iOS Разработчик. Продвинутый курс v 2.0.

Вся мощь Swift 5.1 для развития профессиональных навыков уровня Middle/Senior iOS Developer

Длительность курса: 172 академических часа

1 SwiftUI и основы Combine

1 **Проектируем UI декларативно с SwiftUI. В чем отличия UIKit и SwiftUI**

– Возможности и ограничения SwiftUI
– Property Wrappers: @State, @Binding, @ObservedObject и ObservableObject, @EnvironmentObject, @Environment
– Основы Combine: @Published
– Решение организационных вопросов.
– Настройка environment: Xcode, git, scripts

Домашние задания

1 Создание каркаса приложения на SwiftUI

Цель: Студент
1. Будет целостно понимать навигационный стек SwiftUI/Combine
2. Получит умение сборки иерархии экранов на SwiftUI/Combine

Создать флоу экранов на SwiftUI
1. Добавить TabView
2. На втором табе сделать List с обернутый в NavigationView
2.1 Из листа должны быть переходы с NavigationLink
3. На третьем табе должна быть кнопка открывающая модальное окно
4. На первом табе должна быть кнопка открывающая второй таб и один из пунктов там
5. Протестировать на iPad/iPhone симуляторах, со сменой ориентации девайса

---

2 **Life cycles, Responder Chain, SceneDelegate, Hosting ViewControllers**

— Life cycle приложения
— UIResponder
— SceneDelegate
— Hosting ViewControllers и зачем они
— Combine

---

3 **Использование NavigationView, TabView, ScrollView и продвинутая работа с UIScrollView**

– Особенности навигации в SwiftUI
– Кейсы использования ScrollView в SwiftUI
– Бесконечный скроллинг
– Устройство UIScollView и что такое контейнеры в UIKit

---

4 **Отображение структурированных данных, List, пейджинг, кастомные компоненты на UIViewRepresentable**

– Использование List в SwiftUI
– Реализация пейджинга для List
– Создание кастомных компонентов с UIViewRepresentable и Coordinator

Домашние задания

1 Реализация коллекции таймеров на UICollectionView

Цель: – Студент научится использовать несколько layout для UICollectionView и переключать их
– Получит умение создавать реиспользуемые ячейки и контролировать в них состояние
– Прокачает навык решения проблем состояний

1. На экране Feed:
1.1 Добавить UIViewTable
1.2 Настроить dataProvider на тестовые данные (около 20 ячеек, например ["Item 1", "Item 2", "Item 3", ...]
1.3 Создать новый вьюконтроллер, назвать SessionSummaryViewController или типа того (Это может быть ваш Второй контроллер из домашки 1-3 про Life Cycle)
1.4 При tableView: didSelectRowAt программно инстансировать SessionSummaryViewController и переходить на него
2. На экране Benchmark
2.1 Добавить UICollectionView

2.2 Добавить UIStackView в ячейку UICollectionViewCell
2.3 Также в ячейку компонент бегущего таймера
2.4 Сделать UI для бегущего таймера
2.4 При тапе на ячейку паузить или запускать таймер
2.5 При выходе с таба Benchmark все таймеры должны инвалидироваться
2.6 Состояние таймеров должно сохранятся пока мы на табе Benchmark, при переходе все их сбрасывать и их UI

*3. (невлияющее на оценку) сделать переключение лейаутов на лету(как было показано на вебинаре) 2 или больше

5 **Создание кастомных Shape, SwiftUI Drawing and Animation API**

– Разница CoreGraphics и Drawing API SwiftUI
– Создание кастомных фигур на Drawing API SwiftUI
– Анимация интерфейса в SwiftUI

Домашние задания

1 Создание анимированный диаграммы на CoreGraphics

Цель: — Студент научится работать с CALayer и понимать систему координат используемую в CoreGraphics
— Узнает как компоновать анимации
— Сможет создавать графику с помощью CoreGraphics

1. Сделать компонент диаграммы типа PieChart
1.1 Создать интерфейс компонента для возможности установления:
1.1.1 Размера диаграммы
1.1.2 Количества секторов на диаграмме
1.1.3 Цвета секторов
1.1.4 Значений текста в секторах
1.2 В сектора сделать возможность добавлять текст с также с помощью CoreGraphics
2. Использовать компонент в ячейках CollectionView на экране Benchmark
2.1 Отображать на PieChart пропорцию времени когда таймер был включен, а когда запаузен
2.2 Обновлять пайчарты можно по кнопке в навбаре или в ячейке
3. *(продвинутый пункт) Добавить анимацию в сектора во время обновления пайчарта - каждый сектор стартует на 1% и растет до своего значения — получается типа расхлопывания веера

2 Современная архитектура мобильных приложений

1 **MVP, MVCS, MVVM, архитектурные паттерны, модуляризация, Архитектурные Rx паттерны**

— MV(X) архитектурные паттерны
— Модуляризация приложения, способы: Frameworks, Cocoapods, JSCore, SPM
— Архитектурные Rx паттерны

Домашние задания

1 Создания каркаса для модуляризированного приложения

---

2 **Современные паттерны проектирования, принцип SOLID и его целесообразное применение. Принципы GRASP**

— SOLID и как получить от него пользу
— Другие принципы: KISS, DRY/DIE, YAGNI, BDUF, SOC
— Необходимые паттерны для сегодняшней мобильной разработки: Adapter, Memento, Observer, Strategy, Factory, Command, Composite, Iterator, Mediator, Proxy, Template Method, Singleton и где они приненимы на iOS

---

3 **Dependency Injection, SOA, слоистая архитектура. Protocol Oriented Programming (POP)**

— Inversion of Control паттерн
— ServiceLocator и инжектинг
— Разделение архитектуры на слои и что это дает
– Protocol Oriented Programming (POP)
– Type constraints

Домашние задания

1 Реализация Service Locator и Dependency Injection и изолированности слоев в приложении

---

4 **Связывание разных частей приложения Observing, Signals, Callbacks. PATs (Protocol with Associated Types)**

— Observing и broadcasting, нужен ли нам KVO
— Плюсы и минусы Delegation, виды делегатов
— Callbacks
— Signals ans Slots и причем здесь Rx
– PATs (Protocol with Associated Types)

3 Foundation без сторонних фреймвоков и Swift 5 Standard Library

1 **Необычная система типов Swift, структуры данных, Generics**

— Теория типов и Compound и Named типы
— Метатип и вложенные типы
— Protocol Composition
— Generics
— Создание кастомных структур данных

---

2 **Sequences и коллекции, асимптотический анализ: O(1), O(N), O(N*log(N)), O(n^2)**

— Sequence и IteratorProtocol
— Type-erased типы: AnySequence, AnyIterator, AnyCollection
— Lazy Wrappers
— Wrappers for Algorithms
— Асимпотический анализ встроенных и кастомных структур данных

Домашние задания

1 Реализация механизма тестирования алгоритмов на производительность и отображения результатов

Цель: Вы научитесь создавать кастомные структуры данных на основе протоколов Sequence и IteratorProtocol кастомные структуры данных и решать с помощью них реальные задачи в приложениях

1. Создать SuffixSequence
1.1 Как показано в уроке создать SuffixIterator
1.2 Обернуть в SuffixSequence каждое слово из AlgoProvider
2. Собрать профилирование структуры данных Suffix Array
2.1 Склеить все SuffixSequence в единый массив с элементами кортежами типа (suffix, algoName)
2.2 Отсортировать этот массив по алфавиту

2.3 Создать вью контроллер по типу Array/Set/Dictionary
2.4 Сделать 1-3 теста на прог всех имен аглоритмов
2.4.1 Используя StringGenerator делать поиск по 10 случайный трехбуквенным сочетаниям(подстрокам)
2.4.2 Сделать режим теста с отладкой которая будет показывать сколько раз находить подстроки
*3. Вставить UISearchController на таблицу фида
3.1 Организовать такой же поиск по FeedDataProvider
3.2 Выводить в серч контроллер результаты поиска

---

3 **Использование всей мощи String, Literals vs. UnicodeScalar, UTF-16**

— Сравнение суффиксов и префиксов и другие способы сравнения строк
— Работа с utf8 и utf16 представлениями
— Ипользование подстрок и Ranges, StringProtocol
— Regex

Домашние задания

1 Продвинутая локализация приложения на несколько языков

---

4 **Региональные форматы и локализация iOS приложения**

— Парсинг и представление телефонных номеров
— Форматирование дат согласно региону и локали, POSIX спецификация
— Работа с единицами измерения и валютами
— Корректная локализация приложения на несколько языков и регионов

Домашние задания

1 Создание ячейки с поддержкой региональной локализации

---

5 **Ассоциативные типы, Type Erasure, «сахарные» типы данных, диспетчеризация вызовов в Swift 5**

— 3 типа диспатчеризации в Swift: direct, dynamic, message
— Associated Types
— PATs и динамическая диспатчеризация
— Другие способы реализации паттерна Type Erasure
— Как работают типы в SIL (Swift Intermediate Language)

Домашние задания

1 Создание 3х реализаций расчета и отображения и вывод результатов на диаграмму

---

6 **Компилятор LLVM, AST, создание собственных операторов**

— Как работает LLVM: SIL, IR
— Как некоторые типы представлены в SIL и для чего это нужно знать
— Особенности и хитрости компиляции
— Перезагрузка и создание операторов

Домашние задания

1 Проектирование и реализация своей собственной структуры данных и оценка ее эффективности

4 Многозадачность

1 **Проблемы многозадачности и способы их решения, GCD**

— Антипаттерны и проблемы: Priority Inversion, Race condition, Deadlock, Resource contention, Starvation, Non-deterministic and Fairness
— Использование GCD: QoS, Queues, Main Queue и Main Thread

Домашние задания

1. Реализация асинхронного выполнения задач и оценка эффективности подхода

---

2 **Внутренности GCD(libdispatch), OperationQueue**

— Плюсы и минусы OperationQueue
— Внутренности libdispatch: пул тредов, continuation, QoS и как зная это лучше использовать очереди

---

3 **RunLoop & POSIX Threads, Инструменты синхронизации, Lock, Mutex**

— RunLoop и чем сегодня он может нам быть полезен
— pthreads
— Виды локов: NSLock, NSRecursiveLock, Spinlock, Mutex, Semaphore
— Dispatch Barriers
— Trampoline техника

Домашние задания

1. Реализация общей конкурентной очереди на приложение, создание внутренней системы тасков

5 Networking и хранение данных

1 **Новый Network-фреймвок, URLSession, Codable**

— Network феймвок, HTTP, REST, Sockets, GraphQL
— URLSession
— Сериализация и десериализация с помощью Codable

---

2 **Когда использовать Files, чистый SQLite, способы кеширования**

— Виды кеширования
— SQLite и другие DB* альтернативы
— NoSQL
— Files и File System

Домашние задания

1 Имплементация сериализации и сохранения данных на backend

---

3 **CoreData — основные стратегии использования**

— NSManagedObject, NSPersistentStoreCoordinator, NSManagedObjectContext
— NSPersistentContainer
— Модель данных
— CRUD на Core Data

---

4 **Realm**

— Плюсы и минусы Realm
— Модель данных на Realm
— Realm Browser
— CRUD на Realm

Домашние задания

1 Реализация поддержки оффлайн режима в приложении

6 Создание приложений для watchOS, tvOS, перенос приложений с помощью Mac Catalyst

1	**watchOS**	— Отличия SwiftUI для watchOS — Создание приложения для Apple Watch — Интеграция с iOS приложением
2	**tvOS**	— Отличия SwiftUI для tvOS — Создание приложения для Apple TV
3	**iPadOS, Mac Catalyst, перенос приложения на macOS**	— Нюансы SwiftUI для iPadOS/macOS — Перенос приложения на macOS

7 Мультиплатформенная разработка, кодогенерация, перенос приложения на Android

1 **Мета-программирование на чистый Swift и Kotlin и внедрение кодогенерации**

— Основы метапрограммирования
— OpenAPI
— Sourcery

---

2 **Виды нативной мультиплатформы: Pure Swift + Pure Kotlin, общее ядро на Kotlin Multiplatform**

— Android Studio & IntelliJ IDEA
— Обзор Kotlin Multiplatform и целесообразность общего ядра
— Разработка Pure Swift + Pure Kotlin

Домашние задания

1 Реализация каркаса приложения используя декларативный подход и Rx

---

3 **Одновременная реализация фич на iOS + Android. Необходимый tool-set**

— Как Swift разработчику писать на Kotlin
— SwiftUI + Jetpack Compose
— Combine + RxAndroid/RxKotlin/RxJava

Домашние задания

1 Создание UI слоя приложения на Rx

8 Организация разработки

1 **Тестирование кода XCTest, UITest, fastlane и CI**

— Test-Driven Development (TDD)
— Теория тестирования
— Использование XCTest
— Зачем нужен UITest
— Сборка CI (Continuous Integration) на fastlane

---

2 **Git-flow, TBD, автоматизизация workflow**

— Продвинутое использование git
— Фича, хотфикс, релизный цикл, master ветка
— Использование комманд git-flow
— Trunk Based Development и ветки для вич

Домашние задания

1 Покрытие приложения юнит-тестами

---

3 **Как правильно написать резюме и развивать hard-skills**

— Почему важно корректно писать резюме
— Как выбирать работодателя чтобы развивать свой hard-skills
— Какие бывают работодатели
— Какие скрытые критерии отбора используются

Домашние задания

1 Написание работающего резюме

9 Проектная работа

1	**Написание приложения с нуля**	— Выбор темы для приложения — Как генерировать идеи для простых приложений на основе известных «болей» пользователей — Как использовать iOS платформу для генерации идей для приложений — Подбор инструментов, помощь с стартом написания приложения Домашние задания 1 Написание приложения с нуля
2	**Консультация по проекту**	
3	**Защита проекта**	Домашние задания 1 Сдать ссылку на репозиторий курсового проекта. В репозитории обязательно должен быть заполнен файл Readme.md с описание проекта.