<云作业平台>

软件架构文档

版本 <1.2>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <7/11/2020> | <1.0> | <初步设定> | <第五组全员> |
| <16/11/2020> | <1.1> | <增加数据视图的描述，对质量部分进行细化> | <孙云帆> |
| <18/11/2020> | <1.2> | <对设计战术进行修改> | <孙云帆> |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 7

3.1 概述 7

3.2 在构架方面具有重要意义的设计包 7

4. 进程视图 9

5. 部署视图 10

6. 实现视图 11

7. 数据视图 13

8. 质量 14

8.1.1 <性能需求一> 14

8.1.2 <性能需求二> 14

8.1.3 <性能需求三> 14

8.1.4 <性能需求四> 14

8.1.5 <性能需求五> 14

8.1.6 <可靠性需求一> 14

8.1.7 <可靠性需求二> 14

8.1.8 <可靠性需求三> 14

8.1.9 <可靠性需求四> 14

8.1.10 <可靠性需求五> 14

8.1.11 <可靠性需求六> 15

软件架构文档 （简化版）

# 简介

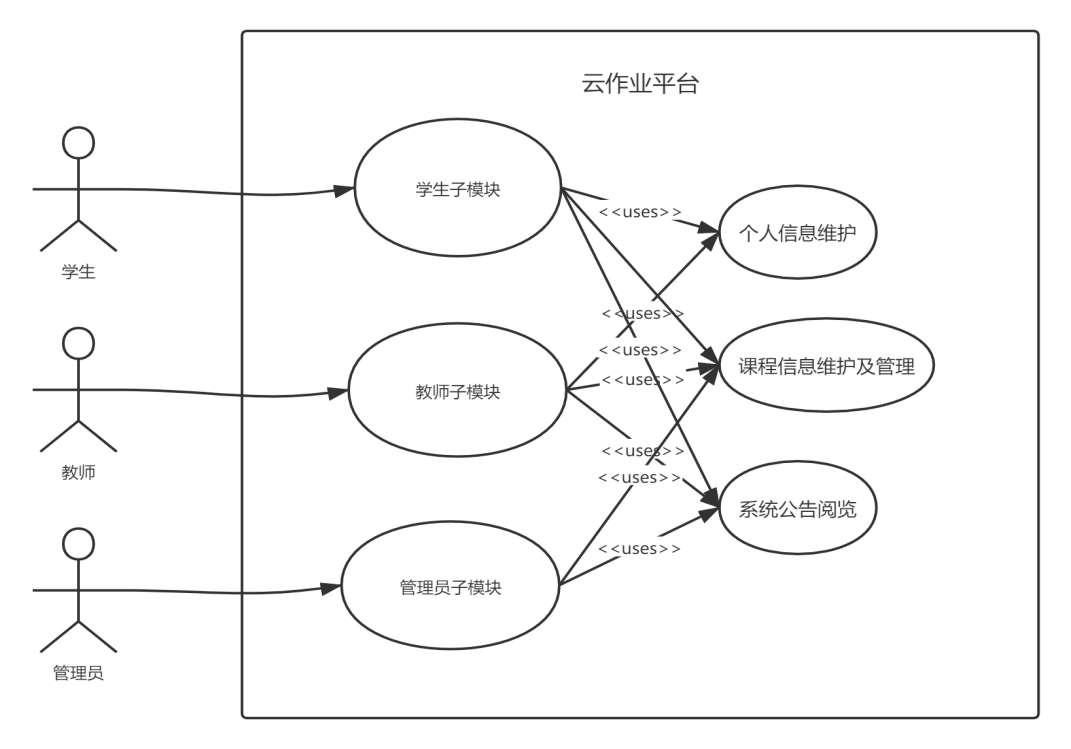
## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

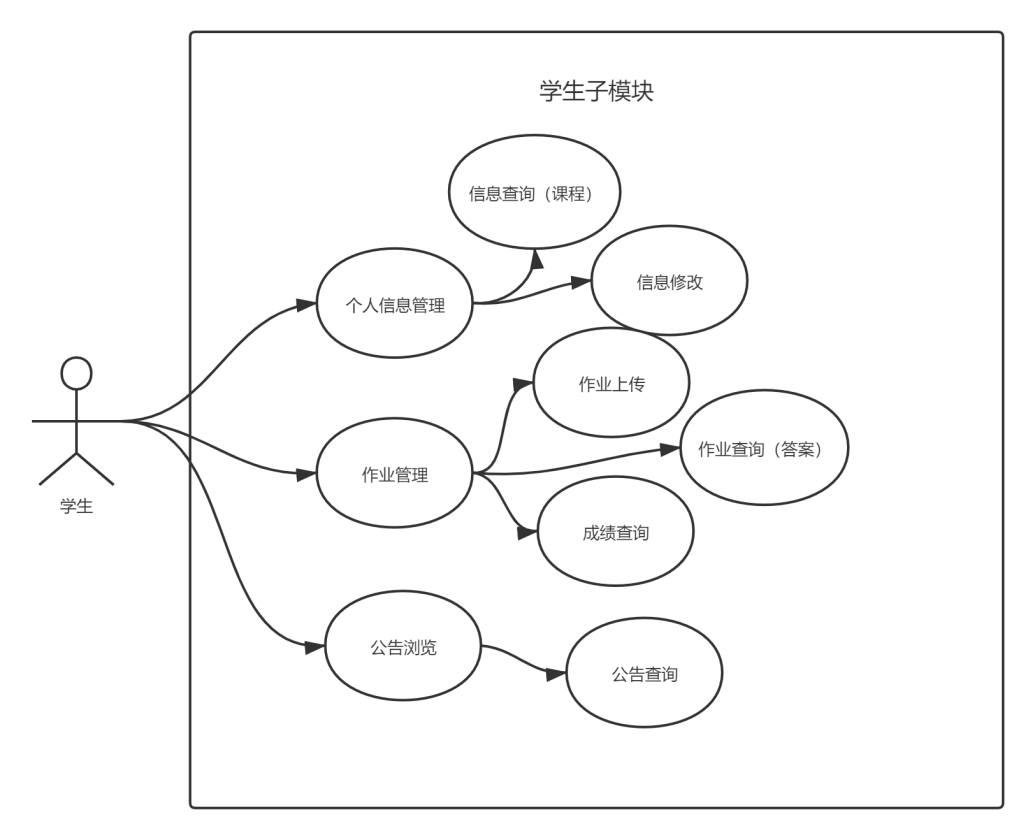
## 参考资料

[1]沈备军，陈昊鹏，陈雨亭.软件工程实践[M].北京：高等教育出版社.2013.2

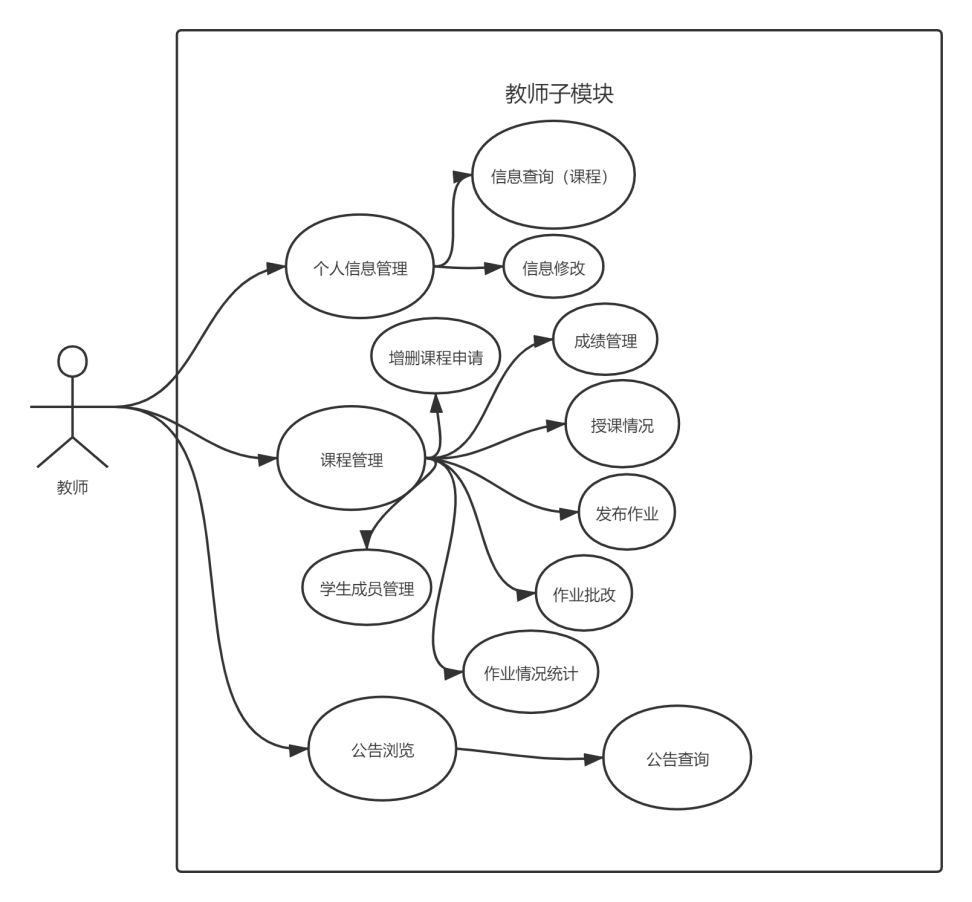
# 用例视图



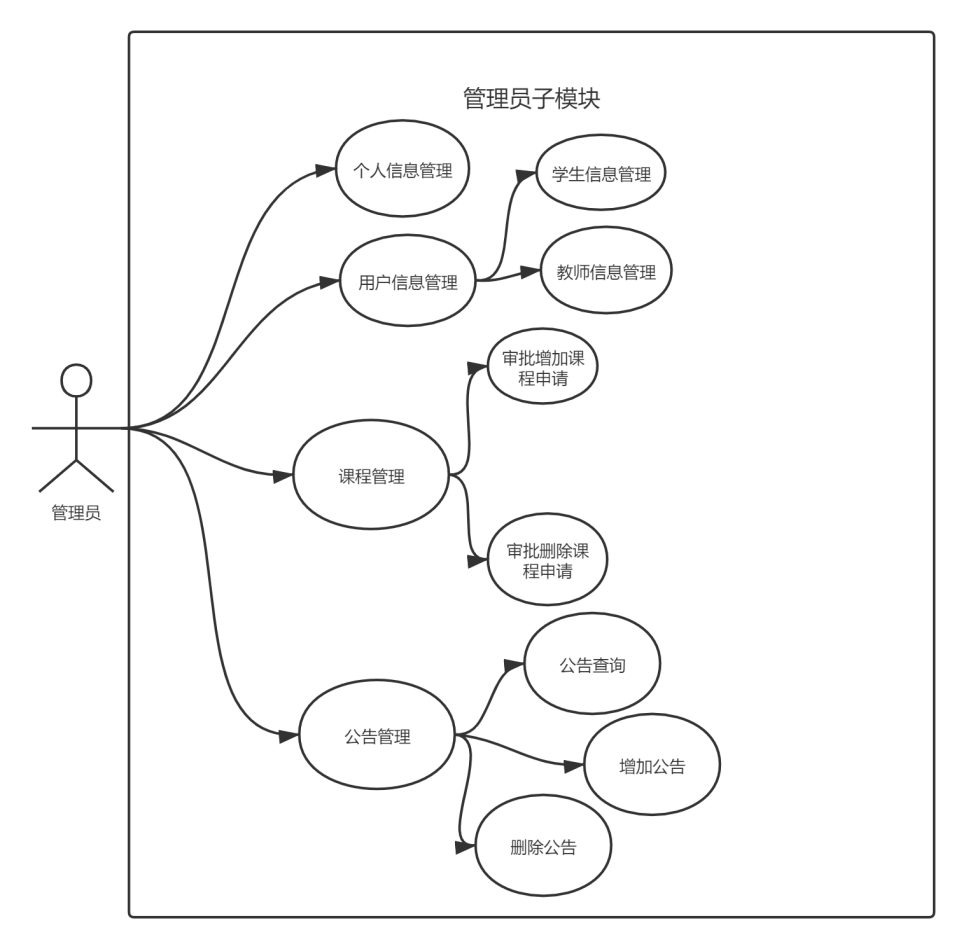
用例图总览



学生子模块



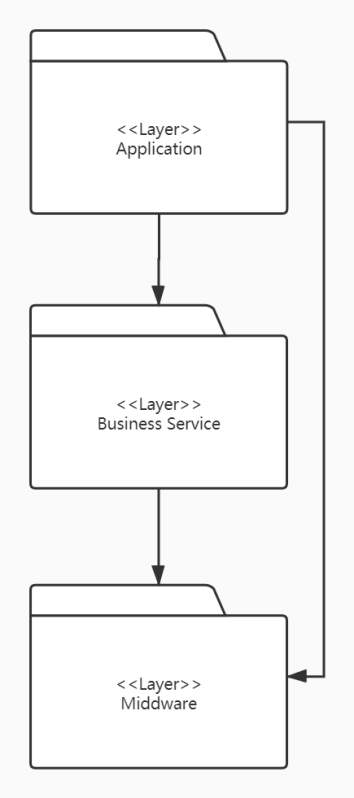
教师子模块



管理员子模块

# 逻辑视图

## 概述



云作业平台系统的逻辑视图主要由三层组成，分别是Application层，Business Service层以及Middle层。

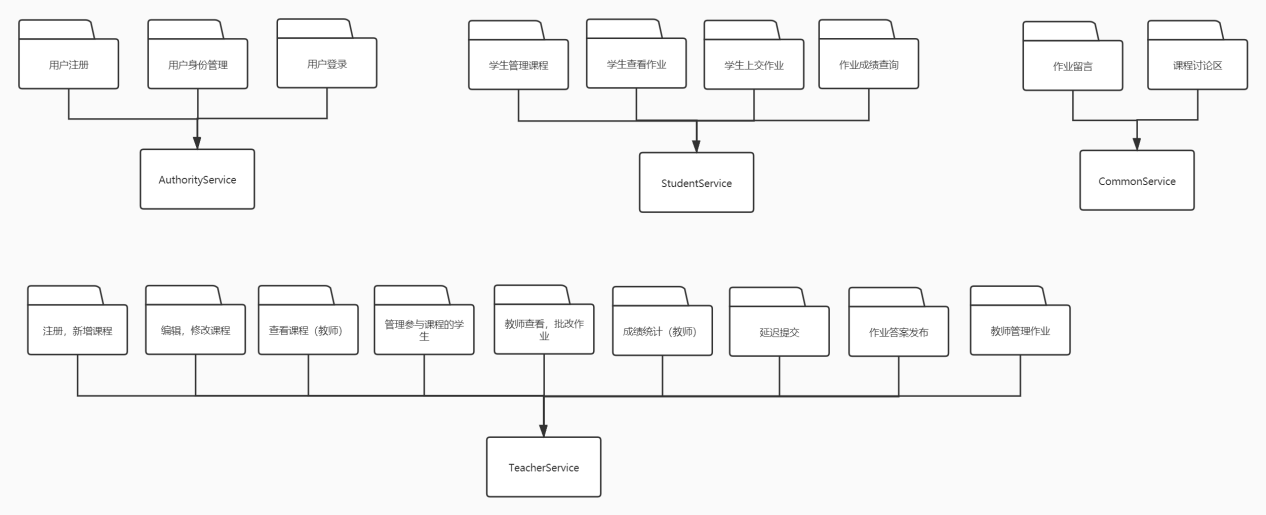
Application层主要由相应各种用户界面请求的动作类组成，它会调用Business Service层中的函数进行业务逻辑处理，同时根据结果显示不同的界面给用户。

Business Service层主要完成实际的业务逻辑，同时包括与数据库的表对应的实体类，以及访问数据库的DAO类。

Middle层为SSM框架的函数库。

## 在构架方面具有重要意义的设计包

1.Application层

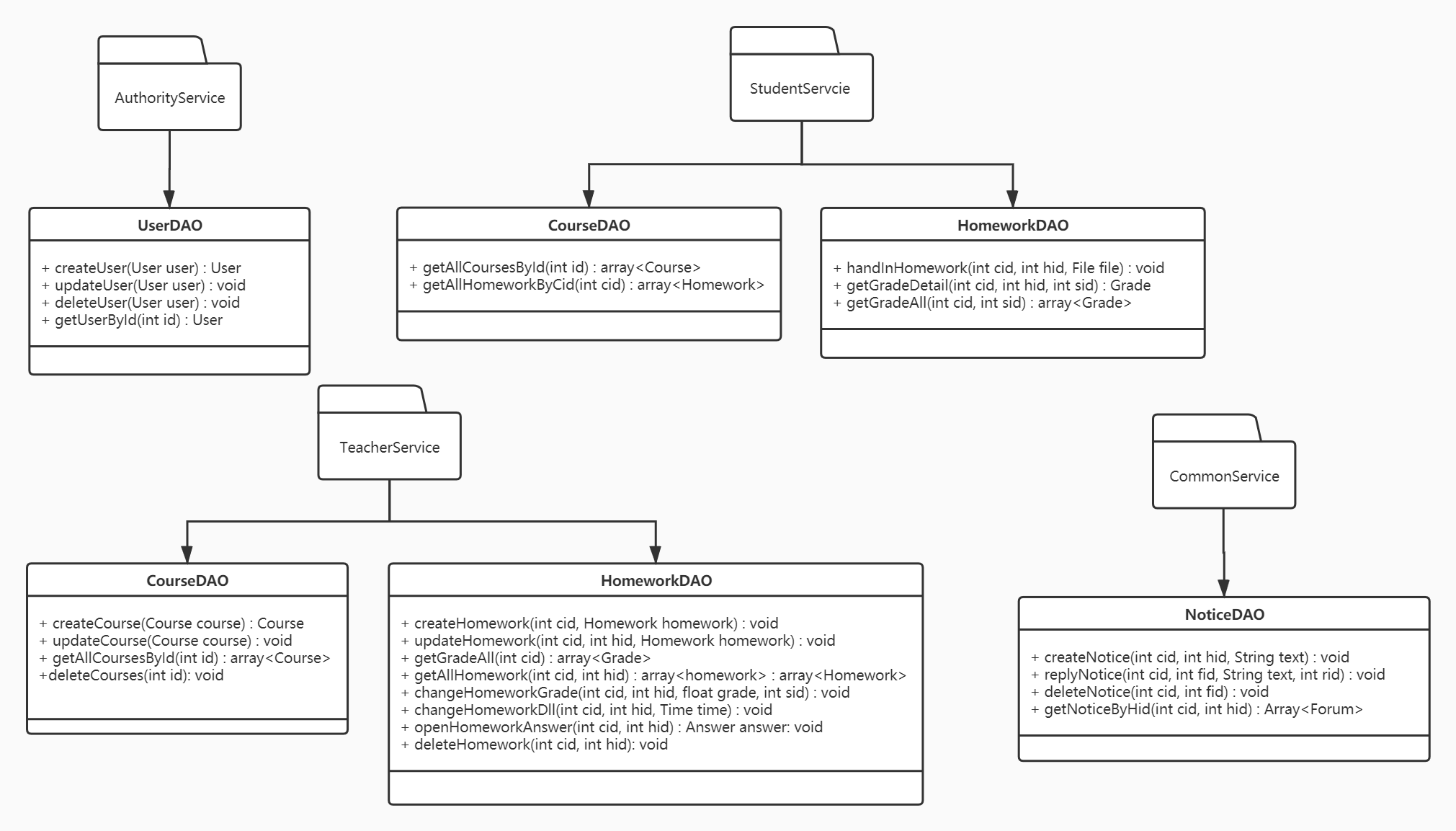


Application层由4个子包组成，这四个子包分别是Authority Service包, Student Service包, Common Service包, Teacher Service包，这四个子包基本覆盖云作业平台的全部功能。

2.Business Service层

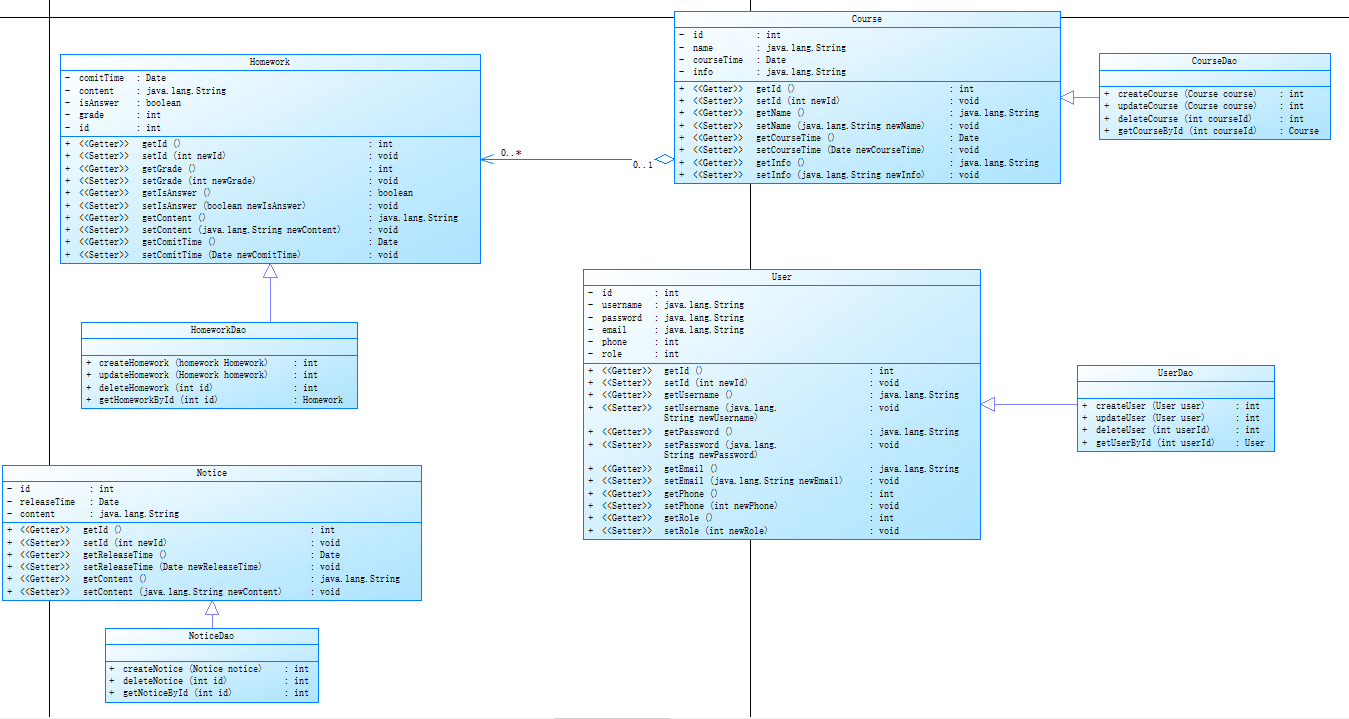
Business Service层包括Service和Model两个子包

(1)Service包



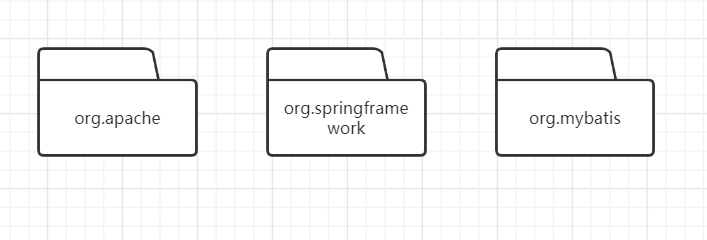
四个子包中，Authority Service包负责用户的登录认证与个人信息的修改；Student Service包负责学生端的提交作业，查看作业详情与查看成绩等功能；Teacher Service包负责教师端的布置作业，查看全部作业，查看学生成绩，登记成绩等一系列教师端功能；Common Service则是针对公众模块功能（讨论区与公告栏）的子包。

（2）Model包



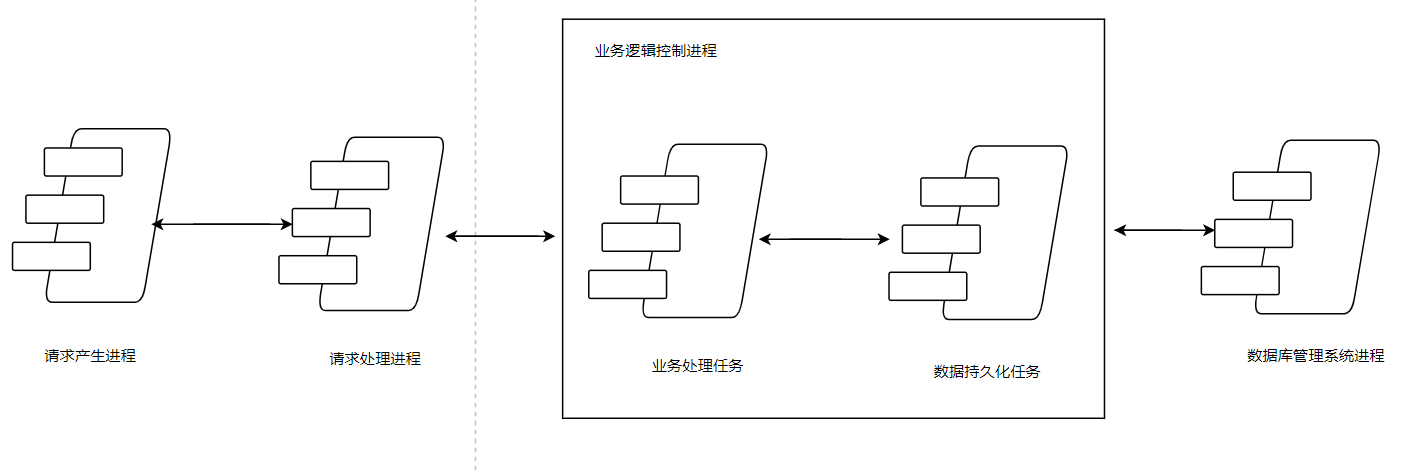
Model包由Homework，Course, User, Notice四个实体类以及其对应的DAO类组成。

3.Middleware层



由于使用SSM框架，使用Spring+Spring MVC+Mybatis的函数库。

# 进程视图



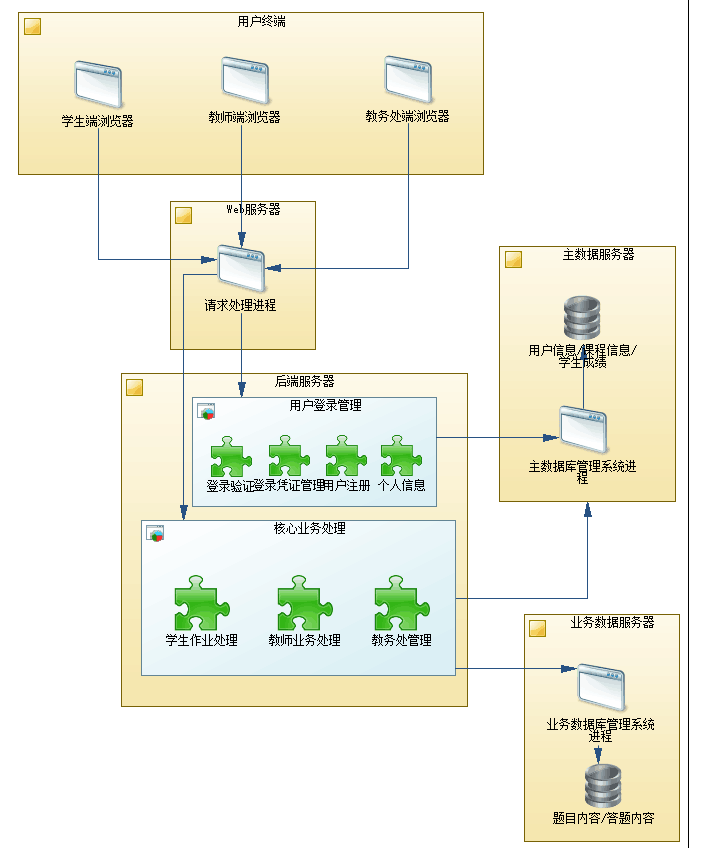
请求产生进程：用户访问所使用的浏览器

请求处理进程：web层各种任务，例：多用户登录页面查看各自作业，同时提交上传作业等

业务逻辑进程：两部分任务通过Tomcat部署处理各种业务，jdbc和mysql数据服务器连接

数据库管理系统进程：mysql数据库

# 部署视图



1. 用户终端

用户使用的电脑、手机等设备，用户通过浏览器访问系统，按照用户身份的不同进行相应的操作。

1. Web服务器

前端系统通过nginx部署，根据用户的操作像后端发送相应的请求并向用户返回处理结果。

1. 后端服务器

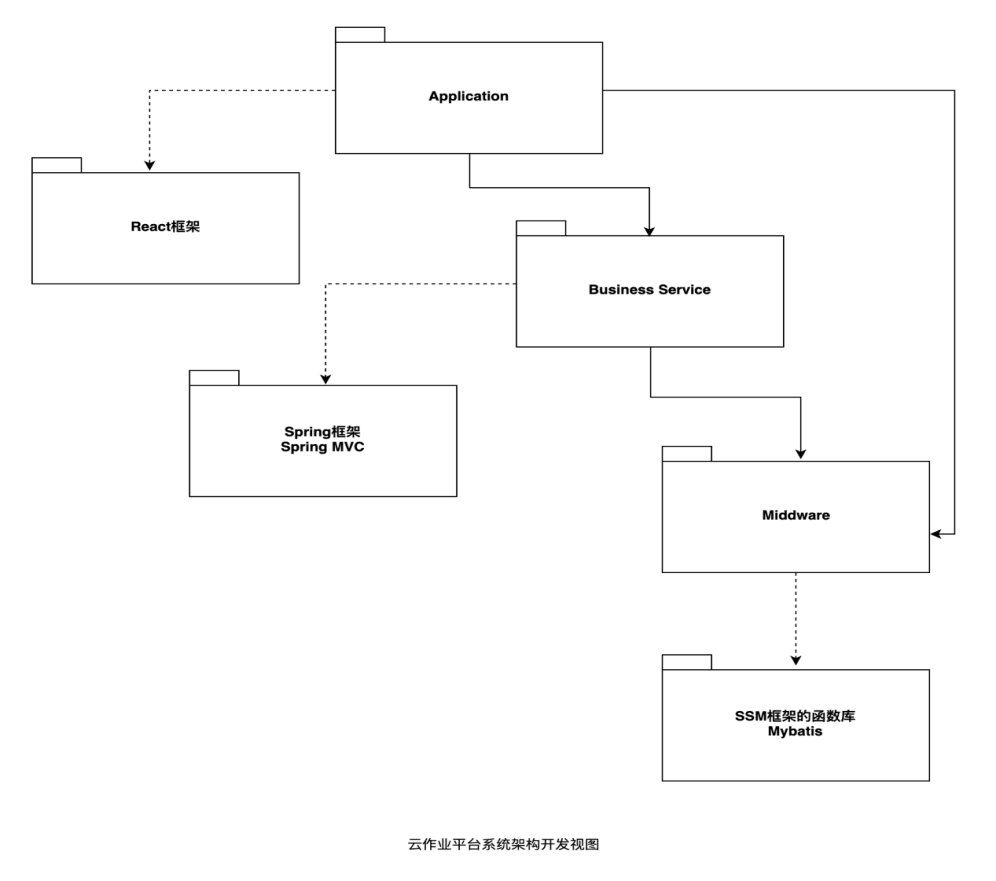
后端系统通过Tomcat部署，处理各种业务，根据前端发送的请求进行业务处理。 后端系统通过jdbc和数据服务器连接

1. 数据服务器

数据服务器运行mysql8.0.22数据库。其分为主数据服务器和业务数据服务器。主数据服务器负责处理用户信息、课程信息、学生成绩等主数据，着重读取和数据安全性优化；业务数据服务器负责处理题目和学生答题内容，着重读写速度优化，尤其是极短时间内的高并发读写。

# 实现视图

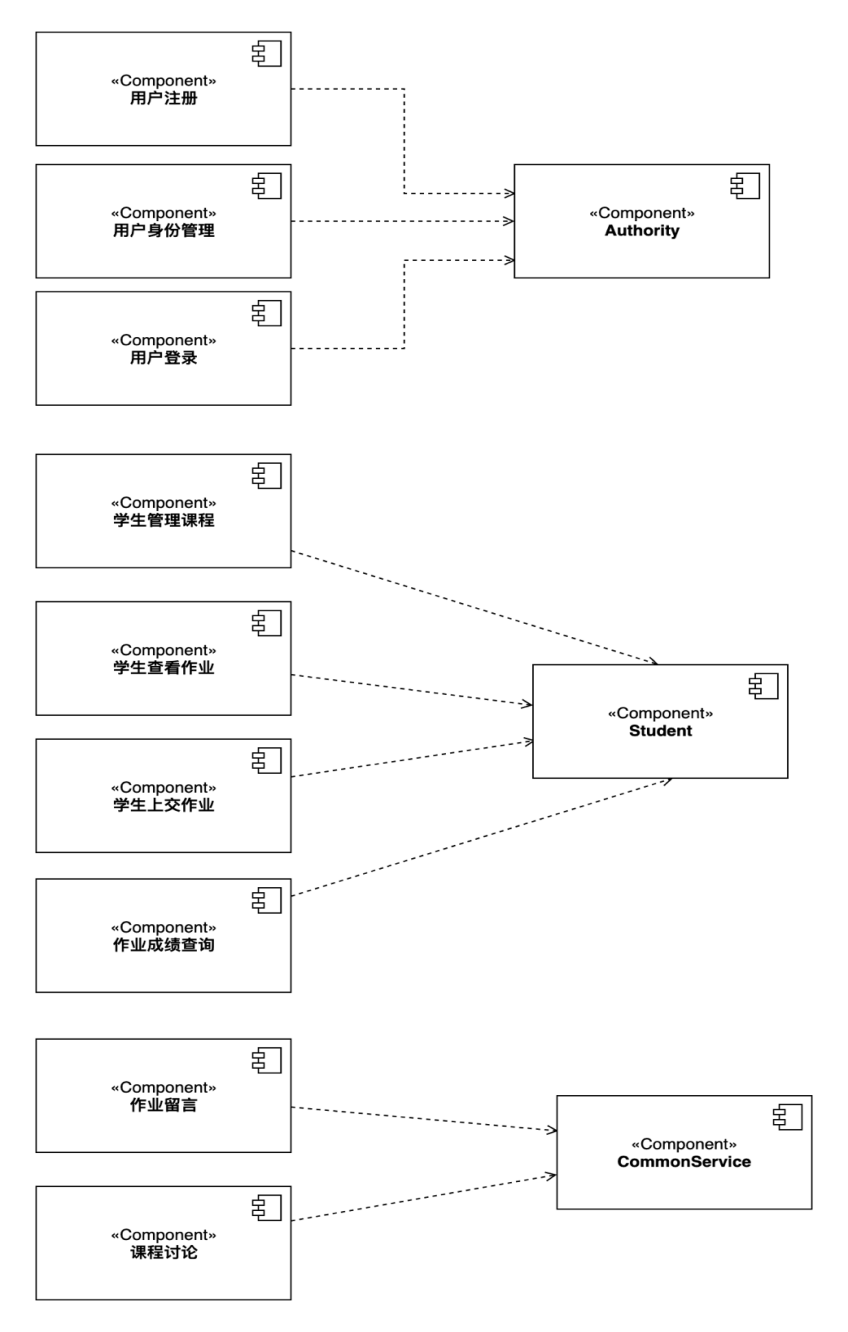
1. 云作业平台系统架构开发视图

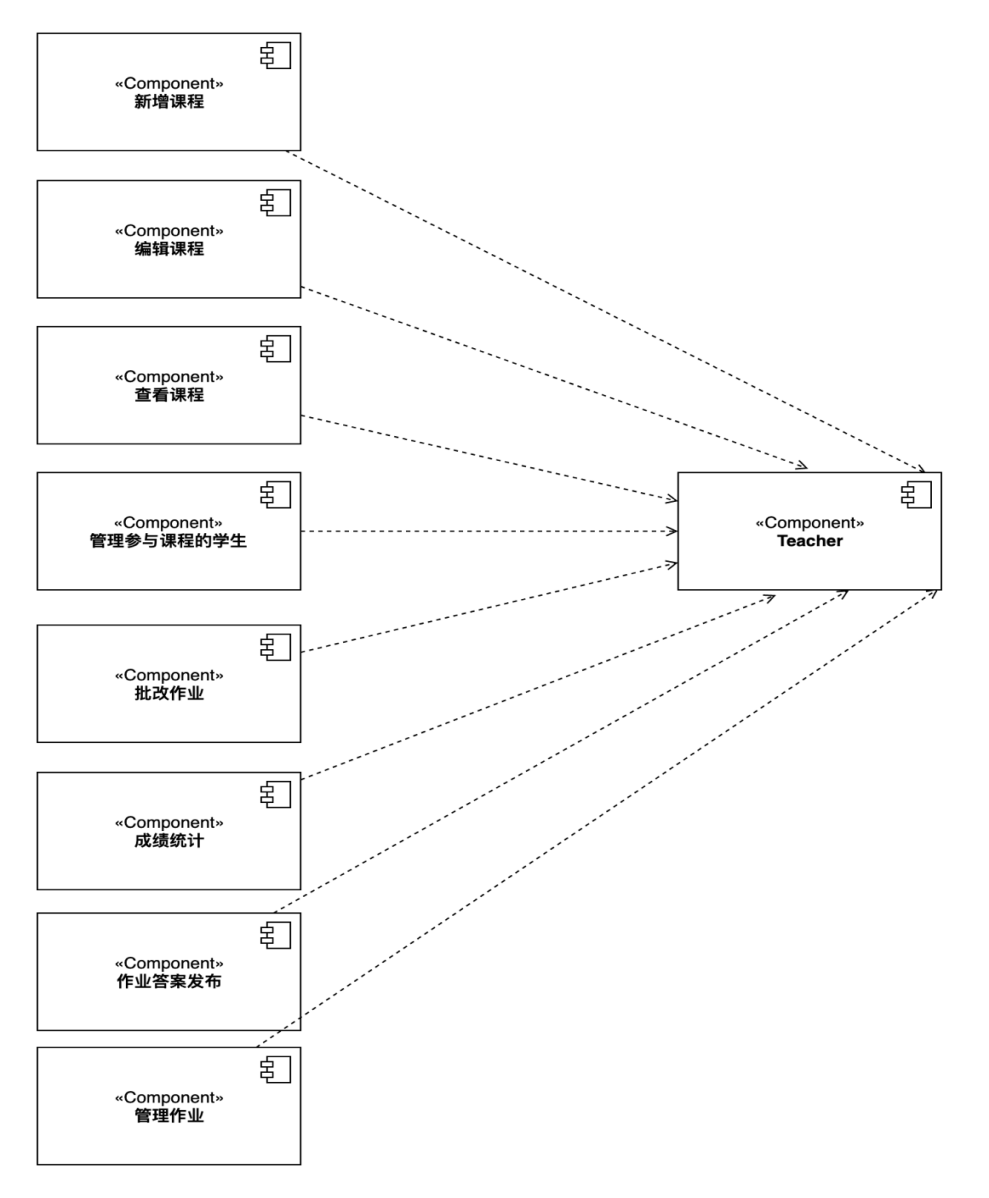


2. 开发架构的设计

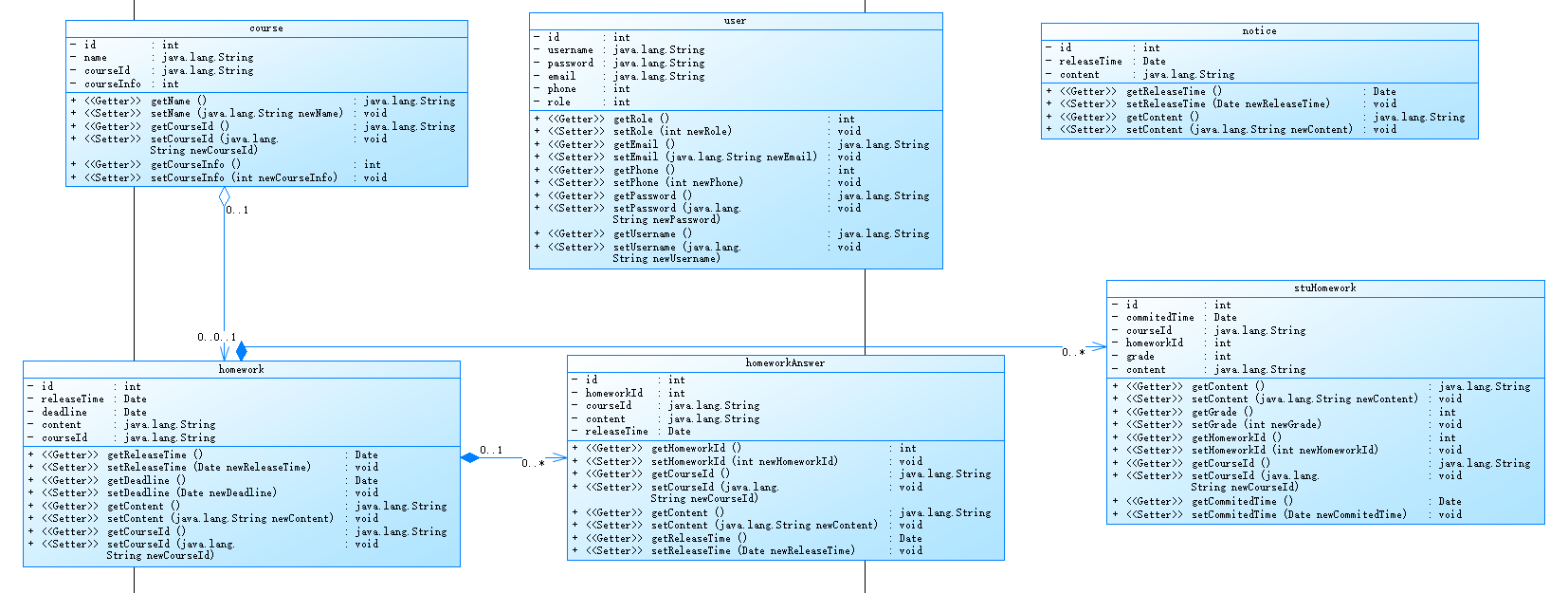
采用水平分割方案，将主要的三个包分为三组开发人员进行开发。即将WebPages分给一组擅长页面制作，具有一定美工能力的开发者，将Spring Class包分给对Spring框架实现业务逻辑非常熟练的开发者，将Hibernate Class包分给对数据库结构与操作熟悉的开发者。

1. 组件图





# 数据视图



数据库中有course，homework，stuHomework，user，notice，homeworkAnswer六个类。Notice和user类是相对独立的两个类，course对homework是一对多的关系，每个课程可以布置多个homework，而homework对stuHomework是一对多的关系，每个homework可以有多个学生提交的答案，并且每个homework有标准答案homeworkAnswer。

# 质量

### <性能需求一>

对事务的响应时间（平均、最长）,在1000并发量下,最长的响应时间不超过3s(不包括网络延迟),如教师批改作业后上传,多个学生同时上传作业等的情形。对于维护的管理员，其所有请求的响应时间不应该超过2s(不包括网络延迟)。

### <性能需求二>

本系统对于吞吐量的需求为: 每秒处理的事务数需超过300,以保证对于在一千并发下对响应时间的需求满足。

### <性能需求三>

本系统对于容量的需求为: 可以容纳至少50000名学生以及4000名教师的注册, 同时支持200个事务。

### <性能需求四>

本系统对于降级模式的需求为：当进入降级模式时,系统将只提供网页浏览功能,不接受作业上传/批改，课程的更改删除增加等。

### <性能需求五>

资源利用情况的需求为: 对于挂载本系统的服务器至少1GB内存、100GB的磁盘空间并且已经连接了Internet。同时网络连接的速度也应大于10MB/S.

对于客户端要求: 已经连接了Internet并且至少拥有1GB的磁盘空间,网络连接的速度大于5MB/S.

### <可靠性需求一>

系统的可用时间应达98%以上,持续可运行时间应达1500小时以上。除一般用户以外,同时设有管理员账户进行维护。

### <可靠性需求二>

平均故障间隔时间 (MTBF)是指两次相邻故障之间的工作时间的平均值,本系统的平均故障时间应为两个月以上。

### <可靠性需求三>

平均修复时间 (MTTR)是指系统在发生故障后可以暂停运行的时间,本系统的平均修复时间应为5小时以下。

### <可靠性需求四>

对于本系统的精确度输出要求如下:1.对于时间计量,应精确到秒;2.对于学生作业成绩计量,应精准到小数点后两位。

### <可靠性需求五>

最高错误或缺陷率通常表示为每千行代码的错误数目 (bugs/KLOC) 或每个功能点的错误数目 (bugs/function-point)。本系统要求缺陷率为:每千行代码的错误数目少于30个。

### <可靠性需求六>

错误或缺陷率方面的需求:

1. 小错误:网页部分异常,如图片无法显示,CSS样式异常等

错误率: 应小于9%

1. 大错误:向服务器发出请求无响应等

错误率: 应小于1%

1. 严重错误: 系统崩溃,数据库完全被锁定,数据丢失

错误率: 应小于0.01%

## 质量设计战术

### 可测试性设计战术

使用Cypress进行前端的端到端测试，要求覆盖95%以上的页面和按钮。

使用Jest进行前端单元测试，要求覆盖line达到95%+（去除request影响）。

使用Junit进行后端单元测试，要求覆盖自己代码line达到95%。

使用Postman进行后端API接口测试，具体接口相应要求如8.1所示。

### 性能设计战术

对存储的图片大小进行限制。

对数据库学号工号部分添加索引，对数据库部分表的结构进行优化。

存有数据库副本，暂时制定为半天自动复制一次数据库。

后端最终会采用微服务架构。

中间层提供的结构函数在迭代三会全部进行优化。

### 可用性设计战术

使用Prometheus+Grafana进行后端监控。

对后端处理时间过长的请求返回时间超时，避免长时间占用后端资源。

若发生故障，可以依靠备份数据库进行数据回滚，避免脏数据的影响。

使用token识别前端的请求，防止被不明软件长时间调用接口。