Clean Code – Entrega 2

Alunos: Gabriel Angelo e Jefferson Barzan

1. ANÁLISE DOS PRINCIPAIS PROBLEMAS IDENTIFICADOS

1.1 Estrutura de Roteamento Inconsistente

Local: App.jsx
Problema: Foi identificado roteamento duplicado com caminhos “/” e “” sem clara distinção entre eles, o que pode causar conflitos de navegação e comportamentos inesperados. Essa inconsistência prejudica a manutenibilidade e a clareza do fluxo de navegação.

1.2 Gestão de Estado Inadequada

Local: BookForm.jsx
Problema: Foi detectado uso incorreto do hook useState, com reatribuição direta de variáveis. Essa prática viola as regras do React, pois o estado deve ser atualizado apenas por meio da função *setter*. Isso faz com que a interface não reaja adequadamente às mudanças, gerando inconsistências para o usuário.

1.3 Arquitetura HTML Desatualizada e Fragmentada

Locais: login.html, mainMenuColaborador.html, novaSenha.html, recSenha.html, usuarios.html, livros.html
Problema: A aplicação utiliza múltiplos arquivos HTML estáticos, o que representa uma abordagem obsoleta. Essa fragmentação causa duplicação de código, dificuldade de manutenção e inconsistência visual entre páginas.

1.4 Tipografia e Estilos Inconsistentes

Locais: app.css, styleSenha.css e múltiplos arquivos HTML
Problema: Há o uso simultâneo de Tailwind CSS, CSS personalizado e estilos inline. Essa mistura gera conflitos, sobreposição de regras e inconsistência visual. A falta de padronização compromete a identidade e a performance da aplicação.

1.5 Lógica de Negócio nos Componentes

Problema: Há dados *hardcoded* e lógica de manipulação de estado espalhada por diversos arquivos, incluindo scripts embutidos em HTML. Isso causa baixo reaproveitamento de código, dificulta testes e cria forte acoplamento entre partes da aplicação.

1.6 Problemas de Acessibilidade e Semântica

Problema: Foram usadas tags <a> para ações que deveriam ser botões, prejudicando a acessibilidade e o SEO. Além disso, há má utilização de elementos semânticos, dificultando a navegação por leitores de tela.

1.7 Nomenclatura Inconsistente

Problema: Existe mistura de português e inglês em nomes de variáveis e componentes, como “BookList” e “ListaCompras”. Essa falta de padrão compromete a legibilidade e a colaboração entre desenvolvedores.

1.8 Código Duplicado e Estruturas Repetitivas

Locais: BookList.jsx e diversos HTMLs
Problema: Há duplicação de tabelas, formulários e estruturas semelhantes em várias partes da aplicação. Isso aumenta a chance de erros e dificulta evoluções futuras.

1.9 Falta de Componentização e Reutilização

Problema: As páginas HTML são blocos monolíticos, sem reutilização de headers, footers ou formulários. Essa ausência de componentização gera duplicação de código e dificulta mudanças globais.

1.10 Gestão de Estado Primitiva

Locais: usuarios.html, livros.html
Problema: A aplicação utiliza arrays simples para gerenciar dados, sem persistência ou sincronização entre componentes. Isso é insuficiente para aplicações reais e escaláveis.

2. ESTRATÉGIA COMPREENSIVA DE REFATORIAÇÃO

2.1 Ferramentas

- **ESLint + Prettier:** padronização e formatação automática do código.
- **React Hook Form:** simplificação da manipulação e validação de formulários.
- **Styled Components ou CSS Modules:** unificação da estratégia de estilização.

2.2 Plano de Ação em Três Fases

Fase 1 – Configuração do Ambiente e Migração da Arquitetura

Criação da estrutura React e substituição dos arquivos HTML estáticos por componentes reutilizáveis.

Comandos de configuração inicial:

```
npm install -D eslint prettier @typescript-eslint/parser
npm install react-hook-form zustand
npm install react-router-dom
```

Fase 2 – Correções Estruturais Imediatas

- **Roteamento:** eliminar caminhos duplicados e estruturar navegação hierárquica.
- **BookForm:** corrigir manipulação incorreta de estado.
- **Componentização:** extrair formulários, tabelas e botões em componentes reutilizáveis.

Fase 3 – Padronização e Otimização

- **Nomenclatura:** padronização total em inglês.
- **Biblioteca de Componentes:** criação de elementos reutilizáveis (botões, formulários, modais).

3. MIGRAÇÃO DOS ARQUIVOS HTML PARA REACT

3.1 Estrutura Proposta de Componentes

Auth Components

- LoginForm
- PasswordRecovery
- NewPasswordForm

Layout Components

- MainLayout
- NavigationMenu
- Header

Feature Components

- UserManagement
- BookManagement
- DataTable
- FormModal

3.2 Benefícios da Migração

- Maior reutilização de código e componentização.
- Gestão de estado robusta.
- Navegação dinâmica via *client-side routing*.
- Melhor performance e experiência de desenvolvimento (hot-reload).