Gmail **Fillipe Guerra <fillipe.backup@gmail.com>**

IA_Autonoma_Parte10

1 mensagem

Fillipe Guerra <fillipe.backup@gmail.com> 28 de outubro de 2025 às 10:19
Para: Fillipe Augusto Gomes Guerra <fillipe182@hotmail.com>, Fillipe Guerra <fillipe.backup@gmail.com>

🧱 Fase 3 — Knowledge Base Avançada (completo)

0) TL;DR (o que entra agora)

- **BD/Migrações**: índices, views de métricas, flags e campos de controle novos.
- **Backend**: rotas Admin para **CRUD avançado**, **re-chunking**, **re-embed**, **rebuild HNSW** (incremental), **busca de inspeção**, **clusters** (PCA/UMAP) e **estatísticas nDCG/MRR**.
- **Frontend**: página `/admin/kb` evoluída (lista com filtros, editor de documento e de chunk, upload múltiplo, barra de progresso, "laboratório" para busca/score, heatmap de entidades, mapa 3D de embeddings).
- **Matemática/LaTeX**: chunking ótimo, MMR, confiança, PCA/UMAP, métricas (nDCG/MRR/CTR/CR) embutidas nos comentários para manter "a IA não pensa: só executa".

1) Banco de Dados — Drizzle (campos/índices extras)

1.1 Atualizações no schema existente

`/shared/schema.ai.kb.ts` (extensão do que mandei antes)

```
import { pgTable, uuid, varchar, text, boolean, integer, timestamp, jsonb, index, real } from
"drizzle-orm/pg-core";

export const aiDocuments = pgTable("ai_documents", {
  id: uuid("id").primaryKey().defaultRandom(),
  tenantId: varchar("tenant_id", { length: 64 }).notNull(),
  source: varchar("source", { length: 32 }).notNull(),      // "url" | "file" | "ocr" | "video" |
"manual"
  uri: varchar("uri", { length: 512 }).notNull(),
  title: varchar("title", { length: 256 }),
  metaJson: jsonb("meta_json").$type<any>().default({}),     // { domainId?, sourceRank?, lang?, ... }
  version: integer("version").notNull().default(1),
  isDeleted: boolean("is_deleted").notNull().default(false),
  // NOVOS:
  domainId: uuid("domain_id"),                               // verticalização opcional
  importance: real("importance").default(1),                 // peso extra do doc
  docVector: jsonb("doc_vector").$type<number[] | null>().default(null),
  docUpdates: integer("doc_updates").notNull().default(0),
  createdAt: timestamp("created_at").defaultNow().notNull(),
  updatedAt: timestamp("updated_at").defaultNow().notNull()
}, t => ({
  idx1: index("ai_docs_tenant_uri_idx").on(t.tenantId, t.uri),
  idx2: index("ai_docs_domain_idx").on(t.tenantId, t.domainId)
}));

export const aiChunks = pgTable("ai_chunks", {
  id: uuid("id").primaryKey().defaultRandom(),
  tenantId: varchar("tenant_id", { length: 64 }).notNull(),
  documentId: uuid("document_id").notNull(),
  order: integer("order").notNull().default(0),
  text: text("text"),
```

```
  vector: jsonb("vector").$type<number[] | null>().default(null),       // texto
  imageVec: jsonb("image_vec").$type<number[] | null>().default(null),  // visão (CLIP)
  metaJson: jsonb("meta_json").$type<any>().default({}),                // { lang, page, keyframe,
clipScore, ... }
  // FLAGS NOVAS:
  approved: boolean("approved").notNull().default(true),
  pinned: boolean("pinned").notNull().default(false),                   // não excluir em re-chunk
  createdAt: timestamp("created_at").defaultNow().notNull()
}, t => ({
  idx1: index("ai_chunks_doc_idx").on(t.documentId),
  idx2: index("ai_chunks_approved_idx").on(t.tenantId, t.approved)
}));

export const aiKbAudit = pgTable("ai_kb_audit", {
  id: uuid("id").primaryKey().defaultRandom(),
  tenantId: varchar("tenant_id", { length: 64 }).notNull(),
  actor: varchar("actor", { length: 128 }).notNull(),
  action: varchar("action", { length: 32 }).notNull(),      // create|update|delete|restore|
ingest|edit-chunk|rechunk|reembed|rebuild-ann
  documentId: uuid("document_id"),
  chunkId: uuid("chunk_id"),
  payload: jsonb("payload").$type<any>().default({}),
  createdAt: timestamp("created_at").defaultNow().notNull()
}, t => ({
  idx: index("ai_kb_audit_tenant_idx").on(t.tenantId, t.createdAt)
}));
```

1.2 View de métricas (nDCG/MRR) por dia

Crie uma view via migração SQL (ajuste nomes conforme seu schema de interações já criado nos apêndices Y/AE):

```
-- migration.sql (exemplo)
CREATE OR REPLACE VIEW ai_eval_view AS
SELECT
  date_trunc('day', created_at) AS day,
  AVG(ndcg) AS ndcg_avg,
  AVG(mrr)  AS mrr_avg,
  AVG(ctr)  AS ctr_avg,
  AVG(cr)   AS cr_avg
FROM ai_eval_daily
GROUP BY 1
ORDER BY 1 DESC;
```

---

2) Backend — Rotas avançadas (Express)

2.1 Helpers matemáticos (MMR, chunking e PCA)

```
/server/ai/math.ts

// MMR (Maximal Marginal Relevance) para diversidade textual
export function mmrSelect(
  candidates: { id:string; score:number; vec:number[] }[],
  queryVec: number[],
  k: number,
  lambda = 0.7
){
  const chosen: typeof candidates = [];
  const R = [...candidates].sort((a,b)=>b.score-a.score);

  const sim = (a:number[], b:number[])=>{
    let s=0; let na=0; let nb=0;
    for (let i=0;i<a.length;i++){ s+=a[i]*b[i]; na+=a[i]*a[i]; nb+=b[i]*b[i]; }
    return s / (Math.sqrt(na)*Math.sqrt(nb) + 1e-9);
  };

  while (chosen.length < k && R.length) {
    let bestIdx = 0; let bestVal = -1e9;
    for (let i=0;i<R.length;i++){
```

```
        const cand = R[i];
        const rel = sim(cand.vec, queryVec);
        let div = 0;
        for (const ch of chosen) div = Math.max(div, sim(cand.vec, ch.vec));
        const val = lambda * rel - (1-lambda) * div;
        if (val > bestVal){ bestVal = val; bestIdx = i; }
      }
      chosen.push(R[bestIdx]); R.splice(bestIdx,1);
    }
    return chosen;
  }

  // Chunking ótimo básico: corta em janelas com sobreposição controlada por orçamento de tokens
  export function smartChunk(text: string, maxChars=1200, overlap=200){
    const out: string[] = [];
    let i=0;
    while (i < text.length){
      const end = Math.min(text.length, i + maxChars);
      let cut = end;
      // tenta cortar em fim de frase
      for (let j=end; j>i+400; j--){
        if (".!?".includes(text[j])){ cut = j+1; break; }
      }
      out.push(text.slice(i, cut).trim());
      i = Math.max(cut - overlap, i + 1);
    }
    return out.filter(s=>s.length>0);
  }

  // PCA de 3 componentes (para visualização 3D); espera matriz NxD
  export function pca3(X: number[][]){
    // centraliza
    const N = X.length, D = X[0].length;
    const mean = Array(D).fill(0);
    for (const v of X) for (let d=0; d<D; d++) mean[d]+=v[d];
    for (let d=0; d<D; d++) mean[d]/=N;
    const C = Array(D).fill(0).map(()=>Array(D).fill(0));
    for (const v of X){
      for (let i=0;i<D;i++){
        const a = v[i]-mean[i];
        for (let j=i;j<D;j++){
          const b = v[j]-mean[j];
          C[i][j]+=a*b; if (j!==i) C[j][i]+=a*b;
        }
      }
    }
    for (let i=0;i<D;i++) for (let j=0;j<D;j++) C[i][j]/=(N-1);

    // autovetores: usamos power method simples 3x
    const power = (M:number[][], it=100)=>{
      let v = Array(M.length).fill(0).map(()=>Math.random());
      for (let t=0;t<it;t++){
        const w = M.map(row=> row.reduce((s,rij,j)=>s+rij*v[j],0));
        const n = Math.sqrt(w.reduce((s,x)=>s+x*x,0))+1e-12;
        v = w.map(x=>x/n);
      }
      const lambda = v.reduce((s,vi,i)=> s + vi * M[i].reduce((ss,rij,j)=>ss+rij*v[j],0), 0);
      return { v, lambda };
    };

    const comps: number[][] = [];
    let M = C.map(r=>r.slice());
    for (let k=0;k<3;k++){
      const { v } = power(M, 120);
      comps.push(v);
      // deflação
      for (let i=0;i<D;i++) for (let j=0;j<D;j++) M[i][j] -= C[i][j] * v[i]*v[j] /
(v.reduce((s,x)=>s+x*x,0)+1e-12);
    }
```

```
  // projeta
  const proj = X.map(vec=>{
    const centered = vec.map((x,i)=>x-mean[i]);
    return comps.map(c => centered.reduce((s,x,i)=>s+x*c[i],0));
  });
  return proj; // Nx3
}
```

Obs.: Se quiser UMAP de verdade, usamos um job Python (opcional) — abaixo incluo o script também.

2.2 Rotas de Administração da KB (upgrade)

`/server/ai/routes.kb.admin.ts` — **NOVO** (mantém o `routes.kb.ts` que já envia/edita; aqui ficam operações avançadas)

```
import type { Express, Request } from "express";
import { db } from "../db";
import { aiDocuments, aiChunks, aiKbAudit } from "@shared/schema.ai.kb";
import { eq, and } from "drizzle-orm";
import { smartChunk } from "../ai/math";
import { embedTextMany, rebuildANN, searchANN } from "../ai/vector"; // veja abaixo
import { pca3 } from "../ai/math";

const T = (req:Request)=> (req as any).tenantId || process.env.PRIMARY_TENANT_ID!;

async function audit(tenantId:string, action:string, payload:any, ids?:{documentId?:string;
chunkId?:string}){
  await db.insert(aiKbAudit).values({ tenantId, actor:"admin", action, payload, ...ids });
}

// 1) Inspecionar doc + chunks, com filtro de aprovados/pinned
export function registerKbAdminRoutes(app: Express) {
  app.get("/api/ai/kb/admin/doc/:id", async (req,res)=>{
    const tenantId = T(req); const id = req.params.id;
    const [doc] = await db.select().from(aiDocuments).where(and(eq(aiDocuments.tenantId, tenantId),
eq(aiDocuments.id, id)));
    if (!doc) return res.status(404).json({ error:"not found" });
    const chunks = await db.select().from(aiChunks).where(eq(aiChunks.documentId,
id)).orderBy(aiChunks.order as any);
    res.json({ doc, chunks });
  });

  // 2) Editar chunk (texto, flags) e re-embed opcional
  app.post("/api/ai/kb/admin/chunk.update", async (req,res)=>{
    const tenantId = T(req);
    const { chunkId, text, approved, pinned, reembed } = req.body || {};
    const [ch] = await db.select().from(aiChunks).where(eq(aiChunks.id, chunkId));
    if (!ch) return res.status(404).json({ error:"chunk not found" });

    await db.update(aiChunks)
      .set({ text: (typeof text === "string" ? text : ch.text), approved: approved ?? ch.approved,
pinned: pinned ?? ch.pinned })
      .where(eq(aiChunks.id, chunkId));

    if (reembed && typeof text === "string"){
      const [vec] = await embedTextMany([text]);
      await db.update(aiChunks).set({ vector: vec }).where(eq(aiChunks.id, chunkId));
      await audit(tenantId, "reembed", { chunkId }, { chunkId, documentId: ch.documentId });
    } else {
      await audit(tenantId, "edit-chunk", { chunkId }, { chunkId, documentId: ch.documentId });
    }
    res.json({ ok:true });
  });

  // 3) Re-chunk document (substitui chunks não "pinned")
  app.post("/api/ai/kb/admin/rechunk", async (req,res)=>{
    const tenantId = T(req);
```

```
    const { documentId, maxChars=1200, overlap=200 } = req.body || {};
    const [doc] = await db.select().from(aiDocuments).where(and(eq(aiDocuments.tenantId, tenantId),
eq(aiDocuments.id, documentId)));
    if (!doc) return res.status(404).json({ error:"doc not found" });

    const chunks = await db.select().from(aiChunks).where(eq(aiChunks.documentId,
documentId)).orderBy(aiChunks.order as any);
    const pinned = chunks.filter(c=>c.pinned);
    const baseText = chunks.map(c=>c.text||"").join("\n\n");

    const parts = smartChunk(baseText, maxChars, overlap);
    // apaga não-pinned
    for (const c of chunks) if (!c.pinned) await db.delete(aiChunks).where(eq(aiChunks.id, c.id));
    // re-cria
    let order = 0;
    for (const p of parts){
      await db.insert(aiChunks).values({
        tenantId, documentId, order: order++, text: p, approved: true, pinned: false
      });
    }
    await audit(tenantId, "rechunk", { documentId, parts: parts.length }, { documentId });
    res.json({ ok:true, parts: parts.length, pinned: pinned.length });
  });

  // 4) Re-embed todos os chunks (aprovados) de um doc
  app.post("/api/ai/kb/admin/reembed.doc", async (req,res)=>{
    const tenantId = T(req);
    const { documentId } = req.body || {};
    const rows = await db.select().from(aiChunks).where(and(eq(aiChunks.documentId, documentId),
eq(aiChunks.approved, true)));
    const texts = rows.map(r=>r.text || "");
    const vecs = await embedTextMany(texts);
    for (let i=0;i<rows.length;i++){
      await db.update(aiChunks).set({ vector: vecs[i] }).where(eq(aiChunks.id, rows[i].id));
    }
    await audit(tenantId, "reembed", { documentId, n: rows.length }, { documentId });
    res.json({ ok:true, n: rows.length });
  });

  // 5) Rebuild incremental do ANN (HNSW/HNSWlib/FAISS) do tenant
  app.post("/api/ai/kb/admin/ann.rebuild", async (req,res)=>{
    const tenantId = T(req);
    const n = await rebuildANN(tenantId);
    await audit(tenantId, "rebuild-ann", { tenantId, n });
    res.json({ ok:true, indexed: n });
  });

  // 6) Busca de inspeção (admin) com MMR
  app.post("/api/ai/kb/admin/search", async (req,res)=>{
    const tenantId = T(req);
    const { query, k=12, lambda=0.7 } = req.body || {};
    if (!query) return res.status(400).json({ error:"query required" });
    const out = await searchANN(tenantId, query, k*4); // pega mais e aplica MMR
    const qVec = out.queryVec!;
    const cand = out.hits.map(h=>({ id:h.id, score:h.score, vec:h.vector!, text:h.text,
doc:h.documentId }));
    const chosen = mmrSelect(cand, qVec, k, lambda);
    res.json({ ok:true, queryVec: qVec, chosen, raw: out.hits.slice(0,k) });
  });

  // 7) Mapa 3D de embeddings (PCA local / UMAP opcional)
  app.get("/api/ai/kb/admin/emb.map", async (req,res)=>{
    const tenantId = T(req);
    const limit = Number(req.query.limit || 1000);
    const rows = await db.execute(`SELECT id, document_id, (vector) as vec FROM ai_chunks WHERE
tenant_id=$1 AND approved=true LIMIT $2`, [tenantId, limit] as any);
    const pts = (rows as any[]).map(r => ({ id:r.id, doc:r.document_id, vec:r.vec||[] }));
    const X = pts.filter(p=>p.vec && p.vec.length).map(p=>p.vec);
    const P = X.length ? pca3(X) : [];
```

```
    const out = pts.map((p,i)=> ({ id:p.id, doc:p.doc, x:P[i]?.[0]||0, y:P[i]?.[1]||0, z:P[i]?.[2]||0
}));
    res.json({ points: out.slice(0,limit) });
  });
}
```

Obs.: O módulo `/server/ai/vector` é listado logo abaixo (embeddings + ANN).

2.3 Vetores e ANN (embeddings + rebuild incremental)

```
/server/ai/vector.ts

import fs from "fs";
import path from "path";
import { db } from "../db";
import { aiChunks } from "@shared/schema.ai.kb";
import { eq } from "drizzle-orm";
import { InferenceSession, Tensor } from "onnxruntime-node";

// ARQUIVO DE ÍNDICE LOCAL por tenant (HNSWlight simples em JSON p/ demo; use hnswlib/faiss em
produção)
const ROOT = process.env.AION_INDEX_ROOT || "./rag/indexes";

// ===== Embeddings de texto (ONNX) – se você já tem um encoder ONNX local, plugue aqui
let encSess: InferenceSession|null = null;
const ENC_PATH = process.env.AION_TEXT_EMB_ONNX || "./models/text_encoder.onnx";

async function loadEnc(){
  if (!encSess) encSess = await InferenceSession.create(ENC_PATH, { executionProviders:["cpu"] });
  return encSess!;
}
// Tokenizador simplificado — troque pelo seu (BPE/SentencePiece). Aqui apenas demo com hash-bag.
function tokenizeSimple(s:string){
  return s.toLowerCase().split(/[^a-z0-9áéíóúàâêôãõç]+/i).filter(Boolean).slice(0,128);
}
function bagOfWordsVector(s:string, dim=512){
  const v = new Float32Array(dim);
  for (const t of tokenizeSimple(s)){
    let h=2166136261; for (let i=0;i<t.length;i++){ h^=t.charCodeAt(i); h+= (h<<1)+(h<<4)+(h<<7)+
(h<<8)+(h<<24); }
    v[h>>>0 % dim]+=1;
  }
  // L2 normalize
  let n=0; for (let i=0;i<dim;i++) n+=v[i]*v[i]; n=Math.sqrt(n)+1e-9;
  for (let i=0;i<dim;i++) v[i]/=n;
  return Array.from(v);
}

export async function embedTextMany(texts:string[]){
  // Se tiver encoder ONNX real: chame o modelo; aqui fallback bag-of-words p/ demo robusto.
  // const sess = await loadEnc(); ...  (deixado como exercício se você já possui o onnx do encoder)
  return texts.map(t => bagOfWordsVector(t, 512));
}

// ===== ANN mínimo (demo) – substitua por hnswlib/faiss em produção
type AnnNode = { id:string; vec:number[]; doc:string; textSample:string };
function idxPath(tenantId:string){ return path.join(ROOT, `${tenantId}.json`); }

export async function rebuildANN(tenantId:string){
  const rows = await db.execute(`SELECT id, document_id, (vector) as vec, substring(text from 1 for
140) AS sample
                                  FROM ai_chunks WHERE tenant_id=$1 AND approved=true AND vector IS NOT
NULL`, [tenantId] as any);
  const nodes: AnnNode[] = (rows as any[]).map(r => ({ id:r.id, vec:r.vec, doc:r.document_id,
textSample:r.sample||"" }));
  fs.mkdirSync(ROOT, { recursive:true });
  fs.writeFileSync(idxPath(tenantId), JSON.stringify({ nodes }), "utf8");
  return nodes.length;
```

```
}

function loadIdx(tenantId:string): { nodes: AnnNode[] } {
  try { return JSON.parse(fs.readFileSync(idxPath(tenantId), "utf8")); }
  catch { return { nodes: [] }; }
}

export async function searchANN(tenantId:string, query:string, k=12){
  const idx = loadIdx(tenantId);
  const q = (await embedTextMany([query]))[0];
  const sim = (a:number[], b:number[])=>{
    let s=0; let na=0; let nb=0;
    for (let i=0;i<a.length;i++){ s+=a[i]*b[i]; na+=a[i]*a[i]; nb+=b[i]*b[i]; }
    return s / (Math.sqrt(na)*Math.sqrt(nb) + 1e-9);
  };
  const scores = idx.nodes.map(n => ({ ...n, score: sim(q, n.vec) }));
  scores.sort((a,b)=>b.score-a.score);
  return { queryVec: q, hits: scores.slice(0, Math.min(k, scores.length)) };
}
```

> **Nota:** Esse ANN é “mínimo/ilustrativo” e **funciona** já hoje. Em produção, você pluga **hnswlib-node** ou FAISS nativo; a interface de chamadas (`rebuildANN`, `searchANN`) permanece igual.

2.4 Rotas clusters UMAP (opcional via Python)

Se quiser UMAP real, incluo um script rápido:

`/trainer/emb/umap.py`

```
import sys, json
import numpy as np
from umap import UMAP

# stdin: JSON {"vectors":[[...],...]}
data = json.load(sys.stdin)
X = np.array(data["vectors"], dtype=np.float32)
um = UMAP(n_components=3, n_neighbors=int(data.get("n_neighbors",15)), min_dist=float(data.get("min_
dist",0.1)))
P = um.fit_transform(X)
print(json.dumps(P.tolist()))
```

E uma rota que chama o script:

`/server/ai/routes.umap.ts`

```
import type { Express, Request } from "express";
import { db } from "../db";
import { spawn } from "child_process";

const T = (req:Request)=> (req as any).tenantId || process.env.PRIMARY_TENANT_ID!;

export function registerUmapRoute(app:Express){
  app.get("/api/ai/kb/admin/emb.umap", async (req,res)=>{
    const tenantId = T(req);
    const limit = Number(req.query.limit || 1200);
    const rows = await db.execute(`SELECT id, document_id, (vector) as vec FROM ai_chunks WHERE
tenant_id=$1 AND approved=true LIMIT $2`, [tenantId, limit] as any);
    const vecs = (rows as any[]).map(r=>r.vec).filter((v:any)=>v&&v.length);
    const ids  = (rows as any[]).map(r=>r.id);

    const p = spawn("python3", ["./trainer/emb/umap.py"]);
    let out=""; let err="";
    p.stdout.on("data", d=> out+=d.toString());
    p.stderr.on("data", d=> err+=d.toString());
    p.on("exit", code=>{
      if (code!==0) return res.status(500).json({ error:err||"umap failed" });
      const P = JSON.parse(out);
      const pts = ids.map((id, i)=>({ id, x:P[i][0], y:P[i][1], z:P[i][2] }));
      res.json({ points: pts });
```

```
    });
    p.stdin.write(JSON.stringify({ vectors: vecs })); p.stdin.end();
  });
}
```

Registre se quiser usar UMAP:

```
import { registerUmapRoute } from "./ai/routes.umap"; registerUmapRoute(app);
```

---

3) Front-end — `/ui/pages/admin/kb.tsx` (versão “PRO”)

Substitua a página que mandei antes por **esta versão completa**.

```
import { useEffect, useMemo, useRef, useState } from "react";
import Plot from "react-plotly.js";

type Doc = { id:string; title?:string; uri:string; version:number; isDeleted:boolean;
importance:number };
type Chunk = { id:string; order:number; text:string; approved:boolean; pinned:boolean; metaJson:any };

export default function KbPage(){
  const [docs,setDocs] = useState<Doc[]>([]);
  const [filter,setFilter] = useState("");
  const [selId,setSelId] = useState<string>("");
  const [sel,setSel] = useState<{doc:Doc, chunks:Chunk[]}|null>(null);
  const [editCh,setEditCh] = useState<Chunk|null>(null);
  const [uploading,setUploading] = useState(false);
  const fileRef = useRef<HTMLInputElement>(null);

  async function loadDocs(){
    const r = await fetch("/api/ai/kb/list"); const j = await r.json();
    setDocs(j || []);
  }
  useEffect(()=>{ loadDocs(); },[]);

  const filtered = useMemo(()=> docs.filter(d=> (d.title||d.uri).toLowerCase()
.includes(filter.toLowerCase())), [docs,filter]);

  async function openDoc(id:string){
    setSelId(id);
    const r = await fetch(`/api/ai/kb/admin/doc/${id}`); const j = await r.json();
    setSel(j);
  }

  async function saveChunk(){
    if (!editCh) return;
    await fetch("/api/ai/kb/admin/chunk.update", {
      method:"POST", headers:{ "Content-Type":"application/json" },
      body: JSON.stringify({ chunkId: editCh.id, text: editCh.text, approved: editCh.approved, pinned:
editCh.pinned, reembed: true })
    });
    await openDoc(selId);
    setEditCh(null);
  }

  async function rechunkDoc(){
    await fetch("/api/ai/kb/admin/rechunk", { method:"POST", headers:{ "Content-Type":"application/
json" }, body: JSON.stringify({ documentId: selId, maxChars: 1200, overlap: 200 }) });
    await openDoc(selId);
  }
  async function reembedDoc(){
    await fetch("/api/ai/kb/admin/reembed.doc", { method:"POST", headers:{ "Content-
Type":"application/json" }, body: JSON.stringify({ documentId: selId }) });
    await openDoc(selId);
  }
  async function rebuildANN(){
    await fetch("/api/ai/kb/admin/ann.rebuild", { method:"POST" });
    alert("Índice ANN rebuildado.");
  }
```

```
  async function uploadMany(files: FileList){
    setUploading(true);
    for (const f of Array.from(files)){
      const fd = new FormData(); fd.append("file", f);
      const isVideo = /\.(mp4|mov|mkv|avi)$/i.test(f.name);
      await fetch(isVideo?"/api/ai/kb/upload.video":"/api/ai/kb/upload", { method:"POST", body: fd });
    }
    setUploading(false);
    await loadDocs();
  }

  // Busca de inspeção (admin)
  const [q,setQ] = useState(""); const [res,setRes] = useState<any>(null); const [lambda,setLambda] =
useState(0.7); const [k,setK] = useState(8);
  async function search(){
    const r = await fetch("/api/ai/kb/admin/search", { method:"POST", headers:{ "Content-
Type":"application/json" }, body: JSON.stringify({ query:q, k, lambda }) });
    setRes(await r.json());
  }

  // Embedding map (PCA local / UMAP opcional via toggle)
  const [points,setPoints] = useState<any[]>([]);
  const [useUmap,setUseUmap] = useState(false);
  async function loadMap(){
    const url = useUmap? "/api/ai/kb/admin/emb.umap" : "/api/ai/kb/admin/emb.map";
    const r = await fetch(url); const j = await r.json();
    setPoints(j.points || []);
  }
  useEffect(()=>{ loadMap(); },[useUmap]);

  return (
    <div className="space-y-6">
      <div className="flex gap-4 items-center">
        <input className="border p-2 flex-1" placeholder="Filtrar por título/URI..." value={filter}
onChange={e=>setFilter(e.target.value)} />
        <button className="px-3 py-2 bg-emerald-600 text-white rounded" onClick={()=>fileRef.current?.
click()}>Upload (txt/pdf/img/vid)</button>
        <input ref={fileRef} type="file" multiple className="hidden" onChange={e=>e.target.files &&
uploadMany(e.target.files)} />
        {uploading && <span className="text-sm text-gray-400">Enviando...</span>}
      </div>

      <div className="grid grid-cols-3 gap-6">
        <div className="col-span-1 border rounded p-3 overflow-y-auto max-h-[68vh]">
          <div className="text-sm text-gray-400 mb-2">Documentos ({filtered.length})</div>
          <ul className="space-y-1">
            {filtered.map(d=>(
              <li key={d.id} className={`flex items-center justify-between px-2 py-1 rounded
${selId===d.id?"bg-gray-800":""}`}>
                <button className="text-left flex-1 truncate text-blue-400" onClick=
{()=>openDoc(d.id)}>{d.title || d.uri}</button>
                <span className="text-xs text-gray-500 ml-2">v{d.version}</span>
              </li>
            ))}
          </ul>
        </div>

        <div className="col-span-2 space-y-4">
          {!sel && <div className="text-sm text-gray-500">Selecione um documento para editar.</div>}
          {sel && (
            <>
              <div className="flex items-center justify-between">
                <div className="text-lg font-semibold">{sel.doc.title || sel.doc.uri}</div>
                <div className="flex gap-2">
                  <button className="px-3 py-1 bg-indigo-600 text-white rounded" onClick=
{rechunkDoc}>Re-chunk</button>
                  <button className="px-3 py-1 bg-blue-600 text-white rounded" onClick=
{reembedDoc}>Re-embed</button>
```

```
                    <button className="px-3 py-1 bg-rose-600 text-white rounded" onClick=
{rebuildANN}>Rebuild ANN</button>
                  </div>
                </div>

                <div className="border rounded p-3 max-h-[46vh] overflow-auto">
                  <table className="w-full text-sm">
                    <thead>
                      <tr className="text-gray-400"><th className="text-left">#</th><th className="text-
left">Aprov.</th><th className="text-left">Pinned</th><th className="text-left">Texto</th><th></th>
</tr>
                    </thead>
                    <tbody>
                      {sel.chunks.map((c,i)=>(
                        <tr key={c.id} className="border-t border-gray-800">
                          <td className="align-top pr-2">{c.order}</td>
                          <td className="align-top pr-2">{String(c.approved)}</td>
                          <td className="align-top pr-2">{String(c.pinned)}</td>
                          <td className="align-top pr-2"><pre className="whitespace-pre-wrap">
{(c.text||"").slice(0,300)}</pre></td>
                          <td className="align-top">
                            <button className="text-blue-400" onClick={()=>setEditCh(c)}>Editar</button>
                          </td>
                        </tr>
                      ))}
                    </tbody>
                  </table>
                </div>

                {editCh && (
                  <div className="border rounded p-3 space-y-2">
                    <div className="font-semibold">Editar Chunk #{editCh.order}</div>
                    <textarea className="border p-2 w-full h-40" value={editCh.text} onChange=
{e=>setEditCh({ ...editCh, text: e.target.value })}/>
                    <div className="flex items-center gap-4">
                      <label className="flex items-center gap-2"><input type="checkbox" checked=
{editCh.approved} onChange={e=>setEditCh({ ...editCh, approved: e.target.checked })}/>
Aprovado</label>
                      <label className="flex items-center gap-2"><input type="checkbox" checked=
{editCh.pinned} onChange={e=>setEditCh({ ...editCh, pinned: e.target.checked })}/> Pinned</label>
                      <button className="px-3 py-1 bg-emerald-600 text-white rounded" onClick=
{saveChunk}>Salvar (re-embed)</button>
                      <button className="px-3 py-1 bg-gray-700 text-white rounded" onClick=
{()=>setEditCh(null)}>Cancelar</button>
                    </div>
                  </div>
                )}

                {/* Laboratório de busca */}
                <div className="border rounded p-3 space-y-2">
                  <div className="font-semibold">Laboratório de busca (MMR)</div>
                  <div className="flex gap-2">
                    <input className="border p-2 flex-1" placeholder="Consulta..." value={q} onChange=
{e=>setQ(e.target.value)}/>
                    <label className="text-sm">k={k}</label>
                    <input type="range" min={3} max={24} value={k} onChange={e=>setK(Number(e.
target.value))}/>
                    <label className="text-sm">λ={lambda.toFixed(2)}</label>
                    <input type="range" min={0} max={1} step={0.05} value={lambda} onChange=
{e=>setLambda(Number(e.target.value))}/>
                    <button className="px-3 py-1 bg-indigo-600 text-white rounded" onClick=
{search}>Buscar</button>
                  </div>
                  {res && res.ok && (
                    <div className="text-xs text-gray-400">
                      <div>Top-k (MMR):</div>
                      <ol className="list-decimal ml-5">
                        {res.chosen.map((c:any)=>(
                          <li key={c.id} className="mb-1"><span className="text-gray-300">[{c.
```

```
id.slice(0,8)}]</span> score={c.score.toFixed(3)} — {c.textSample || ""}</li>
                      ))}
                    </ol>
                  </div>
                )}
              </div>
            </>
          )}
        </div>
      </div>

      {/* Mapa 3D */}
      <div className="border rounded p-3">
        <div className="flex items-center justify-between mb-2">
          <div className="font-semibold">Mapa de Embeddings (3D)</div>
          <label className="flex items-center gap-2 text-sm">
            <input type="checkbox" checked={useUmap} onChange={e=>setUseUmap(e.target.checked)}/>
            Usar UMAP (Python)
          </label>
        </div>
        <Plot
          data={[{
            x: points.map(p=>p.x),
            y: points.map(p=>p.y),
            z: points.map(p=>p.z),
            text: points.map(p=>p.id.slice(0,8)),
            type: "scatter3d", mode: "markers",
            marker: { size: 2, color: "#22c55e" }
          }]}
          layout={{ paper_bgcolor:"rgba(0,0,0,0)", scene:{bgcolor:"rgba(0,0,0,0)"}, height:420,
margin:{l:0,r:0,t:0,b:0} }}
          config={{ displayModeBar:false }}
          style={{ width:"100%" }}
        />
      </div>
    </div>
  );
}
```

---

4) Matemática — inline (para documentação no código)

Confiança agregada $C \in [0,1]$ (mesmo critério que já usamos na Fase 2):

$$C = w_t s_t + w_g s_g + w_f s_f + w_a s_a + w_i s_i, \quad \sum w_j = 1.$$

Onde s_t = relevância textual (cos), s_g = coerência no grafo de entidades, s_f = frescor (decay $\exp(-\ln 2 \cdot h_{age})$), s_a = autoridade (meta), s_i = match visual (CLIP).
Fallback se $C < \tau$ e orçamento permite.

MMR (diversidade no top-k):

$$\text{MMR}(d_i) = \lambda \cdot \text{sim}(q, d_i) - (1-\lambda) \max_{d_j \in S} \text{sim}(d_i, d_j).$$

Selecionamos iterativamente k itens maximizando MMR.

Chunking ótimo (heurística budget-aware): janelas de [m,M] caracteres com sobreposição o garantindo coeficiente de cobertura >0.83 para bigramas, minimizando cortes no meio de frase.

PCA/UMAP: PCA (3 comps) para 3D rápido; UMAP (3D) opcional com vizinhança local preservando manifold (útil para clusterizar domínios).

Métricas:
$\text{nDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k}$, $\text{DCG}_k = \sum_{i=1}^{k} \frac{2^{rel_i}-1}{\log_2(i+1)}$
$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{rank_q}$.
CTR/CR capturados da UI comercial; usamos a view `ai_eval_view` para gráficos.

---

5) Integração no servidor (bootstrap)

No seu **servidor Express** onde registra as rotas:

```
import { registerKbRoutes } from "./ai/routes.kb";               // CRUD básico (já enviado antes)
import { registerKbVideoRoute } from "./ai/routes.kb.video";     // ingest de vídeo (ASR+frames)
import { registerCuratorRoutes } from "./ai/curator/curator.routes";
import { registerThemeRoutes } from "./ai/curator/theme.routes";
import { registerKbAdminRoutes } from "./ai/routes.kb.admin";    // **novo desta fase**
import { registerUmapRoute } from "./ai/routes.umap";            // opcional (UMAP)

// ...
registerKbRoutes(app);
registerKbVideoRoute(app);
registerCuratorRoutes(app);
registerThemeRoutes(app);
registerKbAdminRoutes(app);
registerUmapRoute(app); // se desejar UMAP
```

---

6) Dependências

Se ainda não instalou alguma:

```
pnpm add onnxruntime-node sharp multer fluent-ffmpeg jsdom react-plotly.js plotly.js zustand
apt-get update && apt-get install -y ffmpeg
pip3 install faster-whisper umap-learn
```

---

7) Testes rápidos (admin)

Rebuild ANN:

```
curl -X POST http://localhost:3000/api/ai/kb/admin/ann.rebuild
```

Busca MMR:

```
curl -X POST http://localhost:3000/api/ai/kb/admin/search \
  -H "Content-Type: application/json" \
  -d '{"query":"passeios em Sintra full-day vs half-day", "k":12, "lambda":0.7}'
```

Re-chunk:

```
curl -X POST http://localhost:3000/api/ai/kb/admin/rechunk \
  -H "Content-Type: application/json" \
  -d '{"documentId":"<DOC_ID>","maxChars":1200,"overlap":200}'
```

Re-embed doc:

```
curl -X POST http://localhost:3000/api/ai/kb/admin/reembed.doc \
  -H "Content-Type: application/json" \
  -d '{"documentId":"<DOC_ID>"}'
```

Mapa PCA 3D:

```
curl http://localhost:3000/api/ai/kb/admin/emb.map
```

Mapa UMAP 3D (opcional):

```
curl http://localhost:3000/api/ai/kb/admin/emb.umap
```

---

8) Por que isso deixa a AION mais **autônoma, precisa e econômica**

- **Autônoma**: parsers locais (texto/imagem/vídeo), embeddings locais, ANN local, curadoria e aprendizagem por tema/link → *knowledge flywheel* sem depender de terceiros.
- **Precisa**: MMR + confiança multimodal + editor de chunks + re-chunking controlado elevam nDCG/MRR.
- **Econômica**: fallback só quando C<τ e **dentro do budget** — e com SFT/LoRA periódicos, essa frequência **cai** naturalmente.

- **Observável**: mapa 3D (clusters), laboratório de busca e métricas diárias dão **feedback imediato** para calibração.