# Análise dos Elementos de Comunicação entre Front‑end e Back‑end (Gabarit‑AI)

## Visão geral

O Gabarit‑AI é dividido em dois repositórios públicos: gabarita-ai-frontend (Next.js) e gabarita-backend (Flask). O front‑end comunica‑se com a API através do serviço ApiService e assume que o back‑end expõe uma série de endpoints REST em /api/.... Durante a análise, foi constatado que o repositório do back‑end público contém apenas run.py, main.py, render.yaml e requirements.txt, enquanto **os módulos importados em main.py (src/routes/signup, auth, planos, jogos, news, opcoes etc.) não existem no repositório**[[1]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L1-L14). Isso indica uma divergência entre o código implantado no Render e o código versionado no GitHub.

## Front‑end: Mapeamento de endpoints

A classe ApiService define a base da comunicação com o back‑end. Ela configura o baseURL com a variável de ambiente ou assume https://gabarita-ai-backend.onrender.com por padrão[[2]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L16-L25). Abaixo segue o mapeamento dos métodos definidos em services/api.ts com os endpoints utilizados:

| Método da API | Endpoint chamado | Parâmetros Enviados | Observações |
| --- | --- | --- | --- |
| healthCheck() | GET /api/health | – | Ok; endpoint existe em main.py[[3]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L105-L111). |
| login(email, password) | POST /api/auth/login | { email, password } | Implementado apenas como *stub*; retorna sempre o mesmo usuário e ignora cadastro[[4]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L113-L126). |
| signup(userData, firebaseToken?) | POST /api/auth/cadastro | Objeto userData contendo campos como name, email, cpf, password, cargo, bloco[[5]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L78-L87) | **Back‑end não possui rota /api/auth/cadastro no repositório público**; erro “Campo nome é obrigatório” indica que a API espera nome, não name. |
| getProfile() | GET /api/user/profile | – | Não existe no código do back‑end. |
| updateProfile(userData) | PUT /api/user/profile | Objeto parcial de usuário | Não existe no back‑end. |
| generateQuestions(params) | POST /api/questoes/gerar | { subject?, difficulty?, count?, bloco?, cargo? } | Implementado em main.py como gerar\_questao\_endpoint[[6]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L129-L161); aceita cargo e bloco, mas também depende de usuario\_id (não enviado pelo front). |
| submitSimulation(answers, questionIds) | POST /api/simulados/submit | { answers, questionIds } | Não existe no back‑end. |
| getMacetes(id), getPontosCentrais(id), getOutrasExploracoes(id) | GET /api/questoes/macetes/:id, etc. | – | Não existem rotas correspondentes. |
| chatDuvidas(id, usuarioId, mensagem) | POST /api/questoes/chat-duvidas | Objeto com questao\_id, usuario\_id, mensagem | Não há rota. |
| getPerformance() | GET /api/performance | – | Não há rota. |
| getPlans() | GET /api/plans | – | Provavelmente deveria ser /api/planos (Português), conforme main.py[[7]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L25-L31). |
| getRanking() | GET /api/ranking | – | Não existe rota; módulo de ranking previsto no blueprint, mas não implementado. |
| getNews() | GET /api/news | – | Rota depende do blueprint news\_bp, mas não presente no repositório. |
| createPayment(planId) | POST /api/pagamentos/criar | { plano: planId } | Não existe rota. |
| getCargosEBlocos() | GET /api/opcoes/cargos-blocos | – | Repositório não possui rota; em main.py há apenas /api/opcoes/test[[8]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L77-L98). |
| getBlocosCargos() | GET /api/opcoes/blocos-cargos | – | Não há rota correspondente. |
| getCargosPorBloco(bloco) | GET /api/opcoes/cargos/:bloco | – | Não existe. |
| getBlocosPorCargo(cargo) | GET /api/opcoes/blocos/:cargo | – | Não existe. |

## Front‑end: coleta de dados do usuário

O formulário de cadastro (app/signup/page.tsx) define o estado formData com as propriedades name, cpf, email, password, confirmPassword, cargo e bloco[[9]](https://raw.githubusercontent.com/techiaemp-netizen/gabarita-ai-frontend/master/app/signup/page.tsx#:~:text=export%20default%20function%20SignupPage%28%29%20,cargo%3A%20%27%27%2C%20bloco%3A%20%27%27). Ao submeter o formulário, o front‑end chama signup() passando esse objeto e acrescentando campos adicionais (level, xp, accuracy, plan)[[10]](https://raw.githubusercontent.com/techiaemp-netizen/gabarita-ai-frontend/master/app/signup/page.tsx#:~:text=try%20,accuracy%3A%200%2C%20plan%3A%20%27free%27). O rótulo do campo no formulário está em português (“Nome Completo”)[[11]](https://raw.githubusercontent.com/techiaemp-netizen/gabarita-ai-frontend/master/app/signup/page.tsx#:~:text=%7B%2F,2%22%3E%20Nome%20Completo), mas o nome da propriedade é name. Esse dado chega ao back‑end como name e não nome.

## Back‑end: implementação disponível

O arquivo main.py registra vários blueprints (signup, auth, questoes, planos, jogos, news, opcoes) e define algumas rotas auxiliares[[12]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L24-L31). Todavia, os módulos importados não existem no repositório público, impossibilitando verificar a implementação real. As rotas definidas diretamente em main.py são:

* **/api/health e /health** – Retornam status “healthy”[[3]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L105-L111).
* **/api/auth/login** – Retorna um usuário fictício e um token de demonstração[[4]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L113-L126).
* **/api/questoes/gerar** – Gera questões com base em usuario\_id, cargo e bloco[[6]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L129-L161). Se cargo ou bloco não forem encontrados, devolve erro 404[[13]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L147-L155). O front‑end envia bloco e cargo, mas não usuario\_id, então o valor padrão 'user-default' é utilizado.
* **/api/opcoes/test** – Um endpoint de teste para opções, retornando número de cargos carregados[[8]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L77-L98).

Não há nenhuma rota visível para /api/auth/cadastro, /api/user/profile, /api/opcoes/cargos-blocos ou /api/pagamentos/criar. Logo, as chamadas feitas pelo front‑end resultarão em 404 Not Found ou, quando tratadas por blueprints inexistentes, podem causar erros 500.

## Erros e inconsistências identificados

1. **Campo name vs. nome** – O front‑end envia name no objeto de cadastro, enquanto o back‑end aparenta esperar nome. O erro exibido (“Campo nome é obrigatório”) decorre dessa discrepância. A interface mostra que o campo está preenchido, mas o backend rejeita porque a propriedade tem outro nome.
2. **Endpoint de cadastro inexistente** – ApiService envia os dados para POST /api/auth/cadastro[[5]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L78-L87). Esse endpoint não aparece em main.py nem em qualquer módulo publicado; consequentemente, a requisição retorna erro ou 404. Mesmo que o endpoint exista no código privado, o repositório público não o versionou, dificultando correções colaborativas.
3. **Repositório backend incompleto** – Os módulos src/routes/signup, auth, planos, jogos, news, opcoes e config/firebase\_config importados em main.py não estão presentes no repositório[[1]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L1-L14). Essa falta de arquivos pode explicar vários problemas de implantação e impede o diagnóstico preciso das rotas.
4. **Rotas ausentes ou nomenclatura divergente** – Diversos métodos do front‑end apontam para rotas que não existem no back‑end público, como:
5. /api/questoes/macetes/:id, /api/questoes/pontos-centrais/:id, /api/questoes/outras-exploracoes/:id, /api/questoes/chat-duvidas – não implementadas.
6. /api/performance, /api/ranking, /api/news, /api/user/profile – não implementadas.
7. /api/plans – provavelmente deveria ser /api/planos conforme o blueprint de registro dos planos[[12]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L24-L31).
8. /api/opcoes/cargos-blocos, /api/opcoes/blocos-cargos, /api/opcoes/cargos/:bloco, /api/opcoes/blocos/:cargo – não identificadas; o repositório expõe apenas /api/opcoes/test[[8]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L77-L98).
9. **Respostas inconsistentes** – O método signup() do ApiService espera que o back‑end responda com { sucesso: true, token: <string>, usuario: <User> }[[14]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L78-L100). Contudo, o stub de login retorna { success: True, user: {…}, token: … }[[4]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L113-L126). A diferença nos nomes (success vs. sucesso, usuario vs. user) provocará falhas ao interpretar a resposta.
10. **Dependências não configuradas** – O back‑end faz referência a chatgpt\_service, firebase\_config, MercadoPago e outros serviços externos. Se as chaves de ambiente (OPENAI\_API\_KEY, FIREBASE\_…, MERCADO\_PAGO\_…) não forem definidas no Render, as importações podem falhar, causando erros no import ou na inicialização.
11. **Problemas de implantação** – Em interações anteriores foi observado que:
12. A ausência de src/\_\_init\_\_.py causou erro de import ao iniciar o Gunicorn. Adicionar esse arquivo resolveu a importação dos módulos.
13. Configurar o mesmo repositório em “Connect Repository” e “Public Git Repository” no Render gerou conflito, resolvido ao remover a URL duplicada.
14. **Nomenclatura inconsistente de idiomas** – O front‑end alterna entre português e inglês (plans vs. planos, cargo vs. position, nome vs. name), o que dificulta a padronização e gera confusão nos nomes das rotas e campos.
15. **Funcionalidades do blueprint ainda não implementadas** – O blueprint de 17 módulos prevê ranking, barra de progresso de vida, monetização via Pix, sistema de afiliados, análise de notícias e FAQ, entre outros. Nenhuma dessas funções tem rotas disponíveis no back‑end público, indicando que ainda estão em desenvolvimento ou ausentes.

## Sugestão de prompt de correção para Trae (enfatizando erros)

**Diagnóstico do projeto Gabarit‑AI:**

* O formulário de cadastro do front‑end envia o campo name, mas o back‑end rejeita solicitando nome. Isso causa o erro “Campo nome é obrigatório”. Verifique se o endpoint /api/auth/cadastro está lendo a chave correta ou ajuste o front para enviar nome.
* O repositório gabarita-backend no GitHub está incompleto: ele importa blueprints (signup, planos, news, opcoes, etc.) que não existem no repositório público[[1]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L1-L14). Essa ausência impede a validação das rotas e causa erros 404 em diversas chamadas. Confirme se o código implantado no Render corresponde ao repositório público ou se falta publicar arquivos.
* O método de cadastro (/api/auth/cadastro) não está implementado em main.py. Verifique se ele foi definido em algum módulo privado ou se ainda falta criar a rota.
* As respostas das APIs usam formatos diferentes: alguns retornam success, outros sucesso, e os objetos de usuário variam (user vs. usuario). Essa inconsciência gera falhas na interpretação das respostas pelo front‑end.
* Várias rotas chamadas pelo front (/api/questoes/macetes, /api/performance, /api/ranking, /api/news, /api/pagamentos/criar, /api/opcoes/cargos-blocos etc.) não existem no back‑end, resultando em erros 404. Avalie a lista de endpoints necessários no ApiService[[15]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L140-L161)[[16]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L231-L274) e implemente as rotas correspondentes no Flask ou ajuste o front.
* A rota /api/plans usada pelo front parece incoerente com a nomenclatura do back‑end, que usa “planos” em português[[7]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L25-L31).
* O stub de login retorna dados estáticos e não confere com usuários cadastrados. Será necessário substituir pelo serviço real de autenticação.
* Garanta que o arquivo src/\_\_init\_\_.py esteja presente no repositório para permitir importações corretas e que a configuração do Render use apenas “Connect Repository” para evitar conflitos.

Forneça um relatório de como pretende alinhar os nomes de campos e rotas entre front e back, publique o código faltante do backend e padronize as respostas JSON para que o front‑end interprete corretamente.

1. **Contratos divergentes (request/response) em mais chaves**

* Além de name vs nome e success vs sucesso, o front alterna user/usuario, error/erro/message e espera campos diferentes por rota.
* Sugestão: padronizar **tudo** em um mini-contrato (ex.: { success:boolean, data:any, error?:string }) e aplicar em **todas** as rotas.

1. **Mismatch de método/rota**

* O front chama alguns endpoints que não existem **ou** com o método errado (ex.: GET vs POST). Listei do front:
  + POST /api/questoes/gerar ✅ existe
  + GET /api/questoes/macetes/:id ❓ não visto no back
  + GET /api/questoes/pontos-centrais/:id ❓
  + POST /api/questoes/chat-duvidas ❓
  + GET /api/performance ❓
  + GET /api/ranking ❓
  + GET /api/news ❓
  + POST /api/pagamentos/criar ❓ (Mercado Pago)
  + GET /api/opcoes/cargos-blocos / GET /api/opcoes/cargos/:bloco / GET /api/opcoes/blocos/:cargo ❓
  + GET /api/plans (inglês) **vs** back tende a “planos” (português).
* Onde o backend público não tem blueprint, o front vai tomar **404**.

1. **CORS com credenciais e ‘\*’**

* No Flask-CORS você ativou supports\_credentials=True e incluiu '\*' em origins. Especificação não permite Access-Control-Allow-Origin: \* quando há credenciais.
* Mantenha **apenas** a(s) origem(ns) explícita(s) (ex.: https://gabarita-ai-frontend-pied.vercel.app e http://localhost:3000) e remova '\*'.

1. **Ambiente e baseURL**

* O front usa NEXT\_PUBLIC\_API\_BASE\_URL || NEXT\_PUBLIC\_API\_URL || https://gabarita-ai-backend.onrender.com.
* Se o .env do Vercel tiver variável antiga/apontando pra endpoint errado, você terá erro “fantasma”. Confirme as envs no **Projeto Vercel** (Production/Preview/Dev) e no **Render**.

1. **Preflight/Headers**

* axios envia Content-Type: application/json (ok), mas rotas novas precisam responder OPTIONS corretamente (você já tratou em /api/test; replique nos blueprints reais).

1. **Tratamento de erros no front**

* apiService.signup trata error.response?.data?.message || ...?.erro. Outras chamadas usam message ou assumem estrutura diferente. Uniformize a leitura de erro para evitar “Erro ao X” genérico.

1. **Timeout e logs**

* timeout: 10000 no axios é razoável, mas geração de questão pode levar mais; se explodir, o front vai reportar erro de rede. Considere 20–30 s nessa rota específica (ou streaming).
* No back, há print/sys.stdout.flush() na geração de questões (bom pra log). Garanta que **todos** os endpoints sensíveis loguem entrada mínima e status final.

1. **Login stub**

* Você já notou: login de mentira devolve nome = email. Isso pode quebrar UX (exibição de perfil) e testes de autorização. Troque para o serviço real **antes** de abrir o fluxo.

1. **Repositório público desatualizado**

* Além de faltar blueprints, isso gera **drift**: Render roda uma versão, GitHub mostra outra. Defina política:
  + “Tudo que roda em produção está **no Git**”;
  + Deploy **sempre** via **Connect Repository** (sem Public Git) e branch fixada (ex.: main);
  + src/\_\_init\_\_.py versionado (sem ele, pacotes quebram import).

1. **OpenAPI/Insomnia**

* Falta a fonte de verdade do contrato. Gere um **OpenAPI (YAML/JSON)** simples e um **Insomnia/Postman collection**. Sem isso, tende a reocorrer divergência.

**Checklist de conformidade (rápido)**

* Padronizar chaves: nome (ou name) **um só**, success, data, error.
* Unificar endpoints: “planos” **ou** “plans” (escolha 1).
* Implementar (ou remover do front) **todas** as rotas marcadas “❓”.
* Corrigir CORS (sem '\*' com credenciais).
* Confirmar envs do Vercel e Render (um único baseURL).
* Adicionar src/\_\_init\_\_.py e publicar blueprints ausentes.
* Substituir login stub.
* Adicionar OPTIONS às rotas que recebem POST/PUT/DELETE.
* Aumentar timeout da rota de geração (ou paginar/stream).
* Publicar **OpenAPI** e **collection de testes** (CI validando contratos).

**Prompt enxuto pro Trae (foco em diagnóstico + correção)**

**Objetivo:** Eliminar todos os erros de integração FE/BE do Gabarit-AI, padronizar contratos e garantir que o que roda no Render corresponde ao GitHub.

**Tarefas (ordem):**

1. **Mapear contratos reais** do front (services/api.ts) e gerar uma **tabela** com:
   * Rota, método, payload esperado, response esperado (chaves e tipos).
2. **Auditar o backend** publicado no Render e no repo gabarita-backend:
   * Verificar se **cada rota do front existe** (método e path idênticos).
   * Se faltar, **implementar** blueprint e registrar no main.py.
   * Padronizar respostas para { success:boolean, data:any, error?:string }.
3. **Corrigir cadastro**:
   * Backend deve aceitar **nome e name** (normalização server-side) **ou** ajustar o front para enviar nome.
   * Garantir validação e mensagens em campo error (único).
4. **CORS**:
   * Remover '\*' de origins quando supports\_credentials=True.
   * Manter apenas https://gabarita-ai-frontend-pied.vercel.app e http://localhost:3000.
   * Confirmar OPTIONS em todas as rotas que exigem preflight.
5. **Ambiente**:
   * Verificar .env do Vercel: NEXT\_PUBLIC\_API\_BASE\_URL deve apontar para o mesmo host rodando no Render.
   * No Render: usar **apenas** “Connect Repository” e travar branch (ex.: main).
6. **Uniformização de endpoints**:
   * Escolher **português** (/api/planos) ou **inglês** (/api/plans) e alinhar front+back.
7. **Login real**:
   * Substituir stub por autenticação real (JWT), retornando { success, data:{ user, token } }.
8. **Observabilidade**:
   * Adicionar logs estruturados (rota, status, duração, erro).
   * Opcional: /api/health expandido com versão do commit (env GIT\_SHA).
9. **Contrato fonte-de-verdade**:
   * Gerar **OpenAPI** (YAML) a partir do código, commitar no repo e publicar /api/docs.
   * Entregar **collection** (Insomnia/Postman) para CI de contrato.

**Entregáveis:**

* PR no backend com: blueprints faltantes, padronização de respostas, CORS fix, login real, src/\_\_init\_\_.py.
* PR no frontend com: ajuste de nome/name (se optar pelo front), unificação de caminhos (planos vs plans), tratamento de erros centralizado.
* openapi.yaml + collection de testes.
* Relatório final listando: rotas criadas/ajustadas, difs de contrato, envs confirmadas, e captura de tela de Deploy live no Render.

[[1]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L1-L14) [[3]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L105-L111) [[4]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L113-L126) [[6]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L129-L161) [[7]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L25-L31) [[8]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L77-L98) [[12]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L24-L31) [[13]](https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py#L147-L155) main.py

<https://github.com/techiaemp-netizen/gabarita-backend/blob/main/src/main.py>

[[2]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L16-L25) [[5]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L78-L87) [[14]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L78-L100) [[15]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L140-L161) [[16]](https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts#L231-L274) api.ts

<https://github.com/techiaemp-netizen/gabarita-ai-frontend/blob/master/services/api.ts>

[[9]](https://raw.githubusercontent.com/techiaemp-netizen/gabarita-ai-frontend/master/app/signup/page.tsx#:~:text=export%20default%20function%20SignupPage%28%29%20,cargo%3A%20%27%27%2C%20bloco%3A%20%27%27) [[10]](https://raw.githubusercontent.com/techiaemp-netizen/gabarita-ai-frontend/master/app/signup/page.tsx#:~:text=try%20,accuracy%3A%200%2C%20plan%3A%20%27free%27) [[11]](https://raw.githubusercontent.com/techiaemp-netizen/gabarita-ai-frontend/master/app/signup/page.tsx#:~:text=%7B%2F,2%22%3E%20Nome%20Completo) raw.githubusercontent.com

<https://raw.githubusercontent.com/techiaemp-netizen/gabarita-ai-frontend/master/app/signup/page.tsx>