智医办技规 2025-001

文档版本：[V1.0]

日期：[2025.07]

天津市第一中心医院

基于云原生结构、微服务的软件开发要求书

1. 引言

目的：本文档旨在定义基于云原生架构和微服务架构的软件开发要求，以确保开发团队能够构建出可扩展、高可用、易于维护的软件系统。

范围：本文档涵盖了从架构设计、技术选型、开发规范到部署运维的各个方面，适用于参与项目开发的所有团队成员。

2. 架构要求

云原生架构原则：

- 容器化：通过容器化技术打包应用程序及其依赖项，支持灰度发布机制以实现服务的平滑升级。基于实时负载监控数据，可动态调整吞吐量，实现资源弹性伸缩。
- 编排管理：采用 Kubernetes 等容器编排工具进行自动化部署和资源管理。
- 微服务架构：采用中台化设计，以微服务为基础衍生前台应用，其中中台微服务包含网关等核心组件，为客户端前台应用提供支撑。
- 弹性设计：支持自动伸缩和故障恢复机制。

- 云平台适配：具备全面的云平台适配能力，支持与三大运营商（联通云 CSK、天翼云 ECK、移动云 KCS）进行技术适配优化。同时支持阿里云 ACK、腾讯云 TKE、华为云 CCE 等主流云平台，实现业务数据跨境、异地双活及智慧医院建设。
- 持续交付：通过 CI/CD 流水线实现自动化测试和发布，保障迭代效率与稳定性。
- 信创适配：支持对主流信创基础配套软硬件进行适配。
- 可观测性：系统关键检测指标要求全链路记录，包括从客户端、应用服务器、数据库、硬件环境的全链路分析，提高异常跟踪分析能力。

3. 技术选型

前端开发技术：javascript、Vue、React、python

后端开发技术：Spring Boot、Spring　MVC、Spring SECURITY、Spring Data、Spring Cloud、Java

数据库：选择结构化数据库并支持信创，需满足以下核心标准：数据库产品必须纳入国家信创目录，确保核心技术自主可控；具备等保 2.0 三级及以上安全能力，支持数据加密、细粒度权限管控等安全特性；提供分布式高可用架构，保障业务连续性，性能指标需匹配医疗等高并发场景；兼容主流数据库语法并配套迁移工具，降低系统改造难度；例如达梦数据库、海量数据、人大金仓、腾讯 TDSQL 等。

消息队列：

RabbitMQ 是一款基于 AMQP 协议的开源消息中间件，采用 Erlang 语言开发，具有高可靠、高并发的特性。它支持多协议（MQTT/STOMP 等），提供消息持久化、ACK 确认、集群高可用等机制确保消息可靠传输，并通过灵活的 Exchange 路由机制（Direct/Topic/Fanout 等）实现精准消息分发。具备 Web 管理界面和丰富插件，支持 Java/Python/Go 等多语言 SDK。

RocketMQ 是一款高性能、高可靠的分布式消息中间件，支持高吞吐、低延迟、严格消息顺序和事务消息，适用于大规模分布式系统。其主从架构和 Dledger 协议保障高可用，消息回溯和堆积能力满足大数据场景需求，结合完善的监控工具，是构建异步解耦、流量削峰等场景的理想选择。

中间件：

Elasticsearch (ES)：分布式搜索分析引擎，基于 Lucene 构建，提供近实时检索能力，支持水平扩展和 RESTful API，典型应用于日志分析(ELK)、商品搜索等场景。

Redis：高性能内存数据库，支持多种数据结构，具备微秒级读写能力，常用作缓存、会话存储和实时排行榜，支持持久化和集群部署。

Nginx：高性能 Web 服务器/反向代理，C 语言编写支持高并发，提供负载均衡、静态资源托管和 API 网关功能，支持 Lua 扩展。

XXL-job：分布式任务调度平台，提供可视化任务管理、分片执行和故障转移，适用于定时任务和批量数据处理场景。

Consul/Nacos：服务网格解决方案，提供服务发现、健康检查和多数据中心支持，基于 Raft 协议，与云原生技术深度集成。

5. 开发规范

代码命名规范：

代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式也要避免采用。

类名使用 UpperCamelCase 风格，但以下情形例外：DO / BO / DTO / VO / AO / PO / UID 等。

方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从驼峰形式。

常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。

抽象类命名使用 Abstract 或 Base 开头；异常类命名使用 Exception 结尾；测试类命名以它要测试的类的名称开始，以 Test 结尾。

POJO 类中布尔类型的变量，都不要加 is 前缀。

杜绝完全不规范的缩写，避免望文不知义。

如果模块、接口、类、方法使用了设计模式，在命名时需体现出具体模式。

说明：将设计模式体现在名字中，有利于阅读者快速理解架构设计理念。

接口类中的方法和属性不要加任何修饰符号（public 也不要加），保持代码的简洁性，并加上有效的 Javadoc 注释。尽量不要在接口里定义变量，如果一定要定义变量，肯定是与接口方法相关，并且是整个应用的基础常量。

对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部的实现类用 Impl 的后缀与接口区别。

如果是形容能力的接口名称，取对应的形容词为接口名（通常是–able 的形式）。

枚举类名建议带上 Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。说明：枚举其实就是特殊的类，域成员均为常量，且构造方法被默认强制是私有。

代码格式规范：

大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行；如果是非空代码块则： 左大括号前不换行，左大括号后换行，右大括号前换行，右大括号后还有 else 等代码则不换行；表示终止的右大括号后必须换行。

左小括号和字符之间不出现空格；同样，右小括号和字符之间也不出现空格；而左大括号前需要空格。

if/for/while/switch/do 等保留字与括号之间都必须加空格。

任何二目、三目运算符的左右两边都需要加一个空格。说明：运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号等。

注释的双斜线与内容之间有且仅一个空格。

方法参数在定义和传入时，多个参数逗号后边必须加空格。

IDE 的 text file encoding 设置为 UTF-8; IDE 中文件的换行符使用 Unix 格式，不要使用 Windows 格式。

单个方法的总行数不超过 80 行。

不同逻辑、不同语义、不同业务的代码之间插入一个空行分隔开来以提升可读性。

代码注释规范：

类、类属性、类方法的注释必须使用 Javadoc 规范，使用/*内容/格式，不得使用// xxx 方式。

所有的抽象方法（包括接口中的方法）必须要用 Javadoc 注释、除了返回值、参数、异常说明外，还必须指出该方法实现的功能及流程。

所有的类都必须添加创建者和创建日期。

方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释使用/* */注释，注意与代码对齐。

所有的枚举类型字段必须要有注释，说明每个数据项的用途。

代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑等的修改。

代码处理规范：

在 subList 场景中，高度注意对原集合元素的增加或删除，均会导致子列表的遍历、增加、删除产生 ConcurrentModificationException 异常。

使用集合转数组的方法，必须使用集合的 toArray(T[] array)，传入的是类型完全一样的数组，大小就是 list.size()。

使用工具类 Arrays.asList()把数组转换成集合时，不能使用其修改集合相关的方法，它的 add/remove/clear 方法会抛出 UnsupportedOperationException 异常。

禁止在 foreach 循环里进行元素的 remove/add 操作。remove 元素请使用 Iterator 方式，如果并发操作，需要对 Iterator 对象加锁。

创建线程或线程池时请指定有意义的线程名称，方便出错时回溯。

线程资源必须通过线程池提供，不允许在应用中自行显式创建线程。

线程池不允许使用 Executors 去创建,而是通过 ThreadPoolExecutor 的方式，这样的处理方式让写同学更加明确线程池运行规则，避资源耗尽风险。

SimpleDateFormat 是线程不安全的类，一般不要定义为 static 变量，如果定义为 static，必须加锁，或者使用 DateUtils 工具类。

多线程并行处理定时任务时，Timer 运行多个 TimeTask 时，只要其中之一没有捕获抛出的异常，其它任务便会自动终止运行，使用 ScheduledExecutorService 则没有这个问题。

在一个 switch 块内，每个 case 要么通过 break/return 等来终止，要么注释说明程序将继续执行到哪一个 case 为止；在一个 switch 块内，都必须包含一个 default 语句并且放在最后，即使空代码。

在 if/else/for/while/do 语句中必须使用大括号。即使只有一行代码，避免采用单行的编码方式：if (condition) statements;

OOP 规约：

避免通过一个类的对象引用访问此类的静态变量或静态方法，无谓增加编译器解析成本，直接用类名来访问即可。

所有的覆写方法，必须加@Override 注解。

相同参数类型，相同业务含义，才可以使用 Java 的可变参数，避免使用 Object。

外部正在调用或者二方库依赖的接口，不允许修改方法签名，避免对接口调用方产生影响。接口过时必须加@Deprecated 注解，并清晰地说明采用的新接口或者新服务是什么。

不能使用过时的类或方法。

Object 的 equals 方法容易抛空指针异常，应使用常量或确定有值的对象来调用 equals。

所有的相同类型的包装类对象之间值的比较，全部使用 equals 方法比较。

所有的 POJO 类属性必须使用包装数据类型。

定义 DO/DTO/VO 等 POJO 类时，不要设定任何属性默认值。

POJO 类必须写 toString 方法。使用 IDE 中的工具：source> generate toString 时，如果继承了另一个 POJO 类，注意在前面加一下 super.toString。

RPC 方法的返回值和参数必须使用包装数据类型。

所有的局部变量使用基本数据类型。

API 设计规范：

统一使用 HTTP POST 方法进行所有请求。

认证、压缩、MessageId 等自定义扩展等信息包含在 HTTP Header 中。

参数放在 HTTP Body 中。

MIME 类型：Content-Type: application/json; charset=utf-8。

每个 api 服务接口都有对应的编码，例如 /api/v1/base/employee/get 对应编号 1100001.1。

URL 命名方式： `/api/{版本号}/{模块}/{资源}/{动词}/{限定或扩展}`。

对接收的输入参数需要进行类型校验。

新增参数时必须考虑兼容性。

已发布的接口不允许修改参数类型。

已发布的接口不允许删除

异常记录规范：

禁止直接返回错误码，但必须使用包含错误码的基于 RuntimeException 的异常。

尽量捕捉预期的错误。

根据错误类型使用不同的自定义异常类。

参数校验应当发布层进行校验。

Java 类库中定义的可以通过预检查方式规避的 RuntimeException 异常不应该通过 catch 的方式来处理。

异常不要用来做流程控制，条件控制。

不会抛异常的代码，不要在 try-catch 块里。

有 try 块放到了事务代码中，catch 异常后，如果需要回滚事务，一定要注意手动回滚事务。

finally 块必须对资源对象、流对象进行关闭，有异常也要做 try-catch。

不要在 finally 块中使用 return。

public 方法最多只应该抛出一个 checked 异常。

方法块内禁止直接抛出 new CheckedException。

日志记录规范：

日志文件至少保存 15 天，因为有些异常具备以“周”为频次发生的特点。

日志进行分类，如将错误日志和业务日志分开存放，便于开发人员查看，也便于通过日志对系统进行及时监控。

对 debug/info 级别的日志输出，必须使用条件输出形式或者使用占位符的方式。

在高性能要求场景，对肯定会打印的 log，还是用 StringBuilder 进行 String 的组合，效率比占位符还高。

异常信息应该包括两类信息：案发现场信息和异常堆栈信息。如果不处理，那么通过关键字 throws 往上抛出。

生产环境禁止输出 debug 日志；有选择地输出 info 日志；如果使用 warn 来记录刚上线时的业务行为信息，一定要注意日志输出量的问题，避免把服务器磁盘撑爆，并记得及时删除这些观察日志。

6. 部署与运维

持续集成/持续部署（CI/CD）：

通过 CI/CD 流程，能够在代码提交后迅速触发自动化构建和测试，及时发现并修复问题，避免后期集成时出现大量冲突和错误。使用 Jenkins 作为 CI/CD 工具，灵活配置流水线，从代码编译、单元测试到自动化部署。

环境配置：

开发环境需要支持快速迭代和灵活配置。测试环境要求尽可能与生产环境保持一致，以便准确发现潜在问题。预生产环境用于进行最终的验证和性能测试。生产环境则要求高可用性、安全性和稳定性，通常会采用负载均衡、冗余设计等措施。不同环境的配置管理需要标准化和自动化，避免因环境差异而导致问题，同时也要确保环境之间的隔离，防止数据泄露和意外影响。

服务治理：

服务发现机制支持服务实例动态注册和发现，使得系统能够灵活应对服务的扩缩容和故障切换。

负载均衡通过合理分配请求到不同的服务实例，优化资源利用率并提高系统的吞吐量。

熔断器模式则是在服务调用链中，当某个服务出现故障时，快速切断调用链，防止故障蔓延，同时为故障恢复提供缓冲时间。

限流措施可以限制请求的频率和数量，避免系统因流量过高而崩溃。

监控与日志：

使用 Prometheus 作为监控系统，通过拉取或推送的方式收集指标数据，支持灵活的查询语言和丰富的可视化功能，能够实时监控系统性能、资源使用情况等关键指标。通过直观的仪表盘展示监控数据，帮助运维人员快速定位问题。通过监控与日志的结合，运维团队可以全面掌握系统的运行状态，实现主动运维和故障快速恢复。

性能优化：

使用 JMeter 性能测试工具模拟真实场景下的负载，发现系统的瓶颈和潜在问题，支持生成详细的性能报告，帮助开发人员定位问题。调优策略涉及多个层面，包括代码优化、数据库优化、缓存策略优化等。常见的性能指标包括响应时间、吞吐量、资源利用率等,通过持续的性能测试、调优和监控，系统可以不断提升性能，满足用户的需求。

7. 质量保证

代码质量：

使用静态代码分析工具检查代码质量。

使用工具：ArchUnit（Java）进行架构合规性检查。

文档编写：

要求每个服务提供 API 文档、设计文档和使用指南。

建立全生命周期的文档管理体系，包括：需求规格说明书、详细设计说明书、测试用例、部署配置手册等。

版本控制：使用 Git 等版本控制工具进行代码管理。

8. 项目管理

团队协作：建立有效的沟通机制，如每日站会、代码评审、问题跟踪等。

进度跟踪：使用敏捷方法或看板进行项目管理和进度跟踪。

风险管理：识别、评估和缓解项目风险的计划。

9. 附录

前端开发工具：Visual Studio Code

后端开发工具：Intelli IDEA

前端界面设计规范：

布局通过使用统一的元素和间距来规范不同环境及屏幕大小下输出的一致性，确保沟通高效、顺畅。

网页布局设计应依照扁平化原则进行，避免阴影、透视、纹理等复杂装饰设计，也可独立提供内容简约的无障碍大版块网页样式。

在不依赖操作系统和浏览器的前提下，计算机无障碍网页应提供网页的放大设置与大字屏幕服务，放大程度可达 200%。

段落内文字的行距至少为 1.3 倍，且段落间距至少比行距大 1.3 倍，同时兼顾移动应用适用场景和显示效果。

鼠标，或指点，或键盘操作，或以其他方式聚焦到页面各组件时，该组件应有明显的状态提示。

网页应用中，不应包含任何闪光超过 3 次每秒的内容，或闪光低于一般闪光和红色闪光阈值。

以非文本形式的链接，应提供语音阅读其链接的目的或链接用途的无障碍服务。

推荐使用思源黑体 CN、阿里普惠体等，可免费商用字体

接口设计规范：

编写符合业务使用的接口，只返回需要的字段和数据。接口设计应该尽量体现出差异，即根据不同的业务需求设计不同的接口，而不是设计一个大而全的接口。此外，接口命名应符合业务的语义，方便理解和使用，并有助于控制接口的容量。

在新增接口时，要避免类似"allInOne"的设计，而是根据具体需求设计精细化的接口。

在调用下游接口时，设置合理的超时时间，以确保在一定时间内获取响应结果。这可以通过配置框架或手动设置连接和读取超时时间来实现。避免长时间等待和资源浪费。

用回调式设计可以避免我们的系统被第三方系统的阻塞行为所拖慢，并提高系统的并发性能和响应能力。

日志设计规范：

限制日志大小为 512KB 以内，避免日志文件过大对磁盘和性能造成负担。

按需打印接口的入参日志，以方便排障。在关键的接口入口和下游调用的位置，添加代码来记录相关入参信息。

在异常日志中记录业务关键信息，例如请求的关键字段、业务流程等，方便问题定位和分析。

数据模型设计规范：

表命名使用名词或动名词结尾的对象类术语，用于表达对象类的概念

表名长度应控制在 1 - 60 个字符之间

表必须包含主键

多个单词组成的表名用下划线分隔

字段名长度应控制在 1 - 30 个字符之间

索引名长度应控制在 1 - 60 个字符之间

外键索引必须且只能存在一条索引列，且必须是外键

约束名长度应控制在 1 - 60 个字符之间

主键约束名通常以表名_PK 结束

外键约束名通常以 FK_开头，后跟表名和被引用表的表名（或表名的缩写）

唯一约束名通常以 UQ_开头，后跟表名和字段名（或字段名的缩写）

检查约束名通常以 CK_开头，后跟表名和字段名（或字段名的缩写）

默认约束名以 DF_开头，后跟表名和字段名（或字段名的缩写）

实体索引建议不超过 8 个

实体复合索引建议不超过 4 个字段

实体的总字段数量建议不超过 100 个字段

包含 NULL 值的列、TEXT/BLOB 等大字段类型或频繁修改的列不禁止创建索引

10. 评审与更新

评审流程：定义要求的评审和批准流程。

更新周期：规定文档的定期更新和版本控制。

请注意：本文档仅为软件开发要求的框架，具体内容需要根据项目实际情况进行调整。

注：本规范解释权归天津市第一中心医院智慧医院建设管理办公室

天津市第一中心医院智慧医院建设管理办公室

2025 年 7 月 25 日