OTUS

iOS Разработчик. Продвинутый курс

Вся сила и мощь Swift 5.1. Прокачай свои знания и навыки до уровня middle/senior.

Длительность курса: 176 академических часов

1 Эффективный UIKit и CoreGraphics

1 **Проектируем UI без кода и зачем это нужно, storyboard, xib. Как устроен UIKit изнутри**

– Решение организационных вопросов.
– Настройка environment: Xcode, git, scripts
– Возможности UIKit в создании интерфейсов без кода и зачем это нужно

Домашние задания

1 Создание каркаса приложения

Цель: Студент 1. будет целостно понимать стек UIKit/UIStoryboard 2. получит умение сборки иерархии экранов в Storyboard 3. научится создавать полноценные кастомные компоненты 4. получит умение использовать Storyboard Reference

Создать флоу экранов в Storyboard без

кода
1. Добавить UISplitViewController
1.1 В UISplitViewController добавить как master TabBarViewController
1.1.1 В TabBarViewController добавить 3 таба: Feed, Benchmark, Profile
1.1.2 На каждый таб добавить по NavigationViewController
1.1.3 В каждом NavigationViewController добавить по UIViewController
1.1.4 В каждом UIViewController добавить 1-2 из наследников UIView (UILabel, UIImageView, UISwitch, UIButton)
1.1.5 В UIViewController таба Profile добавить кастомную UIButton с @IBInspectable/@IBDesignable
1.2 В UISplitViewController добавить как detail NavigationViewController
1.2.1 Создать и прилинковать NavigationViewController
1.2.2 В NavigationViewController добавить от UIViewController переход по Storyboard Reference на второй Storyboard
1.2.3 Сделать добавить xib-компонент заглушку
1.2.4 На втором Storyboard во ViewController Scene добавить xib-компонент
2. Протестировать на iPad/iPhone симуляторах, со сменой ориентации девайса

---

2 **Preservation и Restoration техники, стек UINavigationController, Life cycles**

— Life cycle приложения
— UIResponder и UIViewController Life cycles
— ViewControllerLifecycleBehaviour
— Preservation и Restoration состояния
— Модификация стека UINavigationController

Домашние задания

1 Добавление в прототипа приложения

Restoration

Цель: – Практически изучить App Life Cycle для использования это в сохранении и восстановлении данных – Научится использовать View controller lifecycle behaviors для того чтобы контролировать экран на каждой стадии – Научимся модифицировать NavigationController стек, для нелинейных переходов в приложениии

1 В AppDelegate реализовать методы App Life Cycle и расставить в них print с названиями стадий или функций
1.1 Для стадии запуска(Launching) приложения:
1.1.1 application: willFinishLaunchingWithOptions
1.1.2 application: didFinishLaunchingWithOptions
1.1.3 applicationWillEnterForeground
1.1.4 applicationDidBecomeActive
1.2 Для стадии выключения(Terminating) приложения:
1.2.1 applicationWillResignActive
1.2.2 applicationDidEnterBackground
1.2.3 applicationWillTerminate
2 Реализовать View controller lifecycle behaviors для вью контоллера Benchmark
2.1 Добавить поведение afterAppearing с запуском таймера:
timer = Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector(runTimed), userInfo: nil, repeats: true)
В runTimed сделайте принт, например так print(Date())
2.2 На beforeDisappearing вызвать таймеру invalidate() чтобы на других экранах он не тикал и не принтил
3 На ViewController таба Profile
3.1 написать поведение со сменой цвета статус бара и view.backgroundColor на

		черный 4 На ViewController(первом) таба Feed сделать кнопку 4.1 По кнопке пушить новый ViewController(второй) 4.2 На новом ViewController добавить еще одну кнопку и по ней показать третий ViewController 4.3 Реализовать переход с третьего ViewController на первый
3	**Изучаем внутренности и хитрости UIScrollView, продвинутое использование компонента**	– Устройство UIScollView и что такое контейнеры в UIKit – Взаимосвязь между contentSize, contentOffset, contentInset – Static UIScrollView – Бесконечный скроллинг – AutoLayout и UIScollView
4	**Отображение структурированных данных UICollectionView, UITableView, UIStackView**	– Использование Flow Layout – Комбинирование UICollectionView с PageControl – UITableView Sections – Оптимизация рендеринга таблицы – UIStackView и Autolayout – Решение проблем использования UIStackView, Compression Resistance и Hugging 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 программрно 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	**SizeClasses, Anchors и анимирование Auto Layout**	– SizeClasses, использование UITraitCollection – Продвинутая адаптивность на iOS без костылей – Safe Area – Layout Anchors – Анимирование Constraints
6		

6

Создание сверхсложных интерфейсов с помощью CoreGraphis и CoreAnimation

– Работа с CALayer, различия с UIKit
– CGContext
– Программное рисование с помощью UIBezierPath
– CAAnimation и CAAnimationGroup

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 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

---

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

3 Современная архитектура мобильных приложений

1 **Современные паттерны проектирования, принцип SOLID и его целесообразное применение**

— 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

---

2 **MVP, MVCS, MVVM, архитектурные паттерны, модуляризация, Clean Architecture**

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

Домашние задания

1 Создания каркаса для модуляризированного приложения

---

3 **Protocol Oriented Programming (POP)**

— Про Наследовании и ООП
— Миксины/трейты: Protocol Extensions
— Type Constraints
— Плюсы и минусы подхода

---

4 **Dependency Injection, SOA, слоистая архитектура**

— Inversion of Control паттерн
— ServiceLocator и инжектинг
— Разделение архитектуры на слои и что это дает

Домашние задания

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

---

5 **Связывание разных частей приложения Observing, Signals, Callbacks**

— Observing и broadcasting, нужен ли нам KVO
— Плюсы и минусы Delegation, виды делегатов
— Callbacks
— Signals ans Slots и причем здесь Rx

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 Целесообразный Rx

1 **Отличие императивного и реактивного подхода, концепция Observable**

— Чем реактивный подход может быть лучше
— Observable Sequences
— RxSwift 4.x
— Marble Diagrams

---

2 **Promises, Signals и как это в Rx. Функциональное программирование**

— Rx подход без RxSwift: PromiseKit, Signals
— Идея распространение изменений
— RxSwift Transformations: Map, FlatMap, Scan, Buffer

Домашние задания

1 Реализация каркаса приложения используя декларативный подход и Rx

---

3 **Основные паттерны: Composition, Aggregation, Cancellatiion**

— Rx паттерны
— Фильтрация: Filter, DistinctUntilChanged
— Комбинирование: StartWith, Merge, Zip

---

4 **UI Patterns: Driver & Action, быстрое создание полноценного UI на Rx**

— MVVM и RxSwift
— RxCocoa
— Создание UI на Rx

Домашние задания

1 Создание UI слоя приложения на Rx

7 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, автоматизизация workflow**

— Продвинутое использование git
— Фича, хотфикс, релизный цикл, master ветка
— Использование комманд git-flow

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

8 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.