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

   ========================= \*/

/\* mapeamento do período \*/

function campoCrescimento(periodoSel){

  if (periodoSel === "1s")  return "taxaCrescimento\_1semana";

  if (periodoSel === "1m")  return "taxaCrescimento\_1mes";

  return "taxaCrescimento\_1ano";

}

function melhorTaxaDisponivel(acao, prefer){

  const ordem = prefer === "taxaCrescimento\_1ano"

    ? ["taxaCrescimento\_1ano","taxaCrescimento\_1mes","taxaCrescimento\_1semana"]

    : [prefer,"taxaCrescimento\_1mes","taxaCrescimento\_1semana","taxaCrescimento\_1ano"];

  for (const k of ordem){

    const v = Number(acao[k] || 0);

    if (v !== 0) return v;

  }

  return 0;

}

function dividirPeriodicidade(dividendo, periodicidade){

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

  if (p === "mensal")     return dividendo \* 12;

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

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

  return dividendo; // anual ou n/a

}

function calcularMetricasAcao(acao, periodoSel, horizonte){

  const prefer = campoCrescimento(periodoSel);

  const taxaPct = melhorTaxaDisponivel(acao.raw || acao, prefer);

  const preco     = Number(acao.valorStock || 0);

  const dividendo = Number(acao.dividendo || 0);

  const per       = acao.periodicidade || "";

  if (!(preco>0)) return null;

  const r = clamp(taxaPct/100, -0.95, 5);  // segurança

  const dividendoAnual = dividirPeriodicidade(dividendo, per);

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

  const mult = Math.pow(1+r, h);

  const valorizacao = preco \* (mult - 1);

  const totalDividendos = dividendoAnual \* h;

  const lucroUnidade = totalDividendos + valorizacao;

  const retornoPorEuro = lucroUnidade / preco;

  return { preco, dividendoAnual, taxaPct, mult, lucroUnidade, retornoPorEuro };

}

/\* distribuição fracionada (proporcional) \*/

function distribuirFracoes(acoes, investimento){

  const somaRetorno = acoes.reduce((s,a)=>s + a.metrics.retornoPorEuro, 0);

  if (somaRetorno <= 0) return { linhas: [], totalLucro: 0, totalGasto: 0, restante: investimento };

  const linhas = acoes.map(a=>{

    const propor = a.metrics.retornoPorEuro / somaRetorno;

    const investido = investimento \* propor;

    const qtd = investido / a.metrics.preco;

    const lucro = qtd \* a.metrics.lucroUnidade;

    return {

      nome: a.nome, ticker: a.ticker,

      preco: a.metrics.preco,

      quantidade: qtd,

      investido,

      lucro,

      taxaPct: a.metrics.taxaPct,

      dividendoAnual: a.metrics.dividendoAnual

    };

  });

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

  const totalGasto = linhas.reduce((s,l)=>s+l.investido,0);

  return { linhas, totalLucro, totalGasto, restante: Math.max(0, investimento - totalGasto) };

}

/\* distribuição por ações inteiras (guloso) \*/

function distribuirInteiras(acoes, investimento){

  const ordenadas = [...acoes].sort((a,b)=>b.metrics.retornoPorEuro - a.metrics.retornoPorEuro);

  const linhasMap = new Map(); // ticker -> linha acumulada

  let restante = investimento;

  const precoMin = Math.min(...ordenadas.map(a=>a.metrics.preco));

  while (restante >= precoMin - 1e-9){

    // escolhe a melhor que caiba agora

    let best = null;

    for (const a of ordenadas){

      if (a.metrics.preco <= restante + 1e-9){ best = a; break; }

    }

    if (!best) break;

    const key = best.ticker;

    const linha = linhasMap.get(key) || {

      nome: best.nome, ticker: best.ticker,

      preco: best.metrics.preco,

      quantidade: 0, investido: 0, lucro: 0,

      taxaPct: best.metrics.taxaPct,

      dividendoAnual: best.metrics.dividendoAnual

    };

    linha.quantidade += 1;

    linha.investido  += best.metrics.preco;

    linha.lucro      += best.metrics.lucroUnidade;

    linhasMap.set(key, linha);

    restante -= best.metrics.preco;

  }

  const linhas = Array.from(linhasMap.values());

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

  const totalGasto = linhas.reduce((s,l)=>s+l.investido,0);

  return { linhas, totalLucro, totalGasto, restante };

}

/\* fetch das ações da BD \*/

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),

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

      periodicidade: d.periodicidade || "Anual",

      taxa\_1s: Number(d.taxaCrescimento\_1semana || 0),

      taxa\_1m: Number(d.taxaCrescimento\_1mes || 0),

      taxa\_1a: Number(d.taxaCrescimento\_1ano || 0),

      raw: d

    });

  });

  return out;

}

/\* principal da distribuição \*/

async function distribuirInvestimento(opts){

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

  const base = await fetchAcoesBase();

  // calcular métricas

  const comMetricas = base

    .map(a=>{

      const metrics = calcularMetricasAcao(a, periodoSel, horizonte);

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

    })

    .filter(Boolean)

    .filter(a=>a.metrics.retornoPorEuro > 0);

  if (comMetricas.length === 0) {

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

  }

  // (opcional) limitar ao TOP\_N melhores por retorno/€:

  const TOP\_N = 10;

  const universo = [...comMetricas]

    .sort((a,b)=>b.metrics.retornoPorEuro - a.metrics.retornoPorEuro)

    .slice(0, TOP\_N);

  // distribuir

  if (acoesCompletas){

    return distribuirInteiras(universo, investimento);

  } else {

    return distribuirFracoes(universo, investimento);

  }

}

/\* render do resultado TOP 10 \*/

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>

    </div>

  `;

}

/\* ler opções do UI (TOP 10) \*/

function getTop10Options() {

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

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

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

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

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

  return { investimento, periodoSel, horizonte, usarTotal, acoesCompletas };

}

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

   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;

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

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

      return;

    }

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

    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 = "";

  });

}