面向工业制造领域的多模态数据治理平台

软件架构文档

版本 <1.2>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 14/10/23 | 1.0 | 根据立项建议书和需求规约编写架构文档 | 曾宇欣、钱麒丹、陈治源 |
| 31/10/23 | 1.2 | 补充并完善软件架构描述、整理复核 | 曾宇欣、陈诺、张瑞轩 |
| 20/11/23 | 2.0 | 根据上课内容完善软件架构内容 | 钱麒丹、陈治源、陈诺 |

目录

1. 简介 5

1.1 目的 5

1.2 范围 5

1.3 定义、首字母缩写词和缩写语 5

1.4 参考资料 5

1.5 概述 5

1.6 架构表示方式 5

2. 架构目标和约束 6

2.1 架构目标 6

2.2 架构约束 6

2.3 用例视图 7

2.3.1 定义数据模型 7

2.3.2 基于相似模型的标签推荐 7

2.3.3 数据模型管理 7

2.3.4 数据源接入 7

2.3.5 导入数据 7

2.3.6 基于安全规则的数据加密 7

2.3.7 基于安全规则的数据脱敏 8

2.3.8 基于质量规则的数据清洗 8

2.3.9 基于质量规则的数据监控 8

2.3.10 数据源管理 8

2.3.11 数据资产检索 8

2.3.12 数据资产导出 8

2.3.13 数据地图展示 8

2.3.14 数据相似度查询 8

2.3.15 大模型交互 8

2.4 逻辑视图 8

2.5 部署视图 9

2.5.1 Web服务器 10

2.5.2 应用服务器 10

2.5.3 数据库服务器 10

2.6 开发视图 11

3. 类设计 12

3.1 DataModel (抽象类) 12

3.2 StructuredDataModel 12

3.3 SemiStructuredDataModel 12

3.4 UnstructuredDataModel 13

3.5 Field 13

3.6 QualityRule (抽象类) 13

3.7 StructuredRule 13

3.8 SemiStructuredRule 13

3.9 UnstructuredRule 13

3.10 DataSource (抽象类) 13

3.11 PostgresqlDataSource 13

3.12 MongodbDataSource 13

3.13 InfluxdbDataSource 14

3.14 DirectoryDataSource 14

4. 测试方案 14

4.1 功能性测试 14

4.2 可靠性测试 14

4.3 性能测试 14

4.4 易用性测试 14

4.5 可维护性测试 14

5. 大小和性能 14

6. 质量 14

软件架构文档

# 简介

## 目的

本文档的主要目的是为面向工业领域的多模态数据治理平台提供详细的架构综述。我们将采用多种不同的架构视图，以清晰地描述系统的各个方面，并记录关键的架构决策，以引导系统的开发过程。本文档将为项目团队和利益相关者提供对系统整体架构的全面理解，指导开发团队在设计、实施和维护系统时遵循一致的架构准则，同时记录已经做出的重要架构决策，以供将来的参考和沟通。

## 范围

本文档适用于高级软件开发与管理课程的第一小组正在开发的面向工业领域的多模态数据治理平台，其范围可能包括：描述系统的整体架构，包括主要组件、模块和其相互关系；强调系统的功能和性能需求，以确保开发团队的一致性理解；提供指导，以确保系统满足工业领域数据治理的最佳实践和标准；记录已经做出的关键架构决策，包括技术选择、架构风格和设计原则。

## 定义、首字母缩写词和缩写语

|  |  |
| --- | --- |
| **定义及缩略语** | **含义** |
| 多模态 | 指的是工业中涉及多种数据类型和传感器模式的系统，包括时序信息、文本和图像。 |
| 数据治理 | 指的是管理、维护和保护数据的一系列过程，以确保数据的规范性、高质量和安全性，以支持数据的有效管理和利用。 |
| 大模型 | 在数据治理平台中使用复杂、高性能的大语言模型，以处理和分析大规模、多模态的工业数据以提取有价值的信息。 |
| ETL | 即Extract-Transform-Load，描述将数据从数据源端经过抽取、转换、加载到目的端的过程。 |

## 参考资料

1. 《面向工业制造领域的多模态数据治理平台-立项建议书》，高级软件开发与管理课程第1小组。
2. 《面向工业领域的多模态数据治理平台-软件需求规约文档》，高级软件开发与管理课程第1小组。
3. 《数据治理技术》，软件学报。
4. 《人工智能在数据治理中的应用》，信息通信技术与政策。
5. 《智能数据治理平台》，信息安全研究。
6. 《大数据之路 阿里巴巴大数据实践》，电子工业出版社。
7. 《软件工程原理》高等教育出版社，2013年2月第一版。

## 概述

本文档将明确面向工业领域的多模态数据治理平台的架构表示方式、架构的模板和约束、大小和性能、并通过以下一系列视图来表示平台的软件架构：用例视图、逻辑视图、进程视图和部署视图。

## 架构表示方式

本文将通过以下一系列视图来表示面向工业领域的多模态数据治理平台架构：用例视图、逻辑视图、部署视图和开发视图。

# 架构目标和约束

## 架构目标

软件架构需要支持本平台的所有功能需求和非功能需求的实现；

软件架构需要进行合理的模块划分，保证平台具有良好的可维护性。

## 架构约束

（1） 前端开发语言：HTML、JavaScript、CSS。

（2） 服务端编程语言：Java、Python。

（3） 前端Web框架：Vue.JS。

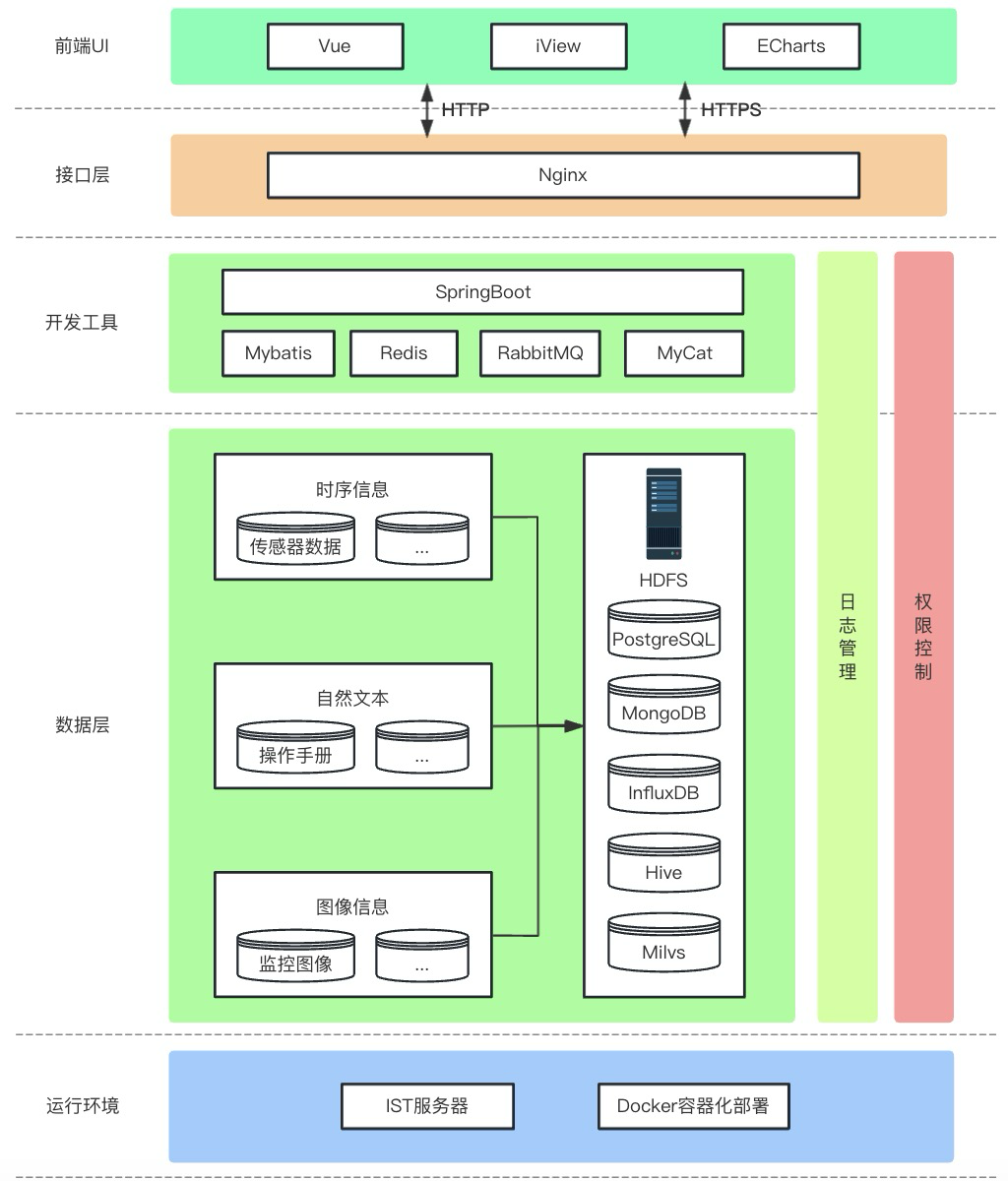
（4） Web UI库：iView、ECharts。

（5） 前端服务器：Nginx。

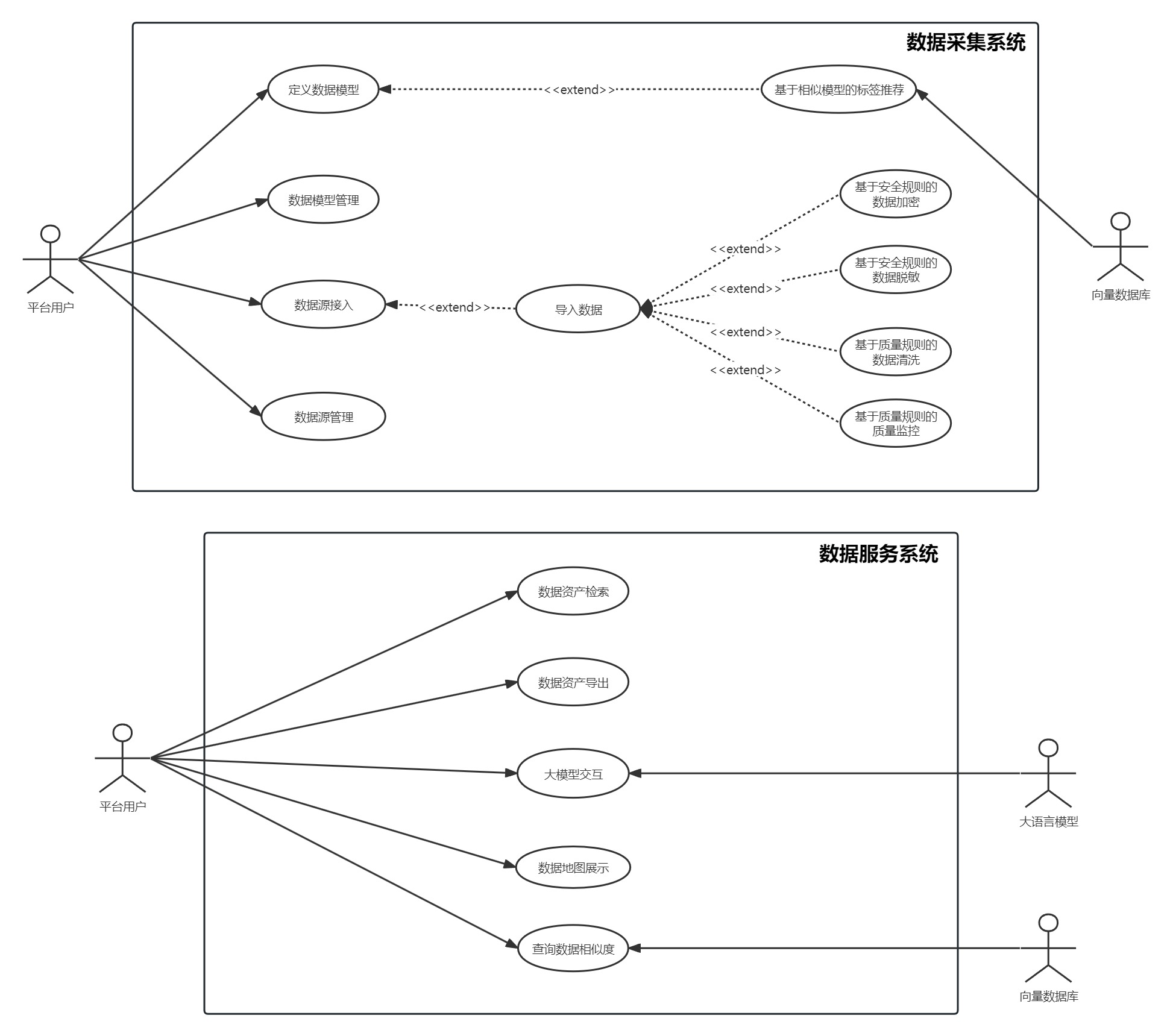
（6） 服务端开发框架及中间件：SpringBoot、MyBatis、Redis、RabbitMQ、MyCat。

（7） 服务端数据库：PostgreSQL、MongoDB、InfluxDB、Milvus、Hive。

（8） 开发工具：Intellij IDEA、PyCharm、Visual Studio Code。



## 用例视图



平台用户相关用例实现：

### 定义数据模型

用户能够根据业务类型或数据源结构来自定义数据模型，包括对标签描述、质量规则以及安全规则的定义。

### 基于相似模型的标签推荐

在定义标签描述的过程中，系统可以通过向量数据库检索相似数据模型，进而给出推荐的标签描述。

### 数据模型管理

用户能够管理已有的数据模型，包括修改模型、删除模型、查看模型详情等。

### 数据源接入

用户能够通过数据源模型的注册信息，将多模态的数据源接入平台中。对已注册的数据源，平台会完成元数据提取、映射的工作。

### 导入数据

在导入已注册数据源的数据时，根据数据源所映射数据模型的质量规则，系统会导出数据的质量监控报告。

### 基于安全规则的数据加密

在导入已注册数据源的数据时，根据数据源所映射数据模型的安全规则，系统会对具备密级的数据进行加密。

### 基于安全规则的数据脱敏

在导入已注册数据源的数据时，根据数据源所映射数据模型的安全规则，系统会对标记为敏感的数据进行脱敏。

### 基于质量规则的数据清洗

在导入已注册数据源的数据时，根据数据源所映射数据模型的质量规则，系统会对数据进行清洗。

### 基于质量规则的数据监控

在导入已注册数据源的数据时，根据数据源所映射数据模型的质量规则，系统会导出数据的质量监控报告。

### 数据源管理

用户能够管理已注册的数据源，包括映射到数据模型、修改信息、查看数据源详情等。

### 数据资产检索

用户能够对平台中的数据资产进行检索，使用包括语义标签、描述和元信息等的条件进行筛选。

### 数据资产导出

用户能够导出平台中的数据资产到自定义位置，同时在导出时也会根据数据模型中的定义对数据进行清洗和脱敏操作。

### 数据地图展示

用户能够查看数据地图来了解多模态数据之间的关系，实现跨模态的数据关联。

### 数据相似度查询

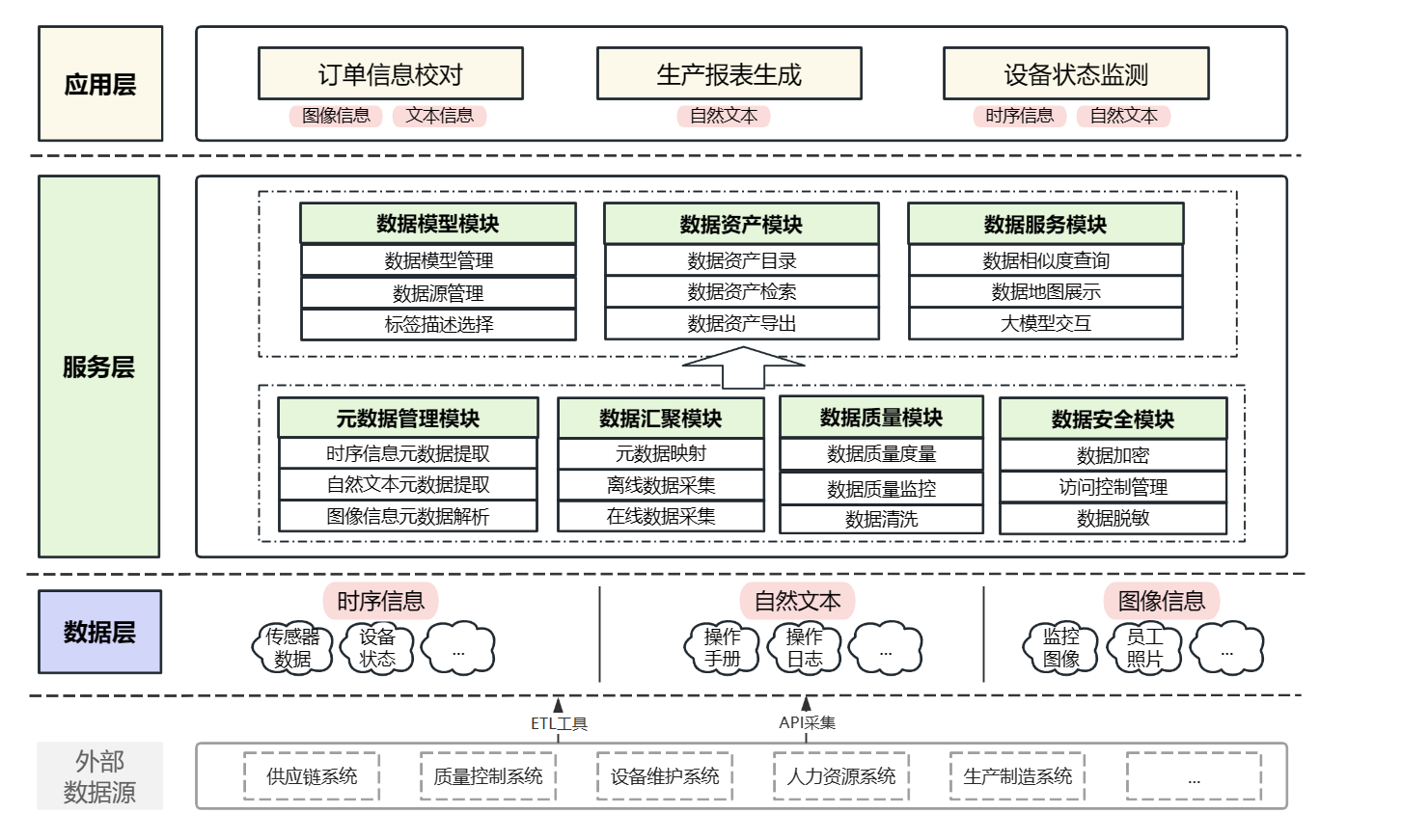
用户能够基于向量数据库进行数据相似度查询，以检索到类似数据。

### 大模型交互

用户能通过和大语言模型的交互来智能分析平台中的数据和生成定时监控脚本。

## 逻辑视图

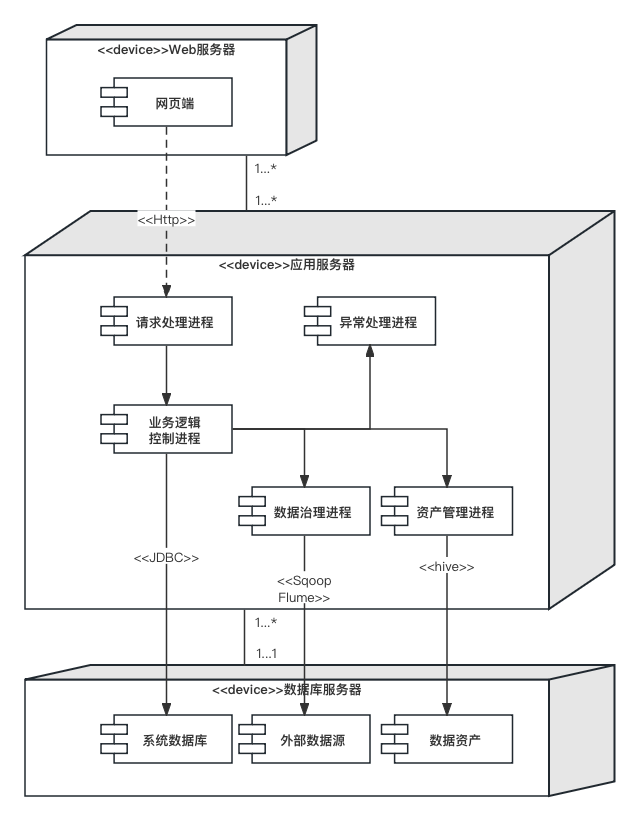
逻辑视图基于三层架构，最终设计为数据接入源、数据层、服务层和应用层。视图绘制如下：



1. 外部数据源：通过 ETL 工具和 API 采集等方式从企业的生产系统、供应链系统、质量控制系统等各类系统中接入包含时序信息、自然文本、图像信息的多模态数据，获取数据治理的原始数据。
2. 数据层：数据层包含数据源信息、数据模型、质量规则、安全密级、数据资产等业务数据，以及订单数据、设备数据、设备指令、操作手册、监控图像等用户保存的经过数据治理后的原始数据。
3. 服务层：服务层包含元数据管理、数据汇聚、数据质量、数据安全、数据模型、数据资产和数据服务七大模块。元数据管理、数据汇聚、数据质量、数据安全通过对多模态数据源的元数据提取、数据汇聚、数据质量管控和访问权限控制及数据加密和脱敏对上层三个模块进行支持。数据模型模块用户可以定义数据模型（数据模型包含数据标准、数据质量规则和数据安全密级），定义完数据模型后可以为数据模型接入数据源，并对数据模型及数据源添加标签描述便于后续检索。数据资产管理模块包含数据资产目录管理、数据资产检索和数据资产导出，让用户可以通过数据资产目录使用标签和描述快速找到数据并按需处理并导出；数据服务模块支持使用自动化脚本对数据进行监控，对数据标准生成数据地图以及使用数据生成报表。
4. 应用层：本数据治理平台可以帮助企业使用数据改善决策支持、缩减管理成本、降低数据风险和挖掘数据价值。

## 部署视图

部署视图描述了软件到硬件的映射以及物理部署的配置。它说明了如何在不同的物理节点上分配系统组件，包括数据库服务器、应用服务器和客户端工作站。本项目的部署视图绘制如下：



由上图可以看出，用户通过手机或电脑上的浏览器进行访问，通过HTTP服务器通讯，得到应用服务器上的响应。项目通过Http协议实现前后端解耦，主要分为三个部分：Web服务器、应用服务器和数据库服务器。

### Web服务器

数据中台运维人员可在Web页面进行数据源管理、数据资产管理、数据模型管理、数据地图展示等操作，通过Http协议与应用服务器交互，并实时更新应用服务器通过Websocket协议推送的消息。

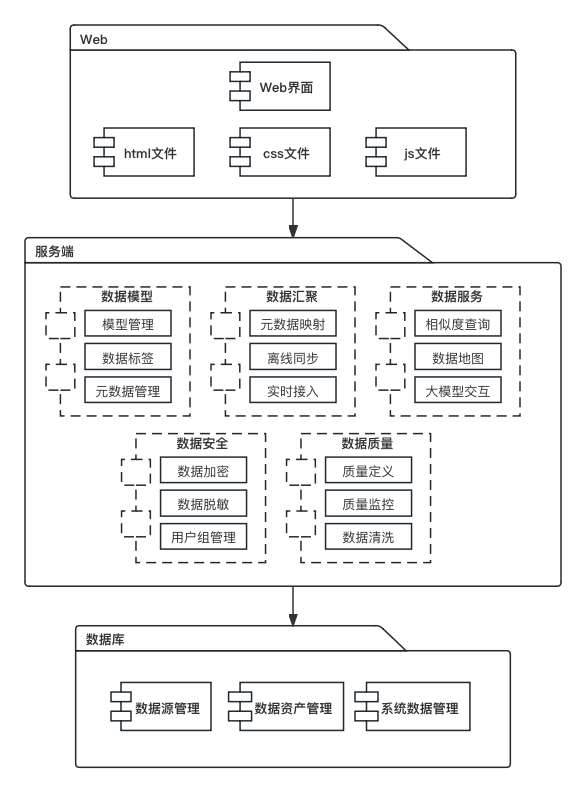
### 应用服务器

应用服务器解析前端发送的请求，通过业务逻辑控制进程、数据治理进程和资产管理进程提供相应的功能，异常处理进程处理全局异常，请求处理进程解析请求并包装回传响应信息。

### 数据库服务器

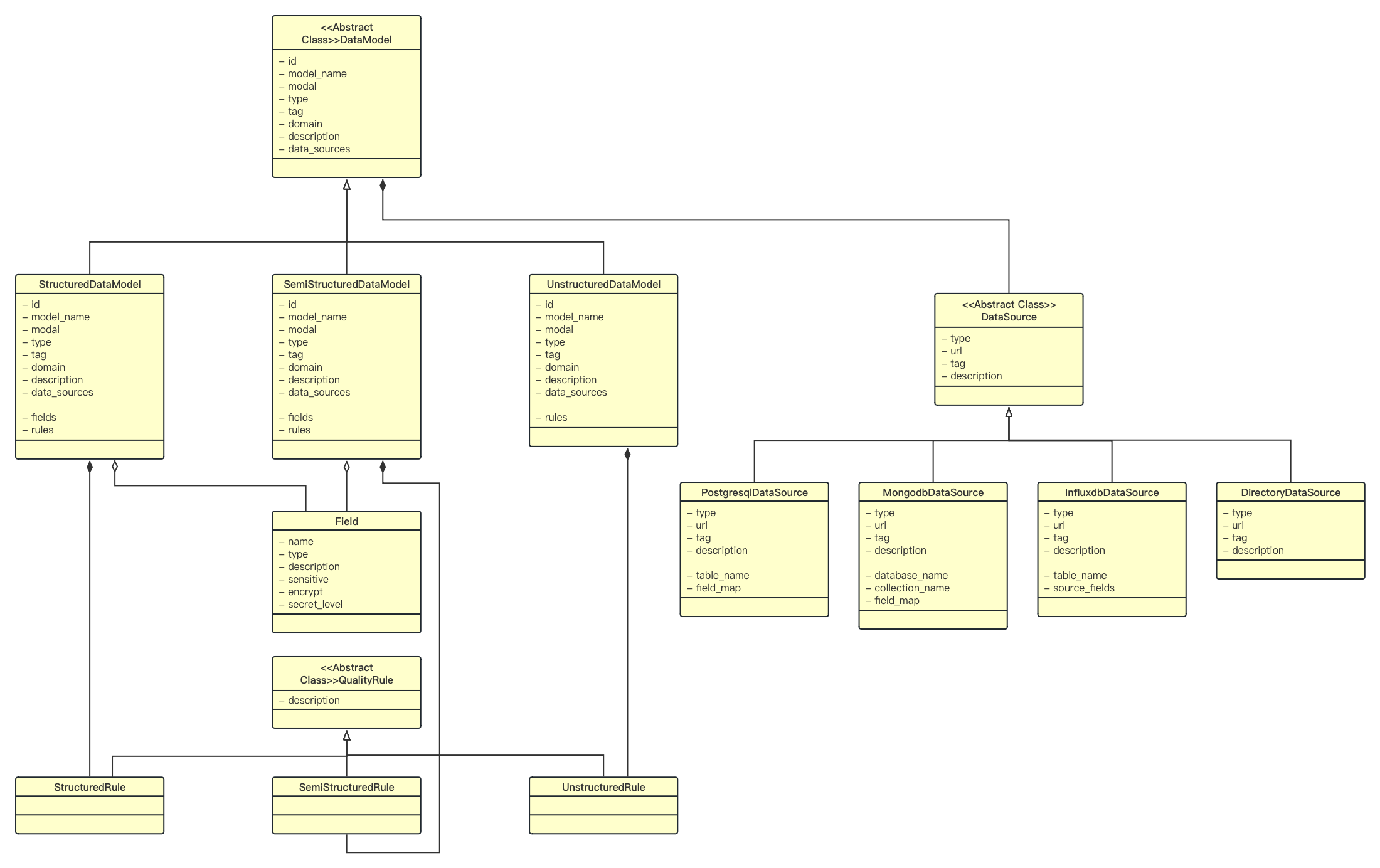
数据库服务器中外部数据源指向第三方数据库；数据资产数据库存储数据脱敏、数据加密、数据清洗阶段形成的结构化数据、半结构化数据和非结构化数据；系统数据库存储了系统自身的数据以及对应数据资产数据库的元数据。

## 开发视图



本项目采用垂直分割方案，将系统主要分为三大部分，web界面，服务端和数据库。Web使用js，html，css进行界面开发。在服务端将多个实现单元进行分类，分为数据模型、数据汇聚、数据服务、数据安全和数据质量5个子系统进行开发。数据库的功能被分为（外部）数据源管理、数据资产管理和系统数据管理分别进行设计和开发。在清晰业务逻辑和流程后，我们将定义好各个层向其上层提供的接口，再采用“桩模式”设计这些接口的桩，各个层通过调用桩而不是接口的业务实现类来实现快速调试。

# 类设计



在这个体系结构中，DataModel作为一个抽象类，为不同类型的数据模型提供了基础属性。它通过组合关系与DataSource类型连接，表示数据模型依赖于数据源。StructuredDataModel、SemiStructuredDataModel和UnstructuredDataModel分别继承自DataModel并且包含对应类型的QualityRule，表示不同类型的数据模型拥有不同的质量规则。DataSource类及其子类提供数据存储和检索的具体实现。类图中的类设计如下：

## DataModel (抽象类)

属性:

* id: 数据模型的唯一标识。
* model\_name: 数据模型的名称。
* model: 数据模型本身。
* type: 数据模型的类型。
* tag: 数据模型的标签。
* domain: 数据模型所属的领域。
* description: 数据模型的描述。
* data\_sources: 数据模型使用的数据源。

## StructuredDataModel

继承自:DataModel

组合关系:包含 StructuredRule

属性:

* fields: 字段的集合。
* rules: 质量规则的集合。

## SemiStructuredDataModel

继承自:DataModel

组合关系:包含 SemiStructuredRule

属性:

* fields: 字段的集合。
* rules: 质量规则的集合。

## UnstructuredDataModel

继承自:DataModel

组合关系:包含 UnstructuredRule

属性:

* rules: 质量规则的集合。

## Field

属性:

* name: 字段名称。
* type: 字段类型。
* description: 字段描述。
* sensitive: 是否敏感字段。
* encrypt: 是否加密。
* secret\_level: 秘密级别。

## QualityRule (抽象类)

属性:

* description: 规则描述。

## StructuredRule

继承自:QualityRule

## SemiStructuredRule

继承自:QualityRule

## UnstructuredRule

继承自:QualityRule

## DataSource (抽象类)

属性:

* type: 数据源类型。
* url: 数据源的URL。
* tag: 数据源的标签。
* description: 数据源的描述。

## PostgresqlDataSource

继承自:DataSource

属性:

* table\_name: 表名。
* field\_map: 字段映射。

## MongodbDataSource

继承自:DataSource

属性:

* database\_name: 数据库名。
* collection\_name: 集合名。
* field\_map: 字段映射。

## InfluxdbDataSource

继承自:DataSource

属性:

* table\_name: 表名。
* source\_fields: 源字段集合。

## DirectoryDataSource

继承自:DataSource

# 测试方案

## 功能性测试

1. 对用例模型进行功能依从性测试，覆盖每个用例的各个基本流和备选流；

2. 测试软件的安全性，包括用户组权限、数据加密过程等；

3. 测试软件与外部数据源、大模型交互能力；

4. 测试软件是否提供具有所需精度的正确或相符的结果或效果的能力；

## 可靠性测试

1. 测试软件在出现故障时维持规定性能级别的能力，在项目中规定系统的可用时间百分比应大于 95% ；

2. 在失效发生情况下，软件重建规定的性能级别并恢复受到直接影响的数据的能力，在项目中规定数据在处理流程中的故障为严重故障，其修复时间不超过2小时，其他故障修复时间不超过1小时；

3. 测试软件遵循与可靠性相关的规约、标准或法规的能力，规定系统平均故障间隔时间应大于 30\*24小时；

## 性能测试

1. 测试软件执行功能时的响应和处理时间能力，在项目中规定对事务的平均响应时间不超过 1.5s，最长响应时间不超过 7s；

2. 测试软件遵循性能相关的规约、标准或法规的能力；

## 易用性测试

1. 测试软件被用户理解、学习其应用的能力；

2. 测试软件吸引用户的能力；

## 可维护性测试

1. 测试软件诊断缺陷或失效原因或识别待修改部分的能力；

2. 测试软件将指定的修改实现的能力；

3. 测试软件避免因修改而造成其他意外结果的能力；

4. 测试软件确认已修改的能力；

# 大小和性能

1. 对事务的平均响应时间不超过 1.5s，最长响应时间不超过 7s
2. 系统的可用时间百分比应大于 95%
3. 系统的平均修复时间应小于 2h
4. 系统平均故障间隔时间应大于 30\*24h

# 质量

1. 系统 PC 端支持各个主流操作系统（至少包括 Windows，Mac OS）通过 Chrome 浏览器进行访问；
2. 开发语言为 Java，Javascript、CSS 和 HTML，技术相对成熟。