UVM GO! FORWARD

BOOTCAMP EN APP DEVELOPER

Híbrido

(224 horas)

Objetivo:

Aprender a desarrollar aplicaciones móviles en IOS y Android, así como sus interfaces de usuario sistemas operativos y arquitectura desde los fundamentos, así como la arquitectura de las aplicaciones móviles.

Dirigido a:

Abierto al público interesado en cambiar de carrera profesional o especializarse en temas de tecnología y altamente demandados por los empleadores.

Reconocimiento:

Al finalizar tu programa recibirás:

- **Diploma Digital UVM con validez curricular y tecnología Blockchain con código QR y de verificación.**
- **Certificado Internacional de Embiz Foundation.**
- **Certificado de competencias laborales DC-3 de la STPS.**

UVM GO! FORWARD

¿Por qué UVM?

60 años de experiencia académica, más de 150 programas educativos y más de 180 programas de excelencia a nivel nacional.

Adquieres conocimientos y habilidades esenciales aplicables de manera inmediata a tu actividad profesional.

Los profesores que imparten las Certificaciones y Diplomados siguen un modelo de enseñanza con ejemplos reales, pues cada uno de ellos es experto y reconocido en su campo.

Flexibilidad educativa que te permite estudiara tu ritmo, a cualquier hora y en cualquier lugar.

Los Diplomados y Certificaciones de UVM enriquecen tu CV y te posiciónan como el mejor candidato.

UVM GO! FORWARD

Al estudiar el programa podrás:

Conocer y usar la programación backend y APIs.

Realizar pruebas y depuración de aplicaciones.

Desarrollar aplicaciones así como su publicación y distribución.

MÓDULOS

01 Introducción al desarrollo de aplicaciones móviles.

1. Fundamentos de programación:
   a. Variables y tipos de datos:
      i. Declaración de variables
      ii. Tipos de datos primitivos (enteros, flotantes, booleanos, etc.)
      iii. Uso de constantes y variables mutables
   b. Estructuras de control:
      i. Condicionales (if/else, switch/case)
      ii. Bucles (for, while, do-while)
      iii. Uso de operadores lógicos y relacionales
   c. Funciones y modularidad:
      i. Declaración y llamada de funciones
      ii. Pasar argumentos y retornar valores
      iii. Encapsulación de código en funciones
2. Arquitectura de aplicaciones móviles:
   a. Patrón de diseño MVC (Modelo-Vista-Controlador):
      i. Roles y responsabilidades de cada componente
      ii. Comunicación entre los componentes del MVC
      iii. Beneficios del patrón MVC en el desarrollo de aplicaciones móviles
   b. Comunicación cliente-servidor:
      i. Protocolo HTTP y REST
      ii. Métodos HTTP (GET, POST, PUT, DELETE)
      iii. Manipulación de datos en formato JSON
   c. Almacenamiento de datos:
      i. Bases de datos locales (SQLite, Core Data)
      ii. Almacenamiento en la nube (Firebase, AWS)
      iii. Acceso y manipulación de datos en el almacenamiento
3. Introducción a los sistemas operativos móviles:
   a. Características principales de iOS:
      i. Estructura de una aplicación iOS
      ii. Ciclo de vida de una app en iOS
      iii. Herramientas de desarrollo (Xcode, Interface Builder)
   b. Características principales de Android:
      i. Estructura de una aplicación Android
      ii. Ciclo de vida de una app en Android
      iii. Herramientas de desarrollo (Android Studio, XML)

MÓDULOS

02 Diseño de interfaces de usuario.

1. Principios de diseño visual.
   a. Teoría del color:
      i. Armonía de colores
      ii. Psicología del color
      iii. Uso de paletas de colores
   b. Tipografía:
      i. Selección de fuentes adecuadas
      ii. Jerarquía de texto
      iii. Tamaño y espaciado de la tipografía
   c. Diseño de iconos y gráficos:
      i. Creación de iconos claros y comprensibles
      ii. Uso de imágenes y gráficos adecuados
      iii. Consideraciones de tamaño y resolución

2. Experiencia de usuario (UX).
   a. Diseño centrado en el usuario:
      i. Investigación de usuarios y análisis de necesidades
      ii. Creación de personas y escenarios de uso
      iii. Pruebas de usabilidad y retroalimentación del usuario
   b. Flujo de interacción:
      i. Diseño de wireframes y prototipos interactivos
      ii. Mapas de flujo de la aplicación
      iii. Retroalimentación y respuesta visual del usuario
   c. Diseño adaptable y accesibilidad:
      i. Diseño responsive para diferentes tamaños de pantalla
      ii. Consideraciones de accesibilidad para usuarios con discapacidades
      iii. Uso de estándares de accesibilidad y guías de diseño inclusivas

3. Herramientas de diseño de interfaces.
   a. Sketch:
      i. Creación de diseños de interfaces
      ii. Uso de símbolos y estilos compartidos
      iii. Exportación de assets y especificaciones de diseño
   b. Adobe XD:
      i. Diseño de experiencias interactivas
      ii. Creación de prototipos navegables
      iii. Diseño colaborativo y compartición de diseños
   c. Figma:
      i. Diseño de interfaces en la nube
      ii. Colaboración en tiempo real con equipos de diseño
      iii. Generación de especificaciones y assets para desarrolladores

MÓDULOS

03 Desarrollo de aplicaciones para iOS.

1. Introducción a Swift:
   a. Sintaxis avanzada:
      i. Optionals: manejo de valores nulos
      ii. Closures: funciones anónimas y su uso práctico
      iii. Control de flujo avanzado: guard, defer, switch avanzado
   b. Programación orientada a objetos en Swift:
      i. Clases, estructuras y enumeraciones
      ii. Herencia, polimorfismo y protocolos
      iii. Propiedades, métodos y propiedades computadas
   c. Gestión de memoria:
      i. ARC (Automatic Reference Counting)
      ii. Ciclos de referencia y captura fuerte.
2. Desarrollo de la interfaz de usuario:
   a. Interface Builder:
      i. Creación de interfaces visuales utilizando Storyboards
      ii. Conexión de elementos de interfaz a código utilizando IBOutlet y IBAction
      iii. Gestión de constraints y Auto Layout
   b. Auto Layout:
      i. Uso de constraints para definir las relaciones espaciales entre los elementos de interfaz
      ii. Uso de stacks (UIStackView) para simplificar la organización y adaptabilidad de la interfaz
      iii. Adaptabilidad a diferentes tamaños de pantalla y orientaciones
   c. Controladores de vista:
      - Uso de UIViewController y sus métodos del ciclo de vida
      - Uso de UITableView y UICollectionView para mostrar listas y grillas de datos
      - Uso de UINavigationController para la navegación entre pantallas
3. Uso de frameworks de iOS:
   - UIKit:
   - Uso de componentes de interfaz de usuario predefinidos (botones, etiquetas, campos de texto, etc.)
   - Gestión de eventos de toque y gestos táctiles
   - Uso de animaciones y transiciones de vista
   - Core Data:
   - Creación de modelos de datos utilizando el Editor de Modelo de Core Data
   - Configuración de Core Data stack (persistent container, managed object context)
   - Realización de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) utilizando Core Data
   - MapKit:
   - Integración de mapas y geolocalización en la aplicación
   - Mostrar marcadores, rutas y polígonos en el mapa
   - Uso de Core Location para acceder a la ubicación del usuario

MÓDULOS

04 Desarrollo de aplicaciones para Android:

- Introducción a Kotlin:
- Sintaxis avanzada:
- Null Safety: manejo de valores nulos de forma segura
- Extension Functions: agregar funcionalidad a clases existentes
- Data Classes: simplificar la creación de clases de datos
- Programación funcional en Kotlin:
- Lambdas: funciones anónimas y su uso en colecciones
- Higher-order Functions: funciones que toman otras funciones como argumentos o las devuelven como resultado
- Operaciones sobre colecciones: map, filter, reduce
- Nullable types y seguridad del tipo en Kotlin:
- Elvis Operator: manejo de valores nulos de manera concisa
- Safe Calls: llamadas seguras a métodos y propiedades de objetos que pueden ser nulos
- Smart Casts: conversión automática de tipos después de una comprobación de nulidad
- Creación de interfaces de usuario en Android:
- Uso de XML y herramientas de diseño visual:
- Definición de layouts utilizando XML
- Diseño de interfaces utilizando el editor de diseño visual
- Uso de atributos y estilos para personalizar la apariencia de los elementos de interfaz
- Diseño adaptable con ConstraintLayout:
- Uso de constraints para definir relaciones espaciales entre elementos
- Uso de directrices, cadenas y barreras para controlar la posición y tamaño de los elementos
- Diseño responsive para adaptarse a diferentes tamaños de pantalla y orientaciones
- Recursos y temas personalizados:
- Creación y uso de recursos: strings, dimensiones, colores, estilos
- Definición de temas personalizados para personalizar la apariencia de la aplicación
- Uso de frameworks de Android:
- RecyclerView:
- Uso de RecyclerView para mostrar listas y grillas eficientes
- Creación de adaptadores personalizados para manejar los datos en el RecyclerView
- Implementación de click listeners y eventos de desplazamiento en RecyclerView
- Room:
- Configuración de la biblioteca de persistencia Room
- Creación de entidades y definición de relaciones
- Uso de consultas y operaciones CRUD con Room DAO
- Servicios de Google Play:
- Integración de servicios de Google como autenticación, mapas y notificaciones push
- Uso de la API de Google Maps para mostrar mapas y ubicaciones
- Uso de Firebase Cloud Messaging para enviar y recibir notificaciones push

MÓDULOS

05 Programación backend y APIs:

• Introducción a Node.js:
• Configuración del entorno de desarrollo con Node.js:
• Instalación de Node.js y npm (Node Package Manager)
• Creación de un proyecto Node.js
• Manejo de dependencias con npm
• Creación de APIs RESTful:
• Diseño de endpoints y estructura de recursos:
• Definición de rutas y verbos HTTP (GET, POST, PUT, DELETE)
• Gestión de parámetros de consulta y rutas dinámicas
• Implementación de operaciones CRUD (Crear, Leer, Actualizar, Eliminar):
• Creación de controladores y modelos para interactuar con la base de datos
• Validación de datos de entrada y manejo de errores
• Uso de códigos de estado HTTP apropiados en las respuestas
• Consumo de APIs en aplicaciones móviles:
• Realización de solicitudes HTTP utilizando bibliotecas como Axios (JavaScript) o URLSession (Swift):
• Configuración de solicitudes con encabezados y datos
• Manejo de respuestas y errores
• Manipulación de respuestas en formato JSON:
• Extracción de datos de respuestas JSON
• Conversión de datos a objetos utilizables en la aplicación móvil
• Autenticación y autorización en el consumo de APIs:
• Uso de tokens de autenticación (JWT, OAuth)
• Protección de rutas y recursos con middleware de autorización

MÓDULOS

06 Pruebas y depuración de aplicaciones:

• Pruebas unitarias:
• Configuración de entornos de prueba y uso de frameworks como Jest (JavaScript) o XCTest (Swift):
• Instalación y configuración de frameworks de pruebas
• Escritura de casos de prueba y ejecución de pruebas unitarias
• Uso de aserciones para verificar resultados esperados
• Cobertura de código y pruebas de mutación:
• Medición de la cobertura de código con herramientas como Istanbul (JavaScript) o Xcode (Swift)
• Identificación y corrección de mutantes para mejorar la robustez de las pruebas
• Técnicas de mock y stub para aislar dependencias externas:
• Creación de objetos falsos para simular el comportamiento de dependencias externas
• Uso de bibliotecas como Sinon (JavaScript) o XCTest (Swift) para crear mocks y stubs
• Pruebas de interfaz de usuario:
• Uso de frameworks de pruebas de UI como Espresso (Android) o XCTest (iOS):
• Configuración y ejecución de pruebas de interfaz de usuario automatizadas
• Interacción con elementos de interfaz de usuario y verificación de resultados
• Automatización de pruebas de UI en diferentes escenarios:
• Uso de casos de prueba y datos de prueba para simular diferentes condiciones y flujos
• Pruebas de casos límite y errores de validación en formularios y entradas de usuario
• Pruebas de accesibilidad y rendimiento de la interfaz de usuario:
• Verificación de cumplimiento de pautas de accesibilidad (como WCAG 2.0)
• Medición de rendimiento y optimización de la interfaz de usuario para mejorar la experiencia del usuario
• Depuración de errores:
• Uso de herramientas de depuración como Xcode Debugger (iOS) o Android Studio Debugger (Android):
• Configuración de puntos de interrupción y seguimiento de la ejecución del código
• Inspección de variables y estado del programa durante la depuración
• Uso de registros y puntos de interrupción:
• Inserción de registros en puntos críticos del código para rastrear el flujo y los valores de las variables
• Uso de puntos de interrupción condicionales para detener la ejecución en situaciones específicas
• Análisis de crash reports y logs para identificar errores:
• Interpretación de informes de errores y excepciones para identificar la causa raíz
• Análisis de registros y seguimiento de eventos para rastrear el flujo y los errores en la aplicación

Beneficios de la modalidad

Clases en vivo, actividades interactivas y casos prácticos. Puedes interactuar con profesores y otros alumnos para tener una experiencia más enriquecedora.

Networking. Tienes la oportunidad de construir una red de contactos profesionales con otras personas que tienen intereses similares o se desempeñan en el mismo ámbito.

Estudia a tu ritmo. Consulta todas las sesiones grabadas en el horario que más te convenga.

*Aplica lo que aprendas de forma inmediata.

Nota: Si no asistes a las sesiones en vivo con el profesor en las fechas y horarios establecidos, tendrás 30 días naturales para ver completa la grabación de la clase en Teams® y realizar la actividad asignada para que acredites el módulo.

UVM GO! FORWARD

SÉ PARTE DE LA UVM

@uvmmx uvm @uvmmx uvm.mx