项目名称：tzlOJ

文档名称：架构设计说明书

*HUSTZL*

*侯皓斐 软件2003班 U202010851*

*刘铭宸 软件2003班 U202010783*

*甘凤轩 软件2003班 U202010822*

版本：V2.1

文 档 信 息

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√ ] 正式发布  [ ] 正在修改 | 文件标识： | HUSTZL\_004 |
| 当前版本： | V2.1 |
| 作者： | 侯皓斐 |
| 发布日期： | < 2023.4.23 > |

文 档 更 改 记 录

| 版本 | 更改日期 | 更改人 | 更改原因 | 说明 |
| --- | --- | --- | --- | --- |
| V1.0 | 2023.4.15 | 侯皓斐 |  | 初版 |
| V2.0 | 2023.4.20 | 甘凤轩 | 校对 | 勘误并调整行文语句 |
| V2.1 | 2023.4.23 | 甘凤轩 | 调整格式 | 完善文档大纲，调整段落格式，修改序列编号 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1.项目概述 1](#_Toc133030971)

[2.软件架构的作用和表示方式 1](#_Toc133030972)

[3.软件架构的目标和约束 2](#_Toc133030973)

[4.架构设计方案 3](#_Toc133030974)

[4.1架构分析 3](#_Toc133030975)

[4.2设计思想 4](#_Toc133030976)

[4.3架构体系，子系统和模块划分 4](#_Toc133030977)

[5.用例视图 5](#_Toc133030978)

[6.逻辑视图 8](#_Toc133030979)

[6.1概述 8](#_Toc133030980)

[6.2影响软件体系结构的重要设计包 8](#_Toc133030981)

[7.进程视图 9](#_Toc133030982)

[8.部署视图 10](#_Toc133030983)

# 1.项目概述

本架构文档编写的目的是为了清晰地描述OnlineJudge系统的整体结构、各组件之间的交互和通信方式，以及系统的部署和维护策略等。这个文档的读者包括关键的利益相关者（高校教师、学生）和开发人员，他们需要了解系统的设计和实现细节，以便更好地使用和开发该系统。

在设计OnlineJudge系统的架构时，我们主要考虑以下参考依据和资料：

* 系统需求文档：OnlineJudge系统需要支持学生提交编程作业并进行自动评测，同时还需要支持教师发布题目和管理学生的作业。因此，架构设计应该满足这些需求，同时保证系统的可靠性、可扩展性和安全性。
* 技术框架：针对OnlineJudge系统的需求和架构要求，还有项目组擅长的框架，我们选择使用Django来实现Web应用程序，使用Docker容器技术来实现系统的部署和运维，使用VueJs编写前端。
* 经验和最佳实践： OnlineJudge系统开发需要考虑到性能、可维护性和可测试性等因素，因此可以借鉴其他在线评测系统的经验和最佳实践，如LeetCode、Codeforces等。

架构文档的内容应该包括以下部分：软件架构的作用和表示方式，软件架构的目标和约束，架构设计方案，子系统和模块划分，用例视图，逻辑视图，进程视图，部署视图，实施视图。

通过编写这样的架构文档，可以帮助开发团队更好地理解系统的整体架构和设计，同时也可以帮助用户更好地使用和维护该系统。

# 2.软件架构的作用和表示方式

软件架构在OnlineJudge系统中的作用是将系统分解为不同的组件并定义它们之间的相互作用和关系，以实现系统的功能和非功能需求，例如性能、可扩展性、可靠性和安全性。通过软件架构，可以有效地降低系统的复杂性，同时提高系统的可维护性和可重用性。

表示软件架构的方式通常是使用不同类型的视图。以下是必需的用例视图、逻辑视图、进程视图、部署视图或实施视图的概述：

* 用例视图：用例视图描述系统的用户需求和功能，并将它们分解为不同的用例。这个视图用于确定系统的功能和边界，以及与用户的交互。
* 逻辑视图：逻辑视图描述系统的逻辑组件和它们之间的关系。这个视图用于定义系统的核心结构，包括模型、服务、控制器和视图等。
* 进程视图：进程视图描述系统的运行时进程和它们之间的交互。这个视图用于定义系统的并发和通信，包括进程、线程和通信协议等。
* 部署视图：部署视图或实施视图描述系统的物理部署和运行环境。这个视图用于定义系统的硬件、网络和软件环境，包括服务器、存储设备、网络连接和操作系统等。
* 实施视图：从开发的角度来描述软件系统架构，包括其整体结构、层次结构、子系统，以及要使用的第三方控件，自定义控件，以及它们之间的接口。

这些模型元素可以帮助开发人员更好地理解和设计系统的不同方面，同时也可以帮助用户更好地了解系统的功能和实现

# 3.软件架构的目标和约束

软件体系结构的设计应该始终从软件需求和用户目标出发，以确保系统能够满足用户的需求和期望。我们考虑tzlOJ的具体场景，我们必须重点考虑以下性能相关的目标：

1. 系统安全性：OnlineJudge系统需要保证用户的数据安全和系统的稳定性。为了实现系统的安全性，应该采取一系列措施，例如使用HTTPS协议来保障用户的数据传输安全，采用安全编程实践来避免代码漏洞，使用防火墙和反向代理等技术来防范网络攻击。
2. 保密性：OnlineJudge系统需要保障用户数据的保密性，例如学生提交的编程作业、教师发布的题目的测试数据等。为了保护用户数据的机密性，应该采取加密和访问控制等措施，例如使用加密算法来保障数据传输和存储的安全，采用访问控制策略来限制敏感数据的访问。
3. 第三方组件的使用：OnlineJudge系统需要使用各种第三方组件和服务，例如自动评测系统（沙箱SandBox）、消息队列，数据库等。在选择和使用第三方组件时，需要考虑其稳定性、安全性、性能和可维护性等因素，以确保系统的可靠性和可扩展性。
4. 发布和重新使用：OnlineJudge系统需要能够方便地发布和重新使用，以便在不同的机器和环境中部署和运行。为了实现系统的发布和重新使用，应该采用自动化构建和部署技术，以便快速地发布和更新系统的版本。

在设计和实施OnlineJudge系统时，我们面临着很多的约束，分析结果如下：

1. 高性能要求： OnlineJudge系统需要支持不是特别多的并发用户，但是单个用户对系统资源的消耗是难以想象的。同时还需要保证系统的响应时间和吞吐量。为了实现高性能，可以采用异步处理、缓存技术等措施。
2. 团队结构： OnlineJudge系统的开发团队，HUSTZL团队（包括架构师、开发人员、测试人员等）参与相关大型项目技能和经验较少。针对OnlineJudge系统的需求和架构要求，还有项目组擅长的框架，我们智能选择使用Django来实现Web应用程序，使用Docker容器技术来实现系统的部署和运维，使用VueJs编写前端。
3. 遗留系统：OnlineJudge系统需要与其他系统进行集成和协作，例如学生管理系统、课程管理系统等。因此，需要对现有的遗留系统进行评估和测试，以确保其与OnlineJudge系统的兼容性和一致性。

# 4.架构设计方案

## 4.1架构分析

把握主要矛盾OnlineJudge系统的场景和问题域分析如下：

OnlineJudge系统的主要场景是高校教师和学生使用系统进行编程作业的提交、评测和管理。系统需要支持多种编程语言，例如C、C++、Java、Python等。系统还需要提供举办比赛，查看排行榜等比较小的功能。

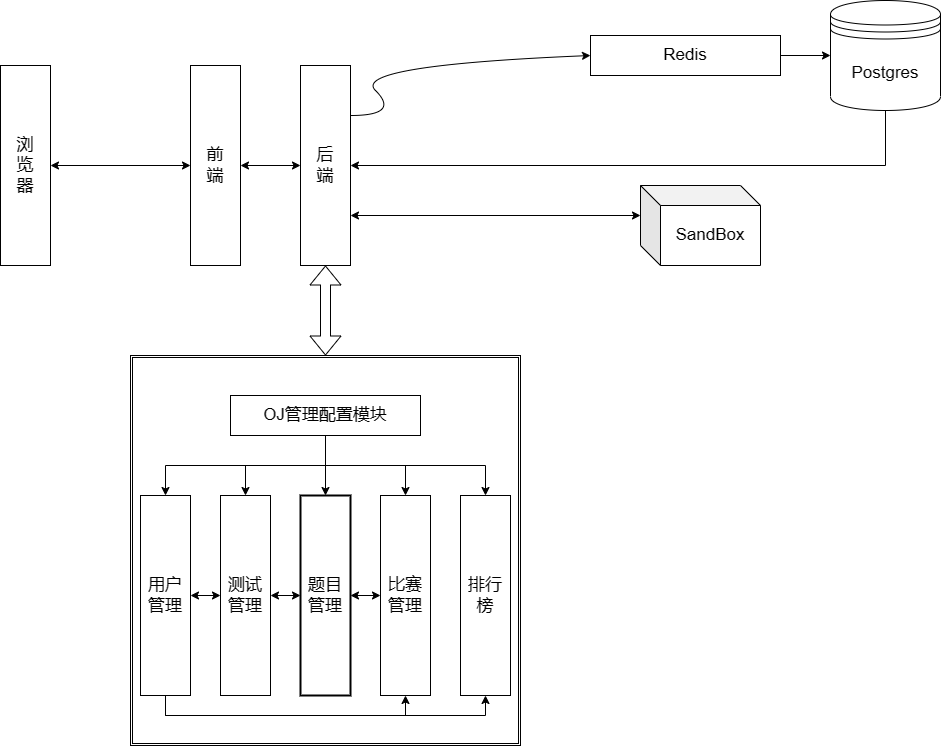
系统的问题域包括自动评测系统、数据库设计、用户界面设计、安全性和可扩展性等方面。自动评测系统是OnlineJudge系统的核心功能之一，需要实现自动编译和运行学生提交的代码，并生成评测报告。性能，安全性是系统的最核心的非功能需求，必须需要采用相应的措施和架构设计来保障系统的性能，安全性。

## 4.2设计思想

综合考虑OnlineJudge系统的特点，可以采用三层B/S模式，MVC模式和分层架构模式。MVC模式可以使系统更易于维护和扩展，同时还可以提高系统的可测试性和可复用性；分层架构可以实现系统的松耦合和高内聚，同时还可以提高系统的可维护性和可扩展性。此外，服务化架构模式和领域驱动设计模式也可以在系统设计中得到应用，以提高系统的灵活性和可维护性。

## 4.3架构体系，子系统和模块划分

采用三层B/S模式，MVC模式和分层架构模式的架构图如下：



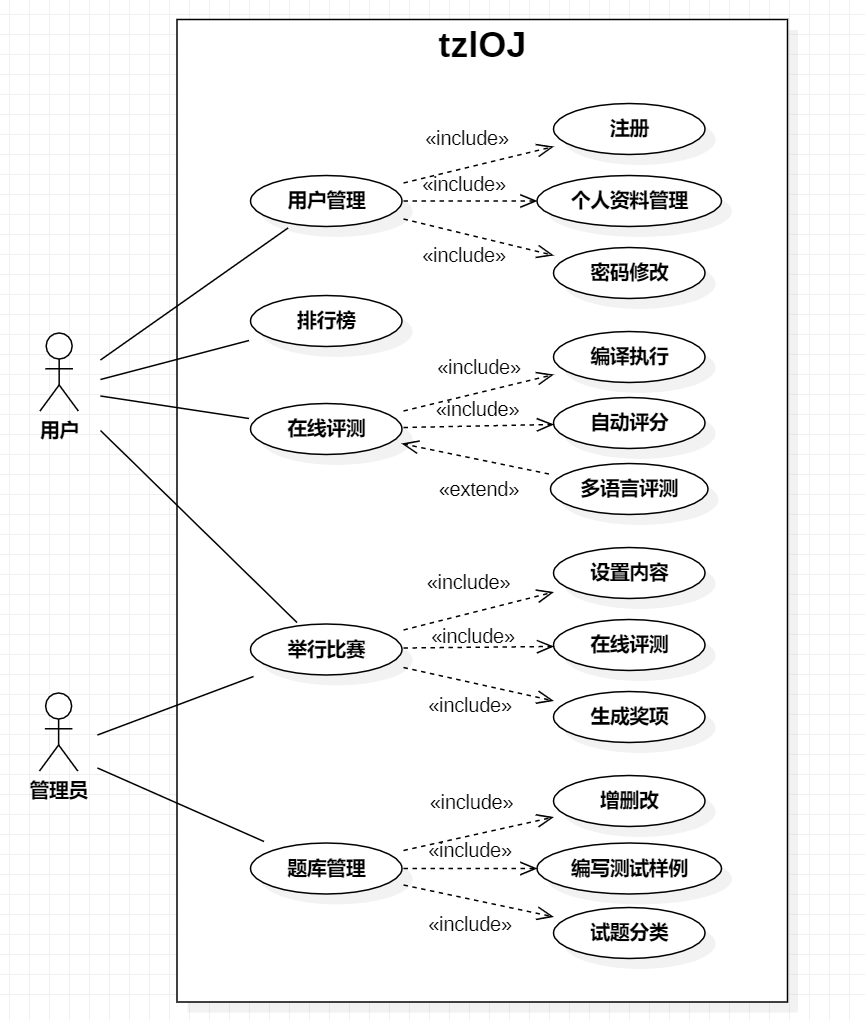
将系统分为三个层次：用户界面层、应用层、领域和基础设施层。

* 用户界面层：用户界面层是OnlineJudge系统的顶层，负责与用户进行交互。该层提供用户界面和交互逻辑，包括学生提交和管理员管理等功能。
* 应用层：应用层负责处理系统的业务逻辑和数据处理。该层包括自动评测系统、用户管理管理和比赛管理等模块。题目管理系统负责管理题目题面和测试数据等信息，测试管理负责利用SandBox自动编译和运行学生提交的代码，并生成评测报告，学生提交作业和查看提交历史等功能，用户管理负责管理账号提供用户信息，比赛管理负责比赛及其所属的题目的管理。排行榜子系统负责用户积分等信息，实现排行榜功能。
* 领域和基础设施层：领域和基础设施层负责系统的数据模型和数据访问。该层包括用户信息、成绩信息和提交历史等领域模型，以及数据访问接口和服务，而且是OnlineJudge系统的底层支撑，负责提供系统的底层服务和基础设施，特别是运行和编译程序的SandBox。领域和基础设施层负责将业务逻辑和数据逻辑进行分离，将业务逻辑和测试逻辑进行分离，并提供标准化的数据访问接口，以便应用层进行数据访问和处理。同时将测评与系统分离，放入一个安全的封闭的模拟沙箱，从而保证了系统的安全性。

# 5.用例视图

OnlineJudge系统是一个面向程序员和计算机科学学生的在线编程题解平台。系统的核心功能是提供编程题目的发布、管理和评测。下面是该系统的一些最重要、最核心的功能部分的用例或场景，较为详细的用例图见后续：

1. 注册：当用户第一次访问OnlineJudge系统时，他们需要创建一个账号。注册需要输入必要的个人信息，如用户名、电子邮件地址和密码。
2. 在线评测：OnlineJudge系统最重要的功能之一是在线评测。当用户提交他们的代码时，系统应该检查代码是否可以编译、是否符合题目的要求以及是否产生正确的输出。系统需要记录每次提交和评测的结果，并向用户提供反馈和错误信息。
3. 举办比赛：OnlineJudge系统还应该支持比赛功能，这是一个面向编程竞赛的特殊功能。管理员可以创建比赛，包括设定比赛开始时间和结束时间、添加比赛题目、设定题目的分值和难度级别等。比赛期间，参赛者只能看到与该比赛有关的题目。比赛结束后，系统应该根据参赛者的提交情况和正确答案的情况自动计算每个参赛者的得分，并公布获胜者。
4. 题库管理：OnlineJudge系统需要有一个完善的题库，包含多种编程语言的编程题目和其它问题。管理员可以创建、编辑和删除题目，并为每个题目设定标题、描述、输入格式、输出格式、样例输入和输出以及解题思路。系统应该能够智能地推荐题目给用户，并根据用户的历史提交情况进行个性化推荐。

、

在线评测是OnlineJudge系统中最重要的用例之一，它是该系统的核心功能。当用户提交代码时，系统需要对代码进行编译、执行并输出结果，然后比较用户的输出与标准输出是否一致，最后根据评测结果为用户计算分数和排名。以下是在线评测的详细描述和解释：

1. 用户提交代码：用户首先需要从题库中选择一道题目，并编写代码以解决该问题。代码可以使用多种编程语言编写，如C++、Java、Python等。用户需要将代码提交到系统中，系统将保存用户的代码，并分配一个唯一的提交ID。
2. 评测机编译和执行代码：系统需要将用户提交的代码发送给评测机进行编译和执行。评测机是一个独立的模块，负责评测用户的代码并输出结果。评测机可以支持多种编程语言和操作系统，并应该能够扩展以支持新的语言和系统。
3. 对比输出结果：当评测机完成代码的执行，系统需要将用户的输出与标准输出进行比较。标准输出是由题目作者提供的预期输出，系统需要将用户的输出与标准输出进行比较以判断用户的代码是否正确。
4. 计算分数和排名：根据比较的结果，系统将为用户计算得分，并更新用户在该题目中的排名。系统需要记录每个用户的提交历史和最佳成绩，并在排行榜上显示排名和成绩。

在线评测涉及到系统架构、模块设计和技术选型等方面的设计，以下是一些重要的设计元素：

1. 分布式系统架构：OnlineJudge系统应该采用分布式系统架构，以实现高可用性和可扩展性。系统中的不同模块应该运行在不同的服务器上，以减少单点故障和提高系统性能。
2. 评测机的选择和管理：OnlineJudge系统需要选择合适的评测机，并对评测机进行管理和监控。评测机应该具有高性能和稳定性，并且能够快速地执行代码和输出结果。
3. 安全性和隔离性：在线评测涉及到用户提交的代码，因此系统需要确保用户的代码不会影响系统的安全和稳定性。系统应该采用安全隔离措施，如沙箱技术和权限控制，以防止恶意代码的执行和攻击。
4. 异步测评：OnlineJudge系统的评测过程是一个异步的过程。当用户提交代码时，系统需要立即返回一个提交ID，并在后台进行评测和计算分数。因此，系统需要支持异步测评，如事件驱动模型和异步回调模型等。
5. 缓存和数据库：在线评测需要大量的计算和存储资源。为了提高系统性能和响应速度，系统需要采用缓存和数据库技术。缓存可以减少评测机的负载和响应时间，数据库可以存储用户提交历史和最佳成绩等信息

# 6.逻辑视图

## 6.1概述

系统分为三个层次：用户界面层、应用层、领域和基础设施层。

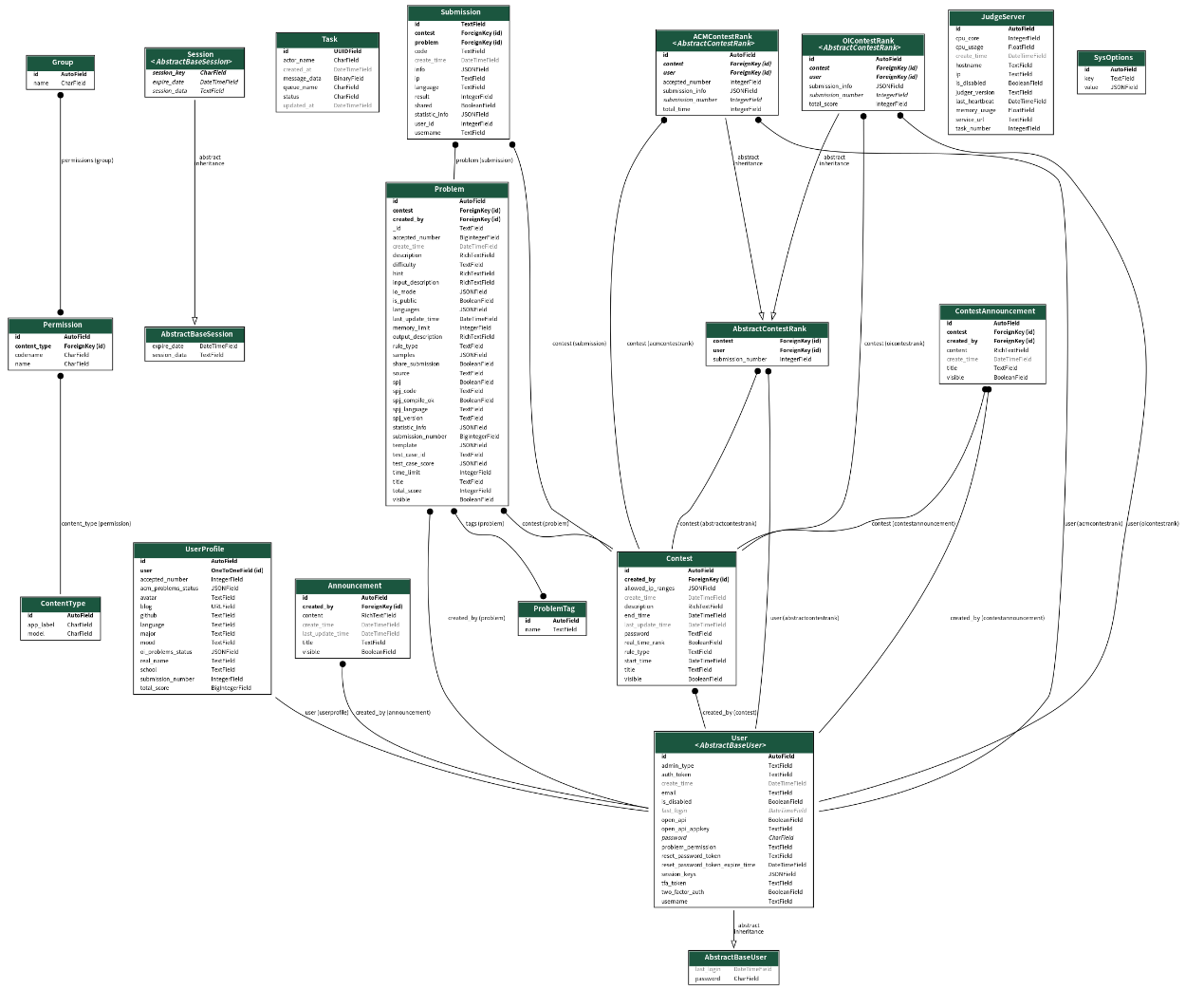
* 用户界面层（前端）：用户界面层是OnlineJudge系统的顶层，负责与用户进行交互。该层提供用户界面和交互逻辑，包括学生提交和管理员管理等功能。
* 应用层（后端）：应用层负责处理系统的业务逻辑和数据处理。该层包括自动评测系统、用户管理管理和比赛管理等模块。题目管理系统负责管理题目题面和测试数据等信息，测试管理负责利用SandBox自动编译和运行学生提交的代码，并生成评测报告，学生提交作业和查看提交历史等功能，用户管理负责管理账号提供用户信息，比赛管理负责比赛及其所属的题目的管理。排行榜子系统负责用户积分等信息，实现排行榜功能。
* 领域和基础设施层（数据库和SandBox运行环境）：领域和基础设施层负责系统的数据模型和数据访问。该层包括用户信息、成绩信息和提交历史等领域模型，以及数据访问接口和服务，而且是OnlineJudge系统的底层支撑，负责提供系统的底层服务和基础设施，特别是运行和编译程序的SandBox。领域和基础设施层负责将业务逻辑和数据逻辑进行分离，将业务逻辑和测试逻辑进行分离，并提供标准化的数据访问接口，以便应用层进行数据访问和处理。同时将测评与系统分离，放入一个安全的封闭的模拟沙箱，从而保证了系统的安全性。

注：详细信息可见层次系统架构图。

## 6.2影响软件体系结构的重要设计包

我们由于约束条件按照设计我们采用了第三方API，使用docker部署和运行，我们redis，postgres均将使用官方提供的标准docker，SandBox将使用HUSTOJ的沙箱运行环境的docker，三者均直接从云端拉取。

所以最关键的重要设计包为后端所在的应用层，我们将基于django框架开发，而我们初步的设计如下：（由于不太好调整，并未按照具体模块划分）



# 7.进程视图

系统运行态下一共运行着4+N个进程：

1. 用户界面层进程：该进程组负责处理用户的请求和响应，并提供用户界面和交互逻辑。该进程组包括Web前端、移动客户端等用户界面，以及与之对应的应用服务器。由于Vue强大的多端能力，前端服务可以仅由一个进程实现。负责处理用户的请求和响应，将用户请求发送给应用服务器，并将应用服务器的响应呈现给用户。该进程使用HTTPS协议和Web应用服务器进行通信。
2. 应用层进程：该进程组负责处理系统的业务逻辑和数据处理，并提供自动评测系统、用户管理和比赛管理等功能。由Django框架实现，该进程主要实现了该进程组仅有一个进程实现，包括题目管理功能、测试管理功能、用户管理功能、比赛管理功能等。
3. 领域和基础设施层进程组：该进程组负责系统的数据模型和数据访问，以及提供运行和编译程序的SandBox。该进程组包括多个子进程，如数据访问进程、评测机进程（SandBox进程）等。
   1. 数据缓存进程：在数据库和应用层之间加入缓存，避免数据丢失和保障数据安全性，通过HTTPS协议与应用层进程交换数据。
   2. 数据库进程：负责将系统的数据存储到数据库中，并从数据库中检索数据。该进程使用数据访问接口和服务与数据层进行通信。
   3. 评测机进程（N个）：负责利用SandBox自动编译和运行学生提交的代码，并生成评测报告。该进程使用HTTPS协议与应用层进程进行通信，同时采用消息队列和消息传递方式保障异步测评，并使用SandBox进行代码的运行和评测。

# 8.部署视图

我们采用docker进行部署，同时可以使用多个物理机，提高我们测评的效率，我们知道，性能限制出现在测评环节，即测评机进程，而且测评机进程极容易被恶意程序卡崩，所以必须采取冗余的战术和心跳的技术，当测评机出现问题时，后端及时调整测评。

我们的物理部署图如下：（见下页）

