目录

[一、 应用背景 3](#_Toc144905398)

[1.1 背景说明 3](#_Toc144905399)

[1.2 开发说明 3](#_Toc144905400)

[二、 需求分析 3](#_Toc144905401)

[2.1 系统目标 3](#_Toc144905402)

[2.2 需求描述 3](#_Toc144905403)

[2.3 功能分析 4](#_Toc144905404)

[三、 概念结构设计 4](#_Toc144905405)

[3.1 实体分析 4](#_Toc144905406)

[3.2 分ER图 4](#_Toc144905407)

[3.3 ER图集成 6](#_Toc144905408)

[四、 逻辑结构设计 7](#_Toc144905409)

[4.1 关系模式 7](#_Toc144905410)

[4.2 关系表设计 7](#_Toc144905411)

[4.3 视图设计 9](#_Toc144905412)

[4.4 索引设计 9](#_Toc144905413)

[五、 物理实现 9](#_Toc144905414)

[5.1 数据库的SQL脚本 9](#_Toc144905415)

[六、 应用开发 10](#_Toc144905416)

[6.1 系统部署 10](#_Toc144905417)

[6.1.1 系统架构设计 12](#_Toc144905418)

[6.1.2 通信接口设计 12](#_Toc144905419)

[6.2 系统展示 14](#_Toc144905420)

[6.2.1 登录功能 14](#_Toc144905421)

[6.2.2 班级信息管理 15](#_Toc144905422)

[6.2.3 管理员信息管理 16](#_Toc144905423)

[6.2.4 学生信息管理 17](#_Toc144905424)

[6.2.5 班费查询 17](#_Toc144905425)

[6.2.6 班费信息明细 18](#_Toc144905426)

[七、 设计总结 18](#_Toc144905427)

[7.1 小组详细分工 18](#_Toc144905428)

[7.2 课程设计的个人总结 19](#_Toc144905429)

[7.2.1 在数据库的构建中 19](#_Toc144905430)

[7.3 在后端的构建中 19](#_Toc144905431)

[7.4 在前端的构建中 20](#_Toc144905432)

# 应用背景

## 背景说明

在学校教育中，班费是指由学生自行筹集并用于班级活动和事务的资金。这些资金通常由学生每学期或每年向班级收取一定金额，并由班级委员会或班级指导老师负责管理和使用。然而，传统的班费管理方式存在着一些问题，如资金使用不透明、记录不准确、管理不便等。为了解决这些问题，开发了班费管理系统。

班费管理系统是一种基于计算机和网络技术的应用系统，旨在提供一个高效、透明、便捷的班费管理平台。该系统通过数字化管理班费，使得资金的流动和使用更加规范和可追溯。以下是该系统的背景说明：

透明度和公正性：传统的班费管理方式容易引发资金使用不透明的问题，导致学生对班费的使用情况产生疑虑。班费管理系统通过记录每一笔资金的流入和流出，以及详细的使用说明，确保班级成员可以清晰地了解资金的使用情况，增加管理的透明度和公正性。

管理效率：传统的班费管理方式通常需要手工记录和核算，耗费时间和精力。而班费管理系统利用计算机和网络技术，实现自动化的资金流动和记录，大大提高了管理的效率。班级委员会或指导老师可以轻松地查看资金的余额、支出明细和项目申请，减少了繁琐的手动操作。

综上所述，班费管理系统通过引入计算机和网络技术，解决了传统班费管理方式存在的问题，提高了管理的效率和透明度，增强了资金的安全性，同时为班级成员提供了更好的参与和监督机制，促进了班级活动和事务的有序进行。

## 开发说明

系统名称：班费管理系统

开发人员：张乐、衷杰

# 需求分析

## 系统目标

本系统为班费管理系统。在系统中，管理员可以查看班级信息，对班级信息的增加、修改和删除；管理员可以增加、修改和删除管理员的用户名或密码；管理员可以导入学生信息；管理员可以对增加班费收入、支出明细记录以及对收支情况查询与统计。

## 需求描述

班级管理员表的需求：

1. 登录班费管理系统
2. 添加更多的班级管理员
3. 更新和删除班级管理员信息
4. 查看所有的班级管理员信息

班级表的需求：

1. 添加班级的信息
2. 更新和删除班级信息
3. 查看所有班级相关信息
4. 对班级信息进行搜索

学生表的需求：

1. 对学生信息进行导入的操作
2. 查看所有学生的相关信息

班费表的需求：

1. 添加每个班级的班费收入、支出明细记录
2. 查看所有班级的班费记录

班费明细表的需求：

1. 添加每个学生对班级班费的使用情况
2. 查看班费表的班费具体使用情况

## 功能分析

|  |  |  |
| --- | --- | --- |
| **使用者** | **功能** | **功能详情** |
| 管理员 | 用户登录 | 根据用户名、密码登录 |
| 管理员用户增加 | 新增管理员用户名和密码 |
| 增删改班级 | 根据每个班级的唯一id进行相关的增删改查 |
| 导入学生信息 | 增加学生信息，导入相关信息，每个学号id绑定唯一的班级id |
| 增加班级班费的信息和查询 | 根据唯一的班级id，增加每个班级对应的班费 |
| 增加学生对班费的使用查询 | 每个学生绑定对应班级，查询学生对班费的使用情况 |

# 概念结构设计

## 实体分析

|  |  |
| --- | --- |
| **实体类** | **描述** |
| Classadmin 班级管理员 | 可增删改所用信息 |
| Grade 班级信息 | 管理班级信息 |
| Student 学生信息 | 管理学生信息 |
| Classfee 班费信息 | 管理班费支出收入信息 |
| Feedetail 班费详细信息 | 管理学生使用班费的详细情况 |

## 分ER图



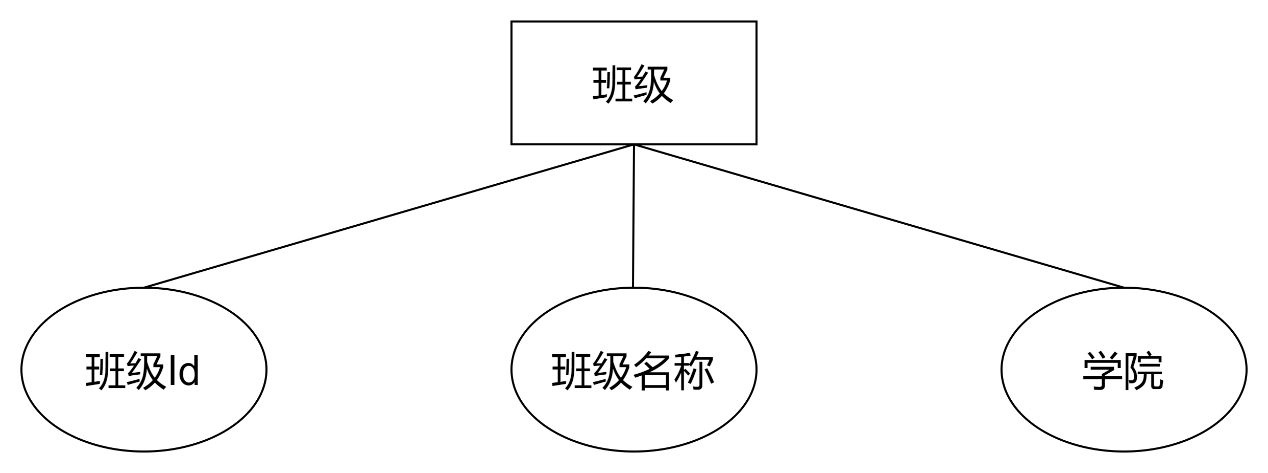


图1：班级E-R图

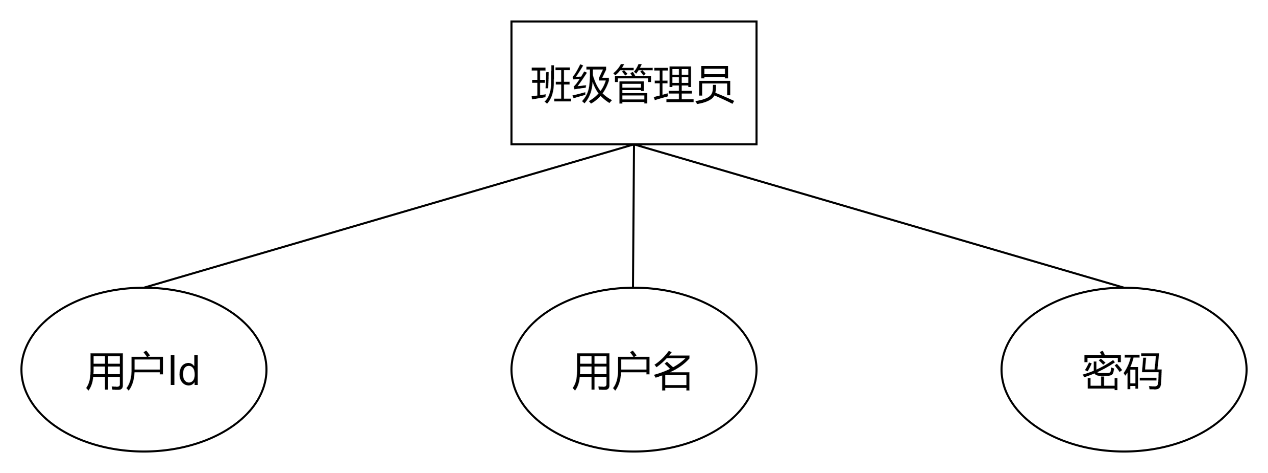


图2：班级管理员E-R图

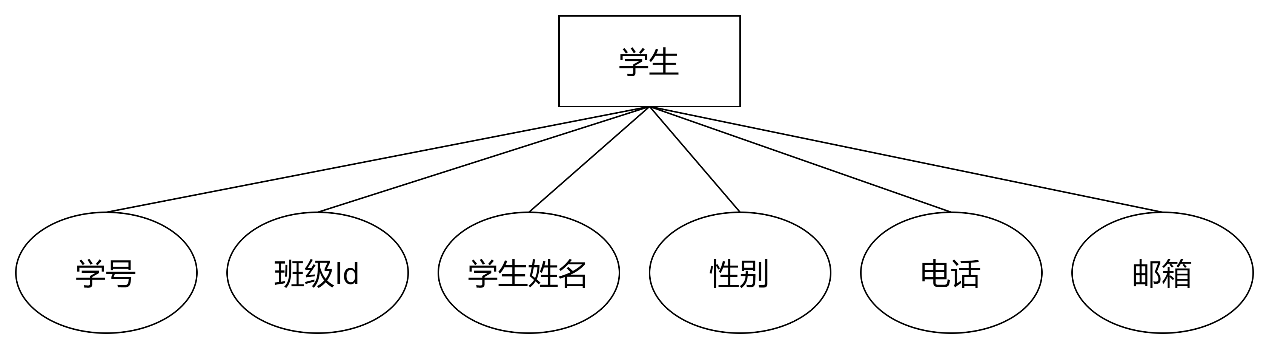


图3：学生E-R图

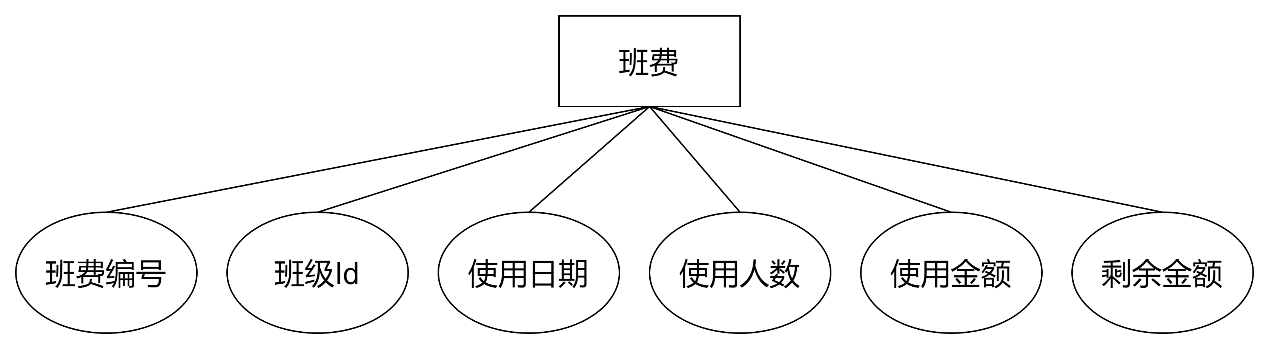


图4：班费E-R图

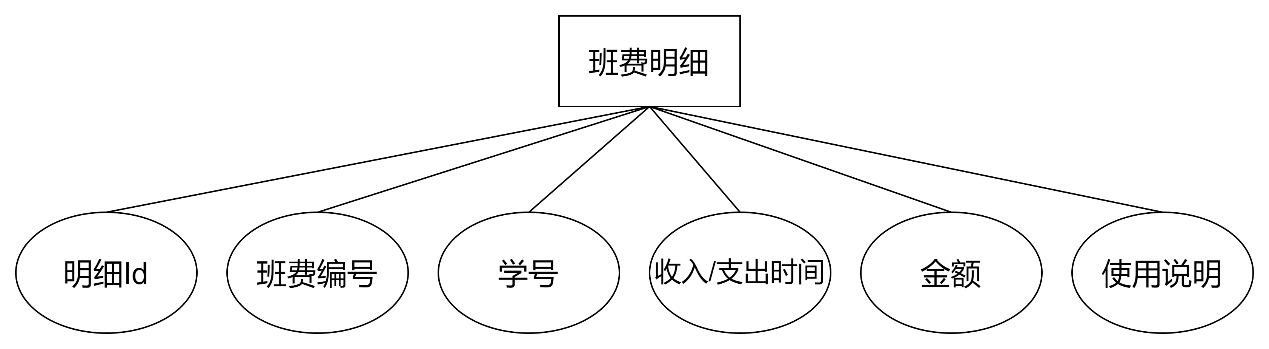


图5：班费明细E-R图

## ER图集成

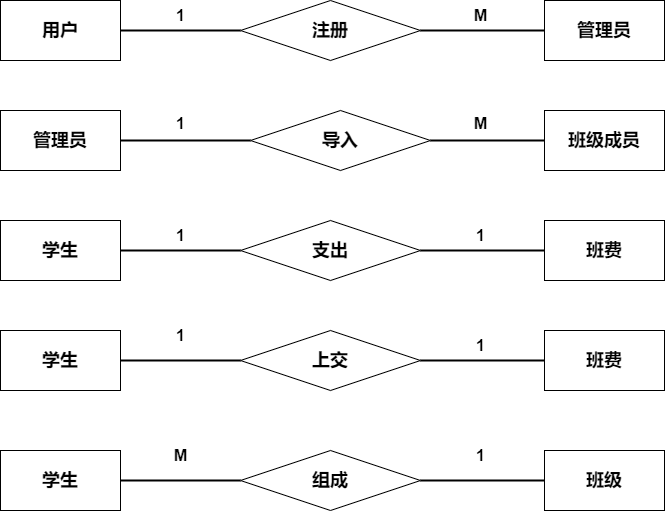


图6：局部E-R图

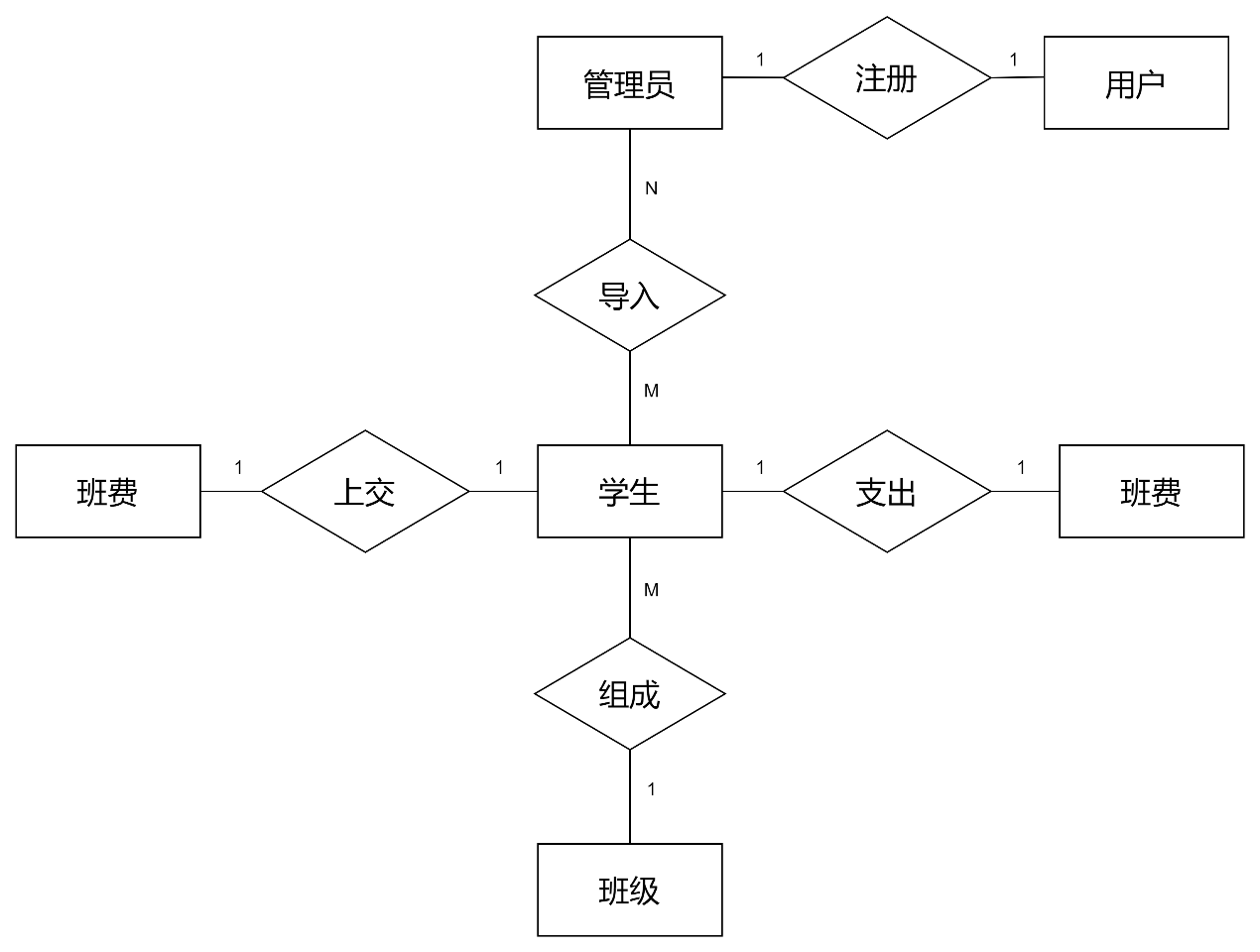


图7：整体E-R图

# 逻辑结构设计

## 关系模式

班级表（班级编号、班级名称、学院）**主键**：班级编号

班级管理员用户表（用户ID、用户名、密码）**主键**：用户ID

学生表（学号、班级编号、姓名、性别、电话、邮箱）**主键**：学号 **外键**：班级编号

班费表（班费编号、班级编号、使用日期、使用人数、使用金额、剩余金额）

**主键**：班费编号 **外键**：班级编号

班费明细表（明细号、班费编号、学号、支出/收入金额、支出/收入时间、使用说明）

**主键**：明细号 **外键**：班费编号、学号

## 关系表设计

表1：班级表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 约束 | 说明 |
| classId | varchar | 128 | 主键 | 班级Id |
| className | varchar | 128 | 非空 | 班级名称 |
| academy | varchar | 128 | 非空 | 学院 |

表2：班级管理员用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 约束 | 说明 |
| userId | int | 自增 | 主键 | 用户Id |
| username | varchar | 128 | 非空 | 用户名 |
| password | varchar | 128 | 非空 | 用户密码 |

表3：学生表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 约束 | 说明 |
| studentId | varchar | 128 | 主键 | 学号 |
| classId | varchar | 128 | 外键 | 班级Id |
| sname | varchar | 128 | 非空 | 学生姓名 |
| gender | varchar | 10 | 非空 | 性别 |
| phone | varchar | 128 | 非空 | 电话 |
| email | varchar | 128 |  | 邮箱 |

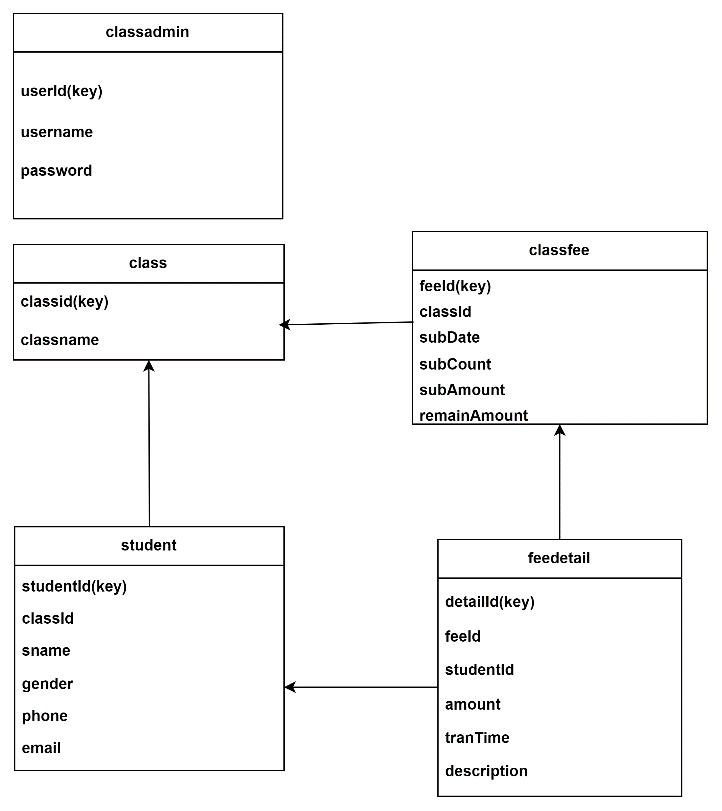
表4：班费表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 约束 | 说明 |
| feeId | int | 自增 | 主键 | 班费编号 |
| classId | varchar | 128 | 外键 | 班级Id |
| subDate | timestamp |  | 非空 | 使用日期 |
| subCount | int |  | 非空 | 使用人数 |
| subAmount | float |  | 非空 | 使用明细，+收入 -支出 |
| remainAmount | float |  | 非空 | 剩余金额 |

表5：班费明细表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 约束 | 说明 |
| detailId | int | 自增 | 主键 | 明细Id |
| feeId | int |  | 外键 | 班费编号 |
| studentId | varchar | 128 | 外键 | 学号 |
| amount | float |  | 非空 | 使用金额，+收入 -支出 |
| tranTime | timestamp |  | 非空 | 使用时间 |
| description | varchar | 128 | 非空 | 使用说明 |

## 视图设计



# 物理实现

## 数据库的SQL脚本

班级表的建立

**create** **table** **class**

(

classId varchar(128) **not** **null** **comment** '班级Id' **primary** **key**,

className varchar(128) **collate** utf8\_general\_ci **not** **null** **comment** '班级名称',

academy varchar(128) **collate** utf8\_general\_ci **not** **null** **comment** '学院名称'

) CHARACTER **SET** utf8mb4

**COLLATE** utf8mb4\_general\_ci, **comment** '班级表';

班级管理员用户表的建立

**create** **table** classadmin

(

userId int auto\_increment **comment** '用户Id' **primary** **key**,

username varchar(128) **not** **null** **comment** '用户名',

password varchar(128) **not** **null** **comment** '用户密码'

) CHARACTER **SET** utf8mb4

**COLLATE** utf8mb4\_general\_ci, **comment** '班级管理员用户表';

班费表的建立

**create** **table** classfee

(

feeId int auto\_increment **comment** '班费编号' **primary** **key**,

classId varchar(128) **not** **null** **comment** '班级Id',

subDate **timestamp** **default** **CURRENT\_TIMESTAMP** **not** **null** **on** **update** **CURRENT\_TIMESTAMP** **comment** '上交日期',

subCount int **not** **null** **comment** '使用人数',

subAmount float **not** **null** **comment** '收入、支出明细 +收入 -支出',

remainAmount float **not** **null** **comment** '剩余金额',

**constraint** classId\_1 **foreign** **key** (classId) **references** **class** (classId)

) CHARACTER **SET** utf8mb4

**COLLATE** utf8mb4\_general\_ci,**comment** '班费表';

学生表的建立

**create** **table** student

(

studentId varchar(128) **not** **null** **comment** '学号' **primary** **key**,

classId varchar(128) **not** **null** **comment** '班级Id',

sname varchar(128) charset utf8 **not** **null** **comment** '学生姓名',

gender varchar(10) charset utf8 **not** **null** **comment** '性别',

phone varchar(128) charset utf8 **not** **null** **comment** '电话',

email varchar(128) charset utf8 **not** **null** **comment** '邮箱',

**constraint** classId\_0 **foreign** **key** (classId) **references** **class** (classId)

) CHARACTER **SET** utf8mb4

**COLLATE** utf8mb4\_general\_ci,**comment** '学生表';

班费明细表的建立

**create** **table** feedetail

(

detailId int auto\_increment **comment** '明细Id' **primary** **key**,

feeId int **not** **null** **comment** '班费编号',

studentId varchar(128) **not** **null** **comment** '学号',

amount float **not** **null** **comment** '支出/收入金额 +代表收入 -代表支出',

tranTime **timestamp** **default** **CURRENT\_TIMESTAMP** **not** **null** **comment** '支出/收入时间',

description varchar(128) charset utf8 **not** **null** **comment** '使用说明',

**constraint** feeId **foreign** **key** (feeId) **references** classfee (feeId),

**constraint** studentId **foreign** **key** (studentId) **references** student (studentId)

) CHARACTER **SET** utf8mb4

**COLLATE** utf8mb4\_general\_ci,**comment** '班费明细表';

# 应用开发

## 系统部署

这是一个前后端分离，并连接数据库的小项目。数据库端采用MySQL搭建，大致构建如上图ER图所示。后端采用SpringBoot 集成MybatisPlus操作数据库。前端采用vue2搭载vue-cli脚手架实现。可实现用户数据，班级数据，学生数据，交易明细数据的增删改查，以及管理员登录等多项基本功能。

在后端，使用maven，通过配置pom.xml的文件来配置依赖属性。

maven工程在pom.xml中添加所需的jar包的坐标，避免jar直接引入进来，在需要用到jar包的时候，只要查找pom.xml文件，再通过pom.xml中的坐标，到一个专门用于存放jar包的仓库中根据坐标从而找到这些jar包，再把这些jar包拿去运行。

配置文件为.yml（使用更加方便的yml文件，而不是默认的properties文件）。

后端创建实体类（domain），分别映射到数据库中所创建的5个表格。

拿Classadmin举例如下：

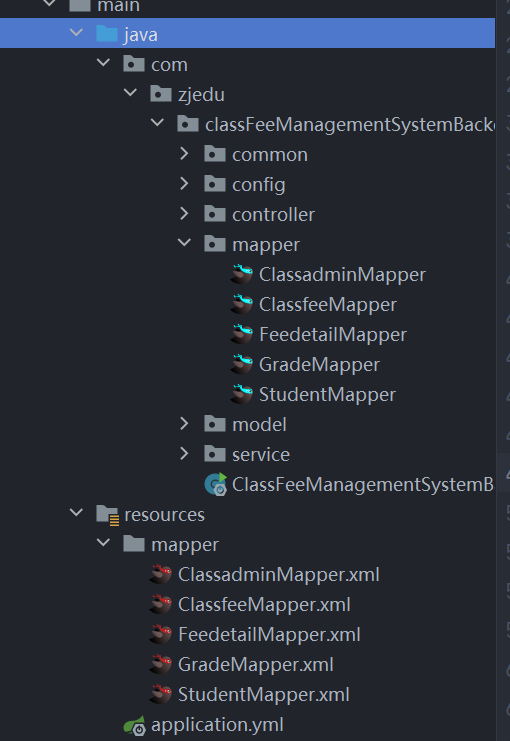
|  |  |
| --- | --- |
|  |  |

后端创建controller层，分别对应5个控制器：用来封装一些方法供前端调用。

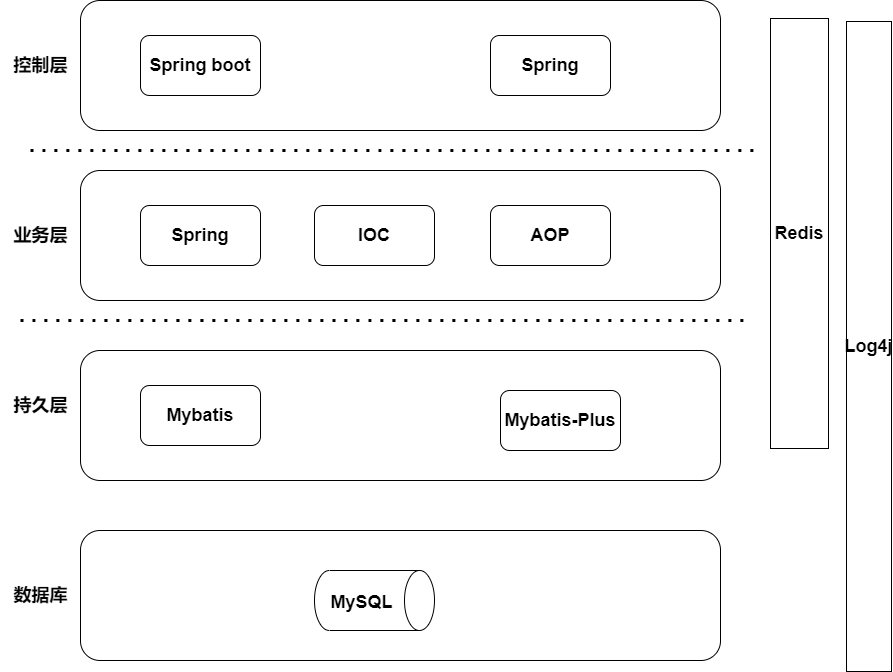
拿UserController举例：

|  |  |
| --- | --- |
|  |  |

我们在后端，引入MyBatis持久层框架。使用注解整合MyBatis：创建Mapper接口文件:@Mapper。通过直接调用库中的方法，而减少直接书写SQL语句。



### 系统架构设计



系统架构图

该技术架构自上而下主要分为四层，最上层为控制层、其次为业务层、然后是持久层、最后一层是数据层。

控制层主要负责参数检验、权限验证等操作，主要处理的是前端发送的HTTP请求。开发方式主要是使用SpringBoot框架进行开发，由于SpringBoot自动配置的特点可以使后端开发更加快捷。

业务层主要负责业务逻辑的处理，由于Spring框架具有中IOC的特点，不仅更好的实现了低耦合的特性，同时可以使我们的代码更加简洁。通过Spring框架中的AOP机制解决了日志记录、异常处理的问题，使我们只要注重于我们的业务逻辑代码而不用关注与业务无关的代码。

持久层主要负责对业务中所涉及到的相关的数据的处理，采用MyBatis-Plus框架来完成对数据库中数据和Java对象之间的映射，代码层次更加清晰，方便后期代码的维护。

数据层主要使用了关系型数据库MySQL对系统的数据进行存储

### 通信接口设计

通过设计前后端通信接口，可以提高工作效率，使前后端分工更加明确。

本次接口的设计风格是按照目前业界最流行的RESTful风格进行设计，对资源的请求方式为GET、PUT、POST、DELETE，分别对应：获取资源、更新资源、推送资源、删除资源。其中参数都是以JSON的形式传递

接口的返回对象只有一个Result，包含三个属性：

1. code: 返回的状态值，成功返回200，失败返回201。
2. description: 请求后返回的描述信息。
3. detail: 成功后返回的信息。

接下来对班费管理系统进行前后端通信接口设计，登录模块接口具体设计如下表所示:

登录模块接口设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 请求方法 | 请求地址 | 请求参数 | 返回结果 |
| 登录 | POST | /user/login | Classadmin | Result |
| 首页列表 | GET | /user/info | token | Result |
| 注销 | POST | /user/logout | token | Result |

班级模块具体设计如下表所示：

班级模块接口设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 请求方法 | 请求地址 | 请求参数 | 返回结果 |
| 添加 | POST | /class/add | Grade | Result |
| 更新 | PUT | /class/update | Grade | Result |
| 删除 | DELETE | /class/delete | Grade | Result |
| 展示 | GET | /class/list | 无 | Result |
| 搜索 | GET | /class/search | className | Result |

班级管理员模块具体设计如下表所示：

班级管理员模块接口设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 请求方法 | 请求地址 | 请求参数 | 返回结果 |
| 添加 | POST | /user/add | Classadmin | Result |
| 更新 | PUT | /user/update | Classadmin | Result |
| 删除 | DELETE | /user/delete | Classadmin | Result |
| 展示 | GET | /user/list | 无 | Result |

学生成员模块具体设计如下表所示：

学生成员模块接口设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 请求方法 | 请求地址 | 请求参数 | 返回结果 |
| 添加 | POST | /student/add | Student | Result |
| 展示 | GET | / student /list | 无 | Result |

班费使用情况模块具体设计如下表所示：

班费使用情况模块接口设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 请求方法 | 请求地址 | 请求参数 | 返回结果 |
| 添加 | POST | /classfee/add | Classfee | Result |
| 展示 | GET | / classfee /list | 无 | Result |

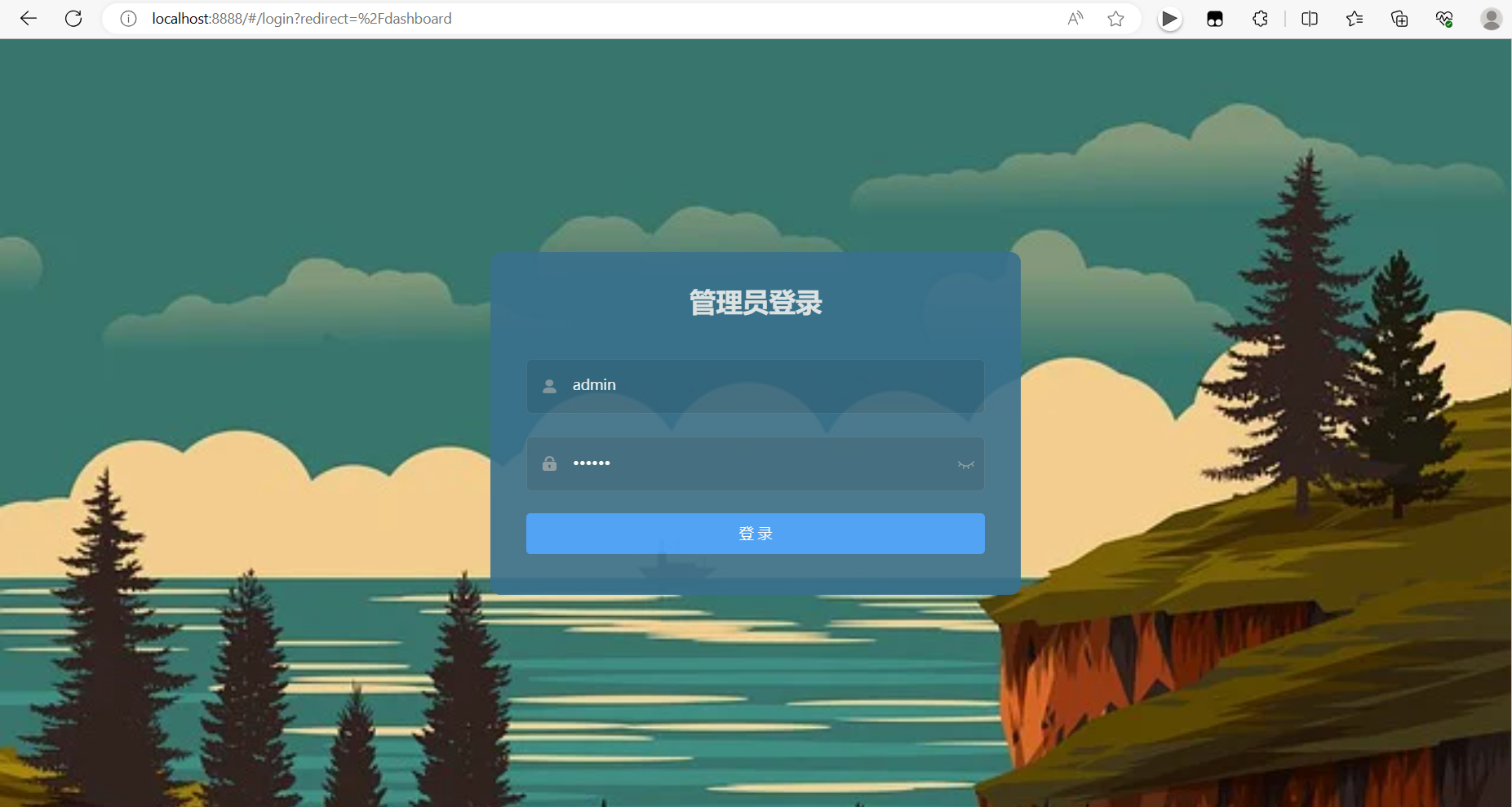
班费具体明细模块具体设计如下表所示：

班费具体明细模块接口设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 请求方法 | 请求地址 | 请求参数 | 返回结果 |
| 添加 | POST | /feedetail/add | Feedetail | Result |
| 展示 | GET | / feedetail /list | 无 | Result |

## 系统展示

### 登录功能

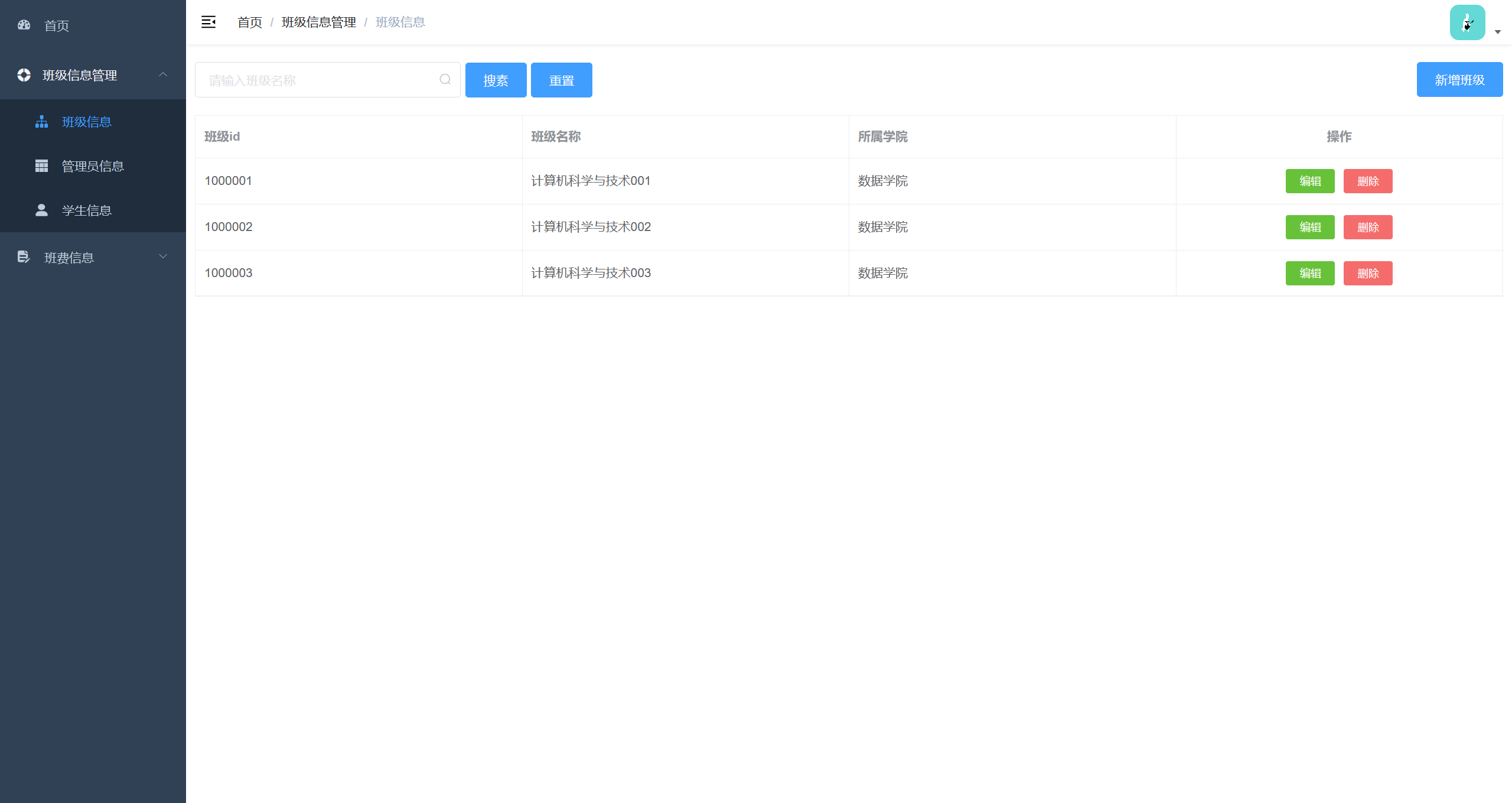


输入数据库内已有的用户名和密码即可成功登录。

登录主页面展示

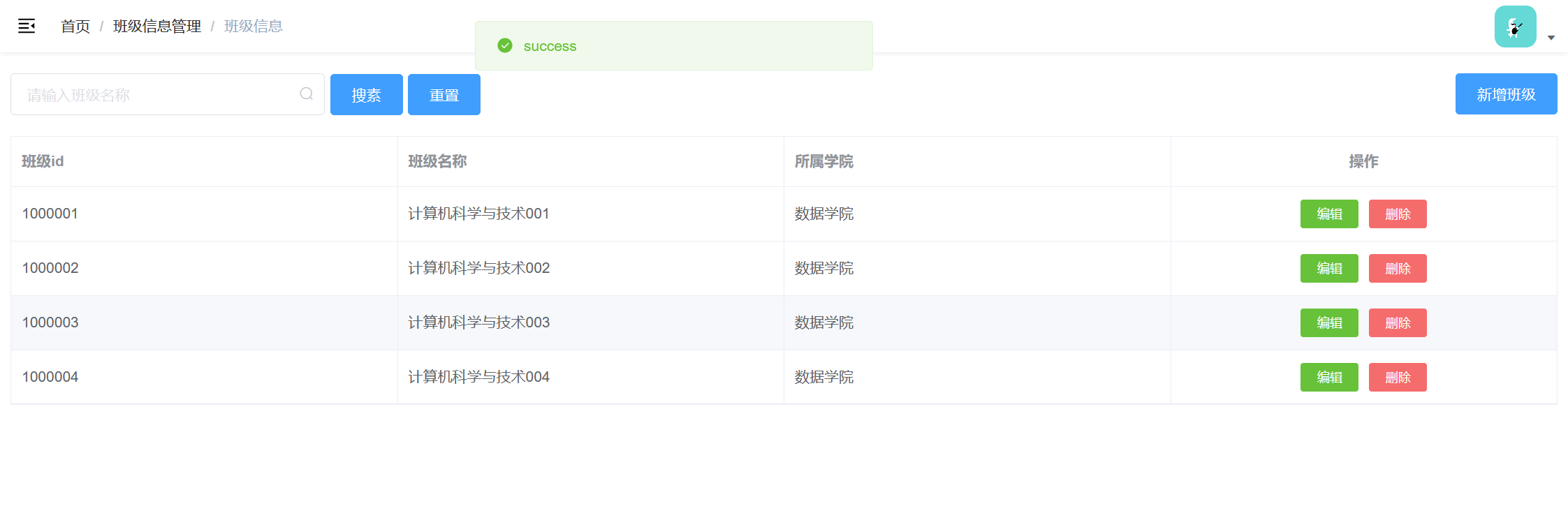


### 班级信息管理



此页面可以实现对班级信息的增加、修改、查询、删除。

增加页面



修改页面



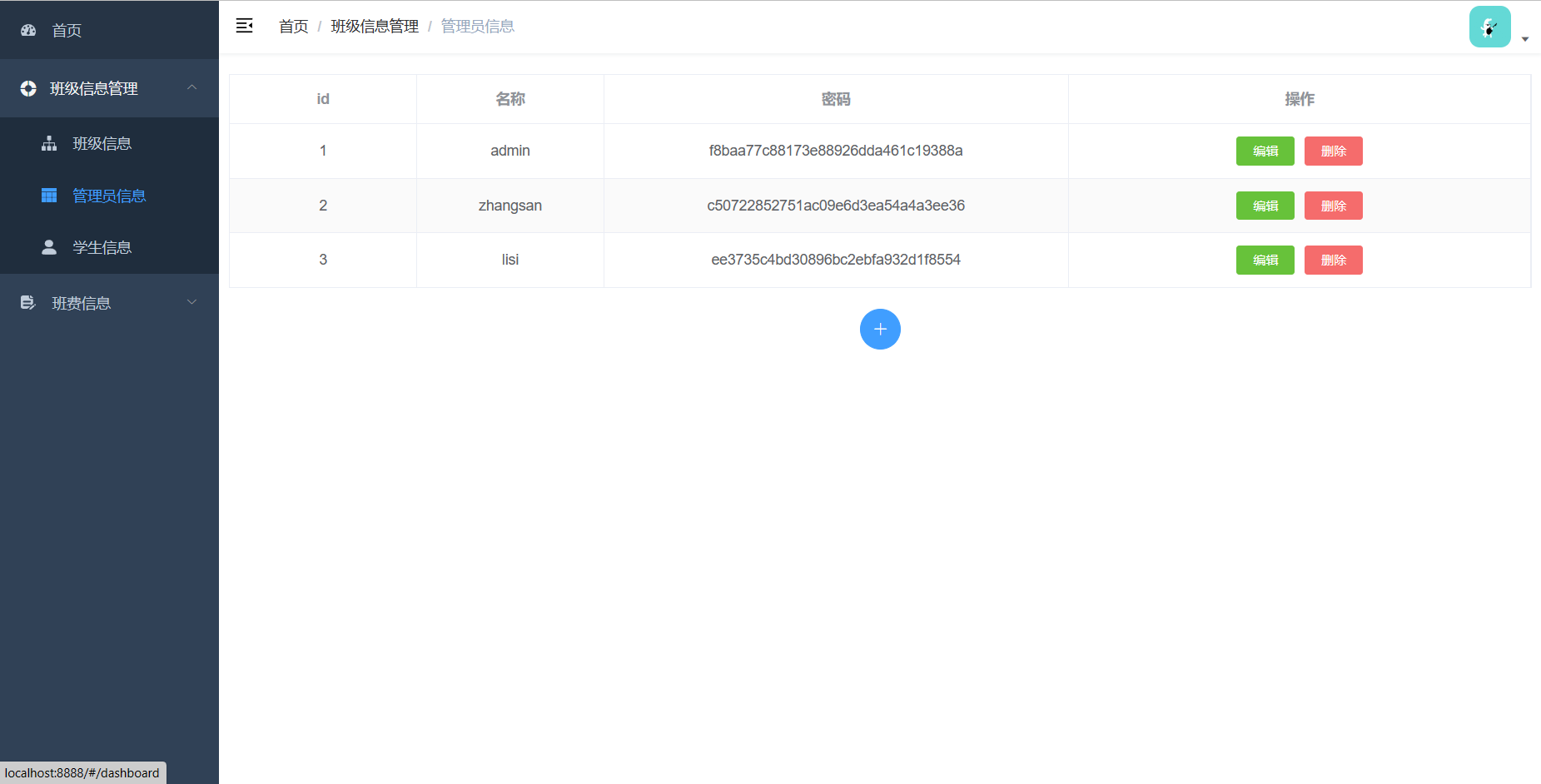
查询页面



删除页面

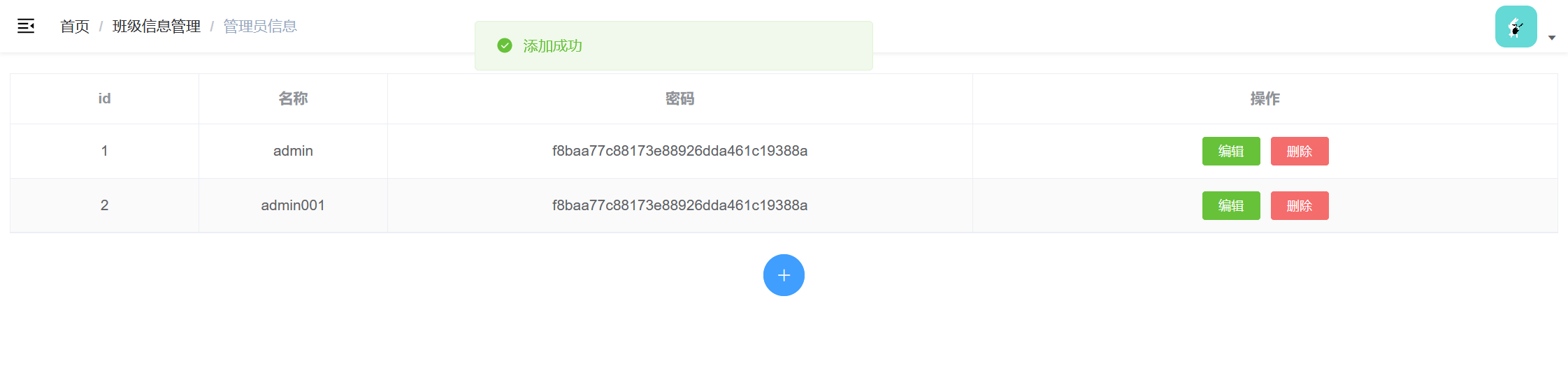


### 管理员信息管理

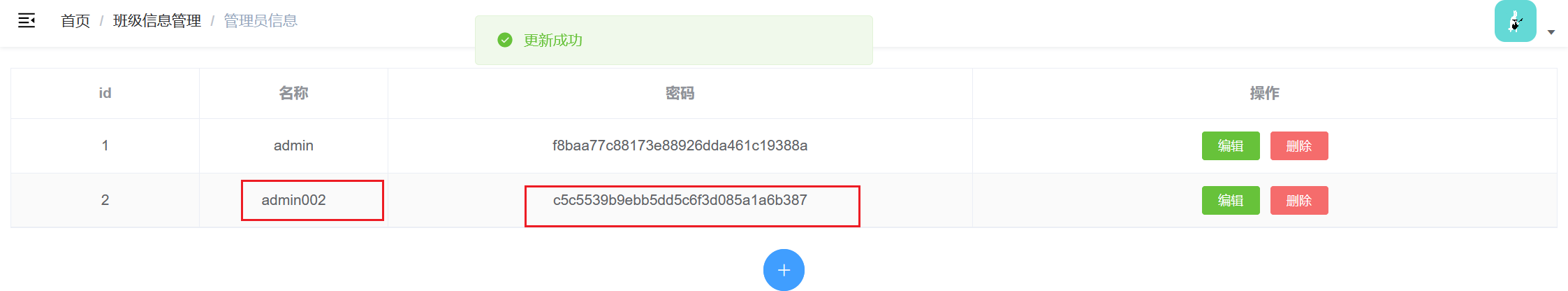


此页面可以实现对管理员信息的增加、修改、删除。由于需要对管理员用户的信息修改删除的效果进行展示，因此将密码做出展示，但密码在后端经过盐值和MD5的处理。

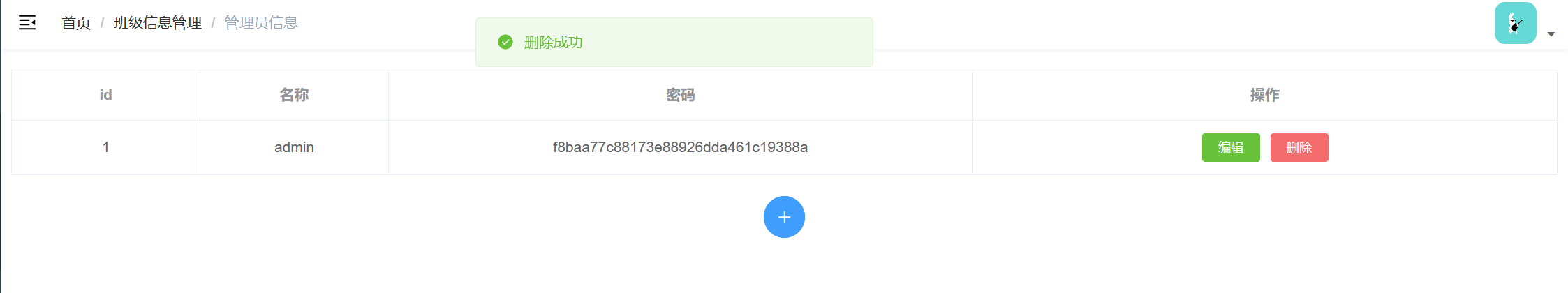
增加页面



修改页面



删除页面



### 学生信息管理



此页面可以实现对学生信息的导入。

### 班费查询



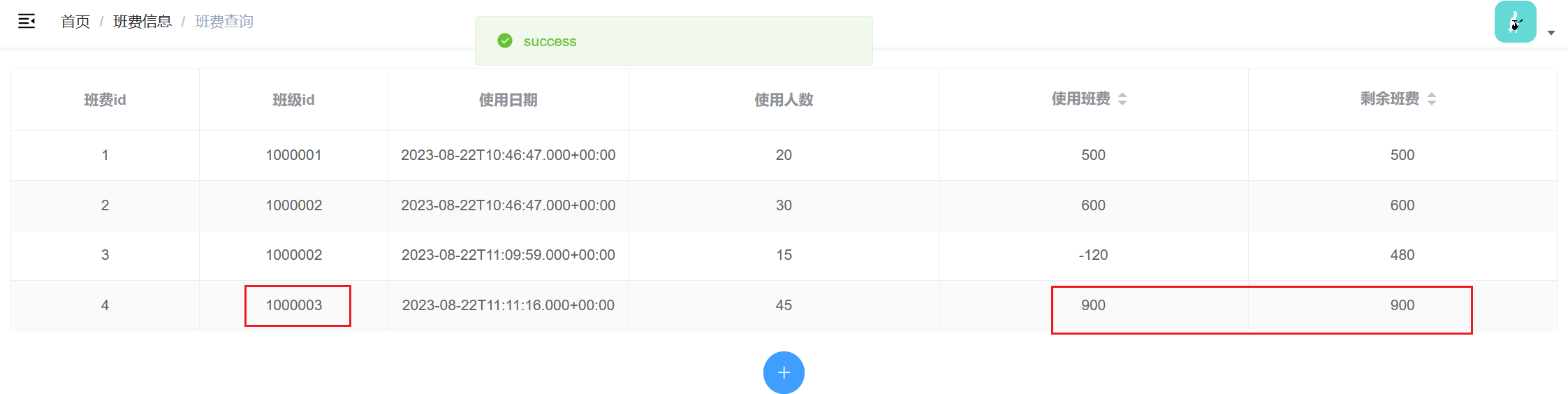
此页面实现了对各班班费信息的查询，可以添加班费信息。使用班费符号为＋代表班费的收入，-代表班费的支出

添加班费

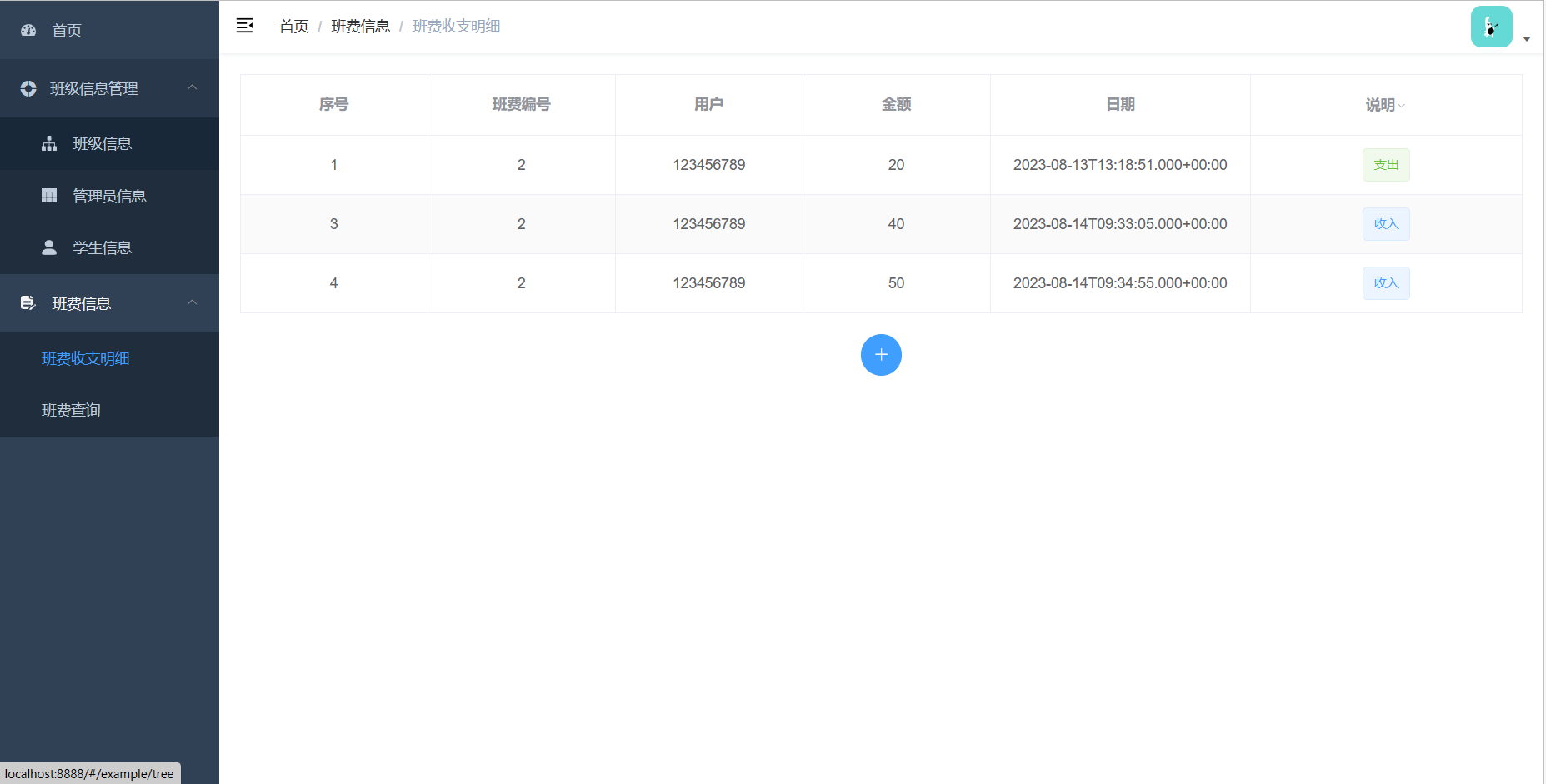


在原有的记录上进行修改剩余班费的修改

若没有改班级的班费记录，则新建一条班费记录，剩余班费则为使用金额



### 班费信息明细



此页面记录了班费信息的明细，可以对班费信息进行添加。

添加详细的班费信息



# 设计总结

## 小组详细分工

数据库设计：衷杰、张乐（主导：衷杰）

后端设计：衷杰

前端设计：张乐

## 课程设计的个人总结

### 在数据库的构建中

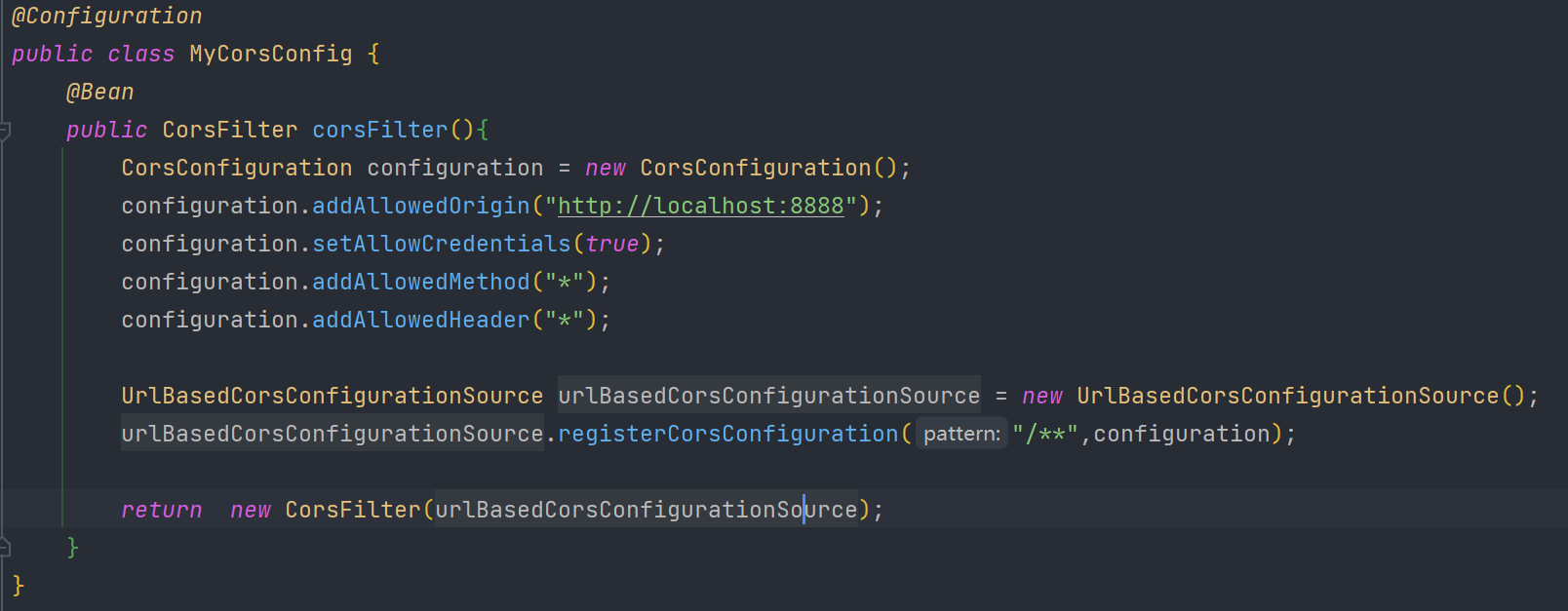
在数据库设计方面，从需求分析到概念结构设计，从概念结构设计到逻辑结构设计再到物理结构设计，环环相扣，体验到了完整的数据库设计基本流程，也让我对相关的数据库设计知识更加深刻的了解。但是很多事情不是想象中的那么简单的，它涉及到的各种实体、属性、数据流程、数据处理等等。

数据库与后端交互的同时，需要注意表设计的字段名避免与Java的特殊词相冲突。我们在设计class班级表中，由于`class`属于Java的关键词，因此在我们疏忽的情况下给我们带来了许多问题。最后通过改变后端的实体类的字段名`class`为`grade`来解决问题

## 在后端的构建中

1、由于要使用Redis来存入用户登录的Token，这样利于用户的退出注销。但我仅、会对Redis的增加和查看，因此相关的Redis配置遇到了较大的麻烦，通过阅读相关的博客，设置了Redis的配置类。

2、在与前端交互的过程中，首先最需要解决的问题即——跨域。我们搜索得到解决跨域问题有许多种解决方法，即在前端解决或者后端解决。我选择了从后端选择配置类解决前后端跨域问题。



如图所示，这段代码是一个用于处理跨域请求的过滤器。

该代码段创建了一个CorsFilter对象，并返回它作为跨域过滤器。具体解释如下：

创建CorsConfiguration对象：

CorsConfiguration是CORS配置的表示，用于设置允许的来源、方法和标头列表等。使用addAllowedOrigin方法添加允许访问的源。在这里，只允许来自"http://localhost:8888 "的请求访问资源。使用setAllowCredentials方法设置是否允许发送身份验证信息（如cookies）。使用addAllowedMethod方法添加允许的HTTP请求方法。使用通配符"\*"表示允许所有方法。使用addAllowedHeader方法添加允许的HTTP请求头。同样使用通配符"\*"表示允许所有头信息。

创建UrlBasedCorsConfigurationSource对象：

UrlBasedCorsConfigurationSource是基于URL路径的CORS配置源，用于根据URL路径匹配CORS配置。使用registerCorsConfiguration方法注册CorsConfiguration对象。在这里，使用"/\*\*"作为URL路径模式，表示匹配所有路径。

返回CorsFilter对象：

CorsFilter是Spring框架提供的一个过滤器，用于处理CORS请求。CorsFilter接受UrlBasedCorsConfigurationSource对象作为参数，以配置CORS过滤器的行为。

当应用程序收到请求时，该过滤器会拦截请求并检查请求的来源、方法和头信息是否符合配置的CORS规则。如果符合规则，服务器会在响应中添加相应的HTTP头信息，允许客户端访问资源。否则，服务器会拒绝请求并返回适当的错误响应。

3、在使用MybatisPlus的时候，虽然可以省略简短的相关sql语句的操作，但是遇到有特殊需求的情况下，还是需要我们自己在Mapper.xml文件中写入相关sql操作语句。

## 在前端的构建中

1、对于前端的 JavaScript、CSS、脚手架等知识学习不够深入,项目经验不足参与过的完整系统开发少,因此经常会出现一些低级错误,影响开发效率;

2、对于系统应用业务场景需求不熟悉,所以在开发的过程中多次出现因为需求的原因,而不断修改、返工的情况；

3、沟通能力不足，与后端人员对于项目的功能实现没有进行清楚的沟通，从而影响了项目的进程。

4、代码质量有待改进。前端的代码需要具备可读性、可维护性和可扩展性。因为是基于提供的框架编程，导致代码可读性较差，过度冗余。