



Manual Técnico Definitivo — Sistema de Chat



CAMADA 01 — DEFINIÇÃO TÉCNICA E PROPÓSITO DO SISTEMA DE CHAT VIBRACIONAL



Entrada

No FriendApp, o **chat** não é apenas uma troca de mensagens, mas um **sistema vibracional** que registra, processa e reflete a energia dos usuários em tempo real. Esta camada define com profundidade o **propósito técnico** e o **papel estratégico** do Sistema de Chat, incluindo seus componentes de arquitetura, privacidade e integração com o ecossistema.



Propósito Estratégico

- Ser o **primeiro chat do mundo** que traduz intenções humanas em dados vibracionais.
- Garantir que toda interação seja **segura, fluida e transparente**.
- Atuar como ponto central de integração com **Feed Sensorial, Eventos, Locais Parceiros, Modo Bora e Mapa de Frequência**.
- Servir de **laboratório vibracional** para a IA Aurah Kosmos: cada conversa gera dados para evolução contínua do matching.




Arquitetura Macro (Visão Geral)

Componente	Função
Interface do Usuário	Área de mensagens com painel vibracional em tempo real.

Componente	Função
Aurah Kosmos (IA)	Guia vibracional, opera sobre metadados e feedbacks pós-chat .
Banco de Dados	Firestore (tempo real), PostgreSQL (histórico/logs), Neo4j (relações).
Modo Pulsar	Mensagens efêmeras que desaparecem após visualização/tempo definido.
Painel Vibracional	Mostra estados Pico, Transição, Colapso com atraso assíncrono.
Logs Vibracionais	Registro criptografado de metadados e curvas energéticas.

Dados Processados (Somente Metadados)

- **Reciprocidade:** tempo de resposta entre usuários.
- **Intensidade:** volume de mensagens em determinado intervalo.
- **Pausas:** duração de intervalos sem interação.
- **Expressividade:** uso de emojis (positivos/negativos).
- **Feedback Pós-Chat:** avaliação direta do usuário.

 **Privacidade garantida:** a IA **não lê conteúdo criptografado**. Apenas metadados são analisados.

Fórmula Simplificada do Score Vibracional

$$\text{FrequênciaChat} = (w1 * \text{Reciprocidade}) + (w2 * \text{Intensidade}) + (w3 * \text{Emojis}) - (w4 * \text{Pausas})$$

- **Output:** float 0.0 – 10.0
- 7.0 – 10.0 → Pico
- 4.0 – 6.9 → Transição
- 0.0 – 3.9 → Colapso

(Pesos ajustados dinamicamente pela IA com base no histórico do usuário.)

Segurança e Privacidade

Item	Medida Aplicada
Criptografia	AES-256 em trânsito e armazenamento seguro em Firestore/PostgreSQL.
End-to-End (E2EE)	Mantido, com IA atuando apenas em metadados fora da camada de conteúdo.
Firewall Vibracional	Bloqueia spam, abusos ou padrões ofensivos de linguagem.
LGPD/GDPR	Consentimento explícito, exclusão de dados sob demanda e logs auditáveis.

Exemplo de API (Criação de Chat)

Endpoint: `POST /api/chat/create`

Request:

```
{
  "user_id": "abc123",
  "target_id": "xyz789",
  "origem": "feed_sensorial",
  "tag_intencao": "casual"
}
```

Response (Sucesso):

```
{
  "status": "sucesso",
  "chat_id": "uuid",
  "estado_inicial": "transicao"
}
```

Response (Erro):

```
{
  "status": "erro",
}
```

```
"mensagem": "Usuário bloqueado ou não verificado."
}
```

Fechamento Profissional

A Camada 01 estabelece que o chat vibracional é a **espinha dorsal do FriendApp**: privado, seguro, assíncrono e enriquecido por insights. Ele garante **fluidez técnica e confiança ética**, criando um ambiente onde tecnologia e vibração se unem.


➡ Na próxima camada (02), vamos mergulhar na **arquitetura visual e funcional completa**, detalhando como os componentes se conectam no fluxo operacional.

CAMADA 02 — ARQUITETURA VISUAL E FUNCIONAL DO SISTEMA DE CHAT

Entrada

Depois de definir o propósito estratégico, precisamos visualizar **como o sistema de chat se organiza internamente**. Esta camada detalha a **arquitetura funcional**, descrevendo entradas, processamento, segurança, bancos de dados e a entrega final da experiência vibracional ao usuário.

Entradas para Abertura de Chat

Origem	Fluxo Técnico	Exemplo de Trigger
Chat Direto	Solicitação enviada de um perfil para outro. Validado por IA em metadados.	Clique em "Conectar"
Feed Sensorial	Botão "Iniciar Chat" visível em postagens com intenção aberta.	Post com tag "  Reflexiva"
Modo Bora	Aceitação mútua em plano ativo dispara criação do chat.	Ambos clicam em "Bora"

Origem	Fluxo Técnico	Exemplo de Trigger
Eventos/Locais	Check-ins simultâneos ativam sugestão de chat coletivo ou individual.	Festival → "Trocar ideia agora?"
Sugestão da IA	IA Aurah Kosmos recomenda abertura com base em metadados e histórico.	"Energia compatível detectada, deseja iniciar chat?"

IA de Curadoria (Aurah Kosmos)

- Atua **somente sobre metadados**: tempo de resposta, emojis, intensidade, pausas.
- Sugere tags de intenção no início e reclassificações durante a conversa.
- Nunca encerra chats ou interfere diretamente: fornece **insights pós-chat**.
- Opera em pipeline **assíncrono**, garantindo que a entrega de mensagens seja instantânea.

Pipeline Operacional Assíncrono

Usuário envia mensagem



Entrega imediata (tempo real via WebSocket)



Pipeline de eventos (Kafka/PubSub) → coleta metadados



IA processa e atualiza estado vibracional (2-3s de atraso)



Painel do chat é atualizado para ambos os usuários



Logs são salvos (Firestore + PostgreSQL + Neo4j)

Segurança e Governança

Camada de Proteção	Implementação Técnica
Criptografia	AES-256 em trânsito + armazenamento seguro.
Firewall Vibracional	Bloqueio de spam, abusos e padrões de linguagem ofensiva.
Privacidade	Conteúdo da mensagem inalterado e inviolável (E2EE).
LGPD/GDPR	Consentimento de uso, logs auditáveis, exclusão sob demanda.
Controle de Sessões	Tokens JWT com expiração e refresh para cada chat.

Bancos de Dados e Funções

Banco	Função Principal
Firestore	Mensagens em tempo real, estados de chat.
PostgreSQL	Histórico de logs, feedbacks, auditoria vibracional.
Neo4j	Relacionamentos entre usuários, clusters de afinidade energética.
Redis	Cache de sessões e últimos estados vibracionais para performance <300ms.

Exemplo de API (Listar Chats Ativos)

Endpoint: `GET /api/chat/listar`

Response (Sucesso):

```
{
  "status": "sucesso",
  "chats": [
    {
      "chat_id": "uuid",
      "intencao": "💬 Casual",
      "estado_vibracional": "pico",
      "ultimo_update": "2025-09-06T14:23:00Z"
    }
  ]
}
```

Response (Erro):

```
{
  "status": "erro",
  "mensagem": "Usuário não possui chats ativos."
}
```

Fechamento Profissional

A arquitetura do Sistema de Chat garante **fluidez operacional** e **segurança vibracional**: múltiplas portas de entrada, IA atuando como curadora, processamento assíncrono e armazenamento robusto.

Na próxima camada (03), detalharemos **cada forma de abertura de chat**, com suas regras técnicas e impactos vibracionais.

CAMADA 03 — FORMAS DE ABERTURA DE CHATS

Entrada

O **ponto de partida** de cada conversa define o tom vibracional que ela seguirá. Nesta camada, detalhamos as **cinco formas oficiais de abertura de chats** no FriendApp, suas regras técnicas, os dados processados e a atuação da IA Aurah Kosmos em cada caso.

Modos de Abertura

1. Abertura Direta (Manual)

- **Fluxo:** Usuário acessa perfil → clica em **"Conectar"** → **"Iniciar Chat"**.
- **Validações Técnicas:**
 - Status de conexão ativo.
 - Verificação documental (DUC/DCO) obrigatória.

- IA valida metadados mínimos: tempo de interação, histórico de bloqueios.

- **Exemplo de API:**

```
POST /api/chat/create
{
  "user_id": "abc123",
  "target_id": "xyz789",
  "origem": "perfil_direto"
}
```

2. Abertura via Feed Sensorial

- **Fluxo:** Postagem com **intenção ativa** exibe botão **"Iniciar Chat sobre isso"**.
- **Validações Técnicas:**
 - Post precisa estar público e ativo.
 - IA sugere **tag inicial** com base no tipo de postagem (ex.: reflexiva, social).
- **Exemplo de Trigger:**

```
{
  "post_id": "post123",
  "tag_sugerida": "🌀 Reflexiva",
  "score": 0.87
}
```

3. Abertura via Modo Bora

- **Fluxo:** Dois usuários clicam em **"Bora"** no mesmo plano.
- **Validações Técnicas:**
 - Aceitação deve ser recíproca.
 - IA atribui tag inicial **"social/espontânea"**.

- **Exemplo de API:**

```
POST /api/chat/bora
{
  "plano_id": "plan456",
  "user_a": "abc123",
  "user_b": "xyz789"
}
```

4. Abertura em Eventos ou Locais Parceiros

- **Fluxo:** Check-ins simultâneos em um evento/local.
- **Validações Técnicas:**
 - IA valida proximidade geográfica (GPS + rede Wi-Fi).
 - Local pode fornecer **tag vibracional base** (ex.: "dança", "gastronomia").
- **Exemplo de Trigger:**

```
{
  "evento_id": "event789",
  "usuarios_conectados": 24,
  "tag_base": "🔥 Festa"
}
```

5. Abertura Sugerida pela IA Aurah Kosmos

- **Fluxo:** IA detecta sinais em metadados (reciprocidade, histórico de matches).
- **Mensagem de Sugestão:**
 - | "Energia compatível detectada. Deseja abrir um chat com [usuário X]?"
- **Validações Técnicas:**
 - Feedbacks prévios positivos.

- Não há bloqueios ou denúncias entre os envolvidos.

- **Exemplo de API:**

```
POST /api/chat/sugestao
{
  "user_id": "abc123",
  "target_id": "xyz789",
  "base_score": 0.92
}
```

Regras Gerais para Todas as Aberturas

Regra Técnica	Ação do Sistema
Usuário bloqueado	Chat não pode ser iniciado.
Falta de verificação documental	Usuário é redirecionado para fluxo de verificação DUC.
Incompatibilidade grave	IA sugere insight alternativo: "Talvez não seja o momento certo."
Logs	Toda tentativa de abertura é registrada no Firestore + PostgreSQL.

Segurança e Privacidade

- Nenhum chat é aberto sem **consentimento mútuo**.
- Todos os processos são **assíncronos**, sem travar a experiência.
- Conteúdo inicial é sempre criptografado antes de trafegar.
- Logs de abertura são **anonimizados** para análise posterior.

Fechamento Profissional

As múltiplas formas de abertura garantem que o chat vibracional **sempre nasce de um contexto legítimo e intencional** — seja espontâneo (Bora), inspirado (Feed), circunstancial (Eventos/Locais) ou sugerido pela IA.




Na próxima camada (04), mergulharemos na **classificação dos estados vibracionais** que modulam a vida útil de cada chat.

CAMADA 04 — ESTADOS VIBRACIONAIS DO CHAT (PICO, TRANSIÇÃO, COLAPSO)

Entrada

Cada conversa vibra em ciclos. O FriendApp traduz esses ciclos em **estados vibracionais** que indicam a qualidade da troca em tempo real. Esta camada define a lógica técnica, os critérios de classificação, exemplos de aplicação e os fluxos de atualização no painel do usuário.

Estados Definidos

Estado	Definição Técnica	Critério Objetivo	Visual no Painel
 Pico	Conversa em alta frequência, reciprocidade intensa, fluxo estável.	Score vibracional ≥ 7.0	Barra dourada pulsante
 Transição	Oscilação leve, pausas moderadas ou mudança gradual de tom.	$4.0 \leq \text{Score} < 7.0$	Barra azul clara
 Colapso	Queda abrupta de reciprocidade, silêncio prolongado, desconexão energética.	Score < 4.0 , tempo sem resposta > 10 min ou feedback negativo explícito.	Barra vermelha estática

Inputs Usados para o Cálculo

Input Técnico	Como é Capturado	Peso Médio
Tempo de resposta	Milissegundos entre envios	0.35
Volume de mensagens	Contagem por minuto	0.25
Emojis expressivos	Classificação positiva/negativa	0.20

Input Técnico	Como é Capturado	Peso Médio
Pausas longas	Intervalos > 5 minutos	0.15
Feedback do usuário	Escala pós-chat (1–5)	0.05

Fórmula Técnica (PseudoCódigo)

```
def calcular_frequencia(resposta_ms, msgs_min, score_emojis, pausas, feedback):
    score = (0.35 * normalizar_resposta(resposta_ms)) \
        + (0.25 * normalizar_volume(msgs_min)) \
        + (0.20 * score_emojis) \
        - (0.15 * pausas) \
        + (0.05 * feedback)
    return round(score, 2)
```

Fluxo Operacional Assíncrono

1. Usuário envia mensagem.
2. Sistema entrega instantaneamente (WebSocket).
3. Pipeline assíncrono processa **metadados** em fila (Kafka ou Pub/Sub).
4. Aurah Kosmos atualiza score vibracional em até **3s**.
5. Painel do chat exibe estado atualizado.
6. Logs são gravados para auditoria.

Exemplo Prático

- Usuário A responde em 2s, envia emojis 🥰🔥.
- Usuário B responde em 5s, mantém frequência alta.
- Resultado → **Score: 8.2** → **Estado: Pico**.

| Painel mostra: "Vocês estão em alta sintonia vibracional."

Segurança e Privacidade

- Apenas **metadados** são processados.
- Conteúdo criptografado permanece **intocado** (E2EE).
- Logs de estados são anonimizados.
- Usuário pode excluir histórico a qualquer momento.

Fechamento Profissional

Os estados vibracionais transformam conversas em métricas claras, ajudando usuários a perceberem a energia da troca em tempo real, sem sacrificar privacidade.

Na próxima camada (05), detalharemos a **atuação da IA Aurah Kosmos**, explicando seu papel como **curadora e guia**, nunca como controladora.

CAMADA 05 — ATUAÇÃO DA IA AURAH KOSMOS NO SISTEMA DE CHAT

Entrada.

A **Aurah Kosmos** é o núcleo inteligente do FriendApp. No chat, seu papel é de **curadora vibracional**: observar metadados, sugerir insights e oferecer feedbacks pós-chat. Nunca é juíza ou controladora da conversa. Esta camada explica com precisão como a IA atua de forma ética, transparente e tecnicamente robusta.

Funções-Chave da Aurah Kosmos

Função	Definição Técnica	Momento de Atuação
Leitura de Metadados	Processa ritmo, tempo de resposta, emojis e pausas longas.	A cada interação

Função	Definição Técnica	Momento de Atuação
Sugestão de Intenção	Propõe ou ajusta tags de intenção com base no fluxo vibracional.	Início e mudanças
Curadoria de Ritmo	Recomenda equilíbrio: "responder com calma" ou "fazer uma pausa".	Transições
Prevenção de Colapso	Identifica quedas e envia aviso privado ao usuário .	Durante quedas
Insights Pós-Chat	Após encerramento, oferece reflexões e feedback vibracional privado.	Fim da conversa
Aprendizado Contínuo	Usa feedbacks e padrões para treinar modelos futuros de matching.	Pós-processamento



Dados Usados pela IA (Sempre Metadados)

- **Reciprocidade:** tempo médio de resposta entre usuários.
- **Intensidade:** volume de mensagens trocadas por minuto.
- **Expressividade:** emojis positivos (+) e negativos (-).
- **Pausas:** intervalos sem resposta.
- **Feedback Pós-Chat:** avaliação do usuário.



Não processa conteúdo criptografado (E2EE).



Algoritmo de Detecção de Colapso (Simplificado)

```
def detectar_colapso(tempo_resposta, score_emojis, pausas, reciprocidade):
    score = (0.4 * reciprocidade) + (0.3 * score_emojis) \
        - (0.2 * pausas) - (0.1 * tempo_resposta)
    if score < 3.5:
        return "colapso"
    elif score < 6.0:
        return "transicao"
    else:
        return "pico"
```

Fluxo Operacional Assíncrono da IA

1. **Mensagem enviada** → entregue ao receptor em tempo real (WebSocket).
 2. **Fila de eventos** (Kafka/PubSub) registra metadados.
 3. **IA processa em segundo plano** → gera estado vibracional em até 3s.
 4. **Painel do chat atualizado**.
 5. **Insights armazenados** para análise pós-chat.
-

Exemplo de Atuação

- Usuário A demora 8 min para responder, envia 😞.
 - Score cai para **3.2** → classificado como **Colapso**.
 - Painel exibe: "Energia baixa detectada".
 - IA não encerra nada: apenas sugere **feedback pós-chat**.
-

Segurança e Ética

Risco	Medida Implementada
Quebra de privacidade	IA opera somente em metadados; mensagens são E2EE.
Controle excessivo	IA nunca encerra chat; apenas recomenda e sugere.
Falta de transparência	Insights explicam sempre "como" e "por que" o estado foi calculado.
Uso indevido de dados	Logs anonimizados e vinculados a LGPD/GDPR.

Fechamento Profissional

A Aurah Kosmos no chat é **uma mentora invisível**: acompanha, interpreta padrões, mas entrega ao usuário apenas conselhos, nunca comandos. Essa postura preserva a **autonomia**, a **privacidade** e fortalece a confiança na plataforma.









➡ Na próxima camada (06), detalharemos o **Sistema de Intenções Vibracionais (Tags de Intenção)**, que organizam o tom das conversas.

CAMADA 06 — SISTEMA DE INTENÇÕES VIBRACIONAIS (TAGS DE INTENÇÃO)

Entrada

No FriendApp, nenhuma conversa é neutra. Cada troca carrega uma **intenção vibracional**, que é capturada, organizada e traduzida em **tags**. Estas tags servem como metadados estruturados para a IA Aurah Kosmos, para o usuário e para o ecossistema, permitindo alinhar o tom, modular o fluxo e gerar histórico energético confiável.

Catálogo Oficial de Tags de Intenção

Tag Vibracional	Descrição Técnica	Exemplo de Uso no Chat
 Casual	Conversa leve, sem foco específico.	"Oi, tudo bem?"
 Atração	Mensagens com energia de paquera, flerte ou interesse afetivo.	"Gostei de você, bora marcar algo?"
 Profunda	Trocas emocionais, reflexivas, espirituais ou de vulnerabilidade.	"Ando pensando em mudanças na minha vida."
 Intelectual	Debates, reflexões, trocas de ideias complexas.	"Você viu o artigo sobre IA quântica?"
 Resenha	Conversa descontraída, divertida, piadas e leveza.	"Esse meme é sua cara kkkk."
 Objetiva	Conversa prática, com meta clara (marcar encontro, resolver algo rápido).	"Amanhã às 20h pode?"
 Introspectiva	Filosófica, existencial, de autoexploração.	"Quem somos além das aparências sociais?"
 Desconforto	Estado vibracional de queda, ruído ou conflito emergente.	"Não tô curtindo muito essa conversa."

Estrutura Técnica das Tags


```
collection: tags_chat
→ tag_id (string)
→ nome (string)
→ emoji (string)
→ categoria (string)
→ descrição (string)
→ prioridade_IA (int)
→ atribuicao: "manual" | "automatica"
→ atualizado_em: timestamp
```

- **Manual:** Usuário escolhe tag explicitamente.
- **Automática:** IA sugere/reclassifica com base em metadados.
- **Prioridade IA:** tags críticas (ex.: ❄️ Desconforto) possuem peso maior na curadoria.



Atuação da IA Aurah Kosmos com Tags

1. Sugestão Inicial:

- Ao abrir o chat, IA propõe tag com base no contexto (ex.: Feed reflexivo → 🌀 Introspectiva).

2. Reclassificação Dinâmica:

- Durante a conversa, se ritmo e expressividade mudam, IA pode atualizar tag.
- Exemplo: de 💬 Casual para 💖 Profunda, caso emojis 💖 🙏 aumentem.

3. Feedback Pós-Chat:

- Tag final registrada no histórico do usuário e usada para próximos matchings.



Algoritmo de Reclassificação (Simplificado)

```
def reclassificar_tag(emojis, intensidade, reciprocidade, feedback):
    if "❤️" in emojis or intensidade > 8:
        return "🔥 Atração"
```

```
elif reciprocidade > 7 and feedback >= 4:
    return "💕 Profunda"
elif "😐" in emojis or feedback <= 2:
    return "❄️ Desconforto"
else:
    return "💬 Casual"
```

Exemplo de API — Autotag

Endpoint: `POST /api/chat/autotag`

Request:

```
{
  "chat_id": "xyz123",
  "user_id": "abc456",
  "metadados": {
    "tempo_resposta": 4,
    "emojis": ["🔥", "😍"],
    "volume_mensagens": 15,
    "feedback_previo": 4
  }
}
```

Response:

```
{
  "tag_sugerida": "🔥 Atração",
  "score": 0.91,
  "atribuicao": "automatica"
}
```

Segurança e Privacidade

- Tags são visíveis apenas **dentro da conversa**.

- Não são exibidas em perfis públicos.
- Usuário pode **remover ou alterar manualmente**.
- Dados são criptografados e anonimizados em relatórios coletivos.

Fechamento Profissional

As **tags vibracionais** são a gramática oculta do FriendApp. Elas traduzem intenções em dados claros, permitem ajustes dinâmicos e alimentam a IA com insumos para matching mais assertivo.

➡ Na próxima camada (07), vamos detalhar o **Modo Pulsar**: mensagens efêmeras com segurança e liberdade.

CAMADA 07— MODO PULSAR (MENSAGENS TEMPORÁRIAS E AUTODESTRUTIVAS)

Entrada

Nem toda mensagem precisa ser eterna. O **Modo Pulsar** cria um espaço seguro e efêmero, permitindo que os usuários troquem mensagens temporárias, que desaparecem após leitura ou tempo configurado. É um recurso que une **expressão livre** com **privacidade vibracional**.

Características Técnicas

Elemento	Implementação Técnica
Temporalidade	Configuração de autodestruição: <code>on_view</code> (após leitura) ou <code>timed</code> (após X segundos).
Persistência	Mensagens não ficam salvas em bancos permanentes, apenas em cache transitório (RAM).
Proteção contra print	Tentativa de captura → bloqueio e log automático.

Elemento	Implementação Técnica
Visual	Balões com borda pulsante, timer regressivo e animação de dissipação.

Estrutura de Dados (Firestore + RAM)

```
temp_collection: pulsar_messages
→ chat_id (string)
→ message_id (uuid)
→ sender_id (string)
→ tipo: "text" | "image" | "audio" | "emoji"
→ expiration_type: "on_view" | "timed"
→ expiration_value: int (segundos)
→ status: "ativa" | "expirada" | "lida"
→ criado_em: timestamp
```

Atuação da IA Aurah Kosmos

- **Sugestão Inteligente:** detecta metadados de mensagens intensas (ex.: emojis 🤔🔥) e sugere o Modo Pulsar.
- **Não impõe:** sempre opcional, apenas recomendado.
- **Feedback Pós-Chat:** se muitos Pulsares forem usados, IA gera insight:

"Notei que você prefere trocas efêmeras. Deseja manter esse estilo ou buscar conexões mais duradouras?"

Segurança Avançada

Risco Identificado	Mitigação
Captura de tela (Android/iOS)	Bloqueio imediato + alerta ao remetente.
OCR ou IA de terceiros	Firewall de OCR detecta padrões suspeitos.
Backup acidental	Proibido salvar mensagens Pulsar em bancos persistentes.

Risco Identificado	Mitigação
Vazamento de conteúdo	Logs registram apenas evento de expiração, nunca conteúdo da mensagem.

Casos de Uso

Cenário	Benefício Vibracional
Nudes ou paqueras diretas	Liberdade com segurança e confiança.
Desabafos emocionais	Catarse vibracional sem rastros permanentes.
Conversa confidencial	Trocas críticas em ambiente seguro.
Mensagens de impacto	Comunicação curta, intensa e irrepetível.

Exemplo de API

Endpoint: `POST /api/chat/pulsar`

Request:

```
{
  "chat_id": "uuid",
  "mensagem": "Essa é só pra você...",
  "tipo": "text",
  "expiration_type": "timed",
  "expiration_value": 30
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem_id": "uuid",
  "autodestruir_em": 30
}
```



Privacidade

- Dados **não entram em histórico**.
- Mensagens Pulsar são **inalteráveis** e não podem ser favoritadas.
- Modo disponível apenas para usuários **Premium** ou em chats de afinidade alta.



Fechamento Profissional

O **Modo Pulsar** é a válvula de escape do FriendApp: cria espaço para trocas intensas, efêmeras e seguras, onde a energia cumpre seu propósito e se dissolve com elegância.



Na próxima camada (08), abordaremos o **Check-in Vibracional e Feedback Pós-Chat**, que fecha o ciclo energético de cada conversa.



CAMADA 08 — CHECK-IN VIBRACIONAL E FEEDBACK PÓS-CHAT



Entrada

Cada conversa no FriendApp gera um rastro energético. O **Check-in Vibracional** e o **Feedback Pós-Chat** são responsáveis por transformar esse rastro em **dados úteis para o usuário e para a IA Aurah Kosmos**, criando um ciclo de autoconhecimento, segurança e aprendizado contínuo.



Objetivos Principais

1. **Autopercepção:** o usuário entende como aquela troca impactou sua energia.
2. **Refinamento do Matching:** IA melhora futuras conexões com base nos feedbacks.
3. **Segurança Vibracional:** detecção de padrões de colapso para prevenção.
4. **Histórico Energético:** formação de uma linha do tempo vibracional para cada usuário.

Fluxo Operacional

1. **Encerramento do Chat** → manual ou por inatividade.
 2. **Prompt Sutil:** *"Como você se sentiu ao final desta troca?"*.
 3. **Respostas Simbólicas:** ícones e frases rápidas.
 - 💖 Conexão profunda
 - 🌀 Intensa, mas breve
 - 😐 Neutra
 - 💔 Ruptura
 - ❄️ Fria/desconectada
 4. **Campo Livre (Opcional):** texto curto ou emoji extra.
 5. **Registro:** dados salvos em Firestore/PostgreSQL.
 6. **Processamento Assíncrono:** IA avalia em lote (não bloqueia o app).
-

Estrutura Técnica (Firestore)

```
collection: chat_feedback
→ chat_id (string)
→ user_id (string)
→ sentimento_final (enum)
→ intensidade (float 0.0-1.0)
→ tag_encerramento (string)
→ comentario_opcional (string)
→ timestamp (datetime)
```

Atuação da IA Aurah Kosmos

- **Inputs:** sentimento final, intensidade declarada, intenção inicial.
- **Processamento:**
 - Correlaciona com histórico vibracional do usuário.

- Compara feedbacks de múltiplos chats.
- **Outputs:**
 - Recomendações personalizadas no feed.
 - Ajustes de matching (evitar conexões reincidentes de colapso).
 - Insights privados:

"Suas últimas conexões foram mais intensas que leves. Deseja equilibrar sua energia?"



Indicadores de Qualidade Interna

Indicador	Finalidade Operacional
Taxa de feedbacks	Medir engajamento do usuário com o check-in (meta > 70%).
Média de ressonância	Avaliar saúde energética global da base.
Correlação IA x Usuário	Validar acurácia das classificações de estados vibracionais.
Reincidência de colapsos	Identificar perfis/locais mais propensos a quedas.



Exemplo de API

Endpoint: `POST /api/chat/feedback`

Request:

```
{
  "chat_id": "xyz123",
  "user_id": "abc456",
  "sentimento_final": "💔 Ruptura",
  "intensidade": 0.2,
  "comentario_opcional": "Achei pesado"
}
```

Response:


```
{
  "status": "sucesso",
  "mensagem": "Feedback registrado com segurança."
}
```

Privacidade e Ética

- Feedback nunca é visível ao outro participante.
- Usuário pode ignorar sem impacto negativo.
- Dados criptografados, com direito de exclusão sob LGPD/GDPR.
- Insights só são exibidos ao dono do perfil.

Fechamento Profissional

O **Check-in Vibracional** e o **Feedback Pós-Chat** fecham o ciclo das conversas com clareza, reforçando a autonomia do usuário e a evolução da IA. É aqui que experiências pessoais se tornam aprendizado coletivo sem comprometer a privacidade.

➡ Na próxima camada (09), vamos estruturar o **Sistema de Logs Vibracionais e Observabilidade Contínua**.

CAMADA 09— LOGS VIBRACIONAIS E OBSERVABILIDADE CONTÍNUA

Entrada

Nenhum sistema é confiável sem **observabilidade total**. No FriendApp, os **logs vibracionais** registram cada detalhe técnico e energético de uma conversa, de

forma criptografada e auditável. Eles são a base para **segurança, aprendizado da IA e insights para o usuário Premium**.

Objetivos dos Logs Vibracionais

1. **Rastreabilidade:** garantir histórico auditável de cada conversa.
2. **Segurança:** identificar abusos, colapsos ou tentativas de manipulação.
3. **IA Training:** alimentar modelos de intenção e compatibilidade.
4. **User Insights:** fornecer métricas vibracionais para usuários Premium.
5. **Governança:** compliance com LGPD/GDPR e auditoria interna.

Estrutura dos Logs

Tipo de Log	Exemplo Capturado	Finalidade Técnica
Intenção declarada	"... Casual" definida no início	Analisar consistência de intenção
Intenção real	Mudança dinâmica para "🔥 Atração"	IA ajusta curadoria
Reciprocidade	Tempo médio de resposta (3.2s)	Score de fluidez
Intensidade	25 msgs/min (pico)	Estado vibracional
Expressividade	Emojis positivos 😍🔥	Score positivo
Pausas críticas	12 minutos sem resposta	Indicador de transição/colapso
Feedback pós-chat	Usuário marcou "💔 Ruptura"	Ajuste no matching futuro
Eventos Pulsar	Mensagem expirada após 30s	Garantia de sigilo
Tentativa de print	Captura bloqueada em Modo Pulsar	Segurança e auditoria

Estrutura Técnica (Firestore + PostgreSQL)

```
collection: vibrational_logs
→ chat_id (string)
→ log_id (uuid)
→ timestamp (datetime)
```

→ user_id (string)
→ tipo_evento (string)
→ valor (string/float)
→ estado_chat (string) // pico, transicao, colapso
→ acao_ia (string/null)
→ origem (mensagem | emoji | pausa | feedback)

Algoritmo de Observabilidade

```
def gerar_log(chat_id, user_id, evento, valor, estado):  
    log = {  
        "chat_id": chat_id,  
        "user_id": user_id,  
        "tipo_evento": evento,  
        "valor": valor,  
        "estado_chat": estado,  
        "timestamp": datetime.now()  
    }  
    salvar_firestore(log)  
    salvar_postgresql(log)  
    return log
```

Painel Interno de Observação

Disponível apenas para **equipe de moderação** e **Aurah Kosmos (modo leitura)**:

- **Heatmap energético:** picos e colapsos por hora.
- **Curva temporal:** evolução do estado vibracional por minuto.
- **Taxa de intervenção da IA:** quando sugeriu insights.
- **Feedbacks agregados:** correlação entre percepção do usuário e logs técnicos.

Métricas Estratégicas

Métrica	Objetivo Interno
Tempo médio até colapso	Avaliar resiliência de chats
Taxa de feedbacks dados	Engajamento dos usuários com check-in vibracional
Intervenções de IA	Ajustar algoritmos de curadoria
Emoções dominantes	Relatórios de padrões coletivos (anonimizados)

Exemplo de API — Consulta de Logs

Endpoint: `GET /api/chat/logs/{chat_id}`

Response:

```
{
  "status": "sucesso",
  "logs": [
    {
      "log_id": "log123",
      "timestamp": "2025-09-06T15:22:00Z",
      "tipo_evento": "emoji",
      "valor": "😍",
      "estado_chat": "pico"
    }
  ]
}
```

Privacidade e Ética

- Logs **não armazenam conteúdo textual criptografado.**
- Apenas metadados, sempre anonimizados para relatórios coletivos.
- Usuário pode solicitar exclusão de histórico sob LGPD/GDPR.
- Moderadores só têm acesso a relatórios em caso de denúncia.

Fechamento Profissional

O **Sistema de Logs Vibracionais e Observabilidade** garante que cada conversa seja auditável, segura e evolutiva. É a ponte entre a experiência individual e a inteligência coletiva do FriendApp.

➡ Na próxima camada (10), vamos detalhar a **Criptografia, Firewall Vibracional e Segurança de Mensagens**.

CAMADA 10 — CRIPTOGRAFIA, FIREWALL VIBRACIONAL E SEGURANÇA DE MENSAGENS

Entrada

Privacidade é a base da confiança. No FriendApp, o chat vibracional só pode existir se cada mensagem for protegida por **camadas sólidas de segurança**, combinando criptografia avançada, firewall vibracional inteligente e políticas rígidas de compliance. Esta camada detalha a engenharia por trás dessa proteção.

Criptografia em Tempo Real

Canal de Dados	Protocolo Aplicado	Observação
Texto de mensagens	AES-256 + HMAC	End-to-End (E2EE)
Áudios e imagens	AES-256 com salt único por envio	Criptografia transitória
Mensagens Pulsar	RAM criptografada com chave volátil	Autodestruição nativa
Logs vibracionais	AES-256 com IDs internos anonimizados	Apenas metadados
Feedbacks pós-chat	AES-256 em repouso + TLS 1.3 em trânsito	Dados opcionais

- **Rotação de chaves:** a cada 24h (automática).

- **Assinatura digital:** SHA-3 para validar integridade.
- **Armazenamento:** somente em servidores certificados (ISO 27001).

Firewall Vibracional Inteligente

O firewall vibracional analisa **metadados + sinais semânticos superficiais**, nunca o conteúdo criptografado. Atua em tempo real para:

- Bloquear **spam energético** (mensagens repetitivas com baixa reciprocidade).
- Interromper **tentativas de manipulação emocional** detectadas por padrão (ex.: insistência após recusa).
- Filtrar **linguagem tóxica** em metadados (palavras-chave ofensivas detectadas antes da criptografia).
- Sugerir **pausa preventiva** ao usuário em colapsos vibracionais.

Exemplo de Pipeline Firewall

Entrada → Análise de metadados (tempo, repetição, emojis)
→ Classificação (normal / alerta / bloqueio)
→ Ação (permitir / logar / bloquear)

Estrutura Técnica de Segurança

```
collection: security_logs
→ log_id (uuid)
→ chat_id (string)
→ tipo_evento: "spam" | "ofensivo" | "colapso" | "normal"
→ acao: "permitido" | "bloqueado" | "alertado"
→ timestamp: datetime
→ ia_confidence: float (0.0–1.0)
```

Exemplo de API — Notificação de Bloqueio

Endpoint: `POST /api/chat/security`

Request:

```
{
  "chat_id": "chat456",
  "user_id": "abc123",
  "evento": "ofensivo_detectado"
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem": "Envio bloqueado por firewall vibracional"
}
```



Compliance e Governança

- **LGPD/GDPR:**

- Consentimento explícito ao abrir chat.
- Direito de exclusão de histórico e feedbacks.
- Logs auditáveis e anonimizados.

- **Políticas de transparência:**

- Usuário é sempre informado em caso de bloqueio.
- Mensagens de erro são amigáveis:

“Essa mensagem não ressoa com a vibração do FriendApp. Tente reformular com mais leveza.”



Fechamento Profissional

O tripé **Criptografia + Firewall Vibracional + Compliance** garante que cada mensagem seja protegida em três dimensões: **tecnológica, ética e vibracional**.

Isso cria um espaço de confiança onde o usuário pode se expressar sem medo.

➡ Na próxima camada (11), vamos detalhar o **Failover Inteligente**, **DRP** e **Backup Geográfico**, assegurando continuidade do sistema mesmo em cenários extremos.

CAMADA 11 — FAILOVER INTELIGENTE, DRP E BACKUP GEOGRÁFICO

Entrada

Mesmo o sistema mais avançado pode enfrentar falhas. No FriendApp, o **Failover Inteligente** e o **DRP (Disaster Recovery Plan)** garantem que o chat vibracional continue funcionando em qualquer cenário, sem perda de dados ou interrupção da experiência do usuário.

Objetivos Principais

1. **Alta Disponibilidade:** manter o chat ativo 24/7 mesmo em falhas regionais.
2. **Resiliência Vibracional:** preservar o estado da conversa e seus logs.
3. **Recuperação Rápida:** failover em segundos, sem percepção do usuário.
4. **Backup Geográfico:** redundância de dados em múltiplas regiões.

Failover Inteligente

- **Monitoramento Contínuo:** latência, uso de CPU, memória e eventos de erro.
- **Aurah Kosmos** atua como sensor, prevendo sobrecargas antes da falha.
- **Tempo de failover:** até **1,8s** para redirecionar usuários a réplicas ativas.

Evento Detectado	Ação do Sistema
Queda de instância primária	Redirecionamento automático para réplica.
Latência acima de 800ms	Redistribuição para regiões alternativas.

Evento Detectado	Ação do Sistema
Erro crítico da IA central	Escalonamento para IA backup (cluster redundante).
Colapso simultâneo em múltiplos chats	Ativação de modo de contenção + cooldown vibracional.

DRP (Disaster Recovery Plan)

Etapas definidas:

1. **Detecção e Isolamento:** sistema identifica falha e isola componente afetado.
2. **Recuperação de Sessão:** logs + metadados restauram estado do chat.
3. **Notificação ao Usuário:** mensagem sutil:

"Tivemos uma breve pausa técnica, mas sua energia foi preservada."
4. **Restabelecimento Completo:** serviço volta a rodar com chave de sessão renovada.

Backup Geográfico

- **Localizações:** São Paulo (AWS), Virgínia (AWS), Frankfurt (GCP).
- **Frequência:**
 - Snapshots a cada 3h.
 - Replicação contínua em tempo real.
- **Dados protegidos:**
 - Mensagens (E2EE).
 - Logs vibracionais.
 - Feedbacks pós-chat.

Região	Tipo de Backup	Delay Máximo
Primária (SP)	Replicação Realtime	0s
Secundária 1	Snapshots diferenciais	3h
Secundária 2	Full Backup Diário	24h

Testes de Resiliência

- **Simulações Semanais:** falhas artificiais testam failover automático.
- **Teste de Carga:** Kafka + Locust simulam picos de tráfego.
- **Auditoria da IA:** Aurah Kosmos registra padrões de falha e propõe melhorias.

Exemplo de API — Status de Failover

Endpoint: `GET /api/system/failover`

Response:


```
{
  "status": "ativo",
  "failover_ativo": true,
  "regiao": "Virginia",
  "latencia_ms": 180
}
```

Segurança e Compliance

- Dados replicados com **criptografia AES-256**.
- Governança de acesso multi-região com **políticas de auditoria**.
- Compatível com **ISO 22301 (Continuidade de Negócios)**.

Fechamento Profissional

O **Failover Inteligente, DRP e Backup Geográfico** transformam o chat do FriendApp em um sistema **resiliente, robusto e inquebrável**, onde a experiência do usuário é preservada mesmo diante de falhas extremas.

 Na próxima camada (12), vamos detalhar a **Performance em Tempo Real e Observabilidade Técnica (<300ms)**.

CAMADA 12 — PERFORMANCE REAL-TIME E OBSERVABILIDADE TÉCNICA (<300ms)

Entrada

Um chat só é confiável se for **rápido e responsivo**. No FriendApp, a meta é clara: **todas as operações principais devem ocorrer em menos de 300ms**. Esta camada define a engenharia de **baixa latência** e o sistema de **observabilidade contínua** que garante desempenho em escala.

Objetivos

1. **Tempo de resposta sub-300ms** em 95% das interações.
 2. **Experiência fluida**: envio e leitura instantâneos.
 3. **Observabilidade total**: métricas e alertas em tempo real.
 4. **Prevenção de gargalos**: balanceamento dinâmico por IA.
-

Metas de Latência (SLA Técnico)

Operação	Tempo Máximo (P95)
Abertura de chat	180ms
Envio + entrega de mensagem	120ms
Processamento de metadados IA	2–3s (assíncrono)
Renderização na UI	70ms

Arquitetura de Performance

- **WebSockets clusterizados**: entrega de mensagens em tempo real.
- **Redis Cache**: armazenamento temporário de sessões e últimos estados vibracionais.
- **CDN Global (Cloudflare)**: distribuição de assets (imagens, áudios, efeitos visuais).

- **Aurah Kosmos Preemptiva:** predição de picos de carga e pré-alocação de recursos.

Observabilidade Técnica

Ferramentas aplicadas:

- **Elastic Stack (ELK):** análise de logs em tempo real.
- **OpenTelemetry:** tracing distribuído ponta a ponta.
- **Prometheus + Grafana:** dashboards de latência, throughput e erros.
- **Aurah Kosmos (modo técnico):** monitora padrões de tráfego anômalos.

Estrutura Técnica — Logs de Performance

```
collection: performance_logs
→ log_id (uuid)
→ chat_id (string)
→ user_id (string)
→ tipo_operacao: "envio" | "abertura" | "render"
→ latencia_ms: int
→ timestamp: datetime
→ status: "sucesso" | "falha"
```

Algoritmo de Balanceamento Dinâmico

```
def balancear_sessao(latencia_ms, cpu_load, usuarios_online):
    if latencia_ms > 300 or cpu_load > 80:
        redirecionar_para_cluster_secundario()
    if usuarios_online > limite_cluster:
        ativar_auto_scaling()
```

Exemplo de API — Métricas de Latência

Endpoint: `GET /api/system/performance`

Response:

```
{
  "status": "sucesso",
  "latencia_media": 142,
  "p95": 280,
  "usuarios_online": 18450,
  "clusters_ativos": 6
}
```



Garantia de Performance

- **Fallback automático:** se latência > 500ms → migrar sessão para cluster secundário.
- **Retry inteligente:** mensagens críticas reprocessadas até 3 vezes.
- **Alertas em tempo real:** devops notificado se latência média subir > 250ms.



Fechamento Profissional

A **Performance Real-Time** e a **Observabilidade Técnica** são os pilares que mantêm o chat vibracional rápido, confiável e transparente. O usuário sente fluidez; os devs têm métricas claras; e a IA previne gargalos.

➡ Na próxima camada (13), exploraremos o **Feedback Vibracional e Check-in Pós-Chat**, conectando experiência do usuário a métricas energéticas.



CAMADA 13— FEEDBACK VIBRACIONAL E CHECK-IN PÓS-CHAT



Entrada

Encerrar um chat não significa apenas “fim da conversa”. No FriendApp, cada término é uma **oportunidade de aprendizado energético**. O sistema de **Feedback Vibracional e Check-in Pós-Chat** coleta percepções do usuário, gera insights pessoais e retroalimenta a IA com dados para melhorar futuros matchings.

Objetivos do Sistema

1. **Autoconsciência do Usuário:** refletir como a conversa impactou seu campo energético.
 2. **Curadoria da IA:** aprimorar o algoritmo de compatibilidade com base em experiências reais.
 3. **Segurança Coletiva:** mapear padrões de colapso ou desconforto.
 4. **Engajamento Premium:** oferecer relatórios e heatmaps de energia pessoais.
-

Fluxo Operacional

1. **Encerramento do Chat** (manual ou por inatividade).
 2. **Prompt Discreto:** *“Como você se sentiu após essa troca?”*
 3. **Opções Rápidas (Pré-configuradas):**
 - 💖 Conexão profunda
 - 🌀 Intensa, mas breve
 - 😐 Neutra
 - 💔 Ruptura
 - ❄️ Fria ou distante
 4. **Campo Livre Opcional:** espaço para comentário/emoji adicional.
 5. **Registro Automático:** dados enviados a Firestore/PostgreSQL.
 6. **Processamento em Lote:** IA Aurah Kosmos correlaciona feedbacks com estados vibracionais detectados.
-

Estrutura Técnica

collection: chat_feedback
→ chat_id (uuid)
→ user_id (uuid)
→ sentimento_final: string (enum)
→ intensidade: float (0.0 – 1.0)
→ tag_encerramento: string
→ comentario_opcional: string
→ timestamp: datetime

Atuação da IA Aurah Kosmos

- **Inputs:** sentimento final, intensidade declarada, intenção inicial e estados vibracionais registrados.
- **Processamento:**
 - Analisa consistência entre percepção do usuário e logs vibracionais.
 - Identifica padrões individuais (ex.: usuário tende a colapsar em chats longos).
- **Outputs:**
 - Ajusta o algoritmo de matching.
 - Sugere novos perfis ou contextos compatíveis.
 - Oferece insights privados:

“Notei que você se sente energizado em conversas rápidas e leves. Deseja priorizar esse tipo de conexão?”

Indicadores de Qualidade

Indicador	Meta Estratégica
Taxa de respostas ao feedback	>70% dos chats finalizados com resposta.
Correlação IA x Usuário	≥85% de alinhamento entre detecção e percepção.
Índice de colapsos precoces	<3% do total de conversas.
Reincidência de desconforto	Perfis sinalizados para análise de moderação.

Exemplo de API

Endpoint: `POST /api/chat/feedback`

Request:

```
{
  "chat_id": "chat987",
  "user_id": "user123",
  "sentimento_final": "💖 Conexão profunda",
  "intensidade": 0.9,
  "comentario_opcional": "Foi uma conversa muito boa"
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem": "Feedback vibracional registrado"
}
```

Privacidade

- Feedback é **privado** e nunca compartilhado com o outro usuário.
- Dados são criptografados e sujeitos a LGPD/GDPR.
- Usuário pode excluir ou editar seu feedback a qualquer momento.
- Insights são **sugeridos** apenas ao usuário dono do perfil.

Fechamento Profissional

O **Feedback Vibracional e Check-in Pós-Chat** transformam cada conversa em aprendizado, equilibrando tecnologia e autoconhecimento. É o ponto em que a experiência pessoal se conecta à inteligência coletiva do ecossistema, sem abrir mão da privacidade.









➡ Na próxima camada (14), vamos detalhar as **Tags de Intenção Energética e o sistema de Autocompletar por IA**, que classificam dinamicamente as conversas.

CAMADA 14 — TAGS DE INTENÇÃO ENERGÉTICA E AUTOCOMPLETAR POR IA

Entrada

As **tags de intenção energética** funcionam como o **vocabulário oculto do FriendApp**. São elas que traduzem a natureza da conversa em metadados claros, permitindo que a IA Aurah Kosmos organize, sugira e reclassifique a energia de cada interação. Nesta camada, detalhamos o catálogo oficial, a lógica de autocompletar por IA e os fluxos técnicos.

Catálogo Oficial de Tags

Tag Vibracional	Descrição Técnica	Exemplo de Aplicação
 Intenção Clara	Conversa objetiva, direcionada, sem rodeios.	"Vamos marcar amanhã às 20h?"
 Casual	Troca leve, sem foco definido.	"E aí, tudo bem?"
 Atração	Energia de flerte, sedução ou paquera.	"Gostei da sua vibe..."
 Profunda	Conexão íntima, emocional ou espiritual.	"Ando refletindo sobre a vida."
 Intelectual	Debate, curiosidades ou troca de ideias.	"Já leu esse artigo de ciência?"
 Resenha	Conversa divertida, piadas e leveza.	"KKK, você viu esse meme?"
 Introspectiva	Reflexão filosófica ou existencial.	"Quem somos além do corpo?"
 Desconforto	Intenção desalinhada, queda vibracional, conflito.	"Não gostei do rumo da conversa."



Estrutura Técnica

```
collection: tags_chat
→ chat_id (string)
→ tag_id (string)
→ nome (string)
→ emoji (string)
→ categoria (string)
→ descrição (string)
→ prioridade_ia (int)
→ atribuicao: "manual" | "automatica"
→ atualizado_em: timestamp
```

- **Manual:** usuário escolhe a intenção no início da conversa.
- **Automática:** IA sugere ou ajusta em tempo real.
- **Prioridade IA:** tags críticas (como ❄️ Desconforto) têm peso maior.



Autocompletar por IA

- **Inputs considerados:** origem do chat (Feed, Bora, Evento), tempo de resposta, emojis, volume inicial de mensagens.
- **Processamento:** Aurah Kosmos aplica embeddings vibracionais e sugere tag inicial.
- **Output:** sugestão exibida ao usuário com score de confiança.

Exemplo de API

Endpoint: `POST /api/chat/autotag`

Request:

```
{
  "chat_id": "xyz123",
  "user_id": "abc456",
  "metadados": {
    "origem": "feed_sensorial",
    "tempo_resposta": 3,
```

```
"emojis": ["🔥", "😍"],  
"volume_inicial": 12  
}  
}
```

Response:

```
{  
  "tag_sugerida": "🔥 Atração",  
  "score_confiança": 0.92,  
  "atribuicao": "automatica"  
}
```



Algoritmo de Reclassificação (Simplificado)

```
def reclassificar_tag(emojis, intensidade, reciprocidade):  
    if "❤️" in emojis or intensidade > 10:  
        return "🔥 Atração"  
    elif reciprocidade > 7:  
        return "💕 Profunda"  
    elif "😞" in emojis:  
        return "❄️ Desconforto"  
    else:  
        return "💬 Casual"
```



Fluxo de Atualização

1. **Chat aberto:** tag inicial atribuída (manual ou automática).
2. **IA monitora metadados:** mudanças de ritmo e expressividade.
3. **Reclassificação dinâmica:** se energia mudar significativamente, a tag é atualizada.

4. **Registro final:** tag dominante armazenada no histórico e usada em matching futuro.

Privacidade e Ética

- Tags nunca são públicas em perfis.
- Só o usuário e a IA têm acesso ao histórico de tags.
- Reclassificação é sempre transparente: usuário é notificado.
- Dados anonimizados quando usados em relatórios agregados.

Fechamento Profissional

As **tags de intenção energética** são a base da curadoria do FriendApp. Elas organizam o caos das interações humanas em padrões claros, permitindo que a IA entenda e sugira caminhos sem nunca interferir no conteúdo.

➡ Na próxima camada (15), exploraremos o **Painel de Status Vibracional do Chat**, que traduz esses estados em uma interface clara e acessível ao usuário.

CAMADA 15 — PAINEL DE STATUS VIBRACIONAL DO CHAT

Entrada






O usuário precisa enxergar, de forma clara e sensorial, como está a energia da conversa em andamento. O **Painel de Status Vibracional** é o componente visual que traduz os estados energéticos do chat em indicadores acessíveis, ajudando a manter a consciência sobre o fluxo sem comprometer a privacidade.

Objetivos do Painel

1. **Transparência:** mostrar em tempo real o estado vibracional da conversa.
2. **Autonomia:** permitir que o usuário perceba quedas e decida agir.

3. **Equilíbrio:** traduzir metadados em representações visuais e intuitivas.
4. **Monetização Premium:** oferecer análises históricas e insights detalhados apenas para assinantes.

Componentes Visuais

Elemento	Função Técnica	Exemplo na UI
 Barra de Ressonância	Representa a força do campo energético (score 0–10).	Pulsa do verde ao vermelho.
 Estado Vibracional	Pico / Transição / Colapso (definido por IA assíncrona).	Ícone colorido no topo.
 Tag Atual	Mostra a intenção ativa (💬 Casual, 🔥 Atração etc.).	Texto + emoji ao lado do nome.
 Indicador de IA	Ícone discreto piscando quando a IA recalcula estados.	Pontinho azul animado.
 Alerta Vibracional	Sugestão de pausa ou insight quando score < 4.0.	Mensagem: "Energia baixa detectada."

Estrutura Técnica (Firestore)

```
collection: chat_status
→ chat_id (uuid)
→ estado_vibracional: "pico" | "transicao" | "colapso"
→ score_ressonancia: float (0.0 – 10.0)
→ tag_ativa: string
→ alerta_ativo: boolean
→ atualizado_em: timestamp
```

Algoritmo de Atualização

```
def atualizar_painel(score, tag):
    if score >= 7:
        estado = "pico"
    elif score >= 4:
```

```
    estado = "transicao"
else:
    estado = "colapso"
return {
    "estado_vibracional": estado,
    "score_ressonancia": score,
    "tag_ativa": tag
}
```

- **Frequência de atualização:** a cada 15 segundos via WebSocket.
- **Pipeline assíncrono:** entrega instantânea de mensagens + cálculo vibracional em até 3s.

Fluxo de Uso

1. Usuário envia mensagem.
2. Sistema atualiza instantaneamente a UI.
3. Pipeline processa metadados.
4. Estado vibracional recalculado.
5. Painel atualizado em tempo real.
6. Alertas surgem em caso de colapso.

Privacidade e Ética

- Estado visível apenas ao usuário (nunca ao outro participante).
- IA atua como guia, nunca força ações.
- Dados criptografados e armazenados apenas como metadados.
- Usuário pode optar por desativar visualizações vibracionais.

Estratégia Premium

Funcionalidade	Disponível em
Status vibracional em tempo real	Gratuito

Funcionalidade	Disponível em
Histórico de estados	Premium
Insights da IA pós-chat	Premium
Recomendações personalizadas	Premium

Fechamento Profissional

O **Painel de Status Vibracional** é o espelho energético da conversa: traduz dados complexos em representações simples, preserva privacidade e amplia a consciência do usuário. Ele transforma o chat em um espaço vivo, onde a energia pode ser percebida e modulada em tempo real.

➡ Na próxima camada (16), exploraremos o **Sistema de Logs Energéticos e Observabilidade de Interações**, que sustenta tecnicamente o painel e a IA.

CAMADA 16 — SISTEMA DE LOGS ENERGÉTICOS E OBSERVABILIDADE DE INTERAÇÕES

Entrada

A confiabilidade de um sistema depende de sua capacidade de **registrar e auditar tudo**. No FriendApp, o **Sistema de Logs Energéticos** documenta cada evento vibracional de uma conversa, criando uma base sólida para segurança, insights e evolução contínua da IA.

Objetivos

1. **Auditabilidade:** permitir rastrear interações sem expor conteúdo privado.
2. **Observabilidade em tempo real:** fornecer visibilidade técnica ao time e à IA.
3. **Segurança vibracional:** detectar comportamentos de risco e padrões de colapso.

4. **Treinamento de IA:** melhorar algoritmos de intenção e compatibilidade.
5. **Insights Premium:** oferecer relatórios energéticos para usuários avançados.

Tipos de Logs Capturados

Tipo de Log	Exemplo Capturado	Finalidade
Intenção inicial	"... Casual" definida ao abrir o chat	Ajustar matching
Mudança de intenção	Reclassificação automática para "🔥 Atração"	Curadoria IA
Reciprocidade	Tempo médio resposta: 2.8s	Score vibracional
Intensidade	Volume: 20 msgs/min (pico)	Estado Pico
Pausas	Intervalo: 15min sem resposta	Estado Colapso
Expressividade	Emojis positivos 😍🔥	Score positivo
Eventos Pulsar	Mensagem expirada em 30s	Auditoria de sigilo
Feedback pós-chat	Usuário marcou ❤️ Ruptura	Refinar IA
Tentativa de print	Bloqueio ativado em mensagem Pulsar	Segurança

Estrutura Técnica dos Logs

```
collection: vibrational_logs
→ log_id (uuid)
→ chat_id (uuid)
→ user_id (uuid)
→ tipo_evento: "intencao" | "emoji" | "pausa" | "pulsar" | "feedback"
→ valor: string | float
→ estado_chat: "pico" | "transicao" | "colapso"
→ timestamp: datetime
→ origem: "mensagem" | "sistema" | "ia"
```

- **Firestore:** armazenamento em tempo real.
- **PostgreSQL:** consolidação para auditoria e relatórios.

- **Neo4j:** análise de padrões relacionais.



Algoritmo Simplificado de Log

```
def registrar_log(chat_id, user_id, evento, valor, estado):  
    log = {  
        "chat_id": chat_id,  
        "user_id": user_id,  
        "tipo_evento": evento,  
        "valor": valor,  
        "estado_chat": estado,  
        "timestamp": datetime.now()  
    }  
    salvar_firestore(log)  
    salvar_postgresql(log)  
    return log
```



Observabilidade Contínua

- **Heatmap Energético:** mapa de picos e colapsos em tempo real.
- **Curvas Temporais:** evolução da vibração por minuto.
- **Taxa de Intervenção da IA:** quantas vezes insights foram sugeridos.
- **Alertas de Anomalia:** silêncio prolongado ou colapso coletivo.



Métricas Estratégicas

Métrica	Uso Técnico
Tempo até colapso	Avaliar resiliência de conexões.
Taxa de feedbacks dados	Medir engajamento do usuário.
Emoções dominantes	Identificar tendências coletivas (anonimizadas).
Logs de Pulsar expirada	Confirmar eficácia do sigilo temporário.



Exemplo de API — Consulta de Logs

Endpoint: `GET /api/chat/logs/{chat_id}`

Response:

```
{
  "status": "sucesso",
  "logs": [
    {
      "log_id": "log987",
      "timestamp": "2025-09-06T18:32:00Z",
      "tipo_evento": "emoji",
      "valor": "😍",
      "estado_chat": "pico"
    }
  ]
}
```

Privacidade e Ética

- Logs **não armazenam conteúdo criptografado**.
- Apenas metadados anonimizados são registrados.
- Usuário pode excluir histórico sob LGPD/GDPR.
- Acesso limitado: somente em casos de denúncia ou para auditoria técnica.

Fechamento Profissional

O **Sistema de Logs Energéticos** garante confiança total: usuários têm segurança, a IA tem insumos para aprendizado e a equipe técnica tem transparência. Cada chat se torna **um registro vibracional auditável e protegido**.

➡ Na próxima camada (17), exploraremos o **Algoritmo de Encerramento Automático e Fechamentos Energéticos**, com ênfase em ética e autonomia do usuário.

CAMADA 17 — ALGORITMO DE ENCERRAMENTO AUTOMÁTICO E FECHAMENTOS ENERGÉTICOS


Entrada

Nem todas as conversas precisam durar para sempre. No FriendApp, o **encerramento de chats** foi projetado para respeitar a autonomia do usuário, mas também oferecer inteligência suficiente para **sugerir pausas ou finais saudáveis**. O objetivo é preservar a vibração positiva e evitar que trocas pesadas drenem a energia dos participantes.

Objetivos do Algoritmo

1. **Autonomia total do usuário:** nenhum chat é encerrado sem ação ou consentimento.
2. **Deteção de padrões de colapso:** IA observa metadados e sugere pausas.
3. **Fechamento elegante:** mensagens suaves e não intrusivas para transições.
4. **Feedback e aprendizado:** todo encerramento gera insights para o usuário e IA.

Critérios Técnicos de Encerramento

Critério	Condição Detectada	Ação da IA
Silêncio prolongado	Sem interação > 24h	Sugere ao usuário encerrar manualmente.
Colapso vibracional	Score < 3.5 por mais de 10 min consecutivos	Exibe insight: <i>"A energia caiu, deseja pausar?"</i>
Feedback negativo	Usuário marca  Ruptura pós-chat	Encerramento imediato, registro no histórico.
Encerramento manual	Usuário clica em <i>"Finalizar chat"</i>	Encerra de forma instantânea.
Eventos críticos	Bloqueio/denúncia → encerramento forçado pelo sistema	Registra log de segurança.



Algoritmo de Detecção (Pseudocódigo)

```
def avaliar_chat(score, tempo_inatividade, feedback):  
    if feedback == "negativo":  
        return "encerrar_imediato"  
    elif tempo_inatividade > 86400: # 24h  
        return "sugerir_encerramento"  
    elif score < 3.5 and tempo_inatividade > 600: # 10 min  
        return "sugerir_pausa"  
    else:  
        return "manter_aberto"
```



Fluxo Operacional

1. **Monitoramento Contínuo:** IA observa metadados em pipeline assíncrono.
2. **Critério acionado:** condição de silêncio ou colapso detectada.
3. **Notificação ao usuário:** sugestão clara e leve:

"Essa conversa parece ter perdido força. Deseja encerrá-la ou dar uma pausa?"
4. **Decisão:** usuário escolhe entre encerrar, pausar ou manter.
5. **Registro:** evento armazenado em logs energéticos e vinculado ao histórico.



Exemplo de API — Encerramento de Chat

Endpoint: `POST /api/chat/end`

Request:

```
{  
  "chat_id": "chat123",  
  "user_id": "user456",  
  "motivo": "manual"  
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem": "Chat encerrado com sucesso."
}
```

Segurança e Ética

- Nenhum encerramento é feito **sem consentimento do usuário**, exceto em casos de denúncia ou bloqueio.
- IA apenas **sugere** pausas/encerramentos, nunca força.
- Transparência: mensagens claras explicam o motivo da sugestão.
- Logs de encerramento são auditáveis e sujeitos a LGPD/GDPR.

Fechamento Profissional

O **Algoritmo de Encerramento Automático** do FriendApp respeita a liberdade do usuário, mas oferece suporte vibracional inteligente para que as conversas terminem com leveza. Essa abordagem equilibra **autonomia, bem-estar e aprendizado coletivo**.

➡ Na próxima camada (18), vamos detalhar o **Sistema de Check-in Energético Final + Feedback Pós-Chat**, aprofundando a coleta de percepções.

CAMADA 18 — SISTEMA DE CHECK-IN ENERGÉTICO FINAL + FEEDBACK PÓS-CHAT

Entrada

Cada conversa deixa uma marca. O **Check-in Energético Final** é a ferramenta que transforma essa marca em **dados estruturados e insights pessoais**,






ajudando usuários a compreenderem seus padrões vibracionais e oferecendo à IA insumos para melhorar o ecossistema.

Objetivos

1. **Autopercepção do Usuário:** registrar como ele se sentiu ao final do chat.
 2. **Treinamento da IA:** correlacionar feedbacks humanos com metadados.
 3. **Segurança Vibracional:** identificar padrões recorrentes de colapso ou desconforto.
 4. **Histórico Energético:** criar linha do tempo vibracional pessoal e coletiva.
-

Fluxo Operacional

1. **Encerramento da Conversa** → manual, inatividade ou denúncia.
 2. **Exibição do Check-in:** tela discreta pergunta:

"Como você se sentiu após essa troca?"
 3. **Respostas Rápidas:** opções visuais com ícones + frases.
 -  Conexão Autêntica
 -  Intensa, mas breve
 -  Neutra
 -  Ruptura
 -  Fria / Sem sintonia
 4. **Campo Livre:** comentário opcional ou emoji extra.
 5. **Registro:** dados criptografados em Firestore + PostgreSQL.
 6. **Processamento em Lote:** IA correlaciona padrões e atualiza algoritmos de matching.
-

Estrutura Técnica (Firestore)

```
collection: chat_checkin
→ chat_id (uuid)
→ user_id (uuid)
```

→ sentimento_final: string
→ intensidade: float (0.0–1.0)
→ comentario_opcional: string
→ estado_chat_final: "pico" | "transicao" | "colapso"
→ timestamp: datetime

Atuação da IA Aurah Kosmos

- **Inputs:** sentimento final, intensidade declarada, estado vibracional final.
- **Processamento:**
 - Compara percepção do usuário com logs vibracionais.
 - Identifica divergências (ex.: IA detectou Pico, usuário sentiu Frieza).
 - Ajusta pesos do algoritmo de matching.
- **Outputs:**
 - Insights pessoais no painel:

“Você tende a se sentir melhor em conversas rápidas e intensas.”
 - Ajustes coletivos no ecossistema: clusters de afinidade.

Indicadores de Qualidade

Indicador	Finalidade Operacional
Taxa de resposta ao check-in	Engajamento do usuário (meta >70%).
Correlação IA x percepção	Medir alinhamento (meta >85%).
Índice de colapso recorrente	Sinalizar perfis/contextos críticos.
Reincidência de desconforto	Ativar protocolos de suporte.

Exemplo de API — Envio de Feedback Final

Endpoint: `POST /api/chat/checkin`

Request:

```
{
  "chat_id": "chat789",
  "user_id": "user123",
  "sentimento_final": "💔 Ruptura",
  "intensidade": 0.3,
  "comentario_opcional": "Energia pesada"
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem": "Check-in registrado com segurança"
}
```



Privacidade

- Feedback nunca é mostrado ao outro participante.
- Usuário pode ignorar sem impacto negativo.
- Dados criptografados e protegidos pela LGPD/GDPR.
- Histórico pessoal pode ser excluído a pedido do usuário.



Fechamento Profissional

O **Check-in Energético Final** é o fecho vibracional do ciclo de conversas. Ele conecta a experiência individual ao aprendizado coletivo, sempre com ética e privacidade. É aqui que o FriendApp evolui a cada interação.

➡ Na próxima camada (19), detalharemos o **Painel Vibracional do Chat e Estados da Conversa**, que mostra ao usuário a energia em tempo real.

CAMADA 19 — PAINEL VIBRACIONAL DO CHAT E ESTADOS DA CONVERSA






Entrada

Durante uma conversa, o usuário precisa ter clareza sobre **como a energia está fluindo em tempo real**. O **Painel Vibracional** é o recurso que traduz metadados em representações visuais, exibindo estados de Pico, Transição e Colapso de forma simples, intuitiva e respeitando a privacidade.

Objetivos

1. **Transparência:** tornar visível o estado vibracional da conversa.
2. **Autonomia:** permitir que o usuário tome decisões com base no fluxo energético.
3. **Consciência Energética:** educar o usuário sobre padrões de interação.
4. **Monetização Premium:** funcionalidades históricas e insights detalhados reservados para assinantes.

Componentes do Painel

Elemento Visual	Função Técnica	Exemplo de Uso
 Barra de Ressonância	Mostra score vibracional (0–10).	Pulsa do verde (alto) ao vermelho (baixo).
 Estado Vibracional	Pico, Transição ou Colapso, conforme cálculo da IA.	Ícone colorido exibido no topo.
 Tag de Intenção Atual	Exibe a intenção ativa (💬 Casual, 🔥 Atração etc.).	Visível no cabeçalho do chat.
 Indicador de IA	Pontinho animado quando a IA recalcula estado vibracional.	Azul piscante.
 Alertas Dinâmicos	Sugestões suaves em caso de queda energética.	"Energia baixa detectada. Deseja pausar?"

Fluxo Técnico de Atualização

1. Usuário envia mensagem.
2. Mensagem é entregue em **tempo real** (WebSocket).
3. Pipeline assíncrono processa metadados → recalcula score.
4. Score vibracional é classificado em **Pico, Transição ou Colapso**.
5. Painel é atualizado na tela em até 3 segundos.
6. Logs são salvos para auditoria futura.

Estrutura Técnica (Firestore)

```
collection: chat_status
→ chat_id (uuid)
→ estado_vibracional: "pico" | "transicao" | "colapso"
→ score_vibracional: float (0.0 – 10.0)
→ tag_ativa: string
→ alerta_ativo: boolean
→ atualizado_em: timestamp
```

Algoritmo de Estado

```
def classificar_estado(score):
    if score >= 7.0:
        return "pico"
    elif score >= 4.0:
        return "transicao"
    else:
        return "colapso"
```

Funcionalidades Premium

Recurso	Disponível em
Estado vibracional em tempo real	Gratuito

Recurso	Disponível em
Histórico de estados do chat	Premium
Insights da IA pós-conversa	Premium
Relatórios de padrões energéticos	Premium

Privacidade

- Estado vibracional é **visível apenas para o usuário** (não compartilhado com a outra parte).
- IA atua como **guia silenciosa**, nunca como moderadora autoritária.
- Dados criptografados, com direito de exclusão sob LGPD/GDPR.

Fechamento Profissional

O **Painel Vibracional do Chat** é o espelho energético da experiência. Ele mostra em tempo real o estado da conexão, preserva a autonomia do usuário e dá ao FriendApp um diferencial estratégico: transformar conversas em dados conscientes e acessíveis.

➡ Na próxima camada (20), vamos explorar como as **Tags de Intenção Energética e a IA de Rotulagem Semântica** trabalham em conjunto para refinar a curadoria de interações.

CAMADA 20 — TAGS DE INTENÇÃO ENERGÉTICA + IA DE ROTULAGEM SEMÂNTICA









Entrada

Se as **Tags de Intenção** são o vocabulário vibracional do FriendApp, a **IA de Rotulagem Semântica** é a tradutora que interpreta sinais e organiza as conversas em categorias claras. Essa camada define como o sistema detecta, atribui e reclassifica intenções ao longo do chat, criando consistência para usuários, IA e banco de dados.

Objetivos

1. **Classificar automaticamente** as conversas com base em sinais energéticos e metadados.
2. **Permitir ajustes manuais** pelos usuários, sempre respeitando autonomia.
3. **Oferecer transparência** sobre como a IA enxerga a intenção dominante.
4. **Retroalimentar o sistema**, ajustando pesos com base em feedbacks reais.

Catálogo de Intenções (Complementar à Camada 06)

Tag Vibracional	Critério Técnico de Identificação	Exemplo Detectado
 Casual	Respostas curtas, emojis neutros, ritmo moderado.	"Oi, tudo bem?"
 Atração	Emojis de paquera (🔥💘), tempo de resposta rápido.	"Quero te ver logo"
 Profunda	Mensagens longas, emojis 💖🙏, reciprocidade alta.	"Tenho refletido sobre nós"
 Intelectual	Uso de palavras-chave técnicas, perguntas complexas.	"Qual sua opinião sobre IA?"
 Resenha	Uso de risadas, memes, emojis 😂😂, leveza na troca.	"KKKK você viu esse vídeo?"
 Objetiva	Ações práticas, linguagem direta.	"Confirma amanhã às 19h?"
 Introspectiva	Expressões filosóficas, retóricas ou reflexivas.	"Quem somos no universo?"
 Desconforto	Pausas longas + feedback neutro ou negativo.	"Não sei se quero continuar"

Estrutura Técnica (Firestore)

```
collection: chat_intencao
→ chat_id (uuid)
→ tag_id: string
→ tag_nome: string
→ atribuicao: "manual" | "automatica"
```

→ score_confiança: float (0.0–1.0)
→ atualizado_em: timestamp

IA de Rotulagem Semântica

Inputs Processados:

- **Metadados:** tempo de resposta, emojis, volume de mensagens.
- **Origem do chat:** Feed, Bora, Evento etc.
- **Feedbacks pós-chat:** percepção explícita do usuário.

Processamento:

- Classificação supervisionada baseada em embeddings vibracionais.
- Reclassificação dinâmica se padrões mudarem.
- Ajuste contínuo de pesos com base em divergência entre IA e feedback humano.

Outputs:

- Tag ativa exibida no Painel Vibracional.
- Score de confiança (0–1).
- Histórico de alterações salvo nos logs.

Algoritmo Simplificado de Classificação

```
def classificar_intencao(emojis, tempo_resposta, volume, feedback):  
    score = 0  
    if "😍" in emojis or "🔥" in emojis:  
        return "atração", 0.9  
    elif volume > 15 and feedback >= 4:  
        return "profunda", 0.85  
    elif tempo_resposta > 300 and feedback <= 2:  
        return "desconforto", 0.7  
    else:
```

```
return "casual", 0.6
```

Exemplo de API — Classificação Automática

Endpoint: `POST /api/chat/classificar_intencao`

Request:

```
{
  "chat_id": "chat456",
  "user_id": "user123",
  "emojis": ["😍", "🔥"],
  "tempo_resposta": 5,
  "volume_mensagens": 20,
  "feedback_previo": 4
}
```

Response:

```
{
  "status": "sucesso",
  "tag_sugerida": "🔥 Atração",
  "score_confiança": 0.92,
  "atribuicao": "automatica"
}
```

Privacidade e Ética

- A classificação é sempre **explicada** ao usuário: mostra tag e score.
- O usuário pode **editar manualmente** se sentir divergência.
- Nenhum dado textual é lido pela IA → apenas metadados e sinais externos.
- Histórico é armazenado de forma criptografada e anonimizável sob LGPD/GDPR.

Fechamento Profissional

A **IA de Rotulagem Semântica** transforma sinais dispersos em categorias claras, tornando as conversas mais compreensíveis e auditáveis. É o filtro que garante consistência entre intenção declarada e energia percebida, respeitando privacidade e autonomia.

➡ Na próxima camada (21), abordaremos o **Sistema de Logs Vibracionais e Observabilidade Energética**, aprofundando o rastreamento e análise contínua.

CAMADA 21 — LOGS VIBRACIONAIS E SISTEMA DE OBSERVABILIDADE ENERGÉTICA

Entrada

Os **logs vibracionais** são a espinha dorsal da confiabilidade do FriendApp. Eles garantem que cada evento seja registrado com precisão, permitindo auditoria, aprendizado da IA e segurança vibracional. Esta camada aprofunda a engenharia do sistema de rastreamento e sua aplicação prática.

Objetivos

1. **Rastrear todos os eventos energéticos** de uma conversa, sem expor conteúdo.
2. **Fornecer observabilidade contínua** para IA e equipe de moderação.
3. **Detectar anomalias** em tempo real (colapsos, abusos, padrões tóxicos).
4. **Oferecer insights** ao usuário Premium (relatórios pessoais energéticos).
5. **Atender compliance** com LGPD/GDPR, garantindo governança.

Tipos de Logs

Tipo de Evento	Exemplo Capturado	Uso Técnico
Intenção inicial	"💬 Casual" definida ao abrir o chat	Matching e curadoria

Tipo de Evento	Exemplo Capturado	Uso Técnico
Mudança de intenção	Atualização para "🔥 Atração"	Reclassificação IA
Reciprocidade	Tempo médio resposta = 2.1s	Score de fluidez
Intensidade	30 msgs/min → estado Pico	Análise vibracional
Pausas críticas	15min sem interação	Sinal de colapso
Emojis dominantes	Uso de 😍🔥 em sequência	Interpretação emocional
Eventos Pulsar	Mensagem expirada em 20s	Sigilo comprovado
Feedback pós-chat	Usuário registrou "💔 Ruptura"	Ajuste de compatibilidade
Tentativa de print	Captura bloqueada em Pulsar	Auditoria de segurança

Estrutura Técnica (Firestore + PostgreSQL)

```
collection: vibrational_logs
→ log_id: uuid
→ chat_id: uuid
→ user_id: uuid
→ tipo_evento: string
→ valor: string | float
→ estado_chat: "pico" | "transicao" | "colapso"
→ origem: "mensagem" | "sistema" | "ia"
→ timestamp: datetime
```

- **Firestore:** captura e visualização em tempo real.
- **PostgreSQL:** armazenamento consolidado e relatórios.
- **Neo4j:** correlação entre padrões de comportamento e clusters.

Algoritmo de Registro

```
def registrar_evento(chat_id, user_id, evento, valor, estado):
    log = {
        "chat_id": chat_id,
        "user_id": user_id,
        "tipo_evento": evento,
```



```
"valor": valor,
"estado_chat": estado,
"timestamp": datetime.now()
}
firestore.save(log)
postgresql.insert(log)
return log
```

Observabilidade Contínua

- **Heatmap energético:** picos e colapsos exibidos em dashboards.
- **Curva temporal:** evolução do estado vibracional por minuto.
- **Alertas:** disparados em caso de anomalias persistentes.
- **Taxa de intervenção da IA:** quantas vezes insights foram sugeridos.

Métricas Internas

Métrica	Aplicação
Tempo até colapso	Avaliar resiliência de conversas.
Emoções dominantes	Relatórios coletivos (dados anonimizados).
Taxa de feedback registrado	Engajamento no check-in vibracional.
Logs de Pulsar expirados	Confirmar eficácia de sigilo temporário.

Exemplo de API

Endpoint: `GET /api/chat/logs/{chat_id}`

Response:

```
{
  "status": "sucesso",
  "logs": [
    {
      "log_id": "log321",
      "timestamp": "2025-09-06T20:30:00Z",
```

```
"tipo_evento": "emoji",  
"valor": "🔥",  
"estado_chat": "pico"  
}  
]  
}
```

Privacidade e Ética

- Logs **não armazenam conteúdo textual criptografado**.
- Apenas metadados anonimizados são usados para análises.
- Usuário pode solicitar exclusão sob LGPD/GDPR.
- Moderadores só acessam logs em caso de denúncia.

Fechamento Profissional

O **Sistema de Logs Vibracionais e Observabilidade** é o guardião invisível do FriendApp. Ele garante segurança, aprendizado e evolução sem violar privacidade. Cada interação se torna um dado auditável, protegido e transformador.

CAMADA 22 — MODO PULSAR: SISTEMA DE MENSAGENS EFÊMERAS E AUTODESTRUTIVAS

Entrada

O **Modo Pulsar** é o mecanismo do FriendApp para mensagens **temporárias e autodestrutivas**, criando um espaço de liberdade, confidencialidade e espontaneidade. Ele é fundamental para garantir que o usuário possa se expressar de forma intensa e momentânea, sem o peso da permanência.

Objetivos

1. **Proteger a privacidade** em interações íntimas ou sensíveis.
2. **Permitir expressão efêmera**, reforçando a liberdade vibracional.
3. **Evitar rastros indesejados**, garantindo sigilo técnico e energético.
4. **Manter segurança máxima**, prevenindo capturas ou cópias não autorizadas.

Características Técnicas

Recurso	Implementação
Autodestruição	Mensagens expiram após leitura ou tempo definido.
Cache transitório	Armazenadas apenas em RAM com chave volátil.
Não persistência	Não entram em Firestore/PostgreSQL (exceto logs de auditoria sem conteúdo).
Proteção contra print	Tentativas bloqueadas e logadas em auditoria.
UI dedicada	Balões pulsantes, timer regressivo, animação de dissipação.

Estrutura de Dados

```
temp_collection: pulsar_messages
→ chat_id (uuid)
→ msg_id (uuid)
→ sender_id (uuid)
→ tipo: "text" | "image" | "audio" | "emoji"
→ expiration_type: "on_view" | "timed"
→ expiration_value: int (segundos)
→ status: "ativa" | "expirada" | "lida"
→ criado_em: timestamp
```

- **Logs de auditoria** registram apenas o evento de expiração, nunca o conteúdo.

Atuação da IA Aurah Kosmos

- Sugere o **Modo Pulsar** em mensagens com alta intensidade (ex.: emojis 🤔🔥).
- Monitora frequência de uso e oferece insights pós-chat:

"Suas últimas conversas tiveram muitas mensagens efêmeras. Deseja manter esse padrão ou buscar conexões mais consistentes?"
- Não força ativação: sempre opcional ao usuário.

Segurança Avançada

Risco Identificado	Mitigação
Print de tela	Bloqueio automático + alerta ao remetente.
OCR/IA externas	Firewall antissequestro detecta padrões suspeitos.
Tentativa de backup	Não há persistência em bancos principais.
Compartilhamento indevido	Logs registram o evento, podendo acionar suporte vibracional.

Casos de Uso

Cenário	Benefício Vibracional
Envio de nudes	Liberdade com confiança e sigilo absoluto.
Desabafos emocionais	Catarse vibracional sem rastros permanentes.
Conversas de impacto imediato	Intensidade momentânea que se dissolve no tempo.
Interações confidenciais	Mensagens sensíveis protegidas de vazamentos.

Exemplo de API — Envio de Mensagem Pulsar

Endpoint: `POST /api/chat/pulsar`

Request:

```
{
  "chat_id": "chat789",
  "user_id": "user456",
```

```
"mensagem": "Essa mensagem desaparecerá em 20s",
"tipo": "text",
"expiration_type": "timed",
"expiration_value": 20
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem_id": "uuid",
  "autodestruir_em": 20
}
```



Privacidade e Ética

- Modo Pulsar é ativado **apenas pelo usuário**, nunca pela IA.
- Conteúdo é **inacessível a terceiros** (nem suporte, nem servidores).
- Logs de auditoria são **anonimizados** e criptografados.
- Recurso premium em alguns casos (histórico de uso e estatísticas).



Fechamento Profissional

O **Modo Pulsar** dá ao usuário poder de controlar o tempo de vida de suas mensagens, equilibrando intensidade com segurança. É a válvula de escape do FriendApp: um espaço para viver a energia do momento sem o peso da permanência.



Na próxima camada (23), detalharemos o **Painel de Feedback Vibracional Pós-Chat**, que transforma a experiência em aprendizado pessoal e coletivo.



CAMADA 23 — PAINEL DE FEEDBACK VIBRACIONAL PÓS-CHAT


Entrada

O encerramento de uma conversa é tão importante quanto seu início. O **Painel de Feedback Vibracional Pós-Chat** transforma a experiência em **aprendizado prático**, coletando percepções do usuário, gerando insights pessoais e alimentando a IA Aurah Kosmos com dados para aprimorar futuras conexões.

Objetivos

1. **Autopercepção:** permitir que o usuário reflita sobre a energia da conversa.
 2. **Aprendizado da IA:** correlacionar feedback humano com metadados.
 3. **Histórico Vibracional:** enriquecer a linha do tempo pessoal de conexões.
 4. **Engajamento Premium:** oferecer relatórios energéticos exclusivos.
-

Componentes do Painel

Elemento	Função Técnica	Exemplo
Escala de Emoções	Ícones pré-definidos para seleção rápida.	
Campo Livre	Comentário opcional do usuário.	"Foi intenso mas cansativo."
Curva de Energia	Gráfico do score vibracional ao longo do chat (Premium).	Pico → Colapso
Insight Pessoal	Mensagem automática da IA baseada no padrão da conversa.	"Você se conecta melhor em diálogos breves."

Fluxo Operacional

1. Chat é encerrado.
 2. Painel de feedback aparece automaticamente.
 3. Usuário escolhe opção rápida ou escreve comentário.
 4. Dados são criptografados e enviados para Firestore/PostgreSQL.
 5. IA processa em lote e gera insights pessoais + ajustes no matching.
 6. Painel histórico é atualizado para usuários Premium.
-

Estrutura Técnica

```
collection: feedback_postchat
→ chat_id (uuid)
→ user_id (uuid)
→ sentimento_final: string
→ score_energia: float
→ comentario: string
→ curva_energia: array<float>
→ timestamp: datetime
```

Atuação da IA Aurah Kosmos

- **Inputs:** sentimento final, curva de energia, comentários do usuário.
- **Processamento:** identifica padrões recorrentes (ex.: tendência a rupturas rápidas).
- **Outputs:**
 - Recomendações no painel do usuário.
 - Ajustes de matching para evitar repetições negativas.
 - Insights coletivos para relatórios vibracionais do ecossistema.

Exemplo de API

Endpoint: `POST /api/chat/feedback`

Request:

```
{
  "chat_id": "chat321",
  "user_id": "user789",
  "sentimento_final": "🌟 Intensa, mas breve",
  "score_energia": 6.7,
  "comentario": "Boa energia, mas cansativa"
```

```
}
```

Response:

```
{  
  "status": "sucesso",  
  "mensagem": "Feedback registrado com segurança"  
}
```

Indicadores Internos

Indicador	Meta Estratégica
Taxa de feedbacks respondidos	>70% dos chats finalizados.
Correlação IA x percepção	≥85% de alinhamento.
Padrões de colapso recorrente	Perfis sinalizados para moderação.
Engajamento Premium	Relatórios vibracionais exclusivos.

Privacidade

- Feedback é **privado**, não compartilhado com outro participante.
- Usuário pode ignorar o painel sem prejuízo.
- Dados anonimizados em relatórios coletivos.
- Regidos pela LGPD/GDPR com direito à exclusão.

Fechamento Profissional

O **Painel de Feedback Pós-Chat** é a ponte entre experiência e aprendizado. Ele fortalece a consciência individual e fornece insumos valiosos para a IA, sempre com transparência e respeito à privacidade.

➡ Na próxima camada (24), veremos o **Histórico de Chats e o Heatmap Vibracional**, que transformam feedbacks e logs em memória energética visual.

CAMADA 24 — HISTÓRICO DE CHATS E HEATMAP VIBRACIONAL

Entrada

Cada conversa no FriendApp deixa um registro energético. O **Histórico de Chats** e o **Heatmap Vibracional** organizam essas memórias em dados claros, permitindo que o usuário veja suas conexões passadas, enquanto a IA Aurah Kosmos aprende padrões e sugere novas experiências alinhadas.

Objetivos

1. **Memória Energética:** manter registros resumidos das conversas anteriores.
2. **Heatmap Visual:** mostrar onde e quando interações vibracionais aconteceram.
3. **Autoconhecimento:** oferecer insights sobre padrões de conexões.
4. **Matching Avançado:** IA utiliza histórico para refinar recomendações.
5. **Governança:** garantir rastreabilidade, sem violar privacidade.

Estrutura do Histórico

Elemento	Detalhe Capturado	Exemplo
chat_id	Identificador único	chat123
user_id_parceiro	Identidade do outro participante (oculto para IA agregada)	user456
data_inicio	Início da conversa	2025-09-06T18:20:00Z
data_fim	Fim da conversa	2025-09-06T19:05:00Z
tags_vibracionais	Intenções associadas ao chat	["🔥 Atração", "💬 Casual"]
score_final	Score médio vibracional	6.8
estado_final	Pico / Transição / Colapso	Transição
feedback_usuario	Sentimento registrado no check-in	💖 Conexão Autêntica

Heatmap Vibracional

- Representa a distribuição de chats no **tempo** e **espaço**.
 - Exibe zonas de maior fluxo energético em mapas:
 - Verde = expansão.
 - Amarelo = transição.
 - Vermelho = colapso.
 - Atualizado em tempo real pela IA com base em check-ins e logs.
-

Estrutura Técnica (Firestore + Neo4j)

```
collection: chat_history
→ chat_id (uuid)
→ user_id (uuid)
→ parceiro_id (uuid)
→ inicio: datetime
→ fim: datetime
→ tags: array<string>
→ score_final: float
→ estado_final: string
→ feedback: string
```

Neo4j (grafo):

```
(user)-[:CONVERSOU_COM {score: 6.8, estado: "transicao"}]→(user)
```

Algoritmo de Geração de Heatmap

```
def gerar_heatmap(chats):
    mapa = {}
    for chat in chats:
        local = chat.get("localizacao")
        estado = chat.get("estado_final")
        if local not in mapa:
```

```
mapa[local] = {"pico":0, "transicao":0, "colapso":0}
mapa[local][estado] += 1
return mapa
```

Exemplo de API — Consulta do Histórico

Endpoint: `GET /api/chat/history/{user_id}`

Response:

```
{
  "status": "sucesso",
  "historico": [
    {
      "chat_id": "chat123",
      "parceiro": "Ana Paula",
      "estado_final": "pico",
      "score_final": 8.4,
      "feedback": "💖 Conexão Autêntica"
    }
  ]
}
```

Funcionalidades Premium

Recurso	Gratuito	Premium
Lista básica de chats anteriores	✓	✓
Heatmap vibracional detalhado	✗	✓
Relatórios de padrões pessoais	✗	✓
Recomendações baseadas no histórico	✗	✓

Privacidade e Ética

- Histórico **visível apenas ao usuário dono**.

- Parceiros não têm acesso ao feedback ou score final.
 - Dados anonimizados em relatórios coletivos.
 - Exclusão total sob LGPD/GDPR garantida.
-

Fechamento Profissional

O **Histórico de Chats** e o **Heatmap Vibracional** transformam memórias em inteligência. Eles ajudam o usuário a entender suas conexões passadas, oferecem relatórios de autoconhecimento e permitem que a IA refine o matching com base em dados reais — sempre preservando a privacidade.

➡ Na próxima camada (25), detalharemos o **Painel Vibracional Ativo e Visível Durante os Chats**, que complementa a experiência em tempo real.

CAMADA 25 — PAINEL VIBRACIONAL ATIVO E VISÍVEL DURANTE OS CHATS






Entrada

Durante uma conversa, o usuário não deve apenas **trocar mensagens**, mas também **sentir a energia**. O **Painel Vibracional Ativo** é o recurso visual e dinâmico que traduz estados vibracionais em tempo real, ampliando a consciência do participante sem comprometer a privacidade.

Objetivos

1. **Clareza energética:** mostrar como a energia da conversa oscila em tempo real.
 2. **Autonomia:** permitir que o usuário escolha pausar, encerrar ou ajustar o tom da interação.
 3. **Educação vibracional:** ensinar padrões de reciprocidade, intensidade e pausas.
 4. **Valor Premium:** entregar análises históricas e relatórios avançados para assinantes.
-

Componentes do Painel Ativo

Elemento Visual	Função Técnica	Exemplo de Uso
 Medidor de Energia	Barra pulsante que reflete o score vibracional (0–10).	Verde (alto) → Vermelho (baixo).
 Estado Atual	Pico / Transição / Colapso, exibido em ícone + texto.	"🔥 Pico de Conexão".
 Tag de Intenção	Mostra a intenção dominante da conversa.	"💖 Profunda".
 Alertas Dinâmicos	Notificações suaves em caso de queda energética.	"Energia em queda, deseja pausar?"
 Histórico Instantâneo	Mini gráfico de evolução dos últimos 5 minutos (Premium).	Linha pulsante em tempo real.

Fluxo Técnico

1. Usuário envia mensagem.
2. Mensagem entregue via WebSocket (latência < 150ms).
3. Metadados processados em pipeline assíncrono (Aurah Kosmos).
4. Estado vibracional atualizado no painel em até 3 segundos.
5. Logs salvos no Firestore/PostgreSQL para auditoria.
6. Premium: histórico visual exibido ao longo da sessão.

Estrutura Técnica (Firestore)

```
collection: painel_vibracional
→ chat_id (uuid)
→ score_vibracional: float
→ estado_atual: "pico" | "transicao" | "colapso"
→ tag_dominante: string
→ alerta_ativo: boolean
→ timestamp: datetime
```

Algoritmo de Atualização

```
def atualizar_estado(score):  
    if score >= 7:  
        return "pico"  
    elif score >= 4:  
        return "transicao"  
    else:  
        return "colapso"
```

- Executado a cada ciclo de análise (~15s).
- Score calculado com base em reciprocidade, intensidade, pausas e emojis.

Funcionalidades Premium

Recurso	Gratuito	Premium
Estado vibracional em tempo real	✓	✓
Histórico de evolução no chat	✗	✓
Relatórios pós-chat da IA	✗	✓
Sugestões proativas avançadas	✗	✓

Privacidade

- Estados são **visíveis apenas ao usuário**.
- Outro participante não tem acesso ao painel vibracional.
- Dados criptografados e armazenados como metadados.
- Usuário pode optar por desativar exibição em tempo real.

Fechamento Profissional

O **Painel Vibracional Ativo** é o espelho vivo da energia no FriendApp. Ele amplia a consciência, respeita a privacidade e entrega valor estratégico, especialmente na versão Premium. O chat deixa de ser apenas troca de palavras e passa a ser uma experiência **medida, sentida e compreendida**.

➡ Na próxima camada (26), estruturaremos a **Seção Final de Integrações e Dependências Cíclicas do Ecossistema**, consolidando como o Chat se conecta a todos os outros módulos do FriendApp.

CAMADA 26 — INTEGRAÇÕES E DEPENDÊNCIAS CÍCLICAS DO ECOSSISTEMA

Entrada

Nenhum sistema no FriendApp opera isolado. O **Chat Vibracional** é um dos módulos mais interligados de todo o ecossistema, recebendo dados de diversas fontes e enviando outputs que alimentam outros subsistemas. Esta camada mapeia as **entradas, saídas, triggers e dependências cíclicas**, garantindo rastreabilidade total para desenvolvedores e clareza estratégica.

Objetivos da Seção

1. **Mapear integrações diretas** do Chat com outros módulos.
2. **Detalhar entradas (inputs) e saídas (outputs)** de dados.
3. **Explicar o tipo de integração**: API, trigger de IA, eventos, sincronização em tempo real.
4. **Garantir rastreabilidade**: dev nunca mais perguntará “de onde vem esse dado?”.

Entradas (Inputs do Chat)

Origem	Dados Fornecidos	Tipo de Integração
Feed Sensorial	Intenção ativa em postagens	Trigger IA + API (<code>/chat/create</code>)
Modo Bora	Aceitação mútua de encontro	Evento direto (<code>/chat/bora</code>)
Eventos e Locais	Check-ins simultâneos, tag base vibracional	Trigger de geofencing + API

Origem	Dados Fornecidos	Tipo de Integração
Mapa de Frequência	Densidade de usuários próximos	Sugestão de chat via trigger IA
Aurah Kosmos (IA)	Compatibilidade vibracional sugerida	Trigger assíncrono + insights
Histórico do Usuário	Reabertura de conversas antigas	API <code>/chat/history</code>

Saídas (Outputs do Chat)

Destino	Dados Enviados	Utilização Posterior
Painel Vibracional	Estados em tempo real (Pico, Transição, Colapso)	Exibição ao usuário
Aurah Kosmos (IA)	Logs de intenção e score vibracional	Aprendizado contínuo da IA
Mapa de Frequência	Zonas ativas de chats	Atualização em tempo real
Feed Sensorial	Insights de interações	Sugestão de novas conexões
Sistema de Reputação	Dados de colapso, bloqueio e feedback negativo	Score de confiança do usuário
Sistema de Moderação	Flags de abusos, prints em Pulsar, denúncias	Ações corretivas
Advisor Pessoal (Premium)	Relatórios energéticos pós-chat	Insights personalizados

Triggers e Eventos

- `onIntentDetected` → abertura de chat via Feed.
- `onMutualAcceptance` → criação de chat pelo Modo Bora.
- `onCheckInMatch` → chat por evento/local parceiro.
- `onConversationCollapse` → alerta enviado à IA + moderação.
- `onChatEnd` → envio de dados para Feedback e Histórico.

Tipos de Integração

Tipo Técnico	Descrição	Exemplo
API REST	Criação, envio de mensagens, encerramento.	<code>/api/chat/create</code> , <code>/api/chat/send</code>
Triggers IA	Sugestões automáticas da Aurah Kosmos.	"Energia compatível detectada."
Eventos em Tempo Real	Ativação por check-ins, cliques em Feed, aceitação em Bora.	<code>onCheckInMatch</code>
Sincronização Banco	Estados vibracionais atualizados em Firestore/PostgreSQL/Neo4j.	<code>collection: chat_status</code>

Exemplo Visual Simplificado


Feed Sensorial → (Trigger) → Chat Vibracional → Painel Vibracional
 Modo Bora → (Evento) → Chat Vibracional → Histórico/Feedback
 Eventos/Locais → (Geofencing) → Chat Vibracional → Mapa de Frequência
 Aurah Kosmos → (Sugestão) → Chat Vibracional → IA Learning

Governança e Privacidade

- Integrações respeitam **E2EE** (IA só acessa metadados).
- Todos os eventos são **logados** para auditoria.
- Dados seguem LGPD/GDPR com exclusão sob demanda.

Fechamento Profissional

O **Mapa de Integrações do Chat** garante que cada fluxo de dados seja rastreável, auditável e tecnicamente claro. Ele conecta o coração vibracional do FriendApp aos demais módulos, transformando o chat em **fonte e destino de inteligência energética** no ecossistema.

 Na sequência, podemos estruturar as **Camadas Extras** (Contratos de API, Algoritmos-Chave, Mensagens de Erro e Glossário Vibracional), consolidando o manual como 100% à prova de dúvidas.

CAMADAS EXTRAS

◆ Camada Extra 01 — Contratos Técnicos de API (Chat e Mensagens)

- Endpoints completos (`/chat/create` , `/chat/send` , `/chat/end` , `/chat/pulsar` , etc.).
 - Estrutura de **request/response**.
 - Exemplos de **erros tratados**.
 - Tabelas de parâmetros obrigatórios/opcionais.
-

◆ Camada Extra 02 — Algoritmos-Chave do Chat Vibracional

- Fórmulas detalhadas (frequência vibracional, colapso, reclassificação de intenção).
 - Inputs → processamento → outputs, no estilo “receita do bolo”.
 - Pseudocódigos claros para devs.
 - Explicação da arquitetura assíncrona.
-

◆ Camada Extra 03 — Mensagens de Erro e Fluxos para o Usuário

- Tabela com **condições de erro**, mensagens exibidas e ações do sistema.
 - Ex.: usuário sem verificação tentando abrir chat, mensagens em colapso, etc.
 - Design de mensagens de erro **amigáveis e vibracionais**, no tom FriendApp.
-

◆ Camada Extra 04 — Glossário Vibracional Único

- Consolidação de todos os termos proprietários (Modo Pulsar, Firewall Vibracional, Check-in Energético, etc.).
 - Definições técnicas e uso em sistema.
 - Padrão de referência única para todo o ecossistema.
-

◆ Camada Extra 05 — Privacidade, Ética e Arquitetura Assíncrona

- Correção conceitual (IA só em metadados, nunca conteúdo).
- Modelo de análise assíncrona (Kafka/PubSub).

- Garantias de **E2EE + LGPD/GDPR**.
- Regras para insights pós-chat (sem controle autoritário)

CAMADA EXTRA 01 — CONTRATOS TÉCNICOS DE API (CHAT E MENSAGENS)

Entrada

O sistema de chat vibracional só é executável porque possui **contratos de API claros, padronizados e auditáveis**. Esta camada define cada endpoint, seus parâmetros obrigatórios, exemplos de requisição/resposta e cenários de erro. O objetivo é eliminar dúvidas dos devs e garantir consistência em todas as integrações.

Objetivos

1. **Padronizar** todos os contratos REST do chat.
2. **Definir requests e responses** em formato JSON validado.
3. **Mapear erros comuns** e mensagens exibidas ao usuário.
4. **Garantir consistência** entre frontend, backend e IA.

Endpoints Principais

1. Criar Novo Chat

POST `/api/chat/create`

Request Body:

```
{
  "user_id": "uuid",
  "target_id": "uuid",
  "origem": "feed_sensorial",
  "tag_intencao": "💬 Casual"
```

```
}
```

Response (Sucesso):

```
{
  "status": "sucesso",
  "chat_id": "uuid",
  "estado_inicial": "transicao"
}
```

Response (Erro):

```
{
  "status": "erro",
  "mensagem": "Usuário não verificado."
}
```

2. Enviar Mensagem

POST `/api/chat/send`

Request Body:

```
{
  "chat_id": "uuid",
  "user_id": "uuid",
  "conteudo": "Oi! Tudo bem?",
  "tipo": "texto",
  "intencao": "💬 Casual"
}
```

Response (Sucesso):

```
{
  "status": "sucesso",
```

```
"mensagem_id": "uuid",
"timestamp": "2025-09-06T20:30:00Z"
}
```

Response (Erro):

```
{
  "status": "erro",
  "mensagem": "Chat encerrado ou inativo."
}
```

3. Encerrar Chat

POST `/api/chat/end`

Request Body:

```
{
  "chat_id": "uuid",
  "user_id": "uuid",
  "motivo": "manual"
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem": "Chat encerrado com sucesso."
}
```

4. Listar Chats Ativos

GET `/api/chat/listar/{user_id}`

Response (Sucesso):

```
{
  "status": "sucesso",
  "chats": [
    {
      "chat_id": "uuid",
      "intencao": "🔥 Atração",
      "estado": "pico",
      "ultimo_update": "2025-09-06T19:55:00Z"
    }
  ]
}
```

5. Enviar Mensagem Pulsar

POST `/api/chat/pulsar`

Request Body:

```
{
  "chat_id": "uuid",
  "user_id": "uuid",
  "mensagem": "Essa desaparecerá em 30s",
  "tipo": "texto",
  "expiration_type": "timed",
  "expiration_value": 30
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem_id": "uuid",
  "autodestruir_em": 30
}
```

6. Enviar Feedback Pós-Chat

POST `/api/chat/feedback`

Request Body:

```
{
  "chat_id": "uuid",
  "user_id": "uuid",
  "sentimento_final": "💔 Ruptura",
  "intensidade": 0.3,
  "comentario": "Não fluiu bem"
}
```

Response:

```
{
  "status": "sucesso",
  "mensagem": "Feedback registrado com segurança."
}
```



Tabela de Erros Comuns

Cenário de Erro	Código HTTP	Mensagem JSON
Usuário não autenticado	401	<code>{"status":"erro","mensagem":"Token inválido"}</code>
Usuário não verificado (DUC)	403	<code>{"status":"erro","mensagem":"Usuário não verificado."}</code>
Chat não encontrado	404	<code>{"status":"erro","mensagem":"Chat inexistente."}</code>
Chat encerrado/inativo	410	<code>{"status":"erro","mensagem":"Chat encerrado ou inativo."}</code>
Payload inválido	422	<code>{"status":"erro","mensagem":"Dados inválidos."}</code>
Erro interno de servidor	500	<code>{"status":"erro","mensagem":"Erro inesperado, tente novamente."}</code>

Segurança

- Todas as rotas exigem **token JWT válido**.
 - Payloads validados por **JSON Schema**.
 - Tentativas de abuso são registradas em **security_logs**.
-

Fechamento Profissional

A **Camada Extra 01** garante que o sistema de chat vibracional seja **tecnicamente executável** por qualquer dev, com contratos claros, erros padronizados e segurança embutida. É a fundação de integração entre frontend, backend e IA.

➡ Na próxima camada extra (02), vamos detalhar os **Algoritmos-Chave do Chat Vibracional**, que processam estados energéticos, intenções e colapsos.

CAMADA EXTRA 02 — ALGORITMOS-CHAVE DO SISTEMA DE CHAT VIBRACIONAL

Entrada

Para além da infraestrutura, o que diferencia o FriendApp é a **inteligência vibracional aplicada**. Esta camada descreve os **algoritmos centrais** que processam metadados, detectam estados, classificam intenções e previnem colapsos. Não é código pronto, mas sim a “receita do bolo” que garante clareza para devs.

Objetivos

1. Explicar os **inputs, processamento e outputs** de cada algoritmo.
 2. Padronizar fórmulas e cálculos de score vibracional.
 3. Demonstrar pseudocódigos claros e executáveis.
 4. Reduzir dúvidas técnicas e evitar ambiguidades na execução.
-

Algoritmo 01 — Cálculo de Frequência Vibracional

Inputs (ingredientes):

- Tempo de resposta médio (ms).
- Volume de mensagens/minuto.
- Emojis positivos/negativos.
- Pausas > 5 minutos.
- Feedback pós-chat (opcional).

Processamento (modo de preparo):

- Normalizar valores (0 a 10).
- Aplicar pesos: reciprocidade (0.35), intensidade (0.25), emojis (0.20), pausas (-0.15), feedback (0.05).

Output (bolo pronto):

- Score de 0.0 a 10.0.
- Estados:
 - ≥ 7.0 = Pico
 - 4.0–6.9 = Transição
 - < 4.0 = Colapso

Pseudocódigo:

```
def calcular_frequencia(resp, volume, emojis, pausas, feedback):  
    score = (0.35 * normalizar(resp)) + \  
            (0.25 * normalizar(volume)) + \  
            (0.20 * score_emojis(emojis)) - \  
            (0.15 * pausas) + \  
            (0.05 * feedback)  
    return round(score,2)
```

Algoritmo 02 — Detecção de Colapso Vibracional

Inputs:

- Score vibracional calculado.
- Tempo de inatividade.
- Taxa de reciprocidade.

Processamento:

- Se score < 3.5 por 10 min consecutivos OU inatividade > 15 min → sinalizar colapso.

Output:

- Insight sugerido ao usuário: *"A energia caiu. Deseja pausar ou encerrar?"*

Pseudocódigo:

```
def detectar_colapso(score, inatividade, reciprocidade):
    if score < 3.5 and inatividade > 600:
        return "colapso"
    elif reciprocidade < 0.3:
        return "alerta"
    else:
        return "estavel"
```

Algoritmo 03 — Reclassificação de Intenção

Inputs:

- Emojis dominantes.
- Volume de mensagens.
- Feedback pós-chat.

Processamento:

- Se volume > 15 e emojis positivos = Profunda.
- Se emojis 🔥💕 em destaque = Atração.
- Se muitos 😐😞 + pausas longas = Desconforto.

Output:

- Nova tag de intenção associada ao chat.

Pseudocódigo:

```
def reclassificar_intencao(emojis, volume, feedback):  
    if "😍" in emojis or "🔥" in emojis:  
        return "atração"  
    elif volume > 15 and feedback >= 4:  
        return "profunda"  
    elif "😞" in emojis or feedback <= 2:  
        return "desconforto"  
    else:  
        return "casual"
```

Algoritmo 04 — Encerramento Inteligente

Inputs:

- Score vibracional.
- Tempo de inatividade.
- Feedback negativo.

Processamento:

- Encerramento nunca automático: apenas sugestão ao usuário.
- Regras:
 - 24h de silêncio → sugestão de encerrar.
 - Score < 3.5 por 10 min → sugestão de pausa.
 - Feedback = negativo → encerramento manual sugerido.

Output:

- Ação recomendada: manter, pausar ou encerrar.

Algoritmo 05 — Sugestão de Matching Pós-Chat

Inputs:

- Feedback final do usuário.
- Tags de intenção do histórico.

- Score médio vibracional.

Processamento:

- IA compara padrões com base coletiva.
- Sugere novos perfis/locais compatíveis.

Output:

- Recomendações exibidas no Feed Sensorial e no Mapa de Frequência.

Fechamento Profissional

Os **Algoritmos-Chave do Chat Vibracional** são a lógica que garante previsibilidade, clareza e segurança para devs. Definindo inputs, processamento e outputs, evitamos lacunas e criamos um sistema pronto para execução.

➡ Na próxima camada extra (03), estruturaremos as **Mensagens de Erro e Fluxos para o Usuário**, garantindo experiência clara mesmo em falhas.

CAMADA EXTRA 03 — MENSAGENS DE ERRO E FLUXOS PARA O USUÁRIO

Entrada

Mesmo sistemas perfeitos podem gerar erros — e o que define a experiência do usuário é **como esses erros são comunicados**. O FriendApp transforma mensagens de erro em **orientações vibracionais claras, acolhedoras e técnicas**, sem jargões ou opacidade.

Objetivos

1. **Clareza:** garantir que o usuário saiba o que aconteceu.
2. **Ação imediata:** sempre oferecer um próximo passo.
3. **Tom vibracional:** mensagens leves, acolhedoras e consistentes com o FriendApp.

4. Padronização: erros estruturados em API + mensagens visuais no app.

Tabela de Erros Críticos e Fluxos

Condição de Erro	Mensagem na Tela	Ação do Sistema
Usuário não autenticado	"⚠️ Sua sessão expirou. Faça login novamente para manter sua energia ativa."	Redireciona para tela de login.
Usuário não verificado (DUC/DCO)	"🔒 Apenas perfis verificados podem abrir este portal vibracional. Toque aqui e conclua sua verificação."	Redireciona para fluxo de verificação.
Chat inexistente ou ID inválido	"❌ Não encontramos este chat. Ele pode ter sido encerrado ou removido."	Sugere voltar ao Feed Sensorial.
Chat já encerrado	"🔔 Esta conversa já foi finalizada. Deseja abrir um novo fluxo com essa pessoa?"	Botão: "Iniciar novo chat".
Tentativa de envio em estado de colapso	"⚠️ Este espaço entrou em colapso vibracional. Pausa sugerida para preservar sua energia."	Campo de mensagem é desativado temporariamente.
Upload de mídia não suportada	"📷 Formato não aceito. Use imagens, áudios ou emojis compatíveis com a vibração do sistema."	Reabre seletor de mídia.
Tentativa de print em Modo Pulsar	"🛡️ Para proteger sua conexão, não é permitido capturar este momento efêmero."	Loga tentativa em security_logs.
Erro interno inesperado (500)	"⚠️ Tivemos uma oscilação técnica. Respire fundo e tente novamente em instantes."	Retry automático até 3 vezes.

Estrutura Técnica de Respostas de Erro (API)

Formato Padrão JSON:

```
{  
  "status": "erro",
```

```
"codigo": "403",
"mensagem": "Usuário não verificado para iniciar chat.",
"acao_sugerida": "verificacao_documental"
}
```

Campos Obrigatórios:

- `status` : "erro"
- `codigo` : HTTP status (401, 403, 404, 410, 422, 500)
- `mensagem` : descrição em linguagem natural (amigável)
- `acao_sugerida` : chave para UI tomar ação automática



Fluxos de Tratamento no App

1. **Erro detectado na API** → UI exibe mensagem vibracional adaptada.
2. **Ação sugerida** → botão/link para próximo passo (login, feed, verificação, retry).
3. **Log de auditoria** → evento salvo em `error_logs` com `user_id` e contexto.
4. **Feedback do usuário** (opcional) → botão "reportar problema".



Estrutura de Logs de Erro

```
collection: error_logs
→ log_id (uuid)
→ user_id (uuid)
→ chat_id (uuid/null)
→ codigo_http: int
→ mensagem: string
→ acao_sugerida: string
→ timestamp: datetime
```



Privacidade e Ética

- Nenhum erro expõe detalhes sensíveis.
 - Logs são criptografados e anonimizados em relatórios coletivos.
 - Usuário pode consultar erros críticos que afetaram sua conta.
 - Tratamento em conformidade com LGPD/GDPR.
-

Fechamento Profissional

O **sistema de Mensagens de Erro e Fluxos** do FriendApp transforma falhas em **momentos de acolhimento**. Em vez de frustração, o usuário recebe clareza, orientação e uma ação prática. Para devs, os contratos são objetivos; para usuários, a experiência é leve.

➡ Na próxima camada extra (04), vamos consolidar o **Glossário Vibracional Único**, que unifica toda a linguagem proprietária do FriendApp.

CAMADA EXTRA 04 — GLOSSÁRIO VIBRACIONAL ÚNICO DO FRIENDAPP

Entrada

A consistência de linguagem é tão vital quanto a consistência técnica. O **Glossário Vibracional Único** consolida todos os termos proprietários do FriendApp em um documento de referência, garantindo que desenvolvedores, designers, moderadores e usuários Premium falem a mesma língua.

Objetivos

1. **Unificar a linguagem vibracional** em todos os manuais.
 2. **Definir tecnicamente** cada termo proprietário.
 3. **Facilitar a comunicação** entre equipes e IA.
 4. **Evitar ambiguidades** e manter coerência no ecossistema.
-

Glossário Oficial — Termos e Definições

Termo	Definição Técnica	Aplicação no Sistema
Chat Vibracional	Sessão de troca de mensagens analisada por metadados vibracionais.	Base do sistema de comunicação.
Intenção Vibracional	Propósito energético de uma conversa, definido por tags.	Classificação inicial e dinâmica do chat.
Tags de Intenção	Metadados que categorizam conversas (Casual, Atração, Profunda, etc.).	Curadoria IA e Painel Vibracional.
Score Vibracional	Valor numérico (0–10) que representa a energia da conversa.	Cálculo de estados Pico, Transição e Colapso.
Estados Vibracionais	Classificações do chat em tempo real (🔥 Pico, 🌊 Transição, ❄️ Colapso).	Painel Vibracional e Logs.
Modo Pulsar	Sistema de mensagens efêmeras autodestrutivas.	Privacidade e sigilo.
Firewall Vibracional	Camada de segurança que bloqueia abusos ou linguagem tóxica em metadados.	Proteção em tempo real.
Check-in Energético	Feedback pós-chat que registra sentimento final do usuário.	Histórico e aprendizado da IA.
Painel Vibracional	Interface que mostra estados e score em tempo real ao usuário.	Experiência Premium e básica.
Logs Energéticos	Registro de metadados vibracionais (intenção, emojis, pausas, etc.).	Auditoria e observabilidade.
Advisor Aurah Kosmos	IA que sugere insights e recomendações baseadas em padrões vibracionais.	Matching, curadoria e pós-chat.
Colapso Vibracional	Estado de queda energética em uma conversa (score < 4.0).	Trigger de insights e sugestões.
Feedback Vibracional	Relatório final do usuário sobre sua experiência energética no chat.	Aprendizado coletivo e individual.
Heatmap Vibracional	Mapa que mostra intensidade de chats em tempo/espaço.	Relatórios e matching geográfico.
Encerramento Elegante	Fechamento de conversa feito pelo usuário com sugestão leve da IA.	Ética e autonomia preservadas.
Autonomia Energética	Garantia de que o usuário sempre decide o fluxo (IA nunca encerra sozinha).	Base ética do sistema.

Estrutura Técnica (Firestore)

collection: glossario_vibracional

→ termo: string

→ definicao: string

→ aplicacao: string

→ atualizado_em: timestamp

Governança e Ética

- Glossário é documento **oficial** referenciado em todos os manuais.
- Atualizações devem ser auditadas e aprovadas.
- É parte integrante de treinamentos internos e validações de API.

Fechamento Profissional

O **Glossário Vibracional Único** é o alicerce semântico do FriendApp. Ele garante que cada termo seja **claro, padronizado e aplicado com precisão técnica**, evitando ruídos e fortalecendo a identidade vibracional da plataforma.

➡ Na próxima e última camada extra (05), consolidaremos as **regras de Privacidade, Ética e Arquitetura Assíncrona**, que são a fundação ética do chat.

CAMADA EXTRA 05 — PRIVACIDADE, ÉTICA E ARQUITETURA ASSÍNCRONA

Entrada

O **Chat Vibracional** só pode existir com **confiança total**. Essa camada define as garantias de **privacidade do usuário**, a postura **ética da IA Aurah Kosmos** e a arquitetura **assíncrona** que assegura fluidez sem vigilância invasiva. É o alicerce que equilibra tecnologia, ética e experiência humana.

Objetivos

1. **Proteger a privacidade absoluta** das mensagens (E2EE).
2. **Garantir autonomia total do usuário**, sem intervenções autoritárias da IA.
3. **Explicar a arquitetura assíncrona**, que preserva fluidez sem aumentar latência.
4. **Estabelecer princípios éticos claros**, alinhados a LGPD/GDPR e transparência digital.

Privacidade

- **Criptografia ponta a ponta (E2EE)**: conteúdo de mensagens é inacessível a terceiros, inclusive à IA.
- **IA processa apenas metadados**: tempo de resposta, emojis, pausas, intensidade.
- **Logs anonimizados**: relatórios vibracionais coletivos nunca incluem dados pessoais.
- **Direito ao esquecimento**: usuário pode solicitar exclusão total de histórico.

Ética da IA Aurah Kosmos

Princípio	Aplicação no Sistema
Não intervenção direta	IA nunca encerra chats; apenas sugere pausas/insights.
Transparência	Usuário sempre sabe quando e por que um estado foi sugerido.
Consentimento	Qualquer uso de dados é autorizado pelo usuário.
Autonomia	Usuário decide quando iniciar, pausar ou encerrar chats.
Suporte, não controle	IA atua como guia pessoal, nunca como moderadora autoritária.

Arquitetura Assíncrona

- **Entrega de mensagens**: sempre imediata (<150ms via WebSocket).

- **Processamento vibracional:** feito em **fila assíncrona** (Kafka/PubSub), sem bloquear a entrega.
- **Atualização de estados:** painel vibracional recebe recálculo em até 3 segundos.
- **Resiliência:** se pipeline de IA falhar, mensagens continuam fluindo normalmente.

Fluxo Simplificado:

Envio de mensagem → Entrega instantânea (WebSocket) → Fila de eventos (Kafka)
→ Processamento assíncrono pela IA → Atualização do painel vibracional



Estrutura Técnica de Garantias

```
collection: privacy_audit
→ user_id: uuid
→ acao: "consentimento" | "exclusao" | "acesso_dados"
→ resultado: string
→ timestamp: datetime
```

- **Auditoria de consentimento:** cada aceite ou exclusão é registrado.
- **Monitoramento ético:** logs verificados regularmente para compliance.



Exemplos de Insights Éticos da IA

- Pós-colapso:

"A energia caiu na última troca. Deseja refletir sobre o motivo ou apenas seguir em frente?"
- Pós-série de chats intensos:

"Percebi que suas últimas conversas foram intensas. Deseja buscar algo mais leve?"

(Sempre apresentados como **convites** e nunca como comandos.)

Fechamento Profissional

A **Camada Extra 05** é a fundação ética do Chat Vibracional. Combinando **E2EE**, **análise apenas de metadados**, **arquitetura assíncrona e princípios de autonomia**, o sistema garante confiança plena. Aqui o FriendApp se diferencia: não apenas conecta, mas **respeita profundamente a liberdade vibracional de cada usuário**.