iOS alapú szoftverfejlesztés - Labor 07

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

A labor témája:

* Auto Layout
* NSUserDefaults
* ScrollView

A labor célja az Auto Layout használatának a gyakorlása egy alkalmazás kezdeti képernyőin keresztül.

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

https://gist.github.com/DjCantor/c55d7a3be17d6c23d744

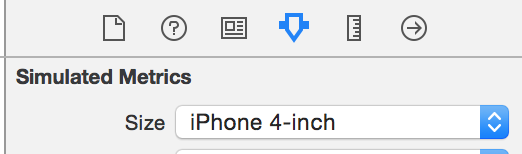
# Az Auto Layout bemutatása

## Bejelentkezési ablak létrehozása

Hozzunk létre egy új Single View Applictaion projektet iAUT névvel és töltsük le az képi erőforrássokat a következő URL-ről és rakjuk be a projektbe a mappát:

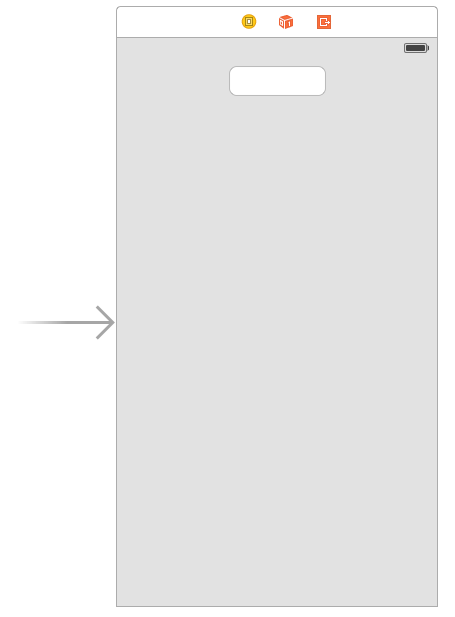
***https://dl.dropboxusercontent.com/u/82267346/iOSLabor07Resources.zip***

A Storyboard-ban a létrejött ViewController méretét állítsuk át 4 incses iPhone-ra a Size Inspector Simulated Metrics tulajdonságánál.



Ezzel egy iPhone 5-ös képernyőjének a méretével fogunk dolgozni, legalábbis, ezt szimulálja majd a Storyboard Editor.

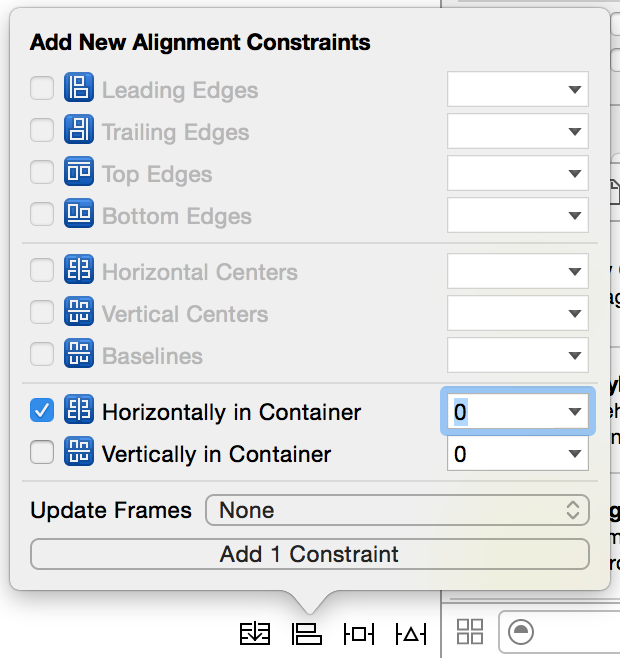
A létrejött ViewControlleren lévő View hátterét állítsuk be valamilyen sötétebb színre, majd tegyünk be felülre egy UITextFieldet vízszintesen középre használva a segéd vonalakat:



Futtassuk le az alkalmazást a szimulátorban és forgassuk el a kijelzőt, ehhez nyomjuk le a Cmd gombot és valamelyik horizontális nyilat attól függően, hogy milyen irányba szeretnénk fordítani a szimulátort.

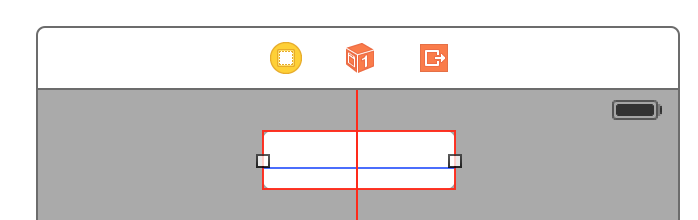
A Text Field sajnos nem maradt középen fekvő módban. Ha szeretnénk, hogy középen maradjon, akkor ehhez szükség van kényszerekre.

Ahhoz, hogy horizontálisan középen maradjon adjunk hozzá egy kényszert, ami ezt biztosítja. Ehhez jelöljük ki és az Align menüből válasszuk ki a Horizontal Center In Container-t:



Kattintsunk az Add 1 Constaint-ra!

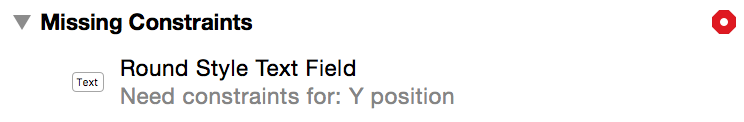
Ekkor megváltozik a nézetvezérlő képe piros vonalak jelennek meg.



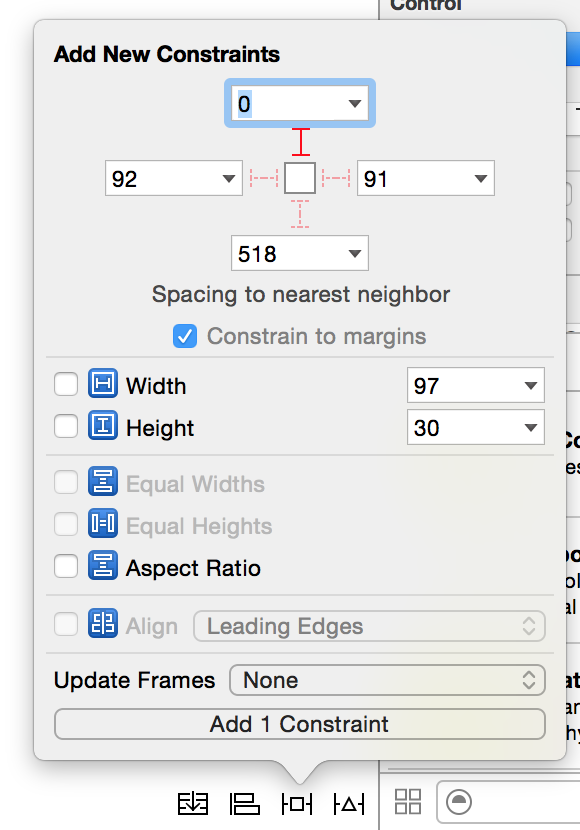
Piros vonallakkal jelezve a kényszer hiányzt. Ennek az, az oka, hogy amíg nem adtunk hozzá kényszert, a rendszer fordítási időben legenerálja az abszolút pozíció és méret kényszereket. Viszont azzal, hogy már egy kényszert hozzáadtunk manuálisan a rendszer nem tudja, hogy milyen egyéb kényszereket generáljon magától. Ahogy ezt a Document Outline is pontosan mutatja



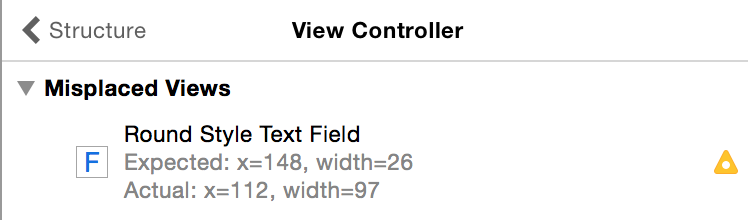
Kattintsunk a piros nyílra:



A hiba, hogy függőlegesen nem tudja meghatározni a pozícióját, ehhez a Pin menüből állítsunk be a szülő nézet tetejétől mért távolságát:

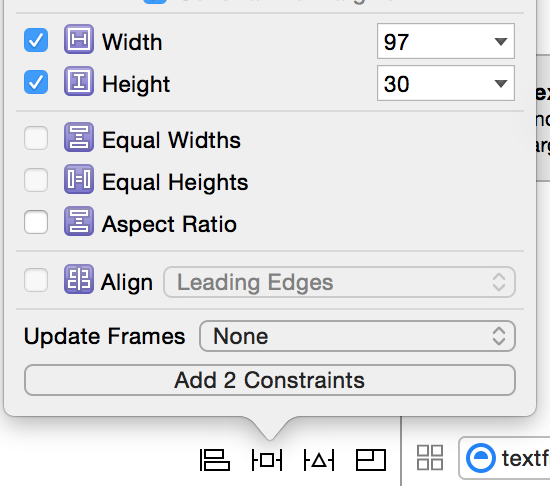


Továbbra is figyelmeztetést kapunk:



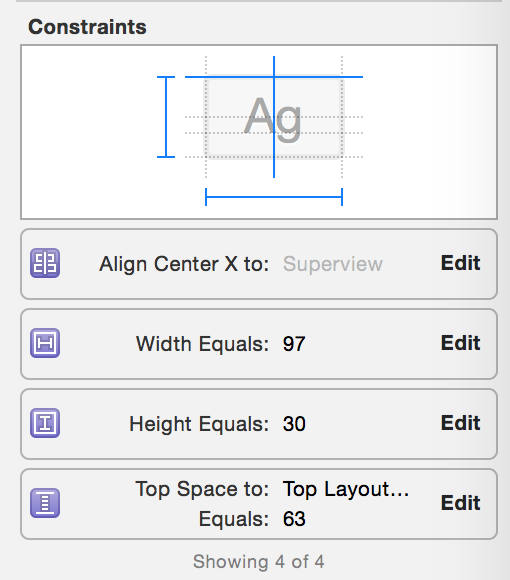
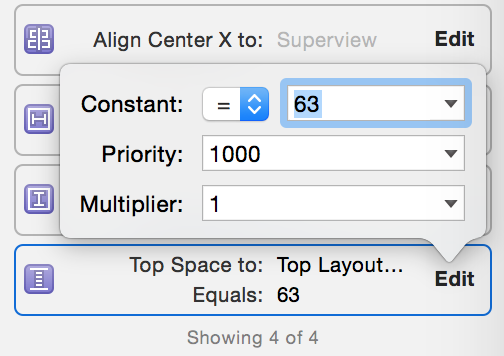
A hiba, hogy nem adtunk meg kényszereket a szövegdoboz méretére vonatkozóan.

Rögzítsük a méretét a Pin menü segítségével:

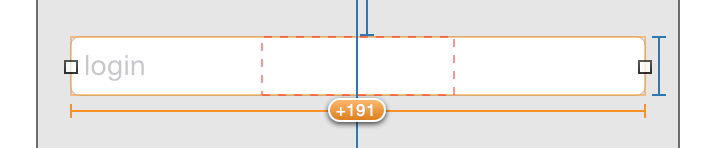


Futtassuk az alkalmazást és forgassuk is el a szimulátort!

Állítsuk be a textfield tulajdonságainál placeholdernek a login feliratot, majd tegyük lejjebb. Jelöljük ki, majd módosítsuk a Top Layout Guide-hoz rögzített kényszert a Size Inspectorban, konstansnak állítsunk be 160-at!

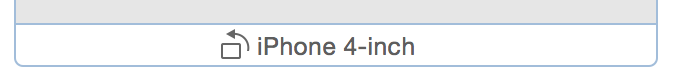
Növeljük meg a TextField méretét egészen a szaggatott vonalakig mindkét irányba:



Mivel explicit méret kényszerek vannak, így a narancssárga vonalakkal azt jelzi a rendszer, hogy futási időben hová fog kerülni az elem. Ahhoz, hogy megtudjuk, hogy pontosan hogyan néz ki az adott képernyő futási időben, nem muszáj folyton a szimulátorban futtatni az alkalmazást. Nyissuk meg az Assistant Editor-t és a felső menüből válasszuk ki a Preview funkciót!



Az előnézetet a képernyő alatti ikonnal tudjuk elforgatni:

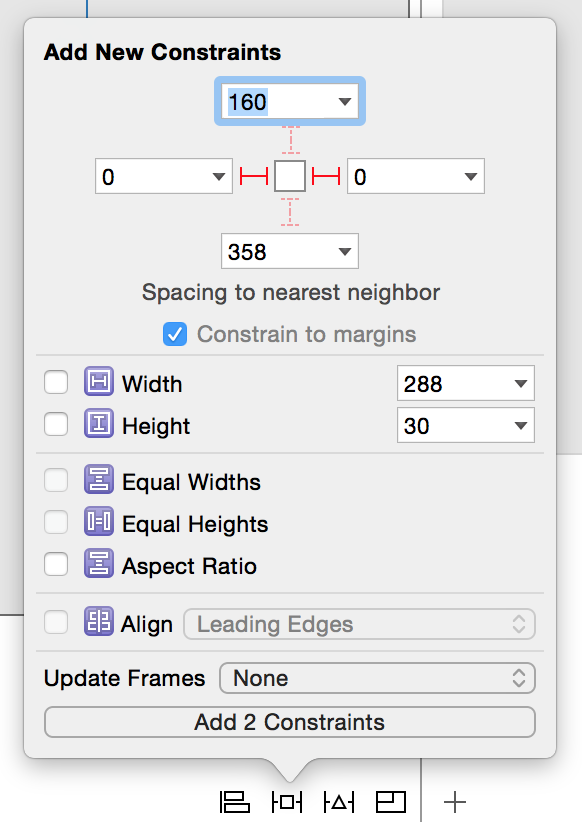


A bal alsó sarokban található + gombbal adjunk hozzá egy másik 4 inches Preview-t és így mindkét tájolásban látjuk, hogy fog kinézni az alkalmazás.

Látjuk, hogy hiába növeltük meg a méretet a Storyboard editorban, a kényszerek determinálják majd futási időben a méretet.

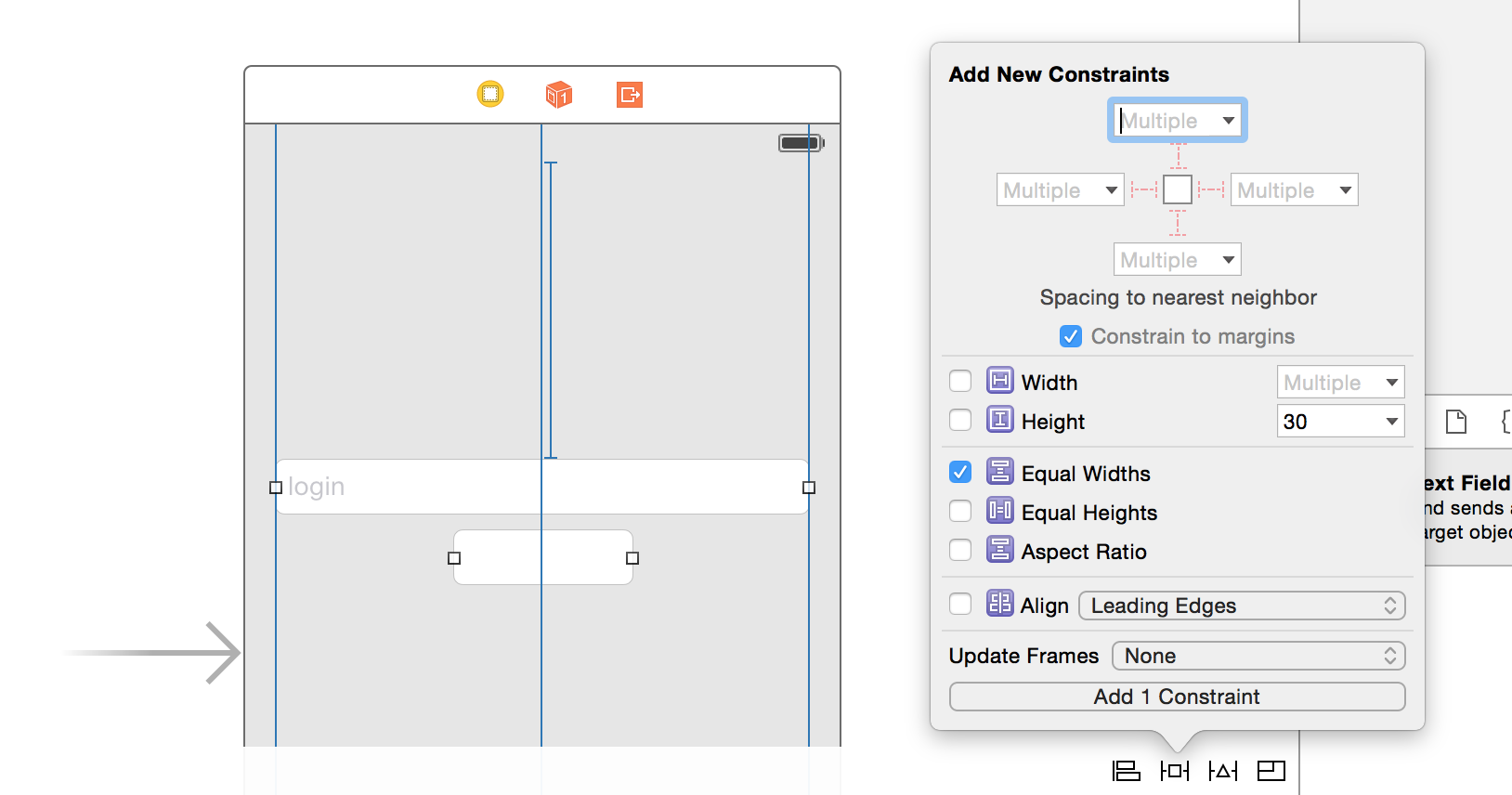
Töröljük ki a méretre vonatkozó kényszereket, ehhez jelöljük ki őket tervezési területen, majd backspace vagy a Size Inspectorban jelöljük és utána backspace.

Rögzítsük a szülő nézet széleitől mért távolságot:

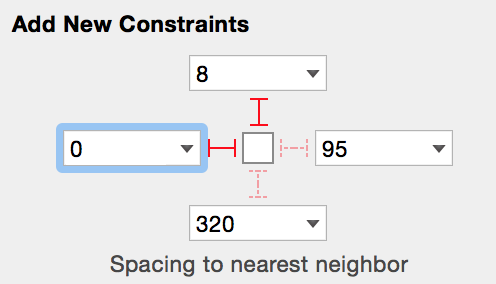


Tegyünk be alá egy másik TextField-et az alsó szaggatott vonalhoz igazítva.

Jelöljük ki mindkét textfield-et és a Pin menüből állítsuk be, hogy ugyanakkora a szélességük:

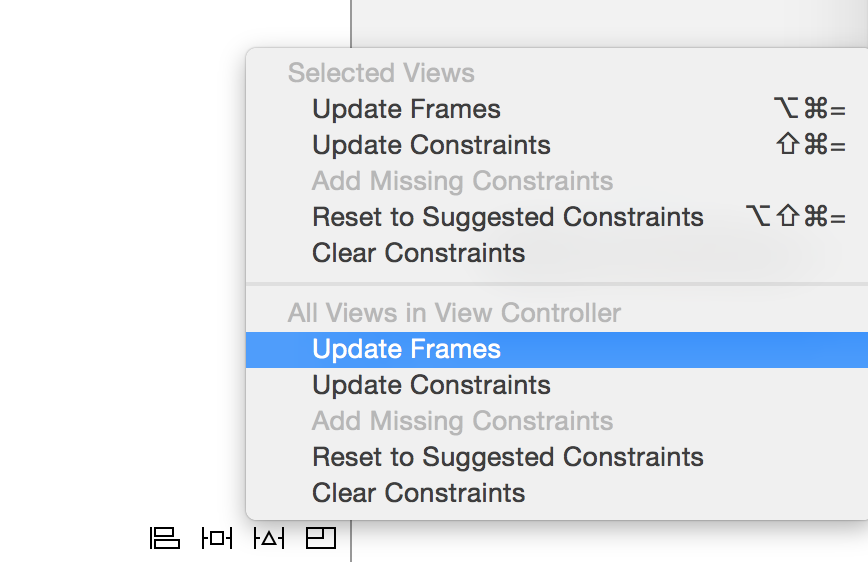


Majd jelöljük ki az alsót és rögzítsük a felette lévő textField-től mért távolságát 8-ra és a balszéltől mért távolságot 0-ra:



Az Assistant Editorban látjuk, hogy jó helyre fog kerülni a Text Field, de a Storyboard editorban még nem megfelelő, amit a narancssárga vonalak jeleznek is.

Ahhoz, hogy a megfelelő pozícióba kerüljön a megfelelő mérettel a Resolve Auto Layout Issues menüből válasszuk ki az Update Frames-t:



A TextField tulajdonságainál állítsuk be Placeholder-nek a password-öt, és pipáljuk be a Secure Text Entry-t!

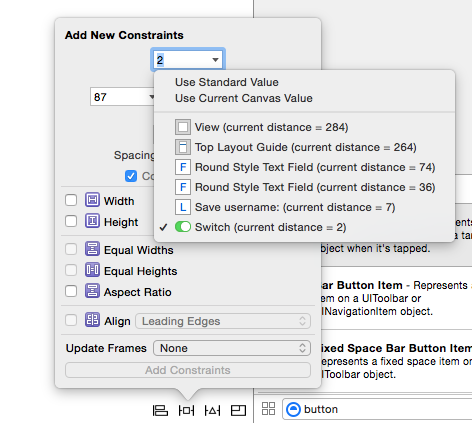
Rakjunk be egy labelt az alsó Text Field alá, a szövege legyen "Save username": A pozíció és méret meghatározásához használjuk a következő kényszereket:

1. A bal oldalát igazítsuk a felette lévő TexField-hez (***Align / Leading Edges***),
2. A köztük lévő távolságot rögzítsük (***Pin / Vertical Spacing***) értéke legyen standard (8 egység)!

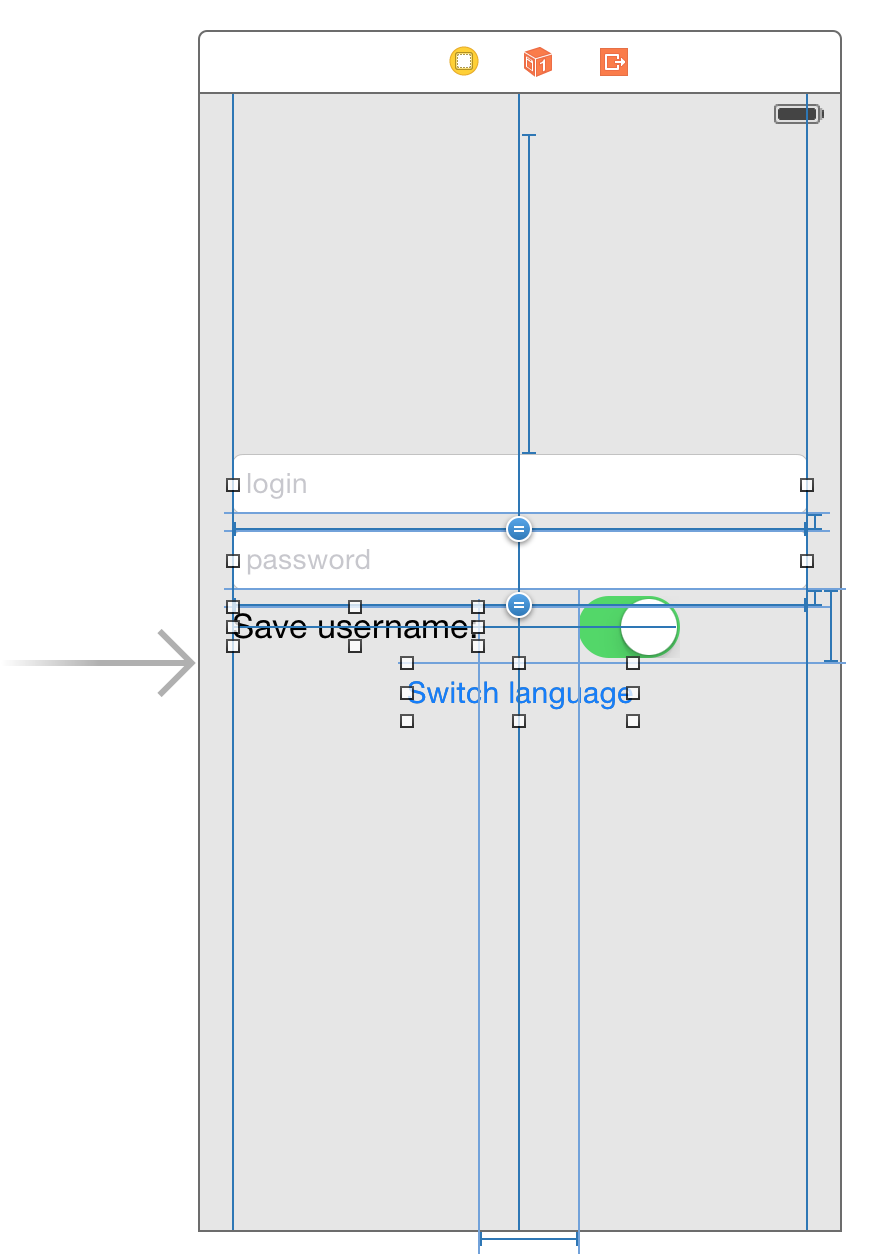
Ezek után rakjunk be a label mellé egy UISwitch-et, állítsuk be a következőket:

1. A két elem közepe, mindig egy vonalban legyen (***Align / Vertical Centers***)
2. köztük mindig 50 egységnyi távolság legyen, ehhez (***Pin / Horizontal*** ***Spacing***), majd jelöljük ki a köztük lévő kényszert és a tulajdonságainál a konstansát állítsuk át 50-re.

Végül emeljünk be egy új gombot switch language, az elemek alá középre, és rögzítsük a távolságot a felette lévő TextField-hez képest 36 egységre (***Pin / Vertical Spacing***). Ehhez válasszuk ki a megfelelő Texfield-et a Pin menüben:



Ez utóbbihoz továbbá a középre rendezést (***Align / Horizontal Center In Container***).



Hozzunk létre egy új LoginViewController-t és állítsuk ezt be a Storyboardban a nézetvezérlő osztályának. Hozzunk létre egy outletet a labelnek saveUserNameLabel néven. Vegyünk fel egy Bool típusú, „english” nevű változót, amit inicializáljuk true-val.

Majd írjuk meg a gomblenyomás akció metódusát:

@IBAction func loginButtonTouchUpInside(sender: AnyObject) {

if !english {

saveUsernameLabel.text = "Save username:"

}

else{

saveUsernameLabel.text = "Benutzernamen speichern:"

}

english = !english

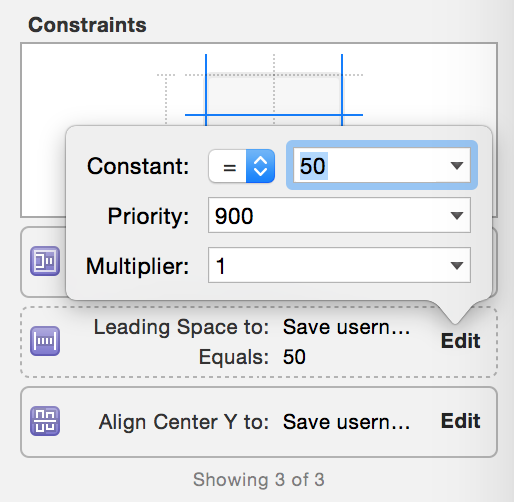
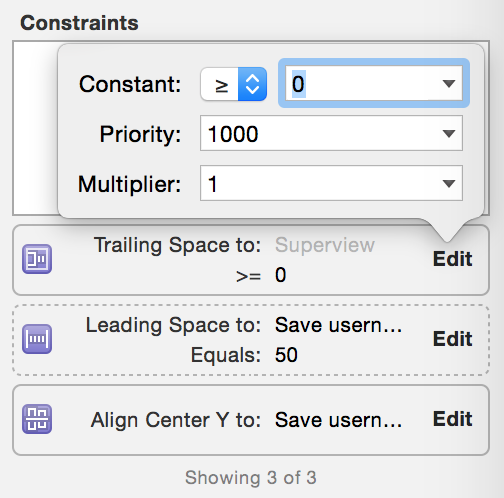
}

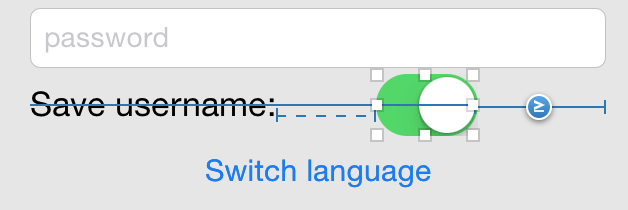
Futtassuk le az alkalmazást, elforgatva is jól néz ki! Azonban ha rákattintunk a Switch Language gombra, akkor a Switch álló módban kilóg, pedig nem történt más, mint kipróbáltuk az alkalmazást német szöveggel.

Szerencsére az Auto Layout segítségével könnyedén tudjuk kezelni a dinamikus változásokat is a kényszerek prioritásai segítségével. Itt olyan kényszerekre van szükségünk, hogy “*ha lehetőség van rá, akkor a label és a switch közti távolság legyen 50, de a switch és a képernyő széle között mindig legyen legalább 20 egységnyi távolság*”. Az első kényszernek kisebb lesz a prioritása, mert a második kényszer fontosabb.

Ehhez jelöljük ki a Switchet és rögzítsük a view jobb széle és a közte lévő távolságot (***Pin / Trailing Space To Superview***), állítsuk be, hogy az érték legalább ekkora legyen (Relation: Greater Than Or Equal) illetve, állítsuk be, hogy a konstansa 0!

A kettő közti távolság kényszerének a prioritását meg vegyünk 900-ra!

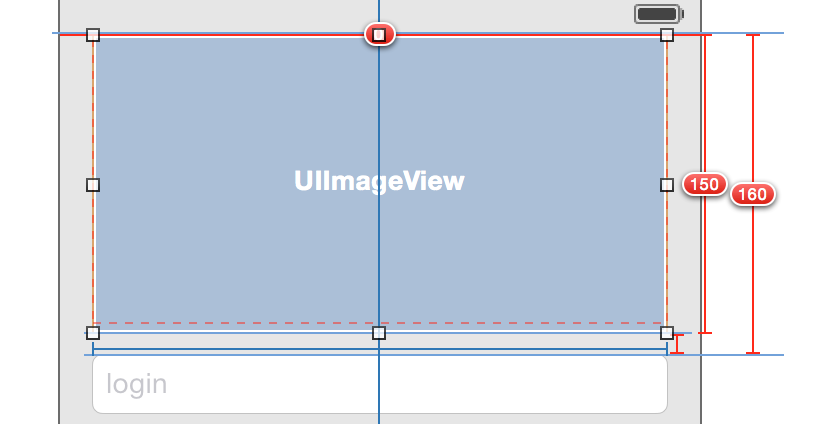


Próbáljuk ki az alkalmazást újra!

Tegyünk be egy Image View-t a Text Field fölé 288 x 150-es méretben középre. Adjunk hozzá egy középre rendezést biztosító kényszert (***Align / Horizontal Center In Container***), majd pedig rögzítsük a méreteit (***Pin / Width és Height***) és a Top Layout Guide-tól mért távolságot ez legyen 0 (***Pin / Top Layout Guide***).

Ezek után a felső Text Fieldhez adjuk hozzá egy kényszert, ami a képtől mért távolságot rögzíti 15 egységre.

Ez ellentmond a Text Field másik kényszerének, ami Top Layout Guide-tól miért távolságát rögzíti, ezt piros színnel jelzi a rendszer:



Ha futtatjuk az alkalmazást, akkor hiba nélkül fut, ugyanakkor konzolban látjuk, hogy a kényszereink egy időben nem teljesíthetők.

Töröljük hát a Top Layout Guide és a Text Field teteje közti távolság kényszert! Mivel minden elem a felső Text Field-hez igazodik, így rossz helyen vannak, ahogy azt a Document Outline is mutatja.

Ahhoz, hogy minden elem a helyére kerüljön, jelöljük ki a teljes ViewController-t és a Resolve Auto Layout Issues menüből válasszuk az Update frames-t!

Képnek állítsuk be a sötétebbik Aut logót!

Futtassuk az alkalmazást és kattintsunk bele a TextField-ekbe. Szerencsére a billentyűzet nem takarja ki egyiket sem. Legalábbis portrait módban... Landscape állapotban mindkettőt.

A legegyszerűbb megoldás, ha billentyűzet megjelenésekor minden elemet feltolunk. Függőlegesen minden elem valójában a ImageView-hoz igazodik, így ennek a pozícióját kell megváltoztatni annak megfelelően, hogy megjelenik a billentyűzet.

Vegyünk fel egy outletet a kép Top Layout Guide kényszeréhez, ehhez jelöljük ki a kényszert, majd a szokásos módon húzzuk át a fájlba az Assistant Editorban az outlet neve legyen imageViewTopConstraint.

Továbbá vegyünk fel egy outletet az alsó textFieldhez passwordTextField néven.

Ezek után iratkozzunk fel a billentyűzet megjelenése és eltűnése rendszer eseményekre a viewWillAppear és viewVillDisapear metódusokban:

override func viewWillAppear(animated: Bool) {

NSNotificationCenter.defaultCenter().addObserver(self, selector: "keyboardWillShow:", name: UIKeyboardWillShowNotification, object: nil)

NSNotificationCenter.defaultCenter().addObserver(self, selector: "keyboardWillHide:", name: UIKeyboardWillHideNotification, object: nil)

}

override func viewWillDisappear(animated: Bool) {

NSNotificationCenter.defaultCenter().removeObserver(self)

}

https://gist.github.com/DjCantor/c55d7a3be17d6c23d744#file-viewappearingfunctions

Valósítsuk meg a két függvényt:

func keyboardWillShow(notification: NSNotification) {

if let userInfo = notification.userInfo {

if let keyboardSize = (userInfo[UIKeyboardFrameBeginUserInfoKey] as? NSValue)?.CGRectValue() {

if (self.passwordTextField.frame.origin.y + self.passwordTextField.frame.size.height) > (self.view.frame.size.height - keyboardSize.height) {

imageViewTopConstraint.constant = -((self.passwordTextField.frame.origin.y + self.passwordTextField.frame.size.height) - (self.view.frame.size.height - keyboardSize.height))

}

}

}

}

func keyboardWillHide(notification: NSNotification) {

imageViewTopConstraint.constant = 0

}

https://gist.github.com/DjCantor/c55d7a3be17d6c23d744#file-keyboardfunctions

Végül a TextField eltűntetésért felelő DidEndOnExit esemény metódust (amit el ne felejtsünk bekötni a TexField-ekhez!

@IBAction func editingDidEndOnExit(sender: UITextField) {

sender.resignFirstResponder()

}

Sokat javít a felhasználói élményen, ha a konstans beállítás nem azonnal, hanem animálva történik:

...

self.view.layoutIfNeeded()

UIView.animateWithDuration(0.5, animations: {

self.imageViewTopConstraint.constant = -((self.passwordTextField.frame.origin.y + self.passwordTextField.frame.size.height) - (self.view.frame.size.height - keyboardSize.height))

self.view.layoutIfNeeded()

})

...

self.view.layoutIfNeeded()

UIView.animateWithDuration(0.5, animations: {

self.imageViewTopConstraint.constant = 0

self.view.layoutIfNeeded()

})

## Adatok elmentése

Auto Layouttól való pihenés képpen valósítsuk meg, hogy a felhasználónév mentése valóban működjön! Ehhez a NSUserDefaults-ot fogjuk használni az adatok tárolására. Először változtassuk meg a gomb címkéjét Loginra.

Hozzunk létre a Switch-hez egy saveUsernameSwitch outletet. A gombnyomás implementációját cseréljük le:

@IBAction func loginButtonTouchUpInside(sender: AnyObject) {

let alertController = UIAlertController(title: "Succesfull login!", message: "Congratulations!", preferredStyle: .Alert)

let defaultAction = UIAlertAction(title: "OK", style: .Default, handler: nil)

alertController.addAction(defaultAction)

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

if saveUsernameSwitch.on {

NSUserDefaults.standardUserDefaults().setBool(true, forKey: "usernameSaved")

NSUserDefaults.standardUserDefaults().setValue(self.usernameTextField.text, forKey: "username")

}

else{

NSUserDefaults.standardUserDefaults().setBool(false, forKey: "usernameSaved")

}

NSUserDefaults.standardUserDefaults().synchronize()

}

Végül a viewDidLoad-ot is frissítsük:

override func viewDidLoad() {

super.viewDidLoad()

self.saveUsernameSwitch.setOn(NSUserDefaults.standardUserDefaults().boolForKey("usernameSaved"), animated: false)

if (saveUsernameSwitch.on){

self.usernameTextField.text = NSUserDefaults.standardUserDefaults().valueForKey("username") as String

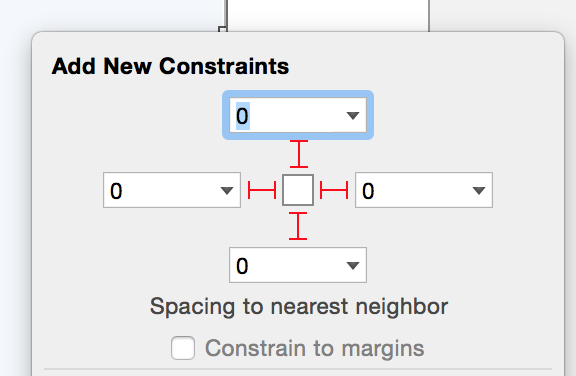
}

}

Próbáljuk ki az alkalmazást! Ne felejtsünk el rányomni a gombra a mentéshez.

## Scroll View használata

Tegyünk be egy új ViewController-t amit állítsunk be kezdő ViewControllernek (Initial View Controller) a gyökérnézetbe helyezzünk be egy UIScrollView-t, úgy, hogy kitöltse a teljes rendelkezésre álló területet és állítsunk be olyan kényszereket, hogy minden oldala és a szülő nézet oldala közötti távolság 0 legyen!

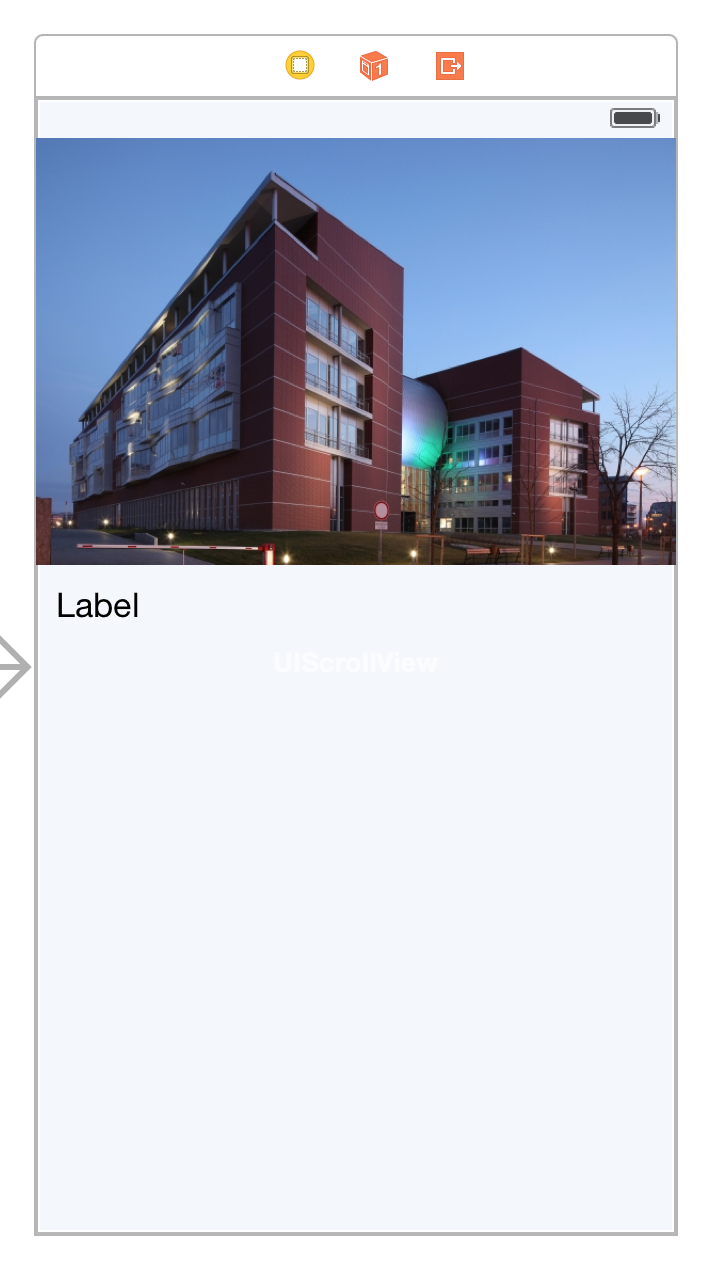


Tegyünk be egy ImageView-t felülre és állítsuk be háttérképnek a BMEQBuilding.jpeg képet!

Az ImageView-n alkalmazzuk a következő kényszereket:

* A Top Layout Guide-tól miért távolsága legyen 20 egység (***Pin / Top Spacing***)
* Rögzítsük a kép arányait (***Pin / Aspect Ratio***)
* A szélessége legyen egyenlő a gyökér View szélességével

Helyezzünk be egy labelt-t a kép alá!



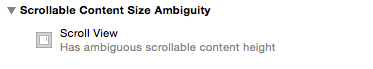
A label tulajdonságainál állítsuk be, hogy több soros szöveget tartalmaz, ehhez a line break mode-ot állítsuk word wrap-re, a line number-t pedig 0-ra.

Alkalmazzuk a következő kényszereket a labelen:

* A label és az imageView között legyen 10 egység
* A label és a szülő nézet szélei között legyen 10 egység

A label szövegének állítsuk be a Aut portálon lévő Rólunk szöveget!

Ha futtatjuk akkor a nézet nem görgethető... Erre a Storyboard Editor is felhívja a figyelmet:



Ahhoz, hogy a Scroll View ki tudja számolni, hogy görgethetővé kell-e tennie a benne lévő területet meg kell tudnia pontosan határozni, hogy mekkora a benne lévő tartalom.

Ehhez:

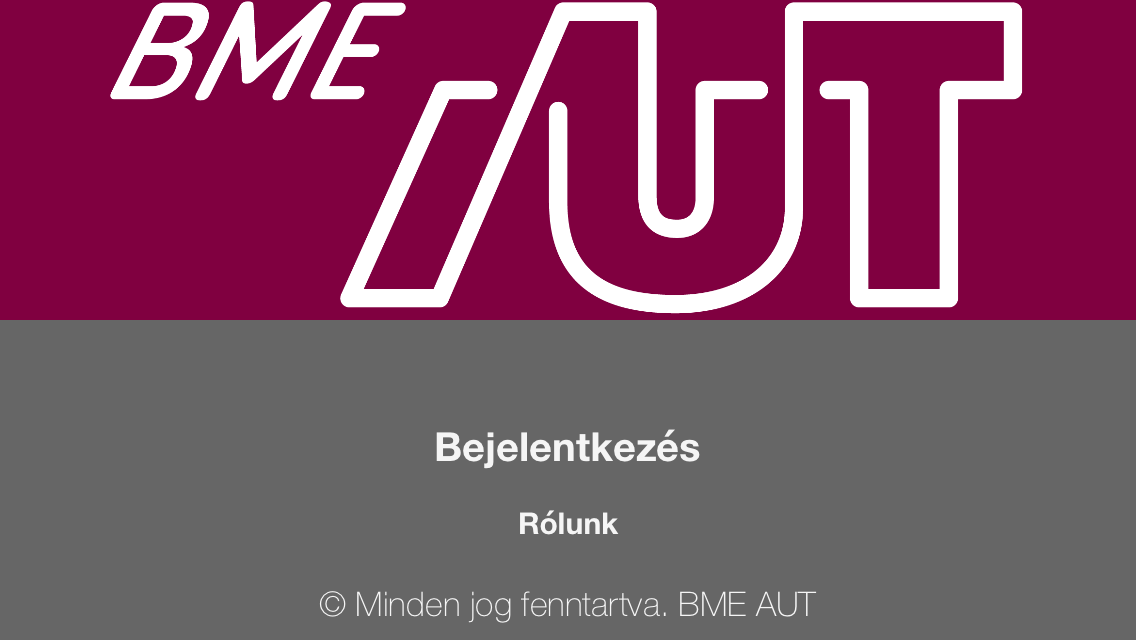
* Rögíztsük az ImageView széleit a szülő nézet széleihez
* A label alját 10 egységnyire a szülő nézet aljához.

Próbáljuk ki az alkalmazást! Forgassuk el, látjuk, hogy mindkét formában scrollozható.

Ágyazzuk be egy navigaton controller-be a nézetvezérlőt. Ekkor a teljes nézet lejjebb csúszik, mivel megváltozik a TopLayoutGuide.

## Önálló feladat: kezdő képernyő elkészítése

Készítsük el a következő képernyőt Auto Layout kényszerek segítségével és valósítsuk meg a gombok működését!

Segítség a képernyő felépítése:

* Alapja két db. azonos magasságú konténer view, amely szélességében kitölti a rendelkezésre álló teret és a köztük lévő távolság 0.
* A felső konténer View háttere vörös az alsójé szürke.
* A felső konténer tartalmaz egy képet, amely kitölti a rendelkezésre álló területet, ugyanakkor nem torzítja a képet.
* Az alsó konténer két középre rendezett gombot tartalmaz, amely vertikális pozíciója is rögzített.
* Az alsó konténer tartalmaz továbbá egy alulra pozícionált © Minden jog fenntartva szöveget.