

Integrações

← Índice da documentação da API

Este documento separa o **que existe e foi comprovado no código** do que seria **possibilidade futura** (marcada como "não implementado"). Nada em "Possibilidades" funciona hoje sem desenvolvimento.

1. Integrações existentes

#	INTEGRAÇÃO	SENTIDO	PROTOCOLO	AUTENTICAÇÃO	SITUAÇÃO
1	Frontend → API REST	cliente → servidor	HTTPS/JSON	JWT (<code>Authorization: Bearer</code>)	Em uso
2	API → Frontend (tempo real)	servidor → cliente	STOMP sobre WebSocket	JWT no <code>CONNECT</code>	Em uso
3	Coletor industrial (Edge) → API	dispositivo → servidor	HTTP/JSON	<code>X-Edge-Key</code>	Implementado (3 rotas)
4	PDF de Ordem de Serviço → API (PCP)	usuário envia arquivo	<code>multipart/form-data</code>	JWT com permissão de planejamento	Implementado (prévia, não grava)
5	OCR (Tesseract)	API → programa local	linha de comando	n/a	Opcional, exige instalação no servidor
6	Swagger / OpenAPI da aplicação	consulta	HTTP	pública quando habilitada	Desligado por padrão (<code>SWAGGER_ENABLED=false</code>)

1.1 Frontend → API REST

- URL base: `EXPO_PUBLIC_API_BASE_URL` (ver arquitetura.md §5). Todas as rotas começam com `/api`.
- O frontend chama as rotas em português (`/api/areas` etc.) e em inglês (`/api/products` etc.) conforme o módulo; os dois aliases são equivalentes (endpoints.md).
- Login: ver autenticao.md. Todas as chamadas seguintes levam o token.

1.2 Tempo real (STOMP)

ITEM	VALOR
Endpoints	<code>/ws</code> (SockJS) e <code>/ws-native</code> (WebSocket puro — é o que o frontend usa)
Prefixo de destino de aplicação	<code>/app</code> configurado, mas não existe nenhum <code>@MessageMapping</code> no backend: o cliente só recebe, não envia mensagens
Tópicos de assinatura	<code>/topic/estoque</code> (geral), <code>/topic/area/{areaId}</code> , <code>/topic/entidade/{entidade}</code>
Autenticação	JWT no frame <code>CONNECT</code> (<code>Authorization: Bearer ...</code> , <code>authorization</code> , <code>X-Authorization</code> , <code>token</code>) ou <code>?token=</code> no handshake

Corpo do evento (`RealtimeEvent`, JSON):

```
{
  "tipo": "ITEMESTOQUE_ATUALIZADO",
  "entidade": "ItemEstoque",
  "entidadeId": 123,
  "areaEstoqueId": 4,
  "payload": {},
  "origemUsuario": "usuario@exemplo.com",
  "momento": "2026-10-15T12:00:00Z"
}
```

json

- `tipo` = `<ENTIDADE em maiúsculas>_<AÇÃO>`, com ação `CRIADO`, `ATUALIZADO` ou `REMOVIDO` (`EntityChangeListener`). Também publicam eventos próprios `NivelMaterialService` e `TabelaEncertoService` (listas), com tipos definidos por esses serviços.
- O módulo MES/PCP não publica eventos STOMP (nenhum `RealtimeEvent` sob `mes/`); a atualização da tela PCP é por consulta (botão "Atualizar" e consulta automática a cada 1 segundo enquanto a aba está em foco, conforme `MesScreen.tsx`).
- O evento é publicado **depois do commit** da transação.
- `origemUsuario` é "best effort" e pode ser nulo.

1.3 Edge (coletor industrial) → API

Fluxo comprovado (`MesEdgeController`, `MesService`, `MesRules`):

Coletor	API
GET /api/mes/edge/config?machineId=<TAG>[&edgeId=]	
	devolve revisão, execução ativa, ciclo ideal,
	peças por ciclo, falha mecânica, lâmpada dese
POST /api/mes/edge/events (SNAPSHOT CYCLE CYCLE_RESULT GAP)	
	idempotente por eventId; valida sequência e h
	{eventId, duplicate}
POST /api/mes/edge/heartbeat {machineId, adamReadAt, outboxDepth, quarantineDepth}	
	registra saúde do coletor
	{accepted, receivedAt}

Regras confirmadas (detalhes na ficha `modulos/edge.md`):

- Cabeçalho `X-Edge-Key` obrigatório. Chave global (`MES_EDGE_KEY` , mínimo 16 bytes); sem ela configurada, as rotas respondem `503` .
- `machineId` deve ser a **Tag** da injetora cadastrada no sistema (a tela de Injetoras diz: "A Tag deve ser a mesma configurada no serviço edge; não muda depois de criada"). Máquina sem vínculo → erro `Edge nao vinculado a maquina` .
- **Idempotência**: o mesmo `eventId` com o mesmo corpo é aceito como duplicado (`duplicate: true`); com corpo diferente, `409` . Reutilizar a `sequence` com outro `eventId` também gera conflito.
- **Sequência**: saltos exigem validação manual e **não contabilizam peças daquele ciclo**.
- **Horário**: eventos com mais de 5 minutos no futuro são rejeitados.
- **Tipos**: `CYCLE` exige execução original (`executionId`) e tempos; `schemaVersion ≥ 3` exige `namedSignals` (`BOMBA` , `MOLDE_FECHADO` , `ENCOSTE_BICO` , `INJECAO` , `ALARME`).
- **Lâmpada**: `lamp` ∈ `OFF | YELLOW | RED | BOTH` ; `outputStatus` ∈ `DISABLED | CONFIRMED | ERROR` .
- **Meta da OS**: o servidor continua recebendo eventos depois da meta (`TARGET_REACHED` no histórico), mas **não soma peças nem ciclos** aos indicadores; o bloqueio é de contabilização, **não envia comando de parada física** (ver `docs/PCP-META-E-SINALIZACAO-2026-09-15.md`).
- O firmware/serviço do coletor **não faz parte** dos dois projetos analisados.

1.4 Importação de OS por PDF

POST `/api/mes/orders/import-preview` (multipart, campo `file` , permissão de planejamento `mesAccess.plan`):

1. Extrai o texto do PDF (PDFBox). Se o PDF não tem camada de texto, tenta **OCR da primeira página** com Tesseract (idioma `por` , 300 dpi, limite de 30 s). O executável é procurado em `MES_TESSERACT_PATH` , em dois caminhos fixos do Windows e no `PATH` .
2. Reconhece rótulos fixos: `OS:` , `Data:` , `Produto:` , `Código:` , `Múltiplo:` , `Lote:` , `N° Máquina:` , `Ciclo:` e as linhas de produto/ferramenta.
3. Devolve uma **prévia** (não cria a ordem). Sem código de OS reconhecido, devolve erro.
4. O usuário confere e confirma; só então o frontend chama **POST** `/api/mes/orders` .

No frontend, a tela de Ordem de Serviço aceita vários PDFs por vez e avisa quando produto ou molde do PDF não existem no cadastro ("garanta que o produto e o molde já estejam cadastrados no sistema").

2. TUTOM — situação atual e possibilidades

O que o código afirma: a palavra "TUTOM" **não aparece** no código-fonte dos dois projetos. Ela aparece nas propostas e especificações do projeto (`02_PROPOSTA_V3_1_OPERACAO_PCP.md` , `docs/superpowers/specs/2026-09-15-pcp-operacao-design.md`), que descrevem o PDF importado como sendo do **TUTOM**, e afirmam que **não há integração direta (API) com o TUTOM**. Portanto:

ITEM	SITUAÇÃO
Importação de OS do TUTOM via PDF	Implementada (item 1.4), dependente do layout do PDF
Chamada do sistema Wester ao TUTOM (saída)	Não implementada
Chamada do TUTOM à API Wester (entrada)	Não implementada como integração dedicada; a API não tem chave de integração para sistemas externos além do Edge
Webhooks / eventos de saída para sistemas externos	Não implementado (o único canal de saída é o STOMP para o frontend)

2.1 Possibilidades (não implementadas) — com o que o código já oferece

POSSIBILIDADE	COMO SERIA, COM BASE NAS ROTAS EXISTENTES	PONTOS A DECIDIR
Cadastro automático de OS (sem PDF)	O TUTOM, ou um serviço intermediário, chama POST <code>/api/mes/orders</code> com <code>MesOrderInput</code> (<code>orderCode</code> , <code>productCode</code> , <code>productName</code> , <code>quantityPieces</code> , <code>piecesPerCycle</code> , <code>idealCycleMs</code> , <code>toolCode</code> , <code>toolSettings.lot</code> , <code>notes</code>). A rota já existe e exige permissão de planejamento	Não há chave de API para terceiros: seria preciso um usuário de serviço com perfil <code>MES_PCP</code> + <code>FULL_ACCESS</code> que faz login e renova o token a cada 8 h. Definir quem é a fonte da verdade quando a OS já existe (a tela do PCP avisa "A OS ... já existe na injetora ..."; a regra equivalente no backend não foi isolada neste levantamento)
Conferir produto/molde antes de enviar a OS	GET <code>/api/produtos/codigo/{codigoWester}</code> e GET <code>/api/molde</code> para validar códigos; é o mesmo cruzamento que o PDF faz hoje	Permissões <code>SCREEN_PRODUCTS</code> / <code>SCREEN_MOLDS</code> para o usuário de serviço
Devolver produção ao TUTOM	Consulta periódica a GET <code>/api/mes/history</code> e GET <code>/api/mes/reports</code> (permissão de leitura MES) para ler peças boas, refugo, ciclos e paradas por OS	Frequência de consulta, formato esperado pelo TUTOM, e que indicadores são confiáveis: o próprio sistema registra que OEE, MTBF e MTTR exigem validação
Notificação em tempo real	Um cliente STOMP autenticado assinando <code>/topic/entidade/...</code> poderia reagir a mudanças	Definir o controle de acesso por tópico; sem eventos específicos do MES documentados neste levantamento
Reconhecimento mais robusto do PDF	Evoluir <code>MesOrderPdfParser</code> com amostras reais do TUTOM	A própria proposta lista "variações de layout" como risco

Nenhuma dessas possibilidades foi testada. A tabela mostra apenas o que as rotas existentes permitiriam.

3. Segurança das integrações

- Credenciais de integração (`MES_EDGE_KEY` , `SECURITY_JWT_SECRET`) vêm de variáveis de ambiente; **não estão no código** e **não devem** ser copiadas para documentação ou repositório.
- O Edge e a API principal usam cadeias de segurança separadas; a chave do Edge não abre outras rotas.
- Qualquer integração de servidor para servidor não depende de CORS; um navegador de outra origem depende da lista de origens em `autenticacao.md` §8.