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

IA_Autonoma_Parte9

1 mensagem

Fillipe Guerra <fillipe.backup@gmail.com> 28 de outubro de 2025 às 10:11
Para: Fillipe Augusto Gomes Guerra <fillipe182@hotmail.com>, Fillipe Guerra <fillipe.backup@gmail.com>

Fase 2 — Configurações Avançadas, Personas e Verticalização.
Essa fase transforma o painel que você acabou de montar num **centro de controle total da IA AION**, onde você regula *quem ela é*, *como pensa*, *em que áreas atua* e *até quanto pode gastar* com serviços externos.
Tudo continua dentro do mesmo stack Node + Express + React + Tailwind + Drizzle + PostgreSQL.

🧱 Fase 2 — Configurações, Personas e Verticalização

📁 1. Novas tabelas SQL (Drizzle)

Crie `/shared/schema.ai.persona.ts` e `/shared/schema.ai.vertical.ts`.

```
// shared/schema.ai.persona.ts
import { pgTable, uuid, varchar, text, real, jsonb, timestamp } from "drizzle-orm/pg-core";

export const aiPersonas = pgTable("ai_personas", {
  id: uuid("id").primaryKey().defaultRandom(),
  tenantId: varchar("tenant_id", { length:64 }).notNull(),
  name: varchar("name", { length:128 }).notNull(),
  tone: varchar("tone", { length:64 }).default("neutro"),
  description: text("description"),
  temperature: real("temperature").default(0.7),
  styleWeights: jsonb("style_weights").$type<Record<string,number>>().default({}),
  createdAt: timestamp("created_at").defaultNow().notNull(),
  updatedAt: timestamp("updated_at").defaultNow().notNull()
});

// shared/schema.ai.vertical.ts
export const aiDomains = pgTable("ai_domains", {
  id: uuid("id").primaryKey().defaultRandom(),
  tenantId: varchar("tenant_id", { length:64 }).notNull(),
  name: varchar("name", { length:128 }).notNull(),
  description: text("description"),
  weight: real("weight").default(1),
  embeddingsMeta: jsonb("embeddings_meta").$type<any>().default({}),
  createdAt: timestamp("created_at").defaultNow().notNull(),
  updatedAt: timestamp("updated_at").defaultNow().notNull()
});
```

Essas tabelas permitem modelar matematicamente:

θpersona=(tone,T,wstyle)ωdomain=‖ed‖ed

onde T é a temperatura de criatividade e ed são os vetores de embeddings do domínio.

⚙️ 2. Rotas Express para CRUD

Crie `/server/ai/routes.persona.ts` e `/server/ai/routes.domain.ts`.

```
// server/ai/routes.persona.ts
import type { Express, Request } from "express";
import { db } from "../db";
import { aiPersonas } from "@shared/schema.ai.persona";
```

```
import { eq } from "drizzle-orm";

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

export function registerPersonaRoutes(app:Express){
  app.get("/api/ai/personas", async (req,res)=>{
    const r = await db.select().from(aiPersonas).where(eq(aiPersonas.tenantId,T(req)));
    res.json(r);
  });
  app.post("/api/ai/personas", async (req,res)=>{
    const b=req.body||{};
    const [p]=await db.insert(aiPersonas).values({tenantId:T(req),...b}).returning();
    res.json(p);
  });
  app.put("/api/ai/personas/:id", async (req,res)=>{
    await db.update(aiPersonas).set({...req.body,updatedAt:new Date()})
      .where(eq(aiPersonas.id,req.params.id));
    res.json({ok:true});
  });
  app.delete("/api/ai/personas/:id", async (req,res)=>{
    await db.delete(aiPersonas).where(eq(aiPersonas.id,req.params.id));
    res.json({ok:true});
  });
}

// server/ai/routes.domain.ts
import type { Express, Request } from "express";
import { db } from "../db";
import { aiDomains } from "@shared/schema.ai.vertical";
import { eq } from "drizzle-orm";

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

export function registerDomainRoutes(app:Express){
  app.get("/api/ai/domains", async (req,res)=>{
    const r=await db.select().from(aiDomains).where(eq(aiDomains.tenantId,T(req)));
    res.json(r);
  });
  app.post("/api/ai/domains", async (req,res)=>{
    const [d]=await db.insert(aiDomains).values({tenantId:T(req),...req.body}).returning();
    res.json(d);
  });
  app.put("/api/ai/domains/:id", async (req,res)=>{
    await db.update(aiDomains).set({...req.body,updatedAt:new Date()})
      .where(eq(aiDomains.id,req.params.id));
    res.json({ok:true});
  });
  app.delete("/api/ai/domains/:id", async (req,res)=>{
    await db.delete(aiDomains).where(eq(aiDomains.id,req.params.id));
    res.json({ok:true});
  });
}
```

Registre no bootstrap:

```
import { registerPersonaRoutes } from "./ai/routes.persona";
import { registerDomainRoutes } from "./ai/routes.domain";
registerPersonaRoutes(app);
registerDomainRoutes(app);
```

💻 3. Front-end React – Personas

/ui/pages/admin/personas.tsx

```
import { useEffect,useState } from "react";

export default function PersonasPage(){
  const [rows,setRows]=useState<any[]>([]);
```

```
  const [edit,setEdit]=useState<any|null>(null);
  const [form,setForm]=useState({name:"",tone:"neutro",temperature:0.7,description:""});

  async function load(){ const r=await fetch("/api/ai/personas"); setRows(await r.json()); }
  useEffect(()=>{load();},[]);

  async function save(){
    if(edit) await fetch(`/api/ai/personas/${edit.id}`,{method:"PUT",headers:{"Content-
Type":"application/json"},body:JSON.stringify(form)});
    else await fetch("/api/ai/personas",{method:"POST",headers:{"Content-
Type":"application/json"},body:JSON.stringify(form)});
    setForm({name:"",tone:"neutro",temperature:0.7,description:""}); setEdit(null); load();
  }
  async function del(id:string){ await fetch(`/api/ai/personas/${id}`,{method:"DELETE"}); load(); }

  return(
    <div className="space-y-6">
      <h1 className="text-2xl font-bold">Personas da IA</h1>
      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
        <div className="space-y-2">
          <input className="border p-2 w-full" placeholder="Nome" value={form.name} onChange=
{e=>setForm({...form,name:e.target.value})}/>
          <input className="border p-2 w-full" placeholder="Tom (ex.: amigável, formal...)" value=
{form.tone} onChange={e=>setForm({...form,tone:e.target.value})}/>
          <label className="text-sm">Temperatura (0–1): {form.temperature}</label>
          <input type="range" min={0} max={1} step={0.05} value={form.temperature} onChange=
{e=>setForm({...form,temperature:Number(e.target.value)})}/>
          <textarea className="border p-2 w-full h-32" placeholder="Descrição" value=
{form.description} onChange={e=>setForm({...form,description:e.target.value})}/>
          <button className="px-4 py-2 bg-emerald-600 text-white rounded" onClick=
{save}>Salvar</button>
        </div>
        <div className="overflow-y-auto max-h-[70vh] border rounded p-2">
          {rows.map(p=>(
            <div key={p.id} className="border-b border-gray-700 py-2">
              <div className="font-semibold">{p.name}</div>
              <div className="text-sm text-gray-400">{p.tone} · T={p.temperature}</div>
              <div className="text-xs text-gray-500 mb-2">{p.description}</div>
              <button className="text-blue-400 mr-2" onClick={()=>{setEdit(p);
setForm(p);}}>Editar</button>
              <button className="text-rose-400" onClick={()=>del(p.id)}>Excluir</button>
            </div>
          ))}
        </div>
      </div>
    </div>
  );
}
```

💻 4. Front-end React – Verticalização (Domínios)

```
/ui/pages/admin/domains.tsx

import { useEffect,useState } from "react";

export default function DomainsPage(){
  const [rows,setRows]=useState<any[]>([]);
  const [form,setForm]=useState({name:"",description:"",weight:1});
  const [edit,setEdit]=useState<any|null>(null);

  async function load(){ const r=await fetch("/api/ai/domains"); setRows(await r.json()); }
  useEffect(()=>{load();},[]);

  async function save(){
    const method=edit?"PUT":"POST";
    const url=edit?`/api/ai/domains/${edit.id}`:"/api/ai/domains";
    await fetch(url,{method,headers:{"Content-Type":"application/json"},body:JSON.stringify(form)});
    setForm({name:"",description:"",weight:1}); setEdit(null); load();
```

```
  }
  async function del(id:string){ await fetch(`/api/ai/domains/${id}`,{method:"DELETE"}); load(); }

  return(
    <div className="space-y-6">
      <h1 className="text-2xl font-bold">Verticalização / Domínios</h1>
      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
        <div className="space-y-2">
          <input className="border p-2 w-full" placeholder="Nome do domínio" value={form.name}
onChange={e=>setForm({...form,name:e.target.value})}/>
          <textarea className="border p-2 w-full h-32" placeholder="Descrição" value=
{form.description} onChange={e=>setForm({...form,description:e.target.value})}/>
          <label className="text-sm">Peso ω (0 a 2): {form.weight}</label>
          <input type="range" min={0} max={2} step={0.05} value={form.weight} onChange=
{e=>setForm({...form,weight:Number(e.target.value)})}/>
          <button className="px-4 py-2 bg-emerald-600 text-white rounded" onClick=
{save}>Salvar</button>
        </div>
        <div className="overflow-y-auto max-h-[70vh] border rounded p-2">
          {rows.map(d=>(
            <div key={d.id} className="border-b border-gray-700 py-2">
              <div className="font-semibold">{d.name}</div>
              <div className="text-sm text-gray-400">ω = {d.weight}</div>
              <div className="text-xs text-gray-500 mb-2">{d.description}</div>
              <button className="text-blue-400 mr-2" onClick={()=>{setEdit(d);
setForm(d);}}>Editar</button>
              <button className="text-rose-400" onClick={()=>del(d.id)}>Excluir</button>
            </div>
          ))}
        </div>
      </div>
    </div>
  );
}
```

Adicione a rota no Sidebar:

```
{ to: "/admin/domains", label: "Domínios", icon: Layers }
```

🧮 5. Lógica matemática interna (resumo no código)

Cada persona atua como um vetor no espaço de estilo:

soutput=(1−λ)hbase+λ(θpersona·ωdomain)

onde λ é o “fator de influência” (ajustável no painel de Configurações).
Esse bloco será aplicado no próximo módulo de Configuração (“AION Engine Weights”).

📘 6. Próximos passos

Agora o painel já deixa você:

- Criar e editar **personas** com temperatura, tom e estilo;
- Criar **domínios/verticalizações** com pesos;
- Tudo persistente no PostgreSQL e integrado ao tenant.