## Mini Program 设计文档

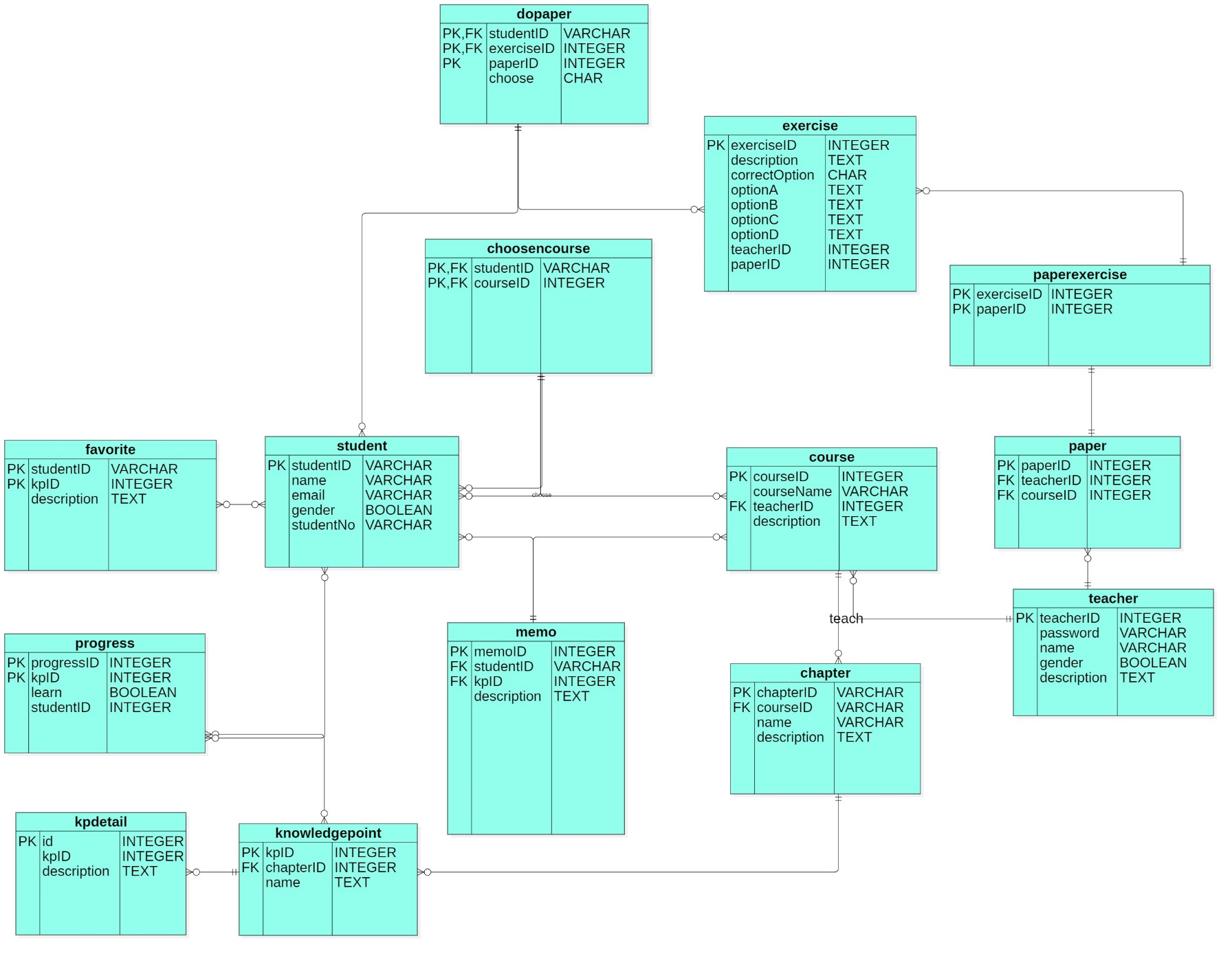
#### 项目组成

* **前端**
  + **微信端**
    - **使用微信的框架，实现一个微信端的小程序**
* **教师端**
  + **使用Vue框架实现一个教师端的浏览器管理功能**
* **后端**
  + **Spring Boot + MyBatis 实现的后台与数据库交互**
* **数据库**
  + **MySQL数据库**

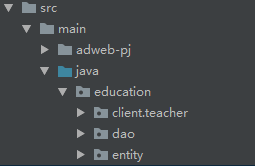
**注：后台只有一个，即数据库的访问层以及实体对象和映射只有一个，只是对业务的逻辑分层分为了student和teacher的操作**

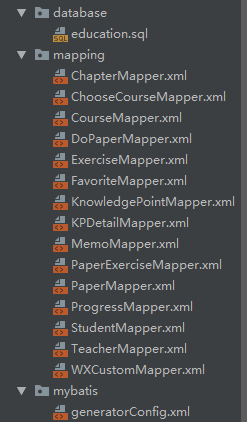
#### 实现功能

* **学生端后端**
  + **负责人：黄文浦**
  + **src/main/java/education/student 目录下**
  + **StudentController.java中**
  + **登录 user/loginAuth**
  + **获得和完善用户信息的接口,用id取得学生全部个人信息 user/getStu**
  + **更新个人信息 user/updataStu**
  + **CourseController.java中**
  + **得到学生所选课程 course/getCourse**
  + **得到全部课程 course/getAllCourse**
  + **学生添加课程 course/addcourse**
  + **得到course所有章节id，name，decription course/getChapter**
  + **得到对应章节的所有知识点 course/getkpoint**
  + **得到知识点对应每条小内容 course/getKpDetail**
  + **通过学生id查询到对应收藏的知识点 course/getFav**
  + **添加收藏 course/addFav**
  + **删除收藏 course/delFav**
  + **获取笔记列表 course/getNote**
  + **添加笔记 course/addNote**
  + **删除笔记 course/delNote**
  + **得到对应课程的测验题 如果没有数据会报错 course/getPaper**
  + **得到对应课程的测验题目 course/getExercise**
  + **记录学生做得习题答案 course/submitPaper**
* **数据库**
  + **负责人：安鑫**
  + Schema



* + 与Database相关的文件





* + - **dao-数据访问层**

一些Java文件，为接口，包含一些操作数据库的函数

* + - **entity：与数据库相关的试题Java对象**

一些Java文件，为类，包含由MyBatis生成的类和一些自定义的需要操作数据库的类

* + - **database：本次项目数据相关的脚本**

该数据库的脚本文件，能够新建的数据库和一些数据

* + - **mapping：映射文件**

具体的dao层中函数接口的实现，即一些SQL语句

* + - **mybatis：自动化生成的mybatis的配置文件**

自动化生成dao和entity的配置文件，只能根据已有的数据库生成

注：已将数据库部署在远程服务器，52.45.153.150:3306/education，通过root访问，可以修改其数据库，然后让写后端的可以直接使用，方便管理

#### 开发流程

* + - **数据库先行**

项目开始的起点应该是数据库，先行分析需求，然后构建关系型数据库，建表，将其部署在远程服务器上，该过程可以让管理数据库的人员只用修改远程服务器即可，然后所有后台开发人员使用该数据库，方便维护。根据数据库生成简单的数据库操作对象。

* + - **项目分解**

这是一个团队项目的重要分割点，出去数据库相关的人员，可以将一个网络项目分为前后端，根据前端可以分为不同的前端开发，根据后端可以分为不同逻辑业务的开发或者不同处理层的开发，这里我们选择了不同业务让不同的人员的开发，这样可以让其耦合性降低，各自开发均可独自进行前期测试。

* + - **模块测试**

各个项目部分进行各自独立的模块测试

* + - **集成测试**

集成整个项目的测试，查看项目

* + - **项目部署测试**

#### 附加功能

* + - 使用Docker部署，具体的是Docker部署Spring Boot的后台项目，Docker部署Vue教师前端，直接部署MySQL在远程服务器上

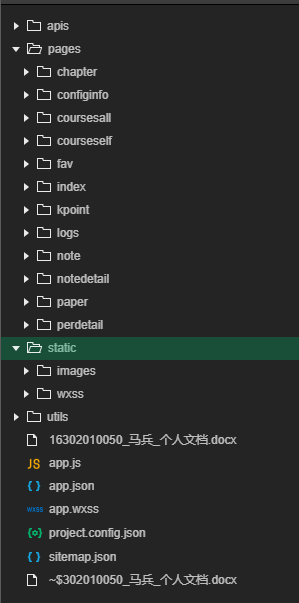
原理：通过在系统运行一个容器，而容器中有相关的运行环境，这样就进行了一层封装，减少了环境带来的差异，如同jvm对Java程序的运行，通过部署在远程云上，程序一直处于运行状态，远程服务器暴露该相关的端口，那么就可以访问远程服务器上该相关的程序

* + - 场景美观作为组长觉得很不错，所以可以算得上是附加功能

**Adweb-pj个人说明文档**

负责内容：微信小程序前端开发，微信小程序后端API的定义

具体实现代码目录：



├─apis /主要是lab2中使用过的request.js封装网络请求和一些具体事务请求函数封装

├─pages /微信小程序页面目录

│ ├─chapter /展示课程章节

│ ├─configinfo /主页之个人信息bar

│ ├─coursesall /全部课程界面

│ ├─courseself /学生已选课程界面

│ ├─fav /收藏界面

│ ├─index /默认界面

│ ├─kpoint /知识点界面

│ │ └─kpdetail /具体学习界面

│ ├─logs/ log界面

│ ├─note /笔记界面

│ ├─notedetail /笔记详情界面，包括修改、删除笔记

│ ├─paper /试卷界面

│ └─perdetail /个人信息更改界面

├─static /静态文件目录

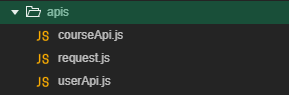
│ ├─images /小程序使用的图片存放目录

│ └─wxss / 小程序使用的第三方wxss库存放目录

└─utils /公共操作函数存放

1. **Apis**

主要使用request.js封装请求，具体代码不贴了

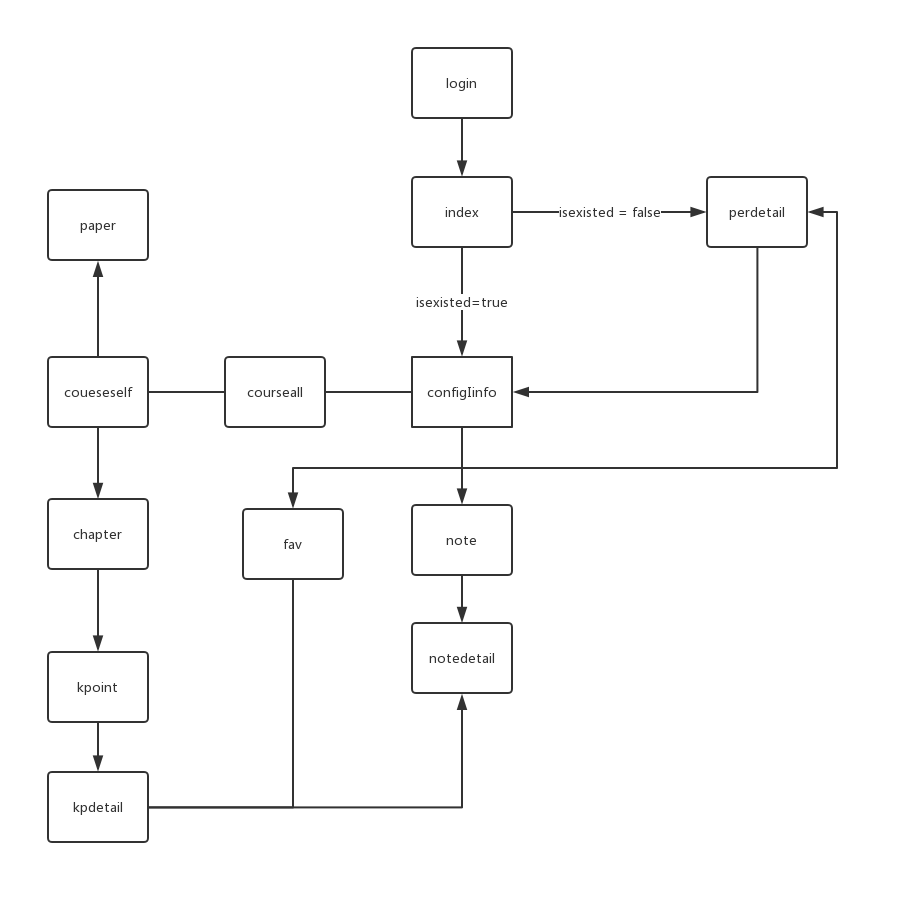


courseApi：所有与课程相关的网络请求，包括getcourse，getchapter等等

userApi： 所有与用户相关的网络请求，包括loginAuth，getStu等等

1. **Pages**

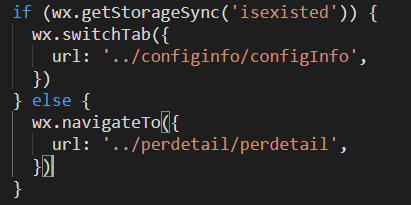
页面交互逻辑如下图：



2.1登陆

点击授权 => 发送res.code到服务器去微信官方API得到openid，并存储为全局变量

2.2 完善个人信息



从服务器得到关键参数isexisted判断是否为第一次登陆小程序，是跳转到个人信息界面进行完善，否则直接进入主界面tabBar的个人信息页

2.3 tabBar

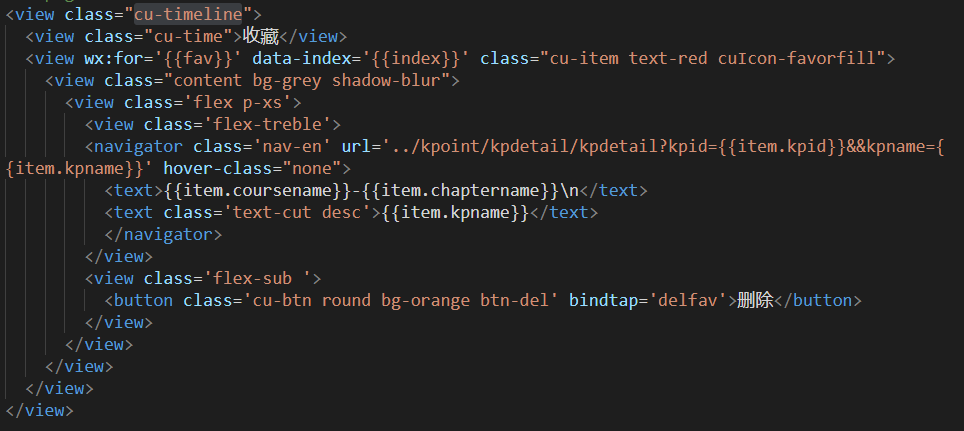
三项对应三个page，实现代码



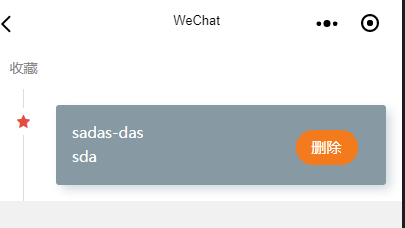
。

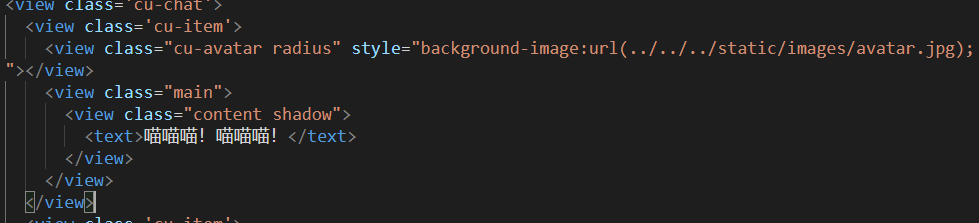
2.4 configinfo页面下有三个按钮，点击进入对应的界面收藏、笔记、个人信息

收藏、笔记、课程章节、知识点的展示使用的都是colorui的一个组件cu-timeline

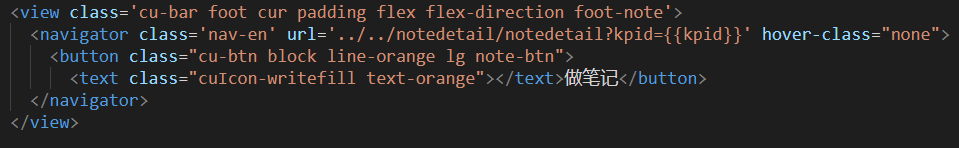


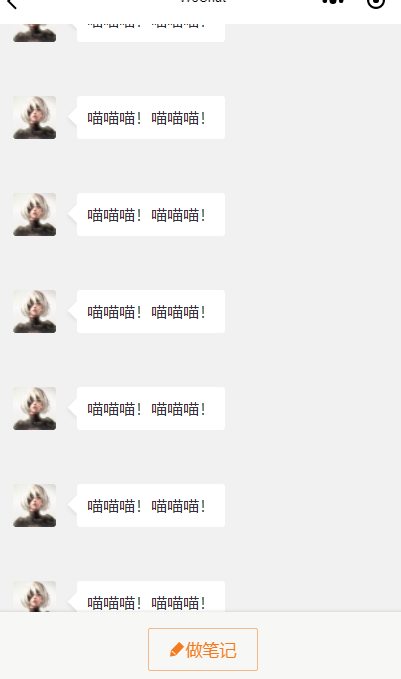
效果如下：



* 1. 主要学习界面，kpdetail，对话式展示知识点内容，在下方可以选择点击做笔记。使用colorui的chat组件

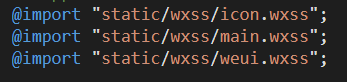
末尾固定显示记笔记一栏，使用bar和foot配合设置page的wxss为padding-bottom 100rpx实现



最终效果如图：  


**3. Static**

主要介绍wxss，使用了weui和colorui两个库，直接在app.wxss使用@import导入即可在全局使用。



**教师端的项目文档**

**接口：（加粗的为前后端需要统一的字段）**

接口名称： **getStuPapList**（得到学生问卷完成情况的接口）

方法： **post**

传递参数： **courseID**: this.$route.params.courseID,（为课程id，string类型）

**papperID**: this.$route.params.papperID,（为问卷id，string类型）

接收参数：this.finishStus = response.**finishStus**;

（完成问卷的学生的列表，每个学生包含属性name）

this.unfinishStus = response.**unfinishStus**;

（没有完成问卷的学生的列表，每个学生包含属性name）

接口名称： **getStuList**（得到某一门课的选课学生情况的接口）

方法： **post**

传递参数： **courseID**: this.$route.params.courseID,（为课程id，string类型）

接收参数：this.myStus = response.**myStus**;（学生列表，每个学生有属性**name**）

接口名称： **getClasses**（得到某一个老师所有开设的课程列表的接口）

方法： **post**

传递参数： **teacherID**: this.username（用户的id）

接收参数：this.classes = response.**courseList**

（返回的是课程列表，每一个课程包含属性（如下json数据））：

{

**courseName**: '多媒体',

**courseID**: 123454532617847,

**stunum**: 56,

**discription**: '多媒体技术，学了都说好'

},

接口名称： **deleteChapter**（删除一门课某一个章节的接口）

方法： **post**

传递参数： **chapterID**: this.$routr.params.chapterID（章节的id）

接收参数：无

接口名称： **deletePoints**（删除一个章节某个知识点的接口）

方法： **post**

传递参数： **pointID**: this.points[index].pointID,（知识点的id）

接收参数：无

接口名称： **getKnowledge**（进入某一个章节后获取知识点的的接口）

方法： **post**

传递参数： **chapterID**: this.$routr.params.chapterID（章节的id）

接收参数：this.points = response.**knowledgePoints**

（为知识点的列表，每个知识点包含属性如以下Json数据）

{

**text**: '..',（知识点的名字）

**pointID**: '.',（知识点的id）

**contents**: [ （知识点细节的列表）

{

**con**\_**det**: '..', （每个细节包含它的内容）

}

]

}

接口名称： **getMyPappers**（获取一门课所有已经发布问卷的接口）

方法： **post**

传递参数： **courseID**: this.$route.params.courseID,（课程id）

接收参数： this.studentnum = response.**studentnum**;（学生总数）

this.questions = response.**papperList**;

（问卷的list，每一个问卷包含json数据如下）

{

**ques\_title**: '问卷1',（问卷名字）

**papperId:** '12345',（问卷id）

**finishnum:** '55' （问卷已经作答的学生人数）

},

接口名称： **addChapter**（添加一个新的章节）

方法： **post**

传递参数： **courseID**: this.$route.params.id,（课程的id）

**chapterName**: this.cha\_name,（章节的名字）

**description**: '.'（章节的描述，传递一个无用的数据，作废）

接收参数：this.chapters = response.**chapters**（返回章节的列表）包含如下属性

{

**chapterName**: '第一章',（章节的名称）

**chapterID**: '12345'（章节的id）

}

接口名称： **getChapter**（获取某门课所有章节的接口）

方法： **post**

传递参数： **courseID**: this.$route.params.id（课程的id）

接收参数：this.chapters = response.**chapters**（返回章节的列表）包含如下属性

{

**chapterName**: '第一章',（章节的名称）

**chapterID**: '12345'（章节的id）

}

接口名称： **addCourse**（新添加一门课程的接口）

方法： **post**

传递参数： **teacherID**: this.username,（教师的id）

**courseName**: this.sub\_name,（课程的名称）

**discription**: this.sub\_brief,（课程的简介）

接收参数：无

接口名称： **delPapper**（删除问卷的接口）

方法： **post**

传递参数： **papperID**: this.$route.params.papperID,（传递问卷的id）

接收参数：无

接口名称： **getPapperDet**（查看问卷详细信息的接口）

方法： **post**

传递参数： **papperID**: this.$route.params.papperID,（问卷的id ）

接收参数：this.pappername = response.**pappername**;（问卷的名字）

this.questions = response.**questions**;

（问卷的内容——题目信息的列表，每一个题目包含如下信息）

{

**ques\_title**: 'qq',（问题描述）

**optionA**: '123',（选项A的内容）

**optionB**: '22', （选项B的内容）

**optionC**: '24', （选项C的内容）

**optionD**: '88', （选项D的内容）

**num**: 'D',（正确答案的选项）

**correctnum**: '66'（选对的学生人数）

},

this.finishnum = response.**finishnum**;（总共作答此问卷的学生人数）

接口名称： **getKpDtails**（得到知识点详情的接口）

方法： **post**

传递参数： **knowledgeID**: this.$route.params.pointID（知识点的id）

接收参数：this.point\_name = response.**point\_name**;（知识点的名字）

this.point\_details = response**.point\_details**（知识点细节的列表，每个细节数据如下）

{

**point\_text:** ''（为某个知识点的细节的内容）

}

接口名称： **addKnowledge**（新建知识点的接口）

方法： **post**

传递参数： **knowledgeName**: this.point\_name,（知识点的名字）

**chapterID**: this.$route.params.chapterID,（章节的id）

**knowledgePoints**: this.point\_details,（知识点细节的列表）结构如下

{

**point\_text**: ''（知识点细节的内容）

}

接收参数：无

接口名称： **modKnowledge**（修改知识点的接口）

方法： **post**

传递参数： **pointID**: this.$route.params.pointID,（知识点的id）

**knowledgeName**: this.point\_name,（知识点的名字）

**point**\_**details**: this.point\_details（知识点细节列表，详情见上一个接口）

接收参数：无

接口名称： **addPapper**（添加问卷的接口）

方法： **post**

传递参数： **pappername**: this.pappername,（问卷的名字）

**courseID**: this.$route.params.courseID,（课程的id）

**questions**: this.questions,（问卷中问题的列表，详细数据如下）

{

**ques\_title:** '',（问题题目）

**optionA:** '', （选项A）

**optionB:** '',（B）

**optionC:** '',（C）

**optionD:** '',（D）

**num:** '',（正确答案）

}

接收参数：无

接口名称： **register**（注册）

方法： **post**

传递参数： **username**: this.input,（用户名）

**password**: this.pass（密码）

接收参数：response.**isnew** === true

（接收isnew的布尔值，确认数据库中没有这个用户）

接口名称： **login**（登录接口）

方法： **post**

传递参数： **username**: this.input,（用户名）

**password**: this.pass （密码）

接收参数：response.**success（布尔值，判断能否登录成功）**

response.**ID**（存入cookie的id）

接口名称： **modify**（修改个人信息）

方法： **post**

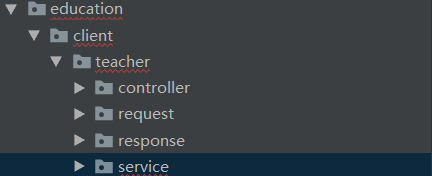
传递参数： **username**: this.user,（用户名）

**password**: this.newpass（新的密码）

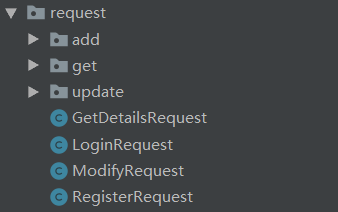
接收参数：无

网页教师后端

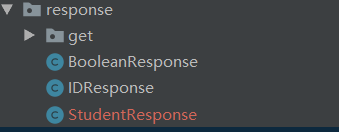
代码主要在 src/main/java/education/teacher 目录下



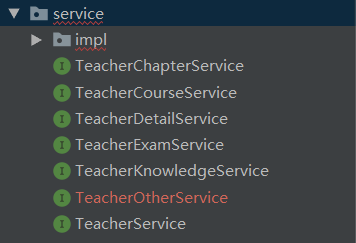
request包：负责封装教师前端传来的数据（add、get、update包分别是课程相关的新增、查看、更新请求；其他是试题新增查看更新、教师登陆注册修改密码、查看选课学习情况等）



response包：负责封装服务器发送回前端的数据（get包是教师前端发来查看请求时返回的数据；BooleanResponse是修改操作返回的布尔值封装；IDResponse是添加新对象时返回对应在数据库的唯一ID；StudentResponse是返回学生链表时所用的封装）



service包：业务层接口及实现，负责与数据库的交互（主要是试题、课程、章节、知识点及细节的新增、修改、查看；选课情况、学习情况、试题誊写情况、试题对错情况的查看等）



controller包：负责接收教师前端发来的请求，调用业务层进行处理，返回数据（实现的功能有试题、课程、章节、知识点及细节的新增、修改、查看；选课情况、学习情况、试题誊写情况、试题对错情况的查看；教师登陆注册修改密码等等）

