



POLITÉCNICA

ETSIT
UPM

dit
UPM

Desarrollo de Apps para iOS Auto Layout

IWEB,LSWC 2013-2014
Santiago Pavón

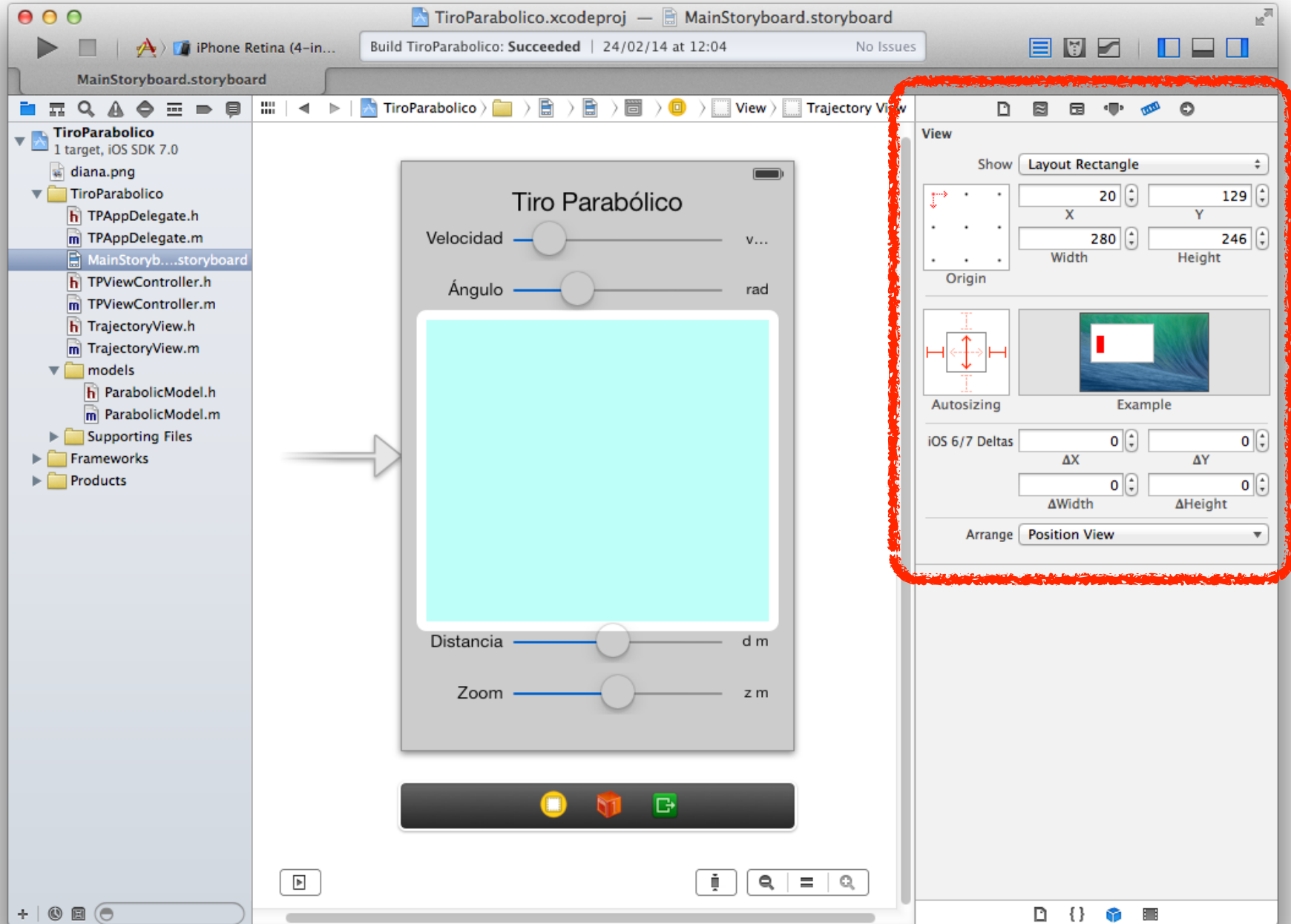
ver: 2014.03.02

Gestión del Layout de las Views

- Objetivo:
 - Decidir la **posición** y el **tamaño** de las views.
 - Reajustarse para soportar diferentes dispositivos, rotación del terminal, ...
- Opciones:
 - **Springs y Structs**
 - Proporcionar explícitamente la posición y el tamaño de cada view.
 - Configurar máscaras de reposicionamiento y reescalado para ajustarse a los cambios de tamaño de la superview.
 - **Autolayout**
 - Proporcionar condiciones (restricciones/ constraints) a satisfacer entre las views.
 - Ejemplo:
 - Que el final de una view y el principio de otra view estén separados 10 pixeles.
 - Que el ancho de dos views sea igual.
 - Del conjunto total de restricciones debe poder calcularse cuál es la posición y tamaño de todas las views.
 - No puede haber ni ambigüedades ni conflictos.



Springs y Structs



Auto Layout

Prohibido el uso de .frame

- Al usar Auto Layout ya no puede usarse la propiedad **frame** de las views.
- El valor de la propiedad frame lo calculará el sistema de Auto-Layout en función de las restricciones definidas.

Las Restricciones son Fórmulas

- Cada restricción es implementada siguiendo la siguiente fórmula:

$$\text{view1_atributoA} = K * \text{view2_atributoB} + C$$

- *El valor del atributo A de la view 1 es igual al valor del atributo B de la vista 2 multiplicado por la constante K y sumando la constante C.*
 - En vez de = también pueden usarse las relaciones <= o >=.
- Ejemplo:

```
self.buttonOK.width = 2*self.titleLabel.height + 50
```

- *La altura del botón Ok es el doble que el ancho de la etiqueta del título más 50.*

Crear las Restricciones

- Programáticamente:

```
NSLayoutConstraint *constraint = [NSLayoutConstraint  
    constraintWithItem: self.buttonOK  
        attribute: NSLayoutConstraintAttributeWidth  
        relatedBy: NSLayoutConstraintRelationEqual  
        toItem: self.titleLabel  
        attribute: NSLayoutConstraintAttributeHeight  
        multiplier: 2.0f  
        constant: 50.0f];  
  
[self.view addConstraint:constraint];
```


- Con lenguaje de formato visual:

Borde izquierdo

Borde derecho

|-[**boton1**]-(>=8)-[**boton2(120)**]-[**boton3(30)**]-|

Separación

Separación >= 8

Ancho 120

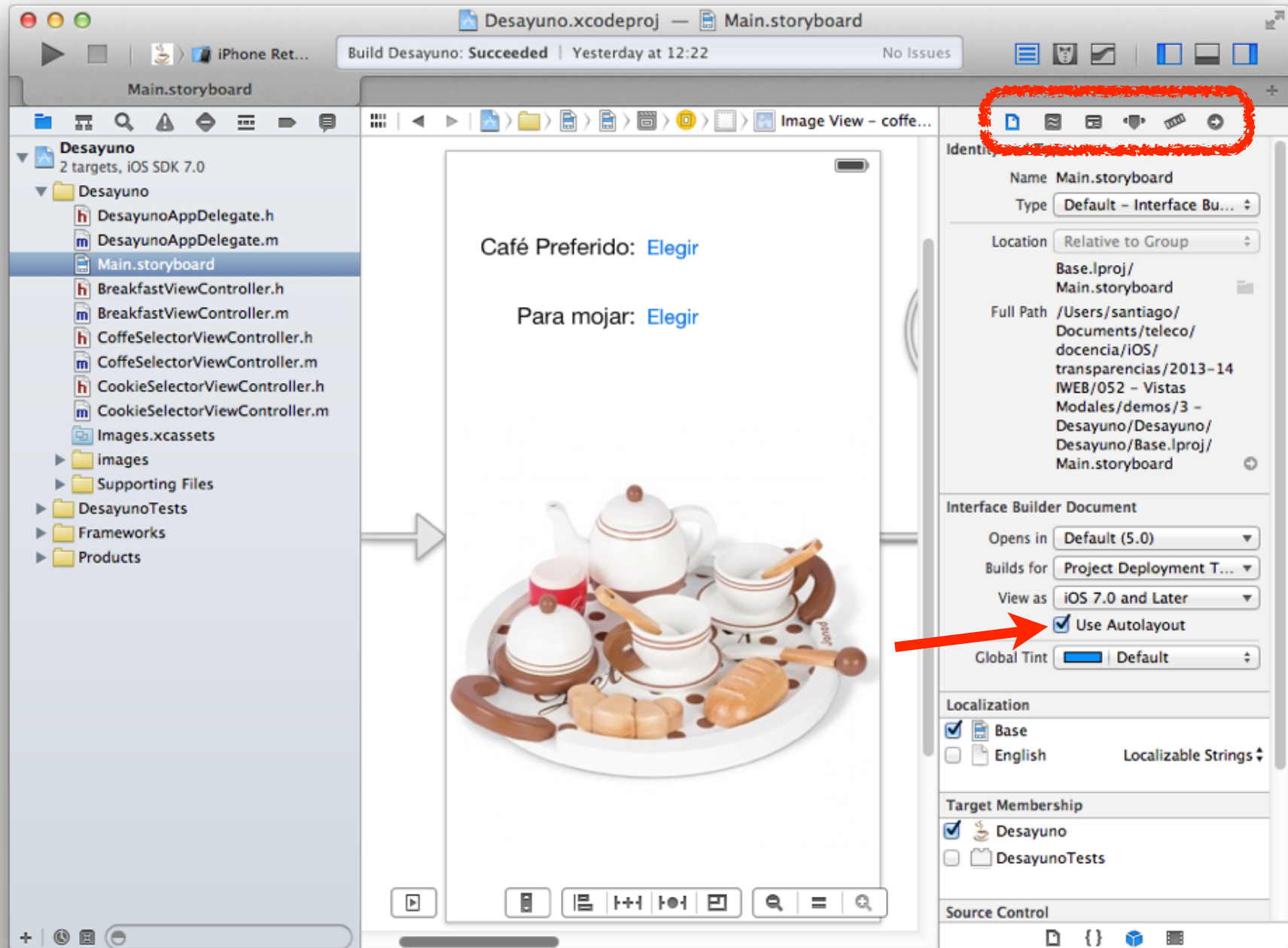
Ancho 30

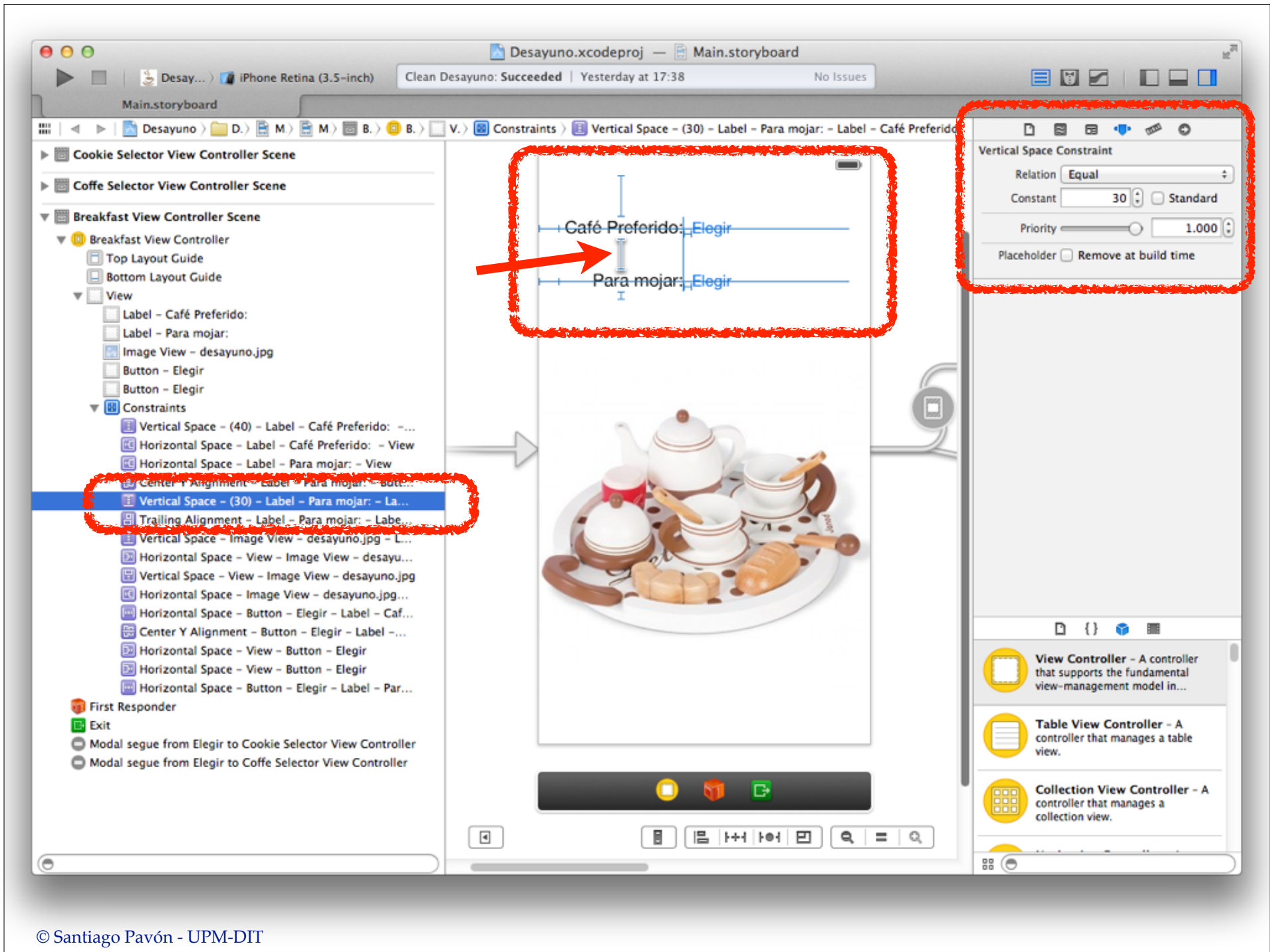
```
NSDictionary *dic = @{@"boton1":boton1,
                      @"boton2":boton2,
                      @"boton3":boton3};
```

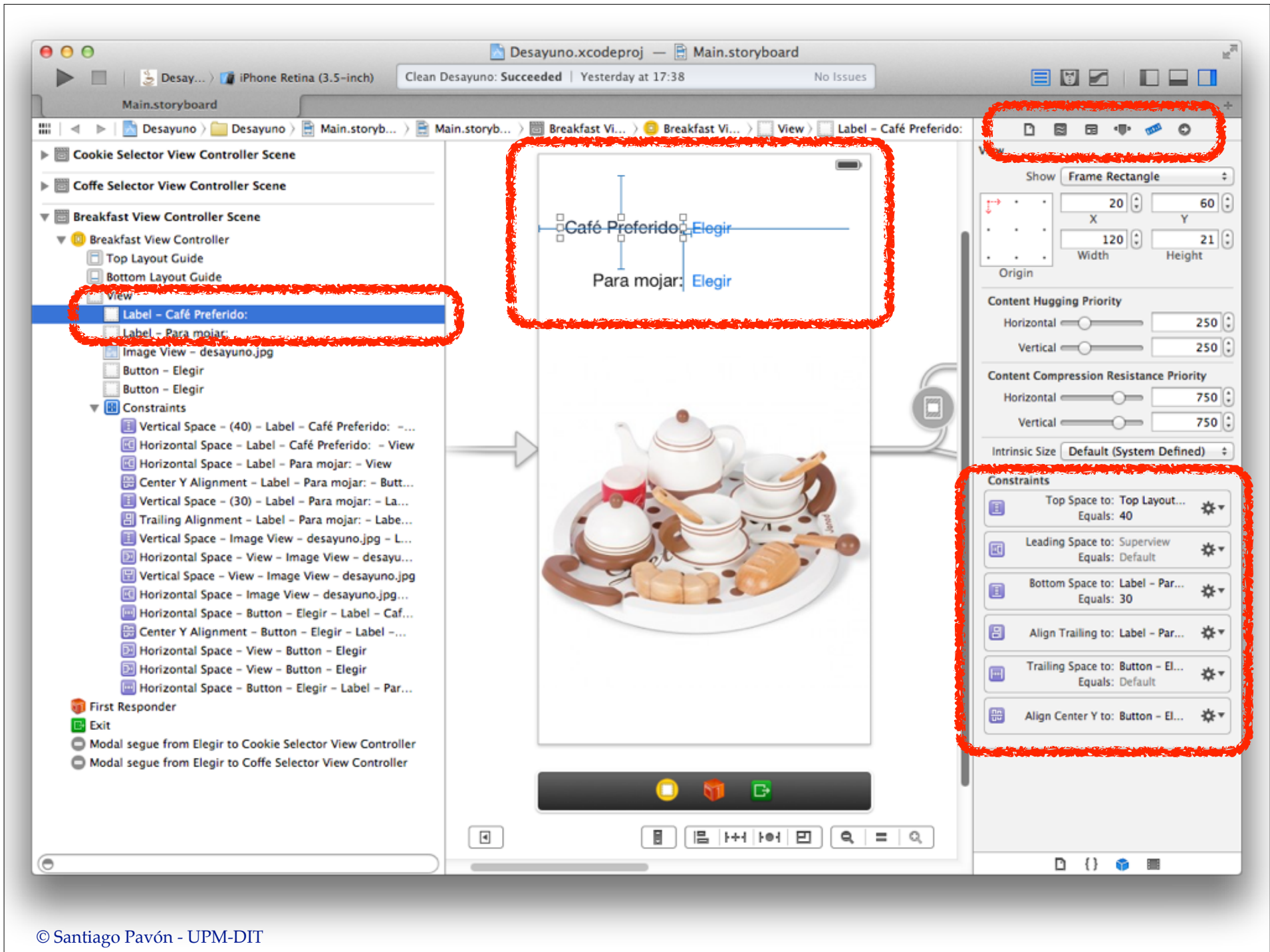
```
NSArray *constraints = [NSLayoutConstraint
    constraintsWithVisualFormat:@"|-[boton1]-(>=8)-[boton2(120)]-[boton3(30)]-|"
    options:NSLayoutFormatAlignAllTop
    metrics:nil
    views:dic];
```

```
[self.view addConstraints:constraints];
```

• Interactivamente con el Interface Builder:







The image shows the Xcode storyboard editor for a mobile application. The central canvas displays a storyboard with a coffee-themed UI. At the top, there is a label "Para mojar, Elegir" in blue text. Below it is a large image of a coffee set on a tray. At the bottom, there is a dark navigation bar with three buttons: a yellow one, a red one (highlighted by a red arrow), and a green one. The left sidebar shows the storyboard's hierarchy, including labels, image views, and buttons, along with a list of constraints. The right sidebar shows the 'Content Hugging Priority' and 'Constraints' settings. The 'Add New Alignment Constraints' menu is open, showing options like 'Horizontal Center in Container' and 'Vertical Center in Container'.

Add New Alignment Constraints

- Leading Edges
- Trailing Edges
- Top Edges
- Bottom Edges
- Horizontal Centers
- Vertical Centers
- Baselines
- Horizontal Center in Container 0
- Vertical Center in Container 0

Update Frames: None

Add Constraints

The screenshot displays the Xcode storyboard editor. On the left, the 'Constraints' panel lists various constraints for the storyboard elements. The central canvas shows a coffee set image and a bottom navigation bar with three buttons. A red arrow points to the 'Add New Constraints' dialog box, which is open over the navigation bar. The dialog shows a vertical spacing constraint of 40, with 20 on the left and 8 on the right. Other settings include Width: 120, Height: 21, Spacing to nearest neighbor, and various alignment and update frame options.

Horizontal

Vertical

Content Compression Resistance Priority

Horizontal

Vertical

Intrinsic Size Default (System Defined)

Constraints

- Top Space to: Top Layout... Equals: 40
- Leading Space to: Superview Equals: Default
- Bottom Space to: Label - Par... Equals: 30
- Align Trailing to: Label - Par...
- Trailing Space to: Button - El... Equals: Default
- Align Center Y to: Button - El...

Add New Constraints

Spacing to nearest neighbor

Width

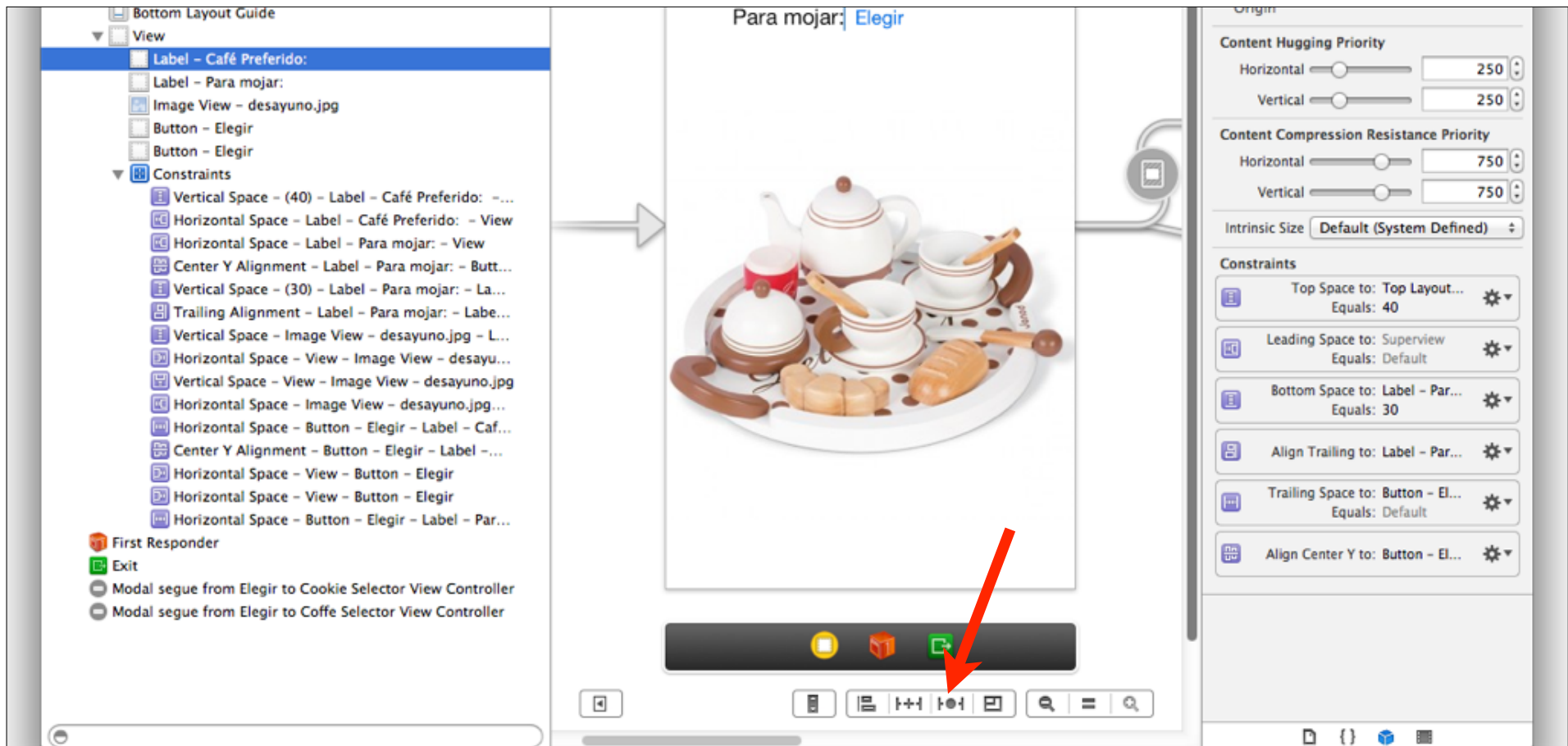
Height

Equal Widths

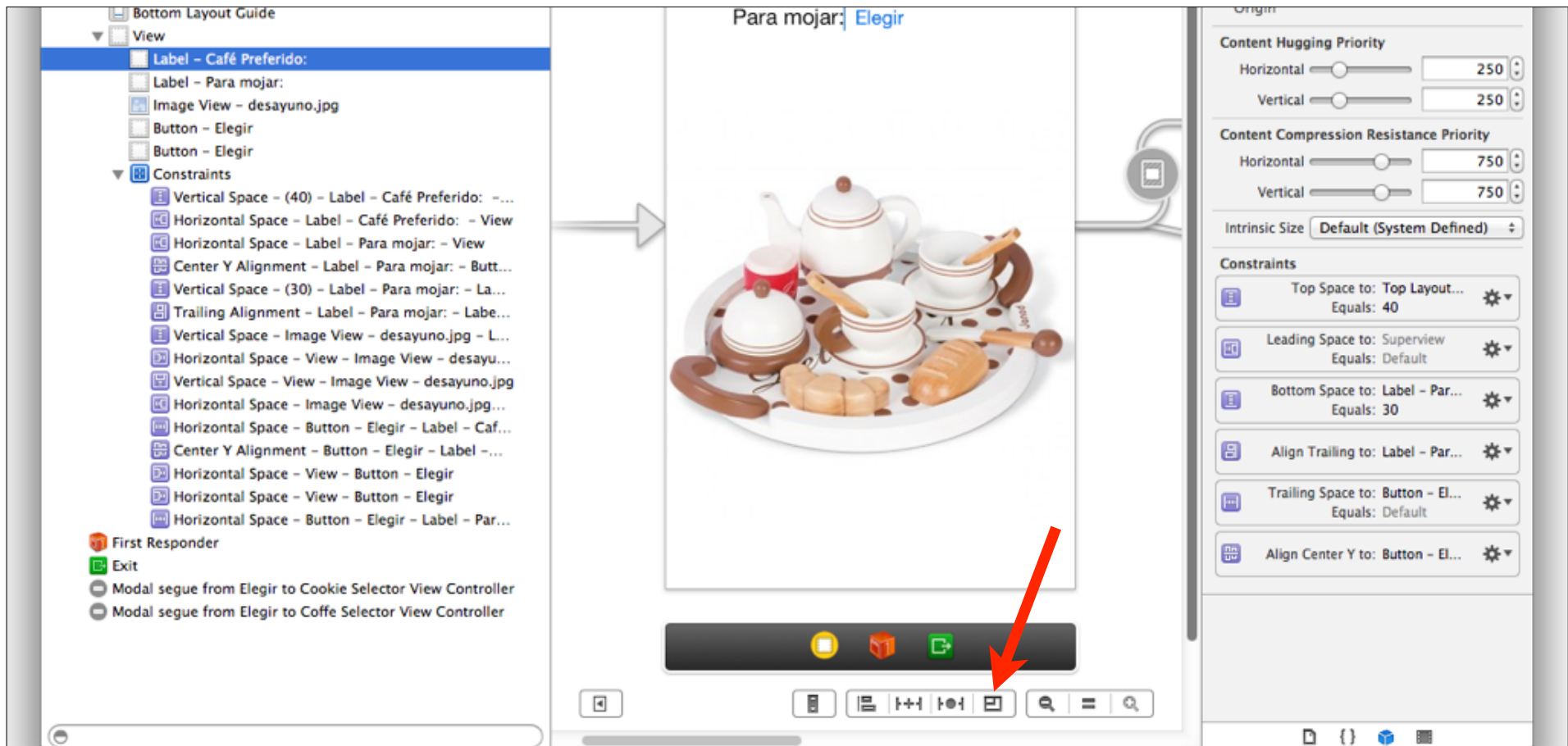
Equal Heights

Align

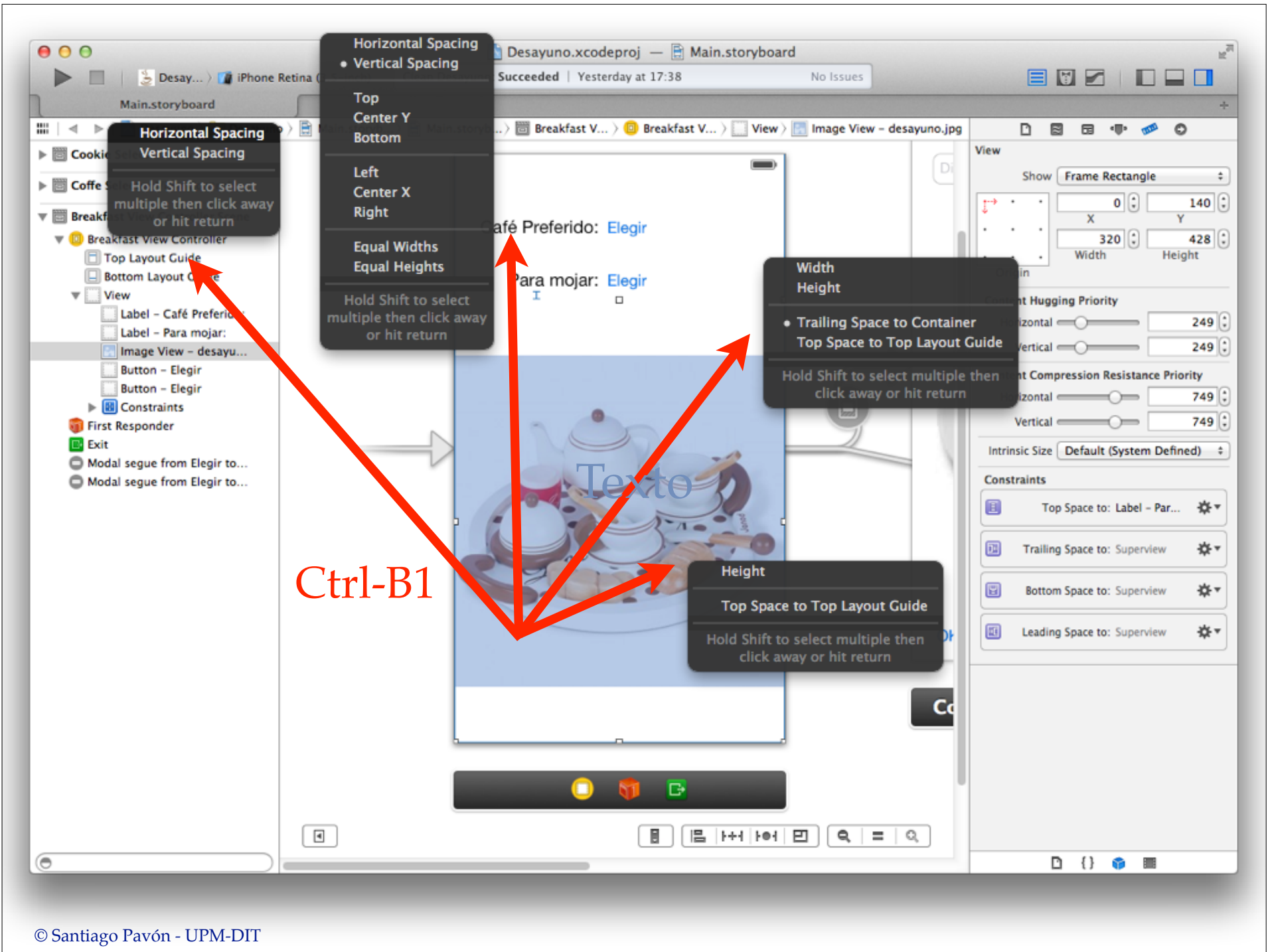
Update Frames



- Update Frames ⌘⇧=
 - Update Constraints ⇧⇧=
 - Add Missing Constraints
 - Reset to Suggested Constraints ⌘⇧⇧=
 - Clear Constraints
-
- Update All Frames in Breakfast View Controller
 - Update All Constraints in Breakfast View Controller
 - Add Missing Constraints in Breakfast View Controller
 - Reset to Suggested Constraints in Breakfast View Controller
 - Clear All Constraints in Breakfast View Controller



When Resizing Views Apply Constraints to...
 Siblings and Ancestors
 Descendants



:-(iOS 6 - iOS 7 :-)

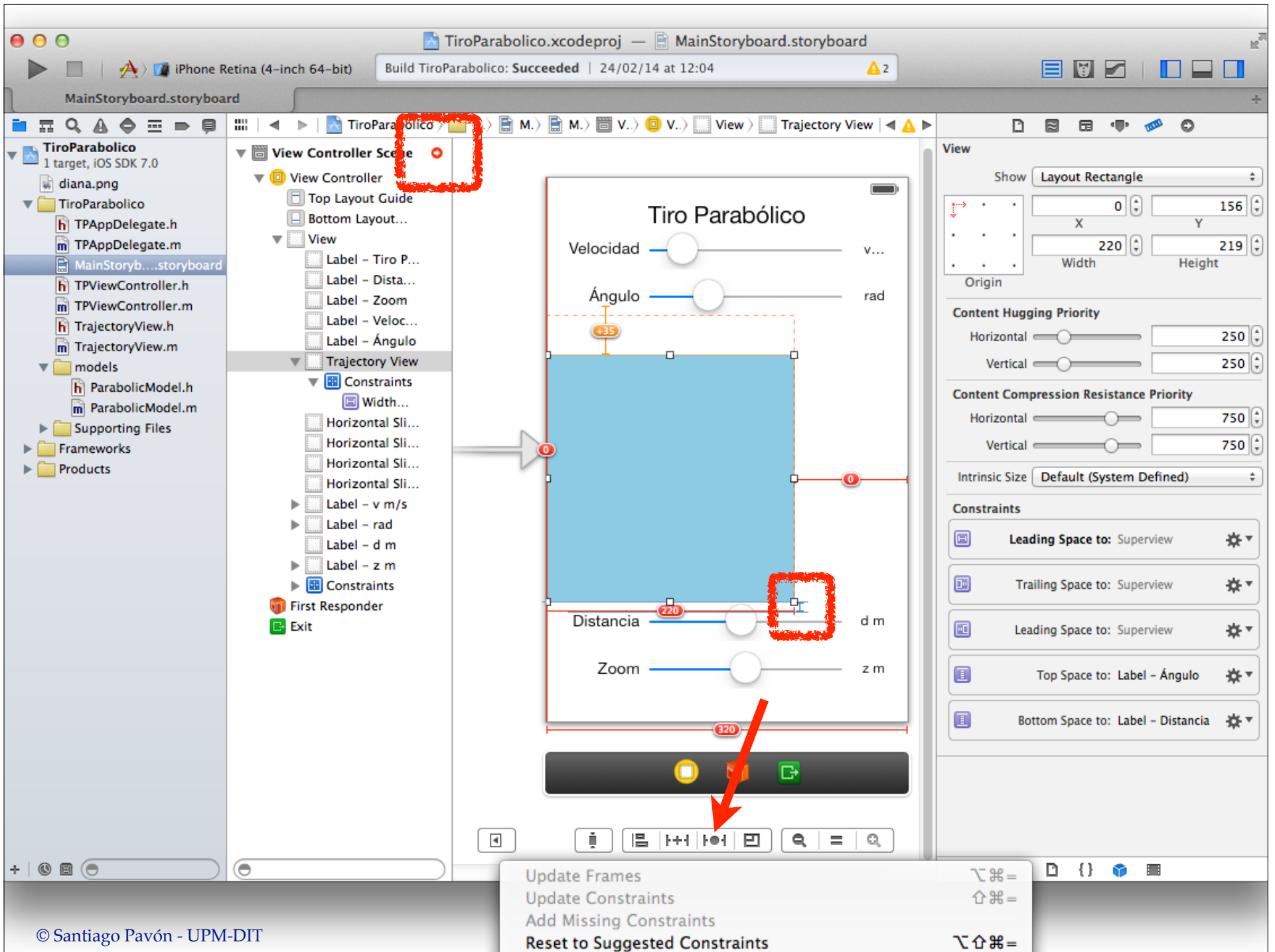
- AutoLayout se introdujo en iOS 6.
- En iOS 6 era muy desagradable de usar interactivamente con Interface Builder.
 - El sistema continuamente está metiendo sus propias restricciones para eliminar ambigüedades.
 - Ante el menor cambio se rehacen todas las restricciones, incluidas las introducidas por el desarrollador. Esto era **inaceptable**.
- En iOS 7 se mejora el interface de uso con Interface Builder
 - Ahora si puede usarse cómodamente.
 - No se modifican automáticamente las restricciones medidas por el desarrollador al mover alguna view.
 - Pero cuando el desarrollador lo desee, puede solicitar que se generen automáticamente las restricciones de algunas views.
 - Con Ctrl-Arrastrar se pueden crear restricciones muy fácilmente.
 - Se han añadido controles para resolver automáticamente problemas comunes de autolayout.
 - Añadir restricciones olvidadas, resetearlas, borrarlas, actualizar los frames de las views para que se ajusten a las restricciones, etc...
 - ...

Completitud

- El conjunto de restricciones debe permitir calcular la posición y el tamaño de todas las views.
- Las restricciones que faltén se crean automáticamente al compilar.
 - se añaden restricciones fijando el tamaño y posición de las views al valor visualizado en el editor.
- Mientras se está editando, el sistema no añade restricciones automáticamente.
- En cuanto se ponga una restricción en una view, hay que poner todas las que se necesiten para eliminar las ambigüedades.

Conflictos

- El editor Interface Builder muestra las restricciones existentes con líneas de diferentes colores.
 - Las **líneas azules** representan restricciones correctas.
 - Las **líneas rojas** representan conflictos con otras restricciones.
 - No pueden cumplirse las restricciones introducidas. No hay solución.
 - Las **líneas naranjas** representan conflictos entre las restricciones introducidas y la colocación de las views en la pantalla del editor.
 - Se arregla recolocando adecuadamente las views en la pantalla.
- Cuando hay conflictos, el **Document Outline** muestra una flecha naranja o roja junto al nombre de cada escena.
 - Estas flechas abren un panel que describe los conflictos existentes, y su posible solución.
- Para solucionar los problemas existentes con las restricciones se puede:
 - Editar manualmente las restricciones problemáticas.
 - Usar las opciones del menú **Resolve Auto Layout Issues**.
 - Usar las opciones que se ofrecen desde el **Document Outline**.



Tamaño Intrínseco del Contenido

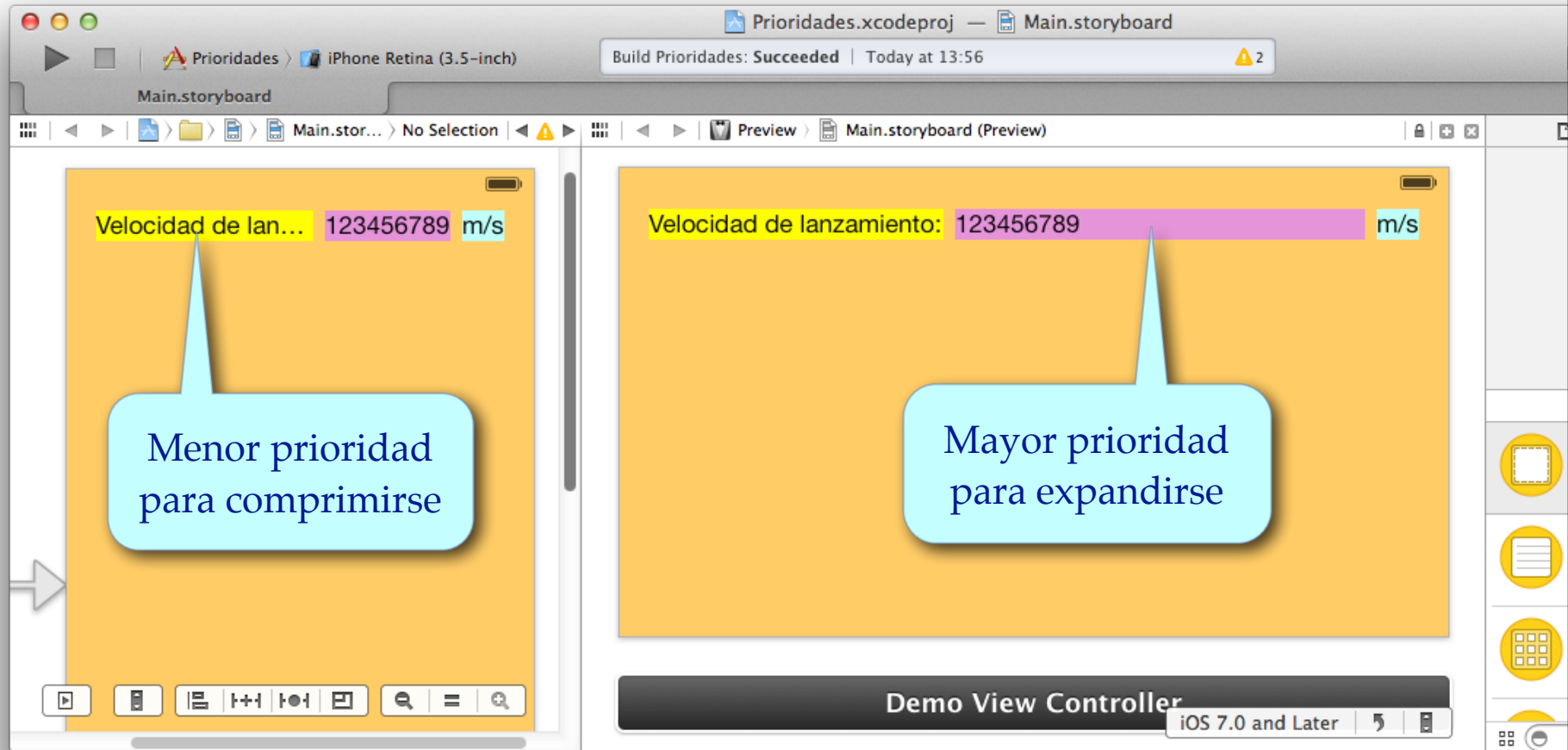
- Es el tamaño exacto que requieren tener algunas views (UILabel, UIButton, UIImageView, ...) para que quepa su contenido.
- Para redimensionar una view a su tamaño intrínseco, seleccionarla y ejecutar:

`Menú Editor > Size to Fit Content`

Prioridades

- Las restricciones tiene prioridades.
 - Prioridad por defecto = 1000.
- Las views que tienen un tamaño del contenido intrínseco tienen otras prioridades adicionales,
 - que indican **cuánto se resisten** a usar otro tamaño que no sea el intrínseco:
 - **Resistencia a estirarse** = 250.
 - **Resistencia a comprimirse** = 750.
 - Hay prioridades independientes para cambios en **horizontal** y en **vertical**.
- Las prioridades con mayor prioridad se aplican antes que las de menor prioridad.

- La prioridad de la resistencia a expandirse o comprimirse solo es importante para las views que tienen un tamaño intrínseco.
- Si una view contiene en su interior varias subviews de las que tienen un tamaño intrínseco,
 - y estas subviews están conectadas entre sí,
 - entonces es necesario ajustar en las subviews la prioridad de la resistencia a ser estiradas o comprimidas
 - para resolver las posibles ambigüedades que puedan darse al calcular el layout.



Velocidad de lanzamiento

123456789

m/s

Content Hugging Priority

Horizontal 250

Vertical 250

Content Compression Resistance Priority

Horizontal 749

Vertical 750

Content Hugging Priority

Horizontal 249

Vertical 250

Content Compression Resistance Priority

Horizontal 750

Vertical 750

Content Hugging Priority

Horizontal 250

Vertical 250

Content Compression Resistance Priority

Horizontal 750

Vertical 750

Top y Botton Layout Guides

- Las guías del layout Top y Botton son las líneas donde acaban las barras de estado, la barra del Navigation Bar Controller, la barra del Tab Bar Controller,
- Podemos poner constraints contra estas guías para evitar que molesten en el posicionamiento del contenido de la pantalla.

