CAPÍTULO 20

APÊNDICE B: LAYOUTS ADAPTÁVEIS PARA DIFERENTES TAMANHOS DE TELA

Nos primórdios da App Store os desenvolvedores não tinham que se preocupar com diferentes tamanhos de tela, pois só tínhamos o iPhone. Com o lançamento do iPhone 6 / 6s e 6 / 6s Plus, os iPhones agora estão disponíveis em diferentes tamanhos de tela, incluindo 3,5 polegadas, 4 polegadas, 4,7 polegadas e 5,5 polegadas. Um aplicativo bem desenhado deve suportar todos os tamanhos de tela, sem que o cliente sinta diferenças ou dificuldades ao utilizar seu aplicativo.

Neste capítulo veremos uma introdução ao sistema de Auto Layout, da Apple, que permite que os desenvolvedores criem interfaces adaptáveis a qualquer tamanho de tela.

20.1 O Auto Layout

O Auto Layout é um sistema de alinhamento e posicionamento baseado em regras, ou *constraints*. Ele permite que os desenvolvedores criem uma interface de usuário adaptável, que responda adequadamente a alterações no tamanho da tela e na orientação do dispositivo. Existe um certo receio de alguns desenvolvedores, mesmo mais experientes, em lidar com esse recurso. Mas não há mais como fugir dele, pois temos que preparar o nosso aplicativo para rodar em qualquer tamanho de tela.

20.2 Por que Auto Layout é necessário?

Veja o exemplo a seguir, onde o desenvolvedor desenhou um Label e pensou tê-lo colocado no centro da tela, mas ao rodar o App em simuladores com tamanhos de tela diferentes, teve a seguinte surpresa:

Observe que o **Label** está no centro apenas no layout do iPhone 7, onde foi desenhado originalmente pelo desenvolvedor. Mas nos demais ficou desposicionado. Se o dispositivo ainda for rotacionado, colocando sua orientação em modo paisagem, o sintoma irá piorar.

Por que isso ocorre?

Sabemos que os iPhones possuem diferentes tamanhos de tela:

- Para o iPhone 5 / 5s / SE, a tela no modo retrato consiste em 320 pontos (ou 640 pixels) horizontalmente, e 568 pontos (ou 1136 pixels) verticalmente.
- Para o iPhone 6 / 6s / 7, a tela consiste em 375 pontos (ou 750 pixels) horizontalmente, e 667 pontos (ou 1.334 pixels) verticalmente.
- Para o iPhone 6 / 6s / 7 Plus, a tela consiste em 414 pontos (ou 1242 pixels) horizontalmente, e 736 pontos (ou 2208 pixels) verticalmente.
- Para o iPhone 4s, a tela consiste em 320 pontos (ou 640 pixels) e 480 pontos (ou 960 pixels).

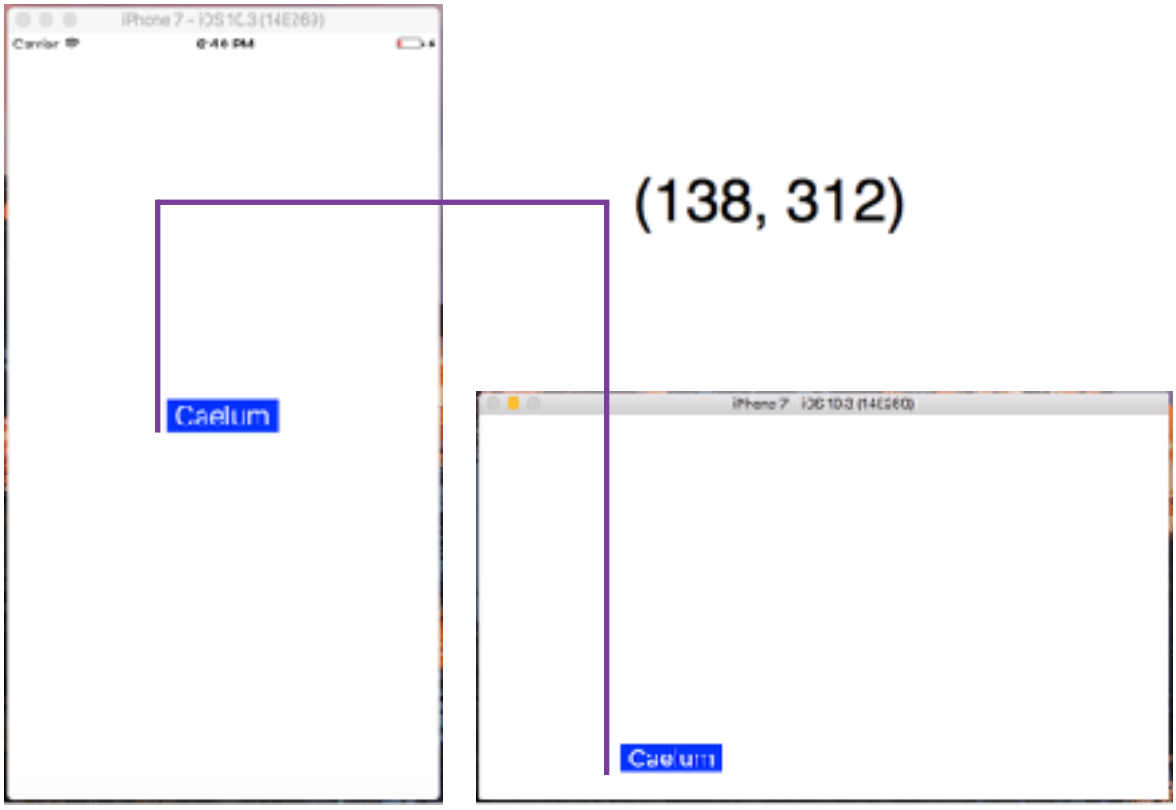
20.3 Pontos ou Pixels?

Em 2007, a Apple apresentou o iPhone original com uma tela de 3,5 polegadas e uma resolução de 320 × 480. Isso significa 320 pixels horizontalmente e 480 pixels verticalmente. A Apple manteve esta resolução de tela com o iPhone 3G e o iPhone 3GS. Para quem estava desenvolvendo um aplicativo naquela época, um ponto correspondia a um pixel, portanto o desenvolvedor nem se preocupava em ajustar o layout. Mais pra frente, a Apple introduziu o iPhone 4, com a tela retina. A resolução daquela tela foi dobrada para 640 × 960 pixels. A partir daquele momento, um ponto correspondia a dois pixels.

Em desenvolvimento iOS trabalhamos em pontos, não em pixels. O sistema de pontos torna a nossa vida fácil, pois mesmo com o aumento de resolução de tela, ainda continuamos trabalhando com a resolução base, em pontos. Como regra geral, o número de pontos é a metade do número de pixels. A exceção são os modelos Plus, que inauguraram uma nova resolução, onde essa relação passa para praticamente 3x.

Quando o Auto Layout não é utilizado, a posição do *Label* definida na *storyboard* fica fixa. No nosso exemplo, nós "cravamos" a origem do quadro do label "Caelum", numa posição fixa (138, 312). Portanto, independente se você estiver usando um simulador de 3,5 polegadas, 4 polegadas ou 5,5 polegadas, o iOS irá desenhar o *Label* sempre nesta mesma posição!

A figura abaixo ilustra como, mesmo no iPhone 7, onde o objeto foi originalmente desenhado, ocorre o deslocamento da mensagem quando o mesmo dispositivo é colocado em modo paisagem.



20.4 EXERCÍCIO - Constatando o nosso problema

1. Em nosso App de Contatos, iremos nos concentrar na tela de Temperatura, para trabalharmos com as questões de Layout.

2. Utilizando o Interface Builder, abra o arquivo `Main.storyboard` e selecione a *View* de **Temperatura**.

3. Inicialmente execute o App utilizando o simulador do iPhone 7. Acesse a *View* de "Visualizar Clima". Como a tela foi construída neste tamanho de device, não deve haver problemas gritantes de *layout*.

4. Execute novamente o app, mas utilizando agora os simuladores do iPhone SE e do iPhone 7 Plus. Observe que os elementos em tela não ficam bem posicionados, ou mesmo centralizados, nestes devices. Adicionalmente, rotacione a tela do simulador (através do atalho *command+seta - esquerda ou direita*), e veja que o problema se agrava.

5. Na realidade, no Xcode 8, há uma forma mais simples de constatar problemas de layout, que é a nova barra de configuração de layout. Com ela, não é necessário rodar o app para visualizar o layout de tela em cada device ou orientação. Basta clicar em cada device da barra para refletir, na View selecionada no storyboard, como ficaria o layout em cada caso. Esse recurso é chamado de *Live Preview*:

6. Explore a barra, altere os devices e a orientação de cada um, e veja que conseguimos constatar os problemas de layout da tela de Temperatura, sem a necessidade de executar o app várias vezes.

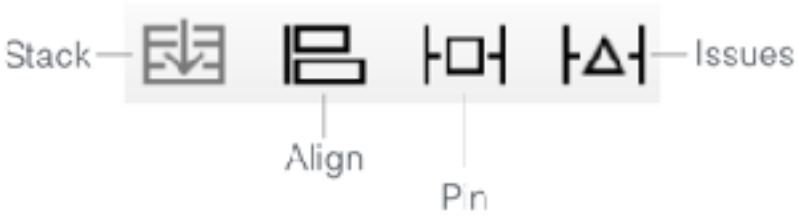
7. Ok, então, sabemos que temos um problema, e nos próximos tópicos iremos trabalhar para que o nosso app fique com a *View* de **Temperatura** adaptável para qualquer tamanho de tela e orientação.

20.5 Os recursos do Xcode para Auto Layout

Existem basicamente duas formas de se trabalhar com Auto Layout no Xcode:

- A barra de Auto Layout
- Efetuar operações de *control-drag* diretamente na tela.

A barra de Auto Layout é composta dos seguintes botões:



Cada botão tem a sua própria função:

- **Align** - Criar regras de alinhamento, como alinhar as margens esquerdas de dois elementos de tela.
- **Pin** - Criar regras de espaçamento, como definir a largura de um controle de interface do usuário.
- **Issues** - Resolver problemas de layout.
- **Stack View** - Empilhar elementos de tela em um único *container*. Essa é uma nova funcionalidade, implementada no Xcode 7.

20.6 Auto Layout é um sistema de regras

O segredo para utilizar com eficiência o Auto Layout é compreender o seu sistema de regras.

Na realidade o sistema prevê dois tipos de regras:
- Regras de Alinhamento
- Regras de Espaçamento

Desta forma, para cada elemento de tela que deverá ser posicionado na tela, você deve ser perguntar:

“O que eu quero fazer com esse elemento?”

É a resposta a essa pergunta é que definirá qual tipo de regra deverá ser utilizada.

20.7 Regras de Alinhamento

Vejamos um exemplo, voltando ao nosso *Label* do início do capítulo. O nosso desejo com ele é posicioná-lo sempre no centro da tela, independente do device que iremos utilizar.

Traduzindo o nosso desejo em uma forma descritiva podemos dizer:

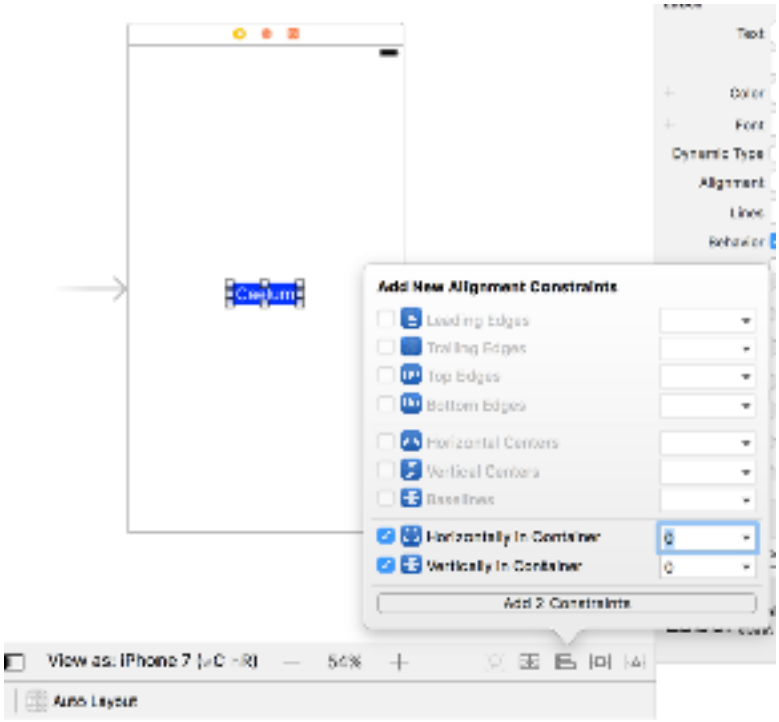
O Label deve ser centralizado horizontalmente e verticalmente, independente da resolução e orientação de tela.

Para atingir esse objetivo, você acha que devemos seguir regras de **alinhamento** ou **espaçamento**?

ALINHAMENTO!

Queremos:
- Centralizar (ou alinhar) horizontalmente
- Centralizar (ou linhar) verticalmente

Portanto, para implementar esse alinhamento na prática, utilizaremos o menu **Align** do Xcode, centralizando a nossa mensagem “Caelum" através do sistema de regras de Auto Layout:



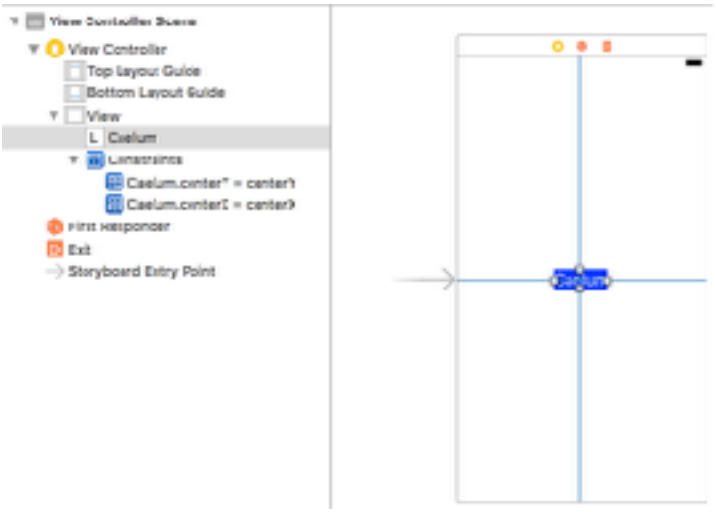
Veja que os seguintes procedimentos foram seguidos:

1. O *Label* com a descrição “Caelum" foi selecionado.

2. O menu Pin da barra de Auto Layout foi selecionado.

3. Duas regras foram selecionadas: *Horizontally in Container* e *Vertically in Container.*

4. Ao clicar no botão **Add 2 Constraints**, as regras de alinhamento serão criadas, o que pode ser confirmado através das linhas que passam a ficar visíveis na View:

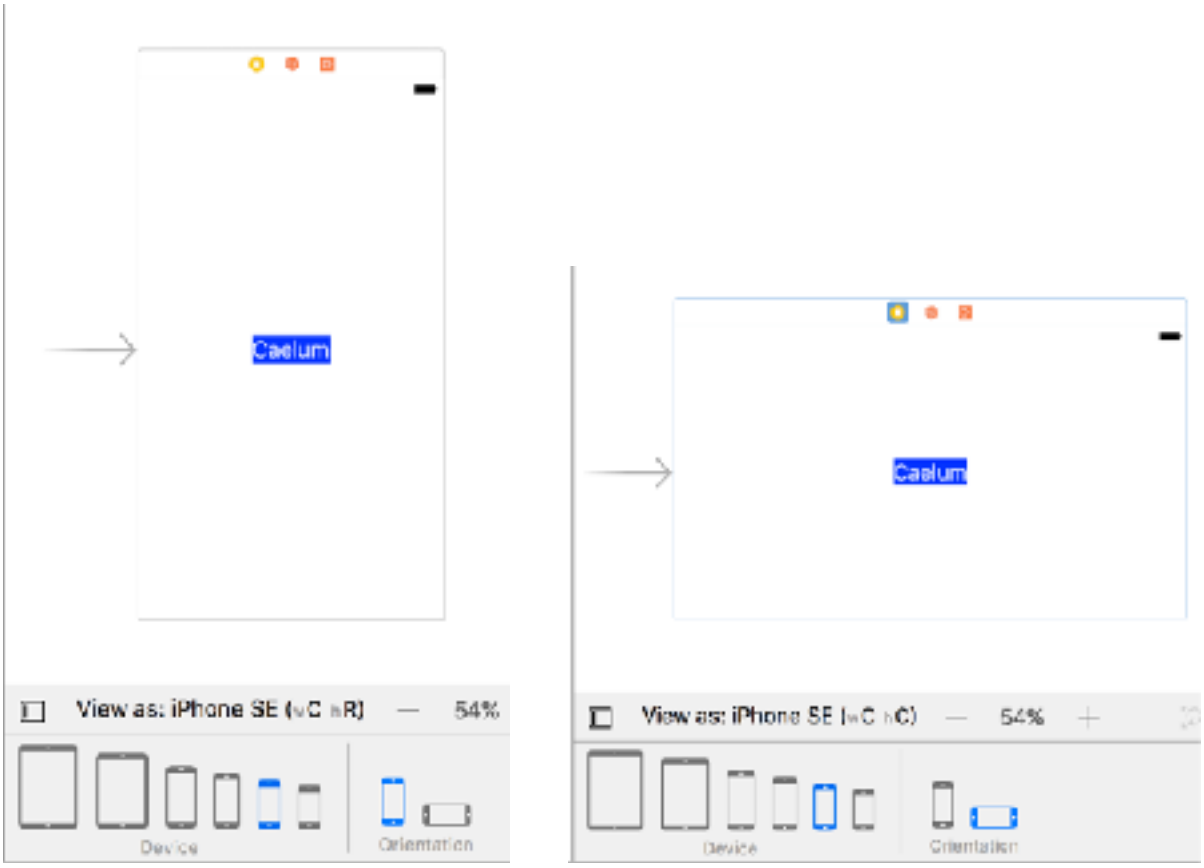


5. Da mesma forma, podemos visualizar as regras criadas na seção de *Document Outline*, à esquerda da *View*, que mostra a hierarquia dos objetos na tela.

6. Agora chegou a hora de tirar a prova. Utilizando a *Live Preview* do Xcode, vamos selecionar todos os devices disponíveis, variando adicionalmente a orientação.

7. No cenário abaixo, podemos verificar que, no iPhone SE, temos o *Label* centralizado na tela, tanto em modo Retrato, como em Paisagem!

8. Problema resolvido!



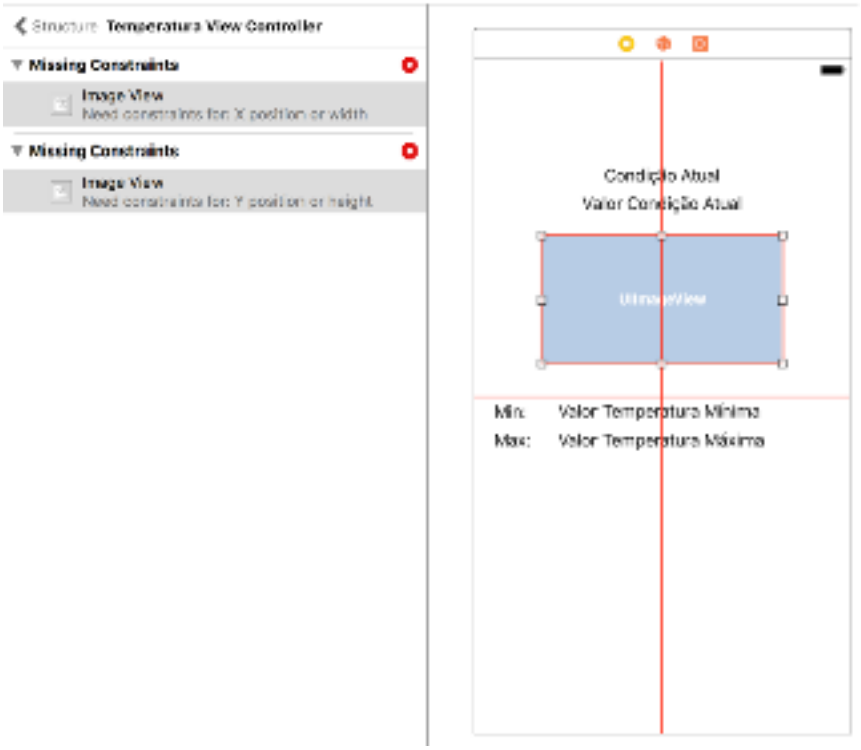
20.8 EXERCÍCIO - Centralizando os elementos da *View* de Temperatura

1. Ao examinar a *View* de **Temperatura**, vemos que os elementos são próximos, e gravitam em torno da figura onde é exibido o status do clima:

2. Sendo assim, iremos centralizar a imagem, horizontalmente e verticalmente, e depois reposicionar os demais elementos, para eles fiquem posicionados próximos à imagem.

3. Selecione a ImageView, abra o menu **Align** da barra de Auto Layout, e marque as opções *Horizontally in Container* e *Vertically in Container,* conforme a figura a seguir:

4. Confirme a configuração através do botão <**Add 2 Constraints**>:

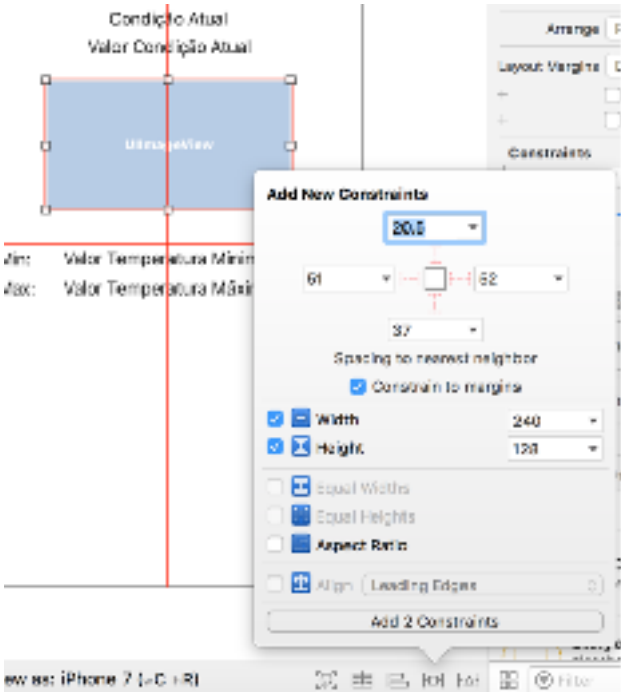


5. Temos um pequeno problema aqui. As linhas de regras (*constraints*) ficaram vermelhas e o objeto não foi centralizado. Isso quer dizer que o Xcode precisa de mais informações para alinhar esse objeto.

6. É importante não entrar em pânico. O Auto Layout segue uma lógica e requer que o desenvolvedor sempre tente manter a calma e entender o que está acontecendo, para atuar e ir corrigindo os problemas.

7. O que está acontecendo é que, como estamos lidando com uma imagem, precisamos, além do alinhamento, definir os limites da imagem, ou seja, sua largura (*width*) e altura (*height*). Esse passo não seria necessário se o objeto fosse um *Label*, pois o *Label* já tem os seus limites definidos. Vamos corrigir então.

8. Selecione novamente a imagem, abra agora o outro menu da barra de Auto Layout, o **Add New Constraints**, e marque as regras **Width** e **Height**:



9. Confirme as configurações através do botão <**Add 2 Constraints**>.

10. Podemos observar agora que as linhas estão azuis! Excelente! O Xcode está nos dizendo que entendeu todas as regras que aplicamos e já refletiu no objeto, ou seja, temos a imagem alinhada e centralizada.

11. Mas e os demais elementos de tela? Precisamos reposicioná-los novamente no entorno da imagem, como fizemos na criação da tela.

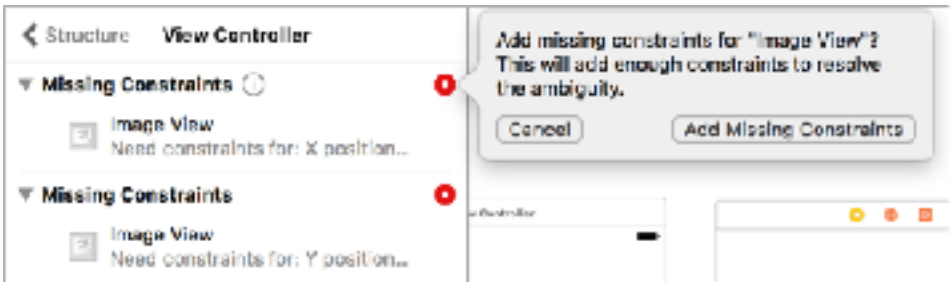
12. Por enquanto, não iremos aplicar outras regras pois ainda precisamos estudar melhor as regras de espaçamento.

13. Manualmente, selecione e reposicione os demais objetos e recupere a configuração original desta tela, que deve ficar parecida com a referência a seguir:

AS CORES DAS LINHAS DE *CONSTRAINTS*
Dizemos que com Auto Layout devemos sempre trabalhar no azul. A cor azul indica que o Xcode entendeu as regras aplicadas. Quando as linhas são na cor laranja, o Xcode está sinalizando que entendeu as regras, mas elas ainda não são suficientes para alinhar ou posicionar o objeto. As linhas vermelhas são o pior caso, pois indica que as regras são insuficientes ou estão em conflito.

Nos casos das linhas laranjas ou vermelhas você deve atuar para acertar a situação. As *constraints* com problemas são indicadas na seção *Document Outline* do Xcode.

Ao clicar sobre o ponto de erro, você pode verificar exatamente quais são os problemas, mas cuidado com as correções automáticas que o Xcode sugere. O ideal é que você sempre tenha domínio e entendimento sobre as regras que está aplicando, para ter total controle quando precisar efetuar alguma manutenção:



20.9 Regras de Espaçamento

Hipoteticamente, vamos considerar o seguinte cenário: O designer de nosso time de desenvolvimento nos entregou um **Asset**, contendo o logo da Caelum, e nos solicitou o seguinte:

Designer:
Em todas as telas em que o logo da empresa for ser exibido, você deve posicioná-lo no canto superior esquerdo

Ok, mas nós, desenvolvedores, precisamos da informação exata para posicionar o objeto, e portanto devemos devolver a seguinte pergunta para o designer:

Desenvolvedor:
Ok, mas a que distância exatamente o logo deve ficar exatamente do topo e da margem esquerda da tela?

E o designer:
Você precisa dessa informação em pixels ou em pontos?

Vimos anteriormente que o sistema de regras do Auto Layout trabalha em pontos, não é mesmo?

Desenvolvedor:
Em pontos, por favor.

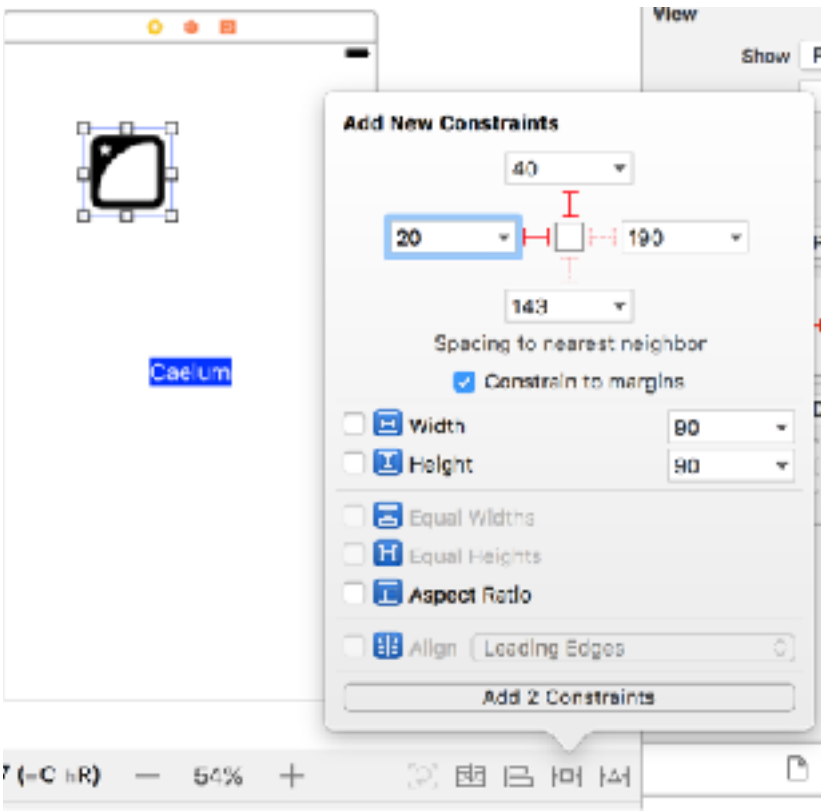
O designer portanto nos passou a seguinte informação:

- **O logo deve ficar a 40 pontos do topo da tela**
- **E a 20 pontos da margem esquerda**

Bem melhor agora, não é mesmo?

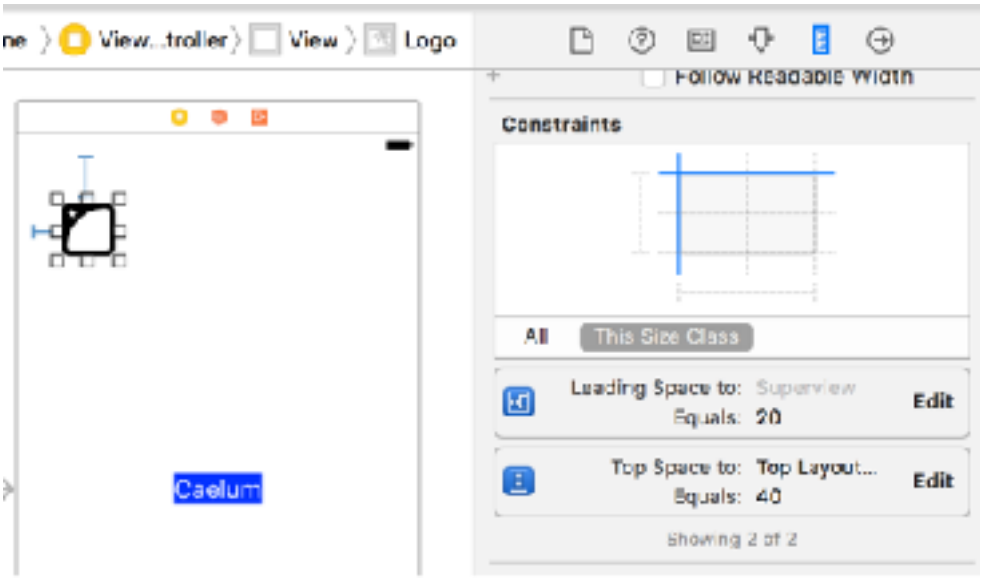
Mas para implementar precisaremos de um outro tipo de regra de layout, que são as regras de espaçamento ou *spacing constraints*. Essas regras estão disponíveis no menu **Pin** da barra de Auto Layout do Xcode.

No exemplo a seguir, iremos posicionar o logo da empresa exatamente na posição indicada pelo designer. Observe que iremos adicionar 2 *constraints* para posicionar o objeto na distância desejada, a 40 pontos do topo e 20 pontos da margem esquerda (são os regras que estão assinaladas com um faixa vermelha).



Após a aplicação das regras, o objeto ficará posicionado exatamente na posição desejada, para qualquer tamanho de tela ou orientação.

Veja que é possível consultar as *constraints* aplicadas a um objeto através da guia **Size Inspector** da janela de propriedades do Xcode:



20.10 EXERCÍCIO - Posicionando os demais elementos da tela

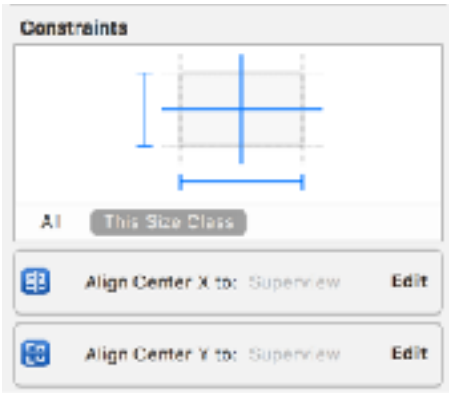
1. Utilizando agora uma combinação das regras de alinhamento com as regras de espaçamento, vamos posicionar os demais elementos da tela de **Temperatura**.

2. Iremos começar pelos elementos mais simples, que são os 2 *Labels* que estão acima da imagem:

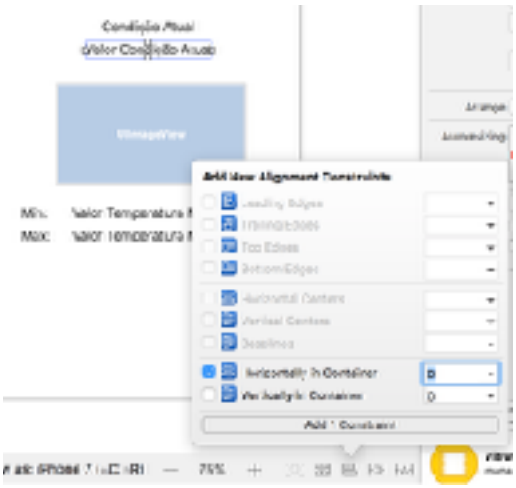
3. Observando esses elementos podemos verificar que eles se encontram naturalmente no centro da tela, horizontalmente, ou seja, no eixo X da tela.

4. A nossa estratégia será utilizar a imagem do clima como ponto de origem para o posicionamento dos demais elementos de tela. Como a imagem já está posicionada no centro da tela, os outros elementos serão “pendurados” uns nos outros, irradiando os posicionamentos a partir do centro, onde está a imagem.

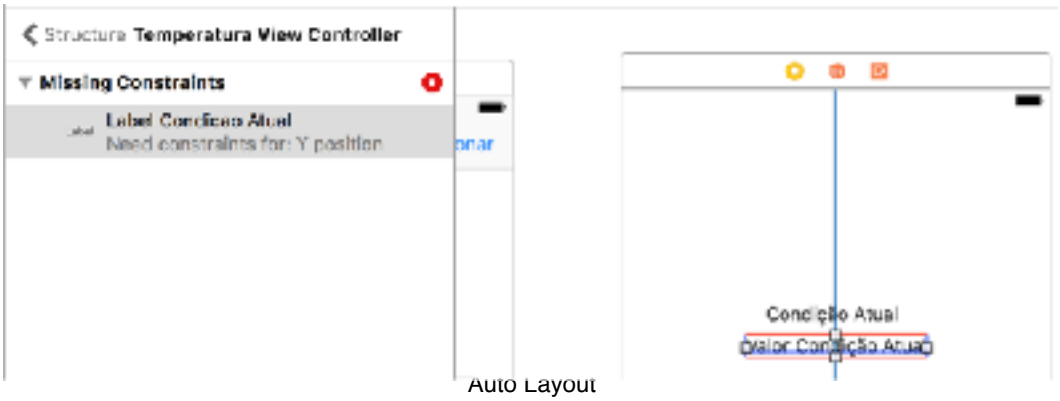
LÓGICA MATEMÁTICA
Fazendo um paralelo com os eixos cartesianos da Matemática, o alinhamento Horizontal equivale ao eixo X e o alinhamento Vertical equivale ao eixo Y:



5. Vamos começar pelo 1º Label, que contém a descrição "**Valor Condição Atual**", pois é o objeto que, na parte superior da tela, se encontra mais próximo da imagem. Inicialmente iremos posicioná-lo no centro da tela, horizontalmente (eixo X):

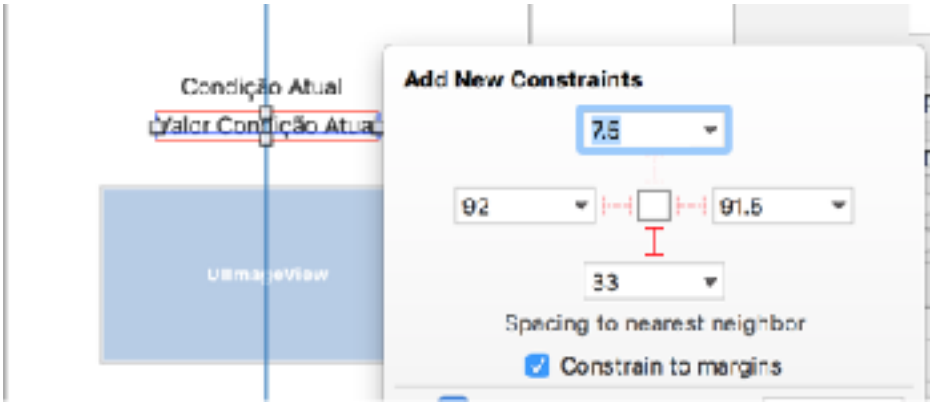


6. Ok, mas o Xcode está nos indicando que apenas essa regra não é suficiente para posicionar o objeto. Ao clicar sobre o indicador de problemas com *constraints*, veremos que é necessária uma outra regra, para posicionar o elemento verticalmente, ou seja, no eixo Y:



7. Neste momento daremos início à nossa estratégia de utilizar a imagem para ancorar os demais elementos. Se criarmos uma regra de espaçamento entre o *label* e a imagem, estaremos de quebra criando uma regra de posicionamento vertical, que é o que precisamos!

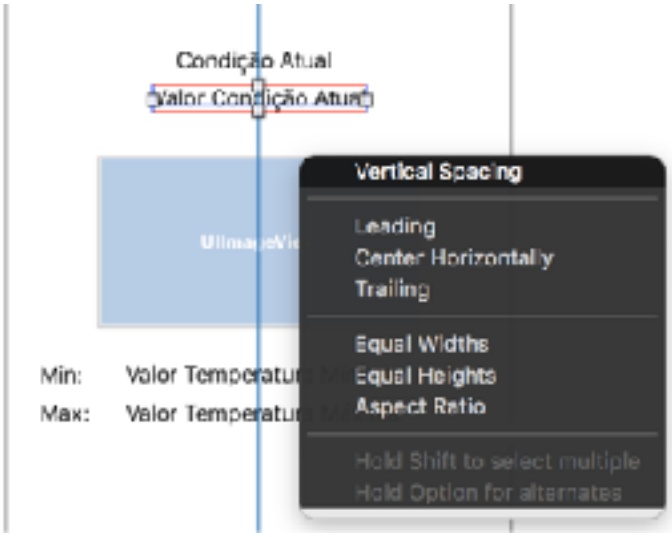
8. Sabemos que as regras de posicionamento são determinadas pelo menu **Pin** da barra de Auto Layout. Basta então selecionar o *Label* novamente, abrir o menu **Pin**, e determinar a regra. Como a imagem está abaixo do *label*, marcaremos a *constraint* inferior. Observe que a imagem, que é o vizinho mais próximo abaixo do label, está a 33 pontos deste. Caso quiséssemos definir o numero de pontos exato, bastaria alterar esse valor:



9. Sucesso! Temos agora as regras necessárias, conforme mostra o Xcode (tudo azul!):

DEFININDO CONSTRAINTS COM DRAG-AND-DROP
Existe uma forma alternativa de especificar *constraints*, diretamente na tela. Por exemplo, a regra de espaçamento definida do *Label* para a imagem poderia ter sido efetuada através de uma operação *drag-and-drop*, selecionando o *Label*, mantendo a tecla **control** pressionada, e arrastando o cursor até a imagem. Ao soltar o cursor, o Xcode exibe um menu sugerindo algumas possibilidades de *constraints*. No nosso caso, seria a opção **Vertical Spacing**.

Depende do desenvolvedor escolher a forma a qual se sinta mais confortável de trabalhar: o menu **Pin** ou as operações efetuadas diretamente na tela.



10. Iremos seguir a mesma linha para o Label superior, que contém a mensagem "**Condição Atual**". Alinhar horizontalmente (no eixo Y) e ancorar no *Label* inferior (no eixo Y).

11. Utilizando o menu **Align**, alinhe horizontalmente o *Label*. E através do menu **Pin**, crie uma regra de espaçamento para o *label* inferior. Ao final das operações, devemos ter as seguintes regras:

12. Caso queira verificar se tudo o que fizemos até agora está ok, verifique o comportamento a tela em diferentes dispositivos e orientações. Se tudo foi feito conforme o esperado, a parte de cima da tela deve estar bem resolvida, restando acertar agora a parte abaixo da imagem.

13. O nosso problema é um pouco mais complicado aqui, pois temos 2 conjuntos de *labels*, que estão lado a lado. A estratégia que utilizamos para resolver os objetos superiores não será suficiente. Mas a Apple criou um recurso para facilitar as coisas nestes casos.

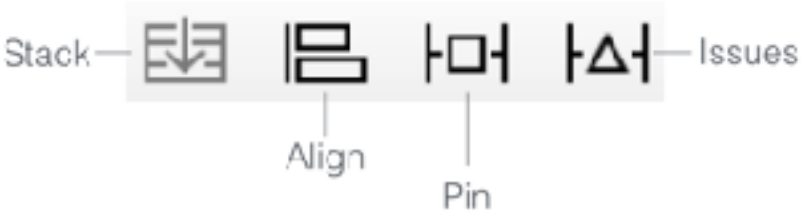
14. Vamos portanto dar uma paradinha para conhecermos esse recurso.

20.11 O *Stack View*

Antes do *Stack View*, ou simplesmente *Stack*, os desenvolvedores rebolavam quando tinham que criar regras de Auto Layout para objetos localizados lado a lado, tanto horizontalmente como verticalmente:

Posicionar os objetos entre eles e também com relação às bordas era um trabalho bem complicado, mesmo para desenvolvedores mais experientes. O que eles acabavam fazendo era criar um *View* como um *Container*, e posicionavam os objetos lá dentro, diminuindo assim o número de *constraints* necessárias.

A Apple estava acompanhando esse processo, e, no lançamento do Xcode 7, junto ao iOS 9, a interface de desenvolvimento veio com esse recurso:

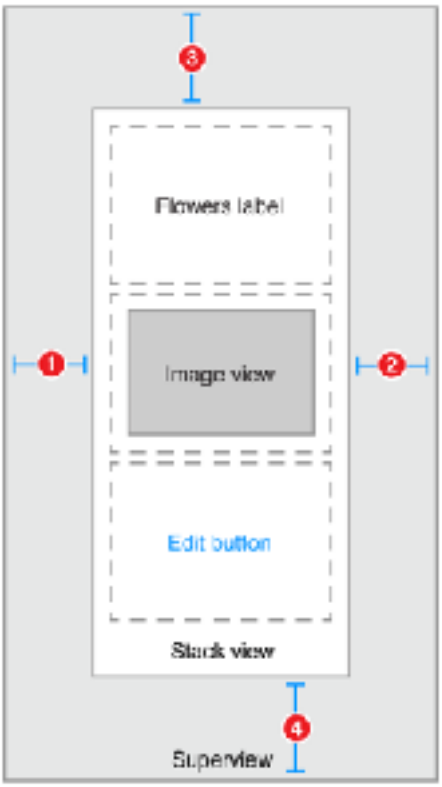


O interessante é que esse novo recurso passou despercebido por muitos profissionais, mas é uma funcionalidade muito útil.

Vejamos um exemplo, extraído do Guide de Auto Layout da própria Apple. O desafio é posicionar 3 elementos visuais na vertical, um abaixo do outro - um *Label*, uma *Imagem* e um *Botão*.

Se o *Stack* não existisse, várias regras de alinhamento e posicionamento teriam que ser criadas para posicionar os elementos entre eles, e também para posicioná-los em relação às bordas e margens. O fato de um dos objetos ser uma imagem iria complicar a situação, pois sabemos que um imagem pode ter tamanho variável.

Ao empilhar esses objetos num *Stack*, no entanto, são necessárias apenas 4 regras, para posicionar o *Stack* em relação às margens, como vemos a seguir. Não é necessário criar regras para os elementos internos, pois eles são “abraçados" pelo *Stack:*



No exercício a seguir, veremos como o *Stack View* irá ajudar a resolver o nosso problema.

AUTO LAYOUT GUIDE
Para quem deseja se aprofundar no assunto, segue link reduzido do Guide da própria Apple:
https://goo.gl/CTu5TM

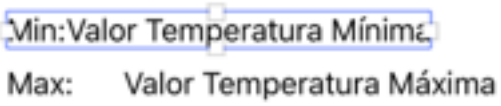
20.12 EXERCÍCIO - Posicionando elementos lado a lado com o *Stack View*

1. Voltando ao nosso problema, precisamos resolver as questões de layout dos campos inferiores de nossa tela, que mostram os valores mínimo e máximo da temperatura. A nossa dificuldade é que são dois elementos visuais posicionados lado a lado.

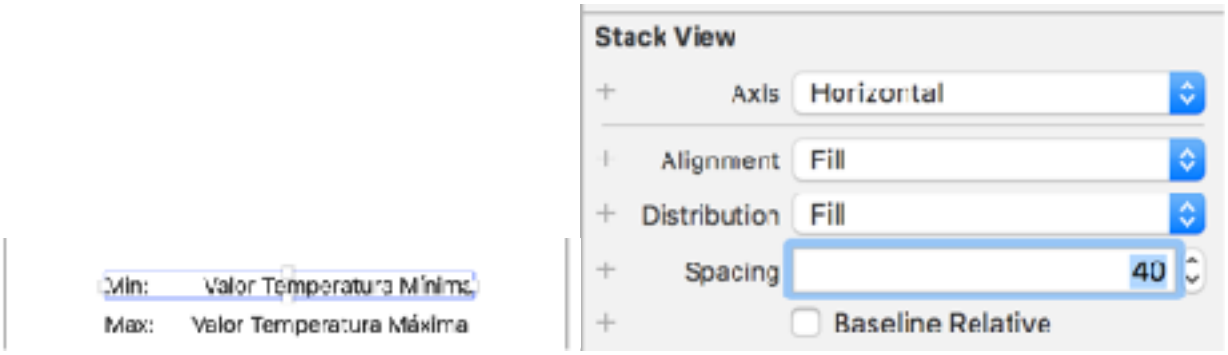
2. O Stack irá facilitar a nossa vida. Vamos lá.

3. Inicialmente, selecione e reposicione todos os elementos, para que fiquem centralizados na tela, seguindo assim a nossa estratégia inicial:

4. Vamos começar pelos *labels* superiores. Selecione ambos os elementos e clique no *Stack*. Você verá que o *Stack* irá abraçar e juntar os *labels*:



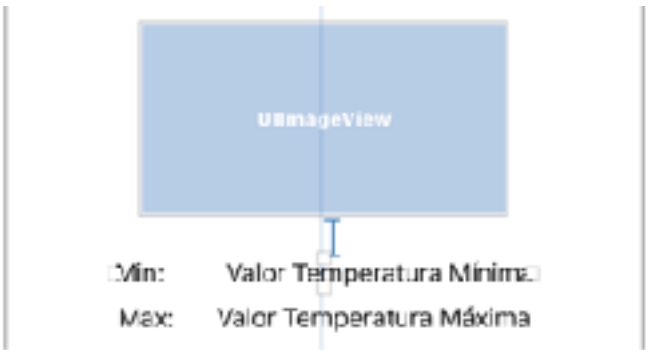
5. Inicialmente iremos “separar” os elementos. Para isso existe uma propriedade específica no *Stack*, para determinar esse espaçamento, que é a **Spacing**. Como o Stack selecionado, altere o valor da propriedade para **40**, e veja que os elementos ficaram separados por 40 pontos:



6. Neste momento, se observarmos na *Document Outline*, veremos que o Xcode está apontando problemas de *constraints*. Vamos consertá-los. Para isso, basta seguir a nossa estratégia inicial de alinhar na horizontal e ancorar no objeto mais próximo. Iremos fazer isso com o próprio *Stack View*.

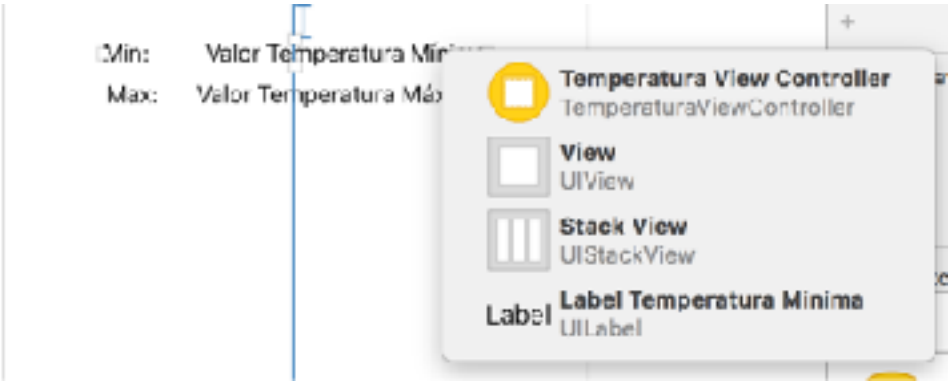
7. Certifique-se que o Stock View este selecionado. Em seguida aplique as seguintes constraints:
   - Alinhamento Horizontal
   - Espaçamento em relação à imagem que está acima do *Stack*.

8. Após a aplicação das regras, os problemas de constraints devem ter sido resolvidos e os nossos labéus estarão devidamente posicionados:



9. Perceberam o benefício do *Stack View*? Criamos regras apenas para o próprio *Stack*. Para os elementos empilhados dentro dele não é necessário!

ATALHO PARA O STACK
Pode ser um pouco complicado selecionar um Stock View, quando ele já contém itens empilhados ou está muito próximo de outros elementos.
Uma boa forma de selecionar um *stack* é segurar a tela **shift** e, na sequência, pressionar o botão direito do mouse ou trackpad. Será exibida uma tecla de atalho onde se pode escolher o que deve ser selecionado, incluindo, é claro, o próprio *Stack View*:



10. Agora é com vocês. Repitam o procedimento com os labels restantes, de temperatura máxima, para empilhá-los em um Stack e, na sequência, aplicar as regras de *constraints* sobre o Stack. Na prática, vocês terão que repetir os procedimentos 4 a 8.

11. Sucesso total! Todos os elementos visuais de nossa tela de temperatura estão devidamente adaptáveis para qualquer tamanho de tela e orientação.

12. Faça todos os testes necessários para verificar o bom funcionamento do layout. Utilize a *Live Preview* e rode pelos simulares para certificar-se que tudo está OK e que nenhuma regra se perdeu pelo caminho.

UM NOVO DESAFIO
Como desafio a ser encarado extra curso, tente trabalhar sobre a *View* de **Cadastro**, para tornar seu layout adaptável também. Nesta tela haverão alguns desafios adicionais, como um botão e vários elementos lado a lado, mas em termos gerais as técnicas que desenvolvemos em aula serão suficientes para resolver todas as situações.
Talvez uma estratégia interessante seja posicionar os elementos a partir do topo da tela, e não do centro, como fizemos com a tela de temperatura.
Não se pode esquecer da questão da usabilidade do aplicativo. Se, por exemplo, o modo paisagem não trouxer nenhuma experiência adicional ao cliente é melhor restringir o uso do App ao modo retrato.
BOA SORTE!