ChatGPT

# Relatório de inconsistências e plano de correção – Dashboard de Manutenção

## 1. Visão geral do ambiente

O projeto **Dashboard de Manutenção** é composto por um *frontend* (React/Vite) e um *backend* (FastAPI) rodando localmente. O *frontend* está em `apps/manutencao/frontend` e consome dados do backend via serviço configurado no arquivo `maintenance-api.ts`. A API do backend expõe as rotas sob o prefixo `/api/v1/manutencao` e, por padrão, escuta na porta **8010**. O arquivo `maintenance-api.ts` usa `VITE_API_BASE_URL` para compor o endereço da API; se a variável não existir, usa o fallback `http://127.0.0.1:8011/api/v1` [1], o que indica um **possível desvio da porta padrão (8010)**.

O mecanismo de atualização automática do dashboard é controlado por variáveis de ambiente definidas no `.env` do frontend:

- `VITE_REALTIME_POLL_INTERVAL_MS` ou `VITE_REALTIME_POLL_INTERVAL_SEC` definem o intervalo de consulta (polling). O código converte a unidade de segundos para milissegundos e usa esse valor para agendar recargas periódicas dos dados (ver linhas 232–241 de `MaintenanceDashboard.tsx` [2]). A ausência ou o uso incorreto dessas variáveis impede o refresh constante.
- `VITE_CATEGORY_CAROUSEL_INTERVAL_MS` / `VITE_CATEGORY_CAROUSEL_INTERVAL_SEC` definem somente a velocidade do carrossel de categorias e **não influenciam** a frequência de requisições ao backend [3].

No backend, as variáveis importantes são:

- `API_URL`, `GLPI_APP_TOKEN` e `GLPI_USER_TOKEN`, usadas para autenticação com o GLPI.
- `CACHE_TTL_SEC`, definindo o tempo de vida dos dados em cache. O valor demasiado baixo faz com que a API consulte o GLPI muito frequentemente; demasiado alto provoca dados defasados.
- `SESSION_TTL_SEC`, tempo de vida da sessão com o GLPI; sessões muito curtas forçam reautenticação.

## 2. Rotas e parâmetros expostos pelo backend

Os arquivos de rota do backend (`maintenance_stats_router.py` e demais) deixam claro quais endpoints e parâmetros são aceitos. O resumo abaixo utiliza o código e o serviço do frontend para inferir as rotas e seus requisitos:

| Endpoint (Manutenção) | Parâmetros | Evidência ou origem | Observações |
|---|---|---|---|
| **/api/v1/manutencao/status-totais** | Nenhum | Função `get_status_totais()` no backend gera `MaintenanceStatusTotals` [4] | Retorna chaves `novos`, `nao_solucionados`, `planejados`, `solucionados`, `fechados` e `resolvidos`. A chamada no frontend deveria usar `fetchMaintenanceStatusTotals`, mas o dashboard atual não consome esse endpoint. |
| **/api/v1/manutencao/stats-gerais** | `inicio`, `fim` (formato `YYYY-MM-DD`) | Função `get_maintenance_general_stats()` recebe `inicio` e `fim` [5] | Retorna `novos`, `pendentes`, `planejados`, `resolvidos` [6]. A ausência de parâmetros resulta em erro HTTP 422. |
| **/api/v1/manutencao/ranking-entidades** | `inicio`, `fim`, `top` | Função `fetchEntityRanking()` monta a query string com `inicio`, `fim` e `top` [7] | Se omitidos, pode retornar todos os registros ou usar valores padrão; o código envia o `top` configurado e os períodos em vigor. |
| **/api/v1/manutencao/ranking-categorias** | `inicio`, `fim`, `top` | Função `fetchCategoryRanking()` usa as mesmas regras do ranking de entidades [8] | O frontend triplica o `top` para garantir itens suficientes nos três grupos de categorias [9]. |
| **/api/v1/manutencao/top-atribuicao-entidades** | `top` | `fetchTopEntityAttribution()` monta a URL com apenas `top` [10] | Este endpoint não é consumido no dashboard atual. |
| **/api/v1/manutencao/top-atribuicao-categorias** | `top` | `fetchTopCategoryAttribution()` monta a URL com apenas `top` [11] | Também não usado no dashboard. |
| **/api/v1/manutencao/tickets-novos** | `limit` | `fetchMaintenanceNewTickets()` define `limit` (padrão 10) [12] | Retorna lista de tickets novos. |

### Mapeamento DTIC ↔ Manutenção

| Rotas DTIC | Rotas Manutenção | Comentário |
|---|---|---|
| `/api/v1/dtic/metrics-gerais` | `/api/v1/manutencao/stats-gerais` | Ambos fornecem estatísticas gerais; a rota de Manutenção exige `inicio` e `fim`. |
| `/api/v1/dtic/status-niveis` | `/api/v1/manutencao/status-totais` | A rota de Manutenção acrescenta chaves extras (`fechados`, `solucionados`). |

| Rotas DTIC | Rotas Manutenção | Comentário |
|---|---|---|
| `/api/v1/dtic/ranking-tecnicos` | `/api/v1/manutencao/ranking-entidades` e `/ranking-categorias` | DTIC junta técnicos e categorias numa rota; Manutenção separa rankings por entidades e por categorias. |
| `/api/v1/dtic/tickets-novos` | `/api/v1/manutencao/tickets-novos` | Comportamento semelhante. |

## 3. Inconsistências identificadas

1. **Porta da API configurada no frontend** – A função `fetchFromAPI` usa `VITE_API_BASE_URL` ou fallback `http://127.0.0.1:8011/api/v1` [13]. Entretanto, o backend de Manutenção foi documentado para rodar na porta **8010**, causando falhas de comunicação caso `VITE_API_BASE_URL` aponte para 8010 e o fallback permaneça em 8011. Recomenda-se alinhar a porta ou configurar explicitamente `VITE_API_BASE_URL=http://127.0.0.1:8010/api/v1` no `.env` do frontend.

2. **Variáveis de polling** – O frontend aceita **duas** variáveis para definir o intervalo de atualização: `VITE_REALTIME_POLL_INTERVAL_MS` (milissegundos) e `VITE_REALTIME_POLL_INTERVAL_SEC` (segundos) [2]. Caso ambas estejam definidas ou se um valor em milissegundos for atribuído à variável em segundos, o resultado será um intervalo incorreto. Para obter refresh a cada 12 s, configure **apenas uma** das variáveis: `VITE_REALTIME_POLL_INTERVAL_SEC=12` ou `VITE_REALTIME_POLL_INTERVAL_MS=12000`.

3. **Obrigatoriedade dos parâmetros** `inicio` **e** `fim` – A rota `/stats-gerais` no backend exige estes parâmetros; sem eles o FastAPI retorna HTTP 422. O frontend, entretanto, define valores padrão apenas se a URL não contiver parâmetros [14]; se o usuário abrir o dashboard sem query string, os valores padrão (últimos 30 dias) são aplicados. Isto funciona, mas qualquer outra ferramenta que consuma a API precisa garantir o envio de `inicio` e `fim`.

4. **Não utilização de** `status-totais` **no dashboard** – Embora o endpoint `/status-totais` retorne diversos totais (incluindo `em_atendimento` / `solucionados` e `fechados`), o frontend usa apenas `stats-gerais` para compor os quatro cartões de métricas (“Novos”, “Pendentes”, “Planejados”, “Resolvidos”) [6]. Isso causa ausência dos cartões “Em atendimento”/“Em progresso” e “Fechados” no dashboard e pode explicar divergências entre o comportamento esperado pelo DTIC (que exibe estes status) e o de Manutenção.

5. **Cache TTL e polling** – O backend usa `CACHE_TTL_SEC` para definir o tempo de vida dos dados em cache. Caso `CACHE_TTL_SEC` seja menor que o intervalo de polling do frontend (12 s), a API gerará carga excessiva sobre o GLPI; se for muito maior (ex.: 600 s), o dashboard exibirá dados desatualizados. A documentação recomenda valores entre 60 e 180 s para manter equilíbrio; contudo, o valor real deve ser verificado no `.env` do backend.

6. **Sessão GLPI** – A variável `SESSION_TTL_SEC` controla a duração da sessão com o GLPI. Valores muito baixos (<300 s) causam reautenticações frequentes, o que pode impactar o tempo de resposta e gerar erros intermitentes. É sugerido configurar **600 s** para Manutenção, em linha com a recomendação do DTIC.

7. **Nomenclatura de rotas** – O DTIC utiliza rotas com nomes diferentes (`metrics-gerais`, `status-niveis`, `ranking-tecnicos`). Para viabilizar uma migração sem alteração de código do frontend, seria necessário configurar um **proxy ou alias** no servidor, redirecionando as rotas do DTIC para as equivalentes de Manutenção. Essa camada de compatibilidade não existe atualmente.

## 4. Evidências de chamadas e respostas

Como o ambiente de API não está disponível durante esta análise, as evidências baseiam-se nos trechos de código que montam as requisições e nos comentários de logging do backend.

- O serviço do frontend constrói a URL de `stats-gerais` incluindo `?inicio=YYYY-MM-DD&fim=YYYY-MM-DD` [15].
- Para `ranking-entidades` e `ranking-categorias`, o código acrescenta `top` ao query string [7].
- O backend registra no log os parâmetros recebidos: no endpoint `/manutencao/stats-gerais` o log inclui `inicio`, `fim`, `novos`, `pendentes`, etc. [16].
- Para `/status-totais`, o log do backend informa os totais retornados, listando `novos`, `nao_solucionados`, `planejados`, `solucionados`, `fechados` e `resolvidos` [17].

Estas evidências confirmam que as rotas estão implementadas e que os parâmetros mencionados na documentação são necessários.

## 5. Plano de correção operacional (sem alterar código)

A seguir estão as recomendações para alinhar o comportamento do Dashboard de Manutenção ao do DTIC, sem modificar o código fonte.

1. **Ajustar variáveis de ambiente do frontend**
2. Definir explicitamente `VITE_API_BASE_URL=http://127.0.0.1:8010/api/v1` para apontar para a porta correta do backend.
3. Configurar **somente uma** variável de polling: `VITE_REALTIME_POLL_INTERVAL_SEC=12` (ou, alternativamente, `VITE_REALTIME_POLL_INTERVAL_MS=12000`). Remover a outra variável do `.env` para evitar ambiguidade.
4. Verificar as outras variáveis (`VITE_CATEGORY_CAROUSEL_INTERVAL_MS/SEC`) para garantir que não ultrapassem a frequência desejada; como afetam apenas a animação, podem manter os valores atuais.

5. **Verificar e ajustar variáveis de ambiente do backend**

6. Garantir que `API_URL`, `GLPI_APP_TOKEN` e `GLPI_USER_TOKEN` estão corretas e válidas.
7. Definir `CACHE_TTL_SEC` entre **60 e 180 s**; por exemplo, `CACHE_TTL_SEC=60` permite que cada bloco do dashboard, que atualiza a cada 12 s, consulte dados ainda relativamente recentes sem causar sobrecarga no GLPI.
8. Ajustar `SESSION_TTL_SEC` para **600 s** para reduzir reautenticações.

9. **Configurar proxy ou alias para rotas**

10. Se houver outros sistemas ou scripts que consomem as rotas do DTIC (`/metrics-gerais`, `/status-niveis`, etc.), configurar um **proxy reverso (nginx, Traefik, etc.)** que redirecione essas chamadas para as rotas equivalentes em Manutenção sem alterar o código. Exemplo: mapear `GET /api/v1/dtic/metrics-gerais` → `GET /api/v1/manutencao/stats-gerais` (com os mesmos parâmetros).

11. **Verificar parâmetros obrigatórios nas integrações**

12. Certificar-se de que qualquer cliente que consuma `/stats-gerais` sempre inclui `inicio` e `fim` no formato ISO 8601 (`YYYY-MM-DD`). Para simplificar, o proxy pode injetar um período padrão (p.ex.: últimos 30 dias) se os parâmetros estiverem ausentes.

13. **Avaliar uso de `/status-totais` no frontend**

14. O dashboard de Manutenção ainda não exibe totais de status como "Em atendimento"/"Solucionados". Para alinhar ao DTIC sem modificar o código, considerar uma regra de negócio temporária: reutilizar os valores de `stats-gerais` e apresentá-los com rótulos equivalentes a "Em atendimento" (por exemplo, somar `pendentes` e `planejados`). No entanto, a solução recomendada é posterior: alterar o frontend para consumir diretamente `/status-totais`.

15. **Testar periodicidade de atualização e verificar logs**

16. Após configurar as variáveis, reiniciar o backend e o frontend.
17. Acessar o dashboard no endereço `http://localhost:5002/` (ou porta configurada) e abrir as ferramentas de desenvolvedor do navegador (tab **Network**). Verificar que as requisições às rotas (`/stats-gerais`, `/ranking-entidades`, `/ranking-categorias`, `/tickets-novos`) ocorrem a cada ~12 s.
18. Confirmar, visualmente, que as métricas e rankings são atualizados quando há alteração nos dados. Caso algum bloco não atualize, anotar a rota correspondente e checar se a requisição está sendo feita.
19. Verificar se não há erros 4xx/5xx no console ou na aba de rede.

# 6. Checklist de validação

Para garantir que o dashboard atualiza em aproximadamente 12 s e que o ambiente está consistente, siga este roteiro:

1. **Configuração**
2. [ ] `VITE_API_BASE_URL` aponta para `http://127.0.0.1:8010/api/v1`.
3. [ ] Somente uma das variáveis `VITE_REALTIME_POLL_INTERVAL_*` está presente e vale **12 s** (ou 12 000 ms).
4. [ ] `CACHE_TTL_SEC` ≥ 60 s e `SESSION_TTL_SEC` ≈ 600 s.

5. [ ] Variáveis de autenticação (`API_URL`, `GLPI_APP_TOKEN`, `GLPI_USER_TOKEN`) configuradas e válidas.

6. **Rotas e parâmetros**

7. [ ] Testar `GET /api/v1/manutencao/status-totais` → responder 200 OK com chaves esperadas (novos, em_atendimento ou solucionados, nao_solucionados, planejados, solucionados, fechados, resolvidos).
8. [ ] Testar `GET /api/v1/manutencao/stats-gerais?inicio=YYYY-MM-DD&fim=YYYY-MM-DD` → responder 200 OK com métricas (novos, pendentes, planejados, resolvidos).
9. [ ] Testar `GET /api/v1/manutencao/ranking-entidades?...&top=10` e `GET /api/v1/manutencao/ranking-categorias?...&top=30` → responder 200 OK.
10. [ ] Testar `GET /api/v1/manutencao/tickets-novos?limit=8` → responder 200 OK.
11. **Dashboard**
12. [ ] Ao abrir `http://localhost:5002/`, conferir que “Stats gerais”, rankings e tickets novos atualizam a cada ~12 s.
13. [ ] Ao alterar o período (`inicio`, `fim`) no seletor de datas, confirmar que as requisições usam os novos parâmetros.
14. [ ] Monitorar o console do navegador para garantir ausência de erros de rede.

# 7. Conclusão

O Dashboard de Manutenção pode alcançar um comportamento de atualização em 12 s alinhado ao DTIC sem mudanças no código, mediante ajustes de configuração e infraestrutura. A principal inconsistência observada é a diferença de nomes e parâmetros das rotas, agravada pela configuração incorreta do `VITE_API_BASE_URL` e por uma definição ambígua do intervalo de polling. O backend expõe as informações necessárias, incluindo totais por status, mas o frontend atualmente consome apenas parte delas. Após a regularização das variáveis de ambiente e do cache, recomenda-se monitorar o dashboard com atenção ao intervalo de atualização e, em uma etapa posterior, evoluir o código para consumir `/status-totais`, incluir cartões adicionais e reduzir a dependência de proxies.

---

1 2 3 6 9 14 MaintenanceDashboard.tsx
https://github.com/jonathanmoletta17/dtic_dashboard/blob/main/apps/manutencao/frontend/src/MaintenanceDashboard.tsx

4 5 16 17 maintenance_stats_router.py
https://github.com/jonathanmoletta17/dtic_dashboard/blob/main/apps/manutencao/backend/api/maintenance_stats_router.py

7 8 10 11 12 13 15 maintenance-api.ts
https://github.com/jonathanmoletta17/dtic_dashboard/blob/main/apps/manutencao/frontend/src/services/maintenance-api.ts