

# MANUAL TÉCNICO DEFINITIVO SISTEMA ECONÔMICO, MONETIZAÇÃO E FRIENDCOINS (FRIENDAPP)

## ◆ CAMADA 0 — PRINCÍPIO FILOSÓFICO + DIRETRIZES DE EXECUÇÃO ECONÔMICA

### Objetivo

Definir o **núcleo filosófico e matemático** do sistema econômico do FriendApp, garantindo equilíbrio entre **propósito vibracional** e **sustentabilidade financeira real**, prevenindo que o app se torne um jogo viciante ou apenas acumulativo.

### Fundamentos

#### 1. Economia da Consciência

- Toda transação representa não apenas valor digital, mas **circulação energética**.
- Moeda interna = **FriendCoins (FC)** → não convertível em dinheiro real, mas com valor funcional concreto no ecossistemaDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

#### 2. Equilíbrio entre Impacto e Ação

- O ganho de moedas não é fixo apenas por "ação", mas ponderado pelo **impacto vibracional da ação**.
- Fórmula base de distribuição (antes de ajustes antifraude e inflação):

$$\text{FriendCoins\_Ganho} = (\text{Peso\_Ação} * \text{Índice\_Impacto}) * \text{Fator\_Aleatório}$$

Onde:

- **Peso\_Ação** = valor inicial de referência (ex: criar Bora = 10).

- **Índice\_Impacto** = score de impacto medido pela IA Aurah Kosmos (0.0 a 2.0).
- **Fator\_Aleatório** = componente surpresa (0.8 a 1.2) que garante imprevisibilidade e evita grinding.

### 3. Controle Anti-Grinding

- Recompensas decrescem em interações repetitivas.
- Exemplo: Se usuário interage 10 vezes seguidas com o mesmo contato, as recompensas caem progressivamente (100%, 80%, 60% ... até mínimo de 10%).

### 4. Princípio do Presente

- A IA Aurah Kosmos pode premiar com **bônus surpresa**:

Exemplo de output para usuário:

"Sua interação com [Nome] atingiu uma harmonia rara. O universo reconhece e você recebe +50 FCs."

## Diretrizes Técnicas

- **Medidas Concretas de Variáveis:**
  - Frequência\_Energética = score vibracional médio do usuário nas últimas 24h (0 a 100).
  - Tempo\_Consciente = tempo ativo (toques, digitação, interação real) ÷ tempo total app aberto.
  - Índice\_Impacto = (Feedback Positivo + Resposta Vibracional + Engajamento Autêntico) ÷ 3.
- **Educação do Usuário**
  - Painel Sensorial mostra dicas contextuais → sem revelar fórmulas exatas, mas educando de forma prática.

## Blindagem Contra Abusos

- IA Antifraude roda análises de **grafo social** (Neo4j) BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM... para detectar clusters de interações fechadas (farming).

- Retornos decrescentes aplicados automaticamente → uma amizade muito usada não gera farm infinito.

---

## Saída da Camada

- Estabelecimento do **pilar filosófico + matemático**.
- Garantia de que o sistema nunca se desvirtue em “caça a moedas”, mas preserve autenticidade.
- Base para as próximas camadas (arquitetura, APIs, fórmulas avançadas de inflação e antifraude).

## ◆ CAMADA 1 — ARQUITETURA GERAL DO SISTEMA ECONÔMICO

### Objetivo

Definir a **arquitetura macro** do Sistema Econômico, integrando todos os fluxos de geração, uso e monitoramento de FriendCoins, Premium e monetização externa (eventos, locais, doações).

---

### Componentes Principais

#### 1. Moeda Interna — FriendCoins (FC)

- Ganha por check-ins, eventos, desafios, missões, indicações, postsDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....
- Uso: benefícios, boosts, ingressos, experiências, skins, RA, stickersDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....
- Armazenada em **users\_wallet** (Firestore + PostgreSQL).

#### 2. Plano Premium

- Ativado via middleware de pagamentos (ex: RevenueCat/Chargebee)BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
- Benefícios: feed completo, filtros, eventos exclusivos, check-in duplo, jogos ilimitados.
- Estados possíveis: **Ativo, Pendente, Cancelado, Expirado**.

#### 3. Sistema de Indicação

- 11 convites = 1 mês Premium grátis + FriendCoins extrasDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....
- Rastreado em **transactions\_log** com hash único de convite.

#### 4. Monetização de Locais Parceiros

- Locais pagam por destaque em mapa, feed, eventosDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....
- Contratos e pagamentos → **partners\_payments** (PostgreSQL).

#### 5. Eventos Pagos

- Usuário pode criar eventos pagos em FC ou moeda real.
- FriendApp retém % da transação (configurável).

#### 6. Sistema de Doações

- Dinheiro real ou FC para causas sociaisDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....
- 100% rastreável em **donations\_log**.

---

### Arquitetura de Dados

- **Firestore (NoSQL)** → saldo atual, status Premium, check-ins em tempo realBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
  - **PostgreSQL (SQL)** → histórico de transações, logs, contratos, pagamentosBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
  - **Neo4j (Grafo)** → análise antifraude, clusters econômicos, relações de redeBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
- 

### Pipeline Operacional

#### 1. Entrada

- Dados de interações, check-ins, eventos, missões, pagamentosBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

#### 2. Processamento

- IA Econômica + IA Antifraude.
- Cálculo de moedas, verificação de impacto, controle de inflação, antifraude.

### 3. Saída

- FriendCoins creditadas/debitadas.
- Atualização de status Premium.
- Logs financeiros gravados.
- Painéis de observabilidade atualizados.

---

### Segurança e Governança

- Criptografia AES-256 em trânsito e repousoBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
- Firewall econômico + firewall vibracional.
- Compliance → LGPD + GDPR.
- Auditoria em **logs econômicos + logs energéticos**.

---

### Saída da Camada

- Modelo **macro da arquitetura econômica** definido.
- Fluxo operacional fechado da moeda → do usuário ao back-end.
- Garantia de **segurança, rastreabilidade e escalabilidade global**.

## ◆ CAMADA 2 — MODELAGEM MATEMÁTICA DE RECOMPENSAS E IMPACTO VIBRACIONAL

### Objetivo

Definir fórmulas matemáticas exatas para calcular ganhos de FriendCoins de forma justa, transparente e alinhada ao propósito vibracional, evitando **grinding** e garantindo que **impacto > ação**.

### Fórmula Geral de Recompensa

Cada ação dentro do FriendApp gera FriendCoins com base em:

$$FC\_Ganho = (Peso\_Ação * Índice\_Impacto * Fator\_Surpresa) * Redução\_Repetição$$

### Variáveis definidas:

- **Peso\_Ação (PA):** valor base de referência da ação.
  - Post no Feed = 4
  - Check-in = 5
  - Participar de evento = 10
  - Indicar amigo = 15
- **Índice\_Impacto (II):** métrica de qualidade vibracional (0,0 a 2,0).

$$II = (\text{Feedback\_Pos} + \text{Engajamento\_Autêntico} + \text{Resposta\_Vibracional}) / 3$$

- Feedback\_Pos = média de avaliações positivas (0 a 2).
- Engajamento\_Autêntico = proporção de respostas genuínas ÷ totais.
- Resposta\_Vibracional = score de harmonia calculado pela IA Aurah Kosmos (0 a 2).
- **Fator\_Surpresa (FS):** (0,8 a 1,2)
  - Randomizado pela IA Aurah Kosmos.
  - Evita previsibilidade e incentiva experiências mágicas.
- **Redução\_Repetição (RR):**

$$RR = 1 / (1 + \ln(N_{\text{interações\_mesmas}}))$$

- N\_interações\_mesmas = número de vezes que o mesmo par de usuários interagiu nas últimas 72h.
- Evita farming entre pares fixos.

## Exemplos Práticos

### 1. Usuário faz check-in em local parceiro:

PA = 5  
II = 1,4  
FS = 1,1  
RR = 1 (primeira vez)  
 $FC\_Ganho = (5 * 1,4 * 1,1) * 1 = 7,7 \approx 8 \text{ FC}$

## 2. Usuário posta no Feed com baixo impacto:

PA = 4  
II = 0,5  
FS = 1,0  
RR = 1  
 $FC\_Ganho = (4 * 0,5 * 1,0) * 1 = 2 \text{ FC}$

## 3. Usuário participa de evento repetido com mesmo grupo:

PA = 10  
II = 1,6  
FS = 0,9  
RR = 0,5  
 $FC\_Ganho = (10 * 1,6 * 0,9) * 0,5 = 7,2 \approx 7 \text{ FC}$

## Ajustes Dinâmicos

- **Controle de Inflação:** IA ajusta o **Peso\_Ação** em tempo real se detectar excesso de moedas circulando.
- **Recompensas Surpresa:** bônus aleatórios (ex: +50 FC) concedidos pela IA quando interações alcançam picos de harmonia.
- **Missões Semanais:** incentivos extras para estimular gasto e evitar estagnação.

## Segurança e Antifraude

- Análise de grafo social (Neo4j) detecta clusters fechados com interações perfeitas → risco de farming.
- Logs auditáveis em **transactions\_log** com métricas de cálculo completas.
- Retornos decrescentes aplicados automaticamente após repetição.

## Saída da Camada

- Fórmula **100% definida e aplicável** para cálculo de recompensas.
- Variáveis transparentes e rastreáveis.
- Garantia de justiça, imprevisibilidade e proteção contra abusos.

## ◆ CAMADA 3 — ESTRUTURA DE BANCO DE DADOS ECONÔMICO (FIRESTORE + POSTGRESQL + NEO4J)

### Objetivo

Definir a **modelagem de dados completa** que sustenta o sistema econômico, garantindo **velocidade, rastreabilidade e antifraude**. Combina:

- **Firestore (NoSQL)** → operações em tempo real (saldo, check-ins, Premium).
- **PostgreSQL (SQL)** → histórico auditável e financeiro.
- **Neo4j (Grafo)** → análise antifraude, clusters e comportamento econômicoBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

## Estrutura de Dados

### 1. Firestore — Operações Imediatas

Coleções principais:

- **users\_wallet**

```
{
  "user_id": "UUID",
  "saldo_FC": 230,
  "status_premium": "ativo",
  "ultima_transacao": "2025-09-13T14:20:00Z",
```



```
"checkins_ativos": 3
}
```

- **active\_benefits**

```
{
  "user_id": "UUID",
  "beneficio": "checkin_duplo",
  "expira_em": "2025-09-20T23:59:59Z"
}
```

## 2. PostgreSQL — Histórico e Logs

Tabelas:

- **transactions\_log**

```
CREATE TABLE transactions_log (
  id SERIAL PRIMARY KEY,
  user_id UUID NOT NULL,
  tipo VARCHAR(50), -- ganho, gasto, doacao, upgrade
  valor INT NOT NULL,
  indice_impacto FLOAT,
  fator_surpresa FLOAT,
  reducao_repeticao FLOAT,
  timestamp TIMESTAMP DEFAULT NOW()
);
```

- **partners\_payments**

```
CREATE TABLE partners_payments (
  id SERIAL PRIMARY KEY,
  partner_id UUID NOT NULL,
  valor NUMERIC(10,2),
  status VARCHAR(20), -- pendente, pago, falhou
```

```
contrato JSONB,  
timestamp TIMESTAMP DEFAULT NOW()  
);
```

- **donations\_log**

```
CREATE TABLE donations_log (  
  id SERIAL PRIMARY KEY,  
  user_id UUID NOT NULL,  
  destino VARCHAR(100),  
  valor INT,  
  moeda VARCHAR(10), -- FC ou BRL  
  auditoria_hash VARCHAR(64),  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

### 3. Neo4j — Grafo Econômico

Nós e arestas:

- **Nodes:** Usuários, Locais, Eventos.
- **Edges:** Interações, Doações, Indicações.

Exemplo de query para detectar clusters suspeitos (farming):

```
MATCH (u:User)-[:INTERAGE_COM]→(v:User)  
WITH u, collect(v) as conexoes  
WHERE size(conexoes) < 3  
RETURN u
```

### Integração entre Bancos

- **Firestore → PostgreSQL:** replicação assíncrona a cada transação concluída.

- **PostgreSQL → Neo4j**: batch diário com interações e transações para análise antifraude.
  - **Fallback**: caso Firestore caia, operações básicas de wallet migradas para PostgreSQL em modo degradado.
- 

## **Segurança**

- AES-256 em todos os bancos.
  - Hash SHA-256 para registros de auditoria em **donations\_log**.
  - Backup multi-region (RPO ≤ 5min / RTO ≤ 1min).
- 

## **Saída da Camada**

- Estrutura de dados **definida e pronta para execução**.
- Combinação de **tempo real + histórico auditável + análise antifraude**.
- Base sólida para camadas de APIs e fluxos econômicos.

## **CAMADA 4 — PIPELINE OPERACIONAL E FLUXO DE PROCESSAMENTO ECONÔMICO**

### **Objetivo**

Definir como os **dados econômicos** entram, são processados, verificados e retornam em forma de FriendCoins, status Premium ou registros financeiros. Este pipeline garante **baixa latência (<300ms)** e **consistência global**  
BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

---

## **Estrutura do Pipeline**

### **1. Entrada de Dados**

- Fontes principais:
  - Check-ins vibracionais
  - Posts no feed sensorial
  - Participação em eventos
  - Doações (FriendCoins ou moeda real)

- Indicações aceitas
- Compras via gateway de pagamento
- Formato padrão de request:

```
{
  "user_id": "UUID",
  "acao": "checkin_local",
  "timestamp": "2025-09-15T16:05:23Z",
  "contexto": {
    "local_id": "UUID",
    "impacto": 1.4,
    "fator_surpresa": 1.1
  }
}
```

---

## 1. Processamento em Camadas

- **IA Econômica**
  - Calcula recompensas conforme fórmula da Camada 2.
  - Ajusta pesos dinâmicos para evitar inflação.
- **IA Antifraude**
  - Verifica inconsistências (duplicação, bots, farming).
  - Executa análise em grafo (Neo4j).
- **IA Vibracional (Aurah Kosmos)**
  - Atribui índice de impacto (II).
  - Identifica momentos de harmonia → bônus surpresa.
- **Módulo de Premium**
  - Middleware de gestão de assinaturas (RevenueCat/Chargebee).
  - Consulta status e autorizações.

---

## 1. Saída / Outputs

- **Atualização de Wallet (Firestore):** saldo atualizado em tempo real.

- **Registro em transactions\_log (PostgreSQL):** histórico completo da transação.
- **Atualização de Painéis:**
  - Painel Financeiro (admin).
  - Painel do Usuário (saldo, histórico).
  - Heatmap econômico (clusters em expansão/colapso).
- **Alertas:** eventos suspeitos enviados para revisão manual.

## Fluxo Lógico (Pseudocódigo)

```
def processar_acao(user_id, acao, contexto):
    dados = validar_request(user_id, acao, contexto)
    indice_impacto = aurah_kosmos.calcular_indice(contexto)
    fc = calcular_friendcoins(acao, indice_impacto, contexto)

    if antifraude.detectar_anomalia(user_id, contexto):
        log_evento("fraude_suspeita", user_id, contexto)
        return {"status": "revisao", "saldo": wallet.get(user_id)}

    wallet.atualizar(user_id, fc)
    transactions_log.inserir(user_id, acao, fc, indice_impacto)

    if premium.check(user_id) == "ativo":
        aplicar_beneficios(user_id, acao)

    return {"status": "ok", "fc": fc, "saldo": wallet.get(user_id)}
```

## Segurança e Resiliência

- **Redis Cache** → mantém saldo em memória para latência mínimaBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
- **Failover Econômico** → caso um nó caia, Redis replica para PostgreSQL.
- **Modo Degradado** → se IA Antifraude ficar offline, sistema assume apenas IA Econômica com limitação de recompensas.

## Saída da Camada

- Pipeline econômico 100% definido.
- Processamento rápido, auditável e protegido contra abusos.
- Pronto para integração com APIs (Camada 5).

## ◆ CAMADA 5 — APIs ECONÔMICAS CORE (WALLET, PREMIUM, DOAÇÕES, PARCEIROS)

### Objetivo

Definir os **endpoints centrais** do sistema econômico, garantindo padronização REST, segurança, rastreabilidade e integração total com o ecossistema BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

### Estrutura de Endpoints

#### 1. Wallet

- **GET /api/wallet/get**

Consulta saldo atual, status Premium e benefícios ativos.

```
{
  "user_id": "UUID",
  "saldo_FC": 245,
  "status_premium": "ativo",
  "beneficios": ["checkin_duplo", "skin_ra"]
}
```

- **POST /api/wallet/transaction**

Realiza transação (ganho/gasto de FC).

```
{
  "user_id": "UUID",
  "tipo": "ganho",
  "acao": "checkin_local",
  "valor": 8,
```

```
"indice_impacto": 1.4,  
"fator_surpresa": 1.1  
}
```

## 2. Premium

- **POST /api/premium/upgrade**

Ativa/renova plano Premium (mensal, anual, vitalício).

Integração via **RevenueCat/Chargebee** → abstrai complexidade das app stores.

- **GET /api/premium/status**

Retorna status Premium e histórico de upgrades.

```
{  
  "user_id": "UUID",  
  "status": "ativo",  
  "plano": "anual",  
  "expira_em": "2026-09-15"  
}
```

## 3. Parceiros

- **POST /api/partner/payment**

Processa pagamento de locais/eventos parceiros.

```
{  
  "partner_id": "UUID",  
  "valor": 500.00,  
  "moeda": "BRL",  
  "status": "pago"  
}
```

- **GET /api/partner/contracts**

Lista contratos ativos e pagamentos vinculados.

---

## 4. Doações

- **POST /api/donation/send**

Envia doação em FriendCoins ou moeda real.

```
{
  "user_id": "UUID",
  "destino": "ProjetoX",
  "valor": 100,
  "moeda": "FC",
  "auditoria_hash": "sha256(xxx)"
}
```

- **GET /api/donation/history**

Histórico completo de doações (compliance LGPD/GDPR).

---

## Padrões Técnicos

- **Autenticação:** OAuth 2.0 + JWT.
  - **Criptografia:** AES-256 em payloads sensíveis.
  - **Logs:** toda chamada registrada em `economy_logs`.
  - **Rate Limiting:** 100 req/min por usuário.
  - **Timeout:** 200ms para Wallet, 500ms para Premium/Parceiros.
- 

## Integração com Outros Sistemas

- **Aurah Kosmos** → valida impacto vibracional em transações.
  - **Neo4j Antifraude** → todas as transações passam por verificação de cluster.
  - **Firestore** → update imediato de wallet.
  - **PostgreSQL** → persistência de logs e contratos.
  - **Painéis Admin** → visualização de métricas, logs e heatmaps.
-



## Saída da Camada

- Conjunto de APIs **core** definido e pronto para implementação.
- Fluxo padronizado para usuários, parceiros e doações.
- Infraestrutura blindada contra fraudes e falhas de pagamento.

## ◆ CAMADA 6 — OBSERVABILIDADE E PAINÉIS ECONÔMICOS (LOGS, HEATMAPS, DASHBOARDS)

### Objetivo

Definir a camada de **observabilidade econômica e vibracional**, permitindo que times internos e usuários Premium acompanhem a circulação de FriendCoins, assinaturas, doações e clusters econômicos em tempo real  
BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

### Componentes de Observabilidade

#### 1. Painel de Logs Econômicos

- Registro **100% rastreável** de todas as transações.
- Variáveis logadas:
  - user\_id, tipo\_transacao, valor, índice\_impacto, fator\_surpresa, RR (redução repetição).
  - timestamp, origem (feed, evento, check-in, doação).
- Exemplo de log em JSON:





```
{
  "user_id": "UUID",
  "tipo": "ganho",
  "valor": 15,
  "acao": "indicacao",
  "indice_impacto": 1.8,
  "fator_surpresa": 1.05,
  "RR": 1,
  "timestamp": "2025-09-15T18:45:20Z"
```

```
}
```

## 2. Painel Financeiro (Admin)

- Indicadores:
  - Volume de FriendCoins circulando.
  - Número de assinaturas Premium ativas.
  - Total de pagamentos de parceiros.
  - Doações realizadas (FC + BRL).
- Exportável em CSV/JSON para auditorias externas.

## 3. Heatmaps Econômicos + Energéticos

- **Mapa global e local** destacando clusters de atividadeBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
- Cores:
  -  Verde → economia em expansão.
  -  Vermelho → colapso/queda.
  -  Amarelo → pico temporário.
  -  Azul → transição vibracional.
- Exemplo de query para gerar heatmap em SQL:

```
SELECT local_id, SUM(valor) as total_fc
FROM transactions_log
WHERE timestamp > NOW() - interval '24 hours'
GROUP BY local_id
ORDER BY total_fc DESC;
```

## 4. Painel de Doações

- Transparência total (destino, valor, hash de auditoria).

- Exibido em tempo real no app e no admin.

---

## 5. Painel Vibracional Econômico

- Métricas combinadas de economia + vibração:
  - Pico energético → interações altamente harmônicas.
  - Colapso → excesso de moedas sem gasto.
  - Transição → novos clusters se formando.

---

### Requisitos Técnicos

- **Backend:** Grafana + Kibana integrados aos bancos.
- **Frontend:** React Native + Web Admin.
- **Latência:** ≤ 1s atualização nos dashboards.
- **Segurança:** acessos diferenciados (usuário, Premium, admin, auditoria).
- **Armazenamento:** logs replicados por 5 anos (compliance LGPD/GDPR).

---

### Saída da Camada

- Observabilidade completa da economia.
- Transparência para usuários e investidores.
- Ferramentas internas para auditoria e ajustes dinâmicos.

## ◆ CAMADA 7 — SEGURANÇA ECONÔMICA E BLINDAGEM ANTIFRAUDE

### Objetivo

Blindar o sistema econômico contra **fraudes, abusos e manipulações coordenadas**, preservando a integridade vibracional e financeira do FriendApp  
BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

---

### Ameaças Identificadas

1. **Coin Farming:** grupos criando interações falsas para gerar moedas em massa.

2. **Bots Automatizados:** scripts simulando check-ins e posts.
3. **Duplicação de Transações:** tentativas de reenvio de requisições (replay attacks).
4. **Fraudes em Doações:** desvio de recursos ou manipulação de valores.
5. **Ataques a Assinaturas Premium:** uso indevido de receipts falsificados.

---

## Estratégias de Blindagem

### 1. IA Antifraude (Neo4j + ML)

- Análise de **grafo social** para identificar clusters fechados.
- Sinalização automática de padrões suspeitos:

```
MATCH (u:User)-[:INTERAGE_COM]→(v:User)
WITH u, collect(v) as conexoes
WHERE size(conexoes) < 3
RETURN u
```

- Modelo de ML classifica comportamento como **legítimo** ou **suspeito** (precision ≥ 92%).

---

### 2. Retornos Decrescentes

- Redução automática de recompensas em interações repetidas com o mesmo grupo.
- Fórmula:

```
RR = 1 / (1 + ln(N_interações_mesmas))
```

- Evita que pares ou pequenos clusters “farmem” moedas indefinidamente.

---

### 3. Firewall Econômico

- **Firewall Digital:** bloqueio de requisições anômalas (ex: >50 check-ins em 1h).

- **Firewall Vibracional:** IA identifica padrões de energia incoerentes (ex: feedbacks sempre máximos entre mesmos perfis).
- 

## 4. Logs e Auditoria

- Todas as transações registradas com:
    - Hash SHA-256.
    - Timestamp imutável.
    - Geolocalização aproximada.
  - Amostragem aleatória enviada para **auditoria manual**.
- 

## 5. Blindagem de Premium e Pagamentos

- Middleware (RevenueCat/Chargebee) → valida recibos das app stores.
  - Tokens de pagamento validados em tempo real via HMAC.
  - Queda do provedor → failover automático para modo "Premium temporário" (usuário não é penalizado).
- 



## Resposta a Incidentes

- **Detecção:** IA Antifraude sinaliza anomalia.
  - **Isolamento:** cluster suspeito entra em modo "restrito" (recompensas suspensas).
  - **Análise:** logs enviados ao painel admin.
  - **Correção:** rollback imediato se fraude confirmada.
- 



## Saída da Camada

- Blindagem antifraude definida em múltiplas camadas.
- Prevenção de abusos (farming, bots, duplicação).
- Garantia de justiça e sustentabilidade econômica.

## ◆ CAMADA 8 — DISASTER RECOVERY PLAN (DRP) E RESILIÊNCIA ECONÔMICA

## Objetivo

Garantir a **continuidade do sistema econômico** em cenários de falha, ataque cibernético ou corrupção de dados, assegurando que **nenhuma transação de FriendCoins, Premium ou doações seja perdida**

BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

---

## Cenários de Risco

1. **Falha de Servidor Econômico:** indisponibilidade total de um nó de processamento.
  2. **Queda no Sistema de Moedas:** inconsistência entre Firestore e PostgreSQL.
  3. **Ataque Cibernético:** DDoS, injeção maliciosa ou fraude massiva.
  4. **Corrupção de Dados:** perda ou alteração de registros transacionais.
  5. **Queda de Middleware de Pagamentos:** falha em RevenueCat/Chargebee ou app stores.
- 

## Estratégias de Resiliência

### 1. Failover Automático

- Redistribuição instantânea de cargas para outros nós.
  - Replicação multi-region → latência < 1s.
  - Firestore e PostgreSQL configurados em **clusters redundantes**.
- 

### 2. Backup e Rollback

- Snapshots de PostgreSQL a cada 5 min (RPO ≤ 5).
  - Backups incrementais criptografados (AES-256).
  - Rollback automático da última versão consistente caso corrupção seja detectada.
- 

### 3. Modo Degradado

- Caso IA Antifraude ou Aurah Kosmos fiquem offline:

- Sistema econômico opera apenas com **cálculo base das moedas** (sem impacto vibracional).
  - Premium e saldo mantidos localmente no cache Redis.
  - Logs marcados como "modo\_degradado = true" para futura auditoria.
- 

## 4. Blindagem contra Ataques

- Firewall Econômico + Firewall Vibracional ativam isolamento de clusters comprometidosBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
  - Estratégia **Rate Limiting**: max 100 req/min por usuário.
  - Bloqueio geográfico em ataques distribuídos detectados.
- 

## 5. Queda de Middleware de Pagamentos

- Em caso de falha no RevenueCat/Chargebee:
    - Usuário recebe status temporário "Premium Emergencial" válido por 72h.
    - Após restabelecimento, sistema valida retroativamente recibos e atualiza saldo/status.
- 



### Fluxo de Recuperação

1. **Deteção**: monitoramento identifica falha.
  2. **Failover**: tráfego desviado automaticamente.
  3. **Rollback**: banco volta ao último snapshot válido.
  4. **Reprocessamento**: logs de transações reexecutados.
  5. **Normalização**: auditoria garante consistência final.
- 



### Saída da Camada

- Economia resiliente contra falhas, ataques e corrupções.
- Zero perda de FriendCoins, doações ou status Premium.
- Continuidade operacional mesmo em cenários críticos.

## ◆ CAMADA 9 — MODELAGEM AVANÇADA DE FÓRMULAS ECONÔMICAS

*(Inflação, Estagnação, Ajustes Dinâmicos)*

### Objetivo

Garantir que a circulação de FriendCoins seja **sustentável, justa e equilibrada**, evitando colapso por inflação (excesso de moedas) ou estagnação (baixa circulação).

### Fórmula Geral da Oferta Monetária

A cada ciclo de 24h, a IA Econômica avalia:

$$\text{Oferta\_Total} = \Sigma (\text{FC\_Ganhos}) - \Sigma (\text{FC\_Queimados})$$

- **FC\_Ganhos:** todas as moedas distribuídas.
- **FC\_Queimados:** moedas retiradas via gastos, doações, boosts, upgrades.

### Fórmula de Controle de Inflação

Se a taxa de crescimento da oferta exceder o limite de 7% semanais:

$$\text{Ajuste\_Peso\_Ação} = \text{PA} * (1 - \text{Taxa\_Excedente})$$

Exemplo:

- PA (post feed) = 4.
- Inflação semanal = 10% (>7%).
- Taxa\_Excedente = 0.03.
- Novo PA =  $4 * (1 - 0.03) = 3.88 \approx 3$ .

### Fórmula de Controle de Estagnação

Se volume de transações cair abaixo de 30% da média histórica:



$$\text{Missão\_Bônus} = \text{FC\_Ganhos} * (1 + \text{Estímulo})$$

- Estímulo varia entre 0.1 e 0.3 (10% a 30%).
- Aplicado apenas em missões vibracionais e eventos coletivos → evita farming.

## Índice de Saúde Econômica (ISE)

IA calcula um score de 0 a 100 diariamente:

$$\text{ISE} = (\text{Velocidade\_Circulação} * 0.4) + (\text{Taxa\_Inflação\_Controlada} * 0.3) + (\text{Diversidade\_Transações} * 0.3)$$

- **Velocidade\_Circulação:** nº médio de transações ÷ nº de moedas circulantes.
- **Taxa\_Inflação\_Controlada:**  $100 - (\text{excedente } \%) \times 10$ .
- **Diversidade\_Transações:** nº de tipos de transações únicas por usuário.

ISE < 50 = risco de colapso → IA ativa incentivos extras.

## Ajustes Dinâmicos Automatizados

### 1. Inflação detectada:

- Redução automática de PA em posts e check-ins.
- Aumento do custo de skins/boosts temporários.

### 2. Estagnação detectada:

- Missões semanais com recompensas maiores.
- Bônus surpresa em eventos coletivos.

### 3. Clusters desequilibrados:

- IA aumenta peso de ações menos exploradas (ex: doações).

## Exemplo Prático

- Dia 1: +1.000.000 FC distribuídos, -600.000 FC gastos.
  - Oferta\_Total = 400.000 FC.
  - Crescimento semanal = 8% (>7%).
  - Ajuste aplicado: PA médio reduzido em 1%.
- 

## Segurança

- Fórmulas executadas em ambiente **IA Serverless** com logs em PostgreSQL.
  - Todos os ajustes registrados e auditáveis → visíveis em **painel econômico**.
  - Nenhum ajuste é oculto → transparência total.
- 

## Saída da Camada

- Economia autorregulada contra inflação e estagnação.
- Algoritmos claros, matemáticos e auditáveis.
- Fluxo dinâmico que preserva sustentabilidade do ecossistema.

## ◆ CAMADA 10 — SISTEMA PREMIUM: ESTRUTURA, PLANOS E BENEFÍCIOS TÉCNICOS

### Objetivo

Definir o funcionamento técnico e econômico do **Sistema Premium**, garantindo escalabilidade, rastreabilidade e benefícios diferenciados para usuários que optarem pela assinatura

DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC...

BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

---

### Estrutura de Planos

- **Mensal** → recorrência automática via middleware de pagamento.
- **Trimestral** → desconto de 10%.
- **Anual** → desconto de 20%.
- **Vitalício (Lifetime)** → número limitado de licenças, pagamento único.

Middleware recomendado: **RevenueCat** ou **Chargebee**, para abstrair complexidade da App Store/Google Play

BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

---

## Estados do Premium

- **Ativo:** usuário com acesso total aos benefícios.
- **Pendente:** aguardando confirmação de pagamento.
- **Cancelado:** cancelado pelo usuário ou falha de pagamento.
- **Expirado:** fim do período sem renovação.

Exemplo de payload:

```
{
  "user_id": "UUID",
  "status": "ativo",
  "plano": "anual",
  "expira_em": "2026-09-15"
}
```

## Benefícios Técnicos

1. **Feed Sensorial Completo** → acesso a filtros energéticos avançados e leitura vibracional expandida.
  2. **Check-in Energético Duplicado** → +100% FriendCoins em cada check-inDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....
  3. **Eventos Exclusivos** → entrada em experiências privadas.
  4. **Jogos e Realidade Aumentada Ilimitados** → sem restrição de acessos.
  5. **Descontos em Locais Parceiros** → integração automática com `partners_payments` .
  6. **Acesso a Painel Vibracional Premium** → métricas energéticas extras.
- 

## Fluxo Operacional

1. Usuário solicita upgrade → `POST /api/premium/upgrade` .

2. Middleware processa pagamento → envia status.
3. Banco `users_wallet` atualizado com flag Premium.
4. Firestore mantém cache em tempo real.
5. Logs registrados em `transactions_log` para auditoria.

## Pseudocódigo de Upgrade

```
def ativar_premium(user_id, plano):  
    recibo = revenuecat.validar_pagamento(user_id, plano)  
    if not recibo.valido:  
        return {"status": "erro", "motivo": "pagamento_invalido"}  
  
    wallet.set_status_premium(user_id, plano, recibo.expira_em)  
    transactions_log.registrar(user_id, "premium_upgrade", plano)  
    return {"status": "ok", "expira_em": recibo.expira_em}
```

## Segurança

- Tokens de pagamento assinados via HMAC.
- Receipts validados contra servidores oficiais da Apple/Google.
- Logs criptografados com AES-256.
- Acesso Premium emergencial (72h) em caso de falha no middlewareBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

## Saída da Camada

- Sistema Premium totalmente especificado.
- Integração via middleware para reduzir falhas.
- Benefícios técnicos definidos e auditáveis.
- Transparência e rastreabilidade em todo fluxo de assinatura.

## ◆ CAMADA 11 — SISTEMA DE INDICAÇÃO E BONIFICAÇÃO (REFERRAL PROGRAM)

### Objetivo

Definir o funcionamento técnico do sistema de **indicação de usuários**, criando uma mecânica justa, auditável e integrada com FriendCoins e Premium  
DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

### Estrutura do Programa

- **Regra base:** a cada **11 convites aceitos**, o usuário ganha **1 mês Premium grátis** e um bônus adicional em FriendCoinsDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....
- Convites rastreados por **códigos únicos (hash)** vinculados ao usuário.
- Indicações fraudulentas (multi-contas, farming) identificadas via IA Antifraude (grafo social).

### Fluxo Operacional

1. Usuário gera link de convite → código único (UUID + hash).
2. Novo usuário realiza cadastro → validação do código.
3. IA verifica se a relação é legítima (checagem em grafo social + IP/device).
4. Ao atingir múltiplo de 11 convites confirmados:
  - Usuário ganha 1 mês Premium (ou extensão da assinatura existente).
  - Recebe bônus de FriendCoins (configurável, ex: +50 FC).

### Estrutura de Dados

Tabela **referrals\_log (PostgreSQL):**

```
CREATE TABLE referrals_log (  
  id SERIAL PRIMARY KEY,  
  inviter_id UUID NOT NULL,  
  invited_id UUID NOT NULL,  
  status VARCHAR(20), -- pendente, aceito, inválido  
  codigo_hash VARCHAR(64),
```

```
timestamp TIMESTAMP DEFAULT NOW()
);
```

## Fórmula de Recompensa

$$\text{Bonus\_Referral} = (\text{Qtd\_Indicacoes} \div 11) * \text{Premium\_1Mês} + (\text{Qtd\_Indicacoes} * \text{FC\_Bonus})$$

- **Qtd\_Indicacoes:** nº de convites aceitos.
- **Premium\_1Mês:** extensão automática da assinatura.
- **FC\_Bonus:** FriendCoins extras (default = 5 por convite).

## Pseudocódigo

```
def processar_indicacao(inviter_id, invited_id, codigo):
    if not validar_codigo(codigo):
        return {"status": "erro", "motivo": "codigo_invalido"}

    referrals_log.inserir(inviter_id, invited_id, "aceito", codigo)
    total = referrals_log.contar_aceitos(inviter_id)

    if total % 11 == 0:
        premium.extender(inviter_id, meses=1)
        wallet.creditar(inviter_id, 50, acao="bonus_referral")

    return {"status": "ok", "convites": total}
```

## Segurança

- IA Antifraude analisa clusters: múltiplos cadastros do mesmo IP/device → bloqueados.
- Hash único por convite → impede duplicação.

- Logs imutáveis registrados em PostgreSQL + auditoria manual.
- 

## Saída da Camada

- Sistema de indicação justo, escalável e auditável.
- Bonificações automáticas em FriendCoins e Premium.
- Proteção robusta contra fraudes e farming.

## ◆ CAMADA 12 — MONETIZAÇÃO DE LOCAIS PARCEIROS (PLANOS, CONTRATOS E INTEGRAÇÕES)

### Objetivo

Definir o modelo técnico e operacional de **monetização dos Locais Parceiros**, permitindo que estabelecimentos comprem visibilidade, integração e benefícios dentro do FriendApp

DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

---

## Estrutura de Planos Comerciais

### 1. Plano Básico

- Inclusão no Mapa de Frequência.
- Painel financeiro com relatórios básicos.
- Custo mensal fixo.

### 2. Plano Avançado

- Destaque no mapa (selo vibracional).
- Integração com eventos criados no local.
- Ofertas exclusivas para usuários Premium.
- Custo mensal + % de comissão em vendas/eventos.

### 3. Plano Premium Parceiro

- Visibilidade no Feed Sensorial (postagens patrocinadas).
- Check-ins com bonificação em FriendCoins.
- Painel de métricas avançado (heatmaps econômicos).

- Contratos personalizáveis.

## Estrutura de Dados

Tabela **partners\_payments** (PostgreSQL):

```
CREATE TABLE partners_payments (  
  id SERIAL PRIMARY KEY,  
  partner_id UUID NOT NULL,  
  plano VARCHAR(50),  
  valor NUMERIC(10,2),  
  status VARCHAR(20), -- pendente, pago, falhou  
  contrato JSONB,  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

Tabela **partners\_contracts** (PostgreSQL):

```
CREATE TABLE partners_contracts (  
  id SERIAL PRIMARY KEY,  
  partner_id UUID NOT NULL,  
  inicio TIMESTAMP,  
  fim TIMESTAMP,  
  termos JSONB,  
  assinatura_hash VARCHAR(64)  
);
```

## Fluxo Operacional

1. Parceiro escolhe plano → API `/api/partner/payment`.
2. Middleware processa cobrança → status salvo em `partners_payments`.
3. Contrato digital registrado em `partners_contracts` (hash SHA-256).
4. Benefícios aplicados no ecossistema (mapa, feed, eventos).
5. Painel Admin gera relatórios financeiros em tempo real.



## Pseudocódigo

```
def ativar_plano_parceiro(partner_id, plano, contrato):
    pagamento = processar_pagamento(partner_id, plano)
    if pagamento.status != "pago":
        return {"status": "erro", "motivo": "falha_pagamento"}

    partners_contracts.inserir(partner_id, contrato)
    aplicar_beneficios(parceiro=partner_id, plano=plano)

    return {"status": "ok", "plano": plano}
```

## Segurança

- Contratos digitais com hash SHA-256 → imutáveis.
- Logs de pagamentos auditáveis (auditoria externa possível).
- Integração antifraude para evitar falsos parceiros.
- Compliance → LGPD/GDPR para dados de empresas.

## Saída da Camada

- Monetização estruturada para parceiros locais.
- Contratos digitais rastreáveis e seguros.
- Integração total com mapa, feed e eventos.
- Fluxo transparente e escalável de receita B2B.

## ◆ CAMADA 13 — MONETIZAÇÃO DE EVENTOS (GRATUITOS, PAGOS E INTEGRAÇÃO ECONÔMICA)

### Objetivo

Definir o modelo técnico e econômico para a **monetização de eventos** criados no FriendApp, permitindo tanto experiências gratuitas quanto pagas, com

integração total ao sistema de FriendCoins e pagamentos externos  
DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

---

## Tipos de Eventos

### 1. Eventos Gratuitos

- Check-ins geram FriendCoins para os participantes.
- Criador pode receber FriendCoins pelo **impacto vibracional positivo** do evento (avaliado pelos usuários).

### 2. Eventos Pagos em FriendCoins

- Ingressos adquiridos com FC.
- Parte do valor é destinado ao criador, parte ao FriendApp (taxa de serviço).

### 3. Eventos Pagos em Dinheiro Real

- Integração com gateways de pagamento via middleware.
  - FriendApp retém % configurável.
- 

## Estrutura de Dados

Tabela **events\_payments** (PostgreSQL):

```
CREATE TABLE events_payments (  
  id SERIAL PRIMARY KEY,  
  event_id UUID NOT NULL,  
  creator_id UUID NOT NULL,  
  valor NUMERIC(10,2),  
  moeda VARCHAR(10), -- FC ou BRL  
  taxa NUMERIC(5,2),  
  status VARCHAR(20), -- pendente, pago, falhou  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

Tabela **events\_feedback** (PostgreSQL):

```
CREATE TABLE events_feedback (  
  id SERIAL PRIMARY KEY,  
  event_id UUID NOT NULL,  
  user_id UUID NOT NULL,  
  score_vibracional FLOAT, -- 0 a 2  
  comentario TEXT,  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

## Fluxo Operacional

1. Usuário cria evento → define se é gratuito, pago em FC ou BRL.
2. Sistema gera contrato digital → salvo em `events_payments`.
3. Participantes se inscrevem → check-in validado por geolocalização.
4. Ao final do evento:
  - Se gratuito → criador recebe recompensa proporcional ao impacto vibracional.
  - Se pago em FC → FriendApp retém % definido (default 10%).
  - Se pago em BRL → middleware processa, FriendApp retém comissão.

## Fórmula de Recompensa por Impacto (Eventos Gratuitos)

$$\text{Recompensa\_Criador} = \Sigma (\text{Feedback\_Participantes} * \text{Peso\_Evento})$$

- **Feedback\_Participantes:** média de avaliações (0.0 – 2.0).
- **Peso\_Evento:** variável baseada no nº de check-ins válidos.

Exemplo:

- 50 participantes, média de feedback 1.6.
- $\text{Peso\_Evento} = 10$ .
- $\text{Recompensa\_Criador} = 1.6 * 50 * 10 = 800 \text{ FC}$ .

## Pseudocódigo

```
def finalizar_evento(event_id):
    feedbacks = events_feedback.media(event_id)
    participantes = contar_checkins(event_id)

    if evento.tipo == "gratuito":
        recompensa = feedbacks * participantes * evento.peso
        wallet.creditar(evento.criador, recompensa)
    elif evento.tipo == "pago_FC":
        receita = calcular_receita(event_id)
        comissao = receita * 0.10
        wallet.creditar(evento.criador, receita - comissao)
    elif evento.tipo == "pago_BRL":
        processar_pagamento_gateway(event_id)

    events_payments.atualizar_status(event_id, "pago")
```

## Segurança

- Check-in validado via **geolocalização + timestamp**.
- IA Antifraude detecta clusters de feedbacks falsos.
- Hash SHA-256 aplicado em recibos de eventos.
- Logs de auditoria em `transactions_log`.

## Saída da Camada

- Modelo econômico de eventos definido.
- Criadores recompensados por impacto real.
- Estrutura justa, auditável e escalável.
- Receita adicional para FriendApp sem comprometer autenticidade.

## ◆ CAMADA 14 — SISTEMA DE MICROTRANSAÇÕES (BOOSTS, SKINS, UPGRADES, STICKERS ENERGÉTICOS)

### Objetivo

Definir a estrutura técnica e econômica das **microtransações internas**, criando um ecossistema sustentável de personalização e upgrades, sempre alinhado ao propósito vibracional

DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

### Tipos de Microtransações

#### 1. Boosts Vibracionais

- Multiplicadores temporários de ganhos em check-ins ou interações.
- Exemplo: "x2 FriendCoins por 1h".
- Configuração: duração + intensidade.

#### 2. Skins de Realidade Aumentada (RA)

- Visualizações especiais no Mapa de Frequência e RA.
- Não afetam jogabilidade → apenas estética.

#### 3. Upgrades Econômicos

- Slots extras em missões vibracionais.
- Aumento de limite diário de check-ins válidos.

#### 4. Stickers Energéticos

- Figurinhas premium para chat e feed.
- Alguns exclusivos para usuários Premium.

### Estrutura de Dados

Tabela **microtransactions\_log** (PostgreSQL):

```
CREATE TABLE microtransactions_log (  
  id SERIAL PRIMARY KEY,  
  user_id UUID NOT NULL,  
  item VARCHAR(50),  
  tipo VARCHAR(20), -- boost, skin, upgrade, sticker
```

```
custo INT,  
moeda VARCHAR(10), -- FC ou BRL  
status VARCHAR(20), -- pendente, concluido, falhou  
timestamp TIMESTAMP DEFAULT NOW()  
);
```

## Fluxo Operacional

1. Usuário acessa loja interna → escolhe item.
2. API `/api/wallet/transaction` → verifica saldo.
3. Se suficiente → transação debitada → item liberado.
4. Se compra em BRL → middleware processa → libera item após confirmação.
5. Registro feito em `microtransactions_log`.

## Fórmulas de Consumo

- **Preço Dinâmico:** IA ajusta custo conforme inflação:

$$\text{Custo\_Item} = \text{Preço\_Base} * (1 + \text{Inflação\_Semanal})$$

- **Validade de Boost:**

$$\text{Boost\_Ativo} = (\text{Agora} < \text{Timestamp\_Compra} + \text{Duração})$$

## Pseudocódigo

```
def comprar_item(user_id, item, custo, moeda):  
    if wallet.saldo(user_id, moeda) < custo:  
        return {"status": "erro", "motivo": "saldo_insuficiente"}  
  
    wallet.debitar(user_id, custo, moeda)  
    microtransactions_log.inserir(user_id, item, custo, moeda, "concluido")
```

```
liberar_item(user_id, item)

return {"status": "ok", "item": item}
```

## Segurança

- Transações assinadas digitalmente (HMAC).
- Itens vinculados ao user\_id → não transferíveis (evita mercado paralelo).
- Logs imutáveis → auditoria completa.
- IA Antifraude detecta compras anômalas (ex: padrão de bots).

## Saída da Camada

- Sistema de microtransações sólido, auditável e justo.
- Personalização para engajamento sem criar pay-to-win.
- Fluxo transparente de receita adicional.

## ◆ CAMADA 15 — SISTEMA DE DOAÇÕES E IMPACTO SOCIAL (DINHEIRO REAL + FRIENDCOINS)

### Objetivo

Implementar um **sistema de doações rastreável, transparente e seguro**, permitindo que usuários contribuam com **causas sociais** dentro do FriendApp usando **FriendCoins (FC)** ou **moeda real (BRL/USD)**

DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

## Estrutura de Doações

### 1. Doações em FriendCoins (FC)

- Dedução direta da wallet do usuário.
- Valor convertido em “Impacto Social” dentro do app.

### 2. Doações em Dinheiro Real

- Processadas via middleware de pagamentos (RevenueCat/Chargebee ou gateway externo).
- Associadas a projetos sociais aprovados.

### 3. Transparência Total

- Todas as doações são **públicas e rastreáveis** em um painel acessível.
- Usuário pode escolher anonimato (nome oculto, mas transação registrada).

---

## Estrutura de Dados

### Tabela **donations\_log** (PostgreSQL):

```
CREATE TABLE donations_log (  
  id SERIAL PRIMARY KEY,  
  user_id UUID NOT NULL,  
  destino VARCHAR(100),  
  valor INT,  
  moeda VARCHAR(10), -- FC ou BRL  
  auditoria_hash VARCHAR(64),  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

### Tabela **donations\_projects** (PostgreSQL):

```
CREATE TABLE donations_projects (  
  id SERIAL PRIMARY KEY,  
  nome VARCHAR(100),  
  descricao TEXT,  
  status VARCHAR(20), -- ativo, concluido, suspenso  
  arrecadado INT,  
  meta INT,  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```



## Fluxo Operacional

1. Usuário seleciona projeto → escolhe valor + moeda.
2. API `/api/donation/send` processa transação.
3. Sistema gera `auditoria_hash` (SHA-256).
4. Valor atualizado em `donations_log` e `donations_projects`.
5. Painel de Transparência mostra progresso da meta.

## Fórmula de Impacto

Cada doação gera um score de impacto para o usuário:

$$\text{Impacto\_Social} = \text{Valor\_Doadado} * \text{Fator\_Projeto}$$

- **Fator\_Projeto:** peso (0,5 a 2,0) definido pela relevância social.
- Exemplo:
  - Doação: 100 FC
  - Fator\_Projeto: 1,5
  - Impacto\_Social = 150 pontos

Esses pontos podem ser exibidos em **perfil vibracional** como indicador de contribuição social.

## Pseudocódigo

```
def doar(user_id, destino, valor, moeda):  
    if moeda == "FC":  
        if wallet.saldo(user_id) < valor:  
            return {"status": "erro", "motivo": "saldo_insuficiente"}  
        wallet.debitar(user_id, valor)  
    else:  
        pagamento = processar_gateway(user_id, valor, moeda)  
        if pagamento.status != "pago":  
            return {"status": "erro", "motivo": "falha_pagamento"}
```

```
hash_auditoria = gerar_hash(user_id, destino, valor, moeda)
donations_log.inserir(user_id, destino, valor, moeda, hash_auditoria)
atualizar_meta_projeto(destino, valor)

return {"status": "ok", "hash": hash_auditoria}
```

## Segurança

- Hash SHA-256 em cada doação.
- Painel público auditável → garante transparência.
- Logs replicados em multi-region.
- Compliance com LGPD/GDPR.

## Saída da Camada

- Sistema de doações robusto e rastreável.
- Transparência absoluta para usuários e investidores.
- Integração direta com perfil vibracional (Impacto Social).

## ◆ CAMADA 16 — ONBOARDING ECONÔMICO PROGRESSIVO (EDUCAÇÃO FINANCEIRA + UX)

### Objetivo

Introduzir o usuário ao **Sistema Econômico do FriendApp** de forma gradual e educativa, evitando sobrecarga cognitiva e garantindo **adesão natural** à economia vibracional

DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

## Princípios de Design

1. **Progressividade** → recursos econômicos liberados em etapas, conforme engajamento.
2. **Educação Implícita** → dicas contextuais em vez de tutoriais extensos.

3. **Gamificação Consciente** → recompensas surpresa e narrativas vibracionais, sem incentivar vício.
  4. **Transparência** → usuário sempre entende para onde suas moedas fluem.
- 

## Estrutura de Etapas do Onboarding

### 1. Etapa 1 — Introdução Básica

- Usuário aprende o que são FriendCoins.
- Ganha suas primeiras moedas com check-in inicial.
- Interface mostra apenas saldo e usos básicos (stickers, boosts).

### 2. Etapa 2 — Engajamento

- Após 3 dias de uso ou 5 interações, desbloqueia missões vibracionais.
- Tutorial rápido mostra como ganhar moedas de impacto.

### 3. Etapa 3 — Integração

- Usuário é introduzido ao Painel de Economia Vibracional.
- Dicas contextuais:

“Organizar Boras bem avaliados aumenta sua energia e gera mais FriendCoins.”

### 4. Etapa 4 — Expansão

- Premium apresentado como evolução natural → não forçado.
- Locais Parceiros e Eventos aparecem com descontos exclusivos.

### 5. Etapa 5 — Plenitude

- Usuário tem acesso completo ao ecossistema econômico.
  - Painel mostra também contribuições sociais (doações).
- 

## Estrutura de Dados

Tabela **user\_onboarding** (PostgreSQL):

```
CREATE TABLE user_onboarding (  
  user_id UUID PRIMARY KEY,  
  etapa INT, -- 1 a 5
```

```
concluido BOOLEAN DEFAULT false,  
desbloqueios JSONB,  
ultima_acao TIMESTAMP DEFAULT NOW()  
);
```

## Fluxo Operacional

1. Usuário cria conta → entra na Etapa 1.
2. Cada ação relevante atualiza status em `user_onboarding`.
3. IA Aurah Kosmos envia **dicas personalizadas** conforme etapa.
4. Ao concluir todas as etapas, flag `concluido = true`.

## Pseudocódigo

```
def atualizar_onboarding(user_id, acao):  
    etapa = user_onboarding.get_etapa(user_id)  
  
    if etapa == 1 and acao == "checkin_inicial":  
        user_onboarding.set_etapa(user_id, 2)  
    elif etapa == 2 and acao in ["missao_completada", "post_feed"]:  
        user_onboarding.set_etapa(user_id, 3)  
    elif etapa == 3 and acao == "evento_participado":  
        user_onboarding.set_etapa(user_id, 4)  
    elif etapa == 4 and acao == "premium_upgrade":  
        user_onboarding.set_etapa(user_id, 5)
```

## Segurança

- Onboarding não pode ser burlado (IA valida interações autênticas).
- Logs de cada etapa salvos em `transactions_log` com flag `onboarding`.
- Protege contra contas criadas apenas para farm de moedas.

## Saída da Camada

- Introdução suave e educativa ao sistema econômico.
- Evita sobrecarga e incentiva engajamento genuíno.
- Estrutura progressiva que conecta o usuário à **Economia da Consciência**.

## ◆ CAMADA 17 — PAINEL SENSORIAL ECONÔMICO (EDUCAÇÃO CONTÍNUA E FEEDBACK CONTEXTUAL)

### Objetivo

Criar um **painel interativo e sensorial** que eduque o usuário continuamente sobre a economia do FriendApp, fornecendo **feedback contextual** sobre suas ações e incentivando escolhas conscientes

DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC...

BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

### Funcionalidades do Painel

#### 1. Visão Geral Econômica

- Saldo atual de FriendCoins.
- Histórico de ganhos e gastos.
- Status Premium.

#### 2. Dicas Contextuais (Educação Contínua)

- IA Aurah Kosmos gera mensagens adaptativas:

"Seus eventos com alta avaliação vibracional multiplicam sua energia e geram mais FriendCoins."

"Você expandiu sua rede! Conexões autênticas rendem mais que repetições."

#### 3. Indicadores de Impacto

- Gráfico de contribuição vibracional (feedbacks positivos, harmonia das conexões).
- Exibição de **Impacto Social** (doações realizadas).

#### 4. Alertas Econômicos

- Notificação de inflação controlada:

"Moedas circulando em alta, boosts temporários estão mais valiosos."

- Aviso de estagnação:

"Pouca circulação detectada. Missões semanais oferecem +30% de bônus."

## 5. Recompensas Surpresa

- Interface mostra quando a IA concede bônus inesperados.
- Exemplo: animação com efeitos vibracionais ao liberar +50 FC.

## Estrutura de Dados

Tabela **economy\_tips\_log** (PostgreSQL):

```
CREATE TABLE economy_tips_log (  
  id SERIAL PRIMARY KEY,  
  user_id UUID NOT NULL,  
  mensagem TEXT,  
  tipo VARCHAR(20), -- dica, alerta, bonus  
  gerado_por VARCHAR(50), -- aurah_kosmos, antifraude, sistema  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

## Fluxo Operacional

1. Usuário realiza ação econômica (ex: evento, doação, check-in).
2. IA Aurah Kosmos processa impacto e decide se gera dica, alerta ou recompensa surpresa.
3. Registro gravado em `economy_tips_log`.
4. Painel exibe feedback em tempo real.

## Pseudocódigo

```
def gerar_feedback_economico(user_id, acao, contexto):
    indice = aurah_kosmos.calcular_indice(contexto)

    if indice > 1.5 and acao == "evento":
        mensagem = "Seu evento gerou alta harmonia! Continue criando experiências."
        economy_tips_log.inserir(user_id, mensagem, "dica", "aurah_kosmos")
    elif acao == "doacao":
        mensagem = "Sua doação fortalece o impacto social da comunidade."
        economy_tips_log.inserir(user_id, mensagem, "bonus", "sistema")

    return mensagem
```

## Segurança

- Logs imutáveis garantem que mensagens não sejam manipuladas.
- IA antifraude verifica se recompensas surpresa não são exploradas.
- Feedback sempre vinculado ao histórico de ações reais do usuário.

## Saída da Camada

- Painel sensorial cria **educação contínua** sem sobrecarga.
- Usuário entende o valor de suas ações na economia vibracional.
- Sistema preserva transparência, justiça e magia das recompensas.

# ◆ CAMADA 18 — INTEGRAÇÃO ECONÔMICA COM LOCAIS PARCEIROS, EVENTOS, FEED, CHAT E RA

## Objetivo

Definir como o sistema econômico se conecta de forma **orgânica e técnica** com os demais módulos centrais do FriendApp (Locais Parceiros, Eventos, Feed Sensorial, Chat Vibracional e Realidade Aumentada), garantindo **circulação fluida de FriendCoins** em todo ecossistema

## Integrações por Subsistema

### 1. Locais Parceiros

- Check-in gera FriendCoins extras para usuários Premium.
- Locais pagam planos → monetização registrada em `partners_payments`.
- Descontos aplicados automaticamente via `/api/partner/payment`.

### 2. Eventos

- Participação gera ganhos em FC (eventos gratuitos).
- Ingressos podem ser pagos em FC ou BRL (eventos pagos).
- Criadores recebem bônus proporcional ao **impacto vibracional**.

### 3. Feed Sensorial

- Posts no feed podem gerar ganhos em FC proporcional ao impacto (índice de engajamento vibracional).
- Boosts e posts patrocinados pagos em FC → visibilidade ampliada.
- Stickers energéticos premium podem ser adquiridos via microtransações.

### 4. Chat Vibracional

- Stickers especiais desbloqueados via FriendCoins.
- IA Advisor Econômico (Aurah Kosmos) sugere onde investir ou gastar moedas.
- Transações de FC podem ser feitas como "presentes vibracionais".

### 5. Realidade Aumentada (RA)

- Skins, filtros e camadas visuais adquiridas via microtransações.
- RA mostra "hotspots econômicos" (locais mais ativos em FC).
- Eventos e parceiros destacados em RA com benefícios econômicos.

---

## Estrutura de Dados



### Tabela **economy\_integration\_log** (PostgreSQL):

```
CREATE TABLE economy_integration_log (  
  id SERIAL PRIMARY KEY,  
  user_id UUID,  
  modulo VARCHAR(50), -- feed, evento, chat, parceiro, ra  
  acao VARCHAR(50),  
  valor INT,  
  moeda VARCHAR(10), -- FC ou BRL  
  impacto FLOAT,  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

### Fluxo Operacional

1. Usuário realiza ação em qualquer módulo.
2. API de integração chama `/api/wallet/transaction`.
3. Valor registrado em `economy_integration_log`.
4. IA Aurah Kosmos atualiza índices de impacto → pode liberar bônus surpresa.
5. Painel Admin mostra distribuição de moedas por módulo.

### Pseudocódigo

```
def registrar_integracao(user_id, modulo, acao, valor, moeda, impacto):  
  wallet.transaction(user_id, valor, acao, moeda)  
  economy_integration_log.inserir(user_id, modulo, acao, valor, moeda, im  
  pacto)  
  if impacto > 1.5:  
    aurah_kosmos.enviar_bonus(user_id, valor * 0.2)
```

### Segurança

- Todas as integrações autenticadas com OAuth 2.0 + JWT.

- Logs imutáveis → auditoria possível em qualquer módulo.
  - IA Antifraude monitora **clusters suspeitos** (ex: mesmo grupo sempre interagindo).
- 

## Saída da Camada

- Economia totalmente interligada ao ecossistema FriendApp.
- Cada módulo gera valor vibracional e econômico real.
- Transparência e rastreabilidade em todas as integrações.

## ◆ CAMADA 19 — SISTEMA DE LOGS ECONÔMICOS E AUDITORIA INTERNA

### Objetivo

Garantir **rastreabilidade absoluta** de todas as operações financeiras e vibracionais do FriendApp, permitindo **auditoria interna e externa**, além de prevenir inconsistências ou fraudes

BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

---

### Tipos de Logs

#### 1. Logs de Transações

- Ganho, gasto, transferência, microtransações, upgrades.
- Campos: user\_id, ação, valor, moeda, impacto, hash de segurança.

#### 2. Logs de Premium

- Ativações, renovações, cancelamentos.
- Origem validada via middleware (RevenueCat/Chargebee).

#### 3. Logs de Parceiros

- Pagamentos de contratos e planos.
- Vinculados a contratos digitais com hash SHA-256.

#### 4. Logs de Eventos

- Ingressos pagos em FC/BRL.

- Feedback vibracional e recompensas ao criador.

## 5. Logs de Doações

- Destino, valor, moeda, auditoria\_hash.
- Transparência pública garantida.

## Estrutura de Dados

Tabela **economy\_logs** (PostgreSQL):

```
CREATE TABLE economy_logs (  
  id SERIAL PRIMARY KEY,  
  user_id UUID,  
  tipo VARCHAR(30), -- transacao, premium, parceiro, evento, doacao  
  acao VARCHAR(50),  
  valor NUMERIC(10,2),  
  moeda VARCHAR(10), -- FC ou BRL  
  impacto FLOAT,  
  hash_auditoria VARCHAR(64),  
  origem VARCHAR(50), -- feed, evento, chat, parceiro  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

## Fluxo Operacional

1. Toda API econômica gera entrada em `economy_logs`.
2. Logs replicados em storage multi-region (backup redundante).
3. Auditoria interna executa checagens semanais:
  - Consistência entre Firestore ↔ PostgreSQL.
  - Comparação entre contratos e pagamentos reais.
4. Logs com suspeita de fraude enviados para **IA Antifraude**.

## Pseudocódigo

```
def registrar_log(user_id, tipo, acao, valor, moeda, impacto, origem):
    hash_auditoria = gerar_hash(user_id, tipo, acao, valor, moeda, impacto, o
rigem)
    economy_logs.inserir(user_id, tipo, acao, valor, moeda, impacto, hash_au
ditoria, origem)
    return {"status": "ok", "hash": hash_auditoria}
```

## Segurança

- Hash SHA-256 em todos os registros → imutabilidade.
- Criptografia AES-256 em trânsito e repouso.
- Logs com validade legal → exportáveis em PDF/CSV para auditorias externas.
- Compliance total com **LGPD/GDPR**.

## Saída da Camada

- Sistema econômico **100% rastreável e auditável**.
- Transparência e governança operacional garantidas.
- Blindagem contra fraudes ou manipulações internas.

# ◆ CAMADA 20 — SISTEMA DE CONTROLE DE INFLAÇÃO E ESTAGNAÇÃO ECONÔMICA (MECÂNICAS REGULATÓRIAS)

## Objetivo

Garantir a **sustentabilidade econômica de longo prazo**, prevenindo riscos de **inflação descontrolada** (excesso de FriendCoins) e **estagnação** (baixa circulação de moedas).

## Variáveis de Monitoramento

- **Oferta\_Total (OT)**: soma de todas as moedas circulantes.

- **Velocidade\_Circulação (VC):** nº médio de transações ÷ nº de moedas em circulação.
- **Inflação\_Semanal (IS):** % de crescimento da oferta em 7 dias.
- **Taxa\_Estagnação (TE):** queda percentual no volume de transações em relação à média histórica.

## Fórmulas de Regulação

### 1. Ajuste Anti-Inflação

Se  $IS > 7\%$ :

$$PA = PA * (1 - (IS - 0.07))$$

- PA = Peso das ações (posts, check-ins, etc.).
- Reduz ganhos unitários para desacelerar emissão.

### 2. Ajuste Anti-Estagnação

Se  $TE > 30\%$ :

$$\text{Bonus\_Missões} = \text{Bonus\_Missões} * (1 + 0.2)$$

- Missões semanais aumentam em até 20% de recompensa.
- Estimula circulação.

### 3. Índice de Saúde Econômica (ISE)

$$ISE = (VC * 0.4) + ((1 - IS) * 0.3) + ((1 - TE) * 0.3)$$

- Score 0–100.
- $ISE < 50 \rightarrow$  IA ativa incentivos extras (ex: boosts gratuitos).

## Fluxo Operacional

1. Sistema coleta métricas diariamente.
2. IA Econômica aplica fórmulas de regulação.

3. Ajustes propagados automaticamente em:

- Peso de ações.
- Custos de itens de microtransação.
- Bônus de missões/eventos.

4. Logs de ajustes salvos em `economy_logs` com tag `ajuste_dinamico`.

## Pseudocódigo

```
def controle_economico():  
    inflacao = calcular_inflacao()  
    estagnacao = calcular_estagnacao()  
    ise = calcular_ISE(inflacao, estagnacao)  
  
    if inflacao > 0.07:  
        ajustar_pesos(1 - (inflacao - 0.07))  
  
    if estagnacao > 0.3:  
        aumentar_bonus_missoes(0.2)  
  
    registrar_log("ajuste_dinamico", inflacao, estagnacao, ise)
```

## Segurança

- Ajustes automáticos validados por IA + logs auditáveis.
- Nenhum ajuste oculto → transparência no painel econômico.
- Painel Admin exibe histórico de inflação, estagnação e ações corretivas.

## Saída da Camada

- Sistema econômico **autorregulado** contra extremos.
- Prevenção de inflação e estímulo contra estagnação.
- Sustentabilidade e confiança para usuários e investidores.

## ◆ CAMADA 21 — MIDDLEWARE DE PAGAMENTOS (REVENUECAT/CHARGE BEE) E INTEGRAÇÃO COM APP STORES

### 🎯 Objetivo

Simplificar a **gestão de pagamentos e assinaturas Premium**, evitando complexidade direta com APIs da Apple App Store e Google Play, utilizando um **middleware robusto (RevenueCat ou Chargebee)**

BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

### 🧩 Funções do Middleware

#### 1. Gestão de Assinaturas

- Criação, renovação, cancelamento, expiração.
- Suporte a múltiplos planos (mensal, anual, lifetime).

#### 2. Gestão de Compras Únicas

- Packs de FriendCoins.
- Microtransações (skins, boosts, stickers).

#### 3. Integração Multi-Plataforma

- SDK único → abstrai Apple e Google.
- Sincronização automática de recibos.

#### 4. Webhooks e Logs

- Notificações em tempo real sobre mudanças de status.
- Registro de transações em `transactions_log`.

### 📐 Estrutura de Dados

Tabela **payments\_log** (PostgreSQL):

```
CREATE TABLE payments_log (  
  id SERIAL PRIMARY KEY,  
  user_id UUID NOT NULL,  
  tipo VARCHAR(20), -- assinatura, compra_unica  
  plano VARCHAR(50),  
  valor NUMERIC(10,2),
```

```
moeda VARCHAR(10), -- BRL, USD
status VARCHAR(20), -- ativo, cancelado, expirado, pendente
recibo_id VARCHAR(100),
provedor VARCHAR(50), -- revenuecat, chargebee
timestamp TIMESTAMP DEFAULT NOW()
);
```

## Fluxo Operacional

1. Usuário solicita upgrade ou compra.
2. SDK do middleware processa pagamento.
3. Middleware valida recibo com App Store/Google Play.
4. Status retornado → atualizado em `payments_log`.
5. Middleware dispara **webhook** → backend do FriendApp aplica benefícios (Premium, FCs).

## Pseudocódigo

```
def processar_pagamento(user_id, tipo, plano, valor, moeda):
    recibo = revenuecat.processar(user_id, plano, valor, moeda)

    if recibo.status == "valido":
        payments_log.inserir(user_id, tipo, plano, valor, moeda, "ativo", recibo.
id, "revenuecat")
        aplicar_beneficios(user_id, tipo, plano)
        return {"status": "ok", "recibo": recibo.id}
    else:
        payments_log.inserir(user_id, tipo, plano, valor, moeda, "falhou", recibo.id, "revenuecat")
        return {"status": "erro", "motivo": "pagamento_invalido"}
```

## Segurança

- Tokens assinados via HMAC para comunicação middleware ↔ backend.



- Recibos validados diretamente nos servidores das app stores.
  - Failover: em caso de queda do middleware → status temporário de **Premium Emergencial (72h)** até revalidaçãoBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
- 

## Saída da Camada

- Sistema de pagamentos simplificado e robusto.
- Redução de riscos e bugs em assinaturas.
- Transparência e rastreabilidade total via logs.
- Escalabilidade para múltiplos países e moedas.

## ◆ CAMADA 22 — GAMIFICAÇÃO CONSCIENTE (RECOMPENSAS SURPRESA + IMPACTO COLETIVO)

### Objetivo

Aplicar **mecânicas de gamificação alinhadas ao propósito vibracional**, sem gerar vício ou "grinding", reforçando a **autenticidade das conexões** e incentivando o impacto positivo coletivo  
DOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

---

### Princípios de Gamificação Consciente

1. **Surpresa e Magia** → recompensas inesperadas concedidas pela IA Aurah Kosmos.
  2. **Impacto Coletivo** → conquistas compartilhadas que incentivam colaboração.
  3. **Justiça e Transparência** → sem mecânicas pay-to-win, sempre rastreável.
  4. **Equilíbrio** → mecânica desenhada para evitar repetição mecânica excessiva.
- 

### Mecânicas de Recompensas

1. **Recompensas Surpresa Individuais**
  - IA avalia interações de alto impacto vibracional.

- Usuário recebe bônus de FriendCoins ou benefícios extras.
- Exemplo:

"Sua conversa com [Nome] atingiu harmonia rara. O universo reconhece: +50 FC."

Fórmula:

$$\text{Bônus\_Surpresa} = \text{Valor\_Base} * \text{Índice\_Impacto} * \text{Fator\_Sorte}$$

- Valor\_Base = 10 FC (ajustável).
- Índice\_Impacto (0–2).
- Fator\_Sorte (0.5–2.0).

## 1. Missões Coletivas Vibracionais

- Grupo de usuários pode completar desafios em conjunto.
- Exemplo: "100 check-ins em locais parceiros esta semana".
- Se meta for atingida → todos os participantes recebem FriendCoins extras.

Fórmula:

$$\text{Recompensa\_Grupo} = \Sigma(\text{Ações\_Grupo}) * \text{Multiplicador\_Grupo}$$

- Multiplicador\_Grupo varia entre 1.2 e 1.5.

## 1. Eventos Globais de Energia

- Semana temática → todos os usuários ganham boosts.
- Exemplo: "Semana da Conexão Autêntica" → +20% em ganhos de eventos.

## Estrutura de Dados

Tabela **gamification\_log** (PostgreSQL):

```
CREATE TABLE gamification_log (  
  id SERIAL PRIMARY KEY,  
  user_id UUID,  
  tipo VARCHAR(30), -- surpresa, missao_coletiva, evento_global  
  descricao TEXT,  
  valor INT,  
  impacto FLOAT,  
  origem VARCHAR(50),  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

## Fluxo Operacional

1. IA Aurah Kosmos analisa interações em tempo real.
2. Quando detecta impacto elevado → gera recompensa surpresa.
3. Missões coletivas monitoradas por contadores globais.
4. Eventos globais definidos via painel admin → aplicados a todos.
5. Registros inseridos em `gamification_log`.

## Pseudocódigo

```
def avaliar_interacao(user_id, acao, impacto):  
  if impacto > 1.7:  
    bonus = calcular_bonus_surpresa(impacto)  
    wallet.creditar(user_id, bonus)  
    gamification_log.inserir(user_id, "surpresa", acao, bonus, impacto, "aurah_kosmos")
```

## Segurança

- Recompensas surpresa não previsíveis → antifraude evita exploração.
- Logs imutáveis para cada bônus concedido.

- Auditoria em missões coletivas → participação real validada via check-ins e RA.

---

## Saída da Camada

- Gamificação que mantém a **magia do inesperado**.
- Incentivo a impacto coletivo e colaborações reais.
- Preservação do propósito vibracional sem gerar vício.

## ◆ CAMADA 23 — SISTEMA DE RESTRIÇÕES E PENALIDADES ECONÔMICAS

*(Fraudes, Abusos e Comportamentos Anti-Conscientes)*

### Objetivo

Criar um sistema robusto de **penalidades econômicas**, aplicando restrições automáticas em casos de fraude, abuso ou comportamento contrário ao propósito vibracional, protegendo a integridade do ecossistema BLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....

---

### Tipos de Comportamentos Penalizáveis

1. **Farming de Moedas** → criação de interações artificiais para gerar FriendCoins.
2. **Uso de Bots ou Scripts** → automação para check-ins/posts.
3. **Clusters Fechados** → grupos que só interagem entre si para inflar ganhos.
4. **Fraudes em Doações** → tentativas de manipulação de impacto social.
5. **Receipts Premium Falsificados** → upgrades ilegítimos.
6. **Conteúdo Anti-Consciente** → posts ou eventos que violem diretrizes vibracionais.

---

### Penalidades Econômicas

1. **Redução de Ganhos Temporária**
  - Fórmula:

$$\text{Ganho\_Final} = \text{Ganho\_Base} * (1 - \text{Penalidade\%})$$

- Penalidade% varia entre 30% e 100%.

## 2. Bloqueio de Recompensas

- Usuário continua ativo no app, mas não recebe FriendCoins até revisão.

## 3. Multa Econômica em FriendCoins

- Débito direto da wallet.
- Valor proporcional ao ganho irregular identificado.

## 4. Suspensão Premium

- Benefícios temporariamente desativados até validação.

## 5. Banimento Econômico Definitivo

- Em casos graves → wallet congelada, impossibilidade de transações.

## Estrutura de Dados

Tabela **penalties\_log** (PostgreSQL):

```
CREATE TABLE penalties_log (  
  id SERIAL PRIMARY KEY,  
  user_id UUID NOT NULL,  
  tipo VARCHAR(30), -- reducao, bloqueio, multa, suspensao, banimento  
  motivo TEXT,  
  valor INT,  
  duracao INTERVAL,  
  status VARCHAR(20), -- ativo, expirado, revisao  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

## Fluxo Operacional

1. IA Antifraude identifica comportamento suspeito.
2. Evento registrado em `penalties_log`.

3. Penalidade aplicada automaticamente (ou revisão manual para casos graves).
4. Usuário notificado com mensagem clara → ex:  

"Detectamos comportamento suspeito em suas interações. Suas recompensas foram temporariamente reduzidas."
5. Penalidade expira automaticamente ou após revisão humana.

## Pseudocódigo

```
def aplicar_penalidade(user_id, tipo, motivo, valor=0, duracao=None):
    penalties_log.inserir(user_id, tipo, motivo, valor, duracao, "ativo")

    if tipo == "reducao":
        wallet.aplicar_modificador(user_id, -0.5) # -50% ganhos
    elif tipo == "multa":
        wallet.debitar(user_id, valor)
    elif tipo == "suspensao":
        premium.suspender(user_id, duracao)
    elif tipo == "banimento":
        wallet.congelar(user_id)
```

## Segurança

- Todas as penalidades auditáveis via `penalties_log`.
- IA Antifraude + revisão manual garantem justiça.
- Logs protegidos por hash SHA-256.
- Notificações enviadas ao usuário → transparência total.

## Saída da Camada

- Economia protegida contra fraudes e abusos.
- Sistema justo, auditável e proporcional às infrações.
- Preservação da integridade vibracional e confiança dos usuários.

## ◆ CAMADA 24 — RELATÓRIOS ECONÔMICOS PARA INVESTIDORES, PARCEIROS E ADMINISTRAÇÃO INTERNA

### Objetivo

Fornecer **relatórios econômicos claros, técnicos e auditáveis** para três públicos distintos:

- **Investidores** → visão estratégica de crescimento e sustentabilidade.
  - **Parceiros Comerciais** → métricas de performance e retorno.
  - **Administração Interna** → governança, auditoria e ajustes econômicos.
- 

### Tipos de Relatórios

#### 1. Relatórios para Investidores

- Volume total de FriendCoins emitidas e circulantes.
- Receita recorrente (Premium + Locais Parceiros + Eventos).
- Taxa de crescimento da economia (mês a mês).
- Índice de Saúde Econômica (ISE).
- Distribuição de gastos dos usuários (microtransações, eventos, Premium).

#### 2. Relatórios para Parceiros

- Nº de check-ins em seus locais.
- Receita gerada via FriendCoins e BRL.
- Taxa de conversão de eventos pagos.
- Feedback vibracional médio dos clientes.

#### 3. Relatórios Internos (Administração)

- Logs de transações (resumidos e agregados).
  - Análise de clusters suspeitos (fraude/farming).
  - Projeções de inflação/estagnação.
  - Relatórios de impacto social (doações).
-

## Estrutura de Dados

Tabela **reports\_generated** (PostgreSQL):

```
CREATE TABLE reports_generated (  
  id SERIAL PRIMARY KEY,  
  tipo VARCHAR(30), -- investidor, parceiro, interno  
  destino VARCHAR(100), -- email ou painel  
  conteudo JSONB,  
  gerado_por VARCHAR(50), -- sistema, admin  
  timestamp TIMESTAMP DEFAULT NOW()  
);
```

## Fluxo Operacional

1. Scheduler gera relatórios automáticos (diário, semanal, mensal).
2. Dados consolidados de **transactions\_log**, **partners\_payments**, **donations\_log**, **penalties\_log**.
3. Relatório formatado em JSON/PDF.
4. Envio:
  - **Investidores** → portal seguro + export PDF.
  - **Parceiros** → painel exclusivo.
  - **Administração** → dashboards internos (Grafana/Kibana).

## Exemplo de Relatório (Investidor)

```
{  
  "periodo": "Agosto 2025",  
  "fc_emitidos": 1200000,  
  "fc_circulantes": 450000,  
  "receita_total_BRL": 98500.75,  
  "assinaturas_ativas": 1240,  
  "ise": 82,  
  "doacoes_realizadas": 32000
```



```
}
```

## Pseudocódigo

```
def gerar_relatorio(tipo, destino):  
    dados = consolidar_dados(tipo)  
    relatorio = formatar_relatorio(dados, formato="json")  
    reports_generated.inserir(tipo, destino, relatorio, "sistema")  
    enviar_relatorio(destino, relatorio)  
    return {"status": "ok", "tipo": tipo}
```

## Segurança

- Relatórios assinados digitalmente (HMAC + hash SHA-256).
- Acesso segmentado → investidores ≠ parceiros ≠ administração.
- Logs de geração salvos para compliance.
- Compliance com LGPD/GDPR para dados de usuários.

## Saída da Camada

- Relatórios claros, confiáveis e personalizáveis.
- Suporte à tomada de decisão de investidores e administração.
- Transparência para parceiros → fortalece credibilidade do ecossistema.

## ◆ CAMADA 25 — CONCLUSÃO ESTRATÉGICA E VISÃO DE SUSTENTABILIDADE ECONÔMICA

### Objetivo

Encerrar a arquitetura técnica do **Sistema Econômico do FriendApp** mostrando sua visão de longo prazo, a sustentabilidade do ecossistema e sua diferenciação em relação a modelos de mercado tradicionais.

## Princípios Consolidados

### 1. Economia da Consciência

- FriendCoins não são apenas moeda digital → representam energia circulante.
- Valor não vem do consumo vazio, mas do **impacto autêntico** das interaçõesDOCUMENTO\_FUNCIONAL\_\_SISTEMA\_EC....

### 2. Diversificação de Receita

- Premium (assinaturas).
- Locais Parceiros (planos comerciais).
- Eventos (ingressos pagos em FC/BRL).
- Microtransações (skins, boosts, stickers).
- Doações (impacto social real).

### 3. Segurança e Justiça

- Blindagem antifraude multi-camadasBLUEPRINT\_TCNICO\_\_SISTEMA\_ECONM....
- Controle de inflação/estagnação dinâmico.
- Penalidades para abusos e farming.

### 4. Escalabilidade Global

- Multi-cloud, multi-region, failover econômico.
- Middleware de pagamentos integrado (RevenueCat/Chargebee).
- Estrutura pronta para expansão internacional.

---

## Visão de Sustentabilidade

- **Equilíbrio Monetário:** algoritmos mantêm a economia estável, sem hiperinflação ou estagnação.
- **Expansão Natural:** onboarding progressivo educa o usuário, aumentando retenção.
- **Impacto Social:** doações transparentes fortalecem a imagem ética do ecossistema.

- **Valor para Parceiros:** relatórios e métricas tornam o FriendApp atrativo para empresas.
- **Investidores:** relatórios econômicos claros garantem confiança e visão de retorno.

## Métrica de Sustentabilidade (MS)

Indicador final calculado mensalmente para medir saúde econômica do FriendApp:

$$\begin{aligned} MS = & (\text{Receita\_Real} / \text{Meta\_Receita}) * 0.4 + \\ & (\text{ISE} / 100) * 0.3 + \\ & (\text{Taxa\_Usuarios\_Ativos} / \text{Total\_Usuarios}) * 0.3 \end{aligned}$$

- **Receita\_Real:** total em BRL/USD no período.
- **ISE:** Índice de Saúde Econômica (Camada 9 e 20).
- **Taxa\_Usuarios\_Ativos:** % de usuários ativos vibrando dentro da economia.

Exemplo:

- Receita\_Real = 90% da meta.
- ISE = 82.
- Ativos = 70%.

$$\begin{aligned} MS &= (0.9 * 0.4) + (0.82 * 0.3) + (0.7 * 0.3) \\ MS &= 0.36 + 0.246 + 0.21 = 0.816 \rightarrow 81,6\% \end{aligned}$$

Resultado: economia **saudável e sustentável**.

## Saída da Camada

- O FriendApp não é apenas um app → é um **ecossistema econômico vibracional sustentável**.
- Moeda, Premium, parceiros, eventos e doações conectam-se em **fluxo regenerativo**.

- Estrutura pronta para expansão global, com base ética e tecnológica sólida.

## ◆ SEÇÃO FINAL — INTEGRAÇÕES E DEPENDÊNCIAS CÍCLICAS DO ECOSISTEMA

### Objetivo

Mapear **todas as conexões do Sistema Econômico, Monetização e FriendCoins** com os demais módulos do FriendApp, mostrando **entradas, saídas e ciclos de dados**. Isso garante que os desenvolvedores compreendam a posição estratégica do sistema dentro do ecossistema.

---

### Integrações Principais

#### 1. IA Aurah Kosmos

- **Entrada:** dados de impacto vibracional, harmonia de interações, padrões energéticos.
- **Saída:** cálculo de índice de impacto, recompensas surpresa, recomendações econômicas.

#### 2. Sistema de Locais Parceiros

- **Entrada:** contratos comerciais, check-ins em locais.
- **Saída:** monetização para parceiros, descontos aplicados, relatórios financeiros.

#### 3. Sistema de Eventos

- **Entrada:** criação de eventos gratuitos/pagos, feedback vibracional dos participantes.
- **Saída:** monetização via ingressos, recompensas por impacto ao criador, taxas do FriendApp.

#### 4. Feed Sensorial & Painel Vibracional

- **Entrada:** posts, interações e feedbacks.

- **Saída:** FriendCoins proporcionais ao impacto, boosts, posts patrocinados.

## 5. Chat Vibracional

- **Entrada:** envio de mensagens, uso de stickers.
- **Saída:** microtransações em FC (stickers premium), presentes vibracionais.

## 6. Sistema de Doações e Impacto Social

- **Entrada:** doações em FC ou BRL, escolha de causas sociais.
- **Saída:** registros públicos, métricas de impacto no perfil vibracional, auditoria externa.

## 7. Sistema de Segurança Vibracional

- **Entrada:** logs de transações, análise de clusters suspeitos (farming/bots).
- **Saída:** restrições, penalidades e rollback de fraudes detectadas.

## 8. Middleware de Pagamentos (RevenueCat/Chargebee)

- **Entrada:** solicitações de upgrades Premium, compras em moeda real.
- **Saída:** status de assinatura, confirmação de pagamentos, webhooks para FriendApp.

---

## Dependências Cíclicas

- **Ciclo Econômico-Vibracional:**

Interações → Impacto Vibracional (IA) → Recompensas em FriendCoins → Uso em Eventos/Locais/Chat → Novas Interações → Recalibração do sistema.

- **Ciclo Premium-Economia:**

Upgrade Premium → mais benefícios → mais uso econômico → mais valor para FriendApp → relatórios para investidores → reinvestimento em infraestrutura Premium.

- **Ciclo Social-Econômico:**

Doações → impacto social → aumento de reputação do usuário → mais engajamento → maior circulação de moedas → reforço da economia

consciente.

## Representação Técnica (simplificada em YAML)

Sistema\_Economico:

entradas:

- Interações de usuários
- Pagamentos (BRL/USD)
- Feedback vibracional
- Contratos de parceiros
- Dados de segurança antifraude

saídas:

- FriendCoins creditadas/debitadas
- Benefícios Premium
- Relatórios econômicos
- Ajustes inflacionários/estagnação
- Penalidades e bloqueios

integrações:

- IA\_Aurah\_Kosmos
- Locais\_Parceiros
- Eventos
- Feed\_Sensorial
- Chat
- Doações
- Segurança\_Vibracional
- Middleware\_Pagamentos

## Saída da Seção

- Mapeamento completo de **entradas, saídas e integrações**.
- Clareza para desenvolvedores sobre dependências cíclicas.
- Garantia de **coerência e sincronização** em todo ecossistema FriendApp.