📄 Guia Completo - Sistema de Exportação de PDFs

📄 Guia Completo - Sistema de Exportação de PDFs

Este guia explica como usar o sistema de geração de PDFs implementado no projeto usando `@react-pdf/renderer`.

📋 Índice

🎯 Visão Geral

O sistema de PDFs foi criado para facilitar a geração de documentos PDF consistentes em toda a aplicação. Ele utiliza componentes reutilizáveis para Header, Footer e Tabelas, garantindo um padrão visual uniforme.

Tecnologias Utilizadas

- **@react-pdf/renderer**: Biblioteca para criar PDFs usando componentes React
- **TypeScript**: Tipagem forte para segurança
- **React Hooks**: Para gerenciamento de estado e lógica

📁 Estrutura de Componentes

```
app/components/pdf/
├── common/                    # Componentes reutilizáveis
│   ├── PDFHeader.tsx         # Cabeçalho com logo e informações
│   ├── PDFFooter.tsx         # Rodapé com paginação
│   ├── PDFTable.tsx          # Tabela reutilizável
│   └── index.ts              # Exports
├── templates/                 # Templates específicos de cada CRUD
│   ├── DriversPDFTemplate.tsx
│   ├── TrucksPDFTemplate.tsx
│   └── index.ts
├── utils/                     # Utilitários
│   ├── pdfStyles.ts          # Estilos compartilhados
│   └── usePDFImage.ts        # Hook para converter imagem para base64
└── index.ts                   # Exports principais
```

🧩 Componentes Reutilizáveis

PDFHeader

Cabeçalho padrão com logo, título, subtítulo e data.

```
import { PDFHeader } from "@/app/components/pdf/common"
;<PDFHeader
  title="Relatório de Motoristas"
  subtitle="Listagem completa de motoristas cadastrados"
  logoBase64={logoBase64} // Imagem em base64
  date={new Date()}
/>
```

Props:

- `title` (string, obrigatório): Título principal
- `subtitle` (string, opcional): Subtítulo
- `logoBase64` (string, opcional): Logo em formato base64
- `date` (Date, opcional): Data/hora (padrão: new Date())

PDFFooter

Rodapé padrão com informações da empresa e numeração de páginas.

```
import { PDFFooter } from "@/app/components/pdf/common"
;<PDFFooter company="IAGRO - Sistema de Controle Industrial" version="0.
```

Props:

- `company` (string, opcional): Nome da empresa
- `version` (string, opcional): Versão do sistema

PDFTable

Tabela reutilizável com suporte a múltiplas colunas.

```
import { PDFTable } from "@/app/components/pdf/common"
;<PDFTable
  headers={["Coluna 1", "Coluna 2", "Coluna 3"]}
  data={[
    ["Valor 1", "Valor 2", "Valor 3"],
    ["Valor 4", "Valor 5", "Valor 6"],
  ]}
  columnWidths={[33, 33, 34]} // Em porcentagem
/>
```

Props:

- `headers` (string[]): Array com nomes das colunas
- `data` (string[][]): Array de arrays com os dados das linhas
- `columnWidths` (number[], opcional): Larguras das colunas em porcentagem

---

🚀 Como Criar um Novo Template PDF

Passo 1: Criar o Template

Crie um novo arquivo em `app/components/pdf/templates/`:

```tsx
// app/components/pdf/templates/MeuCRUDPDFTemplate.tsx
import { Document, Page, View, Text, StyleSheet } from "@react-pdf/ren
import { PDFHeader, PDFFooter, PDFTable } from "../common"
import { MeuTipo } from "@/app/types/meu-tipo"
import { pdfStyles } from "../utils/pdfStyles"

interface MeuCRUDPDFTemplateProps {
  items: MeuTipo[]
  logoBase64?: string
  sortConfig?: {
    key: keyof MeuTipo
    direction: "asc" | "desc"
  } | null
  searchTerm?: string
}

export const MeuCRUDPDFTemplate = ({
  items,
  logoBase64,
  sortConfig,
  searchTerm = "",
}: MeuCRUDPDFTemplateProps) => {
  const styles = StyleSheet.create({
    page: pdfStyles.page,
    summary: {
      marginBottom: 15,
      padding: 10,
      backgroundColor: "#f9f9f9",
      borderRadius: 4,
    },
    summaryText: {
      fontSize: 9,
      color: "#666",
    },
    summaryLine: {
      marginBottom: 2,
    },
  })

  // Aplicar ordenação se houver
  const sortedItems = sortConfig
    ? [...items].sort((a, b) => {
        const aValue = a[sortConfig.key]
        const bValue = b[sortConfig.key]

        if (!aValue && !bValue) return 0
        if (!aValue) return 1
        if (!bValue) return -1

        if (aValue < bValue) {
          return sortConfig.direction === "asc" ? -1 : 1
        }
        if (aValue > bValue) {
          return sortConfig.direction === "asc" ? 1 : -1
        }
        return 0
      })
    : items

  // Preparar dados para a tabela
```

```
  const tableData = sortedItems.map((item) => [
    item.campo1 || "-",
    item.campo2 || "-",
    item.campo3 || "-",
  ])

  const headers = ["Campo 1", "Campo 2", "Campo 3"]
  const columnWidths = [33, 33, 34]

  // Estatísticas
  const total = sortedItems.length
  const ativos = sortedItems.filter((i) => i.status === "ATIVO").lengt
  const inativos = sortedItems.filter((i) => i.status === "INATIVO").l

  // Informação de ordenação
  const getSortInfo = () => {
    if (!sortConfig) return null
    const columnNames: Record<keyof MeuTipo, string> = {
      id: "ID",
      campo1: "Campo 1",
      campo2: "Campo 2",
      // ... mapear todos os campos
    }
    const directionText =
      sortConfig.direction === "asc" ? "Crescente" : "Decrescente"
    return `Ordenado por: ${columnNames[sortConfig.key]} (${directionT
  }

  return (
    <Document>
      <Page size="A4" style={styles.page}>
        <PDFHeader
          title="Relatório de Meu CRUD"
          subtitle="Listagem completa"
          logoBase64={logoBase64}
        />

        <View style={styles.summary}>
          <Text style={[styles.summaryText, styles.summaryLine]}>
            Total: {total} | Ativos: {ativos} | Inativos: {inativos}
          </Text>
          {searchTerm && (
            <Text style={[styles.summaryText, styles.summaryLine]}>
              Filtro de busca: "{searchTerm}"
            </Text>
          )}
          {getSortInfo() && (
            <Text style={styles.summaryText}>{getSortInfo()}</Text>
          )}
        </View>

        <PDFTable
          headers={headers}
          data={tableData}
          columnWidths={columnWidths}
        />

        <PDFFooter />
      </Page>
    </Document>
  )
}
```

Passo 2: Exportar no index.ts

```
// app/components/pdf/templates/index.ts
```

```
export { DriversPDFTemplate } from "./DriversPDFTemplate"
export { TrucksPDFTemplate } from "./TrucksPDFTemplate"
export { MeuCRUDPDFTemplate } from "./MeuCRUDPDFTemplate"
```

Passo 3: Criar o Dialog de Visualização

```
// app/components/MeuCRUD/MeuCRUDPDFViewDialog.tsx
"use client"

import { MeuCRUDPDFTemplate } from "@/app/components/pdf/templates/Meu
import { usePDFImage } from "@/app/components/pdf/utils/usePDFImage"
import { usePDFExport } from "@/app/hooks/usePDFExport"
import { MeuTipo } from "@/app/types/meu-tipo"
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
} from "@/app/components/ui/dialog"
import { Button } from "@/app/components/ui/button"
import { Loader2, Download, FileText } from "lucide-react"
import { BlobProvider } from "@react-pdf/renderer"

interface MeuCRUDPDFViewDialogProps {
  open: boolean
  onOpenChange: (open: boolean) => void
  items: MeuTipo[]
  sortConfig?: {
    key: keyof MeuTipo
    direction: "asc" | "desc"
  } | null
  searchTerm?: string
}

export default function MeuCRUDPDFViewDialog({
  open,
  onOpenChange,
  items,
  sortConfig,
  searchTerm = "",
}: MeuCRUDPDFViewDialogProps) {
  const { imageSrc: logoBase64, loading: logoLoading } =
    usePDFImage("/iagro-logo.png")
  const { exportToPDF, isExporting } = usePDFExport()

  const handleExport = () => {
    if (!logoBase64) {
      setTimeout(() => {
        if (logoBase64) {
          exportToPDF(
            <MeuCRUDPDFTemplate
              items={items}
              logoBase64={logoBase64}
              sortConfig={sortConfig}
              searchTerm={searchTerm}
            />,
            {
              filename: `meu-crud-${new Date().toISOString().split("T"
            },
          )
        }
      }, 500)
      return
    }
```

```
61
62      exportToPDF(
63        <MeuCRUDPDFTemplate
64          items={items}
65          logoBase64={logoBase64}
66          sortConfig={sortConfig}
67          searchTerm={searchTerm}
68        />,
69        { filename: `meu-crud-${new Date().toISOString().split("T")[0]}.
70      )
71    }
72
73    if (!items || items.length === 0) {
74      return (
75        <Dialog open={open} onOpenChange={onOpenChange}>
76          <DialogContent>
77            <DialogHeader>
78              <DialogTitle>Visualizar PDF</DialogTitle>
79              <DialogDescription>
80                Não há itens para exibir no PDF.
81              </DialogDescription>
82            </DialogHeader>
83            <DialogFooter>
84              <Button onClick={() => onOpenChange(false)}>Fechar</Button
85            </DialogFooter>
86          </DialogContent>
87        </Dialog>
88      )
89    }
90
91    return (
92      <Dialog open={open} onOpenChange={onOpenChange}>
93        <DialogContent className="max-h-[90vh] max-w-[90vw] sm:max-w-[90
94          <DialogHeader>
95            <DialogTitle className="flex items-center gap-2">
96              <FileText className="h-5 w-5" />
97              Visualizar PDF - Meu CRUD
98            </DialogTitle>
99            <DialogDescription>
100             Visualize o documento PDF antes de exportar. Total: {items
101           </DialogDescription>
102         </DialogHeader>
103
104         <div className="flex-1 overflow-auto rounded-lg border">
105           {logoLoading ? (
106             <div className="flex h-96 items-center justify-center">
107               <Loader2 className="text-primary h-8 w-8 animate-spin" /
108               <span className="ml-2">Carregando PDF...</span>
109             </div>
110           ) : (
111             <BlobProvider
112               document={
113                 <MeuCRUDPDFTemplate
114                   items={items}
115                   logoBase64={logoBase64}
116                   sortConfig={sortConfig}
117                   searchTerm={searchTerm}
118                 />
119               }
120             >
121               {({ blob, url, loading, error }) => {
122                 if (loading) {
123                   return (
124                     <div className="flex h-96 items-center justify-cen
125                       <Loader2 className="text-primary h-8 w-8 animate
126                       <span className="ml-2">Gerando preview...</span>
127                     </div>
```

```
                  )
                }

                if (error) {
                  return (
                    <div className="text-destructive flex h-96 items-c
                      <span>Erro ao gerar preview: {error.message}</sp
                    </div>
                  )
                }

                return (
                  <iframe
                    src={url || undefined}
                    className="h-[600px] w-full border-0"
                    title="PDF Preview"
                  />
                )
              }}
            </BlobProvider>
          )}
        </div>

        <DialogFooter>
          <Button variant="outline" onClick={() => onOpenChange(false)
            Fechar
          </Button>
          <Button
            onClick={handleExport}
            disabled={isExporting || logoLoading || !logoBase64}
          >
            {isExporting ? (
              <>
                <Loader2 className="mr-2 h-4 w-4 animate-spin" />
                Exportando...
              </>
            ) : (
              <>
                <Download className="mr-2 h-4 w-4" />
                Exportar PDF
              </>
            )}
          </Button>
        </DialogFooter>
      </DialogContent>
    </Dialog>
  )
}
```

🔗 Integração com CRUD

Passo 1: Adicionar Estados no Componente Principal

```
// app/components/MeuCRUD/MeuCRUD.tsx
import { useState } from "react"
import MeuCRUDPDFViewDialog from "./MeuCRUDPDFViewDialog"

export default function MeuCRUD({ searchTerm = "", statusFilter = "todo
  const [showPDFDialog, setShowPDFDialog] = useState(false)
  const [sortConfig, setSortConfig] = useState<{
    key: keyof MeuTipo
    direction: "asc" | "desc"
  } | null>(null)

```

```
  // ... resto do código

  return (
    <div>
      {/* Seus componentes */}

      <MeuCRUDPDFViewDialog
        open={showPDFDialog}
        onOpenChange={setShowPDFDialog}
        items={effectiveList}
        sortConfig={sortConfig}
        searchTerm={searchTerm}
      />
    </div>
  )
}
```

Passo 2: Adicionar no ActionButtons

```
// app/components/MeuCRUD/MeuCRUDActionButtons.tsx
interface MeuCRUDActionButtonsProps {
  handleExport: (tipo: string, formato: "csv" | "pdf") => void
  onExportPDF?: () => void
}

export default function MeuCRUDActionButtons({
  handleExport,
  onExportPDF,
}: MeuCRUDActionButtonsProps) {
  return (
    <DropdownMenu>
      <DropdownMenuTrigger asChild>
        <Button variant="outline">
          <Download className="mr-2 h-4 w-4" />
          Exportar
        </Button>
      </DropdownMenuTrigger>
      <DropdownMenuContent>
        <DropdownMenuItem onClick={() => handleExport("MeuCRUD", "csv")}
          Exportar CSV
        </DropdownMenuItem>
        <DropdownMenuItem
          onClick={onExportPDF || (() => handleExport("MeuCRUD", "pdf")}
        >
          Exportar PDF
        </DropdownMenuItem>
      </DropdownMenuContent>
    </DropdownMenu>
  )
}
```

Passo 3: Conectar com a Tabela (se precisar de ordenação)

```
// Na sua tabela
<Tabela
  items={items}
  onSortChange={setSortConfig} // Para capturar a ordenação
/>
```

📚 Exemplos Práticos

Exemplo 1: Drivers (Motoristas)

Arquivo: `app/components/pdf/templates/DriversPDFTemplate.tsx`

Características:

- ✅ Header com logo IAGRO
- ✅ Tabela com: Matrícula, Nome, Contato, CNH, Expiração CNH, Status
- ✅ Suporte a ordenação e filtro de busca
- ✅ Estatísticas: Total, Ativos, Inativos

Uso:

```
import { DriversPDFTemplate } from "@/app/components/pdf/templates/Drive
;<DriversPDFTemplate
  drivers={drivers}
  logoBase64={logoBase64}
  sortConfig={sortConfig}
  searchTerm={searchTerm}
/>
```

Exemplo 2: Trucks (Caminhões)

Arquivo: `app/components/pdf/templates/TrucksPDFTemplate.tsx`

Características:

- ✅ Header com logo IAGRO
- ✅ Tabela com: Placa, Patrimônio, Volume, Modelo, Marca, Ano, Status
- ✅ Suporte a ordenação e filtro de busca
- ✅ Estatísticas: Total, Ativos, Inativos

Uso:

```
import { TrucksPDFTemplate } from "@/app/components/pdf/templates/Trucks
;<TrucksPDFTemplate
  trucks={trucks}
  logoBase64={logoBase64}
  sortConfig={sortConfig}
  searchTerm={searchTerm}
/>
```

🎨 Personalização de Estilos

Os estilos padrão estão em `app/components/pdf/utils/pdfStyles.ts`. Você pode:

1. **Modificar estilos existentes**: Edite o arquivo `pdfStyles.ts`
2. **Adicionar estilos específicos**: Use `StyleSheet.create()` no seu template

```
const styles = StyleSheet.create({
```

```
  page: pdfStyles.page, // Herda estilos base
  customSection: {
    marginTop: 20,
    padding: 15,
    backgroundColor: "#e3f2fd",
  },
})
```

📝 Dicas e Boas Práticas

✅ Faça

- ✅ Sempre use os componentes `PDFHeader` e `PDFFooter` para manter consistência
- ✅ Converta imagens para base64 usando `usePDFImage` hook
- ✅ Inclua informações de ordenação e busca no resumo do PDF
- ✅ Use o hook `usePDFExport` para gerenciar a exportação
- ✅ Passe `sortConfig` e `searchTerm` para refletir o estado atual da tabela

❌ Evite

- ❌ Criar novos componentes de Header/Footer para cada CRUD
- ❌ Usar URLs de imagens diretamente (sempre converta para base64)
- ❌ Esquecer de passar filtros e ordenação para o PDF
- ❌ Criar estilos inline muito complexos (prefira StyleSheet)

🔧 Troubleshooting

Imagem não aparece no PDF

Problema: Logo não é exibida no PDF gerado.

Solução: Certifique-se de usar o hook `usePDFImage` para converter a imagem para base64:

```
const { imageSrc: logoBase64, loading: logoLoading } =
  usePDFImage("/iagro-logo.png")
```

Ordenação não reflete no PDF

Problema: PDF não está ordenado como a tabela.

Solução: Certifique-se de passar o `sortConfig` da tabela para o template:

```
// Na tabela
const handleSort = (key: keyof MeuTipo) => {
  const newSortConfig = { key, direction: "asc" }
  setSortConfig(newSortConfig)
  onSortChange?.(newSortConfig) // Passa para o componente pai
}

// No componente principal
```

```
;<MeuCRUDPDFViewDialog
  sortConfig={sortConfig} // Passa para o PDF
/>
```

Dialog não abre em tela cheia

Problema: Dialog de visualização não usa toda a largura disponível.

Solução: Use as classes corretas:

```
<DialogContent className="max-h-[90vh] max-w-[90vw] sm:max-w-[90vw]">
```

📦 Dependências Necessárias

Certifique-se de ter instalado:

```
{
  "@react-pdf/renderer": "^4.3.1"
}
```

Para instalar:

```
pnpm add @react-pdf/renderer
```

📞 Suporte

Para dúvidas ou problemas:

1. Consulte os exemplos em `DriversPDFTemplate.tsx` e `TrucksPDFTemplate.tsx`
2. Verifique a documentação oficial: react-pdf.org
3. Analise os hooks: `usePDFExport.ts` e `usePDFImage.ts`