Capítulo 04 - Regras de Negócio

4.1 Visão Geral

Este capítulo documenta as decisões algorítmicas e regras de validação que governam o comportamento interno do sistema E.Rastreio Portal. As regras estão organizadas por domínio e foram extraídas diretamente do código-fonte (`app/Domains/*/` e `app/Models/`).

4.2 Módulo: Jornada de Trabalho

4.2.1 Detecção de Movimento e Parada

O sistema determina se um veículo está em movimento ou parado com base em dados brutos de posição GPS (tabela `tc_positions` do Traccar). O serviço central é o `VehicleMovementService`, que implementa a seguinte lógica:

- **Limiar de velocidade padrão:** velocidade < **5 km/h** classifica o veículo como parado. A velocidade é convertida de nós para km/h (fator: × 1.852).
- **Parada mínima válida:** um intervalo de parada só é reconhecido se tiver duração ≥ **5 minutos** contínuos.
- **Paradas dentro de raio geográfico (carga/descarga):** quando configurado um raio (`radiusKm`), mesmo que o veículo supere o limiar de velocidade, ele ainda é considerado "parado" enquanto permanece dentro da distância do ponto de referência. Isso modela cenários de carga e descarga.
- **Intervalo de posição inválido:** posições com intervalo superior a 600 segundos entre si são descartadas e reiniciam o acumulador de parada.

Esses parâmetros (`speed_threshold_kmh`, `min_stop_minutes`, `min_driving_seconds`) podem ser sobrescritos por veículo por meio da tabela `veiculo_parametros`, permitindo calibração individual por frota.

4.2.2 Derivação dos Períodos de Direção (Ponto Automático)

Os períodos de direção são derivados como os **gaps entre as paradas válidas**, delimitados pela primeira e última posição GPS do dia. O algoritmo (`PontoAutomaticoService::getDrivingPeriodsFromPositions`) funciona da seguinte forma:

1. Busca todas as posições do device no dia (00:00–23:59:59).
2. Chama o `VehicleMovementService` para obter os intervalos de parada.
3. A janela entre o fim de uma parada e o início da próxima é classificada como **período de direção**, desde que sua duração seja ≥ `min_driving_seconds` (padrão: 60 s).
4. Um período de direção final é adicionado se ainda houver tempo entre a última parada e a última posição do dia.

Se não houver posições para o dia, o registro de ponto é criado com todos os tempos zerados.

4.2.3 Classificação dos Tempos da Jornada

O `DriverTimesheetService` possui uma lógica alternativa (modo legado) para segmentação de posições brutas, com as seguintes constantes baseadas na **Lei 13.103/2015 (Lei do Motorista)**:

Parâmetro	Valor
Pausa mínima (intervalo)	15 minutos (900 s)
Refeição mínima	1 hora (3.600 s)
Descanso mínimo	8 horas (28.800 s)
Tempo máximo de direção	8 horas (28.800 s)
Limiar de velocidade (movimento)	> 1 km/h
Viagem mínima	5 minutos / 500 m

As paradas são classificadas hierarquicamente:

- Duração ≥ 28.800 s → **Descanso**
- Duração ≥ 3.600 s → **Refeição/Almoço**
- Duração ≥ 900 s → **Pausa**
- Abaixo disso → descartada

4.2.4 Cálculo do Tempo de Descanso (Ponto Automático)

No fluxo principal (automático), o tempo de descanso é calculado por subtração:

```
descanso = 24 horas − tempo_total_de_direção − 1 hora (almoço fixo)
```

O resultado nunca é negativo (`max(0, descanso)`). O almoço de 1 hora é tratado como fixo e não deduzido do cálculo de horas extras.

4.2.5 Cálculo de Horas Noturnas

Horas noturnas são contabilizadas para os períodos de **direção** que ocorram dentro das faixas:

- Madrugada: **00:00 – 05:00**
- Noite: **22:00 – 23:59:59**

O sistema calcula a interseção exata (em segundos) entre cada período de direção e as faixas noturnas.

4.2.6 Cálculo de Horas Extras

As regras seguem a legislação trabalhista brasileira para motoristas profissionais:

- **Jornada regular:** configurável por veículo (`horas_regulares`); padrão = **8 horas**.
- **Hora extra 50%:** aplicável apenas em dias **úteis**. Todo tempo de direção que exceda a jornada regular é computado como extra 50%.
- **Hora extra 100%:** aplicável quando o dia for **domingo** ou **feriado nacional**. Nesse caso, todo tempo de direção acima da jornada regular é computado como extra 100%.
- Em dias úteis, **não há extra 100%** pelo simples fato de ultrapassar certo limite — todo o excesso é 50%.
- Se o veículo trabalhou menos que a jornada regular, os campos de hora extra são zerados.

4.2.7 Detecção de Feriados

O `HolidayService` consulta a **BrasilAPI** (`brasilapi.com.br/api/feriados/v1/{ano}`) para obter os feriados nacionais do ano em questão. O resultado é cacheado por **24 horas** para evitar chamadas repetidas. Em caso de falha da API, o sistema usa um fallback com os 9 feriados nacionais fixos (ex.: Ano Novo, Tiradentes, Dia do Trabalho etc.).

4.2.8 Vínculo Motorista–Veículo por Data

A identidade do motorista em um registro de ponto é resolvida consultando a tabela `veiculo_motorista_historico`, que registra períodos de vigência do vínculo (`data_inicio`, `data_fim`). O sistema localiza o motorista cuja janela de vigência contém a data do ponto. Se houver vários vínculos, o mais recente é selecionado. O vínculo pode estar aberto (`data_fim IS NULL`), indicando que ainda está em vigor.

4.2.9 Vínculo Device–Veículo por Data

Para processamento retroativo ou por motorista, o sistema resolve qual rastreador (`id_rastreador`) estava associado ao veículo em cada data específica, consultando `rastreador_historico` com o filtro de período de vigência (`is_principal = true`).

4.2.10 Edição Manual de Registros de Ponto

Registros de ponto podem ser editados manualmente. As regras aplicadas no momento da edição são:

- **Validação de formato:** horário de início e fim devem estar no formato `H:i:s`.
- **Período noturno:** se o horário de fim for anterior ao de início, adiciona-se 1 dia ao fim (ex.: turno 23:00–02:00).
- **Tipo de dia:** o dia pode ser marcado como **DSR** ou **Férias**, zerando todos os períodos de direção e atribuindo 24 horas de descanso.
- Após qualquer edição, o sistema **recalcula automaticamente** todos os totais: tempo de direção, descanso, horas noturnas, horas extras 50% e 100%, com respeito ao flag `is_holiday` que pode ser marcado manualmente para dias de feriado não reconhecido pela API.
- O campo `notes.manually_edited = true` é gravado para rastreabilidade.

4.3 Módulo: Telemetria e Score de Condução

4.3.1 Detecção de CAN Bus

Antes de calcular qualquer score, o sistema detecta se o rastreador do veículo transmite dados de CAN Bus (barramento do veículo). Isso é feito amostrando até 50 posições recentes: se ao menos **10%** possuírem os atributos `rpm`, `throttle` ou `fuelConsumption`, o veículo é marcado como `tem_can = true`. A presença de CAN desbloqueia indicadores adicionais de pontuação.

4.3.2 Resolução do Device para Telemetria

Para veículos com múltiplos rastreadores ativos, o sistema prioriza a seguinte hierarquia ao selecionar o device para telemetria:

1. Rastreador marcado com `is_telemetria = true`
2. Rastreador com `is_principal = true`
3. Qualquer rastreador ativo

4.3.3 Cálculo de Métricas Brutas (por Posição)

O `TelemetriaScoreService` processa cada par de posições consecutivas e acumula métricas. As regras de filtragem são:

- Pares com intervalo **≤ 0 s ou > 600 s** são descartados (interrupção de sinal).
- **Motor ligado:** definido como `velocidade ≥ 2 km/h` (movimento estrito), OU para veículos com CAN: `rpm > 0`, OU sem CAN: flag `ignition = true`.
- **Idle (motor ocioso):** motor ligado, mas parado por **mais de 120 segundos** consecutivos. Paradas curtas (≤ 120 s, ex.: semáforo) **não contam como idle** — esse comportamento replica o cálculo do GOBRAX.
- **Excesso de velocidade:** posição com velocidade acima do `limite_velocidade` configurado no veículo (padrão: 90 km/h).
- **Aceleração/frenagem brusca:** calculada pela variação de velocidade entre duas posições dividida pelo intervalo. Limiar: > **3,0 m/s²** (positivo = aceleração, negativo = frenagem).
- **Contagem de viagens:** incrementada a cada transição `ignition: true → false`.

4.3.4 Métricas CAN-Exclusivas

Quando `tem_can = true`, são calculadas métricas adicionais por faixa:

Faixas de RPM (modelo GOBRAX):

Faixa	Limiar (configurável no `ScoreProfile`)	Classificação
Início (ideal)	≤ `rpm_inicio_max` (padrão: 1.300 RPM)	Positivo
Final (bom)	1.301 – `rpm_final_max` (padrão: 1.500 RPM)	Positivo parcial
Amarelo	1.501 – `rpm_freio_motor_max` (2.300), acelerando (throttle ≥ 5%)	Negativo
Freio motor (positivo)	1.501 – 2.300, pé fora (throttle < 5%) + combustível < 0,1	Positivo (corte de injeção real)
Vermelho	> 2.300 RPM	Negativo forte

Faixas de Pressão do Acelerador (throttle):

Faixa	Limiar	Classificação
Ideal	≤ `throttle_ideal_max` (padrão: 60%)	Positivo
Atenção	61% – `throttle_atencao_max` (padrão: 70%)	Penalidade leve
Crítico	> 70%	Penalidade forte

O sistema usa o **pico** (máximo entre as duas amostras do intervalo) para o throttle, a fim de capturar picos que caem nas bordas da amostragem.

Embalo (aproveitamento de inércia): veículo em movimento (velocidade ≥ 2 km/h), pé fora do acelerador (throttle < 5%) e combustível < 3 L/h e RPM > 300. Sinaliza aproveitamento da inércia do veículo.

Piloto automático (estimado): pé fora do pedal (throttle < 5%) + motor injetando (combustível ≥ 3 L/h) + velocidade ≥ 20 km/h. Proxy para uso de cruise control, que o rastreador não reporta diretamente.

4.3.5 Cálculo de Distância (km/dia)

Para veículos **com CAN**, o sistema prefere o odômetro do veículo (`obdOdometer` ou `odometer`, em metros) ao invés do cálculo GPS Haversine. Regras de sanidade aplicadas ao delta do odômetro:

- Delta > **2.000 km/dia** → descartado (considerado dado inválido).
- Delta > **3× o km calculado por GPS** (quando GPS > 5 km) → descartado (possível reset ou spike).

Para veículos **sem CAN**, ou quando o odômetro não passar na sanidade, o km é calculado pela fórmula **Haversine** entre posições consecutivas. Segmentos com velocidade implícita > **150 km/h** são descartados como saltos de GPS.

4.3.6 Cálculo de km/L (Eficiência de Combustível)

A eficiência é calculada usando o campo `fuelUsed` (acumulado pelo CAN). O sistema captura o **delta** entre o primeiro e o último valor não-zero do dia. Valores acima de **1.000 L/dia** são descartados como inválidos (unidade errada ou reset do dispositivo).

4.3.7 Normalização das Notas por Indicador (0–100)

Cada indicador é normalizado para uma nota de 0 a 100:

Indicador	Fórmula resumida
Idle	100 até ~3% de idle/ignição, caindo linearmente a 0 próximo de 40%
Excesso	100 sem excesso; cada ponto percentual de tempo em excesso desconta 10 pts

Indicador	Fórmula resumida
Aceleração brusca	100 − (eventos/100 km) × 5
Freada brusca	100 − (eventos/100 km) × 5
RPM	Ponderado: RPM_início × 1,0 + RPM_final × 0,7 + Freio_motor × 1,0, dividido pelo tempo em movimento
Throttle	100 − (crítico% × 300) − (atenção% × 100)
Embalo	Linear; 100 pts a partir de ~20% do tempo em movimento em embalo
Piloto automático	0 abaixo de 50% do tempo em movimento, 100 pts a 85% (linear)
km/L	(km_L_real / meta_km_L) × 100, máximo 100

4.3.8 Score Final

O score final é a **média ponderada** das notas pelos pesos configurados no `ScoreProfile` do grupo do veículo. Quando não há CAN, apenas os indicadores GPS (idle, excesso, aceleração, freada) são usados; seus pesos são **renormalizados** para somar 100%, descartando os indicadores CAN.

O score **do período** (ex.: mensal) é calculado de forma **agregada** — somando os tempos/eventos e pontuando uma vez sobre os totais, não pela média das notas diárias. Isso replica o comportamento do GOBRAX e evita distorções em dias com poucos km.

4.3.9 Cálculo de Desperdício e Economia de Combustível

Baseados na meta de eficiência (`meta_kmL`) do `ScoreProfile` e no preço do diesel configurado:

```
desperdício_litros = max(0, litros_consumidos − (km_total / meta_kmL))
desperdício_R$ = desperdício_litros × preco_diesel

economia_litros = max(0, (km_total / meta_kmL) − litros_consumidos)
economia_R$ = economia_litros × preco_diesel
```

4.3.10 Premiação por Desempenho

O `PremiacaoService` calcula o valor de premiação de um motorista com o seguinte algoritmo:

1. **Gate de km mínimo:** se o motorista rodou menos que `km_minimo_mes`, recebe R$ 0,00.

2. **Fator de nota:** proporcional ao intervalo entre `nota_minima` e 100. Abaixo da nota mínima, o valor base é zero.
3. **Fator de km:** `min(1,0, km_rodado / meta_km_mes)`. Penaliza quem não atingiu a meta de quilometragem.
4. **Base = premiacao_max × fator_nota × fator_km**
5. **Premiação extra (opcional):**
   - Ativada pelo flag `extra_ativo`.
   - Calculada sobre os km acima de `extra_km_minimo`: `(km_elegível / extra_km_considerado) × extra_valor_por_km`.
   - Se `extra_atrelar_nota = true`, o valor extra é multiplicado por `nota/100`.
   - Se `extra_zerar_abaixo_range = true` e a nota está abaixo do mínimo, o valor total (base + extra) é zerado.

4.4 Módulo: Rastreadores e Monitoramento em Tempo Real

4.4.1 Classificação de Status do Veículo

O `VehicleTrackingService` classifica cada veículo em três estados, usando a última posição GPS recebida:

Status	Condição
em_movimento	Velocidade > 5 km/h **E** última posição há ≤ 5 horas
parado	Velocidade ≤ 5 km/h **E** última posição há ≤ 5 horas
sem_sinal	Última posição há > **5 horas** (independente da velocidade)

4.4.2 Relatório Diário de Utilização

O `VehicleUsageReportService` produz um resumo diário com km rodado, segundos de direção e velocidade média. A lógica de cálculo de tempo de direção é **idêntica** à do Ponto Automático (`getDrivingPeriodsFromPositions`), garantindo que os valores sejam consistentes entre os dois módulos.

A velocidade média é calculada como `km / (driving_seconds / 3600)`, sem contar os períodos de parada.

4.4.3 Conversão de Placa (Padrão Mercosul)

O modelo `Veiculo` implementa a conversão bidirecional entre o padrão antigo (ex.: `ABC1234`) e o Mercosul (ex.: `ABC1D34`), mapeando a 5ª posição: `A=0, B=1, C=2, ..., J=9`. As buscas por placa sempre consideram **ambas as variantes** para não perder registros.

4.5 Módulo: Acessos e Permissões

4.5.1 Hierarquia de Usuários

O sistema tem três perfis de acesso principais:

- **Master Admin (`is_master_admin = true`):** acesso irrestrito a todos os módulos, veículos e grupos. As verificações de permissão retornam `true` imediatamente para esse perfil, sem consultar o banco.
- **Analista (`is_analista = true`):** usuário operacional, associado a um gestor (`gestor_id`).
- **Usuário padrão:** acesso determinado pelas permissões de módulo vinculadas.

4.5.2 Controle de Acesso a Veículos

Um usuário não-admin enxerga apenas os veículos que pertençam a:

1. **Grupos** aos quais ele está vinculado (`user_grupo`), ou
2. **Veículos** diretamente vinculados a ele (`user_veiculo`).

A union dos dois conjuntos (sem duplicatas) forma o escopo de visibilidade do usuário.

4.5.3 Permissões por Módulo e Ação

Permissões são armazenadas em JSON na tabela pivot `user_module_permissions`. Cada registro contém um mapa de `{action_slug: true/false}`. A verificação de permissão (`hasPermission`) exige que o módulo esteja ativo (`is_active = true`) e que a ação específica esteja marcada como `true` no JSON.

4.5.4 Operação em Massa de Permissões

O `BulkPermissionService` aplica permissões a múltiplos usuários (até **500 por operação**, em lotes de 100). Regras de segurança:

- **Master admins são ignorados** automaticamente (nunca são alterados por operações em massa).
- **Auto-revogação bloqueada:** o sistema impede que um usuário revogue a própria permissão `admin.ver` de si mesmo.

- A operação é **não-destrutiva**: ao revogar, se após a remoção de uma ação ainda houver outras `true` no JSON, o registro é atualizado; caso contrário, é deletado integralmente.
- Toda a operação é executada dentro de uma **transação de banco de dados**.

As ações válidas são validadas contra o método `actionsAvailable()` do módulo. Slugs inválidos causam rejeição de toda a requisição.

4.5.5 Aplicação de Perfil de Acesso

Ao aplicar um `AccessProfile` a um usuário, o serviço `AccessProfileApplicationService`:

1. Verifica se o usuário é Master Admin; se for, **não faz nada** (proteção de integridade).
2. Propaga o flag `is_analista` do perfil para o usuário.
3. Vincula os módulos do perfil ao usuário com as permissões definidas no perfil (via `attach`).

4.6 Módulo: Produtividade e SLA de Analistas

4.6.1 Métricas de Output

O output de um analista é calculado contando os registros de `veiculo_contato` criados por ele no período, classificados por origem:

Tipo	Critério
Sinal	Registro vinculado a um `alerta_sinal_id`
Pernoite	Registro vinculado a um `alerta_pernoite_id`
Auditoria (veículo)	Registro vinculado a `auditoria_sistema_id`
Auditorias extras	Auditorias de categoria `grupo` ou `usuario` resolvidas pelo analista no período
Notas	Contatos sem vínculo com nenhum alerta ou auditoria

4.6.2 Apuração de SLA

O SLA é apurado por origem de alerta, usando as configurações da tabela `sla_configs` (resolvida via `SlaConfig::resolver(origem, subtipo)`):

- **Alerta de Sinal:** dois subtipos — `<24h` (alerta recente) e `>=24h` (crítico). O SLA é medido em minutos entre `created_at` e `resolvido_em`.
- **Alerta de Pernoite:** SLA único (`pernoite`), medido entre `created_at` e `resolvido_at`.
- **Auditoria:** SLA por `tipo_pendencia`. O prazo começa na `data_execucao` da auditoria (início do dia).

Um atendimento é classificado como **"no prazo"** se o tempo de resolução (em minutos) for ≤ `sla_minutos`. A **aderência** (%) é `no_prazo / (no_prazo + estourado) × 100`.

4.6.3 Alertas Críticos em Tempo Real

O sistema identifica atendimentos com risco de estouro de SLA combinando a configuração `sla_minutos` com o campo `warning_pct` (percentual de decurso do prazo a partir do qual o alerta entra na lista crítica). Um alerta é considerado crítico quando:

```
pct_decorrida = (minutos_desde_abertura / sla_minutos) × 100 ≥ warning_pct
```

4.6.4 Score de Qualidade

O score de qualidade de um analista é composto por três fatores ponderados:

Fator	Peso	Critério
Observações ricas	60%	% de resoluções com descrição ≥ **30 caracteres**
Taxa sem reabertura	30%	1 − (reaberturas / total de resoluções)
Taxa de resolução	10%	% de eventos do tipo `resolucao` entre todos os eventos

O score máximo é 100. **Reabertura** é contada quando uma auditoria que o analista havia resolvido voltou a aparecer como pendente para a mesma entidade + tipo.

4.7 Módulo: Cadastros

4.7.1 Auditoria Automática do Sistema

O `SystemAuditService` executa verificações diárias (`runFullAudit`) e registra pendências na tabela `auditoria_sistema`. As verificações são:

Grupos:

- Grupo sem `id_grupo_rastreadores` (ID eRastreio não configurado).
- Grupo sem `matricula_coocamais` (Matrícula Cooca+ não configurada).

Usuários (excluindo Master Admins):

- Usuário sem nenhum grupo vinculado.
- Usuário sem grupo **e** sem veículo vinculado.

Veículos (apenas status "Ativo"):

- Veículo sem grupo vinculado (`id_grupo` nulo).
- Veículo sem rastreador principal ativo (`is_principal = true` na tabela `rastreador_historico`).
- Veículo que passou pela **Região Metropolitana de São Paulo** nas últimas 24 horas **sem motorista vinculado**. A detecção é feita por busca textual no campo `address` das posições Traccar, cobrindo 15 municípios da Grande SP.

Idempotência: antes de apagar os registros do dia atual e regerá-los, o serviço preserva quais itens já foram marcados como `resolvido = true` e reaplicará esse status após a nova execução, evitando perda de trabalho do analista.

4.7.2 Consulta de Valor FIPE

O `FipeService` consulta o valor de mercado de veículos (categoria: caminhão) na tabela FIPE via API oficial. O resultado é cacheado por **24 horas** por código FIPE + ano. A resposta textual no formato `"R$ 99.332,90"` é convertida para `float` removendo símbolo, pontos de milhar e substituindo a vírgula decimal por ponto.

4.8 Módulo: Câmeras Embarcadas

4.8.1 Gestão de Gravações (Jimi IoT)

O `RecordingService` gerencia gravações de câmeras Jimi IoT. As regras de operação são:

- **Todos os horários** são enviados ao dispositivo em **UTC**, independente do fuso horário do servidor.
- **Modo demo:** se a variável `JIMI_DEMO_STREAM_URL` estiver configurada, o sistema retorna dados fictícios sem enviar comandos reais ao dispositivo. Isso permite demonstrações sem hardware.
- A listagem de gravações tolera variações no formato da resposta da API do dispositivo, buscando a lista nos campos `list`, `items`, `records`, `fileList`, `_list` ou `data`.

4.9 Módulo: Alertas de Pernoite

4.9.1 Ciclo de Estados do Alerta de Pernoite

Um alerta de pernoite percorre os seguintes estados:

```
avaliar → aprovado
        → reprovado
        → sem_sinal (veículo sem sinal no momento do pernoite)
```

O estado inicial é `avaliar`. A transição para `aprovado` ou `reprovado` registra `resolvido_por_user_id` e `resolvido_at`. Alertas no estado `avaliar` ou `sem_sinal` são listados na fila de trabalho do analista e entram no cálculo de SLA.

4.10 Módulo: Alertas de Sinal

4.10.1 Ciclo de Estados do Alerta de Sinal

```
aberto → resolvido
```

Alertas no estado `aberto` entram na fila de trabalho. A resolução registra `resolvido_por_user_id` e `resolvido_em`. O campo `horas_sem_sinal` determina o **subtipo de SLA** aplicável: `<24h` ou `>=24h`.

4.11 Módulo: Financeiro

4.11.1 Tipos de Receita e Validação Condicional

As receitas possuem dois tipos com conjuntos de campos distintos:

Tipo	Campos obrigatórios	Campos exclusivos
`frete`	`origem`, `destino`	`odometro_inicial`, `odometro_final`, `peso`, `tipo_carga`
`outro`	`descricao`	—

A validação é aplicada condicionalmente com base no `tipo`. Para o tipo `frete`, os campos de `origem` e `destino` são obrigatórios, enquanto para `outro` a `descricao` passa a ser obrigatória. Campos que não se aplicam ao tipo selecionado são **explicitamente zerados** antes de persistir.

4.11.2 Resolução Automática do Motorista

Tanto nas receitas quanto nas despesas, o motorista responsável é **determinado automaticamente** a partir do veículo selecionado. O sistema consulta a relação `motoristaAtual` do veículo (via `VeiculoMotoristaHistorico`) e atribui o `pessoa_id` resultante. Se nenhum veículo for selecionado, `motorista_id` é nulo. O usuário não informa o motorista manualmente.

4.11.3 Tratamento de Máscara Monetária

Todos os campos de valor monetário chegam no formato de exibição brasileiro (ex.: `R$ 1.234,56`). O sistema remove o símbolo `R$`, pontos de milhar e espaços, depois converte a vírgula decimal em ponto antes de persistir como decimal: `"R$ 1.234,56"` → `1234.56`.

4.11.4 Despesas de Abastecimento

Despesas categorizadas como **"Abastecida"** desbloqueiam campos extras: `produto` (tipo de combustível), `odometro` (leitura do hodômetro) e `litros` (volume abastecido, mínimo 0,01 L). Para qualquer outra categoria, esses três campos são **zerados automaticamente** antes de salvar, independente do que foi enviado no formulário.

4.11.5 Imutabilidade após Acerto

Receitas, despesas e transferências que já foram incluídas em um **Acerto** (`acerto_id IS NOT NULL`) tornam-se imutáveis. Qualquer tentativa de editar ou excluir esses registros é bloqueada antes de qualquer processamento de dados, com redirecionamento e mensagem de aviso. Esse é o mecanismo de integridade contábil do módulo.

4.11.6 Cálculo do Saldo do Acerto

O Acerto é a operação de fechamento financeiro de um motorista para um período. Ele agrega itens de três tabelas distintas e calcula o saldo pela seguinte fórmula:

```
saldo = total_receitas − total_despesas − total_adiantamentos + total_devolucoes
```

As **transferências de adiantamento** são tratadas como saída de caixa (equivalente a despesa), enquanto as **devoluções** são tratadas como entrada (equivalente a receita). Toda a criação do acerto ocorre dentro de uma **transação de banco de dados**, incluindo o vínculo dos itens (`acerto_id`) e o cálculo dos totais.

4.11.7 Vínculo de Transferências ao Motorista

As transferências são vinculadas a **veículos**, não diretamente a motoristas. Ao buscar as transferências elegíveis para um acerto, o sistema identifica todos os veículos que o motorista conduziu no período (via `veiculo_motorista_historico`) e então busca as transferências desses veículos que ainda não possuem `acerto_id`.

4.11.8 Exclusão de Acerto (Liberação dos Itens)

A exclusão de um acerto utiliza soft-delete, mas antes de deletar, todos os itens vinculados (receitas, despesas e transferências) têm seu `acerto_id` zerado, ficando disponíveis para um novo fechamento. Todo o processo ocorre dentro de uma transação.

4.11.9 Dashboard Financeiro

O dashboard agrega os dados do mês selecionado (padrão: mês atual). Para o gráfico dos últimos 6 meses, as **devoluções são somadas às receitas** e os **adiantamentos são somados às despesas**, tratando transferências como equivalentes de seus respectivos fluxos financeiros.

4.12 Módulo: Portal de Compras (Cooca+ / SigaMC)

4.12.1 Visão Geral

Este módulo é um **painel de consulta** a dois sistemas externos de cooperativa. Não há lógica de negócio local além de filtragem e orquestração de chamadas às APIs. Os dados exibidos são inteiramente provenientes das APIs de terceiros e não são persistidos no banco do portal.

4.12.2 Escopo de Consulta por Usuário

A consulta à Cooca+ é feita pelos **grupos** do usuário logado. O sistema filtra apenas os grupos que possuem o campo `matricula_coocamais` preenchido e realiza uma consulta por matrícula para cada um deles. Grupos sem matrícula configurada são silenciosamente ignorados.

Exceção Master Admin: quando o usuário é Master Admin, é possível selecionar qualquer usuário do sistema e visualizar os dados dos grupos desse usuário-alvo, não do admin logado.

4.12.3 Operações Disponíveis por API

Operação	API	Parâmetro-chave	Finalidade
Limites do cooperado	Cooca+	`userId` (código do cooperado)	Consulta saldo e limites disponíveis
Reservas / Pedidos	Cooca+	`userId`	Lista pedidos em aberto (tipo 2, status 0)
Títulos a pagar	SigaMC	`matricula`	Consulta financeiro do cooperado na SigaMC

Ambas as APIs usam autenticação configurada via variáveis de ambiente (`services.coocamais.*` e `services.sigamc.*`). Falhas de comunicação são logadas e retornam um array com a chave `error` — o sistema não lança exceções para o usuário.

4.13 Módulo: Espelhamento (Compartilhamento Público de Frota)

4.13.1 Ciclo de Vida de um Espelhamento

Um espelhamento é um link público e temporário que expõe a posição em tempo real de um subconjunto de veículos sem exigir autenticação do visualizador. Seu ciclo de estados é:

```
Ativo (is_active=true + expires_at no futuro)
  → Expirado (is_active=true + expires_at no passado)
  → Desativado (is_active=false)
```

- A **exclusão** via interface é uma desativação lógica (`is_active = false`), não uma deleção física.
- Um espelhamento **expirado não pode ser reativado** diretamente; é necessário editar a data de validade antes de reativar.
- Cada acesso público incrementa o contador `access_count` e atualiza `last_accessed_at`.

4.13.2 Geração e Segurança do Token

O token de acesso público é uma string aleatória de **48 caracteres** (`Str::random(48)`), gerada no momento da criação. Não há rotação automática de token — ele é permanente para o ciclo de vida do espelhamento.

4.13.3 Resolução do Token

Em todos os endpoints públicos, o token é validado contra três condições simultâneas:

1. `token` existe na tabela.
2. `is_active = true`.
3. `expires_at > now()`.

Se qualquer condição falhar, a resposta é HTTP 410 (Gone) com mensagem de link inválido ou expirado.

4.13.4 Dados Expostos Publicamente

O endpoint público de veículos **sanitiza** os dados antes de retorná-los, expondo apenas: placa, marca, modelo, cor, ano, status de movimento, velocidade, latitude, longitude, nome do motorista, endereço e tecnologia do rastreador. Dados sensíveis de negócio (ID do rastreador, IMEI, ESN, linha telefônica, grupo, associação) são omitidos da resposta pública.

Veículos sem coordenadas GPS válidas são **filtrados** e não aparecem no mapa.

4.13.5 Funcionalidade de Geofence no Espelhamento

O criador do espelhamento pode habilitar geofences (áreas de alerta geográfico) para o link público. As regras são:

Autenticação de geofence:

- Protegida por uma senha numérica de **6 dígitos**, gerada aleatoriamente e armazenada com hash bcrypt. A senha em texto plano também é persistida para exibição ao criador.
- A sessão de autenticação tem **TTL de 30 minutos**. Após esse prazo, o visitante precisa autenticar novamente para editar geofences.
- Visitantes não autenticados podem **visualizar** as geofences, mas o e-mail de notificação é **mascarado** (ex.: `j***@empresa.com.br`).

Limites e validações:

- Máximo de **20 geofences ativas** por espelhamento. Tentativas de criação além desse limite são rejeitadas com HTTP 422.
- Um geofence só pode ser criado para veículos que façam parte do mesmo espelhamento. Tentativas com veículos externos são rejeitadas.
- Ao editar um geofence, o estado `currently_inside` é **resetado para** `false`, evitando estados inconsistentes após alteração de coordenadas ou raio.

Cálculo de contenção: O sistema utiliza a fórmula **Haversine** (raio da Terra = 6.371.000 m) para calcular a distância entre a última posição do veículo e o centro da geofence. Um ponto é

considerado "dentro" se a distância calculada for ≤ `raio_metros` configurado.

Resumo dos Domínios

Domínio	Status de Implementação	Nível de Lógica Interna
Jornada de Trabalho	Implementado e maduro	Alto — cálculos legislativos complexos
Telemetria / Score	Implementado e maduro	Alto — engine de scoring configurável
Rastreadores	Implementado	Médio — detecção de movimento e status
Acessos	Implementado	Médio — permissões granulares
Produtividade / SLA	Implementado	Médio — KPIs e SLA de analistas
Cadastros / Auditoria	Implementado	Médio — auditoria automática diária
Financeiro	Implementado (sem service layer)	Médio — ciclo de acerto bem definido
Portal Cooca+ / SigaMC	Implementado (gateway puro)	Baixo — proxy de APIs externas, sem persistência
Espelhamento	Implementado	Médio — ciclo de vida de link, geofence com auth própria
Câmeras	Implementado	Baixo–Médio — orquestração de hardware Jimi IoT

Revisão #2
Criado 2026-07-24 16:32:22 UTC por Nader
Atualizado: 2026-07-24 17:26:43 UTC por Nader