**C#技术管理规范**

分级标准：强制（a）、推荐（b）、允许（c）

## **命名规范**

1. a. 所有类名、接口名必须使用 PascalCase（如：CustomerService）。
2. a. 局部变量、方法参数必须使用 camelCase。
3. a. 常量使用全大写加下划线（如：MAX\_LENGTH）。
4. a. 接口命名应以 I 开头（如：ILogger）。
5. b. 异步方法应以 Async 结尾（如：LoadDataAsync）。
6. b. 枚举成员使用 PascalCase，枚举类型可加 Enum 后缀。
7. c. 私有字段可使用下划线前缀（如：\_userRepository）。
8. b. 避免使用模糊或无意义命名（如：data1, temp 等）。

## **格式与结构**

1. a. 使用 4 个空格缩进，禁止使用 Tab。
2. a. 每个类一个文件，类名与文件名保持一致。
3. a. 方法之间必须空一行。
4. b. 花括号 {} 必须独占一行（Allman 风格）。
5. b. 成员顺序：字段 → 属性 → 构造函数 → 公共方法 → 私有方法。
6. b. 所有成员必须显式指定访问修饰符（public / private / internal）。
7. c. 对于大型类，可使用 #region 折叠逻辑分组。
8. c. 可用空行提升可读性，但不能连续两行空行。
9. b. 属性访问器要换行书写（避免一行写完 get/set）。
10. a. 禁止使用过时（[Obsolete]）的 API，除非明确有替代方案。

## **注释规范**

1. a. 公共类、接口、方法必须使用 /// XML 注释。
2. a. 注释需准确、简洁，避免描述性废话（如：“加法方法返回两个数的和”）。
3. b. 所有 TODO 必须附责任人和日期（如：// TODO(john): 优化循环逻辑 2025-06）。
4. b. 复杂算法前必须有注释说明核心思路。
5. b. 使用 nameof() 替代硬编码字符串注释中的类名/属性名。
6. c. 使用英文注释为主，必要时可使用中英文对照。

## **异常与日志**

1. a. 禁止使用空 catch 块；所有异常必须处理或抛出。
2. a. 日志中必须记录异常堆栈信息及上下文参数。
3. b. 优先使用结构化日志系统（如 Serilog、NLog）记录日志。
4. b. 对于已知业务错误使用自定义异常（如 BusinessException）。
5. c. try-catch 代码块应尽量小，仅包裹可能抛错语句。
6. c. 对于外部接口调用，应明确记录异常来源（如 API 名称、参数）。

## **异步与并发**

1. a. 禁止使用 .Result 或 .Wait() 阻塞异步方法。
2. a. 所有 I/O 操作必须使用异步版本（如 ReadAsync）。
3. b. 避免 async void 方法，除事件处理器外。
4. b. 支持取消操作的异步方法必须接收 CancellationToken 参数。
5. c. 并发执行任务可用 Task.WhenAll()，但需考虑异常聚合。

## **安全性与可维护性**

1. a. 禁止拼接 SQL 语句，必须使用参数化查询或 ORM。
2. a. 不得在客户端暴露敏感信息（如异常堆栈、连接字符串）。
3. b. 所有业务逻辑不得硬编码常量，应使用配置中心或枚举。
4. b. 对外 API 应统一通过中间件进行异常处理与返回格式封装。
5. c. 可使用 readonly 修饰常驻内存服务，防止误修改。

## **单元测试与代码质量**

1. a. 所有业务逻辑方法必须具备单元测试。
2. a. 单元测试必须可重复、无副作用、无外部依赖。
3. b. 使用 xUnit、NUnit 等主流测试框架，统一规范。
4. c. 测试文件命名应以 XXXTests.cs 结尾。

## **代码审查与版本控制**

1. a. 所有提交必须通过 Pull Request，经至少一名开发者 Review。
2. a. 禁止直接提交 main / master 分支。
3. b. Commit 信息应遵循约定格式（如 Conventional Commits）。
4. b. 合并前必须确保编译通过、无警告、无冲突。
5. c. 推荐使用预提交钩子（如检查格式、lint、测试）提高质量。