



POLITÉCNICA

ETSIT
UPM

dit
UPM

Desarrollo de Apps para iOS Auto Layout

IWEB,LSWC 2014-2015

Santiago Pavón

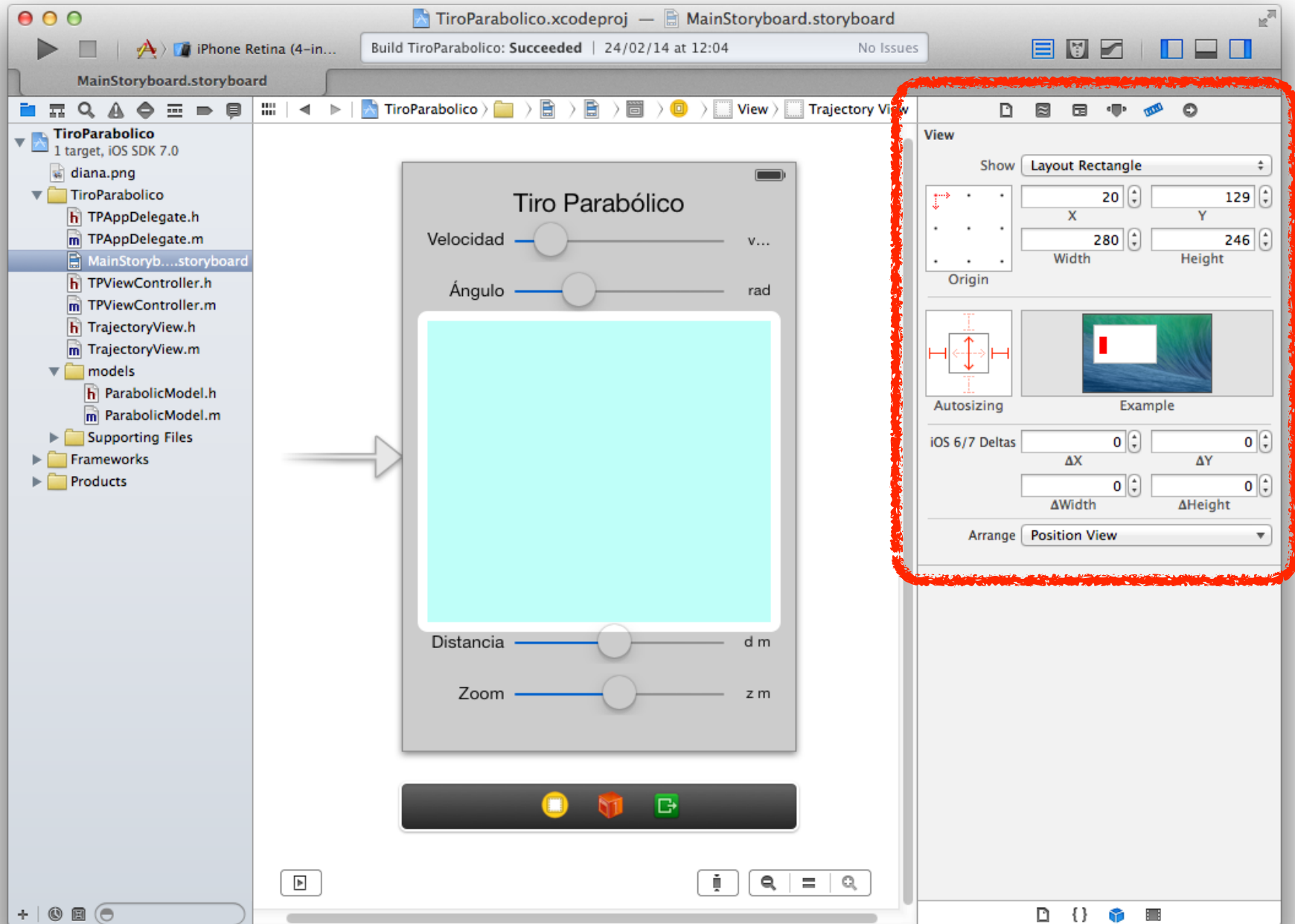
ver: 2014.09.25

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 píxeles.
 - 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:

```
let constraint = NSLayoutConstraint(  
    item: buttonOK,  
    attribute: .Width,  
    relatedBy: .Equal,  
    toItem: titleLabel,  
    attribute: .Height,  
    multiplier: 2,  
    constant: 50)
```

```
view.addConstraint(constraint)
```




```
NSLayoutConstraint *constraint = [NSLayoutConstraint  
    constraintWithItem: self.buttonOK  
        attribute: NSLayoutConstraintWidth  
        relatedBy: NSLayoutConstraintEqual  
        toItem: self.titleLabel  
        attribute: NSLayoutConstraintHeight  
    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

```
let dic = ["boton1":boton1,
          "boton2":boton2,
          "boton3":boton3];
```

```
let format = @"|-[boton1]-(>=8)-[boton2(120)]-[boton3(30)]-|"
```

```
let constraints =
```

```
    NSLayoutConstraint.constraintsWithVisualFormat:(format,
                                                    options: .AlignAllTop
                                                    metrics:nil
                                                    views:dic)
```

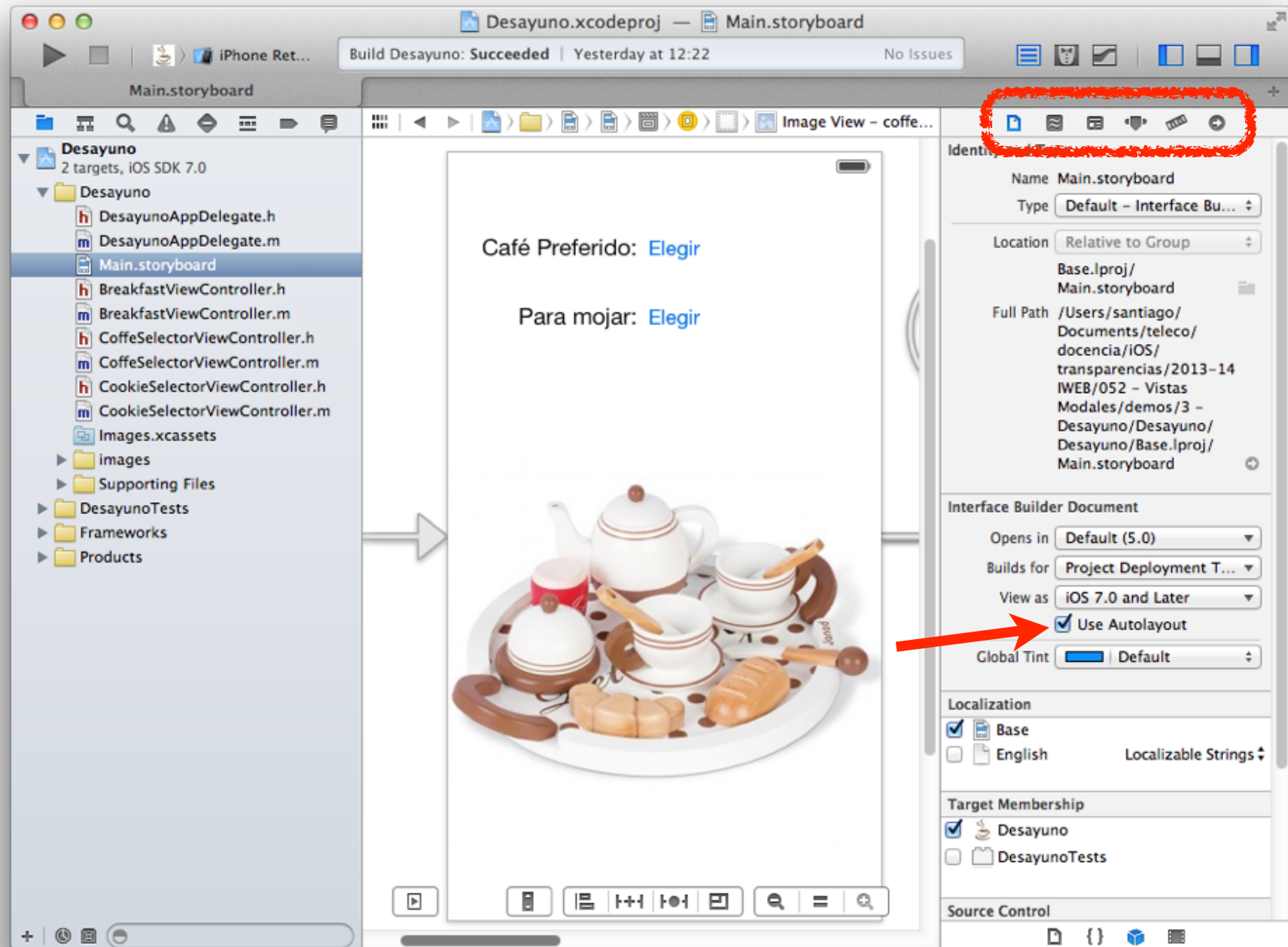
```
view.addConstraints(constraints)
```

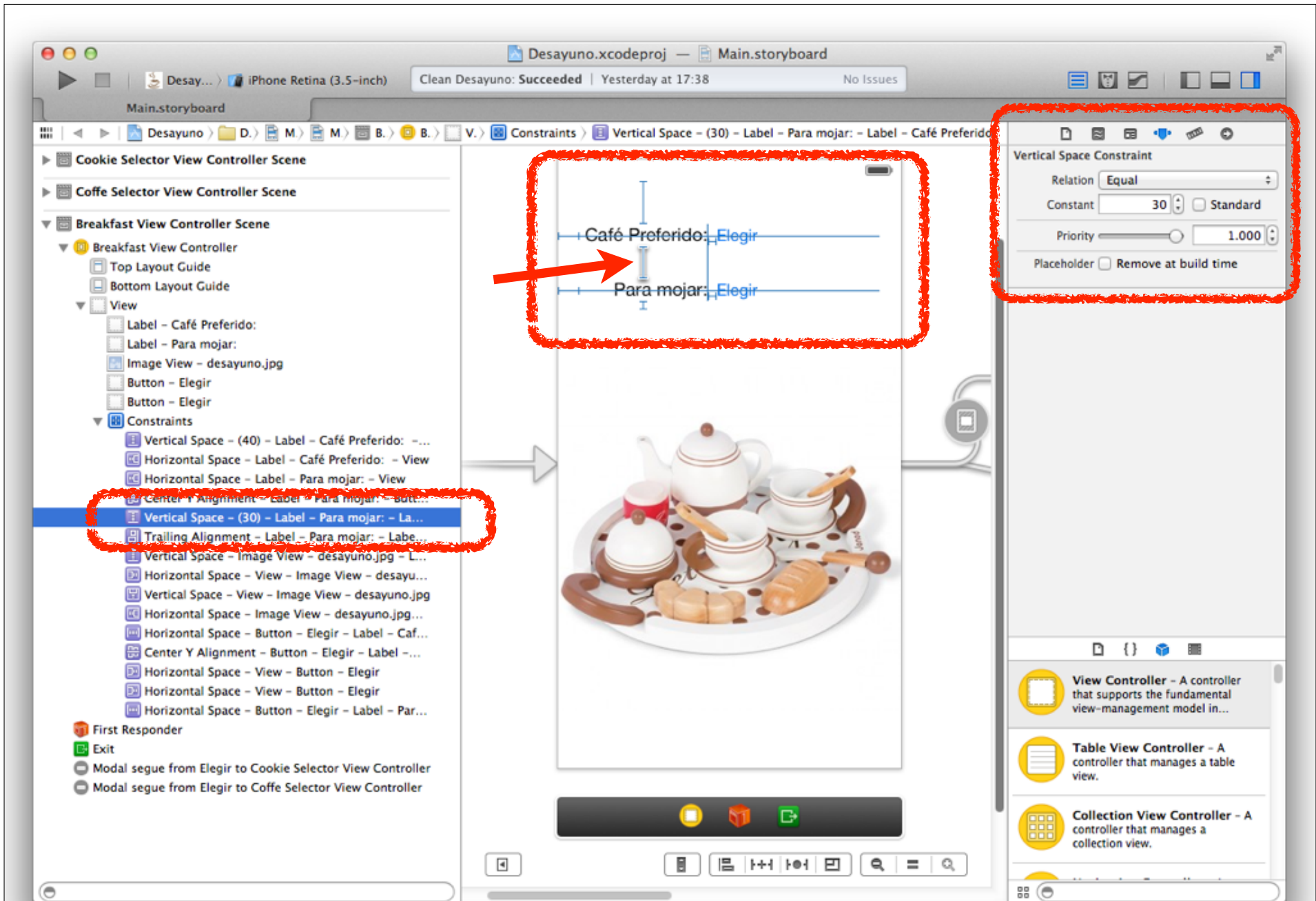


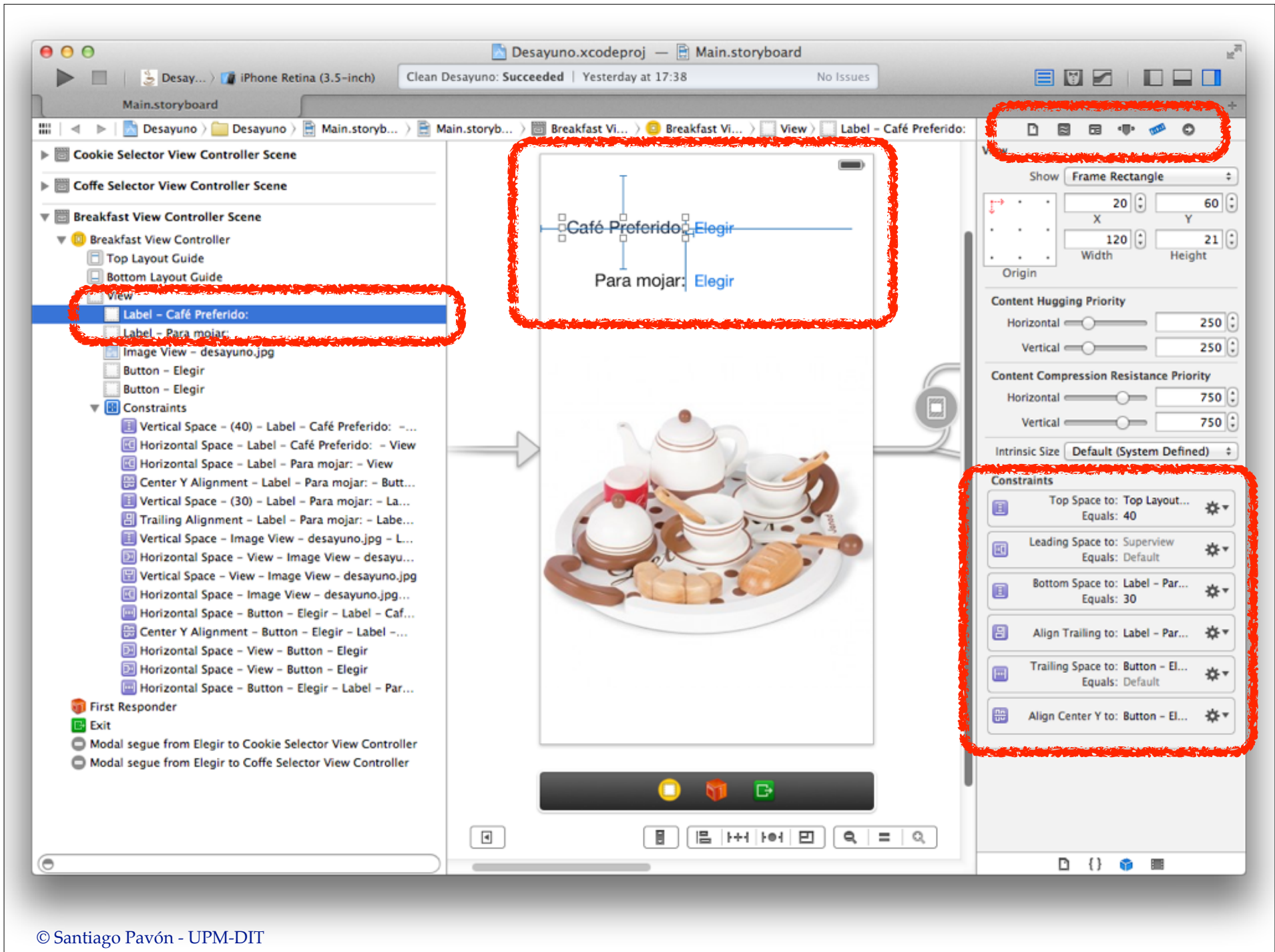


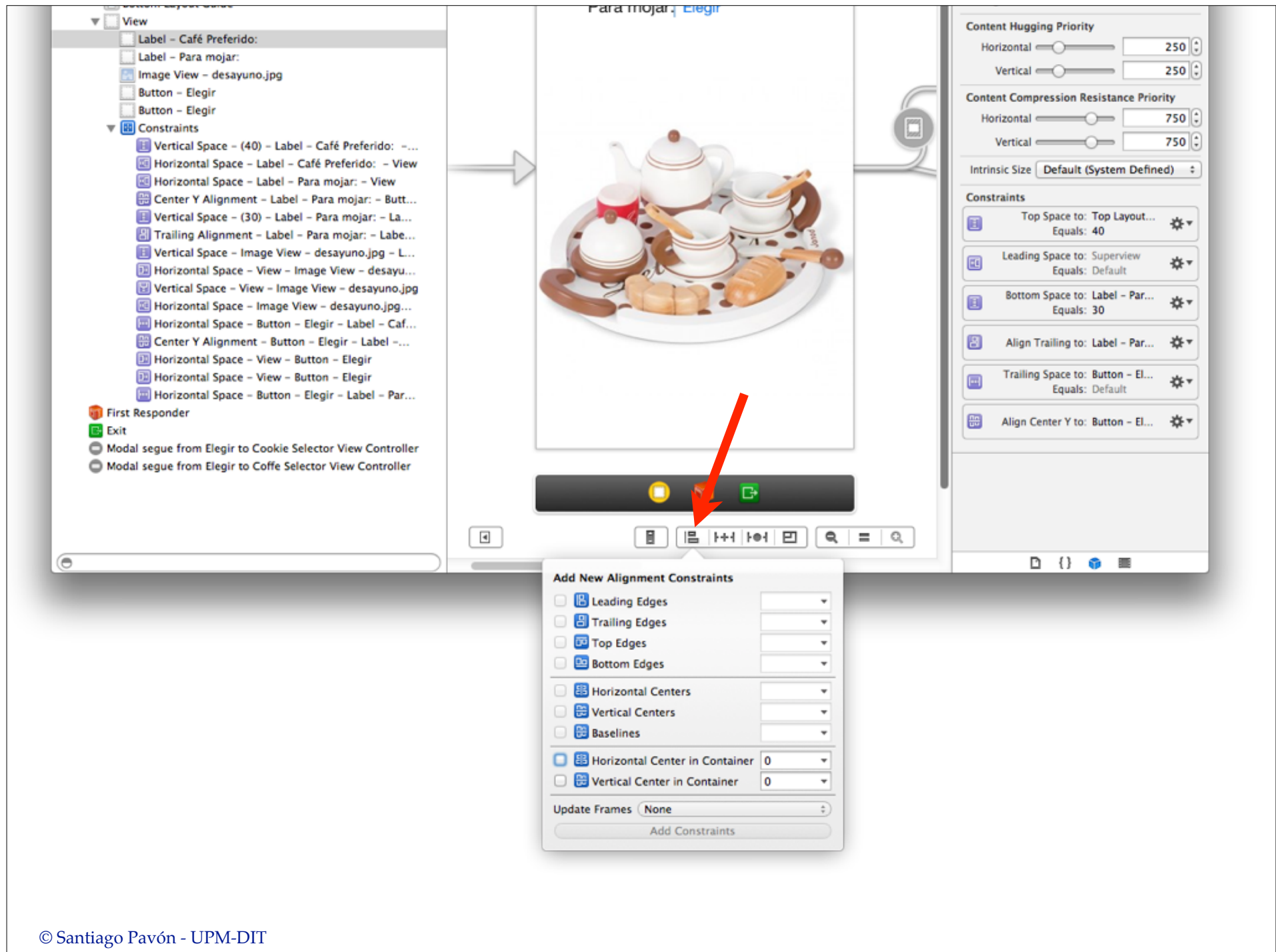
```
NSDictionary *dic = @{@"boton1":boton1,  
                    @"boton2":boton2,  
                    @"boton3":boton3};  
  
NSArray *constraints = [NSLayoutConstraint  
    constraintsWithVisualFormat:@" |[boton1]-(>=8)-[boton2(120)]-[boton3(30)]-|"  
    options:NSLayoutFormatAlignAllTop  
    metrics:nil  
    views:dic];  
  
[self.view addConstraints:constraints];
```

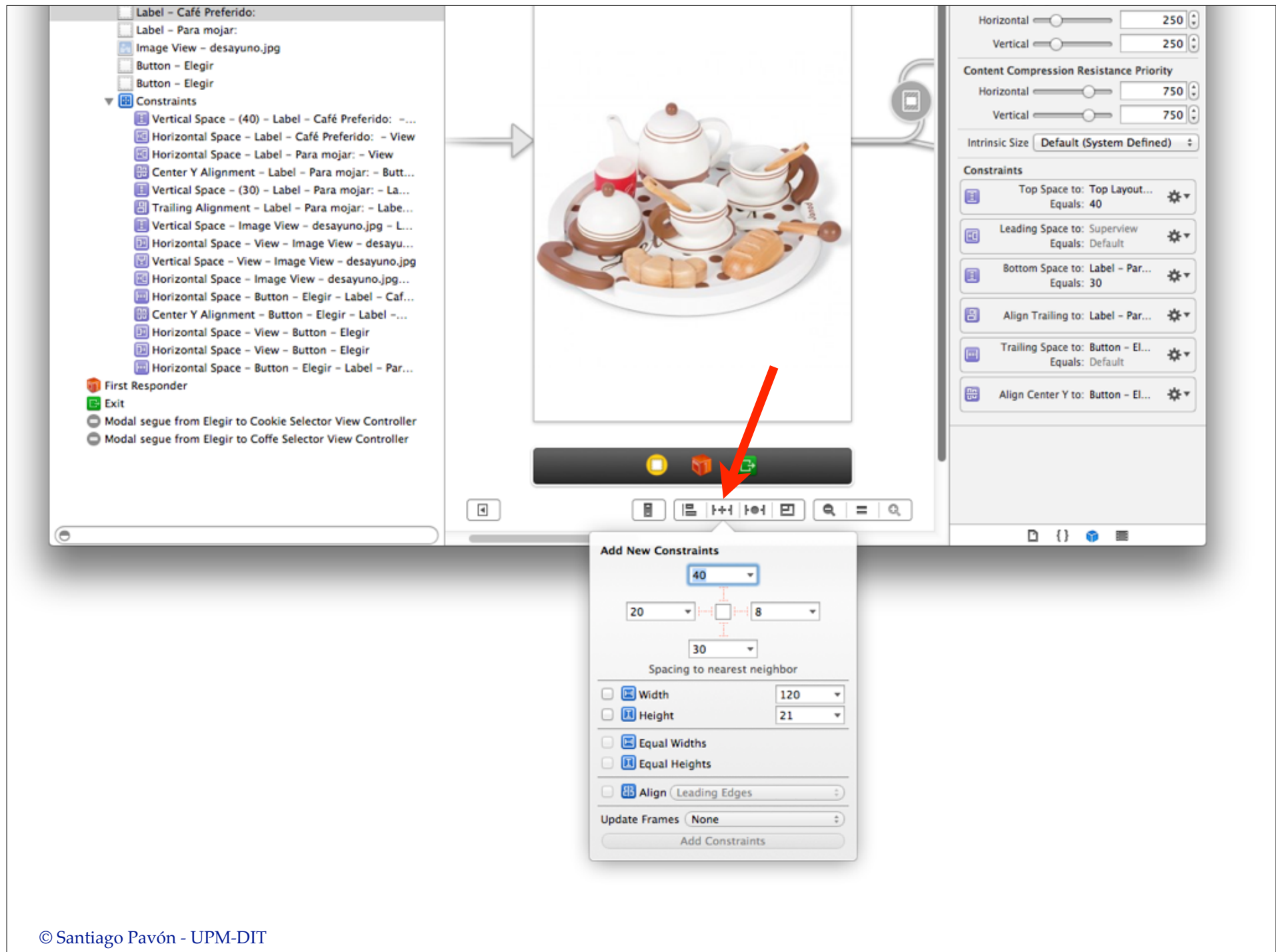
• Interactivamente con el Interface Builder:

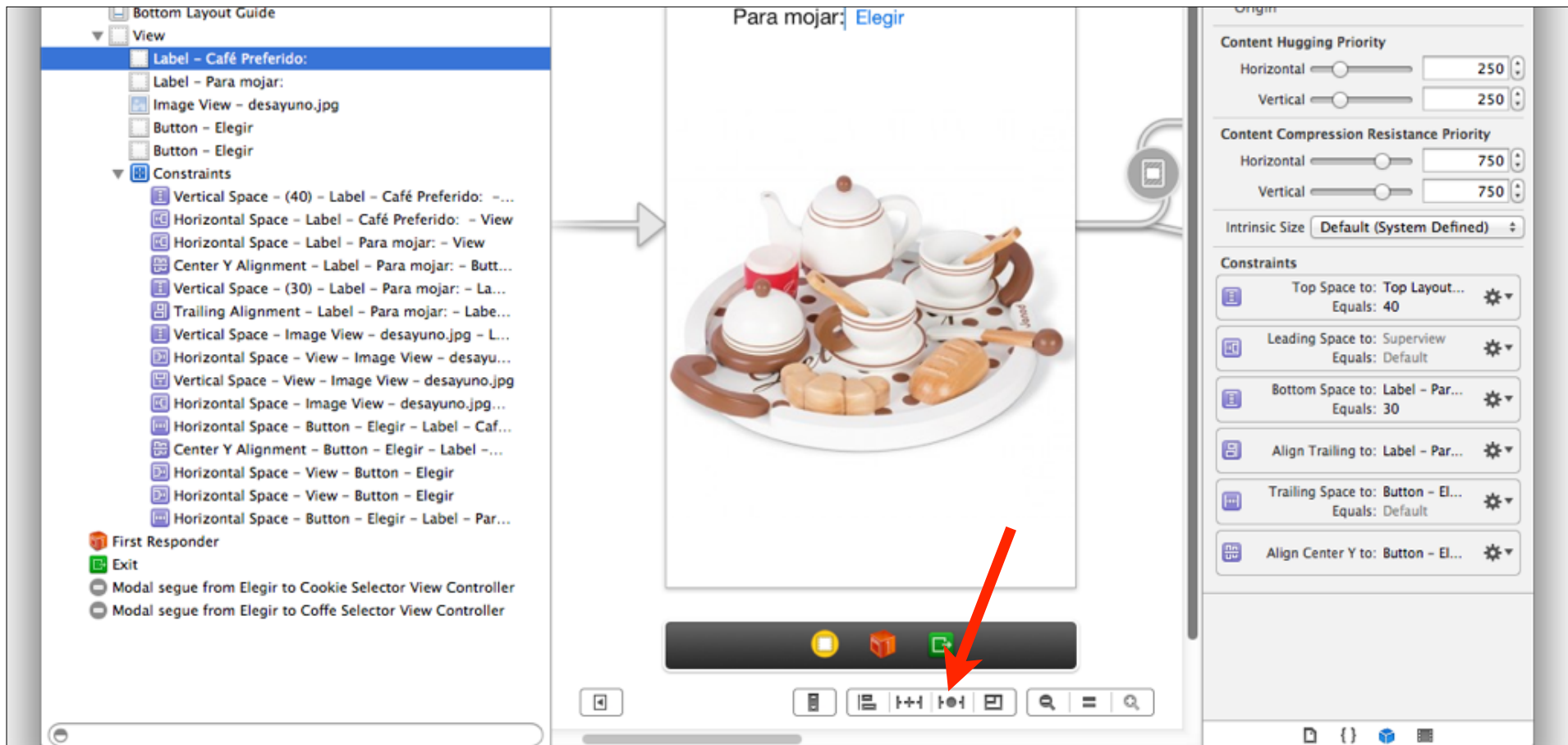




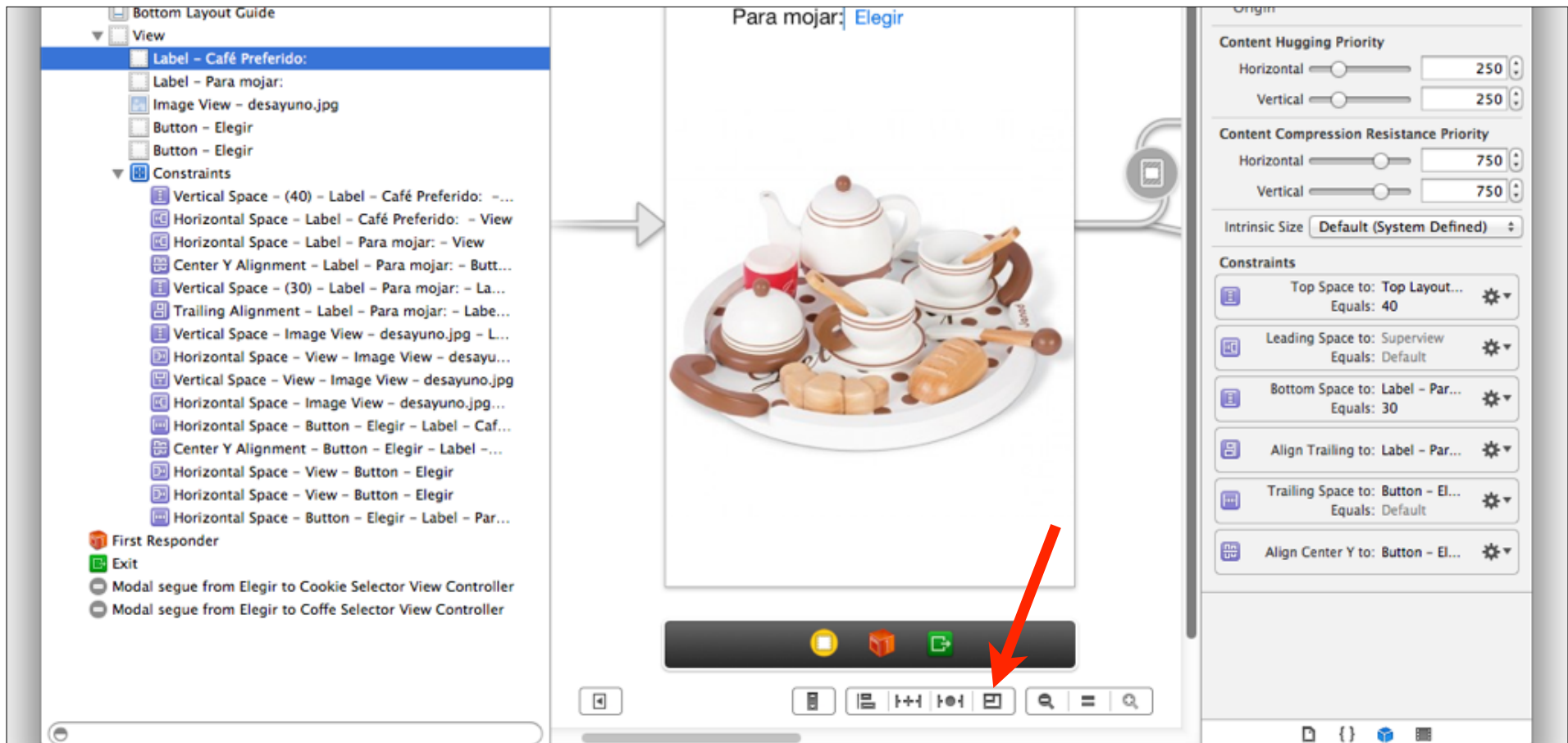




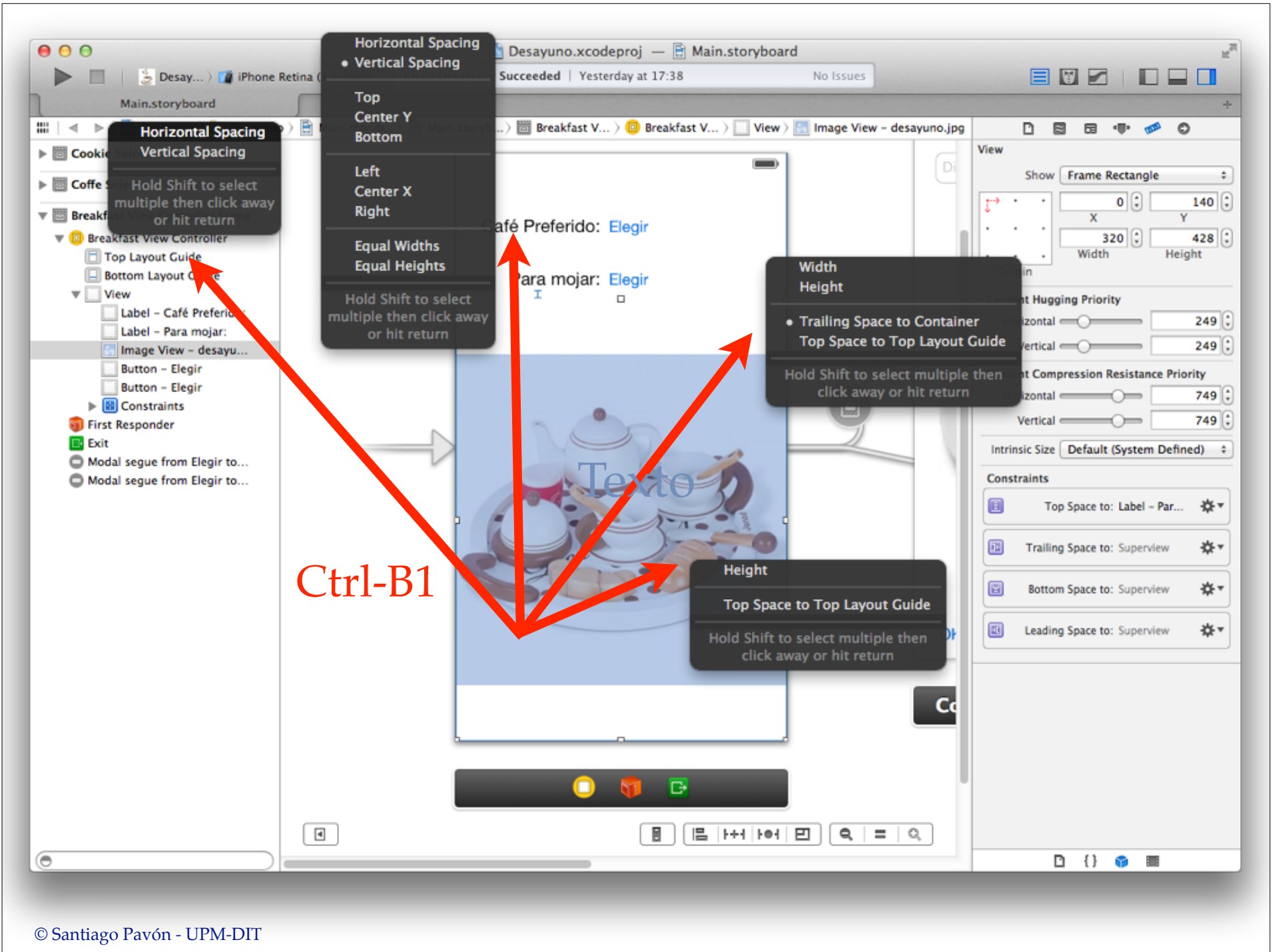




- Update Frames ⌘⇧=
 - Update Constraints ⇧⌘=
 - Add Missing Constraints
 - Reset to Suggested Constraints ⌘⇧⇧=
 - Clear Constraints
-
- Update All Frames in Breakfast View Controller
 - Update All Constraints in Breakfast View Controller
 - Add Missing Constraints in Breakfast View Controller
 - Reset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

iOS 8

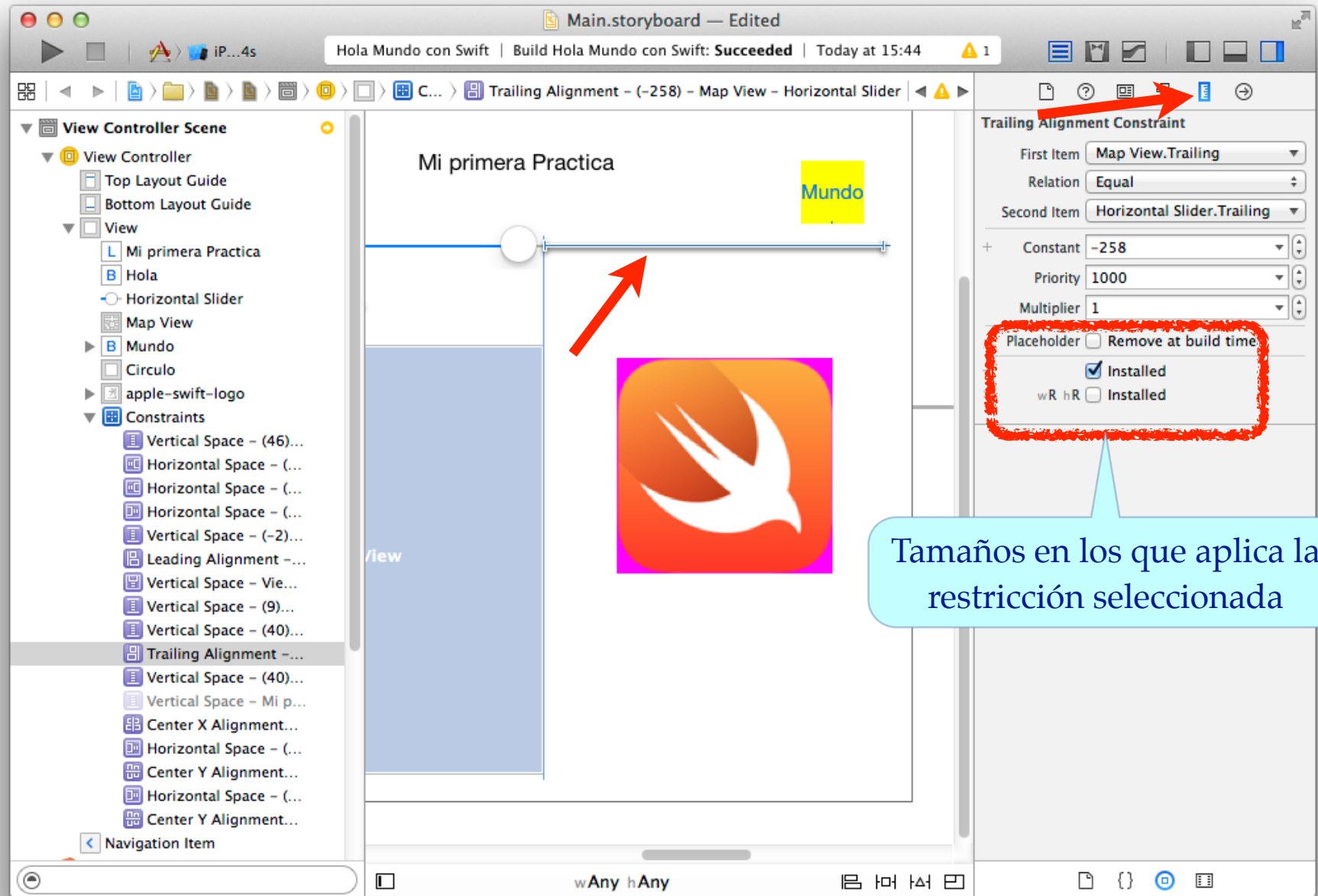
- Con iOS 8 se introducen los Storyboards Universales.
 - No hay que crear dos Storyboards: uno para iPhone y otro para iPad.
 - Con un único storyboard se diseña el interface para todos los tamaños de terminal.
- Activar **Use Size Classes** en el **File Inspector** del **Storyboard**
- **Size Classes** define:
 - tres tamaños: **Compact**, **Regular** y **Any**.
 - tanto en **horizontal**, como en **vertical**.
 - En función del tamaño disponible para la vista de los ViewControllers se configura que restricciones de autolayout se aplican, y que views aparecen.
- Usar **Preview** en el modo **Asistente** con el storyboard para previsualizar como queda el interfaz de usuario para los distintos terminales existentes.
- Además, en función de tamaño del terminal pueden usarse recursos diferentes de los *.xcassets, los segues se adaptan, etc...

The screenshot shows the Xcode Interface Builder environment for a storyboard named 'Main.storyboard'. The left sidebar displays the project structure, with 'Main.storyboard' selected and highlighted by a red arrow. The central canvas shows a storyboard scene with a 'View Controller' containing a yellow button labeled 'Mundo'. A 'Size Inspector' panel is overlaid on the canvas, showing 'Any Width | Any Height' and 'Base Values For all layouts' with 'wAny hAny' selected. A red arrow points to the 'Use Size Classes' checkbox in the 'Interface Builder Document' panel on the right, which is also highlighted by a blue callout bubble containing the text 'Usar Size Classes'. Another blue callout bubble at the bottom center contains the text 'Seleccionar Tamaño'.

The image shows a screenshot of the Xcode storyboard editor. The main canvas displays a storyboard with a scene titled "Mi primera Practica" containing a yellow box labeled "Mundo" and a blue box labeled "lapView". A red arrow points to an image view of the Swift logo. Another red arrow points to the "Image View" properties panel on the right. In this panel, the "View" section is expanded, and a red dashed box highlights the "Width" and "Height" sections, which contain the following options:

- + Installed
- x wAny hC Installed
- x wR hR Installed

A light blue callout bubble at the bottom right contains the text: "Tamaños en los que aparece la view seleccionada".



Algunos atajos de teclado para la edición de los Storyboards:

- **Cmd-Delete** sobre una **view**: elimina esa view solo para el tamaño (Size Class) seleccionado.
- **Delete** sobre una **view**: elimina esa view completamente, es decir, para todos los tamaños (Size Class) existentes.

- **Cmd-Delete** sobre una **restricción**: elimina esa restricción solo para el tamaño (Size Class) seleccionado.
- **Delete** sobre una **restricción**: elimina esa restricción solo para el tamaño (Size Class) seleccionado.

- **Shift-Ctrl-B1** sobre una **view**: muestra un popup para elegir la view que se quiera seleccionar.

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**.

The screenshot shows the Xcode interface for a project named 'TiroParabolico'. The main storyboard, 'MainStoryboard.storyboard', is open, displaying a scene titled 'View Controller Scene' containing a 'View Controller' with a 'Trajectory View'.

The 'Trajectory View' contains several UI elements:

- Two sliders: 'Velocidad' (Velocity) and 'Ángulo' (Angle).
- A blue rectangular area representing the trajectory.
- Two more sliders: 'Distancia' (Distance) and 'Zoom'.

Annotations in red include:

- A red dashed box around the 'View Controller' in the hierarchy.
- A red dashed box around the 'Trajectory View' in the hierarchy.
- A red dashed box around the 'Trajectory View' widget in the storyboard.
- A red dashed box around the 'Distancia' slider widget.
- A red arrow pointing to the 'Reset to Suggested Constraints' button in the context menu.

The right-hand 'View' inspector shows the 'Layout Rectangle' for the selected widget with the following values:

X	0	Y	156
Width	220	Height	219

 Below this, 'Content Hugging Priority' and 'Content Compression Resistance Priority' are set to 250 and 750 respectively for both horizontal and vertical axes.

The 'Constraints' section shows:

- Leading Space to: Superview
- Trailing Space to: Superview
- Leading Space to: Superview
- Top Space to: Label - Ángulo
- Bottom Space to: Label - Distancia

A context menu is open at the bottom, listing:

- Update Frames
- Update Constraints
- Add Missing Constraints
- Reset to Suggested Constraints

Tamaño Intrínseco del Contenido

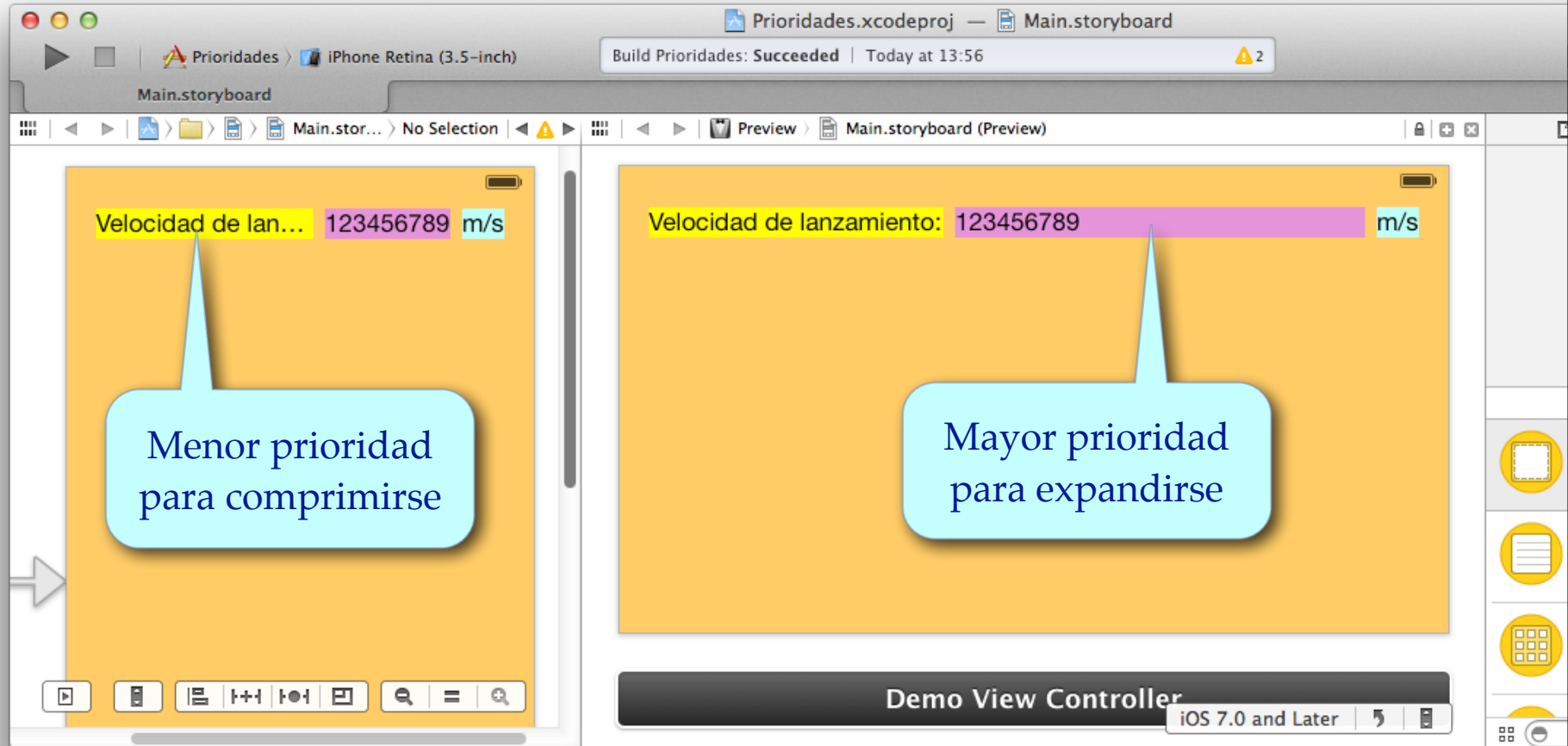
- Es el tamaño exacto que requieren tener algunas views (UILabel, UIButton, UIImageView, ...) para que quepa su contenido.
- Para redimensionar una view a su tamaño intrínseco, seleccionarla y ejecutar:

`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.

