API接口实现优化TODO

优化目标

根据各模块接口文档，确保后端实现与接口文档完全一致，遵循DDD分层架构规范。

通用优化指令

每个模块需执行的优化步骤

第1步：对比接口文档与现有实现

1. 阅读对应的接口文档（如 `03-用户管理模块接口文档.md`）
2. 查找对应的REST控制器（如 `UserRest.java`）
3. 对比接口端点、HTTP方法、请求参数、响应结构

第2步：DTO/VO参数检查

- [ ] 检查DTO参数是否与接口文档请求体一致
- [ ] 检查VO参数是否与接口文档响应体一致
- [ ] 补充缺失的DTO类（如状态更新、批量操作等）
- [ ] 补充缺失的VO类（如操作响应、统计信息等）
- [ ] 使用 `@Schema` 注解描述所有字段
- [ ] 使用Bean Validation注解进行参数校验，**不需要添加message属性**

```
// 正确示例
@NotBlank
@Size(max = 100)
private String name;

// 错误示例（不需要message）
@NotBlank(message = "名称不能为空")
private String name;
```

第3步：日期字段规范

- [ ] 所有日期字段使用 `LocalDateTime` 类型
- [ ] 不需要手动格式化（框架自动处理）
- [ ] 检查VO中的日期字段类型

第4步：枚举类规范

- [ ] 检查接口文档中的枚举值
- [ ] 将枚举值定义为枚举类，**不需要考虑枚举属性字段和Message**

- ☐ 枚举类放到对应Entity所在目录（`domain/xxx/enums/`）

```
// 正确示例（简单枚举）
public enum UserStatus {
  ACTIVE,
  INACTIVE,
  PENDING,
  LOCKED
}

// 错误示例（不需要额外属性和message）
public enum UserStatus {
  ACTIVE("active", "活跃"),
  INACTIVE("inactive", "未激活");

  private final String code;
  private final String message;
}
```

第5步：门面层查询规范

- ☐ 门面层(Facade)查询**必须**使用 `XXXQuery` 类方法
- ☐ **禁止**门面层直接使用 `XXXRepo` 类方法
- ☐ 如Query接口缺少方法，需先在Query接口和实现类中添加

第6步：方法顺序规范

接口层、门面层、业务层统一按以下顺序排列方法：

1. **创建**（create, add, insert）
2. **更新**（update, modify, edit）
3. **修改状态**（enable/disable, lock/unlock, activate, resetPassword等）
4. **删除**（delete, remove, batchDelete）
5. **查询详细**（getDetail, findById, get）
6. **查询列表**（list, find, search）
7. **查询统计**（getStats, count, statistics）

> **注意**：其他业务操作放在统计之后，按业务逻辑分组

第7步：代码注释规范

- ☐ 不使用分组注释分隔不同类型的方法（不需要下面注解）

```
// ==================== 创建 ====================
// ==================== 更新 ====================
// ================== 修改状态 ==================
// ==================== 删除 ====================
```

```
// ==================== 查询详细 ====================
// ==================== 查询列表 ====================
// ==================== 查询统计 ====================
```

第8步：JPA JSON字段规范

- ☐ JSON字段使用对象类型，不要手动编写序列化
- ☐ 使用 `@Type(JsonType.class)` 注解

```
// 正确示例
@Type(JsonType.class)
@Column(name = "tags", columnDefinition = "json")
private List<String> tags;

@Type(JsonType.class)
@Column(name = "config", columnDefinition = "json")
private ConfigObject config;  // 使用对象类型

// 错误示例（不要手动序列化）
@Column(name = "config")
private String configJson;  // 然后手动JSON.parse/stringify
```

模块处理清单

已完成

- ☑ **03-用户管理模块** - UserRest.java ✅

待处理

认证授权模块

- ☐ **01-认证授权模块接口文档.md**
  - 控制器: 待确认
  - 主要接口: 登录、登出、刷新Token、获取权限等

租户管理模块

- ☐ **02-租户管理模块接口文档.md**
  - 控制器: TenantRest.java
  - 主要接口: 租户CRUD、租户配置、租户统计

部门管理模块

- ☐ **04-部门管理模块接口文档.md**
  - 控制器: DepartmentRest.java / DeptRest.java

- 主要接口: 部门CRUD、部门树、部门成员

组管理模块

- [ ] **05-组管理模块接口文档.md**
  - 控制器: GroupRest.java
  - 主要接口: 用户组CRUD、组成员管理

应用管理模块

- [ ] **06-应用管理模块接口文档.md**
  - 控制器: AppRest.java / ApplicationRest.java
  - 主要接口: 应用CRUD、应用配置、应用授权

服务管理模块

- [ ] **07-服务管理模块接口文档.md**
  - 控制器: ServiceRest.java
  - 主要接口: 服务CRUD、服务配置

接口管理模块

- [ ] **08-接口管理模块接口文档.md**
  - 控制器: ApiRest.java / InterfaceRest.java
  - 主要接口: API接口CRUD、接口权限

标签管理模块

- [ ] **09-标签管理模块接口文档.md**
  - 控制器: TagRest.java
  - 主要接口: 标签CRUD、标签分类

权限策略模块

- [ ] **10-权限策略模块接口文档.md**
  - 控制器: PolicyRest.java / PermissionRest.java
  - 主要接口: 权限策略CRUD、策略分配

授权管理模块

- [ ] **11-授权管理模块接口文档.md**
  - 控制器: AuthorizationRest.java
  - 主要接口: 授权CRUD、角色授权

消息通知模块

- [ ] **12-消息通知模块接口文档.md**
  - 控制器: NotificationRest.java / MessageRest.java

    - 主要接口: 消息发送、消息列表、消息已读

备份恢复模块

- [ ] **13-备份恢复模块接口文档.md**
    - 控制器: BackupRest.java
    - 主要接口: 备份创建、备份列表、恢复操作

短信消息模块

- [ ] **14-短信消息模块接口文档.md**
    - 控制器: SmsRest.java
    - 主要接口: 短信发送、短信模板、发送记录

电子邮件模块

- [ ] **15-电子邮件模块接口文档.md**
    - 控制器: EmailRest.java
    - 主要接口: 邮件发送、邮件模板、发送记录

安全设置模块

- [ ] **16-安全设置模块接口文档.md**
    - 控制器: SecurityRest.java
    - 主要接口: 安全策略、密码策略、登录策略

系统监控模块

- [ ] **17-系统监控模块接口文档.md**
    - 控制器: MonitorRest.java / SystemRest.java
    - 主要接口: 系统状态、资源监控、性能指标

接口监控模块

- [ ] **18-接口监控模块接口文档.md**
    - 控制器: ApiMonitorRest.java
    - 主要接口: 接口调用统计、性能监控

LDAP集成模块

- [ ] **19-LDAP集成模块接口文档.md**
    - 控制器: LdapRest.java
    - 主要接口: LDAP配置、同步操作、连接测试

资源配额模块

- [ ] **20-资源配额模块接口文档.md**
    - 控制器: QuotaRest.java

- 主要接口: 配额配置、配额使用、配额告警

审计日志模块

- [ ] **21-审计日志模块接口文档.md**
  - 控制器: AuditRest.java / AuditLogRest.java
  - 主要接口: 日志查询、日志导出、日志统计

系统版本模块

- [ ] **22-系统版本模块接口文档.md**
  - 控制器: VersionRest.java / SystemVersionRest.java
  - 主要接口: 版本信息、更新检查

DDD分层结构参考

```
core/src/main/java/cloud/xcan/angus/core/gm/
│
├── interfaces/           # 接口层
│   └── {module}/
│       ├── {Module}Rest.java              # REST控制器
│       └── facade/
│           ├── {Module}Facade.java        # 门面接口
│           ├── dto/                       # 请求DTO
│           │   ├── {Module}CreateDto.java
│           │   ├── {Module}UpdateDto.java
│           │   ├── {Module}FindDto.java
│           │   └── {Module}XXXDto.java    # 其他操作DTO
│           ├── vo/                        # 响应VO
│           │   ├── {Module}DetailVo.java
│           │   ├── {Module}ListVo.java
│           │   ├── {Module}StatsVo.java
│           │   └── {Module}XXXVo.java     # 其他响应VO
│           └── internal/
│               ├── {Module}FacadeImpl.java
│               └── assembler/
│                   └── {Module}Assembler.java
│
├── application/          # 应用层
│   ├── cmd/
│   │   └── {module}/
│   │       ├── {Module}Cmd.java           # 命令接口
│   │       └── impl/
│   │           └── {Module}CmdImpl.java   # 命令实现
│   └── query/
│       └── {module}/
│           ├── {Module}Query.java         # 查询接口
│           └── impl/
│               └── {Module}QueryImpl.java # 查询实现
```

```
│
├── domain/                # 领域层
│   └── {module}/
│       ├── {Module}.java                    # 领域实体
│       ├── {Module}Repo.java                # 仓储接口
│       └── enums/                           # 枚举目录
│           └── {Enum}Status.java            # 状态枚举
│
└── infra/                 # 基础设施层
    └── persistence/
        └── mysql/
            └── {module}/
                └── {Module}RepoMysql.java
```

使用说明

处理单个模块

```
请根据 {XX-模块名接口文档.md} 和 后端开发接口说明.md,
按照上述优化指令，修改 {ModuleRest.java} 接口实现。
```

示例指令

```
请根据 04-部门管理模块接口文档.md 和 后端开发接口说明.md,
对比现有的DepartmentRest.java实现，按照以下要求进行优化：
1. 检查DTO和VO参数是否有遗漏
2. 日期字段使用LocalDateTime
3. 枚举值定义成枚举类（不需要考虑枚举属性字段和Message）
4. 门面层查询使用Query类方法
5. 方法顺序调整为：创建、更新、修改状态、删除、查询详细、查询列表、查询统计
6. JPA JSON字段使用对象类型，不要手动序列化
7. DTO校验注解不需要添加message属性
```

注意事项

1. **保持一致性**: 所有模块遵循相同的规范和结构
2. **先查后改**: 修改前先充分了解现有实现
3. **逐层修改**: 按 REST → Facade → Cmd/Query 顺序修改
4. **编译验证**: 每次修改后确保编译通过
5. **保留TODO**: 未实现的业务逻辑用TODO标记

优化要求摘要

序号	优化项	说明
1	DTO/VO参数检查	与接口文档保持一致
2	日期字段	使用LocalDateTime，不手动格式化
3	枚举类	简单枚举，不需要属性字段和Message
4	门面层查询	使用XXXQuery，禁止直接使用Repo
5	方法顺序	创建→更新→修改状态→删除→查询详细→查询列表→查询统计
6	JPA JSON字段	使用对象类型 + @Type(JsonType.class)
7	DTO校验注解	不需要添加message属性

文档创建时间: 2024年
最后更新: 用户管理模块优化完成