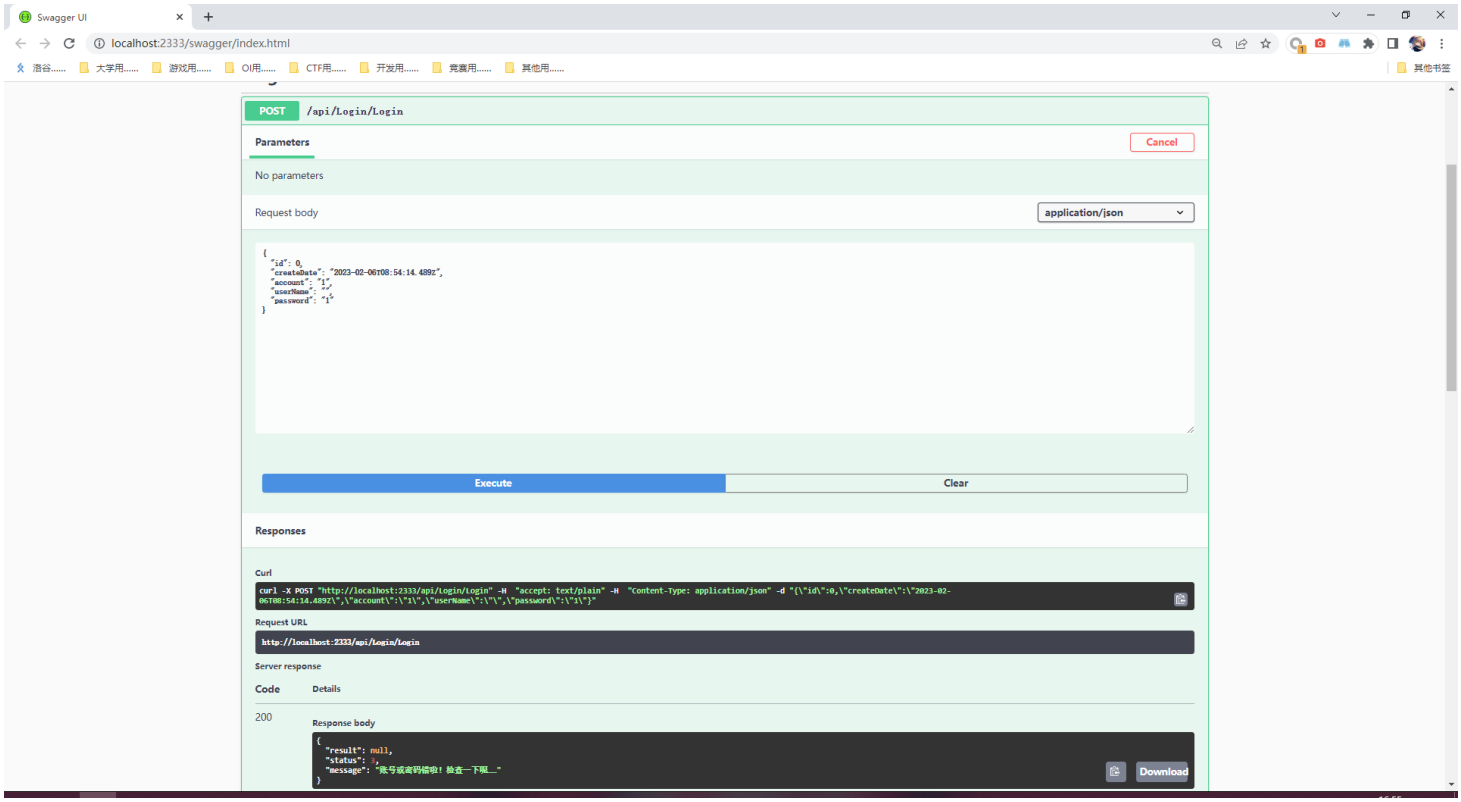
技术方案

初步框架分享：

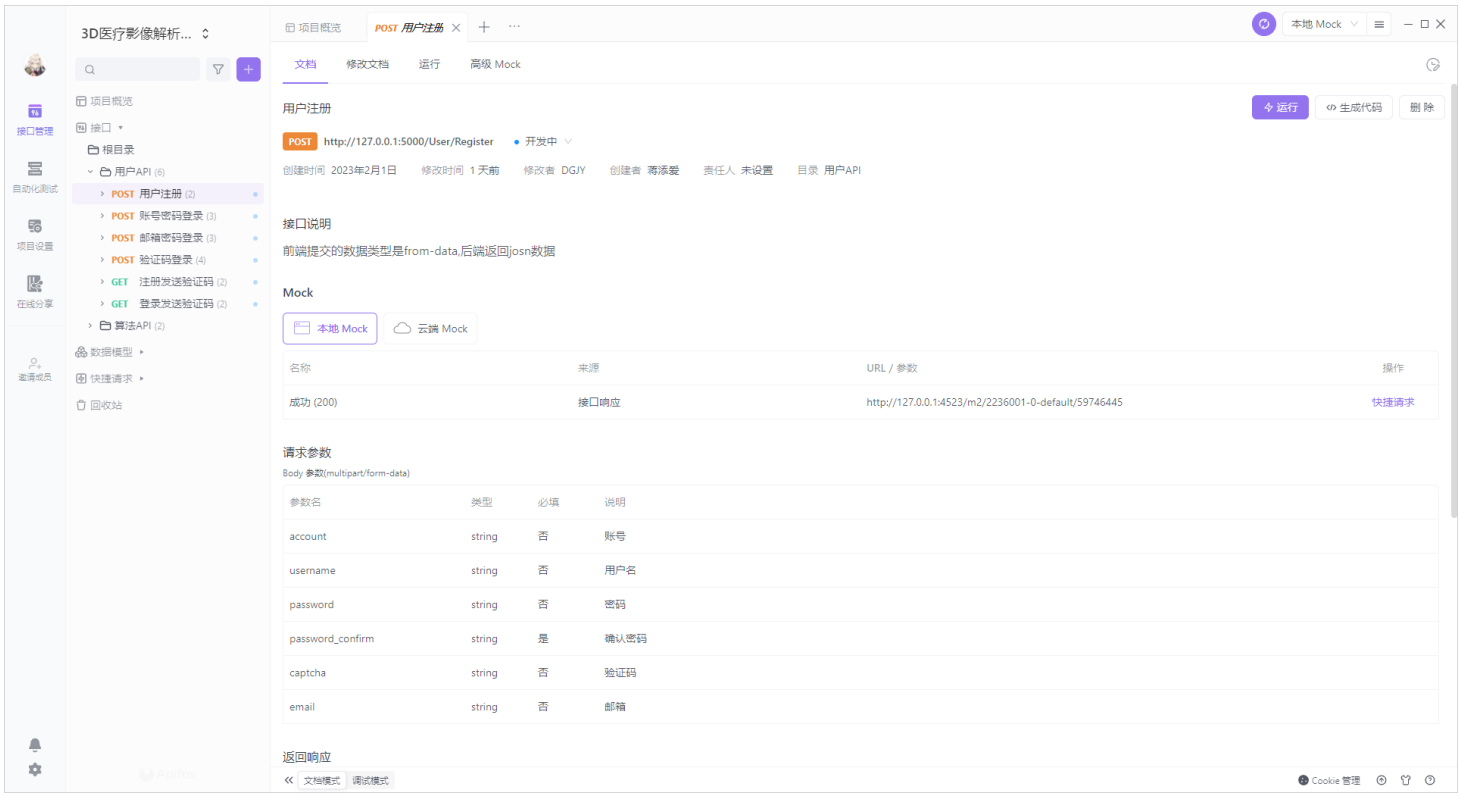
- 前端 - 面向用户，提供交互和界面
  - Vue
  - React
- 后端 - 面向服务，提供业务处理和数据持久化
  - Java: Sprintboat
  - Python: Flask
  - C#: ASP.NET

前后端分离、独立地开发，通信通过“接口(API, 为HTTP协议)”来约定和实现。

API接口示例：



API接口文档工具 - Apifox：



1. 用户系统

很多成熟的解决方案和框架。

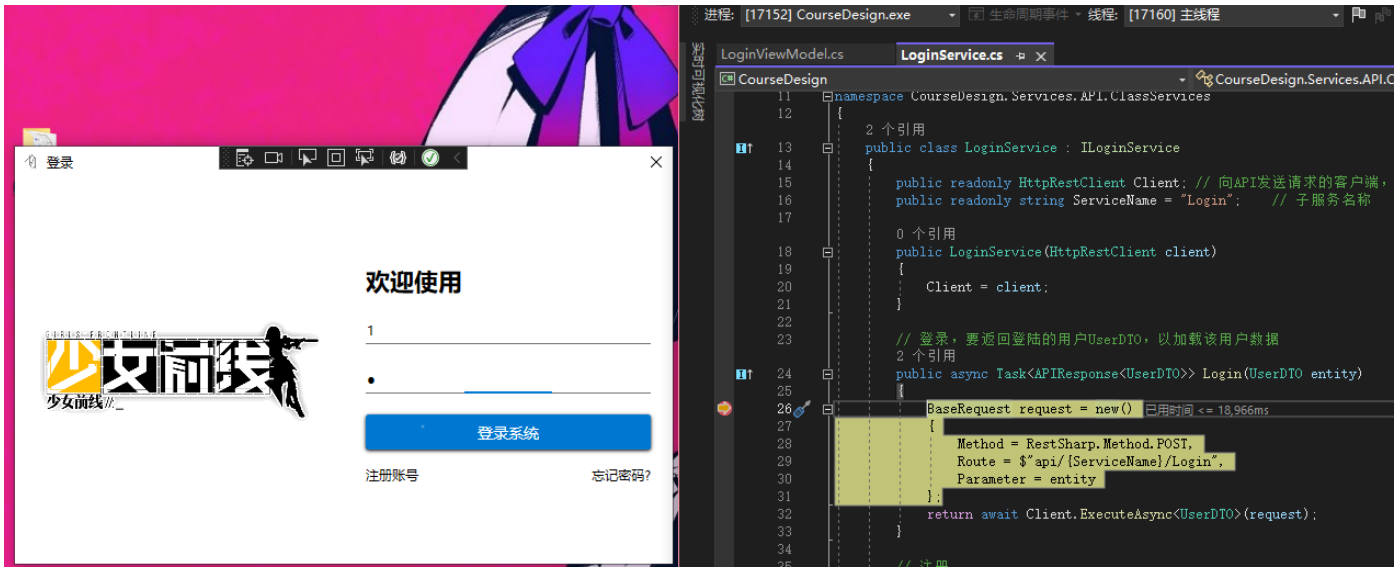
- 前端 - 提供交互界面，调用后端提供的接口
- 后端 - 提供用户系统的相关接口，处理前端发来的请求并给与回应

功能：

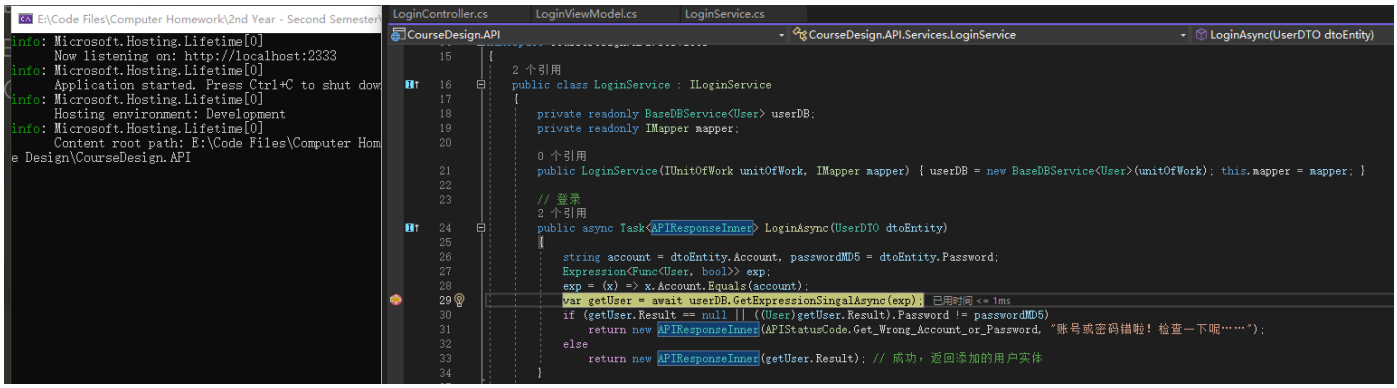
- 注册 - 前端通过接口发送表单信息，后端检查合法性后，存储到后端数据库实体。
- 登陆 - 前端发送信息，后端进行密码验证鉴权后，前端存储后端返回的Token。

演示 - 登录功能：

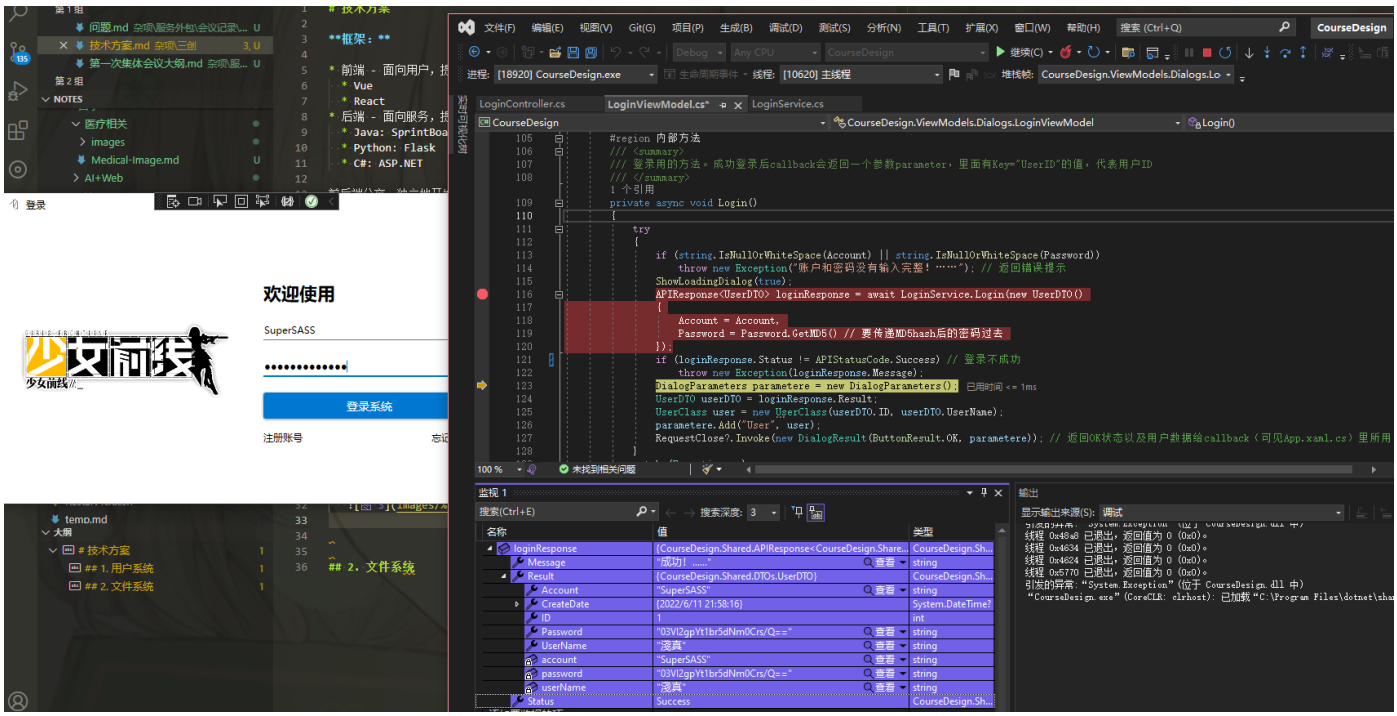
1. 前端调用API接口，发送HTTP请求(Request)。



2. 后端接收到API请求，进行相应前逻辑处理，然后向数据库发送查询请求，最后根据查询结果进行后逻辑处理再返回响应(Response)。



3. 前端根据API响应里返回的的字段，执行对应的逻辑操作（更新用户上下文，完成登录）。



2. 文件系统

目前没做过类似的，仅提出实现想法。

有关文件存储方面：

- 云端（SSO，如又拍云） - 省心、不用管理，需要学习API
- 服务器端 - 需要自己写一个简单的文件管理系统，但全归自己掌握

上传用例（以SSO为例）：

- 前端通过SSO提供的API，HTTP的PUT请求，Body为form-data，上传文件到SSO。
  包含字段：
  - 用户ID
  - 文件ID
  - 文件
  - 鉴权用字段(如Token)
- SSO自行根据API调用，保存上传的文件。
- 前端通过后端提供的API，记录用户与文件的关联。
  包含字段：
  - 用户ID
  - 文件ID
  - 鉴权用字段(如Token)
- 后端鉴权后，将“用户ID”和“文件ID”两个外键，记录到“用户文件”实体表中，建立关联。

参考资料：

- Java后端如何实现大文件存储下载?
- 又拍云云存储SSO
- Vue实战之element文件上传
- springboot上传图片的两种方式详解（本地/OSS对象存储）