EveryFin - Dashboard com Gráficos (Documento Técnico e Negocial) Versão Detalhada e Aprimorada

---

Sumário

---

1. Visão Geral do Dashboard

O Dashboard do EveryFin é a porta de entrada para a saúde financeira do usuário, oferecendo uma visão consolidada e interativa das suas finanças. Ele é projetado para ser intuitivo e fornecer insights rápidos sobre entradas, saídas e saldo geral.

1.1. Objetivo

O objetivo principal do Dashboard é exibir de forma clara e concisa o saldo atual, o total de entradas e o total de saídas para um período selecionado. Além disso, ele visa apresentar a distribuição das categorias de transações através de gráficos visuais (barras ou pizza), facilitando a compreensão do comportamento financeiro do usuário.

1.2. Escopo e Valor de Negócio

- **Visão Financeira Centralizada**: A tela atua como um hub estratégico para análise financeira centralizada, permitindo que o usuário tenha uma visão rápida e clara de sua situação financeira no mês e um histórico financeiro.
- **Insights Rápidos**: Através de componentes visuais e interativos, o Dashboard permite uma análise gerencial eficiente, conectando diretamente com os dados financeiros do cliente.
- **Informação Acessível e Atrativa**: A utilização de gráficos torna a informação mais acessível, compreensível e visualmente atrativa, aumentando o valor percebido da solução e contribuindo para a fidelização do usuário.
- **Base para Tomada de Decisões**: Fornece os dados essenciais para que o usuário possa tomar decisões financeiras mais informadas.

2. Tecnologias e Ferramentas

A seleção de tecnologias para o Dashboard prioriza performance, interatividade, visualização de dados e uma experiência de desenvolvimento eficiente.

2.1. Framework e Componentização

- **React (v18)**: A base do frontend, proporcionando uma estrutura declarativa e eficiente para construir a interface do usuário. A componentização permite a criação de blocos reutilizáveis como cards e gráficos.
- **TypeScript**: Garante tipagem estática, aumentando a robustez do código e a produtividade no desenvolvimento.

2.2. Visualização de Dados

- **Recharts**: Uma biblioteca de gráficos construída sobre React e D3.js, ideal para criar gráficos interativos e responsivos como barras e pizza. Sua integração nativa com React simplifica a manipulação de dados e a customização visual.

2.3. Animação e Interatividade

- **Framer Motion**: Uma biblioteca de animação poderosa e declarativa para React, que será utilizada para adicionar animações suaves e comportamento interativo aos componentes do Dashboard, melhorando a percepção da experiência do usuário (UX).

2.4. Comunicação com o Backend

- **Axios**: Cliente HTTP baseado em Promises, utilizado para realizar as requisições à API do backend.
- **Interceptors Customizados**: Configurados no Axios para interceptar requisições e respostas, automaticamente adicionando tokens de autenticação (JWT) e tratando erros HTTP, como o redirecionamento para o login em caso de erro 401 (Unauthorized).

3. Componentes do Dashboard

O Dashboard será composto por módulos reutilizáveis, cada um com uma responsabilidade clara, facilitando a manutenção e a escalabilidade.

3.1. `CardResumo.tsx` (Cards de Resumo Financeiro)

- **Funcionalidade**: Componente reutilizável para exibir valores financeiros de forma destacada, como o "Saldo Total", "Total de Entradas" e "Total de Saídas".
- **Design**: Deverá ter um design limpo com sombra suave e bordas arredondadas, alinhado ao design system do EveryFin.
- **Dados**: Receberá props como `label` (título), `value` (valor monetário) e opcionalmente `trend` (indicador de tendência, ex: seta para cima/baixo, porcentagem).
- **Acessibilidade**: Terá `aria-label` apropriado para leitores de tela.

3.2. `GraficoResumo.tsx` (Gráficos Financeiros)

- **Funcionalidade**: Componente responsável por renderizar os gráficos de barras ou pizza que visualizam a distribuição das categorias de transações (receitas e despesas) para o período selecionado.
- **Tipos de Gráfico**: Suporte a gráficos de pizza (para proporção de categorias) e/ou barras (para comparar valores entre categorias).
- **Interatividade**: Hover effects em fatias da pizza ou barras do gráfico para exibir tooltips com detalhes (valor e porcentagem).
- **Responsividade**: Implementado com Recharts para garantir que os gráficos se adaptem a diferentes tamanhos de tela.
- **Dados**: Receberá um array de dados contendo categorias e seus respectivos valores.
- **Acessibilidade**: Contará com legendas visíveis e descrições de dados via `aria-label`.

3.3. `FiltrosDashboard.tsx` (Controles de Filtro)

- **Funcionalidade**: Componente que permitirá ao usuário selecionar o período de visualização dos dados (ex: "Mês Atual", "Últimos 30 dias", "Ano Atual", "Período Personalizado") e, futuramente, a "Empresa" (para cenários multi-empresa).
- **Controles**: Deverá conter dropdowns para seleção de período e um seletor de data para períodos personalizados.
- **Estado**: Gerenciará o estado dos filtros e comunicará as mudanças ao componente pai (Dashboard.tsx) para que este possa refetch dos dados da API.

3.4. `TransacoesRecentes.tsx` (Tabela de Transações Recentes)

- **Funcionalidade**: Uma seção adicional do Dashboard que exibe uma tabela compacta das transações mais recentes, permitindo que o usuário veja os últimos movimentos sem sair da tela principal.
- **Colunas**: Data, Descrição, Valor, Categoria, e talvez um ícone para Entrada/Saída.

- **Interatividade**: Cada linha pode ser clicável para direcionar o usuário aos detalhes da transação na tela de Entradas/Saídas.
- **Link "Ver Todas"**: Um link ou botão para navegar para a tela completa de "Entradas" ou "Saídas".

4. Integração com Backend (API de Dados)

O Dashboard dependerá de uma API robusta e otimizada para obter os dados financeiros de forma eficiente.

4.1. Endpoint Principal

- **GET `/dashboard/resumo`**: Este será o endpoint principal para buscar todos os dados necessários para popular o Dashboard.
- **Parâmetros**: O endpoint aceitará parâmetros de query para filtros de período (ex: `startDate`, `endDate`, `periodType`).

4.2. Retorno Esperado da API

O retorno da API para `/dashboard/resumo` será um objeto JSON contendo:

- `totalEntradas`: Valor total das receitas no período selecionado.
- `totalSaidas`: Valor total das despesas no período selecionado.
- `saldo`: Saldo líquido (totalEntradas - totalSaidas) no período.
- `categorias`: Um array de objetos, onde cada objeto representa uma categoria de transação (ex: `{ nome: 'Alimentação', valor: 500, tipo: 'saida' }`, `{ nome: 'Salário', valor: 2000, tipo: 'entrada' }`), para alimentar os gráficos.
- `historicoTransacoes`: Um array limitado das transações mais recentes para a seção de "Transações Recentes".

4.3. Estratégia de Comunicação

- Utilização do `useApi` (custom hook) que encapsula as chamadas `Axios`, tratamento de estados de carregamento e erro, e cache de dados (se necessário, com `react-query` ou `SWR` para otimização de performance).
- Envio automático do token JWT via interceptor do Axios em todas as requisições autenticadas.

4.4. Tratamento de Erros e Autenticação

- **Axios Interceptors**: Continuarão interceptando erros, especialmente o código `401 Unauthorized`, para redirecionar o usuário para a tela de login, garantindo que a sessão seja renovada ou encerrada corretamente.
- **Feedback Visual de Erro**: Em caso de falha na requisição da API, o Dashboard exibirá uma mensagem de erro clara para o usuário, indicando que não foi possível carregar os dados.

5. Experiência do Usuário (UX) e Responsividade

O Dashboard será projetado para ser intuitivo, agradável e funcional em qualquer dispositivo.

5.1. Design Responsivo

- O layout do Dashboard será totalmente adaptável para dispositivos móveis e desktops.
- Utilização das classes de responsividade do Tailwind CSS (breakpoints `sm`, `md`, `lg`, etc.) para ajustar o layout dos cards, gráficos e tabelas automaticamente.
- Em telas menores, os cards podem ser empilhados, os gráficos podem ocupar a largura total e a tabela de transações pode se tornar rolável horizontalmente ou ter colunas colapsáveis.

5.2. Animações e Transições

- **Framer Motion**: Implementação de animações leves e transições fluidas para:
  - Carregamento inicial dos componentes.
  - Mudanças de estado (ex: atualização de dados após um filtro).
  - Interações do usuário (ex: hover em elementos, expansão/colapso de seções).
- O objetivo é uma experiência "leve" e envolvente sem ser intrusiva.

5.3. Feedback Visual Completo

- **Estados de Carregamento (Loading States)**: Exibição de spinners ou placeholders (skeletons) enquanto os dados da API estão sendo carregados.
- **Mensagens de Erro**: Mensagens claras e amigáveis para o usuário em caso de falha no carregamento dos dados ou outros problemas.
- **Estados Vazios**: Se não houver dados para um período (ex: "Nenhuma transação encontrada para este mês"), exibir uma mensagem informativa em vez de um espaço em branco.

5.4. Carregamento Otimizado

- **Lazy Loading**: Ações como a importação de componentes de gráfico complexos podem ser adiadas até que sejam visíveis no viewport ou necessários, usando `React.lazy` e `Suspense`.
- **Otimização de Imagens**: Garantir que quaisquer ícones ou ilustrações no Dashboard sejam otimizados para a web.

6. Acessibilidade (A11y)

A acessibilidade será uma prioridade para garantir que o Dashboard seja utilizável por uma ampla gama de usuários, incluindo aqueles com deficiência.

6.1. Legendas e Descrições

- **Legendas Visíveis**: Todos os elementos visuais (gráficos, cards) terão legendas claras e visíveis para contextualizar a informação.
- **Descrição de Dados com `aria-label`**: Para usuários de leitores de tela, serão fornecidas descrições detalhadas e contextuais usando atributos `aria-label` ou `aria-describedby` para gráficos e elementos interativos.

6.2. Navegação por Teclado

- Suporte completo à navegação por teclado. Todos os elementos interativos (botões, filtros, links) serão acessíveis via tecla `Tab` e ativáveis com `Enter` ou `Espaço`.
- Foco visual claro nos elementos interativos (outline ou borda).

6.3. Cores e Contraste

- As cores utilizadas nos gráficos e texto seguirão as diretrizes de contraste da WCAG (Web Content Accessibility Guidelines) para garantir legibilidade para usuários com deficiência visual.

6.4. Leitura de Telas (Screen Readers)

- A estrutura semântica do HTML (ex: `<section>`, `<main>`, `<nav>`) e o uso de atributos ARIA contribuirão para uma experiência fluida para usuários de leitores de tela.

7. Testes Automatizados

Uma estratégia de testes robusta garantirá a qualidade, a estabilidade e a integridade do Dashboard.

7.1. Estratégia de Testes

- Testes Unitários: Foco em componentes isolados como `CardResumo.tsx` e `GraficoResumo.tsx`.
- Testes de Integração: Para o componente principal `Dashboard.tsx`, verificando a interação entre os subcomponentes, a chamada à API (mockada) e a renderização correta dos dados.
- Testes End-to-End (E2E): Com Cypress, para simular o fluxo completo do usuário no Dashboard, incluindo carregamento de dados, aplicação de filtros e navegação, garantindo que o sistema funcione como um todo.

7.2. Ferramentas de Teste

- **Jest**: Framework de testes para JavaScript.
- **React Testing Library (RTL)**: Para testes de componentes, focando no comportamento do usuário.
- **Cypress**: Para testes End-to-End.

7.3. Cobertura de Testes

- Meta de alta cobertura de testes (ex: 90%) para os componentes do Dashboard, especialmente para a lógica de filtros e a renderização dos gráficos.

8. CI/CD

O Dashboard será parte integrante do pipeline de Integração Contínua e Entrega Contínua (CI/CD) do frontend.

8.1. Build Validado por Testes

- O workflow de CI/CD (GitHub Actions) garantirá que o build do frontend só seja gerado e liberado para deployment se todos os testes automatizados (unitários, de integração e E2E) passarem com sucesso.

8.2. Deploy Automático

- Após a validação do build pelos testes, o deploy do frontend (incluindo o Dashboard) para ambientes de staging e produção será automático, utilizando plataformas como Vercel (para branch `main`) e preview deploys para cada Pull Request.

8.3. Pipeline de Qualidade

- O pipeline de CI/CD também incluirá etapas de linting, formatação de código e verificação de vulnerabilidades em dependências, garantindo que o código do Dashboard mantenha um alto padrão de qualidade.

9. Proposta Negocial Aprimorada

A implementação do Dashboard é um investimento estratégico que agrega valor significativo à solução EveryFin.

9.1. Valor Estratégico da Tela

- **Análise Financeira Centralizada**: Oferece uma visão unificada e em tempo real da saúde financeira, economizando tempo e esforço do usuário na coleta de informações dispersas.
- **Visão Clara do Mês e Histórico**: Permite um acompanhamento preciso das entradas e saídas correntes e a análise do comportamento financeiro ao longo do tempo.
- **Conexão Direta com Dados do Cliente**: Garante que os insights sejam baseados em dados reais e atualizados, aumentando a confiabilidade da informação.

9.2. Diferenciais e Benefícios

- **Gráficos Acessíveis e Atrativos**: A visualização de dados através de gráficos intuitivos torna a informação complexa mais fácil de digerir e aumenta o engajamento do usuário.
- **Aumento do Valor Percebido da Solução**: Um Dashboard bem projetado e funcional é um diferencial competitivo, posicionando o EveryFin como uma ferramenta sofisticada e completa para gestão financeira.
- **Contribuição para Fidelização**: Uma experiência de usuário superior, impulsionada por um Dashboard útil e agradável, fortalece a relação do usuário com a plataforma, incentivando o uso contínuo e a lealdade.
- **Facilitação da Tomada de Decisão**: Empodera o usuário com informações claras e rápidas, auxiliando-o a identificar tendências, controlar gastos e planejar finanças de forma mais eficaz.

10. Prompt Reutilizável

Este prompt pode ser utilizado para solicitar a criação de telas de dashboard financeiras semelhantes em outros contextos.

"Olá, [Nome do Modelo]!

Crie uma tela de dashboard financeira completa com os seguintes requisitos:

1. **Resumo de Entradas, Saídas e Saldo**: Componentes visuais destacando esses valores.
2. **Gráficos Financeiros**: Gráfico de barras ou pizza para visualizar a distribuição de categorias (receita/despesa).
3. **Interatividade e Animações**: Componentes reativos e animados (React + Tailwind + Recharts + Framer Motion).
4. **Integração de Dados**: Dados vindos de uma API backend, com autenticação JWT e tratamento de erros via Axios interceptors.
5. **Design e Responsividade**: Layout totalmente responsivo (mobile e desktop) e uso de design tokens.
6. **Acessibilidade**: Legendas visíveis, `aria-label`, e suporte a navegação por teclado.
7. **Testes Automatizados**: Inclusão de testes unitários (Jest/RTL) e testes E2E (Cypress).
8. **CI/CD**: Integração com CI/CD (build validado por testes, deploy automático para Vercel).
9. **Proposta Negocial**: Valor agregado, cronograma e investimento.

Finalize com um documento técnico-comercial detalhado em PDF, incluindo visão geral, tecnologias, componentes, integração, UX/A11y, testes, CI/CD e a proposta de valor.

Documento expandido para excelência máxima e alinhamento com padrões UX e de negócios."

Fontes

Criar Resumo em Áudio