# 第一次迭代结果

采用MVC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。

# 第二次迭代过程

选取直播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、实时性有密切关系。

## 识别所选模块的ASR

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Architectural Drivers** | **Importance** | **Difficulty** |
| **1** | Scenario 3：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 4：可修改性 | medium | low |
| **3** | Scenario 5：负载性能：系统能够允许500个用户同时观看讲座直播 | high | high |
| **4** | Scenario 7：实时性 | high | high |

## 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 |
| 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 |

* 实时性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **分析** |
| 使用RTMP(Real Time Messaging Protocol)协议 | 适合长时间播放，延时较低，但是又累积延时；是Adobe开发的协议，无法再iPhone中兼容 |
| 使用HLS（ HTTP Live Streaming）协议 | 兼容性比RTMP好，但是延时比其高 |

## 设计决策的选择及分析

* 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。仅是关于直播的服务。 |
| 使用控制机制来管理、协调服务的调用 | 不采用。仅是关于直播的服务。 |
| 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。 |
| 运行时注册 | 不采用。会带来额外的管理开销。 |

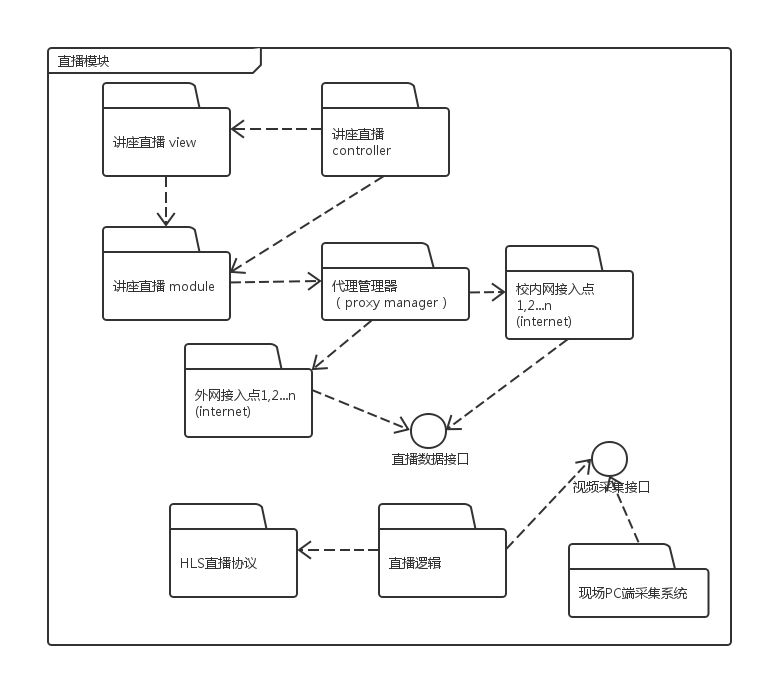
* 负载性能

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 维持计算的多个副本 | 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。适用于此类开发成本不应太大的系统。 |

* 实时性

|  |  |
| --- | --- |
| **可选的设计决策或模式** | **决策理由** |
| 使用RTMP(Real Time Messaging Protocol)协议 | 不采用。并没有很好的兼容性。 |
| 使用HLS（ HTTP Live Streaming）协议 | 采用。该技术成熟且容易使用。此外，开源有益于控制开发系统的成本。 |

## 第二次迭代结果



# 第三次迭代过程

选取讲座报名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。

## 识别所选模块的ASR

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Architectural Drivers** | **Importance** | **Difficulty** |
| **1** | Scenario 2：可获得性 | high | medium |
| **2** | Scenario 4：可修改性 | medium | low |
| **3** | Scenario 5：负载性能：系统能够允许500个用户同时进行正常的访问、报名操作 | high | high |

## 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 |

## 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。该系统对于选课报名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，对于讲座报名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。 |

## 第三次迭代结果

