#### 项目简介

基于平时的学习体验，本组抓住大学生线上学习资料查找的痛点，设计实现了线上学习平台。SOS平台是面向大学生，尤其是考研学生群体的线上学习平台。用户可以查看大学的课程、书籍及其相关题目，并可查看题目的详细解析并可以通过文本检索相关信息，并对系统的问题进行反馈。管理员对反馈进行处理，并可以对系统数据进行编辑。专家对专业问题进行解答。

#### 系统设计与实现

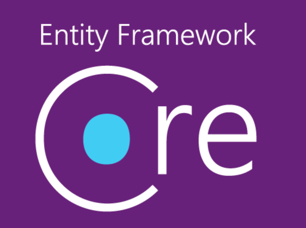
#### 1.技术选型

 在项目伊始，整个项目各个环节的技术选型便是最先需要确定的问题。首先在B/S与C/S架构的选择中，为了使用户能够方便的使用到本平台提供的相关服务，我们最终决定使用B/S架构，其次在整体开发架构的选择上，我们为了降低整个项目的耦合性，选择了在业内已经很成熟的前后端分离的架构模式：对于后端来说，由于预先规定了以C#作为开发语言，/\*且为了能够适配Oracle提供的Develop Tools for VS2019，我们选择了VS2019能够支持的.NET框架的最高版本，且\*/为了后续能够支持多平台的移植工作，我们最终选择了ASP.NET Core 3.1.27。对于前端而言，考虑到社区活跃程度以及目前业内的流行趋势，我们并没有采用与ASP.NET同在MicroSoft生态下的RazorPage以及Blazor的相关组件，而是采用了Vue作为前端的javascript框架。

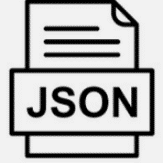
#### 2.数据库部署

 为了能够使全组同学能够聚焦在项目的开发上而不被同步数据库等一系列繁琐的问题所困扰，我们决定舍弃本地oracle而是将数据库部署在云端服务器上，为了方便后续的备份以及移植工作，我们通过Docker镜像的方式，/\*镜像源为truevoly/oracle-12c\*/，部署在服务器的1521端口之上，并开通公共账户admin，即可让所有同学通过Navicat Premium16连接并同步数据库。

#### 3.数据库连接

 在后端开发的起始阶段，我们决定采用Oracle提供的Develop Tools for VS2019作为数据库管理组件辅助开发，但后续很快遇到了问题，首先是裸SQL语句对于条件查询的种种限制，其次是没有数据持久化层作为缓冲而频繁访问数据库可能触及轻量级应用服务器的性能瓶颈问题，于是我们决定使用Entity Framwork Core作为ORM框架对数据库进行持久化。

#### 6.前后端连接

 在分别确定前后端的架构之后，我们着手开发了一个Demo项目，但在分别完成了后端接口建设以及前端页面搭建之后，我们在连接的过程中遇到了跨域问题，在开发过程中，前端通过Vue的proxy代理的功能实现了跨域资源共享，从而方便进行接口测试；在部署过程中，我们通过Cors向后端注入跨域服务、声明允许任意请求头，并在后端配置中开启相应中间件实现跨域资源共享。同时为了方便前后端沟通，我们使用Swagger UI生成可测试接口的API文档，并且暴露在同属后端的5000端口上供前端同学查阅及使用。另外，考虑到Swagger只能测试传参接口的局限性，我们还采用了Postman作为表单相关接口的测试工具。在前后端的数据交换上，后端的所有返回都遵循了业内后端数据的返回规范，以JSON作为数据交换的格式，并且建立了一套错误代码code与错误信息msg的对应手册，方便前端同学及系统管理员快速定位错误并解决问题。

#### 7. 云端部署

 在完成后端API的开发之后，为提高前端开发的效率，我们决定将后端项目打包发布到服务器上方便前端同学进行API查询及调用。为了简化相关环境的配置，我们同样是通过Docker镜像的方式，以微软官方的asp.net:3.1作为基础镜像，通过Dockerfile进行配置，包括建立工作目录，暴露访问端口，并且确定容器的EntryPoint，同时为了简化整个部署流程，我们通过shell脚本同步开发了一个轻量级的自动化部署套件，方便后端同学在进行修改后一键更新。

前端项目上云也采用了同样的解决方案，截止目后端项目在服务器上已经迭代了52个版本，前端项目迭代了28个版本，所有的版本已经进行了备份，方便项目维护以及必要时进行版本回退。

在所有开发及部署工作完成之后，为了能让更多的人使用并测试我们的项目，最后的工作就是将整个项目上线，即将ip解析并转发到我们之前认证的域名www.venushui.top之上，并且向工信部及公安部门申请ICP备案，目前备案已通过管局审核，编号为沪ICP备2022026340号。

#### 系统演示

数据库

#### 4.后端架构

由于项目采用的是前后端分离的架构模式，所以后端整体上一定是ASP.NET Web API的模式，但由于引入了Entity Framework Core，后端可以通过Scaffold组件为每一张实体表和关系表与后端代码建立连接，这样势必难以进行代码的复用，从而降低系统的可维护性。所以我们决定在Entities和Controllers之间抽象出来一层Models，不仅能够实现每个Model下方法的复用性和可维护性，而且能向Controller隐藏数据库的连接细节，提高后端系统的安全性。

#### 5.前端架构

前端采用vue-cli脚手架搭建Vue2项目，部分组件使用Element UI。使用vue-router实现了前端路由，实现了路由导航与嵌套路由，并借助vue-router实现了页面之间参数的传递，使局部刷新更高效，引入了store储存前端数据，使用axios与后端接口建立通信。为了增强代码的复用性，我们引入了许多自定义组件：例如书籍组件和课程组件，在使页面更加美观的同时，也让整个项目秩序井然。