.NET期末项目文档

**项目选题：虚拟仿真实验平台教师端系统**

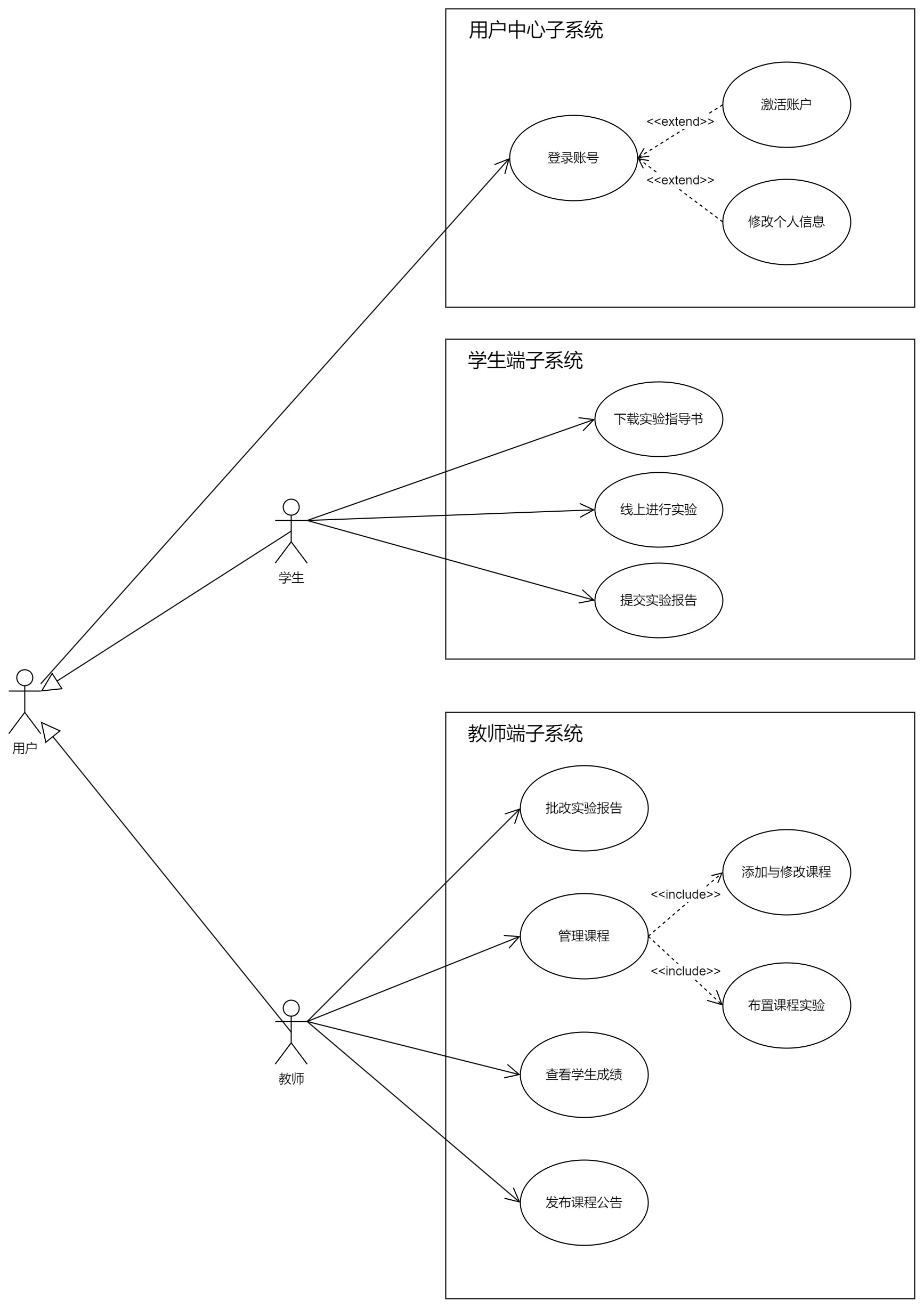
**小组成员：2050865 黄彦铭 2051196 刘一飞**

# 项目简介

在我院“软件工程管理与经济”一课中，有许多实验供学生学习。但是，目前该课还处于学生手写实验报告，教师纸质批改实验报告的阶段。为了能够让该课的实验流程更加便捷，黄杰老师作为甲方，要求我们小组开发一个线上虚拟仿真实验平台。

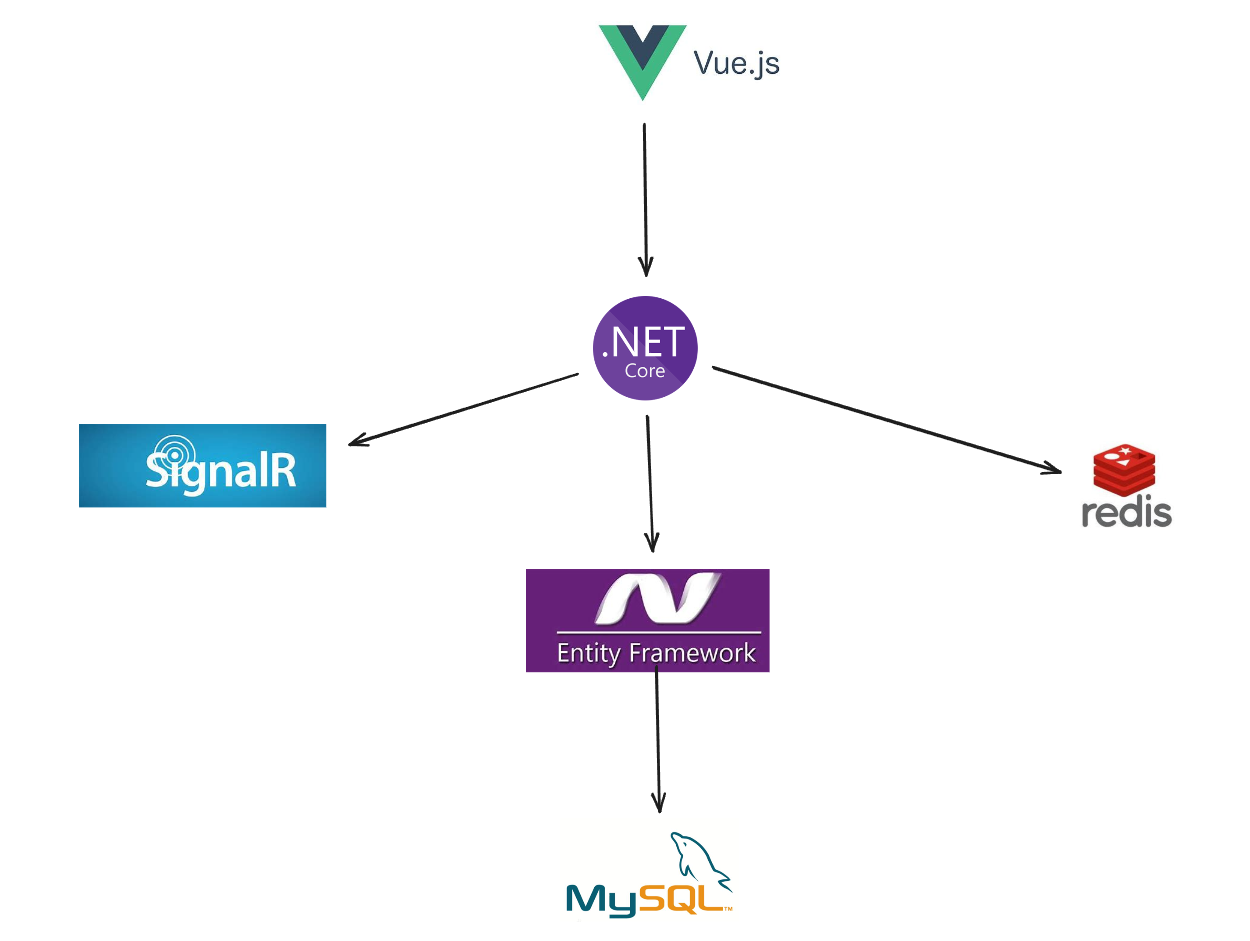
虚拟仿真实验平台分为学生端与教师端。学生可以在学生端系统中提交实验报告，教师可以在教师端系统中批改实验报告，以及能够更方便地管理课程信息。

系统的用例图如下：



在本次的课程项目中，我们的.NET部分主要涉及到了用户中心子系统与教师端子系统。

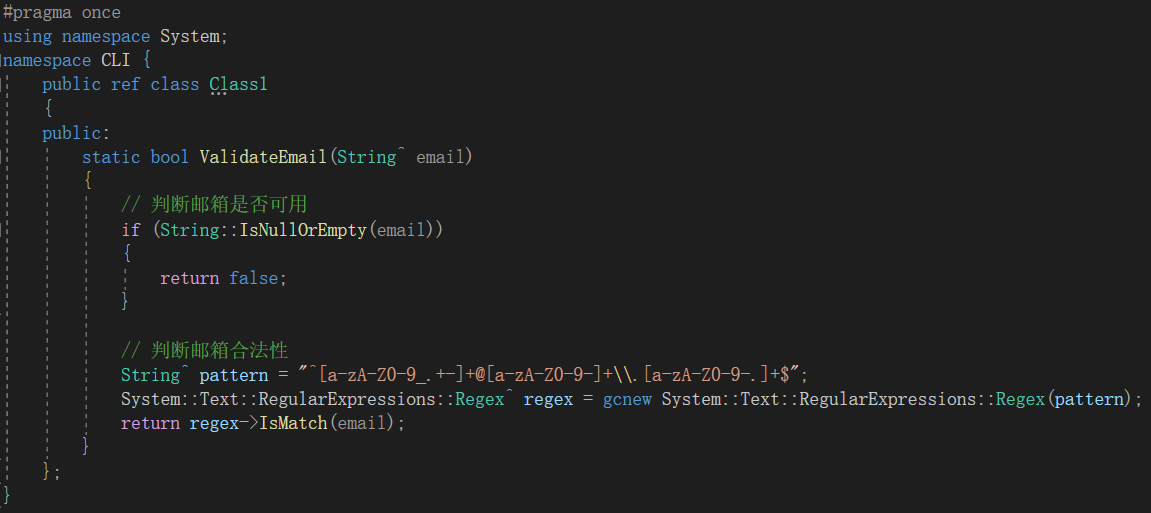
# 技术架构



# 项目基本要求

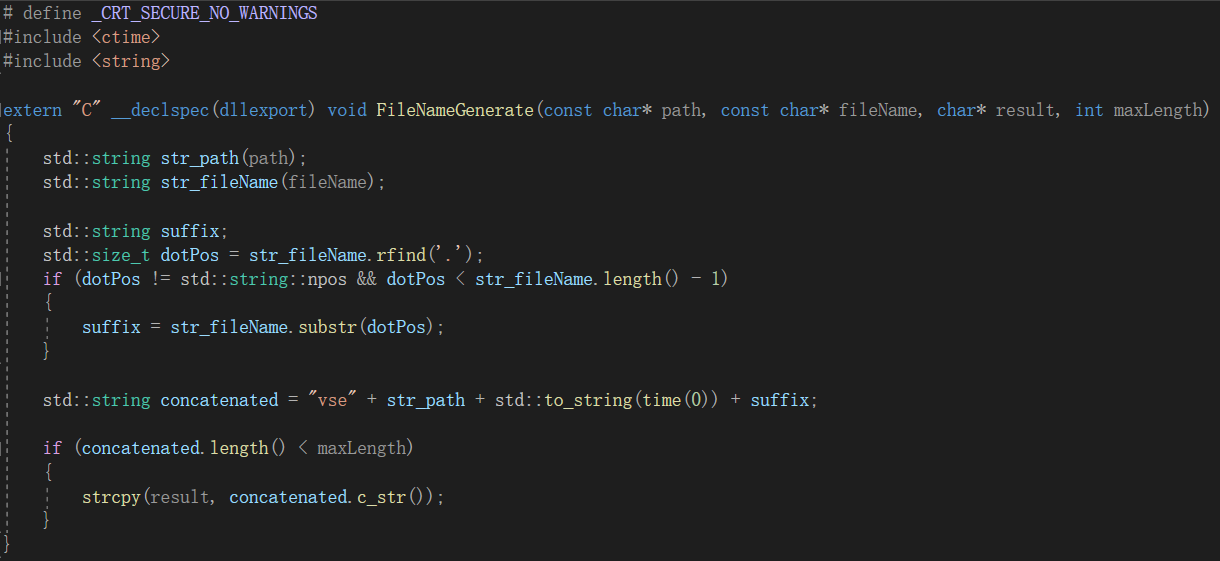
## 3.1 C++/CLI

使用C++/CLI实现了邮箱合法性校验。



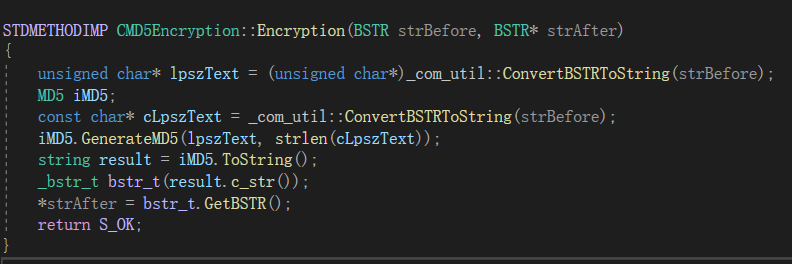
## 3.2 Win32 DLL

使用Win32 DLL实现了上传文件时，将用户提供的路径转化为服务器上文件系统的路径。



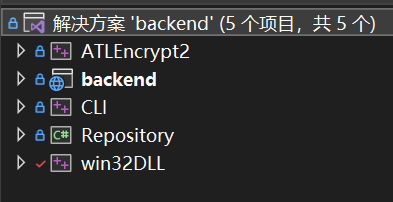
## 3.3 COM组件

使用COM组件实现了MD5加密算法，输入输出均为一个字符串。



## 3.4 程序集

我们的项目总共分为5个程序集，如下图所示：



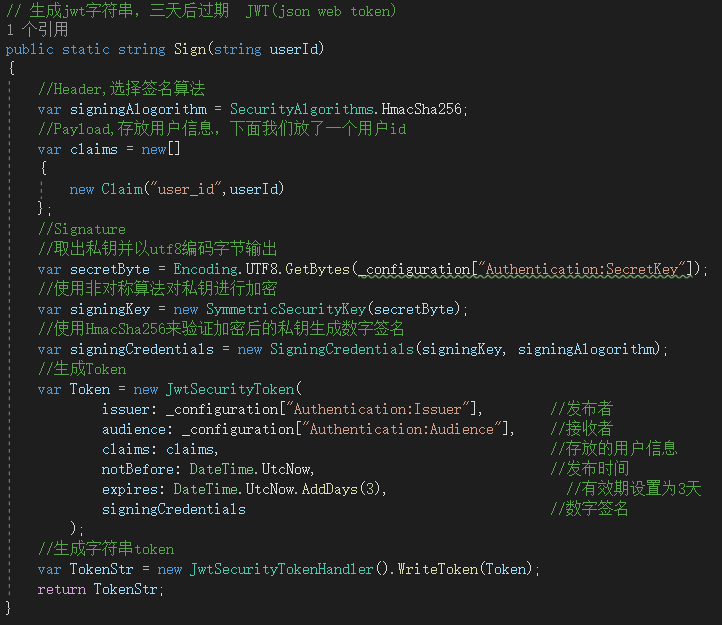
* backend：负责提供后端提供WebAPI的所有逻辑，包括所有的后端接口以及拦截器。
* Repository：负责服务器端程序与数据库交互，主要是利用Entity Framework将实体类与数据库表做关系对象映射。
* ATLEncrypt2：该COM组件负责加密算法。该程序集实现了MD5加密算法，在后端存入数据库的密码都要经过MD5加密。
* CLI：负责验证邮箱合法性。用户在登录界面中，如果登录的邮箱格式错误，后端会返回错误信息并显示在前端。
* win32DLL：负责将用户提供的文件路径转换为服务器上的文件路径。

# 技术亮点

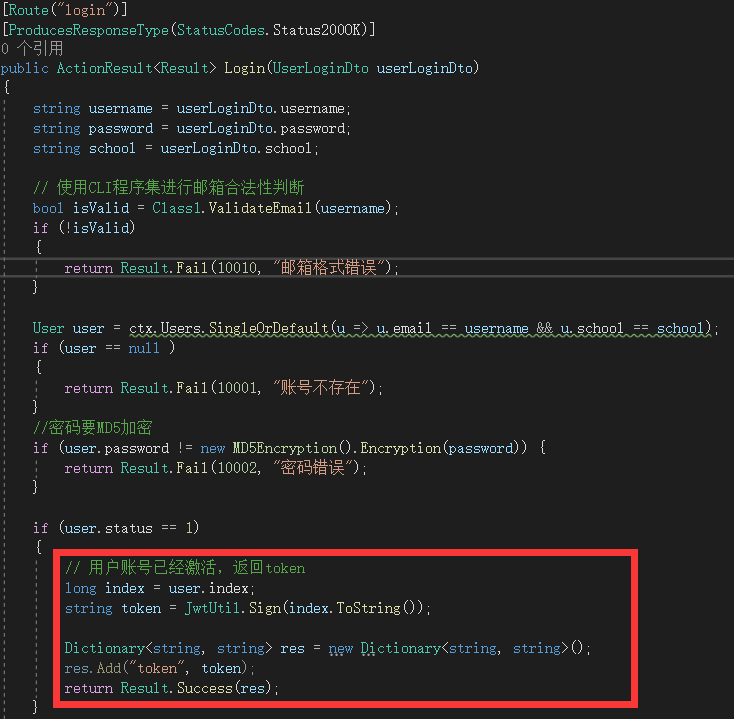
## 4.1 JWT

JWT（JSON Web Token）是一种用于在网络应用中进行身份验证和信息传递的开放标准。它是一种轻量级的安全令牌，以JSON格式表示，并使用数字签名或加密进行验证和保护信息的完整性。在本项目中，使用JWT作为身份验证和授权的机制，提供安全的身份验证方式。

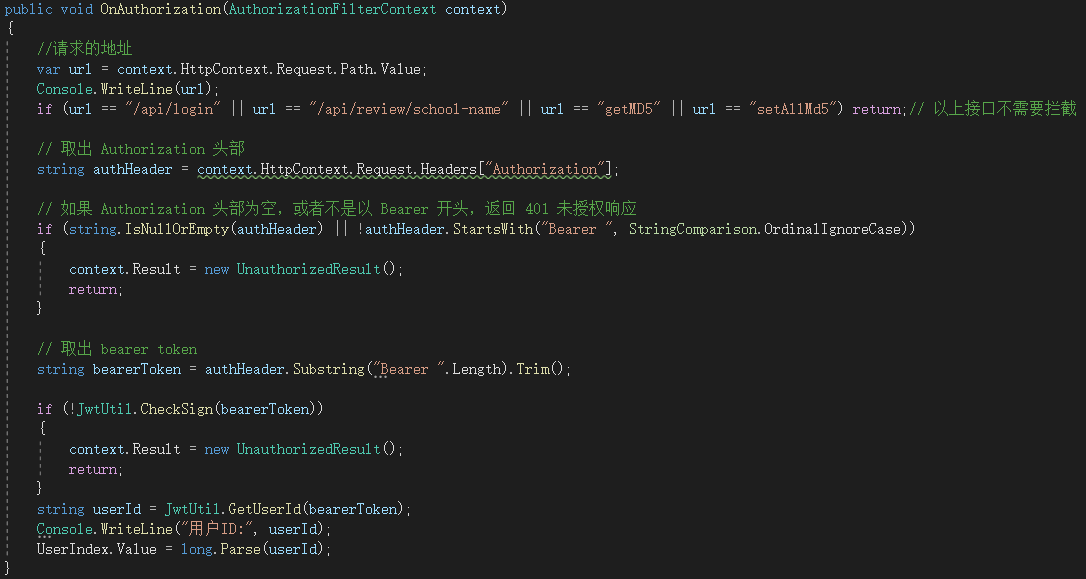
在项目中的 JwtUtil类中，定义了生成 JWT的算法：



在用户登录接口中，如果用户账号已经激活并且成功登录，则调用 Sign 方法生成 JWT 令牌，并将 JWT 令牌传递给客户端，存储在本地储存中。



在服务器端的过滤器中，同样定义了如下代码：

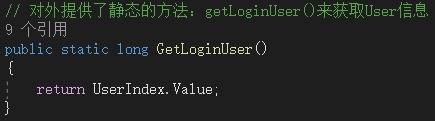


即除了用户在登录阶段需要使用的接口不需要拦截之外，其他接口会被拦截，并且判断请求头中是否携带了合法的 JWT 令牌，如果没有携带或 JWT 不合法，则接口请求失败。

与此同时，在 JWT 的载荷中还携带了用户的ID信息，因此在拦截器验证 JWT 合法性从而验证用户身份的同时，还能够从 Payload 中提取用户的ID信息，用于后续逻辑的实现。

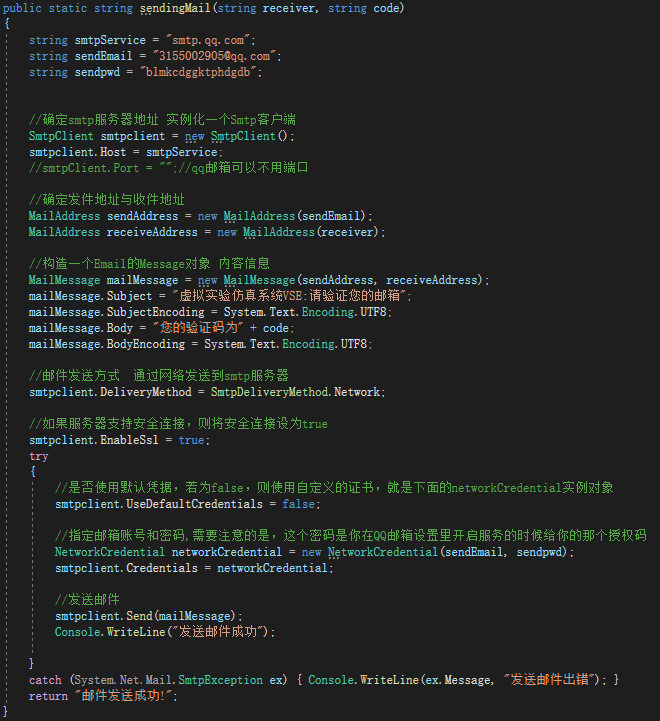
descript

为了能够使拦截器中获取到的 userId 能够在Controller 中可用，拦截器中定义了一个线程域单独用于存放登录用户的Id，并且对外提供静态方法，来获取用户信息。



## 4.2 邮件发送

项目中使用了 .NET 框架中的 SMTP（Simple Mail Transfer Protocol）邮件发送技术，来实现用户在激活账户时发送验证码邮件的功能。

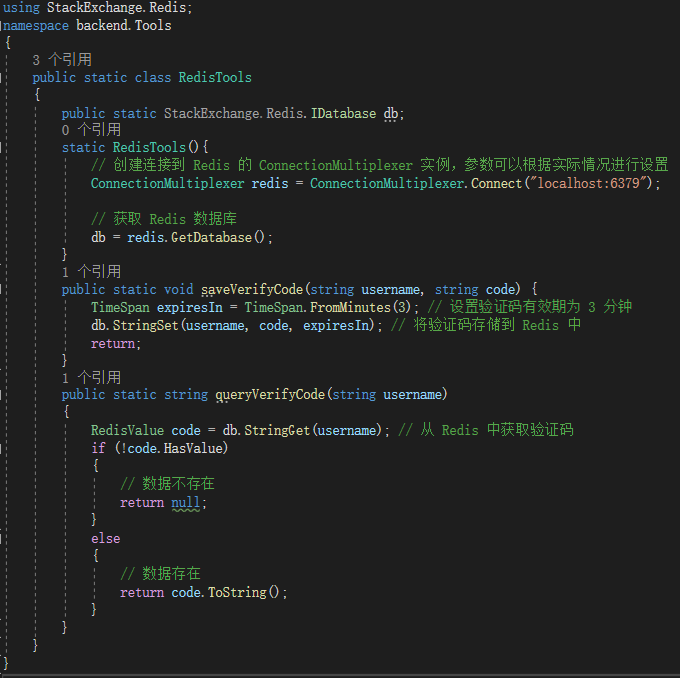


主要使用了 [System.Net.Mail](http://System.Net.Mail) 命名空间中的相关类，包括：

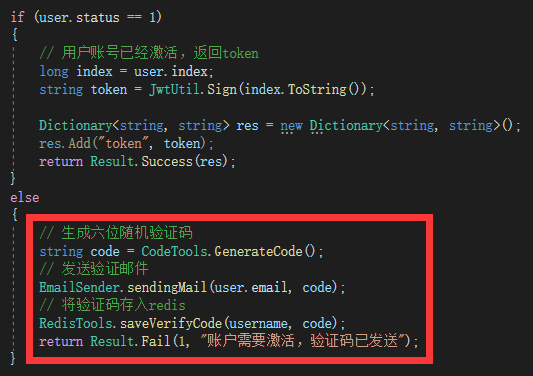
1. SmtpClient 类：用于与 SMTP 服务器进行通信，并发送电子邮件。
2. MailMessage 类：用于构建电子邮件的内容，包括主题、正文和发件人/收件人等信息。
3. MailAddress 类：用于表示邮件地址，可设置发件人和收件人的电子邮箱地址。
4. NetworkCredential 类：用于指定发件人邮箱的账号和密码（或授权码），以进行身份验证。

## 4.3 Redis实现验证码功能

项目中，使用 StackExchange.Redis 这个 .NET Redis 客户端库来连接和操作 Redis 数据库，以实现的验证码的存储功能。



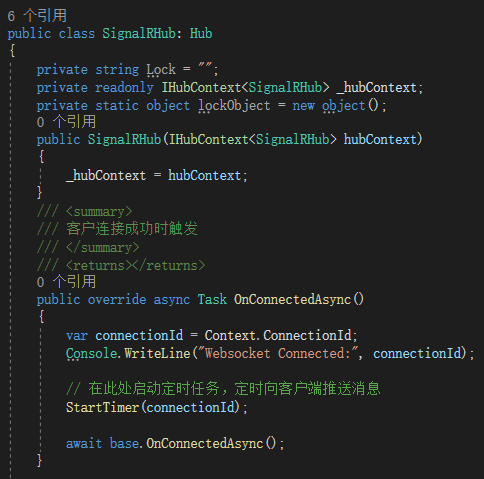
上面的 RedisTools 类中，实现了 saveVerifyCode 和 queryVerifyCode 两个方法，用于实现验证码在 redis 中的存储和查询。



在用户登录接口中，如果用户的账户没有被激活，则会生成六位随机验证码存入 Redis 中，用于此后用户激活账号时验证身份使用。

## 4.4 SignalR

SignalR 是一个开源的.NET库，用于在 Web 应用程序中实现实时通信功能，包括服务器到客户端的推送通知和客户端到服务器的双向通信。在项目中，使用 SignalR 实现了系统消息的时时通知。



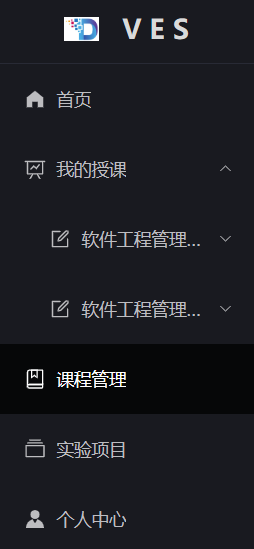
继承自 Hub 类的 SignalRHub 类定义了服务器端与客户端之间的通信方法。在 SignalRHub 类中，每当有客户端连接，都会调用 OnConnectedAsync 启动定时任务，定时向客户端推送消息。

在客户端中，创建了 SignalR 的连接，并且使用 connection.on 注册事件处理程序，当服务器端发送命名为 "ReceiveMessage" 的消息时，该事件处理程序将会被触发，重新去服务器端获取当前用户的所有消息，更新消息列表。

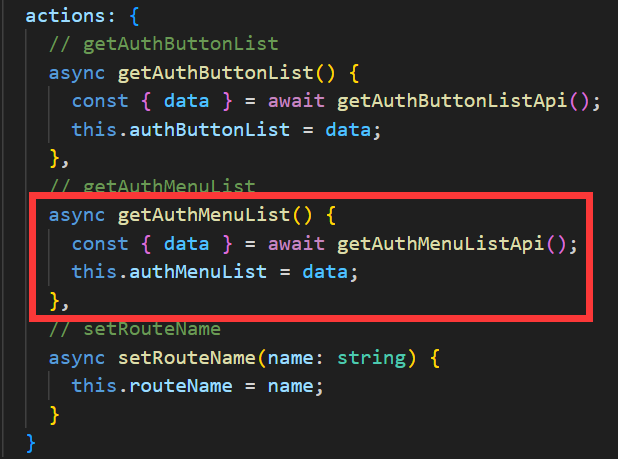


## 4.5 动态路由

教师可以增减课程，课程的前端路由必须随之改变，这就要求前端显示的侧边栏路由，需要后端动态返回：



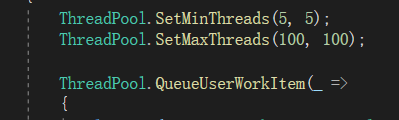
后端需要动态地查询该名教师目前教了哪些课，生成的路由以json形式返回给前端。前端每次加载页面的时候都需要调用该接口，以生成侧边栏的菜单。



## 4.6 线程池

在系统中，学生提交实验报告这一行为是流量不平稳的。在大部分时间，学生提交实验报告的接口访问并不频繁。但如果到了大家共同的ddl时间前，例如临近晚上12点时，提交实验报告的接口访问量会骤增。

为了保证在高并发时能够有足够多的线程数处理接口调用，又要保证在接口访问低谷期不造成资源浪费，我们选择采用线程池来统一管理并发资源：

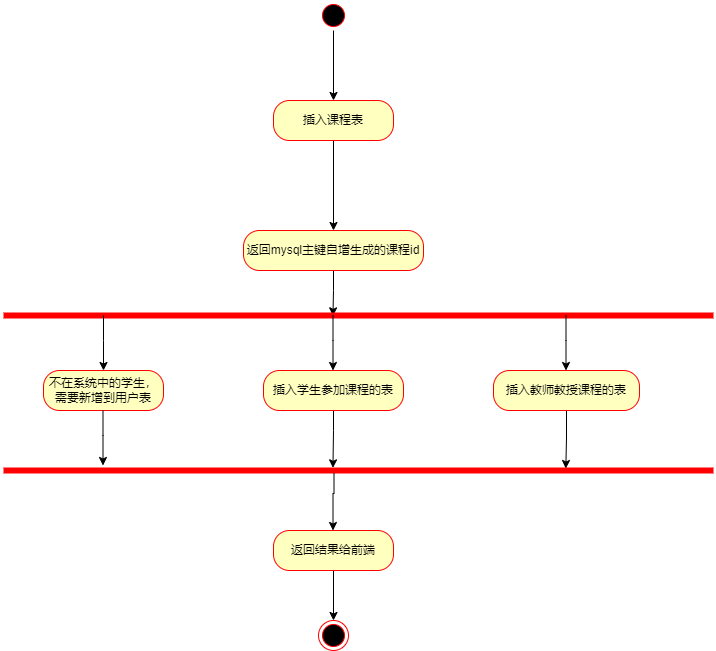


我们将最小空闲线程数设为5，最大线程数设为100。这样在并发访问频率小的时间段，只需要很少的几个线程就可以处理并发；当在提交作业的高峰期时，线程数能够增长到较大的数目，处理高并发的问题。

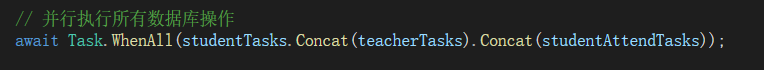
## 4.7 异步编程

后端中对于一些比较复杂的接口，使用了基于任务的异步编程。

例如，在添加课程的接口，需要涉及到4个表的插入，具体如下图所示：



其中，有三类插入操作可以并行，故采用异步编程的思想。让三类插入操作并行，等待这三类操作全部结束后再返回给前端。



## 4.8 事务

如果在一个接口内涉及到多个对数据库进行更改的操作，则有可能出现某个操作失败而产生脏写。为了避免这种情况，我们需要将整个接口的操作当成一个事务，如果事务中有任何一个环节出错，都必须回滚这个事务。

在代码中，使用TransactionScope实现事务。只有当事务运行到了ts.Complete()，才会提交事务，否则回滚事务。

