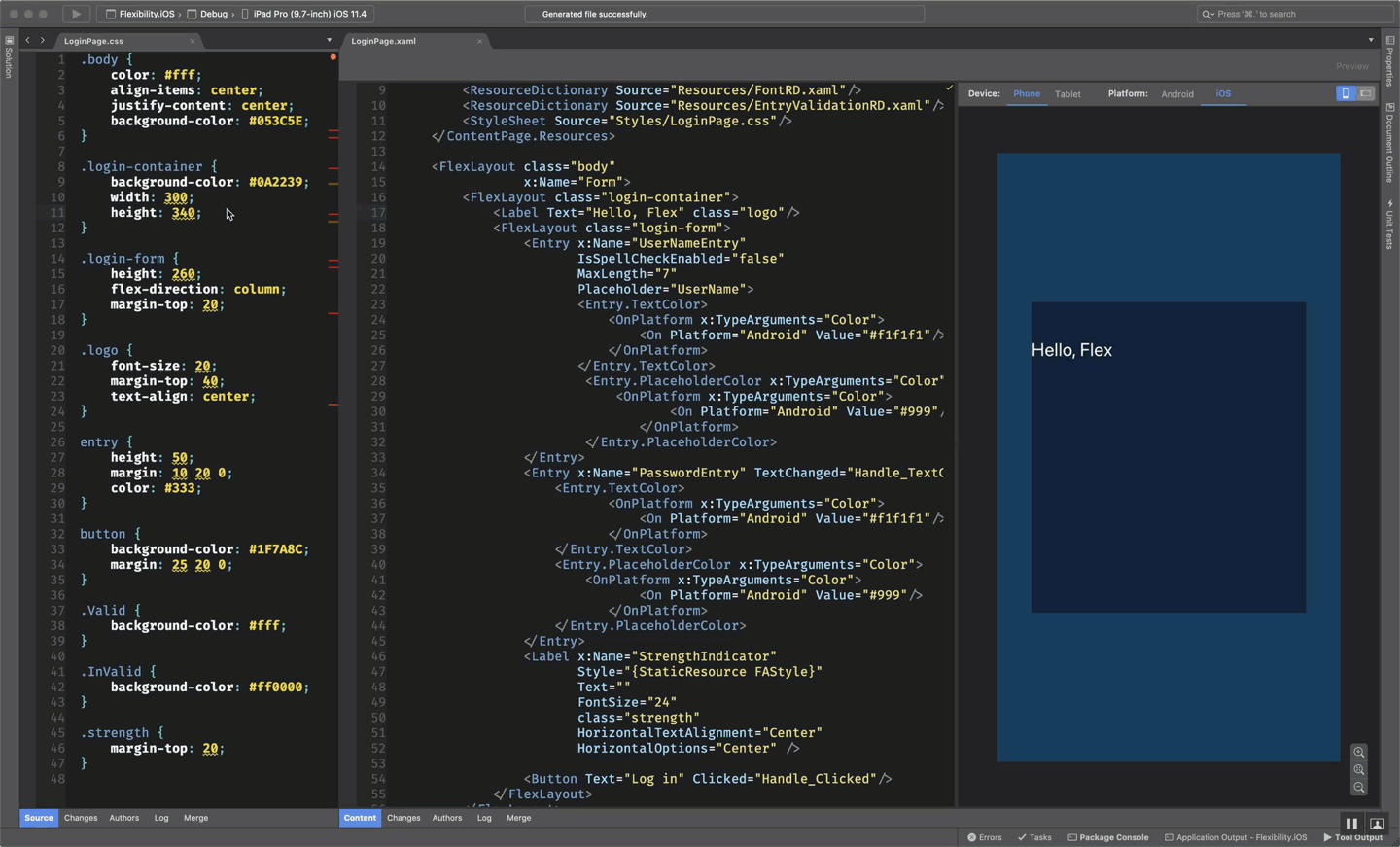
**MOBILE DESIGNS ADAPTATIVOS COM O FLEXLAYOUT**

**Escrito em 3 de Agosto por David Ortinau, traduzido por João Vitor Rogacheski.**

O Xamarin.Forms é fantástico para um rápido envio de um único código para varias plataformas e dispositivos. Assim que estiver presentes em todas essas telas diferentes, você precisará ter ser certeza que sua Interface de usuário pareça como você imaginou, e até adapte o layout para seus objetivos específicos. O FlexLayout faz isso mais fácil do que nunca e prove novas opções para você espaçar e distribuir sua interface de usuário para colocar em diferentes dimensões.

**Deixa Fluir**

As tags Children colocada em um FlexLayout container irão fluir tanto na linha horizontal quanto em uma coluna vertical, chamado de eixo principal. Isso é setado pela propriedade de Direção. Usando essa propriedade sozinha, o container se comporta como um StackLayour. Abaixo temos uma página de login que irá mostrar.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <FlexLayout class="body">      <FlexLayout class="login-container">          <Label Text="Login" class="logo"/>          <FlexLayout class="login-form">              <Entry Placeholder="UserName"/>              <Entry Placeholder="Password" IsPassword="true"/>              <Button Text="Log in"/>          </FlexLayout>      </FlexLayout>  </FlexLayout> |

O interior do layout coloca um label e entra de forma vertical configurado pelo flex-directon com a propriedade coluna do CSS

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | .login-container {      background-color: #803A3F44;      width:300;      height:380;      flex-direction: column;  } |

Nota: Se o CSS não for da sua preferência, as propriedades estão disponíveis no XAML e também no C#. <FlexLayour Direction=”Column”...>

Para adicionar a linha e a coluna (padrão), você também pode mudar a ordem da “Children” usando a “ColumnReverse” e “RowReverser”. Vamos ver um StackLayout fazer isso”

Note como o form de Login está centralizado na tela. Isso é algo que o FlexFlayout faz facilmente usando “align-items” para centralizar junto com os eixos escolhidos (nesse caso o Y ou eixo vertical), e o “justify-content” para distribuir as “children” para o centro do container (nesse caso junto com outros eixos).

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | .body {      align-items: center;      justify-content: center;      color: #fff;      background-color: #16222a;  } |

Baixe este exemplo do GitHub <https://github.com/davidortinau/flexibility>.

**Amarre**

Agora que você viu como o FlexLayout irá fluir o conteúdo da sua view, o que acha quando seu conteúdo ultrapassa a largura ou altura?

É agora que você toma vantagem de amarrar. Você pode escolher muitas opções :

* wrap
* nowrap
* reverse



Vamos dizer que seu aplicativo tem um grid de conquistas assim como esse mostrado no demo da Conference Vision da Build 2018. Você popula o container com uma serie de itens iguais e quer ter a certeza que eles fluiram na mesma direção.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <FlexLayout class="container">      <local:AchievementView class="item" BindingContext="{Binding Achievements[0]}" />      <local:AchievementView class="item" BindingContext="{Binding Achievements[1]}" />      <local:AchievementView class="item" BindingContext="{Binding Achievements[2]}" />      ...  </FlexLayout> |

|  |  |
| --- | --- |
| 1  2  3  4  5 | .container {      flex-direction: row;      flex-wrap: wrap;      justify-content: space-between;  } |
|  |  |

Assim como você usou o justify-content no exemplo prévio para centralizar o container de login com a view, você pode usar essa propriedade aqui para uniformizar o espaço do conteúdo. Escolha:

|  |  |
| --- | --- |
| Center | Indicates that child elements will be grouped toward the center of the parent. |
| End | Indicates that child elements will be justified to the end of the line. |
| SpaceAround | Indicates that child elements will be spaced with one unit of space at the beginning and end, and two units of space between them, so that the elements and the space fill the line. |
| SpaceBetween | Indicates that child elements will be spaced with equal space between units and no space at either end of the line, so that the elements and the space fill the line. |
| SpaceEvenly | Indicates that child elements will be spaced to have the same space between each element as there is between the beginning and end elements and the nearest edges of the parent. |
| Start | Indicates that child elements will be justified to the start of the line. |

Fonte: Docs

Explore mais como o FlexLayout foi usado no app Conference Vision pelo link do GitHub:

<https://github.com/microsoft/ConferenceVision>.

**Posicionando no espaço**

Construindo nesses exemplos, vamos explorar um layout mais avançado onde você quer espaçar o conteúdo proporcionalmente. Voltando ao exemplo da flexibilidade, há uma foto da pagina que amarra fotos e também redimensiona por porcentagem. Vamos dizer que você tem 3 imagens na linha por padrão, mas ocorre que você quer a imagem cheia, ou 2 imagens por linha? Você pode usar a propriedade “flex-basis” para ver por posição.

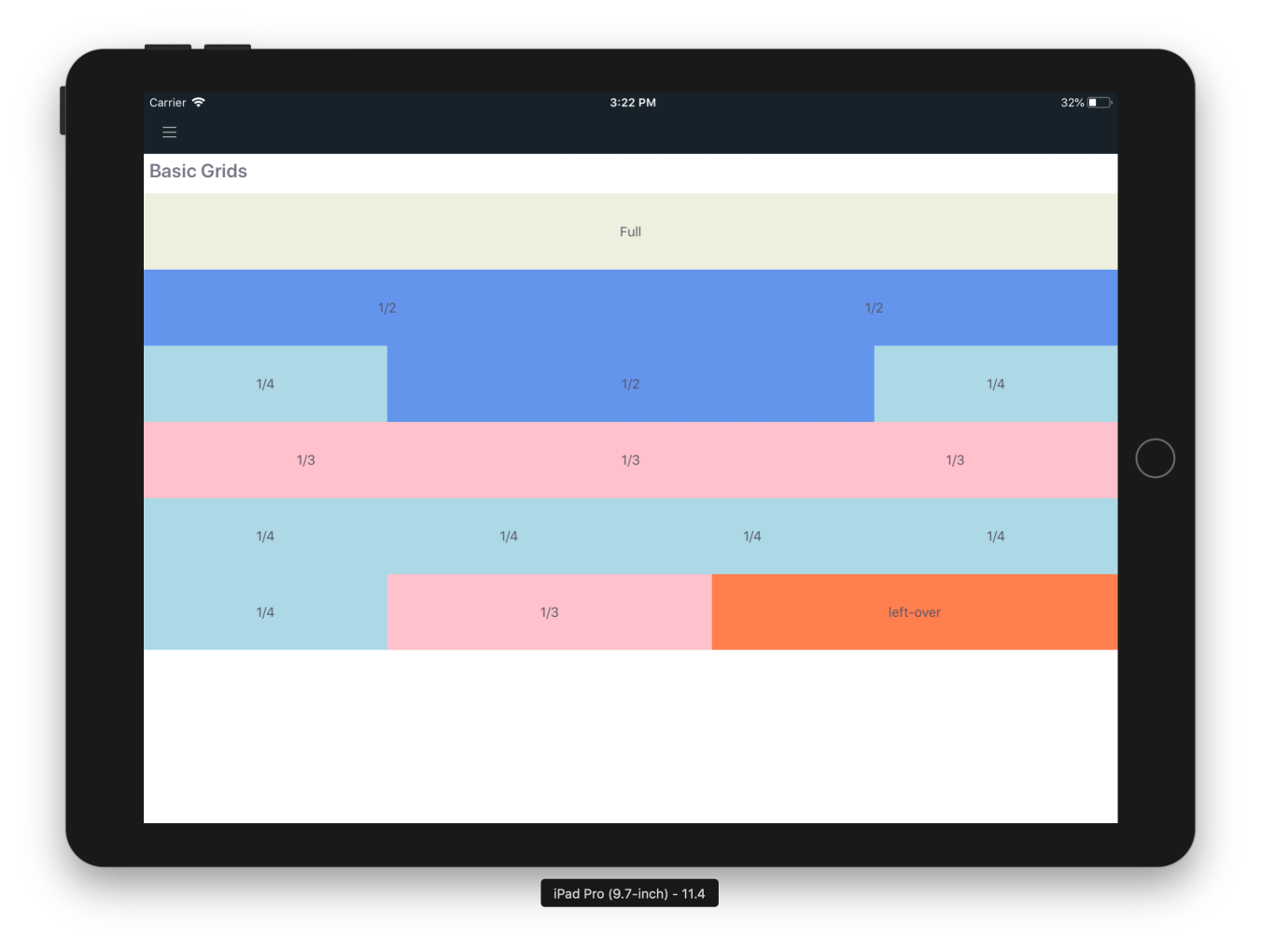
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | .photo {      flex-basis: 33.33%;      height: 100;  }    .half {      flex-basis: 50%;      height: 150;  }    .full {      flex-basis: 100%;      height: 200;  } |
|  |  |

O “Basis” é o quanto o item toma dos eixos principais. Nesse caso estamos trabalhando juntamente com o eixo X (flex-direction:row). O “Basis” é colocado no dispositivo por pixels ou porcentagem, fazendo então, um poderoso layout responsivo. Atribuindo as classes abaixo para as imagens do conteúdo, você pode posicionar nesse incrível grid.



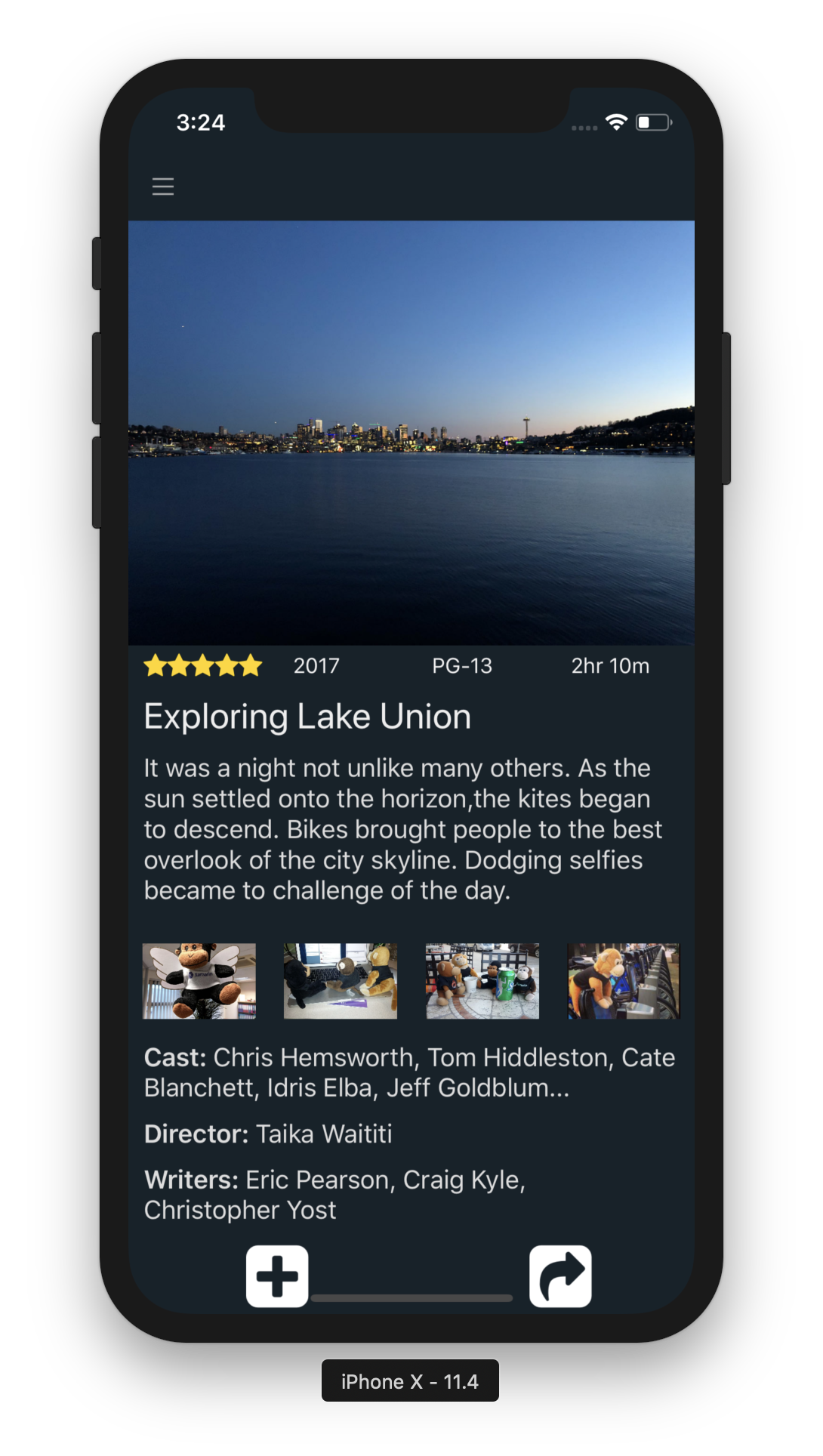
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | <FlexLayout Style="{StaticResource flex}">      <Image Source="m\_01.jpg"              class="photo"              Aspect="AspectFill" />        <Image Source="m\_02.jpg"              class="photo"              Aspect="AspectFill" />        <Image Source="m\_04.png"              Aspect="AspectFill"              class="photo" />        <Image Source="m\_03.png"              Aspect="AspectFill"              class="photo, full" />        <Image Source="m\_05.jpg"              Aspect="AspectFill"              class="photo" />        <Image Source="m\_06.jpg"              Aspect="AspectFill"              class="photo" />        <Image Source="m\_07.jpg"              Aspect="AspectFill"              class="photo" />        <Image Source="m\_08.jpg"              Aspect="AspectFill"              class="photo, half" />        <Image Source="m\_09.jpg"              Aspect="AspectFill"              class="photo, half" />        ...  </FlexLayout> |

Para um exemplo adicional de como usar o “Basis”, de uma olhada neste [this Grid page sample](https://github.com/davidortinau/flexibility/blob/master/Flexibility.Shared/GridPage.xaml).



Layout Avançado

O FlexLayout também pode ser usado para situações mais avançadas como esta pagina. Você pode construir nos mesmos métodos abaixo para facilmente conseguir. Abaixo está um layout que amarra linhas e usa um truque para alinhar. Para forçar a quebra de linha, adicione um elemento vazio com a “basis” em 100%.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | <FlexLayout class="container">      <Image class="poster" Source="seattle.jpg"/>      <Label Style="{StaticResource FAStyle}" class="popularity" Text="&#xf005;&#xf005;&#xf005;&#xf005;&#xf005;" />      <Label class="year" Text="2017" />      <Label class="rating" Text="PG-13" />      <Label class="length" Text="2hr 10m" />        <Label class="title" Text="Exploring Lake Union" />      <Label class="description" Text="It was a night not unlike many others. As the sun settled onto the horizon, the kites began to descend. Bikes brought people to the best overlook of the city skyline. Dodging selfies became to challenge of the day."/>        <Image Source="m\_11.jpeg"             Aspect="AspectFill"             class="photo"               />      <Image Source="m\_12.jpg"             Aspect="AspectFill"             class="photo" />        <Image Source="m\_13.jpg"             Aspect="AspectFill"             class="photo" />        <Image Source="m\_14.jpg"             Aspect="AspectFill"             class="photo"/>        <Label class="description">          <Label.FormattedText>              <FormattedString>                  <Span Text="Cast: " FontAttributes="Bold" />                  <Span Text="Chris Hemsworth, Tom Hiddleston, Cate Blanchett, Idris Elba, Jeff Goldblum..." />              </FormattedString>          </Label.FormattedText>      </Label>      <Label class="description">          <Label.FormattedText>              <FormattedString>                  <Span Text="Director: " FontAttributes="Bold" />                  <Span Text="Taika Waititi" />              </FormattedString>          </Label.FormattedText>      </Label>      <Label class="description">          <Label.FormattedText>              <FormattedString>                  <Span Text="Writers: " FontAttributes="Bold" />                  <Span Text="Eric Pearson, Craig Kyle, Christopher Yost" />              </FormattedString>          </Label.FormattedText>      </Label>      <Label class="action" Style="{StaticResource FAStyle}" Text="&#xf0fe;"/>      <Label class="action" Style="{StaticResource FAStyle}" Text="&#xf14d;"/>      </FlexLayout> |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65 | .container {      align-items: start;      align-content: start;      flex-wrap: wrap;      justify-content: space-around;  }    .poster {      flex-basis: 100%;  }    .summary {      background-color: black;      flex-direction: row;      justify-content: spacebetween;      height: 20;      margin: 10;  }    .popularity {      color: gold;      margin: 5 10 0;  }    .year, .rating, .length {      font-size: 14;      color: #f1f1f1;      margin: 5 10 0;      flex-grow: 1;  }    .break {      flex-basis: 100%;      height: 0;  }    .spacer {      flex-basis: 100%;      height: 0;  }    .title {      color: #eee;      font-size: 24;      margin: 10;      flex-basis: 100%;  }    .description {      color: #ddd;      margin: 10,0,10,10;      flex-basis: 100%;  }    .action {      color: #FFFFFF;      font-size: 48;      margin: 0 0 0 10;  }    .photo {      flex-basis: 20%;      height: 50;      margin: 15 0;  } |

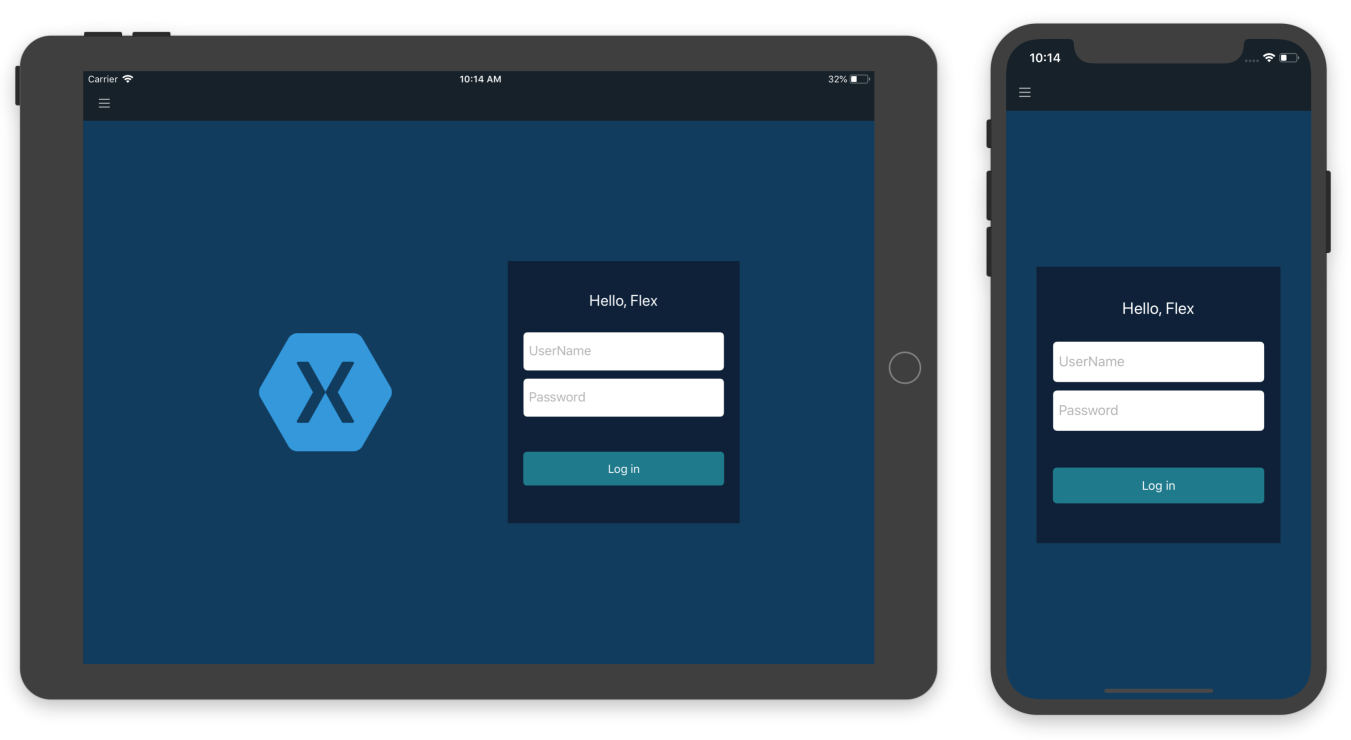
Há muitas outras propriedades que virão na sua mão quando começar a usar o FlexLayout, como “Grow” e “Shrink”. Essas propriedades fazem o que elas dizem. O “Grow” padroniza o 0 e qualquer numero positivo que ira ocupar o espaço disponível junto com o eixo principal. Isso é útil quando você quer que alguma coisa consuma o espaço disponível entre os eixos. O “Shrink”, por sua vez, fará o oposto, o padrão 0 ira ter certeza que não ira encolher se o irmão tentar ocupar mais espaço, quando o padrão usa um numero positivo, ira permitir que o elemento entregue espaço a outro.

Note que os elementos “.year”, .”rating”, “.lenght” tem “flex-grow:1;” que ira forçar eles a ocupar mais espaço do que eles conseguem, dando um bom espaço e puxando a estrela principal para o lado. Use essas propriedades e veja como elas se adaptam.

**Adaptando ao Layout do Tablet**

Quando o design do layout não for apenas para mobiles, mas também para tablets, você pode ir longe com a flexibilidade do FlexLayout e usar a porcentagem para setar o “basis”. Irá chegar um tempo que você precisará de um layout diferente, que levemente se ajustara o outro tamanho de tela. Para isso, você pode usar o “OnIdiom”. Esse pequeno ajudante, similar ao “OnPlataform”, deixara você especificar o estilo baseado no “idioma”: desktop, celular, tv e relógio.

Veja o primeiro exemplo, de uma pagina de login, e considere como você pode mover o form de login para a direita da tela no tablete enquanto coloca uma imagem na esquerda.



Nesse layout, colocamos a nova imagem logo após o form de login, e usamos a margem para dar mais espaço. Para ter certeza que aparecera apenas em tablets, usamos o “OnIdiom” para deixar visível.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <FlexLayout class="body"              x:Name="Form">      <Image Source="xamagon"              Aspect="AspectFit"              WidthRequest="275"              Margin="0,0,100,0"              IsVisible="{OnIdiom Tablet=true, Phone=false}"              />      <FlexLayout x:Name="LoginForm"                  class="container">          <Label Text="Hello, Flex" class="logo"/> |

Nota: O “OnIdiom” é novo no Xamarin.Forms 3.2.0, então tenha certeza de atualizar para a ultima versão para usar. Para um exemplo antigo acesse [unit test sample here](https://github.com/xamarin/Xamarin.Forms/blob/master/Xamarin.Forms.Xaml.UnitTests/OnPlatform.xaml).

Quando usamos “justify-content: center” o FlexLayout irá espaçar as coisas para melhor, Isso deixara o layout Incrivel!

**Começando a usar o FlexLayout hoje com o Xamarin.Forms**

Atualize seus projetos para o Xamarin.Forms 3.1.0 hoje via Gerenciador de Pacotes do Visual Studio e aproveite a ultima release estável com o FlexLayout. E se quiser receber as novas atualizações, olhe a prévia do Xamarin.Forms 3.2.0 como o “OnIdiom” e muito mais!

FONTE: https://blog.xamarin.com/adaptive-mobile-designs-with-flexlayout/