KEEPCODING
Tech School

Programa formativo a medida | Full Stack Mobile/Android| Openbank

23 may 2023

Objetivos

Dar formación a perfiles juniors sin conocimientos previos en programación.

Duración

Distribución, fechas y calendario definidos en documento “calendario” previamente acordado por KeepCoding y Openbank.

Fecha

Arranque 4 de julio.

Modalidad

Online Live: Programa de manera Online en directo.

KEEPCODING
Tech School

Resumen Programa:

Track Común
Programación 101
Mobile 101
Git & Github
Modelado SQL
Agile
Patrones de diseño y Clean
Diseño y UX
Bitrise CI/CD.
Proyecto final

Track IOS
Lenguaje Swift
Fundamentos de IOS
IOS Avanzado
IOS : SwiftUI y Combine
Accesibilidad
Testing. Unit tests

Track ANDROID
Lenguaje Kotlin
Fundamentos de Android
Android Avanzado
Android: Compose y Flow
Accesibilidad
Testing. Unit tests

KEEPCODING
Tech School

Detalle de cada módulo

Track Común

Programación 101

1. Qué es programar
2. Abstracción de variable
3. Abstracción de función
4. Estrategia de evaluación
5. Booleanos
6. Tipos de funciones
    a. Funciones totales
    b. Funciones parciales
    c. Tipos en Python y gestión de errores
7. Secuencias
    a. Explícitas
    b. Implícitas
    c. Iteración for y while
8. Bucles infinitos y bucle de eventos
9. Abstracción de datos
10. Funciones de alto nivel
11. Funciones de alto nivel II
12. Bucles anidados, listas de listas, árboles y matrices
13. Map / Reduce
14. ETUDE: Histograma de un texto largo.
15. Estructuras de datos
    a. Funciones hash
    b. Set
    c. Dict
    d. Colas y Pilas
16. Funciones memorizadas
    a. Cambiar tiempo por espacio
    b. Casos reales
17. Map, reduce y filter de Python
18. ETUDE : “Suavizar” una matriz.
19. Programación basada en objetos

a. Objetos sencillos sin jerarquías
b. ¿Textual para “ver” objetos, como botones?

20. ETUDE: Un tipo Vector

KEEPCODING
Tech School

Modelado de datos y SQL

Parte I – Modelado:

- Modelo entidad-relación
  - Entidades
  - Atributos y tipos de datos
  - ¿Qué es una relación?
- Relaciones entre entidades
- Cardinalidad
  - Dinámica de grupo. Primer diagrama ER
- Normalización
  - Primera, Segunda y Tercera Forma Normal
  - Desnormalización
  - Dinámica de grupo. Mejorando el diagrama ER
- Break

Parte II – Lenguaje SQL:

- Conectar a base de datos externa o SQLite (dependiendo disponibilidad).
- SQL
  - DDL (Data Definition Language)
- Creación de Esquema de base de datos
- Creación de tablas
- Modificación de tablas
- Manipulación de columnas
- Creación de relaciones (probar tipos de borrado)
- Creación de índices.
  - DML (Data Manipulation Language)
- Extracción de datos con Select
- Filtrados con Where
- Agregación con Group By
- Ordenación con Order By
- Funciones de agrupación (SUM, MAX, COUNT)
  - Filtrado con Having
- Concatenación de consultas con Union
- Unión de tablas con Join
  - Tipos de Join

    - Producto Cartesiano
- Creación de vistas (aunque es DDL)
- Inserción de registros con Insert
- Actualización con Update
- Actualizar con valores procedentes de otras tablas con Join

Mobile 101

Objetivo: **Este módulo dará una visión "Big Picture" de todo lo que implica el desarrollo de Apps. Es el mapa del territorio que se va a explorar a lo largo de los próximos meses**
Conceptos que se verán:

- Herramientas de hardware que vas a necesitar: tipo de ordenador (un Mac), dispositivos móviles, etc.
- Lenguajes de programación que se usan para el desarrollo de Apps.
- Herramientas profesionales.
- Arquitecturas comunes, como MVC, MVP, MVVM o Viper.
- Ventajas y desventajas de Apps nativas frente a multiplataforma.
- Cómo enfrentarte a un bootcamp, contado por quién ya lo ha hecho con éxito:
    - Cómo seguir las clases
    - Cómo organizar tu día a día
    - Cómo sobrevivir a base de sangre, sudor, código, cafeína y pizza rancia los próximos meses de tu vida.

Git & GitHub

Objetivo: Git es una herramienta esencial para cualquier desarrollador, tanto si trabaja solo como en equipo. Se usará para todas las prácticas y para acceder a todo material no audiovisual del bootcamp

Agile

Objetivo del módulo: Aprender a gestionar y planificar el trabajo en torno a un proyecto específico, con el fin de alcanzar los objetivos y los deadlines establecidos.

- Introducción y conceptos básicos
- Principales metodologías y procesos
- Gestión de personas y liderazgo
- Organización de los tiempos y entregas
- Fases principales del proyecto
    - Objetivo
    - Problema-Solución
    - Prototipado

    - Test
    - Lanzamiento
    - Resultados
    - Revisión
    - Implementación
- Herramientas (Lucid, Sketch, etc.)
- Glosario de conceptos clave y bibliografía

Patrones de diseño y Clean

- ¿Qué son los patrones de diseño y para qué se usan?
- MVVM. Inyección dependencias
- COORDINATOR dentro de MVVM
- ¿Cómo se elige uno u otro?. Qué nos encontramos en las empresas
- Crear una app sencilla con MVVM.
- Clean Architecture
- Solid

Diseño y UX

- Color, tipografías, espacios, guías puntales de diseño de un app
- UX
- Android guidelines
- IOS guidelines
- Formatos de imágenes
- Casos reales trabajos reales del profesor

Bitrise CI/CD

1. Introducción a CI/CD
2. Conceptos clave de CI/CD
3. Herramientas y tecnologías de CI/CD
4. Introducción a Bitrise
5. Flujo de trabajo (Workflows) en Bitrise
6. Integración y pruebas en Bitrise
7. Despliegue y distribución con Bitrise
8. Buenas prácticas con Bitrise (Crear Workflows, variables de entorno para información sensible, gestionar y cachear dependencias etc.)

KEEPCODING
Tech School

Track iOS

Lenguaje Swift

Swift Básico

- Crear proyecto tipo Playgrounds y cómo funciona
- Tipos de datos (variables /Constantes), inferencia de tipos, tipo por valor y referencia, datos no vacíos y type alias.
- Operadores (asignación, aritméticos, comparativos,lógicos, valores aleatorios)
- Controles de Flujo (Ámbitos, If/else, switch, bucles for in bucles while y repeat)
- Opcionales
- Cadenas (interpolación, concatenación, contar y buscar subcadenas, Unicode y emojis en cadenas, métodos de ayuda en uso de cadenas)
- Colecciones
    - Arrays
    - Inicializando, creando, añadiendo, eliminando 7 elementos, búsqueda enumerando, métodos auxiliares de trabajo con arrays), arrays de más de una dimensión. Programación funcional con reduce,map, forEach etc.
    - Diccionarios (que son, creando, inicializando, acceso a los datos, añadiendo, modificado, eliminando, iterando sobre diccionarios, array de un diccionario)
    - Sets o conjuntos(que son, inicialización, trabajando con ellos, suma y resta de conjuntos
- Conversión de tipos (upcasting, Downcasting) y verificación del tipo
- Tuplas (concepto, descomponer tuplas, arrays de tuplas, asignaciones de variables a través de tuplas

Swift Intermedio

- Funciones
    - Introducción a las funciones
    - Parámetros de entrada (normales y por defecto)
    - Parámetro de salida
    - Parámetros de entrada y salida
    - Devolver más de un valor
    - Polimorfismo
- Enumeraciones
- Clases y herencia
    - Introducción a programación orientada a objetos
    - Concepto de clase
    - Crear una clase héroe
    - Inicializadores normales y de conveniencia
    - Herencia de clases, sobre escritura
    - Clases finales
    - Destructor de la clase

    - Inicializadores falibles
- Estructuras
    - Concepto, inicializadores
    - Mutando propiedades
    - Diferencias struct y class. Cuando usar una u otra.
    - Inicializadores falibles

Swift Avanzado

- Closures
    - Concepto de closure o función anónima.
    - Closure como parámetros en funciones
    - Closures de cierre
    - Reducción de los closures

    - Propiedades
        - Propiedades calculadas
        - Observadores de propiedad (didset / willset)
        - Propiedades perezosas (lazy)
    - Protocolos
        - Que es un protocolo y ejemplo de uso
        - Creamos protocolo
        - Mutating
        - Limitar el uso de un protocolo a clases: class
        - Elementos opcionales: @objc optional
        - Herencia de protocolos
    - Extensiones
        - Qué son y para qué valen
        - Extendemos una clase
        - Extensiones en protocolos
    - Control de errores
        - Qué es el control de errores
        - Do, try, Catch
        - Creando funciones throws
        - Resultados opcionales con try? Y try!
        - Enumeraciones de carga
        - Result type (tipo de resultado)

Fundamentos de iOS

- SandBox
- Xcode
- Autolayout
- El MVC

- Uso de XIB y Stoyboards en el mismo proyecto
- Delegados
- Testing (TDD) sencillo
- Codable y JSONDecoder.
- URLSession
- UserDefaults
- Consumo de web services REST de KeepCoding (codable,
- JSONDecoder).

Testing. Unit tests

1. Conceptos clave en Testing
2. Testing en el ecosistema iOS
    a. Lenguajes de programación (Swift, Objective-C)
    b. Herramientas y frameworks de pruebas (XCTest)
3. Testing en iOS
    a. Configurar un proyecto con pruebas unitarias en Xcode
    b. Crear y ejecutar pruebas con XCTest
    c. Entender el informe de pruebas y errores en Xcode

4. Escribir pruebas unitarias en iOS
    a. Estructura de una prueba unitaria
    b. Test doubles: Mocks, Stubs y Fakes en Swift
    c. Pruebas de casos exitosos y fallidos
    d. Pruebas de comportamiento y rendimiento

iOS Avanzado

- Repaso de closures y genéricos
- GCD,
- Notification Center
- Aplicar Clean (organización de carpetas y clases)
- KeyChain
- Testing y cobertura de código
- Uso de programación funcional de alto nivel (map, reduce etc)

iOS SwiftUI y Combine

- SwiftUI. Viendo todo tipo de controles y navegación.
- SwiftUI: Aplicaciones multiplataforma. Creamos una app simple para

- iPhone, iPad y Apple watch.
- Combine Programación reactiva usado en SwiftUI y UIKit
- Uso de async-await dentro del proyecto.
- Modelo MVVM con SwiftUI y sus properties wrappers
- Creación de una app usando async/await, combine, últimos controles
- de SwiftUI 3 etc. usando el api de Keepcoding.
- Testing y testing asíncrono.

Accesibilidad iOS

- Introducción a la accesibilidad
- Principios y conceptos clave de accesibilidad
- Accesibilidad en iOS
- VoiceOver en iOS
- Implementación de accesibilidad en iOS
- Mejorar la navegación y experiencia de usuario
- Accesibilidad en componentes específicos de iOS
- Pruebas de accesibilidad en iOS

Track Android

Lenguaje Kotlin

- JVM
- Tipado estático
- Inferencia de tipos
- Control de flujos
- Variables
- POO
- Programación Funcional
- Mutabilidad
- Tipos básicos
- Cadenas (interpolación)
- Clases, interfaces, clases abstractas, data class
- Enumeraciones
- Object
- Sealed class
- Opcionales
- Casting
- When

- Extensiones funciones
- Extensiones de propiedades
- Colecciones
- Programación funcional de nivel alto con colecciones (map, forEACH, flatMap)
- Bloques de ejecución en Kotlin (closures)
- Control de excepciones
- Corrutinas y asincronismo

Fundamentos de Android

- Android Studio
- Grade
- Android manifest
- Estructura proyecto
- Recursos (string, Color, Dimens, Styles etc). Design system Themes
- Actividades
- Contexto
- Intents
- Ciclo vida de la actividad
- Vistas en Android
- Listas
- ConstraintLayout
- Fragments y fragments manager
- Ciclo vida fragments
- LiveData para MVVM
- Persistencia local de valores como token JWT
- Inyección dependencias KOIN
- Uso de programación funcional de alto nivel (map, reduce etc.)
- Acceso de red y decodificación de modelos usando Retrofit (o similar) usando closures (no corrutinas)
- Navegaciones Jetpack
- Customs Views (componentes)

Testing. Unit tests

- Introducción al Desarrollo Guiado por Pruebas (TDD)
- Conceptos clave en TDD

- TDD en el ecosistema Android
    - Lenguajes de programación (Kotlin)
    - Conceptos básicos de JUnit5: anotaciones, assertions, runners y extensiones.)
- TDD en Android
    - Estructura de un proyecto Android con TDD.
    - Creación de pruebas unitarias para clases de dominio y utilidades de la aplicación.
    - Entender el informe de pruebas y errores en Xcode
    - Utilización de Mockito para crear objetos simulados y controlar el comportamiento de las dependencias.
- Implementación de TDD en Android
    - Escribir pruebas antes de la implementación (Red, Green, Refactor)
    - Desarrollar funcionalidades guiadas por pruebas
    - Refactorizar y mejorar el código sin romper las pruebas

Android Avanzado

- Modelo concurrencia, Corrutinas.
- Persistencia usando Room
- Permisos
- Clean y ordenación de carpetas
- Flavors
- Uso de programación funcional de alto nivel (map, reduce etc.)
- Testing por lo menos de los modelos. Testing asíncrono de una llamada de red.

Android: Compose y Flow

- Jetpack Compose
- Versión de Android Studio
    - Controles
    - Navegación
    - LiveData y MVVM con Compose.
- Flow
    - Hacer MVVM con Compose y Flow.
- Uso de Room para apps offline con Compose
- Testing con Android y asíncrono de llamadas de Flow.
- Cacheo de imágenes
- Testing de UI de Compose

KEEPCODING
Tech School

Accesibilidad Android

- Introducción a la accesibilidad en Android: qué es y por qué es importante.
- Conceptos básicos de accesibilidad: tipos de discapacidad y herramientas de apoyo.
- Diseño accesible de la interfaz de usuario: principios y directrices de Material Design.
- Creación de contenido accesible: etiquetado de imágenes, texto alternativo y descripciones.
- Uso de recursos accesibles: herramientas de accesibilidad de Android y ajustes de sistema.
- Desarrollo de aplicaciones accesibles: compatibilidad con TalkBack y otros lectores de pantalla.
- Optimización de la navegación y el control: uso de teclas de acceso rápido, gestos y comandos de voz.
- Creación de pruebas de accesibilidad: evaluación de la accesibilidad y seguimiento de las mejoras.
- Ejemplos de buenas prácticas de accesibilidad en aplicaciones móviles.
- Estrategias para la implementación de accesibilidad en proyectos reales de Android.

Proyecto Final

Se deberá desarrollar, de manera individual, un proyecto completo con clientes nativos y backend. Se replicará por completo la experiencia de un proyecto real, con KeepCoding actuando como cliente y en Sprints de dos semanas. Se utilizarán varias APIs públicas (Marvel, etc) como ejemplo.