OTUS

Специальная цена

iOS Разработчик. Продвинутый курс v 2.0.

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

Длительность курса: 168 академических часов

1 SwiftUI и основы Combine

1	**Проектируем UI декларативно с SwiftUI. В чем отличия UIKit и SwiftUI**	после занятия студент сможет: настроить окружение для работы на курсе и выполнения домашних заданий; использовать Xcode; создавать базовые интерфейсы на SwiftUI/Combine. 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**	после занятия студент сможет: понимать в каком состоянии приложение; добавлять логику на изменение состояния приложения; будет знать как устроен UIKit и его иерархия классов; делать навигацию в SwiftUI разными способами.

3	**Использование NavigationView, TabView. Создание собственного стека Навигации**	после занятия студент сможет: Использовать NavigationView и TabView в SwiftUI; Создавать собственный стек Навигации Использовать Transitions
4	**Отображение структурированных данных, List, пейджинг, кастомные компоненты на UIViewRepresentable**	после занятия студент сможет: правильно использовать List; реализовывать пейджинг на SwiftUI; кодогенерить Network слой в ДЗ; привязывать List к реальному API с помощью кодогенерации. Домашние задания 1 Реализация пейджинга на реальном API Цель: Список с пейджингом работающий на реальном серверном API 1. Используйте открытое API https://github.com/public-apis/public-apis 2. Сделайте несколько рубрик по разным запросам новостей или городов по погоде (переключение через горизонтальный ScrollView либо SegmentedControl) 3. При переключении рубрик должен изменять содержимое List, пейджинг должен работать 4. Сделать глубину в 3 экрана с помощью кастомного навигейшен-стека
5	**Создание кастомных Shape, SwiftUI Drawing and Animation API**	после занятия студент сможет: работать с CALayer и понимать систему координат используемую в CoreGraphics; программно рисовать Shapes в SwiftUI; использовать анимацию в SwiftUI.

2 Современная архитектура мобильных приложений

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

Домашние задания

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

Цель: Научится внедрять DI, понять плюсы подхода

1. Создать ServiceLocator (можно на дженериках)
2. Перевести существующие сервисы на него
3. Добавить инжектинг в переменные инстанса класса, чтобы в каждома классе можно было видеть зависимости не скролля файл
*4. Выделить уровень Core сервисов (сеть, парсинг, хранение)

---

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

Домашние задания

1 Создания каркаса для модуляризированного приложения

Цель: Прокачать умение делать архитектурный рефакторинг всего приложения.

1. Модуляризировать свое приложение одним из известным способом
1.1 Вынести UI компоненты в отдельный модуль и импортить его в местах использования
2. В основной части приложения сделать рефакторинг до архитектуры MVVM
2.1 Выделить ViewModel-s там где это возможно
2.2 Как это делается описано в статье на appcoda в материалах

3	**Необычная система типов Swift, структуры данных, Generics**	после занятия студент сможет: создать кастомные структуры данных.
4	**Современные паттерны проектирования, принцип SOLID и его целесообразное применение. Принципы GRASP**	
5	**Связывание разных частей приложения Observing, Signals, Callbacks. PATs (Protocol with Associated Types)**	

3 Foundation без сторонних фреймвоков и Swift 5 Standard Library

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

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

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

после занятия студент сможет:
работать с utf8 и utf16 представлениями;
использовать подстрок и Ranges, StringProtocol.

Домашние задания

1 Продвинутая локализация приложения на несколько языков

---

3 **Региональные форматы и локализация iOS приложения**

после занятия студент сможет:
работать с единицами измерения и валютами;

Домашние задания

1 Создание ячейки с поддержкой региональной локализации

Цель: Получить умение создавать App Extension. Научиться разрабатывать поддержку в приложении нескольких региональных локалей

1. Реализовать для приложения ShareExtension
1.1 Настроить чтобы с любого сайта можно было выделить текст и отправить в приложение
1.2 При полученииии ревеста от ShareExtension в приложении показывать ViewController
2. В этом ViewController:
2.1 Отображать Segmented Control переключения между локалями, например: английской(США), французской, китайской
2.2 Под ним Label с пошаренным текстом
2.3 Текст должен быть разобран существующие в нем даты и единицы измерений
3. При переключении сегментов в тексте поменять в тексте даты и единицы измерения на локализованные

---

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

Домашние задания

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

---

5 **Компилятор LLVM, AST, создание собственных 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**

после занятия студент сможет:
использовать GCD: QoS, Queues, Main Queue и Main Thread.

Домашние задания

1 Реализация асинхронного выполнения задач и оценка эффективности подхода

Цель: Научиться внедрять сервис очереди в существующую инфраструктуру приложение, развиваем навык рефакторинга для не-UI кода приложения

1. Реализовать структуру данных Job Queue
2. Создать сервис Job Scheduler
3. В хедере таблицы экрана Feed сделать возможность запускать один конкретный тест по всем струкутурам например только Create или Access
3.1 Предварительно лучше уточнить чтобы кол-во ранов теста везде было одинаково например 10_000
4. В ячейку выводить время теста (без этого принтом можно)
*5. Красить в зеленый лучшее время и в красный худшее, или градацие от зеленого к красному

---

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

---

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

Домашние задания

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

5 Networking и хранение данных

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

---

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

Домашние задания

1 Имплементация сериализации и сохранения данных на backend

Цель: Научится сохранять Codable структуры в файлы, реализовывать кэш

1. Проверить, что модели поддерживают протокол Codable в вашем каркасе приложения
1.1 Либо реализовать кеш другим способом (UserDefaults, NoSQL(CoachDBLite, PinCache, ...), Keychain :), Files, Core Data, Realm)
2. Реализовать сохранение в файл при возвращении из экрана тестирования структуры данных
3. Чтение сделать в момент запуска приложения или перезахода в экран тестирование структуры данных
4. Должно сохранятся предыдущее состояние тестирование даже при перезапуске приложения
5. Можно использовать библиотеку Disk

---

3 **CoreData — основные стратегии использования**

---

4 **Realm**

Домашние задания

1 Реализация поддержки оффлайн режима в приложении

Цель: Создать приложение, которое будет получать данные из сети.
1 Кэширование реализовать на Realm или CoreDara.
2 Полученными данными заполнить таблицу согласно архитектурному паттерну MVC или другому

Факт сдачи дз - 40 баллов
Сдача во время - 10 баллов
Наличие механизма сохранения - 25 баллов
Наличие отображения- 25 балов

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

1	**watchOS**	после зянятия студент сможет: создать приложение для Apple Watch.
2	**tvOS**	после занятия студент сможет: создать приложение для Apple TV.
3	**iPadOS, Mac Catalyst, перенос приложения на macOS**	

7 Мультиплатформенная разработка, кодогенерация, перенос приложения на Android

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

---

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

Домашние задания

1 Реализация каркаса приложения используя декларативный подход и Rx

---

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

Домашние задания

1 Создание UI слоя приложения на Rx

8 Организация разработки

1	**Тестирование кода XCTest, UITest, fastlane и CI**	после занятия студент сможет: собрать CI (Continuous Integration) на fastlane; использовать XCTest.
2	**Git-flow, TBD, автоматизизация workflow**	после занятия студент сможет: использовать комманд git-flow. Домашние задания 1 Покрытие приложения юнит-тестами
3	**Как правильно написать резюме и развивать hard-skills**	Домашние задания 1 Написание работающего резюме

9 Проектная работа

1	**Написание приложения с нуля**	выбрать и обсудить тему проектной работы; спланировать работу над проектом; ознакомиться с регламентом работы над проектом; генерировать идеи для простых приложений на основе известных «болей» пользователей; использовать iOS платформу для генерации идей для приложений. Домашние задания 1 Написание приложения с нуля
2	**Консультация по проектам и домашним заданиям**	получить ответы на вопросы по проекту, ДЗ и по курсу.
3	**Защита проектных работ**	защитить проект и получить рекомендации экспертов. Домашние задания 1 Сдать ссылку на репозиторий курсового проекта. В репозитории обязательно должен быть заполнен файл Readme.md с описание проекта.