УДК 004.021

ПРОЕКТИРОВАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ «МУЗЫКАНТЫ РОССИИ» ПОД ОПЕРАЦИОННУЮ СИСТЕМУ iOS

А.В. ВЕЧЕРОВ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматриваются принципы построения графического интерфейса пользователя в мобильном приложении под операционную систему iOS. Показаны их плюсы и минусы. Представлены примеры работы с каждой из систем.

В каждой операционной системе свои принципы построения графического интерфейса. В iOS для этого используются XML-файлы – Stroyboards, либо программное создание разметки посредством кода. Расположение объектов формируется через задание рамки, координат и размера элементов интерфейса либо с помощью системы Autolayout-ов, которые задают взаимные отступы.

Данная работа направлена на подробное описание технологий построения разметки в iOS.

Как правило, приложения в iOS строятся по шаблону MVC (Model-View-Controller или Модель-Вид-Контроллер). Идея данного шаблона проста – разделение обязанностей: задача контроллера – обработка действий пользователя (нажатия по кнопкам, обработка запросов к серверу и т.д.); модель предоставляет контроллеру представление данных, запрашиваемых пользователем; вид, в свою очередь, обеспечивает представление данных, полученных из модели.

Storyboard – файл разметки, который предоставляет разработчику удобный набор инструментов для построения графического интерфейса. Несмотря на то, что файл использует язык разметки XML, программисту не нужно его знать, так как принцип работы построен на перетаскивании элементов на рабочий стол файла и указании свойств данного объекта через меню.

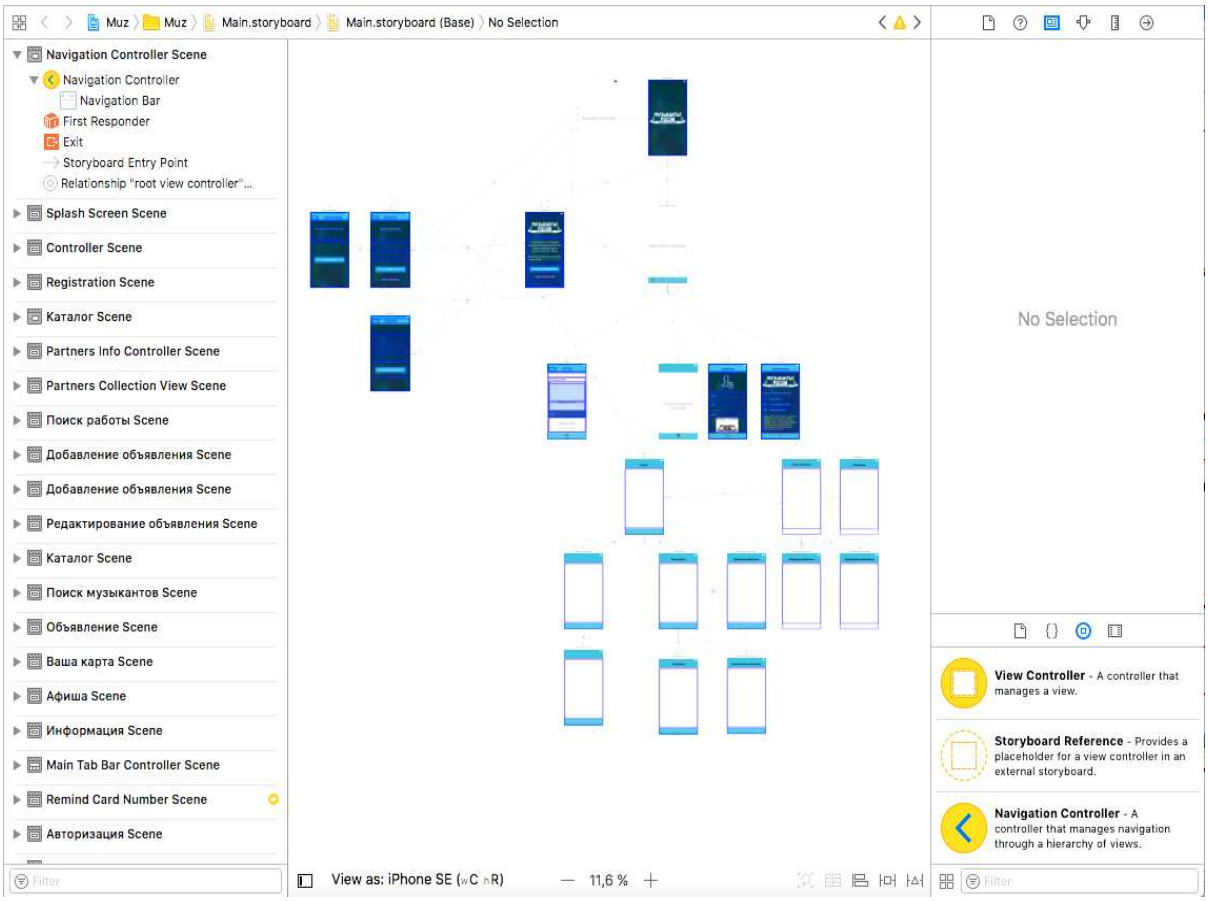

Рисунок 1. – Storyboard файл разметки

Данная технология имеет свои плюсы:
- наглядность и простота верстки;
- простое формирование переходов между экранами приложения;
- не нужно знать тип перехода, достаточно указать его идентификатор;
- SizeClasses – возможность задавать расположение элементов для различных экранов устройств

Однако есть и минусы:
- сложность при работе в команде, данные файлы плохо поддаются слиянию;
- большую часть кода анимации не вставить в данные файлы;
- большая нагрузка в случае увеличения размеров проекта;
- ограниченный выбор элементов разметки.

Как видно из рисунка 1, все переходы представлены стрелками между экранами, у каждого перехода есть свой идентификатор для доступа к ним. Также в правом нижнем углу расположено меню с выбором доступных элементов, однако список их ограничен.

Ручная разметка прописывается в файлах Controller-ов и View. Однако сделать это можно двумя способами:
- через фреймы;
- через систему Autolayout.

Работа с фреймами заключается в непосредственном указании координат расположения у элементов интерфейса, а также их размеров. Данный способ не является лучшим решением, так как он не предоставляет достаточной гибкости, т.е. при изменении ориентации экрана интерфейс не перестроится сам, придется предусматреть и этот случай. Однако данный способ работает с ресурсоемкими элементами быстрее, чем Autolayout.

Autolayout – система работы с GUI, которая динамически вычисляет позицию и размер элементов, на основе определенных правил (constraints), заданных для этих элементов. Преимущество данной системы в гибкости, так как нет необходимости в подгонке размеров и положений элементов для различных экранов и состояний. Для облегчения работы с Autolayout существует множество открытых фреймворков: SnapKit, PureLayout и др.

Преимущество ручной разметки перед разметкой через Storyboard состоит в следующем:
- гибкость;
- быстрая работа;
- можно работать с расположением элементов в процессе работы с приложением;
- проще работать с анимацией;
- отсутствие ограничений в выборе элементов, так как можно написать свои элементы и использовать готовые.

Однако использование ручной разметки не дает наглядности, в приложении становится больше кода.

Пример ручной разметки представлен в листинге 1.

Листинг 1 – пример ручной разметки

```
classVKCheckbox: UIView
{
   var line = VKCheckboxLine.Normal
var button    = UIButton()
   var checkmark = VKCheckmarkView()
   overrideinit(frame: CGRect){
      super.init(frame: frame)
      self.setupView()
   }
   requiredinit?(coder aDecoder: NSCoder){
      super.init(coder: aDecoder)
      self.setupView()
   }
   funcsetupView(){
      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)
    }
    overridefunclayoutSubviews(){
        super.layoutSubviews()
        self.button.bounds    = self.bounds
        self.checkmark.bounds = self.bounds
    }
}
```

Заключение

Рассмотрены основные способы построения графического интерфейса для мобильных приложений под операционную систему iOS, а также их плюсы и минусы. Представлены примеры работы с каждой из систем.

ЛИТЕРАТУРА

1. Ruseller [Электронный ресурс] Частная коллекция качественных материалов для тех, кто делает сайты. – Режим доступа: https://ruseller.com/lessons.php?id=666. – Дата доступа: 26.09.2017.
2. Habrahabr [Электронныйресурс] Ручная верстка vsStoryboard/Nib. – Режим доступа: https://habrahabr.ru/post/osbd/271355/. – Дата доступа: 26.09.2017.
3. Apple Inc [Электронныйресурс] Understanding Auto Layout. – Режим доступа: https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG/index.html. Дата доступа: 26.09.2017.
4. EnvatoTutsPlus [Электронныйресурс] iOS Fundamentals: Frames, Bounds and CGGeometry. – Режим доступа: https://code.tutsplus.com/tutorials/ios-fundamentals-frames-bounds-and-cggeometry--cms-21196. – Дата доступа: 26.09.2017.