**FinanceAPP 2.0 | ficheiros conteúdo projecto**

***JS | dashboard.js***

// js/screens/dashboard.js

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

import {

  getDocs, collection, query, orderBy, limit

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

export async function initScreen() {

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

  // --- ELEMENTOS DA UI ---

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

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

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

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

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

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

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

  // --- 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(d.ticker.toUpperCase(), d.valorStock);

      }

    });

    // Agrupar ativos por TICKER (para KPIs por ativo agregado)

    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 (até decidires outra regra)

      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;

    // 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)}%`;

  } catch (err) {

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

  }

  // 2) Atividades recentes com barra de progresso

  await carregarAtividadeRecenteComProgresso();

}

// ------- Atividade recente com barra de progresso -------

async function carregarAtividadeRecenteComProgresso() {

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

  if (!cont) return;

  try {

    const q = query(

      collection(db, "ativos"),

      orderBy("dataCompra", "desc"),

      limit(8)

    );

    const snapAtivos = await getDocs(q);

    if (snapAtivos.empty) {

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

      return;

    }

    // Preços atuais

    const mapaPrecoAtual = {};

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

    snapAcoes.forEach(doc => {

      const d = doc.data();

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

        mapaPrecoAtual[d.ticker.toUpperCase()] = d.valorStock;

      }

    });

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

    const fmtDate = new Intl.DateTimeFormat("pt-PT",{ year:"numeric", month:"2-digit", day:"2-digit" });

    const html = [];

    snapAtivos.forEach(doc => {

      const d = doc.data();

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

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

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

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

      const objetivoLucro = Number(d.objetivoFinanceiro || 0);

      // Data

      let dataTxt = "sem data";

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

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

      }

      // Preço atual

      const precoAtual = typeof mapaPrecoAtual[ticker] === "number"

        ? mapaPrecoAtual[ticker]

        : precoCompra;

      // Lucro atual

      const investido   = precoCompra \* qtd;

      const atual       = precoAtual \* qtd;

      const lucroAtual  = atual - investido;

      // Progresso vs objetivo

      let pct = 0, widthPct = 0, pctText = "—";

      const positive = lucroAtual >= 0;

      if (objetivoLucro > 0) {

        pct     = (lucroAtual / objetivoLucro) \* 100;

        widthPct= Math.min(100, Math.abs(pct));

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

      }

      html.push(`

        <div class="activity-item">

          <div class="activity-left">

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

            <div>

              <p><strong>Compra - ${nome}</strong></p>

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

              <div class="activity-meta">

                <span>Objetivo (lucro): <strong>${fmtEUR.format(objetivoLucro || 0)}</strong></span>

                <span>${pctText}</span>

              </div>

              <div class="progress">

                <div class="progress-track">

                  <div class="progress-fill ${positive ? "positive" : "negative"}" style="width:${widthPct}%"></div>

                </div>

              </div>

            </div>

          </div>

          <span class="date">${dataTxt}</span>

        </div>

      `);

    });

    cont.innerHTML = html.join("");

  } catch (e) {

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

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

  }

}

**JS | atividade.js**

// js/screens/atividade.js

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

import {

  collection, getDocs, query, orderBy, where

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

export async function initScreen() {

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

  if (!cont) return;

  cont.innerHTML = "A carregar…";

  try {

    // 1) Busca as compras

    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;

    }

    // 2) Recolhe tickers únicos para ir buscar dados atuais (com 1 query por chunk)

    const ativos = [];

    const tickers = new Set();

    snap.forEach(doc => {

      const d = doc.data();

      d.\_\_id = doc.id;

      ativos.push(d);

      if (d.ticker) tickers.add(d.ticker);

    });

    // Helper: divide em chunks de 10 (limite do Firestore 'in')

    const toChunks = (arr, size=10) => {

      const out = [];

      for (let i = 0; i < arr.length; i += size) out.push(arr.slice(i, i+size));

      return out;

    };

    // 3) Mapa ticker->dados atuais a partir de acoesDividendos

    const tickersArr = Array.from(tickers);

    const chunks = toChunks(tickersArr, 10);

    const tickerInfo = new Map();

    for (const chunk of chunks) {

      const qPrices = query(

        collection(db, "acoesDividendos"),

        where("ticker", "in", chunk)

      );

      const s2 = await getDocs(qPrices);

      s2.forEach(doc => {

        const d = doc.data();

        if (d.ticker) tickerInfo.set(d.ticker, d);

      });

    }

    // 4) Formatters

    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" });

    // 5) Utilitário para estimar tempo (n períodos) com base na taxa

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

      // growthPct ex.: 4.5 (%/mês, ou %/semana, ou %/ano consoante a taxa)

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

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

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

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

      // devolve já com unidade correta

      if (periodLabel === "semana") {

        return `${n.toFixed(1)} semanas`;

      } else if (periodLabel === "mês") {

        return `${n.toFixed(1)} meses`;

      } else {

        return `${n.toFixed(1)} anos`;

      }

    }

    // 6) Constroi HTML

    const html = ativos.map(d => {

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

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

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

      const objetivo = Number(d.objetivoFinanceiro || 0);

      const tipoObj  = (d.tipoObjetivo || "lucro").toLowerCase();

      let dataTxt = "sem data";

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

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

      }

      // Dados atuais do ticker

      const info = d.ticker ? tickerInfo.get(d.ticker) : null;

      const precoAtual = Number(info?.valorStock || precoCompra);

      // Cálculos base

      const investido   = qtd \* precoCompra;

      const lucroAtual  = (precoAtual - precoCompra) \* qtd;

      // Progresso vs objetivo (só faz sentido para tipo "lucro" com objetivo > 0)

      let progresso = null;

      let tp2Necessario = null;

      let estimativaTempo = "—";

      if (tipoObj === "lucro" && objetivo > 0 && qtd > 0) {

        progresso = Math.max(0, Math.min(1, lucroAtual / objetivo)); // clamp 0..1

        tp2Necessario = precoCompra + (objetivo / Math.max(qtd, 1)); // P tal que (P - precoCompra)\*qtd = objetivo

        // Escolhe melhor taxa disponível e estima o tempo

        // prioridade: 1 mês > 1 semana > 1 ano

        let period = null;

        let taxa = null;

        if (typeof info?.taxaCrescimento\_1mes === "number") {

          taxa = info.taxaCrescimento\_1mes;

          period = "mês";

        } else if (typeof info?.taxaCrescimento\_1semana === "number") {

          taxa = info.taxaCrescimento\_1semana;

          period = "semana";

        } else if (typeof info?.taxaCrescimento\_1ano === "number") {

          taxa = info.taxaCrescimento\_1ano;

          period = "ano";

        }

        if (taxa !== null && taxa !== undefined) {

          estimativaTempo = estimateTime(precoAtual, tp2Necessario, taxa, period);

        }

      }

      const progressoPct = progresso !== null ? (progresso\*100).toFixed(0) : "—";

      return `

        <div class="activity-item">

          <div class="activity-left">

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

            <div>

              <p><strong>Compra - ${nome}</strong></p>

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

              <p class="muted">Investido: ${fmtEUR.format(investido)} · Preço atual: ${fmtEUR.format(precoAtual)}</p>

              <p class="muted">Lucro atual: <strong>${fmtEUR.format(lucroAtual)}</strong></p>

              ${tipoObj === "lucro" && objetivo > 0 ? `

                <div class="progress-row">

                  <span class="muted">Objetivo (lucro): ${fmtEUR.format(objetivo)}</span>

                  <div class="progress">

                    <div class="progress-bar" style="width:${progressoPct}%"></div>

                  </div>

                  <span class="muted small">${progressoPct}%</span>

                </div>

                <p class="muted">TP2 necessário: <strong>${fmtEUR.format(tp2Necessario)}</strong> · Estimativa: <strong>${estimativaTempo}</strong></p>

              ` : ``}

            </div>

          </div>

          <span class="date">${dataTxt}</span>

        </div>

      `;

    });

    cont.innerHTML = html.join("");

  } catch (e) {

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

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

  }

}

**HTML | dasboard.html**

<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</p>

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

        </div>

        <div class="icon-box">

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

        </div>

      </div>

    </div>

    <div class="card success">

      <div class="card-content">

        <div>

          <p class="label">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</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">Nova Simulação</button>

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

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

      <button class="btn ghost full">Analisar Progresso</button>

    </div>

  </div>

</div>

**HTML | atividade.html**

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

  <div class="dashboard-header">

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

  </div>

  <div id="listaAtividades"></div>

</div>

<script>

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

</script>

<div class="dashboard">

  <div class="dashboard-header">

    <h1>Dashboard</h1>

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

  </div>

  <div class="metrics-grid">

    <div class="card premium">

      <div class="card-content">

        <div>

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

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

        </div>

        <div class="icon-box">💰</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">📈</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">📊</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">🎯</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">🚀</div>

      </div>

    </div>

  </div>

  <div class="dashboard-columns">

    <div class="card glass">

      <h3>Atividade Recente</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">Nova Simulação</button>

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

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

      <button class="btn ghost full">Analisar Progresso</button>

    </div>

  </div>

</div>

**HTML | atividade.html**

<div class="page">

  <div class="dashboard-header">

    <h1>Lista de Atividades</h1>

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

  </div>

  <div id="listaAtividades"></div>

</div>

<script>

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

</script>