《Java 企业级应用》实验报告

年级、专业、班级	202 级计卓 1 班	姓名	李宽宇	学号	20215279
实验题目	基于 Web 的成绩管理系统				
实验时间	2024.5.11	实验地点	DS3401		
学年学期	2023-2024(2)	实验性质	□验证性 ■设计性 □综合性		

一、实验目的

1. 本次实验的目的是掌握 Java 企业级应用开发工具的使用方法；掌握 Java Web 编程技术；掌握常用 Java 后端开发框架的使用，例如 SpringBoot 等；理解所用框架的工作原理。
2. 设计开发完成一个基于 Web 的成绩管理系统。
3. 抄袭计 0 分。

二、实验项目内容

设计开发完成一个基于 Web 的成绩管理系统。要求如下：

1、后端可以使用 SpringBoot 等框架，**也可以使用其他框架**。要求数据存储在数据库中。自由选择使用什么数据库。

2、前端使用浏览器访问软件系统界面。**自由选择**前端界面技术。自由选择是否前后端分离架构。

3、提供账号注册、登录、注销等功能。教师账号管理学生成绩，学生账号查看学生成绩。教务账号管理班级，课程等。

4、创建类，实现基本对象和他们关系的管理。包括学生、教学班、课程、成绩、教师等。学生至少包含学号、姓名、性别等信息。教学班至少包含教师、课程名字、总人数、教学班号、开课学期等信息。课程至少包含课程编号、课程名字等信息。教师至少包含教师编号、姓名等信息。可以根据自己的分析设计增加其他类。

5、随机生成学生，数量不少于 100。课程数量不少于 3 门。一个教学班有一个教师上一门课程，一个教学班的学生数量不少于 20。教师数量不少于 6 个。一门课至少有两个老师上课。每个学生至少选择 3 门课程。一个学生在一个教学班上一门课，考试后取得一个成绩。一门课的成绩构成有 4 部分构成，包括平时成绩、期中考试、实验成绩和期末考试成绩，然后计算出综合成绩。自定义各项成绩的产生策略，均为整数。

6、提供合适的操作界面完成上述功能。可以录入成绩，可以批量产生成绩等；能够格式规范地显示一个教学班的学生的成绩，可以根据学号排序，可以根据成绩排序。可以统计学生各科、总成绩的分数段分布。可以通过名

字或者学号查询一个学生的所有科目的成绩和总成绩。可以按照学号、各科成绩和总成绩对所有学生进行排名显示。

7、可以实现自己的扩展功能或自定义功能。注意操作使用的方便性，注意类和类之间的关系。充分利用继承，多态等特性，使用上抽象类，接口，泛型，内部类等设计元素，使用好集合类。注意程序的总执行流程和分支执行流程。注意设计思想的表达，注意优化代码结构，优化类的职责分工，注意使用各种框架。代码有注释。

8、在报告中注明自己的创新点、特色等。

9、提交：（1）本实验报告，（2）源代码压缩文件 zip，（3）软件演示的 MP4 视频，视频大小不超过 40M，视频请在**搜狗浏览器**测试能否正常播放。注意源代码加注释。注意文件名称的规范性。文件名：学号姓名 4.docx，学号姓名 4.zip, 学号姓名 4.mp4。**三个文件分别提交**。

三、实验过程或算法（写明创新点或特色、设计思想、前后端各种框架的使用、程序的结构、功能关系图、类的说明和类之间的关系图、程序主要执行流程图，最后是核心源代码，截图等）

1. 创新点或特色

(1)前后端分离：

服务器端可以使用 SpringBoot + Spring Data JPA 框架，使得数据库查询更加简洁，数据存储在本地数据库 mysql 中，提供了 19 个 api 接口供前端使用。

前端界面技术使用 javascript，使用 ajax 等技术向服务器端发送请求。

(2)功能特色：用 excel(.xlsx)文件批量录入、批量导出学生成绩，数据统计绘制饼状图。

(3) 使用 maven 管理文件结构、git 管理项目进度，源代码链接: https://github.com/guoluguodong/scoreSystemOnweb

(4) 使用了桥接模式、装饰器模式、静态工厂、单例模式等设计模式。

(5)数据可视化

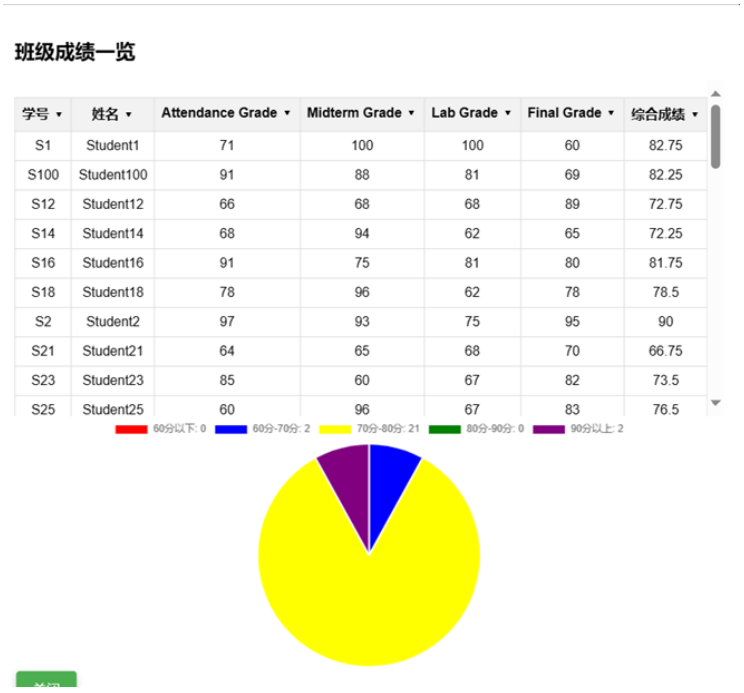


图 1 数据可视化图

2. 设计思想

（1）程序设计重点使用 mysql、DATA JPA、poi-ooxml 等技术，采用了多种设计元素，包括集合框架，格式化器、字节流、Iterable<T>、Optional<T>、@CrossOrigin、@ResponseBody、@RequestParam、@Autowired 等技术充分利用异常处理，使用 maven 管理项目结构，git 管理项目进度。

（2）从代码的规范角度，包名:全小写;类名:首字母大写,每个单词的首字母大写;方法名:小写字母开头,每个单词的首字母大写;变量名: 写字母开头,每个单词的首字母大写;常量名:基本类型的常量名全大写。

（3）前后端分离的设计思路，服务器端可以使用 SpringBoot + Spring

Data JPA 框架，使得数据库查询更加简洁，数据存储在本地数据库 mysql 中，提供了 19 个 api 接口供前端使用。前端界面技术使用 javascript，使用 ajax 等技术向服务器端发送请求。，使得程序呈现出高内聚、低耦合的特点。

(4)设计模式方面，使用了桥接模式、装饰器模式、静态工厂、单例模式、依赖注入等设计模式，所有类满足单一职责原则。

（5）注重程序的实用性与操作使用的方便性。图形界面美观简洁

3.设计原理

Spring Data JPA 是 Spring Data 项目的一部分，它提供了一种简化的数据访问方式，用于与关系型数据库进行交互。它基于 Java Persistence API（JPA） 标准，并提供了一套简洁的 API 和注解，使开发人员能够通过简单的 Java 对象来表示数据库表，并通过自动生成的 SQL 语句执行常见的 CRUD 操作。Spring Data JPA 通过封装 JPA 的复杂性，简化了数据访问层的开发工作，使开发人员能够更专注于业务逻辑的实现。它还提供了丰富的查询方法的定义、分页和排序支持、事务管理等功能，使开发人员能够更方便地进行数据访问和操作。

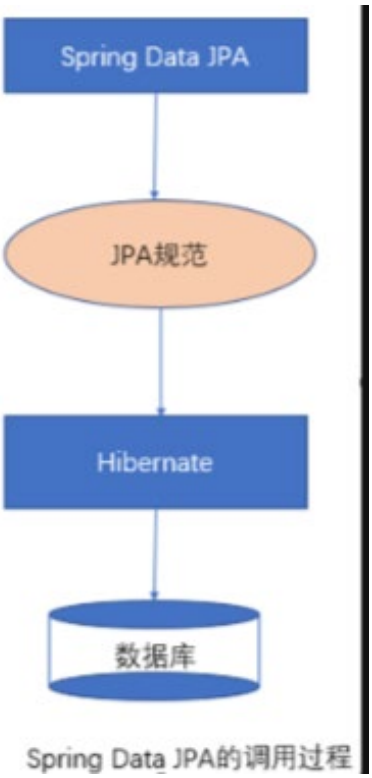


图 2 spring data jpa 的调用过程

4.设计模式的使用

(1)桥接模式，桥接模式（Bridge Pattern）：JDBC 规范中使用了桥接模式。这是一种结构型设计模式，基于类的最小设计原则。通过使用封装、聚合及继承等行为，让不同的类承担不同的职责，DriverManager 是 JDBC 中的一个关键类，它负责加载数据库驱动并管理数据库连接。它充当了桥接模式中的桥接角色，将应用程序与具体的数据库驱动隔离开来。JDBC 使用反射方法加载数据库驱动。每个数据库厂商的驱动类必须实现 java.sql.Driver 接口，并在加载时将自身注册到 DriverManager 中。例如，对于 MySQL，我们可以通过 Class.forName("com.mysql.jdbc.Driver")

来加载驱动。

(2) 工厂模式 (Factory)：Spring 使用工厂模式创建 Bean 实例。Spring IoC 容器就是一个 Bean 工厂，通过配置元数据（XML 文件或注解），Spring 容器可以灵活地创建和管理 Bean。

```
@Configuration
public class AppConfig {
    @Bean
    public MyService myService() {
        return new MyServiceImpl();
    }
}
```

(3)装饰器模式：装饰器模式（Decorator Pattern）允许向一个现有的对象添加新的功能，同时又不改变其结构。这种类型的设计模式属于结构型模式，它是作为现有的类的一个包装。实验中使用了@CrossOrigin、@ResponseBody、@RequestParam、@Autowired、@Controller、@Service

(4) 单例模式:userService 作为 service 的单例。单例模式(Singleton Pattern）这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。这种模式涉及到一个单一的类，该类负责创建自己的对象，同时确保只有单个对象被创建。这个类提供了一种访问其唯一的对象的方式，可以直接访问，不需要实例化该类的对象。

(5) 依赖注入 (Dependency Injection) 这是 Spring 框架的核心模式，通过构造函数注入、setter 注入或字段注入，将依赖对象注入到目标对象中。

```
    @Autowired
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }
}
```

4. 程序的结构

(1)前后端代码分开。浏览器发送 http 请求，服务器接收请求，返回数据、资源或者结果。

(2)对于后端代码，使用 springboot 框架，其遵循遵循试图控制器模式（Model-View-Controler, MVC），Controlller 接收前端请求和并返回结果。

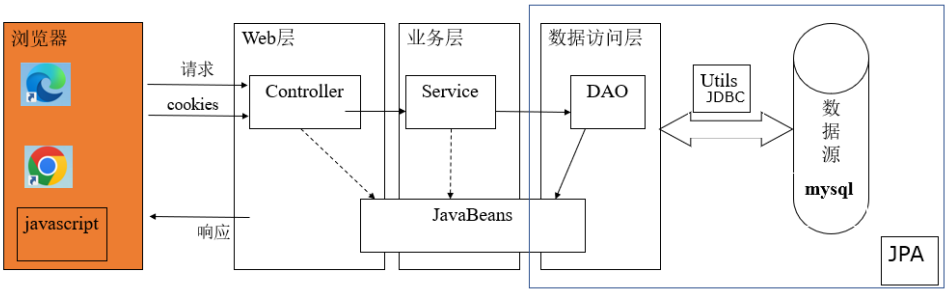


图 3 前后端框架

目录结构如下

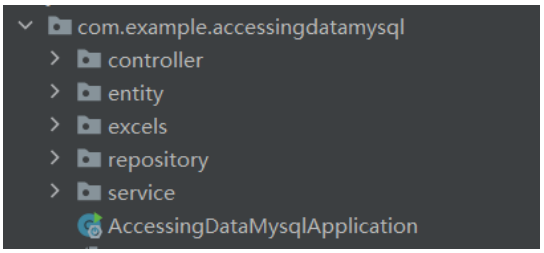


图 4 后端目录结构

controller 目录下使用多个 controller 对外提供 19 个 api 接口

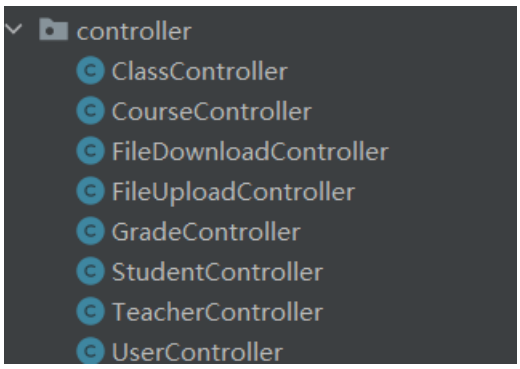


图 5 controller 包下的 api

entity 目录下存储实体类 user、teacher 等

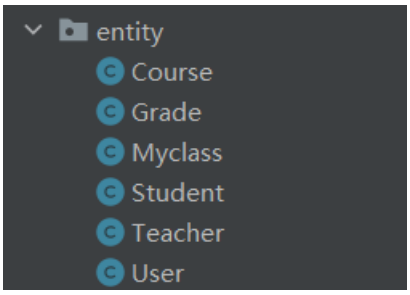


图 6 entity 包下实体类

Excels 存储导入或者导出的学生成绩文件

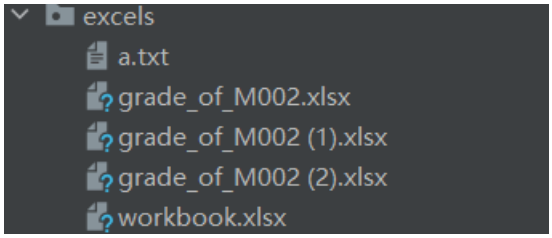


图 7 存储临时文件的目录

Repository 目录下有多个用于访问数据库的接口

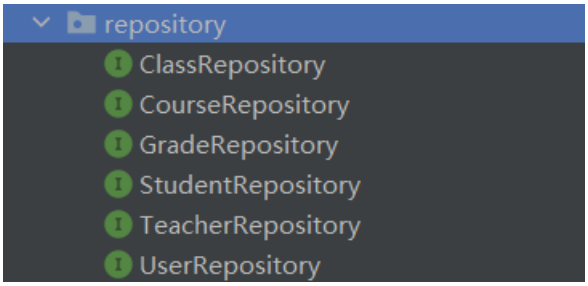


图 8 repository 接口

Service 目录下是业务层，为 controller 提供更深的服务

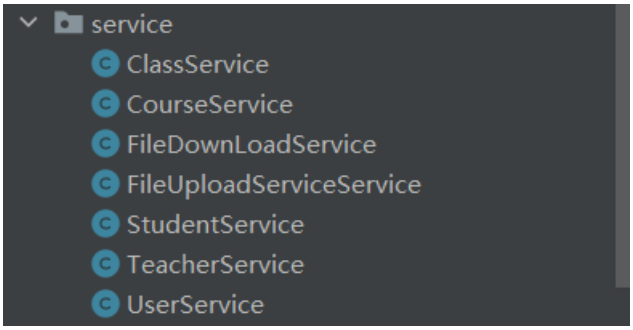


图 9 service

Rescouces 目录下是 appliction.poperties

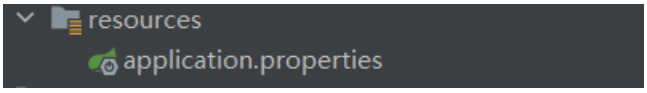


图 10 Rescouces 目录

配置如下，连接数据库

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/lab4
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.jpa.show-sql: true
```

图 11 appliction.poperties 的属性

(3)前端代码结构，Webapp 存放 jsp 文件和资源文件。index.jsp 是登录页面。

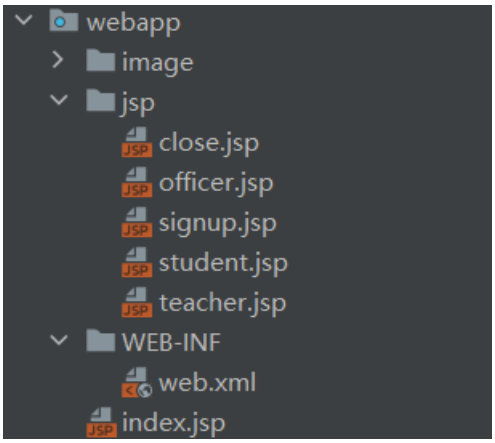


图 12 webapp 目录

5. 功能关系图

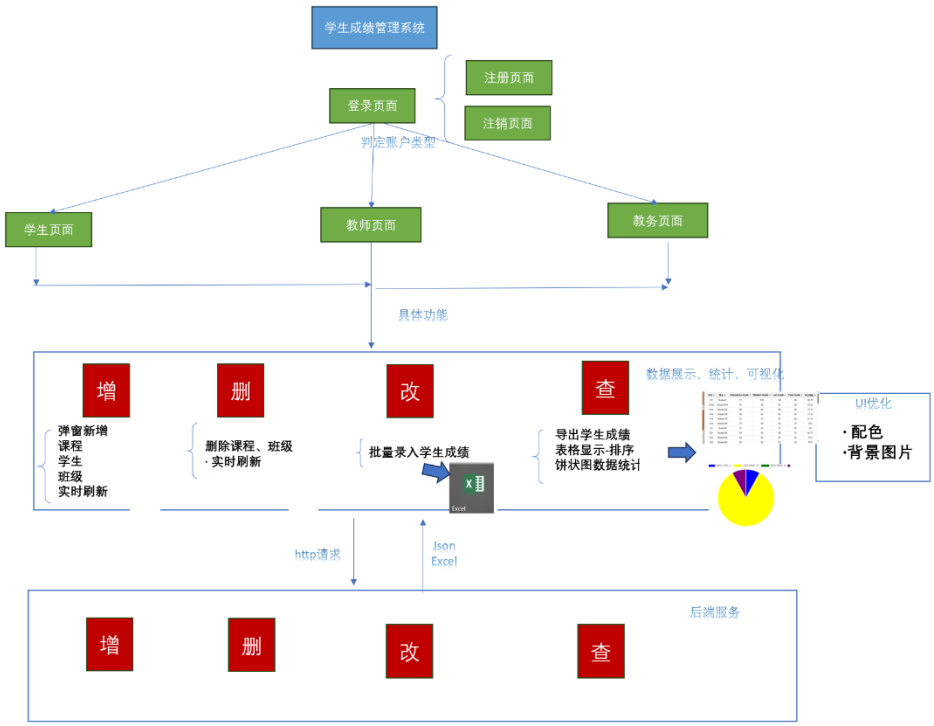


图 13 功能关系图

6. 类的说明和类之间的关系图

（一）类的说明

（1）实体类

User 类，包括用户名、密码、账户类型

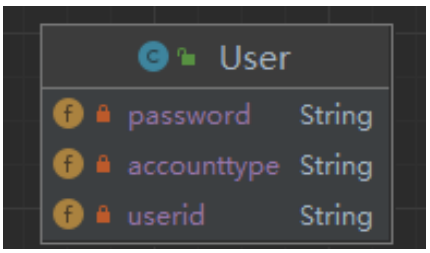


图 14 User 类

教师类，包括用户名，教师编号（主键），姓名

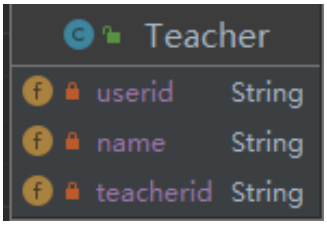


图 15 Teacher 类

学生类,包括学号、平均分、学生 id,用户 id,排名，姓名

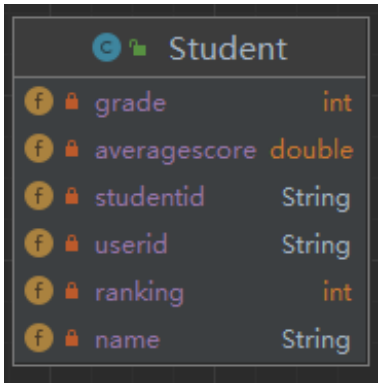


图 16 Student 类

课程类，包括课程编号、课程名

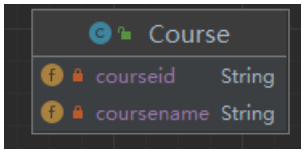


图 17 Course 类

班级类，包括班级编号、课程编号、教师编号、学生总数

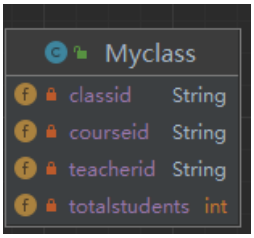


图 18 Myclass 类

分数类，也是学生选课的中间类，包括编号、班级 id，学生 id，综合成绩、期末成绩、平时成绩、期中成绩、实验成绩

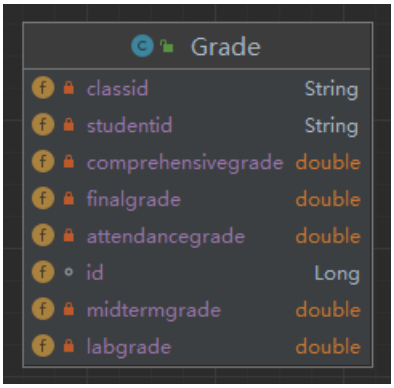


图 19 Grade 类

(2)接口

Jpa 使用接口查询数据库

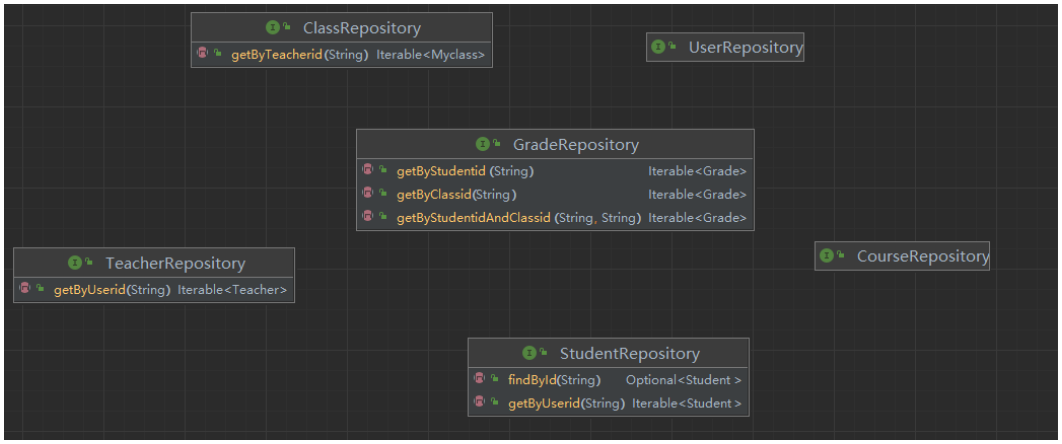


图 20 全部接口 repository

以 GradeRepository 为例，只定义根据每个变量的值进行查询

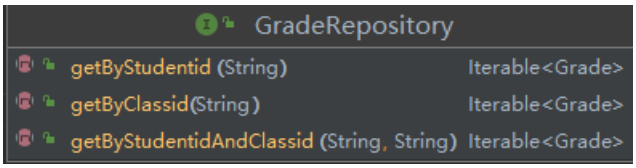


图 21 GradeRepository 接口

(3)@Controller 的类

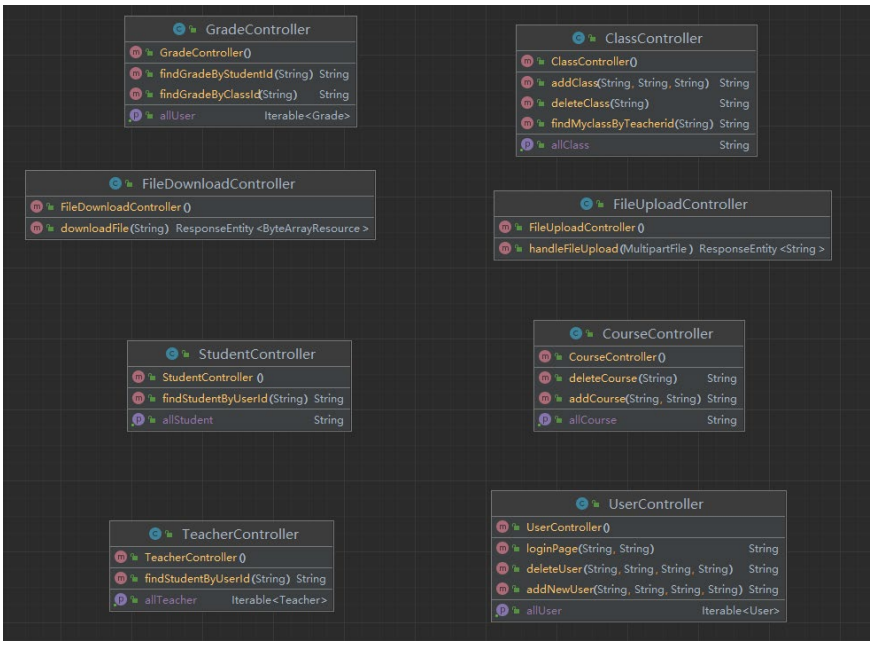


图 22　Controller 类

以 StudentController 为例，findStudentByUserid()利用 userid 查询学生信息

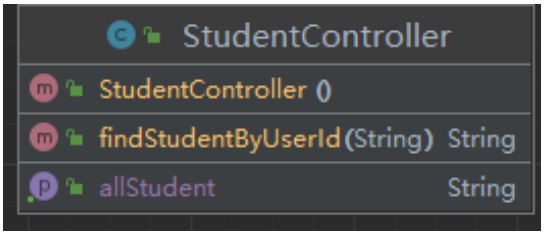


图 23 StudentController 类

（二）类关系图

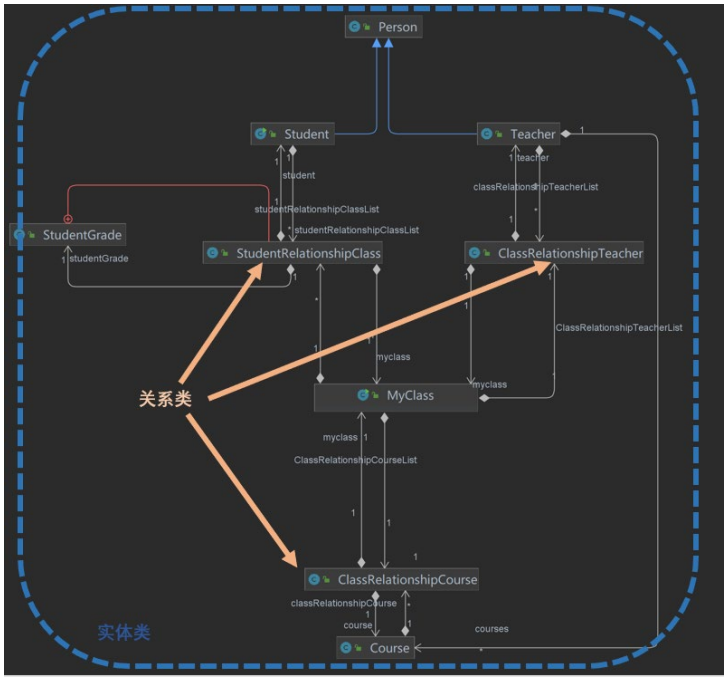


图 24 类关系图

7. 程序主要执行流程图

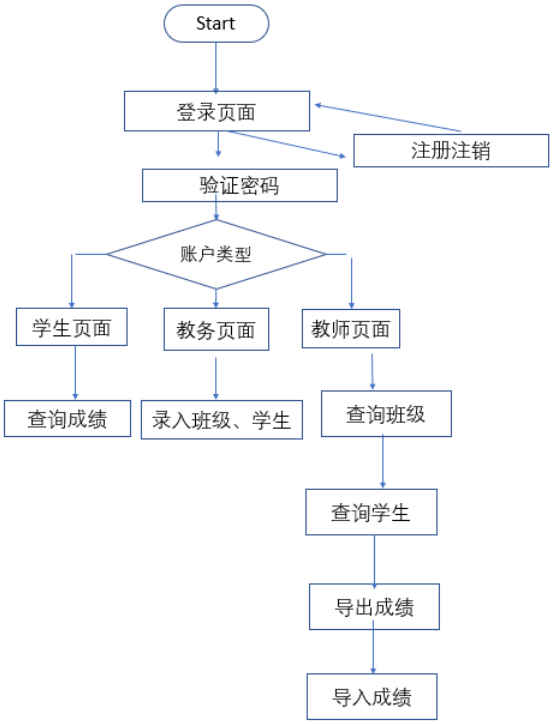


图 25 程序主要流程图

8. 核心源代码、截图

(1) 创建 User 实体，加装饰器@Entity，主键加@Id

```
@Entity // This tells Hibernate to make a table out of this class
public class User {
    @Id
    private String userid;

    3 usages
    private String accounttype;

    3 usages
    private String password;
```

(2) 创建 UserRepository 接口，spring 将自动实现并注入到一个叫做 UserRepository 的 Bean

```
// This will be AUTO IMPLEMENTED by Spring into a Bean called userRepository
// CRUD refers Create, Read, Update, Delete

3 usages
public interface UserRepository extends CrudRepository<User, String> {
}
```

(3) 创建 UserContrller，@Controller 表明该类是一个控制器，用于处理 Web

请求，并返回视图（一般用于 MVC 模式）。@RequestMapping(path = "/demo")：这个注解将该控制器的请求路径前缀设为/demo。这意味着在应用程序的路径之后，所有以/demo 开头的请求都将由这个控制器处理。@Autowired：这个注解用于自动装配依赖。在这里，它自动将 UserRepository、StudentRepository 和 TeacherRepository 的实例注入到 UserController 中。

```
@Controller    // This means that this class is a Controller
@RequestMapping(path = "/demo") // This means URL's start
public class UserController {
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private StudentRepository studentRepository;
    @Autowired
    private TeacherRepository teacherRepository;
```

(4) 创建处理用户登录请求的 Spring 控制器方法。@CrossOrigin(origins = "http://localhost:8082")：这个注解允许来自 http://localhost:8082 的跨域请求。@ResponseBody：这个注解表示方法的返回值将直接作为 HTTP 响应的主体，而不是解析为视图名称。@RequestParam String userid, @RequestParam String password:这两个注解表示 HTTP 请求中包含的 userid 和 password 参数将被绑定到方法参数上。

```
@CrossOrigin(origins = "http://localhost:8082")
@PostMapping(path = "/login") // Map ONLY GET Requests
public @ResponseBody String loginPage(@RequestParam String userid, @RequestParam String password) {
    Optional<User> result = userRepository.findById(userid);
    User user = result.orElse( other: null);
    if (user != null && Objects.equals(user.getPassword(), password)) {
        return user.getAccountType();
    }
    return "密码错误";
}
```

(5) 定义了一个名为 Grade 的实体类，用于表示学生的成绩。它使用了 JPA（Java Persistence API）来映射到数据库表。@Entity：这个注解表明 Grade 是一个 JPA 实体。JPA 实体是轻量级的持久化领域对象，它的实例将被持久化到数据库中。@Id：这个注解指定 id 字段为实体的主键。
@GeneratedValue(strategy = GenerationType.AUTO)：这个注解定义了主键生成策略。GenerationType.AUTO 表示 JPA 将自动选择合适的主键生成策略，通常是基于底层数据库的自动递增字段或序列。

```java
@Entity
public class Grade {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    Long id;

    3 usages
    private String studentid;
    3 usages
    private String classid;
    4 usages
    private double attendancegrade = 0;

    4 usages
    private double midtermgrade;

    4 usages
    private double labgrade;

    4 usages
    private double finalgrade;

    3 usages
    private double comprehensivegrade;
```

(6) 读取本地文件系统中的一个 Excel 文件，并将其封装为一个 ByteArrayResource 对象，以便通过 HTTP 响应将文件发送到客户端。Path path = Paths.get(...)：创建一个 Path 对象，表示文件系统中的路径。这条语句中指定的路径是一个绝对路径，指向 Windows 系统中的一个 Excel 文件 workbook.xlsx。byte[] data = Files.readAllBytes(path);：使用 Files.readAllBytes 方法读取文件的全部内容，并将其存储在一个字节数组中。这种方法适用于小型文件，因为它将整个文件加载到内存中。HttpHeaders headers = new HttpHeaders();：创建一个新的 HttpHeaders 对象，用于设置 HTTP 响应头

```java
Path path = Paths.get( first: "C:\\Users\\a3840\\Desktop\\gs-accessing-data-mysql-main\\complete\\src\\main\\java\\com\\exampl
byte[] data = Files.readAllBytes(path);

// 将文件内容封装为ByteArrayResource对象
ByteArrayResource resource = new ByteArrayResource(data);

// 设置HTTP响应头
HttpHeaders headers = new HttpHeaders();
String filename= "grade_of_"+classid+".xlsx";
headers.add(HttpHeaders.CONTENT_DISPOSITION, headerValue: "attachment; filename="+filename);
headers.setContentType(MediaType.APPLICATION_OCTET_STREAM);

// 返回ResponseEntity对象，包含文件内容和响应头
return ResponseEntity.ok()
        .headers(headers)
        .body(resource);
```

(7)用户登录的表单

```html
<form id="loginForm">
    <input type="text" name="userid" id="userid" placeholder="用户名/邮箱">
    <input type="password" name="password" id="password" placeholder="密码">
    <input type="button" id="loginButton" value="登录">
</form>
```

(8)表单提交发出请求

```
#loginButton').click(function() {
  var userid = $('#userid').val();
  var password = $('#password').val();

  $.ajax({
      type: 'POST',
      url: 'http://localhost:8080/demo/login',
      data: { userid: userid, password: password },
      success: function(response) {
          $('#loginMessage').text(response);
          console.log(response)
          if(response!=="密码错误")
              window.location.href = 'http://localhost:8082/jsp/'+response+'.jsp?u
      },
      error: function(xhr, status, error) {
          console.error('Error:', error);
      }
  });
```

(9)绘制表格。classes.forEach(function (myclass) {...})：遍历解析后的班级数组，每个班级对象生成一个表格行。tableHTML += '<tr>'：开始一个新的表格行。tableHTML += '<td>' + myclass.classid + '</td>'：将班级 ID 添加到表格单元格中。

```
var tableHTML = '<table border="1"><tr><th>教学班号</th><th>教职工编号</th><th>课程编号</th><th>学生总数</th><th>功能</th><th>功能</th>
classes.forEach(function (myclass) {
    tableHTML += '<tr>';
    tableHTML += '<td>' + myclass.classid + '</td>';
    tableHTML += '<td>' + myclass.teacherid + '</td>';
    tableHTML += '<td>' + myclass.courseid + '</td>';
    tableHTML += '<td>' + myclass.totalstudents + '</td>';
    var message = myclass.classid;
    var functionCall = "showWindow('" + message + "')";
    tableHTML += '<td><button onClick="' + functionCall + '" style="width: 100%; padding: 10px; \
    border: none; background-color: #007bff; color: #fff; cursor: pointer; border-radius: 3px">查看成绩</button></td>';

    var message1 = myclass.classid;
    var functionCall1 = "DownloadFile('" + message1 + "')";
    tableHTML += '<td><button onClick="' + functionCall1 + '" style="width: 100%; padding: 10px; \
    border: none; background-color: #007bff; color: #fff; cursor: pointer; border-radius: 3px">导出班级成绩</button></td>';
    tableHTML += '</tr>';
});
```

(10)绘制饼状图

```
var data = {
    labels: ["60分以下: "+ String(comprehensivedata[0]), "60分-70分: "+ String(comprehensivedata[1]), "70分-80分: "+ String(comprehensivedata[2]), "80分-90分: "
            + String(comprehensivedata[3]), "90分以上: "+ String(comprehensivedata[4])],
    datasets: [{
        data: comprehensivedata,
        backgroundColor: [
            'red',
            'blue',
            'yellow',
            'green',
            'purple'
        ]
    }]
};
var options = {
    responsive: true,
    maintainAspectRatio: false
};
var ctx = document.getElementById('myPieChart').getContext('2d');
var myPieChart = new Chart(ctx, {
    type: 'pie',
    data: data,
    options: options
});
```

四、实验结果及分析和（或）源程序调试过程（界面截图和文字）、实验总结与体会

（一）实验结果及分析

（1）访问 http://localhost:8082/

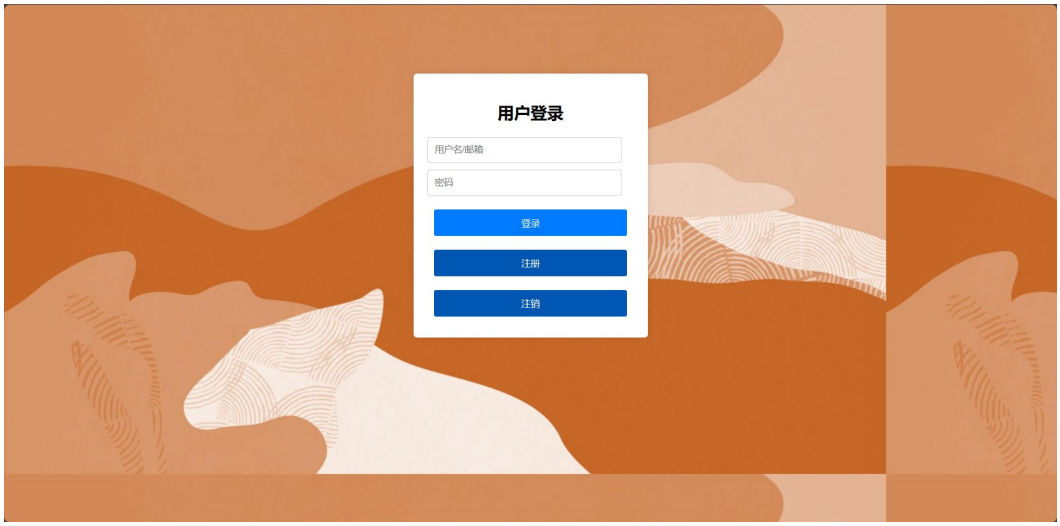


图 26 访问登录页

(2)初始化后，学生账户、教师账户、教务账号并未与学生、教师进行绑定。所以要进行注册，点击注册

图 27 注册账号

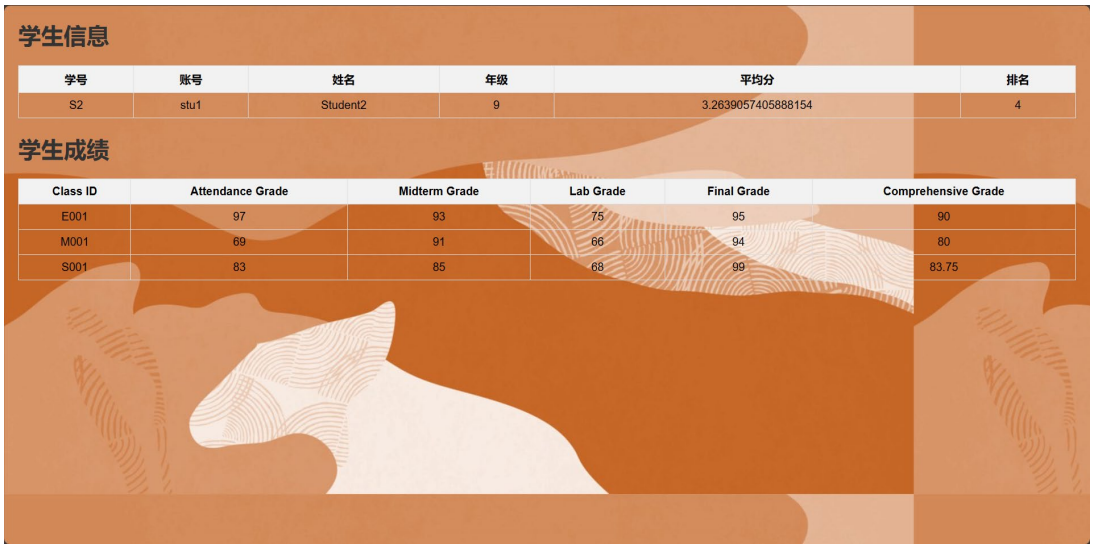
(3)首先注册一个学生账号，并将账号绑定到学工号上

图 28 绑定账号到学生，注册成功

(4)返回登录页面，登录该账户

图 29 登录账号

（5）跳转到学生页面，可以查看学生的个人学习，平均分、排名，以及各科成绩及其明细



学生信息

学号	账号	姓名	年级	平均分	排名
S2	stu1	Student2	9	3.2639057405888154	4

学生成绩

Class ID	Attendance Grade	Midterm Grade	Lab Grade	Final Grade	Comprehensive Grade
E001	97	93	75	95	90
M001	69	91	66	94	80
S001	83	85	68	99	83.75

图 30 学生主页

（6）同理注册教师账户并登录

图 31 注册教师账号

(7) 查看教师信息和所在教学班信息

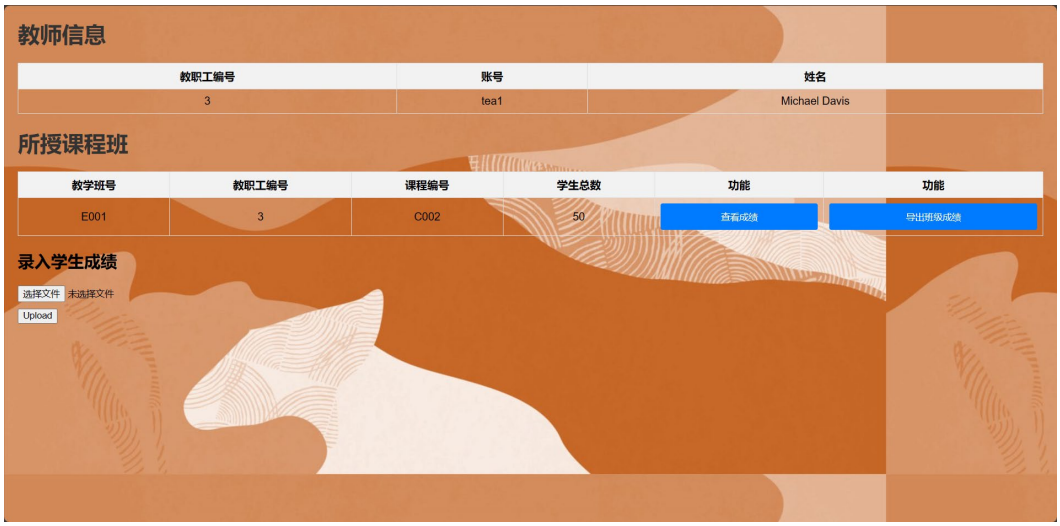


图 32　教师主页

(8) 点击查看成绩，可以看到班级成绩一览和统计的饼状图

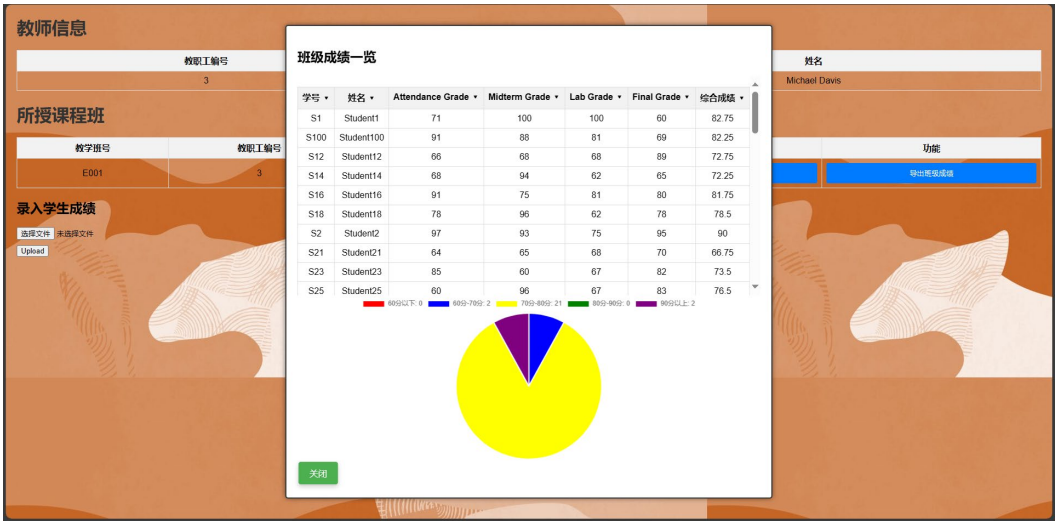


图 33　班级成绩一览

(9) 点击任意 1 列的表头都会排序，再点时反向排序

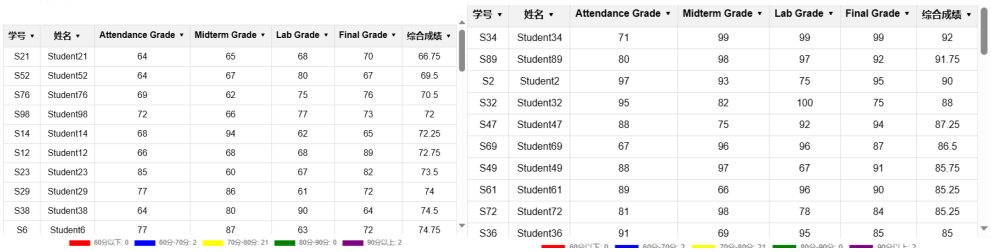


图 34 排序

(10) 点击绿色按钮关闭，关闭成绩一览，再点击导出学生成绩，会下载 xlsx 的文件，

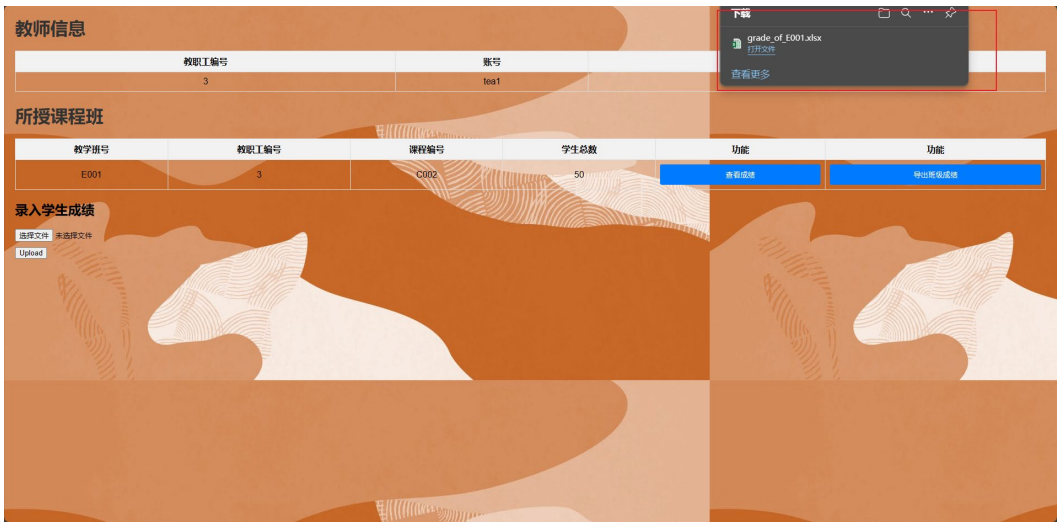
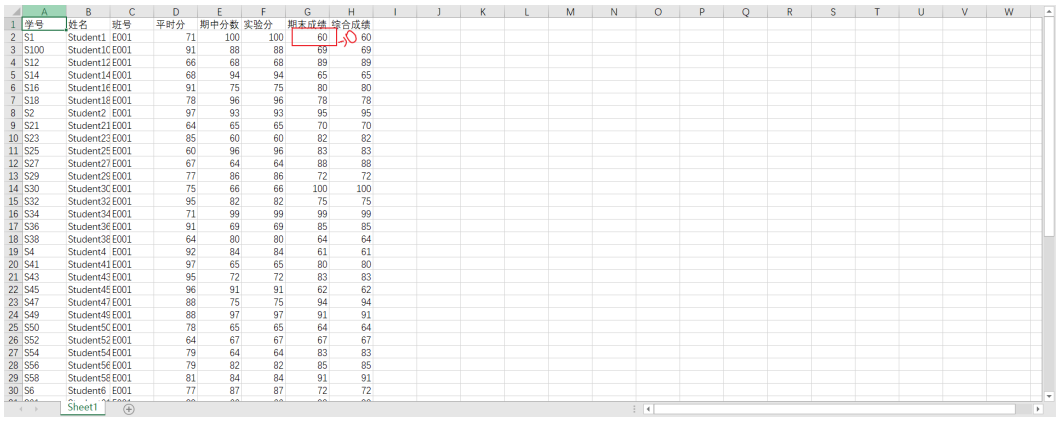


图 35　导出成绩

(11) 用 excel 打开下载的成绩，进行修改，这里将第 1 名学生的 finalgrade 成绩改为 0



学号	姓名	班号	平时分	期中分数	实验分	期末成绩	综合成绩
S1	Student1	E001	71	100	100	60	60
S100	Student10	E001	91	88	88	69	69
S12	Student12	E001	66	68	68	89	89
S14	Student14	E001	68	94	94	65	65
S16	Student16	E001	91	75	75	80	80
S18	Student18	E001	78	96	96	78	78
S2	Student2	E001	97	93	93	95	95
S21	Student21	E001	64	65	65	70	70
S23	Student23	E001	85	60	60	82	82
S25	Student25	E001	60	96	96	83	83
S27	Student27	E001	67	64	64	88	88
S29	Student29	E001	77	86	86	72	72
S30	Student30	E001	75	66	66	100	100
S32	Student32	E001	95	82	82	75	75
S34	Student34	E001	71	99	99	99	99
S36	Student36	E001	91	69	69	85	85
S38	Student38	E001	64	80	80	64	64
S4	Student4	E001	92	84	84	61	61
S41	Student41	E001	97	65	65	80	80
S43	Student43	E001	95	72	72	83	83
S45	Student45	E001	96	91	91	62	62
S47	Student47	E001	88	75	75	94	94
S49	Student49	E001	88	97	97	91	91
S50	Student50	E001	78	65	65	64	64
S52	Student52	E001	64	67	67	67	67
S54	Student54	E001	79	64	64	83	83
S56	Student56	E001	79	82	82	85	85
S58	Student58	E001	81	84	84	91	91
S6	Student6	E001	77	87	87	72	72

Sheet1

图 36 下载的 xlsx

(12) 回到教师页面，点击录入学生成绩处的选择文件，选择刚才修改后的 xlsx 文件，点击 Upload

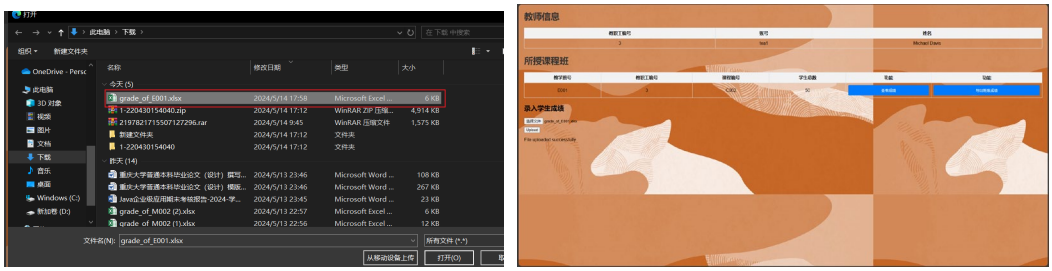


图 37 上传文件批量录入

(13) 再次点击查看学生成绩，发现学生成绩已经修改，通过这种方式，就可以批量录入学生成绩

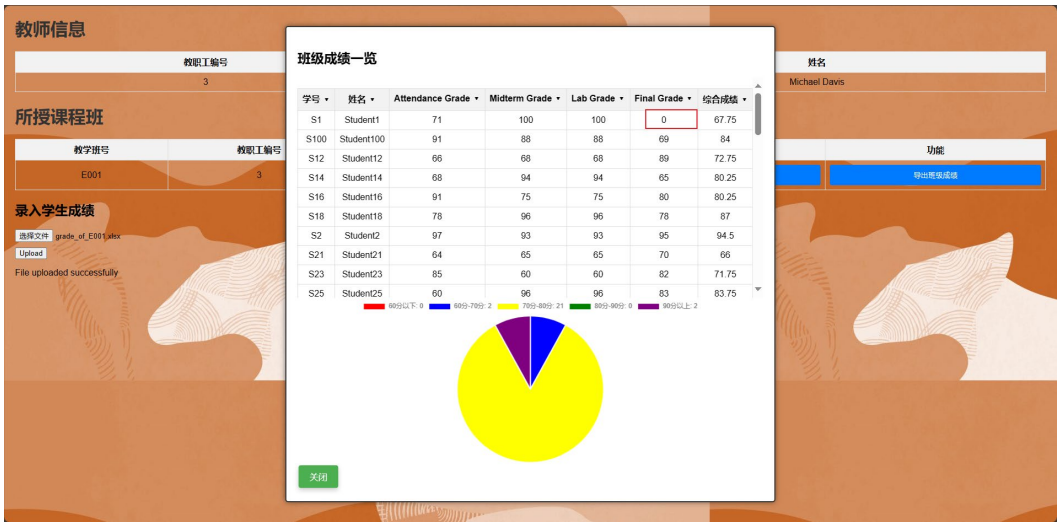


图 38　查看修改后成绩

(14) 再注册 1 个教务账号，不用绑定学工号，注册成功登录账号

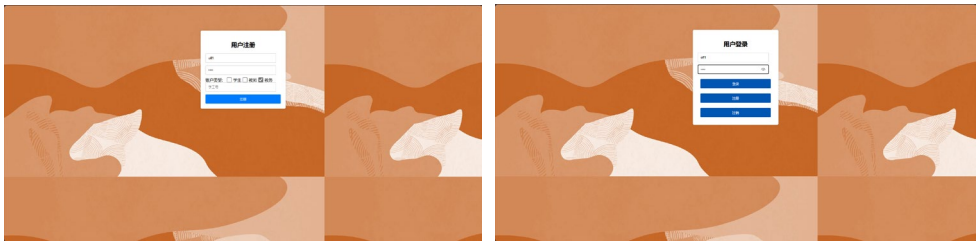


图 39　注册教务账号

(15) 可以查看课程信息、班级信息、以及学生信息

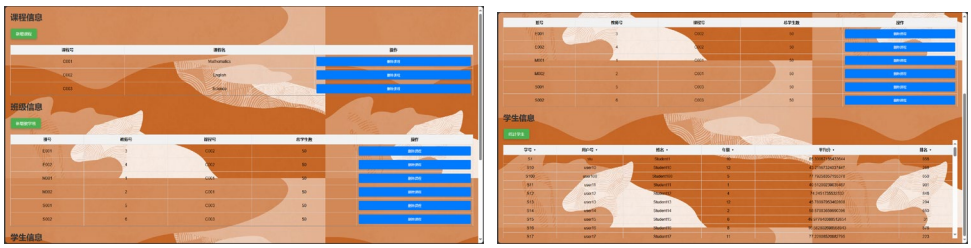


图 40　教务主页

(16) 点击新增课程，输入课程信息，点击添加，可以看到已经新增了一门

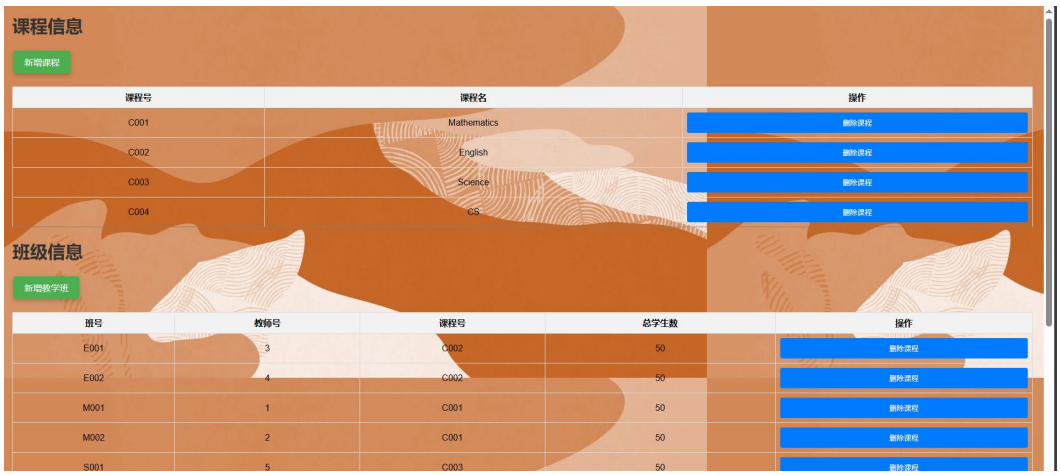


图 41 新增课程

(17) 点击删除课程即可删除该课程

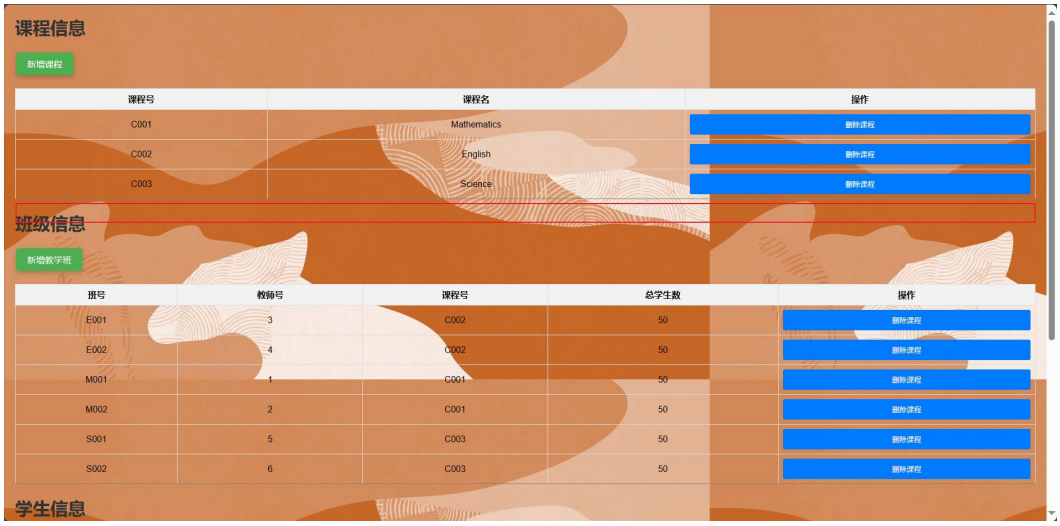


图 42 删除账号

(18) 点击统计学生信息，弹出全校学生统计

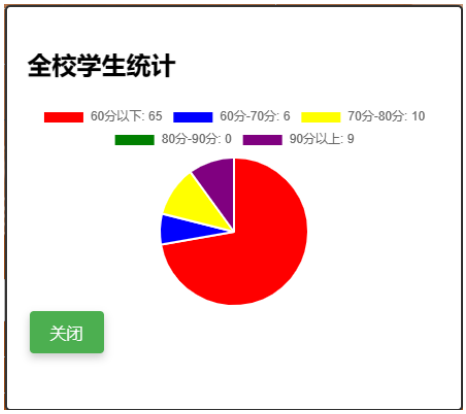


图 43 统计全校学生

(19) 销户，返回登录页面，点击注销，

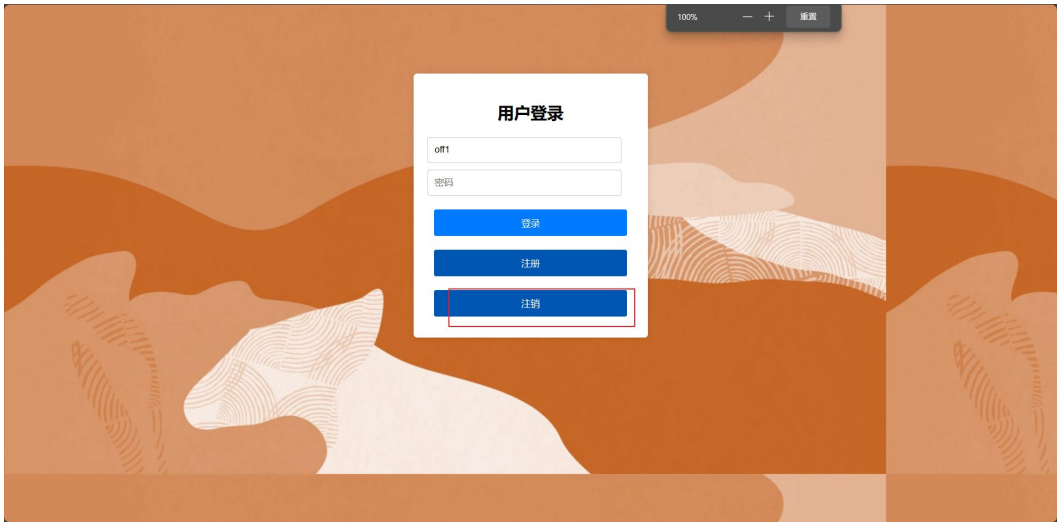


图 44 销户

(20) 输入信息

图 45 销户

(21)返回登录页面，就不能注册了

图 46 尝试登录已经销户的账号

（二）实验总结与体会

总结和体会

在本项目的开发过程中，我深刻体会到了前后端分离架构的优势。通过 SpringBoot 和 Spring Data JPA 的结合，不仅简化了数据库操作，还提高了系统的整体性能和可维护性。同时，前端利用 AJAX 技术实现了动态交互，提升了用户体验。

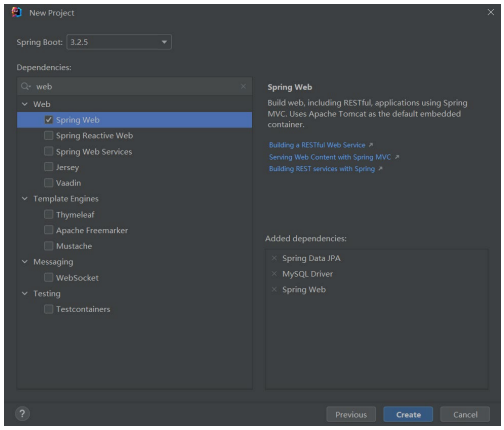
批量操作功能的实现，让成绩管理变得更加高效和便捷，特别是 Excel 文件的批量导入导出，极大地减少了人工操作的时间成本。数据统计功能的引入，提供了直观的可视化数据分析工具，帮助用户更好地理解成绩分布情况。

在项目管理方面，Maven 和 Git 的使用规范了项目的构建和版本控制流程，确保了团队协作的有序进行。通过设计模式的合理应用，项目在代码复用性、扩展性和维护性方面得到了显著提升。

总体而言，本项目不仅实现了预期的功能目标，还在开发过程中积累了宝贵的实践经验。这些经验将对我未来的项目开发和技术提升起到积极的推动作用。

遇到的问题和解决：

1.在配置 spring data jpa 过程中，Spring intitalizr，配置 Spring Data JPA 依赖不能正常运行



解决方案：项目结构写的不对，没有正确配置 controller 和 repository，使得不能正确注入 repository。从官网下载显目实例，顺利解决。