EDUCACIÓN CONTINUA UVM

APP DEVELOPER

HÍBRIDO

(9 meses)

EDUCACIÓN CONTINUA UVM

DIPLOMADO EJECUTIVO

¡Transforma tu perfil profesional y destaca en el mercado laboral!

- **Obtén tres insignias digitales que validan tus habilidades.**

- **Credenciales verificables con tecnología blockchain, que fortalecen tu perfil profesional.**
- **Accede a +20 programas en diversas áreas de estudio, diseñados para responder a las demandas actuales de los empleadores**

Reconocimiento Diplomado Ejecutivo

Obtén un Constancia digital UVM del programa con tecnología blockchain y código de verificación

Reconocimiento Curso IA

Adquiere nuevos conocimientos y desarrolla tus habilidades para el uso práctico y estratégico de las herramientas de IA en tu entorno profesional.

Constancia DC-3 por STPS

Documento avalado por la Secretaría de Trabajo y Previsión Social (STPS) que acredita las habilidades y competencias adquiridas.

EDUCACIÓN CONTINUA UVM

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** con **validez curricular** y **tecnología Blockchain** con código QR y de verificación.
  - Certificado Internacional de COLTEC.
  - Constancia DC3 por la STPS.
  - Insignia digital del programa.

Nota: El pago del curso incluye 1 oportunidad de examen de certificación, en caso de requerir oportunidades extra, el costo corre por cuenta del alumno.

EDUCACIÓN CONTINUA UVM

¿Por qué UVM?

Tenemos **más de 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** que puedes **aplicar de inmediato** en tu **actividad profesional**.

Los **profesores** que imparten las **Certificaciones y Diplomados** son **expertos reconocidos** en **sus campos**.

Tienes **flexibilidad educativa** que te permite **estudiar a tu ritmo**, a **cualquier hora** y en **cualquier lugar**.

Los **Diplomados y Certificaciones UVM** enriquecen tu **CV** y te posicionan como **el mejor candidato**.

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)

02 Diseño de interfaces de usuario UX | UI

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

03 Desarrollo de aplicaciones híbridas con React Native y Expo

1. Introducción a React Native y Expo
   i. Fundamentos de React Native
   ii. Comparación entre React Native y desarrollos nativos
   iii. Introducción a Expo y sus ventajas
2. Configuración del Entorno de Desarrollo
   i. Instalación de Node.js, React Native y Expo CLI
   ii. Configuración de un nuevo proyecto con Expo
   iii. Estructura básica de un proyecto en React Native
3. Componentes y Navegación en React Native
   i. Uso de componentes básicos (View, Text, Image, etc.)
   ii. Creación de componentes personalizados
   iii. Implementación de navegación con React Navigation
4. Estado y Ciclo de Vida en React Native
   i Manejo del estado con Hooks (useState, useEffect)
   i. Context API para gestión de estados globales
   iii. Patrones de diseño y buenas prácticas
5. Estilos y Diseño Responsive
   i. Uso de StyleSheet para estilizar componentes
   ii. Diseño responsive y adaptativo con Flexbox
   iii. Uso de Dimensiones y Media Queries
6. Integración con APIs y Backend
   i. Consumo de APIs RESTful con Fetch o Axios
   ii. Conexión con servicios de backend
   iii. Uso de Firebase para autenticación y almacenamiento
7. Expo SDK y APIs Nativas
   i. Uso del Expo SDK para acceso a funcionalidades del dispositivo
   ii. Geolocalización, cámara, notificaciones, entre otros
   iii. Manejo de permisos en iOS y Android
8. Pruebas y Depuración
   i. Herramientas de depuración para React Native
   ii. Pruebas unitarias y de integración
   iii. Uso de Expo Tools para pruebas en tiempo real
9. Publicación de Aplicaciones
   i. Proceso de build con Expo
   ii. Publicación en Google Play Store y Apple App Store
   iii. Actualizaciones OTA (Over-The-Air) con Expo

04 Desarrollo de aplicaciones para Android

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

05 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:
      i. Uso de UIViewController y sus métodos del ciclo de vida
      ii. Uso de UITableView y UICollectionView para mostrar listas y grillas de datos
      iii. Uso de UINavigationController para la navegación entre pantallas
3. Uso de frameworks de iOS:
   a. UIKit:
      i. Uso de componentes de interfaz de usuario predefinidos (botones, etiquetas, campos de texto, etc.)
      ii. Gestión de eventos de toque y gestos táctiles
      iii. Uso de animaciones y transiciones de vista
   b. Core Data:
      i. Creación de modelos de datos utilizando el Editor de Modelo de Core Data
      ii. Configuración de Core Data stack (persistent container, managed object context)
      iii. Realización de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) utilizando Core Data
   c. MapKit:
      i. Integración de mapas y geolocalización en la aplicación
      ii Mostrar marcadores, rutas y polígonos en el mapa
      Uso de Core Location para acceder a la ubicación del usuario

06 Programación backend y APIs

1. Introducción a Node.js:
   a. Configuración del entorno de desarrollo con Node.js:
      i. Instalación de Node.js y npm (Node Package Manager)
      ii. Creación de un proyecto Node.js
      iii. Manejo de dependencias con npm
2. Creación de APIs RESTful:
   a. Diseño de endpoints y estructura de recursos:
      i Definición de rutas y verbos HTTP (GET, POST, PUT, DELETE)
      ii. Gestión de parámetros de consulta y rutas dinámicas
   b. Implementación de operaciones CRUD (Crear, Leer, Actualizar, Eliminar):
      i. Creación de controladores y modelos para interactuar con la base de datos
      ii. Validación de datos de entrada y manejo de errores
      iii. Uso de códigos de estado HTTP apropiados en las respuestas
3. Consumo de APIs en aplicaciones móviles:
   a. Realización de solicitudes HTTP utilizando bibliotecas como Axios (JavaScript) o URLSession (Swift):
      i. Configuración de solicitudes con encabezados y datos
      ii. Manejo de respuestas y errores
   b. Manipulación de respuestas en formato JSON:
      i. Extracción de datos de respuestas JSON
      ii. Conversión de datos a objetos utilizables en la aplicación móvil
   c. Autenticación y autorización en el consumo de APIs:
      i. Uso de tokens de autenticación (JWT, OAuth)
      ii. Protección de rutas y recursos con middleware de autorización
4. Pruebas unitarias:
   a. Configuración de entornos de prueba y uso de frameworks como Jest (JavaScript) o XCTest (Swift):
      i. Instalación y configuración de frameworks de pruebas
      ii. Escritura de casos de prueba y ejecución de pruebas unitarias
      iii. Uso de aserciones para verificar resultados esperados
   b. Cobertura de código y pruebas de mutación:
      i. Medición de la cobertura de código con herramientas como Istanbul (JavaScript) o Xcode (Swift)
      ii. Identificación y corrección de mutantes para mejorar la robustez de las pruebas
   c. Técnicas de mock y stub para aislar dependencias externas:
      i. Creación de objetos falsos para simular el comportamiento de dependencias externas
      ii. Uso de bibliotecas como Sinon (JavaScript) o XCTest (Swift) para crear mocks y stubs
5. Pruebas de interfaz de usuario:
   a. Uso de frameworks de pruebas de UI como Espresso (Android) o XCTest (iOS):
      i. Configuración y ejecución de pruebas de interfaz de usuario automatizadas
      ii. Interacción con elementos de interfaz de usuario y verificación de resultados
   b. Automatización de pruebas de UI en diferentes escenarios:
      i. Uso de casos de prueba y datos de prueba para simular diferentes condiciones y flujos
      ii. Pruebas de casos límite y errores de validación en formularios y entradas de usuario

c. Pruebas de accesibilidad y rendimiento de la interfaz de usuario:
   i. Verificación de cumplimiento de pautas de accesibilidad (como WCAG 2.0)
   ii. Medición de rendimiento y optimización de la interfaz de usuario para mejorar la experiencia del usuario

6. Depuración de errores:
   a. Uso de herramientas de depuración como Xcode Debugger (iOS) o Android Studio Debugger (Android):
      i Configuración de puntos de interrupción y seguimiento de la ejecución del código
      ii. Inspección de variables y estado del programa durante la depuración
   b. Uso de registros y puntos de interrupción:
      i. Inserción de registros en puntos críticos del código para rastrear el flujo y los valores de las variables
      ii. Uso de puntos de interrupción condicionales para detener la ejecución en situaciones específicas
   c. Análisis de crash reports y logs para identificar errores:
      i. Interpretación de informes de errores y excepciones para identificar la causa raíz
      ii. Análisis de registros y seguimiento de eventos para rastrear el flujo y los errores en la aplicación

07 Publicación y distribución de aplicaciones, y entrega de proyectos

1. Requisitos de envío de la App Store de Apple y Google Play Store:
   a. Preparación de la aplicación para el envío:
      i. Iconos de la aplicación y pantallas de inicio
      ii. Capturas de pantalla y descripciones de la aplicación
      iii. Requisitos de tamaño de archivo y formatos de imagen
   b. Cumplimiento de las directrices de cada tienda:
      i. Políticas de contenido y restricciones
      ii. Uso de marcas registradas y derechos de autor
      iii. Requisitos de privacidad y seguridad de los datos del usuario
   c. Proceso de envío y revisión:
      i. Creación de una cuenta de desarrollador en cada tienda
      ii. Configuración de la información de la aplicación y envío de la solicitud
      iii. Tiempos de revisión y posibles acciones requeridas
2. Políticas de revisión y consideraciones de seguridad:
   a. Prácticas recomendadas para pasar la revisión de la tienda de aplicaciones:
      i. Diseño y experiencia de usuario de alta calidad
      ii. Cumplimiento de las políticas de contenido y restricciones
      iii. Calidad y rendimiento de la aplicación
   b. Consideraciones de seguridad y privacidad de los datos de los usuarios:
      i Manejo adecuado de datos confidenciales
      ii. Cumplimiento de regulaciones y estándares de seguridad
      iii. Implementación de medidas de seguridad, como cifrado y protección de datos
3. Estrategias de promoción de aplicaciones y análisis de rendimiento:
   a. Técnicas de ASO (App Store Optimization) para mejorar la visibilidad y descargas de la aplicación:
      i. Optimización de palabras clave y descripción de la aplicación
      ii. Obtención de reseñas y calificaciones positivas
      iii. Aprovechamiento de herramientas de ASO, como Sensor Tower o App Annie
   b. Monitoreo y análisis del rendimiento de la aplicación utilizando herramientas como Google Analytics o App Store Connect:
      i Seguimiento de métricas clave, como descargas, usuarios activos y retención
      ii. Análisis de comportamiento del usuario y flujo de la aplicación
      iii. Identificación de áreas de mejora y oportunidades de crecimiento
   c. Estrategias de marketing y promoción para aumentar la adquisición y retención de usuarios:
      i. Campañas publicitarias en redes sociales y plataformas de anuncios
      ii. Colaboraciones con influencers y embajadores de marca
      iii. Uso de estrategias de retención de usuarios, como programas de fidelidad y notificaciones push
   d. Entrega de proyecto.

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.

Asesoría y acompañamiento. Cuentas con un facilitador por módulo para guiarte durante tu curso.

Aplica lo que aprendas de forma inmediata.

Nota: En esta Certificación, las clases son obligatorias debido a las prácticas y actividades que tiene el participante.

EDUCACIÓN CONTINUA UVM

SÉ PARTE DE LA UVM

@uvmmx uvm @uvmmx uvm.mx