iOS alapú szoftverfejlesztés - Labor 08

A laborsegédletet összeállította: Kántor Tibor - kantor.tibor@bmeautsoft.hu

A labor témája:

* Adaptive Layout
* SplitViewController
* PopoverController

A labor célja az Adaptive Layout használatatának a gyakorlása egy névnapos alkalmazás kezdeti képernyőin keresztül.

A laborhoz tartozó nagyobb kódrészletek a következő url-en érhetők el "copy-paste barát" formában:

https://gist.github.com/DjCantor/ea45f07c0c0df7095d9d#file-namehandler

# AdaptiveLayout bemutatása

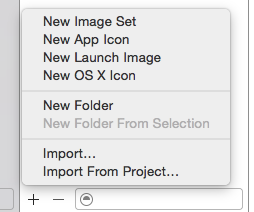
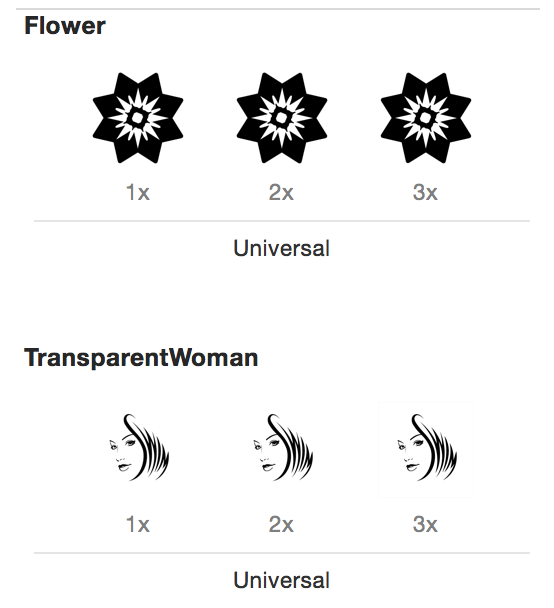
## iNames

Hozzunk létre egy új Single View Application universal projektet iNames névvel.

Töltsük le a következő URL-ről a képi erőforrásokat:

***https://dl.dropboxusercontent.com/u/82267346/iOS8LaborResource.zip***

Kattintsunk rá az ImageAssets mappára, majd importáljuk be a Flower és TransparentWoman képeket az Import... menü segítségével.

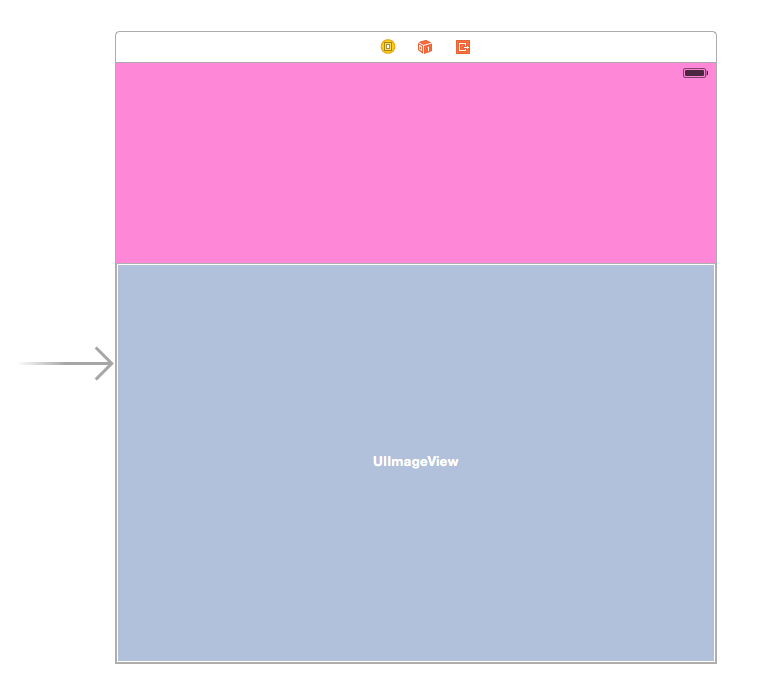
Váltsunk át Storyboard-ra és háttérnek állítsuk be 255,245,245-ös színt. A Storyboardban adjunk hozzá egy sötét rózsaszín (255,146,217) View-t és alkalmazzuk a következő kényszereket:

* A két széle és a teteje legyen rögzítve 0 távolságra a szülő nézettől.
* A magassága legyen 200 egység.

Rakjunk be egy Image View-t és alkalmazzuk a következő kényszereket:

* Az Image View és a szülő nézet két széle és az alja között legyen 0 a távolság.
* Az Image View és a felette lévő view között legyen 0 a távolság!

A következő nézetet kell látnunk:

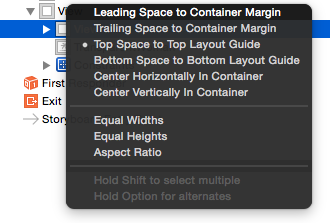


Állítsuk be a kép tulajdonságainál, hogy a megjelentés Aspect Fit, és állítsuk be a *TransparentWoman* nevű képet.

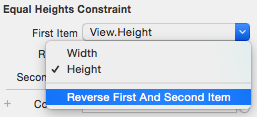
Nyissuk le az Assistant Editor-t és válasszuk ki benne a Preview funkciót és adjunk hozzá még egy iPhone és még egy iPad eszközt. A következő a probléma: az iPhone-on Landscape esetben túl nagy a felső sáv, míg az iPad-en portrait módban túl kicsi.

Töröljük a Document Outline-ban a View magasság kényszerét és állítsuk be, hogy a magassága mindig a szülő nézet magasság 0.4-szerese!

Ezt a Document Outline-ban tehetjük meg, úgy, hogy a Ctrl-t lenyomva ráhúzzuk a vonalat a gyerek View-ról a szülő view-ra:

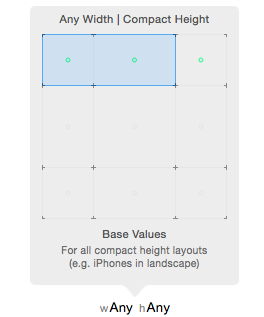


Ezek után a kényszer tulajdonságainál állítsuk be, hogy a szorzó (multiplier) 0.4 legyen. Ha piros vonalakat kapnánk, akkor az azt jelenti, hogy a kényszer rossz változókkal jött létre (még pedig azzal, hogy a gyerek legyen 2.5-ször akkora, mint a szülője, ami a Image View szülőhöz történt rögzítése miatt nem teljesülhet). Cseréljük fel őket:

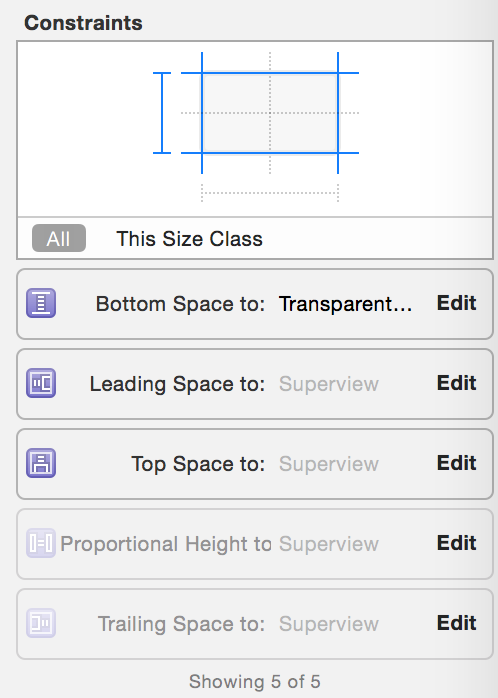


Nézzük meg az alkalmazást. Az iPhone kijelzőn Landscape módban az Image View indokolatlanul sok helyet foglal a kép méretéhez viszonyítva.

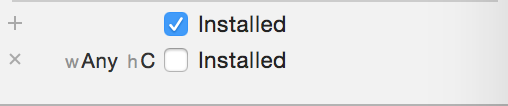
Válasszuk ki az Size Class menüből az *any width compact height* méretosztályt:



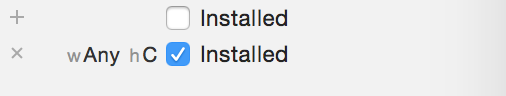
Ekkor megváltozik a Storyboard Editor képe. Jelöljük ki a felső View-t és töröljük a Size Inspectorban a View magasságát és jobb oldalát rögzítő kényszert a backspace-szel:



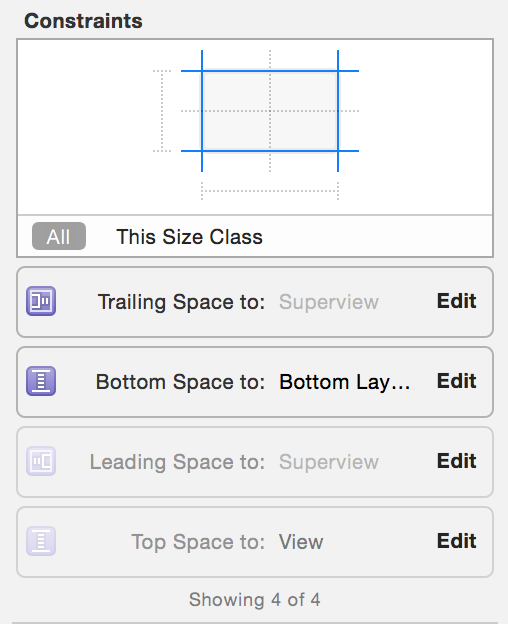
Ha rákattintunk egy elhalványított kényszerre, látjuk, hogy nincs installálva:



Módosítsuk a view-t az egérrel, az alját húzzuk le a szülő széléig, a szélességét pedig csökkentsük le. Az alját rögzítsük a szülő nézet aljához, a szélessége meg legyen a szülő nézet 0.7-szerese. Ha egy frissen hozzáadott kényszerre kattintunk a tulajdonságainál látjuk, hogy csak erre a méretosztályra van installálva:

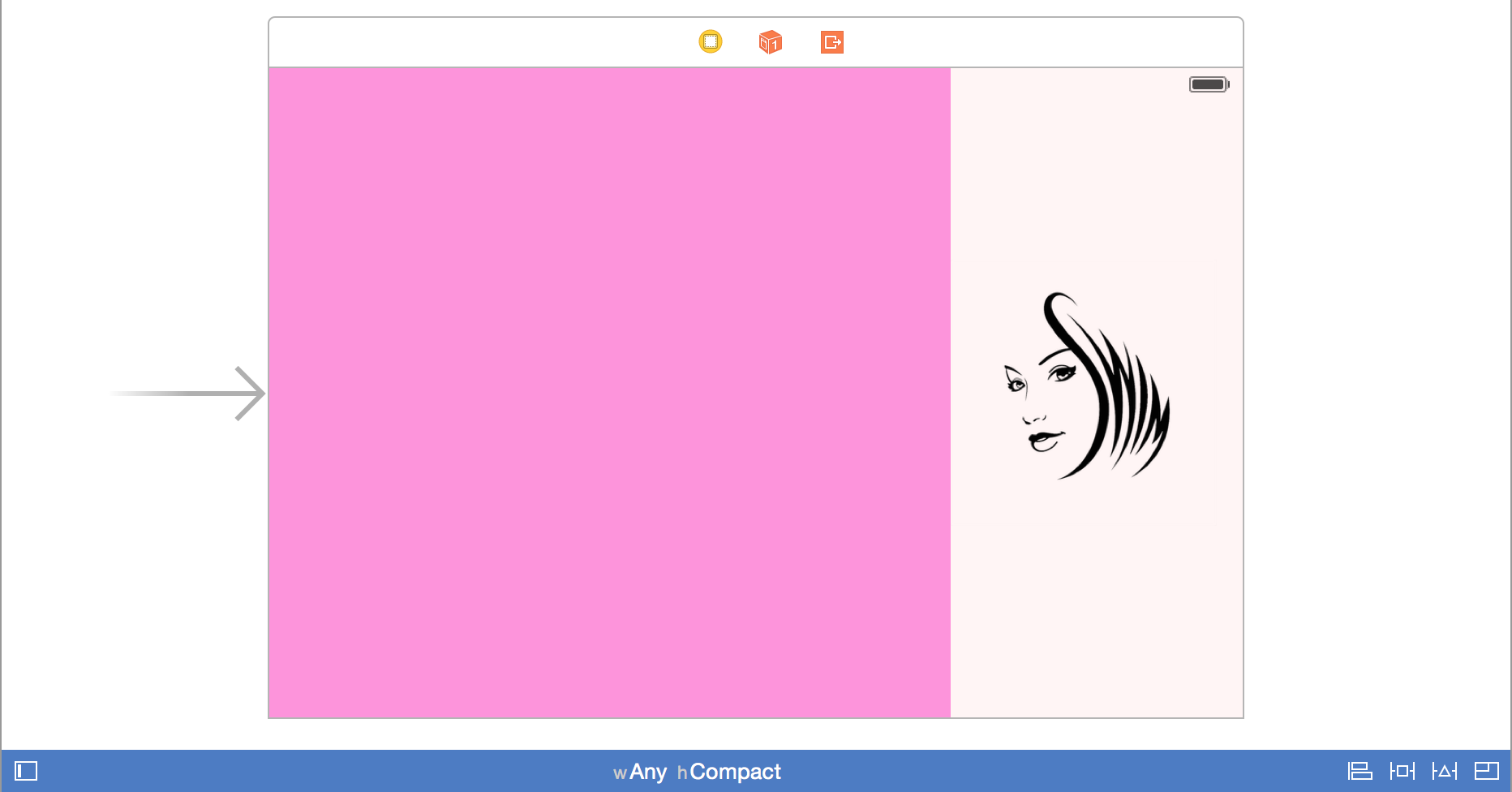


Az Image View-n töröljük a bal oldali kényszert, továbbá a tetejének a rögzítését is backspace-szel:



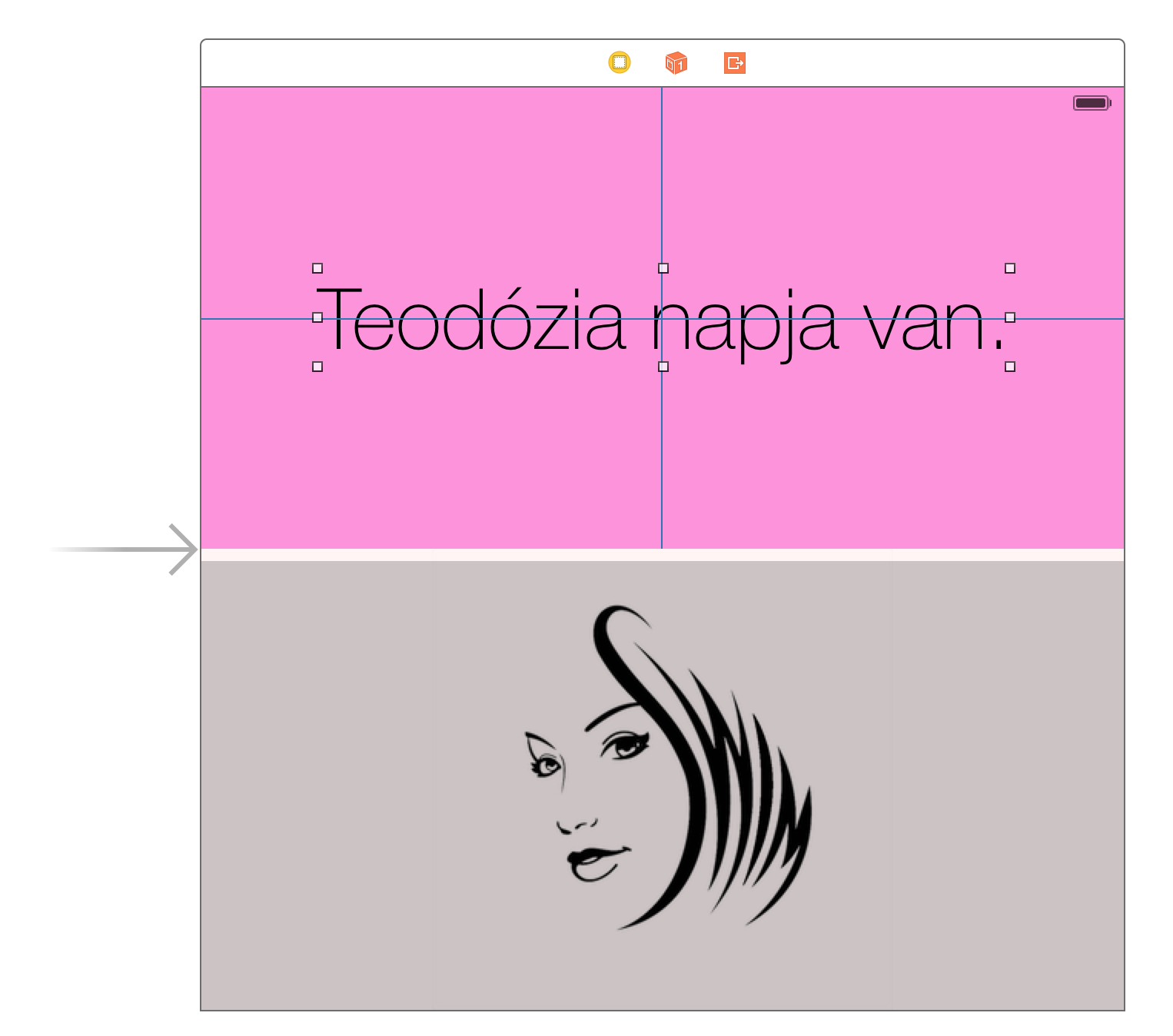
Majd adjuk hozzá a szülő nézet tetejétől mért 0 távolság kényszert, a bal oldalát pedig rögzítsük 0 távolságra a mellette lévő View-tól.

Egy Update Frames hívása után a következőt kell látnunk:



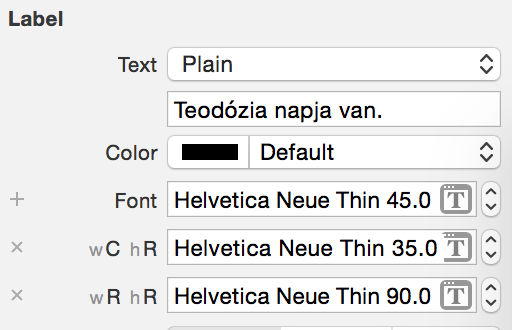
Ellenőrizzük az eredményt a Preview-ban különböző szimulátorok elforgatásával. Váltunk vissza any x any méret osztályba és ellenőrízzük a kényszereket a Document Outline-ban, mind pedig az egyes View-k esetében.

Adjunk hozzá egy labelt a View-hoz, rendezzük középre, állítsunk be 45-ös méretű *Helvetica Neue Thin* betűtípust, a szöveget pedig írjuk át "Teodózia napja van."-ra:



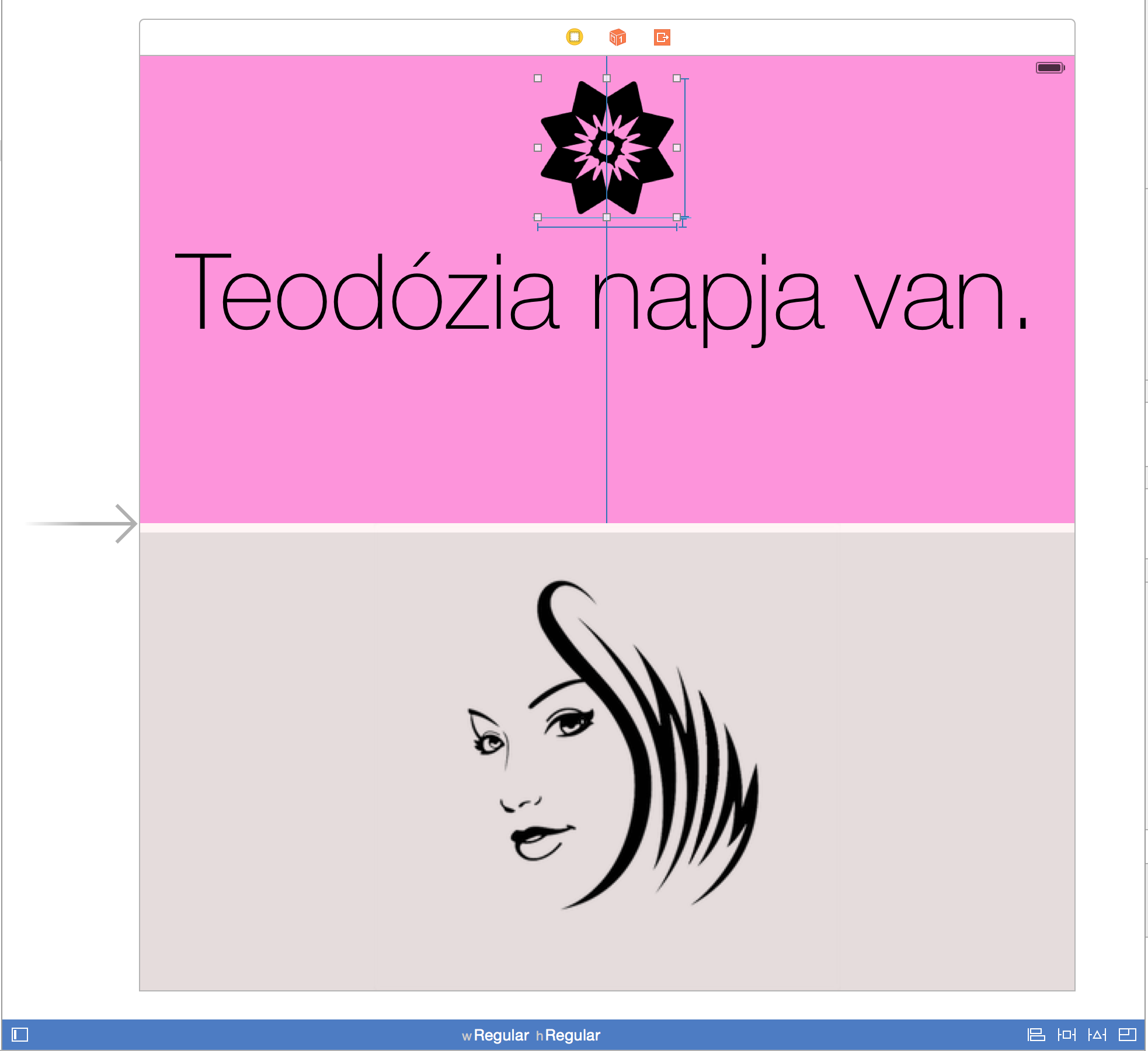
A preview-ban láthatjuk, hogy iPhone portait-ban ez túl nagy betűtípus, míg az iPad esetében túl kicsi.

Kattintsunk a font melletti plusz ikonra és adjunk hozzá az iPad és az iPhone portait esetében két hasonló stílusú, de eltérő méretű fontot (35 és 90).



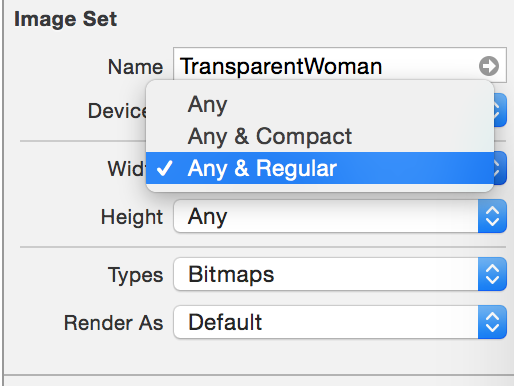
Láthatóan az iPad kijelzőn van még hely, tegyünk be egy képet szöveg felé!

Váltsunk át *Regular x Regular* size class-be és tegyünk be egy ImageView-t a szöveg fölé, rendezzük középre, a méretét rögzítsük 90 x 90-as a kép és a szöveg közötti távolság legyen 0.

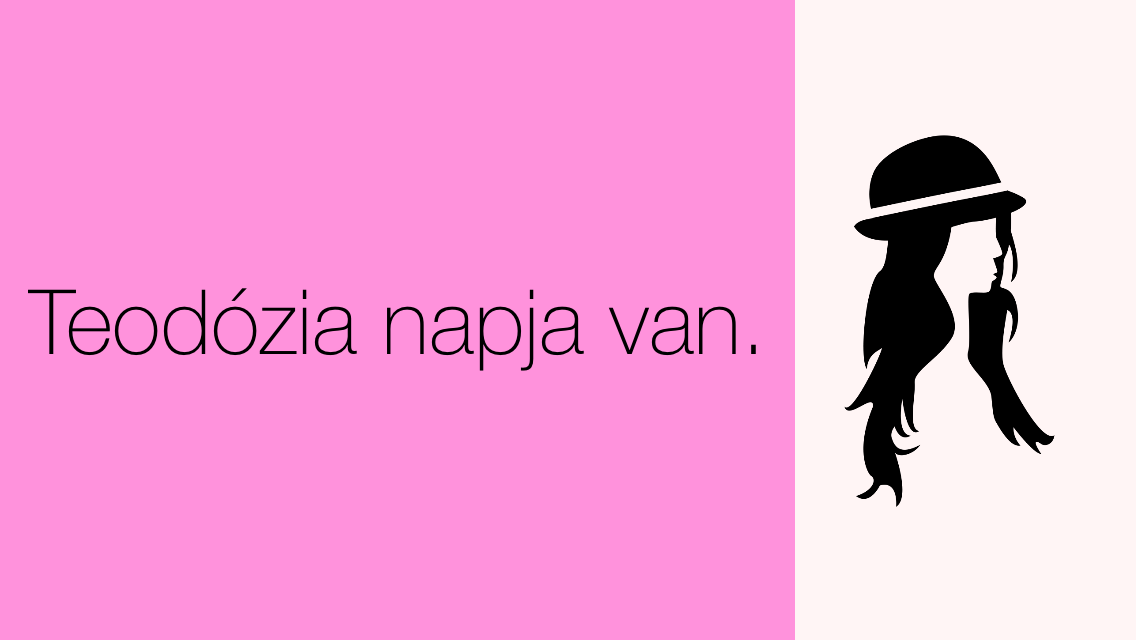


Ellenőrizzük le a Preview-ban, hogy az iPaden valóban megjelenik majd a kép, de a többi eszközön nem.

Landscape módban célszerűbb lenne egy magasabb képet használni. Váltsunk át a image assets mappába és a TransparentWoman kép tulajdonságainál állítsuk be, hogy külön képet akarunk megadni a Any x Compact méretosztályhoz:



A [\* -]-gal jelölt Any x Compact méreosztályhoz tartozó helyekre húzzuk be a GirlWithHate képet és ellenőrizzük szimulátorban(!) az eredményt:



## Több névnap egy napon - SplitViewController

Adjuk hozzá a projekthez a Names.plist fájlt. Vegyünk fel egy NameHandler singleton osztályt, ami az adott nap névnapjait fogja visszaadni:

import UIKit

private let handler = NameHandler()

class NameHandler: NSObject {

let names : [AnyObject]?

class var sharedInstance : NameHandler {

return handler

}

override init() {

super.init()

let path = NSBundle.mainBundle().pathForResource("Names", ofType: "plist")

self.names = NSArray(contentsOfFile: path!)!

}

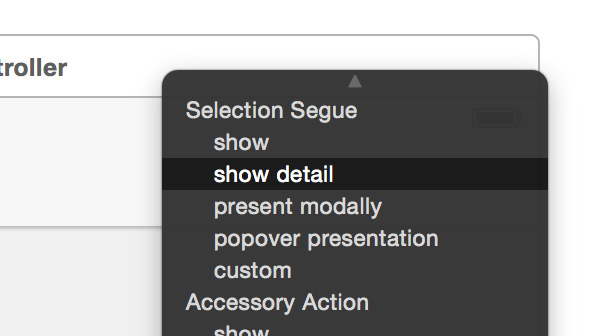
}

https://gist.github.com/DjCantor/ea45f07c0c0df7095d9d#file-namehandler

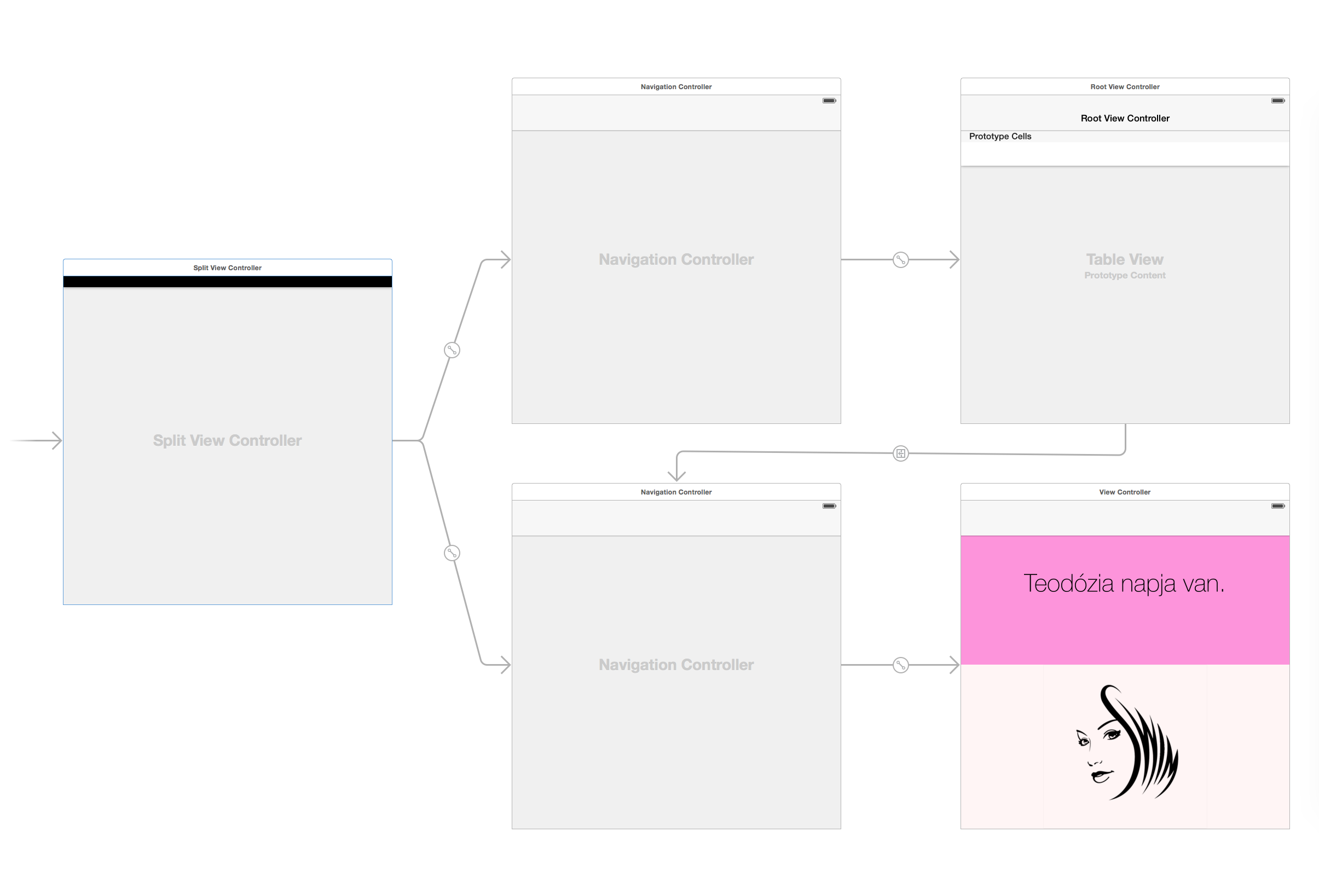
Térjünk vissza a Storyboardhoz és ágyazzuk be a nézetvezérlőt egy Navigation Controller-be.

Tegyünk be egy Split View Controller-t a storyboardba, a létrejött Detail View Controller-t töröljük ki, és a SplitViewController Detail Segue-hez állítsuk be az az imént létrehozott Navigation Controller-t.

A Master ViewController cellájából hozzunk létre a Navigation Controllerre mutató Selection Segue show detail-t, amit nevezzünk is el ShowDetailSegue-nek.



Állítsuk be kezdőképernyőnek a SplitViewController-t. A következő elrendezést kell látnunk:



Az AppDelegate-ben egészítsük ki a didFinishLaunchingWithOptions metódust:

let splitViewController = self.window!.rootViewController as! UISplitViewController

let navigationController = splitViewController.viewControllers[splitViewController.viewControllers.count-1] as! UINavigationController

navigationController.topViewController!.navigationItem.leftBarButtonItem = splitViewController.displayModeButtonItem()

splitViewController.delegate = self

Valósítsuk meg a SplitViewController delegate kötelező metódusát:

func splitViewController(splitController: UISplitViewController, collapseSecondaryViewController secondaryViewController: UIViewController, ontoPrimaryViewController primaryViewController: UIViewController) -> Bool {

return true

}

Próbáljuk ki az alkalmazást, forgassuk el!

Majd adjunk hozzá egy új ViewController leszármazottat NameViewController néven.

Hozzunk létre outletet a labelnek TitleLabel néven, illetve egy nameToDisplay változót, majd módosítsuk a viewDidLoad metódust:

import UIKit

class NameViewController: UIViewController {

@IBOutlet weak var titleLabel: UILabel!

var nameToDisplay : [NSString : NSString]?

override func viewDidLoad() {

super.viewDidLoad()

if nameToDisplay == nil {

nameToDisplay = NameHandler.sharedInstance.names!.first as? [NSString : NSString]

}

let name = nameToDisplay!["name"]

self.titleLabel.text = "\(name!) napja van."

}

}

https://gist.github.com/DjCantor/ea45f07c0c0df7095d9d#file-nameviewcontroller

Adjunk hozzá egy új TableViewController származott osztályt. Állítsuk be a Storyboard Editorban, a MasterViewController identitásának, a cella azonosítónak állítsuk be, hogy NamesViewController néven a tartalmát pedig módosítsuk

import UIKit

class NamesViewController: UITableViewController {

var names = [AnyObject]()

override func viewDidLoad() {

super.viewDidLoad()

names = NameHandler.sharedInstance.names!

title = "Mai névnapok"

}

override func numberOfSectionsInTableView(tableView: UITableView) -> Int {

return 1

}

override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

return names.count

}

override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {

let cell = tableView.dequeueReusableCellWithIdentifier("NameCell", forIndexPath: indexPath) as UITableViewCell

let name = names[indexPath.row] as! [NSString : NSString]

cell.textLabel?.text = name["name"] as! String

cell.imageView?.contentMode = UIViewContentMode.ScaleAspectFill

cell.imageView?.image = UIImage(named: "Flower")

return cell

}

override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {

if segue.identifier == "ShowDetailSegue" {

if let indexPath = self.tableView.indexPathForSelectedRow {

let object = names[indexPath.row] as! [NSString : NSString]

let controller = (segue.destinationViewController as! UINavigationController).topViewController as! NameViewController

controller.nameToDisplay = object

controller.navigationItem.leftBarButtonItem = self.splitViewController?.displayModeButtonItem()

controller.navigationItem.leftItemsSupplementBackButton = true

}

}

}

}

https://gist.github.com/DjCantor/ea45f07c0c0df7095d9d#file-namesviewcontroller

Próbáljuk ki az alkalmazást több különböző szimulátorral: iPhone, iPhone 6+ és iPad és figyeljük meg, hogyan alkalmazkodik a SplitViewController a különböző környezetekhez.

Az alapbeállítás szerint iPaden eltűnik a MasterViewController portait módban. Ezen a legegyszerűbben úgy javíthatunk, ha beállítjuk, hogy mi az előnyben részesített megjelenése a Split View Controllernek, az AppDelegate egészítsük ki :

...

splitViewController.delegate = self

splitViewController.preferredDisplayMode = UISplitViewControllerDisplayMode.AllVisible

...

Sokat javíthatunk az alkalmazás kinézetén, ha a gombokat nem kéken jelenítjük meg (ugyanakkor ezek rendszerszintű gombok, amelyek folytonos átszínezgetése kicsit körülményes volna). Be lehet azonban állítani a Storyboard-ra egy globális TintColort, amit az egyes vezérlők alapértelmezett színként felvesznek. Tegyük ezt meg a Storyboard fájl tulajdonságainál állítsunk be valamilyen sötétebb rózsaszín színt:



## Nevek jelentése - PopoverPresentationViewController

Hozunk létre egy új UIViewController leszármazottat NameFactsViewController néven és az implementációt cseréljük le:

class NameFactsViewController: UIViewController {

@IBOutlet weak var nicksLabel: UILabel!

@IBOutlet weak var originLabel: UILabel!

@IBOutlet weak var meaningLabel: UILabel!

var nameToDisplay : [NSString : NSString]?

override func viewDidLoad() {

super.viewDidLoad()

self.nicksLabel.text = nameToDisplay!["nicks"]

let origin = nameToDisplay!["origin"]!

self.originLabel.text = "Eredete : \(origin)"

let meaning = nameToDisplay!["meaning"]!

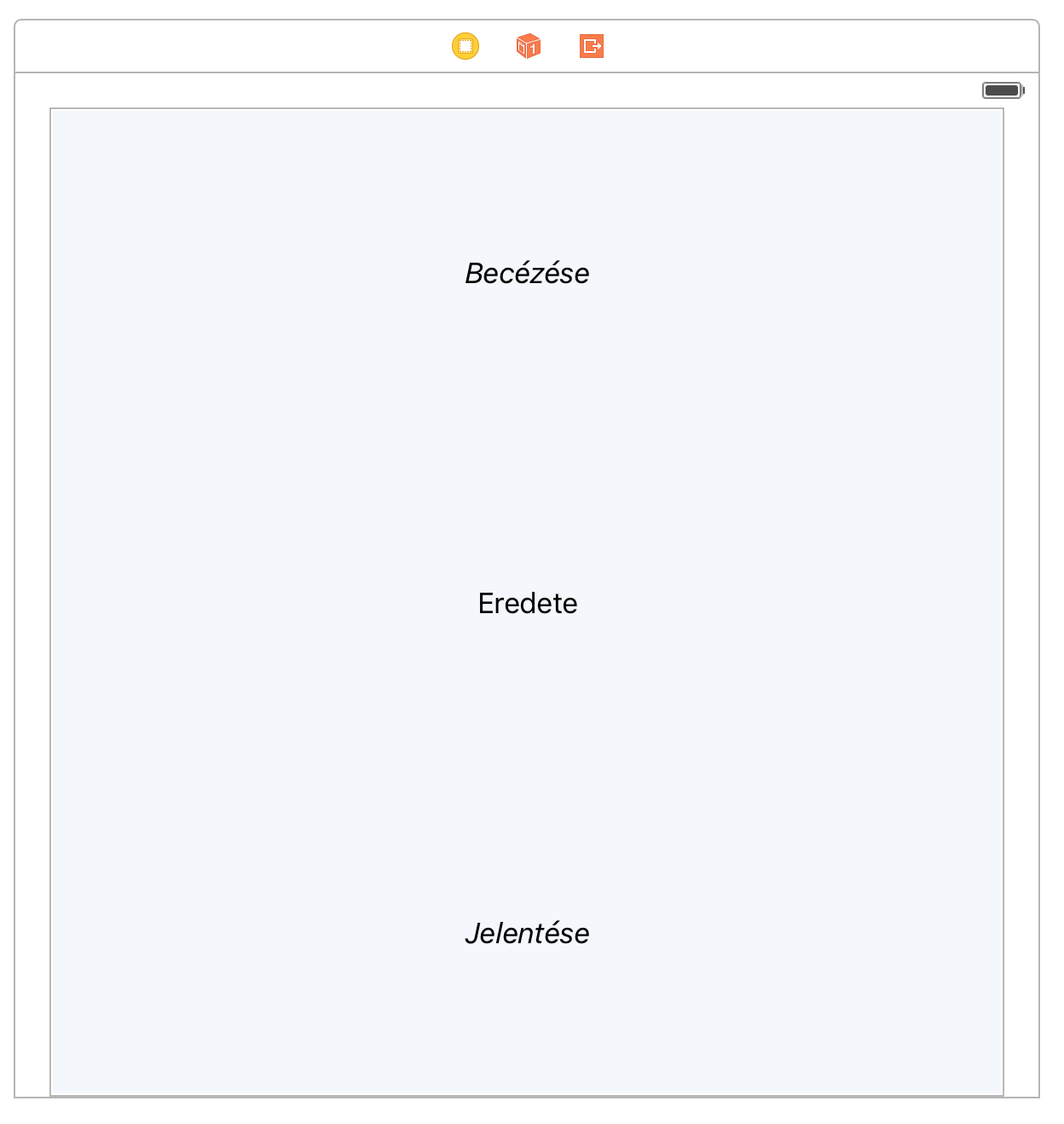
self.meaningLabel.text = "Jelentése: \(meaning)"

}

}

https://gist.github.com/DjCantor/ea45f07c0c0df7095d9d#file-namefactsviewcontroller

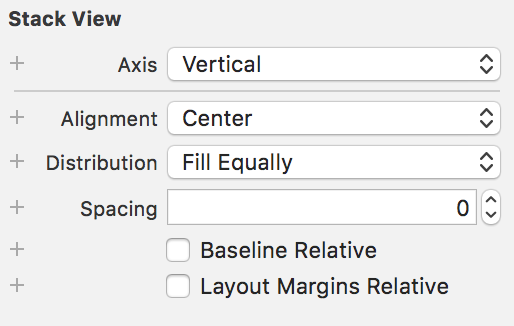
Helyezzünk be egy új nézetvezérlőt, majd állítsuk össze a következő képernyőt:



Ehhez rakjunk be 3 labelt, majd használjuk a stack gombot:

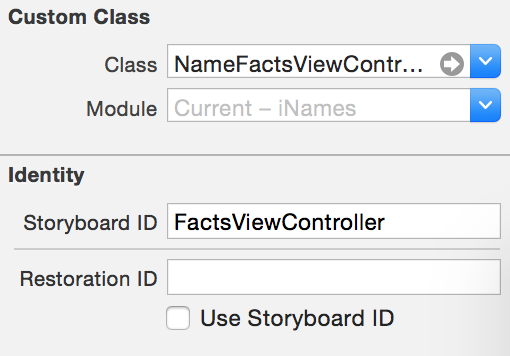
../Screen%20Shot%202015-11-04%20at%2022.16.24.png

A StackView-n állítsuk be, hogy a szülőnézet széleitől a távolsága, a tartamat középre rendezze (Aligment: Center), és, hogy egyforma méretüek legyenek Distriution: Fill Equally.



Állítsuk be, hogy ez a NameFactsViewController és kössük be az outleteit!

Végül adjunk neki azonosítót a storyboardban FactsViewController néven az Identity Inspectorban!



A NameViewControllerben Adjunk hozzá egy új Bar Button item-et a NameViewControllerhez kódból a viewDidLoad-ban:

let detailsButton: UIBarButtonItem = UIBarButtonItem(title: "Tények", style: UIBarButtonItemStyle.Plain, target: self, action: "displayFacts:")

navigationItem.rightBarButtonItem = detailsButton

Majd valósítsuk meg a displayFacts függvényt:

func displayFacts(sender: UIBarButtonItem) {

let storyboard : UIStoryboard = UIStoryboard(name: "Main", bundle: nil)

let contentViewController : NameFactsViewController = storyboard.instantiateViewControllerWithIdentifier( "FactsViewController") as! NameFactsViewController

contentViewController.nameToDisplay = nameToDisplay

contentViewController.modalPresentationStyle = UIModalPresentationStyle.Popover

let detailPopover: UIPopoverPresentationController = contentViewController.popoverPresentationController!

detailPopover.barButtonItem = sender;

detailPopover.permittedArrowDirections = UIPopoverArrowDirection.Any

presentViewController(contentViewController, animated: true, completion:nil)

}

https://gist.github.com/DjCantor/ea45f07c0c0df7095d9d#file-displayfacts

Futtassuk az alkalmazást iPad és iPhone szimulátorokon is. iPhone kijelzőn sajnos nem tudjuk bezárni a modálisan megjelenített nézetvezérlőt. Ha olyan működést szeretnénk, mint iPad-en, akkor meg kell valósítani a UIPopoverPresentationControllerDelegate interfészt. Tegyük is ezt meg a NameViewControllerben, majd állítsuk be a PopoverPresentationController létrehozásánál:

detailPopover.permittedArrowDirections = UIPopoverArrowDirection.Any

detailPopover.delegate = self

...

Végül írjuk meg a callback-et:

func adaptivePresentationStyleForPresentationController( controller: UIPresentationController!) -> UIModalPresentationStyle {

return UIModalPresentationStyle.None

}

## Önálló feladatok plusszpontért

1. A Split View Controller behelyezésével bizonyos képernyőméreteken pl. iPhone 6+ landscape vagy iPad elromlott a labor első felében nehéz munkával elkészített layout. Javítsuk ki a hibákat, hogy minden képernyőn jól nézzen ki az alkalmazás új adaptív kényszerek hozzáadásával vagy a meglévő kényszerek módosításával!
2. Nem minden szöveg fér el a Popover Controller-ben, módosítsuk a NameFactsViewController kényszereit, hogy ez ne legyen probléma!