中国物品编码中心

一体化开发平台设计与建设规划总体标准及规范手册

V1.0

目录

[1. 建设目标 - 2 -](#_Toc18253)

[2. 总体框架 - 2 -](#_Toc13193)

[3. 总体标准 - 2 -](#_Toc24905)

[3.1 标准规范体系 - 2 -](#_Toc17952)

[3.2 业务术语 - 4 -](#_Toc15619)

[4. 需求开发标准 - 6 -](#_Toc22804)

[4.1 需求评审上线流程 - 6 -](#_Toc28645)

[4.1.1 流程示意图 - 6 -](#_Toc25304)

[4.1.2 常规流程说明 - 7 -](#_Toc14973)

[4.2 应急规则 - 15 -](#_Toc24068)

[5. 数据标准 - 15 -](#_Toc62)

[5.1 数据库设计 - 16 -](#_Toc3728)

[5.1.1 设计原则 - 16 -](#_Toc29260)

[5.1.2 设计过程规范 - 16 -](#_Toc4596)

[5.2 数据仓库设计 - 19 -](#_Toc11667)

[5.2.1 基本分层结构 - 19 -](#_Toc6492)

[5.2.2 数据仓库的开发方法及步骤 - 19 -](#_Toc20160)

[5.3 数据库设计命名规范 - 24 -](#_Toc26182)

[6. 技术标准 - 25 -](#_Toc21203)

[6.1 技术架构规范 - 25 -](#_Toc9539)

[6.1.1 开放性 - 25 -](#_Toc4444)

[6.1.2 扩展性 - 27 -](#_Toc24514)

[6.1.3 安全性 - 27 -](#_Toc17839)

[6.1.4 可靠性 - 27 -](#_Toc17660)

[6.2 性能设计规范 - 28 -](#_Toc1672)

[6.2.1 性能指标 - 28 -](#_Toc19327)

[6.2.2 可靠性指标 - 28 -](#_Toc18420)

[7. 接口标准 - 30 -](#_Toc30709)

[7.1 接口要求 - 30 -](#_Toc8582)

[7.2 技术方式 - 30 -](#_Toc15111)

[7.3 报文格式 - 31 -](#_Toc8984)

[8. 代码开发标准 - 32 -](#_Toc10138)

[8.1 管理规范 - 32 -](#_Toc5287)

[8.1.1 项目立项 - 32 -](#_Toc2998)

[8.1.2 项目技术调研 - 32 -](#_Toc8722)

[8.1.3 产品原型和设计评审 - 32 -](#_Toc2515)

[8.1.4 需求变更 - 32 -](#_Toc14783)

[8.1.5 计划管理 - 32 -](#_Toc32099)

[8.1.6 任务和计划执行跟进 - 33 -](#_Toc23919)

[8.1.7 系统设计 - 33 -](#_Toc3278)

[8.1.8 代码编写 - 33 -](#_Toc24278)

[8.1.9 系统测试和Bug修复 - 34 -](#_Toc7235)

[8.1.10 系统运维 - 34 -](#_Toc3491)

[8.1.11 项目资源管理 - 34 -](#_Toc12288)

[8.2 开发环境 - 34 -](#_Toc19232)

[8.2.1 操作系统环境 - 35 -](#_Toc521)

[8.2.2 开发工具 - 35 -](#_Toc22429)

[8.2.3 版本/代码管理工具 - 35 -](#_Toc28262)

[8.2.4 代码管理服务 - 35 -](#_Toc4144)

[8.3 开发框架 - 36 -](#_Toc12882)

[8.3.1 整体结构规范 - 36 -](#_Toc18334)

[8.3.2 开发框架使用 - 37 -](#_Toc7697)

[8.3.3 代码结构规范 - 37 -](#_Toc31592)

[8.3.4 模块命名规范 - 37 -](#_Toc5697)

[8.3.5 配置文件规范 - 37 -](#_Toc28234)

[8.4 代码规范 - 38 -](#_Toc26849)

[8.5 数据库规范 - 38 -](#_Toc15284)

[9. 应用框架规范 - 39 -](#_Toc23398)

[10. UI设计风格规范 - 40 -](#_Toc21504)

引 言

为深入贯彻中共中央、国务院《关于加强中国特色新型智库建设的意见》，全面落实集团公司“一个目标、三型五化、七个一流”总体发展战略，在《国家能源集团技术经济研究院建设一流智库行动计划（2021-2025年）》的总体指导下，按照集团公司《关于加强网络安全和信息化工作的指导意见》的工作要求，通过技术中台标准体系建设，为技经院智能化一流智库一体化平台建设与管理奠定基础，为后续其他课题建设提供指导。

# 建设目标

技术中台标准体系框架参考国标、行标、借鉴领先企业实践经验，并在融合优化集团已有标准体系成果的基础上，以构建“平台化、组件化、标准化”的业务应用为设计原则，承载智库平台整体的技术中台规范体系建设，包括数据规范、技术标准、接口标准、应用框架规范等中台相关技术规范。

# 总体框架

智能化一体化平台的标准体系主要分为总体标准、技术标准、应用框架规范三大类。信息化标准规范体系总体架构,如下图所示。

# 总体标准

总体标准是技经院智能化一流智库建设所需的总体性、通用性的标准和规范，应满足技经院智能化一流智库建设的总体设计、总体规划的要求，包括标准规范体系和业务术语。

## 标准规范体系

本次标准规范体系规定了技经院智能化一体智库标准规范框架及明细表。

|  |  |  |
| --- | --- | --- |
| 序号 | 规范标准分类 | 名称 |
| 1 | 数据库设计规范 | 应用系统数据库设计 |
| 2 | 数据仓库设计 |
| 3 | 数据库设计命名规范 |
| 4 | 大数据应用规范 | 分布式计算 |
| 5 | 分布式存储 |
| 6 | 数据采集 |
| 7 | 数据加工 |
| 8 | 数据挖掘 |
| 9 | 数据可视化 |
| 10 | 数据服务 |
| 11 | 数据沙箱 |
| 12 | 在线分析 |
| 13 | 数据分类 | 指标数据 |
| 14 | 主数据 |
| 15 | 通用基础数据 |
| 16 | 数据资源编码规范 | 指标数据编码 |
| 17 | 主数据编码 |
| 18 | 通用基础数据编码 |
| 19 | 技术标准 | 技术架构规范 |
| 20 | 性能设计规范 |
| 21 | 性能指标 |
| 22 | 可靠性指标 |
| 23 | 接口标准 | 技术方式 |
| 24 | 报文格式 |
| 25 | 安全管理 |
| 26 | 应用框架规范 | 应用技术分层架构 |
| 27 | 框架技术选型 |
| 28 | 总体应用要求 |
| 29 | UI设计风格规范 | 移动端设计规范 |
| 30 | PC端设计规范 |

## 业务术语

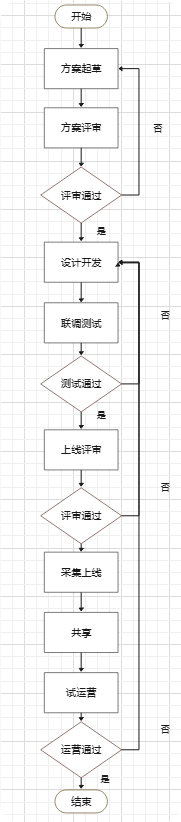
该类标准规定了技经院核心业务相关术语，包含选词原则、术语全称、简称、解释等。

* 智库：又称头脑企业、智囊集团或思想库、智囊机构、顾问班子，是指专门从事开发性研究的咨询研究机构。
* 研究项目：指通过中央和国务院相关部委及研究单位，以及集团公司、技经院或其他单位在科研计划或经费计划中安排，由技经院承担或参与，通过技术服务、技术咨询等合同（计划任务书）约定，并在规定时间内实施的软科学研究、技术开发、工业性试验、高技术产业化和成果应用等活动。
* 项目评估：通过集团公司相关业务职能部门委托，由技经院承担或参与，在规定时间内实施的评估咨询活动。
* 评估报告：评估报告是在尊重客观事实和科学规律基础上，有确切的评判依据和深入的论证分析，做到科学、客观、公正。
* 评估档案：评估档案包括项目委托书、可研报告及相关附件、专家意见、项目评审纪要、评估报告、项目完成反馈意见书以及其他相关资料。
* 评估专家费用：指在评估类项目评估过程中支付给临时聘用专家的咨询费。
* 情报服务：情报服务是情报机构为用户提供情报的工作，主要目的帮助用户克服搜集、选择过程中的困难，及时获取情报信息。
* 情报产品：是经过情报劳动加工而生产出来的产品，主要包括日报、周报、月报、季报、年报和专题专报等。

# 需求开发标准

## 需求评审上线流程

### 流程示意图



### 常规流程说明

#### 步骤1 方案起草

|  |  |
| --- | --- |
| **步骤1** | **方案起草** |
| **流程描述** | 1、需求收集和分析阶段，形成初步方案；  2、确定需求的优先级和时间表； |
| **要点** | 1、收集并分析业务需求；  2、起草初步需求方案，包括需求说明、目标、范围以及初步设计；  3、会议讨论确定需求优先级和时间表 |
| **角色与职责** | 产品经理：负责需求收集和初步方案起草  业务相关人员：协助需求分析，提供业务背景和建议 |
| **输出物** | 1、需求文档（需求说明书、需求规格说明书）  2、初步方案文档（需描述包含目标、范围、时间计划等关键要素） |
| **异常处理** | 无 |
| **备注** | 无 |

#### 步骤2方案评审

|  |  |
| --- | --- |
| **步骤2** | **方案评审** |
| **流程描述** | 1、需求方案的评审和确认；  2、确保所有项目参与方对方案达成一直的理解与认可 |
| **要点** | 1、组织方案评审会议  2、收集评审意见与反馈，进一步修改和完善方案 |
| **角色与职责** | 产品经理：组织评审会议，收集反馈  业务相关人员：确认业务需求是否被准确理解和反映  技术架构师：评审技术可行性  开发负责人：评估开发工作量和时间表 |
| **输出物** | 1、评审会议记录  2、修改后的需求方案文档 |
| **异常反馈** | 若评审意见分歧较大，需组织进一步讨论或寻求高层决策。对于重大需求认知偏差情况需重新进行方案起草。 |
| **备注** | 无 |

#### 步骤3 设计开发

|  |  |
| --- | --- |
| **步骤3** | **设计开发** |
| **流程描述** | 1、根据确认的需求方案进行详细设计和开发 |
| **要点** | 1、制定详细设计文档  2、进行代码开发  3、定期进行代码评审和质量检查 |
| **角色与职责** | 开发人员：负责代码开发和实现  技术架构师：提供设计指导和技术支持  测试人员：准备测试用例和测试环境 |
| **输出物** | 1、详细设计文档  2、源代码  3、单元测试报告 |
| **异常反馈** | 如果开发过程中发现需求变更或技术难题，需及时反馈给产品经理和技术架构师。 |
| **备注** | 无 |

#### 步骤4 联调测试

|  |  |
| --- | --- |
| **步骤4** | **联调测试** |
| **流程描述** | 1、进行系统集成测试，确保各模块之间的兼容性和正确性  2、发现并解决系统中的问题和缺陷 |
| **要点** | 1、进行模块集成和系统测试  2、收集并记录测试结果和缺陷报告 |
| **角色与职责** | 测试人员：负责测试执行和缺陷报告  开发人员：修复测试中发现的缺陷  测试经理：协调测试进度和资源 |
| **输出物** | 1、测试报告  2、缺陷报告  3、修复后的代码 |
| **异常反馈** | 测试中发现重大缺陷时，需紧急修复，并进行相应的回归测试 |
| **备注** | 无 |

#### 步骤5上线评审

|  |  |
| --- | --- |
| **步骤5** | **上线评审** |
| **流程描述** | 1、对系统进行上线前的最终评审，确保系统准备就绪  2、确认所有的测试已通过，系统达到上线标准 |
| **要点** | 1、组织上线评审会议  2、检查系统的准备状态和上线计划  3、收集并确认评审意见 |
| **角色与职责** | 产品经理：组织评审会议  技术架构师：确认技术准备情况  测试经理：提供最终测试报告  运维人员：确认系统上线准备 |
| **输出物** | 1、上线评审会议记录  2、上线准备确认书 |
| **异常反馈** | 如果评审中发现系统未达到上线标准，需返回设计开发步骤进行进一步完善开发工作，上线计划需根据评审意见进行调整 |
| **备注** | 无 |

#### 步骤6 采集上线

|  |  |
| --- | --- |
| **步骤6** | **采集上线** |
| **流程描述** | 1、系统正式上线，进行实际环境的部署和配置  2、确保系统在生产环境中正常运行 |
| **要点** | 1、部署系统到生产环境  2、配置相关服务和组件  3、进行上线后的初步验证测试 |
| **角色与职责** | 运维人员：负责系统部署和配置  开发人员：协助解决上线过程中的技术问题  测试人员：进行上线后的初步验证测试 |
| **输出物** | 1、上线部署报告  2、验证测试报告 |
| **异常反馈** | 如果上线不成功，则按照既定的回退方案进行回退 |
| **备注** | 无 |

#### 步骤7共享

|  |  |
| --- | --- |
| **步骤7** | **共享** |
| **流程描述** | 1、系统上线后，向相关方共享上线信息和使用指南  2、确保用户了解系统的功能和使用方法 |
| **要点** | 1、编写并分发上线通知  2、提供系统使用手册和培训材料  3、组织用户培训和答疑 |
| **角色与职责** | 产品经理：编写上线通知和使用指南  技术支持：提供技术培训和支持  用户代表：参加培训并提供反馈 |
| **输出物** | 1、上线通知  2、系统使用手册  3、培训材料 |
| **异常反馈** | 无 |
| **备注** | 无 |

#### 步骤8试运营

|  |  |
| --- | --- |
| **步骤9** | **试运营** |
| **流程描述** | 1、系统在正式运营前进行试运营，收集用户反馈和系统性能数据。  2、发现并解决试运营中的问题，确保系统稳定运行 |
| **要点** | 1、一般需求试运行期为7天，以具体需求而定，可以账期为单位  2、监控系统性能和用户反馈  3、记录并解决试运营中的问题  4、根据试运营结果进行系统优化 |
| **角色与职责** | 运维人员：监控系统性能  技术支持：收集并处理用户反馈  产品经理：分析试运营数据，提出优化建议 |
| **输出物** | 1、试运营报告  2、问题处理记录  3、系统优化方案 |
| **异常反馈** | 如果试运营中发现重大问题，需紧急修复并重新进行相关部分设计开发。 |
| **备注** | 1、提供连续7天的试运营报告，包括数据稽核的结果（及时性、完整性、准确性、一致性）  2、提供连续7天的监控报表  3、提供连续7天的故障报告记录 |

## 应急规则

* **启动应急流程前置条件**：

1. 紧急需求上线，需先邮件与质量处相关负责人确认，确认通过后才可进入上线流程；
2. 需求实施方案等相关资料中不影响上线的内容可以后置，保障上线所需必要内容具备，在上线后2周内各域负责人需进行资料补齐；

* **应急流程具体机制**：

1. 轮调选择项目应急人员机制：轮调选择应急工作人员，要求其能够在轮调日内24/7待命，迅速响应和处理问题；
2. 应急沟通机制：规定建立钉钉工作紧急情况报警群，通知相关负责人员，要求不允许发送紧急情况以外信息；
3. 快速决策流程：高层管理人员在紧急情况发生时有权临时调配资源，底层人员在紧急情况发生时可以跨级进行汇报；
4. 备用计划机制：在相关重要节点，应有相关负责人制定备用计划并知晓相关工作人员。

# 数据标准

数据标准制定是一体化开发平台建设的重要基础性工作，其目的是为编码中心核心业务发展提供统一的信息视图、数据规范及编码标准，实现跨应用、跨业务协同的信息共享，为战略研究、项目评价、信息服务三大主业提供支撑，助力编码中心向一体化快速迈进。

结合当前实践的研究，数据库设计规范分类分为面向应用系统的关系型数据库的数据和面向数据仓库的数据。关系型数据是指面向关系型数据库的数据，海量规模数据是面向数据仓库存储的数据。数据库标准+数据仓库标准可满足面向未来大数据应用、编码中心各类应用。

## 数据库设计

### 设计原则

* 按阶段实施并形成项目阶段的成果物
* 符合3NF式要求，同时兼顾合规与效率
* 使用编码中心内规定的数据库设计软件工具
* 命名符合编码中心内的标准和规范

### 设计过程规范

#### 数据分析阶段

在数据分析阶段，应注意搜集和分析数据相关的容，并形成相关成果物，包括数据流图和数据字典等，以此作为数据库设计的根底和依据。数据流图从数据传递和加工的角度，以图形的方式刻画数据流从输入到输出的移动变换过程。数据字典是对数据流图中的各种成分进展详细说明，作为数据流图的细节补充。数据字典一般应包括对数据项，数据结构、数据流、数据存储和处理过程的说明。

数据项描述={数据项名，数据项含义说明，别名，数据类型，长度，取值范围，取值含义，与其他数据项的逻辑关系}；

数据结构描述={数据结构名，含义说明，组成：{数据项或数据结构}}；

数据流描述={数据流名，说明，数据流来源，数据流去向，组成：{数据结构}，平均流量，高峰期流量}；

数据存储描述={数据存储名，说明，编号，流入的数据流，流出的数据流，组成：{数据结构}，数据量，存取方式}；

处理过程描述={处理过程名，说明，输入：{数据流}，输出：{数据流}，处理：{简要说明}}。

#### 概念设计阶段

* 目的

在数据分析的根底上，使用E-R模型技术，将现实世界中的客观对象抽象为实体和关系，形成概念数据模型〔CDM〕。CDM具有较强的语义表达能力，能够方便、直接地表达应用中的各种语义知识，应该简单、清晰、易于用户理解，是用户与数据库设计人员之间进行交流的语言。

* 方法

有自顶向下、自底向上、逐步扩张、混合策略的方法。

* 概念设计过程

集成

抽象

全局E-R模型

局部E-R模型

需求分析数据

第一步：数据抽象与局部E-R模型设计

1. E-R方法：建立E-R图；
2. 数据抽象：在多层数据流图中选择一个适当层次作为设计E-R图的出发点；确定每个局部应用包含哪些实体，实体包含哪些属性，实体之间的联系；划分实体和属性。

实体和属性划分的检验标准：

|  |  |  |  |
| --- | --- | --- | --- |
| 构成要素 | 检验基准 | 构成要素 | 检验基准 |
| 核心实体 | 集合性 | 核心关系 | 可选性 |
| 识别性 | 关系的度 |
| 永久性 | 业务指向性 |
| 使用性 | 核心属性 | 原子性 |
| 关系性 | 含义性 |

1. 局部E-R模型设计：

第二步：全局E-R模型设计

1. 集成各局部E-R模型，形成全局模型；
2. 集成的方法：多元集成发、二元集成法。

#### 逻辑设计阶段

* 目的

将E-R模型转换DBMS(数据库管理系统)支持的数据模型，逻辑模型要以数据为中心进行设计,并且必须在充分理解业务需求的基础上进行设计。设计的逻辑模型要具有扩展性强、一致性高、共享性强等特点。设计逻辑模型不仅仅是为了构建数据库，更重要的是起到对业务进行细致分析、准确定义的作用。

* 逻辑设计过程

逻辑模型整合

属性定义及范式化

实体选定及关系定义

从理论上来说，转换过程一般有7个步骤：

* 转换强实体
* 转换弱实体
* 转换1：1关系
* 转换1：N关系
* 转换M：N关系
* 转换多值属性〔Multi-Valued Attribute〕
* 转换n元关系〔n-ary Relation〕

E-R模型和关系模型的映射如下：

|  |  |
| --- | --- |
| **E-R模型** | **关系模型** |
| 实体类型 | 关系 |
| 1：1或1：N关系类型 | 外键 |
| M：N关系类型 | 两个外键 |
| n元关系类型 | n个外键 |
| 简单属性 | 属性 |
| 复合属性 | 简单属性的集合 |
| 多值属性 | 关系和外键 |
| 关键属性 | 主键/唯一键 |

#### 物理设计阶段

* 目的

基于给定的DBMS建立面向计算机物理表示的模型，描述了数据在储存介质上的组织结构，它不但与具体的DBMS有关，而且还与操作系统和硬件有关。

* 方法和过程

可以用建模工具直接将LDM转换为PDM。对于关系模型来说，进展物理数据库设计主要包括：

* 使用逻辑模型建立一系列的表〔如果在CDM和LDM中使用中文，应在转换后使用英文〕；
* 使用索引以提升性能；
* 实施约束和安全限制；
* 对数据进展分区和分布式处理等。

## 数据仓库设计

### 基本分层结构

本项目数据仓库应从存储的内容可分为STAGE层（接口层）、DWD层（细节沉淀层）、DWA层（汇总衍生层）、DM层（数据应用层）。

STAGE层：生产系统数据源的直接拷贝，由数据采集过程对数据源进行直接抽取，在格式和数据定义上不作任何改变。提供业务系统数据文件的临时存储，数据稽核，数据质量保证，屏蔽对业务系统的干扰。

DWD层：数据仓库的细节数据层，对STAGE层数据进行沉淀，减少抽取的复杂性，将数据进行集中。系统将数据按业务域分类存放，跟STAGE层的粒度一致，属于分析的公共资源。

DWA层：数据仓库的衍生汇总数据层，该层通过对DWD层数据进行处理（轻度汇总、衍生），提高后续数据处理和访问性能，其特点是面向应用但不直接支持应用，将应用过程中的常用信息进行共同沉淀和处理，与DWD库共同构成企业级数据仓库。

DM层：是一个主题的、集成的、相对稳定的、反映历史变化的数据集合，主要用于面向业务分析应用场景的数据应用。

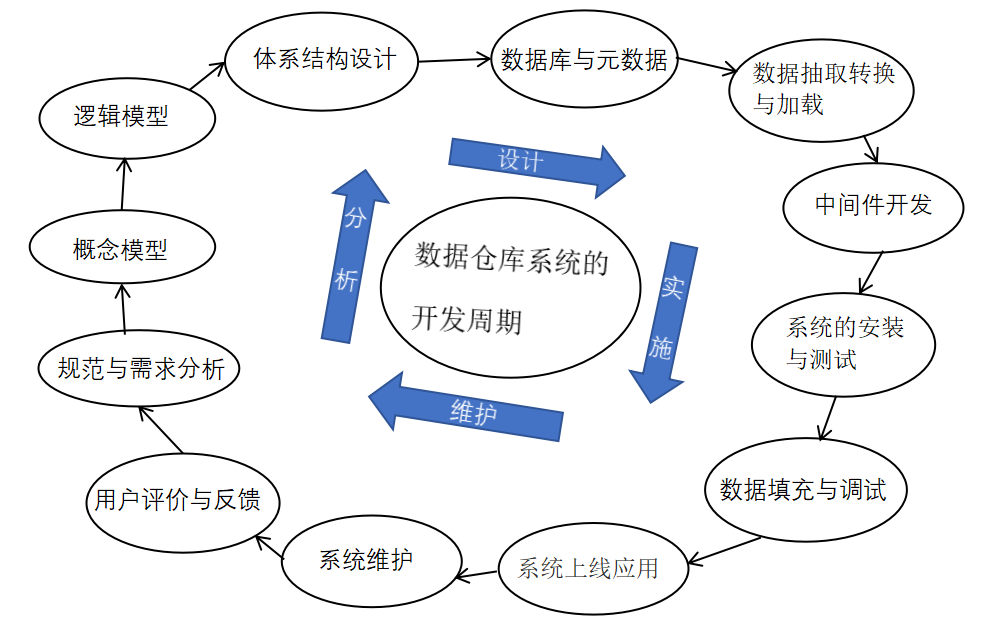
### 数据仓库的开发方法及步骤

#### 数据仓库开发周期

数据仓库的开发应用具有其特有的、完整的生命周期，其周期可以分成:

* 数据仓库规划分析阶段
* 数据仓库设计实施阶段
* 数据仓库的使用维护阶段

这三个阶段是一个不断循环、完善、提高的过程。在一般情况下数据仓库系统不可能在一个循环过程中完成，而是经过多次循环开发，每次循环都会为系统增加新的功能，使数据仓库的应用得到新的提高。



#### 数据模型设计方法

数据模型设计采用自顶向下与自底向上相结合的方法进行设计。

以企业级数据模型为指导，结合生产系统相关规范，参考业界标杆SID、行业成熟模型，先建立概念模型，再在概念模型的基础上进行细化设计逻辑模型，物理实现时依据逻辑模型针对具体的分析需求和物理平台采取相应的优化策略。

* 3NF设计方法

三范式建模是一种消除数据冗余的设计方法，数据被分成很多离散的实体，每一个实体在关系数据库中都对应一个数据表，可以通过实体来存储与这些特征有关的信息。

* 维度建模方法

维度事实建模将客观世界划分为事实和维度，事实是由机构部门的业务过程和支持他们的业务源系统获得的，以数值形式出现并且具有可加性。

#### 数据模型建模原则

为保证模型的稳定性和对业务支持的灵活性，建模阶段将遵循以下的原则：

继承性原则：以现有用户应用系统数据模型规范作为基础，在不影响理解情况下，尽量不提出新的概念；

稳定性原则：为保证模型的稳定性，实体与规则分离，突出核心实体的描述，提出规则点，对规则本身不做详尽描述；

前瞻性原则：为保证模型的前瞻性，同时采用自底向上和自顶向下的方式设计模型，其中自顶向下主要基于业务需求进行模型设计，使其可以完全覆盖到所有需求，自底向上主要基于业务逻辑而非业务需求进行模型设计，模型设计为囊括用户现有的各种业务关系，保证在有新需求时，底层模型能够对其进行支撑；

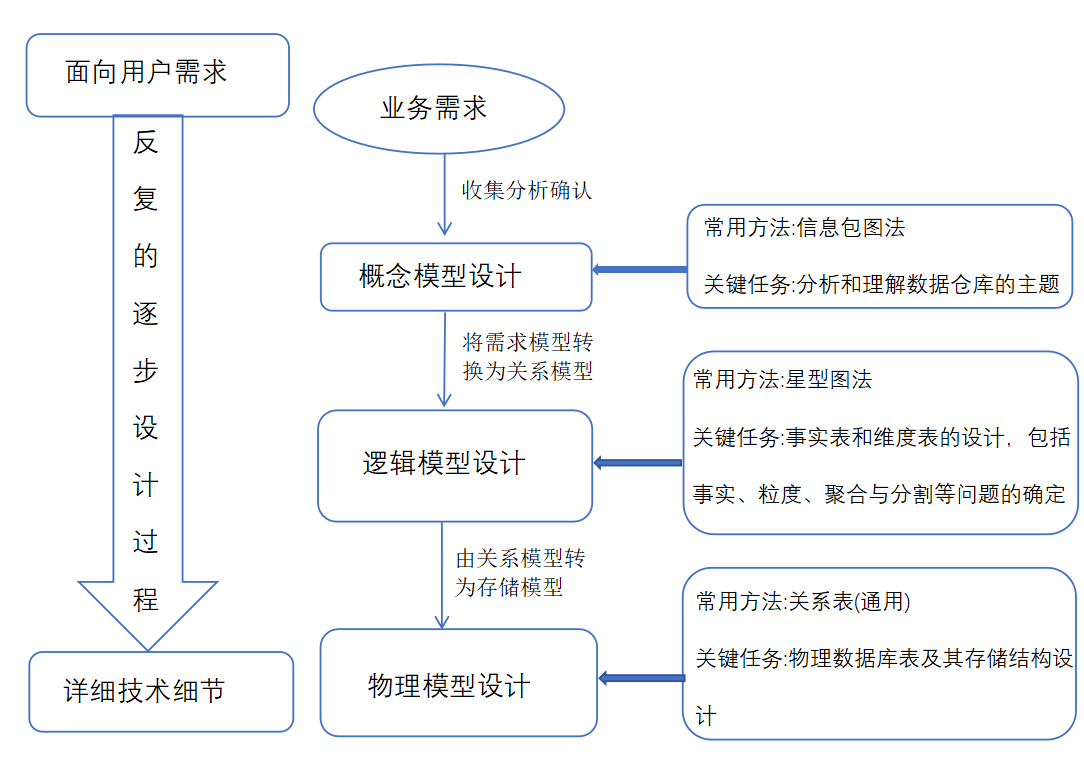
兼顾实际原则：在模型设计时针对当前用户系统建设的实际情况进行调研，保障模型能正常落地；

扩展性原则：模型的扩展包括实体属性的扩展和实体关系的扩展，模型设计为实体内只保留最细粒度的基本属性，粗粒度或上层的属性通过雪花结构的属性依赖关系实体来表现，这样在扩充属性或者扩充实体关系时，只增加表现属性依赖关系的实体即可。

#### 数据初始化策略

由于数据仓库的复杂性，针对不同的数据源采用不同的初始化策略。数据初始化后，系统运行一段时间，可能因为维护加载过程中的异常导致平台中数据与源端数据不一致，影响分析结果及对内决策的准确性。为保证核心数据准确性、一致性，应采取对账的手段保证数据加载过程的准确性，保证数据维护输入与输出端的一致性。

#### 数据仓库的设计步骤



数据仓库的设计主要分为以下步骤：

* 业务需求：分析组织的业务状况及数据源结构，进行需求调研并收集分析需求。
* 概念模型设计：

1. 界定系统边界

从某种意义上讲，界定系统边界的工作也可以看作是数据仓库系统设计的需求分析，它将决策者的数据分析的需求用系统边界的定义形式反映出来。

1. 确定主要的主题域及其内容

在这一步中，要确定系统所包含的主题域，然后对每个主题域的内容进行较明确的描述，描述的内容包括：

* 主题域的公共码键
* 主题域之间的联系
* 充分代表主题的属性组
* 逻辑模型设计：

1. 分析主题域，确定当前要装载的主题

数据仓库的设计方法是一个逐步求精的过程，在进行设计时，一般是一次一个主题或一次若干个主题地逐步完成的。从得出的所有主题中选择首先要实施的主题域。

1. 确定粒度层次划分

粒度层次划分适当与否直接影响到数据仓库中的数据量和所适合的查询类型。确定数据仓库的粒度划分，可以通过估算数据的总行数来确定是采用单一粒度还是多重粒度，以及粒度划分的层次。

1. 确定数据分割策略

要选择适当的数据分割的标准，一般要考虑以下几方面因素:数据量(而非记录行数)、数据分析处理的实际情况、简单易行以及粒度划分策略等。

* 数据量的大小是决定是否进行数据分割和如何分割的主要因素；
* 数据分析处理的要求是选择数据分割标准的一个主要依据，因为数据分割是跟数据分析处理的对象紧密联系的；
* 考虑数据分割的标准与粒度划分层次是适应的。

1. 关系模型定义

数据仓库的每个主题都是由多个表来实现的，表之间依靠主题的公共码键联系在一起，形成一个完整的主题。

* 物理模型设计

1. 确定数据的存储结构

一个数据库管理系统往往都提供多种存储结构，不同的存储结构有不同的实现方式，在选择合适的存储结构时应该权衡三个方面的主要因素：存取时间、存储空间利用率和维护代价。

1. 确定索引策略

数据仓库的数据量很大，数据都是不常更新的，因而可以设计多种多样的索引结构来提高数据存取效率。

1. 确定数据存放位置

同一个主题的数据并不要求存放在相同的介质上。常常要按数据的重要程度、使用频率以及对响应时间的要求进行分类，并将不同类的数据分别存储在不同的存储设备中。

1. 确定存储分配优化

许多数据库管理系统提供了一些存储分配的参数如：数据块的尺寸、缓冲区大小和个数等。

## 数据库设计命名规范

* 数据库文件名规范

命名方式：系统名\_文件名

* 用户

用户名的名称应该采用同系统应用相似的英文字符或字符缩写。从逻辑上分出的各数据层应当在数据库中用户实现其分离，规定如下：

* 数据库分区表规范

原则上数据仓库中的全量数据表都要建立分区（码表除外）。

* 数据库表索引规范

索引名命名方式：IDX\_表名\_字段名

* 数据库字段名规范

数据库字段命名按照对应英文单词大写英文字母定义，英文单词中间以下划线来隔开，如果英文单词较多则只取代表主要含义的单词，英文字母字数较多的一般截取前四位来简写。

* 数据库存储过程规范

（1）存储过程命名规则：“P\_”加上实体名称。

（2）存储过程要求有注释，注释内容为：列出创建人，创建用途，创建时间。

（3）存储过程日志规范：

每一存储过程均应记录执行存储过程的日志信息。必须调用专用写日志的存储过程，同时有exception时的处理机制。

（4）存储过程修改规范

修改时应注释清楚修改人，修改日期，修改原因和修改内容。

* 数据库函数命名规范

函数命名规则：F\_<功能名>

* 物理模型命名规范

（1）STAGE层

STAGE层接口表分各系统内部数据接口层和互联网爬虫数据两部分，两部分所有表的命名保持一致。

（2）DWD层

实体命名为前缀“DWD\_”加上“业务域代码\_分类代码\_内容描述”。

（3）DWA层

实体命名为前缀“DWA\_”加上“主题代码\_子类代码\_内容描述”。

（4）DM层

实体命名为前缀“DM\_”加上“应用代码\_内容描述”。

（4）临时表

测试或临时处理手动创建的表，为方便管理，统一命名规范

表名：T\_姓名缩写\_表标识[\_时间]

（5）中间过程表

为方便数据处理，临时创建的表

表名：数据分层前缀\_MID\_表标识

# 技术标准

## 技术架构规范

技术架构应采用自主掌握的技术和国产化信创软件，遵循开放性、扩展性、安全性和可靠性原则，提升架构自主掌控和灵活配置能力。

### 开放性

开放性原则主要是从业务、技术、管理等方面充分考虑技术架构的开放性规范。

#### 业务开放性

在系统开放性设计中，为了最大限度地增强系统的价值，最大限度地吻合各业务应用的需求，通过对业务深入理解，对业务的发展方向进行预判，整个系统设计过程始终遵循面向业务价值，围绕系统应用，依靠业务部门，注重实效的方针，保证业务的开放性，满足业务需求不断发展变化的要求，便于应用系统的升级。

#### 技术开放性

技术开放性需要从组件化结构、标准化结构、开放功能包、分层架构和平台通用性方面设计

* 组件化结构

采用全组件化结构设计，每个组件都被独立地实现，并通过标准接口联系在一起。每个功能组件在功能上独立，同时可根据用户需求灵活配置、组合，实现平滑升级扩容，功能实体可使业务和开发人员根据具体使用要求增加或减少系统应用模块。

* 标准化接口

采用标准统一的接口设计，所有功能实体间的数据交换以及对其他模块的数据引用都通过标准接口完成，使多个组件对接时在开放性、稳定性、扩展性与集成性上有着很好的适配空间。

* 开放功能包

系统处理组件化结构设计与标准化接口设计以支撑开放体系结构外，为了方便用户个性应用的开发，还封装系统及其组件所需的二次开发应用工具包，使其他技术团队对平台进行二次开发时能够更好地复用。

* 分层架构设计

采用横向分层和纵向分割架构设计。将层与层之间相互分离，每层的应用和服务，采用独立的模块开发和部署，模块间交互标准化，新增功能模块分解到各层，以插件形式加入原系统，既不影响整体架构，也不影响本层功能提供，具备高模块化设计，保证了系统功能的可扩展性。纵向分割是将业务和可复用服务分离出来，通过分布式服务框架调用。新增产品可以通过调用可复用的服务实现自身的业务逻辑。

* 平台通用性

系统采用的软件开发技术可运行于通用的主流硬件平台上，不依赖于特定的、专用的硬件设备或者系统软件。系统配置（硬件系统、操作系统、数据库系统）的升级一般情况下，不会引起系统的修改和再次开发。

#### 管理开放性

管理开放性是指软件系统在业务流程、规则、审核和组织架构方面采用结构化配置和管理模式，根据系统管理流程，开放软件系统管理功能的可配置性，从而灵活快捷的支持管理流程开放性能力，适应不同的管理模式和流程。

### 扩展性

为了满足本项目的长远目标，适应业务和技术不断发展得需要，系统软件架构采用模块化体系结构，新功能、新业务的增加能够在不影响系统运行的情况下实现，使架构在一定范围内支持业务的快速变化，并能够适应新技术的应用。

随着用户和数据量增加，系统可能遇到资源瓶颈，如数据库出现容量和内存使用居高，应用服务器出现并发请求排队等。此时，可以通过横向扩展或物理分服机制提升系统的负荷能力。

* 横向扩展：通过集群的负载均衡机制，通过增加集群节点来传播负载和增加可靠性。作为系统扩展的主要方式，这将在绝大多数情况下满足系统的扩容需求；
* 物理分服：当用户和数据量持续增加到横向扩展无法提升的情况时，可以考虑通过物理分服的方式扩展系统。在系统总入口处通过用户路由到相应的服务群。

### 安全性

在设备选型和系统设计过程中，从安全架构、安全策略及安全机制等多方面充分地考虑安全问题。

系统及运行环境采取全面的安全保护措施，具有防病毒感染、防黑客攻击措施，同时在防雷击、过载、断电和人为破坏方面进行加强，具有高度的安全性和保密性。对接入系统的设备和用户，进行严格的接入认证，以保证接入的安全性。同时系统支持对关键设备、关键数据、关键程序模块采取备份、冗余措施，有较强的容错和系统恢复能力，确保系统长期正常运行。

### 可靠性

系统采用故障检查、告警和处理机制具有强的容错能力和错误恢复的能力，保证数据不因意外情况丢失或损坏，保证系统安全稳定可靠运行。

* 当操作错误时，能迅速有效地实现回滚。对系统硬件或网络故障，数据库系统也应能提供一定的保护措施，保证系统安全稳定可靠运行；
* 系统采用灵活的服务部署方案实现负载均衡，防止“瓶颈”产生；
* 系统具备多种冗余、备份和集群处理的机制和功能，关键部件、数据库具备冗余备份和负载分担机制，系统具备冗余配置，保证系统无单一故障点，且应易于扩容和维护；
* 系统提供完善的日志记录能力，对系统关键数据的每一次增加、修改和删除都能记录相应的修改时间、操作人和修改前的数据记录；
* 系统具有监控功能，能监视系统各功能模块的运行情况，随时发现系统自身的问题；
* 本系统提供容错机制，避免单点故障的发生，如采用集群、负载均衡等技术。

## 性能设计规范

### 性能指标

性能指标要求主要包括以下几个方面：

* 数据采集性能要求(在网络环境具备情况下)

结构化数据采集每秒大于10000条；

非结构化数据采集每秒大于一个80M文件。

* 页面调用性能要求(在网络环境具备情况下)

网内访问静态页面响应时间小于2秒，动态页面响应时间小于3秒；

网外访问静态页面响应时间小于3秒，动态页面响应时间小于5秒。

* 接口性能要求

并发处理：智库平台即可并发处理能力需要具备不低于100的并发。

### 可靠性指标

可靠性指运维团队快速响应7\*24小时服务能力，包括系统可靠性、接口可靠性、数据可靠性。

#### 系统可靠性

1、系统的Web服务器出现故障应能及时告警，应具有完整的操作权限管理功能和完善的系统安全响应机制。

2、 Web服务器应采用双机或者集群方式对外提供服务，在某台服务器出现故障时能保证页面能被正常访问。

3、通过前置负载均衡器，以一定的算法将外部请求分发到WEB应用，从而提升WEB服务的整体处理能力。

4、Web服务器升级时应通过负载均衡器在前端逐个隔离待升级节点，在业务节点升级完成后再将其加入集群。这样就能保证整个升级过程较为平稳，不会造成大量用户在升级期间无法访问业务。

5、为了减少异常断电对系统造成过大影响，尽量将Web服务注册到操作系统的自启动脚本中，从而保证操作系统正常运行后能自动启动应用，以便系统能尽快向外提供服务。

6、Web应用端口隔离，当不同的Web应用需要部署在同一主机时，应该尽量使每个Web应用享有独立的Web容器，达到各Web应用的进程级隔离目的；从主机上隔离会达到更好的资源隔离效果。

#### 接口可靠性

1、接口出现故障应能及时告警，应具有完整的操作权限管理功能和完善的系统安全机制。接口失败时应建立重试机制，出现故障时需要及时地响应。

2、接口升级、割接等过程中，应保证有系统回退机制。

3、接口应防止消耗过多的系统资源而使系统崩溃。

#### 数据可靠性

1. 数据完整性：要求智库平台提供的数据符合实体完整性约束、参照完整性约束。

2、数据及时性：要求智库平台中提供的内外部数据应是最新状态的数据。

3、数据一致性：要求智库平台提供的数据对内对外应保持一致。。

4、 数据准确性：要求智库平台提供的数据是准确的。

# 接口标准

## 接口要求

* 信息通讯安全

保证接口的自身安全，通过接口实现技术上的安全控制，做到对安全事件的“可知、可控、可预测"，是实现系统安全的一个重要基础。可对其进行：安全评估、访问控制、防恶意代码、加密等方法确保接口通讯安全。

* 支持高开发

系统在设计时考虑到多线程的情况，可采用了分布式架构和负载均衡技术，能够支持大规模的并发访问。

* 可监控

具备完善的监控功能，可以实时监控系统运行状态、用户访问情况等，及时发现并解决问题，保证系统的稳定性和可靠性。

* 系统资源的动态扩展

保证在充分利用系统资源的前提下，实现系统平滑的移植和扩展，同时在系统并发增加时提供系统资源的动态扩展，以保证系统的稳定性。

* 异常处理机制

表单验证、唯一性检查、或其他可预期的错误。我们需要编写特定代码来捕获这类错误，并抛出一个包含提示信息的全局异常，捕获并返回给客户端。

* 业务扩展

系统在运行过程中，根据需要扩展业务功能例如，当用户需求发生变化时，可以增加新的业务功能以满足用户需求。这种方式可以提高系统的灵活性和可扩展性，但是需要考虑业务变化的频率和影响范围。。

## 技术方式

提供的数据接口技术主要包括以下类型:

* HTTP接口

通过HTTP协议传输的接口，可以传输文本表单数据，也可以传输json类型的对象数据或xm1类型的数据。

* SOAP

简单面向对象协议，基于HTTP，使用xml作为默认传输格式。

* Web Serveice

相比传统的HTTP接口只传输文本请求和文本响应，通过Web Service可以直接拿到远程的一个对象，并能够直接调用对该对象的属性和方法，比HTTP更高级。

* FTP

FTP是用于在网络上进行文件传输的一套标准协议，客户在和服务器建立连接前要经过一个“三次握手”的过程，保证客户与服务器之间的连接是可靠的，而且是面向连接，为数据传输提供可靠保证。

## 报文格式

数据接口的报文格式主要分为报文内容、异常代码标准、报文格式说明规范组成。

* 接口内容主要分为XML和JSON两种报文格式。
* XML可扩展标记语言，标准通用标记语言的子集，是一种用于标记电子文件使其具有结构性的标记语言。
* JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。大数据基础平台对外开放接口采用REST风格，使用HTTP+JSON报文方式承载交易信息，支持GET、POST两种请求方式。
* 接口异常代码标准

即调用数据服务接口异常场景下返回的错误码，返回异常码的同时也会返回中文详细描述，以增强异常码的可读性。异常码通常表示某种异常场景，具有唯一指向性。

* 报文格式说明规范
* 简介
* 接口描述
* 请求参数、响应
* 应用示例
* 异常码说明

# 代码开发标准

## 管理规范

### 项目立项

部门经理指定开发经理跟主要开发人员，初步构建开发团队。

### 项目技术调研

开发经理和核心开发负责，根据项目实际情况进行技术框架选型、关键技术难点调研，并输出技术调研和可行性分析报告。

### 产品原型和设计评审

在原型评审前一天把完整的原型设计文档，功能清单及其他相关文档发送开发经理、核心开发及部门经理。在原型会议评审前，开发经理及核心开发同事必须阅读产品设计文档，从技术实现的角度评估原型和产品设计中的业务逻辑，并形成问题清单，开发经理负责检查，部门经理负责跟进。

开发经理及主要核心开发人员参与原型的会议评审，在会议期间开发人员和产品经理针对问题进行讨论确认，如果会议中不能达成一致意见的，必须在会议纪要中体现并后续跟进直到达成一致意见，如有必要的话，再次进行会议评审，所有的变更都需要更新到最新的原型中并跟进确认。

### 需求变更

需求变更须知会开发经理和部门经理， 每次变更时都需要确定技术方案，并评估对开发进度的影响， 并将影响及时反馈到项目经理或者产品经理以及部门经理。

### 计划管理

在原型评审完之后， 由开发经理负责制定开发计划， 部门经理审核。在开发计划制定的过程中需要与其它依赖部门沟通确认前置条件。开发计划制定完之后要知会项目团队成员和项目经理以及其它相关人员。

如果项目上采用迭代式开发， 则开发计划采用双周作为一个周期， 每周五梳理和更新双周计划。开发计划的制定过程中，任务分解的粒度到人到天，同时列出依赖条件和可能存在的风险。计划的变更要及时知会到项目经理或者产品经理以及部门经理或者其它项目团队成员。

### 任务和计划执行跟进

开发经理组织每日晨会，总结昨天工作完成情况以及遇到问题和今天预期的工作计划。

开发经理负责处理遇到的问题， 如果有什么问题及时反馈到部门经理和产品经理。

开发经理将项目的周报以及本组内成员的周报，内容包括本周工作完成情况、风险把控以及下周计划，及时反馈到项目经理，各部门经理，产品经理手中。

开发经理参加项目每周跟踪会并及时反映情况，内容包括本周工作完成情况、风险把控、下周计划等。

### 系统设计

包括技术选型和框架搭建、数据库设计和接口设计。

项目中必须复用已有成果的部分，不允许重写，但可以在原来组件的基础上进行优化和功能扩展。

项目中能封装成公用组件的部分尽量封装成共用组件，积累技术成果。

开发经理负责技术选型和框架搭建、数据库设计和接口设计，需要进行评审， 部门经理进行审核。

框架需要使用开发经理选择的统一框架结构， 如果需要使用新的框架必须进行技术调研和评估，部门经理跟进和审核。

数据库设计和接口设计必须遵守规范，必须由开发经理或者高级开发（除开发经理外，其他人员不可建库建表，如有需要，跟开发经理反馈）承担，需要进行评审，开发经理把关。

### 代码编写

严格遵循编码规范要求。

开发经理或组长负责代码功能检查并负责本组内开发人员的代码质量。

核心业务逻辑（算法）必须进行内部评审讨论和代码评审。

新员工的代码必须进行代码评审。

### 系统测试和Bug修复

提测前必须内测，开发经理负责，内测结果反馈部门经理。提测时形成提测说明文档，并邮件发送项目经理，产品经理，质量经理，项目总监。提测时数据库变更必须提供变更脚本。

S1、S2级别Bug优先修复，修复完的Bug，需要自测，避免重新激活，且不能产生新的Bug，在修改Bug的过程中，如针对Bug描述不理解的情况，跟相关测试人员，产品经理进行沟通确认。根据实际情况，S1/S2级别Bug，可以指定修改期限。

每一次提测冒烟测试必须通过。

每一次提测后测试小结（质量发送的）如有必要，问题特别多，而且比较复杂，开发经理可组织内部针对性讨论，并根据计划安排修改（部门经理根据严重情况参加）（如果跟数据相关，主动跟进，同时反馈部门经理和项目经理或产品经理）

开发经理在每日晨会时，跟进Bug修改情况。

在Bug的修复过程中， 开发经理或者组长需要承担本组内或者所负责模块的Bug修复方案、修复质量。针对系统性能优化等严重问题（比如数据加载慢），开发经理组织专题讨论会，进行修改（如有必要，部门经理参与并制定优化方案）。

### 系统运维

系统上线后出现的问题，开发经理或组长需要负责安排相关人员进行修改维护。

### 项目资源管理

系统上线后，与项目经理确认后续的运维工作内容， 并评估工作量以及据此确认预留资源， 其它资源释放出项目。部门内的核心技术人员需要在多个项目中复用承担架构设计和技术难题的解决。

## 开发环境

统一规定公司的所有软件项目的开发环境，使公司的所有软件项目都在统一的开发环境下进行开发，以便软件项目的移植，同时也避免了因为环境不一致而导致的软件项目不能共享和开发不能协作等问题。

### 操作系统环境

建议开发操作系统为Windows 10 或 Linux环境。

### 开发工具

开发工具的指定，是从开发工具的版权、开发工具的功能以及软件的可扩展性等多方面进行考虑的，集成开发环境、工具包与应用服务器分别指定如下：

1、集成开发环境：对于JAVA开发，推荐使用IntelliJ IDEA，IntelliJ IDEA 是一款功能强大的集成开发环境，提供了丰富的功能和强大的调试能力，其对于大型JAVA项目具有有效的支持，可以提高开发效率和代码质量。对于C#开发，推荐使用Visual Studio 2022，Visual Studio 是微软推出的一体化开发环境，其提供了强大的编辑器、调试器、代码分析工具和丰富的扩展支持，适合于开发大型C#应用和跨平台开发。

2、开发工具包：对于JAVA开发，推荐使用JDK 17， JDK 17 是当前的LTS版本，提供了稳定性和长期支持。对于C#开发，推荐使用.NET 6，.NET 6是微软最新主要版本，提供了更快的性能、新的语言特性和改进工具集。

3、应用服务器：对于JAVA开发，推荐使用Apache Tomcat，Tomcat 提供了简单易用的配置和管理，性能可靠，支持Servlet 5.0和JSP 3.0规范。对于C#开发，推荐使用Microsoft Internet Information Services（IIS），其完全集成在Windows操作系统中，提供了出色的性能、扩展性和安全性。

具体的版本选择由技术经理或组长选择并准备，以压缩包或仓库链接形式发送至开发人员使用。

### 版本/代码管理工具

推荐使用Git作为版本/代码管理工具，Git 是分布式版本控制系统，广泛用于各类开发项目。使用Git可以轻松管理代码版本，进行分支和合并操作。

### 代码管理服务

推荐使用Azure Repos作为代码管理服务平台，Azure Repos 是 Azure DevOps 提供的版本控制服务，支持 Git 和 Team Foundation Version Control (TFVC)。它提供了无限的私有 Git 存储库，并集成了 Azure DevOps 平台的其他功能。

## 开发框架

本章主要阐述项目目录结构的规定以及各部分的命名规范。关于框架组件选用将在后续章节规定。

### 整体结构规范

项目应当采用统一的结构，以便系统的项目维护和开发团队之间进行共享资源，对于项目整体目录规范结构描述如下：

1、JAVA项目目录规范：

1. /ProjectName
2. ├── .gitignore
3. ├── README.md
4. ├── docs/                    # 项目文档目录
5. ├── src/                     # 源代码目录
6. │   ├── main/
7. │   │   ├── java/            # Java代码目录
8. │   │   │   └── com/
9. │   │   │       └── example/
10. │   │   │           └── project/
11. │   │   │               ├── controllers/
12. │   │   │               ├── services/
13. │   │   │               ├── models/
14. │   │   │               └── utils/
15. │   │   ├── resources/      # 资源文件目录
16. │   │   │   ├── application.properties
17. │   │   │   └── log4j2.xml
18. │   ├── test/
19. │   │   ├── java/            # 测试代码目录
20. │   │   └── resources/       # 测试资源目录
21. ├── lib/                     # 第三方库和依赖目录
22. ├── build/                   # 构建脚本和CI/CD配置
23. └── .github/                 # GitHub相关配置和文档

2、C#项目目录规范：

1. /ProjectName
2. ├── .gitignore
3. ├── README.md
4. ├── docs/                      # 项目文档目录
5. ├── src/                       # 源代码目录
6. │   ├── ProjectName/           # 主项目目录
7. │   │   ├── Controllers/       # 控制器
8. │   │   ├── Services/          # 服务
9. │   │   ├── Models/            # 数据模型
10. │   │   ├── Repositories/      # 数据仓库
11. │   │   ├── ViewModels/        # 视图模型
12. │   │   ├── Utils/             # 工具类
13. │   │   ├── Properties/        # 项目属性
14. │   │   ├── appsettings.json   # 应用程序设置
15. │   │   ├── Program.cs         # 程序入口
16. │   │   └── Startup.cs         # 启动配置
17. │   ├── ProjectName.Tests/     # 测试项目目录
18. │   │   ├── UnitTests/         # 单元测试
19. │   │   ├── IntegrationTests/  # 集成测试
20. │   │   └── TestUtilities/     # 测试工具
21. ├── lib/                       # 第三方库和依赖目录
22. ├── build/                     # 构建脚本和CI/CD配置
23. └── .github/                   # GitHub相关配置和文档

### 开发框架使用

项目开始开发（编码）时，项目组首先从代码服务器上获取最新版本的系统框架，系统框架包括系统整体框架、相关组件包、框架界面、登录界面等。

### 代码结构规范

1、JAVA代码结构规范：

基于Java平台的项目，所有包名均以 com.<业务模块> 开头。命名规则是：com.业务模块.模块名称，如用户登录模块名称为：com.userconter.login。

2、C# 代码结构规范：

基于C# 平台的项目，所有命名空间均以 ProjectName 开头。命名规则是：ProjectName.ModuleName，如用户登录模块命名为 ProjectName.Login。

### 模块命名规范

模块是指系统中的各个业务功能模块，各个模块在组织上和功能上相对独立。模块的命名要有含义，对于Java下的各个模块的名称统一采用小写英文字母命名，而C#下的各个模块的名称则通常以大写字母开始。

### 配置文件规范

JAVA配置文件规范：

Java项目的配置文件主要涉及SSM框架配置、日志组件配置文件及其他配置文件。配置文件位置及命名规范如下：

1、Spring配置文件位置及命名规范：Spring配置文件统一存储在 src/main/resources 目录位置，命名为 applicationContext.xml，spring-servlet.xml，系统默认公共的配置文件为 applicationContext-common.xml。

2、日志组件配置文件：日志组件配置文件统一命名为 log4j.properties，存放在 src/main/resources 目录位置。

3、数据库配置文件：数据库组件配置文件统一命名为 jdbc.properties，存放在 src/main/resources 目录位置。

C# 配置文件规范：

C# 项目的配置文件主要涉及应用程序配置、日志组件配置文件以及数据库配置文件。配置文件位置及命名规范如下：

1、应用程序配置文件：应用程序配置文件统一存储在 src/ProjectName/ 目录位置。常见文件包括 appsettings.json 和 appsettings.<Environment>.json（如 appsettings.Development.json、appsettings.Production.json）。

2、日志组件配置文件：日志组件配置文件存放在 src/ProjectName/ 目录位置，命名为 log4net.config 或 nlog.config，根据使用的日志库而定。

3、数据库配置文件：数据库配置文件通常包含在 appsettings.json 中的 ConnectionStrings 部分。如果使用单独的数据库配置文件，则存放在 src/ProjectName/ 目录位置，命名为 dbsettings.json。

## 代码规范

JAVA代码规范请参阅：[https://google.github.io/styleguide/javaguide.html#s4.8.8-numeric-literals](https://google.github.io/styleguide/javaguide.html" \l "s4.8.8-numeric-literals)

C#代码规范请参阅：<https://learn.microsoft.com/zh-cn/dotnet/csharp/fundamentals/coding-style/coding-conventions>

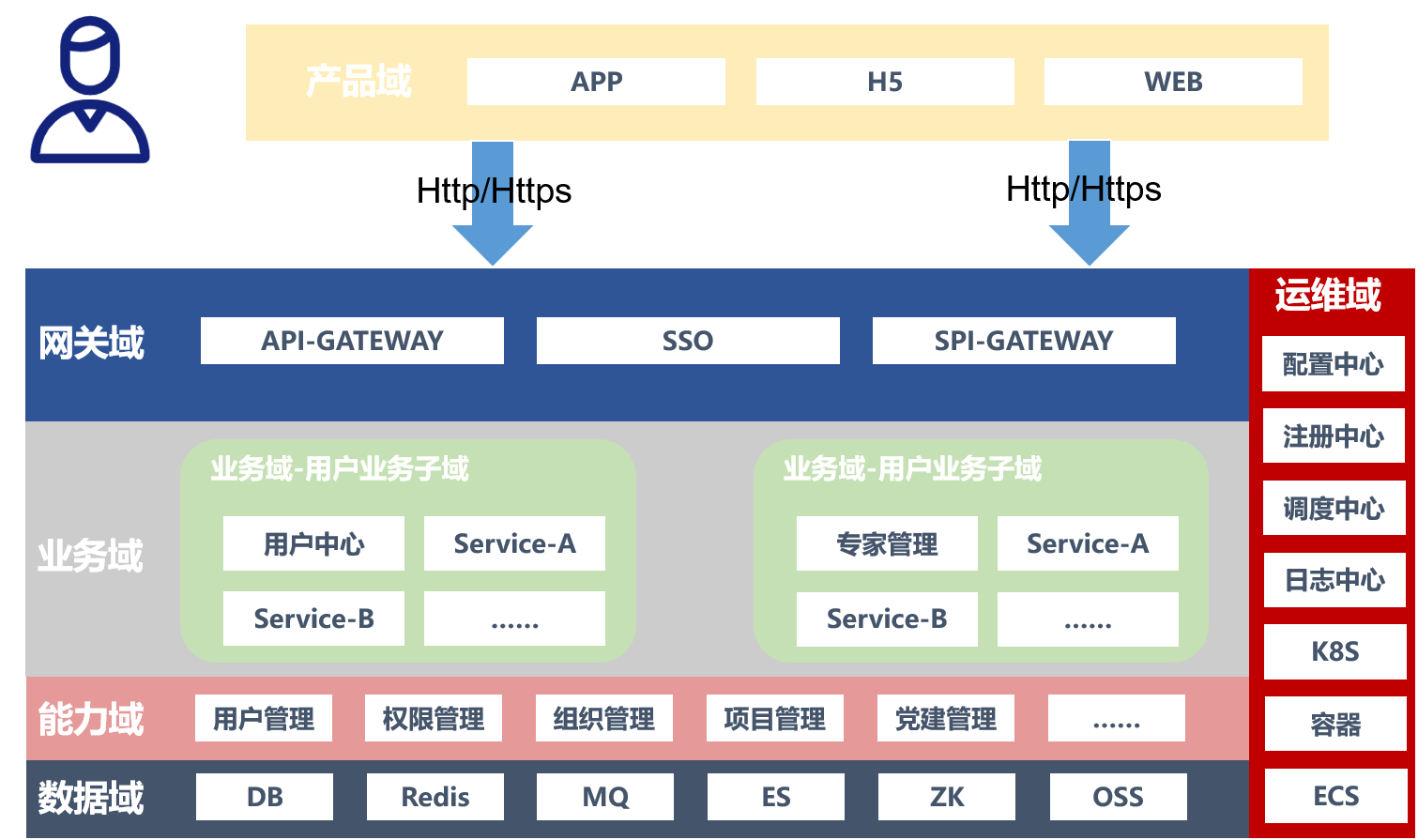
## 数据库规范

数据库规范请参阅：

<https://www.sqlstyle.guide/>

# 应用框架规范

应用系统建设模式基于插件化和服务化，采用主流的轻量架构，可对服务和资源的整体管理。平台在PaaS层采用更加轻量的Docker容器，组件化架构采用微服务架构。



1、**⽹关层：**为业务平台提供内外网隔离、统⼀鉴权、限流、降级等，提高系统整体安全性、稳定性。

* 网关域仅负责通用平台的进出口请求业务分发、协议转换、权限控制等通用功能，不提供任何实际业务代码；
* 平台各个业务系统均不提供外网权限，外部系统主动请求平台通过API网关转发；
* SSO服务仅提供多渠道登录鉴权、分布式session、AccessToken验证功能，不提供其他业务逻辑。

2、**业务层：**基于能力层功能，完成平台各个业务流程组织，根据业务范围做业务编排。

* 业务域仅负责不同业务流程的组织，而不负责核⼼领域数据的持久化及状态维护（例如业务域负责微信授权注册、网站⼿机注册2种不同流程主旨，但是实际的⽤户创建和持久化通过调用能力层完成）；
* 业务域仅负责不同业务流程的组织，而不负责访问权限、登录会话等信息维护。（权限和会话统⼀由网关层统⼀维护）。

3、**能力层：**核⼼业务领域模型、标准作业流程维护，对外输出通⽤能⼒，⼀边个性化业务流程组织。

* 能力域各业务系统仅负责核⼼领域模型等持久化、状态流转（例如：⽤户服务）；
* 业务域各个系统通过RPC接⼝调⽤能⼒域的通⽤能⼒。（例如注册）。

4、**数据域层：**各个业务板块涉及的数据保存、缓存、等功能架构。

* 数据域统⼀由运维、数仓维护，各业务系统不再搭建数据存储服务；
* 各个业务系统根据业务需求申请特定规模的数据存储配额和账号；
* 数据域暂时以私有云服务为主，以提升系统底层数据稳定性。

5、**运维域：**平台运维管控相关业务架构，包括系统监控、运维保障等。

* 需提供运维相关⼯具类服务，不实际涉及业务逻辑。

# UI设计风格规范

详见《UI设计风格规范及原则分册》