УДК 004.438
М.М. Поліщук, Ю.С. Повстяна, О.Р. Кулакевич
Луцький національний технічний університет

ВИКОРИСТАННЯ МОВИ ПРОГРАМУВАННЯ SWIFT ДЛЯ РЕАЛІЗАЦІЇ ВЛАСНОГО IOS ПРОЕКТУ

М.М. Поліщук, Ю.С. Повстяна, О.Р. Кулакевич. Використання мови програмування Swift для реалізації власного iOS проекту. У статті розглянуто сучасні технології та інструменти проектування, які компанія Apple надає розробникам програмного забезпечення для власних операційних систем iOS, macOS, watchOS та tvOS. Проведено аналіз перспектив впровадження у власні проекти відносно нової мови програмування Swift, а також продемонстровано етапи створення власного iOS додатку для IPhone.
Ключові слова: Apple, Objective-C, Swift, Swift Framework, UIKit, MacBook, macOS, мобільний додаток, інтегроване середовище розробки, Xcode

Н.Н. Полищук, Ю.С. Повстяна, О.Р. Кулакевич, Использование языка программирования Swift для реализации собственного iOS проекта. В статье рассмотрены современные технологии и инструменты проектирования, которые компания Apple предоставляет разработчикам программного обеспечения для собственных операционных систем iOS, macOS, watchOS и tvOS. Проведен анализ перспектив внедрения в собственные проекты относительно нового языка программирования Swift, а также продемонстрировано этапы создания собственного iOS приложения для IPhone.
Ключевые слова: Apple, Objective-C, Swift, Swift Framework, UIKit, MacBook, macOS, мобильное приложение, интегрированная среда разработки, Xcode

M.M. Polischuk, Y.S. Povstyana, O.R. Kulakevich. Using Swift programming language to implement a personalized iOS project. The article discusses modern technologies and design tools that Apple provides software developers for their own operating systems iOS, macOS, watchOS and tvOS. The analysis of the prospects for implementation in their own projects regarding the new Swift programming language has been carried out, and the stages of creating their own iOS apps for the iPhone have been demonstrated.
Keywords: Apple, Objective-C, Swift, Swift Framework, UIKit, MacBook, macOS, mobile application, integrated development environment, Xcode

Постановка проблеми: Щороку компанія Apple проводить конференцію The worldwide developers conference (WWDC), презентуючи нові технології та розвиток власного програмного забезпечення. Саме там збираються розробники мобільного програмного забезпечення Apple, які обговорюють сучасний ринок технологій, що розвивається з неабиякою швидкістю. Якщо раніше, будь-яка компанія для представлення власної продукції чи послуг, потребувала лише веб - сайту, то зараз для цього потрібно мати власний мобільний додаток, щоб у декілька дій замовити послугу і відразу ж його оплатити. Користуючись мобільними додатками стало значно простіше вивчати іноземну мову, замовити таксі, знайти необхідне місце на карті. Мобільний додаток дозволяє аналізувати дії користувачів, їх бажання щодо продукції компанії та на основі цього пропонувати контекстну рекламу, яка в окремих випадках значно скорочує час пошуку необхідного користувачу. На сьогоднішній день стек технологій, які пропонує компанія Apple для розробників є повноцінним та багатим функціональністю, при правильному його використанні переваги мобільної розробки над веб - розробкою неабияк зростають.

Аналіз останніх досліджень і публікацій: Основним джерелом інформації за даною тематикою, є технічні статті представників компанії Apple, власників тематичних ресурсів присвячених розробці під операційні системи Apple, журналістів, які ставлять за мету донести своїм читачам необхідну інформацію про актуальні зміни технологій. Проаналізувавши кілька веб - ресурсів таких як: SwiftBook, Hacking with Swift, Medium, можна побачити, що кожен ресурс пропонує власний підхід до створення мобільних додатків, використання тих чи інших технологій. Для початківця в сфері розробки варто визначитися, з мовою програмування, фреймворком, який стане фундаментом майбутнього проекту, та принципом взаємодії користувача із застосунком. Ті чи інші питання розкриті досить поверхнево у вузько направлених статтях, більшість яких написані для досвідчених програмістів. Тому доцільно розглянути тему швидкого входу у сферу мобільної розробки, та акцентувати моменти, на які варто звернути увагу.

Мета даної роботи полягає у дослідженні необхідних технологій для розробки власного мобільного додатку, аналізу перспектив використання мови програмування Swift, та представлення ряду етапів розробки додатку з кінцевим представленням iOS програми *"Мої улюблені місця"*.

Виклад основного матеріалу дослідження. iOS – це власницька мобільна операційна система від Apple. Розроблена спочатку для iPhone, згодом вдосконалена також для iPad, iPod Touch та Apple TV (до 9 вересня 2015, коли на спеціальному заході Apple було представлено tvOS). iOS є

похідною від OS X, отже, є за своєю природою Unix-подібною операційною системою. Користувацький інтерфейс iOS заснований на концепції прямої маніпуляції з використанням жестів Multi-Touch. Елементи інтерфейсу управління складаються з повзунків, перемикачів і кнопок. Він призначений для безпосереднього контакту користувача з екраном пристрою [1]. 10 липня 2008 року компанія представила магазин застосунків App Store, як розширення для iTunes, тим самим надала програмістам технології для створення та реалізації власних мобільних додатків. Основним інструментом для розробки програми для iOS стало інтегроване середовище розробки Xcode із вбудованими можливостями побудови візуального інтерфейсу, фреймворками, та основною мовою програмування у системі для розробників стала Objective - C.

Objective-C – це компільована об'єктно-орієнтована мова програмування корпорації Apple, побудована на основі мови C і парадигм Smalltalk. На відміну від C++, мова Objective-C повністю сумісна з Сі (мова Objective-C є варіацією мови Сі) і код на Сі компілюється. Об'єктна модель побудована в стилі Smalltalk, тобто об'єктам надсилаються повідомлення. Компілятор Objective-C входить в GCC і доступний на більшості основних платформ. Мова використовується в першу чергу для Mac OS X (Cocoa) і GNUstep - двох реалізацій об'єктно-орієнтованого інтерфейсу OpenStep. ObjC був створений Бредом Коксом на початку 1980х в його компанії Stepstone.

Ще однією з особливостей Objective-C є те, що вона message-oriented в той час як C++ - function-oriented. Це означає, що в ній виклики методу інтерпретуються не як виклик функції (хоча до цього зазвичай все зводиться), а саме як посилка повідомлення (з ім'ям і аргументами) об'єкту, подібно до того, як це відбувається в Smalltalke. Такий підхід дає цілий ряд плюсів - так будь-якого об'єкта можна послати будь-яке повідомлення. Об'єкт може замість обробки повідомлення просто переслати його іншому об'єкту для обробки (так зване делегування), зокрема саме так можна легко реалізувати розподілені об'єкти (тобто об'єкти, що знаходяться в різних адресних просторах і навіть на різних комп'ютерах). Прив'язка повідомлення до відповідної функції відбувається безпосередньо на етапі виконання.

У мові Objective-C є підтримка протоколів (тобто поняття інтерфейсу об'єкта і протоколу чітко розділені). Для об'єктів підтримується спадкування (не множинне), для протоколів підтримується множинне спадкування. Об'єкт може бути успадкований від іншого об'єкта і підтримувати відразу кілька протоколів.

Структура іменування файлів: файли з розширенням - h є заголовками з описом класів, функцій також як в C і C ++, файли з розширенням - m відповідно містять реалізацію класів і методів. [2]

Разом із мовою програмування Objective - C програмістам необхідно було надати інструменти розробки, одними з яких стали IDE Xcode та iOS SDK - комплект засобів розробки для iOS, випущений в березні 2009 року корпорацією Apple. iOS SDK випускається тільки для Mac OS X. 17 жовтня 2007 року Стів Джобс, засновник компанії Apple, анонсував SDK, який повинен був бути наданий стороннім розробникам в лютому 2008 року. Однак інструментарій вийшов лише 6 березня, він дозволяє розробляти додатки під iPhone, iPod Touch і iPad, а також тестувати їх на емуляторі iPhone. Проте, завантаження програми на пристрої можлива тільки після оплати ліцензії. Починаючи з Xcode 3.1 він є засобом розробки для iOS SDK.

Xcode — інтегроване середовище розробки (IDE) виробництва Apple. Дозволяє створювати програмне забезпечення з використанням таких технологій як GCC, GDB, Java та ін. На сьогодні є єдиним засобом написання «універсальних»(Universal Binary) прикладних програм для Mac OS X. Xcode включає в себе більшу частину документації розробника від Apple та Interface Builder — застосунок, який використовується для створення графічних інтерфейсів.

Пакет Xcode містить змінену версію вільного набору компіляторів GNU Compiler Collection і підтримує мови C, C++, Objective-C, Swift, Java, AppleScript, Python і Ruby з різними моделями програмування, включаючи (але не обмежуючись) Cocoa, Carbon і Java [3].

Apple iOS використовує варіант того ж ядра XNU, що і Mac OS X. SDK розбита на наступні набори: Cocoa Touch - мультитач управління, підтримка акселерометра, ієрархія видів, локалізація, підтримка камери; мультимедіа - OpenAL, мікшування і запис аудіо, відтворення відео, формати зображень, Quartz, анімаційне ядро, OpenGL ES; сервісне ядро - Мережа, Вбудована база даних SQLite, локаційне ядро, Багатозадачність, CoreMotion; ядро OS X - TCP / IP, сокети, управління харчуванням, файлова система, Безпека

Поряд з набором інструментів Xcode, SDK містить iPhone Simulator, який використовується для імітації зовнішнього вигляду iPhone на комп'ютері розробника, що раніше називався «Aspen Simulator».

Варто взяти до уваги потужний фреймворк Cocoa для операційної системи Mac, який у свої молодшій реалізації для iOS інтерпретується, як Cocoa Touch. [4].

Cocoa Touch — це фреймворк для створення додатків для пристроїв під управлінням операційної системи iOS (iPhone, iPod Touch, iPad, тощо).

Фреймворк Cocoa Touch надає рівень абстракції для iOS (операційної системи iPhone, iPad і iPod touch). Cocoa Touch реалізована на базі класів фреймворка Cocoa, що використовується в Mac OS X. Як і Cocoa, Cocoa Touch використовує мову розробки Objective-C або Swift. Cocoa Touch спроектований за шаблоном проектування Model-View-Controller. Інструменти для розробки додатків за допомогою Cocoa Touch присутні в iOS SDK. Основні технології та можливості, присутні в Cocoa Touch: Core Animation, багатозадачність, розпізнавачі мультитач-жестів.

Cocoa Touch надає основні фреймворки для розробки додатків для пристроїв на базі iOS. Ось деякі з них: Foundation Framework — основна бібліотека, яка містить класи з префіксом NS; UIKit Framework — бібліотека, яка містить специфічні для інтерфейсу користувача iOS класи; Game Kit Framework — бібліотека для взаємодії з сервісом Game Center; iAd Framework — бібліотека для створення сервісів контекстної реклами iAd у додатку; MapKit Framework – бібліотека, що здійснює взаємодію з картами та навігаційними можливостями iOS-пристроїв

iOS-технології можна розглядати як набір рівнів, де Cocoa Touch знаходиться на найвищому рівні, а Core OS та ядро Mac OS X — на більш низьких. Це дозволяє розробникам створювати програмний код на більш високому рівні, що дозволяє значно скоротити термін розробки. Але в той же час розробники мають можливість використовувати більш низькі рівні абстракції, якщо це необхідно.

Розташування рівнів абстрагування можна представити в наступному виді (від вищого до нижчого): Cocoa Touch, Media / Application Services, Core Services, Core OS / ядро Mac OS X [5].

Говорячи про Cocoa Touch варто відмітити, один із найважливіших його компонентів UIKit - це структура, яку ви найчастіше будете використовувати при розробці додатків для iOS. Він визначає основні компоненти програми iOS, від ярликів і кнопок до табличних уявлень і контролерів навігації. У той час як платформа Foundation визначає класи, протоколи і функції для розробки під iOS і OS X, платформа UIKit призначена виключно для розробки під iOS. Це еквівалент Application Kit або платформи AppKit для розробки під OS X.

Як і Foundation, UIKit визначає класи, протоколи, функції, типи даних і константи. Він також додає додаткові функціональні можливості до різних базових класів, таким як NSObject, NSString і NSValue, завдяки використанню категорій Objective-C.

З постійним прагненням Apple до оптимізації власних технологій, та активним розширенням можливостей операційної системи, які мова програмування Objective - C була не в змозі гнучко підтримувати, компанія прийняла рішення про створення абсолютно нової мови програмування Swift - багатопарадигмована компільована мова програмування, розроблена компанією Apple для того, щоб співіснувати з Objective C і бути стійкішою до помилкового коду. Swift була представлена на конференції розробників WWDC 2014. Мова побудована з LLVM компілятором, включеного у Xcode 6 beta.

Компілятор Swift побудований з використанням технологій вільного проекту LLVM. Swift успадковує найкращі елементи мов C і Objective-C, тому синтаксис звичний для знайомих з ними розробників, але водночас відрізняється використанням засобів автоматичного розподілу пам'яті і контролю переповнення змінних і масивів, що значно збільшує надійність і безпеку коду.

При цьому Swift-програми компілюються у машинний код, що дозволяє забезпечити високу швидкодію. За заявою Apple, код Swift виконується в 1.3 рази швидше коду на Objective-C. Замість збирача сміття Objective-C в Swift використовуються засоби підрахунку посилань на об'єкти, а також надані у LLVM оптимізації, такі як автовекторизація.

Мова також пропонує безліч сучасних методів програмування, таких як замикання, узагальнене програмування, лямбда-вирази, кортежі і словникові типи, швидкі операції над колекціями, елементи функційного програмування. Основним застосуванням Swift є розробка користувацьких застосунків для macOS, iOS, tvOS, watchOS з використанням тулкіта Cocoa і Cocoa Touch. При цьому Swift надає об'єктну модель, сумісну з Objective-C. Сирцевий код мовою Swift може змішуватися з кодом на C і Objective-C в одному проекті.

Swift щільно інтегровано до середовища розробки Xcode, проте може бути викликано з терміналу, що уможливлює її використання на операційний системах, відмінних від macOS, наприклад, на Linux.[6]

Розглянемо основні нововведення у Swift, саме playground (рис. 1), який виглядає як просте вікно редактора, куди можна писати код. Код цей відразу ж компілюється і виконується. Не потрібно збирати проект, а потім запускати емулятор, щоб подивитися. Все відбувається миттєво: пишеш код – бачиш результат. Класи більше не розбиваються на інтерфейс і реалізацію, що вдвічі зменшує кількість файлів в проекті, що в свою чергу спрощує навігацію по ньому, спрощено синтаксис створення полів і властивостей класів. Властивості більше не мають потреби в instance variables, що раніше в останніх версіях ObjC, ці iVars створювалися автоматично, але їх можна було прописувати і вручну. Тепер же їх не можна створити в принципі, а значить їх більше не потрібно враховувати. Властивості в Swift можуть бути змінними (оголошується як `var myField`) і константами (оголошується як `let myField`).

З'явилися Optional Types, вони використовуються, коли значення змінної з якоїсь причини може не бути. Додано нову функціональність для оператора switch, перерахувань enum, реалізовано generics та значно покращено модель керування пам'ятю ARC (Automatic Reference Counting), що зменшує кількість випадків втрати додатком оперативної пам'яті [7].

Розглянувши усі доступні технології, спробуємо реалізувати конкретний невеликий проект (додаток "Мої улюблені місця") мовою програмування Swift.

Оскільки розробка ведеться виключно у системі MacOS, потрібно завантажити IDE Xcode з Mac App Store. Після запуску, якого у вікні створення проекту обираємо пункт Create a new Xcode project, що дозволить вікрити Xcode у режимі написання власного проекту. По ліву сторону програми знаходиться область Navigator, по центру основний відділ роботи з проектом, та по правій стороні знаходиться Inspector, якій вміщує в собі усю необхідну інформацію, щодо налаштувань окремих об'єктів та їх властивостей. Розробка додатку ведеться за технологією Model - View - Controller, яку рекомендує використовувати власне Apple. Процес створення передбачає два варіанти написання інтерфейсу, а саме розміщення елементів інтерфейсу за допомогою коду, або ж за допомогою storyboard (файл main.storyboard), що є так званим візуальним полотном, яке надає можливість додати елементи керування на екран відображення, перетягнувши їх з бібліотеки об'єктів у власний проект, саме цей варіант ми оберемо для кращого розуміння наших дій.

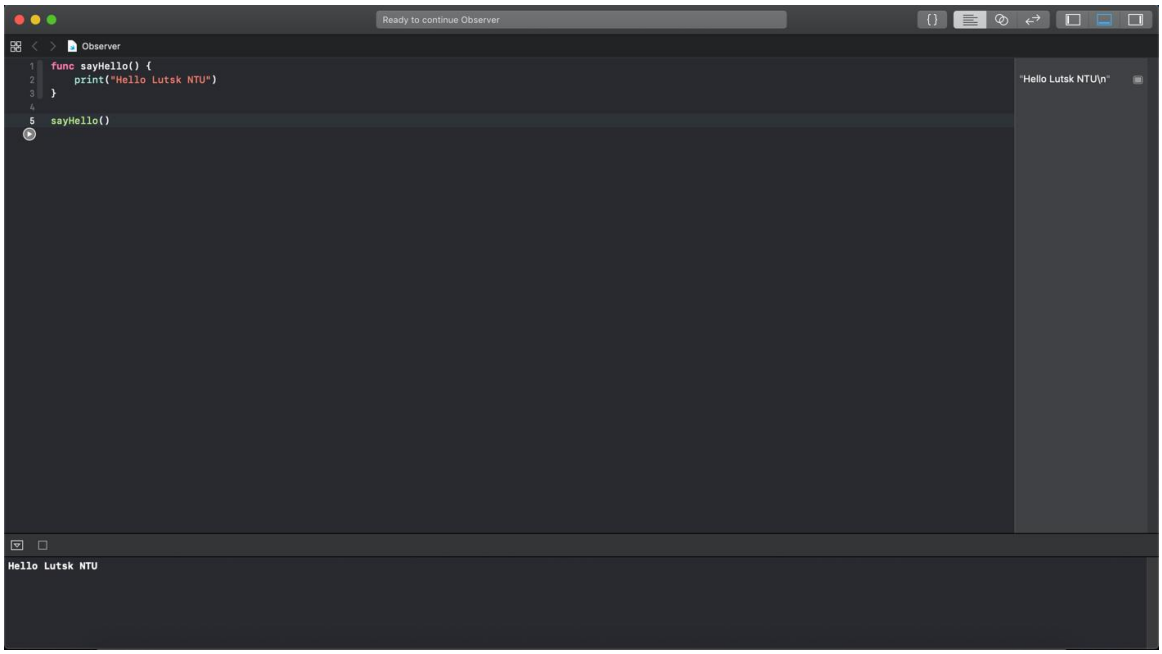

Рис. 1. Xcode playground

Кожен об'єкт у storyboard має власний клас, який передбачає у собі усю логіку роботи елементів екрану. У storyboard по замовчуванню при створенні нового проекту, вже є один екран, так званий ViewController, який запускається при старті додатку, використаємо його для відображення

списку улюблених місць. Для наших цілей ідеально підійде елемент керування TableView, який дозволяє розміщувати, контент у вигляді списку. Для використання цього елементу доцільно перетягнути його на ViewController та закріпити за допомогою constraints на екрані. Constraints - обмеження, які використовуються для фіксованого розміщення елементів, та закріплення їх відносно інших елементів, оскільки ряд мобільних пристроїв Apple чималий, та розміри екранів відрізняються, то компанія пропонує розробникам використовувати constraints для чіткого розуміння системою, де повинен розміщуватися то чи інший елемент відносно іншого. Для того аби TableView виглядав призентабильно та не займав усю площу екрану, прикріпимо до ViewController, такий вид екрану, як Navigation ViewController, що дозволяє додатку вгорі відобразити панель навігації, та активує змогу повертатися від одного екрану до іншого. Тепер коли в результаті ми має два екрана створимо посилання на об'єкт TableView у коді файлу ViewController.swift для цього використаємо IBOutlet, у Assistant Editor перетягнувши ПКМ від TableView до класу.

Робта із storyboard завершена, переходимо до файлу ViewController.swift де і реалізуємо логіку відображення наших улюблених місць. Даний клас унаслідується від класу UIViewController, що дозволяє перевизначити та використовувати методи суперкласу. Перший метод, який доступний у файлі – це viewDidLoad(), який активізується відразу після завантаження екрану відображення, саме у цьому методі варто викликати усі необхідні налаштування та ініціалізацію змінних. Для відображення будь-яких елементів у TableView, варто підписати клас ViewController на протокол делегата та власника даних, а саме UITableViewDelegate та UiTableViewDataSource, які вимагають реалізації двох обов'язкових методів cellForRow (повертає рядок із клітинкою, яка в майбутньому використовується об'єктом TableView) та numberOfRowInSection (повертає кількість рядків у секції, по замовчуванню це 0), також для tableView потрібно назначити клас делегатор та клас постачальник даних за допомогою фрагменту tableView.delegate = self та tableView.dataSource = self. Отож після реалізації цих методів, вже є можливість запустити симулятор та побачити результат, у вигляді декількох пустих рядків.

Для заповнення рядків потрібно отримати у об'єкта TableView його клітинку за допомогою методу `tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)`, та звернутися до властивості textLabel.text присвоївши їй значення типу String "Моя перша стрічка". Фрагменти коду реалізації даного екрану описані нижче по тексту та результат коду (рис. 2).

```
import UIKit

class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource
{

    @IBOutlet weak var tableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.delegate = self
        tableView.dataSource = self
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -
> Int {
        return 3
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for:
indexPath)
        cell.textLabel?.text = "My first string"
        return cell
     }
}
```

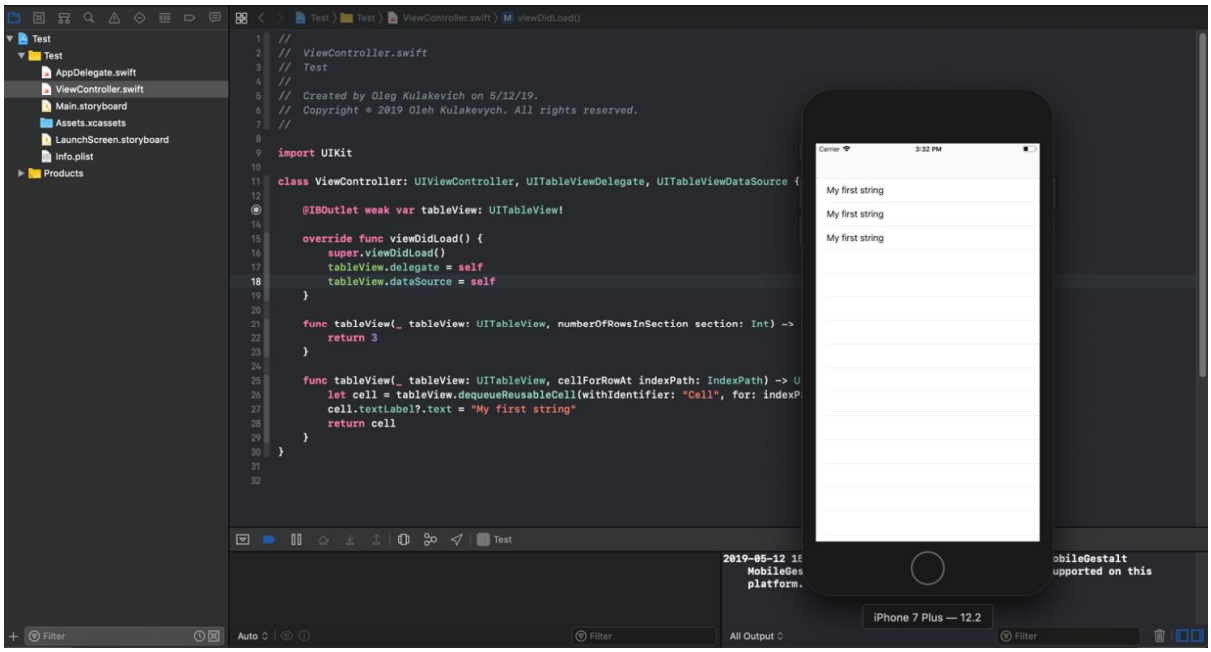

Рис. 2. Процес та результат реалізації об'єкту TableView

Саме за таким алгоритмом відбувається початкове створення більшості додатків. Swift надає потужні можливості, щодо налаштування користувацьких об'єктів, велику кількість доступних вже реалізованих об'єктів готових до використання у власних проектах. Реалізувавши декілька методів, та перетягнувши декілька об'єктів, вже є можливість побачити готовий результат своєї роботи. Оскільки готовий проект у даній статті передбачає декілька сторінок коду. На рис. 3 зображено storyboard готового проекту, та на рис. 4 та 5 результат виконання програми у вигляді симулятора Iphone.

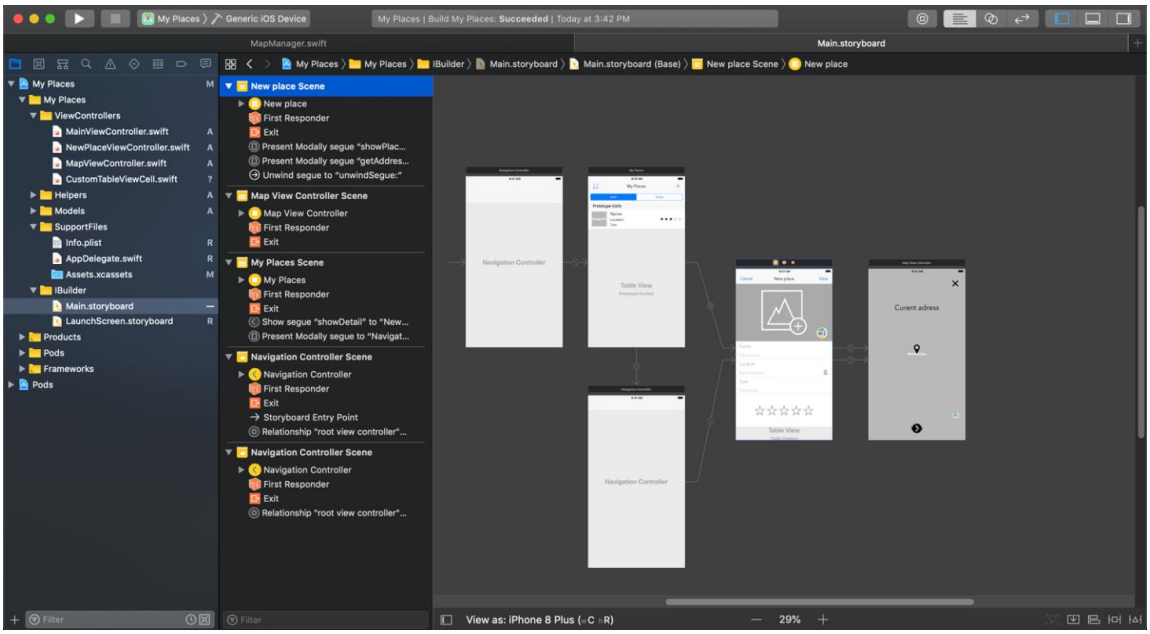

Рис. 3. StoryBoard проекту My Places

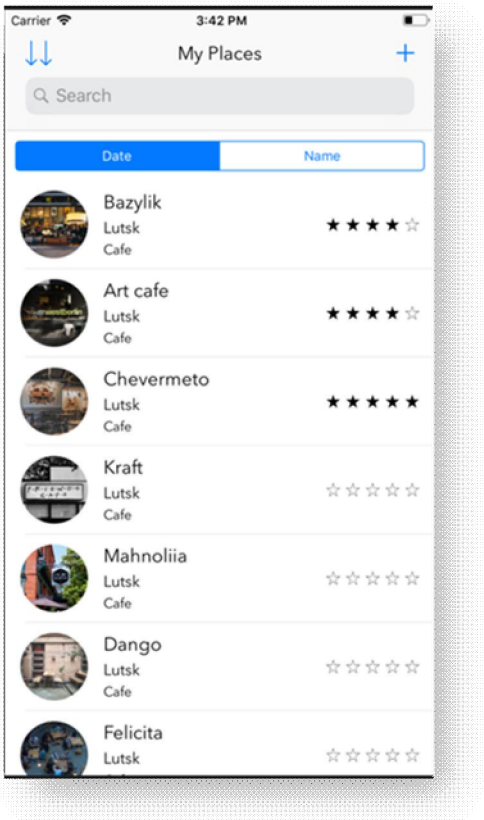

Рис. 4. Головний екран проекту My Places

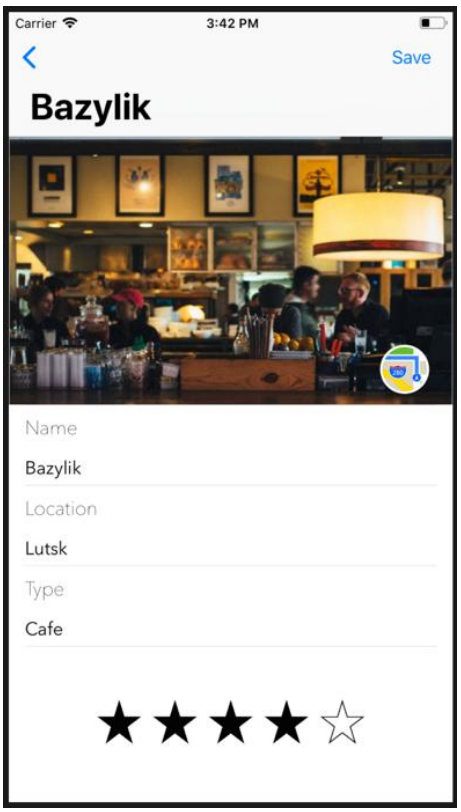

Рис. 5. Екран додавання місця у проекті My Places

Висновки: Актуальність використання мобільних додатків чимала, з кожним роком кількість розробників зростає, компанії намагаються активно впроваджувати нові технології програмування,

аби залучити як найбільше аудиторії до власних платформ. Країни намагаються задовільними рівень попиту на мобільну розробку навчаючи студентів різним технологіям, так як велика кількість компаній зараз готові оплати утримання висококваліфікованого програміста, який власними знаннями та креативністю розвиватиме технологічний рівень країни так і компанії в цілому.

В перспективі планується освоєння нових технологій серверного програмування мовою Swift та розробки за допомогою React Native, що використовується для швидкого програмування під усі популярні платформи, як iOS так і Android.

1. Objective-C [Електронний ресурс] // MacBug – Режим доступу до ресурсу: http://macbug.ru/cocoa/objc.
2. Xcode [Електронний ресурс]. – 17. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Xcode.
3. iOS SDK [Електронний ресурс]. – 14. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/IOS_SDK.
4. Cocoa Touch [Електронний ресурс]. – 8. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Cocoa_Touch.
5. Первые шаги с UIKit [Електронний ресурс] // 17 грудня 2012 – Режим доступу до ресурсу: https://code.tutsplus.com/ru/tutorials/first-steps-with-uikit--mobile-14013.
6. Swift и Objective-C: 10 отличий [Електронний ресурс]. – 20. – Режим доступу до ресурсу: https://dou.ua/lenta/articles/swift-vs-objective-c/.
7. Усов В. Swift. Основы разработки приложений под iOS и macOS / Василий Усов. – Санкт-Петербург: Питер, 2016. – 448 с. – (4 - е издание).
8. Bring Your Ideas to Life [Електронний ресурс] – Режим доступу до ресурсу: https://developer.apple.com/develop/.
9. Функциональное Программирование В Swift [Електронний ресурс] – Режим доступу до ресурсу: https://swiftbook.ru/post/tutorials/funkcionalnoe-programmirovanie-v-swift/.
10. Pylypenko D. Swift 5. What is new? [Електронний ресурс] / Dmytro Pylypenko – Режим доступу до ресурсу: https://medium.com/@dimpiax/swift-5-what-is-new-examples-fbb64d525486.