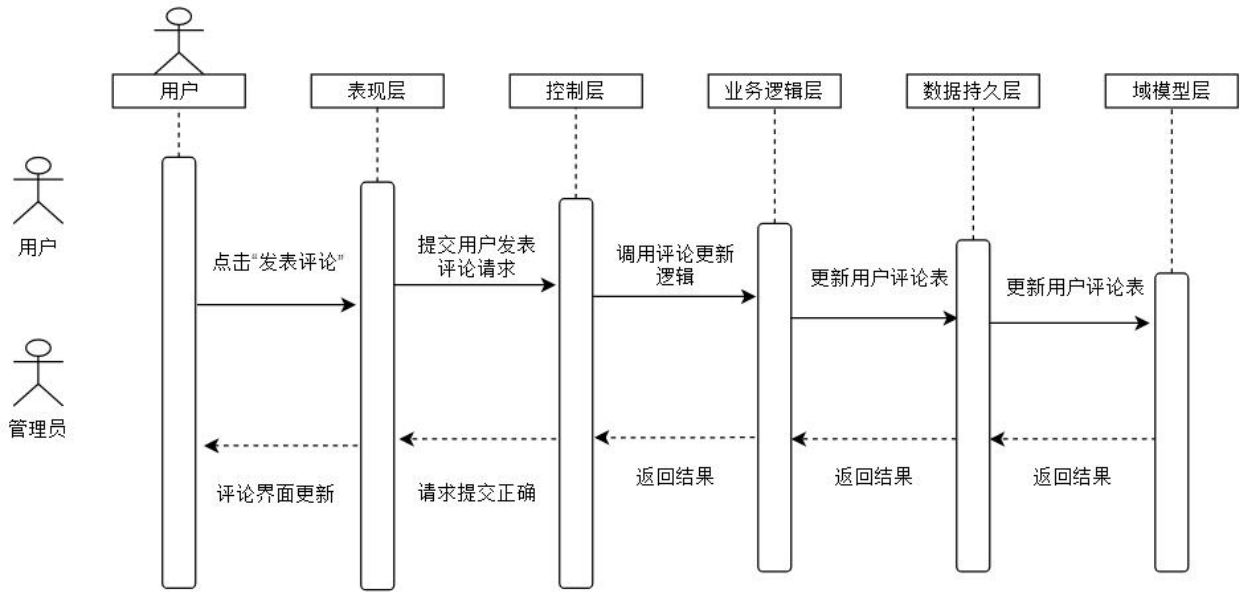

图 5-2 4 论坛系统操作流程

5.2.5 模块 CM5：用户搜索

编号：CM5；

模块名称：用户搜索；

功能简介：用户进入网站搜索模块，通过输入歌曲名或歌手的关键字可进行精确搜索，输入筛选条件也可进行条件搜索。点击立即搜索按钮系统将访问数据库中的歌曲表和歌手表，筛选返回符合条件的信息，响应成功后刷新界面并显示满足条件的信息；

输入：文本信息、鼠标点击事件；

输出：页面刷新；

操作流程：如图 5-2 5 用户搜索操作流程所示。

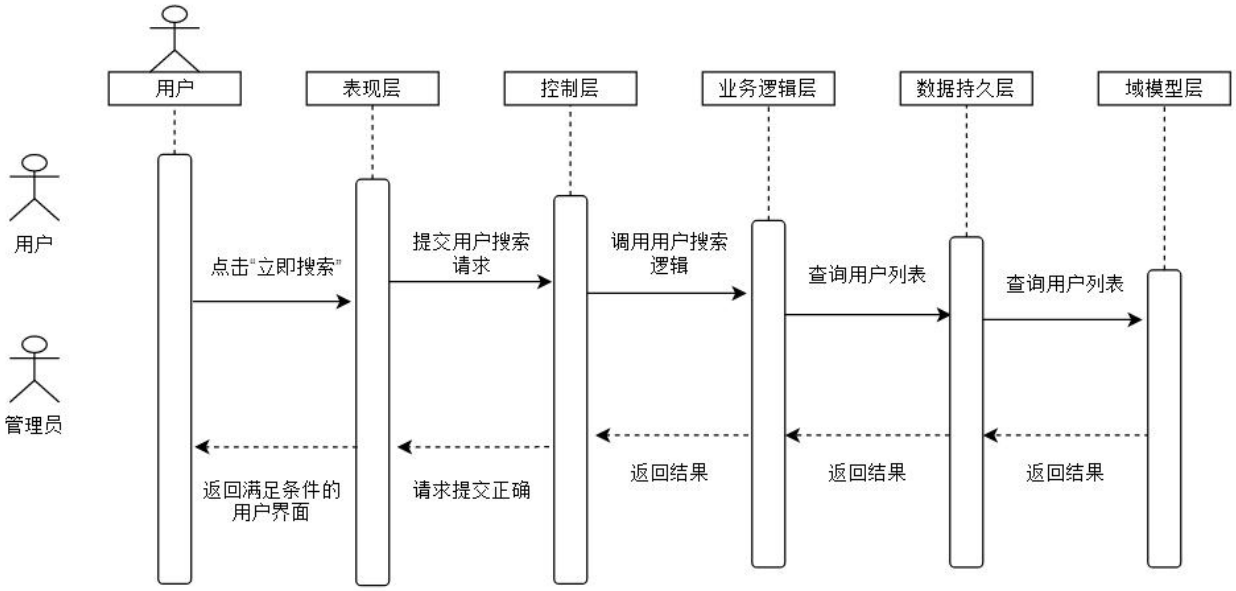

图 5-2 5 用户搜索操作流程

5.2.6 模块 CM6：私信管理

编号：CM6；

模块名称：私信管理系统；

功能简介：用户进入播放界面点击用户或歌手的头像，进入该用户的主页，点击发私信按钮在弹出来的对话框中输入想要发送的信息，点击发送按钮，系统将给收件人发送消息，提醒其有新的私信；

输入：文本信息；

输出：页面刷新；

操作流程：如图 5-2 6 私信管理系统操作流程所示。

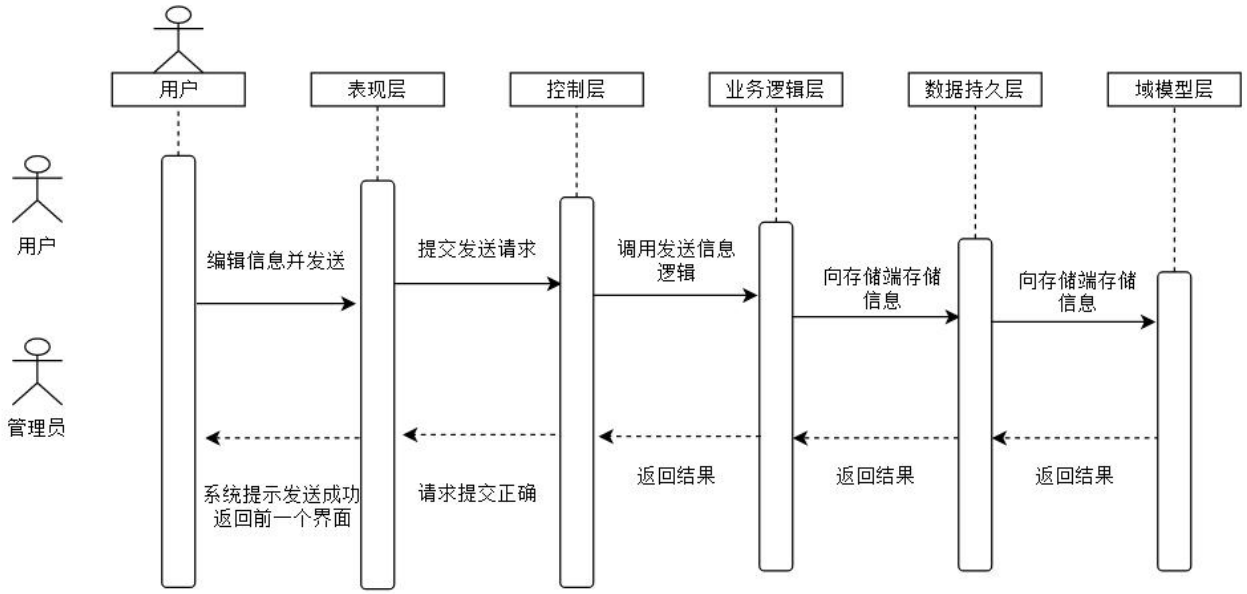

图 5-2 6 私信管理系统操作流程

5.2.7 模块 CM7：信息修改

编号：CM7；

模块名称：信息修改；

功能简介：用户点击网页的个人中心模块，点击个人资料选项，这里呈现的是用户的个人详细信息，鼠标点击信息选项卡可进行信息修改，修改完毕点击保存，个人信息将会存入用户信息表，更新的用户信息替换掉原来信息保存在个人中心的个人资料里；

输入：鼠标点击事件；

输出：页面刷新；

操作流程：如图 5-2 7 信息修改操作流程所示。

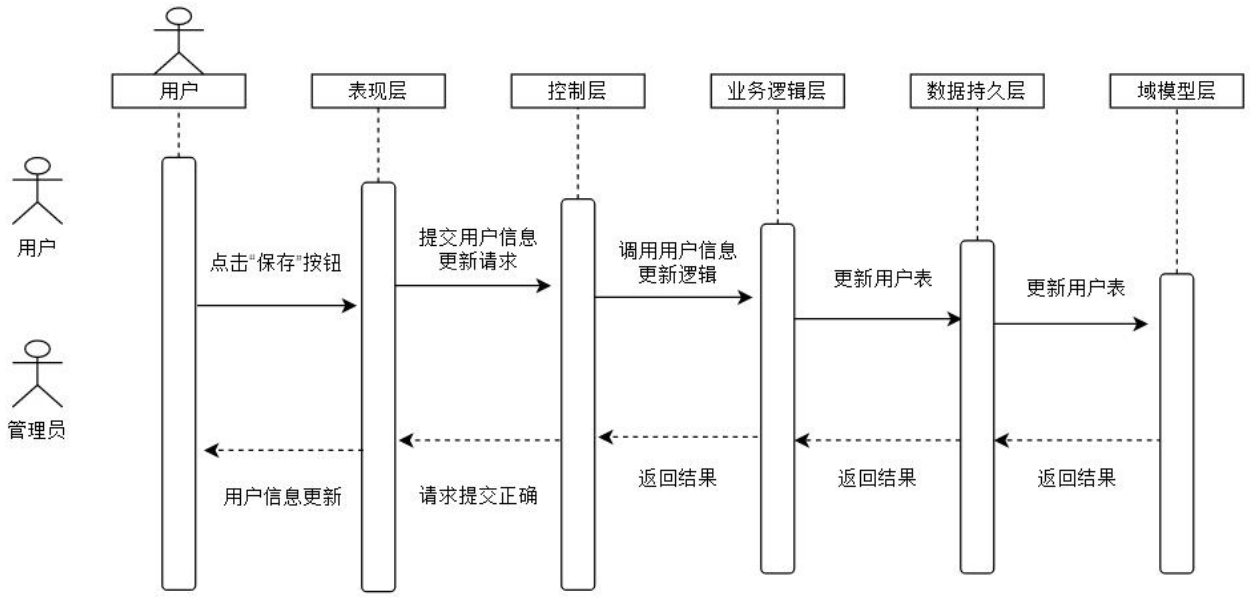

图 5-2 7 信息修改操作流程

5.2.8 模块 CM8：会员管理

编号：CM8；

模块名称：会员管理；

功能简介：用户进入个人中心界面，点击会员按钮进入会员界面，用户可以在该界面开通/续费会员，若开通/续费成功过，系统更新会员表，页面刷新并提示用户开通/续费成功的信息；

输入：鼠标点击事件；

输出：页面刷新；

操作流程：如图 5-2 8 信息修改操作流程所示。

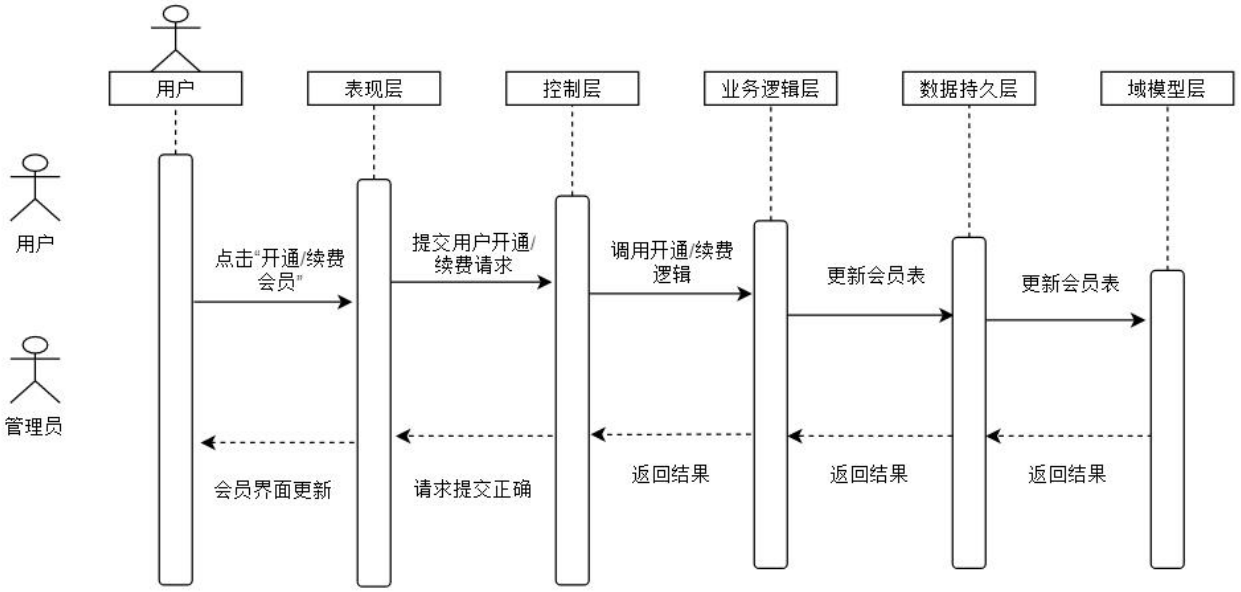

图 5-2 8 信息修改操作流程

5.3 后台管理模块设计

5.3.1 模块 AM1：用户管理

编号：AM1；

模块名称：用户管理；

功能简介：管理员进入“用户管理”模块，选择查看系统中已经存在的用户信息。系统显示全部用户信息，包括用户名，用户邮箱，用户状态， 并提供封禁，解禁，查询操作。管理员选择相应的用户进行封禁，解禁，并点击系统提示的确认操作。系统更新用户列表；

管理员输入完整用户邮箱名，选择查询操作。系统显示该用户，并提供封禁，解禁操作；

输入：鼠标点击事件；

输出：页面刷新；

操作流程：如图 5-3 1 用户管理操作流程所示。

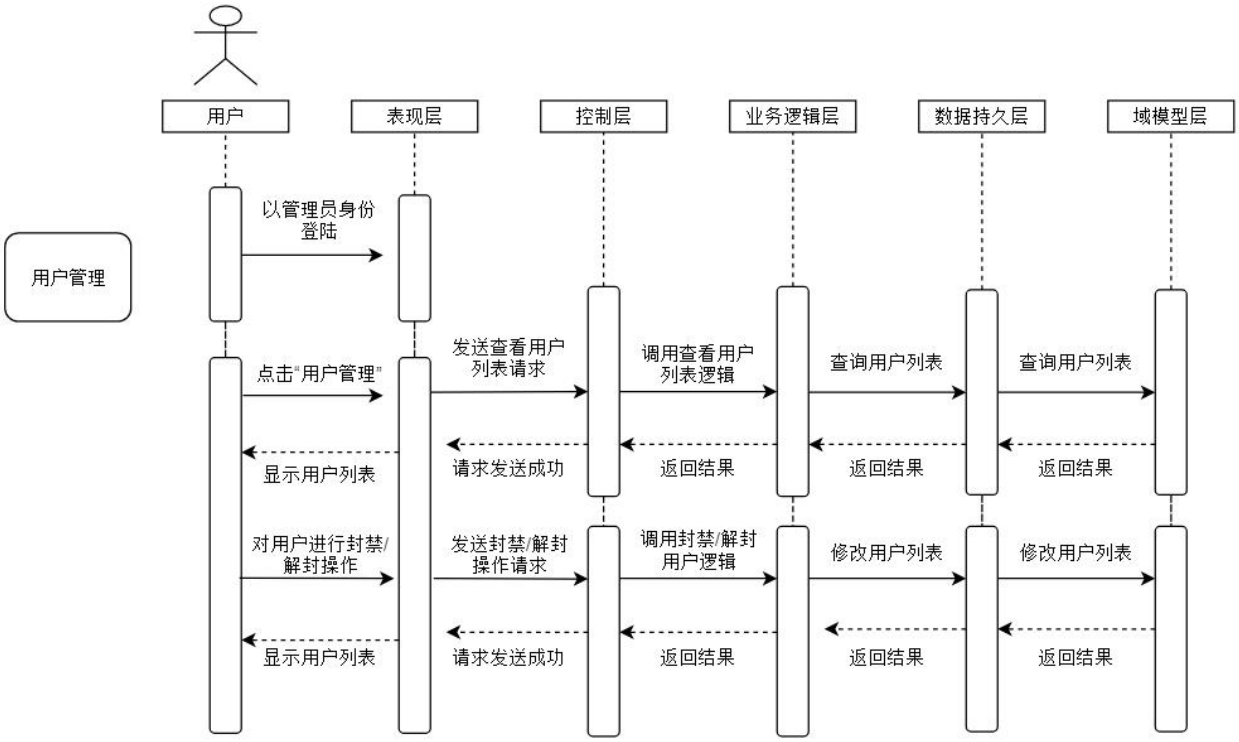

图 5-3 1 用户管理操作流程

5.3.2 模块 AM2：歌曲管理

编号：AM2；

模块名称：歌曲管理；

功能简介：管理员进入“歌曲管理”模块，系统显示当前系统中存在的“歌曲信息”列表，包括歌曲名称，音乐人，并提供上传和删除操作。管理员选择上传歌曲，并确认操作。系统更新歌曲信息列表。管理员选择删除歌曲，并确认操作。系统更新歌曲信息列表；

输入：鼠标点击事件；

输出：页面刷新；

操作流程：如图 5-3 2 歌曲管理操作流程所示。

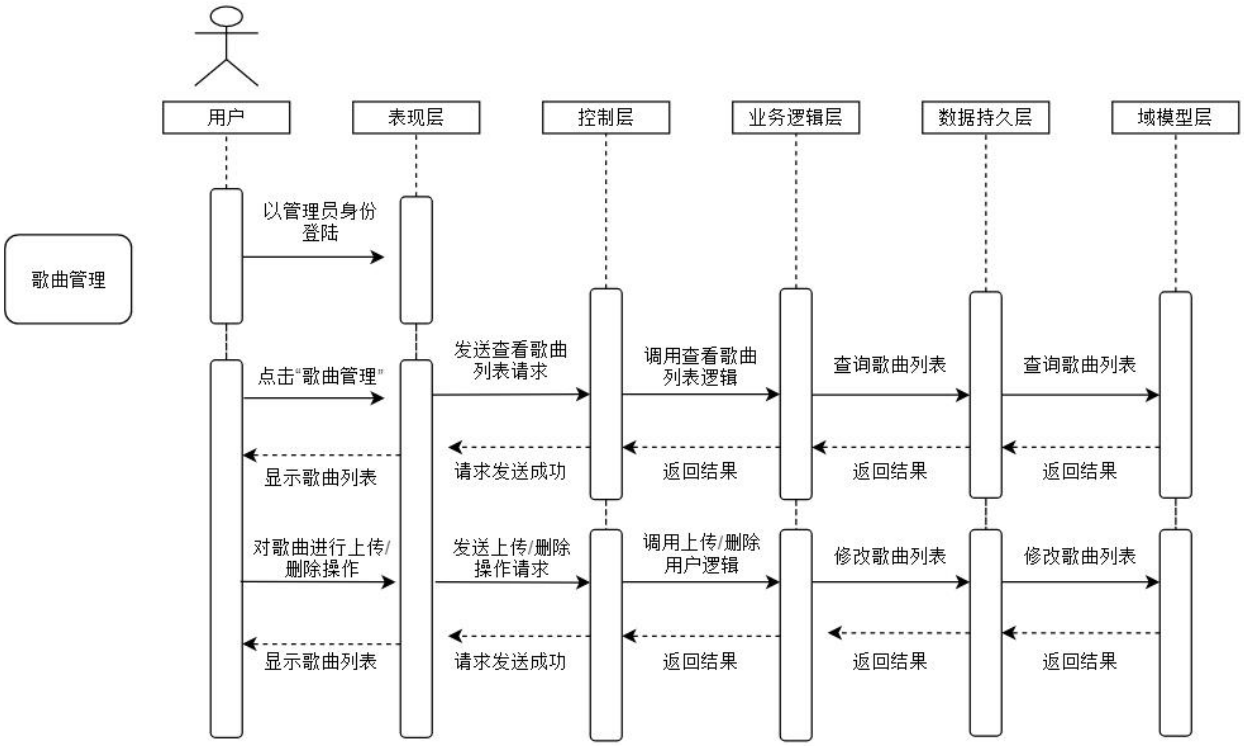

图 5-3 2 歌曲管理操作流程

6. 数据库设计

6.1 数据库种类特点

本项目采用了 MySQL 5.7.23 的数据库。MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 公司，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL 具有以下几个特性：

（1）MySQL 为多种编程语言提供了 API。这些编程语言包括 C、C++、Python、Java、Perl、PHP、Eiffel、Ruby,.NET 和 Tcl 等。

（2）优化的 SQL 查询算法，有效地提高查询速度。

（3)提供多语言支持,常见的编码如中文的 GB2312、BIG5,日文的 Shift_JIS 等都可以用作数据表名和数据列名。

（4）提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接途径。

（5）提供用于管理、检查、优化数据库操作的管理工具。

（6）支持大型的数据库。可以处理拥有上千万条记录的大型数据库。

（7）MySQL 使用标准的 SQL 数据语言形式。

6.2 数据库逻辑结构

经过充分的调研和分析，我们将“音乐吧 online 平台”的数据中设计了 11 个实体，其中包括 Users 表，Forum Passage 表，Hobby 表，Comment 表，Music Box 表，Member 表，Music 表，PlayMusic 表，SearchHistory 表，Message 表，Manager 表 Users_has_PlayMusic。针对每个实体，又有其对应的属性。他们之间的关系主要包括以下几项：

- 用户表，搜索历史表，论坛文章表，兴趣爱好表独立存在，用户可以有多个搜索历史，用户可以在论坛发表多篇动态，可以有多个爱好，因此用户与爱好，论坛文章，搜索历史是一对多的关系。用户发表的动态可以有多个评论，所以文章和评论是一对多的关系。音乐盒中可以有多个音乐播放列表，所以音乐盒和播放列表是一对多的关系
- 一首音乐可以被放入多个音乐播放列表，一个音乐播放列表可以放入多首音乐，所以音乐与音乐播放列表是多对多的关系。
- 用户只有一个音乐盒，而每个音乐盒对应的用户只有一个，所以用户和音乐盒是一对一的关系。
- 用户发表的一篇文章可以有多个评论，所以文章和评论是一对多的关系。
- 管理员表是独立存在的，它负责管理员登录后台对网站进行管理。

综上分析，音乐吧 online 平台前端的实体关系图（概念数据模型）如图 6-2 1 前端实体关系图所示。

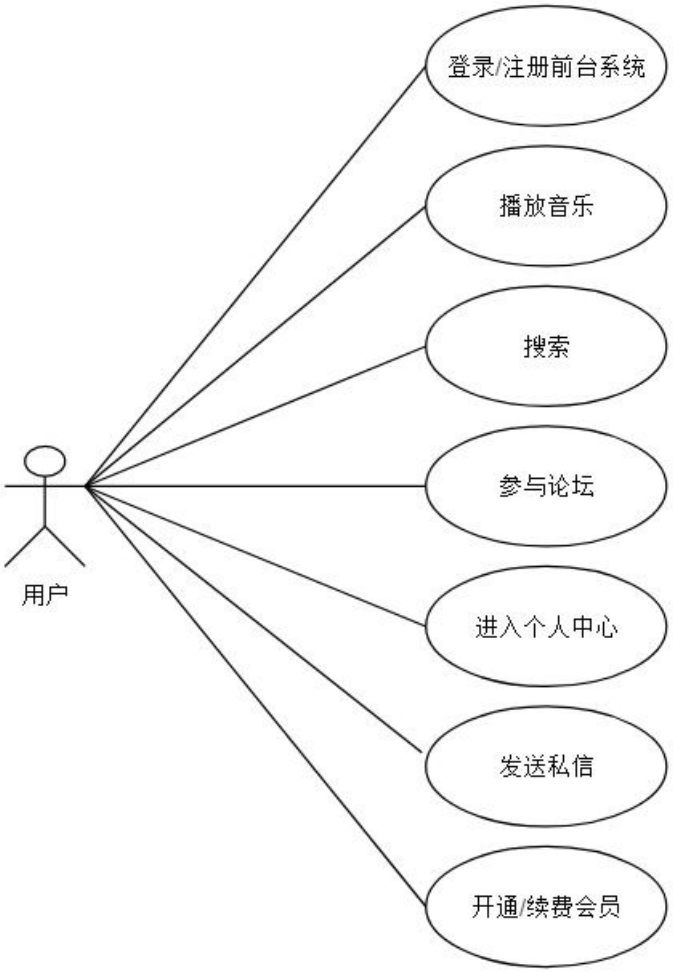

图 6-2 1 前端实体关系图

6.3 物理结构设计

根据上面的实体关系，设计数据库表以及根据数据模型图 5-2 可知，系统一共应有 12 张表。下面是关于表的详细说明。如图 6-3 1 数据库表的关系图所示：

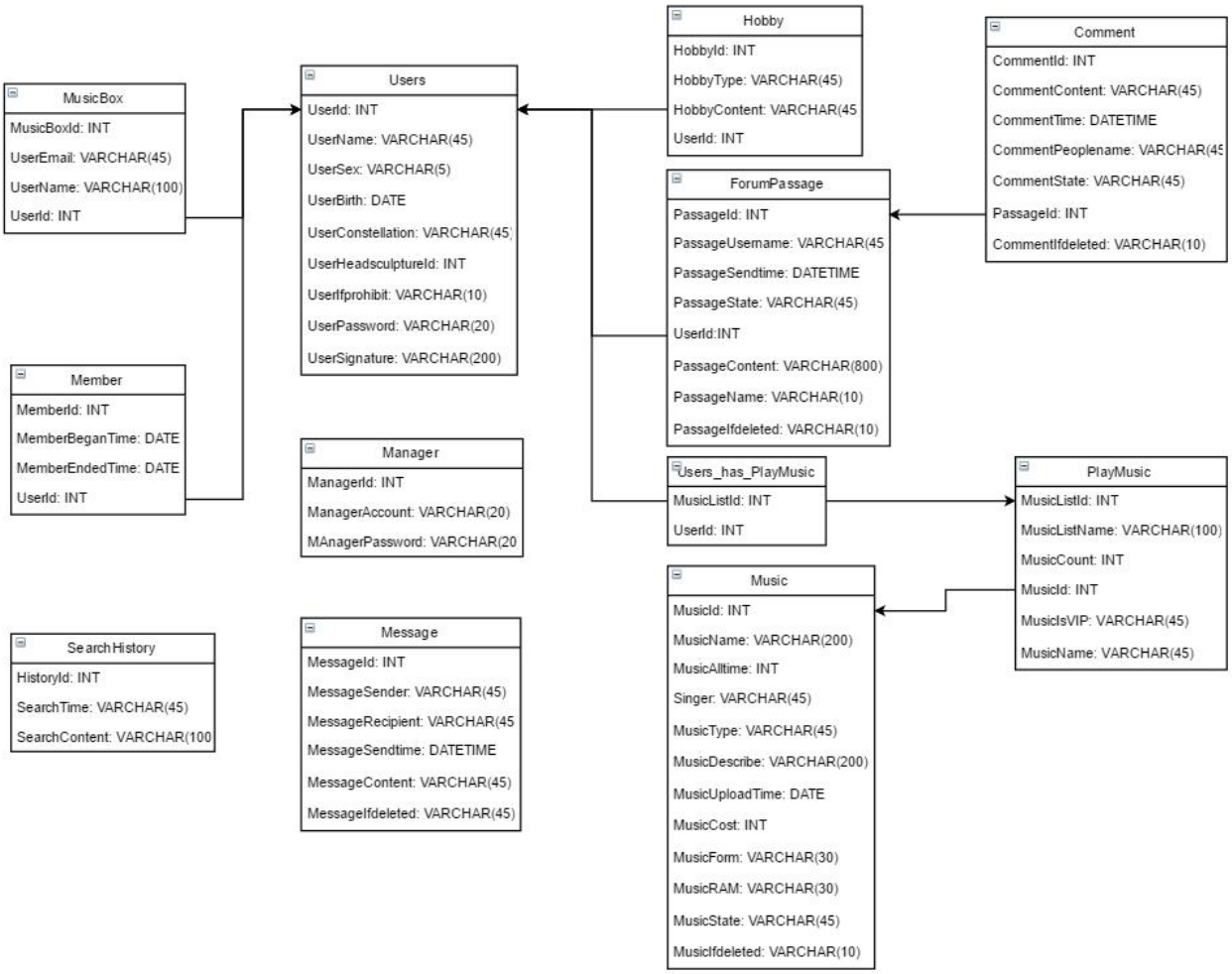

图 6-3 1 数据库表的关系图

6.3.1 表 1：Users 表

用户表标识用户信息，其表结构如表 6-3 2 用户表所示：

表 6-3 2 用户表

Users 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	UserId	Int(11)	N	主键
用户名	UserName	Varchar(45)	N	
用户性别	UserSex	Varchar(5)	N	
用户出生日期	UserBirth	Date	Y	
用户星座	UserConstellation	Varchar(45)	Y	
用户头像标号	UserHeadsculptureId	Int(11)	Y	
用户是否被封禁	UserIfprohibit	Varchar(10)	Y	
用户密码	UserPassword	Varchar(20)	N	
用户个人说明	UserSignature	Varchar(200)	Y	

6.3.2 表 2：ForumPassage 表

论坛文章表标识的是用户发表音乐相关评论动态的信息，如表 6-3 3 论坛文章表所示：

表 6-3 3 论坛文章表

ForumPassage 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	PassageId	Int(11)	N	主键
动态作者	PassageUsrname	Varchar(45)	N	
动态发表时间	PassageSendtime	Datetime	N	
动态状态	PassageState	Varchar(45)	Y	
作者邮箱	UserEmail	Varchar(45)	N	外键
动态内容	PassageContent	Varchar(800)	N	
动态标题	PassageName	Varchar(10)	N	
动态是否被删除	PassageIfdeleted	Varchar(10)	Y	

6.3.3 表 3：Hobby 表

兴趣爱好表标识的是用户个人的兴趣爱好，其表结构如表 6-3 4 兴趣爱好表所示：

表 6-3 4 兴趣爱好表

Hobby 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	HobbyId	Int(11)	N	主键
爱好类型	HobbyType	Varchar(45)	N	
爱好内容	HobbyContent	Varchar(45)	N	
用户邮箱	UserEmail	Varchar(45)	N	外键

6.3.4 表 4：Comment 表

评论表标识的是用户对音乐文章动态进行评论的信息，其表结构如表 6-3 5 评论表所示：

表 6-3 5 评论表

Comment 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	CommentId	Int(11)	N	主键
评论内容	CommentContent	Varchar(45)	N	

评论时间	CommentTime	Datetime	N	
评论用户名字	CommentPeoplename	Varchar(45)	Y	
评论状态	CommentState	Varchar(45)	N	
被评论动态标号	PassageId	Int(11)	N	外键
评论是否被删除	CommentIfdeleted	Varchar(10)	N	

6.3.5 表 5：MusicBox 表

音乐盒表标识的是用户对应的音乐盒的信息，其表结构如表 6-3 6 音乐盒表所示：

表 6-3 6 音乐盒表

MusicBox 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	MusicBoxId	Int(11)	N	主键
用户邮箱	UserEmail	Varchar(45)	Y	
用户名字	UserName	Varchar(100)	Y	
用户标号	UserId	Datetime	N	外键

6.3.6 表 6：Member 表

会员表标识的是用户是否为会员、会员相关的信息，其表结构如表 6-3 7 会员表所示：

表 6-3 7 会员表

Member 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	MemberId	Int(11)	N	主键
成为会员时间	MemberBeganTime	date	N	
会员到期时间	MemberEndedTime	date	N	
用户 Id	UserId	Int(11)	N	外键

6.3.7 表 7：Music 表

音乐表标识的是网站中所有歌曲的信息，其表结构如表 6-3 8 歌曲表所示：

表 6-3 8 歌曲表

Music 表				
字段名	字段代码	字段类型	可否为	备注

			空	
自增主键	MusicId	Int(11)	N	主键
音乐名称	MusicName	Varchar(200)	N	
音乐时长	MusicAlltime	Int(11)	N	
演唱歌手	Singer	Varchar(45)	N	
音乐类型	MusicType	Varchar(45)	N	
音乐描述	MusicDescribe	Varchar(200)	Y	
音乐上传时间	MusicUploadTime	Date	N	
音乐是否为会员歌曲	MusicCost	Int(11)	N	
音乐格式	MusicForm	Varchar(30)	N	
音乐大小	MusicRAM	Varchar(30)	N	
音乐状态	MusicState	Varchar(45)	N	
音乐是否被删除	MusicIfdeleted	Varchar(10)	N	

6.3.8 表 8：PlayMusic 表

音乐播放列表标识的是歌单的名字，歌单中歌曲的名字、数量等，其表结构如表 6-3 9 音乐播放表所示：

表 6-3 9 音乐播放表

PlayMusic 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	MusicListId	Int(11)	N	主键
列表名称	MusicListName	Varchar(100)	N	
歌曲数量	MusicCount	Int(11)	N	
歌曲 ID	MusicId	Varchar(100)	N	外键
歌曲是否为会员歌曲	MusicIsVIP	Varchar(45)	N	
歌曲名称	MusicName	Varchar(45)	N	

6.3.9 表 9：SearchHistory 表

搜索历史表标识的是用户在搜索界面搜索歌曲的信息，其表结构如表 6-3 10 搜索历史表所示：

表 6-3 10 搜索历史表

SearchHistory 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	HistoryId	Int(11)	N	主键
搜索时间	SearchTime	Varchar(45)	N	
搜索内容	SearchContent	Varchar(100)	N	

6.3.10 表 10：Users_has_PlayMusic 表

过渡表标识的是音乐播放列表和用户多对多之间的过渡信息，其表结构如表 6-3 11 过渡表所示：

表 6-3 11 过渡表

Usrs_has_PlayMusic 表				
字段名	字段代码	字段类型	可否为空	备注
列表标号	MusicListId	Int(11)	N	联合主键
用户标号	UsrId	Int(11)	N	联合主键

6.3.11 表 11：Message 表

信息表标识的是所有网站内私信的信息，其表结构如表 6-3 12 私信表所示：

表 6-3 12 私信表

Message 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	MessageId	Int(11)	N	主键
信息发送人	MessageSender	Varchar(45)	N	外键
信息收件人	MessageRecipient	Varchar(45)	N	外键
信息发送时间	MessageSendtime	Datetime	N	
信息内容	MessageContent	Varchar(45)	N	
信息是否被删除	MessageIfdeleted	Varchar(45)	Y	

6.3.12 表 12：Manager 表

管理员表标识的是后台管理员的登录信息，其表结构如表 6-3 13 管理员表所示：

表 6-3 13 管理员表

Manager 表				
字段名	字段代码	字段类型	可否为空	备注
自增主键	ManagerId	Int(11)	N	主键
管理员账号	ManagerAccount	Varchar(20)	Y	
管理员密码	ManagerPassword	Varchar(20)	Y	

7. 界面设计

7.1 首页设计

平台首页采用图 6-1 中的格式设计。网页顶部为“音乐吧 Oline”的图标。图标右方为搜索栏，登录/注册，私信。

搜索栏会弹出下拉列表，包含搜索历史及相关热点。

如果登录则显示用户头像，如图 6-2.

点击私信图标跳转到私信界面，点击登录/注册跳转到登录/注册界面。

首页主要包括推荐歌曲及新歌速递。

下方为歌曲播放条，点击歌曲封面跳转到歌曲播放界面，点击右上角列表图标跳转到歌曲列表界面。如图 7-1 1 首页设计图　图 7-1 2 首页设计图所示:

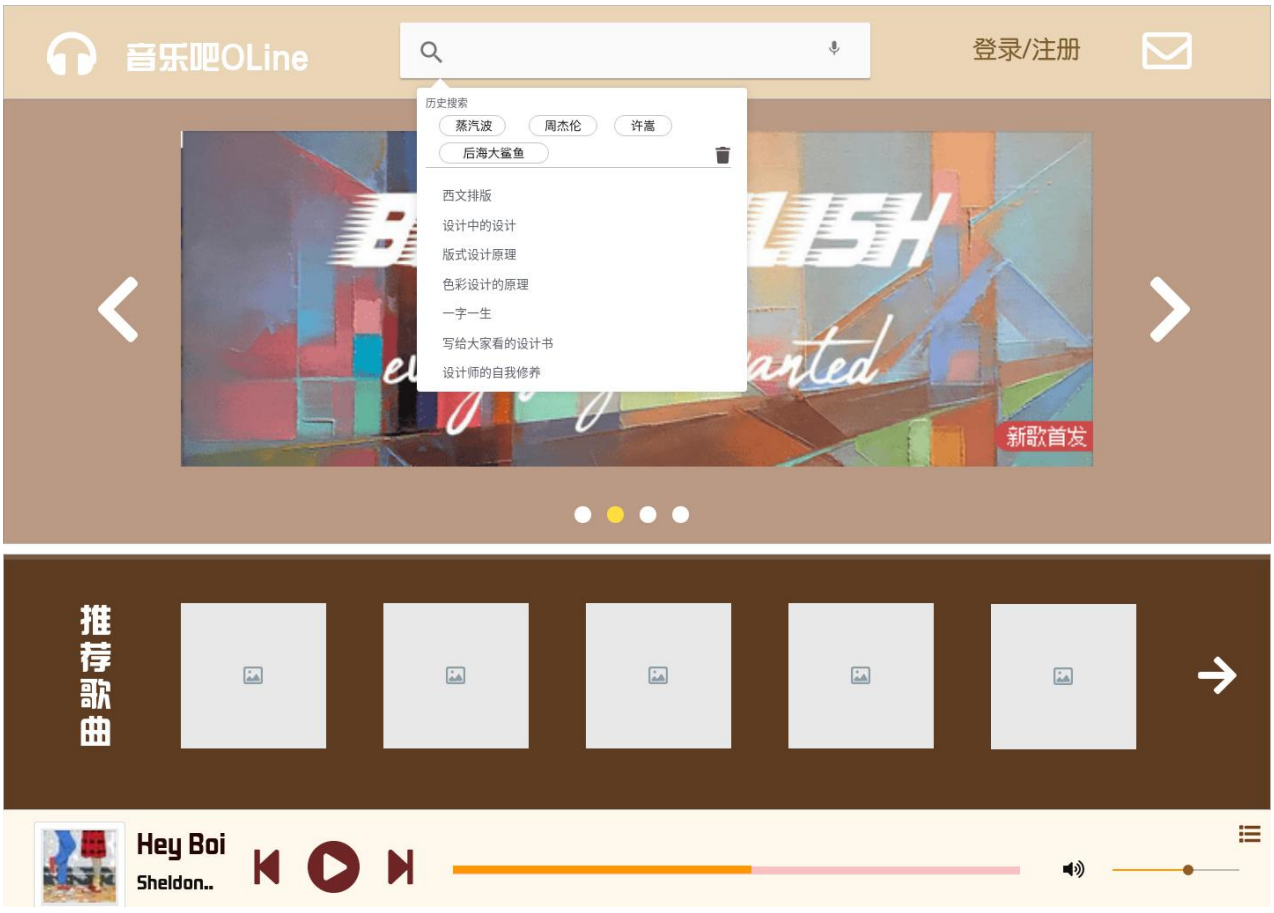

图 7-1 1 首页设计图

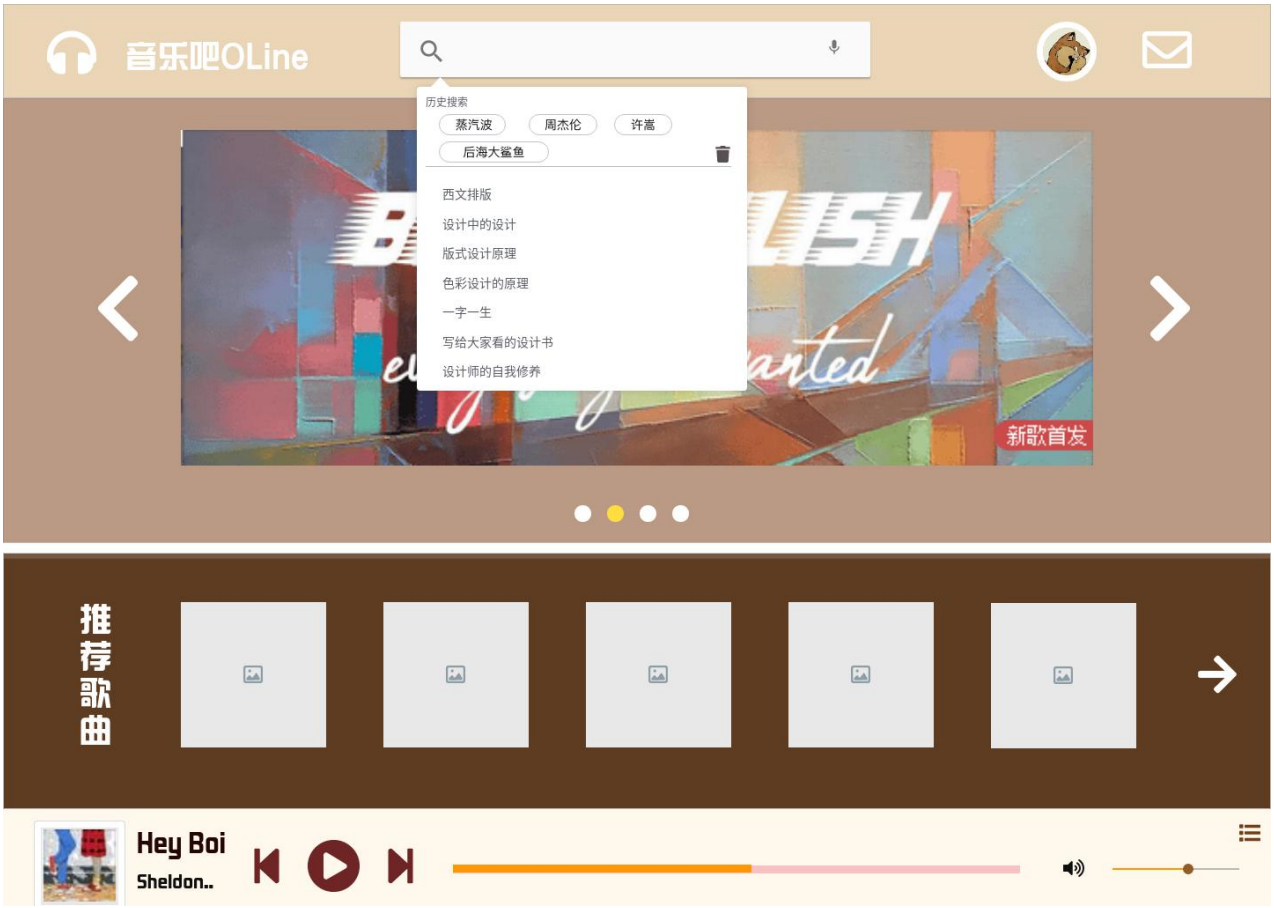

图 7-1 2 首页设计图

7.2 登录/注册界面

注册界面要求填写用户名，绑定邮箱，手机号，密码信息。

如图 7-2 1 所示：

图 7-2 1

7.3 歌曲播放界面

播放列表主要为展示当前播放的歌曲列表，表示播放为哪一首歌曲，左侧有推荐歌曲图标，左上角文字点击可跳转至首页。如图 7-3 1 音乐吧 online 网站个人中心页面设计图所示：

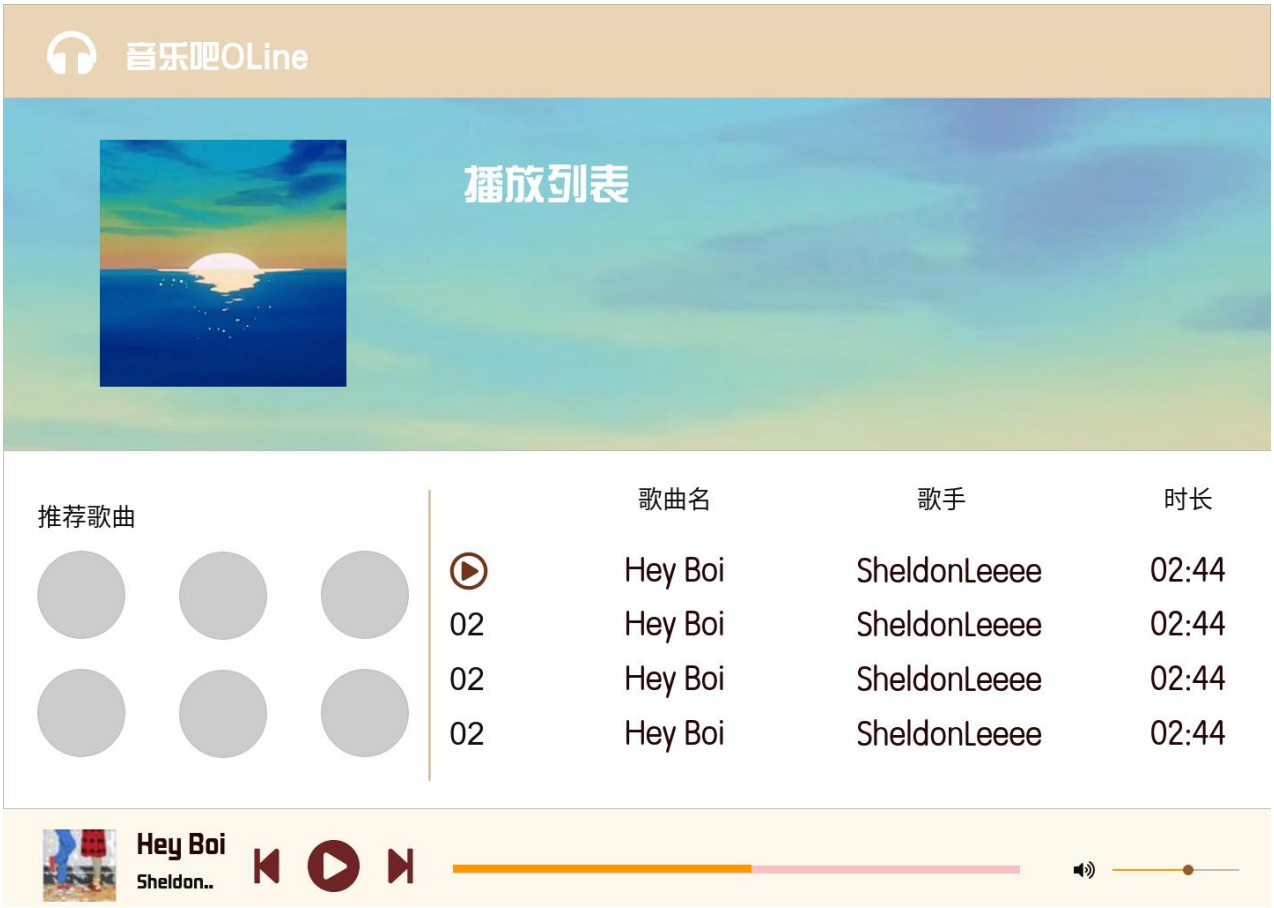

图 7-3 1 音乐吧 online 网站个人中心页面设计图

7.4 论坛界面

论坛界面可以展示用户转发，分享，自行编辑的动态。可对动态进行点赞，评价，评论。左上角文字点击可跳转至首页，右上角用户头像可跳转至个人中心。如图 7-4 1 音乐吧 online 网站活动页面设计图所示：

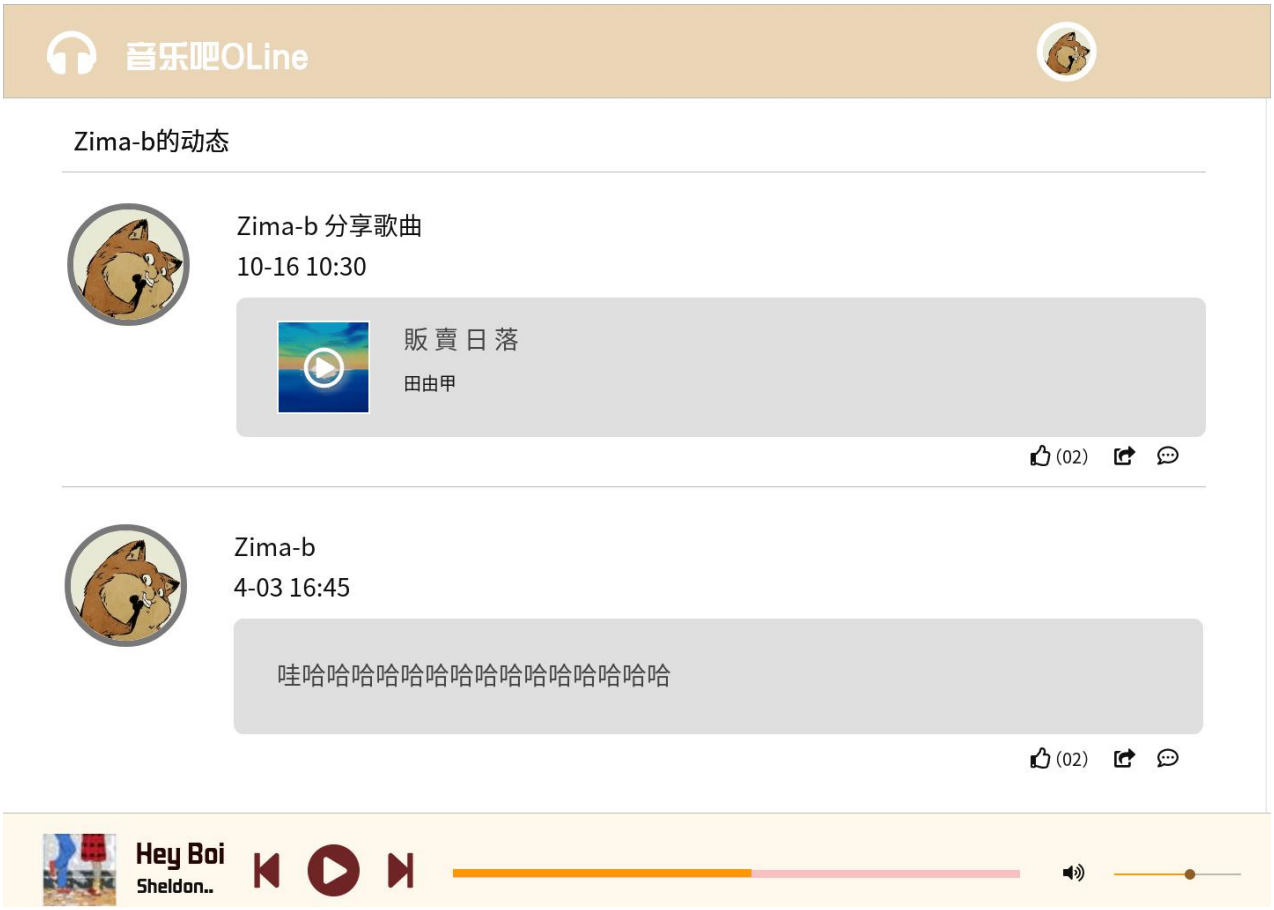

图 7-4 1 音乐吧 online 网站活动页面设计图

7.5 私信界面

分为私信，评论，@我三个部分。

如图 7-5 1 私信界面图所示：

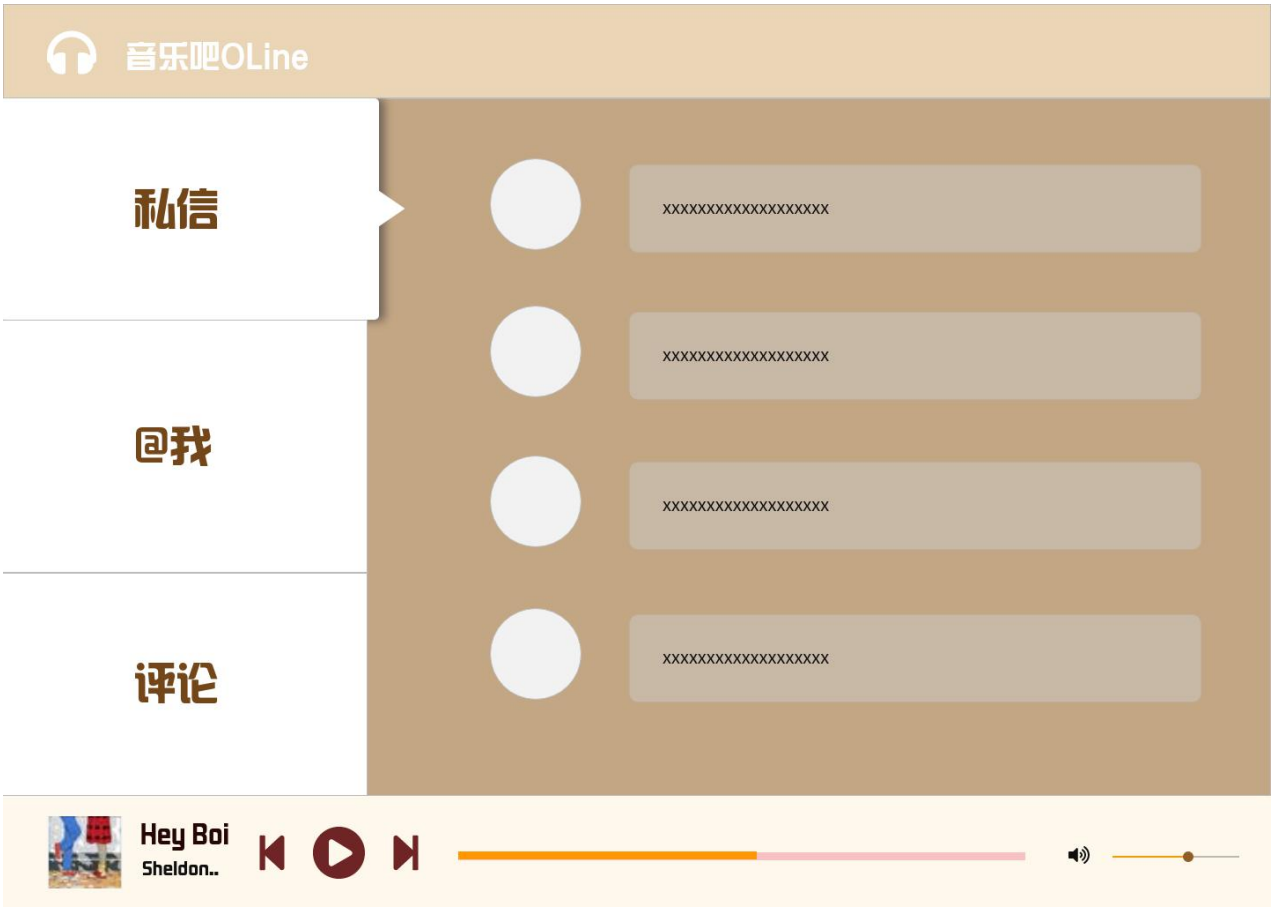

图 7-5 1 私信界面图

7.6 歌曲上传

如图 7-5 1 音乐吧 online 网站用户搜索页面设计图所示：

图 7-5 1 音乐吧 online 网站用户搜索页面设计图

对于所有用户提供精确搜索服务，用户可以根据自身理想要求在线搜索满足条件的用户，用户可以输入昵称进行精确搜索，也可以输入用户的基本信息如性别，年龄区间，工作地区，身高区间，学历，收入区间，职业和星座进行大致搜索。点击立即搜索按钮后，相同昵称的用户将会以头像照片的形式显示在下方区域内，同时一个区域最多显示 8 名用户，最初随机推荐 8 名用户进行显示。点击用户头像便可出现给我写信按钮，方便给被搜索的用户留言或交流，搜索服务为用户提供更精确的用户查找，提更多和心仪伴侣沟通交流的机会。

管理员登录对管理员提供对登录网站的服务，登录界面如图 6-22 所示。

图 7-6 1 后台管理员登录页面设计图

7.7 个人主页

后台主界面显示后台基本信息，具体设计如图 7-7 1 个人主页面设计图 图 7-7 1 后台主页面设计图所示。

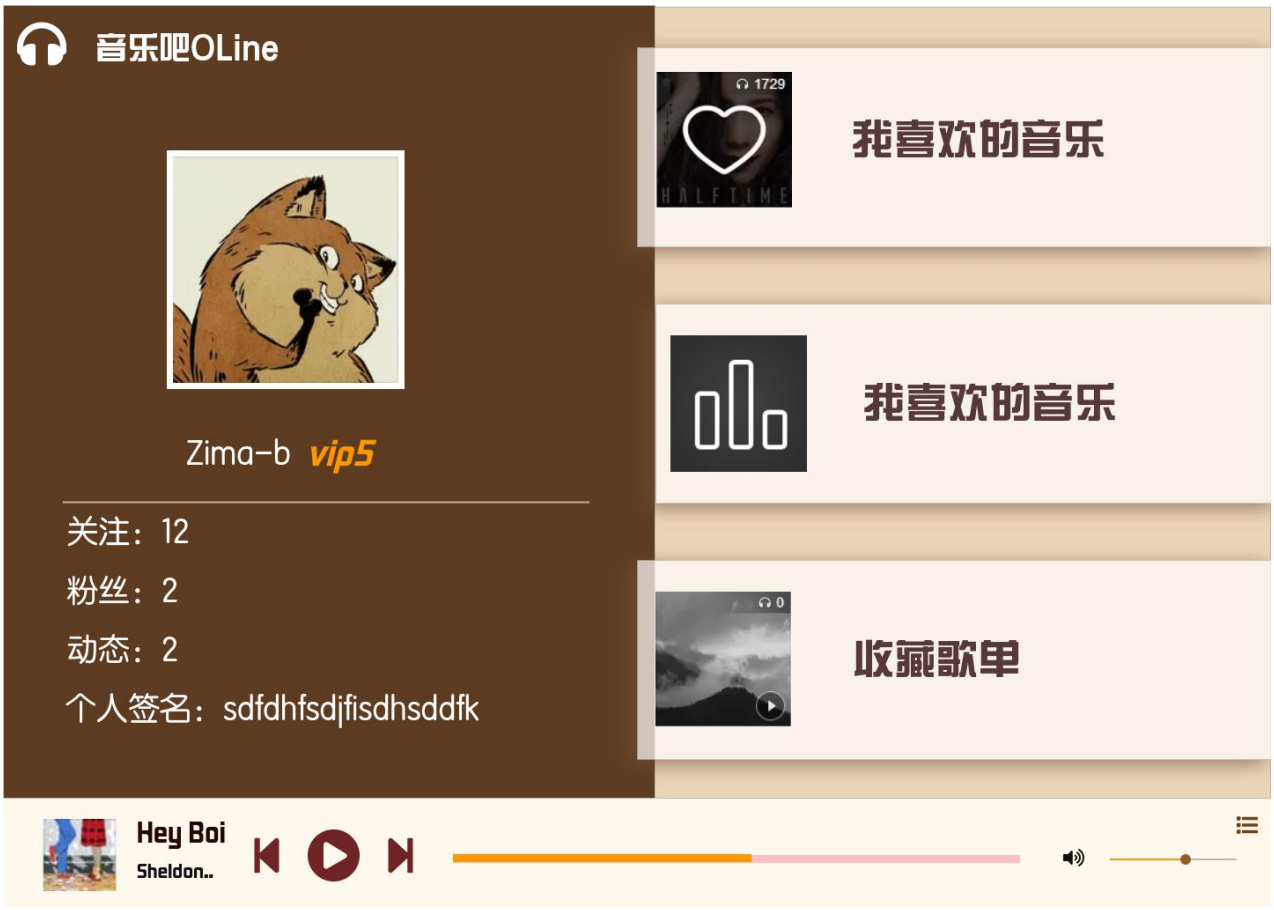

图 7-7 1 个人主页面设计图

7.8 会员界面

后台会员管理界面显示所有会员基本信息，为管理员提供对用户的封禁、解封操作。管理员通过上方搜索栏输入用户完整邮箱精确搜索用户。具体设计如图 7-8 1 会员管理页面设计图所示。

图 7-8 1 会员管理页面设计图

7.9 搜索界面

搜索界面后台活动管理界面显示所有活动的基本信息，为管理员提供对活动的删除、恢复操作。管理员通过上方搜索栏输入活动的完整 id 精确搜索活动。具体设计如图 7-9 1 搜索管理设计图所示：

图 7-9 1 搜索界面设计图

7.10 音乐播放

音乐播放界面显示音乐播放的所有信息。具体设计如图 7-10 1 音乐播放设计图所示：

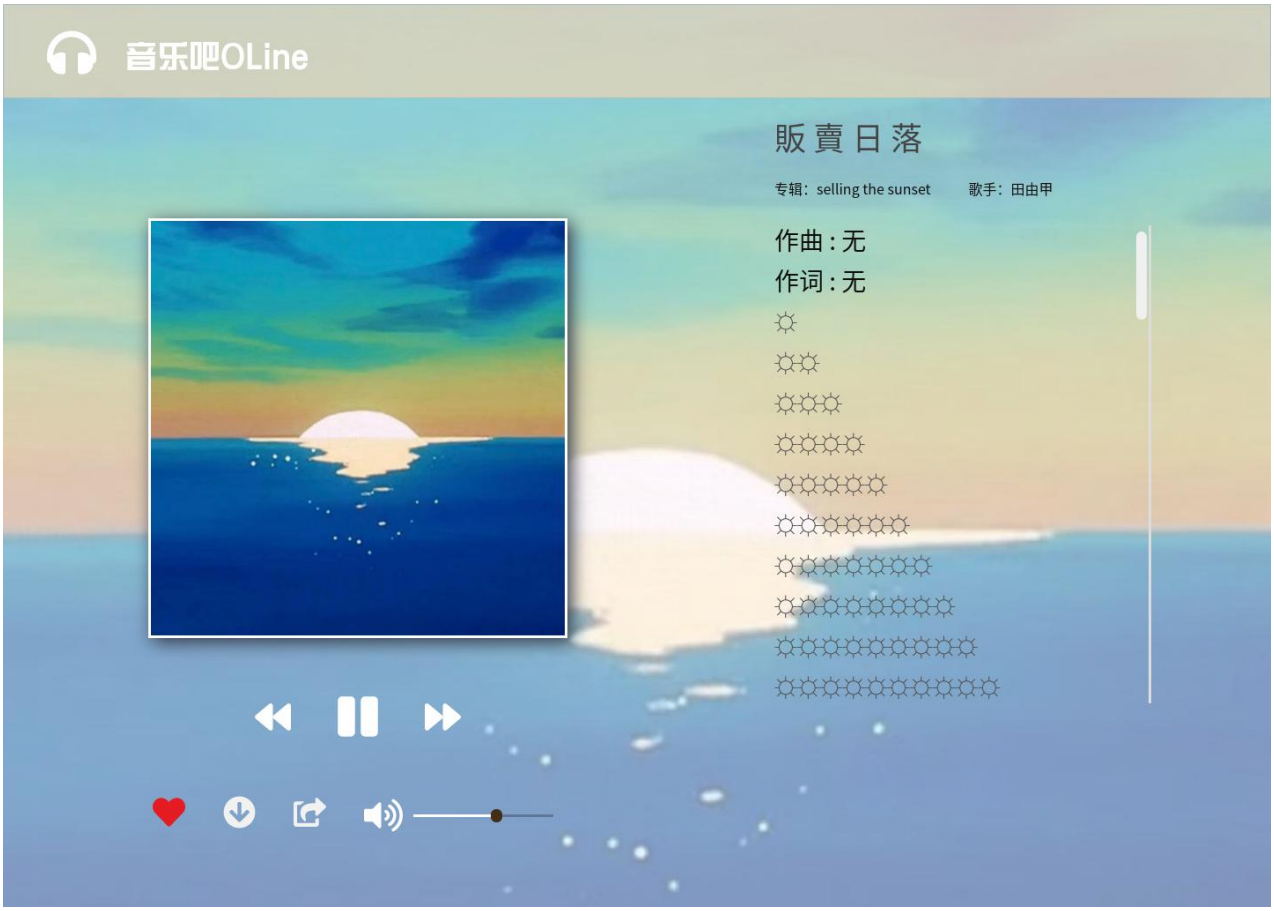

图 7-10 1 音乐播放页面设计图