UDC 004.021

DESIGNING THE GRAPHIC INTERFACE FOR IOS

ALEXANDER VECHEROV, OKSANA GOLUBEVA
Polotsk State University, Belarus

The article discusses the development principles of the graphical user interface in the mobile application for iOS using various technologies.

Introduction

Each operating system has its own principles of UI building. iOS uses XML-files – also called Storyboards, or code markup creation. Location of objects. The arrangement of objects is formed through the setting of the frame, coordinates and size of the interface elements, or using the Autolayout system, which specifies mutual indentation.

This article describes UI creation technologies for iOS in detail.

Main body

MVS (Model-View-Controller) pattern is usually used in application development for iOS. The main idea of this pattern is simple – duty separations: the controller processes user's activity (clicks on the buttons, server's requests etc.); the model provides data that the user needs; theView provides data presentation from the model. [1]

Storyboard is a markup file that provides a developer with a convenient set of tools for building a graphical interface. [2] Despite the fact that the file uses XML markup language, the programmer does not need to know it, because the principle of operation is based on dragging items onto the desktop of the file and specifying the properties of this object through the menu.

This technology has its own advantages:

– Layout visibility and simplicity
– Simple creation of transitions
– All that you need to know is the identifier of the transition (segue id)
– SizeClasses – the ability to specify the location of the elements for different device screens

However, there are some disadvantages:

– Complexity when working in a team, these files are very difficult to merge
– Most of the animation code cannot be inserted into these files
– Huge load in case of the increase in a project size
– Limited selection of layout elements

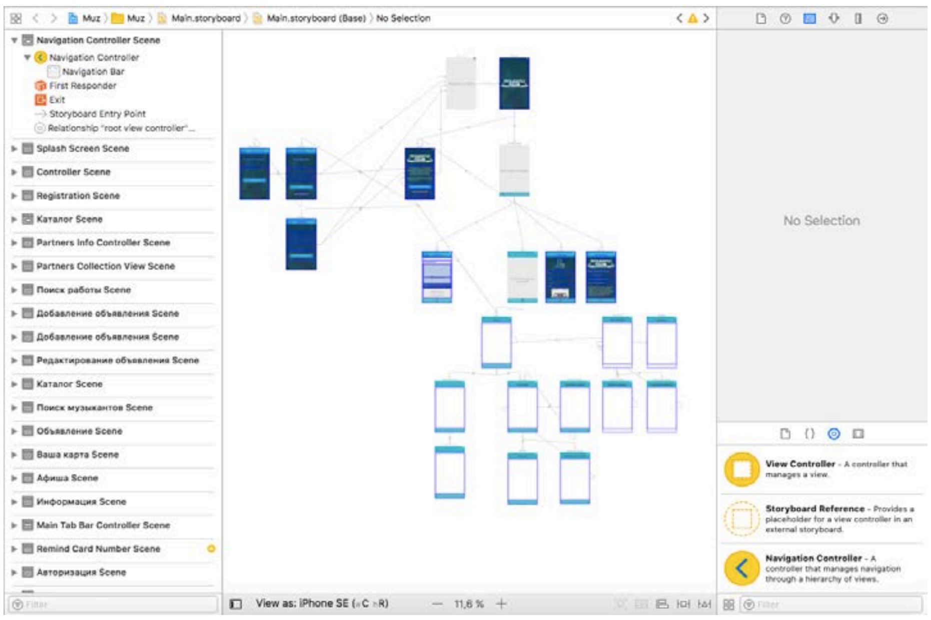
Muz
Muz
Main.storyboard
Main.storyboard (Base)
No Selection
Navigation Controller Scene
Navigation Controller
Navigation Bar
First Responder
Exit
Storyboard Entry Point
Relationship "root view controller"...
Splash Screen Scene
Controller Scene
Registration Scene
Каталог Scene
Partners Info Controller Scene
Partners Collection View Scene
Поиск работы Scene
Добавление объявления Scene
Добавление объявления Scene
Редактирование объявления Scene
Каталог Scene
Поиск музыкантов Scene
Объявление Scene
Ваша карта Scene
Афиша Scene
Информация Scene
Main Tab Bar Controller Scene
Remind Card Number Scene
Авторизация Scene
Filter
View as: iPhone SE (wC hR)
11,6 %
No Selection
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.
Navigation Controller - A controller that manages navigation through a hierarchy of views.
Filter

Fig. 1. Storyboard markup file

All segues are represented by arrows between screens, each transition has its own identifier to access them. In the bottom righthand corner there is a menu with a selection of available elements, however their list is limited. [3]

Manual marking is prescribed in the Controller files and the View. However, you can do this in two ways:

– Via frames

– Via Autolayout system

Working with frames is the direct indication of the location coordinates of the interface elements, as well as their sizes. This method is not the best solution, because it does not provide sufficient flexibility; When you change the orientation of the screen, the interface cannot rebuild itself. However, this method works faster with resource-intensive elements than Autolayout.

Autolayout is a GUI system that dynamically calculates the position and size of elements based on certain constraints assigned to these elements. The advantage of this system is its flexibility. There is no need to fit the sizes and positions of the elements for different screens and states. To facilitate the work with Autolayout, there are many open frameworks: SnapKit, PureLayout and other. [4]

The advantage of manual markup before marking through Storyboard is as follows:

– Flexibility

– Speed

– You can work with the arrangement of elements in the process of working with the application

– It's easier to work with animation

– No restrictions in the choice of elements, because you can write your own and use the readymade ones.

However, the use of manual markup does not provide clarity, the application gets more coding.

An example of manual markup is shown in Listing 1.

```
class VKCheckbox: UIView
{
   var line = VKCheckboxLine.Normal
   var button    = UIButton()
   var checkmark = VKCheckmarkView()
   override init(frame: CGRect) {
      super.init(frame: frame)
      self.setupView()
   }
   required init?(coder aDecoder: NSCoder) {
      super.init(coder: aDecoder)
      self.setupView()
   }
   func setupView() {
      self.backgroundColor = UIColor.clear
      self.cornerRadius = 8
      self.borderWidth = 3
      self.borderColor = UIColor.darkGray
      self.color = UIColor(red: 46/255, green: 119/255, blue: 217/255, alpha: 1)
      self.checkmark.frame = self.bounds
      self.checkmark.autoresizingMask = [.flexibleWidth, .flexibleHeight];
      self.addSubview(self.checkmark)
      self.button.frame = self.bounds
      self.button.autoresizingMask = [.flexibleWidth, .flexibleHeight];
      self.button.addTarget(self, action: #selector(VKCheckbox.buttonDidSelected), for: .touchUpInside)
      self.addSubview(self.button)
   }
   override func layoutSubviews() {
      super.layoutSubviews()
      self.button.bounds    = self.bounds
      self.checkmark.bounds = self.bounds
   }
}
```

Conclusion.

This article discusses the main ways to design a graphical interface for mobile applications for the iOS operating system, as well as their pros and cons. The examples of work with each of the systems are presented.

REFERENCES

1. Cocoa Core Competencies: Model-View-Controller / Apple Inc. [Electronic resource]. – Mode of access: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html. – Date of access: 09.02.2018.
2. Cocoa Application Competencies for iOS: Storyboard / Apple Inc. [Electronic resource]. – Mode of access: https://developer.apple.com/library/content/documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html. – Date of access: 10.02.2018.
3. Storyboards Tutorial for iOS: Part 1 / Nicholas Sakaimbo // Razeware [Electronic resource]. – Mode if access: https://www.raywenderlich.com/160521/storyboards-tutorial-ios-11-part-1. – Date of access: 10.02.2018.
4. Auto Layout Guide: Understanding Auto Layout / Apple Inc. [Electronic resource]. – Mode of access: https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG. – Date of access: 10.02.2018.