华北水利水电大学 信息工程学院

2023-2024 学年第 2 学期

《计算与软件工程 III》

结课作业

题目：基于 SSM 的教务系统后端的设计与实现

专业班级：2021184

学　　号：202107927

姓　　名：康问樵

摘要 基于 SSM 的教务系统后端的设计与实现

本项目旨在设计和实现一个基于 SSM（Spring、Spring MVC、MyBatis）的教务系统后端。该系统涵盖了教务管理的各个方面，包括学生管理、教师管理、班级管理、课程管理、考试管理和成绩管理等功能模块。通过利用 SSM 框架，系统能够实现高效的数据库操作、清晰的业务逻辑分层和灵活的扩展性。

系统采用了经典的三层架构设计，包括表现层、业务逻辑层和数据访问层。表现层负责处理用户请求和响应，主要使用 Spring MVC 框架来实现；业务逻辑层负责实现具体的业务逻辑，通过 Spring 框架进行管理；数据访问层则负责与数据库的交互，采用 MyBatis 框架进行 ORM（对象关系映射）操作。

在数据库设计方面，系统包含多个主要表，如学院表（college）、教务表（dean）、专业表（speciality）、班级表（class）、学生表（student）、教师表（teacher）、学期表（term）、课程表（course）、考试表（exam）和成绩表（student_course_score）。这些表通过外键和索引进行关联，以确保数据的完整性和查询的高效性。

系统采用 Spring 框架来管理业务逻辑，通过 Spring MVC 框架来处理 HTTP 请求，并使用 MyBatis 框架进行数据库操作。系统还集成了 Element Plus 组件库来实现前端界面，通过 Vue.js 框架进行前端开发。系统设计了详细的接口文档和数据库表结构，并实现了数据访问层的接口和具体实现类，以确保数据操作的高效性和可靠性。

通过采用 SSM 框架，本教务系统后端实现了清晰的架构设计和高效的数据管理。系统功能完善，能够满足教务管理的各项需求。未来，可以根据实际需求进一步扩展系统功能，如增加权限管理模块、实现更多的数据统计和分析功能等，以提升系统的实用性和用户体验。项目 GitHub 地址

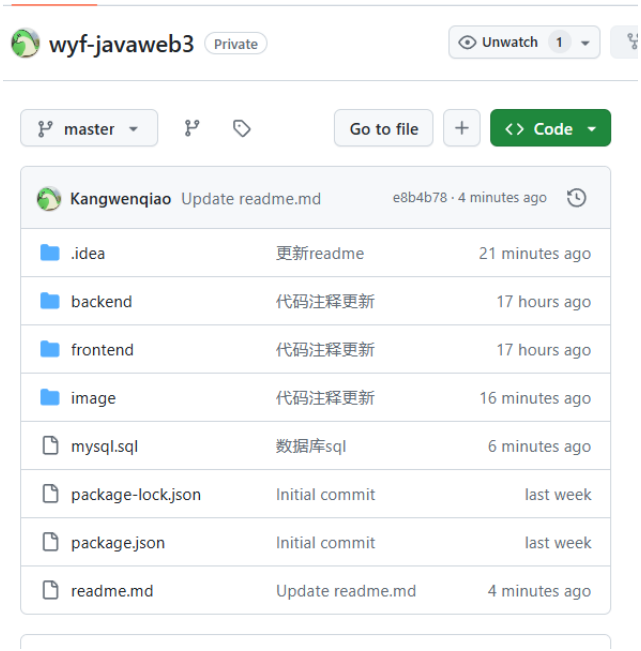

图 1 项目 2024.6.28

目录

第一章 开发背景

开发教育管理系统解决数据冗余、效率低下和实时性差等问题，提高管理效率、数据一致性和安全性，支持教学管理和决策分析。

1.1 问题概述

现代高等教育机构面临着日益复杂的管理需求，尤其是在学生、教师和课程管理方面。传统的手工管理方式已难以应对庞大数据处理和实时信息共享的需求。具体而言，现有系统面临以下几个主要问题：

数据冗余和不一致性：手工管理或使用多个独立系统容易导致数据冗余和不一致性。比如，学生信息、课程安排和成绩记录可能存储在不同的系统中，难以统一管理和查询。

效率低下：手工处理大量的学生和课程数据，不仅耗费时间，而且容易出错。特别是在课程安排、成绩录入和教师评估等方面，效率问题尤为明显。

实时性差：传统的管理系统缺乏实时数据更新和共享能力，信息传递滞后，影响决策的及时性和准确性。

数据安全和隐私保护不足：手工或分散管理系统中的数据安全和隐私保护机制不完善，容易导致数据泄露和不当使用。

1.2 系统开发的现实意义

为了应对上述问题，开发一个综合性的教育管理系统，旨在实现学生、教师、课程和成绩的统一管理，具有重要的现实意义。具体意义如下：

1. **提升管理效率**：通过信息化手段，将学生、教师和课程管理等日常事务集成到一个系统中，大幅提升管理效率，减少手工操作和人为错误。
2. **保证数据一致性和准确性**：系统内的数据集中管理和统一更新，确保了数据的一致性和准确性，避免了数据冗余和冲突。
3. **实时数据更新和共享**：系统提供实时的数据更新和共享功能，使得各级管理人员能够及时获取最新的信息，支持快速决策。
4. **提高信息安全和隐私保护**：系统设计中引入了严格的数据安全和隐私保护机制，有效防止数据泄露和不当使用，保障用户信息的安全。
5. **支持教学管理和决策分析**：系统不仅能够实现日常管理，还能够提供教学质量评估和决策支持功能，帮助管理人员进行科学的决策和教学优化。

1.3 目标用户和使用场景

本系统的主要目标用户包括高校的教务管理人员、教师和学生。具体使用场景如下：

- **教务管理人员**：通过系统实现学生注册、课程安排、成绩管理和教师评估等功能，提高工作效率和管理水平。

- **教师**：方便教师查看和管理课程安排、录入学生成绩、与学生进行沟通和反馈等。
- **学生**：学生可以通过系统查看个人信息、课程安排、成绩查询等，方便与教师和管理人员进行交流。

综上所述，开发一个综合性的教育管理系统，不仅能够解决现有系统中的诸多问题，还能显著提升高校的管理水平和效率，具有重要的现实意义。

第二章 技术栈及其详细说明

本项目使用了 Spring Boot、MyBatis 和 Lombok 等后端技术，结合 Vue.js、Vue Router 和 Element Plus 等前端技术，以及 MySQL 数据库。Spring Boot 简化微服务开发，MyBatis 提供灵活的 SQL 操作，Lombok 减少样板代码。前端使用 Vue.js 的组件化开发和双向数据绑定，Vue Router 实现页面导航，Element Plus 构建现代化 UI。MySQL 作为关系型数据库，提供高效的数据管理和检索。通过这些技术，我们构建了一个高效、可维护的应用程序。

2.1 后端技术

Spring Boot 简化微服务开发，MyBatis 提供灵活的 SQL 操作，Lombok 通过注解减少样板代码，提升开发效率和代码可读性。

2.1.1 Spring Boot

Spring Boot 是一个基于 Spring 框架的微服务开发框架，旨在简化 Spring 应用的开发与部署。它通过约定优于配置的理念，提供了一套开箱即用的默认配置，从而减少了大量的样板代码和配置。

表 1 Spring Boot 和其它框架比较

架构名称	自动配置	内嵌服务器	CLI 支持	生态系统	学习曲线	配置文件	微服务架构支持
Spring Boot	是	是	是	丰富	低	少量	优秀
Spring Framework	否	否	否	丰富	中	大量	一般
Java EE	否	否	否	较少	高	大量	一般
Play Framework	是	是	否	较少	低	少量	一般

Spring Boot 的主要目标是减少开发人员在应用程序开发和部署过程中的工作量和复杂性。通过自动配置和嵌入式服务器，开发人员可以快速启动和运行应用程序，而无需进行繁琐的设置。Spring Boot 的 CLI 工具使得快速构建原型应用变得更加容易。此外，Spring Boot 拥有一个庞大且成熟的生态系统，提供了丰富的第三方集成和扩展选项，进一步提升了开发效率和应用的可维护性。

2.1.2 MyBatis

MyBatis 是一个优秀的持久层框架，它支持自定义 SQL、存储过程以及高级映射。与 Hibernate 等 ORM 框架不同，MyBatis 直接使用 SQL 语句操作数据库，并通过 XML 或注解将 SQL 语句与 Java 方法关联。

表 2 MyBatis 框架特征

特性	详细描述
SQL 直接编写	开发人员可以直接编写 SQL 语句，更加灵活和高效。
动态 SQL	支持动态 SQL，通过条件语句灵活生成 SQL 语句。
缓存机制	内置一级缓存和二级缓存机制，提升查询性能。
易于扩展	通过插件机制可以方便地扩展 MyBatis 功能。

MyBatis 的设计思想是让开发者能够充分利用 SQL 的强大功能和灵活性，同时提供简洁的 API，使得数据库操作更加直接和高效。开发人员可以通过 XML 文件或注解将 SQL 语句与 Java 方法进行映射，并可以动态生成 SQL，从而满足各种复杂的业务需求。MyBatis 的缓存机制有效地提高了数据库查询的性能，而其插件机制则提供了强大的扩展能力，使得开发者可以根据需要自定义 MyBatis 的行为。

2.1.3 Lombok

Lombok 是一个 Java 库，它通过注解的方式简化了 Java 类的开发，减少了样板代码的编写。Lombok 在编译时生成常见的方法，如 getter、setter、构造函数、equals、hashCode、toString 等，从而使代码更加简洁和易读。

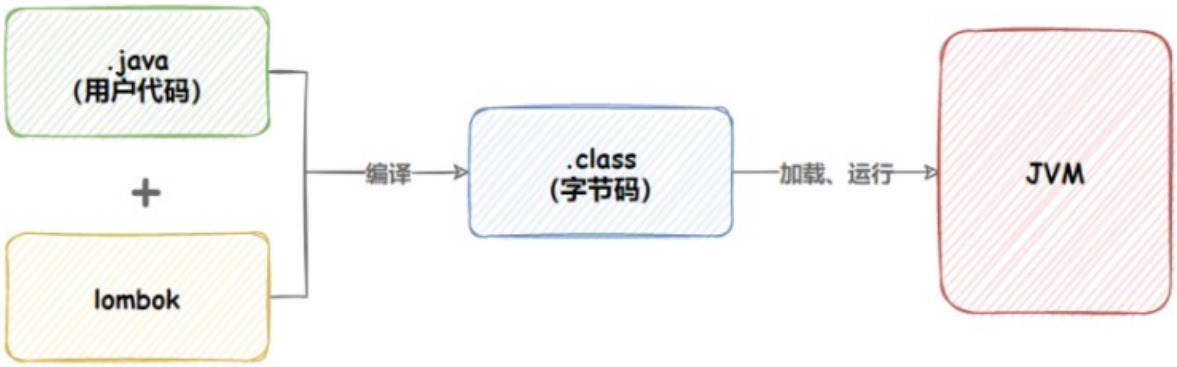

图 2 Lombok 的原理

Lombok 的使用极大地提升了开发效率，通过减少样板代码使得类定义更加简洁和清晰。开发人员只需在类或字段上添加相应的注解，Lombok 就会在编译期间生成所需的代码。这不仅减少了代码量，还降低了手动编写代码引入错误的风险。此外，Lombok 提供了多种注解，满足不同场景的需求，如自动生成无参和全参构造函数、覆盖 toString 方法、实现 equals 和 hashCode 方法等。通过这些注解，开发人员可以专注于业务逻辑的实现，提高代码的可维护性。

2.2 前端技术

在本项目中，Vue.js 通过组件化开发和双向数据绑定极大地提高了开发效率和代码可维护性。利用 Vue Router 实现单页应用的导航，并通过 Element Plus 构建现代化的用户界面。Vue 的虚拟 DOM 技术确保了页面的高效更新，其渐进式框架特性使项目可以灵活扩展和维护。综合对比其他前端框架，Vue.js 在性能、生态系统和学习曲线等方面具有明显优势，适合作为本项目的前端框架。

2.2.1 Vue.js

在本项目中，Vue.js 的组件化开发方式和双向数据绑定特性极大地提高了开发效率和代码可维护性。使用 Vue Router 实现了页面的导航，利用 Element Plus 组件库构建了现代化的用户界面。同时，虚拟 DOM 确保了页面的高效更新，渐进式框架特性使得项目可以灵活地扩展和维护。双向数据绑定：通过 v-model 实现双向数据绑定，简化表单和用户输入的处理。

表 3 Vue 和其他前端框架比较

特点	Vue.js	React	Angular	Svelte
双向数据绑定	是，通过 v-model 实现	否，只支持单向数据流	是，通过 ngModel 实现	是，通过绑定指令（如 bind:this）实现
组件化	是，基于单文件组件（.vue 文件）	是，基于 JSX 和函数式组件	是，基于类和装饰器的组件	是，基于 Svelte 文件和声明式组件
虚拟 DOM	是，采用虚拟 DOM 实现高效更新	是，采用虚拟 DOM 实现高效更新	否，使用增量 DOM 更新	否，编译时生成优化的原生 DOM 操作
渐进式框架	是，可以逐步引入特性	否，需要结合其他库（如 React Router）	否，是一个完整的框架	否，单一用途的框架
学习曲线	平缓，易于上手	中等，需要学习 JSX 和单向数据流	陡峭，需要学习 TypeScript 和完整框架	平缓，语法简单，编译器友好
性能	高，虚拟 DOM 高效更新	高，虚拟 DOM 高效更新	高，优化的增量 DOM 更新	非常高，编译时生成高效的原生代码
生态系统	丰富，拥有 Vue Router、Vuex 等生态	丰富，拥有 React Router、Redux 等生态	完整，内置路由、表单、HTTP 等全套功能	新兴，相对较小，但快速发展
社区支持	活跃，良好的社区支持	活跃，庞大的社区和企业支持	活跃，广泛的社区和企业支持	新兴，社区较小，但增长迅速

文件大小	小，核心库较小	中等，核心库较小，但需配合其他库使用	大，完整框架，包含大量功能	非常小，编译输出的代码非常紧凑

通过上述比较表格，可以看出 Vue.js 在本项目中的优势和适用性。选择 Vue.js 作为本项目的前端框架，可以充分利用其双向数据绑定、组件化开发、虚拟 DOM 和渐进式框架的特点，构建高效、易维护的用户界面。

2.2.2 Vue Router

Vue Router 是 Vue.js 的官方路由管理器，用于构建单页应用（SPA）。它与 Vue.js 核心深度集成，让构建单页应用变得更加容易。

特点

- **声明式路由**：通过配置对象声明路由规则，清晰直观。
- **嵌套路由**：支持嵌套路由，方便构建复杂的视图结构。
- **路由守卫**：提供导航守卫机制，控制路由访问权限和跳转逻辑。
- **动态路由**：支持动态路由匹配，灵活处理路由参数。

简短示例代码

以下是一个简单的 Vue Router 示例，展示了如何配置路由以及使用导航守卫。

```
// 导入必要的模块
import { createRouter, createWebHistory } from 'vue-router';
import HomeView from '../views/Home.vue';
import Login from '../views/Login.vue';
import Profile from '../views/Profile.vue';

// 定义路由配置
const routes = [
    { path: '/', name: 'Home', component: HomeView },
    { path: '/login', name: 'Login', component: Login },
    { path: '/profile', name: 'Profile', component: Profile, meta:
{ requiresAuth: true } }
];

// 创建路由实例
const router = createRouter({
    history: createWebHistory(),
```

```
    routes
});

// 导航守卫
router.beforeEach((to, from, next) => {
    const isAuthenticated = !!localStorage.getItem('auth-token');
    if (to.meta.requiresAuth && !isAuthenticated) {
        next({ name: 'Login' });
    } else {
        next();
    }
});

// 导出路由实例
export default router;
```

2.2.3 Element Plus

Element Plus 是一个基于 Vue 3 的组件库，提供了一整套丰富的 UI 组件，旨在帮助开发者快速构建现代化的 Web 应用。在本项目中，Element Plus 被广泛应用于构建各种表单、对话框、按钮、表格等界面组件。

在 main.js 中引入 Element Plus：

```
import ElementPlus from 'element-plus';
import 'element-plus/dist/index.css';
import { createApp } from 'vue';
import App from './App.vue';
import router from './router';

const app = createApp(App);

app.use(router);
app.use(ElementPlus);

app.mount('#app');
```

此配置确保了 Element Plus 在整个应用中可用，并且应用了其默认样式。组件库中的组件如 <el-button>, <el-table> 等在各个视图组件中被使用，提升了开发效率和界面一致性。

2.3 数据库技术

MySQL 是一个由 Oracle 公司维护和开发的关系型数据库管理系统（RDBMS），使用 SQL 进行数据管理。作为全球最流行的开源数据库之一，MySQL 以其可靠性、高性能和易用性，广泛应用于 Web 开发和中小型应用。本项目包括学院表、教务表、专业表等多个表格及其关系，确保数据高效管理和检索。结合这些技术，我们构建了一个功能强大、性能优越且易于维护的应用程序。

2.3.1 MySQL

MySQL 是一个关系型数据库管理系统（RDBMS），使用结构化查询语言（SQL）进行数据库的访问和管理。它是世界上最流行的开源数据库之一，由 Oracle 公司维护和开发。MySQL 是许多中小型应用程序的首选数据库，尤其是在 Web 开发中，因为它的可靠性、高性能和易用性。

表 4 MySQL 和其它数据库比较

特性	MySQL	PostgreSQL	Oracle	Microsoft SQL Server	MongoDB (NoSQL)
开源/商业	开源	开源	商业	商业	开源
性能	高效，适合高并发	高效，支持复杂查询和事务	高效，适合大规模企业应用	高效，适合大规模企业应用	高效，适合大数据和快速查询
平台支持	跨平台	跨平台	跨平台	Windows 主要	跨平台
安全性	强大的安全性功能	强大的安全性和扩展性	企业级安全性和管理功能	企业级安全性和管理功能	安全性较高，基于角色的访问控制
扩展性	高扩展性，支持分区、复制	高扩展性，支持并行查询	极高扩展性，支持大规模部署	高扩展性，支持大规模部署	高扩展性，适合分布式数据库
事务支持	支持 ACID 事务	完全支持 ACID 事务	完全支持 ACID 事务	完全支持 ACID 事务	不支持（默认），可选多文档事务
社区支持	活跃的社区和丰富的文档	活跃的社区和丰富的文档	官方支持和社区支持	官方支持和社区支持	活跃的社区和丰富的文档
适用场景	Web 开发、中小型应用	复杂查询、数据分析、GIS 应用	大规模企业应用、金融系统	大规模企业应用、BI 系统	大数据应用、实时分析、物联网

2.3.2 数据库设计

数据库设计是通过分析和设计，将数据结构化存储在数据库中，以便高效地管理和检索数据。在本项目中，数据库设计包括多个表格及其关系，如学院表（college）、教务表（dean）、专业表（speciality）等。

主要特性：

规范化设计：通过规范化设计，减少数据冗余，提高数据一致性。

外键约束：使用外键约束维护表与表之间的关系，确保数据完整性。

索引优化：通过合理设计索引，提高查询性能和数据检索速度。

以上是本项目所使用的主要技术栈及其详细说明。通过这些技术的结合，我们可以构建一个功能强大、性能优越且易于维护的应用程序。

第三章 系统设计

本项目的数据库设计涵盖学院、教务、专业、班级、学生、教师、学期、课程、考试及成绩的表结构和外键约束。DAO/Mapper 设计提供了插入、删除、更新和查询等数据库操作，Controller 设计实现了对 Mapper 的调用，支持 RESTful 接口。前端通过 Vue Router 配置路由，每个路径对应一个组件，如 Home.vue、Login.vue 等。应用初始化包括创建 Vue 实例、使用路由和 Element Plus 组件库，并将应用挂载到 DOM 元素上，确保了应用的可维护性和扩展性。

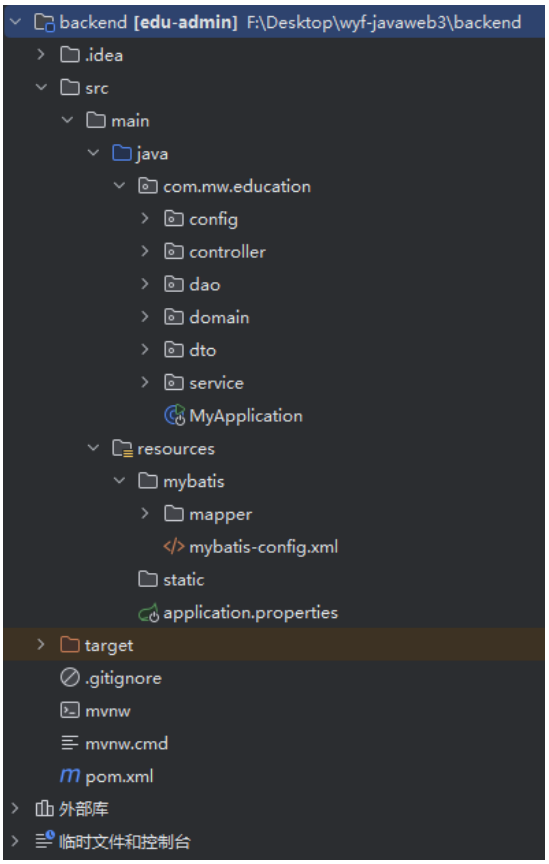

图 3 后端目录结构

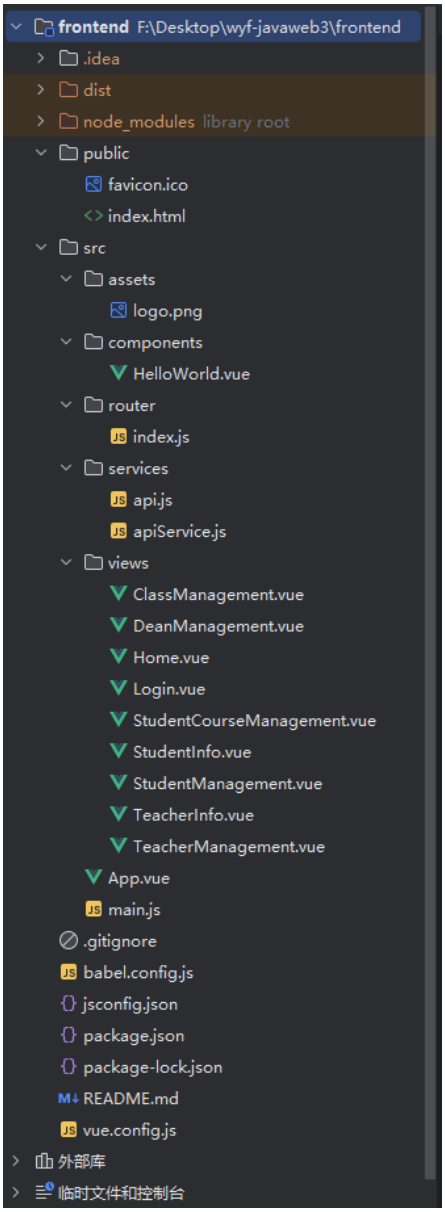

图 4 前端目录结构

3.1 系统功能设计

3.1.1 登录功能

用户可以以三种身份（Dean、Teacher、Student）登录系统，每种身份具有不同的功能权限。

3.1.2 Dean 身份功能

当以 Dean 身份登录时，可以使用以下功能：

表 5 Dean 身份可以使用功能

功能	描述
教师管理	展示学院中所有教师信息，具有基础 CRUD 操作，以及为老师分配课程
学生管理	展示学生信息，具有基础 CRUD 操作
教务管理	展示教务人员信息，具有基础 CRUD 操作
班级管理	展示班级信息，具有基础 CRUD 操作，以及分配课程

教师管理功能

表 6 管理员身份的教师管理

功能	描述
展示教师信息	以表格形式展示所有教师的姓名、工号和学院信息
创建	添加新教师，输入教师姓名、工号和学院信息
读取	查看教师详细信息
更新	修改教师信息
删除	删除教师信息
查看负责课程	点击某个教师可以查看其负责的课程信息，包括课程名称、学期和课程备注
添加课程	为教师分配新的课程
删除课程	从教师负责的课程列表中移除课程

学生管理功能

表 7 管理员身份下的学生管理

功能	描述
展示学生信息	以表格形式展示所有学生的姓名、学号、班级、专业和学院信息
创建	添加新学生，输入学生姓名、学号、班级、专业和学院信息
读取	查看学生详细信息
更新	修改学生信息
删除	删除学生信息

教务管理功能

表 8 管理员身份下的教务人员管理

功能	描述
展示教务人员信息	以表格形式展示所有教务人员的姓名、工号和备注
创建	添加新教务人员，输入教务人员姓名、工号和备注信息
读取	查看教务人员详细信息
更新	修改教务人员信息
删除	删除教务人员信息

班级管理功能

表 9 管理员身份下的班级管理

功能	描述
展示班级信息	以表格形式展示所有班级的名称、代码、专业名称、专业年级和学院名称

创建	添加新班级，输入班级名称、代码、专业名称、专业年级和学院信息
读取	查看班级详细信息
更新	修改班级信息
删除	删除班级信息
查看班级课程	点击某个班级可以查看其课程信息，包括课程名称、学期和课程备注
添加课程	为班级分配新的课程
删除课程	从班级课程列表中移除课程

3.1.3 Teacher 身份功能

当以 Teacher 身份登录时，可以使用以下功能：

表 10 教师身份登录功能

功能	描述
查看基本信息	查看登录教师基本信息
查看教授课程信息	查看教师教授的课程信息

3.1.4 Student 身份功能

当以 Student 身份登录时，可以使用以下功能：

表 11 学生身份登录功能

功能	描述
查看基本信息	查看登录学生基本信息
查看课程信息	查看学生的课程信息
查看成绩	查看学生的成绩

3.2 数据库设计

本项目的数据库设计包括多张表结构，涵盖学院、教务、专业、班级、学生、教师、学期、课程、考试及其关联和成绩。每张表都有详细的字段定义和外键约束。

3.2.1 表结构设计

表 12 college 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		学院 ID，主键，自增
name	varchar(32)	否		名称

字段名	数据类型	是否为空	默认值	备注
remark	varchar(256)	是	NULL	辅助描述

表 13 dean 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		教务 ID，主键，自增
name	varchar(16)	否		名字
password	varchar(32)	否	'123456789'	密码
code	varchar(16)	否		工号（登录）
remark	varchar(256)	是	NULL	辅助描述

表 14 speciality 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		专业 ID，主键，自增
name	varchar(32)	否		名称
grade	year	否		年级
code	varchar(16)	否		学科代码
college_id	int	是	-1	学院 ID，外键参考 college(id)
remark	varchar(256)	是	NULL	辅助描述

表 15 class 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		班级 ID，主键，自增
name	varchar(32)	否		名称
code	varchar(16)	否		班级编号
speciality_id	int	是	-1	专业 ID，外键参考 speciality(id)
remark	varchar(256)	是	NULL	辅助描述

表 16 student 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		学生 ID，主键，自增
name	varchar(32)	否		姓名
password	varchar(64)	否	'123456789'	密码

字段名	数据类型	是否为空	默认值	备注
code	varchar(32)	否		学号（登录）
class_id	int	是	-1	班级 ID，外键参考 class(id)
remark	varchar(256)	是	NULL	辅助描述

表 17 teacher 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		教师 ID，主键，自增
name	varchar(32)	否		姓名
password	varchar(64)	否	'123456789'	密码
code	varchar(32)	否		工号（登录）
college_id	int	是	-1	学院 ID，外键参考 college(id)
remark	varchar(256)	是	NULL	辅助描述

表 18 term 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		学期 ID，主键，自增
name	varchar(100)	否		学期名称
start_date	date	否		开始日期
end_date	date	否		结束日期
remark	varchar(255)	是	NULL	备注

表 19 course 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		课程 ID，主键，自增
name	varchar(100)	否		课程名称
code	varchar(50)	否		课程编号
term_id	int	否		学期 ID，外键参考 term(id)
remark	varchar(255)	是	NULL	备注

表 20 class_course 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		关联 ID，主键，自增
class_id	int	否		班级 ID，外键参考 class(id)

字段名	数据类型	是否为空	默认值	备注
course_id	int	否		课程 ID，外键参考 course(id)

表 21 exam 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		考试 ID，主键，自增
course_id	int	否		课程 ID，外键参考 course(id)
date	date	否		考试日期
location	varchar(100)	否		考试地点
remark	varchar(255)	是	NULL	备注

表 22 student_course_score 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		成绩 ID，主键，自增
student_id	int	否		学生 ID，外键参考 student(id)
course_id	int	否		课程 ID，外键参考 course(id)
score	decimal(5, 2)	否		成绩
remark	varchar(255)	是	NULL	备注

表 23 teacher_course 表

字段名	数据类型	是否为空	默认值	备注
id	int	否		关联 ID，主键，自增
teacher_id	int	否		教师 ID，外键参考 teacher(id)
course_id	int	否		课程 ID，外键参考 course(id)

3.2.2 表之间的关系

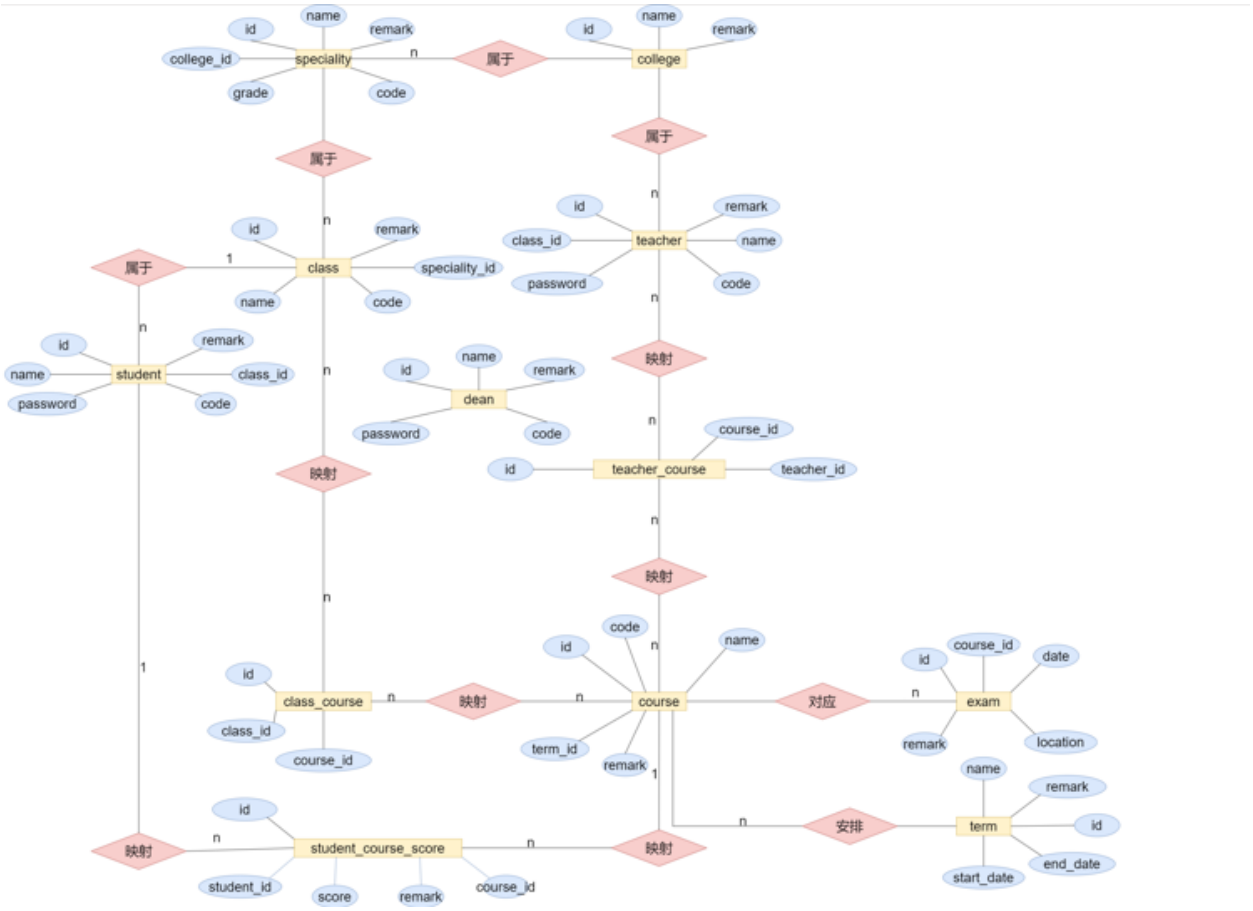

图 5 数据库表 ER 图

3.3 后端设计

本项目的 DAO/Mapper 设计包括各类 Mapper 接口用于数据库操作，涵盖插入、删除、更新和查询等功能。每个 Mapper 对应一个数据库表，并提供 CRUD 方法及特定查询功能，如 ClassCourseMapper 的 selectByClassId，StudentMapper 的 selectAllWithDetails 等。Controller 设计对应 HTTP 请求方法，实现对 Mapper 的调用，支持 RESTful 接口，如 ClassCourseController 提供插入、删除、更新和查询 ClassCourse 记录的 API。这种设计确保了数据操作的灵活性和接口调用的规范性。

3.3.1 DAO/Mapper 设计

表 24 ClassCourseMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	ClassCourse row	受影响的行数

方法名	描述	参数	返回值
selectByPrimaryKey	根据主键查询记录	Integer id	ClassCourse 对象
selectAll	查询所有记录	无	List<ClassCourse>
updateByPrimaryKey	根据主键更新记录	ClassCourse row	受影响的行数
selectByClassId	根据班级 ID 查询记录	Integer classId	List<ClassCourse>
deleteByClassId	根据班级 ID 删除记录	Integer classId	受影响的行数

表 25 ClassGroupMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	ClassGroup row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	ClassGroup 对象
selectAll	查询所有记录	无	List<ClassGroup>
updateByPrimaryKey	根据主键更新记录	ClassGroup row	受影响的行数
selectByPrimaryKeyJoinedSpecialityAndCollege	根据主键查询记录及其关联的 Speciality 和 College	Integer id	ClassGroupAndSpecialityAndCollege 对象
selectAllJoinedSpecialityAndCollege	查询所有记录及其关联的 Speciality 和 College	无	List<ClassGroupAndSpecialityAndCollege>

表 26 CollegeMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	College row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	College 对象
selectAll	查询所有记录	无	List<College>
updateByPrimaryKey	根据主键更新记录	College row	受影响的行数

表 27 CourseMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	Course row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	Course 对象

方法名	描述	参数	返回值
selectAll	查询所有记录	无	List<Course>
updateByPrimaryKey	根据主键更新记录	Course row	受影响的行数

表 28 DeanMapper

方法名	描述	参数	返回值
findAll	查询所有记录	无	List<Dean>
findById	根据主键查询记录	int id	Dean 对象
insert	插入新记录	Dean dean	无
update	更新记录	Dean dean	无
delete	根据主键删除记录	int id	无
countByCodeAndPassword	根据账号和密码统计记录数量	@Param("code") String code, @Param("password") String password	记录数量

表 29 ExamMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	Exam row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	Exam 对象
selectAll	查询所有记录	无	List<Exam>
updateByPrimaryKey	根据主键更新记录	Exam row	受影响的行数
getExamsByTermAndSpeciality	根据学期和专业 ID 查询考试记录	@Param("termId") int termId, @Param("specialityId") int specialityId	List<Exam>

表 30 SpecialityMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	Speciality row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	Speciality 对象
selectAll	查询所有记录	无	List<Speciality>
updateByPrimaryKey	根据主键更新记录	Speciality row	受影响的行数

方法名	描述	参数	返回值
selectByPrimaryKeyJoinedCollege	根据主键查询记录及其关联的 College	@Param("id") Integer id	SpecialityAndCollege 对象
selectAllJoinedCollege	查询所有记录及其关联的 College	无	List<SpecialityAndCollege>

表 31 StudentCourseScoreMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	StudentCourseScore row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	StudentCourseScore 对象
selectAll	查询所有记录	无	List<StudentCourseScore>
updateByPrimaryKey	根据主键更新记录	StudentCourseScore row	受影响的行数
getStudentsAndScoresByCourseId	根据课程 ID 查询学生及成绩	@Param("courseId") int courseId	List<StudentScoreDTO>
deleteByStudentId	根据学生 ID 删除成绩记录	@Param("studentId") Integer studentId	无
selectScoresByStudentCode	根据学生代码查询成绩	@Param("studentCode") String studentCode	List<StudentCourseScore>

表 32 StudentMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	Student row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	Student 对象
selectAll	查询所有记录	无	List<Student>
updateByPrimaryKey	根据主键更新记录	Student row	受影响的行数
countByCodeAndPassword	根据账号和密码统计记录数量	@Param("code") String code, @Param("password") String password	记录数量
selectAllWithDetails	查询所有学生及其详细信息	无	List<StudentAndClassAndSpecialityAndCollege>
selectByIdWithDetails	根据 ID 查询学生	@Param("id") Integer	StudentAndClassAndSpecial

方法名	描述	参数	返回值
	及其详细信息	id	ityAndCollege 对象
selectByClassId	根据班级 ID 查询学生	int classId	List<Student>
selectByCodeWithDetails	根据学号查询学生及其详细信息	@Param("code") String code	StudentAndClassAndSpecialityAndCollege 对象
selectByCode	根据学号查询学生	@Param("code") String code	Student 对象

表 33 TeacherCourseMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	TeacherCourse row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	TeacherCourse 对象
selectAll	查询所有记录	无	List<TeacherCourse>
updateByPrimaryKey	根据主键更新记录	TeacherCourse row	受影响的行数
selectByTeacherId	根据教师 ID 查询课程	@Param("teacherId") int teacherId	List<Course>
selectTeachersByCourseId	根据课程 ID 查询教师	@Param("courseId") Integer courseId	List<Teacher>
countByTeacherId	根据教师 ID 统计记录数量	@Param("teacherId") int teacherId	记录数量
deleteByTeacherId	根据教师 ID 删除记录	@Param("teacherId") int teacherId	无
selectTeacherCoursesByTeacherId	根据教师 ID 查询教师课程记录	@Param("teacherId") int teacherId	List<TeacherCourse>

表 34 TeacherMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	Teacher row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	Teacher 对象
selectAll	查询所有记录	无	List<Teacher>
updateByPrimaryKey	根据主键更新记录	Teacher row	受影响的行数
selectByPrimaryKeyJoinedCollege	根据主键查询记录及其关联的 College	Integer id	TeacherAndCollege 对象
selectAllJoinedCollege	查询所有记录及其	无	List<TeacherAndCollege>

方法名	描述	参数	返回值
	关联的 College		
countByCodeAndPassword	根据账号和密码统计记录数量	@Param("code") String code, @Param("password") String password	记录数量
selectByCode	根据工号查询教师	@Param("code") String code	Teacher 对象

表 35 TermMapper

方法名	描述	参数	返回值
deleteByPrimaryKey	根据主键删除记录	Integer id	受影响的行数
insert	插入新记录	Term row	受影响的行数
selectByPrimaryKey	根据主键查询记录	Integer id	Term 对象
selectAll	查询所有记录	无	List<Term>
updateByPrimaryKey	根据主键更新记录	Term row	受影响的行数

3.3.2 Controller 设计

在本节中，我们详细介绍了各个 Mapper 类对应的 Controller 设计，这些 Controller 负责处理前端请求并调用相应的服务层方法来操作数据库。对于大部分只包含基础 CRUD 操作的 Controller，这里不再详述，仅展示具有复杂查询功能的 Controller 设计。

1. ClassGroupController

   - selectByPrimaryKeyJoinedSpecialityAndCollege: 根据主键查询 ClassGroup 记录，同时获取关联的 Speciality 和 College 信息。
   - selectAllJoinedSpecialityAndCollege: 查询所有 ClassGroup 记录，同时获取每条记录关联的 Speciality 和 College 信息。

2. ExamController

   - getExamsByTermAndSpeciality: 根据指定的学期和专业 ID 查询对应的考试记录。

3. SpecialityController

   - selectByPrimaryKeyJoinedCollege: 根据主键查询 Speciality 记录，同时获取关联的 College 信息。

- selectAllJoinedCollege：查询所有 Speciality 记录，同时获取每条记录关联的 College 信息。

4. StudentCourseScoreController

- getStudentsAndScoresByCourseId：根据课程 ID 查询所有学生及其对应的成绩。
- selectScoresByStudentCode：根据学生代码查询该学生的所有成绩记录。

5. StudentController

- selectAllWithDetails：查询所有学生记录及其详细信息，包括关联的 Class、Speciality 和 College 信息。
- selectByIdWithDetails：根据学生 ID 查询其详细信息，包括关联的 Class、Speciality 和 College 信息。

6. TeacherCourseController

- selectByTeacherId：根据教师 ID 查询该教师所教授的课程。
- selectTeachersByCourseId：根据课程 ID 查询所有教授该课程的教师。
- selectTeacherCoursesByTeacherId：根据教师 ID 查询该教师的所有课程记录。

7. TeacherController

- selectByPrimaryKeyJoinedCollege：根据主键查询 Teacher 记录，同时获取关联的 College 信息。
- selectAllJoinedCollege：查询所有 Teacher 记录，同时获取每条记录关联的 College 信息。

这些复杂查询功能的 Controller 方法扩展了基础 CRUD 操作，提供了更为丰富的数据查询功能，以满足前端的多样化需求。这些方法通过精确的参数查询，能够高效地获取相关联的信息，为前端应用提供了极大的便利。

3.4 前端设计

在前端设计中，我们通过 Vue Router 配置了项目中的各种路由。每个路由路径对应一个组件，并通过路径名称来访问这些组件。下面是项目中的路由配置详情。

3.4.1 路由配置

如表 37 所示，以下是项目中的路由配置：

表 36 项目路由配置

路径	名称	组件
/login	Login	Login.vue
/	Home	HomeView.vue
/teacherManagement	TeacherManagement	TeacherManagement.vue
/studentManagement	StudentManagement	StudentManagement.vue
/deanManagement	DeanManagement	DeanManagement.vue
/studentInfo	StudentInfo	StudentInfo.vue
/studentCourseManagement	StudentCourseManagement	StudentCourseManagement.vue
/teacherInfo	TeacherInfo	TeacherInfo.vue
/classManagement	ClassManagement	ClassManagement.vue

每个路径都映射到一个特定的组件。例如，当用户访问 /login 路径时，会加载 Login.vue 组件；访问 /teacherManagement 路径时，会加载 TeacherManagement.vue 组件。通过这样的配置，我们能够清晰地定义应用程序中的页面结构，并且可以在这些页面之间进行导航。

3.4.2 组件定义

在项目中，每个路由路径都关联了一个 Vue 组件，这些组件负责渲染对应的页面内容和处理相关的业务逻辑。下面列出了各个组件的定义和描述：

表 37 项目前端组件

组件名称	组件文件	描述
HomeView	Home.vue	主页视图，显示欢迎信息和基本统计数据
Login	Login.vue	用户登录页面
TeacherManagement	TeacherManagement.vue	教师管理页面，提供教师的增删改查和课程分配
StudentManagement	StudentManagement.vue	学生管理页面，提供学生的增删改查
DeanManagement	DeanManagement.vue	教务管理页面，提供教务的增删改查
StudentInfo	StudentInfo.vue	学生信息页面，显示学生的详细信息
StudentCourseManagement	StudentCourseManagement.vue	学生课程与成绩管理页面，显示学生课程和成绩

组件名称	组件文件	描述
TeacherInfo	TeacherInfo.vue	教师信息页面，显示教师的详细信息和教授课程
ClassManagement	ClassManagement.vue	班级管理页面，提供班级的增删改查和课程分配

这些组件通过 Vue 组件体系结构，构建了一个结构化且功能丰富的前端应用。通过组件化的设计，使得每个组件可以独立开发、测试和维护，从而提升了应用的可维护性和扩展性。

3.4.2 应用初始化

应用初始化是整个前端项目的入口，通过以下步骤完成应用的创建和挂载：

1. **创建应用实例**：使用 createApp(App) 方法创建 Vue 应用实例。
2. **使用路由**：通过 app.use(router) 使用 Vue Router 进行路由管理。
3. **使用 ElementPlus**：通过 app.use(ElementPlus) 使用 Element Plus 组件库。
4. **挂载应用**：使用 app.mount('#app') 将 Vue 应用挂载到 DOM 元素上。

这四个步骤确保了应用程序正确地初始化并渲染在浏览器中，为用户提供交互体验。

第四章 系统运行效果

图 6 用户登录

图 7 Home 页面

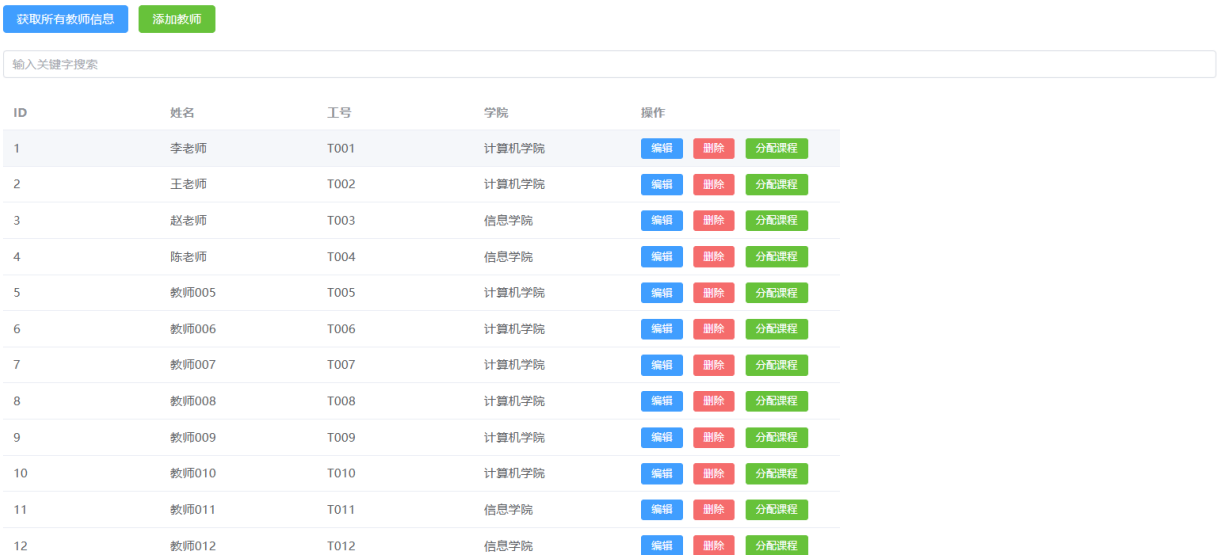

图 8 Dean 的教师管理功能

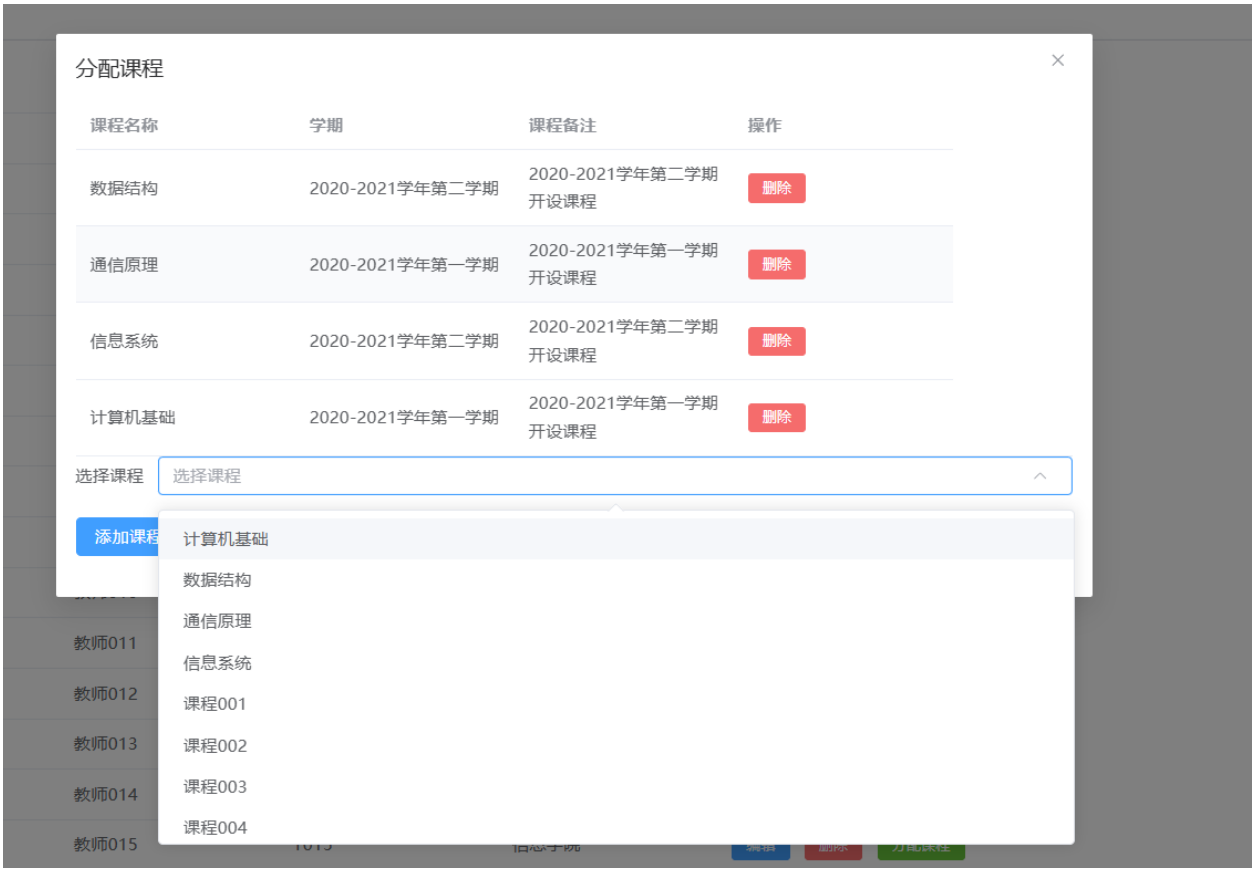

图 9 为教师分配课程

图 10 为班级分配课程

图 11 学生身份登录的学生页面

图 12 学生课程与成绩

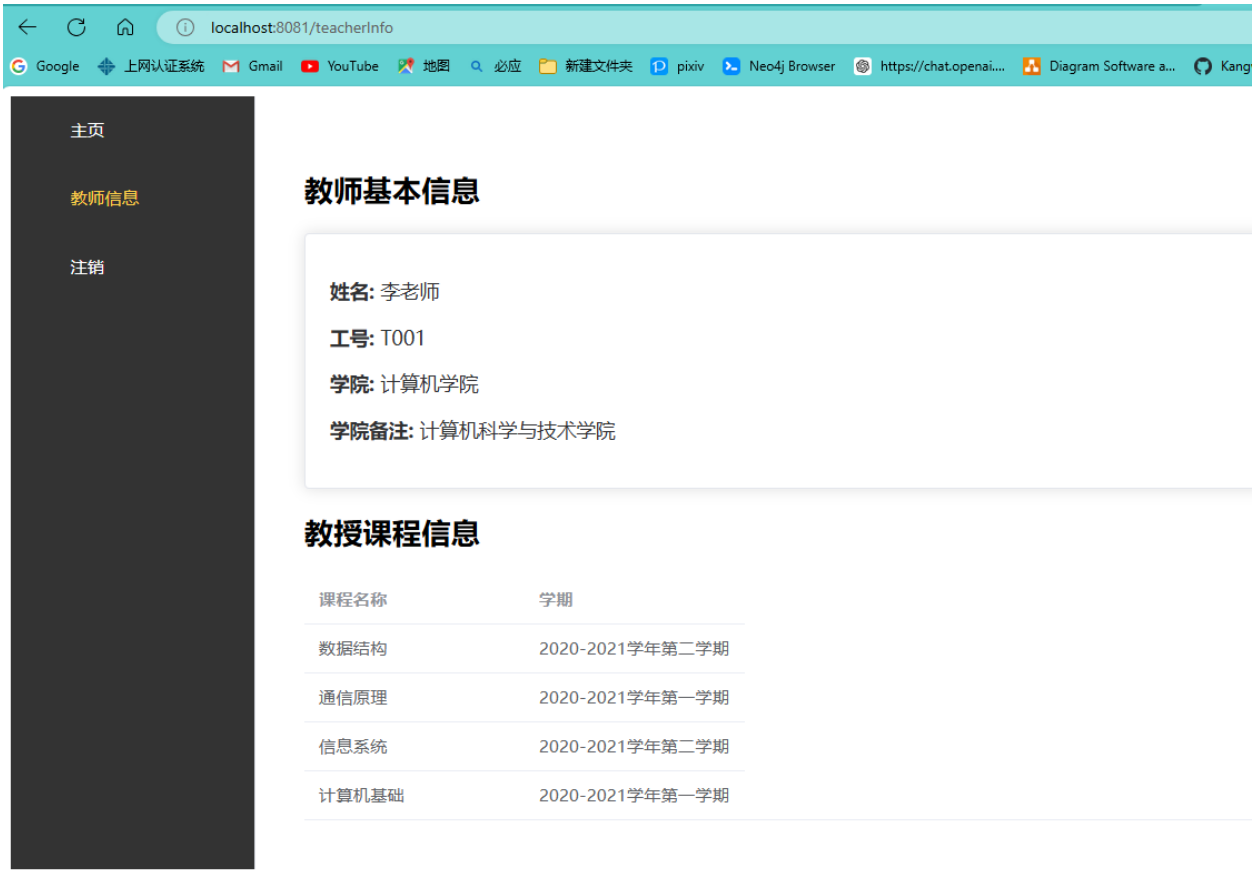

图 13 Teacherinfo

第五章 总结及展望

5.1 项目总结

本项目通过整合前端、后端和数据库的各个部分，成功实现了一个完整的教育管理系统。

后端采用了基于 Java 的 Spring 框架，结合 MyBatis 进行持久层操作，具体实现了定义并实现各类 DAO 接口以进行数据库中各类实体的 CRUD 操作，通过服务层逻辑处理业务操作和数据传输以确保数据的完整性和一致性，并对外提供 RESTful API 服务以支持前端与后端的数据交互，从而确保系统的可扩展性和维护性。

前端采用了 Vue.js 框架，并结合 Element Plus 组件库，提供了用户友好的界面交互，主要实现了通过 Vue Router 进行路由管理以确保用户在不同角色下访问不同的功能模块，采用组件化开发模式以提高代码的可维护性和可复用性，并通过 API 与后端进行数据交互以确保系统的实时性和数据的一致性。

数据库设计遵循了规范化原则，确保了数据的完整性和一致性，具体实现了定义学院、教务、专业、班级、学生、教师、课程和成绩等表及其关联关系，并通过设置外键约束和索引提高了查询效率和数据操作的安全性。

5.2 未来展望

在现有系统的基础上，未来可以从以下几个方面进行扩展和优化：首先，在功能扩展方面，增加对移动端的支持，开发对应的移动端 APP 或响应式 Web 页面，使用户可以随时随地访问系统；引入消息通知功能，及时通知用户重要信息（如课程安排变动、考试安排等），提升系统的实用性；增加数据报表和分析功能，通过图表展示各类统计信息，为决策提供数据支持。

在性能优化方面，引入缓存机制，减少数据库查询次数，提升系统响应速度，可以采用 Redis 等缓存技术；通过分表、分库等手段优化大数据量下的数据库性能，并对查询语句进行优化，确保执行效率。

在安全性提升方面，细化用户权限管理，确保不同角色的用户只能访问和操作与其权限相关的数据，提升系统的安全性；对敏感数据进行加密存储和传输，防止数据泄露和篡改，可以采用 SSL/TLS 等技术对数据传输进行加密。

在用户体验优化方面，聘请专业的 UI/UX 设计师对系统界面进行优化，提升用户体验，确保界面简洁、美观，操作便捷；增加多语言支持，使系统能够服务于更多不同语言的用户，提升系统的国际化水平。通过以上的功能扩展和优化，系统将能够更加高效、便捷地服务于用户，为教育管理工作提供有力支持。