**FinanceAPP 2.0 | ficheiros conteúdo projecto**

Indices

[JS| Auth 2](#_Toc208499754)

[HTML | Auth 4](#_Toc208499755)

[JS | dashboard.js 5](#_Toc208499756)

[JS | simulador.js 17](#_Toc208499757)

[JS | atividade.js 26](#_Toc208499758)

[HTML | dasboard.html 45](#_Toc208499759)

[HTML | atividade.html 51](#_Toc208499760)

[HTML | Simulador.html 54](#_Toc208499761)

[HTML | analise.html 57](#_Toc208499762)

[JS | analise.js 59](#_Toc208499763)

**Push para o gitHub**

git add .

git commit -m "Mensagem"

git push

# JS| Auth

[ir para o topo](#_top)

// js/screens/auth.js

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

import {

  getAuth,

  setPersistence,

  browserLocalPersistence,

  onAuthStateChanged,

  createUserWithEmailAndPassword,

  signInWithEmailAndPassword,

  sendPasswordResetEmail,

  GoogleAuthProvider,

  signInWithPopup,

  signOut,

} from "https://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js";

let auth;

const $ = (sel) => document.querySelector(sel);

function showMsg(msg, type = "info") {

  const box = $("#authMsg");

  if (!box) return;

  box.textContent = msg;

  box.style.color = type === "error" ? "#b00020" : "var(--muted-foreground)";

}

function mapAuthError(err) {

  const code = err?.code || "";

  const msgs = {

    "auth/invalid-email": "Email inválido.",

    "auth/email-already-in-use": "Este email já está registado.",

    "auth/weak-password": "Palavra-passe fraca (mín. 6 caracteres).",

    "auth/invalid-credential": "Credenciais inválidas.",

    "auth/wrong-password": "Palavra-passe incorreta.",

    "auth/user-not-found": "Utilizador não encontrado.",

    "auth/popup-closed-by-user": "Janela fechada antes de concluir.",

  };

  return msgs[code] || "Ocorreu um erro. Tenta novamente.";

}

async function doSignIn(email, password) {

  await signInWithEmailAndPassword(auth, email, password);

}

async function doRegister(email, password) {

  await createUserWithEmailAndPassword(auth, email, password);

}

async function doGoogle() {

  const provider = new GoogleAuthProvider();

  await signInWithPopup(auth, provider);

}

async function doReset(email) {

  await sendPasswordResetEmail(auth, email);

}

export function protectPage() {

  const auth = getAuth(app);

  return new Promise((resolve) => {

    const unsub = onAuthStateChanged(auth, (user) => {

      unsub();

      if (!user) {

        document.body.classList.add("auth-screen");

        window.navigateTo?.("auth");

        resolve(false);

      } else {

        document.body.classList.remove("auth-screen");

        resolve(true);

      }

    });

  });

}

export function initScreen() {

  // esconder footer e afins no ecrã de auth (aproveita o CSS existente)

  document.body.classList.add("auth-screen");

  auth = getAuth(app);

  setPersistence(auth, browserLocalPersistence).catch(() => {});

  // Se já estiver autenticado, segue para a dashboard

  onAuthStateChanged(auth, (user) => {

    if (user) {

      document.body.classList.remove("auth-screen");

      window.navigateTo("dashboard");

    }

  });

  const emailEl = $("#authEmail");

  const passEl = $("#authPassword");

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

    showMsg("");

    try {

      await doSignIn(emailEl.value.trim(), passEl.value);

      showMsg("Sessão iniciada.");

      document.body.classList.remove("auth-screen");

      window.navigateTo("dashboard");

    } catch (e) {

      showMsg(mapAuthError(e), "error");

      console.error(e);

    }

  });

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

    showMsg("");

    try {

      await doRegister(emailEl.value.trim(), passEl.value);

      showMsg("Conta criada. A iniciar sessão…");

      document.body.classList.remove("auth-screen");

      window.navigateTo("dashboard");

    } catch (e) {

      showMsg(mapAuthError(e), "error");

      console.error(e);

    }

  });

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

    showMsg("");

    try {

      await doGoogle();

      document.body.classList.remove("auth-screen");

      window.navigateTo("dashboard");

    } catch (e) {

      showMsg(mapAuthError(e), "error");

      console.error(e);

    }

  });

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

    const email = emailEl.value.trim();

    if (!email) return showMsg("Introduz o teu email para recuperar.", "error");

    try {

      await doReset(email);

      showMsg("Email de recuperação enviado (verifica o spam).");

    } catch (e) {

      showMsg(mapAuthError(e), "error");

      console.error(e);

    }

  });

  // Opcional: expor logout global (podes chamar em Settings)

  window.appSignOut = async () => {

    try {

      await signOut(auth);

    } finally {

      document.body.classList.add("auth-screen");

      window.navigateTo("auth");

    }

  };

}

export async function doLogout() {

  const auth = getAuth(app); // garante que obtemos a instância mesmo fora do init

  try {

    await signOut(auth);

  } finally {

    document.body.classList.add("auth-screen");

    window.navigateTo("auth");

  }

}

# HTML | Auth

[ir para o topo](#_top)

<!-- screens/auth.html -->

<section class="page" data-screen-title="Autenticação">

  <div style="max-width: 420px; margin: 24px auto;">

    <div class="card">

      <div class="card-content" style="align-items:flex-start; gap:12px;">

        <div>

          <div class="label"></div>

          <div class="value" style="font-size:1.4rem;">Entrar ou criar conta</div>

          <div class="subvalue">Acede ao teu portefólio e simuladores especializados.</div>

        </div>

      </div>

      <div class="form-grid" style="margin-top:12px;">

        <div style="grid-column:1 / -1;">

          <label class="label" for="authEmail">Email</label>

          <input id="authEmail" type="email" placeholder="ex: nome@exemplo.com" required />

        </div>

        <div style="grid-column:1 / -1;">

          <label class="label" for="authPassword">Palavra-passe</label>

          <input id="authPassword" type="password" placeholder="••••••••" required />

        </div>

      </div>

      <div class="actions-row">

        <button id="btnSignIn" class="btn premium full">Entrar</button>

        <button id="btnRegister" class="btn outline full">Criar conta</button>

        <button id="btnGoogle" class="btn ghost full">

          <i class="fa-brands fa-google"></i> Entrar com Google

        </button>

        <button id="btnReset" class="btn ghost full">Recuperar palavra-passe</button>

      </div>

      <div id="authMsg" class="subvalue" style="margin-top:6px;"></div>

    </div>

  </div>

</section>

# JS | dashboard.js

[ir para o topo](#_top)

// js/screens/dashboard.js

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

import {

  getDocs, collection, query, orderBy, limit, addDoc, serverTimestamp

} from "https://www.gstatic.com/firebasejs/10.12.0/firebase-firestore.js";

export async function initScreen() {

  console.log("✅ dashboard.js iniciado");

  // --- ELEMENTOS DA UI ---

  const valorTotalEl    = document.getElementById("valorTotal");

  const retornoEl       = document.getElementById("retornoTotal");

  const lucroTotalEl    = document.getElementById("lucroTotal");

  const posicoesEl      = document.getElementById("posicoesAtivas");

  const objetivosEl     = document.getElementById("objetivosFinanceiros");

  const taxaSucessoEl   = document.getElementById("taxaSucesso");

  const objetivoTotalEl = document.getElementById("objetivoTotal");

  const valorCarteiraEl = document.getElementById("valorCarteira");

  // --- ACUMULADORES ---

  let totalInvestido = 0;

  let totalLucro = 0;

  let objetivoFinanceiroTotal = 0;

  let objetivosAtingidos = 0;

  try {

    // 1) Buscar ativos e preços atuais

    const [ativosSnapshot, acoesSnapshot] = await Promise.all([

      getDocs(collection(db, "ativos")),

      getDocs(collection(db, "acoesDividendos")),

    ]);

    // Mapa de preços atuais por TICKER

    const valorAtualMap = new Map();

    acoesSnapshot.forEach((doc) => {

      const d = doc.data();

      if (d.ticker && typeof d.valorStock === "number") {

        valorAtualMap.set(String(d.ticker).toUpperCase(), d.valorStock);

      }

    });

    // Agrupar ativos por TICKER

    const agrupadoPorTicker = new Map();

    ativosSnapshot.forEach((doc) => {

      const a = doc.data();

      const ticker = (a.ticker || "").toUpperCase();

      if (!ticker) return;

      const grupo = agrupadoPorTicker.get(ticker) || {

        quantidade: 0,

        investimento: 0,

        objetivoFinanceiro: 0,

        objetivoDefinido: false,

      };

      const quantidade   = parseFloat(a.quantidade || 0);

      const precoCompra  = parseFloat(a.precoCompra || 0);

      const objetivo     = parseFloat(a.objetivoFinanceiro || 0);

      grupo.quantidade   += quantidade;

      grupo.investimento += precoCompra \* quantidade;

      // objetivo conta uma única vez por ticker

      if (!grupo.objetivoDefinido && objetivo > 0) {

        grupo.objetivoFinanceiro = objetivo;

        grupo.objetivoDefinido   = true;

      }

      agrupadoPorTicker.set(ticker, grupo);

    });

    let totalObjetivos = 0;

    // Calcular KPIs globais

    agrupadoPorTicker.forEach((g, ticker) => {

      const precoAtual = valorAtualMap.get(ticker) || 0;

      const atual      = g.quantidade \* precoAtual;

      const lucro      = atual - g.investimento;

      totalInvestido += g.investimento;

      totalLucro     += lucro;

      if (g.objetivoDefinido) {

        totalObjetivos++;

        objetivoFinanceiroTotal += g.objetivoFinanceiro;

        if (lucro >= g.objetivoFinanceiro) objetivosAtingidos++;

      }

    });

    const retorno     = totalInvestido > 0 ? (totalLucro / totalInvestido) \* 100 : 0;

    const taxaSucesso = objetivoFinanceiroTotal > 0 ? (totalLucro / objetivoFinanceiroTotal) \* 100 : 0;

    const valorCarteira = totalInvestido + totalLucro; // 💰 investido + lucro/prejuízo

    // Atualizar UI

    if (valorTotalEl)     valorTotalEl.textContent     = `€${totalInvestido.toFixed(2)}`;

    if (lucroTotalEl)     lucroTotalEl.textContent     = `€${totalLucro.toFixed(2)}`;

    if (retornoEl)        retornoEl.textContent        = `${retorno.toFixed(1)}%`;

    if (posicoesEl)       posicoesEl.textContent       = agrupadoPorTicker.size;

    if (objetivosEl)      objetivosEl.textContent      = `${objetivosAtingidos}/${totalObjetivos}`;

    if (objetivoTotalEl)  objetivoTotalEl.textContent  = `€${objetivoFinanceiroTotal.toFixed(2)}`;

    if (taxaSucessoEl)    taxaSucessoEl.textContent    = `${taxaSucesso.toFixed(1)}%`;

    if (valorCarteiraEl)  valorCarteiraEl.textContent  = `€${valorCarteira.toFixed(2)} valor em carteira`;

  } catch (err) {

    console.error("❌ Erro nos KPIs:", err);

  }

  // 2) Atividades recentes (apenas campos solicitados, sem barras) — com expand/colapse

  await carregarAtividadeRecenteSimplificada();

  // 3) Botões

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

    import("../main.js").then(({ navigateTo }) => navigateTo("simulador"));

  });

  document.getElementById("btnOportunidades")?.addEventListener("click", openOportunidades);

  document.getElementById("opClose")?.addEventListener("click", closeOportunidades);

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

    if (e.target.id === "opModal") closeOportunidades();

  });

  // Chips do período no popup

  document.querySelectorAll("#opModal .chip").forEach(btn => {

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

      const p = btn.getAttribute("data-periodo");

      opPeriodoAtual = p || "1s";

      setActiveChip(opPeriodoAtual);

      carregarTop10Crescimento(opPeriodoAtual);

    });

  });

    // --- Modal: Adicionar Ação ---

  const btnAdd = document.getElementById("btnAddAcao");

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

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

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

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

  const tipoAcaoSel = document.getElementById("tipoAcao");

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

  // abre

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

    addModal?.classList.remove("hidden");

  });

  // fecha

  function closeAddModal() {

    addModal?.classList.add("hidden");

    addForm?.reset();

    // repõe o label (caso tenha mudado para Venda)

    if (labelPreco) labelPreco.firstChild.textContent = "Preço da transação (€)";

  }

  addClose?.addEventListener("click", closeAddModal);

  addCancel?.addEventListener("click", closeAddModal);

  addModal?.addEventListener("click", (e) => {

    if (e.target.id === "addModal") closeAddModal();

  });

  // muda o label conforme compra/venda (puramente visual)

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

    if (!labelPreco) return;

    if (tipoAcaoSel.value === "venda") {

      labelPreco.firstChild.textContent = "Preço de venda (€)";

    } else {

      labelPreco.firstChild.textContent = "Preço de compra (€)";

    }

  });

  // submit

  addForm?.addEventListener("submit", async (e) => {

    e.preventDefault();

    const tipoAcao = (document.getElementById("tipoAcao")?.value || "compra").toLowerCase();

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

    const ticker   = (document.getElementById("tickerAtivo")?.value || "").trim().toUpperCase();

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

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

    const qtdRaw   = Number(document.getElementById("quantidadeAtivo")?.value || 0);

    const preco    = Number(document.getElementById("precoAtivo")?.value || 0);

    const objetivo = Number(document.getElementById("objetivoAtivo")?.value || 0);

    if (!nome || !ticker || !qtdRaw || !preco) {

      alert("Preenche pelo menos: Tipo, Nome, Ticker, Quantidade e Preço.");

      return;

    }

    // ── regra simples: vendas entram como quantidade negativa e usam o mesmo campo 'precoCompra'

    const quantidade = tipoAcao === "venda" ? -Math.abs(qtdRaw) : Math.abs(qtdRaw);

    const payload = {

      tipoAcao,              // "compra" ou "venda" (útil para auditoria)

      nome,

      ticker,

      setor,

      mercado,

      quantidade,            // negativo na venda

      precoCompra: preco,    // mantém a compatibilidade com os teus cálculos atuais

      objetivoFinanceiro: isNaN(objetivo) ? 0 : objetivo,

      dataCompra: serverTimestamp(), // data/hora automática

    };

    try {

      await addDoc(collection(db, "ativos"), payload);

      closeAddModal();

      // Atualiza o painel rapidamente. Para máxima simplicidade, recarrega:

      // (se preferires sem refresh, dá para chamar as funções de KPI e atividade novamente)

      window.location.reload();

    } catch (err) {

      console.error("❌ Erro ao guardar ativo:", err);

      alert("Não foi possível guardar. Tenta novamente.");

    }

  });

}

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

   ATIVIDADE RECENTE (SIMPLIFICADA) + EXPAND/COLAPSE

   ========================= \*/

let atividadesCache = [];     // guarda todos os docs formatados

let atividadesExpandido = false; // estado de expansão

async function carregarAtividadeRecenteSimplificada() {

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

  if (!cont) return;

  try {

    // Trazemos MAIS do que 4 para já termos tudo em cache

    const q = query(collection(db, "ativos"), orderBy("dataCompra", "desc"), limit(50));

    const snapAtivos = await getDocs(q);

    if (snapAtivos.empty) {

      cont.innerHTML = `<p class="muted">Sem atividades ainda.</p>`;

      return;

    }

    const fmtEUR   = new Intl.NumberFormat("pt-PT",{ style:"currency", currency:"EUR" });

    const fmtDateL = new Intl.DateTimeFormat("pt-PT", {

      year: "numeric", month: "long", day: "numeric",

      hour: "2-digit", minute: "2-digit", second: "2-digit",

      timeZoneName: "short"

    });

    atividadesCache = []; // reset cache

    snapAtivos.forEach(doc => {

        const d = doc.data();

        const nome        = d.nome || d.ticker || "Ativo";

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

        const setor       = d.setor || "-";

        const mercado     = d.mercado || "-";

        const precoCompra = Number(d.precoCompra || 0);

        const quantidade  = Number(d.quantidade || 0);

        // NOVO → badge COMPRA/VENDA

        const tipo  = (d.tipoAcao || "compra").toLowerCase();

        const badge = tipo === "venda"

          ? '<span class="tag venda">VENDA</span>'

          : '<span class="tag compra">COMPRA</span>';

        let dataTxt = "sem data";

        if (d.dataCompra && typeof d.dataCompra.toDate === "function") {

          dataTxt = fmtDateL.format(d.dataCompra.toDate());

        }

      atividadesCache.push(`

        <div class="activity-item">

          <div class="activity-left">

            <span class="activity-icon">🛒</span>

            <div>

              <p>${badge} <strong>${nome}</strong> <span class="muted">(${ticker})</span></p>

              <p class="muted">${setor} • ${mercado}</p>

              <p class="muted">${quantidade} ${quantidade === 1 ? "ação" : "ações"} @ ${fmtEUR.format(precoCompra)}</p>

              <p class="muted">Data: ${dataTxt}</p>

            </div>

          </div>

        </div>

      `);

    });

    // Render inicial: só 2

    renderAtividades(cont, 2);

    atividadesExpandido = false;

    // Ligar o botão "Ver Toda Atividade" (sem mexer no HTML: apanha o primeiro .btn.outline.full nessa card)

    const btnVerTodos = document.querySelector(".dashboard .card.glass .btn.outline.full");

    if (btnVerTodos) {

      btnVerTodos.textContent = "Ver Toda Atividade";

      btnVerTodos.onclick = () => {

        atividadesExpandido = !atividadesExpandido;

        if (atividadesExpandido) {

          renderAtividades(cont, atividadesCache.length);

          btnVerTodos.textContent = "Mostrar menos";

        } else {

          renderAtividades(cont, 2);

          btnVerTodos.textContent = "Ver Toda Atividade";

        }

      };

    }

  } catch (e) {

    console.error("Erro ao carregar atividade:", e);

    cont.innerHTML = `<p class="muted">Não foi possível carregar a lista.</p>`;

  }

}

function renderAtividades(container, howMany) {

  const slice = atividadesCache.slice(0, howMany);

  container.innerHTML = slice.join("");

}

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

   POPUP: TOP 10 OPORTUNIDADES

   ========================= \*/

let opInterval = null;

let opPeriodoAtual = "1s";

async function carregarTop10Crescimento(periodo = "1s") {

  const lista = document.getElementById("listaTop10");

  if (!lista) return;

  lista.innerHTML = "🔄 A carregar...";

  const campos = {

    "1s":  "taxaCrescimento\_1semana",

    "1m":  "taxaCrescimento\_1mes",

    "1ano":"taxaCrescimento\_1ano",

  };

  const campo = campos[periodo] || campos["1s"];

  try {

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

    const arr = [];

    snap.forEach(doc => {

      const d = doc.data();

      const crescimento = Number(d[campo] ?? 0);

      if (Number.isFinite(crescimento) && d.ticker) {

        arr.push({

          nome: d.nome || d.ticker,

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

          crescimento

        });

      }

    });

    const top10 = arr.sort((a,b) => b.crescimento - a.crescimento).slice(0, 10);

    if (top10.length === 0) {

      lista.innerHTML = "<li>😕 Nenhuma ação com crescimento positivo.</li>";

      return;

    }

    lista.innerHTML = top10.map(item => `

      <li>

        <div class="left">

          <strong>${item.nome}</strong>

          <span class="ticker">(${item.ticker})</span>

        </div>

        <span class="${item.crescimento >= 0 ? "gain" : "loss"}">

          ${item.crescimento >= 0 ? "+" : ""}${item.crescimento.toFixed(2)}%

        </span>

      </li>

    `).join("");

  } catch (err) {

    console.error("Erro ao carregar Top 10:", err);

    lista.innerHTML = "<li style='color:#f88;'>Erro ao carregar dados.</li>";

  }

}

function openOportunidades() {

  const modal = document.getElementById("opModal");

  if (!modal) return;

  modal.classList.remove("hidden");

  setActiveChip(opPeriodoAtual);

  carregarTop10Crescimento(opPeriodoAtual);

  clearInterval(opInterval);

  opInterval = setInterval(() => {

    carregarTop10Crescimento(opPeriodoAtual);

  }, 30000);

}

function closeOportunidades() {

  const modal = document.getElementById("opModal");

  if (!modal) return;

  modal.classList.add("hidden");

  clearInterval(opInterval);

  opInterval = null;

}

function setActiveChip(periodo) {

  document.querySelectorAll("#opModal .chip").forEach(ch => {

    const p = ch.getAttribute("data-periodo");

    ch.classList.toggle("active", p === periodo);

  });

}

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

// Simulador Botão Definir Objetivo (versão wizard + dropdowns)

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

// ---------- Helpers comuns ----------

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 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 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\_1mes", "taxaCrescimento\_1semana", "taxaCrescimento\_1ano"];

  for (const k of ordem){

    const v = Number(acao[k] || 0);

    if (v !== 0) return v;

  }

  return 0;

}

function calcularMetricasAcao(acao, periodoSel, horizonte){

  const prefer = campoCrescimento(periodoSel);

  const taxaPct = melhorTaxaDisponivel(acao, prefer);

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

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

  const per       = acao.periodicidade || "Anual";

  if (!(preco>0)) return null;

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

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

  const mult = Math.pow(1+r, h);

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

  const totalDividendos = dividirPeriodicidade(dividendo, per) \* h;

  const lucroUnidade = totalDividendos + valorizacao;

  const retornoPorEuro = lucroUnidade / preco;

  return { preco, dividendoAnual: dividirPeriodicidade(dividendo, per), taxaPct, lucroUnidade, retornoPorEuro };

}

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) };

}

function distribuirInteiras(acoes, investimento){

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

  const linhasMap = new Map();

  let restante = investimento;

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

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

    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 };

}

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">Nenhuma ação selecionada com retorno positivo.</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.inteiras ? 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.periodo}</th><th>Dividendo</th>

            </tr>

          </thead>

          <tbody>${rows}</tbody>

        </table>

      </div>

      <p style="margin-top:.6rem">

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

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

        <br/>

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

      </p>

    </div>

  `;

}

// --- opções dropdowns

const OPT\_SETORES = [

  "", "ETF iTech","ETF Finance","ETF Energia","ETF Materiais",

  "Alimentação","Automóvel","Bens de Consumo","Consumo Cíclico","Consumo Defensivo",

  "Defesa","Energia","Finanças","Imobiliário","Indústria",

  "Infraestruturas / Energia","Materiais","Restauração","Saúde","Tecnologia","Telecomunicações"

];

const OPT\_MERCADOS = ["", "Portugal","Europeu","Americano","Americano SP500"];

const OPT\_MESES = ["", "n/A","Janeiro","Fevereiro","Março","Abril","Maio","Junho","Julho","Agosto","Setembro","Outubro","Novembro","Dezembro"];

const OPT\_PERIODICIDADE = ["", "n/A","Anual","Semestral","Trimestral","Mensal"];

function fillSelect(id, opts, placeholder){

  const el = document.getElementById(id); if(!el) return;

  el.innerHTML = opts.map(v=>{

    const label = v || (placeholder||"Todos");

    return `<option value="${v}">${label}</option>`;

  }).join("");

}

// ---------- Estado do modal ----------

let GOAL\_CACHE = [];

const GOAL\_SELECTED = new Map();

// fetch acoesDividendos (inclui setor/mercado/mes/periodicidade)

async function fetchAcoesGoal(){

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

  const out = [];

  snap.forEach(doc=>{

    const d = doc.data();

    if (!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),

      mes: d.mes || "",

      periodicidade: d.periodicidade || "Anual",

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

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

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

      raw: { setor: d.setor || "", mercado: d.mercado || "" }

    });

  });

  return out;

}

// ----- Wizard helpers -----

function showGoalStep(n){

  document.querySelectorAll('#goalModal .goal-step').forEach(sec=>{

    sec.hidden = sec.dataset.step !== String(n);

  });

}

function ensureListVisible(){

  const evt = new Event('click');

  document.getElementById('goalBtnFiltrar')?.dispatchEvent(evt);

}

// ----- Abrir/fechar -----

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

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

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

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

  // always open at step 1

  showGoalStep(1);

  // preencher dropdowns

  fillSelect("goalFiltroSetor", OPT\_SETORES, "Setor");

  fillSelect("goalFiltroMercado", OPT\_MERCADOS, "Mercado");

  fillSelect("goalFiltroMes", OPT\_MESES, "Mês do dividendo");

  fillSelect("goalFiltroPeriodicidade", OPT\_PERIODICIDADE, "Periodicidade");

  goalModal?.classList.remove("hidden");

  if (GOAL\_CACHE.length === 0) GOAL\_CACHE = await fetchAcoesGoal();

  renderGoalList(GOAL\_CACHE);

  renderGoalSelected();

  syncGoalCheckboxes();

});

goalClose?.addEventListener("click", closeGoalModal);

goalModal?.addEventListener("click", (e) => {

  if (e.target.id === "goalModal") closeGoalModal();

});

function closeGoalModal() {

  goalModal?.classList.add("hidden");

  goalModal?.classList.remove("goal-show-results");

  GOAL\_SELECTED.clear();

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

  if (box) box.innerHTML = "";

  const bar = document.getElementById("goalResultsBar");

  if (bar) bar.remove();

  showGoalStep(1);

}

// ----- Passos -----

document.getElementById('goalNext1')?.addEventListener('click', ()=>{

  showGoalStep(2);

  ensureListVisible();

});

document.getElementById('goalNext2')?.addEventListener('click', ()=>{

  if (GOAL\_SELECTED.size === 0) { alert('Seleciona pelo menos uma ação.'); return; }

  showGoalStep(3);

});

document.getElementById('goalBack2')?.addEventListener('click', ()=> showGoalStep(1));

document.getElementById('goalBack3')?.addEventListener('click', ()=> showGoalStep(2));

document.getElementById('goalBackToEdit')?.addEventListener('click', ()=> showGoalStep(3));

// ----- Filtrar/Limpar -----

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

  const t   = (document.getElementById("goalFiltroTicker")?.value || "").trim().toLowerCase();

  const n   = (document.getElementById("goalFiltroNome")?.value   || "").trim().toLowerCase();

  const s   = (document.getElementById("goalFiltroSetor")?.value  || "").trim().toLowerCase();

  const m   = (document.getElementById("goalFiltroMercado")?.value|| "").trim().toLowerCase();

  const mes = (document.getElementById("goalFiltroMes")?.value    || "").trim().toLowerCase();

  const per = (document.getElementById("goalFiltroPeriodicidade")?.value||"").trim().toLowerCase();

  const res = GOAL\_CACHE.filter(a => {

    const hitT   = !t   || a.ticker.toLowerCase().includes(t);

    const hitN   = !n   || a.nome.toLowerCase().includes(n);

    const hitS   = !s   || String(a.raw?.setor||"").toLowerCase()      === s;

    const hitM   = !m   || String(a.raw?.mercado||"").toLowerCase()    === m;

    const hitMes = !mes || String(a.mes||"").toLowerCase()             === mes;

    const hitPer = !per || String(a.periodicidade||"").toLowerCase()   === per;

    return hitT && hitN && hitS && hitM && hitMes && hitPer;

  });

  renderGoalList(res);

});

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

  ["goalFiltroTicker","goalFiltroNome","goalFiltroSetor","goalFiltroMercado","goalFiltroMes","goalFiltroPeriodicidade"]

    .forEach(id=>{ const el = document.getElementById(id); if (el) el.value = ""; });

  renderGoalList(GOAL\_CACHE);

});

// ----- Lista (toggle por botão “＋/✓”) -----

function renderGoalList(arr){

  const ul = document.getElementById("goalListaAcoes");

  if (!ul) return;

  if (!arr || arr.length === 0) {

    ul.innerHTML = `<li><span class="meta">Sem resultados.</span></li>`;

    return;

  }

  ul.innerHTML = arr.map(a => {

    const selected = GOAL\_SELECTED.has(a.ticker);

    const setor   = a.raw?.setor || "-";

    const mercado = a.raw?.mercado || "-";

    const preco   = Number(a.valorStock||0).toFixed(2);

    return `

      <li>

        <div style="display:flex;align-items:center;gap:10px;flex:1;">

          <div>

            <div><strong>${a.nome}</strong> <span class="meta">(${a.ticker})</span></div>

            <div class="meta">${setor} • ${mercado} · Preço: €${preco}</div>

          </div>

        </div>

        <button class="icon-btn goal-toggle" data-ticker="${a.ticker}" title="${selected?'Remover':'Adicionar'}">

          ${selected ? '✓' : '＋'}

        </button>

      </li>

    `;

  }).join("");

  ul.querySelectorAll('.goal-toggle').forEach(btn=>{

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

      const t = btn.dataset.ticker;

      const obj = arr.find(x=>x.ticker===t) || GOAL\_CACHE.find(x=>x.ticker===t);

      if (!obj) return;

      if (GOAL\_SELECTED.has(t)) {

        GOAL\_SELECTED.delete(t);

        btn.textContent = '＋';

        btn.title = 'Adicionar';

      } else {

        GOAL\_SELECTED.set(t, obj);

        btn.textContent = '✓';

        btn.title = 'Remover';

      }

      renderGoalSelected();

    });

  });

}

// ----- Selecionadas (tags) -----

function renderGoalSelected(){

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

  if (!wrap) return;

  const list = Array.from(GOAL\_SELECTED.values());

  if (list.length === 0) {

    wrap.innerHTML = `<span class="muted">Nenhuma ação selecionada.</span>`;

    return;

  }

  wrap.innerHTML = list.map(a => `

    <span class="goal-tag">

      ${a.ticker}

      <button title="Remover" data-del="${a.ticker}">x</button>

    </span>

  `).join("");

  wrap.querySelectorAll('button[data-del]').forEach(btn=>{

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

      const t = btn.dataset.del;

      GOAL\_SELECTED.delete(t);

      renderGoalSelected();

      const b = document.querySelector(`#goalListaAcoes .goal-toggle[data-ticker="${t}"]`);

      if (b){ b.textContent = '＋'; b.title = 'Adicionar'; }

    });

  });

}

// ----- Exclusividade das duas checkboxes -----

function syncGoalCheckboxes(){

  const chkInt = document.getElementById("goalAcoesCompletas");

  const chkTot = document.getElementById("goalUsarTotal");

  if (!chkInt || !chkTot) return;

  function onChange(e){

    if (e.target === chkInt && chkInt.checked) chkTot.checked = false;

    if (e.target === chkTot && chkTot.checked) chkInt.checked = false;

  }

  chkInt.onchange = onChange;

  chkTot.onchange = onChange;

  onChange({target: chkInt.checked ? chkInt : chkTot});

}

// ----- Simular -----

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

  const invest   = Number(document.getElementById("goalInvest")?.value || 0);

  const periodo  = (document.getElementById("goalPeriodo")?.value || "1ano");

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

  const inteiras = !!document.getElementById("goalAcoesCompletas")?.checked;

  const usarTot  = !!document.getElementById("goalUsarTotal")?.checked;

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

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

  const baseSelecionada = Array.from(GOAL\_SELECTED.values());

  if (baseSelecionada.length === 0){ box.innerHTML = `<div class="card"><p class="muted">Seleciona pelo menos uma ação.</p></div>`; return; }

  if (invest <= 0){ box.innerHTML = `<div class="card"><p class="muted">Indica o montante a investir.</p></div>`; return; }

  try{

    const comMetricas = baseSelecionada

      .map(a=>{ const m = calcularMetricasAcao(a, periodo, horizonte); return m ? {...a, metrics:m} : null; })

      .filter(Boolean)

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

    if (comMetricas.length === 0){

      box.innerHTML = `<div class="card"><p class="muted">As ações selecionadas não têm retorno positivo para o período escolhido.</p></div>`;

      return;

    }

    const resultado = inteiras

      ? distribuirInteiras(comMetricas, invest)

      : distribuirFracoes(comMetricas, invest);

    renderResultado(box, resultado, { periodo, horizonte, inteiras, usarTot });

    goalEnterResultsMode();        // mostra modo resultado

  }catch(err){

    console.error(err);

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

  }

});

// === UI: modo resultado ===

function goalEnterResultsMode() {

  const modal = document.getElementById("goalModal");

  if (!modal) return;

  showGoalStep(4);                           // step 4 = resultados

  modal.classList.add("goal-show-results");

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

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

    bar.id = "goalResultsBar";

    bar.className = "goal-results-bar";

    bar.innerHTML = `

      <button id="goalBackToEdit" class="icon-btn close" title="Voltar">×</button>

      <span class="title">Resultado da simulação</span>

    `;

    const res = document.getElementById("goalResultado");

    if (res && res.parentNode) res.parentNode.insertBefore(bar, res);

    document.getElementById("goalBackToEdit")?.addEventListener("click", goalExitResultsMode);

  }

}

function goalExitResultsMode() {

  const modal = document.getElementById("goalModal");

  if (!modal) return;

  modal.classList.remove("goal-show-results");

  showGoalStep(3);                            // regressa ao passo 3

}

# JS | simulador.js

[ir para o topo](#_top)

// 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 = "";

  });

}

# JS | atividade.js

[ir para o topo](#_top)

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

import {

  collection, getDocs, query, orderBy, where,

  addDoc, serverTimestamp, deleteDoc, doc

} from "https://www.gstatic.com/firebasejs/10.12.0/firebase-firestore.js";

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

   Carregar Chart.js on-demand

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

async function ensureChartJS(){

  if (window.Chart) return;

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

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

    s.src = "https://cdn.jsdelivr.net/npm/chart.js@4.4.3/dist/chart.umd.min.js";

    s.onload = resolve;

    s.onerror = reject;

    document.head.appendChild(s);

  });

}

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

   Paleta / Tema

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

function isDark(){

  return document.documentElement.getAttribute("data-theme") === "dark";

}

function chartColors(){

  const dark = isDark();

  return {

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

    ticks: dark ? "rgba(255,255,255,.85)" : "rgba(0,0,0,.7)",

    title: dark ? "#fff" : "#111",

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

    tooltipFg: dark ? "#fff" : "#111",

  };

}

const PALETTE = [

  "#4F46E5","#22C55E","#EAB308","#EF4444","#06B6D4","#F59E0B",

  "#A855F7","#10B981","#3B82F6","#F472B6","#84CC16","#14B8A6"

];

/\* manter instâncias \*/

const charts = {

  setores:null, mercados:null, top5:null, top5Yield:null, timeline:null, divCal:null

};

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

   Helpers (mantidos + novos)

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

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

function isFiniteNum(v){ return Number.isFinite(Number(v)); }

function formatNum(n){ return Number(n || 0).toLocaleString("pt-PT"); }

async function fetchDividendInfoByTickers(tickers){

  const out = new Map();

  const chunks = [];

  for (let i=0; i<tickers.length; i+=10) chunks.push(tickers.slice(i, i+10));

  for (const chunk of chunks){

    const q2 = query(collection(db, "acoesDividendos"), where("ticker", "in", chunk));

    const snap = await getDocs(q2);

    snap.forEach(docu => {

      const d = docu.data();

      if (d.ticker) out.set(String(d.ticker).toUpperCase(), d);

    });

  }

  return out;

}

function pickBestRate(info){

  if (typeof info?.taxaCrescimento\_1mes === "number") return { taxa: info.taxaCrescimento\_1mes, periodLabel: "mês" };

  if (typeof info?.taxaCrescimento\_1semana === "number") return { taxa: info.taxaCrescimento\_1semana, periodLabel: "semana" };

  if (typeof info?.taxaCrescimento\_1ano === "number") return { taxa: info.taxaCrescimento\_1ano, periodLabel: "ano" };

  return { taxa: null, periodLabel: null };

}

function estimateTime(currentPrice, targetPrice, growthPct, periodLabel) {

  const r = Number(growthPct || 0) / 100;

  if (r <= 0 || !isFiniteNum(currentPrice) || !isFiniteNum(targetPrice) || currentPrice <= 0 || targetPrice <= 0) return "—";

  const n = Math.log(targetPrice / currentPrice) / Math.log(1 + r);

  if (!isFinite(n) || n < 0) return "—";

  if (periodLabel === "semana") return `${n.toFixed(1)} semanas`;

  if (periodLabel === "mês") return `${n.toFixed(1)} meses`;

  return `${n.toFixed(1)} anos`;

}

/\* ---- novos helpers (dividendos & calendário) ---- \*/

const MES\_IDX = {

  "janeiro":0,"fevereiro":1,"março":2,"marco":2,"abril":3,"maio":4,"junho":5,"julho":6,

  "agosto":7,"setembro":8,"outubro":9,"novembro":10,"dezembro":11

};

function pagamentosAno(periodicidade){

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

  if (p.startsWith("mensal")) return 12;

  if (p.startsWith("trimes")) return 4;

  if (p.startsWith("semes")) return 2;

  if (p.startsWith("anual"))  return 1;

  return 0;

}

function mesesPagos(periodicidade, mesTipico){

  // devolve array de índices [0..11] dos meses esperados

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

  const baseIdx = MES\_IDX[String(mesTipico||"").trim().toLowerCase()];

  if (p.startsWith("mensal")) return Array.from({length:12},(\_,i)=>i);

  if (p.startsWith("trimes")){

    const start = Number.isFinite(baseIdx) ? baseIdx : 0;

    return [start, (start+3)%12, (start+6)%12, (start+9)%12];

  }

  if (p.startsWith("semes")){

    const start = Number.isFinite(baseIdx) ? baseIdx : 0;

    return [start, (start+6)%12];

  }

  if (p.startsWith("anual")){

    return Number.isFinite(baseIdx) ? [baseIdx] : [];

  }

  return [];

}

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

   Renders — antigos + novos

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

function renderSetorDoughnut(map){

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

  if (!el) return;

  charts.setores?.destroy();

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

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

  if (!labels.length){ el.getContext("2d").clearRect(0,0,el.width, el.height); return; }

  charts.setores = 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%",

      plugins:{

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

        tooltip:{ backgroundColor: chartColors().tooltipBg, titleColor: chartColors().tooltipFg, bodyColor: chartColors().tooltipFg,

          callbacks:{ label:(ctx)=>` ${ctx.label}: €${Number(ctx.parsed).toLocaleString("pt-PT",{minimumFractionDigits:2})}` }

        }

      }

    }

  });

}

function renderMercadoDoughnut(map){

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

  if (!el) return;

  charts.mercados?.destroy();

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

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

  if (!labels.length){ el.getContext("2d").clearRect(0,0,el.width, el.height); return; }

  charts.mercados = new Chart(el, {

    type: "doughnut",

    data: { labels, datasets:[{ data, backgroundColor: labels.map((\_,i)=>PALETTE[(i+5)%PALETTE.length]), borderWidth:1 }] },

    options: {

      responsive:true, maintainAspectRatio:false, cutout:"62%",

      plugins:{

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

        tooltip:{ backgroundColor: chartColors().tooltipBg, titleColor: chartColors().tooltipFg, bodyColor: chartColors().tooltipFg,

          callbacks:{ label:(ctx)=>` ${ctx.label}: €${Number(ctx.parsed).toLocaleString("pt-PT",{minimumFractionDigits:2})}` }

        }

      }

    }

  });

}

function renderTop5Bar(gruposArr){

  const el = document.getElementById("chartTop5");

  if (!el) return;

  charts.top5?.destroy();

  const ativos = gruposArr.filter(g=>g.qtd>0)

    .sort((a,b)=> (b.investido||0)-(a.investido||0)).slice(0,5);

  if (!ativos.length){ el.getContext("2d").clearRect(0,0,el.width, el.height); return; }

  const labels = ativos.map(a=>a.ticker);

  const invest = ativos.map(a=>a.investido||0);

  const lucro  = ativos.map(a=>a.lucroAtual||0);

  charts.top5 = new Chart(el, {

    type:"bar",

    data:{ labels, datasets:[

      { label:"Investido (€)", data:invest, backgroundColor:"#3B82F6" },

      { label:"Lucro Atual (€)", data:lucro, backgroundColor:"#22C55E" }

    ]},

    options:{

      responsive:true, maintainAspectRatio: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)=>` ${ctx.dataset.label}: €${Number(ctx.parsed.y||0).toLocaleString("pt-PT",{minimumFractionDigits:2})}` } }

      }

    }

  });

}

function renderTop5YieldBar(rows){

  const el = document.getElementById("chartTop5Yield");

  if (!el) return;

  charts.top5Yield?.destroy();

  const ativos = rows.filter(r=>r.active && isFiniteNum(r.yieldCur))

    .sort((a,b)=> b.yieldCur - a.yieldCur).slice(0,5);

  if (!ativos.length){ el.getContext("2d").clearRect(0,0,el.width, el.height); return; }

  const labels = ativos.map(a=>a.ticker);

  const ys     = ativos.map(a=>Number(a.yieldCur\*100).toFixed(2));

  charts.top5Yield = new Chart(el, {

    type:"bar",

    data:{ labels,

      datasets:[{ label:"Yield (%)", data:ys, backgroundColor:"#EAB308" }] },

    options:{

      indexAxis:"y", responsive:true, maintainAspectRatio: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}: ${Number(ctx.parsed.x||0).toFixed(2)}%` } }

      }

    }

  });

}

function renderTimeline(points){

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

  if (!el) return;

  charts.timeline?.destroy();

  if (!points.length){ el.getContext("2d").clearRect(0,0,el.width, el.height); return; }

  const labels   = points.map(p=>p.label);

  const invested = points.map(p=>p.cumInvest);

  const valueNow = points.map(p=>p.valueNow);

  charts.timeline = new Chart(el,{

    type:"line",

    data:{ labels, datasets:[

      { label:"Investido acumulado (€)", data:invested, tension:.25, borderWidth:2 },

      { label:"Avaliação atual (€)",     data:valueNow, tension:.25, borderWidth:2 }

    ]},

    options:{

      responsive:true, maintainAspectRatio:false, elements:{ point:{ radius:0 } },

      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)=>` ${ctx.dataset.label}: €${Number(ctx.parsed.y||0).toLocaleString("pt-PT",{minimumFractionDigits:2})}` } }

      }

    }

  });

}

function renderDividendoCalendario12m(arrEuros12){

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

  if (!el) return;

  charts.divCal?.destroy();

  const labels = ["Jan","Fev","Mar","Abr","Mai","Jun","Jul","Ago","Set","Out","Nov","Dez"];

  charts.divCal = new Chart(el,{

    type:"bar",

    data:{ labels, datasets:[{ label:"€ / mês (estimado)", data:arrEuros12 }] },

    options:{

      responsive:true, maintainAspectRatio: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)=>` ${ctx.dataset.label}: €${Number(ctx.parsed.y||0).toLocaleString("pt-PT",{minimumFractionDigits:2})}` } }

      }

    }

  });

}

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

   Quick Actions (teu código)

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

function wireQuickActions(gruposArr){

  const byTicker = new Map(gruposArr.map(g => [g.ticker, g]));

  const $ = s => document.querySelector(s);

  const modal   = $("#pfAddModal");

  const title   = $("#pfAddTitle");

  const form    = $("#pfAddForm");

  const close   = $("#pfAddClose");

  const cancel  = $("#pfAddCancel");

  const tipoSel = $("#pfTipoAcao");

  const labelP  = $("#pfLabelPreco");

  const vendaTotWrap = $("#pfVendaTotalWrap");

  const vendaTot = $("#pfVendaTotal");

  const fTicker = $("#pfTicker");

  const fNome   = $("#pfNome");

  const fSetor  = $("#pfSetor");

  const fMerc   = $("#pfMercado");

  const fQtd    = $("#pfQuantidade");

  const fPreco  = $("#pfPreco");

  const fObj    = $("#pfObjetivo");

  function open(kind, ticker){

    const g = byTicker.get(ticker);

    if (!g) return;

    modal.classList.remove("hidden");

    title.textContent = kind === "compra" ? "Comprar ativo" : "Vender ativo";

    tipoSel.value = kind;

    fTicker.value = g.ticker;

    fNome.value   = g.nome;

    fSetor.value  = g.setor;

    fMerc.value   = g.mercado;

    fQtd.value    = "";

    fPreco.value  = "";

    fObj.value    = g.objetivo || "";

    vendaTot.checked = false;

    vendaTotWrap.style.display = kind === "venda" ? "block" : "none";

    labelP.firstChild.textContent = kind === "venda" ? "Preço de venda (€)" : "Preço de compra (€)";

  }

  function closeModal(){

    modal.classList.add("hidden");

    form.reset();

    vendaTot.checked = false;

    vendaTotWrap.style.display = "none";

    labelP.firstChild.textContent = "Preço de compra (€)";

  }

  close?.addEventListener("click", closeModal);

  cancel?.addEventListener("click", closeModal);

  modal?.addEventListener("click", (e)=>{ if (e.target.id === "pfAddModal") closeModal(); });

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

    const buy = e.target.closest("[data-buy]");

    const sell = e.target.closest("[data-sell]");

    if (buy)  open("compra", buy.getAttribute("data-buy"));

    if (sell) open("venda",  sell.getAttribute("data-sell"));

  });

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

    const isVenda = tipoSel.value === "venda";

    labelP.firstChild.textContent = isVenda ? "Preço de venda (€)" : "Preço de compra (€)";

    vendaTotWrap.style.display = isVenda ? "block" : "none";

  });

  form?.addEventListener("submit", async (e) => {

    e.preventDefault();

    const tipo  = (tipoSel.value || "compra").toLowerCase();

    const nome  = fNome.value.trim();

    const ticker= fTicker.value.trim().toUpperCase();

    const setor = fSetor.value.trim();

    const merc  = fMerc.value.trim();

    const qtd   = Number(fQtd.value || 0);

    const preco = Number(fPreco.value || 0);

    const obj   = Number(fObj.value || 0);

    const vendaTotal = vendaTot.checked;

    if (!ticker || !nome || !qtd || !preco) {

      alert("Preenche Ticker, Nome, Quantidade e Preço.");

      return;

    }

    const quantidade = tipo === "venda" ? -Math.abs(qtd) : Math.abs(qtd);

    const payload = {

      tipoAcao: tipo, nome, ticker, setor, mercado: merc,

      quantidade, precoCompra: preco,

      objetivoFinanceiro: isNaN(obj) ? 0 : obj,

      dataCompra: serverTimestamp(),

    };

    try {

      await addDoc(collection(db, "ativos"), payload);

      if (tipo === "venda" && vendaTotal) {

        const toDelQ = query(collection(db, "ativos"), where("ticker", "==", ticker));

        const snapDel = await getDocs(toDelQ);

        const promises = [];

        snapDel.forEach(d => promises.push(deleteDoc(doc(db, "ativos", d.id))));

        await Promise.all(promises);

      }

      closeModal();

      location.reload();

    } catch (err) {

      console.error("❌ Erro ao guardar movimento:", err);

      alert("Não foi possível guardar. Tenta novamente.");

    }

  });

}

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

   INIT (screen)

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

export async function initScreen() {

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

  if (!cont) return;

  cont.innerHTML = "A carregar…";

  await ensureChartJS();

  try {

    // 1) Buscar movimentos e agrupar por ticker

    const qSnap = await getDocs(

      query(collection(db, "ativos"), orderBy("dataCompra", "desc"))

    );

    if (qSnap.empty) {

      cont.innerHTML = `<p class="muted">Sem atividades ainda.</p>`;

      renderSetorDoughnut(new Map());

      renderMercadoDoughnut(new Map());

      renderTop5Bar([]);

      renderTop5YieldBar([]);

      renderTimeline([]);

      renderDividendoCalendario12m(new Array(12).fill(0));

      return;

    }

    const grupos = new Map();

    const movimentosAsc = [];

    qSnap.forEach((docu) => {

      const d = docu.data();

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

      if (!ticker) return;

      const qtd = toNum(d.quantidade);

      const preco = toNum(d.precoCompra);

      const invest = qtd \* preco;

      const g = grupos.get(ticker) || {

        ticker,

        nome: d.nome || ticker,

        setor: d.setor || "-",

        mercado: d.mercado || "-",

        qtd: 0,

        investido: 0,

        objetivo: 0,

        anyObjSet: false,

        lastDate: null,

      };

      g.qtd += qtd;

      g.investido += invest;

      const obj = toNum(d.objetivoFinanceiro);

      if (!g.anyObjSet && obj > 0) {

        g.objetivo = obj;

        g.anyObjSet = true;

      }

      const dt =

        d.dataCompra && typeof d.dataCompra.toDate === "function"

          ? d.dataCompra.toDate()

          : null;

      if (!g.lastDate || (dt && dt > g.lastDate)) g.lastDate = dt;

      g.nome = d.nome || g.nome;

      g.setor = d.setor || g.setor;

      g.mercado = d.mercado || g.mercado;

      grupos.set(ticker, g);

      movimentosAsc.push({ date: dt || new Date(0), ticker, qtd, preco });

    });

    const gruposArr = Array.from(grupos.values());

    // 2) Info de cota / dividendos

    const tickers = gruposArr.map((g) => g.ticker);

    const infoMap = await fetchDividendInfoByTickers(tickers);

    const fmtEUR = new Intl.NumberFormat("pt-PT", {

      style: "currency",

      currency: "EUR",

    });

    const fmtDate = new Intl.DateTimeFormat("pt-PT", {

      year: "numeric",

      month: "short",

      day: "2-digit",

    });

    // Enriquecer grupos com métricas para gráficos/cards

    const rowsForYield = []; // para Top5 Yield

    gruposArr.forEach((g) => {

      const info = infoMap.get(g.ticker) || {};

      const precoAtual = isFiniteNum(info.valorStock)

        ? Number(info.valorStock)

        : null;

      const precoMedio = g.qtd > 0 ? g.investido / g.qtd : 0;

      g.lucroAtual =

        precoAtual !== null ? (precoAtual - precoMedio) \* g.qtd : 0;

      g.precoAtual = precoAtual;

      // yields & valuation

      const dividendo = toNum(info.dividendo);

      const dmed24 = toNum(info.dividendoMedio24m);

      const pe = isFiniteNum(info.peRatio) ? Number(info.peRatio) : null;

      const sma50 = isFiniteNum(info.sma50) ? Number(info.sma50) : null;

      const sma200 = isFiniteNum(info.sma200) ? Number(info.sma200) : null;

      const yCur = precoAtual && dividendo > 0 ? dividendo / precoAtual : null; // 0..1

      const y24m = precoAtual && dmed24 > 0 ? dmed24 / precoAtual : null; // 0..1

      g.\_yCur = yCur;

      g.\_y24m = y24m;

      g.\_pe = pe;

      g.\_sma50 = sma50;

      g.\_sma200 = sma200;

      rowsForYield.push({

        ticker: g.ticker,

        active: g.qtd > 0,

        yieldCur: yCur,

      });

    });

    // 2.1) Distribuições

    const setoresMap = new Map();

    const mercadosMap = new Map();

    for (const g of gruposArr) {

      if ((g.qtd || 0) <= 0) continue;

      const setor = g.setor || "—";

      setoresMap.set(setor, (setoresMap.get(setor) || 0) + (g.investido || 0));

      const merc = g.mercado || "—";

      mercadosMap.set(merc, (mercadosMap.get(merc) || 0) + (g.investido || 0));

    }

    // 2.2) KPIs agregados

    const totalInvestido = gruposArr

      .filter((g) => g.qtd > 0)

      .reduce((a, g) => a + (g.investido || 0), 0);

    const lucroTotal = gruposArr

      .filter((g) => g.qtd > 0)

      .reduce((a, g) => a + (g.lucroAtual || 0), 0);

    const retornoPct = totalInvestido ? (lucroTotal / totalInvestido) \* 100 : 0;

    // rendimento anual esperado

    let rendimentoAnual = 0;

    const eurosMes = new Array(12).fill(0); // calendário 12m

    for (const g of gruposArr) {

      if (g.qtd <= 0) continue;

      const info = infoMap.get(g.ticker) || {};

      const div = toNum(info.dividendo);

      const per = info.periodicidade;

      const mesT = info.mes;

      const payN = pagamentosAno(per);

      const anualTicker = g.qtd \* div \* payN;

      rendimentoAnual += anualTicker;

      // distribuir pelos meses estimados

      const meses = mesesPagos(per, mesT);

      for (const m of meses) {

        eurosMes[m] += g.qtd \* div;

      }

    }

    // exposição acima da SMA200 (peso por investido)

    let somaPesosAcima = 0;

    for (const g of gruposArr) {

      if (g.qtd <= 0 || !totalInvestido) continue;

      const w = (g.investido || 0) / totalInvestido;

      const p = g.precoAtual,

        s200 = g.\_sma200;

      if (isFiniteNum(p) && isFiniteNum(s200) && Number(p) > Number(s200)) {

        somaPesosAcima += w;

      }

    }

    const expSMA200Pct = somaPesosAcima \* 100;

    // preencher KPIs

    const elTI = document.getElementById("prtTotalInvestido");

    const elRA = document.getElementById("prtRendimentoAnual");

    const elRP = document.getElementById("prtRetorno");

    const elEX = document.getElementById("prtExpSMA200");

    if (elTI) elTI.textContent = fmtEUR.format(totalInvestido);

    if (elRA) elRA.textContent = fmtEUR.format(rendimentoAnual);

    if (elRP) elRP.textContent = `${retornoPct.toFixed(1)}%`;

    if (elEX) elEX.textContent = `${expSMA200Pct.toFixed(0)}%`;

    // 2.3) Timeline (como já tinhas)

    movimentosAsc.sort((a, b) => a.date - b.date);

    const qtyNow = new Map();

    const priceNow = new Map();

    gruposArr.forEach((g) => {

      if (isFiniteNum(g.precoAtual))

        priceNow.set(g.ticker, Number(g.precoAtual));

      qtyNow.set(g.ticker, 0);

    });

    let cumInvest = 0;

    const timelinePoints = [];

    for (const m of movimentosAsc) {

      const deltaInvest = Number(m.qtd) \* Number(m.preco);

      cumInvest += deltaInvest;

      qtyNow.set(m.ticker, (qtyNow.get(m.ticker) || 0) + m.qtd);

      let valueNow = 0;

      qtyNow.forEach((q, tk) => {

        const p = priceNow.get(tk);

        if (isFiniteNum(p)) valueNow += q \* Number(p);

      });

      timelinePoints.push({

        label: isFinite(m.date?.getTime?.()) ? fmtDate.format(m.date) : "",

        cumInvest: cumInvest,

        valueNow: valueNow,

      });

    }

    // 3) Render GRÁFICOS

    renderSetorDoughnut(setoresMap);

    renderMercadoDoughnut(mercadosMap);

    renderTop5Bar(gruposArr);

    renderTop5YieldBar(rowsForYield);

    renderTimeline(timelinePoints);

    renderDividendoCalendario12m(eurosMes);

    // 4) Render LISTA com badges extra

    const html = gruposArr

      .filter((g) => g.qtd > 0)

      .map((g) => {

        const info = infoMap.get(g.ticker) || {};

        const precoAtual = g.precoAtual;

        const precoMedio = g.qtd > 0 ? g.investido / g.qtd : 0;

        const lucroAtual = g.lucroAtual || 0;

        // barra zero ao centro (teu código)

        let pctText = "—";

        let barHTML = "";

        const objetivo = g.objetivo > 0 ? g.objetivo : 0;

        if (objetivo > 0) {

          const progresso = (lucroAtual / objetivo) \* 100;

          const clamped = Math.max(-100, Math.min(100, progresso));

          const sideWidthPct = (Math.abs(clamped) / 2).toFixed(1);

          const positive = clamped >= 0;

          pctText = `${clamped.toFixed(0)}%`;

          barHTML = `

            <div class="progress-dual">

              <div class="track">

                <div class="fill ${

                  positive ? "positive" : "negative"

                }" style="width:${sideWidthPct}%"></div>

                <div class="zero"></div>

              </div>

            </div>

          `;

        }

        const tp2Necessario =

          objetivo > 0 && g.qtd > 0 ? precoMedio + objetivo / g.qtd : null;

        const { taxa, periodLabel } = pickBestRate(info);

        const estimativa =

          tp2Necessario && precoAtual

            ? estimateTime(precoAtual, tp2Necessario, taxa, periodLabel)

            : "—";

        const dataTxt = g.lastDate ? fmtDate.format(g.lastDate) : "sem data";

        // badges rápidas

        const yCur = g.\_yCur; // 0..1

        const y24 = g.\_y24m; // 0..1

        const pe = g.\_pe;

        const s50 = g.\_sma50;

        const s200 = g.\_sma200;

        const yPct = isFiniteNum(yCur) ? (yCur \* 100).toFixed(2) + "%" : "—";

        const y24Pct = isFiniteNum(y24) ? (y24 \* 100).toFixed(2) + "%" : "—";

        const yBadge =

          isFiniteNum(yCur) && isFiniteNum(y24)

            ? yCur > y24

              ? "↑ acima da média"

              : "↓ abaixo da média"

            : "";

        const peBadge = isFiniteNum(pe)

          ? pe < 15

            ? "Barato"

            : pe <= 25

            ? "Justo"

            : "Caro"

          : "—";

        const d50 =

          isFiniteNum(s50) && isFiniteNum(precoAtual)

            ? ((precoAtual - s50) / s50) \* 100

            : null;

        const d200 =

          isFiniteNum(s200) && isFiniteNum(precoAtual)

            ? ((precoAtual - s200) / s200) \* 100

            : null;

        const d50Txt = d50 === null ? "—" : `${d50.toFixed(1)}%`;

        const d200Txt = d200 === null ? "—" : `${d200.toFixed(1)}%`;

        const stopLight =

          isFiniteNum(precoAtual) &&

          isFiniteNum(s200) &&

          precoAtual < s200 &&

          Number(info.taxaCrescimento\_1mes || 0) < 0

            ? "🔴"

            : (isFiniteNum(precoAtual) &&

                isFiniteNum(s200) &&

                precoAtual < s200) ||

              Number(info.taxaCrescimento\_1mes || 0) < 0

            ? "🟡"

            : isFiniteNum(precoAtual) &&

              isFiniteNum(s200) &&

              precoAtual > s200 &&

              Number(info.taxaCrescimento\_1mes || 0) > 0

            ? "🟢"

            : "⚪️";

        // ações

        const actions = `

          <div class="actions-row" style="margin-top:.5rem">

            <button class="btn outline" data-buy="${g.ticker}">Comprar</button>

            <button class="btn ghost"  data-sell="${g.ticker}">Vender</button>

          </div>

        `;

        // linha de análise extra

        const analysis = `

          <p class="muted" style="margin-top:.4rem">

            ${stopLight} Yield: <strong>${yPct}</strong> (${yBadge || "—"}) •

            Yield 24m: <strong>${y24Pct}</strong> •

            P/E: <strong>${

              isFiniteNum(pe) ? pe.toFixed(2) : "—"

            } (${peBadge})</strong> •

            Δ50d: <strong>${d50Txt}</strong> •

            Δ200d: <strong>${d200Txt}</strong>

          </p>

          <p class="muted">

            ${String(info.periodicidade || "n/A")} • paga em <strong>${String(

          info.mes || "n/A"

        )}</strong>

          </p>

        `;

        return `

          <div class="activity-item">

            <div class="activity-left">

              <span class="activity-icon">📦</span>

              <div>

                <p><strong>${g.nome}</strong> <span class="muted">(${

          g.ticker

        })</span></p>

                <p class="muted">${g.setor} • ${g.mercado}</p>

                <p class="muted">Última compra: ${dataTxt}</p>

                <p class="muted">

                  Qtd: <strong>${formatNum(g.qtd)}</strong> ·

                  Preço médio: <strong>${fmtEUR.format(

                    precoMedio || 0

                  )}</strong> ·

                  Preço atual: <strong>${

                    precoAtual !== null ? fmtEUR.format(precoAtual) : "—"

                  }</strong>

                </p>

                <p class="muted">

                  Investido: <strong>${fmtEUR.format(

                    g.investido || 0

                  )}</strong> ·

                  Lucro atual: <strong>${fmtEUR.format(lucroAtual)}</strong>

                </p>

                ${

                  objetivo > 0

                    ? `

                  <div class="activity-meta">

                    <span>Objetivo (lucro): <strong>${fmtEUR.format(

                      objetivo

                    )}</strong></span>

                    <span>${pctText}</span>

                  </div>

                  ${barHTML}

                  <p class="muted">

                    TP2 necessário: <strong>${

                      tp2Necessario ? fmtEUR.format(tp2Necessario) : "—"

                    }</strong>

                    ${

                      taxa !== null

                        ? `· Estimativa: <strong>${estimativa}</strong>`

                        : ``

                    }

                  </p>

                `

                    : `

                  <p class="muted">Sem objetivo definido para este ticker.</p>

                `

                }

                ${analysis}

                ${actions}

              </div>

            </div>

          </div>

        `;

      });

    cont.innerHTML = html.join("");

    // 5) Quick actions (teu)

    wireQuickActions(gruposArr);

    // 6) Re-render ao mudar de tema (com guard)

    if (window.\_\_prtThemeHandler) {

      window.removeEventListener("app:theme-changed", window.\_\_prtThemeHandler);

    }

    window.\_\_prtThemeHandler = () => {

      if (!document.getElementById("chartSetores")) return;

      renderSetorDoughnut(setoresMap);

      renderMercadoDoughnut(mercadosMap);

      renderTop5Bar(gruposArr);

      renderTop5YieldBar(rowsForYield);

      renderTimeline(timelinePoints);

      renderDividendoCalendario12m(eurosMes);

    };

    window.addEventListener("app:theme-changed", window.\_\_prtThemeHandler);

    // Popup Métricas - Explicação

    // Ajuda das Métricas: ligar handlers e mostrar ao entrar

    wirePortfolioHelpModal();

    showPortfolioHelp(/\* force = \*/ true); // ← abre SEMPRE quando entras neste ecrã

    // 7) Auto-clean quando saíres do ecrã

    const observer = new MutationObserver(() => {

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

        charts.setores?.destroy?.();

        charts.mercados?.destroy?.();

        charts.top5?.destroy?.();

        charts.top5Yield?.destroy?.();

        charts.timeline?.destroy?.();

        charts.divCal?.destroy?.();

        if (window.\_\_prtThemeHandler) {

          window.removeEventListener(

            "app:theme-changed",

            window.\_\_prtThemeHandler

          );

          window.\_\_prtThemeHandler = null;

        }

        observer.disconnect();

      }

    });

    observer.observe(document.body, { childList: true, subtree: true });

  } catch (e) {

    console.error("Erro ao carregar atividades:", e);

    cont.innerHTML = `<p class="muted">Não foi possível carregar a lista.</p>`;

    renderSetorDoughnut(new Map());

    renderMercadoDoughnut(new Map());

    renderTop5Bar([]);

    renderTop5YieldBar([]);

    renderTimeline([]);

    renderDividendoCalendario12m(new Array(12).fill(0));

  }

}

/\* Métricas Popup \*/

/\* ========= Ajuda das Métricas (popup) ========= \*/

const HELP\_KEY = "prt.help.dismissed";

function wirePortfolioHelpModal() {

  const modal   = document.getElementById("prtHelpModal");

  if (!modal || modal.\_\_wired) return;

  modal.\_\_wired = true;

  const closeBtn = document.getElementById("prtHelpClose");

  const okBtn    = document.getElementById("prtHelpOK");

  const laterBtn = document.getElementById("prtHelpLater");

  const dontShow = document.getElementById("prtHelpDontShow");

  const close = (persist) => {

    if (persist && dontShow?.checked) {

      try { localStorage.setItem(HELP\_KEY, "1"); } catch {}

    }

    modal.classList.add("hidden");

  };

  closeBtn?.addEventListener("click", () => close(false));

  laterBtn?.addEventListener("click", () => close(false));

  okBtn?.addEventListener("click", () => close(true));

  // fecha ao clicar fora do diálogo

  modal.addEventListener("click", (e) => {

    if (e.target === modal) close(false);

  });

  // fecha com ESC

  document.addEventListener("keydown", (e) => {

    if (modal.classList.contains("hidden")) return;

    if (e.key === "Escape") close(false);

  });

}

function showPortfolioHelp(force = false) {

  const modal = document.getElementById("prtHelpModal");

  if (!modal) return;

  // se NÃO for force e o user marcou "não mostrar", não abre

  if (!force) {

    try {

      if (localStorage.getItem(HELP\_KEY) === "1") return;

    } catch {}

  }

  modal.classList.remove("hidden");

}

# HTML | dasboard.html

[ir para o topo](#_top)

<div class="dashboard" data-screen-title="Dashboard">

  <div class="dashboard-header">

    <div class="dashboard-title">

      <p class="subtitle">Visão geral do seu portfólio de investimentos</p>

    </div>

  </div>

  <div class="metrics-grid">

    <div class="card premium">

      <div class="card-content">

        <div>

          <p class="label">Valor Total Investido</p>

          <p class="value valor-total" id="valorTotal">€0.00</p>

          <p id="valorCarteira" class="subvalue muted">—</p>

        </div>

        <div class="icon-box">

          <i class="fas fa-coins"></i>

        </div>

      </div>

    </div>

    <div class="card success">

      <div class="card-content">

        <div>

          <p class="label">Retorno Total</p>

          <p class="value retorno-total" id="retornoTotal">0%</p>

        </div>

        <div class="icon-box">

          <i class="fas fa-chart-line"></i>

        </div>

      </div>

    </div>

    <div class="card info">

      <div class="card-content">

        <div>

          <p class="label">Lucro Total</p>

          <p class="value lucro-total" id="lucroTotal">€0.00</p>

        </div>

        <div class="icon-box">

          <i class="fas fa-wallet"></i>

        </div>

      </div>

    </div>

    <div class="card default">

      <div class="card-content">

        <div>

          <p class="label">Posições Ativas</p>

          <p class="value posicoes-ativas" id="posicoesAtivas">0</p>

          <p class="subvalue">

            Objetivos:

            <span class="objetivos-financeiros" id="objetivosFinanceiros"

              >0/0</span

            >

          </p>

        </div>

        <div class="icon-box">

          <i class="fas fa-bullseye"></i>

        </div>

      </div>

    </div>

    <div class="card warning">

      <div class="card-content">

        <div>

          <p class="label">Taxa de Sucesso</p>

          <p class="value taxa-sucesso" id="taxaSucesso">0%</p>

          <p class="subvalue">

            Objetivo Total: <span id="objetivoTotal">€0.00</span>

          </p>

        </div>

        <div class="icon-box">

          <i class="fas fa-rocket"></i>

        </div>

      </div>

    </div>

  </div>

  <div class="dashboard-columns">

    <div class="card glass">

      <h3>Atividade Recente (compra/venda)</h3>

      <div id="atividadeRecente"></div>

      <button class="btn outline full" onclick="navigateTo('atividade')">

        Ver Toda Atividade

      </button>

    </div>

    <div class="card glass">

      <h3>Ações Rápidas</h3>

      <button class="btn premium full" id="btnDashBuy">Registar Compra</button>

      <button class="btn outline full" id="btnOportunidades">

        Analisar Oportunidades

      </button>

      <button class="btn outline full" id="btnAddAcao">Adicionar Ação</button>

    </div>

  </div>

  <!-- POPUP: Top 10 Oportunidades -->

  <div id="opModal" class="modal hidden">

    <div class="modal-dialog">

      <div class="modal-header">

        <h3>Top 10 Oportunidades</h3>

        <button class="icon-btn" id="opClose" title="Fechar">✖</button>

      </div>

      <div class="modal-toolbar">

        <button class="chip" data-periodo="1s">1 semana</button>

        <button class="chip" data-periodo="1m">1 mês</button>

        <button class="chip" data-periodo="1ano">1 ano</button>

      </div>

      <div class="top10-box">

        <ul id="listaTop10"></ul>

      </div>

    </div>

  </div>

  <!-- Modal: Adicionar Ação -->

  <div id="addModal" class="modal hidden">

    <div class="modal-dialog">

      <div class="modal-header">

        <h3>Adicionar Ação</h3>

        <button class="icon-btn" id="addClose" title="Fechar">✖</button>

      </div>

      <form id="addForm" style="padding: 14px 16px">

        <div class="form-grid">

          <label>

            Tipo de operação

            <select id="tipoAcao" required>

              <option value="compra" selected>Compra</option>

              <option value="venda">Venda</option>

            </select>

          </label>

          <label>

            Nome

            <input

              id="nomeAtivo"

              type="text"

              placeholder="Ex.: Apple Inc."

              required

            />

          </label>

          <label>

            Ticker

            <input

              id="tickerAtivo"

              type="text"

              placeholder="Ex.: AAPL"

              required

            />

          </label>

          <label>

            Setor

            <select id="setorAtivo" required>

              <option value="">Selecione um Setor</option>

              <option value="ETF iTech">ETF iTech</option>

              <option value="ETF Finance">ETF Finance</option>

              <option value="ETF Energia">ETF Energia</option>

              <option value="ETF Materiais">ETF Materiais</option>

              <option value="Alimentação">Alimentação</option>

              <option value="Automóvel">Automóvel</option>

              <option value="Bens Consumidor">Bens Consumidor</option>

              <option value="Consumo Cíclico">Consumo Cíclico</option>

              <option value="Consumo Defensivo">Consumo Defensivo</option>

              <option value="Criptomoedas">Criptomoedas</option>

              <option value="Defesa">Defesa</option>

              <option value="Energia">Energia</option>

              <option value="Finanças">Finanças</option>

              <option value="Imobiliário">Imobiliário</option>

              <option value="Indústria">Indústria</option>

              <option value="Infraestruturas / Energia">Infraestruturas / Energia</option>

              <option value="Materiais">Materiais</option>

              <option value="Mineração (Ouro)">Mineração (Ouro)</option>

              <option value="Restauração">Restauração</option>

              <option value="Saúde">Saúde</option>

              <option value="Tecnologia">Tecnologia</option>

              <option value="Telecomunicações">Telecomunicações</option>

            </select>

          </label>

          <label>

            Mercado

            <select id="mercadoAtivo" required>

              <option value="">Seleciona o Mercado</option>

              <option value="Portugal">Portugal</option>

              <option value="Europeu">Europeu</option>

              <option value="Americano">Americano</option>

              <option value="Americano SP500">Americano SP500</option>

            </select>

          </label>

          <label>

            Quantidade

            <input

              id="quantidadeAtivo"

              type="number"

              step="0.0001"

              min="0.0001"

              required

            />

          </label>

          <label id="labelPreco">

            Preço da transação (€)

            <input

              id="precoAtivo"

              type="number"

              step="0.0001"

              min="0"

              required

            />

          </label>

          <label>

            Objetivo financeiro (€) <small class="muted">(opcional)</small>

            <input id="objetivoAtivo" type="number" step="0.01" min="0" />

          </label>

        </div>

        <div class="actions-row" style="margin-top: 12px">

          <button type="submit" class="btn premium">Guardar</button>

          <button type="button" class="btn outline" id="addCancel">

            Cancelar

          </button>

        </div>

        <p class="muted" style="margin-top: 8px">

          A data/hora é registada automaticamente.

        </p>

      </form>

    </div>

  </div>

</div>

# HTML | atividade.html

[ir para o topo](#_top)

<div class="page" data-screen-title="Portfólio">

  <div class="dashboard-header">

    <p class="subtitle">Todos os ativos em carteira</p>

  </div>

  <!-- 🧭 KPIs -->

  <div class="portfolio-summary">

    <div class="card premium">

      <div class="card-content">

        <div>

          <p class="label">Valor Total Investido</p>

          <p class="value" id="prtTotalInvestido">€0,00</p>

        </div>

        <div class="icon-box"><i class="fas fa-coins"></i></div>

      </div>

    </div>

    <div class="card success">

      <div class="card-content">

        <div>

          <p class="label">Dividendos anuais esperado</p>

          <p class="value" id="prtRendimentoAnual">€0,00</p>

        </div>

        <div class="icon-box"><i class="fas fa-wallet"></i></div>

      </div>

    </div>

    <div class="card info">

      <div class="card-content">

        <div>

          <p class="label">Retorno global (P/L%)</p>

          <p class="value" id="prtRetorno">0%</p>

        </div>

        <div class="icon-box"><i class="fas fa-chart-line"></i></div>

      </div>

    </div>

    <div class="card default">

      <div class="card-content">

        <div>

          <p class="label">Exposição acima da SMA200</p>

          <p class="value" id="prtExpSMA200">0%</p>

        </div>

        <div class="icon-box"><i class="fas fa-trending-up"></i></div>

      </div>

    </div>

  </div>

  <!-- 📈 GRÁFICOS (linha 1: Setores + Mercado) -->

  <div class="portfolio-charts">

    <div class="card">

      <h3>Distribuição por Setores</h3>

      <div class="chart-wrap"><canvas id="chartSetores"></canvas></div>

    </div>

    <div class="card">

      <h3>Distribuição por Mercado</h3>

      <div class="chart-wrap"><canvas id="chartMercados"></canvas></div>

    </div>

  </div>

  <!-- 📈 GRÁFICOS (linha 2: Top5 Investido + Top5 Yield) -->

  <div class="portfolio-charts">

    <div class="card">

      <h3>Top 5 por Valor Investido</h3>

      <div class="chart-wrap"><canvas id="chartTop5"></canvas></div>

    </div>

    <div class="card">

      <h3>Top 5 por Yield</h3>

      <div class="chart-wrap"><canvas id="chartTop5Yield"></canvas></div>

    </div>

  </div>

  <!-- 📈 GRÁFICOS (linha 3: Timeline + Calendário Dividendos) -->

  <div class="portfolio-charts">

    <div class="card">

      <h3>Evolução (Investido vs Avaliação Atual)</h3>

      <div class="chart-wrap"><canvas id="chartTimeline"></canvas></div>

    </div>

    <div class="card">

      <h3>Calendário de Dividendos (12 meses)</h3>

      <div class="chart-wrap"><canvas id="chartDivCalendario"></canvas></div>

    </div>

  </div>

  <!-- 🔎 LISTA -->

  <div id="listaAtividades"></div>

</div>

<!-- Modal QUICK BUY/SELL (mantém o teu modal existente) -->

<!-- ... o teu #pfAddModal aqui, sem alterações ... -->

<!-- 📘 Ajuda das Métricas -->

<div id="prtHelpModal" class="modal hidden">

  <!-- nota: acrescentei a classe prt-help-dialog -->

  <div class="modal-dialog prt-help-dialog">

    <div class="modal-header">

      <h3>Como ler este ecrã</h3>

      <button class="icon-btn" id="prtHelpClose" title="Fechar">✖</button>

    </div>

    <!-- corpo rolável -->

    <div class="help-body">

      <div class="page" style="padding:14px">

        <div class="muted" style="display:grid; gap:.6rem; line-height:1.4">

          <div><strong>Valor Total Investido</strong> — soma de todo o capital investido nas posições ativas.</div>

          <div><strong>Rendimento anual esperado</strong> — estimativa: Σ(qtd × dividendo × pagamentos/ano) por ativo, de acordo com a periodicidade.</div>

          <div><strong>Retorno global (P/L%)</strong> — (Avaliação atual − Investido) / Investido.</div>

          <div><strong>Exposição acima da SMA200</strong> — percentagem do portefólio (ponderada pelo investidor) com preço acima da média móvel 200.</div>

          <hr>

          <div><strong>Distribuição por Setores / Mercado</strong> — donuts com o peso investido por categoria.</div>

          <div><strong>Top 5 por Investido</strong> — barras com as maiores posições (investido e lucro atual).</div>

          <div><strong>Top 5 por Yield</strong> — barras horizontais ordenadas por dividend yield atual.</div>

          <div><strong>Evolução</strong> — linha “Investido acumulado” vs “Avaliação atual”.</div>

          <div><strong>Calendário de Dividendos (12m)</strong> — soma estimada de € por mês, com base no mês típico e periodicidade.</div>

          <hr>

          <div><strong>Badges por ativo</strong>:

            <ul style="margin:.3rem 0 0 1.1rem">

              <li><em>Yield</em> (dividendo/valorStock) e <em>Yield 24m</em> (média 24m) → indica se está ↑ acima ou ↓ abaixo da média.</li>

              <li><em>P/E</em> (faixas: &lt;15 barato, 15–25 justo, &gt;25 caro).</li>

              <li><em>Δ50d / Δ200d</em> — variação vs. SMA 50/200.</li>

              <li><em>Semáforo</em> — tendência rápida (🟢/🟡/🔴) com base em SMA200 e crescimento de 1 mês.</li>

              <li><em>TP2 &amp; ETA</em> — preço alvo e estimativa de tempo para lá chegar (usando a melhor taxa disponível).</li>

            </ul>

          </div>

        </div>

      </div>

    </div>

    <!-- ações fixas em baixo -->

    <div class="actions-row help-actions" style="justify-content:space-between">

      <label class="muted" style="display:flex; align-items:center; gap:.5rem">

        <input type="checkbox" id="prtHelpDontShow">

        Não mostrar novamente neste dispositivo

      </label>

      <div>

        <button class="btn outline" id="prtHelpLater">Fechar</button>

        <button class="btn premium" id="prtHelpOK">Percebi</button>

      </div>

    </div>

  </div>

</div>

# HTML | Simulador.html

[ir para o topo](#_top)

<div class="Simulador" data-screen-title="Simulador">

  <div class="Simulador-header">

    <div class="Simulador">

      <p class="subtitle">Screen para simular investimentos ou posições</p>

    </div>

  </div>

  <div class="screen-body sim-layout">

    <!-- COLUNA ESQUERDA: BOTÕES -->

    <aside class="sim-sidebar">

      <button class="btn dark full" data-target="panel-grafico">

        Simular com Gráfico

      </button>

      <button class="btn dark full" data-target="panel-reforco">

        Simular Reforço

      </button>

      <button class="btn dark full" data-target="panel-tp2">

        Calcular TP2

      </button>

      <button class="btn dark full" data-target="panel-top10">

        TOP 10 para Investir

      </button>

    </aside>

    <!-- COLUNA DIREITA: “POPUPS” (PAINÉIS) -->

    <section class="sim-content">

      <!-- PAINEL: SIMULAR COM GRÁFICO -->

      <div id="panel-grafico" class="panel active">

        <h3>📈 Simulador com Gráfico</h3>

        <div class="form-grid">

          <input type="text" id="nomeAcao" placeholder="Nome da Ação" />

          <input type="number" id="tp1" placeholder="TP1 (Preço inicial)" />

          <input type="number" id="tp2" placeholder="TP2 (Preço final)" />

          <input type="number" id="investimento" placeholder="Investido (€)" />

          <input

            type="number"

            id="dividendo"

            placeholder="Dividendo por ação (€)"

          />

        </div>

        <div class="quick-buttons">

          <button class="btn ghost" data-quick="500">💶 500</button>

          <button class="btn ghost" data-quick="1000">💶 1000</button>

          <button class="btn ghost" data-quick="1500">💶 1500</button>

          <button class="btn ghost" data-quick="2000">💶 2000</button>

        </div>

        <div class="actions-row">

          <button class="btn premium" id="btnSimularGrafico">

            Simular com gráfico

          </button>

          <button class="btn outline" id="btnSomarLucros">Somar Lucros</button>

          <button class="btn outline" id="btnLimparTabela">Limpar</button>

          <button class="btn outline" id="btnLimparGrafico">

            Limpar Gráfico

          </button>

          <button class="btn outline" id="btnEnviarEmail">Enviar Email</button>

        </div>

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

          <table id="tabelaSimulacoes">

            <thead>

              <tr>

                <th>Ação</th>

                <th>TP1</th>

                <th>TP2</th>

                <th>Investido</th>

                <th>Lucro</th>

                <th>% Crescimento</th>

              </tr>

            </thead>

            <tbody></tbody>

          </table>

        </div>

        <div style="height: 300px">

          <canvas id="graficoLucro"></canvas>

        </div>

      </div>

      <!-- PAINEL: REFORÇO -->

      <div id="panel-reforco" class="panel">

        <h3>Simular Reforço de Investimento</h3>

        <div class="form-grid">

          <input

            type="number"

            id="invest1"

            placeholder="Investimento Inicial (€)"

          />

          <input type="number" id="preco1" placeholder="Preço por Ação 1 (€)" />

          <input type="number" id="invest22" placeholder="Reforço (€)" />

          <input type="number" id="preco2" placeholder="Preço por Ação 2 (€)" />

        </div>

        <div class="actions-row">

          <button class="btn premium" id="btnCalcularReforco">Calcular</button>

          <button class="btn outline" id="btnLimparReforco">Limpar</button>

        </div>

        <div id="resultadoReforco" class="resultado-popup"></div>

      </div>

      <!-- PAINEL: CALCULAR TP2 -->

      <div id="panel-tp2" class="panel">

        <h3>🎯 Calcular TP2</h3>

        <div class="form-grid">

          <input

            type="number"

            id="tp1Input"

            placeholder="TP1 (preço de compra €)"

          />

          <input

            type="number"

            id="investimentoInput"

            placeholder="Valor Investido (€)"

          />

          <input

            type="number"

            id="lucroDesejadoInput"

            placeholder="Lucro Desejado (€)"

          />

        </div>

        <div class="actions-row">

          <button class="btn premium" id="btnCalcularTP2">Calcular TP2</button>

          <button class="btn outline" id="btnLimparTP2">Limpar</button>

        </div>

        <div id="resultadoTP2" class="resultado-popup"></div>

      </div>

      <!-- PAINEL: TOP 10 -->

    <div id="panel-top10" class="panel">

      <h3>🔎 TOP 10 para Investir</h3>

      <div class="form-grid">

        <label>

          Montante a investir (€)

          <input

            type="number"

            id="inputInvestimento"

            placeholder="Ex: 1500"

            min="0"

            step="0.01"

            required

          />

        </label>

        <label>

          Período de crescimento

          <select id="inputPeriodo">

            <option value="1ano" selected>1 ano</option>

            <option value="1m">1 mês</option>

            <option value="1s">1 semana</option>

          </select>

        </label>

        <!-- (Opcional) multiplicador de horizonte: 1x (1 ano), 2x (2 anos), etc. -->

        <!-- podes remover se não quiseres -->

        <label>

          Horizonte (anos)

          <input type="number" id="inputHorizonte" value="1" min="1" step="1" />

        </label>

      </div>

      <div class="actions-row" style="align-items:center;">

        <label style="display:flex; gap:.5rem; align-items:center;">

          <input type="checkbox" id="chkUsarTotal" checked />

          <span>Usar todo o investimento</span>

        </label>

        <label style="display:flex; gap:.5rem; align-items:center;">

          <input type="checkbox" id="chkAcoesCompletas" />

          <span>Apenas ações completas</span>

        </label>

      </div>

  <div class="actions-row">

    <button class="btn premium" id="btnSimularTop10">Simular</button>

    <button class="btn outline" id="btnLimparTop10">Limpar</button>

  </div>

  <div id="resultadoSimulacao" class="resultado-container"></div>

</div>

    </section>

  </div>

</div>

<!-- Chart.js (apenas aqui, porque este screen usa gráfico) -->

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script type="module" src="js/screens/simulador.js"></script>

# HTML | analise.html

[ir para o topo](#_top)

<div class="page" data-screen-title="Análise Fina">

  <div class="dashboard-header">

    <p class="subtitle">Métricas detalhadas por ticker — base de dados completa</p>

  </div>

  <!-- KPIs -->

  <div class="analysis-summary">

    <div class="card info">

      <div class="card-content">

        <div>

          <p class="label">Nº de Tickers</p>

          <p class="value" id="anTotalTickers">0</p>

        </div>

        <div class="icon-box"><i class="fas fa-database"></i></div>

      </div>

    </div>

    <div class="card success">

      <div class="card-content">

        <div>

          <p class="label">Yield média (amostra)</p>

          <p class="value" id="anYieldMedia">0%</p>

        </div>

        <div class="icon-box"><i class="fas fa-percentage"></i></div>

      </div>

    </div>

    <div class="card premium">

      <div class="card-content">

        <div>

          <p class="label">P/E médio (amostra)</p>

          <p class="value" id="anPEmedio">0.0</p>

        </div>

        <div class="icon-box"><i class="fas fa-balance-scale"></i></div>

      </div>

    </div>

  </div>

  <!-- GRÁFICOS -->

  <div class="analysis-charts">

    <div class="card">

      <h3>Distribuição por Setor (nº de tickers)</h3>

      <div class="chart-wrap"><canvas id="anChartSetor"></canvas></div>

    </div>

    <div class="card">

      <h3>Distribuição por Mercado (nº de tickers)</h3>

      <div class="chart-wrap"><canvas id="anChartMercado"></canvas></div>

    </div>

    <div class="card">

      <h3>Yield vs P/E (scatter)</h3>

      <div class="chart-wrap"><canvas id="anChartYieldPE"></canvas></div>

    </div>

  </div>

  <div class="card">

  <!-- 🔎 TOOLBAR POR CIMA DA TABELA -->

  <div class="analysis-toolbar">

    <!-- Linha 1: pesquisa (sempre 100%) -->

    <div class="toolbar-row">

      <input id="anSearch" type="search" placeholder="Pesquisar por ticker / nome…" />

    </div>

    <!-- Linha 2: filtros + ordenar (100%; em desktop distribui em colunas largas) -->

    <div class="toolbar-row toolbar-grid">

      <select id="anFiltroSetor"    title="Filtrar por setor"><option value="">Todos os setores</option></select>

      <select id="anFiltroMercado"  title="Filtrar por mercado"><option value="">Todos os mercados</option></select>

      <select id="anOrdenar"        title="Ordenar resultados">

        <option value="ticker.asc">Ticker (A→Z)</option>

        <option value="yield.desc">Yield ↓</option>

        <option value="pe.asc">P/E ↑</option>

        <option value="delta200.desc">Δ200d% ↓</option>

      </select>

    </div>

  </div>

  <!-- TABELA -->

        <div class="table-responsive">

          <table class="analysis-table">

            <thead>

              <tr>

                <th data-sort="ticker">Ticker</th>

                <th>Nome</th>

                <th data-sort="yield">Yield</th>

                <th>Yield 24m</th>

                <th data-sort="pe">P/E</th>

                <th>Δ50d%</th>

                <th data-sort="delta200">Δ200d%</th>

                <th>1S</th>

                <th>1M</th>

                <th>1A</th>

                <th>Periodicidade</th>

                <th>Mês típico</th>

                <th>Notas</th>

              </tr>

            </thead>

            <tbody id="analysisTableBody"></tbody>

          </table>

        </div>

      </div>

      </div>

    </div>

  </div>

</div>

# JS | analise.js

[ir para o topo](#_top)

// 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";

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

   Chart.js on-demand

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

async function ensureChartJS() {

  if (window.Chart) return;

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

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

    s.src = "https://cdn.jsdelivr.net/npm/chart.js@4.4.3/dist/chart.umd.min.js";

    s.onload = resolve; s.onerror = reject; document.head.appendChild(s);

  });

}

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,.7)",

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

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

   Helpers

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

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));

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

   Dividendos

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

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 pagamentosAno(periodicidade) {

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

  if (p === "Mensal") return 12;

  if (p === "Trimestral") return 4;

  if (p === "Semestral") return 2;

  if (p === "Anual") return 1;

  return 0;

}

function anualPreferido(doc){

  const d24 = toNum(doc.dividendoMedio24m);

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

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

}

function perPayment(doc){

  const base = toNum(doc.dividendo);           // por pagamento (se existir)

  if (base > 0) return base;

  const anual = anualPreferido(doc);

  const pAno = pagamentosAno(doc.periodicidade);

  return pAno > 0 ? anual / pAno : 0;

}

function computeYieldPct(annualDividend, valorStock){

  if (!Number.isFinite(annualDividend) || !Number.isFinite(valorStock) || valorStock<=0) return 0;

  return (annualDividend / valorStock) \* 100;

}

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

   Crescimento  ❗ FIX: usar g1w/g1m/g1y (normalizados)

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

function campoCrescimentoPreferido(periodoSel) {

  if (periodoSel === "1s") return "g1w";

  if (periodoSel === "1m") return "g1m";

  return "g1y";

}

function melhorTaxaCrescimento(rowNormalizada, prefer) {

  // rowNormalizada tem g1w/g1m/g1y já calculados ao carregar da BD

  const ordem = [prefer, "g1m", "g1w", "g1y"];

  for (const k of ordem) {

    const v = Number(rowNormalizada?.[k] ?? 0);

    if (v !== 0 && Number.isFinite(v)) return v;

  }

  return 0;

}

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

   Métricas por AÇÃO (linear)

   Lucro por unidade (por ANO) = Dividendo anual + (Preço × taxa)

   No horizonte h anos: multiplicamos por h.

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

function calcularMetricasAcao(acao, { periodo="1m", horizonte=1, incluirDiv=true } = {}) {

  const precoAtual = toNum(acao.valorStock);

  const anualDiv   = anualPreferido(acao);     // €/ação/ano

  const prefer     = campoCrescimentoPreferido(periodo);

  const taxaPct    = melhorTaxaCrescimento(acao, prefer); // % já dos campos g1\*

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

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

  // Linear

  const growthPorAno = precoAtual \* r;               // €/ação/ano

  const valorizacaoNoHorizonte = growthPorAno \* h;   // linear

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

  const lucroUnidade = dividendosNoHorizonte + valorizacaoNoHorizonte;

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

  return {

    preco: precoAtual,

    dividendoAnual: anualDiv,             // €/ação/ano

    taxaPct,

    totalDividendos: dividendosNoHorizonte, // €/ação no horizonte

    valorizacao: valorizacaoNoHorizonte,    // €/ação no horizonte

    lucroUnidade,

    retornoPorEuro

  };

}

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

   Distribuição — FRAÇÕES (gasta tudo)

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

function distribuirFracoesFracionadas(acoes, investimento){

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

  if (soma <= 0) {

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

  }

  const linhas = acoes.map(a=>{

    const propor    = a.metrics.retornoPorEuro / soma;

    const investido = investimento \* propor;

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

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

    const divAnualAlloc   = qtd \* a.metrics.dividendoAnual;   // anual (por ano)

    const divPeriodoAlloc = qtd \* a.metrics.totalDividendos;  // no horizonte

    const valorizAlloc    = qtd \* a.metrics.valorizacao;      // no horizonte

    return {

      nome: a.nome, ticker: a.ticker,

      preco: a.metrics.preco,

      quantidade: qtd,

      investido,

      lucro,

      taxaPct: a.metrics.taxaPct,

      dividendoAnual: a.metrics.dividendoAnual,

      divAnualAlloc, divPeriodoAlloc, valorizAlloc

    };

  });

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

  const totalGasto      = linhas.reduce((s,l)=>s + l.investido, 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, totalDivAnual, totalDivPeriodo, totalValoriz, restante: Math.max(0, investimento - totalGasto) };

}

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

   Distribuição — APENAS INTEIROS (greedy top-up)

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

function distribuirFracoesInteiras(acoes, investimento){

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

  if (soma <= 0) {

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

  }

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

  // alvo proporcional + arredonda

  const base = ordered.map(a=>{

    const propor      = a.metrics.retornoPorEuro / soma;

    const investAlvo  = investimento \* propor;

    const preco       = a.metrics.preco;

    const qtd         = Math.max(0, Math.floor(preco > 0 ? investAlvo / preco : 0));

    return { a, qtd };

  });

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

  let restante = investimento - gasto;

  // greedy top-up

  while (true){

    const idx = ordered.findIndex(o => o.metrics.preco <= restante);

    if (idx === -1) break;

    const found = base.find(x => x.a === ordered[idx]);

    if (!found) break;

    found.qtd += 1;

    gasto += ordered[idx].metrics.preco;

    restante = investimento - gasto;

    if (restante < Math.min(...ordered.map(o => o.metrics.preco))) break;

  }

  const linhas = base.map(({a, qtd})=>{

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

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

    const divAnualAlloc   = qtd \* a.metrics.dividendoAnual;

    const divPeriodoAlloc = qtd \* a.metrics.totalDividendos;

    const valorizAlloc    = qtd \* a.metrics.valorizacao;

    return {

      nome: a.nome, ticker: a.ticker,

      preco: a.metrics.preco,

      quantidade: qtd,

      investido,

      lucro,

      taxaPct: a.metrics.taxaPct,

      dividendoAnual: a.metrics.dividendoAnual,

      divAnualAlloc, divPeriodoAlloc, valorizAlloc

    };

  });

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

  const totalGasto      = linhas.reduce((s,l)=>s + l.investido, 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, totalDivAnual, totalDivPeriodo, totalValoriz, restante: Math.max(0, investimento - totalGasto) };

}

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

   Seleção / Ordenação / Tabela / Heatmap

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

const selectedTickers = new Set();

const updateSelCount = ()=>{ const el = document.getElementById("anlSelCount"); if (el) el.textContent = String(selectedTickers.size); };

let sortKey = null, 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 \*/

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());

  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%",

      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}: ${pct}%`;

          } } } }

    }

  });

}

function renderTopYield(elId, rows) {

  const el = document.getElementById(elId); if (!el) return null;

  const top = [...rows].filter(r => Number.isFinite(r.yield) && r.yield > 0).sort((a,b)=>b.yield-a.yield).slice(0,5);

  return new Chart(el, {

    type: "bar",

    data: { labels: top.map(r=>r.ticker), datasets: [{ label:"Yield (%)", data: top.map(r=>r.yield), backgroundColor:"#22C55E" }]},

    options: {

      indexAxis:"y", responsive:true, maintainAspectRatio: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 = r[key] || "—"; map.set(k, (map.get(k) || 0) + 1); });

    return map;

  };

  charts.setor?.destroy(); charts.mercado?.destroy(); charts.topYield?.destroy();

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

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

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

}

/\* Heatmap \*/

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"); if (!body) return;

  const values = rows.map(r => perPayment(r)).filter(v => v > 0).sort((a,b)=>a-b);

  const q1 = values.length ? values[Math.floor(values.length \* 0.33)] : 0.01;

  const q2 = values.length ? values[Math.floor(values.length \* 0.66)] : 0.02;

  body.innerHTML = rows.map(r => {

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

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

    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("");

}

/\* Tabela \*/

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

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

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 = anualPreferido(d);

    const y = computeYieldPct(anual, valor);

    rows.push({

      ticker,

      nome: d.nome || "",

      setor: d.setor || "",

      mercado: d.mercado || "",

      valorStock: valor,

      dividendo: toNum(d.dividendo),                // por pagamento

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

      periodicidade: d.periodicidade || "",

      mes: d.mes || "",

      observacao: d.observacao || "",

      // derivados

      divPer: perPayment(d),

      divAnual: anual,

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

      // ❗ Normalizados para g1w/g1m/g1y (é ISTO que o algoritmo vai usar)

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

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

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

      // outros

      yield24: Number.isFinite(d.yield24) ? Number(d.yield24) : null,

      pe: Number.isFinite(d.pe) ? Number(d.pe) : null,

      delta50: Number.isFinite(d.delta50) ? Number(d.delta50) : null,

      delta200: Number.isFinite(d.delta200) ? Number(d.delta200) : 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);

  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);

}

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

   Modal (open/close + render)

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

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.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(4)}</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>

  `;

}

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

   INIT

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

export async function initScreen() {

  await ensureChartJS();

  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();

  });

  // Seleção

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

    const check = e.target.checked;

    // aplica sobre a lista filtrada atual

    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/Fechar modal

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

    if (selectedTickers.size === 0) { alert("Seleciona pelo menos uma ação para simular."); return; }

    openSimModal();

  });

  document.getElementById("anlSimClose")?.addEventListener("click", closeSimModal);

  document.getElementById("anlSimModal")?.addEventListener("click", (e)=>{ if (e.target.id === "anlSimModal") closeSimModal(); });

  // Exclusividade das opções

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

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

  cbTotal?.addEventListener("change", ()=>{ if (cbTotal.checked) cbInteiro.checked = false; else cbInteiro.checked = true; });

  cbInteiro?.addEventListener("change", ()=>{ if (cbInteiro.checked) cbTotal.checked = false; else cbTotal.checked = true; });

  // Calcular simulação

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

    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;

    if (!(investimento>0)){ alert("Indica um investimento total válido."); return; }

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

    const acoes = selecionadas.map(a=>{

      const metrics = calcularMetricasAcao(a, { periodo, horizonte, incluirDiv });

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

    }).filter(Boolean);

    if (acoes.length === 0){ alert("Sem métricas suficientes nas ações selecionadas."); return; }

    const res = (apenasInteiros && !usarFracoes)

      ? distribuirFracoesInteiras(acoes, investimento)

      : distribuirFracoesFracionadas(acoes, investimento);

    renderResultadoSimulacao(res);

  });

  // Exportar seleção

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

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

      .map(r => ({

        ticker: r.ticker, nome: r.nome, preco: r.valorStock,

        divPer: r.divPer, divAnual: r.divAnual, periodicidade: r.periodicidade, mes: r.mes, yield: r.yield

      }));

    localStorage.setItem("simulacaoSelecionados", JSON.stringify(selecionadas));

    alert(`Exportado ${selecionadas.length} tickers. Podes abrir o ecrã do simulador.`);

    closeSimModal();

  });

  // Render inicial

  markSortedHeader(); applyFilters();

}

/\* Auto-init \*/

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

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

  initScreen().catch(console.error);

}