

BASIC UIKIT

Jiri Ostatnicky, iOS Dev at STRV

STRV

LOGIN SCREEN

Sketch 9:41 AM 100%

STRV ACADEMY

✉ Email Address

🔒 Password

Done

Sketch 9:41 AM 100%

STRV ACADEMY

✉ |

🔒 Password

Done

q w e r t y u i o p
a s d f g h j k l
⌵ z x c v b n m ⌵
123 😊 🗣 space return

Sketch 9:41 AM 100%

STRV ACADEMY

✉ adam.roller@strv.com

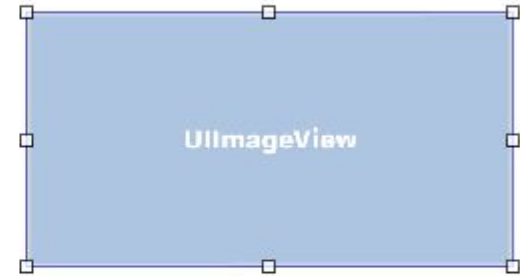
🔒 SHOW

Done

q w e r t y u i o p
a s d f g h j k l
⌵ z x c v b n m ⌵
123 😊 🗣 space return

BASIC UI COMPONENTS

- UIView
- UILabel
- UIButton
- UITextField
- UIImageView
- UIStackView



UIView

- An object that manages the content for a rectangular area on the screen
- <https://developer.apple.com/documentation/uikit/uiview>



```
// Rectangular red view
let redViewRect = CGRect(x: 100, y: 200, width: 200, height: 100)
let redView = UIView(frame: redViewRect)
redView.backgroundColor = .red
view.addSubview(redView)
```

UILabel

- A view that displays one or more lines of read-only text
- <https://developer.apple.com/documentation/uikit/UILabel>



```
// Label
let label = UILabel(frame: CGRect(x: 0, y: 40, width: 200, height: 40))
label.text = "Hello, World!"
label.textColor = .gray
view.addSubview(label)
```

UIButton

- A control that executes your custom code in response to user interactions
- <https://developer.apple.com/documentation/uikit/uibutton>



```
// Button
```

```
let button = UIButton(type: .system)
button.frame = CGRect(x: 0, y: 100, width: 200, height: 40)
button.setTitle("Button", for: .normal)
button.setTitleColor(.red, for: .normal)
button.backgroundColor = .yellow
button.addTarget(self, action: #selector(didPressButton),
                for: .touchUpInside)
view.addSubview(button)
```

UITextField

- An object that displays an editable text area in your interface
- <https://developer.apple.com/documentation/uikit/uitextfield>



```
// Text field
```

```
let textFieldRect = CGRect(x: 0, y: 150, width: 200, height: 40)
let textField = UITextField(frame: textFieldRect)
textField.textColor = .gray
textField.placeholder = "Placeholder"
textField.tintColor = .red // Change the cursor color
view.addSubview(textField)
```

UIImageView

- Displays a single image or a sequence of animated images in your interface
- <https://developer.apple.com/documentation/uikit/uiimageView>

```
// Image view
```

```
let strvAcademyLogo = UIImage(named: "STRV Academy")  
let imageView = UIImageView(image: strvAcademyLogo)  
view.addSubview(imageView)
```



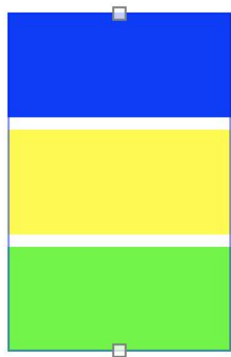
UIStackView

- A streamlined interface for laying out a collection of views in either a column or a row
- <https://developer.apple.com/documentation/uikit/uistackview>

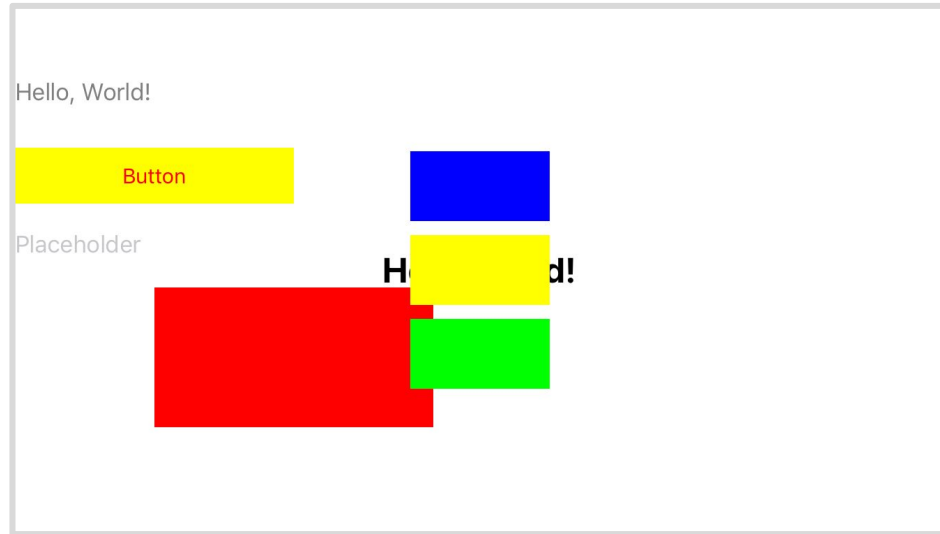
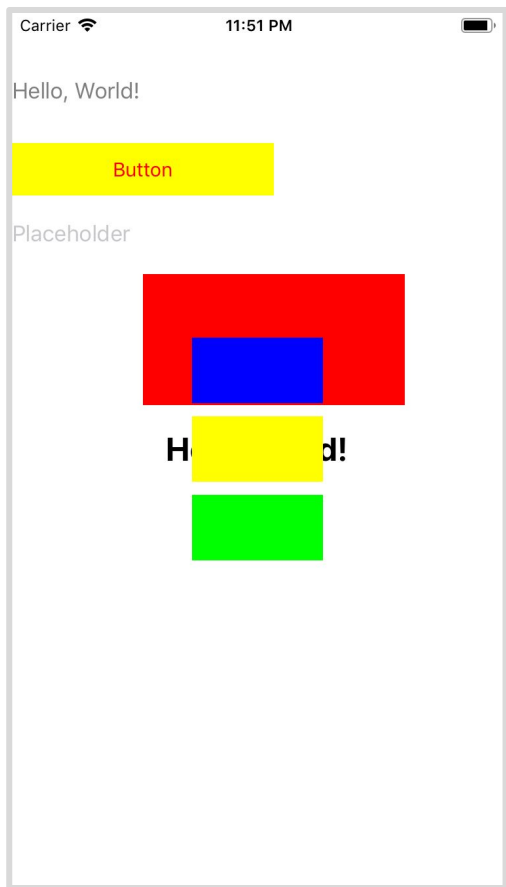
```
// Stack view
```

```
let stackView = UIStackView(arrangedSubviews: colorfulViews)
// colorfulViews are on the next slide
stackView.axis = .vertical // .horizontal
stackView.distribution = .equalSpacing // .fillEqually, .fill
stackView.alignment = .center
stackView.spacing = 10
stackView.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(stackView)
```

```
stackView.centerXAnchor.constraint(equalTo: view.centerXAnchor,
                                   constant: 0).isActive = true
stackView.centerYAnchor.constraint(equalTo: view.centerYAnchor,
                                   constant: 0).isActive = true
```

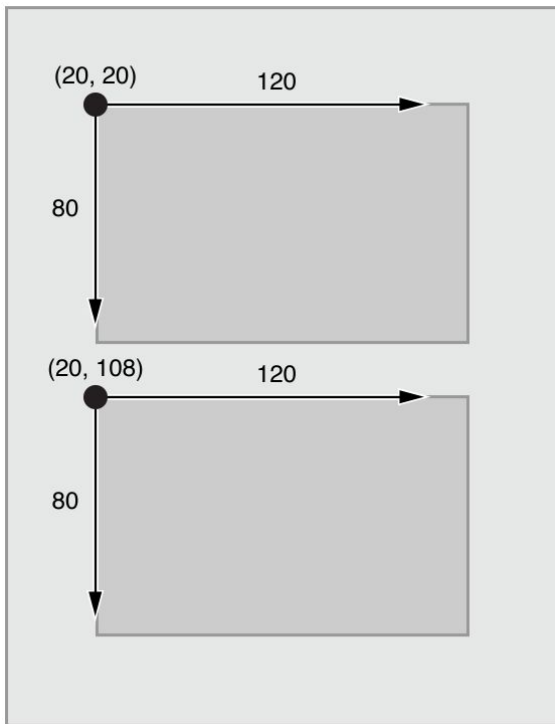


```
// Insert before stackView initialization
let colors: [UIColor] = [.blue, .yellow, .green]
let colorfulViews: [UIView] = colors.map { color in
    let smallView = UIView()
    smallView.backgroundColor = color
    smallView.widthAnchor.constraint(equalToConstant: 100).isActive = true
    smallView.heightAnchor.constraint(equalToConstant: 50).isActive = true
    return smallView
}
// ... stackView ...
```

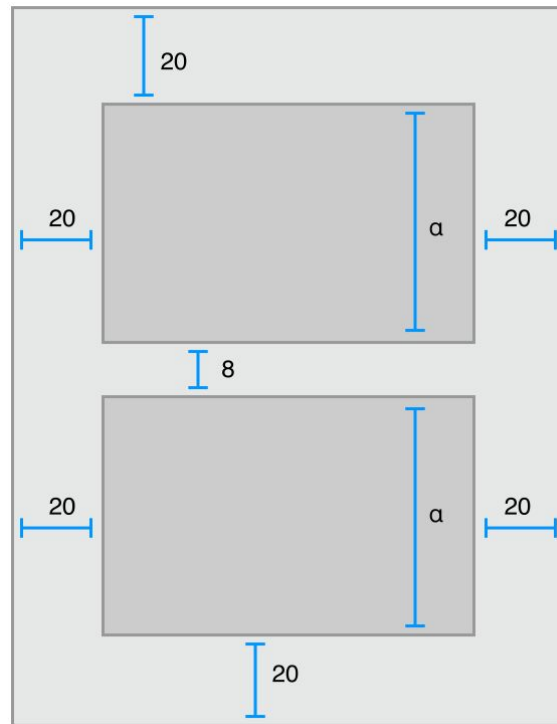


AUTO LAYOUT

FRAME-BASED LAYOUT VS. AUTO LAYOUT



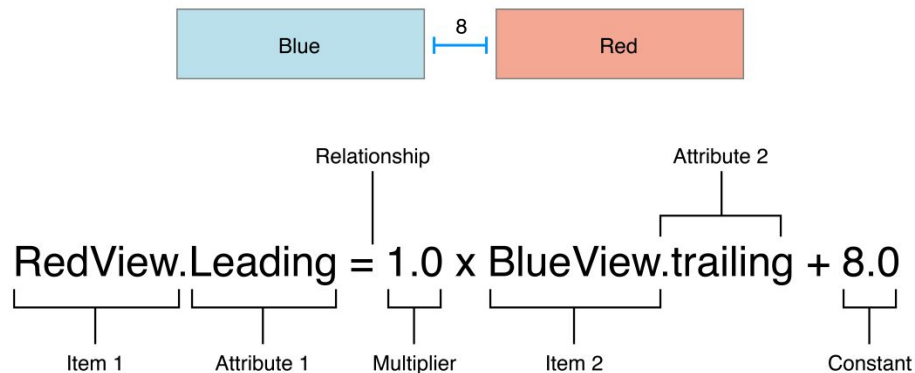
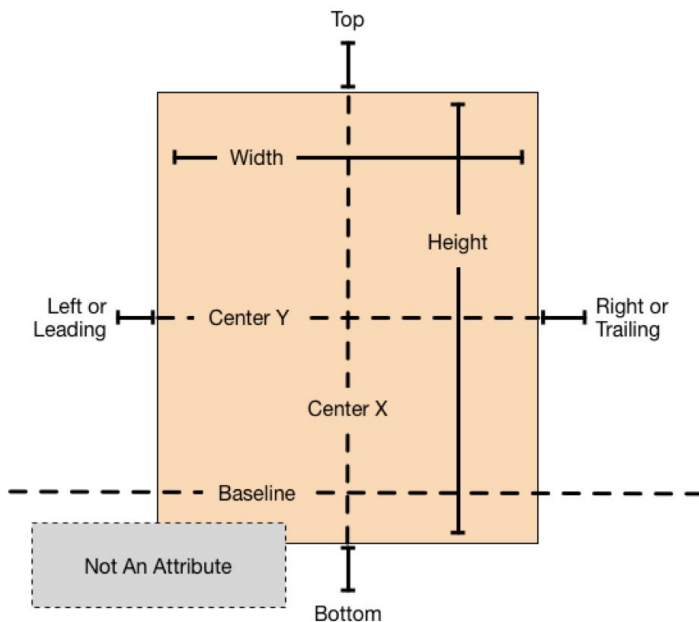
Frame-Based Layout



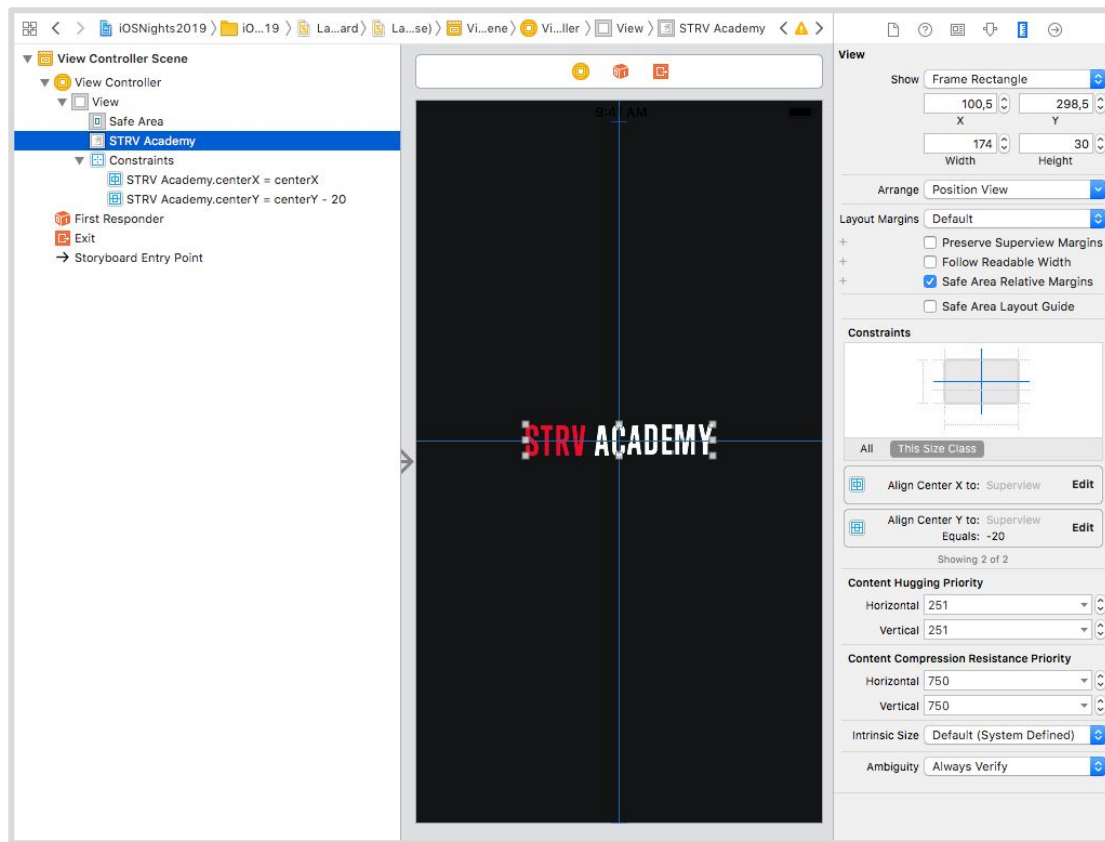
Auto Layout

CONSTRAINTS

- https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AnatomyofaConstraint.html#//apple_ref/doc/uid/TP40010853-CH9-SW1



EXAMPLE: LOGO IN LAUNCH SCREEN



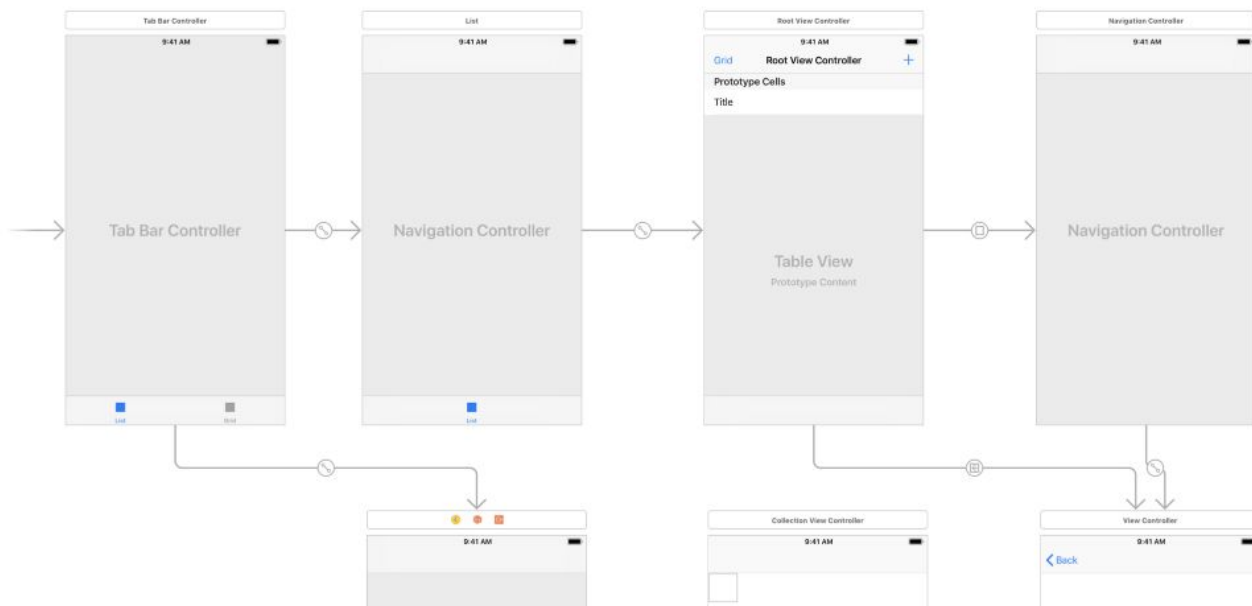
AUTO LAYOUT

- Auto Layout dynamically calculates the size and position of all the views in your view hierarchy, based on constraints placed on those views.
- **External changes** — when the size or shape of your superview changes
 - *The device rotates, support different screen sizes*
- **Internal changes** — when the size of the views or controls in your user interface change
 - *The content displayed by the app changes, Dynamic Fonts*

WHERE TO CREATE UI?

STORYBOARD & XIB (NIB)

- Options to define UI visually (not in a code)
- XML files
- Editing in Interface Builder



STORYBOARD

- Define ViewControllers (Tabs, Navigations)
- Can contain more of them
- Connections between of them
- File -> New -> File -> Storyboard (in User Interface section)

```
let vc = UIStoryboard(name: "LoginViewController",  
                      bundle: nil).instantiateInitialViewController()
```

XIB (NIB)

- Only for defining one (or more) view
- Usually for cells in table view
- **XIB** = Xml Interface Builder
- **NIB** = Nxt Interface Builder (NXT = NextStep = NS) — old one, replace by XIB
- File -> New -> File -> View (in User Interface section)

```
let myView = Bundle.main.loadNibNamed("MyView", owner: nil,  
                                     options: nil)?[0] as? UIView
```

LET'S BUILD LOGIN SCREEN!

LOGIN SCREEN

STRV ACADEMY

✉ Email Address

🔒 Password

Done

STRV ACADEMY

✉ |

🔒 Password

Done

q w e r t y u i o p
a s d f g h j k l
⌵ z x c v b n m ⌵
123 😊 🗣 space return

STRV ACADEMY

✉ adam.roller@strv.com

🔒 SHOW

Done

q w e r t y u i o p
a s d f g h j k l
⌵ z x c v b n m ⌵
123 😊 🗣 space return

STATUS BAR

- Status bar style only for one screen. Add into LoginViewController class:

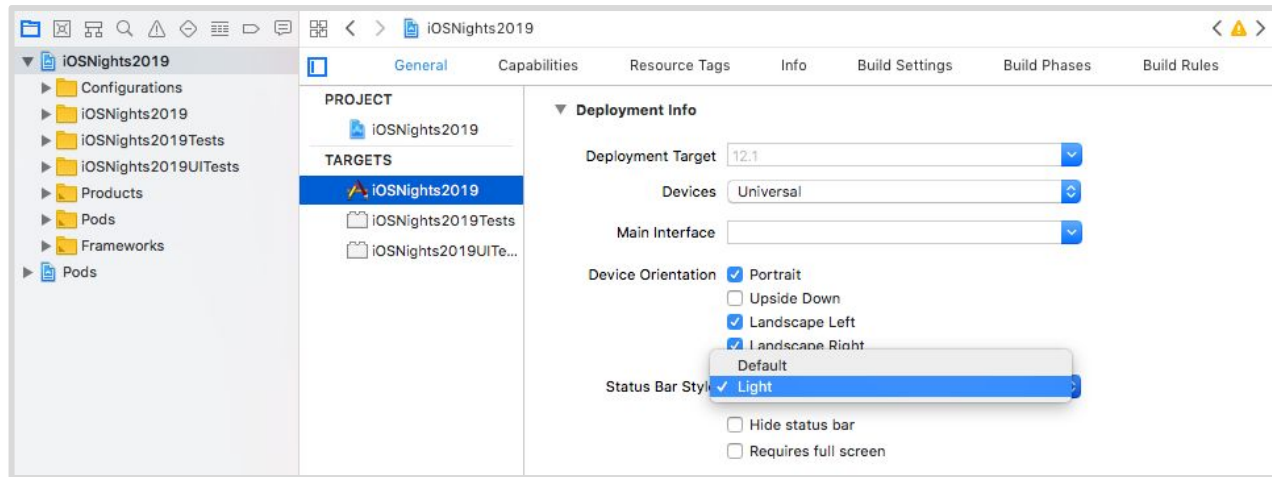
```
override var preferredStatusBarStyle: UIStatusBarStyle {  
    return .lightContent  
}
```

- Status bar style for while app. Add into Info.plist (Open As -> Source Code):

```
<key>UIViewControllerBasedStatusBarAppearance</key>  
<false/>
```

STATUS BAR (IN LAUNCH TIME)

- iOSNights2019 -> iOSNights2019 in Targets -> General in top tabs -> Deployment Info



- Or add to Info.plist (Open As -> Source Code):
`<key>UIStatusBarStyle</key>`
`<string>UIStatusBarStyleLightContent</string>`

CUSTOM FONTS

https://developer.apple.com/documentation/uikit/text_display_and_fonts/adding_a_custom_font_to_your_app

- Add to Info.plist (Open As -> Source Code):

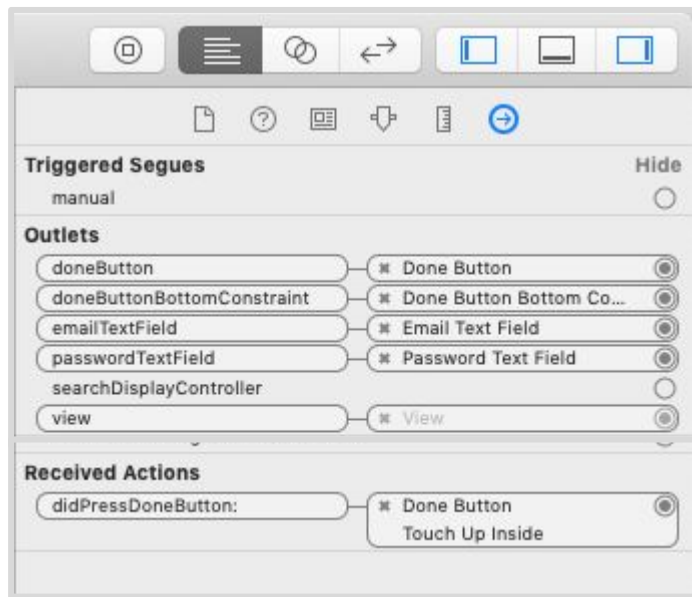
```
<key>UIAppFonts</key>
<array>
  <string>Maison Neue Book.otf</string>
  <string>Maison Neue Bold.otf</string>
  <string>Maison Neue Demi.otf</string>
  <string>Maison Neue Medium.otf</string>
  <string>trump_gothic_east_bold.ttf</string>
</array>
```

IBOUTLET & IBACTION

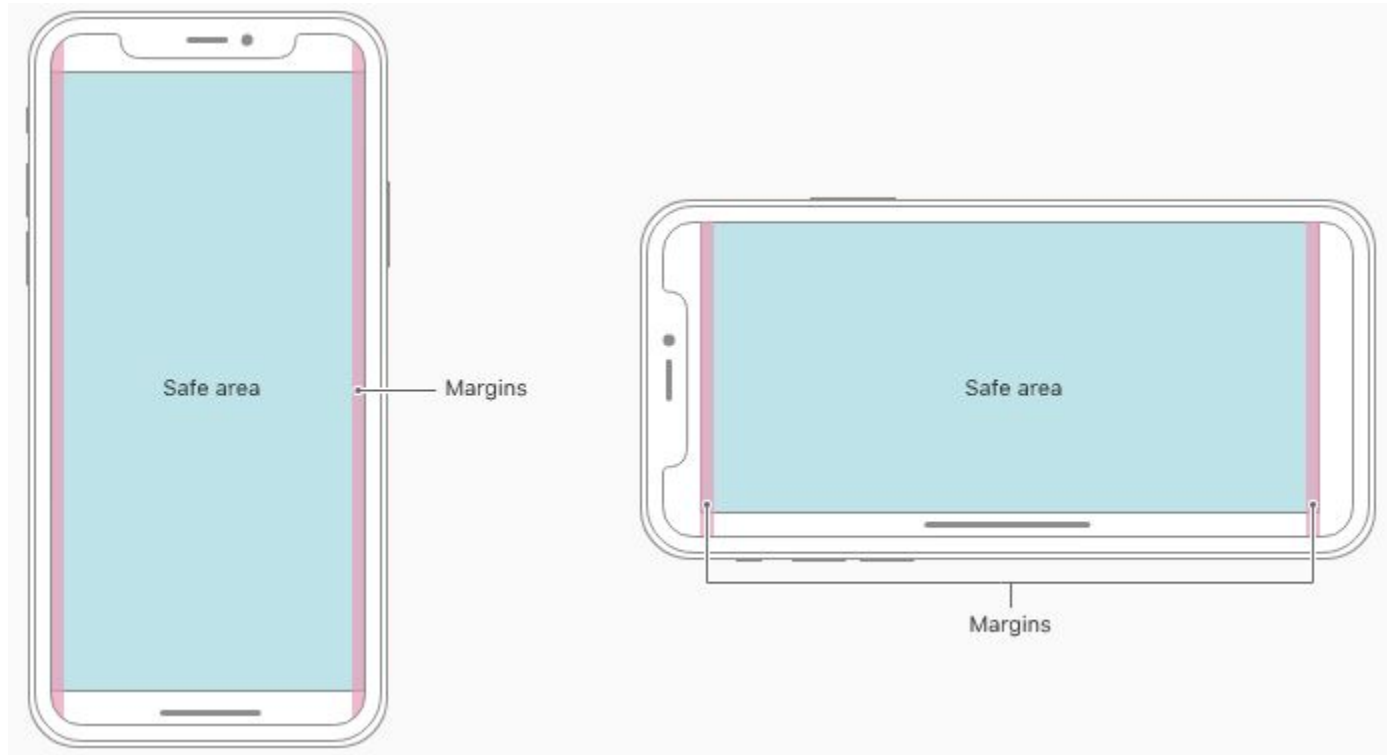
- **IBAction** and **IBOutlet** are macros defined to denote variables and methods that can be referred to in Interface Builder (<https://stackoverflow.com/a/1643039/1054550>)

```
@IBOutlet var emailTextField: UITextField!

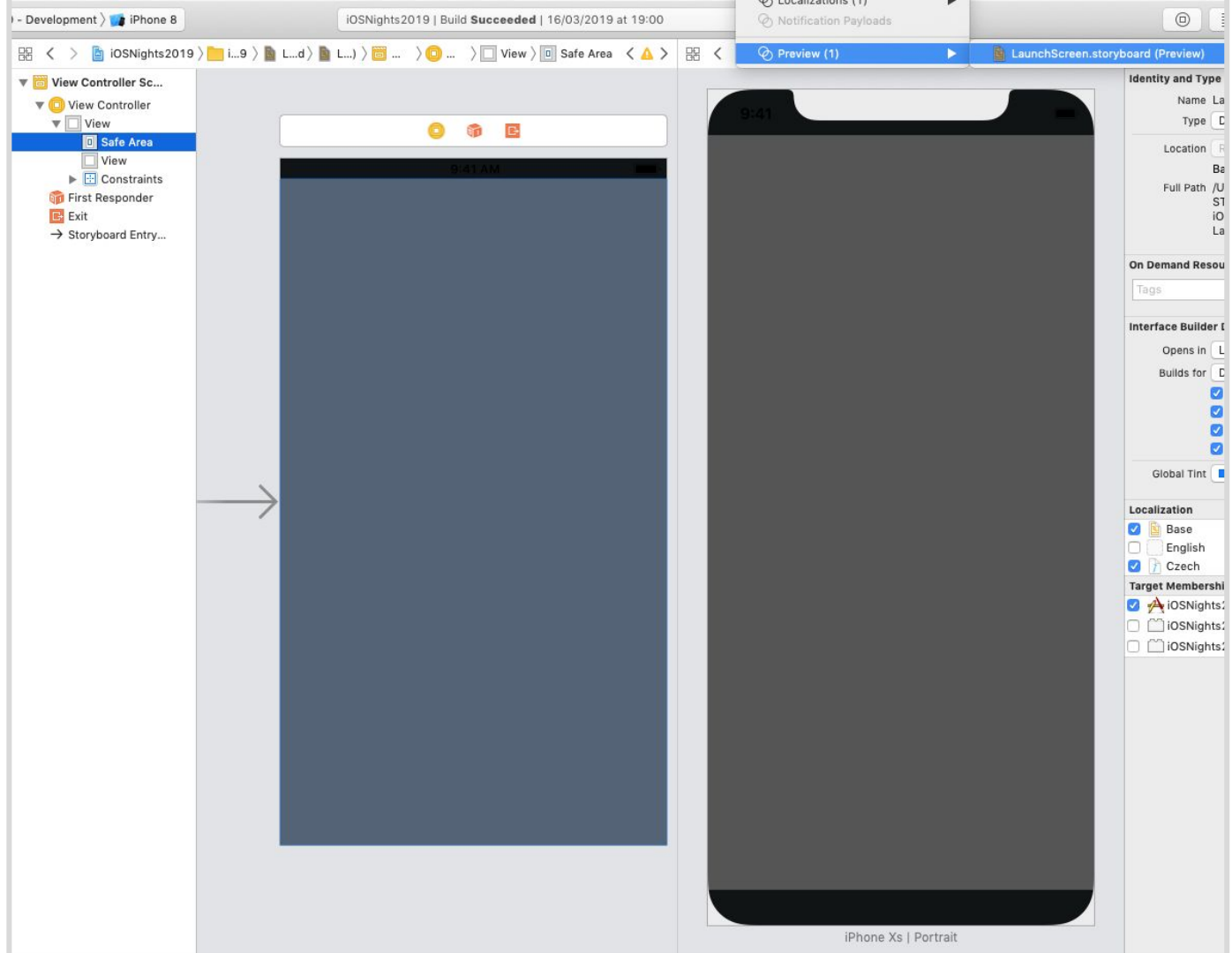
@IBAction func didPressDoneButton(_: Any) {
    print("Did press button")
}
```



SAFE AREA



SAFE AREA



COLORS SNIPPET

- UIColorExtension.swift —

<https://gist.github.com/ostatnicky/9562aa29fe560576812397405ede2247>

TAP GESTURE

- <https://developer.apple.com/documentation/uikit/uitapGestureRecognizer>

```
// Tap gesture
let tap = UITapGestureRecognizer(target: self, action:
#selector(didTapOnView))
view.addGestureRecognizer(tap)

@objc func didTapOnView() {
    view.endEditing(true)
}
```

UITextField INPUT OBSERVING

```
emailTextField.addTarget(self, action: #selector(didChangeInput),  
    for: .editingChanged)  
passwordTextField.addTarget(self, action: #selector(didChangeInput),  
    for: .editingChanged)
```

```
@objc func didChangeInput() {  
    print("Did change input")  
}
```

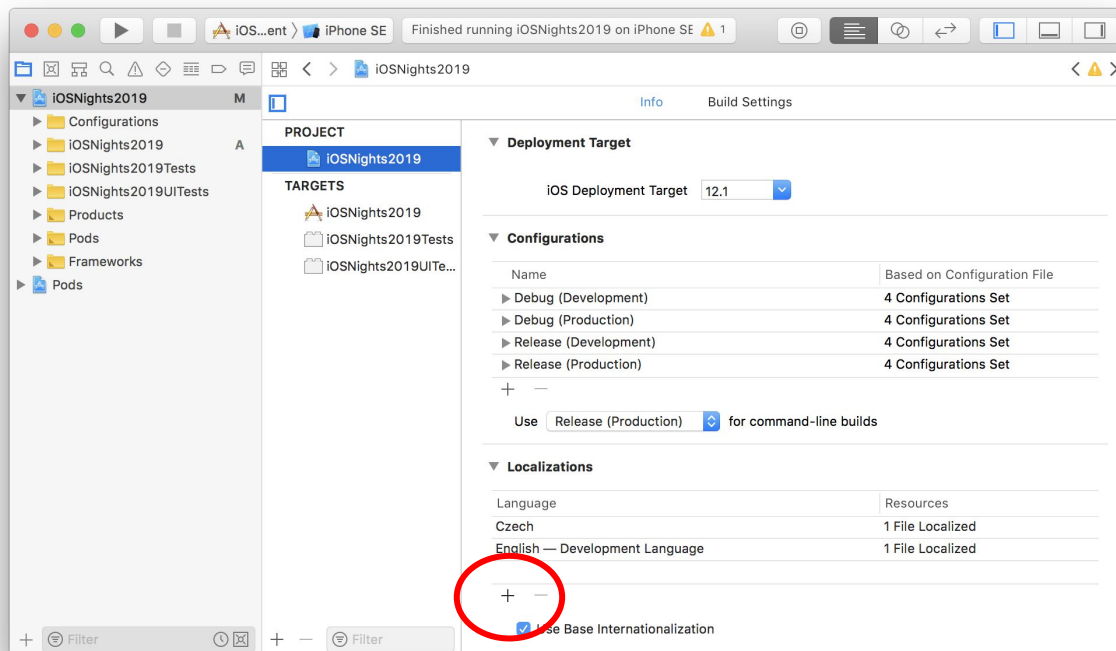
KEYBOARD NOTIFICATIONS

- <https://gist.github.com/ostatnicky/835137abc05778a3be3206362630e78b>
- ... and add `registerKeyboardListeners()` to bottom of `setupUI`

LOCALIZATION

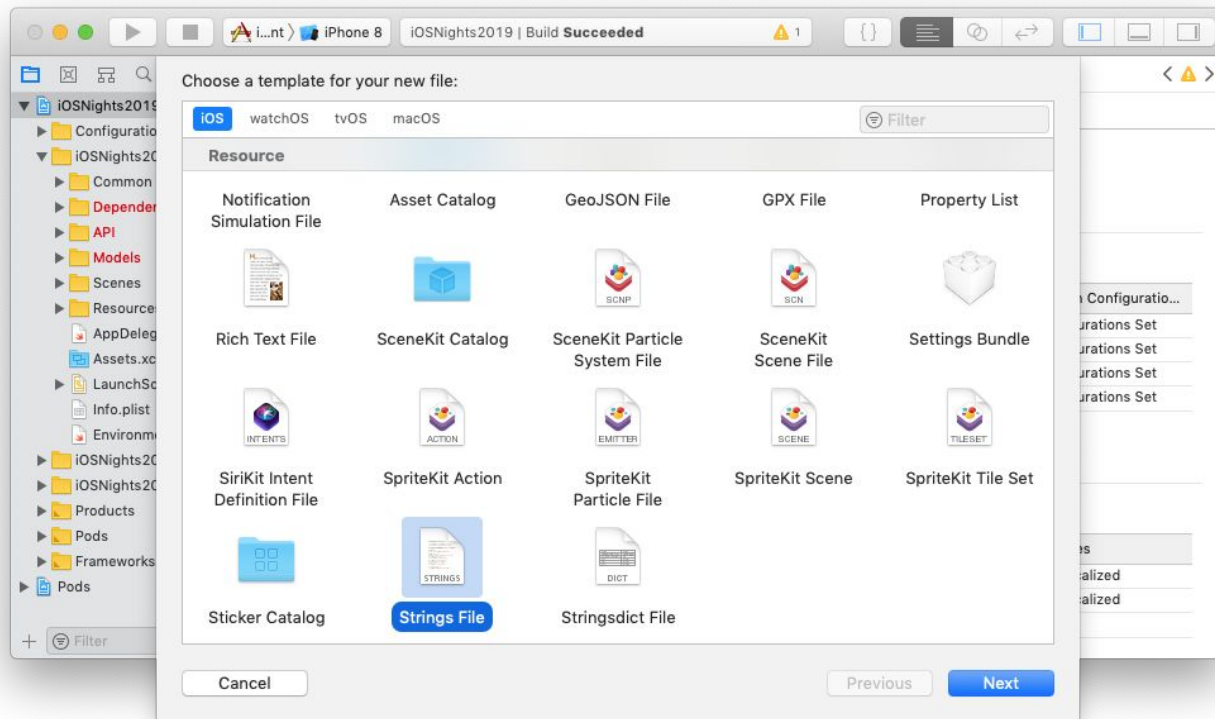
LOCALIZATION

- Add new language in the project localizations



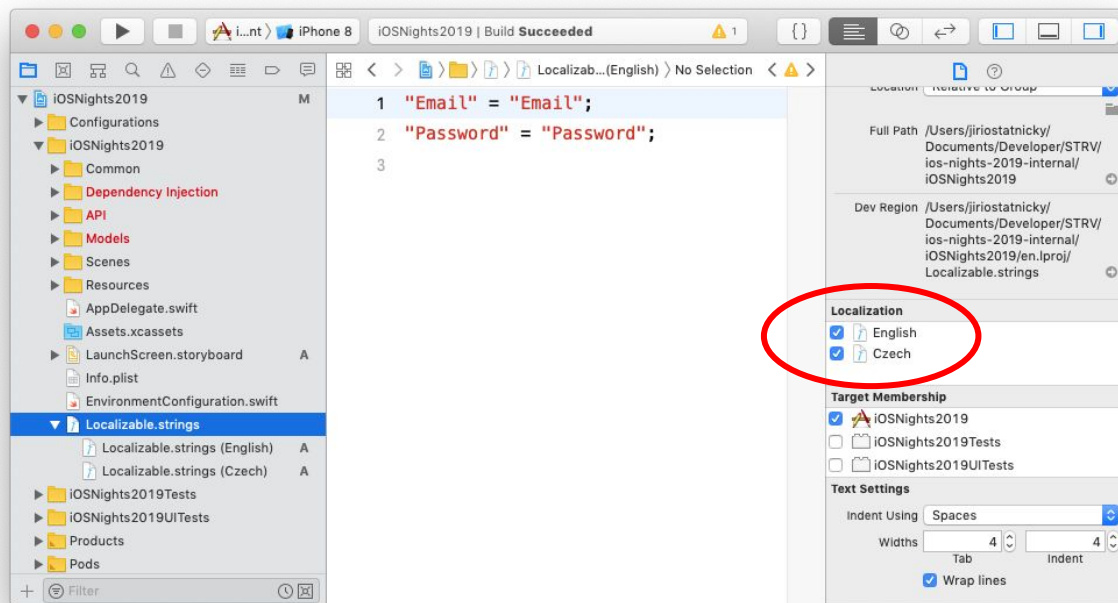
LOCALIZATION

- Create **Localizable.strings** by **File -> New -> File -> Strings File** under Resource tab of iOS



LOCALIZATION

- Localize that **Localizable.strings**
- Add “key” = “value”; pairs



LOCALIZATION

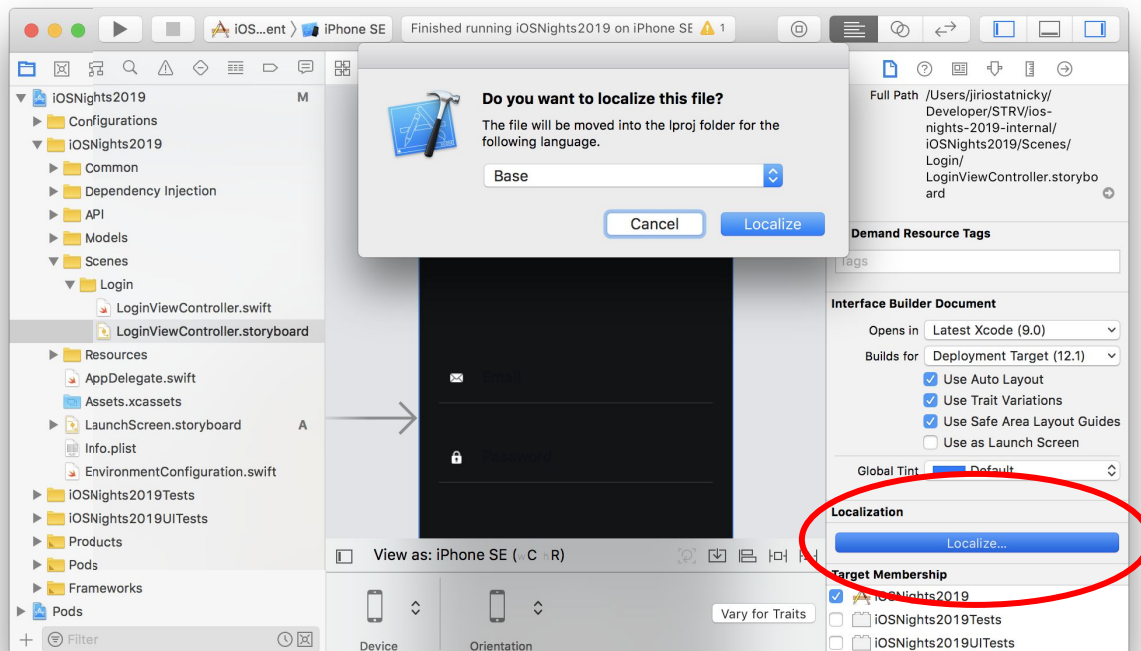
- Use **keys** in code by `NSLocalizedString(key: String, comment: String)`

```
let emailPlaceholderText = NSLocalizedString("Email", comment: "")  
let passwordPlaceholderText = NSLocalizedString("Password", comment: "")
```

LOCALIZATION IN UI

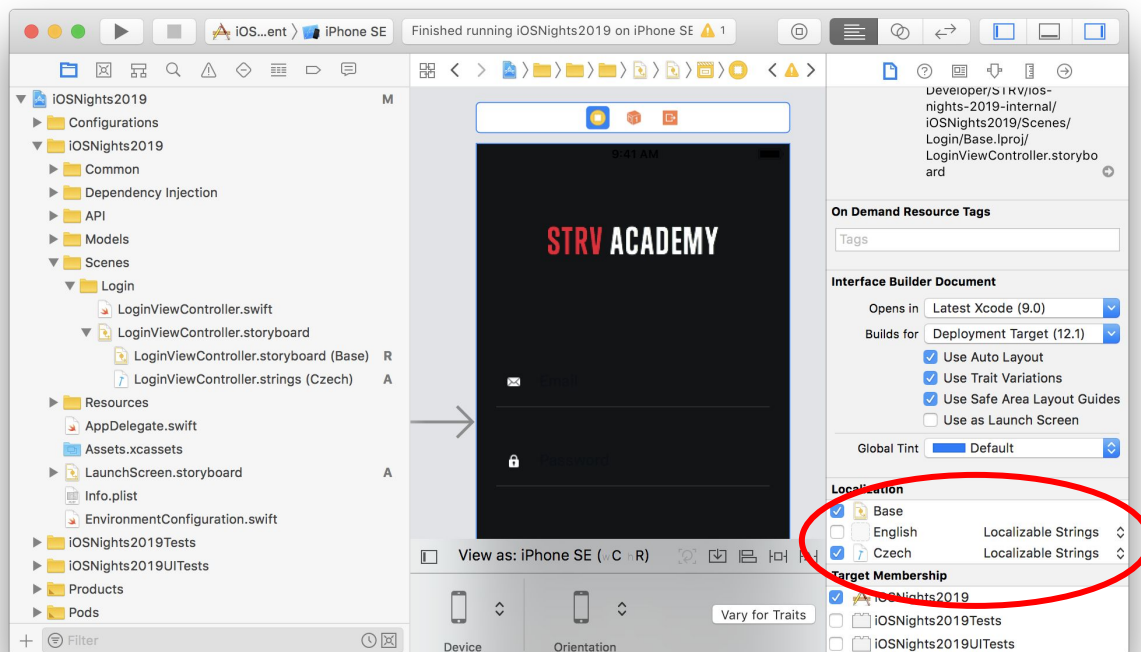
LOCALIZATION IN UI

- Localize LoginViewController.storyboard



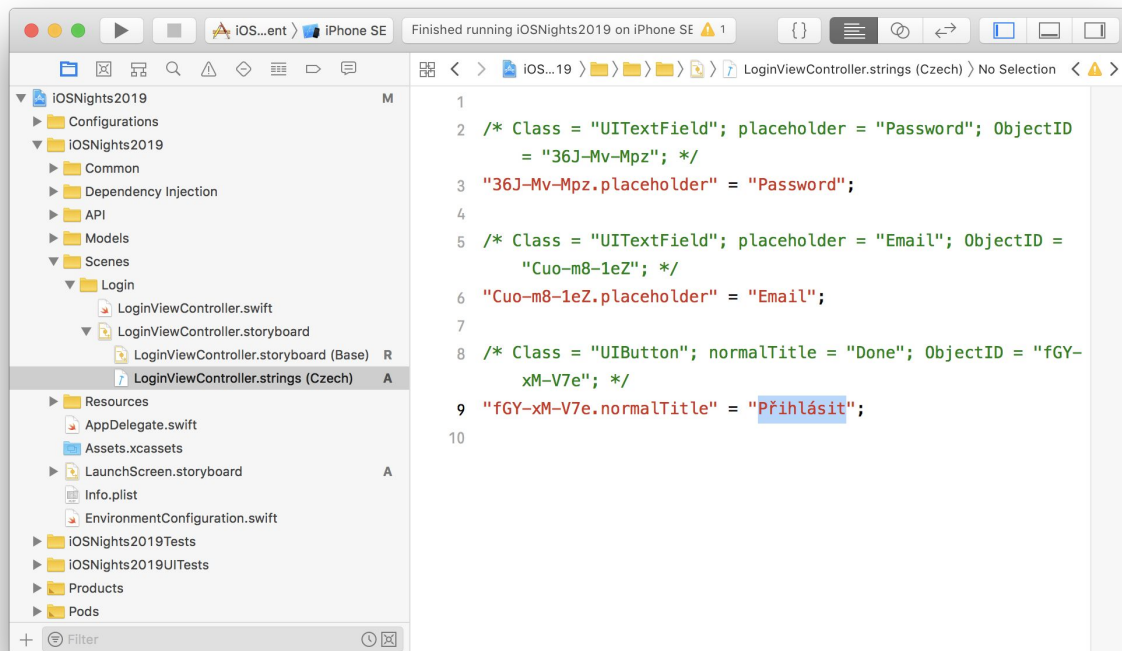
LOCALIZATION IN UI

- Check required language



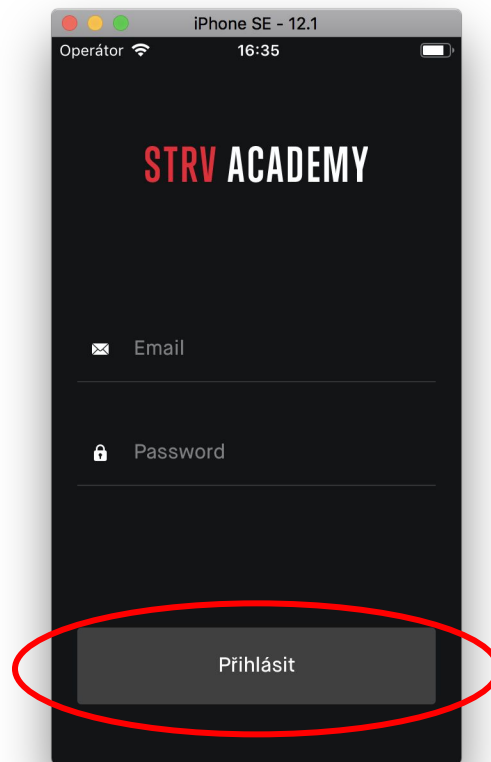
LOCALIZATION IN UI

- Translate to new language



LOCALIZATION

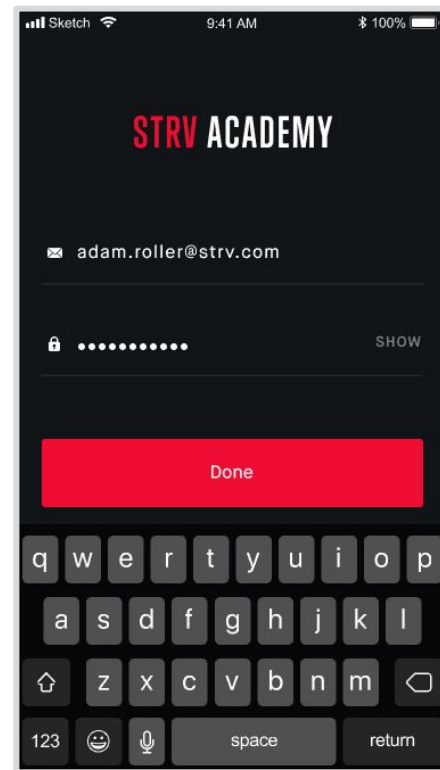
- Test it — be sure that you have the same language in iOS Settings
- OR BETTER: Product -> Scheme -> Edit Scheme... -> Run -> Options -> Application Language



HOMework

HOMEWORK

- **Add Show/Hide button for password textfield**
 - Hint: `passwordTextField.isSecureTextEntry`
- **Validate the email input format**
 - Done button is enabled only for a valid email



THAT'S IT

Jiri Ostatnický
jiri.ostatnický@strv.com

STRV

REFERENCE

- UIKit <https://developer.apple.com/documentation/uikit>
- Auto Layout
https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html#//apple_ref/doc/uid/TP40010853-CH7-SW1
- Localization
<https://medium.com/lean-localization/ios-localization-tutorial-938231f9f881>

QUESTIONS

STRV

STRV