

春景智能 QMES SaaS 平台 — 软件功能详细设计

文档版本: V1.0

编写日期: 2025-07-17

覆盖产品: QMES / QMS / 设备数采 / ESOP / 安灯 / AI / 联动版

目录

- [1. 第一性原理设计](#)
- [2. 通用设计规范](#)
- [3. QMES 功能详细设计](#)
- [4. QMS 功能详细设计](#)
- [5. 设备数采功能详细设计](#)
- [6. ESOP 功能详细设计](#)
- [7. 安灯功能详细设计](#)
- [8. AI 功能详细设计](#)
- [9. 联动版功能详细设计](#)
- [10. BOSS/运营功能详细设计](#)

1. 第一性原理设计

详细设计核心命题

每个功能模块详细设计回答一个核心决策问题: "这个模块的代码实现如何帮助用户做出更好的决策?"

模块	实现核心	用户决策收益
工单状态机	5个状态的严谨流转 + 进度自动计算	车间主任知道"哪个工单卡住了"
OEE计算	可用性×性能×质量 三因子模型	设备主管知道"哪台设备效率最低"
合格率趋势	按日/周/月自动聚合 + 连续下降检测	质量经理知道"质量是否在恶化"
安灯响应时间	从呼叫到响应的分钟级精确计算	管理层知道"安灯响应KPI是否达标"
AI视觉检测	异步调用AI模型 + 回调结果	质检主管知道"AI发现了多少缺陷"

模块	实现核心	用户决策收益
全链路追溯	工单→报工→检验→入库 时间线聚合	审核员一键验证"产品完整履历"

2. 通用设计规范

2.1 API响应格式

所有API统一响应格式：

```
{
  "code": 200,
  "msg": "操作成功",
  "data": { ... }
}
```

2.2 分页规范

```
{
  "list": [...],
  "total": 100,
  "page": 1,
  "page_size": 15,
  "total_pages": 7
}
```

2.3 控制器通用模式

```
class XxxController extends Controller
{
  public function __construct()
  {
    $this->middleware('permission:qmes:xxx:list')->only(['index', 'show']);
    $this->middleware('permission:qmes:xxx:create')->only(['store']);
    $this->middleware('permission:qmes:xxx:edit')->only(['update']);
    $this->middleware('permission:qmes:xxx:delete')->only(['destroy']);
  }

  public function index(Request $request): JsonResponse
  {
    $tenantId = getTenantId();
    $query = Model::byTenant($tenantId);
    // 筛选条件...
    $result = paginate($query, $request->get('page_size', 15));
    return success($result);
  }
}
```

```
}  
}
```

2.4 Vue前端组件规范

```
<template>  
  <div class="xxx-container">  
    <!-- 搜索区域 -->  
    <el-form :model="queryParams" inline>  
      <el-form-item label="关键字">  
        <el-input v-model="queryParams.keyword" />  
      </el-form-item>  
      <el-button type="primary" @click="fetchData">查询</el-button>  
    </el-form>  
  
    <!-- 表格区域 -->  
    <el-table :data="list" v-loading="loading">  
      <el-table-column prop="xxx" label="xxx" />  
    </el-table>  
  
    <!-- 分页 -->  
    <el-pagination @current-change="handlePageChange" />  
  </div>  
</template>
```

3. QMES 功能详细设计

3.1 工单管理 (WorkOrderController)

3.1.1 状态机



cancelled: pending → cancelle (取消)

3.1.2 工单号生成算法

```
public static function generateOrderNo($tenantId): string
{
    $prefix = 'W0';
    $date = now()->format('Ymd');
    $lastOrder = self::byTenant($tenantId)
        ->where('order_no', 'like', "{$prefix}{$date}%")
        ->orderBy('id', 'desc')
        ->first();

    $seq = $lastOrder
        ? str_pad((int)substr($lastOrder->order_no, -4) + 1, 4, '0', STR_PAD_LEFT)
        : '0001';

    return "{$prefix}{$date}{$seq}";
}
// 示例: W0202507170001
```

3.1.3 报工逻辑

```
public function report(Request $request): JsonResponse
{
    DB::transaction(function () {
        // 1. 创建 WorkReport 记录
        WorkReport::create([...]);

        // 2. 更新工单完成数量
        $workOrder->increment('completed_qty', $request->qty);
        $workOrder->increment('defect_qty', $defectQty);

        // 3. 如果全部完成, 自动完工
        if ($workOrder->completed_qty >= $workOrder->qty) {
            $workOrder->complete();
        }
    });
}
```

3.1.4 进度计算

```
public function getProgressAttribute(): float
{
    if ($this->qty <= 0) return 0;
```

```
return round(($this->completed_qty / $this->qty) * 100, 1);
}
```

3.2 产品管理 (ProductController)

3.2.1 BOM结构

```
[
  {"material_code": "RAW-001", "material_name": "钢材A", "qty": 2.5, "unit": "kg"},
  {"material_code": "RAW-002", "material_name": "螺丝M6", "qty": 10, "unit": "个"},
  {"material_code": "RAW-003", "material_name": "轴承6205", "qty": 2, "unit": "个"}
]
```

3.2.2 BOM物料计数

```
$bom = $product->bom_json;
$product->bom_count = is_array($bom) ? count($bom) : 0;
$product->routing_count = ProductRouting::byTenant($tenantId)
    ->where('product_id', $product->id)
    ->count();
```

3.3 生产排程 (ProductionScheduleController)

3.3.1 排程生成

```
public function generate(Request $request): JsonResponse
{
    // 根据工单的计划时间、产品优先级、资源约束
    // 自动生成最优排程方案

    // 按优先级排序 pending 工单
    $pendingOrders = WorkOrder::byTenant($tenantId)
        ->where('status', 'pending')
        ->orderBy('priority')
        ->orderBy('planned_start')
        ->get();

    // 为每个工单分配日期和资源
    // ...
}
```

3.4 甘特图 (WorkOrderController@gantt)

数据源:

- start_date ~ end_date 范围内有 planned_start/planned_end 的工单
- 每个工单: id, order_no, product_name, qty, completed_qty, status, priority, planned_start, planned_end

颜色映射:

状态	颜色
-----	-----
pending	#909399 (灰色)
in_progress	#409EFF (蓝色)
completed	#67C23A (绿色)
closed	#909399 (灰色)
cancelled	#F56C6C (红色)

4. QMS 功能详细设计

4.1 检验管理 (InspectionController)

4.1.1 检验阶段

stage	中文	说明
iqc	来料检验	供应商到货后、入库前
ipqc	过程检验	生产过程巡检和首件检验
oqc	出货检验	成品入库/发货前最终检验

4.1.2 检验单号生成

```
public static function generateInspectionNo($tenantId, $stage): string
{
    // INS + 阶段 + 日期 + 4位序号
    // 示例: INSIQC202507170001
    $prefix = 'INS' . strtoupper($stage);
    $date = now()->format('Ymd');
    // ...
}
```

4.1.3 检验项目结构

```
[
  {
    "item_name": "外观检查",
    "spec_value": "无划痕、无变形",
    "tolerance": "",
    "unit": "",
    "actual_value": "合格",
    "judge": "pass"
  },
  {
```

```

    "item_name": "尺寸A",
    "spec_value": "100",
    "tolerance": "±0.5",
    "unit": "mm",
    "actual_value": "100.2",
    "judge": "pass"
  },
  {
    "item_name": "硬度",
    "spec_value": "HRC 58-62",
    "tolerance": "",
    "unit": "HRC",
    "actual_value": "60",
    "judge": "pass"
  }
]

```

4.2 质量趋势分析

4.2.1 聚合算法

```

// 按阶段 + 时间段分组
$records = InspectionRecord::byTenant($tenantId)
    ->whereBetween('inspected_at', [$startDate, $endDate])
    ->get();

// 按日期/周/月分组
$grouped = $records->groupBy(fn($item) =>
    match($period) {
        'weekly' => Carbon::parse($item->inspected_at)->format('Y-W'),
        'monthly' => Carbon::parse($item->inspected_at)->format('Y-m'),
        default => Carbon::parse($item->inspected_at)->format('Y-m-d'),
    }
);

// 计算每组合合格率
foreach ($grouped as $label => $items) {
    $passRate = $items->where('result', 'pass')->count() / $items->count() * 100;
}

```

4.2.2 连续下降预警

```

$consecutiveDrop = 0;
for ($i = 0; $i < $rates->count() - 1; $i++) {
    $drop = $rates[$i + 1] - $rates[$i]; // 注意是逆序
    if ($drop < -$dropThreshold) {
        $consecutiveDrop++;
    } else {
        $consecutiveDrop = 0;
    }
}

```

```
}
if ($consecutiveDrop >= $minDropBatches) {
    // 推送企微告警
    qmes_boss_push($tenantId, 'quality_decline', $alertMsg);
}
```

4.3 NCR不合格品管理 (NcrController)

4.3.1 NCR处置方式

action	中文	说明
scrap	报废	缺陷不可修复，直接报废
rework	返工	缺陷可修复，返回生产线修复
return	退货	来料缺陷，退货给供应商
accept	让步接收	缺陷不影响功能，让步接收

4.3.2 NCR关联

```
WorkOrder (source_type='work_order')
└─ NcrRecord.source_type='work_order', source_id=work_order.id

InspectionRecord (source_type='inspection')
└─ NcrRecord.source_type='inspection', source_id=inspection.id
```

5. 设备数采功能详细设计

5.1 设备与网关管理

5.1.1 设备生命周期

```
创建(offline) → 绑定网关 → 网关上线 → 设备上线(online)
→ 采集数据开始 → 设备告警(alarm) → 处理告警 → 恢复(online)
→ 网关离线 → 设备离线(offline) → 删除
```

5.1.2 设备状态映射

```
public static array $statusMap = [
    'offline' => '离线',
    'online' => '在线',
    'alarm' => '告警',
];
```

5.2 设备数据 (DeviceData)

5.2.1 数据存储

表结构特点:

- device_data 是最大的表, 预计10亿+行
- 按 (device_id, point_id, ts) 复合索引查询
- 高频写入, 低频查询

5.2.2 最新数据获取

```
// 获取设备每个点位的最新数据
$latestData = DeviceData::byTenant($tenantId)
    ->where('device_id', $id)
    ->whereIn('point_id', $device->points->pluck('id'))
    ->selectRaw('point_id, value, ts')
    ->orderBy('ts', 'desc')
    ->get()
    ->groupBy('point_id')
    ->map(fn($items) => $items->first());
```

5.3 设备告警 (DeviceAlert)

5.3.1 告警级别

level	中文	响应时间要求
warning	警告	30分钟内确认
critical	严重	15分钟内确认
emergency	紧急	5分钟内确认

5.3.2 告警升级

```
// 定时任务: BossPushCheckController 检查未处理的告警
$unresolvedAlerts = DeviceAlert::where('status', 'new')
    ->where('created_at', '<', now()->subMinutes(30))
    ->get();

foreach ($unresolvedAlerts as $alert) {
    // 升级推送
    $content = "...";
    qmes_boss_push($tenantId, 'device_offline', $content);
}
```

6. ESOP 功能详细设计

6.1 SOP文档模型

6.1.1 结构内容

```
[
  {"title": "操作准备", "body": "1. 穿戴好防护用品\n2. 检查设备状态\n3. 准备所需工具"},
  {"title": "装夹工件", "body": "1. 将工件放置在夹具上\n2. 用扳手拧紧固定螺丝\n3. 检查装夹是否牢固"},
  {"title": "启动加工", "body": "1. 关闭防护门\n2. 按下启动按钮\n3. 观察首件加工情况"}
]
```

6.1.2 版本管理

每次发布自动创建版本快照：

```
public function publish(Request $request, int $id): JsonResponse
{
    SopVersion::create([
        'sop_id' => $sop->id,
        'version' => $sop->version,
        'content' => $sop->content,
        'change_log' => $request->change_log,
    ]);

    $sop->update([
        'status' => 'published',
        'version' => $sop->version + 1,
        'published_at' => now(),
    ]);
}
```

6.2 工位SOP查询

```
public function getWorkstationSop(int $workstationId): JsonResponse
{
    // 工位关联产品/工序 → 查找已发布且关联的SOP
    $sop = SopDocument::byTenant($tenantId)
        ->where('status', 'published')
        ->where(function ($q) use ($workstationId) {
            $q->where('process_id', $workstationId)
                ->orWhere('product_id', $workstationId);
        })
        ->orderBy('version', 'desc')
        ->first();

    return success($sop);
}
```

7. 安灯功能详细设计

7.1 呼叫流程

操作工发起呼叫 (POST /andons)

- 状态: pending
- 系统记录: call_no, caller_id, workstation_id, call_type, priority, title

班组长/维修工响应 (PUT /andons/{id}/respond)

- 状态: processing
- 记录: responder_id, response_at

问题解决 (PUT /andons/{id}/resolve)

- 状态: completed
- 记录: resolved_at

超时未响应 → 系统自动升级通知

7.2 响应时间计算

```
// 平均响应时间
$avgResponse = AndonCall::byTenant($tenantId)
    ->whereNotNull('response_at')
    ->selectRaw('AVG(TIMESTAMPDIFF(MINUTE, created_at, response_at)) as avg_minutes')
    ->first();

// P95响应时间
$records = AndonCall::byTenant($tenantId)
    ->whereNotNull('response_at')
    ->selectRaw('TIMESTAMPDIFF(MINUTE, created_at, response_at) as response_minutes')
    ->pluck('response_minutes')
    ->sort()
    ->values();

$p95Index = (int) ceil($total * 0.95) - 1;
$p95 = $records[$p95Index];
```

7.3 呼叫类型分布

```
public static array $callTypeMap = [
    'quality' => '质量异常',
    'machine' => '设备故障',
    'material' => '物料短缺',
    'other' => '其他',
];

public static array $priorityMap = [
    'high' => '紧急',
```

```
'normal' => '普通',  
'low' => '低',  
];
```

8. AI 功能详细设计

8.1 AI视觉检测流程

前端上传图片 → AiVisionController@submitTask

- 创建任务 (status=running)
- 调用第三方AI API (POST /detect)
- 同步检测: 直接返回结果
- 异步检测: AI回调 /api/ai-vision/tasks/callback
- 更新任务结果 (status=completed/failed)

8.2 任务类型

task_type	中文	用途
defect_detect	缺陷检测	检测产品表面缺陷 (划痕/裂纹/气孔)
assembly_check	装配检测	检测装配是否正确 (缺件/错装)
sop_qa	SOP问答	根据图片回答操作是否正确

8.3 AI API调用

```
$response = Http::timeout(60)  
->withHeaders([  
    'Authorization' => 'Bearer ' . $aiApiKey,  
)  
)  
->post($aiApiUrl . '/detect', [  
    'task_id' => $task->id,  
    'task_type' => $request->task_type,  
    'image_url' => $request->image_url,  
    'callback_url' => url('/api/ai-vision/tasks/callback'),  
]);
```

8.4 批量检测

```
// 一次提交最多50张图片  
public function batchSubmit(Request $request): JsonResponse  
{  
    foreach ($request->images as $imageUrl) {  
        // 为每张图片创建任务并提交检测  
        // 支持批量并发调用  
    }  
}
```

```
}
return success([
  'total' => count($tasks),
  'success' => $succCount,
  'failed' => $failCount,
]);
}
```

9. 联动版功能详细设计

9.1 全链路追溯 (IntegratedController@trace)

追溯链路:

输入：工单号/批次号/产品条码

↓

找到关联工单 WorkOrder

↓

列出该工单的所有：

- ├ 生产报工事件 (WorkReport)
- ├ 质量检验事件 (InspectionRecord)
- ├ NCR记录 (NcrRecord)
- └ 仓储出入库事件 (InventoryRecord)

↓

按时间排序 → 返回时间线

时间线事件格式:

```
{
  "type": "production",
  "type_text": "生产报工",
  "time": "2025-07-17 14:30:00",
  "operator": "张三",
  "detail": "报工数量: 100, 良品: 98, 不良品: 2",
  "source_id": 1024
}
```

9.2 综合看板 (IntegratedController@summary)

返回数据:

```
{
  "production": {
    "today_qty": 1500,
    "today_good": 1470,
    "today_defect": 30,
    "yield_rate": 98.0,
  }
}
```

```
"total_orders": 12,
"completed_orders": 5,
"in_progress_orders": 5
},
"quality": {
  "total_inspections": 45,
  "pass_count": 42,
  "fail_count": 3,
  "pass_rate": 93.3
},
"warehouse": {
  "receive_qty": 5000,
  "ship_qty": 3000,
  "material_count": 120,
  "total_qty": 25000
}
}
```

9.3 端到端流程 (IntegratedController@flow)

```
// 每个工单的状态
$flowStatus = [
  'production' => ['status' => 'pending/processing/done', 'text' => '...'],
  'quality'    => ['status' => 'pending/done', 'text' => '待检验/合格/不合格'],
  'warehouse'  => ['status' => 'pending/ready/done', 'text' => '未入库/待入库/已入库'],
];
```

9.4 追溯报告导出 (IntegratedController@exportTraceReport)

生成包含工单信息、报工记录、检验记录、NCR记录、出入库记录的HTML报告，浏览器打印为PDF。

10. BOSS/运营功能详细设计

10.1 老板企微推送 (BossAlertController)

10.1.1 推送配置

```
{
  "name": "车间告警推送",
  "webhook_url": "https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=xxx-xxx",
  "events": [
    "device_offline",
    "quality_decline",
    "low_stock",
    "work_order_overdue",
    "andon_timeout"
  ],
}
```

```
"enabled": true
}
```

10.1.2 推送检查定时任务

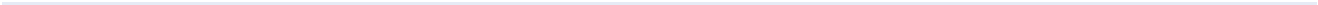
检查项	接口	频率
设备离线检查	BossPushCheckController@checkDeviceOffline	每5分钟
工单超时检查	BossPushCheckController@checkWorkOrderOverdue	每30分钟
安灯超时检查	BossPushCheckController@checkAndonTimeout	每5分钟
库存告警检查	BossPushCheckController@checkLowStock	每30分钟

10.2 运营看板 (OperationController)

接口	数据	决策
summary	总租户数/活跃租户/今日新增/总收入/月收入/流失率	平台运营状况一览
tenantTrend	按日/周/月的新增租户趋势	业务增长趋势
revenueTrend	月度收入趋势	营收增长健康度
tenantRank	按用户数/工单数排名的活跃租户TOP20	头部客户运营
systemHealth	数据库状态/平均响应时间/错误率/在线率	系统运行健康度

10.3 版本发布 (ReleaseController)

功能	说明
创建发布	填写版本号、标题、描述、更新内容
审核	平台内部审核发布内容
发布	正式发布版本
回滚	出现问题时可回滚到上一版本



11. 条码标签管理 (BarcodeLabelController)

11.1 功能概述

条码标签管理提供条码规则定义、条码生成、标签模板设计、批量打印功能，支持产品/工单/物料/设备/库位的条码化管理。

11.2 功能列表

功能	说明	控制器
条码规则管理	定义条码编码规则(前缀/分隔符/日期/流水号)	BarcodeRuleController
条码解析	扫描条码自动识别实体类型	BarcodeRuleController@parse
标签模板设计	可视化设计打印模板(尺寸/字段/二维码)	LabelTemplateController
条码生成	按规则生成条码/二维码	BarcodeController@generate
批量打印	多实体批量打印标签	BarcodeController@batchPrint
扫码记录	记录扫码历史用于追溯	BarcodeScanController

11.3 用户操作流程

【管理员】配置条码规则

- 设置编码规则(前缀+日期+流水号)
- 选择关联实体类型(产品/工单/物料)
- 设置条码长度和校验位
- 保存规则

【管理员】设计标签模板

- 选择模板尺寸(40x30mm / 60x40mm / 100x60mm)
- 拖入字段(产品名称/编码/批次/日期)
- 生成二维码/条码
- 预览并保存模板

【操作工】条码打印

- 选择实体(产品/工单清单)
- 选择标签模板
- 预览标签
- 执行批量打印(按打印份数和起始序号)
- 标签输出到打印机

【操作工】扫码操作

- 使用PDA/扫码枪扫描条码
- 系统解析条码 → 识别实体
- 跳转到对应业务操作页面(报工/检验/入库)

11.4 数据流

BarcodeRule (条码规则)

↓ 关联

Product / WorkOrder / Material (实体)

↓ 生成

BarcodeRecord (条码记录) ← BarcodePrintJob (打印任务)

↓ 扫描

BarcodeScanLog (扫码日志)

11.5 条码生成算法

```
public function generateBarcode(string $ruleCode, array $entityData): string
{
    $rule = BarcodeRule::byTenant(getTenantId())
        ->where('rule_code', $ruleCode)->firstOrFail();

    $parts = [];
    foreach ($rule->segments as $segment) {
        switch ($segment['type']) {
            case 'prefix':
                $parts[] = $segment['value'];
                break;
            case 'date':
                $parts[] = now()->format($segment['format'] ?? 'Ymd');
                break;
            case 'serial':
                $parts[] = str_pad($entityData['serial'], $segment['length'], '0', STR_PAD_LE);
                break;
            case 'field':
                $parts[] = $entityData[$segment['field']] ?? '';
                break;
        }
    }

    $barcode = implode($rule->separator ?? '-', $parts);

    // 校验位
    if ($rule->has_check_digit) {
        $barcode .= $this->calculateCheckDigit($barcode);
    }

    return $barcode;
}
```

12. 消息通知预警中心 (MessageCenterController)

12.1 功能概述

消息通知预警中心提供统一的消息通道管理、通知规则引擎、消息推送、预警监控功能，将系统事件通过多种渠道(企微/钉钉/飞书/站内)推送到对应人员。

12.2 功能列表

功能	说明	控制器
消息通道管理	配置企微/钉钉/飞书/自定义Webhook通道	MessageChannelController
通知规则引擎	基于事件/级别/来源/时间/频率的条件匹配	NotificationRuleController
消息记录	查看/搜索/筛选历史消息	MessageController
消息模板	定义各类事件的推送模板	MessageTemplateController
预警监控	基于阈值的异常预警配置	AlertConfigController
推送统计	消息送达率/成功率统计	MessageStatsController

12.3 用户操作流程

【管理员】配置消息通道

- 选择通道类型(企微机器人/钉钉/飞书/自定义HTTP)
- 填写Webhook URL
- 选择订阅事件类型
- 设置频率限制
- 测试推送 → 启用通道

【管理员】配置通知规则

- 选择事件类型(设备离线/质量下降/库存告警)
- 设置条件(级别=critical / 时间=工作时间)
- 选择推送通道
- 设置频率控制(同一设备10分钟内不重复)
- 保存规则

【管理员】配置预警阈值

- 选择监控指标(设备离线时长/质量合格率/库存量)
- 设置阈值(80% / 10件)
- 选择预警级别(警告/严重)
- 关联通知规则

【用户】查看消息

- 进入消息中心
- 按事件类型/级别/时间筛选
- 查看消息详情
- 确认已读/标记处理

12.4 数据流

系统事件触发 (设备离线/质量下降/安灯超时/库存告警)

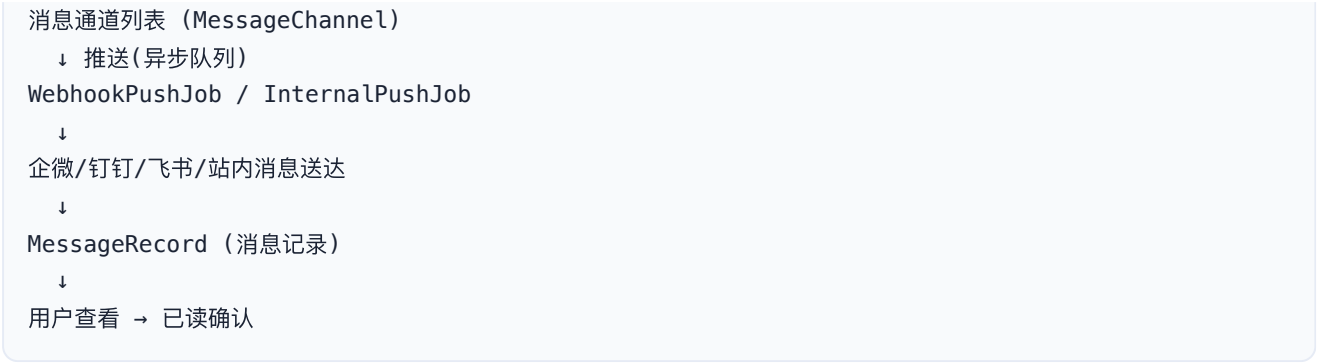
↓

MessageCenterService@dispatch

↓

通知规则引擎评估 (NotificationRule)

↓ 匹配



12.5 消息模板渲染

```
// 消息模板支持占位符替换
$template = "【{event_type_text}】{title}\n来源: {source_name}\n内容: {content}\n时间: {ts}";
$rendered = str_replace(
    ['{event_type_text}', '{title}', '{source_name}', '{content}', '{ts}'],
    [$eventTypeText, $title, $sourceName, $content, $ts],
    $template
);
```

13. 报表BI看板引擎 (BiDashboardController)

13.1 功能概述

报表BI看板引擎提供可视化数据分析能力，支持自定义图表、看板布局、数据聚合、看板分享，帮助管理层通过数据驱动决策。

13.2 功能列表

功能	说明	控制器
数据源管理	配置BI数据源(模型/字段/聚合方式)	BiDataSourceController
图表配置	创建/编辑折线图/柱状图/饼图/表格等	BiChartController
看板布局	拖拽式看板布局设计	BiDashboardController
看板分享	生成分享链接/嵌入外部页面	BiDashboardController@share
数据刷新	按周期自动刷新/手动刷新	BiDataController@refresh
数据导出	导出图表数据为CSV/Excel	BiDataController@export
权限管理	看板可见范围/数据行权限	BiDashboardController@permission

13.3 用户操作流程

【管理员】创建图表

- 选择数据源(WorkReport/InspectionRecord/DeviceData等)
- 选择图表类型(折线图/柱状图/饼图/数字卡片)
- 配置维度字段(日期/产品/部门)
- 配置指标字段(产量/合格率/告警数) + 聚合方式(sum/avg/count)
- 设置筛选条件(状态/时间范围)
- 预览图表 → 保存

【管理员】设计看板

- 创建看板(名称/描述/布局类型)
- 添加图表到看板
- 拖拽调整位置和大小
- 设置刷新间隔(5分钟/15分钟/1小时)
- 设置可见范围(全部用户/指定角色)
- 保存看板布局

【管理员】分享看板

- 选择看板 → 点击分享
- 生成分享链接(公开/加密)
- 可嵌入外部系统(iframe)
- 分享给指定用户

【管理层】查看看板

- 进入BI看板列表
- 选择看板
- 查看图表数据
- 点击图表下钻查看明细
- 导出数据

13.4 数据流

数据源配置 (BiDataSource)

↓ 引用

图表配置 (BiChart) → 维度/指标/筛选

↓ 放入

看板布局 (BiDashboard) → 图表列表+布局坐标

↓ 渲染

GET /bi/data/{chartId} → 数据聚合查询 → 图表渲染

↓ 分享

BiDashboardShare (分享链接+权限)

13.5 看板分享实现

```
public function share(int $dashboardId, Request $request): JsonResponse
{
    $dashboard = BiDashboard::byTenant(getTenantId())->findOrFail($dashboardId);

    $share = BiDashboardShare::create([
        'dashboard_id' => $dashboard->id,
```

```
'share_token' => Str::random(32),
'open_access' => $request->open_access ?? false,
'password' => $request->password ? bcrypt($request->password) : null,
'expire_at' => $request->expire_days
    ? now()->addDays($request->expire_days)
    : null,
'embed_enabled' => $request->embed_enabled ?? false,
]);

$shareUrl = url("/bi/shared/{$_share->share_token}");

return success([
    'share_token' => $_share->share_token,
    'share_url' => $shareUrl,
    'embed_url' => $_share->embed_enabled
        ? url("/bi/embed/{$_share->share_token}") : null,
    'expire_at' => $_share->expire_at,
]);
}
```

13.6 图表类型与聚合方式

图表类型	支持聚合	适用场景
line(折线图)	sum/avg/count	日产量趋势、合格率趋势
bar(柱状图)	sum/count	部门产量对比、产品排名
pie(饼图)	count/sum	告警类型分布、缺陷占比
table(表格)	-	明细数据展示
number(数字卡片)	sum/avg/count/last	今日产量、合格率、设备数
gauge(仪表盘)	当前值+目标值	产能达成率、质量目标

14. 追溯链全景图 (TraceChainController)

14.1 功能概述

追溯链全景图提供从原材料→生产→检验→入库→出货的全链路可视化追溯能力，支持按条码/批次/工单维度追溯，以图谱形式展示产品完整生命周期。

14.2 功能列表

功能	说明	控制器
追溯查询	按条码/批号/工单号查询追溯链路	TraceChainController@query

功能	说明	控制器
链路图谱	可视化展示上下游关系(图谱/时间线/表格)	TraceChainController@graph
节点详情	点击节点查看详细数据	TraceChainController@nodeDetail
链路分析	异常节点标记/瓶颈分析	TraceChainController@analysis
报告导出	导出追溯报告(HTML/PDF)	TraceChainController@export
告警关联	追溯链路中标记告警/异常节点	TraceChainController@alerts

14.3 用户操作流程

【质检员】追溯查询

- 输入条码/批次号/工单号(支持扫码输入)
- 系统解析并查找关联数据
- 展示追溯链路图谱

【质检员】查看链路

- 图谱模式：展示原材料→生产→检验→入库链路
 - └ 每个节点显示状态(正常/异常/待检)
 - └ 异常节点红色高亮
 - └ 点击节点展开详情
- 时间线模式：按时间顺序排列所有事件
- 表格模式：列表展示所有节点

【质检员】节点详情

- 点击"生产报工"节点 → 显示报工时间/数量/操作人
- 点击"质量检验"节点 → 显示检验项目/结果/判定
- 点击"NCR"节点 → 显示缺陷描述/处置方式
- 点击"入库"节点 → 显示入库时间/库位/数量

【质检员】导出报告

- 设置报告范围(全链路/指定节点)
- 选择导出格式(HTML/PDF)
- 包含追溯编号/时间戳/签名栏
- 导出并下载

14.4 数据流

输入：条码/批次号/工单号

↓

BarcodeRule解析 / 直接查询

↓

查找核心实体 (ProductBatch / WorkOrder)

↓

递归上下游关联：

- └ 原材料 (MaterialReceipt → MaterialLot)
- └ 生产过程 (WorkOrder → WorkReport)

├─ 质量检验 (InspectionRecord → NcrRecord)
├─ 仓储流转 (InventoryRecord)
└─ 设备关联 (Device × WorkOrder)

↓

构建链路图 (有序节点 + 边关系)

↓

渲染展示 / 导出报告

14.5 链路图数据结构

```
{
  "trace_id": "TRC-20250717-001",
  "root_entity": {
    "type": "work_order",
    "id": 1024,
    "code": "W0202507170001",
    "name": "精密轴承组件-500套"
  },
  "nodes": [
    {
      "id": "node_1",
      "type": "material_receipt",
      "type_text": "原材料入库",
      "label": "轴承钢-500kg",
      "status": "normal",
      "ts": "2025-07-15 08:00:00"
    },
    {
      "id": "node_2",
      "type": "production",
      "type_text": "生产报工",
      "label": "报工-车削工序",
      "status": "normal",
      "ts": "2025-07-16 14:30:00"
    },
    {
      "id": "node_3",
      "type": "inspection",
      "type_text": "质量检验",
      "label": "IPQC巡检-合格",
      "status": "normal",
      "ts": "2025-07-16 15:00:00"
    },
    {
      "id": "node_4",
      "type": "warehouse",
      "type_text": "成品入库",
      "label": "入库-成品仓A01",
      "status": "normal",
      "ts": "2025-07-17 09:00:00"
    }
  ]
}
```

```
],
"edges": [
  {"from": "node_1", "to": "node_2", "label": "投料"},
  {"from": "node_2", "to": "node_3", "label": "送检"},
  {"from": "node_3", "to": "node_4", "label": "合格入库"}
],
"summary": {
  "total_nodes": 4,
  "normal_count": 4,
  "abnormal_count": 0,
  "pending_count": 0,
  "duration_hours": 49
}
}
```

15. 模具刀具管理 (MoldToolController)

15.1 功能概述

模具刀具管理提供模具/刀具的全生命周期管理，包括档案管理、使用记录、寿命追踪、维修保养、库存管理功能。

15.2 功能列表

功能	说明	控制器
模具/刀具档案	创建/编辑模具刀具基本信息(编号/类型/规格/寿命)	MoldToolController
分类管理	模具分类(冲压模/注塑模/压铸模)和刀具分类(车刀/铣刀/钻头)	MoldToolCategoryController
领用归还	模具刀具领用/归还/转交记录	MoldToolLendController
寿命追踪	按使用次数/加工时间追踪剩余寿命	MoldToolLifeController
保养计划	制定保养计划/记录保养执行	MoldToolMaintainController
维修记录	记录维修时间/内容/费用	MoldToolRepairController
库存管理	模具刀具库存/入库/出库/盘点	MoldToolInventoryController
报废管理	模具刀具报废审批流程	MoldToolScrapController

15.3 用户操作流程

- 【管理员】创建模具/刀具档案
- 填写模具编号/名称/类型/规格/供应商
 - 设置额定寿命(使用次数/加工时长)
 - 上传图纸/照片附件
 - 设置存放位置 → 保存

【操作工】领用模具

- 扫描模具条码/搜索模具
- 查看模具状态(在库/使用中/维修中)
- 填写领用用途/预计归还时间
- 确认领用 → 模具状态更新为"使用中"

【操作工】归还模具

- 扫描模具条码
- 录入使用次数/加工时长
- 检查模具状态(正常/需保养/需维修)
- 确认归还 → 累计寿命 + 更新库存位置

【管理员】寿命追踪

- 查看模具/刀具寿命列表
- 筛选临近寿命预警(剩余<10%)
- 查看单个模具寿命使用曲线
- 设置寿命预警阈值
- 到达寿命 → 自动触发保养/报废提醒

【管理员】保养计划

- 创建保养计划(模具/周期/保养项目)
- 设置保养提醒提前天数
- 到期自动推送保养提醒
- 记录保养执行情况
- 查看保养历史

15.4 数据流

MoldTool (模具刀具档案)

- ├─ MoldToolCategory (分类)
- ├─ MoldToolLife (寿命记录: 使用次数/时长/剩余%)
- ├─ MoldToolLend (领用归还: 领用人/时间/用途)
- ├─ MoldToolMaintain (保养: 计划/执行/下次保养)
- ├─ MoldToolRepair (维修: 故障/维修内容/费用)
- ├─ MoldToolInventory (库存: 入库/出库/盘点)
- └─ MoldToolScrap (报废: 审批/日期/原因)

15.5 寿命计算

```
// 模具寿命追踪
public function calculateLifeRemaining(MoldTool $mold): array
{
    // 按使用次数
    if ($mold->life_type === 'count') {
        $usedCount = MoldToolLend::where('mold_tool_id', $mold->id)
            ->sum('usage_count');
        $remainingPercent = $mold->rated_life > 0
            ? (($mold->rated_life - $usedCount) / $mold->rated_life) * 100
            : 100;
    }
}
```

```
// 按加工时长(小时)
if ($mold->life_type === 'hours') {
    $usedHours = MoldToolLend::where('mold_tool_id', $mold->id)
        ->sum('usage_hours');
    $remainingPercent = $mold->rated_life > 0
        ? (($mold->rated_life - $usedHours) / $mold->rated_life) * 100
        : 100;
}

$status = $remainingPercent <= 0 ? 'expired'
    : ($remainingPercent <= $mold->warning_threshold ? 'warning' : 'normal');

return [
    'used' => $usedCount ?? $usedHours,
    'rated' => $mold->rated_life,
    'remaining_percent' => round(max(0, $remainingPercent), 1),
    'status' => $status,
];
}
```

16. 文档管理DMS (DocumentController)

16.1 功能概述

文档管理系统(DMS)提供企业文档的集中存储、分类、版本管理和权限控制，支持生产相关文档(SOP/图纸/工艺文件/质检标准)的统一管理。

16.2 功能列表

功能	说明	控制器
文档目录	树形目录结构(文件夹/子文件夹)	DocumentCategoryController
文档上传	上传文档(支持PDF/Word/Excel/图片/视频)	DocumentController@upload
文档预览	在线预览(PDF/Office/图片)	DocumentController@preview
版本管理	文档版本历史/差异对比/版本回滚	DocumentController@versions
审批流程	文档发布审批(起草→审核→发布→归档)	DocumentApprovalController
权限控制	文件夹/文档级别的读写权限	DocumentController@permission
全文搜索	按文件名/标签/内容搜索文档	DocumentController@search
文档水印	预览时自动添加水印(用户名/时间)	DocumentController@preview
文档关联	文档与产品/工单/设备关联	DocumentController@link

16.3 用户操作流程

【管理员】创建文档目录

- 创建根文件夹(技术文档/质量文件/工艺文件)
- 创建子文件夹(SOP/图纸/检验标准/设备手册)
- 设置文件夹权限(可见角色/可编辑角色)

【用户】上传文档

- 选择目标文件夹
- 上传文件(支持拖拽/批量上传)
- 填写文档属性(标题/编号/版本/标签)
- 选择关联实体(产品/工单/设备)
- 设置审批流程(可选)
- 保存 → 文档状态为"草稿"

【审批人】文档审批

- 查看待审批文档列表
- 预览文档内容
- 填写审批意见
- 通过/驳回
- 通过后文档状态更新为"已发布"

【用户】查看文档

- 浏览目录树
- 搜索文档(关键词/标签/编号)
- 预览文档(在线预览/水印)
- 下载文档(权限控制)
- 查看版本历史
- 关联查看(从产品/工单查看关联文档)

【管理员】文档归档

- 选择已发布文档
- 归档操作
- 归档文档只读不可编辑
- 保留归档历史版本

16.4 数据流

DocumentCategory (文档目录)

↓ 包含

Document (文档)

- ├─ DocumentVersion (版本: v1/v2/v3, 文件存储路径)
- ├─ DocumentApproval (审批: 起草→审核→发布→归档)
- ├─ DocumentPermission (权限: 角色/用户级别)
- ├─ DocumentTag (标签: 类型/分类/关键词)
- └─ DocumentLink (关联: product_id/work_order_id/device_id)

16.5 版本管理实现

```
// 上传新版本时自动创建版本记录
public function uploadNewVersion(Request $request, int $docId): JsonResponse
{
    $doc = Document::byTenant(getTenantId())->findOrFail($docId);

    // 保存文件
    $path = $request->file('file')->store("documents/{$doc->id}");

    // 创建版本记录
    $version = DocumentVersion::create([
        'document_id' => $doc->id,
        'version' => $doc->current_version + 1,
        'file_path' => $path,
        'file_name' => $request->file('file')->getClientOriginalName(),
        'file_size' => $request->file('file')->getSize(),
        'mime_type' => $request->file('file')->getMimeType(),
        'change_log' => $request->change_log,
        'uploaded_by' => auth()->id(),
    ]);

    // 更新文档
    $doc->update([
        'current_version' => $version->version,
        'file_path' => $path,
        'file_name' => $version->file_name,
    ]);

    return success($version);
}
```

16.6 文档类型支持

文档类型	预览方式	支持格式
PDF	浏览器内置PDF预览	.pdf
Office文档	LibreOffice转PDF预览	.doc/.docx/.xls/.xlsx/.ppt/.pptx
图片	直接显示	.jpg/.png/.gif/.bmp/.svg
文本	代码高亮显示	.txt/.csv/.log
视频	HTML5播放器	.mp4/.webm/.avi
压缩包	仅显示文件列表	.zip/.rar/.7z

17. 行业模板配置系统 (IndustryTemplateController)

17.1 功能概述

行业模板配置系统（Industry Template System）是QMES SaaS平台的第15个产品线。它不提供独立的业务功能页面，而是作为“配置层”贯穿所有现有产品线，实现一套代码多套皮肤的行业差异化体验。核心能力包括：行业定义、模板管理、租户设置、配置加载合并、界面文案替换、License联动。

17.2 功能列表

功能	说明	控制器
行业管理	平台运营创建/启禁用行业	IndustryController (Admin)
模板管理	对每个行业创建/版本迭代JSON模板配置	IndustryTemplateController (Admin)
租户设置	租户选择行业、继承模板、自定义覆盖	TenantIndustrySettingController (Tenant)
配置预览	查看模板+覆盖深度合并后的最终配置	TenantIndustrySettingController@preview
配置加载	IndustryTemplateMiddleware 加载行业配置	IndustryTemplateMiddleware
术语替换	industry_terms() 辅助函数替换界面文案	helpers.php

17.3 7个JSON配置维度详解

行业模板的 `config` 字段是一个JSONB结构，包含7个配置维度。每个维度控制QMES平台的一个表现层面：

17.3.1 menu（菜单配置）

控制左侧导航菜单的可见性、排序和标签文案：

```
// 数据结构
'menu' => [
    'hidden_routes' => ['production.food_trace', 'quality.fmea'], // 要隐藏的路由
    'custom_order'  => ['production' => 1, 'mold_management' => 2], // 自定义排序
    'labels'        => ['production' => '机台作业', 'mold_management' => '模具管理'], // 菜单标签
]

// 前端渲染逻辑
$hiddenRoutes = $industryConfig['menu']['hidden_routes'] ?? [];
$menuTree = array_filter($menuTree, fn($menu) => !in_array($menu['path'], $hiddenRoutes));
```

17.3.2 dict_presets（预设字典数据）

行业创建时自动初始化的字典数据，如工作中心类型、缺陷分类：

```
// 数据结构（注塑行业示例）
'dict_presets' => [
    'work_center_type' => ['注塑机', '机械手', '模温机'], // 工作中心组
    'defect_categories' => ['缩水', '毛刺', '缺料', '银纹'], // 缺陷分类
]
```

```
// 初始化流程（租户入驻时）
foreach ($config['dict_presets'] as $dictKey => $items) {
    foreach ($items as $item) {
        DictEntry::create([
            'tenant_id' => $tenantId,
            'dict_key' => $dictKey,
            'dict_value' => $item,
            'source' => 'industry_template', // 标记来源
        ]);
    }
}
```

17.3.3 terminology（界面文案映射）

替换系统各处硬编码的界面术语。通过 `industry_terms()` 函数动态读取：

```
// 数据结构
'terminology' => [
    'work_order' => '生产工单', // 替代默认的"工单"
    'process' => '机台', // 替代默认的"工序"
    'semi_finished_product' => '注塑件', // 替代默认的"半成品"
    'workstation' => '工位', // 替代默认的"工作站"
]

// 使用方式
// Blade模板中: @industry('work_order', '工单')
// PHP代码中: industry_terms('work_order', '工单')
// Vue前端: {{ industryConfig.terminology.work_order || '工单' }}
function industry_terms($key, $default = null) {
    $config = app('industry_config');
    return $config['terminology'][$key] ?? $default ?? trans("terms.$key");
}
```

17.3.4 dashboard（仪表盘卡片）

控制租户首页仪表盘显示哪些KPI卡片：

```
// 数据结构（注塑行业）
'dashboard' => [
    'cards' => [
        ['title' => '机台稼动率', 'type' => 'gauge', 'source' => 'equipment.iot'],
        ['title' => '当日产量', 'type' => 'counter', 'source' => 'production.output'],
        ['title' => '模具剩余寿命', 'type' => 'list', 'source' => 'mold.life'],
        ['title' => '不良率趋势', 'type' => 'line', 'source' => 'quality.trend'],
    ]
]

// 渲染逻辑
foreach ($config['dashboard']['cards'] as $card) {
    // 按 card.source 从对应的数据服务获取数据
}
```

```

    $data = DashboardDataService::getSource($card['source'], $tenantId);
    // 按 card.type 渲染不同的UI组件
}

```

17.3.5 feature_flags (功能开关)

控制行业特性模块的可见性。与License联动，License有最终决定权：

```

// 数据结构（注塑行业开启模具管理，关闭食品追溯）
'feature_flags' => [
    'enable_mold_management' => true,    // 注塑特有：模具管理
    'enable_batch_trace' => false,       // 注塑通常不需要批次追溯
    'enable_cleanroom_mode' => false,    // 注塑无洁净室需求
    'enable_food_traceability' => false, // 食品追溯
]

// 最终可见性 = 行业模板开关 && License模块开启
$activeFlags = array_merge(
    $industryConfig['feature_flags'],
    $tenant->license->modules ?? []
);

```

17.3.6 reports (预置报表模板)

为行业预设专用报表模板ID和默认参数：

```

// 数据结构
'reports' => [
    'daily_production' => [
        'template_id' => 'report_injection_daily',
        'params' => ['industry' => 'injection_molding']
    ],
    'quality_weekly' => [
        'template_id' => 'report_defect_rate',
        'params' => ['group_by' => 'defect_type']
    ]
]

```

17.3.7 workflow (流程预设)

定义行业专用的业务流程参数（如巡检频率、审批流程）：

```

// 数据结构
'workflow' => [
    'patrol_inspection' => [
        'enabled' => true,
        'frequency' => '每2小时', // 注塑行业2小时巡检一次
    ],
    'quality_alert' => [

```

```

        'consecutive_drop_threshold' => 3, // 连续3批下降触发预警
        'notification_channels' => ['wecom', 'inapp'],
    ]
}

```

17.4 完整示例：注塑行业模板配置

```

{
  "menu": {
    "hidden_routes": ["production.food_trace", "quality.fmea", "equipment.cleanroom"],
    "custom_order": {
      "production": 1, "mold_management": 2, "quality": 3, "equipment": 4, "wms": 5
    },
    "labels": { "production": "机台作业", "mold_management": "模具管理" }
  },
  "dict_presets": {
    "work_center_type": ["注塑机", "机械手", "模温机"],
    "defect_categories": ["缩水", "毛刺", "缺料", "银纹"]
  },
  "terminology": {
    "work_order": "生产工单", "process": "机台", "semi_finished_product": "注塑件"
  },
  "dashboard": {
    "cards": [
      { "title": "机台稼动率", "type": "gauge", "source": "equipment.iod" },
      { "title": "当日产量", "type": "counter", "source": "production.output" },
      { "title": "模具剩余寿命", "type": "list", "source": "mold.life" },
      { "title": "不良率趋势", "type": "line", "source": "quality.trend" }
    ]
  },
  "feature_flags": {
    "enable_mold_management": true, "enable_batch_trace": false,
    "enable_cleanroom_mode": false
  },
  "reports": {
    "daily_production": { "template_id": "report_injection_daily", "params": {} }
  },
  "workflow": {
    "patrol_inspection": { "enabled": true, "frequency": "每2小时" }
  }
}

```

17.5 配置加载与合并逻辑

```

// TenantIndustrySetting@getMergedConfig
public function getMergedConfig(): array
{
    $templateConfig = $this->template ? $this->template->config : [];
    $overrides = $this->overrides ?? [];
}

```

```

// 递归深度合并：模板配置 + 租户覆盖
$merged = array_merge_recursive($templateConfig, $overrides);

// 如果租户显式设置了feature_flags中的某个值为false，覆盖模板的true
foreach ($overrides['feature_flags'] ?? [] as $key => $value) {
    $merged['feature_flags'][$key] = $value;
}

return $merged;
}

// 租户覆盖示例（某注塑厂不想用模具管理）
// overrides:
{
    "menu": { "hidden_routes": ["mold_management"], "labels": { "production": "我的注塑车间" } },
    "feature_flags": { "enable_mold_management": false }
}
// 合并后：mold_management 路由隐藏，生产菜单变为"我的注塑车间"，模具功能关闭

```

17.6 与License的配合

License中增加行业特性模块声明，优先级最高（可覆盖模板配置）：

```

{
    "modules": {
        "production": true,
        "quality": true,
        "mold_management": true,
        "food_traceability": false
    }
}

```

系统最终功能可见性 = 行业模板开关 && License模块开启。即使模板开启了 `enable_mold_management: true`，如果License中没有该模块，功能仍然不可见。

17.7 租户入驻初始化流程

```

// TenantService@initialize 中新增
public function initialize(Tenant $tenant, string $industryCode): void
{
    // 1. 查找行业
    $industry = Industry::where('code', $industryCode)->firstOrFail();

    // 2. 查询默认模板
    $defaultTemplate = IndustryTemplate::where('industry_id', $industry->id)
        ->where('is_default', true)
        ->first();

    // 3. 创建租户行业设置
    TenantIndustrySetting::create([

```

```
        'tenant_id' => $tenant->id,  
        'industry_id' => $industry->id,  
        'template_id' => $defaultTemplate?->id,  
        'overrides' => [],  
    });  
  
    // 4. 初始化预设字典  
    if ($defaultTemplate) {  
        $this->initDictPresets($tenant, $defaultTemplate->config['dict_presets'] ?? []);  
    }  
  
    // 5. 写入License模块  
    if ($defaultTemplate) {  
        $licenseModules = $defaultTemplate->config['feature_flags'] ?? [];  
        $tenant->license->update(['modules' => $licenseModules]);  
    }  
}
```

文档结束