iOS alapú szoftverfejlesztés - Labor 05

A laborsegédletet összeállította: Kelényi Imre - [imre.kelenyi@aut.bme.hu](mailto:imre.kelenyi@aut.bme.hu), Kántor Tibor – [kantor.tibor@bmeautsoft.hu](mailto:kantor.tibor@bmeautsoft.hu)

A labor témája:

* Storyboards
* Tab Bar Controller
* Navigation Controller
* Modális View Controller megjelenítés
* Segue
* Véletlenszám generálás
* (property list betöltés)

A labor során egy több nézetes, "képkitaláló játékot" készítünk el. A játék véletlenül sorsol képeket és feliratokat a projektbe bedrótozott adatokból. A képekből egy véletlenül kivágott ("crop") részlet kerül megjelenítésre. A felhasználó feladata a képhez a helyes cím kiválasztása.

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

https://gist.github.com/DjCantor/d62d07b0466846e25bdd

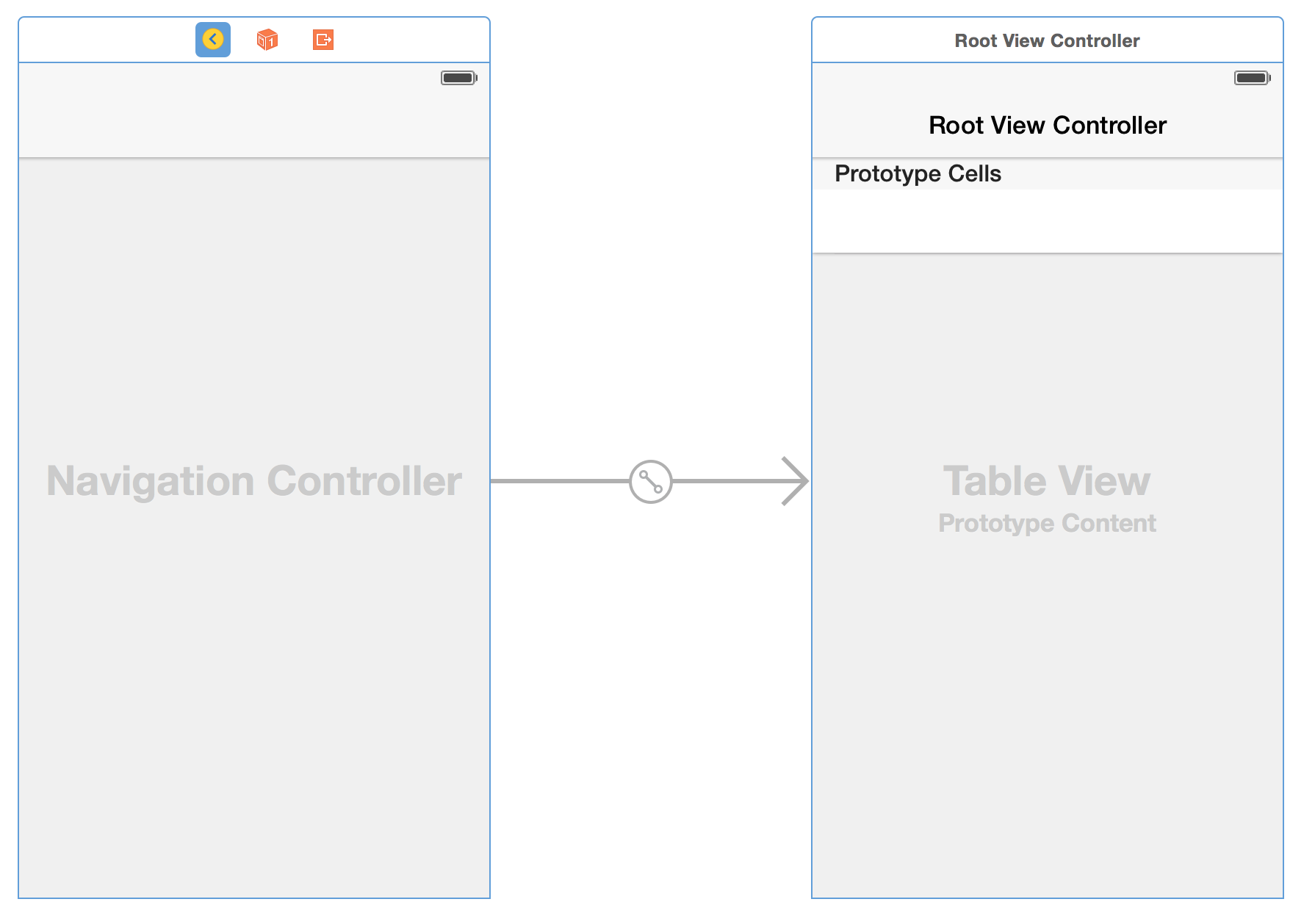
# PictureGuess

## Játékválasztó nézet

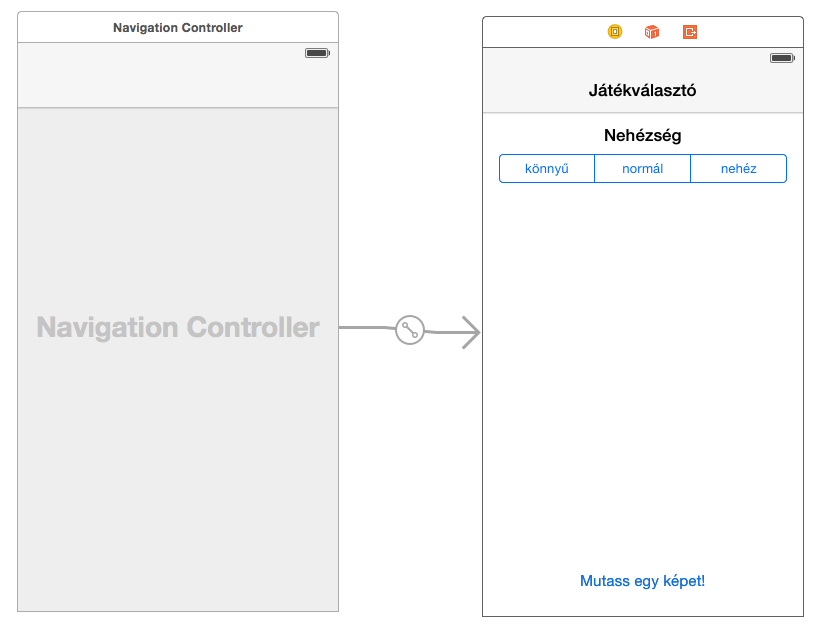
Hozzunk létre egy új "Single View" application-t, "PictureGuess" névvel, iPhone-ra és töröljük ki a létrejött ViewController.swift fájlt.

A storyboard tulajdonságainál (File Inspector) kapcsoljuk ki a Use Size Classes beállítást.

Adjunk a storyboard-hoz, egy Navigation Controller-t és állítsuk be Initial Viewcontrollernek az Attributes Inspectorban :



A Navigation Controller-hez az Xcode alapból létrehoz egy hozzácsatolt Table View Controller-t. Ezt töröljük ki és helyette adjunk a storyboard-hoz egy sima View Controller-t, majd kössük be a Navigation Controller "root view controller" Segue-ére. Ennek tartalomnézetéhez adjunk hozzá egy Label-t, egy Segmented Control-t és egy Button-t! A Segmented Control-nak három értéke legyen: "könnyű", "normál" és "nehéz". Az új View Controller-hez tartozó Navigation Item-ben nevezzük át a Title-t "Játékválasztó"-nak:



Hozzunk létre egy új osztályt, UIViewController-ből leszármaztatva, GameSelectorViewController névvel! Állítsuk be a storyboard-on belül a játékválasztó view controller osztályát GameSelectorViewController-re! Ehhez jelöljük ki a teljes nézetvezérlőt, majd Identity inspector-ban adjuk meg az osztály nevét:



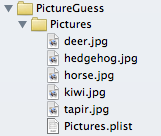
Hozzunk létre egy Outlet-et "difficultySegmentedControl" néven a Segmented Control-hoz!

## Picture Manager

Töltsük le a következő ZIP fájlt és tömörítsük ki a projekt mappájába:

**https://dl.dropbox.com/u/152439/aut-ios/PictureGuess-Pictures.zip**

Ezek után adjuk hozzá a kitömörített "Pictures" mappát a projekthez (jobb klikk Projekt navigátorban, majd "Add Files to "PictureGuess"...):



Hozzunk létre egy új, PictureManager nevű osztályt, NSObject ősosztállyal. Ez fogja tárolni a játékhoz tartozó képeket és ennek feladata lesz, hogy véletlenszerűen választott képeket és képcímeket adjon vissza.

Vegyünk fel egy pictures tagváltozót:

let pictures : [AnyObject]

Hozzunk létre egy inicializálót és töltsük be benne a tömb tartalmát a projekthez hozzáadott "Pictures.plist" fájlból:

override init() {

let filePath = NSBundle.mainBundle().pathForResource("Pictures", ofType: "plist")

pictures = NSArray(contentsOfFile: filePath!)! as [AnyObject]

super.init()

}

A plist egy Mac OS környezetben gyakran használt fájl formátum, amelyben tömböket, dictionary-ket és alap típusokat pl. string lehet egyszerű módon tárolni.

A Pictures.plist-ben minden képhez el van tárolva egy képcím és a képfájl neve:



Vegyünk fel egy új metódust, mely visszatér egy véletlenül kiválasztott képpel és képcímkék egy listájával:

/\*\*

\* picture: egy véletlenül kiválasztott kép

\* titles: véletlenszerűen kiválasztott egyedi képcímek

\* pictureTitleIndex: a kiválasztott képhez tartozó képcím index a titles tömbön belül

\*/

func getRandomPicture(inout picture:UIImage?, inout titles:[String]?, inout pictureTitleIndex:Int){

//kiválasztott kép indexe a pictures tömbben

let selectedPictureIndex = Int(arc4random\_uniform(UInt32(pictures.count-1)))

//tömb a még ki nem választott képcímek indexével

var rangeArray = [Int]()

for index in 0..<pictures.count{

rangeArray.append(index)

}

rangeArray.removeAtIndex(selectedPictureIndex)

print("range array: \(rangeArray)")

//egyedi véletlen képcímek kiválasztása

var titlesToReturn = [String]()

for \_ in 0..<kChoices-1{

let randomIndex = Int(arc4random\_uniform(UInt32(rangeArray.count-1)))

if let picture = pictures[rangeArray[randomIndex]] as? [String : String] {

titlesToReturn.append(picture["title"]!)

}

rangeArray.removeAtIndex(randomIndex)

print(rangeArray)

}

print("titles: \(titlesToReturn)")

pictureTitleIndex = Int(arc4random\_uniform(UInt32(kChoices)))

if let chosenPicture = pictures[selectedPictureIndex] as? [String:String] {

titlesToReturn.insert(chosenPicture["title"]!, atIndex: pictureTitleIndex)

picture = UIImage(named: chosenPicture["image"]!)!

}

titles = titlesToReturn

}

https://gist.github.com/DjCantor/d62d07b0466846e25bdd#file-getrandompicture

A metódus paraméterei megvannak jelölve inoutminősítővel. Erre azért van szükség, mert ezek a paraméterek kimeneti paraméterek, melyekbe egy-egy, a metóduson belül létrehozott új objektum fog beleíródni.

Egy konstans formájában adjuk meg a válaszlehetőségek számát (hány képcím közül kell kiválasztani a helyes címet):

let kChoices = 3

Vegyünk fel PictureManager típusú property-t az AppDelegate-be és inicializáljuk:

var pictureManager = PictureManager()

Ahhoz, hogy a programkódból bárhonnan könnyen elérhessük az App Delegate-et (és rajta keresztül a Picture Manager-t), vegyünk fel hozzá egy új osztálymetódust, mely visszatér az alkalmazás App Delegate-jével:

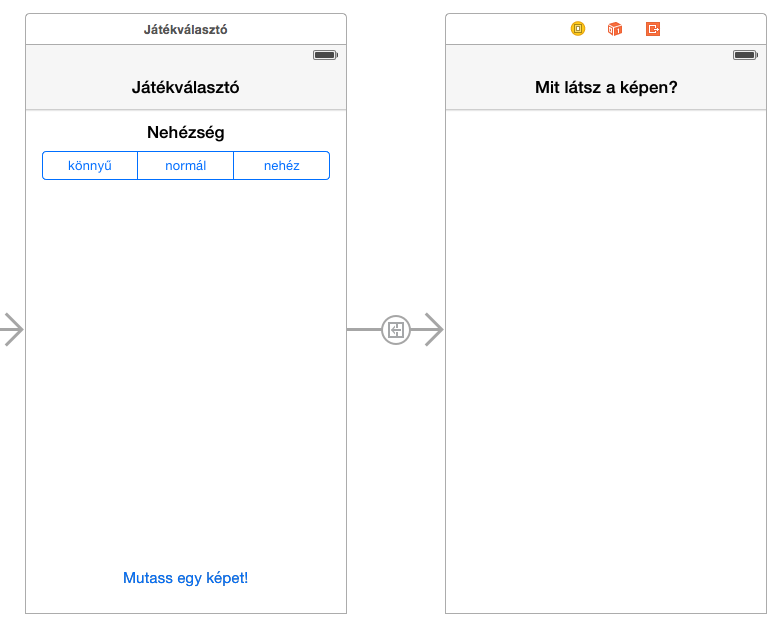
class func sharedAppDelegate() -> AppDelegate{

return UIApplication.sharedApplication().delegate as! AppDelegate

}

## Játék nézet

Vegyünk fel egy új View Controller-t a storyboard-ba, majd kössük be a játékindítás gombra egy "push" Segue-el. Módosítsuk az új View Controller-hez tartozó Navigation Item cimkéjét "Mit látsz a képen?"-re:

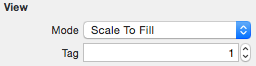


Hozzunk létre egy GameViewController nevű osztály (UIViewController ősosztállyal) és adjuk meg ezt az új View Controller osztályának az Attribute inspector-ban.

Adjunk a storyboard-hoz egy Image View-t és 3 Button-t. Ügyeljünk rá, hogy a képernyő alján maradjon némi hely (később itt még el kell férnie majd egy Tab Bar-nak):



Állítsuk be a gombok "Tag" attribútumát az Attribute inspectorban 1-3-ig (minden gomb más Tag-et kapjon):



A Tag egy integer azonosító, melyen keresztül lekérhetjük a nézetet a kódból.

Vegyünk fel két tagváltozót PGGameViewController osztályba, melybe eltárolják az éppen mutatott képet és a helyes válasz felirat sorszámát, továbbá egy outletet a képnézethez:

@IBOutlet weak var pictureView: UIImageView!

var correctAnswer = -1

var baseImage : UIImage?

Módosítsuk a PGGameViewController viewDidLoad metódusát, mely lekéri a véletlen képet és címkéket a Picture Manager-től, majd beállítja ezeket az Image View-hoz és a gombokhoz:

override func viewDidLoad() {

super.viewDidLoad()

let pictureMgr = AppDelegate.sharedAppDelegate().pictureManager

var titles : [String]?

pictureMgr.getRandomPicture(&baseImage, titles: &titles, pictureTitleIndex:&correctAnswer)

for index in 1...titles!.count{

var button = view.viewWithTag(index) as UIButton

button.setTitle(titles![index-1], forState: .Normal)

}

let cropSize = CGSize(width:300/difficultyFactor, height:200/difficultyFactor)

let cropRect = CGRect(x:CGFloat(random() % Int(baseImage!.size.width - cropSize.width)), y:CGFloat(random() % Int(baseImage!.size.height - cropSize.height)), width:(cropSize.width\*baseImage!.scale), height:(cropSize.height\*baseImage!.scale))

let croppedImageRef = CGImageCreateWithImageInRect(baseImage!.CGImage, cropRect)

let croppedImage = UIImage(CGImage: croppedImageRef, scale: baseImage!.scale, orientation: baseImage!.imageOrientation)

pictureView.image = croppedImage

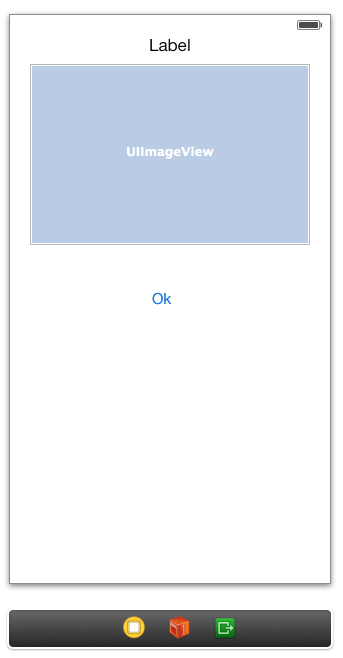
}

https://gist.github.com/DjCantor/d62d07b0466846e25bdd#file-viewdidload

Próbáljuk ki az alkalmazást!

## Eredmény nézet

Hozzunk létre egy View Controller-t a storyboard-ban, majd vegyünk fel bele egy Label-t, egy Image View-t és egy Button-t:



Hozzunk létre egy egy ResultsViewController nevű osztályt, melyet rendeljünk az imént létrehozott ViewController-hez a storyboard-on belül. Rendeljünk egy-egy Outlet-et a Label-hez és az Image View-hoz (pl. resultsLabel, pictureView).

@IBOutlet weak var resultsLabel: UILabel!

@IBOutlet weak var pictureView: UIImageView!

A View Controller által mutatott tartalom inicializálásához vegyünk fel két propertyt, egyet a bemutatott képhez, egyet pedig a felirathoz. Ezekre azért van szükség, mert a View Controller tartalmát még az előtt be kell majd állítanunk, hogy az ahhoz tartozó nézetek megjelennének (tehát nem tudjuk közvetlenül a Label-t és az Image View-t módosítani):

var picture: UIImage?

var caption : String?

Ezen property-k alapján inicializáljuk a nézeteket a View Controller viewDidLoad metódusában:

- (void)viewDidLoad

{

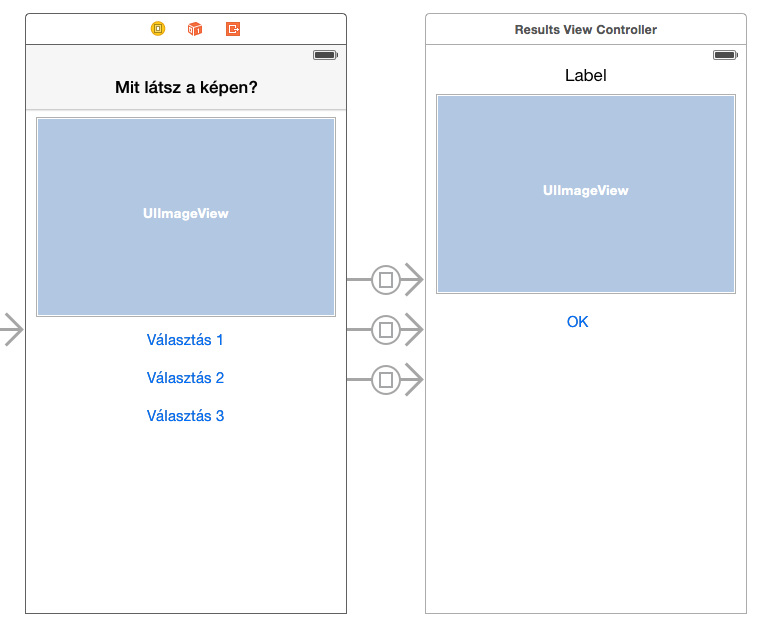
[super viewDidLoad];

resultsLabel.text = caption!

pictureView.image = picture!

}

ResultsViewController megjelenítése modális segue-en keresztül fog történni, kössük be a játék jelentből **mind a három gombot** egy "modal" segue-el az eredményeket tartalmazó View Controller-hez (Ctrl+klikk+drag a gombról a cél View Controller-re):



Ezzel azt értük el, hogy megjelenik az új nézet, de még teljesen üres, hisz adatokat nem adunk át neki. Ahhoz, hogy egy lejátszódó Segue előtt adatokat adhassunk át a megjelenő View Controller-nek, felül kell definiálnunk a kiinduló View Controller (esetünben GameViewController) prepareForSegue:sender: metódusát, melyet UIViewController-ből örökölt. A metódus sender paramétere a segue-t elindító gombra fog mutatni, a segue "célját", a destinationViewController property-vel tudjuk elérni. Ezen keresztül fel tudjuk tölteni adatokkal a Results View Controller-t. A helyes válaszhoz tartozó gombot a gombokhoz rendelt "Tag" és az eltárolt \_correctAnswerIndex alapján kapjuk meg:

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

if let button = sender as? UIButton {

let resultsVC : ResultsViewController = segue.destinationViewController as ResultsViewController

resultsVC.picture = baseImage

if button.tag-1 == correctAnswer {

resultsVC.caption = "Talált ez egy \(button.titleLabel!.text!.lowercaseString)"

}

else{

let correctButton = view.viewWithTag(correctAnswer+1) as? UIButton

resultsVC.caption = "Sajnos nem talált! Ez egy \(correctButton!.titleLabel!.text!.lowercaseString)"

}

}

}

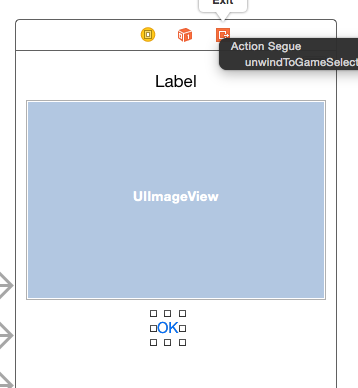
https://gist.github.com/DjCantor/d62d07b0466846e25bdd#file-prepareforsegue

Ezek után már megkapjuk az eredményt, de sajnos még nem tudjuk bezárni a Results jelenetet (az Ok gomb nem reagál).

Ahhoz, hogy bezárjuk a modális VC-t és visszlépjünk egy korábbi VC-re, egy Unwind Segue-t fogunk használni. Vegyünk fel egy új akció metódust az IPGameSelectorViewController-be, melyet az Unwind Segue-hez kapcsolahatunk (azért a Game Selector-ba vesszük fel, mert rögtön ide szeretnénk visszauggrani, először bezárva a modális VC-t, majd visszalépve a Navigation Controller-en):

@IBAction func unwindToGameSelector(segue:UIStoryboardSegue){

}



Az Unwind Segue-ek olyan korábbi VC-kre tudnak visszatérni, melyekben található egy (IBAction)-el visszatérő metódus, mely egy UIStoryboarSegue paraméter-t vár. Ez a metódus meghívódik az Unwind Segue bekövetkezte előtt és paraméterül kapott segue-ből kiolvasható a kiindulási VC (segue.sourceViewController), amitől átvehetnénk az esetleg szükséges adatokat.

## Nehézség választó

Vegyünk fel egy új property-t GameViewController-be a nehézség meghatározásához:

@property (assign, nonatomic) double difficultyFactor;

Módosítsuk a viewDidLoad metódusban a cropSize kiszámítását oly módon, hogy difficultyFactor-t is belekalkulálja:

let cropSize = CGSize(width:300/difficultyFactor, height:200/difficultyFactor)

Valósítsuk meg PGGameSelectorViewController prepareForSegue:sender: metódusát, hogy a játéknézetre váltás előtt beállítsa a nehézséget a Segmented Control indexe alapján:

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

let gameVC = segue.destinationViewController as! GameViewController

gameVC.difficultyFactor = Double(difficultySegmentedControl.selectedSegmentIndex) + 1.0

}

Egy konkrét View Controller-ről kiinduló összes Segue hatására ugyanaz a prepareForSegue metódus hívódik meg. Ha egynél több Segue-t indítunk ugyanarról a Controller-ről, akkor szükség lehet a Segue azonosítására. Ehhez a storyboard-ban megadhatjuk a Segue azonosítóját:



Majd a kódban tudjuk ellenőrizni, hogy melyik az épp bekövetkező Segue:

if segue.identifier == "gameViewControllerSegue" { ...

## Játékstatisztika

Vegyünk fel 3 új property-t PictureManager-be, melyek eltárolják a lekért képek számát, a helyes és rossz válaszok számát:

var picturesQueriedCount = 0

var correctAnswerCount = 0

var wrongAnswerCount = 0

Módosítsuk getRandomPicture metódust, hogy növelje a lekért képek számát:

func getRandomPicture(inout picture:UIImage?, inout titles:[String]?, inout pictureTitleIndex:Int){

picturesQueriedCount++

Módosítsuk PGGameViewController prepareForSegue: metódusát a helyes és helytelen válaszok számának frissítésével:

if button.tag-1 == correctAnswer {

resultsVC.caption = "Talált ez egy \(button.titleLabel!.text!.lowercaseString)"

AppDelegate.sharedAppDelegate().pictureManager.correctAnswerCount++

}

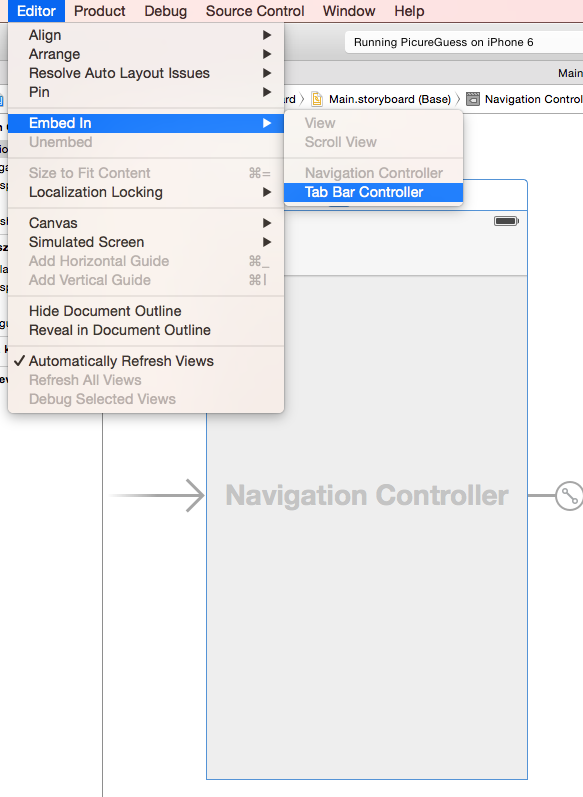
else{

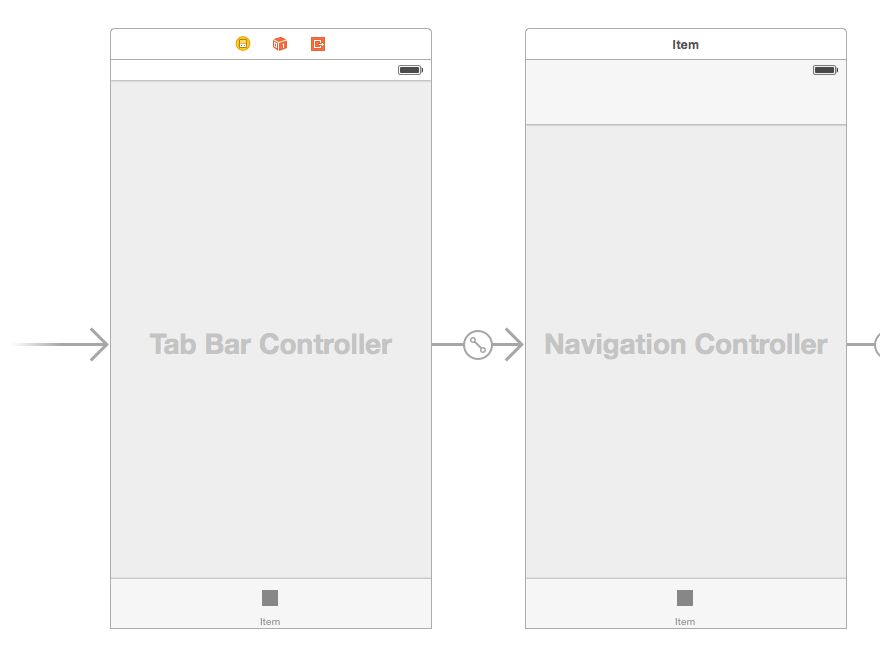
resultsVC.caption = "Sajnos nem talált! Ez egy \(button.titleLabel!.text!.lowercaseString)"

AppDelegate.sharedAppDelegate().pictureManager.wrongAnswerCount++

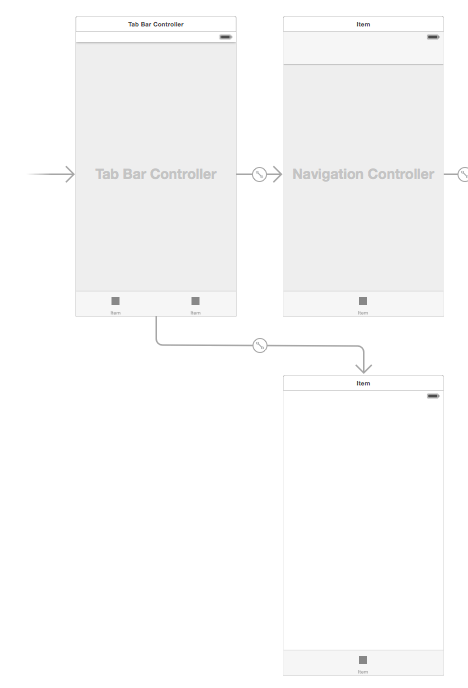
}

A storyboard-ban hozzunk létre egy új Tab Bar Controller-t és rendeljük hozzá a Navigation Controller-t. Ezt legegyszerűbben úgy tehetjük meg, hogy kiválasztjuk a Navigation Controller-t, majd a menüben az Editor/Embed In/Tab Bar Controller-t választjuk. Ennek hatására létrejön egy új Tab Bar Controller, ami magához rendeli (relationship segue) a Navigation Controller-t, így az megjelenik az első tabon:

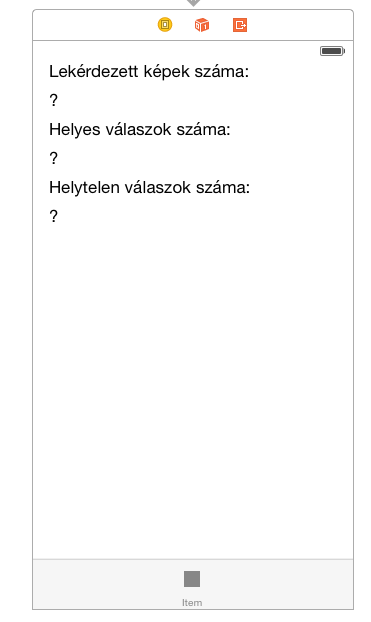




Hozzunk létre egy új View Controller-t a storyboard-ban és kössük be a Tab Bar controller "view controllers" segue-éhez:



Készítsünk egy nézetet a statisztikák megjelenítéséhez! Mindenképp legyen 3 UILabel-ünk az egyes adatokhoz, ezen kívül tetszőleges egyéb elemeket felvehetünk:



Hozzunk létre egy új UIViewController leszármazott osztályt PGStatsViewController névvel és rendeljük ezt a storyboard-ban imént létrehozott View Controller-hez az Identity inspectorban.

Vegyünk fel 3 Outlet-et 3 statisztikát megjelenítő Label-hez:

@IBOutlet weak var queriedPicturesCountLabel: UILabel!

@IBOutlet weak var correctAnswerCountLabel: UILabel!

@IBOutlet weak var wrongAnswerCountLabel: UILabel!

Definiáljuk felül az új View Controller viewWillAppear:animated: metódusát, mely mindíg meghívódik mikor az adott View Controller nézete éppen megjelenik a képernyőn (pl. a felhasználó a nézet Tab-jára vált). Módosítsuk a Label-ek szüvegét a Picture Manager-ből lekérdezett statisztikák alapján.

override func viewWillAppear(animated: Bool) {

let pictureMgr = AppDelegate.sharedAppDelegate().pictureManager

queriedPicturesCountLabel.text = "\(pictureMgr.picturesQueriedCount)"

correctAnswerCountLabel.text = "\(pictureMgr.correctAnswerCount)"

wrongAnswerCountLabel.text = "\(pictureMgr.wrongAnswerCount)"

super.viewWillAppear(animated)

}

A storyboard szerkesztőben állítsuk be a két tabhoz tartozó View Controller-ben lévő Tab Bar Item feliratát "Játék"-ra és "Statisztika"-ra.

Töltsük le a következő ZIP fájlt és adjuk a benne lévő "Images" könyvtárat a projekthez:

**https://dl.dropbox.com/u/152439/aut-ios/PictureGuess-Images.zip**

Állítsuk be a két Tab Bar Item "Image" beállítását "game.png"-re és "stats.png"-re:

