# 体系结构文档

**同城互助系统**

**项目报告**

目录

[1. 主要功能点和操作场景分析](#_Toc17872)

[1.1主要功能点](#_Toc26438)

[1.2操作场景分析](#_Toc13185)

[1.2.1用户管理](#_Toc5485)

[1.2.2学生功能](#_Toc21120)

[1.2.3管理员功能](#_Toc3765)

[2. 两种架构选择及对应的架构图设计](#_Toc27644)

[2.1 MVC架构](#_Toc6623)

[2.1.1模块视图](#_Toc3263)

[2.1.3组件-连接件视图](#_Toc5494)

[2.2 SOA架构](#_Toc4738)

[2.2.1模块视图](#_Toc25659)

[2.2.2组件-连接件视图](#_Toc10534)

[3. 非功能需求及ASR描述](#_Toc11791)

[3.1安全性](#_Toc24200)

[3.2可用性](#_Toc8430)

[3.3互操作性](#_Toc27135)

[3.4可修改性](#_Toc31476)

[3.5性能](#_Toc27850)

[3.6可维护性](#_Toc11670)

[4. 系统类图设计](#_Toc23290)

[5. 组件/连接件到实现类的映射](#_Toc11107)

[6. 两种架构的比较及最终选择](#_Toc30874)

[6.1整体比较](#_Toc15628)

[6.2基于系统实际的最终选择](#_Toc29986)

[7. 具体实现技术的选择与解释](#_Toc24461)

[8. 基于MVC架构的ADD过程](#_Toc16248)

[8.1第一次迭代结果](#_Toc31906)

[8.2第二次迭代过程](#_Toc16144)

[8.2.1识别所选模块的ASR](#_Toc6134)

[8.2.2每个ASR可选的设计决策](#_Toc17124)

[8.2.3设计决策的选择及分析](#_Toc7849)

[8.2.4第二次迭代结果](#_Toc15710)

[8.3第三次迭代过程](#_Toc13342)

[8.3.1识别所选模块的ASR](#_Toc23664)

[8.3.2每个ASR可选的设计决策](#_Toc28215)

[8.3.3设计决策的选择及分析](#_Toc18271)

[8.3.4第三次迭代结果](#_Toc17505)

[8.4第四次迭代之后](#_Toc22553)

[9.基于SOA架构的ADD过程](#_Toc20519)

[9.1第一次迭代结果](#_Toc5133)

[9.2第二次迭代过程](#_Toc13084)

[9.2.1识别所选模块的ASR](#_Toc2263)

[9.2.2每个ASR可选的设计决策](#_Toc21856)

[9.2.3设计决策的选择及分析](#_Toc4931)

[9.2.4第二次迭代结果](#_Toc2321)

[9.3第三次迭代过程](#_Toc2931)

[9.3.1识别所选模块的ASR](#_Toc16987)

[9.3.2每个ASR可选的设计决策](#_Toc26413)

[9.3.3设计决策的选择及分析](#_Toc20749)

[9.3.4第三次迭代结果](#_Toc11161)

[9.4第四次迭代之后](#_Toc12255)

[10.ATAM分析过程](#_Toc4604)

[10.1质量属性效用树](#_Toc24264)

[10.2ATAM分析](#_Toc7982)

[10.3敏感点和权衡点](#_Toc10511)

[10.4风险和非风险](#_Toc11230)

[11.挑战和经验](#_Toc32012)

[12.组员和分工](#_Toc3043)

# 1.1主要功能点

* 用户
  + 注册/登录
  + 个人信息管理
  + 发布任务
  + 查看任务列表
  + 领取任务
  + 任务结算与评价
* 系统
  + 智能推荐
* 管理员功能
  + 资质审核
  + 投诉反馈
  + 任务列表管理

## 1.2操作场景分析

### 1.2.1用户

表1 登录/注册

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 希望访问该系统的用户 |
| 刺激 | 用户希望访问该系统 |
| 制品 | 登录/注册子系统 |
| 环境 | 正常操作 |
| 响应 | 验证用户登录信息/注册用户 |
| 响应度量 | 响应时间不超过0.5秒。  用户的个人信息及密码应被有效保护，被攻击盗取的概率不大于0.01% |

表2 个人信息管理

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 希望修改个人信息的用户 |
| 刺激 | 用户希望修改个人信息 |
| 制品 | 个人信息修改子系统 |
| 环境 | 正常操作 |
| 响应 | 修改个人信息并保存 |
| 响应度量 | 响应时间不超过0.5秒。个人信息存储不应出错。 |

表3 发布任务

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 希望发布任务的用户 |
| 刺激 | 用户希望发布任务 |
| 制品 | 任务发布子系统 |
| 环境 | 正常操作 |
| 响应 | 发布任务信息并更新 |
| 响应度量 | 响应时间不超过0.5秒。任务信息存储不应出错。 |

表4 查看任务列表

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 用户 |
| 刺激 | 用户希望查看当前/个人的任务信息 |
| 制品 | 查看任务信息子系统 |
| 环境 | 正常操作 |
| 响应 | 显示当前/个人的任务信息 |
| 响应度量 | 响应时间不超过0.5秒。任务信息显示不应有误。 |

表5 领取任务

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 用户 |
| 刺激 | 用户希望领取该任务 |
| 制品 | 任务领取子系统 |
| 环境 | 正常操作 |
| 响应 | 若任务在规定时间内未被他人领取，则用户领取成功；否则显示领取失败。 |
| 响应度量 | 系统应该允许5000个用户同时进行正常的访问、领取操作 |

表6 任务结算与评价

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 用户 |
| 刺激 | 发布的任务已经被及时完成并反馈 |
| 制品 | 任务结算与评论子系统 |
| 环境 | 正常操作 |
| 响应 | 显示任务完成信息，结算评价后并保存 |
| 响应度量 | 响应时间不超过0.5秒。任务完成信息显示不得有误。 |

### 1.2.2系统

表7 智能推荐

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 系统 |
| 刺激 | 用户查看推荐任务界面 |
| 制品 | 任务智能推荐子系统 |
| 环境 | 正常操作 |
| 响应 | 若用户是老用户则根据用户选取任务的习惯推荐任务，若是新用户则推荐最新发布的任务 |
| 响应度量 | 响应时间不超过1秒。系统成功处理不同推荐情况的概率不小于95% |

### 1.2.3管理员功能

表8 资质审核

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 管理员 |
| 刺激 | 管理员接受到用户上传的证书 |
| 制品 | 资质审核系统 |
| 环境 | 正常操作 |
| 响应 | 记录用户的证书信息并保存 |
| 响应度量 | 响应时间不超过0.5秒 |

表9 投诉反馈

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 管理员 |
| 刺激 | 管理员接受到来自用户的投诉 |
| 制品 | 投诉反馈子系统 |
| 环境 | 正常操作 |
| 响应 | 显示投诉信息与证据，将管理员处理结果通知投诉人与被投诉人 |
| 响应度量 | 响应时间不超过0.5秒  投诉信息显示不得有误  允许投诉信件多达500封 |

表10 任务列表管理

|  |  |
| --- | --- |
| **场景要素** | **可能的值** |
| 刺激源 | 管理员 |
| 刺激 | 出现违规或者信息出错的任务 |
| 制品 | 任务管理子系统 |
| 环境 | 正常操作 |
| 响应 | 显示所有任务信息，汇成列表，并筛选出违规任务，进行删除或修改 |
| 响应度量 | 响应时间不超过0.5秒  系统至少能够存储500T的任务信息  系统成功筛选违规任务的成功率高达99% |

# 两种架构选择及对应的架构图设计

注：清晰的架构图设计详见附属的images文件夹。

## 2.1 MVC架构

### 2.1.1模块视图

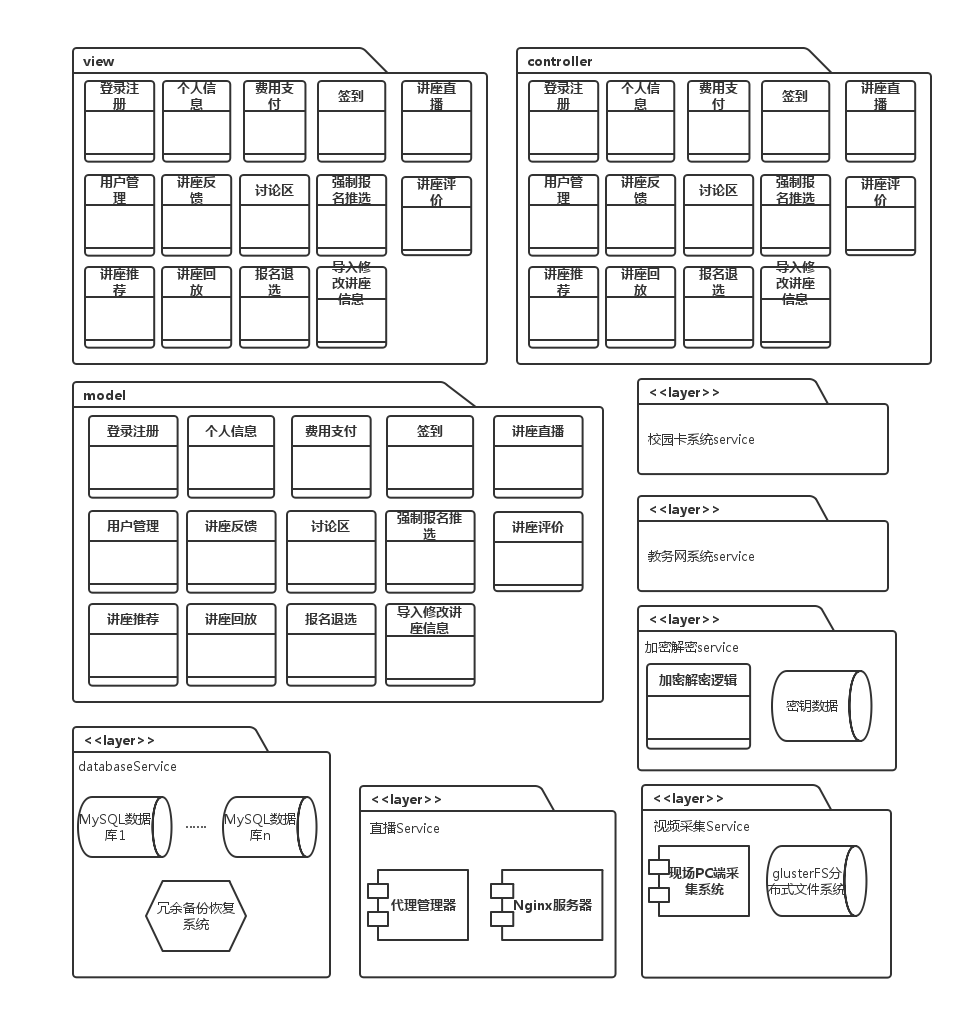


图1：MVC模块视图

### 2.1.3组件-连接件视图

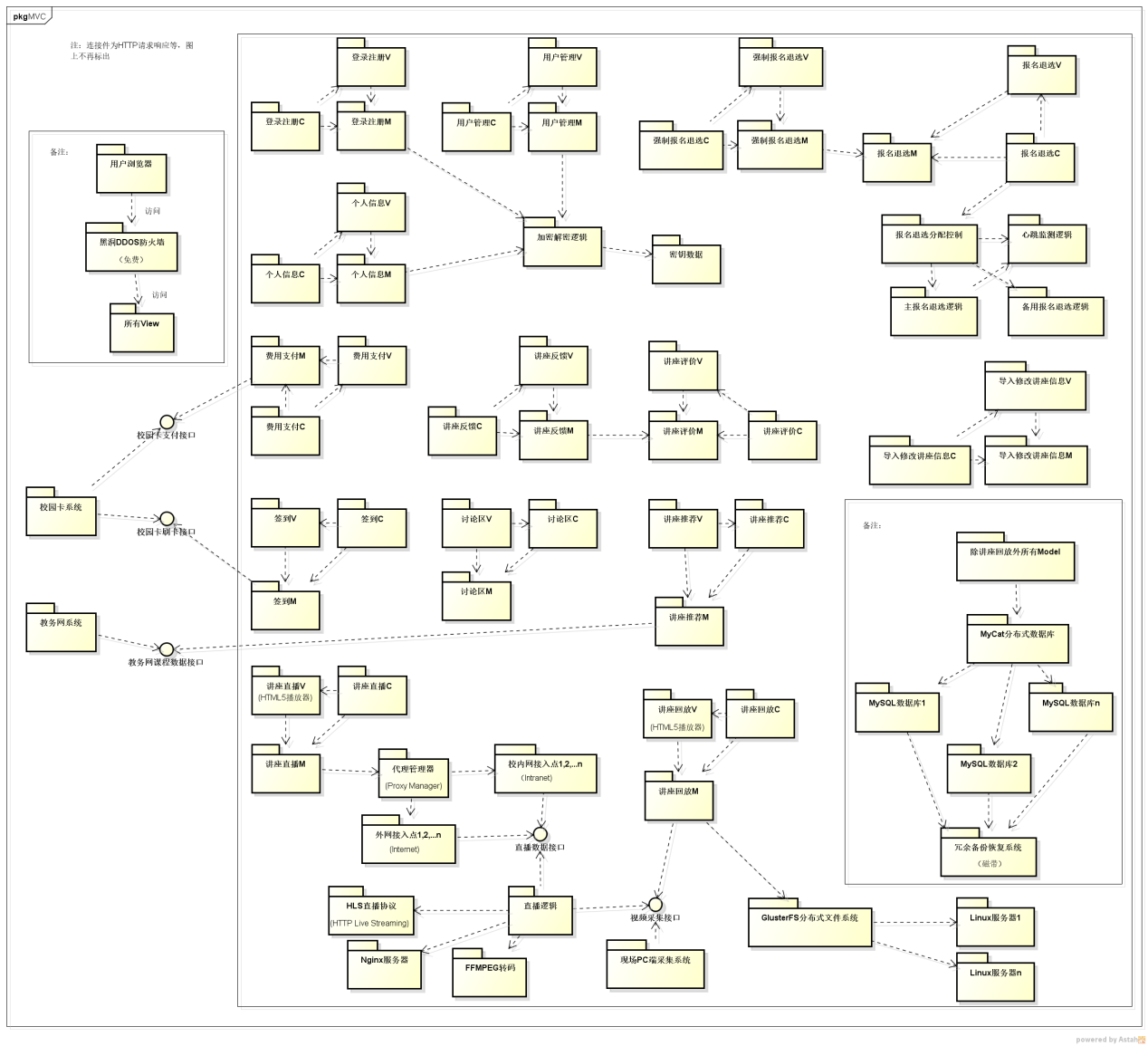


图2：MVC组件-连接器视图

## 2.2 SOA架构

### 2.2.1模块视图

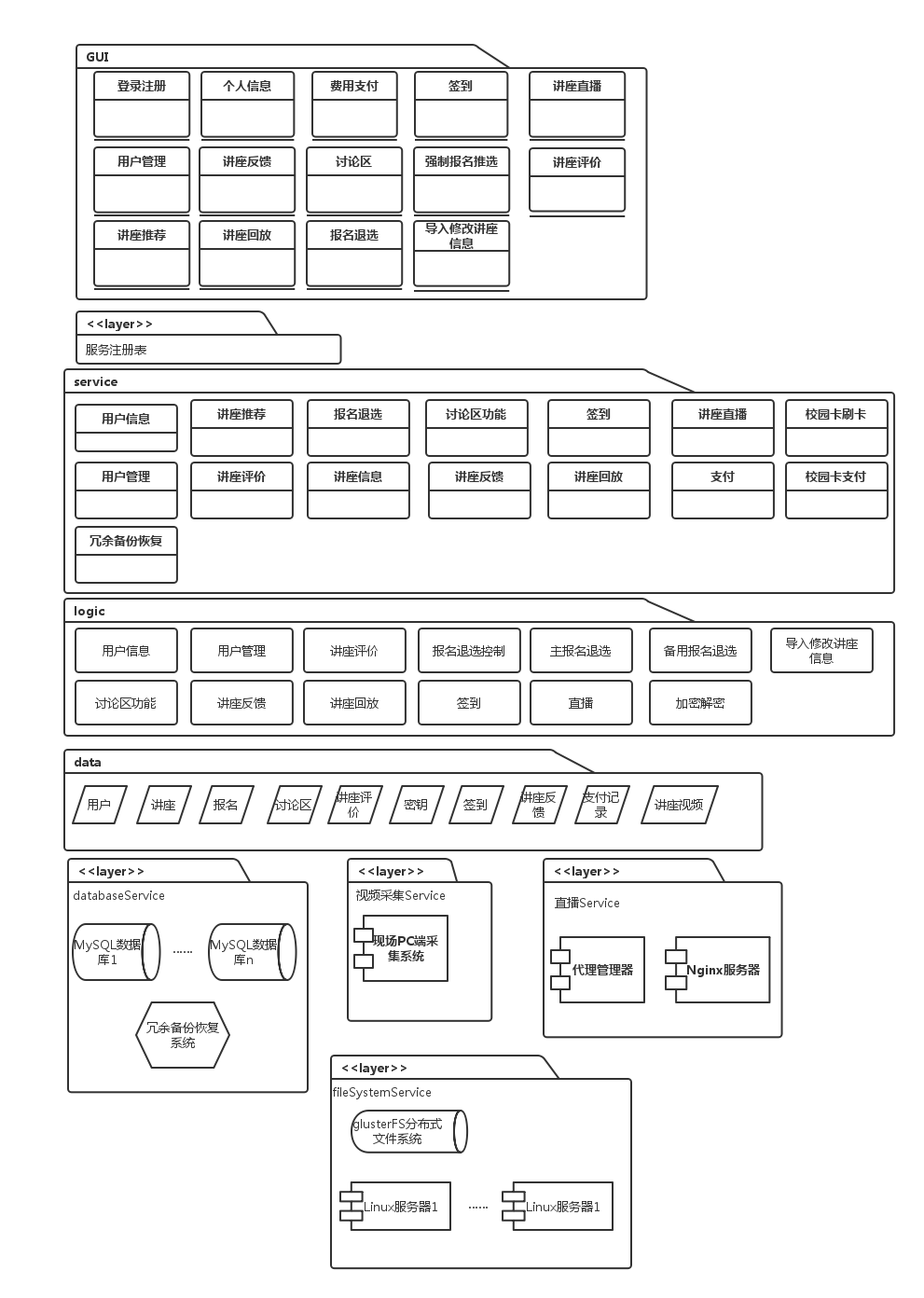


图3：SOA模块视图

### 2.2.2组件-连接件视图

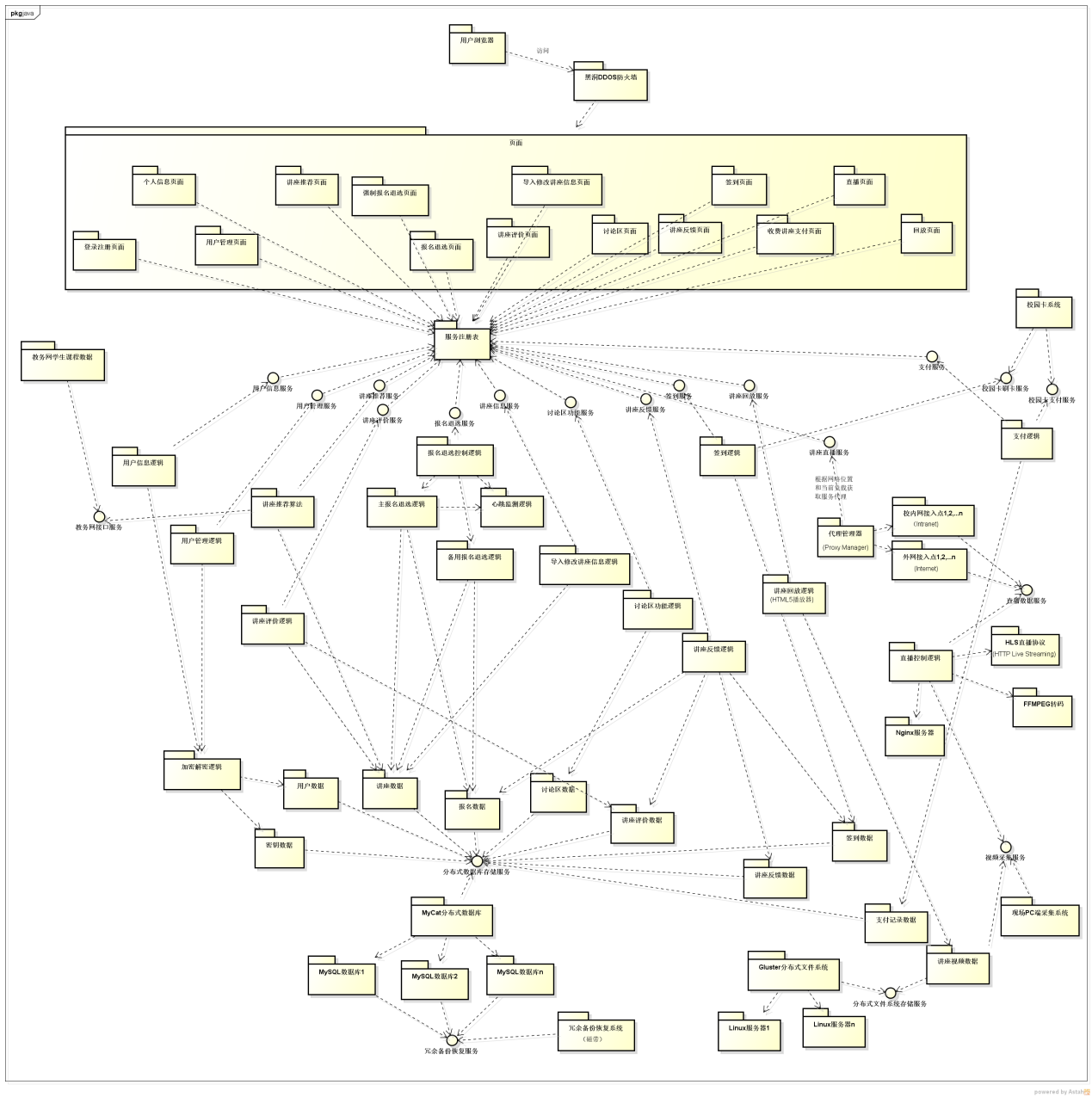


图4：SOA组件-连接器视图

# 非功能需求及ASR描述

## 3.1安全性

Scenario 1

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 来自内部/外部的经过了授权/未经过授权的个人或系统 |
| **刺激** | 试图修改/删除数据，访问系统服务，窃取用户信息 |
| **制品** | 系统服务、系统中的数据 |
| **环境** | 在线/离线，联网/断网，连接有防火墙或直接连接到了网络上 |
| **响应** | 对用户进行验证；加密用户的账户信息；阻止未授权用户访问； |
| **响应度量** | 未认证用户无法访问数据和发布控制指令  数据被恶意修改/删除后可以在10min内进行恢复 |

## 3.2可用性

Scenario 2

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 来自授权的个人用户 |
| **刺激** | 用户希望使用系统功能，尤其是讲座报名、直播等 |
| **制品** | 整个系统 |
| **环境** | 系统运行时，尤其是高峰期 |
| **响应** | 系统能够为用户提供服务而不能频繁崩溃 |
| **响应度量** | 系统崩溃频率不超过1次/周，并能在崩溃发生后1分钟之内记录错误日志并恢复正常功能。 |

## 3.3互操作性

Scenario 3

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 系统发起/收到与校园卡系统/教务网系统/现场PC端采集系统的请求/响应 |
| **刺激** | 希望与上述的系统进行请求/响应 数据交换 |
| **制品** | 希望进行互操作的上述系统 |
| **环境** | 希望进行互操作的上述系统正在运行 |
| **响应** | 请求被正确获取并成功交换数据 |
| **响应度量** | 交换信息数据成功的概率不低于99% |

## 3.4可修改性

Scenario 4

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 开发者 |
| **刺激** | 开发者希望修改系统用户界面、数据标准、控制逻辑等 |
| **制品** | 代码 |
| **环境** | 设计、开发、维护系统时 |
| **响应** | 需要修改的模块被正确的修改，并不影响其他功能的实现 |
| **响应度量** | 每个模块的修改可以在2人月内完成  修改预算不超过总预算的10%  不影响无关的系统功能 |

## 3.5性能

性能场景1：负载

Scenario 5

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 访问系统的学生个人用户 |
| **刺激** | 学生用户希望在系统上进行讲座报名操作/查看讲座直播 |
| **制品** | 系统报名参加讲座子系统/讲座直播子系统 |
| **环境** | 联网状态 |
| **响应** | 系统能够正常工作，为每一个用户提供响应 |
| **响应度量** | 系统应该允许5000个用户同时进行正常的访问、报名操作/观看讲座直播  用户访问所需页面的时间不超过0.1s |

性能场景2：容量

Scenario 6

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 系统 |
| **刺激** | 系统希望进行讲座视频的存储 |
| **制品** | 存储视频数据的系统数据库 |
| **环境** | 运行时 |
| **响应** | 系统正确、完整、及时的存储大量讲座视频数据 |
| **响应度量** | 系统至少能够存储50T的视频数据，保持数据的完整性、正确性、一致性 |

性能场景3：实时性

Scenario 7

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 访问系统的学生个人用户 |
| **刺激** | 学生用户希望在系统上观看讲座直播 |
| **制品** | 系统讲座直播子系统 |
| **环境** | 联网状态 |
| **响应** | 系统直播视频清晰流畅，延时短 |
| **响应度量** | 系统直播延时不超过5s |

## 3.6可维护性

Scenario 8

|  |  |
| --- | --- |
| **场景的部分** | **可能的值** |
| **源** | 系统开发者 |
| **刺激** | 系统开发者希望对系统进行维护，修改其部分功能 |
| **制品** | 系统各个功能模块 |
| **环境** | 系统维护时 |
| **响应** | 及时、方便的进行修改，并对无关模块不造成影响 |
| **响应度量** | 每个模块的修改时间不超过5人日 |

# 系统类图设计

注：清晰的系统类图设计详见附属的images文件夹。

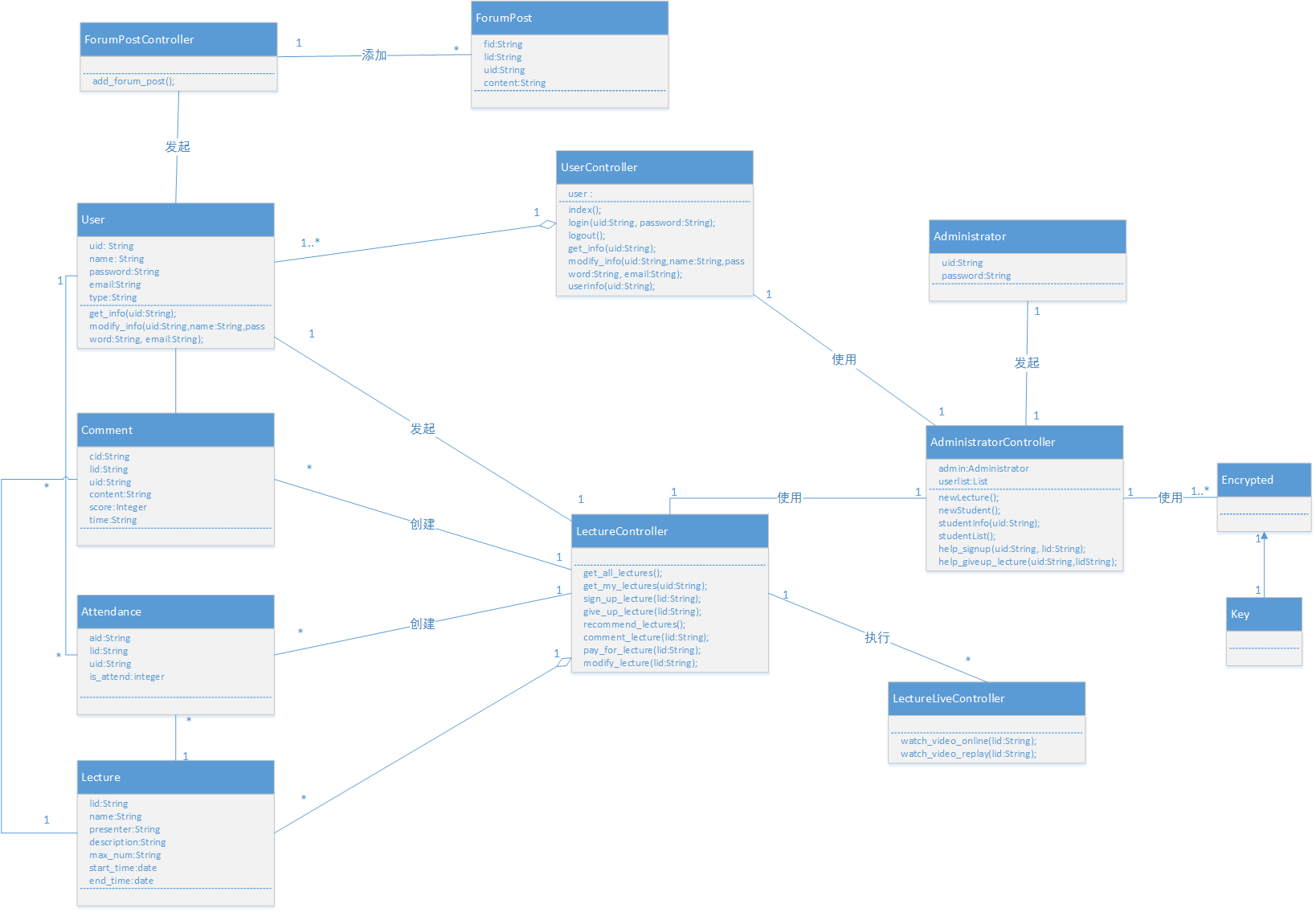


图5：系统类图

# 组件/连接件到实现类的映射

|  |  |
| --- | --- |
| 实现类 | 组件连接器 |
| UserController | 登陆注册C  个人信息C |
| User | 个人信息M |
| Administrator |  |
| AdministratorController | 用户管理C  强制报名退选C  导入修改讲座信息C |
| Lecture | 讲座直播M  讲座回放M |
| LectureController | 报名退选C  费用支付C  讲座反馈C  讲座评价C  讲座推荐C  讲座出席C |
| Encrypted | 加密解密 |
| Key | 密钥数据 |
| Comment | 讲座评价M |
| Attendance | 讲座出席M |
| LectureLiveController | 讲座直播C  讲座回放C |
| ForumPost | 讨论区M |
| ForumPostController | 讨论区C |

# 6.两种架构的比较及最终选择

## 6.1整体比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **管道过滤器** | | **MVC** | |
| **比较方面** | **pros** | **cons** | **pros** | **cons** |
| 安全性 |  | 管道过滤器架构的每个过滤器都可能被单独攻击 | MVC没有安全性方面的明显缺点 |  |
| 可用性 |  | 系统是线性连续的，一旦一个环节发生问题整个系统就会崩溃 | MVC没有可用性方面的明显缺点 |  |
| 互操作性 |  | 管道过滤器风格不适合处理处理交互应用，且每个过滤器都必须被看成完整的输入输出转换 | MVC没有互操作性方面的明显缺点 |  |
| 可修改性 | 管道过滤器架构耦合性低，添加和替换新的过滤器简单。 |  | MVC架构耦合性低，MVC的三个部件相对独立，尤其适用于View变更频繁的Web系统。 |  |
| 性能：负载 |  | 每个过滤器都有解析和合成数据的工作导致系统性能下降 | MVC没有负载性能方面的明显缺点 |  |
| 性能：容量 | 每个过滤器作为单独任务，所以支持并行执行 |  |  | 不支持并行执行 |
| 性能：实时性 | 两种架构没有特别大的差别 | | | |
| 可维护性 | 系统维护简单，旧的过滤器可以被改进过的替换掉 |  | 分离M、V、C也使得WEB应用更易于维护和修改。 |  |
| 可扩展性 | 管道过滤架构器的可扩展性很好，新的过滤器可以添加到系统中来 |  |  | MVC的可扩展性较弱 |
| 成本约束 |  | 数据传输没有通用标准，编写过滤器复杂，成本较高 | 容易掌握，开发快速 |  |

## 6.2基于系统实际的最终选择

系统最终选择MVC架构，这一决策是考虑项目前景和约束而做出的。

本系统是面向大众的、单独的Web系统，本身不是某个应用服务体系的一部分，并不是像“某某投资银行的企业应用体系中的下单系统”这样的应用，未来也没有为其开发配套系统的预期，考虑这种情况，管道过滤器风格良好的易修改性和易扩展性没有显示出来，而管道过滤器风格的安全性和稳定性低、开发周期长、成本高的缺点却是显著的。与之相对的，MVC的简单、快速、高效成为其最大的优势。

依照上述原因，我们最终选择的是MVC架构。

# 7具体实现技术的选择与解释

7.1设计决策1

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR1 个人信息管理  DC1 用户初次使用应用时，需要对用户进行短暂的操作示例教程  DC2 系统采用移动端（ iOS 版本和 Android 版本）开发 | |
| 候选方案 | CS1.1 | 个人信息设置在本机APP上，通过登陆验证后显示在信息管理页面，可自由修改 |
| CS1.2 | 通过第三方api将信息与微信或QQ或手机号绑定，通过第三方验证后方可查看并自由修改。 |
| 选择选项 | CS1.2 | |
| 选择理由 | 加入第三方绑定可以增强数据安全性，同时便于第一次给用户引导教程。 | |
| 影响 | 开发视图 | |
| 详细设计约束 | 1.需要增加第三方数据接口以及相应的数据交换实现方法  2.采用适配器模式，将相应数据转换格式后进行交互。 | |
|  | | |

7.2设计决策2

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR2 发布任务  DC3 在1000人并发使用时，系统不能崩溃 | |
| 候选方案 | CS2.1 | 为正在发布中的任务以HTTP请求来传递消息，需要手动刷新或等待系统自动更新来触发信息同步。 |
| CS2.2 | 通过创建会话来传递数据，保证实时数据通讯与同步。 |
| CS2.3 | 仅对部分内容使用长连接（任务被接取的消息提醒），其余部分保留为基础HTTP请求。 |
| 选择选项 | CS2.3 | |
| 选择理由 | 保证效率的同时能够提高并发性能，减轻服务器压力 | |
| 影响 | 逻辑视图、开发视图 | |
| 详细设计约束 | 1.采用webSocket双向通信 | |
|  | | |

7.3设计决策3

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR3 领取任务  DC4 在进行数据的下载和上传中，如果网络出现故障，系统不能出现故障 | |
| 候选方案 | CS3.1 | 在用户界面设置限制，当系统欠响应时短时间内不允许连续领取任务 |
| CS3.2 | 通过后台增加限制，同域请求同一时间只受理一个请求。 |
| 选择选项 | CS3.2 | |
| 选择理由 | 1.保证系统故障率下降，尽管可能因为网络问题丢包，但一定程度上可以抵御恶意攻击 | |
| 影响 | 进程视图、逻辑视图 | |
| 详细设计约束 | 1.增加过滤器，拦截未验证他域请求  2.给respose方法增加逻辑约束，使得每次至多只有一个同域请求占据系统性能 | |
|  | | |

7.4设计决策4

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR4 任务结算与评价  DC5 任务完成后，双方需确认完成，提交反馈评价 | |
| 候选方案 | CS4.1 | 领取任务者点击确认完成任务后，可以对任务进行评价；然后发布任务者可以点击确认完成任务，并评价。双方可以向系统管理员投诉恶意评价 |
| 选择选项 | CS4.1 | |
| 选择理由 | 1.领取任务者点击确认完成任务相当于通知发布任务者已有任务完成需要再确认和评价，而不用让发布者自己注意有无任务已完成，使用更便捷。 | |
| 影响 | 开发视图 | |
| 详细设计约束 | 1.需要增加第三方数据接口以及相应的数据交换实现方法 | |
|  | | |

7.5设计决策5

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR5 资质审核  DC9 要求用户上传身份证照片和专业技能资格证明（证书拍照），并填写对应的专业资格技能明晨，由管理员审核 | |
| 候选方案 | CS5.1 | 设置专门的审核方法模块与数据模块进行交接 |
| 选择选项 | CS5.1 | |
| 选择理由 | 1. 技能证明不一定只是证书照片，也可以是电子书信等形式，所以应该提供 多种链接 2. 增加数据安全性 3. 保证审核策略可灵活变更，具备可扩展性 | |
| 影响 | 逻辑视图，开发视图 | |
| 详细设计约束 | 1.需要增加第三方数据接口以及相应的数据交换实现方法 | |
|  | | |

7.6设计决策6

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR6 投诉处理  DC7 投诉处理结果要反馈给投诉人和被投诉人 | |
| 候选方案 | CS6.1 | 开发一个反馈模块，每次处理自动调用该模块 |
| 选择选项 | CS6.1 | |
| 选择理由 | 成本低，保障了系统效率 | |
| 影响 | 逻辑视图，进程视图 | |
| 详细设计约束 | 1.使用隐式调用的方法  2.保证审查策略可灵活变更，具备可扩展性 | |
|  | | |

7.7设计决策7

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR7 任务列表管理  DC8 使用网络爬虫技术，每天自动从网上更新任务信息 | |
| 候选方案 | CS7.1 | 为了满足DC13,可以使用适配器模式对爬虫爬取的不同格式数据进行处理 |
| CS7.2 | 可以将爬虫封装成一个独立模块，里面有网页提取器（可以自己定制网页解析规则），爬虫任务生产器（生产URL）等，可以自己定制爬取策略和配置文件，爬虫策略由解析配置文件决定 |
| 选择选项 | CS7.2结合CS7.1 | |
| 选择理由 | CS7.2能够保证实现对数据的规范化处理  CS7.1可以实现爬取策略内容的灵活可变更 | |
| 影响 | 逻辑视图、开发视图、进程视图、部署视图 | |
| 详细设计约束 | 1.需要按一定规则写配置文件（例如xml），配置文件描述爬取策略的接口  2.适配器中定制数据统一规则  3.模块内部爬虫技术使用策略模式 | |
|  | | |

7.8设计决策8

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR8 智能推荐  DC9 推荐结果根据用户习惯得出 | |
| 候选方案 | CS8.1 | 对任务添加多个属性描述其特性 |
| CS8.2 | 建立过滤器类，利用策略模式实现 |
| 选择选项 | CS8.2结合CS8.1 | |
| 选择理由 | 1.和设计决策1的Sorter类似，可以为设计提供经验  2.增加新的屏蔽策略不必修改以前的代码，具有更好的灵活性和可扩展性 | |
| 影响 | 逻辑视图、开发视图 | |
| 详细设计约束 | 1..封装屏蔽规则，使用策略模式 | |
|  | | |

7.9设计决策9

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR9 可靠性  DC10 存储出现故障要能够快速恢复，恢复应在半个小时内完成  DC11 采取RAID机制进行数据备份  DC12 故障频率每月不能超过3次  DC13 服务器出现故障时要能够及时发现并通知系统管理员，并且不能影响网站访问  AR11 容量  DC16 最大要能够存储1千万个任务的相关信息  DC17 系统数据分布式存储在多台设备上  AR12 并发性  DC18 在高峰期时间（例如节假日），要允许5百万用户同时在线使用 | |
| 候选方案 | CS9.1 | 使用分布式数据存储，对数据进行热备份，可以提供数据查询存储效率，并且一个物理节点故障，系统可以使用备用节点。可以通过其他数据库进行数据恢复，不会影响操作。 |
| CS9.2 | 为数据恢复和数据备份建立独立的模块 |
| CS9.3 | 对数据库进行一个备份，故障时人工切换备份 |
| 选择选项 | CS9.1结合CS9.2 | |
| 选择理由 | CS9.1可以一举多得的解决对于系统可靠性、并发性、容量的需求，可以显著提高数据库性能  CS9.3需要产生额外的人工费用，且容错度很低 | |
| 影响 | 逻辑视图、开发视图、进程视图、部署视图 | |
| 详细设计约束 | 1. 要建立数据访问的分配规则，并且提供统一的访问接口 2. 使用分布式数据存储，屏蔽物理节点差异 3. 使用多个备用节点，保证数据可恢复 4. dataBackup和dataRevovery内部实现策略模式 | |
|  | | |

7.10设计决策10

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR12 并发性  DC18 在高峰期时间（例如节假日），要允许5百万用户同时在线使用  AR13 安全性  DC19 实现短时间内频繁访问的IP（即插件扫描攻击）并屏蔽  DC20 发现特定IP地址（例如搜索引擎Agent）的访问并屏蔽 | |
| 候选方案 | CS10.1 | 采取分布式存储数据的方法，将数据保存在多台服务器，提高系统并发性能 |
| CS10.2 | 实现负载均衡，合理安排访问请求，保证服务器资源能够有效利用，提高访问效率 |
| CS10.3 | 由于负载均衡系统需要将请求分派，所以会随时得到各个服务器的状态，则依据此使用Message通信机制进行故障检测。 |
| 选择选项 | CS10.1结合CS10.2，CS10.3 | |
| 选择理由 | 1.能够提供服务器访问性能，多台服务器可以避免服务器故障的影响。  2.分布式处理能够协调对任务的处理  3.增加服务器等方式来应对高峰吞吐量。 | |
| 影响 | 逻辑视图、开发视图、进程视图、部署视图 | |
| 详细设计约束 | 1. 封装负载均衡，屏蔽实现细节 2. 分布式存储数据 3. 多台服务器 | |
|  | | |

7.11设计决策11

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR9 可靠性  DC10 存储出现故障要能够快速恢复，恢复应在半个小时内完成  DC11 采取RAID机制进行数据备份  DC12 故障频率每月不能超过3次  DC13 服务器出现故障时要能够及时发现并通知系统管理员，并且不能影响网站访问 | |
| 候选方案 | CS11.1 | 使用ping/echo验证服务器状态，进行故障检测 |
| CS11.2 | 使用心跳机制，即各个服务器定期发送心跳，进行故障检测 |
| CS11.3 | 由于负载均衡系统需要将请求分派，所以会随时得到各个服务器的状态，则依据此使用Message通信机制进行故障检测。 |
| CS11.4 | 开发一个Monnitor模块，通过message检测ServerStatus |
| 选择选项 | CS11.1结合CS11.2,CS11.3 | |
| 选择理由 | 这样选择可以帮助ServerControl中负载均衡的处理，CS11.1，CS11.2方式成本低，且能够迅速检测服务器故障 | |
| 影响 | 逻辑视图，开发视图 | |
| 详细设计约束 | 1. 服务器端要实现健壮的代码，多使用错误处理机制 2. 建立日志模块，可以查询服务器端错误 3. 建立一个服务器管理来实现ping/echo,心跳机制 | |
|  | | |

7.12设计决策12

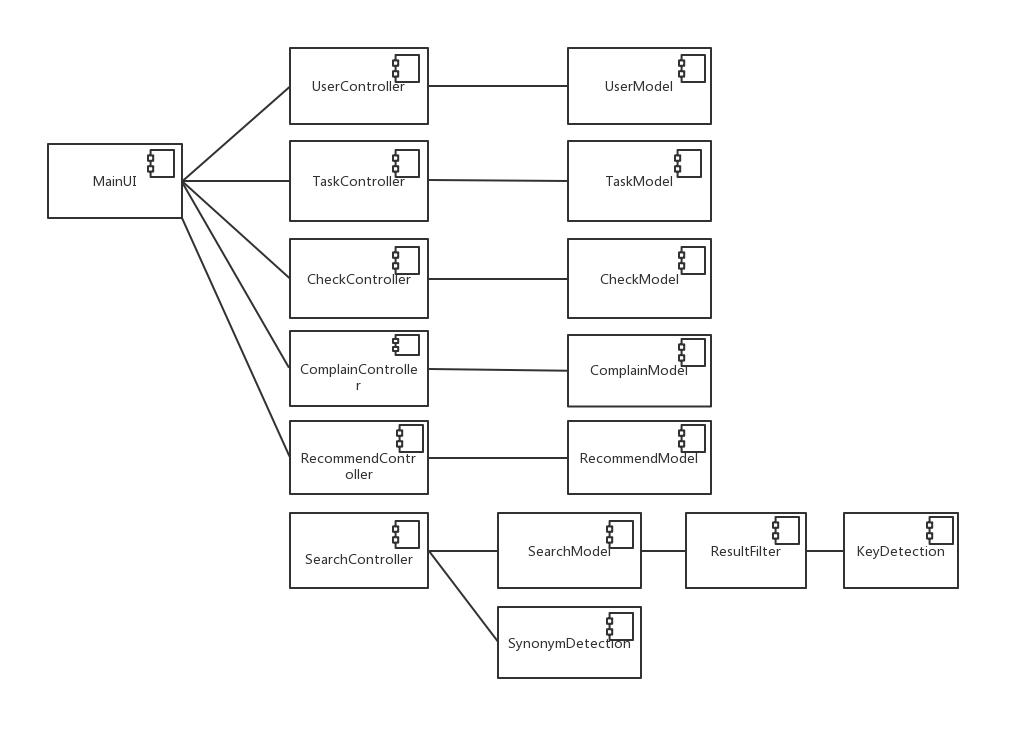
|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR13 安全性  DC19 实现短时间内频繁访问的IP（即插件扫描攻击）并屏蔽  DC20 发现特定IP地址（例如搜索引擎Agent）的访问并屏蔽 | |
|  | CS12.1 | 实现短时间内频繁访问的IP（即插件扫描攻击）并屏蔽 |
| CS12.2 | 实现一个联系人列表，用于侦测异常后通知 |
| CS12.3 | 实现一个防火墙模块，对于攻击 过滤和防御 |
| 选择选项 | CS12.1结合CS12.2 CS12.3 | |
| 选择理由 | 1.封装成独立模块具有可扩展性  2.使用隐式调用的方式通知相关人员 | |
| 影响 | 逻辑视图、开发视图、进程视图、部署视图 | |
| 详细设计约束 | 1.使用隐式调用的方式通知相关人员  2.将攻击防御和攻击检测分离  3.模块内部使用策略模式 | |
|  | | |

7.13设计决策13

|  |  |  |
| --- | --- | --- |
| 需求&约束 | AR10 兼容性  DC14 兼容各种浏览器，如firefox, chrome, internet explorer | |
| 候选方案 | CS13.1 | 采用适配器模式对于不同的格式进行处理 |
| CS13.2 | 使用bootstrap之类的框架实现，调用固有的api |
| 选择选项 | CS13.2结合CS13.1 | |
| 选择理由 | 利用bootstrap框架能实现大部分兼容。对于不兼容的部分自己实现兼容性 | |
| 影响 | 逻辑视图、开发视图 | |
| 详细设计约束 | 1使用bootstrap等前端框架实现浏览器兼容  2.对于不兼容部分自己实现兼容性 | |
| @JF$YMX6Z]~FDCD_VQ)ANPX | | |

# 8 基于MVC架构的ADD过程

8.1 第一次迭代结果



MVC第一次迭代结果

8.2 第二次迭代过程

选取用户模块作为系统元素进行分解。在质量属性方面，该模块与互操作性、可修改性、负载性能、安全性有密切关系。

8.2.1 识别所选模块的ASR

|  |  |  |  |
| --- | --- | --- | --- |
| # | Architectural Drivers | Importance | Difficulty |
| 1 | Scenario 1：互操作性：系统发起现场PC端采集系统的请求被正确获取并成功交换数据 | high | medium |
| 2 | Scenario 2：可修改性：系统用户种类未来可能新增 | medium | low |
| 3 | Scenario 3：负载性能：系统能够允许500个用户同时在线发布或领取任务 | high | high |
| 4 | Scenario 4：安全性：系统要阻止非法用户操作系统 | high | high |

8.2.2 每个ASR可选的设计决策

* 互操作性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 在提供的服务列表中定位所需要的服务 | 主要针对所需的服务较多且有层次性的情况 |
| 使用控制机制来管理、协调服务的调用 | 针对所要服务的访问需求量较多的情况 |
| 抽取并剪裁接口 | 针对请求响应交换的数据不一致情况 |

* 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 划分模块 | 可以有效缩小要修改的范围 |
| 抽象通用服务 | 将多中用户都需要的服务提取出来，形成可供多种用户共同使用的通用服务 |
| 添加接口 | 将功能和实现分离，隔离需要更改的模块 |

* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 维持计算的多个副本 | 需要考虑如何使副本保持一致和同步 |
| 增加可用资源 | 使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本 |
| 区分校内校外网络接入 | 可以充分利用高速廉价的内网资源，但是增加了系统的复杂性 |

* 安全性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 用户模块操作前进行身份验证 | 虽然每一步操作都要进行身份验证增加了系统开销，但是可以防止飞防用户访问系统，造成破坏 |
| 使用防火墙 | 对不合理访问IP拒绝其访问 |

8.2.3 设计决策的选择和分析

* 互操作性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 在提供的服务列表中定位所需要的服务 | 不采用。仅是关于用户的服务。 |
| 使用控制机制来管理、协调服务的调用 | 不采用。仅是关于用户的服务。 |
| 抽取并剪裁接口 | 采用。用于统一不同的数据格式。 |

* 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 划分模块 | 采用。是一种常用且有效的策略。 |
| 抽象通用服务 | 不采用。并没有通用服务需要被抽象。 |
| 添加接口 | 采用。增加可修改性常用的策略。 |

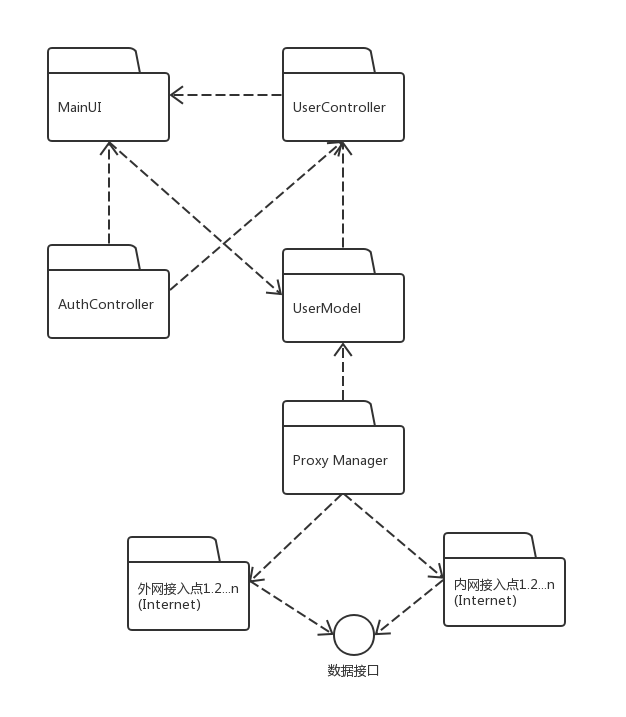
* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 维持计算的多个副本 | 不采用。考虑同步需要的代价太大。 |
| 增加可用资源 | 不采用。会增加较多成本。 |
| 区分校内校外网络接入 | 采用。校园内网速率远远优于因特网，需要充分利用其资源。 |

* 安全性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 用户模块操作前进行身份验证 | 采用。身份验证可以有效的避免非法用户损害系统安全 |
| 使用防火墙 | 不采用。该技术不成熟且代价较大，并且准确度不高 |

8.2.4 第二次迭代结果



MVC第二次迭代结果

8.3 第三次迭代过程

选取任务模块作为系统元素进行分解。在质量属性方面，该模块与可获得性、可修改性、负载性能有密切关系。

### 8.3.1识别所选模块的ASR

|  |  |  |  |
| --- | --- | --- | --- |
| # | Architectural Drivers | Importance | Difficulty |
| 1 | Scenario 5：可获得性：任务在99%的时间可以正常访问 | high | medium |
| 2 | Scenario 6：可修改性：修改任务的同时不影响其他模块的工作 | medium | low |
| 3 | Scenario 7：负载性能：系统能够允许500个用户同时进行正常的访问、有关任务的操作 | high | high |

### 8.3.2每个ASR可选的设计决策

* 可获得性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 命令/响应 | 一个组件发出一个命令，并希望在预定义的时间内收到一个来自审查组件的响应。与对所有进程发出命令的远程错误探测器相比，这种策略所使用的通信带宽更少 |
| 心跳 | 一个组件定期发出一个心跳信息，另一个组件收听该信息。可以在起到心跳的同时传递数据。 |
| 主动冗余（热启动） | 所有的冗余组件都以并行的方式对事件作出响应。错误发生时，使用该战术的系统的停机时间通常是几毫秒，因为备份是最新的，所以恢复所需的时间就是切换时间。 |
| 被动冗余 | 一个组件（主要的）对事件作出响应，并通知其他组件（备用的）必须进行的状态更新。该战术依赖于能够可靠地接管工作的备用组件。停机时间通常为几秒钟。 |
| 备件 | 备用件是计算平台配置用于更换各种不同的故障组件。当出现故障时，必须将其重新启动为适当的软件配置，并对其状态进行初始化。该战术的停机时间通常为几分钟。 |

* 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 划分模块 | 可以有效缩小要修改的范围 |
| 抽象通用服务 | 将多个模块都需要的服务提取出来，形成可供多个模块共同使用的通用服务 |
| 添加接口 | 将功能和实现分离，隔离需要更改的模块 |
| 运行时注册 | 支持即插即用操作，但需要管理注册的额外开销 |

* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 维持计算的多个副本 | 需要考虑如何使副本保持一致和同步 |
| 增加可用资源 | 使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本 |

### 8.3.3设计决策的选择及分析

* 可获得性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 命令/响应 | 不采用。 |
| 心跳 | 采用。常用的方式，且使系统的可获得性更高。 |
| 主动冗余（热启动） | 采用。该系统对于任务的可获得性要求很高，采用热备份可以获得更高的可获得性。 |
| 被动冗余 | 不采用。停机时间过长，对于任务这种短时间内对可获得性要求极高的子系统而言并不够好。 |
| 备件 | 不采用。理由同上。 |

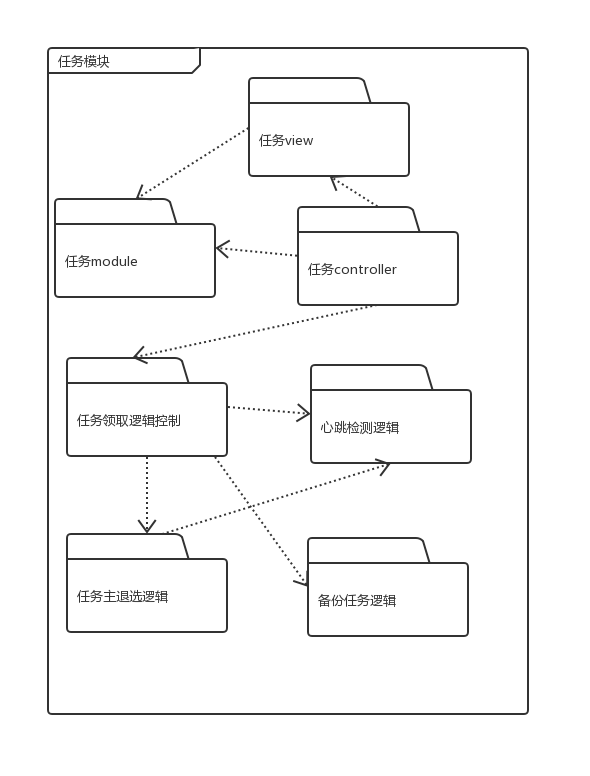
* 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 划分模块 | 采用。是一种常用且有效的策略。 |
| 抽象通用服务 | 不采用。并没有通用服务需要被抽象。 |
| 添加接口 | 不采用。不适用于该子系统。 |
| 运行时注册 | 不采用。会带来额外的管理开销。 |

* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 维持计算的多个副本 | 采用。保证在一个计算逻辑失效后可以在较短的时间内启用备用逻辑。 |
| 增加可用资源 | 不采用。会增加较多成本。 |

8.3.4 第三次迭代结果



MVC第三次迭代结果

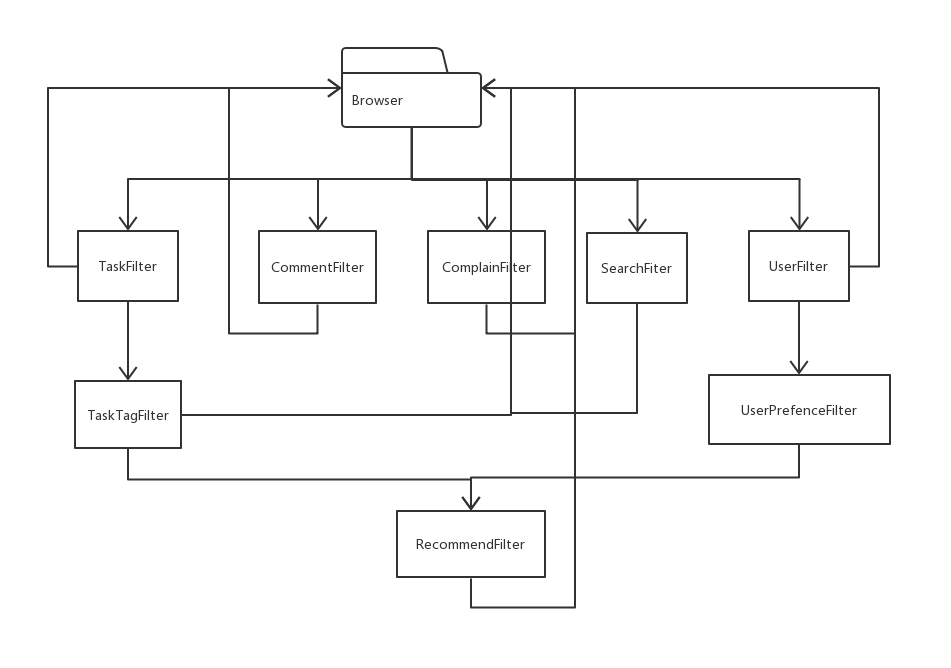
8.4 第四次迭代之后

受篇幅所限，其余模块相对上面两个模块简单，第四次迭代直至完成架构的过程此处不再给出。

# 9.基于管道过滤架构的ADD过程

9.1 第一次迭代结果

采用管道过滤的架构，参考所需的功能需求，形成第一次迭代后的整体设计图。由于该系统的功能性需求整体较为清晰，因此在第一次迭代结束后，已经将大部分功能性需求纳入架构设计考虑。

****

管道过滤第一次迭代结果

9.2 第二次迭代过程

选取用户模块作为系统元素进行分解。在质量属性方面，该模块与互操作性、可修改性、负载性能、安全性有密切关系。

9.2.1 识别所选模块的ASR

|  |  |  |  |
| --- | --- | --- | --- |
| # | Architectural Drivers | Importance | Difficulty |
| 1 | Scenario 1：互操作性：系统发起现场PC端采集系统的请求被正确获取并成功交换数据 | high | medium |
| 2 | Scenario 2：可修改性：系统用户种类未来可能新增 | medium | low |
| 3 | Scenario 3：负载性能：系统能够允许500个用户同时在线发布或领取任务 | high | high |
| 4 | Scenario 4：安全性：系统要阻止非法用户操作系统 | high | high |

9.2.2 每个ASR可选的设计决策

* 互操作性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 在提供的服务列表中定位所需要的服务 | 主要针对所需的服务较多且有层次性的情况 |
| 使用控制机制来管理、协调服务的调用 | 针对所要服务的访问需求量较多的情况 |
| 抽取并剪裁接口 | 针对请求响应交换的数据不一致情况 |

* 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 划分模块 | 可以有效缩小要修改的范围 |
| 抽象通用服务 | 将多中用户都需要的服务提取出来，形成可供多种用户共同使用的通用服务 |
| 添加接口 | 将功能和实现分离，隔离需要更改的模块 |

* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 维持计算的多个副本 | 需要考虑如何使副本保持一致和同步 |
| 增加可用资源 | 使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本 |
| 区分校内校外网络接入 | 可以充分利用高速廉价的内网资源，但是增加了系统的复杂性 |

* 安全性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 用户模块操作前进行身份验证 | 虽然每一步操作都要进行身份验证增加了系统开销，但是可以防止飞防用户访问系统，造成破坏 |
| 使用防火墙 | 对不合理访问IP拒绝其访问 |

9.2.3 设计决策的选择和分析

* 互操作性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 在提供的服务列表中定位所需要的服务 | 不采用。仅是关于用户的服务。 |
| 使用控制机制来管理、协调服务的调用 | 不采用。仅是关于用户的服务。 |
| 抽取并剪裁接口 | 采用。用于统一不同的数据格式。 |

* 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 划分模块 | 采用。是一种常用且有效的策略。 |
| 抽象通用服务 | 不采用。并没有通用服务需要被抽象。 |
| 添加接口 | 采用。增加可修改性常用的策略。 |

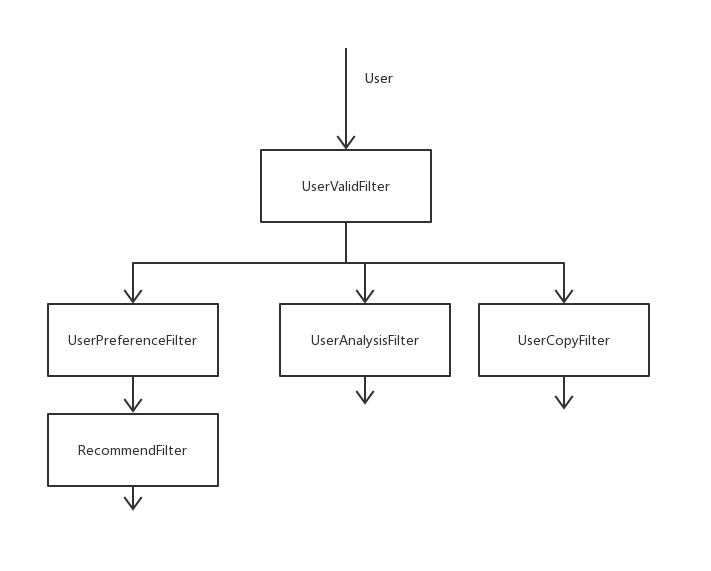
* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 维持计算的多个副本 | 不采用。考虑同步需要的代价太大。 |
| 增加可用资源 | 不采用。会增加较多成本。 |
| 区分校内校外网络接入 | 采用。校园内网速率远远优于因特网，需要充分利用其资源。 |

* 安全性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 用户模块操作前进行身份验证 | 采用。身份验证可以有效的避免非法用户损害系统安全 |
| 使用防火墙 | 不采用。该技术不成熟且代价较大，并且准确度不高 |

9.2.4 第二次迭代结果



管道过滤器第二次迭代结果

9.3 第三次迭代过程

选取任务模块作为系统元素进行分解。在质量属性方面，该模块与可获得性、可修改性、负载性能有密切关系。

### 9.3.1识别所选模块的ASR

|  |  |  |  |
| --- | --- | --- | --- |
| # | Architectural Drivers | Importance | Difficulty |
| 1 | Scenario 5：可获得性：任务在99%的时间可以正常访问 | high | medium |
| 2 | Scenario 6：可修改性：修改任务的同时不影响其他模块的工作 | medium | low |
| 3 | Scenario 7：负载性能：系统能够允许500个用户同时进行正常的访问、有关任务的操作 | high | high |

### 9.3.2每个ASR可选的设计决策

* 可获得性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 命令/响应 | 一个组件发出一个命令，并希望在预定义的时间内收到一个来自审查组件的响应。与对所有进程发出命令的远程错误探测器相比，这种策略所使用的通信带宽更少 |
| 心跳 | 一个组件定期发出一个心跳信息，另一个组件收听该信息。可以在起到心跳的同时传递数据。 |
| 主动冗余（热启动） | 所有的冗余组件都以并行的方式对事件作出响应。错误发生时，使用该战术的系统的停机时间通常是几毫秒，因为备份是最新的，所以恢复所需的时间就是切换时间。 |
| 被动冗余 | 一个组件（主要的）对事件作出响应，并通知其他组件（备用的）必须进行的状态更新。该战术依赖于能够可靠地接管工作的备用组件。停机时间通常为几秒钟。 |
| 备件 | 备用件是计算平台配置用于更换各种不同的故障组件。当出现故障时，必须将其重新启动为适当的软件配置，并对其状态进行初始化。该战术的停机时间通常为几分钟。 |

* 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 划分模块 | 可以有效缩小要修改的范围 |
| 抽象通用服务 | 将多个模块都需要的服务提取出来，形成可供多个模块共同使用的通用服务 |
| 添加接口 | 将功能和实现分离，隔离需要更改的模块 |
| 运行时注册 | 支持即插即用操作，但需要管理注册的额外开销 |

* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 维持计算的多个副本 | 需要考虑如何使副本保持一致和同步 |
| 增加可用资源 | 使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本 |

### 9.3.3设计决策的选择及分析

* 可获得性

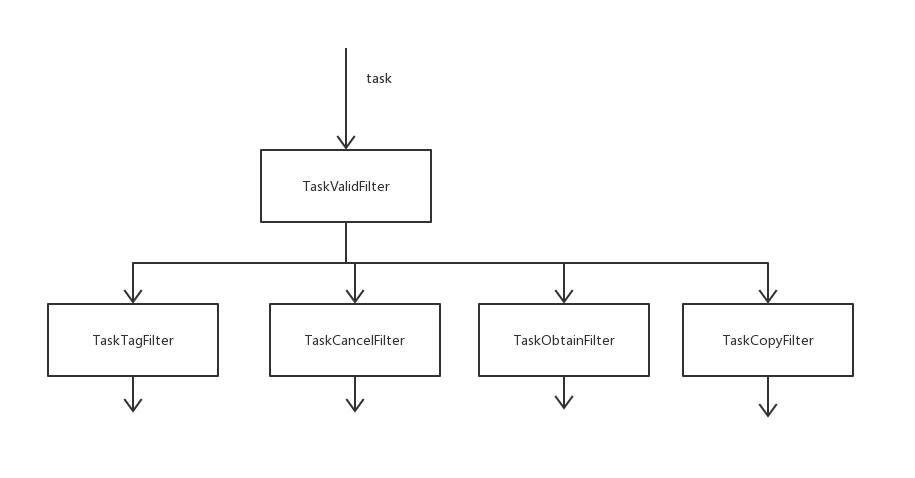
|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 命令/响应 | 不采用。 |
| 心跳 | 采用。常用的方式，且使系统的可获得性更高。 |
| 主动冗余（热启动） | 采用。该系统对于任务的可获得性要求很高，采用热备份可以获得更高的可获得性。 |
| 被动冗余 | 不采用。停机时间过长，对于任务这种短时间内对可获得性要求极高的子系统而言并不够好。 |
| 备件 | 不采用。理由同上。 |

* 可修改性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 划分模块 | 采用。是一种常用且有效的策略。 |
| 抽象通用服务 | 不采用。并没有通用服务需要被抽象。 |
| 添加接口 | 不采用。不适用于该子系统。 |
| 运行时注册 | 不采用。会带来额外的管理开销。 |

* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 维持计算的多个副本 | 采用。保证在一个计算逻辑失效后可以在较短的时间内启用备用逻辑。 |
| 增加可用资源 | 不采用。会增加较多成本。 |

****

管道过滤第三次迭代结果

9.3.4 第四次迭代结果、、

受篇幅所限，第四次迭代直至完成架构的过程此处不再给出。

# 10.ATAM分析过程

10.1 质量属性效用树

| 质量属性 | 属性求精 | 场景 |
| --- | --- | --- |
| 安全性 | 数据加密 | A1：用户数据被窃取，窃取者无法破译加密后的用户密码等敏感信息(H, H) |
| A2：定期更换加密密钥，当窃取者破解之前的加密密钥时，密钥已更换，窃取者仍然无法破译加密后的用户密码等敏感信息(M, M) |
| 身份验证 | A3：某用户未通过身份验证试图向探测器发送控制指令和访问数据，被系统拒绝 |
| 攻击侦测和防御 | A4：系统受到恶意攻击，系统侦测到攻击后，记录攻击日志，提醒管理员并锁住数据，直到确认安全(M, L) |
| 可用性 | 故障监测 | A5：采用命令/响应，一个组件发出一个命令，并在0.5s内收到一个来自审查组件的响应(M, H) |
| A6：采用心跳机制，系统某部分发生故障失去心跳后，系统在5min内检测到故障源(H, H) |
| 功能备份 | A7：当检测到故障后，系统在1s内切换至备用控制逻辑，恢复正常功能(H, H) |
| 故障重启 | A8：其他部分发生故障，若不能恢复，在5min内通过重启恢复正常功能(H, M) |
| 可修改性 | 接口修改 | A10：当现有接口发生修改时，可以在1人周内完成(M, M) |
| 功能修改 | A11：当现在功能发生修改时，可以在10人月内完成(M, M) |
| 性能 | 数据存储延迟 | A12：数据存储时间不可超过400ms(H, H) |
|  | 数据检索延迟 | A13：用户检索数据时，延迟时间不可超过1s(H, H) |
| 可维护性 |  | A14：系统开发者希望修改加密密钥时，修改时间不超过1人日(M, L) |
|  |  |  |

## 10.2ATAM分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **场景：A1** | 经过授权的个人用户希望进入系统报名参与讲座，但系统无法提供正确的反馈，系统记录错误日志并在5s之内从错误中恢复 | | | |
| **质量属性** | 可用性 | | | |
| **环境** | 联网状态 | | | |
| **刺激** | 希望进入系统报名参与讲座时系统无法提供正确的反馈 | | | |
| **响应** | 系统记录错误日志并从错误中恢复 | | | |
| **架构决策** | **敏感点** | **权衡点** | **风险** | **非风险** |
| 心跳 | S4 | T6 | R2 |  |
| 主动冗余 | S5 | T2 | R3 |  |
| **理由说明** | 系统出错后恢复的速度会极大影响用户体验，心跳可在2s内检测到故障，主动冗余可以保证在出现故障后，系统的停机时间只有几毫秒。虽然会降低部分性能，但提高了可用性，也是值得的。 | | | |
| **相关架构图** |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **场景：A2** | 系统受到恶意攻击，系统侦测到攻击后，防火墙会识别攻击并阻断攻击 | | | |
| **质量属性** | 安全性 | | | |
| **环境** | 运行时 | | | |
| **刺激** | 外部攻击 | | | |
| **响应** | 系统侦测到攻击后，防火墙识别攻击并阻断攻击 | | | |
| **架构决策** | **敏感点** | **权衡点** | **风险** | **非风险** |
| 在系统服务收到访问时验证用户身份，授权给用户应有的权限 | S2 |  |  | **N2** |
| 自动攻击侦测 | S3 | T1 | R1 |  |
| 使用DDOS分防火墙 | S7 |  |  | N3 |
| **理由说明** | 本项目与学生的校园卡相关，并于教务网相关联，所以安全性尤为重要，所以对用户进行权限设置。  攻击者若不能通过用户认证，可能使用各种方式对系统发动攻击。为安全起见，自动侦测攻击并使用防火墙阻止攻击。虽然可能降低部分性能，但是这种风险是可以接受的，而泄密的风险是不可接受的。 | | | |
| **相关架构图** |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **场景：A3** | 系统发起/收到与校园卡系统/教务网系统/现场PC端采集系统的请求/响应，希望与其进行请求/响应数据交换 | | | |
| **质量属性** | 互操作性 | | | |
| **环境** | 运行时 | | | |
| **刺激** | 系统希望与外界系统进行请求/响应数据交换 | | | |
| **响应** | 请求被成功获取并成功交换数据 | | | |
| **架构决策** | **敏感点** | **权衡点** | **风险** | **非风险** |
| 抽取并裁剪接口 | S13 |  |  | N6 |
| **理由说明** | 本项目拥有校园卡支付，校园卡签到，教务网信息导入等功能，所以互操作性是很重要的质量属性。采用合适的接口，可以方便与外界进行数据交换，提高互操作性。 | | | |
| **相关架构图** |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **场景：A4** | 开发者修改系统用户界面、数据标准、控制逻辑等 | | | |
| **质量属性** | 可修改性 | | | |
| **环境** | 设计、开发系统时 | | | |
| **刺激** | 开发者希望修改系统用户界面、数据标准、控制逻辑等 | | | |
| **响应** | 需要修改的模块被正确修改，并不影响其他功能的实现 | | | |
| **架构决策** | **敏感点** | **权衡点** | **风险** | **非风险** |
| 划分模块 | S7 |  |  | N4 |
| 独立存储密钥数据的模块 | S11 |  |  | N5 |
| **理由说明** | 对于一个复杂的系统来说，系统的可修改性和可维护性都是非常重要的。采用模块划分的决策，增强内聚，可以有效缩小要修改的范围，是成熟的设计原则，可增强可修改性。并独立存储密钥数据的模块，可提高可维护性 | | | |
| **相关架构图** |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **场景：A5** | 有500名用户同时在系统上进行讲座报名操作/查看讲座直播 | | | |
| **质量属性** | 性能（负载） | | | |
| **环境** | 联网状态 | | | |
| **刺激** | 学生用户希望在系统上进行讲座报名操作/查看讲座直播 | | | |
| **响应** | 系统能够正常工作，为每一个用户提供相应 | | | |
| **架构决策** | **敏感点** | **权衡点** | **风险** | **非风险** |
| 增加可用资源 | S8 |  | R4 |  |
| 使用分布式数据库服务器 | S9 | T3 | R5 |  |
| **理由说明** | 使用速度更快的处理器、额外的处理资源、额外的内存、更快的网络，可以保证在负载方面提高系统性能。但这会使得成本偏高，造成风险。因此需要在可以负担的范围内尽可能增加可用资源。分布式数据服务器的性价比显著优于单台商业中型机和大型机，能够提供较为优质的数据服务，提高性能。 | | | |
| **相关架构图** |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **场景：A6** | 用户在系统上观看讲座直播 | | | |
| **质量属性** | 性能 | | | |
| **环境** | 运行时 | | | |
| **刺激** | 学生用户希望在系统上观看讲座直播 | | | |
| **响应** | 系统直播视频清晰流畅，延时短 | | | |
| **架构决策** | **敏感点** | **权衡点** | **风险** | **非风险** |
| 使用HLS（ HTTP Live Streaming）协议 | S10 | T4 | R6 |  |
| 使用FFMPEG转码 | S14 |  |  | N7 |
| **理由说明** | 使用HLS(HTTP LIVE STREAMING)协议，延时虽然较高，但仍在可接受的范围内。且兼容性比较好，提高了可移植性。在性能和可移植性兼备的情况下，选择了HLS新协议。使用FFMPEG转码，可以保证视频质量，同时提高了可移植性 | | | |
| **相关架构图** |  | | | |

## 10.3敏感点和权衡点

#### 10.3.1敏感点

|  |  |  |
| --- | --- | --- |
| S1 | 对用户信息进行加密 | 保护用户信息，影响了安全性(正面)，是**安全性的敏感点** |
| S2 | 在系统服务收到访问时验证用户身份，授权给用户应有的权限 | 有利于保证系统服务仅受到合法访问，是**安全性的敏感点** |
| S3 | 自动攻击侦测 | 出现攻击时可以自动侦测到，提高了安全性，是**安全性的敏感点**，同时会增加服务器负担，降低性能，是**性能的敏感点** |
| S4 | 心跳 | 一个组件定期发出一个心跳信息，另一个组件收听该信息。可以用来识别错误，是**可用性的敏感点**。同时可能占用资源，导致性能的下降，是**性能的敏感点** |
| S5 | 主动冗余 | 所有的冗余组件都以并行的方式对事件作出响应，错误发生时，使用该战术的系统停机时间通常是几毫秒，是**可用性的敏感点**。但备份占用了系统资源，影响了性能，是**性能的敏感点** |
| S6 | 使用DDOS分防火墙 | 对各种常见的攻击行为进行识别，并通过集成的机制实时对这些攻击流量进行处理及阻断，是**安全性的敏感点**。 |
| S7 | 划分模块 | 将系统模块化，在修改时可以有效缩小要修改的范围，是**可修改性的敏感点** |
| S8 | 增加可用资源 | 使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络，可提高性能，是**性能的敏感点**。但会影响成本 |
| S9 | 采用分布式数据服务器 | 分布式数据服务器的性价比显著优于单台商业中型机和大型机，能够提供较为优质的数据服务，是**性能的敏感点（正面）**。但是由于数据分散存储，技术门槛较高，对维护工作，特别是数据完整性、一致性的维护提出了较高的要求，是**可维护性的敏感点（负面）** |
| S10 | 使用HLS(HTTP Live Streaming)协议 | 兼容性较好，提高了可移植性，是**可移植性的敏感点**。但延时比较高，影响了性能，是**性能的敏感点**。 |
| S11 | 独立存储密钥数据的模块 | 将密钥数据独立存储，可以提高可修改性，是**可修改性的敏感点** |
| S12 | 采用GlusterFS分布式文件系统 | 具有线性横向拓展能力，提高了可拓展性，是**可拓展性的敏感点**。增加了客户端的负载，占用了相当的CPU和内存，降低了性能，是**性能的敏感点**。 |
| S13 | 抽取并裁剪接口 | 针对请求响应交换的数据不一致情况，分别通过接口与外部系统相交互，是**互操作性的敏感点** |
| S14 | 使用FFMPEG转码 | FFMPEG转码提供了录制、转换以及流化音视频的完整解决方案，有助于播放清晰的视频，是**性能的敏感点**(正面),同时它包含了非常先进的音频/视频编解码库libavcodec，保证了高可移植性，是**可移植性的敏感点** |

#### 10.3.2权衡点

|  |  |  |
| --- | --- | --- |
| T1 | 自动攻击侦测 | 提高了安全性，但会增加服务器负担，降低性能，是**安全性和性能的权衡点** |
| T2 | 主动冗余 | 错误发生后恢复时间短，提高了可用性，但备份占用了系统资源，影响了性能，是**可用性和性能的权衡点** |
| T3 | 分布式数据服务器 | 性价比高，以相同成本提供更佳的容量和读写速率，但是维护工作技术门槛高，是**性能和可维护性的权衡点** |
| T4 | 使用gHLS(HTTP Live Streaming)协议 | 兼容性较好，提高了可移植性，但延时比较高，影响了性能，是性能和可移植性的权衡点 |
| T5 | 采用GlusterFS分布式文件系统 | 提高了可拓展性，同时降低了性能，是**可拓展性和性能的权衡点** |
| T6 | 心跳 | 提高了可用性，但降低了性能，是**可用性和性能的权衡点** |

## 10.4风险和非风险

#### 10.4.1风险

|  |  |  |
| --- | --- | --- |
| R1 | 自动攻击侦测 | 攻击不经常发生，系统一直自动侦测攻击，可能导致对资源的很大浪费，影响性能 |
| R2 | 心跳 | 可能造成性能负担，影响系统性能 |
| R3 | 主动冗余 | 备份占用了系统资源，可能会影响系统性能 |
| R4 | 增加可用资源 | 可提高性能，但成本较高 |
| R5 | 分布式数据服务器 | 分布式数据服务器技术门槛高，提高人力成本。如果维护不当，数据的完整性和一致性可能出现问题。 |
| R6 | 使用gHLS(HTTP Live Streaming)协议 | 延时比较高，可能影响系统性能 |
| R7 | 采用GlusterFS分布式文件系统 | 可能会增加客户端的负载，占用相当的CPU和内存，降低性能 |

#### 10.4.2非风险

|  |  |  |
| --- | --- | --- |
| N1 | 对用户信息进行加密 | 多用户系统和涉密系统的标准配置 |
| N2 | 在系统服务收到访问时验证用户身份，授权给用户应有的权限 | 多用户系统和涉密系统的标准配置 |
| N3 | 使用DDOS分防火墙 | 可以有效识别并阻断攻击，有效提高安全性 |
| N4 | 划分模块 | 成熟的设计原则，显著利于增强可修改性、可移植性 |
| N5 | 独立存储密钥数据的模块 | 同上 |
| N6 | 抽取并裁剪接口 | 通过接口与外部系统交互，提高了互操作性 |
| N7 | 使用FFMPEG转码 | 提高了性能和可移植性 |

# 挑战和经验

在本次项目实践中，针对校园讲座管理系统所需的诸多功能性需求和非功能性需求，我们将安全性、可获得性、互操作性、可修改性、性能和可维护性列为本次架构设计需要着重考虑的方面，提出了MVC以及SOA两种候选方案，由于系统对安全性以及成本有所限制，最终选择了MVC。

我们认为在这次实践中遇到的挑战主要有以下几点：首先，对于一些前人没有做过的新系统，通常难以一下子设计出合适的架构。在架构初期，通常都要经历一个不断探索的阶段；其次，在对ASR提出设计决策时，不可避免的会涉及到一些不熟悉的领域（比如此次的视频直播），这就需要我们快速了解一个陌生领域，从而提出合适的解决方案。

通过这次实践，我们掌握了一个系统架构的完整的设计流程，对架构设计方法ADD和架构评估方法ATAM有了更加深刻的理解：ADD是一个递归的分解过程，在每个阶段都选择战术和架构模式来满足一组质量属性场景，然后对功能进行分配，以实例化由该模式所提供的模块类型; ATAM作为一种架构评估的综合方法，需要我们根据质量属性需求列表以及每个高优先级场景相关的构架决策，判断这些决策是否有风险，从而找到架构中任何存在问题的地方，有利于构建一个健壮的软件架构。

以下为团队负责人的个人感想：经过实践，我充分认识到架构活动深深触及了软件系统“本质上的困难”即复杂度，软件系统作为一个有机整体，其架构设计和分析活动具有本质意义上的困难性和挑战性。也是由于这种原因，架构活动的各个阶段难以分工和并行进行。认识到这种困难，并激发出对这种挑战性工作的兴趣，是我从课程实践中得到的最大的收获。

# 12.组员和分工

|  |  |  |
| --- | --- | --- |
| **组员** | **学号** | **分工** |
| 梁思宇 | 131250129 | 在本次作业中负责对软件架构进行评估，采用了ATAM方法，分析构架方法，列出优先级场景，把相关构架决策编成文档，确定其有风险决策和无风险决策，敏感点和权衡点，并对其进行分类，以确信该方法的实例化适合满足所要达到的质量属性需求，最终进行归档。 |
| 邹卓晋 | 131250158 | 在本次作业中，我主要负责原型中MVC框架的搭建以及后端model和controller部分的编写。在代码编写完成后，我进行了原型的演示以及用户使用说明的编写。 |
| 曾婧 | 131250159 | 在这次项目中，经过商定决定选择了MVC的架构方法后，我负责了编写View层的部分代码，以及View层与Controller层的数据交互的代码，同时原型的页面设计。与需求撰写人员共同讨论了需求是否合理，如何实现等问题。同时还与负责Controller层的同学共同制定了接口，包括接口的命名、规范和详细参数。 |
| 吴超月 | 131250168 | 1. 负责本文档中功能点操作场景的文档编写 2. 负责非功能性需求和ASR的场景描述及文档编写 3. 负责MVC和SOA两种架构的模块视图绘制 4. 负责两个架构的ADD过程中每个ASR可选取的设计决策的编写 5. 负责MVC架构的ADD过程及文档编写，具体包括三个迭代，在每次迭代过程中对所选模块进行ASR的识别，进行设计决策的选择及分析，并进行每次迭代结果的产物图绘制 6. 负责本文档内容的整合和样式的统一、整理 |
| 罗瑶 | 131250177 | 在本次大作业中，我主要负责我们系统的ADD过程，部分参与ATAM过程。首先我开始对系统采用SOA架构进行ADD过程，一共进行3个迭代：分别对直播模块和补退选模块进行识别所选模块的ASR，进行设计决策的选择及分析；之后在ATAM过程中，我根据之前的开发画出了系统效用树；最后我进行了MVC和SOA两个架构选择的比较，最后选择出MVC架构。 |
| 陈云龙 | 131250181 | 在本次作业中，我主要负责原型设计、Controllor层、Model层和部分View层代码的编写、调试、项目部署与维护，根据MVC架构和SOA架构的比较结果选择MVC架构设计代码，以及系统UML的类图设计和绘图、组件/连接器与实现类的mapping，参与系统需求、功能，架构的讨论，和其他原型设计人员定义接口与前后端的交互。 |
| 倪小凡 | 131250185 | 在这次项目中，我负责了编写view层的部分代码。此外，还根据项目架构设计过程完成了整个项目设计过程中关于挑战和经验的总结。 |
| 丁霄汉 | 131250207 | 在团队项目中，我担任团队负责人，所做的工作主要包括分工，组织会议和讨论，架构模式的分析、选择和绘图，具体实现策略（如Gluster文件系统、HLS协议）的分析、选择，课堂PPT演示，以及对应的文档工作。  在提交产物中，我所贡献的部分包括：proposal和最终报告中的C&C视图，课堂演示所用的PPT，最终报告中的“两种架构模式的比较及最终选择”以及“具体实现技术的选择与解释”相关部分。 |