iOS Разработчик. Продвинутый курс

Вся мощь Swift 5.x для развития профессиональных навыков уровня Middle/Senior iOS Developer

Длительность курса: 128 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	**SwiftUI List, List с кастомным лейаутом, Hosting**	после занятия студент сможет: понимать в каком состоянии приложение; добавлять логику на изменение состояния приложения;

	ViewControllers	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: – реализовывать пейджинг на SwiftUI; – кодогенерить Network слой в ДЗ; – привязывать List к реальному API с помощью кодогенерации.
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. Домашние задания 1 Реализация пейджинга на реальном API Цель: создать 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

2 Современная архитектура мобильных приложений

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

после занятия студент сможет:
создать кастомные структуры данных.

---

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

после занятия вы сможете:
работать с разными подходами связи сущностей в приложении: коллбеками , сигналами , классическим Observing.

Домашние задания

1 Создания каркаса для модуляризированного приложения с Dependency Injection

Цель: Прокачать умение делать архитектурный рефакторинг всего приложения.
Научится внедрять DI, понять плюсы подхода

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

---

3 **Protocol Oriented Programming (POP), SOA, модуляризация, архитектурные Rx паттерны: Flux/Redux**

после занятия вы сможете:
применять паттерн Inversion of Control;
использовать ServiceLocator;
объяснить в чем польза слоистой архитектуры.

---

4 **MVx, VIP, анализ архитектурных паттернов. SOLID, GRASP, Coupling/Cohesion принципы.**

после занятия вы сможете:
разобраться в семействе MV(x) паттернов;
объяснить, что такое модуляризация и как ей пользоваться;
использовать Clean Architecture подход
рассказать о паттернах в мобильной разработке

Домашние задания

1 Модуляризация MV(x) приложения с помощью SwiftPM

Цель: В этом домашнем задании закрепляем изученный материал по архитектуре и модуляризации приложения с помощью SwiftPM.
Вы можете выбрать для реализации любой понравившийся паттерн MV(x).

1. Создайте небольшое приложение, которое с помощью сервиса обращается к API из списка https://github.com/public-apis/public-apis и выводит данные в список.
1.1. При реализации бизнес-логику нужно отделять от UI. Используйте изученные на прошлых занятиях сведения по SOA.

2. Выберите понравившийся паттерн MV(x) и примените его к вашему приложению. Обратите внимание, что для SwiftUI нужны другие решения, чем для классических приложений iOS.

3. Выделите вашу бизнес-логику в Swift Package и подключите к своему приложению.

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)**	после занятия вы сможете: использовать Sequence и Collection для реализации собственных структур данных; объяснить как работают lazy collection, будете понимать концепцию type-erased.
2	**Использование всей мощи String: StringInterpolation, Expressible. Региональные форматы.**	после занятия студент сможет: работать с utf8 и utf16 представлениями; использовать подстрок и Ranges, StringProtocol. работать с единицами измерения и валютами;
3	**Ассоциативные типы, Type Erasure, «сахарные» типы данных, диспетчеризация вызовов в Swift 5**	после занятия вы сможете: объяснить, что такое Method Dispatch и почему это важно. создавать generic протоколы. объяснить как Swift работает с типами.
4	**Компилятор LLVM, AST, создание собственных операторов**	после занятия вы сможете: объяснить, как работает компилятор LLVM; создавать собственные операторы. Домашние задания 1 Создание расширения для копирования текста в приложения и построение на основе него суффиксного массива Цель: Получить умение создавать App Extension. 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. 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 View 2. Перед показом View разложить все полученные слова в тексте на SuffixSequence 2.1 Как показано в уроке создать SuffixIterator 2.2 Обернуть в SuffixSequence каждое слово из полученное из шаринга 3. В этом ViewController: 2.1 Отображать Segmented Control(Picker в SwiftUI) переключения между

2.1.1 листом всех суффиксов, повторяющиеся помечать кол-вом, остортировать по алфавиту, сделать переключение сортировки ASC/DESC
2.1.2 топом 10 самых популярных 3х буквенных суффиксов, отсортированных по кол-ву нахождений
2.1.3 топом 10 самых популярных 5х буквенных суффиксов, отсортированных по кол-ву нахождений
*4. Добавить поиск на лист 2.1.1
4.1 Искать по совпадением используя debounce в 500мс из Combine

4 Многозадачность

1	**Проблемы многозадачности и способы их решения, GCD**	после занятия студент сможет: использовать GCD: QoS, Queues, Main Queue и Main Thread.
2	**Внутренности GCD(libdispatch), OperationQueue**	после занятия вы сможете: рассказать о проблемах многозадачности; избегать антипаттерны; пользоваться средствами GCD.
3	**RunLoop & POSIX Threads, Инструменты синхронизации, Lock, Mutex**	после занятия вы сможете: разобраться как работает RunLoop; использовать инструменты синхронизации; разобраться с POSIX. 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. Сделать таб историю шарингов на основе предыдущего таьа 2. Реализовать структуру данных Job Queue 3. Создать сервис Job Scheduler 4. В хедере таблицы экрана Feed сделать возможность запускать один конкретный тест по всем айтемам истории на построение суффиксного массива 5. В ячейку выводить время построение *6. Красить в зеленый лучшее время и в красный худшее, или градацие от зеленого к красному

5 Networking и хранение данных

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

после занятия вы сможете:
попробовать разные способы формированияи работы с URL;
объяснить, как работать с чистой URLSession;
познакомиться с GraphQL и как его использовать на iOS.

---

2 **Использование Firebase Cloud Messaging (Мессенджеры, пуши и пр.)**

после занятия вы сможете:

---

3 **SQLite, способы кеширования, Files, Сравнение CoreData и Realm**

после занятия вы сможете:
разобрать строение файловой системы iOS;
познакомиться с некоторыми способами кэширования;
рассмотреть такие решения для сохранения данных, как SQLite, Realm, NoSQL, CoreData

---

4 **Аутентификация с помощью Firebase через Apple/e-mail/Facebook/Google/Twitter и т. д**

после занятия вы сможете:
Понимать как работает OAuth

Домашние задания

1 Реализация поддержки оффлайн режима в приложении

Цель: Научится сохранять Codable структуры в файлы, реализовывать кэш

Создать приложение, которое будет получать данные из сети.
1 Кэширование реализовать на Realm/Files/Firebase/CoreData (лучше что еще не использовали)
2. Проверить, что модели поддерживают протокол Codable в вашем каркасе приложения
2.1 Либо реализовать кеш другим способом (UserDefaults, NoSQL(CoachDBLite, PinCache, ...), Keychain :), Files, Core Data, Realm)
3. Реализовать сохранение в файл при возвращении из экрана тестирования структуры данных
4. Чтение сделать в момент запуска приложения или перезахода в экран тестирование структуры данных
5. Должно сохранятся предыдущее состояние тестирование даже при перезапуске приложения

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	**Кросс-платформенный код для iOS/iPadOS, watchOS, macOS, tvOS**	после занятия вы сможете: использовать NavigationView и TabView в SwiftUI; делать компоненты с помощью @ViewBuilder; использовать Transitions и Animation; сСоздавать собственный стек Навигации. Домашние задания 1 Приложение работающее на всех платформах Apple Цель: Прокачать навык apple-кроссплатформенной разработки 1. Можно использовать шаблон https://github.com/shial4/SpriteKit-SwiftUI-Template или сделать свой для SwiftUI компонентов 2. Сделать меню игры с разделами 2.1 Play 2.2 Settings 2.3 Leaderboard 3. На каждом экране показывать как минимум заголовок и тестовые данные 4. Можно сделать какой-нибудь геймлей :) *4. Добавть watchOS

7 CoreML и Vision, нейронные сети и машинное обучение

1	**CoreML: работа с внешними моделями через API сервисов Google и Яндекс — это сильно не отличается от работы с обычным API**	разобраться как устроен CoreML 3+; использовать внешние обученные модели.
2	**CoreML: получение моделей с помощью AutoML Vision и использование их на устройстве**	объяснить как создавать обучать модели на AutoML и Vision.

8 Мультиплатформенная разработка: перенос на Android, Vulkan/Metal

1	**Мультиплатформа для Rich Media: Metal и Vulkan, разработки игр, Video/Image процессинг**	после занятия вы сможете: писать кроссплатформенный GPU код; настраивать окружение для этого.
2	**Jetpack Compose**	после занятия вы сможете: сравнить SwiftUI и Compose; переносить логику разработки на SwiftUI в Compose.
3	**Одновременная реализация фич на iOS + Android. Необходимый tool-set**	после занятия вы сможете: настраивать окружение; собирать KMM стек для iOS; решать проблемы с gradle. Домашние задания 1 Создание мультиплатформенного сетевого слоя с помощью KMM и openapi-generator Цель: Научится собирать мультиплатформу с кодогенерацией 1. Установить IDEA Community Edition https://www.jetbrains.com/idea/download 2. Создать мультиплатформенный проект как показано было на занятии 3. Пересоздать для SwiftUI iosApp как показано было на занятии 4. Обновить openapi-generator с помощью brew install openapi-generator 5. Можно взять из материалов recipepuppy_openapi.yaml (нужно в Info.plist разрешить http) или найти другое публично апи и написать спеку https://github.com/public-apis/public-apis 6. Сгенерировать сетевой слой openapi-generator generate -g kotlin -i recipepuppy_openapi.yaml --library multiplatform -o NetworkLayer (показывается на занятии) 7. Подключить в build.gradle io.ktor и kotlinx.serialization 8. Сбилдить KMM в IDEA и затем app фреймвок в Xcode 9. Вывести рецепты в SwiftUI лист по нескольким ингридиентам или поиском (частично показано в конце занятия)

9 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.
3	**Как правильно написать резюме и развивать hard-skills**	после занятия вы сможете: корректно писать резюме; выбирать работодателя, чтобы развивать свой hard-skills. Домашние задания 1 Написание работающего резюме Цель: Создать/доработать резюме, чтобы получать больше откликов и оно меньше вызывало вопросов со стороны рекрутеров и работодателей 1. Посмотреть запись занятия с разбором резюме 2. Доработать свое 3. Прислать ссылкой или файлом

10 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 Цель: Создание проектной работы, которая будет указана в сертификате выбрать тему/название для приложения; закрепить тему приложения в чат с преподавателем; обсудить тонкости; сделать проектную работу
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. Цель: Предложить оговоренную/одобренную ранее тему через строку в окне ниже. Сдать ссылку на репозиторий курсового проекта. В репозитории обязательно должен быть заполнен файл Readme.md с описание проекта.