计算机科学与工程学院课程设计报告

题目全称： 基于B/S架构的学生管理系统

题目难度等级： 5

指导老师： 李玉军 **职称：** 副教授

学生姓名： 杨骐榕 **学号：** 2021080903002

专业： 计算机科学与技术 **班号（大类分流后班号）：** 2021080903

A计算机使用技能（100分）	B课程设计报告（100分）	C计算机专业技能（100分）	总分（A*10%+B*30%+C*60%）

备注：如参加答辩，请答辩老师给出C计算机专业技能的分数。请指导教师给出除计算机专业技能以外的其他分数。

如参加答辩，请答辩老师签字： _____

主要任务： 编写学生管理系统软件，包括前端和后台处理系统。

详细功能描述：

（1）熟悉HTML。

（2）熟悉前端网页设计与编程。

（3）熟悉后端脚本处理程序编程。

（4）熟悉Web服务器的搭建。

（5）后端数据库采用MySQL数据库，设计两张表，一张用户表用于用户验证，另一张学生信息表用于存放学生基本信息，如姓名、学号、电话、邮件等信息。

（6）实现用户登录验证功能。

（7）实现学生基本信息的增、删、改、查功能。

扩展功能：

（1）密码经过哈希处理以密文形式保存。

（2）实现班级信息的增、删、改、查功能。

（3）学生忘记密码可以让管理员重置密码。

预期成果或目标： 综合课程设计报告；可演示的程序和代码。

指导老师评语：

指导教师签字： _____

摘　要

学生管理系统是针对学校和教育机构而设计的信息管理软件，用于管理学生的个人信息和学习情况，提供便捷的管理和教育手段。随着现代学校对学生管理需求的增加，需要一个功能强大、易于使用和高效的系统来满足学校的需求。

传统的学生管理方法存在一些问题，如手动记录数据、信息不易查询和统计，以及效率低下等。因此，需要一种新的学生管理系统，以解决这些问题并提供更好的管理和教育体验。

本系统采用了 Go 语言的 Gin 框架作为后端技术栈，并结合 Vue3+TypeScript 作为前端技术架构，构建了一个功能强大且易于扩展的学生管理系统。后端采用了 Gin 框架，其具有高效、轻量级和易于扩展的特点，能够快速构建高性能的 Web 应用程序。前端采用了 Vue3+TypeScript 技术堆栈，提供了灵活、高效和易于维护的开发环境，帮助开发者快速构建用户友好的界面。

通过该学生管理系统，管理员可以方便地进行用户登录注册、学生信息与班级信息的增删改查，以及学生信息的数据统计和分析等操作。系统提供了便捷的界面和功能，使学校能够更好地管理学生信息，并进行数据统计和分析。这进一步提高了学校的管理效率，为学生提供了更好的教育服务。

关键词：B/S 架构；学生管理系统；Gin；MySQL；Vue3

目 录

第一章 绪 论

1.1 学生管理系统介绍

1.1.1 历史与发展

学生管理系统是一种用于学校或教育机构管理学生信息的系统。该系统的历史可以追溯到计算机技术的早期。早期的学生管理系统主要是由一些基本的数据库系统和简单的数据输入和输出功能组成。随着计算机技术和互联网的不断发展，学生管理系统也在不断演变和发展。

1990 年代，学生管理系统开始应用于大学和高等教育机构。这些系统通常由自己开发的软件或第三方供应商提供的商业软件构成。这些系统包括学生信息管理、课程安排、成绩管理、图书管理、学费管理等模块。

2000 年代，随着互联网的兴起，学生管理系统逐渐向网络化发展。学校可以通过 Web 应用程序来管理学生信息，学生和家长也可以通过互联网查询和更新个人信息。此外，随着移动互联网的兴起，学生管理系统也开始提供移动应用程序，方便学生和家长随时随地查询和更新学生信息。

目前，学生管理系统已成为学校和教育机构不可或缺的系统之一。随着技术的不断进步，学生管理系统也在不断发展，增加了更多的功能和特性，例如智能化数据分析、机器学习、大数据等技术的应用，帮助学校更好地管理学生信息，提高学校的教育质量和管理水平。

1.1.2 操作方式

配置好坏境后，启动相关项目，打开网址 http://localhost:8080，即可进入该系统的主页。

管理员通过邮箱注册账号。登录系统后页面会出现左侧栏，点击左侧栏的用户的子栏目“班级信息”进入班级管理页面，点击“学生信息”进入学生信息管理页面。可以对班级和学生进行增、删、改、查等操作。左侧栏的“设置”可以修改用户名和密码。

管理员添加学生后，学生即可登录系统，登录后页面出现左侧栏，点击左侧栏的用户，子栏目中“个人主页”可以查看个人信息，“修改信息”可以修改个人信息。左侧栏的“设置”可以修改密码。

1.2 B/S 架构介绍

B/S 架构（Browser/Server Architecture），也称为 Web 架构，是一种应用程序架构模式。它将应用程序分成两部分：客户端浏览器和服务端。浏览器作为客户端，发送请求并接收服务器响应，而服务器作为服务端，处理请求并返回响应结果。通常使用 HTTP 协议进行通信。

B/S 结构是在 C/S 结构基础上的技术拓展，实际上具有三层。第一层表示层：Web 浏览器，完成用户接口的功能；第二层功能层：Web 服务器，完成客户的应用功能；第三层数据层：数据库服务器，进行各种数据处理[1]。

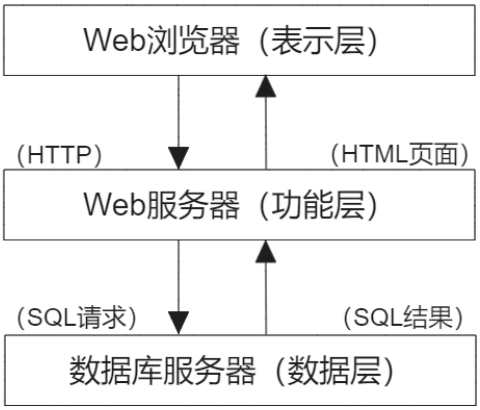

图 1 B/S 结构

1.3 开发框架介绍

1.3.1 Gin 开发框架

Gin 是一种使用 Go 语言编写的 Web 框架，它被广泛应用于构建高性能的 Web 应用程序和 API[2]。以下是 Gin 开发框架的特点：

1. 高性能：Gin 采用了高性能的路由器和处理器，它可以处理大量的 HTTP 请求，并具有低延迟和高吞吐量的特点。Gin 的性能比其他流行的 Web 框架如 Beego 和 Martini 要更高效。

2. 轻量级：Gin 是一个轻量级的框架，它只包含了一些基本的功能，例如路由、中间件和错误处理等。因此，Gin 的二进制文件大小很小，它可以快速启动和运行。

3. 易于扩展：Gin 框架提供了很多有用的中间件和插件，例如 JSON 解析、文件上传、CORS、GZIP 压缩等，这些中间件可以轻松地添加到应用程序中，扩展

应用程序的功能。

4. 简单易用：Gin 的 API 设计非常简单，易于使用和理解。开发者可以很快地上手使用 Gin 框架开发 Web 应用程序和 API。

5. 丰富的文档和社区支持：Gin 框架有着完善的文档和社区支持，开发者可以轻松地找到答案和解决问题。

1.3.2 Gorm 开发框架

Gorm 是一款基于 Go 语言的 ORM（对象关系映射）框架，用于处理数据库中对象的持久化和查询。它支持多种关系型数据库，如 MySQL、PostgreSQL、SQLite、SQL Server 等，同时还提供了类似于 ActiveRecord 的查询语法和一些高级功能，如数据库迁移、关联查询、预加载、事务管理等[3]。

Gorm 采用简单的 API 设计，易于上手，并提供了大量的文档和示例代码，使得开发者可以快速地开发出高效且易于维护的数据库应用程序。它还支持许多常用的特性，如批量插入、批量更新、钩子函数等，使得处理大量数据的情况下更加高效。

1.3.3 Vue3 开发框架

Vue.js 是一款轻量级的 JavaScript 前端框架，用于构建交互式的用户界面。它采用 MVVM（Model-View-ViewModel）架构，将用户界面和业务逻辑进行分离。Vue.js 的核心库只关注视图层，使其更容易集成到其他项目或库中[4]。Vue3 是 Vue.js 的最新版本，相对于 Vue2，它带来了一些重大的改进和升级，包括性能优化、编程体验、可维护性等方面的提升。下面是 Vue3 开发框架的一些特点：

1. 更快的渲染速度：Vue3 采用了更先进的响应式系统，使用 Proxy 代理对象来跟踪数据变化，相比 Vue2 的响应式系统，性能提升了很多。

2. 更小的包大小：Vue3 通过 Tree-shaking 和优化打包算法，将包的大小缩小了约 30%。

3. 更好的 TypeScript 支持：Vue3 对 TypeScript 的支持更加友好，可以提供更好的类型推断和代码智能提示。

4. 更好的编程体验：Vue3 通过 Composition API 重新设计了组件的 API，提供了更好的代码组织结构和可读性，使得编写 Vue 应用更加便捷。

5. 更好的可维护性：Vue3 使用了模块化设计，将不同的功能模块分开，使得应用的代码结构更加清晰，可维护性更强。

1.4 Ant Design Vue 组件库介绍

Ant Design Vue 是基于 Ant Design 设计规范的 Vue UI 组件库。它为 Vue 开发者提供了丰富的 UI 组件和工具集，用于快速构建现代化、美观、易用的 Web 应用程序。

Ant Design Vue 包含了众多的 UI 组件，例如按钮、表单、表格、消息提示、对话框、日历等等，这些组件都是经过精心设计和开发的，能够满足各种不同的界面需求。此外，Ant Design Vue 还包括了一系列的工具和服务，例如数据可视化、国际化支持、主题定制等等。

第二章 系统分析

2.1 系统功能需求分析

在学生信息管理系统中，分为 2 个身份，分别为管理员和学生。管理员根据邮箱和密码进进行注册，在登录后管理员可以先进入班级模块录入班级信息，对班级进行增删改查，班级信息的改动会影响学生信息。管理员还可以在学生模块查看学生基本信息，在必要时进入添加、删除、修改模块进行相关操作。学生主要是查看与修改自己的部分信息，班级、专业与学院只有管理员可以修改。

主要结构如下图所示：

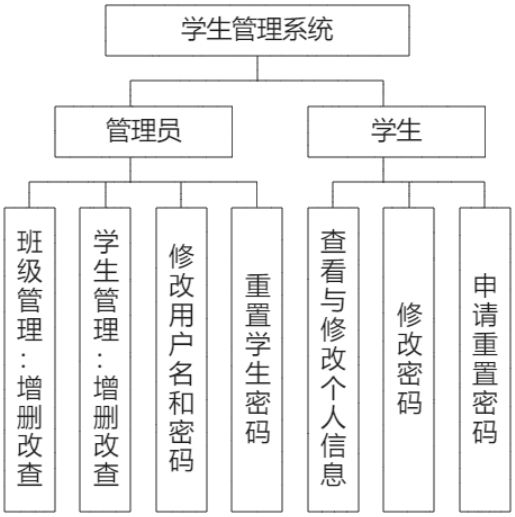

图 2 学生管理系统主要结构

2.2 项目结构分析

2.2.1 数据库设计

该管理系统采用 MySQL 作为后台数据库。MySQL 是一种流行的开源关系型数据库管理系统，用于存储和管理大量结构化数据。它具有可扩展性、高性能、多平台支持和强大的安全性。MySQL 拥有庞大的开源社区支持和丰富的工具生态系统。它是一个功能强大、可靠且易于使用的数据库系统，适用于各种规模和类型的应用程序开发。

在学生管理系统中有如下几个基本表：管理员表、学生登录表、学生信息表、

班级表、专业表、学院表以及重置密码信息表。一个学生只能属于一个班级，一个班级只能属于一个专业，一个专业只能属于一个学院。其中，学生信息表包含学号、姓名、性别、邮箱、电话、班级、专业和学院八个属性。

学生管理系统数据库表结构如下图所示：

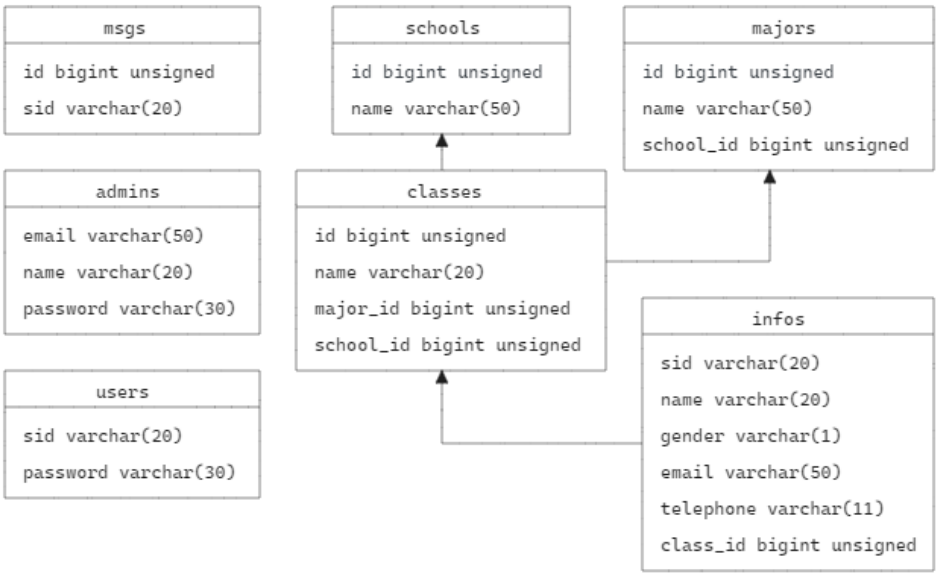

图 3 数据库表结构

2.2.2 后端项目结构分析

common 目录：有两个文件，database.go 和 jwt.go。database.go 文件用于初始化数据库连接，jwt.go 文件用于实现 JWT 认证和授权相关功能。

config 目录：用于存储项目配置信息，例如数据库连接信息、日志配置等。

controller 目录：包含了控制器逻辑，用于处理 HTTP 请求，与前端交互，并通过 model 中定义的结构体与数据库交互。

dto 目录：定义请求和响应的数据传输对象，主要用于控制器层与服务层的数据传递。

middleware 目录：定义中间件逻辑，用于处理 HTTP 请求前的一些处理，例如跨域处理、认证和授权等。

model 目录：定义应用程序中的数据模型，与数据库表进行映射。

response 目录：用于定义通用的响应结构体，例如成功响应和失败响应。

routes 目录：定义应用程序的路由，将 HTTP 请求分配给相应的控制器处理。util 目录定义一些通用的工具函数。

go.mod 文件是用于管理应用程序依赖的模块的版本控制的。

main.go 文件是应用程序的入口点。它初始化应用程序，并启动 HTTP 服务器监听端口，等待来自客户端的请求。

2.2.3 前端项目结构分析

主要分析 src 目录下项目源码部分的结构。

assets 目录：存放了一些静态资源文件，比如图片、字体等。

router 目录：存放了路由相关的代码，没有进行分组都放在 index.ts 文件。

service 目录：存放了和服务端交互的代码，比如 adminService.ts、userService.ts 等。

store 目录：存放了 Vuex 相关的代码，主要包含了 index.ts 和 module 目录，module 目录下存放了每个 Vuex 模块的代码。

utils 目录：存放了一些工具函数，比如表单验证等。

views 目录：存放了该项目中所有的页面组件，按照功能划分到不同的子目录中。比如 login 目录下包含了登录相关的组件，manage 目录下包含了数据管理相关的组件，以此类推。

App.vue 文件：项目的根组件。

main.ts 文件：Vue.js 应用的入口文件。

第三章 详细设计及实现

3.1 后端基础

3.1.1 相关配置

将相关配置信息写到 config 文件夹的 application.yml 文件中。在 server 中配置服务端的信息，注意其中 port: 1016 表示服务端口为 1016，在前端项目完成过程中需要用到；在 datasource 中配置数据源的信息，其中包含了数据库驱动名、主机地址、端口号、数据库名称、用户名、密码、字符集等信息，注意数据库名为 demo，需要自行创建名为 demo 的数据库。

在项目一开始就需要读取配置，调用函数 InitConfig()来读取。在 InitConfig()中通过 Viper 来读取配置信息。Viper 是一个 Go 语言的配置管理库，它可以帮助我们在应用程序中加载、解析和管理配置[5]。

3.1.2 连接数据库

首先安装好 Gorm 和 Gorm 的 mysql 驱动。按照上文分析的关系模型，在 model 文件夹下定义需要的 model，Admin、User 和 Info。然后在 common 包中的 database.go 文件中完成 MySQL 数据库的初始化连接函数 InitDB()。

使用 Viper 来读取应用程序的配置文件，获取数据库连接参数。然后使用这些参数连接到 MySQL 数据库，并使用 Gorm 的 AutoMigrate 函数对用户、管理员和信息三个表进行自动创建，以确保数据库的正确性。最后还提供了一个名为 GetDB 的函数，用于获取这个数据库连接实例。

main.go 中读取完文件配置后，需要调用 common.InitDB()方法获取数据库实例 db，接着注册一个延迟函数，该函数获取数据库连接实例，调用其 Close()方法关闭数据库连接。这样就可以保证在函数执行结束时，数据库连接得到了正确的关闭。

3.1.3 JWT 实现用户认证和维护登录状态

JWT（JSON Web Token）是一种用于身份验证和授权的开放标准，可以将用户身份信息封装在安全的、可靠的 JSON 对象中进行传输。JWT 的设计目的是为了在网络应用间传递声明，以便实现基于标准化的身份验证和授权。在使用 JWT 时，服务器可以生成一个包含用户身份信息的 JWT 并将其发送给客户端，客户端

在随后的每个请求中都将该 JWT 包含在请求头中发送回服务器，服务器收到 JWT 后可以验证其有效性并提取其中的用户身份信息。JWT 具有自包含性、可扩展性、跨语言支持等优点，在跨域访问和分布式系统中被广泛应用[6]。

首先安装好 JWT 包到项目中，然后在 common 包的 jwt.go 文件中实现生成和解析 JWT token 的功能。定义了一个 JWT 加密的密钥 jwtKey，并定义了 Claims 结构体，用于存储 token 中的声明信息，例如用户 ID 和过期时间等。ReleaseToken 函数用于生成 JWT token。传入一个用户 ID 参数，函数首先计算 token 的过期时间（设定有效时间是一周），然后使用声明信息初始化一个新的 JWT token，并对其进行签名。签名使用 HS256 算法和 jwtKey 密钥，最后将签名后的 JWT token 字符串返回。ParseToken 函数用于解析 JWT token。传入一个 JWT token 字符串参数，函数首先定义一个 Claims 结构体变量，用于存储从 token 中解析出的声明信息。然后调用 jwt.ParseWithClaims 函数解析 JWT token，解析过程中使用 HS256 算法和 jwtKey 密钥进行解码。最后将解析出的 JWT token 和 Claims 结构体变量一起返回。

在注册和登录完成后都需要调用 commom. ReleaseToken 发放 token，并给前端返回 token。

3.1.4 中间件

一些接口只允许认证过的用户才能够访问，所以要使用中间件在请求到达处理函数之前进行拦截。在 middleware 包中分别编写管理员和学生的中间件。

两个身份的用户的中间件基本相同，首先从请求头中获取 JWT token，验证 token 是否符合格式要求（应该以“Bearer”开头）。如果 token 为空或格式不正确，则返回 401 错误响应并终止请求处理。如果 token 格式正确，则使用共享密钥将 token 解密，验证其是否有效。如果 token 无效或验证失败，则返回 401 错误响应并终止请求处理。如果 token 验证成功，则从 token 的 payload 中获取用户 ID，然后使用该 ID 从数据库中检索出用户信息。如果用户不存在，则返回 401 错误响应并终止请求处理。否则，将用户信息添加到请求上下文中，并继续请求处理。

3.1.5 浏览器的同源策略

如果出现请求跨域，由于浏览器的同源策略的原因请求会被阻止。同源策略是一个重要的安全策略，它用于限制一个源的文档或者它加载的脚本如何能与另一个源的资源进行交互。它能帮助阻隔恶意文档，减少可能被攻击的媒介。例如，它可以防止互联网上的恶意网站在浏览器中运行 JS 脚本，从第三方网络邮件服

务（用户已登录）或公司内网（因没有公共 IP 地址而受到保护，不会被攻击者直接访问）读取数据，并将这些数据转发给攻击者。可以使用 CORS 来允许跨源访问。CORS 是 HTTP 的一部分，它允许服务端来指定哪些主机可以从这个服务端加载资源[7]。

在 middleware 包中创建一个新的中间件 CORSMiddleware.go。代码如下：

```
func CORSMiddleware() gin.HandlerFunc {
    return func(ctx *gin.Context) {
        ctx.Writer.Header().Set("Access-Control-Allow-Origin", "http://localhost:8080")
        ctx.Writer.Header().Set("Access-Control-Max-Age", "86400")
        ctx.Writer.Header().Set("Access-Control-Allow-Methods", "*")
        ctx.Writer.Header().Set("Access-Control-Allow-Headers", "*")
        ctx.Writer.Header().Set("Access-Control-Allow-Credentials", "true")
        if ctx.Request.Method == http.MethodOptions {
            ctx.AbortWithStatus(200)
        } else {
            ctx.Next()
        }
    }
}
```

CORS 是一个用于在浏览器和服务器之间传递跨源 HTTP 请求和响应头信息的机制，它通过在响应头中设置 Access-Control-Allow-Origin 、Access-Control-Allow-Methods 等字段来控制跨域请求的权限。在这个中间件中，将响应头中的 Access-Control-Allow-Origin 设置为"http://localhost:8080"，表示允许来自该源的请求。Access-Control-Max-Age 表示响应的最长有效时间，Access-Control-Allow-Methods 和 Access-Control-Allow-Headers 分别表示允许跨域请求的 HTTP 方法和请求头字段。最后，设置 Access-Control-Allow-Credentials 为 true，表示允许跨域请求携带 cookie 等凭证信息。如果请求的方法为 OPTIONS，则直接返回 200 状态码，否则继续执行后续的处理函数。

3.2 前端基础

3.2.1 环境

该系统使用的是 18.15.0 版本的 Node.js，包管理器 npm 的版本为 9.6.3，yarn

的版本为 1.22.19，使用 Vue 的脚手架@vue/cli 的版本为 5.0.8。

3.2.2 本地缓存服务与 axios 实例化

axios 是一个基于 Promise 的 HTTP 客户端，可以用在浏览器和 Node.js 中，它可用于发送异步请求并处理响应，支持发送多种类型的请求[8]。使用 axios 可以方便地与后端 API 进行交互，获取数据和更新数据。

首先需要安装 axios 和 vue-axios，并且在 main.ts 中引入它们。在 utils 文件夹下的 request.ts 文件中创建 axios 实例 service，并将其导出。首先需要创建环境变量，让 service 的 baseURL 可配置。在项目根目录下创建两个文件，.env.development 和.env.development.local，内容为 VUE_APP_BASE_URL= http://localhost:1016/api/。

接下来在 service 文件夹的 storageService.ts 文件中创建一个本地缓存服务。需要定义一些常量和两个方法：

```
// 本地缓存服务
const PREFIX = 'demo_';
// user 模块
const USER_PREFIX = `${PREFIX}user_`;
const USER_TOKEN = `${USER_PREFIX}token_`;
const USER_INFO = `${USER_PREFIX}info`;
// 储存与读取
const set = (key:any, data:any) => { localStorage.setItem(key, data); }
const get = (key:any) => localStorage.getItem(key);
```

然后回到 request.ts 文件，修改实例 service 的 headers。为了让每次请求都能动态地变化 service 实例中 headers 里的 token，需要定义一个 axios interceptor。最终代码如下：

```
const service = axios.create({
  baseURL: process.env.VUE_APP_BASE_URL,
  timeout: 1000 * 5,
});
service.interceptors.request.use((config) => {
  Object.assign(config.headers,
  { Authorization: `Bearer ${storageService.get(storageService.USER_TOKEN)}` });
  return config;
}, (error) => Promise.reject(error);
```

3.2.3 向服务器发起请求

在 service 文件夹下创建 userService.ts 和 adminService.ts，分别导入创建的 axios 实例，完善各自和服务端交互的代码。根据需求向服务器发送合适类型的请求。通过把和服务端交互的代码都放在 service 中，可以将组件和业务逻辑分离开来，使得组件更加专注于页面的展示和交互逻辑，同时也让 service 层的代码更加专注于业务逻辑和数据请求。

3.2.4 Vuex 重构

Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。通过使用 Vuex，我们可以更好地管理 Vue.js 应用程序中的数据流，实现更好的代码组织和维护。

在 store 文件夹下创建 module 文件夹，完成 user.ts 和 admin.ts。以下以 admin.ts 为例，user.ts 同理。

定义一个 adminModule。首先开启 namespaced，使模块具有命名空间。接着定义两个 state，保存了用户的 token 和管理员信息（adminInfo），从 storageService 初始化 token 和 adminInfo。mutations 对象中定义了 SET_TOKEN 和 SET_ADMININFO 两个方法，用来分别设置用户的 token 和管理员信息。这两个方法会在 actions 对象中的各个方法中被调用来修改 state 中的数据。actions 对象中定义了各种方法。这些方法会在组件中被调用，从而触发相应的操作。例如，在注册方法中，该模块会先调用 adminService 的注册接口，然后调用 adminService 的获取用户信息接口获取用户信息，最后调用 mutations 中的 SET_TOKEN 和 SET_ADMININFO 方法更新 state 中的数据。

这样，模块中的数据和方法可以被其他模块或组件所引用，从而实现对用户信息和状态的统一管理。使用时还可以通过 mapState 和 mapActions 这两个 Vuex 的辅助函数，用于简化组件中对 state 和 action 的使用。

3.3 管理员注册功能

3.3.1 注册功能的实现

前端注册页面的关键部分是表单，需要收集邮箱、用户名、密码以及确认密码四个信息，其中用户名不是必选项。需要四个输入框，其值通过 v-model:value

指令绑定到 admin 对象的对应属性，并进行相关表单数据验证。然后调用 adminModule 中的注册方法与服务端进行交互。

后端在 controller 包中的 AdminController.go 文件中完成处理管理员注册请求的函数。该函数主要需要完成以下功能：首先从请求中获取用户提交的注册信息（邮箱，名称，密码）。对密码进行长度验证，如果不足 6 位返回错误信息，如果没有提供名称，则生成一个 10 位大随机字符串作为名称。接着检查邮箱是否已经被注册，如果已被注册，返回错误信息。验证通过后将相关信息写入后端数据库的 admins 表中，并返回前端 token 与成功信息。

3.3.2 密码加密

目前密码在数据库中是以明文的形式保存的。但是，在数据库中将密码以明文的形式保存是一种非常不安全的做法，如果数据库的安全性遭受攻击，攻击者可以很容易地获取用户的密码，从而导致用户的账户被盗用或信息泄露。为了保护用户的隐私和安全，通常需要将密码进行加密或哈希处理后再保存在数据库中。这样即使数据库遭受攻击，攻击者也无法轻易地获取用户的密码，从而提高了用户的安全性。

该系统使用 bcrypt 库中的 GenerateFromPassword 函数将明文密码 password 转换成一个不可逆的哈希值，然后将该哈希值保存在数据库中。在用户登录时，通过 bcrypt 库中的 CompareHashAndPassword 函数将用户输入的密码与数据库中保存的哈希值进行比对，如果返回值为 nil，则认为用户输入的密码是正确的，否则认为是错误的，返回密码错误的错误信息。

3.4 登录与登出功能

首先完成 adminModule 和 userModule 中的 login 和 logout 方法。登录首先清除 Vuex 中存储的 token 和用户信息，以确保登录前数据的干净和一致性。接着，调用封装好的 login 函数，发送 POST 请求以验证登录信息并获取 token，并通过 Promise 实例进行异步处理。在登录成功后，是通过 Vuex 中的 commit 方法，将 mutations 的值更新为后端返回的 token 和用户信息。最后，通过 resolve 将异步处理结果返回给调用者。登出时，只需要清除本地存储中的用户 token 和用户信息。

然后完善登录页面表单的 handleFinish 函数。在登录表单中输入信息后，点击登录按钮时，会使用 store.dispatch 方法调用相关 Module 中的 login 方法与服务端进行交互。

后端在 controller 包中完成处理管理员和学生的登录请求的函数。登录函数主要需要完成以下功能：首先从请求参数中获取邮箱（或学号）和密码，进行数据验证，如果密码长度小于 6 则返回密码长度不能少于 6 位的错误信息。接着，从数据库中查找该邮箱（或学号）对应的用户，如果不存在则返回用户不存在的错误信息。如果存在则比较输入的密码和数据库中保存的密码是否一致，如果不一致则返回密码错误的错误信息。如果密码一致则登录成功，向前端返回 token 与成功信息。如果登录成功，前端页面将弹出一个登录成功的模态框并跳转到主页，否则将通过 notification 组件显示错误信息。

登出按钮在侧栏，调用相关 Module 中的 logout 方法然后跳转首页即可。

3.5 密码（或用户名）修改

点击左侧栏的“设置”栏目可以进入设置界面进行密码（或用户名）的修改。该页面显示一个表单，允许用户编辑他们的密码。用户信息前端验证通过，提交表单时将调用 Module 中的相关方法与服务端进行交互。服务端进一步进行数据验证，加密密码，并将改动保存到数据库。如果在验证或保存过程中出现错误，将使用通知组件显示错误消息。如果操作成功，将重定向到管理页面。

3.6 忘记密码

3.6.1 学生申请重置密码

前端登录页面下方有忘记密码的提示，点击该链接进入找回密码页面。该页面有一个表单收集学号信息，管理员可以看到申请重置密码的学生的学号然后进行处理。相关表单数据验证通过后调用 Module 中的方法与服务端进行交互。后端在 controller 包中的 MessageController.go 文件中完成处理申请重置密码请求的函数。该函数主要需要完成以下功能：首先从请求中获取用户提交的学号。对学号进行验证是否存在，如果不存在返回错误信息。并且验证 msgs 表中是否已经存在该记录，如果已经存在就返回“申请尚未处理”的错误信息。数据验证通过后，将学号写入后端数据库的 msgs 表，并返回前端成功信息。

3.6.2 管理员处理申请

重置密码页面有一个表格，一列是申请重置密码的学生的学号，另一列是重

置按钮。点击重置按钮将会把学号作为参数传给 Module 中的相关方法与服务端进行交互。后端将修改 users 表中相应学号的元组的密码为默认密码，并且删除 msgs 表中的该条记录。

3.7 学生模块

3.7.1 信息展示

调用 userService 中的 info 方法向后端发送 GET 请求。后端服务器根据上下文返回当前登录用户的相关信息。前端将返回的信息展示在页面中。

3.7.2 信息修改

点击左侧栏中的“修改信息”栏目进入修改个人信息的页面。窗口显示一个表单，允许用户编辑他们的个人信息，包括姓名、性别、电子邮件和电话号码（班级、专业、学院只能由管理员编辑）。用户前端信息验证通过，提交表单时将调用 userModule 中的相关方法与服务端进行交互。后端进一步数据验证通过后将改动保存到数据库的 infos 表中，并返回成功信息。如果在验证或保存过程中出现错误，前端页面将使用通知组件显示错误消息。如果操作成功，将重定向到个人信息页面。

3.8 管理员后台页——学生信息

3.8.1 数据展示与编辑删除

首先完成用于显示用户数据的表格。Table 组件定义了两个子组件——Delete 和 Edit，它们作为每行表格中的操作按钮呈现。用户数据对象由 data.ts 文件导入，该文件定义了表格的列和初始数据。userData.columns 和 userData.list 属性分别作为 a-table 组件的列和数据源。

data.ts 定义了一个名为 userData 的对象，它具有以下属性：columns：表示表格列的对象数组；list：表示要在表格中显示的数据的响应式数组对象；selectedRowkeys：一个数组，用于跟踪当前选定行的键。此外，还定义了一个名为 GetData()的函数，该函数返回一个 promise，使用 adminService.show()方法从服务器获取数据，并返回符合 UserInfo 接口的对象数组。最后，调用 GetData()函数来使用从服务器获取的数据填充 userData.list 属性。

Edit 组件定义了一个用于编辑用户账户信息的按钮。当点击按钮时会触发一个

窗口。窗口显示一个表单，允许用户编辑他们的个人信息，包括学校、姓名、性别、电子邮件和电话号码。用户信息验证通过，提交表单时将调用 userModule 中的方法与服务端进行交互。后端进一步数据验证通过后将把改动保存到数据库 infos 表中，并返回成功信息。如果在验证或保存过程中出现错误，前端将使用通知组件显示错误消息。如果操作成功，将重定向到管理页面。

Delete 组件定义了一个用于删除用户账户的按钮。当点击按钮时，使用 ant-design-vue 库显示一个确认对话框。如果用户确认删除，则会使用 store.dispatch 方法调用 userModule/delete 来从后端服务器删除 users 表和 infos 表中的相应记录，并显示一个成功通知。然后通过调用 GetData()函数刷新用户数据，并使用新数据更新 userData 响应对象。最后，使用路由器导航回管理页面。Table 组件中还有一个 removeBatch 方法被用作 Delete 组件发出的 remove 事件的回调函数。它从 userData.list 数组中删除相应的用户数据，并相应地更新表格视图。

3.8.2 添加用户

Create 组件定义一个添加用户到系统的表单。当点击“添加用户”按钮时，会显示一个模态对话框，其中包含一个输入字段用于输入用户的“学号”（学生编号）。数据验证通过后，当用户在模态对话框中点击“确定”按钮时，将调用 handleOk 方法，该方法会使用 store.dispatch 方法调用 userModule/register 以向服务器注册用户。后端在 users 表和 infos 表中都新增相应条目，并返回成功信息。如果请求成功，前端则显示成功消息并刷新用户数据。如果出现错误，则显示错误通知。

3.8.3 按学号搜索用户

在学生信息页面添加一个搜索框，当提交搜索输入时，调用 onSearch 函数，触发使用 searchData 函数搜索输入值。searchData 是一个 Promise，它通过调用 adminService 中的 search 方法来实现搜索用户信息，参数 word 是搜索关键字。后端根据参数 word 在 infos 表中的 sid 字段中进行匹配搜索，返回搜索结果和成功信息。前端将返回的用户数据设置为 userData.list，并将路由推送到具有新时间戳查询参数的相同路径。

3.8.4 按不同条件筛选用户

直接利用 ant-design-vue 库的表格中相关参数的设置，可以实现在表头处对数据进行筛选。拷贝网站上的代码然后进行修改。可以通过输入框的方式对姓名字段和班级字段进行筛选。因为该系统不能对专业和学院进行修改，所以对于学生

和管理员来说专业和学院都是固定的，可以在表格参数中录入所有的学院与专业后，采用下拉选择的方式进行筛选。

3.9 管理员后台页——班级信息

3.9.1 数据展示与编辑删除

首先完成用于显示班级数据的表格。CTable 组件定义两个子组件——CDelete 和 CEdit，它们作为每行表格中的操作按钮呈现。用户数据对象由 classdata.ts 文件导入，该文件定义了表格的列和初始数据。classData.columns 和 classData.list 属性分别作为 a-table 组件的列和数据源。

classdata.ts 定义了一个名为 classData 的对象，它具有以下属性：columns：表示表格列的对象数组；list：表示要在表格中显示的数据的响应式数组对象；selectedRowkeys：一个数组，用于跟踪当前选定行的键。此外，还定义了一个名为 GetClassData()的函数，该函数返回一个 promise，使用 adminService.showclass()方法从服务器获取数据，并返回符合 ClassInfo 接口的对象数组。最后，调用 GetClassData()函数来使用从服务器获取的数据填充 classData.list 属性。

CEdit 组件定义了一个用于编辑班级信息的按钮。当点击按钮时会触发一个窗口。窗口显示一个级联选择器，允许管理员编辑班级的专业和学院。提交表单时将调用 adminModule/updateclass 与服务端进行交互。后端进一步数据验证通过后将把改动保存到数据库 classes 表中，并返回成功信息。如果在验证或保存过程中出现错误，前端将使用通知组件显示错误消息。如果操作成功，将重定向到管理页面。

CDelete 组件定义了一个用于删除用户账户的按钮。当点击按钮时，使用 ant-design-vue 库显示一个确认对话框。如果用户确认删除，则会使用 store.dispatch 方法调用 adminModule/deleteclass 来从后端服务器删除 classes 表中的相应记录，并显示一个成功通知。然后通过调用 GetClassData()函数刷新用户数据，并使用新数据更新 classData 响应对象。最后，使用路由器导航回管理页面。Table 组件中还有一个 removeBatch 方法被用作 Delete 组件发出的 remove 事件的回调函数。它从 classData.list 数组中删除相应的用户数据，并相应地更新表格视图。

3.9.2 创建班级

CCreate 组件定义一个创建班级的表单。当点击“创建班级”按钮时，会显示一个模态对话框，其中包含一个输入字段用于输入班级的编号。数据验证通过后，当用户在模态对话框中点击“确定”按钮时，将调用 handleOk 方法，该方法会使用 store.dispatch 方法调用 adminModule/addclass 以向服务器注册用户。后端在 classes 表中新增相应条目，并返回成功信息。如果请求成功，前端则显示成功消息并刷新用户数据。如果出现错误，则显示错误通知。

3.9.3 按不同条件筛选班级

直接利用 ant-design-vue 库的表格中相关参数的设置，可以实现在表头处对数据进行筛选。拷贝网站上的代码然后进行修改。可以通过输入框的方式对班级字段进行筛选。因为该系统不能对专业和学院进行修改，所以对于学生和管理员来说专业和学院都是固定的，可以在表格参数中录入所有的学院与专业后，采用下拉选择的方式进行筛选。

第四章 测试

4.1 系统主页

开启前后端服务后即可访问该系统。访问网址 http://localhost:8080/出现系统主页。此时处于未登录状态，首页只有顶部导航栏，在登录以后会出现侧栏。

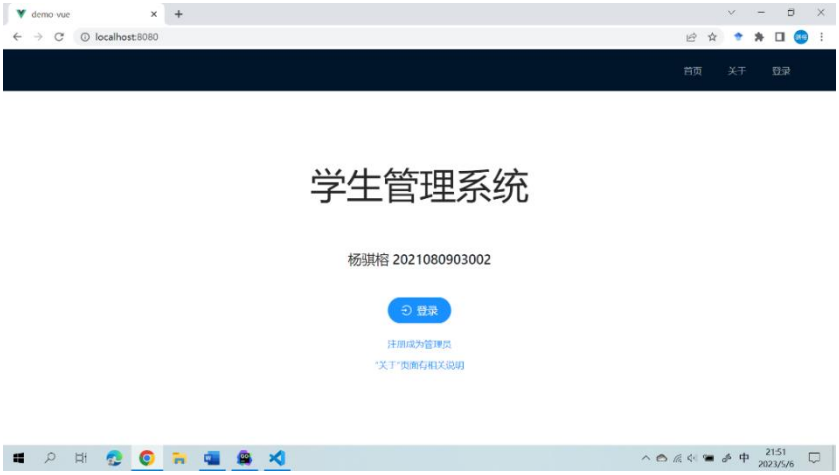

图 4 系统主页

4.2 管理员注册

点击首页“注册成为管理员”链接按钮，跳转到管理员注册页面。

如果数据输入不符合规则，则会进行提示，且无法提交。用户名非必填，若为空自动生成十位随机字母。若注册的邮箱已存在，虽然可以提交表单，但会有后端的错误消息反馈。

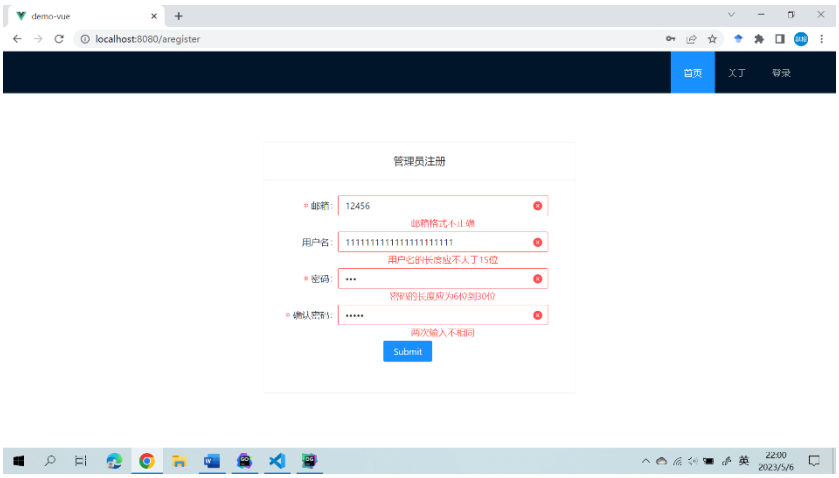

图 5 管理员注册-错误输入 1

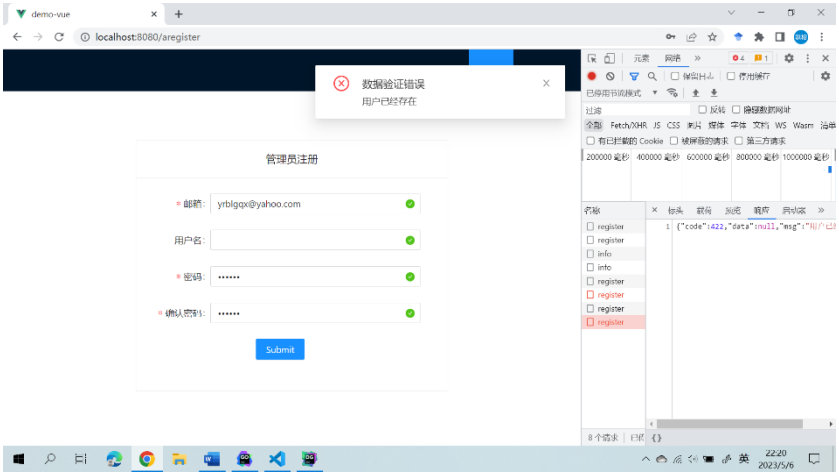

图 6 管理员注册-错误输入 2

正确填写信息后就可以正常提交，提交后数据库出现新纪录。注册后直接进入已登录状态并跳转到首页。

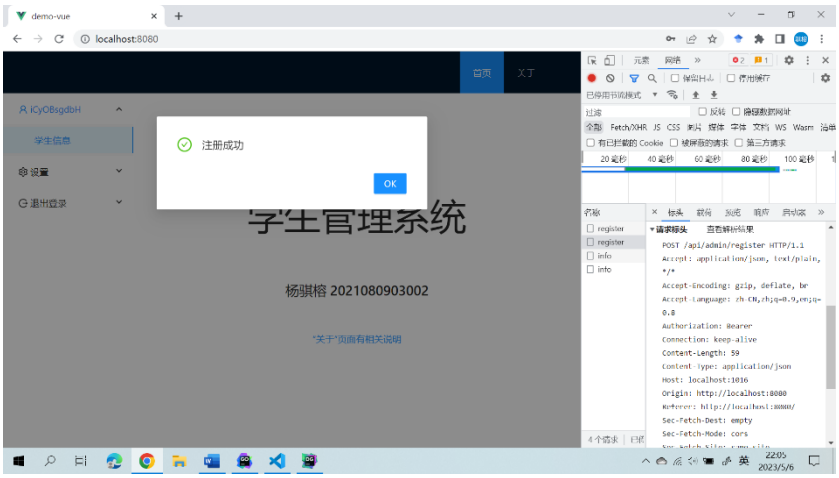

图 7 管理员注册成功 1

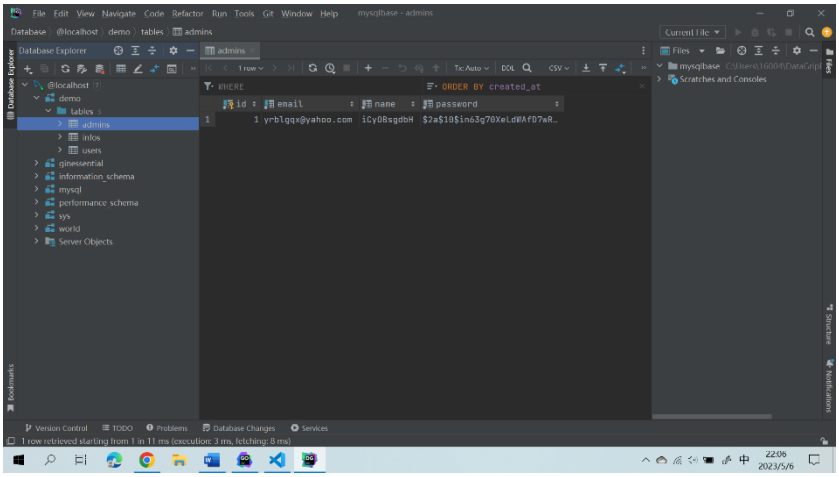

图 8 管理员注册成功 2

4.3 登录

点击首页或者顶部导航栏的“登录”跳转到登录页面。登录页面可以选择学生身份或管理员身份，以下以管理员身份为例。

如果数据输入不符合规则，则会进行提示，且无法提交。

用户不存在或者密码不正确，提交后会收到错误反馈。

正确输入后就可以正常登录。

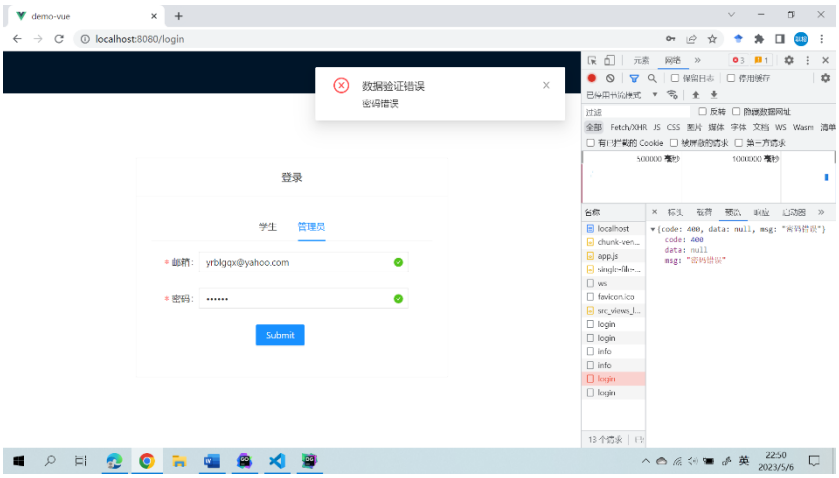

图 9 管理员登录-错误输入

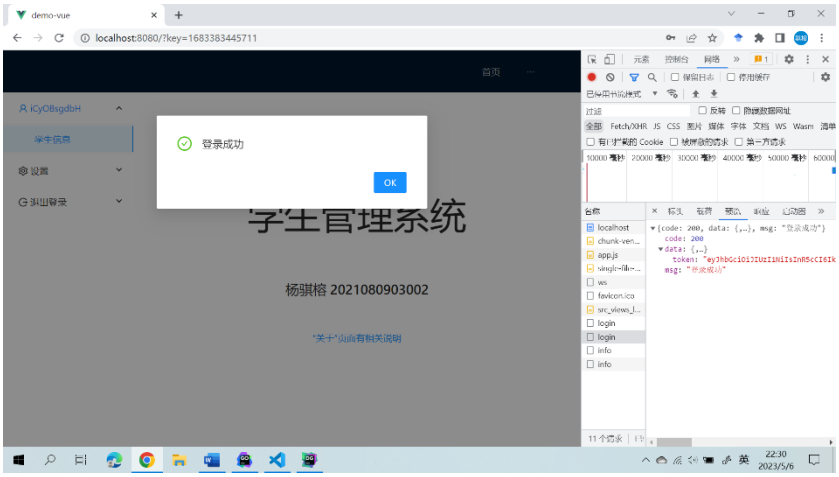

图 10 管理员登录成功

4.4 学生个人主页

学生登录以后页面会出现左侧栏，即可通过点击侧栏的“个人空间”查看自己的信息。

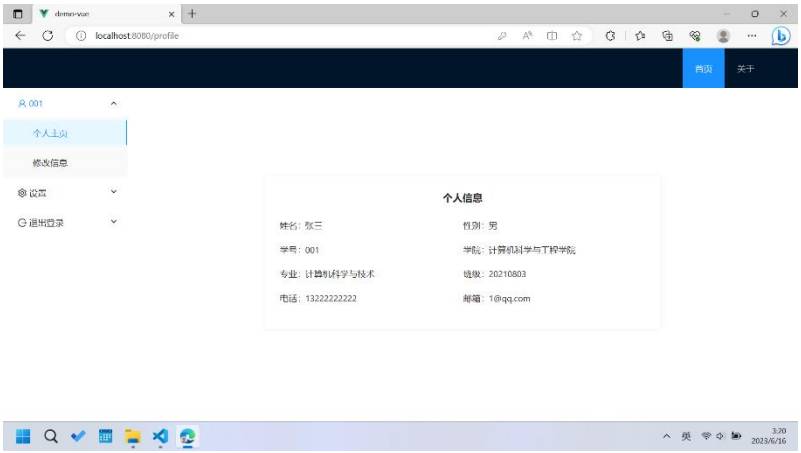

图 11 学生查看个人信息

4.5 学生修改个人信息

点击左侧栏的“修改信息”跳转到信息修改页面。修改成功跳转到个人主页。

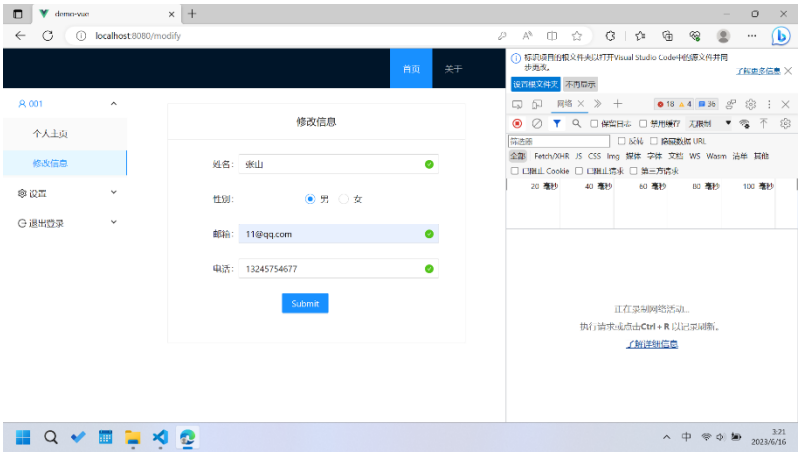

图 12 学生修改信息页

图 13 学生修改信息成功

4.6 设置

4.6.1 管理员修改用户名或密码

管理员登录以后通过侧栏进入设置页面，填写表单即可以修改用户名或密码。点击最下方的注销账号可以删除自己的账号。修改后退出登录返回登录页面。

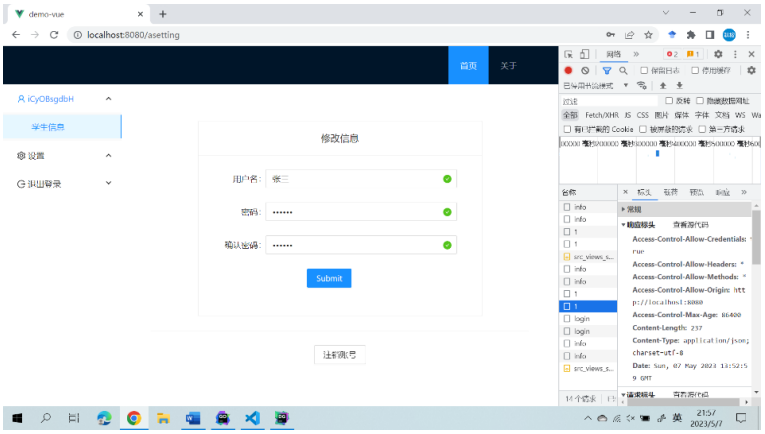

图 14 管理员设置页

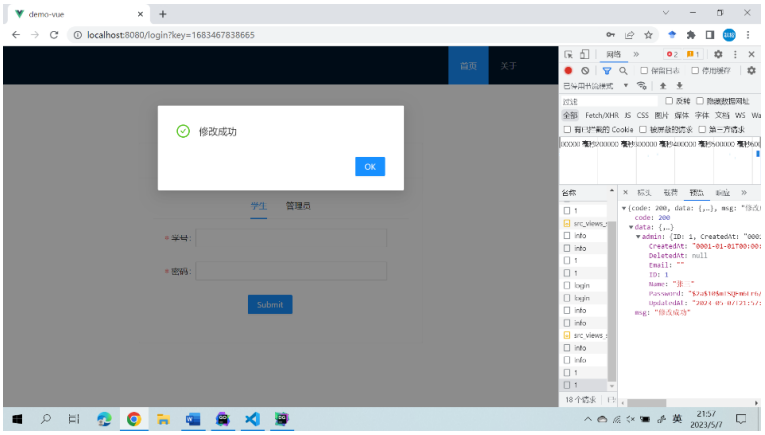

图 15 管理员设置修改成功

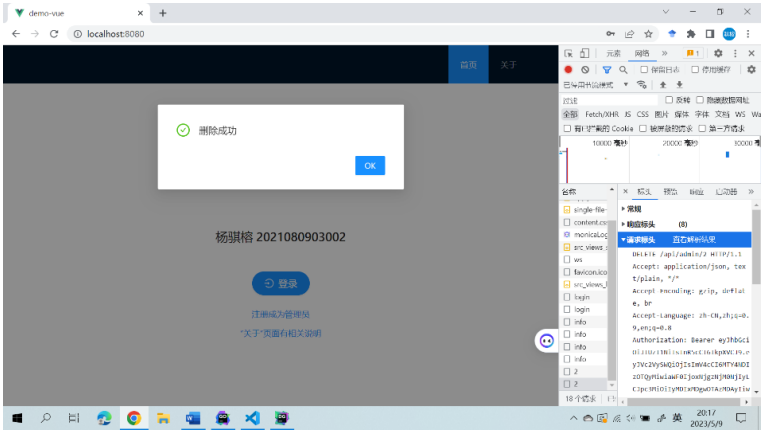

图 16 管理员删除账号

4.6.2 学生修改密码

学生登录以后通过侧栏进入设置，按规则填写表单即可以修改自己的密码。

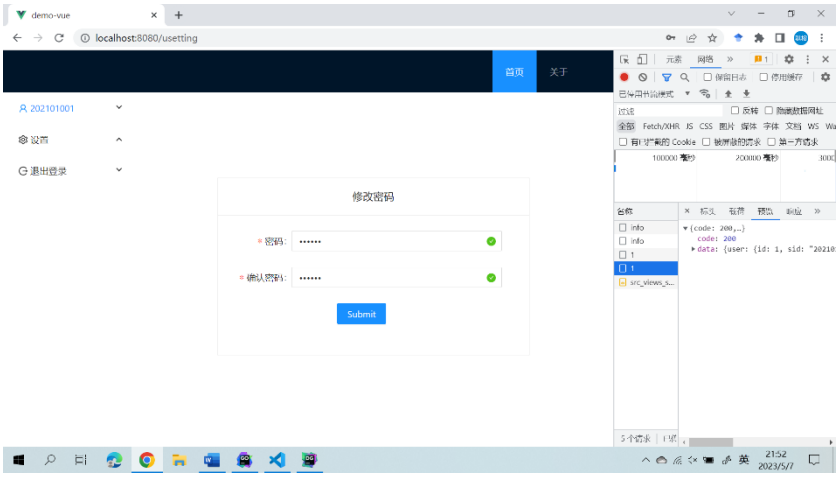

图 17 学生设置修改密码

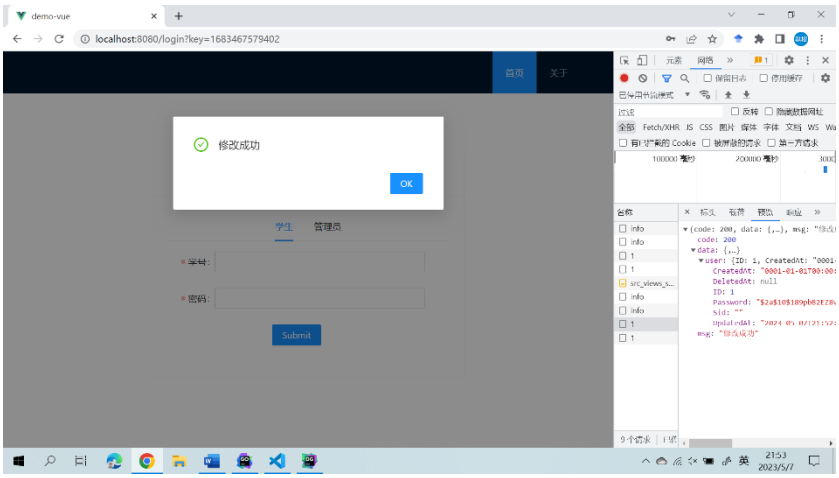

图 18 学生修改密码成功

4.7 管理员后台页——学生信息

管理员登录以后可以通过侧栏的“学生信息”进入学生信息详情页。

4.7.1 信息展示

进入学生信息页即可查看所有学生信息。

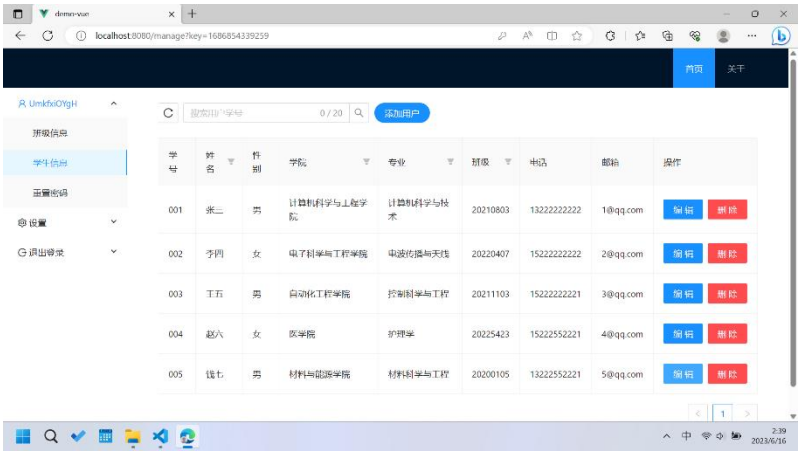

图 19 管理员后台页

4.7.2 创建用户

点击“创建用户”按钮，在表单中填写添加的学生学号，数据验证通过后即可创建学生用户。

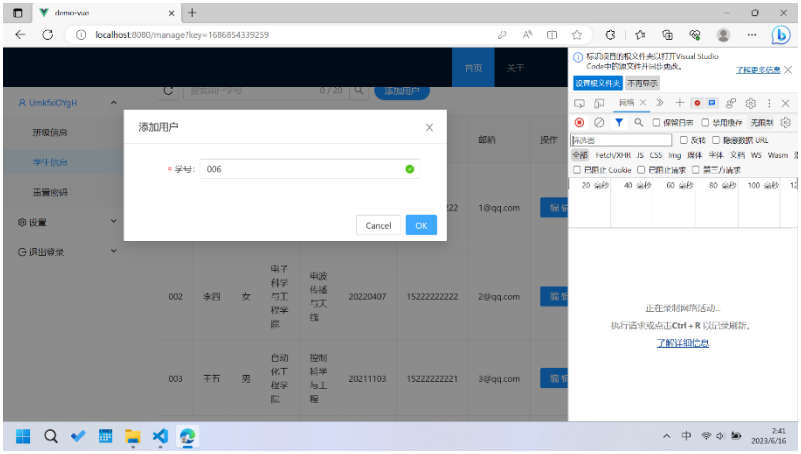

图 20 管理员添加用户

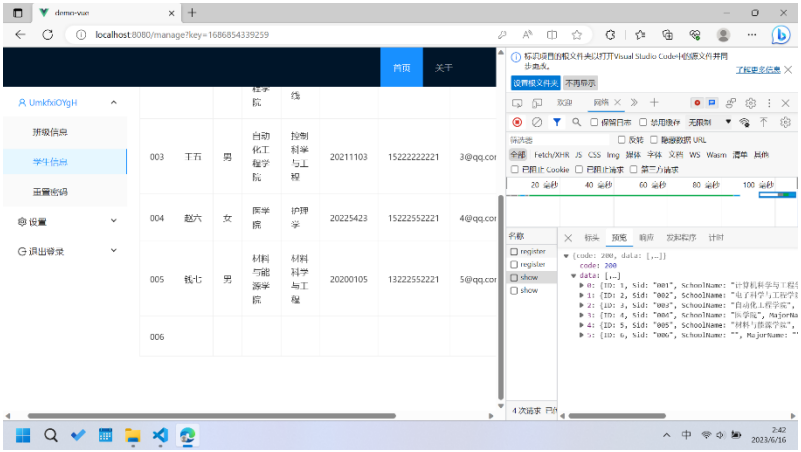

图 21 管理员添加用户成功

4.7.3 按学号查找用户

在搜索框中输入要查找的学生的学号，下方表格中即可展示搜索结果。

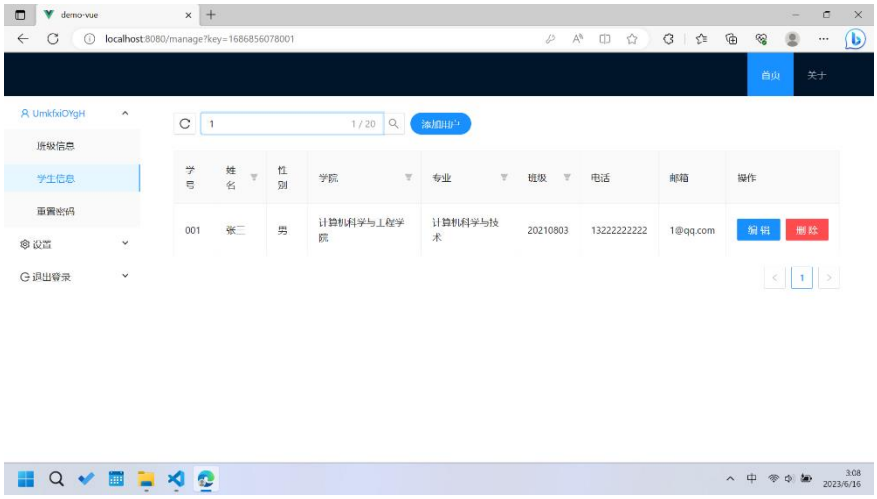

图 22 按学号查找用户

4.7.4 按不同条件筛选用户

可以点击表头的筛选图标按照不同的条件对班级进行筛选。不同列之间的条件取交集，下拉选择可以多选。

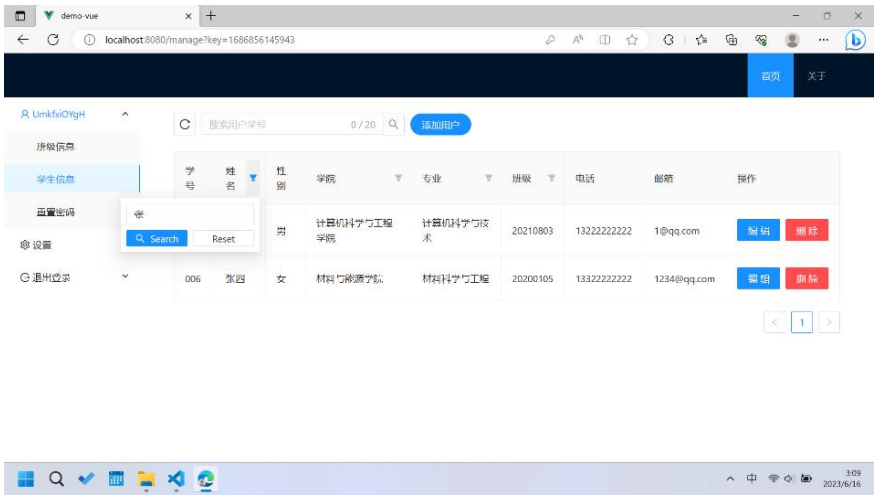

图 23 按姓名筛选学生

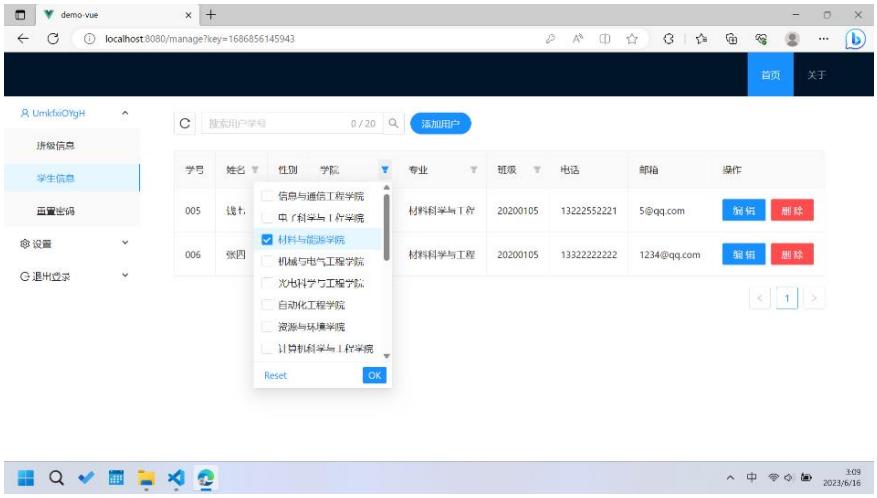

图 24 按学院筛选学生

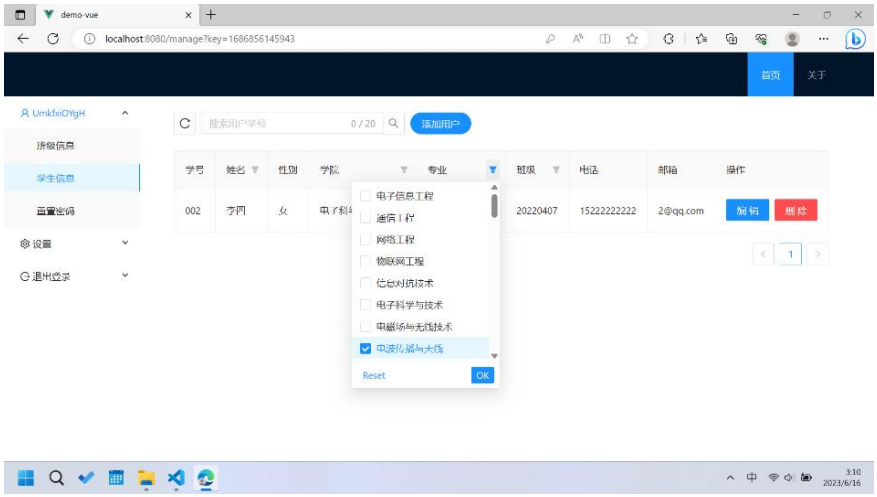

图 25 按专业筛选学生

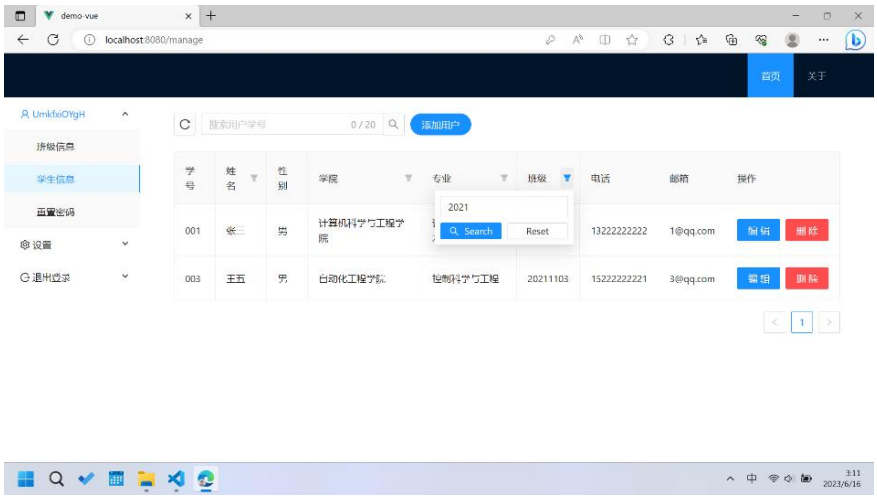

图 26 按班级筛选学生

4.7.5 编辑用户

点击表格最后一列“操作”的“编辑”按钮，按规则填写好表单后进行提交，即可修改学生的信息。

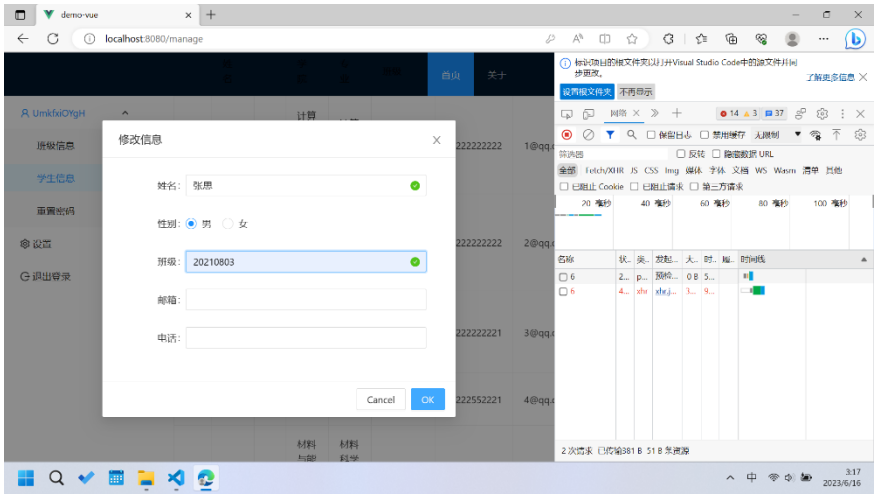

图 27 编辑学生信息

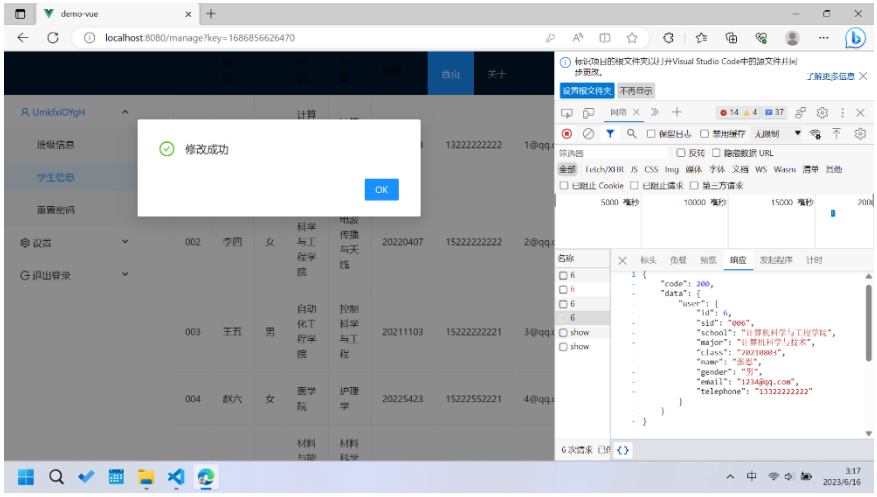

图 28 编辑学生信息成功

4.7.5 删除用户

点击表格最后一列“操作”的“删除”按钮，即可删除学生。

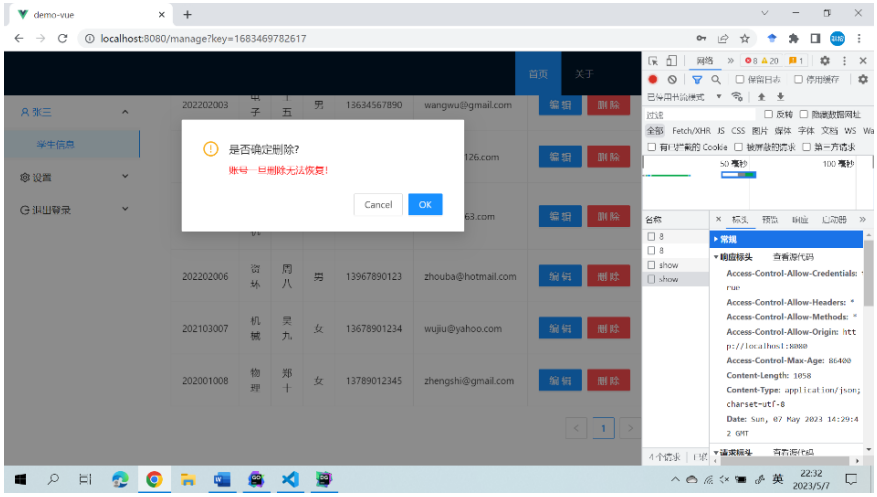

图 29 删除学生

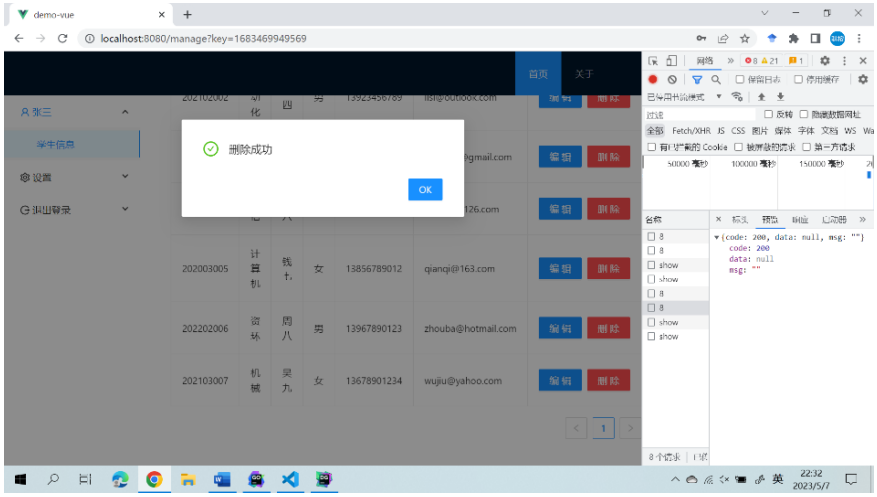

图 30 删除成功

4.8 管理员后台页——班级信息

管理员登录以后可以通过侧栏的“班级信息”进入班级信息详情页。

4.8.1 信息展示

进入班级信息页即可查看所有班级信息。

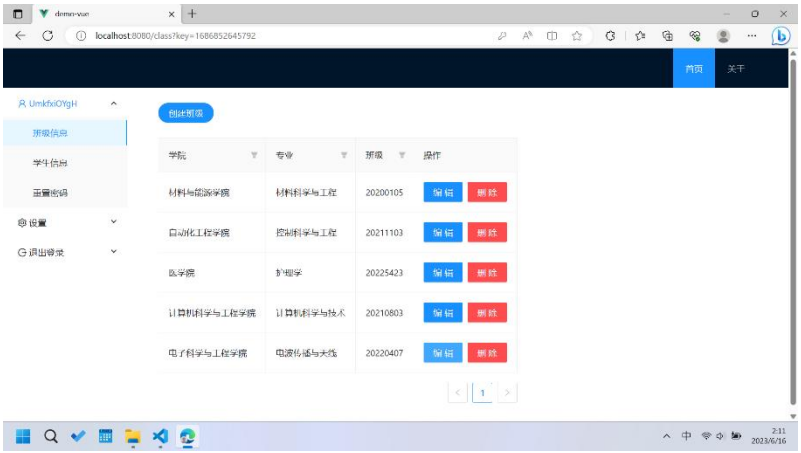

图 31 班级信息

4.8.2 创建班级

点击“创建班级”按钮，在表单中填写添加的班级编号，数据验证通过后即可创建班级。

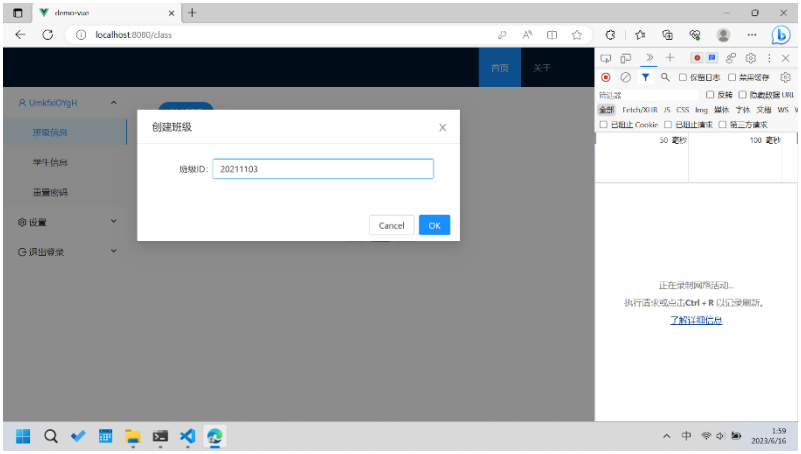

图 32 管理员创建班级

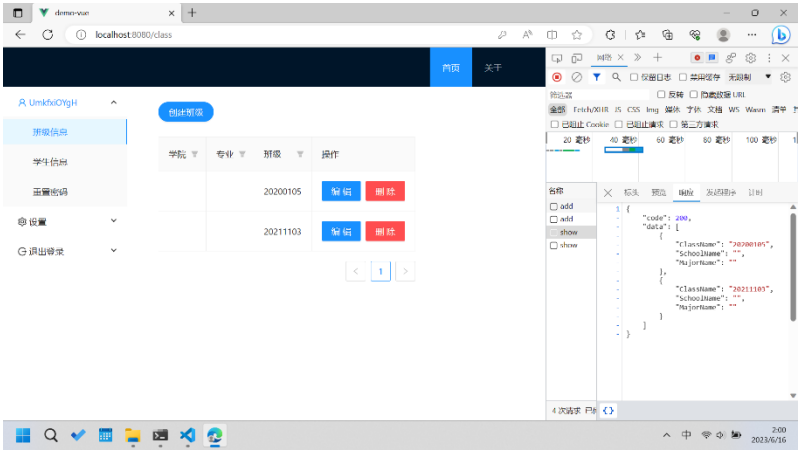

图 33 管理员创建班级成功

4.8.3 按不同条件筛选班级

可以点击表头的筛选图标按照不同的条件对班级进行筛选。不同列之间的条件取交集，下拉选择可以多选。

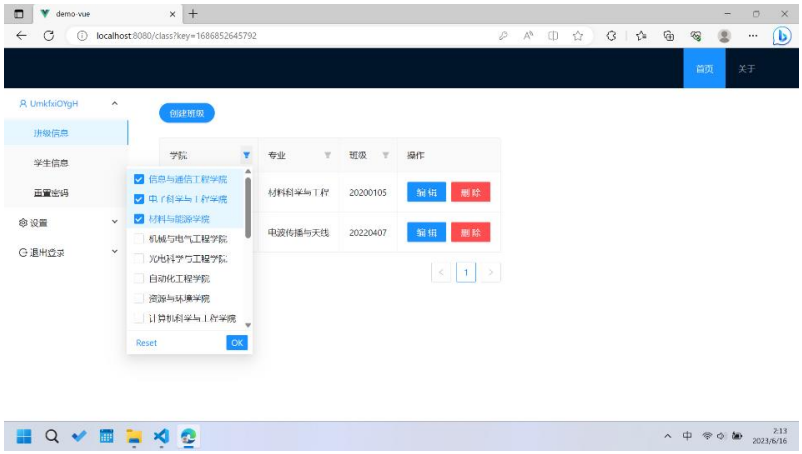

图 34 按学院筛选班级

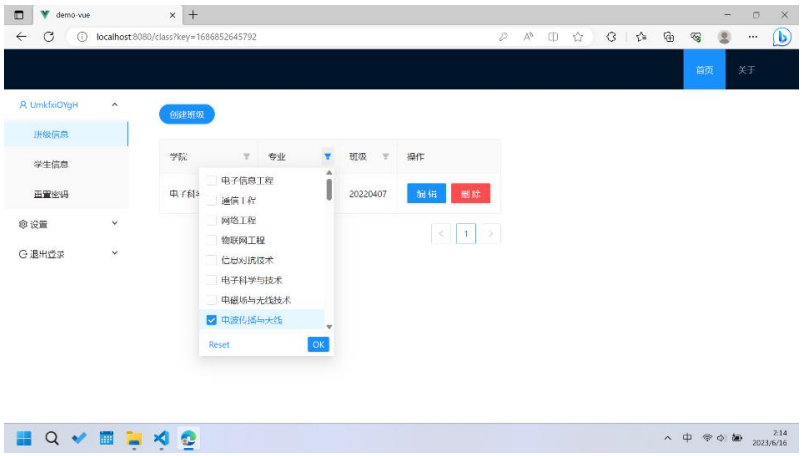

图 35 按专业筛选班级

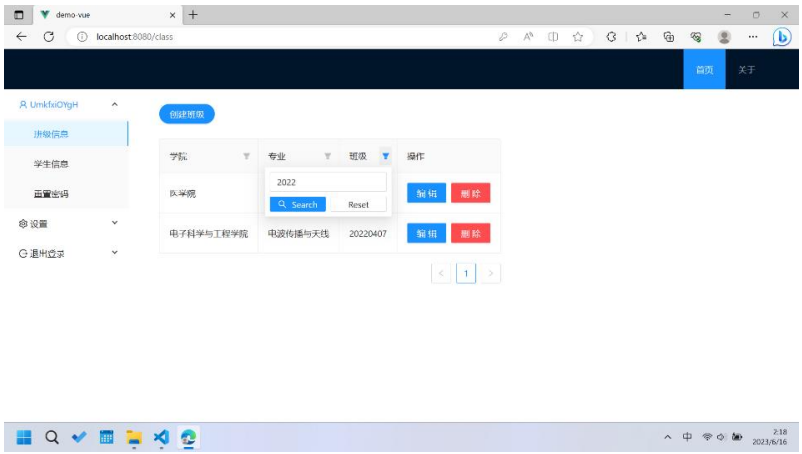

图 36 按班级编号筛选

4.8.4 编辑班级

点击表格最后一列“操作”的“编辑”按钮，选择学院或专业后进行提交，即可修改班级的信息。班级信息修改后，在该班级中学生的专业或学院也会发生改变。

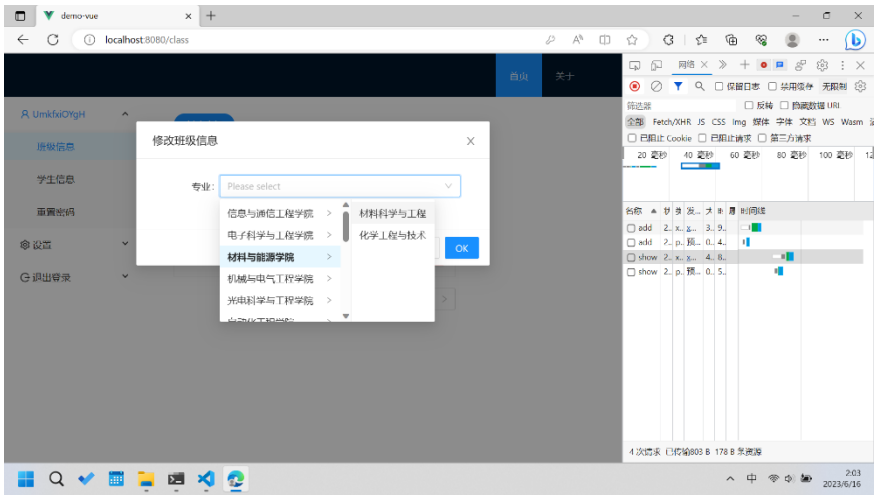

图 37 编辑班级信息

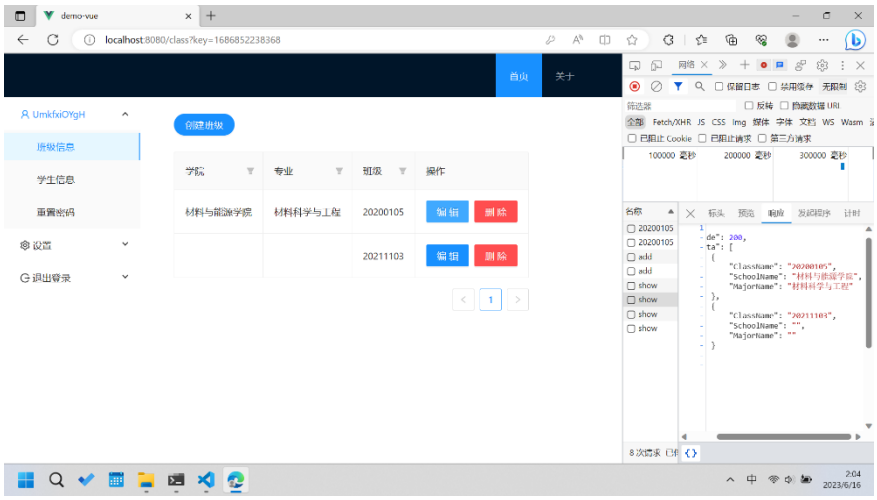

图 38 编辑班级信息成功

4.8.5 删除班级

点击表格最后一列“操作”的“删除”按钮，即可删除班级。

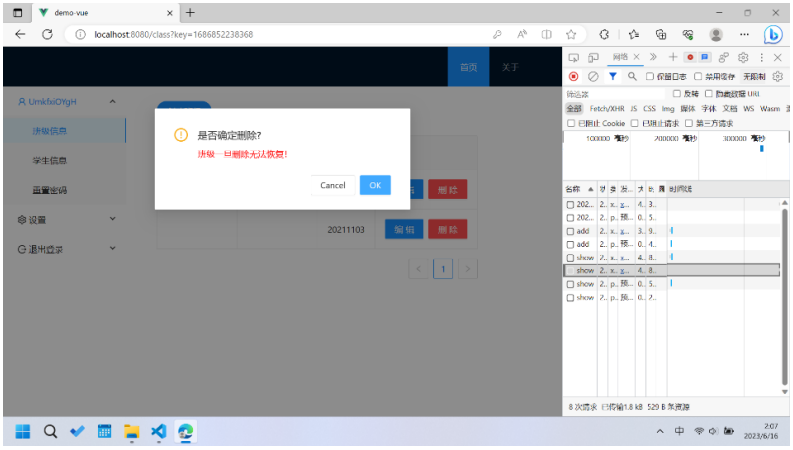

图 39 删除班级

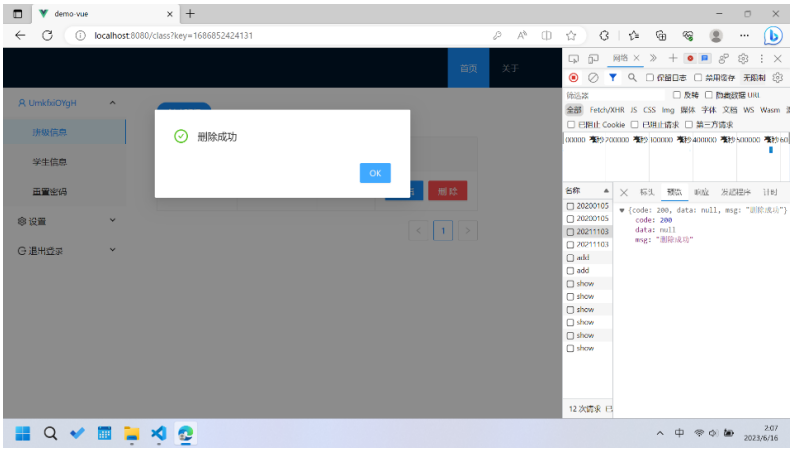

图 40 删除班级成功

4.9 忘记密码

学生忘记密码后可以通过登录页面下方的忘记密码链接跳转到忘记密码页面，填写学号，向管理员申请重置密码。

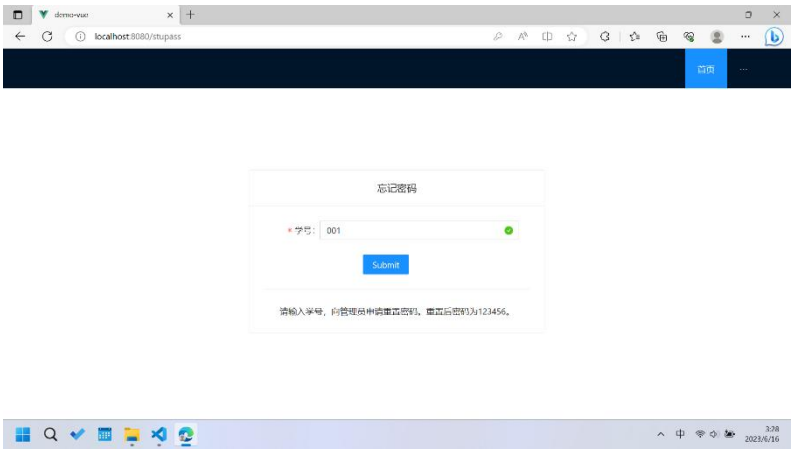

图 41 学生申请重置密码

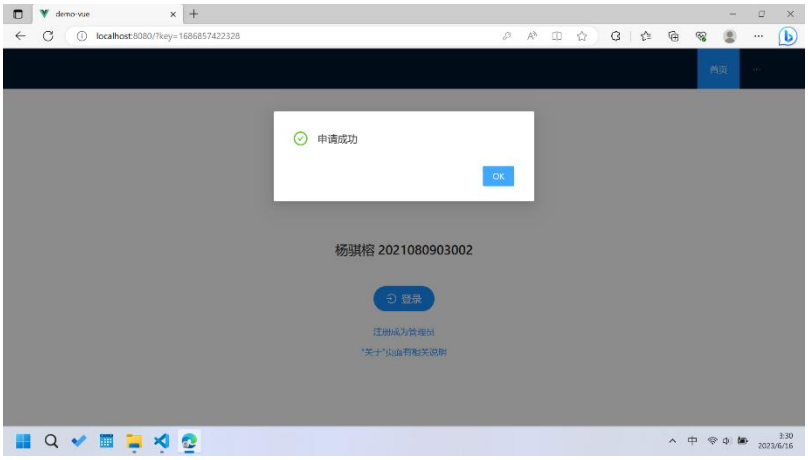

图 42 申请成功

管理员可以在重置密码页面看到学生的申请。

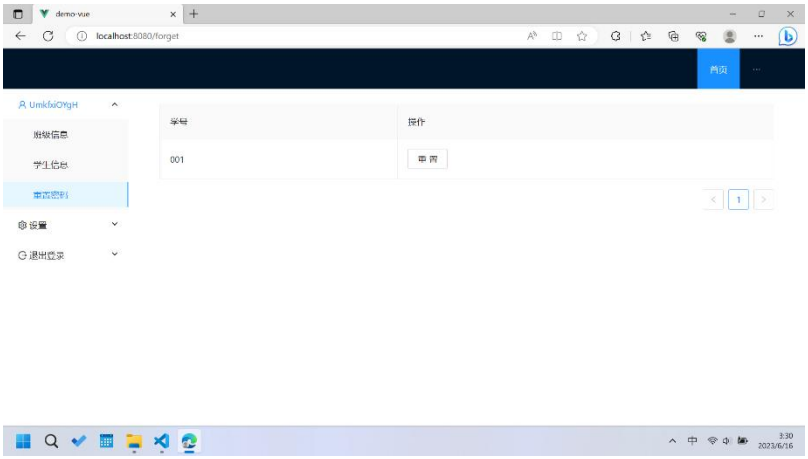

图 43 查看学生申请

点击重置按钮处理该申请。该学生的密码被设置为默认密码 123456。

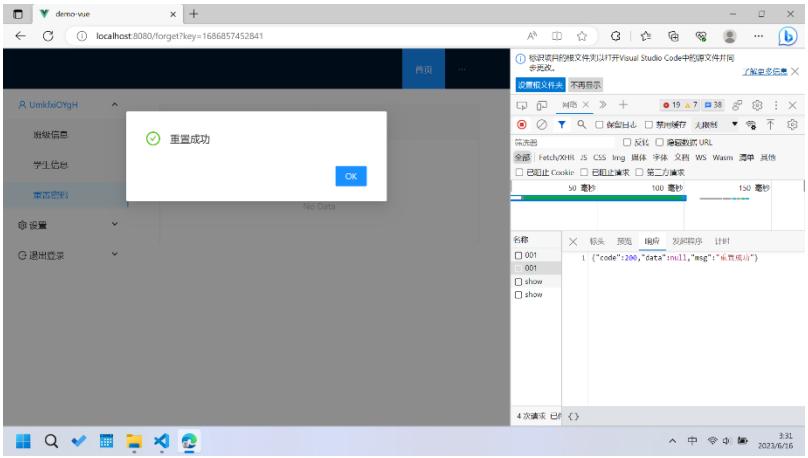

图 44 重置成功

参考文献

[1] 蔡长安，王琪. 基于 B/S 模式的学生信息管理系统设计与实现[J]. 计算机工程与设计, 2006, 27(14): 2585-2587.

[2] Gin Web Framework[EB/OL]//Gin Web Framework. https://gin-gonic.com/zh-cn/docs/.

[3] GORM Guides[EB/OL]//GORM. https://gorm.io/docs/index.html.

[4] Vue.js[EB/OL]. https://cn.vuejs.org/.

[5] FRANCIA S. spf13/viper[CP/OL]. https://github.com/spf13/viper.

[6] 范展源，罗福强. JWT 认证技术及其在 WEB 中的应用[J]. 数字技术与应用，2016，0(2): 114-114.

[7] 何良，方勇，方昉，等. 浏览器跨域通信安全技术研究[J]. 信息安全与通信保密，2013(4): 59-61.

[8] axios[CP/OL]. axios, 2023. https://github.com/axios/axios.