

Documentação Técnica do Pipeline de Análise (IA) — EcommPilot

Versão do código (git): e8c46644ccb05ce5b0ef6f3bba3ae5ca10e383a3 • Data do documento: 14/02/2026

Este documento descreve, passo a passo e em detalhes, como o EcommPilot executa a análise FULL (9 etapas) e LITE (2 etapas), quais dados entram e saem de cada agente, quais cálculos são determinísticos no backend e como as recomendações são geradas, criticadas e filtradas até o resultado final exposto para a API/UI.

Nota importante sobre “cálculos”: parte do pipeline é determinística (código) e parte é inferencial (modelo de IA seguindo regras do prompt). Quando um valor depende de interpretação da IA, este documento explica a regra e como ela deve ser aplicada, mas deixa claro que a decisão final é do agente.

Sumário

1. Objetivo e escopo
2. Visão geral do pipeline
3. Fontes de dados e estruturas de entrada
4. Pipeline FULL (9 etapas)
5. Pipeline LITE (diferenças)
6. Cálculos determinísticos do backend
7. Deduplicação, saturação e similaridade
8. Contratos de saída (JSON) por agente
9. Persistência, API e UI (como os dados aparecem)
10. Exemplos fim-a-fim (concretos)
11. Apêndice (glossário, listas e referências)

1) Objetivo e escopo

O objetivo desta documentação é permitir que qualquer pessoa (produto, engenharia, suporte ou operação) entenda, com precisão, como o EcommPilot:

- prepara dados internos (pedidos, produtos, estoque, cupons) para análise;
- coleta dados externos (Trends, preços de mercado e concorrentes), quando habilitado;
- busca benchmarks e estratégias via RAG (base de conhecimento/embeddings);
- orquestra agentes de IA (ProfileSynthesizer → Collector → Analyst → Strategist → Critic);
- executa filtros de qualidade (dedup, saturação de temas, similaridade por embeddings, diversificação);
- persiste resultados e os expõe em API/Frontend.

O pipeline existe em duas variações:

- **FULL:** 9 etapas com 4 agentes principais + ProfileSynthesizer + pós-processamento.
- **LITE:** 2 etapas (Analyst + Strategist) com dados compactos e menos sugestões.

Escopo do documento: descreve comportamento conforme o código no commit indicado na capa. Se houver alterações futuras em prompts, thresholds ou campos, esta documentação deve ser atualizada.

2) Visão geral do pipeline

2.1) Componentes principais

- **Orquestração FULL:** app/Services/AI/Agents/StoreAnalysisService.php
- **Orquestração LITE:** app/Services/AI/Agents/LiteStoreAnalysisService.php
- **Agentes (serviços):** ProfileSynthesizer, Collector, Analyst, Strategist, Critic
- **Prompts:** app/Services/AI/Prompts/*.php
- **RAG:** app/Services/AI/RAG/KnowledgeBaseService.php
- **Dados externos:** app/Services/ExternalData/*
- **Dedup/similaridade:** app/Services/Analysis/* e app/Services/AI/EmbeddingService.php
- **Persistência/API:** app/Http/Resources/AnalysisResource.php, SuggestionResource.php

2.2) Etapas FULL (progress tracking)

O progresso da análise FULL é exposto como 9 etapas (0–9), com nomes legíveis para UI:

Etapa	Nome (UI)	O que acontece
1	Identificando nicho	Define nicho/subcategoria (configurado ou detectado) + carrega benchmarks estruturados.
2	Carregando histórico	Carrega análises e sugestões anteriores, identifica temas saturados e categorias bloqueadas.
3	Buscando benchmarks	Busca benchmarks e estratégias via RAG (embeddings ou fallback textual).
4	Coletando dados externos	Google Trends, preços (Google Shopping) e scraping de concorrentes (opcional).
5	Executando Collector	Organiza contexto completo (histórico + benchmarks + mercado + concorrentes) para o Analyst.
6	Executando Analyst	Diagnóstico, alertas, oportunidades, posicionamento e Health Score (com regras explícitas).
7	Executando Strategist	Gera 18 sugestões (6/6/6) + premium_summary (Growth Intelligence Framework™).

Etapa	Nome (UI)	O que acontece
8	Executando Critic	Valida fatos, corrige números, rejeita/recria e seleciona as melhores sugestões finais.
9	Filtrando sugestões	Dedup intra-batch, similaridade por embeddings, saturação de temas, diversificação e recovery.

Etapa 4.5 (sem número no progresso): o ProfileSynthesizer roda entre as etapas 4 e 5 como uma etapa intermediária ("4.5") e não altera o contador de progresso 1–9.

3) Fontes de dados e estruturas de entrada

3.1) Dados internos da loja (backend)

O pipeline FULL prepara um pacote de dados internos com foco em operação e performance recente. Em geral, a análise trabalha com um período de **15 dias** (FULL) ou **7 dias** (LITE), incluindo o dia atual, com corte em `startOfDay()`.

3.1.1) Estatísticas globais (`store_stats`)

Antes de chamar agentes, o sistema calcula estatísticas globais da loja (independentes do período de 15/7 dias), usadas como “contexto de operação”:

- `operation_time`: tempo desde o primeiro pedido (ou desde criação da loja se sem pedidos);
- `total_orders`, `total_customers`, `total_products`;
- `active_products`: produtos com `is_active = true`;
- `recent_orders_15d`: quantidade de pedidos recentes em 15 dias;
- `total_revenue`: soma de `total` apenas de pedidos pagos (pagamento confirmado);
- `connected_at`: data (YYYY-MM-DD) de conexão/criação.

3.1.2) Resumo operacional do período (`store_data FULL`)

Para o FULL, o método de preparação de dados retorna um JSON com quatro blocos: `orders`, `products`, `inventory` e `coupons`. Estes blocos são passados ao Analyst (e parte deles aparece indiretamente em Collector/Strategist via contexto).

Regra crítica (brindes/amostras): produtos identificados como brinde/amostra são excluídos dos cálculos de estoque e de listas operacionais. Isso evita que “SKUs gratuitos” distorçam alertas de ruptura e sugestões de reposição.

3.1.3) Detecção e exclusão de brindes (`excludeGifts / isGift`)

A exclusão de brindes ocorre via escopo `excludeGifts()` aplicado nas consultas de produtos. A detecção é baseada em termos no nome do produto (ex.: *brinde*, *amostra*, *gratis*, *gift*, *travel size*, etc.). Se o nome contiver qualquer termo da lista, o produto é considerado brinde.

3.2) Dados externos de mercado (opcional)

A coleta externa é opcional e controlada por configurações (SystemSettings). Quando habilitada, o pipeline agrega:

- **Google Trends (SerpAPI):** interesse de busca (0–100), tendência (alta/estável/queda), sazonalidade;
- **Google Shopping (SerpAPI):** faixa de preços (min/max/média/mediana) e produtos referência;
- **Concorrentes (scraping):** faixa de preços, categorias foco, promoções, avaliações e diferenciais.

Mesmo quando ocorre erro em algum serviço externo, o pipeline FULL **não falha** por esse motivo: ele registra o erro e continua com `external_data` vazio ou parcial.

3.3) RAG / Base de conhecimento (benchmarks e estratégias)

Benchmarks e estratégias são buscados em uma base interna (`knowledge_embeddings`). Quando embeddings estão configurados, a busca é semântica; caso contrário, há fallback para busca textual e/ou defaults.

Dependência técnica: busca semântica com embeddings exige que o banco seja PostgreSQL e que exista armazenamento em `pgvector`. Caso contrário, o serviço cai para busca textual/fallback.

3.4) Histórico, feedback e aprendizado

O pipeline carrega até 5 análises anteriores concluídas e até 50 sugestões anteriores. A partir disso, ele constrói um `learning_context` com:

- casos de sucesso semelhantes (mesmo nicho/subcategoria);
- casos de falha/ignorado da própria loja;
- taxa de sucesso por categoria (global, agregada em tabela de stats);
- sugestões agrupadas por status (pendente, em progresso, concluída, rejeitada, ignorada);
- categorias bloqueadas (3+ rejeições) e temas saturados (3+ ocorrências; ou 2+ rejeições).

Em formato `analysis.format_version = v2`, existe um modo adicional de resumo de histórico (`history_summary`) para reduzir tokens, consolidando as últimas 100 sugestões em um resumo estruturado.

3.5) Configurações e feature flags relevantes

As principais chaves de configuração (SystemSettings) envolvidas neste pipeline incluem:

Grupo	Chaves	Efeito prático
Análise	<code>analysis.format_version</code> , <code>analysis.v2.use_history_summary</code>	Controla formato/normalização e uso de resumo do histórico para economizar tokens.
IA (chat)	<code>ai.provider</code> (+ chaves de API/model/temperature/max_tokens por provedor)	Define provedor primário (Gemini/OpenAI/Anthropic) e parâmetros padrão.
Embeddings	<code>ai.embeddings.provider</code> (+ chaves por provedor)	Habilita embeddings para RAG e filtro de similaridade por vetores.
Dados externos	<code>external_data.enabled</code> , <code>external_data.trends.enabled</code> , <code>external_data.market.enabled</code> , <code>external_data.competitors.enabled</code> , <code>external_data.serpapi_key</code>	Ativa/parametriza coleta externa de Trends/Shopping/Concorrentes.
Concorrentes (scraping)	<code>external_data.competitors.max_per_store</code> , <code>external_data.competitors.scrape_timeout</code> , <code>external_data.competitors.request_delay</code>	Limita quantidade, timeout e atraso entre scraping de concorrentes.

4) Pipeline FULL (9 etapas)

4.1) Ponto de entrada, logs e progresso

A execução FULL é orquestrada por `StoreAnalysisService`. A cada etapa, o serviço registra logs e atualiza o progresso da análise (campos `current_stage`, `total_stages` e dados auxiliares).

- **Tracking de etapa (admin/debug):** criação/atualização de `AnalysisExecutionLog` com status `running` → `completed` ou `failed`.
- **Tracking para UI:** a API retorna `progress_percentage` e `current_stage_name` (ex.: "Executando Analyst") para exibição no frontend.
- **Resiliência:** falhas em dados externos (etapa 4) e `ProfileSynthesizer` (etapa 4.5) não derrubam a análise; falhas nas etapas principais (Collector/Analyst/Strategist/Critic) interrompem o pipeline.

4.2) Etapa 1 — Identificação de nicho e subcategoria

Esta etapa define `niche` e `subcategory`, carrega metas configuradas da loja e busca benchmarks estruturados do nicho/subcategoria.

4.2.1) Caminhos possíveis

Condição	Resultado	Motivo
Loja tem <code>niche</code> e <code>niche_subcategory</code> configurados	Usa os valores configurados (não tenta detectar)	Prioriza configuração explícita do cliente
Loja tem apenas <code>niche</code> configurado	Detecta somente a subcategoria dentro do nicho	Preserva nicho configurado, detalha subcategoria
Loja não tem nicho configurado	Detecta nicho e subcategoria	Automação para lojas sem setup

4.2.2) Auto-detecção (RAG + fallback)

A detecção preferencial é via RAG (embeddings) usando categorias mais frequentes e títulos de produtos ativos. Se o RAG retornar `general` para nicho, existe fallback por keywords (nome da loja e categorias).

Importante: a auto-detecção depende da qualidade dos dados de catálogo (categorias e títulos). Lojas com categorias genéricas ("Produtos", "Novidades") tendem a cair em fallback/geral.

4.2.3) Benchmarks estruturados

Além dos textos de benchmarks (RAG), o pipeline carrega **benchmarks estruturados** (ex. ticket médio, conversão, abandono). Eles são usados para:

- comparações quantitativas (ex.: ticket da loja vs benchmark);
- cálculo guiado do Health Score no Analyst;
- ancoragem de decisões no Strategist (premium_summary e sugestões HIGH estratégicas).

4.3) Etapa 2 — Contexto histórico e aprendizado

Nesta etapa, o sistema carrega o histórico e constrói um contexto de aprendizado para reduzir repetição e aumentar eficácia. Os principais produtos desta etapa são:

- previous_analyses: últimas análises concluídas (score/status/insight);
- previous_suggestions: últimas sugestões, com status e metadados;
- saturated_themes: temas sugeridos 3+ vezes (ou rejeitados 2+);
- blocked_categories: categorias com 3+ rejeições (sinal de "cliente não quer");
- learning_context: casos de sucesso/falha + taxas de sucesso por categoria.

Exemplo simplificado do learning_context (formato ilustrativo):

```
{
  "success_cases": [
    {"category": "conversion", "title": "Reducir abandono...", "metrics_impact": {"revenue": 12.5}}
  ],
  "failure_cases": [
    {"category": "coupon", "title": "Cupom X", "failure_reason": "Margem caiu"}
  ],
  "category_success_rates": {
    "conversion": {"total_implemented": 8, "total_successful": 5, "success_rate": 62.5}
  },
  "suggestions_by_status": {
    "accepted_successful": [],
    "accepted_failed": [],
    "rejected": [],
    "in_progress": [],
    "pending": []
  },
  "blocked_categories": {"pricing": 3}
}
```

Como isso afeta recomendações: temas saturados e categorias bloqueadas alimentam “zonas proibidas” nos prompts (Strategist/Critic) e também são reforçados por filtros programáticos na etapa 9.

4.4) Etapa 3 — RAG: benchmarks e estratégias

Com nicho e subcategoria definidos, o sistema busca:

- benchmarks: textos de benchmarks do nicho/subcategoria;
- rag_strategies: estratégias e “melhores práticas” contextualizadas para o nicho.

Os resultados são passados para Collector, Analyst e Strategist. Caso não haja embeddings configurados, o serviço cai para fallback textual e/ou valores padrão (quando aplicável).

4.5) Etapa 4 — Coleta de dados externos (Trends, preços e concorrentes)

A coleta externa roda como “Etapa 4/9” no tracking, mas internamente é descrita no código como “Etapa 3.1”. Nesta etapa, o sistema:

- seleciona até 5 nomes de produtos ativos (não brindes) mais recentemente atualizados;
- monta palavras-chave (nicho, subcategoria, top produtos) para Trends;
- consulta Trends e Shopping via SerpAPI;
- faz scraping de concorrentes informados na loja (se existir lista de concorrentes).

Comportamento em falha: se um serviço externo falhar (timeout, HTTP inválido, sem chave de API), a etapa é marcada como falha no log, mas o pipeline continua com dados externos vazios/parciais.

4.6) Etapa 4.5 — ProfileSynthesizer (perfil sintetizado)

O ProfileSynthesizer gera um store_profile e um contexto_analise concisos, que servem como “perfil compartilhado” para os agentes seguintes. Ele roda com baixa temperatura (0.1) e regras estritas anti-alucinação (usar nao_determinado quando não houver dado).

- **Inputs principais:** nome, plataforma, nicho/subcategoria, URL, store_stats e benchmarks.
- **Outputs principais:** porte estimado, maturidade digital, público-alvo estimado, diferenciais mensuráveis e sazonalidade.
- **Falha:** se não extrair JSON, retorna perfil padrão (porte/maturidade “nao_determinado”).

4.7) Etapa 5 — Collector Agent (organização do contexto)

O Collector é responsável por **organizar** (não “diagnosticar”) todo o contexto necessário para o Analyst: histórico, benchmarks, dados externos e “zonas proibidas” (temas saturados/categorias bloqueadas).

- **Temperatura:** 0.1 (prioriza consistência e estrutura).
- **Saída:** JSON com identificação da loja, resumo histórico (5–7 fatos com números), posicionamento de mercado (comparação tripla) e análise competitiva (quando disponível).
- **Alertas para o Analyst:** o Collector já antecipa sinais (critical/warnings/info), mas não substitui o diagnóstico.
- **Data quality:** registra dados ausentes e nível de completude para orientar confiança dos agentes seguintes.

Por que existe o Collector: ele reduz carga cognitiva/tokens do Analyst ao entregar uma visão organizada e rastreável do contexto (incluindo o que não deve ser repetido).

4.8) Etapa 6 — Analyst Agent (diagnóstico + Health Score + briefing)

O Analyst produz o diagnóstico estruturado. Ele recebe dados operacionais do período, histórico da loja, dados externos e benchmarks. O output do Analyst alimenta diretamente:

- o resumo exibido no frontend (`health_score`, `health_status`, `main_insight`);
- os alertas (`alerts`) e oportunidades (`opportunities`);
- o briefing para o Strategist (5 causas raiz) e o comparativo competitivo.

Regras centrais (impostas no prompt do Analyst):

- **Health Score (0–100):** calculado por 5 componentes com pesos e regras de override graduais;
- **Alertas:** máximo 5, sempre com evidência numérica; confiança baixa → severidade “monitorar”;
- **Oportunidades:** exatamente 5, cada uma com potencial específico em R\$ (ou faixa se baixa confiança);
- **Posicionamento:** comparação tripla (benchmark vs mercado vs concorrentes);
- **Briefing strategist:** 5 problemas como *causas raiz* (não sintomas), diferentes entre si.

Normalização do output: o backend suporta formatos legados e converte alertas para um formato compatível com a UI. Se o JSON vier inválido, o Analyst retorna uma estrutura padrão para evitar quebra do pipeline.

4.9) Etapa 7 — Strategist Agent (18 sugestões + premium_summary)

O Strategist transforma o diagnóstico em um plano de ação: ele gera **18 sugestões** (6 HIGH + 6 MEDIUM + 6 LOW) e também um **premium_summary** completo seguindo o Growth Intelligence Framework™.

- **Temperatura:** rotativa por contagem de análises ($0.5 \rightarrow 0.7$) para balancear criatividade sem perder aderência.
- **HIGH estratégicas:** devem usar categorias estratégicas (strategy/investment/market/growth/financial/positioning) e referenciar dados externos/benchmarks quando disponíveis.
- **MEDIUM/LOW táticas:** ações operacionais, viáveis e específicas (estoque, pricing, produto, conversão, marketing, cupom, operacional).
- **Validação programática:** sugestões HIGH sem dados específicos (R\$, %, contagens) podem ser rebaixadas para MEDIUM.

4.10) Etapa 8 — Critic Agent (verificação, correção e seleção final)

O Critic revisa a saída do Strategist com foco em qualidade e veracidade: valida números, originalidade, viabilidade e alinhamento ao diagnóstico. O Critic executa 6 verificações (V1–V6) por sugestão e documenta o resultado.

- **Entrada:** sugestões do Strategist + dados originais + dados detalhados de loja (para validar cálculos).
- **Saída:** 9 sugestões selecionadas (3 HIGH, 3 MEDIUM, 3 LOW), com versão final e score de qualidade.
- **Substituição:** sugestões rejeitadas devem ser substituídas por alternativas novas e verificáveis.

4.11) Etapa 9 — Filtros finais (similaridade, saturação, recovery e diversificação)

Após o Critic, o backend aplica filtros determinísticos para garantir:

- mínimo de repetição (intra-batch e histórica);
- bloqueio de temas saturados (3+ ocorrências ou rejeição recorrente);
- controle de similaridade por embeddings (quando configurado);

- distribuição final 3-3-3 e diversidade por categoria (máximo 2 por categoria).

Regra de negócio: o objetivo do pipeline FULL é entregar **exatamente 9 sugestões** finais, na distribuição **3 HIGH / 3 MEDIUM / 3 LOW**. Quando filtros removem sugestões, o sistema tenta recuperar do pool do Strategist (já pré-validado contra histórico) e, se necessário, promove níveis adjacentes.

A etapa 9 é detalhada em profundidade na seção 7) Deduplicação, saturação e similaridade, incluindo thresholds e exemplos de como uma sugestão é descartada/recuperada.

5) Pipeline LITE (diferenças)

O pipeline LITE existe para cenários em que há necessidade de reduzir custo/tokens e acelerar execução. Ele troca profundidade por velocidade e reduz o número de sugestões finais.

5.1) Principais diferenças vs FULL

Dimensão	FULL	LITE
Etapas	9 etapas (com dados externos, Collector, Analyst, Strategist, Critic e filtros finais)	2 etapas (Lite Analyst + Lite Strategist) + validação básica + dedup histórico
Período de dados operacionais	15 dias	7 dias
Quantidade de sugestões	9 finais (3 HIGH, 3 MEDIUM, 3 LOW), derivadas de pool maior (18 do Strategist)	6 finais (2 HIGH, 2 MEDIUM, 2 LOW)
Critic	Sim (validação factual e seleção final)	Não (validação básica inline no código)
Dados externos e RAG	Usados (quando habilitados) e influenciam HIGH/premium_summary	Em geral, não há coleta externa no fluxo LITE padrão (foco em dados compactos)

5.2) Fluxo LITE passo a passo

- Resolver módulo especializado:** usa AnalysisRouter para obter foco extra (se aplicável).
- Identificar nicho:** executa detecção (mais simples) para contextualizar prompts.
- Preparar dados compactos:** período de 7 dias, top 20 best sellers, resumo de cupons (sem lista detalhada).
- Lite Analyst:** retorna metrics, até 3 anomalies e overall_health.
- Lite Strategist:** retorna 6 sugestões com distribuição 2-2-2, ações implementáveis em até 1 semana.
- Validação básica:** exige campos mínimos e normaliza expected_impact para high/medium/low.
- Dedup histórico:** aplica validateSuggestionUniqueness contra sugestões anteriores.
- Persistência:** salva análise e sugestões (sem Critic).

Trade-off: sem Critic, o LITE tem menor “garantia de qualidade” em verificação numérica e originalidade. Em compensação, ele executa mais rápido e consome menos tokens.

6) Cálculos determinísticos do backend

Esta seção descreve **apenas** os cálculos feitos pelo código (determinísticos). Ela é essencial porque vários agentes (principalmente Analyst/Critic) dependem desses números como "fonte de verdade" para evitar alucinação, recalcular impactos e validar consistência.

6.1) Janela temporal e definições

- **FULL:** período operacional = 15 dias (ANALYSIS_PERIOD_DAYS = 15), incluindo o dia atual.
- **LITE:** período operacional = 7 dias (ANALYSIS_PERIOD_DAYS = 7 no serviço LITE).
- **Pedidos pagos:** várias métricas de receita/ticket usam *apenas pedidos pagos*.
- **Brindes:** produtos brinde são excluídos de consultas e métricas de estoque.

6.2) Pedidos (orders)

Resumo calculado para o período do pipeline:

Campo	Como é calculado	Observações
total	Contagem de pedidos no período	Inclui todos os status; receita é calculada separadamente com pagos
by_payment_status	Agrupamento por status de pagamento → contagem	Chave = valor do enum (ou unknown)
total_revenue	Soma de total apenas dos pedidos pagos	Não inclui pedidos pendentes/cancelados
average_order_value	Média de total dos pedidos pagos	Se não houver pagos, fica 0
by_day	Agrupamento por data (Y-m-d) → contagem	Útil para detectar sazonalidade/volatilidade
cancellation_rate	(Pedidos cancelados ÷ total do período) × 100	Se total = 0, fica 0

6.3) Produtos (products)

Resumo do catálogo (excluindo brindes) + derivações importantes:

Campo	Como é calculado	Observações
total	Contagem de produtos após excludeGifts()	O total bruto é usado para calcular gifts_filtered
active	Produtos com is_active = true e não brinde	Filtro duplo: status ativo + checagem !isGift()
out_of_stock	Contagem de produtos com stock_quantity <= 0	Base para alertas de ruptura e override do Health Score
out_of_stock_list	Lista (limitada) com detalhes dos esgotados	Inclui id, name, price, sku, last_updated
best_sellers	Top 10 por quantidade vendida em pedidos pagos (agregado por product_id dos itens)	Inclui quantidade, receita, preço e estoque atual
no_sales_period	Contagem de produtos ativos com estoque > 0 e sem vendas no período	Útil para "estoque parado" e ações de giro
gifts_filtered	Total bruto – total após excludeGifts()	Indica quantos itens foram removidos por serem brindes

6.4) Estoque (inventory)

Resumo de inventário (produtos excluindo brindes):

Campo	Como é calculado	Observações
total_value	$\Sigma(\text{stock_quantity} \times \text{cost})$	Depende de custo preenchido; custo ausente reduz a precisão
low_stock_products	Contagem de produtos com stock_quantity < 10	O threshold padrão do modelo é 10
excess_stock_products	Contagem de produtos com stock_quantity > 100	Heurística simples para "estoque alto/excesso"

6.5) Cupons (coupons)

O pipeline calcula tanto a situação de cadastro (quantos cupons existem/ativos) quanto o uso real em pedidos pagos no período.

Campo	Como é calculado	Interpretação
usage_rate_percent	(Pedidos pagos com cupom ÷ total de pedidos pagos) × 100	Alta dependência sugere risco de margem e preço âncora "desconto"
ticket_impact_percent	((Ticket médio com cupom – ticket médio sem cupom) ÷ ticket	Negativo = cupom reduz ticket; positivo = cupom aumenta ticket

Campo	Como é calculado	Interpretação
	médio sem cupom) × 100	(ex.: cupom condicionado)
total_discount_given	Soma de discount dos pedidos pagos com cupom	Estimativa de “custo do desconto” no período
most_used_coupons	Top 5 por times_used (agregado por código)	Ajuda a identificar cupons “viciantes” e oportunidades de reestruturação

6.6) Cálculos de dados externos

6.6.1) Google Trends (SerpAPI)

- `interesse_busca`: média dos pontos (arredondada) do interesse no tempo (0–100).
- `tendencia`: compara a média dos **4 primeiros** pontos com a média dos **4 últimos**: variação > +15% → alta; variação < -15% → queda; caso contrário → estavel.
- `sazonalidade`: meses cujo interesse médio fica > 20% acima da média geral.
- `termos_relacionados`: até 5 queries “rising”.

6.6.2) Preços de mercado (Google Shopping via SerpAPI)

- **Query**: se houver top produtos, usa até 3 nomes; caso contrário, usa rótulos de nicho + subcategoria.
- **Extração de preço**: prioriza `extracted_price`; fallback para `parse` de string `price`.
- **Outliers**: remove preços fora de $1.5 \times \text{IQR}$ (método do intervalo interquartil).
- **Estatísticas**: calcula min, max, média e mediana; mantém até 10 produtos referência.

6.6.3) Concorrentes (scraping + parsing)

A análise de concorrentes depende de uma lista de concorrentes informada no cadastro da loja. Para cada URL, o sistema tenta coletar conteúdo (via proxy ou request direto) e extrai sinais estruturados:

- preços (regex e padrões comuns, em HTML/Markdown/JSON embutido);
- produtos (nome + preço, limitado a 50);
- avaliações (nota média, quantidade quando possível);
- categorias e promoções; diferenciais (frete grátis, cashback, parcelamento, etc.).

6.7) Benchmarks estruturados e RAG (detecção e busca)

O serviço de base de conhecimento faz três tarefas distintas:

- **Buscar benchmarks/estratégias (texto):** para inclusão em prompts;
- **Fornecer benchmarks estruturados:** números (ticket médio, conversão, abandono, etc.) com fallback;
- **Identificar nicho/subcategoria:** via embeddings + config/keywords como reforço.

Onde a IA usa esses números: o Analyst usa benchmarks estruturados para pontuar componentes do Health Score e justificar gaps; o Strategist usa para “ancorar” expected_result e projeções do premium_summary.

7) Deduplicação, saturação e similaridade

Esta seção explica, detalhadamente, as camadas de proteção contra repetição e baixa qualidade. Elas existem porque um agente pode gerar ideias boas porém repetidas (por linguagem diferente) e porque cada loja acumula histórico.

7.1) Conceitos

- **Duplicata interna (intra-batch):** duas sugestões muito similares dentro da mesma execução.
- **Duplicata histórica:** sugestão muito similar a uma sugestão já entregue para a mesma loja.
- **Tema saturado:** um tema apareceu 3+ vezes (ou foi rejeitado repetidamente), devendo ser evitado.
- **Similaridade por embeddings:** comparação semântica via vetores (cosine) no banco (pgvector).

7.2) Normalização de títulos (para comparação semântica)

Antes de calcular similaridade, os títulos são normalizados para reduzir ruído linguístico:

- converte para minúsculo;
- remove acentos (quando possível) e caracteres não alfanuméricos;
- remove stopwords (de/da/do/para/com/...);
- remove verbos genéricos (implementar, otimizar, criar, ativar, melhorar, etc.);
- remove palavras genéricas de domínio (produto, cliente, loja, etc.);
- remove palavras muito curtas (≤ 2);
- ordena alfabeticamente as palavras restantes (comparação independe da ordem).

Exemplo (ilustrativo):

Título original:
"Implementar cupom de primeira compra 10% para captar e-mails"

Após normalização (exemplo):
"captar emails primeira compra 10"

7.3) Similaridade Jaccard (word-based)

A similaridade entre dois títulos normalizados é calculada por Jaccard:

```
similaridade = |interseção(palavras)| ÷ |união(palavras)|
```

Onde “palavras” são os tokens únicos do título normalizado.

7.4) Deduplicação intra-batch (mesma execução)

Antes de comparar com histórico, o pipeline remove duplicatas dentro do próprio lote:

- **Duplicata exata:** título igual (case-insensitive, trim) → descartada.
- **Duplicata semântica:** Jaccard ≥ 0.85 entre títulos normalizados → descartada.

7.5) Temas saturados (ThemeKeywords)

O sistema identifica temas saturados varrendo título + descrição de sugestões anteriores em busca de palavras-chave canônicas por tema (ex.: *quiz*, *frete grátis*, *fidelidade*, *kits*, *reviews*, etc.).

- **Saturação por repetição:** tema com 3+ ocorrências → considerado saturado.
- **Saturação por rejeição:** tema rejeitado/ignorado 2+ vezes também é marcado como saturado para o prompt.

Importante (enforcement programático): o filtro programático de “tema saturado” bloqueia temas com contagem ≥ 3 . Temas “saturados por rejeição” (2+ rejeições) são sinalizados no contexto/prompt e devem ser evitados pelos agentes, mas podem não ser bloqueados programaticamente se a contagem total ainda for < 3 .

7.6) Categorias bloqueadas por rejeição

Se uma categoria tiver 3+ sugestões com status `rejected` ou `ignored`, ela é considerada “bloqueada” para reduzir atrito com o lojista. Esse dado entra no `learning_context` e influencia prompts (principalmente Critic/Strategist).

7.7) Similaridade por embeddings (pgvector)

Quando embeddings estão configurados, o pipeline gera um vetor para o texto (título + descrição) e consulta o banco para encontrar sugestões similares da mesma loja.

- **Métrica:** cosine distance via operador `<=` (pgvector).
- **Conversão:** similaridade = $1 - \text{distance}$.

- **Threshold:** similaridade > **0.85** → “muito similar” (filtra).
- **Janela:** considera apenas sugestões dos últimos **90 dias** (quando filtrado).
- **Status excluídos:** normalmente exclui rejected, ignored e completed (não compara com esses).

Por que embeddings: Jaccard captura similaridade de palavras; embeddings capturam similaridade de significado mesmo quando o texto muda (paráfrase).

7.8) Pré-validação do pool do Strategist (para recovery)

Após o Strategist gerar 15–20 sugestões (alvo 18), o pipeline pré-valida esse pool contra histórico:

- se Jaccard ≥ 0.85 vs histórico → remove do pool de recovery;
- se passar Jaccard, pode haver checagem por embedding (threshold 0.85) para remover duplicatas semânticas.

Isso garante que, se o Critic/embeddings removerem sugestões, o recovery não reintroduz “mais do mesmo”.

7.9) Recovery para garantir 9 sugestões (3-3-3)

Se, após Critic e filtros, o conjunto final ficar abaixo de 9, o pipeline tenta preencher lacunas seguindo ordem:

1. recuperar sugestões do Strategist no mesmo nível de impacto (high/medium/low);
2. se faltar, promover sugestões de níveis adjacentes (ex.: medium → high) mantendo diversidade;
3. executar checagem de embedding antes de aceitar candidatos recuperados/promovidos.

Além disso, o recovery filtra sugestões similares a títulos previamente rejeitados usando um threshold mais baixo (Jaccard ≥ 0.75), assumindo que “se parece com rejeitada” tende a gerar frustração.

7.10) Diversificação por categoria (máximo 2 por categoria)

Para evitar que o resultado final traga, por exemplo, “6 sugestões de cupom”, o pipeline aplica um limite de **2 sugestões por categoria**. Se alguma categoria exceder o limite:

- mantém as 2 primeiras e remove excedentes;
- tenta substituir usando o pool do Strategist com categorias diferentes;

- descarta candidatos cujo título for muito similar aos já usados (heurística `similar_text` $\geq 75\%$).

7.11) Guardas finais ao salvar

Antes de persistir, o pipeline aplica uma última proteção contra duplicatas na mesma análise:

- bloqueia títulos duplicados (case-insensitive) dentro da análise;
- renumera prioridades após qualquer remoção;
- anexa metadados do Critic/Strategist em `specific_data` (rastreabilidade).

8) Contratos de saída (JSON) por agente

Nesta seção, cada agente tem seu contrato de saída documentado com:

- **Objetivo:** o que o agente deve produzir.
- **Formato (schema):** o JSON esperado (conforme prompt).
- **Significado dos campos:** o que cada campo representa e como é usado.

Chave interna “_prompt_used”: os serviços dos agentes adicionam _prompt_used ao resultado para auditoria/logs. Essa chave é removida antes de alimentar etapas seguintes e não deve ser tratada como parte do contrato público.

8.1) ProfileSynthesizer (perfil compartilhado)

Gera um perfil sintetizado e factual da loja. É usado para aumentar coerência entre agentes e reduzir “chute”. Campos sem evidência devem ser preenchidos com nao_determinado.

8.1.1) Schema (JSON)

```
{
  "store_profile": {
    "nome": "nome da loja",
    "url": "url da loja",
    "plataforma": "nuvemshop|shopify|vtex|tray|outro",
    "nicho": "nicho identificado",
    "nicho_detalhado": "sub-nicho se identificável",
    "porte_estimado": "micro|pequeno|medio|grande",
    "maturidade_digital": "iniciante|intermediario|avancado",
    "publico_alvo_estimado": "descrição do público",
    "diferenciais_visiveis": [
      "diferencial com dado mensurável (ex: '168 kits no catálogo', 'frete grátis acima de R$99')"
    ],
    "sazonalidade_relevante": "descreva eventos sazonais do nicho"
  },
  "contexto_analise": {
    "data_analise": "data atual",
    "eventos_sazonais_proximos": ["lista de datas comerciais próximas"],
    "observacoes_iniciais": "primeiras impressões sobre a loja"
  }
}
```

8.1.2) Campos e uso

Campo	Descrição	Como é usado no pipeline
porte_estimado	Classificação por faturamento mensal (micro/pequeno/médio/grande).	Contextualiza recomendações (ex.: soluções de baixo custo para micro).
maturidade_digital	Estimativa de nível operacional/digital (iniciante/intermediário/avançado).	Modula complexidade/ambição das sugestões.
diferenciais_visiveis	Lista de diferenciais observáveis com números/condições.	Evita sugestões redundantes ("ativar frete grátis") quando já existe.
eventos_sazonais_proximos	Datas comerciais próximas relevantes ao nicho.	Ajuda a ajustar urgência/tempo das sugestões (30/60/90 dias).

8.2) Collector Agent (contexto organizado para diagnóstico)

Organiza dados e histórico para o Analyst. Seu papel é estruturar contexto e registrar o que está faltando (data quality) e o que deve ser evitado (prohibited suggestions/temas saturados).

8.2.1) Schema (JSON)

```
{
  "store_identification": {
    "name": "string",
    "niche": "string",
    "subcategory": "string",
    "platform": "string",
    "operation_time_months": 0,
    "total_orders": 0,
    "total_revenue": 0
  },
  "historical_summary": ["fato1 com número", "fato2 com número", "fato3", "fato4", "fato5", "fato6 (opcional)", "fato7 (opcional)"],
  "success_patterns": [
    {"title": "título", "category": "categoria", "what_worked": "o que funcionou"}
  ],
  "suggestions_to_avoid": [
    {"title": "título", "category": "categoria", "why_failed": "motivo"}
  ],
  "prohibitedSuggestions": {
    "total": 0,
    "saturated_themes": [],
    "by_category": {},
    "all_titles": []
  }
},
```

```

"relevant_benchmarks": {},
"market_positioning": {
    "ticket_loja": 0,
    "vs_benchmark": {"valor": 0, "diferenca": "+X% ou -X%"},
    "vs_mercado": {"valor": 0, "diferenca": "+X% ou -X%"},
    "vs_concorrentes": {"valor": 0, "diferenca": "+X% ou -X%"}
},
"competitive_analysis": {
    "total_concorrentes": 0,
    "por_concorrente": [],
    "insights": {
        "categorias_populares": [],
        "maior_desconto": "X%",
        "faixa_preco": {"min": 0, "max": 0, "media": 0}
    },
    "diferenciais_que_loja_nao_tem": [],
    "oportunidades": []
},
"identified_gaps": [],
"data_not_available": [],
"market_context": {"tendencia": "string", "interesse": 0},
"alerts_for_analyst": {
    "critical": [{"descricao": "string", "dados": "evidência numérica", "recorrente": false, "vezes_reportado": 0}],
    "warnings": [{"descricao": "string", "dados": "evidência numérica", "recorrente": false, "vezes_reportado": 0}],
    "info": [{"descricao": "string", "dados": "evidência numérica", "recorrente": false, "vezes_reportado": 0}]
},
"data_quality": {
    "completeness": "alta|media|baixa",
    "missing_data": ["lista de dados que não foram encontrados"],
    "confidence_notes": "Observações sobre a confiabilidade dos dados"
}
}
}

```

8.2.2) Campos e uso

Campo	O que significa	Por que é importante
historical_summary	Lista de fatos do histórico com números.	Garante que o diagnóstico do Analyst comece de dados verificáveis, não de generalidades.
prohibited_suggestions	Mapa do que já foi sugerido e o que saturou.	Evita repetição e instrui Strategist/Critic sobre “zonas proibidas”.
market_positioning	Comparação tripla do ticket: benchmark, mercado e concorrentes.	Base para decisões de preço, margem e proposta de valor.

Campo	O que significa	Por que é importante
alerts_for_analyst	Sinais antecipados (não é o diagnóstico final).	Ajuda o Analyst a priorizar, especialmente em pipelines com pouco tempo.
data_quality	Qualidade/completude e dados ausentes.	Controla confiança e reduz risco de alucinação ("dados insuficientes").

8.3) Analyst Agent (diagnóstico, alertas e Health Score)

O Analyst é responsável pelo diagnóstico numérico da loja. Ele produz: um **Health Score (0–100)**, alertas priorizados, 5 oportunidades com potencial estimado em R\$ e um briefing para o Strategist contendo 5 problemas (causas raiz).

Importante sobre “mês”: na execução FULL, o pipeline usa uma janela operacional fixa de **15 dias**. Mesmo assim, alguns prompts exibem “Pedidos/Mês” e “Faturamento/mês” como proxy dessa janela. Interprete qualquer “/mês” nos resultados considerando a janela do pipeline (FULL=15 dias, LITE=7 dias), a menos que o agente declare explicitamente uma premissa de extração.

8.3.1) Schema (JSON — conforme prompt)

```
{
  "reasoning": {
    "data_quality": "Avaliação da qualidade e completude dos dados recebidos",
    "key_metrics": ["métrica 1: valor (bom/ruim/neutro)", "métrica 2: valor"],
    "anomalies_detected": ["anomalia 1 com threshold", "anomalia 2"],
    "score_calculation": "Componente1: X pts + Componente2: Y pts + ... = Total. Override: -Z pts. Final: W"
  },
  "resumo_executivo": "2-3 frases: saúde geral, problema principal, oportunidade principal",
  "health_score": {
    "score_calculado": 0,
    "componentes": {
      "ticket_vs_benchmark": {"pontos": 0, "detalhe": "X% do benchmark"},
      "estoque_disponivel": {"pontos": 0, "detalhe": "X% zerado"},
      "taxa_cancelamento": {"pontos": 0, "detalhe": "X%"},
      "saude_cupons": {"pontos": 0, "detalhe": "X% uso, Y% impacto"},
      "tendencia_vendas": {"pontos": 0, "detalhe": "crescendo|estável|queda"}
    },
    "override_aplicado": false,
    "motivo_override": null,
    "score_final": 0,
    "classificacao": "critico|atencao|saudavel|excelente"
  },
  "alertas": [
}
```

```
{
  "severidade": "critico|atencao|monitorar",
  "tipo": "estoque|cancelamento|pricing|cupons|vendas",
  "titulo": "Descrição curta do problema",
  "dados": "Números específicos que comprovam",
  "impacto": "R$ X/mês ou X% de perda",
  "causa_raiz": "Análise de POR QUE isso está acontecendo, não apenas O QUE",
  "acao": "O que fazer",
  "fonte": " dado_direto|inferencia|benchmark_geral",
  "confianca": "alta|media|baixa"
},
],

"oportunidades": [
{
  "tipo": "reativacao|upsell|estoque|pricing|conversao",
  "titulo": "Descrição da oportunidade",
  "dados": "Números que embasam",
  "potencial": "R$ X/mês",
  "acao": "Como capturar",
  "fonte": " dado_direto|inferencia|benchmark_geral",
  "confianca": "alta|media|baixa",
  "ja_sugerido_anteriores": false
},
],
"posicionamento": {
  "ticket_loja": 0,
  "vs_benchmark": {"valor": 0, "diferenca": "+X% ou -X%"},
  "vs_mercado": {"valor": 0, "diferenca": "+X% ou -X%"},
  "vs_concorrentes": {"valor": 0, "diferenca": "+X% ou -X%"},
  "interpretacao": "Loja está acima/abaixo/dentro do mercado porque..."
},
"anomalias": [
{
  "metrica": "nome",
  "atual": 0,
  "historico": 0,
  "variacao": "+X% ou -X%",
  "tipo": "positiva|negativa",
  "explicacao_sazonal": "É ou não explicado pela sazonalidade"
}
],
"comparativo_concorrentes": {
  "ticket_medio": {
    "loja": 0,
    "concorrentes_media": 0,
    "gap": "+X% ou -X%",
    "insight": "Loja está acima/abaixo porque..."
  },
  "categorias": {
    "loja_foco": ["categoria1", "categoria2"],
    "concorrentes_foco": ["categoria1", "categoria2"],
    "oportunidade": "Categoria que concorrentes têm e loja não"
  }
},
```

```

"promocoes": {
    "loja_tipos": ["tipo1", "tipo2"],
    "concorrentes_tipos": ["tipo1", "tipo2", "tipo3"],
    "gap": "Concorrentes usam X tipos de promoção que loja não usa"
},
"avaliacoes": {
    "melhor_concorrente": "Nome (nota/5)",
    "insight": "Se loja deve investir em programa de reviews"
}
},
"briefing_strategist": {
    "problema_1": "Causa raiz #1 – detalhada, com dados, diferente dos problemas anteriores",
    "problema_2": "Causa raiz #2 – diferente do problema_1",
    "problema_3": "Causa raiz #3 – diferente dos dois anteriores",
    "problema_4": "Causa raiz #4 – diferente dos três anteriores",
    "problema_5": "Causa raiz #5 – diferente dos quatro anteriores",
    "oportunidade_principal": "Maior oportunidade não explorada",
    "restricoes": ["O que NÃO fazer ou limitações da loja"],
    "temas_ja_saturados": ["temas que já foram sugeridos 3+ vezes em análises anteriores"],
    "dados_chave": {
        "faturamento_mes": 0,
        "ticket_medio": 0,
        "taxa_conversao": 0,
        "estoque_zerado_percent": 0,
        "uso_cupons_percent": 0
    }
},
"dados_insuficientes": ["áreas onde não há dados suficientes para análise completa"]
}

```

8.3.2) Campos e uso (visão completa)

Campo	O que representa	O que o sistema faz com isso
health_score	Score detalhado (componentes + override + classificação).	O backend deriva um resumo em overall_health e persiste o score/status na summary.
alertas	Lista de alertas com evidência numérica e causa raiz.	Normaliza para estrutura por severidade e persiste no máximo 3 alertas para UI.
oportunidades	Lista de oportunidades com potencial (R\$) e ação.	Persistidas como opportunities e usadas para orientar HIGH/MEDIUM do Strategist.
briefing_strategist	5 problemas (causas raiz) + restrições e dados-chave.	Vira alertas_para_strategist normalizado e alimenta a geração de sugestões.
posicionamento / comparativo_concorrentes	Comparações ticket/mercado/concorrentes e gaps competitivos.	Ajuda a justificar pricing/posicionamento e dar base para sugestões estratégicas.

Campo	O que representa	O que o sistema faz com isso
dados_insuficientes	Lista explícita do que faltou para concluir análise com confiança.	Reduz risco de alucinação; pode rebaixar confiança e levar a recomendações "de dados" primeiro.

8.3.3) Cálculo do Health Score (passo a passo)

O prompt define pesos e faixas de pontuação para 5 componentes. Depois, aplica penalizações graduais ("override") para cenários críticos (estoque, cancelamento, queda forte, dependência de cupons).

8.3.3.1) Componentes (pontos)

Componente	Peso	Regra de pontuação (conforme prompt)	Dados típicos de entrada
Ticket vs Benchmark	25	$\geq 100\% = 25$; $80\%-99\% = 20$; $60\%-79\% = 15$; $<60\% = 10$	orders.average_order_value e structured_benchmarks.ticket_medio.media
Estoque disponível	25	$\leq 10\%$ zerado = 25; $11\%-20\% = 20$; $21\%-35\% = 15$; $>35\% = 10$	products.out_of_stock ÷ products.total
Taxa de cancelamento	15	$\leq 3\% = 15$; $4\%-7\% = 12$; $8\%-12\% = 8$; $>12\% = 4$	orders.cancellation_rate (%)
Saúde de cupons	15	Uso $<50\%$ e impacto $<15\% = 15$; senão proporcional	coupons.usage_rate_percent e coupons.ticket_impact_percent
Tendência de vendas	20	crescendo = 20; estável = 15; queda leve = 10; queda forte = 5	orders.by_day + historical_metrics (quando disponível)

8.3.3.2) Overrides (penalizações graduais)

Após calcular o score, o Analyst aplica penalizações acumuladas. A regra é: aplicar todas as penalizações, mas respeitar pisos/máximos quando definidos no prompt.

- **Estoque zerado:** 20–30% (-10); 30–40% (-20, mínimo 30); 40–50% (-30, mínimo 20); >50% (máximo 15).
- **Cancelamento:** 8–12% (-10); 12–18% (-20, mínimo 25); >18% (máximo 15).

- **Queda de vendas vs histórico:** 20–30% (-5); 30–45% (-15, mínimo 25); >45% (máximo 20).
- **Dependência de cupons:** 60–75% (-5); 75–90% (-15, mínimo 30); >90% (-25, mínimo 20).

8.3.3.3) Exemplo numérico (completo)

```

Ticket vs benchmark (75%) = 15
Estoque (35% zerado)     = 15
Cancelamento (10%)       = 8
Cupons (uso 72%; -12,5%) = 8 (exemplo de proporcionalidade coerente)
Tendência (estável)      = 15
-----
Score calculado          = 61

Overrides:
Estoque 35% → -20 (mínimo 30)
Cancel. 10% → -10
Cupons 72% → -5
61 - 20 - 10 - 5 = 26 → aplicar mínimo 30 → score_final = 30

```

8.3.4) Normalização no backend (AnalystAgentService)

Para manter compatibilidade com versões anteriores, o backend normaliza campos e formatos. Em especial, ele:

- deriva overall_health a partir de health_score.score_final e resumo_executivo;
- converte alertas “flat” (V5) em estrutura por severidade (V4: criticos, atencao, monitoramento).

8.3.4.1) Estrutura normalizada de alertas (V4)

```
{
  "alertas": {
    "criticos": [{"tipo": "", "titulo": "", "descricao": "", "impacto_estimado": "", "acao": "",
      "fonte": "", "confianca": ""}],
    "atencao": [{"tipo": "", "titulo": "", "descricao": "", "impacto_estimado": "", "acao": "",
      "fonte": "", "confianca": ""}],
    "monitoramento": [{"tipo": "", "titulo": "", "descricao": "", "impacto_estimado": "", "acao": "",
      "fonte": "", "confianca": ""}]
  }
}
```

8.4) Strategist Agent (18 sugestões + Premium Summary)

O Strategist é o “motor de recomendações”. Ele recebe o diagnóstico do Analyst (incluindo problemas prioritários, alertas e oportunidades), além de contexto de mercado e execução

(benchmarks, concorrentes, recursos da plataforma, sazonalidade e histórico). A partir disso, ele deve gerar:

- **18 sugestões** (alvo): 6 HIGH + 6 MEDIUM + 6 LOW;
- **premium_summary**: análise executiva estruturada no Growth Intelligence Framework™.

Regra crítica do prompt: HIGH devem ser estratégicas (categorias como strategy/market/growth/financial/positioning) e referenciar dados externos (concorrentes/mercado/benchmarks). Se HIGH for tática e/ou sem evidência, o Critic deve rebaixar/substituir.

8.4.1) Schema (JSON — conforme prompt)

```
{
  "reasoning": {
    "strategic_diagnostic": "Onde a loja está vs. onde deveria estar. Ex: 'Fatura R$ 45k/mês, mercado suporta R$ 100k+ (benchmark). Ticket 52% abaixo da média. Zero investimento em aquisição.'",
    "goal_gap_analysis": "Se meta definida: gap atual e como as 18 sugestões cobrem pelo menos 80%",
    "top_5_problems": ["problema 1 com dado", "problema 2 com dado", "problema 3 com dado", "problema 4 com dado", "problema 5 com dado"],
    "market_opportunities": ["oportunidade 1", "oportunidade 2", "oportunidade 3", "oportunidade 4", "oportunidade 5"],
    "categories_to_cover": ["strategy", "investment", "market", "growth", "conversion", "product", "coupon", "pricing", "customer", "inventory"],
    "themes_to_avoid": ["tema saturado 1", "tema saturado 2"],
    "approach_rationale": "Explicação de 2-3 frases: por que estas 6 estratégicas + 12 táticas",
    "high_alternatives": [
      {
        "chosen": "Título da HIGH #1 escolhida",
        "alternative_1": "Abordagem alternativa - descartada: motivo",
        "alternative_2": "Outra alternativa - descartada: motivo"
      }
    ],
    "analysis_context": {
      "main_problems": ["problema 1", "problema 2", "problema 3"],
      "main_opportunities": ["oportunidade 1", "oportunidade 2"],
      "avoided_themes": ["tema já sugerido antes 1", "tema já sugerido antes 2"]
    },
    "suggestions": [
      {
        "react": {
          "thought": "Qual dado/problema motivou esta sugestão (com números)",
          "action": "Qual ação específica resolve isso (passos resumidos)",
          "observation": "Qual resultado esperar (R$ ou %)"
        },
        "priority": 1,
        "text": "Sugestão 1: Descrever a estratégia HIGH com base na análise executiva estruturada no Growth Intelligence Framework™. Incluir benchmarks, concorrentes, recursos da plataforma, sazonalidade e histórico. Especificar 18 sugestões (6 HIGH, 6 MEDIUM, 6 LOW) com evidências sólidas e impacto potencial. Implementar a estratégia HIGH para alcançar a meta definida, considerando o gap atual e a cobertura das 18 sugestões. Abordar os 5 principais problemas identificados e explorar 5 oportunidades de mercado. Cobrir todas as categorias estratégicas: strategy, investment, market, growth, conversion, product, coupon, pricing, customer, inventory. Evitar temas saturados. Explorar 12 táticas adicionais para complementar as 6 estratégicas principais. Selecionar a alternativa HIGH #1 com base em critérios específicos, mencionando as outras alternativas descartadas e seus motivos. Considerar o contexto principal de problemas e oportunidades, bem como temas evitados. Propor ações e observações para o resultado esperado. Priorizar a sugestão 1 com base em critérios estabelecidos."
      }
    ]
  }
}
```

```

    "expected_impact": "high",
    "category": "strategy|investment|market|growth|financial|positioning|inventory|pricing|product|customer|conversion|marketing|coupon|operational",
    "title": "Título específico com número quando possível",
    "problem": "Descrição do problema com dados específicos da loja",
    "action": "Passos numerados e específicos",
    "expected_result": "Resultado esperado com número (R$ ou %)",
    "data_source": "De onde veio o dado que embasa esta sugestão",
    "implementation": {
        "type": "nativo|app|terceiro",
        "app_name": "nome se aplicável ou null",
        "complexity": "baixa|media|alta",
        "cost": "R$ X/mês ou R$ 0"
    },
    "competitor_reference": "Se HIGH: qual dado de concorrente ou mercado embasa isso. Se não há: null",
    "insight_origem": "problema_1|problema_2|problema_3|problema_4|problema_5|best_practice (qual problema do Analyst esta sugestão resolve)",
    "nivel_confianca": "alto|medio|baixo"
},
],
"premium_summary": {
    "executive_summary": {
        "resumo_direto": {
            "nao_precisa": "Frase do que NÃO precisa (ex: 'vender mais barato')",
            "precisa": ["ação estratégica 1", "ação estratégica 2", "ação estratégica 3"],
            "potencial_real": ["área de potencial 1", "área de potencial 2", "área de potencial 3"]
        },
        "diagnosticos_principal": "Diagnóstico central em até 5 linhas",
        "maior_gargalo": "Principal gargalo estrutural",
        "maior_oportunidade": "Maior oportunidade financeira escondida",
        "risco_mais_relevante": "Risco mais relevante",
        "potencial_crescimento_estimado_percentual": 0
    },
    "growth_score": {
        "overall_score": 0,
        "efficiency_score": 0,
        "margin_health": 0,
        "retention_score": 0,
        "scale_readiness": "Operacional|Estruturada|Escalável|Otimizada|Dominante"
    },
    "diagnosticos_quantitativo": {
        "ticket_medio_vs_benchmark": "Avaliação com números",
        "dependencia_desconto": "Avaliação com números",
        "risco_margem": "Avaliação com números",
        "estrutura_catalogo": "Avaliação com números",
        "potencial_retencao": "Avaliação com números"
    },
    "gaps_estrategicos": {
        "dados_ausentes": ["gap 1", "gap 2"],
        "estruturais": ["gap 1", "gap 2"],
        "operacionais": ["gap 1", "gap 2"],
        "estrategicos": ["gap 1", "gap 2"]
    },
    "financial_opportunities": [
    ]
}

```

```

        "action": "Descrição da oportunidade",
        "impact_type": "ticket|retention|conversion|margin",
        "estimated_monthly_impact": 0,
        "estimated_annual_impact": 0
    }
],
"prioritized_roadmap": {
    "30_dias": ["ação quick win 1", "ação quick win 2"],
    "60_dias": ["ação estruturação 1", "ação estruturação 2"],
    "90_dias": ["ação escala 1", "ação escala 2"]
},
"impact_effort_matrix": {
    "quick_wins": ["alto impacto, baixo esforço"],
    "high_impact": ["alto impacto, alto esforço"],
    "fill_ins": ["baixo impacto, baixo esforço"],
    "avoid": ["baixo impacto, alto esforço"]
},
"growth_scenarios": {
    "conservador": {
        "crescimento_percentual": 10,
        "receita_mensal_projetada": 0,
        "receita_anual_projetada": 0,
        "o_que_precisa_melhorar": ""
    },
    "base": {
        "crescimento_percentual": 25,
        "receita_mensal_projetada": 0,
        "receita_anual_projetada": 0,
        "o_que_precisa_melhorar": ""
    },
    "agressivo": {
        "crescimento_percentual": 50,
        "receita_mensal_projetada": 0,
        "receita_anual_projetada": 0,
        "o_que_precisa_melhorar": ""
    }
},
"strategic_risks": ["risco 1", "risco 2", "risco 3"],
"final_verdict": {
    "conclusao_estrategica": "Resumo executivo final em até 5 linhas",
    "current_stage": "Operacional|Estruturada|Escalável|Otimizada|Dominante",
    "next_stage_requirement": "O que falta para subir de nível"
}
}
}
}

```

8.4.2) Campos de sugestão (detalhe por detalhe)

Campo	O que significa	O que deve conter (regras do prompt)
priority	Ordem de prioridade 1..18.	HIGH normalmente ocupam 1–6; MEDIUM 7–12; LOW 13–18 (não é obrigatório, mas é esperado).
expected_impact	Nível do impacto (high/medium/low).	O agente deve garantir 6-6-6; cada item precisa ser coerente com o impacto estimado.

Campo	O que significa	O que deve conter (regras do prompt)
category	Categoria do tema (estratégico ou tático).	HIGH não devem cair em categorias táticas; se cair, deve ser rebaixada para MEDIUM.
title	Título curto e acionável.	Deve conter número (R\$ ou %) sempre que possível e ser específico da loja.
problem	Problema diagnosticado com evidência.	Deve citar números reais do input; em HIGH, deve ser inevitavelmente "executivo/estrutural".
action	Passos numerados e executáveis.	Sem generalidades; deve apontar como fazer e, quando possível, usar recursos da plataforma.
expected_result	Resultado estimado (R\$ ou %).	O prompt exige quantificação. Deve explicitar premissas (base × taxa = impacto) quando necessário.
implementation	Como implementar e custo.	Tipo (nativo/app/terceiro), complexidade e custo aproximado.
competitor_reference	Âncora externa.	Para HIGH, deve citar concorrentes/mercado/benchmark quando dados existirem; caso contrário pode ser null.
insight_origem	Rastreabilidade com o Analyst.	Deve apontar qual problema do briefing está sendo resolvido (problema_1..problema_5).
nivel_confianca	Confiança do Strategist.	Deve refletir qualidade do dado (alto/medio/baixo).

8.4.3) Premium Summary (o “Resumo Estratégico” exibido na UI)

O premium_summary é o artefato “executivo” do pipeline: uma leitura estruturada (de negócio) que transforma diagnóstico + dados + benchmarks em narrativa de crescimento, oportunidades financeiras e plano de ação. Diferente das *sugestões* (itens acionáveis isolados), o Premium Summary organiza a história do crescimento em seções fixas (Growth Intelligence Framework™) para responder, com clareza: “onde estamos”, “por que” e “o que fazer agora”.

8.4.3.1) Onde ele aparece (persistência/API/UI)

No backend FULL, o premium_summary é extraído do JSON do Strategist e salvo dentro de analysis.summary.premium_summary. A API expõe esse objeto em premium_summary (atalho) e também dentro de summary. No Frontend, o componente StrategicSummaryPanel.vue renderiza cada seção do framework.

- **Estrutura rígida:** a UI não “inventa” campos; se uma seção faltar, o painel perde blocos.

- **Números podem ser estimativas:** mas devem ser coerentes e rastreáveis (base × premissa = impacto).
- **Sem validação forte no código:** o sistema confia no contrato do prompt (qualidade depende do agente).

Armadilha comum (janela de dados): apesar de nomes como `pedidos_mes` e `faturamento_mes` aparecerem no contexto/prompt, no pipeline FULL esses valores normalmente representam o **período de análise** (ex.: 15 dias; ver `orders_summary.period_days`). Se o Strategist calcular cenários “mensais” usando um faturamento de 15 dias, a projeção pode ficar inflada ou subestimada. A forma correta é normalizar para 30 dias quando o texto/valor exigir “por mês”.

8.4.3.2) Regras do prompt (o que é obrigatório)

- **Seções obrigatórias:** `executive_summary`, `growth_score`, `diagnostico_quantitativo`, `gaps_estrategicos`, `financial_opportunities`, `prioritized_roadmap`, `impact_effort_matrix`, `growth_scenarios`, `strategic_risks` e `final_verdict`.
- **executive_summary.resumo_direto_completo:** `nao_precisa` (string), `precisa` (array 3–5 itens) e `potencial_real` (array 3–4 itens).
- **growth_score preenchido:** `overall_score`, `efficiency_score`, `margin_health` e `retention_score` devem estar entre 0 e 100, e `scale_readiness` deve refletir a maturidade (Operacional → Dominante).
- **growth_scenarios com cálculo:** cada cenário deve ter `receita_mensal_projetada` e `receita_anual_projetada` calculados com base em dados reais do contexto (quando existirem).

8.4.3.3) executive_summary (como preencher, campo a campo)

Esta seção é a “capa executiva”. Ela precisa ser curta, mas **não genérica**: deve citar 1–3 números que sustentem o diagnóstico (ex.: ticket vs benchmark, % de ruptura, % de cupom, receita do período).

- **resumo_direto.nao_precisa:** o que *não* resolve o gargalo central (ex.: “vender mais barato”, “postar mais sem funil”).
- **resumo_direto.precisa:** 3–5 ações executivas (ex.: corrigir ruptura em best-sellers, elevar ticket via kits, criar investimento mínimo em aquisição com CAC controlado).
- **resumo_direto.potencial_real:** 3–4 alavancas de potencial (ex.: ticket, conversão, retenção, margem).

- **potencial_crescimento_estimado_percentual:** número 0–100 de potencial (leitura executiva; não confundir com growth_scenarios).

```
{
  "executive_summary": {
    "resumo_direto": {
      "nao_precisa": "baixar preços para competir",
      "precisa": [
        "repor best-sellers (ruptura está travando conversão)",
        "re定位ar ticket com kits (ticket está 40% abaixo do benchmark)",
        "investir em aquisição com CAC controlado (demanda do nicho em alta)"
      ],
      "potencial_real": [
        "aumentar ticket médio via bundles",
        "reduzir dependência de cupom com proposta de valor",
        "elevar conversão nas páginas mais visitadas"
      ]
    },
    "diagnostico_principal": "A loja tem demanda e produtos vencedores, mas perde receita por ruptura e por ticket abaixo do benchmark. O crescimento depende de corrigir operação (estoque) e ajustar posicionamento (valor percebido) antes de escalar aquisição.",
    "maior_gargalo": "Ruptura de best-sellers e baixa disponibilidade do catálogo",
    "maior_oportunidade": "Aumentar ticket com kits e precificação baseada em benchmark",
    "risco_mais_relevante": "Margem corroída por cupom/frete sem elevar valor percebido",
    "potencial_crescimento_estimado_percentual": 35
  }
}
```

8.4.3.4) growth_score (scores 0–100 + maturidade)

O Growth Score é uma quantificação (heurística) do quanto preparada a loja está para crescer com eficiência. O prompt exige valores de 0 a 100, mas **não define uma fórmula determinística**; o Strategist deve inferir a pontuação a partir das métricas disponíveis (internas + benchmarks + externos).

- **efficiency_score:** capacidade de converter e operar (cancelamento, ruptura, funil).
- **margin_health:** saúde de margem e disciplina de descontos (cupom/frete vs ticket).
- **retention_score:** potencial de recompra/recorrência (quando houver sinais no histórico).
- **overall_score:** leitura agregada (recomendado: média ponderada dos anteriores).

O **scale_readiness** deve traduzir a maturidade em um estágio: Operacional (base frágil) → Estruturada (base ok, gaps claros) → Escalável (pode investir e crescer) → Otimizada (processos e métricas sob controle) → Dominante (lidera nicho).

8.4.3.5) diagnostico_quantitativo (strings que devem conter números)

Apesar de serem campos textuais, estes itens devem conter comparações numéricas (diferença absoluta e/ou percentual), sempre que o contexto trouxer dados suficientes.

- **ticket_medio_vs_benchmark:** ticket atual vs benchmark do nicho + gap (%).
- **dependencia_desconto:** uso de cupons (% de pedidos) e impacto no ticket.
- **risco_margem:** riscos de margem (desconto total, frete, pricing vs mercado).
- **estrutura_catalogo:** catálogo ativo, ruptura, best-sellers vs long tail.
- **potencial_retencao:** sinais de recompra/relacionamento ou gaps de pós-compra.

8.4.3.6) financial_opportunities (como calcular impacto)

Cada item deve representar uma alavancagem com impacto financeiro estimado. Recomendação de cálculo (conforme filosofia do prompt): explicitar a base e a premissa.

- **Ticket:** pedidos_base × +R\$ ticket.
- **Conversion:** visitas_base × (taxa_nova - taxa_atual) × ticket.
- **Retention:** clientes_base × +% recompra × ticket.
- **Margin:** receita_base × +% margem.

Observação: quando só existirem dados de pedidos/ticket do período, o agente pode estimar impacto “mensal” normalizando para 30 dias usando orders_summary.period_days.

8.4.3.7) growth_scenarios (projeções conservador/base/agressivo)

Os cenários devem transformar um baseline em projeções mensais e anuais. O baseline mais comum é a receita do período (ticket_medio × pedidos_mes) ajustada para 30 dias quando o contexto não for mensal.

```
// Exemplo de normalização (quando period_days = 15):
receita_15d = ticket_medio * pedidos_15d
receita_mensal_estimada = receita_15d * (30 / 15)
receita_anual_estimada = receita_mensal_estimada * 12
```

Para cada cenário, o Strategist deve explicar o_que_precisa_melhorar (ex.: “estoque + ticket + aquisição”) e manter as projeções coerentes com o restante do diagnóstico.

8.4.3.8) final_verdict (conclusão + próximo degrau)

Esta seção fecha o framework: qual o estágio atual e o que falta para subir de nível. Deve ser coerente com growth_score, com o diagnóstico quantitativo e com as recomendações HIGH.

8.4.4) Normalização no backend (StrategistAgentService)

O backend valida e normaliza as sugestões antes de passá-las para o Critic. Ele aceita variações de formato (V4/V5), converte nomes de campos e padroniza expected_impact.

Também pode rebaixar HIGH sem dados específicos.

Nota técnica: a execução do Strategist rota temperatura por contagem de análises anteriores (0.5, 0.6, 0.7, 0.65, 0.55).

8.5) Critic Agent (validação e seleção final de 9 sugestões)

O Critic é o “filtro de qualidade” final de IA. Ele recebe as 18 sugestões do Strategist (6 HIGH + 6 MEDIUM + 6 LOW) e deve:

- validar números e consistência (não aceitar métricas inventadas);
- rejeitar temas repetidos/saturados e sugestões genéricas;
- melhorar sugestões promissoras (corrigir cálculos, tornar ações específicas e viáveis);
- selecionar as **9 melhores** para entrega final, com distribuição **3 HIGH + 3 MEDIUM + 3 LOW**.

Regra crítica (HIGH): as 3 HIGH finais devem ser estratégicas (categorias como strategy/investment/market/growth/financial/positioning) e, quando existirem dados externos, precisam ancorar em evidência (concorrentes/mercado/benchmarks).

8.5.1) Schema (JSON — conforme prompt)

```
{
  "reasoning": {
    "quality_assessment": "Avaliação geral das 18 sugestões recebidas",
    "selection_criteria": "Por que estas 9 foram escolhidas e não as outras 9",
    "decisions_summary": "X selecionadas, Y melhoradas, Z descartadas",
    "weak_spots": ["sugestão N: motivo da fraqueza/descarte"],
    "improvements_made": ["sugestão N: o que foi melhorado e por quê"]
  },
  "review_summary": {
    "received": 18,
    "selected": 9,
    "approved": 0,
    "improved": 0,
    "discarded": 0,
    "replacements_created": 0
  },
  "suggestions": [
    {
      "review_react": {
        "thought": "Análise da qualidade: dados reais? ação viável? resultado quantificado?",
        "action": "APROVAR/MELHORAR/REJEITAR - justificativa",
        "observation": "O que mudou, quality score estimado"
      },
      "original_title": "Título original do Strategist",
    }
  ]
}
```

```

    "status": "approved|improved|replaced",
    "changes_made": "Nenhuma | Descrição das melhorias | Motivo da substituição",
    "verificacoes": {
        "V1_numeros": {"resultado": "ok|corrigido|nao_verificavel", "detalhe": "Ticket médio confere: R$ 85"}, 
        "V2_originalidade": {"resultado": "ok|rejeitado", "detalhe": "Tema inédito"}, 
        "V3_especificidade": {"resultado": "ok|rejeitado|melhorado", "detalhe": "Usa dados específicos da loja"}, 
        "V4_viabilidade": {"resultado": "ok|rejeitado", "detalhe": "Viável via Nuvemshop nativo"}, 
        "V5_impacto": {"resultado": "ok|corrigido|nao_verificavel", "detalhe": "120 pedidos × R$85 × 15% = R$1.530/mês"}, 
        "V6_alinhamento": {"resultado": "ok|nao_aplicavel", "detalhe": "Resolve problema_1 do Analyst"} 
    },
    "verificacao_status": "VERIFICADA|DADO_CORRIGIDO|NAO_VERIFICAVEL",
    "score_qualidade": 8,
    "final": {
        "priority": 1,
        "expected_impact": "high|medium|low",
        "category": 
    }
}
"strategy|investment|market|growth|financial|positioning|inventory|pricing|product|customer|conversion|marketing|coupon|operational",
    "title": "Título final (pode ser igual ao original ou melhorado)",
    "problem": "Problema com dado específico",
    "action": "Passos numerados",
    "expected_result": "Resultado com número (R$ ou %)",
    "data_source": "Fonte do dado",
    "implementation": {
        "type": "nativo|app|terceiro",
        "app_name": "nome ou null",
        "complexity": "baixa|media|alta",
        "cost": "R$ X/mês ou R$ 0"
    },
    "competitor_reference": "Obrigatório para HIGH quando dados disponíveis; opcional para MEDIUM/LOW",
    "insight_origem": 
}
"problema_1|problema_2|problema_3|problema_4|problema_5|best_practice",
    "nivel_confianca": "alto|medio|baixo"
}
],
"distribution_check": {"high": 3, "medium": 3, "low": 3, "valid": true},
"competitor_citations_check": {
    "count": 2,
    "minimum_required": 2,
    "valid": true,
    "competitors_cited": ["Concorrente A", "Concorrente B"]
},
"temas_rejeitados_por_saturacao": ["Tema 1", "Tema 2"],
"quality_summary": {
    "total_verificadas": 0,
    "total_corrigidas": 0,
    "total_nao_verificaveis": 0,
    "score_medio": 0
}

```

```

    }
}

```

8.5.2) Framework de verificação V1–V6 (como o Critic decide)

O prompt obriga o Critic a executar 6 verificações em **cada** sugestão. O objetivo é impedir recomendações “bonitas” porém erradas, repetidas ou inviáveis.

- **V1 — Números:** checa se ticket, quantidade de SKUs, percentuais e faturamento conferem com os dados de entrada (ex.: `orders_summary`, `products_summary`, `inventory_summary`, `coupons_summary`). Se divergir, corrige e recalcula impacto.
- **V2 — Originalidade:** rejeita temas já saturados no histórico. O Critic recebe lista de temas saturados e também sugestões anteriores para comparar e evitar repetição.
- **V3 — Especificidade:** se a sugestão servir para “qualquer loja”, ela é rejeitada ou reescrita para ficar específica (com números reais e ações compatíveis com a loja analisada).
- **V4 — Viabilidade:** valida se é possível na plataforma (Nuvemshop) conforme recursos permitidos no prompt. Sugestões em zonas “IMPOSSÍVEL” devem ser rejeitadas.
- **V5 — Impacto:** exige que o resultado seja verificável ($\text{base} \times \text{premissa} = \text{impacto}$). Se faltar, o Critic completa o cálculo e/ou ajusta impacto para algo realista.
- **V6 — Alinhamento:** verifica se resolve um dos 5 problemas do Analyst (obrigatório para HIGH; prioriza problema_1..problema_3).

8.5.3) Normalização no backend (CriticAgentService)

O backend aceita variações de formato e normaliza para um array de `approvedSuggestions`, cada uma com `finalVersion` (o que será persistido) e metadados de qualidade/validação.

- suporta formato “V5” (itens com `status + final`) e formato legado “V4” (`approvedSuggestions`);
- normaliza campos: `problem → description`, `action → recommendedAction`, `dataSource → dataJustification`;
- propaga metadados: `verificacaoStatus` e `scoreQualidade` são anexados em `specificData` ao salvar;
- **enforceDistribution:** mantém no máximo 3 por nível (HIGH/MEDIUM/LOW) e reatribui prioridades 1..N (alvo 9).

Importante: a distribuição 3-3-3 é reforçada por código (pós-normalização). Se o Critic retornar mais itens em algum nível, o excesso pode ser descartado pelo slicing.

8.6) Lite Analyst (diagnóstico rápido: métricas + até 3 anomalias)

O Lite Analyst é usado exclusivamente no pipeline LITE. Ele existe para reduzir tokens e latência: recebe um `store_data` compactado (7 dias) e retorna:

- **metrics**: métricas essenciais (vendas, AOV, cancelamento, estoque e cupons);
- **anomalies**: no máximo 3 anomalias priorizadas por impacto financeiro;
- **overall_health**: score 0–100 + classificação + pontos principais.

8.6.1) Entrada (`store_data` compactado — `LiteStoreAnalysisService`)

O backend monta um payload com janela reduzida (`period_days = 7`) e sem listas extensas. Estrutura (exemplo de shape):

```
{
  "period_days": 7,
  "orders": {
    "total": 0,
    "total_revenue": 0,
    "average_order_value": 0,
    "cancellation_rate": 0
  },
  "products": {
    "total": 0,
    "active": 0,
    "out_of_stock": 0,
    "low_stock": 0,
    "best_sellers_count": 0,
    "gifts_filtered": 0
  },
  "coupons": {
    "usage_rate": 0,
    "ticket_impact": 0,
    "total_discount": 0
  }
}
```

Observação: o pipeline LITE não envia “top 10 best sellers com detalhes” nem “lista completa de produtos sem estoque” — ele trabalha com contagens e poucos agregados.

8.6.2) Schema (JSON — conforme prompt)

```
{
  "metrics": {
    "sales": {"total": 0, "daily_average": 0, "trend": "crescendo|estável|caindo"},
    "average_order_value": {"value": 0, "benchmark": 150},
    "cancellation_rate": 0,
    "inventory": {"out_of_stock_products": 0, "critical_stock_products": 0},
    "coupons": {"usage_rate": 0, "ticket_impact": 0}
  },
  "anomalies": [
    ...
  ]
}
```

```

    {"type": "string", "description": "string", "severity": "alto|médio|baixo"}
  ],
  "overall_health": {
    "score": 0,
    "classification": "crítico|atenção|saudável|excelente",
    "main_points": ["ponto 1", "ponto 2"]
  }
}

```

8.6.3) Cálculo do Health Score (Lite — pesos do prompt)

O prompt define 5 componentes (somando 100 pontos) e orienta “subtrair proporcionalmente” quando a métrica estiver abaixo do ideal:

- **Estoque saudável (sem ruptura de best-sellers): +30**
- **Cancelamento ≤ 3%: +25**
- **Ticket médio ≥ benchmark: +20**
- **Tendência de vendas estável/crescendo: +15**
- **Uso de cupons sem impacto negativo no ticket: +10**

Nota: diferente do FULL (onde o Analyst tem regras mais extensas e um modelo de overrides), aqui o cálculo é mais simplificado e depende de heurística do agente ao “subtrair proporcionalmente”.

8.6.4) Normalização no backend (LiteStoreAnalysisService)

O backend normaliza a estrutura para garantir chaves mínimas, mesmo quando o JSON vier incompleto:

- faz merge de metrics e overall_health com defaults;
- se o parse falhar, usa um default (score = 50, classification = attention) e mantém a análise concluída;
- transforma anomalies diretamente em alerts na análise LITE (não há oportunidades).

8.7) Lite Strategist (6 sugestões — distribuição 2-2-2)

O Lite Strategist é a etapa de recomendações do pipeline LITE. Ele recebe o resultado do Lite Analyst e deve gerar **exatamente 6** sugestões acionáveis, com impacto distribuído em **2 HIGH + 2 MEDIUM + 2 LOW**. Não existe Critic no LITE; a validação é básica (campos obrigatórios + normalização de impacto) e depois entram os filtros de unicidade contra histórico.

8.7.1) Critérios de impacto (regras do prompt)

- **HIGH:** potencial > R\$ 5.000/mês ou melhoria de conversão > 20%.
- **MEDIUM:** potencial R\$ 1.000–5.000/mês ou melhoria de conversão 5–20%.
- **LOW:** potencial < R\$ 1.000/mês ou melhoria operacional/UX.

8.7.2) Schema (JSON — conforme prompt)

```
{
  "suggestions": [
    {
      "category": "strategy|investment|market|growth|financial|positioning|inventory|pricing|product|customer|conversion|marketing|coupon|operational",
      "title": "Título com número específico (máx 100 chars)",
      "description": "Problema identificado com base nos dados",
      "recommended_action": "1. Passo um\n2. Passo dois\n3. Passo três",
      "expected_impact": "high|medium|low",
      "target_metrics": [
        "receita|conversao|ticket_medio|volume_pedidos|estoque|margem|recompra|abandono"],
        "implementation_time": "immediate|1_week",
        "specific_data": {"chave": "valor extraído da análise"},
        "data_justification": "Fonte do dado na análise fornecida"
      ]
    }
  ]
}
```

8.7.3) Normalização/validação no backend (LiteStoreAnalysisService)

Após extrair o JSON, o backend aplica validação mínima e adapta o formato para o salvamento reutilizando o método comum de persistência:

- exige campos obrigatórios: category, title, description, expected_impact;
- normaliza expected_impact para high|medium|low (valores fora disso viram medium);
- embrulha cada sugestão em { final_version, final_priority } para compatibilidade com o salvamento;
- deduplica contra histórico (tema/título) antes de persistir — isso pode reduzir o total abaixo de 6.

Como não existe Critic no LITE, a qualidade final depende muito do cumprimento das regras do prompt (título com número, ações implementáveis e impacto coerente).

8.8) SimilarityCheck (prompt auxiliar — atualmente não utilizado no pipeline)

Existe um prompt chamado `SimilarityCheckPrompt` cujo objetivo é analisar **todas** as sugestões anteriores e produzir “zonas proibidas” (reformulações a evitar) e “abordagens permitidas” (oportunidades ainda não exploradas). Ele foi desenhado para reduzir repetição semântica *antes* do Strategist gerar novas recomendações.

Importante: na versão atual do código, esse prompt **não é executado** em nenhuma etapa (não há chamadas para `SimilarityCheckPrompt` fora do próprio arquivo). A prevenção de repetição é feita por: ThemeKeywords (saturação), Jaccard nos títulos e embeddings (pgvector), conforme seção 7.

8.8.1) Schema (JSON — conforme prompt)

```
{
  "prohibited_zones": [
    {
      "id": 0,
      "original_title": "string",
      "problem_category": "estoque|ticket|conversao|retencao|cupons|marketing|operacional|produto",
      "problem_description": "string",
      "solution_type": "reposicao|desconto|email|fidelidade|upsell|crosssell|bundle|social|conteudo|ux",
      "keywords": [],
      "prohibited_variations": ["variação 1", "variação 2", "variação 3"]
    }
  ],
  "allowed_approaches": {
    "estoque": [],
    "ticket": [],
    "conversao": [],
    "retencao": [],
    "cupons": [],
    "marketing": [],
    "operacional": [],
    "produto": []
  },
  "coverage_summary": {
    "categories_covered": [],
    "categories_gaps": [],
    "total_analyzed": 0
  },
  "strategist_guidance": "Resumo do que evitar e onde há oportunidades"
}
```

8.8.2) Como seria usado (se fosse integrado)

- entrada: `previous_suggestions` (histórico completo ou recorte);
- saída: `prohibited_zones` alimentaria o bloco de “zonas proibidas” do Strategist/Critic;

- `allowed_approaches` ajudaria a forçar diversidade por categoria (oportunidades ainda não cobertas).

Como esse prompt não roda hoje, a diversidade é garantida por regras do Strategist/Critic e por filtros programáticos (saturação por keyword + similaridade por título/embedding).

9) Persistência, API e UI (como os dados aparecem)

Esta seção descreve como os resultados do pipeline viram dados persistidos e, em seguida, como são expostos pela API e renderizados no Frontend.

9.1) Persistência: Analysis (tabela analyses)

A entidade Analysis funciona como o “envelope” da execução. Ao final, o serviço marca como concluída (método `markAsCompleted()`) e salva:

- **summary (array):** resumo leve para UI (health + insight + premium_summary no FULL).
- **alerts (array):** até 3 alertas derivados do diagnóstico (FULL: alertas V4 ou fallback anomalies).
- **opportunities (array):** oportunidades derivadas de oportunidades (V3) ou identified_patterns (legado).
- **raw_agent_outputs (array):** outputs brutos por agente (debug/admin).
- **progresso:** current_stage, total_stages, stage_data e logs.

9.2) Estrutura do summary (FULL vs LITE)

O backend monta um summary com as seguintes chaves principais:

Chave	Tipo	Como é calculado
health_score	number	FULL: <code>health_score.score_final</code> (V3) ou <code>overall_health.score</code> (legado). LITE: <code>overall_health.score</code> .
health_status	string	FULL: <code>health_score.classificacao</code> ou <code>overall_health.classification</code> . LITE: <code>overall_health.classification</code> .
main_insight	string	FULL: <code>resumo_executivo</code> (preferencial), fallback em <code>health_score.explicacao</code> ou pontos do legacy.
premium_summary	object null	Somente FULL: extraído do Strategist (Growth Intelligence Framework™).

9.3) Como alerts são extraídos (FULL/LITE)

O sistema expõe alertas “quentes” (danger/warning) para chamar atenção sem poluir a UI:

- **FULL:** prioriza `alerts.criticos` (danger) e `alerts.atencao` (warning) e limita a 3.

- **Fallback FULL:** se não houver alertas V4, usa anomalies (ordenadas por severidade) e também limita a 3.
- **LITE:** converte todas as anomalies em alerts (sem oportunidades).

9.4) Como opportunities são extraídas (FULL)

O FULL mapeia oportunidades do output do Analyst (V3: oportunidades, legado: identified_patterns) para um formato consistente:

- title e description (textos)
- potential_revenue (quando existir)
- type, base_dados, calculo_roi (rastreabilidade do cálculo)

9.5) Persistência: Suggestion (tabela suggestions)

O pipeline FULL persiste apenas as sugestões aprovadas (pós-Critic e filtros). Cada registro salva a versão final normalizada, com rastreabilidade e metadados:

- **category, title, description:** texto final.
- **recommended_action:** pode ser string (com quebras de linha) ou JSON array (melhor para "steps").
- **expected_impact:** high|medium|low.
- **priority:** ordem final 1..N (a UI usa como priority_order).
- **specific_data:** anexa detalhes como expected_result, implementation, competitor_reference, insight_origem, nivel_confianca, verificacao_status e score_qualidade.

Observação: ao aceitar uma sugestão para acompanhamento, o backend tenta sincronizar "steps" a partir de recommended_action — isso só funciona automaticamente quando recommended_action é um array.

9.6) Contrato da API (Resources)

A API expõe a análise via AnalysisResource e sugestões via SuggestionResource. Exemplo (shape, campos principais):

```
{
  "id": "uuid",
  "status": "Completed",
  "analysis_type": "general",
  "summary": {
    "health_score": 72,
    "details": [
      {
        "name": "Insight 1",
        "value": "High Risk"
      },
      {
        "name": "Insight 2",
        "value": "Medium Risk"
      }
    ]
  }
}
```

```

"health_status": "saudavel",
"main_insight": "Resumo executivo do Analyst",
"premium_summary": { "...": "Growth Intelligence Framework" }
},
"premium_summary": { "...": "atalho do summary.premium_summary" },
"alerts": [{"type": "danger|warning", "title": "...", "message": "..."}],
"opportunities": [{"title": "...", "description": "...", "potential_revenue": 0}],
"suggestions": [
{
"id": "uuid",
"category": "inventory|pricing|...",
"title": "...",
"description": "...",
"recommended_action": ["passo 1", "passo 2"],
"expected_impact": "high|medium|low",
"priority": "high|medium|low",
"priority_order": 1,
"specific_data": {"expected_result": "...", "score_qualidade": 8}
}
],
"current_stage": 9,
"total_stages": 9,
"progress_percentage": 100
}

```

9.7) Como a UI consome (Frontend)

- **Resumo Estratégico:** StrategicSummaryPanel.vue lê premium_summary e renderiza seções (executivo, score, oportunidades, roadmap, cenários, riscos e veredito).
- **Passos da sugestão:** SuggestionStepsPanel.vue aceita recommended_action como array ou string; se for string, tenta parsear JSON e depois quebra por linhas/numeração.
- **Prioridade:** a UI usa priority (que mapeia expected_impact) e usa priority_order para ordenação exibida.

10) Exemplos fim-a-fim (concretos)

Os exemplos abaixo são **hipotéticos** e servem para demonstrar, com números e passo a passo, como o pipeline conecta dados → diagnóstico → recomendações → validação → resultado persistido/API/UI.

Nota: como parte do processo é inferencial (IA), o conteúdo textual exato varia. O que não deveria variar é o *contrato* (campos/estrutura) e a coerência numérica (base × premissa = impacto).

10.1) Exemplo FULL (15 dias) — ruptura + dependência de cupom

10.1.1) Dados internos (entrada do pipeline)

O backend prepara agregados da janela FULL (period_days = 15). Exemplo simplificado:

```
{
  "orders_summary": {
    "period_days": 15,
    "total": 120,
    "total_revenue": 10200,
    "average_order_value": 85,
    "cancellation_rate": 6.0
  },
  "products_summary": {
    "total": 300,
    "active": 240,
    "out_of_stock": 60
  },
  "coupons_summary": {
    "usage_rate": 40,
    "avg_discount_percent": 18,
    "ticket_impact_percent": -12
  }
}
```

A partir disso, o pipeline também calcula auxiliares usados pelos agentes (ex.: out_of_stock_pct, ticket_medio, pedidos_mes = total do período).

10.1.2) Analyst: diagnóstico + Health Score + alertas/oportunidades

Com os números acima, o Analyst gera um briefing (problema_1..problema_5), oportunidades e calcula o Health Score. Aplicando a regra de pontuação documentada na seção 8.3:

- **Estoque:** 60/300 = 20% sem estoque → **+20**

- **Cancelamento:** 6% → **+0** (acima de 5%)
- **Ticket:** R\$ 85 vs benchmark (ex.: R\$ 110) → 0,77 → **+10**
- **Pedidos:** tendência estável → **+10**
- **Cupons:** 40% de uso → **+0** (acima de 35%)

Score base = 20 + 0 + 10 + 10 + 0 = **40**. Sem overrides adicionais, score_final = 40 e classificacao = atencao (26–50).

Exemplo de shape de saída do Analyst (parcial):

```
{
  "resumo_executivo": "A loja está perdendo vendas por ruptura e opera com dependência alta de cupom, corroendo ticket e margem.",
  "health_score": {
    "score_base": 40,
    "score_final": 40,
    "classificacao": "atencao",
    "componentes": { "...": "..." },
    "overrides": []
  },
  "alertas": {
    "criticos": [{"tipo": "stock_out", "titulo": "Ruptura de estoque", "descricao": "20% do catálogo sem estoque..."}],
    "atencao": [{"tipo": "coupon_dependency", "titulo": "Alta dependência de cupom", "descricao": "40% dos pedidos com cupom..."}],
    "monitoramento": []
  },
  "oportunidades": [
    {"tipo": "bundle_opportunity", "titulo": "Kits para elevar ticket", "potencial_receita": "R$ 3.000-6.000/mês"}
  ]
}
```

10.1.3) Strategist: 18 sugestões + Premium Summary

O Strategist transforma o briefing em 18 sugestões (6/6/6) e preenche o premium_summary. Exemplo de sugestão HIGH estratégica (parcial):

```
{
  "expected_impact": "high",
  "category": "market",
  "title": "Reposicionar ticket de R$ 85 para R$ 105 (+24%) com kits e valor percebido (benchmark nicho R$ 110)",
  "problem": "Ticket médio R$ 85 está ~23% abaixo do benchmark do nicho (R$ 110), limitando margem e capacidade de investir.",
  "action": "1. Criar 6 kits com best-sellers (faixa R$ 99-149)\n2. Ajustar pricing dos top 5 SKUs +8%\n3. Adicionar prova social (reviews) nas páginas dos kits",
  "expected_result": "Normalizando para 30 dias: receita_15d=R$10.200 → ~R$20.400/mês. Se +R$20 no ticket em 240 pedidos/mês: +R$4.800/mês.",
  "data_source": "orders_summary + benchmark do nicho + catálogo/estoque",
```

```

    "competitor_reference": "Concorrentes do nicho operam com ticket 20-30% maior e usam kits
para elevar valor percebido."
}

```

Para o Premium Summary, o agente deve normalizar o baseline para “mensal” quando necessário (se o contexto vier em 15 dias). Exemplo de cenário base:

```

receita_15d = 10.200
receita_mensal_estimada = 10.200 * (30/15) = 20.400
cenário base (+25%) = 25.500/mês | 306.000/ano

```

10.1.4) Critic: valida, corrige e seleciona 9 (3-3-3)

O Critic checa V1–V6. Exemplo de correção típica: o Strategist citou ticket ou pedidos errados e o Critic recalculta o impacto no expected_result e no score_qualidade.

```

{
  "original_title": "Reposicionar ticket de R$ 85 para R$ 105...",
  "status": "improved",
  "verificacao_status": "DADO_CORRIGIDO",
  "score_qualidade": 9,
  "verificacoes": {
    "V1_numeros": {"resultado": "corrigido", "detalhe": "Ticket médio confirmado: R$ 85 (não R$ 92)" },
    "V5_impacto": {"resultado": "corrigido", "detalhe": "240 pedidos/mês × +R$20 = +R$4.800/mês" }
  },
  "final": {
    "expected_impact": "high",
    "category": "market",
    "title": "Reposicionar ticket de R$ 85 para R$ 105 (+24%) com kits",
    "problem": "...",
    "action": "...",
    "expected_result": "+R$ 4.800/mês (240 pedidos × +R$ 20)",
    "data_source": "orders_summary + benchmarks"
  }
}

```

10.1.5) O que é salvo e aparece na UI

Ao final, o backend salva analysis.summary (incluindo premium_summary) e salva 9 registros em suggestions. A UI mostra:

- Health Score + insight (do Analyst);
- Resumo Estratégico (premium_summary);
- 9 sugestões (com passos em recommended_action).

10.2) Exemplo LITE (7 dias) — diagnóstico compacto

10.2.1) Entrada compactada

```
{
  "period_days": 7,
  "orders": {"total": 40, "total_revenue": 3200, "average_order_value": 80,
  "cancellation_rate": 2.5},
  "products": {"total": 250, "active": 210, "out_of_stock": 30, "low_stock": 15,
  "best_sellers_count": 20},
  "coupons": {"usage_rate": 20, "ticket_impact": -3, "total_discount": 180}
}
```

10.2.2) Lite Analyst: métricas + anomalias + overall_health

```
{
  "metrics": {
    "sales": {"total": 3200, "daily_average": 457, "trend": "estável"},
    "average_order_value": {"value": 80, "benchmark": 150},
    "cancellation_rate": 2.5,
    "inventory": {"out_of_stock_products": 30, "critical_stock_products": 15},
    "coupons": {"usage_rate": 20, "ticket_impact": -3}
  },
  "anomalies": [
    {"type": "estoque", "description": "30 produtos sem estoque e 15 em estoque crítico",
    "severity": "alto"},
    {"type": "ticket", "description": "Ticket médio 47% abaixo do benchmark (R$80 vs R$150)",
    "severity": "médio"}
  ],
  "overall_health": {"score": 55, "classification": "atenção", "main_points": ["Estoque crítico e ticket abaixo do benchmark"]}
}
```

10.2.3) Lite Strategist: 6 sugestões (2-2-2)

```
{
  "suggestions": [
    {
      "expected_impact": "high",
      "category": "inventory",
      "title": "Repor 5 best-sellers esgotados que representavam R$ 1.200/semana",
      "description": "...",
      "recommended_action": "1. Identificar SKUs\n2. Repor urgente\n3. Ativar avise-me",
      "data_justification": "products.out_of_stock + histórico de pedidos"
    }
  ]
}
```

No LITE não há Critic nem Premium Summary. Após validação mínima, as sugestões passam por dedup contra histórico e são persistidas.

11) Apêndice (glossário, listas e referências)

11.1) Glossário

Termo	Significado no EcommPilot
FULL	Pipeline completo (9 etapas) com dados externos, RAG, Strategist + Critic e filtros finais.
LITE	Pipeline compacto (2 etapas) com janela menor (7 dias), 6 sugestões e sem Premium Summary.
Health Score	Score 0–100 calculado pelo Analyst (FULL) ou Lite Analyst (LITE) segundo regras do prompt.
Alertas	Mensagens curtas priorizadas (até 3 no FULL) para destacar risco imediato (danger/warning).
Oportunidades	Leituras de potencial (ex.: bundles, retenção, pricing) extraídas do diagnóstico do Analyst (FULL).
RAG	Busca de benchmarks/estratégias em base interna (texto/embeddings) para contextualizar recomendações.
Embeddings / pgvector	Representação vetorial para busca semântica e filtro de similaridade (threshold típico 0.85).
Saturação de tema	Detecção de repetição por palavras-chave (ThemeKeywords). 3+ ocorrências bloqueiam programaticamente.

11.2) Enumerações relevantes (API/UI)

11.2.1) Status de Analysis

- Pending, Processing, Completed, Failed

11.2.2) Status de Suggestion

- **Tela de análise:** new, rejected (aliases: pending, ignored)
- **Acompanhamento:** accepted, in_progress, completed

11.2.3) Impacto / prioridade

- high, medium, low
- Na API, SuggestionResource.priority é um alias de expected_impact para compatibilidade com o Frontend.

11.3) Categorias aceitas nas sugestões (Strategist/Critic)

O ecossistema usa categorias para organizar recomendações. No Strategist/Critic V7, HIGH devem ser estratégicas e MEDIUM/LOW tendem a ser táticas.

Grupo	Categorias	Interpretação
Estratégicas (HIGH)	strategy, investment, market, growth, financial, positioning	Decisões de negócio: metas, investimento, mercado, posicionamento, alavancas financeiras.
Táticas (MEDIUM/LOW)	inventory, pricing, product, customer, conversion, marketing, coupon, operational	Execução: estoque, páginas, checkout, cupons, aquisição, atendimento e operações.

11.4) Temas e palavras-chave (ThemeKeywords)

A saturação por tema usa palavras-chave em título + descrição para contar repetição e bloquear temas com 3+ ocorrências. Lista atual (centralizada em ThemeKeywords.php):

Tema	Palavras-chave
Quiz de Recomendação	quiz, questionário, personalizado, recomendação, personalização
Frete Grátis	frete grátis, frete gratuito, entrega grátis, frete gratis
Programa de Fidelidade	fidelidade, pontos, cashback, loyalty, recompensa
Kits e Combos	kit, combo, bundle, cronograma, pack
Gestão de Estoque	estoque, avise-me, reposição, inventário, ruptura
Email Marketing	email, e-mail, newsletter, automação, pós-venda
Conteúdo em Vídeo	vídeo, video, tutorial, youtube, reels
Modelo de Assinatura	assinatura, recorrência, subscription, clube
Cupons e Descontos	cupom, desconto, promoção, voucher, código
Otimização de Checkout	checkout, carrinho, conversão, abandono, finalização
Atendimento via WhatsApp	whatsapp, telegram, chat, mensagem, zap
Reviews e UGC	review, ugc, avaliação, depoimento, fotos, vídeos, antes e depois
Pós-Compra	pós-compra, pos-compra, follow-up, acompanhamento
Marketing de Influenciadores	influenciador, micro-influenciador, embaixador, embaixadora, parceria, afiliado

Tema	Palavras-chave
Gamificação	gamificação, gamificacao, desafio, milhas, níveis
Hub de Conteúdo	conteúdo, conteudo, hub, guia, educativo
Campanha de Carnaval	carnaval, folia, fantasia, bloco
Aumento de Ticket Médio	ticket médio, ticket, aov, valor médio
Redução de Cancelamento	cancelamento, cancelado, desistência, churn
Reativação de Clientes	reativação, reativar, inativos, dormentes, win-back
Upsell	upsell, up-sell, upgrade, premium
Cross-sell	cross-sell, cross sell, venda cruzada, produtos relacionados, compre junto
Estratégia de Precificação	preço, pricing, margem, precificação
SEO	seo, google, busca, orgânico, ranqueamento
Remarketing	remarketing, retargeting, pixel, público similar

11.5) Chaves/flags que costumam alterar comportamento

- `analysis.v2.use_history_summary`: usa resumo otimizado de histórico (reduz tokens) quando habilitado.
- `ai.embeddings.provider`: habilita embeddings para RAG e filtro semântico de similaridade (requer pgvector).