实训总结

学号	姓名	导师	研究方向	报告周期
17343105	田皓	余阳	GoOnline 在线编程项目开发	总结

实训内容

概述

完成 GoOnline classroom 的签到功能后端的开发与测试。

实训小团队由我和两位负责前端的同学组成。

API 设计

老师获取签到结果

[GET /users/{username}/courses/{courseID}/classes/{classID}/checkResult]

- Request
  - Headers

```
Authorization: token
```

- Response 200 (application/json)

```
{
    "ok": true,
    "data": [
        {
            "name": "用户名",
            "real_name": "真实姓名",
            "student_id": "学号",
            "check_status": "0-未签到 1-正在签到"
        }
    ]
}
```

学生签到

[POST /users/{username}/checkIn]

- Request
  - Headers

    ```
    Authorization: token
    ```

  - Body

    ```
    {
        "code": "签到码"
    }
    ```

- Response 200 (application/json)

```
{
    "ok": true,
    "data": [
        {
            "ok": false,
            "data": ""
        }
    ]
}
```

老师发起/结束签到 (WebSocket)

[GET /users/{username/courses/{courseID}/classes/{int:classID}/checkIn]

- Request
  - Headers

```
Authorization: token
```

- Response 200 (application/json)

```
{
    "code": "签到码",
    "check_list": [
        {
            "name": "用户名",
            "real_name": "真实姓名",
            "student_id": "学号",
            "check_status": "0-未签到 1-正在签到"
        }
    ]
}
```

增加，修改数据库模型

在 MySQL 中，修改课时模型，增加签到状态字段。

```
# models.py-Class
# 课时签到状态 0-未签到/签到结束 1-正在签到
status = models.IntegerField(default=0)
```

在 MongoDB 中，增加集合 CheckCollection 存放每个课时中每个学生的签到情况。

```
# models.py-CheckCollection
class CheckCollection(mongoengine.Document):
    class_id = mongoengine.IntField(required=True, default=0)
    students_check = mongoengine.ListField(mongoengine.ListField(mongoe
ngine.StringField()))
```

每个课时刚创建的时候 status 为 0，老师发起签到之后改为 1，签到结束再改回 0，学生签到时会输入课程码-课时码-随机数组成的签到码，在 MySQL 中查找对应课时的状态是否正在签到，如果在签到中则更新 CheckCollection。

老师查看签到结果只需要查 MongoDB 的 CheckCollection 即可。

操作数据库

```
# 查询MongoDB 的CheckCollection 来获取学生签到列表
def getClassCheckList(courseID, classID):
    try:
        res = CheckCollection.objects.get(class_id=classID)
    except Exception as e:
        print(e)
        return [], False
    return res['students_check'], True

# 修改课时签到状态
def setClassStatus(courseID, classID, status):
    try:
        res = Class.objects.get(course_id=courseID, id=classID)
    except Exception as e:
        print(e)
        return False
    res.status = status
    res.save()
    return True
```

使用 WebSocket 进行签到

在 Django 中使用 Channels 来使用 WebSocket 功能。

连接开始，判断各种信息符合要求，新开一个线程每隔固定时间向前端发送签到码，收到断开信号之后关闭子线程并断开连接。

遇到的问题是连接断开后后端还一直在 print 签到码，所以得想办法结束子线程，最后还是用一个比较笨的方法解决：发送签到码的子线程用一个全局变量 flag 控制，disconnect 的时候 flag = 0。下面是发送签到码的函数：

```
# consumers.py- CheckConsumer
def send_code(self, courseID, classID):
    while True and flag == 1:
        utils.generateCode(courseID, classID)
        checkCode = utils.getCheckCode()
        print(checkCode)
        classCheckList, ok = utils.getClassCheckList(courseID, classID)
        res = []
        for i in classCheckList:
            resTmp = User.objects.get(username=i[0])
            res.append({
                "name": i[0],
                "real_name":resTmp.real_name,
                "student_id":resTmp.student_id,
                "check_status": i[1]
            })
            self.send(text_data=json.dumps({
```

```
            'code': checkCode,
            'check_list': res
        }))
        time.sleep(30)
```

学习收获

软件开发基本流程

软件开发的一般流程为：需求分析，系统设计，开发，测试，部署，在这个基础上，一个 Web 应用开发的流程又将**开发**这个步骤细分：API 设计->前端、后端、测试分别开发->完成。其中本次任务的需求分析由产品经理完成，系统设计，部署由架构师完成，我主要负责后端的开发和简单测试。

后端基础

后端要干的事情就是根据制定好的 API，接受前端的请求，给前端数据。本次开发使用的是 python 的 Django 框架，测试接口使用 Chrome 插件 Postman 和 Github 的开源项目 Postwoman，在开发的同时还需要维护 API 文档。

大多数接口在本地使用 Postman 测试：

左边为接口测试列表，中间是“老师获取课程详情”的测试，下方是返回 JSON 数据。

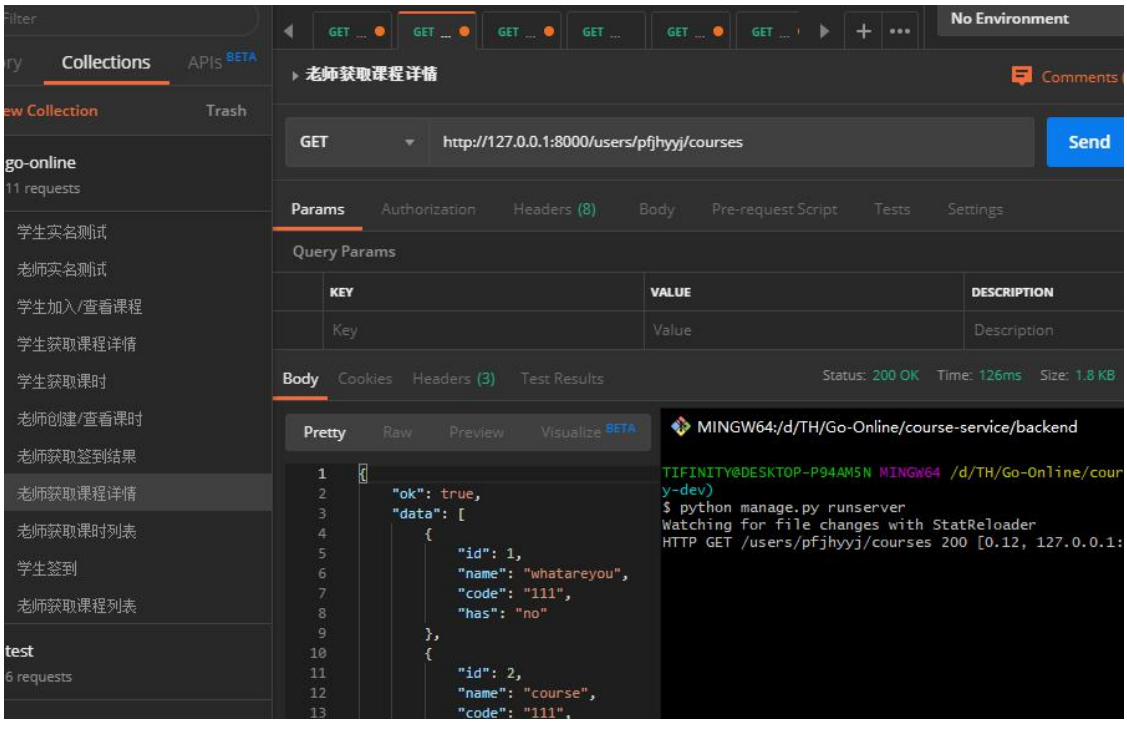

image-20191212115647953

Websocket 的接口使用 Postwoman 测试：

可以看到在 Postwoman 的网页上不断收到签到码和签到列表。

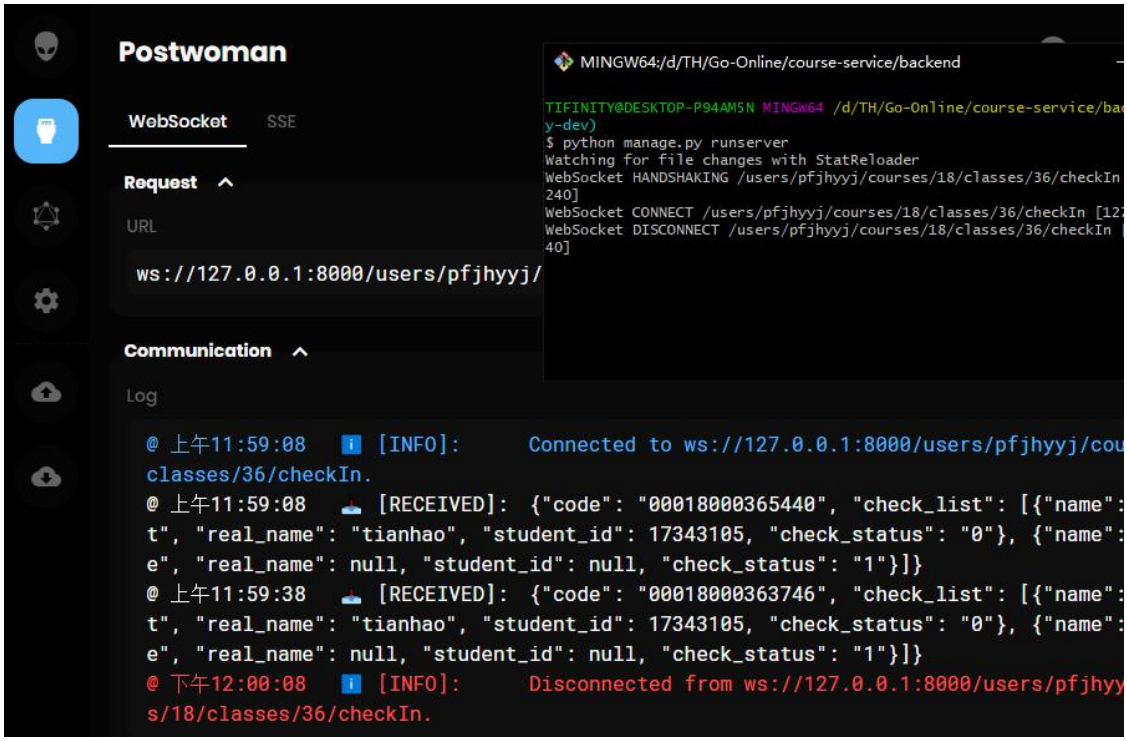

image-20191212120035453

JWT

GoOnline 使用 JWT 进行身份验证，判断当前是否登陆，用户是老师还是学生，通过签到功能的完成基本了解了 JWT 的分发与验证过程。

WebSocket

关于 WebSocket 的使用在实训报告 3，实训报告 4 中有作说明，不再赘述。这是以前没有接触过的技术，通过本次实训才了解到使用 WebSocket 可以在 Web 上进行全双工通信。

数据库

在 Danjgo 中使用 ORM，将每张表都当作类来操作，比较方便。

使用了 MySQL 和 MongoDB，MySQL 存储用户，课程等表，MongoDB 存储课程与该课程的学生的集合和签到集合等。在开发的过程中经常需要进行数据库的操作，

使用 Navicat Premium 12 进行数据库的可视化操作，从而测试自己写的函数，非常方便。

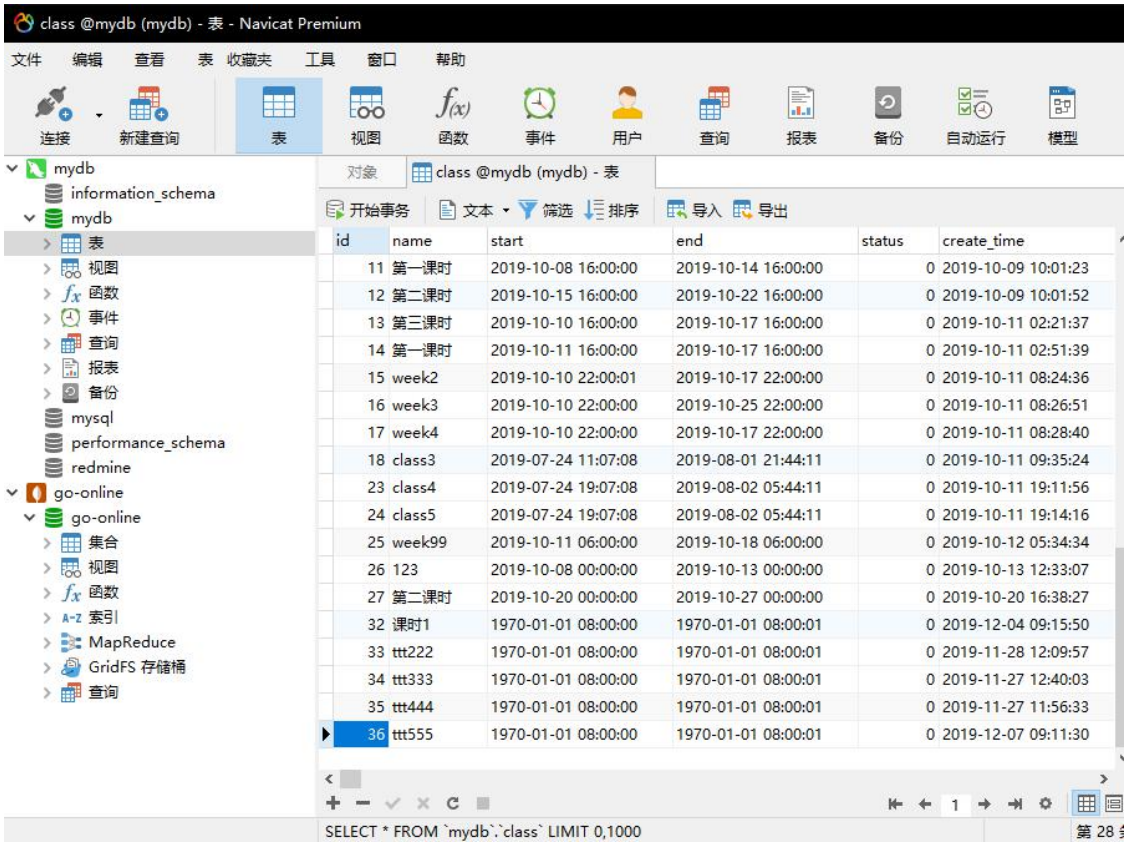

image-20191212115312582

Docker

实训的过程中没有用到，不过 GoOnline 的服务是跑在 Docker 上的，而且也是现在流行的技术，所以也简单学习了 Docker 的使用，具体学习过程已在实训报告 4 中说明。

最后

本次实训算是我第一次参与一个大项目的开发，虽然做的工作不多，但是对我来说学到的东西是非常多的。除了签到的开发过程，在 GoOnline 的技术分享会上也从其他成员的分享中收获了许多，自己也在分享会上分享了一点学习收获。

感谢唐玄昭师兄让我加入 GoOnline 团队，进行后端开发工作。

感谢张书博师兄对我所有问题地解答以及开发过程的指导。

感谢签到功能开发小组两位大哥带领我完成实训任务。

中级实训到此结束，自己的不足之处还是很多，希望以后继续努力。