JS | SIMULADOR

// js/screens/simulador.js

// Requer Chart.js incluído na página (ex.: <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>)

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

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

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

   ESTADO

   ========================= \*/

let simulacoes = [];

let grafico = null;

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

   HELPERS GERAIS

   ========================= \*/

function setScreenTitleIfAvailable() {

  if (typeof window.setScreenTitle === "function") {

    window.setScreenTitle("Simulador");

  }

}

function toNumber(v) {

  const n = parseFloat(String(v ?? "").replace(",", "."));

  return isNaN(n) ? 0 : n;

}

function euro(v){ return new Intl.NumberFormat("pt-PT",{style:"currency",currency:"EUR"}).format(v||0); }

function clamp(v,min,max){ return Math.max(min, Math.min(max, v)); }

function limparInputsSimulacao() {

  ["nomeAcao","tp1","tp2","investimento","dividendo"].forEach(id => {

    const el = document.getElementById(id);

    if (el) el.value = "";

  });

}

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

   SIMULAÇÃO + GRÁFICO

   ========================= \*/

function guardarSimulacao({ nomeAcao, tp1, tp2, valorInvestido, dividendo = 0 }) {

  const crescimento = tp1 > 0 ? ((tp2 - tp1) / tp1) \* 100 : 0;

  const numeroAcoes = tp1 > 0 ? valorInvestido / tp1 : 0;

  const lucroValorizacao = (tp2 - tp1) \* numeroAcoes;

  const lucroDividendos  = numeroAcoes \* dividendo;

  const lucroTotal       = lucroValorizacao + lucroDividendos;

  const novaSimulacao = {

    nomeAcao: (nomeAcao || "—").trim(),

    tp1: Number(tp1.toFixed(2)),

    tp2: Number(tp2.toFixed(2)),

    valorInvestido: Number(valorInvestido.toFixed(2)),

    lucro: Number(lucroTotal.toFixed(2)),

    crescimentoPercentual: Number(crescimento.toFixed(2)),

  };

  simulacoes.push(novaSimulacao);

  atualizarTabela();

  atualizarGrafico();

}

function atualizarTabela() {

  const corpo = document.querySelector("#tabelaSimulacoes tbody");

  if (!corpo) return;

  corpo.innerHTML = "";

  simulacoes.forEach((sim, index) => {

    const tr = document.createElement("tr");

    tr.innerHTML = `

      <td>${sim.nomeAcao}</td>

      <td>${sim.tp1.toFixed(2)}</td>

      <td>${sim.tp2.toFixed(2)}</td>

      <td>${sim.valorInvestido.toFixed(2)}</td>

      <td>${sim.lucro.toFixed(2)}</td>

      <td>${sim.crescimentoPercentual.toFixed(2)}%</td>

      <td>

        <button class="btn outline btn-remove" data-index="${index}">❌</button>

      </td>

      <td>

        <input type="checkbox" class="checkbox-lucro" data-lucro="${sim.lucro}">

      </td>

    `;

    corpo.appendChild(tr);

  });

  // linha total (0 por defeito; atualiza quando marcarem checkboxes)

  mostrarTotalLucro(0);

  // Se clicarem nas checkboxes, recalcula automático

  corpo.querySelectorAll(".checkbox-lucro").forEach(cb => {

    cb.addEventListener("change", atualizarSomaLucros);

  });

}

function removerSimulacao(index) {

  simulacoes.splice(index, 1);

  atualizarTabela();

  atualizarGrafico();

}

function atualizarSomaLucros() {

  const checkboxes = document.querySelectorAll(".checkbox-lucro");

  let total = 0;

  checkboxes.forEach(cb => {

    if (cb.checked) total += toNumber(cb.dataset.lucro);

  });

  mostrarTotalLucro(total);

}

function mostrarTotalLucro(valor) {

  const corpo = document.querySelector("#tabelaSimulacoes tbody");

  if (!corpo) return;

  let totalRow = document.getElementById("linha-total-lucro");

  if (!totalRow) {

    totalRow = document.createElement("tr");

    totalRow.id = "linha-total-lucro";

    totalRow.innerHTML = `

      <td colspan="4"><strong>Total Lucro Selecionado:</strong></td>

      <td colspan="4" id="valorTotalLucro"><strong>${valor.toFixed(2)} €</strong></td>

    `;

    corpo.appendChild(totalRow);

  } else {

    totalRow.querySelector("#valorTotalLucro").innerHTML =

      `<strong>${valor.toFixed(2)} €</strong>`;

  }

}

function atualizarGrafico() {

  const canvas = document.getElementById("graficoLucro");

  if (!canvas) return;

  const labels = simulacoes.map(s => s.nomeAcao);

  const dados  = simulacoes.map(s => s.lucro);

  if (grafico) grafico.destroy();

  const ctx = canvas.getContext("2d");

  grafico = new Chart(ctx, {

    type: "bar",

    data: {

      labels,

      datasets: [

        {

          label: "Lucro (€)",

          data: dados,

          backgroundColor: dados.map(v => v >= 0 ? "rgba(46, 204, 113, 0.6)" : "rgba(231, 76, 60, 0.6)"),

          borderColor:     dados.map(v => v >= 0 ? "rgba(46, 204, 113, 1)"   : "rgba(231, 76, 60, 1)"),

          borderWidth: 1

        }

      ],

    },

    options: {

      responsive: true,

      maintainAspectRatio: false,

      animation: { duration: 300 },

      layout: { padding: 0 },

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

      scales: { y: { beginAtZero: true } },

    },

  });

}

function simularEGUardar() {

  document.querySelector(".tabela-scroll-wrapper")?.classList.remove("hidden");

  const nome         = document.getElementById("nomeAcao")?.value?.trim();

  const tp1          = toNumber(document.getElementById("tp1")?.value);

  const tp2          = toNumber(document.getElementById("tp2")?.value);

  const investimento = toNumber(document.getElementById("investimento")?.value);

  const dividendo    = toNumber(document.getElementById("dividendo")?.value);

  if (!nome || tp1 <= 0 || tp2 <= 0 || investimento <= 0) {

    alert("Preenche todos os campos com valores > 0!");

    return;

  }

  guardarSimulacao({ nomeAcao: nome, tp1, tp2, valorInvestido: investimento, dividendo });

  limparInputsSimulacao(); // limpa inputs mas mantém tabela/gráfico

}

function limparGrafico() {

  simulacoes = [];

  atualizarTabela();

  if (grafico) {

    grafico.destroy();

    grafico = null;

  }

}

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

   REFORÇO (MÉDIA PONDERADA)

   ========================= \*/

function calcularMediaPonderada() {

  const invest1 = toNumber(document.getElementById("invest1")?.value);

  const preco1  = toNumber(document.getElementById("preco1")?.value);

  const invest2 = toNumber(document.getElementById("invest22")?.value);

  const preco2  = toNumber(document.getElementById("preco2")?.value);

  const out = document.getElementById("resultadoReforco");

  if (invest1 > 0 && preco1 > 0 && invest2 > 0 && preco2 > 0) {

    const qtd1 = invest1 / preco1;

    const qtd2 = invest2 / preco2;

    const totalQtd = qtd1 + qtd2;

    const totalInvestido = invest1 + invest2;

    const precoMedio = totalInvestido / totalQtd;

    out.innerHTML = `

      <p>📊 <strong>Preço Médio:</strong> ${precoMedio.toFixed(2)} €</p>

      <p>📦 <strong>Total de Ações:</strong> ${totalQtd.toFixed(2)}</p>

      <p>💰 <strong>Total Investido:</strong> ${totalInvestido.toFixed(2)} €</p>

    `;

  } else {

    out.innerHTML = `<p style="color:red;">⚠️ Insere valores válidos.</p>`;

  }

}

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

   TP2 (alvo para lucro desejado)

   ========================= \*/

// Fórmula: n = investimento / tp1 ; tp2 = tp1 + lucroDesejado / n

function calcularTP2() {

  const tp1   = toNumber(document.getElementById("tp1Input")?.value);

  const inv   = toNumber(document.getElementById("investimentoInput")?.value);

  const lucro = toNumber(document.getElementById("lucroDesejadoInput")?.value);

  const out = document.getElementById("resultadoTP2");

  if (tp1 <= 0 || inv <= 0 || lucro <= 0) {

    out.innerHTML = `<p style="color:red;">⚠️ Preenche TP1, Investimento e Lucro Desejado com valores > 0.</p>`;

    return;

  }

  const nAcoes = inv / tp1;

  const tp2 = tp1 + (lucro / nAcoes);

  out.innerHTML = `

    <p>🎯 <strong>TP2 necessário:</strong> ${tp2.toFixed(2)} €</p>

    <small>(${nAcoes.toFixed(2)} ações estimadas)</small>

  `;

}

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

   TOP 10 — Distribuição

   ========================= \*/

// === MOTOR LOCAL (não depende do analise.js) ===

// === MOTOR LOCAL (TOP 10) — configurações mais realistas ===

const TOP\_CFG = {

  // Máximo/mínimo anual admitido depois da anualização

  MAX\_ANNUAL\_RETURN: 0.35,   // antes 0.8 → agora 35%/ano

  MIN\_ANNUAL\_RETURN: -0.5,   // -50%/ano

  // Limite prudente por ticker (quando fracionado/inteiras)

  CAP\_PCT\_POR\_TICKER: 0.30,  // 30% do capital por ativo

  // Blends por período (mantém)

  BLEND\_WEIGHTS: {

    "1s": { w: 0.60, m: 0.25, y: 0.15 }, // um pouco menos peso na semana

    "1m": { w: 0.15, m: 0.65, y: 0.20 },

    "1a": { w: 0.10, m: 0.20, y: 0.70 },

  },

  // Se a taxa "primária" for alta, capamos o blend

  REALISM\_CAP: { enabled: true, trigger: 0.35, cap: 0.25 },

  // se a taxa primária ≥35%/ano, capar o blend a 25%/ano

};

const clamp2 = (v, min, max) => Math.max(min, Math.min(max, v));

const toNum   = (x) => Number(x || 0);

// === Render do resultado TOP 10 (drop-in) ===

function renderResultado(destEl, resultado, opts){

  const { linhas, totalLucro, totalGasto, restante=0 } = resultado;

  if (!linhas || linhas.length===0){

    destEl.innerHTML = `<div class="card"><p class="muted">Sem ações elegíveis com retorno positivo para o período selecionado.</p></div>`;

    return;

  }

  const rows = linhas.map(l=>`

    <tr>

      <td>${l.nome} <span class="muted">(${l.ticker})</span></td>

      <td>${euro(l.preco)}</td>

      <td>${l.quantidade.toFixed( opts.acoesCompletas ? 0 : 4 )}</td>

      <td>${euro(l.investido)}</td>

      <td>${euro(l.lucro)}</td>

      <td>${(l.taxaPct||0).toFixed(2)}%</td>

      <td>${euro(l.dividendoAnual||0)}/ano</td>

    </tr>

  `).join("");

  destEl.innerHTML = `

    <div class="card">

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

        <table style="width:100%; border-collapse:collapse;">

          <thead>

            <tr>

              <th>Ativo</th>

              <th>Preço</th>

              <th>Qtd</th>

              <th>Investido</th>

              <th>Lucro Estim.</th>

              <th>Tx ${opts.periodoSel}</th>

              <th>Dividendo</th>

            </tr>

          </thead>

          <tbody>${rows}</tbody>

        </table>

      </div>

      <p style="margin-top:.6rem">

        <strong>Total investido:</strong> ${euro(totalGasto)}

        ${opts.acoesCompletas && restante>0 ? `· <strong>Resto:</strong> ${euro(restante)}` : ``}

        <br/>

        <strong>Lucro total estimado (${opts.horizonte} ${opts.horizonte>1?"períodos":"período"}):</strong> ${euro(totalLucro)}

      </p>

      <p class="muted" style="margin-top:.3rem">

        ${opts.incluirDiv === false ? "Dividendo excluído do cálculo." : "Dividendo incluído no cálculo."}

      </p>

    </div>

  `;

}

function annualizeRate\_TOP(row, periodoSel) {

  const asRate = (x) => {

    const n = Number(x);

    if (!Number.isFinite(n)) return 0;

    return Math.abs(n) > 1 ? n / 100 : n; // aceita % ou fração

  };

  const w = asRate(row?.g1w);

  const m = asRate(row?.g1m);

  const y = asRate(row?.g1y);

  const rw = Math.pow(1 + (w || 0), 52) - 1; // 1s → ano

  const rm = Math.pow(1 + (m || 0), 12) - 1; // 1m → ano

  const ry = y || 0;                         // 1a → ano

  const clampAnnual = (r) => clamp2(r, TOP\_CFG.MIN\_ANNUAL\_RETURN, TOP\_CFG.MAX\_ANNUAL\_RETURN);

  const Rw = clampAnnual(rw), Rm = clampAnnual(rm), Ry = clampAnnual(ry);

  const BW = TOP\_CFG.BLEND\_WEIGHTS[periodoSel] || TOP\_CFG.BLEND\_WEIGHTS["1m"];

  const r\_blend = (BW.w \* Rw) + (BW.m \* Rm) + (BW.y \* Ry);

  // corta exageros se a taxa primária for “irrealista”

  const r\_primary = (periodoSel === "1s") ? Rw : (periodoSel === "1m") ? Rm : Ry;

  if (TOP\_CFG.REALISM\_CAP?.enabled && r\_primary >= TOP\_CFG.REALISM\_CAP.trigger) {

  return Math.min(r\_blend, TOP\_CFG.REALISM\_CAP.cap);

  }

  return r\_blend;

}

function anualizarDividendo\_TOP(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\_TOP(doc){

  const d24 = toNum(doc.dividendoMedio24m);

  if (d24 > 0) return d24;

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

}

function calcularMetricasBase\_TOP(acao, { periodo="1m", horizonte=1, incluirDiv=true } = {}){

  const precoAtual = toNum(acao.valorStock);

  if (!(precoAtual > 0)) return null;

  const anualDiv = toNum(acao.divAnual ?? anualPreferido\_TOP(acao));

  const rAnnual = annualizeRate\_TOP(acao, periodo);

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

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

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

  const lucroUnidade = dividendos + valorizacao;

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

  return {

    preco: precoAtual,

    dividendoAnual: anualDiv,

    taxaPct: rAnnual \* 100,

    totalDividendos: dividendos,

    valorizacao,

    lucroUnidade,

    retornoPorEuro,

  };

}

// “Lucro Máximo” = score baseado só em retorno/€

function prepararCandidatos\_TOP(rows, { periodo, horizonte, incluirDiv, modoEstrito=true }){

  let cands = rows.map(a => {

      const metrics = calcularMetricasBase\_TOP(a, { periodo, horizonte, incluirDiv });

      return metrics ? { ...a, metrics } : null;

  }).filter(Boolean)

    .filter(c => 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(rets.sort((x,y)=>x-y)[Math.floor((rets.length-1)\*0.99)] || 1, 1e-9);

  cands = cands.map(c => {

    const R = clamp2(c.metrics.retornoPorEuro / p99, 0, 1);

    const score = modoEstrito ? R : R; // (se no futuro quiseres outro score, muda aqui)

    return { ...c, score, \_\_R: R };

  }).filter(c => c.score > 0);

  return cands;

}

function makeLinha\_TOP(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\_TOP(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),

  };

}

// FRACIONADO (proporcional ao score) com cap por ticker

function distribuirFracoes\_porScore\_TOP(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 = (TOP\_CFG.CAP\_PCT\_POR\_TICKER ?? 0.35) \* investimento;

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

  const alvos = new Map(ord.map(c => [c, (c.score/somaScore)\*investimento]));

  const alloc = new Map(); let restante = 0;

  for (const c of ord){

    const alvo = alvos.get(c) || 0;

    const investido = Math.min(alvo, capAbs);

    alloc.set(c, investido);

    if (alvo > capAbs) restante += (alvo - capAbs);

  }

  let progress = true;

  while (restante > 1e-6 && progress){

    progress = false;

    const elig = ord.filter(c => (alloc.get(c)||0) + 1e-9 < capAbs);

    if (!elig.length) break;

    const somaElig = elig.reduce((s,c)=>s+c.score,0) || 1;

    for (const c of elig){

      if (restante <= 1e-6) break;

      const share = (c.score/somaElig) \* restante;

      const margem = Math.max(0, capAbs - (alloc.get(c)||0));

      const add = Math.min(share, margem);

      if (add > 1e-6){

        alloc.set(c, (alloc.get(c)||0) + add);

        restante -= add;

        progress = true;

      }

    }

  }

  const linhas = [];

  let gasto = 0;

  for (const c of ord){

    const investido = alloc.get(c) || 0;

    if (investido <= 0) continue;

    const qtd = (c.metrics.preco > 0) ? investido / c.metrics.preco : 0;

    if (qtd <= 0) continue;

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

    gasto += investido;

  }

  return sumarizar\_TOP(linhas, investimento, gasto);

}

// INTEIROS (cap prudente + guloso por retorno/€)

function distribuirInteiros\_porScore\_capped\_TOP(cands, invest){

  if (!(invest > 0) || !Array.isArray(cands) || cands.length === 0) {

    return { linhas: [], totalLucro: 0, totalGasto: 0, restante: invest };

  }

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

  const n = ord.length;

  const capTop = 0.35 \* invest;

  // targets prudentes

  const targets = new Map();

  if (n === 1){

    targets.set(ord[0], Math.min(capTop, invest));

  } else if (n === 2){

    targets.set(ord[0], 0.65 \* invest);

    targets.set(ord[1], 0.35 \* invest);

  } else {

    const top = ord[0], rest = ord.slice(1);

    const s = rest.reduce((x,c)=>x+c.score,0) || 1;

    const topT = Math.min(capTop, invest);

    const rem = Math.max(0, invest - topT);

    targets.set(top, topT);

    for (const c of rest) targets.set(c, rem \* (c.score / s));

  }

  const base = ord.map(c=>{

    const p = c.metrics.preco, alvo = targets.get(c) || 0;

    const q = Math.max(0, Math.floor(p>0 ? (alvo/p) : 0));

    return { c, q };

  });

  // ✅ bug fix: usar x.c

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

  let restante = invest - gasto;

  while (restante >= Math.min(...ord.map(c=>c.metrics.preco)) - 1e-9){

    let best = null, bestR = -Infinity;

    for (const c of ord){

      const p = c.metrics.preco;

      if (p > restante + 1e-9) continue;

      const invNow = (base.find(x=>x.c===c)?.q || 0) \* p;

      if (invNow + p > capTop + 1e-9) continue;

      const rpe = (c.metrics.lucroUnidade || 0) / p;

      if (rpe > bestR){ bestR = rpe; best = c; }

    }

    if (!best) break;

    const rec = base.find(x=>x.c===best);

    rec.q += 1;

    gasto += best.metrics.preco;

    restante = invest - gasto;

  }

  const linhas = base.filter(x=>x.q>0).map(x=>makeLinha\_TOP(x.c, x.q));

  return sumarizar\_TOP(linhas, invest, gasto);

}

async function fetchAcoesBase(){

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

  const out = [];

  snap.forEach(doc=>{

    const d = doc.data(); if (!d || !d.ticker) return;

    out.push({

      nome: d.nome || d.ticker,

      ticker: String(d.ticker).toUpperCase(),

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

      // dividendos: podes ter anual direto (dividendoMedio24m) ou por pagamento+periodicidade

      dividendoMedio24m: Number(d.dividendoMedio24m || 0),

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

      periodicidade: d.periodicidade || "Anual",

      // crescimento → nomes esperados pelo motor local

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

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

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

      raw: d

    });

  });

  return out;

}

async function distribuirInvestimento(opts){

  const { investimento, periodoSel, horizonte, acoesCompletas, usarTotal, incluirDiv } = opts;

  const periodo = (periodoSel === "1ano") ? "1a" : periodoSel;

  const base = await fetchAcoesBase();

  const cands = prepararCandidatos\_TOP(base, {

    periodo,

    horizonte: Number(horizonte || 1),

    incluirDiv: (incluirDiv ?? true),  // ← aqui

    modoEstrito: !!usarTotal           // “Lucro Máximo” quando ligado

  });

  if (!cands.length){

    return { linhas: [], totalLucro: 0, totalGasto: 0, restante: investimento };

  }

  const TOP\_N = 10;

  const universo = [...cands].sort((a,b)=>b.score - a.score).slice(0, TOP\_N);

  return acoesCompletas

    ? distribuirInteiros\_porScore\_capped\_TOP(universo, investimento)

    : distribuirFracoes\_porScore\_TOP(universo, investimento);

}

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

   EMAIL (mailto)

   ========================= \*/

function enviarEmailResumo() {

  const emailDestino = prompt("Para que email queres enviar o resumo?");

  if (!emailDestino) return;

  if (simulacoes.length === 0) {

    alert("Faz pelo menos uma simulação primeiro.");

    return;

  }

  const assunto = encodeURIComponent("Resumo de Simulações Financeiras");

  let corpo = "Resumo das Simulações:\n\n";

  simulacoes.forEach((s, i) => {

    corpo += `Simulação ${i + 1}:\n`;

    corpo += `Ação: ${s.nomeAcao}\n`;

    corpo += `TP1: €${s.tp1.toFixed(2)}\n`;

    corpo += `TP2: €${s.tp2.toFixed(2)}\n`;

    corpo += `Investimento: €${s.valorInvestido.toFixed(2)}\n`;

    corpo += `Lucro: €${s.lucro.toFixed(2)}\n`;

    corpo += `Crescimento: ${s.crescimentoPercentual.toFixed(2)}%\n\n`;

  });

  const body = encodeURIComponent(corpo);

  const mailtoLink = `mailto:${encodeURIComponent(emailDestino)}?subject=${assunto}&body=${body}`;

  window.location.href = mailtoLink;

}

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

   INIT + Wiring UI

   ========================= \*/

export function initScreen() {

  setScreenTitleIfAvailable();

  // Alternância de painéis (se usares sidebar + content)

  const buttons = document.querySelectorAll(".sim-sidebar .btn[data-target]");

  const panels = document.querySelectorAll(".sim-content .panel");

  function activatePanel(id) {

    panels.forEach(p => p.classList.remove("active"));

    const t = document.getElementById(id);

    if (t) {

      t.classList.add("active");

      if (window.matchMedia("(max-width: 820px)").matches) {

        t.scrollIntoView({ behavior: "smooth", block: "start" });

      }

    }

  }

  buttons.forEach(btn => {

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

      const targetId = btn.getAttribute("data-target");

      activatePanel(targetId);

    });

  });

  // Quick amount

  document.querySelectorAll("[data-quick]").forEach(el => {

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

      const v = toNumber(el.getAttribute("data-quick"));

      const investInput = document.getElementById("investimento");

      if (investInput) investInput.value = v;

    });

  });

  // Simular com gráfico

  document.getElementById("btnSimularGrafico")?.addEventListener("click", simularEGUardar);

  // 🔹 Limpar só inputs (NÃO mexe em tabela/gráfico)

  document.getElementById("btnLimparInputs")?.addEventListener("click", limparInputsSimulacao);

  // 🔹 Limpar gráfico + tabela (tudo)

  document.getElementById("btnLimparGrafico")?.addEventListener("click", limparGrafico);

  // Enviar email

  document.getElementById("btnEnviarEmail")?.addEventListener("click", enviarEmailResumo);

  // Delegation: remover linha + checkboxes

  document.querySelector("#tabelaSimulacoes tbody")?.addEventListener("click", (e) => {

    const rm = e.target.closest(".btn-remove");

    if (rm) {

      const idx = parseInt(rm.dataset.index, 10);

      if (!isNaN(idx)) removerSimulacao(idx);

    }

  });

  // Reforço (média ponderada)

  document.getElementById("btnCalcularReforco")?.addEventListener("click", calcularMediaPonderada);

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

    ["invest1","preco1","invest22","preco2"].forEach(id => {

      const el = document.getElementById(id);

      if (el) el.value = "";

    });

    const out = document.getElementById("resultadoReforco");

    if (out) out.innerHTML = "";

  });

  // TP2

  document.getElementById("btnCalcularTP2")?.addEventListener("click", calcularTP2);

  document.getElementById("btnLimparTP2")?.addEventListener("click", () => {

    ["tp1Input","investimentoInput","lucroDesejadoInput"].forEach(id => {

      const el = document.getElementById(id);

      if (el) el.value = "";

    });

    const out = document.getElementById("resultadoTP2");

    if (out) out.innerHTML = "";

  });

  // === TOP 10 ===

  // estado inicial das checkboxes

  const chkUsarTotal = document.getElementById("chkUsarTotal");

  const chkAcoesCompletas = document.getElementById("chkAcoesCompletas");

  if (chkUsarTotal) chkUsarTotal.checked = true;

  if (chkAcoesCompletas) chkAcoesCompletas.checked = false;

  // exclusividade

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

    if (chkUsarTotal.checked) chkAcoesCompletas.checked = false;

  });

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

    if (chkAcoesCompletas.checked) chkUsarTotal.checked = false;

  });

  // simular

  document.getElementById("btnSimularTop10")?.addEventListener("click", async () => {

  const investimento = Number(document.getElementById("inputInvestimento")?.value || 0);

  const periodoSel   = (document.getElementById("inputPeriodo")?.value || "1ano");

  const horizonte    = Math.max(1, Number(document.getElementById("inputHorizonte")?.value || 1));

  const usarTotal      = !!document.getElementById("chkUsarTotal")?.checked;

  const acoesCompletas = !!document.getElementById("chkAcoesCompletas")?.checked;

  const incluirDiv     = document.getElementById("chkIncluirDiv")

                          ? !!document.getElementById("chkIncluirDiv").checked

                          : true; // default: inclui dividendos

  if (!investimento || investimento <= 0){

    alert("Indica o montante a investir.");

    return;

  }

  const opts = { investimento, periodoSel, horizonte, usarTotal, acoesCompletas, incluirDiv };

  const box = document.getElementById("resultadoSimulacao");

  if (box) box.innerHTML = `<div class="card">A simular…</div>`;

  try{

    const resultado = await distribuirInvestimento(opts);

    if (box) renderResultado(box, resultado, opts);

  }catch(err){

    console.error(err);

    if (box) box.innerHTML = `<div class="card"><p class="muted">Ocorreu um erro na simulação.</p></div>`;

  }

});

  // limpar

  document.getElementById("btnLimparTop10")?.addEventListener("click", () => {

    const investEl = document.getElementById("inputInvestimento");

    const perEl = document.getElementById("inputPeriodo");

    const horEl = document.getElementById("inputHorizonte");

    const box = document.getElementById("resultadoSimulacao");

    if (investEl) investEl.value = "";

    if (perEl) perEl.value = "1ano";

    if (horEl) horEl.value = 1;

    if (chkUsarTotal) chkUsarTotal.checked = true;

    if (chkAcoesCompletas) chkAcoesCompletas.checked = false;

    if (box) box.innerHTML = "";

  });

}