任务众包平台

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <14/5/24> | 1.0 | 草稿 | 滕昊益 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 技术视图 5

8. 数据视图（可选） 5

9. 核心算法设计（可选） 5

10. 质量属性的设计 5

软件架构文档

# 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

无

# 用例视图

1. **用户登录和权限管理：**
   * 用户通过用户名和密码登录系统，系统验证身份并授予对应权限。
   * 管理员用户可以管理用户权限、添加新用户等操作。
2. **任务处理与管理：**
   * 用户可以创建新需求，包括选择众包者、描述需求信息等。
   * 管理员可以查看和管理任务，包括修改任务状态、取消任务等操作。
3. **支付与结算：**
   * 用户可以选择支付方式并完成订单支付。
   * 系统能够处理支付请求、生成支付记录，并与第三方支付平台对接实现结算功能。
4. **数据同步与备份：**
   * 能够进行数据同步和备份，保证数据安全性和完整性。
   * 管理员可以手动触发数据备份和恢复操作。
5. **用户评价与反馈：**
   * 用户在公会详情页下方可以填写评价和评论。
   * 管理员可以查看用户评价和反馈，进行回复或处理投诉。
   * 系统定期汇总用户评价，生成最受好评的公会或众包者和用户满意度报告。

# 逻辑视图

## 概述

1. 逻辑架构：使用MVC架构+分层（Layer）架构。

应用层（MVC架构）：客户发出请求后，路由会根据客户请求找到对应界面，并将界面交给界面处理器。界面处理器根据界面信息通过数据库交互模型从数据库获取数据后，将数据与代码逻辑结合，最后返回客户需要的界面代码。用户接收到结果后将结果渲染为前端界面。

分层(Layer)架构:

**表示层（Presentation Layer）：**

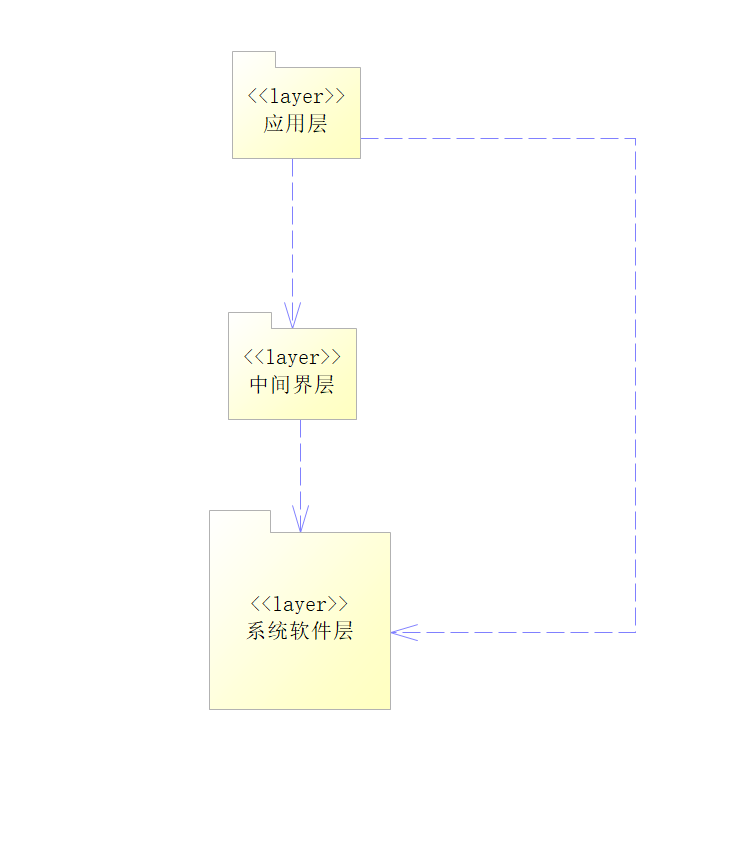
* + 负责用户界面的展示和用户交互逻辑。
  + 包括页面展示子系统和用户交互子系统。

**中间界层（Middle-tier Layer）：**

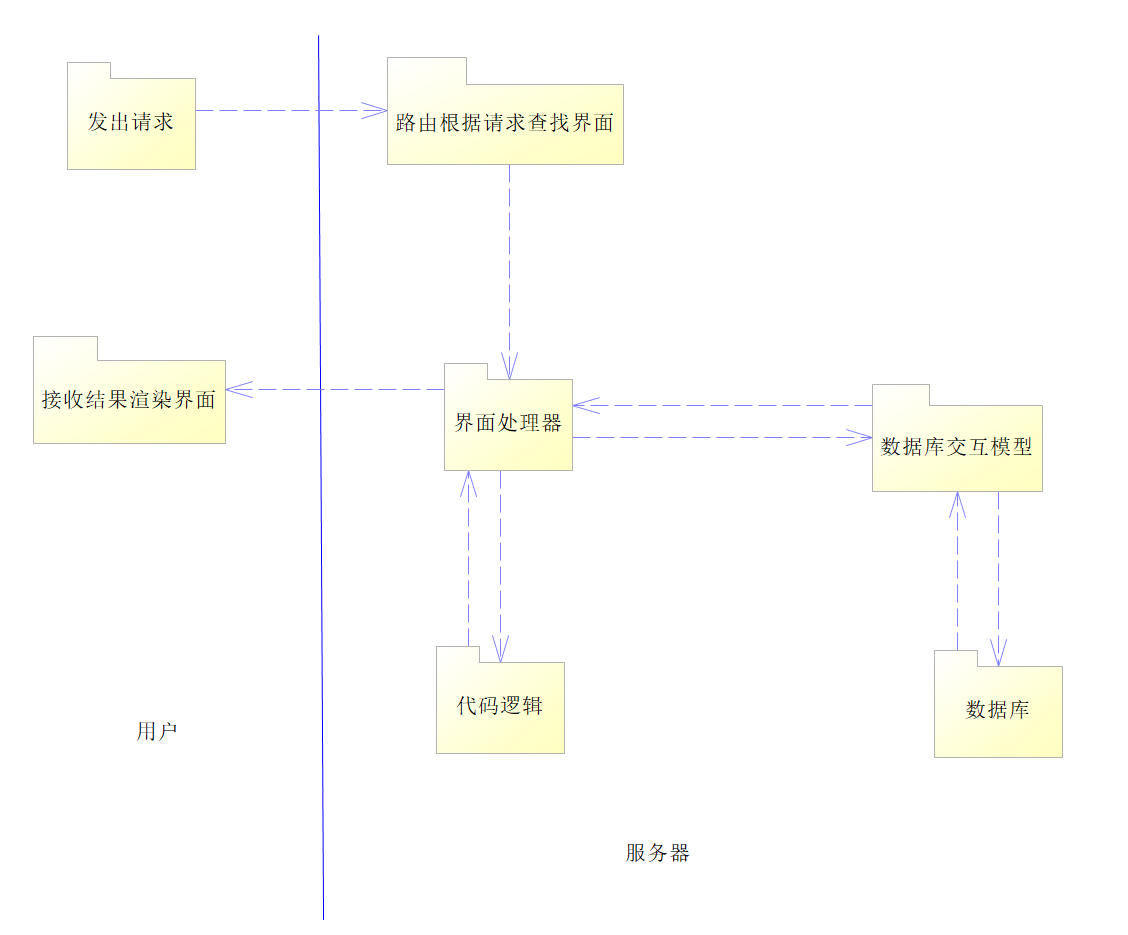
* + 执行业务逻辑和处理用户请求。
  + 包括业务逻辑子系统和数据访问子系统。

**数据层（Data Layer）：**

* + 负责数据的存储和管理。
  + 包括数据库系统和数据访问工具。



## 在构架方面具有重要意义的设计包



中间界层：



软件系统层：



# 进程视图

1. **用户请求进程组：**

后端界面进程：接收前端界面的请求，根据请求类型，进入相应逻辑处理线程处理，处理业务逻辑，并返回结果。

通信模式：HTTP请求-响应模式、MQ消息队列存储转发。

1. **数据处理进程组：**

数据处理进程：负责对数据库中数据进行增删查改，进行数据查询、分析和存储，与数据库进行交互。

通信模式：Jpa与数据库进行交互。

1. **通知与日志进程组：**

通知发送进程：负责发送系统通知、任务状态更新等信息给用户。

日志记录进程：记录系统运行日志和用户操作日志。

上报线程：记录系统状态日志，方便运维进行管理。

通信模式：消息传递（发送通知消息）、文件记录（写入日志文件）。

# 部署视图

**物理节点和互连情况**

数据库节点：负责存储和管理数据，进行分布式存储管理数据，与服务器节点通过以太网连接。

前端服务器节点：托管前端页面文件，如webpack打包的HTML、CSS、JavaScript等。使用Nginx进行与后端服务器节点连接。

后端服务器节点：进行处理业务逻辑和数据交互。使用docker进行后端节点的部署，使用Nginx对后端服务器节点进行反向代理，从而与前端服务器节点连接。

# 实现视图

1. **整体结构：**
   * 前端层：负责用户界面和交互逻辑，包括HTML、CSS、JavaScript等文件。
   * 后端层：为前端传递业务逻辑处理结果和进行数据交互，包括后端服务、数据库等组件。
2. **层和子系统的分解：**
   * 前端层：
     + 页面展示子系统：负责展示用户界面，包括各种页面和组件,如公会页，任务页，需求页等。
     + 用户交互子系统：处理用户输入、交互逻辑和页面跳转等功能。
   * 后端层：
     + 业务逻辑子系统：实现具体的业务逻辑，通过处理分发到各个进程,处理用户请求并调用相关服务。
     + 数据访问子系统：负责与数据库交互，包括读取、写入和修改数据等操作。
3. **重要构件：**
   * 前端框架：如React等，用于构建前端界面和交互逻辑。
   * 后端框架：如Spring Boot等，用于构建后端服务和业务逻辑。
   * 数据库系统：如MySQL等，用于数据存储和管理。
   * 日志系统：集成日志的存储和生成。
4. **系统通信与集成：**
   * 前后端通信：通过RESTful API和Axios实现前后端的数据交互和通信。
5. **安全和性能考虑：**
   * 安全性措施：包括用户认证、授权、数据加密等，保障系统的安全性。
   * 性能优化：通过缓存、异步处理、负载均衡等手段优化系统性能，提高响应速度和并发能力。

# 技术视图

编程语言: 前端是使用js和css, 后端使用java和sql

开发工具: webstorm、intellij、datagrip、postman

框架: 前端使用react框架，后端使用springboot框架

数据库： 使用mysql

中间件： nginx、docker、redis、kafka

# 数据视图（可选）

1. **数据实体：**
   * 用户信息实体：包括用户ID、众包者ID、用户名、密码哈希、电子邮箱等信息。
   * 众包者信息实体：包括众包者ID、用户ID、描述、封面、头像。
   * 订单信息实体(已接单的任务)：

包括订单ID、用户ID、接单者ID、任务ID、订单详情、状态等信息。

* + 任务信息实体：包括商品ID、名称、描述、价格等信息。

1. **数据关系：**
   * 用户与订单之间的关系：

一个用户可以有多个订单或接取多个订单，而一个订单只属于一个用户。

* + 订单与任务之间的关系：一个订单可以包含即是一个任务，两者一一对应。
  + 众包者与用户的关系：一个用户可以是一名众包者，也可以不是，或是公会（众包者一种）的一员。

1. **数据库设计与规范：**
   * 使用规范化设计原则：将数据分解为符合第三范式的多个表，避免数据冗余和更新异常。
   * 数据表之间的关联：使用外键关联用户、订单和商品等表，确保数据的一致性和完整性。
2. **数据库操作与维护：**
   * 数据库查询与更新：使用spring jpa进行数据访问和操作，包括查询用户信息、创建任务、添加任务到订单等操作。
   * 数据库事务管理：通过事务管理保证数据的一致性和可靠性，例如在创建订单时同时更新商品库存和订单状态，在发生错误时进行回滚。
3. **数据库管理和优化：**
   * 数据库备份与恢复：定期进行数据库备份，以防数据丢失或损坏。
   * 数据库性能优化：包括索引优化、查询优化、表结构优化等，提高系统的响应速度和性能。

# 核心算法设计（可选）

1. 使用Redis对Jwt进行缓存实现单点登录。
2. 使用Redis构造bloom过滤器，防止缓存击穿。

# 质量属性的设计

1. **性能：**
   * 并发和并行处理：利用多线程、异步任务等技术实现并发和并行处理，提高系统的并发性能。
2. **可扩展性：**
   * 模块化设计：采用模块化的架构设计，使得系统各个模块之间松耦合，便于扩展和维护。
3. **可靠性：**
   * 异常处理和容错机制：设计健壮的异常处理和容错机制，保证系统在面对异常情况时能够正确响应并保持稳定运行。
   * 数据备份和恢复：实现定期数据备份和恢复机制，防止数据丢失和损坏。
4. **易用性：**
   * 用户界面设计：设计简洁清晰、易于操作的用户界面，提高用户体验。
   * 用户反馈和改进：定期收集用户反馈，不断改进系统功能和界面设计，提高系统的易用性。
5. **可移植性：**
   * 跨平台兼容性：通过网页端，实现采用跨平台的开发技术和框架，确保系统能够在不同平台上运行和部署。

* 安全性设计：使用https进行加密传输，不明文存储用户登录秘钥，采用加密算法、访问控制、身份认证等技术，保护系统免受攻击和数据泄露。
* 保密性保护：对敏感数据进行加密存储和传输，限制权限访问，确保数据的保密性和隐私性。