Backend

Basic Structure

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Server.js

/\*\*

\* PZ Auth+API Backend – Versão 1.6.5 – 2025-09-01 – “FS-Debug+SA-Compat”

\*

\* Diff funcional vs 1.6.4 (“SA-JSON-Direct+Compat”):

\* - 🧰 Logs de Firestore muito mais verbosos (status + corpo da resposta em falhas).

\* - 🩺 Endpoints de diagnóstico (protegidos por DEBUG\_TOKEN):

\* - GET /api/debug/fs-token → valida SA, escopo e expiração do token.

\* - POST /api/debug/fs-write → tenta escrever em auth\_events e retorna id/erro detalhado.

\* - POST /api/debug/ping-fs → agora aceita ?verbose=1 (ou header X-Debug-Verbose: 1)

\* e retorna motivo detalhado em caso de erro.

\* - 🔐 Nenhum segredo é exposto em respostas públicas; detalhes só com DEBUG\_TOKEN.

\* - 🔑 Suporte a FIREBASE\_SERVICE\_ACCOUNT\_JSON “puro” preservado; compat GCP\_\* mantida.

\*/

const express = require('express');

const cors = require('cors');

let cookieParser = null;

try { cookieParser = require('cookie-parser'); }

catch (\_) { console.warn('[BOOT] cookie-parser não encontrado; usando parser leve de header (fallback).'); }

const { OAuth2Client, JWT } = require('google-auth-library');

// fetch robusto (nativo primeiro; fallback ESM)

const fetch = (typeof globalThis.fetch === 'function')

? globalThis.fetch.bind(globalThis)

: ((...args) => import('node-fetch').then(({ default: f }) => f(...args)));

const app = express();

/\* ──────────────────────────────────────────────────────────────

1) Config / Vars

─────────────────────────────────────────────────────────────── \*/

const VERSION = '1.6.5';

const BUILD\_DATE = '2025-09-01';

const PORT = process.env.PORT || 8080;

/\*\* Client IDs aceitos (audiences) \*/

const PRIMARY\_CLIENT\_ID = '270930304722-pbl5cmp53omohrmfkf9dmicutknf3q95.apps.googleusercontent.com';

const CLIENT\_IDS = [

process.env.GOOGLE\_CLIENT\_ID,

...(process.env.GOOGLE\_CLIENT\_IDS ? String(process.env.GOOGLE\_CLIENT\_IDS).split(',') : []),

PRIMARY\_CLIENT\_ID

].map(s => (s || '').trim()).filter(Boolean);

/\*\* Origens permitidas \*/

const allowedOrigins = (process.env.ALLOWED\_ORIGINS ||

[

'https://pzadvisors.com',

'https://www.pzadvisors.com',

'https://auth.pzadvisors.com',

'https://api.pzadvisors.com',

'http://localhost:3000',

'http://127.0.0.1:3000',

'http://127.0.0.1:8080',

'http://127.0.0.1:8081'

].join(',')

).split(',').map(o => o.trim()).filter(Boolean);

const TRACK\_OPEN = (process.env.TRACK\_OPEN || 'false').toLowerCase() === 'true';

const TRACK\_TOKEN = process.env.TRACK\_TOKEN || '';

const DEBUG\_TOKEN = process.env.DEBUG\_TOKEN || '';

app.set('trust proxy', true);

/\* ──────────────────────────────────────────────────────────────

1.1) CORS (no topo)

─────────────────────────────────────────────────────────────── \*/

function isAllowedOrigin(origin) {

if (!origin) return true; // server-to-server

try {

const { hostname, protocol } = new URL(origin.trim());

const host = String(hostname || '').toLowerCase();

const isLocal = protocol === 'http:' && (host === 'localhost' || host === '127.0.0.1');

if (isLocal) return true;

if (host === 'pzadvisors.com' || host === 'www.pzadvisors.com' || host.endsWith('.pzadvisors.com')) return true;

return allowedOrigins.includes(origin.trim());

} catch { return false; }

}

app.use(cors({

origin(origin, cb) {

const ok = isAllowedOrigin(origin);

if (!ok && origin) { try { console.warn(JSON.stringify({ tag: 'cors\_denied', origin })); } catch {} }

return cb(null, ok);

},

methods: ['GET', 'POST', 'OPTIONS', 'HEAD'],

allowedHeaders: [

'Content-Type','Authorization',

'X-PZ-Version','x-pz-version',

'X-Trace-Id','x-trace-id',

'X-Api-Token','x-api-token',

'X-Debug-Token','x-debug-token',

'X-Debug-Verbose','x-debug-verbose'

],

optionsSuccessStatus: 204

}));

/\* ──────────────────────────────────────────────────────────────

1.2) Firestore REST (Service Account) – JSON puro + compat

─────────────────────────────────────────────────────────────── \*/

let SA\_SOURCE = 'split\_env';

let SA\_JSON = null;

const SA\_RAW = process.env.FIREBASE\_SERVICE\_ACCOUNT\_JSON;

if (SA\_RAW) {

try {

SA\_JSON = JSON.parse(SA\_RAW);

SA\_SOURCE = 'env\_json';

} catch (e) {

console.error('[FS] FIREBASE\_SERVICE\_ACCOUNT\_JSON inválido (JSON parse falhou):', e?.message || e);

}

}

// Valores base vindos do JSON (se houver)

let GCP\_PROJECT\_ID = SA\_JSON?.project\_id || '';

let GCP\_SA\_EMAIL = SA\_JSON?.client\_email || '';

let GCP\_SA\_PRIVATE\_KEY = SA\_JSON?.private\_key || '';

// Compatibilidade: se variáveis GCP\_\* existirem, têm prioridade final

GCP\_PROJECT\_ID = process.env.GCP\_PROJECT\_ID || GCP\_PROJECT\_ID || '';

GCP\_SA\_EMAIL = process.env.GCP\_SA\_EMAIL || GCP\_SA\_EMAIL || '';

GCP\_SA\_PRIVATE\_KEY = process.env.GCP\_SA\_PRIVATE\_KEY || GCP\_SA\_PRIVATE\_KEY || '';

// Normaliza quebra de linha quando veio escapada como "\\n"

if (GCP\_SA\_PRIVATE\_KEY) GCP\_SA\_PRIVATE\_KEY = String(GCP\_SA\_PRIVATE\_KEY).replace(/\\n/g, '\n');

const FS\_BASE = `https://firestore.googleapis.com/v1/projects/${GCP\_PROJECT\_ID}/databases/(default)`;

const FS\_SCOPE = 'https://www.googleapis.com/auth/datastore';

// Normaliza caminhos: remove barras extras e prefixo "documents/"

function normalizePath(p) {

return String(p || '').replace(/^\/+|\/+$/g, '').replace(/^documents\//, '');

}

// Gera o "full document name" para APIs que exigem (ex.: transforms)

function makeDocFullName(collectionPath, id) {

const col = normalizePath(collectionPath);

return `${FS\_BASE}/documents/${col}/${encodeURIComponent(String(id))}`;

}

function ensureSA() {

const miss = {

project: !!GCP\_PROJECT\_ID,

email : !!GCP\_SA\_EMAIL,

key : !!GCP\_SA\_PRIVATE\_KEY && GCP\_SA\_PRIVATE\_KEY.includes('BEGIN PRIVATE KEY')

};

if (!miss.project || !miss.email || !miss.key) {

const msg = `[FS] Service Account config incompleta: ${JSON.stringify(miss)} (source=${SA\_SOURCE})`;

console.error(msg);

throw Object.assign(new Error('sa\_not\_configured'), { code: 'sa\_not\_configured', meta: { ...miss, source: SA\_SOURCE } });

}

}

// Auth com retorno expandido (inclui expiry\_date)

async function getSAClient() {

ensureSA();

return new JWT({ email: GCP\_SA\_EMAIL, key: GCP\_SA\_PRIVATE\_KEY, scopes: [FS\_SCOPE] });

}

async function getSAToken() {

const client = await getSAClient();

const { token, res, expiry\_date } = await client.authorize();

// Guardamos info útil para diagnósticos

return { token, expiry\_date: expiry\_date || null };

}

// Converte JS → Firestore Value

function fsVal(v) {

if (v === null || v === undefined) return { nullValue: null };

if (typeof v === 'string') return { stringValue: v };

if (typeof v === 'boolean') return { booleanValue: v };

if (typeof v === 'number') return Number.isInteger(v) ? { integerValue: String(v) } : { doubleValue: v };

if (v instanceof Date) return { timestampValue: v.toISOString() };

if (typeof v === 'object') {

const fields = {}; for (const k of Object.keys(v)) fields[k] = fsVal(v[k]);

return { mapValue: { fields } };

}

return { stringValue: String(v) };

}

/\* Helpers de Firestore com logs detalhados \*/

async function fsCreate(collectionPath, data) {

const { token } = await getSAToken();

const col = normalizePath(collectionPath);

const url = `${FS\_BASE}/documents/${col}`;

const body = { fields: {} };

for (const k of Object.keys(data)) body.fields[k] = fsVal(data[k]);

const r = await fetch(url, {

method: 'POST',

headers: { Authorization: `Bearer ${token}`, 'Content-Type': 'application/json' },

body: JSON.stringify(body)

});

if (!r.ok) {

const t = await safeText(r);

const errMsg = `fsCreate ${col} failed: ${r.status} ${t}`;

console.error('[FS][CREATE][ERR]', errMsg);

throw new Error(errMsg);

}

return r.json();

}

async function fsUpsert(docPath, data) {

const { token } = await getSAToken();

const doc = normalizePath(docPath);

const url = new URL(`${FS\_BASE}/documents/${doc}`);

url.searchParams.set('allowMissing', 'true');

const fields = Object.keys(data);

fields.forEach(f => url.searchParams.append('updateMask.fieldPaths', f));

const body = { fields: {} };

for (const k of fields) body.fields[k] = fsVal(data[k]);

const r = await fetch(url.toString(), {

method: 'PATCH',

headers: { Authorization: `Bearer ${token}`, 'Content-Type': 'application/json' },

body: JSON.stringify(body)

});

if (!r.ok) {

const t = await safeText(r);

const errMsg = `fsUpsert ${doc} failed: ${r.status} ${t}`;

console.error('[FS][UPSERT][ERR]', errMsg);

throw new Error(errMsg);

}

return r.json();

}

async function fsCommitTransforms(docFullName, fieldPaths = []) {

if (!fieldPaths.length) return;

const { token } = await getSAToken();

const url = `${FS\_BASE}/documents:commit`;

const writes = [{

transform: {

document: docFullName,

fieldTransforms: fieldPaths.map(fp => ({ fieldPath: fp, setToServerValue: 'REQUEST\_TIME' }))

}

}];

const r = await fetch(url, {

method: 'POST',

headers: { Authorization: `Bearer ${token}`, 'Content-Type': 'application/json' },

body: JSON.stringify({ writes })

});

if (!r.ok) {

const t = await safeText(r);

const errMsg = `fsCommitTransforms failed: ${r.status} ${t}`;

console.error('[FS][COMMIT][ERR]', errMsg);

throw new Error(errMsg);

}

return r.json();

}

async function safeText(resp) {

try { return await resp.text(); } catch { return '(no-body)'; }

}

/\* ──────────────────────────────────────────────────────────────

2) Middlewares

─────────────────────────────────────────────────────────────── \*/

if (cookieParser) {

app.use(cookieParser());

} else {

// Fallback: parser simples de cookies (g\_csrf\_token)

app.use((req, \_res, next) => {

const raw = req.headers.cookie || '';

req.cookies = {};

raw.split(';').forEach(p => {

const [k, ...v] = p.split('=');

if (!k) return;

req.cookies[k.trim()] = decodeURIComponent((v.join('=') || '').trim());

});

next();

});

}

// Body parsers (robustos)

app.use(express.json({ limit: '2mb', type: ['application/json', 'application/\*+json'] }));

app.use(express.urlencoded({ extended: true }));

// Fallback: se body veio como string/sem Content-Type, tenta JSON

app.use((req, \_res, next) => {

if (req.method === 'GET' || req.method === 'HEAD' || req.method === 'OPTIONS') return next();

if (req.body && typeof req.body === 'object' && !Buffer.isBuffer(req.body)) return next();

let data = '';

req.on('data', c => { data += c; if (data.length > 2 \* 1024 \* 1024) req.destroy(); }); // 2MB

req.on('end', () => {

if (!req.body || typeof req.body !== 'object') {

if (data && /^[\s{\[]/.test(data)) {

try { req.body = JSON.parse(data); } catch { /\* ignore \*/ }

}

}

next();

});

});

// Headers padrão + trace id

app.use((req, res, next) => {

const rid = req.headers['x-trace-id'] || `${Date.now()}-${Math.random().toString(36).slice(2,8)}`;

req.rid = rid;

res.setHeader('X-Trace-Id', rid);

res.setHeader('X-PZ-Version', `PZ Auth+API Backend v${VERSION} (${BUILD\_DATE})`);

res.setHeader('Vary', 'Origin, Access-Control-Request-Method, Access-Control-Request-Headers');

next();

});

// Logger

app.use((req, res, next) => {

const t0 = Date.now();

res.on('finish', () => {

try {

console.log(JSON.stringify({

rid: req.rid, method: req.method, path: req.path,

status: res.statusCode, ms: Date.now() - t0, origin: req.headers.origin || null

}));

} catch {}

});

next();

});

/\* ──────────────────────────────────────────────────────────────

3) Rotas básicas / Health / Versão

─────────────────────────────────────────────────────────────── \*/

app.get('/', (\_req, res) => res.status(200).send('🚀 PZ Auth+API Backend ativo. Use /healthz, /api/version ou /auth/google.'));

app.get('/healthz', (\_req, res) => res.status(200).json({ ok:true, uptime:process.uptime(), ts:new Date().toISOString() }));

app.head('/healthz', (\_req, res) => res.sendStatus(200));

app.get('/api/healthz', (\_req, res) => res.status(200).json({ ok:true, uptime:process.uptime(), ts:new Date().toISOString() }));

app.head('/api/healthz', (\_req, res) => res.sendStatus(200));

app.get('/api/version', (\_req, res) => {

res.status(200).json({

service: 'PZ Auth+API Backend',

version: VERSION,

build\_date: BUILD\_DATE,

client\_ids\_configured: CLIENT\_IDS,

cors\_allowed\_origins: allowedOrigins,

wildcard\_pzadvisors: '\*.pzadvisors.com',

track\_open: TRACK\_OPEN,

has\_track\_token: Boolean(TRACK\_TOKEN),

debug\_ping\_enabled: Boolean(DEBUG\_TOKEN),

firestore\_auth\_mode: 'REST+ServiceAccount',

has\_cookie\_parser: Boolean(cookieParser),

project\_id: GCP\_PROJECT\_ID,

sa\_email: !!GCP\_SA\_EMAIL,

sa\_source: SA\_SOURCE

});

});

// CORS diag

app.get('/api/cors-check', (req, res) => {

const origin = req.headers.origin || null;

return res.status(200).json({

ok:true, rid:req.rid, origin,

allowed: isAllowedOrigin(origin),

ua: req.headers['user-agent'] || null,

ts: new Date().toISOString()

});

});

/\* ──────────────────────────────────────────────────────────────

3.1) Debug – Credenciais SA / Firestore (protegido)

─────────────────────────────────────────────────────────────── \*/

function assertDebugAccess(req, res) {

const tok = req.headers['x-debug-token'] || req.headers['X-Debug-Token'];

if (!DEBUG\_TOKEN || tok !== DEBUG\_TOKEN) {

res.status(403).json({ ok:false, error:'forbidden' });

return false;

}

return true;

}

app.get('/api/debug/env-has-sa', (\_req, res) => {

const hasProj = !!GCP\_PROJECT\_ID;

const hasEmail = !!GCP\_SA\_EMAIL;

const hasKey = !!GCP\_SA\_PRIVATE\_KEY && GCP\_SA\_PRIVATE\_KEY.includes('BEGIN PRIVATE KEY');

res.status(200).json({ hasProj, hasEmail, hasKey, sa\_source: SA\_SOURCE });

});

// Novo: detalha token e expiração

app.get('/api/debug/fs-token', async (req, res) => {

if (!assertDebugAccess(req, res)) return;

try {

const client = await getSAClient();

const { token, res: rawRes, expiry\_date } = await client.authorize();

const expires\_in\_s = expiry\_date ? Math.max(0, Math.floor((expiry\_date - Date.now())/1000)) : null;

res.status(200).json({

ok: true,

project\_id: GCP\_PROJECT\_ID,

sa\_email: GCP\_SA\_EMAIL ? true : false,

scope: FS\_SCOPE,

expiry\_date,

expires\_in\_s

});

} catch (e) {

res.status(500).json({ ok:false, error: e.message || String(e) });

}

});

// Novo: tentativa de escrita com retorno detalhado

app.post('/api/debug/fs-write', async (req, res) => {

if (!assertDebugAccess(req, res)) return;

try {

const r = await fsCreate('auth\_events', {

type: 'debug\_write',

rid: req.rid,

note: (req.body && req.body.note) || 'manual',

ts: new Date()

});

const id = (r.name || '').split('/').pop() || null;

res.status(200).json({ ok:true, rid:req.rid, doc:id });

} catch (e) {

res.status(500).json({ ok:false, rid:req.rid, error: e.message || String(e) });

}

});

/\* ──────────────────────────────────────────────────────────────

4) Google OAuth – One Tap

─────────────────────────────────────────────────────────────── \*/

const oauthClient = new OAuth2Client(CLIENT\_IDS[0] || PRIMARY\_CLIENT\_ID);

app.options('/auth/google', (\_req, res) => res.sendStatus(204));

app.options('/api/auth/google', (\_req, res) => res.sendStatus(204));

function decodeJwtPayload(idToken) {

try {

const base64 = idToken.split('.')[1].replace(/-/g,'+').replace(/\_/g,'/');

const json = Buffer.from(base64, 'base64').toString('utf8');

return JSON.parse(json);

} catch { return null; }

}

async function handleAuthGoogle(req, res) {

try {

const ct = (req.headers['content-type'] || '').toLowerCase();

const body = req.body || {};

const credential = (typeof body.credential === 'string' && body.credential) ||

(typeof body.id\_token === 'string' && body.id\_token) || null;

const context = body.context;

console.log(JSON.stringify({ route:'/auth/google', rid:req.rid, content\_type:ct, body\_keys:Object.keys(body||{}) }));

if (!credential) {

console.error(JSON.stringify({ route:'/auth/google', rid:req.rid, ok:false, error:'missing\_credential' }));

return res.status(400).json({ error:'missing\_credential' });

}

// CSRF (se usar login\_uri)

if (('g\_csrf\_token' in body) || (req.cookies && 'g\_csrf\_token' in req.cookies)) {

const csrfCookie = req.cookies?.g\_csrf\_token;

const csrfBody = body?.g\_csrf\_token;

if (!csrfCookie || !csrfBody || csrfCookie !== csrfBody) {

console.error(JSON.stringify({ route:'/auth/google', rid:req.rid, ok:false, error:'csrf\_mismatch' }));

return res.status(400).json({ error:'csrf\_mismatch' });

}

}

res.set({ 'Cache-Control': 'no-store, no-cache, must-revalidate, private', Pragma: 'no-cache', Expires: '0' });

// Log leve do token (aud/iss/exp)

const decoded = decodeJwtPayload(credential) || {};

if (decoded && decoded.aud) {

console.log(JSON.stringify({

route: '/auth/google', rid: req.rid,

token\_aud: decoded.aud, token\_iss: decoded.iss || null, token\_exp: decoded.exp || null

}));

}

// Verificação (assinatura + audiences)

const ticket = await oauthClient.verifyIdToken({ idToken: credential, audience: CLIENT\_IDS });

const payload = ticket.getPayload();

if (!payload) {

console.error(JSON.stringify({ route:'/auth/google', rid:req.rid, ok:false, error:'invalid\_token\_empty\_payload' }));

return res.status(401).json({ error:'invalid\_token' });

}

if (payload.iss && !/accounts\.google\.com$/.test(String(payload.iss))) {

console.warn(JSON.stringify({ route:'/auth/google', rid:req.rid, warn:'unexpected\_issuer', iss: payload.iss }));

}

const { sub, email, name, picture, email\_verified } = payload;

const user\_id = String(sub);

// Upsert user (REST) – merge + REQUEST\_TIME em last\_seen/updated\_at

try {

await fsUpsert(`users/${encodeURIComponent(user\_id)}`, {

user\_id, sub, email: email || null, name: name || null, picture: picture || null,

email\_verified: !!email\_verified

});

const docName = makeDocFullName('users', user\_id);

await fsCommitTransforms(docName, ['last\_seen', 'updated\_at']);

} catch (e) {

console.error(JSON.stringify({ route:'/auth/google', rid:req.rid, ok:true, warn:'firestore\_upsert\_failed', error:e.message || String(e) }));

}

// Event (não bloqueante)

try {

await fsCreate('auth\_events', {

type: 'auth\_google',

rid: req.rid,

trace\_id: req.headers['x-trace-id'] || null,

user\_id,

email: email || null,

context: context || null,

ua: req.headers['user-agent'] || null,

origin: req.headers.origin || null,

ts: new Date()

});

} catch (e) {

console.error(JSON.stringify({ route:'/auth/google', rid:req.rid, warn:'auth\_event\_log\_failed', error:e.message || String(e) }));

}

console.log(JSON.stringify({ route:'/auth/google', rid:req.rid, ok:true, user\_id, email: email || null }));

return res.status(200).json({ user\_id, email: email || null, name: name || null, picture: picture || null });

} catch (err) {

const msg = err?.message || String(err || '');

let code = 'auth\_failed';

if (/Wrong recipient|audience/.test(msg)) code = 'audience\_mismatch';

if (/expired/i.test(msg)) code = 'token\_expired';

if (/invalid/i.test(msg)) code = 'invalid\_token';

console.error(JSON.stringify({ route:'/auth/google', rid:req.rid, ok:false, error: msg, code }));

return res.status(401).json({ error: code });

}

}

app.post('/auth/google', handleAuthGoogle);

app.post('/api/auth/google', handleAuthGoogle);

/\* ──────────────────────────────────────────────────────────────

5) Endpoints auxiliares

─────────────────────────────────────────────────────────────── \*/

app.post('/api/echo', (req, res) => {

res.set({ 'Cache-Control':'no-store, no-cache, must-revalidate, private', Pragma:'no-cache', Expires:'0' });

return res.status(200).json({ ok:true, rid:req.rid, echo:req.body || null, ts:new Date().toISOString() });

});

app.post('/api/track', async (req, res) => {

try {

if (!TRACK\_OPEN) {

const tok = req.headers['x-api-token'] || req.headers['X-Api-Token'];

if (!TRACK\_TOKEN || tok !== TRACK\_TOKEN) return res.status(403).json({ ok:false, error:'forbidden' });

}

const { event, payload } = req.body || {};

if (!event || typeof event !== 'string') return res.status(400).json({ ok:false, error:'missing\_event' });

await fsCreate('auth\_events', {

type: event,

rid: req.rid,

payload: payload || null,

origin: req.headers.origin || null,

ua: req.headers['user-agent'] || null,

ts: new Date()

});

return res.status(200).json({ ok:true, rid:req.rid });

} catch (e) {

console.error(JSON.stringify({ route:'/api/track', rid:req.rid, error:e.message || String(e) }));

return res.status(500).json({ ok:false, error:'track\_failed' });

}

});

/\* atualizado: aceita verbose=1 para detalhes do erro \*/

app.post('/api/debug/ping-fs', async (req, res) => {

const verbose = (String(req.query.verbose || req.headers['x-debug-verbose'] || '') === '1');

try {

const tok = req.headers['x-debug-token'] || req.headers['X-Debug-Token'];

if (!DEBUG\_TOKEN || tok !== DEBUG\_TOKEN) return res.status(403).json({ ok:false, error:'forbidden' });

const r = await fsCreate('auth\_events', {

type: 'debug\_ping\_fs',

rid: req.rid,

origin: req.headers.origin || null,

ts: new Date()

});

const id = (r.name || '').split('/').pop() || null;

return res.status(200).json({ ok:true, rid:req.rid, doc: id });

} catch (e) {

const payload = { route:'/api/debug/ping-fs', rid:req.rid, error:e.message || String(e) };

console.error(JSON.stringify(payload));

if (e.code === 'sa\_not\_configured') return res.status(503).json({ ok:false, error:'sa\_not\_configured', meta:e.meta });

return res.status(500).json(verbose ? { ok:false, ...payload } : { ok:false, error:'ping\_failed' });

}

});

/\* ──────────────────────────────────────────────────────────────

6) Erros globais

─────────────────────────────────────────────────────────────── \*/

process.on('unhandledRejection', (reason) => { console.error('[UNHANDLED\_REJECTION]', reason); });

process.on('uncaughtException', (err) => { console.error('[UNCAUGHT\_EXCEPTION]', err); });

/\* ──────────────────────────────────────────────────────────────

7) Start

─────────────────────────────────────────────────────────────── \*/

app.listen(PORT, () => {

console.log('──────────────────────────────────────────────────────────────');

console.log(`✅ Server UP on port ${PORT}`);

console.log(`📦 Version: v${VERSION} (${BUILD\_DATE})`);

console.log('🔧 Vars:');

console.log(' CLIENT\_IDS :', CLIENT\_IDS);

console.log(' ALLOWED\_ORIGINS :', allowedOrigins);

console.log(' wildcard\_pzadvisors : \*.pzadvisors.com');

console.log(' TRACK\_OPEN :', TRACK\_OPEN);

console.log(' TRACK\_TOKEN set :', Boolean(TRACK\_TOKEN));

console.log(' DEBUG\_TOKEN set :', Boolean(DEBUG\_TOKEN));

console.log(' FIRESTORE auth mode : REST+ServiceAccount');

console.log(' PROJECT\_ID :', GCP\_PROJECT\_ID);

console.log(' SA\_EMAIL set :', !!GCP\_SA\_EMAIL, `(source=${SA\_SOURCE})`);

console.log(' HAS cookie-parser :', Boolean(cookieParser));

console.log(' NODE\_ENV :', process.env.NODE\_ENV || '(not set)');

console.log('──────────────────────────────────────────────────────────────');

});

package.json

{

"name": "pz-authapi-backend-v1.6.5-2025-09-01-railway-sa-json-direct-compat",

"version": "1.6.5-2025-08-31",

"description": "PZ Auth+API Backend – v1.6.5 – 2025-08-31 – SA-JSON-Direct+Compat (suporte nativo a FIREBASE\_SERVICE\_ACCOUNT\_JSON “puro”, compat c/ variáveis GCP\_\*; Google One Tap AUTH; Firestore via REST/Service Account; fetch nativo com fallback; CORS no topo; fallback de JSON).",

"main": "server.js",

"scripts": {

"start": "node server.js",

"dev": "PORT=8080 node server.js",

"dev:win": "set PORT=8080&& node server.js"

},

"engines": {

"node": ">=18"

},

"author": "",

"license": "ISC",

"dependencies": {

"cookie-parser": "^1.4.6",

"cors": "^2.8.5",

"express": "^4.18.3",

"google-auth-library": "^10.2.0",

"node-fetch": "^3.3.2"

}

}

Railway

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

ALLOWED\_ORIGINS  
https://pzadvisors.com,https://api.pzadvisors.com,http://localhost:3000,https://www.pzadvisors.com,https://auth.pzadvisors.com

DEBUG\_TOKEN

4bb13d5922104fd1aa16d824628284db

FIREBASE\_SERVICE\_ACCOUNT\_JSON

{

"type": "service\_account",

"project\_id": "pz-tracking",

"private\_key\_id": "67b10f54dc4c86024eaede9b765d839bf072a95d",

"private\_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDd8ck4S0Ai/kkJ\nj2uC9wvYkT80k/9gkk1obPBMKe16QEMqaBmNmP/9LncSZUVTbbEeIkM7OFRV5Mqz\nYndDtMdVrK2whos+eYlK7cytBveBwL0zfqkfsi2PSBjZ3yx4gyaswBm5ZfqKV6Qc\nVp2BcPeQM5KEeNYzDjLyW4WnVkJluG8adBRkOK/aQ/wdk17xjC6UX7qvpMeqEpHm\n/i48Ggf+Dq9WXunPkgZD3KmtYfYRfgfqnx1/aHxS4XIoBwgMB/rgtlqQbMiMihAR\nYhip2sg6tHdWENSGv1A/nWKJ3IPanJ9H+Ela3Jr9JABuVNH4qvm72ERSj6K8sIlZ\n1gZ5xzivAgMBAAECggEAUFbUes8qgUBA2fgTd83LePJiybqE58Utxg4hEcDfmLAo\nX/JcxZfLmi82pO8JoECFq53Fbmteg9UU1she2aBXw5iFKvAY+V1vKDNaMoviDhLh\nqgFAYRvTSBlV/Y++fkUjO8cxyJ20il1dJ3DnBNMKmiVA9Tj0eHi8UmR0xfUAhecH\nVtISav19CUr4nzTjALfm7bZ0W08Mu+kyMUizw5EHpEDfpz7e6UkUWgAgwX+KnR0L\nSW6dKIrwIKibsH492NjtvCshHEYRsd+SHuuxhD2qVvPncuP8YDNsOogVI5XcVIHN\ngU3s9JqRv7A6Ne9Rf4bnje2x7GCjs5YfHrc3yOsWGQKBgQD9LUmMUTIWJjNffZDp\nZ+Hs/b39youIR2zDpG66rDrryaFV0j5M4j22s1/yMko5xgU6K8/q75+aVRma4uBj\nJ8bC3JAm6AasutwaZzw2iSddL95BGQJnU8vj2R+HCdPseY2SkucUd2TdpBYB/GfY\n/s9zBi6wB+J5o93wgFqzvTn/pQKBgQDga1fn8VuzG7blArYb6QGodxiBVzugwUFI\nz4byihFtZ9d7dsim8IuTl2Nsu+8VdJ9bPBBnBuAhb+7WKxdGFDwkAhx0TbqofnRu\nhVzcHiQw3DdRU6B7l62hy7h9WbE/hh1Ieoqzu8sm5NrozpADKiIRSArlYww1bctp\nzTVef1smwwKBgQC47GRFPmVkMkDh5bh8E6m8zdYBUvk58FmsSSWr6fMbmGnkYq2o\nwLOtCDXzlPIMYuwcgjXa923Pb4tFS6P0KPCQEkSkdmOxtMbh7jfX6bQ756PiTp8n\n09ggM85Id1A48MwMnIp2s4KUNa7xgmhj4jKqZICjiQ7OJHgOm+IkWPnNFQKBgA3J\nSlsrTpDQMQocgodwZvOsbUr0yurzNEl+xjF8Hdn0uR4Si/0FZe2pLShu2x8cTPXT\n2MYbfAVR1QA/IWzirVcg8hKK+96B9jEb7Dpxd5Ow040hcAhnpvj7hNfefK8EVCQx\nsvebn71B7djDZ8FqAzHAlTrGPOfDWO2k95RYiob1AoGAUjvdJQoLJxN2u/Z9gnEN\nT1xotdMDrJMRzG48pUJnvTvBYpTXaDp3//fnE1d7tINT3MNSno6UT5X5/9uE3gIO\nhJJe7AAghNasNyD3XSOwdCGMe4wpyCETuqs38V5/hSqnrBu6D6juiggpVD6miy7E\nnG9bxRifLv8Z5ANY5vVVfmo=\n-----END PRIVATE KEY-----\n",

"client\_email": "firebase-adminsdk-fbsvc@pz-tracking.iam.gserviceaccount.com",

"client\_id": "105309635317365581578",

"auth\_uri": "https://accounts.google.com/o/oauth2/auth",

"token\_uri": "https://oauth2.googleapis.com/token",

"auth\_provider\_x509\_cert\_url": "https://www.googleapis.com/oauth2/v1/certs",

"client\_x509\_cert\_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-fbsvc%40pz-tracking.iam.gserviceaccount.com",

"universe\_domain": "googleapis.com"

}

GCP\_PROJECT\_ID

pz-tracking

GCP\_SA\_EMAIL

firebase-adminsdk-fbsvc@pz-tracking.iam.gserviceaccount.com

GCP\_SA\_PRIVATE\_KEY

-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDd8ck4S0Ai/kkJ\nj2uC9wvYkT80k/9gkk1obPBMKe16QEMqaBmNmP/9LncSZUVTbbEeIkM7OFRV5Mqz\nYndDtMdVrK2whos+eYlK7cytBveBwL0zfqkfsi2PSBjZ3yx4gyaswBm5ZfqKV6Qc\nVp2BcPeQM5KEeNYzDjLyW4WnVkJluG8adBRkOK/aQ/wdk17xjC6UX7qvpMeqEpHm\n/i48Ggf+Dq9WXunPkgZD3KmtYfYRfgfqnx1/aHxS4XIoBwgMB/rgtlqQbMiMihAR\nYhip2sg6tHdWENSGv1A/nWKJ3IPanJ9H+Ela3Jr9JABuVNH4qvm72ERSj6K8sIlZ\n1gZ5xzivAgMBAAECggEAUFbUes8qgUBA2fgTd83LePJiybqE58Utxg4hEcDfmLAo\nX/JcxZfLmi82pO8JoECFq53Fbmteg9UU1she2aBXw5iFKvAY+V1vKDNaMoviDhLh\nqgFAYRvTSBlV/Y++fkUjO8cxyJ20il1dJ3DnBNMKmiVA9Tj0eHi8UmR0xfUAhecH\nVtISav19CUr4nzTjALfm7bZ0W08Mu+kyMUizw5EHpEDfpz7e6UkUWgAgwX+KnR0L\nSW6dKIrwIKibsH492NjtvCshHEYRsd+SHuuxhD2qVvPncuP8YDNsOogVI5XcVIHN\ngU3s9JqRv7A6Ne9Rf4bnje2x7GCjs5YfHrc3yOsWGQKBgQD9LUmMUTIWJjNffZDp\nZ+Hs/b39youIR2zDpG66rDrryaFV0j5M4j22s1/yMko5xgU6K8/q75+aVRma4uBj\nJ8bC3JAm6AasutwaZzw2iSddL95BGQJnU8vj2R+HCdPseY2SkucUd2TdpBYB/GfY\n/s9zBi6wB+J5o93wgFqzvTn/pQKBgQDga1fn8VuzG7blArYb6QGodxiBVzugwUFI\nz4byihFtZ9d7dsim8IuTl2Nsu+8VdJ9bPBBnBuAhb+7WKxdGFDwkAhx0TbqofnRu\nhVzcHiQw3DdRU6B7l62hy7h9WbE/hh1Ieoqzu8sm5NrozpADKiIRSArlYww1bctp\nzTVef1smwwKBgQC47GRFPmVkMkDh5bh8E6m8zdYBUvk58FmsSSWr6fMbmGnkYq2o\nwLOtCDXzlPIMYuwcgjXa923Pb4tFS6P0KPCQEkSkdmOxtMbh7jfX6bQ756PiTp8n\n09ggM85Id1A48MwMnIp2s4KUNa7xgmhj4jKqZICjiQ7OJHgOm+IkWPnNFQKBgA3J\nSlsrTpDQMQocgodwZvOsbUr0yurzNEl+xjF8Hdn0uR4Si/0FZe2pLShu2x8cTPXT\n2MYbfAVR1QA/IWzirVcg8hKK+96B9jEb7Dpxd5Ow040hcAhnpvj7hNfefK8EVCQx\nsvebn71B7djDZ8FqAzHAlTrGPOfDWO2k95RYiob1AoGAUjvdJQoLJxN2u/Z9gnEN\nT1xotdMDrJMRzG48pUJnvTvBYpTXaDp3//fnE1d7tINT3MNSno6UT5X5/9uE3gIO\nhJJe7AAghNasNyD3XSOwdCGMe4wpyCETuqs38V5/hSqnrBu6D6juiggpVD6miy7E\nnG9bxRifLv8Z5ANY5vVVfmo=\n-----END PRIVATE KEY-----

GOOGLE\_CLIENT\_ID

270930304722-pbl5cmp53omohrmfkf9dmicutknf3q95.apps.googleusercontent.com

TRACK\_OPEN

true

TRACK\_TOKEN

45015ad973d24607b966469e4d5f72a3

Google Cloud

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Client ID

270930304722-pbl5cmp53omohrmfkf9dmicutknf3q95.apps.googleusercontent.com

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Firebase

Interface gráfica do usuário, Aplicativo, Word

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

sa.json

{

"type": "service\_account",

"project\_id": "pz-tracking",

"private\_key\_id": "67b10f54dc4c86024eaede9b765d839bf072a95d",

"private\_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDd8ck4S0Ai/kkJ\nj2uC9wvYkT80k/9gkk1obPBMKe16QEMqaBmNmP/9LncSZUVTbbEeIkM7OFRV5Mqz\nYndDtMdVrK2whos+eYlK7cytBveBwL0zfqkfsi2PSBjZ3yx4gyaswBm5ZfqKV6Qc\nVp2BcPeQM5KEeNYzDjLyW4WnVkJluG8adBRkOK/aQ/wdk17xjC6UX7qvpMeqEpHm\n/i48Ggf+Dq9WXunPkgZD3KmtYfYRfgfqnx1/aHxS4XIoBwgMB/rgtlqQbMiMihAR\nYhip2sg6tHdWENSGv1A/nWKJ3IPanJ9H+Ela3Jr9JABuVNH4qvm72ERSj6K8sIlZ\n1gZ5xzivAgMBAAECggEAUFbUes8qgUBA2fgTd83LePJiybqE58Utxg4hEcDfmLAo\nX/JcxZfLmi82pO8JoECFq53Fbmteg9UU1she2aBXw5iFKvAY+V1vKDNaMoviDhLh\nqgFAYRvTSBlV/Y++fkUjO8cxyJ20il1dJ3DnBNMKmiVA9Tj0eHi8UmR0xfUAhecH\nVtISav19CUr4nzTjALfm7bZ0W08Mu+kyMUizw5EHpEDfpz7e6UkUWgAgwX+KnR0L\nSW6dKIrwIKibsH492NjtvCshHEYRsd+SHuuxhD2qVvPncuP8YDNsOogVI5XcVIHN\ngU3s9JqRv7A6Ne9Rf4bnje2x7GCjs5YfHrc3yOsWGQKBgQD9LUmMUTIWJjNffZDp\nZ+Hs/b39youIR2zDpG66rDrryaFV0j5M4j22s1/yMko5xgU6K8/q75+aVRma4uBj\nJ8bC3JAm6AasutwaZzw2iSddL95BGQJnU8vj2R+HCdPseY2SkucUd2TdpBYB/GfY\n/s9zBi6wB+J5o93wgFqzvTn/pQKBgQDga1fn8VuzG7blArYb6QGodxiBVzugwUFI\nz4byihFtZ9d7dsim8IuTl2Nsu+8VdJ9bPBBnBuAhb+7WKxdGFDwkAhx0TbqofnRu\nhVzcHiQw3DdRU6B7l62hy7h9WbE/hh1Ieoqzu8sm5NrozpADKiIRSArlYww1bctp\nzTVef1smwwKBgQC47GRFPmVkMkDh5bh8E6m8zdYBUvk58FmsSSWr6fMbmGnkYq2o\nwLOtCDXzlPIMYuwcgjXa923Pb4tFS6P0KPCQEkSkdmOxtMbh7jfX6bQ756PiTp8n\n09ggM85Id1A48MwMnIp2s4KUNa7xgmhj4jKqZICjiQ7OJHgOm+IkWPnNFQKBgA3J\nSlsrTpDQMQocgodwZvOsbUr0yurzNEl+xjF8Hdn0uR4Si/0FZe2pLShu2x8cTPXT\n2MYbfAVR1QA/IWzirVcg8hKK+96B9jEb7Dpxd5Ow040hcAhnpvj7hNfefK8EVCQx\nsvebn71B7djDZ8FqAzHAlTrGPOfDWO2k95RYiob1AoGAUjvdJQoLJxN2u/Z9gnEN\nT1xotdMDrJMRzG48pUJnvTvBYpTXaDp3//fnE1d7tINT3MNSno6UT5X5/9uE3gIO\nhJJe7AAghNasNyD3XSOwdCGMe4wpyCETuqs38V5/hSqnrBu6D6juiggpVD6miy7E\nnG9bxRifLv8Z5ANY5vVVfmo=\n-----END PRIVATE KEY-----\n",

"client\_email": "firebase-adminsdk-fbsvc@pz-tracking.iam.gserviceaccount.com",

"client\_id": "105309635317365581578",

"auth\_uri": "https://accounts.google.com/o/oauth2/auth",

"token\_uri": "https://oauth2.googleapis.com/token",

"auth\_provider\_x509\_cert\_url": "https://www.googleapis.com/oauth2/v1/certs",

"client\_x509\_cert\_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-fbsvc%40pz-tracking.iam.gserviceaccount.com",

"universe\_domain": "googleapis.com"

}