# SynthonIA Chakra — Arquitetura v1

## Visão (mínimo de telas, máximo de sentido)

**Módulos (v1):**

1. **Recovery & Prontidão**
2. **Treinos**
3. **Rehabilitation**
4. **Análise com IA**
5. **Analytics (dashboard universal com seleção de variáveis + IA em tempo real)**
6. **Configurações**

**Papéis:** admin, atleta, treinador, fisioterapeuta (e perfis *atleta escolhe treinador*, *paciente escolhe fisio*). Todos com RLS robusto.

**Tema Chakra (fundo escuro):** paleta saturada por chakra em cards/gráficos:  
Muladhara #D7263D · Svadhisthana #F46036 · Manipura #FFC145 · Anahata #19A974 · Vishuddha #2E86AB · Ajna #4747FF · Sahasrara #8A2BE2.  
Textos: #E6E6E6 / #BDBDBD. Fundo: #0D0E12/#12131A.

**Stack recomendada (otimizada p/ Netlify + Supabase):**

* **Frontend:** React/Next.js (TypeScript) com Recharts/Plotly e Tailwind.
* **Backend:** **Supabase** (Postgres + Auth + Storage + Edge Functions Deno/TS).
* **IA:** OpenAI API (env var), prompts determinísticos e guardrails.
* **Motivo para *não***\*\* Python no core v1:\*\* Netlify + Supabase têm fluxo nativo TS/Edge Functions; menos latência, deploy 1‑click, RLS no banco faz o “trabalho pesado”. **Python** pode entrar como microserviço externo (Cloud Run/Lambda) para análises especiais futuramente.

## Fluxo de navegação (mínimo de telas)

**Home/Hub (Roda de Chakras):** 7 ícones circulares (os 6 módulos + Admin visível só p/ admin).  
**Recovery & Prontidão:** formulário diário 5 itens + PRS/TQR (1‑clique); histórico com sparkline; card **IIERP** (0–100).  
**Treinos:** registro rápido de sessão (duração, PSE, notas); importador CSV; detalhamento por sessão; tonelagem/densidade.  
**Rehabilitation:** plano de reabilitação por paciente, tarefas/observações do fisio; evolução de dor (VAS/NPRS).  
**Análise com IA:** “Pergunte aos seus dados” + *insights* gerados por variável/seleção (abaixo de cada gráfico).  
**Analytics:** grade de gráficos (ATL/CTL/TSB, monotonia/strain, sRPE, TSS, TRIMP, humor/BRUMS, dor, sono) + seletor de variáveis e janelas; exportar PDF/CSV.  
**Configurações:** perfil, vínculo atleta‑treinador/fisio, notificações, privacidade.  
**Comentários Contextuais:** em qualquer card/gráfico, botão “Comentar” (thread por objeto).

## Banco de Dados (Postgres/Supabase)

**Convenções:**

* chaves: id uuid default gen\_random\_uuid()
* timestamps: created\_at timestamptz default now(), updated\_at com trigger
* soft delete opcional: deleted\_at
* fks com on delete cascade

### Tabelas núcleo

**1. profiles**

* id uuid primary key (== auth.users.id)
* role text check (role in ('admin','atleta','treinador','fisioterapeuta')) not null
* full\_name text, birthdate date, gender text, avatar\_url text
* created\_at, updated\_at

**2. relationships** (vínculos)

* id uuid pk
* athlete\_id uuid references profiles(id)
* trainer\_id uuid references profiles(id)
* patient\_id uuid references profiles(id)
* physio\_id uuid references profiles(id)
* active boolean default true
* *Regra:* ou atleta↔treinador **ou** paciente↔fisioterapeuta.

**3. sessions** (sessões de treino/reabilitação)

* id uuid pk
* owner\_id uuid references profiles(id) (quem realizou)
* type text check ('treino'|'reab')
* modality text (força, corrida, bike, funcional…)
* date date, start\_time time, duration\_min numeric
* sRPE numeric (0–10), session\_load numeric (sRPE×duração)
* notes text

**4. strength\_sets** (opcional p/ tonelagem)

* id uuid pk, session\_id uuid fk
* exercise text, sets int, reps int, load\_kg numeric, rest\_s int
* **Views** derivam: tonelagem = sets\*reps\*load\_kg, rest\_min = rest\_s/60

**5. endurance\_metrics** (para TSS/TRIMP quando houver dados)

* id uuid pk, session\_id uuid fk
* avg\_hr int, hr\_zones jsonb, if numeric, np numeric, tss numeric, trimp numeric

**6. wellness\_daily** (Recovery)

* id uuid pk, owner\_id fk, date date unique(owner)
* sleep\_quality int, fatigue int, soreness int, stress int, mood int (1–5 ou 0–10)
* PRS int (0–10), TQR int (6–20)

**7. pain\_map** (opcional, dor por segmento)

* id uuid pk, owner\_id, date, body\_part text, pain\_0\_10 int

**8. psych\_scales** (v1 inclui BRUMS e PSS‑10)

* id uuid pk, owner\_id, date
* scale text check ('BRUMS'|'PSS10')
* payload jsonb (BRUMS 24 itens, PSS10 10 itens)
* scores jsonb (sumários por subescala; índice de perturbação)
* consent boolean default true

**9. readiness** (cálculos diários)

* id uuid pk, owner\_id, date
* IIERP numeric (0–100)
* readiness\_comp numeric, recovery\_comp numeric (componentes normalizados)
* TSB numeric, ATL numeric, CTL numeric, monotony numeric, strain numeric
* índices auxiliares para dashboards

**10. weekly\_loads** (agregados)

* id uuid pk, owner\_id, year int, week int
* total\_load numeric, mean\_daily\_load numeric, sd\_daily\_load numeric
* monotony numeric, strain numeric

**11. comments**

* id uuid pk, author\_id fk, entity\_type text, entity\_id uuid, message text, created\_at

**12. numerology\_profiles** (módulo lúdico)

* id uuid pk, owner\_id, birthdate, full\_name
* numbers jsonb (destino, expressão, alma…)
* cycles jsonb (ano/mês/dia pessoal)

**13. breathwork\_sessions**

* id uuid pk, owner\_id, date, type text, duration\_min int, perceived\_effect text

**14. tests\_catalog** (protocolos)

* id uuid pk, code text unique (e.g., ‘YOYO\_IR1’,‘1RM\_BACKSQUAT’)
* name text, description text, refs text

**15. test\_runs** (execuções)

* id uuid pk, owner\_id, test\_code text fk -> tests\_catalog(code)
* date, result jsonb, derived jsonb (e.g., VO2max estimado)

**16. audit\_events** (admin analytics)

* id uuid pk, actor\_id, event text, metadata jsonb, created\_at

### Índices recomendados

* sessions(owner\_id, date); wellness\_daily(owner\_id, date); readiness(owner\_id, date)
* weekly\_loads(owner\_id, year, week); GIN para psych\_scales.payload/scores

## Funções SQL / Views (cálculo server-side)

**Carga da Sessão (sRPE):** session\_load = sRPE \* duration\_min (persistido em sessions).

**Tonelagem por sessão (view):**

create view v\_session\_tonnage as  
select s.id as session\_id,  
 coalesce(sum(ss.sets\*ss.reps\*ss.load\_kg),0) as tonnage\_kg,  
 coalesce(sum(ss.rest\_s),0)/60.0 as rest\_min  
from sessions s  
left join strength\_sets ss on ss.session\_id = s.id  
where s.type = 'treino'  
group by s.id;

**Densidade (kg/min):** density = tonnage\_kg / nullif(duration\_min,0) (view com sessions × v\_session\_tonnage).

**ATL/CTL (EMA):** janelas configuráveis por usuário; default ATL=7 dias, CTL=42.  
Exemplo de função (pseudo‑SQL, calcular via procedure diária):

-- Exemplo conceitual (implementar em SQL/Edge Func):  
ATL\_today = EMA(load\_daily, 1/7)  
CTL\_today = EMA(load\_daily, 1/42)  
TSB\_today = CTL\_today - ATL\_today

**Monotonia / Strain (semanal):**

monotony = mean\_daily\_load / nullif(sd\_daily\_load,0)  
strain = total\_load \* monotony

**IIERP (Índice Integrado de Estado Recuperação‑Prontidão 0–100):**

* **Recuperação (0–100)**: normalizar (z→0–100) sleep\_quality, fatigue(−), soreness(−), stress(−), mood, PRS, TQR com pesos: sono 0.2, fadiga 0.15, dor 0.15, estresse 0.15, humor 0.15, PRS 0.1, TQR 0.1.
* **Prontidão (0–100)**: TSB (mapeado a 0–100 via faixas seguras), monotony(−), strain(−), tendência de carga (variação 7d), *outlier flags*.
* **Composição:** IIERP = 0.6\*Recuperação + 0.4\*Prontidão.
* **Zonas:** 0–39 vermelho; 40–59 âmbar; 60–79 verde; 80–100 dourado.

Implementar como função (Edge Function) que escreve em readiness a cada novo input diário.

## RLS — Políticas de segurança

Ativar RLS em todas as tabelas com acesso a dados pessoais.

**profiles**

alter table profiles enable row level security;  
create policy "self read" on profiles for select using (auth.uid() = id or exists(  
 select 1 from profiles p where p.id = auth.uid() and p.role='admin'  
));  
create policy "self update" on profiles for update using (auth.uid() = id);

**sessions / wellness\_daily / readiness / psych\_scales / pain\_map / test\_runs / breathwork\_sessions**

alter table sessions enable row level security;  
create policy "owner read" on sessions for select using (owner\_id = auth.uid() or exists(  
 select 1 from profiles me  
 where me.id = auth.uid()  
 and (  
 me.role='admin'  
 or (me.role='treinador' and exists(select 1 from relationships r where r.trainer\_id=me.id and r.athlete\_id=sessions.owner\_id and r.active))  
 or SynthonIA Chakra — Arquitetura v1  
  
Visão (mínimo de telas, máximo de sentido)  
  
Módulos (v1):  
  
Recovery & Prontidão  
  
Treinos  
  
Rehabilitation  
  
Análise com IA  
  
Analytics (dashboard universal com seleção de variáveis + IA em tempo real)  
  
Configurações  
  
Papéis: admin, atleta, treinador, fisioterapeuta (e perfis atleta escolhe treinador, paciente escolhe fisio). Todos com RLS robusto.  
  
Tema Chakra (fundo escuro): paleta saturada por chakra em cards/gráficos:Muladhara #D7263D · Svadhisthana #F46036 · Manipura #FFC145 · Anahata #19A974 · Vishuddha #2E86AB · Ajna #4747FF · Sahasrara #8A2BE2.Textos: #E6E6E6 / #BDBDBD. Fundo: #0D0E12/#12131A.  
  
Stack recomendada (otimizada p/ Netlify + Supabase):  
  
Frontend: React/Next.js (TypeScript) com Recharts/Plotly e Tailwind.  
  
Backend: Supabase (Postgres + Auth + Storage + Edge Functions Deno/TS).  
  
IA: OpenAI API (env var), prompts determinísticos e guardrails.  
  
Motivo para não Python no core v1: Netlify + Supabase têm fluxo nativo TS/Edge Functions; menos latência, deploy 1‑click, RLS no banco faz o “trabalho pesado”. Python pode entrar como microserviço externo (Cloud Run/Lambda) para análises especiais futuramente.  
  
Fluxo de navegação (mínimo de telas)  
  
Home/Hub (Roda de Chakras): 7 ícones circulares (os 6 módulos + Admin visível só p/ admin).Recovery & Prontidão: formulário diário 5 itens + PRS/TQR (1‑clique); histórico com sparkline; card IIERP (0–100).Treinos: registro rápido de sessão (duração, PSE, notas); importador CSV; detalhamento por sessão; tonelagem/densidade.Rehabilitation: plano de reabilitação por paciente, tarefas/observações do fisio; evolução de dor (VAS/NPRS).Análise com IA: “Pergunte aos seus dados” + insights gerados por variável/seleção (abaixo de cada gráfico).Analytics: grade de gráficos (ATL/CTL/TSB, monotonia/strain, sRPE, TSS, TRIMP, humor/BRUMS, dor, sono) + seletor de variáveis e janelas; exportar PDF/CSV.Configurações: perfil, vínculo atleta‑treinador/fisio, notificações, privacidade.Comentários Contextuais: em qualquer card/gráfico, botão “Comentar” (thread por objeto).  
  
Banco de Dados (Postgres/Supabase)  
  
Convenções:  
  
chaves: id uuid default gen\_random\_uuid()  
  
timestamps: created\_at timestamptz default now(), updated\_at com trigger  
  
soft delete opcional: deleted\_at  
  
fks com on delete cascade  
  
Tabelas núcleo  
  
1. profiles  
  
id uuid primary key (== auth.users.id)  
  
role text check (role in ('admin','atleta','treinador','fisioterapeuta')) not null  
  
full\_name text, birthdate date, gender text, avatar\_url text  
  
created\_at, updated\_at  
  
2. relationships (vínculos)  
  
id uuid pk  
  
athlete\_id uuid references profiles(id)  
  
trainer\_id uuid references profiles(id)  
  
patient\_id uuid references profiles(id)  
  
physio\_id uuid references profiles(id)  
  
active boolean default true  
  
Regra: ou atleta↔treinador ou paciente↔fisioterapeuta.  
  
3. sessions (sessões de treino/reabilitação)  
  
id uuid pk  
  
owner\_id uuid references profiles(id) (quem realizou)  
  
type text check ('treino'|'reab')  
  
modality text (força, corrida, bike, funcional…)  
  
date date, start\_time time, duration\_min numeric  
  
sRPE numeric (0–10), session\_load numeric (sRPE×duração)  
  
notes text  
  
4. strength\_sets (opcional p/ tonelagem)  
  
id uuid pk, session\_id uuid fk  
  
exercise text, sets int, reps int, load\_kg numeric, rest\_s int  
  
Views derivam: tonelagem = sets\*reps\*load\_kg, rest\_min = rest\_s/60  
  
5. endurance\_metrics (para TSS/TRIMP quando houver dados)  
  
id uuid pk, session\_id uuid fk  
  
avg\_hr int, hr\_zones jsonb, if numeric, np numeric, tss numeric, trimp numeric  
  
6. wellness\_daily (Recovery)  
  
id uuid pk, owner\_id fk, date date unique(owner)  
  
sleep\_quality int, fatigue int, soreness int, stress int, mood int (1–5 ou 0–10)  
  
PRS int (0–10), TQR int (6–20)  
  
7. pain\_map (opcional, dor por segmento)  
  
id uuid pk, owner\_id, date, body\_part text, pain\_0\_10 int  
  
8. psych\_scales (v1 inclui BRUMS e PSS‑10)  
  
id uuid pk, owner\_id, date  
  
scale text check ('BRUMS'|'PSS10')  
  
payload jsonb (BRUMS 24 itens, PSS10 10 itens)  
  
scores jsonb (sumários por subescala; índice de perturbação)  
  
consent boolean default true  
  
9. readiness (cálculos diários)  
  
id uuid pk, owner\_id, date  
  
IIERP numeric (0–100)  
  
readiness\_comp numeric, recovery\_comp numeric (componentes normalizados)  
  
TSB numeric, ATL numeric, CTL numeric, monotony numeric, strain numeric  
  
índices auxiliares para dashboards  
  
10. weekly\_loads (agregados)  
  
id uuid pk, owner\_id, year int, week int  
  
total\_load numeric, mean\_daily\_load numeric, sd\_daily\_load numeric  
  
monotony numeric, strain numeric  
  
11. comments  
  
id uuid pk, author\_id fk, entity\_type text, entity\_id uuid, message text, created\_at  
  
12. numerology\_profiles (módulo lúdico)  
  
id uuid pk, owner\_id, birthdate, full\_name  
  
numbers jsonb (destino, expressão, alma…)  
  
cycles jsonb (ano/mês/dia pessoal)  
  
13. breathwork\_sessions  
  
id uuid pk, owner\_id, date, type text, duration\_min int, perceived\_effect text  
  
14. tests\_catalog (protocolos)  
  
id uuid pk, code text unique (e.g., 'YOYO\_IR1','1RM\_BACKSQUAT')  
  
name text, description text, refs text  
  
15. test\_runs (execuções)  
  
id uuid pk, owner\_id, test\_code text fk -> tests\_catalog(code)  
  
date, result jsonb, derived jsonb (e.g., VO2max estimado)  
  
16. audit\_events (admin analytics)  
  
id uuid pk, actor\_id, event text, metadata jsonb, created\_at  
  
Índices recomendados  
  
sessions(owner\_id, date); wellness\_daily(owner\_id, date); readiness(owner\_id, date)  
  
weekly\_loads(owner\_id, year, week); GIN para psych\_scales.payload/scores  
  
Funções SQL / Views (cálculo server-side)  
  
Carga da Sessão (sRPE): session\_load = sRPE \* duration\_min (persistido em sessions).  
  
Tonelagem por sessão (view):  
  
create view v\_session\_tonnage as  
select s.id as session\_id,  
 coalesce(sum(ss.sets\*ss.reps\*ss.load\_kg),0) as tonnage\_kg,  
 coalesce(sum(ss.rest\_s),0)/60.0 as rest\_min  
from sessions s  
left join strength\_sets ss on ss.session\_id = s.id  
where s.type = 'treino'  
group by s.id;  
  
Densidade (kg/min): density = tonnage\_kg / nullif(duration\_min,0) (view com sessions × v\_session\_tonnage).  
  
ATL/CTL (EMA): janelas configuráveis por usuário; default ATL=7 dias, CTL=42.Exemplo de função (pseudo‑SQL, calcular via procedure diária):  
  
-- Exemplo conceitual (implementar em SQL/Edge Func):  
ATL\_today = EMA(load\_daily, 1/7)  
CTL\_today = EMA(load\_daily, 1/42)  
TSB\_today = CTL\_today - ATL\_today  
  
Monotonia / Strain (semanal):  
  
monotony = mean\_daily\_load / nullif(sd\_daily\_load,0)  
strain = total\_load \* monotony  
  
IIERP (Índice Integrado de Estado Recuperação‑Prontidão 0–100):  
  
Recuperação (0–100): normalizar (z→0–100) sleep\_quality, fatigue(−), soreness(−), stress(−), mood, PRS, TQR com pesos: sono 0.2, fadiga 0.15, dor 0.15, estresse 0.15, humor 0.15, PRS 0.1, TQR 0.1.  
  
Prontidão (0–100): TSB (mapeado a 0–100 via faixas seguras), monotony(−), strain(−), tendência de carga (variação 7d), outlier flags.  
  
Composição: IIERP = 0.6\*Recuperação + 0.4\*Prontidão.  
  
Zonas: 0–39 vermelho; 40–59 âmbar; 60–79 verde; 80–100 dourado.  
  
Implementar como função (Edge Function) que escreve em readiness a cada novo input diário.  
  
RLS — Políticas de segurança  
  
Ativar RLS em todas as tabelas com acesso a dados pessoais.  
  
profiles  
  
alter table profiles enable row level security;  
create policy "self read" on profiles for select using (auth.uid() = id or exists(  
 select 1 from profiles p where p.id = auth.uid() and p.role='admin'  
));  
create policy "self update" on profiles for update using (auth.uid() = id);  
  
sessions / wellness\_daily / readiness / psych\_scales / pain\_map / test\_runs / breathwork\_sessions  
  
alter table sessions enable row level security;  
create policy "owner read" on sessions for select using (owner\_id = auth.uid() or exists(  
 select 1 from profiles me  
 where me.id = auth.uid()  
 and (  
 me.role='admin'  
 or (me.role='treinador' and exists(select 1 from relationships r where r.trainer\_id=me.id and r.athlete\_id=sessions.owner\_id and r.active))  
 or (me.role='fisioterapeuta' and exists(select 1 from relationships r where r.physio\_id=me.id and r.patient\_id=sessions.owner\_id and r.active))  
 )  
));  
create policy "owner write" on sessions for insert with check (owner\_id = auth.uid());  
create policy "owner update" on sessions for update using (owner\_id = auth.uid());  
  
Replicar padrão para tabelas de dados do usuário. comments permite leitura a quem tem acesso ao entity e escrita ao autor/vínculos.  
  
admin (audit\_events, weekly\_loads view global): somente admin.  
  
Edge Functions (Deno/TS) — /functions  
  
compute\_readiness\_dailyEntrada: owner\_id, date. Busca dados de wellness\_daily, cargas dos últimos 42d, calcula ATL/CTL/TSB, monotonia/strain da semana, normaliza recuperação/prontidão e grava readiness.  
  
insights\_for\_selectionEntrada: lista de variáveis/intervalo. Retorna estatísticas (correlações Pearson/Spearman, tendências, outliers) + prompt para IA gerar explicação neurofisiológica.  
  
ask\_your\_dataEntrada: linguagem natural → parser (RegEx/LLM) → SQL segura (whitelist) → resultados tabulares + resumo. Proteções contra SQL injection (gerar apenas de templates permitidos).  
  
admin\_metrics\_trackEvento de page/screen view, horário, usuário, feature usage. Popula audit\_events.  
  
Frontend — estrutura de pastas (Next.js)  
  
src/  
 app/  
 (auth)/login  
 (dashboard)/  
 page.tsx // Hub “Roda de Chakras”  
 recovery/  
 treinos/  
 rehab/  
 analytics/  
 ai/  
 settings/  
 admin/ // visível só pra admin  
 components/  
 charts/ // Line, Bar, Radar, Pareto  
 cards/  
 forms/  
 comments/  
 lib/  
 supabaseClient.ts  
 fetchers.ts  
 calc.ts // mapeamentos normalização, cores, zonas  
 prompts.ts // prompts IA padronizados  
 styles/  
 types/  
  
Design Tokens Chakra:  
  
export const chakraColors = {  
 root: '#D7263D', sacral: '#F46036', solar: '#FFC145', heart: '#19A974',  
 throat: '#2E86AB', thirdEye: '#4747FF', crown: '#8A2BE2'  
}  
  
Componentes-chave:  
  
KPI Cards (IIERP, TSB, Monotonia, Strain) com semáforos.  
  
Selector de variáveis (multiselect) → gráficos reativos.  
  
Painel de comentários ancorado ao gráfico/card.  
  
Export PDF/CSV.  
  
IA — Padrões de Prompt (resumo)  
  
1) Insight curto abaixo do gráfico  
  
Contexto: [variáveis X, período Y, coeficientes, tendências].Tarefa: explique achados em linguagem dupla (técnica e leiga).Inclua hipótese neurofisiológica plausível; recomende micro‑ajustes de treino/recuperação; 80–120 palavras.  
  
2) Pergunte aos seus dados  
  
Intenção do usuário → mapear para consulta segura (templates).IA apenas explica e contextualiza resultados já retornados pelo SQL/estatística.  
  
Guardrails: proibir diagnósticos médicos; reforçar educação; mostrar incerteza.  
  
CSV Template (upload Treinos)  
  
 date,modality,type,duration\_min,sRPE,exercise,sets,reps,load\_kg,rest\_s,notes  
 2025-08-05,forca,treino,60,5,Back Squat,5,5,100,180,força máxima  
 2025-08-06,corrida,treino,45,6,,,,,treino intervalado  
  
Campos vazios (for endurance) serão ignorados no cálculo de tonelagem.  
  
Fórmulas e Normalizações (detalhe)  
  
Normalização 0–100: score = clamp( (x - p10) / (p90 - p10), 0, 1 ) \* 100 com p10/p90 do histórico individual (robusto a outliers).  
  
Variáveis “quanto menor melhor” (fadiga, dor, estresse, monotonia, strain) usam inversão 100 - score.  
  
Mapeamento TSB → 0–100 (exemplo):TSB ≤ -30 → 5; -15 → 35; 0 → 60; +15 → 85; ≥ +30 → 95 (curva logística suave).  
  
Monotonia Limiares: >2 alerta; >2.5 crítico (cards em vermelho, tooltip educativo).  
  
Densidade: kg/min (mostrar badge “alta/baixa” comparado à mediana dos últimos 30 dias).  
  
IIERP composição (pesos v1):  
  
Recuperação: sono(0.2) fadiga(0.15) dor(0.15) estresse(0.15) humor(0.15) PRS(0.1) TQR(0.1)  
  
Prontidão: TSB(0.4) monotonia(0.2) strain(0.2) tendência carga 7d(0.2)  
  
Analytics (dashboard universal)  
  
Selector de: variáveis (multi), janela (7/28/90d), agregação (dia/semana), smoothing (EMA).  
  
Gráficos disponíveis: Linha (séries), Barras empilhadas, Radar (perfil), Pareto (fatores mais associados a outcome), Dispersão com regressão (Pearson/Spearman).  
  
Outputs IA: insight curto + drilldown expandível.  
  
Export: PNG/PDF/CSV.  
  
Comentários: thread por gráfico/intervalo salvo.  
  
Administração (métricas de uso)  
  
Usuários ativos/dia, retenção 7/28d, telas mais vistas, horário de pico, tempo de sessão.  
  
Eventos: login, upload, criação de sessão, comentários, execução de IA.  
  
Painel apenas para admin.  
  
Segurança e Privacidade  
  
RLS ativo desde o início; admin via role em profiles.  
  
Consentimento explícito para psych\_scales; visibilidade padrão somente para o próprio, e para vínculos se autorizado.  
  
Env Vars: nunca commitar chaves. Usar NETLIFY.toml + UI do Netlify para secrets.  
  
Auditoria: audit\_events + logs de Edge Functions.  
  
Backups: habilitar Point‑in‑Time Recovery/backups no Supabase; script de export diário opcional.  
  
Variáveis .env (exemplo):  
  
NEXT\_PUBLIC\_SUPABASE\_URL=  
NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY=  
OPENAI\_API\_KEY=  
READINESS\_WINDOW\_ATL=7  
READINESS\_WINDOW\_CTL=42  
  
Deploy — Passo a passo  
  
Supabase: criar projeto → importar SQL (tabelas, views, policies) → ativar RLS → criar bucket uploads.  
  
GitHub: repositório (usar o seu) com template Next.js/TS.  
  
Netlify: conectar ao GitHub → setar env vars → build npm run build → publish.  
  
Edge Functions: deploy via Supabase CLI; criar secrets das funções conforme necessidade.  
  
Seeds: script para gerar 60 dias de dados fictícios por usuário (ver seção seguinte).  
  
Testes: verificar RLS (tentativas cruzadas devem falhar), gráficos, IA e comentários.  
  
Seeds — dados fictícios (resumo)  
  
Criar 1 atleta, 1 treinador, 1 fisio; vincular relações.  
  
Gerar wellness diário 60d (ruído normal); gerar 4–5 sessões semanais (variação sRPE/duração); calcular cargas.  
  
Rodar compute\_readiness\_daily para preencher readiness/weekly\_loads.  
  
Roadmap técnico curto (2 sprints)  
  
Sprint 1 (base funcional): Auth, profiles, vínculos; Recovery (form + histórico); Treinos (cadastro + cálculo sRPE); Readiness básico (IIERP parcial); Analytics com 3 gráficos; Comentários; RLS.Sprint 2 (profundo): Import CSV; Tonelagem/Densidade; ATL/CTL/TSB; Monotonia/Strain; BRUMS/PSS‑10; Edge Functions (insights); Admin métricas; Export PDF.  
  
Alternativa Python (quando usar)  
  
Criar microserviço Python (FastAPI) para análises pesadas (modelos de previsão, ARIMA, SHAP) em Cloud Run/Lambda; exposto via endpoint privado; autenticado por token.  
  
Vantagem: ecossistema científico (numpy/pandas/statsmodels); Desvantagem: complexidade de devops.  
  
Recomendação: v2 após v1 estável.  
  
Apêndice — Snippets úteis  
  
Trigger updated\_at  
  
create or replace function set\_updated\_at()  
returns trigger language plpgsql as $$  
begin new.updated\_at = now(); return new; end; $$;  
  
Comentário vinculado a gráfico/card (modelo genérico):  
  
-- entity\_type: 'session'|'readiness'|'chart:analytics:<hash>'  
-- entity\_id: uuid ou hash de contexto  
  
Consulta para weekly\_loads (exemplo):  
  
insert into weekly\_loads (owner\_id, year, week, total\_load, mean\_daily\_load, sd\_daily\_load, monotony, strain)  
select owner\_id,  
 extract(isoyear from date) as year,  
 extract(week from date) as week,  
 sum(session\_load) as total\_load,  
 avg(session\_load) as mean\_daily,  
 stddev\_samp(session\_load) as sd\_daily,  
 case when stddev\_samp(session\_load)=0 then null else avg(session\_load)/stddev\_samp(session\_load) end as monotony,  
 sum(session\_load) \* case when stddev\_samp(session\_load)=0 then null else avg(session\_load)/stddev\_samp(session\_load) end as strain  
from sessions  
where type='treino'  
 and owner\_id = $1  
 and date between $2 and $3  
group by owner\_id, extract(isoyear from date), extract(week from date);  
  
Pseudocódigo do cálculo IIERP  
  
function to0100(x:number, p10:number, p90:number, invert=false){  
 const t = Math.max(0, Math.min(1, (x - p10) / Math.max(1e-6, p90 - p10)));  
 const s = t\*100; return invert ? 100 - s : s;  
}  
  
function iierp(day){  
 const rec = 0.2\*to0100(day.sleep,p10.sleep,p90.sleep)  
 + 0.15\*to0100(day.fatigue,p10.fatigue,p90.fatigue,true)  
 + 0.15\*to0100(day.soreness,p10.soreness,p90.soreness,true)  
 + 0.15\*to0100(day.stress,p10.stress,p90.stress,true)  
 + 0.15\*to0100(day.mood,p10.mood,p90.mood)  
 + 0.10\*to0100(day.PRS,0,10)  
 + 0.10\*to0100(day.TQR,6,20);  
  
 const prd = 0.4\*mapTSB(day.TSB)  
 + 0.2\*to0100(day.monotony,p10.monotony,p90.monotony,true)  
 + 0.2\*to0100(day.strain,p10.strain,p90.strain,true)  
 + 0.2\*to0100(day.loadTrend,p10.loadTrend,p90.loadTrend,true);  
  
 return 0.6\*rec + 0.4\*prd;  
}  
  
Conclusão  
  
Este documento consolida arquitetura, esquema de dados, fórmulas, RLS, funções de borda, UI, deploy e guardrails para que você recupere o projeto a qualquer momento. A v1 foca em confiabilidade e clareza: poucas telas, cálculos sólidos, IA explicativa. Python pode chegar na v2 como motor analítico avançado sem comprometer a simplicidade operacional da v1.  
  
(me.role='fisioterapeuta' and exists(select 1 from relationships r where r.physio\_id=me.id and r.patient\_id=sessions.owner\_id and r.active))  
 )  
));  
create policy "owner write" on sessions for insert with check (owner\_id = auth.uid());  
create policy "owner update" on sessions for update using (owner\_id = auth.uid());

Replicar padrão para tabelas de dados do usuário. comments permite leitura a quem tem acesso ao entity e escrita ao autor/vínculos.

**admin (audit\_events, weekly\_loads view global)**: somente admin.

## Edge Functions (Deno/TS) — */functions*

1. compute\_readiness\_daily  
   Entrada: owner\_id, date. Busca dados de wellness\_daily, cargas dos últimos 42d, calcula **ATL/CTL/TSB**, **monotonia/strain** da semana, normaliza recuperação/prontidão e grava readiness.
2. insights\_for\_selection  
   Entrada: lista de variáveis/intervalo. Retorna estatísticas (correlações Pearson/Spearman, tendências, outliers) + **prompt** para IA gerar explicação neurofisiológica.
3. ask\_your\_data  
   Entrada: linguagem natural → parser (RegEx/LLM) → SQL segura (whitelist) → resultados tabulares + resumo. Proteções contra SQL injection (gerar apenas de templates permitidos).
4. admin\_metrics\_track  
   Evento de page/screen view, horário, usuário, *feature usage*. Popula audit\_events.

## Frontend — estrutura de pastas (Next.js)

src/  
 app/  
 (auth)/login  
 (dashboard)/  
 page.tsx // Hub “Roda de Chakras”  
 recovery/  
 treinos/  
 rehab/  
 analytics/  
 ai/  
 settings/  
 admin/ // visível só pra admin  
 components/  
 charts/ // Line, Bar, Radar, Pareto  
 cards/  
 forms/  
 comments/  
 lib/  
 supabaseClient.ts  
 fetchers.ts  
 calc.ts // mapeamentos normalização, cores, zonas  
 prompts.ts // prompts IA padronizados  
 styles/  
 types/

**Design Tokens Chakra:**

export const chakraColors = {  
 root: '#D7263D', sacral: '#F46036', solar: '#FFC145', heart: '#19A974',  
 throat: '#2E86AB', thirdEye: '#4747FF', crown: '#8A2BE2'  
}

**Componentes-chave:**

* **KPI Cards** (IIERP, TSB, Monotonia, Strain) com semáforos.
* **Selector** de variáveis (multiselect) → gráficos reativos.
* **Painel de comentários** ancorado ao gráfico/card.
* **Export** PDF/CSV.

## IA — Padrões de Prompt (resumo)

**1) Insight curto abaixo do gráfico**

Contexto: [variáveis X, período Y, coeficientes, tendências].  
Tarefa: explique achados em linguagem dupla (técnica e leiga).  
Inclua hipótese neurofisiológica plausível; recomende micro‑ajustes de treino/recuperação; 80–120 palavras.

**2) Pergunte aos seus dados**

Intenção do usuário → mapear para consulta segura (templates).  
IA apenas **explica** e **contextualiza** resultados já retornados pelo SQL/estatística.

**Guardrails:** proibir diagnósticos médicos; reforçar educação; mostrar incerteza.

## CSV Template (upload Treinos)

date,modality,type,duration\_min,sRPE,exercise,sets,reps,load\_kg,rest\_s,notes  
 2025-08-05,forca,treino,60,5,Back Squat,5,5,100,180,força máxima  
 2025-08-06,corrida,treino,45,6,,,,,treino intervalado

Campos vazios (for endurance) serão ignorados no cálculo de tonelagem.

## Fórmulas e Normalizações (detalhe)

**Normalização 0–100:** score = clamp( (x - p10) / (p90 - p10), 0, 1 ) \* 100 com p10/p90 do histórico individual (robusto a outliers).

* Variáveis “quanto menor melhor” (fadiga, dor, estresse, monotonia, strain) usam inversão 100 - score.

**Mapeamento TSB → 0–100 (exemplo):**  
TSB ≤ -30 → 5; -15 → 35; 0 → 60; +15 → 85; ≥ +30 → 95 (curva logística suave).

**Monotonia Limiares:** >2 alerta; >2.5 crítico (cards em vermelho, tooltip educativo).

**Densidade:** kg/min (mostrar *badge* “alta/baixa” comparado à mediana dos últimos 30 dias).

**IIERP composição (pesos v1):**

* Recuperação: sono(0.2) fadiga(0.15) dor(0.15) estresse(0.15) humor(0.15) PRS(0.1) TQR(0.1)
* Prontidão: TSB(0.4) monotonia(0.2) strain(0.2) tendência carga 7d(0.2)

## Analytics (dashboard universal)

* **Selector** de: variáveis (multi), janela (7/28/90d), agregação (dia/semana), smoothing (EMA).
* **Gráficos disponíveis:** Linha (séries), Barras empilhadas, Radar (perfil), Pareto (fatores mais associados a outcome), Dispersão com regressão (Pearson/Spearman).
* **Outputs IA:** insight curto + *drilldown* expandível.
* **Export:** PNG/PDF/CSV.
* **Comentários:** thread por gráfico/intervalo salvo.

## Administração (métricas de uso)

* Usuários ativos/dia, retenção 7/28d, telas mais vistas, horário de pico, tempo de sessão.
* Eventos: login, upload, criação de sessão, comentários, execução de IA.
* Painel apenas para admin.

## Segurança e Privacidade

* **RLS** ativo desde o início; admin via role em profiles.
* **Consentimento** explícito para psych\_scales; visibilidade padrão **somente** para o próprio, e para vínculos se autorizado.
* **Env Vars:** nunca commitar chaves. Usar NETLIFY.toml + UI do Netlify para secrets.
* **Auditoria:** audit\_events + logs de Edge Functions.
* **Backups:** habilitar **Point‑in‑Time Recovery**/backups no Supabase; script de export diário opcional.

**Variáveis .env (exemplo):**

NEXT\_PUBLIC\_SUPABASE\_URL=  
NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY=  
OPENAI\_API\_KEY=  
READINESS\_WINDOW\_ATL=7  
READINESS\_WINDOW\_CTL=42

## Deploy — Passo a passo

1. **Supabase**: criar projeto → importar SQL (tabelas, views, policies) → ativar RLS → criar bucket uploads.
2. **GitHub**: repositório (usar o seu) com template Next.js/TS.
3. **Netlify**: conectar ao GitHub → setar env vars → build npm run build → publish.
4. **Edge Functions**: deploy via Supabase CLI; criar *secrets* das funções conforme necessidade.
5. **Seeds**: script para gerar 60 dias de dados fictícios por usuário (ver seção seguinte).
6. **Testes**: verificar RLS (tentativas cruzadas devem falhar), gráficos, IA e comentários.

## Seeds — dados fictícios (resumo)

* Criar 1 atleta, 1 treinador, 1 fisio; vincular relações.
* Gerar wellness diário 60d (ruído normal); gerar 4–5 sessões semanais (variação sRPE/duração); calcular cargas.
* Rodar compute\_readiness\_daily para preencher readiness/weekly\_loads.

## Roadmap técnico curto (2 sprints)

**Sprint 1 (base funcional):** Auth, profiles, vínculos; Recovery (form + histórico); Treinos (cadastro + cálculo sRPE); Readiness básico (IIERP parcial); Analytics com 3 gráficos; Comentários; RLS.  
**Sprint 2 (profundo):** Import CSV; Tonelagem/Densidade; ATL/CTL/TSB; Monotonia/Strain; BRUMS/PSS‑10; Edge Functions (insights); Admin métricas; Export PDF.

## Alternativa Python (quando usar)

* Criar **microserviço** Python (FastAPI) para análises pesadas (modelos de previsão, ARIMA, SHAP) em Cloud Run/Lambda; exposto via endpoint privado; autenticado por token.
* Vantagem: ecossistema científico (numpy/pandas/statsmodels); Desvantagem: complexidade de devops.
* Recomendação: **v2** após v1 estável.

## Apêndice — Snippets úteis

**Trigger updated\_at**

create or replace function set\_updated\_at()  
returns trigger language plpgsql as $$  
begin new.updated\_at = now(); return new; end; $$;

**Comentário vinculado a gráfico/card** (modelo genérico):

-- entity\_type: 'session'|'readiness'|'chart:analytics:<hash>'  
-- entity\_id: uuid ou hash de contexto

**Consulta para weekly\_loads** (exemplo):

insert into weekly\_loads (owner\_id, year, week, total\_load, mean\_daily\_load, sd\_daily\_load, monotony, strain)  
select owner\_id,  
 extract(isoyear from date) as year,  
 extract(week from date) as week,  
 sum(session\_load) as total\_load,  
 avg(session\_load) as mean\_daily,  
 stddev\_samp(session\_load) as sd\_daily,  
 case when stddev\_samp(session\_load)=0 then null else avg(session\_load)/stddev\_samp(session\_load) end as monotony,  
 sum(session\_load) \* case when stddev\_samp(session\_load)=0 then null else avg(session\_load)/stddev\_samp(session\_load) end as strain  
from sessions  
where type='treino'  
 and owner\_id = $1  
 and date between $2 and $3  
group by owner\_id, extract(isoyear from date), extract(week from date);

**Pseudocódigo do cálculo IIERP**

function to0100(x:number, p10:number, p90:number, invert=false){  
 const t = Math.max(0, Math.min(1, (x - p10) / Math.max(1e-6, p90 - p10)));  
 const s = t\*100; return invert ? 100 - s : s;  
}  
  
function iierp(day){  
 const rec = 0.2\*to0100(day.sleep,p10.sleep,p90.sleep)  
 + 0.15\*to0100(day.fatigue,p10.fatigue,p90.fatigue,true)  
 + 0.15\*to0100(day.soreness,p10.soreness,p90.soreness,true)  
 + 0.15\*to0100(day.stress,p10.stress,p90.stress,true)  
 + 0.15\*to0100(day.mood,p10.mood,p90.mood)  
 + 0.10\*to0100(day.PRS,0,10)  
 + 0.10\*to0100(day.TQR,6,20);  
  
 const prd = 0.4\*mapTSB(day.TSB)  
 + 0.2\*to0100(day.monotony,p10.monotony,p90.monotony,true)  
 + 0.2\*to0100(day.strain,p10.strain,p90.strain,true)  
 + 0.2\*to0100(day.loadTrend,p10.loadTrend,p90.loadTrend,true);  
  
 return 0.6\*rec + 0.4\*prd;  
}

## Conclusão

Este documento consolida **arquitetura**, **esquema de dados**, **fórmulas**, **RLS**, **funções de borda**, **UI**, **deploy** e **guardrails** para que você recupere o projeto a qualquer momento. A v1 foca em confiabilidade e clareza: poucas telas, cálculos sólidos, IA explicativa. Python pode chegar na v2 como motor analítico avançado sem comprometer a simplicidade operacional da v1.