



Programowanie mobilne

Programowanie w systemie iOS:
widoki, przejścia, zasoby aplikacji, gesty
i pobieranie danych z sieci

dr inż. Andrzej Wilczyński - www.awilczynski.me

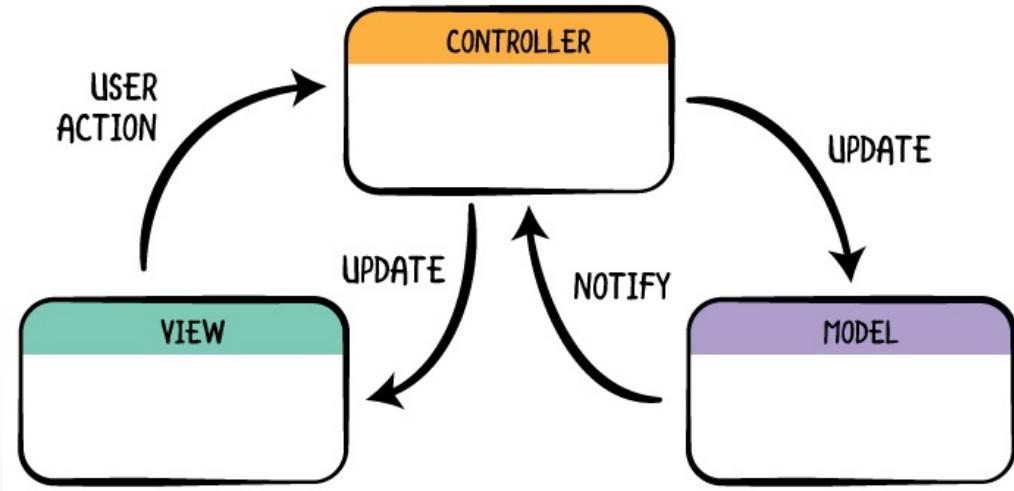
UIKit

Jest to podstawowy zestaw narzędzi do tworzenia interfejsu użytkownika. Pozwala na stworzenie listy widoków pomiędzy którymi można przechodzić za pomocą przejść. Każde przejście ma swój typ, który determinuje sposób wyświetlania nowego widoku. Podstawowe elementy to:

- `UIView` jako składowa sceny w aplikacji
- `UIViewController` – kontrolery poszczególnych widoków głównych
- `Storyboard` – narzędzie do konfiguracji kontrolerów

Model-View-Controller (MVC)

UIKit jest oparty na wzorcu projektowy Model-View-Controller.



Model-View-Controller (MVC) in iOS – A Modern Approach, fot. www.raywenderlich.com



Storyboard – segues (przejścia)

- Nawigacja i ekranы powinny być określone na początku
- Każdy widok powinien być samowystarczalny
- Segues pozwalają określić jak widoki będą wyświetlane i jakie zależności będą pomiędzy nimi
- Przejścia mogą być wywoływane ręcznie lub uruchamiane automatycznie



Najczęściej używane rodzaje przejść

- Show – wraz z UINavigationController pozwala na uzyskanie efektu stosu ekranów, kolejne ekranы przesuwają się z prawej strony, zastępując poprzedni, dostępny jest przycisk wstecz
- PresentModally – pokazuje nowy widok modalnie, zasłaniając poprzedni, standardowo animowany z dołu
- Present as Popover – pokazuje wyskakujące okienko
- Custom – można samemu zaimplementować klasę przejścia

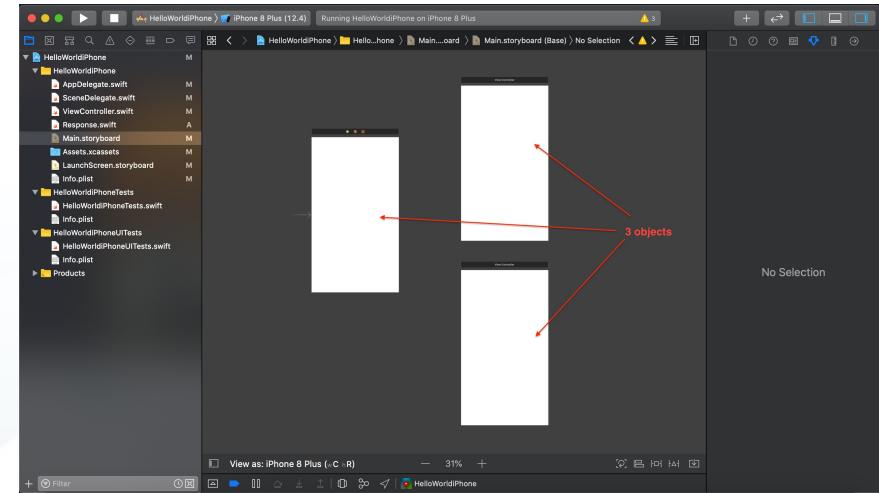
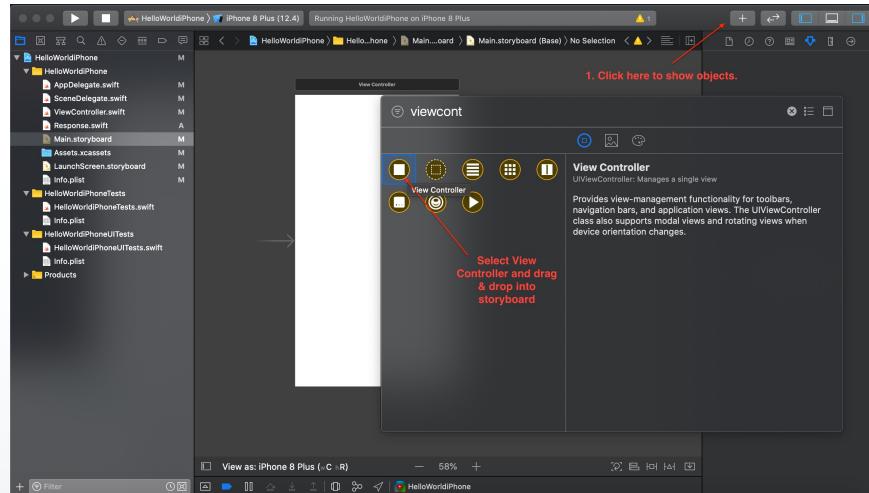


Projektowanie przestrzeni roboczej - kroki

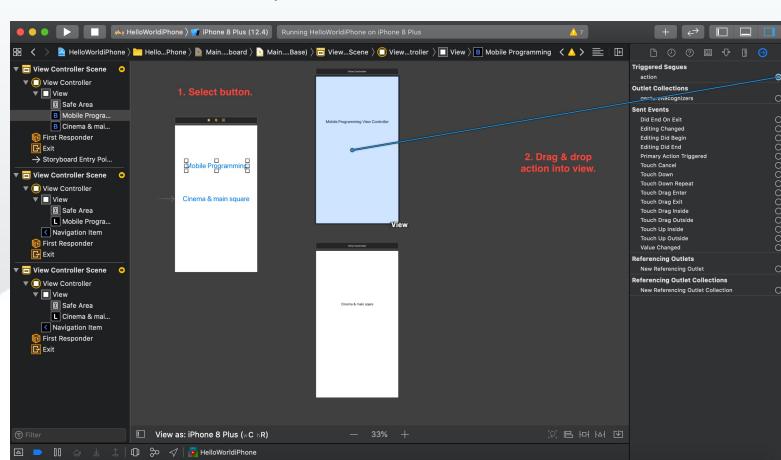
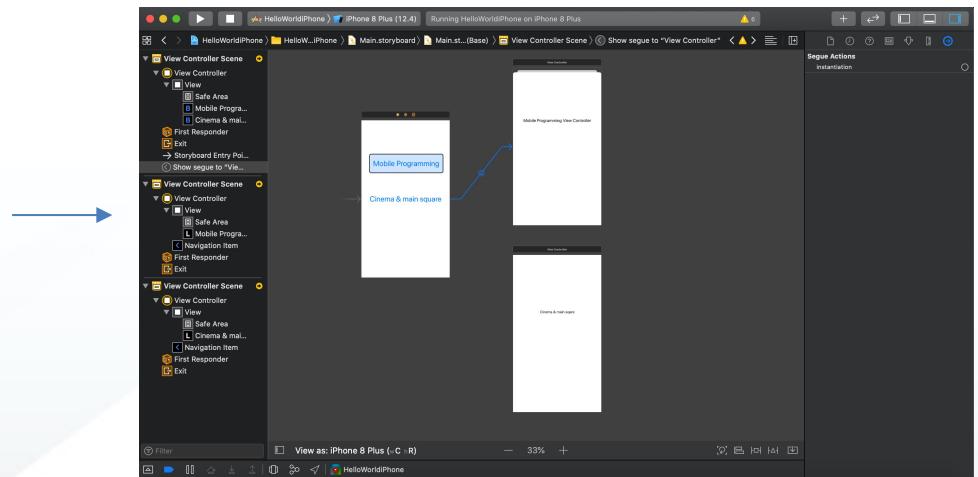
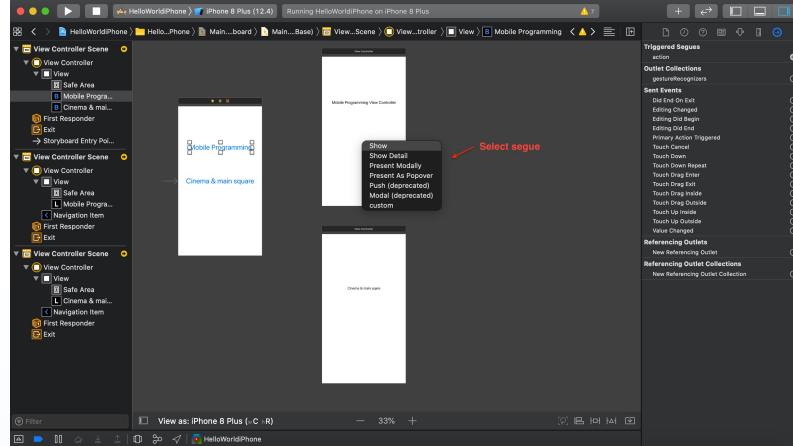
1. Tworzymy sceny i ViewController'y.
2. Tworzymy przejścia.
3. Programujemy wywołanie przejść.



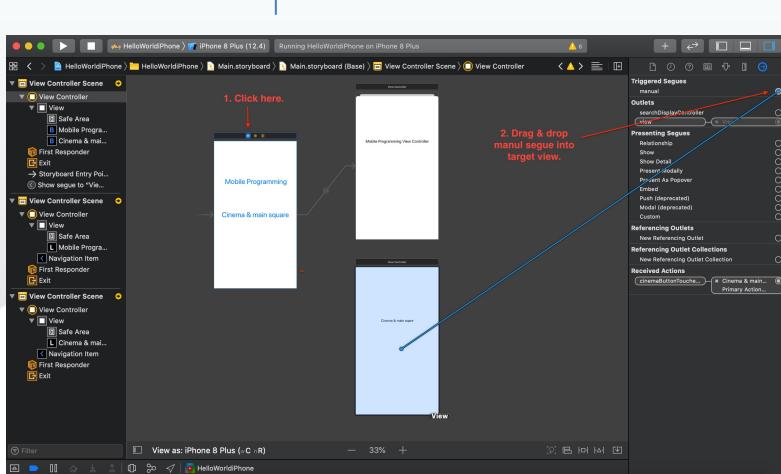
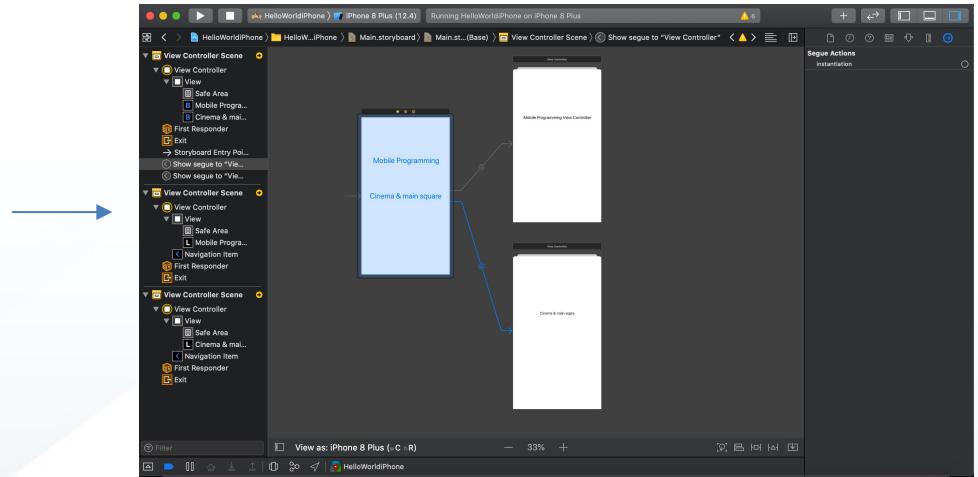
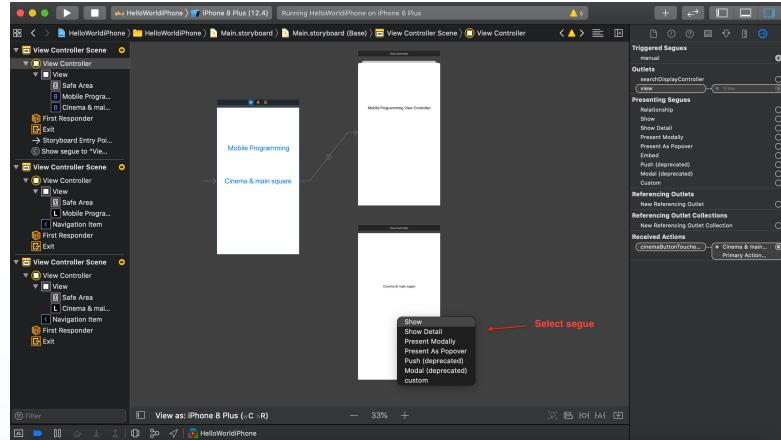
Dodawanie nowych widoków



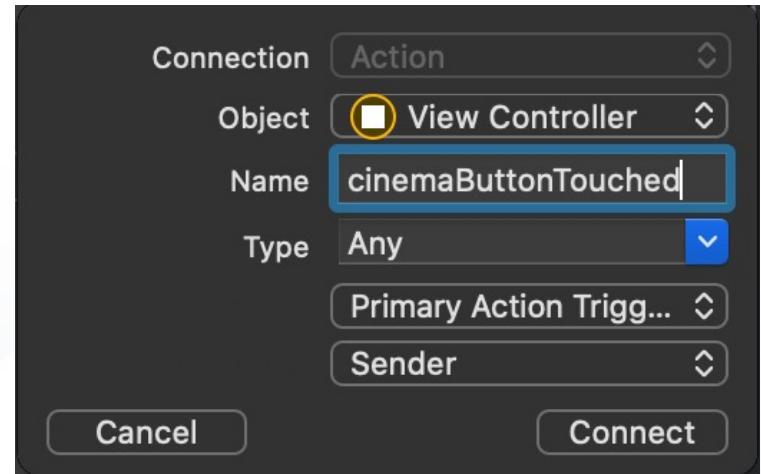
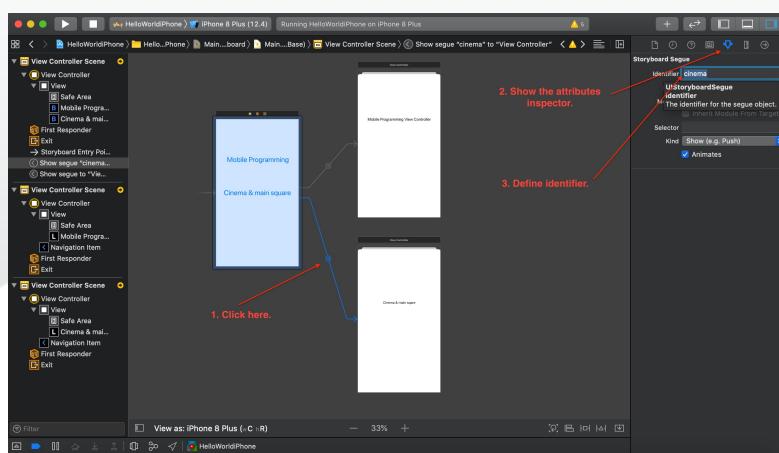
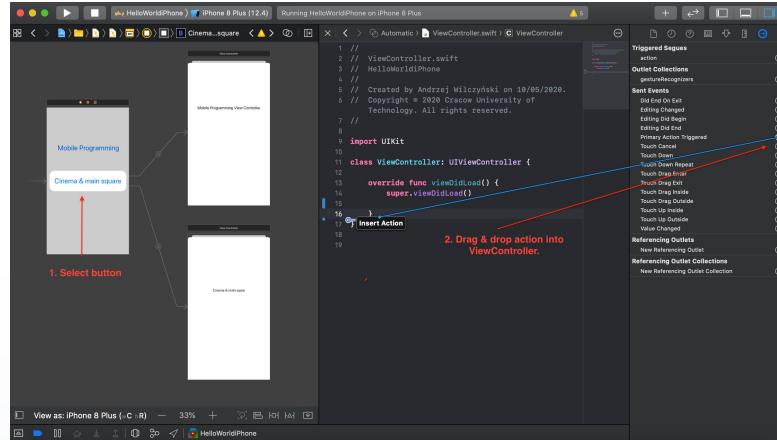
Programowanie przejść między obiektami za pomocą akcji



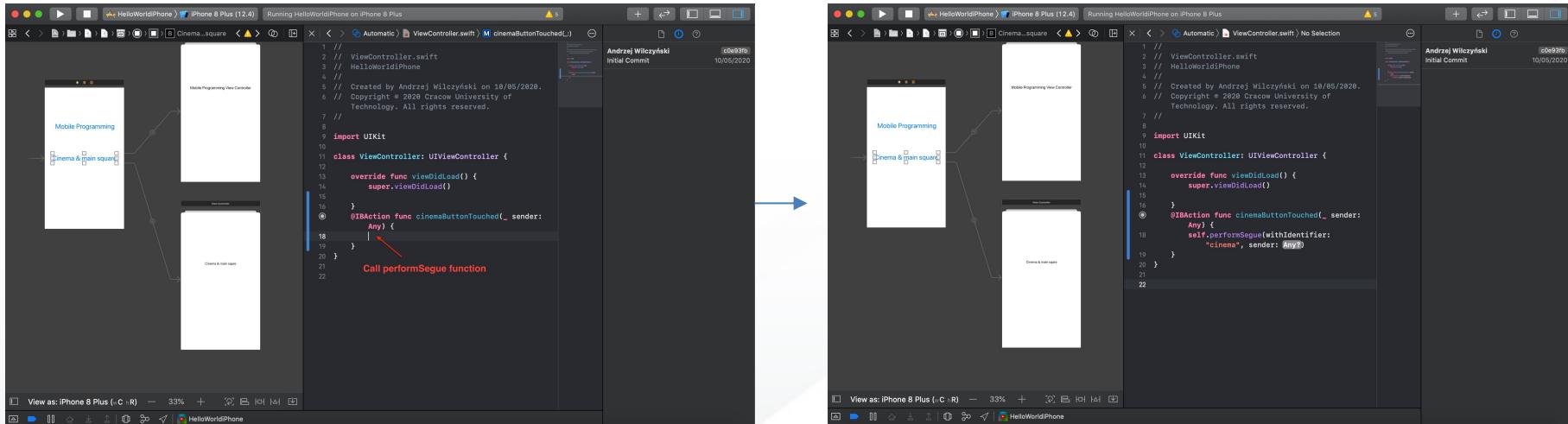
Programowanie przejść między obiektami za pomocą kodu (1)



Programowanie przejść między obiektami za pomocą kodu (2)



Programowanie przejść między obiektami za pomocą kodu (3)



The image shows two screenshots of the Xcode IDE side-by-side, illustrating the process of implementing a segue via code.

Left Screenshot: Shows the storyboard with a segue from a button to another view controller. The code in ViewController.swift is as follows:

```
1 // ViewController.swift
2 // HelloWorldiPhone
3 // Created by Andrzej Wilczyński on 10/05/2020.
4 // Copyright © 2020 Kraków University of Technology. All rights reserved.
5
6 import UIKit
7
8 class ViewController: UIViewController {
9
10     override func viewDidLoad() {
11         super.viewDidLoad()
12     }
13
14     @IBAction func cinemaButtonTouched(_ sender: Any) {
15         // Call performSegue function
16     }
17
18 }
```

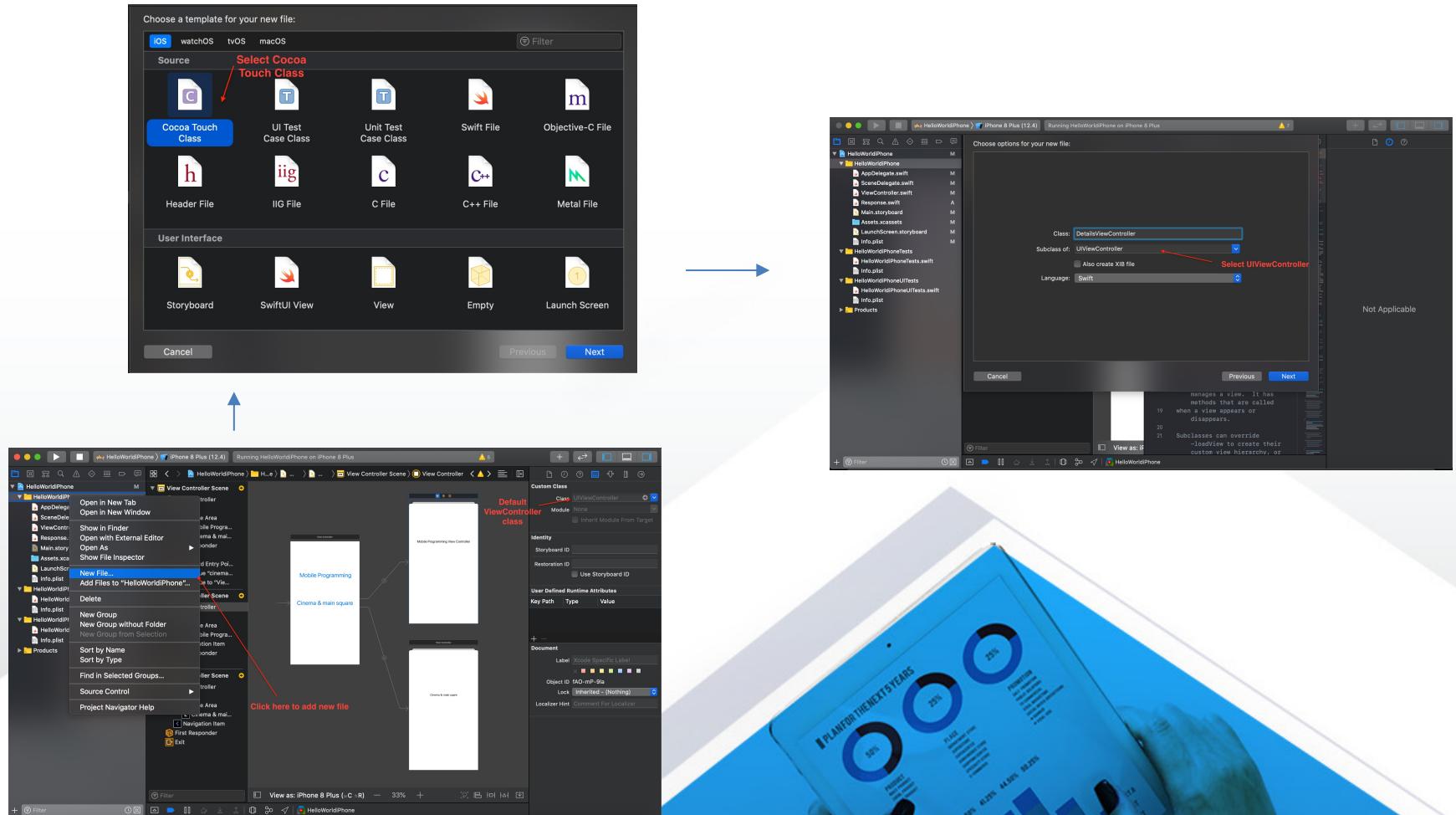
A red arrow points to the line `// Call performSegue function`, indicating where the segue logic should be implemented.

Right Screenshot: Shows the storyboard with the same segue setup. The code in ViewController.swift has been updated:

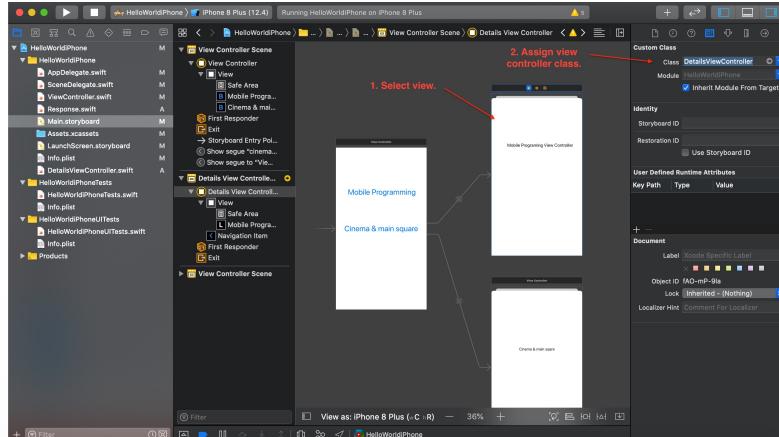
```
1 // ViewController.swift
2 // HelloWorldiPhone
3 // Created by Andrzej Wilczyński on 10/05/2020.
4 // Copyright © 2020 Kraków University of Technology. All rights reserved.
5
6 import UIKit
7
8 class ViewController: UIViewController {
9
10     override func viewDidLoad() {
11         super.viewDidLoad()
12     }
13
14     @IBAction func cinemaButtonTouched(_ sender: Any) {
15         self.performSegue(withIdentifier: "cinema", sender: Any())
16     }
17
18 }
```

The red arrow now points to the line `self.performSegue(withIdentifier: "cinema", sender: Any())`, which implements the segue using the identifier "cinema".

Dodawanie nowego kontrolera (1)



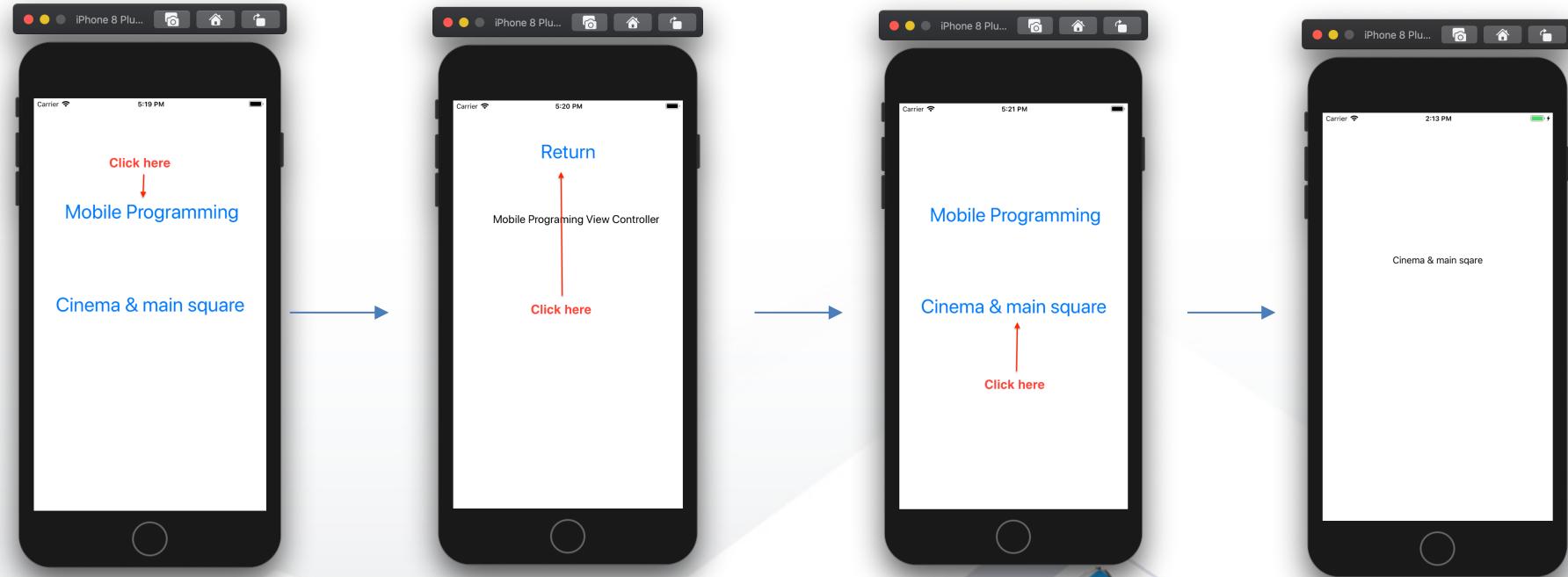
Dodawanie nowego kontrolera (2)



```
1 // MARK: - Navigation
2 // In a storyboard-based application, you will often want to do a little
3 // preparation before navigation
4 override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
5     // Get the new view controller using segue.destination.
6     // Pass the selected object to the new view controller.
7 }
8
9 import UIKit
10
11 class DetailsViewController: UIViewController {
12     override func viewDidLoad() {
13         super.viewDidLoad()
14         // Do any additional setup after loading the view.
15     }
16 }
```

```
13
14     override func viewDidLoad() {
15         super.viewDidLoad()
16
17         // Do any additional setup after loading
18         // the view.
19     }
20
21     @IBAction func returnTouched(_ sender: Any) {
22         self.dismiss(animated: true, completion: nil)
23     }
24
25     /* MARK: - Navigation
26
27     // In a storyboard-based application, you will
28     // often want to do a little preparation
29     // before navigation
30     override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
31         // Get the new view controller using segue.destination.
32         // Pass the selected object to the new
33         // view controller.
34     }
35 }
```

Działający przykład



Zasoby aplikacji

Zasoby używane w aplikacji mogą być dodawane w postaci:

- pojedynczych plików – ważne jest nazewnictwo w przypadku grafik @2x, @3x np. background@3x.png
- zbioru plików (xcassets) – rekommendowana forma dodawania

Zależności zasobów

- Grafika – wielkość ekranu i mnożnik
- Pamięć – 1GB/2GB
- Grafika – Metal1/Metal2
- Urządzenie – iPhone/iPad/Mac/Watch

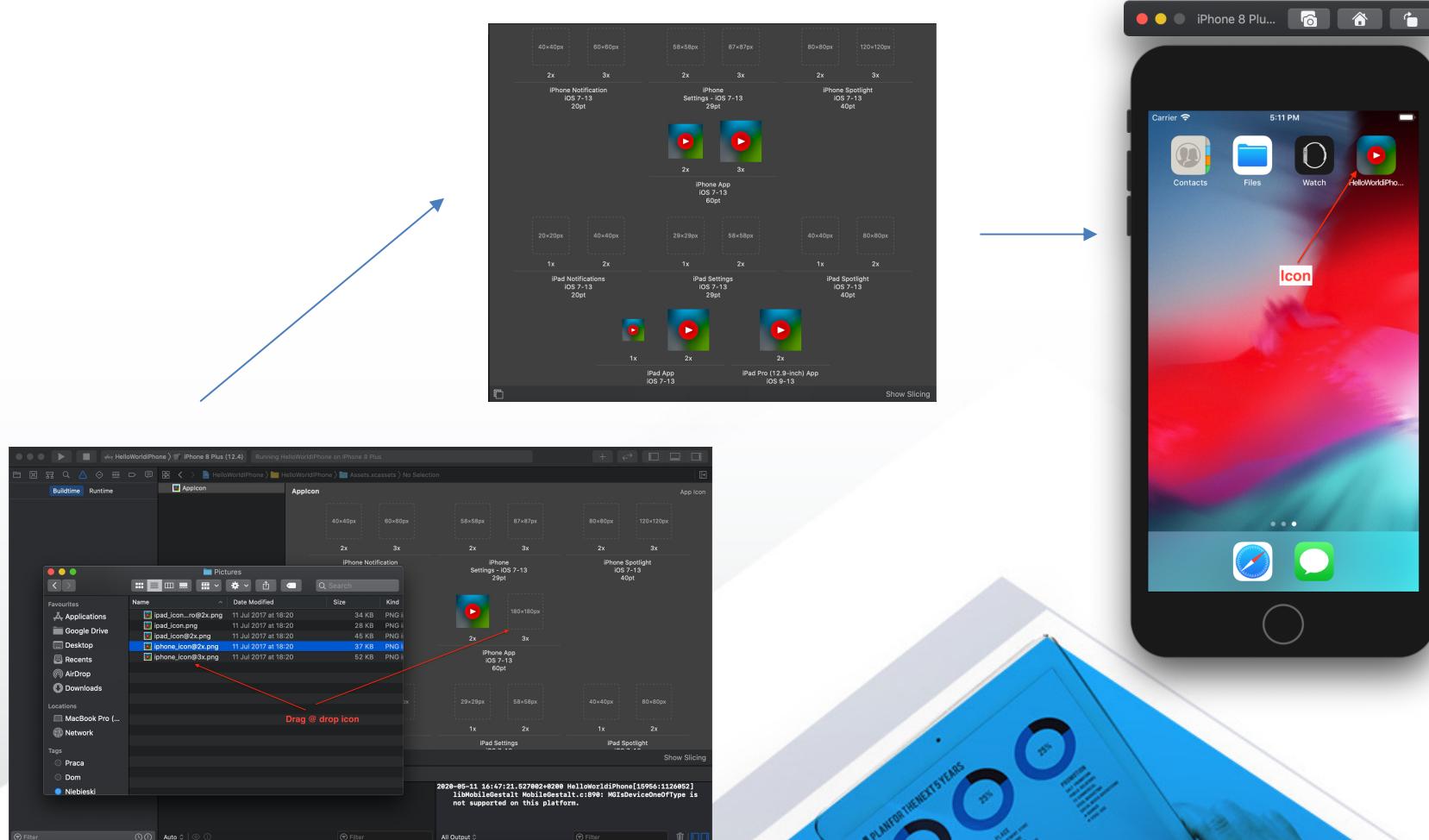
“Slicing” – jako technika dobierania zasobów przeznaczonych dla konkretnej platformy sprzętowej.

App Thining

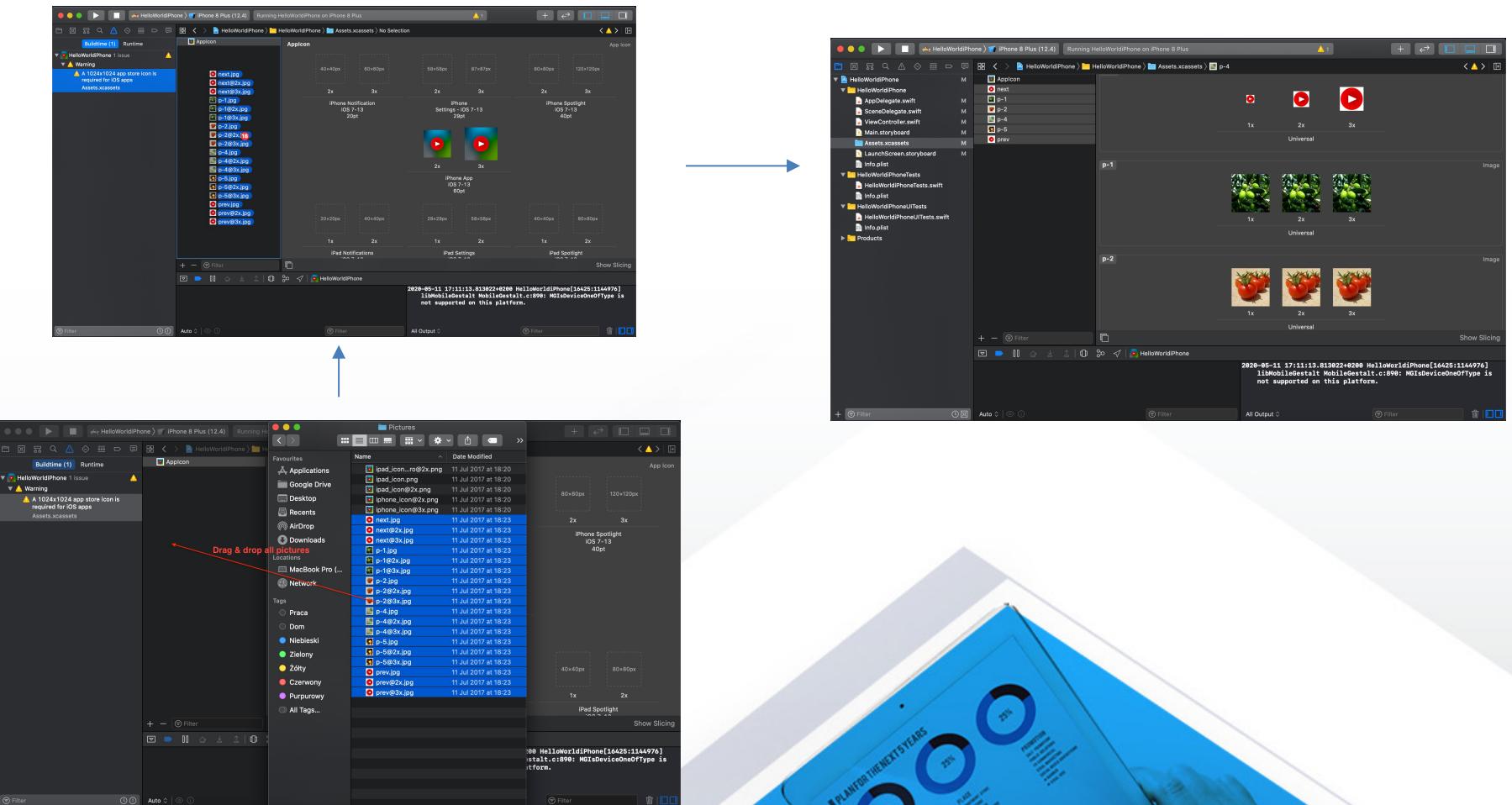
Aplikacja może działać na różnych urządzeniach, kod jest rekompilowany po stronie serwera dzięki czemu na przykład do Iphone 8 razem z paczką zostaną przesłane tylko te zasoby, które są dla niego przeznaczone. Zastosowanie tego narzędzia pozwala na zmniejszenie wielkości aplikacji o około 30-40%.



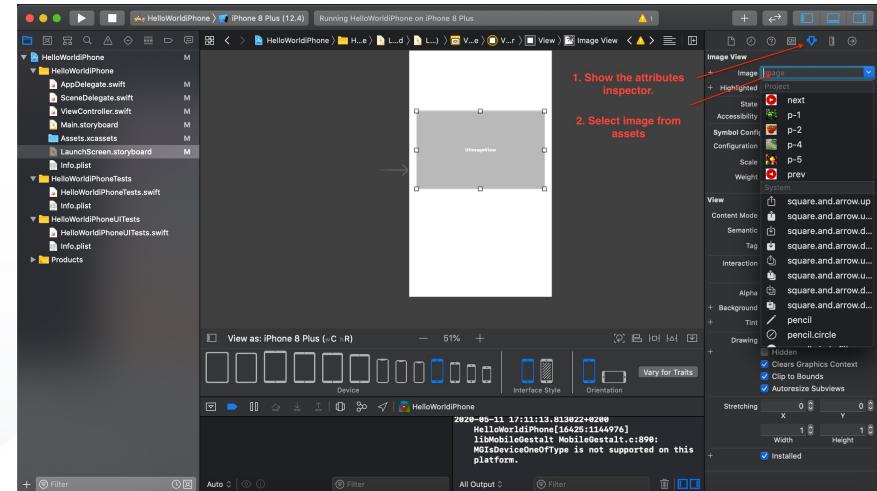
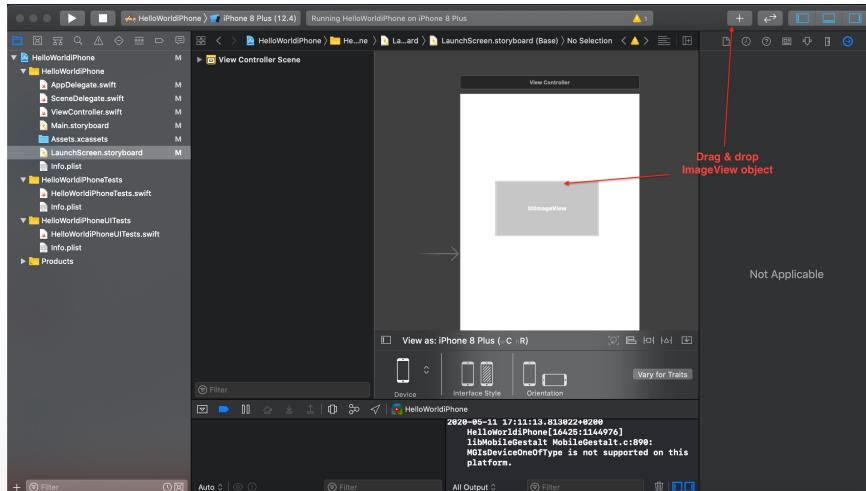
Dodawanie ikon aplikacji



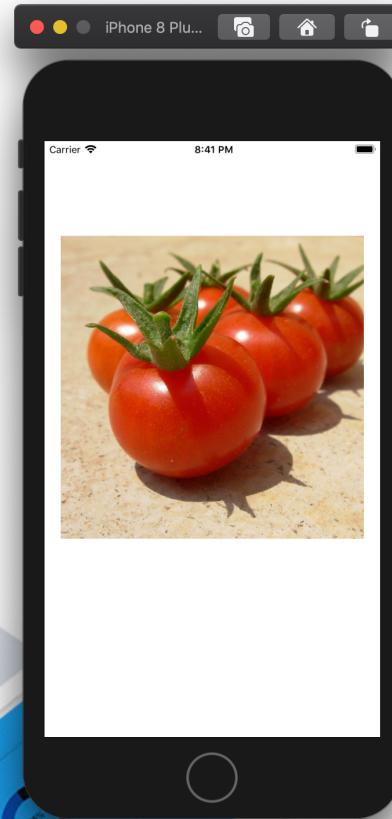
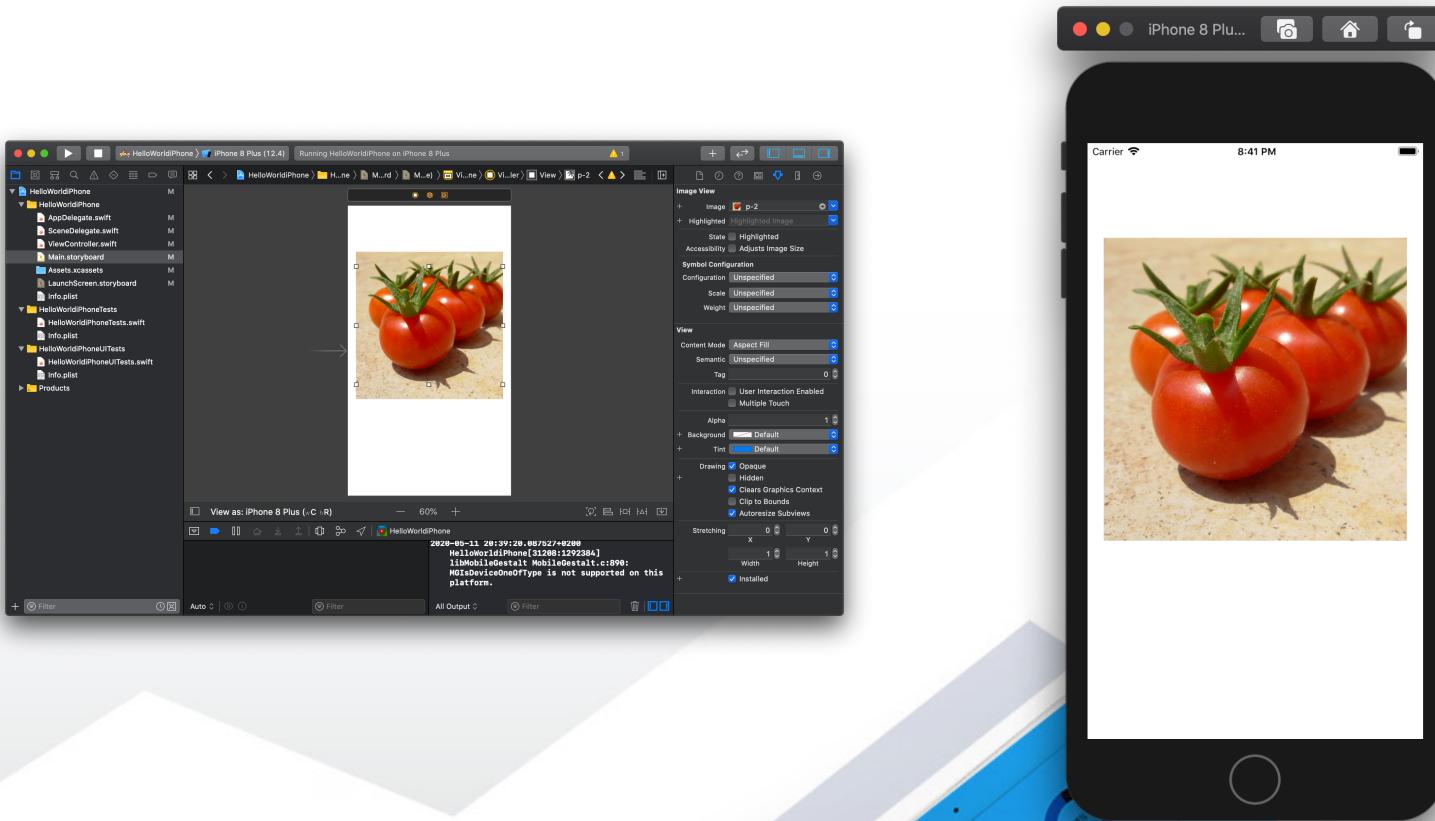
Dodawanie assetów



Dodawanie obrazu do widoku (1)



Dodawanie obrazu do widoku (2)



UIGestureRecognizer – rozpoznawanie dotyku (1)

Klasa `UIGestureRecognizer` umożliwia obsługę różnego rodzaju gestów. Gesty mogą być rozpoznawane automatycznie, aby to umożliwić należy przypisać do odpowiednich obiektów znajdujących się na widoku odpowiednie obiekty rozpoznawaczy gestów.

Możemy skorzystać z następujących gestów:

- puknięcie lub pewna ilość puknięć
(`UITapGestureRecognizer`)



UIGestureRecognizer – rozpoznawanie dotyku (2)

- szczypanie (UIPinchGestureRecognizer)
- przeciąganie (UIPanGestureRecognizer)
- machnięcia (UISwipeGestureRecognizer)
- obracanie (UIRotationGestureRecognizer)
- długie przyciśnięcie
(UILongPressGestureRecognizer)



UIGestureRecognizer – rozpoznawanie dotyku (3)

Aby dokonać połączenia należy w kodzie lub za pomocą *Interface Buildera* stworzyć obiekt rozpoznawacza i przypisać go do konkretnego elementu interfejsu, należy przy tym pamiętać, że:

- jeden rozpoznawacz może być związany z jednym widokiem, natomiast jeden widok może mieć wiele rozpoznawaczy
- jeśli gesty mają być rozpoznawane, należy dołączyć obiekt *gesture recognizer* do widoku
- *gesture recognizer* jako pierwszy (przed widokiem) otrzymuje informacje o dotyku



Dodawanie gesture recognizer'a i przypisywanie go do obiektu

The screenshot shows the Xcode interface with the storyboard and code editor open. In the storyboard, there is a main view containing an image of tomatoes and a play button. In the code editor, the ViewController.swift file contains the following code:

```
1 // ViewController.swift
2 // HelloWorldPhone
3 // Created by Andrzej Wilczyński on 10/05/2028.
4 // Copyright © 2028 Cracow University of Technology. All rights reserved.
5
6 import UIKit
7
8 class ViewController: UIViewController {
9
10     override func viewDidLoad() {
11         super.viewDidLoad()
12         // Do any additional setup after loading the view.
13     }
14
15     @IBOutlet var play: UIImageView!
16
17 }
```

An orange arrow points from the text "Add outlet" to the line "@IBOutlet var play: UIImageView!". The bottom status bar shows the build log: "2028-05-11 20:48:49.930987+0200 HelloWorldiPhone[312A7:1293885] libMobileGestalt MobileGestalt.c:890: MOIsDeviceOneOffType is not supported on this platform."

The screenshot shows the Xcode interface with the storyboard and code editor open. In the storyboard, there is a main view containing an image of tomatoes and a play button. In the code editor, the ViewController.swift file contains the following code:

```
1 // ViewController.swift
2 // HelloWorldPhone
3 // Created by Andrzej Wilczyński on 10/05/2028.
4 // Copyright © 2028 Cracow University of Technology. All rights reserved.
5
6 import UIKit
7
8 class ViewController: UIViewController {
9
10     override func viewDidLoad() {
11         super.viewDidLoad()
12         // Do any additional setup after loading the view.
13     }
14
15     let test = UITapGestureRecognizer(target: self, action: #selector(ViewController.playPressed))
16     play.addGestureRecognizer(test)
17
18     @objc func playPressed() {
19         //DO SOMETHING
20     }
21
22     @IBOutlet var play: UIImageView!
23 }
```

A red arrow points from the text "Check interaction" to the line "play.addGestureRecognizer(test)". The bottom status bar shows the build log: "2028-05-11 20:48:49.930987+0200 HelloWorldiPhone[312A7:1293885] libMobileGestalt MobileGestalt.c:890: MOIsDeviceOneOffType is not supported on this platform."



Swift – obsługa JSON (1)

Swift zapewnia wsparcie dla zapytań HTTP oraz dla dekodowania JSON'a. Funkcjonalność konwersji formatu JSON na rzeczywiste obiekty zdefiniowane w aplikacji zapewnia protokół *Codable*. Dekodowanie JSONA'a odbywa się za pomocą dwóch klas:

- JSONEncoder
- JSONDecoder

Klasy te obsługują wszelkie inne klasy i struktury, które implementują protokół *Codable*.



Swift – obsługa JSON (2)

Przykład:

```
struct Student: Codable{
    let name:String
    let lastname:String?
}

let str = "{\"name\": \"Harry\", \"lastname\": \"Potter\"}";
let jsonData = str.data(using: .utf8)!;
let jsonDecoder = JSONDecoder();
let decodedString = try! jsonDecoder.decode(Response.self, from:
    jsonData)
```



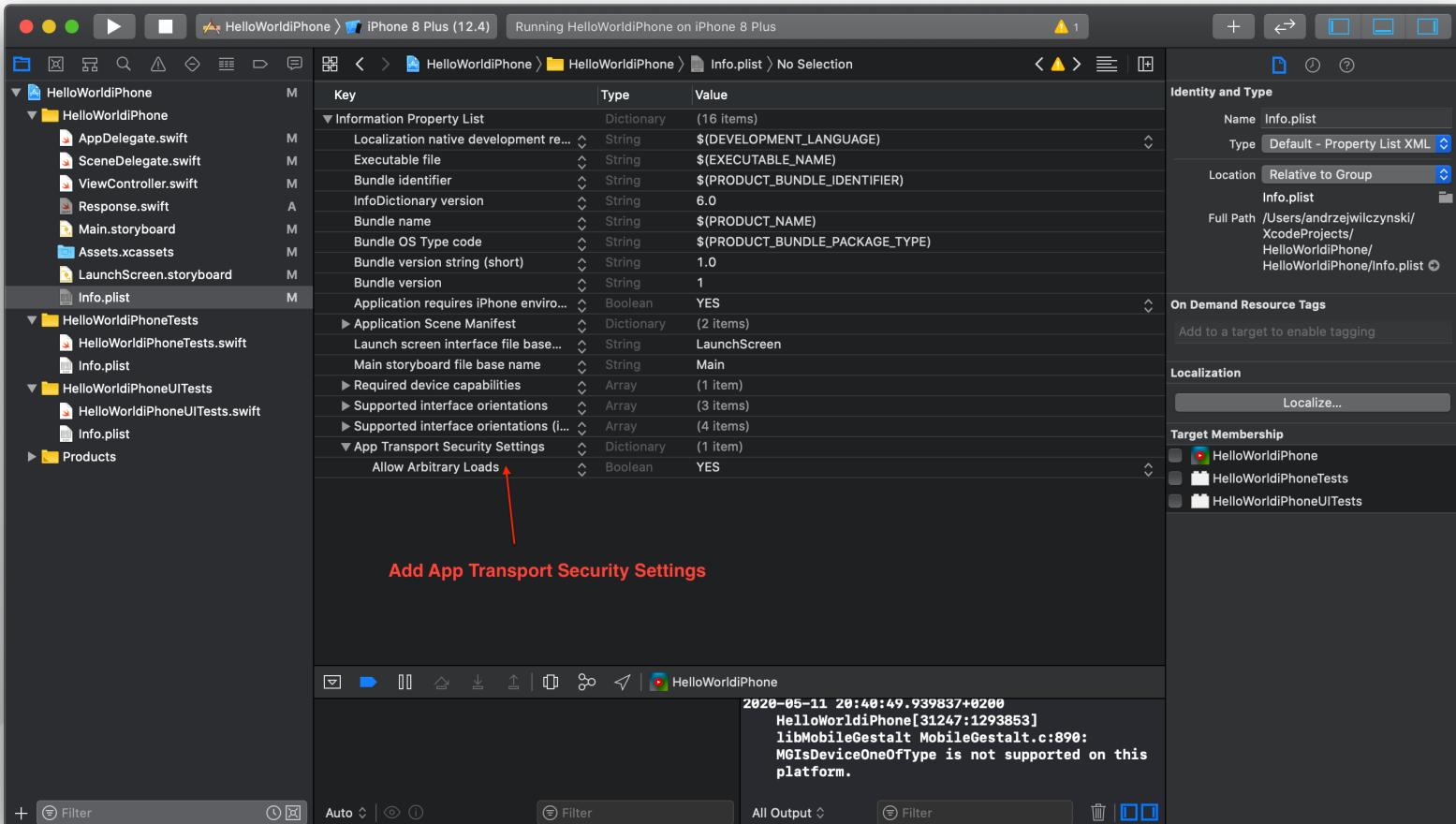
Swift – pobieranie danych z sieci, zapytania po http

Domyślnie w aplikacji dozwolone są tylko zapytania po bezpiecznym protokole https. Jeśli aplikacja ma korzystać również z połączeń http należy zezwolić na tego typu operacje poprzez dodanie odpowiedniego wpisu w pliku Info.plist:

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key>
<true/>
</dict>
```



Klucz pozwalający na korzystanie z protokołu HTTP



Swift – pobieranie danych z sieci

Przykład:

```
struct Response: Codable{
    let name:String
    let lastname:String?
}

let urlString = "www.awilczynski.me/testrest.php"
let content = try! String(contentsOf: URL(string: urlString)!)
let jsonData = content.data(using: .utf8)!
let jsonDecoder = JSONDecoder()
let response = try! jsonDecoder.decode(Response.self, from: jsonData)
print(response)
```

Dziękuję! ☺



Bibliografia

1. Matt Neuburg, *OS 12. Wprowadzenie do programowania w Swiftie. Wydanie V*, Helion, 2019
2. *Model-View-Controller (MVC) in iOS – A Modern Approach*, 11.05.2020, <https://www.raywenderlich.com/1000705-model-view-controller-mvc-in-ios-a-modern-approach>

