**FinanceAPP 2.0 | ficheiros conteúdo projecto**

Indices

[JS| Auth 2](#_Toc208242188)

[HTML | Auth 4](#_Toc208242189)

[JS | dashboard.js 5](#_Toc208242190)

[JS | simulador.js 17](#_Toc208242191)

[JS | atividade.js 27](#_Toc208242192)

[HTML | dasboard.html 32](#_Toc208242193)

[HTML | atividade.html 36](#_Toc208242194)

[HTML | Configurações 38](#_Toc208242195)

[HTML | Simulador.html 40](#_Toc208242196)

**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 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">APPFinance</div>

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

          <div class="subvalue">Acede ao teu portefólio e simulador.</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 | Configurações

[ir para o topo](#_top)

// js/screens/settings.js

// ⚠️ AJUSTA ESTE CAMINHO conforme a estrutura do teu projeto:

// - Se este ficheiro está em js/screens/, usa "../auth.js" (como abaixo).

// - Se estiver lado a lado com auth.js, usa "./auth.js".

import { doLogout } from "./auth.js";

const SETTINGS\_STORAGE\_KEY = "app.settings";

const defaultSettings = {

  // Notificações

  emailNotifications: true,

  pushNotifications: false,

  weeklyReports: true,

  // Segurança

  twoFactor: false,

  loginNotifications: true,

  // Interface

  darkMode: false,          // mantém compat com versões antigas

  language: "pt-PT",

  currency: "EUR",

};

/\* ---------------- helpers de storage ---------------- \*/

function loadSettings() {

  try {

    const raw = localStorage.getItem(SETTINGS\_STORAGE\_KEY);

    if (!raw) return { ...defaultSettings };

    const parsed = JSON.parse(raw);

    return { ...defaultSettings, ...parsed };

  } catch {

    return { ...defaultSettings };

  }

}

function saveSettings(s) {

  try {

    localStorage.setItem(SETTINGS\_STORAGE\_KEY, JSON.stringify(s));

  } catch {}

}

/\* ---------------- tema (light/dark) ---------------- \*/

function applyTheme(dark) {

  const mode = dark ? "dark" : "light";

  document.documentElement.setAttribute("data-theme", mode);

  // atualizar theme-color do mobile (opcional)

  const meta = document.querySelector('meta[name="theme-color"]');

  if (meta) meta.content = dark ? "#0b0e13" : "#ffffff";

  // notificar a app inteira (outros screens podem ouvir este evento)

  window.dispatchEvent(new CustomEvent("app:theme-changed", { detail: { dark } }));

}

/\* ---------------- init do screen ---------------- \*/

export function initScreen() {

  // ⚠️ NUNCA fazer querySelector/getElementById fora do initScreen,

  // porque o HTML do screen só existe depois da navegação injetar o markup.

  // Elementos

  const elLanguage = document.getElementById("cfgLanguage");

  const elCurrency = document.getElementById("cfgCurrency");

  const elDark     = document.getElementById("cfgDarkMode");

  const elEmailN = document.getElementById("cfgEmailNotifications");

  const elPush   = document.getElementById("cfgPushNotifications");

  const elWeekly = document.getElementById("cfgWeeklyReports");

  const el2FA   = document.getElementById("cfgTwoFactor");

  const elLogin = document.getElementById("cfgLoginNotifications");

  const btnSave   = document.getElementById("cfgSave");

  const btnCancel = document.getElementById("cfgCancel");

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

  if (!elLanguage || !elCurrency || !elDark || !btnSave || !btnCancel) {

    console.warn("⚠️ settings.js: elementos não encontrados. Confirma o HTML dos IDs.");

    return;

  }

  // 🔒 Logout — liga AQUI (agora o botão existe no DOM)

  if (btnLogout) {

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

      e.preventDefault();

      doLogout(); // exportado do auth.js

    });

  }

  // Carrega estado atual (se não houver, segue sistema e grava já)

  let state = loadSettings();

  if (!("darkMode" in state)) {

    state.darkMode =

      window.matchMedia &&

      window.matchMedia("(prefers-color-scheme: dark)").matches;

    saveSettings(state);

  }

  // Preenche UI

  elLanguage.value = state.language;

  elCurrency.value = state.currency;

  elDark.checked   = !!state.darkMode;

  if (elEmailN) elEmailN.checked = !!state.emailNotifications;

  if (elPush)   elPush.checked   = !!state.pushNotifications;

  if (elWeekly) elWeekly.checked = !!state.weeklyReports;

  if (el2FA)   el2FA.checked   = !!state.twoFactor;

  if (elLogin) elLogin.checked = !!state.loginNotifications;

  // Aplica tema no arranque deste screen

  applyTheme(!!state.darkMode);

  // Listeners (atualizam o estado em memória)

  elLanguage.addEventListener("change", () => {

    state.language = elLanguage.value;

  });

  elCurrency.addEventListener("change", () => {

    state.currency = elCurrency.value;

  });

  elDark.addEventListener("change", () => {

    state.darkMode = !!elDark.checked;

    applyTheme(state.darkMode); // aplica imediatamente

  });

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

    state.emailNotifications = !!elEmailN.checked;

  });

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

    state.pushNotifications = !!elPush.checked;

  });

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

    state.weeklyReports = !!elWeekly.checked;

  });

  el2FA?.addEventListener("change", () => {

    state.twoFactor = !!el2FA.checked;

  });

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

    state.loginNotifications = !!elLogin.checked;

  });

  // Botões

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

    saveSettings(state);

    // showToast?.("Configurações guardadas!");

  });

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

    // Recarrega do storage e volta a preencher/Aplicar

    state = loadSettings();

    elLanguage.value = state.language;

    elCurrency.value = state.currency;

    elDark.checked   = !!state.darkMode;

    if (elEmailN) elEmailN.checked = !!state.emailNotifications;

    if (elPush)   elPush.checked   = !!state.pushNotifications;

    if (elWeekly) elWeekly.checked = !!state.weeklyReports;

    if (el2FA)   el2FA.checked   = !!state.twoFactor;

    if (elLogin) elLogin.checked = !!state.loginNotifications;

    applyTheme(state.darkMode);

  });

  // (Opcional) Seguir alterações do sistema se o user nunca “forçou”

  try {

    const mq = window.matchMedia("(prefers-color-scheme: dark)");

    const handler = (e) => {

      // só respeita o sistema se o utilizador nunca mudou manualmente

      const saved = loadSettings();

      if (!("userForcedTheme" in saved) || !saved.userForcedTheme) {

        state.darkMode = e.matches;

        applyTheme(state.darkMode);

        saveSettings(state);

        elDark.checked = state.darkMode;

      }

    };

    // marca que o user escolheu manualmente quando mexer no switch

    elDark.addEventListener("change", () => {

      const s = loadSettings();

      s.userForcedTheme = true;

      saveSettings(s);

    });

    mq.addEventListener?.("change", handler);

  } catch {}

}

# JS | atividade.js

[ir para o topo](#_top)

// js/screens/atividade.js

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";

export async function initScreen() {

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

  if (!cont) return;

  cont.innerHTML = "A carregar…";

  try {

    // 1) Buscar movimentos 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;

    }

    const grupos = new Map();

    snap.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);

    });

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

    // 2) Preços atuais

    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" });

    // 3) Render

    const html = gruposArr

      .filter(g => g.qtd > 0) // não mostra posições encerradas

      .map(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;

        const lucroAtual = (precoAtual !== null)

          ? (precoAtual - precoMedio) \* g.qtd

          : 0;

        // barra zero ao centro (–100..+100 → 0..50% de cada lado)

        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";

        // ⚡️ Botões Compra/Venda (prefill do modal)

        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>

        `;

        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>

                `}

                ${actions}

              </div>

            </div>

          </div>

        `;

      });

    cont.innerHTML = html.join("");

    // 4) Ligações dos botões de ação rápida

    wireQuickActions(gruposArr);

  } catch (e) {

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

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

  }

}

/\* ===== Quick Actions (modal deste screen) ===== \*/

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");

  // abrir modal com prefill

  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(); });

  // delegação clicks

  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"));

  });

  // mudar label por tipo (se o user alternar)

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

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

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

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

  });

  // submit

  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);

      // Se for VENDA TOTAL: remove todos os docs do ticker (após registar a venda)

      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();

      // Recarrega o ecrã de portfólio

      location.reload();

    } catch (err) {

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

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

    }

  });

}

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

   Helpers

   ========================= \*/

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"); }

// busca info em acoesDividendos por lotes de 10

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(docu => {

      const d = docu.data();

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

    });

  }

  return out;

}

// 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

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

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

  <!-- 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 de Consumo">Bens de Consumo</option>

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

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

  <!-- Modal: Definir Objetivo -->

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

  <div class="modal-dialog modal-goal">

    <div class="modal-header">

      <h3>Definir Objetivo (seleção personalizada)</h3>

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

    </div>

    <!-- WIZARD -->

    <div class="goal-wizard">

      <!-- PASSO 1 · Filtros -->

      <section class="goal-step" data-step="1">

        <h4>Filtros</h4>

        <div class="form-grid">

          <input id="goalFiltroTicker"  type="search" placeholder="Ticker (ex.: AAPL)" />

          <input id="goalFiltroNome"    type="search" placeholder="Nome (contém...)" />

          <select id="goalFiltroSetor"></select>

          <select id="goalFiltroMercado"></select>

          <select id="goalFiltroMes"></select>

          <select id="goalFiltroPeriodicidade"></select>

        </div>

        <div class="actions-row" style="justify-content:flex-end">

          <button class="btn outline btn-black-shadow" id="goalBtnLimpar">Limpar</button>

          <button class="btn premium" id="goalNext1">Seguinte</button>

        </div>

      </section>

      <!-- PASSO 2 · Escolher ações -->

      <section class="goal-step" data-step="2" hidden>

        <div class="goal-columns-2">

          <div class="goal-list-wrap">

            <ul id="goalListaAcoes" class="goal-list"></ul>

          </div>

          <div class="goal-selected">

            <h4>Selecionadas</h4>

            <p class="muted">Ficam guardadas mesmo se fizeres novo filtro. Remove com o “x”.</p>

            <div id="goalSelecionadas" class="goal-tags"></div>

          </div>

        </div>

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

          <button class="btn outline" id="goalBack2">Voltar</button>

          <div>

            <button class="btn outline btn-black-shadow" id="goalBtnFiltrar">Aplicar filtros</button>

            <button class="btn premium" id="goalNext2">Seguinte</button>

          </div>

        </div>

      </section>

      <!-- PASSO 3 · Parâmetros e Simular -->

      <section class="goal-step" data-step="3" hidden>

        <h4>Parâmetros</h4>

        <div class="form-grid">

          <input type="number" id="goalInvest" placeholder="Montante a investir (€) – ex.: 1500" />

          <select id="goalPeriodo">

            <option value="1ano" selected>Taxa de Crescimento: 1 ano</option>

            <option value="1m">Taxa de Crescimento: 1 mês</option>

            <option value="1s">Taxa de Crescimento: 1 semana</option>

          </select>

          <input type="number" id="goalHorizonte" placeholder="Horizonte (nº períodos)" value="1" min="1" />

        </div>

        <div class="actions-row">

          <label class="chk"><input type="checkbox" id="goalAcoesCompletas" /> <span>Apenas ações completas</span></label>

          <label class="chk"><input type="checkbox" id="goalUsarTotal" checked /> <span>Usar todo o investimento</span></label>

        </div>

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

          <button class="btn outline" id="goalBack3">Voltar</button>

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

        </div>

      </section>

      <!-- PASSO 4 · Resultado “full” -->

      <section class="goal-step" data-step="4" hidden>

        <div class="goal-result">

          <div id="goalResultado"></div>

        </div>

      </section>

    </div>

  </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">Todas os ativos em carteira</p>

  </div>

  <div id="listaAtividades"></div>

</div>

<script>

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

</script>

<!-- Modal QUICK BUY/SELL (portfolio) -->

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

  <div class="modal-dialog">

    <div class="modal-header">

      <h3 id="pfAddTitle">Adicionar Ação</h3>

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

    </div>

    <form id="pfAddForm" class="page" style="padding: 14px">

      <div class="form-grid">

        <label>

          Tipo

          <select id="pfTipoAcao">

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

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

          </select>

        </label>

        <label>

          Ticker

          <input id="pfTicker" placeholder="ex.: AAPL" />

        </label>

        <label>

          Nome

          <input id="pfNome" placeholder="Nome do ativo" />

        </label>

        <label>

          Setor

          <select id="pfSetor">

            <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 de Consumo">Bens de Consumo</option>

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

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

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

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

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

          </select>

        </label>

        <label>

          Mercado

          <select id="pfMercado">

            <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="pfQuantidade" type="number" step="0.0001" />

        </label>

        <label id="pfLabelPreco">

          Preço de compra (€)

          <input id="pfPreco" type="number" step="0.0001" />

        </label>

        <label>

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

          <input id="pfObjetivo" type="number" step="0.01" />

        </label>

        <label id="pfVendaTotalWrap" style="display:none">

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

          Venda total (remove da carteira após registo)

        </label>

      </div>

      <div class="actions-row">

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

        <button type="button" class="btn outline" id="pfAddCancel">Cancelar</button>

      </div>

    </form>

  </div>

</div>

# HTML | Configurações

[ir para o topo](#_top)

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

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

  <div class="settings" style="max-width: 960px; margin: 0 auto; display: grid; gap: 16px;">

    <!-- PERFIL -->

    <div class="card">

      <div class="card-content" style="align-items:flex-start; gap:16px;">

        <div style="flex:1;">

          <p class="label">Perfil</p>

          <div class="form-grid">

            <label>

              Nome completo

              <input id="cfgName" type="text" placeholder="O teu nome" />

            </label>

            <label>

              Email (Firebase)

              <input id="cfgEmail" type="email" placeholder="o.teu@email" disabled />

            </label>

            <label>

              Telefone

              <input id="cfgPhone" type="tel" placeholder="+351 ..." />

            </label>

          </div>

          <p class="muted" style="margin-top:6px">O email vem da tua conta Firebase.</p>

        </div>

      </div>

    </div>

    <!-- SEGURANÇA -->

    <div class="card">

      <div class="card-content" style="align-items:flex-start; gap:16px;">

        <div style="flex:1;">

          <p class="label">Segurança</p>

          <div class="form-grid">

            <label style="display:flex; align-items:center; justify-content:space-between; gap:12px;">

              <span>Autenticação de dois fatores</span>

              <label class="switch">

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

                <span class="slider"></span>

              </label>

            </label>

            <label style="display:flex; align-items:center; justify-content:space-between; gap:12px;">

              <span>Notificações de login</span>

              <label class="switch">

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

                <span class="slider"></span>

              </label>

            </label>

          </div>

        </div>

      </div>

    </div>

    <!-- NOTIFICAÇÕES -->

    <div class="card">

      <div class="card-content" style="align-items:flex-start; gap:16px;">

        <div style="flex:1;">

          <p class="label">Notificações</p>

          <div class="form-grid">

            <label style="display:flex; align-items:center; justify-content:space-between; gap:12px;">

              <span>Email</span>

              <label class="switch">

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

                <span class="slider"></span>

              </label>

            </label>

            <label style="display:flex; align-items:center; justify-content:space-between; gap:12px;">

              <span>Push</span>

              <label class="switch">

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

                <span class="slider"></span>

              </label>

            </label>

            <label style="display:flex; align-items:center; justify-content:space-between; gap:12px;">

              <span>Relatórios semanais</span>

              <label class="switch">

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

                <span class="slider"></span>

              </label>

            </label>

          </div>

        </div>

      </div>

    </div>

    <!-- INTERFACE -->

    <div class="card">

      <div class="card-content" style="align-items:flex-start; gap:16px;">

        <div style="flex:1;">

          <p class="label">Interface</p>

          <div class="form-grid">

            <label>

              Idioma

              <select id="cfgLanguage">

                <option value="pt-PT">Português (PT)</option>

                <option value="en-US">English (US)</option>

                <option value="es-ES">Español</option>

              </select>

            </label>

            <label>

              Moeda

              <select id="cfgCurrency">

                <option value="EUR">Euro (€)</option>

                <option value="USD">Dólar ($)</option>

                <option value="BRL">Real (R$)</option>

              </select>

            </label>

            <label style="display:flex; align-items:center; justify-content:space-between; gap:12px;">

              <div>

                <div><strong>Tema escuro</strong></div>

                <div class="muted">Interface em modo escuro</div>

              </div>

              <label class="switch">

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

                <span class="slider"></span>

              </label>

            </label>

          </div>

        </div>

      </div>

    </div>

    <!-- PRIVACIDADE -->

    <div class="card">

      <div class="card-content" style="align-items:flex-start; gap:16px;">

        <div style="flex:1;">

          <p class="label">Privacidade</p>

          <div class="form-grid">

            <label style="display:flex; align-items:center; justify-content:space-between; gap:12px;">

              <span>Partilha de dados</span>

              <label class="switch">

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

                <span class="slider"></span>

              </label>

            </label>

            <label style="display:flex; align-items:center; justify-content:space-between; gap:12px;">

              <span>Analytics (anónimo)</span>

              <label class="switch">

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

                <span class="slider"></span>

              </label>

            </label>

          </div>

        </div>

      </div>

    </div>

    <!-- AÇÕES -->

    <div class="card">

      <div class="card-content" style="justify-content:flex-end; gap:8px;">

        <button class="btn outline" id="cfgCancel">Cancelar</button>

        <button class="btn premium" id="cfgSave">Guardar configurações</button>

        <button class="btn outline" id="btnLogout">Sair</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>