

Tabelas pra que te quero

Transcrição

Vamos começar supondo que trabalhamos para a Bytebank, uma empresa de serviços digitais bancários, que oferece para os clientes uma conta 100% digital. Seu principal produto é o cartão de crédito.

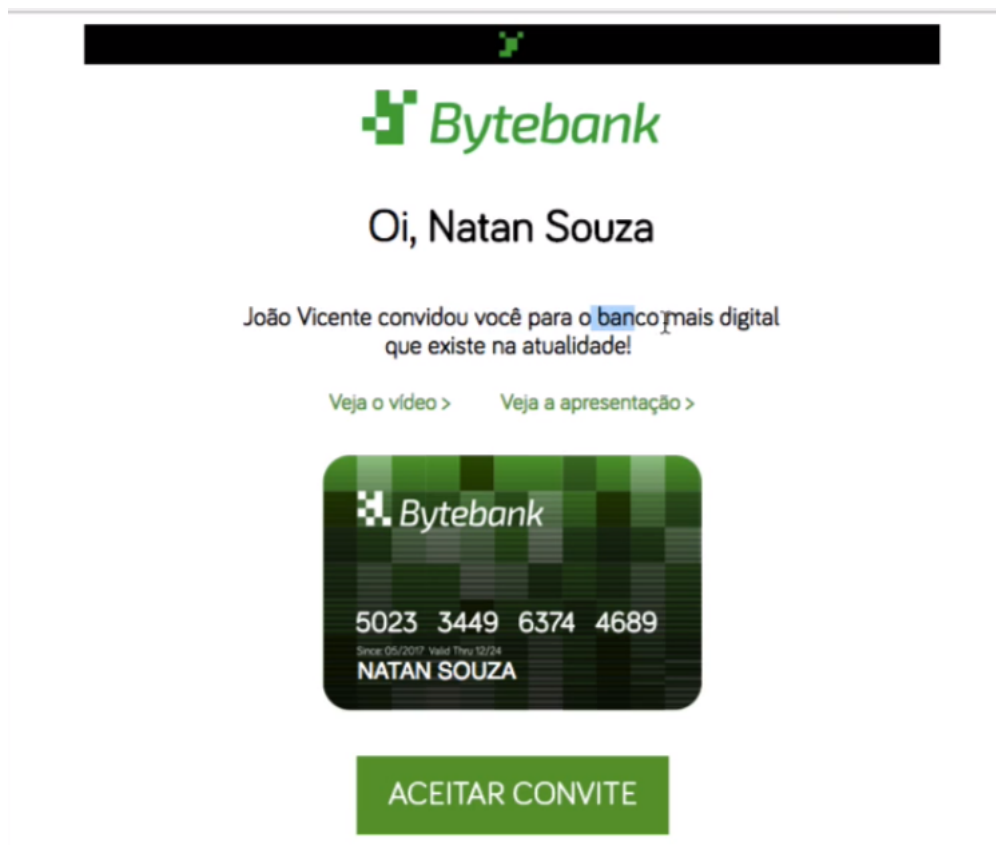
A equipe de Marketing do Bytebank pediu ao time de Front-end a criação de um e-mail marketing de convite do aceite do cartão. Na minha época, para ingressarmos no Orkut, por exemplo, também tínhamos que aceitar um convite enviado por quem já fazia parte da rede social. A Bytebank vai funcionar da mesma maneira.

Nosso e-mail terá um pequeno texto:

"Oi, Natan Souza

João Vicente convidou você para o banco mais digital que existe na atualidade!"

Abaixo teremos links convidando o usuário para assistir um vídeo de apresentação.



Para melhorar a primeira impressão do usuário e trabalharmos com a parte de UX também, incluímos o nome do potencial cliente na imagem do cartão. Assim, a pessoa consegue visualizar a aparência do cartão após aceitar o convite, cujo texto será uma preocupação da equipe de Back-end.

Na parte de baixo do e-mail, são citadas duas vantagens do Bytebank: o oferecimento de ingresso sem burocracias e o fato de ser 100% digital.



Há algumas informações da empresa, e links para as redes sociais no rodapé do e-mail. Se você fez o curso de E-mail Marketing (espero que sim!), você passou por todas as etapas, incluindo compreensão do *briefing*, elaboração do layout, consideração dos respiros internos, além de tudo que envolve a criação do HTML.

Para este curso, usaremos o editor *Sublime*, mas tenha a liberdade de usar a ferramenta que preferir para a edição de texto.

Você encontra o arquivo HTML do e-mail marketing nos [exercícios \(https://github.com/caiocmsouza/emailmkt-responsivo1/archive/66333bedcca43085d826080021049022fd493849.zip\)](https://github.com/caiocmsouza/emailmkt-responsivo1/archive/66333bedcca43085d826080021049022fd493849.zip), disponível para uso no decorrer do curso.

Ao abrirmos o arquivo `convite-original.html`, é possível perceber que estamos usando um Doctype antigo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN" "http://www.w3.org/TR/xhtml1/DTD/xht
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Convite do ByteBank</title>
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
<!-- ... -->
```

Se analisarmos o arquivo, veremos que ele possui uma tabela logo após a abertura da tag `<body>`:

```
<body>

  <center>
    <table align="center" cellpadding="0" cellspacing="0" border="0" width="600">

      <tbody>
```

```
<tr bgcolor="#000000">
  <td colspan="3" align="center" style="padding: 5px;">
    
  </td>
</tr>

<!-- ... -->
```

Estamos usando esse Doctype que vai contra as boas práticas comumente adotadas, as quais inclusive foram apresentadas nos cursos da Alura. No entanto, não temos opção! Trabalharemos com esse código porque nossos usuários não utilizam apenas Chrome, Firefox, Safari, Edge ou IE. Dependemos de outros *client-side emails* ("clientes de e-mail", traduzido para o português).

Por exemplo, dependemos do Gmail, do Outlook.com, dentre outros. É preciso considerar ainda se o usuário está abrindo o Gmail no navegador ou no aplicativo, e todos esses cenários se distinguem uns dos outros.

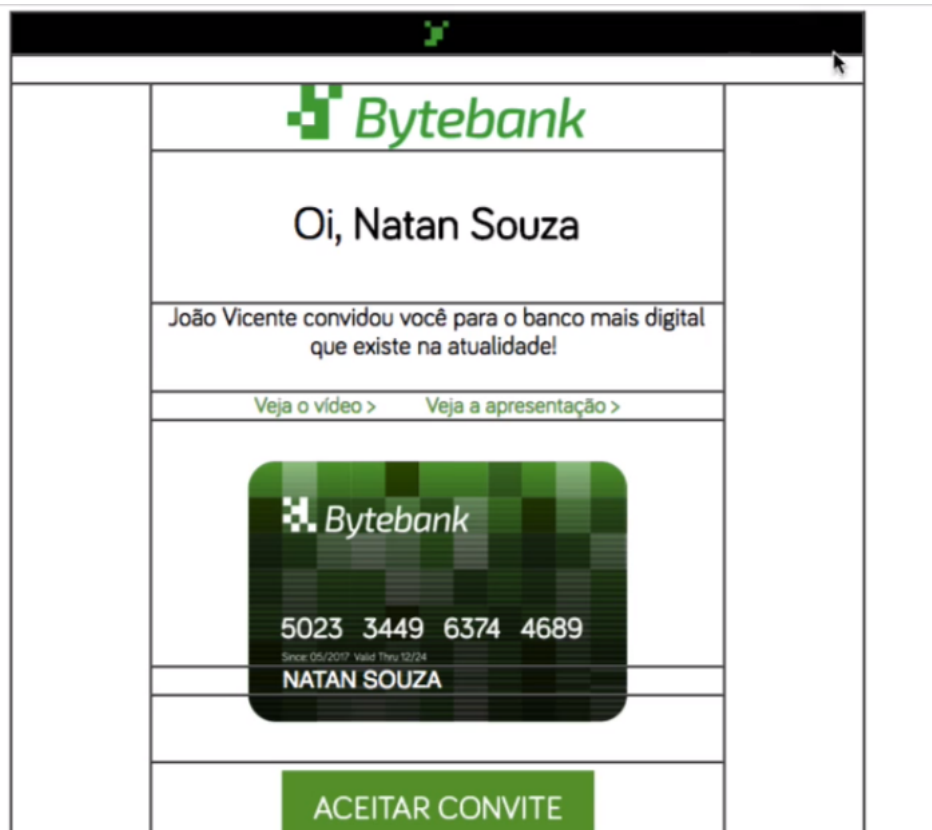
As funcionalidade continuarão as mesmas, mas se o navegador não der suporte para uma borda arredondada usada no e-mail, nosso layout vai ser alterado. Significa que precisamos lidar com essas particularidades e, para evitar problemas, utilizaremos tabelas.

Caso o suporte para diferentes tipos de clientes não seja necessário, há a liberdade de usar `<div>` e outras tags do HTML5. Minha recomendação, no entanto, é a adoção do `<table>`. Desta forma, temos a garantia de que nosso e-mail funcionará em diversos *e-mail clients*.

Se alterarmos o valor da borda da tabela para `1`, conseguiremos visualizar a estrutura da tabela:

```
<body>
  <center>
    <table align="center" cellpadding="0" cellspacing="0" border="1" width="600">

<!-- ... -->
```



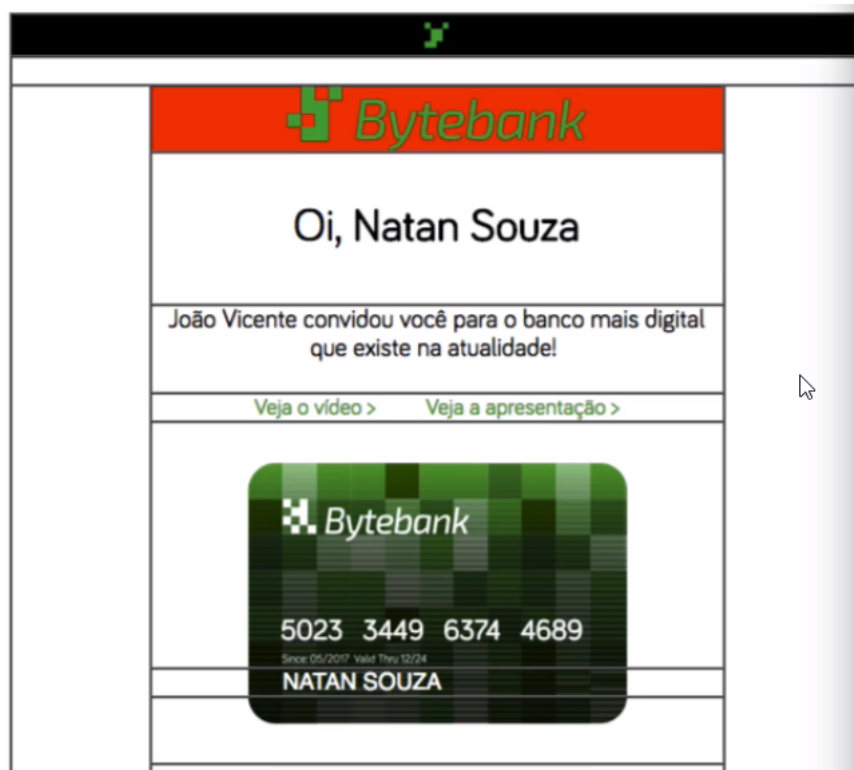
Vamos analisar o código de uma tag `<tr>` , localizada dentro do `<table>` :

```
<tr>
  <td width="100" rowspan="8">&nbsp;</td>
  <td width="400" align="center">
    
  </td>
  <td width="100" rowspan="8">&nbsp;</td>
</tr>
```

Este trecho é referente a primeira linha da tabela. Adicionaremos um `bgcolor` vermelho dentro de uma `<td>` para facilitar sua localização na tabela.

```
<tr>
  <td width="100" rowspan="8">&nbsp;</td>
  <td width="400" align="center" bgcolor="red">
    
  </td>
  <td width="100" rowspan="8">&nbsp;</td>
</tr>
```

A primeira `<td>` possui valor de `rowspan` igual a 8 , ou seja, ela está ocupando oito linhas.



Podemos testar modificando os valores das `td`s e verificar quais são as áreas relacionadas na tabela. Porém trabalharemos com o código como estava anteriormente:

```
<tr>
  <td width="100" rowspan="8">&nbsp;</td>
  <td width="400" align="center">
    
  </td>
  <td width="100" rowspan="8">&nbsp;</td>
</tr>
```

Na `<tr>` seguinte, encontraremos no código um atributo não utilizado com frequência atualmente.

```
<tr>
  <td align="center">
    <font size="6" face="Bariol, Arial, sans-serif"><br>
    Oi, <strong>Natan Souza</strong><br><br>
    </font>
  </td>
</tr>
```

O atributo `face` era bastante utilizado na tag ``, para indicação das alterações da família de fonte. Observe que no lugar do `font size` no CSS, colocamos o atributo `size` diretamente na ``.

Se descermos mais no código, encontraremos pequenas "gambiarras" com uma linha inteira com diversos ` ` (*non-breaking space*), usada quando não queremos uma quebra automática da linha.

```
<tr>
  <td align="center">
    <a href="#" style="text-decoration: none">
      <font size="3" face="Bariol, Arial, sans-serif" color="#47a138">
```

[Veja o vídeo >](#)

[illegible]

```
<a href="#" style="text-decoration: none">
```

```
<font size="3" face="Bariol, Arial, sans-serif" color="#47a138">
```

Veja a apresentação >

</td>

No caso do e-mail marketing, não é possível focarmos em boas práticas se queremos atender o maior número de *client-side*. Infelizmente, nosso código ficará um pouco "sujo".

A seguir, vamos analisar o trecho referente à imagem exibida do cartão, na qual o nome do cliente em potencial é colocado de forma dinâmica:

|

```
<td align="center">
```



</td>

|
 |

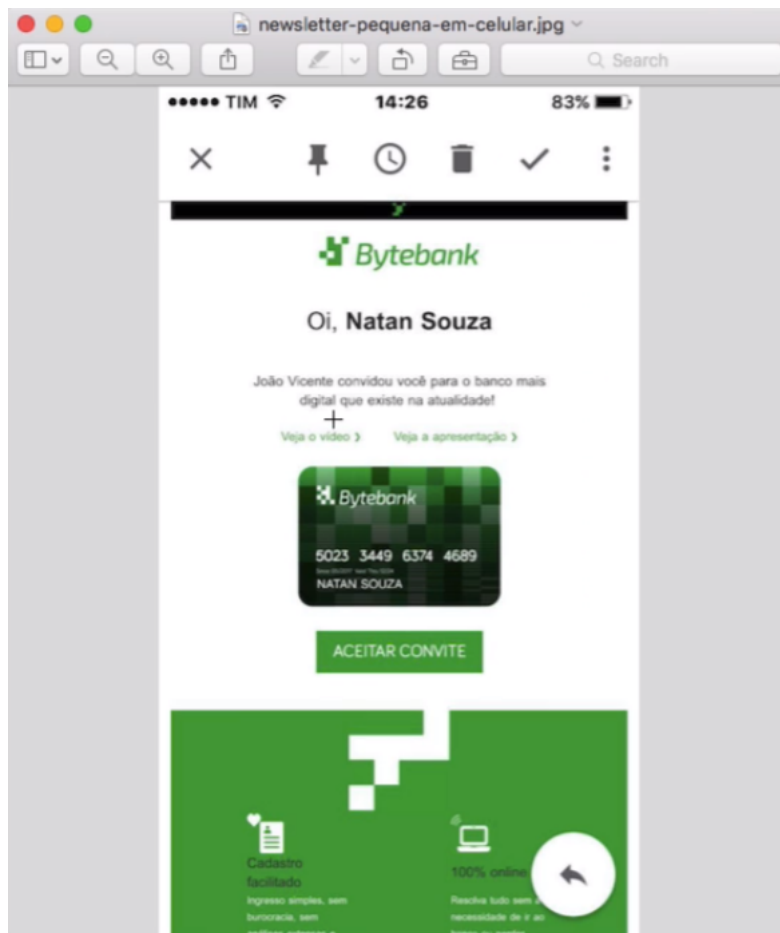
```
<font color="#FFFFFF" style="font-family: sans-serif; padding-left: 23%">
```

NATAN SOUZA

</td>

A imagem poderia vir pronta do Back-end, mas optei por resolver o problema de outra maneira, incluindo o nome do cliente.

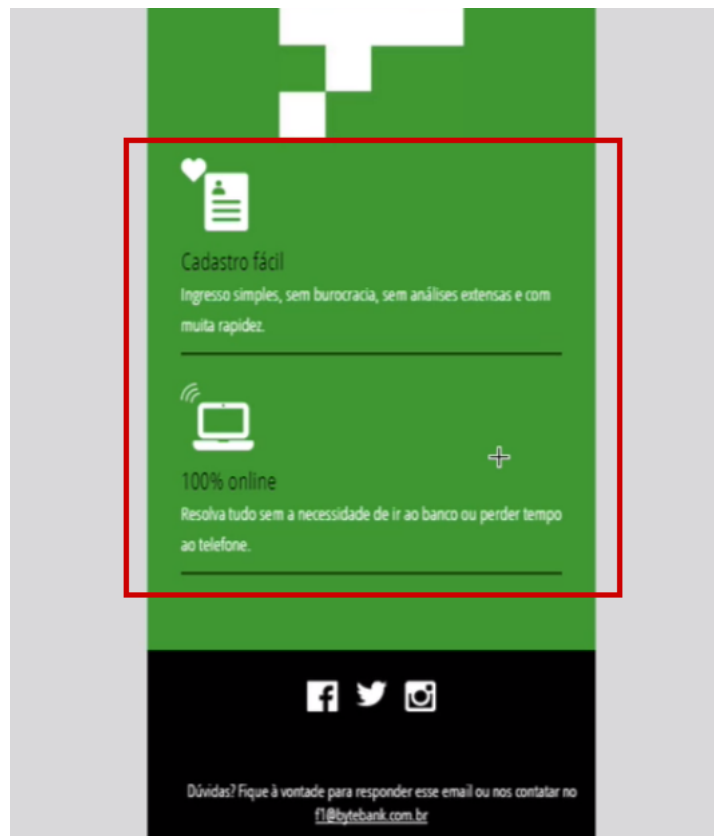
Então, temos nosso HTML e acreditamos que tudo está funcionando corretamente. No entanto, ao abrirmos o e-mail no celular, ele é exibido da seguinte forma:



O e-mail é exibido integralmente, e por isso as fontes ficaram bem pequenas. Precisaríamos dar zoom com os dedos para ler o texto.

No curso, apresentaremos os *clients email* a que daremos suporte, sendo que o primeiro é o Gmail, mas o objetivo é evitar que o usuário tenha que dar um zoom durante a leitura da mensagem, prejudicando a experiência.

Na versão responsiva que construiremos, o conteúdo vai se adaptando proporcionalmente ao dispositivo utilizado, tornando a leitura mais agradável. O conteúdo do rodapé, por exemplo, poderão ser exibidas uma embaixo da outra, em coluna:



A legibilidade melhorou muito, e o layout faz mais sentido. É o que faremos nas próximas aulas!