MANUAL TÉCNICO — SISTEMA DE CADASTRO

📘 CAMADA 01 — PROPÓSITO CENTRAL DO SISTEMA

🎯 Função Técnica

O **Sistema de Cadastro Consciente e Perfil Vibracional** é responsável por conduzir todo o processo de **entrada inicial** do usuário no ecossistema FriendApp.

Ele substitui o conceito tradicional de "cadastro de usuário" por uma **jornada estruturada em etapas técnicas**, garantindo:

- **Identidade validada (DCO/DUC)** com segurança jurídica e compliance LGPD.
- **Perfil vibracional técnico-matemático**, traduzindo conceitos abstratos (energia, aura, frequência) em **vetores numéricos e scores mensuráveis**.
- **Ativação dos subsistemas** essenciais (IA Aurah Kosmos, Feed, Conexões, Locais Parceiros, Modo Viagem, Segurança Vibracional).

🌍 Missão Técnica & Estratégica

O primeiro contato com o app não é apenas uma tela de login, mas um **pipeline de autenticação + onboarding sensorial**, que:

- Cria uma **base técnica de dados vibracionais** antes de liberar qualquer interação.
- Garante **proteções invisíveis para adolescentes (<18 anos)** e pessoas em vulnerabilidade.
- Evita vícios de redes sociais tradicionais, privilegiando **fluxos éticos, seguros e auditáveis**.

💡 Motivos Técnicos de Existência

1. **Confiabilidade e Compliance**: garantir que cada perfil tenha uma estrutura segura e auditável.
2. **Conversão de Conceitos Abstratos em Dados Técnicos**: ex.:
    - Frequência Vibracional → vetor multidimensional (512D).
    - Aura Inicial → classificação de cores HEX + score emocional.
    - Score Vibracional → fórmula ponderada com pesos e métricas objetivas.
3. **Base de Ativação de Todo o Ecossistema**: sem o cadastro consciente, nenhum módulo (Feed, Locais, Conexões, IA) pode operar corretamente.

🔐 Elementos Obrigatórios no Fluxo

- Portal Sensorial não pulável (mín. 20s, recomend. 60–90s).
- Coleta de dados básicos + intenção vibracional.
- Verificação documental (DUC/DCO).
- Teste de Personalidade Energética → geração de vetores de essência, intenção e proteção.
- Mapa de Frequência Inicial.
- Onboarding pela IA Aurah Kosmos.
- Ativação sincronizada dos subsistemas.

🧬 Princípios Técnicos Fundamentais

- **Registro com Validação Dupla**: idade + verificação vibracional/documental.
- **Proteção para Menores**: fluxos diferenciados, bloqueios invisíveis e filtros de segurança.
- **Adaptabilidade da IA**: onboarding e sugestões ajustados conforme vetor vibracional.
- **Arquitetura Assíncrona**: uso de filas (RabbitMQ/SQS) para processar IA sem travar UX.

- **Logs e Observabilidade**: todas as etapas registradas no ELK + Grafana com KPIs definidos.

📘 CAMADA 02 — ESTRUTURA DO FLUXO + ENTRADA SENSORIAL NÃO-PULÁVEL

🌀 Estrutura Geral da Jornada

O fluxo de entrada não começa com formulário, mas com um **portal sensorial técnico + pipeline de autenticação inicial**.

Esse design cria uma barreira de intenção, filtrando automaticamente usuários incoerentes (bots, cadastros falsos ou pessoas buscando apenas entretenimento/sexo).

🔗 Fluxo Lógico (Revisado)

```
graph TD
A[Primeira abertura do app] --> B[Portal Sensorial (vídeo 60–90s)]
B --> C[Formulário de Identidade + Intenção Vibracional]
C --> D{Idade < 18?}
D -->|Sim| E[Fluxo Protegido para Adolescentes + IA restrita]
D -->|Não| F[Verificação DUC/DCO + Ética Vibracional]
E --> G[Teste de Personalidade Energética]
F --> G
G --> H[Geração do Mapa de Frequência + Vetores]
H --> I[Onboarding Aurah Kosmos]
I --> J[Ativação de Subsistemas (Feed, Conexões, Locais, IA)]
```

🎬 Entrada Sensorial — Texto + Vídeo Não Pulável

🎯 Objetivo Técnico

- Interromper automatismos cognitivos.
- Criar um “checkpoint vibracional” antes de liberar o fluxo de cadastro.

- Capturar dados de **comportamento inicial** para análise de risco e coerência.

🔒 Regras Técnicas de Exibição

Componente	Regra Técnica
Tipo de tela	Fullscreen, sem botões visíveis até o término
Duração mínima	60s (recomendado 90s), adaptativo por device
Botão "Continuar"	Liberado apenas após tempo mínimo ou ≥80% do vídeo
Acessibilidade	Legendas automáticas + narração em áudio
Repetição	Apenas na primeira abertura real (`onFirstLaunch == true`)
Gatilho de transição	`onVideoEnd → iniciarCadastro()`
Coleta de dados	Tempo de permanência, tentativas de pular, abandono precoce, microexpressões (se câmera ativa com consentimento)

🛡️ Segurança e Anti-abandono

- Usuários que abandonam antes de 20s → **marcados com flag de risco** (`risk_score += 15`).
- Tentativas repetidas de pular → registradas em `skip_attempts_log`.
- Voz opcional → se ativada, análise de sentimento em tempo real.

🧠 Pipeline Técnico (Pseudocódigo)

```
def portal_sensorial(user_id):
    start_timer(0)
    video.play("intro_portal.mp4", min_duration=60)
    log_event(user_id, "portal_start")

    while video.is_playing:
        capture_interactions(user_id)

    if video.completed:
        log_event(user_id, "portal_complete")
```

```
        allow_continue(user_id)
    else:
        mark_incoherence(user_id, reason="abandon_portal")
```

📊 Dados Coletados para Score Vibracional Inicial

- `tempo_visualizacao` → tempo real assistido.
- `skip_tentativas` → nº de interações forçadas.
- `frequencia_portal` → coerência do comportamento (ex.: abandono precoce = baixa coerência).

Exemplo de JSON de Log

```
{
  "user_id": "temp_9237",
  "portal_status": "completo",
  "tempo_visualizacao": 92,
  "skip_tentativas": 0,
  "frequencia_portal": "alta_coerencia",
  "timestamp": "2025-09-08T18:12:53Z"
}
```

🧬 Justificativa Estratégica (Traduzida para Devs)

- Funciona como **filtro inicial de risco** (semelhante a CAPTCHA invisível).
- Reduz cadastros incoerentes e bots logo na primeira camada.
- Garante que a entrada no FriendApp tenha **intenção consciente + dados técnicos coletados** para IA Aurah Kosmos iniciar o processamento.

📘 CAMADA 03 — FORMULÁRIO DE IDENTIDADE BASE + INTENÇÃO

VIBRACIONAL

🧩 Objetivo Técnico

Coletar os **dados básicos + intenção inicial do usuário**, de forma neutra (sem viés de namoro), e gerar um **pré-vetor vibracional inicial** que servirá de insumo para o Teste Energético e o Mapa de Frequência.

🧠 Estrutura do Formulário (Revisada)

Campo	Tipo de Entrada	Regras Técnicas
`nome`	Texto aberto	Pode ser real, artístico ou espiritual. Normalizado em UTF-8.
`data_nascimento`	DatePicker	Validada via `age >= 13`. Para menores, aciona fluxo protegido.
`genero` (opcional)	Lista aberta	Opções: ["Masculino", "Feminino", "Outro", "Prefiro não informar"]. IA adapta tom de comunicação.
`cidade_atual`	GPS ou manual	GPS via `navigator.geolocation` com consentimento. Se falhar, input manual obrigatório.
`intencao_vibracional`	Botões sensoriais	Ex.: "Amizades reais", "Conversas profundas", "Afeto leve", "Explorar experiências reais".
`palavra_energia`	Texto livre	Input opcional. Analisado via NLP para sentimento.

🔒 Regras de Validação

- **Idade < 18 anos** → bloqueio de funcionalidades e ativação de proteção invisível.
- **Campos incoerentes** (ex.: linguagem sexual no campo de intenção) → flag `risk_score += 25`.
- **Cidade não localizada** → fallback manual sem bloquear fluxo.

🧠 Pipeline Técnico de Processamento

```
def process_form(user_id, form_data):
    # Validação
    if form_data['idade'] < 13:
        block_registration(user_id, reason="underage")

    # Pré-análise vibracional
    sentimento = nlp.sentiment(form_data['palavra_energia'])
    intencao = classify_intention(form_data['intencao_vibracional'])

    # Criação de vetor inicial
    vetor_inicial = aurah.embedding([
        form_data['nome'],
        form_data['intencao_vibracional'],
        form_data['palavra_energia']
    ])

    save_profile(user_id, form_data, vetor_inicial)
    log_event(user_id, "form_completed")
```

📊 Pré-Vetor Vibracional Inicial

- **Componentes**:
  - Análise de sentimento (`sent_score` de -1 a 1).
  - Intenção escolhida (categorias pré-mapeadas).
  - Palavra-chave → embedding NLP.
- **Formato JSON**:

```
{
  "user_id": "u123",
  "idade": 27,
  "intencao": "Conversas profundas",
  "sentimento": 0.72,
  "vetor_inicial": [0.12, 0.34, 0.56, ...],
  "timestamp": "2025-09-08T18:45:00Z"
```

```
}
```

🔗 Integrações Ativas

- `perfil-e-frequencia-service` → grava dados base + vetor inicial.
- `aurah-kosmos-core` → usa intenção + palavra para preparar onboarding.
- `feed-sensorial-core` → recebe primeira curadoria leve.
- `conexoes-reais-service` → ativa tags energéticas iniciais.

🧬 Justificativa Técnica

- Transforma dados subjetivos em **objetos quantificáveis** (vetores, scores).
- Evita viés de namoro → reforça que o FriendApp é **sobre conexões autênticas**.
- Garante que já no primeiro formulário exista **validação técnica + pré-análise vibracional** para suportar segurança e personalização.

📘 CAMADA 04 — VERIFICAÇÃO INTELIGENTE (DUC/DCO) + ALERTAS ÉTICOS DE SEGURANÇA

🛡️ Propósito Técnico

Garantir que cada usuário entre no FriendApp de forma **segura, confiável e em conformidade legal (LGPD)**, usando verificação documental (DUC) ou vibracional/simbólica (DCO).

A verificação define **níveis de confiança do perfil** e ativa **restrições técnicas invisíveis** quando o usuário opta por não se verificar.

🔍 Tipos de Verificação

Tipo	Nome Completo	Aplicação	Obrigatoriedade
DUC	Documento Único de Cidadania	RG, CNH ou documento oficial + selfie	Obrigatório para menores de 18 e criadores de grupo
DCO	Documento de Conexão Original	Frase + arquétipo + microteste vibracional	Opcional para maiores de 18

🔁 Fluxo Técnico de Verificação (Revisado)

```
flowchart TD
A[Formulário Base Preenchido] --> B{Idade < 18?}
B -->|Sim| C[DUC Obrigatório]
B -->|Não| D[Exibir Benefícios da Verificação]
D --> E{Usuário aceita verificar?}
E -->|Sim| F[Iniciar DCO ou DUC com OCR + FaceMatch]
E -->|Não| G[Seguir com restrições + alertas sutis]
```

🔒 Regras de Segurança e Armazenamento

- **Retenção de documentos**: máx. 24h, criptografia AES-256 em bucket privado (S3/GCP).
- **Exclusão segura**: método *Secure Erase* após processamento.
- **Resultado armazenado**: apenas status booleano (`verified = true/false`) e dados extraídos (nome, idade).
- **Recomenda-se API terceirizada**: Unico, idwall ou similar, para reduzir risco de falhas internas.

🧠 Pipeline Técnico DUC (Exemplo)

```
def verificar_duc(user_id, doc, selfie):
    ocr_data = run_ocr(doc)
    face_score = compare_faces(selfie, ocr_data["photo"])
```

```
if face_score > 0.85 and validate_doc_integrity(ocr_data):
    approve_verification(user_id, type="DUC")
    enable_verified_features(user_id)
else:
    log_event(user_id, "duc_failed")
    if attempts(user_id) >= 3:
        trigger_manual_review(user_id)
```

🚨 Alertas Éticos de Segurança

- **Usuário verificado x não verificado**: mensagem no topo → *"Esse perfil ainda não possui verificação. Fique atenta a sinais incoerentes."*
- **Dois não verificados**: chat segue, mas sem suporte da IA nem acesso a áreas críticas.
- **Perfil denunciado**: IA ativa monitoramento intensivo invisível.

📊 Logs e Regras Técnicas

- Todas as escolhas (verificar ou não) são registradas em `logs-service` com hash + timestamp.
- Cross-verificação em conexões é obrigatória:

```
if connection_request(from_user, to_user):
    if not to_user.is_verified():
        inject_warning(from_user)
```

🔗 Integrações Ativas

- `verificacao-service` → processamento OCR/FaceMatch, controle de tempo de retenção.
- `aurah-kosmos-core` → adapta tom de onboarding conforme status de verificação.
- `logs-service` → armazena eventos de verificação e falhas.

- `chat-core` → injeta alertas leves em interações com risco.

🧬 Justificativa Técnica

- Eleva a segurança e confiança do ecossistema sem criar barreiras excessivas.
- Garante conformidade com LGPD (retenção mínima + exclusão segura).
- Substitui termos vagos (“alerta vibracional”) por **pipelines auditáveis e logs hashados**.
- Flexibiliza experiência → usuários podem entrar sem verificação, mas **com restrições técnicas claras**

📘 CAMADA 05 — TESTE DE PERSONALIDADE ENERGÉTICA (ENTRADA OFICIAL NA FREQUÊNCIA PESSOAL)

🌌 Propósito Técnico

Transformar a etapa de “teste vibracional” em uma **análise estruturada e quantificável**, que gera os primeiros **vetores multidimensionais** usados pela IA Aurah Kosmos.

Esse teste marca a passagem de um usuário comum para um **perfil vivo, dinâmico e matematicamente representado** no FriendApp.

🧬 Estrutura Técnica do Teste

Etapa	Componente	Regra Técnica
Início	Geração de `session_token`	Endpoint: `/api/personality/start`
Estímulos sensoriais	Imagens, sons, frases	Exibidos em ordem randômica, coletando tempo de resposta

Etapa	Componente	Regra Técnica
Escolhas guiadas	Botões, imagens, múltipla escolha	Salvos em banco como `choice_id + timestamp`
Tempo de resposta	Milissegundos	Usado como métrica de impulsividade/reflexão
Linguagem	NLP em frases livres	Extração de sentimentos + embeddings
Interação por áudio (opcional)	Speech-to-text	Processado com modelo ASR + análise de entonação
Processamento final	IA Aurah	Gera vetores e score inicial
Persistência	Banco híbrido SQL + NoSQL	PostgreSQL (logs), Firestore (vetores dinâmicos)

🧠 Pipeline Técnico (Pseudocódigo)

```
def executar_teste(user_id):
    session = start_session(user_id)

    for stimulus in stimuli_list:
        show(stimulus)
        choice, time = record_response(user_id, stimulus)
        save_response(user_id, stimulus, choice, time)

    sentimento = nlp.sentiment_analysis(user_id)
    embeddings = aurah.generate_embeddings(user_id)

    score = calcular_score(sentimento, embeddings)
    save_profile(user_id, score)

    return score
```

📊 Outputs Gerados

Elemento	Resultado Técnico
Personalidade Energética	Categoria inicial (Guardião, Visionário, Pulsante, etc.)

Elemento	Resultado Técnico
Frequência Dominante	Vetor principal (`freq_vector[512D]`)
Paleta de Cores	HEX codes derivados do score emocional
Ponto de Luz	Símbolo matemático (hash gráfico)
Score Vibracional Inicial	Fórmula ponderada:

Score=(Sentimento∗W1)+(TempoResposta∗W2)+(Consiste^nciaEscolhas∗W3)−(Ruı´do/Tentativas∗W4)Score = (Sentimento * W1) + (TempoResposta * W2) + (ConsistênciaEscolhas * W3) - (Ruído/Tentativas * W4)

Score=(Sentimento∗W1)+(TempoResposta∗W2)+(Consiste^nciaEscolhas∗W3)−(Ruı´do/Tentativas∗W4)

🔗 Integrações Ativas

- `aurah-kosmos-core` → processa sentimentos, tempo de resposta e gera embeddings.
- `perfil-e-frequencia-service` → grava score + vetores.
- `feed-sensorial-core` → usa frequência dominante para curadoria inicial.
- `conexoes-reais-service` → define arquétipos de matching inicial.

🛡️ Casos de Borda

- **Teste abandonado** → marcar `status = incomplete`, oferecer retry em até 24h.
- **Respostas incoerentes (aleatórias em <200ms)** → flag `risk_score += 30`.
- **Usuário discorda do resultado** → botão *“Ajustar Perfil”* → IA coleta feedback e recalibra pesos.

🧬 Justificativa Técnica

- Converte conceitos abstratos (“personalidade energética”) em **outputs mensuráveis** (vetores, scores, JSON).
- Garante escalabilidade via **processamento assíncrono** → workers de IA consomem fila (`user_personality_pending`).
- Oferece mecanismos de **feedback do usuário**, evitando frustração caso o resultado não corresponda.

📘 CAMADA 06 — GERAÇÃO DO MAPA DE FREQUÊNCIA E AURA INICIAL

🌌 Propósito Técnico

Converter os resultados do Teste de Personalidade Energética em um **Mapa de Frequência inicial**, que representa o estado vibracional do usuário em **dados matemáticos + visualização simbólica**.

Esse mapa serve como **chave primária de personalização** para Feed, Conexões, IA Aurah Kosmos e demais sistemas.

🧬 Componentes do Mapa Vibracional

Elemento	Definição Técnica
`frequencia_dominante`	Vetor numérico 512D → reduzido por PCA em 3 eixos principais
`paleta_vibracional`	Cores HEX atribuídas conforme score emocional (ex.: alegria >0.7 = #F0E68C)
`ponto_de_luz`	Hash gráfico baseado em seed aleatória + score (ex.: fractal animado)
`aura_visual`	Renderização em shader → animação pulsante adaptada ao score
`centro_de_intencao`	Arquétipo inicial (Guardião, Visionário, etc.), baseado no maior peso vetorial
`zona_de_frequencia`	Posição inicial no grafo global (Node2Vec / GraphQL DB)

🧠 Pipeline Técnico (Pseudocódigo)

```
def gerar_mapa_frequencia(user_id, dados_teste):
    vetor = calcular_vetor(dados_teste)
    sentimento = nlp.sentiment(dados_teste['frases'])

    # Redução dimensional para visualização
    coords = pca_reduce(vetor, dims=3)
```

```
    # Definição de cores
    cor_base = definir_cor(sentimento, dados_teste['arquétipo'])

    mapa = {
        "user_id": user_id,
        "frequencia_dominante": vetor,
        "coords": coords,
        "paleta": cor_base,
        "ponto_de_luz": hash_symbol(user_id, vetor),
        "centro_intencao": escolher_arquetipo(vetor),
        "timestamp": datetime.utcnow()
    }

    salvar_mapa(user_id, mapa)
    return mapa
```

📊 Exemplo de Estrutura JSON

```
{
  "user_id": "u456",
  "frequencia_dominante": [0.12, 0.34, 0.56, ...],
  "coords": [0.45, -0.22, 0.78],
  "paleta": ["#F0E68C", "#4682B4"],
  "ponto_de_luz": "hash_9f8a7c",
  "centro_intencao": "Visionário",
  "zona_frequencia": "Cluster_Solar_03",
  "timestamp": "2025-09-08T19:30:00Z"
}
```

🔒 Regras Técnicas

- **Geração única por ciclo vibracional inicial**.
- **Recalibração** só por evento → login após >24h, 10+ novas interações ou variação emocional detectada.

- **Mapa não é público**: usado apenas para personalização interna.
- **Proteção LGPD**: dados armazenados com versionamento (append-only) + hash criptográfico.

🔗 Integrações Ativas

- `aurah-kosmos-core` → traduz vetor em linguagem natural e tom de IA.
- `feed-sensorial-core` → recebe dados para curadoria de conteúdo inicial.
- `conexoes-reais-service` → usa posição no grafo para sugerir conexões.
- `chat-core` → ajusta estilo conversacional conforme centro de intenção.
- `painel-vibracional-premium` → exibe visualização da aura (somente Premium).

🛡️ Casos de Borda

- **Mapa não gerado (falha IA)** → fallback para arquétipo + frase inicial apenas.
- **Cores neutras (<|sentimento| 0.2)** → atribuição padrão `#D3D3D3` (cinza neutro).
- **Vetor incoerente (ruído)** → IA recalcula no próximo login.

🧬 Justificativa Técnica

- Transforma dados subjetivos em **representação visual e vetorial objetiva**.
- Evita processamento síncrono → geração do mapa roda em **fila assíncrona** (SQS/RabbitMQ).
- Fornece base sólida para todos os módulos → Feed, Conexões, IA, Locais, Segurança.

📘 CAMADA 07 — ONBOARDING SENSORIAL COM AURAH KOSMOS

🌌 Propósito Técnico

O onboarding é a **primeira interação profunda do usuário com a IA Aurah Kosmos** após a criação do Mapa de Frequência.

Ele garante que o usuário sinta acolhimento imediato, enquanto a IA traduz dados técnicos (vetores, score vibracional) em uma **experiência sensorial e personalizada**.

✨ Estrutura Técnica da Experiência

Etapa	Componente	Execução Técnica
Áudio de boas-vindas	Narrativa suave (opcional)	Texto → TTS neural (`gTTS` , Amazon Polly, ElevenLabs) adaptado à frequência
Texto personalizado	Frase dinâmica	`aurah.generate_onboarding_message(user_id)`
Animação visual	Aura pulsante	Renderização em shader WebGL/Unity com base no vetor vibracional
Sugestão de primeiro passo	Ação inicial	Ex.: visitar grupo, explorar feed → endpoint `/api/suggestion/firststep`

🧠 Pipeline Técnico (Pseudocódigo)

```
def onboarding_aurah(user_id):
    mapa = fetch_frequency_map(user_id)
    personalidade = fetch_personality_type(user_id)

    mensagem = aurah.generate_onboarding_message(mapa, personalidad
e)

    if user_opted_audio(user_id):
        audio = tts_engine.synthesize(mensagem.text, voice=mapa["frequenci
a"])
        play_audio(audio)

    render_aura(mapa["frequencia_dominante"], mapa["paleta"])
    display_text(mensagem.text)
    suggest_next_step(user_id)
```

```
    log_event(user_id, "onboarding_completed")
```

📊 Exemplo de Saída JSON

```
{
  "user_id": "u789",
  "mensagem_texto": "Você vibra com coragem silenciosa. Aqui, o mundo v
ai te escutar.",
  "audio_gerado": true,
  "animacao_aura": "shader_fractal_v2",
  "primeira_sugestao": "Visitar grupo Guardiões da Essência",
  "timestamp": "2025-09-08T20:00:00Z"
}
```

🛡️ Regras de Adaptação

Situação	Ação Técnica
Usuário <18	IA simplifica linguagem, reforça proteção vibracional
Frequência muito densa	IA foca em acolhimento + sugestões leves (grupos de suporte)
Perfil com sinais de reclusão social	IA prioriza conexões seguras e espaços privados
Feedback negativo (“não corresponde”)	IA reprocessa mensagem → loga em `aurah_feedback_service`

🔗 Integrações Ativas

- `aurah-kosmos-core` → gera mensagem, voz e animação de aura.
- `perfil-e-frequencia-service` → fornece dados da frequência e personalidade.
- `feed-sensorial-core` → adapta o conteúdo inicial mostrado.
- `conexoes-reais-service` → sugere primeiras conexões vibracionais.

🧬 Justificativa Técnica

- Garante que o usuário **não entre em um app vazio** → já inicia com mensagem e ação concreta.
- Tradução de dados abstratos (vetores, scores) em **UX palpável** (áudio, texto, animação).
- Logs permitem auditoria da primeira impressão → se o onboarding falha, a IA aprende e recalibra.

📘 CAMADA 08 — ATIVAÇÃO DOS SISTEMAS INTERNOS (IA, Feed, Conexões, Locais, Modo Viagem)

1. 🎯 Propósito Técnico

Após o onboarding sensorial, esta camada **sincroniza e ativa todos os módulos principais do FriendApp**, conectando-os ao **Mapa de Frequência inicial** e ao **perfil vibracional calculado**.

Sem essa ativação, o usuário teria apenas um cadastro estático; com ela, o app se torna **dinâmico, vivo e personalizado** já no primeiro login.

2. 🔧 Componentes Técnicos

Módulo	Ação Inicial	Dependências
`feed-sensorial-core`	Ativa feed inicial filtrado por frequência/intenção	`perfil-e-frequencia-service`
`conexoes-reais-service`	Sugere conexões vibracionais compatíveis	`aurah-kosmos-core`, `vetores`
`locations-service`	Mostra locais parceiros compatíveis com a energia	`mapa-frequencia-core`
`travel-service`	Detecta viagem + ativa resenhas e encontros	GPS + IA Aurah
`aurah-kosmos-core`	Assume a jornada como guia do usuário	Vetores de essência/intenção

Módulo	Ação Inicial	Dependências
`painel-vibracional-premium`	Exibe insights vibracionais	Apenas Premium

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def ativar_sistemas(user_id):
    freq_map = fetch_frequency_map(user_id)
    personalidade = fetch_personality(user_id)
    vetores = fetch_vetores(user_id)

    # Ativação dos módulos
    activate_feed(user_id, freq_map)
    activate_connections(user_id, personalidade, vetores)
    activate_locations(user_id, freq_map)
    activate_travel_mode(user_id, vetores)
    iniciar_aurah(user_id, freq_map)

    if is_premium(user_id):
        activate_painel_vibracional(user_id)

    log_event(user_id, "sistemas_ativados")
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u101",
  "feed_status": "activated",
  "connections_status": "activated",
  "locations_status": "activated",
  "travel_mode": "inactive",
  "aurah_status": "listening",
  "painel_vibracional": false,
  "timestamp": "2025-09-08T20:45:00Z"
```

```
}
```

5. 🔒 Segurança e LGPD

- Usuários `<18` → ativação parcial (feed filtrado, conexões limitadas, locais restritos).
- Usuários sem DUC/DCO → ativação com restrições (chat limitado, sem criação de grupos).
- Logs de ativação são criptografados com hash SHA-256 + timestamp.
- Dados sensíveis armazenados apenas em `perfil-e-frequencia-service` (não expostos em payloads de API).

6. ⚠️ Casos de Borda

Situação	Fluxo de Fallback
IA Aurah indisponível	Feed e conexões ativados apenas por `vetores` → fallback rule
GPS não acessível	Travel Mode entra em "standby" até nova permissão
Falha no feed inicial	Exibe feed neutro (conteúdo universal, seguro)
Usuário sem verificação	Ativação com alertas sutis + bloqueio de áreas críticas
Latência > 3s em ativação	Mensagem *"Estamos ajustando sua frequência..."* e loading otimizado

7. 🔗 Integrações Ativas

- **Entrada**:
  - `perfil-e-frequencia-service` → fornece aura, vetores, arquétipo.
  - `aurah-kosmos-core` → interpreta dados e personaliza onboarding.
- **Saída**:
  - `feed-sensorial-core` → ativa curadoria inicial.
  - `conexoes-reais-service` → ativa sugestões compatíveis.
  - `locations-service` → entrega locais parceiros.

- `travel-service` → libera Modo Viagem.
- `painel-vibracional-premium` → exibe insights para Premium.

---

8. 📈 KPIs e Observabilidade

- **Dashboard de Ativação**
  - `taxa_ativacao_completa` (% de cadastros que chegam a essa camada)
  - `latencia_p95_ativacao` (meta: < 3s total)
  - `taxa_falha_feed` (% de erros na ativação do feed)
  - `ativacao_parcial_menores` (% de menores de idade com restrições aplicadas)
  - `ativacao_sem_verificacao` (% de cadastros não verificados)
- **Logs Técnicos (Exemplo)**

```
{
  "event": "ativacao_sistema",
  "user_id": "u101",
  "feed": "ok",
  "connections": "ok",
  "aurah": "fallback_mode",
  "latencia_total": 2.8,
  "timestamp": "2025-09-08T20:46:00Z"
}
```

🧬 Justificativa Técnica

- Evita que o usuário entre em um app “vazio” → garante **personalização instantânea**.
- Processos críticos (IA, Feed, Conexões) usam **fila assíncrona** → não travam UX.
- KPIs permitem medir gargalos e corrigir rápido.
- Estrutura preparada para **premium vs free**, **menores vs adultos**, e fluxos de erro auditáveis.

📘 CAMADA 09 — SISTEMA DE STATUS VIBRACIONAL DO PERFIL (COMPORTAMENTO DINÂMICO + ATUALIZAÇÕES DE AURA)

1. 🎯 Propósito Técnico

Transformar o perfil do usuário em um **estado dinâmico e vivo**, que se atualiza continuamente com base em:

- uso real do app,
- interações emocionais,
- variações linguísticas e temporais.

Esse sistema é responsável por manter o **Mapa de Frequência sincronizado com a vida real do usuário**, evitando perfis estáticos.

2. 🔧 Componentes Técnicos

Elemento	Função Técnica
`aura_mutavel`	Campo visual/energético que muda com variações de sentimentos (cores HEX, intensidade).
`atualizacoes_frequencia`	Recalibração de vetores a partir de triggers de interação.
`radar_emocional`	NLP contínuo em mensagens + tempo de uso.
`expressao_atual`	Frase de status gerada pela IA (ex.: “Vibrando leveza e presença”).
`registro_de_jornada`	Log interno (não visível ao público) com histórico vibracional.

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def atualizar_status_vibracional(user_id):
    padrao_emocional = aurah.read_emotional_pattern(user_id)
    novo_mapa = aurah.update_frequency_map(user_id, padrao_emocional)
```

```
    save_new_state(user_id, novo_mapa)
    atualizar_feed(user_id, novo_mapa)
    atualizar_status_frase(user_id, novo_mapa)

    log_event(user_id, "status_atualizado", novo_mapa)
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u202",
  "aura": {
    "cor_principal": "#6A5ACD",
    "intensidade": 0.81,
    "ponto_de_luz": "espiral"
  },
  "status_frase": "Em reconexão com o que realmente importa.",
  "radar_emocional": {
    "sentimento": 0.65,
    "palavras_chave": ["esperança", "mudança"]
  },
  "timestamp": "2025-09-08T21:15:00Z"
}
```

5. 🔒 Segurança e LGPD

- Todos os logs de estado são **append-only** → impossibilidade de alteração retroativa.
- Dados sensíveis (palavras-chave, sentimentos) armazenados anonimizados em `vibrational_log`.
- Usuários podem solicitar **reset de status vibracional** (LGPD: direito à eliminação).

6. ⚠️ Casos de Borda

Situação Detectada	Ação do Sistema
Vibração muito densa (tristeza crônica)	IA sugere apoio emocional e oculta conteúdos pesados.
Energia dispersa (uso inconsistente)	IA propõe âncoras → textos curtos, conexões estáveis.
Aura fechada (baixa interação)	Sistema sugere resenhas seguras e grupos pequenos.
Evolução positiva	Feed e conexões são aprofundados automaticamente.
Falha de processamento IA	Último status válido é mantido até nova recalibração.

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → leitura de padrão emocional + atualização de vetores.
  - `chat-core` → coleta vocabulário e sentimento de mensagens.
- **Saída**
  - `feed-sensorial-core` → ajusta conteúdo em tempo real.
  - `conexoes-reais-service` → altera matching de acordo com abertura/retração de campo.
  - `painel-vibracional-premium` → mostra insights profundos para usuários Premium.

8. 📈 KPIs e Observabilidade

- **Dashboard de Status Vibracional**
  - `taxa_recalibracao_diaria` (nº de updates por usuário/dia).
  - `latencia_media_update` (meta: < 2s).
  - `frases_mais_frequentes` (ranking global).
  - `score_medio_emocional` (média ponderada dos usuários).

- `usuarios_em_estado_critico` (% de perfis com tristeza crônica detectada).

- **Log Exemplo**

```
{
  "event": "status_update",
  "user_id": "u202",
  "sentimento": 0.65,
  "expressao": "Em reconexão com o que realmente importa.",
  "vetor_hash": "f83ac91",
  "timestamp": "2025-09-08T21:16:00Z"
}
```

🧬 Justificativa Técnica

- Mantém o perfil **vivo e coerente com a realidade emocional** do usuário.
- Reduz risco de descompasso → feed e conexões sempre refletem estado atual.
- Usa IA e NLP de forma **assíncrona** para não travar UX.
- Garante segurança e auditabilidade com **logs imutáveis + métricas claras**.

📘 CAMADA 10 — REGRAS DE VERIFICAÇÃO, ALERTAS DE SEGURANÇA E REPUTAÇÃO ENERGÉTICA

---

1. 🎯 Propósito Técnico

Estabelecer o **sistema de credibilidade e segurança** do FriendApp, unindo:

- **Verificação documental (DUC)**
- **Verificação vibracional (DCO)**
- **Sistema de alertas inteligentes**

- **Score de Reputação Energética (quantitativo e auditável)**

Esse conjunto protege contra **perfis falsos, abusivos ou incoerentes**, garantindo confiança e fluidez no ecossistema.

---

2. 🔧 Componentes Técnicos

Componente	Função Técnica
`DUC`	Documento Único de Cidadania (OCR + FaceMatch + prova de vida).
`DCO`	Documento de Conexão Original (frase + arquétipo + microteste vibracional).
`verificado_plus`	Selo duplo (DUC + DCO) → acesso irrestrito ao ecossistema.
`alertas_inteligentes`	Banners contextuais e discretos para risco percebido.
`score_vibracional`	Reputação invisível do usuário, calculada continuamente.

---

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def processar_verificacao(user_id, metodo):
    if metodo == "DUC":
        return verificar_duc(user_id)
    elif metodo == "DCO":
        return verificar_dco(user_id)
    else:
        raise ValueError("Método inválido")

def calcular_score_vibracional(user_id):
    denuncias = get_denuncias(user_id)
    interacoes_toxicas = get_toxicidade(user_id)
    reciprocidade = get_reciprocidade(user_id)
    verificacoes = get_status_verificacoes(user_id)

    score = (5 * verificacoes) - (2 * denuncias) - (1.5 * interacoes_toxicas) +
(3 * reciprocidade)
    return max(0, min(100, score))  # Score de 0 a 100
```

---

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u303",
  "duc": true,
  "dco": false,
  "verificado_plus": false,
  "score_vibracional": 78,
  "alerta_ativo": "perfil_nao_verificado",
  "timestamp": "2025-09-08T21:45:00Z"
}
```

5. 🔒 Segurança e LGPD

- Documentos armazenados por máx. 24h (bucket S3 criptografado AES-256).
- Após verificação → exclusão com *Secure Erase*.
- Apenas status (`true/false`) e dados extraídos (nome, idade) persistem em banco.
- Logs de verificação → imutáveis, hashados em `logs-service`.

6. ⚠️ Casos de Borda

Situação	Ação Técnica
Conexão entre verificado e não verificado	Banner discreto: *"Esse perfil ainda não concluiu verificação. Conecte-se com consciência."*
Tentativa de criar grupo sem DUC	Fluxo bloqueado → mensagem: *"Necessário DUC válido para criar grupos."*
Usuário com muitas denúncias	Invisibilidade automática + análise manual.
Score < 30	Ativa firewall vibracional → bloqueio de interações sensíveis.
Falha no OCR/FaceMatch (3x)	Escalada para revisão manual em até 12h.

7. 🔗 Integrações Ativas

- **Entrada**
  - `verificacao-service` → processa DUC/DCO.
  - `logs-service` → armazena tentativas e falhas.
- **Saída**
  - `aurah-kosmos-core` → ajusta tom e interações conforme score.
  - `feed-sensorial-core` → oculta conteúdos de perfis com baixa reputação.
  - `chat-core` → limita conversas em risco.
  - `painel-reputacao` (interno/moderadores) → exibe histórico completo.

---

8. 📈 KPIs e Observabilidade

- `taxa_verificacao_duc` (% de usuários que completam DUC)
- `taxa_verificacao_dco` (% de usuários que completam DCO)
- `tempo_medio_verificacao` (meta: < 2 min OCR/FaceMatch)
- `usuarios_baixa_reputacao` (% global < 30 pontos)
- `taxa_alertas_exibidos` (% de conexões que geraram alerta)

Log Exemplo:

```
{
  "event": "score_reputacao_update",
  "user_id": "u303",
  "score": 78,
  "denuncias": 0,
  "tox_interacoes": 1,
  "reciprocidade": 15,
  "timestamp": "2025-09-08T21:46:00Z"
}
```

🧬 Justificativa Técnica

- Torna o “vibracional” mensurável via **fórmula de score objetiva**.
- Evita falsos positivos → alertas sempre discretos e contextuais.
- Permite auditoria (logs hashados + dashboard reputação).
- Garante que a reputação não é “punitiva”, mas usada para **proteger e calibrar interações**.

📘 CAMADA 11 — EXPERIÊNCIA DE USUÁRIOS NÃO VERIFICADOS (FREE VS VERIFICADOS)

1. 🎯 Propósito Técnico

Definir **regras claras de permissão e restrição** para usuários com diferentes níveis de verificação:

- **Não verificados (sem DUC/DCO)**
- **Parcialmente verificados (apenas DUC ou apenas DCO)**
- **Totalmente verificados (DUC + DCO → selo Verificado+)**

O objetivo é proteger a comunidade sem punir o usuário, incentivando a verificação através de **limitações graduais e alertas sutis**.

2. 🔧 Componentes Técnicos

Área	Free (Não verificado)	Parcial (DUC ou DCO)	Verificado+ (DUC + DCO)
Feed Sensorial	Permitido (curadoria leve)	Permitido (curadoria média)	Completo
Conexões Autênticas	Só pode convidar verificados	Convites limitados	Livre
Chat	Restrito (não pode iniciar com verificados)	Pode conversar com conhecidos	Livre

Área	Free (Não verificado)	Parcial (DUC ou DCO)	Verificado+ (DUC + DCO)
Postagens (Momentos 24h)	Permitido (passa por IA moderadora)	Permitido	Livre
Modo Viagem / Bora	Apenas leitura	Pode entrar em grupos com convite	Livre
Criação de Grupos	Bloqueado	Limitado	Livre
Locais Parceiros	Apenas leitura básica	Leitura + promoções básicas	Destaque e integrações
IA Aurah Kosmos	Versão básica (respostas genéricas)	Semi-personalizada	Profunda, evolutiva
Painel Vibracional Premium	Bloqueado	Parcial	Livre

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def aplicar_regras_experiencia(user_id):
    status = get_verification_status(user_id)

    if status == "nao_verificado":
        limitar_chat(user_id, modo="restrito")
        limitar_feed(user_id, curadoria="leve")
        bloquear_grupos(user_id)

    elif status == "parcial":
        limitar_chat(user_id, modo="parcial")
        liberar_feed(user_id, curadoria="media")
        permitir_grupos_com_convite(user_id)

    elif status == "verificado_plus":
        liberar_todas_funcoes(user_id)

    log_event(user_id, "regras_experiencia_aplicadas", {"status": status})
```

4. 📊 Representação de Dados (JSON Exemplo)

```
{
  "user_id": "u404",
  "status_verificacao": "parcial",
  "feed": "curadoria_media",
  "chat": "restrito_a_conhecidos",
  "grupos": "convite_necessario",
  "painel_vibracional": false,
  "timestamp": "2025-09-08T22:10:00Z"
}
```

5. 🔒 Segurança e LGPD

- Status de verificação armazenado em `users.verificacao_status`.
- Logs de tentativas bloqueadas → enviados para `logs-service` com hash.
- Nenhuma informação pessoal (RG, selfie) é usada nesta camada → apenas o **status booleano** (verificado/não verificado).

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Usuário tenta iniciar chat com verificado sem DUC	Bloqueio → mensagem: *"Essa função é exclusiva para perfis verificados."*
Usuário cria grupo sem DUC	Fluxo negado + CTA: *"Verifique seu perfil para criar grupos."*
Usuário denuncia bloqueio injusto	Log manual enviado para análise → `painel-admin`
Usuário Premium sem verificação	Premium ativo, mas bloqueado em áreas críticas → incentivo forte à verificação.

7. 🔗 Integrações Ativas

- **Entrada**
  - `verificacao-service` → status DUC/DCO.
- **Saída**

- `chat-core` → restringe interações.
- `feed-sensorial-core` → aplica curadoria leve/média.
- `conexoes-reais-service` → define permissões de convites.
- `aurah-kosmos-core` → adapta nível de profundidade da IA.

---

8. 📈 KPIs e Observabilidade

- `usuarios_nao_verificados` (% global)
- `usuarios_parcialmente_verificados` (%)
- `taxa_conversao_para_duc` (%)
- `tentativas_bloqueadas_chat` (# por semana)
- `grupos_bloqueados` (# por mês)

Exemplo de Log Técnico:

```
{
  "event": "acao_bloqueada",
  "user_id": "u404",
  "acao": "chat_com_verificado",
  "status": "nao_verificado",
  "timestamp": "2025-09-08T22:12:00Z"
}
```

🧬 Justificativa Técnica

- Garante que **usuários Free** ainda tenham acesso, mas com barreiras sutis → incentivo natural à verificação.
- Cria diferenciação clara entre níveis (Free → Parcial → Verificado+).
- Logs permitem auditoria e métricas → entender conversão de Free para verificados.
- Balanceia **experiência positiva + segurança invisível**.

📘 CAMADA 12 — FLUXO COMPLETO DE VERIFICAÇÃO DUC (DOCUMENTO ÚNICO DE CIDADANIA)

1. 🎯 Propósito Técnico

Validar juridicamente a identidade do usuário por meio de **OCR + FaceMatch + Prova de Vida**, garantindo segurança contra perfis falsos, bots e fraudes, em conformidade com a LGPD.

2. 🔧 Etapas Técnicas do Fluxo

Etapa	Ação Técnica	Regras
1. Upload Documento	Captura frente/verso do RG, CNH, ou passaporte	Apenas formatos `.jpg/.png/.pdf`, max 5MB
2. Selfie Prova de Vida	Captura via câmera com micro-gesto (piscada/movimento)	Validação automática em tempo real
3. OCR + IA Antifraude	Extração de dados (nome, data de nascimento, validade, foto)	Bloqueio de documentos adulterados ou vencidos
4. Comparação Facial	FaceMatch entre selfie e foto do documento	Score mínimo `>=0.85`
5. Processamento	Microserviço valida metadados, assinatura digital e integridade	SLA: < 2 min
6. Resultado	Aprovação automática ou envio para revisão manual	Revisão manual em até 12h

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def verificar_duc(user_id, doc_frente, doc_verso, selfie):
    ocr_data = run_ocr([doc_frente, doc_verso])
    face_score = compare_faces(selfie, ocr_data["photo"])

    if face_score >= 0.85 and validate_doc_integrity(ocr_data):
        approve_verification(user_id, tipo="DUC")
        enable_verified_features(user_id)
```

```
        log_event(user_id, "duc_success", {"score": face_score})
    else:
        increment_attempts(user_id)
        log_event(user_id, "duc_failed", {"score": face_score})

        if get_attempts(user_id) >= 3:
            trigger_manual_review(user_id)
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u505",
  "duc_status": "approved",
  "ocr_nome": "Thayssa Gomes",
  "ocr_data_nascimento": "1997-07-11",
  "face_match_score": 0.92,
  "selfie_liveness": true,
  "timestamp": "2025-09-08T22:35:00Z"
}
```

5. 🔒 Segurança e LGPD

- **Retenção temporária**: documentos armazenados por no máximo 24h em bucket S3/GCP criptografado (AES-256).
- **Exclusão**: após verificação → *Secure Erase*.
- **Persistência mínima**: apenas status (`approved/rejected`) e dados essenciais (nome, idade).
- **Logs hashados**: cada tentativa é registrada para auditoria.

6. ⚠️ Casos de Borda

Situação	Ação Técnica
Documento ilegível	Mensagem: *“Foto ilegível. Por favor, envie novamente.”*
Falha FaceMatch (<0.85)	Retry automático até 3x, depois revisão manual
Documento adulterado detectado	Bloqueio imediato + alerta para moderação
Tentativa de fraude repetida	Usuário bloqueado temporariamente, flag `risk_score = high`
Timeout no microserviço	Retry automático com fallback de fila assíncrona

---

7. 🔗 Integrações Ativas

- **Entrada**
  - `verificacao-service` → motor OCR + FaceMatch.
- **Saída**
  - `auth-service` → vincula selo DUC ao token de autenticação.
  - `perfil-e-frequencia-service` → libera novas funções.
  - `aurah-kosmos-core` → adapta IA conforme status de verificação.
  - `logs-service` → armazena tentativas, falhas e aprovações.

---

8. 📈 KPIs e Observabilidade

- `taxa_aprovacao_duc` (% de aprovações automáticas)
- `taxa_rejeicao_duc` (% de falhas no OCR/FaceMatch)
- `tempo_medio_verificacao` (meta: < 2 min)
- `revisoes_manuais` (# por semana)
- `tentativas_fraude_bloqueadas` (# por mês)

Exemplo de Log Técnico:

```
{
  "event": "duc_verification",
  "user_id": "u505",
  "status": "reprovado",
```

```
  "motivo": "documento_ilegivel",
  "tentativa": 2,
  "timestamp": "2025-09-08T22:36:00Z"
}
```

🧬 Justificativa Técnica

- Fluxo evita cadastros falsos com OCR + IA antifraude.
- Reduz risco → usa provedores externos especializados (Unico, idwall).
- Garante **escalabilidade** via filas assíncronas → não trava UX.
- Cumpre LGPD → retenção mínima + exclusão segura.

📘 CAMADA 13 — FLUXO DCO (DOCUMENTO DE CONEXÃO ORIGINAL + INTENÇÃO ENERGÉTICA DO USUÁRIO)

1. 🎯 Propósito Técnico

O **DCO** não é um documento físico, mas sim um **manifesto digital de intenção**, que traduz a entrada do usuário em **parâmetros técnicos vibracionais**.

Serve para:

- Diferenciar perfis reais de bots ou contas vazias.
- Ativar arquétipos iniciais e intenção energética no sistema.
- Alimentar o **Mapa de Frequência inicial** com dados subjetivos transformados em vetores.

2. 🔧 Etapas Técnicas do Fluxo

Etapa	Entrada do Usuário	Processamento	Saída
1. Assinatura de Verdade	Frase livre escrita pelo usuário	NLP (sentimento + embedding)	Score de intenção + vetor textual
2. Escolha de Arquétipo	Guardião, Criativo, Introspectivo, Visionário	Validação coerência	Arquétipo armazenado
3. Microteste Vibracional	Escolhas rápidas de imagens/sensações	Tempo de resposta + consistência	Vetor complementar
4. Confirmação com IA Aurah	IA analisa tom + arquétipo	Validação final	Selo DCO simbólico
5. Registro	Persistência em banco	Hash criptográfico	Registro imutável

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def processar_dco(user_id, frase, arquetipo, respostas):
    sentimento = nlp.sentiment(frase)
    embedding = aurah.embedding(frase)
    tempo_resposta = calcular_tempo_resposta(respostas)

    if sentimento > 0.2 and validar_arquetipo(arquetipo):
        emitir_dco(user_id, frase, arquetipo, embedding)
        log_event(user_id, "dco_success", {"arquetipo": arquetipo})
        return True
    else:
        sugerir_reflexao(user_id)
        log_event(user_id, "dco_failed", {"frase": frase})
        return False
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u606",
  "frase": "Quero me reconectar com pessoas reais.",
  "sentimento": 0.74,
  "embedding": [0.11, 0.29, 0.48, ...],
  "arquetipo": "Guardião",
```

```
  "tempo_resposta": 3.2,
  "status_dco": "aprovado",
  "timestamp": "2025-09-08T22:55:00Z"
}
```

5. 🔒 Segurança e LGPD

- Frases são armazenadas em formato **hash + embedding**, nunca como texto puro.
- Dados de intenção podem ser anonimizados para análises globais.
- Revisões são possíveis caso o usuário solicite ajuste → direito LGPD.

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Frase muito curta ou genérica	IA solicita mais contexto: *"Expresse um pouco mais de você."*
Linguagem incoerente (sarcasmo, ofensa)	IA rejeita e pede reescrita
Escolha de arquétipo incoerente	IA propõe reflexão com exemplo alternativo
Microteste feito rápido demais (<500ms)	Marcado como suspeito (`risk_score += 20`)
Usuário abandona fluxo	Cadastro segue, mas com restrições leves até completar DCO

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → analisa frase, sentimentos, embeddings.
- **Saída**
  - `perfil-e-frequencia-service` → registra arquétipo e vetor inicial.
  - `feed-sensorial-core` → define curadoria inicial com base no DCO.
  - `conexoes-reais-service` → ativa tags energéticas.

- `painel-vibracional-premium` → exibe assinatura e arquétipo (Premium).

---

8. 📈 KPIs e Observabilidade

- `taxa_conclusao_dco` (% de usuários que completam o manifesto)
- `taxa_rejeicao_dco` (% de frases incoerentes/suspeitas)
- `tempo_medio_dco` (meta: 1–2 min)
- `arquétipos_mais_escolhidos` (ranking global)
- `frases_flag_suspeita` (#/semana)

Log Exemplo:

```
{
  "event": "dco_emitido",
  "user_id": "u606",
  "arquetipo": "Guardião",
  "sentimento": 0.74,
  "status": "aprovado",
  "timestamp": "2025-09-08T22:56:00Z"
}
```

🧬 Justificativa Técnica

- Converte um conceito místico (“assinatura de alma”) em **pipeline objetivo com NLP + embeddings**.
- Permite detectar **incoerências e sarcasmo** via IA, aumentando a confiabilidade.
- Garante segurança de dados com **hashing + anonimização**.
- Reforça a entrada consciente no app, mas sem bloquear totalmente quem não conclui.

📘 CAMADA 14 — GERAÇÃO DO MAPA DE FREQUÊNCIA E AURA INICIAL (DCO + TESTE ENERGÉTICO)

1. 🎯 Propósito Técnico

Consolidar os dados coletados no **DCO (Documento de Conexão Original)**, no **Teste de Personalidade Energética** e no **formulário inicial**, transformando-os em um **Mapa de Frequência Inicial**.

Este mapa é a base **matemática e simbólica** que orienta:

- Feed Sensorial,
- Conexões Autênticas,
- IA Aurah Kosmos,
- Locais Parceiros,
- Experiências Premium.

2. 🔧 Fontes de Dados e Pesos

Fonte	Peso (%)	Processamento
Arquétipo (DCO)	35%	Define o tom principal da frequência
Frase vibracional (assinatura)	25%	Analisada via NLP (embedding + sentimento)
Tempo de resposta no cadastro	10%	Média → impulsividade/reflexão
Região geográfica e horário	10%	Contexto de entrada (geolocalização)
Teste de Personalidade Energética	20%	Vetores multidimensionais (512D)

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def gerar_mapa_inicial(user_id):
    dados_dco = coletar_dados_dco(user_id)
    dados_teste = coletar_teste_energetico(user_id)
```

```
    assinatura = dados_dco["frase"]
    arquetipo = dados_dco["arquetipo"]

    sentimento = nlp.sentiment(assinatura)
    embedding = aurah.embedding(assinatura)
    tempo_resposta = dados_dco["tempo"]
    geo = get_geolocalizacao(user_id)

    mapa = calcular_frequencia(
        arquetipo=arquetipo,
        assinatura=embedding,
        tempo=tempo_resposta,
        geo=geo,
        teste=dados_teste
    )

    salvar_mapa(user_id, mapa)
    return mapa
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u707",
  "arquetipo": "Visionário",
  "assinatura": "Quero ajudar a transformar o mundo com novas ideias",
  "sentimento": 0.82,
  "embedding": [0.12, 0.44, 0.58, ...],
  "tempo_resposta": 4.2,
  "geo": "São Paulo, BR",
  "aura_inicial": "Aura Ativa",
  "zona_frequencia": "Cluster_Solar_05",
  "timestamp": "2025-09-08T23:15:00Z"
}
```

5. 🔒 Segurança e LGPD

- Dados textuais são armazenados apenas em formato **embedding + hash**, nunca como texto cru.
- Logs de mapa são versionados em `vibrational_log` → cada recalibração gera um histórico.
- Usuário pode solicitar reset de mapa → compliance LGPD (direito de eliminação).

---

6. ⚠️ Casos de Borda

Situação	Ação Técnica
Assinatura muito curta/incoerente	IA pede nova frase ou gera fallback neutro
Teste energético incompleto	Gera mapa parcial + flag `incomplete`
Localização indisponível	Sistema atribui cluster “genérico”
Vetor inconsistente	IA recalcula no próximo login
Score vibracional neutro (<0.2)	Atribuição de cor padrão cinza `#D3D3D3`

---

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → interpreta assinaturas e vetores.
  - `perfil-e-frequencia-service` → centraliza dados.
- **Saída**
  - `feed-sensorial-core` → curadoria inicial de conteúdo.
  - `conexoes-reais-service` → matching inicial.
  - `chat-core` → tom de conversação inicial.
  - `painel-vibracional-premium` → visualização avançada.

---

8. 📈 KPIs e Observabilidade

- `taxa_mapa_gerado` (% de cadastros que chegam a gerar mapa)
- `tempo_medio_mapa` (meta: < 2.5s)

- `mapas_incompletos` (%)
- `aura_inicial_mais_comum` (ranking global)
- `falhas_geracao_mapa` (# por semana)

Exemplo de Log Técnico:

```
{
  "event": "mapa_gerado",
  "user_id": "u707",
  "aura": "Aura Ativa",
  "cluster": "Solar_05",
  "tempo_execucao": 1.9,
  "status": "completo",
  "timestamp": "2025-09-08T23:16:00Z"
}
```

🧬 Justificativa Técnica

- Converte dados subjetivos (frase, arquétipo, tempo de resposta) em **parâmetros objetivos**.
- Usa embeddings NLP para representar intenção em vetores matemáticos.
- Reforça robustez via **fallbacks + recalibrações** → nunca deixa usuário sem mapa.
- Garante **escala** com processamento assíncrono (fila → workers).

📘 CAMADA 15 — ENTRADA NO MODO BOAS-VINDAS

---

1. 🎯 Propósito Técnico

Transformar a conclusão do cadastro em uma **transição guiada**, que:

- Consolida o **Mapa de Frequência** e ativa micro-missões iniciais.

- Reduz ansiedade e sobrecarga cognitiva.
- Entrega ao usuário uma experiência **imersiva, suave e auditável**.

2. 🔧 Componentes Técnicos

Elemento	Função Técnica
Mensagem de Boas-Vindas	Texto animado, gerado dinamicamente pela IA
Trilha Sonora Imersiva	Frequência sonora (432Hz)
Tela de Aura	Renderização visual do mapa inicial
Respiração Guiada (opcional)	Animação 4 ciclos (inspira/expira)
Micro-Missões	Primeiras 3 ações sugeridas

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def ativar_boasvindas(user_id):
    mapa = get_user_frequency_map(user_id)
    mensagem = aurah.generate_boasvindas(user_id)

    render_aura(mapa)
    play_audio("432hz_intro.mp3")
    display_text(mensagem)

    missoes = gerar_missoes_iniciais(user_id)
    atribuir_missoes(user_id, missoes)

    log_event(user_id, "modo_boasvindas_ativado", {"missoes": missoes})
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u808",
  "mensagem": "Você chegou. Aqui, a verdade vibra.",
  "aura_visual": "shader_cluster_solar",
  "missoes": [
```

```
    {"id": "m1", "acao": "Explorar feed"},
    {"id": "m2", "acao": "Conectar com leveza"},
    {"id": "m3", "acao": "Respirar e se observar"}
  ],
  "timestamp": "2025-09-08T23:30:00Z"
}
```

5. 🔒 Segurança e LGPD

- Nenhum dado sensível exibido → apenas outputs já processados (mapa e aura).
- Logs imutáveis (`boasvindas_log`) armazenam data/hora + missões atribuídas.
- Em menores de idade, missões são adaptadas para **exploração segura**.

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Falha ao renderizar aura	Exibir fallback estático (cor base do cluster)
Áudio não carregado	Mensagem textual apenas, sem travar fluxo
IA não gera missão	Carregar missões padrão (`feed`, `conexão`, `respiração`)
Usuário abandona fluxo	Missões ficam salvas e reativadas no próximo login

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → gera mensagem e missões iniciais.
  - `perfil-e-frequencia-service` → fornece aura e vetor inicial.
- **Saída**
  - `feed-sensorial-core` → ativa feed curado.
  - `conexoes-reais-service` → sugere conexões leves.
  - `seguranca-vibracional-core` → protege primeiras 24h.
  - `painel-vibracional-premium` → mostra indicadores extras (Premium).

8. 📈 KPIs e Observabilidade

- `taxa_boasvindas_completo` (% usuários que finalizam etapa)
- `latencia_media_renderizacao` (meta: < 2s)
- `uso_micro_missoes` (% que completam ao menos 1 missão)
- `abandono_boasvindas` (% que saem antes da conclusão)

Exemplo de Log Técnico:

```
{
  "event": "boasvindas",
  "user_id": "u808",
  "missoes_atribuidas": ["m1", "m2", "m3"],
  "status": "concluido",
  "timestamp": "2025-09-08T23:31:00Z"
}
```

🧬 Justificativa Técnica

- Evita sensação de “cadastro frio” → experiência inicial é guiada e calorosa.
- Garante **uniformidade** → todos iniciam com feed, conexões e segurança ativa.
- Reduz abandono no primeiro uso → missões simples e auditáveis.
- Se a IA falhar, há sempre fallback estável → nunca deixa o usuário sem direção.

📘 CAMADA 16 — INTEGRAÇÕES E DEPENDÊNCIAS CÍCLICAS DO ECOSSISTEMA

1. 🎯 Propósito Técnico

Mapear todas as **dependências técnicas e fluxos de dados** que conectam o **Cadastro Consciente e Perfil Vibracional** com os demais módulos do FriendApp.

Essa camada garante que os desenvolvedores saibam:

- De onde vêm os dados,
- Para onde eles fluem,
- Quais microserviços precisam estar ativos,
- Como os módulos interagem em tempo real.

---

2. 🔧 Componentes e Dependências Diretas

Módulo	Tipo de Integração	Justificativa Técnica
`auth-service`	Trigger de criação	Gera `user_id`, token inicial e autenticação JWT
`verificacao-service`	Entrada (DUC/DCO)	Processa documentos e status de verificação
`perfil-e-frequencia-service`	Leitura + Escrita	Armazena vetores, mapa de frequência e aura
`aurah-kosmos-core`	Leitura + Processamento	Interpreta arquétipos, frases e emoções em tempo real
`conexoes-reais-service`	Escrita + Sugestão	Gera conexões compatíveis logo após cadastro
`feed-sensorial-core`	Curadoria inicial	Ajusta feed conforme vetor e intenção
`seguranca-vibracional-core`	Leitura + Gatilhos	Ativa escudos e bloqueios invisíveis
`painel-vibracional-premium`	Leitura simbólica	Exibe aura, arquétipo e assinatura (Premium)

---

3. 🧠 Fluxo Técnico (Pipeline Cíclico)

```
flowchart TD
  A[Cadastro Consciente] —>|DUC/DCO| B[verificacao-service]
  A —>|Mapa Frequência| C[perfil-e-frequencia-service]
```

```
  C --> D[aurah-kosmos-core]
  D --> E[feed-sensorial-core]
  D --> F[conexoes-reais-service]
  D --> G[seguranca-vibracional-core]
  C --> H[painel-vibracional-premium]
```

4. 📊 Exemplo de Estrutura de Dados (JSON)

```
{
  "user_id": "u909",
  "auth": "jwt_token_abc",
  "duc_status": true,
  "dco_status": true,
  "vetores": {
    "essencia": [0.12, 0.54, 0.77, ...],
    "intencao": [0.34, 0.28, 0.66, ...],
    "protecao": [0.91, 0.05, 0.02, ...]
  },
  "aura": "Aura Pacífica",
  "feed_curado": true,
  "conexoes_iniciais": 5,
  "seguranca_aplicada": true,
  "timestamp": "2025-09-08T23:45:00Z"
}
```

5. 🔒 Segurança e LGPD

- Todos os dados trafegam via **API Gateway com TLS 1.3**.
- Integrações usam **OAuth2.0 com scopes específicos**.
- Logs de integração armazenados com hash imutável (`sha256`).
- Dados vibracionais sensíveis são criptografados em repouso (AES-256).

6. ⚠️ Casos de Borda

Situação	Ação Técnica
`verificacao-service` indisponível	Usuário entra em **modo restrito** até confirmação
`aurah-kosmos-core` offline	Fallback → Feed básico + conexões neutras
`perfil-e-frequencia-service` falha	Cadastro bloqueado até sincronização
Dados inconsistentes entre módulos	Trigger de **reprocessamento via fila**
Premium sem painel ativo	Reprocessar integração → fallback padrão

7. 🔗 Integrações Ativas (Entrada → Saída)

- **Entrada**
  - Dados do cadastro (nome, idade, intenção, frase, arquétipo).
- **Saída**
  - `feed-sensorial-core`: ativa feed inicial.
  - `conexoes-reais-service`: popula sugestões.
  - `seguranca-vibracional-core`: aplica proteções.
  - `painel-vibracional-premium`: gera visualização expandida.

8. 📈 KPIs e Observabilidade

- `latencia_media_integracao` (meta: < 3s)
- `taxa_falha_integracao` (%)
- `usuarios_restritos_por_falha` (#/dia)
- `taxa_sincronizacao_sucesso` (%)
- `tempo_reprocessamento_falhas` (meta: < 10s)

Exemplo de Log Técnico:

```
{
  "event": "integracao_cadastro",
  "user_id": "u909",
  "feed": "ativado",
  "conexoes": "5_sugeridas",
  "seguranca": "aplicada",
  "latencia_total": 2.7,
```

```
  "status": "sucesso",
  "timestamp": "2025-09-08T23:46:00Z"
}
```

🧬 Justificativa Técnica

- Garante que o **Cadastro Consciente não é isolado**, mas **raiz de todo o ecossistema**.
- Estrutura resiliente com **fallbacks + filas assíncronas** evita falhas críticas.
- KPIs claros permitem medir gargalos de integração.
- LGPD aplicada com retenção mínima e logs auditáveis.

📘 CAMADA 17 — TELA INICIAL DE RECEPÇÃO SENSORIAL + BARREIRA DE FREQUÊNCIA

1. 🎯 Propósito Técnico

Estabelecer o **primeiro checkpoint vibracional** antes do cadastro.

Essa tela:

- Alinha a intenção do usuário com a proposta do FriendApp.
- Bloqueia cadastros incoerentes (entretenimento, sexo, spam).
- Coleta dados de comportamento inicial para **risk_score**.

2. 🔧 Componentes Técnicos

Elemento	Função Técnica
Vídeo vibracional	Duração mínima 60s (ideal 90s), com música binaural
Texto da Alma	Exibido em sincronia com a narração

Elemento	Função Técnica
Botão “Continuar com o Coração”	Surge apenas após ≥90% do vídeo
Coleta de Interações	Tempo de permanência, tentativas de pular, swipes
Reconhecimento de voz (opcional)	Captura palavras ditas

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def portal_entrada(user_id):
    video.play("intro_portal.mp4", min_duration=60)
    log_event(user_id, "portal_start")

    interacoes = capturar_interacoes(user_id)

    if video.completed and interacoes['tempo'] >= 60:
        liberar_botao(user_id)
        log_event(user_id, "portal_complete")
    else:
        marcar_incoerencia(user_id, reason="abandono_portal")
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u1010",
  "portal_status": "completo",
  "tempo_visualizacao": 92,
  "skip_tentativas": 0,
  "frequencia_portal": "alta_coerencia",
  "voz_detectada": ["esperança", "verdade"],
  "timestamp": "2025-09-08T23:55:00Z"
}
```

5. 🔒 Segurança e LGPD

- Voz → opcional, só com consentimento explícito.
- Dados coletados (tempo, tentativas de pular, voz) → armazenados de forma anônima e criptografada.
- Logs hashados → `portal_logs` para auditoria.

---

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Usuário sai antes de 20s	Marcado como risco (`risk_score += 15`)
Tentativas de pular vídeo	Logadas como comportamento suspeito
Áudio não carrega	Exibir fallback: texto + imagem estática
Usuário abandona no fim	Status: `resonancia_fraca` , ativação parcial no cadastro

---

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → interpreta voz e intenção inicial.
- **Saída**
  - `logs-service` → armazena comportamento inicial.
  - `perfil-e-frequencia-service` → registra coerência da entrada.
  - `cadastro-flow-controller` → libera próximo passo apenas após vídeo.

---

8. 📈 KPIs e Observabilidade

- `taxa_completude_portal` (% de usuários que assistem até o fim).
- `abandono_precoce` (% antes de 20s).
- `tentativas_skip` (# por usuário).
- `tempo_medio_visualizacao` (meta: >80s).
- `coerencia_entrada` (% marcado como “alta coerência”).

Exemplo de Log Técnico:

```
{
  "event": "portal_stats",
  "user_id": "u1010",
  "tempo": 92,
  "skip": 0,
  "status": "coerente",
  "timestamp": "2025-09-08T23:56:00Z"
}
```

🧬 Justificativa Técnica

- Funciona como **firewall invisível** contra usuários incoerentes.
- Garante que apenas quem tem **intenção real** siga no fluxo.
- Transforma conceitos abstratos (barreira vibracional) em **métricas de risco mensuráveis**.
- Fornece logs e KPIs auditáveis para evolução contínua.

📘 CAMADA 18 — SISTEMA ANTIBOT, PROTEÇÃO CONTRA SPAM E CADASTROS SUSPEITOS

1. 🎯 Propósito Técnico

Proteger o FriendApp no **ponto mais vulnerável (cadastro inicial)** contra:

- Bots automatizados,
- Fluxos de spam,
- Perfis falsos ou duplicados,
- Ataques de preenchimento automático.

Essa camada atua de forma **100% invisível ao usuário humano**, sem fricção na UX.

2. 🔧 Mecanismos Técnicos Ativados

Mecanismo	Função Técnica
Fingerprinting de dispositivo	Coleta OS, browser, IP, timezone, ASN
Análise de tempo de preenchimento	Humanos = oscilações; bots = preenchimento instantâneo
Risk Score heurístico	Score contínuo (0 a 100)
BotTrap invisível	Campo oculto no HTML → apenas bots preenchem
Bloqueio por IP/ASN	IPs conhecidos de proxies/VPN maliciosos
Crosscheck IA Aurah	Detecta padrões vibracionais artificiais
Delay tático	Introduz atraso sutil em fluxos suspeitos

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def avaliar_cadastro(user_input, device_data):
    score = 0

    if detect_bottrap_click(user_input):
        score += 50
    if time_to_complete_form(user_input) < 1.5:
        score += 20
    if device_fingerprint_in_blacklist(device_data):
        score += 40
    if aurah.detect_vibrational_anomaly(user_input):
        score += 30

    if score >= 80:
        bloquear_usuario(user_input.ip)
        log_event(user_input, "cadastro_bloqueado", {"score": score})
    elif score >= 60:
        ativar_delay(user_input.session)
        log_event(user_input, "cadastro_suspeito", {"score": score})
    else:
        aprovar_usuario(user_input)
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "temp_1212",
  "fingerprint": "hash_89af32",
  "tempo_preenchimento": 1.1,
  "bottrap_detectado": true,
  "risk_score": 85,
  "acao": "bloqueado",
  "timestamp": "2025-09-09T00:10:00Z"
}
```

5. 🔒 Segurança e LGPD

- Nenhum dado sensível é coletado além do necessário para fingerprint.
- Dados anonimizados (hash do fingerprint + ASN).
- Logs armazenados por máx. 30 dias → apenas para auditoria antifraude.

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Usuário humano em rede corporativa suspeita	Delay + monitoramento, não bloqueio imediato
Preenchimento muito rápido mas legítimo	Flag `medium_risk`, aprovado após segunda validação
Bot que passa por OCR simulado	Crosscheck IA Aurah → detecta padrão vibracional incoerente
Ataque em massa (IP flooding)	Firewall ativa bloqueio ASN + rate limiting API Gateway

7. 🔗 Integrações Ativas

- **Entrada**

- `verificacao-service` → bloqueia cadastros repetidos com mesmo doc/email.
- `aurah-kosmos-core` → avalia coerência vibracional.

- **Saída**
  - `logs-service` → registra cadastros suspeitos/bloqueados.
  - `firewall-vibracional` → ativa escudos baseados em ASN/IP.
  - `perfil-e-frequencia-service` → marca perfis de risco (`flag_risco`).

---

8. 📈 KPIs e Observabilidade

- `taxa_bloqueio_bots` (% cadastros bloqueados)
- `falsos_positivos` (% usuários humanos bloqueados por engano)
- `tempo_medio_deteccao` (meta: <1s)
- `cadastros_flag_medium_risk` (#/semana)
- `ataques_mitigados` (#/mês)

Exemplo de Log Técnico:

```
{
  "event": "antibot",
  "user_id": "temp_1212",
  "risk_score": 85,
  "acao": "bloqueado",
  "detalhes": ["bottrap", "preenchimento_instantaneo"],
  "timestamp": "2025-09-09T00:12:00Z"
}
```

🧬 Justificativa Técnica

- Substitui CAPTCHA → UX fluida, sem fricção.
- Risk Score contínuo permite **resposta graduada** (delay, bloqueio, monitoramento).
- Logs auditáveis → diferenciar bots de falsos positivos.

- IA Aurah atua como filtro final, detectando incoerências não captadas pelos heurísticos.

📘 CAMADA 19 — SISTEMA DE PERGUNTAS VIBRACIONAIS + PERFIL EXPANDIDO

1. 🎯 Propósito Técnico

Permitir que o usuário expresse sua **essência, intenção e estado emocional atual** através de perguntas vibracionais cuidadosamente desenhadas.

Essa camada não cria rótulos fixos, mas sim dados dinâmicos que alimentam:

- **Mapa de Frequência**
- **Feed Sensorial**
- **Conexões Autênticas**
- **Modo Viagem & Eventos**

2. 🔧 Categorias de Perguntas

Categoria	Objetivo Técnico
Frequência emocional atual	Captar o estado do usuário (leve, denso, expansivo).
Intenção no app	Identificar motivação inicial (amizade, grupos, acolhimento).
Modo de processar emoções	Mapear padrões de reação (reflexão, impulsividade).
Forma de acolhimento preferida	Adaptar tom da IA (palavras, gestos, silêncio).
Ciclo de vida atual	Reconhecer contexto (cura, expansão, recomeço).

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def interpretar_respostas(user_id, respostas):
    perfil = {}

    perfil["essencia"] = aurah.analisar_palavras(respostas["palavra_mund
o"])
    perfil["estado_emocional"] = detectar_frequencia(respostas["energia_ho
je"])
    perfil["intencao"] = mapear_intencao(respostas["pulsando_agora"])
    perfil["acolhimento"] = respostas["forma_acolhimento"]
    perfil["ciclo_vida"] = classificar_ciclo(respostas["ciclo"])

    salvar_perfil_vibracional(user_id, perfil)
    log_event(user_id, "perguntas_vibracionais", perfil)

    return perfil
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u1313",
  "essencia": "curiosidade introspectiva",
  "estado_emocional": "leve",
  "intencao": "amizades reais",
  "acolhimento": "palavras",
  "ciclo_vida": "recomeço",
  "timestamp": "2025-09-09T00:25:00Z"
}
```

5. 🔒 Segurança e LGPD

- Respostas armazenadas em **formato hash + embedding**, não em texto cru.
- Usuário pode **editar ou apagar** respostas a qualquer momento (direito LGPD).

- Logs de respostas → append-only, versionados para auditoria.

---

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Resposta em branco	IA gera sugestão neutra → não bloqueia fluxo
Linguagem ofensiva	Flag `risk_score += 25` + moderação leve
Resposta muito curta	IA sugere reflexão: *"Expresse um pouco mais de você."*
Usuário menor de idade	IA adapta perguntas para versão simplificada

---

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → NLP das respostas.
- **Saída**
  - `mapa-frequencia-core` → atualiza mapa vibracional.
  - `feed-sensorial-core` → ajusta conteúdos.
  - `conexoes-reais-service` → filtra compatibilidades emocionais.
  - `eventos-service` → sugere experiências alinhadas.
  - `selos-service` → gera identificadores fluídos.

---

8. 📈 KPIs e Observabilidade

- `taxa_resposta_completa` (% que respondem todas perguntas).
- `tempo_medio_respostas` (meta: 2–3 min).
- `respostas_flag_suspeita` (# por semana).
- `usuarios_revisando_respostas` (% que ajustam depois).
- `impacto_no_matching` (melhora nas conexões sugeridas).

Exemplo de Log Técnico:

```
{
  "event": "resposta_vibracional",
```

```
  "user_id": "u1313",
  "categoria": "intencao",
  "valor": "amizades reais",
  "status": "valido",
  "timestamp": "2025-09-09T00:26:00Z"
}
```

🧬 Justificativa Técnica

- Transforma o abstrato (“essência”, “acolhimento”) em **dados estruturados**.
- Garante personalização em múltiplos sistemas (feed, conexões, IA).
- Permite evolução contínua → perguntas podem ser reabertas em ciclos futuros.
- Reduz riscos de respostas incoerentes com **IA + validações automáticas**.

📘 CAMADA 20 — BANCO DE DADOS ESTRUTURADO PARA ESSÊNCIA + FREQUÊNCIA

1. 🎯 Propósito Técnico

Definir a **arquitetura de armazenamento** dos dados do Cadastro Consciente e Perfil Vibracional, garantindo:

- Segurança (LGPD + criptografia),
- Versionamento (logs imutáveis),
- Performance (consultas rápidas),
- Suporte híbrido (SQL para dados fixos, NoSQL para dados dinâmicos).

2. 🔧 Modelo Híbrido

PostgreSQL (SQL) → dados relacionais (identidade e verificações).

Tabela `users`

Campo	Tipo	Descrição
user_id (PK)	UUID	Identificador único
nome	VARCHAR(120)	Nome escolhido
email	VARCHAR(255)	Opcional
idade	INT	Validada via DUC/DCO
verificacao_duc	BOOLEAN	Status DUC
verificacao_dco	BOOLEAN	Status DCO
created_at	TIMESTAMP	Data de criação
updated_at	TIMESTAMP	Última atualização

Tabela `vibrational_log`

Campo	Tipo	Descrição
log_id (PK)	UUID	ID do log
user_id (FK)	UUID	Referência ao usuário
evento	VARCHAR	Tipo de evento (mudança, teste, aura)
dados_vibracionais	JSONB	Dados resumidos
timestamp	TIMESTAMP	Hora do evento

Firestore (NoSQL) → dados vivos e mutáveis (vetores, aura, histórico).

Coleção `/perfil_vibracional/{user_id}`

```
{
  "essencia": [0.12, 0.44, 0.66, ...],
  "aura_atual": {
    "frequencia": "Solar",
    "cor": "#F0E68C",
    "intensidade": 0.82
  },
  "historico": [
    {"data": "2025-09-01", "estado": "leve"},
```

```
    {"data": "2025-09-07", "estado": "expansivo"}
  ],
  "intencao": "amizades reais",
  "vetores": {
    "essencia": [...],
    "intencao": [...],
    "protecao": [...]
  },
  "last_update": "2025-09-09T00:40:00Z"
}
```

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def salvar_perfil_vibracional(user_id, perfil):
    # Registro SQL
    db.sql("INSERT INTO vibrational_log (user_id, evento, dados_vibracionai
s, timestamp) VALUES (...)")

    # Registro NoSQL
    firestore.collection("perfil_vibracional").document(user_id).set({
        "essencia": perfil["essencia"],
        "aura_atual": perfil["aura"],
        "historico": perfil["historico"],
        "intencao": perfil["intencao"],
        "vetores": perfil["vetores"],
        "last_update": datetime.utcnow()
    })
```

4. 🔒 Segurança e LGPD

- Criptografia em repouso: AES-256.
- Criptografia em trânsito: TLS 1.3.
- Retenção mínima → logs não são apagados, apenas versionados (*append-only*).

- Usuário pode solicitar **anonimização completa** (LGPD).

---

5. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Conflito SQL/NoSQL	Priorizar Firestore (última escrita) + sincronização automática
Falha no Firestore	Salvar somente em PostgreSQL → reprocessar na fila
Payload grande (>1MB)	Compressão GZIP antes do insert
Solicitação de exclusão LGPD	Anonimização imediata → manter apenas metadados legais

---

6. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → grava vetores e mapas.
- **Saída**
  - `feed-sensorial-core` → lê dados de intenção e aura.
  - `conexoes-reais-service` → usa vetores para matching.
  - `eventos-service` → filtra sugestões.
  - `painel-vibracional-premium` → exibe aura/histórico.

---

7. 📈 KPIs e Observabilidade

- `latencia_gravacao_sql` (meta: < 200ms)
- `latencia_gravacao_nosql` (meta: < 500ms)
- `erros_sincronizacao` (#/dia)
- `tamanho_medio_documento` (NoSQL)
- `solicitacoes_anonimizacao` (#/mês)

Exemplo de Log Técnico:

```
{
  "event": "perfil_salvo",
```

```
  "user_id": "u1313",
  "status_sql": "ok",
  "status_nosql": "ok",
  "latencia_sql": 0.12,
  "latencia_nosql": 0.43,
  "timestamp": "2025-09-09T00:41:00Z"
}
```

🧬 Justificativa Técnica

- Modelo híbrido garante **confiabilidade (SQL)** + **flexibilidade (NoSQL)**.
- Evita perda de dados → versionamento e redundância.
- Respeita LGPD → direito de exclusão e anonimização.
- Escalável para milhões de usuários com baixo custo de consulta.

📘 CAMADA 21 — VETORES DE INTENÇÃO EVOLUTIVA

1. 🎯 Propósito Técnico

Traduzir os dados coletados no **cadastro consciente** (DUC, DCO, Teste Energético, Perguntas Vibracionais) em **vetores matemáticos multidimensionais**, que serão usados pela IA Aurah Kosmos para:

- Guiar a evolução do usuário,
- Proteger contra conexões incoerentes,
- Personalizar todo o ecossistema do FriendApp.

---

2. 🔧 Estrutura dos Vetores

Vetor	Finalidade	Fonte de Dados	Dimensão
`vetor_essencia`	Representa essência central do usuário	DCO (frase, arquétipo) + Teste Energético	512D
`vetor_intencao`	Direção evolutiva atual	Perguntas vibracionais + interações recentes	512D
`vetor_protecao`	Barreiras invisíveis de segurança	Radar emocional + IA antifraude	256D

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def gerar_vetores(user_id, dados):
    essencia = aurah.embedding(dados["frase"] + dados["arquétipo"])
    intencao = aurah.embedding(dados["intencao"] + dados["respostas"])
    protecao = aurah.protecao_energetica(dados)

    salvar_vetores(user_id, essencia, intencao, protecao)
    log_event(user_id, "vetores_gerados")

    return essencia, intencao, protecao
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u1414",
  "vetor_essencia": [0.12, 0.55, 0.78, ...],
  "vetor_intencao": [0.32, 0.18, 0.66, ...],
  "vetor_protecao": [0.02, 0.99, 0.05, ...],
  "last_update": "2025-09-09T00:50:00Z"
}
```

5. 🔒 Segurança e LGPD

- Vetores são **hashados + criptografados** em repouso.

- Não armazenam texto cru, apenas embeddings numéricos.
- Direito de exclusão → vetores podem ser removidos sob solicitação LGPD.
- Logs versionados para auditoria de evolução.

---

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Dados insuficientes para vetor	IA gera fallback neutro (`[0,0,0,...]`) até recalibração
Inconsistência entre vetores	IA prioriza `vetor_protecao` para segurança
Vetor corrompido	Reprocessamento automático via fila assíncrona
Alteração de intenção brusca	Disparo de evento `nova_intencao_detectada` → recalibração imediata

---

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → gera embeddings e recalibra vetores.
- **Saída**
  - `feed-sensorial-core` → ajusta conteúdos com base em intenção.
  - `conexoes-reais-service` → matching por distância vetorial.
  - `seguranca-vibracional-core` → usa `vetor_protecao` para firewall invisível.
  - `eventos-service` → sugere experiências compatíveis.
  - `jogo-transmutacao` → mede evolução do usuário.

---

8. 📈 KPIs e Observabilidade

- `taxa_geracao_vetores` (% cadastros com vetores completos).
- `tempo_medio_calculo` (meta: < 1s).
- `usuarios_recalibrados` (% recalibrados/dia).
- `falhas_geracao_vetores` (#/semana).

- `distancia_media_matching` (métrica de compatibilidade).

Exemplo de Log Técnico:

```
{
  "event": "vetores_update",
  "user_id": "u1414",
  "essencia_hash": "a91f7c",
  "intencao_hash": "b33d9a",
  "protecao_hash": "c28f0d",
  "status": "completo",
  "timestamp": "2025-09-09T00:51:00Z"
}
```

🧬 Justificativa Técnica

- Representa conceitos vibracionais como **dados vetoriais objetivos**.
- Garante escalabilidade (vetores armazenados em Firestore e Redis para baixa latência).
- Protege a comunidade com `vetor_protecao`.
- Permite evolução contínua → recalibração periódica e sob eventos.

📘 CAMADA 22 — GATILHOS AURAH + PERSONALIZAÇÃO INTELIGENTE DA IA

---

1. 🎯 Propósito Técnico

Transformar os **vetores vibracionais** em **gatilhos dinâmicos** que ajustam continuamente a experiência do usuário no FriendApp.

Esses gatilhos:

- Personalizam feed, conexões e eventos,
- Reforçam a segurança em tempo real,

- Permitem evolução adaptativa da jornada.

2. 🔧 Tipos de Gatilhos

Tipo	Ativação	Função Técnica
Inicial	Logo após cadastro	Define tom da primeira experiência
Periódico (evolutivo)	A cada 7 dias OU após 10 interações significativas	Recalibra vetores e ajusta feed/conexões
Proteção (imediato)	Ao detectar interação incoerente ou abusiva	Bloqueia ou suaviza conexões em tempo real
Nova Intenção	Usuário altera padrão vibracional	Reconfigura feed, IA e eventos sugeridos

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def processar_gatilhos(user_id):
    vetores = obter_vetores(user_id)
    aura = obter_aura(user_id)

    if cadastro_recente(user_id):
        disparar_gatilho("inicial", user_id)

    if precisa_recalibrar(user_id):
        disparar_gatilho("periodico", user_id)

    if detectar_risco(user_id):
        disparar_gatilho("protecao", user_id)

    if nova_intencao_detectada(user_id, vetores):
        disparar_gatilho("nova_intencao", user_id)
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u1515",
  "gatilho": "periodico",
  "acao_tomada": [
    "feed_recalibrado",
    "sugestoes_atualizadas",
    "firewall_reforcado"
  ],
  "timestamp": "2025-09-09T01:00:00Z"
}
```

5. 🔒 Segurança e LGPD

- Logs de gatilhos armazenados em `gatilhos_log`, com hash imutável.
- Nenhum dado sensível adicional coletado.
- Alterações no perfil são **transparentes e auditáveis**.
- Usuário pode consultar quando sua aura foi recalibrada.

6. ⚠️ Casos de Borda

Situação	Ação Técnica
Falha no processamento IA	Usar últimos vetores válidos como fallback
Gatilhos em excesso	Limitador → máx. 3 recalibrações/dia
Usuário discorda de ajuste	Botão *"Não corresponde"* → IA registra feedback
Evento crítico (toxidade alta)	Ativação instantânea do firewall vibracional

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → gera e processa gatilhos.
  - `perfil-e-frequencia-service` → fornece vetores e aura.
- **Saída**

- `feed-sensorial-core` → recalibra conteúdo.
- `conexoes-reais-service` → atualiza matching.
- `firewall-vibracional` → reforça bloqueios invisíveis.
- `mapa-frequencia-core` → reposiciona usuário no ecossistema.

---

8. 📈 KPIs e Observabilidade

- `gatilhos_disparados_total` (#/dia)
- `gatilhos_periodicos` (%)
- `gatilhos_protecao` (%)
- `tempo_medio_execucao` (meta: < 2s)
- `feedback_negativo_usuarios` (% usuários que pedem correção)

Exemplo de Log Técnico:

```
{
  "event": "gatilho_disparado",
  "user_id": "u1515",
  "tipo": "protecao",
  "motivo": "linguagem_toxica",
  "acao": "conexao_bloqueada",
  "timestamp": "2025-09-09T01:01:00Z"
}
```

🧬 Justificativa Técnica

- Garante que o FriendApp **não é estático**, mas evolui junto com o usuário.
- Torna o abstrato (“mudança de intenção”) em **triggers técnicos claros**.
- Proporciona **segurança em tempo real** com firewall vibracional.
- Reforça transparência com logs e opção de feedback do usuário.

📘 CAMADA 23 — LOGS TÉCNICOS E OBSERVABILIDADE DO CADASTRO CONSCIENTE

1. 🎯 Propósito Técnico

Garantir **rastreabilidade total**, **auditoria contínua** e **monitoramento em tempo real** de todo o fluxo de cadastro consciente.

Essa camada assegura que cada evento, desde o portal sensorial até a geração do Mapa de Frequência, seja registrado de forma **imutável e auditável**.

2. 🔧 Tipos de Logs

Tipo	Conteúdo Registrado	Exemplo
Logs de Entrada	Portal, formulário, DUC/DCO, teste energético	`cadastro.iniciado`, `duc_verificacao`
Logs de Vetores e Aura	Vetores de essência, intenção, proteção; recalibrações	`vetores_update`
Logs de Gatilhos IA	Inicial, periódico, proteção, nova intenção	`gatilho_disparado`
Logs de Segurança	Detecção de bot, fraude, comportamento suspeito	`antibot_detectado`

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def registrar_log(user_id, evento, dados):
    log = {
        "timestamp": datetime.utcnow(),
        "user_id": user_id,
        "evento": evento,
        "dados": dados,
        "hash": gerar_hash(dados)
    }
    elasticsearch.index(index="cadastro_logs", document=log)
```

```
    return log
```

4. 📊 Estrutura de Dados (Exemplo JSON)

```
{
  "timestamp": "2025-09-09T01:15:00Z",
  "user_id": "u1616",
  "evento": "duc_verificacao",
  "dados": {
    "status": "aprovado",
    "face_score": 0.91,
    "ocr_nome": "Thayssa Gomes"
  },
  "hash": "e7c1d91b27f3a..."
}
```

5. 🔒 Segurança e LGPD

- Todos os logs possuem **hash criptográfico (SHA-256)** para evitar adulteração.
- Dados sensíveis são **anonimizados** (ex.: nome → hash).
- Retenção: 12 meses, com possibilidade de exportação para auditoria LGPD.
- Acesso restrito a moderadores e DevOps via VPN corporativa.

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Falha no envio para ElasticSearch	Retentar via fila (`retry_log_queue`)
Latência alta (>2s)	Priorizar logs críticos; logs secundários armazenados localmente até sincronização
Log corrompido	Rejeitado e marcado para reprocessamento

Situação	Tratamento Técnico
Solicitação de exclusão LGPD	Dados anonimizados; log preserva apenas hash

7. 🔗 Integrações Ativas

- **Entrada**
  - `cadastro-flow-controller` → dispara eventos por etapa.
  - `aurah-kosmos-core` → envia logs de vetores e gatilhos.
- **Saída**
  - `logs-service` → armazena logs imutáveis.
  - `observabilidade-core` → dashboards em Kibana/Grafana.
  - `seguranca-vibracional-core` → monitora eventos de risco em tempo real.

8. 📈 KPIs e Observabilidade

- `latencia_media_log` (meta: < 200ms)
- `percentual_logs_sucesso` (% de logs indexados sem erro)
- `eventos_criticos_detectados` (#/dia)
- `tempo_medio_reprocessamento` (meta: < 5s)
- `falhas_envio_logs` (#/semana)

Exemplo de Log Observável:

```
{
  "dashboard": "cadastro_logs",
  "metricas": {
    "latencia_media": 0.18,
    "logs_sucesso": 99.7,
    "falhas_envio": 3
  },
  "periodo": "2025-09-09T00:00:00Z → 2025-09-09T01:00:00Z"
}
```

🧬 Justificativa Técnica

- Garante **transparência total** do fluxo de cadastro.
- Facilita auditorias internas e compliance LGPD.
- Detecta rapidamente falhas ou fraudes via dashboards.
- Cria um histórico imutável → útil para evolução contínua da IA Aurah.

📘 CAMADA 24 — UX SENSORIAL DE ENTRADA E PÓS-CADASTRO

1. 🎯 Propósito Técnico

Definir a **experiência visual, sonora e interativa** imediatamente após o término do cadastro consciente, garantindo que o usuário:

- Tenha uma **transição suave** entre cadastro e uso real,
- Sinta impacto emocional positivo,
- Seja guiado de forma clara para os próximos passos.

2. 🔧 Componentes da UX Sensorial

Elemento	Função Técnica
Transição Final	Animação da aura do usuário expandindo em tela cheia
Música Harmônica	Frequência 432Hz, duração 15–20s
Mensagem Personalizada	Texto gerado pela IA Aurah
Microtour Guiado	Roteiro para explorar feed, mapa de frequência e preferências

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def pos_cadastro(user_id):
    aura = get_user_aura(user_id)
```

```
    mensagem = aurah.generateMensagemBoasVindas(user_id)

    render_transicao_aura(aura)
    play_audio("432hz_welcome.mp3")
    display_text(mensagem)

    iniciar_microtour(user_id, steps=["feed", "mapa", "preferencias"])
    log_event(user_id, "pos_cadastro_completo")
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u1717",
  "aura_visual": "shader_fractal_pacifica",
  "mensagem": "Seu campo vibracional foi reconhecido. Bem-vindo(a) à su
a nova rede de conexões verdadeiras.",
  "microtour_steps": ["feed", "mapa", "preferencias"],
  "timestamp": "2025-09-09T01:30:00Z"
}
```

5. 🔒 Segurança e LGPD

- Nenhum dado sensível é exibido nesta etapa.
- Logs armazenam apenas status de conclusão do onboarding.
- Caso o usuário seja menor de idade, mensagens e microtour são adaptados para **ambiente protegido**.

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Falha ao renderizar aura	Exibir cor base estática do arquétipo
Áudio não carregado	Continuar com mensagem textual sem travar fluxo
Microtour não iniciado	CTA de fallback: *“Explorar feed agora”*

Situação	Tratamento Técnico
Usuário abandona antes do final	Microtour fica disponível no próximo login

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → gera mensagem personalizada.
  - `perfil-e-frequencia-service` → fornece dados de aura inicial.
- **Saída**
  - `feed-sensorial-core` → prepara feed curado para exploração.
  - `mapa-frequencia-core` → habilita visualização inicial.
  - `conexoes-reais-service` → sugere primeiras conexões.

8. 📈 KPIs e Observabilidade

- `taxa_conclusao_poscadastro` (% usuários que finalizam etapa).
- `tempo_medio_transicao` (meta: < 2s).
- `abandono_poscadastro` (% que saem antes de concluir).
- `uso_microtour` (% que completam roteiro guiado).

Exemplo de Log Técnico:

```
{
  "event": "pos_cadastro",
  "user_id": "u1717",
  "status": "concluido",
  "microtour_completo": true,
  "latencia_transicao": 1.7,
  "timestamp": "2025-09-09T01:31:00Z"
}
```

🧬 Justificativa Técnica

- Garante que o cadastro **não termine em vazio**, mas em uma experiência marcante.
- Reduz ansiedade inicial → transição guiada + microtour.
- Cria primeiro vínculo emocional com a IA Aurah.
- Estrutura robusta → sempre há fallback em caso de falha de renderização ou áudio.

📘 CAMADA 25 — ALERTAS INTELIGENTES + UX PROTEGIDA

1. 🎯 Propósito Técnico

Garantir que interações incoerentes, arriscadas ou não verificadas sejam **sinalizadas de forma sutil e protetiva**, sem quebrar a experiência sensorial do FriendApp.

Essa camada cria uma **rede invisível de segurança**, mantendo a liberdade do usuário, mas prevenindo abusos.

2. 🔧 Tipos de Alertas

Tipo	Contexto	Exemplo de Mensagem
Preventivos	Conexão entre verificado e não verificado	"Este perfil ainda não concluiu a verificação DUC. Conecte-se com consciência."
De Segurança	Linguagem abusiva detectada	"Esta interação foi limitada para sua proteção vibracional."
Educativos	Incentivo à verificação ou boas práticas	"Perfis verificados têm acesso a experiências mais seguras e completas."

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def verificar_interacao(user_a, user_b):
    if not user_b.is_verified():
        exibir_alerta(user_a, tipo="preventivo", codigo="perfil_nao_verificad
```

```
o")

    if aurah.detectar_risco(user_b):
        bloquear_interacao(user_a, user_b)
        exibir_alerta(user_a, tipo="seguranca", codigo="interacao_bloquead
a")

    log_event(user_a, "alerta_emitido", {"alvo": user_b.id})
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u1818",
  "tipo_alerta": "preventivo",
  "descricao": "perfil_nao_verificado",
  "contexto": "chat",
  "alvo": "u2222",
  "acao": "exibir_banner",
  "timestamp": "2025-09-09T01:45:00Z"
}
```

5. 🔒 Segurança e LGPD

- Alertas não expõem dados pessoais, apenas status de verificação ou risco.
- Logs de alertas são hashados em `alertas_log` para auditoria.
- Usuário pode consultar histórico de alertas aplicados em seu perfil (transparência LGPD).

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Usuário contesta alerta	Registro enviado ao `painel-admin` para revisão manual

Situação	Tratamento Técnico
Múltiplos alertas em sequência	Limite de exibição → no máx. 1 alerta por interação
Alerta exibido em erro (falso positivo)	Usuário pode sinalizar via botão *"Não corresponde"*
Usuário menor de idade	Alertas reforçados e exibidos de forma mais clara

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → avalia risco em tempo real.
  - `verificacao-service` → status DUC/DCO.
- **Saída**
  - `chat-core` → exibe banners nos diálogos.
  - `conexoes-reais-service` → injeta avisos em convites.
  - `seguranca-vibracional-core` → bloqueia interações de alto risco.

8. 📈 KPIs e Observabilidade

- `taxa_alertas_emitidos` (#/semana).
- `taxa_interacoes_bloqueadas` (% sobre total de interações).
- `falsos_positivos` (% de alertas contestados).
- `tempo_medio_resposta_alerta` (meta: < 1s).
- `usuarios_com_alertas_reincidentes` (#/mês).

Exemplo de Log Técnico:

```
{
  "event": "alerta_emitido",
  "user_id": "u1818",
  "alvo": "u2222",
  "tipo": "seguranca",
  "codigo": "linguagem_toxica",
  "acao": "bloqueio",
```

```
  "timestamp": "2025-09-09T01:46:00Z"
}
```

🧬 Justificativa Técnica

- Substitui alertas genéricos por **mensagens contextuais e discretas**.
- Mantém experiência fluida → nunca trava a navegação sem motivo.
- Protege especialmente **menores e vulneráveis**.
- Garante **auditoria e transparência** via logs imutáveis.

📘 CAMADA 26 — TESTES A/B E OTIMIZAÇÃO DO FLUXO DE ENTRADA

1. 🎯 Propósito Técnico

Permitir que o **fluxo de cadastro consciente** evolua continuamente através de **testes controlados (A/B e multivariáveis)**, medindo impacto em:

- Taxa de conclusão do cadastro,
- Tempo médio por etapa,
- Engajamento inicial,
- Redução de abandono.

2. 🔧 Elementos Passíveis de Teste

Elemento	Variações Possíveis
Ordem das perguntas vibracionais	Antes ou depois do DCO
Tipo de entrada inicial	Vídeo 60s vs 90s vs versão interativa
Mensagem da IA Aurah	Estilo poético vs direto vs acolhedor
Microinterações	Botões animados vs estáticos
Trilhas sonoras	432Hz suave vs binaural profundo

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def fluxo_cadastro(user_id):
    variante = ab_test_engine.assign_variant("cadastro_flow", user_id)

    if variante == "A":
        executar_fluxo_A(user_id)
    elif variante == "B":
        executar_fluxo_B(user_id)
    else:
        executar_fluxo_padrao(user_id)

    log_event(user_id, "ab_test_assigned", {"variante": variante})
```

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u1919",
  "experimento": "cadastro_flow",
  "variante": "B",
  "tempo_conclusao": 210,
  "concluido": true,
  "timestamp": "2025-09-09T02:00:00Z"
}
```

5. 🔒 Segurança e LGPD

- Nenhum dado sensível usado para definição de variantes.
- Variantes atribuídas via `ab_test_engine` → pseudo-random com persistência.
- Logs de participação armazenados para auditoria, mas sempre anonimizados.

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Usuário troca de dispositivo	Variantes mantidas via token persistente
Variante com desempenho ruim	Auto-desativação pelo `feature-flags-engine`
Falha no motor A/B	Fallback → fluxo padrão
Overload de variantes simultâneas	Limite de máx. 5 variações ativas por teste

---

7. 🔗 Integrações Ativas

- **Entrada**
  - `feature-flags-engine` → controle de ativação de testes.
  - `aurah-kosmos-core` → adapta mensagens e estilos testados.
- **Saída**
  - `observabilidade-core` → coleta métricas.
  - `feed-sensorial-core` → ajusta conteúdos iniciais.

---

8. 📈 KPIs e Observabilidade

- `taxa_conclusao_variante_A/B` (% usuários que concluem cadastro).
- `tempo_medio_variante` (s).
- `abandono_variante` (%).
- `satisfacao_inicial` (micro feedback, 1–5).
- `melhor_variante_ativa` (automação).

Exemplo de Log Técnico:

```
{
  "event": "ab_test_result",
  "user_id": "u1919",
  "experimento": "cadastro_flow",
  "variante": "B",
  "resultado": {
    "conclusao": true,
```

```
    "tempo_total": 210,
    "feedback": 4
  },
  "timestamp": "2025-09-09T02:01:00Z"
}
```

🧬 Justificativa Técnica

- Garante que o fluxo evolua com **dados reais de uso**, não apenas hipóteses.
- Reduz abandono → otimização contínua baseada em métricas.
- Mantém UX sensorial → testes nunca quebram estética.
- Escalável → múltiplos experimentos em paralelo com `feature-flags`.

📘 CAMADA 27 — PAINEL ADMINISTRATIVO INTERNO DO CADASTRO

1. 🎯 Propósito Técnico

Fornecer uma interface segura e estruturada para **moderadores e equipe técnica** acompanharem, auditarem e intervirem em casos especiais do fluxo de cadastro consciente.

O painel deve permitir:

- Visualizar status de verificação, aura e vetores,
- Analisar tentativas suspeitas,
- Gerir erros e cadastros pendentes,
- Realizar ações manuais seguras (ex.: aprovar documento).

2. 🔧 Funcionalidades Principais

Área	Função Técnica
Dashboard de Entrada	Total de cadastros, taxa de abandono, alertas ativos
Filtro por Status	Exibir “Verificado”, “Pendente”, “Bloqueado”
Detalhe do Usuário	Histórico vibracional, logs de tentativas, vetores
Gestão de Erros	Reprocessar IA, desbloquear cadastros, corrigir falhas
Ações Manuais	Aprovar/reprovar documento, disparar recalibração
Segurança	Autenticação MFA + controle de permissões

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
@app.route("/painel/cadastro/<user_id>")
@login_required
@require_role("moderador", "admin")
def visualizar_cadastro(user_id):
    dados = obter_dados_usuario(user_id)
    logs = obter_logs(user_id)
    return render_template("painel_cadastro.html", dados=dados, logs=logs)

@app.route("/painel/cadastro/acao", methods=["POST"])
def acao_moderador():
    payload = request.json
    if payload["acao"] == "aprovar_doc":
        aprovar_documento(payload["user_id"])
    elif payload["acao"] == "reprocessar":
        reprocessar_fluxo(payload["user_id"])
    log_event(payload["user_id"], "acao_moderador", payload)
```

4. 📊 Estrutura de Dados (Exemplo JSON)

```
{
  "user_id": "u2020",
  "status_verificacao": "pendente",
  "duc": "aguardando",
  "dco": "completo",
```

```
  "aura": "Aura Pacífica",
  "vetores": {
    "essencia": [0.12, 0.44, 0.55],
    "intencao": [0.34, 0.22, 0.11],
    "protecao": [0.98, 0.04, 0.03]
  },
  "moderador_responsavel": "admin01",
  "ultimo_evento": "duc_reprovado",
  "timestamp": "2025-09-09T02:15:00Z"
}
```

5. 🔒 Segurança e LGPD

- **Autenticação obrigatória** com 2FA.
- Acesso restrito a IPs internos e VPN.
- Todas as ações manuais registradas em `logs-admin`.
- Dados exibidos no painel → mascarados quando sensíveis (ex.: RG, selfie).

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Moderador acessa sem permissão	Bloqueio imediato + log de tentativa
Documento reprovado injustamente	Usuário pode recorrer → análise manual
Fluxo travado por erro da IA	Moderador pode disparar “reprocessar fluxo”
Sobrecarga de acessos	Painel desativa consultas pesadas e prioriza logs críticos

7. 🔗 Integrações Ativas

- **Entrada**
  - `verificacao-service` → status documental.
  - `perfil-e-frequencia-service` → dados de aura e vetores.

- `logs-service` → histórico de tentativas.

- **Saída**
  - `moderacao-service` → ações manuais de aprovação/reprovação.
  - `observabilidade-core` → métricas de uso do painel.

---

8. 📈 KPIs e Observabilidade

- `tempo_medio_resolucao` (meta: < 12h).
- `acoes_manuais_totais` (# por semana).
- `erros_reprocessados` (#).
- `tentativas_acesso_negadas` (#/mês).
- `usuarios_pendentes` (% cadastros não concluídos).

Exemplo de Log Técnico:

```
{
  "event": "acao_moderador",
  "user_id": "u2020",
  "acao": "aprovar_doc",
  "moderador": "admin01",
  "timestamp": "2025-09-09T02:16:00Z"
}
```

🧬 Justificativa Técnica

- Permite que a equipe tenha **controle operacional** em casos excepcionais.
- Mantém compliance LGPD → dados mascarados e logs de todas ações.
- Reforça resiliência → moderadores podem corrigir falhas da IA.
- Dashboard centraliza status e agiliza resolução de pendências.

📘 CAMADA 28 — API E ENDPOINTS DO CADASTRO CONSCIENTE

1. 🎯 Propósito Técnico

Definir todos os **endpoints e contratos técnicos** responsáveis por orquestrar o fluxo de Cadastro Consciente e Perfil Vibracional.

A camada garante **segurança, clareza e escalabilidade** na integração entre frontend e backend.

2. 🔧 Arquitetura

- **API Gateway** → centraliza requisições, aplica rate limiting e autenticação.
- **REST JSON** como padrão de comunicação.
- **JWT temporário** gerado no início do cadastro → válido apenas durante a sessão.

3. 🧠 Endpoints Principais

Endpoint	Método	Função	Autenticação
`/api/cadastro/iniciar`	POST	Cria sessão temporária de cadastro	Não (gera JWT)
`/api/cadastro/duc`	POST	Upload frente/verso documento	JWT sessão
`/api/cadastro/selfie`	POST	Upload de selfie + prova de vida	JWT sessão
`/api/cadastro/dco`	POST	Registro de frase + arquétipo	JWT sessão
`/api/cadastro/teste-vibracional`	POST	Respostas do teste energético	JWT sessão
`/api/cadastro/finalizar`	POST	Conclui fluxo e retorna vetores + aura	JWT sessão
`/api/cadastro/status`	GET	Consulta status atual do cadastro	JWT sessão

4. 📊 Exemplo de Requisição/Resposta

Início do Cadastro

Request

```
POST /api/cadastro/iniciar
{
  "nome": "Thayssa",
  "email": "thayssa@email.com",
  "idade": 27
}
```

Response

```
{
  "user_id": "u2828",
  "token": "jwt_temp_123",
  "status": "iniciado"
}
```

Finalização do Cadastro

Request

```
POST /api/cadastro/finalizar
{
  "user_id": "u2828"
}
```

Response

```
{
  "status": "concluido",
  "vetores": {
    "essencia": [0.12, 0.55, 0.78, ...],
```

```
    "intencao": [0.32, 0.18, 0.66, ...],
    "protecao": [0.02, 0.99, 0.05, ...]
  },
  "aura_inicial": "Aura Ativa",
  "token_autenticacao": "jwt_auth_456"
}
```

5. 🔒 Segurança e LGPD

- **Autenticação JWT** → renovação automática a cada etapa.
- **Refresh token seguro** armazenado apenas em Keychain (iOS) / Secure Enclave (Android).
- **Rate limiting** → máx. 5 tentativas/minuto por endpoint sensível.
- **Logs imutáveis** → todos os requests/responses registrados em `logs-api`.

6. ⚠️ Casos de Borda

Situação	Resposta API
Dados inválidos	HTTP 400 → `{ "erro": "dados_invalidos" }`
Sessão expirada	HTTP 401 → `{ "erro": "sessao_expirada" }`
Documento não reconhecido	HTTP 422 → `{ "erro": "documento_invalido" }`
Erro interno	HTTP 500 → `{ "erro": "erro_interno" }`

7. 🔗 Integrações Ativas

- **Entrada**
  - `verificacao-service` → valida DUC/DCO.
  - `aurah-kosmos-core` → gera vetores.
- **Saída**
  - `perfil-e-frequencia-service` → grava perfil vibracional.
  - `logs-service` → registra eventos de API.

8. 📈 KPIs e Observabilidade

- `latencia_media_endpoints` (meta: < 200ms)
- `taxa_erro_endpoints` (%)
- `cadastros_concluidos_api` (#/dia)
- `tentativas_invalidas` (#/semana)
- `abandono_fluxo_api` (%)

Exemplo de Log Técnico:

```
{
  "event": "api_call",
  "endpoint": "/api/cadastro/duc",
  "user_id": "u2828",
  "status": "sucesso",
  "latencia": 0.23,
  "timestamp": "2025-09-09T02:25:00Z"
}
```

🧬 Justificativa Técnica

- Define um **contrato estável** para integração frontend-backend.
- Mantém UX fluida com JWT temporário e refresh seguro.
- Escalável → suporta milhões de cadastros simultâneos com API Gateway.
- Observabilidade completa → logs + KPIs evitam gargalos e falhas silenciosas.

📘 CAMADA 29 — TRADUÇÃO MULTILÍNGUE E LOCALIZAÇÃO CULTURAL DO FLUXO

---

1. 🎯 Propósito Técnico

Garantir que o **Cadastro Consciente e Perfil Vibracional** seja acessível em múltiplos idiomas e adaptado a diferentes culturas, preservando a **essência sensorial** do FriendApp.

---

2. 🔧 Estratégia Multilíngue

Componente	Implementação
Idiomas suportados	Português (PT-BR/PT-PT), Inglês, Espanhol, Francês, Alemão
Futuro	Árabe (RTL), Japonês, Chinês
Motor de Tradução	IA Aurah + fallback `i18n` (arquivos JSON)
Estilo	Tradução adaptativa (sem literalismo)

---

3. 🧠 Pipeline Técnico (Pseudocódigo)

```
def traduzir_texto(chave, locale, variaveis=None):
    if locale in traducoes:
        texto = traducoes[locale].get(chave, traducoes["en"][chave])
    else:
        texto = traducoes["en"][chave]

    if variaveis:
        texto = substituir_variaveis(texto, variaveis)

    return aurah.ajustar_tom(texto, locale)
```

---

4. 📊 Estrutura de Arquivos (Exemplo)

```
/locales/en/cadastro.json
/locales/es/cadastro.json
/locales/pt/cadastro.json
```

Exemplo de `cadastro.json`:

```
{
  "boas_vindas": "Bem-vindo(a) à sua nova jornada",
  "continuar": "Continuar com o coração",
  "erro_documento": "Não conseguimos validar seu documento. Tente nova
mente."
}
```

5. 🔒 Segurança e LGPD

- Mensagens traduzidas não armazenam dados pessoais.
- Todas as strings dinâmicas passam por sanitização → previne injeção.
- Logs de idioma preferido do usuário armazenados apenas como `locale`.

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Idioma não suportado	Fallback automático para Inglês
Texto não encontrado	Exibe chave default do `i18n`
Diferença cultural (ex.: idade mínima)	Ajuste dinâmico conforme legislação local
Fluxo RTL (árabe/hebraico)	UI adaptada automaticamente (`dir="rtl"`)

7. 🔗 Integrações Ativas

- **Entrada**
  - `aurah-kosmos-core` → adapta mensagens sensoriais ao idioma.
  - `feature-flags-engine` → ativa idiomas por região.
- **Saída**
  - `observabilidade-core` → coleta métricas de performance por idioma.
  - `ui-render-service` → renderiza telas adaptadas (RTL, layout cultural).

8. 📈 KPIs e Observabilidade

- `usuarios_por_idioma` (%).
- `erros_traducao_detectados` (#/semana).
- `tempo_medio_renderizacao_localizada` (meta: <200ms).
- `usuarios_fallback_en` (% que caíram no fallback).
- `satisfacao_localizada` (NPS segmentado por idioma).

Exemplo de Log Técnico:

```
{
  "event": "localizacao_fluxo",
  "user_id": "u2929",
  "locale": "es",
  "fallback": false,
  "latencia": 0.15,
  "timestamp": "2025-09-09T02:40:00Z"
}
```

🧬 Justificativa Técnica

- Garante expansão global sem perder **identidade vibracional**.
- Evita erros culturais e legais (ex.: idade mínima adaptada).
- Escalável → novos idiomas adicionados via `i18n` + IA Aurah.
- Transparência → métricas de fallback e satisfação monitoradas.

📘 CAMADA 30 — INTEGRAÇÕES E DEPENDÊNCIAS CÍCLICAS DO ECOSSISTEMA

---

1. 🎯 Propósito Técnico

Consolidar todas as **integrações diretas e indiretas** do **Cadastro Consciente e Perfil Vibracional** com o ecossistema FriendApp.

Essa camada garante que o cadastro funcione como **raiz do sistema**, alimentando e sincronizando todos os módulos.

2. 🔧 Fluxo Resumido

```
flowchart TD
  A[Cadastro Consciente] ⟶ B[DUC/DCO + Teste Energético]
  B ⟶ C[Mapa de Frequência + Vetores]
  C ⟶ D[Aurah Kosmos IA Central]
  D ⟶ E[Feed Sensorial]
  D ⟶ F[Conexões Autênticas]
  D ⟶ G[Segurança Vibracional]
  D ⟶ H[Eventos & Modo Viagem]
  C ⟶ I[Painel Vibracional Premium]
```

3. 🧠 Estrutura de Dados e Dependências

Funcionalidade	Dados Recebidos	Dependência
Verificação (DUC/DCO)	Documentos validados, selfie, frase vibracional	`verificacao-service`
Mapa de Frequência	Vetores, aura inicial	`perfil-e-frequencia-service`
IA Aurah Kosmos	Vetores, intenção, estado emocional	`aurah-kosmos-core`
Conexões Autênticas	Vetores + preferências	`conexoes-reais-service`
Feed Sensorial	Vetores + intenção evolutiva	`feed-sensorial-core`
Firewall Vibracional	Risk Score inicial	`seguranca-vibracional-core`
Modo Viagem + Bora	Aura inicial + intenção	`travel-service`
Eventos & Experiências	Dados vibracionais de compatibilidade	`eventos-service`
Painel Premium	Aura + histórico	`painel-vibracional-premium`

4. 📊 Representação de Dados (Exemplo JSON)

```
{
  "user_id": "u3030",
  "duc": true,
  "dco": true,
  "vetores": {
    "essencia": [0.12, 0.55, 0.78],
    "intencao": [0.32, 0.18, 0.66],
    "protecao": [0.02, 0.99, 0.05]
  },
  "aura_inicial": "Aura Solar",
  "feed": "curado",
  "conexoes_sugeridas": 8,
  "modo_viagem": "ativo",
  "painel_vibracional": true,
  "timestamp": "2025-09-09T02:55:00Z"
}
```

5. 🔒 Segurança e LGPD

- Todas as integrações passam por **API Gateway com TLS 1.3**.
- Tokens JWT com escopo limitado por serviço.
- Logs de integração armazenados em `logs-integracao` com hash.
- Dados vibracionais sensíveis nunca trafegam em texto cru, apenas como **vetores anonimizados**.

6. ⚠️ Casos de Borda

Situação	Tratamento Técnico
Serviço crítico indisponível (ex.: IA Aurah)	Fallback → feed e conexões baseadas apenas no arquétipo e assinatura
Sincronização falha entre módulos	Trigger de reprocessamento via fila assíncrona

Situação	Tratamento Técnico
Usuário Premium sem painel ativo	Reprocessar integração até 3x → fallback com versão simplificada
Dados divergentes (SQL vs NoSQL)	Priorizar NoSQL (última escrita) e gerar log de inconsistência

---

7. 🔗 Integrações Ativas

- **Entrada**
  - Dados do cadastro (DUC/DCO, frases, arquétipo, respostas vibracionais).
- **Saída**
  - `feed-sensorial-core` → conteúdos personalizados.
  - `conexoes-reais-service` → sugestões vibracionais.
  - `eventos-service` → experiências compatíveis.
  - `painel-vibracional-premium` → aura expandida.
  - `seguranca-vibracional-core` → firewall invisível.

---

8. 📈 KPIs e Observabilidade

- `taxa_integracao_sucesso` (%)
- `tempo_medio_integracao` (meta: <3s)
- `falhas_reprocessadas` (#/dia)
- `usuarios_afetados_por_falhas` (%)
- `latencia_por_servico` (Aurah, Feed, Conexões, Painel)

Exemplo de Log Técnico:

```
{
  "event": "integracao_cadastro",
  "user_id": "u3030",
  "servicos": ["feed", "conexoes", "aurah"],
  "status": "sucesso",
  "latencia_total": 2.8,
```

```
  "timestamp": "2025-09-09T02:56:00Z"
}
```

🧬 Justificativa Técnica

- Consolida o Cadastro Consciente como **núcleo do ecossistema FriendApp**.
- Estrutura resiliente → sempre existe fallback em falhas críticas.
- Observabilidade garante identificação de gargalos.
- Prepara o sistema para **escala global**, com arquitetura modular e auditável.