JS | ANALISE

// ============================================================

// ANALISE.JS — VERSÃO ANOTADA (estrutura por secções, sem alterar lógica)

// ------------------------------------------------------------

// Objetivo:

//  - Isolar e documentar as PARTES do algoritmo para afinar parâmetros

//  - NÃO muda comportamento: apenas adiciona comentários e marcadores

//

// Índice de Secções (procura por estes marcadores):

//  [S1] Imports & Dependências Dinâmicas

//  [S2] Helpers de Aparência / Formatação / Utils

//  [S3] Configurável (CFG) — Pesos/Limites do algoritmo

//  [S4] Estado & Cache em Memória (ALL\_ROWS, filtros, seleção)

//  [S5] Firestore — Carregamento e Normalização dos Dados

//  [S6] Filtros & Ordenação — Construção da tabela base

//  [S7] Gráficos — Setor, Mercado, Top Yield

//  [S8] Calendário de Dividendos (Heatmap 12 meses)

//  [S9] Tabela — Renderização e Interação (seleção, ordenação)

// [S10] Simulação (selecionados) — Preparação & Distribuição

// [S11] Relatório (PDF) — Geração a partir da seleção

// [S12] Interações de UI (event listeners) & Init

// ============================================================

// [S1] Imports & Dependências Dinâmicas

// screens/analise.js

import { db } from "../firebase-config.js";

import { collection, getDocs, query } from "https://www.gstatic.com/firebasejs/10.12.0/firebase-firestore.js";

/\* =========================================================

Carregamento “on-demand” de libs (Chart.js, html2canvas, jsPDF)

========================================================= \*/

async function ensureScript(src) {

if ([...document.scripts].some(s => s.src === src)) return;

await new Promise((resolve, reject) => {

const s = document.createElement("script");

s.src = src; s.onload = resolve; s.onerror = reject;

document.head.appendChild(s);

});

}

async function ensureChartJS() {

if (window.Chart) return;

await ensureScript("https://cdn.jsdelivr.net/npm/chart.js@4.4.3/dist/chart.umd.min.js");

}

async function ensurePDFLibs() {

if (!window.html2canvas) await ensureScript("https://cdn.jsdelivr.net/npm/html2canvas@1.4.1/dist/html2canvas.min.js");

if (!window.jspdf) await ensureScript("https://cdn.jsdelivr.net/npm/jspdf@2.5.1/dist/jspdf.umd.min.js");

}

async function ensureAutoTable() {

  // só carrega o plugin se ainda não existir

  if (!window.jspdf?.autoTable) {

    await ensureScript(

      "https://cdn.jsdelivr.net/npm/jspdf-autotable@3.8.2/dist/jspdf.plugin.autotable.min.js"

    );

  }

}

/\* =========================================================

Aparência / helpers

========================================================= \*/

const isDark = () => document.documentElement.getAttribute("data-theme") === "dark";

const chartColors = () => ({

grid: isDark() ? "rgba(255,255,255,.12)" : "rgba(0,0,0,.12)",

ticks: isDark() ? "rgba(255,255,255,.85)" : "rgba(0,0,0,.75)",

tooltipBg: isDark() ? "rgba(17,17,17,.95)" : "rgba(255,255,255,.95)",

tooltipFg: isDark() ? "#fff" : "#111",

});

const PALETTE = ["#4F46E5","#22C55E","#EAB308","#EF4444","#06B6D4","#F59E0B","#A855F7","#10B981","#3B82F6","#F472B6","#84CC16","#14B8A6"];

const mesesPT = ["Janeiro","Fevereiro","Março","Abril","Maio","Junho","Julho","Agosto","Setembro","Outubro","Novembro","Dezembro"];

const mesToIdx = new Map(mesesPT.map((m, i) => [m, i]));

const toNum = (v) => { const n = Number(v); return Number.isFinite(n) ? n : 0; };

const fmtEUR = (n) =>

  Number(n || 0).toLocaleString("pt-PT", {

    style: "currency",

    currency: "EUR",

  });

const clamp = (v, min, max) => Math.max(min, Math.min(max, v));

const canon = (s) =>

  String(s ?? "")

    .replace(/\u00A0/g, " ")

    .replace(/[\u200B-\u200D]/g, "")

    .replace(/\s+/g, " ")

    .trim();

/\* =========================================================

   Config ajustável — pesos/limites do algoritmo (visível)

   ========================================================= \*/

const CFG = {

  // limites prudentes (crescimento anualizado composto)

  MAX\_ANNUAL\_RETURN: 0.8, // +80%/ano

  MIN\_ANNUAL\_RETURN: -0.8, // -80%/ano

  // peso dos componentes no score [0..1] (R = retorno/€; V = P/E; T = tendência; Rsk = fator “constante”)

  WEIGHTS: {

    R: 0.55, // retorno por euro investido

    V: 0.15, // valuation por P/E

    T: 0.25, // técnica (SMA50/SMA200)

    Rsk: 0.05, // risco base

  },

  // percentagem máxima do total por ticker no modo frações

  MAX\_PCT\_POR\_TICKER: 0.35,

};

window.ANL\_CFG = CFG; // podes ajustar via consola se quiseres

/\* =========================================================

   Cálculos de dividendos / yield

   - alpha\_update\_sheet grava:

     • dividendoMedio24m = ANUAL (média 24m)

     • dividendo         = POR PAGAMENTO (média por pagamento 24m)

     • periodicidade + mes (distribuição mensal)

   ========================================================= \*/

function anualizarDividendo(dividendoPorPagamento, periodicidade) {

  const d = toNum(dividendoPorPagamento);

  const p = String(periodicidade || "").toLowerCase();

  if (d <= 0) return 0;

  if (p === "mensal") return d \* 12;

  if (p === "trimestral") return d \* 4;

  if (p === "semestral") return d \* 2;

  return d; // anual (ou n/A)

}

function anualPreferido(doc) {

  const d24 = toNum(doc.dividendoMedio24m);

  if (d24 > 0) return d24; // anual (média 24m)

  return anualizarDividendo(doc.dividendo, doc.periodicidade);

}

function perPayment(doc) {

  const base = toNum(doc.dividendo); // por pagamento (média 24m)

  if (base > 0) return base;

  const anual = anualPreferido(doc);

  const per = String(doc.periodicidade || "");

  if (per === "Mensal") return anual / 12;

  if (per === "Trimestral") return anual / 4;

  if (per === "Semestral") return anual / 2;

  if (per === "Anual") return anual;

  return 0;

}

function computeYieldPct(annualDividend, valorStock) {

  if (

    !Number.isFinite(annualDividend) ||

    !Number.isFinite(valorStock) ||

    valorStock <= 0

  )

    return 0;

  return (annualDividend / valorStock) \* 100;

}

/\* =========================================================

   Seleção / Ordenação / Tabela

   ========================================================= \*/

const selectedTickers = new Set();

const updateSelCount = () => {

  const el = document.getElementById("anlSelCount");

  if (el) el.textContent = String(selectedTickers.size);

};

let sortKey = null;

let sortDir = "desc";

const SORT\_ACCESSORS = {

  ticker: (r) => r.ticker,

  nome: (r) => r.nome || "",

  setor: (r) => r.setor || "",

  mercado: (r) => r.mercado || "",

  yield: (r) => (Number.isFinite(r.yield) ? r.yield : -Infinity),

  yield24: (r) => (Number.isFinite(r.yield24) ? r.yield24 : -Infinity),

  divPer: (r) => (Number.isFinite(r.divPer) ? r.divPer : -Infinity),

  divAnual: (r) => (Number.isFinite(r.divAnual) ? r.divAnual : -Infinity),

  pe: (r) => (Number.isFinite(r.pe) ? r.pe : Infinity),

  delta50: (r) => (Number.isFinite(r.delta50) ? r.delta50 : -Infinity),

  delta200: (r) => (Number.isFinite(r.delta200) ? r.delta200 : -Infinity),

  g1w: (r) => (Number.isFinite(r.g1w) ? r.g1w : -Infinity),

  g1m: (r) => (Number.isFinite(r.g1m) ? r.g1m : -Infinity),

  g1y: (r) => (Number.isFinite(r.g1y) ? r.g1y : -Infinity),

  periodicidade: (r) => r.periodicidade || "",

  mes: (r) => r.mes || "",

  observacao: (r) => r.observacao || "",

};

function sortRows(rows) {

  if (!sortKey) return rows;

  const acc = SORT\_ACCESSORS[sortKey] || ((r) => r[sortKey]);

  const dir = sortDir === "asc" ? 1 : -1;

  return [...rows].sort((a, b) => {

    const va = acc(a),

      vb = acc(b);

    if (va < vb) return -1 \* dir;

    if (va > vb) return 1 \* dir;

    return 0;

  });

}

function markSortedHeader() {

  document

    .querySelectorAll("#anlTable thead th.sortable")

    .forEach((th) => th.classList.remove("sorted-asc", "sorted-desc"));

  if (sortKey) {

    const th = document.querySelector(

      `#anlTable thead th[data-sort="${sortKey}"]`

    );

    if (th) th.classList.add(sortDir === "asc" ? "sorted-asc" : "sorted-desc");

  }

}

/\* =========================================================

   Charts (gerais) — sem tremer (animation: false)

   ========================================================= \*/

let charts = { setor: null, mercado: null, topYield: null };

function destroyCharts() {

  charts.setor?.destroy();

  charts.mercado?.destroy();

  charts.topYield?.destroy();

  charts = { setor: null, mercado: null, topYield: null };

}

function renderDonut(elId, dataMap) {

  const el = document.getElementById(elId);

  if (!el) return null;

  const labels = Array.from(dataMap.keys());

  const data = Array.from(dataMap.values());

  if (!data.length) {

    el.parentElement?.classList.add("muted");

    return null;

  }

  return new Chart(el, {

    type: "doughnut",

    data: {

      labels,

      datasets: [

        {

          data,

          backgroundColor: labels.map((\_, i) => PALETTE[i % PALETTE.length]),

          borderWidth: 1,

        },

      ],

    },

    options: {

      responsive: true,

      maintainAspectRatio: false,

      cutout: "62%",

      animation: false,

      plugins: {

        legend: { position: "bottom", labels: { color: chartColors().ticks } },

        tooltip: {

          backgroundColor: chartColors().tooltipBg,

          titleColor: chartColors().tooltipFg,

          bodyColor: chartColors().tooltipFg,

          callbacks: {

            label: (ctx) => {

              const total = data.reduce((a, b) => a + b, 0) || 1;

              const v = Number(ctx.parsed);

              const pct = ((v / total) \* 100).toFixed(1);

              return ` ${ctx.label}: ${v} (${pct}%)`;

            },

          },

        },

      },

    },

  });

}

function renderTopYield(elId, rows) {

  const el = document.getElementById(elId);

  if (!el) return null;

  const top = [...rows]

    .map((r) => ({ tk: r.ticker, y: Number.isFinite(r.yield) ? r.yield : 0 }))

    .filter((x) => x.y > 0)

    .sort((a, b) => b.y - a.y)

    .slice(0, 8);

  if (!top.length) return null;

  return new Chart(el, {

    type: "bar",

    data: {

      labels: top.map((x) => x.tk),

      datasets: [

        {

          label: "Yield (%)",

          data: top.map((x) => x.y),

          backgroundColor: "#22C55E",

        },

      ],

    },

    options: {

      indexAxis: "y",

      responsive: true,

      maintainAspectRatio: false,

      animation: false,

      scales: {

        x: {

          ticks: { color: chartColors().ticks },

          grid: { color: chartColors().grid },

        },

        y: {

          ticks: { color: chartColors().ticks },

          grid: { color: chartColors().grid },

        },

      },

      plugins: {

        legend: { labels: { color: chartColors().ticks } },

        tooltip: {

          backgroundColor: chartColors().tooltipBg,

          titleColor: chartColors().tooltipFg,

          bodyColor: chartColors().tooltipFg,

          callbacks: {

            label: (ctx) =>

              ` ${ctx.dataset.label}: ${ctx.parsed.x.toFixed(2)}%`,

          },

        },

      },

    },

  });

}

function renderCharts(rows) {

  const groupBy = (key) => {

    const map = new Map();

    rows.forEach((r) => {

      const k = canon(r[key] || "—");

      map.set(k, (map.get(k) || 0) + 1);

    });

    return map;

  };

  destroyCharts();

  charts.setor = renderDonut("anlChartSetor", groupBy("setor"));

  charts.mercado = renderDonut("anlChartMercado", groupBy("mercado"));

  charts.topYield = renderTopYield("anlChartTopYield", rows);

}

/\* =========================================================

   Calendário (12 meses) — por pagamento (média 24m)

   ========================================================= \*/

function mesesPagamento(periodicidade, mesTipicoIdx) {

  if (!Number.isFinite(mesTipicoIdx)) return [];

  if (periodicidade === "Mensal")

    return Array.from({ length: 12 }, (\_, i) => i);

  if (periodicidade === "Trimestral")

    return [0, 3, 6, 9].map((k) => (mesTipicoIdx + k) % 12);

  if (periodicidade === "Semestral")

    return [0, 6].map((k) => (mesTipicoIdx + k) % 12);

  if (periodicidade === "Anual") return [mesTipicoIdx];

  return [];

}

function renderHeatmap(rows) {

  const body = document.getElementById("anlHeatmapBody");

  const headMonths = document.getElementById("anlHeatmapHeaderMonths");

  if (!body || !headMonths) return;

  headMonths.innerHTML = mesesPT

    .map((m) => `<div class="cell"><strong>${m}</strong></div>`)

    .join("");

  // thresholds (com base em per-payment)

  const perPayments = rows

    .map((r) => perPayment(r))

    .filter((v) => v > 0)

    .sort((a, b) => a - b);

  const q1 = perPayments.length

    ? perPayments[Math.floor(perPayments.length \* 0.33)]

    : 0.01;

  const q2 = perPayments.length

    ? perPayments[Math.floor(perPayments.length \* 0.66)]

    : 0.02;

  body.innerHTML = rows

    .map((r) => {

      const per = String(r.periodicidade || "n/A");

      const idxMes = mesToIdx.get(String(r.mes || "")) ?? NaN;

      const meses = mesesPagamento(per, idxMes);

      const perPay = perPayment(r);

      const klass =

        perPay > 0

          ? perPay <= q1

            ? "pay-weak"

            : perPay <= q2

            ? "pay-med"

            : "pay-strong"

          : "";

      const cells = Array.from({ length: 12 }, (\_, m) => {

        if (!meses.includes(m)) return `<div class="cell"></div>`;

        const tt = `${r.ticker} • ${mesesPT[m]} • ~${fmtEUR(perPay)}`;

        return `<div class="cell tt ${klass}" data-tt="${tt}">${

          perPay ? fmtEUR(perPay) : ""

        }</div>`;

      }).join("");

      const nome = r.nome ? ` <span class="muted">— ${r.nome}</span>` : "";

      return `

      <div class="row">

        <div class="cell sticky-col"><strong>${r.ticker}</strong>${nome}</div>

        <div class="months">${cells}</div>

      </div>`;

    })

    .join("");

  // sincroniza header ao scroll

  const headerScroll = document.getElementById("anlHeatmapHeaderScroll");

  const onScroll = (e) => {

    headMonths.scrollLeft = e.target.scrollLeft;

    headerScroll.scrollLeft = e.target.scrollLeft;

  };

  body.removeEventListener("scroll", onScroll);

  body.addEventListener("scroll", onScroll, { passive: true });

  // ir para Dezembro na 1ª renderização

  setTimeout(() => {

    const maxX = body.scrollWidth - body.clientWidth;

    if (maxX > 0) {

      body.scrollLeft = maxX;

      headMonths.scrollLeft = maxX;

      headerScroll.scrollLeft = maxX;

    }

  }, 0);

}

/\* =========================================================

   Tabela principal

   ========================================================= \*/

function renderTable(rows) {

  const tb = document.getElementById("anlTableBody");

  if (!tb) return;

  const badgePE = (pe) => {

    if (!Number.isFinite(pe) || pe <= 0)

      return `<span class="badge muted">—</span>`;

    if (pe < 15) return `<span class="badge ok">${pe.toFixed(2)} Barato</span>`;

    if (pe <= 25)

      return `<span class="badge warn">${pe.toFixed(2)} Justo</span>`;

    return `<span class="badge danger">${pe.toFixed(2)} Caro</span>`;

  };

  const badgeYield = (y, y24) => {

    if (!Number.isFinite(y)) return `<span class="badge muted">—</span>`;

    let base = "muted";

    if (y >= 6) base = "warn";

    else if (y >= 2) base = "ok";

    const curr = `<span class="badge ${base}">${y.toFixed(2)}%</span>`;

    if (Number.isFinite(y24)) {

      const comp =

        y - y24 >= 0

          ? `<span class="badge up">↑ acima da média</span>`

          : `<span class="badge down">↓ abaixo da média</span>`;

      return `${curr} ${comp}`;

    }

    return curr;

  };

  const pct = (v) => {

    if (!Number.isFinite(v)) return `—`;

    const cls = v >= 0 ? "up" : "down";

    const sign = v >= 0 ? "+" : "";

    return `<span class="${cls}">${sign}${v.toFixed(2)}%</span>`;

  };

  tb.innerHTML = rows

    .map((r) => {

      const checked = selectedTickers.has(r.ticker) ? "checked" : "";

      const y = Number.isFinite(r.yield) ? r.yield : null;

      const y24 = Number.isFinite(r.yield24) ? r.yield24 : null;

      const divPerTxt = r.divPer > 0 ? fmtEUR(r.divPer) : "—";

      const divAnualTxt = r.divAnual > 0 ? fmtEUR(r.divAnual) : "—";

      return `

      <tr>

        <td class="sticky-col"><input type="checkbox" class="anlRowSel" data-ticker="${

          r.ticker

        }" ${checked} /></td>

        <td class="sticky-col"><strong>${r.ticker}</strong></td>

        <td>${r.nome || "—"}</td>

        <td>${r.setor || "—"}</td>

        <td>${r.mercado || "—"}</td>

        <td>${badgeYield(y, y24)}</td>

        <td>${

          Number.isFinite(r.yield24) ? `${r.yield24.toFixed(2)}%` : "—"

        }</td>

        <td>${divPerTxt}</td>

        <td>${divAnualTxt}</td>

        <td>${badgePE(r.pe)}</td>

        <td>${pct(r.delta50)}</td>

        <td>${pct(r.delta200)}</td>

        <td>${pct(r.g1w)}</td>

        <td>${pct(r.g1m)}</td>

        <td>${pct(r.g1y)}</td>

        <td>${r.periodicidade || "—"}</td>

        <td>${r.mes || "—"}</td>

        <td>${r.observacao || "—"}</td>

      </tr>`;

    })

    .join("");

  tb.querySelectorAll(".anlRowSel").forEach((ch) => {

    ch.addEventListener("change", (e) => {

      const t = e.target.getAttribute("data-ticker");

      if (!t) return;

      if (e.target.checked) selectedTickers.add(t);

      else selectedTickers.delete(t);

      updateSelCount();

    });

  });

}

/\* =========================================================

   Firestore (fetch)

   ========================================================= \*/

let ALL\_ROWS = [];

async function fetchAcoes() {

  const snap = await getDocs(query(collection(db, "acoesDividendos")));

  const rows = [];

  snap.forEach((doc) => {

    const d = doc.data();

    const ticker = String(d.ticker || "").toUpperCase();

    if (!ticker) return;

    const valor = toNum(d.valorStock);

    const anual = toNum(d.dividendoMedio24m) || anualPreferido(d); // anual (média 24m preferida)

    const y = computeYieldPct(anual, valor);

    rows.push({

      ticker,

      nome: d.nome || "",

      setor: canon(d.setor || ""),

      mercado: canon(d.mercado || ""),

      valorStock: valor,

      dividendo: toNum(d.dividendo), // POR PAGAMENTO (média 24m)

      dividendoMedio24m: toNum(d.dividendoMedio24m), // ANUAL (média 24m)

      periodicidade: d.periodicidade || "",

      mes: d.mes || "",

      observacao: d.observacao || d["Observação"] || "",

      // derivados

      divPer: perPayment(d),

      divAnual: anual,

      yield: Number.isFinite(y) ? y : null,

      // crescimento (cuidado: podem vir strings)

      g1w: Number(d.taxaCrescimento\_1semana) || 0,

      g1m: Number(d.taxaCrescimento\_1mes) || 0,

      g1y: Number(d.taxaCrescimento\_1ano) || 0,

      // valuation/técnicos (podem vir como string)

      yield24: Number(d.yield24) || null, // se existir, opcional

      pe:

        Number(d.pe) ||

        Number(d.peRatio) ||

        Number(d["P/E ratio (Preço/Lucro)"]) ||

        null,

      delta50: Number(d.delta50) || 0,

      delta200: Number(d.delta200) || 0,

      sma50: Number(d.sma50) || Number(d.SMA50) || null,

      sma200: Number(d.sma200) || Number(d.SMA200) || null,

    });

  });

  ALL\_ROWS = rows;

}

/\* =========================================================

   Filtros

   ========================================================= \*/

const keyStr = (s) =>

  String(s ?? "")

    .normalize("NFD")

    .replace(/\p{Diacritic}/gu, "")

    .trim()

    .toLowerCase();

function applyFilters() {

  const term = keyStr(document.getElementById("anlSearch")?.value || "");

  const setor = document.getElementById("anlSetor")?.value || "";

  const mercado = document.getElementById("anlMercado")?.value || "";

  const periodo = document.getElementById("anlPeriodo")?.value || "";

  let rows = [...ALL\_ROWS];

  if (term)

    rows = rows.filter(

      (r) => keyStr(r.ticker).includes(term) || keyStr(r.nome).includes(term)

    );

  if (setor) rows = rows.filter((r) => r.setor === setor);

  if (mercado) rows = rows.filter((r) => r.mercado === mercado);

  if (periodo) rows = rows.filter((r) => (r.periodicidade || "") === periodo);

  rows = sortRows(rows);

  renderCharts(rows);

  renderHeatmap(rows);

  hookHeatmapScrollSync();

  renderTable(rows);

  const selAll = document.getElementById("anlSelectAll");

  if (selAll)

    selAll.checked =

      rows.length > 0 && rows.every((r) => selectedTickers.has(r.ticker));

}

function populateFilters() {

  const setorSel = document.getElementById("anlSetor");

  const mercadoSel = document.getElementById("anlMercado");

  const setSet = new Set(),

    merSet = new Set();

  ALL\_ROWS.forEach((r) => {

    if (r.setor) setSet.add(r.setor);

    if (r.mercado) merSet.add(r.mercado);

  });

  const addOpts = (sel, values) => {

    const cur = sel.value;

    sel.innerHTML =

      `<option value="">Todos</option>` +

      [...values]

        .sort()

        .map((v) => `<option>${v}</option>`)

        .join("");

    sel.value = cur || "";

  };

  if (setorSel) addOpts(setorSel, setSet);

  if (mercadoSel) addOpts(mercadoSel, merSet);

}

/\* =========================================================

   === LUCRO MÁXIMO — versão prudente e configurável ===

   ========================================================= \*/

// helpers de anualização prudente (compounding)

function annualizeRate(row, periodoSel) {

  const w = Number(row?.g1w ?? 0) / 100;

  const m = Number(row?.g1m ?? 0) / 100;

  const y = Number(row?.g1y ?? 0) / 100;

  let rAnnual;

  if (periodoSel === "1s") {

    rAnnual = Math.pow(1 + w, 52) - 1;

  } else if (periodoSel === "1m") {

    rAnnual = Math.pow(1 + m, 12) - 1;

  } else {

    rAnnual = y; // já anual

  }

  return clamp(rAnnual, CFG.MIN\_ANNUAL\_RETURN, CFG.MAX\_ANNUAL\_RETURN);

}

function scorePE(pe) {

  if (!Number.isFinite(pe) || pe <= 0) return 0.5;

  if (pe <= 12) return 1.0;

  if (pe <= 15) return 0.85;

  if (pe <= 20) return 0.7;

  if (pe <= 25) return 0.5;

  if (pe <= 30) return 0.35;

  return 0.2;

}

function scoreTrend(preco, sma50, sma200) {

  let t = 0;

  if (Number.isFinite(preco) && Number.isFinite(sma50) && preco > sma50)

    t += 0.2;

  if (Number.isFinite(preco) && Number.isFinite(sma200) && preco > sma200)

    t += 0.3;

  if (Number.isFinite(sma50) && Number.isFinite(sma200) && sma50 > sma200)

    t += 0.1;

  return clamp(t, 0, 0.6);

}

function percentile(arr, p) {

  if (!arr.length) return 0;

  const a = [...arr].sort((x, y) => x - y);

  const idx = Math.floor((a.length - 1) \* clamp(p, 0, 1));

  return a[idx];

}

function calcularMetricasBase(

  acao,

  { periodo = "1m", horizonte = 1, incluirDiv = true } = {}

) {

  const precoAtual = toNum(acao.valorStock);

  const anualDiv = toNum(acao.divAnual ?? anualPreferido(acao)); // ANUAL (média 24m)

  const rAnnual = annualizeRate(acao, periodo);

  const h = Math.max(1, Number(horizonte || 1));

  const valorizacaoNoHorizonte = precoAtual \* (Math.pow(1 + rAnnual, h) - 1);

  const dividendosNoHorizonte = incluirDiv ? anualDiv \* h : 0;

  const lucroUnidade = dividendosNoHorizonte + valorizacaoNoHorizonte;

  const retornoPorEuro = precoAtual > 0 ? lucroUnidade / precoAtual : 0;

  return {

    preco: precoAtual,

    dividendoAnual: anualDiv,

    taxaPct: rAnnual \* 100,

    totalDividendos: dividendosNoHorizonte,

    valorizacao: valorizacaoNoHorizonte,

    lucroUnidade,

    retornoPorEuro,

  };

}

function prepararCandidatos(

  rows,

  { periodo, horizonte, incluirDiv, modoEstrito = false }

) {

  let cands = rows

    .map((a) => ({

      ...a,

      metrics: calcularMetricasBase(a, { periodo, horizonte, incluirDiv }),

    }))

    .filter(

      (c) =>

        c.metrics.preco > 0 &&

        isFinite(c.metrics.lucroUnidade) &&

        c.metrics.lucroUnidade > 0

    );

  if (!cands.length) return [];

  const rets = cands

    .map((c) => c.metrics.retornoPorEuro)

    .filter((x) => x > 0 && isFinite(x));

  const p99 = Math.max(percentile(rets, 0.99), 1e-9);

  cands = cands

    .map((c) => {

      const R = clamp(c.metrics.retornoPorEuro / p99, 0, 1);

      if (modoEstrito)

        return { ...c, score: R, \_\_R: R, \_\_V: 0, \_\_T: 0, \_\_Rsk: 0 };

      const V = scorePE(c.pe);

      const T = scoreTrend(c.metrics.preco, c.sma50, c.sma200);

      const Rsk = 1.0;

      const W = CFG.WEIGHTS;

      const score = clamp(W.R \* R + W.V \* V + W.T \* T + W.Rsk \* Rsk, 0, 1);

      return { ...c, score, \_\_R: R, \_\_V: V, \_\_T: T, \_\_Rsk: Rsk };

    })

    .filter((c) => c.score > 0);

  return cands;

}

function makeLinha(c, qtd) {

  const investido = qtd \* c.metrics.preco;

  return {

    nome: c.nome,

    ticker: c.ticker,

    preco: c.metrics.preco,

    quantidade: qtd,

    investido,

    lucro: qtd \* c.metrics.lucroUnidade,

    taxaPct: c.metrics.taxaPct,

    dividendoAnual: c.metrics.dividendoAnual,

    divAnualAlloc: qtd \* c.metrics.dividendoAnual,

    divPeriodoAlloc: qtd \* c.metrics.totalDividendos,

    valorizAlloc: qtd \* c.metrics.valorizacao,

  };

}

function sumarizar(linhas, investimento, gasto) {

  const totalLucro = linhas.reduce((s, l) => s + l.lucro, 0);

  const totalDivAnual = linhas.reduce((s, l) => s + l.divAnualAlloc, 0);

  const totalDivPeriodo = linhas.reduce((s, l) => s + l.divPeriodoAlloc, 0);

  const totalValoriz = linhas.reduce((s, l) => s + l.valorizAlloc, 0);

  return {

    linhas,

    totalLucro,

    totalGasto: gasto,

    totalDivAnual,

    totalDivPeriodo,

    totalValoriz,

    restante: Math.max(0, investimento - gasto),

  };

}

function distribuirFracoes\_porScore(cands, investimento) {

  const somaScore = cands.reduce((s, c) => s + c.score, 0);

  if (!(somaScore > 0))

    return {

      linhas: [],

      totalLucro: 0,

      totalGasto: 0,

      totalDivAnual: 0,

      totalDivPeriodo: 0,

      totalValoriz: 0,

      restante: investimento,

    };

  const capAbs = CFG.MAX\_PCT\_POR\_TICKER

    ? CFG.MAX\_PCT\_POR\_TICKER \* investimento

    : Infinity;

  let restante = investimento;

  const linhas = [];

  const ord = [...cands].sort((a, b) => b.score - a.score);

  for (const c of ord) {

    const investAlvo = (c.score / somaScore) \* investimento;

    const investido = Math.min(investAlvo, capAbs, restante);

    if (investido <= 0) continue;

    const qtd = investido / c.metrics.preco;

    if (qtd > 0 && isFinite(qtd)) {

      linhas.push(makeLinha(c, qtd));

      restante -= investido;

      if (restante <= 0) break;

    }

  }

  if (restante > 0 && capAbs < Infinity) {

    for (const c of ord) {

      const ja = linhas.find((l) => l.ticker === c.ticker);

      const jaInvest = ja ? ja.investido : 0;

      const margem = Math.max(0, capAbs - jaInvest);

      if (margem <= 0) continue;

      const investido = Math.min(margem, restante);

      if (investido <= 0) continue;

      const qtd = investido / c.metrics.preco;

      if (!(qtd > 0 && isFinite(qtd))) continue;

      if (ja) {

        ja.quantidade += qtd;

        ja.investido += investido;

        ja.lucro += qtd \* c.metrics.lucroUnidade;

        ja.divAnualAlloc += qtd \* c.metrics.dividendoAnual;

        ja.divPeriodoAlloc += qtd \* c.metrics.totalDividendos;

        ja.valorizAlloc += qtd \* c.metrics.valorizacao;

      } else {

        linhas.push(makeLinha(c, qtd));

      }

      restante -= investido;

      if (restante <= 0) break;

    }

  }

  const gasto = investimento - restante;

  return sumarizar(linhas, investimento, gasto);

}

function distribuirInteiros\_porScore(cands, investimento) {

  const soma = cands.reduce((s, c) => s + c.score, 0);

  if (!(soma > 0))

    return {

      linhas: [],

      totalLucro: 0,

      totalGasto: 0,

      totalDivAnual: 0,

      totalDivPeriodo: 0,

      totalValoriz: 0,

      restante: investimento,

    };

  const ordered = [...cands].sort((a, b) => b.score - a.score);

  const base = ordered.map((c) => {

    const propor = c.score / soma;

    const investAlvo = investimento \* propor;

    const qtd = Math.max(

      0,

      Math.floor(c.metrics.preco > 0 ? investAlvo / c.metrics.preco : 0)

    );

    return { c, qtd };

  });

  let gasto = base.reduce((s, x) => s + x.qtd \* x.c.metrics.preco, 0);

  let restante = investimento - gasto;

  while (true) {

    let escolhido = null;

    for (const cand of ordered) {

      if (cand.metrics.preco <= restante && cand.metrics.lucroUnidade > 0) {

        escolhido = cand;

        break;

      }

    }

    if (!escolhido) break;

    const reg = base.find((x) => x.c === escolhido);

    if (!reg) break;

    reg.qtd += 1;

    gasto += escolhido.metrics.preco;

    restante = investimento - gasto;

    if (

      !ordered.some(

        (o) => o.metrics.preco <= restante && o.metrics.lucroUnidade > 0

      )

    )

      break;

  }

  const linhas = base

    .filter(({ qtd }) => qtd > 0)

    .map(({ c, qtd }) => makeLinha(c, qtd));

  return sumarizar(linhas, investimento, gasto);

}

/\* =========================================================

   Modal Simulação (open/close + render)

   ========================================================= \*/

   // --- Estado do último resultado de simulação (para o Relatório) ---

let \_\_ANL\_LAST\_SIM = {

  rows: [],            // linhas normalizadas para o relatório

  opts: null           // { horizonte, periodo, incluirDiv, investimento }

};

function openSimModal() {

  document.getElementById("anlSimModal")?.classList.remove("hidden");

}

function closeSimModal() {

  document.getElementById("anlSimModal")?.classList.add("hidden");

}

function renderResultadoSimulacao(res) {

  const cont = document.getElementById("anlSimResultado");

  if (!cont) return;

  if (!res || !res.linhas || res.linhas.length === 0) {

    cont.innerHTML = `<p class="muted">Sem resultados. Verifica o investimento e a seleção.</p>`;

    return;

  }

  const horizonte = Number(document.getElementById("anlSimHoriz")?.value || 1);

  const periodoSel = document.getElementById("anlSimPeriodo")?.value || "1m";

  const incluirDiv = !!document.getElementById("anlSimIncluiDiv")?.checked;

  const periodoLabel =

    periodoSel === "1s" ? "1 semana" : periodoSel === "1m" ? "1 mês" : "1 ano";

  const retornoTotal = res.totalDivPeriodo + res.totalValoriz;

  const retornoPct =

    res.totalGasto > 0 ? (retornoTotal / res.totalGasto) \* 100 : 0;

  const rows = res.linhas

    .filter((l) => l.quantidade > 0 && l.investido > 0)

    .map((l) => {

      const lucroLinha = l.divPeriodoAlloc + l.valorizAlloc;

      const noGrowth = Math.abs(l.valorizAlloc) < 1e-8;

      return `

      <tr>

        <td><strong>${l.ticker}</strong></td>

        <td>${l.nome || "—"}</td>

        <td>${fmtEUR(l.preco)}</td>

        <td>${Number(l.quantidade).toFixed(2)}</td>

        <td>${fmtEUR(l.investido)}</td>

        <td>${fmtEUR(lucroLinha)}${

        noGrowth

          ? ` <span class="badge muted" title="Sem valorização (taxa=0)">r=0%</span>`

          : ""

      }</td>

        <td>${fmtEUR(l.divAnualAlloc)}</td>

        <td>${fmtEUR(l.divPeriodoAlloc)}</td>

        <td>${fmtEUR(l.valorizAlloc)}</td>

      </tr>`;

    })

    .join("");

  cont.innerHTML = `

    <div class="card" style="margin-bottom:10px;">

      <div class="card-content" style="display:flex; gap:14px; flex-wrap:wrap; align-items:center;">

        <div><strong>Horizonte:</strong> ${horizonte} ${

    horizonte === 1 ? "ano" : "anos"

  }</div>

        <div><strong>Período de crescimento:</strong> ${periodoLabel}</div>

        <div><strong>Dividendos:</strong> ${

          incluirDiv ? "incluídos" : "excluídos"

        }</div>

      </div>

    </div>

    <div class="tabela-scroll-wrapper">

      <table class="fine-table" style="width:100%">

        <thead>

          <tr>

            <th>Ticker</th><th>Nome</th><th>Preço</th><th>Qtd.</th>

            <th>Investido</th>

            <th>Lucro estimado (= Div. no horizonte + Valorização)</th>

            <th>Dividendo anual (aloc.)</th>

            <th>Dividendos no horizonte (h=${horizonte})</th>

            <th>Valorização no horizonte</th>

          </tr>

        </thead>

        <tbody>${rows}</tbody>

        <tfoot>

          <tr>

            <th colspan="4" style="text-align:right;">Totais</th>

            <th>${fmtEUR(res.totalGasto)}</th>

            <th>${fmtEUR(retornoTotal)}</th>

            <th>${fmtEUR(res.totalDivAnual)}</th>

            <th>${fmtEUR(res.totalDivPeriodo)}</th>

            <th>${fmtEUR(res.totalValoriz)}</th>

          </tr>

        </tfoot>

      </table>

    </div>

    <div class="card" style="margin-top:10px;">

      <div class="card-content" style="display:flex; gap:16px; flex-wrap:wrap;">

        <div><strong>Retorno total (€):</strong> ${fmtEUR(retornoTotal)}</div>

        <div><strong>Retorno total (%):</strong> ${retornoPct.toFixed(2)}%</div>

        <div><strong>Dividendos anuais (soma aloc.):</strong> ${fmtEUR(

          res.totalDivAnual

        )}</div>

        <div><strong>Dividendos no horizonte:</strong> ${fmtEUR(

          res.totalDivPeriodo

        )}</div>

        <div><strong>Valorização no horizonte:</strong> ${fmtEUR(

          res.totalValoriz

        )}</div>

        ${

          res.restante > 0

            ? `<div><strong>Restante não investido:</strong> ${fmtEUR(

                res.restante

              )}</div>`

            : ""

        }

      </div>

    </div>`;

}

/\* === Pizza (selecionados) — sem “tremores” === \*/

let chartSelSetor = null;

async function renderSelectedSectorChart(rowsSelecionadas) {

  await ensureChartJS();

  const wrap = document.getElementById("anlSelSectorChartWrap");

  const el = document.getElementById("anlSelSectorChart");

  if (!wrap || !el) return;

  chartSelSetor?.destroy();

  const map = new Map();

  rowsSelecionadas.forEach((r) => {

    const k = canon(r.setor || "—");

    map.set(k, (map.get(k) || 0) + 1);

  });

  const labels = Array.from(map.keys());

  const data = Array.from(map.values());

  if (!data.length) {

    wrap.classList.add("muted");

    return;

  }

  const colors = labels.map((\_, i) => PALETTE[i % PALETTE.length]);

  chartSelSetor = new Chart(el, {

    type: "doughnut",

    data: {

      labels,

      datasets: [{ data, backgroundColor: colors, borderWidth: 1 }],

    },

    options: {

      responsive: true,

      maintainAspectRatio: true,

      cutout: "62%",

      animation: false,

      plugins: {

        legend: { position: "bottom", labels: { color: chartColors().ticks } },

        tooltip: {

          backgroundColor: chartColors().tooltipBg,

          titleColor: chartColors().tooltipFg,

          bodyColor: chartColors().tooltipFg,

          callbacks: {

            label: (ctx) => {

              const total = data.reduce((a, b) => a + b, 0) || 1;

              const v = Number(ctx.parsed);

              const pct = ((v / total) \* 100).toFixed(1);

              return ` ${ctx.label}: ${v} (${pct}%)`;

            },

          },

        },

      },

    },

  });

}

/\* =========================================================

[S11] Relatório (PDF) — completo

   ========================================================= \*/

/\* =========================================================

[S11] Relatório (PDF) — V2 profissional (única parte alterada)

========================================================= \*/

// Helpers específicos da V2

const \_fmtEUR = (n) => Number(n || 0).toLocaleString("pt-PT", { style: "currency", currency: "EUR" });

const \_pct    = (x) => `${(Number(x || 0) \* 100).toFixed(1)}%`;

// Tenta obter imagem de um canvas; se não existir, devolve null

function getChartImageByCanvasId(id){

  const cnv = document.getElementById(id);

  if(!cnv) return null;

  try{

    const chart = cnv.\_\_chartist || cnv.\_\_chart || cnv.chart || null;

    if(chart?.toBase64Image) return chart.toBase64Image();

    return cnv.toDataURL("image/png");

  }catch{ return null; }

}

// Cria gráficos “temporários” (invisíveis) só para o PDF, caso não existam no DOM

async function buildTempReportCharts(rows) {

  await ensureChartJS();

  // container offscreen

  const wrap = document.createElement("div");

  wrap.style.position = "fixed";

  wrap.style.left = "-10000px";

  document.body.appendChild(wrap);

  // Helpers para dados

  const labels = rows.map(r => r.ticker);

  const invest = rows.map(r => Number(r.investido||0));

  const lucro  = rows.map(r => Number(r.lucro||0));

  const divH   = rows.map(r => Number((r.divHoriz ?? r.dividendoHorizonte) || 0));

  const valH   = rows.map(r => Number(r.valorizacao||0));

  const mkCanvas = (id) => {

    const c = document.createElement("canvas");

    c.width = 900; c.height = 500; c.id = id;

    wrap.appendChild(c);

    return c.getContext("2d");

  };

  const imgs = [];

  // Pizza — distribuição do investimento

  if (!document.getElementById("chartDistInvest")) {

    const ctx = mkCanvas("chartDistInvest");

    new Chart(ctx, {

      type: "pie",

      data: { labels, datasets: [{ data: invest, backgroundColor: labels.map((\_,i)=>PALETTE[i%PALETTE.length]) }]},

      options: { animation:false, plugins:{ legend:{ display:false } } }

    });

    imgs.push({ id:"chartDistInvest", title:"Distribuição do Investimento por Ativo (Pizza)", img: getChartImageByCanvasId("chartDistInvest") });

  }

  // Barras — Lucro estimado por ativo

  if (!document.getElementById("chartLucroPorAtivo")) {

    const ctx = mkCanvas("chartLucroPorAtivo");

    new Chart(ctx, {

      type: "bar",

      data: { labels, datasets: [{ label:"Lucro (€)", data: lucro }] },

      options: { animation:false, plugins:{ legend:{ display:false } } }

    });

    imgs.push({ id:"chartLucroPorAtivo", title:"Lucro Estimado por Ativo (Barras)", img: getChartImageByCanvasId("chartLucroPorAtivo") });

  }

  // Barras agrupadas — Dividendos vs Valorização

  if (!document.getElementById("chartDivVsVal")) {

    const ctx = mkCanvas("chartDivVsVal");

    new Chart(ctx, {

      type: "bar",

      data: {

        labels,

        datasets: [

          { label:"Dividendos (H)", data: divH },

          { label:"Valorização (H)", data: valH }

        ]

      },

      options: { animation:false }

    });

    imgs.push({ id:"chartDivVsVal", title:"Dividendos vs Valorização por Ativo (Barras Agrupadas)", img: getChartImageByCanvasId("chartDivVsVal") });

  }

  // Barras — Investido por ativo

  if (!document.getElementById("chartInvestPorAtivo")) {

    const ctx = mkCanvas("chartInvestPorAtivo");

    new Chart(ctx, {

      type: "bar",

      data: { labels, datasets: [{ label:"Investido (€)", data: invest }] },

      options: { animation:false, plugins:{ legend:{ display:false } } }

    });

    imgs.push({ id:"chartInvestPorAtivo", title:"Valor Investido por Ativo (Barras)", img: getChartImageByCanvasId("chartInvestPorAtivo") });

  }

  // limpa canvases temporários após captura

  requestAnimationFrame(()=> document.body.removeChild(wrap));

  return imgs.filter(x => !!x.img);

}

// Mantém API antiga: encaminha para a V2

export async function generateReportPDF(selecionadas = [], opts = {}) {

  return generateReportPDF\_v2(selecionadas, opts);

}

// Nova V2

export async function generateReportPDF\_v2(rows = [], opts = {}) {

  await ensurePDFLibs();

  await ensureAutoTable();

  const { jsPDF } = window.jspdf;

  // 1) se houver simulação recente, usa-a; senão, normaliza 'rows' básicas

  const horizonteUI = Number(document.getElementById("anlSimHoriz")?.value || 1);

  const horizonte = Math.max(1, Number(opts.horizonte ?? \_\_ANL\_LAST\_SIM?.opts?.horizonte ?? horizonteUI ?? 1));

  let data;

  if (\_\_ANL\_LAST\_SIM?.rows?.length) {

    data = \_\_ANL\_LAST\_SIM.rows.map(r => ({

      nome: r.nome, ticker: r.ticker,

      investido: Number(r.investido||0),

      divAnual: Number(r.dividendoAnual||0),

      divHoriz: Number(r.dividendoHorizonte||r.divHoriz||0),

      valorizacao: Number(r.valorizacao||0),

      lucro: Number(r.lucro||0),

    }));

  } else {

    // fallback (sem simulação): tudo 0 excepto anual → horizonte = anual\*h

    data = (rows||[]).map(r=>{

      const nome = String(r.nome ?? r.Nome ?? r.ativo ?? "").trim() || (r.ticker||"Ativo");

      const ticker = String(r.ticker ?? r.Ticker ?? "").toUpperCase();

      const investido = Number(r.investido ?? ((r.quantidade||0)\*(r.preco||r.valorStock||0)) ?? 0);

      const divAnual  = Number(r.dividendoAnual ?? r.dividendo ?? 0);

      const divHoriz  = Number(r.dividendoHorizonte ?? ((divAnual\*horizonte) || 0));

      const valoriz   = Number(r.valorizacao ?? 0);

      const lucro     = Number(r.lucro ?? ((divHoriz + valoriz) || 0));

      return { nome, ticker, investido, divAnual, divHoriz, valorizacao: valoriz, lucro };

    });

  }

  // 2) KPIs

  const totInvest   = data.reduce((s,a)=>s+a.investido,0);

  const totDivAnual = data.reduce((s,a)=>s+a.divAnual,0);

  const totDivHoriz = data.reduce((s,a)=>s+a.divHoriz,0);

  const totVal      = data.reduce((s,a)=>s+a.valorizacao,0);

  const totLucro    = data.reduce((s,a)=>s+a.lucro,0);

  const retornoPct  = totInvest>0 ? (totLucro/totInvest) : 0;

  // 3) doc

  const doc = new jsPDF({ unit:"pt", format:"a4" });

  const pageW = doc.internal.pageSize.getWidth();

  const pageH = doc.internal.pageSize.getHeight();

  const M = 36; let y = M;

  const COLOR\_PRIMARY=[79,70,229], COLOR\_MUTED=[90,97,110];

  const hoje = new Date();

  const titulo = opts.titulo || "Relatório Financeiro do Portefólio";

  // capa

  doc.setFont("helvetica","bold"); doc.setFontSize(18); doc.setTextColor(...COLOR\_PRIMARY);

  doc.text(titulo, M, y);

  doc.setFont("helvetica","normal"); doc.setFontSize(11); doc.setTextColor(...COLOR\_MUTED);

  doc.text(`Emitido em ${hoje.toLocaleDateString("pt-PT")} — Horizonte: ${horizonte} ${horizonte>1?"períodos":"período"}`, M, y+=18);

  y += 12; doc.setDrawColor(...COLOR\_PRIMARY); doc.setLineWidth(1); doc.line(M, y, pageW-M, y); y += 16;

  // resumo executivo

  doc.setFont("helvetica","bold"); doc.setFontSize(14); doc.setTextColor(20); doc.text("Resumo Executivo", M, y); y+=16;

  const boxW=(pageW-M\*2-24)/2, boxH=64;

  function kpiBox(x,y0,title,value,subtitle){ doc.setDrawColor(230); doc.setFillColor(248); doc.roundedRect(x,y0,boxW,boxH,6,6,"S"); doc.setFont("helvetica","normal"); doc.setFontSize(10); doc.setTextColor(...COLOR\_MUTED); doc.text(title,x+12,y0+18); doc.setFont("helvetica","bold"); doc.setFontSize(13); doc.setTextColor(20); doc.text(value,x+12,y0+36); if(subtitle){ doc.setFontSize(10); doc.setTextColor(...COLOR\_MUTED); doc.text(subtitle,x+12,y0+52); } }

  kpiBox(M, y, "Valor Investido", \_fmtEUR(totInvest));

  kpiBox(M+boxW+24, y, "Retorno Total", `${\_fmtEUR(totLucro)} (${\_pct(retornoPct)})`); y+=boxH+12;

  kpiBox(M, y, "Dividendos (Anual / Horizonte)", `${\_fmtEUR(totDivAnual)} / ${\_fmtEUR(totDivHoriz)}`, `H = ${horizonte}`);

  kpiBox(M+boxW+24, y, "Valorização Projetada", \_fmtEUR(totVal)); y+=boxH+18;

  // gráficos: tenta usar canvases na página; senão cria temporários

  const chartsWanted = [

    { id:"chartDistInvest", title:"Distribuição do Investimento por Ativo (Pizza)" },

    { id:"chartLucroPorAtivo", title:"Lucro Estimado por Ativo (Barras)" },

    { id:"chartDivVsVal", title:"Dividendos vs Valorização por Ativo (Barras Agrupadas)" },

    { id:"chartInvestPorAtivo", title:"Valor Investido por Ativo (Barras)" },

  ];

  let chartImgs = chartsWanted

    .map(c => ({ ...c, img: getChartImageByCanvasId(c.id) }))

    .filter(c => !!c.img);

  if (chartImgs.length === 0 && data.length) {

    chartImgs = await buildTempReportCharts(

      data.map(a => ({

        ticker:a.ticker,

        investido:a.investido,

        lucro:a.lucro,

        divHoriz:a.divHoriz,

        valorizacao:a.valorizacao

      }))

    );

  }

  for (const c of chartImgs) {

    if (y + 280 > pageH - M) { doc.addPage(); y = M; }

    doc.setFont("helvetica","bold"); doc.setFontSize(12); doc.setTextColor(20);

    doc.text(c.title, M, y);

    doc.addImage(c.img, "PNG", M, y+8, pageW - M\*2, 220, undefined, "FAST");

    y += 240;

  }

  // tabela

  if (y+120 > pageH-M) { doc.addPage(); y = M; }

  doc.setFont("helvetica","bold"); doc.setFontSize(14); doc.setTextColor(20); doc.text("Análise Individual por Ativo", M, y); y+=10;

  const head = [[ "Ativo","Ticker","Investido (€)","Div. Anuais (€)","Div. Horizonte (€)","Valorização (€)","Lucro Estimado (€)","% Port." ]];

  const denom = totInvest>0? totInvest : 1;

  const body = data.map(a => [

    a.nome, a.ticker,

    \_fmtEUR(a.investido), \_fmtEUR(a.divAnual), \_fmtEUR(a.divHoriz),

    \_fmtEUR(a.valorizacao), \_fmtEUR(a.lucro),

    ((a.investido/denom)\*100).toFixed(1) + "%"

  ]);

  doc.autoTable({

    startY: y + 10,

    head, body,

    styles:{ font:"helvetica", fontSize:9, cellPadding:4, overflow:"linebreak" },

    headStyles:{ fillColor:[79,70,229], textColor:[255,255,255] },

    columnStyles:{ 0:{cellWidth:120}, 2:{halign:"right"}, 3:{halign:"right"}, 4:{halign:"right"}, 5:{halign:"right"}, 6:{halign:"right"}, 7:{halign:"right"} },

    margin:{ left:M, right:M }

  });

  y = doc.lastAutoTable.finalY + 16;

  // Indicadores-Chave

  if (y+90 > pageH-M) { doc.addPage(); y = M; }

  doc.setFont("helvetica","bold"); doc.setFontSize(14); doc.setTextColor(20); doc.text("Indicadores-Chave", M, y); y += 16;

  doc.setFont("helvetica","normal"); doc.setFontSize(11); doc.setTextColor(50);

  const lines = [

    `Retorno Total: ${\_fmtEUR(totLucro)} (${\_pct(retornoPct)})`,

    `Dividendos Anuais (soma): ${\_fmtEUR(totDivAnual)}`,

    `Valorização no Horizonte: ${\_fmtEUR(totVal)}`,

    `Rácio Dividendos/Valorização (global): ${ totVal>0 ? (totDivHoriz/totVal).toFixed(2) : "—" }`,

  ];

  lines.forEach((t,i) => doc.text(`• ${t}`, M, y + i\*16)); y += lines.length\*16 + 8;

  // rodapé & save

  doc.setFontSize(9); doc.setTextColor(...COLOR\_MUTED);

  doc.text(`© ${new Date().getFullYear()} APPFinance — Relatório gerado automaticamente`, M, pageH - 14);

  const fileName = `Relatorio\_Portefolio\_${new Date().toISOString().slice(0,10)}.pdf`;

  doc.save(fileName);

}

/\* =========================================================

   Heatmap scroll header sync (extra)

   ========================================================= \*/

function hookHeatmapScrollSync() {

  const body = document.getElementById("anlHeatmapBody");

  const head = document.getElementById("anlHeatmapHeaderScroll");

  if (!body || !head) return;

  body.addEventListener(

    "scroll",

    () => {

      head.scrollLeft = body.scrollLeft;

    },

    { passive: true }

  );

}

/\* =========================================================

   INIT

   ========================================================= \*/

export async function initScreen() {

  await ensureChartJS();

  if (!db) {

    console.error("Firebase DB não inicializado!");

    return;

  }

  await fetchAcoes();

  populateFilters();

  // Ordenação

  document.querySelectorAll("#anlTable thead th.sortable").forEach((th) => {

    th.addEventListener("click", () => {

      const key = th.getAttribute("data-sort");

      if (!key) return;

      if (sortKey === key) sortDir = sortDir === "asc" ? "desc" : "asc";

      else {

        sortKey = key;

        sortDir = key === "pe" ? "asc" : "desc";

      }

      markSortedHeader();

      applyFilters();

    });

  });

  // Filtros

  document.getElementById("anlSearch")?.addEventListener("input", applyFilters);

  document.getElementById("anlSetor")?.addEventListener("change", applyFilters);

  document

    .getElementById("anlMercado")

    ?.addEventListener("change", applyFilters);

  document

    .getElementById("anlPeriodo")

    ?.addEventListener("change", applyFilters);

  document.getElementById("anlReset")?.addEventListener("click", () => {

    document.getElementById("anlSearch").value = "";

    document.getElementById("anlSetor").value = "";

    document.getElementById("anlMercado").value = "";

    document.getElementById("anlPeriodo").value = "";

    applyFilters();

  });

  // Selecionar todos (da lista filtrada)

  document.getElementById("anlSelectAll")?.addEventListener("change", (e) => {

    const check = e.target.checked;

    const term = keyStr(document.getElementById("anlSearch")?.value || "");

    const setor = document.getElementById("anlSetor")?.value || "";

    const mercado = document.getElementById("anlMercado")?.value || "";

    const periodo = document.getElementById("anlPeriodo")?.value || "";

    let rows = [...ALL\_ROWS];

    if (term)

      rows = rows.filter(

        (r) => keyStr(r.ticker).includes(term) || keyStr(r.nome).includes(term)

      );

    if (setor) rows = rows.filter((r) => r.setor === setor);

    if (mercado) rows = rows.filter((r) => r.mercado === mercado);

    if (periodo) rows = rows.filter((r) => (r.periodicidade || "") === periodo);

    rows.forEach((r) => {

      if (check) selectedTickers.add(r.ticker);

      else selectedTickers.delete(r.ticker);

    });

    updateSelCount();

    renderTable(sortRows(rows));

  });

  document.getElementById("anlClearSel")?.addEventListener("click", () => {

    selectedTickers.clear();

    updateSelCount();

    applyFilters();

  });

  // Abrir modal + pizza dos selecionados

  document.getElementById("anlSimular")?.addEventListener("click", async () => {

    if (selectedTickers.size === 0) {

      alert("Seleciona pelo menos uma ação para simular.");

      return;

    }

    const selecionadas = ALL\_ROWS.filter((r) => selectedTickers.has(r.ticker));

    await renderSelectedSectorChart(selecionadas);

    openSimModal();

  });

  // Fechar modal

  document

    .getElementById("anlSimClose")

    ?.addEventListener("click", closeSimModal);

  document.getElementById("anlSimModal")?.addEventListener("click", (e) => {

    if (e.target.id === "anlSimModal") closeSimModal();

  });

  // Exclusividade Total vs Inteiros

  const cbTotal = document.getElementById("anlSimInvestirTotal");

  const cbInteiro = document.getElementById("anlSimInteiros");

  cbTotal?.addEventListener("change", () => {

    if (cbTotal.checked) cbInteiro && (cbInteiro.checked = false);

    else cbInteiro && (cbInteiro.checked = true);

  });

  cbInteiro?.addEventListener("change", () => {

    if (cbInteiro.checked) cbTotal && (cbTotal.checked = false);

    else cbTotal && (cbTotal.checked = true);

  });

  // Calcular simulação

  document

    .getElementById("anlSimCalcular")

    ?.addEventListener("click", async () => {

      const investimento = Number(

        document.getElementById("anlSimInvest")?.value || 0

      );

      const horizonte = Number(

        document.getElementById("anlSimHoriz")?.value || 1

      );

      const periodo = document.getElementById("anlSimPeriodo")?.value || "1m";

      const incluirDiv = !!document.getElementById("anlSimIncluiDiv")?.checked;

      const usarFracoes = !!document.getElementById("anlSimInvestirTotal")

        ?.checked;

      const apenasInteiros =

        !!document.getElementById("anlSimInteiros")?.checked;

      const modoEstrito = !!document.getElementById("anlSimEstrito")?.checked;

      if (!(investimento > 0)) {

        alert("Indica um investimento total válido.");

        return;

      }

      const selecionadas = ALL\_ROWS.filter((r) =>

        selectedTickers.has(r.ticker)

      );

      let candidatos = prepararCandidatos(selecionadas, {

        periodo,

        horizonte,

        incluirDiv,

        modoEstrito,

      });

      if (candidatos.length === 0) {

        alert(

          "Nenhum ativo com retorno positivo ou dados válidos para este cenário."

        );

        return;

      }

      const res =

        apenasInteiros && !usarFracoes

          ? distribuirInteiros\_porScore(candidatos, investimento)

          : distribuirFracoes\_porScore(candidatos, investimento);

      await renderSelectedSectorChart(selecionadas);

      renderResultadoSimulacao(res);

      // --- Normaliza linhas da simulação para o Relatório (v2) ---

      const linhasReport = (res.linhas || [])

        .filter((l) => l.quantidade > 0 && l.investido > 0)

        .map((l) => ({

          nome: l.nome,

          ticker: l.ticker,

          investido: l.investido,

          // ATENÇÃO: para o relatório v2, estes nomes são os esperados:

          dividendoAnual: l.divAnualAlloc, // soma anual alocada

          dividendoHorizonte: l.divPeriodoAlloc, // no horizonte H

          valorizacao: l.valorizAlloc, // no horizonte H

          lucro: l.divPeriodoAlloc + l.valorizAlloc, // lucro total estimado

        }));

      \_\_ANL\_LAST\_SIM = {

        rows: linhasReport,

        opts: { horizonte, periodo, incluirDiv, investimento },

      };

    });

  // Relatório (PDF) — substitui “Exportar para simulador”

  // Relatório (PDF)

  const relBtn =

    document.getElementById("anlSimRelatorio") ||

    document.getElementById("btnRelatorio");

  relBtn?.addEventListener("click", async () => {

    // Se houver simulação feita, usa-a; senão, usa seleção crua

    let rowsForReport = \_\_ANL\_LAST\_SIM?.rows?.length

      ? \_\_ANL\_LAST\_SIM.rows

      : ALL\_ROWS.filter((r) => selectedTickers.has(r.ticker));

    if (!rowsForReport.length) {

      alert(

        "Seleciona pelo menos uma ação (e idealmente executa a simulação)."

      );

      return;

    }

    const horizonte = Number(

      document.getElementById("anlSimHoriz")?.value ||

        \_\_ANL\_LAST\_SIM?.opts?.horizonte ||

        1

    );

    try {

      await generateReportPDF\_v2(rowsForReport, {

        titulo: "Relatório Financeiro (v2)",

        horizonte,

      });

    } catch (e) {

      console.error("[relatorio] falhou:", e);

      alert("Não consegui gerar o PDF. Vê a consola para mais detalhes.");

    }

  });

  // Render inicial

  markSortedHeader();

  applyFilters();

}

/\* Auto-init seguro \*/

if (!window.\_\_ANL\_AUTO\_INIT\_\_) {

  window.\_\_ANL\_AUTO\_INIT\_\_ = true;

  initScreen().catch((e) => {

    console.error("[analise] init error", e);

  });

}