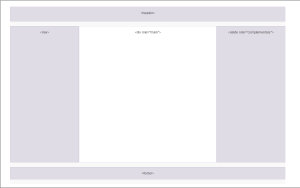
**Bootstrap 3: Criando Layouts e Grids**

**http://webdevacademy.com.br/tutoriais/ux/bootstrap-3-layouts-sistema-grids/**

[](http://i2.wp.com/media.webdevacademy.com.br/2014/02/layout-com-bootstrap.png)

Neste artigo, você vai entender como funciona o principal conceito do Bootstrap, que é o Grid System.

Vamos ver, também, alguns dos principais itens e conceitos como container, rows, e columns.

Depois, você pode ver como fazer os quatro principais tipos de layout com o Bootstrap:

* [Como Criar um Layout Simples (Fixo e Fluido)](http://webdevacademy.com.br/tutoriais/bootstrap-template-layout-simples/)
* [Como Criar um Layout com Duas Colunas](http://webdevacademy.com.br/tutoriais/ux/bootstrap-template-layout-duas-colunas/)
* [Como Criar um L](http://webdevacademy.com.br/tutoriais/ux/bootstrap-3-layouts-sistema-grids/#two)[ayout com Três Colunas](http://webdevacademy.com.br/tutoriais/bootstrap-template-layout-tres-colunas/)

Mas, antes de começar, você precisa conhecer algumas regras que o Bootstrap tem para poder posicionar os elementos no layout, e utilizar o framework da melhor forma possível.

**Entendendo o Grid System do Bootstrap**

Primeiramente, você precisa saber que o Bootstrap funciona com um sistema de grids (grades) para posicionar os elementos na página.

Esse mecanismo funciona como **uma espécie de tabela abstrata**, e é responsivo (responsive), orientado a dispositivos móveis (mobile first) e se ajusta de acordo com a tela (fluid), quando ela muda de tamanho ou de orientação.

**É extremamente importante que você entenda, e domine, esse grid system para trabalhar bem com o Bootstrap.**

Vamos ver, agora, alguns conceitos importantes antes de começar a criar os nossos layouts.

**Mobile First**

O Bootstrap é Mobile First.

Isso quer dizer que o framework assume, inicialmente, que a tela é de um dispositivo móvel, com tamanho pequeno. Assim, ele adapta todos os conteúdos para o tamanho menor.

Depois, ele verifica o tamanho real da tela e vai ajustando os itens para que fiquem posicionados corretamente, conforme o tamanho e a resolução.

Quando você for projetar, e implementar, o layout e o design do site (ou app) você deve projetar primeiro para as telas menores (celulares), depois para telas médias (tablets) e, por fim, para telas maiores (desktops, etc.).

Isso é a base do conceito de Mobile First.

**Container**

No Bootstrap, existe o conceito de container.

O container é uma *div*, que garante que o seu layout vai ficar alinhado na página, e com margens para as laterais. Ele também centraliza o conteúdo na tela do browser. Dependendo do tamanho da tela, o container definirá automaticamente as larguras do seu layout, para que o conteúdo seja melhor visualizado.

Você deve usar um container para englobar o posicionamento de todos os elementos do layout da página.

Então, dependendo do dispositivo e da orientação tela, o seu container pode ficar com o tamanho de acordo com a tabela abaixo:

|  |  |  |  |
| --- | --- | --- | --- |
| **Em Celulares** | **Em Tablets** | **Desktops** | **Telas Grandes** |
| Largura Automática | Largura Máx. de 750px | Largura Máx. de 970px | Largura Máx. de 1170px |

Para criar um container com largura fixa, e ajustada ao tamanho da tela, você pode criar uma div com a classe***.container,*** como no exemplo:

<div class"container">  
...  
</div>

Ou, você pode usar a classe ***.container-fluid*** para fazer o container (e o layout) ficar com 100% de largura:

<div class"container-fluid">  
...  
</div>

A sua página deve ter, pelo menos,**um container geral** (também conhecido como wrapper). Ele abrange todo o escopo da página. E todos os outros elementos visuais da página devem estar dentro dele.

Há situações em que você poderá (e precisará) usar o container dentro de áreas como o header e o footer.

**Row**

As rows (linhas), no Bootstrap, definem as divisões horizontais do seu layout.

Essas rows devem ficar dentro do container, e podem ser aplicadas a qualquer tag que defina estrutura, como div, header e footer.

Para criar uma *row* você pode definir uma *div* com a classe ***.row,***como no exemplo:

<div class"container">  
    <div class="row">  
    ...  
    </div>  
</div>

Seu layout pode ter quantas rows forem necessárias. E você pode colocar *rows* dentro de *rows*, também.

As *rows* sempre irão ficar uma abaixo da outra.

**Columns**

As columns (colunas), no Bootstrap, definem as divisões verticais das rows (linhas) do seu layout.

Columns devem estar sempre dentro das rows, e elas definem espaços na row para que você coloque os itens visuais ou conteúdos que foram projetados.

Então, no Bootstrap, você tem linhas (rows) e colunas (columns) para definir um layout. É, exatamente, como uma grade ou matriz (ou tabela), porém, utilizando div’s com classes.

Para criar uma *column*você pode criar uma *div* com os prefixos pré-definidos pelo Bootstrap***,***como no exemplo:

<div class"container">  
    <div class="row">  
        <div class="col-md-6"> </div>  
        <div class="col-md-6"> </div>  
    </div>  
</div>

Neste exemplo, temos uma linha (.row) com duas colunas (.col-md-6).

O número seis, no final de cada classe de coluna, define o espaço que ela ocupa na linha. Assim, neste exemplo, teríamos a linha (row) dividida exatamente no meio por duas colunas, já que usamos o número seis.

Os prefixos de colunas servem para indicar em quais tipos de tela a coluna vai se manter posicionada como no design principal. Os prefixos têm o seguinte padrão:

|  |  |  |  |
| --- | --- | --- | --- |
| **Para Celulares** | **Para Tablets** | **Para Desktops** | **Para Telas Grandes** |
| .col-xs-\* | .col-sm-\* | .col-md-\* | .col-lg-\* |

Você deve substituir os asteriscos pelo tamanho da coluna (de 1 a 12).

**O mais comum é utilizar o prefixo .col-md-\* para os sites e web apps**, porque ele mantém o design principal em computadores e nos principais tablets. Apenas nos celulares o conteúdo passa a ficar vertical, ou seja, “um embaixo do outro” em uma única coluna.

Agora, eu vou lhe explicar o “pulo do gato” do Bootstrap:

**A soma dos tamanhos das colunas deve ser igual a doze (ou menor).**

No exemplo, nos tínhamos a seguinte soma:

<div class="row"> <!-- 6 + 6 = 12 -->  
    <div class="col-md-6"> </div>  
    <div class="col-md-6"> </div>  
</div>

Mas, você pode fazer qualquer combinação:

<div class="row"> <!-- 4 + 8 = 12 -->  
    <div class="col-md-4"> </div>  
    <div class="col-md-8"> </div>   
</div>

<div class="row"> <!-- 3 + 6 + 3 = 12 -->  
    <div class="col-md-3"> </div>  
    <div class="col-md-6"> </div>  
    <div class="col-md-3"> </div>  
</div>

<div class="row"> <!-- 10 + 2 = 12 -->  
    <div class="col-md-10"> </div>  
    <div class="col-md-2"> </div>  
</div>

O importante é que, no final, a soma dê 12. Se a soma der mais de 12 em uma linha, as colunas excedentes serão posicionadas em uma nova linha, abaixo da atual.

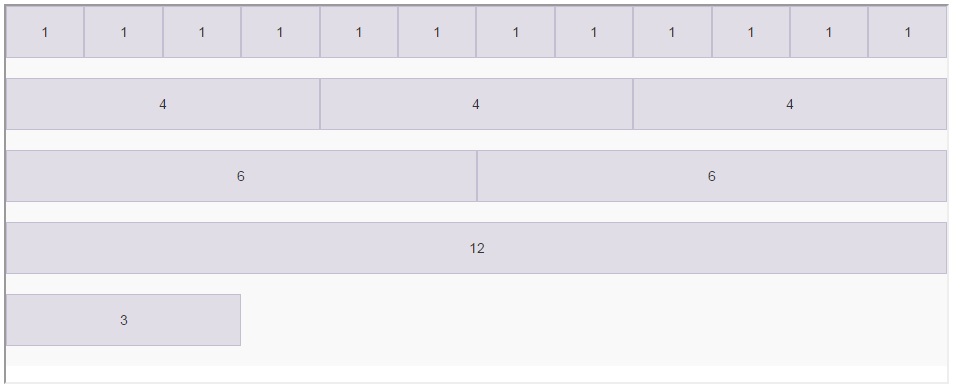
É muita coisa, né?! Tudo isso que vimos até agora serve apenas para criar o grid system do Bootstrap…

**Grid System, ou o Bootstrap Grid…**

O Bootstrap tem um mecanismo de grades (Grid System) para calcular as posições dos itens na tela.

Esse grid system é responsivo, mobile first e flexível. Ele pode aumentar, ou diminuir, conforme a tela mude (ou o viewport). Para isso, ele usa aquelas classes e conceitos que você viu anteriormente, além de algumas combinações para gerar layouts mais semânticos.

Veja um exemplo de grid do Bootstrap:



Neste exemplo temos o seguinte:

* **cada linha cinza-escuro é uma row**, ou seja, uma div com a class=”row” (mas pode ser qualquer outra tag de estrutura)
* **cada item dentro das rows é uma column**, ou seja, uma div com um prefixo de coluna (class=”col-md-1″, como na primeira linha por exemplo)

A primeira linha tem doze *colunas*de tamanho *unitário (12 \* 1 = 12). A segunda tem três colunas de tamanho quatro (3 \**4 = 12).  A terceira tem duas colunas de tamanho 6 (2 \* *6 = 12).  A quarta tem uma columa de tamanho doze (1 \**12 = 12).  E a última tem uma coluna de tamanho 3 (1 \* 3 < 12).

Acho que você conseguiu visualizar, agora.

Lembre-se que o grid system padrão do Bootstrap utiliza **12 colunas** para criar o container (é possível mudar isso, criando uma *build* personalizada do Bootstrap).

Para adaptar um design de website, ou de web app, ao Bootstrap você pode pegar o projeto desse design e dividir em 12 partes iguais verticalmente. Depois, basta fazer os ajustes necessários para que a estrutura se encaixe no grid.

O maior benefício disso é que deixa os layouts matematicamente corretos, e flexíveis para qualquer tamanho de tela.

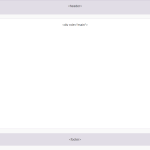
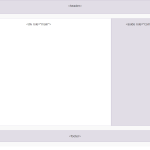
Vamos ver, agora, como fazer os layouts mais comuns com o Bootstrap…

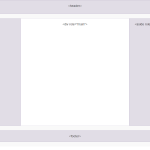
**Como Criar Layouts com o Bootstrap**

Agora que você já conhece os conceitos iniciais, já é possível começar a escrever a marcação de layouts com o Bootstrap.

Para que este tutorial não fique mais extenso, e que você não fique confuso com os conceitos do Grid System, eu separei os principais tipos de layout para você aprender.

Estes tutoriais são passo-a-passo:

[](http://webdevacademy.com.br/tutoriais/bootstrap-template-layout-simples/)Como Criar um Layout Simples[](http://webdevacademy.com.br/tutoriais/ux/bootstrap-template-layout-duas-colunas/)Como

Criar um Layout de Duas Colunas[](http://webdevacademy.com.br/tutoriais/bootstrap-template-layout-tres-colunas/)Como Criar um Layout de Três Colunas

Estude eles, e tente implementar a marcação. Vai facilitar, e muito, seu entendimento do Grid do Bootstrap.

**Outros Layouts**

É possível criar diversos tipos de layouts usando as combinações de rows, columns e classes de prefixo.

Tudo depende de um bom protótipo e da marcação correta para encaixar os itens no grid system do Bootstrap.

**Próximos Passos…**

A partir daqui, você conseguirá criar várias coisas com o Bootstrap. Lembre-se de sempre consultar a documentação para ver as possíveis classes do framework.

Aproveite para conferir os outros [tutoriais de Bootstrap](http://webdevacademy.com.br/tag/bootstrap/) aqui da Academy. Se ficou alguma dúvida é só deixar abaixo nos comentários.

Até a próxima!