

Data Masters - AI Engineer

Ana Carolina da Silva
anasilva@f1rst.com.br

Repositório do projeto com instruções para execução: <https://github.com/anacds/dm-orquestra>

O problema de negócio.....	3
Explicação do case.....	3
Atores e perfis de acesso.....	4
Diagrama geral da solução.....	6
Frontend.....	7
Endpoints e telas.....	7
Arquitetura do serviço.....	8
API Gateway.....	9
Agent Registry.....	9
Endpoints.....	9
Arquitetura do serviço.....	10
Fluxo de funcionamento.....	11
Auth Service.....	12
Endpoints.....	12
Arquitetura do serviço.....	13
Fluxo de autenticação.....	14
Fluxo de validação do token.....	15
Campaigns Service.....	16
Endpoints.....	16
Tools MCP.....	17
Arquitetura do serviço.....	18
Briefing Enhancer Service (Agente).....	19
Arquitetura do serviço.....	20
Endpoints.....	20
HTML Converter Service.....	21
Arquitetura do serviço.....	21
Endpoints.....	21
Tools MCP.....	21
Branding Service.....	22
Tools MCP.....	22
Arquitetura do serviço.....	23
Legal Service (Agente).....	24
Endpoints.....	24
Arquitetura do serviço.....	25
Content Validation Service (Orquestrador).....	26
O Padrão BFA - Backend for Agents.....	26
Arquitetura do serviço.....	27
Fluxo de funcionamento.....	27
Streaming de progresso na validação de peças.....	28

Endpoints.....	30
Evaluations para escolha dos modelos e parâmetros.....	31
Briefing Enhancer Service.....	31
Avaliação #1.....	32
Avaliação #2.....	33
Resultados.....	35
Legal Service.....	35
Avaliação.....	35
Resultados.....	36
Observabilidade e dashboards.....	38
Traces dos agentes.....	38
Dashboards técnicos.....	39
Dashboard #1 - API Gateway.....	39
Dashboard #2 - Auth Service.....	40
Dashboard #3 - Campaigns Service.....	40
Dashboard #4 - Content Validation Service.....	41
Dashboard #5 - Legal Service.....	41
Dashboard #6 - Briefing Enhancer Service.....	42
Dashboards de negócio.....	42
Dashboard #1 - Funil de campanhas.....	42
Dashboard #2 - Qualidade e compliance.....	43
Dashboard #3 - Adoção e eficácia da IA.....	44
Considerações éticas e de responsabilidade com IA.....	46
Supervisão humana.....	46
Moderação de conteúdo.....	46
Auditoria e rastreabilidade.....	46
Transparência e explicabilidade.....	46
Prevenção de abuso.....	47
Observabilidade.....	47
Melhorias futuras.....	48

O problema de negócio

Criar e publicar campanhas de CRM envolve alguns desafios, como a interação entre múltiplas equipes e processos manuais que geram atrasos, retrabalho e riscos. As dificuldades têm início já na etapa de solicitação das campanhas. Em modelos tradicionais, o uso de e-mails, planilhas e documentos distintos fragmenta a comunicação e dificulta o rastreamento de decisões e versões. Cada equipe trabalha com informações parciais, os comentários ficam dispersos em canais diferentes e o histórico de decisões se perde ao longo do processo.

Outro ponto relevante refere-se à qualidade das informações presentes nos briefings. É comum que analistas de negócios descrevam objetivos e filtros de forma genérica, como "aumentar vendas" ou "melhorar o relacionamento com o cliente", sem a definição clara de métricas, prazos ou critérios de sucesso. Essa falta de especificidade gera ambiguidades durante a fase de criação, ocasiona ciclos adicionais de ajuste e pode resultar em peças que não refletem adequadamente os objetivos estratégicos da campanha.

Explicação do case

A proposta deste projeto é automatizar e padronizar esse fluxo através do "Orquestra - Campaign Hub", uma plataforma desenvolvida para a Orquestra, empresa do setor financeiro. A ferramenta atua como uma camada de inteligência sobre o processo de criação de campanhas, reduzindo fricções entre os envolvidos e aumentando a qualidade e velocidade da operação de CRM.

A plataforma atinge esse objetivo centralizando o fluxo em um *workflow* único, com status definidos e transições controladas de acordo com o perfil de cada usuário. Dessa forma, cada equipe tem acesso apenas às informações pertinentes à sua atuação, os comentários ficam consolidados em um único ambiente e o histórico de decisões permanece registrado ao longo de todo o ciclo de vida da campanha. O processo passa a ser estruturado por meio de um formulário organizado em etapas, apoiado por um *workflow* previamente definido, assegurando padronização e consistência entre as campanhas desde a sua concepção.

Para o problema da qualidade dos *briefings*, a plataforma conta com assistência de IA generativa no preenchimento dos formulários. A IA atua como um mecanismo de apoio à elaboração do *briefing*, sugerindo aprimoramentos que transformam descrições vagas em objetivos mais específicos, mensuráveis e coerentes entre os diferentes campos do documento. As sugestões geradas são sempre submetidas à avaliação do usuário, que pode aceitá-las ou rejeitá-las, mantendo o controle decisório sobre o conteúdo final.

Na etapa de criação das peças de comunicação, a plataforma oferece validação automatizada do conteúdo produzido. As peças criativas - sejam SMS, Push, E-mail ou App - passam por uma verificação que combina regras exatas (limites de caracteres, dimensões de imagem, peso de arquivo) com análise de conformidade regulatória baseada em IA. Esta última utiliza RAG (*Retrieval Augmented Generation*) sobre a base de documentos normativos da Orquestra para identificar possíveis violações antes que a peça chegue à etapa de aprovação humana. Adicionalmente, a plataforma valida a aderência da peça aos padrões visuais da marca. Esse conjunto de validações reduz o ciclo de revisão e diminui o risco de publicação de comunicações fora dos padrões regulatórios ou de marca.

Atores e perfis de acesso

1. Analista de negócios

Descrição:

Profissional responsável por definir a estratégia da campanha, traduzindo objetivos de negócio em briefings claros para execução operacional e criativa. É quem conecta a necessidade comercial com a execução de CRM, garantindo que a campanha faça sentido para o negócio e para o cliente final.

Permissões:

- Cria campanhas
- Edita campanhas
- Envia para o time de criação

Usuário de exemplo:

- ana@email.com / 123

2. Analista de criação

Descrição:

Profissional responsável por transformar o briefing em peças criativas, garantindo que a comunicação esteja clara, atrativa e adequada aos canais definidos. As peças podem ter sido produzidas por um time interno ou por agências externas, mas os criativos sempre entram no sistema através do Analista de criação.

Permissões:

- Consulta briefings na fase de criação
- Faz upload de peças
- Consulta peças enviadas para ajuste
- Solicita validação por IA como apoio

Usuário de exemplo:

- maria@email.com / 123

3. Gestor de marketing

Descrição:

Profissional responsável pela governança e qualidade das campanhas de CRM, atuando como instância de revisão e aprovação final antes do disparo. Analisa os conteúdos produzidos pela equipe criativa sob a ótica de marca, compliance e alinhamento estratégico, podendo utilizar validação assistida por IA como apoio à decisão. É quem garante que nenhuma comunicação chegue ao cliente final sem passar por uma avaliação humana qualificada.

Permissões:

- Visualiza campanhas

- Revisa peças criativas
- Aprova ou reprova peças com justificativa
- Solicita validação por IA como apoio à decisão
- Envia campanhas para a etapa de criação

Usuário de exemplo:

- eric@email.com / 123

4. Analista de CRM

Descrição:

Profissional responsável pela construção técnica da campanha nos sistemas de CRM, garantindo que o briefing e as peças aprovadas sejam corretamente implementados e disparados. Atua mais próximo da execução operacional, respeitando regras de audiência, canal e governança.

Está representado na plataforma, mas não possui ação prática na ferramenta.

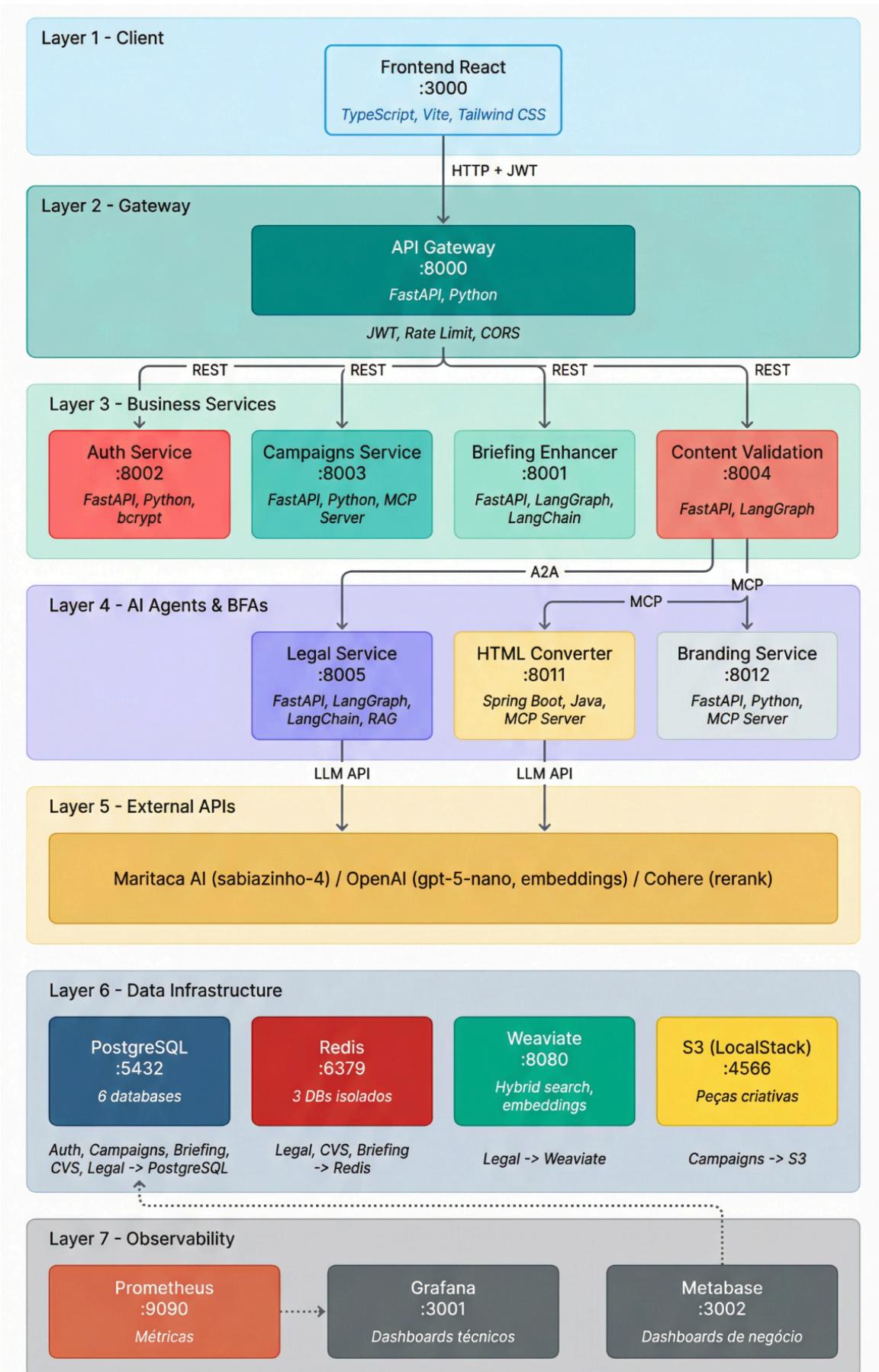
Permissões:

- Consulta briefings na fase de construção
- Constrói as campanhas (externo ao sistema)
- Publica as campanhas

Usuário de exemplo:

- jose@email.com / 123

Diagrama geral da solução



Frontend

A interface da aplicação é uma página web interativa construída em React, um dos frameworks mais utilizados para construção de interfaces modernas. A estilização dos componentes utiliza Tailwind CSS e a biblioteca de componentes Radix UI, que juntos fornecem uma aparência visual consistente e responsiva.

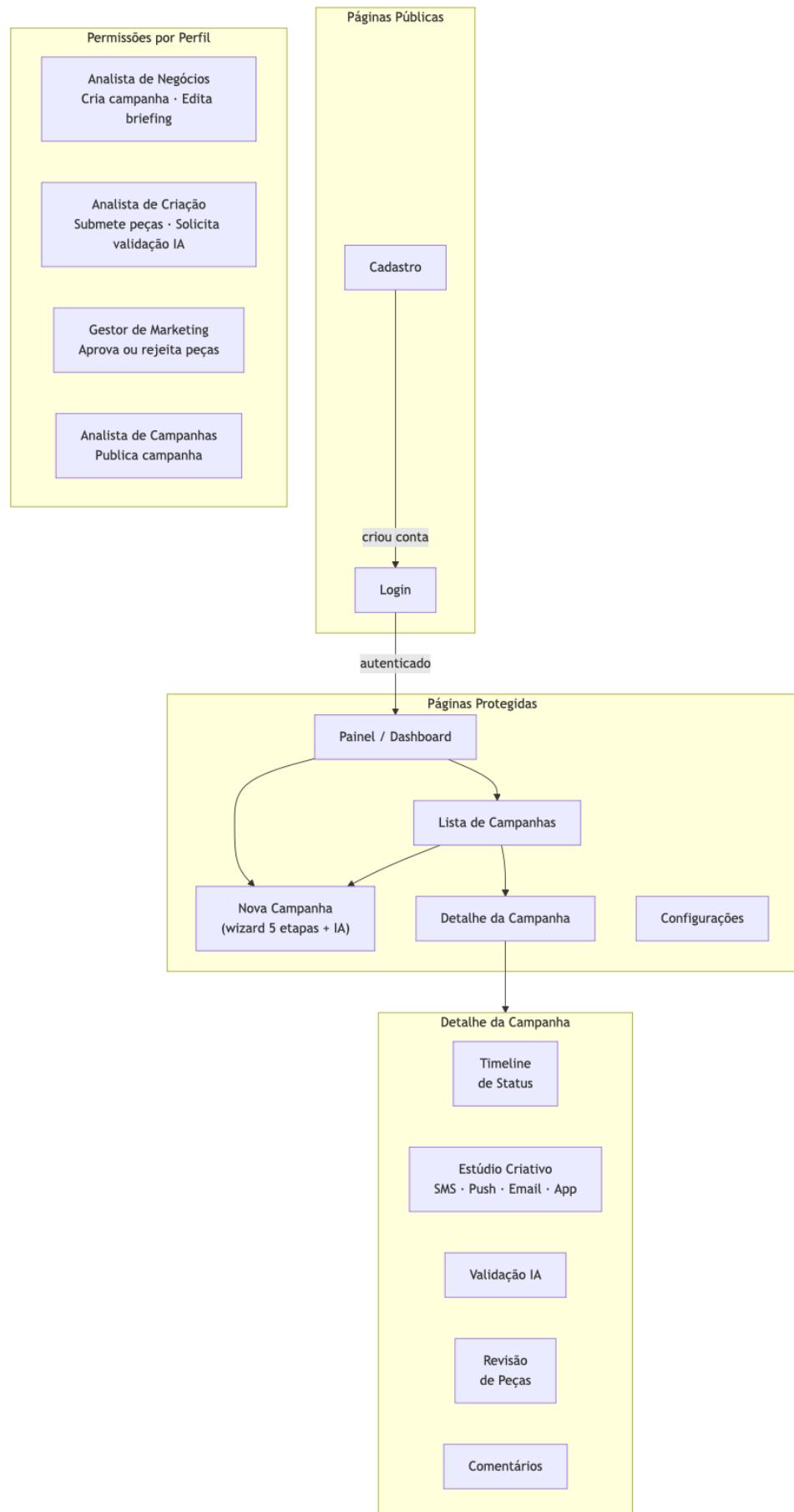
A comunicação com o backend é feita exclusivamente através do API Gateway, e a autenticação é baseada em cookies. Ou seja, o usuário faz login uma vez e as credenciais são mantidas automaticamente pelo navegador nas requisições seguintes.

O controle de acesso é feito por perfil de usuário. As páginas protegidas verificam a sessão ativa e direcionam para o login quando necessário. Dentro de cada página, os elementos são exibidos ou ocultados de acordo com o perfil de acesso, garantindo que cada usuário veja apenas as ações pertinentes à sua atuação.

Endpoints e telas

Rota	Página	Descrição
/login	Login	Formulário de e-mail e senha. Redireciona para o painel se já autenticado.
/register	Cadastro	Formulário de nome, e-mail e senha para criação de conta.
/	Painel/Dashboard	Página inicial com saudação personalizada, indicadores (campanhas ativas, aprovações pendentes, aprovadas no mês), lista de tarefas por perfil e campanhas recentes.
/campaigns	Lista de Campanhas	Listagem com busca por nome e filtros por status. Cada linha exibe nome, status com badge e link para detalhe.
/campaigns/new	Nova Campanha	Wizard de 5 etapas (Informações Básicas, Objetivos e Público, Período e Volume, Comunicação, Revisão). Campos de texto suportam aprimoramento por IA generativa, com diálogo de aceitar ou rejeitar a sugestão. Restrito ao Analista de Negócios.
/campaigns/:id	Detalhe da Campanha	Página central do workflow. Inclui: timeline horizontal de status com datas e duração entre etapas; banner de próxima ação por perfil; estúdio criativo com abas por canal (SMS, Push, E-mail, App); validação de peças com IA; timeline vertical de eventos de revisão; seção de comentários; e botões de transição de status conforme perfil e estado atual.
/settings	Configurações	Exibe os dados do usuário logado (nome, e-mail, perfil, status da conta).

Arquitetura do serviço



API Gateway

O serviço API Gateway atua como ponto único de entrada para o frontend, centralizando a comunicação com os serviços do backend. Ele recebe requisições HTTP provenientes da interface e as encaminha aos respectivos serviços com base no path da requisição (por exemplo, requisições para /api/auth são direcionadas ao serviço de autenticação, enquanto /api/campaigns são encaminhadas ao serviço de campanhas), funcionando como um proxy reverso.

Nesse processo, o gateway preserva headers, cookies e corpo da requisição, além de gerenciar a configuração de CORS, permitindo que o frontend consuma os serviços sem conflitos de origem cruzada. Além do roteamento, o gateway implementa autenticação centralizada, validando tokens JWT e extraíndo o contexto do usuário antes de encaminhar as requisições aos serviços seguintes. Esse contexto é repassado via headers personalizados (X-User-Id, X-User-Email, X-User-Role, X-User-Is-Active), permitindo que os serviços backend identifiquem o usuário sem necessidade de validar tokens diretamente.

O gateway também implementa rate limiting configurável por IP, com limites específicos por serviço e path, processamento automático de cookies Set-Cookie dos serviços backend, e endpoints de health check para monitoramento.

Agent Registry

O endpoint de discovery implementado no API Gateway (GET /a2a/discovery) atua como um registro centralizado de agentes, aderente ao conceito de Curated Registry descrito na especificação do protocolo Agent-to-Agent (A2A). Sua função é agregar e expor os Agent Cards dos agentes inteligentes disponíveis na plataforma, permitindo que clientes externos descubram as capacidades oferecidas pelo sistema sem necessidade de conhecer previamente as URLs individuais de cada serviço.

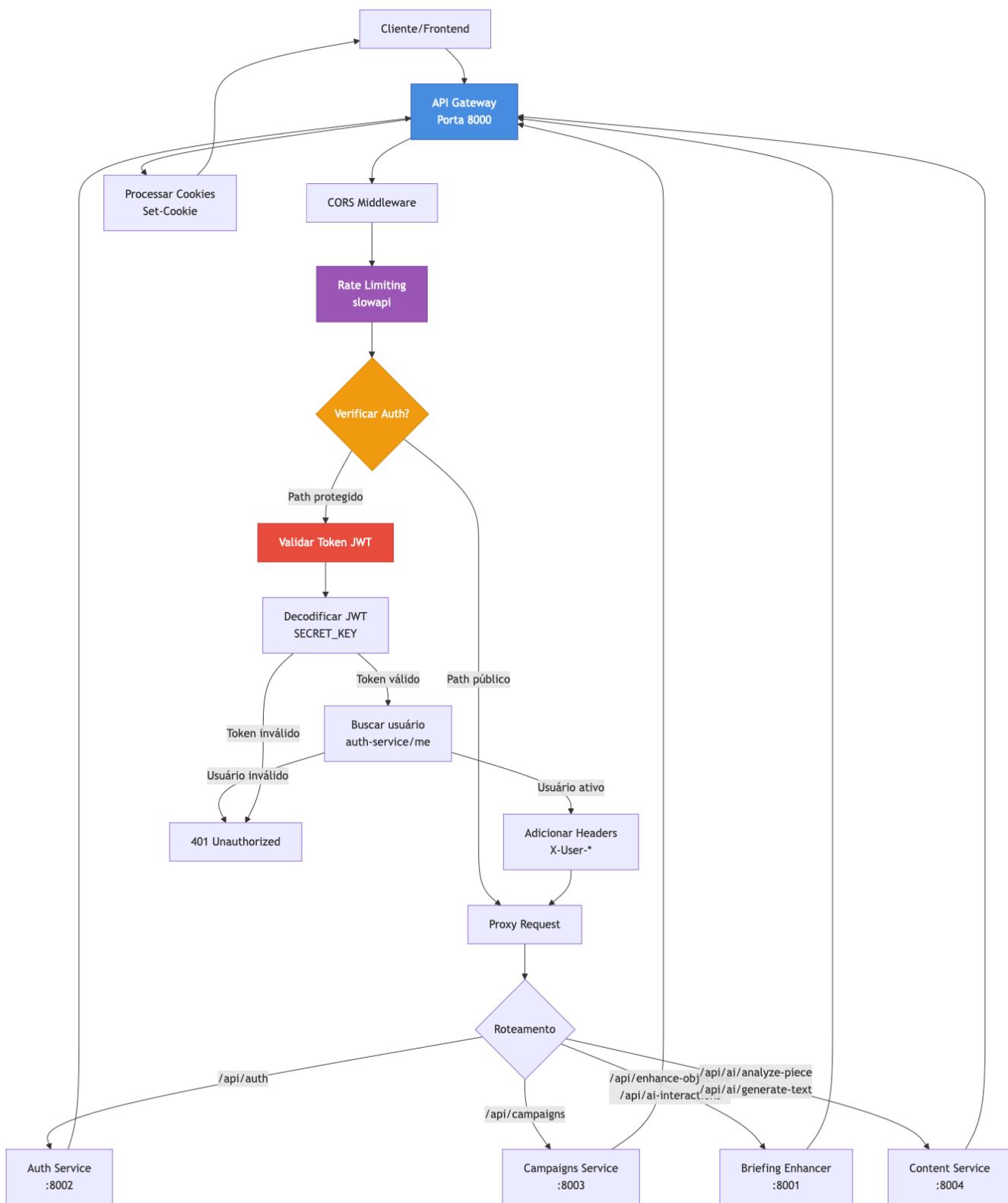
Ao receber uma requisição, o gateway realiza chamadas HTTP aos endpoints padrão de discovery de cada agente registrado (/.well-known/agent-card.json), conforme estabelecido pela RFC 8615 e adotado pela especificação A2A como mecanismo primário de descoberta. Os Agent Cards retornados contêm metadados padronizados (nome, descrição, URL do serviço, versão do protocolo, capacidades suportadas e skills disponíveis) que permitem a um agente cliente avaliar a adequação de cada agente remoto antes de iniciar a comunicação.

Endpoints

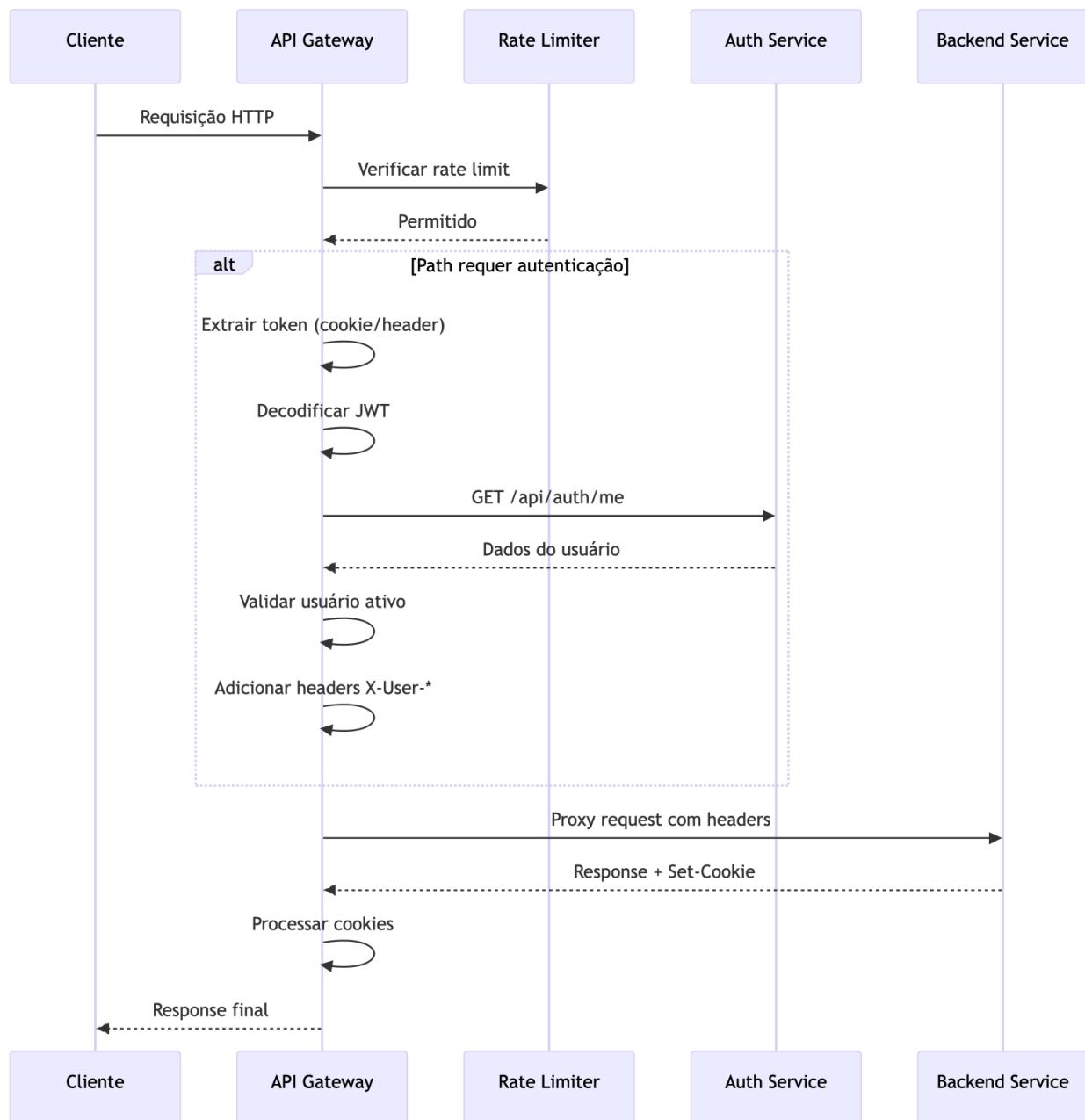
<http://localhost:8000/docs>

- GET /
 - Endpoint raiz que retorna informações básicas do Gateway (nome, versão, status). Não requer autenticação.
- GET /a2a/discovery
 - Agent Registry.

Arquitetura do serviço



Fluxo de funcionamento



Auth Service

O Auth Service é responsável pela autenticação e pelo gerenciamento de usuários do sistema. Ele implementa autenticação baseada em JWT, com uso de *refresh tokens* para renovação automática de sessões, cobrindo todo o ciclo de autenticação, desde o registro do usuário até o logout e a revogação de tokens. Durante o processo de login, todas as tentativas são auditadas, com registro de IP, usuário e resultado da tentativa (sucesso ou falha, incluindo o motivo), contribuindo para segurança, rastreabilidade e monitoramento.

Como mecanismo de proteção contra ataques de força bruta, o serviço implementa *rate limiting* configurável, com limites específicos para operações sensíveis, como login e registro.

O serviço oferece suporte a múltiplos papéis de usuário, possibilitando controle de acesso baseado em permissões. Também disponibiliza endpoints para consulta de informações de usuários, permitindo que outros serviços obtenham dados específicos quando necessário, como no caso do campaigns-service, que recupera informações do criador da campanha para exibição na interface.

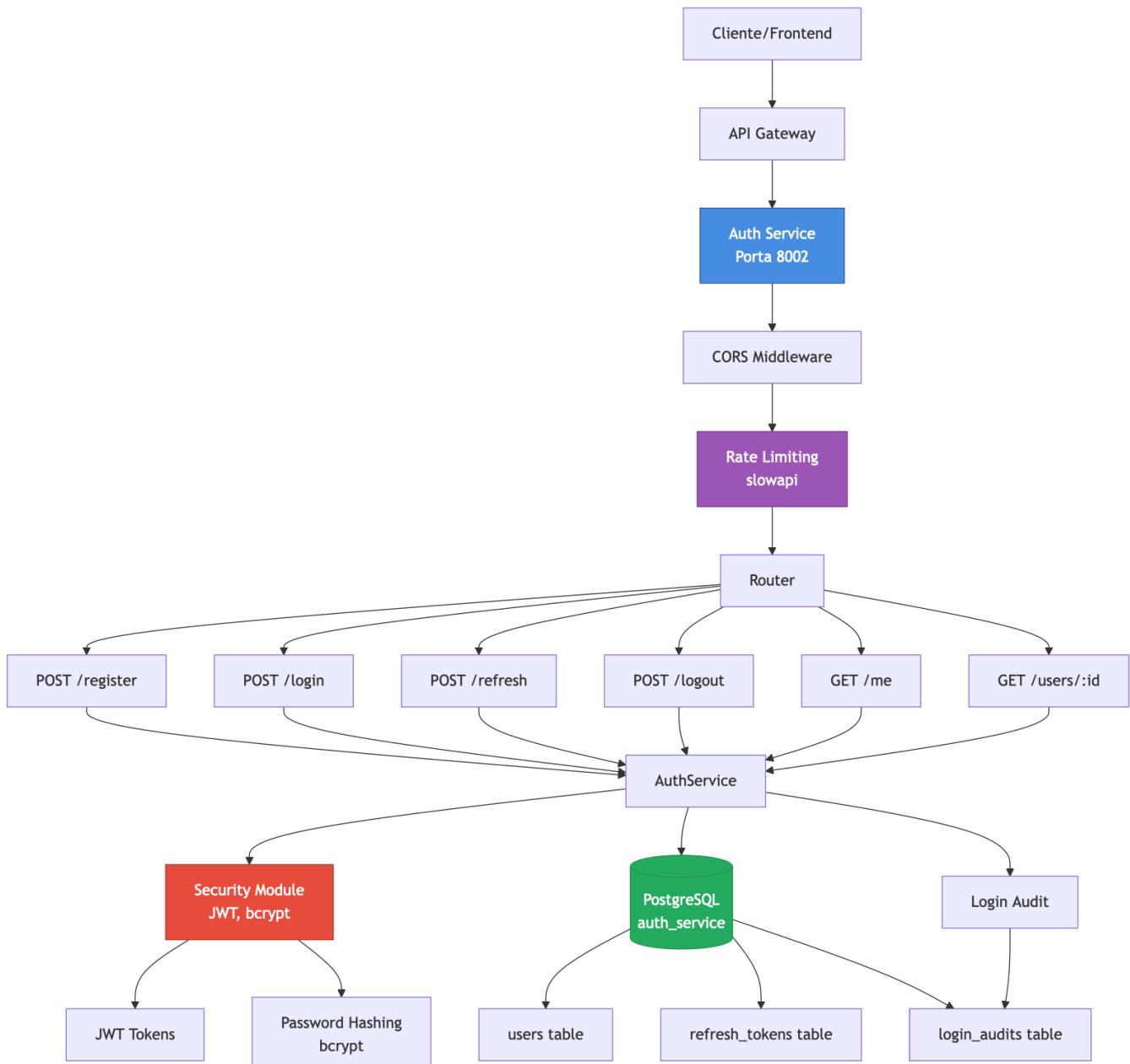
Endpoints

<http://localhost:8002/docs>

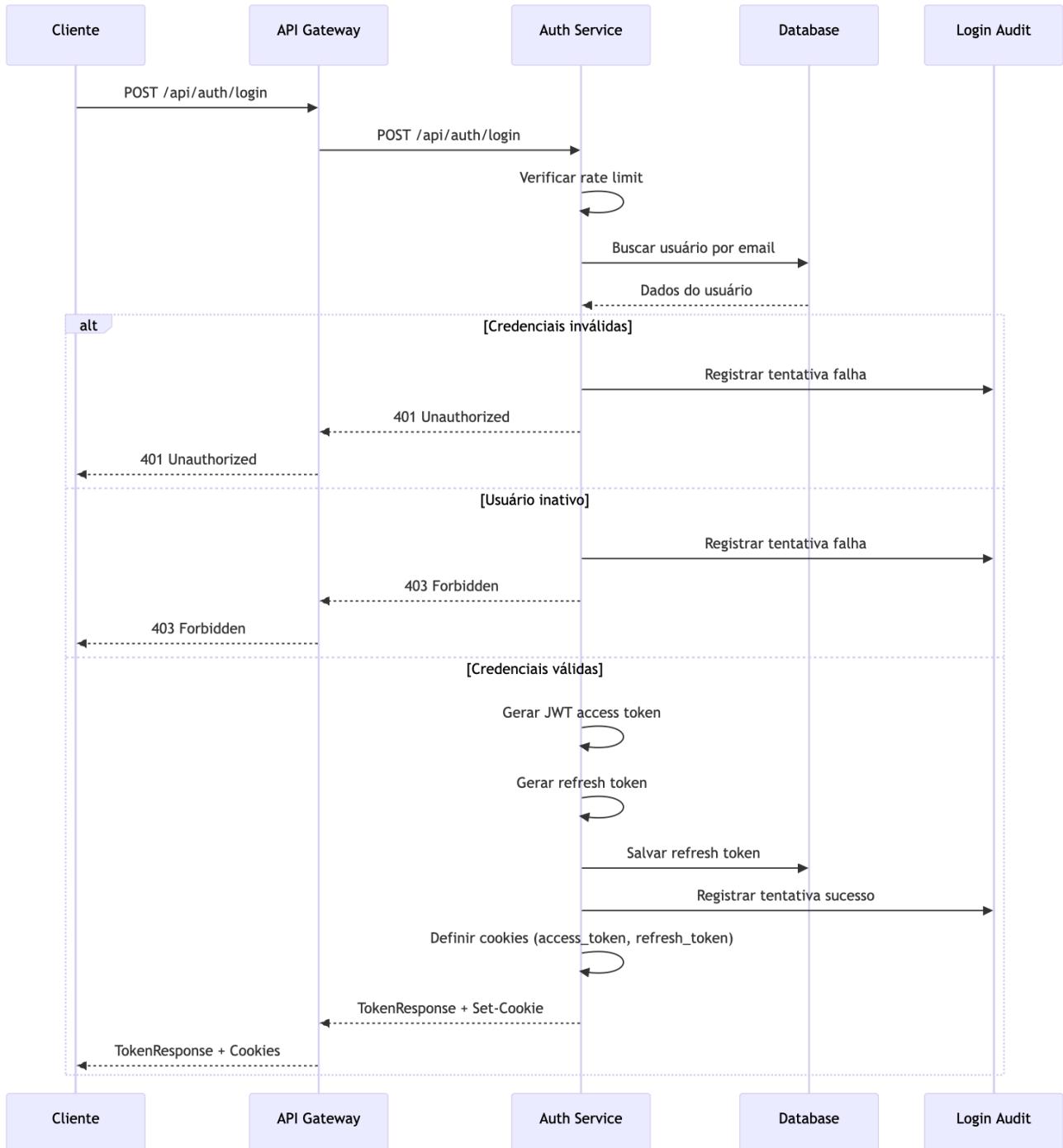
- POST /api/auth/register
 - Registra um novo usuário no sistema. Aceita email, senha, nome completo e role.
- POST /api/auth/login
 - Autentica um usuário com email e senha. Retorna access token JWT e refresh token, armazenando ambos em cookies. Registra tentativa de login na auditoria. Valida se o usuário está ativo.
- POST /api/auth/refresh
 - Renova o access token usando um refresh token válido. Valida se o refresh token não está revogado e não expirou. Atualiza os cookies com o novo access token. Se o refresh token for renovado, atualiza também o cookie do refresh token.
- POST /api/auth/logout
 - Revoga o refresh token do usuário e remove os cookies de autenticação. Requer autenticação.
- GET /api/auth/me
 - Retorna informações do usuário autenticado (id, email, nome, role, status ativo). Usado principalmente pelo API Gateway para validar tokens e obter contexto do usuário. Requer autenticação.

- GET /api/auth/users/{user_id}
 - Retorna informações de um usuário específico pelo ID. Usado por outros serviços (como campaigns-service) para obter dados de usuários. Requer autenticação.
- GET /api/health
 - Endpoint de health check que retorna status do serviço.
- GET /
 - Endpoint raiz que retorna informações básicas do serviço (nome, versão, status).

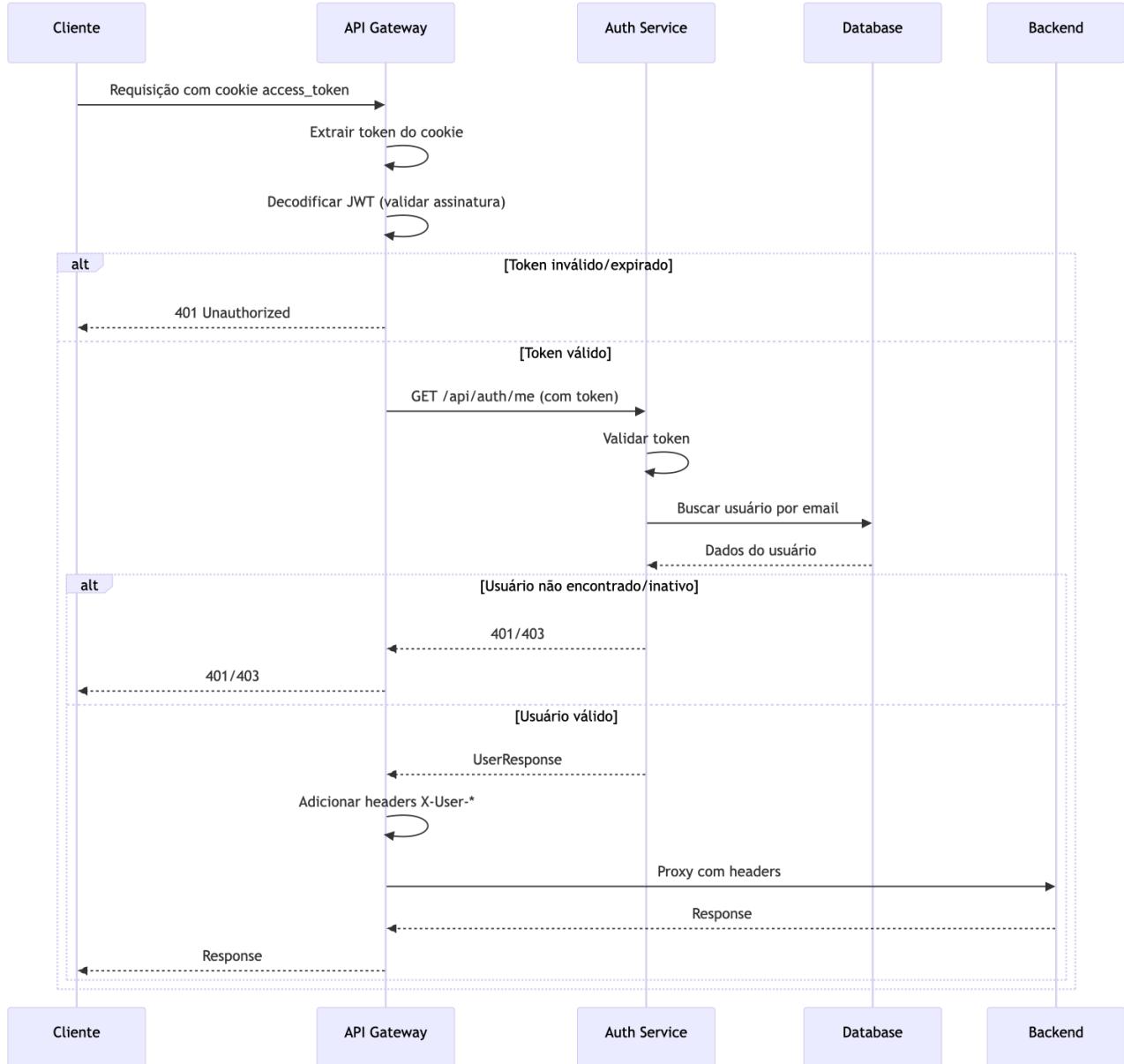
Arquitetura do serviço



Fluxo de autenticação



Fluxo de validação do token



Campaigns Service

O Campaigns Service é responsável pelo gerenciamento completo do ciclo de vida das campanhas de CRM, abrangendo sua criação, atualização, exclusão e acompanhamento ao longo de diferentes fases e status. O serviço implementa um modelo de controle de acesso baseado em papéis, no qual cada tipo de usuário possui permissões e níveis de visibilidade específicos, definidos de acordo com o status atual da campanha. As campanhas são gerenciadas por meio de um *workflow* baseado em status, no qual cada transição é validada conforme regras de negócio e o papel do usuário que executa a ação.

O serviço também permite que campanhas tenham comentários associados, viabilizando a comunicação entre diferentes papéis ao longo do processo de revisão e aprovação. Além disso, gerencia as peças criativas, que podem assumir quatro formatos distintos: SMS (texto simples), Push (título e corpo), App (arquivos PNG associados a espaços comerciais específicos) e E-mail (arquivos HTML). Os arquivos são armazenados em um bucket S3 (utilizando LocalStack no ambiente local) com visualização em tela.

Cada peça criativa passa por um fluxo de revisão composto por validação automatizada e aprovação humana. O serviço mantém, para cada peça, um registro de revisão que armazena o veredito da inteligência artificial e a decisão do revisor humano, além de um log de eventos que preserva o histórico completo de ações realizadas sobre aquela peça, como submissões, aprovações e rejeições com suas respectivas justificativas. O serviço também expõe um servidor MCP (Model Context Protocol) que disponibiliza ferramentas para que outros agentes da plataforma possam consultar o conteúdo das peças e as especificações técnicas de cada canal, como limites de caracteres para SMS e dimensões esperadas para imagens de App.

Endpoints

<http://localhost:8003/docs>

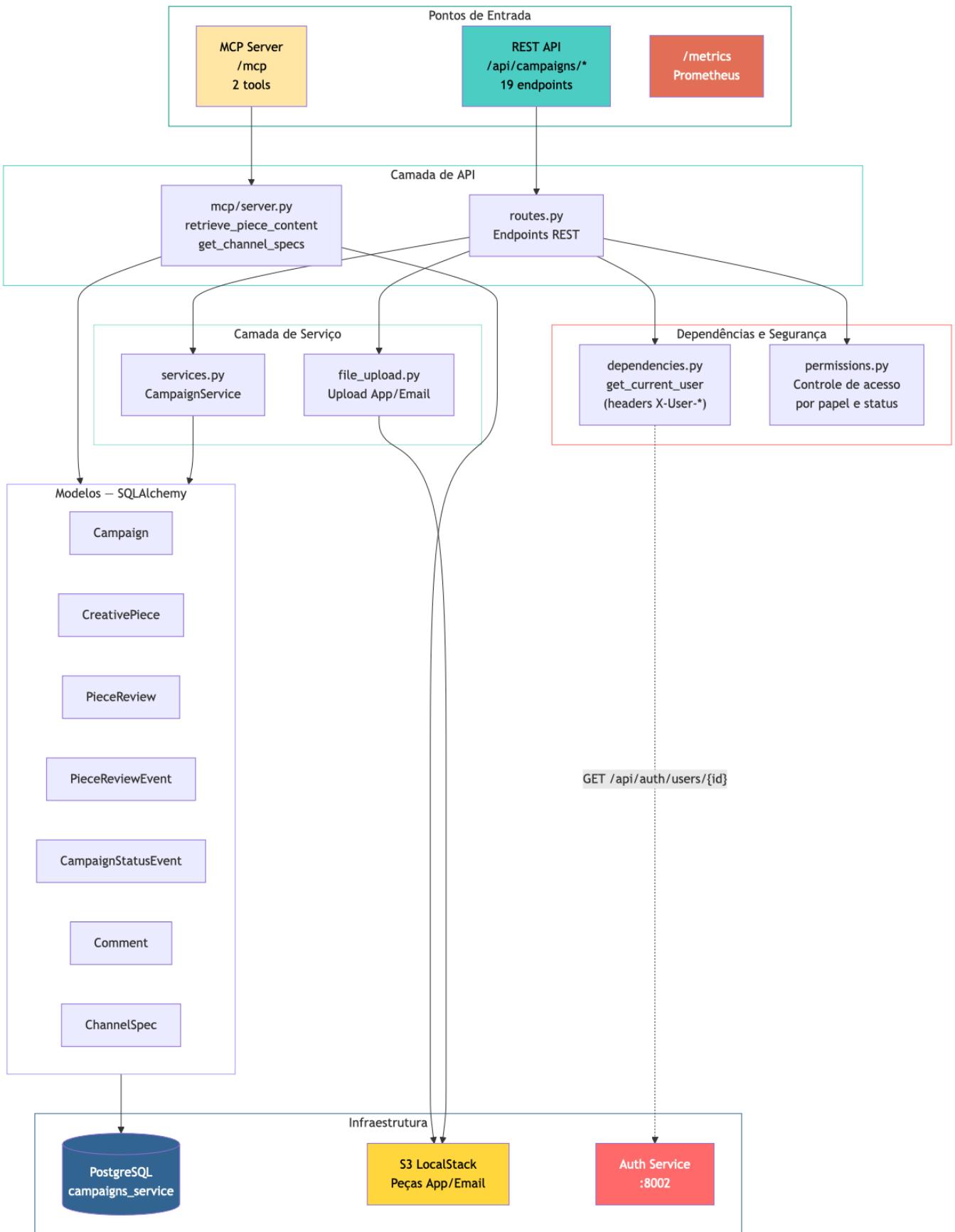
- GET /api/campaigns
 - Lista campanhas visíveis ao usuário baseado no seu role.
- GET /api/campaigns/my-tasks
 - Retorna tarefas personalizadas do usuário.
- GET /api/campaigns/{campaign_id}
 - Obtém uma campanha específica por ID.
- POST /api/campaigns
 - Cria uma nova campanha.
- PUT /api/campaigns/{campaign_id}
 - Atualiza uma campanha existente.
- DELETE /api/campaigns/{campaign_id}
 - Deleta uma campanha.
- POST /api/campaigns/{campaign_id}/submit-for-review
 - Analista de criação submete campanha para revisão.
- POST /api/campaigns/{campaign_id}/pieces/review
 - Gestor de marketing aprova/rejeita uma peça.
- PATCH /api/campaigns/{campaign_id}/piece-reviews/ia-verdict
 - Gestor de marketing atualiza veredito IA de uma peça.
- GET /api/campaigns/{campaign_id}/piece-review-history
 - Histórico de eventos de revisão de peças.
- GET /api/campaigns/{campaign_id}/status-history
 - Histórico de transições de status da campanha.

- POST /api/campaigns/{campaign_id}/comments
 - Adiciona um comentário a uma campanha.
- POST /api/campaigns/{campaign_id}/creative-pieces
 - Submete uma peça criativa (SMS ou Push).
- POST /api/campaigns/{campaign_id}/creative-pieces/upload-app
 - Faz upload de arquivo PNG para canal App.
- POST /api/campaigns/{campaign_id}/creative-pieces/upload-email
 - Faz upload de arquivo HTML para canal E-mail.
- GET /api/campaigns/{campaign_id}/creative-pieces/{piece_id}/content
 - Retorna conteúdo da peça (HTML ou imagem base64).
- DELETE /api/campaigns/{campaign_id}/creative-pieces/app/{commercial_space}
 - Deleta um arquivo App específico de um espaço comercial.
- GET /api/campaigns/{campaign_id}/download-piece
 - Download de arquivo (Email HTML ou App PNG) com Content-Disposition.
- DELETE /api/campaigns/{campaign_id}/creative-pieces/email
 - Deleta o arquivo HTML do canal E-mail.
- GET /api/health
 - Health check do serviço.

Tools MCP

- retrieve_piece_content
 - Busca conteúdo de uma peça criativa. E-mail retorna HTML; App retorna imagem em base64.
- get_channel_specs
 - Retorna especificações técnicas de um canal/espaço comercial (limites de caracteres, peso, dimensões).

Arquitetura do serviço



Briefing Enhancer Service (Agente)

O Briefing Enhancer Service é responsável por aprimorar os textos de briefing das campanhas de CRM com auxílio de inteligência artificial. Quando um Analista de Negócios submete o preenchimento de um campo do briefing - como objetivo de negócio, resultado esperado, descrição de público-alvo ou critérios de exclusão - o serviço aciona um agente construído com LangGraph que executa duas etapas sequenciais: primeiro, recupera do banco de dados as diretrizes e expectativas específicas daquele campo (`fetch_field_info`); em seguida, invoca um modelo de linguagem com um prompt estruturado que considera essas diretrizes e o contexto dos campos já preenchidos na campanha (`enhance_text`). O resultado é uma versão aprimorada do texto acompanhada de uma explicação das melhorias realizadas.

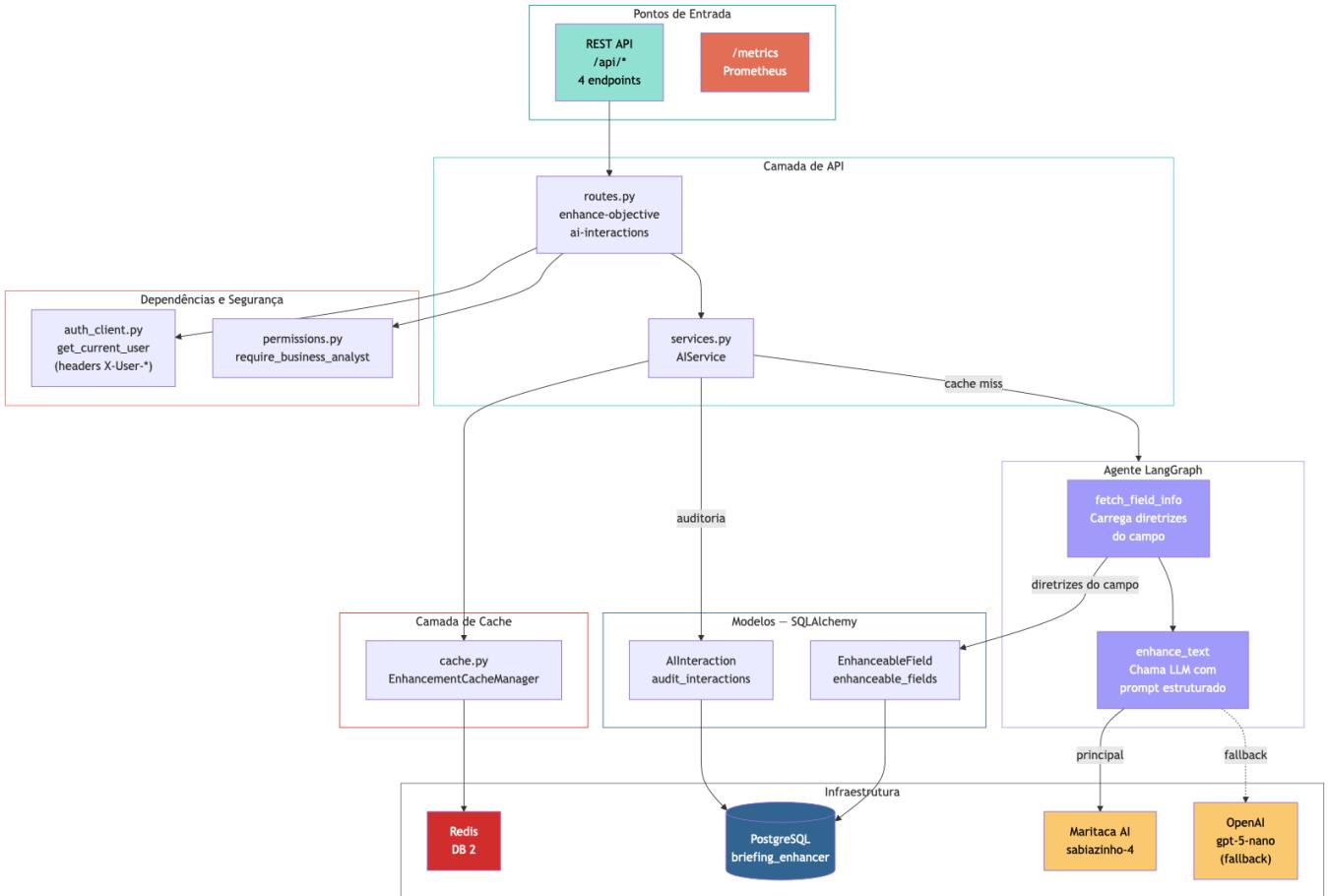
O modelo principal utilizado é o sabiazinho-4 da Maritaca AI, com o gpt-5-nano da OpenAI como fallback caso a chave da Maritaca não esteja configurada. As motivações por trás da escolha desses modelos está melhor descrita no capítulo [Evaluations para escolha dos modelos e parâmetros](#).

O serviço utiliza Redis como cache para evitar chamadas redundantes ao LLM. O PostgreSQL mantém um registro permanente de auditoria na tabela `audit_interactions`, incluindo o modelo LLM utilizado, o texto original, o texto aprimorado, a explicação e a decisão do usuário (aceitar ou rejeitar a sugestão). Essa decisão é registrada posteriormente via endpoint dedicado e, quando o usuário rejeita, a entrada correspondente no Redis é invalidada para garantir que uma nova sugestão seja gerada na próxima solicitação.

O agente inclui uma camada de moderação de conteúdo via OpenAIModerationMiddleware, utilizando o modelo `omni-moderation-latest` da OpenAI. A moderação é aplicada ao texto de entrada antes do envio ao LLM. Quando o conteúdo viola as políticas de uso, a requisição é rejeitada com uma mensagem descrevendo as categorias de violação, sem que o LLM seja invocado. A configuração da moderação é externalizada no `models.yaml` e as rejeições são contabilizadas via métricas Prometheus.

O acesso ao serviço é restrito ao perfil de Analista de Negócios, validado a partir dos headers `X-User-*` injetados pelo API Gateway. O serviço também utiliza checkpointing do LangGraph em PostgreSQL para manter o estado das conversações, e expõe métricas para o Prometheus que incluem contadores de aprimoramentos, latência de chamadas ao LLM e rejeições do middleware de moderação da OpenAI.

Arquitetura do serviço



Endpoints

<http://localhost:8001/docs>

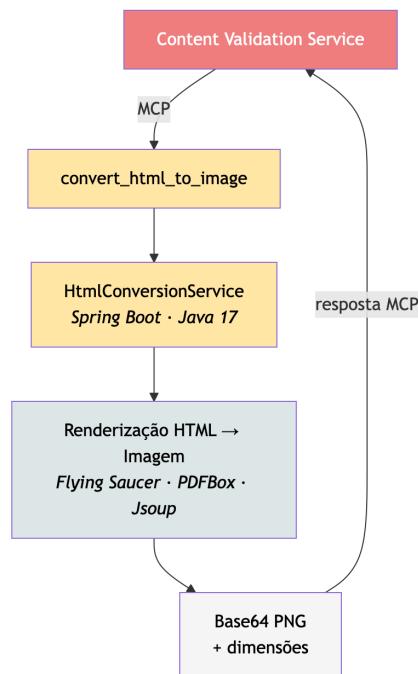
- GET /
 - Informações do serviço (nome, versão, status)
- GET /api/health
 - Health check
- POST /api/enhance-objective
 - Aprimora o texto de um campo do briefing com IA
- PATCH /api/ai-interactions/{interaction_id}/decision
 - Registra a decisão do usuário (approved/rejected) sobre uma sugestão da IA

HTML Converter Service

O HTML Converter Service é um serviço construído em Java com Spring Boot, responsável por converter conteúdo HTML de peças criativas de e-mail em imagens Base64. O serviço é utilizado como uma tool pelo Content Validation Service durante a validação de peças do canal E-mail: o HTML é convertido em imagem para que o Legal Service possa analisar visualmente o conteúdo renderizado, em complemento aos elementos textuais da peça.

O serviço expõe a tool convert_html_to_image por meio de um servidor MCP implementado com o SDK Java do Model Context Protocol, utilizando transporte SSE (Server-Sent Events) sobre o endpoint /mcp/message. A tool aceita três parâmetros: htmlContent (obrigatório), scale (fator de escala, padrão 0.5) e imageFormat (PNG ou JPEG, padrão PNG). O retorno é um JSON contendo a imagem convertida em base64 junto com suas dimensões em pixels.

Arquitetura do serviço



Endpoints

<http://localhost:8005/docs>

- POST /api/v1/html-to-image/convert
 - Recebe HTML, escala e formato. Retorna a imagem em Base64 com dimensões.
- GET /api/v1/html-to-image/health
 - Health check do serviço.

Tools MCP

- MCP /mcp/message
 - Tool convert_html_to_image com parâmetros htmlContent (obrigatório), scale (padrão 0.5) e imageFormat (padrão PNG).

Branding Service

O branding-service é um microsserviço de validação determinística de marca, responsável por garantir que todas as peças criativas (e-mails HTML e imagens do App) estejam em conformidade com as diretrizes visuais da marca Orquestra. Ele não utiliza IA/LLM, todas as validações são baseadas em regras fixas codificadas diretamente no serviço. O serviço é exposto exclusivamente via MCP (Model Context Protocol), o que permite que agentes de IA (como o do content-validation-service) consumam suas ferramentas de forma nativa.

O serviço possui duas "linhas" de validação distintas:

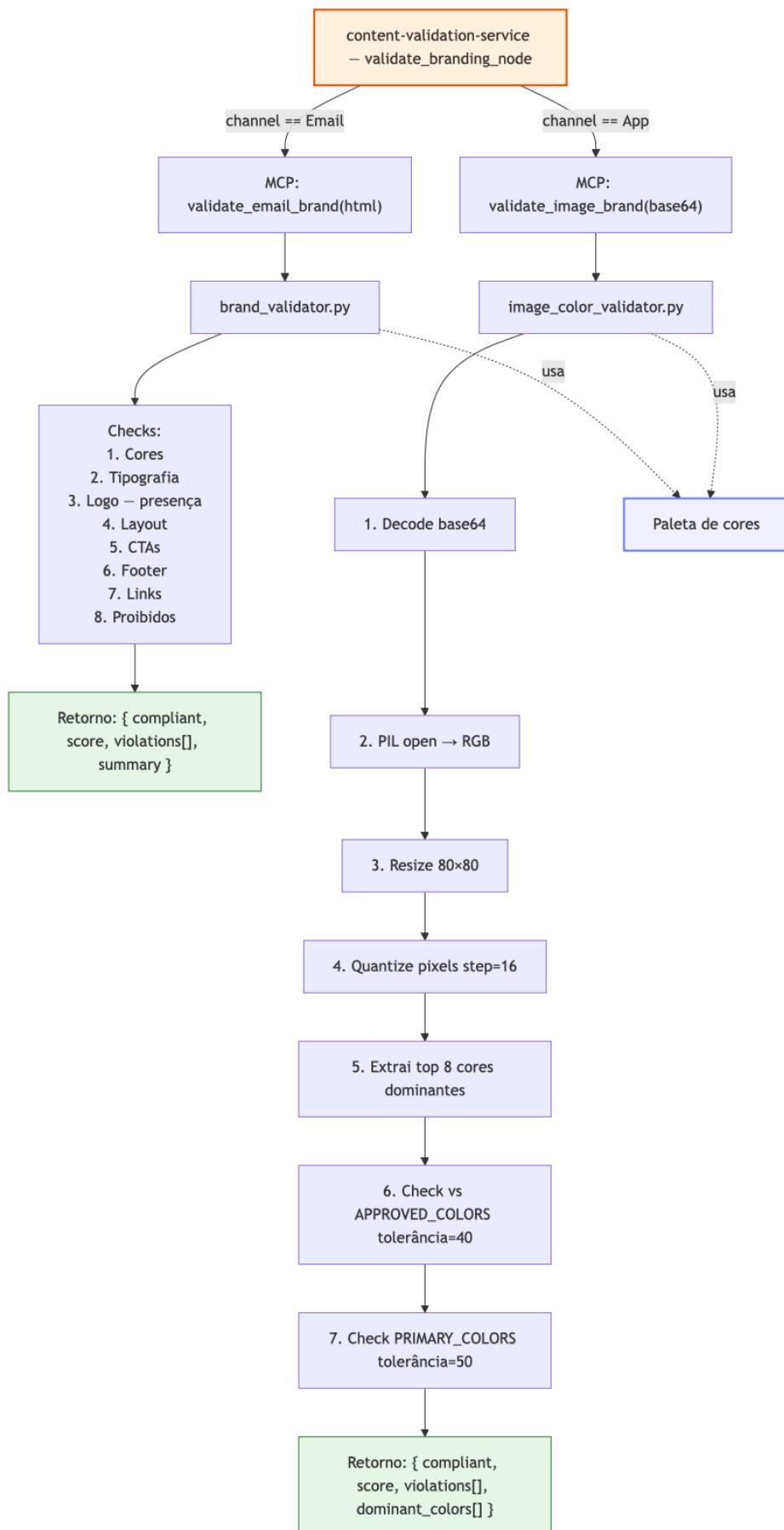
- Validação de HTML (E-mail): Analisa o HTML completo via BeautifulSoup, verificando cores, tipografia, logo, layout, CTAs, footer, links e elementos proibidos.
- Validação de Imagem (App/banners): Recebe a imagem em base64, extrai as cores dominantes via quantização de pixels com PIL, e verifica conformidade com a paleta de cores aprovada.

O resultado de toda validação segue a mesma estrutura: um score de 0 a 100 (onde cada violação crítica desconta 20 pontos e cada aviso desconta 5), uma flag compliant (true se não houver nenhuma violação crítica ou aviso), e a lista detalhada de violações.

Tools MCP

- validate_email_brand(html: str)
 - Valida HTML de e-mail contra as diretrizes da marca.
- validate_image_brand(image: str)
 - Valida cores dominantes de uma imagem contra a paleta da marca.
- get_brand_guidelines()
 - Retorna as diretrizes completas de marca da Orquestra. Derivado automaticamente das mesmas constantes usadas nas validações, garantindo que guidelines e regras estejam sempre sincronizadas. Retorna cores, tipografia, logo, layout, CTAs, footer, links e elementos proibidos.

Arquitetura do serviço



Legal Service (Agente)

O Legal Service é o agente responsável pela validação regulatória das peças criativas de CRM. O serviço implementa um pipeline de RAG (Retrieval-Augmented Generation) que recupera trechos de documentos regulatórios da empresa armazenados em uma base vetorial Weaviate, e os utiliza como contexto para que um modelo de linguagem avalie a conformidade do conteúdo submetido.

O grafo LangGraph é composto por dois nós sequenciais: retrieve, que executa uma busca híbrida (vetorial + BM25) com filtragem por canal, e generate, que invoca o LLM com os chunks recuperados e o conteúdo da peça para emitir um veredito estruturado (APROVADO ou REPROVADO) acompanhado de justificativa e fontes.

A seleção do modelo de linguagem é feita por canal: o sabiazinho-4 da Maritaca AI atende SMS, Push e E-mail, enquanto o gpt-5-nano da OpenAI é utilizado para os canais App e E-mail com imagem, por sua capacidade de processar entradas visuais. Os embeddings para a busca vetorial são gerados pelo text-embedding-3-small da OpenAI, e o Weaviate está configurado com reranking via Cohere, embora essa funcionalidade esteja desabilitada por padrão. A escolha dos modelos e demais parâmetros está melhor justificada no capítulo [Evaluations para escolha dos modelos e parâmetros](#).

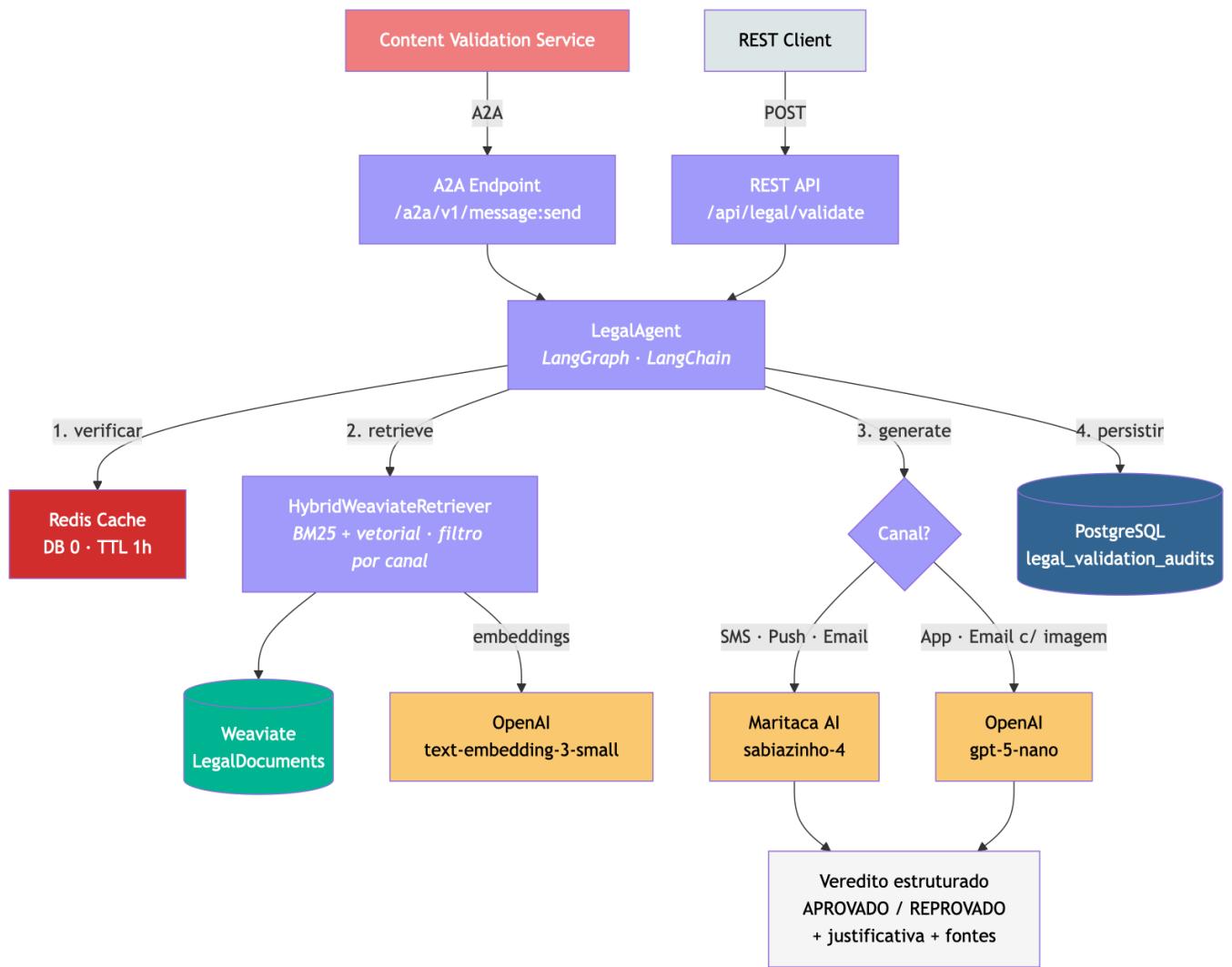
O serviço expõe duas interfaces de comunicação: uma API REST convencional e um servidor A2A (Agent-to-Agent), sendo este último o protocolo utilizado pelo Content Validation Service para solicitar validações. Os resultados são armazenados em cache no Redis para evitar chamadas redundantes ao LLM, e cada validação é registrada na tabela legal_validation_audits do PostgreSQL, incluindo a decisão, o resumo, as fontes consultadas, o LLM utilizado e a quantidade de chunks recuperados.

Endpoints

<http://localhost:8005/docs>

- GET /a2a/.well-known/agent-card.json
 - Card de descoberta do agente no protocolo A2A.
- POST /a2a/v1/message:send
 - Endpoint A2A para receber solicitações de validação do Content Validation Service.
- POST /api/legal/validate
 - Valida uma peça criativa contra as diretrizes regulatórias via RAG + LLM. Aceita conteúdo de SMS, Push, E-mail (HTML e/ou imagem) e App (imagem).
- GET /api/legal/health
 - Health check.

Arquitetura do serviço



Content Validation Service (Orquestrador)

O Content Validation Service é o serviço responsável por orquestrar a validação completa de peças criativas no Orquestrador. Diferentemente do Legal Service e do Briefing Enhancer Service, que utilizam modelos de linguagem para executar suas tarefas, este serviço não possui LLM próprio. Sua função é coordenar a execução de validações distribuídas entre múltiplos serviços especializados, agregando os resultados em um veredito final.

O serviço foi implementado como um grafo de estados utilizando LangGraph. Embora o LangGraph seja comumente associado à construção de agentes de IA, sua abstração - nós, arestas condicionais e estado compartilhado - é agnóstica ao uso de LLM e se aplica a qualquer fluxo que exija roteamento condicional, paralelismo e convergência entre etapas assíncronas. No caso do Content Validation Service, essas três características são centrais: o serviço precisa disparar chamadas simultâneas a três serviços externos, aguardar a conclusão de todas antes de emitir o veredito e rotear o fluxo de forma diferente conforme o canal da peça. Implementar essa lógica de forma declarativa, em vez de encadear chamadas assíncronas manualmente, tornou o fluxo mais legível e rastreável, além de facilitar futuras evoluções para um cenário agêntico caso se torne necessário.

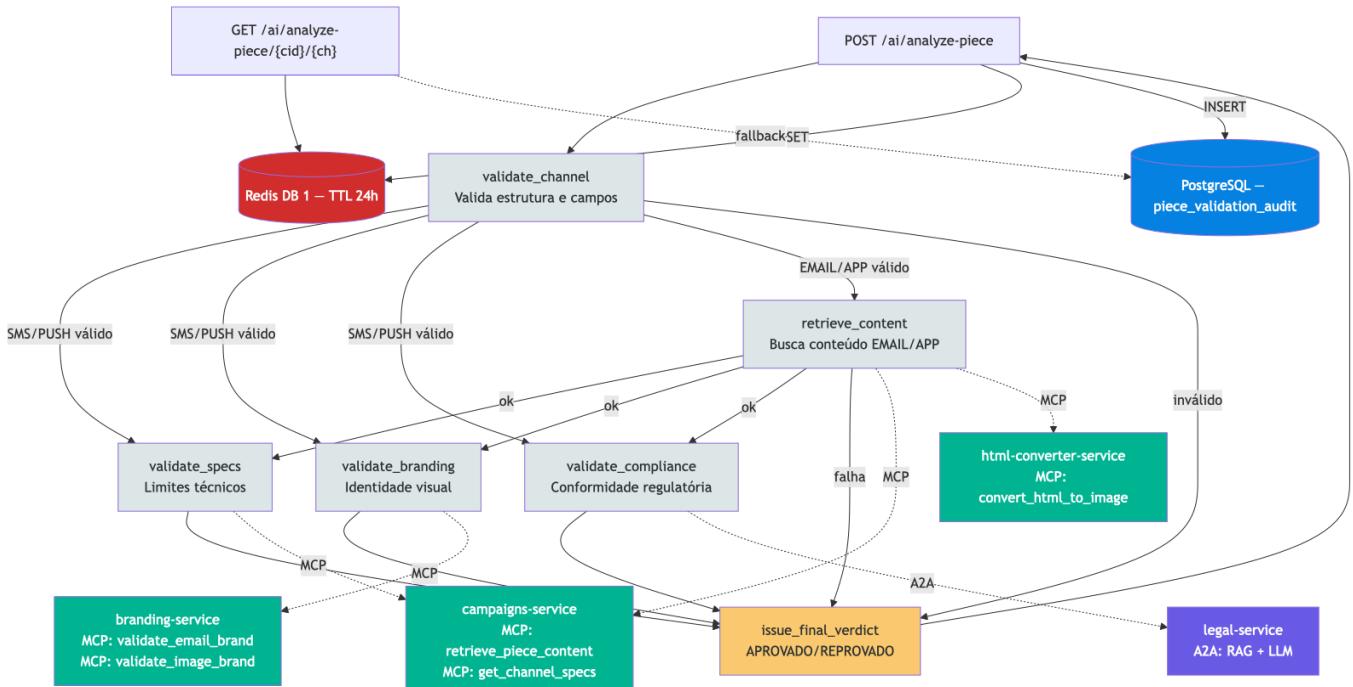
O Padrão BFA - Backend for Agents

O padrão Backend For Agents (BFA), proposto por Michael Douglas Barbosa Araujo, é um design pattern arquitetural inspirado no Backend For Frontend (BFF). Enquanto o BFF resolve o problema de acoplamento entre frontends e backends monolíticos ao introduzir uma camada intermediária dedicada a cada interface de usuário, o BFA aplica a mesma lógica ao ecossistema de agentes de inteligência artificial.

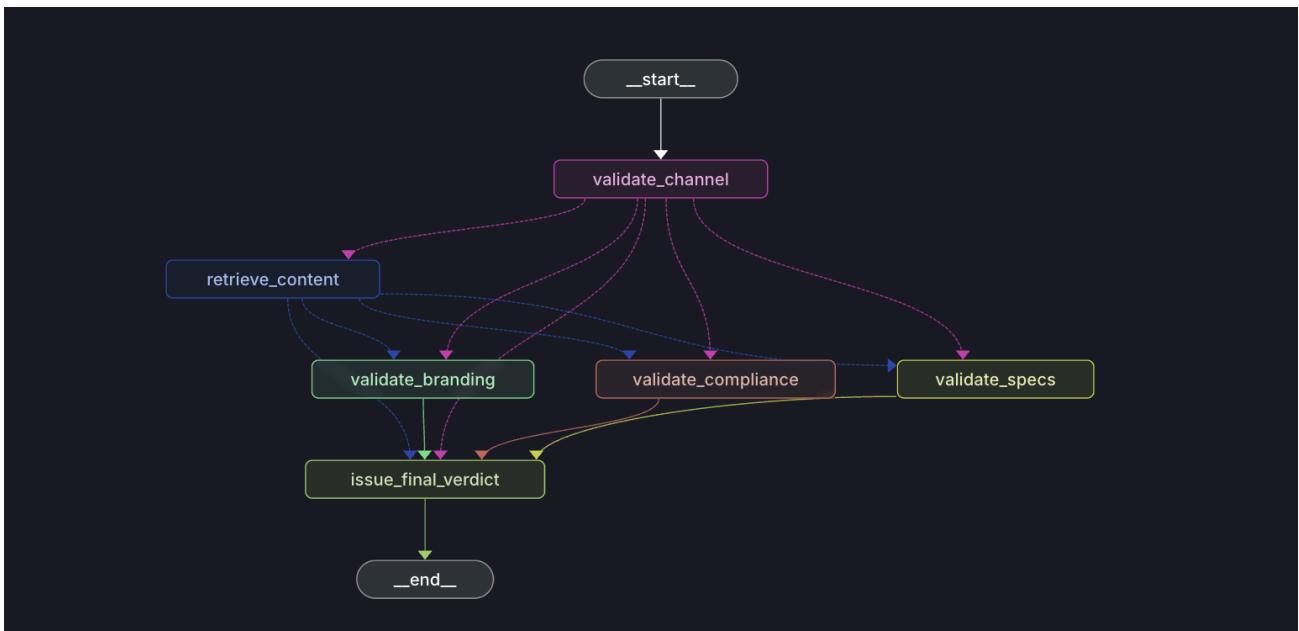
A premissa central é que agentes de IA, quando se acoplam diretamente a APIs de domínio e ferramentas externas, tendem a duplicar lógica, dificultar o reuso e fragmentar a observabilidade. A solução proposta pelo BFA consiste em introduzir uma camada mediadora entre o agente e os serviços de backend. Essa camada encapsula as chamadas a APIs de domínio, aplica políticas de autenticação, caching e logging, e expõe operações padronizadas por meio do protocolo MCP (Model Context Protocol). Dessa forma, o agente não precisa conhecer os detalhes de implementação dos serviços que consome; ele invoca ferramentas de alto nível expostas pelo BFA, como "validar conformidade de marca" ou "obter especificações do canal", sem saber exatamente o que há por trás.

O serviço de validação de conteúdo adota uma arquitetura que se inspira levemente nesse padrão. Implementado como um grafo de estados com LangGraph, o agente de validação não acessa diretamente os serviços de domínio. Em vez disso, ele consome três servidores MCP especializados: o branding-service, que encapsula regras determinísticas de identidade visual da marca; o html-converter-service, responsável pela conversão de HTML em imagem; e o campaigns-service, que expõe ferramentas para recuperação de conteúdo das peças criativas e especificações técnicas de cada canal. Cada um desses serviços atua como um BFA dedicado a um domínio específico, expondo contratos estáveis via MCP e isolando o agente da complexidade "por baixo dos panos".

Arquitetura do serviço



Fluxo de funcionamento



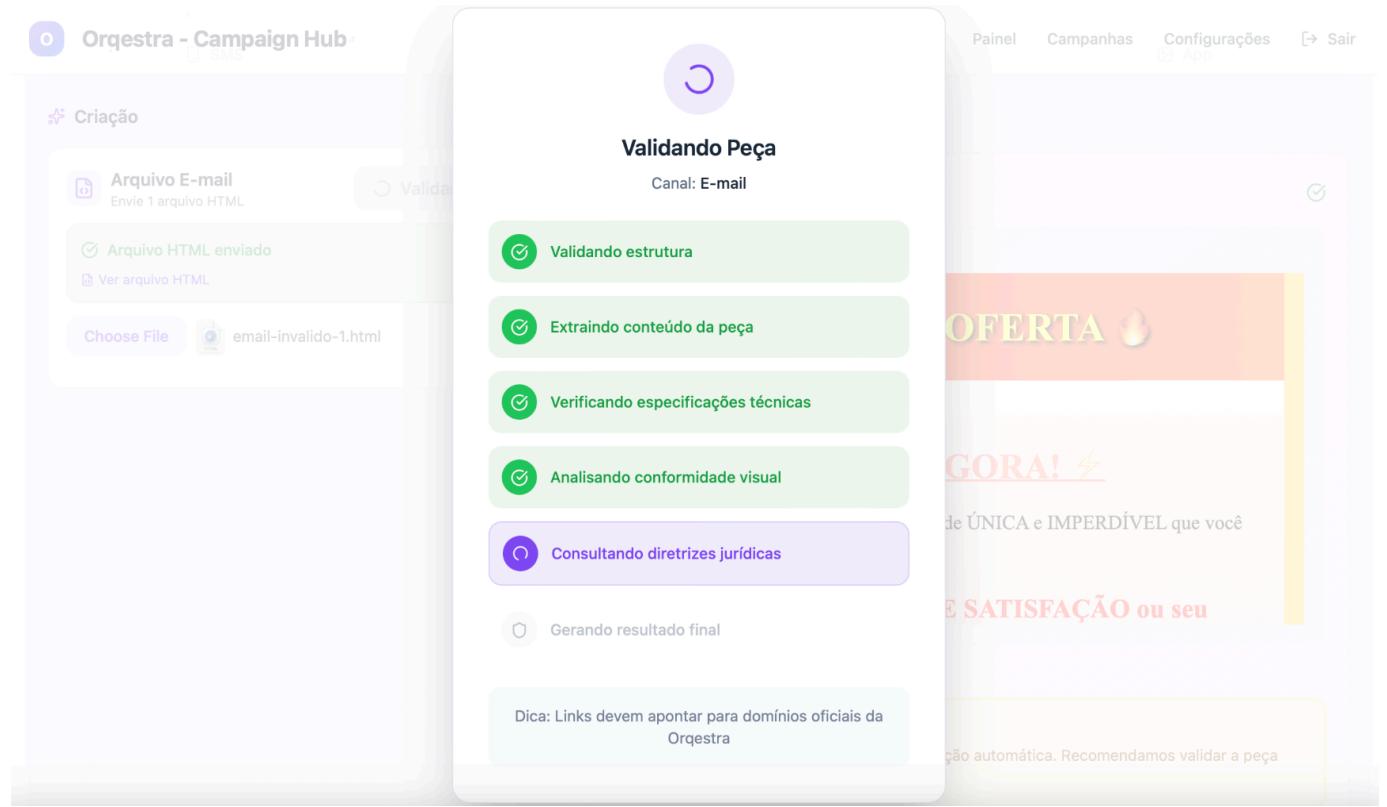
O grafo se inicia pelo nó `validate_channel`, que verifica a estrutura e os campos obrigatórios da peça conforme o canal. Para SMS e PUSH, cujo conteúdo já vem no payload da requisição, o fluxo dispara imediatamente três nós em paralelo: `validate_specs` (limites técnicos como caracteres e tamanho), `validate_branding` (identidade visual da marca) e `validate_compliance` (conformidade regulatória via legal-service). Para EMAIL e APP, cujo conteúdo está armazenado no S3, o fluxo passa primeiro pelo nó `retrieve_content`, que busca o arquivo via MCP no campaigns-service e, no caso de email, converte o HTML em imagem via MCP no `html-converter-service`. Depois disso, os três nós paralelos são disparados.

Os três nós paralelos consomem serviços externos distintos: validate_specs consulta as especificações técnicas do canal via MCP no campaigns-service, validate_branding valida cores, fontes e layout via MCP no branding-service e validate_compliance envia o conteúdo ao legal-service para análise regulatória com RAG e LLM. Quando os três terminam, o grafo aciona o nó issue_final_verdict, que agrupa os resultados e emite um veredito de aprovação.

Nas primeiras versões do serviço, foi considerada a execução dos nós determinísticos em primeiro lugar (validação de especificações de canal e branding) numa estratégia de fail fast para economia de tokens. A lógica era de que se uma peça já está reprovada, não seria necessário submetê-la à validação de compliance/legal, que é mais custosa por utilizar LLMs.

Contudo, pensando no cenário real de uma operação de campanhas de CRM, onde as peças precisam retornar para correção do time criativo interno ou de agências externas, é importante apontar todos os aspectos de correção o quanto antes, o que economiza tempo e evita idas e vindas desnecessárias. Os custos dessas ineficiências operacionais são frequentemente maiores que as possíveis chamadas extras a LLMs.

Streaming de progresso na validação de peças



Uma das limitações identificadas na experiência do usuário durante a validação de peças dos canais era a ausência de feedback real em tela sobre o progresso da análise. Por se tratar de um fluxo que envolve múltiplas etapas, o tempo de resposta pode levar alguns segundos, sendo mais demorado para E-mail e App.

A solução inicialmente adotada simulava a progressão dos passos por meio de intervalos fixos no frontend, sem qualquer correspondência com a execução real no backend. Essa abordagem, embora funcional do ponto de vista visual, apresentava uma desconexão entre o que era exibido ao usuário e o estado real do processamento. Para resolver essa limitação, foi implementado um mecanismo de streaming baseado em Server-Sent Events (SSE), permitindo que cada etapa do grafo de validação

emita eventos de progresso em tempo real, consumidos pelo frontend à medida que ocorrem. A implementação se distribui em três camadas: o agente de validação (backend), o API Gateway (proxy) e a interface do usuário (frontend).

Para viabilizar o streaming de progresso, cada nó foi instrumentado com `get_stream_writer()` do LangChain, que permite a emissão de eventos personalizados durante a execução do grafo sem alterar seu fluxo de dados. Ao iniciar, cada nó emite um evento com status “started” e um rótulo descritivo. Ao concluir, emite um evento com status “done”. Na classe `ContentValidationAgent`, foi adicionado o método `astream_with_progress`, que consome o grafo utilizando `stream_mode=["custom", "values"]`. Esse modo permite receber simultaneamente os eventos personalizados emitidos pelos nós (via `get_stream_writer`) e os snapshots de estado do grafo. O método itera sobre os chunks retornados e os classifica por tipo, emitindo-os como dicionários tipados:

```
async def astream_with_progress(self, task=None, channel=None, content=None):
    async for mode, chunk in self.app.astream(
        initial, config=langsmith_config, stream_mode=["custom", "values"]
    ):
        if mode == "custom":
            yield {"type": "step", "data": chunk}
        elif mode == "values":
            final_state = dict(chunk)

    yield {"type": "result", "data": final_state}
```

Um novo endpoint POST `/ai/analyze-piece/stream` foi criado no `content-validation-service`, retornando uma `StreamingResponse` com media type `text/event-stream`. Os eventos são formatados segundo o protocolo SSE, com campos `event` e `data` separados por quebras de linha duplas. O resultado final da análise é emitido como um evento do tipo `result`, acompanhado da persistência em cache (Redis) e da auditoria em banco de dados, garantindo que o endpoint de streaming mantenha as mesmas garantias do endpoint síncrono. No API Gateway, foi implementada uma função de proxy específica para streaming (`proxy_request_stream`), que utiliza `httpx.AsyncClient().stream()` para consumir a resposta do serviço downstream e repassar os chunks ao cliente sem buffering:

```
async def stream_generator():
    async with httpx.AsyncClient(timeout=180.0) as client:
        async with client.stream("POST", url, headers=proxy_headers, content=body) as response:
            async for chunk in response.aiter_bytes():
                yield chunk

    return StreamingResponse(stream_generator(), media_type="text/event-stream",
                           headers={"Cache-Control": "no-cache", "X-Accel-Buffering": "no"})
```

No cliente frontend, foi criado o método `analyzePieceStream`, que realiza a requisição via `fetch` e consome o corpo da resposta como um `ReadableStream`. O parser SSE implementado acumula linhas em um buffer, identifica o tipo de evento (`step`, `result` ou `error`) e invoca o callback `onStep` a cada evento de progresso recebido:

```
const reader = response.body!.getReader();
const decoder = new TextDecoder();
let buffer = "";

while (true) {
```

```
const { done, value } = await reader.read();
if (done) break;
buffer += decoder.decode(value, { stream: true });
if (currentEvent === "step") {
  onStep(parsed as ValidationStepEvent);
} else if (currentEvent === "result") {
  result = parsed;
}
}
```

Com essa implementação, o Orquestra oferece visibilidade total sobre o pipeline de validação em tempo real, sem comprometer a arquitetura existente: o endpoint síncrono permanece ativo, o fluxo de SMS e Push não foi alterado, e a infraestrutura de cache e auditoria é compartilhada por ambos os endpoints.

Endpoints

<http://localhost:8004/docs>

- GET /health
 - Health check do serviço
- POST /ai/analyze-piece
 - Executa a validação completa da peça (dispara o grafo LangGraph)
- GET /ai/analyze-piece/{campaign_id}/{channel}
 - Recupera o resultado da última validação (Redis, fallback PostgreSQL)
- GET /a2a/.well-known/agent-card.json
 - A2A Agent card
- POST /a2a/v1/message:send
 - A2A Endpoint para receber requisições de outros agentes

Evaluations para escolha dos modelos e parâmetros

Porque o sistema foi pensado para ser usado em português brasileiro somente, ao procurar LLMs com suporte ou especializados no idioma, surgiram os modelos da empresa Maritaca AI.

A Maritaca AI é uma empresa brasileira dedicada à pesquisa e ao desenvolvimento de modelos de linguagem natural, com foco especial no português. A organização atua na criação de soluções de inteligência artificial voltadas à compreensão e geração de texto, buscando equilibrar rigor técnico, aplicabilidade prática e responsabilidade no uso da tecnologia.

O **sabiazinho-4** foi lançado recentemente, em janeiro de 2026, e é apresentado como a primeira instância da quarta geração da família “Sabiá”, projetado com ênfase em eficiência de custo e latência, além de melhor aderência a tarefas complexas em português brasileiro. Em relação a versões anteriores, foram ampliadas tanto a janela de contexto quanto a capacidade de seguimento de instruções complexas e integração com agentes, o que amplia sua aplicabilidade em fluxos estruturados de trabalho. O modelo apresenta avanços notáveis em *benchmarks* nacionais, tais como OAB-Bench (avaliação de escrita jurídica) e testes de compreensão legislativa, demonstrando desempenho competitivo inclusive com reduções de custo e melhorias de velocidade quando comparado a outras soluções de mercado. Esses atributos o situam como um recurso relevante para aplicações que exigem processamento linguístico avançado em contextos profissionais e institucionais no Brasil.

Paralelamente, foi avaliado o uso do **gpt-5-nano**, atualmente o modelo de menor custo disponibilizado pela OpenAI. Em comparação ao sabiazinho-4, o gpt-5-nano apresenta uma janela de contexto maior e suporte explícito a mecanismos de *reasoning*, características que podem torná-lo mais adequado para determinadas tarefas de maior complexidade. Adicionalmente, o modelo oferece suporte a múltiplos formatos de entrada além de texto, como imagens e vídeos, e possui um custo aproximadamente inferior ao do sabiazinho-4, o que o torna atraente do ponto de vista econômico.

Entretanto, um efeito colateral observado durante as experimentações é que, à medida que aumenta o nível de *reasoning* exigido pelo caso de uso, os tempos de resposta do gpt-5-nano tendem a se elevar de forma perceptível. Esse comportamento impacta negativamente a experiência do usuário em cenários interativos, especialmente em fluxos nos quais a latência é um fator relevante para a usabilidade do sistema.

Diante dessas características, foram conduzidas avaliações comparativas com o objetivo de definir, de forma mais pragmática, a adequação de cada modelo aos diferentes componentes do sistema, bem como identificar cenários em que eles pudessem ser usados de forma combinada.

Briefing Enhancer Service

- Path /briefing-enhancer-service/evals

O módulo de avaliação do Briefing Enhancer Service implementa uma metodologia de comparação entre modelos de linguagem baseada na abordagem “*LLM as a judge*”, com o objetivo de determinar qual modelo produz aprimoramentos de maior qualidade para textos de *briefing* de campanhas de CRM. O processo de avaliação opera sobre um *dataset* composto por 80 registros, correspondentes a 20 campanhas fictícias, cada uma com quatro campos de briefing: objetivo de negócio, resultado esperado, descrição do público-alvo e critérios de exclusão. Os textos originais simulam o preenchimento realizado

por um analista de negócios humano, sendo contextualmente adequados ao tema de cada campanha, porém deliberadamente vagos, genéricos e carentes de especificidade.

Para cada registro do dataset, o script submete o texto original aos dois modelos competidores utilizando o mesmo pipeline de aprimoramento do serviço em produção, incluindo a consulta às diretrizes armazenadas em banco de dados para cada tipo de campo. Os dois textos aprimorados resultantes são então submetidos a um modelo juiz (GPT-5.2 da OpenAI), que atua como avaliador imparcial. O juiz recebe como contexto as diretrizes oficiais do campo (expectativas e orientações de melhoria), o texto original como referência e os dois textos aprimorados. Com base nesses elementos, o juiz elege um vencedor, atribui uma pontuação de 1 a 5 para cada texto e fornece uma justificativa concisa para sua decisão. Adicionalmente, o tempo de resposta de cada modelo é registrado individualmente para cada invocação, permitindo a análise comparativa de latência como fator complementar à qualidade textual na decisão final.

Avaliação #1

O primeiro experimento avaliou 80 linhas do *dataset*, porém apenas 68 foram efetivamente julgadas. As 12 linhas restantes foram classificadas como erro, todos exclusivamente por *timeout* do gpt-5-nano, que não foi capaz de retornar uma resposta dentro dos 20 segundos estabelecidos.

```
=====
AVALIAÇÃO – Briefing Enhancer: sabiazinho-4 vs gpt-5-nano
=====

Dataset:      evals/eval_dataset.csv
Total:        80 linhas
Juiz:         gpt-5.2
Duração:     2827s

-----
sabiazinho-4 vence: 26/68 (38.2%)
gpt-5-nano vence:   40/68 (58.8%)
Empates:          2/68 (2.9%)
Erros:            12

Score médio sabiazinho-4: 3.5
Score médio gpt-5-nano:   3.79
Diferença:        +0.29 para gpt-5-nano

-----
POR CAMPO:
    Objetivo de Negócio:
        sabiazinho-4: 10/20 (50.0%) score=3.55
        gpt-5-nano:   10/20 (50.0%) score=3.6
        empates:     0/20
    Critérios de Exclusão:
        sabiazinho-4: 9/16 (56.2%) score=4.06
        gpt-5-nano:   5/16 (31.2%) score=3.56
        empates:     2/16
    Resultado Esperado / KPI Principal:
        sabiazinho-4: 0/19 (0.0%) score=2.79
        gpt-5-nano:   19/19 (100.0%) score=4.32
        empates:     0/19
    Descrição do Público-Alvo:
        sabiazinho-4: 7/13 (53.8%) score=3.77
        gpt-5-nano:   6/13 (46.2%) score=3.62
        empates:     0/13

=====
```

Considerando apenas as 68 linhas válidas, o gpt-5-nano venceu 40 comparações (58,8%) contra 26 vitórias do sabiazinho-4 (38,2%) e 2 empates (2,9%). O score médio do gpt-5-nano foi de 3,79 contra 3,50 do sabiazinho-4, uma diferença de 0,29 a favor do modelo da OpenAI. No entanto, os 12 erros concentraram-se desproporcionalmente em determinados campos e campanhas, eliminando da amostra justamente os casos em que o gpt-5-nano falhou.

Olhando os números por campo, o menos equilibrado foi “Resultado Esperado” (*expectedResult*), onde houve domínio absoluto do gpt-5-nano. Para este campo, o sabiazinho-4 frequentemente substituiu

valores numéricos presentes no texto original por *placeholders* genéricos (por exemplo, trocando "5000 contas" por "[X] contas" ou "R\$ 100 milhões" por "[X] milhões"), o que foi severamente penalizado pelo juiz por reduzir a especificidade e a fidelidade ao briefing. O gpt-5-nano, em contraste, preservou os valores originais e adicionou métricas complementares (taxa de conversão, prazo, custo), tornando os KPIs mais acionáveis sem perder informação.

Ainda que a latência não tenha sido levada em consideração na análise do juiz, o tempo de resposta do gpt-5-nano foi consistentemente maior em todas as interações. Esse é um fator importante, pois a experiência do usuário de aprimorar o preenchimento do *briefing* é síncrona, e a demora de 10 a 20 segundos para uma tarefa simples prejudica essa experiência.

Avaliação #2

Para o segundo experimento, foram introduzidas as modificações:

- Diminuição do parâmetro de *reasoning* do gpt-5-nano para mínimo, a fim de diminuir os tempos de resposta e avaliar o impacto na qualidade;
- Ajuste do *prompt* explicitando que valores numéricos do texto original devem ser mantidos e o uso de *placeholders* é apenas para valores que não foram informados;
- Coleta dos tempos de resposta.

```

=====
AVALIAÇÃO - Briefing Enhancer: sabiazinho-4 vs gpt-5-nano
=====
Dataset:      evals/eval_dataset.csv
Total:        80 linhas
Juiz:         gpt-5.2
Duração:     752s
-----
sabiazinho-4 vence: 55/80 (68.8%)
gpt-5-nano vence:   25/80 (31.2%)
Empates:       0/80 (0.0%)
Erros:         0
-----
Score médio sabiazinho-4: 4.09
Score médio gpt-5-nano:   3.51
Diferença:    +0.58 para sabiazinho-4
-----
LATÊNCIA (segundos):
sabiazinho-4: avg=4.79 p50=4.5 p95=7.27
gpt-5-nano:   avg=2.17 p50=2.05 p95=3.16
→ gpt-5-nano é 2.2x mais rápido
-----
POR CAMPO:
Objetivo de Negócio:
sabiazinho-4: 11/20 (55.0%) score=4.2
gpt-5-nano:   9/20 (45.0%) score=4.05
empates:     0/20
Critérios de Exclusão:
sabiazinho-4: 17/20 (85.0%) score=4.25
gpt-5-nano:   3/20 (15.0%) score=3.05
empates:     0/20
Resultado Esperado / KPI Principal:
sabiazinho-4: 14/20 (70.0%) score=4.2
gpt-5-nano:   6/20 (30.0%) score=3.5
empates:     0/20
Descrição do Público-Alvo:
sabiazinho-4: 13/20 (65.0%) score=3.7
gpt-5-nano:   7/20 (35.0%) score=3.45
empates:     0/20
=====
```

As 80 linhas do *dataset* foram processadas com sucesso por ambos os modelos, eliminando os 12 erros de *timeout* que invalidaram parte da avaliação anterior.

O sabiazinho-4 venceu 55 das 80 comparações (68,8%), contra 25 vitórias do gpt-5-nano (31,2%) e nenhum empate. O score médio atribuído pelo juiz ao sabiazinho-4 foi de 4,09 contra 3,51 do gpt-5-nano, uma diferença de 0,58 ponto numa escala de 1 a 5. Estes números indicam uma superioridade consistente do modelo da Maritaca para a tarefa de aprimoramento de textos de briefing em português brasileiro.

Voltando a olhar para os números por campo, o maior domínio do sabiazinho-4 esteve no “Critérios de Exclusão” (*exclusionCriteria*). A análise qualitativa das justificativas do juiz revela que o sabiazinho-4 demonstrou maior fidelidade ao texto original, organizou consistentemente os critérios em formato de bullets conforme exigido pelas diretrizes, e incluiu os critérios obrigatórios (menores de 18 anos e restrições na base de Riscos) de forma clara e objetiva. O gpt-5-nano, por sua vez, tendeu a adicionar exclusões especulativas não sustentadas pelo briefing original, o que foi penalizado pelo juiz por reduzir a aderência e a precisão.

Também houve uma boa recuperação do sabiazinho-4 no preenchimento do campo “Resultado Esperado” (*expectedResult*), com 70% de vitórias. Aqui ele foi mais consistente em preservar os valores numéricos presentes no texto original (metas, prazos, volumes) e adicionar métricas complementares.

Nos parâmetros atuais, o sabiazinho-4 demonstrou superioridade para a tarefa de aprimoramento de textos de briefing em português brasileiro, com vantagem em qualidade avaliada pelo juiz em todos os

quatro campos. Sua principal limitação é a latência, sendo cerca de 2 vezes mais lento que o concorrente. O gpt-5-nano oferece velocidade de resposta superior quando o *reasoning* está setado para “minimal”, e um desempenho um pouco pior no aprimoramento dos textos.

Resultados

A partir dos resultados observados, foi decidido como abordagem definitiva que o Briefing Enhancer Service utilize o sabiazinho-4 como seu modelo principal e o gpt-5-nano como *fallback*.

Legal Service

- **Path /legal-service/evals**

A necessidade do Legal Service é diferente da do Briefing Enhancer. Em vez de apenas aprimorar textos escritos por humanos, esse agente tem como objetivo avaliar peças criativas e decidir pela sua aprovação ou reprovação antes que sejam exibidas ao cliente final. Aqui, os tempos de resposta mais altos podem ser tolerados pela criticidade da decisão, e os impactos na experiência do usuário devem ser mitigados de outras formas.

A introdução do RAG como solução de recuperação de contexto relevante também adiciona novas dúvidas ao desenvolvimento, como a melhor estratégia de *chunking* dos documentos e necessidade de *reranking*.

Avaliação

O dataset de avaliação consiste em 80 comunicações fictícias de CRM, cada uma com uma decisão esperada definida manualmente. As comunicações incluem exemplos propositalmente construídos para testar alguns cenários: mensagens com linguagem promocional vedada, uso indevido de dados pessoais, ausência de mecanismo de *opt-out*, *links* encurtados suspeitos, promessas absolutas de resultado, e também alguns exemplos conformes que devem ser aprovados.

Uma característica central da avaliação é o sistema de experimentos parametrizável, configurado por meio de um arquivo YAML. Cada experimento combina quatro variáveis:

- **Estratégia de *chunking* dos documentos regulatórios:** por seção estrutural dos documentos ou por segmentação semântica, cada uma armazenada em uma *collection* distinta na base vetorial;
- **Estratégia de *retrieval*:** busca híbrida com alpha=0.5, busca puramente por palavras-chave com BM25 e alpha=0.0, ou busca puramente vetorial com alpha=1.0;
- **Modelo de linguagem:** sabiazinho-4 da Maritaca AI ou gpt-5-nano da OpenAI;
- **Habilitação do *reranking*.**

As diferentes combinações entre esses fatores produziu 24 experimentos que mediram as métricas:

- **Accuracy (acurácia):** proporção de classificações corretas sobre o total de exemplos. No contexto deste trabalho, indica a fração das 80 peças de comunicação para as quais o agente emitiu a decisão correta (APROVADO ou REPROVADO). É uma métrica intuitiva, mas afetada

pelo desbalanceamento entre classes. Por exemplo, um classificador que reprovasse tudo atingiria 63,75% de acurácia neste dataset.

- **Precision (precisão):** entre todas as peças que o agente classificou como pertencentes a uma determinada classe, quantas de fato pertencem a ela. Por exemplo, uma precision de 96% para a classe REPROVADO significa que, de cada 100 peças reprovadas pelo agente, aproximadamente 96 eram realmente irregulares e 4 eram peças legítimas rejeitadas indevidamente.
- **Recall (recuperação):** entre todas as peças que realmente pertencem a uma classe, quantas o agente identificou corretamente. Por exemplo, um recall de 98% para a classe REPROVADO significa que, das 51 peças efetivamente irregulares no dataset, o agente detectou cerca de 50 e deixou aproximadamente 1 escapar como falso negativo. Esta foi a métrica principal adotada neste trabalho, dado que o custo de aprovar conteúdo irregular supera o custo de rejeitar conteúdo legítimo.
- **F1-score:** média harmônica entre precision e recall. Sintetiza o equilíbrio entre os dois em um único valor. Um F1 alto exige que tanto precision quanto recall sejam altos; se um dos dois for baixo, o F1 é penalizado de forma mais severa do que na média aritmética.

Resultados

RELATÓRIO COMPARATIVO DE EXPERIMENTOS										
#	Experimento	Chunking	Retrieval	Rerank	LLM	Recall	REP	F1 REP	Accuracy	Latência
1.	Sem + Hybrid + Maritaca (sem Rerank)	semantic	hybrid	Não	maritaca	98.04%	96.15%	95.00%	4.4s	
2.	Sem + Vector + GPT (sem Rerank)	semantic	semantic	Não	gpt_nano	98.04%	95.24%	93.75%	4.1s	
3.	Sem + BM25 + Maritaca (sem Rerank)	semantic	bm25	Não	maritaca	98.04%	95.24%	93.75%	4.3s	
4.	Sem + BM25 + GPT (sem Rerank)	semantic	bm25	Não	gpt_nano	98.04%	95.24%	93.75%	4.3s	
5.	Sem + Vector + Maritaca (sem Rerank)	semantic	semantic	Não	maritaca	98.04%	94.34%	92.50%	4.4s	
6.	Sem + Hybrid + GPT (sem Rerank)	semantic	hybrid	Não	gpt_nano	98.04%	94.34%	92.50%	4.5s	
7.	Sem + BM25 + GPT + Rerank	semantic	bm25	Sim	gpt_nano	96.08%	95.15%	93.75%	4.4s	
8.	Sem + Hybrid + Maritaca + Rerank	semantic	hybrid	Sim	maritaca	96.08%	94.23%	92.50%	5.2s	
9.	Sem + Vector + GPT + Rerank	semantic	semantic	Sim	gpt_nano	94.12%	94.12%	92.50%	4.2s	
10.	Sem + Hybrid + GPT + Rerank	semantic	hybrid	Sim	gpt_nano	94.12%	94.12%	92.50%	4.3s	
11.	Sem + Vector + Maritaca + Rerank	semantic	semantic	Sim	maritaca	94.12%	94.12%	92.50%	4.3s	
12.	Sem + BM25 + Maritaca + Rerank	semantic	bm25	Sim	maritaca	94.12%	94.12%	92.50%	4.5s	
13.	Sec + Vector + Maritaca (sem Rerank)	section	semantic	Não	maritaca	92.16%	92.16%	90.00%	4.0s	
14.	Sec + BM25 + GPT (sem Rerank)	section	bm25	Não	gpt_nano	92.16%	92.16%	90.00%	4.1s	
15.	Sec + Hybrid + Maritaca (sem Rerank)	section	hybrid	Não	maritaca	92.16%	92.16%	90.00%	4.3s	
16.	Sec + Vector + GPT (sem Rerank)	section	semantic	Não	gpt_nano	92.16%	92.16%	90.00%	4.4s	
17.	Sec + Vector + GPT + Rerank	section	semantic	Sim	gpt_nano	92.16%	91.26%	88.75%	4.1s	
18.	Sec + Hybrid + GPT (sem Rerank)	section	hybrid	Não	gpt_nano	92.16%	91.26%	88.75%	4.2s	
19.	Sec + BM25 + Maritaca (sem Rerank)	section	bm25	Não	maritaca	92.16%	91.26%	88.75%	4.2s	
20.	Sec + Vector + Maritaca + Rerank	section	semantic	Sim	maritaca	92.16%	91.26%	88.75%	4.4s	
21.	Sec + Hybrid + Maritaca + Rerank	section	hybrid	Sim	maritaca	92.16%	91.26%	88.75%	4.7s	
22.	Sec + BM25 + Maritaca + Rerank	section	bm25	Sim	maritaca	92.16%	90.38%	87.50%	4.5s	
23.	Sec + Hybrid + GPT + Rerank	section	hybrid	Sim	gpt_nano	92.16%	89.52%	86.25%	4.8s	
24.	Sec + BM25 + GPT + Rerank	section	bm25	Sim	gpt_nano	90.20%	90.20%	87.50%	5.5s	

A estratégia de chunking demonstrou ser o fator dominante: todos os 12 experimentos baseados em chunking semântico superaram, sem exceção, os 12 experimentos baseados em chunking por seção na métrica de recall de REPROVADO.

O segundo achado diz respeito ao reranking. Dentro do grupo de chunking semântico, as 6 configurações sem reranking obtiveram recall superior às 6 configurações com reranking. O modelo utilizado foi o Cohere Rerank, que aparentemente não foi adequado para o caso de uso e estava deslocando chunks pertinentes para fora do contexto.

As variáveis da estratégia de retrieval e o modelo de linguagem não apresentaram diferenças significativas. As variações observadas na acurácia global entre busca híbrida, BM25 e vetorial, bem

como entre o sabiazinho-4 e o GPT-5-nano, decorreram de uma a duas classificações distintas em um universo de 80 exemplos, o que é insuficiente para afirmar sobre seus impactos reais.

Ainda assim, a configuração **chunking semântico + busca híbrida + sabiazinho-4 + reranking desabilitado** mostrou-se uma das mais adequadas para o caso de uso. O **gpt-5-nano** ficou como fallback quando uma chave da Maritaca AI não for informada, e como modelo principal para os canais de E-MAIL e APP por sua compatibilidade com imagens na entrada.

Observabilidade e dashboards

Traces dos agentes

A solução utiliza o **LangSmith** como camada de observabilidade dedicada aos agentes de IA, complementando o Prometheus/Grafana que monitora métricas de infraestrutura. Com a variável `LANGCHAIN_TRACING_V2` habilitada, o LangSmith captura automaticamente todos os traces gerados pelo LangChain e LangGraph (chamadas ao LLM, execução de nodes, transições condicionais e uso de tokens) sem necessidade de alteração no código dos agentes.

Cada serviço envia seus traces para um projeto isolado no LangSmith por meio da variável `LANGCHAIN_PROJECT`, permitindo análise independente de cada agente. A taxa de amostragem é configurável via `LANGSMITH_TRACING_SAMPLING_RATE`, definida em 100% no ambiente de desenvolvimento e ajustável para valores menores em produção.

Para cobrir operações que o rastreamento automático não alcança, como as chamadas MCP a serviços externos, requisições A2A ao agente jurídico e consultas ao Weaviate com geração de embeddings, foram adicionados decoradores `@traceable` com tipos específicos (`run_type="retriever"` para buscas vetoriais, `run_type="embedding"` para embeddings, `run_type="tool"` para chamadas a serviços).

Além disso, cada invocação de grafo propaga metadados de negócio (canal, campanha, tarefa, modelo LLM) que permitem filtrar e correlacionar traces no painel do LangSmith. Com isso, o operador consegue visualizar o trace completo de uma validação de peça, desde a entrada no content-validation-service, passando pelas chamadas paralelas a specs, branding e compliance, até a resposta do LLM no legal-service. Isso expõe os gargalos de latência, falhas em serviços externos e comportamento dos modelos em cada canal de comunicação.

Os traces ficam disponíveis em <https://smith.langchain.com/>, como no exemplo abaixo:

The screenshot shows the LangSmith trace visualization interface. On the left, there's a sidebar with a tree view of traces, including 'LangGraph' (selected), 'validate_channel', 'validate_branding', 'validate_compliance', 'validate_legal_compliance' (which has a child node 'A2A: legal-service'), 'validate_specs', 'fetch_channel_specs', and 'issue_final_verdict'. The main panel shows a detailed trace for the 'issue_final_verdict' run. It includes sections for 'Run', 'Feedback', and 'Metadata'. The 'Run' section is expanded, showing a hierarchical tree of steps: 'Final Verdict' (with 'Decision APROVADO', 'Summary [Legal] Comunicação clara, sem dados s...', 'Branding null', 'Failure Stage null'), 'Legal' (with 'Decision APROVADO', 'Summary Comunicação clara, sem dados sensi...', 'Requires Human Review false'), 'Sources' (listing three items), and 'Specs' (listing one item). The 'Feedback' and 'Metadata' sections are collapsed.

Dashboards técnicos

Todos os serviços Python expõem métricas técnicas via endpoint /metrics no formato **Prometheus**, coletadas automaticamente a cada 15 segundos.

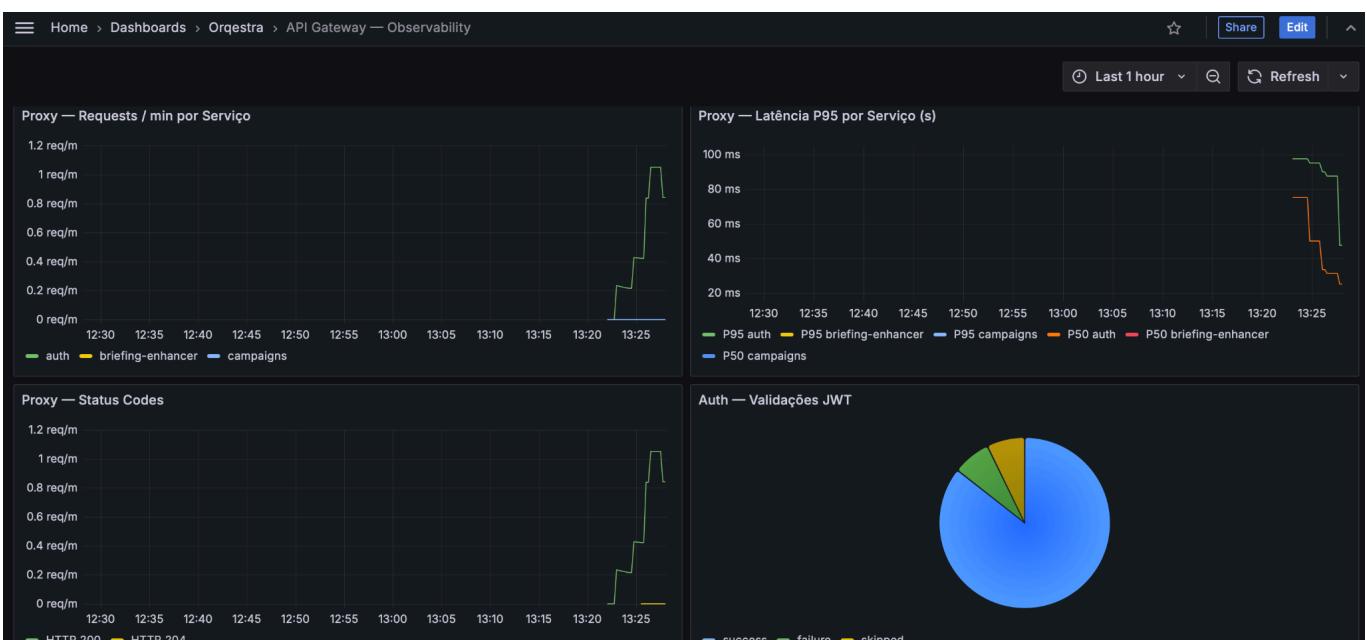
O **Grafana** disponibiliza dashboards pré-provisionados com painéis de latência de requisições HTTP, taxa de erros, uso de recursos, duração de chamadas LLM, tempo de retrieval no Weaviate e volume de validações por canal, permitindo monitoramento em tempo real da saúde e performance de toda a plataforma.

URL: <http://localhost:3001>

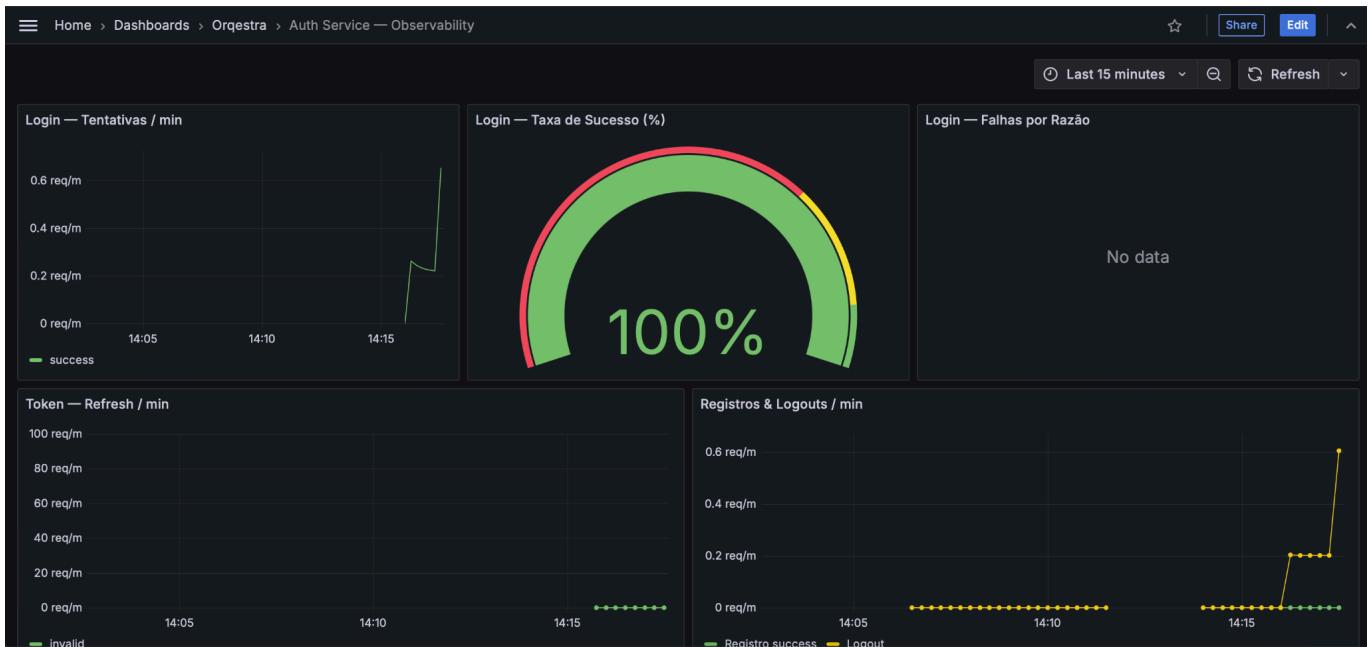
Usuário: admin/orquestra

Acessar o menu lateral > Dashboards > pasta Orquestra.

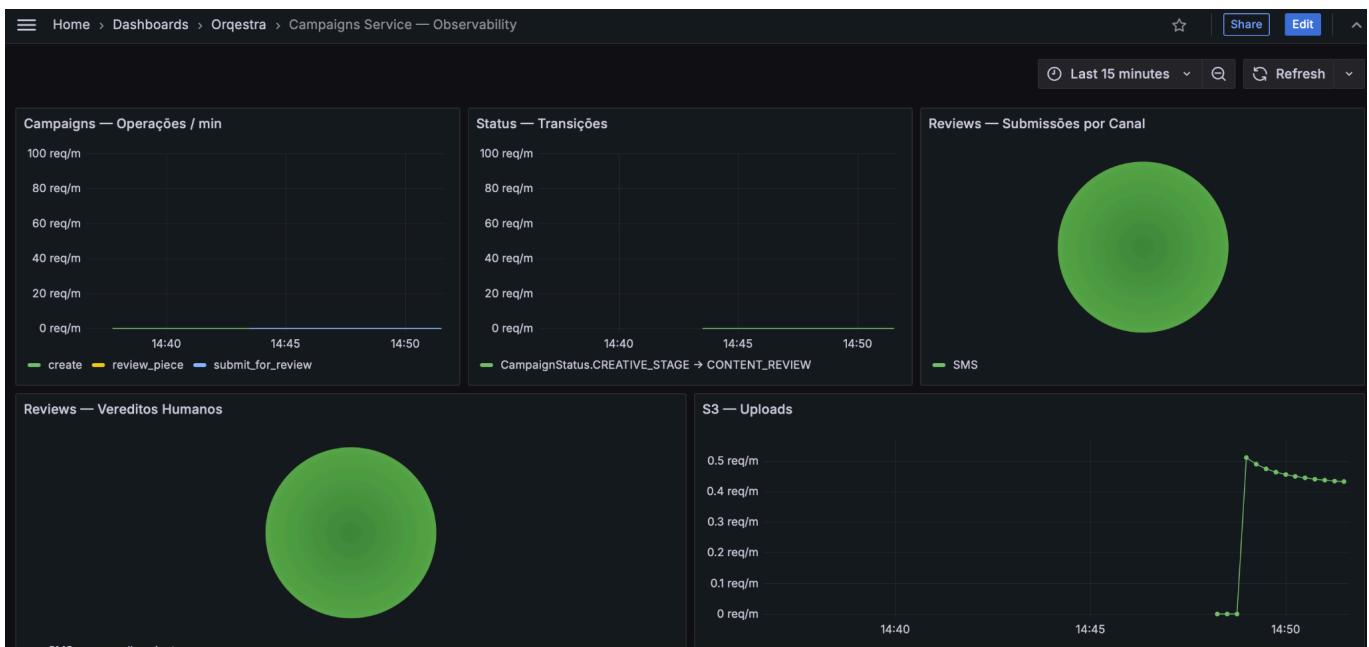
Dashboard #1 - API Gateway



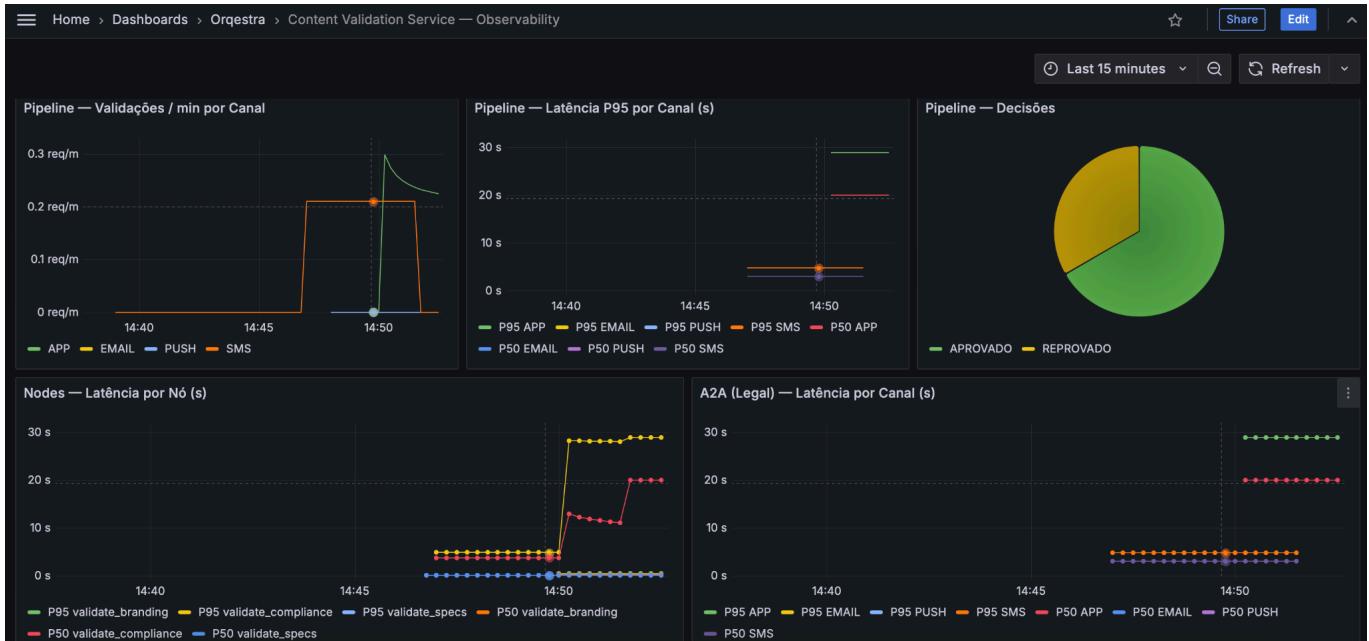
Dashboard #2 - Auth Service



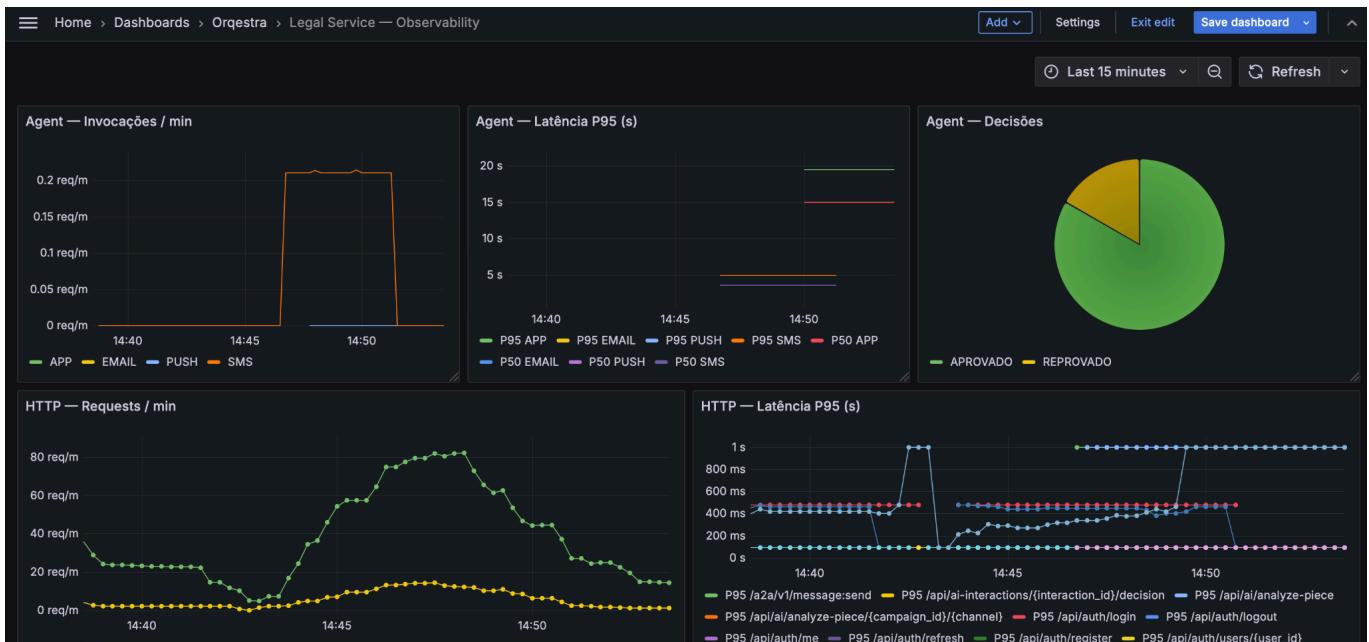
Dashboard #3 - Campaigns Service



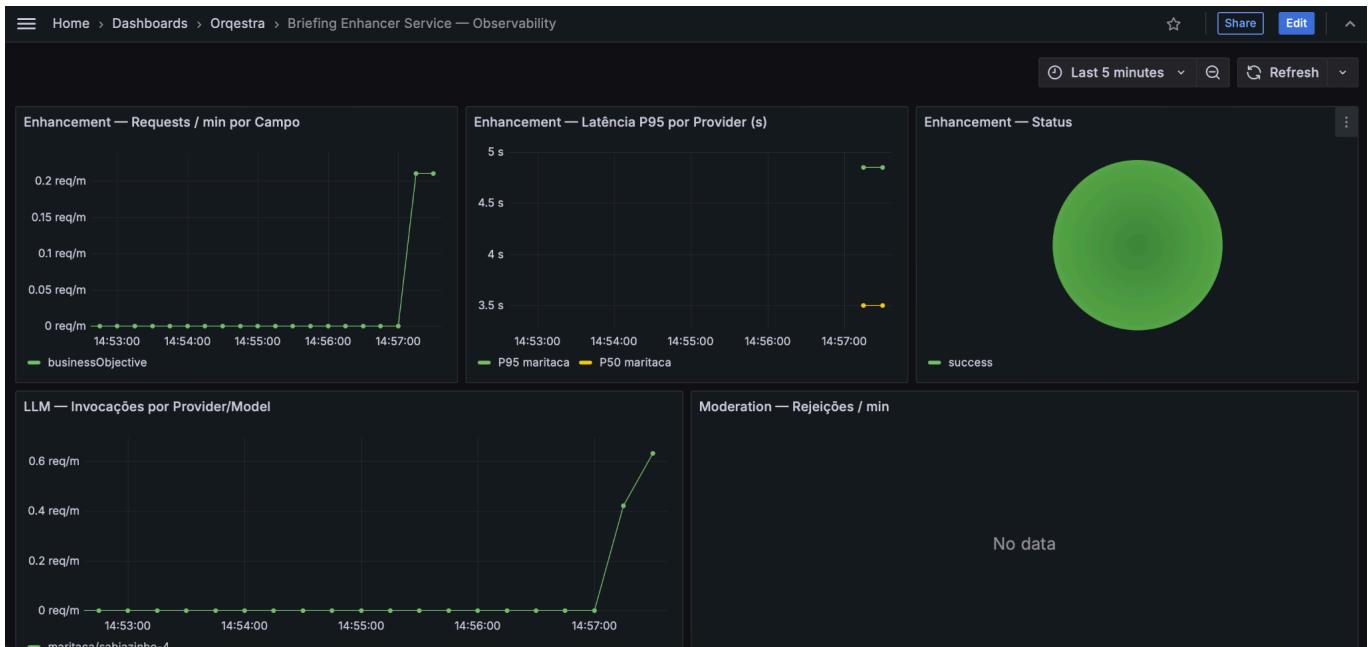
Dashboard #4 - Content Validation Service



Dashboard #5 - Legal Service



Dashboard #6 - Briefing Enhancer Service



Dashboards de negócio

Para visibilidade de negócio, o **Metabase** conecta-se diretamente aos bancos PostgreSQL dos serviços e oferece dashboards pré-configurados com indicadores como volume de campanhas por status, taxa de aprovação/reprovação de peças, distribuição de validações por canal, adoção da IA pelos analistas e tempo médio de ciclo das campanhas, fornecendo aos gestores uma visão consolidada da operação sem necessidade de acesso técnico.

URL: <http://localhost:3002>

Usuário: admin@orquestra.com/Orquestra2026!

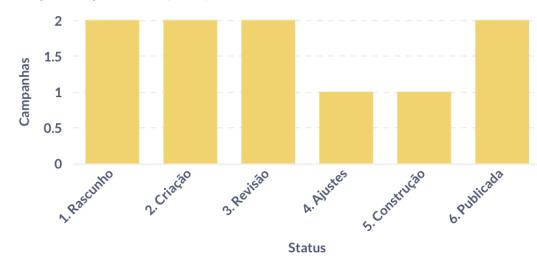
Dashboard #1 - Funil de campanhas

Acompanha o pipeline de campanhas do rascunho até a publicação.

[Orquestra] Funil de Campanhas

Exportar | Imprimir | Recortar | Copiar | Desfazer | Refazer | Busca | Ajuda

Campanhas por Status (Funil)



Volume de Impacto (BRL) por Status



Campanhas por Categoria

Aquisição	20%
Cross-sell	20%
Retenção	20%
Relacionamento	10%
Upsell	10%
Educativo	10%
Regulatório	10%



Campanhas por Canal

SMS	50%
Push	50%



Campanhas por Prioridade

Alta	50%
Normal	40%
Regulatório / Obrigatório	10%



Tempo Médio por Transição (horas)

De	Para	Horas (média)	Transições
CONTENT REVIEW	CAMPAIN BUILDING	21	2
CREATIVE STAGE	CONTENT REVIEW	21	4

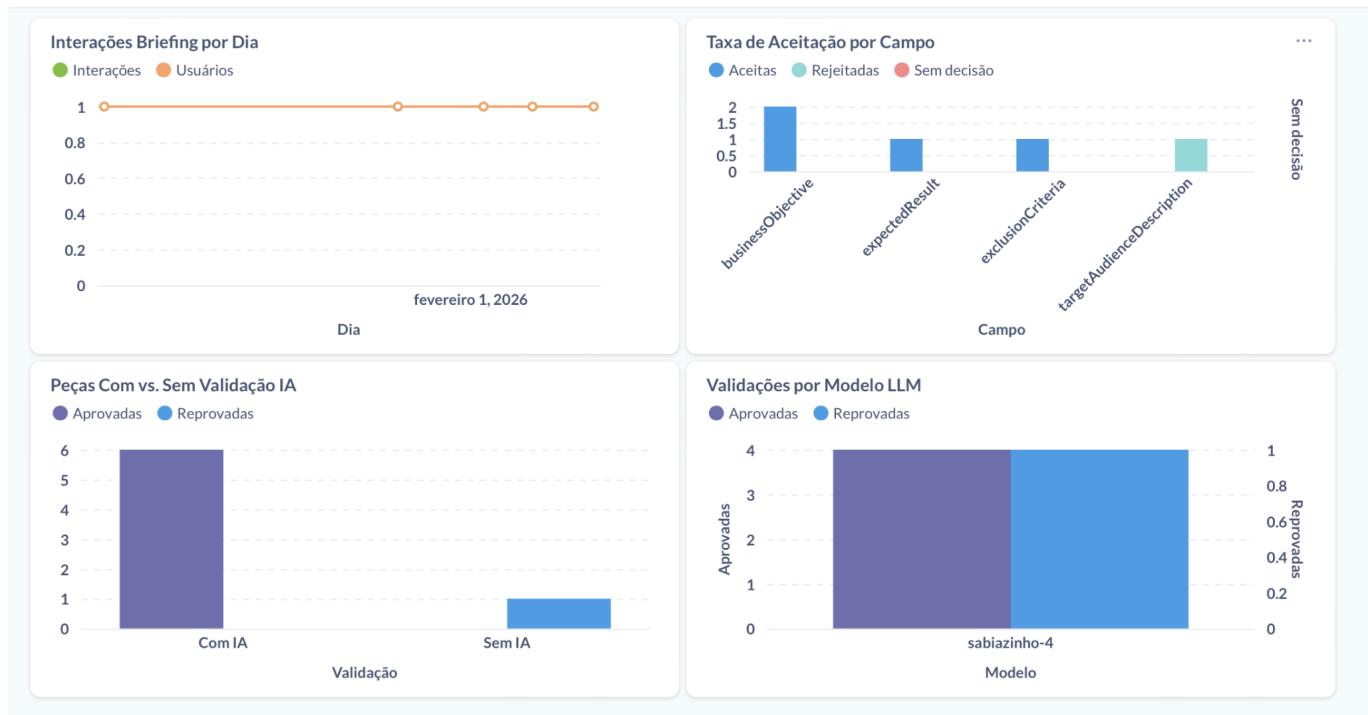
Dashboard #2 - Qualidade e compliance

Monitora a qualidade das peças criativas e a conformidade regulatória.



Dashboard #3 - Adoção e eficácia da IA

Mede o quanto os recursos com Inteligência Artificial embutida estão sendo utilizados pelos usuários da plataforma.



Considerações éticas e de responsabilidade com IA

A adoção de modelos de linguagem em sistemas que impactam decisões organizacionais exige atenção a aspectos éticos, de transparência e de governança. Na solução proposta, essas considerações foram tratadas como requisitos arquiteturais, incorporadas ao desenho dos serviços. A seguir, são apresentados os mecanismos implementados, organizados por dimensão ética.

Supervisão humana

O princípio central adotado no projeto é que a inteligência artificial atua exclusivamente como ferramenta de apoio à decisão, nunca como decisor autônomo. Essa premissa se materializa em três mecanismos complementares. No workflow de revisão de peças criativas, o veredito da IA e o veredito humano coexistem como campos independentes no modelo de dados. O gestor de marketing pode aprovar, rejeitar ou, de forma explícita, sobreescriver uma aprovação da IA por meio da ação de rejeição manual.

No serviço de enriquecimento de briefings, cada sugestão gerada pelo modelo de linguagem é registrada na tabela de auditoria com um campo de decisão do usuário, que permanece nulo até que o analista de negócios aceite ou rejeite a proposta. Adicionalmente, o serviço de validação de conteúdo e o serviço jurídico sinalizam, por meio do campo `requires_human_approval`, situações em que a revisão humana é imprescindível (tipicamente em reprovações ou quando ocorrem falhas no processamento). Essa sinalização é propagada até a interface do usuário, impedindo que decisões automatizadas passem despercebidas.

A própria base de conhecimento jurídica utilizada pelo sistema contém, entre suas diretrizes, a determinação de que a IA não substitui a aprovação humana final e que todas as sugestões automatizadas devem permitir aceitação ou rejeição pelo usuário.

Moderação de conteúdo

O serviço de enriquecimento de briefings integra a API de moderação da OpenAI como middleware no grafo de execução do agente. Toda entrada fornecida pelo usuário é submetida ao modelo de moderação antes de ser processada pelo modelo de linguagem. Caso o conteúdo seja classificado como inadequado, a execução é interrompida, uma mensagem explicativa é retornada ao usuário e a rejeição é contabilizada por meio de métricas Prometheus, permitindo o monitoramento contínuo de tentativas de uso indevido.

Auditoria e rastreabilidade

O projeto mantém tabelas de auditoria distribuídas entre os serviços: validações jurídicas, validações de peças, interações com IA no enriquecimento de briefings, eventos de revisão de peças, transições de status de campanhas e tentativas de autenticação. Todas as tabelas dos agentes de IA registram o modelo de linguagem utilizado em cada operação, permitindo rastrear qual versão do modelo produziu cada decisão. Esse registro viabiliza análises retrospectivas de qualidade e a identificação de eventuais degradações de desempenho associadas a atualizações de modelo.

Transparência e explicabilidade

A explicabilidade das decisões automatizadas é assegurada em múltiplas camadas. O serviço jurídico retorna, junto com cada veredito, a lista de documentos normativos consultados durante a análise,

permitindo que o usuário identifique em quais diretrizes a decisão se fundamenta. O serviço de enriquecimento de briefings acompanha cada sugestão com uma explicação textual das alterações propostas. O serviço de validação de conteúdo discrimina, no resultado, cada etapa do pipeline de verificação - especificações técnicas, conformidade visual e compliance jurídico -, indicando em qual estágio e por qual motivo uma peça foi eventualmente reprovada. Essa granularidade evita que o sistema funcione como uma caixa-preta e permite ao usuário compreender e contestar as decisões apresentadas.

Prevenção de abuso

O API Gateway implementa limitação de taxa configurável por endpoint, com restrições mais severas para os serviços que consomem modelos de linguagem. O serviço de autenticação impõe limites adicionais para criação de contas, restringindo o registro a cinco cadastros por hora. Os agentes de IA tratam erros de limitação de taxa dos provedores de forma controlada, retornando mensagens comprehensíveis ao usuário em vez de expor detalhes de infraestrutura.

Observabilidade

Métricas operacionais são exportadas via Prometheus e visualizadas em painéis Grafana, possibilitando o monitoramento contínuo de indicadores relevantes para a governança de IA: volume de rejeições por moderação, distribuição de invocações por modelo e provedor, taxa de erros por categoria e volume de validações por canal e veredito. Essa instrumentação permite a detecção precoce de anomalias e subsidia decisões de manutenção e evolução dos modelos adotados.

Melhorias futuras

A arquitetura atual foi concebida para ambiente de desenvolvimento com Docker Compose, o que permitiu iterar rapidamente sobre os serviços e validar a solução de ponta a ponta. No entanto, uma eventual ida à produção demandaria evoluções em diferentes frentes para garantir escalabilidade, resiliência e uma operação sustentável. Algumas considerações:

- **Orquestração de containers com Kubernetes**

A substituição do Docker Compose por um cluster Kubernetes (EKS na AWS, por exemplo) permitiria escalar cada serviço de forma independente com base na demanda real. Serviços com uso intensivo de LLM, como o Legal Service e o Briefing Enhancer, poderiam ter políticas de autoscaling baseadas em latência ou profundidade de fila, enquanto serviços determinísticos como o Branding e o HTML Converter operariam com réplicas fixas e recursos mínimos. O Kubernetes também traria rolling deployments com zero downtime, health checks nativos com liveness e readiness probes, e gerenciamento declarativo de configurações e segredos.

- **Segurança e governança**

- A rotação automática de chaves de API via Secrets Manager, a criptografia em trânsito (mTLS entre serviços) e em repouso (RDS e ElastiCache com encryption at rest), e a implementação de audit logs centralizados complementariam a segurança já existente no API Gateway. Um sistema de governança de IA, com registro versionado de prompts e modelos utilizados, garantiria rastreabilidade regulatória.
- A comunicação entre os agentes via A2A e com as tools via MCP não possui autenticação. Em um ambiente produtivo, seria necessário que esses serviços estivessem isolados em uma VPC interna ou que mecanismos de autenticação fossem implementados entre eles.

- **Análise das peças de e-mail**

- A abordagem de converter as peças para imagens e passá-las para o Legal Service junto ao HTML produziu payloads bastante extensos. Por mais que tenha sido definido um limite de tamanho das peças, uma abordagem melhor provavelmente seria fazer extrações das imagens individuais e extração dos links. Dessa forma seria possível separar com maior precisão o que poderia ser enviado para o Legal Service e o que poderia ser validado com regras fixas, sem LLM. O mesmo se aplica ao Branding Service, que faz análise de todo o HTML e logotipo da empresa, mas não analisa outros tipos de imagem que podem estar presentes na peça.

- **Histórico de interações com os agentes e dashboards de negócio**

- Implementar expurgo dos bancos de dados de cada serviço, com ingestões periódicas desses dados para um Data Lake que pudesse manter o histórico completo.
- Migrar a fonte dos dashboards de negócio para o Lake (em vez de consumir diretamente dos bancos de dados de serviço) ou preparar os dados em um schema apartado e exclusivo aos dashboards.

- **Content Validation Service**

- O serviço pode ser evoluído para utilizar um LLM que emita um parecer único, dados os retornos individuais dos serviços chamados.
- Em caso de reprovação, o serviço poderia sugerir uma variação da mensagem original que atenda às especificações e possa ser aceita pelo usuário.