<CloudHomework>

软件架构文档

版本 <1.1>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 08/11/2020 | 1.0 | 整理逻辑、部署视图，添加用例、进程视图 | 吴方诺 |
| 08/11/2020 | 1.1 | 添加实现视图、数据视图 | 刘金珂 |
|  |  |  |  |
|  |  |  |  |

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 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 6

*3.2.1* *中间件层* 6

*3.2.2* *业务层* 6

4. 进程视图 8

4.1 概览 9

4.2 具体进程说明 9

4.2.1 HomeworkSystemAccess 9

4.2.2 HomeworkProcess 9

4.2.3 HomeworkController 9

4.2.4 StudentApplication 9

4.2.5 HomeworkCache和HomeworkOfferingCache 9

5. 部署视图 10

6. 实现视图 10

7. 数据视图（可选） 10

8. 核心算法设计（可选） 13

9. 质量属性 13

9.1 可用性 13

9.2 平均故障间隔 13

9.3 平均修复时间 13

9.4 最高错误或缺陷率 13

9.5 错误或缺陷率 13

9.6 对事务的响应时间 14

9.7 容量 14

9.8 吞吐量 14

9.9 资源利用情况 14

服务器端需要1G以上的磁盘空间，4G以上内存，采用ipv4协议进行互联网通信。 14

9.10 降级模式 14

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。

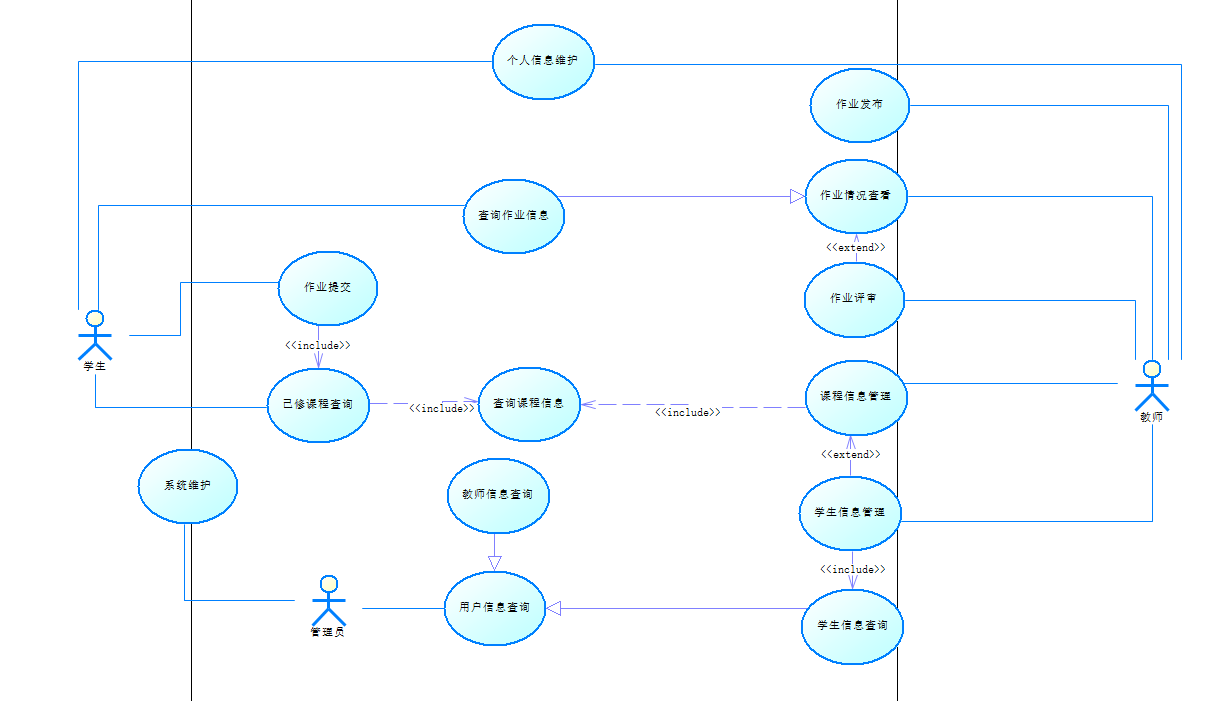
## 参考资料

1.沈备军 陈昊鹏《软件工程原理》第5章：面向对象的分析建模

2.Mikko Kontio (2008, July) Architectural manifesto: Designing software architectures, Part 5

3.Kruchten, Philippe (1995, November). Architectural Blueprints — The “4+1” View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50.

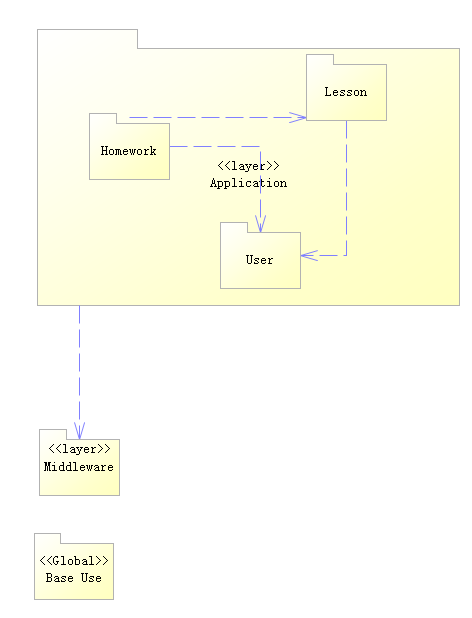
# 用例视图



# 逻辑视图

## 概述

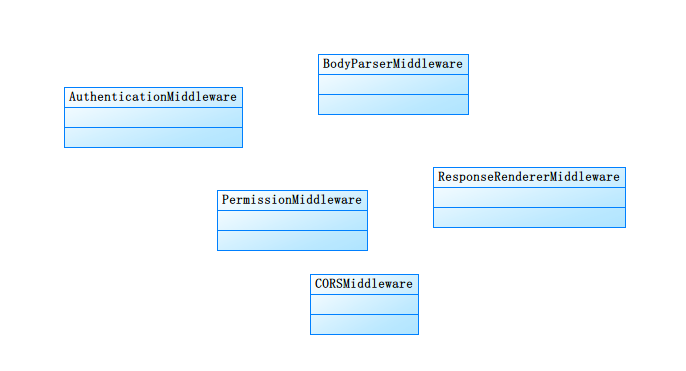
本节按照设计模型中包的层次结构来说明设计模型的整体分解情况。



我们系统架构逻辑视图概览如上图所示。在概览层我们主要采用分层架构设计，主要分为具体业务模块（Application）层以及中间件（Middleware）层。业务模块主要分为用户模块（User）、课程（Lesson）模块、作业（Homework）模块。

## 在构架方面具有重要意义的设计包

### *中间件层*

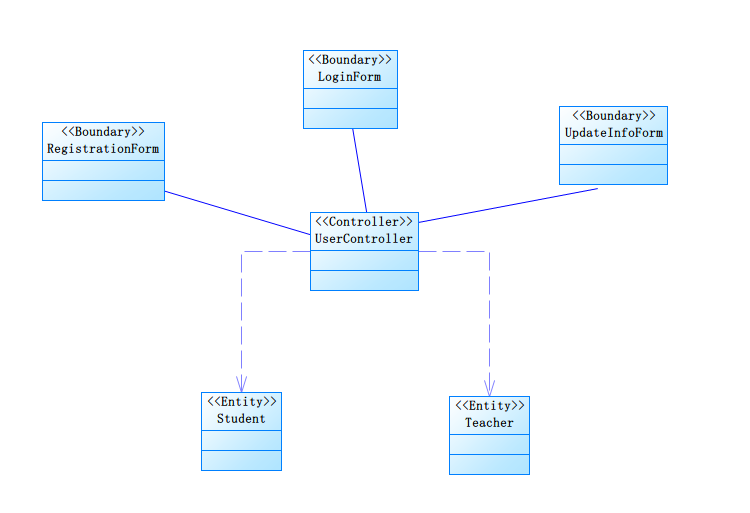


中间件层主要包含上述中间件，各自功能如下：

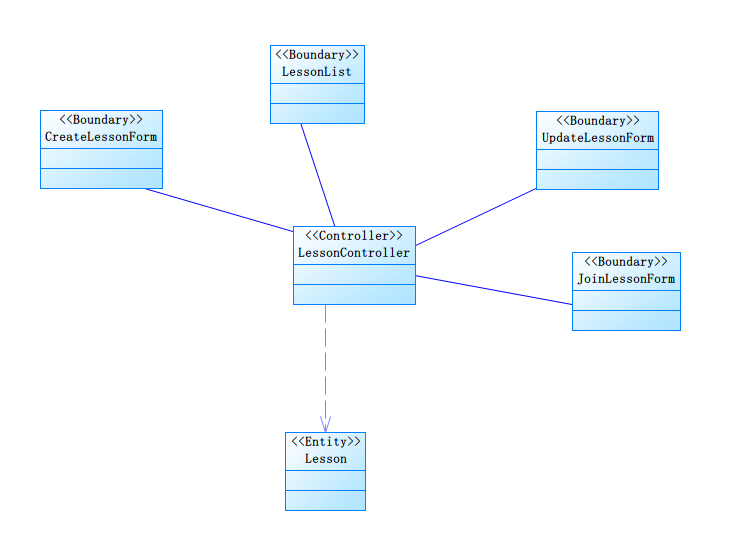
1. AuthenticationMiddleware：负责处理HTTP请求中的鉴权信息，并在HTTP请求对象上附加用户认证信息。
2. PermissionMiddleware：负责根据HTTP请求对象上的用户认证信息和后端API的权限声明，判断允许用户请求或拒绝用户请求。
3. BodyParserMiddleware：负责处理HTTP请求的包体数据，解析为便于后端处理的格式，如处理JSON/Multipart-form等。
4. ResponseRendererMiddleware：负责将后端返回的对象序列化为便于传输的形式，如JSON。
5. CORSMiddleware：负责对后端响应添加CORS头部，以符合浏览器的CORS要求。

### *业务层*

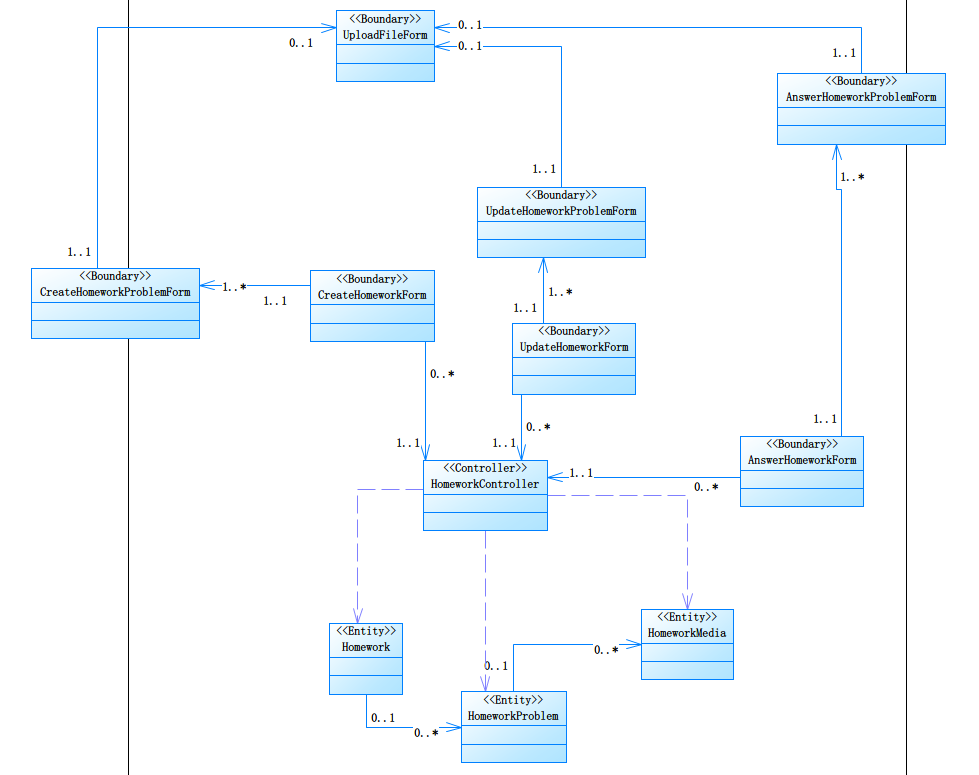
业务层采用MVC架构模式，大致架构如下：



上图为用户模块MVC架构图。主要包含用户登录、注册、修改个人信息功能。



上图为课程模块MVC架构图，主要包含列出课程、教师创建课程、修改课程、学生加入课程功能。

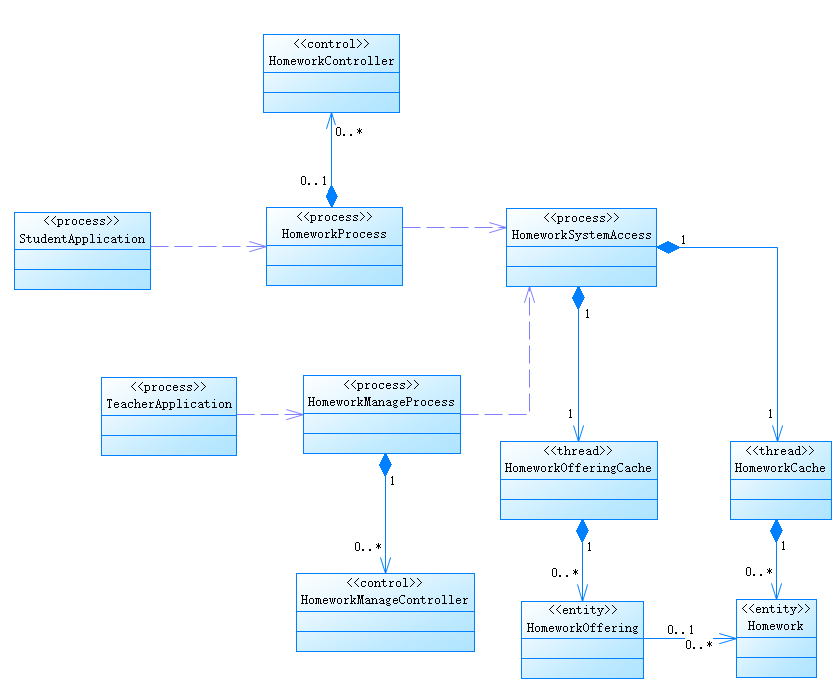


上图为作业模块MVC架构图，主要包含教师发布作业、修改作业、学生提交作业功能。其中所涉及到的Form类包含一个主要的Form与该Form中包含的子Form。另外，每一个Form中还包含UploadFileForm，用于向服务器上传题目内容或答案内容中所涉及到的图片等文件。

# 进程视图

本节描述系统的进程视图。说明在系统执行过程中涉及到的进程和线程。

## 概览



## 具体进程说明

以学生相关的进程为例，教师相关的相似。

### HomeworkSystemAccess

此进程提供了对系统的访问，多个用户可以同时提交作业答案、发布和修改题目。提供了一个作业答案和提供的作业的缓存。

### HomeworkProcess

这是这个进程的一个实例，可以让每个学生提交自己的作业。

### HomeworkController

支持了学生查看并且提交作业的用例。

### StudentApplication

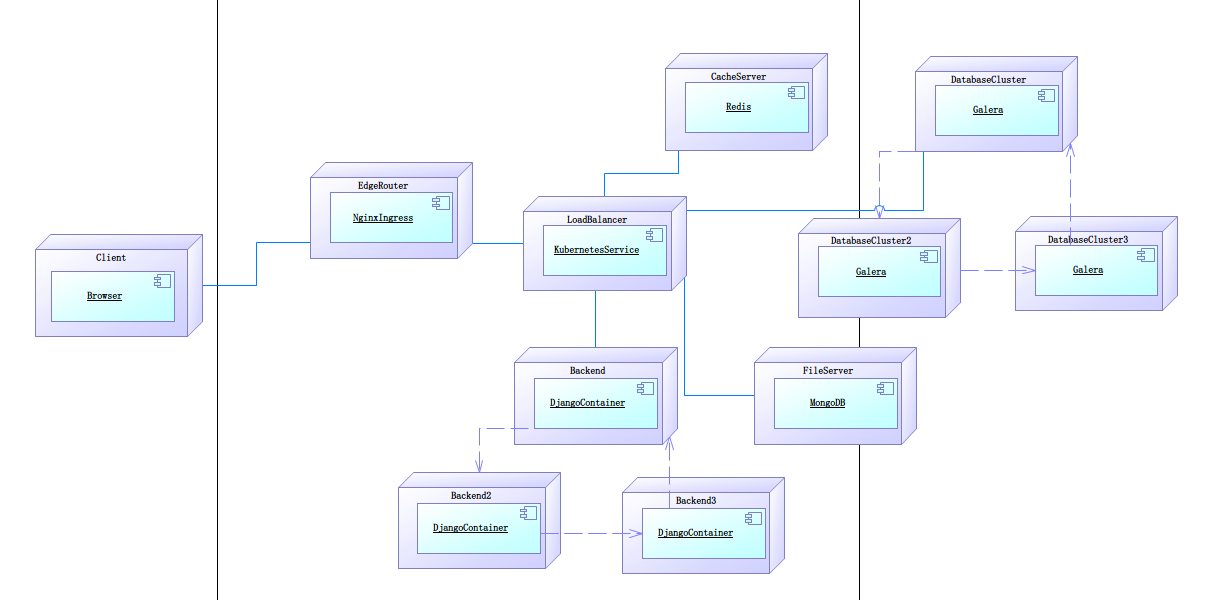
管理学生功能，包括用户界面处理和与业务流程的协调。对于当前正在查看或提交作业的每个学生，都有一个此进程的实例。

### HomeworkCache和HomeworkOfferingCache

Homework和HomeworkOffering缓存线程用于从作业系统异步检索项目。分别代表写好的作业和提供的作业框架。

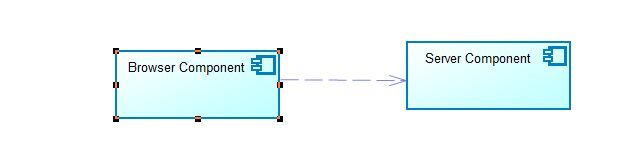
# 部署视图

[本节说明用来部署和运行该软件的一种或多种物理网络（硬件）配置。对于每种配置，它至少应该指出执行该软件的物理节点（计算机、CPU）及其互连情况（总线连接、LAN 连接、点到点连接等）。另外还要包括**进程视图**中的各进程到物理节点的映射。]

上上图为我们系统架构的部署视图。

系统整体采用B/S架构。我们计划基于Google Cloud Platform或Amazon Web Service搭建集群平台，在此基础上使用Kubernetes搭建容器平台。其中，用户使用浏览器与EdgeRouter进行交互。后者是集群与外界的边界路由器，使用Nginx-ingress实现，负责处理用户的请求并转发给集群内相应的容器集群。我们使用Kubernetes的Service机制来管理各个服务的容器集群，进行负载均衡。在后端服务（Backend）方面，我们使用Django框架编写后端，其是无状态的，因此便于横向扩展。我们使用MariaDB的Galera项目处理数据库集群和读写分离问题。另外，我们还使用Redis提供关键数据的缓存服务；使用MongoDB的GridFS功能提供文件存储服务。

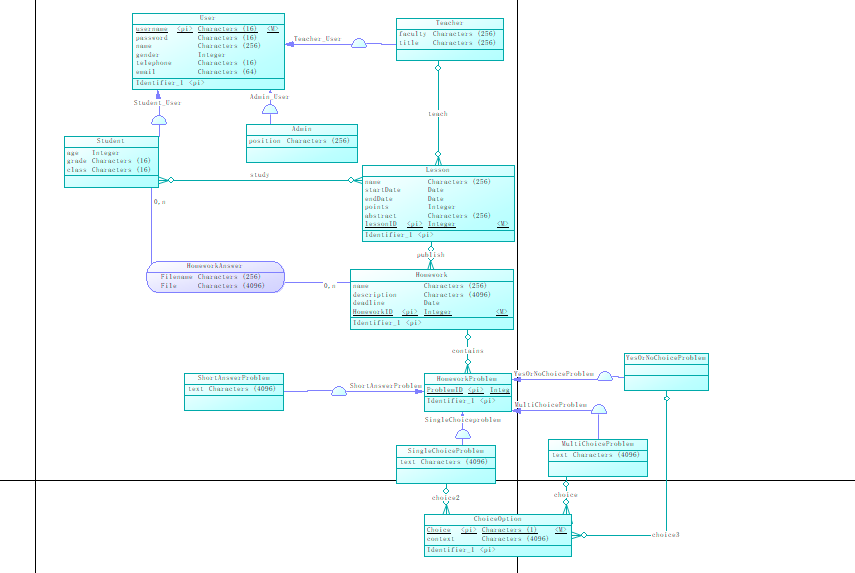
# 实现视图



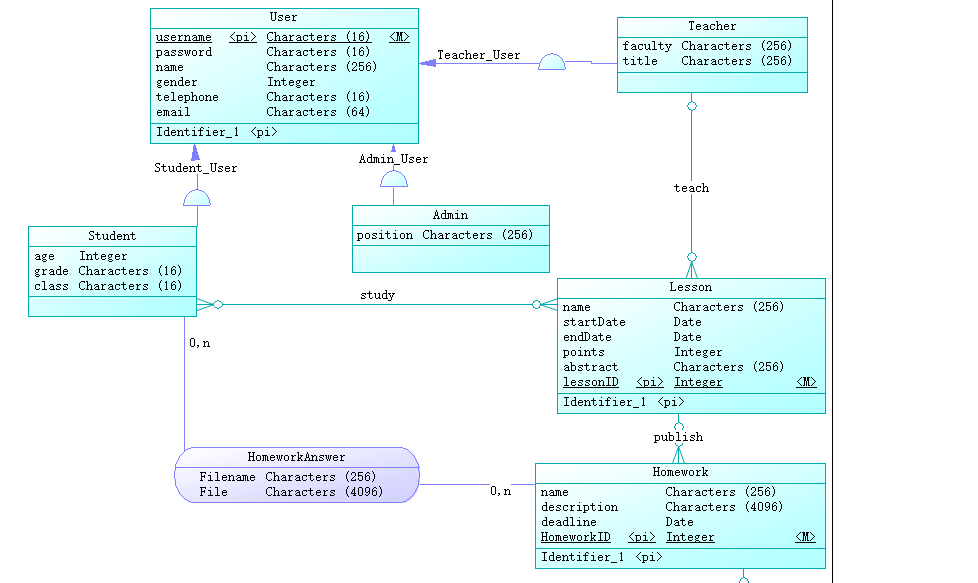
系统使用B/S架构，整体可以分为Browser Component和Server Component。

# 数据视图（可选）

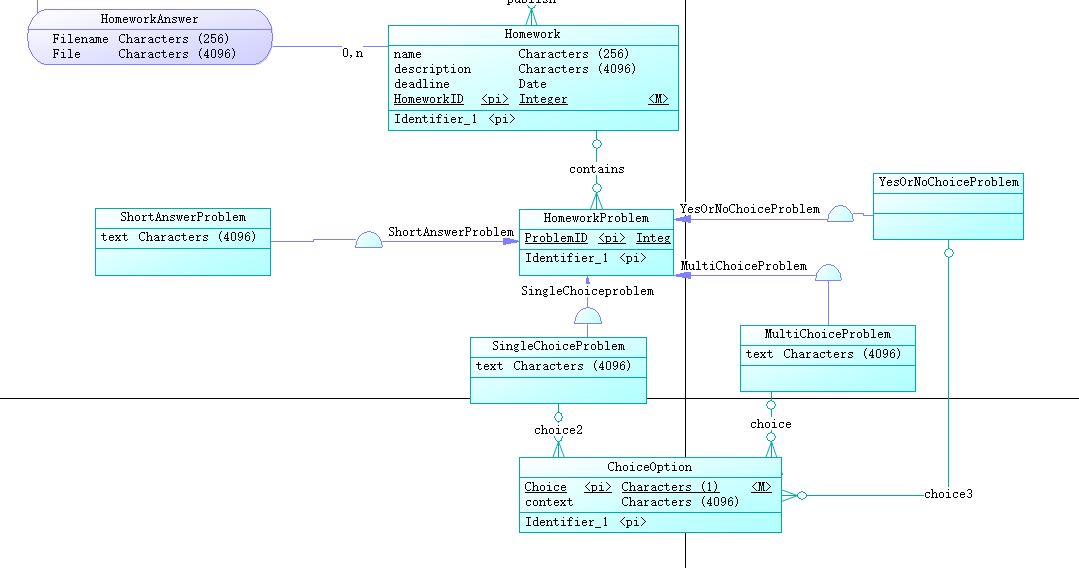
数据视图整体如下：



为保证清晰度，再截以下两张部分视图



图一：数据视图上半部分

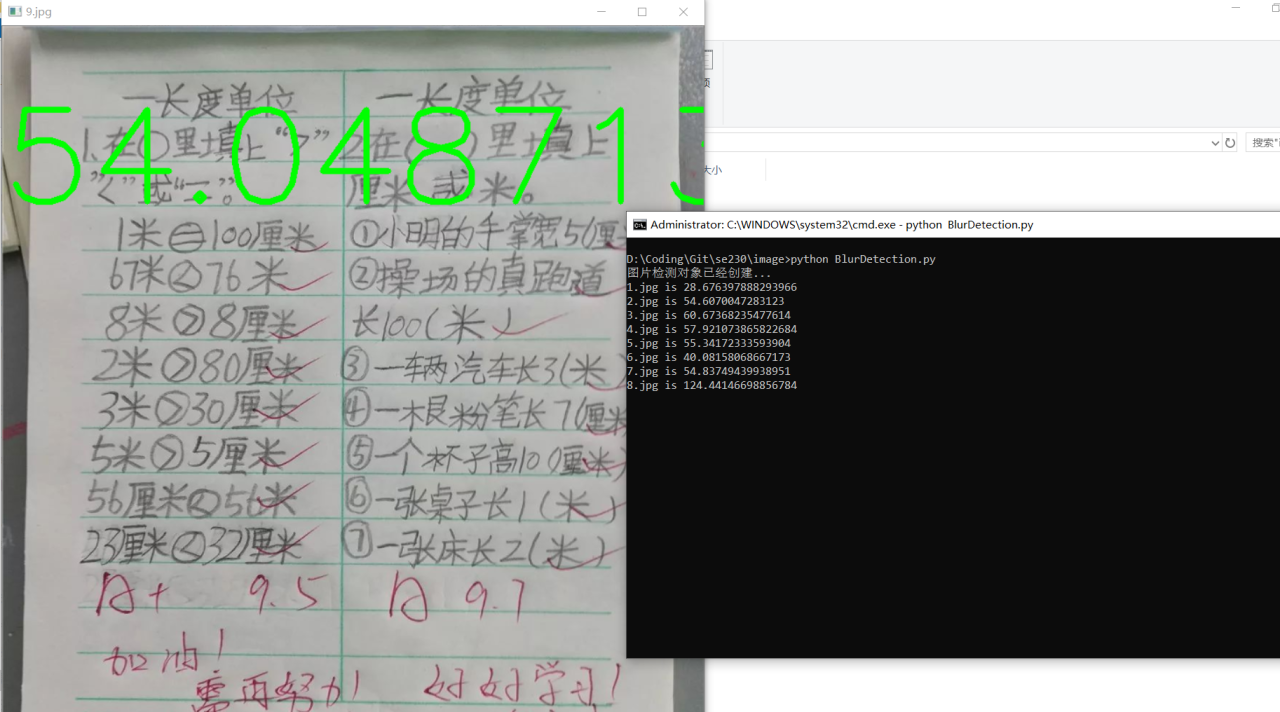


图二：数据视图下半部分

数据库的ER图如上图所示。

# 核心算法设计（可选）

为检测图像清晰度，我们使用图像清晰度算法SVD（奇异值分解）对图像清晰度进行评分，在python的numpy库中有svd函数可以直接调用，效果如下：



可以得到作业图片的清晰度评分，进而决定是否重新上传或者自动提升清晰度

# 质量属性

## 可用性

系统能够保证一周7天，一天24小时不间断运行，一年内系统的平均正常运行时间达到99.9%，系统能够正确处理发生的异常或者错误，并返回错误信息。当系统出错时，能自动报告错误并在用户允许时发送，便于开发人员修复。

## 平均故障间隔

平均故障间隔时间不小于1年。

## 平均修复时间

平均修复时间不大于24小时。

## 最高错误或缺陷率

最高错误率不超过10bugs/KLOC并且不超过5bugs/function-point。

## 错误或缺陷率

小错误率不超过5bugs/KLOC，大错误不超过3bugs/KLOC，严重错误率不超过1bugs/KLOC。严重错误指执行用户代码失败导致系统崩溃，或者服务器出现问题导致出现联机故障。

## 对事务的响应时间

用户向服务器发起联机请求后，服务器的平均响应时间不超过1.5秒，在1000并发下最长响应时间不超过3秒（不包括网络延迟）。

## 容量

系统应容纳10万名用户使用。

## 吞吐量

每秒能处理超过350个事务

## 资源利用情况

### 服务器端需要1G以上的磁盘空间，4G以上内存，采用ipv4协议进行互联网通信。

## 降级模式

在特殊时间段（例如学期初用户并发量巨大时，或是线上考试统一提交试卷时）关闭部分功能，从而满足高并发下的时间需求。