**FinanceAPP 2.0 | ficheiros conteúdo projecto**

Indices

[JS | dashboard.js 2](#_Toc205640781)

[JS | simulador.js 6](#_Toc205640782)

[JS | atividade.js 12](#_Toc205640783)

[HTML | dasboard.html 16](#_Toc205640784)

[HTML | atividade.html 17](#_Toc205640785)

[HTML | Simulador.html 18](#_Toc205640786)

**Push para o gitHub**

git add .

git commit -m "Mensagem"

git push

# JS | dashboard.js

[ir para o topo](#_top)

// js/screens/dashboard.js

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

import {

  getDocs, collection, query, orderBy, limit

} 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;

    // 💰 Valor em carteira (total investido + lucro/prejuízo)

    const valorCarteira = totalInvestido + totalLucro;

    // 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)

  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);

    });

  });

}

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

   ATIVIDADE RECENTE (SIMPLIFICADA)

   ========================= \*/

async function carregarAtividadeRecenteSimplificada() {

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

  if (!cont) return;

  try {

    const q = query(collection(db, "ativos"), orderBy("dataCompra", "desc"), limit(8));

    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" // ex.: GMT+1/UTC+1 (depende do ambiente)

    });

    const html = [];

    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);

      // Data completa (ex.: "1 de agosto de 2025, 00:00:00 GMT+1")

      let dataTxt = "sem data";

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

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

      }

      html.push(`

        <div class="activity-item">

          <div class="activity-left">

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

            <div>

              <p><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>

      `);

    });

    cont.innerHTML = html.join("");

  } catch (e) {

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

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

  }

}

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

   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);

  });

}

# JS | simulador.js

[ir para o topo](#_top)

// js/screens/simulador.js

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

let simulacoes = [];

let grafico = null;

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

   HELPERS

   ========================= \*/

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 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", // podes mudar para 'line' se quiseres

    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, // permite ocupar altura do container

      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 });

  // Limpa apenas os inputs (mantém gráfico+tabela)

  limparInputsSimulacao();

}

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 (placeholder local)

   ========================= \*/

function simularTop10() {

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

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

  const cres  = toNumber(document.getElementById("inputCrescimento")?.value); // opcional (%)

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

  if (inv <= 0 || lucro <= 0) {

    out.innerHTML = `<p class="muted">Preenche pelo menos Montante a investir e Lucro desejado.</p>`;

    return;

  }

  const candidatos = ["AAPL","NVDA","AMD","MSFT","VUAA"].map((ticker, i) => {

    const tp1 = 100 + i \* 25; // preço base fictício

    const n   = inv / tp1;

    const tp2 = tp1 + (lucro / n);

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

    const override = cres > 0 ? ` | Cresc. ref: ${cres.toFixed(1)}%` : "";

    return { ticker, tp1, n, tp2, percent, note: override };

  });

  out.innerHTML = `

    <table class="table-like">

      <thead>

        <tr>

          <th>Ticker</th><th>TP1 (€)</th><th>Qtd</th><th>TP2 alvo (€)</th><th>Δ%</th>

        </tr>

      </thead>

      <tbody>

        ${candidatos.map(c => `

          <tr>

            <td>${c.ticker}</td>

            <td>${c.tp1.toFixed(2)}</td>

            <td>${c.n.toFixed(2)}</td>

            <td>${c.tp2.toFixed(2)}</td>

            <td>${c.percent.toFixed(1)}% ${c.note}</td>

          </tr>

        `).join("")}

      </tbody>

    </table>

    <p class="muted">Nota: isto é apenas uma simulação local. Podemos ligar à coleção <code>acoesDividendos</code> mais tarde para resultados reais.</p>

  `;

}

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

   EMAIL (mailto: resumo)

   ========================= \*/

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 (mock)

  document.getElementById("btnSimularTop10")?.addEventListener("click", simularTop10);

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

    ["inputInvestimento","inputLucro","inputCrescimento"].forEach(id => {

      const el = document.getElementById(id);

      if (el) el.value = "";

    });

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

    if (out) out.innerHTML = "";

  });

}

# JS | atividade.js

[ir para o topo](#_top)

// js/screens/atividade.js

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

import {

  collection, getDocs, query, orderBy, where

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

export async function initScreen() {

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

  if (!cont) return;

  cont.innerHTML = "A carregar…";

  try {

    // 1) Buscar TODAS as compras (ativos) e agrupar por ticker

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

    const snap = await getDocs(q);

    if (snap.empty) {

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

      return;

    }

    // Agrupador por TICKER

    const grupos = new Map(); // ticker -> { nome, setor, mercado, qtd, investido, objetivo, lastDate, anyObjSet }

    snap.forEach(doc => {

      const d = doc.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, // última compra

      };

      g.qtd += qtd;

      g.investido += invest;

      const obj = toNum(d.objetivoFinanceiro);

      // regra: considerar o primeiro objetivo > 0 por ticker (podes trocar para max, se preferires)

      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;

      // mantém nome/setor/mercado “mais recentes” caso venham diferentes

      g.nome = d.nome || g.nome;

      g.setor = d.setor || g.setor;

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

      grupos.set(ticker, g);

    });

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

    // 2) Ir buscar preços atuais + taxas a acoesDividendos (em chunks de 10 por causa do where "in")

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

    const infoMap = await fetchDividendInfoByTickers(tickers);

    // 3) Helpers de formatação

    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" });

    // 4) Construir linhas por TICKER

    const html = gruposArr.map(g => {

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

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

      // preço médio de compra

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

      // lucro atual (se tivermos preçoAtual; senão fica 0)

      const lucroAtual = (precoAtual !== null)

        ? (precoAtual - precoMedio) \* g.qtd

        : 0;

      // progresso vs objetivo (zero ao centro)

      let pctText = "—";

      let barHTML = "";

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

      if (objetivo > 0) {

        // progresso real em relação ao objetivo (pode ser negativo)

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

        // limitar entre -100 e 100

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

        // largura do lado a preencher: 0..50 (porque a barra tem duas metades)

        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>

        `;

      }

      // TP2 necessário (para atingir o objetivo de lucro por ticker)

      const tp2Necessario = (objetivo > 0 && g.qtd > 0)

        ? (precoMedio + (objetivo / g.qtd))

        : null;

      // Estimativa temporal para atingir TP2 (usa melhor taxa: 1m > 1s > 1ano)

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

      const estimativa = (tp2Necessario && precoAtual)

        ? estimateTime(precoAtual, tp2Necessario, taxa, periodLabel)

        : "—";

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

      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>

              `}

            </div>

          </div>

        </div>

      `;

    });

    cont.innerHTML = html.join("");

  } catch (e) {

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

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

  }

}

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

   Helpers

   ========================= \*/

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

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

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

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

// vai buscar info da coleção acoesDividendos por lotes de 10 tickers

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

    const snap = await getDocs(q);

    snap.forEach(doc => {

      const d = doc.data();

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

    });

  }

  return out;

}

// escolhe a melhor taxa disponível: 1m > 1s > 1ano

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 };

}

// estima nº de períodos até atingir targetPrice, com crescimento composto r (%) por período

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`;

}

# 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="btnNovaSimulacao">Nova Simulação</button>

      <button class="btn outline full" id="btnOportunidades">Analisar Oportunidades</button>

      <button class="btn ghost full">Adicionar Ação</button>

      <button class="btn ghost full">Definir Objetivo</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>

</div>

# HTML | atividade.html

[ir para o topo](#_top)

<div class="page" data-screen-title="Atividade Recente">

  <div class="dashboard-header">

    <p class="subtitle">Todas as operações realizadas</p>

  </div>

  <div id="listaAtividades"></div>

</div>

<script>

  if (typeof initScreen === "function") initScreen();

</script>

# 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">

          <input

            type="number"

            id="inputInvestimento"

            placeholder="Montante a investir (€) — Ex: 1500"

          />

          <input

            type="number"

            id="inputLucro"

            placeholder="Lucro desejado (€) — Ex: 100"

          />

          <input

            type="number"

            id="inputCrescimento"

            placeholder="Crescimento estimado (%) — opcional"

          />

        </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>