OASIS系统项目设计文档

[0. 版本更新 2](#_Toc632529498)

[1．引言 3](#_Toc879220236)

[1.1编写目的 3](#_Toc219531445)

[1.2定义 3](#_Toc288090569)

[1.3参考资料 3](#_Toc1510052845)

[2．任务概述 4](#_Toc496425669)

[2.1目标 4](#_Toc452250288)

[2.2运行环境 4](#_Toc1025963683)

[2.3需求概述 4](#_Toc1225418418)

[2.4条件与限制 4](#_Toc1239176596)

[3．总体设计 5](#_Toc544640366)

[4. 逻辑视角 5](#_Toc1195327848)

[5．架构设计 7](#_Toc165623651)

[5.1 静态站点架构分解 7](#_Toc497895845)

[5.2 服务端架构分解 8](#_Toc1539178203)

[5.3 接口定义 10](#_Toc380046059)

[6. 信息视角 12](#_Toc817747435)

[6.1 信息持久化对象 12](#_Toc2133282892)

[6.2 数据源 12](#_Toc1846079179)

[6.3 领域建模设计 12](#_Toc209029597)

# 版本更新

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人** | **日期** | **变更原因** | **版本号** |
| 丁玲燕 | 2020.02.17 | 创建草稿 | V0.1 |
| 陆放明 | 2020.02.27 | 目录更改，更新服务端架构设计说明 | V0.2 |
| 丁玲燕 | 2020.02.29 | 更新前端架构设计说明 | V0.3 |

# 1．引言

## 1.1编写目的

本文档详细完成对学术关系图谱系统OASIS的设计，达到指导详细设计和开发的目的，同时实现测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户编写，是了解系统的导航。

## 1.2定义

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| OASIS | 学术关系图谱系统 |  |

## 1.3参考资料

1. 项目启动文档

2. 需求规格说明书

3. 测试文档

4. 计划文档

# 2．任务概述

## 2.1目标

将不同数据源的学术数据集成到数据库，抽取其中的实体以及实体与实体之间的关系，构建一个学术关系图谱系统（OASIS）并提供用户搜索与相应展示。

## 2.2运行环境

前端运行在主流浏览器上，包括Chrome、Firefox、Edge等；后端运行在阿里云服务器上。

## 2.3需求概述

OASIS需要构建学术关系，提供高效的论文查询、学术关系查询，构建学者画像及学术机构画像，提供学术同行评价、专家推荐系统、学术机构评价等，并将其可视化。

## 2.4条件与限制

CON1：采用Java语言及其它相关的Web开发

CON2：系统使用的是基于Web的数据库应用系统

CON3：项目需要完整的单元测试、集成测试、系统级测试

CON4：项目后期会增加需求及开放式功能

CON5：将个人工程行为尽可能地记录在Gitlab上

CON6：每次迭代产品均必须完成部署（使用Jenkins实现一键部署）

# 3．总体设计

* 系统主要以信息分发的方式进行架构的构建，根据已有的系统边界和参与者，限定使用者仅拥有相关的数据访问权限。参与者可以直接以游客访问的身份获取到平台的信息资源，从而不需要相关的权限认证模块搭建。
* 系统采取前后端完全解耦的方式进行架构设计，以 Tire 架构作为系统的主要架构风格，按照**数据源、中间件、服务、静态站点**进行集群的搭建与分配，从而尽可能降低各个Tire之间的相互影响性。
* 系统采用DDD进行领域模型设计

# 4. 逻辑视角

OASIS 系统中，采用前后端解耦的基本架构，其中UI展示部分采用单Tire进行架构构建，服务端架构采用多Tire逻辑协同进行

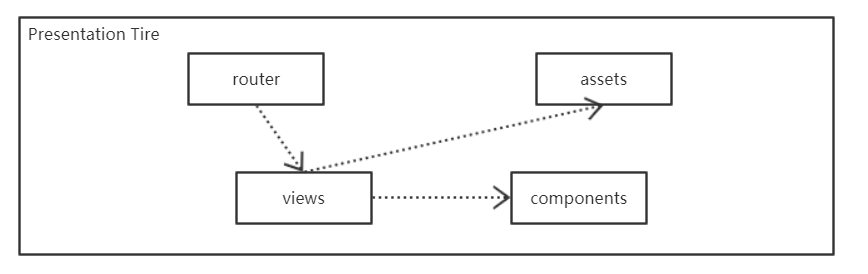
图1：UI展示层架构视图

图2：服务端架构视图

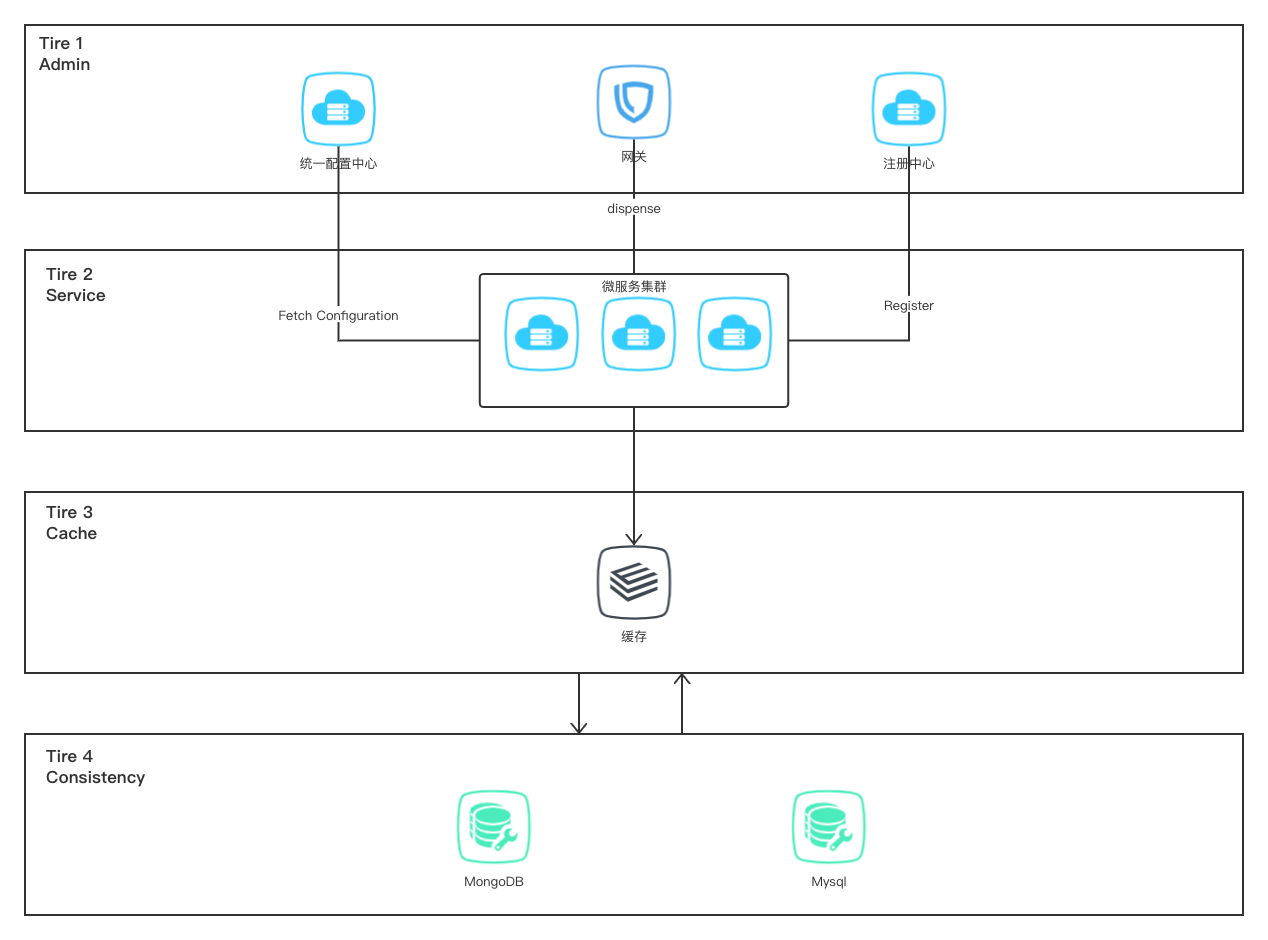
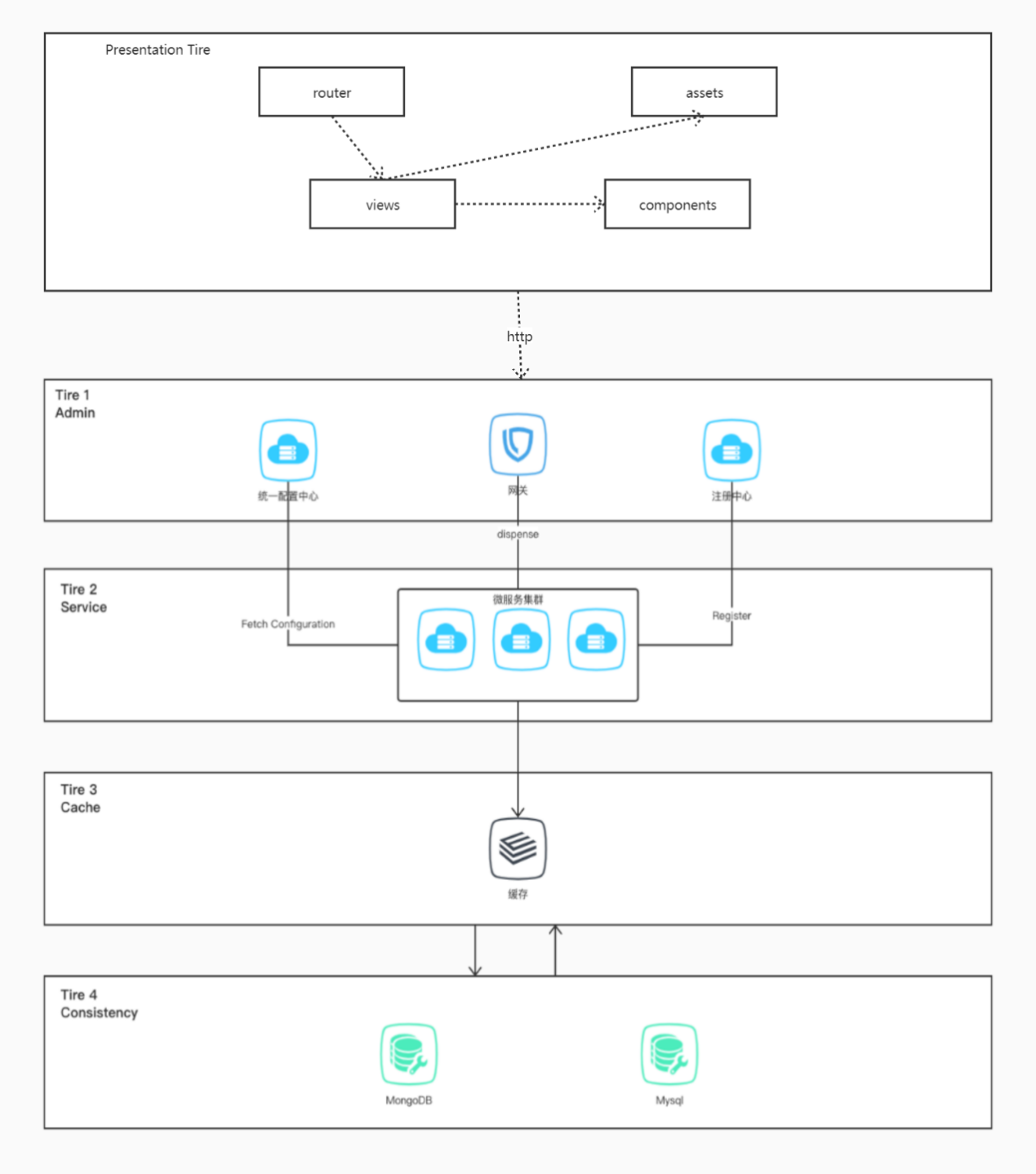


图3：系统整体架构视图



# 5．架构设计

## 5.1 静态站点架构分解

技术栈：Vue.js + Vue-router + ElementUI + axios

主要实现本系统的前端页面展示。在基于vue-cli 2.5的模板基础上加入了vue-router等配套设施，由vue-router来控制views中各页面的跳转。开发时采用组件化策略，降低与页面之间的耦合，并使用ElementUI组件库进行美化，views中各页面则复用已开发好的组件。使用axios与后端进行数据通信。

## 5.2 服务端架构分解

1. 主从式架构设计

技术选型：Spring Cloud eureka , zuul

主要实现服务注册与服务发现。系统的主要业务由一组独立的微服务组成，Worker启动之后，将会以心跳机制注册到Admin注册中心，并且从配置中心获取服务相对应的配置内容，目前我们通过服务注册与发现来让微服务可以感知彼此，微服务框架在启动的时候，将自己的信息注册到注册中心，同时从注册中心订阅自己需要引用的服务。

此外，对外采用统一的路由接入机制，使用Zuul 进行统一的路由分发和熔断。

通过统一配置管理来实现一个中心化的外部配置。

1. 缓存架构设计

当前数据源为MongoDB与Mysql ，对于耗时较大的搜索查询服务，使用Redis 集群进行中间件配置，起到缓存的作用。具体缓存策略如下：

|  |  |
| --- | --- |
| 服务类别 | 缓存变更行为 |
| 搜索资源 | 根据业务请求查询缓存，若存在则直接返回，否则加入缓存之后再进行返回 |
| 更新资源 | 实行饿汉式加载，若缓存中存在相应的内容，先进行缓存更新，后进行底层冷数据更新 |
| 删除资源 | 若缓存中存在该资源，则需要先进行缓存删除，后进行低层冷数据的删除 |
| 添加资源 | 若缓存中不存在该资源，则需要先进行缓存添加，保证下一次查询可以命中，后进行低层冷数据的添加 |

3. 整体部署架构设计

多个Tire之间采用 docker swarm 来进行一体化CI/CD，使用overlay网络进行容器间通信，具体网络协议为UDP / TCP 协议。

针对系统可用性问题进行了Docker镜像构建上的优化，主要使用JIB插件来进行构建时的镜像同步更新，借助阿里云镜像平台进行快速镜像构建与部署。

## 5.3 接口定义

Paper导入

**接口信息**

接口名称:paper导入

接口路径:/api/paper

请求协议:HTTP

请求方法:POST

接口使用状态:正常启用

**请求参数**

参数类型：Json

根类型: Object

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数名** | **说明** | **必填** | **类型** | **限制** | **示例** |
| id | 论文id | 是 | [number] |  |  |
| title | 论文题目 | 是 | [string] |  | title |
| abstract | 摘要 | 是 | [string] |  |  |
| conference | 会议名称 | 是 | [string] |  |  |
| affiliation | 隶属机构名,以分号隔开 | 是 | [string] |  |  |
| authors | 作者名,以分号隔开 | 是 | [string] |  |  |
| terms | 术语 | 是 | [string] |  |  |
| keywords |  | 是 | [string] |  |  |

**论文全局搜索**

**接口信息**

接口名称:论文全局搜索

接口路径:/api/query/paper/list?query=

请求协议:HTTP

请求方法:GET

接口使用状态:正常启用

**GET参数**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数名** | **说明** | **必填** | **类型** | **限制** | **示例** |
| query | 搜索关键字，支持【paper名，作者、机构、会议名、研究方向名】的查找 | 是 | [string] |  |  |
| pageSize | 每一页的大小，若实际数据不够则只返回部分 【默认值为10】 | 否 | [number] |  | 10 |
| pageNum | 页号 start from 0【默认值为0】 | 否 | [number] |  | 0 |

**返回参数**

参数类型：Json

根类型: Object

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数名** | **说明** | **必填** | **类型** | **限制** | **示例** |
| id | 论文id | 是 | [number] |  |  |
| title | 论文题目 | 是 | [string] |  | title |
| abstract | 摘要 | 是 | [string] |  |  |
| conference | 会议名称 | 是 | [string] |  |  |
| affiliation | 隶属机构名,以分号隔开 | 是 | [string] |  |  |
| authors | 作者名,以分号隔开 | 是 | [string] |  |  |
| terms | 术语 | 是 | [string] |  |  |
| keywords |  | 是 | [string] |  |  |

# 6. 信息视角

## 6.1 信息持久化对象

Raw Data PO

|  |  |
| --- | --- |
| PO | 内容 |
| PaperPO | 包含唯一标识id，论文名称，概要等 |

## 6.2 数据源

同时采用MongoDB的文档化存储和Mysql的数据表进行模型建立

## 6.3 领域建模设计

领域建模设计如下

|  |  |
| --- | --- |
| 名称 | 内容 |
| Paper | 基本论文内容，需要关联其对应的作者、会议 |
| Author | 关注于单个作者的信息，作者可以和多个论文相互对应，需要关联到作者对应的机构 |
| Affiliation | 机构的相关内容，和作者模型相互关联 |
| Conference | 会议实体，是前三个实体的中间联结点 |