Pdfmake

O que é a lib?

Pdfmake é uma biblioteca de gerar pdfs seja no servidor ou cliente ela permite com que a gente gere alguns tipos de conteúdo como um relatório. Dentro do documentão ela dá vários exemplos de como usar essa lib com estilos diferentes com imagens por exemplo, podemos trabalhar com várias gamas de fontes, a gente consegue trabalhar também com colunas divisões de colunas, com tabelas e se quiser trabalhar com texto com uma margem e centralizado é bem possível também.

Como funciona a lib:

Na parte de instalação do pdfmake é bem simples hoje utilizamos duas formas padrão que seria:

npm install pdfmake

```
npm WARN saveError ENOENT: no such file or directory, open 'D:\ReactTreinamento\ktPDFmakes\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'D:\ReactTreinamento\ktPDFmakes\package.json'
npm WARN ktPDFmakes No description
npm WARN ktPDFmakes No repository field.
npm WARN ktPDFmakes No README data
npm WARN ktPDFmakes No license field.

+ pdfmake@0.2.5
added 110 packages from 110 contributors and audited 110 packages in 23.767s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Ou yarn add pdfmake

```
D:\ReactTreinamento\ktPDFmakes2>Yarn add pdfmake
yarn add v1.22.19
info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
success Saved 87 new dependencies.
info Direct dependencies
└─ pdfmake@0.2.5
```

Essas seriam as duas formas de instalação com sucesso mostrado no print do terminal. Agora para importar o PDFmake na aplicação começando pela parte do cliente fazemos da seguinte maneira:

```
import pdfMake from "pdfmake/build/pdfmake";
import pdfFonts from "pdfmake/build/vfs_fonts";
```

Agora já na parte do servidor fazemos da seguinte forma:

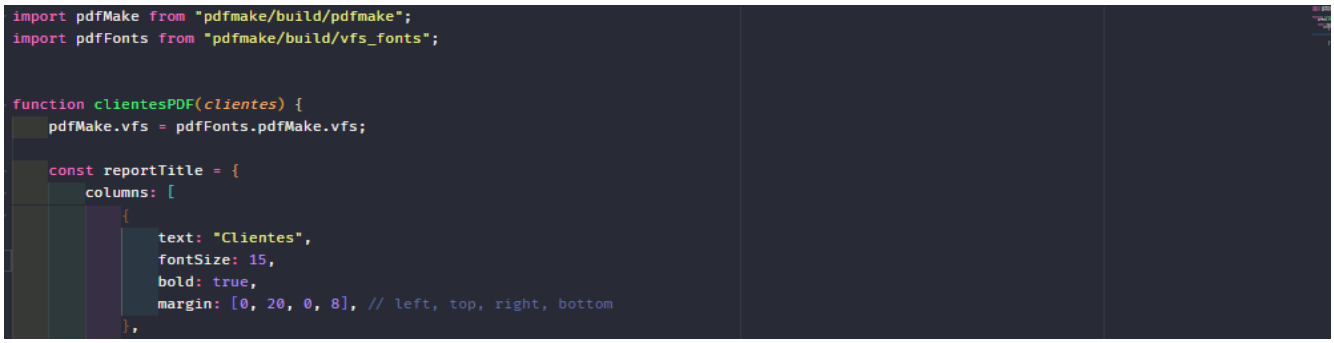
- require('pdfmake/build/pdfmake.js');
- require('pdfmake/build/vfs_fonts.js');

São dessa forma que importamos em nossa aplicação, seja em react ou nodejs. Temos, por exemplo, o PDFmake.

Exemplos

- Como fazer um pdf ?

Dentro do meu Github a modelos de pdfs feitos tirando do Playground, a ideia é simples seria o seguinte:



```
import pdfMake from "pdfmake/build/pdfmake";
import pdfFonts from "pdfmake/build/vfs_fonts";


function clientesPDF(clientes) {
    pdfMake.vfs = pdfFonts.pdfMake.vfs;

    const reportTitle = {
        columns: [
            {
                text: "Clientes",
                fontSize: 15,
                bold: true,
                margin: [0, 20, 0, 8], // left, top, right, bottom
            },
```

Como foi dito no começo desse documento, você importa a lib no seu componente; e quando chamarmos a função dentro, configuramos o nosso pdfMake. Ele seria pdfMake.vfs = pdfFonts.pdfMake.vfs; estamos importando nossa fonts e assim podemos trabalhar com nossa font customizada. Sem isso, poderá acontecer um erro.

ReporTitle é nome que você pode definir do seu gosto. É aqui onde vamos definir uma variável onde vai conter todo texto e toda customização de font, cores e

o nosso logo. Ao qual no meu componente está o logo da netshoes; é aqui onde trabalho o título do relatório.

No nosso Objeto temos um text, é onde colocamos nosso título do relatório que nesse caso é Clientes:

text: "Clientes"

O nosso próximo objeto é o font size, ao qual dará o tamanho do nosso título. Neste caso é:

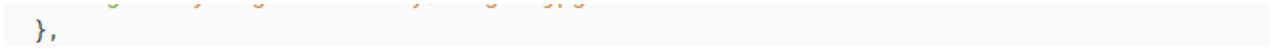
fontSize: 15,

Depois temos mais dois objetos que seriam para deixar nosso título com bold; e onde colocamos nossa margin do título que ficou da seguinte maneira:

bold: true,

margin: [0, 20, 0, 8], // left, top, right, bottom

Já na parte da imagem no exemplo como mostra no print de cima vemos que é usado base64 porem segunda documentação podemos usar assim logos e imagens



Nesse print tirado da documentação fala que podemos usar imagens de diretórios.

Os details é onde vamos ter a listagem de dados do nosso relatório. É aqui que renderizamos o nosso título da coluna. As nossas colunas que advém lá do nosso banco de dados conforme mostra o print adiante:

Essa renderização vai acontecer nessa situação: por exemplo, function clientesPDF(*clientes*), traz para nós os nossos clientes listados do nosso banco de dados.

Dentro do nosso details há o objeto table. O table tem o layout retirado da documentação de lib, assim como o código retirado do meu GitHub e demais exemplos de layouts partiram da mesma documentação. No exemplo adiante, usaremos o layout lightHorizontalLines. Dentro da table temos:

headerRows: 1(seria onde vamos colocar o número de linhas que vamos renderizar do nosso header)

```
//onde você coloca os stilos e tamanhos da header da tabela
const details = [
    {
        table: {
            headerRows: 1,
            widths: ["*", "*", "*", "*"],

            body: [
                [
                    { text: "Código", style: "tableHeader", fontSize: 10 },
                    { text: "Nome", style: "tableHeader", fontSize: 10 },
                    { text: "E-mail", style: "tableHeader", fontSize: 10 },
                    {
                        text: "Telefone",
                        style: "tableHeader",
                        fontSize: 10,
                    },
                ],
                ...dados,
            ],
        },
        layout: "lightHorizontalLines", // headerLineOnly
    },
];
```

widths: ["*", "*", "*", "*"] (Esse é o parâmetro e onde colocamos o tamanho que queremos para a nossa coluna. É onde fazemos essa customização para as colunas. O motivo do widths estar com asterisco é o fato do relatório ser algo dinâmico, pois os widths se ajustam automaticamente).

Depois temos o body, que é conteúdo da nossa table; e dentro deles colocamos o nosso objeto que é o header:

Olhando a imagem acima, podemos ver que temos o nosso text do header que será explicado conforme o exemplo adiante. Primeiro vemos o código, depois o style vem com o

```
{ text: "Código", style: "tableHeader", fontSize: 10 },
{ text: "Nome", style: "tableHeader", fontSize: 10 },
{ text: "E-mail", style: "tableHeader", fontSize: 10 },
{
    text: "Telefone",
    style: "tableHeader",
    fontSize: 10,
},
```

padrão tableHeader definido pela documentação. Após, vêm o fontSize com tamanho 10.

Fechando o componente details, quando for renderizar a nossa table; ele vai capturar o conteúdo passado dentro dele e será obedecido o layout definido que é lightHorizontalLines. Desta forma, é dentro do layout que vamos colocar as configurações sobre o modo que a tabela precisa ser realizada.

Com o const dados já definido, faremos um operador spread ao qual ...dados em que capta todo o conteúdo dentro de const e acrescenta o conjunto de informações que há dentro dos dados. Para renderizar o conteúdo em nossa tabela vamos construir uma const chamado dados que terá dentro esse conteúdo no print:

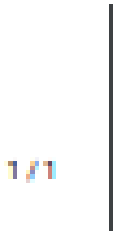
```
const dados = clientes.map((cliente) => {
    return [
        { text: cliente.id, fontSize: 9, margin: [0, 2, 0, 2] },
        { text: cliente.nome, fontSize: 9, margin: [0, 2, 0, 2] },
        { text: cliente.email, fontSize: 9, margin: [0, 2, 0, 2] },
        { text: cliente.fone, fontSize: 9, margin: [0, 2, 0, 2] },
    ];
});
```

Aqui pagamos o nosso cliente passado. No começo da função fazemos map que servirá para fazer loop de todos os nossos objetos contidos em clientes. Assim sendo, o cliente será passado de forma dinâmica. Dentro do return, temos o style ao qual é passado um fontSize que passamos o número 9 e o margin é igual ao contido na documentação. O margin tem como padrão os valores de left, top, right e bottom.

O rodapé do relatório colocamos a função que passa dois parâmetros baseados na documentação: *currentPage (qual é a página atual que está renderizando o nosso relatório)* e *pageCount(total de páginas).* Dentro fica do seguinte modo:

```
function Rodape(currentPage, pageCount) {
    return [
        {
            text: currentPage + " / " + pageCount,
            alignment: "right",
            fontSize: 9,
            margin: [0, 10, 20, 0], // left, top, right, bottom
        },
    ];
}
```

No trecho: text: currentPage + " / " + pageCount, é onde renderiza a paginação do tipo 1-1. Se tiver mais coisas dentro do relatório 1-2; teremos como diferença o alignment dentro de styles, isto é, será mexido no alinhamento do texto. Isto fará que a renderização será da seguinte forma no pdf.



Esse seria o rodapé.

A const DocDefinitios é onde vai conter todas as nossas definições de relatório. Dentro dele vamos colocar o tamanho da página, e demais atributos para chamar: ReporTitle, Details, Rodape e como bônus chamei a marca da água WorkGroup. Essa const foi fundamentada na documentação do PDFMake. Verifique como ficará o código de acordo com a imagem adiante:

```
const docDefinitios = {
    pageSize: "A4",
    pageMargins: [15, 50, 15, 40],
    watermark: { text: 'WorkGroup', color: 'blue', opacity: 0.3, bold: true, italics: false },
    header: [reportTitle],
    content: [details],
    footer: Rodape,
};
```

Dentro da documentação, verifique a parte **Page dimensions, orientation and margins.** Ao qual contém as definições sobre tamanho e margins da página.

pageSize: "A4",

pageMargins: [15, 50, 15, 40], // [left, top, right, bottom] or [horizontal, vertical] or just a number for equal margins

Já a parte do watermark, se você quiser que o relatório vá com marca de água da empresa; coloque conforme a descrição adiante. Para demais informações, consulte a documentação a parte denominada **watermark.**

watermark: { text: 'WorkGroup', color: 'blue', opacity: 0.3, bold: true, italics: false},

Conforme foi citado no começo deste documento, é aqui que chamamos as propriedades começadas pelo footer advindo da função de Rodapé. Dentro do Rodapé, chamamos o contente que oriunda de details e após o header que se origina de reportTitle.

header: [reportTitle], parte do título do nosso relatório

content: [details], corpo do nosso relatório

footer: Rodape, o rodapé do nosso relatório

Para renderizar o nosso relatório, fazemos da seguinte maneira como mostra no print abaixo.

```
    pdfMake.createPdf(docDefinitios).download();
}
```

Conforme o último print, é feita a renderização do relatório na tela através da chamada do docDefinitios. O docDefinitios contém o conjunto de definições para criar o pdf, pois usa o método createPdf. No final desta chamada, realiza o download. Dentro do download há a mágica de gerar o pdf no navegador e disponibilizar ao usuário realizar o download do pdf recém-gerado.